Microcontroladores PIC

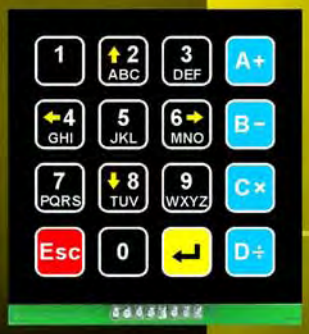
Programación en BASIC

16F62X
16F81X
16F87X

3ª Edición

Volumen 1

CARLOS A. REYES



Construya 4 nuevos grabadores de PIC´s

- Alarma para el hogar
- Control de accesos
- Manejo de módulos LCD
- Manejo de display de 7 seg.
- Control de motores DC y Paso a Paso
- Tacómetro digital
- Comunicación serial
- Generación de Tonos DTMF, etc.

Simulación y ruteado con PROTEUS

Fabricación de circuito impreso con el método de transferencia térmica

MICROCHIP

Incluye lámina para el grabador de PIC

Incluye CD ROM

Microcontroladores PIC 16F62X 16F81X 16F87X

Programación en BASIC

Tercera edición

CARLOS A. REYES

Ing. en Electrónica Digital y Telecomunicaciones
Tlgo. Electrónico en Automatización y Domótica
Profesor de microcontroladores PIC
Director Técnico de AUTOMASIS

Título original:
Microcontroladores PIC Programación en Basic
Tercera edición

Diseño de portada: Carlos A. Reyes
Diagramación: Carlos A. Reyes
Fotografía: Carlos A. Reyes

Pedidos de ejemplares, láminas de transferencia, placas PCB, asesoría en proyectos y tesis de grado a los telfs. 022 070 290 096 136 564
E-mail: automasis@yahoo.es
Http://www.automasis.es.tl

Impreso por: RISPERGRAF
Derechos del autor Nro. 020604
ISBN-10: 9978-45-004-1

ISBN-13: 978-9978-45-004-8

IMPRESO EN QUITO-ECUADOR

CONTENIDO

Prólogo ix
Introducción xi

Capítulo 1

SOFTWARES PARA EL FUNCIONAMIENTO DEL PIC

1 Descargas e instalaciones de los softwares 1
1.1 Descarga del programa gratuito microcode 1
1.2 Descarga del programador Ic-Prog y el drive NT/2000/XP.....5
1.3 Descarga del compilador PICBasic Pro 8

1.4 Instalación del software MicroCode Studio 9
1.5 Instalación del software programador Ic-prog 1.06A 10
1.6 Instalación del driver para Windows NT/2000/XP 12
1.7 Instalación de pbp247 (PicBasic Pro versión 2.47) 14
1.8 Instalación del compilador PicBasic Pro versión DEMO 14

Capítulo 2

EL MICROCONTROLADOR PIC

2 ¿Qué es un microcontrolador? 17
2.1 El microcontrolador PIC16F628A 17
2.2 Arquitectura del PIC6F628A 18
2.3 La memoria de programa 19
2.4 La memoria de datos 20
2.5 Características generales 22
2.6 Diagrama de pines y funciones 22
2.7 Consideraciones básicas pero muy útiles a la hora de montar un proyecto 23

Capítulo 3

EL PROGRAMA MicroCode Studio

3.1 Configuración de MicroCode Studio (IDE) 25
3.2 Manejo de MicroCode Studio 28
3.3 Identificación de errores en la compilación 31

Capítulo 4

PROGRAMANDO EN LENGUAJE BASIC

4.1 Diferencias entre el lenguaje Basic y ensamblador 33
4.2 Aprendiendo a programar el Pic 16F628A con microcode 35
4.3 Grabando el PIC con el IC-prog 1.06A 36
4.4 Diferentes caminos a seguir para conseguir un mismo objetivo 39
4.5 Declaraciones disponibles en el compilador pbp 2.47 41

Capítulo 5

PROYECTOS CON MICROCONTROLADORES PIC

5 Proyectos de aplicación 43

5.1 PROYECTOS CON LEDS
5.1.1 Programa básico para hacer parpadear un led con intervalos de 1 segundo 44
5.1.2 Un semáforo de 2 intersecciones 45
5.1.3 Juego de luces para discoteca 47

5.2 PROYECTOS DE REPETICIONES
5.2.1 Ejercicio con la instrucción FOR NEXT 50
Las variables **BIT**, **BYTE** y **WORD** 50
5.2.2 Luces del auto fantástico (desplazamientos) 51
5.2.3 Proyectos propuestos con leds 52

5.3 PRÁCTICAS CON PULSADORES
5.3.1 Ejercicio con pulsadores 53
La declaración **IF ... THEN** 54
5.3.2 Contador binario con pulsador antirrebote 55
5.3.3 Led intermitente de velocidad variable 57
5.3.4 Utilizando el **MCLR** (reset externo) 59
5.3.5 Proyectos propuestos con pulsadores 61

5.4 PROYECTOS CON DISPLAYS
5.4.1 Manejo de un display de 7 segmentos con el CI. 7447 62
5.4.2 Un contador decimal de un dígito con el CI. 7447 y un pulsador 63
5.4.3 Manejo de un display de 7 segmentos sin el CI. 7447 64
La declaración **LOOKUP** 64
5.4.4 Manejo de 4 displays de 7 segmentos con el CI. 7447 65
5.4.5 Contador decimal de 4 dígitos con el CI. 7447 67
5.4.6 Manejo de 4 displays de 7 segmentos sin el CI. 7447 (Rotulación) 71
5.4.7 Manejo de 4 displays como rótulo en movimiento 73
5.4.8 Manejo de un display de 35 segmentos 74
5.4.9 Proyectos propuestos con displays 78

5.5 MÓDULOS LCD
5.5.1 Manejo de un módulo LCD 79
La declaración **LCDOUT** 79
5.5.2 Presentación de caracter por caracter en LCD 83
5.5.3 Desplazamiento de un texto en LCD 84
5.5.4 Contador de pulsos con LCD 84
La declaración **COUNT** 84

La palabra **DEC**, **HEX**, **BIN** 86
5.5.5 Tacómetro digital 86
5.5.6 Lectura de un potenciómetro con LCD 88
La declaración **POT** 88
5.5.7 Proyectos propuestos con LCD 89

5.6 SONIDO
5.6.1 Generación de Sonido 90
La declaración **FREQOUT** 90
5.6.2 Una sirena policial 91
La declaración **SOUND** 91
Utilizando un cristal de mayor velocidad 91
5.6.3 Generación de un timbre de teléfono celular 93
5.6.4 Llamada telefónica DTMF 94
La declaración **DTMFOUT** 94
5.6.5 Proyecto propuesto 97

5.7 PROYECTOS CON TECLADOS
5.7.1 Lectura de un teclado de 16 pulsadores con display de 7 segmentos 98
5.7.2 Cerradura electrónica con clave en memoria FLASH 101
5.7.3 Cerradura electrónica con clave en memoria RAM y cambio de clave 105
5.7.4 Cerradura electrónica con clave en memoria EEPROM y cambio de clave 109
La declaración **EEPROM**, **READ** y **WRITE** 110
5.7.5 Proyecto propuesto 114

5.8 PROYECTOS CON MOTORES
5.8.1 Manejo del PWM como variador de velocidad de un motor DC 115
5.8.2 Un conversor D/A con el CI. LM358 117
5.8.3 Los motores paso a paso bipolares y unipolares 118
5.8.4 Manejo de un motor paso a paso en secuencia WAVE DRIVE 121
5.8.5 Manejo de un motor paso a paso en secuencia FULL STEP 123
5.8.6 Manejo de un motor paso a paso en secuencia HALF STEP 125
5.8.7 Proyectos propuestos con motores 126

5.9 COMUNICACIÓN
5.9.1 ¿Qué es la comunicación serial? 127
5.9.2 Modos de transmisión de datos 127
5.9.2.1 Simplex 127
5.9.2.2 Half-duplex 127
5.9.2.3 Full-duplex 127
5.9.2.4 Full/full-duplex 128
5.9.3 Comunicación serial RS232 128
5.9.4 Comunicación serial Pic a PC 129
La declaración **SEROUT** 129
5.9.5 Comunicación serial PC a Pic 132
La declaración **SERIN** 133
5.9.6 Comunicación serial con el CI. MAX232 135
5.9.7 Comunicación serial PIC a PIC 137
5.9.8 Comunicación serial RS422/485 139
5.9.9 Comunicación serial PIC a PIC con la interfaz RS485 139
5.9.10 Comunicación serial de VISUAL BASIC Y PIC 142

5.9.11 Comunicación serial sincrónica I²C 152
5.9.12 Comunicación I²C con una memoria serial 24LC04B 152
La declaración **I2CWRITE** y **I2CREAD** 153
5.9.13 Comunicación I²C con el reloj calendario DS 1307 156
5.9.14 Proyectos propuestos de comunicación 159

5.10 INTERRUPCIONES
5.10.1 Utilizando la interrupción del puerto B.0 160
Las declaraciones **ON INTERRUPT, DISABLE, RESUME Y ENABLE** 160
5.10.2 Utilizando la interrupción del puerto B.4 al B.7 162
5.10.3 Reloj digital utilizando la interrupción del TMR0 162
5.10.4 Proyectos propuestos con interrupciones 165

5.11 CONVERSOR A/D
5.11.1 Conversor análogo digital del PIC16F81X 166
5.11.2 Conversor análogo digital del PIC16F87X 168
5.11.3 Termómetro digital con el PIC16F877A 171

5.12 UTILIZANDO EL PIC12F6XX.
5.12.1 Parpadeo de leds en el puerto GPIO 175
5.12.2 Proyectos propuestos con el conversor A/D 177

Capítulo 6
SIMULACIÓN Y RUTEADO CON PROTEUS
6.1 Simulación del led intermitente 179
6.2 Simulación de un LCD 2x16 182
6.3 Generación de PCB (Print Circuit Board) 183
6.4 Impresión del PCB (Tarjeta de circuito impreso) 186

Capítulo 7
MÉTODO DE FABRICACIÓN DE CIRCUITOS IMPRESOS
7.1 Diseño del circuito impreso por software 187
7.2 Impresión de las pistas y screen de los elementos 188
7.3 Preparación de la placa (Baquelita o Fibra de vidrio) 188
7.4 Transferencia térmica del papel hacia la lámina de cobre 190
7.5 Proceso de atacado (reducción) del cobre 191
7.6 Proceso de limpieza de la placa ya atacada con ácido 193
7.7 Transferencia térmica del screen de los elementos 194
7.8 Perforación de la placa 194
7.9 Soldadura de elementos 195
7.10 Chasís o caja para proyectos 203

Apéndices
Apéndice A Sitios web relacionados con este libro 211
Apéndice B Próxima entrega 211

PRÓLOGO

Los microcontroladores pasan muchas veces desapercibidos, trabajan incansablemente sin que nos demos cuenta las 24 horas del día, pueden estar hasta en nuestro bolsillo, algunos están tan cerca como dentro del cuerpo y otros tan lejos como en el planeta Marte. En la actualidad existe un promedio de 40 microcontroladores en cada hogar y esta cifra va en aumento, lo cierto es que cada vez dependemos más de estas pequeñas computadoras que hacen que nuestra vida sea más fácil.

Han pasado más de 30 años desde que los primeros microcontroladores hicieron su aparición, y ningún otro dispositivo ha sido tan versátil, o tiene la misma acogida, todo esto ha motivando a muchos autores a escribir más de este circuito integrado que sobre cualquier otro. Desde su inicio han evolucionado mucho, partiendo del microcontrolador 8048 de Intel, si lo comparamos con los modelos actuales veremos que ahora tienen capacidad de hasta un mega de memoria de programa, procesan señales digitales y manejan todos los periféricos disponibles en la actualidad: serial, paralelo, USB, I²C, one wire, X10, etc., ahora imaginemos todo lo que está por venir.

Para entender su importancia debemos analizar qué pasaría si dejan de funcionar por un minuto: no habría comunicación; centrales telefónicas, celulares y radios dejarían de funcionar, lo mismo sucedería con: computadoras, satélites y con ello el comercio marítimo entraría en caos, el tráfico aéreo estaría en peligro, los aviones no podrían volar sin sus instrumentos de navegación, el sistema electrónico de los vehículos fallaría, las fabricas paralizarían su producción, en los hospitales muchos equipos electrónicos quedarían inservibles, en cada hogar dejaría de funcionar los sistemas de seguridad y de incendio, ascensores, y electrodomésticos en general, en definitiva se paralizaría todo el mundo llegado a un colapso general.

Este libro trata específicamente sobre el microcontrolador **PIC®** de **Microchip Technology Inc**. ya que es el fabricante que lleva el liderazgo por su bajo costo, fácil programación y la gran disponibilidad de modelos a elegir según sea las necesidades.

La metodología de enseñanza será hacer el proyecto poniéndolo en funcionamiento, para luego dar explicaciones bien detalladas ya que también está orientada a principiantes, podría parecer muy obvio pero las explicaciones no están demás, yo mismo he tenido problemas con algunos libros al no poder pasar más allá del primer ejercicio, aunque el libro decía que era para principiantes, y sin contar que me cansaba leyendo tanta teoría. Por eso no pretendo causar desinterés en el lector con una *montaña de teoría*, porque para ello existen muchos libros donde se trata más a fondo sobre la estructura del microcontrolador, más bien intento entusiasmarles con proyectos de fácil aprendizaje y aplicaciones muy útiles en su hogar, empresa y por qué no para realizar proyectos importantes de automatización de fábricas, ya que se incluye circuitos de control y manejo de control computarizado con Visual Basic, al final de todos los capítulos se dará referencia de paginas web en donde se puede encontrar más información al respecto. Ya que

este libro fue realizado con el apoyo de prácticas reales, encontrarán la información y los consejos más importantes que puedo darles en base a las experiencias propias que he adquirido a la hora de montar mis proyectos, este libro es producto de muchos años de trabajo. El motivo más grande que me impulsó a escribir este libro fue cuando un día mientras realizaba una práctica con los microcontroladores PIC, mi hijo de 5 años me hizo las siguientes preguntas: ¿Qué es esto?, ¿Para qué sirve?, ¿Cómo funciona?, etc. En ese momento me puse a pensar cómo podría enseñarle todo lo que sé y con su temprana edad cómo podría entenderlo. Una vez un profesor de la primaria me dijo: "El objetivo del hombre en la vida era, sembrar un árbol tener un hijo y escribir un libro", el árbol significa el trabajo y la contribución para el futuro del planeta; el hijo significaba la experiencia de ser padres y la continuación de la especie humana; el libro es nuestra experiencia y lo que nosotros aprendemos a lo largo de nuestro ciclo de vida y lo resumimos para nuestros hijos, para que así puedan continuar con nuestro trabajo y avancen mucho más que nosotros. Algunas reflexiones de mi padre sobre la muerte, también me hicieron pensar en la importancia de dejar mis conocimientos para los que al igual que yo se apasionan con este tema.

Adicionalmente para ayudar al lector se incluye en este libro una lámina para transferencia térmica del grabador de PIC´S y un CD con todos los ejercicios, diagramas, hojas de dato y fotografías a color de las placas que a lo largo del Capítulo 5 y 7 se van presentando, aquí podrán observar con total claridad cada una de las placas electrónicas (PCB), así como también podrán observar una secuencia fotográfica de cómo hacer placas de circuito impreso con el método de transferencia térmica. *Para utilizar el CD, únicamente copie todo el contenido del CD en una sola carpeta de su disco duro.*

Finalmente quiero expresar mi más sincero agradecimiento a Microchip por su valiosa ayuda tanto en información como en softwares de libre distribución, a mecanique por su excelente editor de textos que se distribuye gratuitamente por internet, a Bonny Gijzen por su programa IC-Prog, a microEngineering Labs., a todos mis profesores y amigos que me han ayudado a despejar mis dudas, a mi esposa que me apoyó en la realización de este libro, y de manera especial al Sr. Ing. Juan Bernardo Tamariz y la señorita Julia León de **Corpoimpex** distribuidores exclusivos de **Microchip** para Ecuador, por facilitarme las hojas de datos que se incluye en el CD, y permitirme utilizar el logo de **Microchip** en este libro, además de softwares, materiales, información y facilidades que me han brindado.

Quito, Ecuador,
Abril del 2008

INTRODUCCIÓN

El Capítulo 1 se trata de la descarga e instalación de los softwares necesarios para la programación de los microcontroladores PIC, mediante ilustraciones gráficas se seguirá paso a paso la descarga de cada uno de los programas, todo esto para que el lector no pueda perderse.

En el Capítulo 2 se da una breve teoría del microcontrolador, específicamente del PIC16F628A, y sus características más sobresalientes, además se da consejos muy útiles para el correcto funcionamiento del microcontrolador PIC.

En el Capítulo 3 se enseña a configurar y programar en microcode, así como también a reconocer los tipos de errores que se pueden dar en la compilación del programa escrito.

En el Capítulo 4 se explica por qué escoger el compilador PicBasic Pro 2.47 y su comparación con el lenguaje acostumbrado el ensamblador, también formas diferentes de escribir los programas, y cómo grabar un PIC con en el software IC-prog, además se enseña a reconocer errores en la grabación del PIC y las declaraciones disponibles en el compilador pbp.

En el Capítulo 5 se inicia el aprendizaje con ejercicios básicos pero muy necesarios para poder comprender el funcionamiento del microcontrolador PIC. También cabe recalcar que es indispensable disponer de un grabador de PIC´S y así poder realizar cada uno de los 52 proyectos que se encuentran en este capítulo, también es necesario seguir en orden cada uno de los proyectos para poder entender más adelante, ya que cada vez serán más grandes pero no muy difíciles de entender, en esta edición se ha mejorado el proyecto 5.11.3 Termómetro digital con el PIC16F877A utilizando el conversor A/D a 10 bits.

El Capítulo 6 trata de la simulación y ruteado utilizando el programa PROTEUS, de igual manera se enseñará paso a paso cómo montar un circuito para simular y adicionalmente para su posterior ruteado, terminando con la impresión de pistas y screen de elementos.

En el Capítulo 7 se enseña a fabricar circuitos impresos, mediante un sencillo método: *la transferencia térmica*, aquí se indicará todos los pasos necesarios para la fabricación del grabador de PIC´S **UNIVERSAL PICmicro5**, para el cual se entrega conjuntamente con este libro una lámina para la transferencia térmica de las pistas y el screen de los elementos, además se enseñará técnicas de soldaduras y terminaremos con la fabricación de un chasís para sus proyectos, este le dará una buena presentación y acabado.

CAPÍTULO 1
SOFTWARES PARA EL FUNCIONAMIENTO DEL PIC

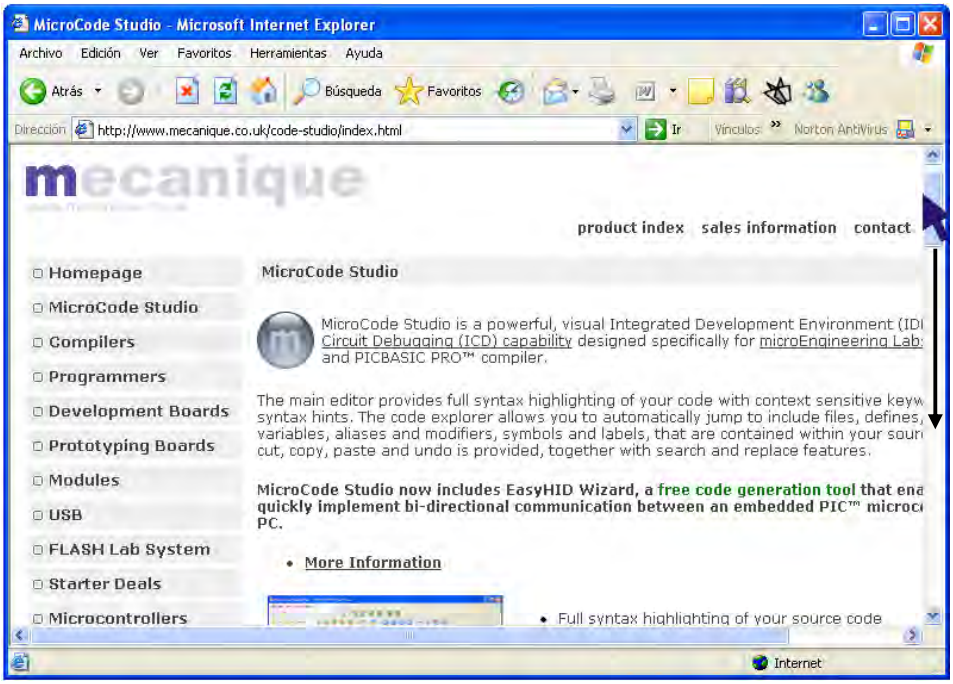
1. DESCARGAS E INSTALACIONES DE LOS SOFTWARES.

En este Capítulo se aprenderá a descargar los softwares necesarios para poder editar, compilar y programar los ejercicios prácticos que se incluye en este libro, se seguirá paso a paso cómo descargar del internet. Recuerde que hay actualizaciones cada 6 meses por lo que se recomienda visitar frecuentemente estas páginas, adicionalmente se incluye algunos de estos softwares libres en **CD:\Soft-Free**.

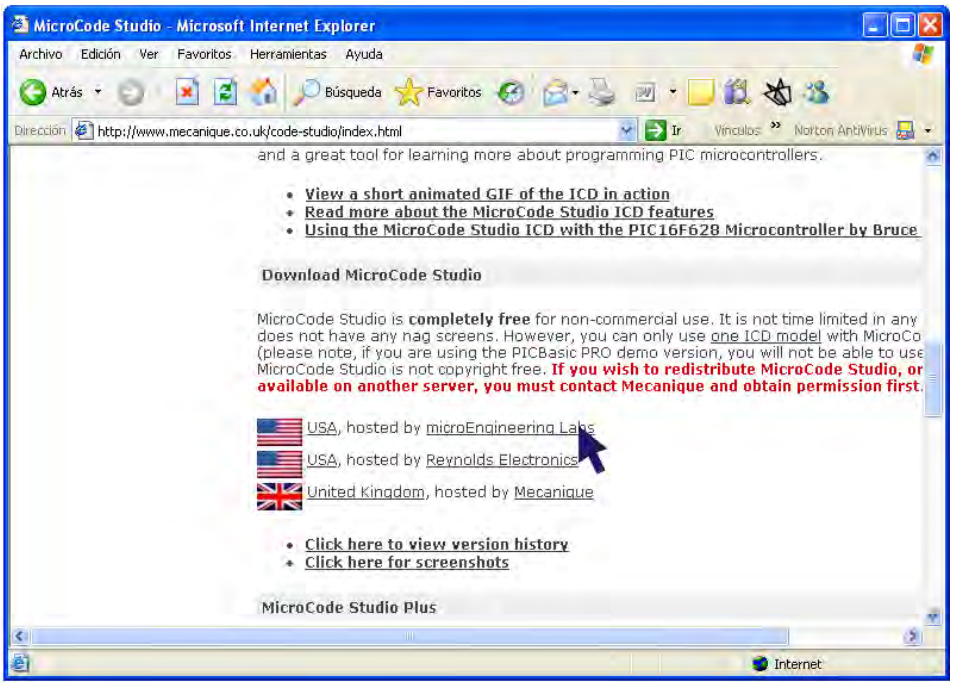
1.1 DESCARGA DEL PROGRAMA GRATUITO MICROCODE.

Este es el primer programa que debe descargar del internet, primero ingrese en la página www.mecanique.co.uk y luego realice los siguientes pasos:
1a) Una vez abierto la página web haga un clic en **Microcode Studio**, como ilustra la siguiente figura:

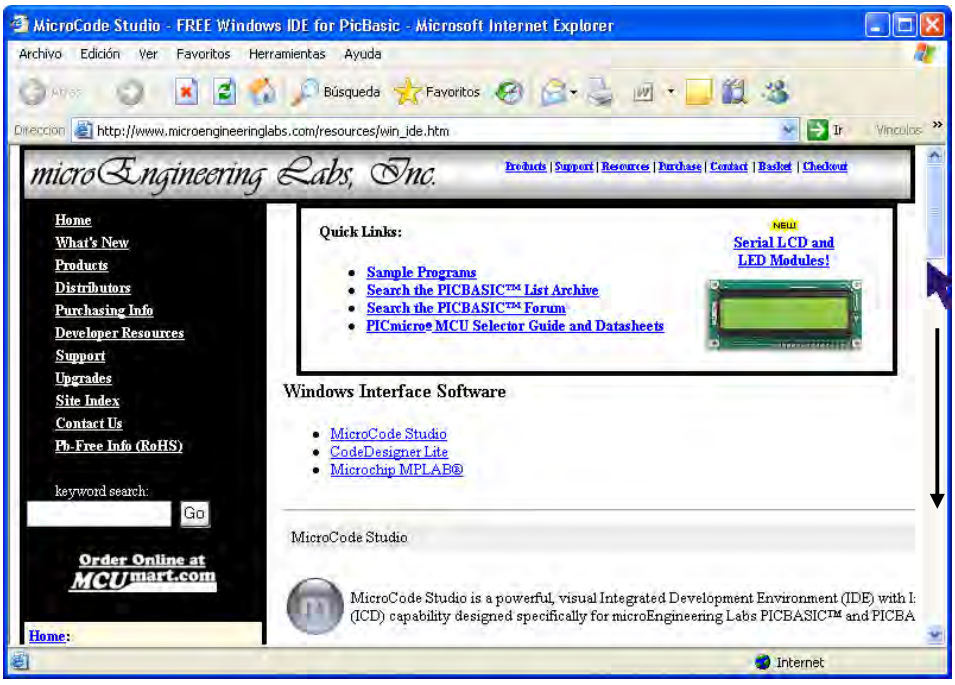
1b) Se presentará una nueva ventana, deslice la barra de desplazamiento que está a la derecha de la pantalla hasta encontrar **USA hosted by microEngineering Labs** literal 1c).



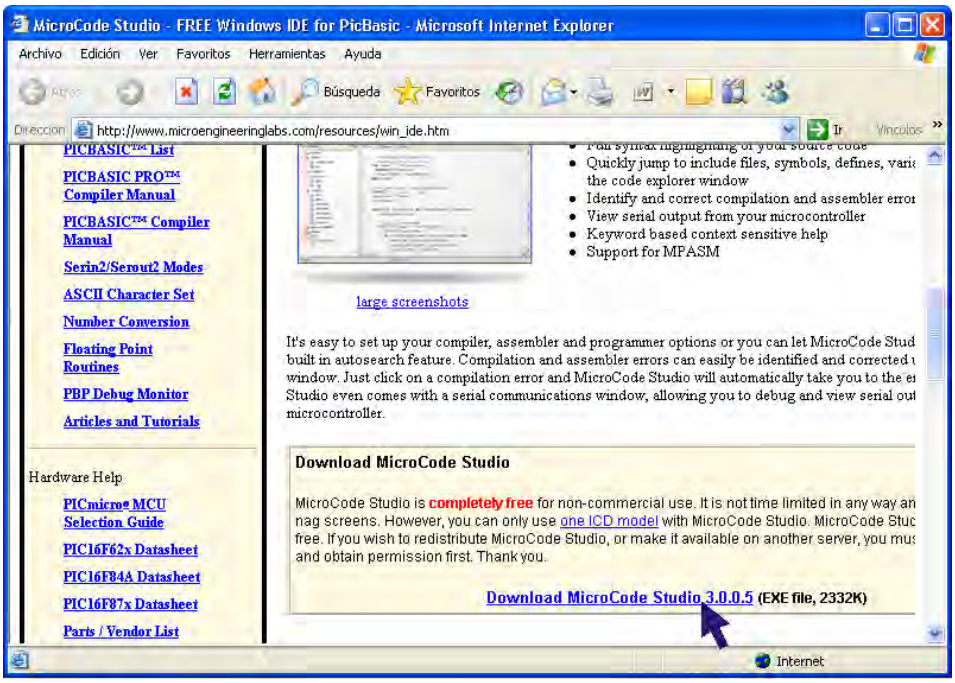
1c) En esta pantalla haga un clic en **USA hosted by microEngineering Labs**, o cualquiera de los otros servidores y espere un momento para pasar al siguiente literal.



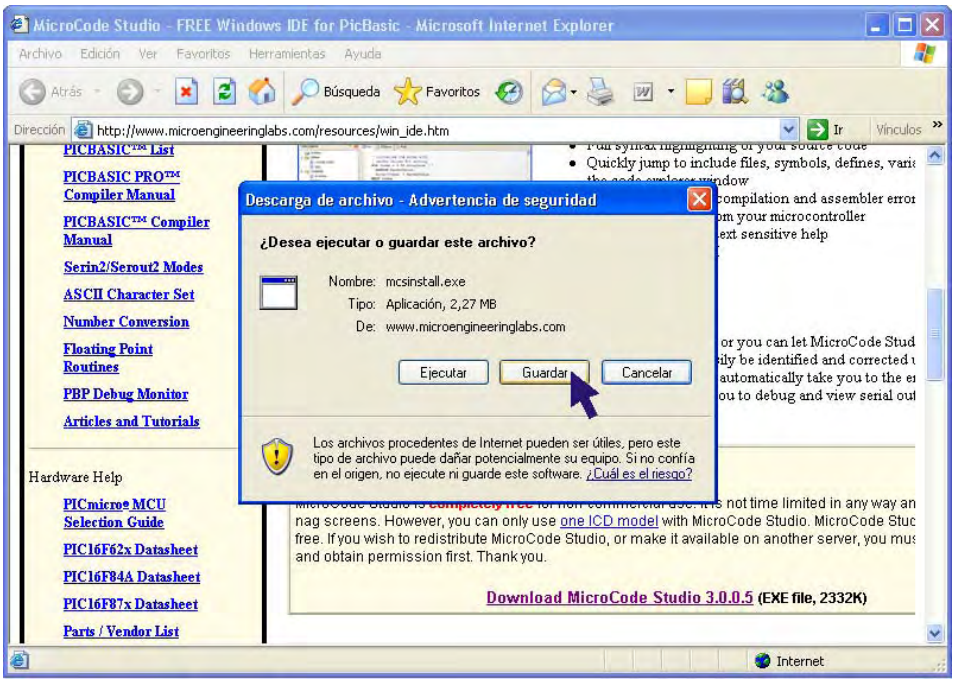
1d) En unos pocos segundos se abrirá una nueva ventana, www.microengineeringlabs.com, deslice la barra de desplazamiento hacia abajo hasta encontrar lo que viene en el literal 1e).



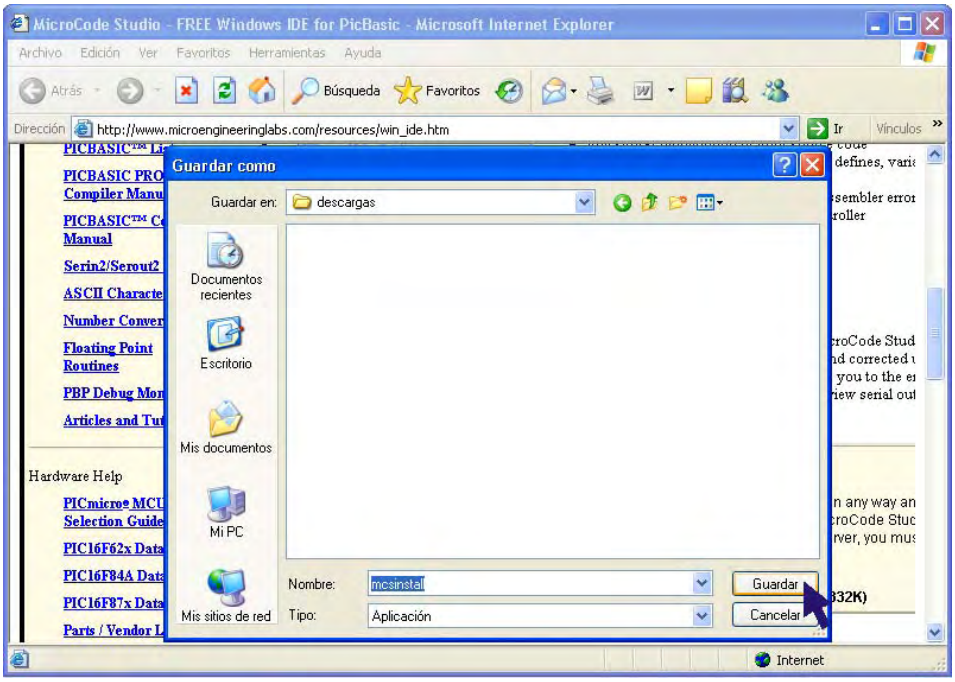
1e) En esta ocasión descargará el software Microcode Studio versión 3.0.0.5. Esta es la versión más reciente que corresponde al mes de abril del 2008, posteriormente podrá descargar de la misma manera las últimas versiones disponibles. Ahora bien proceda dando un clic encima de **Download MicroCode Studio 3.0.0.5**.



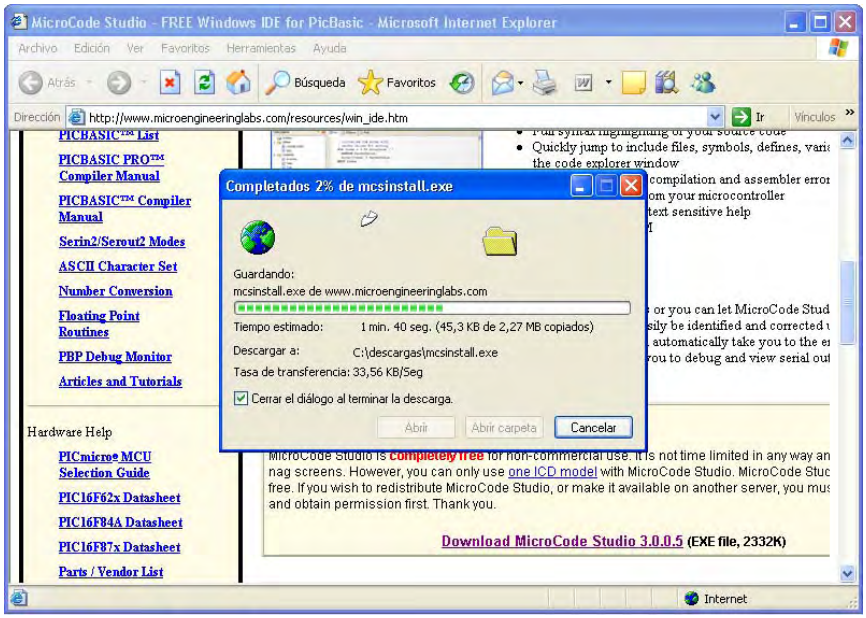
1f) Inmediatamente saldrá una pantalla de **Descarga de archivo**, tendrá el nombre de **mcsinstall.exe**, proceda dando un clic en **Guardar**.



1g) En instantes sale una nueva ventana similar al siguiente gráfico, proceda a guardar el archivo ejecutable en alguna carpeta que elija, en este caso lo haremos en la carpeta llamada descargas que fue previamente creada en C:. luego haga un clic en **Guardar**.



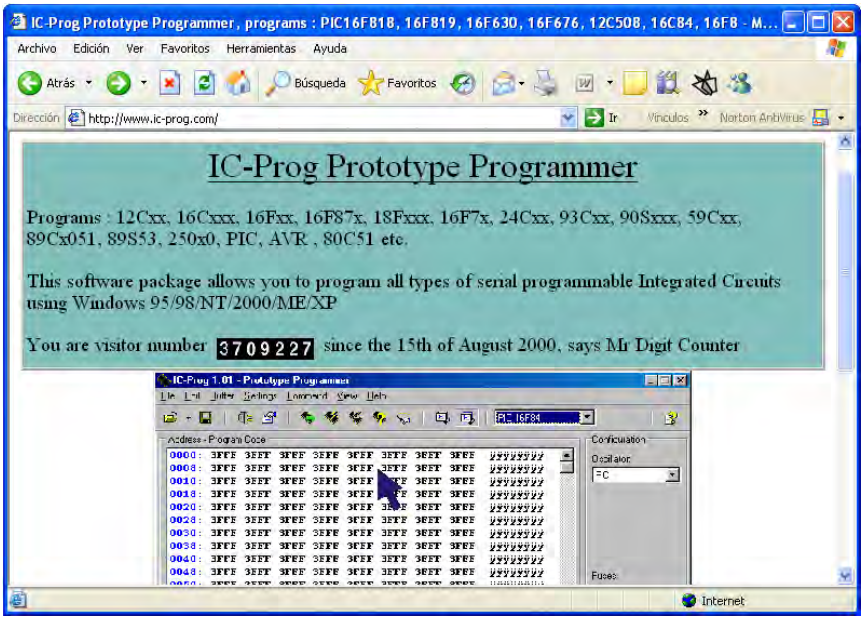
1h) Enseguida se presentará una pantalla de descarga, aquí puede marcar el recuadro de **Cerrar el diálogo al terminar la descarga,** para que se cierre automáticamente al finalizar y emita un sonido, esta descarga puede tardar unos 3 minutos, mientras lo hace puede seguir descargando el siguiente programa.



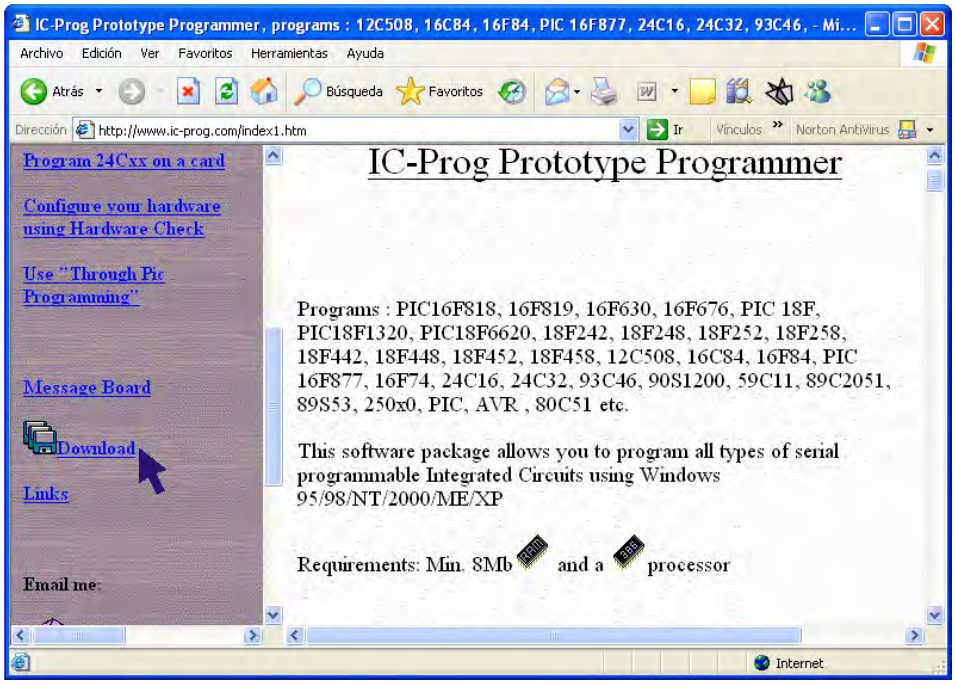
1.2 DESCARGA DEL PROGRAMADOR IC-Prog Y EL DRIVE NT/2000/XP.

2a) Ingrese a la página WWW.IC-prog.com, y haga un clic en el medio del gráfico.

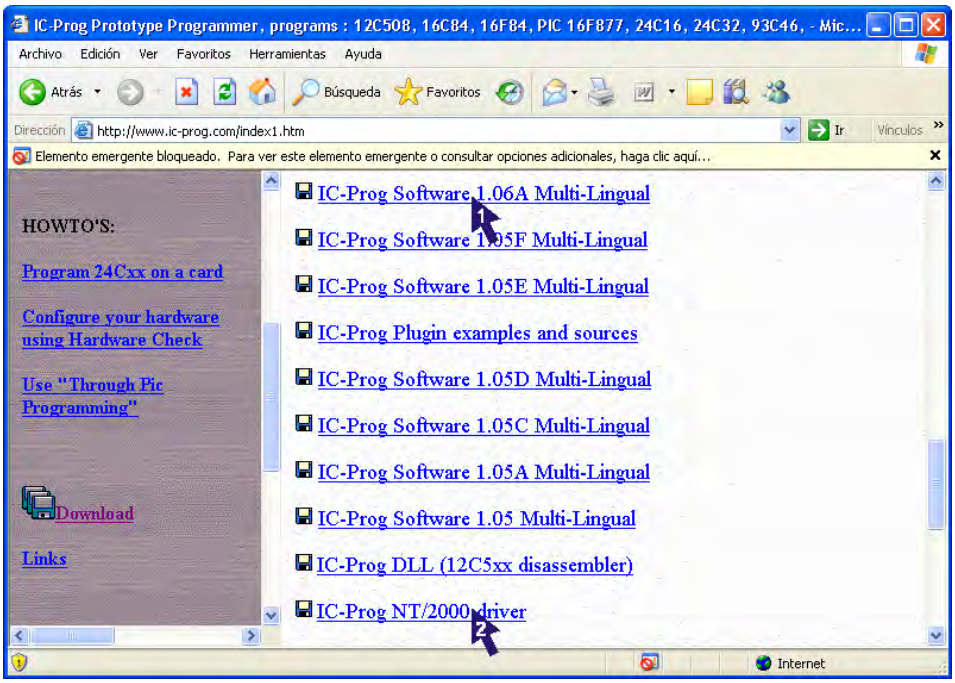
NOTA: este software se incluye en **CD:\Soft-Free** por cortesía de Bonny Gijzen.



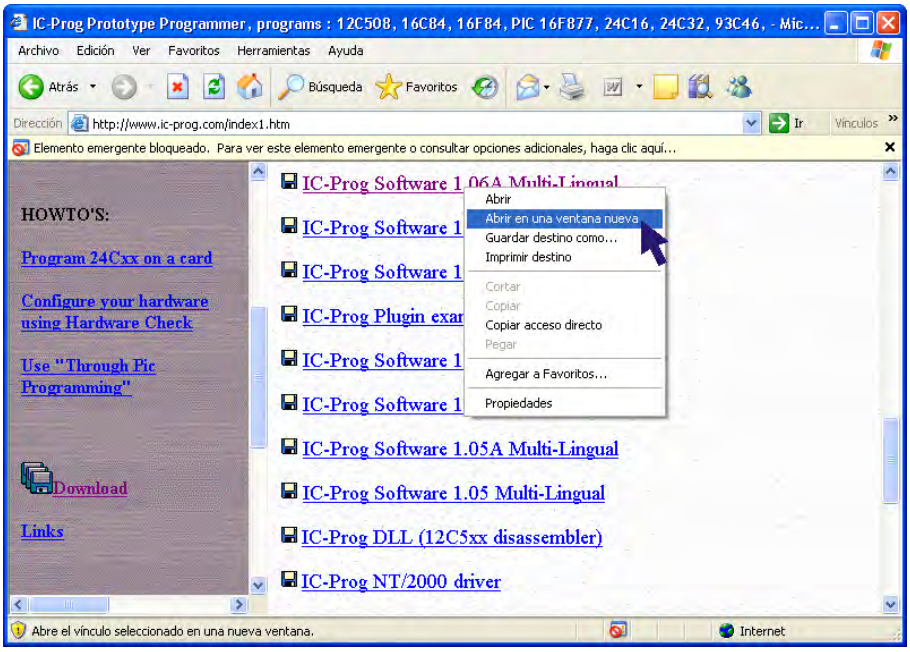
2b) Se presentará una ventana similar al siguiente gráfico. Proceda dando un clic en **Download**.



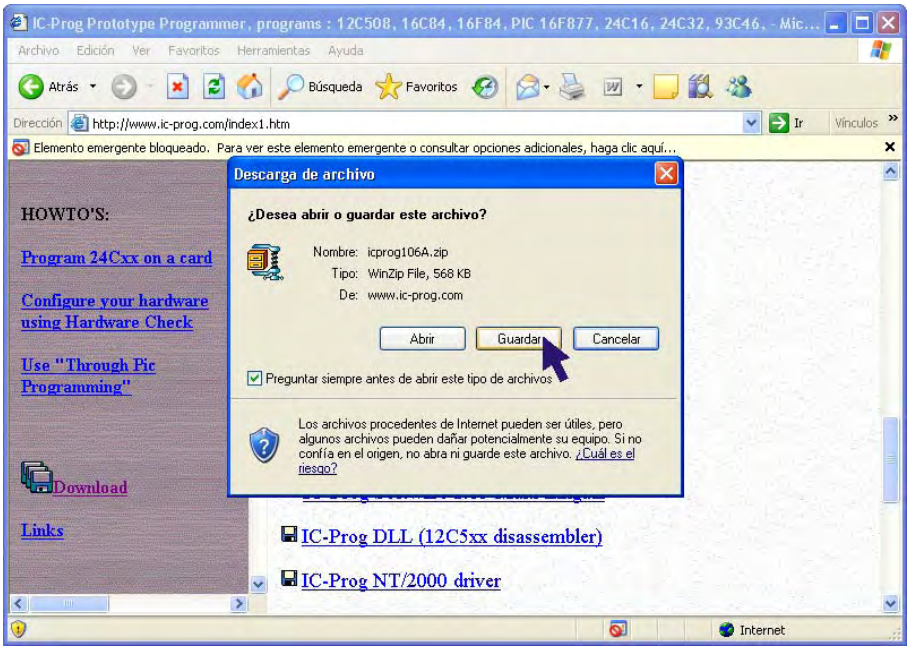
2c) En esta pantalla podrá descargar 2 archivos simultáneamente, el primero es el programa **IC-Prog 1.06A**, el segundo es el **driver para windows NT/2000yXP** que podría necesitar si dispone de estos sistemas operativos, y opcionalmente puede descargar el archivo de ayuda en español **Helpfile in Spanish Language**. Primero descargue IC-Prog 1.06A como se indica en el siguiente literal.



2d) Para poder descargar los 3 archivos simultáneamente, haga un clic con el botón *derecho* del mouse en **IC-Prog Software 1.06A Multi-Lingual**, luego haga clic en **Abrir en una ventana nueva**, se presentará una ventana similar al literal 2e) proceda igualmente como en el literal 1g) y cuando esté en el literal 1h) minimícelo. Vuelva a la pantalla 2c) y proceda a descargar el driver NT/2000 y el archivo de ayuda Helpfile Spanish Language.

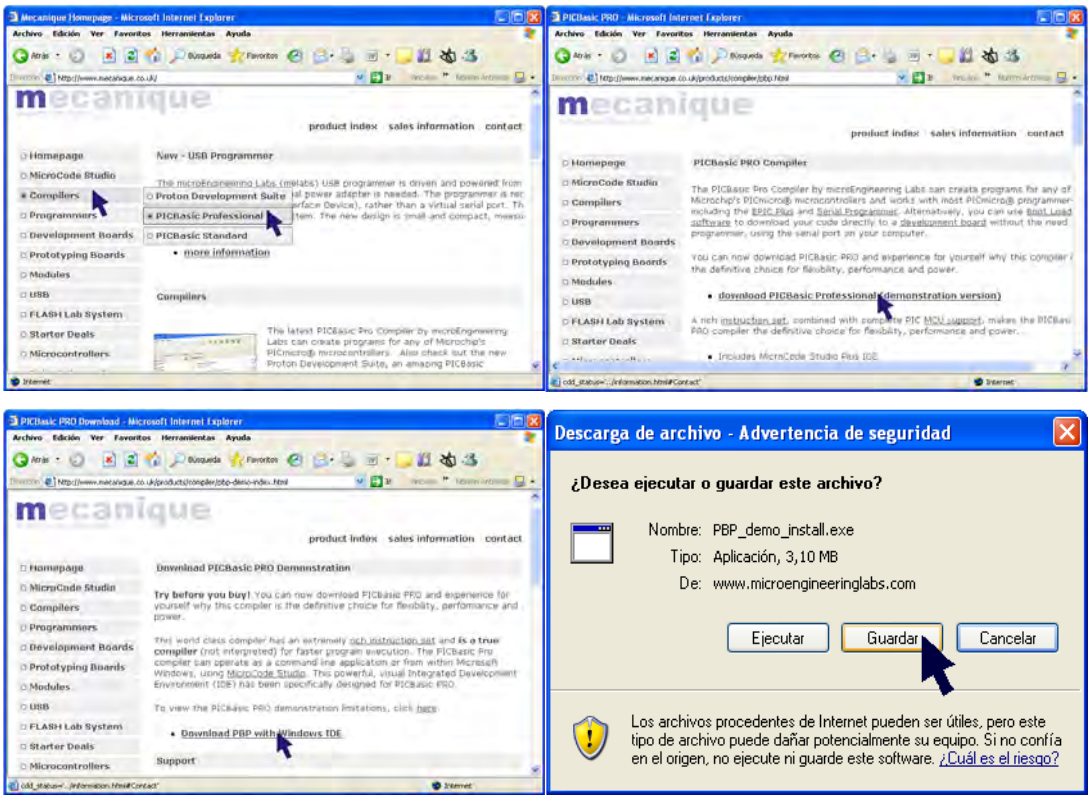


2e) Proceda a guardar el archivo, y luego continúe descargando los otros 2 archivos restantes.

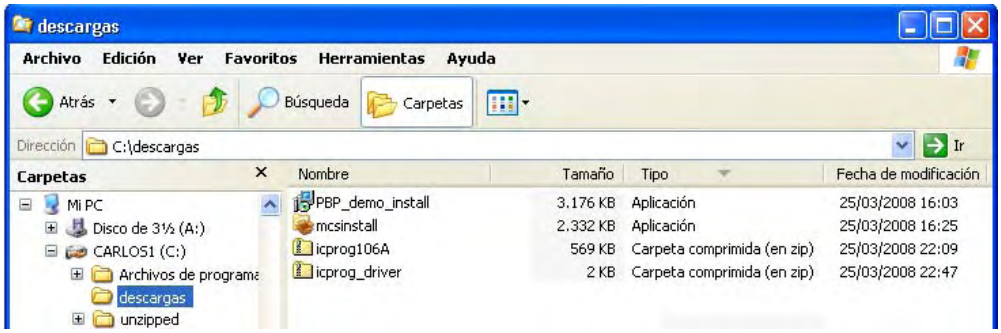


1.3 DESCARGA DEL COMPILADOR PICBasic Pro.

Este es el compilador que nos facilita la programación de los PIC´S, hasta aquí ya dispone de un programador de PIC el IC-Prog 106A, y un editor de texto el Microcode Studio 3.0.0.5 y sólo falta un programa compilador que se encargará de generar el archivo hexadecimal .HEX, necesario para poder grabar en un microcontrolador PIC. Para esto necesita adquirir el programa pbp 2.47 que cuesta alrededor de 250 USD., también puede descargar la versión demo en la misma página www.mecanique.co.uk. haciendo un clic en **Compilers**, y luego en **PICBasic Professional**, a continuación proceda a descargarlo como se aprendió en los casos anteriores.

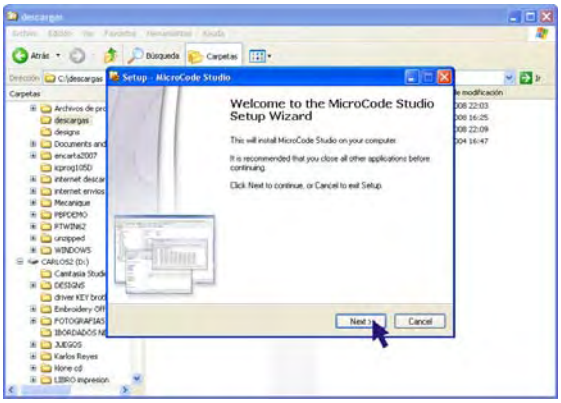


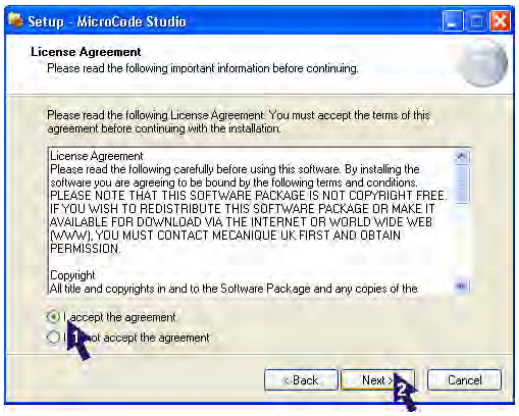
El archivo que descargará se llama PBP_demo_install.exe, este archivo ejecutable contiene en una carpeta (mcs) otro instalador de Microcode Studio versión 2.1.0.7, el cual ya no hace falta porque ya descargó anteriormente la última versión del mismo, esto se explicará más adelante en la instalación (ver pág. 14). A continuación se muestra todos los archivos descargados hasta aquí.



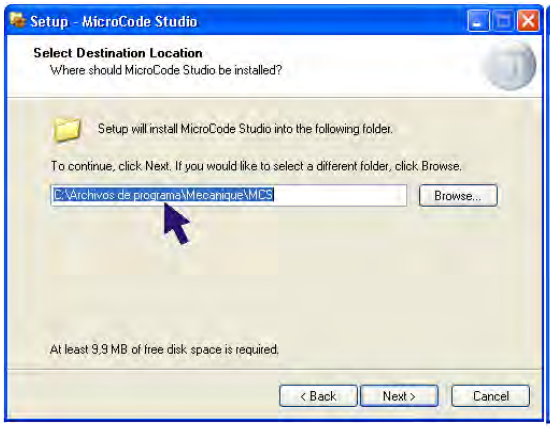
1.4 INSTALACIÓN DEL SOFTWARE MicroCode Studio.

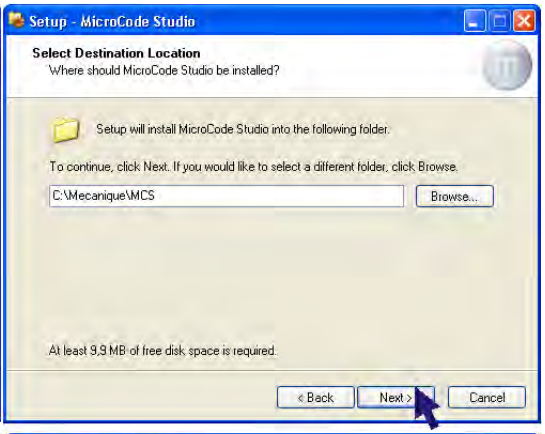
Para instalar el Entorno de Desarrollo Integrado IDE, se debe ejecutar el archivo mcsinstall.exe que terminó de descargar anteriormente, se abrirá una ventana de bienvenida (ver la siguiente fig. izquierda), luego presione **Next** y en la siguiente ventana (figura derecha), marque **I accept the agreement** y presione **Next**.

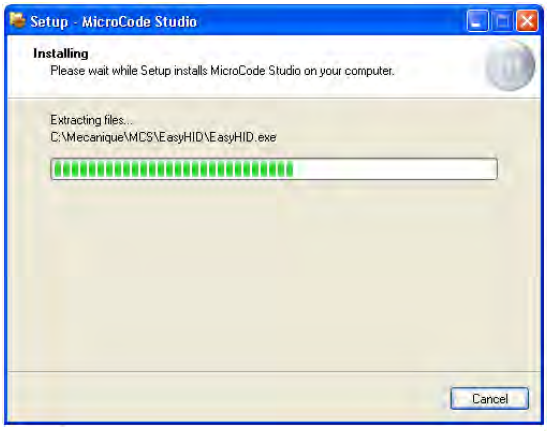


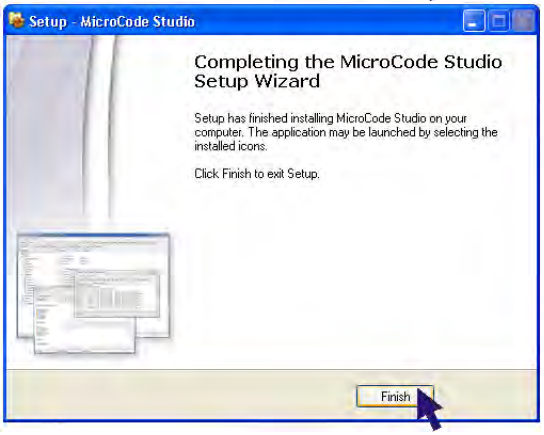


A continuación aparecerá una nueva ventana en donde se muestra el lugar donde se va a instalar el archivo, C:\Archivos de programa\Mecanique\MCS, elimine \Archivos de programa y deje que se instale en **C:\Mecanique\MCS**, presione **Next**, espere unos segundos y finalmente presione **Finish**.





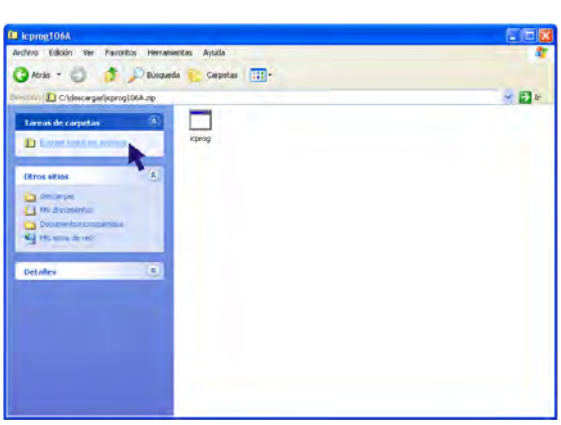


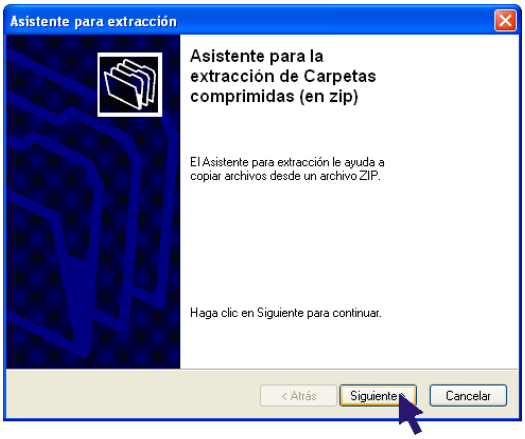


1.5 INSTALACIÓN DEL SOFTWARE PROGRAMADOR IC-Prog 1.06A.

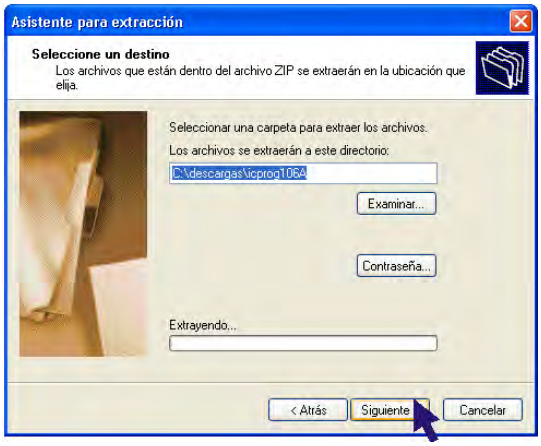
Para instalar este software se necesita descomprimirlo, existen 2 formas de hacerlo, la primera es utilizando el extractor de archivos de WINDOWS, y la segunda es utilizando el extractor de archivos de **WINZIP** que se enseñará más adelante.

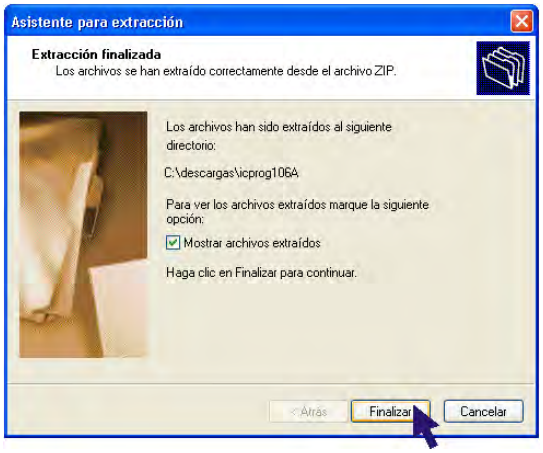
Para descomprimir el archivo con WINDOWS, localice el archivo **icprog106A.zip** a través del explorador de Windows y de doble clic, se abrirá una nueva ventana mostrando su contenido (ver la siguiente fig. izquierda), luego presione en **Extraer todos los archivos** y en la siguiente ventana presione **Siguiente**.



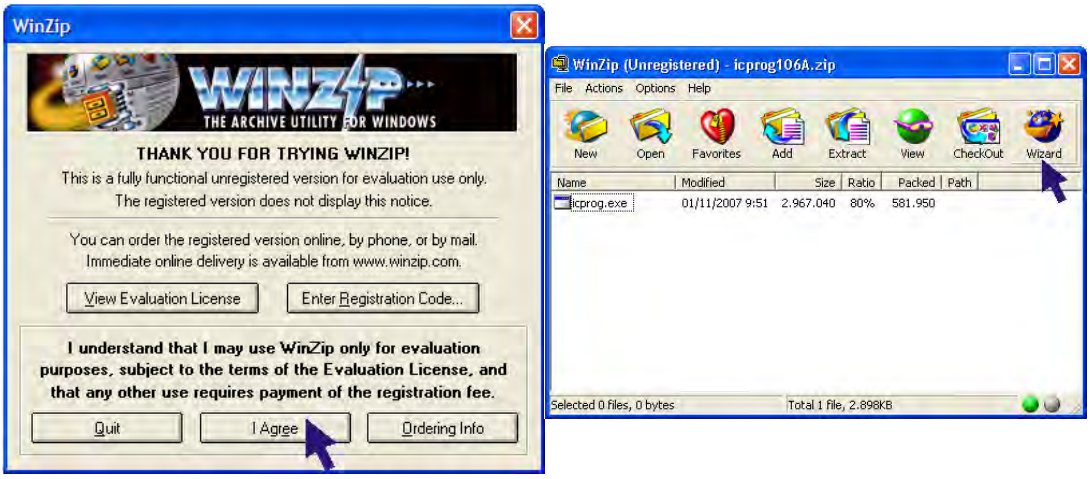


A continuación aparecerá una nueva ventana en donde se muestra el lugar que se va a descargar el archivo, déjelo en la misma carpeta C:\descargas\icprog106A, presione **Siguiente**. Luego se abre una nueva ventana informando que los archivos han sido descomprimidos, marque la casilla **Mostrar archivos extraídos** y presione Finalizar (figura derecha).





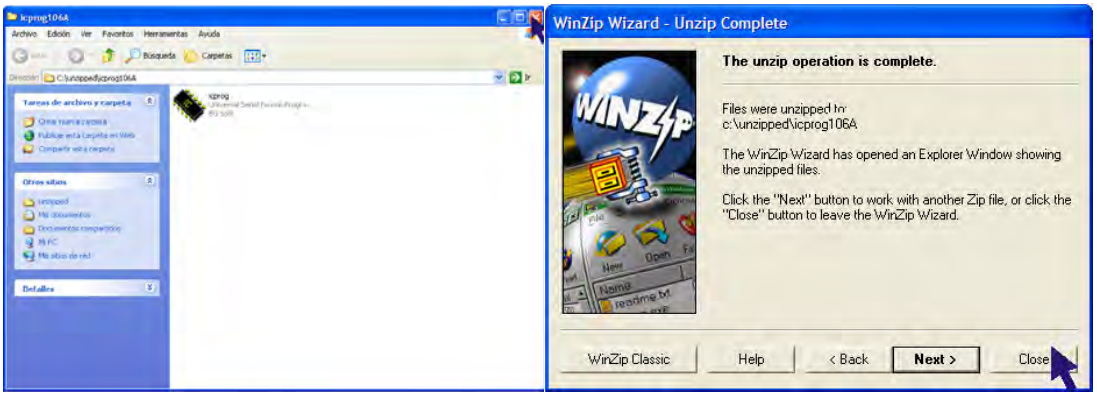
Ahora procedamos a descomprimir el archivo mediante el software **WINZIP**, este software lo puede descargar gratuitamente desde www.winzip.com. O puede utilizar el que se incluye en el CD en **CD:\Soft-Free**. Una vez que esté instalado **WINZIP** en el computador, localice el archivo **icprog106A.zip** a través del explorador de windows y haga doble clic sobre el mismo, en ese instante se ejecutará el programa **WINZIP**, en esta pantalla presione la tecla Agregar (I Agree), sólo si sale una pantalla similar a la figura derecha que se presenta a continuación presione Wizard.



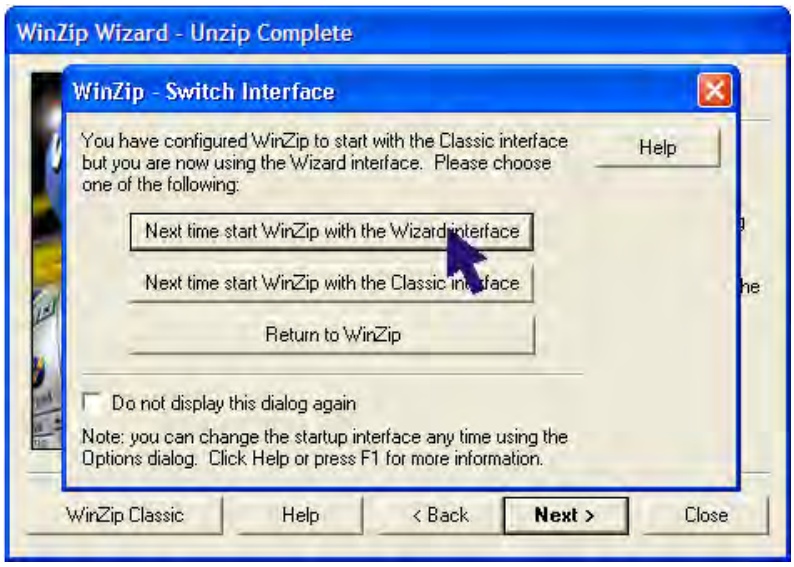
Luego presione **Next**, le preguntará si desea adicionar un fólder favorito ponga **No**.

A continuación seleccione **Unzip or install from "icprog106A.zip"** y presione **Next**, luego le indicará la carpeta en donde se va a ubicar C:\unzipped\icprog106A si lo desea déjelo ahí pero tome en cuenta que después tendrá que reubicarle en C:\mecanique\icprog106A, en todo caso en esta ocasión se lo instalará en C:\unzipped\icprog106A, presione la tecla **Unzip Now** y espere unos segundos.

Inmediatamente aparecerá la ventana de C:\unzipped\icprog106A con su ejecutable en el interior, cierre esta ventana y volverá a la pantalla de diálogo de WinZip, esta vez presione **Close**.

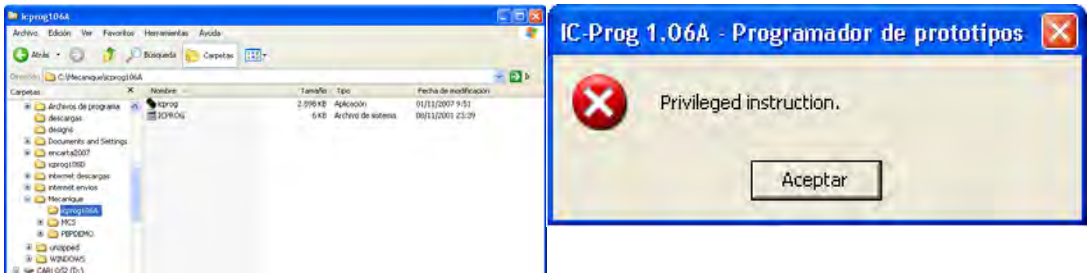


A continuación aparecerá una pantalla similar al siguiente gráfico, en donde le preguntará la forma en que desea que se inicie WinZip para las próximas ocasiones, sea modo Classic o Wizard efectivamente presione Wizard, esto le ahorrará tiempo para la próxima vez que ejecute WinZip.



1.6 INSTALACIÓN DEL DRIVER PARA WINDOWS NT/2000/XP.

Si usted dispone de cualquiera de estos sistemas operativos es necesario tener instalado este driver para que el programador IC-prog106A funcione correctamente. Caso contrario observará una serie de errores en la ejecución del programa, (ver figura derecha). Para instalarlo primero necesita encontrar el archivo **Icprog_driver.zip** que descargó del internet, luego proceda a descomprimirlo como se aprendió anteriormente. Una vez que termine de descomprimir, aparecerá la carpeta icprog_driver el cual contiene el archivo **icprog.sys**, este archivo debe moverlo dentro de la carpeta **C:\mecanique\icprog106A**, junto al archivo ejecutable icprog.exe sólo de esta manera se lo podrá activar, (ver la siguiente figura izquierda).



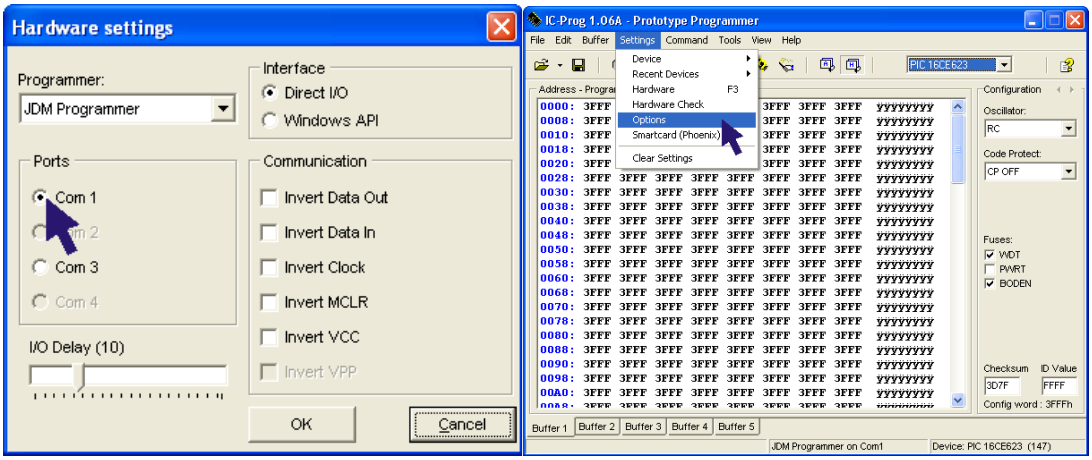
Ahora que ya dispone de este driver, debe activarlo de la siguiente manera: Primero ejecute el archivo **icprog.exe**, la primera vez aparecerá una pantalla en donde se debe seleccionar el puerto com que desea trabajar, hay casos que aparecen dos puertos disponibles, seleccione el puerto en donde está conectado el grabador de micros, si no aparece ningún puerto disponible, tiene dos opciones:

a) Si utiliza un mouse serial y dispone en su computador un conector PS2 o un USB, es aconsejable comprarse un nuevo mouse PS2 o USB y conectarlo, de esta manera quedará libre el puerto serial para la conexión del grabador de micros.

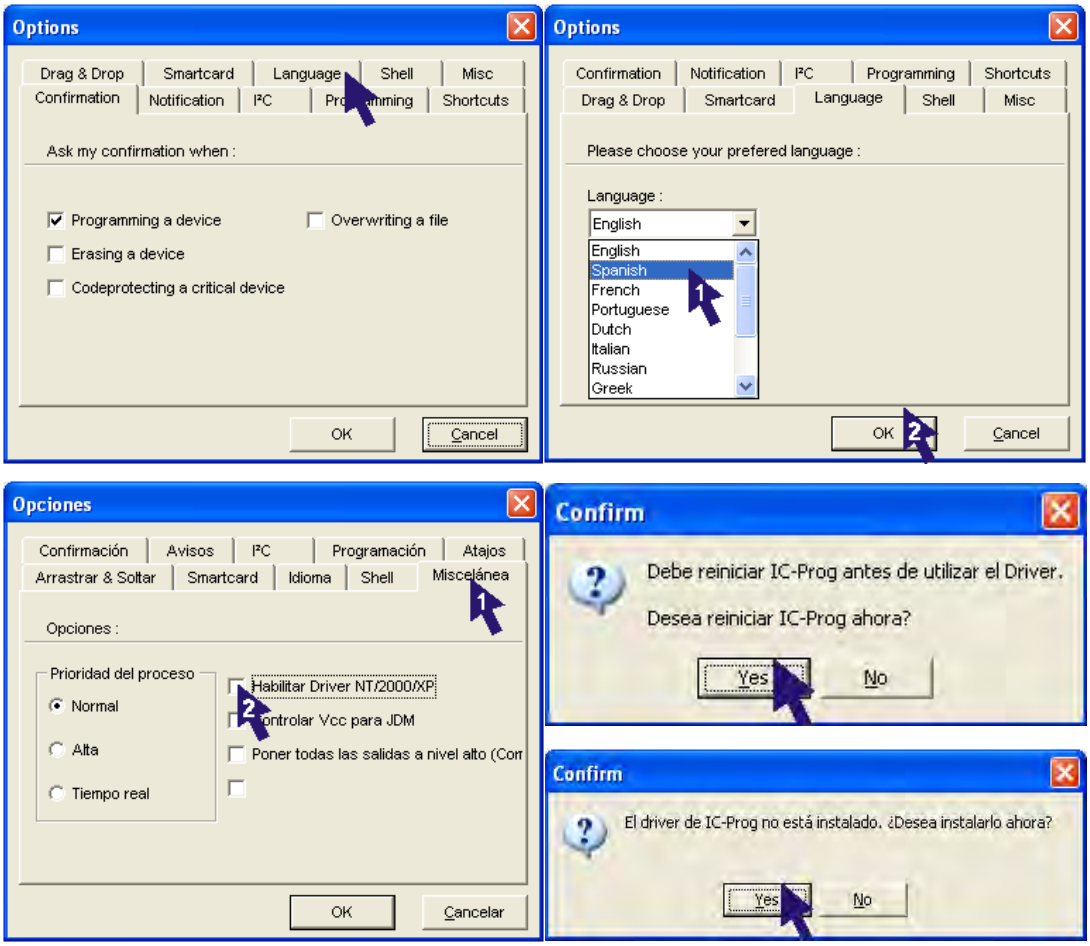
b) Si no dispone de ningún puerto serial, lo aconsejable es comprar una tarjeta de puertos seriales e instalar en su computador.

NOTA: El grabador de micros que se incluye con este libro no funciona en computadoras portátiles ni siquiera con los conversores de USB a serial debido a su bajo voltaje.

Una vez solucionado el problema de los puertos seriales de un clic en OK, aparecerá una pantalla similar al de la derecha de los siguientes gráficos, en esta ventana abra **Settings** y luego haga un clic en **Options**.



En esta nueva pantalla, se puede cambiar el lenguaje, seleccione Spanish y luego presione **OK**, en ese mismo instante observará que todo cambia a español, bien ahora vuelva a abrir **Ajustes** (antes llamado Settings) y luego de un clic en **Opciones**, esta vez de un clic en **miscelánea** para habilitar el driver de Windows **NT/2000/XP**, una vez que se marque el casillero aparecerá un cuadro de diálogo preguntando si desea reiniciar IC-Prog ahora, presione YES, luego aparecerá otro cuadro de diálogo preguntando si desea instalar el driver de IC-Prog, presione YES (ver los siguientes gráficos). **Si el problema persiste deshabilite el DRIVER y vuélvalo a habilitar nuevamente**.

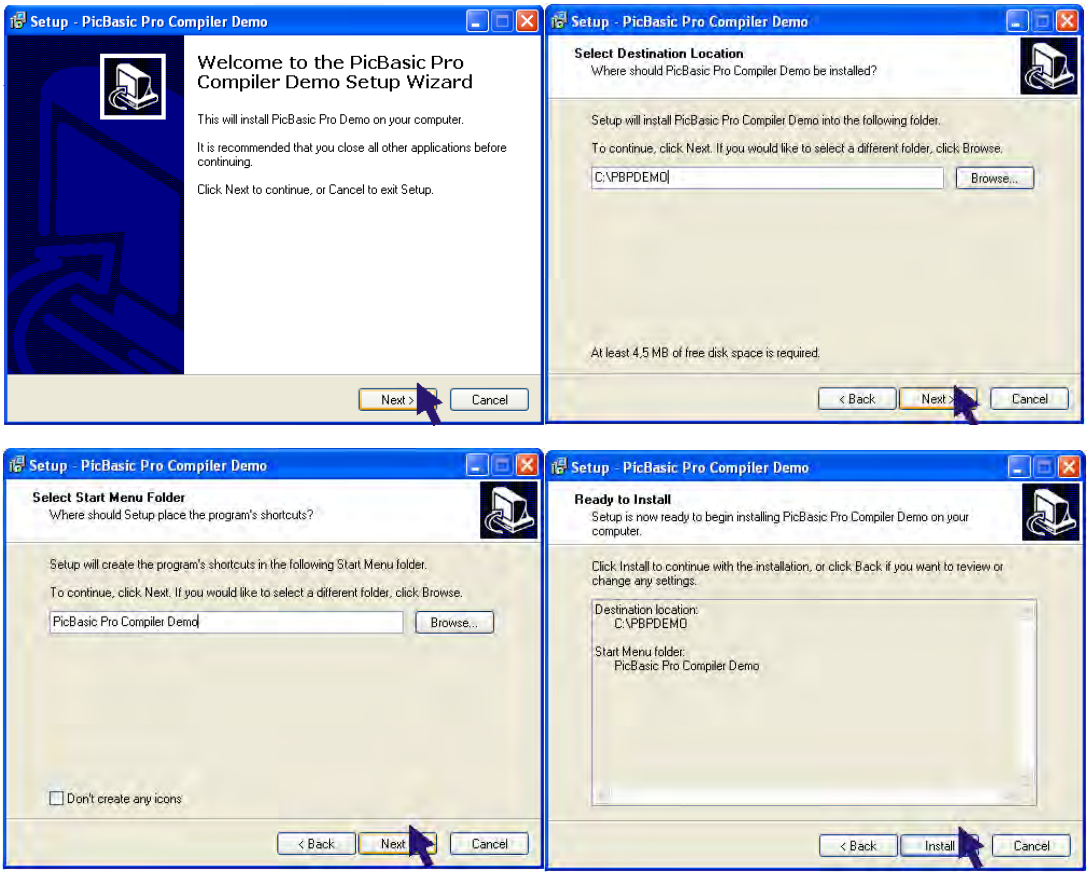


1.7 INSTALACIÓN DE pbp247 (PicBasic Pro versión 2.47).

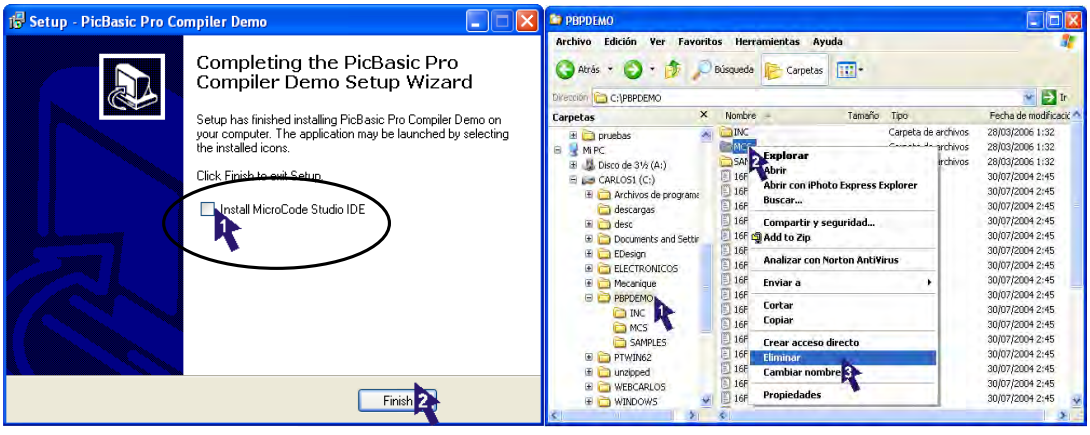
Este programa se debe descomprimir de igual forma que se aprendió anteriormente, si decide utilizar la versión demo debe tener en cuenta que sólo puede compilar 31 líneas de programa, también se debe considerar que no se puede incluir la declaración **INCLUDE**, por lo que no se puede realizar proyectos de comunicación serial. Para este libro se asume que usted tiene la versión completa de PICBasic PRO, de todas maneras la mayoría de las prácticas a realizarse son posibles compilarlos con la versión demo, además si no dispone de ningún compilador se incluye en el CD todos los archivos .HEX, necesarios para grabar en el PIC16F628A, de esta manera se podrá ver el funcionamiento de todos los proyectos que se encuentran en este libro.

1.8 INSTALACIÓN DEL COMPILADOR PicBasic Pro versión DEMO.

Para instalar este compilador, debe ejecutar el archivo **PBP_demo_install.exe**, y seguir los pasos que se muestran en las siguientes figuras, puede ubicarle directamente dentro de la carpeta C:\mecanique\PBPDEMO, si lo prefiere, caso contrario lo deberá mover posteriormente.



En esta última ventana asegúrese que la casilla **Install MicroCode Studio IDE** no esté marcada, pues este es el instalador de Microcode Studio versión 2.1.0.7, el cual ya no lo necesitamos, más bien lo eliminaremos para que no ocupe espacio en el disco duro (ver figura derecha).



CAPÍTULO 2
EL MICROCONTROLADOR PIC

2. ¿QUÉ ES UN MICROCONTROLADOR?.

Un microcontrolador es un circuito integrado, en cuyo interior posee toda la arquitectura de un computador, esto es CPU, memorias RAM, EEPROM, y circuitos de entrada y salida.

Un microcontrolador de fábrica, no realiza tarea alguna, este debe ser programado para que realice desde un simple parpadeo de un led hasta un sofisticado control de un robot. Un microcontrolador es capaz de realizar la tarea de muchos circuitos lógicos como compuertas AND, OR, NOT, NAND, conversores A/D, D/A, temporizadores, decodificadores, etc., simplificando todo el diseño a una placa de reducido tamaño y pocos elementos.

2.1. EL MICROCONTROLADOR PIC16F628A.

Los microcontroladores PIC (Peripheral interface Controller), son fabricados por la empresa **MICROCHIP Technology INC**. cuya central se encuentra en Chandler, Arizona, esta empresa ocupa el primer lugar en venta de microcontroladores de 8 bits desde el año 2002. Su gran éxito se debe a la gran variedad (más de 180 modelos), gran versatilidad, gran velocidad, bajo costo, bajo consumo de potencia, y gran disponibilidad de herramientas para su programación. Uno de los microcontroladores más populares en la actualidad es el PIC16F628A y sus variantes PIC16F627A y PIC16F648A, estos modelos (serie A) soportan hasta 100.000 ciclos de escritura en su memoria FLASH, y 1'000.000 ciclos en su memoria Eeprom, este está reemplazando rápidamente al popular PIC16F84A, pues presenta grandes ventajas como son:

	PIC16F84A	PIC16F627A	PIC16F628A	PIC16F648A
Memoria de programa Flash	1024 x 14	1024 x 14	2048 x 14	4096 x 14
Memoria datos RAM	68 x 8	224 x 8	224 x 8	256 x 8
Memoria datos EEPROM	64 x 8	128 x 8	128 x 8	256 x 8
Pines de entrada/salida	13	16	16	16
Comparadores de voltaje	0	2	2	2
Interrupciones	4	10	10	10
Timers 8/16 bits	1	3	3	3
Módulos PWM / CCP	No	Si	Si	Si
Comunicación serial USART	No	Si	Si	Si

Figura 2.1.1. *Tabla de comparación entre el PIC16F84A y los PIC16F6XX.*

Todas estas y otras ventajas más como el oscilador interno RC de 4MHZ, MCLR programable, mayor capacidad de corriente, Programación en bajo voltaje, etc. Lo hacen al PIC16F628A, como el microcontrolador ideal para estudiantes y aficionados, ya que al tener oscilador interno y el MCLR (master clear) sea programable, es mucho más sencillo ponerlo en funcionamiento, basta con conectar al pin 14 a 5V y el pin 5 a tierra para que empiece a trabajar (ver figura 2.7.1).

2.2. ARQUITECTURA DEL PIC16F628A.

El PIC16F628A utiliza un procesador con arquitectura **Harvard**, consiguiendo mayor rendimiento en el procesamiento de las instrucciones, esta arquitectura a diferencia de la Von Neumann, utiliza dos bloques de memorias independientes, una contiene instrucciones y la otra sólo datos, cada una con su respectivo sistema de buses de acceso, 8 líneas para los datos y 14 líneas para las instrucciones, con lo que es posible realizar operaciones de acceso lectura o escritura simultáneamente en las 2 memorias, a esto se conoce como paralelismo (figura 2.2.1).

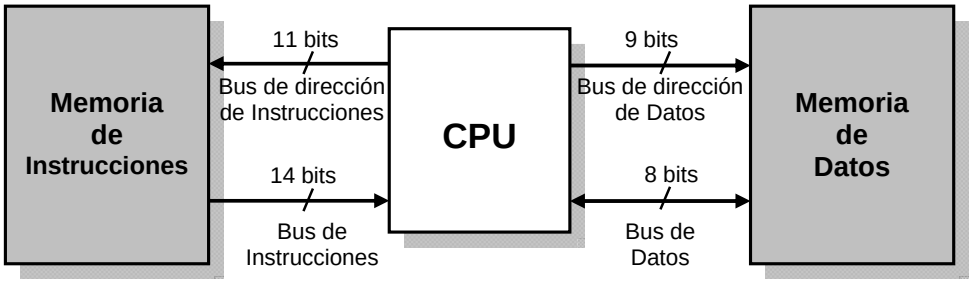


Figura 2.2.1. *La arquitectura Harvard maneja la memoria de datos y la memoria de instrucciones por separado y con diferentes capacidades.*

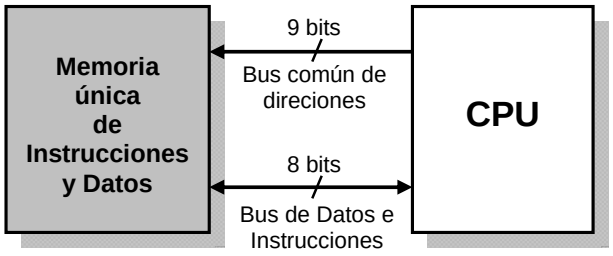


Figura 2.2.2. *En la arquitectura Von Neumann se conecta el CPU con una memoria única en donde se almacenan datos e instrucciones en forma indistinta, compartiendo el mismo bus.*

El CPU del microcontrolador 16F6XX emplea una avanzada arquitectura **RISC** (computador con juego de instrucciones reducido) con un set de 35 instrucciones poderosas pertenecientes a la gama media de la familia de los microcontroladores PIC, la mayoría de instrucciones se ejecutan en un ciclo de instrucción a excepción de los saltos que requieren de 2 ciclos, dentro de su Procesador existe una PILA de 8 niveles que permiten el anidamiento de subrutinas, esto quiere decir que puede retomar 8 lugares diferentes de línea de programa e ir regresando a cada uno en el orden inverso al que fueron anidados (ver figura 2.3.1).

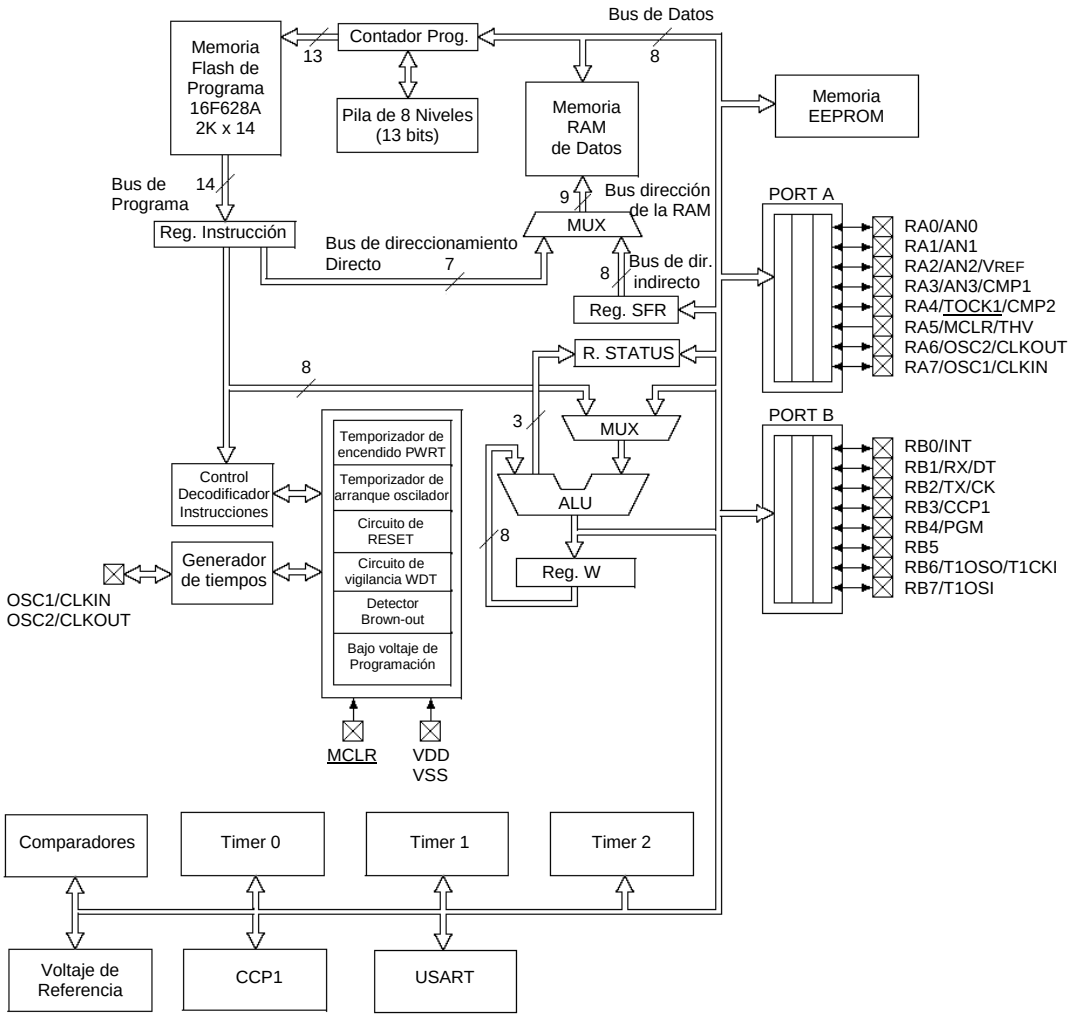


Figura 2.2.3. *Diagrama de los bloques funcionales del PIC, su conexión interna es mediante buses, se aprecia la conexión de las 3 memorias Flash, Ram y Eeprom.*

2.3. LA MEMORIA DE PROGRAMA.

Conocido también como memoria de instrucciones, aquí se escribe las ordenes para que el CPU las ejecute. En el caso del microcontrolador PIC16F628A tiene memoria de programa no volátil tipo **FLASH**, en comparación a su antecesor la memoria EEPROM, este se caracteriza por ser más rápido en el proceso de escritura/borrado eléctrico, además dispone de mayor capacidad de almacenamiento, esta característica hace que sea ideal para prácticas de laboratorio en donde la grabación y el borrado son frecuentes (recuerde que soporta 100.000 ciclos de escritura/borrado). El bus de direcciones de la memoria de programa es de 13 bits, por lo que el Contador de Programa (PC) puede direccional 8192 posiciones de 14 bits cada una (desde la 0000h hasta 1FFFh), de las cuales sólo las primeras 2048 líneas tiene implementadas (desde la 0000h hasta la 07FFh), es decir que el PC sólo utiliza los 11 primeros bits de direcciones los demás bits los ignora.

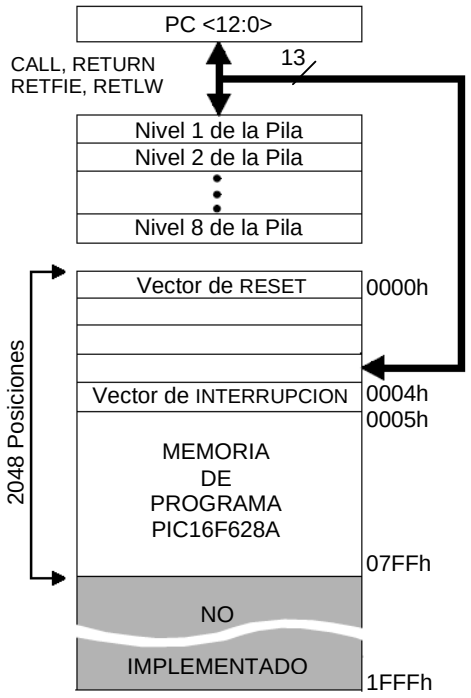


Figura 2.3.1. *Mapa de la memoria de Programa, de las 8196 posiciones, sólo tiene implementado las primeras 2048 posiciones, la dirección 0000h está reservada para el vector de reset y la 0004h está reservada para el vector de interrupción.*

2.4. LA MEMORIA DE DATOS.

El PIC16F628A, tiene dos tipos de memorias de datos, la RAM estática o SRAM (Random Access Memory) o memoria de acceso casual que es un tipo de memoria volátil, es decir sus datos permanecen en la memoria mientras exista alimentación en el dispositivo y es de vital importancia porque ahí residen dos tipos de datos, los registros de propósito general (GPR), en donde se almacenan las variables y los registros especiales (SFR), que son los encargados de llevar el contador de programa, el conteo del Temporizador, el estado de los puertos, la configuración de las interrupciones, etc.

El otro tipo de memoria es una memoria auxiliar no volátil llamada **EEPROM**, con capacidad de 128 posiciones de 8 bits cada una. Esta memoria puede ser accedida por el usuario mediante programación, es muy útil para almacenar datos que el usuario necesita que se conserven aún sin alimentación, tal es el caso de la clave de una alarma, esta puede ser modificada, pero no debe perderse por un corte de energía, el fabricante asegura que la serie PIC16F6XXA, tiene una retención de datos en esta memoria mayor a 100 años.

Como este microcontrolador es fabricado con tecnología **CMOS**, su consumo de potencia es muy bajo (2 mA a 4 Mhz) y además es completamente estático, lo que significa que si el reloj se detiene los datos de la memoria RAM no se pierden, esto mientras el micro sigue alimentado. La memoria de datos RAM, tiene 512 líneas de 8 bits cada una y está particionada por 4 bancos; el banco 0, banco 1, banco 2 y banco 3, cada uno con 128 bytes, el acceso a cada banco de memoria lo realiza los bits RP1 y RP0 del registro STATUS, la mayoría de los bytes son ocupados por los Registros de Funciones Especiales (SFR) o no están implementadas. Para el caso del PIC16F628A sólo 224 posiciones de memoria RAM están disponibles para los Registros de Propósito General (GPR), la distribución de memoria se muestra en la tabla de la figura 2.4.1.

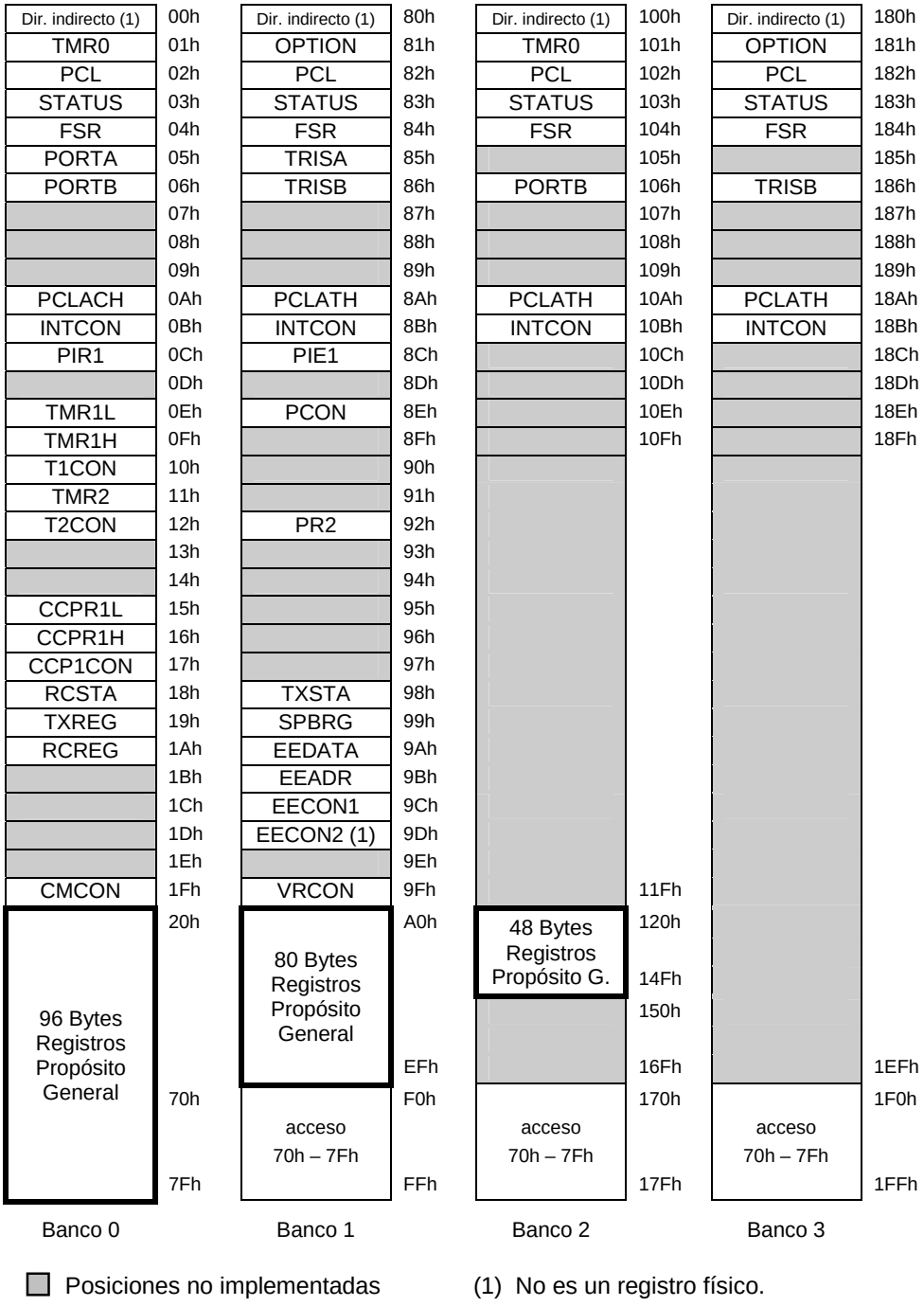


Figura 2.4.1. *Mapa de la memoria de DATOS SRAM, los bloques marcados con tramas no son implementados, se leen "0" las localidades marcadas con (1) no son registros físicos, las localidades 20h a 7Fh, corresponden a los 96 bytes de los registros de propósito general GPR, 80 bytes GPR en el banco 1(A0h – EFh) y 48 bytes GPR en el banco 2 (120h – 14Fh), dando un total de 224 bytes disponibles para el usuario. Para mayor información y utilización de los SFR, refiérase al datasheet del PIC16F6XXA que se incluye en el CD de este libro.*

2.5. CARACTERÍSTICAS GENERALES.

Hasta aquí se puede resumir las características más relevantes del PIC16F628A, estas son:

- Velocidad de operación hasta 20 MHZ con oscilador externo.
- Oscilador interno RC (resistencia condensador) de 4 MHZ calibrado de fábrica al ±1 %.
- Admite 8 configuraciones de oscilador.
- 8 niveles de PILA.
- Procesador con arquitectura HARVARD.
- Conjunto reducido de instrucciones RISC (35) gama media.
- Instrucciones de un ciclo excepto los saltos (200nS por instrucción a 20 MHZ).
- Resistencias PULL-UP programables en el puerto B.
- Pin RA5 MCLR programable como reset externo o pin de entrada.
- Rango de operación desde 3V. hasta 5.5V.
- 15 pines de I/O y 1 sólo de entrada (RA5).
- Temporizador Perro guardián WDT independiente del oscilador.
- Programable con bajo voltaje LPV (5V.).
- Programación serial en Circuito ICSP por 2 pines: RB6 reloj y RB7 datos.
- Código de protección programable por sectores.
- Memoria de programa FLASH 2048K. de 100.000 ciclos escritura/borrado.
- Memoria de datos EEPROM de 1.000.000 ciclos escritura/borrado de 100 años retención.
- 2 circuitos comparadores análogos con entradas multiplexadas.
- 3 Timers, Timer 0 a 8 bits, Timer 1 a 16 bits y Timer 2 a 8 bits.
- Módulos CCP, Captura compara 16 bits, y PWM, modulación de ancho de pulso 10 bits.
- 10 fuentes de interrupción.
- Módulo de comunicación serial USART/SCI.
- Capacidad de corriente para encender leds directamente (25 mA I/O) por cada pin.

2.6. DIAGRAMA DE PINES Y FUNCIONES.

Excluyendo los dos pines de alimentación, todos los 16 pines restantes pueden ser configurados como entradas o salidas, algunos de ellos tienen funciones especiales, ver figura 2.6.2.

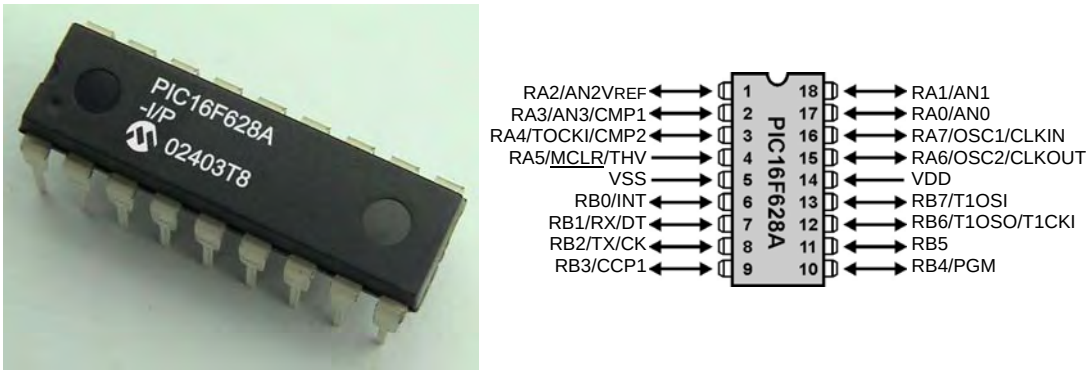


Figura 2.6.1. *Presentación más popular del PIC16F628A el PDIP y su diagrama de pines.*

PIN	NOMBRE	DESCRIPCION
17	RA0/AN0	Pin bidireccional I/O, entrada comparador análogo.
18	RA1/AN1	Pin bidireccional I/O, entrada comparador análogo.
1	RA2/AN2/VREF	Pin bidireccional I/O, entrada comp. análogo y Voltaje de referencia.
2	RA3/AN3/CMP1	Pin I/O, entrada comp. análogo y salida del comparador análogo 1.
3	RA4/T0CKI/CMP2	Pin I/O, entrada reloj TIMER0 y salida del comparador análogo 2.
4	RA5/MCLR/VPP	Pin de entrada, en modo MCLR activa RESET externo.
15	RA6/OSC2/CLKOUT	Pin I/O, entrada oscilador externo, salida de ¼ de la frecuencia OSC 1.
16	RA7/OSC1/CLKIN	Pin I/O, entrada oscilador externo, entrada del reloj externo.
6	RB0/INT	Pin I/O, resistencia Pull-Up programable, entrada de interrupción ext.
7	RB1/RX/DT	Pin I/O, resist. Pull-Up, entrada dato RS232, I/O dato serial asincrónico.
8	RB2/TX/CK	Pin I/O, resist. Pull-Up, salida dato RS232, I/O señal de reloj asincrónico.
9	RB3/CCP1	Pin I/O, resist. Pull-Up, módulo CCP/PWM entrada o salida.
10	RB4/PGM	Pin I/O, resist. Pull-Up, entrada del voltaje bajo de programación.
11	RB5	Pin I/O, resistencia Pull-Up programable.
12	RB6/T1OSO/T1CKI	Pin I/O, resist. Pull-Up, salida oscilador TIMER1, entrada reloj de ICSP.
13	RB7/T1OSI	Pin I/O, resist. Pull-Up, entrada oscilador TIMER1, I/O datos de ICSP.

Figura 2.6.2. *Tabla de pines con sus funciones especiales.*

NOTA: sus 2 puertos el A y el B entregan un total de 200mA cada uno, es decir 25 mA cada pin. En modo sumidero pueden soportar cada uno de sus puertos 200mA. es decir 25 mA. cada pin.

2.7. CONSIDERACIONES BÁSICAS PERO MUY ÚTILES A LA HORA DE MONTAR UN PROYECTO.

Es muy importante tomar en cuenta estas recomendaciones ya que si no se las sigue podría correr el riesgo de dañar el PIC:

1. recuerde que el PIC tiene tecnología CMOS, esto quiere decir que consume muy poca corriente pero que a la vez es susceptible a daños por estática, se recomienda utilizar pinzas para manipular y así poder transportar desde el grabador al protoboard o viceversa, o a su vez utilizar una manilla antiestática.
2. procure utilizar un regulador de voltaje como el 7805 que nos entrega exactamente 5V. y no un adaptador de pared, ya que el voltaje de salida no siempre es el mismo del que indica su fabricante, por último puede utilizar un circuito con un diodo zener de 5.1 V.
3. no sobrepase los niveles de corriente, tanto de entrada como de salida, recuerde que el PIC puede entregar por cada uno de sus pines una corriente máxima de 25 mA. Asimismo soporta una corriente máxima de entrada de 25 mA., esto quiere decir que puede encender un led con una resistencia de 330 Ω, revisemos:

Voltaje que sale de un pin del PIC, si es alimentado con 5 V. = 5V.
Corriente que requiere el led para un encendido normal = 15 mA.
¿Qué resistencia es necesario para encender el led correctamente?.

$$V=R \times I \qquad R=\frac{V}{I} \qquad R=\frac{5\ V.}{0{,}015\ A.} \qquad R=333{,}33\ \Omega \approx 330\ \Omega$$

Ahora un ejemplo de corriente de entrada al PIC, si queremos poner un pulsador ¿Qué resistencia sería la mínima a colocarse?.

Como sabemos la corriente de entrada que soporta por cada pin del PIC es de 25mA entonces para un pulsador tenemos que:

$$V=R \times I \qquad R=\frac{V}{I} \qquad R=\frac{5\ V.}{0{,}025\ A.} \qquad R=200\ \Omega \approx 220\ \Omega$$

Esto quiere decir que la resistencia mínima a colocarse sería de 220 Ω para estar al límite de la capacidad que soporta el PIC, pero no es muy aconsejable trabajar con los límites, por lo que se recomienda utilizar una resistencia de 1 KΩ a 10 KΩ, así el PIC estaría trabajando tranquilamente con una corriente de entrada de 5 mA o 0,5 mA respectivamente.

4. En algunos proyectos es necesario conectar un capacitor de 0,1uF o 1 uF en paralelo al PIC, este evita mal funcionamientos que podrían ocurrirle, en especial cuando se utiliza teclados matriciales y se tiene conectado adicionalmente un buzzer activo (parlante activo o chicharra) y relés.

5. Cuando se necesite precisión en el trabajo del PIC (comunicación serial, tonos DTMF, etc.), se recomienda utilizar un cristal oscilador externo de 4 MHZ en adelante, ya que el oscilador interno RC que posee no tiene muy buena precisión. En un experimento realizado se conectó dos PIC idénticos con el mismo programa el cual consistía en hacer parpadear un led con intervalos de 1 segundo, ambos PIC compartían la misma fuente y al momento de arrancar los dos parpadeos eran iguales, al transcurso de unos minutos los leds se habían desigualado, esto demuestra que la calibración interna no es igual en todos los micros, si utilizáramos cristales externos de 4 MHZ en ambos PIC, no se desigualan nunca, esto debido a que los cristales son muy precisos en cuanto a la frecuencia que entregan.

Con todas estas recomendaciones se tiene el siguiente diagrama para encender un led y conectar un pulsador sin que el PIC sufra ningún daño.

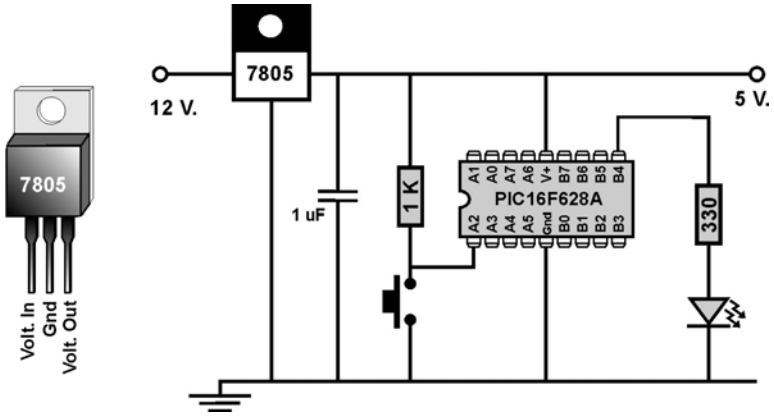


Figura 2.7.1. *Diagrama básico para conectar un PIC con un LED y un pulsador, noten que el PIC no necesita oscilador externo ni tampoco resistencia pull-up conectado al MCLR o puerto RA5 como lo necesitaba su antecesor el PIC16F84A.*

CAPÍTULO 3
EL PROGRAMA MicroCode Studio

3.1 CONFIGURACIÓN DE MicroCode Studio (IDE).

En este Capítulo se enseñará a configurar el editor de texto IDE, para tener el mejor rendimiento posible, lo primero que se debe hacer es agrupar dentro de la carpeta C:\mecanique los dos programas, el pbp 2.47 y el IC-prog 106A, con la finalidad de que la primera vez que ejecute el programa microcode, pueda encontrar inmediatamente su compilador pbp 2.47 y su programador IC-prog 106A. Para esto debemos utilizar el explorador de windows y buscar las dos carpetas que seguramente estarán dentro de C:\unzipped\ o C:\descargas\ y proceda a cortar y pegar dentro de C:\mecanique\, luego es necesario eliminarlos de su ubicación original, el siguiente gráfico muestra la forma de cómo debería quedar ubicadas las carpetas:

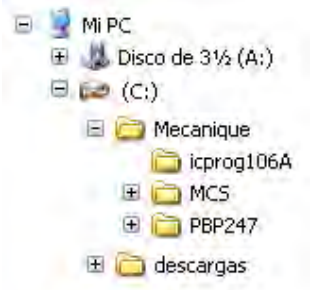
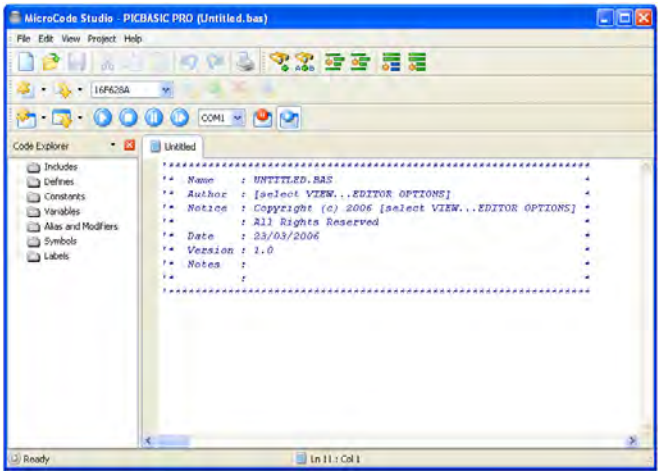
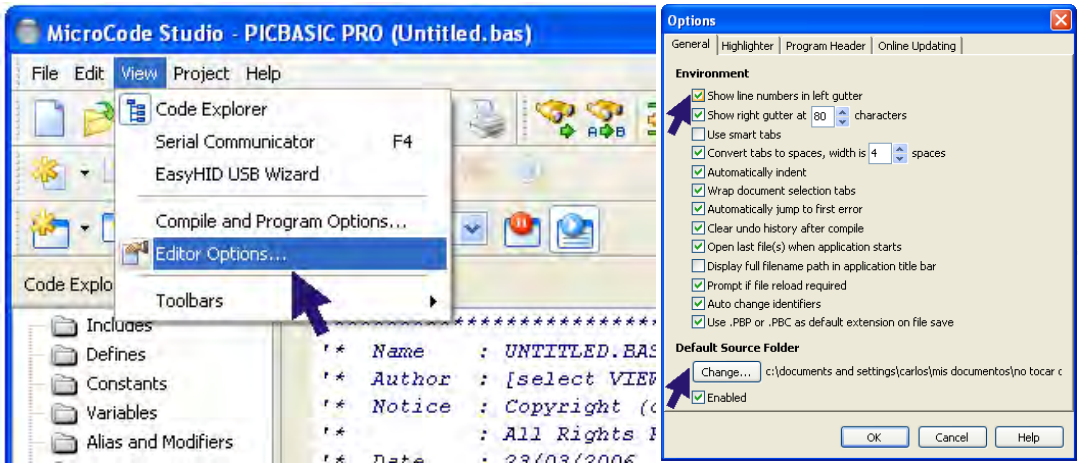


Figura 3.1.1. *Esquema de la ubicación de cada una de las carpetas*

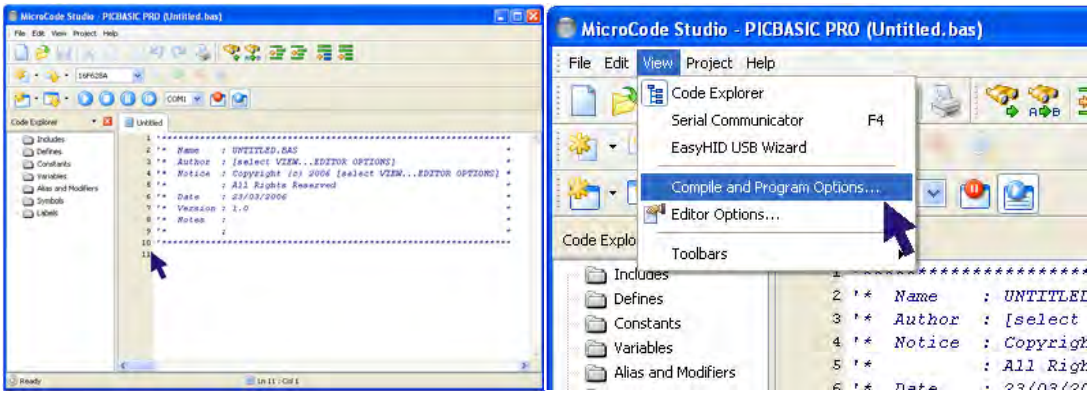
Ejecute C:\mecanique\MCS**CodeStudio.exe**, la primera vez el programa buscará el compilador disponible, en este caso el pbp247, y aparecerá una pantalla similar al siguiente gráfico:



En esta pantalla busque **View** y haga clic en **Editor Options...**, luego marque **show line numbers in left gutter** que sirve para que aparezca el número de la línea que está programando, esto es muy útil al momento de encontrar errores. Si desea también puede indicar la carpeta predefinida en la que desea guardar los archivos *.pbp, por ejemplo podría ser una ubicación donde ya previamente haya creado como: C:\mis documentos\ejercicios PIC, si no desea crear ningún vinculo, sólo presione la tecla **OK**.



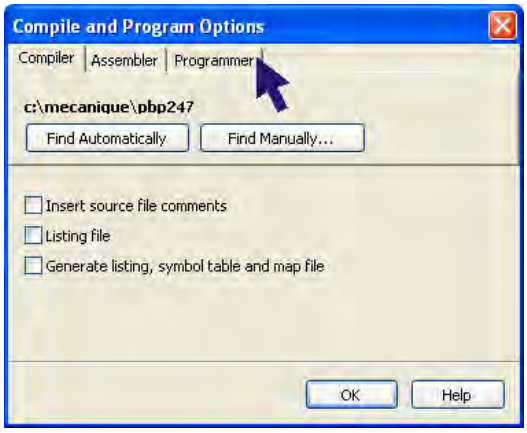
Bien ahora observe que aparecen números al lado izquierdo del editor de texto esto será muy útil en lo posterior, vuelva nuevamente a la parte superior donde dice **View** y de un clic en **Compile and Program Options**...

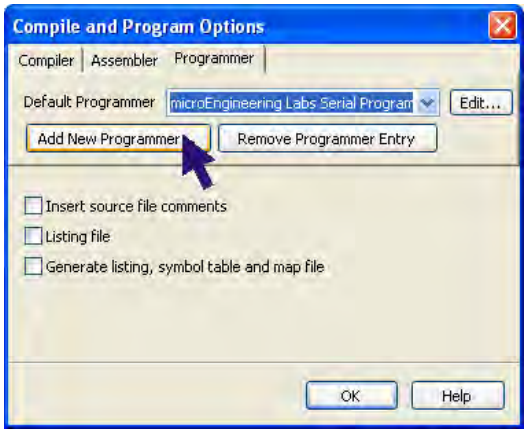


Aparecerá una nueva pantalla más pequeña, en donde verá que ya está predefinido la ubicación C:\mecanique\pbp247, si apareciera C:\Unzipped\pbp247 o cualquier otro destino es porque no lo borró de la ubicación anterior, y microcode tomo el pbp247 de ese lugar, para corregirlo debe eliminar el archivo C:\Unzipped\pbp247, y volver a ejecutar el programa microcode, automáticamente volverá a buscar el compilador.

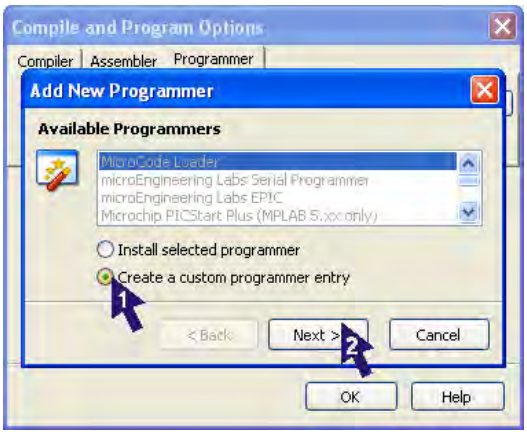
Una vez corregido lo anterior presione la pestaña Programmer, aquí aparece el programador disponible microEngineering Labs serial Programmer, pero no aparece IC-prog, así que debe crearlo con la finalidad de que a futuro pueda llamarlo desde microcode, si no lo hace deberá ejecutar por separado el programa IC-prog creando una demora en la programación.

La forma de adicionar el programa IC-Prog es presionando el botón que dice **Add New Programmer** ver los siguientes gráficos:

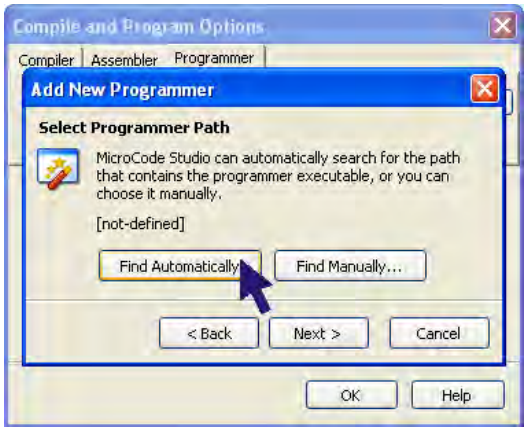




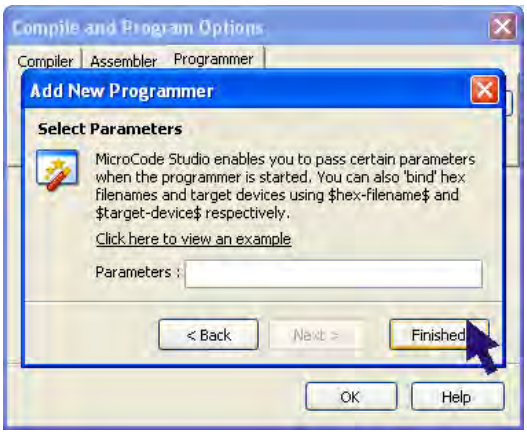
Inmediatamente aparece otra pantalla aun más pequeña en donde debe marcar **create a custom programmer entry**, luego presione **Next**. En la siguiente pantalla escriba cualquier nombre que desee darle al programador, en este caso será icprog y luego presione la tecla **Next**.

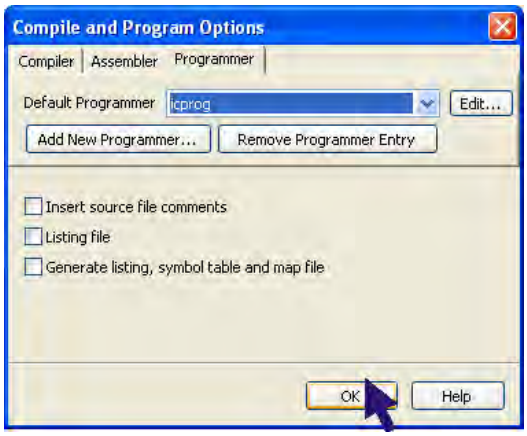


En esta pantalla le pedirá que ponga el nombre del archivo ejecutable escriba icprog.exe y luego presione **Next**. Aparecerá otra pantalla con dos botones el uno localiza automáticamente la carpeta en donde se encuentra el ejecutable, y el otro es para localizar manualmente, si está seguro que es el único archivo icprog.exe puede presionar la tecla **Find Automatically**.



En esta pantalla le pedirá parámetros de programación, como para IC-prog no hace falta no escriba nada y sólo presione la tecla **Finished**, luego desaparece esta pantalla y sólo queda la pantalla de **PICBasic Options**, en donde debe asegurarse de los cambios presionando **OK**. **Todos los ajustes realizados hasta aquí sólo se los reliza una sola vez.**





3.2 MANEJO DE MicroCode Studio.

Microcode Studio es un Entorno de desarrollo Integrado (IDE), diseñado exclusivamente para facilitar la programación de los microcontroladores PIC, los procedimientos para programar son muy sencillos, primero seleccione el modelo del PIC 16F628A, 16F877A, etc.(1), escriba el programa y guárdelo bajo un nombre, en este caso como **led intermitente** y por último presione el botón compilar (8), si el programa está bien escrito y sin fallas compilará y mostrará en la parte inferior izquierda el espacio que requiere en el PIC (4), enseguida se creará automáticamente 3 archivos: led intermitente.mac, led intermitente.asm y led intermitente.hex, este último es el más importante para el PIC y es el que se debe grabar en el microcontrolador. En la figura 3.2.1 se muestra las partes más importantes de la pantalla de MicroCode Studio.

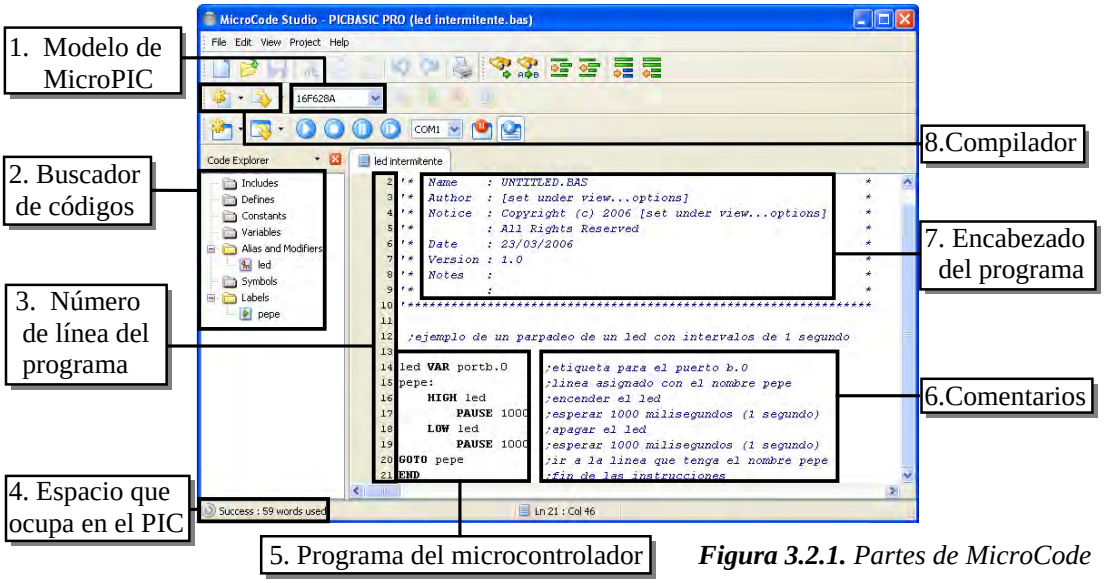


Figura 3.2.1. *Partes de MicroCode*

1. Modelo de MicroPIC.-

Esto es lo primero que debe seleccionar antes de empezar a programar, seleccione de acuerdo al modelo de Pic que se va a programar sea este 16F627, 16F627A, 16F628, 16F628A, 16F818, 16F819, 16F84A, 16F877A, etc.

---

2. Buscador de códigos.-

Aquí se van adicionando cada vez que se crea una variable, al incluir un define, o crear algún nombre de línea, sirve para saber qué componentes incluyen en el programa y también como buscador de líneas, para esto basta con dar un clic en el nombre de la línea que desea encontrar y automáticamente le indicará donde está dicha línea.

---

3. Número de línea del programa.-

Esto por defecto no viene habilitado, debe habilitarlo previamente, y es muy útil a la hora de encontrar errores, porque le indica el número de la línea en donde se halla un error.

NOTA: Para habilitar esta opción refiérase al Capítulo 3 página 26.
Este no es el tamaño en líneas que ocupa el PIC, sino el que ocupa en Basic.

---

4. Espacio que ocupa en el PIC.-

Este sí es el espacio que se requiere en la memoria FLASH del Pic y aparece una vez que se compila el programa, debe fijarse si alcanza en el PIC que dispone o debe reemplazarlo por otro de mayor capacidad.

NOTA: El PIC 16F628A tiene un espacio disponible de **2048** palabras.
El PIC 16F874 tiene un espacio disponible de **4096** palabras.
El PIC 16F877A tiene un espacio disponible de **8192** palabras.

---

5. Programa del microcontrolador.-

En esta parte es donde se debe escribir el programa, Microcode reconoce palabras clave como **VAR, HIGH, LOW, PAUSE**, etc., y los pinta con mayúsculas y negrillas, por lo que no se debe utilizar estas palabras como nombres de subrutinas o variables.

A continuación se interpreta el significado de cada una de las líneas del programa de la figura 3.2.1, cuyo objetivo es hacer parpadear un led con intervalos de 1 segundo.

Línea 14: led **VAR** portb.0, indica que el Pin # 6 del PIC 16F628A se llamará en adelante led
Línea 15: pepe:, estamos asignando una subrutina con el nombre de pepe y se lo crea escribiendo cualquier nombre seguido de 2 puntos (:) ejemplo:
Luis:, LUIS:, LuIS:, Alarma:, LedApagado:, Zona3:, Contador:.

NOTA: No se debe empezar con números y tampoco debe contener espacios, ejemplos de lo que no se debe hacer:
3pepe: en su lugar escriba pepe3:, pepe 3: el espacio no acepta PICBasic Pro, tampoco acepta pepe3 :, porque hay un espacio entre el 3 y los dos puntos.

Línea 16: **HIGH** led, significa sacar 5 voltios por el pin 6, lo cual encendería el led.

Línea 17: PAUSE 1000, genera una pausa o retardo de 1000 milisegundos, que equivale a 1s.

NOTA: Los **PAUSE** que se puede utilizar es de 1 a 65535, es decir que **PAUSE** 65535, equivale a más de 1 minuto y 5.5 segundos, y **PAUSE** 1 equivale a 0,001 segundo.

Línea 18: **LOW** led, significa poner el pin 6 a un estado bajo o 0 voltios, esto apagaría el led.

Línea 19: PAUSE 1000, como ya se explicó antes genera una espera de 1 seg. sin hacer nada.

Línea 20: GOTO pepe, Como el ingles lo dice **ir a** pepe, indica continuar desde la línea 15, con esto se repetiría el parpadeo del led para siempre.

NOTA: PicBasic Pro ejecuta las instrucciones en orden desde arriba hacia abajo, en el caso del ejercicio anterior desde la línea 14, luego la 15, 16,17,18,19,20, luego de esta última salta a la línea 14 por acción del GOTO pepe, y nuevamente repite el proceso.

Línea 21: **END**, Fin de las instrucciones, sirve para indicarle al compilador pbp que hasta aquí es el programa válido.

---

6. Comentarios.-

Es recomendable usar comentarios todo el tiempo, aunque sea obvio para usted, alguien podría necesitarlo, y por qué no para usted mismo, dentro de un tiempo no recordará ni cómo lo hizo ni cómo funciona, ni para qué servía tal instrucción.

NOTA: Los comentarios se crean anteponiendo un punto y coma (;), noten que el texto cambia de color de negro a azul y del tipo cursiva.
Trate de poner comentarios entendibles por ejemplo:
HIGH portb.3 *;activar el relé, que enciende el MOTOR.*

---

7. Encabezado del programa.-

No son nada más que comentarios en los que se puede incluir: nombre, fecha, autor, y una explicación en breves palabras de cómo y para qué sirve el programa. También se puede hacerlo modificando en **View ---Editor Options---Program header**, aquí coloque el autor y la empresa para que se coloque automáticamente cada que abra una nueva página.

---

8. Compilador.-

Estos 2 botones sirven básicamente para compilar el programa y crear el archivo. ASM, .MAC, y el .HEX, el .HEX sirve para grabar en el micro, el .MAC sólo sirve para el PICBasic y el .ASM, para personas interesadas en ver cómo lo hizo el compilador en assembler ya que podemos abrirlo en **MPLAB**.

 **Compile Only - F9**. Este primer botón sirve para compilar, es decir el programa lo cambia a assembler y lo crea el .HEX, más adelante se verá cómo trabaja.

 **Compile and Program - F10**. Este botón tiene doble función, aparte de hacer lo mismo que el botón anterior, es decir compilar, también puede llamar al programador Ic-prog, con la finalidad de ahorrarnos tiempo y no tener que abrir por separado, es aconsejable utilizarlo una sola vez, y una vez que el programador IC-prog ya está abierto, en adelante sólo se debe utilizar el botón **Compile Only – F9**.

3.3 IDENTIFICACIÓN DE ERRORES EN LA COMPILACIÓN.

En el momento que se compila un programa este realiza una previa verificación del mismo, si existen errores microcode señala el primer error que encuentra con una franja CAFÉ, luego en la parte inferior menciona los demás errores con el número de línea y su explicación, por eso se recomienda activar la opción que muestra el número de línea de programación, (si desea activar esta opción refiérase al capítulo 3 página 26), a continuación un ejemplo de error en la compilación en el que se escribió highh en vez de high.

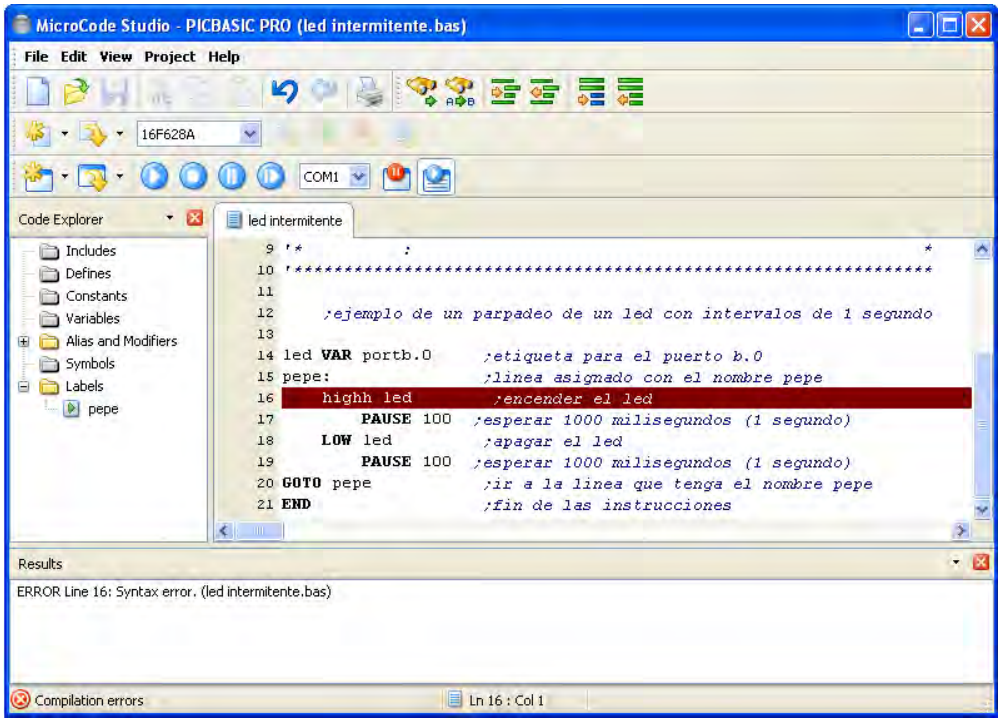


Figura 3.3.1. *Pantalla de error en la compilación*

MENSAJE	EXPLICACIÓN
Syntax error	Error de sintaxis, mal escrito, falta o está demás una letra
Bad expresión	Mala expresión, mal escrito, falta o está demás una letra
ID pep is not a LABEL	La línea pep no es un nivel, o nombre de línea incorrecto
For without a matching next	Cuando falta un next
next without a matching for	Cuando falta un FOR ejem. Fo x = 1 to 12
undefined symbol "portc"	Cuando se pone un Puerto que no dispone el pic
80000 numeric overflow	Exceso del valor límite ejem. **PAUSE** 80000
bad token "."	No se colocó el número del pin 1,2,3. Ejem. **LOW** portb.
bad variable modifier: .O.	Ejem. **LOW** portb.O puso la letra (O) en vez del cero (0)
processor file 12F675	Este error sale en compiladores de versiones antiguas, ya que no dispone de este modelo de PIC por ejemplo en el PBP 2.33
undefined symbol "cncom"	Indica que no existe ese registro en el PIC seleccionado
code crossed boundary @800h	Es una precaución que indica que el programa sobrepasa las 2048 líneas de programación, aunque si compila no es un problema.

Figura 3.3.2. *Tabla de los errores más comunes.*

CAPÍTULO 4
PROGRAMANDO EN LENGUAJE BASIC

4.1 DIFERENCIA ENTRE EL LENGUAJE BASIC Y EL ENSAMBLADOR

Para poder entender la diferencia entre los dos lenguajes de programación, se debe tener en claro qué es un lenguaje de alto nivel y qué es un lenguaje de bajo nivel, a través del siguiente cuadro podemos ver los niveles de programación.

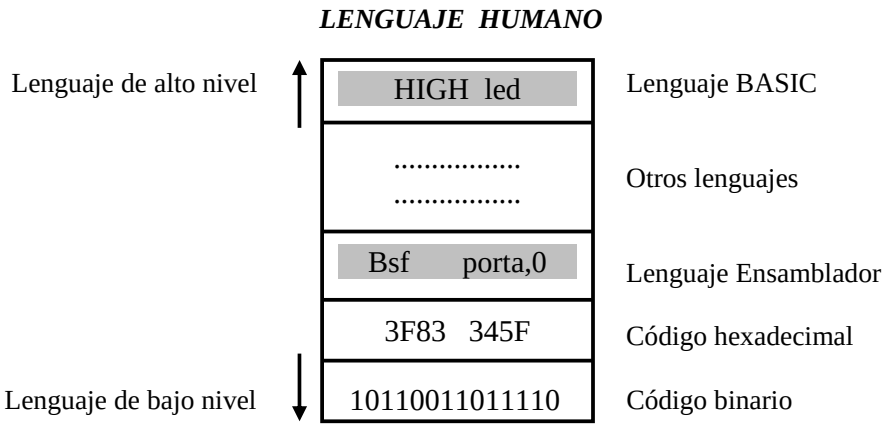


Figura 4.1.1. *Cuadro de los niveles de programación, el lenguaje que más se acerca a los humanos es el de más alto nivel, el lenguaje más próximo al tipo de datos que entiende el microcontrolador es un lenguaje de bajo nivel.*

Por consiguiente vamos a programar con un lenguaje de alto nivel, el que más entendemos los humanos, esta es la gran diferencia entre Ensamblador y BASIC, a continuación un ejemplo de un programa en Basic para el PIC16F628A que hace parpadear un led con intervalos de 1 segundo.

```
inicio:                 ; nombre de subrutina inicio
      HIGH portb.1      ; enciende el led que esta conectado en el pin 7
      PAUSE 1000        ; espera un segundo
      LOW portb.1       ; apaga el led
      PAUSE 1000        ; espera un segundo
GOTO inicio             ; continúa el programa para siempre
```

A continuación el mismo proyecto para el parpadeo del led pero en lenguaje ensamblador.

```
        list p=16F628A

status  equ     03h             ;etiquetamos cada posición de memoria
portb   equ     06h
trisb   equ     86h
cont1   equ     20h             ;etiquetamos cada variable según el lugar que el datasheet
cont2   equ     21h             ; asigne como espacio de memoria RAM
cont3   equ     22h

reset   org     0               ;se escribe en la línea 0 la instrucción
        goto    inicio          ;salta a la línea etiquetada con inicio
        org     5               ;las siguientes líneas se escribirán desde la dirección 5

retardo movlw   D'10'           ;El registro cont1 contiene el número de
        movwf   cont1           ;veces que repite 100 milisegundos
repite1 movlw   D'100'          ;El registro cont2 contiene el número de
        movwf   cont2           ;veces que repite 1 milisegundo
repite2 movlw   D'110'          ;El registro cont3 contiene el número de
        movwf   cont3           ;veces que repite los 9 microsegundos
repite3 nop                     ;de retardo generados
        nop                     ;por los 6 ciclos de las instrucciones nop (6usg)
        nop                     ;más 1 ciclo de la instrucción decfsc (1usg)
        nop                     ;más 2 ciclos del salto goto (2usg)
        nop                     ;dando en total los 9usg, siendo esta la base
        nop                     ;de tiempo, por lo tanto 1sg= 9usg*110*100*10
        decfsz  cont3           ;decrementa el reg cont3 y salta si llega a 0
        goto    repite3         ;si cont3 no es 0 entonces salta a repite3
        decfsz  cont2           ;decrementa el reg cont2 y salta si llega a 0
        goto    repite2         ;si cont2 no es 0 entonces salta a repite2
        decfsz  cont1           ;decrementa el reg cont1 y salta si llega a 0
        goto    repite1         ;si cont1 no es 0 entonces salta a repite1
        retlw   0               ;salida de la subrutina cargando w con 0

inicio  bsf     status,5        ;se ubica en el segundo banco de la RAM
        movlw   00h             ;se carga el registro w con 00h
        movwf   trisb           ;se programa el puerto B como salidas
        bcf     status,5        ;se ubica en el primer banco de la RAM

prog    bsf     portb,1         ;coloca en 1 el pin RB1 para encender el led
        call    retardo         ;Llama a la subrutina retardo de 1 segundo
        bcf     portb,1         ;Coloca en 0 el pin RB1 para apagar el led
        call    retardo         ;Llama a la subrutina retardo de 1 segundo
        goto    prog            ;salta a prog para repetir la secuencia
end
```

Como se puede ver es mucho más largo y difícil de entender, además debe conocer las posiciones de memoria que están disponibles para este PIC, así como también la arquitectura del PIC, también se debe hacer cálculos muy precisos para generar el retardo de 1 segundo.

También hay que considerar el tiempo que se demora en programar en Assembler con el tiempo que se demora en programar en BASIC, las herramientas que nos facilita el compilador de PicBasic Pro son muy útiles y de gran ahorro de trabajo, un ejemplo es la llamada telefónica que sólo con una línea de escritura ya nos genera los tonos DTMF esto es:

DTMFOUT portb.3, [0,9,6,1,3,6,5,6,4] ;genera tonos telefónicos por el pin RB.3

Si bien en esta instrucción se demora 5 segundos en escribir, en ensamblador necesitará más o menos 5 horas para escribir las 260 líneas de programa que se calcula que podría tener, además el trabajo de consultar la frecuencia que genera cada una de las teclas DTMF. En el compilador PicBasic Pro, ya nos facilitan este trabajo, pues los tonos telefónicos están listos sólo hay que llamarlos con la declaración **DTMFOUT** y especificar por cual pin se va a sacar los tonos telefónicos, ejemplos como estos hay muchos.

La única ventaja de programar en Ensamblador es la optimización de espacio en el PIC, si bien es cierto que PicBasic Pro es más fácil y más rápido, necesita más espacio que el que ocuparía al programar en Ensamblador, pero esto ya no es un problema si tomamos en cuenta que cada vez fabrican microcontroladores más baratos y con más espacio de memoria.

4.2 APRENDIENDO A PROGRAMAR EL PIC16F628A CON MICROCODE.

Este es el objetivo primordial de este libro, enseñar a programar micros PIC de la forma más rápida posible, si no ha leído completamente el libro le recomendamos leer el literal 3.2 MANEJO DE MicroCode Studio página 28.

Vamos a proponer nuevamente el proyecto que ya hemos visto antes, pero esta vez lo pondremos en funcionamiento, escriba el programa que viene a continuación, o abra el archivo del CD Ejercicios\ led intermitente.pbp y siga los siguientes pasos. **NOTA *.pbp = *.bas**

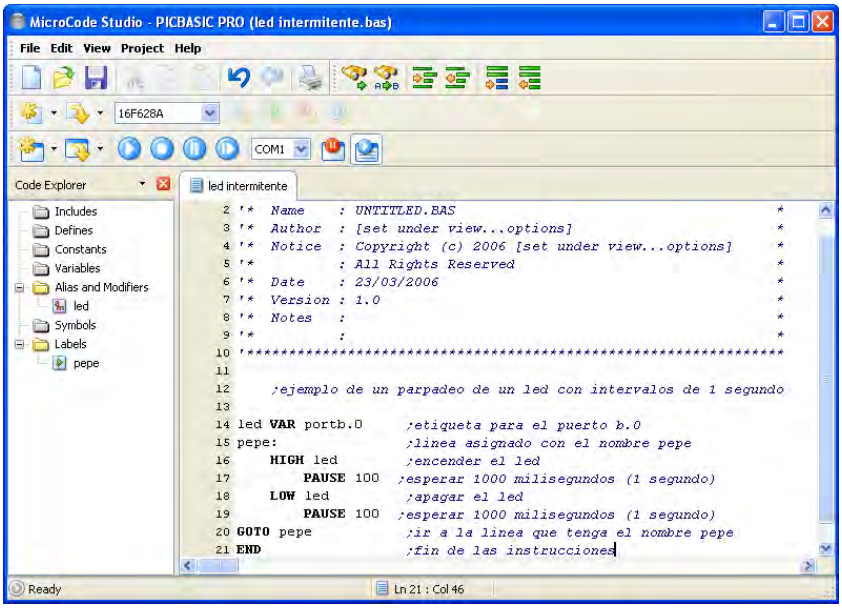


Figura 4.2.1. *Programa microcode con el archivo led intermitente.bas o .pbp*

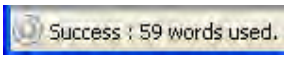
1. Recuerde seleccionar el PIC16F628A.

2. Una vez que esté seguro que el programa está bien escrito presione o **F9**, observe en la parte inferior izquierda como cambia el color del círculo:

Círculo plomo ready, mientras está escribiendo el programa.

Círculo verde mientras está compilando el programa con pbp 2.47.

Círculo amarillo mientras genera el código Assembler y el .Hex.



Círculo plomo finalizar y nos da el tamaño que necesita en el PIC.

Caso contrario si el programa está mal escrito o existen errores en su desarrollo:

Círculo rojo después de compiling indica error en la compilación.

4.3. GRABANDO EL PIC CON EL IC-Prog 1.06A.

Si todo está bien y dice **success : 59 words used.**, presione  o **F10**, espere a que compile nuevamente y se abra el **IC-Prog 1.06A**, si es la primera vez que ejecuta este programa no olvide revisar el Capítulo 1 página 14. A continuación la pantalla de IC-Prog 1.06A.

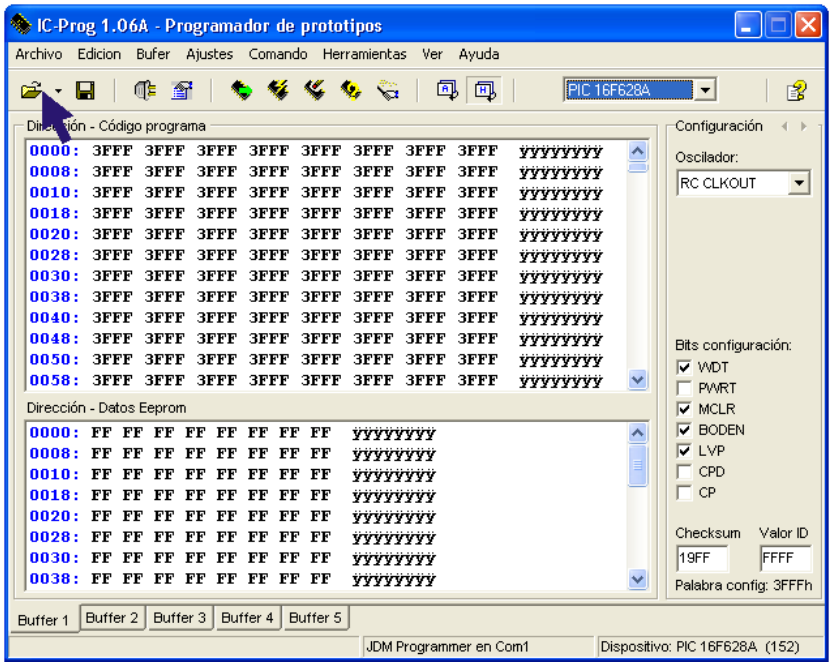


Figura 4.3.1. *Presentación de la pantalla de IC-Prog 1.06A.*

3. Seleccione el PIC que se va a grabar, noten que el código de programa está sólo 3FFF 3FFF esto quiere decir que está vacío, no hay ningún programa a grabarse.

4. Abra Archivo y busque **led intermitente.HEX**, ponga Abrir e inmediatamente verá que el código ha cambiado por algunos números, este es el programa que el PIC entiende, la presentación será similar a la siguiente pantalla:

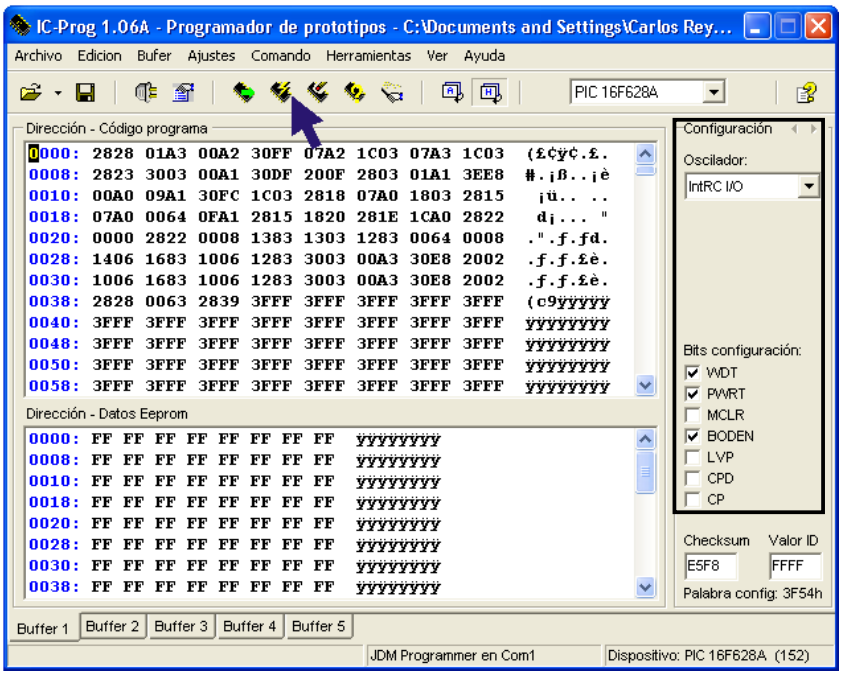


Figura 4.3.2. *Pantalla de IC-Prog con el archivo led intermitente.hex cargado.*

Después de abrir el archivo .HEX (No antes), proceda a cambiar la configuración del oscilador a **intRC I/O** (Oscilador interno resistencia condensador pin de I/O los A6 y A7), el **MCLR** (reset externo) debe estar deshabilitado, y la protección de código apagada. Si ya está listo e instalado el PIC en el Grabador de micros, presione  F5 y espere a que salga el siguiente mensaje:

Figura 4.3.3. *Cuadro de diálogo que indica que el PIC se grabó exitosamente.*

En ocasiones puede salir un mensaje similar al siguiente:

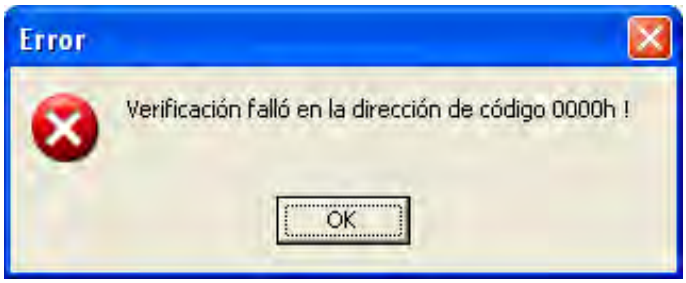


Figura 4.3.4. *Cuadro de diálogo que indica error en la programación del PIC*

Los motivos por los que sale el mensaje error en el código 0000h podrían ser los siguientes:

1. Si el LED rojo del grabador no se encendió mientras estaba programando, revise si está conectado en el puerto com correcto.

2. Si dispone de dos puertos com pruebe cambiando al otro puerto com hasta que se encienda el led rojo del grabador.

NOTA: El LED indicador ROJO del grabador sólo se enciende mientras se está grabando o leyendo un PIC, es posible que el led verde esté encendiéndose, esto no es un problema, simplemente no haga caso.

3. Si el LED rojo del grabador se enciende pero de todas maneras sale el mismo mensaje de error, revise si el PIC está correctamente insertado en el grabador.

4. Si analizado los 3 puntos anteriores, continúa saliendo el mismo mensaje, es muy probable que el PIC se encuentre dañado, reemplace por otro e intente grabar nuevamente.

NOTA: Este mensaje de error también sale cuando se graba un micro con protección en el código de programa, aún cuando el PIC se encuentra en perfectas condiciones, obviamente porque IC-Prog no pudo leer y verificar el contenido del PIC, simplemente en este caso ponga a trabajar el micro y verá que funciona correctamente.

Si ya solucionó el problema y el mensaje es **verificación correcta**, es hora de montar el proyecto y ver funcionar. A continuación conecte como ilustra la siguiente figura.

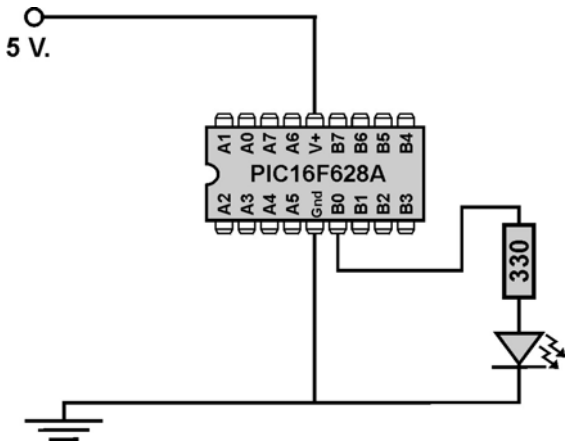


Figura 4.3.5. *Conexión de un LED en el puerto B0 (pin 6)*

NOTA: si decide utilizar algún pin del puerto A, tome en cuenta que estos son análogos y podría observar un funcionamiento defectuoso. Para solucionar esto agregue al principio del programa **cmcon= 7**, esto convierte los pines del puerto A en digitales, un inconveniente también es el puerto A5 este es sólo de entrada, es decir se puede utilizar para un pulsador pero no para encender un LED, otro inconveniente podría ser el puerto A4 este es de colector abierto, necesita conectarse a 5 voltios, como ilustra la siguiente figura.

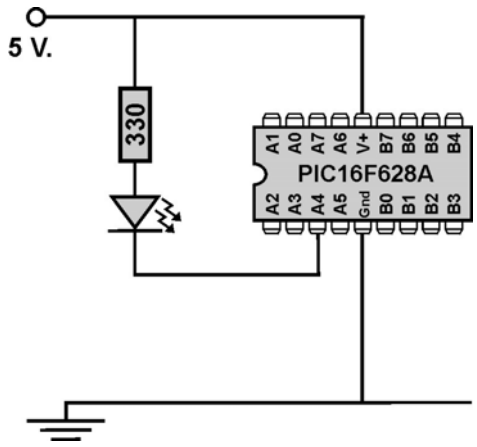


Figura 4.3.6. *Diagrama especial de conexión de un LED en el puerto A4 por ser de colector abierto, su lógica es inversa es decir cuando se pone* ***HIGH*** *se apaga y* ***LOW*** *se enciende*

Para los demás puertos A0, A1, A2, A3, A6, A7, estos funcionan normalmente como el puerto B tanto como para entradas o salidas, siempre que incluya la línea cmcon=7.

```
cmcon = 7          ;apaga los comparadores de voltaje del puerto A y los convierte en digitales
Led VAR porta.0              ; etiqueta asignada al pin 17 o Puerto  RA0
inicio:                      ; nombre de subrutina inicio
        HIGH led             ; enciende el led que está conectado en el pin 17
        PAUSE 1000           ; espera un segundo
        LOW led              ; apaga el led
        PAUSE 1000           ; espera un segundo
GOTO inicio                  ; continúa el programa para siempre
END
```

Figura 4.3.7. *Programa de parpadeo de un LED en el puerto RA0 convirtiendo en digital.*

4.4. DIFERENTES CAMINOS A SEGUIR PARA CONSEGUIR UN MISMO OBJETIVO.

En este literal se pretende aclarar que existe varias formas de desarrollar un programa, se presentarán a continuación varias maneras de escribir un programa que realiza el mismo trabajo final, es decir hacer parpadear un led con intervalos de 1 segundo en el puerto RB0.

```
led VAR portb.0              ; etiqueta asignada al pin 6 o Puerto  RB0
inicio:                      ; nombre de subrutina inicio
        LOW led              ; apaga el led que esta conectado en el pin 6
        PAUSE 1000           ; espera un segundo
        TOGGLE led           ; cambia el estado de portb.0 de on a off o viceversa
        PAUSE 1000           ; espera un segundo
GOTO inicio                  ; continúa el programa para siempre
```

Figura 4.4.1. *Otra forma de programar un parpadeo de un LED a intervalos de 1 segundo utilizando* ***TOGGLE*** *que sirve para invertir el estado de un puerto.*

En el siguiente caso se manejará todo el puerto B como salidas, pero sólo se trabajará con una de ellas el puerto B.0, es importante no olvidar incluir al principio del programa trisb = 0, o trisb=%0 ya que sin este no funciona el manejo de puertos, debe entender que Portb =%00000010 quiere decir encender únicamente el puerto B1 de esta manera tenemos el siguiente ejemplo en donde se explica mejor el manejo del puerto B.

Portb= % 0 1 0 0 1 0 1 0
B7 B6 B5 B4 B3 B2 B1 B0

Indica encender el Puerto B.6, el B.3 y el B.1, para todos los demás significa permanecer apagados, **esto es muy útil en casos en que se necesita encender un grupo de leds**, como los proyectos que más adelante se verá como el semáforo y luces del auto fantástico, de todas maneras lo empleará para hacer parpadear un led.

NOTA: PBP reconoce bases numéricas en decimal, en binario usando el prefijo % y hexadecimal utilizando el prefijo $ ejemplo:
12 es igual que %1100 y también es igual a $C

```
Trisb = 0                   ; indica que todos lo pines del puerto B son de salida
inicio:                     ; nombre de subrutina inicio
        portb = %00000001   ; aunque controla todo el puerto B, sólo enciende el B0
        PAUSE 1000          ; espera un segundo
        portb = %00000000   ; obliga a apagarse a todos los pines del puerto B
        PAUSE 1000          ; espera un segundo
GOTO inicio                 ; continúa el programa para siempre
```

Figura 4.4.2. *Otra forma de programar un parpadeo de un LED a intervalos de 1 segundo manejando el puerto B, noten que es capaz de manipular todos los pines del puerto B desde el B0 que es el primero de la derecha hasta el B7 el último.*

Seguimos con otra forma de programar un parpadeo de un led, esta vez como el ejemplo anterior pero con la diferencia de que sólo manejaremos un pin y no todos en conjunto.

```
Trisb =% 0                  ; indica que sólo el puerto RB.0 es de salida
inicio:                     ; nombre de subrutina inicio
        portb.0 = 1         ; sacar un uno lógico por el puerto RB.0
        PAUSE 1000          ; espera un segundo
        Portb.0 = 0         ; hacer cero lógico el puerto RB.0
        PAUSE 1000          ; espera un segundo
GOTO inicio                 ; continúa con la línea de nombre inicio
```

Figura 4.4.3. *Otra forma de programar un parpadeo de un LED a intervalos de 1 segundo manejando el puerto RB.0 únicamente como salida.*

Como se podrá ver hay distintas formas de escribir un programa y todos tienen el mismo resultado final, así que si un proyecto no funciona correctamente, pruebe escribiendo de una forma diferente.

4.5 DECLARACIONES DISPONIBLES EN EL COMPILADOR PBP 2.47.

Debemos entender que declaraciones son cada una de las palabras que el compilador pbp 2.47 tiene reservado para realizar una tarea específica, las más utilizadas son**: HIGH, LOW, PAUSE, GOSUB, GOTO, LCDOUT, SERIN, SEROUT, FOR, NEXT, IF, THEN, SOUND, END,** un ejemplo:

HIGH portb.3

Esta instrucción **HIGH** es reconocida automáticamente por microcode, lo coloca en negrilla y mayúscula, y sirve para que el compilador realice los ajustes necesarios para cambiarse al segundo banco de la RAM, colocar como salida el puerto B en TRISB, y luego regrese al primer banco de la RAM y setea en 1 al Portb.3, todo esto nos ahorramos gracias al pbp 2.47.

A continuación las 83 instrucciones disponibles con una breve explicación.

DECLARACIÓN	APLICACIÓN
@	Inserta una línea de código ensamblador
ADCIN	Lee el conversor analógico
ASM...ENDASM	Insertar una sección de código ensamblador
BRANCH	GOTO computado (equivale a ON..GOTO)
BRANCHL	BRANCH fuera de página (BRANCH Largo)
BUTTON	Anti-rebote y auto-repetición de entrada en el pin especificado
CALL	Llamada a subrutina de ensamblador
CLEAR	Hace cero todas las variables
CLEARWDT	Hace cero el contador del Watchdog Timer
COUNT	Cuenta el número de pulsos en un pin
DATA	Define el contenido inicial en un chip EEPROM
DEBUG	Señal asincrónica de salida en un pin fijo y baud
DEBUGIN	Señal asincrónica de entrada en un pin fijo y baud
DISABLE	Deshabilita el procesamiento de ON INTERRUPT, ON DEBUG
DISABLE DEBUG	Deshabilita el procesamiento de ON DEBUG
DISABLE INTERRUPT	Deshabilita el procesamiento de ON INTERRUPT
DTMFOUT	Produce tonos telefónicos en un pin
EEPROM	Define el contenido inicial en un chip EEPROM
ENABLE	Habilita el procesamiento de ON INTERRUPT, ON DEBUG
ENABLE DEBUG	Habilita el procesamiento de ON DEBUG
ENABLE INTERRUPT	Habilita el procesamiento de ON INTERRUPT
END	Detiene la ejecución e ingresa en modo de baja potencia
FOR…NEXT	Ejecuta declaraciones en forma repetitiva
FREQOUT	Produce hasta 2 frecuencias en un pin
GOSUB	Llama a una subrutina BASIC en la línea especificada
GOTO	Continua la ejecución en la línea especificada
HIGH	Saca un 1 lógico (5 V.) por un pin
HPWM	Salida de hardware con ancho de pulsos modulados
HSERIN	Entrada serial asincrónica (hardware)
HSEROUT	Salida serial asincrónica (hardware)
I2CREAD	Lee bytes de dispositivos I2C
I2CWRITE	Graba bytes de dispositivos I2C
IF..THEN..ELSE..ENDIF	Ejecuta declaraciones en forma condicional
INPUT	Convierte un pin en entrada
LCDIN	Lee caracteres desde una RAM de un LCD
LCDOUT	Muestra caracteres en un LCD

LET	Asigna el resultado de una expresión a una variable
LOOKDOWN	Busca un valor en una tabla de constantes
LOOKDOWN2	Busca un valor en una tabla de constantes o variables
LOOKUP	Obtiene un valor constante de una tabla
LOOKUP2	Obtiene un valor constante o variable de una tabla
LOW	Hace 0 lógico (0 V.) un pin específico
NAP	Apaga el procesador por un corto período de tiempo
ON DEBUG	Ejecuta un Debug en BASIC
ON INTERRUPT	Ejecuta una subrutina BASIC en un interrupt
OUTPUT	Convierte un pin en salida
OWIN	Entrada de dispositivos one-wire
OWOUT	Salida a dispositivos one-wire
PAUSE	Demora con resolución de 1 milisegundo (mS.)
PAUSEUS	Demora con resolución de 1 microsegundo (uS.)
PEEK	Lee un byte del registro
POKE	Graba un byte en el registro
POT	Lee el potenciómetro en el pin especificado
PULSIN	Mide el ancho de pulso en un pin
PULSOUT	Genera pulso hacia un pin
PWM	Salida modulada en ancho de pulso por un pin especificado
RANDOM	Genera número pseudo-aleatorio
RCTIME	Mide el ancho de pulso en un pin
READ	Lee byte de un chip EEPROM
READCODE	Lee palabra desde un código de memoria
RESUME	Continua la ejecución después de una interrupción
RETURN	Continua en la declaración que sigue al último GOSUB
REVERSE	Convierte un pin de salida en entrada, o uno de entrada en salida
SELECT CASE	Compara una variable con diferentes valores
SERIN	Entrada serial asincrónica (tipo BASIC Stamp 1)
SERIN2	Entrada serial asincrónica (tipo BASIC Stamp 2)
SEROUT	Salida serial asincrónica (tipo BS1)
SEROUT2	Salida serial asincrónica (tipo BS2)
SHIFTIN	Entrada serial sincrónica
SHIFTOUT	Salida serial sincrónica
SLEEP	Apaga el procesador por un período de tiempo
SOUND	Genera un tono o ruido blanco en un pin
STOP	Detiene la ejecución del programa
SWAP	intercambia los valores de dos variables
TOGGLE	Hace salida a un pin y cambia su estado
USBIN	Entrada de USB
USBINIT	Inicializar USB
USBOUT	Salida de USB
WHILE…WEND	Ejecuta declaraciones mientras la condición sea cierta
WRITE	Graba bytes en un chip EEPROM
WRITECODE	Escribe palabra en código de memoria
XIN	Entrada X - 10
XOUT	Salida X - 10

NOTA: si desea más información de cada declaración puede ver la ayuda de microcode en Help topics\ Statement reference, o descarge un manual en español de **WWW.frino.com.ar**.

CAPÍTULO 5
PROYECTOS CON MICROCONTROLADORES PIC

5. PROYECTOS DE APLICACIÓN.

Este es el Capítulo más importante y el más extenso de este libro, los microcontroladores se aprende desarrollando prácticas reales, no hay nada más emocionante y satisfactorio que ver funcionar un proyecto realizado por uno mismo. Al igual que otras carreras la práctica es lo que nos hace mejores, por ejemplo un médico cirujano graduado aprenderá mucho más en la vida real mientras más cirugías realice, una persona que tomó cursos de guitarra aprenderá a tocar cada vez mejor mientras más practique con la guitarra, asimismo nosotros aprenderemos mucho de los PIC'S mientras más proyectos nos propongamos a realizar.

Es importante seguir en orden el avance de los proyectos ya que existen proyectos que requieren de una secuencia de aprendizaje por ejemplo no podrá entender bien cómo funciona el proyecto 5.2.2.Luces del auto fantástico, si no practica el proyecto que explica cómo hacer repeticiones el del literal 5.2.1 Ejercicio con la instrucción FOR NEXT.

Como materiales básicos necesitará un PIC16F628A, un protoboard preferible de 4 regletas, un regulador de voltaje 7805, una fuente de energía y por supuesto tener un *grabador de PIC´S* como el que se incluye en este libro, este es un grabador tipo JDM (Jens Dyekjaer Madsen), muy fácil de utilizar ya que sólo requiere la energía del puerto serial. Para construir este grabador será necesario que primero lea el capítulo 7, donde se enseñará paso a paso cómo ir armando el grabador. El capítulo 6 enseña la simulación del PIC con PROTEUS, este le será muy útil si usted no dispone de materiales para realizar las prácticas.

Adicionalmente para ayuda del lector se incluye en el CD todos los ejercicios que se presentan en este capítulo, tanto en extensión .pbp y .hex. Además si desea utilizar otro modelo de PIC que no sea PIC16F628A, debe considerarse los cambios necesarios para su correcto funcionamiento, por ejemplo para micros que tienen conversores A/D (16F87X,16F81X), se debe reemplazar la línea CMCON=7 por ADCON1=7, (ver literal 5.10.1).Para el PIC16F84A, no se debe incluir ninguna de estas líneas ya que este micro no posee conversores A/D ni tampoco tiene comparadores de voltaje.

NOTA: Para mayor facilidad en la escritura de los programas se ha cambiado el nombre de los pines de los puertos por ejemplo: el pin RB6 se lo llamará únicamente B.6 o simplemente B6, el puerto RB5, se lo llamará B.5 o B5, y así con todos los demás puertos. Otro punto importante que observará en los proyectos es que no hay diagramas de flujos, esto se debe a que no es necesario, porque el programa escrito en sí es como un diagrama de flujo. En esta edición se ha cambiado la extensión de los archivos .bas por .pbp, ambos se puede abrir en cualquier versión de microcode.

5.1 PROYECTOS CON LEDS

5.1.1 PROGRAMA BÁSICO PARA HACER PARPADEAR UN LED CON INTERVALOS DE 1 SEGUNDO.

Este proyecto ya se revisó muchas veces anteriormente, si usted no ha leído todo el libro lea por lo menos la página 29 y 30 y de seguro entenderá el siguiente programa, adicionalmente puede abrir el programa que se encuentra listo en el CD en Ejercicios\ led intermitente.pbp.

```
led VAR portb.0                ; etiqueta para el puerto B.0
pepe:                          ; nombre de subrutina pepe
        HIGH led               ; enciende el led que esta conectado en el pin 6
                PAUSE 1000     ; espera un segundo
        LOW led                ; apaga el led
                PAUSE 1000     ; espera un segundo
GOTO pepe                      ; continúa el programa desde pepe para siempre
END                            ; fin de las instrucciones
```

Figura 5.1.1.1. led intermitente.pbp *Programa para el PIC16F628A que hace parpadear un led.*

Una vez escrito el programa compile y grabe el PIC, si no sabe cómo hacerlo se recomienda leer las páginas 35 a la 38, no olvide poner en el IC-prog oscilador **intRC I/O** y deshabilitar el **MCLR,** luego de que todo esté bien conecte el PIC como ilustra la siguiente figura:

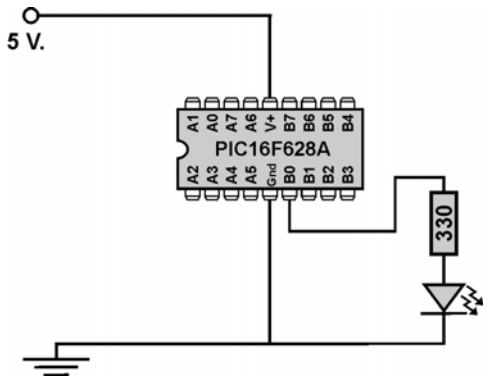


Figura 5.1.1.2. *Diagrama de conexión de un led en el puerto B.0 o pin 6 para hacer un parpadeo de un led.*

Una vez realizado este proyecto siga intentando con diferentes tiempos de **PAUSE**, recuerde que son en milisegundos y sus valores son desde **1 hasta 65535**, pruebe con **PAUSE 100** verá que el parpadeo es más rápido **y PAUSE 2000** es más lento, asimismo ponga de diferentes valores entre los 2 PAUSES ejemplo el primer PAUSE coloque **PAUSE 2000** y en el segundo coloque **PAUSE 500** verá diferentes efectos.

NOTA: recuerde que el PIC ejecuta cada línea de programa en 1 uS. (0,000001 segundos) por lo que si no coloca uno de los 2 PAUSES verá el LED sólo encendido o sólo apagado, esto se debe a que no hay tiempo para ver el efecto de transición del LED.

Para entender mejor pruebe el siguiente programa en el que se eliminó el segundo **PAUSE 1000**:

```
led VAR portb.0              ; etiqueta para el puerto B.0
pepe:                        ; nombre de subrutina pepe
        HIGH led             ; enciende el led que esta conectado en el pin 6
                PAUSE 1000   ; espera un segundo
        LOW led              ; apaga el led
GOTO pepe                    ; continúa el programa desde pepe para siempre
END                          ; fin de las instrucciones
```

Figura 5.1.1.3. *Programa para entender la velocidad a la que trabaja el PIC.*

Si ya colocó en el protoboard verá que el LED permanece sólo encendido, pero no es así el LED se apaga, pero por un período muy corto, lo que a simple vista no lo notamos pues este dura 3 uS., el tiempo en que el PIC cambia a la siguiente instrucción. Analicemos detenidamente lo que hace el PIC desde el momento en que corre la línea **HIGH** led, en ese mismo instante se enciende el LED luego pasa 1 uS. y ejecuta el **PAUSE** 1000, este es un grupo de subrutinas que el compilador pbp genera para dar un retardo de 1 segundo sin hacer nada, por su puesto el LED sigue encendido porque aún no lo decimos que se apague. Una vez terminado el período del **PAUSE** 1000 pasa a **LOW** led en 1 uS., en este mismo instante se apaga el LED, pero la siguiente línea no es otro pause sino ir a pepe y esto se demora 2 uS. por lo que enseguida se enciende el LED al llegar a **HIGH** led.

5.1.2. UN SEMÁFORO DE 2 INTERSECCIONES.

Recuerda que en la página 40 se habla de manejar un grupo de leds, pues bien, este es el ejemplo ideal para entender cuando utilizar HIGH y cuando PORT, se recomienda utilizar HIGH cuando se trata de un sólo led o relé, etc., pero si se va a utilizar un grupo de leds veremos que es mucho más fácil si manejamos todo el puerto sea este el A o el B, pero de todas formas escribiremos el programa de las dos maneras, y usted se darán cuenta cual es la forma más rápida de programar, en la figura 5.1.2.1.se muestra el diagrama de conexión para este proyecto.

MATERIALES.
-además de los materiales básicos, protoboard, regulador 7805 y fuente de voltaje
-6 LEDS, 2 rojos, 2 amarillos, 2 verdes, todos de 5mm.
-6 resistencias de 330Ω a ½ vatio, naranja-naranja-café

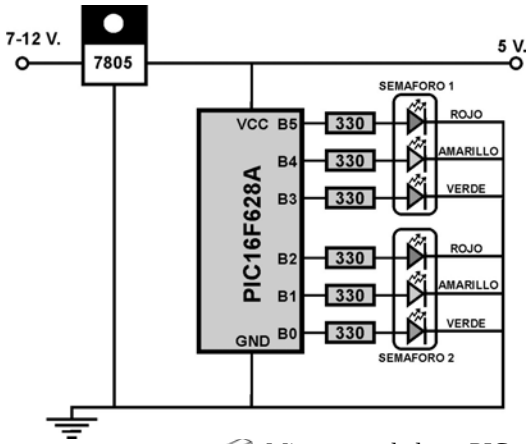


Figura 5.1.2.1 *Diagrama esquemático de conexión para un semáforo de 2 intersecciones.*

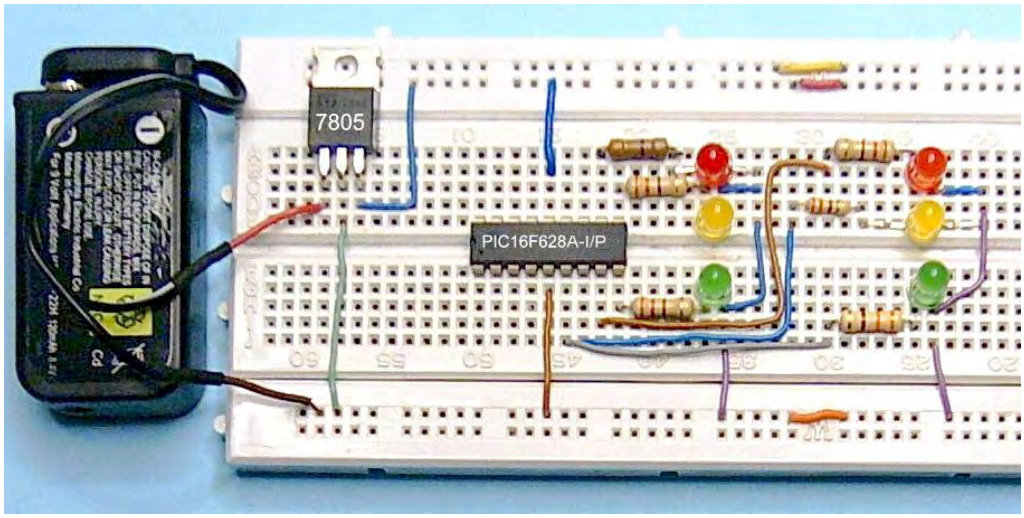


Figura 5.1.2.2. *Fotografía del semáforo armado en un protoboard.*

Bien es momento de escribir el programa, empecemos de la forma que se maneja el puerto completo, para esto se debe entender cómo trabaja los semáforos. Primero nunca se encienden las luces del mismo color, es decir no puede estar en el un semáforo verde y en el otro verde también, existe un cambio de verde a amarillo mientras en el otro semáforo sigue en rojo, en el momento que se pone en rojo el primer semáforo el segundo salta de rojo a verde.

Por considerar que esta es una práctica, se pondrá tiempos estimados de cambio de color, de verde a amarillo durará 9 segundos, de amarillo a rojo sólo 3 segundos.

```
trisb=0                 ;indica que todos los pines del puerto B son de salida
semaforo:               ;nombre de la línea semáforo
        portb=%100001   ;encender rojo del 1er semáforo y verde del 2do semáforo
        PAUSE 9000      ;esperar 9 segundos
        portb=%100010   ;cambiar en el 2do semáforo de verde a amarillo
        PAUSE 3000      ;esperar 3 segundos
        portb=%001100   ;cambiar a verde en el 1er semáforo y rojo el 2do semáforo
        PAUSE 9000      ;esperar 9 segundos
        portb=%010100   ;cambiar en el 1er semáforo de verde a amarillo
        PAUSE 3000      ;esperar 3 segundos
GOTO semaforo           ;continuar con el ciclo para siempre
END                     ; fin de la programación
```

Figura 5.1.2.3. semaforo.pbp *Programa del semáforo manejando el puerto B completo.*

El siguiente es otra forma de escribir el programa, el semáforo funciona igual.

```
rojo1      VAR portb5       ; etiquetas para los puertos
amarillo1  VAR portb.4
verde1     VAR portb3
rojo2      VAR portb2
amarillo2  VAR portb1
verde2     VAR portb.0                                  continúa.....
```

```
semaf:
HIGH rojo1 : HIGH verde2            ;primer semáforo en rojo y 2do en verde
        PAUSE 9000                  ;esperar 9 segundos
LOW verde2 : HIGH amarillo2         ;2do semáf. Pasa de verde a amarillo
        PAUSE 3000                  ;esperar 3 segundos
LOW amarillo2 : LOW rojo1 : HIGH verde1 : HIGH rojo2    ;1er semáf. Verde 2do se. rojo
        PAUSE 9000                  ;esperar 9 segundos
LOW verde1 : HIGH amarillo1         ;1er semáforo cambia de verde a amarillo
        PAUSE 3000                  ;esperar 3 segundos
GOTO semaf                          ;continuar el programa desde semaf
END                                 ;fin de la programación
```

Figura 5.1.2.4. *Programa del semáforo manejando pin por pin con* ***HIGH*** *y* ***LOW****.*

En esta otra manera de escribir el programa, noten que es más largo que el primer programa, y además no escribimos trisb=0 al inicio, porque **HIGH** ya los convierte en salida, también aquí se ve algo nuevo los 2 puntos (:), estos sirven para declaraciones múltiples en una sola línea, para ambos casos el tamaño de código generado es el mismo.
Ejemplo, si queremos expresar en una sola línea las 2 siguientes declaraciones:

HIGH rojo1
HIGH verde2

quedaría así : **HIGH** rojo1 : **HIGH** verde2

NOTA: si desea aplicar este proyecto con focos de 110v., se debe utilizar periféricos de salida como los relés, el siguiente es el diagrama de conexionado de un relé:

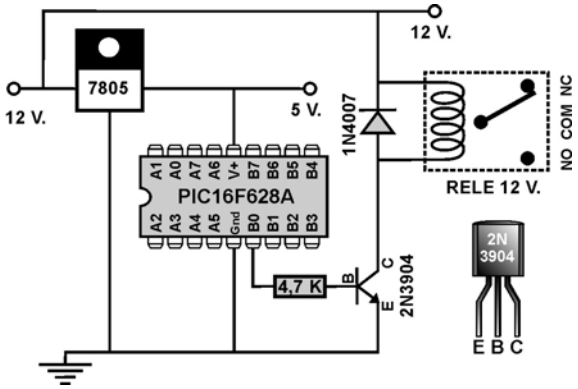


Figura 5.1.2.5. *Diagrama esquemático de Conexión de un relé al PIC.*

Figura 5.1.2.6. *Fotografía de un módulo periférico de salida con un relevo.*

5.1.3. JUEGO DE LUCES PARA DISCOTECA.

Este proyecto propone familiarizar aún más con el manejo de los puertos, esta vez vamos a utilizar las 8 salidas del puerto B, se trata de una secuencia de luces que deben encenderse de

izquierda a derecha una tras otra con un intervalo de 200 milisegundos. En la figura 5.1.3.1. se muestra cómo se debe conectar cada uno de los LEDS.

MATERIALES.
-8 LEDS de 5mm.
-8 resistencias de 330Ω a ½ vatio, naranja-naranja-café

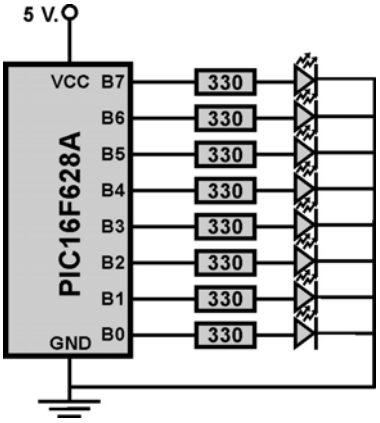


Figura 5.1.3.1. *Diagrama esquemático de conexión de 8 LEDS, para el proyecto de luces de discoteca con PIC. Debido a la capacidad de corriente que puede entregar este PIC, no se necesita de buffers amplificadores.*

A continuación en el siguiente programa veremos una declaración nueva el **GOSUB** y el **RETURN**, estos sirven para cuando se tiene muchas repeticiones de una línea o grupo de líneas de programa, en nuestro caso el **PAUSE** 200, en vez de poner en cada cambio de estado de las luces, lo agruparemos en una sola subrutina y lo llamaremos las veces que queramos, la declaración **RETURN** lo envía de regreso a continuar después del **GOSUB** que lo envió. Una de las ventajas más importantes es que ahorra espacio de memoria en el PIC y otra que si queremos cambiar el **PAUSE** 200 por el de otro valor, basta con cambiar una sola vez y el cambio se ejecuta para todos, lo que al contrario si no lo utilizáramos el **GOSUB** y escribiríamos 30 **PAUSE** 200, deberíamos cambiarlo a los 30 PAUSES por el de otro valor, un ejemplo sería:

```
Prueba1:
        Portb=%00000001
        PAUSE 1000
        Portb=%00000010
        PAUSE 1000
        Portb=%00000100
        PAUSE 1000
GOTO prueba1
```

```
Prueba2:
        Portb=%00000001
        GOSUB pablo
        Portb=%00000010
        GOSUB pablo
        Portb=%00000100
        GOSUB pablo
        GOTO prueba2
Pablo:
        PAUSE 1000
RETURN
```

Figura 5.1.3.2. *Si quisiéramos cambiar el* **PAUSE** 1000 *por* **PAUSE** 500, *en el programa de la izquierda deberíamos cambiar a cada uno de ellos, en total 3, pero para al de la derecha basta con cambiarle al que está dentro del la subrutina pablo, y tendríamos el mismo resultado.*

```
Trisb=%00000000          ;convierte todos los pines del puerto B en salidas
discoteca:               ;nombre de la subrutina
        Portb=%00000001  ;enciende el puerto B.0, los demás permanecen apagados
        GOSUB pedro      ; ir a subrutina pedro y volver cuando diga RETURN
        Portb=%00000010  ;enciende el puerto B.1, los demás les apaga
        GOSUB pedro      ; ir a subrutina tiempo y volver cuando diga RETURN
        Portb=%00000100  ;enciende el puerto B.2, los demás les apaga
        GOSUB pedro      ; ir a subrutina pedro y volver cuando diga RETURN
        Portb=%00001000  ;enciende el puerto B.3, los demás les apaga
        GOSUB pedro      ; ir a subrutina pedro y volver cuando diga RETURN
        Portb=%00010000  ;enciende el puerto B.4, los demás les apaga
        GOSUB pedro      ; ir a subrutina pedro y volver cuando diga RETURN
        Portb=%00100000  ;enciende el puerto B.5, los demás les apaga
        GOSUB pedro      ; ir a subrutina pedro y volver cuando diga RETURN
        Portb=%01000000  ;enciende el puerto B.6, los demás les apaga
        GOSUB pedro      ; ir a subrutina pedro y volver cuando diga RETURN
        Portb=%10000000  ;enciende el puerto B.7, los demás les apaga
        GOSUB pedro      ; ir a subrutina pedro y volver cuando diga RETURN
    GOTO discoteca       ; ir al inicio del programa
pedro:                   ;esta es la subrutina pedro
        PAUSE 200        ;retardo de 200 milisegundos, aquí podemos cambiarlo
        RETURN           ;volver al GOSUB que le envió
```

Figura 5.1.3.3. discoteca.pbp *Programa para las luces de discoteca*.

NOTA: la ubicación de la subrutina pedro es importante fijarse que se encuentre después y fuera de las líneas principales de programación, si esta misma subrutina lo colocáramos al principio del programa, de seguro se nos cuelga porque al encontrar un **RETURN** simplemente no sabe a donde retornar ya que nadie lo ha enviado aún.

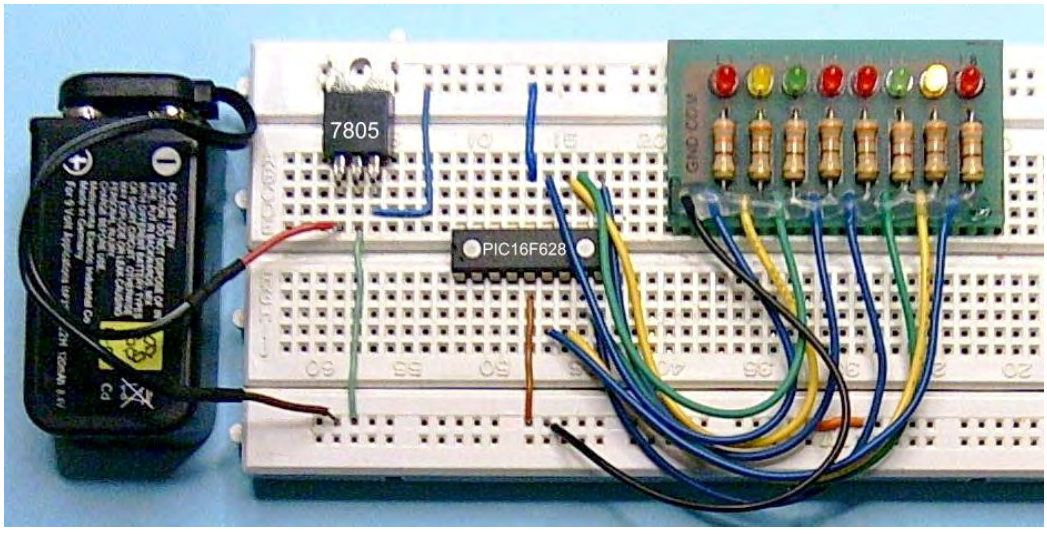


Figura 5.1.3.4. *Fotografía del proyecto de luces para discoteca, se muestra un módulo de 8 leds con sus resistencias, basta con conectar el 1er cable a tierra y los demás a cada uno de los pines del puerto B del PIC, esto ahorra tiempo de instalación.*

5.2 PROYECTOS DE REPETICIONES

5.2.1. EJERCICIO CON LA INSTRUCCIÓN FOR NEXT.

Este proyecto es muy importante entenderlo, ya que el siguiente proyecto de luces de auto fantástico también utiliza la declaración **FOR NEXT**.

Esta declaración sirve para ejecutar un número n veces una línea de programa o grupo de líneas de programa, el siguiente proyecto pretende encender un led en el puerto B.0 5 veces con intervalos de ½ segundo, después debe detenerse por 2 segundos y luego parpadear 3 veces más, detenerse por 3 segundos y luego repetir nuevamente el proceso, se puede utilizar el proto que se armó para las luces de discoteca ya que el mismo nos servirá después para el siguiente proyecto el de las luces del auto fantástico, **FOR NEXT** se utiliza de la siguiente manera:.

```
Peter VAR BYTE      ;primero se crea y asigna un tamaño para la variable peter
FOR peter = 1 TO 5  ;Ejecuta las siguientes instrucciones 5 veces hasta donde dice NEXT
…….                 ;una vez concluido las repeticiones continúa con la declaración que está
…….                 ;después del NEXT, peter debe ser creado como variable, es decir
NEXT                ;asignarle un espacio en la memoria en este caso para 5.
```

LAS VARIABLES BIT, BYTE Y WORD. Estas son creadas para guardar datos en la memoria RAM (Random Access Memory) o memoria de acceso casual, esta memoria trabaja únicamente mientras esté alimentado el PIC, una vez que el PIC es desconectado, los datos de la memoria RAM se borran.

Para crear una variable es muy similar a asignar un nombre de un pin, como peter **VAR** portb.3, la diferencia está en que en vez de poner el pin se pone el tamaño de la memoria a utilizar y estos son los siguientes:

```
Peter VAR BIT       ; crea una variable y asigna un tamaño de un bit es decir  0 o 1
Peter VAR BYTE      ; crea una variable y asigna un tamaño de 8 bits es decir de 0 a 255
Peter VAR WORD      ; crea una variable y asigna un tamaño de 2 bytes es decir de 0 a 65535
```

Para nuestro caso como queremos hacer 5 repeticiones, nos corresponde crear un **BYTE** que nos permite almacenar un número hasta el 255.

```
repe VAR BYTE       ;crea la variable repe y le asigna un espacio de memoria de 0 a 255
Led1 VAR portb.0            ;asigna el nombre de led1 al pin B.0
programa:                   ;nombre de la línea programa
FOR repe = 1 TO 5           ;para repeticiones de 1 a 5 veces
      HIGH led1             ; encender el LED
      PAUSE 500             ; esperar 0,5 segundos
      LOW led1              ; apagar el LED
      PAUSE 500             ; esperar 0,5 segundos
      NEXT                  ; siguiente repetición hasta que sea repe = 5
   PAUSE 2000               ; esperar 2 segundos

FOR repe = 1 TO 3           ;para repeticiones de 1 a 3 veces
      HIGH led1             ; encender el LED                    Continúa .....
```

```
        PAUSE 500           ; esperar 0,5 segundos
        LOW led1            ; apagar el LED
        PAUSE 500           ; esperar 0,5 segundos
        NEXT                ; siguiente repetición hasta que sea repe = 3
    PAUSE 3000              ; esperar 3 segundos
GOTO programa               ; ir a programa
END                         ; fin de la programación
```

Figura 5.2.1.1. repeticiones.pbp *Programa para encender un led n número de veces.*

5.2.2. LUCES DEL AUTO FANTÁSTICO (DESPLAZAMIENTOS).

Este proyecto es muy similar al de las luces para discoteca, con la diferencia de que este se enciende de izquierda a derecha y luego de derecha a izquierda, pensaríamos que el programa sería el doble del tamaño que el que hicimos para las luces de discoteca, pero no es así, recuerden que existen varios caminos para llegar a un mismo objetivo, y este es uno de ellos, esta vez utilizaremos los desplazamientos, que no son nada más que recorrer un uno lógico a la izquierda o a la derecha de la salida de los puertos.

Los desplazamientos utiliza la multiplicación y la división, como sabemos el PIC trabaja con el sistema binario, si tenemos una variable X con un valor inicial de 1 (%00000001) y lo multiplicáramos por 2, el resultado sería 2 (%00000010), y este a su vez lo volveríamos a multiplicar por 2 el resultado sería 4 (%00000100), y así sucesivamente hasta llegar a 128, en donde en binario sería (%10000000), veríamos que los leds se enciende de la misma forma que en las luces para discoteca, para hacer que las luces regresen hasta el puerto B.0 debemos dividir para 2, entonces 128 / 2 es igual a 64 (%01000000), como podemos ver ahora está regresando a su lugar de origen, los desplazamientos se escribe de la siguiente manera:

LEDS = LEDS << 1	equivale a multiplicar por 2 y se desplaza uno a uno, también podemos
LEDS = LEDS << 2	equivale a multiplicar por 4 y se desplaza de dos en dos
LEDS = LEDS >> 1	equivale a dividir por 2 y se desplaza uno a uno hacia la derecha.

Entendido cómo funciona los desplazamientos desarrollamos el siguiente programa:

```
x  VAR    BYTE          ;creamos la variable x y le asignamos tamaño de 255
LEDS  VAR   PORTB       ;todo el puerto B se llamará LEDS
TRISB = 0               ; hacemos salidas a todo el puerto B

        LEDS = 1        ;Cargamos el puerto B con 1 (%00000001)

PROG:
   FOR x = 1 TO 7       ;repetir 7 veces
      LEDS = LEDS << 1  ;desplazar uno a uno a la izquierda
      PAUSE 200         ;esperar 200 milisegundos
      NEXT              ;repetir hasta que x sea = a 7

   FOR x = 1 TO 7       ;repetir 7 veces
      LEDS = LEDS >> 1  ;desplazar uno a uno a la derecha
      PAUSE 200         ;esperar 200 milisegundos                continúa ...
```

```
   NEXT                ;repetir hasta que x sea = a 7
GOTO PROG              ; ir a PROG
    END                ; fin de la programación
```

Figura 5.2.2.1. auto fantastico.pbp *Programa para encender las luces del auto fantástico*.

NOTA: No olviden cargar a LEDS = 1, porque si no lo hace significa que vale cero (0) y esto multiplicado por 2 siempre dará cero, en consecuencia nunca veríamos el desplazamiento.

5.2.3. PROYECTOS PROPUESTOS CON LEDS.

1. Encienda un led conectado en RB4 durante 1,5 seg. y luego apáguelo por 0,5 seg. el proceso debe repetirse sólo 4 veces, luego el led debe permaner apagado.

2. Encienda 2 leds conectados en RB0 y RB1 alternadamente, es decir mientras el un led está encendido, el otro permanece apagado y viceversa, los tiempos de transición son de 700 mls. entre encendido y apagado, el proceso debe continuar indefinidamente.

3. En el proyecto 5.1.3 juego de luces para discoteca, encienda los leds del medio hacia los extremos, es decir empiece por B4 y B3, luego apáguelos y encienda B5 y B2 y así sucesivamente hasta llegar a los extremos B7 y B0, utilice PAUSE 200 y haga que se repita indefinidamente.

4. Encienda una ruleta con leds conectados a todos los pines del micro (15 leds), excepto RA5 y hágalo girar las luces a la velocidad y en el sentido que desee.

5. Genere 6 parpadeos de un led con intervalos de 300 mls. luego haga 2 parpadeos de 1 segundo con un segundo led, luego haga que los 2 leds parpadeen 3 veces, repita el proceso indefinidamente.

5.3 PRÁCTICAS CON PULSADORES

5.3.1. EJERCICIO CON PULSADORES.

Este será el primer contacto del PIC con el mundo exterior, un periférico de entrada, primero se debe entender cómo funcionan los pulsadores con el PIC, existen básicamente 2 tipos de conexión para los pulsadores, el que siempre está en 1 lógico (5 V.) y cuando es pulsado cambia a cero lógico (0 V.), y el que está en cero lógico y cuando se le pulsa pasa a uno lógico, los siguientes son los diagramas de conexión.

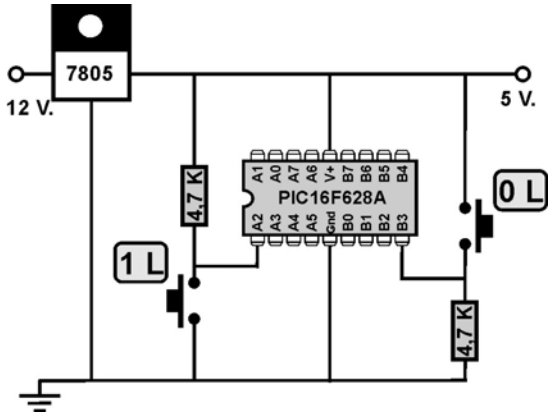


Figura 5.3.1.1. *Diagrama de conexión de 2 pulsadores, el primero es 1 Lógico, es decir siempre permite el ingreso de 5 V. al PIC, cuando es presionado, el voltaje se desvía a tierra y en este caso el PIC detecta un cambio de estado de 1 L a 0 L el funcionamiento del segundo pulsador es totalmente lo contrario.*

Existe otra manera de hacer un pulsador o entrada con más voltaje del que el PIC soporta, y es haciendo un divisor de voltaje, esto es muy utilizado para indicar si una batería de 12 voltios por ejemplo, se encuentra cargada o descargada.

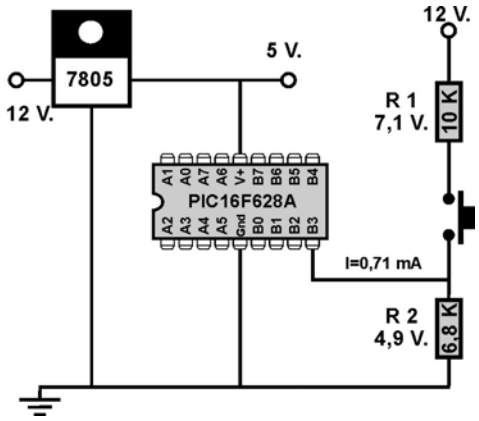


Figura 5.3.1.2. *Diagrama de conexión de un pulsador 0 lógico a una fuente de 12 V. si se le conecta como en el de la figura anterior es muy probable que el PIC se dañe, por esta razón se le conecta en el divisor de voltaje, en donde el voltaje baja a 4,9V. y la corriente que el PIC debe soportar es de 0,71 mA.*

Ejercicio:

Calcular la resistencia R2, para hacer un divisor de voltaje en el que salga aproximadamente 5 Voltios, si la fuente es una batería de 24 Voltios DC.

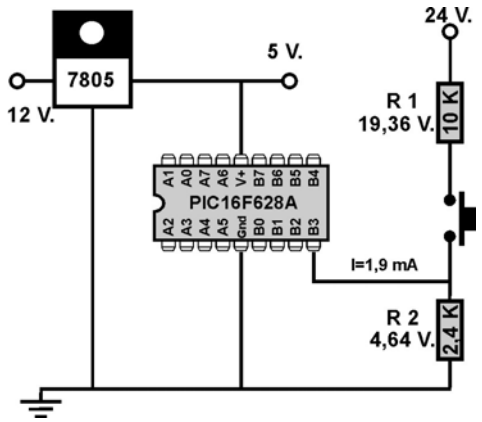


Figura 5.3.1.3. *Diagrama de un pulsador para una fuente de 24 V.*

Explicación.-se necesita un divisor de voltaje de 5 V., por lo que en R1 debería caer 19 V.

V1= $\frac{Vt \times R1}{Req}$ **Req**= $\frac{Vt \times R1}{V1}$ **Req**= $\frac{24 \times 10000}{19}$

Req= 12631,6Ω **R2**=R1-Req **R2= 2631,6 Ω**

Tenemos en el mercado de 2,7 K y 2,4 K, no podemos usar la de 2,7K porque el voltaje pasaría de 5V. utilizaremos la de 2,4 K y tendremos los siguientes cálculos:

V1= $\frac{Vt \times R1}{Req}$ **V1**= $\frac{24V \times 10000\Omega}{12400\Omega}$

V1 = 19,355 V.

V2= $\frac{Vt \times R2}{Req}$ **V2**= $\frac{24V \times 2400\Omega}{12400\Omega}$

V2 = 4,645 V.

La corriente que circula por el PIC sería:

I= $\frac{Vt}{Req}$ **I** = $\frac{24}{12400}$ **I** = 1,93 mA. ;Lo cual estaría bien, considerando que el PIC soporta 25 mA en modo sumidero por cada pin.

LA DECLARACIÓN IF... THEN. Esta sirve de condicionante, si es verdadera ejecuta la operación que sigue al **THEN**, y si es falsa salta a la siguiente línea después del **THEN**, existen varias formas de aplicación:

```
IF portb.0 = 0 THEN Pablo      ;ir a Pablo si la entrada portb.0 es cero lógico
```

```
IF portb.6= 1 THEN juan        ; ir a juan si la entrada portb.6 es 1 lógica
```

```
IF portb.0 =0 THEN             ;si la comparación es verdadera ejecuta todo el contenido que
      HIGH porta.2             ; se encuentra entre el THEN y el ENDIF
      PAUSE 1000
      LOW porta.2
ENDIF
```

```
IF portb3 =0 THEN              ; si la comparación es verdadera ejecuta el contenido entre
      ……                       ;THEN y ELSE y si es falsa ejecuta el contenido entre ELSE
      ELSE                     ;y ENDIF
      …….
      ENDIF
```

```
IF repe > 35 THEN iniciar       ;si la variable rep es mayor que 35 ejecuta iniciar, además se
                                ;puede utilizar los demás operadores:  =, ¡= (NO ES IGUAL),  <,  >,  <=,  >=.
```

```
IF porta.1=0 AND porta.2=0 THEN prog    ;si porta.1 y porta.2 son igual a cero ejecuta prog,
```
asimismo soporta los demás operadores como**: OR, XOR, NOT AND, NOT OR**, **NOT XOR**

El siguiente es el diagrama de conexión para esta práctica de pulsador.

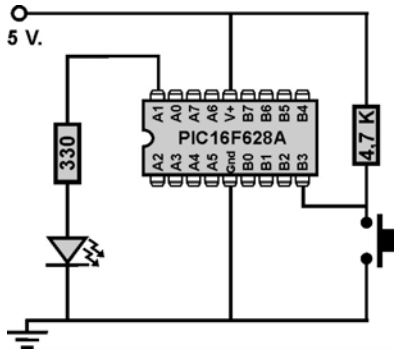


Figura 5.3.1.4. *Diagrama de conexión de un pulsador conectado en el puerto B.3, de estado 1 lógico, en el momento que es presionado este desvía la tención hacia tierra, por lo que el PIC detecta un cambio de estado a cero lógico, en ese instante se enciende el led.*

MATERIALES.
-1 LED de 5mm.
-1 resistencia de 330Ω a ½ vatio, naranja-naranja-café
-1 resistencia de 4,7 KΩ a ½ vatio, amarillo-violeta-rojo
-1 pulsador para protoboard normalmente abierto como el de la figura 5.3.3.2.

A continuación el programa en BASIC para leer el estado de un pulsador.

```
cmcon = 7                       ;convierte todo el puerto A en Digital
pro:
   IF portb.3 =0 THEN encen     ;pregunta si portb.3=0 para ir a encen
   GOTO pro                     ;ir a pro
encen:
   HIGH porta.1                 ;encender el led
   PAUSE 1000                   ;esperar 1 segundo
   LOW porta.1                  ;apagar el led
GOTO pro                        ;ir a pro
END                             ; fin de la programación
```

Figura 5.3.1.5. pulsador.pbp *Programa para leer un pulsador, si este es presionado se enciende un led que está conectado en el puerto A.1 y se apaga después de 1 segundo.*

5.3.2. CONTADOR BINARIO CON PULSADOR ANTIRREBOTE.

En esta práctica haremos un contador binario, el resultado lo veremos en código binario a través de 8 leds conectados en el puerto B. Para esta práctica necesita poner un antirrebote al pulsador,

ya que si presiona por un instante, dada la velocidad que procesa el PIC el programa se ejecutaría varias veces hasta que suelte el pulsador, para aprender de los errores haga el programa sin antirrebote de tecla y luego con antirrebote.

MATERIALES.
-8 LEDS de 5mm.
-8 resistencias de 330Ω a ½ vatio, naranja-naranja-café
-1 resistencia de 4,7 KΩ a ½ vatio, amarillo-violeta-rojo
-1 pulsador para protoboard normalmente abierto como el de la figura 5.3.3.2

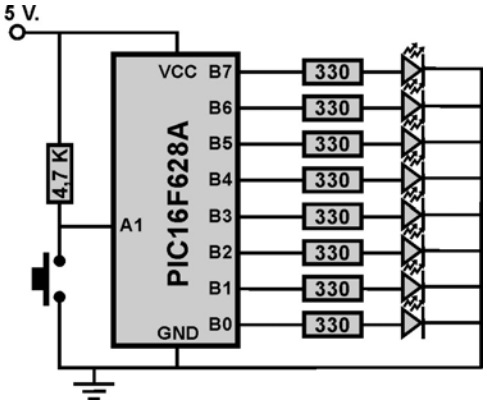


Figura 5.3.2.1. *Diagrama de conexión de 8 leds y un pulsador, cada que se pulsa el botón, los leds aumentan en código binario.*

```
cmcon = 7                       ;convierte todo el puerto A en Digital
Trisb=0                         ; hace todos los pines del puerto B como salidas

boton VAR portA.1               ;asigna el nombre de botón al puerto A.1
num VAR BYTE                    ;crea la variable num con un tamaño de 255
num= 0                          ;carga el valor inicial de 0 a la variable num

pulsar:
        portb=num                       ;indica sacar el valor de num a través del puerto B
        IF boton=0 THEN contar          ;pregunta si el botón ha sido presionado
        GOTO pulsar                     ;ir a pulsar, mantiene encerrado en este loop
contar:
        num=num + 1             ;suma 1 a la variable num y el nuevo valor le guarda en num
GOTO pulsar                     ; volver al principio del programa

END                             ; fin de la programación
```

Figura 5.3.2.2. *Programa para sumar en código binario sin antirrebote de tecla.*

Si tuvo la oportunidad de ver funcionando este proyecto, se dará cuenta que cada que pulsa el botón el contador aumenta demasiado, esto es como se dijo antes porque el PIC trabaja a 1uS. cada instrucción y cuando una persona presiona el botón, por lo menos necesita de 100 mS de tiempo para soltarlo, en ese tiempo el PIC ya sumó alrededor de 25.000 veces. Para solucionar este problema proponemos hacer un programa de antirrebote de tecla, en el cual si presionamos el botón, este le envía a un programa que lo mantiene encerrado, y únicamente sale de esta subrutina

en el momento que el pulsador deja de ser presionado, a continuación incluimos un **PAUSE** 200, que es necesario para que en el momento de soltar la tecla se estabilice la señal.

```
cmcon = 7                    ;convierte todo el puerto A en Digital
Trisb=0                      ; hace todos los pines del puerto B como salidas

boton VAR PORTA.1            ;asigna el nombre de botón al puerto A.1
num VAR BYTE                 ;crea la variable num con un tamaño de 255
num= 0                       ;carga el valor inicial de 0 a la variable num
pulsar:
        portb=num                    ;indica sacar el valor de num a través del puerto B
        IF boton=0 THEN contar       ;pregunta si el botón ha sido presionado
 GOTO pulsar                         ;ir a pulsar, mantiene encerrado
contar:
        IF boton=0 THEN contar       ;espera a que suelte el botón para continuar
        PAUSE 200                    ; espera de 200 mls para estabilizar el botón
        num=num + 1          ;suma 1 a la variable num y el nuevo valor le guarda en num
GOTO pulsar                          ; volver al principio del programa

END                          ; fin de la programación
```

Figura 5.3.2.3. contador binario.pbp *Programa para sumar en código binario con antirrebote.*

Para conocer más sobre los operadores matemáticos disponibles como: resta, multiplicación, división, seno, coseno, etc., existentes en el compilador PICBasic Pro, se recomienda ver la ayuda de microcode en Help Topics\PicBasic Pro Basics\Math operators\introduction o el manual en español del pbp de www.frino.com.ar.

5.3.3. LED INTERMITENTE DE VELOCIDAD VARIABLE.

Este proyecto se basa en dos pulsadores, el 1ro para aumentar la frecuencia del parpadeo del LED, y el 2do para disminuir la frecuencia de parpadeo. Para esto utilizaremos 2 operadores matemáticos, la suma y la resta, la suma incrementará las repeticiones de una instrucción **FOR NEXT**, que contiene un **PAUSE** 5, mientras que la resta disminuirá las repeticiones del mismo pause. Debemos tener en cuenta que una variable **BYTE** no puede exceder su contenido a más de 255, ni tampoco pasar a valores negativos al ser restado consecutivamente, si excediera el valor de la variable a más de 255, el mismo se carga con valor de cero, y viceversa si el resultado de la resta pasara a -5, la variable se carga con 255.

Para comprobar lo dicho anteriormente, después de hacer la práctica principal, en una práctica aparte eliminen o conviértales en comentarios las líneas que dice:

```
; IF veces<10 THEN RETURN
; IF veces>150 THEN RETURN
```

La primera se encarga de no permitir que siga restando, para que el tiempo mínimo de pause sea de 50 milisegundos (10 x **PAUSE** 5), mientras que la segunda se encarga de no permitir que siga sumando si el valor de la variable es mayor que 150, este ejecuta un **RETURN**, con esto el tiempo máximo de pause sería de 750 milisegundos (150 **x PAUSE** 5).

MATERIALES.
-1 LED de 5mm.
-1 resistencias de 330Ω a ½ vatio, naranja-naranja-café
-2 resistencia de 4,7 KΩ a ½ vatio, amarillo-violeta-rojo
-2 pulsadores para protoboard normalmente abierto como los de la figura 5.3.3.2

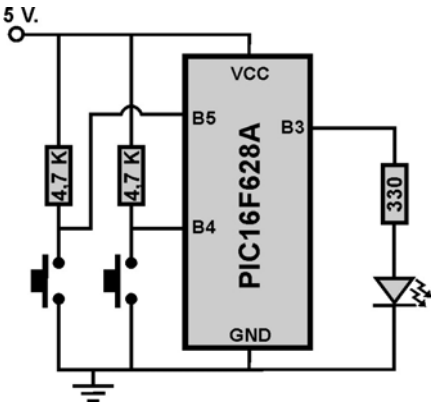


Figura 5.3.3.1. *Diagrama de conexión de 2 pulsadores y un led en el puerto B.*

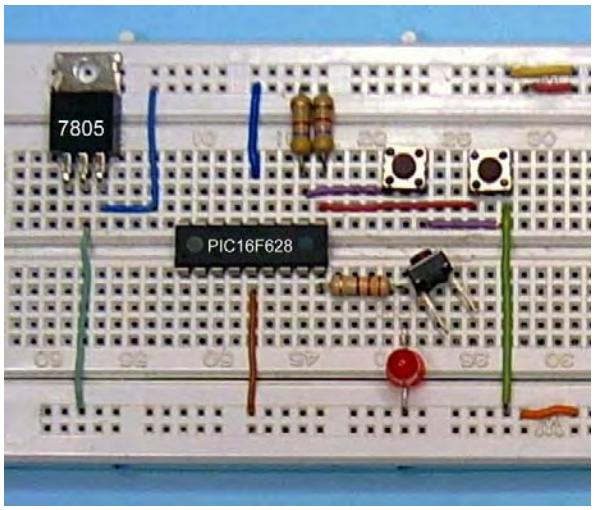


Figura 5.3.3.2. *Fotografía del diseño armado en el protoboard, también muestra el tipo de pulsador de 2 patitas ideal para los protoboards*.

```
Pbaja VAR portb.5      ;el portb.5 se llamará pbaja
Psube VAR portb.4      ;el portb.4 se llamará psube
led VAR portb.3        ;el portb.3 se llamará led
xy VAR byte            ;crea la variable xy con tamaño de 255
veces VAR byte         ;crea la variable veces con tamaño de 255
veces = 100            ;carga con 100 a la variable veces

inicio:
   HIGH led            ;encender el led
   GOSUB timer         ;ir y retornar de timer
    LOW led            ;apaga el led
   GOSUB timer         ;ir y retornar de timer
GOTO inicio

timer:
   IF psube = 0 THEN GOSUB restar      ;pregunta si presionó psube
   IF pbaja = 0 THEN GOSUB sumar       ;pregunta si presionó pbaja
FOR xy = 1 TO veces                    ;repite desde 1 hasta el valor que contenga veces
   pause 5                             ;retardo de 5 mls
   NEXT                                ;siguiente repetición
RETURN                                 ;retornar al que le envió
```

Continúa ....

```
sumar:
IF veces>150 THEN RETURN          ;retorna si veces excede de 150
   veces=veces+5                  ;suma 5 a la variable veces
   RETURN                         ;retorna hacia el que le envió

restar:
   IF veces<10 THEN RETURN        ;retorna si veces es menor que 10
   veces =veces-5                 ; resta 5 a la variable veces
   RETURN                         ;retorna hacia el que le envió


END                               ; fin de la programación
```

Figura 5.3.3.3. led variable.pbp *Programa para el led intermitente de velocidad variable.*

5.3.4. UTILIZANDO EL MCLR (reset externo).

Este es un reset externo que el PIC posee, aparte del reset al encendido que dispone, la utilización del **MCLR** es muy sencillo, sólo debemos instalar un pulsador 1 lógico (pull_up) en el puerto A.5, cuyo pin es específico para el MCLR, el proyecto debe funcionar de la siguiente manera: hacemos un programa para que parpadee un led cada 200 milisegundos (PAUSE 200), para siempre, y al pulsar el botón del MCLR, este parpadeo debe detenerse y al soltarlo debe continuar con el parpadeo del led, es importante tener habilitado el MCLR en el momento de grabar el PIC en el programa IC-prog, el siguiente gráfico muestra cómo debería estar la configuración de los fusibles de configuración del 16F628A en el IC-prog antes de grabar el PIC:

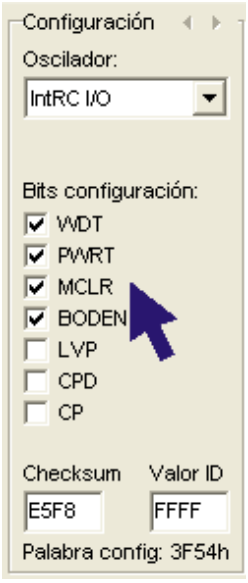


Figura 5.3.4.1. *Configuración del IC-prog, con oscilador Interno RC y habilitado el MCLR (reset externo) necesario para esta práctica.*

MATERIALES.
-1 LED de 5mm.
-1 resistencia de 330Ω a ½ vatio, naranja-naranja-café
-1 resistencia de 4,7 KΩ a ½ vatio, amarillo-violeta-rojo
-1 pulsador para protoboard normalmente abierto

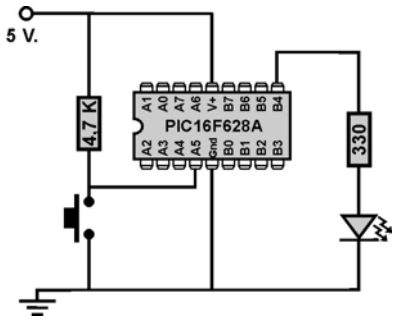


Figura 5.3.4.2. *Esquema de conexión de un LED y un pulsador al MCLR (puerto A.5).*

```
iniciar:                    ; nombre de subrutina iniciar
        HIGH portb.4        ; enciende el led que esta conectado en el pin 10
              PAUSE 200     ; espera  200 milisegundos
        LOW portb.4         ; apaga el led
              PAUSE 200     ; espera  200 milisegundos
GOTO iniciar                ; continúa el programa desde pepe para siempre
END                         ; fin de las instrucciones
```

Figura 5.3.4.3. *Programa para un parpadeo de un led cada 200 mS. con reset externo.*

Con todo lo aprendido hasta aquí usted podría tranquilamente hacer un PLC como el de la siguiente figura:

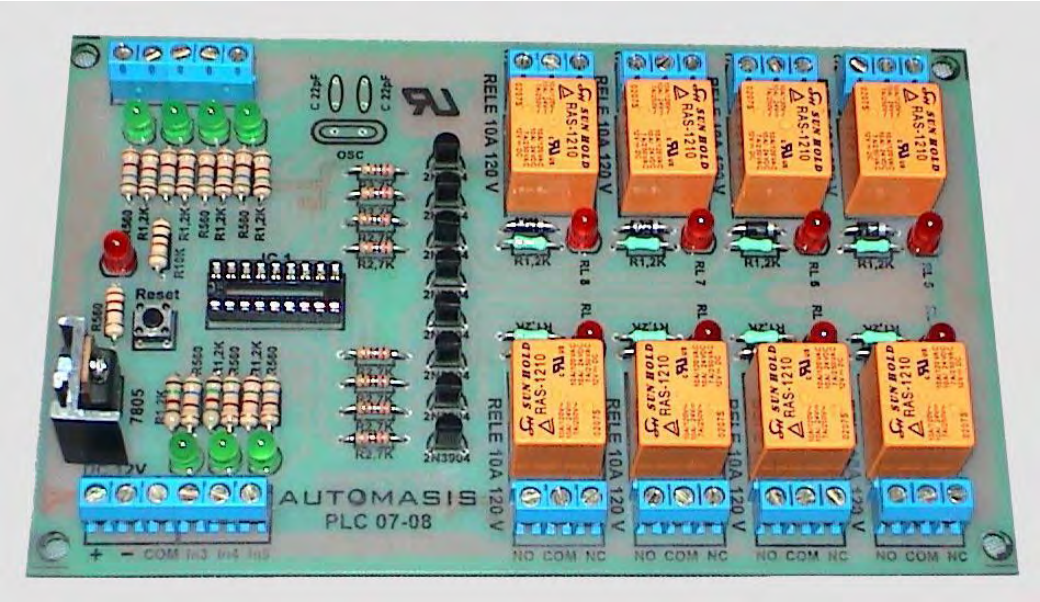


Figura 5.3.4.4. *Fotografía de un PLC de 7 entradas y 8 salidas basado en un PIC16F628A.*

5.3.5. PROYECTOS PROPUESTOS CON PULSADORES.

1. Haga un proyecto en el que al presionar un botón este encienda un led intermitente de 8 repeticiones de 250 mls. Luego el led permanece apagado y el programa vuelve a sensar el pulsador.

2. Con un pulsador haga que 8 leds conectados en el puerto B, se enciendan de derecha a izquierda uno a la vez, empezando de B0 a B7, al final este último permanece encendido, con otro pulsador haga que los leds se desplacen uno a uno hacia la derecha, es decir desde B7 que fue el último y que está actualmente encendido se desplace hasta B0, las pausas son de 300 mls.

3. Haga un proyecto con 2 pulsadores P1 y P2 y 3 leds, led1, led2 y led3, si presiona P1 este hace que se encienda el led1 durante 1 seg. luego este se apaga, si presiona P2, este hace que el led2 se encienda durante 1 seg. y luego se apague, si presiona P1y P2 al mismo tiempo, el led3 parpadea 5 veces con una pausa de 300 mls, luego permanece apagado.

4. Haga un proyecto con 2 pulsadores P1 y P2 y un Led, si presiona P1 y luego P2, el led debe parpadear una sola vez, si presiona P1 3 veces y luego presiona P2, el led debe parpadear 3 veces, y así sucesivamente las veces que presione P1, P2 funciona como arranque del parpadeo.

5. Haga un sistema de clave similar al de CHEVISTAR, esto es utilizando 3 pulsadores, un led rojo y un relé de 12 voltios, si la clave 1223 es presionada correctamente activa el relé, caso contrario se enciende el led rojo por 2 segundos indicando que falló la clave, después de 3 intentos fallidos, el sistema debe bloquearse por 1 minuto, transcurrido ese tiempo puede volver a intentar nuevamente.

5.4 PROYECTOS CON DISPLAYS

5.4.1. MANEJO DE UN DISPLAY DE 7 SEGMENTOS CON EL CI. 7447.

Los displays son muy utilizados para visualizar datos. Para esta práctica se utilizá como periférico de salida un display tipo ánodo común, para lo cual se facilita el diagrama en la figura 5.4.1.2. El proyecto consiste en hacer un contador decimal (0,...,9),con intervalos de 0,5 segundos.

El programa es muy similar al del 5.3.2. contador binario, con la diferencia que sólo se necesita los 4 bits más bajos (B.0, B.1, B.2, y B.3), el decodificador binario a 7 segmentos (7447), es el encargado de transformar el número binario que ingresa a número decimal.

MATERIALES.
-1 DISPLAY ánodo común preferible como el de la figura 5.4.1.2, ideal para protoboards.
-7 resistencias de 330Ω a ½ vatio, naranja-naranja-café
-1 CI. 7447 decodificador BCD

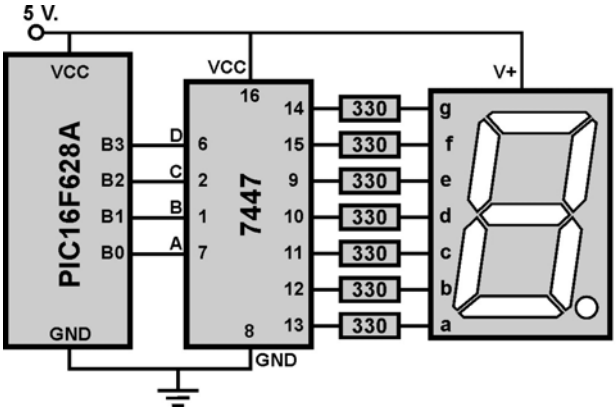


Figura 5.4.1.1. *Diagrama de conexión de un display ánodo común con el BCD 7447*

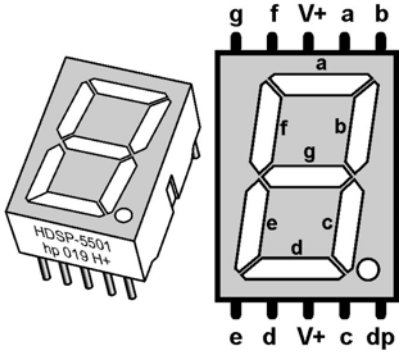


Figura 5.4.1.2.*Esquema de pines de un display a. c.* HDSP-5501

```
trisb=%11110000          ;hace salidas sólo los bits más bajos de Puerto B
numero VAR BYTE          ;crea la variable número con valor 255

encerar:
   numero = 0            ;carga con cero a la variable número
display:
   portb=numero          ;sacar por el puerto b el contenido de número
   PAUSE 500             ;esperar 0,5 segundos
   IF numero=9 THEN encerar  ;si número es =9 encerar número =0
   numero=numero + 1     ;sumar 1 a la variable número
  GOTO display           ;ir a display
END
```

Figura 5.4.1.3. display7seg.pbp *Programa para el display de 7 segmentos.*

5.4.2. UN CONTADOR DECIMAL DE UN DÍGITO CON EL CI. 7447 Y UN PULSADOR.

Adicionando al proyecto anterior un pulsador se hace un contador manual de un dígito. No se olvide hacer un programa antirrebote de tecla, por el caso expuesto anteriormente.

MATERIALES.
-1 DISPLAY ánodo común. preferible como el de la figura 5.4.1.2
-8 resistencias 1 de 4,7KΩ a ½ vatio y 7 de 330Ω a ½ vatio
-1 CI. 7447 decodificador BCD
-1 pulsador para protoboard normalmente abierto como los de la figura 5.3.3.2

El diagrama de este proyecto es el mismo de la figura 5.4.1.1. pero adicionado un pulsador de estado uno lógico normal en el puerto B.4.

```
trisb=%11110000          ;hace salidas sólo los bits más bajos de Puerto B
numero VAR BYTE          ;crea la variable número con valor 255
bot VAR portb.4          ;nombre para el puerto B.4
encerar:
   numero = 0            ;carga con cero a la variable número
display:
   portb=numero          ;sacar por el puerto b el contenido de número
   IF bot=0 THEN aumentar ; si el botón es pulsado ir a aumentar
GOTO display             ;ir a display

aumentar:
   IF bot=0 THEN aumentar        ; si el botón sigue pulsado encerrar en aumentar
   PAUSE 200                     ;esperar 0,2 segundos
   IF numero=9 THEN encerar      ;si número es =9 encerar número =0
   numero=numero + 1             ;sumar 1 a la variable número
  GOTO display                   ;ir a display
END
```

Figura 5.4.2.1. display7seg boton.pbp *Programa para el display de 7 segmentos con pulsador*.

Usted se preguntarán cómo hacer para que el número se incremente apenas se pulsa la tecla y no cuando soltamos como actualmente sucede, pues bien para esto utilizamos banderas que no son nada más que una variable de 1 bit, esta nos indica cuando ha sido pulsada. El siguiente es un ejercicio adicional aplicando la bandera.

```
trisb=%11110000          ;hace salidas sólo los bits más bajos de Puerto B
numero VAR BYTE          ;crea la variable número con valor 255
bot VAR portb.4          ;nombre para el puerto B.4
flag VAR BIT             ;creamos la variable flag de un bit
encerar:
        numero = 0       ;carga con cero a la variable número
display:
        portb=numero                ;sacar por el puerto b el contenido de número
        IF bot=0 THEN aumentar      ; si el botón es pulsado ir a aumentar      continúa .....
```

```
        PAUSE 80                          ; pause para estabilizar el rebote de la tecla
        Flag=0                            ;cargar la variable con cero
    GOTO display                          ;ir a display
aumentar:
        IF  flag = 1 THEN GOTO display    ;pregunta si la variable es uno
        flag = 1                          ;cargar la variable con uno
        IF numero=9 THEN encerar          ;si número es =9 encerar número =0
        numero=numero + 1                 ;sumar 1 a la variable número
    GOTO display                          ;ir a display
END
```

Figura 5.4.2.2. display7seg boton2.pbp *Programa para el display de 7 segmentos con pulsador y utilizando una bandera que indica si se ha presionado la tecla.*

5.4.3. MANEJO DE UN DISPLAY DE 7 SEGMENTOS SIN EL CI. 7447.

Como se dijo en un comienzo al PIC se le puede programar para reemplazar a casi cualquier circuito integrado, en esta ocasión haremos que el propio PIC sea como un CI. 7447, para esto debemos saber que para sacar el número 3 por ejemplo, debemos calcular el número decimal que hace que se enciendan los segmentos correctos del display, esto se hace de la siguiente forma.

Como debemos encender los segmentos a, b, c, d, y g., revisamos los pines del PIC que les corresponde y estos son: B0, B1, B2, B3, Y B6, respectivamente, estos debemos ponerlos en estado cero lógico para que los segmentos se enciendan (recuerde que el display es ánodo común), y los demás 1 lógico para que permanezcan apagados:

B6	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**
0	1	1	0	0	0	0

equivale al decimal 48, el display saca el número 3

LA DECLARACIÓN LOOKUP. Sirve para obtener un valor constante de una tabla, esto lo hace según el número de veces que repita el **FOR NEXT**, por ejemplo: la 1ra vez toma el dato que se encuentra en el lugar 0, es decir el Nro. 64, la segunda el dato del lugar 1el Nro.121, así sucesivamente, y lo va guardando en la variable dat.

MATERIALES.
-1 DISPLAY ánodo común. preferible como el de la figura 5.4.1.2
-7 resistencias de 330Ω a ½ vatio, naranja-naranja-café

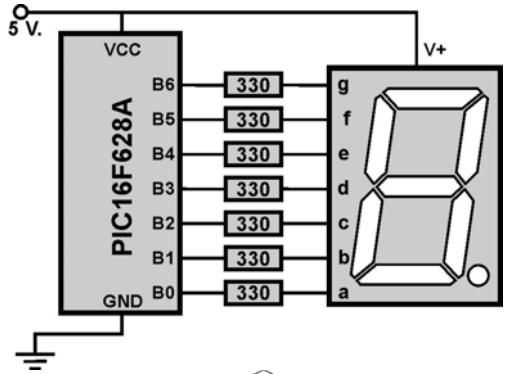


Figura 5.4.3.1. *Diagrama de conexión de un display de 7 segmentos directamente al PIC.*

```
di   VAR BYTE                    ;crea variable di
dat VAR BYTE                     ;crea variable dat
TRISB = 0                        ;todo el puerto B como salida
prog:
   FOR di=0 TO 15                ;para repeticiones de 0 a 15
   LOOKUP di,[64,121,36,48,25,18,2,120,0,16,8,3,70,33,6,14],dat    ;toma uno por uno cada
                                 ; valor de la tabla constante y lo guarda en la variable dat
   portb=dat                     ;sacar el contenido de dat por el puerto B
   PAUSE 500                     ;espera de 0,5 seg.
   NEXT di                       ;siguiente repetición
GOTO prog
END
```

Figura 5.4.3.2. display7seg lookup.pbp *Programa para manejar el display directamente*.

5.4.4. MANEJO DE 4 DISPLAYS DE 7 SEGMENTOS CON EL CI. 7447.

IMPORTANTE: Es muy probable que después de realizar esta práctica el micro no permita cargar un programa nuevo, mostrará un mensaje de **Verificación falló en la dirección de código 0000h**, por lo que en las prácticas **5.4.4** a la **5.4.8** es indispensable activar el MCLR y colocar por lo menos una resistencia de 4,7 KΩ conectado entre puertoA.5 y 5V.,ver pág 59 (el pulsador es opcional), esto permitirá borrar el programa para así poder cargar uno nuevo. La razón de este problema es que el grabador que incluye este libro mantiene al micro alimentado con 5 V., por lo que el programa sigue corriendo aún colocado en el grabador, esto impide el ingreso de la señal de reloj específicamente en el puerto B.6. *Si olvidara activar el MCLR, la única manera de borrar el PIC sería utilizando un grabador de puerto paralelo.*

El siguiente proyecto debe encender 4 displays para poder mostrar cualquier número desde el 0 hasta el 9999, esto lo conseguimos gracias al transistor tipo PNP, que nos ayudará a multiplexar cada uno de los displays, el funcionamiento es bastante sencillo, debemos conectar los 4 bits más altos a cada transistor y los cuatro bits más bajos al CI. 7447, si por ejemplo queremos sacar el número 6874, primero habilitamos el 4to transistor, el de la derecha y enviamos el número 4, el CI. 7447 se encarga de formar el 4 en el display, luego pasamos a cero lógico el 2do transistor, y los demás lo mantenemos en nivel alto, al mismo tiempo sacamos el número 7 por los bits menos significativos del puerto B, y así consecutivamente, el tiempo que debemos mantener activado cada transistor no puede ser mayor que 5 milisegundos, es decir que los cambios son tan rápidos que el ojo humano ve todos los displays encendidos al mismo tiempo, cuando en realidad sólo se enciende uno a la vez.

Ejemplo: para sacar el Nro 8 en las centenas debemos sacar (176+8), es decir el número 184 porque si analizamos en código binario, tenemos que los bits más bajos entran al CI. 7447, y los bits más altos, son los encargados de encender el display que le corresponde a las centenas.

184 = % 10111000 — 1011 1000 — Este número entra al 7447 el cual saca el 8

Este habilita el 3er transistor, el de las centenas

MATERIALES.
-4 DISPLAYS ánodo común
-4 transistores 2N3906
-11 resistencias 7 de 330Ω a ½ vatio, y 4 de 4,7 KΩ a ½ vatio
-1 CI. 7447 decodificador BCD.

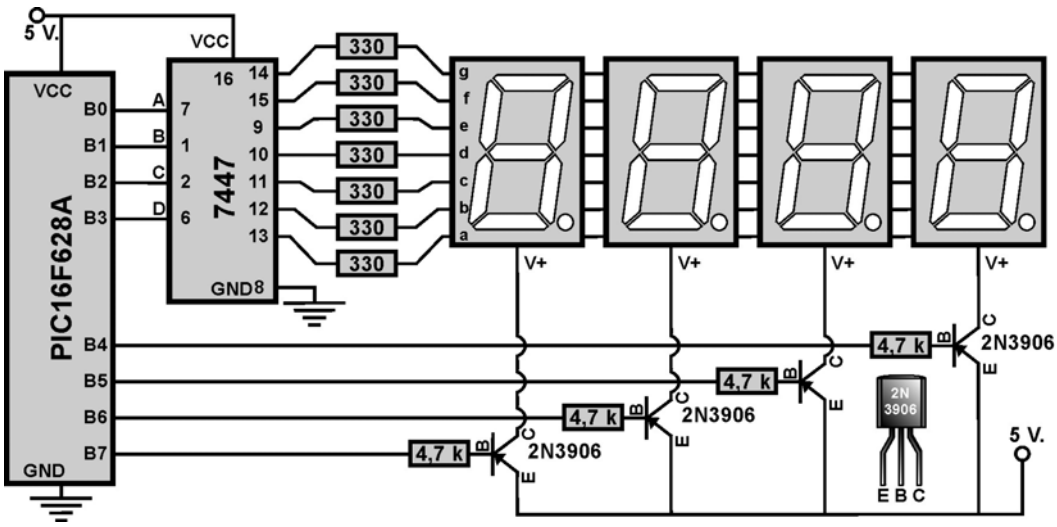


Figura 5.4.4.1. *Configuración para manejar 4 displays de 7 segmentos con el CI. 7447.*

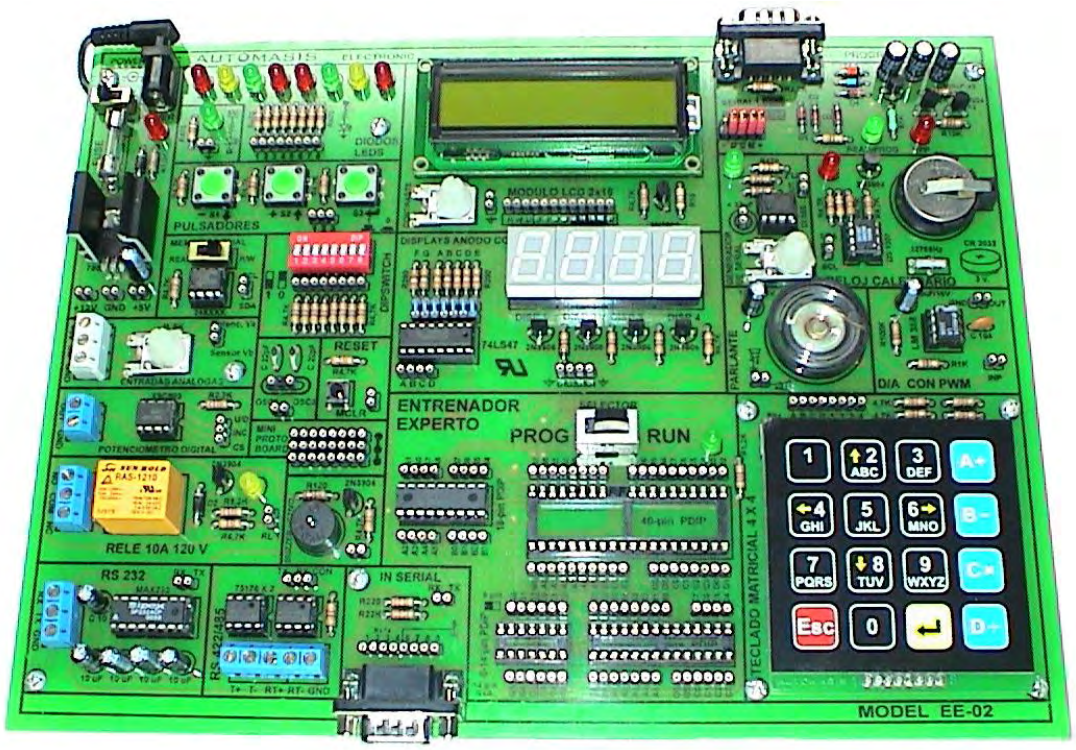


Figura 5.4.4.2. *Fotografía del entrenador experto* EE-02 *de* **AUTOMASIS**, *ideal para estas prácticas por la facilidad en el ensamblaje de proyectos, además dispone de módulos como leds, 4 display de 7 segmentos, LCD, pulsadores, teclado matricial, relé, parlante, dipswitch, comunicación RS232, RS422/485, memoria serial, reloj calendario, chicharra, conversor D/A, etc, también permite programar el PIC sin necesidad de desmontarlo de su zócalo.*

```
trisb=0                 ; convierte en salida todo el puerto B
display:
   portb=224+8          ;%11100000,activa el transistor de las unidades y presenta el 8
  PAUSE 5
   portb=208+7          ;%11010000,activa el transistor de las decenas y presenta el 7
  PAUSE 5
   portb=176+6          ;%10110000,activa el transistor de las centenas y presenta el 6
  PAUSE 5
   portb=112+5          ;%01110000,activa el transistor de los millares y presenta el 5
  PAUSE 5
  GOTO display          ; encierra en este lazo
END
```

Figura 5.4.4.3. numero 5678.pbp *Programa para manejar 4 displays.*

Como experimento para comprobar que el PIC sólo está activando un display a la vez, cambie todos los pauses a **PAUSE** 150.

5.4.5. CONTADOR DECIMAL DE 4 DÍGITOS CON EL CI. 7447.

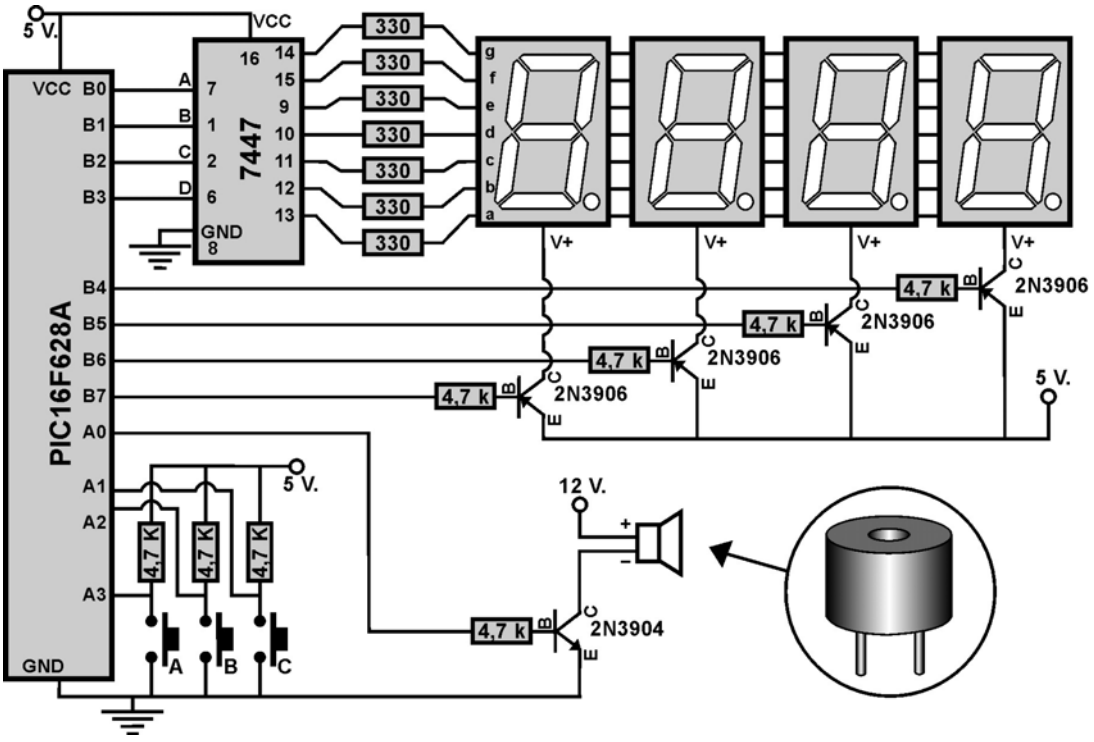


Figura 5.4.5.1. *Esquema de conexionado del contador de 4 dígitos, adicionalmente se muestra la forma de la chicharra activa a 12V de referencia* ***GS1212****.*

Es hora de hacer un proyecto de considerable tamaño, una vez entendido cómo multiplexar 4 displays, y entendido el ejercicio del contador con una bandera de activado el del 5.4.2.2. pues el siguiente proyecto consta en hacer un contador decimal que incremente su valor cada vez que se pulsa el botón A, si pulsamos el botón B se encera y apaga la chicharra, y si pulsamos la tecla C, presenta el número al cual va a comparar, si el número de conteo es igual a 24, activa un aviso auditivo (buzzer activo), este buzzer trabaja a 12 voltios, lo que le diferencia de los parlantes comunes es que no necesita ser activado con una frecuencia, sino basta con alimentarle con 12 voltios para que suene, también lo conocen con el nombre de chicharra a 12 voltios.

MATERIALES.
-4 DISPLAYS ánodo común
-4 transistores 2N3906
-1 transistor 2N3904
-7 resistencias de 330Ω a ½ vatio, naranja-naranja-café
-4 resistencias de 4,7 KΩ a ½ vatio, amarillo-violeta-rojo
-1 CI. 7447 decodificador BCD
-2 pulsadores normalmente abiertos
-1 chicharra de 12 V. como el de la figura 5.4.5.1.

El siguiente es el programa para controlar los 4 displays, contar, encerar, comparar y visualizar el número almacenado en la memoria.

```
unid    VAR BYTE          ;variable unidades
dece    VAR BYTE          ;variable decenas
cent    VAR BYTE          ;variable centenas
mile    VAR BYTE          ;variable miles

setunid   VAR BYTE        ;variable setunidades
setdece   VAR BYTE        ;variable setdecenas
setcent   VAR BYTE        ;variable setcentenas
setmile   VAR BYTE        ;variable setmiles

setunid=4             ;# que queremos que nos de aviso 0024 podemos cambiarlo
setdece=2             ;# que queremos que nos de aviso 0024 podemos cambiarlo
setcent=0             ;# que queremos que nos de aviso 0024 podemos cambiarlo
setmile=0             ;# que queremos que nos de aviso 0024 podemos cambiarlo

chicha  VAR porta.0   ;pin de la chicharra
contar  VAR porta.3   ;pulsos para contar
encera  VAR porta.2   ;tecla encerar
visual  VAR porta.1   ;visualizar el valor a comparar
activar VAR BIT       ;bandera para la tecla contar

trisb=0               ;todo el puerto b es de salida
cmcon=7               ;Todo el puerto A en modo digital

  HIGH chicha     ;encendemos la chicharra para asegurarnos que el PIC está funcionando
  PAUSE 200
```

continúa .....

```
encerar:
   unid=0                          ;carga la variable unid con cero
   dece=0                          ;carga la variable dece con cero
   cent=0                          ;carga la variable cent con cero
   mile=0                          ;carga la variable mile con cero
   LOW chicha                      ;apagar la chicharra
display:
   portb= 224+unid                 ;224 %11100000, activa las unidades
   PAUSE 5
   portb= 208+dece                 ;208 %11010000, activa las decenas
   PAUSE 5
   portb= 176+cent                 ;176 %10110000, activa las centenas
   PAUSE 5
   portb= 112+mile                 ;112 %01110000, activa los miles
   PAUSE 5
   GOSUB teclas                    ;revisar el estado de las teclas
        GOTO display
teclas:
   IF contar=0 THEN sumar          ;si presionan tecla A ir a sumar
   IF encera=0 THEN encerar        ;si presionan tecla B ir a encerar
   IF visual=0 THEN visualizar     ;si presionan tecla C ir a visualizar
   activar=1                       ;bandera de tecla A evita que cuente + de 1 vez
        RETURN
sumar:
   IF activar=0 THEN RETURN        ; bandera de tecla A
   activar=0                       ;bandera de tecla A cuando ya ha sido pulsada
   unid=unid+1                     ;sumar 1 a las unidades
   IF unid<10 THEN comparar        ;si unid es menor a 10 comparar
   unid=0                          ;hace cero a las unidades
   dece=dece+1                     ;y incrementa las decenas
   IF dece<10 THEN comparar
   dece=0
   cent=cent+1
   IF cent<10 THEN comparar
   cent=0
   mile=mile+1
   IF mile<10 THEN comparar
   mile=0
RETURN                             ;retornar a gosub teclas
visualizar:
   portb= 224+setunid              ;224 %11100000,activa las unidades
   PAUSE 15
   portb= 208+setdece              ;208 %11010000,activa las decenas
   PAUSE 15
   portb= 176+setcent              ;176 %10110000,activa las centenas
   PAUSE 15
   portb= 112+setmile              ;112 %01110000,activa los miles
   PAUSE 15
   IF visual=0 THEN visualizar
   RETURN                          ;retornar a gosub teclas
```

continúa .....

```
comparar:
   IF unid!=setunid THEN RETURN          ;si unid no es igual a setunid
   IF dece!=setdece THEN RETURN          ;si dece no es igual a setdece
   IF cent!=setcent THEN RETURN
   IF mile!=setmile THEN RETURN

   HIGH chicha                           ;encender la chicharra
   RETURN                                ;retornar a gosub teclas
END
```

Figura 5.4.5.2. contador 1-9999.pbp *Programa para el contador decimal desde 1 a 9999.*

A continuación otra manera de escribir el mismo programa aún más corto, esta vez utilizando una variable con capacidad de almacenamiento de 65535 (número **VAR WORD**), y para poder tomar cada dígito de esta variable y mostrar en cada uno de los displays, utilizamos el operador matemático **DIG**, que sirve para tomar cualquier dígito que necesitemos de una variable, ejemplo:

Para tomar las decenas de la siguiente variable número:
número = 6789
cent= número **DIG** 1 , como el dígito 0 es el 9, el dígito 1 es el 8, el dígito 2 es 7, y así sucesivamente, en este caso el 8 se almacena en la variable cent.

```
numero  VAR WORD          ;creamos numero para almacenar el conteo
compara VAR WORD          ;creamos compara para guardar el límite
unid    VAR BYTE          ;variable unidades
dece    VAR BYTE          ;variable decenas
cent    VAR BYTE          ;variable centenas
mile    VAR BYTE          ;variable miles
chicha  VAR porta.0       ;pin de la chicharra
contar  VAR porta.3       ;pulsos para contar
encera  VAR porta.2       ;tecla encerar
visual  VAR porta.1       ;visualizar el valor a comparar
activar VAR BIT           ;bandera para la tecla contar

trisb=0                   ;todo el puerto b es de salida
cmcon=7                   ;Todo el puerto A en modo digital
   HIGH chicha            ;encendemos la chicharra para asegurarnos que
   PAUSE 200              ;el PIC está funcionando
   compara=12             ;el número al cual va a comparar es el 12

encerar:
   numero=0               ;carga la variable número con cero
   LOW chicha             ;apagar la chicharra
display:
   unid=numero DIG 0      ;toma el dígito 0 (unidades) y guarda en unid
   dece=numero DIG 1      ;toma las decenas y lo guarda en dece
   cent=numero DIG 2      ;toma el dígito 2 (centenas) y lo guarda en cent
   mile=numero DIG 3      ;toma el dígito 3 (miles)y lo guarda en mile
                                                          continúa .....
```

```
   portb= 224+unid                  ;224 %11100000,activa las unidades
   PAUSE 5
   portb= 208+dece                  ;208 %11010000,activa las decenas
   PAUSE 5
   portb= 176+cent                  ;176 %10110000,activa las centenas
   PAUSE 5
   portb= 112+mile                  ;112 %01110000,activa los miles
   PAUSE 5
   GOSUB teclas                     ;revisar el estado de las teclas
GOTO display

teclas:
   IF contar=0 THEN sumar           ;si presionan tecla A ir a sumar
   IF encera=0 THEN encerar         ;si presionan tecla B ir a encerar
   IF visual=0 THEN visualizar      ;si presionan tecla C ir a visualizar
   activar=1                        ;bandera de tecla A evita que cuente + de 1 vez
    RETURN
sumar:
   IF activar=0 THEN RETURN                ;bandera de tecla A
   activar=0                               ;bandera de tecla A cuando ya ha sido pulsada
   numero=numero+1                         ;sumar 1
   IF numero=compara THEN HIGH chicha      ;si número es = compara
   IF numero>9999 THEN encerar             ;si el número es >9999 ir a encerar
   RETURN                                  ;retornar a gosub teclas

visualizar:
   unid=compara DIG 0               ;toma el dígito 0 (unidades) y guarda en unid
   dece=compara DIG 1               ;toma el dígito 1 (decenas) y lo guarda en dece
   cent=compara DIG 2               ;toma las centenas y lo guarda en cent
   mile=compara DIG 3               ;toma el dígito 3 (miles) y guarda en mile
         portb= 224+unid                   ;224 %11100000,activa las unidades
         PAUSE 15
         portb= 208+dece                   ;208 %11010000,activa las decenas
         PAUSE 15
         portb= 176+cent                   ;176 %10110000,activa las centenas
         PAUSE 15
         portb= 112+mile                   ;112 %01110000,activa los miles
        PAUSE 15
        IF visual=0 THEN visualizar
        RETURN                             ;retornar a gosub teclas
END
```

Figura 5.4.5.3. contador2 1-9999.pbp *Programa para el contador decimal desde 1 a 9999 utilizando el operador matemático DIG.*

5.4.6. MANEJO DE 4 DISPLAYS DE 7 SEGMENTOS SIN EL CI. 7447 (ROTULACIÓN).

Es posible hacer un contador decimal igual que el del proyecto anterior y sin ayuda del CI. 7447,

pero en esta ocasión haremos algo más que eso, como vieron la ventaja de conectar el display directamente al PIC es la de poder sacar casi la mayoría de las letras del alfabeto, pues bien este proyecto consiste en sacar la palabra HOLA a través de los 4 displays.

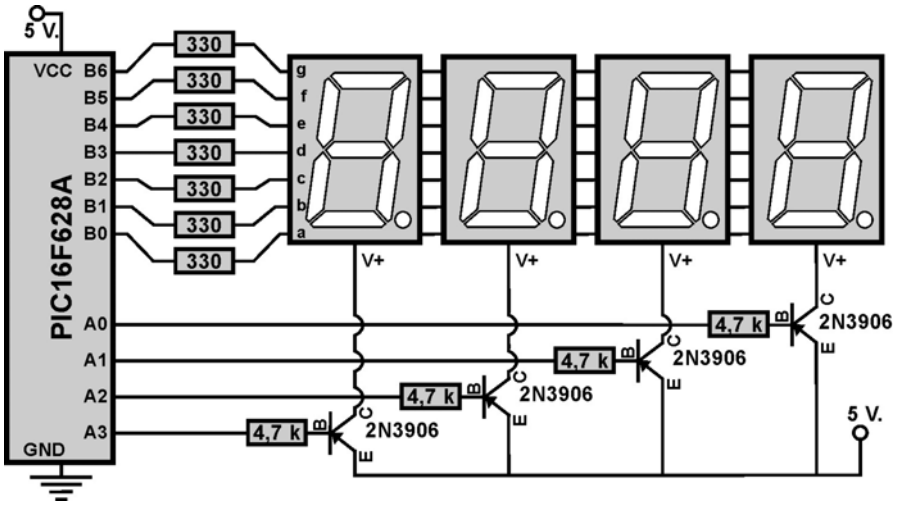


Figura 5.4.6.1. *Esquema de conexionado, para manejar 4 displays directamente desde el PIC y multiplexando por el puerto A.*

MATERIALES.
-4 DISPLAYS ánodo común
-4 transistores 2N3906
-7 resistencias de 330Ω a ½ vatio, naranja-naranja-café
-4 resistencias de 4,7 KΩ a ½ vatio, amarillo-violeta-rojo

```
cmcon=7                 ;convierte todo el puerto A en digital
trisb=0                 ;convierte todos los pines del puerto B en salidas
trisa=0                 ;convierte todos los pines del puerto A en salidas
texto:
   porta=14             ;%1110 activa el display de la derecha
   portb=8              ;%0001000 forma la letra A
   PAUSE 5
   porta=13             ;%1101 activa el siguiente display
   portb=71             ;%1000111 forma la letra L
   PAUSE 5
   porta=11             ;%1011 activa el siguiente display
   portb=64             ;%1000000 forma la letra O
   PAUSE 5
   porta=7              ;%0111 activa el último display
   portb=9              ;%0001001 forma la letra H
   PAUSE 5
GOTO texto
END
```

Figura 5.4.6.2. palabra HOLA 4D.pbp *Programa para presentar la palabra HOLA en los 4 displays.*

5.4.7. MANEJO DE 4 DISPLAYS COMO RÓTULO EN MOVIMIENTO.

Para hacerlo más interesante el proyecto anterior le añadiremos movimiento, con esto podemos ingresar frases completas como "HOLA LUIS", pero para no alargar mucho el programa sólo utilizaremos la palabra HOLA, moviéndose continuamente de derecha a izquierda y con un espacio por palabra.

La variable x es la que regula la velocidad con que se desplazan las letras, para comprobarlo modifique el valor de todas las repeticiones de 1 **TO** 20 al doble 1 **TO** 40, y verá cómo se desplazan las letras más lentamente.

```
cmcon=7                 ;convierte todo el puerto A en digital
trisb=0                 ;convierte todos los pines del puerto B en salidas
trisa=0                 ;convierte todos los pines del puerto A en salidas
x VAR BYTE              ;crea la variable x con un tamaño de 255

texto:
FOR x=1 TO 20           ;repeticiones de este segmento
   porta=14 :portb=8    ;pA%1110 y pB%0001000 forma la letra A
   PAUSE 5
   porta=13 :portb=71   pA%1101 y pB%1000111 forma la letra L
   PAUSE 5
   porta=11 :portb=64   ;pA%1011 y pB%1000000 forma la letra O
   PAUSE 5
   porta=7  :portb=9    ;pA%0111 y pB%0001001 forma la letra H
   PAUSE 5
   NEXT
FOR x=1 TO 20           ; repeticiones de este segmento
   porta=14 :portb=127  ;pA%1110 y pB%1111111 apaga el display
   PAUSE 5
   porta=13 :portb=8    ;pA%1101 y pB%0001000 forma la letra A
   PAUSE 5
   porta=11 :portb=71   ;pA%1011 y pB%1000111 forma la letra L
   PAUSE 5
   porta=7  :portb=64   ;pA%0111 y pB%1000000 forma la letra O
   PAUSE 5
   NEXT
FOR x=1 TO 20           ; repeticiones de este segmento
   porta=14 :portb=9    ;pA%1110 y pB%0001001 forma la letra H
   PAUSE 5
   porta=13 :portb=127  ;pA%1101 y pB%1111111 apaga el display
   PAUSE 5
   porta=11 :portb=8    ;pA%1011 y pB%0001000 forma la letra A
   PAUSE 5
   porta=7  :portb=71   ;pA%0111 y pB%1000111 forma la letra L
   PAUSE 5
   NEXT
FOR x=1 TO 20           ; repeticiones de este segmento
   porta=14 :portb=64   ;pA%1110 y pB%1000000 forma la letra O
   PAUSE 5
   porta=13 :portb=9    ;pA%1101 y pB%0001001 forma la letra H      continúa ....
```

```
   PAUSE 5
   porta=11 :portb=127          ;pA%1011 y pB%1111111 apaga el display
   PAUSE 5
   porta=7  :portb=8            ;pA%0111 y pB%0001000 forma la letra A
   PAUSE 5
   NEXT
FOR x=1 TO 20                   ; repeticiones de este segmento
   porta=14 :portb=71           ;pA%1110 y pB%1000111 forma la letra L
   PAUSE 5
   porta=13 :portb=64           ;pA%1101 y pB%1000000 forma la letra O
   PAUSE 5
   porta=11 :portb=9            ;pA%1011 y pB%0001001 forma la letra H
   PAUSE 5
   porta=7  :portb=127          ;pA%0111 y pB%1111111 apaga el display
   PAUSE 5
   NEXT
GOTO texto
END
```

Figura 5.4.7.1. p HOLA moviendo.pbp *Programa para desplazar la palabra HOLA.*

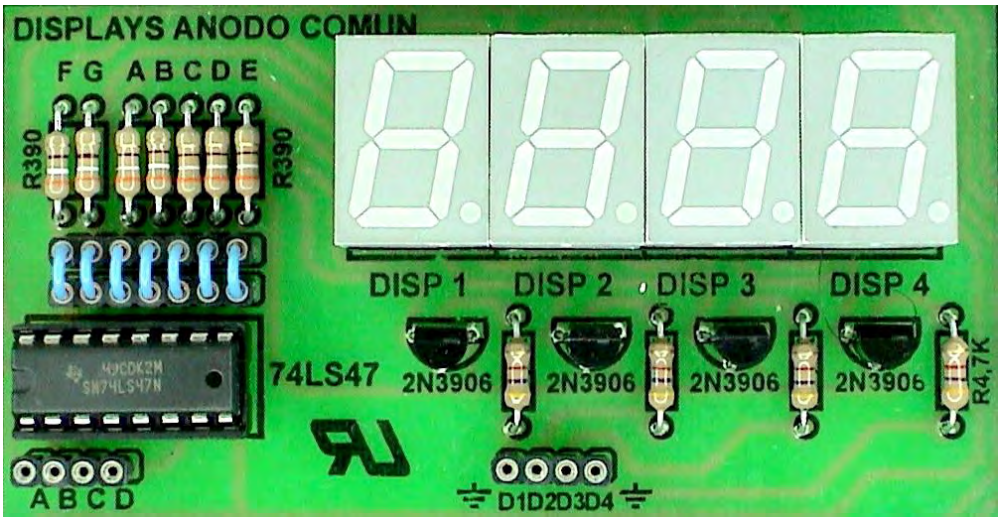
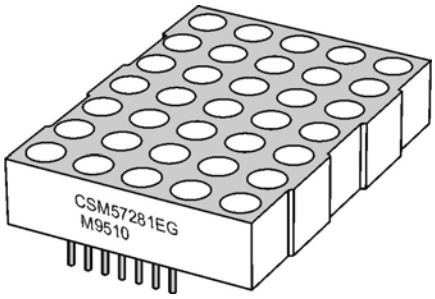


Figura 5.4.7.2. *Fotografía del bloque de 4 displays a. c. del entrenador experto EE 02.*

5.4.8. MANEJO DE UN DISPLAY DE 35 SEGMENTOS.

Estos displays son muy utilizados para transmitir mensajes en bancos y instituciones de atención al publico, la ventaja de estos son la gran cantidad de caracteres que se pueden formar y el gran tamaño de los displays (desde 3 x 2 cm hasta 15 x 8cm), en esta práctica aprenderemos a manejar un display de 35 segmentos de 14 pines, pero también existen displays bicolores y tienen 28

pines. Esta práctica consiste en formar un hombrecito saludando, una vez familiarizado será muy sencillo ir implementando más displays del mismo tipo.



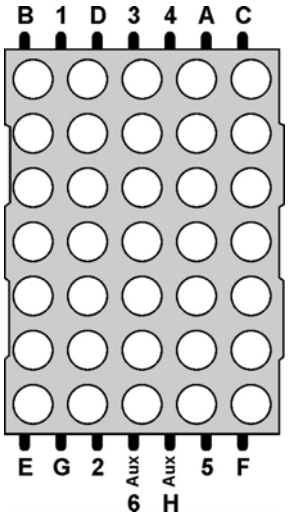
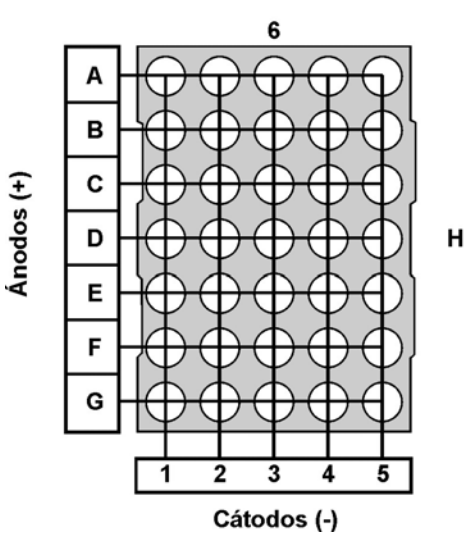


Figura 5.4.8.1. *Presentación de un display 7x5 de 14 pines y su respectivo diagrama de pines.*

Ejemplo: si queremos encender el 2do led de la columna 2, debemos conectar el pin B a 5 V. y el pin 2 a Gnd. Si queremos encender el led del centro podemos conectar el pin D o el pin H a positivo (el pin H es auxiliar) y a tierra el pin 3 o el 6 ya que también tiene un auxiliar.



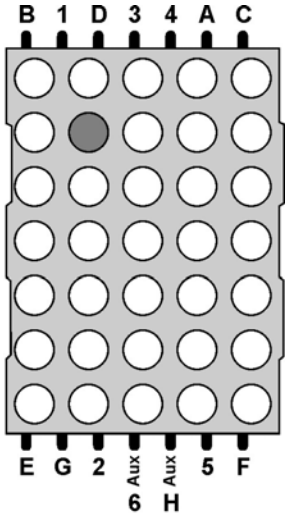


Figura 5.4.8.2. *Esquema de matrices de un display de 35 segmentos, para encendel el LED de la fila 2 columna 2.*

MATERIALES.
-1 DISPLAY 7 x 5 monocolor (14 pines)
-5 transistores 2N3904
-7 resistencias de 330Ω a ½ vatio, naranja-naranja-café
-5 resistencias de 4,7 KΩ a ½ vatio, amarillo-violeta-rojo.

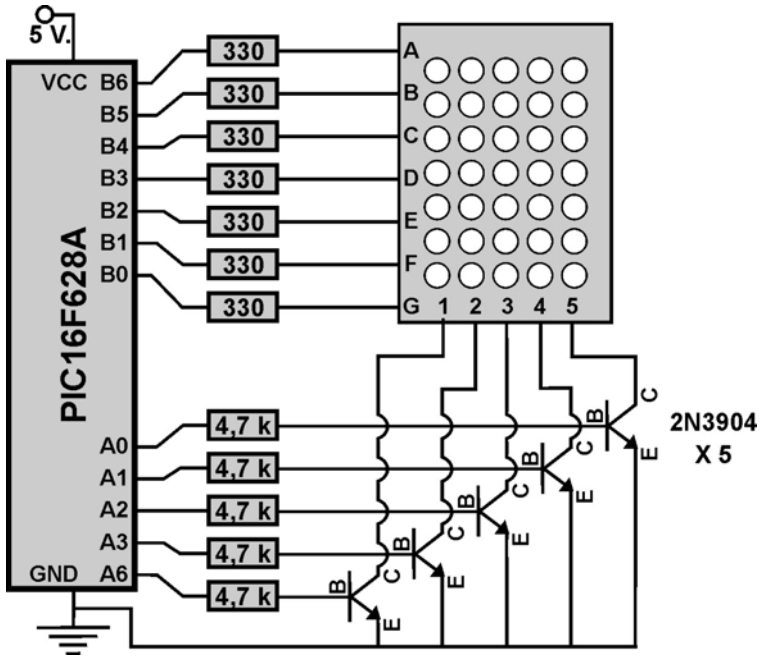


Figura 5.4.8.3. *Esquema de conexión de un display 7 x 5 a un PIC.*

NOTA: En esta práctica se utilizó transistores 2N3904, pero si vamos a utilizar más displays 7x5 y queremos mayor iluminación, se recomienda reemplazarlos por transistores TIP110 y adicionalmente colocar en la salida del puerto B transistores 2N3906, con una resistencia limitadora de 22 Ω a voltaje positivo. (ver figura 5.4.8.6.)

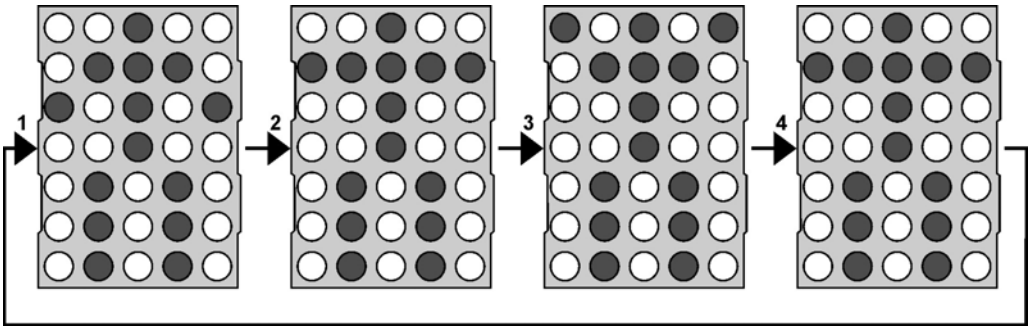


Figura 5.4.8.4. *Secuencia de la animación del hombrecito saludando.*

El programa a realizar, debe multiplexar los leds de forma que se encienda como la secuencia 1, permanece 100 milisegundos y cambia a la secuencia 2, luego a la secuencia 3, y finalmente a la secuencia 4, para luego volver a repetir toda la secuencia desde el principio. Como resultado observaremos un hombrecito que agita los brazos.

```
cmcon=7                                          ;convierte el puerto A en digital
trisa=0                                          ;hace salida todo el puerto A
trisb=0                                          ;convierte en salidas el puerto B
x VAR BYTE                                       ;crea variable x de 255

animacion:
FOR x = 1 TO 10                                  ;repetir esta secuencia 10 veces
   porta=%0000001 :portb=%0010000 :PAUSE 4       ;esperar 4 mls, total 20 mls cada
   porta=%0000010 :portb=%0100111 :PAUSE 4       ;escena
   porta=%0000100 :portb=%1111000 :PAUSE 4
   porta=%0001000 :portb=%0100111 :PAUSE 4
   porta=%1000000 :portb=%0010000 :PAUSE 4
  NEXT
FOR x = 1 TO 10                                  ;repetir esta secuencia 10 veces
   porta=%0000001 :portb=%0100000 :PAUSE 4
   porta=%0000010 :portb=%0100111 :PAUSE 4
   porta=%0000100 :portb=%1111000 :PAUSE 4
   porta=%0001000 :portb=%0100111 :PAUSE 4
   porta=%1000000 :portb=%0100000 :PAUSE 4
  NEXT
FOR x = 1 TO 10                                  ;repetir esta secuencia 10 veces
   porta=%0000001 :portb=%1000000 :PAUSE 4
   porta=%0000010 :portb=%0100111 :PAUSE 4
   porta=%0000100 :portb=%1111000 :PAUSE 4
   porta=%0001000 :portb=%0100111 :PAUSE 4
   porta=%1000000 :portb=%1000000 :PAUSE 4
  NEXT
FOR x = 1 TO 10                                  ;repetir esta secuencia 10 veces
   porta=%0000001 :portb=%0100000 :PAUSE 4
   porta=%0000010 :portb=%0100111 :PAUSE 4
   porta=%0000100 :portb=%1111000 :PAUSE 4
   porta=%0001000 :portb=%0100111 :PAUSE 4
   porta=%1000000 :portb=%0100000 :PAUSE 4
  NEXT
GOTO animacion                                   ;ir a animación
END
```

Figura 5.4.8.5. hombrecito.pbp *Programa para animación.*

Como podemos observar en el programa anterior, tratamos de no utilizar el puerto A.4 y A.5, porque el primero es de colector abierto, y el segundo sólo puede trabajar como entrada, además este programa se puede reducir por lo menos a la mitad si utilizamos **GOSUB** en la secuencia del hombre con los brazos en la mitad, ya que este se repite 2 veces, también en los **PAUSE** 4 y algunas partes de secuencias como las piernas y la cabeza que no se mueven, estos podemos agrupar también en una subrutina de **GOSUB**, pero para poder entender bien el funcionamiento se prefirió no reducir el programa.

Es muy importante considerar el tiempo de multiplexaje, ya que no se debe sobrepasar un total de 20 mls, por ejemplo si utilizamos 2 displays 7x5, debemos bajar el tiempo de pauses a 2 milisegundos para también tener un total de 20 mls.

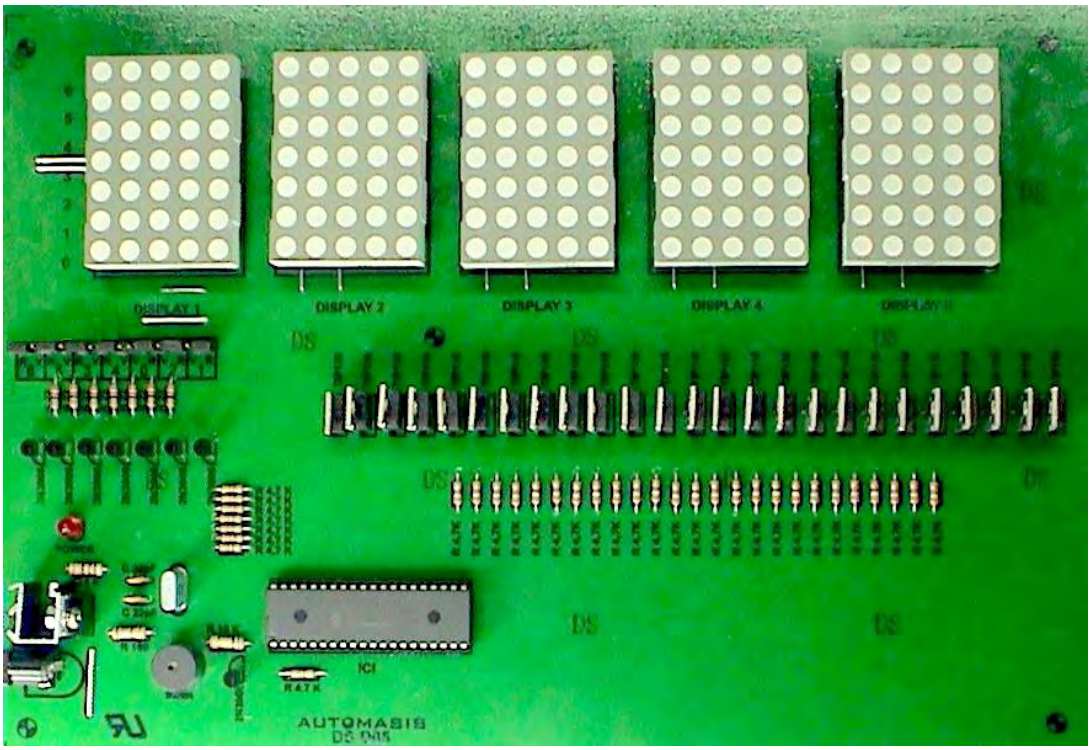


Figura 5.4.8.6. *Fotografía de una placa con 5 displays 7x5, comandado por un PIC16F877A, noten que los transistores 2N3904 fueron reemplazados por transistores TIP110, con la finalidad de conseguir mayor iluminación en los leds.*

5.4.9. PROYECTOS PROPUESTOS CON DISPLAYS.

1. En el proyecto 5.4.3 haga que aparezca la frase HOLA PACO y que vuelva a repetirse.

2. Con cualquiera de los 2 proyectos anteriores del literal 5.4.5 haga que compare el número almacenado ya no por 12 ni por 24 sino por 110.

3. En el proyecto del literal 5.4.7 haga aparecer el mensaje HOLA PACO desplazándose.

4. Con un display de 35 segmentos haga un contador que incremente cada segundo empezando desde el 0 hasta el 9 y luego que se repita.

5.5 MÓDULOS LCD

5.5.1. MANEJO DE UN MÓDULO LCD.

Los módulos LCD (Display de Cristal Líquido), son utilizados para mostrar mensajes que indican al operario el estado de la maquina, o para dar instrucciones de manejo, mostrar valores, etc. El LCD permite la comunicación entre las máquinas y los humanos, este puede mostrar cualquier caracter ASCII, y consumen mucho menos que los displays de 7 segmentos, existen de varias presentaciones por ejemplo de 2 líneas por 8 caracteres, 2x16, 2x20, 4x20, 4x40, etc. sin backlight (14 pines) o con backlight (16 pines, iluminado de pantalla), el LCD más popular es el 2x16, 2 líneas de 16 caracteres cada una.

1 16

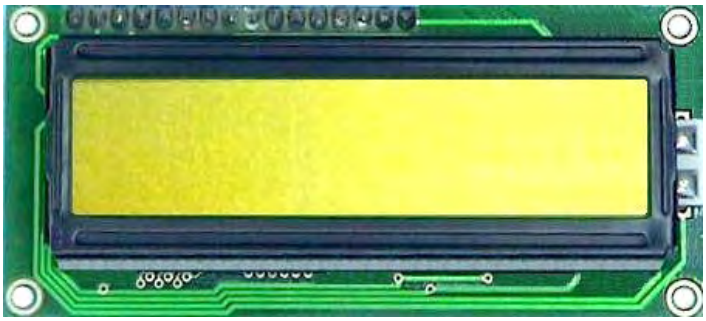


Figura 5.5.1.1. *Fotografía de un LCD 2x16 con controlador Hitachi 44780 y Backlight en color amarillo*

Pin	Simb.	Descripción
1	Vss	Tierra de alimentación GND
2	Vdd	Alimentación de +5V C.D.
3	Vo	Ajuste del contraste del cristal líquido (0 a +5V)
4	RS	Selección del registro control/datos RS=0 reg. control RS=1 reg. datos
5	R/W	Lectura/escritura en LCD R/W=0 escritura (Write) R/W=1 lectura (Read)
6	E	Habilitación E=0 módulo desconectado E=1 módulo conectado
7	D0	Bit menos significativo (bus de datos bidireccional)
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	Bit más significativo (bus de datos bidireccional)
15	A	Alimentación del backlight +3,5 V o +5V C.D. (según especificación técnica)
16	K	Tierra GND del backlight

Figura 5.5.1.2. *Función de cada pin del LCD.*

LA DECLARACIÓN LCDOUT. Sirve para mostrar items en una pantalla de cristal líquido, se utiliza escribiendo: **LCDOUT**, luego escribiendo $FE, y seguido por el comando a utilizar, el siguiente cuadro muestra los comandos más utilizados:

Comando	Operación
$FE, 1	Limpia el visor del LCD
$FE, 2	Vuelve al inicio (comienzo de la primera línea)
$FE, $0C	Apagar el cursor
$FE, $0E	Subrayado del cursor activo (_)
$FE, $0F	Parpadeo del cursor activo (▒)
$FE, $10	Mover el cursor una posición a la izquierda
$FE, $14	Mover el cursor una posición a la derecha
$FE, $80	Mueve el cursor al comienzo de la primera línea
$FE, $C0	Mueve el cursor al comienzo de la segunda línea
$FE, $94	Mueve el cursor al comienzo de la tercera línea
$FE, $D4	Mueve el cursor al comienzo de la cuarta línea

Figura 5.5.1.3. *Tabla de los comandos más utilizados para manejar un LCD.*

Los LCD se puede conectar con el PIC con un bus de 4 u 8 bits, la diferencia está en el tiempo que se demora, pues la comunicación a 4 bits, primero envía los 4 bits más altos y luego los 4 bits más bajos, mientras que la de 8 bits envía todo al mismo tiempo, esto no es un inconveniente si consideramos que el LCD trabaja en microsegundos. Pero la gran ventaja de hacer conexión a 4 bits, son los pocos cables que se deben conectar, como podemos ver en la figura 5.5.1.4. sólo debemos conectar el bit de Registro, el Enable y los 4 bits más altos del LCD, con esto es suficiente para enviar los mensajes.

El compilador PBP soporta módulos LCD´S con controlador Hitachi 44780 o equivalentes y por defecto, asume que se conectó en el pin A4 el bit de Registro, en el pin B3 el bit Enable y en el puerto A empezando desde A0 hasta A3, los bits más altos del LCD. Esta configuración predefinida, se lo puede cambiar de acuerdo a la necesidad, como lo veremos más adelante.

MATERIALES.
-1 DISPLAY LCD 2x16 (preguntar si es compatible con los PIC´S)
-1 resistencia de 10Ω a ½ vatio, café-negro-negro
-1 potenciómetro de 10 KΩ.

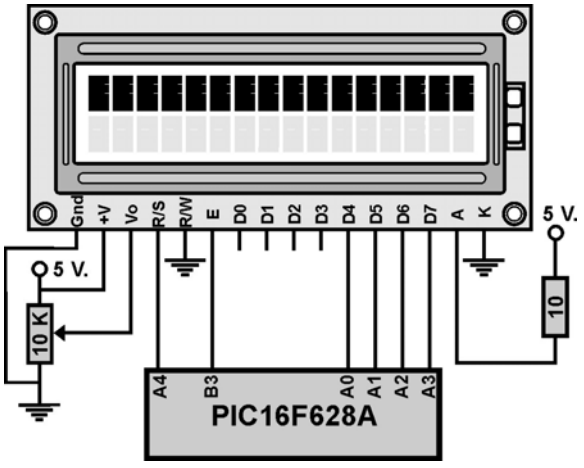


Figura 5.5.1.4. *Conexión de un LCD, a 4 bits predefinido por el compilador PBP, la resistencia de 10 Ω conectado a la alimentación del backlight, sirve para evitar altas temperaturas, noten además que el bit R/W se encuentra conectado a tierra, esto es porque la declaración* ***LCDOUT****, es de escritura únicamente.*

```
PAUSE 200                       ;retardo para esperar que funcione el LCD
LCDOUT $FE, 1,"Hola"            ;limpiar pantalla y sacar el texto Hola
LCDOUT $FE, $C0,"microPIC"      ;pasar al comienzo de la segunda línea
                                ;y escribir microPIC
END                             ;fin de instrucciones
```

Figura 5.5.1.5. LCD Hola.pbp *Programa para presentar la palabra Hola microPIC.*

NOTA: algunos LCD´S no requieren de ningún PAUSE al inicio, pero existen otros modelos que necesitan unos pocos milisegundos para estar listos, por eso colocamos un **PAUSE** 200 al comienzo del programa.

Bien una vez visto el texto notaremos que las dos palabras están al lado izquierdo, si queremos que salgan centradas en nuestro LCD, tenemos 2 maneras de hacerlo, la primera es dando espacios antes de cada palabra ejemplo:

LCDOUT $FE, 1, “ Hola” y **LCDOUT** $FE, $C0, “ microPIC”

Lo cual es sencillo pero no es muy recomendable porque ocupa más espacio en el PIC, la segunda manera es asignando el lugar donde se quiere que aparezca cada palabra ejemplo:

```
LCDOUT $FE, 1                   ;limpia la pantalla y coloca el cursor al comienzo
LCDOUT $FE, $86, “Hola”         ;pasa el cursor al 7ma casilla de la 1era Línea y escribe
LCDOUT $FE, $C4, “microPIC”     ;pasa a la casilla 5 de la 2da línea y escribe microPIC
```

Se debe entender que existe un cursor que aunque no lo vemos, pues este es el que indica donde aparecerá la siguiente letra, para poder entender haremos un ejercicio completo, así podrán aprender más del LCD y las funciones de cada uno de los comandos. Primero que nada haremos visible el cursor y luego pondremos PAUSES para poder ver el funcionamiento.

```
x VAR BYTE                      ;crea la variable x de un tamaño de 255
pepe CON $FE                    ;asigna el nombre de pepe a la constante $FE

   PAUSE 2000
LCDOUT pepe, 1                  ;limpia el visor del LCD
   PAUSE 2000
LCDOUT pepe,$0F                 ;muestra el cursor en casilla negra
   PAUSE 2000
LCDOUT pepe,$0E                 ;subraya el cursor
   PAUSE 2000
LCDOUT pepe,$14                 ;desplaza el cursor una casilla a la derecha
   PAUSE 2000
LCDOUT, "MIKRO"                 ;escribe mikro, desde donde se encuentre el cursor
   PAUSE 2000
 FOR x = 1 TO 3                 ; repite 3 veces las siguientes instrucciones
        LCDOUT pepe,$10         ; desplaza el cursor una casilla a la izquierda
        PAUSE 1000
   NEXT
LCDOUT,  67                     ; envía el caracter ASCII "C" para corregir MICRO
   PAUSE 2000                                                    continúa ....
```

```
LCDOUT pepe,$C0+12,"PIC"        ;escribe en la segunda línea casillero 13
                                ;esto equivale a $CC o 204
PAUSE 2000
LCDOUT pepe,2,"1"               ;vuelve al inicio de la 1era fila y escribe 1
END
```

Figura 5.5.1.6. LCD Hola2.pbp *Programa utilizando la mayoría de los comandos del LCD.*

Observen que la constante $FE se le cambió por pepe, asimismo si se les dificulta memorizar como pasar a la segunda línea, puede definir la constante: lin2 **CON** $C0, y cuando deseen escribir en la segunda línea pondrían: **LCDOUT** pepe, lin2, "hola", o lo que es lo mismo utilizando números decimales: **LCDOUT** 254, 192, "hola". También cabe recalcar que el LCD tiene una memoria RAM (Random Access Memory) que lo explicaremos más adelante, por lo que una vez que se le envía el texto, este permanece en la pantalla y el PIC se lo puede utilizar para otras tareas o podemos desconectarlo si lo deseamos.

En ocasiones especiales se debe cambiar la configuración de los pines del PIC hacia el LCD, por ejemplo para utilizar los comparadores de voltaje que se encuentran en el puerto A, necesitamos dejar disponibles estos pines, esto se logra adicionando al principio lo siguiente:

```
DEFINE  LCD_DREG    PORTB       ; define pines del LCD  B4  a  B7
DEFINE  LCD_DBIT    4           ; empezando desde el Puerto B4 hasta el B7
DEFINE  LCD_RSREG   PORTB       ;define el puerto B para conectar el bit RS
DEFINE  LCD_RSBIT   3           ;este es el puerto B3
DEFINE  LCD_EREG    PORTB       ;define el puerto B para conectar el bit Enable
DEFINE  LCD_EBIT    2           ;este es el puerto B2
```

Una vez que se define la nueva configuración de pines para el LCD, se programa de la misma forma que las ocasiones anteriores, es importante además saber que los 4 bits de datos sólo se pueden configurar en los 4 bits más bajos (B.0 al B.3) o los 4 bits más altos (B.4 al B.7) de un puerto del PIC, y si deseamos hacer una comunicación a 8 bits con el LCD, estos deben estar en un sólo puerto, además debemos definir en el PBP que vamos a utilizar un bus de 8 bits, esto es de la siguiente manera:

```
DEFINE LCD_BITS 8       ; define comunicación a 8 bits con el LCD
```

Y si nuestro LCD posee 4 líneas, también debemos definirlo de la siguiente forma:

```
DEFINE LCD_LINES 4      ; define un LCD de 4 líneas
```

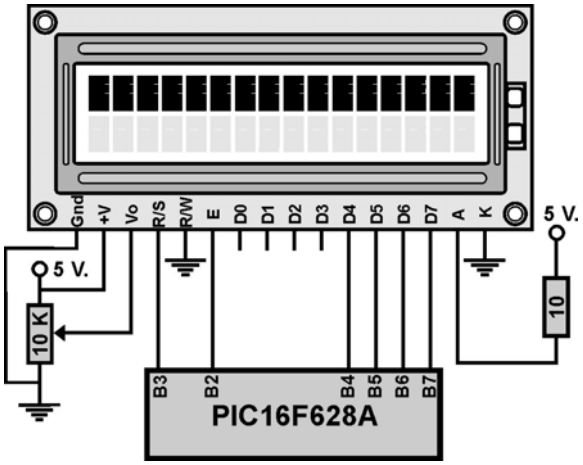


Figura 5.5.1.7*.Conexión de un LCD a 4 bits con una configuración diferente a la predefinida por el PBP, esta vez utilizando el puerto B3 para el bit R/S el B2 para el Enable y desde el B4 al B7 para los bits de comunicación.*

Figura 5.5.1.8. *Fotografía de una tarjeta que controla un LCD 2x16 el cual posee un PIC16F628 que se encarga de enviar los datos y además enciende el backlight a través de un transistor 2N3904 dispone además:*
Un led que indica envío de datos
Una salida de transistor TIP110
Una salida de transistor 2N3904
Un pulsador de reset al MCLR
Un potenciómetro para el ajuste del contraste.

5.5.2. PRESENTACIÓN DE CARACTER POR CARACTER EN LCD.

En las prácticas anteriores se presentaron mensajes completos en un instante, en esta nueva práctica incluimos la declaración **LOOKUP**, que nos servirá para enviar caracter por caracter con un intervalo de 400mls, dando como resultado un efecto especial en la visualización. Como conexión para esta práctica utilizaremos la misma de la figura 5.5.1.4.

```
PAUSE 200                       ;retardo para esperar que funcione el LCD
x VAR BYTE                      ;crear la variable x de 255
abc VAR BYTE                    ;crear la variable abc de 255
ini:
        LCDOUT $FE,1            ;limpiar pantalla
        FOR x = 0 TO 15         ;repetir 16 veces
LOOKUP x,["Microcontrolador"],abc       ;tomar caracter por caracter y guardar en abc
        LCDOUT, abc             ;sacar en LCD el contenido de abc
        PAUSE 400               ;esperar 400 mls
        NEXT                    ;siguiente repetición
        PAUSE 2000
GOTO ini
END
```

Figura 5.5.2.1. LCD especial.pbp *Programa para mostrar uno por uno cada caracter*.

5.5.3. DESPLAZAMIENTO DE UN TEXTO EN LCD.

El LCD dispone en cada una de las líneas 40 posiciones de memoria, de los cuales únicamente 16 son visibles, en el siguiente ejercicio escribiremos un mensaje desde el casillero 17 ($90), el cual no es visible y luego iremos desplazando a la izquierda, como resultado tendremos un texto que se mantiene en movimiento, una vez que este termina recorrerá 16 posiciones en blanco y luego volverá a aparecer los 24 caracteres del texto.

```
PAUSE 200                                   ;retardo para esperar que funcione el LCD
x VAR BYTE                                  ;crear la variable x de 255
abc VAR BYTE                                ;crear la variable abc de 255
LCDOUT $FE,$7                               ;configura para desplazamiento izquierdo
LCDOUT $FE,1                                ;limpiar pantalla
ini:
        LCDOUT $FE,$90                      ;ubica el cursor en la casilla 17
        FOR x = 0 TO 23                     ;repetir 24 veces
LOOKUP x,["Microcontroladores -PIC-"],abc   ;tomar caracter por caracter y guardar en abc
        LCDOUT,  abc                        ;sacar en LCD el contenido de abc
        PAUSE 400                           ;esperar 400 mls
        NEXT                                ;siguiente repetición
GOTO ini
END
```

Figura 5.5.3.1. LCD especial2.pbp *Programa para mostrar un texto en movimiento.*

5.5.4. CONTADOR DE PULSOS CON LCD.

Este proyecto, consiste en contar el número de pulsos que ingresan por un pin en un determinado período, este a su vez se visualiza en un LCD, si la cantidad de este supera a los 120 pulsos por segundo es decir 120 HZ, se encenderá una alarma que en este caso será un led rojo, y si la cantidad de pulsos baja a menos de 100 HZ, este encenderá un led verde, si la frecuencia se mantiene entre estos 2 rangos, no se encenderá ningún led.

Este proyecto tiene muchas aplicaciones como por ejemplo para un regulador de voltaje en el que a más de indicarnos el voltaje de salida podría además indicarnos la frecuencia.

Para esta práctica utilizaremos un CI. 555 que nos ayuda a generar un tren de pulsos variable, el cual lo conectamos al PIC para su posterior conteo.

LA DECLARACIÓN COUNT. Sirve para contar el número de pulsos que ingresan por un pin en un determinado tiempo, este a su vez lo guarda en una variable para su posterior procesamiento, la manera de utilizarlo es la siguiente:

```
COUNT portb.0, 1000, abc
```

El cual se interpreta así: cuenta pulsos a través del puerto B0 en un período de 1000 milisegundos y lo guarda en la variable previamente creada llamada abc, el período podemos variarlo de 1 a 65535.

MATERIALES.
-1 DISPLAY LCD 2x16
-1 resistencia de 10Ω, y 1 resistencia de 1KΩ
-3 leds de 5mm, 1 verde y 2 rojos
-3 resistencia de 330 Ω a ½ vatio, naranja-naranja-café
-2 potenciómetros de 10 KΩ
-1 CI. 555
-1 capacitor de 10uF/25V.

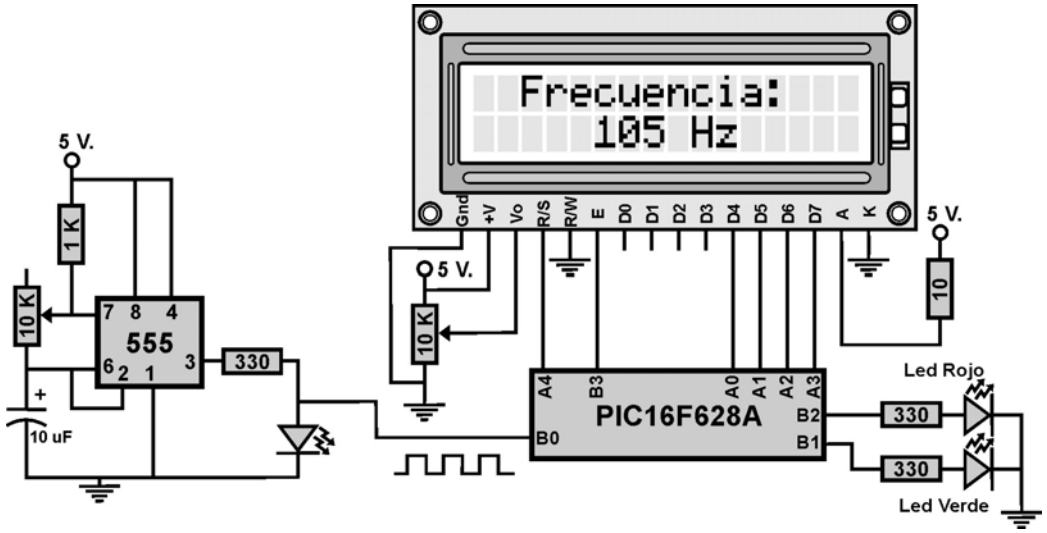


Figura 5.5.4.1. *Esquema de conexión para el frecuencímetro con LCD.*

```
CMCON = 7                          ;convierte en digitales el puerto A
LEDRED VAR PORTB.2                 ;el puerto B.2 se llamará LEDRED
LEDVER VAR PORTB.1                 ;el puerto B.1 se llamará LEDVER
puls VAR WORD                      ;variable puls con capacidad de 0 a 65535
prog:
   COUNT portb.0,1000,puls         ;contar pulsos en el puerto B.0
   LCDOUT $FE, 1                   ;limpiar pantalla LCD
   LCDOUT $FE,$82,"Frecuencia:"    ;cursor en el casillero 3 de la 1era línea
   LCDOUT $FE,$C5, DEC puls," Hz"  ;en el comienzo 2da línea casilla 6 sacar
                                   ;el valor decimal de la variable puls
   IF puls>120 THEN alarmah        ;si la variable excede de 120
   IF puls<100 THEN alarmal        ;si la variable baja de 100
   LOW LEDRED : LOW LEDVER         ;apagar los 2 leds
GOTO prog
alarmah: HIGH LEDRED               ;encender led rojo
         LOW LEDVER                ;apagar led verde
   GOTO prog
alarmal: HIGH LEDVER               ;encender led verde
         LOW LEDRED                ;apagar led rojo
   GOTO prog
```

Figura 5.5.4.2. frecuenciaLCD.pbp *Programa para un contador de pulsos.*

LA PALABRA DEC. Sirve para mostrar el número de la variable en decimal, también se lo puede representar por el signo (#), además existe las palabras **BIN** y **HEX**, el siguiente es un ejemplo de cómo mostraría el LCD si puls = 105:

LCDOUT $FE, $C5, **DEC** puls, " Hz" ; muestra en el LCD así : 105 Hz

También se lo puede utilizar el signo #, que equivale a **DEC**.
LCDOUT $FE, $C5, **#** puls, " Hz" ; muestra en el LCD así : 105 Hz

Si deseamos ver la variable en hexadecimal pondríamos así:
LCDOUT $FE, $C5, **HEX** puls, " Hz" ; muestra en el LCD así : 69 Hz

Y si queremos verlo en binario:
LCDOUT $FE, $C5, **BIN** puls, " Hz" ; muestra en el LCD así : 1101001 Hz

Si no colocamos ninguna instrucción nos mostraría el ASCII que representa el número 105, es decir la letra i.
LCDOUT $FE, $C5, puls, " Hz" ; muestra en el LCD así : i Hz

5.5.5. TACÓMETRO DIGITAL.

Este proyecto es muy similar al anterior, con la diferencia que el tren de pulsos ya no es generado por un C.I. 555, sino más bien por el giro de un motor que se une a un cono de caucho, el cual transmite movimiento a un disco de plástico negro, en el que posee una ranura de 1 a 2 mm, que al pasar por el medio del optoacoplador, polariza 2 transistores y este hace cambiar el estado de 0 a 1, esta señal podemos conectarlo al PIC y visualizarlo en un LCD, el mismo principio utilizan los marcadores de kilometraje de los autos y los tacómetros de los mismos.

MATERIALES.
-1 DISPLAY LCD 2x16
-1 resistencia de 10Ω
-1 resistencia de 330 Ω
-1 resistencia de 1KΩ
-1 resistencia de 4,7 KΩ
-1 potenciómetros de 10 KΩ
-1 eje de giro independiente con un disco de plástico negro
-1 optoacoplador ECG3100 con salida de transistor NPN como el de la figura 5.5.5.2.

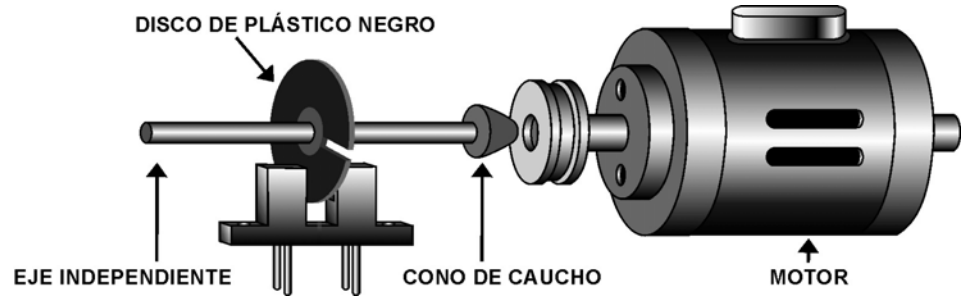


Figura 5.5.5.1. *Esquema para unir el motor con el eje independiente de un tacómetro portátil.*

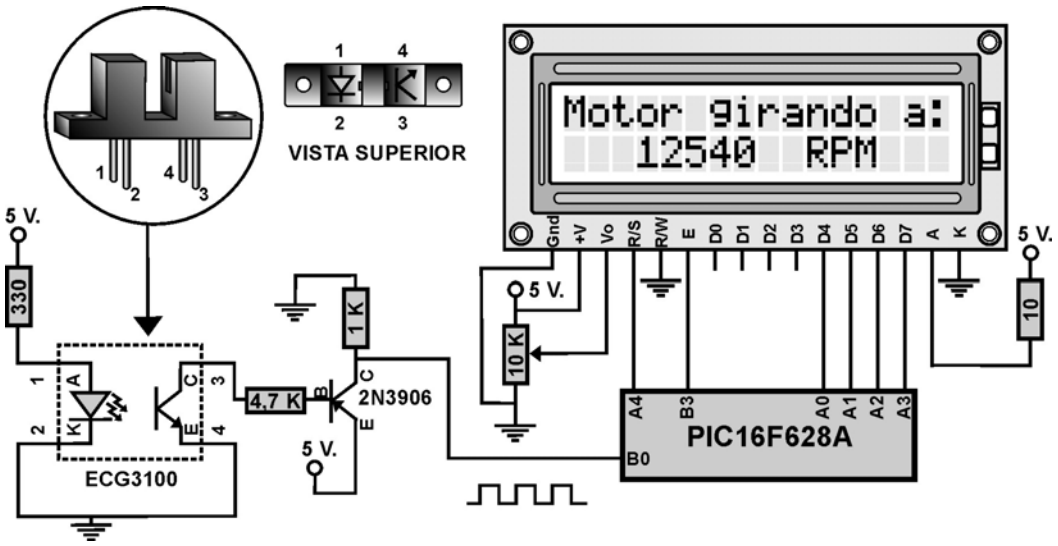


Figura 5.5.5.2. *Esquema de conexión para hacer un tacómetro digital, adicionalmente muestra la forma del optoacoplador con fototransistor ECG 3100.*

```
CMCON = 7                                ;convierte en digitales el puerto A
revo VAR WORD                            ;variable revo con capacidad de 0 a 65535

prog:
   COUNT portb.0,1000,revo               ;contar pulsos en el puerto B.0
   revo = revo * 60                      ;multiplicar por 60 para tener 1 minuto rpm
   LCDOUT $fe, 1,"Motor girando a:"       ;limpiar LCD y escribir
   LCDOUT $fe,$c3, DEC revo              ;sacar el valor de la variable revo
   LCDOUT $fe,$c9," RPM"                 ;ir a 2da línea casilla 9 y escribir RPM
GOTO prog
END
```

Figura 5.5.5.3. tacómetro-LCD.pbp *Programa para un contador de revoluciones por minuto.*

Observen que la variable revo es multiplicada por 60, con la finalidad de que nos de el número de vueltas que daría en un minuto, estos datos salen en múltiplos de 60, por consiguiente no es muy preciso, si deseamos más precisión podemos multiplicar por 30, pero debemos asegurar que la declaración **COUNT** cuente durante 2 segundos, para luego de multiplicar por 30, nos de RPM, la línea de programa quedaría así:

```
   COUNT portb.0,2000,revo     ;contar pulsos en el puerto B.0 durante 2 segundos
   revo = revo * 30            ;multiplicar por 30 para tener 1 minuto rpm.
```

Lo más óptimo sería que la declaración COUNT cuente durante 1 minuto es decir:

```
   COUNT portb.0,60000,revo    ;contar pulsos en el Puerto B.0 durante 60 segundos.
```

En este caso no debemos multiplicar con ningún valor la variable revo, el inconveniente sería que deberíamos permanecer conectados al motor por un minuto, hasta que la variable **COUNT** termine de contar los pulsos, por esto lo más aconsejable sería de 2 a 5 segundos, en este último caso deberíamos multiplicarlo por 12.

```
revo=revo * 12          ;multiplicar la variable revo por 12 para tener RPM
```

NOTA: Se puede utilizar los optoacopladores que vienen en los mouse de los PC, también es importante que el disco sea de color negro, para impedir que la luz infrarroja atraviese hacia el transistor, un disco de metal también podría funcionar bien.

5.5.6. LECTURA DE UN POTENCIÓMETRO CON LCD.

LA DECLARACIÓN POT. Esta declaración sirve para leer el estado de un potenciómetro de 5K hasta uno de 50K, o cualquier elemento resistivo (fotoceldas, termistores, etc.), el principio de funcionamiento es muy sencillo utiliza la ayuda de un condensador de 0,1 uF, al cual lo carga y lo descarga utilizando el potenciómetro para regular la corriente que circula, entonces a mayor resistencia el capacitor se demora más tiempo en cargarse, y la variable nos dará un valor alto y si giramos el potenciómetro a la mínima resistencia, el capacitor se cargará más rápido y la variable nos dará cero, en definitiva estaremos leyendo el estado de un potenciómetro, el cual podemos aplicarlo en la atenuación de una luz por ejemplo o la variación de la velocidad de un motor.

MATERIALES.
-1 DISPLAY LCD 2x16
-1 resistencia de 10Ω
-2 potenciómetros de 10 KΩ
-1 condensador cerámico de 0,1uF (referencia 104)

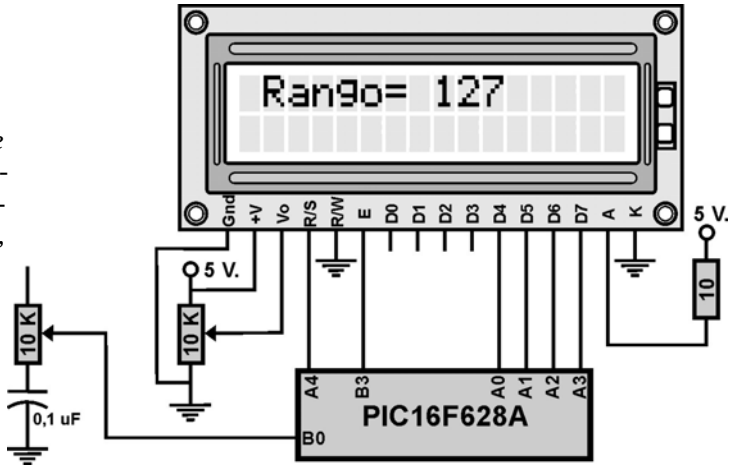


Figura 5.5.6.1. *Conexión de un potenciómetro para leerlo. En este caso el potenciómetro se encuentra en la mitad de su escala es decir 5 KΩ, esto equivale a 127, y si estuviera en 10KΩ, equivaldría a 255.*

```
CMCON = 7                          ;convierte en digitales el puerto A
dato VAR BYTE                      ;variable dato con capacidad de 255

medir:
   POT portb.0,255,dato            ;leer el potenciómetro y guardar en dato
   LCDOUT $FE, 1," Rango= "        ;limpiar pantalla y escribir rango=
   LCDOUT, #dato                   ;mostrar el valor decimal de dato
   PAUSE 100
GOTO medir
END
```

Figura 5.5.6.2. potenciometro.pbp *Programa para leer el estado de un potenciómetro.*

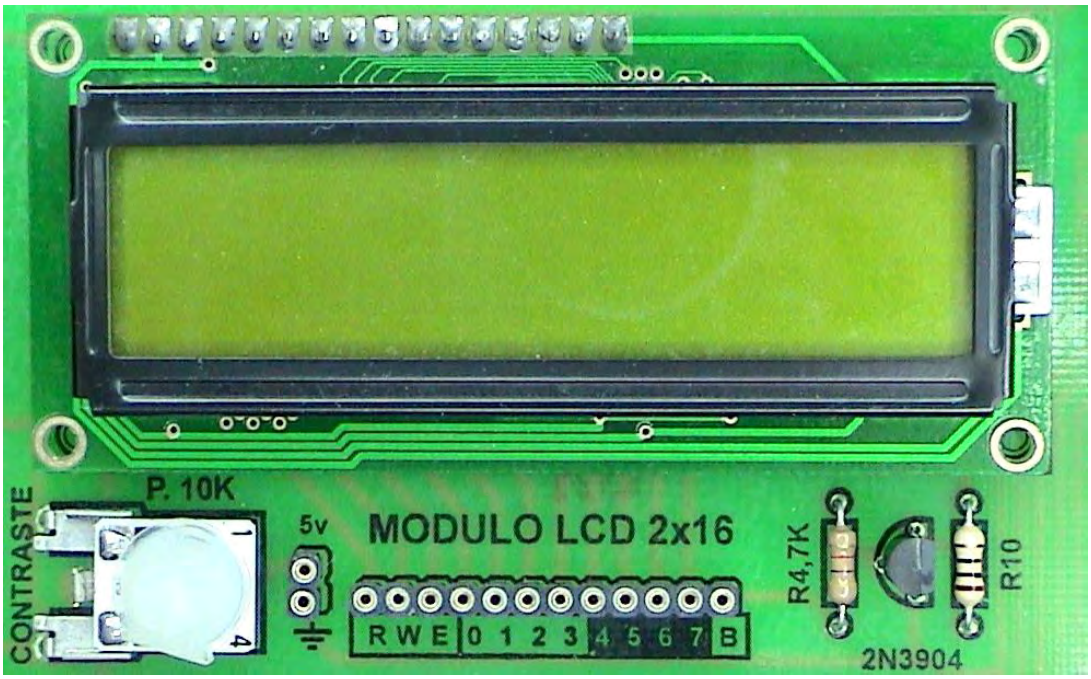


Figura 5.5.6.3. *Fotografía del módulo LCD del entrenador experto de PIC´S EE-02 de* AUTOMASIS.

5.5.7. PROYECTOS PROPUESTOS CON LCD.

1. En la pantalla de un LCD haga que aparezca intermitentemente la palabra PELIGRO.

2. Con un LCD y 3 pulsadores P1, P2 y P3, haga un teclado alfabético que presente mensajes a su gusto en el LCD de la siguiente manera: con P2 haga que aparezca el alfabeto desde la A hasta la Z sin desplazarse, con el pulsador P3 haga desplazar el cursor hacia la derecha para seguir escribiendo, y P1 hace que el cursor regrese hacia la izquierda para corregir el texto.

5.6 SONIDO

5.6.1. GENERACIÓN DE SONIDO.

El compilador PBP es capaz de sacar las frecuencias especificadas por un pin del PIC, para esto aprenderemos la declaración **FREQOUT**.

LA DECLARACIÓN FREQOUT. Saca la o las frecuencias especificadas por un pin del micro, estas pueden ser de 0 a 32767 Hz, su utilización es de la siguiente manera:

FREQOUT portb.0, 2000, 1000

Quiere decir sacar una frecuencia de 1000 ciclos (1 Khz) durante 2 segundos por el puerto B.0

En esta práctica lo utilizaremos para generar un sonido a través de un piezoeléctrico (Buzzer pasivo) o podría ser también un parlante.

MATERIALES.
-1 buzzer pasivo (piezoeléctrico) o un parlante grande o pequeño
-1 capacitor de 10uF/25V.

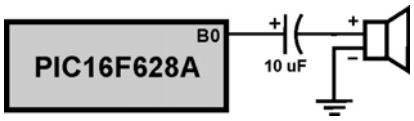


Figura 5.6.1.1. *Diagrama de conexión de un parlante pequeño o un piezoeléctrico, este necesita de un capacitor para poder mejorar la señal del PIC.*

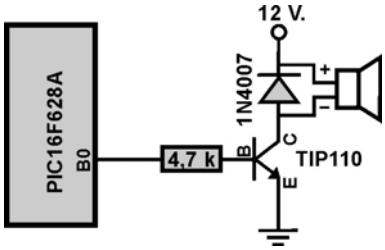


Figura 5.6.1.2. *Diagrama de conexión para un parlante grande con un transistor TIP110.*

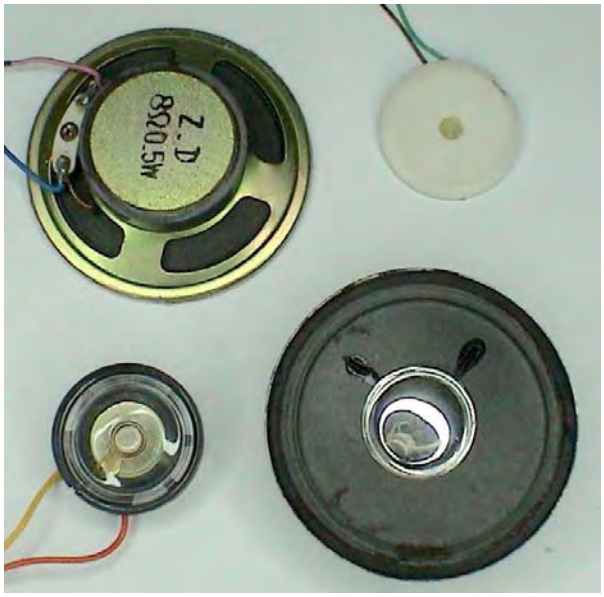


Figura 5.6.1.3. *Fotografía de 2 parlantes grandes un pequeño y un piezoeléctrico.*

NOTA: también es posible utilizar los parlantes grandes con la conexión de la figura 5.6.1.1.

```
FREQOUT portb.0, 2000, 7200        ;sacar una frecuencia de 7,2 Khz
                                   ;durante 2 segundos por el Puerto B.0
END
```

Figura 5.6.1.4. freqout.pbp *Programa para generar sonido a través de un parlante*.

5.6.2. UNA SIRENA POLICIAL.

Esta práctica consiste en sacar por un parlante el sonido característico de una sirena policial, para esto emplearemos la ayuda de la declaración **SOUND**.

LA DECLARACIÓN SOUND. Sirve para generar tonos y/o ruido blanco en un pin del PIC, y es posible combinar hasta 2 frecuencias desde de 1 a 127 que son tonos y 128 a 255 ruido blanco, 0 es silencio, 1 equivale a 78,74 HZ y 127 a 10000 Hz, esto se lo utiliza de la siguiente manera:

SOUND portB.0,[100,10,50,10]

Esto quiere decir sacar 2 tonos por el puerto b.0, el primer tono es 100 que equivale a (7874 Hz) con una duración de 10 milisegundos y luego un tono de 50 (3937 Hz) con una duración de 10 milisegundos también.

En cuanto a los materiales y diagrama de conexión son los mismos de la práctica anterior

```
Programa:
      SOUND portb.0, [100,10,50,10]        ;genera tonos por el Puerto B.0
      GOTO programa
END
```

Figura 5.6.2.1. sirena.pbp *Programa para generar una sirena policial a través de un parlante.*

UTILIZANDO UN CRISTAL DE MAYOR VELOCIDAD. Como sabemos el PIC está trabajando actualmente a una velocidad de 4Mhz, utilizando un oscilador RC interno (resistencia condensador), pero el PIC puede operar también con osciladores externos de hasta 20 Mhz. Este es el momento de aprender a utilizar un oscilador de mayor frecuencia (8,10,12,16,20 Mhz), en este caso notaremos una considerable diferencia en cuanto a la nitidez del sonido respecto al programa 5.6.2.1, esto se logra adicionando un **DEFINE** al inicio del programa de la siguiente manera:

```
DEFINE OSC 20        ; especifica al PBP que se va a utilizar un cristal de 20 Mhz
```

Con un oscilador de 20Mhz, el micro trabaja 5 veces más rápido que antes es decir si con un oscilador interno de 4 Mhz, el PIC ejecutaba cada instrucción en 1 uS., con un oscilador de 20 Mhz lo hará en 0,2 uS (0,0000002 S).

Para esta práctica necesitamos estos nuevos elementos además de los anteriores:

MATERIALES.
-1 cristal de 20 MHZ como el de la figura 5.6.2.2. ideal para protoboards
-2 condensadores de 22 pF (22 picoFaradios)

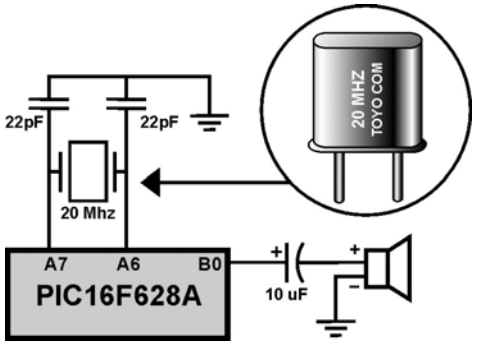


Figura 5.6.2.2. *Diagrama de conexión de un cristal externo, muestra también la forma de este.*
En el caso de utilizar un cristal de 10 Mhz, se debe poner : **DEFINE** OSC 10

```
DEFINE OSC 20                    ;especifica que se va a utilizar un oscilador de 20 Mhz
Programa:
        SOUND portb.0, [100,10,50,10]          ;genera tonos por el Puerto B.0
        GOTO programa
END
```

Figura 5.6.2.3. sirena20Mhz.pbp*Program para generar una sirena policial mejor que la anterior*

IMPORTANTE: Este proyecto no funciona si no se especifica en el IC-prog que se va a utilizar un cristal externo de 20 Mhz, para esto debemos seleccionar HS en donde dice configuración Oscilador.
Para el PIC16F628A se recomienda únicamente utilizar cristales externos de 4, 10, o 20 MHZ.

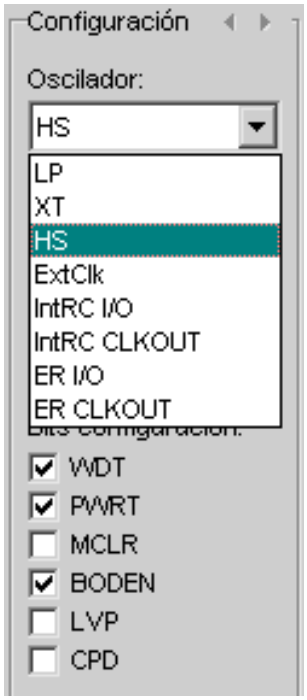


Figura 5.6.2.4*.Configuración manual del Oscilador del programa IC-prog, para este ejercicio en el que se va a utilizar un cristal externo de 20 MHZ, debemos seleccionar* Oscilador HS, *ver tabla de la figura 5.6.2.5.*

Frecuencia	OSC1/ C1	OSC2/ C2	Tipo
4 Mhz	------	------	IntRC I/O
32 Khz	68 – 100 pF	68 – 100 pF	LP
200 Khz	15 – 30 pF	15 – 30 pF	
100 Khz	68 – 150 pF	68 – 150 pF	XT
2 Mhz	15 – 30 pF	15 – 30 pF	
4 Mhz	15 – 30 pF	15 – 30 pF	
8 Mhz	15 – 30 pF	15 – 30 pF	HS
10 Mhz	15 – 30 pF	15 – 30 pF	
12 Mhz	15 – 30 pF	15 – 30 pF	
16 Mhz	15 – 30 pF	15 – 30 pF	
20 Mhz	15 – 30 pF	15 – 30 pF	

Figura 5.6.2.5. *Tabla de Configuración de Oscilador para el IC-prog, según el cristal a utilizar, además indica los valores de capacitores que se deben poner.*

Para mayor facilidad podemos agregar una línea de código ensamblador que se encargará de cambiarnos el oscilador predefinido a HS, para ello debe escribir al principio del programa lo siguiente:

```
@ device HS_OSC                ;cambia automáticamente a oscilador de alta velocidad HS
```

Para comprobarlo una vez compilado abra el archivo en el programa IC-prog y observará que el oscilador ha cambiado a HS sin que usted haga nada.

5.6.3. GENERACIÓN DE UN TIMBRE DE TELÉFONO CELULAR.

Esta práctica consiste en sacar por un parlante el sonido característico de un teléfono celular, para esto emplearemos la ayuda de la declaración **SOUND**.

En cuanto al diagrama de conexión y materiales, podemos utilizar el mismo de la práctica 5.6.1.

```
x VAR BYTE                      ;variable x de 255
prog:
FOR X= 1 TO 15                  ;repetir de 1 a 15 veces
      SOUND portB.0,[125,4,123,5]  ;sacar tonos por el puerto b.0
      NEXT                      ;siguiente repetición
      PAUSE 2500                ;esperar 2,5 segundos antes de volver a timbrar
GOTO prog
```

Figura 5.6.3.1. timbre-celular.pbp *Programa para generar un timbre de teléfono celular*.

Pruebe con algunas combinaciones de tonos, así como también con diferentes tiempos y verá que bien podría componer una melodía.

5.6.4. LLAMADA TELEFÓNICA DTMF.

Esta práctica tiene como objetivo generar tonos DTMF (Dual-Tono MultiFrecuency) a través de un PIC, como los que genera cualquier teléfono fijo o celular, estos tonos no son nada más que el envío de 2 frecuencias específicas asignadas a cada tecla, estas frecuencias podemos ver en la figura 5.6.4.1., lo que sucede cuando pulsamos una tecla por ejemplo la 6, es que este envía una frecuencia del grupo bajo 770 Hz y luego una segunda frecuencia del grupo alto 1477 Hz, esto podríamos hacerlo generando con la declaración **FREQOUT**.

```
FREQOUT  portb.0,200,941,1336     ;equivale a presionar la tecla 0 de DTMF, durante 200 mls
     PAUSE 50                     ;retardo de 50 mls antes de pulsar la siguiente tecla
FREQOUT  portb.0,200,852,1477     ;equivale a presionar la tecla 9 de DTMF, durante 200 mls
     PAUSE 50                     ;retardo de 50 mls antes de pulsar la siguiente tecla
```

Esto deberíamos hacer por cada tecla que deseáramos que marque, pero para facilitarnos las cosas el compilador PBP tiene una declaración específica para este trabajo.

LA DECLARACIÓN DTMFOUT. Esta genera automáticamente los tonos duales correspondientes a cada tecla y los envía cada una con intervalos de 50 milisegundos, aunque los tiempos podemos cambiarlo si lo deseamos (ver manual de pbp) su manera de utilizar es la siguiente:

```
DTMFOUT portb.0, [0,9,6,1,3,6,5,6,4]        ; equivale a presionar  las teclas 0,9,6,1,3,6,5,6,4
```

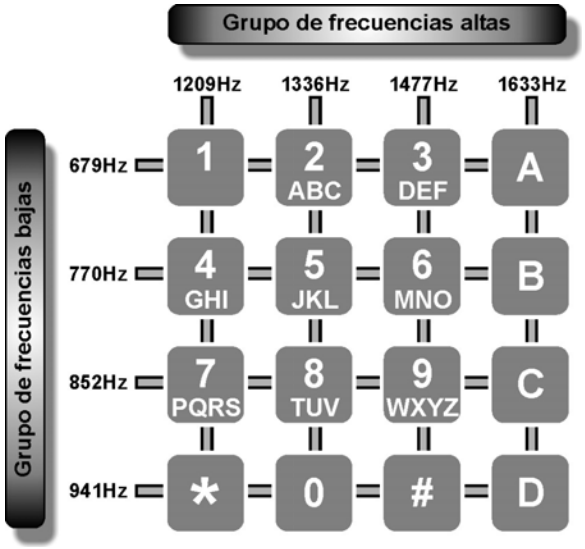


Figura 5.6.4.1. *Tabla de las frecuencias DTMF correspondiente a cada tecla, las teclas A,B,C,D, son para aplicaciones especiales, y no se los encuentra en los teclados comunes.*

En la figura 5.6.4.2., se muestra el diagrama básico de conexión para poder hacer una llamada telefónica, debido a que el oscilador interno que posee el PIC es un RC (resistencia condensador), este no es muy preciso, por lo que experimentalmente se comprobó que el 90 % de los intentos para marcar los tonos DTMF fallaban y sólo el 10% restante de los intentos resultaban con la llamada al número deseado.

Para que los tonos DTMF que generan el PIC sean válidos al 100%, es necesario utilizar un cristal externo, sea este de 10, 12 o 20 MHZ, con sus debidos capacitores, en este caso

usaremos el de 20 MHZ. La resistencia de 560Ω a 1 vatio paralela a la red telefónica, sirve para simular la carga de un teléfono normal, y con esto podremos tener el tono de marcado necesario para hacer la llamada, además notarán que esta resistencia empieza a disipar calor en el momento que se une a la red telefónica, esto debido a la cantidad de voltaje que circula (60 voltios).

El relé hace la conexión y desconexión de la red telefónica, que en este caso sería como el auricular que cuelga y descuelga el teléfono. El capacitor electrolítico de 1 uF a 100 voltios, sirve para poder mejorar la onda que sale del PIC y además como protección para el PIC. Es importante indicar que la red telefónica suministra alrededor de 60 voltios en DC, por lo que el capacitor debe pasar de 60 voltios y además el lado positivo de la red telefónica debe ir a tierra del PIC y el negativo hacia el pin del PIC, por lo que necesitaremos la ayuda de un voltímetro para poder identificar la polaridad de la red.

Para solucionar este inconveniente se propone el diagrama de la figura 5.6.4.3., este es un diagrama más completo en el que la polaridad de la línea no es un problema, puesto que dispone un puente de diodos en donde el lado positivo ya está unido a tierra y el lado negativo va a través del filtro hacia el PIC.

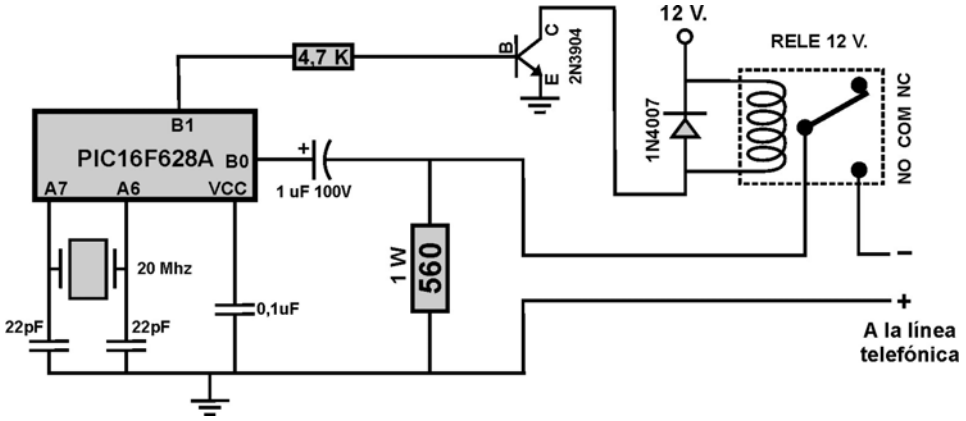


Figura 5.6.4.2. *Diagrama básico de conexión para generar una llamada telefónica, noten que el lado positivo de la red telefónica debe ir a tierra del PIC, caso contrario no se podrá generar la llamada.*

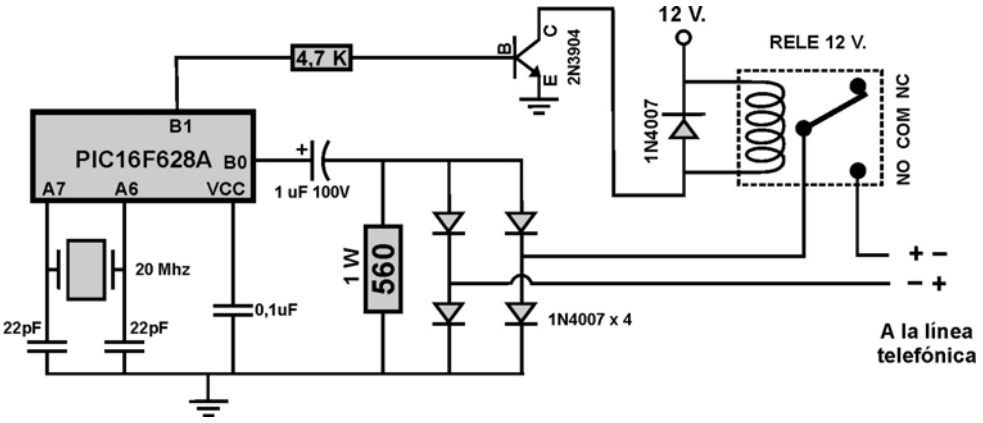


Figura 5.6.4.3. *Diagrama de conexión para generar una llamada telefónica, en la que no importa la polaridad de la red telefónica, puesto que dispone un puente de diodos.*

MATERIALES.
-1 cristal de 20 MHZ
-3 condensadores cerámicos, 2 de 22 pF (22 picoFaradios) y 1 condensador de 0,1 uF (103)
-1 capacitor electrolítico de 1 uF a 100voltios
-5 diodos rectificadores 1N4007
-1 resistencia de 4,7 KΩ
-1 resistencia de 560 Ω a 1 vatio
-1 transistor 2N3904
-1 relé a 12 voltios de 5 patitas

Este proyecto es muy utilizado en sistemas de seguridad, se puede hacer una alarma la cual en el momento de violar su seguridad, este realice automáticamente una llamada al propietario, el propietario en el momento de contestar escuchará una sirena, señal suficiente para saber que alguien ha activado la alarma, este mismo principio utilizan las centrales de monitoreo, con la diferencia que en vez de generar un sonido de sirena, se envía datos en forma serial como: Qué zona se abrió, a qué hora, estado de batería, etc. lo cual aprenderemos más adelante en comunicaciones con microcontroladores PIC.

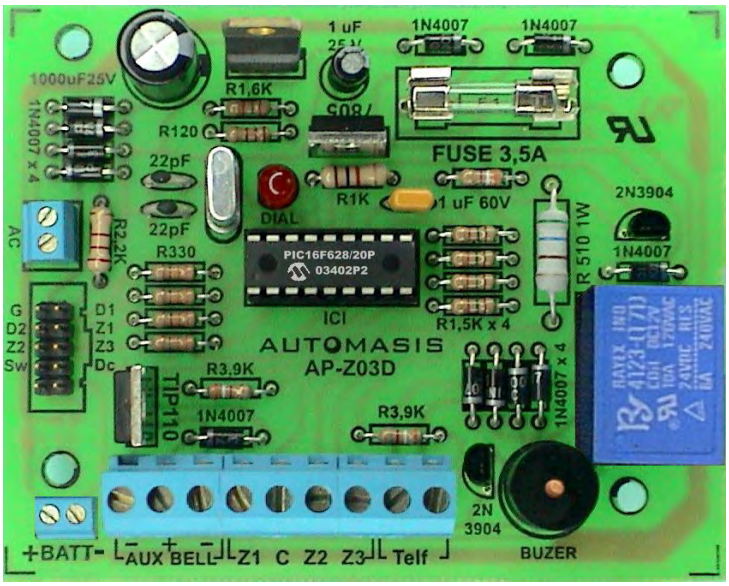


Figura 5.6.4.4. *Fotografía de una tarjeta electrónica de una alarma de 3 zonas con armado de llave, cargador de batería y llamada telefónica, comandado por un PIC16F628.*

```
@ device HS_OSC          ;cambia a oscilador HS en el IC-Prog.
DEFINE OSC 20            ;defíne oscilador externo de 20 MHZ.

rele VAR portb.1         ;nombre relé para el pin B.1
x VAR BYTE               ; variable x con tamaño de 255
iniciar:                                          continúa ...
```

```
PAUSE 2000                          ;espera de 2 segundos antes de empezar
  HIGH rele                         ;conecta a la línea telefónica
PAUSE 1000                          ;espera 1 segundo hasta que exista tono de marcar

DTMFOUT portb.0,[0,9,6,1,3,6,5,6,4]     ;número al cual el PIC va a llamar

PAUSE 4000                          ;esperar 4 seg. hasta que alguien conteste

FOR x = 1 TO 25                     ;repetir 25 veces, equivale a 6 segundos
   SOUND portb.0,[100,10,50,10]     ;enviar el sonido de sirena por el teléfono
   NEXT

LOW rele                            ;desconecta el relé, el cual cierra la llamada
END
```

Figura 5.6.4.5. llamada DTMF.pbp *Programa para hacer una llamada al teléfono celular*.

Figura 5.6.4.6. *Fotografía de la alarma de 3 zonas basado en un PIC16F628A, dispone de 3 leds para indicar la zona que está abierta, un led que indica si está armado o desarmado y un led que indica si hay poder AC, además del cargador de batería y llamada telefónica*.

5.6.5. PROYECTO PROPUESTO.

1. Haga una alarma de 3 zonas (3 pulsadores) con armado de un switch, es decir si el swith está en OFF, los pulsadores no tienen ningún efecto, pero si está en ON, al presionar cualquiera de los 3 pulsadores, se enciende un led indicando qué zona es y hace una llamada telefónica a su celular u otro teléfono fijo.

5.7 PROYECTOS CON TECLADOS

5.7.1. LECTURA DE UN TECLADO DE 16 PULSADORES CON DISPLAY DE 7 SEGMENTOS.

Los teclados matriciales son muy útiles para ingresar datos, un ejemplo es el teclado del computador, el teclado de una alarma que nos permite armar y desarmar un sistema de seguridad, el teclado de una caja fuerte, el de una cerradura eléctrica, etc. Para introducirnos en el manejo de un teclado, haremos un proyecto para aprender a identificar filas y columnas que lo componen un teclado hexadecimal de 16 pulsadores, y su correspondiente barrido de teclas, luego se visualizará en un display de 7 segmentos el número de la tecla presionada.

MATERIALES.
-1 teclado matricial hexadecimal de 16 teclas como el de la figura 5.7.2.3
-7 resistencias de 330Ω
-4 resistencia de 4,7 KΩ
-1 DISPLAY de 7 segmentos ánodo común
-1 CI. 7447 decodificador BCD.

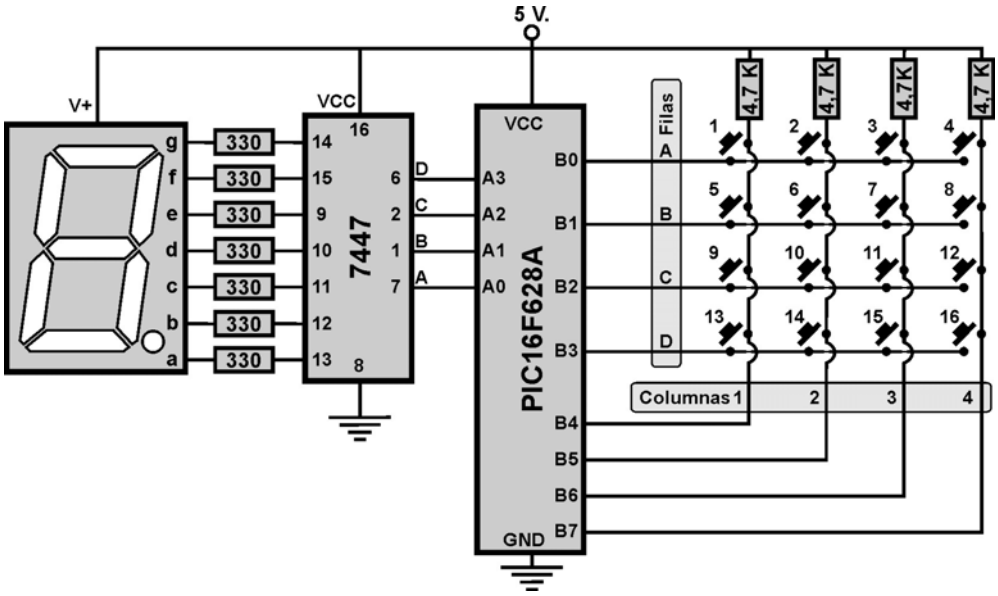


Figura 5.7.1.1. *Diagrama de conexión de un teclado hexadecimal y un display de 7 segmentos.*

```
CMCON=7                 ;convierte en pines digitales el puerto A
fila   VAR   BYTE       ;variable para las filas
colu   VAR   BYTE       ;variable para las columnas
tecla  VAR   BYTE       ;variable para almacenar el número de la tecla
TRISA=0                 ;todo el puerto A configurado como salidas      continúa ...
```

```
prog1:
   PORTB=0                                        ;el puerto B es = %00000000
   TRISB=%11110000            ;configura 4 pines bajos como salida y los demás entrada
   IF ((PORTB >> 4)!=%1111) THEN prog1   ;si la tecla es presionada manténgalo en prog1
prog2:
   FOR fila = 0 TO 3                          ;repetir para las 4 filas del teclado
      PORTB=0                                 ;el puerto B es = %00000000
      TRISB=(DCD fila)^%11111111              ;setea una fila a 1 y los invierte a todos los demás
      colu= PORTB >> 4                        ;desplaza los 4 bits altos al inicio
   IF colu != %1111 THEN numtecla             ;si una tecla es pulsada ir numtecla
      NEXT fila
   GOTO prog2

numtecla:
   tecla = (fila*4)+(NCD (colu^%1111))          ;calcula el valor de la tecla multiplicando
                ;por 4 la fila a la que pertenece  y sumando a la posición que se encuentra
                ;para un teclado de 12 pulsadores, debemos cambiar (fila*3)
   porta=tecla                                  ; sacar por el puerto A el valor de tecla
GOTO prog1
END
```

Figura 5.7.1.2. teclado16-display.pbp *Programa para leer un teclado hexadecimal y mostrarlo en un display de 7 segmentos.*

Debe considerarse que el lugar de las teclas no se pueden cambiar, puesto que este sistema de programación, utiliza operaciones matemáticas para calcular la tecla pulsada, pero debemos reconocer lo pequeño que es el programa, por eso se propone otro modo de programar, en donde los valores se le puede asignar en cualquier lugar, así como también se le puede poner letras.

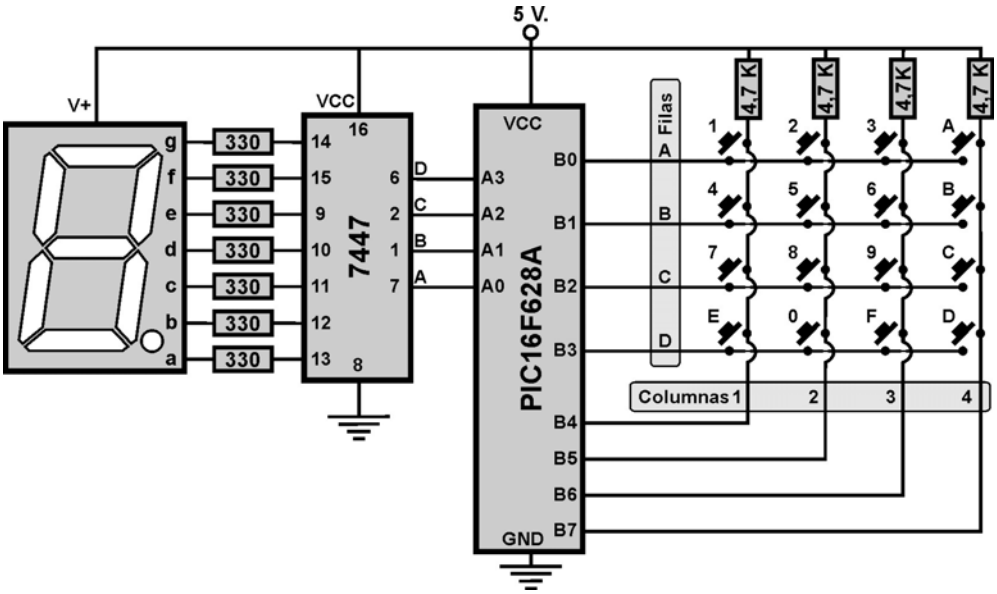


Figura 5.7.1.3. *Diagrama de conexión de un teclado hexadecimal con diferente lugar de teclas.*

```
cmcon=7                                 ;cambiar a modo digital todo el puerto A
TRISA = 0                               ;todo el puerto A es configurado como salidas

A VAR PORTB.0                           ;nombres para los pines de las filas
B VAR PORTB.1
C VAR PORTB.2
D VAR PORTB.3

UNO VAR PORTB.4                         ;nombres para los pines de las columnas
DOS VAR PORTB.5
TRES VAR PORTB.6
CUATRO VAR PORTB.7

BARRIDO:
    LOW A                               ;hacer bajo la 1era fila
  IF UNO    = 0 THEN PORTA = 1          ;si la 1ra tecla es presionada sacar 1
  IF DOS    = 0 THEN PORTA = 2          ;si la 2da tecla es presionada sacar 2
  IF TRES   = 0 THEN PORTA = 3          ;si la 3ra tecla es presionada sacar 3
  IF CUATRO = 0 THEN PORTA = 10         ;si la 4ta tecla es presionada sacar 10
    HIGH A                              ;poner en alto la 1era fila
    LOW B                               ;hacer bajo la 2da fila
  IF UNO    = 0 THEN PORTA = 4          ;si la 1ra tecla es presionada sacar 4
  IF DOS    = 0 THEN PORTA = 5          ; y así sucesivamente
  IF TRES   = 0 THEN PORTA = 6
  IF CUATRO = 0 THEN PORTA = 11
    HIGH B                              ;poner en alto la 2da fila
    LOW C                               ;hacer bajo la 3ra fila
  IF UNO    = 0 THEN PORTA = 7
  IF DOS    = 0 THEN PORTA = 8
  IF TRES   = 0 THEN PORTA = 9
  IF CUATRO = 0 THEN PORTA = 12
    HIGH C                              ;poner en alto la 3ra fila
    LOW D                               ;hacer bajo la 4ta fila
  IF UNO    = 0 THEN PORTA = 14
  IF DOS    = 0 THEN PORTA = 0
  IF TRES   = 0 THEN PORTA = 15
  IF CUATRO = 0 THEN PORTA = 13
    HIGH D                              ;poner en alto la 4ta fila
  PAUSE 10                              ;pausa de 10 milisegundos
  GOTO BARRIDO                          ;continuar con el barrido de teclas

END
```

Figura 5.7.1.4. teclado16-display2.pbp *Programa diferente para leer un teclado hexadecimal y mostrarlo en un display de 7 segmentos.*

Como se puede ver este programa es un poco más largo, pero ocupa menos espacio en la memoria del PIC que el ejercicio anterior, además tiene la ventaja de poder poner en cualquier lugar el valor de las teclas y es más fácil de entender, así que este será la forma que utilizaremos en adelante. Su funcionamiento es sencillo sólo debemos fijarnos cual fila es la que está en **LOW** y

esta es la fila que se está barriendo, si una de las condiciones encuentra la igualdad, pues esta es la tecla pulsada.

Ejemplo: si pulsamos la tecla 6, en algún momento se pondrá en bajo la fila B y detectará un cambio de estado de 1 a 0 en la columna 3 (puerto B.6), por lo que:

```
     LOW B
IF TRES=0 THEN PORTA = 6        ; detecta un cambio de estado y saca por el display el # 6
     HIGH B
```

Debemos considerar que una persona requiere como mínimo 100 milisegundos para presionar una tecla, en ese tiempo el PIC realiza 10 barridos, por lo que de seguro detectará inmediatamente la tecla pulsada.

Para el caso de utilizar un teclado de 12 pulsadores, debemos eliminar una columna, las que corresponde a la tecla A, B, C, y D, es decir eliminamos las siguientes líneas del programa:

```
     IF CUATRO = 0 THEN PORTA= 10
     IF CUATRO = 0 THEN PORTA= 11
     IF CUATRO = 0 THEN PORTA= 12
     IF CUATRO = 0 THEN PORTA= 13
```

En este caso quedaría un teclado con pulsadores del 0 al 9 y dos teclas de propósito especial, la tecla asterisco (*) y la tecla numeral (#).

5.7.2. CERRADURA ELECTRÓNICA CON CLAVE EN MEMORIA FLASH.

Este es un proyecto aplicable en seguridad, se trata de una cerradura electrónica en la cual al ingresar los 4 dígitos correctamente en su teclado, el PIC energiza un relé (puerta), pero si la clave es incorrecta el PIC emite 3 pitos indicando que ingresó una clave errónea y por supuesto que el relé no se conectará, para hacerlo más interesante se le ha agregado sonido a las teclas en el momento de ser pulsadas esto sirve para que el usuario sepa que el PIC reconoció la pulsación, también tiene un programa antirrebote de tecla para asegurarse que ingrese una sola tecla a la vez.

El único inconveniente es que la clave no puede ser cambiada, ya que el número de la combinación (1,2,3,4) se lo grabó en el programa del PIC por lo que reside en la memoria FLASH, más adelante en los próximos proyectos la clave podrá ser cambiada a gusto del usuario, en tal caso este proyecto sirve para poder identificar y diferenciar las 3 memorias que dispone el PIC.

MATERIALES.
-1 teclado matricial hexadecimal como el de la figura 5.7.2.3
-6 resistencia de 4,7 KΩ
-1 resistencia de 330 Ω
-2 transistores 2N3904
-1 chicharra activa (las que suenan directamente al alimentarles con 12 voltios)
-1 relé de 12 voltios de 5 patitas
-1 LED rojo de 5 mm.
-1 Diodo rectificador 1N4007.

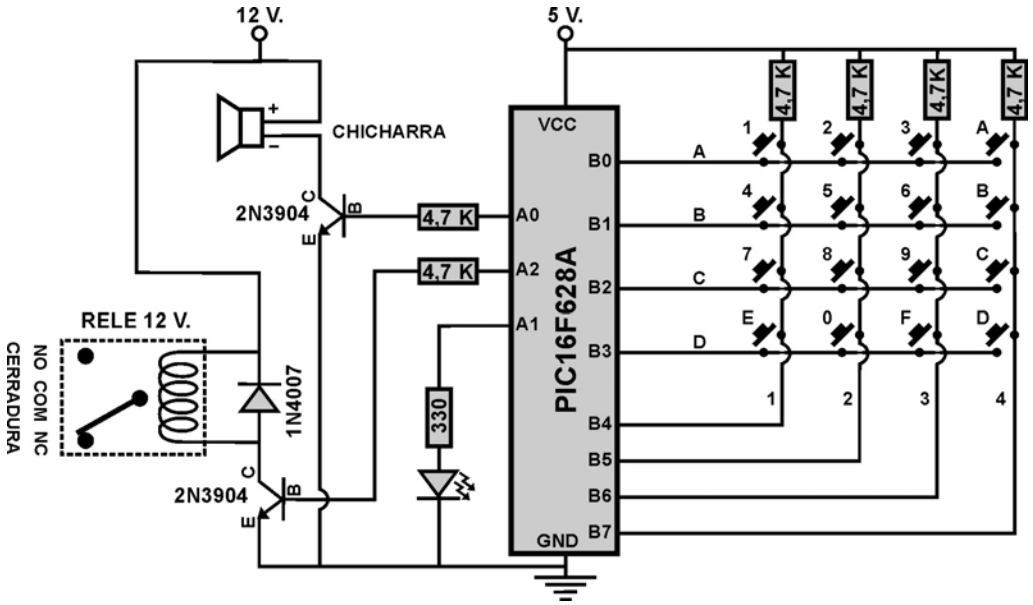


Figura 5.7.2.1. *Diagrama de conexión de un teclado hexadecimal con un relé un led y una chicharra, para hacer una cerradura electrónica.*

```
cmcon=7                          ;cambiar a modo digital todo el puerto A
NUMERO VAR BYTE                  ;variable número para almacenar la tecla pulsada
R VAR BYTE                       ;variable r para hacer repeticiones

BIP VAR PORTA.0                  ;el portA.1  Bip (conectar chicharra activa)
LED VAR PORTA.1                  ;el portA.2 se llamará led
DOOR VAR PORTA.2                 ;el portA.3 conectar relé para la cerradura

A VAR PORTB.0                    ;nombres para los pines de las filas
B VAR PORTB.1
C VAR PORTB.2
D VAR PORTB.3

UNO   VAR PORTB.4                ;nombres para los pines de las columnas
DOS   VAR PORTB.5
TRES  VAR PORTB.6
CUATRO VAR PORTB.7

INICIANDO:                       ;programa del led para saber si está funcionando
   HIGH LED : HIGH BIP
   PAUSE 500
   LOW LED  : LOW BIP

GOTO TECLAUNO                    ;ir a comparar claves

BARRIDO:
     LOW A                       ;sensar la fila A                    continúa ....
```

```
   IF UNO      = 0 THEN NUMERO =1 :RETURN          ;tecla  retorna  cargada con 1
   IF DOS      = 0 THEN NUMERO =2 :RETURN          ;tecla retorna cargada con 2
   IF TRES     = 0 THEN NUMERO =3 : RETURN         ;tecla retorna cargada con 3
   IF CUATRO = 0 THEN NUMERO =10: RETURN           ;tecla  retorna cargada con 10
      HIGH A
      LOW B                                        ;sensar la fila B
   IF UNO      = 0 THEN NUMERO =4 : RETURN
   IF DOS      = 0 THEN NUMERO =5 : RETURN
   IF TRES     = 0 THEN NUMERO =6 : RETURN
   IF CUATRO = 0 THEN NUMERO =11: RETURN
      HIGH B
      LOW C                                        ;sensar la fila C
   IF UNO      = 0 THEN NUMERO =7 : RETURN
   IF DOS      = 0 THEN NUMERO =8 : RETURN
   IF TRES     = 0 THEN NUMERO =9 : RETURN
   IF CUATRO = 0 THEN NUMERO =12: RETURN
      HIGH C
      LOW D                                        ;sensar la fila D
   IF UNO      = 0 THEN NUMERO =14: RETURN
   IF DOS      = 0 THEN NUMERO =0 : RETURN
   IF TRES     = 0 THEN NUMERO =15: RETURN
   IF CUATRO = 0 THEN NUMERO =13: RETURN
      HIGH D
      PAUSE 10
   GOTO BARRIDO

; ***************** programa de antirrebote de teclas *************
PTECLA:
   HIGH LED : HIGH BIP                  ;genera sonido cada que se pulsa tecla
   PAUSE 100                            ;duración 100 milisegundos
   LOW LED : LOW BIP                    ;apagar sonido y led
ESPACIO:                                ;programa de antirrebote de teclas
   IF UNO      = 0 THEN ESPACIO         ;si la tecla sigue pulsada ir a espacio
   IF DOS      = 0 THEN ESPACIO         ;si la tecla sigue pulsada ir a espacio
   IF TRES     = 0 THEN ESPACIO         ;si la tecla sigue pulsada ir a espacio
   IF CUATRO = 0 THEN ESPACIO           ;si la tecla sigue pulsada ir a espacio
   PAUSE 25
   RETURN                               ;retorna si se suelta las teclas

; *****************  comparación de claves ************************

TECLAUNO:
        GOSUB BARRIDO                   ;ir a barrido y retornar con un valor
        GOSUB PTECLA                 ;envía a un programa antirrebote para soltar tecla
 IF NUMERO = 1 THEN TECLADOS            ;si el número es igual a 1 ir teclados
        GOTO FALSO                      ;caso contrario ir a lazo falso

TECLADOS:
        GOSUB BARRIDO: GOSUB PTECLA     ;ir a barrido y retornar con un valor
```

Continúa ...

```
 IF NUMERO = 2 THEN TECLATRES          ;si el número es igual a 2 ir teclatres
       GOTO FALSO1                     ;caso contrario ir a lazo falso

TECLATRES:
       GOSUB BARRIDO :GOSUB PTECLA  ;ir a barrido y retornar con un valor
 IF NUMERO = 3 THEN TECLACUATRO     ;si el número es igual a 3 ir teclacuatro
       GOTO FALSO2                     ;caso contrario ir a lazo falso

TECLACUATRO:
       GOSUB BARRIDO :GOSUB PTECLA  ;ir a barrido y retornar con un valor
 IF NUMERO = 4 THEN OPENGE           ;si el número es igual a 4 conectar relé
       GOTO FALSO3                     ;caso contrario ir a lazo falso

OPENGE:
   FOR R = 1 TO 2                      ;2 pitos indica clave correcta
   PAUSE 100
   HIGH LED : HIGH BIP
   PAUSE 100
   LOW LED : LOW BIP
   NEXT

   HIGH DOOR                           ; se conecta el relé (abrir puerta)
   PAUSE 1000                          ; esperar 1 segundo
   LOW DOOR                            ; desconectar relé
   GOTO TECLAUNO                       ;ir nuevamente a comparar las claves

; ********** lazos falsos teclas erroneas **********
FALSO:
   GOSUB BARRIDO :GOSUB PTECLA      ;estas teclas no comparan ninguna
FALSO1:                                 ;clave sólo espera que termine de
   GOSUB BARRIDO :GOSUB PTECLA      ;pulsar las 4 teclas y no hace nada
FALSO2:
   GOSUB BARRIDO :GOSUB PTECLA

FALSO3:
   FOR R = 1 TO 3                      ;3 pitos indica clave incorrecta
   PAUSE 100
   HIGH LED : HIGH BIP
   PAUSE 100
   LOW LED : LOW BIP
   NEXT
GOTO TECLAUNO                          ;ir nuevamente a comparar las claves
END
```

Figura 5.7.2.2. cerradura_FLASH.pbp *Programa para hacer una cerradura electrónica codificada (1,2,3,4) en la que la clave no se puede cambiar.*

NOTA: Para evitar fallas en su funcionamiento debido a la activación del relé, asegúrese de colocar un capacitor de 0,1 uF paralelo a la alimentación del PIC.

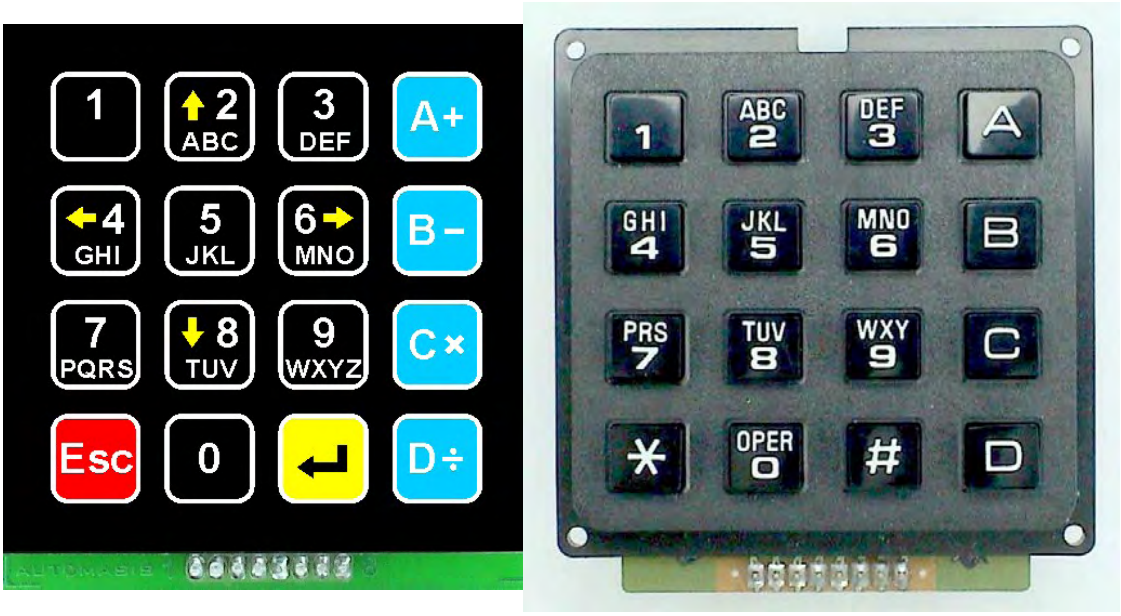


Figura 5.7.2.3. *Fotografía de teclados matriciales hexadecimales comunes en las tiendas electrónicas.*

5.7.3. CERRADURA ELECTRÓNICA CON CLAVE EN MEMORIA RAM Y CAMBIO DE CLAVE.

Este proyecto es muy similar al anterior con la diferencia que este se le puede cambiar la clave predefinida (1,2,3,4) por cualquier otra combinación de teclas, la clave original será cargada en cuatro variables y existirá una manera de cambiar los valores de estas variables lo cual se lo hace de la siguiente manera: después de haber colocado la clave original (1,2,3,4), debemos mantener presionado la tecla D durante 2 segundos, para ser más exactos en el momento que el relé se conecta después de 1 segundo hay una pregunta de si la tecla D es presionada ir a grabar, si no presionamos la tecla D a tiempo, perderemos la oportunidad de cambiar la clave y tendremos que volver a repetir el proceso, en el momento que ingresa al programa de cambio de clave se encenderá el LED y permanecerá encendido esperando a que ingresemos los 4 nuevos dígitos.

Es importante saber que la nueva clave se almacenará en las variables SETPRIME, SETSEGUN, SETERCER, y SETCUART, estas variables ocupan espacio en la memoria RAM (Random Access Memory) o memoria de acceso casual que tiene una capacidad de 224 BYTES, por lo tanto sólo están activas mientras el PIC se encuentra alimentado, una vez que se corta la alimentación al PIC esta memoria se borra (volátil), por consiguiente al momento de volver a prender el micro PIC, la nueva clave se nos habrá perdido y en su lugar se encontrará la clave original (1,2,3,4), esto debido a que esta clave se encuentra en la memoria FLASH y en el momento de correr el programa lo carga nuevamente en la memoria RAM para desde aquí poder ser modificada.

En cuanto a los materiales y el diagrama de conexión, son los mismos que se utilizaron en el ejercicio anterior.

```
cmcon=7                          ;cambiar a modo digital todo el puerto A
NUMERO VAR BYTE                  ;variable número para almacenar la tecla pulsada
R          VAR BYTE              ;variable r para hacer repeticiones

BIP    VAR PORTA.0               ;el portA.1  Bip (conectar chicharra activa)
LED    VAR PORTA.1               ;el portA.2 se llamará led
DOOR VAR PORTA.2                 ;el portA.3 conectar relé para la cerradura

A   VAR PORTB.0                  ;nombres para los pines de las filas
B   VAR PORTB.1
C   VAR PORTB.2
D   VAR PORTB.3

UNO       VAR PORTB.4            ;nombres para los pines de las columnas
DOS       VAR PORTB.5
TRES      VAR PORTB.6
CUATRO VAR PORTB.7

SETPRIME VAR BYTE                ;variable para almacenar la 1era clave
SETSEGUN VAR BYTE                ;variable para almacenar la 2da clave
SETERCER VAR BYTE                ;variable para almacenar la 3era clave
SETCUART VAR BYTE                ;variable para almacenar la 4ta clave

SETPRIME  =1                     ;variable cargada con la 1era clave
SETSEGUN =2                      ;variable cargada con la 2da clave
 SETERCER =3                     ;variable cargada con la 3era clave
SETCUART =4                      ;variable cargada con la 4ta clave

INICIANDO:                       ;programa del led para saber si está funcionando
   HIGH LED : HIGH BIP
   PAUSE 500
   LOW LED  : LOW BIP

GOTO TECLAUNO                    ;ir a comparar claves

GRABAUNO:                        ;programa para cambiar la clave
   GOSUB PTECLA : HIGH LED       ;espera a que suelte las teclas
   GOSUB BARRIDO : GOSUB PTECLA  ;ir a barrido y retorna a un antirrebote
   HIGH LED                      ;mantener encendido el LED
   SETPRIME = NUMERO             ;guardar en setprime el valor de número
GRABADOS:
   GOSUB BARRIDO : GOSUB PTECLA  ;ir a barrido y retorna a un antirrebote
   HIGH LED                      ;mantener encendido el LED
   SETSEGUN = NUMERO             ;guardar el valor de número
GRABATRES:
   GOSUB BARRIDO : GOSUB PTECLA  ;ir a barrido y retorna a un antirrebote
   HIGH LED                      ;mantener encendido el LED
   SETERCER = NUMERO             ;guardar el valor de número
```

continúa ....

```
GRABACUATRO:
   GOSUB BARRIDO : GOSUB PTECLA          ;ir a barrido y retorna a un antirrebote
   HIGH LED                              ;mantener encendido el LED
   SETCUART = NUMERO                     ;guardar el valor de número
GOTO iniciando                           ; ir a iniciando

BARRIDO:
      LOW A                              ;sensar la fila A
IF UNO     = 0 THEN NUMERO =1 :RETURN    ;tecla pulsada retorne  cargada con 1
IF DOS     = 0 THEN NUMERO =2 :RETURN    ;tecla pulsada retorne cargada con 2
IF TRES    = 0 THEN NUMERO =3 :RETURN    ;tecla pulsada retorne cargada con 3
IF CUATRO=0 THEN NUMERO =10:RETURN       ;tecla pulsada retorne  cargada con 10
      HIGH A
      LOW B                              ;sensar la fila B
IF UNO     = 0 THEN NUMERO =4 : RETURN
IF DOS     = 0 THEN NUMERO =5 : RETURN
IF TRES    = 0 THEN NUMERO =6 : RETURN
IF CUATRO=0 THEN NUMERO =11: RETURN
      HIGH B
      LOW C                              ;sensar la fila C
IF UNO     = 0 THEN NUMERO =7 : RETURN
IF DOS     = 0 THEN NUMERO =8 : RETURN
IF TRES    = 0 THEN NUMERO =9 : RETURN
IF CUATRO=0 THEN NUMERO =12: RETURN
      HIGH C
      LOW D                              ;sensar la fila D
IF UNO    = 0 THEN NUMERO =14: RETURN
IF DOS     = 0 THEN NUMERO =0 :RETURN
IF TRES    = 0 THEN NUMERO =15: RETURN
IF CUATRO= 0 THEN NUMERO =13: RETURN
      HIGH D
      PAUSE 10
   GOTO BARRIDO

; ****************** programa de antirrebote de teclas **************
PTECLA:
   HIGH LED : HIGH BIP                   ;genera sonido cada que se pulsa tecla
   PAUSE 100                             ;duración 100 milisegundos
   LOW LED : LOW BIP                     ;apagar sonido y led
  ESPACIO:                               ;programa de antirrebote de teclas
   IF UNO     = 0 THEN ESPACIO           ;si la tecla sigue pulsada ir espacio
   IF DOS     = 0 THEN ESPACIO           ;si la tecla sigue pulsada ir espacio
   IF TRES    = 0 THEN ESPACIO           ;si la tecla sigue pulsada ir espacio
   IF CUATRO= 0 THEN ESPACIO             ;si la tecla sigue pulsada ir espacio
   PAUSE 25
   RETURN                                ;retorna si se suelta las teclas
; ******************  comparación de claves ************************
  TECLAUNO:
   GOSUB BARRIDO                  ;ir a barrido y retornar con un valor       continúa …
```

```
        GOSUB PTECLA                ;envía a un programa antirrebote para soltar tecla
        IF numero = setprime THEN TECLADOS      ;si el número es igual a setprime
        GOTO FALSO                  ;caso contrario ir a lazo falso
TECLADOS:
        GOSUB BARRIDO :GOSUB PTECLA         ;ir a barrido y retornar con un valor
        IF numero = setsegun THEN TECLATRES ;si el número es igual a setsegun
        GOTO FALSO1                 ;caso contrario ir a lazo falso
TECLATRES:
        GOSUB BARRIDO :GOSUB PTECLA         ;ir a barrido y retornar con un valor
        IF numero = setercer THEN TECLACUATRO  ;si el número es igual a setercer
        GOTO FALSO2                 ;caso contrario ir a lazo falso
TECLACUATRO:
        GOSUB BARRIDO :GOSUB PTECLA         ;ir a barrido y retornar con un valor
         IF numero = setcuart THEN OPENGE  ;si número es igual a setcuart conectar relé
        GOTO FALSO3                 ;caso contrario ir a lazo falso

OPENGE:
   FOR R = 1 TO 2                   ;2 pitos indica clave correcta
   PAUSE 100
   HIGH LED : HIGH BIP
   PAUSE 100
   LOW LED : LOW BIP
   NEXT

  HIGH DOOR                         ;se conecta el relé (abrir puerta)
   PAUSE 1000                       ;esperar 1 segundo
   LOW DOOR                         ;desconectar relé
HIGH A: HIGH B :HIGH C :LOW D       ;sensar sólo la fila D
IF CUATRO = 0 THEN GRABAUNO         ;corresponde tecla D grabar
   GOTO TECLAUNO                    ;ir nuevamente a comparar las claves

 ; ****************** lazos falsos teclas erroneas ******************
FALSO:
   GOSUB BARRIDO :GOSUB PTECLA      ;estas teclas no comparan ninguna
FALSO1:                             ;clave sólo espera que termine de
   GOSUB BARRIDO :GOSUB PTECLA      ;pulsar las 4 teclas y no hace nada
FALSO2:
   GOSUB BARRIDO :GOSUB PTECLA
FALSO3:
   FOR R = 1 TO 3                   ;3 pitos indica clave incorrecta
   PAUSE 100
   HIGH LED : HIGH BIP
   PAUSE 100
   LOW LED : LOW BIP
   NEXT
GOTO TECLAUNO                       ;ir nuevamente a comparar las claves
END
```

Figura 5.7.3.1. cerradura_RAM.pbp *Programa para hacer una cerradura electrónica codificada (1,2,3,4) en la que la clave se puede cambiar en la memoria RAM.*

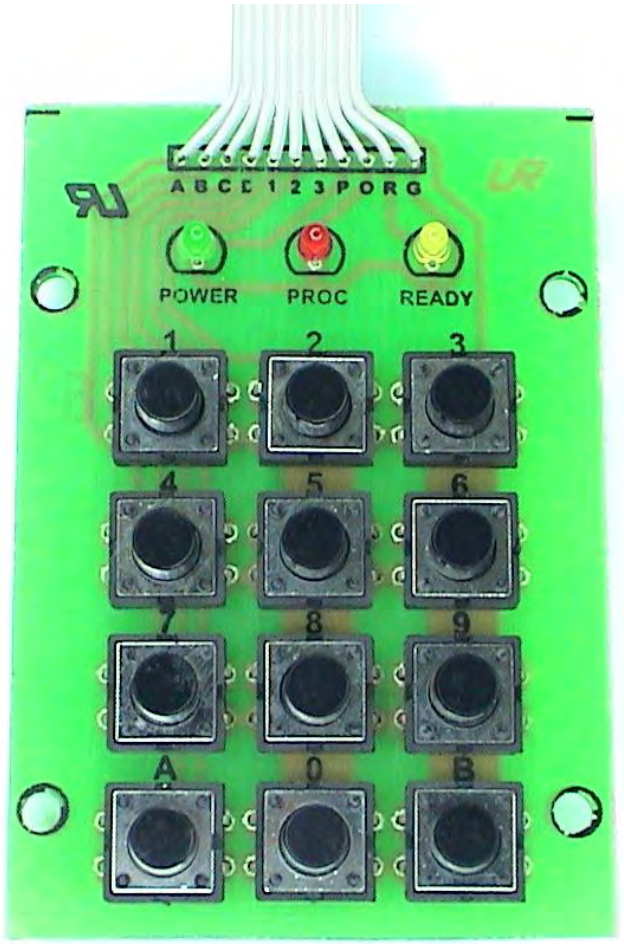


Figura 5.7.3.2. *Fotografía de un teclado de 12 pulsadores y 3 leds indicadores, fabricado en una placa de fibra de vidrio.*

5.7.4. CERRADURA ELECTRÓNICA CON CLAVE EN MEMORIA EEPROM Y CAMBIO DE CLAVE.

Una vez aprendido acerca de las 2 memorias anteriores, es el momento de aprender a utilizar la memoria EEPROM (Electrical Erasable Programable Read Only Memory), o memoria de lectura de programación y borrado eléctrico, que a diferencia de la memoria RAM, esta no es volátil y tiene capacidad para 128 Bytes, lo que quiere decir que si al PIC se le corta la alimentación, los datos almacenados en la memoria EEPROM, permanecen inalterados según su fabricante por un período de 100 años.

Este proyecto cumple todas las expectativas de un electrónico, tiene la posibilidad de cambiar la clave y no borrarse, si se digita una clave errónea el teclado se bloquea y únicamente *lo desbloqueamos presionando al mismo tiempo las teclas 7 y C por 2 segundos*, asimismo para cambiar la clave procedemos como en el proyecto anterior, presionando la *tecla D.*

Algo importante es que si nos olvidamos la clave, la única manera de recuperarlo es leyendo el contenido del PIC a través de programador IC-prog, esto se lo realiza colocando al PIC nuevamente en el grabador y presionando la tecla  , la cual después de leer el contenido del PIC, nos mostrará la nueva clave almacenada en la parte inferior, donde dice Dirección-Datos Eeprom.

En cuanto a los materiales y el diagrama de conexión, son los mismos que se utilizaron en el proyecto anterior.

LA DECLARACIÓN EEPROM, READ Y WRITE. Estas declaraciones las explicaremos con un ejercicio:

EEPROM 5,[3,"K",9,12] ; quiere decir colocar en la memoria EEPROM, dirección 5 el número 3, en la dirección 6 el caracter ASCII de K, es decir el número 75, aunque en el programa IC-prog lo veremos como 4B, esto es porque está en sistema hexadecimal, continuando en la dirección 7 se guardará el número 9 y así sucesivamente, recuerden que el PIC16F628A tiene 128 espacios de memoria EEPROM de 1 BYTE cada una lo que quiere decir que el número más alto que podemos guardar es el 255.

READ 5, pepe ; quiere decir leer la dirección 5 de la memoria EEPROM y guardar en la variable pepe, por consiguiente pepe se carga con el número 3.

WRITE 8, 137 ; quiere decir guardar en la dirección 8 de la memoria EEPROM el número 137, el dato anterior en este caso el 12 automáticamente se borra y se reescribe el número 137.

WRITE 7, pepe ; en este caso la variable pepe estaba cargada con el número 3, por consiguiente la dirección 7 de la memoria EEPROM se borra y se carga con el número 3.

NOTA: Estas declaraciones ya incluyen las PAUSAS necesarias para realizar la grabación, por lo que no es necesario colocar **PAUSE** 10 después de cada declaración, debe considerar también que la memoria EEPROM soporta 1´000.000 de ciclos de borrado/escritura.

```
cmcon=7                         ;cambiar a modo digital todo el puerto A
NUMERO VAR BYTE                 ;variable número para almacenar la tecla pulsada
R VAR BYTE                      ;variable r para hacer repeticiones

BIP     VAR PORTA.0             ;el portA.1  Bip (conectar chicharra activa)
LED     VAR PORTA.1             ;el portA.2 se llamará led
DOOR VAR PORTA.2                ;el portA.3 conectar relé para la cerradura

A VAR PORTB.0                   ;nombres para los pines de las filas
B VAR PORTB.1
C VAR PORTB.2
D VAR PORTB.3
UNO       VAR PORTB.4           ;nombres para los pines de las columnas
DOS       VAR PORTB.5
TRES      VAR PORTB.6
CUATRO  VAR PORTB.7

SETPRIME  VAR BYTE              ;variable para almacenar la 1era clave
SETSEGUN VAR BYTE               ;variable para almacenar la 2da clave
SETERCER  VAR BYTE              ;variable para almacenar la 3era clave
SETCUART VAR BYTE               ;variable para almacenar la 4ta clave

INICIANDO:                      ;programa del led para saber si está funcionando
   FOR R = 1 TO 2                                              continúa ...
```

```
   HIGH LED : HIGH BIP
   PAUSE 1000
   LOW LED  : LOW BIP
   PAUSE 150
   NEXT

;********************* GUARDA LA CLAVE DE FABRICA ********************
    EEPROM 0, [ 1,2,3,4 ]       ;cargar la memoria EEPROM desde la dirección 0 en adelante
;*************************************************************************
RESET:
   FOR R = 1 TO 3
   HIGH LED  : HIGH BIP
   PAUSE 50
   LOW LED  : LOW BIP
   PAUSE 50
  IF (CUATRO=0)AND(UNO=0)THEN RESET          ;corresponden a teclas 7 y C
   NEXT
READ 0,SETPRIME             ;leer el dato de la EEPROM 0 y guardar en setprime
READ 1,SETSEGUN             ;leer el dato de la EEPROM 1 y guardar en setsegun
READ 2,SETERCER             ;leer el dato de la EEPROM 2 y guardar en setercer
READ 3,SETCUART             ;leer el dato de la EEPROM 3 y guardar en setcuart

GOTO TECLAUNO      ;ir a comparar claves

GRABAUNO:                                   ;programa para cambiar la clave
   GOSUB PTECLA : HIGH LED                  ;espera a que suelte las teclas
   GOSUB BARRIDO : GOSUB PTECLA             ;ir a barrido y retorna a un antirrebote
   HIGH LED                                 ;mantener encendido el LED
   WRITE 0,NUMERO                           ;guardar en la EEPROM 0 el valor de número
GRABADOS:
   GOSUB BARRIDO : GOSUB PTECLA             ;ir a barrido y retorna a un antirrebote
   HIGH LED                                 ;mantener encendido el LED
   WRITE 1,NUMERO                  ;guardar en la EEPROM 1 el valor de número
GRABATRES:
   GOSUB BARRIDO : GOSUB PTECLA             ;ir a barrido y retorna a un antirrebote
   HIGH LED                                 ;mantener encendido el LED
   WRITE 2,NUMERO                  ;guardar en la EEPROM 2 el valor de número
GRABACUATRO:
   GOSUB BARRIDO : GOSUB PTECLA             ;ir a barrido y retorna a un antirrebote
   HIGH LED                                 ;mantener encendido el LED
   WRITE 3,NUMERO                  ;guardar en la EEPROM 3 el valor de número
GOTO RESET                         ;ir a reset para cargar el nuevo valor en las variables

BARRIDO:
      LOW A                                          ;sensar la fila A
IF UNO     = 0 THEN NUMERO =1 :RETURN     ;tecla pulsada retorne  cargada con 1
IF DOS     = 0 THEN NUMERO =2 :RETURN     ;tecla pulsada retorne cargada con 2
IF TRES    = 0 THEN NUMERO =3 :RETURN     ;tecla pulsada retorne cargada con 3
IF CUATRO=0 THEN NUMERO =10:RETURN        ;tecla pulsada retorne  cargada con 10
      HIGH A                                                          continua …
```

```
      LOW B                                          ;sensar la fila B
IF UNO     = 0 THEN NUMERO =4 : RETURN
IF DOS     = 0 THEN NUMERO =5 : RETURN
IF TRES    = 0 THEN NUMERO =6 : RETURN
IF CUATRO=0 THEN NUMERO =11: RETURN
      HIGH B
      LOW C                                          ;sensar la fila C
IF UNO     = 0 THEN NUMERO =7 : RETURN
IF DOS     = 0 THEN NUMERO =8 : RETURN
IF TRES    = 0 THEN NUMERO =9 : RETURN
IF CUATRO=0 THEN NUMERO =12: RETURN
      HIGH C
      LOW D                                          ;sensar la fila D
IF UNO     = 0 THEN NUMERO =14: RETURN
IF DOS     = 0 THEN NUMERO =0 :RETURN
IF TRES    = 0 THEN NUMERO =15: RETURN
IF CUATRO= 0 THEN NUMERO =13: RETURN
      HIGH D
      PAUSE 10
   GOTO BARRIDO
; ****************** programa de antirrebote de teclas **************
PTECLA:
   HIGH LED : HIGH BIP                    ;genera sonido cada que se pulsa tecla
   PAUSE 100                              ;duración 100 milisegundos
   LOW LED : LOW BIP                      ;apagar sonido y led
 ESPACIO:                                 ;programa de antirrebote de teclas
   IF UNO     = 0 THEN ESPACIO            ;si la tecla sigue pulsada ir espacio
   IF DOS     = 0 THEN ESPACIO            ;si la tecla sigue pulsada ir espacio
   IF TRES    = 0 THEN ESPACIO            ;si la tecla sigue pulsada ir espacio
   IF CUATRO= 0 THEN ESPACIO              ;si la tecla sigue pulsada ir espacio
   PAUSE 25
   RETURN                                 ;retorna si se suelta las teclas

; *****************  comparación de claves ************************
TECLAUNO:
   GOSUB BARRIDO                                  ;ir a barrido y retornar con un valor
   GOSUB PTECLA                    ;envía a un programa antirrebote para soltar tecla
         IF numero = setprime THEN TECLADOS       ;si el número es igual a setprime
         GOTO FALSO                               ;caso contrario ir a lazo falso
TECLADOS:
         GOSUB BARRIDO :GOSUB PTECLA              ;ir a barrido y retornar con un valor
         IF numero = setsegun THEN TECLATRES      ;si el número es igual a setsegun
         GOTO FALSO1                              ;caso contrario ir a lazo falso
TECLATRES:
         GOSUB BARRIDO :GOSUB PTECLA              ;ir a barrido y retornar con un valor
         IF numero = setercer THEN TECLACUATRO    ;si el número es igual a setercer
         GOTO FALSO2                              ;caso contrario ir a lazo falso
TECLACUATRO:
         GOSUB BARRIDO :GOSUB PTECLA              ;ir a barrido y retornar con un valor
```

continúa ...

```
      IF numero = setcuart THEN OPENGE  ;si número es igual a setcuart conectar relé
      GOTO FALSO3                       ;caso contrario ir a lazo falso

OPENGE:
   FOR R = 1 TO 2                                ;2 pitos indica clave correcta
   PAUSE 100
   HIGH LED : HIGH BIP
   PAUSE 100
   LOW LED : LOW BIP
   NEXT

   HIGH DOOR                                     ;se conecta el relé (abrir puerta)
   PAUSE 1000                                    ;esperar 1 segundo
   LOW DOOR                                      ;desconectar relé
   HIGH A: HIGH B : HIGH C :LOW D                ;sensar sólo la fila D
IF CUATRO = 0 THEN GRABAUNO          ;corresponde a la tecla D para ir a GRABAR
   GOTO TECLAUNO                     ;ir nuevamente a comparar las claves

 ; ****************** lazos falsos teclas erróneas ******************
FALSO:
   GOSUB BARRIDO :GOSUB PTECLA                   ;estas teclas no comparan ninguna
FALSO1:                                          ;clave sólo espera que termine de
   GOSUB BARRIDO :GOSUB PTECLA                   ;pulsar las 4 teclas y no hace nada
FALSO2:
   GOSUB BARRIDO :GOSUB PTECLA

FALSO3:
   FOR R = 1 TO 30                               ;30 pitos indica clave incorrecta
   PAUSE 150
   HIGH LED : HIGH BIP
   PAUSE 150
   LOW LED : LOW BIP
HIGH A: HIGH B :HIGH D :LOW C                    ;sensar sólo la fila C
IF (CUATRO=0)AND(UNO=0)THEN RESET   ;corresponden a teclas 7 y C para resetear
   NEXT

PANICO:
   HIGH LED
   PAUSE 500
   LOW LED
   PAUSE 500
HIGH A: HIGH B :HIGH D :LOW C                    ;sensar sólo la fila C
IF (CUATRO=0)AND(UNO=0)THEN RESET   ;corresponden a teclas 7 y C para resetear
   GOTO PANICO                       ; queda en este lazo para siempre

END
```

Figura 5.7.4.1. cerradura_EEPROM.pbp *Programa para hacer una cerradura electrónica codificada (1,2,3,4) en la que la clave se puede cambiar en la memoria EEPROM.*

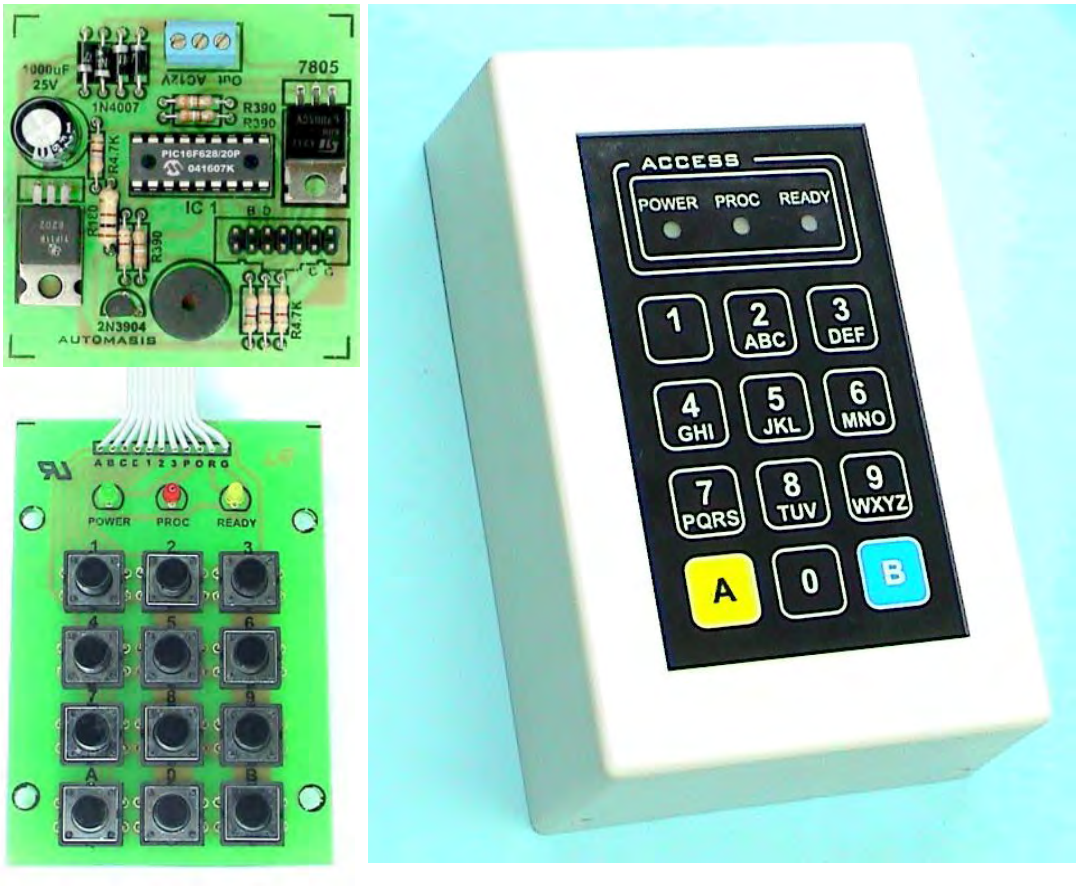


Figura 5.7.4.2. *Fotografía de las partes que componen un control de accesos utilizado para abrir una cerradura eléctrica de 12 v. a través de un TIP110.*

5.7.5. PROYECTO PROPUESTO.

1. Elabore un programa para una alarma con teclado hexadecimal y 3 zonas, en base al proyecto 5.7.4. en el que en vez de energizar el relé arma el sistema de alarma. Con 3 pulsadores simule apertura de zonas y cuando ingresa la clave, emite 10 pitos antes de que el sistema quede completamente armado, esto se conoce como temporizador de salida, luego de esto si se pulsa cualquier botón (zonas), el sistema se activa haciendo sonar a la chicharra, para apagarlo debe ingresar nuevamente la clave de fabrica que en esta ocasión será 6789, esta debe poder ser cambiada a gusto del usuario.

5.8 PROYECTOS CON MOTORES

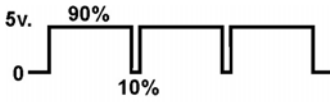
5.8.1. MANEJO DEL PWM COMO VARIADOR DE VELOCIDAD DE UN MOTOR DC.

El PWM (Pulse Width Modulation) o modulación en ancho del pulso, tiene muchas aplicaciones, por ejemplo para atenuar la iluminación de un led, la iluminación del BACKLIGHT de un LCD, para variar la velocidad de un motor DC, que es lo que veremos en este caso. El presente proyecto es un variador de velocidad de un motor DC de juguete que se alimenta a 5 voltios, su funcionamiento es de la siguiente manera:

Al momento de alimentar el circuito, el motor parte desde una velocidad media, es decir (FREC=125), al pulsar el botón (S) incrementa la variable en múltiplos de 25 y la velocidad del motor sube hasta llegar a (FREC=250), si seguimos pulsando la misma tecla, el LED permanecerá encendido, esto nos indica que ya llegó al límite, entonces pulsamos el botón (B), el cual hace que disminuya la velocidad del motor hasta llegar a (FREC=25), de igual manera si seguimos pulsando el botón (B) el LED permanecerá encendido. Si usted no dispone de un motor puede conectar a un LED directamente con una resistencia de 330 Ω, igualmente podrá observar cómo baja o sube la intensidad del LED, la forma de utilizar el PWM es de la siguiente manera:

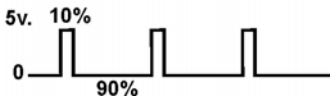
PWM portB.0, 127, 60 ; quiere decir sacar 60 pulsos PWM por el puerto B.0 al 50% en alto
; aproximadamente

La forma de la señal que sale por el PIC es similar a los siguientes gráficos:



PWM portb.0, 228, 3 ;genera 3 ciclos al 90% alto y 10% bajo
;el motor trabaja a velocidad alta

PWM portb.0, 127, 3 ;genera 3 ciclos al 50% alto y 50% bajo
;el motor trabaja a velocidad media



PWM portb.0, 25, 3 ;genera 3 ciclos al 10% alto y 90% bajo
; el motor trabaja a velocidad baja

Por consiguiente 0 representa 0% de ciclo útil y 255 el 100% de nivel alto, el largo de cada ciclo para un oscilador de 4MHZ es de **5 milisegundos** y para un oscilador de 20MHZ es de 1 milisegundo

MATERIALES.
-3 resistencia de 4,7 KΩ
-1 resistencia de 330 Ω
-1 transistor TIP110
-2 pulsadores normalmente abiertos
-1 capacitor cerámico de 0,1 uF (104)
-1 LED rojo de 5 mm.
-1 diodo rectificador 1N4007
-1 motor de juguete.

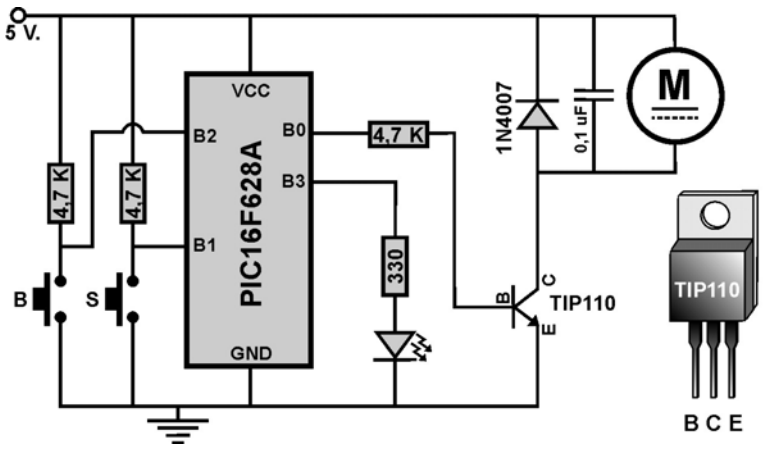


Figura 5.8.1.1. *Conexión de un motor DC para manejar desde el PIC*

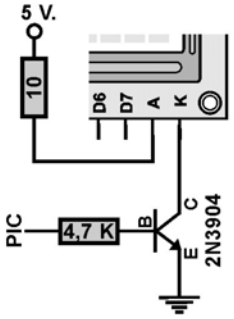


Figura 5.8.1.2. *Conexión del Backlight del LCD, para manipular desde el PIC.*

NOTA: El TIP110, puede manejar un motor de hasta 100 voltios DC a 8 Amperios, para el caso de querer utilizar un motor grande de AC, se recomienda utilizar un optoacoplador a la salida del puerto del PIC.

```
botsubir VAR portb.1                ;nombre para el pin B1
botbajar VAR portb.2                ;nombre para el pin B2
LED      VAR portb.3                ;nombre para el pin B3
FREC     VAR BYTE                   ;variable FREC tamaño 255
bandera VAR BIT                     ;variable bandera de 1 bit
FREC = 125                          ;valor inicial para variable FREC

HIGH LED                            ;encender el led para saber que ya arrancó
PAUSE 500                           ;por medio segundo
LOW LED                             ;apagar el led

PROG:
   PWM PORTB.0,FREC,30              ;sacar PWM 30 ciclos de 125 señal útil
   LOW LED                          ;apagar el led
   IF botsubir =0 THEN SUBIR        ;si se pulsa el botón S ir a subir
   IF botbajar =0 THEN BAJAR        ;si se pulsa el botón B ir a bajar
   bandera =0                       ;bandera cargado con cero
GOTO PROG

SUBIR:
   IF FREC > 249 THEN aviso         ;si supera a 249 ir a aviso
   IF bandera = 1 THEN prog         ;si la bandera esta en 1 salir
   HIGH LED                         ;encender el led
   bandera=1                        ;cargar la bandera con uno
   FREC = FREC +25                  ;sumar 25 a la variable FREC
   GOTO PROG                        ;ir a prog                    continúa ...
```

```
BAJAR:
   IF FREC < 26 THEN aviso          ;si baja de 26 ir a aviso
   IF bandera = 1 THEN prog         ;si la bandera esta en 1 salir
   HIGH LED                         ;encender el led
   bandera=1                        ;cargar la bandera con uno
   FREC = FREC -25                  ;restar 25 a la variable FREC
   GOTO PROG                        ;ir a prog

aviso:
   HIGH LED                         ;encender el led
   GOTO PROG                        ;ir a prog
END
```

Figura 5.8.1.3. PWM motor.pbp *Programa para controlar la velocidad de un motor DC.*

5.8.2. UN CONVERSOR D/A CON EL CI. LM358.

Se puede hacer un pequeño convertidor de digital a análogo para el PWM con una resistencia y un capacitor, pero vamos a proponer realizarlo con el LM358 por sus mejores prestaciones, ya que lograremos mayor rango de voltaje (hasta 32 V.), pero en nuestro caso por motivos experimentales sólo lo conectaremos a los 5 voltios de la misma fuente que está alimentado el PIC, en el caso de un PWM de 255, el LM358 tendrá en su salida 5 V., si sacamos un PWM de 127, tendremos 2,5 V., en definitiva los pulsos que ingresan al LM358 se convierten en salida análoga, desde 0 hasta 5 V.

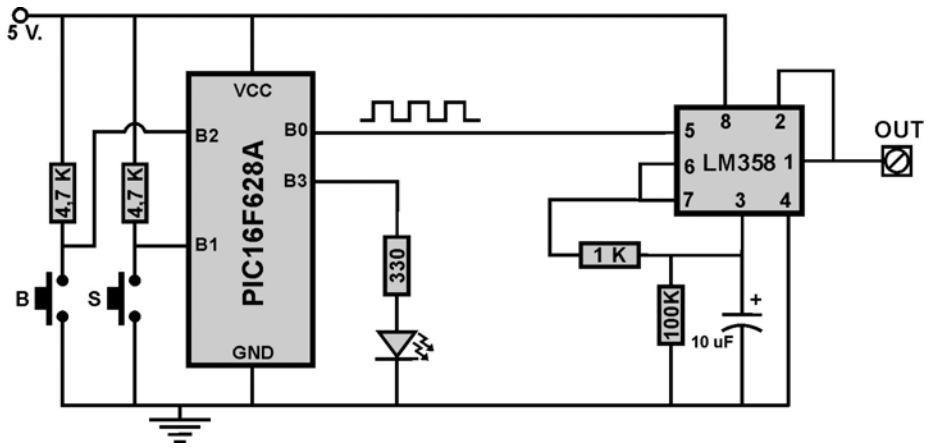


Figura 5.8.2.1. *Conexión de un LM358 como conversor D/A para convertir el PWM en señal análoga de 0 a 5 V.*

Se necesita un voltímetro para medir los niveles de voltaje en la salida, en cuanto al programa podemos utilizar el mismo del ejercicio anterior el PWM_motor.pbp. A la salida del LM 358

podemos colocar un LED con su resistencia de 330 Ω. para poder observar su atenuación, también podemos colocar el circuito del motor de DC, con su capacitor y diodo de protección.

MATERIALES.
-2 resistencia de 4,7 KΩ
-1 resistencia de 330 Ω
-1 resistencia de 1 KΩ
-1 resistencia de 100 KΩ
-1 capacitor electrolítico de 10 uF/100V.
-2 pulsadores normalmente abiertos
-1 LED rojo de 5 mm.
-1 CI. LM358.

5.8.3. LOS MOTORES PASO A PASO BIPOLARES Y UNIPOLARES.

Los motores paso a paso (PAP) son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos, como en robótica, en la tecnología aeroespacial, en maquinarias (tornos, fresadoras, bordadoras), en computadores (CD ROM, Disco duro, DVD, impresoras), etc. A diferencia de los motores de C.C. y los de C.A., los motores PAP tienen la ventaja de poder ser más precisos en cuanto a su velocidad, movimiento, enclavamiento y giros, la señal que requieren para su funcionamiento es de naturaleza digital.

La manera de identificarlos a diferencia de los motores PAP bipolares de 4 hilos y 2 bobinas (ver figura 5.8.3.2.), es que los motores PAP unipolares tienen desde 5 hasta 8 alambres (ver figura 5.8.3.3.) y su funcionamiento es mucho más simple que los motores PAP bipolares, los cuales necesitan un integrado L293 que dispone de 2 puentes H (H-Bridge) o por lo menos debemos hacer un arreglo de 8 transistores, (4 PNP y 4 NPN).

En cuanto al voltaje de alimentación existen desde 1,3V., 1,9V., 4,5V., 5V., 12V. y 24 V., la corriente de consumo de un motor puede variar desde 300mA hasta 3 A.

De acuerdo a la aplicación que deben realizar los motores PAP tienen diferentes grados de precisión como muestra la siguiente tabla:

Grados que gira por impulso Ø	Nro. de pasos para llegar a 360°
0,72°	500
1,8°	200
3,75°	96
7,5°	48
15°	24
90°	4

Figura 5.8.3.1. *Tabla del número de pasos que debe dar un motor PAP para llegar a dar una vuelta completa, según su ángulo de giro.*

El circuito de control para los motores PAP unipolares sea de 5, 6, 8 hilos, es muy sencillo, podemos utilizar un buffer ULN2803 o 4 transistores TIP110 con 4 diodos de protección, para

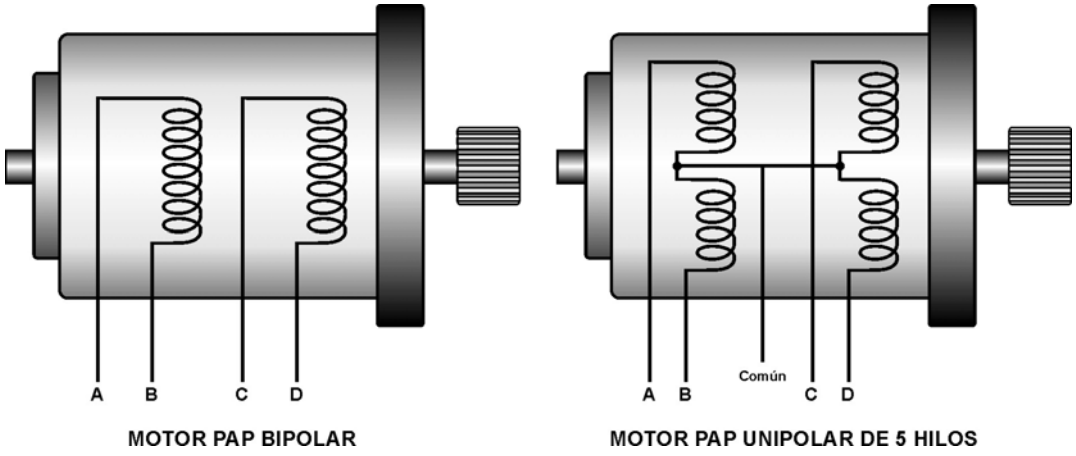


Figura 5.8.3.2. *Diferencia entre un motor PAP bipolar y un motor PAP unipolar de 4 bobinas.*

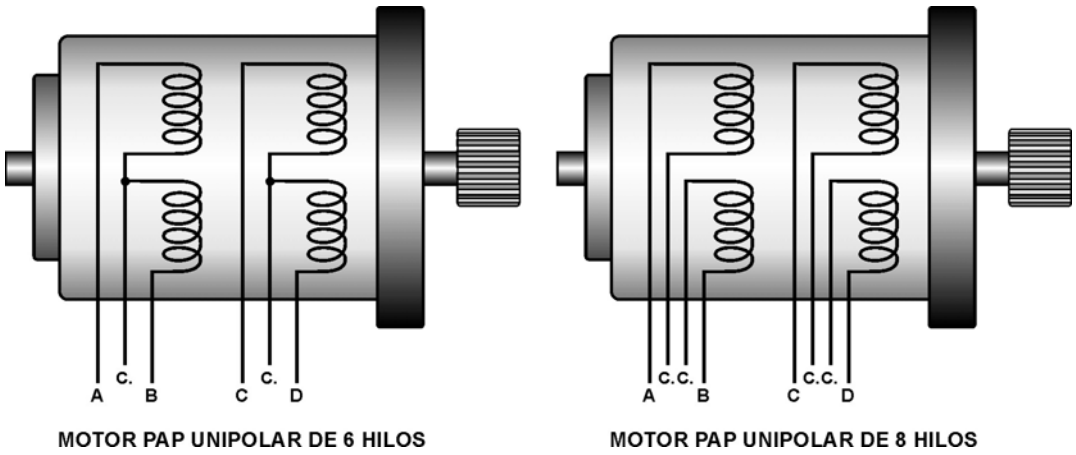


Figura 5.8.3.3. *Motores PAP unipolar de 4 bobinas de 6 y 8 hilos, los cuales se los puede configurar como bipolares.*

que empiece a girar basta con dar una secuencia de pulsos con una duración de 5 milisegundos a cada bobina como muestra la figura 5.8.3.4, mientras que para un motor bipolar se debe invertir la polaridad de cada bobina para que este pueda generar un paso como lo muestra la figura 5.8.3.6.

BOBINA	P 1	P 2	P 3	P 4
A	+V	gnd	gnd	gnd
C	gnd	+V	gnd	gnd
B	gnd	gnd	+V	gnd
D	gnd	gnd	gnd	+V

Figura 5.8.3.4. *Tabla de la secuencia de energizado de bobinas para un motor PAP unipolar para que este gire en sentido antihorario.*

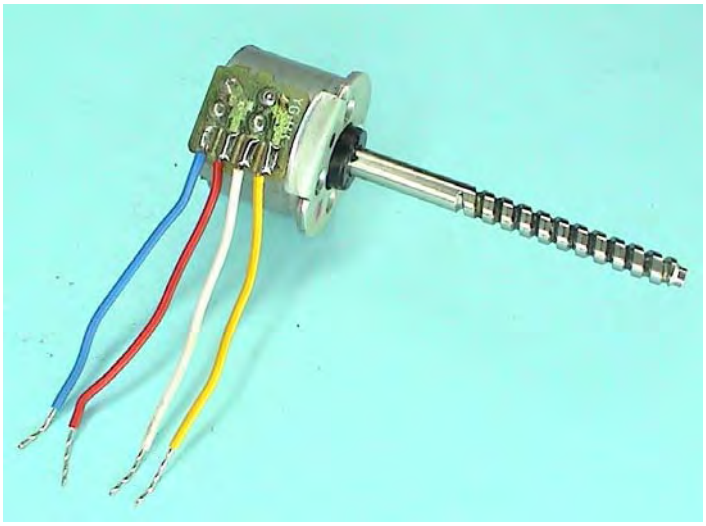


Figura 5.8.3.5. *Fotografía de un Motor PAP bipolar de la disketera de un computador, observen que tiene sólo 4 hilos.*

PASO	A	B	C	D
1	V+	V-	V+	V-
2	V+	V-	V-	V+
3	V-	V+	V-	V+
4	V-	V+	V+	V-

Figura 5.8.3.6. *Tabla de la secuencia de conmutación para un motor PAP bipolar.*

Uno de los mayores inconvenientes a la hora de trabajar con los motores PAP en especial los unipolares es la de poder identificar cual es la bobina A, B, C, y D, para esto simplemente medimos las resistencias de cada una de las bobinas, por ejemplo tomemos el caso de un motor unipolar de 6 hilos cuyos datos de la placa dice:

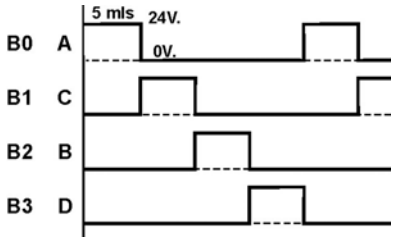
STEPPING MOTOR CBK5-12	
VOLT	24 V
COIL	22 Ω
DEG/STEP	7.5 Ø

Como podemos ver es un motor paso a paso de 7,5 grados de movimiento con una alimentación requerida de 24 V. y cuyas bobinas tienen una resistencia de 22 Ω cada una.

Para identificar qué bobina es la A, debemos buscar el alambre de color amarillo, la B el de color naranja, C el de color negro, el D el de color café, y los dos rojos son los comunes. Para el caso de que no coincidan con esta gama de colores, debemos medir la resistencia entre un cable y otro cable, los que marquen 22 Ω son bobinas común y un terminal y si marca 44 Ω son los terminales A y B o C y D, una vez ubicado cuales son los comunes, estos podemos unirlos en un sólo cable y colocando el motor de la forma que ilustra el gráfico 5.8.3.3, ya podemos asignar los lugares de cada bobina, otra manera de ubicarlos es haciendo pruebas, si un sólo par de cables están conectados incorrectamente el motor no gira y en su lugar permanece temblando, en este caso pruebe cambiando los cables hasta que el motor empiece a girar.

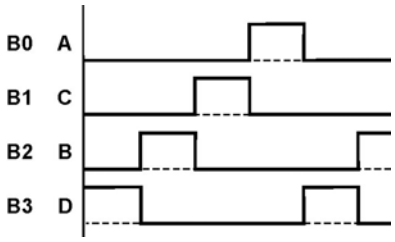
5.8.4. MANEJO DE UN MOTOR PASO A PASO EN SECUENCIA WAVE DRIVE.

Como práctica básica para introducirnos en el manejo de motores PAP vamos a hacer un programa que genere una revolución completa a un motor de 7.5 grados a 24 voltios en secuencia wave drive o secuencia por ola, esta es la forma más fácil de manejar un motor, consiste en energizar una sola bobina a la vez, A, C, B, y por último la D, a continuación veremos la tabla de energizado para conseguir que el motor gire en ambos sentidos.



Bobina	P 1	P 2	P 3	P 4
A	1	0	0	0
C	0	1	0	0
B	0	0	1	0
D	0	0	0	1

Figura 5.8.4.1. *Tabla de energizado de bobinas en secuencia por ola de giro antihorario.*



Bobina	P 1	P 2	P 3	P 4
A	0	0	0	1
C	0	0	1	0
B	0	1	0	0
D	1	0	0	0

Figura 5.8.4.2. *Tabla de energizado de bobinas en secuencia por ola para giro horario.*

Como podemos ver sólo necesitamos activar un pin a la vez durante 5 milisegundos, si ponemos 10 milisegundos esto hará que el motor gire más despacio, pero menos de 4 milisegundos, no serán suficiente para generar el paso y el motor se quedará temblando, adicionalmente se debe poner diodos de protección del colector de cada transistor al voltaje positivo que esté conectado el cable común del motor, esto para proteger al PIC del efecto inductivo que genera el motor. El programa que haremos a continuación hace girar 360° en sentido antihorario, se detendrá por un segundo y luego girará otros 360° en sentido horario, y así indefinidamente, como este es un motor de 7,5° de giro, necesitaremos repetir la secuencia de los pasos 12 veces que multiplicado por 4 pulsos tenemos 48 pasos, luego pruebe con 6 veces y verá que el motor gira 180°.

MATERIALES.
-4 resistencia de 4,7 KΩ
-4 diodos rectificadores 1N4007
-4 transistores TIP110
-1 motor PAP unipolar de cualquier voltaje desde 5 hilos a 8 hilos.

Los motores PAP unipolares de 7,5° los podemos conseguir comúnmente en algunas impresoras de las que ya no utilizamos, estas se encuentran en el mecanismo que mueven los rodillos del alimentador de papel, y algunas impresoras tienen internamente hasta 2 motores PAP.

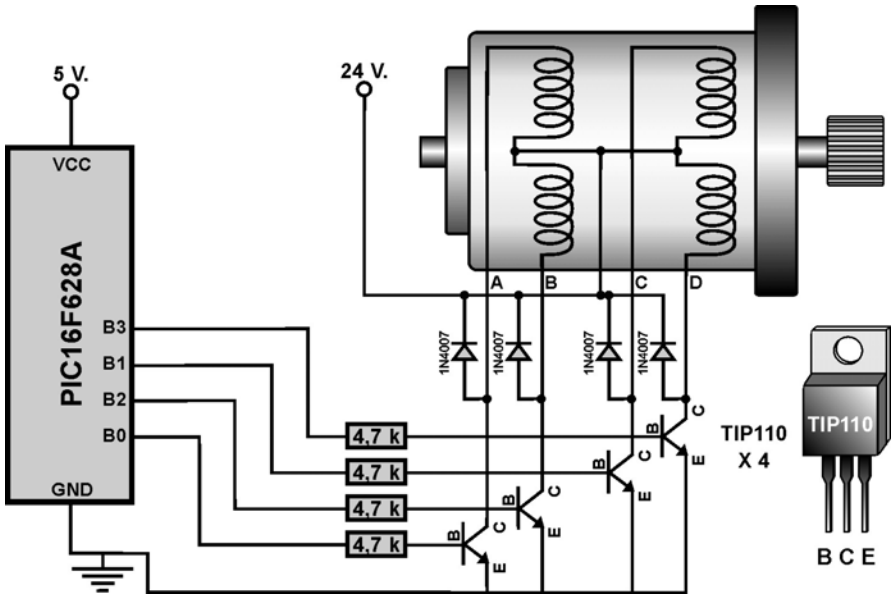


Figura 5.8.4.3. *Conexión de un motor PAP unipolar a las salidas del PIC.*

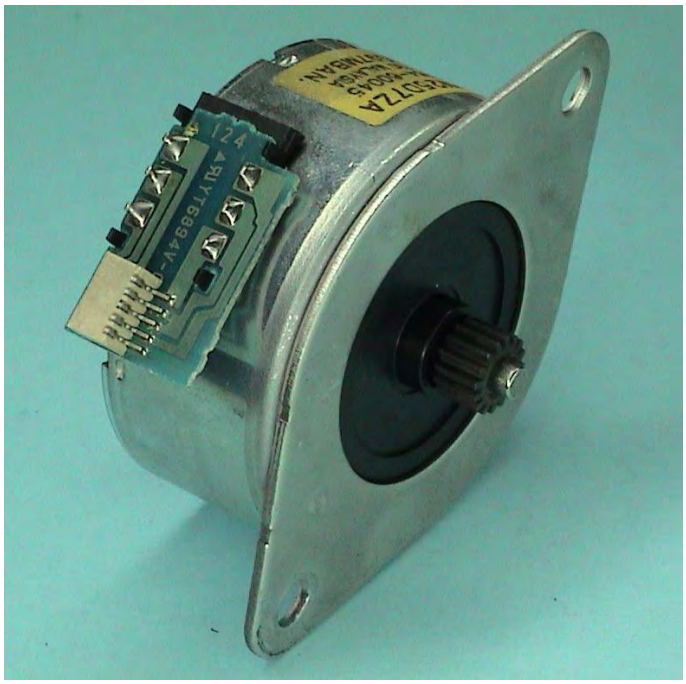


Figura 5.8.4.4. *Fotografía de un motor PAP de una impresora hp 670C.*

```
trisb=0                         ;hacer salidas el puerto b
x VAR BYTE                      ; variable x de 255

antihorario:
   FOR x = 1 TO 12              ;12 veces repetir secuencia de giro antihorario
      portb=%0001               ;energiza bobina A
   GOSUB timer                  ;espera 5 mls
      portb=%0010               ;energiza bobina C
   GOSUB timer                  ;espera 5 mls
      portb=%0100               ;energiza bobina B
   GOSUB timer                  ;espera 5 mls
      portb=%1000               ;energiza bobina D
   GOSUB timer
   NEXT
PAUSE 1000                      ;espera 1 s

   FOR x = 1 TO 12              ;12 secuencias para giro en sentido horario
      portb=%1000               ;energiza bobina D
   GOSUB timer                  ;espera 5 mls
      portb=%0100               ;energiza bobina B
   GOSUB timer                  ;espera 5 mls
      portb=%0010               ;energiza bobina C
   GOSUB timer                  ;espera 5 mls
      portb=%0001               ;energiza bobina A
   GOSUB timer                  ;espera 5 mls
   NEXT
PAUSE 1000                      ;espera 1 s
GOTO antihorario

timer:
   PAUSE 5                      ;pausa de 5 milisegundos
   RETURN
END
```

Figura 5.8.4.5. Motor PAP sec-OLA.pbp *Programa para hacer girar 360° en ambos sentidos a un motor PAP unipolar.*

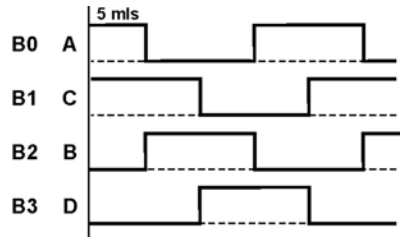
5.8.5. MANEJO DE UN MOTOR PASO A PASO EN SECUENCIA FULL STEP.

También conocida como secuencia por paso completo, este es la manera que recomienda los fabricantes, debido a que siempre se encuentran energizadas 2 bobinas, se obtiene un alto torque de paso y de retención, y consume un 40% más de corriente que el caso anterior.

El siguiente ejercicio hace girar el motor continuamente, noten además la fuerza que tiene tratando de detener el giro con sus dedos.

En cuanto a los materiales y el diagrama de conexión utilizaremos los mismos del ejercicio anterior.

A continuación mostraremos la gráfica de secuencia de pasos y energizado para el manejo del motor a paso completo.



Bobina	P 1	P 2	P 3	P 4
A	1	0	0	1
C	1	1	0	0
B	0	1	1	0
D	0	0	1	1

Figura 5.8.5.1. *Tabla de energizado de bobinas para la secuencia de paso completo.*

```
trisb=0                    ;hacer salidas el puerto b
antihorario:
      portb=%0011          ;energiza bobina A y C
   PAUSE 5                 ;espera 5 mls
      portb=%0110          ;energiza bobina C y B
   PAUSE 5                 ;espera 5 mls
      portb=%1100          ;energiza bobina B Y D
   PAUSE 5                 ;espera 5 mls
      portb=%1001          ;energiza bobina D Y A
   PAUSE 5                 ;espera 5 mls
GOTO antihorario           ;continuar girando
END
```

Figura 5.8.5.2. Motor PAP sec-paso compl.pbp *Programa para hacer girar continuamente en secuencia paso completo.*

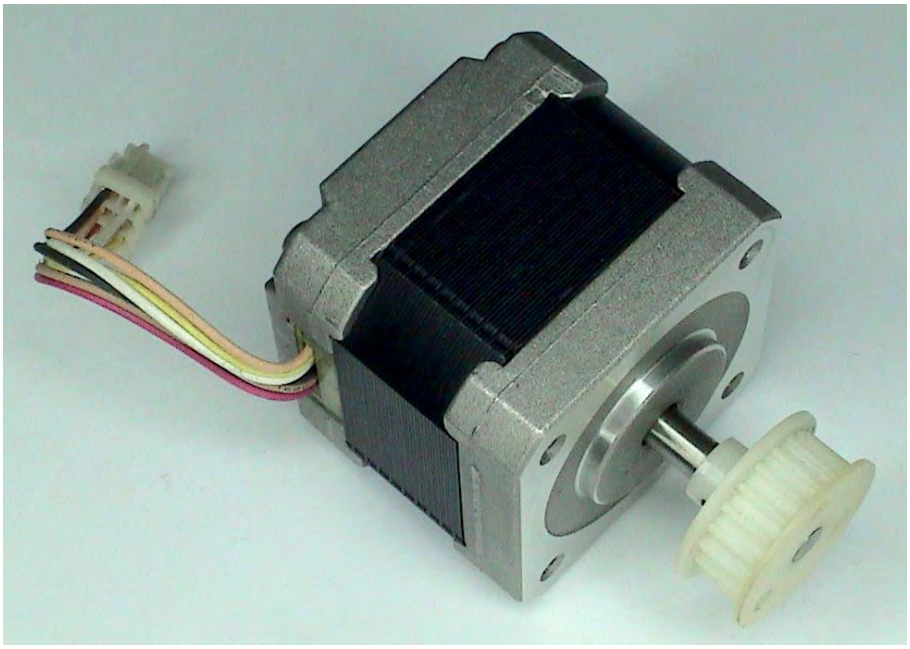


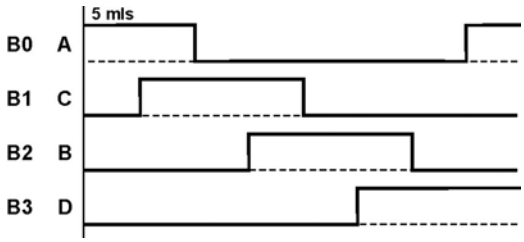
Figura 5.8.5.3. *Fotografía de un motor PAP unipolar de 6 hilos a 6 V. que consume 760 mA.*

5.8.6. MANEJO DE UN MOTOR PASO A PASO EN SECUENCIA HALF STEP.

También conocida como secuencia a medio paso, este es una combinación de las 2 secuencias anteriores, se energiza 2 bobinas, luego 1 bobina, luego otra vez 2 bobinas y así alternadamente, como resultado el rotor avanza medio paso por cada pulso de excitación, la ventaja de esta secuencia es la disminución del avance angular, de 7,5° a 3,75°, por consiguiente para girar una vuelta completa se necesita el doble de pasos, en este caso 96, lo que veremos en el programa será 8 secuencias que multiplicado por 12, nos dará 96 y esto equivale a 360°.

En cuanto a los materiales y el diagrama de conexión utilizaremos los mismos del ejercicio anterior.

A continuación mostraremos la gráfica de secuencia de pasos y energizado para el manejo del motor a medio paso.



Bobina	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8
A	1	1	0	0	0	0	0	1
C	0	1	1	1	0	0	0	0
B	0	0	0	1	1	1	0	0
D	0	0	0	0	0	1	1	1

Figura 5.8.6.1. *Tabla de energizado de bobinas para la secuencia de medio paso.*

```
trisb=0                          ;hacer salidas el puerto b
REPT VAR BYTE                    ;crear variable REPT de 255

FOR REPT = 1 TO 12               ;repetir 12 veces (360 grados)
    portb=%0001                  ;energizar bobina A
  PAUSE 5                        ;retardo de 5 mls
    portb=%0011                  ;energizar bobina A y C
  PAUSE 5                        ;retardo de 5 mls
    portb=%0010                  ;energizar bobina C
  PAUSE 5                        ;retardo de 5 mls
    portb=%0110                  ;energizar bobina C y B
  PAUSE 5                        ;retardo de 5 mls
    portb=%0100                  ;energizar bobina B
  PAUSE 5                        ;retardo de 5 mls
    portb=%1100                  ;energizar bobina B y D
  PAUSE 5                        ;retardo de 5 mls          continúa ...
```

```
    portb=%0100                 ;energizar bobina B
  PAUSE 5                       ;retardo de 5 mls
    portb=%1100                 ;energizar bobina B y D
  PAUSE 5                       ;retardo de 5 mls
    portb=%1000                 ;energizar bobina D
  PAUSE 5                       ;retardo de 5 mls
    portb=%1001                 ;energizar bobina A y D
  PAUSE 5                       ;retardo de 5 mls
NEXT                            ;siguiente repetición

END                             ;fin del movimiento
```

Figura 5.8.6.2. Motor PAP sec-medio paso.pbp *Programa para hacer girar 360°, en sentido antihorario con un avance de medio paso.*

NOTA: En este caso observarán que al terminar de dar la vuelta completa, quedan energizadas las bobinas A y D, por lo tanto el motor quedará retenido fuertemente y empezará a sobrecalentarse, para que esto no suceda adicionen después del **NEXT** la línea portb=%0000, con esto no queda ninguna bobina energizada y por consiguiente notarán que el rotor gira libremente.

5.8.7. PROYECTOS PROPUESTOS CON MOTORES.

1. Utilizando PWM, y sin pulsadores haga que el motor de un juguete baje y suba su velocidad gradualmente.

2. Utilizando PWM un LM358 y 2 pulsadores, genere voltajes variables desde 0V. hasta 12 Voltios.

3. Con un Motor paso a paso unipolar haga que avance 90º y se detenga por 1 seg. luego otros 90º y se detenga igualmente por 1 seg. así debe continuar indefinidamente.

4. Con un motor PAP unipolar haga girar 2 vueltas completas en sentido horario y luego una vuelta en sentido antihorario, el proceso debe repetirse 5 veces, al final el motor debe detenerse por 3 segundos y volver a repetir el proceso.

5. Con un motor PAP unipolar y 2 pulsadores, haga que invierta el sentido de giro con cada pulsador, el motor debe estar en constante movimiento.

5.9 COMUNICACIÓN

5.9.1. ¿QUÉ ES LA COMUNICACIÓN SERIAL?.

Existen dos formas de realizar una comunicación binaria, la paralela y la serial. La comunicación paralela como por ejemplo la comunicación del PIC con el CI. 7447 del ejercicio 5.4.1., en donde los datos viajan simultáneamente a través de los 4 hilos, tiene la ventaja de que la transferencia de datos es más rápida, pero el inconveniente es que necesitamos un cable por cada bit de dato, lo que encarece y dificulta el diseño de las placas, otro inconveniente es la capacitancia que genera los conductores por lo que la transmisión se vuelve defectuosa a partir de unos pocos metros.

La comunicación serial en cambio es mucho más lenta debido a que transmite bit por bit pero tiene la ventaja de necesitar menor cantidad de hilos, y además se puede extender la comunicación a mayor distancia, por ejemplo; en la norma RS232 a 15 mts., en la norma RS422/485 a 1200mts y utilizando un MODEM, pues a cualquier parte del mundo.

Existen dos formas de realizar la comunicación serial: la sincrónica y la asincrónica, la diferencia entre estas dos formas de comunicación es que la comunicación sincrónica además de la línea para la transmisión de datos, necesita otra línea que contenga los pulsos de reloj, estos a su vez indican cuando un dato es válido. Por otra parte la comunicación serial asincrónica no necesita pulsos de reloj, en su lugar utiliza mecanismo como referencia tierra (RS232) o voltajes diferenciales (RS422/485), en donde la duración de cada bit es determinada por la velocidad de transmisión de datos que se debe definir previamente entre ambos equipos.

5.9.2. MODOS DE TRANSMISIÓN DE DATOS.

Se incluye este literal para poder entender mejor las prácticas que más adelante realizaremos, pues mencionaremos algunas palabras que podría encontrar su significado en este literal.

Los modos de transmisión de datos se dividen en cuatro tipos y estos son:

5.9.2.1. Simplex. Se dice a la transmisión que puede ocurrir en un sólo sentido, sea sólo para recibir o sólo para transmitir. Una ubicación puede ser un transmisor o un receptor, pero no ambos a la vez, un ejemplo claro es la radiodifusión, en donde la estación es el transmisor y los radios son los receptores.

5.9.2.2. Half-duplex. Se refiere a la transmisión que puede ocurrir en ambos sentidos pero no al mismo tiempo, en donde una ubicación puede ser un transmisor y un receptor, pero no los dos al mismo tiempo, un ejemplo son los llamados radios WALKING TALKING, en donde un operador presiona el botón y habla, luego suelta el botón y el otro usuario presiona el botón para contestar.

5.9.2.3. Full-duplex. Se dice a la transmisión que puede ocurrir en ambos sentidos y al mismo tiempo, también se los conoce con el nombre de líneas simultaneas de doble sentido, una ubicación puede transmitir y recibir simultáneamente, siempre y cuando la estación a la que está transmitiendo también sea la estación de la cual está recibiendo un ejemplo es la telefonía móvil.

5.9.2.4. Full/full-duplex. Con este modo de transmisión es posible transmitir y recibir simultáneamente, pero no necesariamente entre las dos ubicaciones, es decir una estación puede transmitir a una segunda estación y recibir de una tercera estación al mismo tiempo. Esta transmisión se utilizan casi exclusivamente con circuitos de comunicación de datos.

5.9.3. COMUNICACIÓN SERIAL RS232.

La norma RS232 se incluye actualmente en los computadores, conocido como puerto serial y sirve para comunicarse con otras computadoras además del mouse, programadores, impresoras, etc. A continuación veremos un gráfico que muestra la forma de comunicación serial.

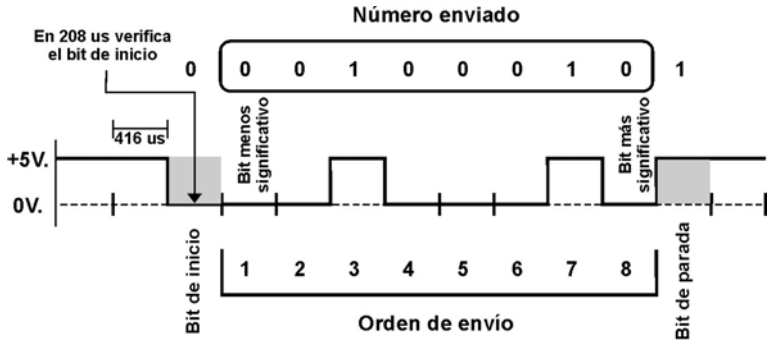


Figura 5.9.3.1. *Estructura de un dato que se envía serialmente a 2400,8N1, (2400bits /seg, sin paridad, 8 bits de dato y 1 bit de parada), correspondiente al número 68 caracter ASCII de "D" (%01000100), el tiempo de un bit es de 416 µs., por lo que el receptor revisa el bit de arranque después de 208 µs., y luego cada 416 µs.*

Como podemos ver la señal permanece en un nivel lógico alto mientras no realiza ninguna transferencia de datos. Para empezar a transmitir datos el transmisor coloca la línea en nivel bajo durante el tiempo de un bit (416 µs para 2400bits/s), este se llama el bit de arranque, a continuación empieza a transmitir con el mismo intervalo de tiempo los bits de datos, que pueden ser de 7 u 8 bits, comenzando por los bits menos significativos y terminando por los más significativos. Al final de la transmisión de datos se envía el bit de paridad, si estuviera activa esta opción y por último los bits de parada, que pueden ser 1 o 2, después de esto la línea vuelve a un estado lógico alto, y el transmisor está listo para enviar el siguiente dato.

Como el receptor no está sincronizado con el transmisor desconoce el momento en que empieza la transmisión, por lo que siempre debe estar en espera del cambio de estado o sea el bit de arranque, una vez que se da este bit, medio bit después vuelve a verificar si está en bajo, si no lo está no lo recibe ya que pudo ser ocasionado por un ruido en la línea, caso contrario si el estado sigue siendo bajo, empieza a recibir la transmisión hasta el bit de parada.

Para que la lectura de los datos sea correcta, ambos equipos deben estar configurados a la misma velocidad y demás parámetros y no exceder más allá de los 2 metros, pasado esta distancia los datos recibidos pueden no ser los correctos debido a la pérdida de voltaje en el cable, ruido, etc. Para distancias mayores existe el protocolo RS232, cuyos niveles de voltaje están establecidos de la siguiente manera: para señal 1 lógica (-5V a –15V) en el transmisor y (-3V a –25V) en el receptor, para señal 0 lógica (+5V a +15 V) en el transmisor y (+3V a +25V) en el receptor, es decir una lógica inversa.

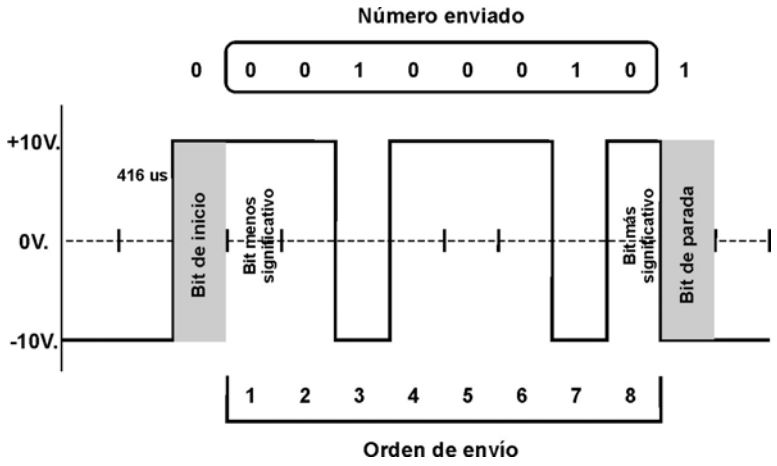


Figura 5.9.3.2. *Comunicación serial con la norma RS232, el dato enviado es el mismo que el de la figura 5.9.3.1., con la diferencia que la lógica es inversa, 1 equivale a –10 y 0 a +10.*

5.9.4. COMUNICACIÓN SERIAL PIC A PC.

Una vez comprendido la teoría de la comunicación serial y su protocolo RS232, haremos un ejercicio de comunicación serial asincrónica modo simplex, que consiste en enviar datos, más específicamente los caracteres ASCII de la palabra "DOG", a través de un cable y directamente desde el PIC al PC, a 2400 bits/seg., a 8 bits de datos, sin paridad, y 1 bit de parada. Como sabemos el computador tiene al menos un puerto serial, con la norma RS232, por lo tanto debemos simular esos voltajes desde el PIC, esto lo conseguimos enviando 0 para representar el 1 lógico y 5V. para representar el 0 lógico, para esto existe la declaración **SEROUT**.

LA DECLARACIÓN SEROUT. Esta declaración sirve para enviar datos seriales en un formato estandar asincrónico usando 8 bits de dato, sin paridad y 1 stop bit, (8N1), y para poder utilizarlo

SEROUT puerto B.1, N2400,["D"] ;enviar el caracter ASCII "D"por el puerto B1 a 24008N1, en dato invertido (figura 5.9.4.1.).

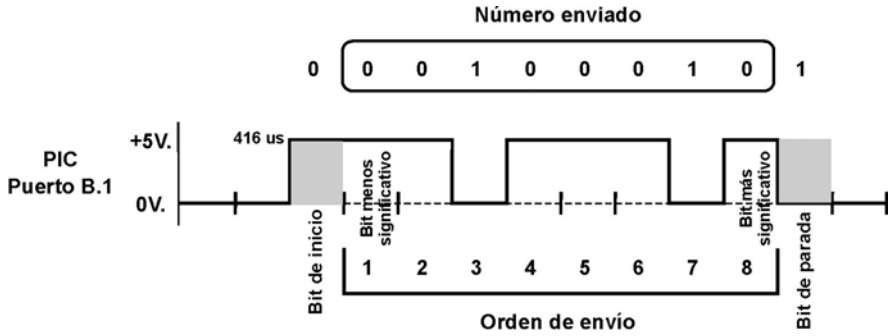


Figura 5.9.4.1. *Esquema del dato enviado por el PIC simulando la norma RS232, noten que es muy similar al esquema 5.9.3.2. pero con diferentes niveles de voltaje.*

debemos incluir al comienzo del programa la siguiente línea:

INCLUDE "modedefs.bas" ;incluir el programa modedefs.bas(modos de comunicación)

Esto significa incluir el programa modedefs.bas en esta línea, aquí se encuentran algunos de los parámetros para las comunicaciones, por ejemplo en nuestro caso las velocidades de transmisión que son: para dato invertido N300, N1200, N2400, N9600, y para dato verdadero: T300, T1200, T2400, T9600. Los datos invertidos por ejemplo el N2400, quiere decir que un 1 lógico vale 0V. y un 0 Lógico vale 5V. (ver figura 5.9.4.1.), en cambio para dato verdadero por ejemplo el T2400 el 1 lógico vale 5V. y el 0 lógico vale 0V. esta señal saldría muy similar al de la figura 5.9.3.1. y este es el que se utiliza para manejar con el CI. MAX232, el cual ya veremos más adelante.

Este comando **INCLUDE** podemos utilizarlo para nuestros propios programas por ejemplo si ponemos **INCLUDE** "freqout.pbp", se incluirá el sonido para un parlante por el puerto B0 que durará 2 segundos.

MATERIALES.
-1 conector DB9 hembra con su respectivo cajetín
-2 resistencias uno de 220 Ω y otro de 330 Ω
-2 mts de cable de 2 hilos para transmisión de datos
-1 led rojo 5mm
-1 cristal de 4 MHZ.
-2 capacitores de 22 pF.

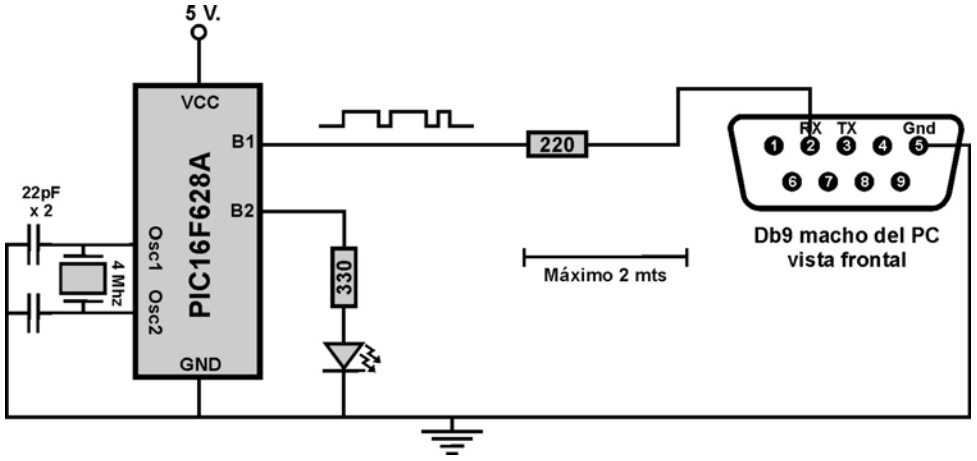


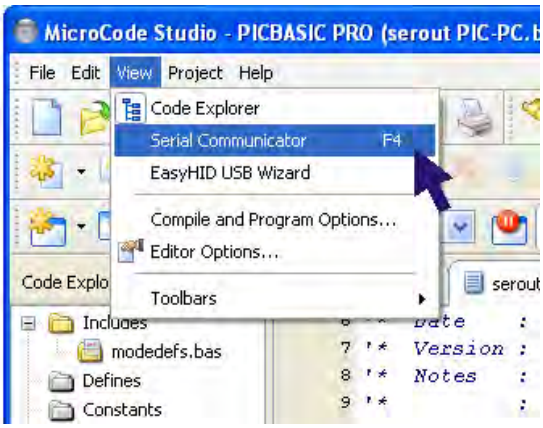
Figura 5.9.4.2. *Diagrama de conexión del PIC para enviar datos al PC sin el CI. MAX232.*

```
INCLUDE "modedefs.bas"          ;incluyen los modos de comunicación
@ device XT_OSC                 ;cambia a oscilador XT en el IC-Prog.
Inicio:
HIGH portb.2                    ;led prueba de funcionamiento
PAUSE 1000
LOW portb.2
PAUSE 500                                              continúa...
```

```
SEROUT portb.1, N2400, ["DOG"]    ;enviar serialmente a 24008N1 los caracteres "DOG"

GOTO inicio                       ;volver a repetir el proceso
END
```

Figura 5.9.4.3. Serout PIC-PC.pbp *Programa para transmitir serialmente desde el PIC a un PC sin utilizar el CI. MAX232.*

Una vez que tenemos listo el proyecto necesitamos una ventana de comunicación serial como el Hyper terminal o la misma ventana de comunicación serial que dispone microcode, para esto presione en la pantalla de microcode **F4** o abra View\Serial communicator y configuramos los parámetros que necesitemos, en este caso 2400\N\8\1.



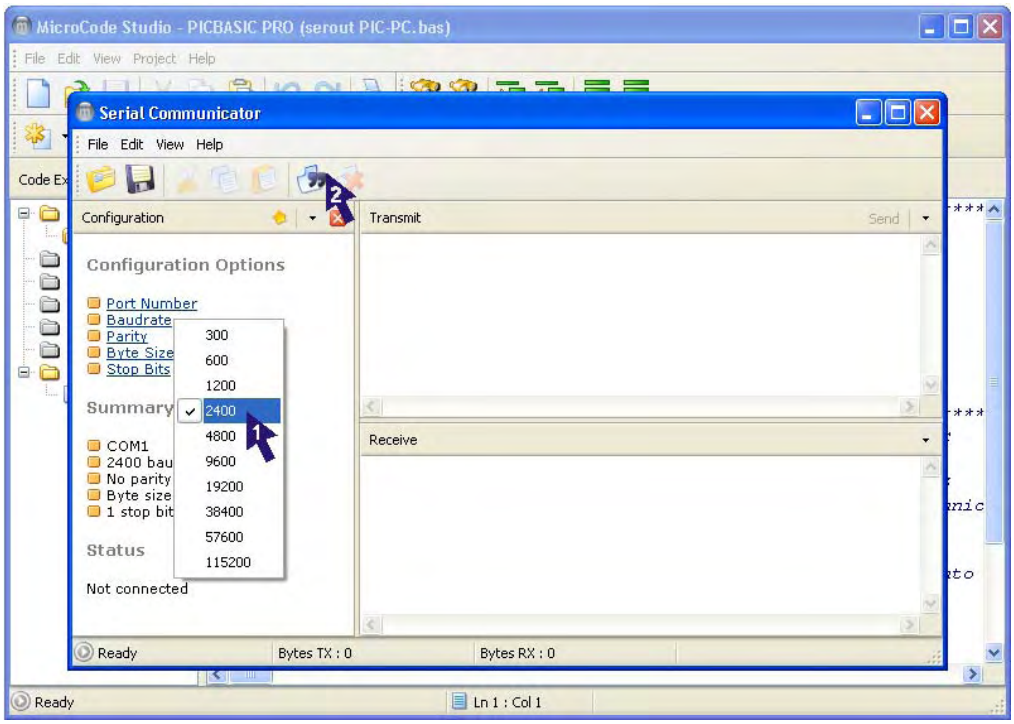


Figura 5.9.4.4. *Pantalla de la ventana de comunicación serial que dispone microcode.*

Para este ejercicio debemos seleccionar el puerto com que vamos a utilizar, luego la velocidad que se transmite el dato, en este caso a 2400 baud, paridad ninguna, 8 bits de datos y 1 stop bit, una vez que estemos listos para iniciar la comunicación presionamos el botón  y notarán en la parte inferior izquierda que decía **Ready** cambia a **Connected**…

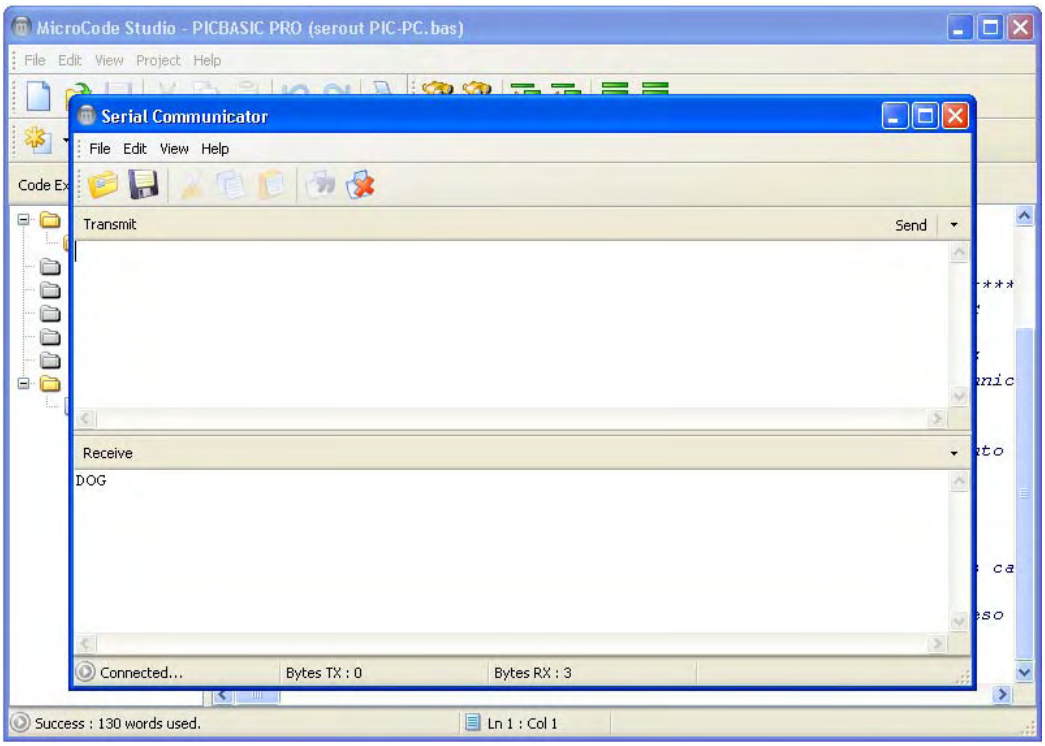


Figura 5.9.4.5. *Pantalla de la ventana de comunicación serial activada.*

Cuando la ventana está activa sale un mensaje en la parte inferior izquierda Connected, y los Bytes enviados o recibidos. Encienda el micro PIC y después de apagarse el LED del puerto B.2 deberá salir el texto enviado en el cuadro que dice Receive, como el programa está en un lazo sin fin el texto DOG seguirá saliendo continuamente.

NOTA: Es importante utilizar un cristal de 4 MHZ para que este proyecto funcione correctamente, sólo así los tiempos de transmisión serán los correctos, si se utiliza el oscilador interno del PIC16F628A, pueda que visualice datos erróneos en la pantalla.

5.9.5. COMUNICACIÓN SERIAL PC A PIC.

Se trata de enviar datos desde el PC al PIC, por lo que es de suponer los voltajes serán desde -10V. hasta +10V. y la distancia podemos extenderlo hasta 15 mts. sin ningún problema, como la conexión es directamente al PIC debemos colocar una resistencia de 22K para no dañar el puerto

del PIC. En este caso el PIC es un receptor por lo que debe permanecer en espera del bit de inicio, para esto tenemos la declaración **SERIN**.

LA DECLARACIÓN SERIN. Esta declaración sirve para recibir datos seriales en un formato estandar asincrónico usando 8 bits de dato, sin paridad y 1 stop bit, (8N1), y para poder utilizarlo debemos incluirlo igualmente que para el SEROUT la línea **INCLUDE** "modedefs.bas" al inicio del programa, su forma de utilizar es la siguiente:

```
SERIN portb.0,N2400,letra        ;esperar un dato serial y lo guarda en la variable
                                 ;previamente creada llamada letra
```

Es importante saber que esta declaración detiene el programa esperando el bit de inicio, y solamente cuando haya recibido un dato, continúa con la siguiente línea de programa, para hacer que sólo espere un determinado tiempo y luego continúe con la siguiente línea debe utilizar Timeout, para mayor información refiérase al manual de pbp, o utilice una interrupción externa para atender el dato a recibir, ver literal **5.10.1 Utilizando la interrupción en el puerto B.0.**

El presente proyecto consiste en enviar un texto desde la ventana de comunicación serial de microcode hacia el PIC y este a su vez este lo mostrará en un LCD 2x16.

MATERIALES.
-1 conector DB9 hembra con su respectivo cajetín
-2 resistencias uno de 10 Ω y otro de 22 KΩ.
-2 mts de cable de 2 hilos para transmisión de datos.
-1 cristal de 4 MHZ.
-2 capacitores de 22 pF.
-1 LCD 2 x 16
-1 potenciómetro de 10 KΩ

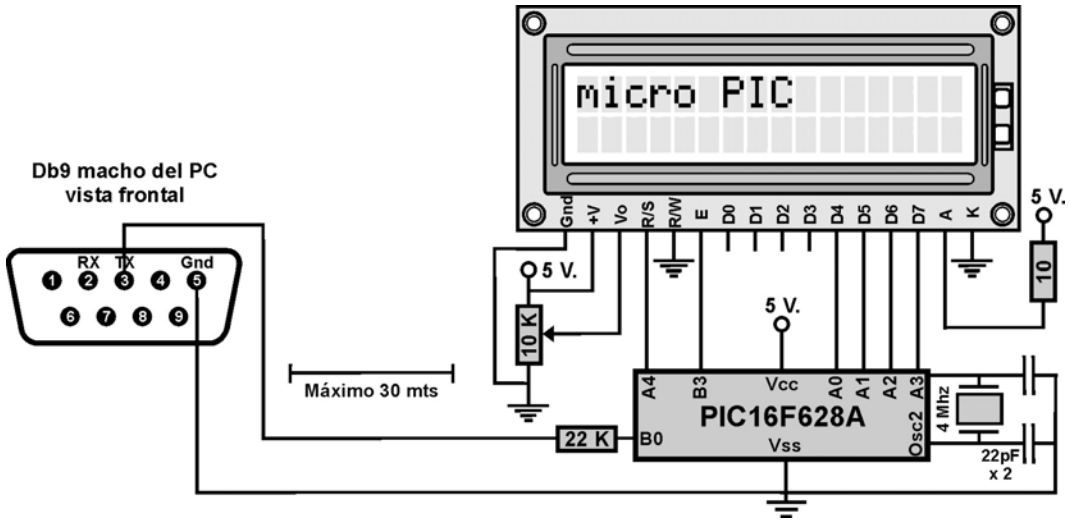


Figura 5.9.5.1. *Diagrama de conexión del PIC para recibir datos del PC sin CI. MAX232.*

```
INCLUDE "modedefs.bas"                ;incluyen los modos de comunicación
@ device XT_OSC                       ;cambia a oscilador XT en el IC-Prog.
dat VAR BYTE                          ;variable de almacenamiento de 255

LCDOUT $fe,1, " LCD listo"            ;texto para verificar la conexión
PAUSE 1000                            ;espera 1 seg. Con el texto "LCD listo"

LCDOUT $fe,1                  ;limpia pantalla del Lcd y ubica cursor en casilla $80

Inicio:
        SERIN portb.0, N2400, dat     ;esperar el dato y guardarlo en dat
        LCDOUT, dat                   ;desplegar el dat en LCD
GOTO inicio                           ;ir a esperar el siguiente caracter

END
```

Figura 5.9.5.2. Serin PC-PIC.pbp *Programa para recibir datos serialmente desde el PC a un PIC sin utilizar el CI. MAX232.*

Una vez que arranca el PIC saldrá un texto que dice LCD listo, un segundo después se borrará y quedará en un lazo de espera del dato serial, si el texto inicial no sale, revise las conexiones al LCD, caso contrario si todo está bien, abrimos la ventana de comunicación serial de microcode de la forma que se aprendió anteriormente y escribimos en la ventana que dice **Transmit**: micro PIC, luego presione **Send**, inmediatamente aparecerá el texto en el LCD.

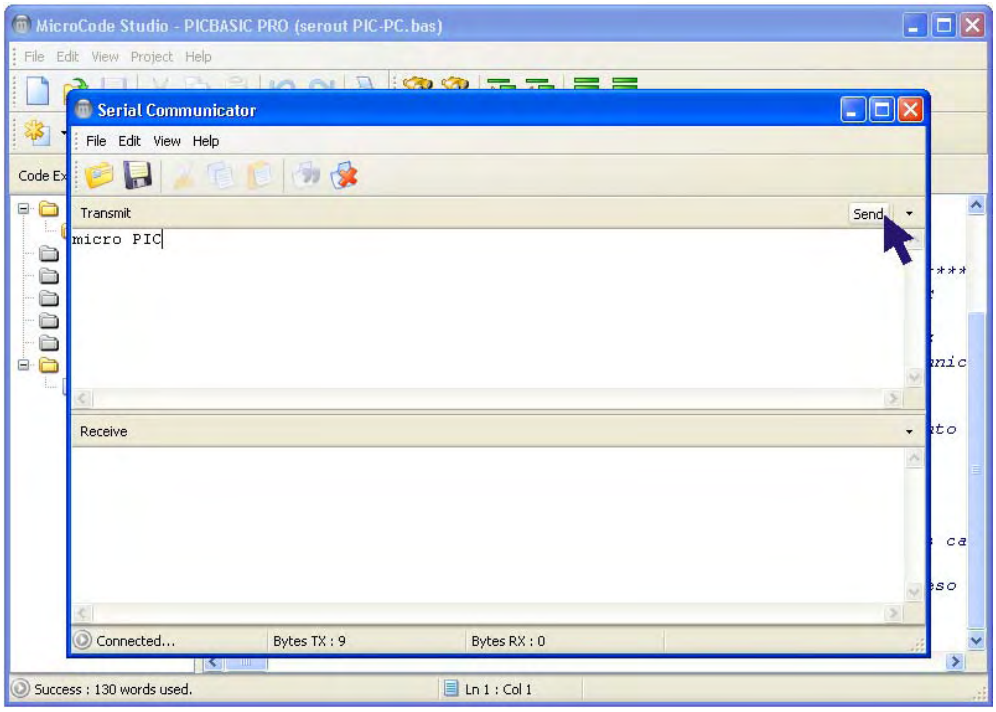


Figura 5.9.5.3. *Pantalla de la ventana de comunicación serial listo para enviar un texto.*

5.9.6. COMUNICACIÓN SERIAL CON EL CI. MAX 232.

El CI. MAX232 es la solución para transmitir a mayor distancia, ya que incrementa los niveles de voltaje de 5 V. a ±10V. gracias a un juego de capacitores que le ayuda a doblar los voltajes, por lo que para su alimentación sólo requiere una fuente de 5V. que puede ser la misma que utiliza el PIC. El MAX232 dispone de 2 juegos de transmisores y receptores, de los cuales sólo ocuparemos un par de ellos.

El MAX232 en este caso nos ayudará a convertir los voltajes TTL del PIC en voltajes de la norma RS232, quiere decir que si enviamos un estado lógico alto (5V.), a la salida del Tout del CI. MAX232 tendrémos –10V. y si enviamos un 0 lógico desde el PIC (0V.), el MAX232 enviará +10V., por lo tanto debemos invertir el dato de la salida del PIC y esto lo conseguimos utilizando T2400 de la siguiente manera:

```
SEROUT portb.1,T2400,["D"]      ;quiere decir enviar el dato serial D por el pin B.1 a
                                ;2400bits/s 8N1 en dato verdadero, ver figura 5.9.3.1
```

El presente proyecto trabaja de la siguiente manera: una vez listo y conectado todo, cuando el PIC arranca debe encender el led y luego apagarlo, esto para asegurarnos que todo está funcionando bien, ahora desde el computador enviamos una letra cualquiera que no sea la C, observarán que el led parpadea cada que se le envía una letra, ahora si enviamos la C mayúscula el led se quedará encendido permanentemente e inmediatamente el PIC empezará a enviar un contador separado por el signo menos (-) empezando desde el 0 hasta el 255, como podemos observar esto es un ejemplo de la transmisión half-duplex.

MATERIALES.
-1 CI. MAX232
-4 capacitores de 10 uF electrolíticos o preferible de tantalio
-de 2 a 30 mts de cable de 2 pares de hilos
-1 conector DB9 hembra con su cajetín
-1 led
-1 resistencia de 330Ω.

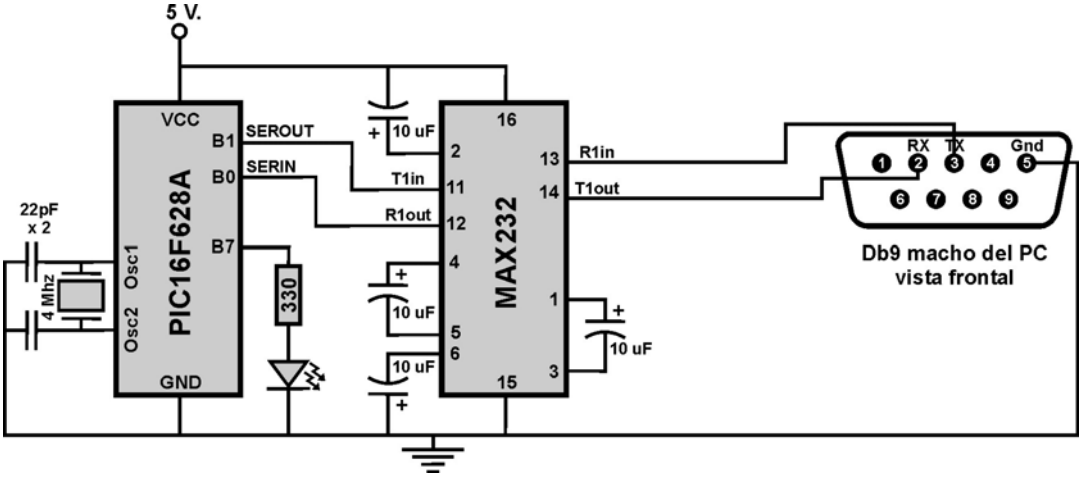


Figura 5.9.6.1. *Diagrama de conexión del PIC y el CI. MAX232 para enviar y recibir datos entre un PC y el PIC.*

```
INCLUDE "modedefs.bas"                ;incluyen los modos de comunicación
@ device XT_OSC                       ;cambia a oscilador XT en el IC-Prog
led VAR portb.7                       ;nombre led al puerto b.7
dat VAR BYTE                          ;variable de almacenamiento de 255
num VAR BYTE                          ;variable para almacenar el contador
num=0                                 ;contenido inicial para la variable num
GOSUB ledr                            ;ir a leds para saber si ya arrancó el PIC
Inicio:
SERIN portb.0,T2400,dat               ;esperar el dato y guardarlo en dat
  IF dat = "C" THEN contar            ;si dat es una C empiece a contar
  GOSUB ledr
GOTO inicio                           ;ir a esperar el siguiente caracter
contar:
   HIGH led                           ;led sólo encendido indica enviando datos
SEROUT portb.1,T2400,[#num,"-"]       ;enviar el contenido decimal de la variable num
                                      ;seguido de un signo menos
   num=num+1                          ;incrementar la variable 1 x 1
   PAUSE 1000                         ;esperar 1 s
GOTO contar                           ;ir a subrutina contar
ledr:                                 ;subrutina ledr
   HIGH led
   PAUSE 200
   LOW led
RETURN
END
```

Figura 5.9.6.2. TX-RX-MAX232.pbp*Programa para intercambiar datos entre una PC y un PIC utilizando el CI. MAX232.*

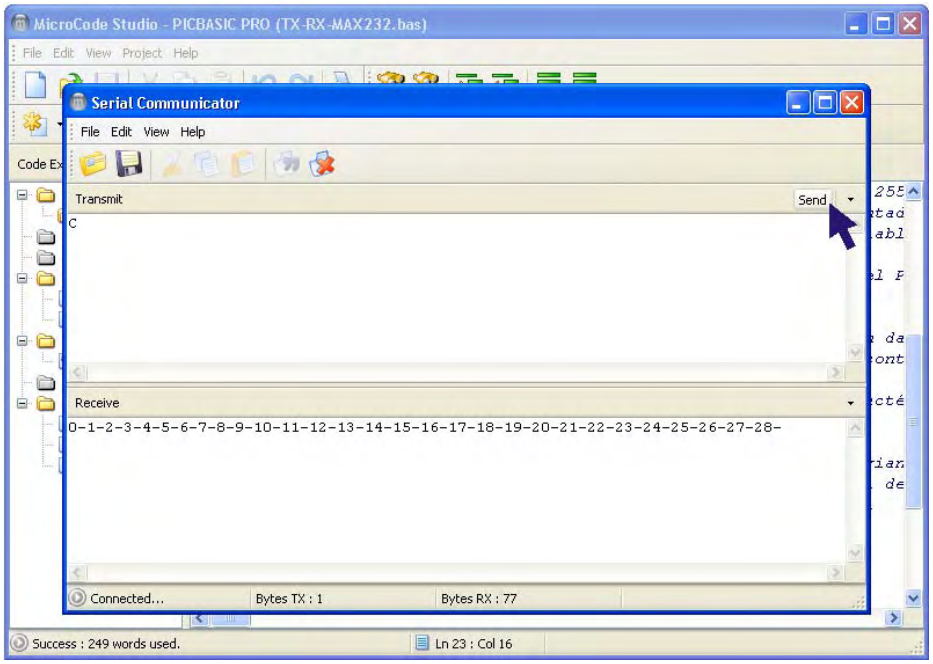


Figura 5.9.6.3. *Pantalla de comunicación serial enviando y recibiendo datos desde el PIC.*

5.9.7. COMUNICACIÓN SERIAL PIC A PIC.

Este proyecto consiste en hacer una transmisión simplex entre un PIC transmisor y un PIC receptor, el primero dispone de 3 botones, el botón A envía la letra "A" el cual el PIC receptor lo detecta y enciende un led rojo por 1 segundo, desde el transmisor presionamos el botón B y transmite la letra "B", el receptor encenderá una led amarillo, igualmente después de un segundo lo apagará y por último desde el transmisor presionamos la tecla C y el receptor encenderá un led verde. Por tratarse de una práctica y no complicarnos con el MAX232, ya que necesitaríamos 2, sólo realizaremos a una distancia de 2 metros conectando directamente de PIC a PIC utilizando dato invertido (N2400).

MATERIALES.
-2 mts de cable de 2 de hilos
-3 leds un rojo, un verde y un amarillo
-3 resistencias de 4,7 K Ω
-3 resistencia de 330Ω
-3 pulsadores NA
-2 PIC16F628A.

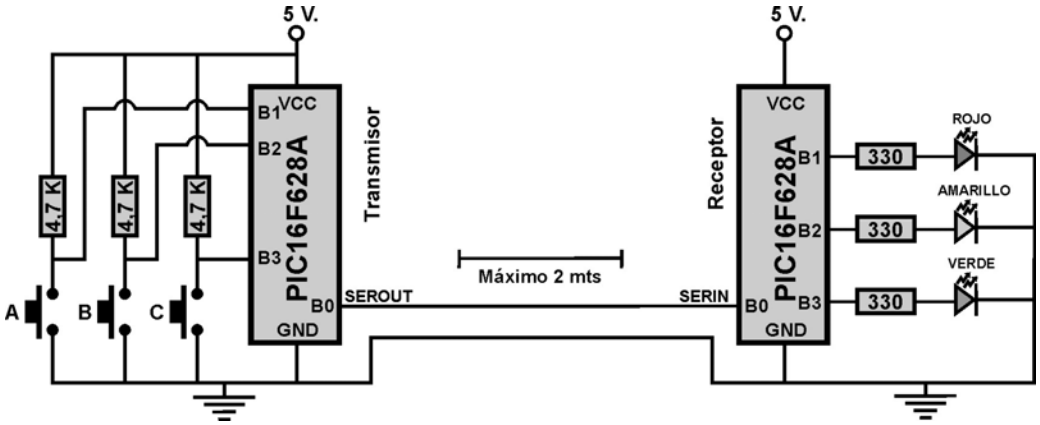


Figura 5.9.7.1. *Diagrama de conexión para hacer una comunicación serial desde un PIC hacia otro PIC.*

Para este proyecto podemos utilizar una o dos fuentes de 5 voltios, pero sería recomendable utilizar 2 fuentes para que la comunicación se vea más real, como es de suponerse se necesitará de 2 programas diferentes uno para cada microcontrolador, por lo que empezaremos con el programa del transmisor.

```
INCLUDE "modedefs.bas"          ;incluyen los modos de comunicación
botonA VAR portb.1              ;nombre botonA al puerto b.1
botonB VAR portb.2
botonC VAR portb.3                                  continúa ...
```

```
transmitir:
IF botonA=0 THEN envio1                ;si el botonA es presionado ir a envio1
IF botonB=0 THEN envio2                ;si el botonB es presionado ir a envio2
IF botonC=0 THEN envio3                ;si el botonC es presionado ir a envio3
GOTO transmitir

envio1:
SEROUT portb.0,N2400,["A"]             ;enviar "A" a 24008N1
   PAUSE 500
   GOTO transmitir
envio2:
SEROUT portb.0,N2400,["B"]             ;enviar "B" a 24008N1
   PAUSE 500
   GOTO transmitir
envio3:
SEROUT portb.0,N2400,["C"]             ;enviar "C" a 24008N1
   PAUSE 500
   GOTO transmitir

END
```

Figura 5.9.7.2. transmisor PIC.pbp *Programa para enviar datos desde un PIC.*

```
INCLUDE "modedefs.bas"                 ;incluyen los modos de comunicación

ledr VAR portb.1                       ;nombre ledr al puerto b.1
leda VAR portb.2
ledv VAR portb.3
datos VAR BYTE                         ;variable para almacenar el dato serial

HIGH ledr                              ;led para saber si ya arrancó el PIC
PAUSE 500
LOW ledr

recibir:
SERIN portb.0, N2400, datos                    ;recibir dato serial y guardar en datos
IF datos="A" THEN HIGH ledr :PAUSE 1000        ;si es A encender ledr y esperar 1 seg.
IF datos="B" THEN HIGH leda :PAUSE 1000
IF datos="C" THEN HIGH ledv :PAUSE 1000

LOW ledr :LOW leda : LOW ledv
GOTO recibir
END
```

Figura 5.9.7.3. Receptor PIC.pbp *Programa para recibir datos desde un PIC.*

5.9.8. COMUNICACIÓN SERIAL RS422/485.

Para entender que es lo que vamos a hacer explicaremos en pocas palabras que es la interfaz serial RS422 y que es la interfaz serial RS485, estas se diseñaron para la conexión física entre computadores y terminales directamente. Estos estándares tienen grandes ventajas con respecto a la norma RS232 como por ejemplo, la distancia de comunicación de hasta 1200 mts, la velocidad de transmisión de hasta 10 Mbits/seg. y el número de elementos a conectarse, para la interfaz RS422 pueden conectarse un transmisor y hasta 10 receptores en un modo de transmisión Full-duplex, mientras que para la interfaz RS485 se pueden conectar simultáneamente hasta 32 transmisores/receptores en un sistema half-duplex, otra ventaja frente al sistema RS232 es que no requiere fuentes duales sino una fuente de alimentación de 5 voltios.

Ambas interfaces utilizan el método de medida diferencial, en la que utilizan dos líneas para la transmisión y dos para recepción, en cada par de conductores la segunda tiene un nivel de voltaje complementario al del primero, y el receptor responde a la diferencia de voltajes entre los dos conductores. Este tipo de líneas de transmisión se llaman balanceadas, y esto permite la eliminación de ruidos electrostáticos y electromagnéticos común en las dos líneas que se utilizan.

El CI. 75176 contiene un transmisor y un receptor y sólo dos líneas diferenciales A y B de entrada/salida de datos, dos líneas adicionales RE y DE determinan la función que debe cumplir el integrado, permitiendo o inhibiendo la recepción o la transmisión de datos. Para este integrado el transmisor es habilitado por un 1 lógico y un 0 lógico habilita el receptor, estas dos líneas RE y DE son unidas a un puerto del PIC, en donde el microcontrolador determinará cuando transmitir y cuando recibir datos, en un sistema half-duplex.

El sistema RS422 establece una comunicación full-duplex para lo cual se requiere 2 líneas adicionales, esto se lo consigue agregando otro CI. 75176 exclusivamente para la transmisión por lo que RE y DE estarán conectados en nivel lógico alto, mientras el otro CI. 75176 se lo configura para recepción conectando los terminales control a un nivel 0 lógico, de esta manera quedan operando los 2 circuitos lineales 75176, con la ventaja de poder transmitir y recibir al mismo tiempo.

Como el microcontrolador PIC16F628A ejecuta línea por línea del programa, no es capaz de leer y recibir un dato a la vez, por lo tanto no se puede realizar una comunicación full-duplex, de esta manera no es aplicable una comunicación con la interfaz RS422, lo ideal es la interfaz RS485 ya que está diseñado para una transmisión half-duplex y de hecho este es el modo de transmisión que se utiliza en la mayoría de dispositivos comerciales basados en microcontroladores, un ejemplo de ello es la comunicación del teclado con la central de alarma, esta lo realiza mediante interfaz RS485 de 4 hilos (A, B, masa y el cuarto hilo alimentación de 12V. para el teclado). Una recomendación importante es que el cable a utilizarse debe ser del tipo par trenzado (Twisted-Pair), que consiste en dos conductores aislados retorcidos entre sí con lo cual se consigue una mayor inmunidad al ruido electromagnético, si el cable tiene adicionalmente una hoja conductora (blindaje) rodeándolo, se obtiene más inmunidad.

5.9.9. COMUNICACIÓN SERIAL PIC A PIC CON LA INTERFAZ RS485.

El presente proyecto consiste hacer una transmisión recepción entre dos PIC´S con una interfaz RS485 y en modo de transmisión full-duplex comúnmente visto en centrales de alarmas.

Su funcionamiento consta de la siguiente manera: al arrancar los micros ambos hacen encender los leds rojos por un instante, esto nos indica que empezaron a funcionar, el transmisor consta de 2 pulsadores y un led, y el receptor de 2 leds, un rojo un verde y un pulsador, cuando presionamos el pulsador A del transmisor enciende el led rojo y presionando el pulsador B enciende el led verde, bien hasta aquí hemos realizado una comunicación de un sólo sentido,

ahora lo intercambiamos para hacer que el PIC receptor se convierta en transmisor y lo mismo con el otro PIC de transmisor a receptor, para esto en el PIC transmisor presionamos las 2 teclas al mismo tiempo por un segundo, luego de esto notarán que cada tecla ya no tiene efecto como lo hacía antes, es decir encendía cada uno de los 2 leds del receptor, ahora vamos al receptor el cual ahora es transmisor y presionamos el único pulsador que dispone, notarán que el led del que antes era transmisor parpadea 2 veces, para volver al modo que iniciaron sólo debemos apagar y volver a encender ambos micros.

MATERIALES.
-cable de 2 pares de hilos, preferible del tipo par trenzado
-3 leds dos rojos y un verde
-3 resistencias de 4,7 K Ω
-3 resistencia de 330Ω
-3 pulsadores NA
-2 PIC16F628A
-2 circuitos lineales 75176.

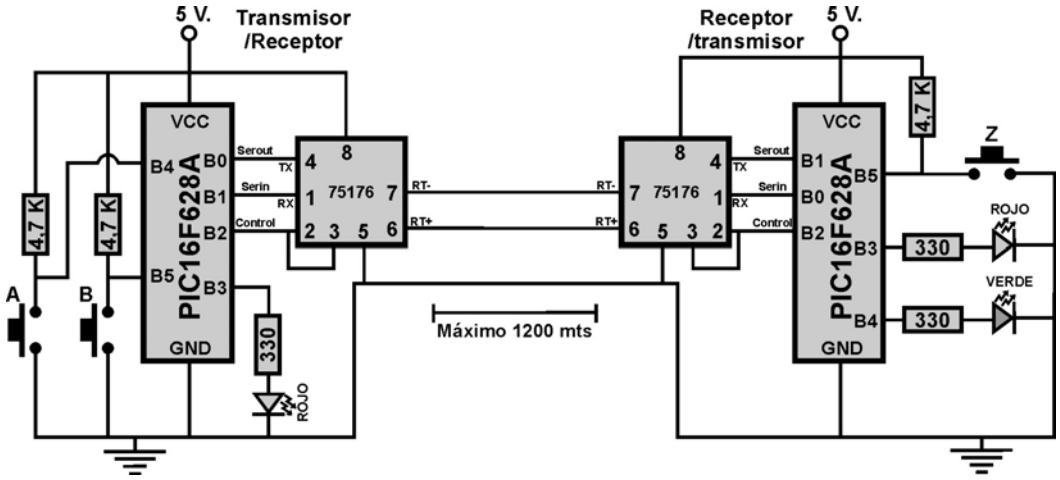


Figura 5.9.9.1. *Diagrama de conexión para hacer una comunicación serial desde un PIC hacia otro PIC con interfaz serial RS485.*

```
INCLUDE "modedefs.bas"                          ;incluyen los modos de comunicación
control VAR portb.2                             ;pin para control TX RX
led VAR portb.3                                 ;nombre para pin b.3
recib VAR BYTE                                  ;variable para almacenar dato

GOSUB rojo                                      ;led para saber si ya arrancó el PIC
envio:                                          ;subrutina envío
HIGH control                                    ;control modo transmisor TX
IF (portb.4=0) AND (portb.5=0)THEN receptor     ;si los 2 botones están presionados
IF portb.4=0 THEN ledrojo                                          continúa …
```

```
IF portb.5=0 THEN ledverde
GOTO envio

ledrojo:                                ;subrutina ledrojo
   GOSUB rojo                           ; encender el led
   SEROUT portb.0,T2400,["A"]           ;enviar la A por puerto b.0
   PAUSE 500
   GOTO envio                           ;ir nuevamente a envío
ledverde:
   GOSUB rojo
   SEROUT portb.0,T2400,["B"]           ; enviar la B por puerto b.0
   PAUSE 500
   GOTO envio                           ;ir nuevamente a envío

receptor:                               ;subrutina receptor
   GOSUB rojo                           ;encender el led
   SEROUT portb.0,T2400,["C"]           ;enviar la C
   PAUSE 500

recibiendo:                             ;subrutina recibiendo
   LOW control                          ;cambiar a modo receptor RX
   SERIN portb.1,T2400,recib            ;esperar dato y guardarlo en recib
   IF recib="Z" THEN perfecto           ;si recib es una Z ir a perfecto
   GOTO recibiendo                      ;caso contrario seguir en recibiendo

rojo:
   HIGH led                             ;encender led rojo
   PAUSE 200                            ;esperar 200 mls
   LOW led                              ;apagar led rojo
   PAUSE 200
   RETURN                               ;retornar al gosub que lo envió
perfecto:                               ;encender 2 veces el led rojo
   GOSUB rojo
   GOSUB rojo
   GOTO recibiendo                      ; ir a recibiendo

END
```

Figura 5.9.9.2. transmisor-RS485.pbp *Programa para enviar datos PIC a PIC en interfaz RS485*

```
INCLUDE "modedefs.bas"                  ;incluye los modos de transmisión
control VAR portb.2                     ;pin para el control
ledr VAR portb.3
ledv VAR portb.4
date VAR BYTE                           ;variable para almacenar dato serial

GOSUB rojo                              ;led para saber si ya arrancó el PIC
```

continúa

```
recibir:                                ;subrutina recibir
   LOW control                          ;control modo receptor
SERIN portb.0,T2400,date                ;esperar por dato serial y guardar
IF date="A" THEN ok1                    ;si dato es una A ir a ok1
IF date="B" THEN ok2
IF date="C" THEN transmisor
   HIGH ledv : HIGH ledr                ;se enciende los 2 leds cuando el
   PAUSE 2000                           ;dato recibido no es A,B, ni C
   LOW ledv :LOW ledr
   PAUSE 500
GOTO recibir                            ; ir a recibir

ok1:                                    ;subrutina ok1
   GOSUB rojo                           ;ir y retornar de rojo
   GOTO recibir
ok2:                                    ;subrutina ok2
   HIGH ledv                            ;encender el led verde
   PAUSE 200
   LOW ledv
   GOTO recibir

transmisor:                             ;subrutina modo transmisor
   HIGH control                         ;control cambia a transmisor
   IF portb.5=0 THEN                    ;si presionamos el botón del portb.5 entonces
     SEROUT portb.1,T2400,["Z"]         ;enviar la Z por puerto b.1
      PAUSE 500                         ;esperar 0,5 seg.
   ENDIF                                ;fin de la condición
GOTO transmisor                          ;volver a transmisor

rojo:                                   ;subrutina rojo
   HIGH ledr                            ;encender el led rojo
   PAUSE 200
   LOW ledr
   PAUSE 200
   RETURN                               ; retornar al gosub que lo envió
END
```

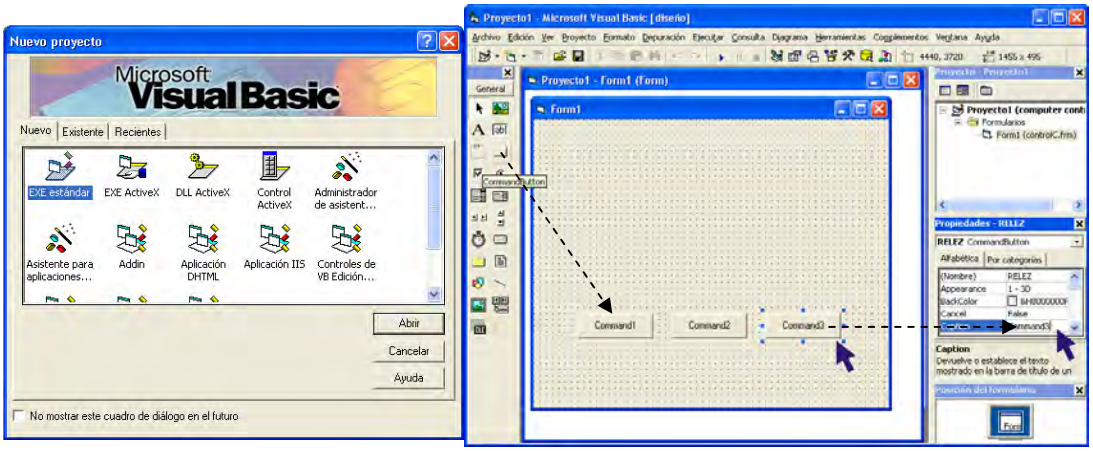
Figura 5.9.9.3. receptor-RS485.pbp *Programa para recibir datos PIC a PIC en interfaz RS485.*

NOTA: Si la comunicación es errónea o está desconectado el transmisor, el receptor encenderá los 2 leds a la vez, el rojo y el verde durante 2 segundos y luego lo apagará por 0,5 seg.

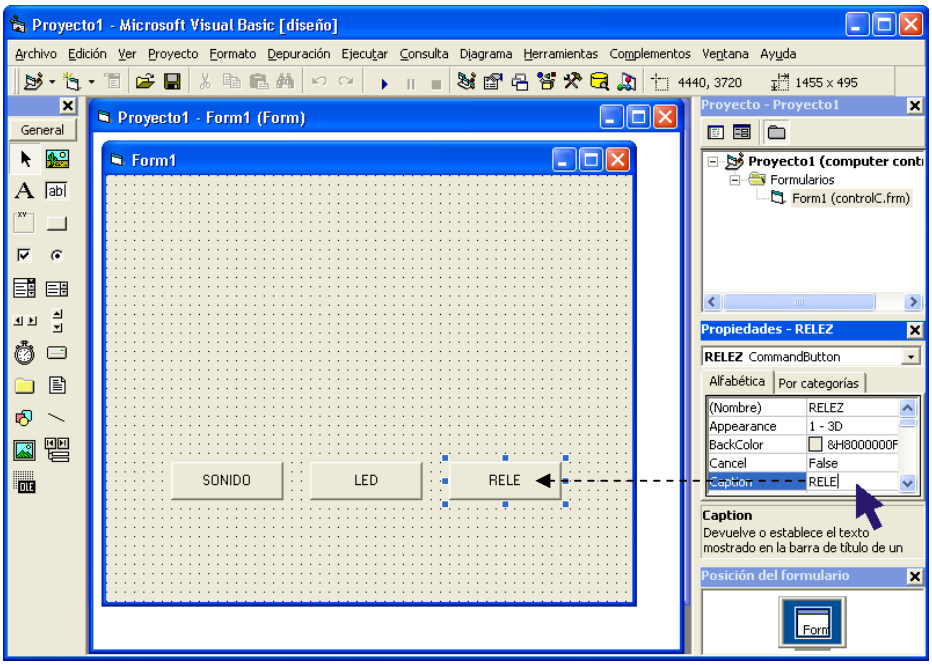
5.9.10. COMUNICACIÓN SERIAL DE VISUAL BASIC Y PIC.

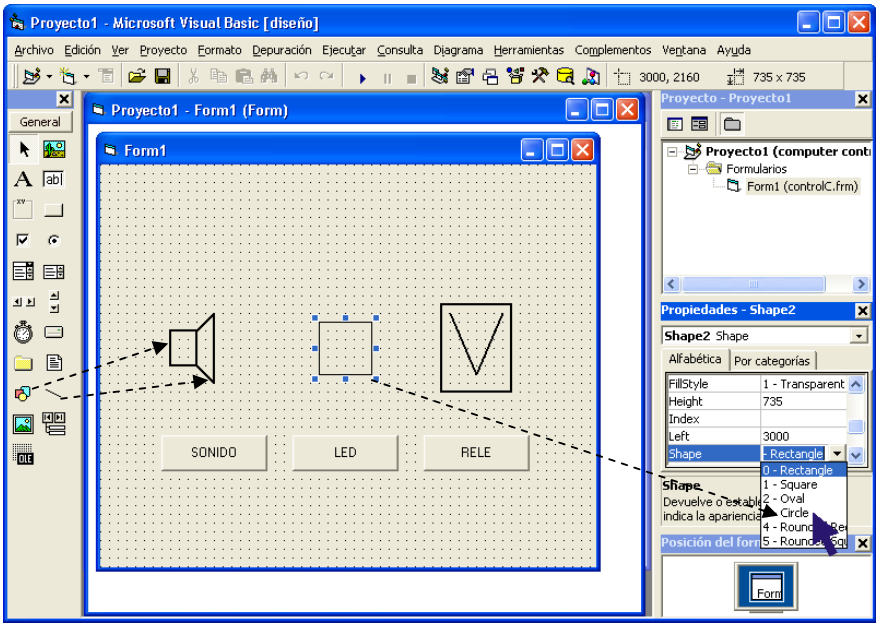
Esta es la parte de la electrónica que más entusiasmo genera, poder controlar a través de un computador todos los periféricos de un edificio (ascensor, luces, alarmas, cerraduras, etc), controlar los movimientos de un brazo robótico, controlar la producción de una fábrica (motores,

bombas, calefactores, etc), todo desde la pantalla de un computador. ¿Les parece interesante?, pues como para introducirnos en el mundo del control computarizado haremos un control de un relé, una chicharra y un led, los cuales nos responderán si están activados o no, para hacer el tablero de control se necesita saber programar en VISUAL BASIC, de todas maneras daremos unas indicaciones para poder crear un tablero básico. Empezaremos por diseñar los botones en un form de VB, para esto ejecutamos el programa VB, en la pantalla principal escogemos exe estándar y damos clic en Abrir.

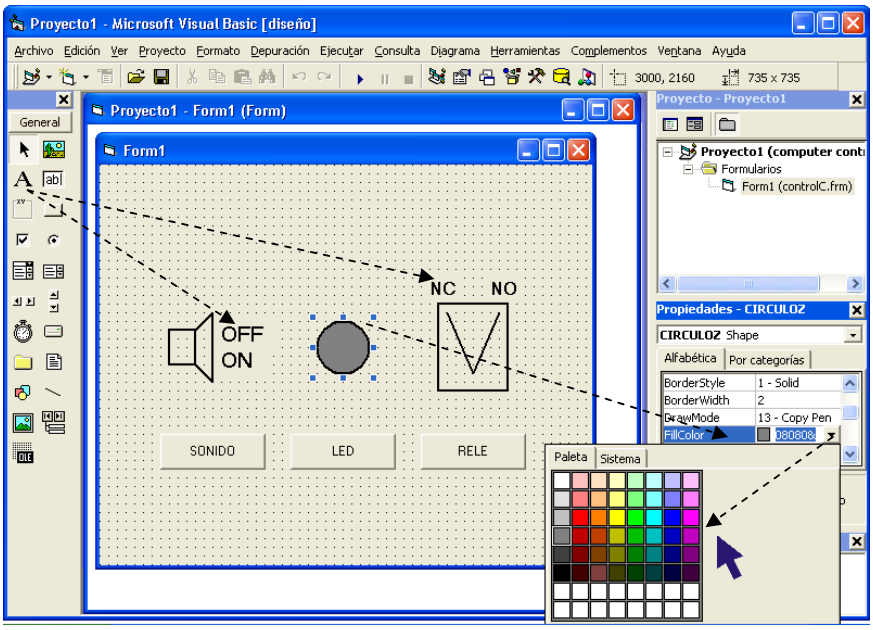


Como pueden ver en la pantalla de la derecha hemos creado 3 botones con la herramienta commandButton, si desean pueden cambiar el texto de command3 por ejemplo por RELE, para esto primero debemos seleccionar el objeto en este caso command3 y en propiedades del objeto que se encuentra al lado derecho específicamente en Caption Command3, lo borramos y escribimos RELE, en el otro botón escribimos LED y en el último SONIDO, quedará como ilustra el siguiente gráfico:



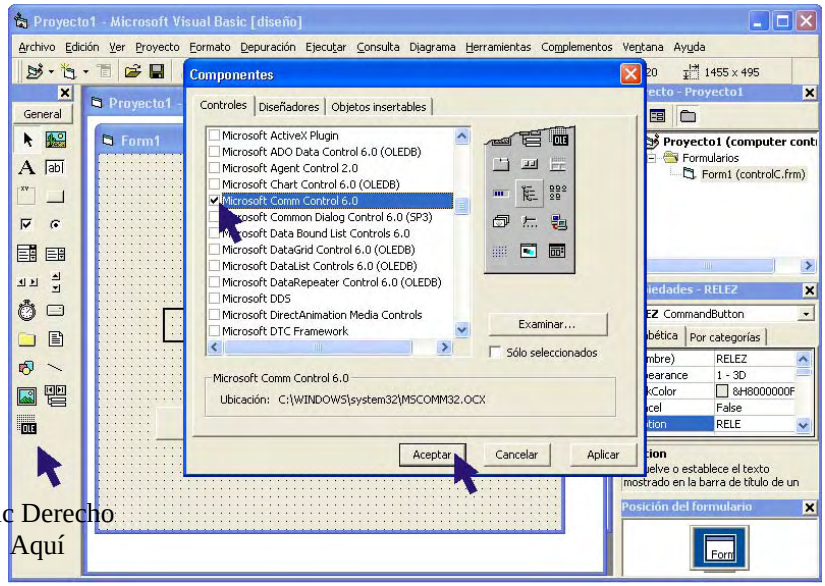


Ahora como ven arriba hemos dibujado un parlante y un relé, explicamos cómo se hizo cada uno, primero escogemos la herramienta del lado izquierdo **Shape**, que sirve para dibujar rectángulos círculos óvalos, en este caso dibujamos el parlante, hacemos un rectángulo y luego cogemos la herramienta **Line** y dibujamos las líneas de la bocina, si queremos que las líneas sean más gruesas, en el lado derecho en propiedades del objeto BorderWidth 1, lo colocamos el 2 y esto hará más gruesas las líneas que seleccionemos. De esta misma forma creamos el relé y ponemos 2 líneas para indicar el cambio de estado del relé. Para crear el LED utilizamos Shape mismo y primero hacemos un cuadrado, luego vamos al lado derecho en propiedades en Shape debe decir 0 rectangle, lo ponemos 3 circle y se convertirá en un círculo, cambiamos el grosor a 2 y listo.

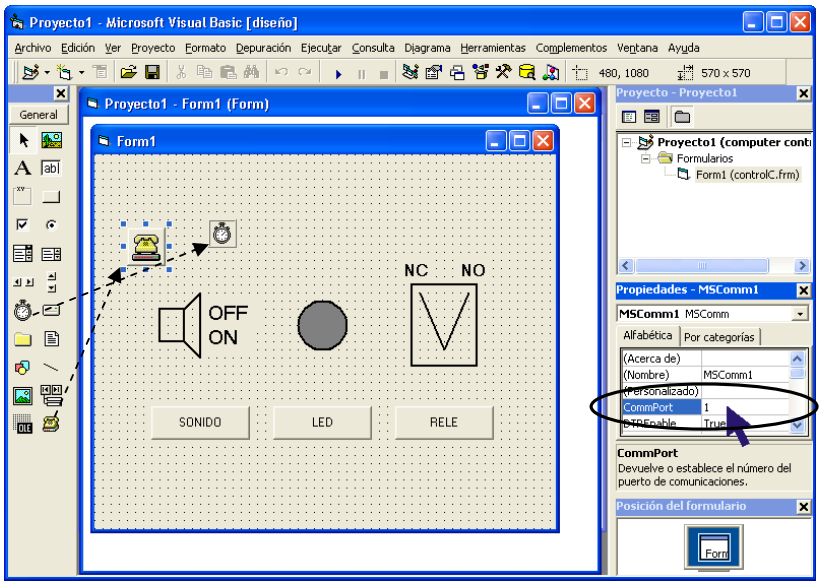


Para crear textos presionamos en el lado izquierdo en label, lo colocamos en el lugar que deseamos y luego en Caption ponemos ON, OFF, NC, NO. Para cambiar el estilo de letra y el tamaño, primero seleccionamos el texto a modificar y en el lado derecho en propiedades Font, elegimos los cambios que deseemos y listo.

Para dar color al LED, 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 el led está apagado.



Para habilitar la comunicación serial, damos un clic con el botón derecho sobre el cuadro General y escogemos la opción componentes, luego saldrá una pantalla con una lista de componentes y buscamos **Microsoft Comm Control 6.0**, seleccionamos y damos 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.



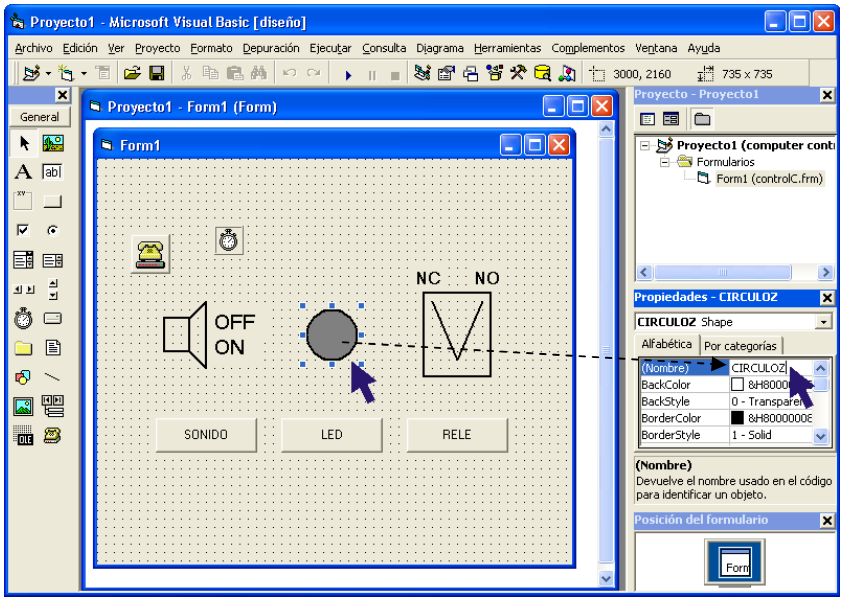
Bien ahora la palabra ON y la línea del relé que señala al NO, no deben aparecer, sino hasta cuando el microcontrolador se los indique, por tal razón debemos ocultarlos, y esto lo realizamos seleccionando a cada uno y en propiedades donde dice Visible True, lo cambiamos por False, bien en ese instante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

Es importante darles nombres a cada objeto sólo a los que necesariamente vamos a modificar, estos se lo realiza de la siguiente manera, primero seleccionamos, digamos que el botón SONIDO, al lado derecho el primer item de propiedades dice (Nombre) Command1 aquí lo ponemos el mismo nombre del objeto con una **Z** al final así SONIDOZ, esto lo hacemos con el objeto de no confundir un nombre Caption SONIDO con el nombre del objeto SONIDOZ, lo mismo hacemos con los siguientes objetos: la palabra ON y OFF, las tres teclas SONIDO, LED y RELE, la línea del relé que señala a la palabra NO y la que señala a NC también, todos ellos su nuevo nombre a cambio de: Command1, Command2, Command3, ShapeX, LabelX, LineX, serán:

Propiedad	N. antiguo	Propiedad	N. nuevo
(Nombre)	CommandX	(Nombre)	SONIDOZ
(Nombre)	CommandX	(Nombre)	LEDZ
(Nombre)	CommandX	(Nombre)	RELEZ
(Nombre)	ShapeX	(Nombre)	CIRCULOZ
(Nombre)	LabelX	(Nombre)	OFFZ
(Nombre)	LabelX	(Nombre)	ONZ
(Nombre)	LineX	(Nombre)	LINEANOZ
(Nombre)	LineX	(Nombre)	LINEANCZ

Figura 5.9.10.1. *Tabla que muestra los cambios de nombre a realizar.*

A continuación una imagen que muestra cómo se cambia el nombre de un objeto, observen que se seleccionó el Círculo y en el lado derecho decía (Nombre) Shape2, se lo cambió por (Nombre) CIRCULOZ, de esta manera cambiamos a los 8 objetos ya indicados anteriormente con los nombres que aparece en la tabla de la figura 5.9.10.1.



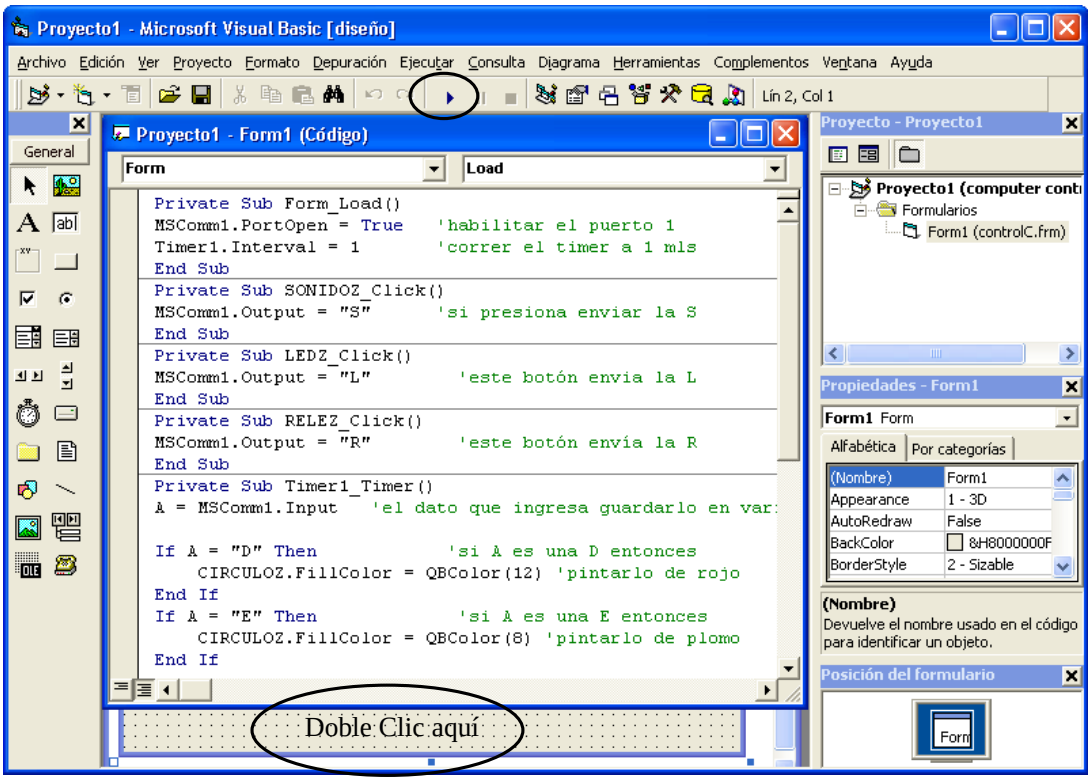
Es hora de programar las funciones de los botones, para esto primero damos doble clic en cualquier parte de la form, esto hará que se abra otra pantalla (Código), en la que sale un texto así:

```
Private Sub Form_Load ( )

End Sub
```

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

```
Private Sub Form_Load()
    MSComm1.PortOpen = True              'habilitar el puerto comm1
    Timer1.Interval = 1                  'correr el timer con intervalos de 1 mls
End Sub
```

Esto quiere decir que cuando se ejecute este programa corra su contenido, es decir abra el puerto com y empiece a correr el Timer con intervalos de 1 mls.



Si se quiere hacer comunicación sólo desde el PC al PIC y no desde el PIC al PC, borramos las líneas desde donde dice: Private Sub Timer1_Timer( ) hasta el final del programa, también debemos borrar la línea que dice Timer1.Interval=1. y en la form borrar el Timer, y dejarlo el teléfono, ya que sí lo necesitamos para enviar datos desde el PC al PIC.

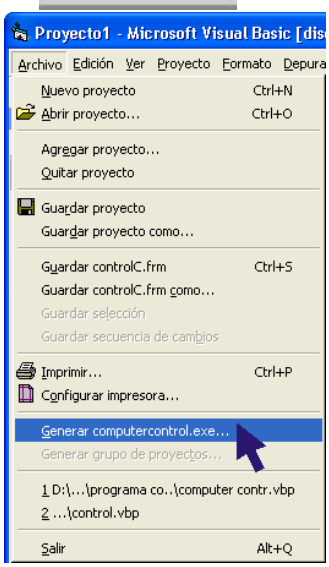
El programa completo quedaría así:

```
Private Sub Form_Load()
        MSComm1.PortOpen = True          'habilitar el puerto 1
        Timer1.Interval = 1              'correr el timer a 1 mls
End Sub
Private Sub SONIDOZ_Click()
        MSComm1.Output = "S"             'si presiona SONIDOZ SONIDO enviar la S
End Sub
Private Sub LEDZ_Click()
        MSComm1.Output = "L"             'si presiona el botón LEDZ LED enviar la L
End Sub
Private Sub RELEZ_Click()
        MSComm1.Output = "R"             'si presiona RELEZ  RELE enviar la R
End Sub
Private Sub Timer1_Timer()

        A = MSComm1.Input                'el dato que ingresa guardarlo en variable A

If A = "D" Then                                  'si A es una D entonces
   CIRCULOZ.FillColor = QBColor (12)             'pintar el círculo de rojo
End If
If A = "E" Then                                  'si A es una E entonces
   CIRCULOZ.FillColor = QBColor (8)              'pintar el círculo de plomo
End If
If A = "G" Then
   LINEANCZ.Visible = False                      'ocultar línea NC relé
   LINEANOZ.Visible = True                       'mostrar la línea NO relé
End If
If A = "F" Then
   LINEANCZ.Visible = True                       'mostrar la línea NC
   LINEANOZ.Visible = False                      'ocultar la línea NO
End If
If A = "H" Then
   ONZ.Visible = True                            'mostrar la palabra ON
   OFFZ.Visible = False                          'ocultar la palabra OFF
End If
If A = "I" Then
   ONZ.Visible = False                           'ocultar la palabra ON
   OFFZ.Visible = True                           'mostrar la palabra OFF
End If
End Sub
```

Figura 5.9.10.1. *Programa para recibir y enviar datos desde el VISUAL BASIC 6.0 a un PIC.*

NOTA: todo lo que está con trama gris, sería el programa necesario para únicamente transmitir desde el PC al PIC, lo demás podemos borrarlo, junto con el icono de la form (el Timer).

Una vez escrito todo el programa hágalo 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 Generar xxx.exe:



MATERIALES
-1 led rojo
-2 resistencias de 4,7 K Ω
-1 resistencia de 330 Ω
-1 resistencia de 220 Ω
- 1 resistencia de 22 K Ω
-2 transistores 2N3904
-1 relé de 12 voltios
-1 chicharra activa
-1 diodo rectificador 1N4007

NOTA: Para mayor comodidad si usted no dispone del programa VISUAL BASIC, facilitamos un archivo ejecutable que usted puede copiar en su computador, junto a otros 2 archivos de sistema necesarios para que corra este programa, una vez que ya disponga de este programa de control, usted podrá realizar esta práctica.

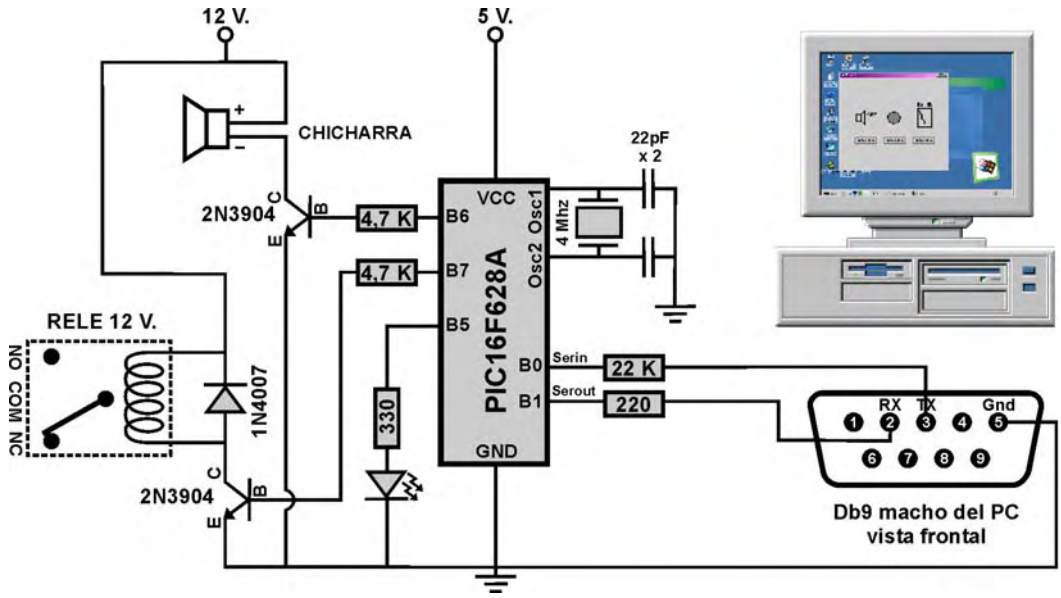


Figura 5.9.10.2. *Diagrama de conexión para el control computarizado con Visual Basic.*

NOTA: Si desea extender la distancia de los cables y decide utilizar el CI. MAX232, debe cambiar todos los N9600 por T9600.

```
 INCLUDE "modedefs.bas"                ;incluyen los modos de comunicación
@ device XT_OSC                        ;cambia a oscilador XT en el IC-Prog
serial VAR BYTE                        ;variable de almacenamiento de 255
sec1 VAR BIT                           ; variable sec1 de 1 bit 0 o 1
sec2 VAR BIT                           ; variable sec2 de 1 bit 0 o 1
sec3 VAR BIT                           ; variable sec3 de 1 bit 0 o 1
sec1=0                                 ;valores iniciales para las variables
sec2=0
sec3=0

led         VAR portb.5                ;nombre para los pines
chicharra VAR portb.6
rele         VAR portb.7

HIGH led:PAUSE 500:LOW led             ;led para saber si ya arrancó el PIC

Inicio:
SERIN portb.0,N9600,serial             ;esperar por dato serial y guardarlo
  IF serial="S" THEN sonidos           ;si el dato es una S ir a sonido
  IF serial="L" THEN leds              ;si el dato es una L ir a leds
  IF serial="R" THEN reles             ;si el dato es una R ir a reles

sonidos:
   IF sec1=0 THEN                      ;bandera para la chicharra 1 On y 0 es OFF
   HIGH chicharra
   SEROUT portb.1,N9600,["H"]          ;enviar H diciendo que está prendido
   sec1=1
   GOTO inicio
  ENDIF
   IF sec1=1 THEN
   LOW chicharra
   SEROUT portb.1,N9600,["I"]          ;enviar I diciendo que está OFF
   sec1=0
  ENDIF
GOTO inicio

leds:
   IF sec2=0 THEN
   HIGH led
   SEROUT portb.1,N9600,["D"]          ;envía D diciendo que el led es ON
   sec2=1
   GOTO inicio
  ENDIF
   IF sec2=1 THEN
   LOW led
   SEROUT portb.1,N9600,["E"]          ;envía E diciendo que el led es OFF
   sec2=0
  ENDIF
GOTO inicio                                              continúa …
```

```
reles:
   IF sec3=0 THEN
   HIGH rele
   SEROUT portb.1,N9600,["G"]          ;envía G diciendo que el relé es ON
   sec3=1
   GOTO inicio
  ENDIF
   IF sec3=1 THEN
   LOW rele
   SEROUT portb.1,N9600,["F"]          ;envía F diciendo que el relé es OFF
   sec3=0
  ENDIF
 GOTO inicio
 END
```

Figura 5.9.10.3. Control-Visual-Basic.pbp *Programa para enviar y recibir datos desde el PIC.*

Como para probar si los datos que salen del PIC son correctos, puede conectar a la ventana de comunicación serial de microcode, a **9600 baudrate,** y en Transmit, envíe las letras mayúsculas una por una la S, S, L, L, R, R, y de respuestas debe recibir lo siguiente: HIDEGF.

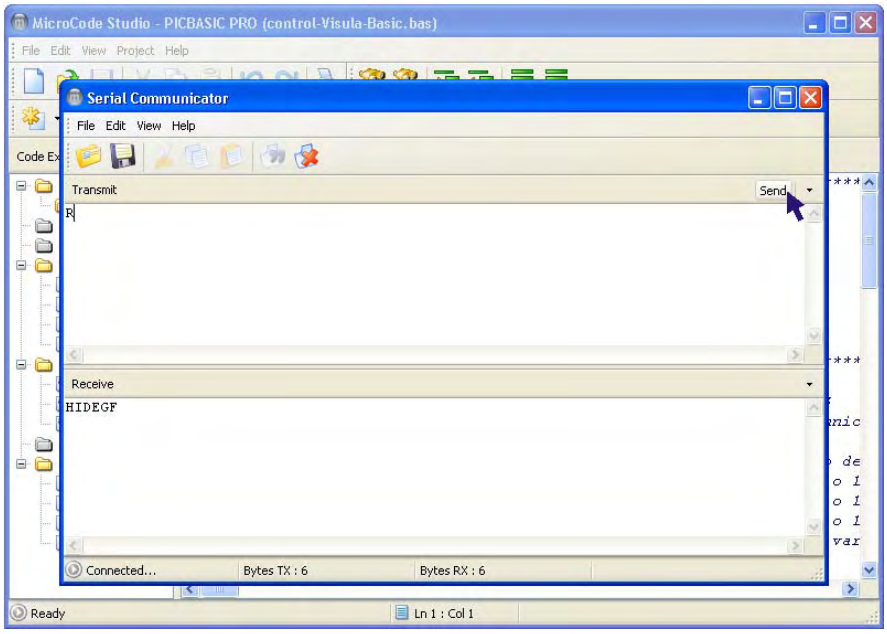


Figura 5.9.10.4. *Pantalla de comunicación serial con los datos de respuestas si está activado la chicharra(H) o no(I), encendido el led (D) o no(E), conectado el relé (G) o no (F).*

La comunicación con el programa hecho en Visual Basic es similar, con la diferencia que esas mismas letras activa la palabra ON, cambia de color el círculo, y activa la línea NO.

5.9.11. COMUNICACIÓN SERIAL SICRÓNICA I²C.

Muchos de los dispositivos electrónicos que se encuentran comúnmente en una tarjeta electrónica, incluyen circuitos integrados con el bus I²C desarrollado por PHILIPS, como por ejemplo las memorias 24CXX, los procesadores de señal, codificadores de video, sensores de temperatura, RTC (reloj en tiempo real), sensores ultrasónicos, etc.

El bus I²C (Inter Integrated Circuit) o interconexión de circuitos integrados necesita sólo 2 líneas para transmitir y recibir datos, estos son: para datos (SDA) y para la señal de reloj (SCL), esta forma de comunicación utiliza una sincronía con un tren de pulsos que viaja en la línea SCL, de tal manera que en los flancos negativos se revisan los datos RX o TX. Ver figura 5.9.12.2., su velocidad de transmisión pueden ser de 100Kbits/seg. en el modo standard, 400Kbits/seg. en el modo rápido y 3,4Mbits/seg. en alta velocidad. Cada dispositivo conectado al bus tiene un código de dirección seleccionable mediante software, por lo que existe una relación permanente Master/Slave. El Master es el dispositivo que inicia la transferencia en el bus y genera la señal de reloj (SCL), y el Slave es el dispositivo direccionado, sin embargo cada dispositivo reconocido por su código (dirección), puede operar como transmisor o receptor de datos, ya que la línea (SDA) es bidireccional.

5.9.12. COMUNICACIÓN I²C CON UNA MEMORIA SERIAL 24LC04B.

Esta es una práctica muy básica para aprender sobre la interfaz I²C, consiste en guardar datos en las cuatro primeras direcciones de la memoria serial, estas son utilizadas para el almacenamiento de datos que pueden ser necesitados más adelante. Para el caso de la memoria 24LC04B tiene un espacio de memoria de 4Kbytes, luego de almacenarlos los volveremos a leer y mostrar en la pantalla de un LCD.

Referencia	Capacidad	Ciclos de E/W	Bloques internos	Dirección			Cantidad de disposit. en el bus	Voltaje de operación
				A0	A1	A2		
24LC01B	1K bits	1,000.000	1	1-0	1-0	1-0	8	2,5-5,5V.
24LC02B	2K bits	1,000.000	1	1-0	1-0	1-0	8	2,5-5,5V.
24LC04B	4K bits	1,000.000	2	X	1-0	1-0	4	2,5-5,5V.
24LC08B	8K bits	1,000.000	4	X	X	1-0	2	2,5-5,5V.
24LC016B	16K bits	1,000.000	8	X	X	X	1	2,5-5,5V.

Figura 5.9.12.1. *Tabla de las capacidades de las memorias EEPROM y su direccionamiento, en nuestro caso sólo podemos poner hasta 4 memorias seriales en una red I²C.*

El principio de funcionamiento es el siguiente: primero se envía el start bit (bit de arranque) cada palabra puesta en el bus SDA debe tener 8 bits, la primera palabra transferida debe contener la dirección del esclavo seleccionado, en este caso se envía el código de la memoria 1010 (este dato lo suministra el fabricante), luego la dirección del dispositivo (A2, A1, A0), y un bit 0 indicando que se desea escribir en la memoria (1=lectura), luego de todo esto la memoria debe enviar un reconocimiento para informarle al microcontrolador que recibió la información, este acuse de recibo se denomina ACK (acknowledge).

Luego el Master lee el ACK, si vale 0 (enviado por el esclavo), el proceso de transferencia continúa. Si vale 1, esto indica que el circuito direccionado no valida la

comunicación, entonces el Maestro genera un bit de stop para liberar el bus I²C, en la cual las líneas SDA y SCL pasan a un estado alto, vamos a suponer que el ACK es 0, entonces el microcontrolador envía los 8 bits correspondientes a la posición de memoria que se desea escribir o leer, nuevamente la memoria envía un reconocimiento, finalmente se envía el dato a ser almacenado y se espera la respuesta de la memoria indicando que el dato llegó correctamente, finalmente se debe enviar el bit de parada.

Como en nuestra práctica vamos a almacenar la palabra HOLA, vamos a suponer que ya guardamos las 3 primeras letras, la H en la dirección 0, la O en la dirección 1 y la L en la dirección 2, nos falta guardar la A en la dirección 3, esto se realizaría enviando pulsos de la siguiente manera:

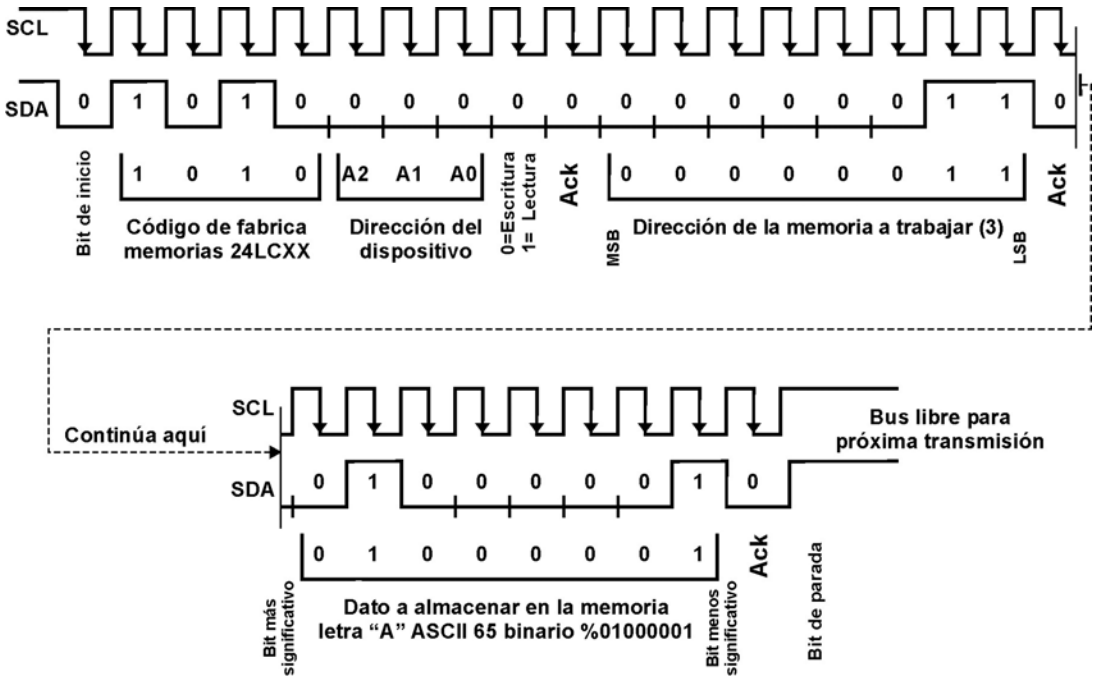


Figura 5.9.12.2. *Esquema de una transmisión completa con la interfaz I²C para guardar el número 65 en la dirección 3 de una memoria serial 24LC04B.*

Noten que la dirección del CI. A2, A1, A0 es 000, lo que quiere decir que estos tres pines van conectados a tierra, según la tabla 5,9,12,1, se pueden instalar 4 dispositivos de memoria en un bus, estos son empezando por A2, A1 y A0 los siguientes: 00x, 01x, 10x y 11x, por ejemplo: 01x esta memoria debe tener conectado a 5 voltios el pin A1 y su control sería 10100100.

LA DECLARACIÓN I2CWRITE Y I2CREAD. Estas declaraciones sirven para escribir y leer datos en un chip EEPROM serial usando una interfaz I²C de 2 hilos, funcionan en modo I²C Master y también puede ser utilizado para comunicarse con otros dispositivos de interfaz I²C como sensores de temperatura, reloj calendarios, conversores A/D, etc.

Los 7 bits de control contienen el código de fábrica del chip y la selección del chip A2, A1, A0, el último bit es una bandera interna que indica si es un comando de lectura o escritura y **no se debe usar**. Por lo tanto el control para nuestro caso en lectura o escritura es %10100000.

Debido a que los pines SDA y SCL de la memoria 24LC04B son de colector abierto, estas deben ir conectadas con resistencias de 4,7 KΩ pull-up, sin embargo existe una línea de comando que hace que no se necesite la resistencia pull-up del SCL, esta se debe agregar al comienzo del programa:

```
DEFINE  I2C_SCLOUT 1            ; no es necesario resistencia pull-up en SCL (reloj)
```

También cabe indicar que existen algunas memorias que necesitan de un período de tiempo para poder ser grabadas, por lo que se adiciona un **PAUSE** 10 después de cada grabación.

Su estructura es de la siguiente manera:

```
I2CWRITE portb.6, portb.7, %10100000, 0, [65]     ;almacenar en la dirección 0 el dato 65
PAUSE 10                                          ;pausa  necesaria para completar la grabación
```

I2CWRITE pin dato, pin reloj, control, posición memoria, variable.

MATERIALES.
-1 LCD 2 x 16
-1 resistencia de 4,7 K Ω
-1 resistencia de 10 Ω
-1 potenciómetro de 10 K Ω
-1 memoria serial 24LC04B de microchip o equivalente
-1 switch selector de 3 pines.

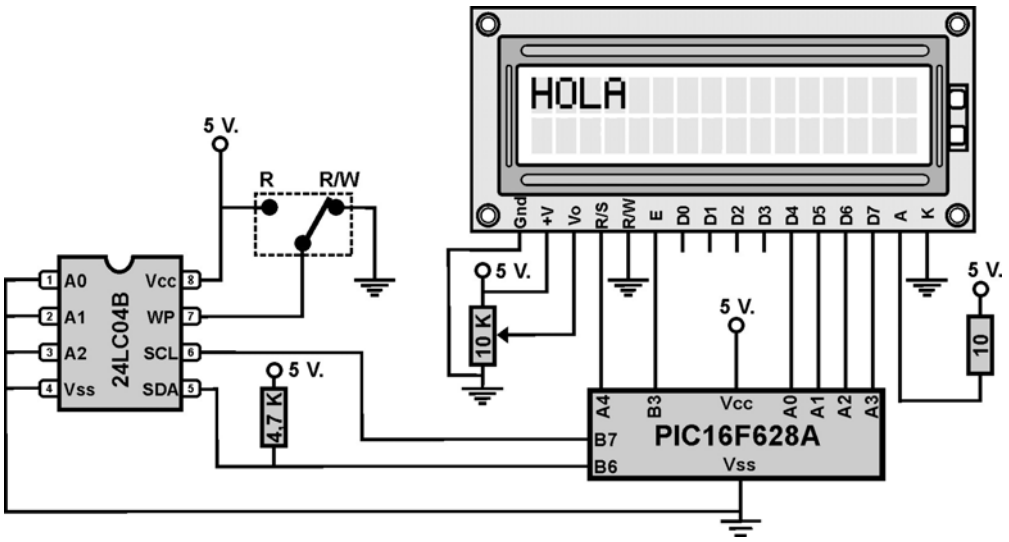


Figura 5.9.12.3. *Esquema de conexionado de una memoria 24LCXX a un PIC, el switch selector permite proteger a la memoria de escrituras accidentales al colocar en R Read.*

En nuestro caso vamos a escribir y leer en la memoria, por lo que el pin WP debe estar colocado en estado bajo, una vez que se haya grabado se podrá colocar este pin en estado alto para proteger la memoria de futuras escrituras, los datos almacenados en esta memoria permanecen aún si se le corta la alimentación al CI. (no volátil), el acceso a estos datos se lo realiza las veces deseadas, recuerde que la memoria serial soporta 1,000.000 de ciclos de borrado y escritura, por lo tanto se debe tener cuidado de no ejecutar un programa que almacene una y otra vez el mismo dato.

```
DEFINE  I2C_SCLOUT  1           ;para que no necesite resistencia pull-up en SCL

contro   CON   %10100000        ;contro contiene valor constante %10100000
PinSCL   VAR   Portb.7                      ;pin señal de reloj I2C
PinSDA   VAR   Portb.6                      ;pin de datos I2C
dato     VAR BYTE                           ;variable para almacenar dato leído

Inicio:
LCDOUT $fe,1," Grabando..."                 ;limpiar y sacar el texto en LCD
PAUSE 1000

I2CWRITE PinSDA,PinSCL,contro,0,["H"]       ;guarda la H en posición 0
PAUSE 10                                    ;pause para la grabación
I2CWRITE PinSDA,PinSCL,contro,1,["0"]       ;guarda la O en posición 1
PAUSE 10                                    ;pause para la grabación
I2CWRITE PinSDA,PinSCL,contro,2,["L"]       ;guarda la L en posición 2
PAUSE 10                                    ;pause para la grabación
I2CWRITE PinSDA,PinSCL,contro,3,["A"]        ;guarda la A en posición 3
PAUSE 10                                    ;pause para la grabación

LCDOUT $fe,1,"Leer memoria"                 ;limpiar y sacar el texto en LCD

PAUSE 1000
LCDOUT $FE,1                                ;limpiar pantalla del LCD

leer:                                       ;programa para leer la memoria serial.

I2CREAD PinSDA,PinSCL,contro,0,[dato]       ;leer la mem. 0 y guardar en dato
  LCDOUT, dato                              ;mostrar dato en el LCD
  PAUSE 1000                                ;esperar 1 seg.
I2CREAD PinSDA,PinSCL,contro,1,[dato]       ;leer la mem. 1 y guardar en dato
  LCDOUT, dato                              ;mostrar dato en el LCD
  PAUSE 1000                                ;esperar 1 seg.
I2CREAD PinSDA,PinSCL,contro,2,[dato]       ;leer la mem. 2 y guardar en dato
  LCDOUT, dato                              ;mostrar dato en el LCD
  PAUSE 1000                                ;esperar 1 seg.
I2CREAD PinSDA,PinSCL,contro,3,[dato]       ;leer la mem. 3 y guardar en dato
  LCDOUT, dato                              ;mostrar dato en el LCD

END
```

Figura 5.9.12.4. memoria 24LCXX.pbp *Programa para escribir y leer datos en un chip EEPROM.*

5.9.13. COMUNICACIÓN I²C CON EL RELOJ CALENDARIO DS1307.

Como habíamos dicho anteriormente el bus I²C, permite la comunicación con algunos dispositivos como las memorias 24CXX, los procesadores de señal, codificadores de video, sensores de temperatura, RTC (reloj en tiempo real), etc. Esta vez haremos un proyecto de lectura y escritura de un RTC (Real Time Clock), este proyecto es muy similar al anterior, pero con la diferencia que el byte de control es %11010000, (propio del fabricante), y su modo de grabación de datos es en sistema hexadecimal.

Una aclaración importante es que por ser un experimento los datos a ser almacenados serán: Lunes 27 de septiembre del 2004 y la hora 13:32:00, pudiendo ser cambiado a gusto del lector. Estos datos serán grabados una sola vez, ya que se utilizará una condición de bandera, esta será almacenada en la memoria EEPROM del mismo PIC y se llamará (actualizado =1), significa que ya corrió una vez el programa, por consiguiente si se lo vuelve a prender el PIC, no guardará nuevamente los datos, esto tiene una finalidad de que los datos fecha y hora sean igualados una sola vez, para que cuando usted vuelva a encender el PIC, muestre la hora actual.

MATERIALES.
-1 LCD 2 x 16
-1 resistencia de 10 Ω
-1 potenciómetro de 10 K Ω
-1 RTC DS1307
-3 resistencias de 4,7 K Ω
-1 resistencia de 470 Ω
-1 transistor 2N3904
-1 led rojo
-1 oscilador cristal de 32768 HZ (comun en los relojes de mano y pared)
-1 pila de 3 V (CR2032), con su porta pilas ver figura 5.9.13.3.

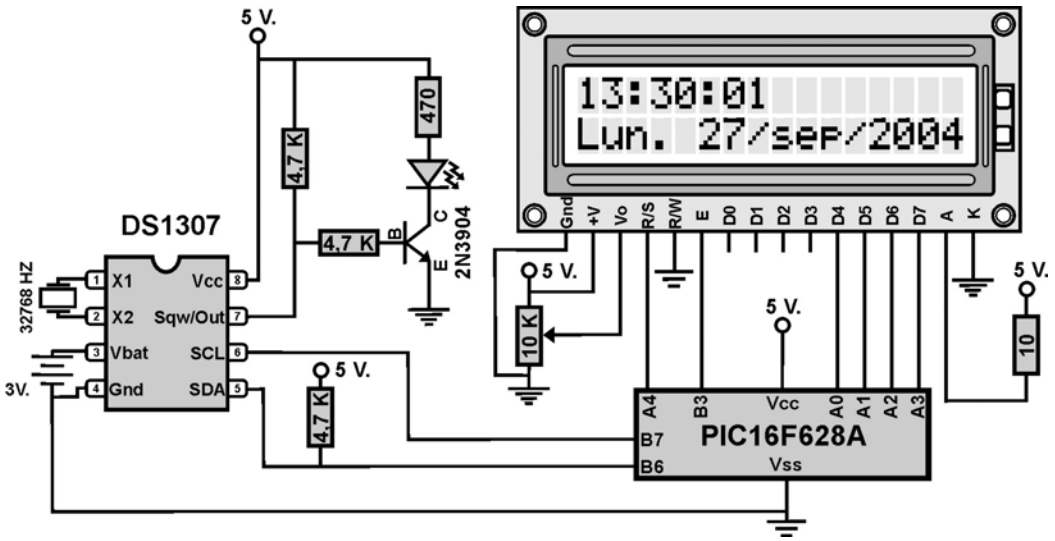


Figura 5.9.13.1. *Esquema de conexionado de un RTC con el PIC para un reloj calendario.*

```
DEFINE  I2C_SCLOUT  1              ;para que no necesite resistencia pull-up en SCL

CPIN  VAR Portb.7                 ;pin señal de reloj I2C
DPIN  VAR Portb.6                 ;pin de datos I2C

segu   VAR BYTE                   ;definir tamaño de variable segundos 1 a 255
minu   VAR BYTE                   ;variable para los minutos
hora   VAR BYTE                   ;variable para las horas
diaS   VAR BYTE                   ;variable día de la semana
diaF   VAR BYTE                   ;variable día fecha del mes
mes    VAR BYTE                   ;variable mes
anio   VAR BYTE                   ;variable año de 2 dígitos

actualizado VAR BIT               ;variable para almacenar un 1 o 0

EEPROM 0,[0]                      ;memoria 0 con el valor inicial 0, sirve para
                                  ;indicar que nunca ha corrido este programa
READ 0,actualizado                ;carga el valor de la memoria EEPROM dirección 0

IF actualizado =0 THEN grabarRTC  ;si es la 1ra vez que corre ir a grabar RTC
                                  ;caso contrario sólo leer el RTC
INICIO:
   I2CREAD DPIN,CPIN,%11010000,0,[segu]        ;leer los datos de mem. 0,
   I2CREAD DPIN,CPIN,%11010000,1,[minu]        ;1,2,..y guardarlos en sus
   I2CREAD DPIN,CPIN,%11010000,2,[hora]        ;respectivas variables
   I2CREAD DPIN,CPIN,%11010000,3,[diaS]
   I2CREAD DPIN,CPIN,%11010000,4,[diaF]
   I2CREAD DPIN,CPIN,%11010000,5,[mes]
   I2CREAD DPIN,CPIN,%11010000,6,[anio]

   LCDOUT $fe,1,HEX2 hora,":", HEX 2 minu,":", HEX 2 segu     ;mostrar la hora
                                                ; min y segs. en 2 dígitos (HEX2)
LCDOUT $fe,$c0                    ;saltar a la 2da línea del LCD
                                  ;mostrar día de la semana
   IF diaS=$1 THEN LCDOUT "Dom."
   IF diaS=$2 THEN LCDOUT "Lun."
   IF diaS=$3 THEN LCDOUT "Mar."
   IF diaS=$4 THEN LCDOUT "Mie."
   IF diaS=$5 THEN LCDOUT "Jue."
   IF diaS=$6 THEN LCDOUT "Vie."
   IF diaS=$7 THEN LCDOUT "Sab."

LCDOUT $fe,$c5, HEX 2 diaF,"/"    ;mostrar el día del mes /
LCDOUT $fe,$cB,"/20", HEX 2 anio  ; mostrar año /20 + 04

LCDOUT $fe,$c8                    ;pasar a la casilla 8
   IF mes=$1 THEN LCDOUT "ene"    ;mostrar el mes
   IF mes=$2 THEN LCDOUT "feb"
   IF mes=$3 THEN LCDOUT "mar"                          continúa …
```

```
   IF mes=$4 THEN LCDOUT "abr"
   IF mes=$5 THEN LCDOUT "may"
   IF mes=$6 THEN LCDOUT "jun"
   IF mes=$7 THEN LCDOUT "jul"
   IF mes=$8 THEN LCDOUT "ago"
   IF mes=$9 THEN LCDOUT "sep"
   IF mes=$10 THEN LCDOUT "oct"
   IF mes=$11 THEN LCDOUT "nov"
   IF mes=$12 THEN LCDOUT "dic"
PAUSE 500                                       ;esperar 0,5 segundos

GOTO inicio                                     ;volver a leer los datos

; ************************ subrutina grabar *********************
grabarRTC:

I2CWRITE DPIN,CPIN,%11010000,0,[$00]            ;setear 00 segundos
   PAUSE 10                                     ;retardo para finalizar grabación
I2CWRITE DPIN,CPIN,%11010000,1,[$30]            ;setear 30 minutos
   PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,2,[$13]            ;setear las 13 horas
   PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,3,[$2]             ;setear día lunes, D=1,L=2
   PAUSE 10                                     ;M=3, M=4, J=5, V=6, S=7
I2CWRITE DPIN,CPIN,%11010000,4,[$27]            ;setear día 27 del mes
   PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,5,[$9]             ;setear mes septiembre
   PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,6,[$04]            ;setear año  04
   PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,7,[$10]            ;control %00010000 para
   PAUSE 10                                     ;encender el led cada 1 seg.


WRITE 0,1                       ;escribe en la memoria 0 el valor de 1 para que no
                                ;se vuelva a grabar otra vez estos datos en el RTC
GOTO inicio                     ;ir a presentar los datos en el LCD
END
```

Figura 5.9.13.2. reloj DS1307.pbp *Programa para escribir y leer datos en un RTC DS1307.*

Como podrán observar la batería es lo que le mantiene en funcionamiento al RTC cuando no hay alimentación DC, por tal razón cuando apagamos todo el circuito, y luego lo volvemos a prender, notamos que el reloj no se ha desigualado, pero si retiramos la batería, el reloj se detiene cuando lo cortamos la alimentación del circuito, y cuando se lo vuelve a conectar, sigue corriendo el tiempo pero continúa en el segundo que se quedó en el instante que se le cortó la alimentación.

El transistor sirve para encender el led con la fuente que alimenta el circuito y se apaga cuando deja de alimentarse el circuito, por lo que la batería sólo alimenta al CI. DS3107 mientras no hay alimentación en el pin VCC.

Se debe entender que el PIC está leyendo los datos del RTC cada 0,5 segundos, cuando en realidad debería leer cada segundo. Para mejorar este programa podemos utilizar una interrupción por cambio de estado en el pin B.0, aquí conectamos la señal SQW del RTC para que el PIC ejecute una subrutina de interrupción y lea los datos del RTC exactamente cuando el RTC le diga que transcurrió un segundo, con esto tenemos al PIC disponible para otras aplicaciones y no se quedaría esclavizado al RTC leyendo datos 2 veces por segundo. Para aplicar la interrupción ver literal 5.10.1 Utilizando la interrupción del puerto B.0.

Figura 5.9.13.3. *Fotografía de un módulo RTC del entrenador experto de PIC´S EE-02.*

5.9.14. PROYECTOS PROPUESTOS DE COMUNICACIÓN.

1. Haga un proyecto en el que el PIC muestre un mensaje en 2 líneas del LCD que son enviados desde la ventana de comunicación serial de microcode.

2. Conecte 2 PIC´S en serie y envíe texto desde un teclado hexadecimal hacia el otro PIC el cual lo presentará en la pantalla del LCD.

3. Haga un tablero de control en VB para un motor PAP, que contenga 2 botones de los cuales el uno hace girar en sentido horario mientras permanece presionado, al soltar el botón el motor debe detenerse, el 2do botón funciona igual sólo que en el otro sentido.

4. Utilice un LCD, un PIC y una memoria serial para el siguiente proyecto: desde la ventana de comunicación serial de microcode, envíe un texto a 2400N81, el PIC lo muestra en la pantalla del LCD y luego almacena en la memoria serial, al enviar desde el computador la letra L, el PIC debe interpretarlo como leer el dato de la memoria y enviarlo al computador, si el PC envía una B, el PIC debe borrar el contenido de la memoria, para probarlo simplemente envíe nuevamente la letra L desde el PC.

5. Con el proyecto del reloj calendario, haga que el PIC active una chicharra por 200mls. cada MINUTO.

5.10 INTERRUPCIONES

5.10.1. UTILIZANDO LA INTERRUPCIÓN DEL PUERTO B.0.

Existen aplicaciones en donde un evento es muy importante atenderlo, por ejemplo cuando algún dispositivo intenta comunicarse con el PIC, en un sistema de seguridad en donde una zona es más importante que las demás zonas o como el ejercicio anterior en donde cada segundo envía un pulso SQW, podríamos utilizar este pulso para indicar al PIC que debe leer los nuevos datos (segundo, minutos, hora, etc.).

Como práctica para poder entender la interrupción en el cambio de estado del Portb.0, haremos un parpadeo de un led rojo cada 200mls, y cuando exista una interrupción externa (pulsador), deja de ejecutarse el programa y atiende un Handler (subrutina) el cual contiene un programa en donde se enciende un led verde por 1 segundo, una vez terminado el programa de interrupción, retorna al programa principal en el lugar mismo donde ocurrió la interrupción.

NOTA: También existen otras fuentes de interrupción a más del puerto B.0, como el cambio de estado del puerto B.4 al Puerto B.7, los TIMER0, 1 y 2, por lo que se recomienda leer las hojas de datos del PIC16F628A.

LAS DECLARACIONES ON INTERRUPT, DISABLE, RESUME Y ENABLE. Todas estas declaraciones sirven para ejecutar un handler (subrutina con RESUME) de interrupción.

ON INTERRUPT GOTO prog2; quiere decir en caso de darse una interrupción suspender el programa actual e ir a prog2.

DISABLE; sirve para deshabilitar la interrupción, en caso de que no deseemos que atienda la interrupción, como por ejemplo cuando ya está atendiendo una interrupción.

RESUME; equivale al RETURN de un GOSUB, en donde retorna a la línea del programa en donde ocurrió la interrupción.

ENABLE; quiere decir habilitar nuevamente la interrupción, después de esto todas las interrupciones, son atendidas.

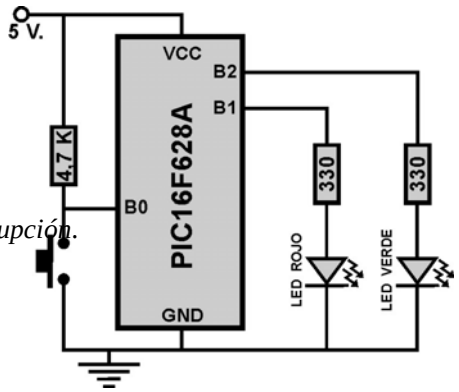


Figura 5.10.1.1. *Conexión de 2 leds y un pulsador para la práctica de interrupción.*

MATERIALES.
-1 led rojo 5 mm.
-1 led verde 5mm.
-2 resistencias de 330 Ω
-1 resistencia de 4,7 K Ω
-1 pulsador NA.

```
                ;programa para manejar una interrupción en el p B.0
LED  VAR PORTB.1
LED2 VAR PORTB.2

ON INTERRUPT GOTO verde        ; en caso de existir una interrupción ir a verde
INTCON = %10010000             ;habilita la interrupción B.0

PROG:                          ;programa principal
   HIGH LED                    ;encender el led rojo
   PAUSE 200
   LOW LED
   PAUSE 200
GOTO PROG                      ;mantener en este lazo

   DISABLE                     ;deshabilita las interrupciones en el handler
verde:                         ;handler de la interrupción
   HIGH LED2
   PAUSE 2000
   LOW LED2
   INTCON = %10010000          ;habilita la interrupción B.0
   RESUME                      ;retorna a programa principal donde ocurrió la inte.
   ENABLE                      ;Habilita las interrupciones después del handler
END
```

Figura 5.10.1.2. interrupt.pbp *Programa para practicar con la interrupción en el puerto B.0.*

Como pudieron observar, en el momento que se presiona el pulsador, automáticamente sale del programa y atiende el handler de interrupción llamado verde, finalizado esto el **RESUME** lo retorna al lugar en donde ocurrió la interrupción, sin embargo se puede poner **RESUME** prog3, esta vez va a una subrutina prog3 e ignora el retorno al lugar de la interrupción, otra recomendación importante es que si queremos atender rápidamente una interrupción, no debemos poner PAUSES muy largos como por ejemplo **PAUSE** 10000, equivalente a 10 segundos, si se da la interrupción en la mitad del pause (5 segundos), deberá esperar a que termine el pause para ir al handler de interrupción, es decir los 5 segundos restantes, lo mejor para estos casos es encerrar el pause en lazos **FOR... NEXT**, de la siguiente manera:

```
FOR x = 1 TO 100          ; repetir 100 veces (equivale a 10 segundos)
PAUSE 100
NEXT
```

Por lo que la atención al handler de interrupción será en 100 mls después de la interrupción.

Cabe también indicar que en esta práctica la interrupción se da sólo cuando existe un cambio de estado de 0 a 1 en el puerto B0, por lo que se habrán dado cuenta que si mantienen pulsado la tecla, no se genera la interrupción, sino cuando la soltamos, para que la interrupción se genere en el flanco de bajada, es decir cambio de estado de 1 a 0, debemos adicionar después de INTCON=%10010000, la siguiente línea para modificar el bit INTEDG del registro OPTION:

```
OPTION_REG.6=0      ;modificar el bit 6 del registro OPTION, activa en flanco de bajada a B.0
```

5.10.2. UTILIZANDO LA INTERRUPCIÓN DEL PUERTO B.4 Al B.7.

Para trabajar con la interrupción por cambio de estado del puerto b.4 al puerto b.7, podemos utilizar el mismo programa 5.10.1.2, solamente cambiando las 2 líneas que dicen INTCON=%10010000 por INTCON=%10001000, que quiere decir habilitar la interrupción de los puertos B.4 al B.7, para mayor información revise en los Datasheets el registro INTCON.

Para el diagrama de conexión debemos colocar 4 pulsadores en los puerto B.4 al B.7, no se debe dejar sueltos estos pines ya que produce errores y salta al handler de interrupción en cualquier momento. Se puede desconectar el pulsador del puerto B.0, ya que no se está atendiendo esta interrupción.

5.10.3. RELOJ DIGITAL UTILIZANDO LA INTERRUPCIÓN DEL TMR0.

El TMR0, es una valiosa herramienta que disponen los PIC´S, para el caso del PIC16F628A, dispone de 3 TIMERS, 1 de 16 bits (TMR1) y 2 de 8 bits, los TMR0 y TMR2, la calibración para estos contadores, se dan en el registro OPTION, aquí se puede seleccionar si el incremento es con flanco de subida o de bajada y si la fuente es externa (pin A4/TOCKI) o interna (Oscilador), en nuestro caso será interna generada por el Oscilador. Cuando el conteo del temporizador TMR0, llega a 256 y pasa a 0, se genera una interrupción, para que esto suceda se debe habilitar el registro INTCON bit 7 (GIE = 1), y también el bit 5 (TOIE =1), quedando así:

```
INTCON=%10100000                    ; habilita interrupción por TMR0
```

En el registro OPTION se debe definir la rata del preescalador, se debe poner:

```
OPTION_REG=%01010110      ; preescalador 1:128, asignado al TMR0 y ciclo de reloj interno.
```

Para poder entender mejor estos registros y sus funciones tenga a la mano la hoja de datos.

La práctica a realizarse consiste en ejecutar una interrupción, cada vez que el contador del TMR0, llega a 256, pero no empieza desde 0 ya que se le asignó un valor inicial de 4 (ver la línea # 70 del programa TMR0-reloj.pbp), por lo que el tiempo sería 128 x 252 = 32256 uS y esto repetido 31 veces, conseguimos acercarnos más al tiempo de 1 segundo (32256 uS x 31 = 999936 uS), luego de esto se incrementa la variable segun = segun +1 y se actualiza el LCD.

El proyecto dispone de 2 pulsadores para poder igualar la hora, uno aumenta los minutos y otro aumenta las horas, la gran desventaja de este proyecto es que si se corta la alimentación

del circuito, se resetea la hora (12:00:00), lo que no sucede con un RTC.

MATERIALES.
-1 LCD 2 x 16
-1 resistencia de 10 Ω
-1 potenciómetro de 10 K Ω
-2 resistencias de 4,7 K Ω
-2 pulsadores NA (Normalmente Abierto)
-1 oscilador cristal de 4 MHZ
-2 capacitores cerámicos de 22pF.

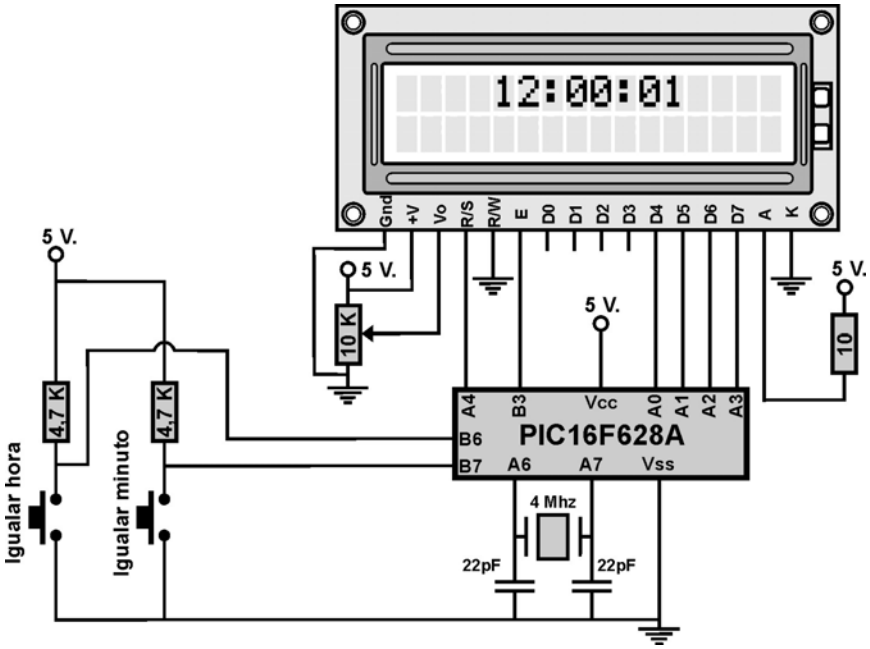


Figura 5.10.3.1. *Diagrama de conexión para hacer un reloj digital con interrupción en el contador del TMR0, se utilizó un cristal externo para conseguir mayor precisión.*

```
@ device XT_OSC                 ;exige utilizar cristal externo  XT de 4MHZ

hora    VAR    BYTE             ;definir variable hora
minut   VAR    BYTE             ;definir variable minuto
segun   VAR    BYTE             ;definir variable segundos
cuenta  VAR    BYTE             ;definir variable contador del timer
actual  VAR    BYTE             ;definir variable actualizar LCD
x       VAR    BYTE             ;definir variable x contador
```

continúa …

```
hora = 12                                   ; setea tiempo a 12:00:00
minut = 0
segun = 0
cuenta= 0
actual= 1
                                ;setea TMR0 de interrupción cada 32768 microsegundos
OPTION_REG = %1010110           ;setea TMR0 configurando y habilita PORTB pullups
INTCON = %10100000                      ;hablilita TMR0 interrupción interna
ON INTERRUPT GOTO interrup

INICIO:
      IF PORTB.7 = 0 THEN incmin                ;botón para igualar minutos
      IF PORTB.6 = 0 THEN inchr                 ;botón para igualar horas

actualiza:  IF actual = 1 THEN                  ;chequea si hay que actualizar LCD

 LCDOUT $fe,1,"    ",dec2 hora,":",dec2 minut,":",DEC2 segun

actual = 0                                      ;pantalla actualizada
      ENDIF
 GOTO INICIO

; ************************para igualar la hora ******************
incmin: minut = minut + 1
      IF minut >= 60 THEN minut=0
      GOTO pausa

inchr:  hora = hora + 1
      IF hora >= 24 THEN hora=0
      GOTO pausa

pausa: FOR x = 1 TO 20                  ; retardo de 200 mls
      PAUSE 10                  ;pasos de 10 mls para no perder interrupciones
      NEXT x
      actual = 1                        ;indica actualizar pantalla LCD
GOTO actualiza

;*************** Handler de interrupciones para incrementar contador ***************
DISABLE                                 ;deshabilita interrupciones durante el proceso
interrup:
   cuenta = cuenta + 1                  ;cuenta las interrupciones del TMR0
 TMR0=4                                 ;resta 4 al contador de 256 del TMR0

      IF cuenta < 31 THEN reset         ;31 cuentas (32256ms = 999936uS)
      cuenta = 0
      segun = segun + 1
      IF segun >= 60 THEN
         segun = 0
         minut = minut + 1                                  continúa …
```

```
      IF minut >= 60 THEN
         minut = 0
         hora = hora + 1
      IF hora >= 24 THEN
         hora = 0
         ENDIF
         ENDIF
         ENDIF
actual = 1                                  ;actualizar LCD

reset:
    INTCON.2 = 0                            ;resetea la bandera de interrupción del TMR0

RESUME
END
```

Figura 5.10.3.2. TMR0-reloj.pbp *Programa para practicar con la interrupción del TMR0.*

5.10.4. PROYECTOS PROPUESTOS CON INTERRUPCIONES.

1. Conecte un pulsador en el Puerto B.0, habilite la interrupción de este pin y muestre en un LCD las veces que se produce una interrupción por cambio de estado en el puerto B.0

2. Haga un parpadeo de un led cada 200 mls. en el puerto B.2, habilite las interrupciones del puerto B4 al B7 y coloque un pulsador en cada pin, en la pantalla del LCD debe indicar el puerto el cual se activó una interrupción, pulse cualquiera de los 4 botones.

5.11 CONVERSOR A/D

5.11.1. CONVERSOR ANÁLOGO DIGITAL DEL PIC 16F81X.

Esta nueva familia de PIC´S los 16F818 y 16F819, iniciaron su aparición a mediados del 2004, por lo que será muy común encontrarlos en las tiendas electrónicas ya que posee 5 conversores A/D de 10 bits c/u. y su memoria RAM y EEPROM, son más extensas que la del PIC16F628A, y además su costo es también inferior.

	PIC16F84A	PIC16F628A	PIC16F819
Memoria de programa	1024	2048	2048
Memoria datos EEPROM	64	128	256
Memoria RAM	68	224	256
Pines de entrada/salida	13	16	16
Comparadores	-	2	-
Conversores A/D	-	-	5

Figura 5.11.1.1. *Tabla de comparación entre el PIC16F84A, el PIC16F628A y el PIC16F819.*

El PIC16F81X al igual que el PIC16F628A, disponen de oscilador interno, pero el PIC16F81X dispone de 8 calibraciones para el oscilador interno, desde 31,25 KHZ hasta 8 MHZ.

Para poder empezar a practicar con este nuevo PIC, haremos un ejercicio de lectura de un conversor A/D del PIC16F819, su funcionamiento es muy simple, los pines del puerto A (A0, A1, A2, A3 y A4), son capaces de detectar el nivel de voltaje que ingresan a ellos, por ejemplo: si utilizamos un conversor A/D de 8 bits, quiere decir que entre los voltajes de referencias, digamos que Vref-= 0V. y Vref+=5V., los 5 V. los dividirá en 255 segmentos, (19,6 mlV.), entonces si a la entrada del pin A/D ingresan 19,6 mlV, el registro ADCIN nos entregará un valor de 1, y así tenemos que para:

0V ADCIN =0
2,49V ADCIN= 127
5V ADCIN= 255

En definitiva nos dice que voltaje está entrando por un pin pero en valores de una variable de 255, para poder entender mejor haremos la siguiente práctica que consiste en calibrar el oscilador interno a 4 MHZ, leer el canal 0 a 8 bit (puede leer a 10 bits) el voltaje que ingresa y mostrarlo en un LCD.

Para poner en funcionamiento los conversores A/D, debemos cambiar unos bits del registro ADCON1, ver tabla de la figura 5.11.1.2., en donde se debe configurar cual es el canal o los canales que deseamos utilizar, así como también cuales son los voltajes de referencia. En nuestro caso utilizaremos ADCON1=%00001110, que significa que sólo vamos a utilizar el canal 0 (A.0) con voltajes de referencias positivo y negativos, los mismos que utiliza el PIC para su alimentación, es decir en este caso 0 y 5V.

MATERIALES.
-1 PIC16F819
-1 LCD 2 x 16
-1 resistencia de 10 Ω
-2 potenciómetros de 10 K Ω.

PCFG	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	AVDD	AVSS	5/0
0001	A	VREF+	A	A	A	AN3	AVSS	4/1
0010	A	A	A	A	A	AVDD	AVSS	5/0
0011	A	VREF+	A	A	A	AN3	AVSS	4/1
0100	D	A	D	A	A	AVDD	AVSS	3/0
0101	D	VREF+	D	A	A	AN3	AVSS	2/1
011x	D	D	D	D	D	AVDD	AVSS	0/0
1000	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1001	A	A	A	A	A	AVDD	AVSS	5/0
1010	A	VREF+	A	A	A	AN3	AVSS	4/1
1011	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1100	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	A	AVDD	AVSS	1/0
1111	D	VREF+	VREF-	D	A	AN3	AN2	1/2

D=Digital A= Análogo

Figura 5.11.1.2. *Tabla de configuración para el registro ADCON1, noten que ADCON1=7 corresponde a PCFG =011x, por lo tanto todo el puerto A se convierte en pines digitales, algo parecido ocurría con el CMCON=7 del PIC16F628A, este apagaba los comparadores de voltaje y los volvía en pines digitales.*

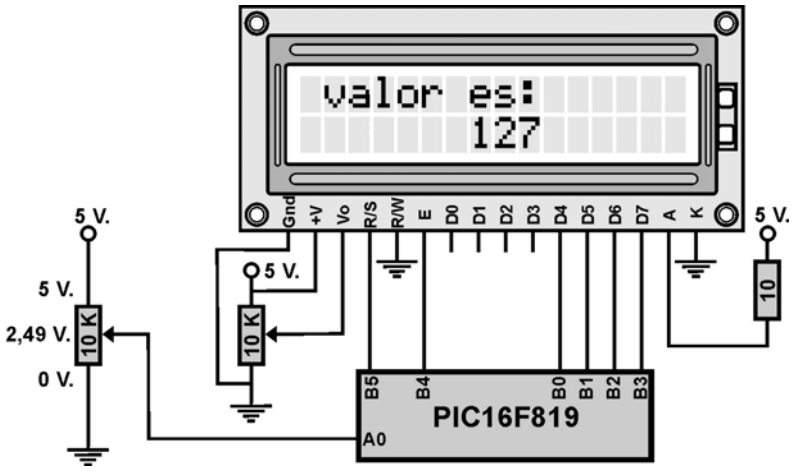


Figura 5.11.1.3. *Conexión de un potenciómetro de 10K para obtener un divisor de voltaje desde 0V. hasta 5 V., trate de poner el potenciómetro en 2,49 V. y observará en el LCD el dato 127.*

Otro cambio importante es que los pines que controlan al LCD, no pueden ser los predefinidos por pbp, por lo tanto debemos cambiarlos de posición utilizando **DEFINE**.

```
;calibraciones   %111 8MHZ   %110 4MHZ   %101 2MHZ   %100 1MHZ  %011 500KHZ
;%010 250KHZ   %001 125KHZ   %000 31.25KHZ adicionarle a todos 0100

OSCCON=%1100100                          ; calibra oscilador interno a 4 MHZ

DEFINE      LCD_DREG    PORTB            ;bit de datos del LCD empezando
DEFINE      LCD_DBIT    0                ;por B.0, B.1, B.2 y B.3
DEFINE      LCD_RSREG   PORTB            ;bit de registro del LCD conectar
DEFINE      LCD_RSBIT   5                ;en el puerto B.5
DEFINE      LCD_EREG    PORTB            ;bit de Enable conectar en el
DEFINE      LCD_EBIT    4                ;puerto B.4

DEFINE      ADC_BITS    8                ;Fija número de bits del resultado (5,8,10)
DEFINE      ADC_CLOCK   3                ;Fije EL CLOCK (rc = 3)
DEFINE      ADC_SAMPLEUS      50         ;Fije el tiempo de muestreo en uS.
            ;ADC_SAMPLEUS es el número de microsegundos que el programa espera
            ;entre fijar el canal y comenzar la conversión análoga/digital.

TRISA =%1                                ;el puerto A.0 es de entrada
ADCON1 = %00001110                ;el puerto A.0 es conversor los demás Digitales

datos   VAR   BYTE                       ;crear variable datos para guardar el resultado
PAUSE 500                                ;esperar 0,5 seg.

inicio:
        ADCIN 0, datos                   ;leer el canal 0 y guardarlo en datos
        LCDOUT $fe, 1, " valor es:"
        LCDOUT $fe,$c7, DEC datos        ;desplegar el valor de datos en decimal

        PAUSE 300

        GOTO inicio                      ;volver a medir el conversor A/D
END
```

Figura 5.11.1.4. AD-16F819-8.pbp *Programa para practicar con el conversor análogo digital*.

5.11.2. CONVERSOR ANÁLOGO DIGITAL DEL PIC 16F87X.

En esta práctica utilizaremos 3 conversores A/D de los 8 que dispone el PIC16F877A, este PIC se caracteriza por tener 40 pines, de los cuales 33 son puertos de entrada/salida, una memoria FLASH de 8192 palabras, una RAM de 368 bytes y una EEPROM de 256 bytes, por lo que este microcontrolador está destinado para proyectos grandes. Esta práctica nos servirá para poder familiarizarnos con PIC´S de mayor capacidad, cabe recalcar que es necesario un oscilador externo, ya que no posee oscilador interno, también debemos utilizar en el MCLR una resistencia Pull-Up, ya que no hay forma de deshabilitarlo como sucede con el PIC16F62X o el PIC16F81X.

PCFG3: PCFG0	AN7(1) RE2	AN6(1) RE1	AN5(1) RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs(2)
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

D=Digital A= Análogo

Figura 5.11.2.1. *Tabla de configuración para el registro ADCON1 del PIC16F877A, noten que ADCON1=7 convierte en pines digitales todos los pines del puerto A y puerto E.*

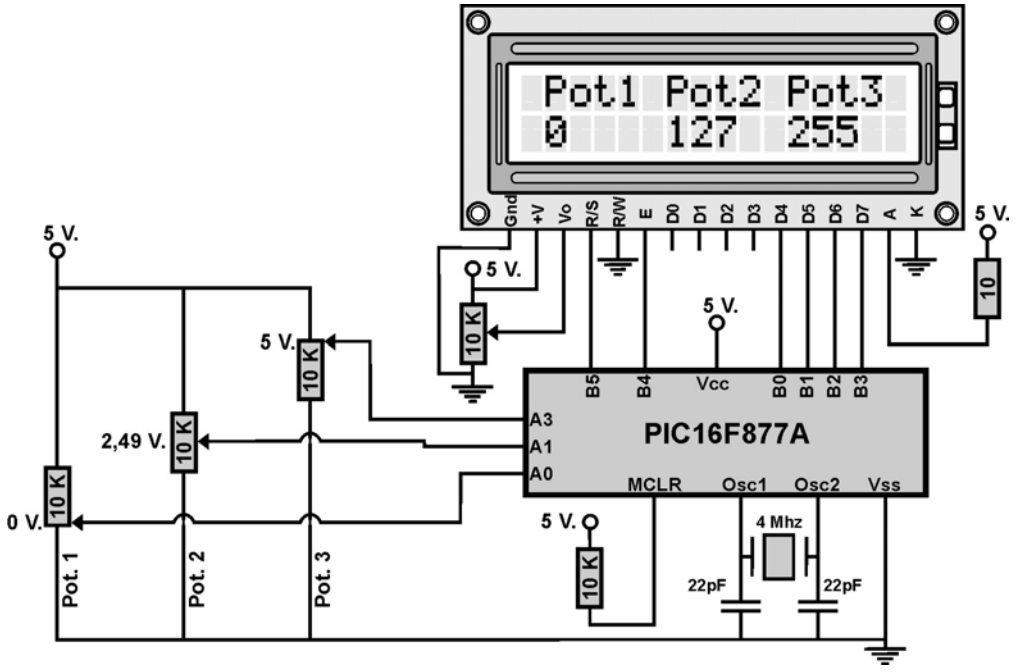


Figura 5.11.2.2. *Esquema de conexión de un PIC16F877A para medir los 3 voltajes que ingresan por el divisor de voltaje de cada potenciómetro.*

MATERIALES.
-1 PIC16F877A
-1 LCD 2 x 16
-1 resistencia de 10 Ω
-1 resistencia de 10 K Ω
-4 potenciómetros de 10 K Ω
-1 oscilador cristal de 4 MHZ
-2 condensadores cerámicos de 22pF.

```
DEFINE      LCD_DREG    PORTB       ;bit de datos del LCD empezando
DEFINE      LCD_DBIT    0           ;por B.0, B.1, B.2 y B.3
DEFINE      LCD_RSREG   PORTB       ;bit de registro del LCD conectar
DEFINE      LCD_RSBIT   5           ;en el puerto B.5
DEFINE      LCD_EREG    PORTB       ;bit de Enable conectar en el
DEFINE      LCD_EBIT    4           ;puerto B.4
p1 VAR BYTE                         ;variable para almacenar potenciómetro 1
p2 VAR BYTE                         ;variable para almacenar potenciómetro 2
p3 VAR BYTE                         ;variable para almacenar potenciómetro 3
  ADCON1 =%100                      ;configura PortA 0,1, 3 en conversores A/D

Inicio:
      PAUSE 300
poten1:
    ADCON0 =%1000001                ;activar canal 0 a Fosc/8
    GOSUB medir
    p1= ADRESH
poten2:
    ADCON0 =%1001001                ;activar canal 1 a Fosc/8
    GOSUB medir
    p2= ADRESH
poten3:
    ADCON0 =%1011001                ;activar canal 3 a Fosc/8
    GOSUB medir
    p3= ADRESH

LCDOUT $fe, 1," Pot1 Pot2 Pot3"     ;limpiar LCD y sacar texto
LCDOUT $fe,$c1,#p1                  ;casilla 1 el valor decimal de p1
LCDOUT $fe,$c6,#p2                  ;casilla 7 el valor decimal de p2
LCDOUT $fe,$cb,#p3                  ;casilla 12 el valor decimal de p3
GOTO inicio

medir:                              ;subrutina para leer el conversor A/D
    PAUSEUS 50                      ;pausa para setear el canal
    ADCON0.2 = 1                    ;iniciar conversión
    PAUSEUS 50                      ;pausa para la conversión
    RETURN                          ;retornar al GOSUB que lo envió
END
```

Figura 5.11.2.3. AD-3-16F877A.pbp *Programa para practicar con 3 conversores A/D.*

5.11.3. TERMÓMETRO DIGITAL CON EL PIC 16F877A.

Para esta práctica necesitaremos el sensor de temperatura LM35, este dispositivo presenta en su pin OUT una variación de 10 mV por grado centígrado, su alimentación puede ser de 4 a 30 Voltios, y su rango de temperatura a sensar entre -55°C hasta 150 °C.

Para medir la temperatura se conecta el pin out del LM35 al puerto A.0, el cual está configurado como conversor A/D a 10 bits, este valor se almacena en la variable dato que tiene capacidad de 2 bytes (16 bits), el cual se lo divide para 128, debido a que la variable del conversor A/D de 10 bits lo presenta en 16 bits, de la siguiente manera: 1111111111000000, los 6 bits que contienen ceros se los debe ignorar, ya que sólo necesitamos los 8 bits del 1er byte más 2 bits del segundo byte, si este dato lo dividimos para 64 conseguiremos eliminar los 6 bits que corresponde a los ceros, de esta manera tendremos el dato a 10 bits, es decir el C A/D mostraría como valor máximo 1024, esto es una resolución de 5 mV, pero como el LM35 tiene incrementos de 10 mV, debemos bajar la resolución a 9 bits y eso se consigue dividiendo para 128, lo cual elimina 7 bits del 2do byte, de esta manera el valor más alto sería 512, esto es lo más cercano a la escala del LM35. El proyecto funciona de la siguiente manera: si la temperatura permanece entre 20°C y 24°C ninguno de los relés se activa, pero si la temperatura no se encuentra entre estos 2 rangos, se activa el relé que le corresponde, sea para calentar o enfriar el ambiente, si deseamos modificar los rangos de temperatura, presionamos el pulsador E, con los otros 2 botones aumentamos o disminuimos la temperatura mínima a comparar, y una vez que estemos de acuerdo presionamos la tecla E nuevamente, luego nos pide programar la temperatura máxima, procedemos igual que el caso anterior y cuando presionemos la tecla E, parpadeará tres veces el led, indicando que los nuevos valores ya fueron guardados en la memoria no volátil.

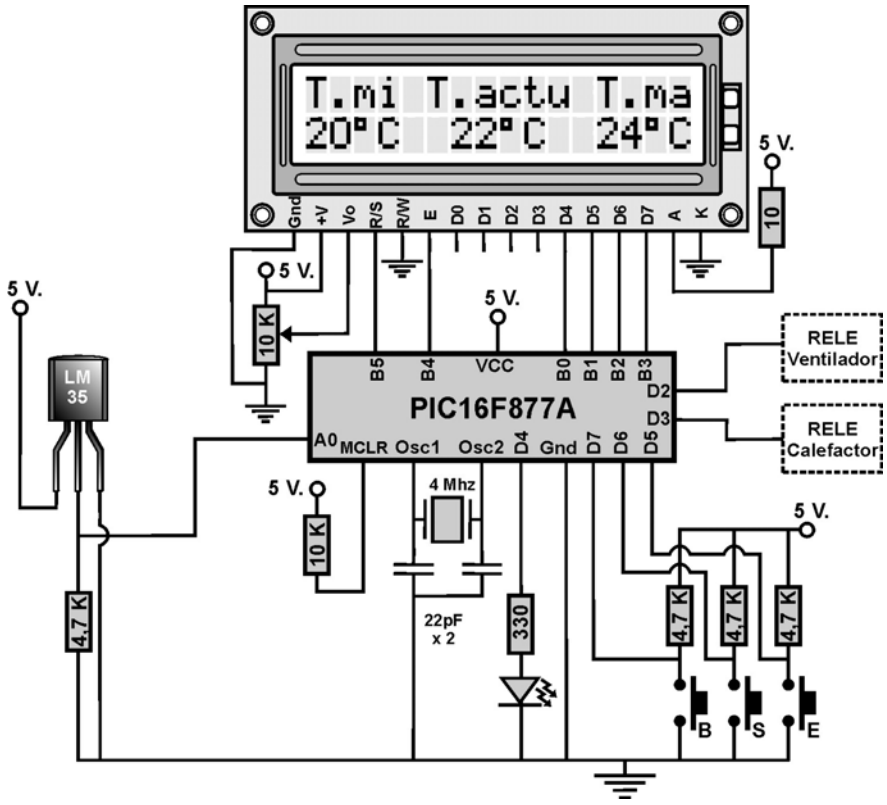


Figura 5.11.3.1. *Esquema de conexión de un PIC16F877A para medir la temperatura ambiental.*

MATERIALES.
-1 PIC16F877A
-1 LCD 2 x 16
-1 sensor de temperatura LM35
-1 resistencia de 10 Ω
-1 resistencia de 330 Ω
-1 resistencia de 10 K Ω
-6 resistencias de 4,7 K Ω
-1 potenciómetro de 10 K Ω
-1 led
-1 cristal oscilador de 4 MHZ
-2 condensadores cerámicos de 22pF
-2 relés 12 V.
-2 diodos rectificadores 1N4007
-2 transistores 2N3904.

```
DEFINE      LCD_DREG    PORTB       ;bit de datos del LCD empezando
DEFINE      LCD_DBIT    0           ;por B.0, B.1, B.2 y B.3
DEFINE      LCD_RSREG   PORTB       ;bit de registro del LCD conectar
DEFINE      LCD_RSBIT   5           ;en el puerto B.5
DEFINE      LCD_EREG    PORTB       ;bit de Enable conectar en el
DEFINE      LCD_EBIT    4           puerto B.4

DEFINE      ADC_BITS    10          ;Fije número de BITS del resultado (5,8,10)
DEFINE      ADC_CLOCK   3           ;Fije EL CLOCK (rc = 3)
DEFINE      ADC_SAMPLEUS 50         ;Fije el tiempo de muestreo en Us
            ;ADC_SAMPLEUS es el número de microsegundos que el programa espera
                  ;entre fijar el canal y comenzar la conversión analogica/digital.
TRISA =%1                           ;el puerto A.0 es de entrada
ADCON1 = %00001110                  ;el puerto A.0 es conversor los demás Digitales

dato     VAR  WORD                  ;crear variable dato para guardar
tempbaj VAR   BYTE
tempalt  VAR  BYTE
x        VAR  BYTE
g         CON  223                  ; g constante 223 este es el ASCII de grados
releF   VAR portD.2                 ;nombres para los pines
releC  VAR portD.3
led     VAR portD.4
enter   VAR portD.5
bsubir VAR portD.6
bbajar VAR portD.7

EEPROM 0,[20,24]                    ;contenido inicial de la EEPROM

inicio:                             ;3 parpadeos del led que indica funciona
FOR x =1 TO 3
   HIGH led                                                  continúa ...
```

```
   PAUSE 200
   LOW led
   PAUSE 200
NEXT

READ 0,tempbaj                          ;lee la EEPROM 0 y lo guarda en tempbaj
READ 1,tempalt                          ;lee la EEPROM 1 y lo guarda en tempalt

sensar:
   ADCIN 0, dato                        ;leer el canal 0 (A0) y guarde en dato
        LCDOUT $fe, 1, "T.mi T.actu T.ma"  ;limpiar LCD y sacar texto
   dato = dato /128                     ; el dato dividir para 128= C/AD de 9 bits
   LCDOUT $fe,$c6,DEC dato,g,"C"        ;Display el decimal de dato
   LCDOUT $fe,$c0,DEC tempbaj,g,"C"     ;Display el decimal de tempbaj
   LCDOUT $fe,$cc,DEC tempalt,g,"C"     ;Display el decimal de tempalt

   FOR x = 1 TO 50                      ;repetir 50 veces
   IF enter =0 THEN grabar1a
   PAUSE 10
   NEXT

   IF dato < tempbaj THEN calentar      ;si dato es<tempbaj ir a calentar
   IF dato > tempalt THEN enfriar
    LOW releC : LOW releF               ;apagar los 2 relés
GOTO sensar                             ;continuar sensando

calentar:
HIGH releC : LOW releF
GOTO sensar

enfriar:
HIGH releF : LOW releC
GOTO sensar

grabar1a:
GOSUB soltar

grabar1:
   LCDOUT $fe, 1, "Programar temp."
   LCDOUT $fe,$c0,"baja= ",DEC tempbaj ,g,"C"
   PAUSE 100
   IF bbajar=0 THEN restar1
   IF bsubir=0 THEN sumar1
   IF enter=0 THEN grabarA
GOTO grabar1

restar1:
   GOSUB soltar                         ;programa antirrebote de tecla
   IF tempbaj < 1 THEN grabar1
   tempbaj= tempbaj -1                                          continúa …
```

```
GOTO grabar1
sumar1:
   GOSUB soltar
   IF tempbaj > 40 THEN grabar1
   tempbaj= tempbaj + 1
GOTO grabar1

grabarA:
   GOSUB soltar
   WRITE 0,tempbaj                    ;escribir en la dirección 0 de la EEPROM

grabar2:
   LCDOUT $fe, 1, "Programar temp."
   LCDOUT $fe,$c0,"alta= ",dec tempalt ,g,"C"
   PAUSE 100
   IF bbajar=0 THEN restar2
   IF bsubir=0 THEN sumar2
   IF enter=0 THEN grabarB
GOTO grabar2

restar2:
   GOSUB soltar
   IF tempalt < 5 THEN grabar2
   tempalt= tempalt -1
GOTO grabar2

sumar2:
   GOSUB soltar
   IF tempalt > 50 THEN grabar2
   tempalt= tempalt + 1
GOTO grabar2

grabarB:
   GOSUB soltar
   WRITE 1,tempalt                    ;escribir en la dirección 1 de la EEPROM
GOTO inicio

soltar:                               ;programa antirrebote de tecla
   HIGH led
   PAUSE 150
   LOW led
soltar2:
   IF bbajar=0 THEN soltar2
   IF bsubir=0 THEN soltar2
   IF enter=0 THEN soltar2
   PAUSE 100
   RETURN
END
```

Figura 5.11.3.2. LM35-temp-16F877A.pbp *Programa para medir la temperatura ambiental.*

5.12 UTILIZANDO EL PIC12F6XX

5.12.1. PARPADEO DE LEDS EN EL PUERTO GPIO.

En ocasiones existen proyectos en los cuales no se necesitan más de 4 0 5 pines del PIC, un PIC de 16 I/O sería un desperdicio, por tal razón se incluye un pequeño ejercicio de un parpadeo de leds en el puerto gpio, el objetivo de esta práctica es familiarizarnos con esta familia de PIC´S.

	PIC12F629	PIC12F675
Memoria de programa	1024	1024
Memoria datos EEPROM	128	128
Memoria RAM	64	64
Pines de entrada/salida	6	6
Comparadores	1	1
Conversores A/D	-	4

Figura 5.12.1.1. *Tabla de comparación entre el PIC16F629 y el PIC12F675.*

Como podemos observar la única diferencia entre estas dos subfamilias es que el uno dispone de conversores A/D y el otro no, el ejercicio lo haremos para el PIC12F675 y para hacer que funcione en un PIC12F629, sólo se debe eliminar la línea que configura el conversor A/D ANSEL=0, ya que este último no dispone de tales conversores A/D, su oscilador interno RC da una frecuencia de 4 MHZ, sin embargo se puede utilizar un oscilador externo de hasta 20 MHZ.

MATERIALES.
-1 PIC12F629 o PIC12F675
-5 resistencias de 330 Ω
-5 diodos leds.
-1 capacitor de 0,1 uF (103)

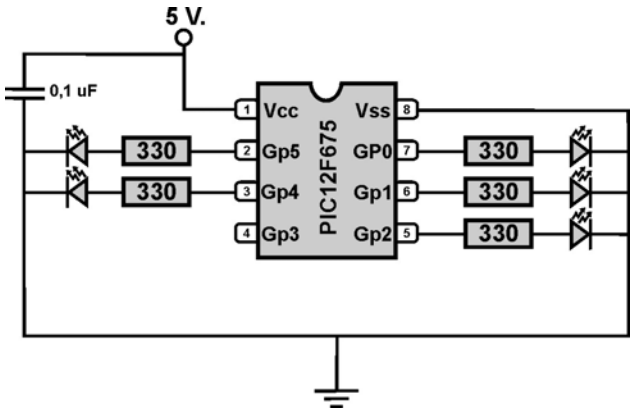


Figura 5.12.1.2. *Esquema de conexión del PIC12F6XX, el GPIO.3 es de colector abierto(MCLR)*

```
CMCON=%111                  ;apaga comparadores de voltaje
ANSEL=%0000                 ;apaga C.A/D todos los pines del GPIO digitales
X VAR BYTE                  ;crea variable de 255

HIGH gpio.1                 ;encender el led del pin gpio.1
PAUSE 500
LOW gpio.1

trisio=0                ;todos los pines gpio son de salida

INICIO:
FOR x=1 TO 3
   gpio=%110111             ;encender los leds de todos los leds (menos el gpio.3)
   PAUSE 200
   gpio=%000000             ;apagar todos los leds
   PAUSE 200
NEXT
  PAUSE 1000
GOTO INICIO

END
```

Figura 5.12.1.3. Prueba-12F6XX.pbp*Programa para practicar con el PIC12F629 o PIC12F675.*

Al igual que el PIC16F628A, el MCLR se puede deshabilitar justamente en los fusibles de programación del IC-prog. También observarán que en el momento de grabar el programa primero lee una calibración interna, este dato se encuentra en el último casillero de la memoria FLASH en la **03F8** y se presenta de la siguiente forma:

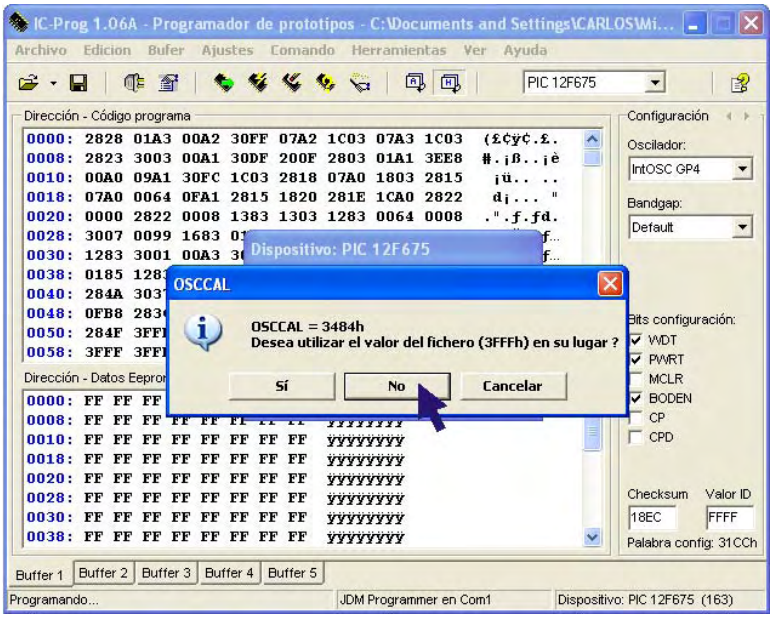


Figura 5.12.1.4. *Pantalla del IC-prog, en la que indica el valor de calibración del oscilador.*

En este caso el dato leído es 3484h (anótelo en un papel porque si usted pone borrar el PIC, este dato se puede perder), el programa IC-prog pregunta si quiere utilizar el valor 3FFFh, usted debe poner **NO**, para que el valor de calibración que le suministra el fabricante sea aceptado, caso contrario si usted presiona SI, está poniendo el valor 3FFFh de calibración para el oscilador interno. Si el valor de calibración se le ha borrado accidentalmente, y si usted anotó el valor en este caso 3484h, puede ayudarlo a colocar en su sitio, escribiéndolo en el programa directamente en la línea **03F8**.

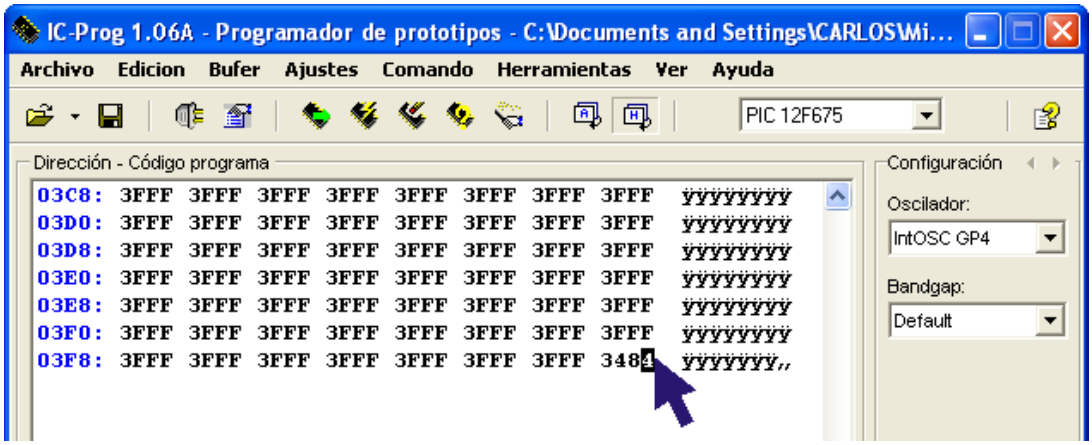


Figura 5.12.1.5. *Pantalla del IC-prog en donde se muestra escribiendo manualmente la calibración del oscilador para que vuelva a reponerse en el PIC12F675.*

5.12.2. PROYECTOS PROPUESTOS CON EL CONVERSOR A/D.

1. Muestre en un LCD el VOLTAJE (0 a 5 V.) con 2 decimales, que mide un PIC16F877A por su conversor A/D, utilice un potenciómetro como divisor de voltaje.

2. En el proyecto 5.11.2 deshabilite el conversor AN3 y muestre en el LCD sólo la diferencia que existe entre las variables de los coversores AN0 y AN1, es decir P1 y P2.

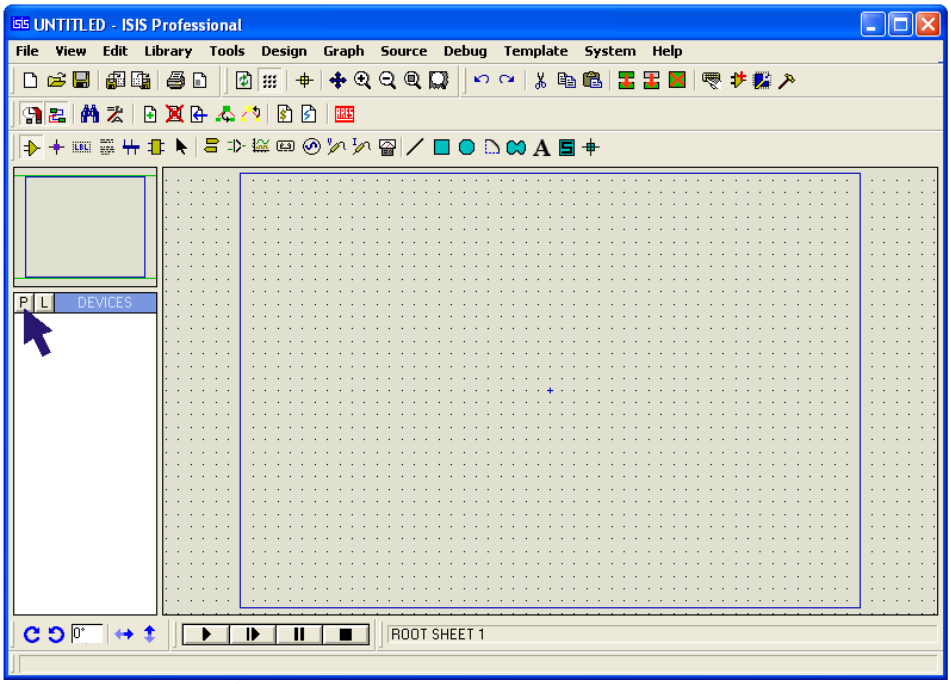
3. En el proyecto 5.11.3 reemplaze el LM35 por una Fotocelda y haga que se encienda un foco de 110 Voltios AC. al oscurecer.

CAPÍTULO 6
SIMULACIÓN Y RUTEADO CON PROTEUS

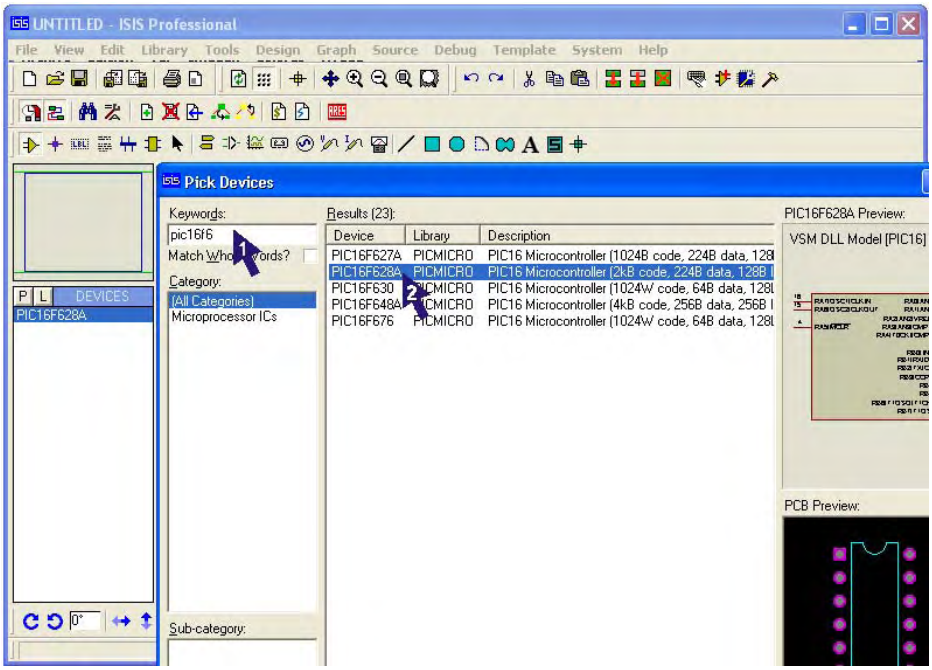
Una de las herramientas más importantes disponibles en Internet es el simulador de circuitos PROTEUS de Labcenter Electronics, dispone de una gran variedad de microcontroladores de la familia PIC, INTEL, ATMEL, ZILOG y MOTOROLA, además de una gran variedad de elementos electrónicos como displays de 7 segmentos, LCD, LCD gráficos, teclados, pulsadores, leds, diodos, resistencias, motores PAP, etc. Por tal motivo en esta edición se ha incluido su manejo, simulación y creación de circuitos impresos ya que en el mismo paquete se incluye el ruteador ARES de PROTEUS, en esta ocasión iniciaremos el montaje del primer proyecto del capítulo 5, el led intermitente.hex.

6.1. SIMULACIÓN DEL LED INTERMITENTE.

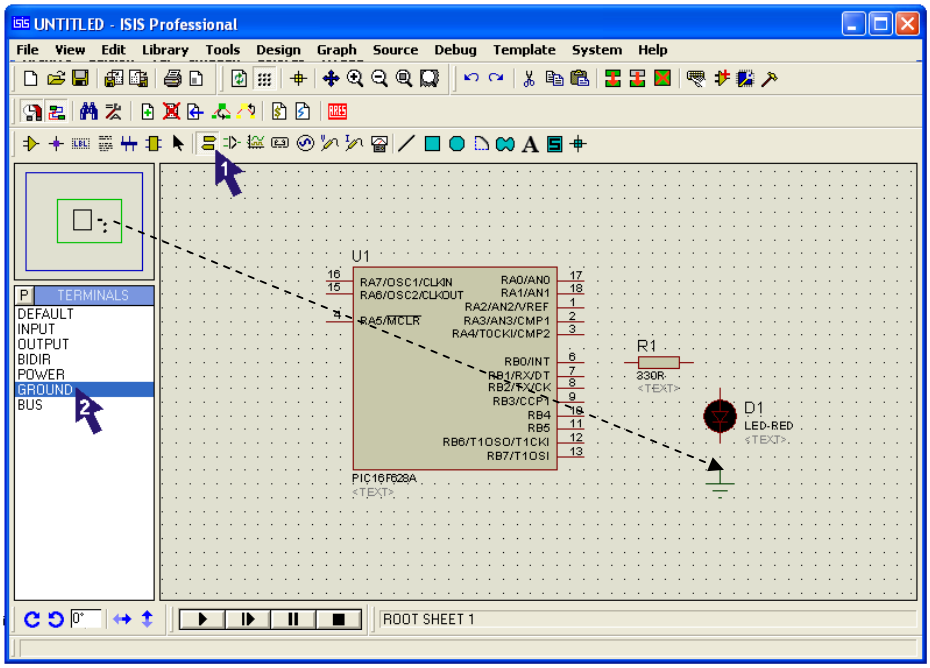
Primero que nada debe tener instalado el simulador PROTEUS, luego ejecute el archivo ISIS, se presentará una pantalla similar al siguiente gráfico, al iniciar el programa por defecto está seleccionado component, si no lo está presione  (component) luego presione en P (Pick Devices).



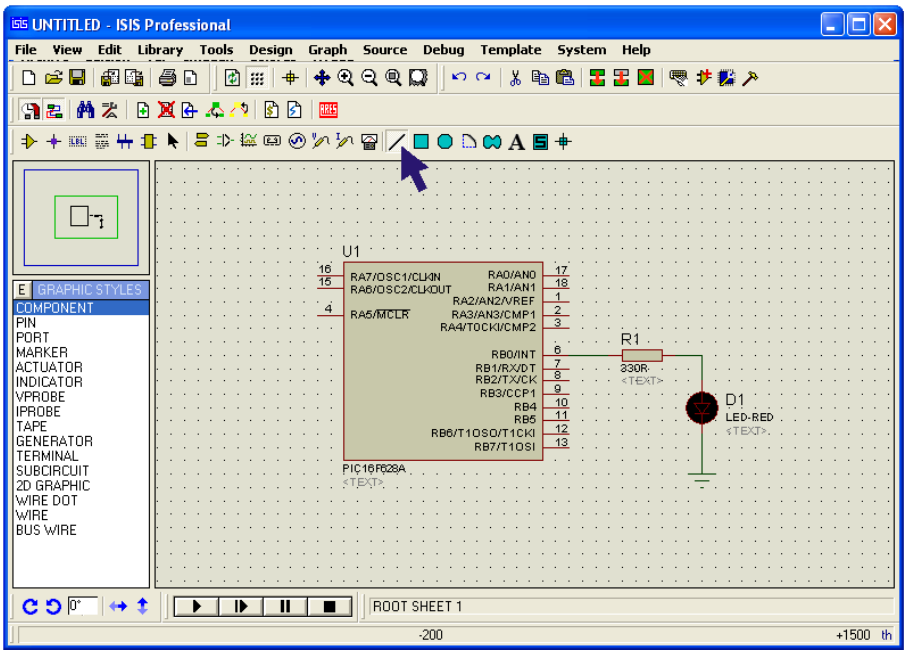
Aparecerá una nueva ventana con una librería que incluye varios dispositivos ya mencionados anteriormente, escriba pic16f6 en la casilla Keywords: luego de doble clic en PIC16F628A, observará que al lado izquierdo en DEVICES van apareciendo los dispositivos que se van seleccionando, escriba minres y seleccione MINRES330R, escriba led y seleccione LED-RED.



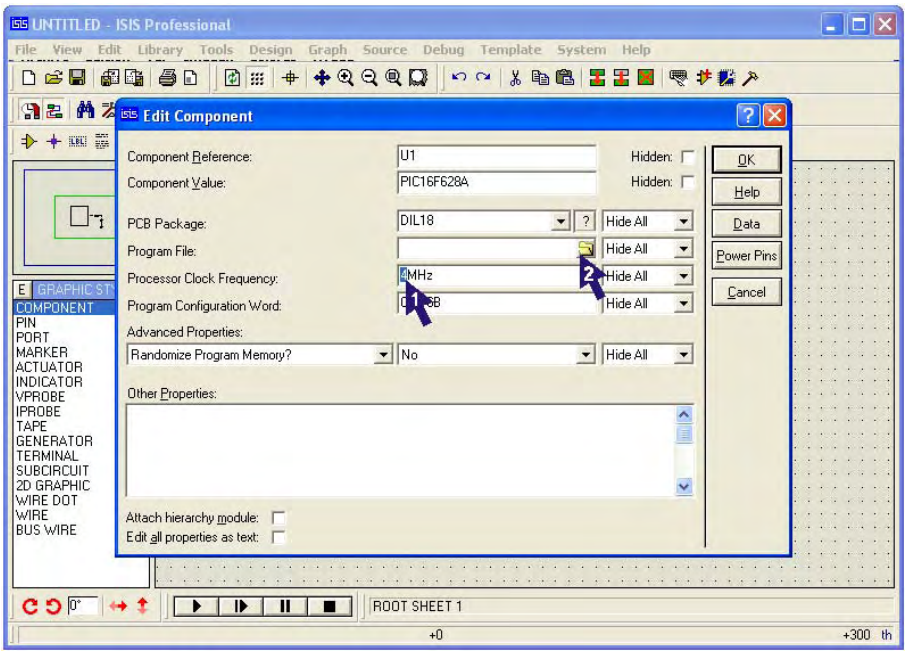
Para poder encender el LED, se debe cerrar el circuito con una puesta a tierra, de un clic en Inter sheet Terminal  luego seleccione GROUND y arrástrelo a la pantalla debajo del LED.



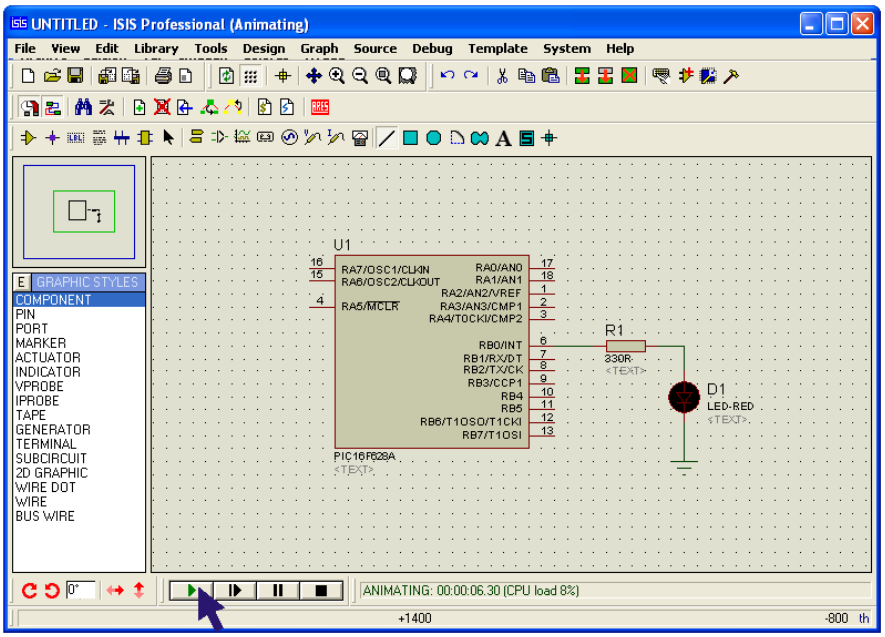
Ahora proceda a unir los dispositivos seleccionando un cable en 2D graphics line, enlace como lo muestra la siguiente figura, notará que cuando se acerca al terminal de cada dispositivo el cursor muestra una equis, en ese momento de un clic y luego otro clic en el dispositivo a unir.



Una vez armado el proyecto proceda a cargar el archivo a correr, para esto de un clic derecho sobre el PIC, notará que cambia a color rojo, luego un clic izquierdo (si da otro clic derecho borra el dispositivo), aparecerá una pantalla nueva de Edición de componente, en este cambie el oscilador de 1 MHZ a 4MHZ y en Program File abra el archivo hexadecimal led intermitente.hex, una vez abierto el archivo presione OK.

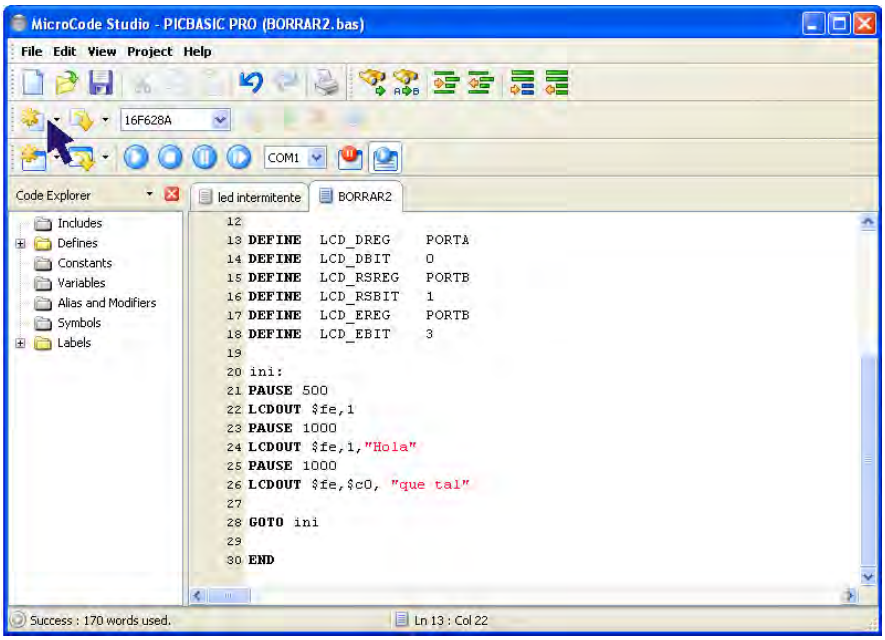


Para iniciar la simulación presione PLAY localizado en la parte inferior, ahora podrá ver la simulación en tiempo real, el led empezará a cambiar de color cada segundo, note además que en esta simulación el PIC no necesita ser alimentado.

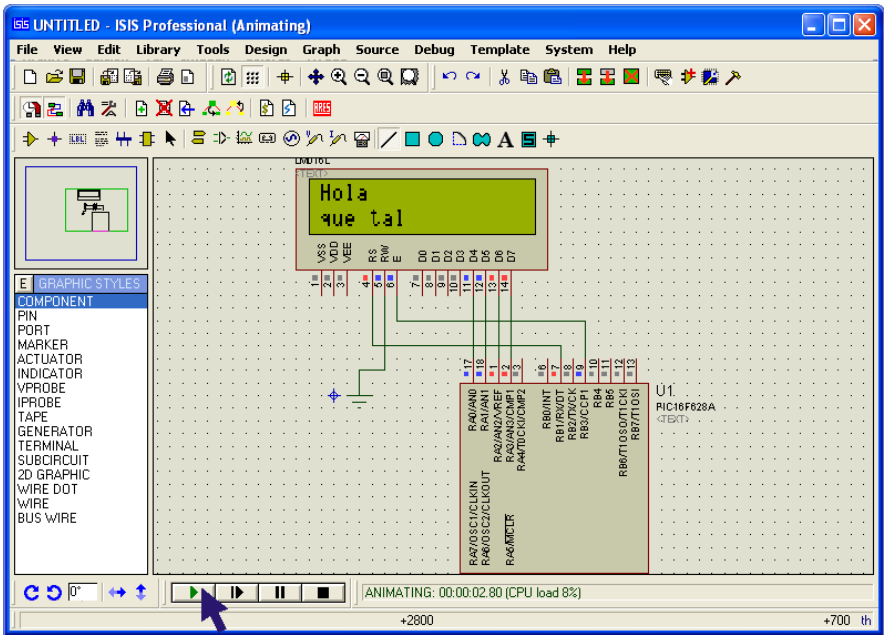


6.2. SIMULACIÓN DE UN LCD 2X16.

Este es un caso especial ya que el PIN RA.4 no puede ser utilizado para el manejo del LCD, por tal razón se debe definir una nueva posición para este PIN, haga un ejercicio similar al siguiente:

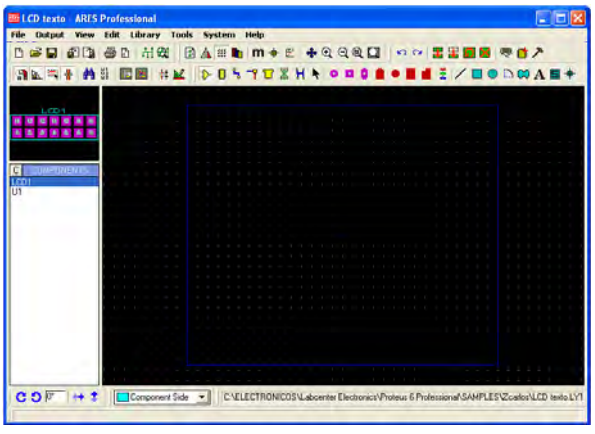


Una vez compilado el ejercicio anterior, tendremos un archivo BORRAR2.HEX, arme el circuito en PROTEUS, con los siguientes elementos: LM016L, PIC16F628A y la referencia GROUND de Inter sheet Terminal. Luego proceda a cargar el archivo .hex, presione PLAY y el texto empezará a aparecer. Note que no es necesario alimentar el LCD, sólamente la referencia a tierra del bit R/W debe colocarlo a un nivel 0L. el bit RS fué cambiado al pin RB.1, ya que el pin RA.4 no funciona como en la práctica real.



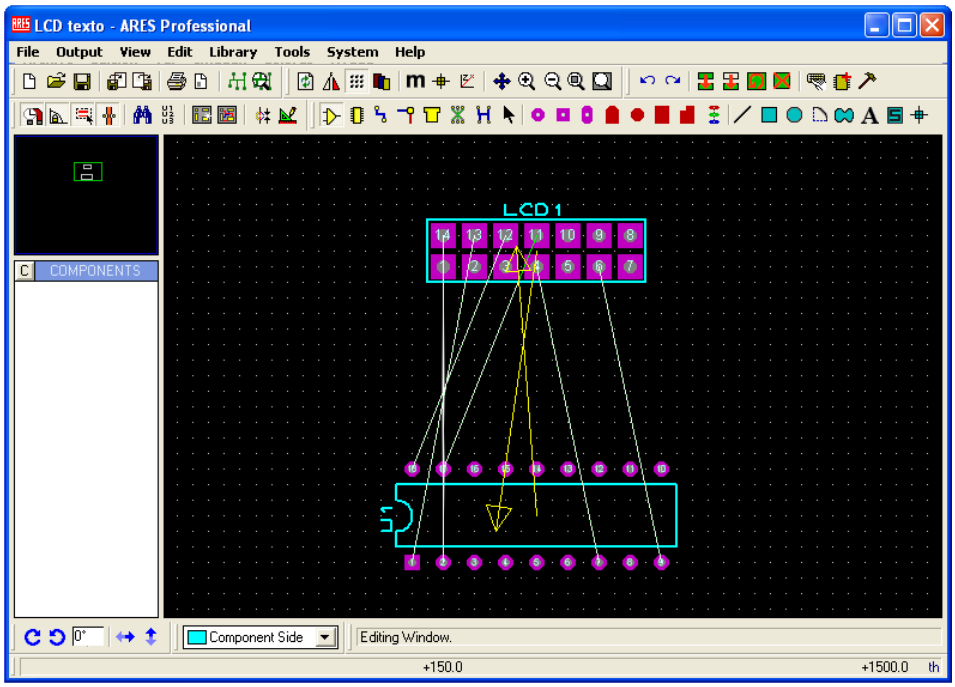
6.3. GENERACIÓN DE PCB (Print Circuit Board).

Esta herramienta es muy indispensable si desea fabricar un circuito impreso, para ello en la misma ventana de la simulación del LCD, presione ARES , se presentará una pantalla similar a la siguiente figura.

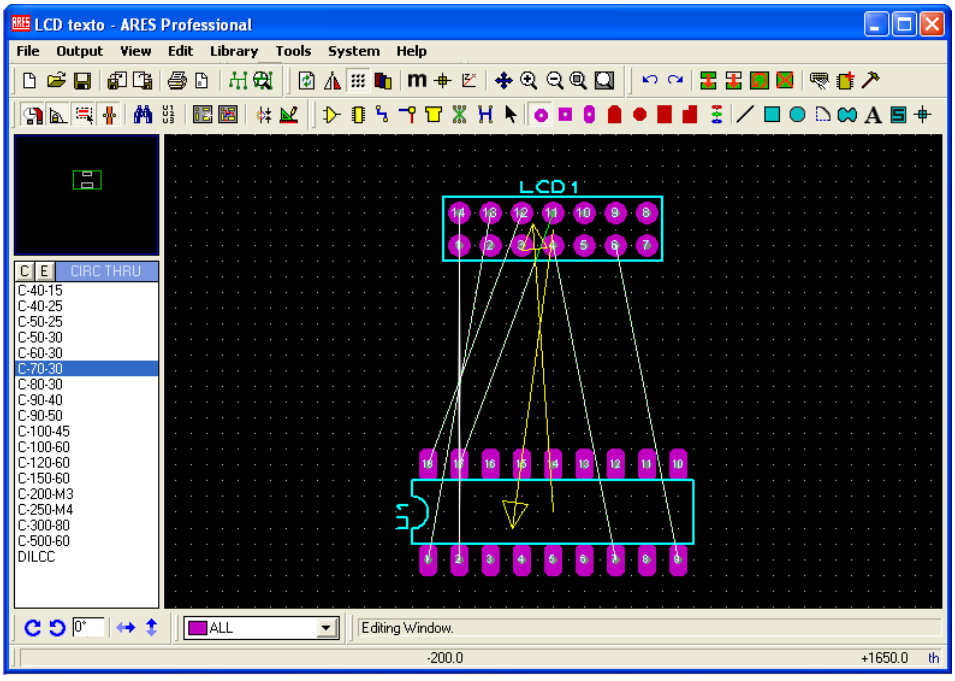


Nota: para poder rutear fíjese que los elementos a rutear existan en la librería, por ejemplo en la pág. 180 note que el PIC16F628A si tiene elemento para PCB, de no ser así saldrá un mensaje de **No PCB Package**. Tal es el caso del LED-RED, el cual no dispone de PCB Package.

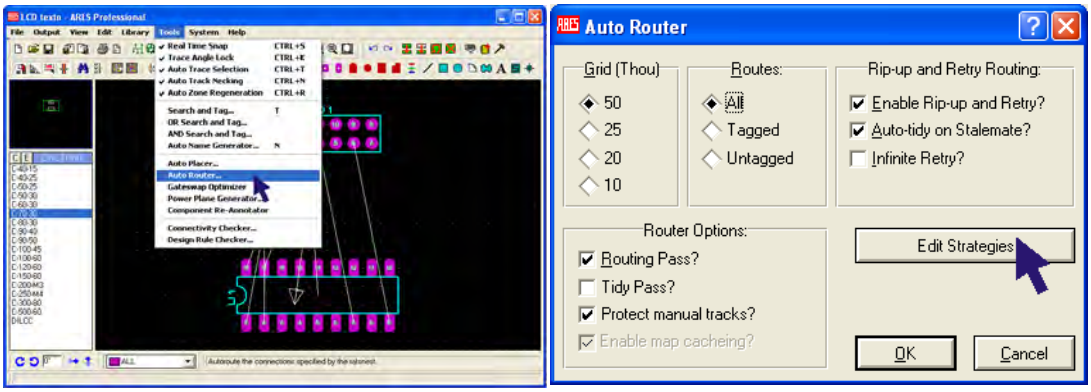
En esta pantalla arrastre los 2 elementos, el LCD y el PIC hacia la pantalla, observará que están unidos por líneas verdes.



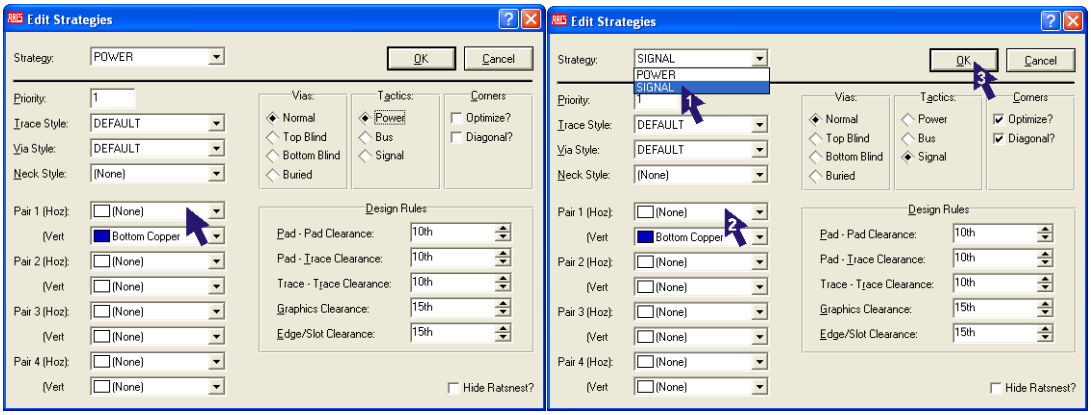
Si desea puede cambiar la forma de las islas, para ello de un clic en  , y ajuste los diámetros que más le convenga.



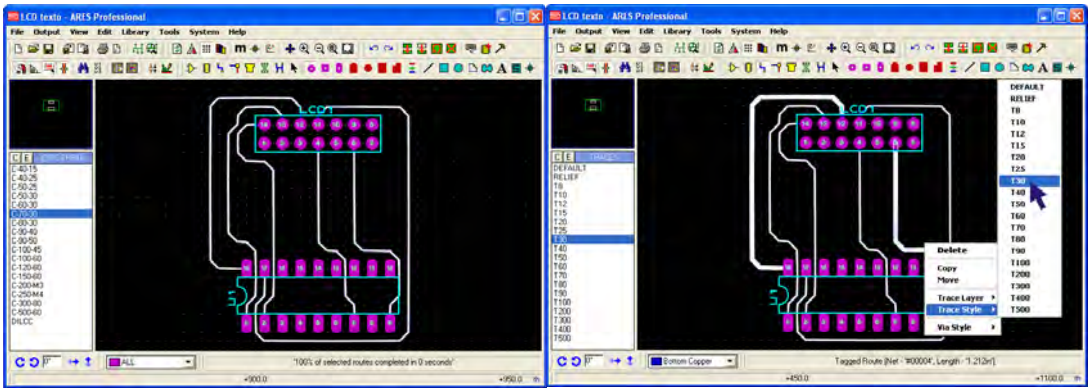
Ahora proceda a rutear, para ello abra la pestaña **Tools** y de un clic en **Auto Router** … Si no le importa rutear en ambos lados sólo de clic en **OK**, pero si necesita que rutee en un sólo lado de un clic en **Edit Strategies**.



Estos son los pasos para rutear en un sólo lado, primero coloque (None) en Pair 1 (Hoz), luego en Strategy cambie de **POWER** a **SIGNAL**, coloque también en (None) en Pair 1 (Hoz), finalmente de un clic en OK y luego otro clic en Ok de la ventana anterior (Auto Router).

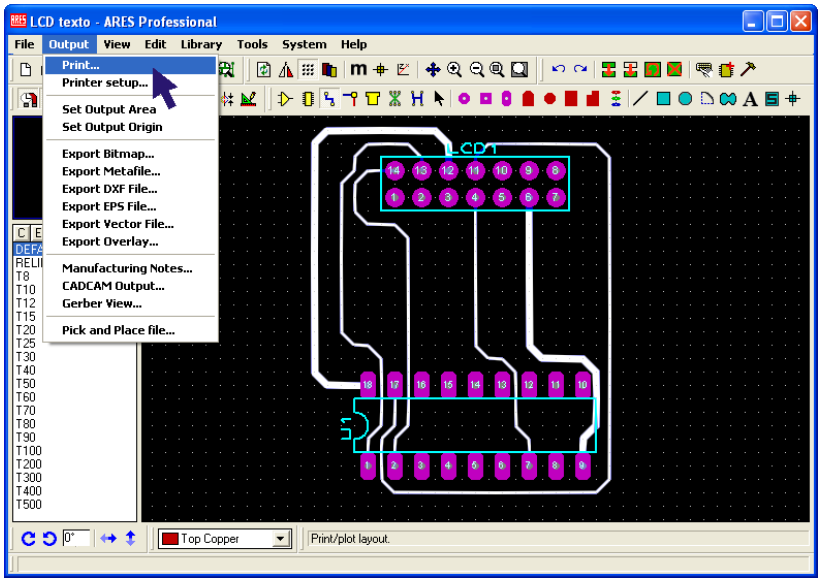


Ahora tendrá el circuito ruteado en un sólo lado, aquí puede ajustar el grosor de las pistas si lo desea, para ello de un clic en Track Placement and editing , luego de dos clic derechos sobre la pista que desee ajustar y en **Trace Style** coloque T30, notará que aumenta el grosor de la pista.

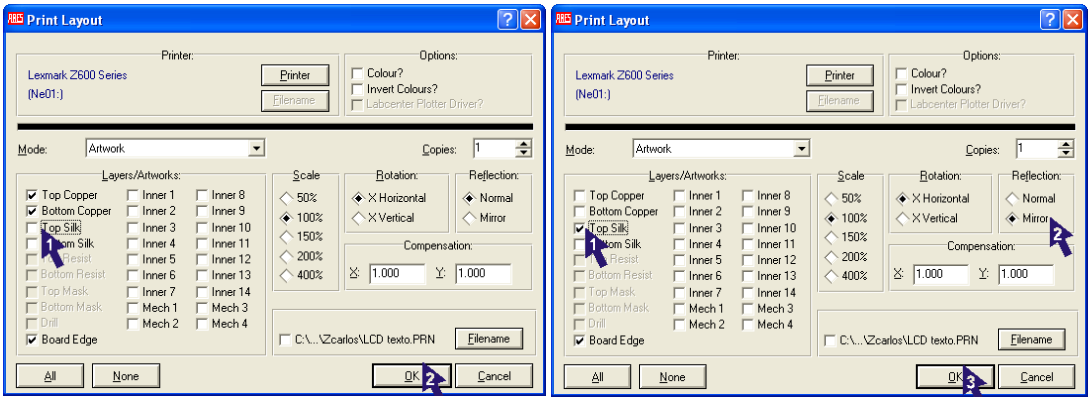


6.4. IMPRESIÓN DEL PCB (Tarjeta de circuito impreso).

Una vez realizado todos los ajustes puede imprimirlo, para ello abra la pestaña **Output** y de un clic en **Print**…



Para imprimir sólo las pistas configure como la siguiente figura izquierda, es decir desactive **Top Silk** y luego de un clic en **OK**. Para imprimir el screen de elementos, tome en cuenta que este debe estar espejeado y sin las pistas, es decir selecionado **Top Silk** y **Mirror** (ver figura derecha).



CAPÍTULO 7
MÉTODO DE FABRICACIÓN DE CIRCUITOS IMPRESOS

Una vez que usted ha probado su proyecto, pueda que le interese hacer una placa de circuito impreso como las que hemos visto a lo largo del capítulo 5, y talvez un chasís para el mismo, en este capítulo aprenderá trucos para dar una buena presentación a su proyecto, se propone un método revolucionario y muy sencillo comparado con los métodos de dibujo con marcador para circuito impreso, revelado y serigrafía, este es la *transferencia térmica*.

Lo primero que se recomienda es tener todos los elementos electrónicos ya comprados y listos, para no tener el inconveniente de que una vez hecho la placa no encuentren el elemento correcto para soldarlo.

Segundo verifique en un protoboard si el proyecto funciona correctamente con todos los elementos que van a soldar en la placa.

Tercero trate de ver los pines que más se le acomoden a las pistas, que no se crucen mucho, que además estén lo más cerca posible al periférico a manejar.

7.1 DISEÑO DEL CIRCUITO IMPRESO POR SOFTWARE.

Lo primero que se necesita para hacer una placa de circuito impreso es un dibujo de las pistas para los elementos, esto se consigue con la ayuda de un computador, y de los softwares CAD como PROTEL, PROTEUS, EAGLE, ORCAD, Corel DRAW, o cualquier software de dibujo en el que usted pueda trazar las líneas y pads del circuito (Paint, Photo SHOP, photo Express, etc.), a continuación el circuito de las pistas y el screen de elementos, ya realizados en un software:

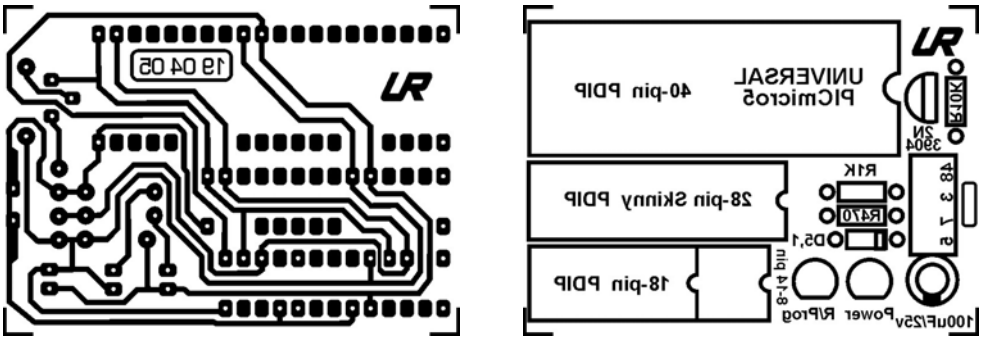


Figura 7.1.1. *Diagrama de pistas y screen de elementos listos para la impresión láser correspondiente al grabador UNIVERSAL Picmicro5.*

Noten que el dibujo del lado derecho de la figura 7.1.1 está realizado un espejo, esto lo necesitamos por la transferencia térmica, en cuanto al tamaño de la placa deben considerar el chasís en donde van a colocar esta placa, como también donde deben ir los agujeros para los tornillos.

7.2 IMPRESIÓN DE LAS PISTAS Y SCREEN DE LOS ELEMENTOS.

Una vez que se tiene el diseño de la placa, se debe imprimirlo con una impresora láser o copiadora (que tengan los cartuchos toner de polvo en color negro), en un papel de transferencia térmica **Press-n-Peel** (o papel de transferencia PCB), que lo podemos conseguir en las tiendas electrónicas a un costo de más o menos 2 USD cada hoja.

Otra alternativa, la que aquí se utiliza, y además se incluye en este libro es el **PAPEL FOTOGRÁFICO** tipo GLOSSY, de la empresa APLI, para el cual se facilitan los datos:

APLI	Glossy Bright Paper	Ref. 04452	de 170g.	10 Und.
APLI	Glossy Bright Paper	Ref. 04135	de 170g.	50 Und.

Este papel fotográfico para impresoras de (inyección de tinta), vienen en cajas de 10 Und. o 50 Und. (Ref. 04135). El costo de la caja de 10 Und. es de 10 USD, y en algunas papelerías venden por unidades. Si no encuentran exactamente la misma hoja, pueden utilizar la hoja APLI de 125 g. de referencia 04451 de 10 Und., o la de ref. 04134 de 50 Und., la única desventaja que tienen las hojas de 125 g. es que el papel se rompe al tratar de separar de la placa, pero sin embargo no es un problema se lo puede remojar toda la placa y así desprenderlo con la yema de los dedos.

No olvide que la impresión debe realizarse en una **IMPRESORA LÁSER** o fotocopiadora y no por una impresora de inyección de tinta. Aquí está el truco, el papel fotográfico que hemos hablado anteriormente, sirve para imprimir fotografías específicamente en impresoras de inyección de tinta, al imprimirlo en una impresora láser, se da un efecto químico al unirse el toner de la impresora con la capa de barniz que tienen estas hojas, (lo mismo sucede con el papel Press-n-Peel), el hecho es que si se imprime con la impresora de inyección a tinta estas hojas ya no sirven.

Vamos a suponer que tenemos listo nuestra hoja con dicha impresión, el siguiente paso es aplicarle calor por el lado revés de las hojas y sobre las placas, para lo cual utilizaremos *la hoja de transferencia que viene con este libro* (UNIVERSAL PICmicro5), el calor de la plancha hace que el toner se derrita, y junto al barniz de la hoja se pegan en la lámina de cobre, para esto se irá explicando paso a paso todos los procedimientos necesarios, así como también pueden ver la secuencia fotográfica sobre cómo hacer placas PCB que contiene el CD en FabricaciónPCB\ 1visualizar.exe.

7.3 PREPARACIÓN DE LA PLACA (BAQUELITA O FIBRA DE VIDRIO).

MATERIALES.
-1 placa de Baquelita o Fibra de vidrio, de una o dos caras de cobre
-1 lija de metal Nro. 150
-1 esponja de acero, de las utilizadas para lavar platos (lustre, estrella, etc.) que sea fina.

Primero se utilizará la lámina que contiene el lado de las pistas, la otra parte es decir el screen de elementos lo utilizaremos posteriormente.

Bien ahora se debe cortar la placa que puede ser de *Baquelita* o *de Fibra de vidrio*, esta última es la más recomendable, ya que el acabado final es mucho mejor, además es más aislante y resistente a la humedad. Para las medidas del corte, se debe considerar 4 mm adicionales a cada lado de la placa en relación al del dibujo que se vaya a transferir, por lo que la medida de corte para nuestro grabador sería (69mm x 50mm). Utilizando una sierra o una caladora con sierra de metal (debido a que este posee dientes más finos), se debe cortar la placa necesaria para el grabador de microcontroladores.

Figura 7.3.1. Paso 1. *cortar la placa de 69 x 50 mm con una sierra o una caladora, esta última les permite cortes más rápidos y perfectos.*

Como pueden observar para utilizar la caladora esta debe estar sujetada en una tabla y con una guía de aluminio, en la que se regula con unos tornillos la distancia de corte, consiguiendo igualdad en el tamaño de las placas para producciones en serie.

Una vez cortado la placa, se debe limpiar las limallas de cobre que quedan en los filos de la placa, con una lija fina de metal (Nro. 150). Luego de esto se debe limpiar el lado del cobre donde se va a transferir las pistas con una esponja de acero, notarán que la lámina de cobre cambia de color, esto porque se está limpiando el óxido creado en la superficie y los rayones que pudiera tener.

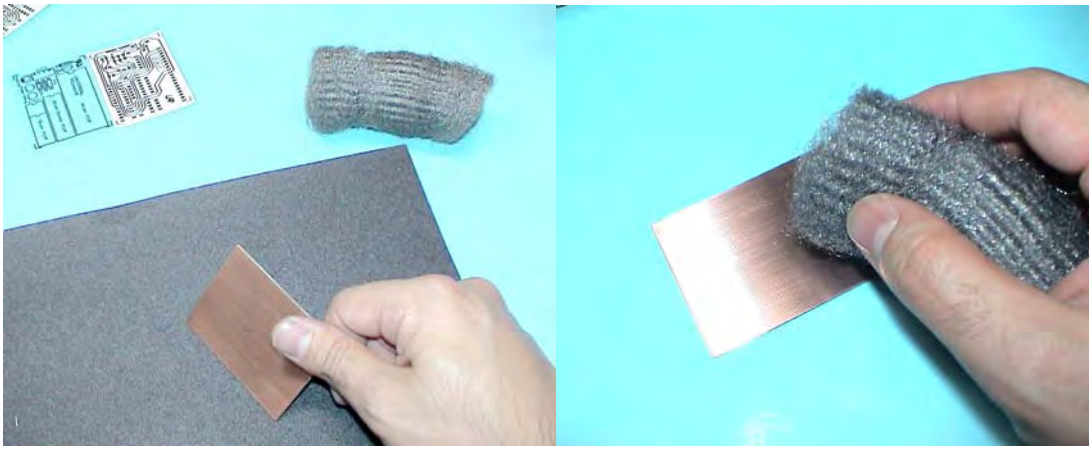


Figura 7.3.2. Paso 2. *limpiar los filos del corte realizado y la lámina de cobre oxidada.*

Una recomendación muy importante, es que una vez limpia la placa, no se debe tocar con los dedos sobre la lámina de cobre, pues la grasa de los dedos genera óxido casi inmediatamente, si lo desea puede lavarlo posteriormente con crema lavaplatos del tipo arranca grasa.

Para el caso de querer guardar las placas se recomienda introducirla en una funda y cerrarlo con cinta adhesiva para así evitar el contacto directo con el aire.

7.4 TRANSFERENCIA TÉRMICA DEL PAPEL HACIA LA LÁMINA DE COBRE.

MATERIALES.
-1 plancha domestica o una estampadora
-2 pedazos de tela de calentador o franela
-La placa íntegramente limpia. (limpiar con esponja de acero)
-El papel con el diseño a transferir (Press-n-Peel o papel fotográfico Glossy)

Ahora vamos a realizar la transferencia propiamente dicha de las pistas sobre el lado de la lámina de cobre, recuerde esta debe estar completamente limpia y no debe ser tocada con los dedos, para manipularlo se debe sujetar por los bordes. Primero coloque el papel fotográfico con el lado de la tinta sobre el lado del cobre, sin moverlo mucho introdúzcalo debajo de la tela, todo esto sobre una mesa rígida y luego pase la plancha que debe estar al máximo de la temperatura, aplique presión con todo el peso del cuerpo por alrededor de 20 a 30 segundos, luego de esto retire e inmediatamente coloque la placa en otra parte de la mesa que se encuentre fría, con otro trapo

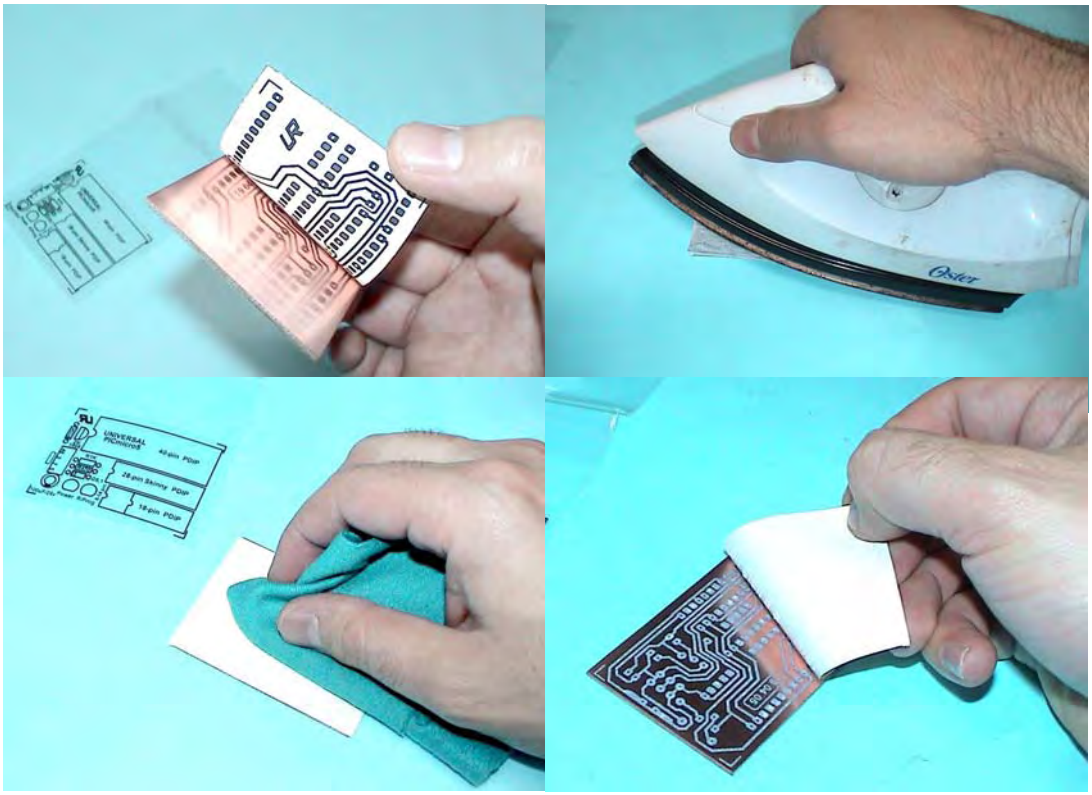


Figura 7.4.1. Paso 3. *Aplique presión con una plancha bien caliente por 30 seg. y déjelo enfriar presionando con otro trapo para luego retirar el papel cuidadosamente.*

aplique presión uniforme frotándolo de un lado a otro hasta que este se enfríe, con la finalidad de que toda la tinta (toner + barniz) se pegue a la lámina de cobre y así poder retirar el papel sin que se presente partes cortadas o faltantes, si permanecen residuos de papel remójelo y sáquelo con la yema de los dedos.

Si las pistas no se pegan puede ser por que la plancha no es suficientemente caliente, en este caso utilice otra plancha, debe notar además que el papel se amarilla un poco por efecto del calor, otra razón puede ser también que esté utilizando una tela muy gruesa, cambie a otro tipo de tela, y por último puede ser la poca presión aplicada con la plancha, se debe prácticamente apoyarse sobre la plancha y frotarlo sobre toda la superficie de la placa.

Una solución muy eficaz es utilizar las planchas estampadoras o fusionadoras, estas son utilizadas para estampar camisetas o forros pegables, tienen un lado de caucho resistente al calor y su presión es muy alta, así como la temperatura que puede entregar es de hasta 500°C, (nosotros utilizaremos de 300 a 400°C), estas planchas tienen una superficie de 40 x 40 cm, ideal para placas de 30 x 20 cm, ya que la plancha doméstica sólo sirve para placas de hasta 10 x 20 cm.

7.5 PROCESO DE ATACADO (REDUCCIÓN) DEL COBRE.

MATERIALES.
-1 recipiente de plástico, (no metálico)
-½ vaso con agua tibia
-1 palillo de pincho o una pinza de plástico
-1 funda de cloruro férrico (en polvo)
-1 calentador de agua para peceras (opcional)

Para reducir el cobre sobrante, es decir el que no está protegido por la tinta y el barniz, necesitamos preparar un atacador, existen 2 tipos de atacadores: los rápidos y los lentos, los rápidos como por ejemplo la combinación de 50 ml de ácido clorhídrico y 50 ml de agua oxigenada, pueden reducir el cobre no protegido al cabo de unos pocos segundos, pero tiene la desventaja de ser difíciles de conseguir en el mercado, los lentos en cambio como el Cloruro férrico se lo encuentra en cualquier tienda electrónica pero el proceso de atacado podría tomar hasta 1 hora. Sin embargo por ser menos agresivo y porque no emana muchos gases tóxicos, utilizaremos el cloruro férrico.

Figura 7.5.1. *Materiales a utilizar, a la derecha mezcla del cloruro férrico con el agua.*

Para su preparación primero se recomienda un lugar con buena ventilación, no utilizar reloj con pulseras de metal, pues el cloruro férrico ataca a los metales, también tome en cuenta que al contacto con la ropa o la piel, produce una mancha amarillenta, por lo que debe tomar las precauciones necesarias, y en caso de darse contacto con la piel debe lavarse con agua y jabón.

Una vez que estemos listos procedemos a preparar la solución ácida, primero colocamos el ½ vaso de agua tibia en el recipiente de plástico, luego colocamos poco a poco si es posible con una cuchara de plástico, todo el contenido de la funda de cloruro férrico, es normal que el agua se empiece a calentar (debido a la reacción química), el palillo de pincho lo utilizaremos para revolver el líquido y así ayudar a disolver el cloruro férrico.

PELIGRO: Nunca ponga todo el cloruro férrico de una sola vez sobre el agua, la reacción muy brusca podría hacer explotar y producir quemaduras en la ropa y en la piel.

Esta solución ya preparada, puede ser almacenada en un envase de plástico o vidrio para luego ser utilizada varias veces, hasta que el ácido se contamine tanto que ya no sea posible corroer placas (debido a que el efecto es cada vez más lento), para entonces se debe desechar.

Figura 7.5.2. *Introduzca la placa virgen en la solución ácida, si el ácido ya ha sido utilizado varias veces y tiene poca acción, introduzca el calentador para peceras.*

Procedemos a introducir la placa del grabador de PIC´S en la solución ya preparada, el tiempo de corrosión por ser la primera vez, puede variar entre 15 y 30 minutos, por lo que debe revisar de vez en cuando si el cobre no protegido ha sido eliminado utilizando el palo de pincho. También podemos ayudar a que el proceso sea más rápido, moviendo el agua de un lado a otro, esto permite que el cobre disuelto, por efecto de la corriente generada por el movimiento, se deposite en el fondo del envase, así permitimos que la lámina tenga contacto directo con el ácido.

En ocasiones especiales cuando por ejemplo la solución ya ha sido utilizada varias veces, el proceso de corrosión es mucho más lento, el calor y el movimiento pueden ayudar a reaccionar al ácido, para esto necesitamos un envase vertical, introducimos en el fondo una generador de oxígeno, del utilizado en las peceras, también introducimos un calentador de peceras de 2 Gls., el ascenso de las burbujas genera movimiento en el agua, que junto al calor generado por el calentador de pecera, ayudan a corroer más rápido el cobre de las placas, en este caso la placa se coloca verticalmente, permitiendo que el cobre disuelto caiga rápidamente al fondo del envase.

7.6 PROCESO DE LIMPIEZA DE LA PLACA YA ATACADA CON ÁCIDO.

Una vez que el ácido terminó de eliminar el cobre expuesto, retiramos la placa del ácido y lo lavamos con abundante agua del grifo, las pistas, pads, etc., en esta fase se ven de color negro, esto se debe a que el papel se encuentra remojado, pero cuando se seca vuelve a ser de color blanco, ahora nos resta limpiar todo el residuo de papel y tinta de 2 maneras posibles, la primera es utilizando thinner, acetona, o cualquier disolvente, pero esto tiene un efecto secundario si bien limpia las pistas, un poco de tinta negra se impregna en algunas partes de la placa, dando la apariencia de sucia, por tal razón es mejor limpiar con la misma esponja de acero y un poco de agua, el resultado de la limpieza no deja rastro de tinta y se ve muy nítido.

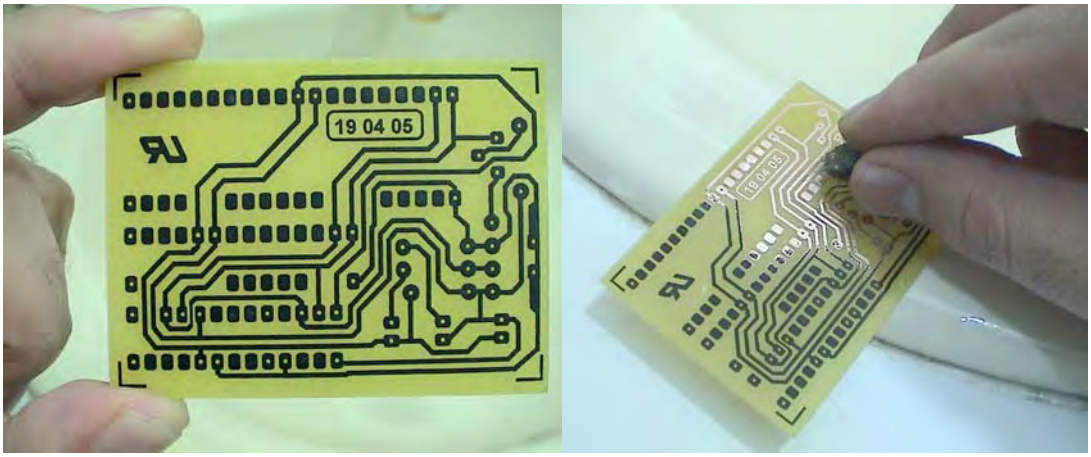


Figura 7.6.1. *Placa recién sacada del ácido, y limpieza de la tinta con esponja de acero y agua.*

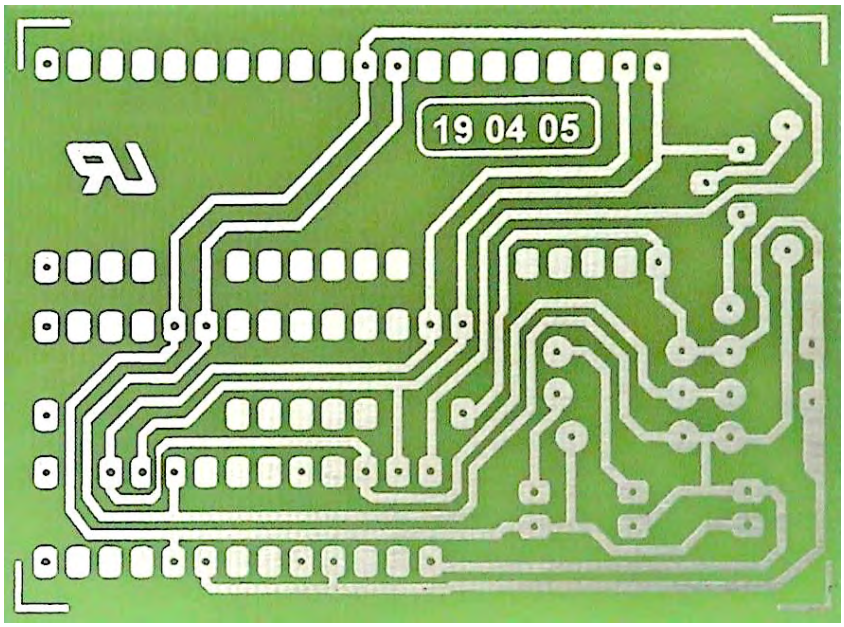


Figura 7.6.2. *Apariencia que debe tener la placa ya atacada y libre de tinta (toner + barniz).*

7.7 TRANSFERENCIA TÉRMICA DEL SCREEN DE LOS ELEMENTOS.

El screen de elementos, no es nada más que textos, información, datos y figuras que indican el lugar donde se debe insertar los elementos electrónicos, son muy útiles ya que evitan que la persona se equivoque de lugar al insertar una resistencia, o coloque al revés un diodo, el material utilizado es un acetato para copiadoras de la marca APLI referencia 859, también se puede utilizar acetatos para impresoras ink-jet de cualquier marca, este se debe colocar a la misma altura y posición que están las pistas y también se debe tener cuidado de no colocar al revés (recuerde que la impresión de este acetato es espejeado). Para que se adhiera bien, en el caso de placas de baquelita, se debe lijar bien el lado donde se va a colocar el screen, ya que tienen una capa de laca o barniz, para el caso de placas de doble cara, al disolverse la cara que no se necesitaba, esta queda bien limpia y porosa, por lo que no hace falta limpiarle.

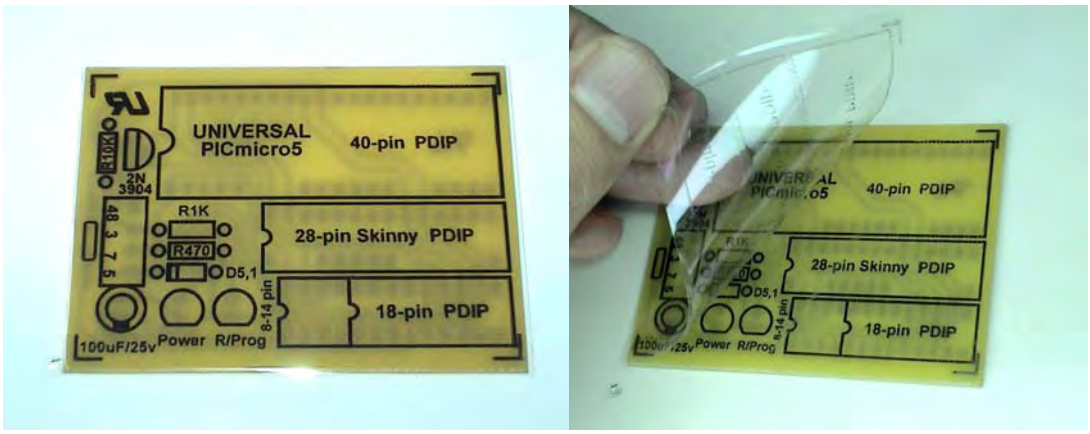


Figura 7.7.1. *Coloque el acetato y proceda a termofijarlo, luego retire el acetato.*

Cuando termine de termofijarlo, igualmente proceda a enfriarlo haciendo presión con un trapo, una vez que esté frío retire cuidadosamente el acetato, tendrá un acabado nítido ya que toda la tinta del acetato debe transferirse a la placa, es importante que reconozca cual acetato tiene mejor resultado, para ello vea el acetato que quede con menor cantidad de tinta de residuo.

Para el caso de hacer el screen con papel Press-n-peel, proceda de igual manera, la diferencia es que debido a una fina capa de material fílmico que posee este acetato el screen sale de color azul.

7.8 PERFORACIÓN DE LA PLACA.

MATERIALES.
-1 broca para metal, de 1mm de diámetro
-1 taladro o moto-tool (taladro miniatura)
-1 pedestal para taladro (opcional).

Lo único que hace falta para que la placa esté lista es realizar los respectivos agujeros, para el caso de nuestro grabador de PIC´S necesitamos hacer 51 agujeros con broca de 1 mm. lo ideal es disponer de un moto-tool o taladro miniatura, ya que estos son de fácil manipulación, además pueden soportar brocas de 0,3 mm en adelante, para este caso se debe hacer una hendidura con un

punzón y un martillo en cada lugar donde se va hacer un agujero, con la finalidad de que sirvan de guía para la broca.

Otra opción es utilizar un taladro cuyo mandril pueda soportar brocas desde 0,5 mm hasta 10 mm, y si dispone de un pedestal, en este caso no necesita hacer las hendiduras con punzón, ya que la perforación se realiza completamente perpendicular a la placa y sin que se desvíen las brocas.

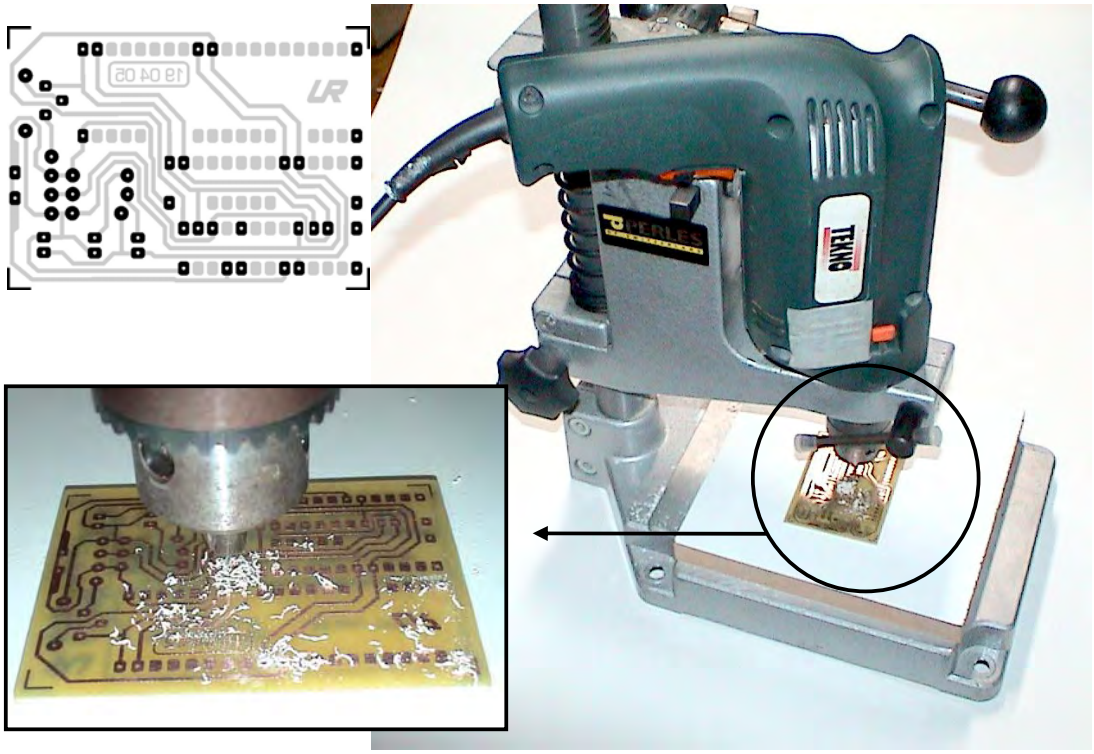


Figura 7.8.1. *Pedestal y taladro con broca de 1 mm. para perforar la placa PCB.*

7.9 SOLDADURA DE ELEMENTOS.

Los materiales que necesita para elaborar el grabador de PIC´S, son los siguientes:

MATERIALES.
a) Un metro de cable de 4 hilos multifilar
b) Un conector DB9 hembra con su respectivo cajetín
c) Dos leds de 5mm, un rojo y un verde
d) Un diodo zener de 5.1 V. a ½ o 1 vatio
e) Una resistencia de 470 Ω a ¼ de vatio amarillo-violeta-café
f) Una resistencia de 1 K Ω a ¼ de vatio café-negro-rojo
g) Una resistencia de 10 K Ω a ¼ de vatio café-negro-naranja
h) Un capacitor electrolítico de 100 uF/25V.
i) Un transistor NPN 2N3904
j) cuatro zócalos, (8,18, 28 y 40 pines).

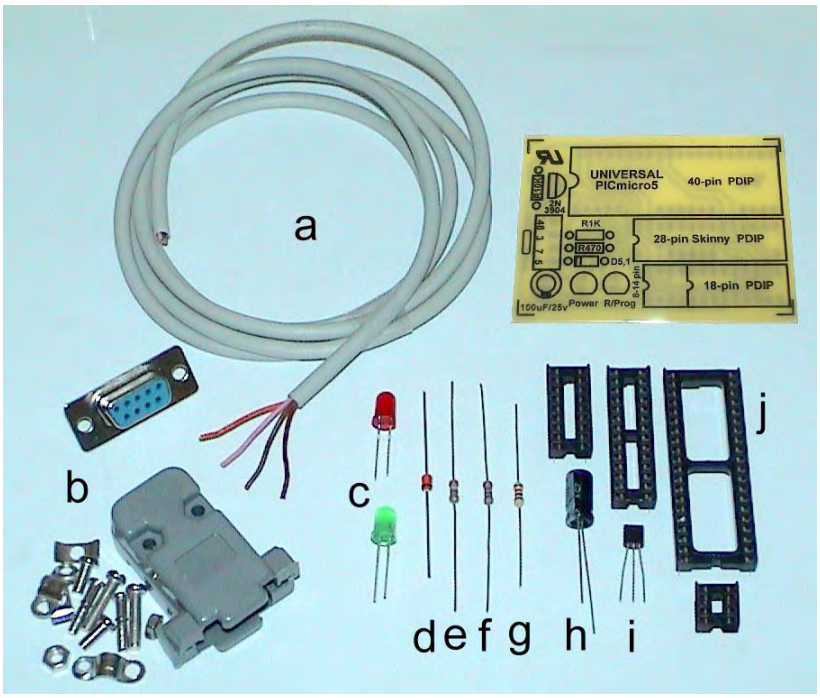


Figura 7.9.1. *Materiales para la fabricación del grabador de PIC´S* **UNIVERSAL PICmicro5**.

Las herramientas y materiales que todo soldador electrónico debe tener son los siguientes:

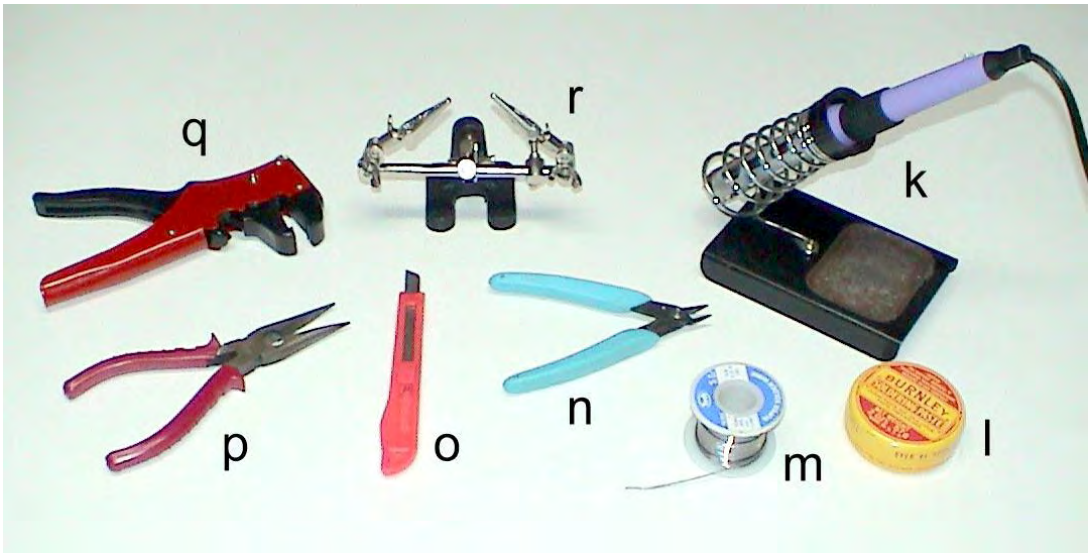


Figura 7.9.2. *Herramientas y materiales que se debe disponer para procesos de suelda*:
PRINCIPALES : *k) cautín tipo lápiz de 20W a 30W con su respectivo soporte, l) pasta de soldar, m) alambre de suelda de 1 mm de diámetro de estaño(60%) y plomo(40%), con centro de resina n) pinza de corte, o) estilete o bisturí, p) alicate de punta.*
OPCIONALES: q*) pinza pelacables, r) sujetador de placas o una entenalla pequeña.*

Una vez que tenga listos los materiales y herramientas, empiece por los zócalos, estos debe prepararlos sacando algunos pines que no se necesita, con el alicate de punta empuje uno por uno los pines no necesarios como muestra la siguiente figura:

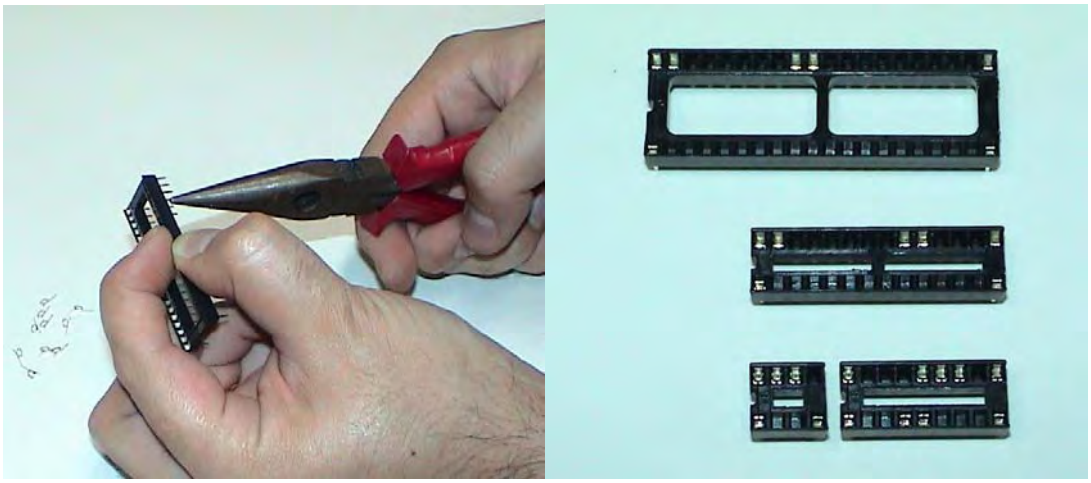


Figura 7.9.3. *Retire los pines que no se necesitan, empujando uno por uno con la pinza de punta, hasta que queden como el de la fotografía derecha.*

Las resistencias y el diodo, debe doblarlos, a la medida necesaria y con la ayuda de la pinza de punta:

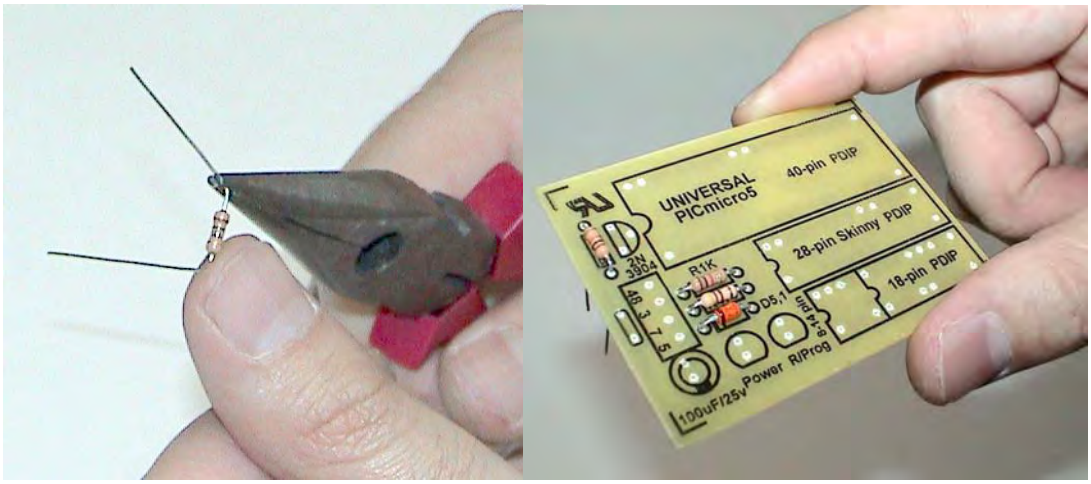


Figura 7.9.4. *Doblar las resistencias y el diodo, de acuerdo a la distancia de las perforaciones, en el momendo de insertar el diodo zener se debe tomar en cuenta la polaridad.*

Se debe seguir una secuencia en la soldadura de los componentes, primero los elementos más bajos y luego los más altos como el capacitor, de esta manera el orden para ir soldando sería: resistencias, diodo, zócalos, transistor, leds, y por último el capacitor, luego de todo esto suelde el cable con las indicaciones posteriores.

Inicie con la suelda de las resistencias y el diodo, estos debe insertarlos y luego doblar las patitas hacia el exterior, con la finalidad de que al dar la vuelta la placa para soldar, estos no se caigan, luego de esto coloque en el soporte para placas y proceda a soldar, el mejor método de suelda, es calentar un poco el elemento a soldar y luego poner el estaño, mover la punta del cautín de arriba abajo, tocando el alambre de suelda y el elemento, esto permite una rápida adherencia y una buena soldadura.

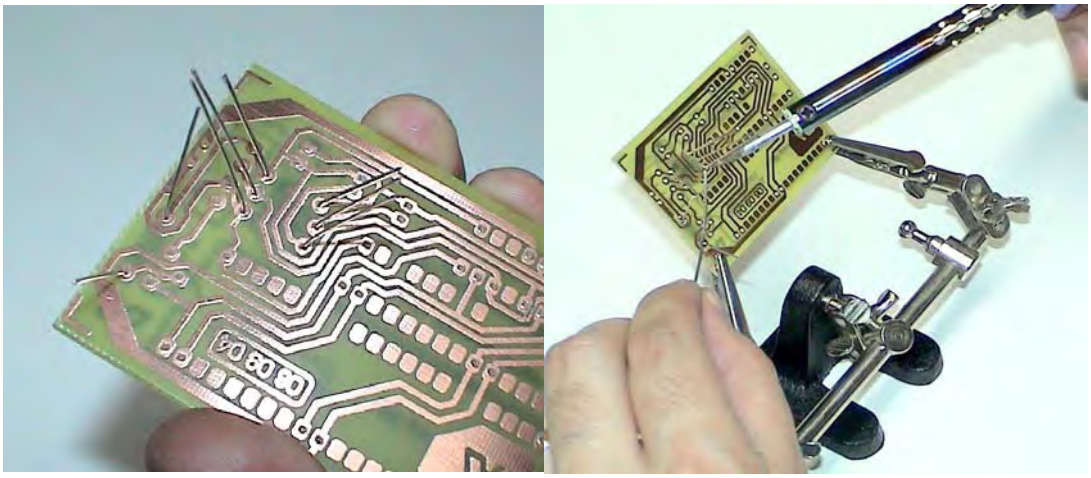


Figura 7.9.5. *Doblar las patitas de las resistencias hacia el exterior, sujetarlo en el sujeta placas y proceder a soldar.*

Si usted no dispone de un soporte para placas, puede soldar de la siguiente manera: con la uña de su dedo índice sujete la resistencia. El alambre de suelda colóquelo al filo de una mesa y con la otra mano manipule el cautín, como lo muestra las siguientes fotografías:

Figura 7.9.6. *Sujete la resistencia con la uña, coloque el alambre de soldar al filo de una mesa y sin soltarlo empiece a soldar.*

Los leds deben ser colocados de la siguiente manera, donde dice Power va el led rojo y donde dice R/Prog, va el led verde, coloque correctamente la polaridad de los leds, asimismo asegúrese de colocar el capacitor de acuerdo a la polaridad que le indica el screen. Una vez que ha terminado de soldar los elementos, la placa debe tener la siguiente apariencia:

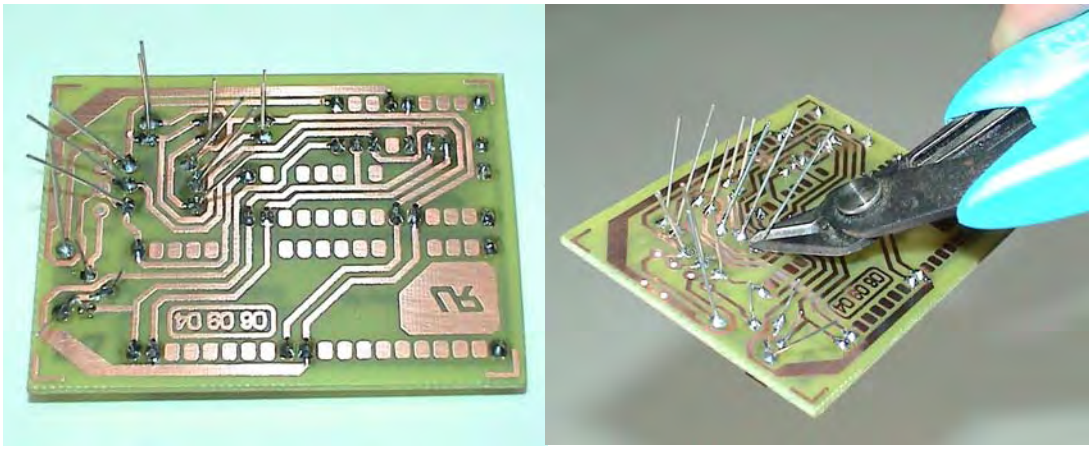


Figura 7.9.7. *Una vez soldado los elementos, corte los alambres sobrantes.*

Ahora con la pinza de corte, corte todos los alambres que sobresalen de la parte posterior, no olvide guardar el alambre que corte del diodo zener, este le servirá luego para sujetar el cable de 4 hilos.

El cable de conexión al PC lo preparamos de la siguiente manera: primero retire la envoltura plástica del alambre, más o menos unos 2 cm, esto con la ayuda del estilete si lo prefiere.

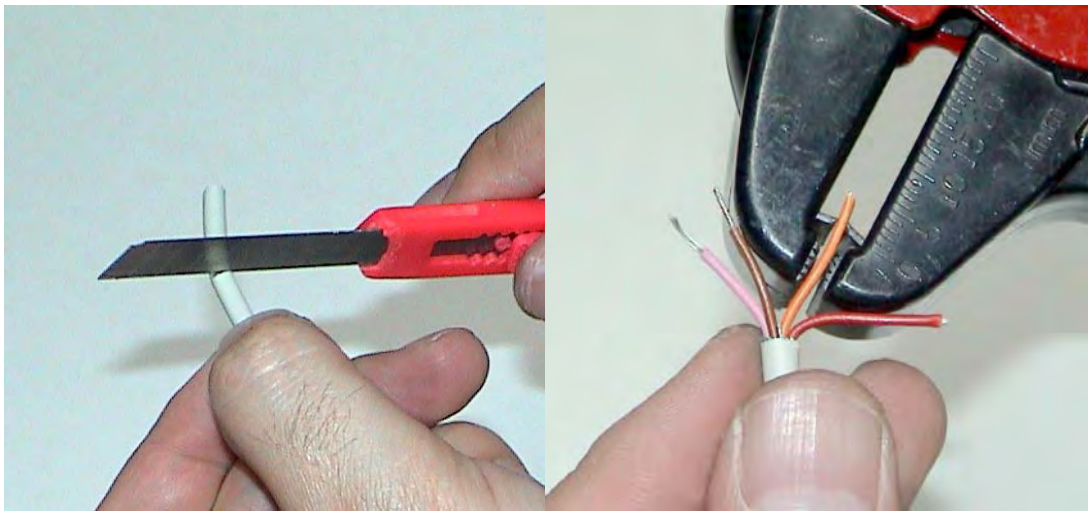


Figura 7.9.8. *Pele los alambres con la ayuda de un estilete o un pelacables, luego debe torcer los filamentos para que permanezcan uniforme.*

Una vez pelado el cable, introduzca las puntas de los alambres en la pasta de soldar, y con la ayuda del cautín estáñelo todos los alambres como muestra las siguientes fotografías:

Figura 7.9.9. *Introduzca las puntas del alambre en la pasta de soldar y luego estáñelo con el cautín previamente cargado de suelda.*

Con las puntas de los alambres estañados, los filamentos permanecen juntos, ahora introduzca en las perforaciones de la placa y sosténgalo con sus dedos hasta que logre soldarlos.

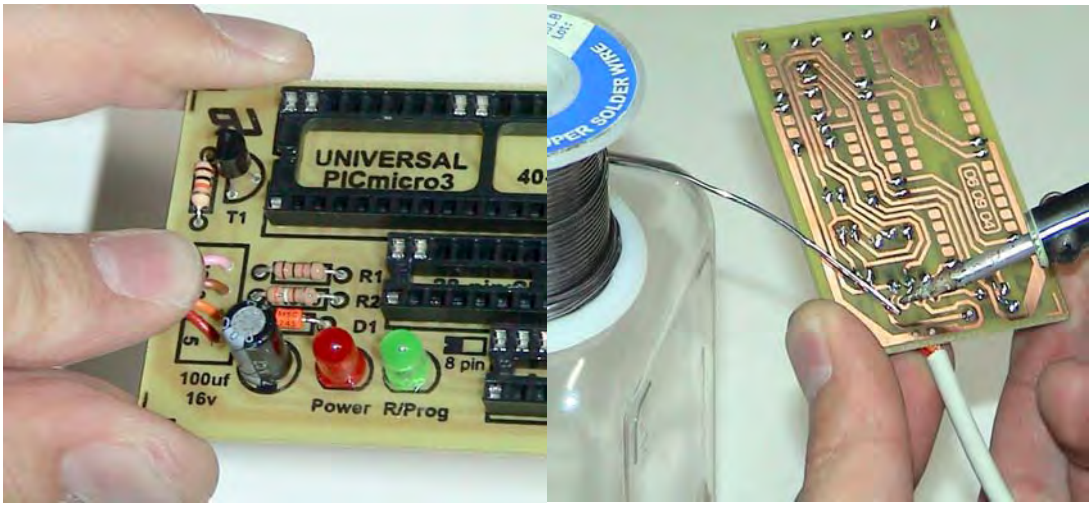


Figura 7.9.10. *Introduzca las 4 puntas de los alambres y sosténgalo con su dedo, luego proceda a soldar como se aprendió en los casos anteriores.*

Una vez soldado los alambres procedamos a sujetarlo a la placa, para que el movimiento no los rompa, para esto necesitamos empujar el alambre del otro extremo hasta que la envoltura recorra

por lo menos 1 cm, lo suficiente para poder ser remordido con el alambre que se debe haber guardado al cortar el diodo zener, este debe doblarse formando un arco y debe atrapar al cable con todo su envoltura (ver Fig. 7.9.11 y 7.9.15.), luego de doblarlo suéldelo a la placa tratando de no calentar mucho al alambre, porque este podría derretir la envoltura del cable multifilar.

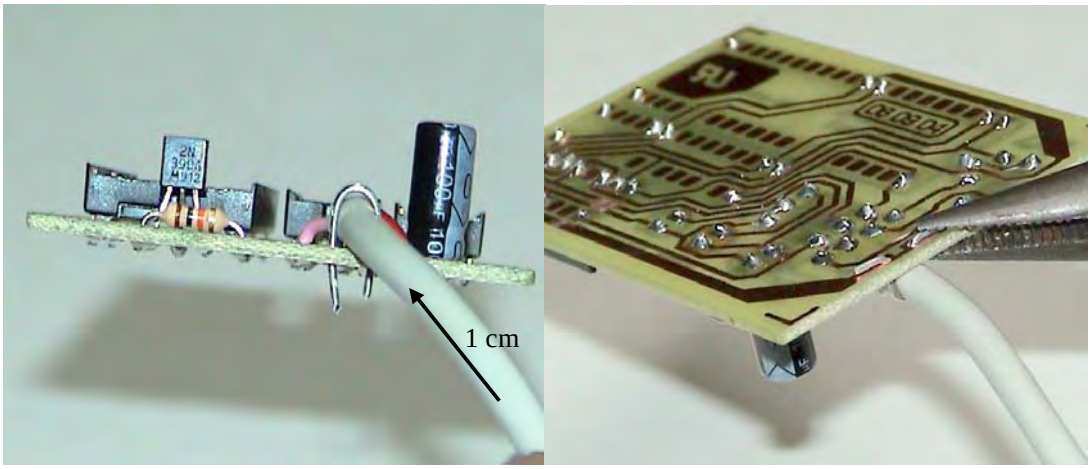


Figura 7.9.11. *Introduzca el alambre (residuo del diodo zener) en las 2 perforaciones, y con la ayuda del alicate de punta doble como ilustra la fotografía derecha.*

En el otro extremo del cable se debe soldar el conector DB9, que nos servirá para la conexión del puerto serial del computador, primero es necesario sujetarlo el conector en un lugar fijo, para tener ambas manos disponibles para la suelda, estañamos los terminales 3,4,5,7,y 8 y luego procedemos a soldar cada uno de los cables previamente estañados como se aprendió en la figura 7.9.9. Se debe tener en cuenta el lugar que le corresponde a cada cable, para esto debe fijarse como soldó en la placa y si por ejemplo el cable café está en el Nro. 3 pues deberá soldar en el pin 3 del DB9, el cable que soldó en el Nro. 4 8, debe soldarse con un puente al pin 4 y 8 (ver Figura 7.9.13).

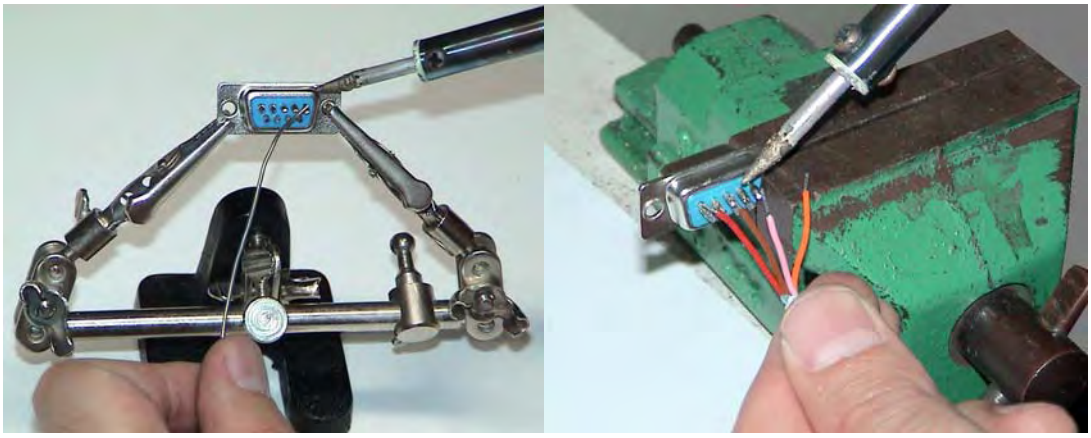


Figura 7.9.12. *Sujete el conector DB9 y estañe los terminales, si no dispone de un sujetador, podría servirle una entenalla pequeña, es importante que disponga de sus 2 manos para soldar.*

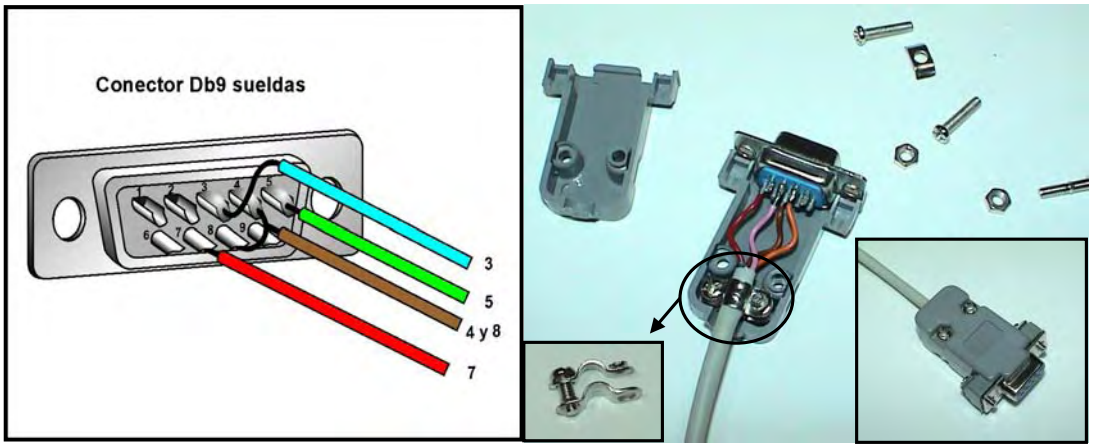


Figura 7.9.13. *Suelde cada cable en el lugar que le indica el screen de la placa UNIVERSAL PICmicro5, luego coloque el cajetín del conector DB9.*

El puente que une el pin 4 y 8 puede hacerlo con el mismo alambre, soldando primero el un pin y luego doblando hasta alcanzar el otro pin. Una vez colocado los cables dentro del cajetín, puede colocar un poco de silicón con la pistola térmica, esto lo ayudará a que no se rompa con la manipulación del conector.

Para limpiar los residuos de suelda (pasta) que se encuentra en las pistas de la placa, podemos utilizar un cepillo de dientes que ya no se utilice, introducimos las cerdas en un poco de thinner, y cepillamos la placa teniendo cuidado de que el thinner no se derrame por el lado posterior de la placa, es decir el screen ya que podría borrarlo.

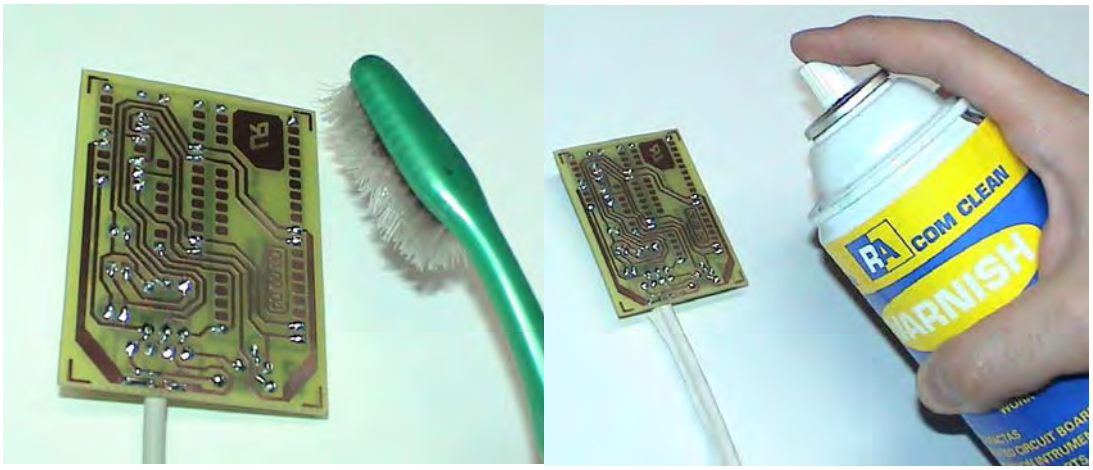


Figura 7.9.14. *Limpie la placa con mucho cuidado, ya que el thinner podría borrar el screen, déjelo secar y posteriormente puede darle una capa de laca o barniz.*

Para evitar que las pistas de cobre se oxiden, se debe dar una capa de barniz en spray para circuito impreso, este también lo utilizan para cubrir el rebobinaje de los motores, una marca conocida es RA com clean VARNISH, este le dará un acabado brillante transparente, pero si desea darle un

acabado más profesional, (ver CD:\Fotos libro\grabador PIC2), puede utilizar laca transparente con un poco de pintura verde, esto se debe hacer preparar en las tiendas que venden y preparan pintura automotriz, se debe llevar una placa de circuito impreso que tenga la máscara antisoldadura de color verde, se le pide que saquen el color de la placa mezclando laca transparente con pintura automotriz. Una vez que el color sea el correcto, y con la ayuda de un compresor y la pistola de pintar se procede a rociar el lado de las pistas, teniendo cuidado de no pintar el cable, para esto se debe cubrir con un poco de cinta adhesiva.

Figura 7.9.15. *Apariencia del grabador de PIC´S con la máscara antisoldadura de color verde.*

7.10 CHASÍS O CAJA PARA PROYECTOS.

En este subcapítulo se pretende dar ideas muy prácticas de cómo construir un chasís o caja para proyectos, primero debemos buscar en el mercado las cajas que podrían servirnos para contener un circuito, una de ellas es la caja DEXSON, que se utilizan para colocar tomacorrientes externos.

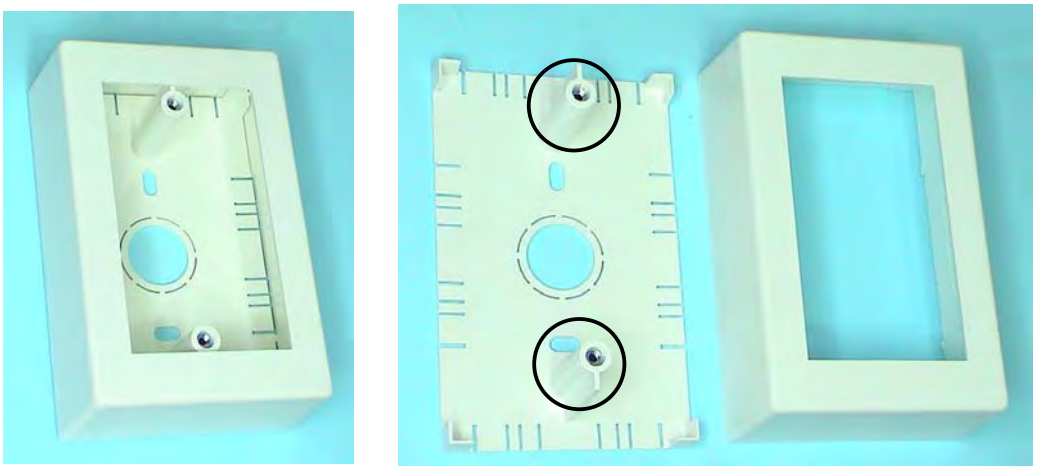


Figura 7.10.1. *La caja para tomacorrientes de la marca DEXSON, es ideal para proyectos.*

Estas cajas también son utilizadas para instalaciones de cableado estructurado (ver Figura 5.7.4.2.), los cajetines telefónicos también nos podrían servir, las cajas para breakers, y si no nos sirven ninguna de estas pues por último mandamos a doblar una caja a la medida.

A continuación mostramos cómo hacer un teclado para el control de accesos, primero utilizamos la caja DEXSON o cualquier otra marca, cortamos las 2 tuercas y lo colocamos en otro lugar, utilizando pega instantánea.

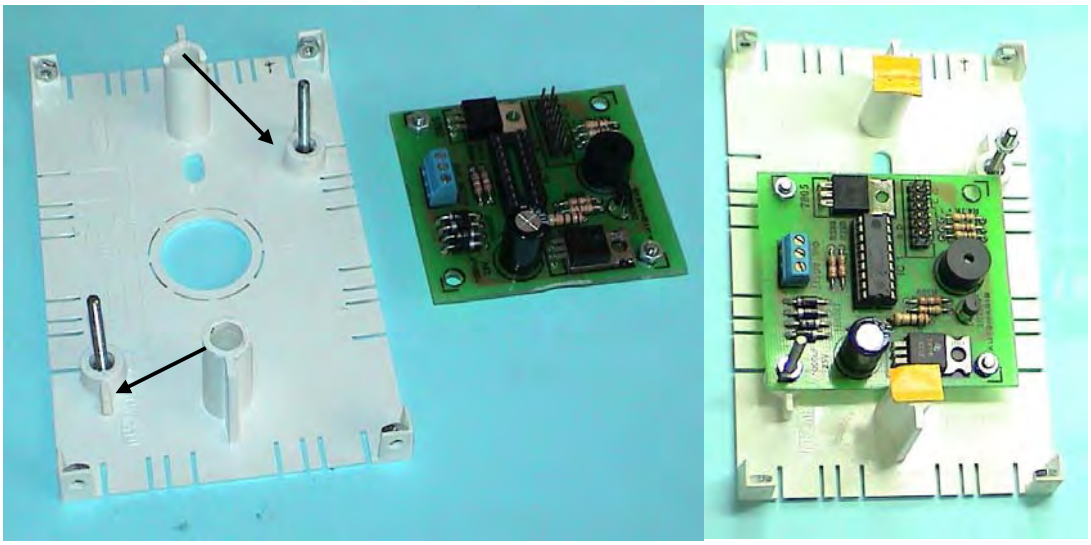


Figura 7.10.2. Cortamos las tuercas y las pegamos en cualquier lugar que deseemos.

Para fabricar el panel del teclado, primero dibujamos en un programa de dibujo todas las teclas, luego mandamos a imprimir en acetato con una impresora láser a color, pegamos un adhesivo blanco por la parte posterior, con la finalidad de que las partes transparentes se vean blancas.

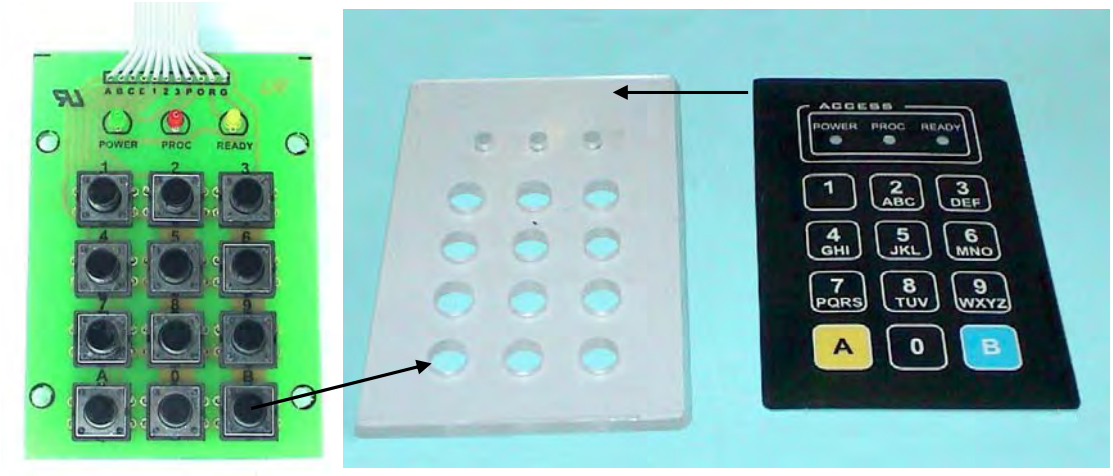


Figura 7.10.3. *Cortamos una lámina de acrílico y lo practicamos los agujeros por donde pasaran las teclas, imprimimos en una lámina de transparencia (acetato) los dibujos de las teclas y lo pegamos sobre el acrílico.*

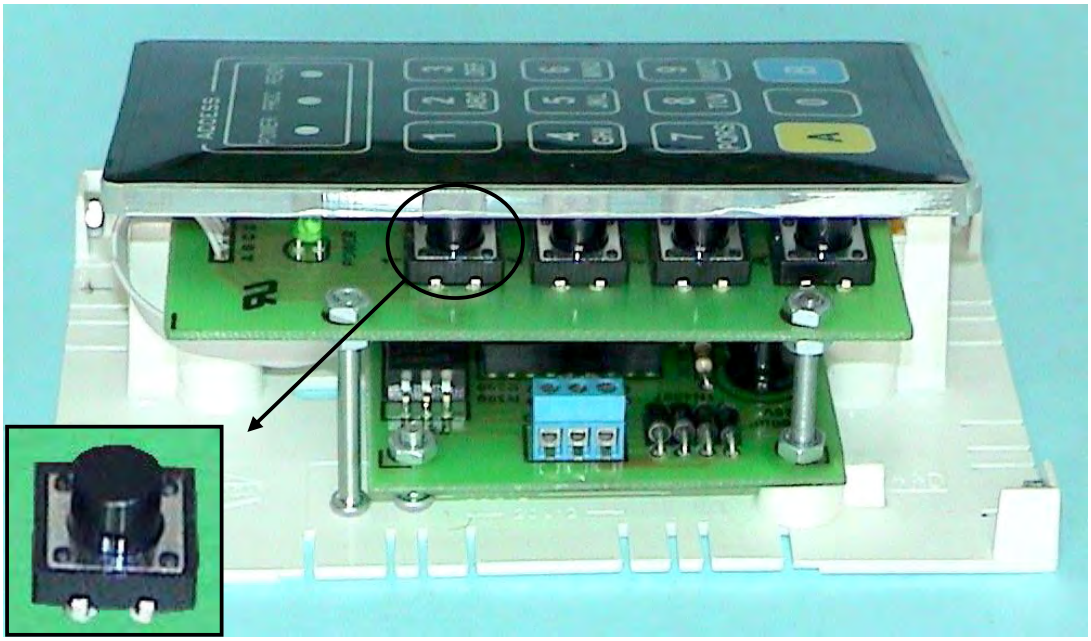


Figura 7.10.4. *Fotografía lateral del teclado para control de accesos, noten que los pulsadores son de 4,5 mm de alto suficiente para que atraviese a la lámina de acrílico que mide 4mm. de espesor*

En el mercado se puede encontrar diferentes cajas metálicas o plásticas, para proyectos o para instalaciones eléctricas, aquí una fotografía de 2 de ellas.

Figura 7.10.5. *Fotografía de una caja metálica y otra de plástico, ambas se utilizan para instalaciones eléctricas, con la caja de la izquierda haremos una alarma de 3 zonas y otra de 9 zonas y con la de la derecha haremos un PLC con LCD.*

Para hacer la caja de la alarma de 3 zonas primero debemos hacer los paneles, estos dibujamos en el computador y lo mandamos a imprimir en papel adhesivo blanco o transparencias.

Figura 7.10.6. *Fotografía de algunos paneles, unos impresos en papel adhesivo y otro en acetato.*

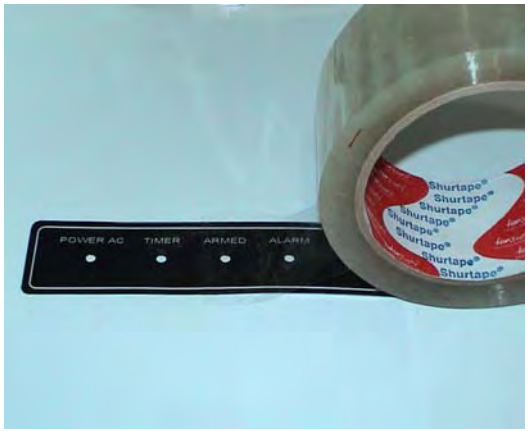


Figura 7.10.7. *Para proteger los dibujos se debe cubrir con una lámina adhesiva transparente, esto ayudará a que con el tiempo no se borren ni se mojen, ni tampoco se ensucien.*

Figura 7.10.8. *Fotografía de paneles de una alarma de 3 zonas y otra de 9 zonas con teclado.*

Figura 7.10.9. *Fotografía de un PLC con LCD fabricado en una caja de breakers de plástico.*

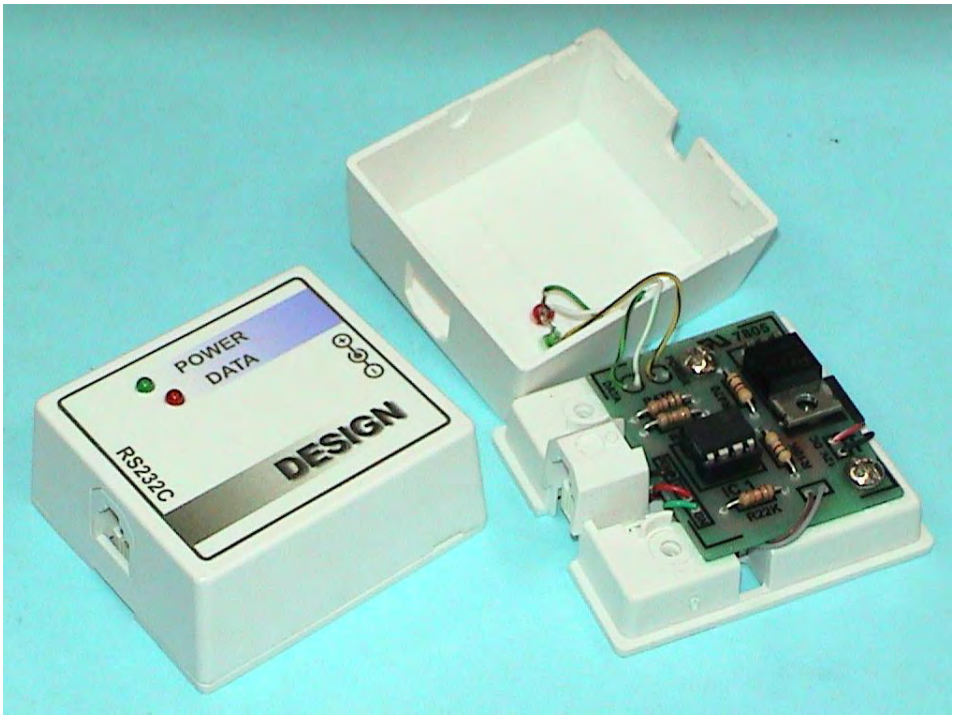


Figura 7.10.10. *Un cajetín telefónico, también nos podría servir para contener un proyecto.*

Figura 7.10.11. *Las cajas para proyectos que se venden en las tiendas electrónicas, también son una buena opción, algunas vienen con los agujeros listos para colocar switchs y leds, y con la ayuda de las impresiones láser a color en papel adhesivo, se les puede dar un acabado muy profesional.*

Si la caja que usted necesita no se encuentra en el mercado, puede mandar a fabricar una a su gusto, por ejemplo si para una fuente de poder necesita una caja que tenga un panel de control inclinado, debe mandar a fabricar unas tablas con un canal para sujetar los tornillos, y

luego se manda a doblar el tool a la medida correcta, se procede con los agujeros, la pintura y el papel adhesivo, el resultado final puede ser como la siguiente fotografía.

Figura 7.10.12. *Chasís de una fuente de poder construida con madera y tool doblado, para las patitas se puede utilizar los cauchos automotrices que se utilizan como topes para las puertas.*

Apéndice A SITIOS WEB RELACIONADOS CON ESTE LIBRO

www.mecanique.co.uk	Descarga de programas: Microcode y Pbp demo
www.IC-prog.com	Descarga de programa Ic-prog106A.zip
www.melabs.com	Compilador PicBasic Pro, Ejemplos en pbp
www.microchip.com	Productos e información de la familia de PIC´S
www.rentron.com	Ejemplos en pbp, información de dispositivos
www.frino.com.ar	Descarga manual de pbp, teoría del PIC
www.todopic.com.ar	Descarga manual en español de pbp
www.Redeya.com	Historia del PIC
www.monografias.com	Todo tipo de temas relacionado al PIC
www.x-robotics.com	Información sobre LCD 2x16, motores PAP, etc.
www.pablin.com.ar	Diagramas de proyectos con PIC´S
www.electronicaestudio.com	Productos con y para PIC´S
www.mikroelektronika.co.yu	Entrenadores, grabadores y libros de PIC´S
automasis@yahoo.es	Correo electrónico del autor de este libro
www.automasis.blogspot.com	Sitio web del libro.

Apéndice B PRÓXIMA ENTREGA

En el próximo volumen se incluirán proyectos con las demás declaraciones del compilador pbp 2.47, y además:
-Comunicación serial con LABview.
-Alarma programable con teclado.
-Circuitos grabadores y reproductores de voz.
-Transmisión y recepción infrarroja.
-Proyectos con Radio Frecuencia.
-Manejo de servo motores.
-Sensores de proximidad.
-Osciloscopio con PIC.
-Dispositivos one wire.
-Voltímetro con PIC.
-Calculadora completa con PIC.
-Y muchos proyectos más.