



**UNIVERSIDAD LAICA “ELOY ALFARO” DE MANABÍ
FACULTAD DE INGENIERIA
ESCUELA DE INGENIERIA ELECTRICA**

**TESIS DE GRADO
PREVIO A LA OBTENCIÓN DEL TÍTULO DE:
INGENIERO ELECTRICO**

**TEMA:
“CIRCUITOS INTEGRADOS PROGRAMABLES O
MICROCONTROLADORES PIC”**

**AUTORES:
SANDINO RIVERA ZAMBRANO
LUIS LUCAS MENDOZA**

**DIRECTOR DE TESIS:
ING. ENRIQUE FIGUEROA
MANTA – MANABÍ – ECUADOR
2011**

UNIVERSIDAD LAICA "ELOY ALFARO" DE MANABÍ

FACULTAD DE INGENIERIA

ESCUELA DE ELECTRICA

TESIS DE GRADO:

PREVIO A LA OBTENCION DEL TITULO DE:

INGENIERO ELECTRICO

TEMA:

"CIRCUITOS INTEGRADOS PROGRAMABLES O MICROCONTROLADORES PIC"

AUTORES:

SANDINO RIVERA ZAMBRANO

LUIS LUCAS MENDOZA

DIRECTOR DE TESIS:

ING. ENRIQUE FIGUEROA

MANTA – MANABÍ – ECUADOR

2011

INDICE DE TESIS

APROBACION DEL DIRECTOR DEL DIRECTOR DE TESIS.....	i
APROBACION DEL TRIBUNAL.....	ii
DEDICATORIA.....	iii
AGRADECIMIENTO.....	iv
DEDICATORIA.....	v
AGRADECIMIENTO.....	vi
RESUMEN.....	vii

CAPITULO I

INTRODUCCIÓN

1.1 TRANSISTOR.....	1
1.2 TRANSISTOR DE UNIÓN BIPOLAR.....	4
1.3 TRANSISTOR DE UNIÓN UNIPOLAR O DE EFECTO DE CAMPO.....	4
1.4 FOTOTRANSISTOR.....	6
1.5 TRANSISTORES Y ELECTRÓNICA DE POTENCIA.....	7
1.6 RECTIFICADOR CONTROLADO DE SILICIO (SCR).....	8
1.7 TRIAC.....	9
1.7.1 APLICACIONES MÁS COMUNES.....	9
1.8 OPTOACOPLADOR.....	10
1.8.1 Tipos.....	11
1.9 RELES.....	12
1.9.1 Ventajas del uso de relés.....	13

CAPITULO II

FUNDAMENTOS DE PROGRAMACIÓN EN LENGUAJE C PARA MICROCONTROLADORES PIC

2.1 Estructura básica de un programa en lenguaje C (sin funciones).....	14
2.2 Los siete elementos básicos de la programación.....	14
2.3 Instrucciones básicas de programación en lenguaje C.....	15
2.3.1 Instrucción de asignación (=)-.....	15
2.3.2 Instrucción de entrada de datos (variable=PORTx.....	16
2.3.3 Instrucción de salida de datos (PORTx=dato).....	16
2.3.4 Instrucción de decisión (if...else).....	17
2.3.5 Instrucción de ciclo controlado por una variable (for).....	19
2.3.6 Instrucción iterativa condicional (while).....	20
2.3.7 Instrucción hacer-mientras (do...while).....	21
2.3.8 Instrucción de selección múltiple (switch).....	22
2.4 Tipos de datos.....	24
2.5 FUNCIONES.....	26
2.5.1 Declaración de la función.....	26
2.5.2 Definición de la función.....	26
2.6 ESTRUCTURA BÁSICA DE UN PROGRAMA EN LENGUAJE C (CON FUNCIONES).....	27
2.7 Para tener en cuenta.....	30
2.8 FUNCIONES DE mikroc™ PARA LCD.....	31

CAPITULO III

INTRODUCCIÓN A LOS MICROCONTROLADORES

3.1 DEFINICION.....	35
3.2 CARACTERÍSTICAS ESPECIALES DEL MICROCONTROLADOR.....	36
3.3 CONFORMACIÓN Y DESCRIPCION DE PINES DE LOS CIRCUITO INTEGRADOS 16F84A, 16F88; 16F628 Y 16F877.	37
3.4 DISPOSICIÓN Y DESCRIPCIÓN DE PATILLAS.....	38
3.5 PIC16F84A vs. PIC16F88, PIC16F628A y PIC16F877A.....	40
3.6 PUERTOS DIGITALES.....	42
3.7 PRINCIPALES CARACTERÍSTICAS.....	42
3.7.1 PUERTO A	43
3.7.2 CARACTERÍSTICAS DE LAS TECNOLOGÍAS DE ENTRADA/SALIDA....	47
3.7.3 PUERTO B.....	48
3.7.4 PUERTOS A, B, C, D y E del PIC16F877A.....	51
3.8 VALORES MÁXIMOS ABSOLUTOS.....	51

CAPITULO IV

4.1 REQUERIMIENTOS BÁSICOS PARA PROGRAMAR UN PIC.....	53
4.2 COMO PROGRAMAR UN PIC.....	54
4.2.1 EDITAR.....	54
4.2.2 COMPILAR.....	55
4.2.3 QUEMAR EL PIC.....	55

4.2.4 PROBAR EL PROGRAMA.....	55
4.3 CIRCUITOS Y COMPONENTES BÁSICOS PARA ARMAR Y HACER FUNCIONAR UN PIC.....	56
4.3.1 Fuente de alimentación de 5 voltios.....	56
4.3.2 CRISTALES DE QUARZO.....	57

CAPITULO V

SOFTWARE Y HARDWARE NECESARIO Y CONVENIENTE PARA PROGRAMAR Y GRABAR UN PIC	
5.1 Tipos de software y hardware disponibles.....	58
5.1.1 SOFTWARE.....	59
5.1.2 HARDWARE.....	59
5.2 PROGRAMANDO LOS CIRCUITOS INTEGRADOS 16F88, 16F628, 16F877..	60
5.3 CONFIGURACIÓN INICIAL BÁSICA EN mikroC™.....	63
5.4 Características de mikroC™.....	5.4
5.5 PROGRAMADOR PICKit2 Clone PARA PUERTO USB.....	72
5.5.1 CIRCUITO IMPRESO VISTO DESDE LA CARA DE COMPONENTES.....	73
5.5.2 ESQUEMA ELÉCTRICO DEL PROGRAMADOR.....	74
5.6 PROCEDIMIENTO DE PROGRAMACIÓN.....	76
5.7 CONSIDERACIONES PRÁCTICAS.....	79
5.7.1 PINES NO UTILIZADOS.....	79
5.7.2 RESET INDESEADO #MCLR.....	80
5.7.3 FUNCIONAMIENTO ERRÁTICO DEL PIC.....	81
5.7.4 CONEXIÓN DE UN RELÉ ELECTROMECAÍNICO AL PIC.....	82

CAPITULO VI

APLICACIONES Y PRACTICAS UTILIZANDO EL MODULO PARA REALIZAR PRACTICAS BASADAS EN CARGAR Y VERIFICAR MICROCONTROLADORES EN INGENIERÍA ELÉCTRICA.....83

INTRODUCCION.....83

6.1 PROGRAMA QUE GENERA DOS SECUENCIAS O COMBINACIONES DIFERENTES, USANDO EL PIC 16F877, PARA SER IMPLEMENTADO COMO DESCONGESTIONADOR VEHICULAR, EN UNA INTERSECCIÓN DE TRANSITO.....84

6.1.1 CÓDIGO FUENTE85

6.2 MEDICIÓN DE MAGNITUDES ELÉCTRICAS USANDO UN MICROCONTROLADOR PIC, CON DISPLAY ALFANUMÉRICO CON PIC16F88.....87

6.2.1 CÓDIGO FUENTE.....88

6.3 VARIADOR DE VELOCIDAD USANDO UN MICROCONTROLADOR PIC PARA EL CONTROL.....89

6.3.1 RESUMEN.....89

6.3.2 ANTECEDENTES.....89

6.3.4 MATERIALES Y MÉTODOS.....90

6.3.5 CRITERIOS UTILIZADOS EN EL DISEÑO.....91

6.3.6 LÓGICA DE CONTROL.....94

6.3.7 EXCITACIÓN DE LOS MOSFET.....96

6.3.8 ETAPA DE POTENCIA.....97

6.3.9 PROCEDIMIENTO DE PRUEBA.....	100
6.3.10 CÓDIGO FUENTE.....	101
6.4 MARCADOR DE 7 SEGMENTOS UTILIZANDO UN MICROCONTROLADOR PIC.....	105
6.4.1 CÓDIGO FUENTE.....	106
CAPITULO VII	
CONCLUSIONES Y RECOMENDACIONES.....	107
7.1 CONCLUSIONES.....	107
7.2 RECOMENDACIONES.....	108
ANEXO	
BIBLIOGRAFIA	

APROBACION DEL DIRECTOR DE TESIS

En mi calidad de Director de Tesis del Trabajo de Investigación y desarrollo de sistema con el tema:

**“ CIRCUITOS INTEGRADOS PROGRAMABLES O
MICROCONTROLADORES PIC”**

De los Egresados Sandino Rivera Zambrano y Luis Lucas Mendosa, considero que el presente trabajo reúne los requisitos y meritos suficientes para ser sometido a la evaluación del Tribunal examinador que la Facultad de Ingeniería designen.

Manta, 10 de Noviembre del 2011.

.....

Ing. Enrique Figueroa

DIRECTOR DE TESIS

APROBACION DEL TRIBUNAL

Los miembros del Tribunal examinador aprueban el informe y proyecto de la investigación sobre **“CIRCUITOS INTEGRADOS PROGRAMABLES O MICROCONTROLADORES PIC”**, A los estudiantes Sr. Sandino Rivera Zambrano y Sr, Luis Lucas Mendoza, luego de haber sido analizado por los señores miembros del tribunal de grado de la Facultad de Ingeniería, Escuela de Eléctrica y en cumplimiento de lo que establece la Ley se da aprobada.

Manta, Noviembre del 2011

Para constancia firman:

MIEMBROS DEL TRIBUNAL

NOTAS DE CALIFICACION

DEDICATORIA

Dedico este trabajo de Tesis de forma espiritual a Dios, por ser guía y fuerza que llena mi espíritu lo que ayuda a enfrentar y alcanzar mis metas.

A mis padres, siendo la razón y mi guía de mis logros, a mis hermanos, siendo ellos el apoyo moral en momentos difíciles de mi vida y a todas las personas que al pasar del día también sueñan alcanzar o llegar al infinito.

SANDINO RIVERA ZAMBRANO

AGRADECIMIENTO

Agradezco al personal instructor docente, siendo los indicados a compartir sus enseñanzas al paso del día a día dando justificaciones de mi cultura y aprendizaje.

También a mis padres y hermanos por darme valor y fuerza para lograr mis metas propuestas.

SANDINO RIVERA ZAMBRANO

DEDICATORIA

Dedico este tema de grado primero a Dios porque es el que hace posible todos los sueños de una persona.

También a mis padres por su inmenso amor que me han dado a lo largo de mi vida y su armonía durante mi etapa de estudio.

A mis hermanos por sus sabios consejos.

LUIS LUCAS MENDOZA

AGRADECIMIENTO

Mis eternos agradecimientos a mis profesores de la Facultad de Ingeniería Eléctrica a todos en general, por compartir sus conocimientos en clases que me han servido hasta el día de hoy para mejorar como persona.

Personalmente a mis hermanos por ser mis amigos, quiero que ellos sepan que los quiero mucho y siempre los llevare en mi corazón.

A mis compañeros que siempre los recuerdo por haber compartido todos los días durante los años de estudio.

Al Ingeniero Enrique Figueroa por habernos ayudado en nuestro tema de tesis.

LUIS LUCAS MENDOZA

RESUMEN

El siguiente proyecto de Tesis se entra a fondo en el estudio de los Circuitos Integrados Programables PIC para el cual se toma en cuenta recursos y equipos didácticos para el mencionado estudio.

Se da a conocer el funcionamiento básico de los componentes utilizados en electrónica de potencia y se ilustra con ejemplos claros, prácticos y funcionales, llegando a la posibilidad de aplicarlo en demostraciones industriales avanzadas.

Con este proyecto se aportara al conocimiento de futuros y actuales estudiantes, para que ellos puedan en un futuro de estar en la capacidad de competir en proyectos mas avanzados, para que los puedan aplicar en su vida profesional.

CAPITULO I

INTRODUCCIÓN.

En el estudio del presente trabajo de tesis es necesario para llegar al objetivo el cual es la comprensión de como trabajar con componentes electrónicos de potencia básicos, ya que para realizar aplicaciones industriales basadas en Microcontroladores se necesita conocer el funcionamiento básico de los siguientes componentes.

1.1 TRANSISTOR

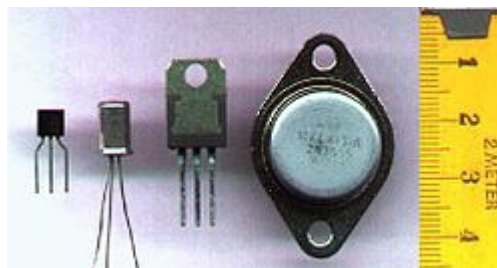
El transistor es un dispositivo electrónico semiconductor que cumple funciones de amplificador, oscilador, conmutador o rectificador. El término "transistor" es la contracción en inglés de *transfer resistor* ("resistencia de transferencia"). Actualmente se los encuentra prácticamente en todos los aparatos domésticos de uso diario: radios, televisores, grabadoras, reproductores de audio y video, hornos de microondas, lavadoras, automóviles, equipos de refrigeración, alarmas, relojes de cuarzo, computadoras, calculadoras, impresoras, lámparas fluorescentes, equipos de rayos X, tomógrafos, ecógrafos, reproductores mp3, teléfonos móviles, etc.



De manera simplificada, la corriente que circula por el "colector" es función amplificada de la que se inyecta en el "emisor", pero el transistor sólo gradúa la corriente que circula a través de sí mismo, si desde una fuente de corriente continua se alimenta la "base" para que circule la carga por el "colector", según el tipo de circuito que se utilice. El factor de amplificación o ganancia logrado entre corriente de colector y corriente de base, se denomina Beta del transistor. Otros parámetros a tener en cuenta y que son particulares de cada tipo de transistor son: Tensiones de ruptura de Colector Emisor, de Base Emisor, de Colector Base, Potencia Máxima, disipación de calor, frecuencia de trabajo, y varias tablas donde se grafican los distintos parámetros tales como corriente de base, tensión Colector Emisor, tensión Base Emisor, corriente de Emisor, etc. Los tres tipos de esquemas(configuraciones) básicos para utilización analógica de los transistores son emisor común, colector común y base común.

Modelos posteriores al transistor descrito, el transistor bipolar (transistores FET, MOSFET, JFET, CMOS, VMOS, etc.) no utilizan la corriente que se inyecta en el terminal de "base" para modular la corriente de emisor o colector, sino la tensión presente en el terminal de puerta o reja de control (graduador) y gradúa la conductancia del canal entre los terminales de Fuente y Drenaje. Cuando la conductancia es nula y el canal se encuentra estrangulado, por efecto de la tensión aplicada entre Compuerta y Fuente, es el campo eléctrico presente en el canal el responsable de impulsar los electrones desde la fuente al drenaje. De este modo, la corriente de salida en la carga conectada al Drenaje (D) será función amplificada de la Tensión presente entre la Compuerta (Gate) y Fuente (Source). Su funcionamiento es análogo al del triodo, con la salvedad que en el triodo los equivalentes a Compuerta, Drenador y Fuente son Reja (o Grilla Control), Placa y Cátodo.

Los transistores de efecto de campo, son los que han permitido la integración a gran escala disponible hoy en día, para tener una idea aproximada pueden fabricarse varios cientos de miles de transistores interconectados, por centímetro cuadrado y en varias capas superpuestas.



Distintos encapsulados de "transistores.

1.2 TRANSISTOR DE UNIÓN BIPOLAR

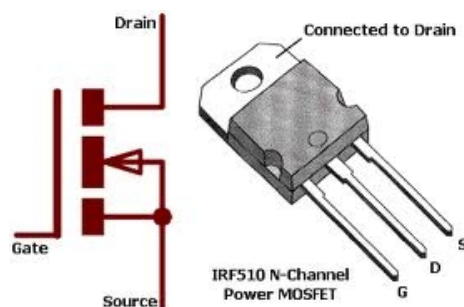
El transistor de unión bipolar, o BJT por sus siglas en inglés, se fabrica básicamente sobre un monocristal de Germanio, Silicio o Arseniuro de galio, que tienen cualidades de semiconductores, estado intermedio entre conductores como los metales y los aislantes como el diamante. Sobre el sustrato de cristal, se contaminan en forma muy controlada tres zonas, dos de las cuales son del mismo tipo, NPN o PNP, quedando formadas dos uniones NP.

La configuración de uniones PN, dan como resultado transistores PNP o NPN, donde la letra intermedia siempre corresponde a la característica de la base, y las otras dos al emisor y al colector que, si bien son del mismo tipo y de signo contrario a la base, tienen diferente contaminación entre ellas (por lo general, el emisor está mucho más contaminado que el colector).

1.3 TRANSISTOR DE UNIÓN UNIPOLAR O DE EFECTO DE CAMPO

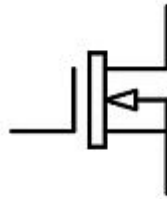
El transistor de unión unipolar, también llamado de efecto de campo de unión (JFET), fue el primer transistor de efecto de campo en la práctica. Lo forma una barra de material semiconductor de silicio de tipo N o P. En los terminales de la barra se establece un contacto óhmico, tenemos así un transistor de efecto de campo tipo N de la forma más básica. Si se difunden dos regiones P en una barra de material N y se conectan externamente entre sí, se producirá una puerta. A uno de estos contactos le llamaremos surtidor

y al otro drenador. Aplicando tensión positiva entre el drenador y el surtidor y conectando a puerta al surtidor, estableceremos una corriente, a la que llamaremos corriente de drenador con polarización cero. Con un potencial negativo de puerta al que llamamos tensión de estrangulamiento, cesa la conducción en el canal.

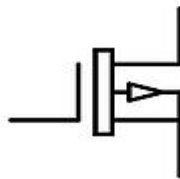


El transistor de efecto de campo, o FET por sus siglas en inglés, que controla la corriente en función de una tensión; tienen alta impedancia de entrada.

- Transistor de efecto de campo de unión, JFET, construido mediante una unión PN.
- Transistor de efecto de campo de compuerta aislada, IGFET, en el que la compuerta se aísla del canal mediante un dieléctrico.
- Transistor de efecto de campo MOS, MOSFET, donde MOS significa Metal-Óxido-Semiconductor, en este caso la compuerta es metálica y está separada del canal semiconductor por una capa de óxido.



Transistor MOSFET de empobrecimiento canal N



Transistor MOSFET de empobrecimiento canal P

1.4 FOTOTRANSISTOR

Los fototransistores son sensibles a la radiación electromagnética en frecuencias cercanas a la de la luz visible; debido a esto su flujo de corriente puede ser regulado por medio de la luz incidente. Un fototransistor es, en esencia, lo mismo que un transistor normal, sólo que puede trabajar de 2 maneras diferentes:

- Como un transistor normal con la corriente de base (I_B) (modo común).
- Como fototransistor, cuando la luz que incide en este elemento hace las veces de corriente de base. (I_P) (modo de iluminación).

1.5 TRANSISTORES Y ELECTRÓNICA DE POTENCIA

Con el desarrollo tecnológico y evolución de la electrónica, la capacidad de los dispositivos semiconductores para soportar cada vez mayores niveles de tensión y corriente ha permitido su uso en aplicaciones de potencia. Es así como actualmente los transistores son empleados en convertidores estáticos de potencia, controles para motores y llaves de alta potencia (principalmente inversores), aunque su principal uso está basado en la amplificación de corriente dentro de un circuito cerrado.

1.6 RECTIFICADOR CONTROLADO DE SILICIO (SCR)

Los rectificadores controlados de silicio SCR se emplea como dispositivo de control.

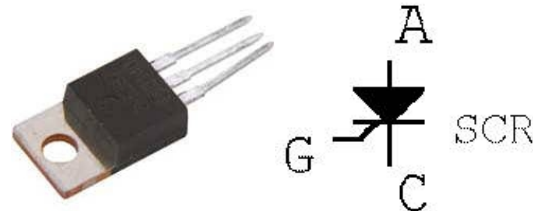
El rectificador controlado de silicio SCR, es un semiconductor que presenta dos estados estables: en uno conduce, y en otro está en corte (bloqueo directo, bloqueo inverso y conducción directa).

El objetivo del rectificador controlado de silicio SCR es retardar la entrada en conducción del mismo, ya que como se sabe, un rectificador controlado de silicio SCR se hace conductor no sólo cuando la tensión en sus bornes se hace positiva (tensión de ánodo mayor que tensión de cátodo), sino cuando siendo esta tensión positiva, se envía un impulso de cebado a puerta.

Como lo sugiere su nombre, el SCR es un rectificador, por lo que pasa corriente sólo durante los semiciclos positivos de la fuente de ca. El semiciclo positivo es el semiciclo en que el ánodo del SCR es mas positivo que el cátodo. Esto significa que el SCR no puede estar encendido más de la mitad del tiempo. Durante la otra mitad del ciclo, la polaridad de la fuente es negativa, y esta polaridad negativa hace que el SCR tenga polarización inversa, evitando el paso de cualquier corriente a la carga.

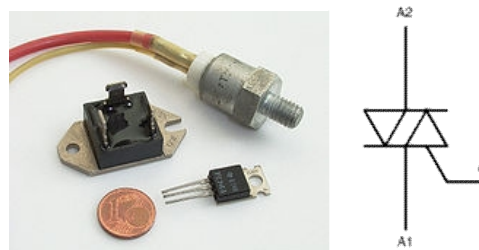
El SCR se asemeja a un diodo rectificador pero si el ánodo es positivo en relación al cátodo no circulará la corriente hasta que una corriente positiva se inyecte en la puerta. Luego el diodo se enciende y no se apagará hasta que

no se remueva la tensión en el ánodo-cátodo, de allí el nombre rectificador controlado.



1.7 TRIAC

Un **TRIAC** o **Triodo para Corriente Alterna** es un dispositivo semiconductor, de la familia de los transistores. La diferencia con un tiristor convencional es que éste es unidireccional y el TRIAC es bidireccional. De forma coloquial podría decirse que el TRIAC es un interruptor capaz de conmutar la corriente alterna.



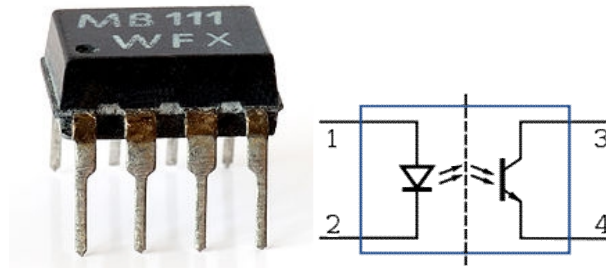
1.7.1 APLICACIONES MÁS COMUNES

- Su versatilidad lo hace ideal para el control de corrientes alternas.
- Una de ellas es su utilización como interruptor estático ofreciendo muchas ventajas sobre los interruptores mecánicos convencionales y los relés.
- Funciona como interruptor electrónico y también a pila.
- Se utilizan TRIACs de baja potencia en muchas aplicaciones como atenuadores de luz, controles de velocidad para motores eléctricos, y en los sistemas de control computarizado de muchos elementos caseros. No obstante, cuando se utiliza con cargas inductivas como motores eléctricos, se deben tomar las precauciones necesarias para asegurarse que el TRIAC se apaga correctamente al final de cada semiciclo de la onda de Corriente alterna.

1.8 OPTOACOPLADOR

Un **optoacoplador**, también llamado *optoaislador* o aislador acoplado ópticamente, es un dispositivo de emisión y recepción que funciona como un interruptor excitado mediante la luz emitida por un diodo LED que satura un componente optoelectrónico, normalmente en forma de fototransistor o fototriac. De este modo se combinan en un solo dispositivo semiconductor, un fotoemisor y un fotorreceptor cuya conexión entre ambos es óptica. Estos elementos se encuentran dentro de un encapsulado que por lo general es del

tipo DIP. Se suelen utilizar para aislar electricamente a dispositivos muy sensibles.

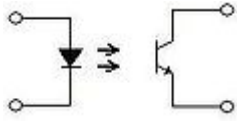


El optoacoplador combina un LED y un fototransistor..

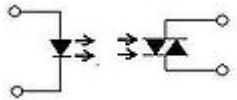
La ventaja fundamental de un optoacoplador es el aislamiento eléctrico entre los circuitos de entrada y salida. Mediante el optoacoplador, el único contacto entre ambos circuitos es un haz de luz. Esto se traduce en una resistencia de aislamiento entre los dos circuitos del orden de miles de M . Estos aislamientos son útiles en aplicaciones de alta tensión en las que los potenciales de los dos circuitos pueden diferir en varios miles de voltios.

1.8.1 Tipos

En general, los diferentes tipos de optoacopladores se distinguen por su diferente etapa de salida. Entre los principales caben destacar el fototransistor, ya mencionado, el fototriac y el fototriac de paso por cero. En este último, su etapa de salida es un triac de cruce por cero, que posee un circuito interno que conmuta al triac sólo en los cruce por cero de la fuente.



Etapa de salida a fototransistor.

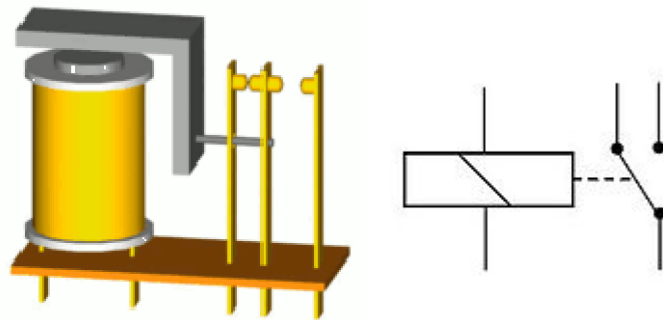


Etapa de salida a fototriac.

1.9 RELES

El relé o relevador es un dispositivo electromecánico. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes. Fue inventado por Joseph Henry en 1835.

Dado que el relé es capaz de controlar un circuito de salida de mayor potencia que el de entrada, puede considerarse, en un amplio sentido, como un amplificador eléctrico. Como tal se emplearon en telegrafía, haciendo la función de repetidores que generaban una nueva señal con corriente procedente de pilas locales a partir de la señal débil recibida por la línea. Se les llamaba "relevadores" De ahí "relé".



1.9.1 Ventajas del uso de relés

La gran ventaja de los relés electromagnéticos es la completa separación eléctrica entre la corriente de accionamiento, la que circula por la bobina del electroimán, y los circuitos controlados por los contactos, lo que hace que se puedan manejar altos voltajes o elevadas potencias con pequeñas tensiones de control. También ofrecen la posibilidad de control de un dispositivo a distancia mediante el uso de pequeñas señales de control.

CAPITULO II

FUNDAMENTOS DE PROGRAMACIÓN EN LENGUAJE C PARA MICROCONTROLADORES PIC

2.1 Estructura básica de un programa en lenguaje C (sin funciones)

Todos los programas (código fuente) en lenguaje C tienen una estructura básica, a partir de la cual se desarrolla cualquier aplicación del usuario:

```
//Nombre_de_programa.c
//Descripción del programa.
//Autor: Ing. Sandino R.
//Declaración de variables
...

//Función principal
void main( ){
    //Instrucciones del programa.
    ...
}
```

2.2 Los siete elementos básicos de la programación

El propósito de la mayoría de los programas es resolver un problema. Los programas resuelven los problemas por medio de la manipulación de información o datos. Normalmente los programas se caracterizan por permitir el ingreso de información, tener uno o varios lugares de almacenamiento de dicha información, contar con las instrucciones para manipular estos datos y obtener algún resultado del programa que sea útil para el usuario. También, las instrucciones se pueden organizar de tal forma que algunas de ellas se

ejecuten sólo cuando una condición específica (o conjunto de condiciones) sea verdadera, otras instrucciones se repitan un cierto número de veces y otras pueden ser agrupadas en bloques que se ejecutan en diferentes partes de un programa.

Lo anterior constituye una breve descripción de los siete elementos básicos de la programación: *entrada de datos, tipos de datos, operaciones, salida, ejecución condicional, lazos y funciones*. Una vez que se dominan estos elementos se puede afirmar que se conocen los fundamentos de la programación, con lo cual ya es posible desarrollar una gran cantidad de aplicaciones de diversa índole.

Nota: La EJECUCIÓN DE UNA INSTRUCCIÓN consiste en la realización de las operaciones especificadas en esa instrucción. De la ejecución se encarga la CPU (unidad central de proceso) del microcontrolador.

2.3 Instrucciones básicas de programación en lenguaje C

2.3.1 Instrucción de asignación (=).-

Permite asignar a una variable un valor constante, el contenido de otra variable o el resultado de una expresión matemática. La asignación va de derecha a izquierda. Por ejemplo,

```
suma=0; //El valor 0 se almacena en la variable suma.
```

```
x0=x1; //El contenido de la variable x1 se almacena en la variable x0.
```

```
dx=(b-a)/n; //El resultado de la expresión matemática se almacena en la variable dx.
```

2.3.2 Instrucción de entrada de datos (variable=PORTx).-

Permite el ingreso de uno o más datos a través de los pines del microcontrolador y almacenarlos en una o más variables. Por ejemplo,

```
variable=PORTA; //Los bits del puerto A se almacenan en la variable.
```

El siguiente es un caso especial utilizado en la programación de microcontroladores:

```
PORTB=PORTA; //Los bits del puerto A se envían hacia los pines del puerto B.
```

También se puede leer el estado individual de cada bit de un puerto:

```
variable=RB3_bit; //Lee el estado del pin RB3 y lo guarda en la variable.
```

2.3.3 Instrucción de salida de datos (PORTx=dato).-

Permite el envío de datos, el contenido de una variable o el resultado de una expresión matemática hacia los pines de un puerto. Por ejemplo,

```
PORTA=0x00; //Todos los pines del puerto A se ponen en 0.  
PORTB=variable; Los bits de la variable son enviados hacia los pines del puerto B.  
PORTB=PORTA+65; //El valor del puerto A más 65 se envía hacia el puerto B.
```

Como caso especial, se pueden enviar bits individuales a cada uno de los pines de un puerto:

```
RB0_bit=0; //El pin RB0 se pone en 0.
```

2.3.4 Instrucción de decisión (if...else).-

Permite la ejecución de las **instrucciones1** si la condición es verdadera, de lo contrario se ejecutan las **instrucciones2**. Las llaves { } no son necesarias cuando hay una sola instrucción.

```
if (condición){
    instrucciones1;
}
else{
    instrucciones2;
}
```

Ejemplo 1:

Si el contenido de la variable **codigo** es igual al contenido de la variable **clave**, se ejecutan las primeras cuatro instrucciones; de lo contrario se ejecutan únicamente los dos últimas instrucciones.

```
if (codigo==clave){
    intentos=0;
    RA7_bit=1;
    Delay_1sec( );
    RA7_bit=0;
}
else{
    intentos++;
    Delay_ms(200);
}
```


Ejemplo 2:

Instrucción de decisión sin **else**. Esta es una variante muy utilizada cuando se desea condicionar la ejecución de un grupo de instrucciones.

Las dos instrucciones se ejecutarán únicamente si la variable **contador** es igual a 2, de lo contrario la ejecución continúa a partir de la línea **//Aquí**.

```
if (contador==2){
    RB6_bit=~RB6_bit;
    contador=0;
}
//Aquí.
```

Ejemplo 3:

Similar al caso anterior pero con una sola instrucción. Si la variable **horas** es igual a 24 se reinicia esta variable con un valor de cero.

```
if (horas==24) horas=0;
```

Nota 1: Las condiciones se obtienen por medio de los operadores de relación y los operadores lógicos.

Nota 2: Operadores de relación:

> Mayor que
>= Mayor o igual que
< Menor que
<= Menor o igual que
== Igual a (nótese la diferencia con el operador de asignación =)
!= No es igual a

Nota 3: Operadores lógicos:

&& Y
|| O

2.3.5 Instrucción de ciclo controlado por una variable (for).-

Permite ejecutar un grupo de instrucciones de manera repetitiva, una determinada cantidad de veces.

```
for (número de veces){  
  instrucciones;  
}
```

Ejemplo 1:

La variable **i** tiene un valor inicial de 7 (**i=7**) y un valor final de 1 (**i>=1**). Esta variable va disminuyendo de 1 en 1 (**i--**). Por lo tanto las dos instrucciones se ejecutarán en 7 ocasiones. La primera vez cuando **i=7**, la segunda cuando **i=6**, la tercera cuando **i=5** y así sucesivamente hasta la séptima vez cuando **i=1**. Luego la ejecución continúa a partir de la línea **//Aquí**.

```
for (i=7; i>=1; i--){  
  PORTB=PORTB<<1;  
  Delay_ms(500);  
}  
//Aquí.
```

Ejemplo 2:

El valor inicial de la variable es 1 y su valor final es 3. La variable **i** se va incrementando de 1 en 1 (**i++**). Por lo tanto la instrucción se ejecuta tres veces, lo que da como resultado un retardo de 3 segundos. Luego la ejecución continúa a partir de la línea **//Aquí**.

```
for (i=1; i<=3; i++)  
  Delay_1sec( );  
//Aquí.
```

2.3.6 Instrucción iterativa condicional (while).-

Permite ejecutar un grupo de instrucciones de manera repetitiva, mientras una condición sea verdadera. Primero se revisa la condición para determinar su valor de verdad (verdadero o falso) y luego se ejecutan las instrucciones.

```
while (condición){  
    instrucciones;  
}
```

Ejemplo 1:

La ejecución del programa permanece indefinidamente en esta línea mientras el bit IOFS del registro OSCCON sea igual a cero. Como caso particular no se ejecuta ninguna instrucción (la cual debería estar antes del punto y coma).

```
while (OSCON.IOFS==0) ;
```

Ejemplo 2:

Ejemplo de un lazo infinito. En lenguaje C, cualquier valor numérico diferente a cero se considera VERDADERO, y un valor numérico igual a cero se considera FALSO.

Al valor numérico del puerto A se le suma el valor 65, el resultado se envía hacia los pines del puerto B. Este proceso se repite continua e indefinidamente, debido a que la condición siempre es verdadera (1).

```
while (1)  
    PORTB=PORTA+65;
```

Ejemplo 3:

Las cuatro instrucciones encerradas por { } se ejecutarán indefinidamente mientras el valor del bit RB0 sea igual a 0.

```
while (RB0_bit==0){
  RB1_bit=1;
  Delay_ms(500);
  RB1_bit=0;
  Delay_ms(200);
}
```

2.3.7 Instrucción hacer-mientras (do...while).-

Permite ejecutar un grupo de instrucciones de manera repetitiva, mientras una condición sea verdadera. Es similar a la instrucción **while**, con la diferencia de que primero se ejecutan las instrucciones y luego se revisa la condición.

```
do{
  instrucciones;
}
while (condición);
```

Ejemplo 1:

La variable **kp** tiene un valor inicial de cero. La instrucción **kp=Keypad_Key_Click();** se ejecuta y luego se revisa la condición (**!kp**). Mientras **kp** sea igual a cero (FALSO) la condición será VERDADERA (**!kp**), debido al operador de negación **!** que cambia el valor de verdad a su estado contrario. Como resultado se tendrá un lazo infinito mientras la variable **kp** siga en cero. Cuando la variable **kp** cambie de valor como consecuencia de

la pulsación de una tecla, la condición será FALSA y la ejecución continuará en la línea **//Aquí**.

```
kp=0;
do
  kp=Keypad_Key_Click( );
while (!kp);
//Aquí.
```

Ejemplo 2:

Las cuatro instrucciones dentro de { } se ejecutarán indefinidamente mientras la variable **tecla** sea diferente a 1.

```
do{
  ingreso( );
  raiz( );
  pn_1( );
  seg_grado( );
}
while (tecla != 1);
```

Nota: A diferencia de la instrucción **while**, en la instrucción **do...while** las instrucciones se ejecutan por lo menos una vez.

2.3.8 Instrucción de selección múltiple (switch).-

Permite la ejecución de un grupo de instrucciones de varios grupos posibles, dependiendo del valor de una variable.

```
switch (variable){
  case 1: instrucciones1;
         break;

  case 2: instrucciones2;
```

```
        break;

    case 3: instrucciones3;
        break;

    ...

    default: instrucciones;
}
```

Si la **variable** es igual a 1 se ejecutan únicamente las **instrucciones1**, si es igual a 2 se ejecutan únicamente las **instrucciones2** y así sucesivamente. Si la **variable** no es igual a ninguno de los casos (**case**) se ejecutan las instrucciones por defecto (**default**)

Ejemplo:

Esta es una función numérica que da como resultado un número hexadecimal dependiendo del valor que tenga la variable **digit**. Si **digit** es igual a 0 la función devuelve (**return**) el valor 0x3F. Si **digit** es igual a 1, la función devuelve el valor 0x06, y así sucesivamente. Este ejemplo es una variante de la instrucción **switch**, donde no aparece el elemento **default**.

```
char Bin2_7seg(char digit){
    switch (digit){
        case 0: return 0x3F;
        case 1: return 0x06;
        case 2: return 0x5B;
        case 3: return 0x4F;
        case 4: return 0x66;
    }
}
```

2.4 Tipos de datos.-

Los tipos de datos permiten la declaración de las variables que se utilizarán en un programa. El tipo de variable dependerá del propósito que ésta tendrá y del valor numérico que almacenará. Todas las variables tienen que ser declaradas para no tener errores durante la compilación. Los tipos de datos más utilizados son:

Tipo	Tamaño (en bytes)	Rango
bit	1 bit	0 ó 1
char	1	0...255
signed char	1	-128...127
int	2	-32768...32767
unsigned	2	0...65535
long	4	-2147483648...2147483647
unsigned long	4	0...4294967295
float	4	$-1.5 * 10^{45} \dots +3.4 * 10^{-38}$

El tipo **float** es para números con punto decimal, mientras que los demás tipos son para números enteros.

Ejemplo 1:

Declaración de diferentes variables de varios tipos.

```
bit a, b, c; //Las variables a, b y c se declaran como tipo bit.
```

```
char Numero=34;//La variable Numero (no se deben colocar tildes) se declara como tipo char y su valor inicial es 34.
```

```
signed char numero, i; //Las variables numero e i se declaran como tipo signed char.
```

```
unsigned decenas, clave=3589; //Las variables decenas y
clave se declaran como tipo unsigned. El valor inicial de
la variable clave es 3.589.
```

```
float pi=3.14159; //La variable pi se declara como tipo
float y su valor inicial es 3.14159
```

Ejemplo 2:

Como caso especial están los **arreglos** (*arrays*). Un arreglo es un conjunto de variables que tienen el mismo nombre y se diferencian por un subíndice; por ejemplo, las variables **texto2₀**, **texto2₁**, **texto2₂** y **texto2₃**. El nombre es el mismo (**texto2**) pero se diferencian por el subíndice (0, 1, 2 ó 3). Un arreglo se indica por medio de los corchetes [], dentro de los cuales se coloca el número de variables del arreglo.

```
char texto2[4]; //Declaración de un arreglo de 4
variables tipo char.
```

Ejemplo 3:

Una **cadena de caracteres** (*string*) se puede declarar como un arreglo de variables tipo **char**. Para hacer referencia a la cadena de caracteres se emplea el nombre del arreglo.

```
char texto1[]="Conteo:" //Declaración de una cadena de
caracteres.
...
Lcd_Out(1,6,texto1); //Escribe el texto "Conteo:" en la
pantalla LCD, a partir de la fila 1 y la columna 6.
```


2.5 FUNCIONES

Una función es una agrupación de instrucciones para formar una nueva instrucción creada por el programador (usuario).

Empleando funciones, la solución total de un determinado problema se divide en varios subproblemas, cada uno de los cuales es resuelto por medio de una función particular, aplicando de esta manera la conocida máxima “Divide y vencerás”.

Las funciones constituyen una de las características fundamentales del lenguaje C, pues todo programa bien escrito hace uso de ellas.

Para poder utilizar una función se deben cumplir los dos siguientes pasos:

2.5.1 Declaración de la función.- Consiste en indicar el tipo, nombre y parámetros de la función:

```
tipo nombre ( parámetro1, parámetro2, ... );
```

2.5.2 Definición de la función.- Consiste en indicar las instrucciones que forman parte de dicha función:

```
tipo nombre ( parámetro1, parámetro2, ... ) {  
    instrucciones;  
}
```

2.6 ESTRUCTURA BÁSICA DE UN PROGRAMA EN LENGUAJE C (CON FUNCIONES).-

Todos los programas (código fuente) en lenguaje C tienen una estructura básica, a partir de la cual se desarrolla cualquier aplicación del usuario:

```
//Nombre_de_programa.c
//Descripción del programa.
//Autor: Ing. Sandino R.
//*****Declaración de funciones (prototipos)*****
...
//*****Fin de declaración de funciones*****

//=====Declaración de variables=====
...
//=====Fin de declaración de variables=====

//*****Función principal*****
void main( ){
    //Instrucciones del programa.
    ...
}
//*****Fin de función principal*****

//=====Definición de funciones=====

función1{
    instrucciones1;
}

función2{
    instrucciones2;
}
//=====Fin de definición de funciones=====
```

Nota 1: Los tipos de funciones más empleadas son numéricas (**char**) y nulas (**void**). Las primeras retornan (**return**) o devuelven como resultado un número, mientras que las segundas simplemente ejecutan un grupo de instrucciones.

Ejemplo 1:

La función es numérica (**char**), su nombre es **Bin2_7seg** y tiene un parámetro **digit** de tipo **char**.

La función se utiliza como si fuera una instrucción cualquiera, tomando en cuenta el tipo de función y su(s) parámetro(s). En este ejemplo se tiene **PORTB=Bin2_7seg(PORTA)**. Esto significa que la variable **PORTA** reemplaza a la variable **digit**. Por lo tanto si **PORTA** es igual a 0, la función devuelve el valor 0x3F que será enviado al puerto B. Si **PORTA** es igual a 1, la función devuelve 0x06 que será enviado al puerto B, y así sucesivamente.

```
//7seg1.c
//Se utiliza la función Bin2_7seg que transforma un
número binario a su
//equivalente en 7 segmentos.

char Bin2_7seg(char digit); //Prototipo de la función.

void main(){
OSCCON=0x40; //Oscilador interno a 1MHz.
while (OSCCON.IOFS==0); //Esperar mientras el oscilador
está inestable.
PORTA=0x00; //Inicialización.
PORTB=0x00;
ANSEL=0x00; //Pines AN<6:0> como E/S digital.
TRISB=0x00; //Puerto B como salida.
while (1) PORTB=Bin2_7seg(PORTA);
}

char Bin2_7seg(char digit){ //Definición de la función.
switch (digit){
case 0: return 0x3F; //0x3F es el código 7-segmentos
del 0.
case 1: return 0x06; //0x06 es el código 7-segmentos
del 1.
case 2: return 0x5B;
case 3: return 0x4F;
case 4: return 0x66;
```

```

    case 5: return 0x6D;
    case 6: return 0x7D;
    case 7: return 0x07;
    case 8: return 0x7F;
    case 9: return 0x67;
}
}

```

Ejemplo 2:

Variante del ejemplo anterior, en el que se hace únicamente la definición de la función (sin declaración). Se debe hacer antes de la función principal, de lo contrario se producirán errores de compilación por tratar de usar una función desconocida.

```

//7seg1.c
//Se utiliza la función Bin2_7seg que transforma un
número binario a su
//equivalente en 7 segmentos.

char Bin2_7seg(char digit){ //Definición de la función.
    switch (digit){
        case 0: return 0x3F; //0x3F es el código 7-segmentos
del 0.
        case 1: return 0x06; //0x06 es el código 7-segmentos
del 1.
        case 2: return 0x5B;
        case 3: return 0x4F;
        case 4: return 0x66;
        case 5: return 0x6D;
        case 6: return 0x7D;
        case 7: return 0x07;
        case 8: return 0x7F;
        case 9: return 0x67;
    }
}

void main(){
    OSCCON=0x40; //Oscilador interno a 1MHz.
    while (OSCCON.IOFS==0); //Esperar mientras el oscilador
está inestable.
    PORTA=0x00; //Inicialización.
}

```

```
PORTB=0x00;
ANSEL=0x00; //Pines AN<6:0> como E/S digital.
TRISB=0x00; //Puerto B como salida.
while (1) PORTB=Bin2_7seg(PORTA);
}
```

2.7 Para tener en cuenta

Los comentarios se inician con la doble barra diagonal **//**.

Los signos de agrupación siempre deben estar en pareja, es decir si hay tres llaves de apertura **{**{, deben haber tres llaves de cierre correspondientes **}}**}.
}}.

Lo mismo con los paréntesis **()**.

Los números hexadecimales se escriben comenzando siempre con **0x**, por ejemplo **0x0A**, **0x16**, **0xFD**, etc.

Los números binarios se escriben comenzando siempre con **0b**, por ejemplo **0b001110**, **0b11101111**, etc.

Los números decimales se escriben de la forma común y corriente, por ejemplo **64**, **126**, **12.75**, etc.

No se debe confundir el operador de asignación **(=)** con el operador de comparación **(==)** igual a.

El punto y coma **(;)** indica el final de una instrucción, por lo tanto hay que tener mucho cuidado para colocarlo en el lugar apropiado.

Las llaves **{ }** no son necesarias en aquellos casos en los que únicamente se va a ejecutar una instrucción (ver los ejemplos a lo largo de este documento).

Todo programa en lenguaje C debe tener una función principal **(main)**, y su nombre no debe ser cambiado.

Uno de los errores más frecuentes es la falta de declaración de las variables empleadas en un programa.

2.8 FUNCIONES DE mikroC™ PARA LCD

mikroC™ proporciona una librería para comunicación con LCDs (con el controlador HD44780 o compatibles) a través de un interfaz de 4 bits para datos. Para el trabajo con el módulo LCD se debe añadir la librería *Lcd*, que contiene las funciones listadas en la tabla. Para poder utilizar estas funciones se debe declarar previamente un total de 12 variables: 6 que definen los pines del PIC y otras 6 que permiten programar su sentido de circulación de datos.

Función	Descripción
Lcd_init()	Inicializa el módulo LCD
Lcd_Out(fila, columna, texto)	Visualiza texto en la posición especificada
Lcd_Out_CP(texto)	Visualiza texto en la posición actual del cursor
Lcd_Chr(fila, columna, carácter)	Visualiza un carácter en la posición especificada
Lcd_Chr_CP(carácter)	Visualiza un carácter en la posición actual del cursor
Lcd_Cmd(comando)	Envía un comando al LCD

Funciones para LCD de la librería Lcd de mikroC™

Comando LCD	Propósito
LCD_FIRST_ROW	Mueve el cursor a la primera fila
LCD_SECOND_ROW	Mueve el cursor a la segunda fila
LCD_CLEAR	Borra la pantalla
LCD_RETURN_HOME	Retorna el cursor a su origen, retorna una pantalla desplazada a su posición original. La RAM del LCD no se ve afectada
JXD_CURSOR_OFF	Apaga el cursor
LCD_UNDERLINE_ON	Enciende el cursor de subrayado
LCD_BLINK_CURSOR_ON	Enciende el cursor con parpadeo
LCD_MOVE_CURSOR_LEFT	Mueve el cursor a la izquierda sin modificar la RAM del LCD
LCD_MOVE_CURSOR_RIGHT	Mueve el cursor a la derecha sin modificar la RAM del LCD
LCD_TURN_ON	Enciende el módulo LCD
LCD_TURN_OFF	Apaga el módulo LCD
LCD_SHIFT_LEFT	Desplaza la pantalla a la izquierda sin modificar la RAM del LCD
LCD_SHIFTJ_RIGHT	Desplaza la pantalla a la derecha sin modificar la RAM del LCD

Lista de comandos disponibles para LCD

CAPITULO III

INTRODUCCIÓN A LOS MICROCONTROLADORES

El uso de los microcontroladores en las aplicaciones de electrónica digital y analógica se ha popularizado a partir de la reducción de sus precios y la integración cada vez mayor de una gran variedad de periféricos, así como el incremento en la capacidad de memoria de datos y programa, en este sentido el PIC16F88, el PIC16F628A y el PIC16F877A pueden considerarse como tres de las mejores alternativas a la hora de seleccionar un microcontrolador, debido a sus altas prestaciones y precio reducido, si se los compara con el conocido PIC16F84A, que presenta grandes limitaciones en periféricos y memoria, con un precio relativamente elevado (de acuerdo a la información obtenida de la página del fabricante: www.microchip.com).

Por otra parte, hoy en día se continúa empleando ampliamente el lenguaje ensamblador para la programación de las aplicaciones con microcontroladores, a pesar de las grandes complicaciones y desventajas que implica, y que seguramente serán conocidas por los lectores: muchas líneas de código, aun para las aplicaciones más simples, falta de librerías de funciones que hagan más eficiente la programación, excesivo tiempo requerido para el desarrollo de una aplicación, conocimiento detallado del hardware y del set de instrucciones, etc. Una de las causas para que esto

sea así es la escasez de bibliografía y ejemplos prácticos existentes acerca de cómo desarrollar aplicaciones con microcontroladores PIC en lenguajes de alto nivel, tales como el muy difundido y conocido lenguaje C. Otra es la falta de difusión de los ambientes de desarrollo integrado para lenguajes de alto nivel, a pesar de que existen varios de ellos desde hace algunos años, por ejemplo: **CCS, PICC, DEM ICT, PCW, PICBASÍC PRO**, mikroC™ PRO for PIC, entre otros.

Existen muchos libros sobre microcontroladores, pero en la gran mayoría hay una vasta cantidad de conceptos y definiciones teóricas, en volúmenes de unas 400 páginas a enciclopedias de unas 800 páginas en varios tomos, que sinceramente 150 brindan solución a la principal necesidad del estudiante: la Implementación. Práctica de los conceptos teóricos, por lo cual al final de largas horas de estudio el alumno se siente decepcionado, pues conoce mucha teoría y no sabe cómo llevarla a la práctica.

Con todos estos antecedentes en mente y pensando en esta necesidad clamorosa se ha desarrollado este texto, que sin dejar de lado los conceptos teóricos relevantes, se enfoca principalmente en la solución de ejemplos de problemas prácticos, de manera directa y eficiente, integrando de manera sistemática y ordenada todos los detalles de la programación en lenguaje C de los microcontroladores PIC16F88, PIC16F628A y PIC16F877A, desde los aspectos más básicos hasta los más avanzados, permitiendo de esta manera

hacer uso eficiente de los recursos en el desarrollo de proyectos de control electrónico, obteniendo los mejores resultados con mínimo esfuerzo.

Una vez que se conocen los procedimientos de programación de un PIC específico se puede emprender la tarea de programar cualquier otro tipo de PIC sin mayor dificultad. La innovación contenida en este libro consiste en la explicación de los métodos de aplicación de las nuevas herramientas de programación en la solución de problemas con microcontroladores PIC.

3.1 DEFINICION

Los circuitos integrados programables (Programmable Integrated Circuits = PIC) son componentes sumamente útiles en la Electrónica de Consumo. Aún cuando son conocidos desde hace más de veinte años, existen en la actualidad nuevos tipos que cumplen con una serie de requisitos y características sumamente útiles. Como una primera aproximación podemos definir a un PIC como “un chip que me permite obtener un circuito integrado a mi medida”, es decir puedo hacer que el PIC se comporte como un procesador de luminancia o un temporizador o cualquier otro sistema mediante un programa que le grabo en una memoria ROM interna.

3.2 CARACTERÍSTICAS ESPECIALES DEL MICROCONTROLADOR:

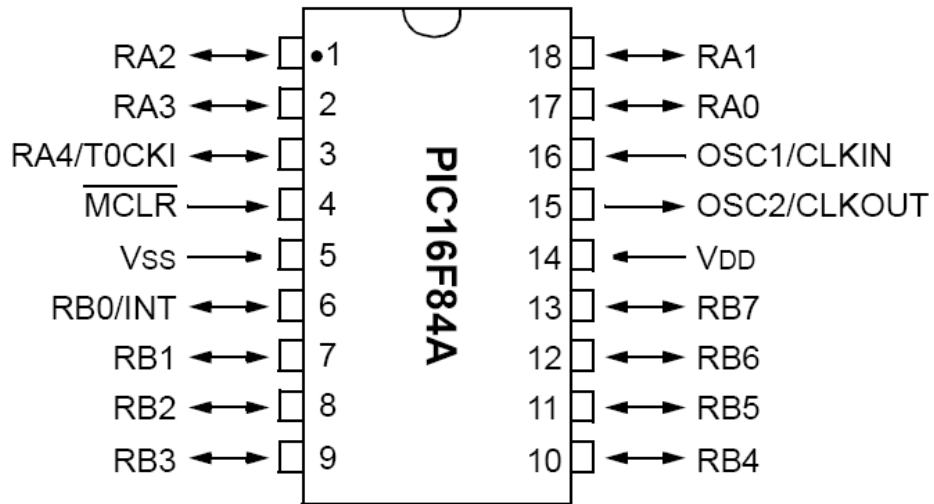
El fabricante en su hoja de especificaciones afirma que la memoria de programa se puede grabar y borrar unas 100.000 veces, la memoria de datos EEPROM 1.000.000 de veces y sus datos permanecen almacenados por más de 40 años. Otras características son la Programación en Serie en el Circuito (ICSP™) que requiere un total de 5 pines, acceso del procesador a la memoria de programa, programación en bajo voltaje, depuración en el circuito por medio de 2 pines, temporizador de vigilancia extendido (período programable desde 1 ms hasta 268 s) y un rango amplio de voltaje de operación (desde 2V a 5,5V)

3.3 CONFORMACIÓN Y DESCRIPCIÓN DE PINES DE LOS CIRCUITO INTEGRADOS 16F84A, 16F88; 16F628 Y 16F877.

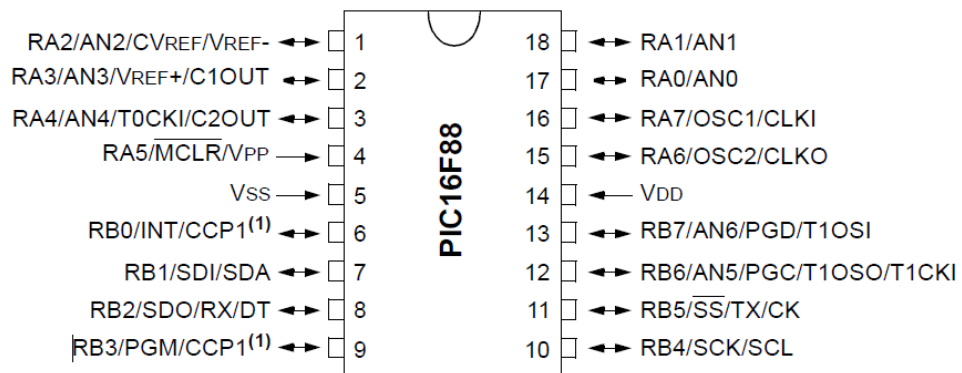
Nombre	Tipo	Descripción
OSC1/CLKIN	I	Entrada del oscilador a cristal/Entrada de la fuente de reloj externa
OSC2/CLKOUT	O	Salida del oscilador a cristal. En el modo RC, es una salida con una frecuencia de $\frac{1}{4}$ OSC1
MCLR	I/P	Reset/Entrada del voltaje de programación.
RA0	I/O	Puerto A bidireccional, bit 0
RA1	I/O	Puerto A bidireccional, bit 1
RA2	I/O	Puerto A bidireccional, bit 2
RA3	I/O	Puerto A bidireccional, bit 3
RA4/T0CKI	I/O	También se utiliza para la entrada de reloj para el TMR0
RB0/INT	I/O	Puerto B bidireccional, bit 0 Puede seleccionarse para entrada de interrupción externa
RB1	I/O	Puerto B bidireccional, bit 1
RB2	I/O	Puerto B bidireccional, bit 2
RB3	I/O	Puerto B bidireccional, bit 3
RB4	I/O	Puerto B bidireccional, bit 4 Interrupción por cambio de estado
RB5	I/O	Puerto B bidireccional, bit 5 Interrupción por cambio de estado
RB6	I/O	Puerto B bidireccional, bit 6 Interrupción por cambio de estado
RB7	I/O	Puerto B bidireccional, bit 7 Interrupción por cambio de estado
Vss	P	Tierra de referencia
Vdd	P	Alimentación

3.4 DISPOSICIÓN Y DESCRIPCIÓN DE PATILLAS

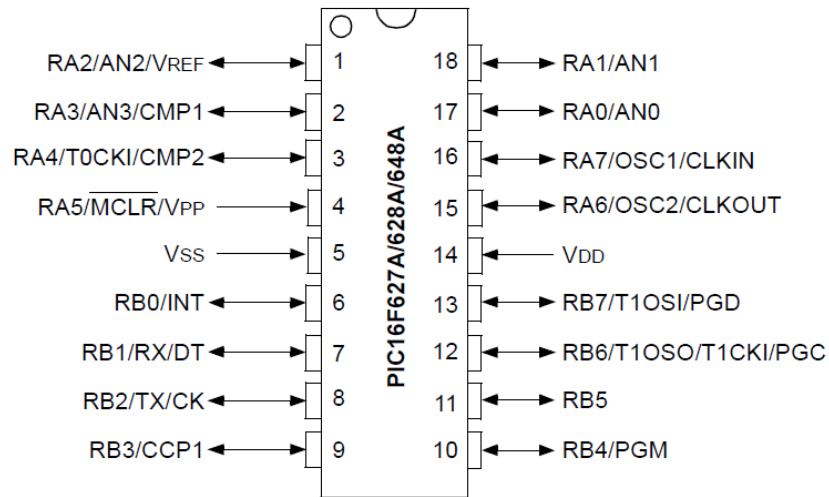
Disposición de patillas del PIC 16F84A para encapsulado DIL 18:



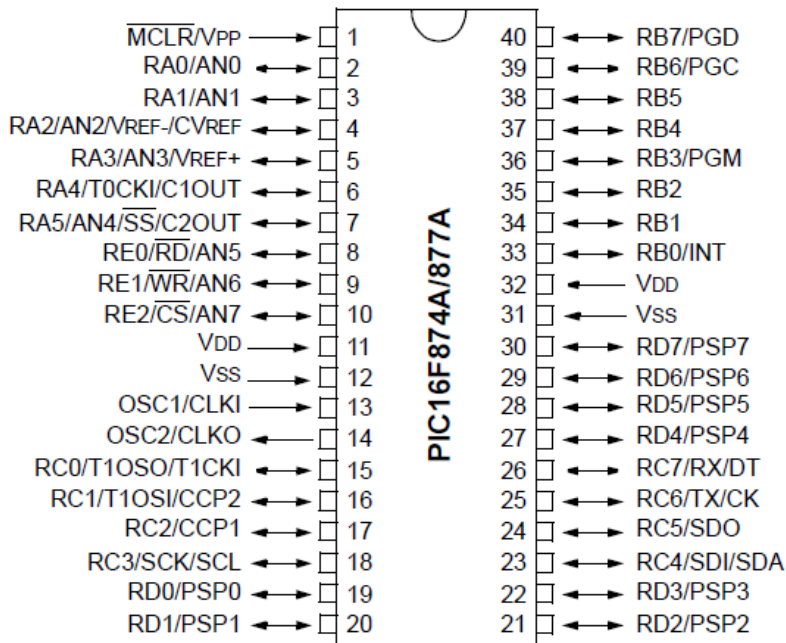
Disposición de patillas del PIC 16F88 para encapsulado DIP 18:



Disposición de patillas del PIC 16F628A para encapsulado DIP 18:



Disposición de patillas del PIC 16F877A para encapsulado DIL 40:



3.5 PIC16F84A vs. PIC16F88, PIC16F628A y PIC16F877A

Para colocar frente a frente a estos microcontroladores y poder sacar una conclusión acerca de sus características, lo mejor es presentar la información que Microchip publica en su página de Internet, donde se tabulan todos los datos relativos a los productos disponibles actualmente. La información presentada es la siguiente:

Producto	Estado	Precio	Tipo de	Memoria de	
Memoria de datos	Memoria	programa	EEPROM	kWords	
PIC16F84A	En producción	\$3,11	Flash	1	64
PIC16F88	En producción	\$2,20	Flash	4	256
PIC16F628A	En producción	\$1,47	Flash	2	128
PIC16F877A	En producción	\$4,26	Flash	8	256

Características de los PICs 16F84A, 16F88, 16F628A y 16F877A

RAM de canales interno	No. de pines A/D	No. total de E/S	Max. velocidad pines	Oscilador	No. MHz
68	13	18	20	NO	0
368	16	IR	20	Hasta 8MHz	7
224	16	IR	20	4MHz	0
368	33	40	20	NO	8

Características de los PICs 16F84A, 16F88, 16F628Ay 16F877A (continuación)

Comunicación Encapsulado digital	Timers	Rango de temperaturaoperación	Rango de voltaje de	
NO	1 de 8 bit	-40 a 85°C	2 V a 6 V	I8PDIP
I -A/E/US A RT 1 - SSP(SPI/I2C)	2 de 8 bit 1 de 16 bit	-40a125°C	2 V a 5,5V	ISPDP
1 -A/H/USART	2 de 8 bit 1 de 16 bit	-40a125°C	2 V a 5.5V	18PDIP
1 -A/K/USART 1 - MSSP(SPI/I2C)	2 de 8 bit 1 de 16 bit	-40a125°C	2 Va5,5V	40 PDIP (0,6 in)

Características de los PICs 16F84A, 16F88, 16F628A y 16F877A (continuación)

Los PICs 16F88 y 16F628A pertenecen a la gama media y pueden ser aplicados en circuitos electrónicos de propósito general. **Se parecen al PIC16F84A en el número de pines y su distribución física.** Como ventajas relevantes se pueden destacar las siguientes: menor precio , mayor capacidad de memoria de programa y datos, mayor disponibilidad de pines E/S, oscilador interno hasta 8 MHz (16F88) y 4MHz (16F628A), convertidor A/D (16F88), módulos de comunicación serie tres temporizadores. Todas estas características los hacen completamente superiores al PIC16F84A, y por estas razones fueron seleccionados como modelos para la escritura de 16F88 es el más completo y avanzado de los tres en cuestión, y su precio siendo un poco mayor que el de 16F628 A, aún es menor que el del 16F84A.

El PIC16F877A incorpora funciones similares a los otros tres microprocesadores, se diferencia fundamentalmente en que es un

microcontrolador de 40 pines y 0.6 pulgadas de ancho, tiene una memoria de programa de 8k que permite realizar aplicaciones muy extensas y gran cantidad E/S (33 en total); estas características hacen que su precio sea un poco más elevado, aun así sigue siendo muy accesible. Una desventaja de este PIC es que no posee oscilador interno, lo que hace necesario la instalación de un cristal externo para su operación.

3.6 PUERTOS DIGITALES

Las aplicaciones que tienen los puertos digitales son muy variadas, debido a la amplia gama de sensores y actuadores de dos estados que existen en la actualidad para el diseño de sistemas electrónicos de control, por ello es imprescindible conocer los detalles de su configuración, alcances y limitaciones. En este capítulo se explica la programación en lenguaje C de los puertos de los microcontroladores 16F88, 16F628Ay 16F877A operando en modo digital.

3.7 PRINCIPALES CARACTERÍSTICAS

Los pines E/S (entrada/salida) de propósito general pueden considerarse como los periféricos más simples. Permiten que el PIC monitoree y controle otros dispositivos. Para añadir flexibilidad y funcionalidad al dispositivo, algunos de los pines de los puertos pueden cumplir funciones de diferentes periféricos. En general, cuando se habilita un periférico, ese pin no puede ser utilizado como un pin E/S de propósito general. El sentido de los datos

(entrada o salida) se controla por el registro TRISx. El registro PORTx es el receptor (latch) de los datos de salida. Cuando se lee un puerto, el PIC lee los niveles presentes en los pines (no en el latch). Esto significa que debe ponerse mucha atención a los comandos leer-modificar-escribir (*read-modify-write* RMW) aplicados a los puertos y al cambiar el sentido de circulación en un pin de entrada a salida.

Un pin configurado como salida y que se encuentre en alto o bajo no debe ser polarizado externamente al mismo tiempo con el propósito de cambiar su nivel lógico. Las altas corrientes de salida que resultan pueden dañar el chip.

Se recomienda que al inicializar un puerto, el registro PORTx se inicialice primero, y luego el registro TRISx, ya que los valores del latch no están definidos al encendido.

3.7.1 PUERTO A

PIC16F88

El puerto A es un puerto bidireccional de 8 bits que puede ser programado a través de 3 registros: PORTA (datos de entrada/salida), TRISA (sentido de circulación de los datos) y ANSEL (selección de modo analógico/digital). Al programar un bit de TRISA con un valor de 1 se consigue que el pin correspondiente del puerto A trabaje como entrada (es decir, coloca el driver de salida en alta impedancia). Al programar un bit de TRISA con 0 se logra

que el pin correspondiente del puerto A opere como salida (es decir, coloca el contenido del latch de salida en el pin seleccionado). La lectura del registro PORTA, lee el estado de los pines, mientras que una escritura en él, escribe en el latch del puerto.

El fabricante recomienda la siguiente secuencia de inicialización de este puerto: escribir 0 en el registro PORTA, escribir 0 en el registro ANSEL para configurar los pines AN<6:0> como E/S digital, escribir en el registro TRISA los bits de sentido de circulación (1=entrada, 0=salida). En la hoja de especificaciones se indica que el pin RA5 sólo puede actuar como entrada, por lo que este detalle debe estar presente a la hora de diseñar una aplicación.

En la tabla se indican las tecnologías de entrada y salida de cada pin de este puerto. Para tener a disposición los pines RA7:RA6 es necesario configurar el microcontrolador para que utilice el oscilador interno, y para disponer del pin RA5 se lo debe configurar como E/S digital (deshabilitar la función *Master Clear Reset*).

Pin	Entrada	Salida
RA0	TTL	CMOS
RA1	TTL	CMOS
RA2	TTL	CMOS
RA3	TTL	CMOS
RA4	ST	CMOS
RA5	ST	No disponible
RA6	ST	CMOS
RA7	ST	CMOS

Tecnologías E/S del puerto A del PIC16F88 en modo digital

PIC16F628A

El puerto A es un puerto bidireccional de 8 bits que puede ser programado a través de 4 registros: PORTA (datos de entrada/salida), TRISA (sentido de circulación de los datos), CMCON (control del comparador) y VRCON (control de la referencia de voltaje). Al programar un bit de TRISA con un valor de 1 se consigue que el pin correspondiente del puerto A trabaje como entrada (es decir, coloca el driver de salida en alta impedancia). Al programar un bit de TRISA con 0 se logra que el pin correspondiente del puerto A opere como salida (es decir, coloca el contenido del latch de salida en el pin seleccionado). La lectura del registro PORTA lee el estado de los pines, mientras que una escritura en él, escribe en el latch del puerto.

El fabricante recomienda la siguiente secuencia de inicialización de este puerto: escribir 0 en el registro PORTA, escribir 0x07 en el registro CMCON para configurar los pines RA<3:0> como E/S digital, escribir en el registro

TRISA los bits de sentido de circulación (1=entrada, 0=salida). En la hoja de especificaciones se indica que el pin RAS sólo puede actuar como entrada, por lo que este detalle debe estar presente a la hora de diseñar una aplicación.

El registro TRISA (y también TRISB) tiene todos sus bits en 1 después de cualquier reset y *no* cambia al despertar por WDT o interrupción, por lo tanto el puerto A (y también el puerto B) están configurados como entradas inicialmente.

En la se indican las tecnologías de entrada y salida de cada pin de este puerto. Para tener a disposición los pines RA7:RA6 es necesario configurar el microcontrolador para que utilice el oscilador interno, y para disponer del pin RA5 se lo debe configurar como E/S digital (deshabilitar la función *Master Clear Reset*).

Pin	Entrada	Salida
RA0	ST	CMOS
RA1	ST	CMOS
RA2	ST	CMOS
RAS	ST	CMOS
RA4	ST	Drenaje abierto
RA5	ST	No disponible
RA6	ST	CMOS
RA7	ST	CMOS

Tecnologías E/S del puerto A del PIC16F628A en modo digital

3.7.2 CARACTERÍSTICAS DE LAS TECNOLOGÍAS DE ENTRADA/SALIDA

El comparador *Schmitt Trigger* (ST) se caracteriza porque su salida estará siempre en uno de los dos niveles binarios posibles (1 o 0), sin importar el nivel de voltaje de entrada (tomando en cuenta los límites máximos), no existe una región de indeterminación como ocurre en TTL. Esta característica se emplea para conformar ondas irregulares y obtener ondas digitales puras. También tiene alta inmunidad al ruido, ya que se requiere una variación importante del voltaje de entrada para que se produzca una transición en la salida.

Los transistores MOS se comportan como interruptores controlados por voltaje, mientras que los transistores bipolares empleados en la tecnología TTL se comportan como interruptores controlados por corriente. Las familias Lógicas MOS; son especialmente susceptibles a daños por carga electrostática. Esto es consecuencia directa de la alta impedancia de entrada. Una pequeña carga electrostática que circule por estas altas impedancias puede dar origen a voltajes peligrosos. Los CMOS están protegidos contra daño por carga estática mediante la inclusión en sus entradas de diodos zéner de protección, diseñados para conducir y limitar la magnitud del voltaje de entrada a niveles muy inferiores a los necesarios para provocar daño. Si bien los zéner por lo general cumplen con su finalidad, algunas veces no comienzan a conducir con la rapidez necesaria para evitar que el circuito integrado sufra daños. Por consiguiente, sigue

siendo buena idea observar las precauciones de manejo empleadas con los dispositivos sensibles a las descargas electrostáticas (ESD). La gran ventaja de los CMOS es que utilizan solamente una fracción de la potencia que se necesita para la serie TTL, adaptándose de una forma ideal a aplicaciones que utilizan la potencia de una batería o con soporte en una batería, El inconveniente de la familia CMOS es que normalmente es más lenta que la familia TTL,

3.7.3 PUERTO B

PIC16F88

Este es un puerto bidireccional de 8 bits. El registro de sentido de datos es TRISB. Al programar un bit de TRISB con un valor de 1 se consigue que el pin correspondiente del puerto B trabaje como entrada (es decir, coloca el driver de salida en alta Impedancia). Al programar un bit de TRISB con 0 se logra que el pin correspondiente del puerto B opere como salida (es decir, coloca el contenido del latch de salida en el pin seleccionado). Cada pin de este puerto tiene *un. pull up* interno (resistencia conectada a V_{DD}). El bit #RBPU del registro OPTION_REG puede activar o desactivar esta función. Cuando el puerto se configura como salida *las pull ups* se desactivan automáticamente y también cada vez que se enciende el PIC (*Power On Reset* POR). Este puerto se puede programar por medio de 4 registros PORTB, TRISB, OPTION_REG (activar/desactivar las pull ups) y ANSEL. En

la tabla. se indican las tecnologías de entrada y salida de cada pin de este puerto.

Pin	Entrada	Salida
RB0	TTL	TTL
RB1	TTL	CMOS
RB2	TTL	CMOS
RB3	TTL	TTL
RB4	TTL	CMOS
RB5	TTL	TTL
RB6	TTL	TTL
RB7	TTL	TTL

. Tecnologías E/S del puerto B del PIC16F88 en modo digital

Cuando se habiliten funciones de periféricos de este puerto, se debe tener cuidado al definir los bits del registro TRISB. Algunos periféricos modifican estos bits para hacer que algunos pines operen como salida y otros como entrada. Esta modificación tiene vigencia mientras el periférico está habilitado, por lo tanto se debería evitar el uso de instrucciones RMW con el registro TRISB.

PIC16F628A

Este es un puerto bidireccional de 8 bits. El registro de sentido de datos es TRISB. Al programar un bit de TRISB con un valor de 1 .se consigue que el pin correspondiente del puerto B trabaje como entrada (es decir, coloca el driver de salida en alta impedancia). Al programar un bit de TRISB con 0 se logra que el pin correspondiente del puerto B opere como salida (es decir, coloca el contenido del latch de salida en el pin seleccionado). Cada pin de

este puerto tiene un *pull up* interno (200 uA aproximadamente). El bit #RBPU del registro OPTION_REG puede activar o desactivar esta función. Cuando el puerto se configura como salida las *pull ups* se desactivan automáticamente y también cada vez que se enciende el PIC (*Power On - Reset POR*). Este puerto se puede programar por medio de 3 registros: PORTE, TRISB, OPTION_REG (activar/desactivar las *pull ups*). En la tabla se indican las tecnologías de entrada y salida de cada pin de este puerto.

Pin	Entrada	Salida
RBO	TTL,	CMOS
RB1	TTL	CMOS
RB2	TTL	CMOS
KB3	TTL	CMOS
RB4	TTL	CMOS
RB5	TIL	CMOS
RB6	TTL	CMOS
RB7	TTL	CMOS

Tecnologías E/S del puerto B del PIC16F628A en modo digital

Cuando se habiliten funciones de periféricos de este puerto, se debe tener cuidado al definir los bits del registro TRISB. Algunos periféricos modifican estos bits para hacer que algunos pines operen como salida y otros como entrada. Esta modificación tiene vigencia mientras el periférico está habilitado.

3.7.4 PUERTOS A, B, C, D y E del PIC16F877A

El PIC16F877A posee cinco puertos con un total de 33 pines, lo que le brinda a este microcontrolador una gran ventaja con respecto a otros PICs que tienen un número de pines relativamente reducido. En la tabla se pueden apreciar en detalle las características de estos puertos.

Puerto	No. de bits	Tecnología	Registros	Estado inicial en POR
A	6	TTL excepto RA4. ST para RA4 (como salida es drenaje abierto).	PORTA TRISA ADCON1	Entradas analógicas.
B	8	TTL	PORTB TRISB	Entradas digitales.
C	8	ST	PORTC TRISC	Entradas digitales.
D	8	ST	PORTD TRISD	Entradas digitales.
E	3	ST	PORTE TRISE ADCON1	Entradas analógicas.

La configuración de los puertos A y E se realiza de forma similar, programando el registro ADCON1 con un valor de 0x06 para que los pines RA<5:0> y RE<2:0> operen como E/S digital, debido a que inicialmente estos pines están configurados como entradas analógicas.

3.8 VALORES MÁXIMOS ABSOLUTOS

En la hoja de especificaciones del fabricante se indican los valores máximos para las intensidades de corriente de los pines del microcontrolador, los cuales deben ser tomados en cuenta para los diseños de las aplicaciones prácticas. Las líneas de los puertos son capaces de entregar niveles TTL cuando el voltaje de alimentación aplicado en V_{DD} es de 5 V.

Pines	Máxima corriente de entrada (mA)	Máxima corriente de salida (mA)
Cualquier pin E/S	25	25
Puerto A en total	100	100
Puerto B en total	100	100
Pin V _{nu}	20 (16F88) 250 (16F628A)	No se aplica a este pin
Pin V _{ss}	No se aplica a este pin	200 (16F88) 300 (16F628A)

Corrientes máximas permitidas en los PICs 16F88y 16F628A

Pines (mA)	Máxima corriente de entrada (mA)	Máxima corriente de salida (mA)
Cualquier pin E/S	25	25
Puertos A, B y E (combinados)	200	200
Puertos C y D (combinados)	200	200
Pin V _{DD}	250	No se aplica a este pin
Pin V _{ss}	No se aplica a este pin	300

Corrientes máximas permitidas en el PIC16F877A

CAPITULO IV

4.1 REQUERIMIENTOS BÁSICOS PARA PROGRAMAR UN PIC.

Los recursos mínimos que requiere el programador de PICs son los siguientes.

Una PC

Mínimo Se recomienda

Pentium 3 o 4

Microsoft Windows XP Internet

500 MB de RAM Explorer 6.0

CD-ROM drive

Un Editor y un Compilador.

Mikro C contiene ambos, el editor y el compilador

Puede bajarlos de la dirección de Microchip www.microchip.com

Por favor instálelos en su PC lo mas pronto posible.

Un Programador

El más económico lo ofrece PICmicro Programador de PIC's para puerto USB Debería adquirirlo e instalar el software: PICKit 2 v2.61, Lv18PICFLASH

Un protoboard para probar sus programas.

4.2 COMO PROGRAMAR UN PIC

Un PICmicro es un circuito integrado programable. Microchip, su fabricante dice: Programmable Integrated Circuit. Programmable quiere decir que se puede planificar la manera como va a funcionar, que se puede adaptar a nuestras necesidades. En otras palabras que el integrado es capaz de modificar su comportamiento en función de una serie de instrucciones que es posible comunicarle. Toda esta actividad : “Programar un PIC”, se puede dividir en cuatro pasos:

EDITAR

COMPILAR

QUEMAR EL PIC

PROBAR EL PROGRAMA

4.2.1 EDITAR

Editar es escribir el programa, es hacer una lista de instrucciones en un lenguaje que nos permita indicarle al PIC lo que deseamos que haga. Existen varios lenguajes como: Ensamblador, Basic, C, etc. Todos ellos pretenden acercarse a nuestra manera de pensar y de hablar. Sin embargo los PIC no conocen mas que unos y ceros. Por eso es necesario el siguiente paso.

4.2.2 COMPILAR

Compilar es traducir el programa al lenguaje de máquina que ¡ Si ! “entiende” el PIC. Para realizar esta traducción hacemos uso de un software que transforma el “Programa Fuente”, aquel que editamos en el paso 1 en otro que si podemos comunicarle al PIC.

4.2.3 QUEMAR EL PIC

En este paso se grava el programa en el PIC. Mediante una tarjeta electrónica y un poco software se pasa el programa compilado de la PC al PIC. Son solamente unos cuantos Cliks y listo. Es necesario hacer una aclaración en este momento. Frecuentemente le llamamos Programador de PIC a la tarjeta electrónica que transfiere el programa compilado de la PC al PIC. Esta bien mientras entendamos que este aparato no va ha pensar por nosotros y que es incapaz de programar instrucciones por sí mismo.

4.2.4 PROBAR EL PROGRAMA

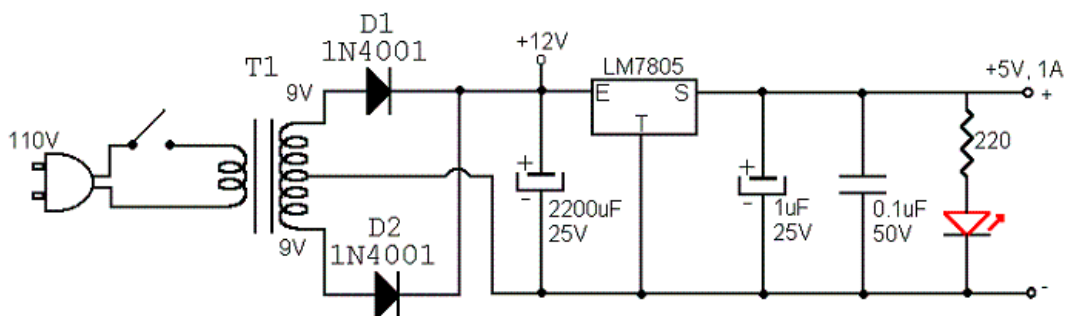
Bueno en este paso se trata de verificar el funcionamiento del programa. Se trata de comprobar que el PIC se comporta como lo programamos. Si todo salió bien, pues fantástico y si no comenzamos de nuevo en Editar Para realizar esta actividad podemos hacer uso de un Protoboard, alambrar los Led's o botones, instalar la fuente, poner el reloj , etc. etc. Pero como no se trata de aprender a armar circuitos en Protos sino de aprender a programar

Pics es mejor hacer uso de una tarjeta "Proyecto" que ya tenga todo esto y este lista para ser usada.

4.3 CIRCUITOS Y COMPONENTES BÁSICOS PARA ARMAR Y HACER FUNCIONAR UN PIC

4.3.1 Fuente de alimentación de 5 voltios

Es un instrumento útil en cualquier mesa de trabajo cuando se trata de circuitos electrónicos digitales que tengan integrados de la familia TTL, se requiere de una fuente regulada de voltaje de 5 voltios. Una fuente regulada entrega en sus bornes de salida un voltaje constante, independiente de las variaciones en la línea de alimentación y en la carga.



En este proyecto construiremos una fuente regulada de 5 voltios con capacidad de alimentar una o varias cargas que consuman hasta 1 amperio. En la figura se muestra el diagrama respectivo Existen, además, otros dos condensadores que sirven para completar el filtrado del voltaje, eliminando señales residuales de alta frecuencia.

4.3.2 CRISTALES DE QUARZO

Estos dispositivos están formados por una fina lámina de cuarzo situada entre dos electrodos. Como es sabido, el cuarzo, también llamado cristal de roca, es un mineral compuesto por silicio y oxígeno, (óxido anhidro de silicio, bióxido de silicio o anhídrido silícico, SiO_2) cuyos cristales tienen forma de prisma hexagonal terminado por dos romboedros que parecen una bipirámide hexagonal. El cuarzo es el mineral más difundido en la corteza terrestre, bien en forma de cristales o formando parte otras rocas, como el granito (cuarzo, feldespato y mica)



CAPITULO V

SOFTWARE Y HARDWARE NECESARIO Y CONVENIENTE PARA PROGRAMAR Y GRABAR UN PIC

5.1 Tipos de software y hardware disponibles.

SOFTWARE.

mikroC™ PRO for PIC (mikroC™) 3:2

IC-Prog

PICkit2 v2.61,

IS1S® de PROTEUS®,

NOPPP en ambiente DOS.

MPLAB, para lenguaje ensamblador.

HARDWARE

Programador AN589, para IC - prog

Programador PTCKit2 Clone para el puerto USB,

Programador NOPPP, con doble fuente, para puerto DB25, en la actualidad descontinuado, por sus limitadas prestaciones.

Para el desarrollo de los ejemplos contenidos de esta tesis se han seleccionado las siguientes herramientas **YA QUE SON TOTALMENTE COMPATIBLES CON LAS TECNOLOGÍAS ACTUALES:**

5.1.1 SOFTWARE:

mikroC™ PRO for PIC (mikroC™) 3:2, ambiente de desarrollo integrado en lenguaje C, que se puede descargar de la página del fabricante, la versión gratuita de mikroC™ es completamente funcional, con todas las librerías. ejemplos y archivos de ayuda incluidos. La única limitación de Inversión gratuita es que no se pueden generar archivos ejecutables (*.hex) de más de 2k palabras de programa. Aun así, esto permite desarrollar todas las aplicaciones prácticas de variada complejidad de este libro.

PICkit2 v2.61, software de grabación de microcontroladores PIC para el programador USB PICkit2 Clone.

IS1S® de PROTEUS®, software de simulación de microcontroladores PIC, en el que se han probado con éxito todos los ejemplos resueltos en esta tesis.

5.1.2 HARDWARE.-

Programador PTCKit2 Clone para el puerto USB,

5.2 PROGRAMANDO LOS CIRCUITOS INTEGRADOS 16F88, 16F628, 16F877.

Se emplea el oscilador interno (oscilador primario) integrado dentro del PIC16F88 para simplificar el diseño del Hardware, así como disponer de dos pines más (RA6 y RA7) como E/S digital. Para seleccionar cualquiera de las 8 frecuencias disponibles se emplea el registro de control del oscilador OSCCON. Al seleccionar una frecuencia se debe esperar a que el bit IOFS de este registro sea igual a 1, indicando de esta forma que el oscilador se ha estabilizado. La frecuencia se selecciona por medio de los bits IRCF.

El temporizador de encendido PWRT se habilita para mantener al PIC en reset hasta que la fuente de alimentación se estabilice. En la configuración básica no se habilitan ni el encendido de doble velocidad, ni el oscilador de seguridad (palabra CONFIG2).

En caso de habilitar el reset por desvanecimiento (*Brown-out Reset BOR*) se debe conectar un capacitor de desacoplo de 100 nF (0.1 uF) lo más cerca posible de los pines de alimentación del PIC ($V_{DD}-V_{SS}$), para evitar que se produzca un reset indeseado cuando cualquiera de las salidas del microcontrolador cambia de estado.

La configuración inicial básica del PIC16F88 es la siguiente:

CONFIG1 {Palabra de configuración 1):

Programación en bajo voltaje deshabilitada (LVP=0).

Reset maestro deshabilitado (el pin RA5 es E/S digital, MCLREK).

Temporizador de encendido habilitado (#PWRTEN=0).

Temporizador de vigilancia deshabilitado (WDTEN=0).

Oscilador interno con RA6 y RA7 como E/S digital (FOSC<2:0>=100).

CONF1G2 (Palabra de configuración 2):

Encendido de doble velocidad deshabilitado (IESO=0).

Oscilador de seguridad deshabilitado (FCMEN=0).

OSCCON (Registro de control del oscilador):

Selección de frecuencia del oscilador interno de acuerdo a los bits IRCF<2:0>.

Bit de indicación de estabilidad de la frecuencia del oscilador interno (IOIS=1: oscilador estable; IOFS=0: oscilador no estabilizado);.

Bits de selección de la fuente de reloj del sistema (SCS<1:0>=00: oscilador definido por los bits FOSC<2:0> de la palabra CONFIG 1)

PIC16F628A

Se emplea el oscilador interno de 4MHz (modo INTOSC) integrado dentro del PIC16F628A para simplificar el diseño del hardware, así como disponer de dos pines más (RA6 y RA7) como E/S digital. Para seleccionar una de las 2

frecuencias disponibles (4MHz por defecto, o 48kHz) se emplea el bit OSCF del registro de control de consumo PCON.

El temporizador de encendido PWRT se habilita para mantener al PIC en reset hasta que la fuente de alimentación se estabilice.

En caso de habilitar el reset por desvanecimiento (*Brown-out Reset* BOR) se debe conectar un capacitor de desacoplo de 100 nF (0.1 uF) lo más cerca posible de los pines de alimentación del PIC ($V_{DD}-V_{SS}$), para evitar que se produzca un reset indeseado cuando cualquiera de las salidas del microcontrolador cambia de estado.

La configuración inicial básica del PIC16F628A es la siguiente:

CONFIG (Palabra de configuración):

Programación en bajo voltaje deshabilitada (LVP=0).

Reset maestro deshabilitado (el pin RA5 es E/S digital, MCLRE=0).

Temporizador de encendido habilitado (#PWRTE=0).

Temporizador de vigilancia deshabilitado (WDTE=0).

Oscilador interno con RA6 y RA7 como E/S digital (FOSC<2:0>=100).

PIC16F877A

Se emplea el oscilador externo de 4MHz (modo HS). El temporizador de encendido PWRT se habilita para mantener al PIC en reset hasta que la fuente de alimentación se estabilice.

En caso de habilitar el reset por desvanecimiento (*Brown-out Reset BOR*) se debe conectar un capacitor de desacoplo de 100 nF (0.1 uF) lo más cerca posible de los pines de alimentación del PIC ($V_{DD}-V_{SS}$), para evitar que se produzca un reset indeseado cuando cualquiera de las salidas del microcontrolador cambia de estado.

La configuración inicial básica del PIC16F877A es la siguiente:

CONFIG (Palabra de configuración):

Programación en bajo voltaje deshabilitada (LVT=0).

Temporizador de encendido habilitado (#PWRTEN=0).

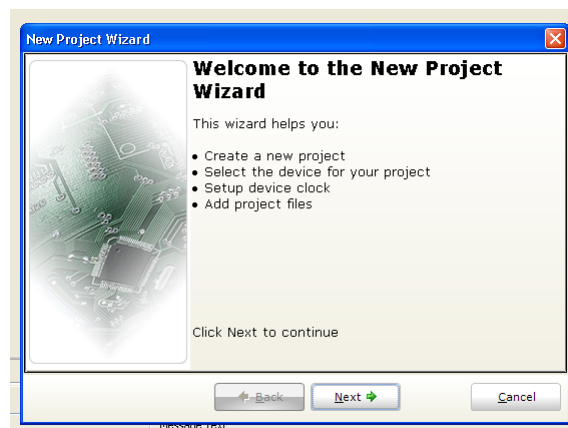
Temporizador de vigilancia deshabilitado (WDTEN=0).

Oscilador externo (HS).

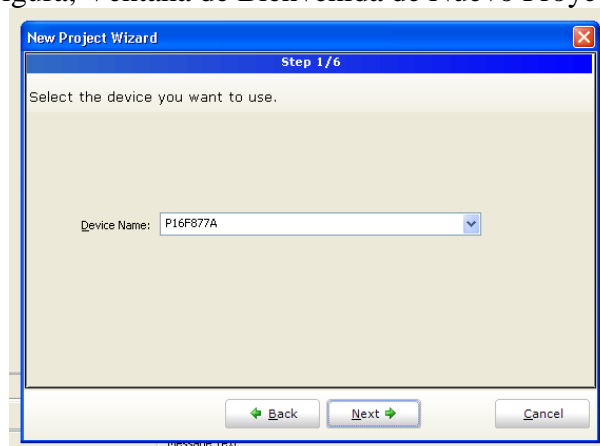
5.3 CONFIGURACIÓN INICIAL BÁSICA EN mikroC™

La configuración inicial es muy sencilla y consiste fundamentalmente en crear un nuevo proyecto a través del comando *Project>New Project* (figura), seleccionar el dispositivo (figura), la frecuencia de operación (figura) de mikroC™ (que debe coincidir con la frecuencia real de operación del PIC: 4MHz para el PIC16F88 y 4MHz para el PIC16F628A y el PIC16F877A), crear una nueva carpeta y definir un nombre de proyecto (figura) donde se

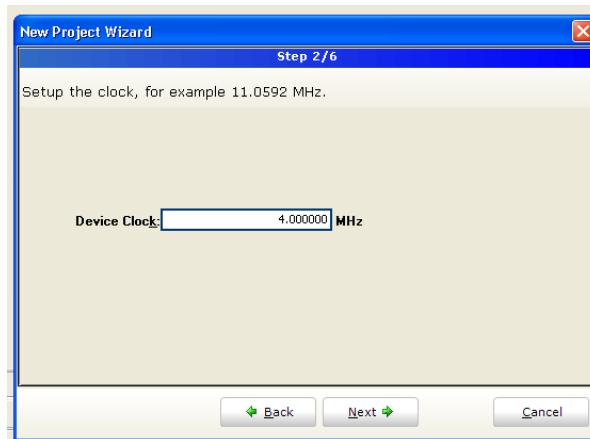
almacenarán todos los archivos relacionados con el programa a desarrollar (código fuente en lenguaje C, código fuente en lenguaje ensamblador, código de máquina a grabarse en el PIC, entre otros), añadir archivos disponibles que hayan sido creados previamente (figura), en este caso no se ha añadido ninguno ya que el proyecto consta de un solo archivo de código fuente, y por último finalizar (figura).



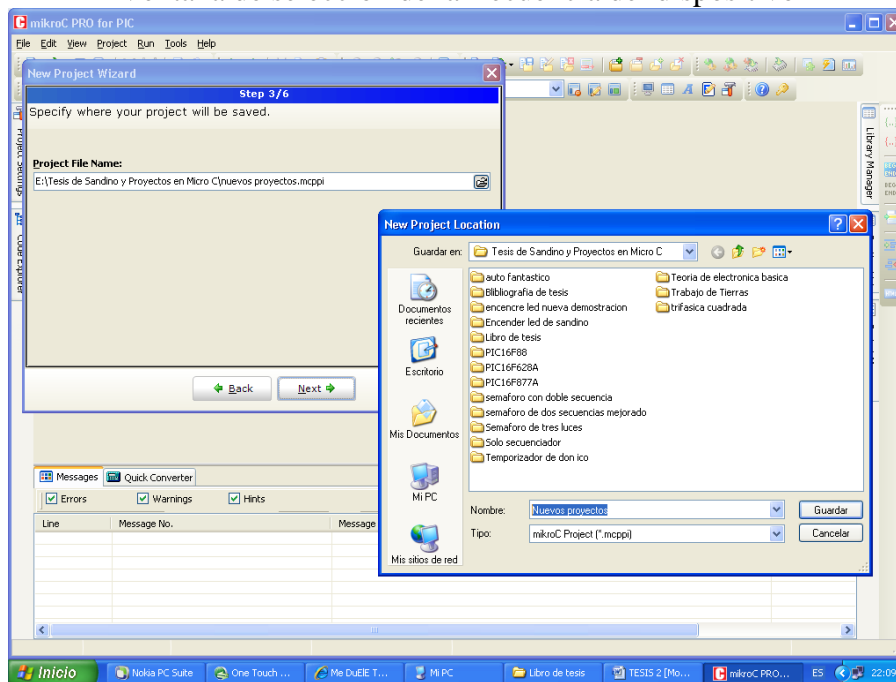
Figura; Ventana de Bienvenida de Nuevo Proyecto



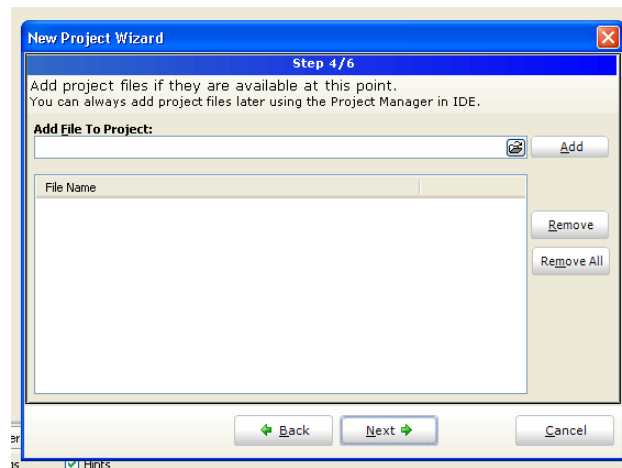
Ventana de selección del dispositivo



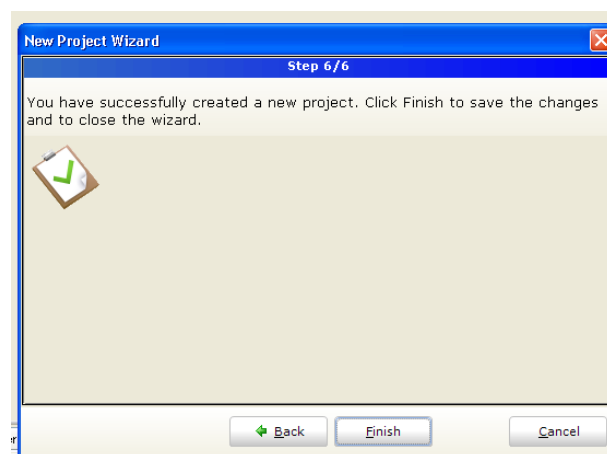
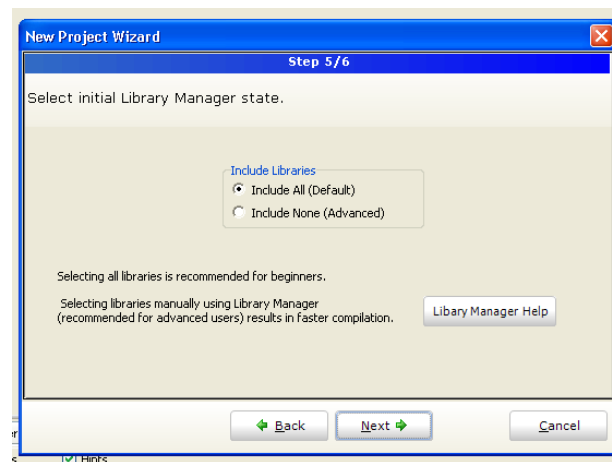
Ventana de selección de la frecuencia del dispositivo



Ventana de ubicación de los archivos del proyecto

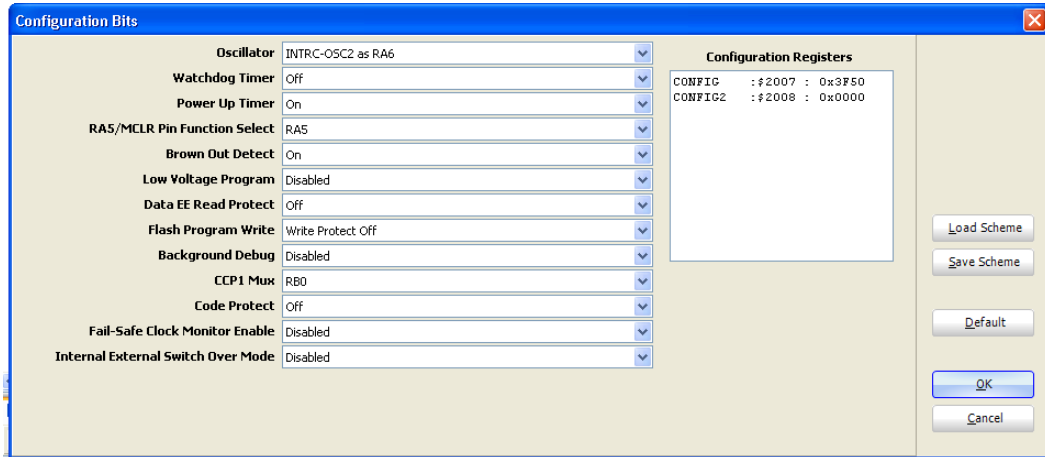


Ventana para adición de archivos al proyecto

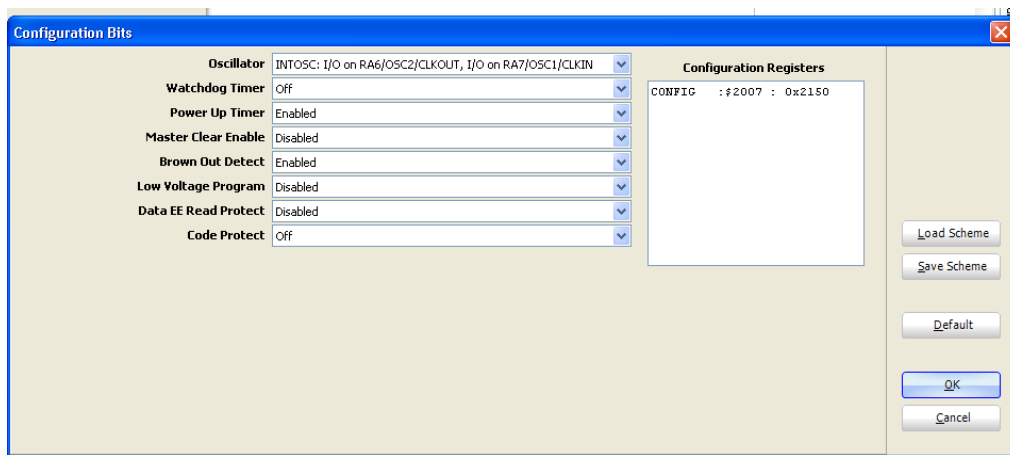


Ventana de finalización de creación de un nuevo proyecto

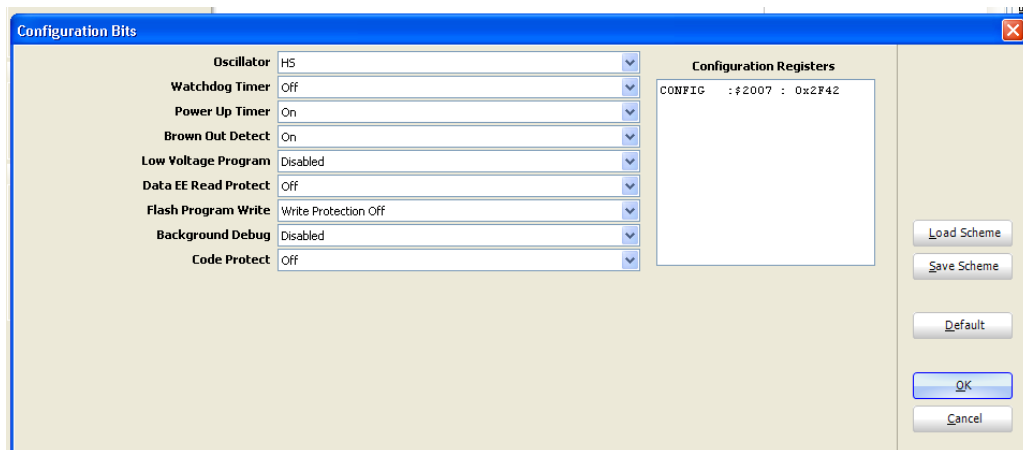
A continuación establecer los bits de configuración en las palabras CONFIG1 y CONFIG2 (PIC16F88) ó (CONFIG (PIC16F628A y PIC16F877A) a través del comando *Proyect > Edit Proyect*



Figura; ventana de bits de configuración del PIC 16F88



Ventana de bits de configuración del PIC16F628A



Ventana de bits de configuración del PIC16F877A

Por último escribir el código del programa (las primeras instrucciones permiten seleccionar la frecuencia del bloque oscilador interno del PIC16F88). Para ilustrar este proceso se ha escrito el sencillo programa **EncenderLED.c** para los PICs 16F88, 16F628A y 16F877A, descrito a continuación, que produce el parpadeo de un LED conectado en el pin RAO con intervalos de 500 ms (nótese la simplicidad del procedimiento si se compara con la misma tarea en lenguaje ensamblador).

```

EncenderLED.c
//PIC16F88
//Configuración inicial básica.
void main () {
    OSCCON=0x40;          //Oscilador interno a 1MHz
    while (OSCCON.IOFS= =0) .-//Esperar mientras al oscilador está inestable.
    PORTA = 0x00;        //Inicialización.
    ANSEL = 0x00 //Fines AN<6:0> como E/S digital.
    TRISA = 0x00 //Puerto A como salida.
    Wile (1)
    {
        RAO _ bit = 1          //Encender LED conectado en RAO(pin 17).
        Delay _ ms(500)
        RAO _ bit = 0; //Apagar LED conectado en RAO.
        Delay _ ms (500);
    }
}

```

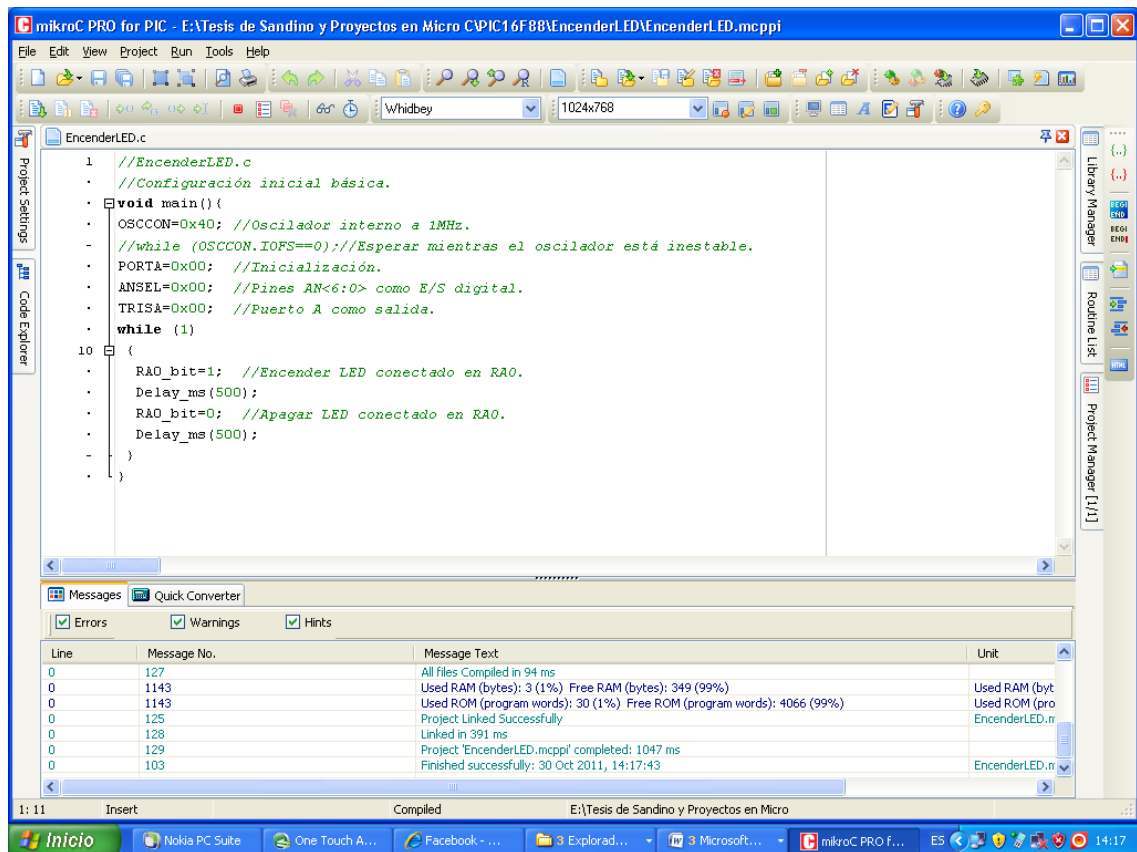
```
//EncenderLED.c
//PIC16F628A
//Configuración inicial básica.
void main () {
PORTA=0x00; //Inicialización.
CMCON = 0x07; //Pines RA<3:0> como E/S digital.
TRISA = 0x00; //Puerto A como salida.
while (1)
{

RAO_bit = 1; //Encender LED conectado en RA0 (pin 17)
Delay_ms(500)
RAO_bit = 0; //Apagar LED conectado en RA0
Delay_ms(500)
}
}

//Encender LED.c
//PIC16F877A
//Configuración inicial básica-
void main (){
PORTA 0x00 //Inicialización.
ADCON1 = 0x06 //Pines RA<5:0> como E/S digital.
TRISA = 0x00 //Puerto A como salida.
Wile (1)
{
RA0_bit = 1 // Encender LED conectado en RA0 (pin 2)
Delay_ms (500);
RA0_bit = 0; // Apagar LED conectado en RA0
Delay_ms (500);
}
}
```

Si se desea probar este ejemplo en el mundo real, se debe compilar por medio del comando

Project>Build (figura 1.10), El compilador generará nuevos archivos en la carpeta de proyecto creada anteriormente: EncenderLED.asm, EncenderLED.lst, EncenderLED.mcl, EncenderLED.hex y otros más. Este último se empleará para programar el microcontrolador.



Pantalla global de mikroC™. En la parte superior se muestra el código fuente en lenguaje C y en la parte inferior los resultados de la compilación (Messages)

MikroC™ es una poderosa herramienta de desarrollo para microcontroladores PIC. Está diseñado para proporcionar al usuario la solución más simple en el desarrollo de aplicaciones para sistemas embebidos, sin comprometer el rendimiento o el control.

Los PICs y el lenguaje C se acoplan perfectamente: los PICs son los microcontroladores de 8 bits más populares actualmente, empleados en una amplia variedad de aplicaciones, y el lenguaje C, muy apreciado por su eficiencia, es la elección natural para el desarrollo de sistemas embebidos.

mikroC™ se caracteriza por su IDE avanzado, compilador conforme a las normas ANSI, amplio conjunto de librerías para hardware, documentación muy fácil de comprender, y muchos ejemplos listos para su ejecución.

5.4 Características de mikroC™

Le permite al usuario desarrollar rápidamente aplicaciones complejas:

Escribir el código fuente en lenguaje C usando el Editor de Código incorporado

(Asistentes de Parámetros y Código, *Syntax Highlighting*, Auto Corrección, Plantillas de Código, y más),

Usar las librerías incluidas para agilizar drásticamente el proceso de desarrollo: adquisición de datos, memoria, *displays*, conversiones, comunicaciones, etc.

Monitorear la estructura del programa, variables, y funciones en el Explorador de Código.

Generar lenguaje ensamblador y archivos HEX estándar, compatibles con todos los programadores.

Inspeccionar el flujo del programa y depurarlo con el Simulador (en *Software*)

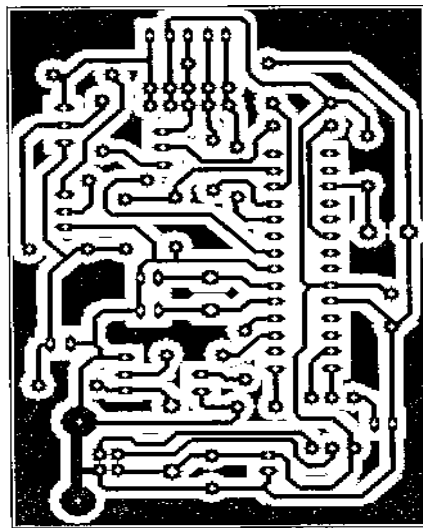
Obtener reportes detallados: mapas RAM y ROM, estadísticas de código, archivos de listado árbol de llamadas, y más.

5.5 PROGRAMADOR PICkit2 Clone PARA PUERTO USB

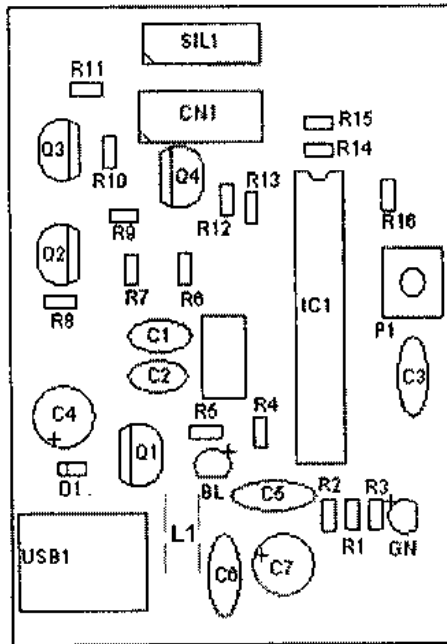
Este programador es una versión simplificada del original PICkit2 de Microchip y trabaja con el programa de aplicación PICkit2 v2.61 de Microchip, por lo tanto su buen funcionamiento está garantizado y asegurado. En la página de Microchip se informa que puede trabajar correctamente en Windows XP y Windows Vista. Adicionalmente ha sido probado exitosamente en el ambiente Windows 7 Home Premium. La lista de micro controladores PIC compatibles con este programador es muy extensa y se puede ver ingresando a *Help->ReadMeen* el programa de aplicación PICkit2 v2.61, o en nuestra página, ingresando a *Ofertas-> Más Detalles*.

5.5.1 CIRCUITO IMPRESO VISTO DESDE LA CARA DE COMPONENTES.

Esta es la imagen del circuito impreso para la construcción de este programador visto desde la cara de componentes (no desde la cara de la soldadura).

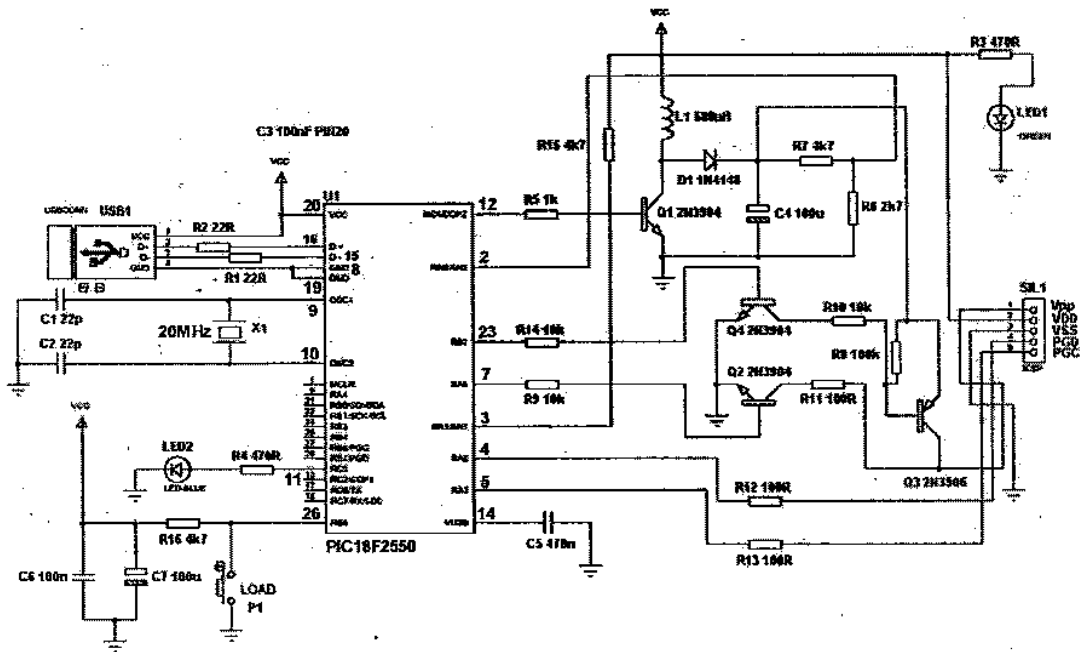


UBICACIÓN DE COMPONENTES



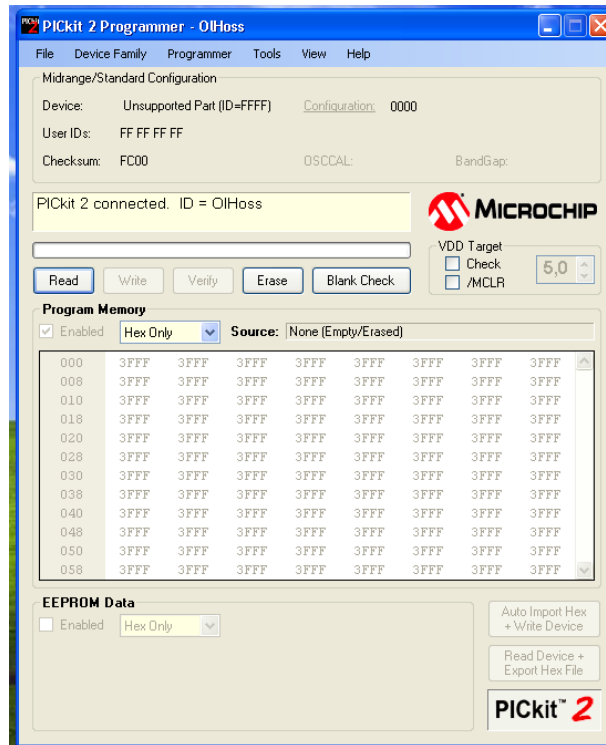
5.5.2 ESQUEMA ELÉCTRICO DEL PROGRAMADOR

Este esquema ha sido probado con el software PTckit2 v2.61 y el *firmware* (PK2V023200.hex) correspondiente a esta versión de la aplicación. El *firmware* es un programa ejecutable que debe ser grabado en el PIC18V2550 (lamentablemente eso requiere tener a disposición otro programador de PICs). El software de aplicación y el *firmware* se pueden descargar de la página web de Microchip (el *firmware* también se puede encontrar en la carpeta *PICkit 2 v2* de la instalación del programa de aplicación PiCkit2 v2.61). Los números de los componentes corresponden con la numeración de la placa de circuito impreso.



5.6 PROCEDIMIENTO DE PROGRAMACIÓN

- Conecte el dispositivo (EEPROM, PIC, dsPIC, etc.) al programador. La conexión se realiza a través de los pines VPP (voltaje de programación), VDD (voltaje de alimentación), VSS (referencia), PGD (datos) y PGC (reloj) del programador y los pines correspondientes del microcontrolador (ver la hoja de especificaciones para cada dispositivo particular para identificar los cinco pines de programación). Debe emplear un tablero de proyectos de inserción a presión.
- Conecte el cable USB al programador y a continuación el otro extremo del cable a un puerto USB del computador u ordenador.
- Inicie el programa de aplicación *"PICkit 2 v2.61"*

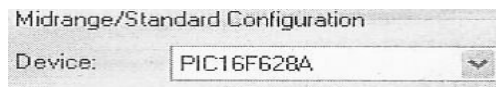


El programador será detectado automáticamente, al igual que el dispositivo a ser programado (.siempre y cuando pertenezca a una de las familias que soporten autodetección). Para que la autodetección del dispositivo tenga efecto, el comando *Programmer->Manual DeviceSelect* debe estar desactivado. Para el PIC16F628A se verá el siguiente mensaje:

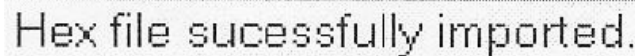
```
PICkit 2 found and connected.
[Parts in this family are not auto-detect.]
```

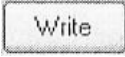
Si el dispositivo no soporta autodetección, debe ser seleccionado manualmente, para lo cual el comando *Programmer-^ Manual DeviceSelect* debe estar activado. Por ejemplo, para el PIC16F628A se debe

seleccionar el comando *DeviceFamily->Midrange-> Standard*, y a continuación seleccionar el PIC de la lista desplegable *Device*:



- Con el comando *File->!ImportHex* abra el archivo ejecutable (*.hex) que va a ser grabado en el dispositivo. Deberá observar el siguiente mensaje:

A screenshot of a message box with a white background and a black border. The text inside the box reads "Hex file successfully imported." in a monospaced font. The message box has a standard Windows-style title bar and a close button in the top right corner.

- Programe el dispositivo haciendo clic en el botón  (*Write*). Espere hasta que aparezca el mensaje de programación exitosa:

- Desconecte el cable USB del computador y extraiga el dispositivo programado.

- Si va a programar otro dispositivo, conéctelo al programador, conecte nuevamente el cable USB al computador y luego seleccione el comando *Tools->CheckCommunication*. Repita los pasos 4 a 7.

Nota: Los dispositivos se encuentran agrupados por familias, así que si no logra encontrarlo en una de las familias vaya al menú *Device Family* búsquelo en las otras familias.

5.7 CONSIDERACIONES PRÁCTICAS

Las condiciones de funcionamiento en un ambiente real (no simulado) pueden ocasionar resultados inesperados e incontables dolores de cabeza, fundamentalmente debido al ruido eléctrico. Con el fin de disminuir al mínimo estos problemas de operación, se sugiere tomar muy en cuenta las siguientes recomendaciones y aplicarlas desde el inicio en el diseño de circuitos prácticos:

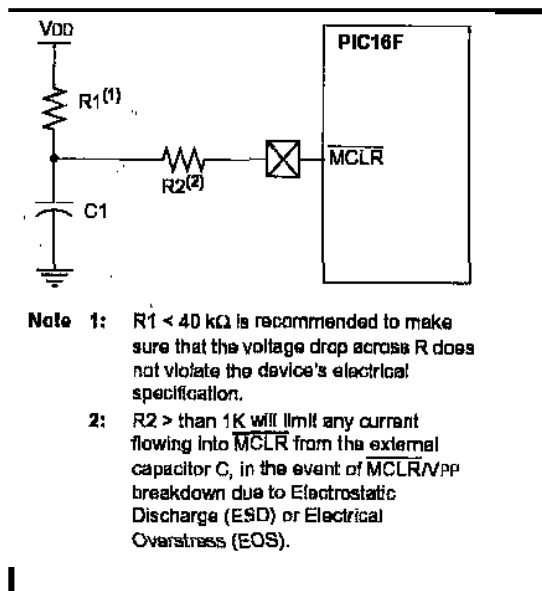
5.7.1 PINES NO UTILIZADOS

Si se deja un pin sin utilizar, puede dejarse desconectado pero configurado como SALIDA y programado en cualquier estado (ALTO o BAJO), o puede configurarse como ENTRADA con un resistor externo de 10k a V_{DD} o V_{SS} .

Las dos opciones permitirán que el pin sea empleado en lo posterior ya sea como entrada o salida sin realizar modificaciones importantes en el hardware.

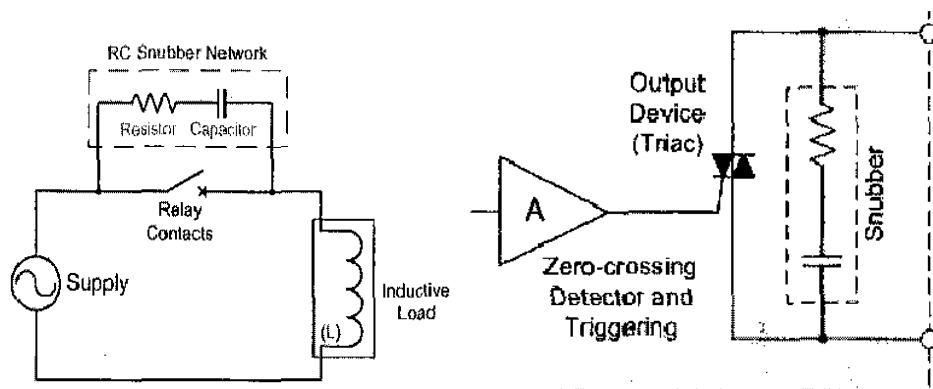
5.7.2 RESET INDESEADO #MCLR

Debido a que este tiempo es muy corto, es muy probable que se produzca un reset indeseado debido al ruido eléctrico (EMJ, ESD o picos de voltaje) en el pin #MCLR. Para evitar este problema, que suele presentarse en ambientes industriales altamente contaminados eléctricamente, el fabricante sugiere emplear una red RCR, la cual puede tener los siguientes valores: $R1=10k$, $R2=1,5k$ y $C1=0,1 \mu F$. El pin RA5/#MCLR debe estar configurado como #MCLR y no como E/S digital.



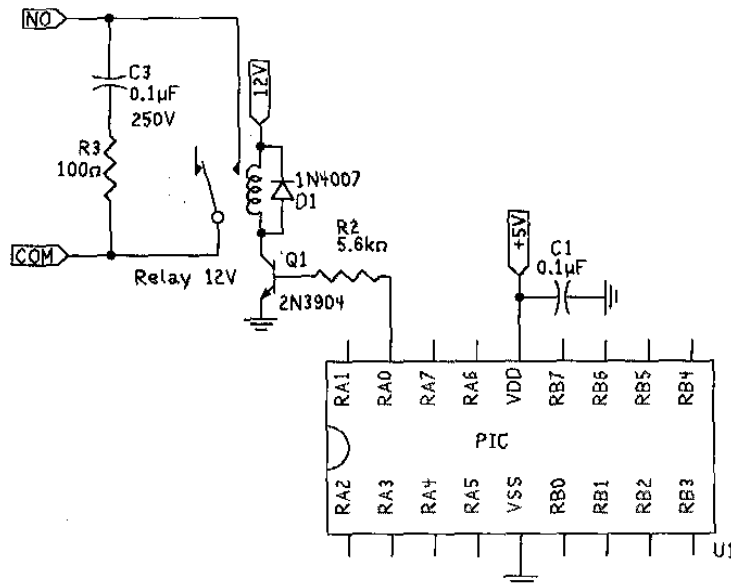
5.7.3 FUNCIONAMIENTO ERRÁTICO DEL PIC

Para evitar que el PIC opere, de manera inesperada (por ejemplo RESET indeseado, ejecución incorrecta del programa) se recomienda colocar una red *snubber* entre los contactos de los relés electromecánicos y de estado sólido (TRIAC), esto ayuda a atenuar los transitorios que se producen en el momento de la conexión y desconexión del relé, especialmente cuando hay cargas inductivas (motores, solenoides). El *snubber* está formado por una conexión RC en serie, que puede ser $R=100\ \Omega, 0,5W$ y $C=0,1\mu F/250V$ (el capacitor debe ser de alto voltaje ya que soportará normalmente los voltajes pico de CA).



5.7.4 CONEXIÓN DE UN RELÉ ELECTROMECAÁNICO AL PIC

Los relés electromecánicos (de contactos metálicos) son muy útiles para el control de un sinnúmero de cargas eléctricas, tanto de CA como de CD. Son ampliamente utilizados por sus prestaciones y bajo costo. Se pueden conectar a un microcontrolador por medio de un transistor de propósito general operando como interruptor de estado sólido y que actúa como amplificador de corriente.



CAPITULO VI

APLICACIONES Y PRACTICAS UTILIZANDO EL MODULO PARA REALIZAR PRACTICAS BASADAS EN CARGAR Y VERIFICAR MICROCONTROLADORES EN INGENIERÍA ELÉCTRICA

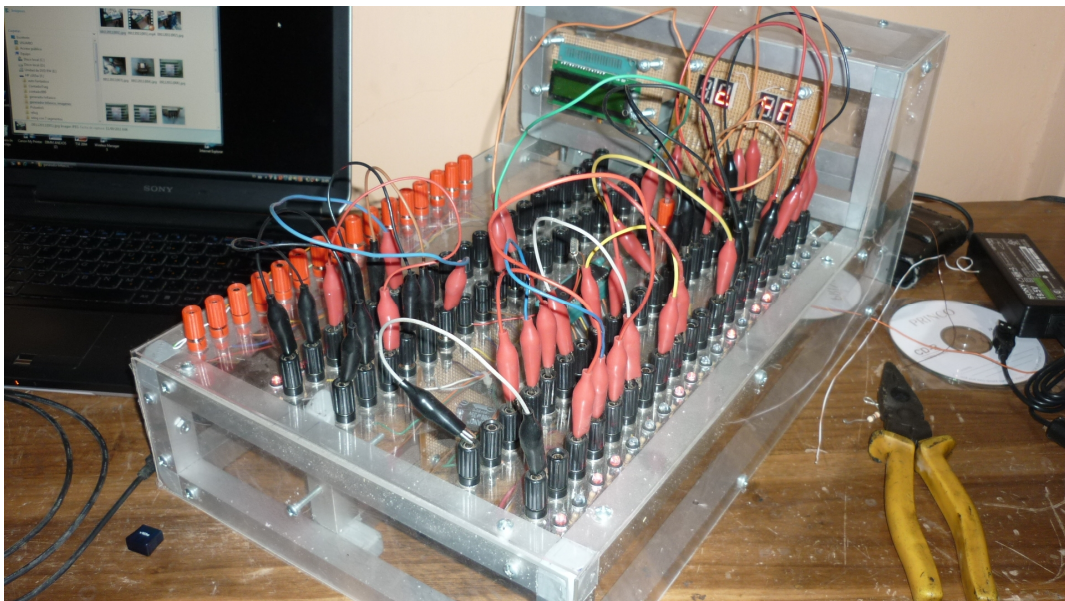
INTRODUCCION

Los microcontroladores se emplean en la actualidad en una inmensa variedad de aplicaciones de control electrónico: producción industrial, sector automotriz, aviación, construcción naval, electrodomésticos, exploración espacial, exploración marina, equipo médico, telecomunicaciones, robótica, etc. Su gran éxito se debe fundamentalmente a su enorme versatilidad, ya que el proceso de diseño se realiza casi exclusivamente en software (programación), mientras que en lo referente al hardware lo más importante es la selección del microcontrolador apropiado y nada más. Esto permite que el mantenimiento y la modificación de los diseños se hagan de una forma muy eficiente. Las instrucciones del programa almacenado se encargan de decirle al microcontrolador lo que debe hacer bajo determinadas condiciones. Además, antes de la construcción del circuito de aplicación final, se pueden realizar todas las simulaciones y correcciones necesarias hasta lograr los resultados esperados. Si en alguna ocasión se desea cambiar, mejorar o potenciar la aplicación, simplemente se debe modificar el programa y listo.

6.1 PROGRAMA QUE GENERA DOS SECUENCIAS O COMBINACIONES DIFERENTES, USANDO EL PIC 16F877, PARA SER IMPLEMENTADO COMO DESCONGESTIONADOR VEHICULAR, EN UNA INTERSECCIÓN DE TRANSITO.

FOTOGRAFIA IMPLEMENTACION EN EL ENTRENADOR

Ya que esta practica es muy sencilla, podemos relizarla simplemente en el entrenador de PIC, usando únicamente un cristal de 4 Mhz



6.1.1 CÓDIGO FUENTE

```
//Semaforo.c
//PIC16F877A
//Cristal externo de 4MHz en los pines OSC1(13) y OSC2(14).
//El arranque y la parada se efectúa al encender y apagar el PIC.

//Entradas:
//RA0(2): Selección de la secuencia (0 => Secuencia1; 1 => Secuencia2).

//Salidas:
//Puerto B: RB<7:0> Luces 1-8
//Puerto C: RC<7:0> Luces 9-16
//Puerto D: RD<7:0> Luces 17-24

void retardo(char tiempo); //Función para generar retardos en segundos.

char i;

void main(){
ADCON1=0x06; //Pines RA<5:0> como E/S digital (como entradas por defecto).
PORTB=0x00; //Inicialización.
PORTC=0x00; //Inicialización.
PORTD=0x00; //Inicialización.
TRISB=0x00; //Puerto B como salida.
TRISC=0x00; //Puerto C como salida.
TRISD=0x00; //Puerto D como salida.
while (1){
//Secuencia1
while (RA0_bit==0){
PORTB=0b1000110; PORTC=0b00010010; PORTD=0b01001001; //1
retardo(10);
PORTB=0b11000111; PORTC=0b00010010; PORTD=0b01001001; //2
retardo(3);
PORTB=0b00110000; PORTC=0b11000010; PORTD=0b01001001; //3
retardo(10);
PORTB=0b00111000; PORTC=0b11100010; PORTD=0b01001001; //4
retardo(3);
PORTB=0b10000100; PORTC=0b10011000; PORTD=0b01001001; //5
retardo(5);
PORTB=0b11000100; PORTC=0b10011100; PORTD=0b01001001; //6
retardo(3);
PORTB=0b00100110; PORTC=0b00010010; PORTD=0b01001100; //7
retardo(5);
PORTB=0b00100111; PORTC=0b00010010; PORTD=0b01001110; //8
retardo(3);
PORTB=0b00100100; PORTC=0b11000011; PORTD=0b00001001; //9
retardo(5);
PORTB=0b00100100; PORTC=0b11100011; PORTD=0b10001001; //10
retardo(3);
PORTB=0b00110000; PORTC=0b10010010; PORTD=0b01100001; //11
retardo(5);
```

```
PORTB=0b00111000; PORTC=0b10010010; PORTD=0b01110001; //12
retardo(3);
}

//Secuencia2
while (RA0_bit==1){
PORTB=0b10000110; PORTC=0b00010010; PORTD=0b01001001; //1
retardo(10);
PORTB=0b10000111; PORTC=0b00010010; PORTD=0b01001001; //2
retardo(3);
PORTB=0b10000100; PORTC=0b10011000; PORTD=0b01001001; //3
retardo(5);
PORTB=0b11000100; PORTC=0b10011100; PORTD=0b01001001; //4
retardo(3);
PORTB=0b00100110; PORTC=0b00010010; PORTD=0b01001100; //5
retardo(5);
PORTB=0b00100111; PORTC=0b00010010; PORTD=0b01001110; //6
retardo(3);
PORTB=0b00110000; PORTC=0b11000010; PORTD=0b01001001; //7
retardo(10);
PORTB=0b00111000; PORTC=0b11000010; PORTD=0b01001001; //8
retardo(3);
PORTB=0b00100100; PORTC=0b11000011; PORTD=0b00001001; //9
retardo(5);
PORTB=0b00100100; PORTC=0b11100011; PORTD=0b10001001; //10
retardo(3);
PORTB=0b00110000; PORTC=0b10010010; PORTD=0b01100001; //11
retardo(5);
PORTB=0b00111000; PORTC=0b10010010; PORTD=0b01110001; //12
retardo(3);
}
}
}

void retardo(char tiempo){
for (i=1; i<=tiempo ; i++)
Delay_1sec();
}
```


6.2.1 CÓDIGO FUENTE

```
//AD_1.c
//Declaración de las 12 variables necesarias para la conexión
//del módulo LCD.
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// Fin de declaración de variables de conexión.
int temp_res;
float voltaje;
char txt[15];

void main(){
  OSCCON=0x40;          //Oscilador interno a 1MHz (TC1=4 us).
  //while (OSCCON.IOFS==0); //Esperar mientras el oscilador está inestable.
  PORTA=0x00;          //Inicialización.
  ANSEL=0x04;          //Bit AN2 como entrada analógica.
  TRISA=0x04;          //RA2 como entrada.
  Lcd_Init();          //Inicializa el LCD.
  Lcd_Cmd(_LCD_CLEAR); //Borra el display.
  Lcd_Cmd(_LCD_CURSOR_OFF); //Apaga el cursor.
  Lcd_Out(1,1,"V-METER 0-15VCD");
  Lcd_Out(2,12,"V");
  do{
    temp_res=ADC_Read(2); //Obtiene 10 bits de la conversión del canal 2.
    voltaje=(float)(temp_res*0.00488); //Transforma el número de 10 bits al
                                     //voltaje correspondiente.
    voltaje*=3.0;          //VT=3VIN.
    FloatToStr(voltaje,txt);
    Lcd_Out(2,1,txt);
  } while(1);
```

6.3 VARIADOR DE VELOCIDAD USANDO UN MICROCONTROLADOR PIC PARA EL CONTROL.

6.3.1 RESUMEN

“Aplicaciones Industriales basadas en Microcontroladores” El mismo consiste en el diseño e implementación de un dispositivo electrónico destinado a variar la velocidad de giro de un motor eléctrico trifásico asíncrono, de manera continua dentro de ciertos límites, mediante la variación de la frecuencia de la tensión aplicada.

6.3.2 ANTECEDENTES

Tanto en la industria, como en aplicaciones específicas, es común encontrarse con la necesidad de variar la velocidad de un motor en forma continua y manteniendo el torque en el eje. También en ocasiones es necesario proporcionar a los motores un arranque suave y gradual. Para ello se idearon gran cantidad de motores específicos y equipos de regulación que permitan satisfacer estas necesidades.

Entre los más comunes en uso, se encuentran los motores de corriente continua controlados mediante la tensión del inducido y los motores asíncronos trifásicos controlados mediante la variación de frecuencia.

Los motores de continua tienen las desventajas de ser más caros, voluminosos y necesitar mayor mantenimiento debido a que utilizan carbones.

Por lo expuesto anteriormente y por el amplio desarrollo de dispositivos de estado sólido que permiten conmutar grandes corrientes a tensiones industriales, se popularizó el uso de variadores de frecuencia para motores asíncronos.

Dado que aun hoy, los equipos disponibles en el mercado son de costo relativamente elevado, motiva nuestro esfuerzo y dedicación al desarrollo de variadores de frecuencia para pequeños motores (del orden de 1/2 H.P.).

6.3.4 MATERIALES Y MÉTODOS

El desarrollo efectuado pretende obtener un prototipo simple de un variador de velocidad, aplicable a motores trifásicos asíncronos pequeños, con el que se puedan estudiar en detalle los problemas que aparecen cuando se intentan controlar motores e incluso extrapolar los resultados para los de mayor potencia y adoptar las soluciones pertinentes.

6.3.5 CRITERIOS UTILIZADOS EN EL DISEÑO

De tipo general:

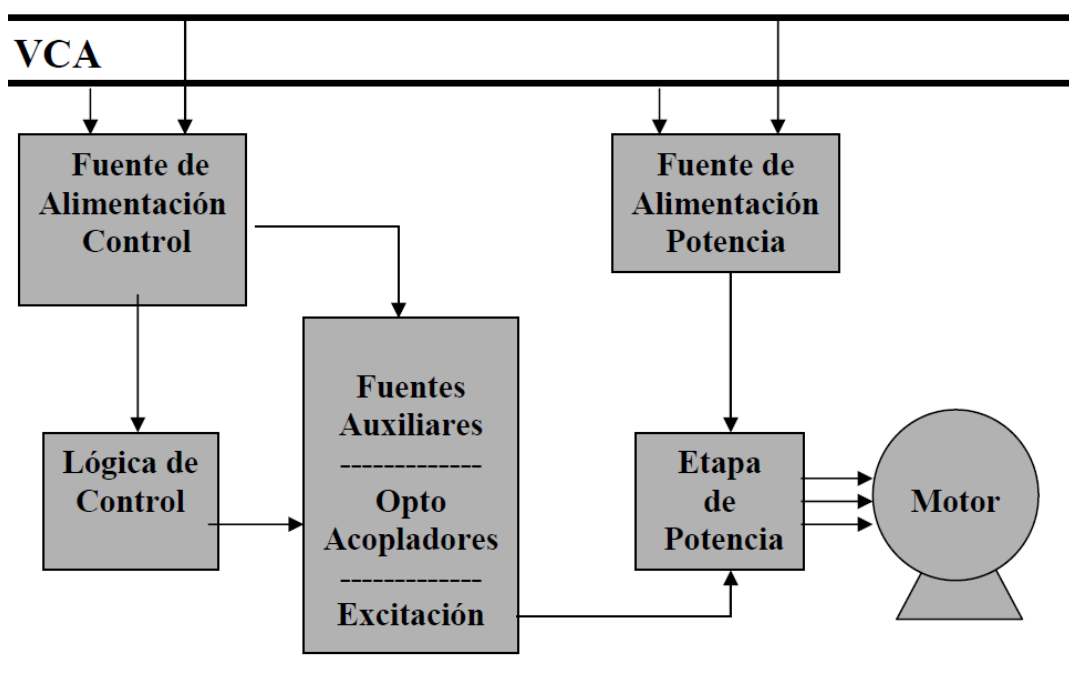
- 1) Empleo de materiales electrónicos comunes, que se puedan conseguir en plaza.
- 2) Minimización del costo de los materiales electrónicos utilizados.
- 3) Seguridad en la operación, simpleza en el manejo.
- 4) Diseño orientado hacia la posibilidad de implementarlo con Microcontroladores en una etapa posterior.

De tipo técnico:

- 1) Forma de Tensión generada del tipo Cuasi senoidal (Escalones Rectangulares).
- 2) Generación de tres fases, desfasadas 120° entre sí.
- 3) Tensión eficaz entre fases debe ser función de la frecuencia, para compensar las variaciones de corrientes debidas al cambio de frecuencia, esta condición limitará el rango de variación de frecuencias.

Desarrollo:

A continuación se presenta un diagrama en bloques de las distintas etapas necesarias para implementar el variador propuesto. Explicaremos brevemente su funcionamiento. El equipo se energiza íntegramente de la red domiciliaria monofásica de 110 voltios, 50 Hz.



Fuentes de alimentación:

1) Fuente de alimentación de la etapa de control, suministra 12 voltios de Corriente Continua a 1 Ampere, consiste en un transformador, un rectificador de onda completa en configuración puente, implementada con diodos 1N4001 y un filtrado de tipo C, con una capacidad de 2200 uF / 16 voltios.

2) Fuente de alimentación de potencia, de aproximadamente 300 voltios de Corriente Continua, destinada a suministrar la potencia que activará el motor, consiste en un puente rectificador de onda completa implementado con diodos de 6 amperes , 1000 voltios y filtrado con condensador de 400 uF / 400 Voltios, conectada directamente a la red.

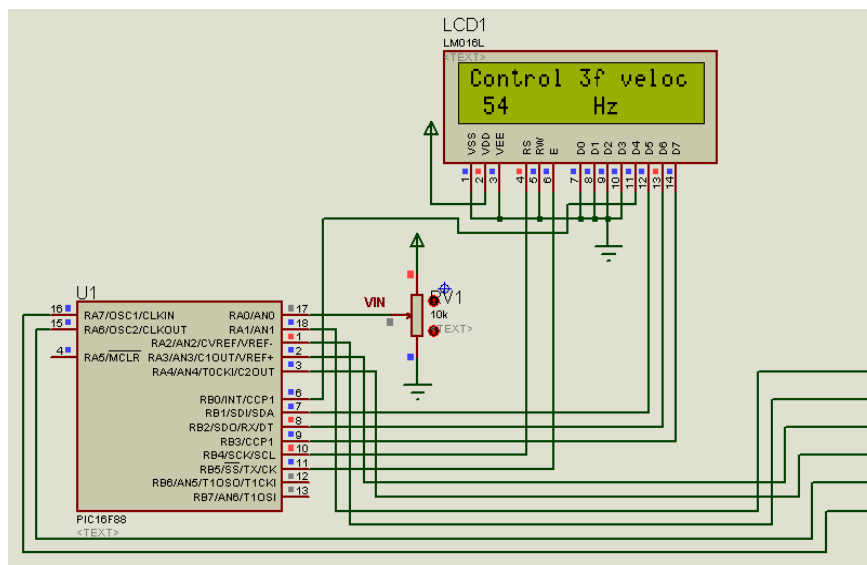
3) Fuentes Auxiliares destinadas a la excitación de los FETs de la etapa de Potencia. Consiste en cuatro fuentes de 12 voltios de Corriente Continua, **independientes.**

6.3.6 LÓGICA DE CONTROL:

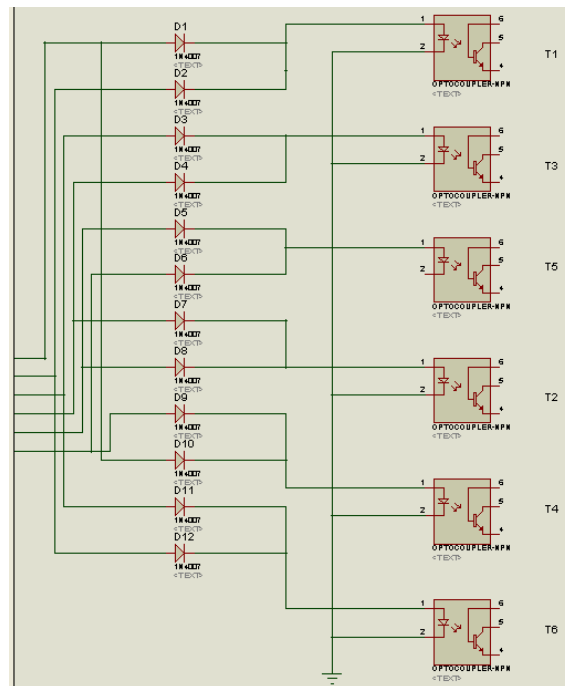
Primeramente se estableció una secuencia de conmutación que cumpla con los requisitos anteriormente enumerados, a los que se agrega una conexión del motor en configuración estrella. T1 a T6 son los optoacopladores que irán conectados a los correspondientes FETs de la etapa de Potencia, M1 a M6.

Para su realización se utilizó un PIC 16F88 seguido de compuertas OR implementadas con diodos 1N4148, los que alimentan los leds de los optoacopladores PC817 a través de resistencias limitadoras de 4.7 kOhm.

Se incorporó un potenciómetro de 10 kOhm, que es el que nos permitirá variar la frecuencia de la tensión aplicada al motor.



Circuito de Control Usando el PIC 16F88



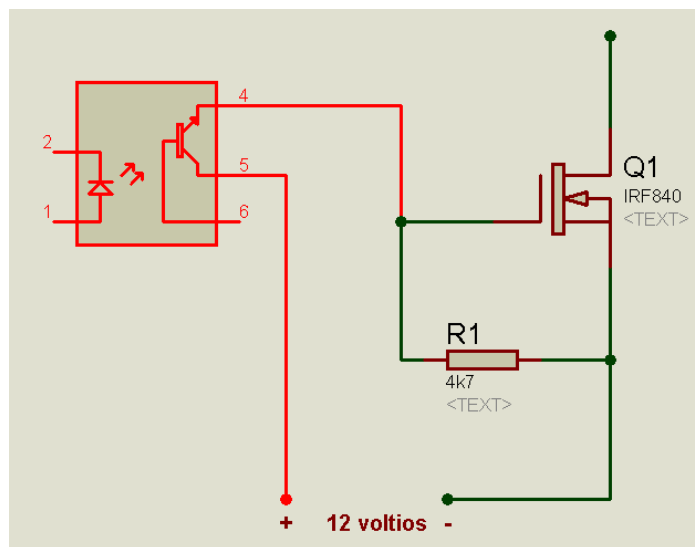
Circuito electrónico, de optocoplamiento

#	Microcontrolador						Optocopladores						Semiciclos en Mosfet		
	1	2	3	4	5	6	T1	T2	T3	T4	T5	T6	R	S	T
1	1	0	0	0	0	0	1	0	0	1	0	0	+	-	0
2	0	1	0	0	0	0	1	0	0	0	0	1	+	0	-
3	0	0	1	0	0	0	0	0	1	0	0	1	0	+	-
4	0	0	0	1	0	0	0	1	1	0	0	0	-	+	0
5	0	0	0	0	1	0	0	1	0	0	1	0	-	0	+
6	0	0	0	0	0	1	0	0	0	1	1	0	0	-	+
1	1	0	0	0	0	0	1	0	0	1	0	0	+	-	0
2	0	1	0	0	0	0	1	0	0	0	0	1	+	0	-
3	0	0	1	0	0	0	0	0	1	0	0	1	0	+	-
4	0	0	0	1	0	0	0	1	1	0	0	0	-	+	0
5	0	0	0	0	1	0	0	1	0	0	1	0	-	0	+
6	0	0	0	0	0	1	0	0	0	1	1	0	0	-	+

Secuencia Logica.

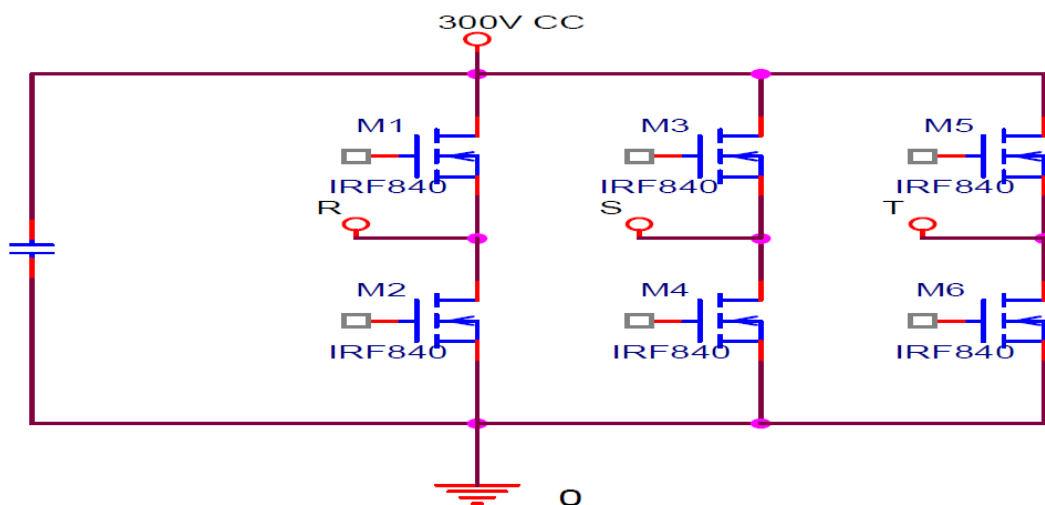
6.3.7 EXCITACIÓN DE LOS MOSFET

Se observa en el esquema utilizado, que los FETs M2, M4 y M6 tienen su terminal de surtidor conectado a la masa general, no ocurre lo mismo con los terminales de surtidor de M1, M3 y M5; en consecuencia para poder excitar cada uno de ellos en forma independiente, respecto de su propio terminal de surtidor es necesario generar cuatro fuentes aisladas. Una de ellas para cada uno de los FETs cuyo drenador se encuentra conectado a 300 Voltios (tres) y otra para los tres FETs cuyo terminal surtidor se halla a potencial de masa general. Para comandar cada FET se utilizó un optoacoplador. R, S y T son los puntos de conexión a los terminales del motor trifásico asincrónico

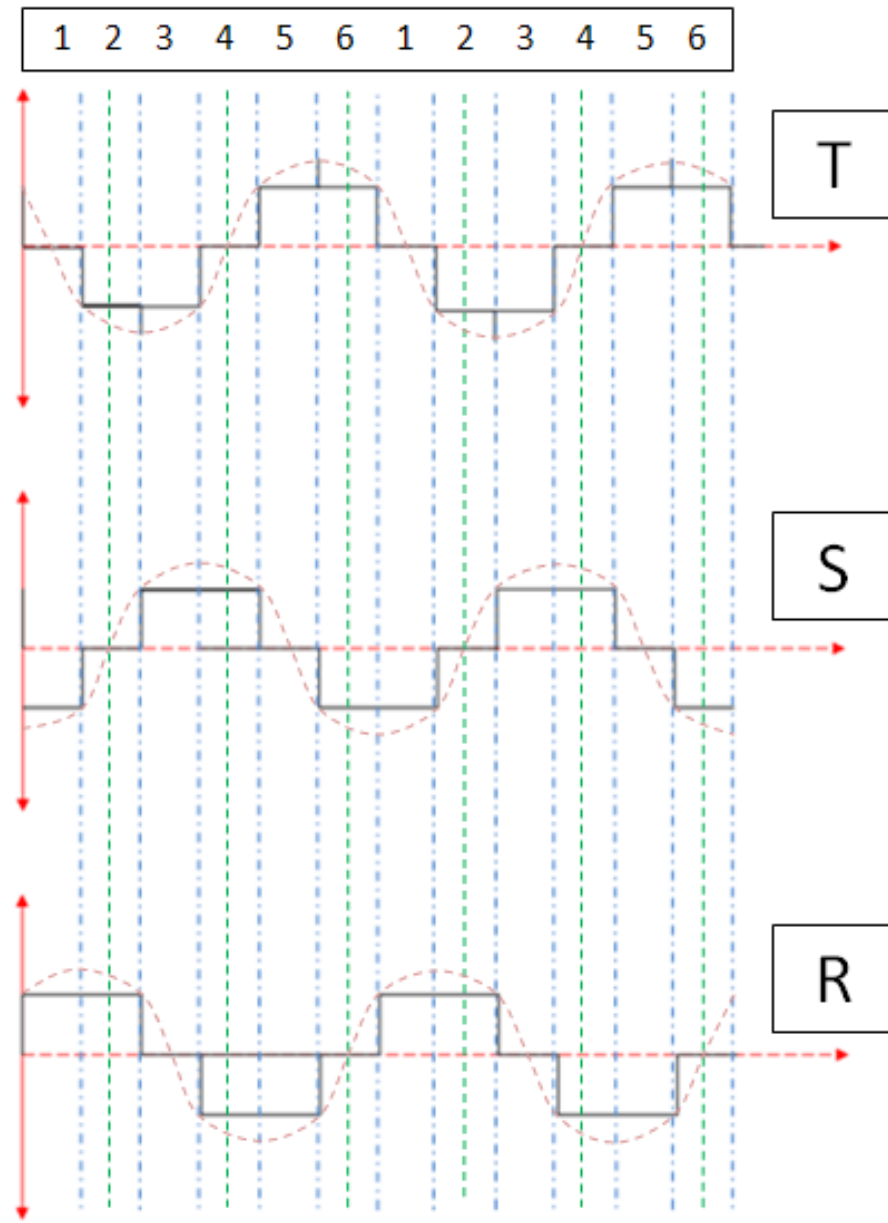


6.3.8 ETAPA DE POTENCIA:

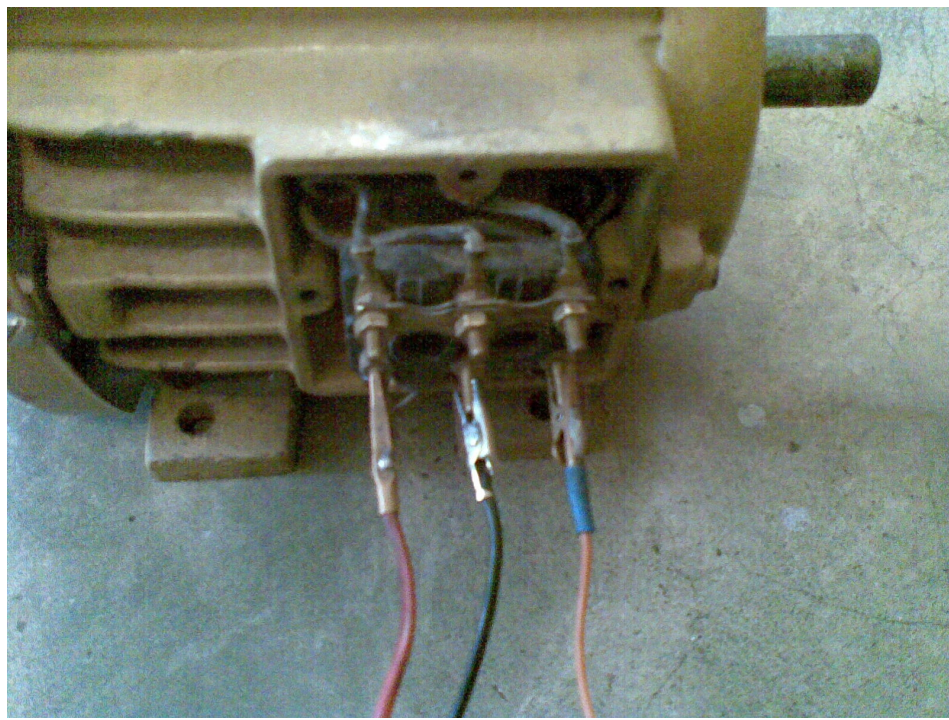
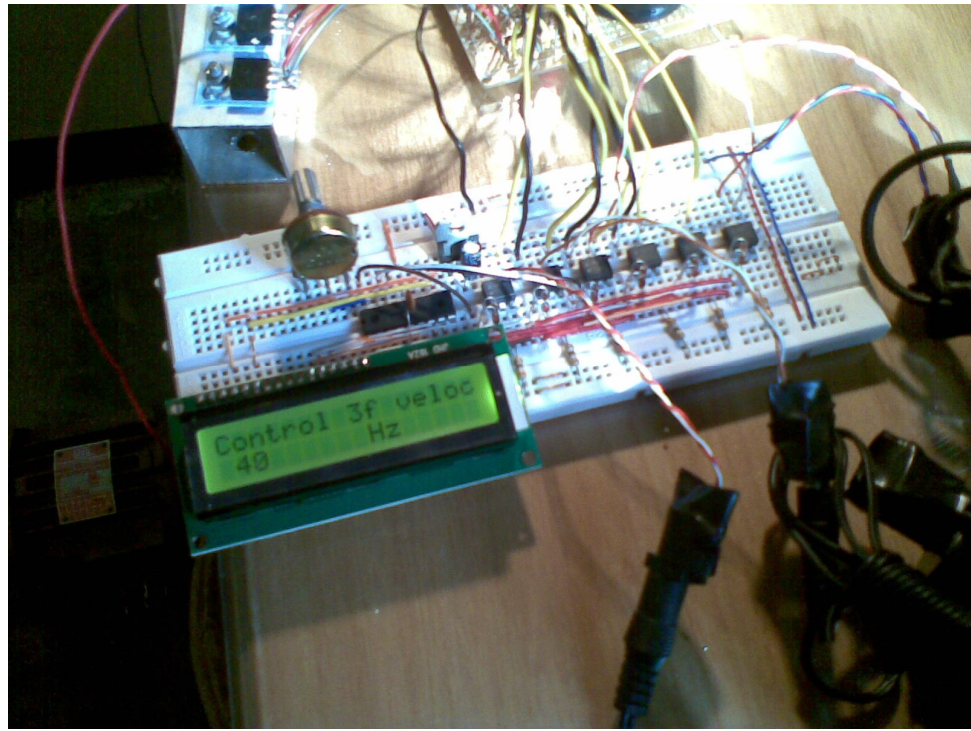
La etapa de potencia consiste en un puente trifásico realizado con seis transistores MOSFET de Canal N (IRF840), estos componentes son de bajo costo y eléctricamente muy robustos ($V_{DSS} = 500$ voltios, $I_D = 8.0$ amperes, $R_{DS} = 0.85$ Ohm). El puente está alimentado desde la fuente de alimentación de potencia, ya descrita en párrafos anteriores



Secuencia General y forma de onda entregada



Fotografía del ensayo realizado en un protoboard



6.3.9 PROCEDIMIENTO DE PRUEBA

- Conecte las fuentes auxiliares.
- Ajuste el potenciómetro de 10K ohmios ubicado en el PIC hasta que consiga la frecuencia deseada.
- Conecte la etapa de Potencia, teniendo la precaución de conectar un foco de 100wattios en serie, esto limitara la corriente durante la carga del condensador y así evitar que el fusible principal se quemara.
- Verifique los voltajes R S T, debe haber equilibrio entre ellos.
- Desconecte la Fuente de Potencia.
- Conecte el Motor, (en conexión estrella).
- Conecte la etapa de Potencia, teniendo la precaución de conectar el foco de 100 wats en serie, este procedimiento evitara que los Transistores Mosfet, se quemen durante el proceso de arranque, limitando esta corriente a 1.5 amperios, se podrá observar que a medida que el motor alcanza su velocidad nominal, el foco se apaga lentamente.
- Regular lentamente el potenciómetro ajustándolo a la frecuencia deseada, notara que a medida que esto sucede el motor varia también la velocidad.
- Se podrá observar que a medida que se realiza un cambio de frecuencia se observa un salto brusco en el Motor, esto se debe a que el Microcontrolador PIC, se actualiza, proceso que dura unos pocos milisegundos.

6.3.10 CÓDIGO FUENTE

```
//Control_vel_3f.c
//Microcontrolador: PIC16F88.
//Oscilador: Interno-8MHz
//Generador de secuencias para un control trifásico de velocidad. La velocidad se modifica por la
//variación de la frecuencia aplicada al motor, en el rango de 36 a 74Hz.
//Asignación de pines (T1 a T6 son optoacopladores 4N25):
//RA1(18)->T1-T4
//RA2(1)->T1-T6
//RA3(2)->T3-T6
//RA4(3)->T3-T2
//RA6(15)->T5-T2
//RA7(16)->T5-T4
//La secuencia de estados se genera por los pines RA<7:1>. Cada uno de los 6 estados permanece
//durante la sexta parte del período (T/6). Esto permite conmutar los MOSFETs de potencia
//adecuadamente para obtener los tres voltajes del sistema trifásico. La forma de onda es
//cuasi senoidal (Escalones rectangulares) con un desfase entre voltajes de 120 grados.

//El microcontrolador PIC16F88 se programa para generar continuamente la secuencia de estados
//para una determinada frecuencia (se puede seleccionar actuando sobre un potenciómetro de 10k).
//Cada vez que se cambia el valor del potenciómetro se actualiza la presentación en el LCD.
//Cada 50 ms se genera una interrupción debida al Timer0. Este tiempo se toma como base para
//obtener una lectura del potenciómetro cada 200 ms y de acuerdo a este valor se selecciona
//la nueva frecuencia, siempre que sea distinta a la anterior.

void retardo_us(char freq); //Función para generar retardos (us) de acuerdo a la frecuencia.

//Declaración de las 12 variables necesarias para la conexión
//del módulo LCD.
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// Fin de declaración de variables de conexión.

//Declaración de variables:
//temp_res: almacena los 10 bits del convertor A/D.
//voltaje, voltaje0: almacenan los voltajes actual y anterior, respectivamente.
//reg_cambio_pot: registra un cambio en el potenciómetro.
//reg_cambio_fr: registra un cambio en la frecuencia.
int temp_res;
```

```

float voltaje, voltaje0=5.3;
char txt[4], frecuencia=60, conteo, reg_cambio_pot, reg_cambio_fr=0;

void main(){
    OSCCON=0x70;           //Oscilador interno a 8MHz (TCI=0,5 us).
    PORTA=0x00;           //Inicialización.
    ANSEL=0x01;          //AN0 entrada analógica. AN<6:1> E/S digital.
    TRISA=0x01;          //RA0 como entrada. RA<7:1> como salidas.
    Lcd_Init();           //Inicializa el LCD.
    Lcd_Cmd(_LCD_CLEAR); //Borra el display.
    Lcd_Cmd(_LCD_CURSOR_OFF); //Apaga el cursor.
    Lcd_Out(1,1,"Control 3f velocidad");
    Lcd_Out(2,10,"Hz");
    ADCS0_bit=1;          //Frecuencia máxima de operación (10MHz) para la
    ADCS1_bit=0;          //conversión A/D.
    ADCS2_bit=1;
    OPTION_REG=0b11010111; //Timer0 como temporizador. Prescaler asignado
                          //al Timer0. Prescaler 1:256.
    GIE_bit=1;           //Interrupciones habilitadas.
    TMR0IE_bit=1;        //Interrupción del Timer0 habilitada.
    TMR0=61;             //Valor inicial del TMR0 (interrupción cada 50ms).
    while (1){
        PORTA=0b00000010;
        retardo_us(frecuencia);
        PORTA=0b00000100;
        retardo_us(frecuencia);
        PORTA=0b00001000;
        retardo_us(frecuencia);
        PORTA=0b00010000;
        retardo_us(frecuencia);
        PORTA=0b01000000;
        retardo_us(frecuencia);
        PORTA=0b10000000;
        retardo_us(frecuencia);
        PORTA=0b00000000;

        //Actualiza el LCD cuando hay cambio de frecuencia:
        if (reg_cambio_fr==1){
            reg_cambio_fr=0;
            ByteToStr(frecuencia,txt);
            Lcd_Out(2,1,txt);
        }
    }

    void retardo_us(char freq){
        switch (freq){
            case 36: Delay_us(4630); //f=36Hz, T=27778 us, (T/6)=4630 us
                    break;
            case 38: Delay_us(4386);
                    break;
        }
    }
}

```

```
case 40: Delay_us(4167);
    break;
case 42: Delay_us(3968);
    break;
case 44: Delay_us(3788);
    break;
case 46: Delay_us(3623);
    break;
case 48: Delay_us(3472);
    break;
case 50: Delay_us(3333);
    break;
case 52: Delay_us(3205);
    break;
case 54: Delay_us(3086);
    break;
case 56: Delay_us(2976);
    break;
case 58: Delay_us(2874);
    break;
case 60: Delay_us(2778);
    break;
case 62: Delay_us(2688);
    break;
case 64: Delay_us(2604);
    break;
case 66: Delay_us(2525);
    break;
case 68: Delay_us(2451);
    break;
case 70: Delay_us(2381);
    break;
case 72: Delay_us(2315);
    break;
case 74: Delay_us(2252);
    break;
}
}

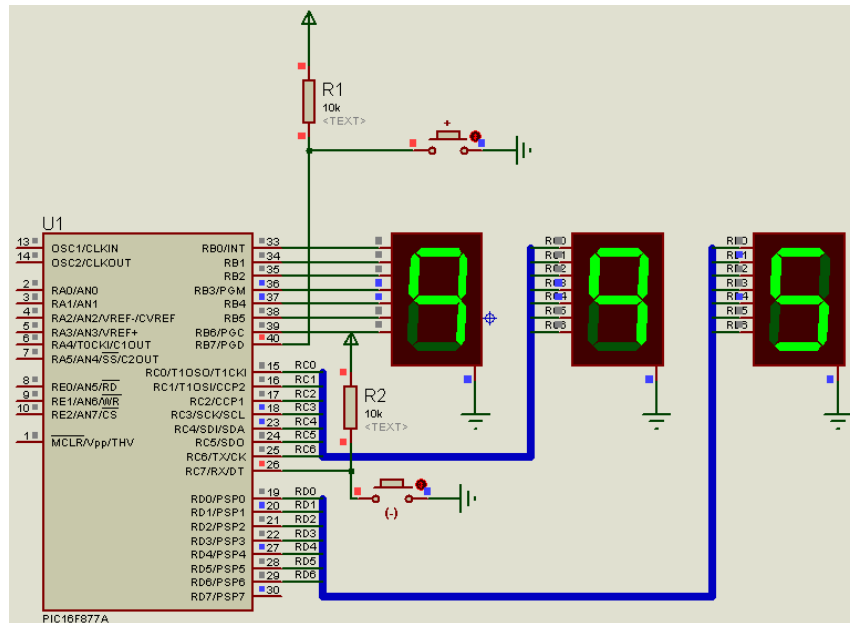
void interrupt(void){
    TMRO=61;          //Valor inicial del TMR0.
    conteo++;
    //Actualiza la entrada del potenciómetro cada 200ms (4x50ms):
    if (conteo==4){
        conteo=0;
        temp_res=ADC_Read(0); //Obtiene 10 bits de la conversión del canal 0.
        voltaje=(float)(temp_res*0.00488); //Transforma el número de 10 bits al voltaje correspondiente.

        //Detecta un cambio en el potenciómetro en función del voltaje actual y del anterior:
        if ((fabs(voltaje-voltaje0))<0.025)
            reg_cambio_pot=0; //No hay cambio en el potenciómetro.
    }
}
```

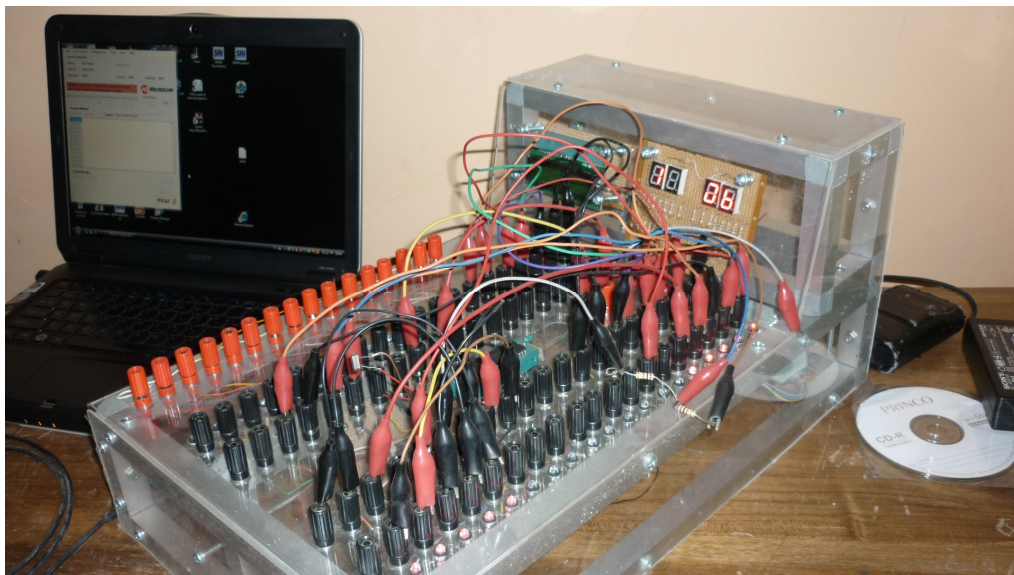
```
else
    reg_cambio_pot=1;    //Sí hay cambio en el potenciómetro.

//Si hay cambio en el potenciómetro se selecciona la frecuencia correspondiente:
if (reg_cambio_pot==1){
    if (voltaje<0.25)
        frecuencia=36;
    if (voltaje>=0.25 && voltaje<0.50)
        frecuencia=38;
    if (voltaje>=0.50 && voltaje<0.75)
        frecuencia=40;
    if (voltaje>=0.75 && voltaje<1.00)
        frecuencia=42;
    if (voltaje>=1.00 && voltaje<1.25)
        frecuencia=44;
    if (voltaje>=1.25 && voltaje<1.50)
        frecuencia=46;
    if (voltaje>=1.50 && voltaje<1.75)
        frecuencia=48;
    if (voltaje>=1.75 && voltaje<2.00)
        frecuencia=50;
    if (voltaje>=2.00 && voltaje<2.25)
        frecuencia=52;
    if (voltaje>=2.25 && voltaje<2.50)
        frecuencia=54;
    if (voltaje>=2.50 && voltaje<2.75)
        frecuencia=56;
    if (voltaje>=2.75 && voltaje<3.00)
        frecuencia=58;
    if (voltaje>=3.00 && voltaje<3.25)
        frecuencia=60;
    if (voltaje>=3.25 && voltaje<3.50)
        frecuencia=62;
    if (voltaje>=3.50 && voltaje<3.75)
        frecuencia=64;
    if (voltaje>=3.75 && voltaje<4.00)
        frecuencia=66;
    if (voltaje>=4.00 && voltaje<4.25)
        frecuencia=68;
    if (voltaje>=4.25 && voltaje<4.50)
        frecuencia=70;
    if (voltaje>=4.50 && voltaje<4.75)
        frecuencia=72;
    if (voltaje>=4.75 && voltaje<5.00)
        frecuencia=74;
    reg_cambio_fr=1;    //Registra el cambio de frecuencia.
}
voltaje0=voltaje;    //Guarda el voltaje medido.
}
TMR0IF_bit=0;    //Borra la bandera de interrupción.
```

6.4 MARCADOR DE 7 SEGMENTOS UTILIZANDO UN MICROCONTROLADOR PIC DIAGRAMA ELECTRONICO



Fotografía de prueba realizada en el Modulo Cargador y comprobador de PIC



6.4.1 CÓDIGO FUENTE;

```
//contador999.c
//Microcontrolador: PIC16F877A
//Oscilador: Externo-4MHz (modo HS)
//Contador UP-DOWN desde 000 a 999. Incrementa con un botón conectado en RB7(40) y
decrementa
//con un botón conectado en RC7(26). El último valor mostrado se mantiene al reiniciar el PIC.
//En el momento de la programación se debe grabar el PIC con los valores iniciales 00 y 00 en las
//direcciones EEPROM 0x00 y 0x01, respectivamente. De esta forma el conteo se iniciará en cero la
//primera vez que se encienda el PIC.
```

```
char Bin2_7seg(char digit); //Prototipo de la función. Transforma de binario a 7 segmentos.
```

```
char unidades, decenas, LSB;
bit estado0, estado1;
unsigned contador, contadorBCD, centenas, conteo;
```

```
void main(){
PORTB=0x00; //Inicialización.
PORTC=0x00;
PORTD=0x00;
TRISB=0x80; //RB7 como entrada. RB<6:0> como salidas.
TRISC=0x80; //RC7 como entrada. RC<6:0> como salidas.
TRISD=0x00; //Puerto D como salida.
```

```
estado0=1;
estado1=1;
```

```
//Obtener el último valor almacenado en EEPROM para reanudar el conteo:
conteo=EEPROM_Read(0x01); //Obtiene el MSB.
conteo=conteo<<8; //Desplazar 8 lugares a la izquierda.
LSB=EEPROM_Read(0x00); //
1,1)}//Si se pulsa y se libera.
contador++;
if (contador>999) contador=0; Obtiene el LSB.
conteo=conteo|LSB; //Operación OR con bits.
contador=conteo; //Inicializar el contador.
```

```
while (1){
if (Button(&PORTB,7,1,0)) estado0=0; //Si se pulsa.
if (estado0==0 && Button(&PORTB,7,
conteo=contador; //Actualiza la variable conteo.
EEPROM_Write(0x00,contador); //Guarda el LSB en la dirección 0x00.
contador=contador>>8; //Desplazar 8 lugares a la derecha.
EEPROM_Write(0x01,contador); //Guarda el MSB en la dirección 0x01.
contador=conteo; //Actualiza el contador a su valor original.
estado0=1;
}
```

```
if (Button(&PORTC,7,1,0)) estado1=0; //Si se pulsa.
if (estado1==0 && Button(&PORTC,7,1,1)){//Si se pulsa y se libera.
```

```
contador--;
if (contador==65535) contador=999;
conteo=contador;
EEPROM_Write(0x00,contador);
contador=contador>>8;
EEPROM_Write(0x01,contador);
contador=conteo;
estado1=1;
}

//Transformar el conteo para su presentación:
contadorBCD=Dec2Bcd16(conteo); //Transforma de binario a BCD.
unidades=0b1111&contadorBCD; //Sacar las unidades.
decenas=0b11110000&contadorBCD; //Sacar las decenas.
centenas=0b111100000000&contadorBCD; //Sacar las centenas.
decenas=decenas>>4; //Desplazar 4 bits a la derecha.
centenas=centenas>>8; //Desplazar 8 bits a la derecha.
PORTD=Bin2_7seg(unidades); //Unidades al puerto D.
PORTC=Bin2_7seg(decenas); //Decenas al puerto C.
PORTB=Bin2_7seg(centenas); //Centenas al puerto B.
}
}

char Bin2_7seg(char digit){ //Definición de la función.
switch (digit)
{ case 0: return 0x3F; //0x3F es el código 7-segmentos del 0.
  case 1: return 0x06; //0x06 es el código 7-segmentos del 1.
  case 2: return 0x5B;
  case 3: return 0x4F;
  case 4: return 0x66;
  case 5: return 0x6D;
  case 6: return 0x7D;
  case 7: return 0x07;
  case 8: return 0x7F;
  case 9: return 0x67;
}
}
```

CAPITULO VII

CONCLUSIONES Y RECOMENDACIONES

7.1 CONCLUSIONES

1.- Concluida la construcción de la presente Tesis se la ensayó con diferentes aplicaciones, tomando en cuenta siempre la facilidad que necesita el estudiante al momento de aprender.

2.- Los Microcontroladores siempre han ocupado un puesto muy importante para quienes trabajamos y subsistimos dentro del campo industrial, y puedo llegar a decir que pueden cumplir varias funciones tan bien como si fuera un PLC ya que los famosos PLC, son fabricados en base a Microcontroladores PIC.

3.- El lenguaje utilizado para el código fuente es de fácil aplicación, y el programa puede ser descargado libremente por internet dando facilidades para la obtención de información.

4.- Los componentes utilizados para realizar las referidas practicas anteriores son de muy bajo costo, y pueden ser obtenidos fácilmente, debido a elevada popularidad para quienes frecuentemente se le presentan problemas, de tipo implementativo, ya que los PIC pueden ser usados en casi cualquier

aplicación y puede llegar en determinado momento a reemplazar a los PLC, en procesos automáticos sencillos y complejos.

7.2 RECOMENDACIONES.

1.- El mundo de la Electrónica ofrece en la actualidad posibilidades inimaginables de mercado, pero para que eso llegue a ser posible los estudiantes deben comprometerse en el estudio práctico, de lo que las tecnologías nos ofrecen.

2.- Para la correcta utilización de este estudio de Tesis para los estudiantes de Ingeniería, se debe intensificar las prácticas tanto con Software como con hardware, tratando de que encuentre interesante el estudio del presente proyecto, tratando de esta manera de hacerlo competitivo entre sus compañeros de clase, ya que de esta manera es posible mejorar la estructura tecnológica y científica de nuestra Facultad de Ingeniería.



VISTA FISICA DE LOS CIRCUITOS INTEGRADOS PROGRAMABLES PIC

BIBLIOGRAFIA

1.- Revista Saber Electrónica Internacional.

2.- Gilberto RodriguezHarper – 2000

3.- Ingenieria de control moderna.

Katsuhiko – Ogata – 2003.

4.- Instrumentación Industrial.

Antonio Creus Solé – 2005

5.- Electronica Industrial Moderna.

Timothy J. Maloney – 2006.

6.- Microcontroladores fundamentos y Aplicaciones con Pic.

Fernando E. Valdes Pérez.

Ramón Pallas Areny.

7.- Programar PIC en C.

Juan Ricardo Peñagos Plazas