UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN

CONTROL DE POSICIÓN DE UN MOTOR DE CD MEDIANTE UNA INTERFASE CON LA COMPUTADORA EMPLEANDO UN MICROCONTROLADOR PIC

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECÁNICO ELÉCTRICISTA

PRESENTA:
LEOPOLDO MARTÍN DEL CAMPO RAMÍREZ

ASESOR: ING. JORGE BUENDÍA GÓMEZ

CUAUTITLAN IZCALLI, EDO. DE MEX. 2006

VNIVERSIDAD NACIONAL
AVTONOMA DE
MEXICC

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
UNIDAD DE LA ADMINISTRACION ESCOLAR
DEPARTAMENTO DE EXAMENES PROFESIONALES

ASUNTO: VOTOS APROBATORIOS

U. N. A. M.
FACULTAD DE ESTUDIOS
SUPERIORES CUAUTITLAN

DEPARTAMENTO DE
EXAMENES PROFESIONALES

DR. JUAN ANTONIO MONTARAZ CRESPO
DIRECTOR DE LA FES CUAUTITLAN
P R E S E N T E

ATN: Q. Ma. del Carmen García Mijares
Jefe del Departamento de Exámenes
Profesionales de la FES Cuautitlán

Con base en el art. 28 del Reglamento General de Exámenes, nos permitimos comunicar a usted que revisamos la TESIS:

Control de posición de un motor de CD mediante una interfase con la computadora empleando un microcontrolador PIC.

que presenta el pasante: Leopoldo Martín del Campo Ramírez
con número de cuenta: 09438347-2 para obtener el título de :
Ingeniero Mecánico Electricista

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VOTO APROBATORIO.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlán Izcalli, Méx. a 27 de Enero de 2006

PRESIDENTE Ing. José Luis Cruz Gutierrez

VOCAL Ing. Jorge Buendía Gómez

SECRETARIO Ing. Juan González Vega

PRIMER SUPLENTE Ing. José Gustavo Orozco Hernandez

SEGUNDO SUPLENTE Ing. Jorge Ramírez Rodríguez

Dedicatoria

Este trabajo y todo el trabajo que he realizado a lo largo de muchos años se lo dedico el primer lugar a mis padres, que hicieron todo a su alcance para que pudiera llegar hasta este punto. Espero hacerlos sentir orgullosos y con la tranquilidad de que me han dado todo necesario para conducir mi vida de una buena manera. Además quiero que sepan que pueden contar conmigo siempre y cada vez que me necesiten haré todo lo que esté en mis manos para apoyarlos y ayudarlos.

A continuación le dedico este trabajo a toda mi familia, mis hermanos, tíos y primos quienes siempre estuvieron al pendiente de mi, sobre todo a Marisol, Luís y Manuel que cuando lo necesité me apoyaron, también quiero hacer una mención especial para Octavio y Pepe que realmente saben como hacer que uno se sienta incómodo y alentado al mismo tiempo.

Finalmente le dedico este esfuerzo a mis amigos, que forman una parte muy importante y entrañable de mi vida pues existen muchas situaciones en la vida en las que sólo los amigos pueden brindar consuelo, alegría y compañerismo como ninguna otra persona y que aunque no los tengo en grandes cantidades si en gran estima. Les doy las gracias por su apoyo y compañía a mis más grandes amigos Osvaldo, Everest, Sergio y Mao en quienes se que puedo poner mi confianza y que pueden poner la suya en mi.

Agradecimientos

Además de las dedicatorias que hago también deseo agradecer a otras personas que han aparecido en mi vida por poco o mucho tiempo pero que han dejado una marca en mí. Es probable que algunos de ellos no lo sepan o que nunca lean esto pero a pesar de eso quiero hacerles un reconocimiento.

A la familia Vigueras, que me ha recibido en su casa y sus vidas durante muchos años, a quienes realmente aprecio y en especial le quiero agradecer a la señora Rosi que siempre me trató como a otro de sus hijos y por lo cual le estoy muy agradecido y considero su casa y compañía como mi segundo hogar.

También agradezco a varios de mis profesores de preparatoria. A mis profesores de química y de biología que aun que no llevábamos una muy buena relación me hicieron ver que en esos campos de estudio no eran para mí. También le agradezco a Vicente García profesor de distintas asignaturas de matemáticas que me mostró que estas no tienen que ser necesariamente frías y rígidas sino que pueden ser vistas desde un punto de vista amable y con buen humor.

Además debo mencionar a algunos de mis profesores de la facultad como al Ing. Filiberto Leyva que a pesar de que es un profesor muy duro también es amigo de sus alumnos y una excelente persona, al Ing. Marcelo Bastida que me mostró que la teoría de electrónica sirve para algo más aparte de estar en los apuntes y muy en especial le quiero agradecer al Ing. Jorge Buendía por aguantarme durante tanto tiempo, primero como alumno, luego como servicio social, después compartiendo cubículo y finalmente como mi asesor. Si le causé inconvenientes o alguna molestia espero me disculpe, sin embargo fue muy agradable para mí pues todos los días aprendí algo nuevo de usted, por eso, por su forma amable de tratar a los demás y por su incansable afán de conocimiento que no lo deja ni un momento del día tiene todo mi respeto.

Índice

3. Desarrollo del sistema de control del motor

Introducción

En la actualidad existen en el mercado una serie de dispositivos conocidos como microcontroladores, estos reúnen dentro de un solo chip muchas de las características propias de una computadora, como pueden ser las memorias RAM y ROM o los puertos, además de otros elementos. El encontrarse todos los elementos del microcontrolador en un solo chip indica que son dispositivos de muy alta densidad de integración. Los microcontroladores suelen ser utilizados en aplicaciones de control debido a que utilizan de manera más eficiente la memoria de que disponen además de poseer un conjunto de instrucciones mucho más reducido en comparación con una computadora. Otra característica que ha contribuido para que los microcontroladores hayan tomado la preferencia de muchos usuarios es su bajo costo pues, aunque varía dependiendo de la capacidad de cada modelo de microcontrolador, generalmente resultan ser muchas veces más baratos que otros medios de control como pueden ser las computadoras o los controladores lógicos programables (PLC's).

Los microcontroladores poseen una gran capacidad de adaptación a las distintas aplicaciones de control que se presentan en la industria, a pesar de que sus capacidades son limitadas. Esto lo compensan con la gran cantidad de modelos existentes, de los cuales se puede seleccionar aquel que cubra las necesidades del sistema a controlar.

Por otro lado, en la actualidad la industria está utilizando de manera intensiva los métodos de automatización de procesos tanto para aumentar su producción como para reducir los riesgos laborales de los empleados. Esto ha propiciado el desarrollo de tecnologías para el control de estos procesos, como son los microcontroladores. Por lo anterior, se ha hecho común observar en la industria que los sistemas automáticos que realizan los procesos son, en muchos casos, impulsados por motores. Un ejemplo de esto pueden ser los brazos robóticos, que dependiendo de la complejidad de sus movimientos utilizan una gama de motores que son controlados de muchas maneras distintas.

El objetivo de éste trabajo es el de proponer un medio de control para motores de CD que pueda ser aplicado a procesos industriales. Este control esta diseñado de forma básica ya que permite seleccionar el sentido de giro y el movimiento de rotación del motor por medio de un control remoto que puede estar ubicado en una computadora. El sistema de control puede ser adecuado para otros tipos de motores o para realizar secuencias predeterminadas de movimientos que deben ser repetitivos. Esto puede lograrse realizando pequeños cambios en la programación contenida dentro del chip, lo que no resulta complicado gracias al reducido número de instrucciones propio del microcontrolador. Lo anterior proporciona una gran ventaja pues los cambios pueden realizarse rápidamente afectando lo menos posible el proceso y deteniéndolo por poco tiempo, lo que es de gran importancia en estos días en donde detener la producción puede derivar en grandes pérdidas para las empresas.

En el capítulo uno de este trabajo trata de las características básicas de un microcontrolador y su diferencia con respecto al microprocesador. Describe de forma general la arquitectura interna de los elementos que constituyen a los microcontroladores y finalmente muestra la arquitectura interna del microcontrolador PIC16F877A que fue el seleccionado para realizar éste trabajo.

En el segundo capítulo se describe el funcionamiento elemental de un motor de CD con los tipos de motores en los que se subdividen. Se da una explicación del método de frenado utilizado en este trabajo, hay que tener en cuenta que existen otros medios para realizar el frenado dependiendo del funcionamiento del motor a controlar. Y para terminar se da una descripción del método que se utilizó para acoplar el circuito de control (etapa digital), a la etapa de potencia correspondiente al motor.

El contenido del tercer capítulo se enfoca a lo es respecta al sistema de control. Trata de los registros utilizados por el sistema de control y los módulos que están involucrados durante el funcionamiento, como pueden ser los puertos o los temporizadores. Cabe mencionar que las

capacidades del microcontrolador utilizado exceden los requerimientos del sistema por lo que no se abordan los módulos y registros que no están involucrados.

A continuación se hace una descripción de la operación del software utilizado tanto para crear la programación del microcontrolador como para trasladar dichos programas al microcontrolador y se dan varios ejemplos de programación con los cuales puede probarse por partes los elementos del sistema para finalmente mostrar el programa de control del motor que es el resultado final del trabajo.

1. Microcontroladores

1.1. Definición y conceptos de un microcontrolador

Un microcontrolador (*µC*) es un circuito de tipo *LST* programable que contiene todos los componentes de una computadora. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que gobierna. Esta característica hace que también reciba el nombre de "controlador incrustado" (embedded controller).

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

Un µC es una computadora completa, aunque de limitadas capacidades, la cual realiza una tarea específica que está contenida en un solo circuito integrado y del que sólo salen al exterior las líneas que gobiernan los periféricos que está destinado a controlar, fig. 1.1. A esto se le conoce como "sistema cerrado".

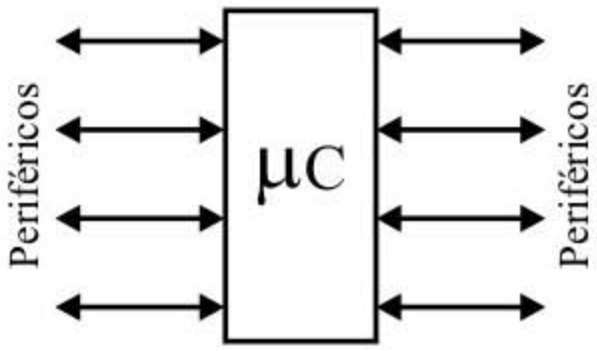


Fig. 1.1. Estructura básica externa de un microcontrolador

Las principales características de un μC se enlistan a continuación:

Sistemas integrados de baja capacidad de memoria

Alta flexibilidad

Aplicación específica un vez programados

Bajo costo de adquisición

Herramientas accesibles

Ciclo diseño-producción de tiempo reducido

Amplia gama de dispositivos

1.1.1. Diferencia entre un microprocesador y un microcontrolador

El microprocesador (*μP*) es un circuito integrado que contiene el *CPU*, también llamado procesador, de la computadora. El CPU está formado por la unidad de control, que interpreta las instrucciones, y el camino de datos, que las ejecuta.

Los pines de un μp conectan con el exterior las líneas de sus buses de direcciones, datos y control, para permitirle conectarse con la memoria y los módulos de *I/O* y configurar una computadora implementada por varios circuitos integrados, fig. 1.2. Se dice que un μp es un "sistema abierto" porque su configuración es variable de acuerdo con la aplicación a la que sea destinado, acoplándole los módulos necesarios.

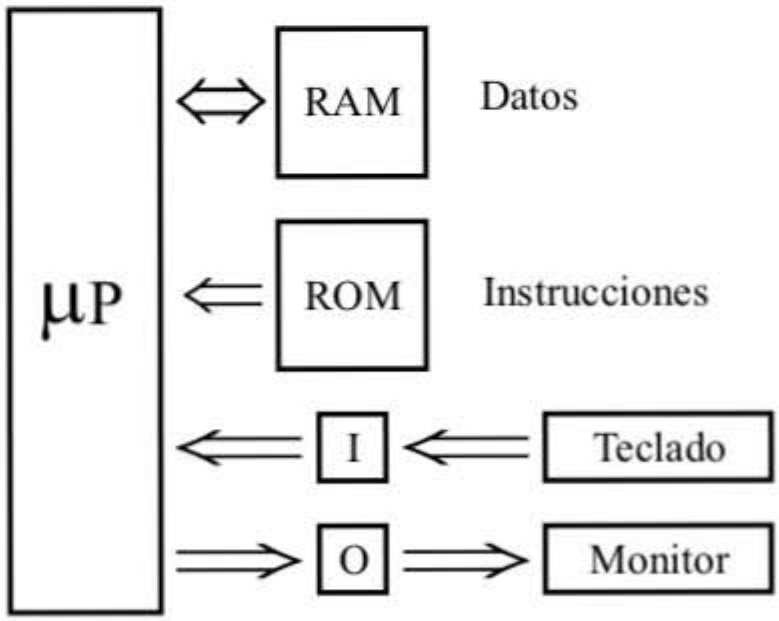


Fig. 1.2. Estructura básica de una computadora

1.2. Arquitectura interna básica de un microcontrolador

Un µC posee todos los componentes de una computadora, pero con características fijas que no pueden alterarse.

Las partes principales de un µC son:

1. Procesador
2. Memoria no volátil para contener el programa
3. Memoria de lectura y escritura para guardar los datos
4. Líneas de I/O para los controladores de periféricos (puertos):
    a) Comunicación paralelo
    b) Comunicación serie
    c) Diversas puestos de comunicación (bus *I²C*, *USB*, etc.)
5. Recursos auxiliares:
    a) Circuito de reloj
    b) Temporizadores
    c) Perro guardián (Watchdog)
    d) Conversores *A/D* y *D/A*
    e) Comparadores analógicos
    f) Protección ante fallos de la alimentación
    g) Estado de reposo o de bajo consumo (Sleep)

1.2.1. El Procesador

La necesidad de conseguir elevados rendimientos en el procesamiento de las instrucciones ha desembocado en el empleo generalizado de procesadores de arquitectura *Harvard* frente a los tradicionales que seguían la arquitectura de *Von Neumann*. El concepto central en la arquitectura Von Neumann es el de programa almacenado, según el cual las instrucciones y los datos tenían que almacenarse juntos en un medio común y uniforme, en vez de separados, como hasta entonces se hacía, fig. 1.3. De esta forma, no sólo se podían procesar cálculos, sino

que también las instrucciones y los datos podían leerse y escribirse bajo el control del programa. A partir de esta idea básica se sigue que un elemento en la memoria tiene una calidad ambigua con respecto a su interpretación; esta ambigüedad se resuelve, sólo temporalmente, cuando se requiere ese elemento y se ejecuta como una instrucción, o se opera como un dato. Un beneficio de esta ambigüedad es el hecho de que un dato, obtenido como resultado de algunas operaciones en la unidad aritmetico-lógica del computador, podía colocarse en la memoria como si fuera cualquier otro dato, para entonces usarlo y ejecutarlo como si fuera una instrucción. Además la Máquina de Von Neumann presentaba como característica importante un pequeño número de registros para mantener la instrucción del programa en curso, y el registro de datos que se estaban procesando. La máquina operaba en un ciclo repetitivo de pasos para localizar y ejecutar en secuencia las instrucciones del programa.

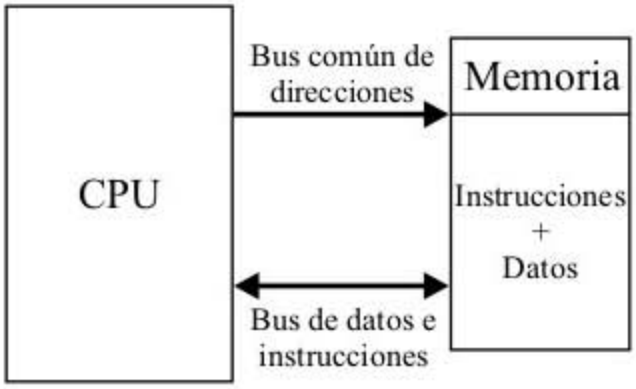


Fig. 1.3. Arquitectura de Von Neumann

En la arquitectura Harvard son independientes la memoria de instrucciones y la memoria de datos y cada una dispone de su propio sistema de buses para el acceso, fig. 1.4. Además propicia el paralelismo, al ser los buses independientes, pues el CPU puede acceder a los datos para completar la ejecución de una instrucción, y al mismo tiempo leer la siguiente instrucción a ejecutar.

Ventajas de esta arquitectura:

1. El tamaño de las instrucciones no esta relacionado con el de los datos, y por lo tanto puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa, logrando así mayor velocidad y menor longitud de programa.
2. El tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad en cada operación.

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

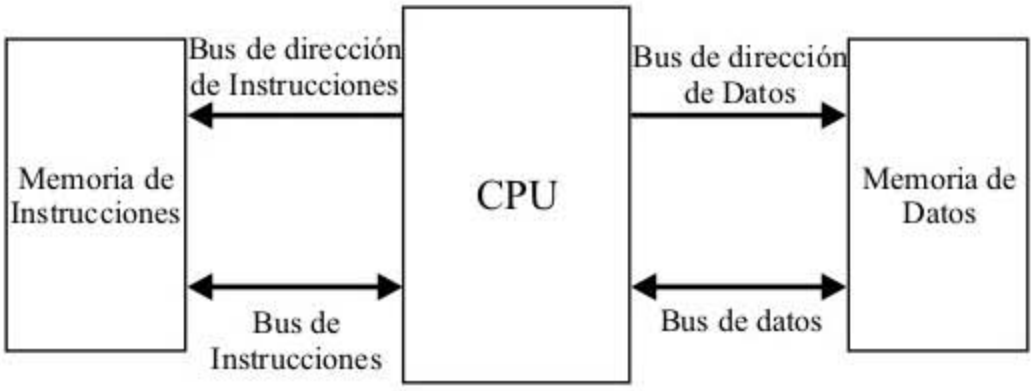


Fig. 1.4. Arquitectura Harvard

El procesador de los modernos µC responde a la arquitectura ***RISC***, que se identifica por poseer un repertorio de instrucciones de máquina pequeño y simple, de forma que la mayor parte de las instrucciones se ejecuta en un ciclo de instrucción. Quizás la más importante ventaja es que los chips con arquitectura RISC requieren muchos menos transistores, lo cual los hace más baratos de diseñar y producir. Desde que emergieron las computadoras RISC, se ha referido a las computadoras convencionales como ***CISC***.

Otra aportación frecuente que aumenta el rendimiento de la computadora es el fomento del paralelismo implícito, que consiste en la segmentación del procesador, descomponiéndolo en etapas que procesan una instrucción diferente en cada una de ellas y trabajar con varias a la vez

1.2.2. Memoria de Programa

El µC está diseñado para que en su memoria de programa se almacenen todas las instrucciones del programa de control. No hay posibilidad de utilizar memorias externas de ampliación.

Como el programa a ejecutar siempre es el mismo, debe de estar grabado de forma permanente. Los tipos de memoria adecuados para soportar esta función admiten cinco versiones diferentes:

1º. ROM con máscara o mascarada

En este tipo de memoria el programa se graba en el chip durante el proceso de su fabricación mediante el uso de ***máscaras***. Los altos costos de diseño e instrumental sólo aconsejan usar este tipo de memoria cuando se precisan series muy grandes.

2º. EPROM

La grabación de esta memoria se realiza mediante un dispositivo físico gobernado desde una computadora personal, que recibe el nombre de grabador. En la superficie de la cápsula del µC existe una ventana de cristal por la cual se puede someter al chip de la memoria a rayos ultravioletas para producir su borrado y emplearla nuevamente. La memoria ***EPROM*** es utilizada principalmente en la fase de diseño y depuración de los programas, pero su costo unitario es elevado.

3º. OTP

El modelo de memoria ***OTP*** sólo se puede grabar una vez por parte del usuario, utilizando el mismo procedimiento que con la memoria EPROM. Una vez grabada no es posible borrarla. Por su bajo precio y la sencillez de la grabación, este tipo de memoria se emplea para prototipos finales y series de producción cortas.

4º. EEPROM

La grabación es similar a las memorias OTP y EPROM, pero el borrado es mucho más sencillo al poderse efectuar de la misma forma que el grabado, o sea, eléctricamente. Sobre el mismo

zócalo del grabador puede ser programada y borrada tantas veces como se quiera, lo cual la hace ideal en la enseñanza y en la creación de nuevos modelos.

Aunque se garantiza 1.000.000 de ciclos de escritura/borrado en una ***EEPROM***, todavía su tecnología de fabricación tiene obstáculos para alcanzar capacidades importantes y el tiempo de escritura de las mismas es relativamente grande y con elevado consumo de energía.

5º. FLASH

Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar en circuito al igual que las EEPROM, pero suelen disponer de mayor capacidad que estas últimas. El borrado solo es posible con bloques completos y no se puede realizar sobre posiciones específicas. En las FLASH se garantizan 1000 ciclos de escritura/borrado.

Son muy recomendables en aplicaciones en las que sea necesario modificar el programa a lo largo de la vida del producto, como consecuencia del desgaste o cambio de piezas, como sucede con los vehículos.

1.2.3. Memoria de Datos

Los datos que manejan los programas varían continuamente, y esto exige que las memorias que los contienen deban ser de lectura y escritura, por lo que la memoria ***RAM*** estática (SRAM) es la más adecuada, aunque sea volátil.

Hay µC que también disponen como memoria de datos una de lectura y escritura no volátil, del tipo EEPROM. De esta forma un corte en el suministro de alimentación no ocasiona la pérdida de la información, que está disponible al reiniciarse el programa.

1.2.4. Líneas de I/O

A excepción de los pines destinados a recibir la alimentación, otros dos para el cristal de cuarzo, que regula la frecuencia de trabajo, y una más para provocar el reset, los restantes pines de un µC sirven para soportar su comunicación con los periféricos externos que controla.

Las líneas de I/O que se adaptan con los periféricos manejan información en paralelo y se agrupan comúnmente en conjuntos de 8, aunque no necesariamente ocurre así siempre. Al agrupamiento de las líneas de I/O se les conoce como "puertos". Hay modelos con líneas que soportan la comunicación serie; otros disponen de conjuntos de líneas que implementan puertos de comunicación para diversos protocolos, como el I^2C, el USB, etc.

1.2.5. Recursos auxiliares

Según las aplicaciones a las que orienta el fabricante cada modelo de µC, implementan una diversidad de complementos que refuerzan la potencia y la flexibilidad del dispositivo. Entre los recursos más comunes se citan los siguientes:

a) *Circuito de reloj*, encargado de generar los impulsos que sincronizan el funcionamiento de todo el sistema.
b) *Temporizadores*, orientados a controlar tiempos.
c) *Perro guardián (watchdog)*, destinado a provocar una re-inicialización cuando el programa queda bloqueado.
d) *Conversores A/D y D/A*, para poder recibir y enviar señales analógicas.
e) *Comparadores analógicos*, para verificar el valor de una señal analógica-
f) *Sistema de protección ante fallos de la alimentación*.
g) *Estado de Reposo (sleep)*, en el que el sistema queda "congelado" y el consumo de energía se reduce al mínimo.

1.3. Arquitectura del microcontrolador PIC16F877A

Las características propias del PIC16F877A lo hacen un µC de gama media, es decir, posee un repertorio de 35 instrucciones de 14 bits cada una. También dispone de interrupciones y una pila de 8 niveles que permite el anidamiento de subrutinas.

La gama media puede clasificarse en las siguientes subfamilias:

a) Gama media estándar (PIC16C55X);
b) Gama media con comparador analógico (PIC16C62X/64X/66X);
c) Gama media con módulo de captura (CCP), modulación de anchura de impulsos (PWM) y puerto serie (PIC16C7X);
d) Gama media con CAD de 8 bits (PIC16C7X);
e) Gama media con CAD de precisión (PIC14000);
f) Gama media con memoria Flash y EEPROM (PIC16F87X/87XX y PIC16X8X);
g) Gama media con driver *LCD* (PIC16C92X)

En el caso del PIC16F877A, este se ubica en la subfamilia que posee memoria Flash y EEPROM; de forma general, las características de este uC son:

Rendimiento del procesador

- Un conjunto de 35 instrucciones
- Memoria de programa FLASH de 8k x 14
- Memoria de datos de 368 x 8 bytes (RAM)
- Memoria de datos EEPROM de 256 x 8 bytes

Características de los periféricos

- 2 Temporizadores de 8 bits y 1 de 16 bits
- 2 Módulos de Captura, Comparación y PWM
- Puerto serial síncrono (*SSP*) con *SPI* e I^2C
- Módulo *USART* con detección de dirección de 9 bits
- Puerto Paralelo Esclavo (*PSP*) con un ancho de 8 bits y control externo de lectura, escritura y borrado

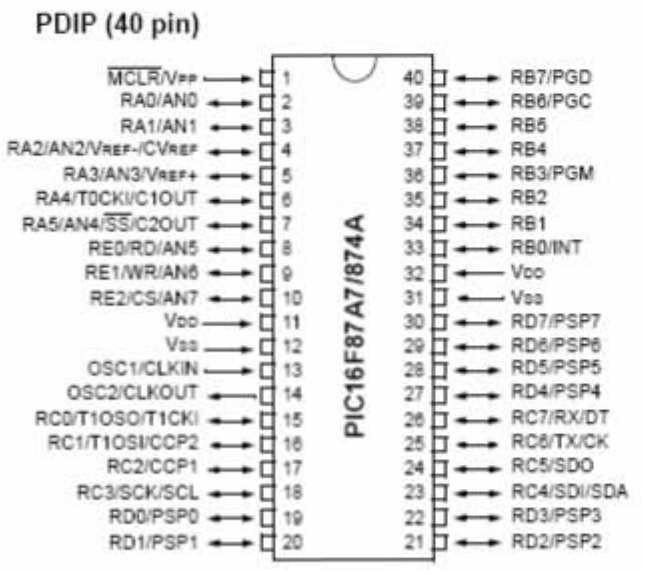
Características Analógicas

- Convertidor Analógico a Digital de 8 canales de 10 bits cada uno.
- Modulo de Comparador Analógico con:
  - ~ 2 comparadores analógicos
  - ~ Módulo programable de voltaje de referencia (Vref)
  - ~ Entrada programable multiplexada desde los dispositivos de entrada y el voltaje de referencia interno
  - ~ Las salidas del comparador son accesibles externamente

Tecnología CMOS

- Tecnología FLASH/EEPROM de alta velocidad
- Diseño completamente estático
- Amplio rango de voltaje de operación (2.0V a 5.5V)
- Bajo consumo de potencia

1.3.1. Aspecto externo del PIC16F877A



La fabricación de este µC está realizada mediante tecnología CMOS en un encapsulado plástico de 40 pines, fig. 1.5. A continuación se describe brevemente la función de cada uno.

Fig.1.5. Diagrama de conexiones de los pines del PIC16F877A

TABLA 1: PIC16F877A DESCRIPCION DE PINES

Nombre del Pin	DIP Pin#	Descripción
OSC1/CLKI	13	Entrada del Oscilador de cristal / Entrada de reloj externo.
OSC2/CLKOUT	14	Salida del Oscilador de cristal / Salida de reloj.
MCLR/VPP MCLR	1	Entrada del Master Clear (Reset) / Entrada del voltaje de programación. Este pin es activo en bajo.
		El Puerto A (PORTA) es un puerto I/O bi-direccional.
RA0/AN0	2	I/O Digital / Entrada Analógica 0.
RA1/AN1	3	I/O Digital / Entrada Analógica 1.
RA2/AN2/VREF-/CVREF	4	I/O Digital / Entrada Analógica 2 / Entrada del voltaje A/D de referencia (bajo) / Salida del Voltaje de VREF del Comparador.
RA3/AN3/VREF+	5	I/O Digital / Entrada Analógica 3 / Entrada del voltaje A/D de referencia (alto).
RA4/T0CKI/C1OUT	6	I/O Digital – Cuando está configurada como salida es de drenaje abierto / Entrada de reloj externo del Timer0 / Salida del Comparador 1.
RA5/SS/AN4/C2OUT	7	I/O Digital / Entrada de selección de SPI / Entrada Analógica 4 / Salida del Comparador 2.
		El Puerto B (PORTB) es un Puerto I/O bi-direccional. El Puerto B puede ser programado por software para activar las resistencias de elevación de voltaje en todas las entradas.
RB0/INT	33	I/O Digital / Interrupción externa.
RB1	34	I/O Digital.
RB2	35	I/O Digital.
RB3/PGM	36	I/O Digital / Pin de habilitación de programación en bajo voltaje ***ICSP***.
RB4	37	I/O Digital.
RB5	38	I/O Digital.
RB6/PGC	39	I/O Digital / Verificador interno y reloj para programación ICSP.
RB7/PGD	40	I/O Digital / Verificador interno y datos para programación ICSP.
		El Puerto C (PORTC) es un puerto I/O bi-direccional.
RC0/T1OSO/T1CKI	15	I/O Digital / Salida del oscilador del Timer1 / Entrada de reloj externo del Timer1.
RC1/T1OSI/CCP2	16	I/O Digital / Entrada del oscilador del Timer1 / Entrada de Captura 2, Salida de Comparación 2, Salida PWM 2.
RC2/CCP1	17	I/O Digital / Entrada de Captura 1, Salida de Comparación 1, Salida PWM 1.
RC3/SCK/SCL	18	I/O Digital / Entrada de reloj serial síncrono para el modo SPI / Entrada de reloj serial síncrono para el modo I^2C.
RC4/SDI/SDA	23	I/O Digital / Entrada de datos de SPI / I/O de datos de I^2C.
RC5/SDO	24	I/O Digital / Salida de datos de SPI.
RC6/TX/CK	25	I/O Digital / Transmisión asíncrona USART / Reloj síncrono del módulo USART.
RC7/RX/DT	26	I/O Digital / Recepción asíncrona USART / Datos síncronos del módulo USART.
		El Puerto D (PORTD) es un puerto I/O bi-direccional o un puerto paralelo esclavo cuando se hace una interfase al bus de un µP.
RD0/PSP0	19	I/O Digital / Datos del puerto paralelo esclavo.
RD1/PSP1	20	I/O Digital / Datos del puerto paralelo esclavo.
RD2/PSP2	21	I/O Digital / Datos del puerto paralelo esclavo.
RD3/PSP3	22	I/O Digital / Datos del puerto paralelo esclavo.
RD4/PSP4	27	I/O Digital / Datos del puerto paralelo esclavo.

Nombre del Pin	DIP Pin#	Descripción
RD5/PSP5	28	I/O Digital / Datos del puerto paralelo esclavo.
RD6/PSP6	29	I/O Digital / Datos del puerto paralelo esclavo.
RD7/PSP7	30	I/O Digital / Datos del puerto paralelo esclavo.
		El Puerto E (PORTE) es un puerto I/O bi-direccional.
RE0/RD/AN5	8	I/O Digital / Control de lectura para el puerto paralelo esclavo / Entrada analógica 5.
RE1/WR/AN6	9	I/O Digital / Control de escritura para el puerto paralelo esclavo / Entrada analógica 6.
RE2/CS/AN7	10	I/O Digital / Control de selección de chip (Chip Select) para el puerto paralelo esclavo / Entrada analógica 7.
VSS	12,31	Ground (GND).
VDD	11,32	Alimentación positiva.
NC	—	Pines no conectados.

1.3.2. Arquitectura del procesador

La arquitectura interna del PIC 16F877A mostrada en la figura 1.6 consta de siete bloques fundamentales:

1) Memoria de programa Flash de 8k x 14bits.
2) Memoria de datos formada por dos áreas. Una RAM donde se alojan registros de propósito general y propósito específico, y otra del tipo EEPROM.
3) Bus de datos con una *ALU* de 8 bits y un registro de trabajo W del que normalmente recibe un operando y envía un resultado. El otro operando puede provenir del bus de datos o del propio código de la instrucción.
4) Diversos recursos conectados al bus de datos, tales como puertos de I/O, temporizadores, etc.
5) Base de tiempos y circuitos auxiliares.
6) Direccionamiento de la memoria de programa en base al Contador de Programa (*PC*) ligado a una Pila (*Stack*) de 8 niveles de profundidad.
7) Direccionamiento directo e indirecto de la memoria.

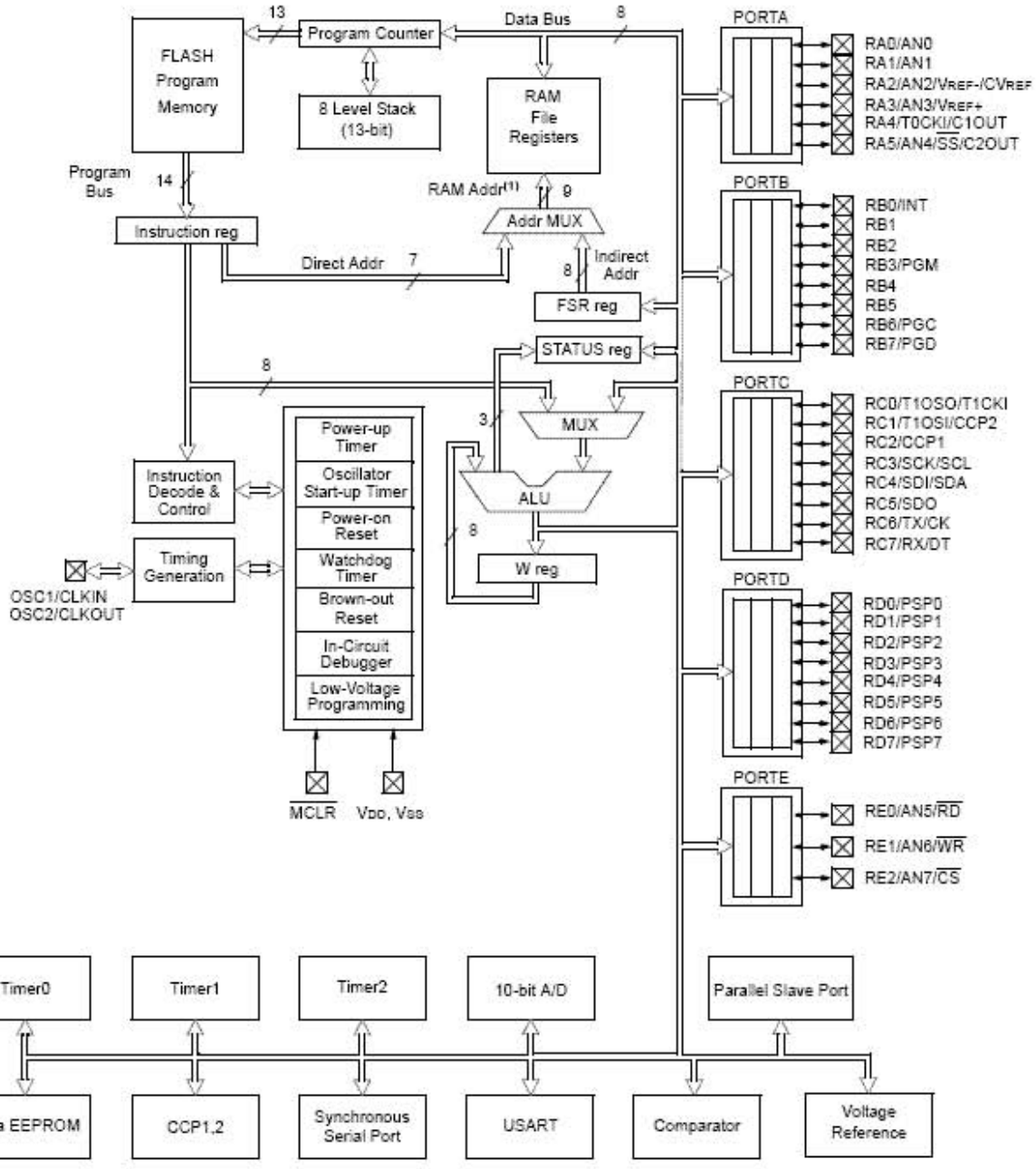


Fig. 1.6. Diagrama de bloques de la estructura interna del PIC16F877A

Un análisis general del funcionamiento del procesador al ejecutar una instrucción comienza con la fase de búsqueda, la cual es iniciada por el contador de programa, facilitando la dirección de la memoria de instrucciones donde se ubica. Su código binario de 14 bits se lee y se carga en el Registro de Instrucciones, desde donde se transfiere al Decodificador y a la Unidad de Control. En ocasiones, dentro del código de la instrucción, existe el valor de un operando

(literal) que se introduce como operando a la ALU, o bien una dirección de la memoria de datos donde reside otro operando.

La ALU es la encargada de realizar la operación lógico-aritmética que implica la instrucción decodificada. Uno de los operandos lo recibe desde el registro W y el otro desde un registro o de la propia instrucción.

1.3.3. Organización de la memoria

Existen tres bloques de memoria en este dispositivo. La Memoria de Programa y la Memoria de Datos que tienen buses separados de modo que la concurrencia de acceso pueda ocurrir y la Memoria de Datos EEPROM.

La memoria de datos puede ser segmentada aún más en la RAM de propósito general y en los registros de Funciones especiales (*SFR*s). La operación de los SFRs que controlan el "núcleo" se describe a continuación. Los SFRs usados para controlar los módulos periféricos se describirán más adelante en otra sección.

1.3.3.1. Memoria de Programa

El contador de Programa

Al igual que todos los registros específicos que controlan la actividad del procesador, el Contador de Programa (PC) está implementado sobre un par de posiciones de la memoria RAM. Cuando se escribe el PC como resultado de una operación de la ALU, los 8 bits de menos peso del PC residen en el registro **PCL**, que ocupa, repetido, la posición 2 de todos los bancos de la memoria de datos. Los bits de más peso, PC<12:8>, residen en los 5 bits de menos peso del registro PCLATH, que ocupa la posición 10 de los bancos de la memoria RAM, fig. 1.7.

En las instrucciones **GOTO**[1] y **CALL**[1] los 11 bits de menos peso del PC provienen del código de la instrucción y los otros dos de los bits PCLATH<4:3>.

[1] Ver Apéndice B

Con los 11 bits que se cargan desde el código de las instrucciones **GOTO** y **CALL**, se puede direccional una página de 2k de memoria. Los bits restantes PC<12:11> tienen la función de indicar el banco de memoria que se encuentra activo, estos bits proceden de PCLATH<4:3>.

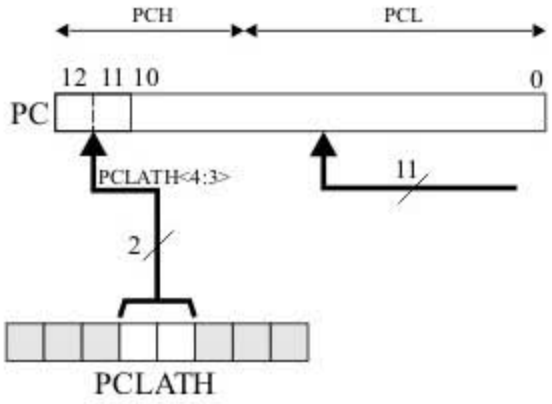


Fig. 1.7. Registro del Contador de Programa

La Pila (Stack), es una zona aislada de las memorias de instrucciones y datos. Tiene una estructura ***LIFO***, en la que el último valor guardado es el primero que sale. Tiene 8 niveles de profundidad, cada uno con 13 bits. Funciona como un buffer circular, de manera que el valor que se obtiene al realizar el noveno "desempilado" es igual al que se obtuvo en el primero.

El vector de Reset

En cualquier dispositivo, un Reset fuerza al PC a ir a la dirección 0h. Está dirección es llamada la "dirección del vector de Reset", desde está dirección la ejecución del programa se ramificará cuando en el dispositivo ocurra un Reset.

Cualquier reset también limpiará el contenido del registro PCLATH. Esto significa que cualquier ramificación en la Dirección del Vector de Reset (0h) saltará a esa ubicación en la página 0 de la memoria de programa.

El vector de Interrupción

Cuando una interrupción es reconocida el PC es forzado a la dirección 0004h. Esta es llamada la "Dirección del Vector de Interrupción". Cuando el PC es forzado hacia el vector de interrupción,

el registro PCLATH no es modificado. Una vez realizada la subrutina de interrupción, y antes de escribir cualquier cosa en el PC, en el registro PCLATH deberá ser cargado con el valor específico de la ubicación de la memoria de programa deseada. Antes de que el registro PCLATH sea modificado por la rutina de servicio de interrupción el contenido de PCLATH puede necesitar ser salvado, de esta manera puede ser restaurado antes de regresar de la rutina de servicio de interrupción.

Stack (La Pila)

El stack permite que ocurra una combinación de hasta 8 llamadas del programa e interrupciones. Contiene la dirección de regreso de una rama en el programa de ejecución.

Los dispositivos como el PIC16F877A tienen un stack de 8 niveles de profundidad por un ancho de 13 bits. El espacio del stack no es parte ni del espacio del programa ni del espacio de datos y el indicador del stack (Stack Pointer) no puede ser leído o escrito. El PC es puesto en la parte superior del stack cuando una instrucción CALL[1] es ejecutada o una interrupción causa una ramificación. El dato que fue guardado en el stack se recupera cuando se da la ejecución de la instrucción RETURN[1], RETLW[1] o un RETFIE[1]. El registro PCLATH no es modificado cuando se introduce o se saca algo del stack.

Después de que han sido puestos en el stack 8 datos, ver fig. 1.8 derecha, el valor del noveno se sobrescribe donde estaba almacenado el primer dato. El décimo dato se sobrescribe donde se encontraba el segundo dato y así sucesivamente.

[1] Ver Apéndice B

PIC16F877A PROGRAM MEMORY MAP AND STACK

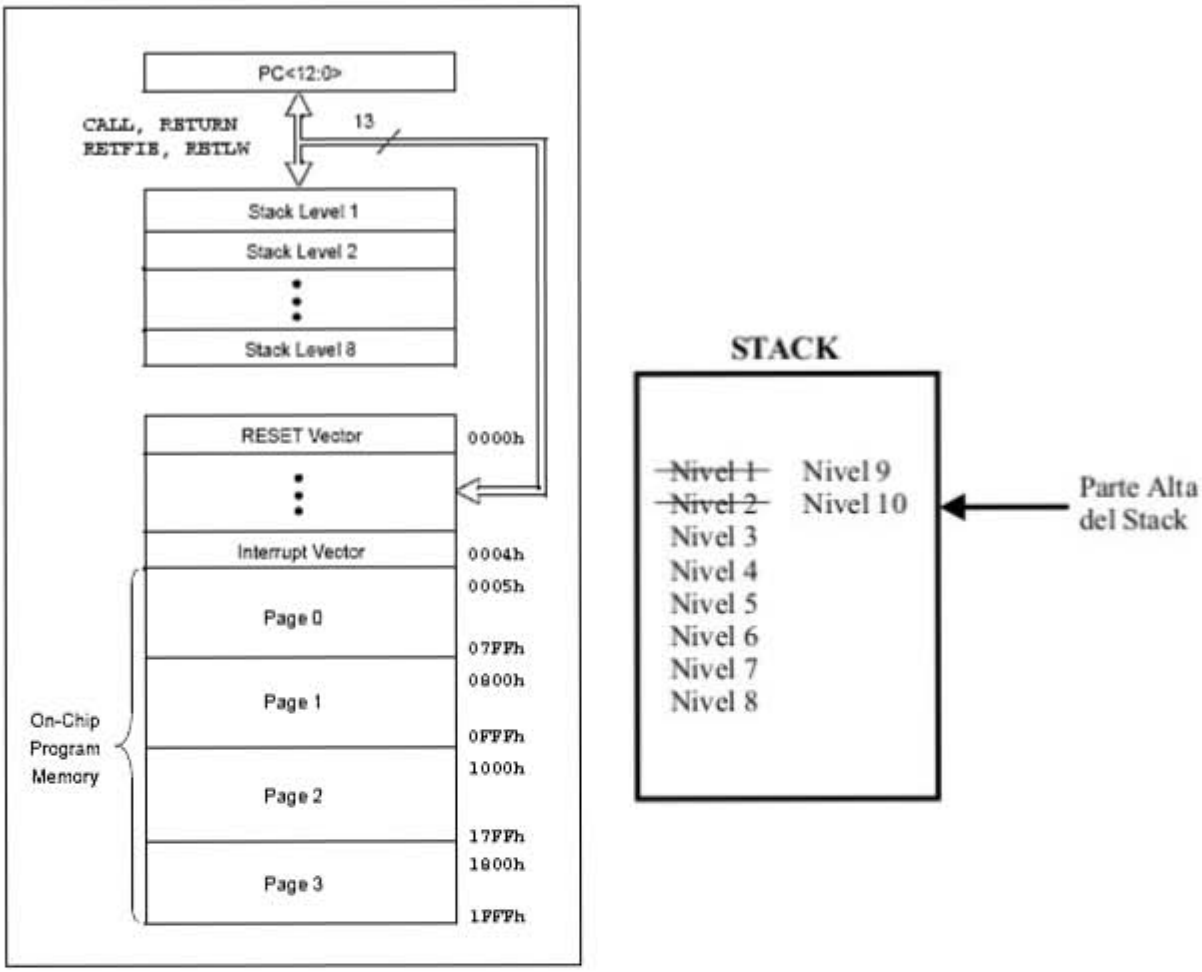
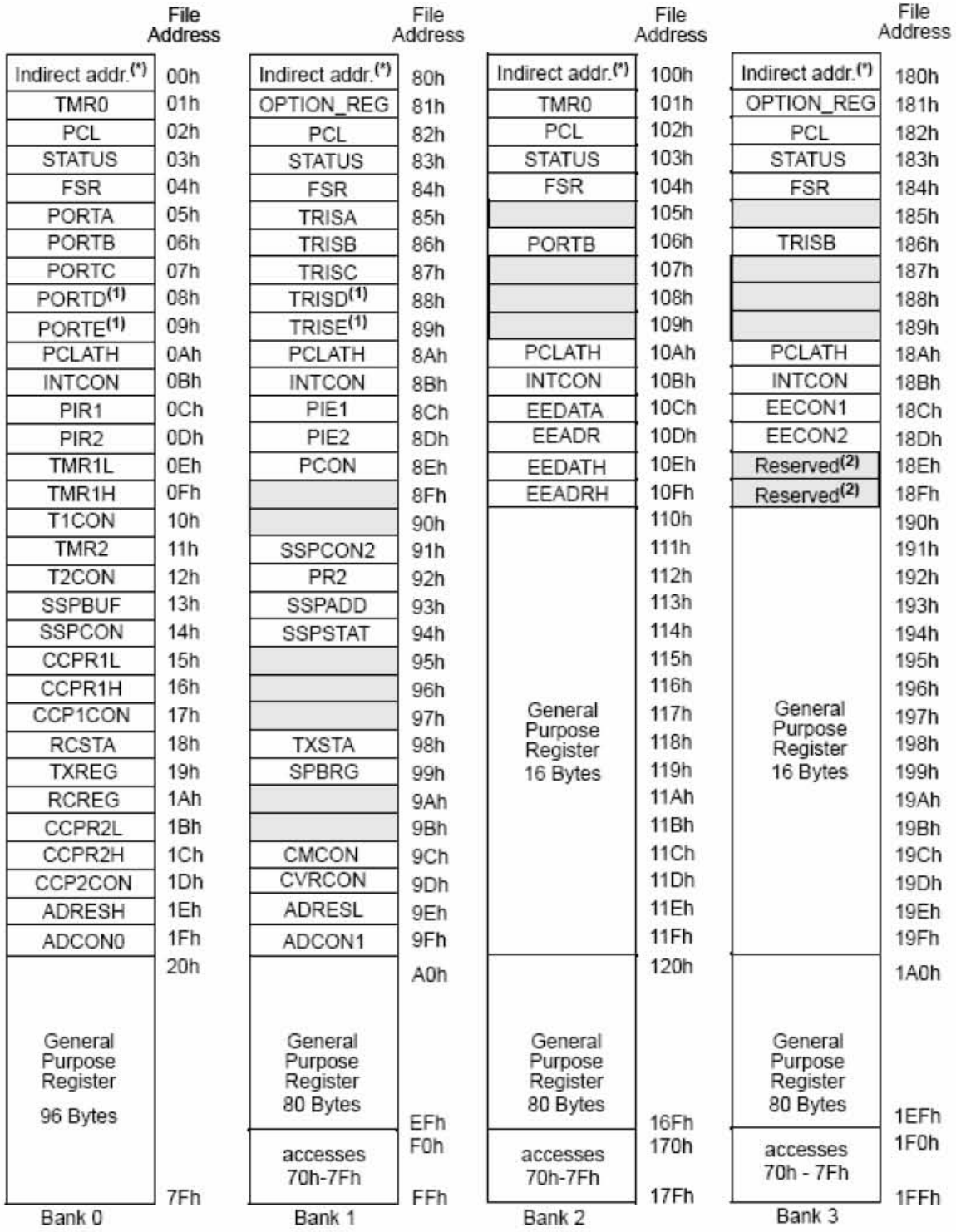


Fig. 1.8. Estructura de la Memora de Programa y del Stack (Pila)

1.3.3.2. Memoria de Datos

La memoria de datos está constituida por el área de los Registros de Funciones Especiales (*SFR*), y el área de los Registros de Propósito General (*GPR*). Los SFR controlan la operación del dispositivo, mientras los registros de propósito general son el área general de almacenamiento de datos y las operaciones de la memoria auxiliar. Ver figura 1.9.

MAPA DE REGISTROS DEL PIC16F876A/877A



Direcciones de la memoria de datos no implementadas, se leen como ’0’.

* No es un registro físico.

Note **1:** Estos registros no están implementados en el PIC16F876A.

2: Estos registros están reservados, mantenga estos registros limpios.

Fig. 1.9. Organización de la Memoria de Datos

El área de los GPR esta en los bancos de memoria para permitir que un número mayor de 96 bytes de la memoria RAM de propósito general puedan ser direccionados. Los SFR son para los registros que controlan los periféricos y las funciones del núcleo. El cambio entre bancos de memoria requiere el uso de bits de control para la selección de banco. Estos bits de control están localizados en el registro **STATUS** (STATUS<7:5>). La siguiente figura muestra el mapa de la organización de la memoria de datos del PIC16F877A.

Para mover los valores de un registro a otro, el valor debe pasar a través del **Registro W**. Esto significa que para todos los movimientos de -registro a registro-, son requeridos dos ciclos de instrucciones.

La memoria de datos en su totalidad puede ser accesada de forma directa o indirecta. El direccionamiento directo puede requerir el uso de los bits **RP1:RP0**[1]. El direccionamiento indirecto requiere el uso del Registro Selector de Archivos (***FSR***). El direccionamiento indirecto usa el bit indicador de registros indirectos (***IRP***) del registro STATUS para acceder a las áreas de la memoria de datos del Banco0/Banco1 o el Banco2/Banco3.

Registros de Propósito General (*GPR*)

Los GPR no están inicializados en el momento del encendido y permanecen sin cambios para todos los otros resets.

El archivo de registros puede ser accesado de cualquiera de las dos formas, ya sea directamente o indirectamente usando el FSR. Algunos dispositivos tienen áreas que son compartidas entre los bancos de la memoria de datos, así que al leer o escribir en esa área aparecerá con el mismo valor en las localidades de memoria a pesar del banco activo. Comúnmente nos referimos a estas áreas como "RAM común". Un ejemplo de estas áreas en el PIC16F877A es el registro STATUS.

[1] Ver detalles en el registro STATUS

Registros de Funciones Especiales (S*FR*)

Los SFR son usados por el CPU y los Módulos Periféricos para controlar las operaciones deseadas de un dispositivo. Estos registros están implementados como RAM Estática (SRAM).

Los SFR pueden ser clasificados en dos grupos, aquellos asociados con las funciones del núcleo y aquellos relacionados a las funciones de los periféricos.

Todos los dispositivos de rango medio tienen en su memoria un área de SFR. Para cambiar entre los bancos de memoria los bits RP0 y RP1 del registro STATUS deben ser configurados, como se muestra en la tabla 2, para accesar al banco deseado. Este es el caso del registro PIR1 que se encuentra en el Banco0 del PIC16F877A. Algunos SFR son inicializados al darse el encendido o una señal de reset mientras otros SFR no son afectados.

Los Bancos

La memoria de datos está particionada en cuatro bancos, como puede observarse en la fig. 1.9. Cada banco contiene Registros de Propósito General y Registros de Funciones Especiales. Cambiar entre estos bancos requiere que los bits RP0 y RP1 del registro STATUS sean configurados, de acuerdo al la tabla 2, para el banco deseado cuando se utiliza direccionamiento directo. El bit IRP en el registro STATUS es usado para direccionamiento indirecto.

Tabla 2: Selección de los Bancos de memoria

Banco Accesado	Directo (RP1:RP0)	Indirecto (IRP)
0	0 0	0
1	0 1	
2	1 0	1
3	1 1	

Cada banco del PIC16F877A, se posee 128 localidades de memoria comprendidas desde la dirección 00h hasta la 7Fh. Las primeras localidades de cada banco están reservadas para los Registros de Funciones Especiales, por lo que a partir de la localidad 00h a la 1Fh para el banco 0, de la 80h a la 9F del banco 1, de la 100h a la 10Fh del banco 2 y de la 180h hasta la 18Fh del banco 3 están dedicadas a estos registros. Las localidades restantes están ocupadas por los

Registros de Propósito General, ver fig. 1.9. Toda la memoria de datos está implementada como RAM estática. Todos los bancos contienen registros de funciones especiales. Algunos registros de funciones especiales del Banco0, que son muy utilizados, están duplicados en otros bancos para la reducción de códigos y un acceso más rápido., éste es el caso de registro Timer0 que está duplicado en el Banco3.

1.3.3.3. Memoria de Datos EEPROM

La Memoria de Datos EEPROM se puede leer y escribir durante la operación normal. Esta memoria no está directamente mapeada en el espacio de los registros de archivos. En lugar de eso esta direccionada indirectamente a través de los Registros de Funciones Especiales. Existen cuatro registros SFR usados para leer y escribir esta memoria. Estos registros son:

- EECON1
- EECON2 (este no es un registro implementado físicamente)
- EEDATA
- EEADR

EEDATA contiene los 8 bits de datos para leer/escribir, y **EEADR** contiene la localidad de la memoria EEPROM que está siendo accesada. Los 8 bits del registro **EEADR** pueden accesar las 256 localidades de Datos de la EEPROM.

EECON1 contiene los bits de control, mientras que el registro EECON2 es usado para inicializar la lectura/escritura.

La memoria de datos EEPROM permite la lectura/escritura por bytes. La escritura de un byte automáticamente borra la localidad y escribe el nuevo dato (borrado antes de escribir). La memoria de datos EEPROM es considerada para una gran cantidad de ciclos de lectura/escritura. El tiempo de escritura es controlado por un temporizador interno del chip. El tiempo de escritura variará con el voltaje y la temperatura.

Protección contra falsas escrituras

Existen condiciones cuando el dispositivo puede no escribir la memoria de datos EEPROM de forma correcta. Para proteger contra falsas escrituras, han sido implementados varios mecanismos en su construcción. En el encendido, el bit **WREN**[1] es puesto en cero. También, el encendido del Temporizador (72ms de duración) previene la escritura de la EEPROM.

Juntos, la secuencia de inicialización y el bit **WREN**, ayudan a prevenir una escritura accidental durante una caída de tensión, rebotes, y malfuncionamiento del software.

[1] Ver detalles en el registro EECON1

2. Motores de CD

2.1. Ingeniería de control de motores

El mercado de los artefactos eléctricos requiere soluciones de bajo costo para el control de motores así como también el mantenimiento de las funciones avanzadas para el funcionamiento eficiente del motor. Es posible cumplir con estos dos requisitos principales mediante un sistema de accionamiento de motor basado en microcontroladores que ayude a los fabricantes de equipos originales a reducir los costos del sistema y a brindar funciones avanzadas para las aplicaciones de control del motor.

Las soluciones de bajo costo basadas en microcontroladores, fueron utilizadas de manera exitosa por los fabricantes de equipos originales para los controles de motor de velocidad variable. Estas soluciones son adecuadas para determinados tipos de motores y aplicaciones que necesitan control mínimo de velocidad o en los casos donde el rendimiento del motor y las fluctuaciones torsionales no son los requerimientos principales.

Las regulaciones de consumo de energía actual y futura demandan artefactos con mayor ahorro energético. Con el fin de cumplir con los requisitos energéticos, se utilizan nuevas tecnologías de motores y topologías de control alternativas. Una topología de control comúnmente utilizada es el inversor, que consiste en extraer voltaje de la línea de corriente alterna (CA) y luego rectificarla para generar un voltaje de corriente continua (CC) y finalmente, se generan voltajes de determinadas amplitudes y frecuencias. Los voltajes de salida del inversor se generan a través de transistores de energía por conmutación, normalmente Transistores Bipolares de Compuerta Aislada (IGBT). Gracias a la topología del inversor, es posible implementar nuevos algoritmos de control para distintos tipos de motores, es decir, control de motores de inducción volts por hertz o control de velocidad de motores de corriente continua sin escobilla.

La solución planteada para las aplicaciones de control de motor apunta a motores de velocidad variable que requieren electrónica de energía por conmutación, tales como los inversores. Los

fabricantes de equipos originales que busquen mejor rendimiento y funcionalidad del motor, además de mantener el bajo costo, deberían tener en cuenta esta solución basada en microcontroladores con módulos especiales para controlar el motor.

Con el objetivo de lograr la funcionalidad de bajo costo del motor en arquitecturas con µC, hoy en día algunas operaciones que llevan demasiado tiempo se realizan mediante el hardware interno del µC con módulos integrados especiales, esto reduce considerablemente el número de componentes externos, reduciendo de esta manera el costo del sistema.

Por lo general, los voltajes de salida de un µC están en el rango de 3,3 y 5 volts, mientras que los IGBT necesitan entre 10 y 20 voltios de compuerta a fuente para encenderse. Se necesita un circuito especial llamado impulsor de compuerta que actúa como interfaz entre el µC y los IGBT, suministrando los voltajes necesarios para accionar los IGBT. En las topologías de inversores, los impulsores de compuerta laterales superiores e inferiores operan los IGBT superiores e inferiores, respectivamente (ver fig. 2.1).

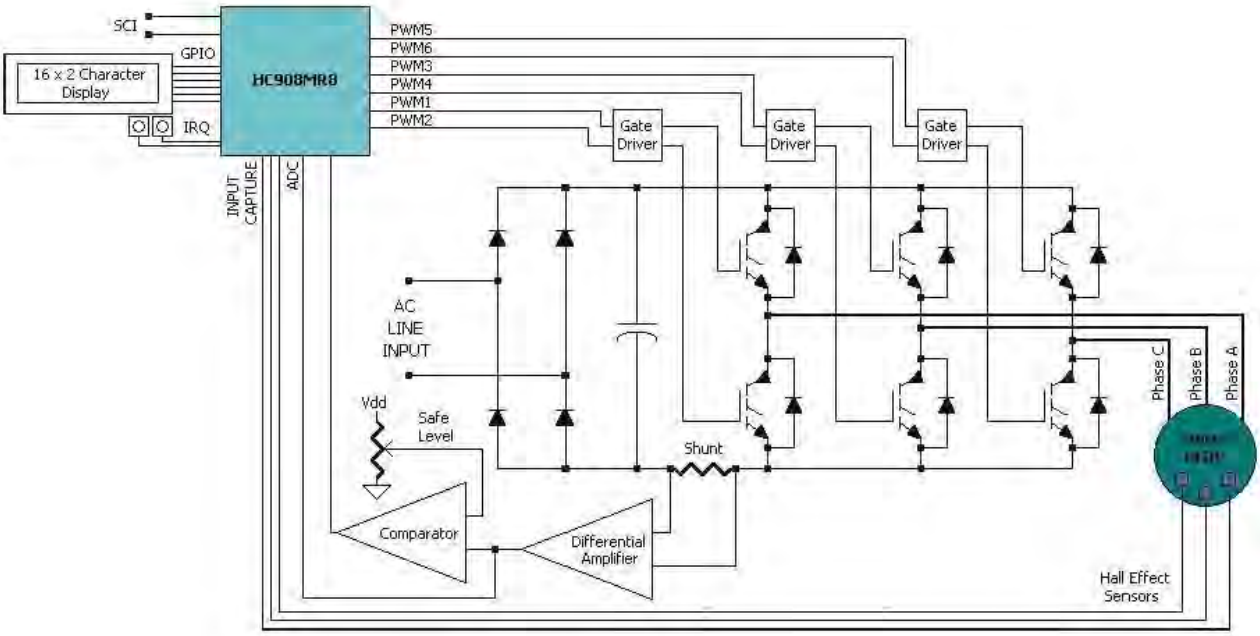


Fig. 2.1. Aplicación de control de motor

Un requerimiento común para las aplicaciones de alta potencia es contar con una barrera de aislamiento entre la parte de control y los dispositivos electrónicos de energía. Por lo general, la

solución consiste en tener aisladores ópticos en el circuito de accionamiento de compuerta, pero la parte del controlador del aislador óptico necesita una corriente para accionar los diodos emisores de luz (LED).

Otro requerimiento para las aplicaciones de control del motor es la tolerancia a fallos del sistema. Estos fallos pueden ocurrir debido a sobrecorrientes y sobrevoltajes, que obligan al µC a apagar todos los dispositivos de potencia para proteger el motor y los dispositivos de potencia. Esta capacidad reduce la complejidad del accionamiento de compuerta, ya que el impulsor de compuerta no necesita tener funcionalidad de desconexión para proteger el sistema contra los fallos.

Otras funciones del µC como por ejemplo la tecnología Flash también representan una reducción de costos para algunas aplicaciones que requieren almacenamiento de datos no volátil, ya que se puede emular el flash interno como una memoria EEPROM para almacenar información.

La nueva tecnología Flash puede ofrecer varias funciones adicionales para los artefactos. Permite la programación en circuito del µC una vez que se haya instalado la placa electrónica en el artefacto, haciendo que se pueda ampliar la capacidad del software y que el fabricante también pueda corregir fallos de software en el momento del testeo. A los artefactos basados en esta tecnología podrán introducirse mejoras y ampliaciones como un servicio de post-venta al cliente.

Los diseñadores de los artefactos, que agregaron valor al sistema total, incluyendo indicadores LED, pantallas con caracteres y botones, utilizaron otros módulos µC, tales como entradas y salidas (I/O) de uso general e interrupciones externas

El control automático de motores de CA ha seguido por costumbre los mismos conceptos que el de CD en cuanto a control convencional. Se usan relevadores electromecánicos para conmutar mecánicamente los contactos que alteran el circuito que controla el motor por diversas razones, esto es, invertir polaridad, conectar o desconectar resistores de interrupción, etc.

Con el desarrollo y mejoramiento continuo de dispositivos de potencia y la disponibilidad de fuentes de activación complejas y precisas para los mismos, este tipo de control demuestra ser

superior al enfoque tradicional en muchos aspectos. Factores importantes como alta confiabilidad, bajo mantenimiento, mejor eficiencia y la ausencia de partes móviles hacen de este control una opción al considerar instalaciones nuevas y la reconversión y actualización de sistemas antiguos.

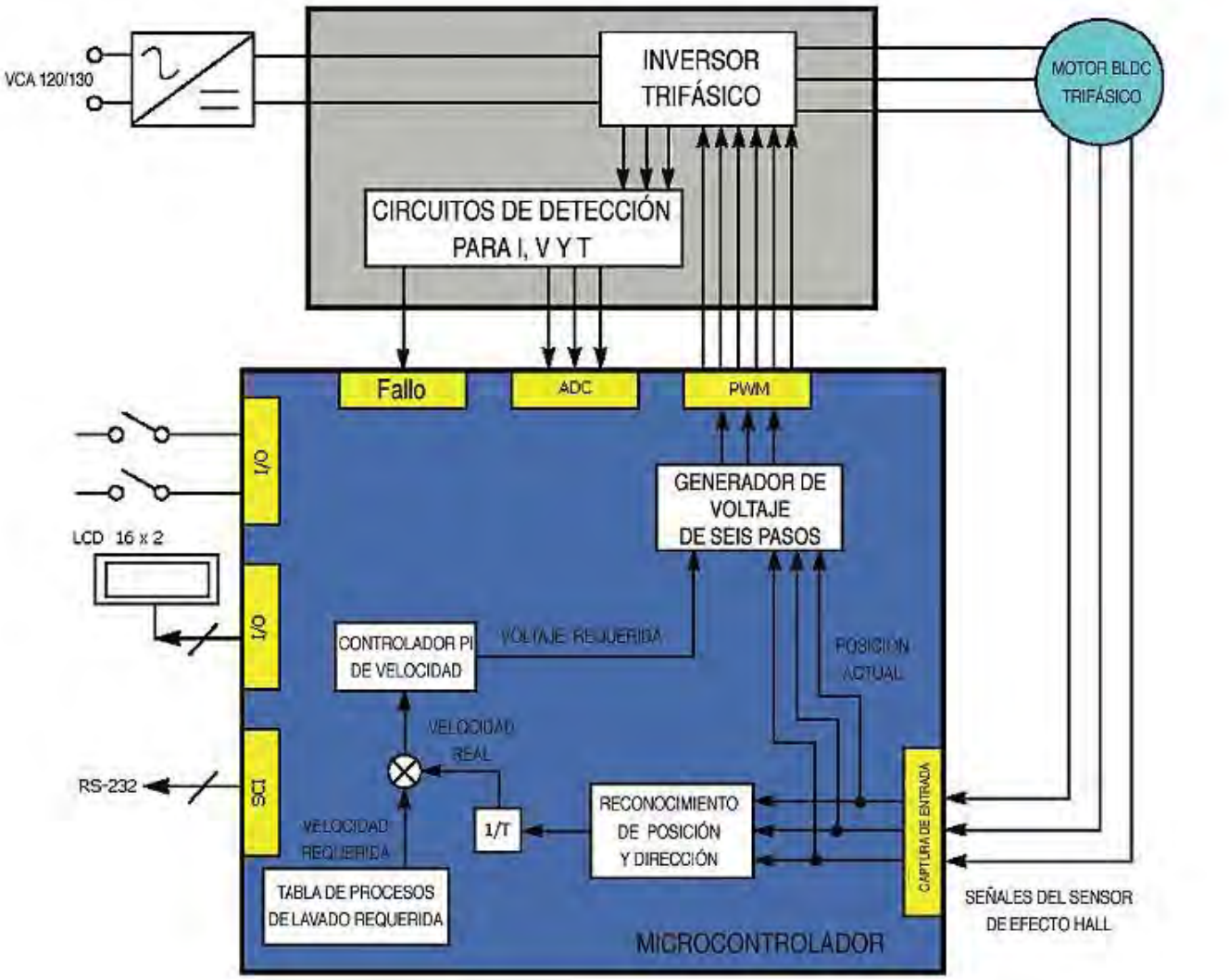


Fig. 2.2. Ejemplo de un sistema de control de motor con componentes agregados

2.2.Funcionamiento del motor de CD

En general, los motores de corriente continua son similares en su construcción a los generadores. De hecho podrían describirse como generadores que funcionan al revés. Cuando

la corriente pasa a través de la armadura de un motor de corriente continua, se genera un par de fuerzas debido a la acción del campo magnético, y la armadura gira. La función del conmutador y la de las conexiones de las bobinas del campo de los motores es exactamente la misma que en los generadores. La revolución de la armadura induce un voltaje en las bobinas de ésta. Este voltaje es opuesto al voltaje exterior que se aplica a la armadura, y de ahí que se conozca como voltaje inducido o fuerza contraelectromotriz. Cuando el motor gira más rápido, el voltaje inducido aumenta hasta que es casi igual al aplicado. La corriente entonces es pequeña, y la velocidad del motor permanecerá constante siempre que el motor no esté bajo carga y tenga que realizar otro trabajo mecánico que no sea el requerido para mover la armadura. Bajo carga, la armadura gira más lentamente, reduciendo el voltaje inducido y permitiendo que fluya una corriente mayor en la armadura.

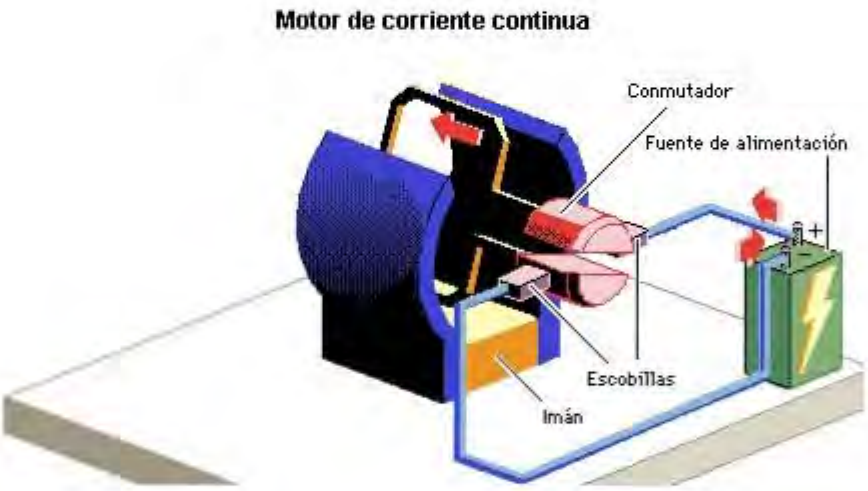


Fig. 2.3. Diagrama básico de un motor de CD

2.2.1. Tipos de motores de CD

Los motores de CD, también conocidos como electromotores, se clasifican según el método de excitación de los mismos, a saber:

- Motores de excitación serie
- Motores de excitación en derivación
- Motores de excitación “compuesta”

Motores de excitación serie

En este tipo de motores, el ***embobinado de campo*** es atravesado por la corriente principal directamente antes de penetrar en el ***embobinado de armadura***, ver figura 2.4; como toda la corriente pasa por éste, la corriente será igual y constante en los dos embobinados.

Como el número de revoluciones del motor se halla en relación con la tensión de la corriente, el número de líneas de fuerza de campo magnético y el de espiras de la armadura dependerán de la intensidad del campo magnético en el que se mueva la armadura del motor.

Esto obliga a tomar ciertas precauciones con el motor serie, pues llega a velocidades realmente peligrosas, que pueden alcanzar valores inadmisibles, con la destrucción de la armadura por los esfuerzos debidos a la fuerza centrífuga desarrollada.

El motor serie posee la propiedad de que su velocidad es inversamente proporcional a la carga aplicada al mismo, y además, su par de arranque es muy elevado por lo que, aumentando la carga progresivamente hasta sobrecargar el motor, se reducen el número de revoluciones del motor, pudiendo llegar a detenerse.

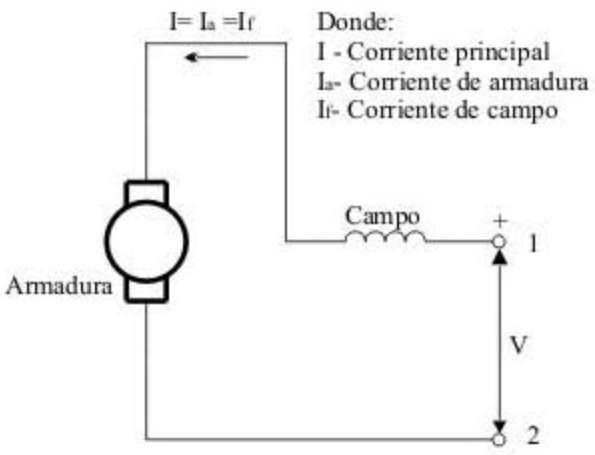


Fig. 2.4. Diagrama del Motor de Excitación Serie

En la figura 2.4 se representa la armadura del motor en serie, alimentado por la corriente de un circuito externo 1,2, que puede ser un reóstato de arranque; de modo que al conectar el motor al circuito externo la resistencia intercalada es muy grande, reduciéndose progresivamente durante la puesta en marcha.

Motores de excitación en derivación

La figura 2.5 muestra un motor en derivación cuyas escobillas se hallan conectadas a los conductores de la red de alimentación; al penetrar la corriente (**I**) en el motor se subdivide en dos fracciones: la más pequeña ($\mathbf{I_f}$) pasa de un borde al otro por el inductor del campo de excitación, y el resto, o sea la corriente principal ($\mathbf{I_a}$), atraviesa la armadura desde la escobilla positiva a la negativa.

Si la alimentación de los conductores al motor es constante, la corriente que circulará por el embobinado de campo (inductancia en derivación) será también constate, así como el número de líneas de fuerza, suponiendo que el voltaje de alimentación es constante (lo que ocurre generalmente).

Como el número de revoluciones del motor es función de estos factores, permanecerá constante durante su funcionamiento, aunque la carga aplicada al mismo sufra variaciones sensibles.

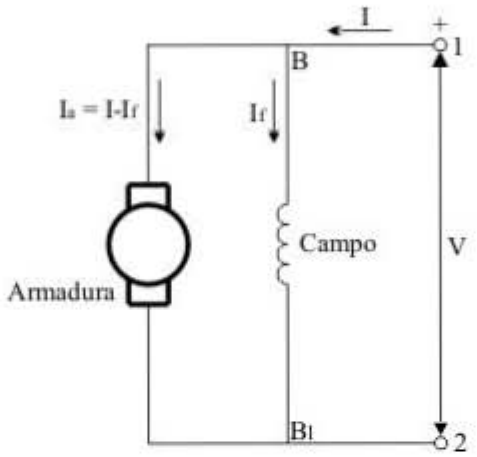


Fig. 2.5. Diagrama del motor de Excitación en Derivación

Puesto que el número de revoluciones depende del voltaje de alimentación, es posible variar el número de revoluciones variando la tensión del motor por medio de un reóstato de excitación.

La figura 2.6 muestra el esquema correspondiente, en el que 1,2 son los conductores de alimentación, B, B1 las escobillas de derivación del embobinado de campo y R el reóstato regulador del voltaje. El número de líneas de fuerza del motor en derivación se modifica variando la intensidad de la corriente de excitación, y para ello se introducirá en el circuito inductor, o sea de excitación, un reóstato regulador cuya resistencia varía según convenga para modificar la intensidad de la fracción de corriente inductora, o sea el número de líneas de fuerza de campo magnético, recordando que al reducir la corriente disminuye el número de líneas de fuerza y aumenta la velocidad del motor; e inversamente, la velocidad disminuye si aumenta la intensidad de la corriente.

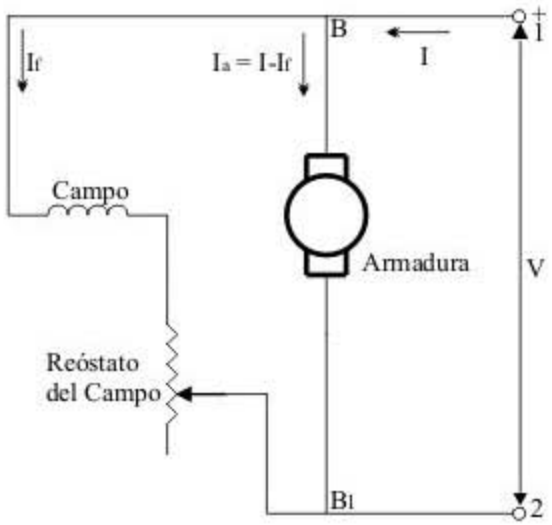


Fig. 2.6. Diagrama del Motor de Excitación en Derivación con Reóstato para variación del número de revoluciones

Motor de excitación compuesta

La figura 2.7 corresponde al esquema de conexiones del motor de excitación compuesta, en el que se agrupan las propiedades características del motor en serie y el de derivación; de modo que funcionando en vacío gira a un número de revoluciones ligeramente superior a la velocidad normal del mismo, pero no llega a velocidades excesivas, pues está restringido por el embobinado en derivación y por el embobinado en serie. Posee un par de arranque superior al par del motor en derivación.

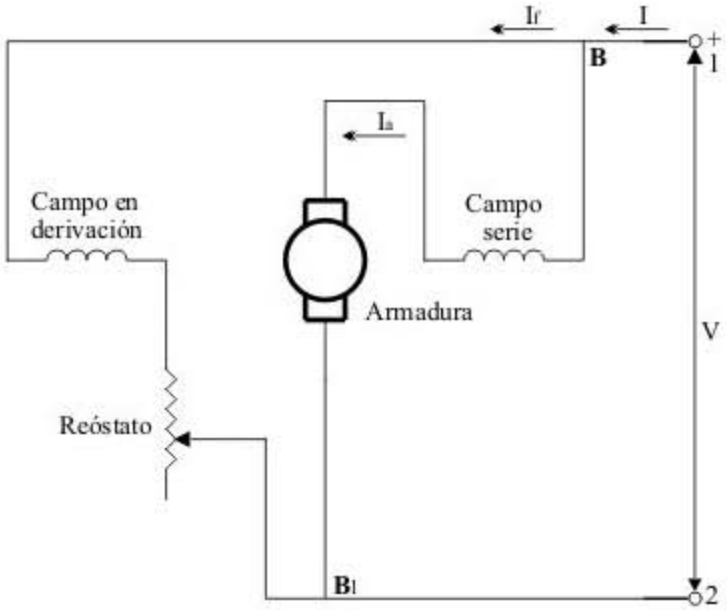


Fig. 2.7. Diagrama del Motor de Excitación Compuesta

En el caso de que los dos embobinados inductores estuvieran conectados de manera que el paso de la corriente en ellos se realizara en sentido contrario, el campo generado por el embobinado en derivación se debilitaría por la influencia del embobinado en serie al aumentar la carga. Entonces aumentaría la velocidad del motor hasta llegar a un valor peligroso.

Así pues, al arrancar un motor compuesto por primera vez, deberá haberse comprobado el sentido de la corriente en los dos embobinados (serie y derivación).

Es importante mencionar que el motor de CD utilizado para realizar este proyecto es del tipo de excitación en derivación. Este fue el seleccionado debido a que mantiene su velocidad constante, que es su característica principal. El motor de excitación compuesta también posee la característica mencionada pero hay que tomar en cuenta que al tener una construcción más compleja, su costo se eleva.

Puesto que en la aplicación a la cual está orientado el proyecto, y otras a las que podría ser adaptado, no se requieren otros tipos de características, es recomendable revisar las consideraciones de selección de un motor expuestas en la siguiente sección.

En caso de necesitarse una variación de aceleración o velocidad, la modificación necesaria deberá realizarse de acuerdo al tipo de control que requiera el motor. Los tipos de control más típicos son mencionados en la sección 2.2.

2.2.2. Características Par-Velocidad

Debido a que el motor serie tiene su campo en serie con la armadura, la corriente de la armadura proporciona la excitación del campo. Consecuentemente a medida que la corriente de la armadura aumenta, el flujo también aumenta proporcionalmente de modo que el par es aproximado al cuadrado de la corriente.

La velocidad del motor serie es sensitiva a la carga, como se muestra en la figura 2.8, y su par de arranque es alto debido a que la alta corriente de arranque también produce un alto valor de flujo.

El motor compuesto serie es adecuado para tranvías eléctricos, autobuses, malacates, grúas y otras aplicaciones que requieren de muy altos pares de arranque, donde la variación de la velocidad no es objetable y donde el motor, bajo una operación normal, siempre impulsa una carga apreciable.

En el motor en derivación la línea alimenta tanto a la armadura como al campo, por lo que la corriente de armadura y el par son proporcionales entre sí. Debido a un debilitamiento en el campo por la reacción a la armadura y el que existe un valor dado de resistencia en el circuito de la armadura, la velocidad es más cercana a ser constante como se muestra en la gráfica.

Dado que el flujo es una función de la corriente del campo, una disminución en la corriente del campo produce un incremento en la velocidad del motor y viceversa.

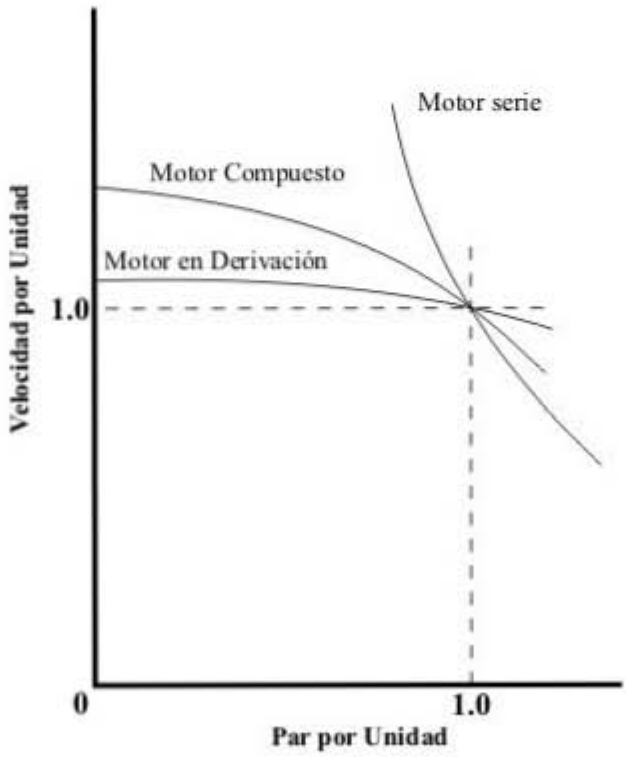


Fig. 2.8. Característica típica del Par-Velocidad de motores

Los motores en derivación se usan en aplicaciones que requieren una velocidad casi constante, pero que no requieren un alto par de arranque. Algunos ejemplos son ventiladores, bombas centrífugas, abanicos, máquinas herramientas.

El motor compuesto tiene un embobinado serie y uno del campo en derivación montados en los polos del campo principal. Un motor tal tiene una velocidad constante regular y es adecuado para cargas oscilatorias con acción de un volante e impulso para cargas tales como bombas de émbolo, prensas punzonadoras, trituradoras, transportadores y malacates. Un pequeño embobinado conocido como embobinado estabilizador se usa en algunos motores en derivación para contrarrestar los efectos desmagnetizantes de la reacción de la armadura previniendo la inestabilidad.

2.3.Tipos de control de motores de CD

En la actualidad, el control de motores de CD se logra por medios automáticos. Cualquier secuencia de arranque que se inicie ya sea al oprimir un botón o cerrar alguna otra forma de dispositivo de control debe tener características automáticas. Incluso el control más sencillo debe restablecerse en forma automática cuando se oprima un botón de para o cuando algún otro dispositivo de demande una interrupción de la corriente.

El propósito de esta sección es el de mencionar algunos de los tipos de control de motores de CD que existen, junto con una breve descripción, además del sistema control desarrollado en este trabajo.

Existen tres formas básicas de controlar un motor que son:

- Control de aceleración
- Control de velocidad
- Control de posición

2.3.1. Control de aceleración

Este método de control es normalmente utilizado durante el periodo de arranque del motor hasta que alcanza su velocidad de operación normal, por esta razón también se les llama arrancadores. Dentro de ésta categoría cabe mencionar dos tipos:

Arrancadores de CD de aceleración en tiempo definido. Esta clase de arrancadores emplea diversas formas de relevadores de retardo para conseguir una secuencia temporal de eliminación de incrementos de la resistencia limitadora de corriente. El retardo deseado se logra al utilizar una de cierto número de variantes:

a) Relevadores de retardo
b) Contactores de retardo
c) Relevadote de retardo que se desactivan después de un periodo de uso
d) Temporizadores de escape mecánico

Arrancadores de CD de aceleración con límite de corriente. Se llaman así para distinguirlos de los anteriores. Existen cuatro variedades específicas:

a) Método de "relevadores de fuerza contraelectromotriz" o de límite de velocidad. En este tipo de control, se conectan relevadores sensibles a voltajes a través de la armadura. Estos relevadores se ponen en circuito a voltajes gradualmente mayores.
b) Método de "relevador de bobina de retención". Mediante el uso de relevadores con dos devanados opuestos, se puede percibir una disminución de corriente de armadura y usarla para cerrar relevadores.
c) Los "arrancadores de relevadores en serie" son otro tipo donde las bobinas de relevador transportan la corriente de armadura. Estos relevadores son de una sola bobina, pero con unas pocas vueltas de alambre grueso y resortes mecánicos relativamente fuertes para mantenerlas desactivadas.

d) Los “arrancadores de bobina de retención y caída de voltaje” combinan las acciones de los incisos a y b, pero tienen desventajas. La corriente de armadura completa no pasa a través de las bobinas del relevador, así que se pueden usar unidades más pequeñas y es factible controlar motores muy grandes.

2.3.2. Control de velocidad

Muchos sistemas relativamente complicados de motor y control trabajan a una velocidad fija una vez que han arrancado. Sin embargo, un gran número de situaciones requiere que se modifique la velocidad.

Existen cuatro medios básicos de controlar la velocidad de un motor de CD, y cada uno tiene un rango aplicable distinto de efectividad:

a) “Control de campo” o, más específicamente, control de flujo magnético de campo.
b) “Control de resistencia de armadura” o control de voltaje de armadura disponible por resistencia en serie.
c) “Control de resistencia de armadura en serie y en derivación”, que utiliza resistencia tanto en serie como en derivación con la armadura. Presenta complicaciones adicionales y mayores pérdidas, pero tiene ciertas características deseables.
d) “Control de voltaje de armadura”, el cual, como su nombre implica, utiliza una fuente de voltaje controlado para la armadura. Esto puede ser en un motor en derivación excitado por separado o en un motor en serie.

2.4. Consideraciones de selección del motor de CD

La lista que se presenta enseguida, y que contiene las consideraciones que se deben estudiar para determinar el alcance de un sistema de control de motores, ayuda a poner el problema de la selección de la etapa de acoplo de potencia del circuito de control en perspectiva. Esto es importante debido a que para la aplicación para la cual se realizaron las pruebas descritas en secciones posteriores la etapa de potencia aísla el circuito de control, que trabaja con baja

potencia, del motor y su fuente de alimentación que pueden ser de potencia media a alta potencia dependiendo del tipo de motor de CD seleccionado.

Consideraciones en cuanto a requerimientos de arranque

(1) Se requiere una ubicación especial del control para seguridad del operario.
(2) Es necesario sincronizar el arranque con otra operación o evitar cierta condición de posición de interferencia, por lo cual se necesite de algún dispositivo de enclavamiento.
(3) El ciclo de trabajo de arranque será frecuente o poco frecuente.
(4) Se tienen requerimientos significativos de par de arranque o tiene la carga una alta inercia.
(5) Es crítico el tiempo del ciclo de arranque.
(6) La suavidad de arranque o el perfil de aceleración un aspecto crítico para la operación.

Consideraciones en cuanto a requerimientos de control de velocidad

(1) La carga requiere de una velocidad constante, de modo que baste con un ajuste inicial de igualación o calibración.
(2) Se tienen ajustes de velocidad seleccionados con regularidad.
(3) Es necesario que la velocidad sea ajustable, como en un proceso cambiante de máquina o un vehículo de velocidad variable.
(4) Se requiere una disminución gradual de velocidad.

Consideraciones en cuanto a los requerimientos de seguridad

(1) Existen requerimientos especiales de salubridad y limpieza, como en entornos de procesamiento de alimentos o salas de operaciones.
(2) Se requiere proveer medios para que en caso de incapacidad del operador se de por resultado una secuencia de paro (control de hombre muerto).
(3) Existe una atmósfera peligrosa o explosiva en torno a la carga, que requiera características especiales.

Consideraciones en cuanto a requerimientos de paro

(1) El motor será detenido por el operador o por un dispositivo automático.

(2) Existe la posibilidad de tener que realizar un paro de emergencia.

(3) El dispositivo tendrá características que requieran que el motor sea detenido por algún medio de frenado o por un par inverso definido (inversión por cambio de polaridad).

(4) Es necesario que el motor retenga la posición después de que se ha detenido.

Todas las consideraciones anteriores deben de tomarse en cuenta si se desea trasladar el proyecto que se describirá en las secciones siguientes a otros entornos de trabajo, modificándolo en caso de ser necesario, ya que este proyecto es una forma básica del control de un motor de CD la cual puede migrarse a aplicaciones más específicas o que interactúen con otros elementos, acción conjunta con otros motores e inclusive cambiando el tipo de motor o su capacidad para aplicaciones de tipo "pesado" siempre teniendo en cuenta la etapa de acoplo entre el control del motor y su carga que sirve para aislar uno del otro..

2.5.Frenado del motor de CD mediante inversión de polaridad

Existen muchas situaciones donde se desea detener o reducir de manera controlada la velocidad de un motor de CD. En estos casos uno de los métodos para realizarlo requiere un par que tenga el sentido opuesto al par de manejo normal. Está situación se conoce como carga de "sobre acarreo". Son ejemplos de esto el descenso de un elevador cargado, un malacate de mina o la carga de un gancho de grúa. En estos casos, si el motor se apaga, la carga continúa acelerándose hacia abajo. De manera similar, si un vehículo que se mueve con electricidad, como un tren, desciende una larga pendiente cuesta abajo, el vehículo completo adquirirá una velocidad peligrosa a menos que se aplique un par. En el caso particular del motor que se utiliza en el proyecto, aunque no tiene una carga conectada ni tiene que impulsar otros elementos, se presenta un efecto de inercia al momento de apagar el motor. Esta inercia provoca que el motor continúe su movimiento por unos instantes después de ser apagado.

Al aplicar un par en el sentido opuesto al del manejo normal durante un periodo de tiempo es posible contrarrestar los efectos de sobre acarreo y de la inercia. Lo que provoca la inserción de este par es la inversión de rotación del motor contraponiéndose a los efectos ya mencionados. Se aplica sólo durante un breve instante de tiempo, suficiente para lograr que iguale el par del sobre acarreo o la inercia de modo que el motor se detenga por completo.

La aplicación del par de sentido opuesto por un periodo mayor al necesario provocará un retroceso en el movimiento del motor haciendo que se pierda la posición deseada y en el caso de aplicaciones específicas pudiendo causar algún mal funcionamiento o dañar el equipo.

Para invertir la dirección de rotación en cualquier motor de CD, es necesario invertir la corriente que pasa por la armadura, con respecto a la corriente del circuito del campo magnético. Esto puede lograrse simplemente invirtiendo la polaridad de la fuente de alimentación del motor aunque puede realizarse por medio de otros dispositivos, como por ejemplo el "Puente H".

Existen otros medios para detener o reducir la velocidad de un motor el más utilizado es de inversión de par a pesar de que éste método posee el gran inconveniente de que, aunque su aplicación es muy sencilla, es complicado determinar el tiempo exacto de aplicación para evitar el giro en contrasentido.

2.6. Aislamiento entre el motor y el circuito de control, el “Puente H”

Los Puentes H (llamados "H BRIDGES" en inglés) son circuitos que permiten controlar motores eléctricos de corriente directa en dos direcciones desde un circuito digital (TTL, CMOS, el puerto de una computadora, desde un microcontrolador, etc...). Se les llama "Puentes H" porque precisamente su forma recuerda vagamente a una letra "H".

El primer problema que se tiene cuando se quiere controlar desde un circuito digital un dispositivo electromecánico (ya sea un motor, un relevador, un alambre muscular, etc) es cómo conectarlo, puesto que NUNCA se debe conectar directamente a la salida digital de un circuito por dos razones que se mencionan a continuación:

Razón 1: Un circuito digital tradicional generalmente no tiene la capacidad de corriente necesaria para hacer que un motor eléctrico de vueltas (y mucho menos capacidad tiene el puerto paralelo de una computadora). Si se conecta directamente un motor aunque sea pequeño, un foco incandescente o algún otro elemento que consuma mucha corriente, lo más probable es que el circuito se sobrecaliente y se queme en unos segundos. La manera más sencilla de manejar un elemento electromecánico pequeño con un circuito digital es utilizar un transistor como interruptor. Así el circuito digital solo habilita y deshabilita el transistor y éste es el que enciende y apaga el motor.

Razón 2: Casi todos los dispositivos electromecánicos (aunque sean pequeños) son muy inductivos. Lo que significa es que no permiten ser apagados de forma instantánea. Es decir, cuando se desconecta un motor eléctrico que está funcionando, el motor (debido a que es un dispositivo inductivo) trata todavía de mantener por una fracción de tiempo la corriente circulando a través de él. Y durante este pequeñísimo tiempo puede generarse una corriente residual en forma de una chispa en la parte del circuito que realizó la desconexión. Esta chispa puede muy fácilmente dañar circuitos electrónicos.

Según el tamaño del motor y según la corriente que esté utilizando, esta chispa puede o no ser visible, pero siempre existe a menos que se coloque en paralelo con el motor un diodo de protección (figura 2.9). Este diodo tiene como finalidad servir de "desahogo" para esta corriente residual que aparece después de que se apaga el motor. Así que, este es el circuito que necesitamos para prender y apagar un motor eléctrico pequeño de corriente directa desde un circuito digital.

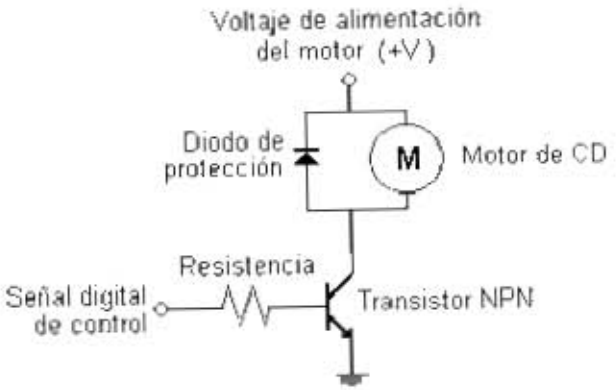


Fig. 2.9. Circuito de desahogo

2.6.1. Diagrama esquemático de un “Puente H

Un puente H es básicamente un arreglo de CUATRO interruptores acomodados de la siguiente manera:

Los interruptores (A, B, C y D) pueden ser transistores bipolares (como el de arriba), MOSFET, JFET, relevadores o de cualquier combinación de elementos. El punto central es que los puentes H se utilizan para que un motor eléctrico de corriente directa funcione en DOS SENTIDOS (adelante y atrás) sin tener que manejar voltajes negativos.

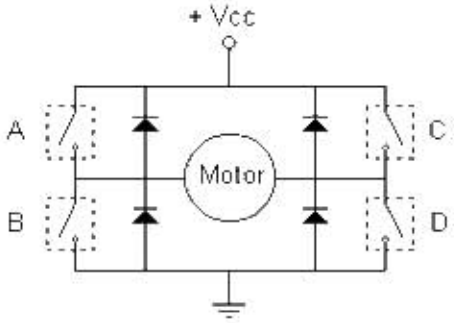


Fig. 2.10. Diagrama básico de un puente H

Si se cierran solamente los contactos A y D la corriente circulará en un sentido a través del motor (o del relevador o de cualquier sistema que esté conectado), y si se cierran solamente los contactos B y C la corriente circulará en sentido contrario. Hay que observar que un puente H también necesita de cuatro diodos de protección para el motor.

3. Desarrollo del sistema de control del motor

3.1.Descripción de Registros Especiales utilizados por el sistema.

Como primer punto en este capítulo, se hablará de los registros SFR que componen la memoria de datos.

Es de gran importancia conocer la operación de cada registro involucrado en el programa de control, pues cada uno posee funciones específicas de control, configuración de módulos o manejo de información. Además existen registros que interactúan entre sí o que son compartidos por varios módulos, un ejemplo de esto es el registro TMR0 que es utilizado por el módulo de comparación, el de conteo y el del temporizador.

3.1.1. Registro de trabajo W

El registro de trabajo W es un registro de 8 bits usado para las operaciones de la ALU. Este no es un registro direccionable.

Dependiendo de la instrucción ejecutada, la ALU puede afectar los valores del bit de carry (C), el bit de dígito de carry (DC), y el bit de cero (Z) en el registro STATUS. En instrucciones de dos operandos, típicamente un operando es el registro W y el otro operando es cualquier otro registro o una constante. En operaciones con instrucciones simples, el operando siempre es ya sea el registro W u otro registro.

Este registro es de gran importancia ya que puede contener datos de cualquiera de los módulos periféricos del μC, contener datos que llegan externamente a los puertos o contiene datos que se están utilizando durante la ejecución del programa.

Dicho de modo simple el registro W puede contener cualquier dato que se esté empleando en ese instante y lo retiene hasta que otro lo reemplaza. Cuando resulta necesario mover el valor de un registro a otro, forzosamente tiene que pasar a través del registro W, es decir, si se desea

mover el dato contenido del registro A al registro B es necesario mover el contenido de A al registro de trabajo W y de ahí se mueve a B.

3.1.2. Registro STATUS (Dirección 03h, 83h, 103h, 183h)

El registro STATUS contiene el estado aritmético de la ALU, el estado del RESET y los bits para la selección de los bancos de la memoria de datos.

El registro STATUS puede ser el destino de cualquier instrucción, como en cualquier otro registro. Si el registro STATUS es el destino para una instrucción que afecte los bits Z, DC o C, entonces la escritura de estos tres bits es deshabilitada. Estos bits son encendidos o apagados de acuerdo al dispositivo lógico. Además, los bits $\overline{TO}$ y $\overline{PD}$ no son de escritura, por lo tanto, el resultado de una instrucción con el registro STATUS como destino puede ser diferente de lo intentado.

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	**RP1**	**RP0**	**$\overline{\text{TO}}$**	**$\overline{\text{PD}}$**	**Z**	**DC**	**C**
Bit 7							bit 0

bit 7 **IRP:** Register Bank Select bit, bit de selección de banco de registros (usado para direccionamiento indirecto)
1 = Banco 2, 3 (100h - 1FFh)
0 = Banco 0, 1 (00h - FFh)

bit 6-5 **RP1:RP0**: Register Bank Select bits, bits de selección de banco de registros (usados para direccionamiento directo)
11 = Bank 3 (180h - 1FFh)
10 = Bank 2 (100h - 17Fh)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)
Cada banco es de 128 bytes

bit 4 **$\overline{\text{TO}}$**: Time-out bit, bit de tiempo finalizado
1 = Después del encendido, una instrucción CLRWDT, o una instrucción SLEEP

0 = Una finalización de tiempo del WDT ha ocurrido

bit 3 $\overline{\mathbf{PD}}$: Power-down bit, bit de apagado

1 = Después de un encendido o por la instrucción CLRWDT

0 = By execution of the SLEEP instruction

bit 2 **Z**: Zero bit, bit de cero

1 = El resultado de una operación aritmética o lógica es cero

0 = El resultado de una operación aritmética o lógica NO es cero

bit 1 **DC**: Digit carry/borrow bit, bit de dígito de carry/borrow (instrucciones ADDWF, ADDLW, SUBLW, SUBWF), (para el borrow, la polaridad es inversa)

1 = Carry del cuarto bit

0 = No hay carry del cuarto bit

bit 0 **C**: Carry/borrow bit, bit de carry/borrow (instrucciones ADDWF, ADDLW, SUBLW, SUBWF)

1 = Hay Carry

0 = No hay carry

Nota: Para el borrow, la polaridad es inversa. Una substracción es ejecutada sumando el complemento a 2 del segundo operando. Para las instrucciones de rotación (RRF,RLF), este bit es cargado ya sea con el bit de mayor orden o con el bit de menor orden del registro fuente.

Etiquetas:			
R = Bit de lectura	W = Bit de escritura	U = bit no implementado, se lee como '0'	
n =Valor al encendido	'1'= Bit encendido	'0'= Bit apagado	x = Bit desconocido

Nota: Los bits C y DC operan como bit de $\overline{\text{borrow}}$ y como bit de $\overline{\text{digit borrow}}$, respectivamente, en la substracción.

3.1.3. Registro OPTION_REG (Dirección 81h, 181h)

El registro OPTION_REG es un registro de lectura y escritura, el cual contiene varios bits de control para configurar el prescaler del TMR0 y el postscaler del Watch Dog Timer (registro de asignación sencilla también conocido como prescaler), la Interrupción Externa INT, TMR0 y las resistencias de elevación del PORTB.

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	**INTEDG**	**T0CS**	**T0SE**	**PSA**	**PS2**	**PS1**	**PS0**
Bit 7							bit 0

bit 7 **RBPU:** PORTB Pull-up Enable bit, bit de habilitación de las resistencias de elevación de voltaje

1 = Las resistencias de PORTB están deshabilitadas

0 = Las resistencias de PORTB están habilitadas por valores individuales de los bit

bit 6 **INTEDG**: Interrupt Edge Select bit, bit de selección de flanco de interrupción

1 = Interrupción en el flanco de subida del pin RB0/INT

0 = Interrupción en el flanco de bajada del pin RB0/INT

bit 5 **T0CS**: TMR0 Clock Source Select bit, bit de selección de la fuente de reloj del TMR0

1 = Pulso del reloj externo en el pin RA4/T0CKI

0 = Instrucción interna de ciclo de reloj (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit, bit de selección del flanco de la fuente de TMR0

1 = Incremento en el flanco de bajada del pin RA4/T0CKI

0 = Incremento en el flanco de subida del pin RA4/T0CKI

bit 3 **PSA**: Prescaler Assignment bit, bit de asignación del prescaler

1 = El prescaler esta asignado al WDT

0 = El prescaler esta asignado al módulo del Timer0

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits, bits de selección de la relación del prescaler

Valor del Bit	Relación del TMR0	Relación del WDT
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Etiquetas:			
R = Bit de lectura	W = Bit de escritura	U = bit no implementado, se lee como ‘0’	
n =Valor al encendido	’1’= Bit encendido	’0’= Bit apagado	x = Bit desconocido

Nota: Cuando se utiliza la programación de bajo voltaje (LVP) y las resistencias de elevación en el PORTB están habilitadas, el bit 3 en el registro TRISB debe ser apagado para deshabilitar la resistencia de elevación en RB3 y asegurar la correcta operación del dispositivo.

3.1.4. Registro INTCON (Dirección 0Bh, 8Bh, 10Bh, 18Bh)

El registro INTCON es de lectura y escritura, el cual contiene varios bits de habilitación y la bandera para el sobre flujo del TMR0, el bit de cambio del puerto RB y las interrupciones externas RB0/INT.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	**PEIE**	**TMR0IE**	**INTE**	**RBIE**	**TMR0IF**	**INTF**	**RBIF**
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit, bit de habilitación de la interrupción global
1 = Habilita todas las interrupciones no mascarables
0 = Deshabilita todas las interrupciones

bit 6 **PEIE**: Peripheral Interrupt Enable bit, bit de habilitación de interrupción periférica
1 = Habilita todas las interrupciones no mascarables periféricas
0 = Deshabilita todas las interrupciones periféricas

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit, bit de habilitación de la interrupción por sobreflujo del TMR0
1 = Habilita la interrupción del TMR0
0 = Deshabilita la interrupción del TMR0

bit 4 **INTE**: RB0/INT External Interrupt Enable bit, bit de habilitación de la interrupción externa RB0/INT
1 = Habilita la interrupción externa de RB0/INT
0 = Deshabilita la interrupción externa de RB0/INT

bit 3 **RBIE**: RB Port Change Interrupt Enable bit, bit de habilitación de la interrupción por cambio de nivel del puerto RB
1 = Habilita la interrupción del cambio de nivel en el puerto RB
0 = Deshabilita la interrupción del cambio de nivel en el puerto RB

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit, bit de bandera de la interrupción por sobreflujo del TMR0

1 = El registro TMR0 tiene sobreflujo (debe ser apagado vía software)

0 = El registro TMR0 no tiene sobreflujo

bit 1 **INTF**: RB0/INT External Interrupt Flag bit, bit de bandera de la interrupción externa RB0/INT

1 = Una interrupción externa de RB0/INT ha ocurrido (debe ser apagado vía software)

0 = Una interrupción externa de RB0/INT no ha ocurrido

bit 0 **RBIF**: RB Port Change Interrupt Flag bit, bit de bandera de la interrupción por cambio de nivel del puerto RB

1 = Al menos uno de los pines RB7:RB4 cambió de estado; una condición de indeterminada en el nivel de voltaje encenderá continuamente el bit. Leer el PORTB terminará esta condición y permitirá al bit ser apagado (debe ser apagado vía software).

0 = Ninguno de los pines RB7:RB4 han cambiado de estado

Etiquetas:			
R = Bit de lectura	W = Bit de escritura	U = bit no implementado, se lee como '0'	
n =Valor al encendido	'1'= Bit encendido	'0'= Bit apagado	x = Bit desconocido

3.1.5. Registro PIE1 (Dirección 8Ch)

El registro PIE1 contiene los bits de habilitación individuales para las interrupciones de los periféricos.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE	**ADIE**	**RCIE**	**TXIE**	**SSPIE**	**CCP1IE**	**TMR2IE**	**TMR1IE**
bit 7							bit 0

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit, bit de habilitación de la interrupción de lectura/escritura del puerto paralelo esclavo

1 = Habilita la interrupción de lectura/escritura de PSP

0 = Deshabilita la interrupción de lectura/escritura de PSP

bit 6 **ADIE**: A/D Converter Interrupt Enable bit, bit de habilitación de la interrupción del convertidor A/D

1 = Habilita la interrupción del convertidor A/D

0 = Deshabilita la interrupción del convertidor A/D

bit 5 **RCIE**: USART Receive Interrupt Enable bit, bit de habilitación de la interrupción de la recepción USART

1 = Habilita la interrupción de la recepción USART

0 = Deshabilita la interrupción del recepción USART

bit 4 **TXIE**: USART Transmit Interrupt Enable bit, bit de habilitación de la interrupción de la transmisión del USART

1 = Habilita la interrupción de la transmisión USART

0 = Deshabilita la interrupción de la transmisión USART

bit 3 **SSPIE**: Synchronous Serial Port Interrupt Enable bit, bit de habilitación de la interrupción del puerto serial síncrono

1 = Habilita la interrupción del SSP

0 = Deshabilita la interrupción del SSP

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit, bit de habilitación de la interrupción de CCP1

1 = Habilita la interrupción de CCP1

0 = Deshabilita la interrupción de CCP1

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit, bit de habilitación de la interrupción de coincidencia de TMR2 con PR2

1 = Habilita la interrupción de coincidencia de TMR2 con PR2

0 = Deshabilita la interrupción de coincidencia de TMR2 con PR2

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit, bit de habilitación de la interrupción por sobreflujo de TMR1

1 = Habilita la interrupción por sobreflujo de TMR1

0 = Deshabilita la interrupción por sobreflujo de TMR1

Etiquetas:			
R = Bit de lectura	W = Bit de escritura	U = bit no implementado, se lee como ‘0’	
n =Valor al encendido	’1’= Bit encendido	’0’= Bit apagado	x = Bit desconocido

Nota: El bit PEIE (INTCON<6>) debe ser encendido para habilitar cualquier interrupción de los periféricos.

3.1.6. Registro PIR1 (Dirección 0Ch)

El registro PIR1 contiene los bits de bandera individuales para las interrupciones de los periféricos.

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF	**ADIF**	**RCIF**	**TXIF**	**SSPIF**	**CCP1IF**	**TMR2IF**	**TMR1IF**
bit 7							bit 0

bit 7 **PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit, bit de bandera de la interrupción de lectura/escritura del puerto serial esclavo
1 = Una operación de lectura o escritura está teniendo lugar (debe ser apagado vía software)
0 = Ninguna lectura o escritura ha ocurrido

bit 6 **ADIF**: A/D Converter Interrupt Flag bit, bit de bandera de la interrupción del convertidor A/D
1 = Una conversión A/D se completo
0 = La conversión A/D no está completa

bit 5 **RCIF**: USART Receive Interrupt Flag bit, bit de bandera de la interrupción de la recepción del USART
1 = El buffer de recepción del USART está lleno
0 = El buffer de recepción del USART está vacío

bit 4 **TXIF**: USART Transmit Interrupt Flag bit, bit de bandera de la interrupción de la transmisión del USART
1 = El buffer de transmisión del USART está vacío
0 = El buffer de transmisión del USART está lleno

bit 3 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag bit, bit de bandera de interrupción del Puerto serial síncrono (SSP)
1 = La condición del SSP ha ocurrido, y debe ser apagado vía software antes de regresar de la rutina de interrupción. Las condiciones que encenderán este bit son:

- SPI
  - Una transmisión/recepción ha tenido lugar.
- I2C Esclavo
  - Una transmisión/recepción ha tenido lugar.

• I2C Maestro

- Una transmisión/recepción ha tenido lugar.
- La condición START fue completada por el modulo SSP.
- La condición STOP fue completada por el modulo SSP.
- La condición de RESET fue completada por el modulo SSP.
- La condición de reconocimiento fue completada por el modulo SSP.
- Una condición de START ocurrió mientras el modulo SSP estaba inactivo.
- Una condición de STOP ocurrió mientras el modulo SSP estaba inactivo

0 = Ninguna interrupción del SSP ha ocurrido

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit, bit de bandera de la interrupción de CCP1

Modo de Captura:

1 = Una captura del registro TMR1 ha ocurrido (debe ser apagado vía software)

0 = Ninguna captura del registro TMR1 ocurrido

Modo de Comparación:

1 = Una coincidencia con el registro TMR1 ha ocurrido (debe ser apagado vía software)

0 = Ninguna coincidencia con el registro TMR1 ha ocurrido

Modo de PWM:

No usado

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit, bit de bandera de la interrupción de coincidencia entre TMR2 y PR2

1 = Una coincidencia entre TMR2 y PR2 ha ocurrido (debe ser apagado vía software)

0 = Ninguna coincidencia entre TMR2 y PR2 ha ocurrido

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit, bit de bandera de la interrupción por sobreflujo del registro TMR1

1 = El registro TMR1 tiene sobreflujo (debe ser apagado vía software)

0 = El registro TMR1 no tiene sobreflujo

Etiquetas:			
R = Bit de lectura	W = Bit de escritura	U = bit no implementado, se lee como '0'	
n =Valor al encendido	'1'= Bit encendido	'0'= Bit apagado	x = Bit desconocido

Nota: Los bits de banderas de interrupción se encienden cuando una condición de interrupción ocurre a pesar del estado de su correspondiente bit de habilitación o del bit de habilitación global, GIE (INTCON<7>). El usuario deberá asegurarse vía software de que los bits apropiados de interrupción estén apagados antes de habilitar una interrupción.

3.1.7. Registro PIE2 (Dirección 8Dh)

El registro PIE2 contiene los bits de habilitación individuales para la interrupción periférica CCP2, la interrupción de colisión de buses SSP, la interrupción de la operación de escritura de la EEPROM y la interrupción del comparador.

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	**CMIE**	**—**	**EEIE**	**BCLIE**	**—**	**—**	**CCP2IE**
bit 7							bit 0

bit 7 **No implementado:** Se lee como ‘0'

bit 6 **CMIE**: Comparator Interrupt Enable bit, bit de habilitación de la interrupción del comparador
1 = Habilita la interrupción del comparador
0 = Deshabilita la interrupción del comparador

bit 5 **No implementado:** Se lee como ‘0'

bit 4 **EEIE**: EEPROM Write Operation Interrupt Enable bit, bit de habilitación de la interrupción de la operación de escritura de la EEPROM
1 = Habilita la interrupción de la escritura de la EEPROM
0 = Deshabilita la interrupción de la escritura de la EEPROM

bit 3 **BCLIE**: Bus Collision Interrupt Enable bit, bit de habilitación de la interrupción de colisión de bus
1 = Habilita la interrupción de colisión de bus
0 = Deshabilita la interrupción de colisión de bus

bit 2-1 **No implementado:** Se lee como ‘0'

bit 0 **CCP2IE**: CCP2 Interrupt Enable bit, bit de habilitación de la interrupción de CCP2
1 = Habilita la interrupción de CCP2
0 = Deshabilita la interrupción de CCP2

Etiquetas:			
R = Bit de lectura	W = Bit de escritura	U = bit no implementado, se lee como ‘0’	
n =Valor al encendido	’1’= Bit encendido	’0’= Bit apagado	x = Bit desconocido

Nota: El bit PEIE (INTCON<6>) debe ser encendido para habilitar cualquier interrupción periférica.

3.1.8. Registro PIR2 (Dirección 0Dh)

El registro PIR2 contiene los bits de bandera para la interrupción CCP2, la interrupción de colisión de bus SSP, la interrupción de la operación de escritura de la EEPROM y la interrupción del comparador.

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	**CMIF**	**—**	**EEIF**	**BCLIF**	**—**	**—**	**CCP2IF**
bit 7							bit 0

bit 7 **No implementado:** Se lee como ‘0'

bit 6 **CMIF**: Comparator Interrupt Flag bit, bit de bandera de la interrupción del Comparador
1 = La entrada del comparador ha cambiado (debe ser apagado vía software)
0 = La entrada del comparador no ha cambiado

bit 5 **No implementado:** Se lee como ‘0'

bit 4 **EEIF**: EEPROM Write Operation Interrupt Flag bit, bit de bandera de la interrupción de la operación de escritura de la EEPROM
1 = La operación de escritura está completa (debe ser apagado vía software)
0 = La operación de escritura está completa o no ha sido comenzada

bit 3 **BCLIF**: Bus Collision Interrupt Flag bit, bit de bandera de la interrupción de colisión de bus
1 = Una collision de bus ha ocurrido en el SSP, cuando está configurado en modo Maestro para I^2C
0 = Ninguna collision de bus ha ocurrido

bit 2-1 **No implementado:** Se lee como ‘0'

bit 0 **CCP2IF**: CCP2 Interrupt Flag bit, bit de bandera de la interrupción de CCP2

Modo de Captura:

1 = Una captura del registro TMR1 ha ocurrido (debe ser apagado vía software)

0 = Ninguna captura del registro TMR1 ha ocurrido

Modo de Comparación:

1 = Una coincidencia del registro TMR1 ha ocurrido (debe ser apagado vía software)

0 = Ninguna coincidencia del registro TMR1 ha ocurrido

Modo de PWM:

No usado

Etiquetas:			
R = Bit de lectura	W = Bit de escritura	U = bit no implementado, se lee como '0'	
n =Valor al encendido	'1'= Bit encendido	'0'= Bit apagado	x = Bit desconocido

Nota: Los bits de banderas de interrupción se encienden cuando una condición de interrupción ocurre a pesar del estado de su correspondiente bit de habilitación o del bit de habilitación global, GIE (INTCON<7>). El usuario deberá asegurarse vía software de que los bits apropiados de interrupción estén apagados antes de habilitar una interrupción.

3.2. Operación de los módulos y puertos que conforman el microcontrolador

Una vez descritos los registros que intervendrán en el programa de control, toca turno a los módulos que conforman el μC y sus funciones.

Cada módulo tiene funciones específicas aunque compartan algunos de sus registros. Es importante mencionar que los módulos que tienen la capacidad de comunicarse con el exterior necesitan, en muchas ocasiones, una configuración específica de alguno de los puertos del μC, como ocurre con el módulo del Timer0 en modo de contador donde es necesario configurar el pin RC0 del PORTC como una entrada.

3.2.1. Memoria de datos EEPROM

Registros EECON1 (Dirección 18Ch) y EECON2 (Dirección 18Dh)

EECON1 es el registro de control para el acceso a la memoria. El bit de control EEPGD determina si el acceso será a la memoria de programa o a la memoria de datos. Cuando está apagado, como cuando está en reset, cualquier operación subsiguiente operará en la memoria de datos. Cuando está encendido, cualquier operación subsiguiente operará en la memoria de programa.

Los bits de control RD y WR inician la lectura y la escritura o borrado, respectivamente. Estos bits no pueden ser apagados, solamente encendidos, por software. Estos bits son apagados vía hardware al completar una operación de lectura o escritura. La restricción para apagar el bit WR vía software previene una prematura terminación de escritura accidental.

El bit WREN, cuando está encendido, permitirá una operación de escritura. En el momento del encendido, el bit WREN es apagado. El bit WRERR es encendido cuando una operación de escritura es interrumpida por un reset del $\overline{\text{MCLR}}$ o del WDT durante la operación normal. En estas situaciones, después de un reset, el usuario puede revisar el bit WRERR y reescribir la localidad. El dato y la dirección permanecerán sin cambios en los registros EEADR y EEDATA.

El bit de interrupción de bandera EEIF en el registro PIR2 se encenderá cuando la escritura sea completada. Este bit debe ser apagado vía software.

EECON2 no es un registro físico. Al leer EECON2 se leerá todo como ceros. El registro EECON2 es usado exclusivamente en la secuencia de escritura de la EEPROM.

R/W-x	U-0	U-0	U-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	**—**	**—**	**—**	**WRERR**	**WREN**	**WR**	**RD**
bit 7							bit 0

bit 7 **EEPGD**: Program/Data EEPROM Select bit, bit de selección de la memoria EEPROM de Programa/datos
1 = Acceso a la memoria de programa
0 = Acceso a la memoria de datos
Este bit no puede cambiarse mientras se realiza una operación de escritura

bit 6-4 **No implementado:** Se lee como '0'

bit 3 **WRERR:** EEPROM Error Flag bit, bit de bandera de error de EEPROM

1 = Una operación de escritura fue prematuramente terminada (cualquier MCLR o WDT reset durante la operación normal)

0 = La operación de escritura fue completada

bit 2 **WREN:** EEPROM Write Enable bit, bit de habilitación de escritura de la EEPROM

1 = Permite los ciclos de escritura

0 = Inhibe la escritura de la EEPROM

bit 1 **WR:** Write Control bit, bit de control de escritura

1 = Inicia un ciclo de escritura. El bit es apagado por hardware una vez que la escritura es completada. El bit WR sólo puede ser encendido (no apagado) en software.

0 = El ciclo de escritura de la EEPROM está completo

bit 0 **RD:** Read Control bit, bit de control de lectura

1 = Inicia la lectura de la EEPROM; RD es apagado por hardware. El bit RD solo puede ser encendido (no apagado) por software

0 = No inició la lectura de la EEPROM

Etiquetas:			
R = Bit de lectura	W = Bit de escritura	U = bit no implementado, se lee como '0'	
n =Valor al encendido	'1'= Bit encendido	'0'= Bit apagado	x = Bit desconocido

Registros EEDATA y EEADR

Cuando se hace interfase con el bloque de la memoria de datos, el registro EEDATA contiene el dato de 8 bits para ser leído o escrito, y el registro EEADR contiene la dirección de la localidad de la memoria EEPROM que está siendo accesada. Este dispositivo (PIC16F877A) tiene una EEPROM de 256 bytes localidades, con un rango de direccionamiento de 00h hasta FFh.

Un byte escrito en la memoria de datos EEPROM automáticamente borra la localidad de memoria y escribe el nuevo dato (borrado antes de escribir).

Tabla 3: Registros Asociados con la memoria de datos EEPROM

Dirección	Nombre	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valor en el encendido	Valor en los demás RESETS
10Ch	EEDATA	Registro de datos de la EEPROM								xxxx xxxx	uuuu uuuu
10Dh	EEADR	Registro de dirección de la EEPROM								xxxx xxxx	uuuu uuuu
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	---0 q000
18Dh	EECON2	Registro de control 2 de la EEPROM (no es un registro físico)								---- ----	---- ----
0Dh	PIR2	—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF	-0-0 0--0	-0-0 0--0
8Dh	PIE2	—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE	-0-0 0--0	-0-0 0--0
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u

Escritura de la memoria de datos EEPROM

Existen dos métodos para realizar la escritura la memoria de datos EEPROM del PIC16F877A, el primero es realizar el proceso establecido por el fabricante para la escritura y el segundo es utilizando el programa de grabación de microcontroladores Ic-Prog1.05C, ya que este posee la capacidad de escribir en la memoria EEPROM de manera directa. A continuación se hace una descripción de cada uno de los procesos, no hay que olvidar que cada uno tiene ventajas y desventajas dependiendo del programa que involucre el uso de la memoria EEPROM.

En el proceso del fabricante están involucrados los registros:

- EECON1 (18Ch)
- EECON2 (18Dh)
- EEDATA (10Ch)
- EEADR (10Dh)
- INTCON (0Bh, 8Bh, 10Bh, 18Bh)

El proceso establecido por el fabricante es un poco complejo y consta de nueve pasos, el usuario debe seguir la secuencia de escritura estrictamente para iniciar la escritura de cada byte, es altamente recomendado que durante el proceso de escritura se deshabiliten las interrupciones. El proceso es el siguiente:

~ Escribir la dirección en el registro EEADR.

~ Se carga el dato de 8 bits que se desea escribir en el registro EEDATA.

~ Se apaga el bit EEPGD (EECON1<7>) para indicarle al programa que se utilizará la memoria de datos.

~ Sen enciende el bit WREN (EECON1<2>) para habilitar la escritura.

~ Deshabilitar las interrupciones apagando el bit GIE (INTCON<7>). (Si se utilizan).

~ Se ejecuta una secuencia especial de cinco instrucciones, esto es para dar tiempo al proceso de escritura pues la memoria EEPROM es lenta en comparación con otras funciones del μC, los valores que se utilizan para cargar el registro EECON2 son dados por el fabricante especialmente para esta operación. La secuencia es:

  - Escribir un 55h en el registro EECON2 en dos pasos (primero al registro de trabajo W, luego a EECON2)
  - Escribir un AAh en el registro EECON2 en dos pasos (primero al registro de trabajo W, luego a EECON2)
  - Encender el bit WR (EECON1<1>)

~ Habilitar las interrupciones encendiendo el bit GIE (INTCON<7>). (Si se utilizan).

~ Apagar el bit WREN (EECON1<2>) para deshabilitar la escritura.

~ Al completar el ciclo de escritura, el bit WR (EECON1<1>) es apagado y el bit de bandera de interrupción EEIF (PIR2<4>) es encendido. (EEIF debe ser apagado vía software).

Es importante hacer notar que para cada localidad de memoria que se desee escribir deberá repetirse el proceso de nueve pasos completo. Un ejemplo del ciclo de escritura de la memoria EEPROM en programación es:

Secuencia Requerida

```
BCF     STATUS, RP0     ;Banco 2
MOVF    DATA_EE_ADDR,W ;Memoria de Datos
MOVWF   EEADR           ;Dirección a escribir
MOVF    DATA_EE_DATA,W ;Valor de la Memoria de Datos
MOVWF   EEDATA          ;a escribir
BSF     STATUS,RP0      ;Banco 3
BCF     EECON1,EEPGD    ;Indicar la memoria de datos
BSF     EECON1,WREN     ;Habilitar la escritura
BCF     INTCON,GIE      ;Deshabilitar interrupciones
MOVLW   55h             ;
MOVWF   EECON2          ;Escribe 55h
MOVLW   AAh             ;
MOVWF   EECON2          ;Escribe AAh
BSF     EECON1,WR       ;Enciende el bit WR para
                        ;iniciar la escritura
BSF     INTCON,GIE      ;Habilita las interrupciones
BCF     EECON1,WREN     ;Deshabilita la escritura
```

El segundo método de escritura de la memoria EEPROM es utilizando el programa Ic-Prog1.05C. Cabe resaltar que este método sólo le permite al programa usar los datos de modo estático y es inútil si se desean utilizar de modo dinámico, es decir, sólo se pueden leer datos de la memoria y no es posible leer y luego escribir en la memoria. Esta forma es más sencilla pues al abrir el programa y configurarlo para el dispositivo deseado, en este caso el PIC16F877A, este muestra un mapa de la memoria EEPROM, lo único que se tiene que hacer es escribir lo que se desea de manera directa en el mapa de memoria e indicar al programa que lo descargue al μC.

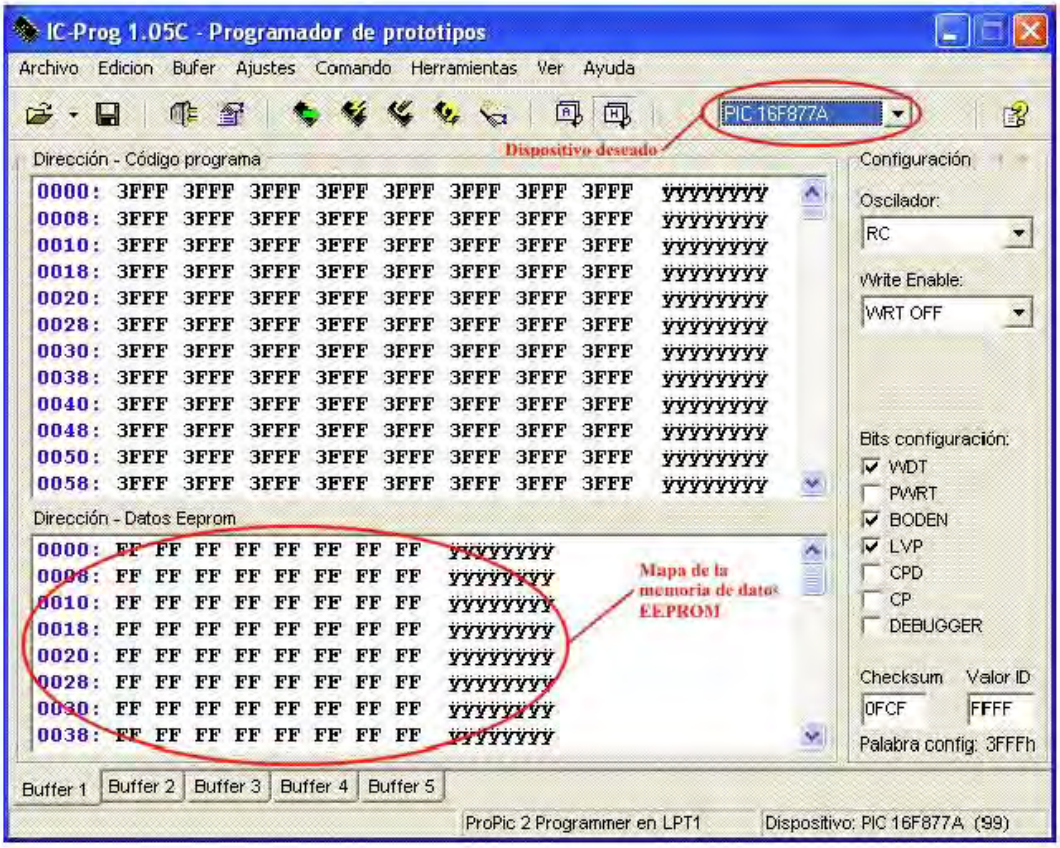


Fig. 3.1. Presentación del programa IC-Prog 1.05C

En la zona marcada con "ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ" es posible escribir directamente caracteres alfanuméricos y el programa mismo se encarga de traducirlos a código ASCII, o se pueden escribir en la zona de "FF" el dato que corresponda a cada localidad de memoria, a un costado puede observarse la dirección correspondiente a cada localidad de memoria.

Aunque éste método es más sencillo y ahorra líneas de programación, tiene la desventaja de que cuando se cambia el código del programa es necesario escribir de nuevo los datos de la memoria EEPROM pues no se pueden guardar la combinación de ambos como un archivo; sólo se puede hacer eso desde el código del programa principal como en el método anterior.

Lectura de la memoria de datos EEPROM

El proceso de lectura de la memoria EEPROM involucra los siguientes registros:

- EECON1 (18Ch)
- EEDATA (10Ch)
- EEADR (10Dh)

Para lograr una lectura correcta de la memoria se deben realizar cuatro pasos, los cuales deben seguirse en estricto orden ya que de otro modo la lectura podría no realizarse o sería incorrecta.

La secuencia es la siguiente:

~ Se escribe la dirección que se desea leer en el registro EEADR.

~ Se apaga el bit EEPGD (EECON1<7>) para indicarle al programa que se quiere accesar a la memoria de datos.

~ Se enciende el bit RD (EECON1<0>) para realizar la operación de lectura de la localidad de memoria deseada.

~ El dato leído se almacena en el registro EEDATA, el cuál mantendrá el dato hasta que se realice una nueva lectura y reciba otro dato.

Para observar un ejemplo práctico de las funciones de escritura y lectura de la memoria EEPROM revisar la sección 3.5 “Programas de prueba de los elementos del sistema”.

3.2.2. PORTB (Dirección 06h, 106h) y el Registro TRISB (Dirección 86h, 186h)

El puerto B (PORTB) es un puerto bi-direccional con un ancho de 8 bits. El correspondiente registro de dirección de datos es TRISB. Encendiendo un bit (=1) en TRISB hará que el pin

correspondiente del PORTB funcione como una entrada. Apagando un bit (=0) en TRISB hará que el pin correspondiente del PORTB funcione como una salida.

Cada pin del PORTB tiene una resistencia interna de elevación de voltaje. Un solo bit de control puede encender todas las resistencias de elevación. Esto es ejecutado apagando el bit RBPU (OPTION_REG<7>). Las resistencias de elevación son automáticamente apagadas cuando el pin del puerto es configurado como una salida. Las resistencias de elevación están deshabilitadas en un reset.

Tabla 4: Funciones de PORTB

Nombre	Bit	Tipo de Buffer	Función
RB0/INT	bit0	TTL/ST**(1)**	Pin de entrada/salida o entrada de interrupción externa. Resistencia de elevación interna programable por software.
RB1	bit1	TTL	Pin de entrada/salida. Resistencia de elevación interna programable por software.
RB2	bit2	TTL	Pin de entrada/salida. Resistencia de elevación interna programable por software.
RB3/PGM**(3)**	bit3	TTL	Pin de entrada/salida o pin de programación en modo LVP. Resistencia de elevación interna programable por software.
RB4	bit4	TTL	Pin de entrada/salida (con interrupción al cambio). Resistencia de elevación interna programable por software.
RB5	bit5	TTL	Pin de entrada/salida (con interrupción al cambio). Resistencia de elevación interna programable por software.
RB6/PGC	bit6	TTL/ST**(2)**	Pin de entrada/salida (con interrupción al cambio) o pin de verificación interna. Resistencia de elevación interna programable por software. Reloj de programación serial.
RB7/PGD	bit7	TTL/ST**(2)**	Pin de entrada/salida (con interrupción al cambio) o pin de verificación interna. Resistencia de elevación interna programable por software. Datos de programación serial.

TTL = Entrada TTL, ST = Entrada Schmitt Trigger

Nota **1:** Este buffer es una entrada Schmitt Trigger cuando está configurada como una interrupción externa.

2: Este buffer es una entrada Schmitt Trigger cuando es usado en modo de programación serial.

3: La programación ICSP a bajo voltaje (LVP) es habilitada por default, la cual deshabilita las funciones de entrada/salida de RB3. El LVP debe ser deshabilitado para habilitar RB3 como un pin de I/O.

3.2.3. PORTC (Dirección 07h) y el Registro TRISC (Dirección 87h)

El puerto C (PORTC) es un Puerto bi-direccional con un ancho de 8 bits. El correspondiente registro de dirección de datos es TRISC. Encendiendo un bit (=1) en TRISC hará que el pin

correspondiente del PORTC funcione como una entrada. Apagando un bit (=0) en TRISC hará que el pin correspondiente del PORTC funcione como una salida.

Tabla 5: Funciones de PORTC ST = Entrada Schmitt Trigger

Nombre	Bit	Tipo de Buffer	Función
RC0/T1OSO/T1CKI	bit0	ST	Pin de entrada/salida o salida del oscilador del Timer1/entrada de reloj del Timer1.
RC1/T1OSI/CCP2	bit1	ST	Pin de entrada/salida o entrada del oscilador o entrada de Captura2/salida de Comparación2/salida de PWM2.
RC2/CCP1	bit2	ST	Pin de entrada/salida o entrada de Captura1/salida de Comparación1/salida de PWM1.
RC3/SCK/SCL	bit3	ST	RC3 también puede ser el reloj serial síncrono para los modos SPI e I^2C.
RC4/SDI/SDA	bit4	ST	RC4 también puede ser la entrada de datos de SPI (en modo SPI) o I/O de datos (en modo I^2C).
RC5/SDO	bit5	ST	Pin de entrada/salida o salida de datos del puerto serial síncrono.
RC6/TX/CK	bit6	ST	Pin de entrada/salida o transmisión asíncrona del USART o reloj síncrono.
RC7/RX/DT	bit7	ST	Pin de entrada/salida o recepción asíncrona del USART o dato síncrono.

El PORTC está multiplexado con varias funciones periféricas (ver tabla 5), por ejemplo, los pines RC6 y RC7 pueden ser utilizados como pines de entrada/salida, también pueden ser utilizados como transmisión y recepción del módulo USART en modo asíncrono o como reloj y datos en modo síncrono. Además, los pines de PORTC tienen buffers de entrada de tipo Schmitt Trigger.

Cuando se habilitan las funciones periféricas se deben tomar precauciones al definir los bits de TRISC para hacer cada pin de entrada o salida, ya que los distintos periféricos que interactúan con este puerto pueden modificar el valor del registro TRISC imponiéndose a los bits previamente cargados en TRISC. Al ocurrir este efecto las instrucciones que modifiquen la lectura o escritura (BSF, BCF, XORWF) que tengan a TRISC como destino, deben ser evitadas. El usuario deberá referirse a la sección del periférico correspondiente en el manual para encender correctamente los bits de TRISC.

3.2.4. Módulo del Timer0

El módulo del Timer0 tiene dos modos de operación, como temporizador y como contador, y tiene las siguientes características:

- Tiene un temporizador/contador de 8 bits
- Es de lectura y escritura
- Posee un prescaler de 8 bits programable por software
- Selección de reloj interno o externo
- Interrupción por sobreflujo en el cambio de FFh a 00h
- Selección del flanco para el reloj externo

La figura 3.2 es un diagrama de bloques del módulo del Timer0 y el prescaler junto con el WDT.

El modo de temporizador es seleccionado apagando el bit T0CS (OPTION_REG<5>). En el modo de temporizador, el módulo del Timer0 se incrementará cada ciclo de instrucción (sin el prescaler). Si el registro TMR0 es escrito, el incremento es inhibido por los siguientes dos ciclos de instrucciones. El usuario puede trabajar esto ajustando el valor escrito en el registro TMR0.

El modo de contador es seleccionado encendiendo el bit T0CS (OPTION_REG<5>). En el modo de contador, el Timer0 se incrementará cada flanco de reloj, ya sea de subida o de bajada, del pin RA4/T0CKI. El conteo se almacena en el registro TMR0 (dirección 01h, 101h). El incremento del flanco está determinado por el bit de selección de flanco del Timer0, T0SE (OPTION_REG<4>). Apagando el bit T0SE se selecciona el flanco de subida y encendiéndolo el flanco de bajada.

Interrupción del Timer0

La interrupción del TMR0 es generada cuando se presenta un sobreflujo en el registro TMR0 de FFh a 00h. Este sobreflujo enciende el bit TMR0IF (INTCON<2>). La interrupción puede ser mascarada apagando el bit TMR0IE (INTCON<5>). El bit TMR0IF debe ser apagado vía software. La interrupción del TMR0 no puede despertar el procesador del modo SLEEP, el cual ofrece un

modo de bajo consumo de corriente. El usuario puede "despertar al dispositivo del modo SLEEP mediante un reset externo, utilizando el watchdog timer o mediante una interrupción.

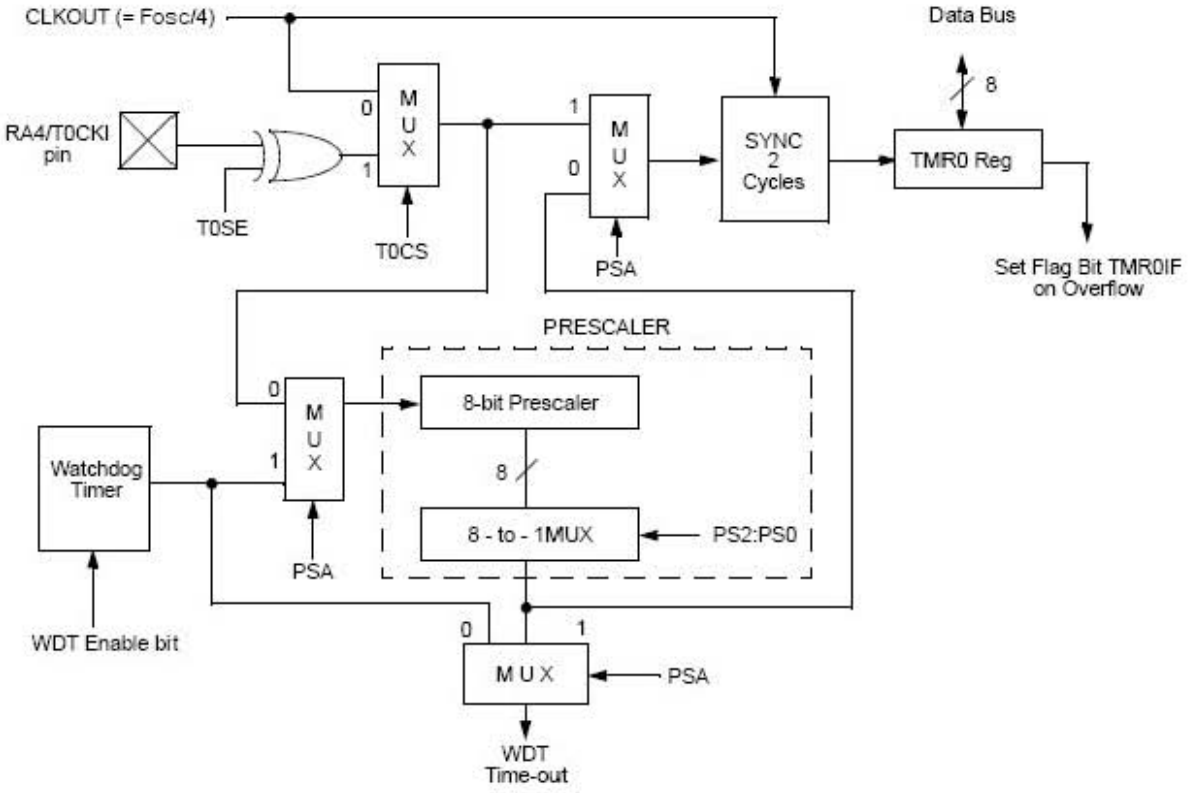


Fig. 3.2. Diagrama de bloques del Timer0

El Prescaler

Solo hay un prescaler disponible, el cual es mutuamente y exclusivamente compartido entre el módulo del Timer0 y el Watchdog Timer. Una asignación del prescaler para el módulo del Timer0 significa que no hay un prescaler disponible para el WDT y viceversa. El prescaler no puede ser leído ni escrito.

Los bits PSA y PS2:PS0 (OPTION_REG<3:0>) determinan la asignación del prescaler y la razón del prescaler. Cuando se asigna al módulo del Timer0, todas las instrucciones que escriban en el registro TMR0 (por ejemplo CLRF, MOVWF, BSF, etc.) pondrán en cero el prescaler. Cuando esté asignado al WDT, una instrucción CLRWDT pondrá en cero el prescaler junto con el WDT.

Tabla 6: Registros Asociados con el módulo del Timer0

Dirección	Nombre	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valor en el encendido	Valor en los demás RESETS
01h, 101h	TMR0	Registro del módulo del Timer0								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
81h, 181h	OPTION _REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

x = Desconocido, u = Desconocido, - = Dirección no implementada leída como '0'. Las celdas sombreadas no son usadas por el Timer0

3.2.5. Módulo del Timer1

El módulo del Timer1 (fig. 3.3) es un temporizador / contador de 16 bits que consiste de dos registros de 8 bits (TMR1H y TMR1L), los cuales son de lectura y escritura. El par de registros TMR1 se incrementan desde 0000h hasta FFFFh y regresan a 0000h. La interrupción de TMR1, si está habilitada, es generada en el sobreflujo y es mostrada en el bit de bandera TMR1IF (PIR1<0>). Esta interrupción puede ser habilitada o deshabilitada encendiendo o apagando el bit de habilitación de la interrupción TMR1IE (PIE1<0>).

El Timer1 puede operar en uno de dos modos:

- Temporizador
- Contador

El modo de operación está determinado por el bit de selección del reloj, TMR1CS (T1CON<1>).

En el modo de temporizador, el Timer1 se incrementa cada ciclo de instrucción. En el modo de contador, este se incrementa en cada flanco de subida de la entrada de reloj externo.

El Timer1 puede ser habilitado o deshabilitado encendiendo o apagando el bit de control TMR1ON (T1CON<0>).

Cuando el oscilador del Timer1 está habilitado, T1OSCEN (T1CON<3>), los pines RC1 y RC0 se vuelven entradas. De esta forma, el valor de TRISC<1:0> es ignorado, y estos pines son leídos como '0'.

T1CON: Registro de Control del TIMER1 (Dirección 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	**—**	**T1CKPS1**	**TICKPS0**	**T1OSCEN**	$\overline{\text{T1SYNC}}$	**TRM1CS**	**TMR1ON**
bit 7							bit 0

bit 7-6 **No implementado:** Se lee como '0'

bit 5-4 **T1CKPS1:T1CKPS0**: Timer1 Input Clock Prescale Select bits, bits de selección del prescaler de la entrada de reloj del Timer1

11 = 1:8 Valor del Prescaler

10 = 1:4 Valor del Prescaler

01 = 1:2 Valor del Prescaler

00 = 1:1 Valor del Prescaler

bit 3 **T1OSCEN**: Timer1 Oscillator Enable Control bit, bit de control de la habilitación del oscilador del Timer1

1 = El oscilador está habilitado

0 = El oscilador está apagado

bit 2 **T1SYNC**: Timer1 External Clock Input Synchronization Control bit, bit de control de sincronización de la entrada de reloj externa del Timer1.

Cuando TMR1CS = 1:

1 = No sincroniza la entrada del reloj externo

0 = Sincroniza la entrada del reloj externo

Cuando TMR1CS = 0:

Este bit es ignorado. El Timer1 usa el reloj interno cuando TMR1CS = 0.

bit 1 **TMR1CS**: Timer1 Clock Source Select bit, bit de selección de la fuente del reloj del Timer1

1 = Reloj externo desde el pin RC0/T1OSO/T1CKI (en el flanco de subida)

0 = Reloj interno (FOSC/4)

bit 0 **TMR1ON**: Timer1 On bit, bit de encendido del Timer1

1 = Habilita el Timer1

0 = Detiene el Timer1

Etiquetas:			
R = Bit de lectura	W = Bit de escritura	U = bit no implementado, se lee como ‘0’	
- n = Value at POR	’1’= Bit encendido	’0’= Bit apagado	x = Bit desconocido

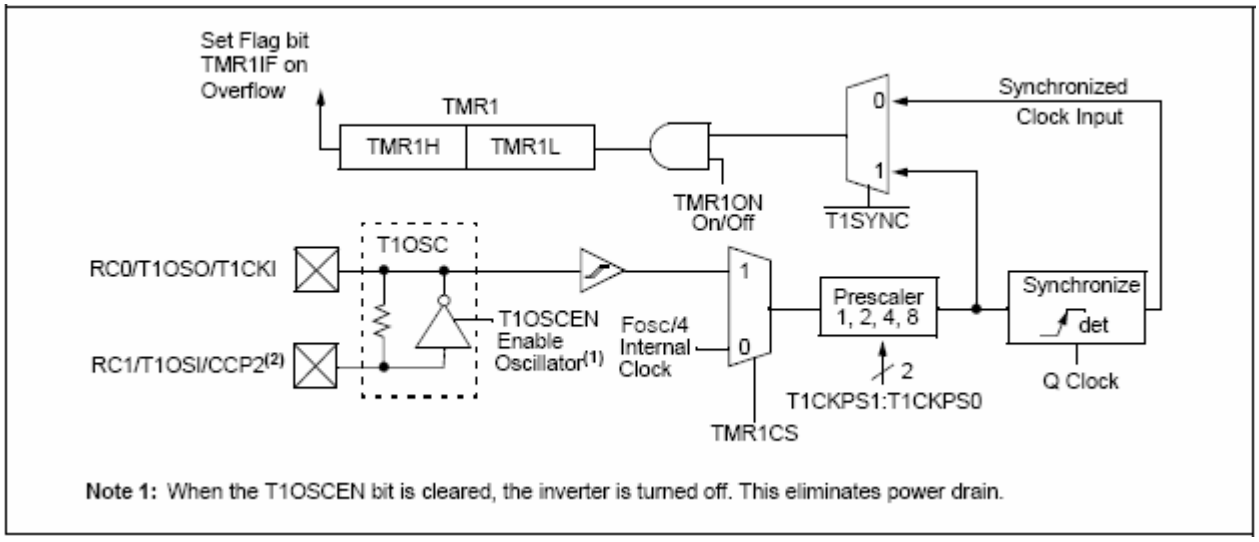


Fig. 3.3. Diagrama de bloques del Timer1

Operación en modo de temporizador

El modo de temporizador es seleccionado apagando el bit TMR1CS (T1CON<1>). En este modo, la entrada de reloj al temporizador es $F_{OSC}/4$. El bit de control de sincronización, $\overline{\text{T1SYNC}}$ (T1CON<2>), no tiene efecto, debido a que el reloj interno está siempre en sincronía.

Operación en modo de contador

El Timer1 puede operar tanto en modo asíncrono como síncrono, dependiendo del bit TMR1CS. Cuando el Timer1 esta siendo incrementado vía una fuente externa, el incremento ocurre en cada flanco de subida. Después de que el Timer1 es habilitado en modo de contador, el módulo debe recibir primero un flanco de bajada antes de que el conteo comience a incrementarse.

Operación del Timer1 en modo de contador sincronizado

El modo de contador es seleccionado encendiendo el bit TMR1CS. En este modo, el conteo se incrementa con la entrada de reloj en el pin RC1 cuando el bit T1OSCEN está encendido, o en el pin RC0 cuando el bit T1OSCEN está apagado.

Si el bit $\overline{\text{T1SYNC}}$ está apagado, entonces la entrada de reloj externa está sincronizada con la fase del reloj interno.

Operación del Timer1 en modo de contador asíncrono

Si el bit de control $\overline{\text{T1SYNC}}$ está encendido, la entrada de reloj interna no está sincronizada. El Timer1 continúa incrementando el conteo de forma asíncrona a la fase del reloj interno. El Timer1 continuará corriendo durante el modo SLEEP y puede generar una interrupción por sobreflujo, la cual puede despertar al procesador.

En el modo de contador asíncrono, el Timer1 no puede ser usado como base de tiempo para las operaciones de captura o comparación.

Tabla 7: Registros Asociados con el módulo del Timer1

Dirección	Nombre	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valor en el encendido	Valor en los demás RESETS
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu

x = Desconocido, u = Desconocido, - = Dirección no implementada leída como '0'. El Timer1 no utiliza las celdas sombreadas

3.2.6. Módulo USART

El módulo del receptor transmisor síncrono asíncrono universal (USART) es uno de los dos módulos I/O seriales. (USART es también conocido como Interfase de Comunicación Serial o SCI por sus siglas en inglés). El USART puede ser configurado como un sistema asíncrono ***full duplex*** que puede comunicarse con dispositivos periféricos, como las computadoras personales, o puede ser configurado como un sistema síncrono ***half Duplex*** que puede comunicarse con dispositivos periféricos, como circuitos integrados A/D o D/A, EEPROM seriales, etc.

El USART puede ser configurado de los siguientes modos:

- Asíncrono (full dúplex)
- Síncrono – Maestro (half dúplex)

- Síncrono – Esclavo (half dúplex)

El bit SPEN (RCSTA<7>) y los bits TRISC<7:6> tienen que estar encendidos de modo que los pines RC6/TX/CK y RC7/RX/DT estén configurados como el transmisor y el receptor síncrono asíncrono universal.

Tabla 8: Registros Asociados con el módulo USART

Dirección	Nombre	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valor en el encendido	Valor en los demás RESETS
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	Registro de transmisión del USART								0000 0000	0000 0000
1Ah	RCREG	Registro de recepción del USART								0000 0000	0000 0000
8Ch	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Registro generador de la tasa de transferencia de baudios								0000 0000	0000 0000

x = Desconocido, u = Desconocido, - = Dirección no implementada se lee como '0'. El USART no utiliza las celdas sombreadas

El módulo USART también tiene un multi-procesador de comunicaciones con capacidad de detección del noveno bit.

TXSTA: Registro de Control y Estado de la Transmisión (Dirección 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	**TX9**	**TXEN**	**SYNC**	**—**	**BRGH**	**TRMT**	**TX9D**
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit, bit de selección de la fuente del reloj.

Modo asíncrono:

No importa

Modo síncrono:

1 = Modo Maestro (el reloj es generado internamente desde BRG)

0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit, bit de habilitación de la transmisión del 9º bit

1 = Selecciona la transmisión del 9 bits

0 = Selecciona la transmisión de 8 bits

bit 5 **TXEN**: Transmit Enable bit, bit de habilitación de la transmisión.

1 = Habilita la transmisión

0 = Deshabilita la transmisión

Nota: Los bits SREN/CREN se imponen al bit TXEN en el modo SYNC.

bit 4 **SYNC**: USART Mode Select bit, bit de selección de modo del USART

1 = Modo síncrono

0 = Modo asíncrono

bit 3 **No implementado:** Se lee como '0'

bit 2 **BRGH**: High Baud Rate Select bit, bit de selección de alta velocidad de transmisión.

Modo asíncrono:

1 = Alta velocidad

0 = Baja velocidad

Modo síncrono:

No utilizado en este modo

bit 1 **TRMT**: Transmit Shift Register Status bit, bit de estado del registro de corrimiento de transmisión

1 = TSR empty

0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data, can be Parity bit, 9º bit de transmisión de datos, puede ser el bit de paridad

Etiquetas:			
R = Bit de lectura	W = Bit de escritura	U = bit no implementado, se lee como ‘0’	
- n = Value at POR	’1’= Bit encendido	’0’= Bit apagado	x = Bit desconocido

RCSTA: Registro de Control y Estado de la Recepción (Dirección 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	**RX9**	**SREN**	**CREN**	**ADDEN**	**FERR**	**OERR**	**RX9D**
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit, bit de habilitación del puerto serial

1 = Habilita el puerto serial (configura los pines RC7/RX/DT y RC6/TX/CK como puertos seriales)

0 = Deshabilita el puerto serial

bit 6 **RX9**: 9-bit Receive Enable bit, habilitación de la recepción del 9º bit

1 = Selecciona la recepción de 9 bits

0 = Selecciona la recepción de 8 bits

bit 5 **SREN**: Single Receive Enable bit, bit de habilitación de recepción sencilla

Modo asíncrono:

No importa

Modo síncrono - Maestro:

1 = Habilita la recepción sencilla

0 = Deshabilita la recepción sencilla

Este bit es apagado una vez que la recepción está completa.

Modo síncrono - Esclavo:

No importa

bit 4 **CREN**: Continuous Receive Enable bit, bit de habilitación de recepción continua

Modo asíncrono:

1 = Habilita la recepción continua

0 = Deshabilita la recepción continua

Modo síncrono:

1 = Habilita la recepción continua hasta que el bit de habilitación CREN es apagado (CREN se impone a SREN)

0 = Deshabilita la recepción continua

bit 3 **ADDEN:** Address Detect Enable bit, bit de habilitación de la detección de dirección

Modo asíncrono de 9 bits (RX9 = 1):

1 = Habilita la detección de dirección, habilita la interrupción y carga del buffer de recepción cuando RSR<8> esté encendido.

0 = Deshabilita la detección de dirección, todos los bytes son recibidos, y el noveno bit puede ser usado como bit de paridad

bit 2 **FERR**: Framing Error bit, bit de error de entramado

1 = Error de entramado (puede ser actualizado leyendo el registro RCREG y recibiendo el siguiente byte válido)

0 = No framing error

bit 1 **OERR**: Overrun Error bit, bit de error de sobreaso

1 = Error de sobrepaso (puede ser apagado poniendo en cero el bit CREB)

0 = No hay error de sobrepaso

bit 0 **RX9D:** 9th bit of Received Data (can be parity bit, but must be calculated by user firmware), 9º bit de recepción de datos (puede ser el bit de paridad, pero debe ser calculado por el software del usuario)

Generador de la velocidad de transmisión (BRG)

El BRG auxilia a ambos, los modo asíncrono y síncrono del USART. Es un generador de la velocidad de transmisión dedicado de 8 bits. El registro SPBRG controla el periodo de un reloj de carrera libre de 8 bits. En modo asíncrono el bit BRGH (TXSTA<2>) también controla la velocidad de transmisión. En modo síncrono, el bit BRGH es ignorado.

A continuación se muestran las fórmulas para el cálculo de la velocidad de transmisión para los diferentes modos de operación del USART.

Tabla 10: Cálculo de la velocidad de transmisión

SYNC	BRGH = 0 (Baja velocidad)	BRGH = 1 (Alta velocidad)
0	(Asíncrono) Vel. de Trans.= Fosc/(64(X+1))	(Asíncrono) Vel. de Trans.= Fosc/(16(X+1))
1	(Síncrono) Vel. de Trans.= Fosc/(4(X+1))	N/A

Dada la velocidad de transmisión y la frecuencia del reloj Fosc, el valor entero más cercano para el registro SPBRG puede ser calculado usando la fórmula de la tabla 10. Con esto el error en la velocidad de transmisión puede ser determinado. Esto puede ser ventajoso para utilizar la velocidad de transmisión alta (BRGH=1), inclusive para relojes de baja velocidad. Esto es debido a que la ecuación puede reducir el error en la velocidad de transmisión en algunos casos.

Tabla 11: Velocidades de Transmisión para el modo Asíncrono (BRGH = 0)

Vel. de Trans. (K)	F_{OSC}= 20 MHz			F_{OSC}= 16 MHz			F_{OSC}= 10 MHz		
	KBAUD	% ERROR	Valor SPBRG (decimal)	KBAUD	% ERROR	Valor SPBRG (decimal)	KBAUD	% ERROR	Valor SPBRG (decimal)
0.3	–	–	–	–	–	–	–	–	–
1.2	1.221	1.75	255	1.202	0.17	207	1.202	0.17	129
2.4	2.404	0.17	129	2.404	0.17	103	2.404	0.17	64
9.6	9.766	1.73	31	9.615	0.16	25	9.766	1.73	15
19.2	19.531	1.72	15	19.231	0.16	12	19.531	1.72	7
28.8	31.250	8.51	9	27.778	3.55	8	31.250	8.51	4
33.6	34.722	3.34	8	35.714	6.29	6	31.250	6.99	4
57.6	62.500	8.51	4	62.500	8.51	3	52.083	9.58	2
HIGH	1.221	–	255	0.977	–	255	0.610	–	255
LOW	312.500	–	0	250.000	–	0	156.250	–	0

Vel. de Trans. (K)	F_{OSC}= 4 MHz			F_{OSC} = 3.6864 MHz		
	KBAUD	% ERROR	Valor SPBRG (decimal)	KBAUD	% ERROR	Valor SPBRG (decimal)
0.3	0.300	0	207	0.3	0	191
1.2	1.202	0.17	51	1.2	0	47
2.4	2.404	0.17	25	2.4	0	23
9.6	8.929	6.99	6	9.6	0	5
19.2	20.833	8.51	2	19.2	0	2
28.8	31.250	8.51	1	28.8	0	1
33.6	–	–	–	–	–	–
57.6	62.500	8.51	0	57.6	0	0
HIGH	0.244	–	255	0.225	–	255
LOW	62.500	–	0	57.6	–	0

Tabla 12: Velocidad de Transmisión para el modo Asíncrono (BRGH = 1)

Vel. de Trans. (K)	F_{OSC}= 20 MHz			F_{OSC}= 16 MHz			F_{OSC}= 10 MHz		
	KBAUD	% ERROR	Valor SPBRG (decimal)	KBAUD	% ERROR	Valor SPBRG (decimal)	KBAUD	% ERROR	Valor SPBRG (decimal)
0.3	–	–	–	–	–	–	–	–	–
1.2	–	–	–	–	–	–	–	–	–
2.4	–	–	–	–	–	–	2.441	1.71	255
9.6	9.615	0.16	129	9.615	0.16	103	9.615	0.16	64
19.2	19.231	0.16	64	19.231	0.16	51	19.531	1.72	31
28.8	29.070	0.94	42	29.412	2.13	33	28.409	1.36	21
33.6	33.784	0.55	36	33.333	0.79	29	32.895	2.10	18
57.6	59.524	3.34	20	58.824	2.13	16	56.818	1.36	10
HIGH	4.883	–	255	3.906	–	255	2.441	–	255
LOW	1250.00	–	0	1000.00		0	625.000	–	0

Vel. de Trans. (K)	F_{OSC}= 4 MHz			F_{OSC} = 3.6864 MHz		
	KBAUD	% ERROR	Valor SPBRG (decimal)	KBAUD	% ERROR	Valor SPBRG (decimal)
0.3	–	–	–	–	–	–
1.2	1.202	0.17	207	1.2	0	191
2.4	2.404	0.17	103	2.4	0	95
9.6	9.615	0.16	25	9.6	0	23
19.2	19.231	0.16	12	19.2	0	11
28.8	27.798	3.55	8	28.8	0	7
33.6	35.714	6.29	6	32.9	2.04	6
57.6	62.500	8.51	3	57.6	0	3
HIGH	0.977	–	255	0.9	–	255
LOW	250.000	–	0	230.4	–	0

Al escribir un nuevo valor en el registro SPBRG causará que el reloj del BRG reciba un reset. Esto asegura que el BRG no espere por un sobreflujo del temporizador antes de producir la nueva velocidad de transmisión.

USART modo asíncrono

En este modo el USART utiliza un formato estándar de no retorno a cero (NRZ) (un bit de inicio, ocho o nueve bits de datos y un bit de paro. El formato más común de datos es el de 8 bits. El USART transmite y recibe el bit menos significativo primero. El transmisor y el receptor funcionan independientemente, pero usan el mismo formato de datos y la misma velocidad de transmisión. El BRG produce un reloj, ya sea x16 o x64, dependiendo del bit BRGH (TXSTA<2>). El hardware no puede soportar la paridad pero puede ser implementada por software (y almacenada como el noveno bit). El modo asíncrono se detiene durante el estado de SLEEP.

El modo asíncrono es seleccionado apagando el bit SYNC (TXSTA<4>).

El módulo asíncrono del USART consiste de los siguientes importantes elementos:

- El generador de la velocidad de transmisión
- El circuito de muestreo
- El transmisor asíncrono
- El receptor asíncrono

Transmisión asíncrona USART

El diagrama de bloques del transmisor USART es mostrado en la figura 3.4. El corazón del transmisor es el *TSR.* El registro de corrimiento obtiene su dato desde el buffer de lectura/escritura de transmisión, TXREG. El registro TXREG es cargado con el dato vía software. El registro TSR no es cargado hasta que el bit de paro ha sido transmitido de la carga previa. Tan pronto como el bit de paro es transmitido, el TSR es cargado con el nuevo dato del registro TXREG (si está disponible). Una vez que el registro TXREG transfiere el dato al registro TSR, el registro TXREG se vacía y el bit de bandera TXIF (PIR1<4>) se enciende. Esta interrupción puede ser habilitada / deshabilitada encendiendo / apagando el bit de habilitación TXIE (PIE1<4>). El bit de bandera TXIF no puede ser apagado por software. Este se apagará solo cuando un nuevo

dato es cargado en el registro TXREG. El registro TSR no se encuentra en la memoria de datos, por lo que no es accesible para el usuario.

La transmisión es habilitada encendiendo el bit de habilitación TXEN (TXSTA<5>). La transmisión en curso no ocurrirá hasta que el registro TXREG haya sido cargado con el dato y el BRG haya producido un pulso de reloj.

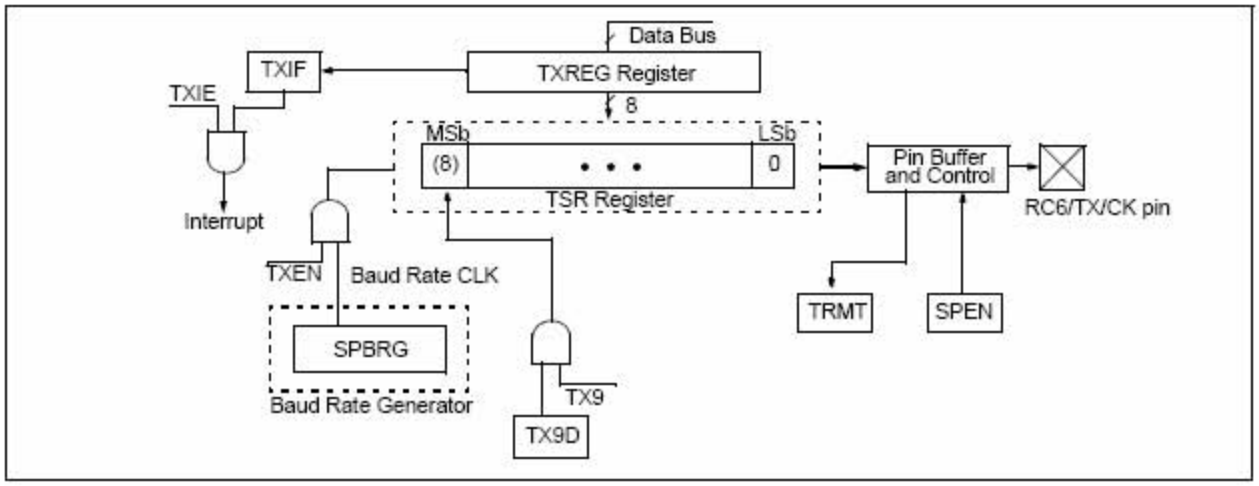


Fig. 3.4. Diagrama de bloques del transmisor USART

Para realizar una transmisión asíncrona, deben de seguirse los siguientes pasos:

1) Inicializar el registro SPBRG para la apropiada velocidad de transmisión. Si se desea una velocidad de transmisión alta, se debe encender el bit BRGH (TXSTA<2>).
2) Habilitar el puerto serial asíncrono apagando el bit SYNC (TXSTA<4>) y encendiendo el bit SPEN (RCSTA<7>).
3) Si se desean las interrupciones, entonces se enciende el bit TXIE (PIE1<4>).
4) Si la transmisión de 9 bits es deseada, entonces se enciende el bit TX9 (TXSTA<6>)
5) Habilitar la transmisión encendiendo el bit TXEN (TXSTA<5>), el cual también encenderá el bit TXIF (PIR1<4>)
6) Si la transmisión de 9 bits está seleccionada, el noveno bit deberá ser cargado en el bit TX9D (TXSTA<0>)
7) Se carga el dato en el registro TXREG (se empieza la transmisión)

8) Si se utilizan las interrupciones, hay que asegurarse que los bits GIE y PEIE (INTCON<7:6>) están encendidos.

Recepción asíncrona USART

En el receptor mostrado en la figura 3.5, se puede observar que el dato es recibido en el pin RC7/RX/DT y conducido al bloque de recuperación de datos (Data recovery block). El bloque de recuperación de datos es un registro de corrimiento de alta velocidad, que opera 16 veces más rápido que la velocidad de transmisión; mientras que, el registro de corrimiento serial principal opera la velocidad de la F_{OSC}.

Una vez que el modo asíncrono es seleccionado, la recepción es habilitada encendiendo el bit CREN (RCSTA<4>).

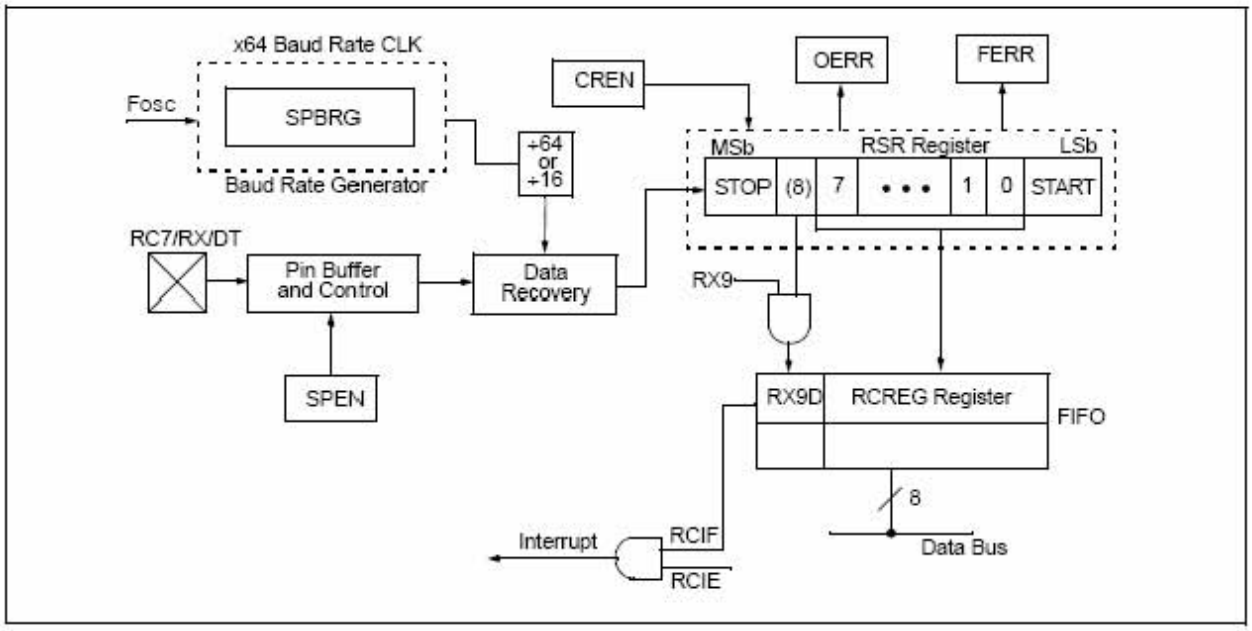


Fig. 3.5. Diagrama de bloques del receptor USART

El corazón del receptor es el ***RSR***. Después de verificar el bit de STOP, el dato recibido en el RSR es transferido al registro RCREG (si está vacío). Si el registro está lleno, el bit de bandera RCIF (PIR1<5>) se enciende, este bit es de sólo lectura y es apagado vía hardware cuando el registro RCREG ha sido leído y vaciado. La interrupción puede ser habilitada / deshabilitada

encendiendo / apagando el bit RCIE (PIE1<5>). El RCREG es un registro con un buffer doble (como si fuera un FIFO de dos niveles de profundidad) lo cual permite a dos bytes de datos ser recibidos y transferidos al FIFO del RCREG y a un tercer byte ser corrido hacia el registro SRS.

Para realizar una recepción asíncrona correcta se deben seguir los siguientes pasos:

1) Inicializar el registro SPBRG para una apropiada velocidad de transmisión. Si se desea una velocidad de transmisión alta, se debe encender el bit BRGH (TXSTA<2>).
2) Habilitar el puerto serial asíncrono apagando el bit SYNC (TXSTA<4>) y encendiendo el bit SPEN (RCSTA<7>)
3) Si se desean las interrupciones, entonces se enciende el bit RCIE (PIE1<5>).
4) Si se desea una recepción de 9 bits, entonces se enciende el bit RX9 (RCSTA<6>).
5) Habilitar la recepción encendiendo el bit CREN (RCSTA<4>).
6) El bit de bandera RCIF se encenderá cuando la recepción este completa y una interrupción será generada si el bit de habilitación RCIE (PIE1<5>) está encendido.
7) Se lee el registro RCSTA para obtener el noveno bit (sise habilitó) y determinar si algún error ha ocurrido durante la recepción.
8) Se obtiene el dato de 8 bits recibido leyendo el registro RCREG.
9) Si algún error ocurrió, limpiar el bit de error apagando el bit de habilitación CREN.
10) Si se están utilizando las interrupciones, asegurarse que los bits asegurarse que los bits GIE y PEIE (INTCON<7:6>) están encendidos.

USART síncrono – Modo maestro

En el modo maestro síncrono el dato es transmitido de manera half duplex. Cuando se está transmitiendo un dato, la recepción es inhibida y viceversa. Se entra en el modo síncrono encendiendo el bit SYNC (TXSTA<4>). En complemento, el bit de habilitación SPEN (RCSTA<7>) es encendido para configurar los pines RC6/TX/CK y RC7/RX/DT como líneas de CK (reloj) y DT (dato). Para entrar en el modo maestro se enciende el bit CSRC (TXSTA<7>).

Transmisión síncrona en modo maestro USART

La transmisión es habilitada encendiendo el bit de habilitación TXEN (TXSTA<5>). La transmisión en curso no ocurrirá hasta que el registro TXREG haya sido cargado con un dato. El primer bit del dato será corrido hacia fuera en el siguiente flanco de subida del reloj en la línea CK. El dato saliente es estable durante el flanco de bajada del reloj síncrono. La transmisión también puede ser comenzada cargando primero el registro TXREG y luego encendiendo el bit TXEN. Esto es ventajoso cuando se selecciona una velocidad de transmisión baja desde que BRG es mantenido en reset cuando los bits TXEN, CREN y SREN están apagados. Encendiendo el bit de habilitación TXEN se encenderá BRG, creando un cambio en el reloj inmediatamente. Normalmente, cuando la transmisión es comenzada por primera vez, el registro TSR está vacío, así que una transferencia al registro TXREG resultará en una transferencia inmediata a TSR, resultando un TXREG vacío. Las transferencias una detrás de otra son posibles.

Para seleccionar la transmisión de 9 bits, el bit TX9 (TXSTA<6>) deberá estar encendido y el noveno bit debe ser escrito en el bit TX9D (TXSTA<0>). El noveno bit debe ser escrito antes de escribir el dato de 8 bits en el registro TXREG. Esto es porque un dato escrito en TXREG puede resultar en una transferencia inmediata del dato al registro TSR (si TSR está vacío).

Recepción síncrona en modo maestro USART

Una vez que el modo síncrono es seleccionado, la recepción es habilitada encendiendo cualquiera de los bits SREN o CREN (RCSTA<5:4>). El pin RC7/RX/DT es muestreado, para obtener el dato, en los flancos de bajada del reloj. Si el bit de habilitación SREN está encendido, entonces sólo una palabra es recibida. Si el bit de habilitación CREN está encendido, la recepción es continua hasta que sea apagado. Si ambos bits están encendidos, CREN toma prioridad. Después de recibir el último bit, el dato recibido en el RSR es transferido al registro RCREG (si está vacío). Cuando la transferencia está completa, el bit de bandera de interrupción RCIF (PIR1<5>) se enciende.

USART síncrono – Modo esclavo

El modo esclavo síncrono difiere del modo maestro en el hecho de que el reloj es suministrado externamente en el pin RC6/TX/CK (en lugar de ser suministrado internamente como en el modo maestro). Esto permite al dispositivo transferir o recibir datos mientras está en el modo SLEEP. El modo esclavo es accesado apagando el bit CSRC (TXSTA<7>).

Transmisión síncrona en modo esclavo USART

La operación de los modos síncronos maestro y esclavo es idéntica, excepto en el caso del modo SLEEP.

Si dos palabras son escritas en el registro TXREG y luego la instrucción del modo SLEEP es ejecutada, ocurrirá lo siguiente:

a) La primera palabra será inmediatamente transferida al registro TSR y transferida.
b) La segunda palabra permanecerá en el registro TXREG.
c) El bit de bandera TXIF no será encendido.
d) Cuando la primera palabra ha sido movida fuera del TSR, el registro TXREG transferirá la segunda palabra a TSR y el bit de bandera TXIF ahora sí será encendido.
e) Si el bit de habilitación TXIE está encendido, la interrupción despertará al chip del modo SLEEP y si la interrupción global está habilitada, el programa se redirigirá al vector de interrupción (0004h).

Recepción síncrona en modo esclavo USART

La operación de los modos síncronos maestro y esclavo es idéntica, excepto en el caso del modo SLEEP. El bit SREN es un “no importa” en el modo esclavo.

Si la recepción es habilitada encendiendo el bit CREN antes de la instrucción SLEEP, entonces una palabra puede ser recibida durante este modo. Al completarse la recepción de la palabra, el registro RSR transferirá el dato al registro RCREG y si el bit de habilitación RCIE está encendido, la interrupción generada despertará al chip del modo SLEEP. Si la interrupción global esté habilitada, el programa se redirigirá al vector de interrupción (0004h).

3.3.Los Programas MPLAB IDE v6.4 e IC–Prog 1.05C

Todas las características que se mencionarán a continuación son sólo las características básicas del programa MPLAB IDE, que es un software gratuito ofrecido por la empresa Microchip en su página de Internet[1], el cual tiene más herramientas y opciones como por ejemplo la posibilidad de descargar los programas al μC una vez terminados mediante un programador producido también por la empresa Microchip conocido como PICSTART. Sin embargo la adquisición de este equipo implica un gasto mayor.

Es por esta razón que se buscó una alternativa al realizar este proyecto, pues aunque MPLAB permite la edición, ensamblado, compilación y ligado de proyectos, solo hace falta un método de descarga al chip. Este paso se logró mediante otro software de tipo gratuito de nombre IC–Prog[1] en combinación con un circuito de programación muy sencillo en su construcción y de un costo relativamente nulo en comparación con los programadores comerciales.

3.3.1. El programa MPLAB IDE v6.4

El programa MPLAB IDE v6.4 es un ambiente de desarrollo integrado basado en el sistema operativo Windows para incorporar la tecnología de las familias de microcontroladores PIC y los microcontroladores de señal digital dsPIC. En el programa MPLAB IDE es posible:

- Crear códigos fuente utilizando el editor de ensamblaje.
- Ensamblar, compilar y ligar códigos fuente usando varias herramientas de lenguaje. Un ensamblador, ligador y librerías vienen incluidas en MPLAB.
- Capacidad de utilizar archivos de C como archivos fuente.
- Realizar pruebas de tiempo.
- Ver variables de interés en las ventanas “Watch”.
- Encontrar respuestas rápidas a preguntas a cerca de MPLAB IDE con la ayuda en línea.

[1] Ver Apéndice E

Esta es una lista de las extensiones que presentan los distintos tipos de archivos relacionados con MPLAB IDE.

Tabla 13: Extensiones utilizadas por los archivos del programa MPLab

Extensión	Definición
asm	Archivo fuente en lenguaje ensamblador
c	Archivo fuente C
chm	Archivo de ayuda de compilación
cod	Contiene información simbólica y el código de objeto
cof	Contiene información simbólica y el código de objeto
err	Archivo de error generado por el ensamblador/compilador
evt	Archivo de evento – MPLAB ICE 2000
exe	Archivo ejecutable
fsti	Archivo de estímulos – MPLAB SIM
gld	Archivo script del ligador – MPLAB LINK30
h	Archivo C incluido
hex	Código máquina en formato hex para microcontrolador PIC
inc	Archivo en lenguaje ensamblador incluido – Ensamblador MPASM
lkr	Archivo script del ligador – MPLINK linker
lst	Archivo de listado absoluto generado por el ensamblador/compilador
mcp	Contiene la información relacionada al proyecto
mcw	Contiene la información relacionada al espacio de trabajo
o	Archivo de objeto – herramienta de lenguaje dsPIC
obj	Archivo de objeto – Herramienta de lenguaje Microchip
psti	Estímulos de pin – MPLAB SIM
rsti	Estímulos de registros – MPLAB SIM
ssti	Estímulos síncronos – MPLAB SIM
s	Archivo fuente del lenguaje ensamblador – MPLAB ASM30
trc	Archivos de gráficas guardadas
trg	Archivo de disparo – MPLAB ICE 2000

En la figura 3.6, se muestra una imagen de la estructura de ventanas básicas que puede desplegar el programa MPLAB para auxiliar al usuario durante la creación y simulación de programas de control, que posteriormente serán descargados al chip seleccionado, utilizando las herramientas de edición, compilación y simulación que ofrece este software.

Es importante mencionar que cada ventana puede ser abierta desde el menú “View” pues no aparecen al crear un nuevo proyecto.

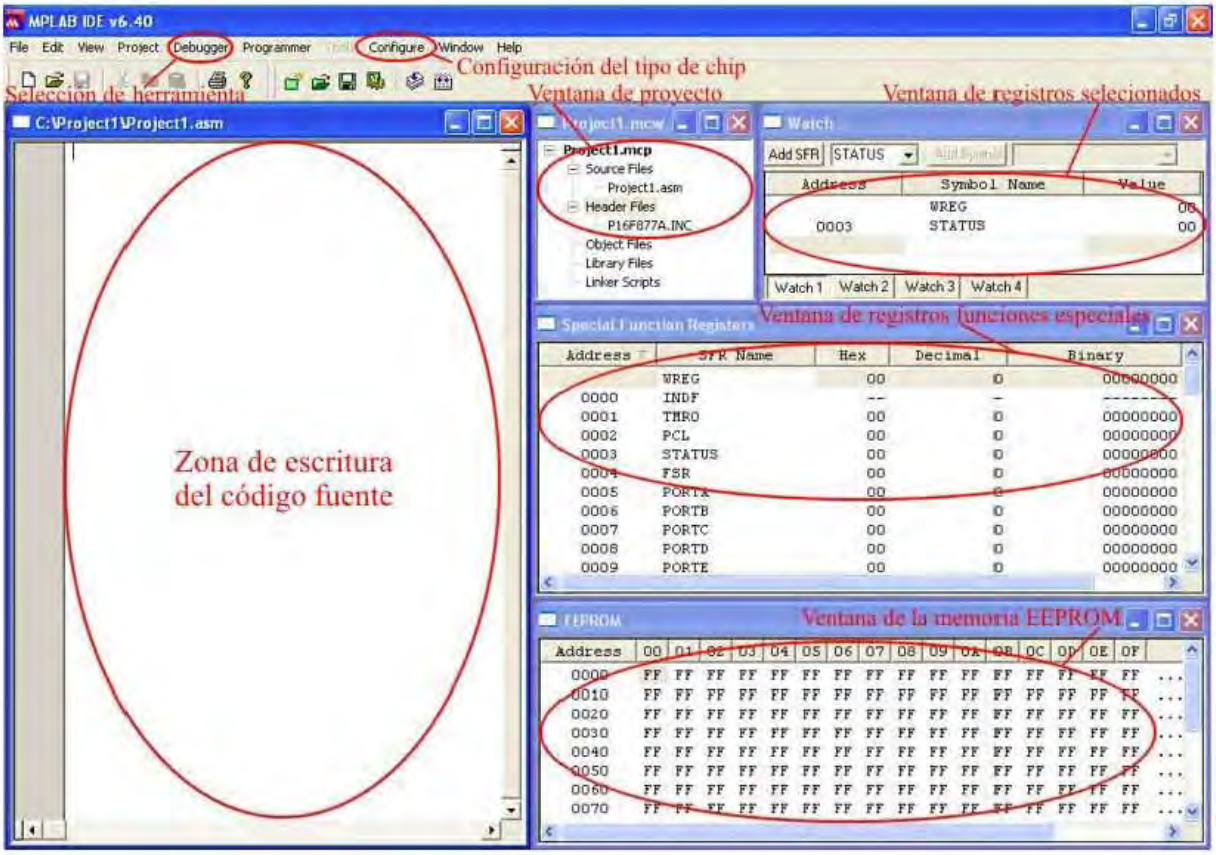


Fig. 3.6. Ventanas que pueden ser mostradas por el programa MPLab

Configuración del tipo de chip (Configure)

Esta es la primera opción en que debe ser revisada al crear un nuevo proyecto pues en este menú se encuentra la opción “Select Device”, “Selección de Dispositivo”. Es aquí donde se selecciona el modelo del microcontrolador de acuerdo a su número de serie, en este caso el dispositivo a seleccionar es “PIC16F877A”. Existe una amplia lista de dispositivos, la cual es ampliada y actualizada cada que se crea una nueva versión del programa.

Si el chip no está correctamente seleccionado habrá conflictos en el momentote escribir el código fuente o al intentar descargar el programa al chip será denegada la acción al no existir coincidencia de los elementos.

Selección de herramienta de trabajo (Menú Debugger)

En éste menú se encuentra la opción “Select Tool”, “Selección de Herramienta”, con la cual es posible seleccionar entre las distintas herramientas del programa entre las que destacan las opciones “None”, “Ninguna”, en donde en realidad se encuentra en el modo de edición de código fuente, y “MPLAB SIM” que es la herramienta de simulación incluida. Al seleccionas la herramienta “None” la apariencia del las ventanas y menús será como se observa en la figura aunque las ventanas “Watch”, “EEPROM” y la de “Registros Especiales” no tendrán relevancia pues estas son utilizadas solo por el simulador. Si es seleccionada la herramienta “MPLAB SIM” aparecerá una barra de herramientas extra en donde se encuentran las opciones para correr una simulación además de que en el mismo menú “Debugger” aparecerá la opción de incluir otra ventana desde donde se pueden incluir estímulos externos que lleguen a los pines del entrada del chip, además de que las ventanas antes mencionadas entrarán en función mostrando cambios en los registros y memoria.

Ventana de proyecto

Al crear un nuevo proyecto es aquí donde se despliegan los archivos que contienen la información del proyecto (.mcp), el archivo del código fuente (.asm) el cual es necesario adicionarlo al proyecto de forma manual y el archivo que contiene el lenguaje ensamblador correspondiente al chip que se está utilizando (.inc), que también debe ser adicionado al proyecto.

Ventana de edición

Esta ventana es en la que en realidad se trabaja. Es aquí donde se escriben las instrucciones[1], rutinas y subrutinas que deberá seguir en programa, o sea el código fuente. En el encabezado de la ventana se muestra el nombre del archivo que se está creando (.asm) y la ubicación en donde está almacenado el proyecto en la memoria de la computadora. Hay que tener cuidado de que la ruta donde se almacena la información del proyecto no sea demasiado larga pues el programa tiene un número limitado de caracteres en este respecto.

[1] Ver Apéndice B

Ventana de registros de funciones especiales (SFR)

Como ya se mencionó esta ventana puede ser abierta desde el menú "View", "Ver" y es utilizada por el simulador. En esta ventana se encuentran enlistados todos los registros de funciones especiales que conforman la memoria de datos.

Durante la simulación es posible observar en esta ventana como se modifica el contenido de los registros de acuerdo a las instrucciones del código fuente.

La información desplegada en esta ventana para cada registro es su dirección en la memoria de datos, el nombre del registro, su contenido en formato hexadecimal, decimal y bit por bit, o sea binario y si el contenido del registro corresponde a algún carácter alfanumérico.

Ventana de registros seleccionados (Watch)

Esta ventana es muy similar a la de registros de funciones especiales con la diferencia de que contiene menos columnas de información. La diferencia principal entre las dos ventanas es que mientras la "ventana SFR" muestra todos los registros que existen en el chip correspondiente, la "ventana Watch" muestra sólo los registros que interesan de forma particular, esto es que pueden ser seleccionados de una lista y adicionados a la ventana.

Ventana de la memoria EEPROM

Aquí se muestra en contenido de cada localidad de la memoria EEPROM, si el chip tiene esta característica. Además de lo anterior también puede observarse si lo contenido en cada localidad corresponde a algún código ASCII[1] .

Botones de Proyecto

Algunas de las funciones del menú "Project" están representadas por los botones de proyecto para facilitar el acceso a estas, sin embargo existen funciones adicionales que conviene revisar dentro del menú. Por mencionar algunas están "Project Wizard" que permite crear y configurar un nuevo proyecto de MPLAB paso a paso, "Close" que cierra el proyecto actual, "Save Project

[1] Ver Apéndice C

As…" para guardar un nuevo proyecto o para hacer una copia del proyecto actual, "Add Files to Project…" para adicionar nuevos archivos al proyecto, etc.

New Project (Proyecto Nuevo). Crea un nuevo proyecto en el espacio de trabajo. Abre la ventana de diálogo "New Project" en donde se pide un nombre para el proyecto y una ruta donde salvarlo.

Open Project (Abrir Proyecto). Adiciona un proyecto existente al espacio de trabajo y es puesto como activo. Abre la ventana de diálogo "Open Project" en donde se solicita el nombre y ruta del proyecto a abrir.

Save Workspace (Guardar espacio de trabajo). Guarda el espacio de trabajo tal y como se encuentre en ese momento, es decir, con todas las ventanas que se encuentren abiertas y con las herramientas que estén activas.

Build All (Construye Todo). Construye el proyecto compilando y ensamblando los archivos del proyecto.

Make (Construir). Construye el proyecto compilando y ensamblando solo los archivos que han sido cambiados desde la última vez que se construyó el proyecto.

Botones de Simulación

Una vez compilado y ensamblado un proyecto existe la opción de realizar una simulación antes de descargar el programa al chip. Al seleccionar la herramienta MPLAB SIM del menú "Debugger" aparecerá la barra de herramientas de simulación que muestra los siguientes botones.

Run (Correr). Ejecuta el código del programa hasta que un punto de ruptura (breakpoint) es encontrado o hasta que la opción "Halt" es seleccionada. Comienza la ejecución del contador de programa (que es mostrado en la barra de status). La ubicación actual del contador de programa es representada como un indicador en la ventana de la memoria de programa. Mientras el programa está corriendo, muchas otras funciones están deshabilitadas.

Halt (Alto). Detiene la ejecución del código de programa. Cuando se da clic en "Halt", la información del status es actualizada.

Animate (Animar). La función concreta que realiza Animar es la de ejecutar el programa paso a paso al tiempo que los valores de los registros son actualizados como si estuviera corriendo, esto permite observar los cambios de los registro en la ventana de Registros de Funciones Especiales o en la ventana Watch.

Step Into (Paso hacia dentro). Da un paso a través del código de programa ejecutando una sola instrucción para luego detenerse. Después de esto, todas las ventanas son actualizadas.

Step Over (Paso Sobre). Ejecuta la instrucción de la ubicación actual del contador de programa. En una instrucción "CALL", "Step Over" ejecuta la subrutina llamada y se detiene en la dirección del siguiente "CALL".

Reset. Emite un MCLR al reset del contador de programa en el vector reset. Si el programa está corriendo, este continuará corriendo desde la dirección del vector reset.

3.3.2. El programa IC-Prog 1.05C

El IC-Prog es un programa que funciona bajo el sistema operativo Windows, que puede controlar un programador de microcontroladores PIC.

Para que el programa funcione se deberá conectar a la computadora un programador, y configurar correctamente tanto a éste como al programa. Hay que notar que, debido a la variedad de programadores y sus diferencias el programa puede no funcionar con ciertas combinaciones de computadoras y equipos programadores.

El IC-Prog requiere Windows 95, 98, ME, NT, 2000 o XP y un coprocesador interno o externo para funcionar. Todos los procesadores compatibles y superiores a un 386 con 8Mb de memoria RAM deberían funcionar correctamente.

El IC-Prog es un programa registrado aunque es de libre distribución y ha sido designado como una aplicación de programación universal para todos los programadores.

Los siguientes programadores se encuentran soportados por el IC-Prog:

- Programador IDM (Ludipipo)
- Programador Conquest
- Programador TAFE
- Programador “Clásico” TAIT
- Programador TAIT Paralelo
- Programador Fun-card
- Programador SCHAER
- Programador ProPic II
- Programador STK200
- Programador AN589
- Programador WILLEPRO
- Programador Fluffy
- Programador DL2TM

El área principal del IC-Prog muestra la información necesaria para programar el dispositivo seleccionado. Todos los dispositivos poseen al menos un área de memoria de Código donde puede almacenarse la información.

Otros dispositivos, como son la mayoría de los microcontroladores, poseen áreas adicionales de almacenamiento como el área de Datos. Normalmente el área de Código contiene el código a ser ejecutado por el microcontrolador mientras que el área de Datos contiene algunos datos fijos, como por ejemplo tablas de cálculo, etc.

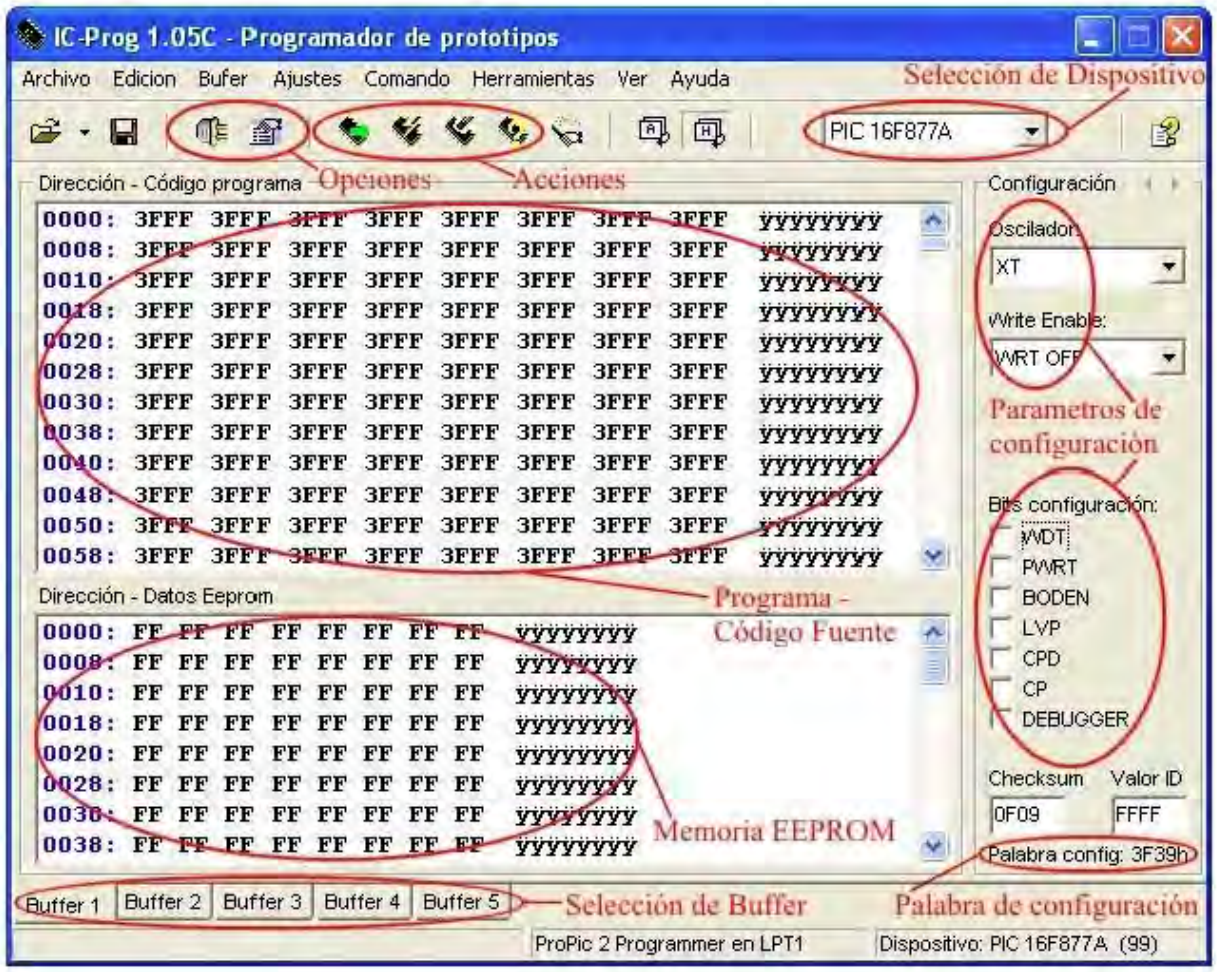


Fig. 3.7. Características del programa IC-Prog

La mayoría de los microcontroladores (como los PIC) poseen también un área de Configuración. La información de configuración precisamente servirá para configurar el microcontrolador con ciertos parámetros iniciales de arranque.

Esta información de configuración es diferente y única para cada tipo de microcontrolador, pudiendo encontrar en las hojas de datos del microcontrolador la información específica correspondiente.

Las áreas de Código y de Datos presentarán la información en valores hexadecimales y en caracteres, el sector izquierdo de las áreas de Código y de Datos contendrá la dirección en la que se almacena la información. El sector central contendrá la información en valores hexadecimales y el sector derecho contendrá la misma información en formato de caracteres.

Área de Código Fuente (Programa)

Cada fila en el área del código fuente mostrará 8 palabras, por lo que de una fila a la otra la dirección se incrementará en 8. Una palabra posee normalmente una longitud de 16 bits por lo que el IC-Prog mostrará un valor hexadecimal entre 0000 y FFFF. Algunos dispositivos sólo poseen longitudes de palabras de 14 bits, 12 bits u 8 bits, por lo que el máximo valor hexadecimal será 3FFF, 0FFF 00FF respectivamente, aunque el programa siempre mostrará el valor hexadecimal utilizando 4 dígitos. El valor en formato de carácter sólo utiliza los 8 bits más bajos de la palabra de 16 bits, ya que el rango estándar de los caracteres sólo va de 0 a 255.

Área de Datos (Memoria EEPROM)

Cada fila en el área de Datos también mostrará 8 palabras, pero en este caso las palabras son siempre de 8 bits por definición Siempre se mostrarán las palabras con 2 dígitos hexadecimales con un valor entre 00 y FF.

Parámetros de Configuración

El área de Configuración del dispositivo (si es que posee una) se mostrará con elementos fácilmente comprensibles, tales como listas de opciones u opciones seleccionables. El usuario puede seleccionar fácilmente la configuración deseada, y el IC-Prog calculará la correspondiente palabra de configuración. Esta palabra de configuración también se mostrará en la parte inferior del área de Configuración.

Estos elementos de configuración dependen del dispositivo, por lo que el área de Configuración será distinta de acuerdo al dispositivo seleccionado. Consulte las hojas de datos del dispositivo para encontrar la información de configuración específica.

Selección de Buffer

Esta es una herramienta que permite editar, copiar y comparar códigos mostrados en los distintos buffers. Su mayor utilidad radica en que es posible realizar comparaciones de manera rápida entre el programa original y el que fue descargado al μC, esto se realiza abriendo el programa en cualquier buffer y leyendo el contenido del μC en otro, a continuación se utiliza la opción "Comparar" de menú "Buffer". Si los códigos son iguales será un indicativo de que la programación del μC fue correcta, de no serlo el programa señalará cada uno de las diferencias existentes entre ambos.

Botones de Acciones

Estos son los controles básicos del programa IC-Prog, como puede observarse son pocos y de fácil manejo.

Leer Todo. Este comando leerá los datos grabados en las memorias Flash y EEPROM de del chip.

Programar Todo. Programará el chip con el contenido del buffer y también programará el chip de a cuerdo a la configuración mostrada en la pantalla.

Borrar Todo. Este comando borrará todo el contenido del chip, incluyendo la memoria EEPROM.

Verificar. Este comando verificará que los contenidos del chip correspondan con los contenidos del buffer del IC-Prog.

Asistente Smartcard. Programará una Smartcard con un BootLoader (programa de arranque) que habilitará la programación directa de la EEPROM del PIC.

Botones de Opciones

Permiten el acceso a las opciones de configuración avanzadas donde, entre otras opciones, se encuentran las selección de programador, tipo de interfase, idioma, compatibilidad, etc., de las cuales sólo se mencionarán las de mayor importancia.

Configurar Hardware. Este comando permite configurar al IC-Prog para el hardware que se va a utilizar.

En lo que respecta a la ventana de configuración de Hardware, en esta pueden definirse el tipo de hardware, interfaz, puerto y parámetros de comunicación como se observa en la figura 3.8.

Fig. 3.8. Ventana de configuración de hardware del programa IC-Prog

Opciones. Mediante este comando se pueden configurar las siguientes opciones:

- Arrastrar y Soltar
- Puerto para Smartcard y tipo de dispositivo (16F84 o 16C84)
- Idioma a utilizar
- Extensión del Shell
- Miscelánea
- Confirmaciones
- Avisos
- I^2C
- Programación
- Atajos

Dentro de estas opciones una de particular importancia es la sección de "Miscelánea" pues al utilizar el programa IC-Prog en el sistema operativo Windows XP, la opción de compatibilidad no está activada por lo que el programa no funcionará de manera correcta. Una vez activada la compatibilidad es necesario cerrar por completo el programa y abrirlo nuevamente para poder

utilizarlo de manera correcta. Otra opción que no está activada inicialmente es la de idioma. Al abrir el programa por primera vez este estará en el idioma inglés, pero una de las ventajas que tiene es que es multi-idioma por lo que puede configurarse al español en la pestaña de opciones "Language" (Idioma).

Dentro de lo que respecta a la opción "Smartcard" el IC-Prog soporta programadores tipo "Phoenix" compatibles. Sin embargo, dado que el programador "Phoenix" no es realmente un programador, no se encuentra listado en el menú Tipo Hardware. En esta página se pueden configurar los ajustes de su Smartcard.

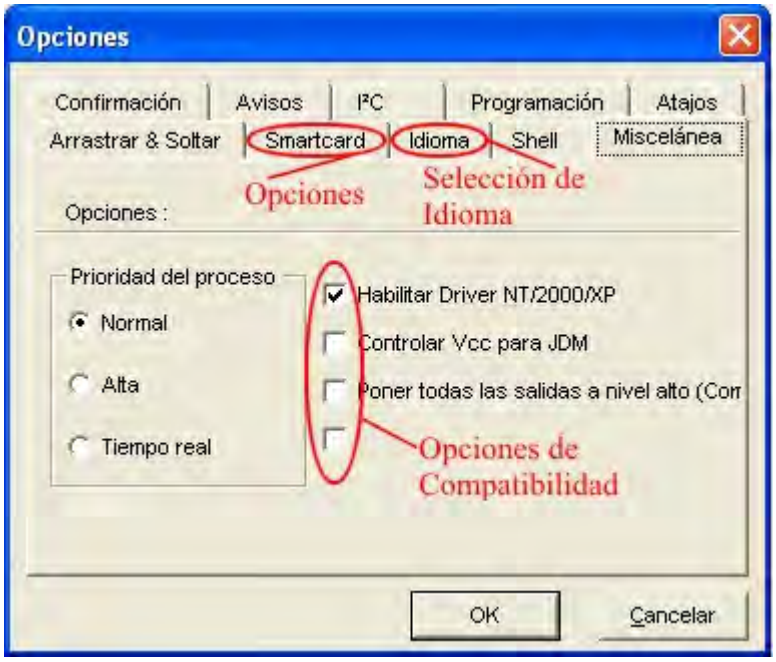


Fig. 3.9. Ventana de opciones del programa IC-Prog

3.3.3. El programador de dispositivos PIC

El programador utilizado en el proyecto es realmente sencillo[1], como puede apreciarse en el diagrama. Es de una gran confiabilidad en su operación y de reparación sencilla debido a la escasa cantidad de componentes que lo forman. Además este hardware es compatible con el software de programación IC-Prog 1.05C si este se configura para trabajar con el programador

[1] Ver en el Apéndice D la dirección de internet

“ProPic 2 Programmer” con una interfaz “Direct I/O” y una comunicación invirtiendo el MCLR. A continuación se da una explicación relativamente detallada de su funcionamiento.

El LED marcado como "Enc." permite observar que el sistema se encuentra alimentado mientras que el LED marcado como "Prog." se enciende indicando que es seguro insertar o quitar un chip (PIC o memoria) y se apaga por instantes breves cuando una lectura o programación de un PIC está en curso. Mientras este último LED este apagado no se debe quitar o insertar ningún chip del zócalo.

El funcionamiento del circuito es muy simple: los pines del puerto paralelo 2, 3, 5,6 y 10 permiten interconectar el circuito con la PC. El pin 2 es el encargado de traer los datos (desde la PC hacia el chip). El pin 3 es el envío de los pulsos de reloj (desde la PC hacia el integrado). En tanto el pin 10 permite a la PC leer los datos desde el programador. Los pines 5 y 6, por último, son los encargados de controlar la tensión de programación (Vpp) necesaria para cuando queremos leer o escribir en un PIC. El interruptor que se encuentra conectado a estos dos pines permite seleccionar entre los distintos zócalos de programación existentes, cuando se cierra hacia el pin 5 los zócalos que trabajan son los de 8 y 18 pines y cuando se encuentra cerrado hacia el pin 6 están activos los zócalos de 28 y 40 pines.

Los microcontroladores PIC se programan utilizando el mismo protocolo que las memorias EEPROM seriales, por consiguiente el programador sirve tanto para PIC's como para memorias. La tensión de programación VPP es necesaria para indicar al PIC que se desea leerlo o programarlo. Si en este pin (que es compartido con la entrada de RESET del chip) es puesto en bajo (0V) el PIC sufre un reset, si se pone en alto (5V) el PIC trabaja normalmente mientras que si es puesto el pin a 12V el PIC se inicializa en modo programación, quedando dos de los pines de I/O destinados a datos (SDA) y reloj (SCL).

El integrado 74LS04 está formado internamente por seis buffers inversores. Estos permiten, por un lado, obtener niveles TTL a su salida y por el otro no cargar de forma excesiva el puerto. Algunos programadores, como el NOPPP utilizan diodos y resistencias para conectar el PIC directamente al puerto paralelo. Esto funciona en muchas computadoras de escritorio con fuentes poderosas pero en la mayoría de las portátiles que no disponen de tanta corriente el

funcionamiento es errático o no funciona. Gracias a la utilización de este buffer podremos utilizar el circuito en cualquier puerto paralelo ya sea de una computadora de escritorio o en un portátil. Se colocan las compuertas en serie para obtener a la salida el mismo nivel de entrada, sin invertir. Las resistencias de 1K dan seguridad al sistema para evitar que circule corriente excesiva.

El control de la tensión de programación lo efectúa el transistor NPN. Estando el pin 5 del puerto paralelo a tierra (en 0V) tendremos al transistor abierto por lo que la corriente proveniente de +V (12V) pasará por el diodo LED el cual no encenderá y se portará como un diodo común polarizado en directa, pasará por la resistencia limitadora de corriente del LED la cual no ofrecerá mucha resistencia y será inyectada al PIC en su terminal MCLR/VPP. Poniendo en 1 (5V) el bit que controla el pin 5 del puerto paralelo, en cambio, el transistor se cierra hacia el PIC haciendo, además, encender el LED al quedar a tierra el otro extremo de la resistencia limitadora de corriente.

El circuito requiere como única alimentación 12V de CD con una corriente de 200mA. Puede usarse cualquier fuente universal siempre que se respete la polaridad. De tener una fuente de mayor tensión (13.5v como mucho) no hay problema, se la puede utilizar sin inconvenientes.
No se necesita que la fuente sea regulada. Si se tiene una fuente de 12V con una corriente mayor a 1A o incluso se puede utilizar también sin inconvenientes.

Para conectarlo a la PC se puede utilizar un conector hembra DB25 directo hacia el puerto paralelo de la impresora (macho).

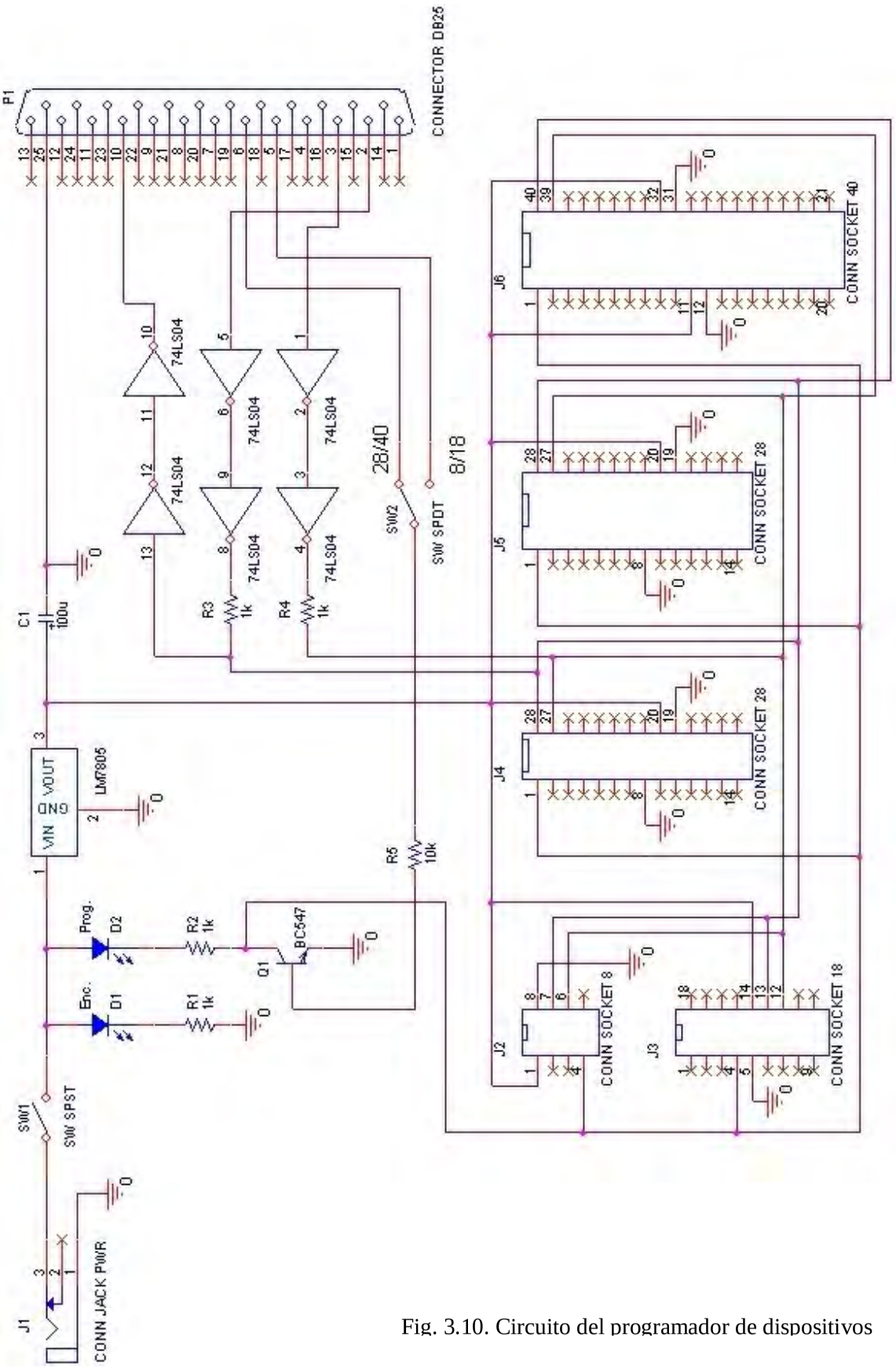


Fig. 3.10. Circuito del programador de dispositivos

3.4.Diseño del Sistema de Control

El objetivo principal del sistema es realizar una detección de la posición del motor permitiendo a un operador, a través de la PC, modificar la posición de éste de acuerdo al las necesidades que se presenten. Es posible indicar el sentido de giro del motor, ya sea que se necesite a la derecha o a la izquierda, y la cantidad de grados a girar, es decir, es posible hacer la indicación al motor de que, por ejemplo, realice un giro hacia la derecha de 75 grados y que después regrese 30 grados hacia la izquierda para tener una posición final de 45 grado.

En el diagrama de bloques de la figura 3.11, se muestra a grandes rasgos el funcionamiento del sistema. Desde la PC se envían instrucciones al μC a través de una etapa de comunicación serial, el μC interpreta las instrucciones y envía las señales de control que operan la etapa de acoplo entre el motor y el circuito digital para activar el motor y desplazarlo hasta la posición indicada, durante este proceso un sensor capta señales enviadas desde el motor y envía la información al μC para que este determine el momento en que se ha llegado a la posición deseada para, llegado ese momento, enviar la señal de paro a la sección de activación del motor.

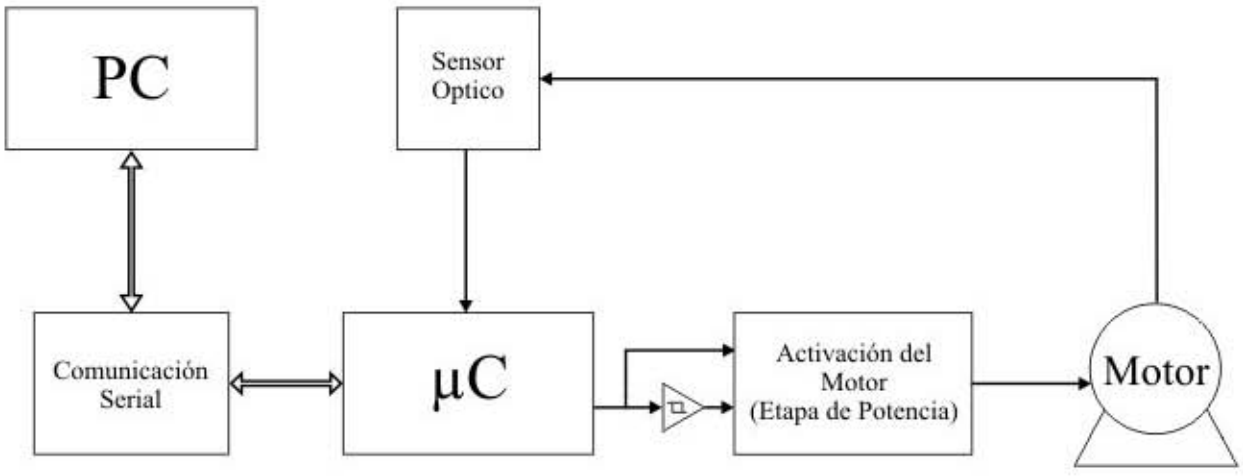
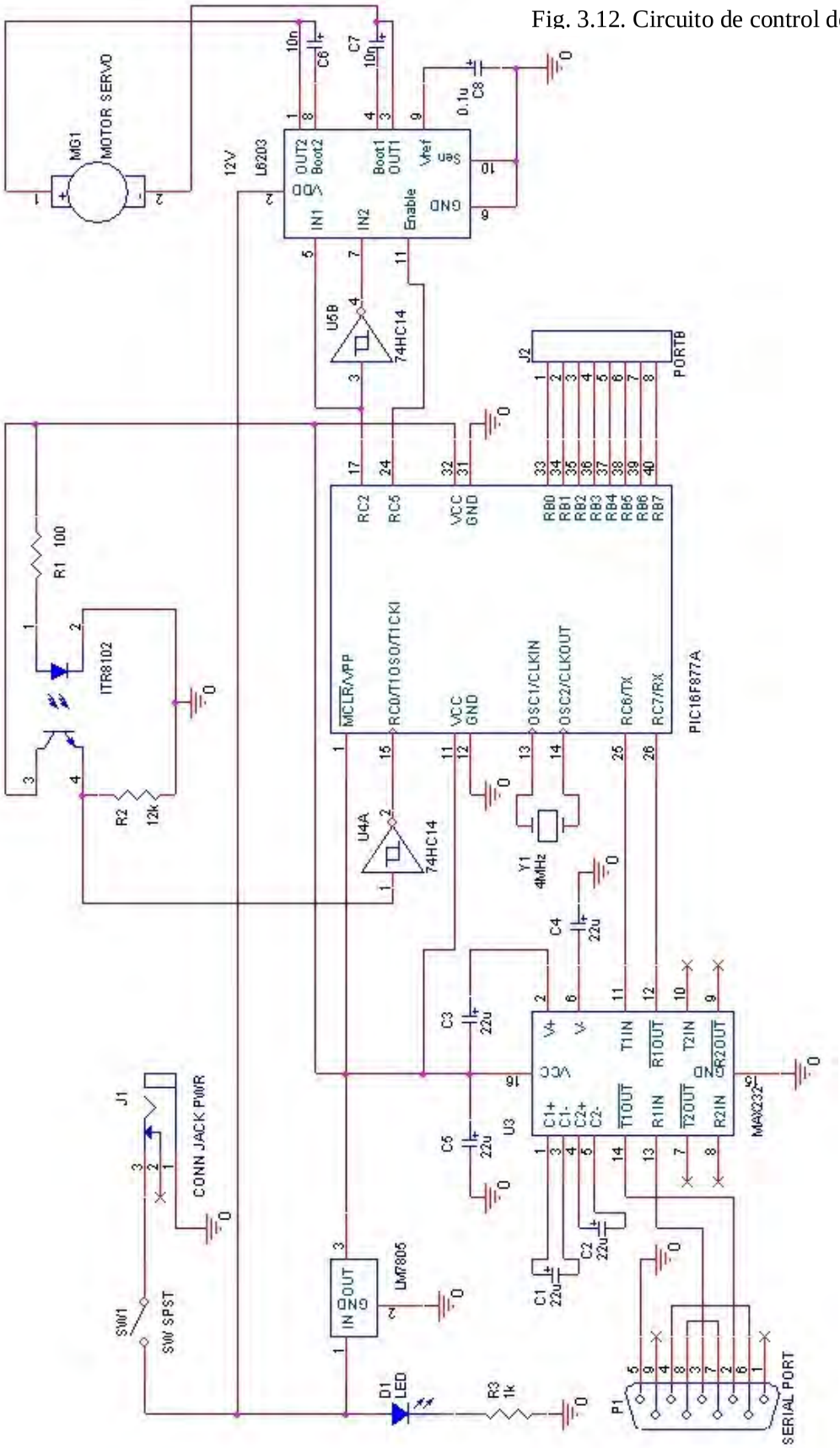


Fig. 3.11. Diagrama de bloques del sistema de control

A continuación se muestra el circuito de control del motor en todo su conjunto para luego dar una descripción de las distintas secciones que lo conforman de modo que quede establecido el funcionamiento de cada sección por separado.

Fig. 3.12. Circuito de control del motor



3.4.1. Sección de alimentación

En primer lugar, es necesario alimentar el circuito. Hay que destacar que el circuito que controla al motor necesita de dos niveles de voltaje distintos, uno para alimentar la parte digital del circuito y otro que es suministrado al circuito que aísla y activa al motor.

Esto se realiza a través de un eliminador de voltaje de 12V que es conectado por medio de un conector JACK al circuito, tiene un interruptor de encendido con un LED y su resistencia de protección. La alimentación se divide en dos ramas, una que es la que lleva el voltaje original de 12V para alimentar el motor y otra que va a un regulador de voltaje LM340 con salida de 5V para la alimentación del circuito digital.

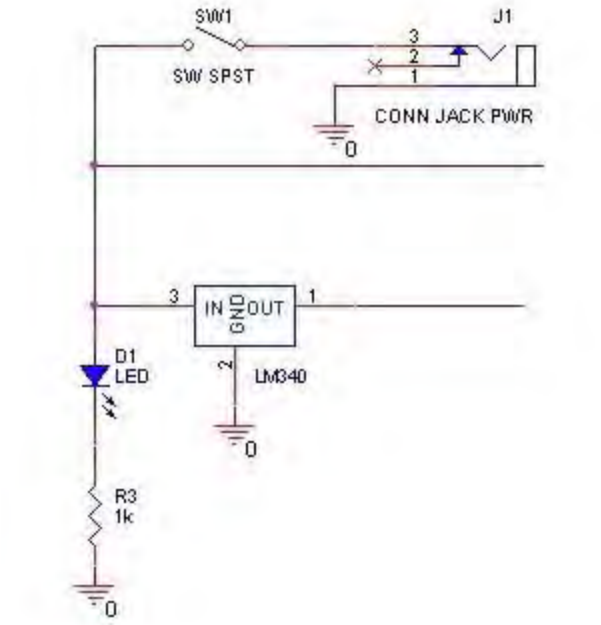


Fig. 3.13. Alimentación del circuito de control

3.4.2. Sección del microcontrolador

Este es el corazón del sistema, el μC realiza los cálculos de posición e interpreta la información que llega desde el sensor del motor y desde la PC.

De forma sencilla el funcionamiento del μC es la siguiente, desde el momento del encendido el μC está pendiente de las instrucciones que pudiera recibir desde la PC, en el momento en el que se le indica la dirección de giro y la cantidad de grados deseada, el μC envía las señales necesarias al motor para que arranque y comienza a recibir las señales desde el sensor óptico

indicando la posición del motor hasta alcanzar la deseada, en ese momento el μC produce las señales necesarias para detener al motor para finalmente entrar en espera de nuevas instrucciones desde la PC.

Todos las demás secciones del circuito se comunican con esta, enviando y recibiendo información y señales de control. Cabe mencionar que el microcontrolador tiene conectado a dos de sus terminales un cristal de cuarzo de 4MHz que tiene la función de reloj del sistema. Además, gracias a las características del μC, se tiene un puerto de 8 bits libre que se comunica al exterior por medio de un conector que puede ser utilizado para distintas funciones, aprovechando la versatilidad del μC, entra las que pueden mencionarse el uso indistinto como puerto de I/O, o ser configurado bit por bit para las distintas entradas y salidas que se deseen o puede dársele a este puerto la tarea de realizar funciones de diagnóstico del μC y de otras partes del circuito. Este puerto corresponde al registro PORTB de la memoria de datos del PIC16F877A y puede ser configurado de manera sencilla a través del registro TRISB.

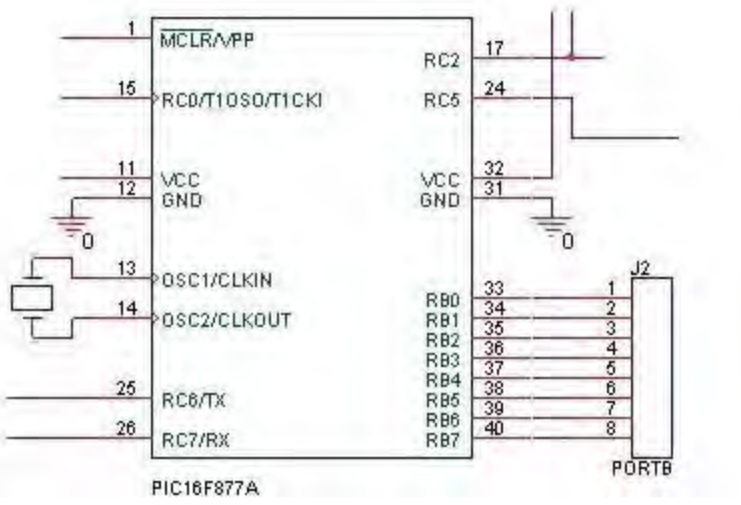


Fig. 3.14. Microcontrolador que opera el sistema de control

3.4.3. Sección de comunicación serial

La comunicación se realiza de forma serial desde el puerto de la PC al módulo USART del μC pasando a través de una etapa intermedia. La función de la etapa intermedia es la de adecuar el

intercambio de información entre ambos, es decir, hace compatible la información intercambiada entre la PC y el μC además controlar este intercambio.

Las funciones se realizan a través de un circuito MAX232, el cual está dividido en 4 secciones[1] que son los convertidores de voltaje DC-DC, drivers RS-232, receptores RS-232 y entradas de control de habilitación de la recepción y transmisión.

Los capacitares son elementos requeridos para el funcionamiento del circuito que sirven para controlar los voltajes de trabajo. Además de lo anterior no tiene más elementos conectados excepto un conector DB9 macho para la conexión al puerto serial de la PC.

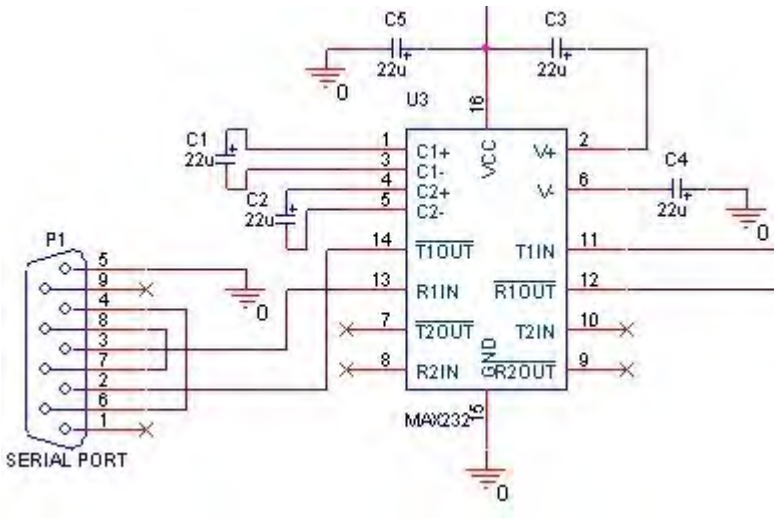


Fig. 3.15. Comunicación serial del circuito de control

3.4.4. Sección del sensor óptico

La posición del motor es determinada mediante un disco, el cual contiene una serie de marcas con 1 grado de separación entre ellas. Utilizando un sensor óptico de obstrucción que es atravesado por el disco del motor se determina la posición de éste mediante un conteo de las marcas y enviando esta información al μC.

Esta sección está conformada únicamente por un sensor óptico de obstrucción y algunas resistencias las cuales son requeridas por los elementos internos del sensor para su operación. De acuerdo a la forma de conexión de los elementos externos, al momento de presentarse la

[1] Ver detalles en las hojas técnicas del Apéndice E

obstrucción, la salida puede ser 0 o 1 lógicos dependiendo de si la configuración de salida es colector común, donde entrega un nivel alto cuando no existe una obstrucción y un nivel bajo cuando se presenta o emisor común, donde hay un nivel bajo cuando no hay obstrucción y un nivel alto cuando esta ocurre, en este caso es una configuración de colector común donde las obstrucciones se dan como niveles bajos (0 lógico) y antes de hacer llegar la señal al microcontrolador se hace pasar esta por un inversor 74HC14 que, además de esta función, cuadra la señal previniendo saltos o caídas en el voltaje.

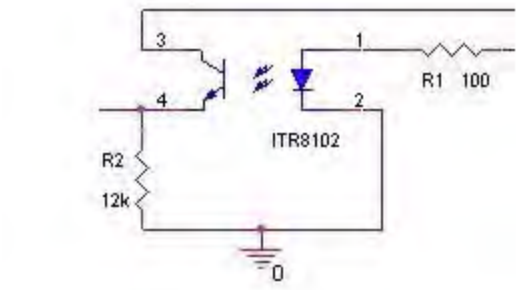


Fig. 3.16. Sensor de posición del sistema de control

3.4.5. Sección de activación del motor

Debido a que el motor requiere de un mayor voltaje de operación no puede ser conectado directamente al circuito de control que, al ser digital, utiliza bajo voltaje. Por esta razón es necesaria una etapa de acoplamiento entre estos dos elementos.

El circuito que controla y activa al motor es un L6203 el cual es un "puente H" que permite, mediante las señales de control recibidas en sus entradas desde el μC rotar el motor en un sentido u otro dependiendo de cual de las entradas recibe un nivel de alto (1 lógico). La compuerta inversora 74HC14 que se encuentra conectada a la entrada IN2 garantiza que las dos entradas no presenten el mismo nivel de voltaje al mismo tiempo.

El L6203 también tiene una entrada de habilitación independiente a las entradas que controlan el sentido de rotación, aunque exista alguna señal presente en las entradas de rotación el motor no girará en ningún sentido hasta recibir un nivel de voltaje alto en la entrada de habilitación.

Al recibir una señal para la rotación y la señal de habilitación, el puente H suministrará un voltaje de 12V al motor de CD, la polaridad del voltaje cambia de acuerdo a si se desea el

sentido de rotación a la derecha o a la izquierda. Aunque en este caso la alimentación del motor de CD a través del L6203 es de 12V este circuito es capaz de manejas voltajes de hasta 48V[1].

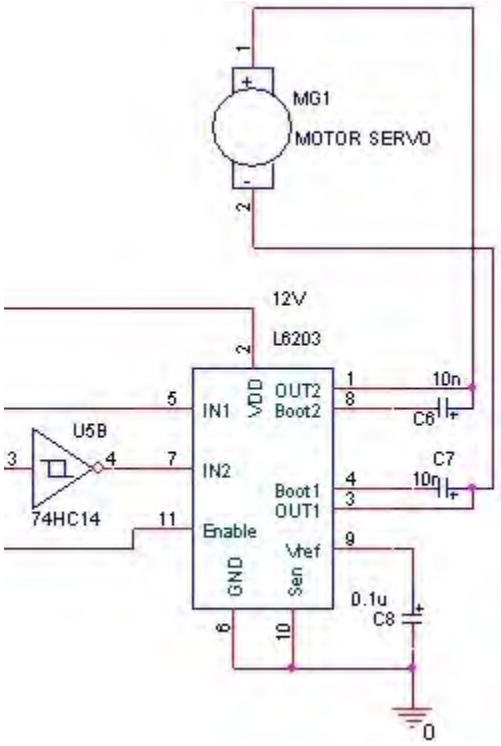


Fig. 3.17. Etapa de potencia para la activación del motor del circuito de control

3.5. Programas de prueba de los elementos del sistema

Una vez armado el circuito de control del motor se llevó a cabo una serie de pruebas de los elementos que lo conforman por separado. Estas pruebas son independientes del programa de control final, su función es la de establecer el funcionamiento individual de cada sección del circuito y realizar programas de prueba para operarlas, estos programas incluyen funciones no incluidas dentro del μC como pueden ser la multiplicación o el modo alternativo de lectura y escritura de la memoria EEPROM.

Muchas de las pruebas involucran el uso del PORTB como puerto de conexión de elementos externos para las pruebas, como pueden ser interruptores, LED's indicadores y otros.

[1] Ver detalles en las hojas técnicas del Apéndice A

Los encabezados que aparecen en los programas de prueba sirven para configurar la descarga al μC y permiten ahorrar pasos al utilizar los programas IC-Prog 1.05C y MPLAB IDE v6.4. Cada encabezado tiene la siguiente función:

`list P=16f877a` Este encabezado le indica al programa MPLAB cuál es el chip para el que se va a compilar de modo que este adecue el código al modelo de μC.

`INCLUDE "P16F877A.inc"` Esta es una parte importante pues es como si fuera un programa previamente escrito que es anexado al programa principal cuando se está trabajando en MPLAB. Este programa alterno contiene todas las referencias que sirven al editar el programa principal, por ejemplo, cuando se trabaja con un μC con el que no se había trabajado anteriormente y se desea hacer un cambio de banco de memoria sería necesario saber que bit del registro STATUS correspondiente a ese μC es el que tiene que ser modificado, sin embargo con el archivo de extensión "inc" sólo se necesita indicar que se desea modificar el bit RP1 sin saber dónde está STATUS y cuál es su bit RP1.

`Radix HEX` Este sirve sólo para indicar que todos los valores numéricos con que se realizan las operaciones entre los registros estarán predeterminadas en base hexadecimal, esto no quiere decir que no sea posible utilizar otras bases numéricas como pueden ser la binaria, octal o decimal durante la escritura y ejecución del programa.

`__CONFIG H'3F39'` Establece la palabra de configuración del μC. Incluyendo esto ya no es necesario configurar la palabra de control en el programa IC-Prog cada vez que se descarga un programa al mismo chip. Hay que hacer notar que

para cada μC la palabra de control es distinta y varía dependiendo del tipo de reloj, el tipo de programación, el WDT y otros parámetros[1].

`#define Banco_0` Sirve para establecer una especie de "Macro". Siempre que el programa encuentre la etiqueta indicada (en este caso Banco_0) realizará la acción que se definió en el encabezado (en este casos realiza la acción de apagar el bit RP0 del registro STATUS lo que causa un cambio de banco de memoria).

`ORG 0` Es la dirección de memoria donde comienzan el programa, es decir, corresponde a la dirección 00h de la memoria.

`ORG 5` Es la dirección de memoria donde se empieza a escribir el programa, esto se hace porque por lo general el vector de interrupción está en la dirección 04h de la memoria así que al empezar a escribir el programa lo hacemos en la dirección 05h para evitar algún problema derivado de las interrupciones.

Antes de cada programa de prueba se da una descripción de la función que debe realizar, los elementos involucrados, los registros utilizados y la lógica del programa.

3.5.1. Prueba 1 – Encendido y apagado del motor

El objetivo del primer programa es el de realizar una prueba de encendido y apagado del motor de forma manual aprovechando el puerto externo que quedó disponible para éste fin. Para realizar la prueba se requieren algunos elementos externos que pueden ser conectados en el puerto de forma externa, por ejemplo, desde una tableta de prototipos. La prueba requiere como elemento externo un interruptor cualquiera, como puede ser del tipo "dip switch", conectado al pin 7 del PORTB y a tierra. Un operador podrá encender y apagar el motor cerrando o abriendo este interruptor, si el interruptor está en la posición de abierto, el μC al trabajar con niveles TTL lo interpretará como un 1 lógico por lo que encenderá el motor y

[1] Ver sección 3.3.2. El programa IC-Prog 1.05C

cuando el interruptor esté en la posición de cerrado el μC lo interpretará como un 0 lógico y detendrá al motor. Este método no es el más adecuado pues en lugar de dejarse abierto circuito debería estar conectado a 5V que representan un 1 lógico pero se realizó de esta forma por simplicidad y utilizar la menor cantidad de elementos externos.

En este caso solo se desea que el motor arranque y se detenga sin importar el sentido que tomo.

En el funcionamiento de este programa sólo se involucran los registros correspondientes a los puertos B y C. Para lograr que los puertos funcionen de manera correcta solo hay que configurar un registro asociado llamado TRIS. Cada bit del registro TRIS controla su correspondiente pin en el puerto. En este caso se están utilizando los puertos B y C por lo que hay que configurar los registros TRISB y TRISC. Encender un bit en TRISB o C (1 lógico) hará que el pin correspondiente en el puerto se configure como entrada. Apagar un bit en TRISB o C (0 lógico) hará que el pin correspondiente en el puerto se configure como salida.

```
      list P=16f877a
      INCLUDE "P16F877A.inc"
      Radix HEX
      __CONFIG H'3F39'

      #define Banco_0   bcf   STATUS,RP0
      #define Banco_1   bsf   STATUS,RP0

            ORG   0
            goto  INICIO
            ORG   5

INICIO      bcf   STATUS,RP1
            Banco_1
            clrf  TRISC       ;configura los puertos
            movlw 80
            movwf TRISB
            Banco_0

CICLO       btfss PORTB,7     ;Revisa el pin 7 que es
            goto  OFF         ;el encendido y realiza
            goto  ON          ;el salto indicado

ON          movlw 24          ;Da las señales de arranque
            movwf PORTC       ;y de habilitación al
            goto  CICLO       ;motor

OFF         clrf  PORTC       ;Mantiene el motor apagado
            goto  CICLO       ;Regresa a la revisión
      END
```

3.5.2. Prueba 2 – Control de giro y habilitación

Este programa está diseñado, al igual que el anterior, para trabajar de forma manual. En este caso se utilizan dos interruptores, que pueden estar contenidos en un solo "dip switch", de estos el primero está conectado al pin 7 del PORTB que realizará la misma función que en el programa anterior. El segundo interruptor estará conectado al pin 0 del PORTB y su función será de indicar el sentido de giro del motor. Si se presenta un nivel bajo en el pin 0 el motor girará a la derecha y si tiene un nivel alto girará hacia la izquierda. Cabe hacer notar que, independientemente del nivel que presente el pin 0 el motor no arrancará hasta que exista un nivel alto en el pin 7 del PORTB. Otro detalle a resaltar es que en el momento de accionar el interruptor del pin 7 para activar al motor, éste girará en el sentido que indique el valor instantáneo del bit de giro.

```
      list P=16f877a
      INCLUDE "P16F877A.inc"
      Radix HEX
      __CONFIG H'3F39'

      #define Banco_0   bcf   STATUS,RP0
      #define Banco_1   bsf   STATUS,RP0

            ORG   0
            goto  INICIO
            ORG   5

INICIO      bcf   STATUS,RP1
            Banco_1
            movlw 81
            movwf TRISB
            clrf  TRISC
            Banco_0

CICLO       btfss PORTB,7
            goto  STOP
            btfss PORTB,0
            goto  DER
            goto  IZQ

STOP        clrf  PORTC
            goto  CICLO

DER         movlw 24
            movwf PORTC
            goto  CICLO

IZQ         movlw 20
            movwf PORTC
            goto  CICLO

            END
```

3.5.3. Prueba 3 – Comunicación entre el PIC y la PC vía USART (Transmisión)

Este es el primer programa de prueba donde se involucra la interacción entre el μC y la PC. El objeto de este programa es comprobar paso a paso el funcionamiento del módulo USART del PIC16F877A y del circuito de comunicación serie MAX232.

Esta prueba consistirá en comprobar el modo de transmisión de los elementos desde el módulo USART del μC hacia la PC, en este caso no es necesario conectar ningún elemento externo al PORTB, solo se tiene que conectar el circuito de control al puerto serie de la PC.

Una vez escrito el programa se enviará desde la memoria flash del μC una palabra, en este caso será "UNAM", que será transmitida letra por letra y desplegada en la pantalla de la PC utilizando la aplicación de Windows "Hyper Terminal" localizada en "Inicio\Todos los programas\Accesorios\Comunicaciones" para la versión XP. Es necesario indicarle a la aplicación una vez abierta un nombre para la conexión, el puerto por el que se va a realizar la conexión, por ejemplo el COM3, la cantidad de bits por segundo, que en este caso es de 9600, y que el control de flujo es de tipo Xon/Xoff. Una vez hechos estos ajustes, en el momento de encender el circuito este transmitirá a la pantalla de la Hyper Terminal la palabra "UNAM".

Para este caso se involucran el PORTC y los registros asociados al módulo USART como son PIR1, SPBRG, RCSTA, RCREG, TXSTA y TXREG. Al momento de encender el circuito el programa enviará una serie de letras a través del módulo USART hacia la PC donde podrán ser vistas por medio de la hyper terminal, la forma en que se envían los caracteres desde el programa hace que sean identificados como caracteres alfanuméricos por la PC, esto no es práctico cuando la cantidad de caracteres es grande pues el programa tiende a hacerse muy grande. Pueden utilizarse otros métodos para almacenar los caracteres para luego enviarlos, por ejemplo almacenándolos en una memoria, pero hay que tener en cuenta que en esta forma no serán interpretados como alfanuméricos por lo que habrá que revisar las tablas de códigos ASCII.

El tipo de transmisión es asíncrono con un ancho de palabra de 8 bits sin bit de paridad y a una velocidad de transmisión de 9600 baudios (High Speed).

A continuación se da una descripción de cómo hay que utilizar y configurar los registros asociados al módulo USART[1]. Los registros asociados son SPBRG, TXSTA, RCSTA, TXREG, RCREG, PIE1 y PIR1.

[1] Para mayores detalles ver la sección correspondiente al módulo USART 3.2.5.

- Registro TXSTA (dirección 98h)

Este es el registro que controla la transmisión y de acuerdo a sus bits de configuración debe ser cargado con el siguiente valor.

$$\frac{b_7}{-}\frac{b_6}{0}\frac{b_5}{1}\frac{b_4}{0}\quad\frac{b_3}{-}\frac{b_2}{1}\frac{b_1}{-}\frac{b_0}{0} = 24\text{h}$$

- Registro RCSTA (dirección 18h)

Este es el registro que controla la recepción y de acuerdo a sus bits de configuración debe ser cargado el siguiente valor.

$$\frac{b_7}{1}\frac{b_6}{0}\frac{b_5}{-}\frac{b_4}{1}\quad\frac{b_3}{0}\frac{b_2}{-}\frac{b_1}{-}\frac{b_0}{-} = 90\text{h}$$

- Registro SPBRG (dirección 99h)

Es como un temporizador que controla la velocidad de transmisión. Como ya se mencionó la velocidad a utilizar es de 9600 baudios (High Speed) en modo asíncrono[1]. El valor que debe ser cargado en este registro se determina por medio de las fórmulas de la velocidad de transmisión por lo que:

$$\text{Vel. de Trans.} = \frac{\text{F}_{\text{osc}}}{16(\text{X}+1)}$$

Donde X = valor del registro SPBRG que es el que controla el periodo del temporizador de 8 bits, por lo que sólo se le pueden cargar valores de 0 a 255.

$$9600 = \frac{4\text{MHz}}{16(\text{X}+1)}$$
$$9600(16)(X+1) = 4MHz$$
$$X = \frac{4MHz}{9600(16)} - 1$$
$$X = 25.041 \approx 25$$

$$X = \frac{b_7}{0}\frac{b_6}{0}\frac{b_5}{0}\frac{b_4}{1}\quad\frac{b_3}{1}\frac{b_2}{0}\frac{b_1}{0}\frac{b_0}{1} = 19\text{h}$$

NOTA: *en las partes marcadas con "-" se refiere a que no importa el estado del bit, pero por convención en este proyecto se toman como si fueran ceros.*

[1] Ver tablas de la velocidad de transmisión para el modo asíncrono.

- Registro TXREG (dir 19h)

Es necesario cargar en este registro cada dato de 8 bits que se desee transmitir a través del puerto serial. Esto se debe hacer de uno en uno y esperar a que el dato cargado haya sido transmitido antes de cargar el siguiente.

```
      list p=16F877A
      INCLUDE "P16F877A.inc"
      Radix HEX
      __CONFIG H'3F39'

#define     Banco_0     bcf   STATUS,RP0
#define     Banco_1     bsf   STATUS,RP0

            ORG   0
            goto  INICIO
            ORG   5

INICIO      bcf   STATUS,RP1
            Banco_1
            movlw 19        ;Zona de
            movwf SPBRG     ;configuración
            movlw 24        ;de registros
            movwf TXSTA     ;asociados
            bcf   PIE1,4    ;al modulo
            Banco_0         ;USART
            movlw 90
            movwf RCSTA

            movlw 'U'       ;Los caracteres
            movwf TXREG     ;a enviar se
            call  CHECA     ;cargan en el
            movlw 'N'       ;registro TXREG
            movwf TXREG     ;y se revisa el
            call  CHECA     ;bit de bandera
            movlw 'A'       ;de transmisión
            movwf TXREG     ;en PIR1 por
            call  CHECA     ;medio de la
            movlw 'M'       ;subrutina CHECA
            movwf TXREG     ;y regresa a la
            call  CHECA     ;letra siguiente
FIN         goto  FIN

CHECA       btfss PIR1,4
            goto  CHECA
            return

      END
```

- Registro RCREG (dir 1Ah)

En este registro es donde se almacena cada dato recibido por el puerto serial. Cuando el registro RCREG es leído también se vacía para estar preparado para el siguiente dato a recibir.

- Registro PIE1 (dir 8Ch)

Este controla las habilitaciones. El bit TXIE (PIE1<4>) controla la habilitación de la interrupción en el modo de transmisión y el bit RCIR (PIE1<5>) controla la habilitación de la interrupción en el modo de recepción. En ambos casos las habilitaciones se deshabilitan apagando los bits correspondientes y se habilitan encendiéndolos.

- Registro PIR1 (dir 0Ch)

Aquí se encuentran contenidos los bits de bandera que proporcionan información sobre el estado de la transmisión TXIF (PIR1<4>) o la recepción RCIF (PIR1<5>). Estos bits son de solo

lectura y son apagador vía hardware. Cuando alguno de estos bits se enciende indica que la transmisión o la recepción, respectivamente, han sido completadas.

3.5.4. Prueba 4 – Comunicación entre el PIC y la PC vía USART (Recepción)

Como la anterior, esta prueba trata sobre la comunicación entre el μC y la PC. En este caso se prueban los elementos en modo de recepción. Una vez más se va a hacer uso del puerto de prueba conectando en cada pin un diodo LED con su resistencia de protección y a tierra. Estos 8 LED's representarán los bits que forman el código ASCII, que puede ser consultado en al "Apéndice C" de este trabajo, que será enviado desde la PC, al presionar cualquier tecla, hacia el circuito del μC utilizando el módulo USART para luego ser mostrado en los LED's en forma binaria. Hay que tener en cuenta que el código ASCII es transmitido en formato hexadecimal y que la mayoría de los códigos ASCII son de 8 bits, sin embargo al visualizarlos no son necesarios los primeros 4 bits, es decir, si se presionó la tecla correspondiente al número '5', el código enviado será un 35h por lo que en los LED's será desplegado en binario el número '0011 0101' pero sólo nos interesan los 4 bits menos significativos es decir '0101' que equivale a '5' en decimal.

```
      list p=16F877A
      INCLUDE "P16F877A.inc"
      Radix HEX
      __CONFIG H'3F39'

      #define  Banco_0  bcf   STATUS,RP0
      #define  Banco_1  bsf   STATUS,RP0

            ORG   0
            goto  INICIO
            ORG   5

INICIO      bcf   STATUS,RP1
            Banco_1
            clrf  TRISB       ;Configuración
            movlw 19          ;de registros
            movwf SPBRG
            movlw 24
            movwf TXSTA
            bcf   PIE1,4      ;Deshabilita las
            bcf   PIE1,5      ;interrupciones
            Banco_0
            movlw 90
            movwf RCSTA

CHECA       btfss PIR1,5      ;Revisa el bit
            goto  CHECA       ;de bandera de
            movf  RCREG,0     ;recepción,
            movwf PORTB       ;mueve el valor
            goto  CHECA       ;de RCREG al
                              ;PORTB y regresa
      END
```

3.5.5. Prueba 5 – Multiplicación y separación en registros

Para la realización del proyecto se presentaron dos problemas de gran importancia. El primero es que para lograr que el sistema tuviera la suficiente resolución para lograr movimientos de 1 grado se necesita que el sistema pueda manejar un rango de valores desde 0 hasta 360 por lo que 8 bits no son suficientes para manejarlo, y como todos los registros que maneja el PIC16F877A son de 8 bits se hace necesario utilizar dos registros para almacenar los datos correspondientes, de modo que hay que realizar un procedimiento para que un registro contenga los 8 bits menos significativo (estos registros tendrán la terminación L, como GRADOSL) y otro que contenga el bit más significativo (registros con terminación H, como GRADOSH). Para lograr la separación se optó por aprovechar el bit de carry como indicador de los datos que requieran del noveno bit, de modo que los 8 bits menos significativos queden almacenados en los registros con terminación 'L' y al existir el carry se cargue con un 01h los registros correspondientes con terminación 'H'.

El segundo problema que se presentó es que como los datos que llegan desde la PC lo hacen en forma serial, cada dato que representa a un dígito llega de forma separada por lo que hay que esperar a que lleguen todos los datos y asignarles su valor correspondiente, es decir, si desde la PC se envía la cantidad de 235° el primer número en llegar que será el 2 al que deben ser asignadas las centenas, el segundo que es el 3 las decenas y el tercero que es el 5 las unidades, sin embargo al hacer esto se presenta el problema de que en el conjunto de instrucciones del PIC16F877A no existe ninguna que pueda realizar una multiplicación haciendo del número 2 un 200 del 3 un 30 y el 5 dejarlo como tal por lo que es necesario realizar un proceso previo de sumas sucesivas en las que de un registro precargado con el número 100 se sume dos veces para logra el valor adecuado, luego otro registro precargado con un 10 sea sumado 3 veces para llegar al valor correcto y luego sumar el resultado de estas dos operaciones con el número 5 obteniendo la cantidad final para luego separarla en los dos registros a los que esté dirigida.

Por todo lo que representan los problemas anteriores este programa es bastante amplio como para sólo poner notas de su funcionamiento por lo que dentro del mismo están indicados una serie de números como etiquetas de texto los cuales serán explicados más adelante.

El objetivo general de este programa de prueba es que al escribir una cantidad en el teclado de la PC, que está sea transferida por el puerto serie hacia el μC donde será interpretada, acondicionada y almacenada en los registros correspondientes para luego ser mostrada en el puerto auxiliar del circuito de control mediante LED's de manera similar a como se realizó en el ejemplo anterior.

```
      list p=16F877A
      INCLUDE "P16F877A.inc"
      Radix HEX
      __CONFIG H'3F39'

#define  Banco_0  bcf   STATUS,RP0
#define  Banco_1  bsf   STATUS,RP0

      GRADOSL     EQU   20    ; -1-
      GRADOSH     EQU   21
      CENTENA     EQU   22
      DECENA      EQU   23
      UNIDAD      EQU   24
      CONTEO      EQU   25
      SUMACEN     EQU   26
      SUMADEC     EQU   27
      TRES        EQU   28
      RESTRIN     EQU   29

            ORG   0
            goto  INICIO
            ORG   5

INICIO      bcf   STATUS,RP1
            Banco_1
            clrf  TRISB       ; -2-
            bcf   TRISC,2
            movlw 19          ; -3-
            movwf SPBRG
            movlw 24
            movwf TXSTA
            bcf   PIE1,4
            bcf   PIE1,5
            Banco_0
            movlw 90
            movwf RCSTA

CORRER      clrf  CENTENA     ; -4-
            clrf  DECENA
```

```
            clrf  UNIDAD
            clrf  PORTB
            clrf  PORTD
            clrf  GRADOSL
            clrf  GRADOSH
            movlw 03
            movwf TRES
            movlw 64
            movwf SUMACEN
            movlw 0A
            movwf SUMADEC
RECIBE      btfss PIR1,5      ; -5-
            goto  RECIBE
            movlw 0D
            subwf RCREG,w
            btfsc STATUS,Z
            goto  MULT
            movf  RCREG,w
            movwf RESTRIN
            movlw 0F0
            andwf RESTRIN,1
            movlw 30
            subwf RESTRIN,1
            btfss STATUS,Z
            goto  RECIBE
            movf  RCREG,w
            movwf RESTRIN
            movlw 0F
            andwf RESTRIN,1
            bcf   STATUS,C
            rrf   RESTRIN,1
            incf  RESTRIN,1
            incf  RESTRIN,1
            incf  RESTRIN,1
            btfsc RESTRIN,3
            goto  RECIBE
            movf  DECENA,w    ; -6-
            movwf CENTENA
            movf  UNIDAD,w
```

```
            movwf DECENA
            movf  RCREG,w
            movwf TXREG
            call  CHECA
            movlw 0F
            andwf RCREG,w
            movwf UNIDAD
            decf  TRES
            btfsc STATUS,Z
            goto  ESPERA
            goto  RECIBE
ESPERA      movlw 0D          ; -7-
            subwf RCREG,w
            btfsc STATUS,Z
            goto  MULT
            goto  ESPERA

MULT        movf  UNIDAD,w    ; -8-
            movwf GRADOSL
            movf  DECENA,w
            movwf CONTEO

DEC10       decf  CONTEO
            btfsc STATUS,Z
            goto  PASO
            movlw 0A
            addwf SUMADEC,1
            goto  DEC10

PASO        movf  CENTENA,w   ;-9-
            movwf CONTEO

DEC100      decf  CONTEO
            btfsc STATUS,Z
            goto  SUMA
            movlw 64
            addwf SUMACEN,1
            goto  DEC100
```

```
SUMA        movf  SUMADEC,w   ;-10-
            addwf GRADOSL,1
            movf  SUMACEN,w
            addwf GRADOSL,1
            btfss STATUS,C
            goto  NOCARRY
            goto  CARRY
CARRY       bsf   GRADOSH,0   ;-11-
            bsf   PORTC,2
            goto  PUERTO

NOCARRY     clrf  GRADOSH     ; -12-
            bcf   PORTC,2

PUERTO      movf  GRADOSL,w   ; -13-
            movwf PORTB

NUEVO       btfss PIR1,5      ; -14-
            goto  NUEVO
            movlw 0D
            subwf RCREG,w
            btfss STATUS,Z
            goto  NUEVO
            movlw 0D
            movwf TXREG
            call  CHECA
            movlw 0A
            movwf TXREG
            call  CHECA
            goto  CORRER

CHECA       btfss PIR1,4
            goto  CHECA
            return

      END
```

-1- Para realizar el programa fue necesario crear una serie de registros en la zona de registros de propósito general (GPR) en la memoria de datos[1] del μC. Estos GPR son creados por el usuario para contener información que utiliza el programa principal y que no se desea perder durante las operaciones re realiza el mismo.

De los registros creados, los nombrados "GRADOS" son los destinados a contener el valor de la posición den motor, que corresponde a un valor entre 0 y 360 grados. Los registros "CENTENA", "DECENA" y "UNIDAD" son utilizados para almacenar los valores

[1] Ver tabla en la sección 1.3.3.2.

correspondientes de la cantidad en grados recibida. Los registros “CONTEO”, “TRES”, “SUMACEN” y “SUMADEC” se crearon para ser usadas en las funciones que realizan la operación de multiplicación. Finalmente el registro “RESTRIN” sirve para limitar las variables recibidas desde el teclado únicamente a número.

-2- Configuración de los puertos B y C. Una vez recibido el dato y luego interpretado por el μC es enviado a puerto para observarlo por medio de LED’s en forma binaria. Debido a que los datos son de nueve bits el PORTB no es suficiente por si solo por lo que se va a aprovechar el bit de encendido del motor en PORTC<2> como indicador del noveno bit.

-3- Configuración y carga de los registros asociados al módulo USART y deshabilitación de las interrupciones.

-4- Limpieza y carga de los registros creados. En este punto lo que se hace es eliminar la posible basura que podrían contener los registros pues no es posible saber que es lo que contienen al momento de encender el sistema. Además, los registros “TRES”, “SUMACEN” y “SUMADEC” son cargados con valores definidos que serán utilizados durante la operación del programa.

-5- Primero hace una revisión de si se ha sido recibido algún dato por el puerto serial por medio del bit de bandera correspondiente en el registro PIR1. Carga el registro W con 0Dh que en ASCII es un retorno de carro, es decir, la tecla ENTER para después restarlo de lo que contiene RCREG y finalmente revisa el bit Z (cero) del registro STATUS para saber si el resultado de la sustracción fue cero. Esto es para que si la cantidad en grados recibida es de uno o dos dígitos el programa no quede esperando por el tercero pues al recibir el ENTER continúa con lo siguiente. Toda la parte siguiente del programa que corresponde a la etiqueta “RECIBE” se encarga de realizar una serie de comprobaciones que involucran al registro “RESTRIN”, esto es para desechar cualquier dato que no sea un número, esto se hace en base a las tablas de código ASCII.

-6- Aquí lo que se hace es una especie de corrimiento en donde lo que contenía el registro DECENA se mueve al registro CENTENA, lo que había en unidad se mueve a DECENA y lo que se recibió en RCREG se mueve a UNIDAD. La parte en donde está involucrado el TXREG es para que el dato recibido y que fue enviado por el teclado pueda ser visto en el monitor de la computadora, pues de otra forma aunque se presionen las teclas no hay manera de saber de forma visual que es lo que se presionó, y llama a la subrutina que revisa que se haya enviado.

La operación lógica AND aplicada al dato que se recibió con un 0Fh es para convertir el dato de formato ASCII a binario para luego almacenarlo en el registro UNDAD. El registro TRES se usa para llevar un control de cuando han sido recibidos los tres dígitos por medio de decrementos.

-7- Una vez recibidos todos los dígitos que componen la cantidad de grados el programa espera por una señal para iniciar el proceso de multiplicación, esta señal viene de la tecla ENTER. Si no se recibe esta señal el programa queda en espera de esta hasta que ocurre y salta a la sección marcada con la etiqueta “MULT”.

-8- Se carga el registro GRADOSL con el valor de UNIDAD y se carga el registro CONTEO con el valor de DECENA. A continuación se realiza una serie de sumas consecutivas controladas por CONTEO que representan la cantidad total de decenas a sumar, es decir, se suma diez con diez las veces que indique CONTEO así si se quiere un treinta se suma diez tres veces. Al finalizar esto el resultado se almacena en SUMADEC y se pasa a la suma de las centenas en la etiqueta PASO.

-9- De manera análoga, en esta parte del programa se realiza lo mismo que en la anterior. Aquí se realiza la suma de las centenas necesarias cargando a CONTEO con el dato de CENTENA y almacenando el resultado de las sumas en el registro SUMACEN.

-10- Ya realizadas las sumas sucesivas de las decenas y centenas se hace una serie final de sumas en donde se conjuntan los registros GRADOSL, que ya contiene el valor del registro UNIDAD, con SUMADEC y SUMACEN. El resultado final queda almacenado en el registro GRADOSL y se revisa si existe el bit de carry en el registro STATUS para saber si el noveno bit existe.

-11- En caso de existir el noveno bit se enciende su correspondiente del registro GRADOSH y se enciende el bit del puerto que se le asignó para la prueba para luego saltar a la etiqueta PUERTO.

-12- Si no hay carry se limpia el registro GRADOSH y el bit correspondiente en PORTC<2> para luego continuar de forma normal con la sección de PUERTO.

-13- Aquí lo único que se hace es mover el valor de GRADOSL al PORTB. Con esto finaliza toda la operación de multiplicación y separación del resultado en dos registros.

-14- La última sección es para esperar si se desea realizar una nueva operación. El programa queda en espera de la tecla ENTER para empezar una nueva operación. En esta sección están incluidas algunas unas instrucciones para desechar cualquier otra señal que no sea el ENTER y en caso de recibirlo cambia a la siguiente línea en la pantalla de la PC y salta hasta la etiqueta CORRER para reiniciar todo el proceso.

NOTA: *es importante mencionar que este método de multiplicación no es aplicable a cantidades mayores de 511 pues el sistema está planeado sólo para un rango de 0 a 360.*

3.5.6. Prueba 6 – Subrutina de retardo empleando en módulo del Timer0

Esta es una prueba para crear una subrutina de retardo empleando el módulo del Timer0. El objeto de esta prueba es que, ya que el método de frenado del motor será por medio de una polarización inversa, se necesario que esta polarización esté controlada y se aplique durante un

periodo de tiempo preciso. Por lo anterior, se hace uso del Timer0, para crear un retardo que tenga la duración necesaria para que la señal aplicada detenga el motor, pero no demasiada como para que el motor llegue a rotar en sentido opuesto.

Por lo tanto, lo que hará esta prueba, es encender un LED conectado al pin 7 del PORTB durante medio segundo, apagarlo durante otro medio segundo y así sucesivamente.

El Timer0 posee dos modos de operación, como temporizador y como contador, ambos de 8 bits. Para seleccionar modo de operación del módulo del Timer0 se utiliza el bit T0CS (OPTION_REG<5>). Encendiendo este bit se selecciona el modo de contador y apagándolo se selecciona el modo de temporizador[1].

En el registro INTCON (dirección 0Bh) se encuentra el bit de bandera que indica el sobreflujo del Timer0, TMR0IF (INTCON<2>) la cual una vez encendida debe ser apagada vía software, es decir, que el µC no puede apagarla por si solo así que el usuario tiene que agregar alguna instrucción en el programa que la apague. Las interrupciones pueden ser habilitadas o deshabilitadas encendiendo o apagando el bit TMR0IE (INTCON<5>).

El registro TMR0 depende de la

```
      list p=16F877A
      INCLUDE "P16F877A.inc"
      Radix HEX
      __CONFIG H'3F39'

#define     Banco_0     bcf STATUS,RP0
#define     Banco_1     bsf STATUS,RP0

      TEMPO1      EQU   21          ; -1-

            ORG   0
            goto  INICIO
            ORG   5

INICIO      bcf   STATUS,RP1
            Banco_1
            movlw 07                ; -2-
            movwf OPTION_REG
            bcf   TRISB,7
            Banco_0
            clrf  PORTB

CICLO       bsf   PORTB,7           ; -3-
            call  DELAY
            bcf   PORTB,7           ; -4-
            call  DELAY
            goto  CICLO             ; -8-

DELAY       movlw 0A                ; -5-
            movwf TEMPO1
            call  DEL50
            return

DEL50       bcf   INTCON,2          ; -6-
            movlw 3D
            movwf TMR0

DEL50_1     btfss INTCON,2          ; -7-
            Goto  DEL50_1
            decfszTEMPO1,1
            goto  DEL50
            return

      END
```

[1] Revisar la sección referente al Timer0 3.2.4 o la sección correspondiente del manual del PIC16F877A.

frecuencia del reloj (cristal) que en este caso es de 4MHz y de una selección del prescaler o divisor que es controlado por el registro OPTION_REG. El tiempo de funcionamiento del Timer0 puede determinarse con la fórmula:

$$T = 4(T_{OSC})(\text{Pr}\, escaler)(TMR0)$$

Donde:

T – Periodo de tiempo

T_{OSC} – Frecuencia del reloj

Prescaler – Valor cargado en el registro OPTION_REG<2:0>

TMR0 – Valor cargado en éste registro

Existe una limitación al realizar el cálculo del tiempo pues el registro TMR0 solo admite valores de 0 a 255, o sea un conteo de 256 estados, por lo que si el valor calculado es mayor, una alternativa puede ser utilizar un tiempo más corto y luego hacer que la subrutina de tiempo se repita el número de veces que sea necesario para cubrir el tiempo total utilizando otra subrutina de repetición.

Así para cubrir el tiempo de 0.5 seg. se puede utilizar un tiempo de duración del temporizador de 50ms con un prescaler de 256 divisiones y repetir la subrutina diez veces, por lo que se tiene:

$$TMR0 = \frac{T}{4(T_{OSC})(\text{Pr}\, escaler)}$$

$$TMR0 = \frac{50mseg}{4(4MHz)(256)}$$

$$TMR0 = \frac{50mseg}{4\left(\frac{1}{4MHz}\right)(256)}$$

$$TMR0 = \frac{0.5seg}{4(0.25\mu s)(256)}$$

$$TMR0 = \frac{50mseg}{(0.256ms)}$$

$$TMR0 = 195.312 \approx 195$$

El valor correcto que debe cargarse en el registro TMR0 (dirección 01h) es la diferencia entre su valor máximo y el calculado, por lo que:

256 – 196 = 61

$$\underset{0}{b_7}\ \underset{0}{b_6}\ \underset{1}{b_5}\ \underset{1}{b_4}\quad \underset{1}{b_3}\ \underset{1}{b_2}\ \underset{0}{b_1}\ \underset{1}{b_0} = 3\text{Dh}$$

Mientras que el registro OPTION_REG (dirección 81h), después de seleccionar el prescaler y el modo de operación, quede da la siguiente forma:

$$\underset{-}{b_7}\ \underset{-}{b_6}\ \underset{0}{b_5}\ \underset{-}{b_4}\quad \underset{-}{b_3}\ \underset{1}{b_2}\ \underset{1}{b_1}\ \underset{1}{b_0} = 07\text{h}$$

NOTA: *En los espacios marcadas con* "-" *se refiere a que no importa el estado del bit, pero por convención en este proyecto se toman como si fueran ceros.*

-1- Este es un registro creado por el usuario para que contenga el número de veces que es necesario que se repita la subrutina de retardo.

-2- Configura el registro OPTION_REG y el PORTB asegurándose además de limpiarlo.

-3- Enciende el pin 7 del PORTB y llama la subrutina DELAY.

-4- Apaga el pin 7 del PORTB y llama la subrutina DELAY.

-5- Carga un diez (0Ah) en el registro TEMPO1 y llama la subrutina DEL50.

-6- Apaga el bit de bandera de sobreflujo del registro INTCON y carga el valor calculado por medio de la fórmula en el registro TMR0

-7- Revisa si la bandera de sobreflujo está encendida, lo que indica que el retardo ya terminó, si está apagada continúa esperando y si está encendida decrementa el valor del registro TEMPO1. Cuando este registro llegue a cero el programa dará por terminada la subrutina regresando hasta la sección con la etiqueta CICLO para ahora apagar el pin 7 del PORTB y repetir todas las subrutinas de retardo.

-8- Esto mantiene el programa en una forma cíclica alternando entre el periodo de encendido y el de apagado.

3.5.7. Prueba 7 – Prueba del sensor óptico y el Timer1 en modo de contador

Ya que la detección de la posición del motor se realiza mediante la detección de una serie de marcas en el disco del motor mediante el sensor óptico, el μC interpreta estas señales como una serie de pulsos que llegan desde el sensor.

Para cuantificar esta serie de pulsos se utilizó el módulo del Timer1 en modo de contador. Cada que uno de estos pulsos llega al μC el Timer1 se incrementa en '1' y es comparado contra los registros que contienen la información de la posición indicada al motor desde la PC.

La prueba es para cerciorarse de que los pulsos recibidos desde el sensor óptico de interrupción como reloj externo son bien recibidos y no existen rebotes u otros efectos indeseables en la señal.

Aprovechando esto, también se puede probar la operación del módulo del Timer1 en el modo de contador (ver la parte del programa donde se configura el Timer1).

El funcionamiento es el siguiente, al encender el sistema este se activará utilizando la señal proveniente del sensor óptico como reloj externo, y comenzará a incrementar el conteo cada que sea interrumpido el haz infrarrojo del sensor. Cuando el conteo llegue a 16 (1Fh), el cuál será mostrado por LED's en el PORTB, detendrá el conteo apagando todos los LED's para iniciar un nuevo conteo hasta 7 (07h) para volver a apagar todos los LED's y reiniciar de nueva cuenta el conteo de 16, y así sucesivamente.

Es importante hacer notar que al realizar este programa se empleo una nueva estructura para realizar el cambio entre los bancos de la memoria de datos. Esta estructura es un "macro", los cuales deben ser creados al principio del programa, antes de los ORG. El macro está constituido de tres parte, una etiqueta o nombre que se asigna al macro, las acciones que debe realizar y una instrucción adicional "ENDM" que significa que ahí es donde termina el macro. De esta manera cada vez que el programa encuentre una etiqueta que corresponda al nombre de un macro realizará un salto rápido al principio del programa para ejecutar las instrucciones correspondientes al macro y regresará al punto del programa en donde se encontraba. Esto es útil pues sin importar el banco de memoria en el que se encuentre el programa interpretará la

instrucción sin problemas a diferencia del método anterior donde había que modificar, de forma individual, los bits del registro STATUS para poder cambiar del banco 0 al banco 3.

```
      List p=16F877A
      INCLUDE "P16F877A.inc"
      Radix HEX
      __CONFIG H'3F39'

;******** Sección de Macros ********

      Banco_0     MACRO       ; -1-
      bcf         STATUS,RP0
      bcf         STATUS,RP1
            ENDM

      Banco_1     MACRO       ; -2-
      bsf         STATUS,RP0
      bcf         STATUS,RP1
            ENDM
;**********************************

            ORG   0
            goto  CONPORT
            ORG   5

CONPORT     Banco_1
            clrf  TRISB       ; -3-
            bsf   TRISC,0

CONTIMER    Banco_1
            bcf   PIE1,0      ; -4-
            Banco_0
            clrf  TMR1L       ; -5-
            movlw 03          ; -6-
            movwf T1CON

INICIO      movf  TMR1L,w     ; -7-
            movwf PORTB
            movlw 10          ; -8-
            subwf TMR1L,0
            btfss STATUS,Z
            goto  INICIO      ; -9-
            clrf  TMR1L
            clrf  PORTB

PARTE2      movf  TMR1L,w     ;-10-
            movwf PORTB
            movlw 07          ;-11-
            subwf TMR1L,0
            btfss STATUS,Z
            goto  PARTE2      ;-12-
            clrf  TMR1L
            clrf  PORTB
            goto  INICIO

      END
```

-1- Crea un macro para cambiar de forma automática al banco 0 de la memoria.

-2- Crea un macro para cambiar de forma automática al banco 1 de la memoria.

-3- Configura los puertos. Todo el PORTB es configurado como salidas y el PORTC<0> como entrada.

-4- Deshabilita la interrupción.

-5- Limpia el registro TMR1L en caso de que en el momento del encendido tenga algún dato al azar cargado.

-6- Configura el registro de control del módulo del Timer1.

-7- Mueve el contenido del registro TMR1L al PORTB para ser mostrado.

-8- Comprueba si el conteo ha llegado al valor deseado por medio de una resta y de revisar el bit de cero del registro STATUS.

-9- Según el estado del bit de cero del registro STATUS regresa para seguir incrementando la cuenta o limpia el registro TMR1L y el PORTB para continuar con el resto del programa.

-10- Mueve el contenido del registro TMR1L al PORTB para ser mostrado.

-11- Comprueba si el conteo ha llegado al valor deseado por medio de una resta y revisando el bit de cero del registro STATUS.

-12- Dependiendo del estado del bit de cero del registro STATUS regresa para seguir incrementando la cuenta o limpia tanto el registro TMR1L como el PORTB para saltar a la etiqueta INICIO del programa y comenzar el ciclo nuevamente.

3.5.8. Prueba 8 – Escritura de la memoria EEPROM

Con esta y la siguiente pruebas se va a realizar una comprobación del funcionamiento de la memoria EEPROM contenida en el μC.

Como se mencionó en la sección correspondiente a la memoria EEPROM existen dos métodos de escribir en ella, una aprovechando el programa de grabación IC-Prog 1.05C y la otra utilizando el método descrito en el manual de operación del PIC16F877A. En este ejemplo se va a utilizar el método de grabación recomendado por el fabricante. Hay que hacer notar que la secuencia de habilitación de la memoria, su escritura y otros pasos deben de seguirse en un orden específico o de lo contrario el proceso de lectura no se realizará o por lo menos no de manera correcta. Otra cosa a mencionar es que las cantidades que se cargan en el registro EECON2 son necesarias para realizar las funciones relacionadas a la escritura de la memoria y los valores son dados por el fabricante.

El funcionamiento del programa es el siguiente, el μC esperará a que llegue un dato desde el teclado de la PC a través del puerto serial mostrándolo en la pantalla para luego almacenarlo en la memoria EEPROM, para no hacerlo demasiado complicado se va a limitar a solo 16 localidades

de la memoria EEPROM aunque puede ocuparse la totalidad de localidades de que se disponga. Todo lo que sea escrito en la memoria será leído en la prueba siguiente.

```
      List p=16F877A
      INCLUDE "P16F877A.inc"
      Radix HEX
      __CONFIG H'3F39'

;******** Sección de Macros ********

Banco_0     MACRO             ; -1-
            bcf   STATUS,RP0
            bcf   STATUS,RP1
      ENDM

Banco_1     MACRO
            bsf   STATUS,RP0
            bcf   STATUS,RP1
      ENDM

Banco_2     MACRO
            bcf   STATUS,RP0
            bsf   STATUS,RP1
      ENDM

Banco_3     MACRO
            bsf   STATUS,RP0
            bsf   STATUS,RP1
      ENDM
;*********************************

      DIREC       EQU   110   ; -2-
      Limite      EQU   111

            ORG   0
            goto  CONSERIE
            ORG   5

CONSERIE    Banco_1           ; -3-
            movlw 19
            movwf SPBRG
            movlw 24
            movwf TXSTA
            bcf   PIE1,4
            bcf   PIE1,5
            Banco_0
            movlw 90
            movwf RCSTA
            bcf   INTCON,7

CONREG      Banco_2
            clrf  DIREC       ; -4-
            movlw 0F
            movwf Limite

INICIO      Banco_0
            btfss PIR1,5      ; -5-
            goto  INICIO
            Banco_2
            movf  DIREC,0     ; -6-
            movwf EEADR
            Banco_0
            movf  RCREG,0     ; -7-
            movwf TXREG
            call  CHECA
            Banco_2
            movwf EEDATA      ; -8-
            Banco_3
            bcf   EECON1,7    ; -9-
            bsf   EECON1,2
            movlw 55
            movwf EECON2
            movlw 0AA
            movwf EECON2
            bsf   EECON1,1
            bcf   EECON1,2
            Banco_2
            incf  DIREC,1    ; -10-
            decfsz Limite,1
            goto  INICIO
FIN         goto  FIN

CHECA       btfss PIR1,4
            goto  CHECA
            return

      END
```

-1- Macros creados para realizar los cambios entre los bancos de memoria de manera más sencilla pues los registros correspondientes a la memoria EEPROM se encuentran en los bancos 2 y 3.

-2- Registros creados por el usuario que se utilizan en los procesos que realiza el programa principal.

-3- Configuración de los registros asociados al módulo USART y deshabilitación de las interrupciones.

-4- Configuración de los registros creados por el usuario.

-5- Revisa si se ha recibido algún dato desde la PC a través del puerto serial.

-6- Carga en el registro EEADR la dirección de la localidad de la memoria EEPROM en la que será almacenado el dato recibido.

-7- Envía el dato recibido de regreso a la PC para que pueda ser observado en la pantalla.

-8- Carga el dato que será grabado en la memoria en el registro EEDATA.

-9- Comienza la secuencia requerida para realizar la escritura de la memoria EEPROM, en donde se involucran el registro de control de la memoria EECON1 y el registro EECON2 que se utiliza para cargar valores especificados por el fabricante.

-10- Una vez terminada la secuencia de escritura incrementa el registro DIREC que se está utilizando para determinar la localidad de memoria que será grabada y decrementa el registro "Límite" que se utiliza para limitar la escritura a las primeras 16 direcciones de la memoria (si no se desea limitar el número de localidades se eliminan las instrucciones relacionadas con el registro "Límite" y la instrucción "goto FIN").

-11- Subrutina para revisar si el dato que fue enviado por el puerto serial hacia la PC ha sido transmitido.

3.5.9. Prueba 9 – Lectura de la memoria EEPROM

La última de las pruebas realizadas es la de lectura de la memoria EEPROM. En esta prueba se leerán los datos que fueron grabados en la memoria en la prueba anterior, para luego mostrarlos en el monitor de la PC por medio del módulo USART y el puerto serial. Existe una subrutina de retardo incluida en el programa debido a que, como la memoria EEPROM es un dispositivo "lento" en comparación con la velocidad de operación del módulo USART, se pueden presentar errores en la transmisión pues la memoria no proporciona los datos con la velocidad con que se transmiten. Por esto, la subrutina de retardo se ejecuta entre cada transmisión para dar tiempo de que sea leída la siguiente localidad de la memoria EEPROM. Este retardo esta hecho para un tiempo de 10ms repetidos 50 veces, es decir, medio segundo. El retardo podría ser menor pero de esta forma el la pantalla da la impresión de que los datos van apareciendo en secuencia como en un anuncio donde se muestran distintos letreros con movimiento.

Como en el programa anterior, que se limitó momentáneamente a 16 localidades de memoria, en esta prueba también va a limitarse a leer las mismas 16 localidades.

```
      list p=16F877A
      INCLUDE "P16F877A.inc"
      Radix HEX
      __CONFIG H'3F39'

      TEMPO1      EQU   0A0   ; -1-
      Limite      EQU   122
      DIREC       EQU   123

;********Sección de macros********

Banco_0     MACRO             ; -2-
            bcf   STATUS,RP0
            bcf   STATUS,RP1
      ENDM

Banco_1     MACRO
            bsf   STATUS,RP0
            bcf   STATUS,RP1
      ENDM

Banco_2     MACRO
            bcf   STATUS,RP0
            bsf   STATUS,RP1
      ENDM
```

```
      Banco_3     MACRO
            bsf   STATUS,RP0
            bsf   STATUS,RP1
      ENDM
;*********************************

      ORG   0
      goto  INICIO
      ORG   5

INICIO      Banco_1           ; -3-
            movlw 19
            movwf SPBRG
            movlw 24
            movwf TXSTA
            Banco_0
            movlw 90
            movwf RCSTA
            Banco_1
            bcf   PIE1,4      ; -4-
            bcf   PIE1,5
            Banco_2
            movlw 0F          ; -5-
            movwf Limite
            clrf  EEDATA
            clrf  DIREC
```

```
DIR         movf  DIREC,w     ; -6-
            movwf EEADR
            Banco_3
            bcf   EECON1,7    ; -7-
            bsf   EECON1,0
            Banco_2
            movf  EEDATA,w    ; -8-
            Banco_0
            movwf TXREG

CHECA       btfss PIR1,4      ; -9-
            goto  CHECA
            Banco_1
            movlw 32          ; -10-
            movwf TEMPO1
            call  DEL10
            Banco_2
            incf  DIREC,1     ; -13-
            decfsz Limite,1
            goto  DIR
FIN         goto  FIN


DEL10       Banco_1           ; -11
            bcf   INTCON,2
            movlw 0xD9
            movwf OPTION_REG

DEL10_1     btfss INTCON,2    ; -12-
            goto  DEL10_1
            decfsz TEMPO1,1
            goto  DEL10
            return

      END
```

-1- Estos son registros creados por el usuario que se ocuparán durante la ejecución del programa.

-2- Crea los macro que se utilizarán para realizar los cambios entre los bancos de memoria.

-3- Configuración de los registros asociados al módulo USART.

-4- Deshabilita las interrupciones.

-5- Carga el registro "Límite" con un valor predeterminado y limpia los registros EEDATA y DIREC para evitar que al momento de encender el sistema puedan contener algún valor no deseado.

-6- Mueve el valor contenido en el registro DIREC al registro EEADR.

-7- Configura el registro EECON1 que es el registro de control de la memoria EEPROM.

-8- El dato leído de la memoria y almacenado en el registro EEDATA es movido para su transmisión por el puerto serial.

-9- Realiza una comprobación para saber si el dato ya fue transmitido.

-10- Carga el registro TEMPO1 con un valor determinado y llama a la subrutina de retardo.

-11- Realiza la subrutina de retardo.

-12- Revisa si ya termino el retardo incluyendo todas las veces que debe de repetirse y al terminar regresa de la subrutina.

-13- Incrementa el registro DIREC, decrementa el registro "Límite" y si éste último no ha llegado a cero regresa a la etiqueta DIR para repetir todo el programa. Si el registro ha llegado a cero el programa termina.

3.6.Programa de control del motor de CD

Este es el programa que realizará las funciones de control del motor de CD desplegando en la pantalla de la PC las distintas opciones con las que se cuentan. El usuario podrá determinar por medio de estas opciones el sentido de giro del motor (rotar a la izquierda o derecha), la cantidad de grados que deberá desplazarse el disco del motor, desde 0° hasta 360° con una precisión de 1° por movimiento (ingresando datos de 3 dígitos), o en caso de haberse presentado algún error al indicar esta instrucciones podrá cancelarse la operación.

Una vez realizada la operación del motor se dará opción a dar por terminado el programa, regresar el motor de forma automática al punto origen para realizar un nuevo movimiento o continuar realizando movimientos desde el último punto en que se halla posicionado el disco del motor.

```
                ;Leopoldo Martín del Campo Ramírez

          ;***Programa Final de Control del Motor de CD***

;Prueba del funcionamiento total del sistema en donde se involucran
;los módulos del Timer0, el Timer1, el modulo USART de comunicación
;serie, la memoria de datos EEPROM y dos de los puertos de 8 bits
;denominados PORTB y PORTC

;Este programa está diseñado para realizar el control de posición
;de un motor de CD, indicando desde una PC, el sentido de giro y
;la cantidad de grados a girar por parte del motor.
;La posición del motor puede determinarse visualmente por medio
;de un disco graduado situado en el eje del motor de CD.

                        list p=16F877A
                        INCLUDE "P16F877A.inc"
                        Radix HEX
                        __CONFIG H'3F39'

;*****************************************************************
;          Sección de creación de MACROS para realizar los
;                          cambios de banco
;*****************************************************************

Banco_0         MACRO
                bcf         STATUS,RP0
                bcf         STATUS,RP1
            ENDM

Banco_1         MACRO
                bsf         STATUS,RP0
                bcf         STATUS,RP1
            ENDM

Banco_2         MACRO
                bcf         STATUS,RP0
                bsf         STATUS,RP1
            ENDM

Banco_3         MACRO
                bsf         STATUS,RP0
                bsf         STATUS,RP1
            ENDM
```

```
;*****************************************************************
;           Declaración de registros de propósito general
;*****************************************************************

                        GRADOSL     EQU   20
                        GRADOSH     EQU   21
                        CENTENA     EQU   22
                        DECENA      EQU   23
                        UNIDAD      EQU   24
                        CONTEO      EQU   25
                        SUMACEN     EQU   26
                        SUMADEC     EQU   27
                        TRES  EQU   28
                        RESTRIN     EQU   29
                        LIM_EE      EQU   120
                        DIR_EE      EQU   121

                        ORG   0
                        goto  INICIO
                        ORG   5

;*****************************************************************
;               Sección de llamadas de configuración
;*****************************************************************

INICIO                  Banco_0
                        call  CONPORTS
                        call  CONUSART
                        call  CONTIMER0
                        call  CONTIMER1
                        call  CONMEM
                        call  CONREG

;*****************************************************************
;                    Sección del PROGRAMA principal
;*****************************************************************

MAIN        Banco_0
            call  TEXTO1      ;Llama la subrutina que muestra
                              ;el TEXTO1
            call  GIRO        ;Llama la subrutina que determina
                              ;hacia donde girará el motor
            call  TEXTO2      ;Llama la subrutina que muestra
                              ;el TEXTO2
            call  POSICION    ;Llama la subrutina que lee desde
                              ;la PC la posición que se le indico
            call  TEXTO3      ;Llama la subrutina que muestra
                              ;el TEXTO3
            call  DECIDE      ;Llama la subrutina que determina
                              ;si se continúa o se cancela
            call  MOVE        ;Activa al motor y detecta el momento
                              ;en que llega a la posición deseada
            call  TEXTO4      ;Llama la subrutina que muestra
                              ;el TEXTO4
            call  OPCIONES    ;Dependiendo de la opción elegida
                              ;ejecuta una acción determinada
```

```
;*****************************************************************
;          Sección de configuración de PUERTOS y MODULOS
;*****************************************************************

CONPORTS          Banco_1
                  bsf   TRISC,0     ;Pin O de PORTC es entrada
                  bcf   TRISC,2     ;Pines 2 y 5 del PORTC son
                  bcf   TRISC,5     ;de salida
                  movlw 0FF         ;El PORTB es por completo de
                  movwf TRISB       ;entrada aunque no se utilice
                  return

CONUSART          Banco_1           ;Transmisión asíncrona
                  movlw 19          ;Vel. de transmisión 9600
                  movwf SPBRG       ;Ancho de palabra de 8 bits
                  movlw 24          ;Sin bit de paridad
                  movwf TXSTA
                  bcf   PIE1,4
                  bcf   PIE1,5
                  Banco_0
                  movlw 90
                  movwf RCSTA
                  Return

CONTIMER0         Banco_1
                  movlw 07          ;Modo de temporizador con un
                  movwf OPTION_REG  ;prescaler de 256 divisiones
                  Banco_0
                  return

CONTIMER1         Banco_1
                  bcf   PIE1,0
                  Banco_0
                  clrf  TMR1L
                  clrf  TMR1H
                  movlw 02          ;Modo de contador con señal
                  movwf T1CON       ;externa en flanco de subida
                  return

CONMEM            Banco_2
                  clrf  EEADR       ;Limpia el contenido de los
                  clrf  EEDATA      ;registros asociados a la
                  Banco_0           ;memoria EEPROM
                  return

CONREG            Banco_2           ;Limpia y configura el contenido
                  movlw 24          ;de los GPR creados para evitar
                  movwf LIM_EE      ;que contengan algún valor falso
                  clrf  DIR_EE      ;al iniciar
                  Banco_0
                  clrf  GRADOSL
                  clrf  GRADOSH
                  clrf  RESTRIN
                  return
```

```
;*****************************************************************
;       Sección de TEXTOS que desplegará el Programa Principal
;*****************************************************************

;******** Muestra el TEXTO1 que pregunta el sentido de giro *********

TEXTO1            Banco_2
                  movlw 5D
                  movwf DIR_EE
                  movlw 00
                  movwf EEADR
                  goto  DATO_EE

;**** Una vez indicado el sentido de giro, se muestra el TEXTO2 *****
;******** que pregunta los grados que el motor deberá girar  ********

TEXTO2            Banco_2
                  movlw 23
                  movwf DIR_EE
                  movlw 5D
                  movwf EEADR
                  goto  DATO_EE

;***** El TEXTO3 pregunta si el dato indicado es correcto  ******
;****** para iniciar o si se desea cancelar la operación ********

TEXTO3            Banco_2
                  movlw 1A
                  movwf DIR_EE
                  movlw 80
                  movwf EEADR
                  goto  DATO_EE

;*********  Cuando el motor alcanzó la posición deseada se  *********
;**********  muestran tres opciones para elegir que hacer  **********

TEXTO4            Banco_2
                  movlw 62
                  movwf DIR_EE
                  movlw 9A
                  movwf EEADR
                  goto  DATO_EE

;********* Una vez elegida una opción de las mostradas en el ********
;********* TEXTO3, si se seleccionó la de salir se dará por  ********
;********              terminado el programa                *********

TEXTO5            Banco_2
                  movlw 04
                  movwf DIR_EE
                  movlw 0FC
                  movwf EEADR
                  goto  DATO_EE
```

```
;*****************************************************************
;     Sección de OPERACIONES que realizará el Programa Principal
;*****************************************************************


;********  En esta parte se realiza la lectura y el manejo  *********
;********   de los datos almacenados en la memoria EEPROM   *********

DATO_EE           call  READ

VERIFICA1         movf  EEDATA,w    ;Realiza dos comprobaciones,
                  subwf LIM_EE,w    ;la primera revisa si ya es
                  btfsc STATUS,Z    ;necesario realizar un
                  goto  NEW_LINE    ;cambio de línea en la pantalla
                  movf  EEDATA,w    ;de la PC
                  Banco_0
                  movwf TXREG
                  call  CHECATX

VERIFICA2         Banco_2
                  incf  EEADR,f     ;y la segunda es si ya se han
                  decf  DIR_EE,f    ;leído todas las localidades
                  btfsc STATUS,Z    ;de memoria correspondientes al
                  return            ;texto
                  goto  DATO_EE


;***** Espera a que se le indique el sentido de giro del motor ******
;******   y prepara las señales de control que se requerirán  *******

GIRO              Banco_0
                  btfss PIR1,5
                  goto  GIRO
                  movlw 34
                  subwf RCREG,w
                  btfsc STATUS,Z
                  goto  IZQ
                  movlw 36
                  subwf RCREG,w
                  btfsc STATUS,Z
                  goto  DER
                  goto  GIRO

IZQ               call  ESPERA
                  bcf         PORTC,2
                  return

DER               call  ESPERA
                  bsf         PORTC,2
                  return
```

```
;***** Recibe el dato de la posición, espera la señal de inicio *****
;***** y realiza el cálculo de la posición almacenando el dato  *****

POSICION          Banco_0
                  movlw 64
                  movwf SUMACEN
                  movlw 0A
                  movwf SUMADEC
                  movlw 03
                  movwf TRES
                  clrf  CENTENA
                  clrf  DECENA
                  clrf  UNIDAD

RECIBE            btfss PIR1,5
                  goto  RECIBE
                  movf  RCREG,w
                  movwf RESTRIN
                  movlw 0F0
                  andwf RESTRIN,f
                  movlw 30
                  subwf RESTRIN,f
                  btfss STATUS,Z
                  goto  RECIBE
                  movf  RCREG,w
                  movwf RESTRIN
                  movlw 0F
                  andwf RESTRIN,f
                  bcf   STATUS,C
                  rlf   RESTRIN,f
                  incf  RESTRIN,f
                  incf  RESTRIN,f
                  incf  RESTRIN,f
                  btfsc RESTRIN,3
                  goto  RECIBE
                  movf  DECENA,w
                  movwf CENTENA
                  movf  UNIDAD,w
                  movwf DECENA
                  movf  RCREG,w
                  movwf TXREG
                  call  CHECATX
                  movlw 0F
                  andwf RCREG,w
                  movwf UNIDAD
                  decf  TRES
                  btfss STATUS,Z
                  goto  RECIBE

MULT              movf  UNIDAD,w    ;Realiza la serie de sumas
                  movwf GRADOSL     ;que equivalen a una
                  movf  DECENA,w    ;multiplicación
                  movwf CONTEO
```

```
DEC10             decf  CONTEO
                  btfsc STATUS,Z
                  goto  PASO
                  movlw 0A
                  addwf SUMADEC,f
                  goto  DEC10

PASO              movf  CENTENA,w
                  movwf CONTEO
DEC100            decf  CONTEO
                  btfsc STATUS,Z
                  goto  SUMA
                  movlw 64
                  addwf SUMACEN,f
                  goto  DEC100

SUMA              movf  SUMADEC,w   ;Hechas las suman necesarias
                  addwf GRADOSL,f   ;de las decenas y las unidades
                  movf  SUMACEN,w   ;todo se suma para obtener
                  addwf GRADOSL,f   ;el resultado final
                  btfss STATUS,C
                  goto  NOCARRY
                  goto  CARRY

CARRY             movlw 01          ;En caso de existir el carry
                  movwf GRADOSH     ;el número que se va a guardar
                  call  ESPERA      ;es de 9 bits por lo que
                  movlw 0D          ;involucra al registro GRADOSH
                  movwf TXREG
                  call  CHECATX
                  movlw 0A
                  movwf TXREG
                  call  CHECATX
                  return

NOCARRY           clrf  GRADOSH     ;En caso de no existir el carry
                  call  ESPERA      ;el registro GRADOSH se limpia
                  movlw 0D          ;pues no se va a utilizar
                  movwf TXREG
                  call  CHECATX
                  movlw 0A
                  movwf TXREG
                  call  CHECATX
                  return

;**** El usuario, después de indicar la posición, decide iniciar*****
;****** el movimiento si todo es correcto o en caso contrario *******
;******************  puede cancelar la operación  ******************

DECIDE            Banco_0
                  btfss PIR1,5
                  goto  DECIDE
                  movlw 34
                  subwf RCREG,w
                  btfsc STATUS,Z
                  return
```

```
            movlw 36
            subwf RCREG,w
            btfsc STATUS,Z
            goto  INICIO
            goto  DECIDE
            return

;***** Una vez recibida la posición, activa al motor, espera a  *****
;***** que la alcance y llama la subrutina de frenado del motor *****

MOVE        call  ESPERA
            comf  GRADOSH,w   ;Complementa los datos de los dos
            movwf TMR1H       ;registros y los mueve a los
            comf  GRADOSL,w   ;registros correspondientes
            movwf TMR1L
            incf  TMR1L,1
            bsf   T1CON,0     ;Activa el Timer1 en modo de contador
            bsf   PORTC,5     ;Activa al motor
MOVE_1      btfss PIR1,0      ;Revisa si ya alcanzó la posición
            goto  MOVE_1      ;indicada
            bcf   PIR1,0
            call  FRENO       ;Llama la subrutina de frenado
            return


;****** Ya que se mostraron las 3 opciones del TEXTO3 espera  *******
;************** a que se le indique cuál debe ejecutar **************

OPCIONES    Banco_0
            btfss PIR1,5
            goto  OPCIONES
            movlw 31
            subwf RCREG,w
            btfsc STATUS,Z
            goto  OP1
            movlw 32
            subwf RCREG,w
            btfsc STATUS,Z
            goto  OP2
            movlw 33
            subwf RCREG,w
            btfsc STATUS,Z
            goto  OP3
            goto  OPCIONES

OP1         call  TEXTO5
FIN         goto  FIN

OP2         call  MOVE
            goto  INICIO

OP3         goto  MAIN
```

```
;*****************************************************************
;                  Sección de subrutinas anidadas
;*****************************************************************

READ            Banco_3          ;Indica al programa que debe
                bcf   EECON1,7   ;accesar a la memoria de datos
                bsf   EECON1,0   ;y lee una de las localidades
                Banco_2
                return

NEW_LINE        Banco_0
                movlw 0D         ;Subrutina que realiza el
                movwf TXREG      ;cambio de línea en la pantalla
                call  CHECATX
                movlw 0A
                movwf TXREG
                call  CHECATX
                goto  VERIFICA2

CHECATX         btfss PIR1,4     ;Revisa si la transmisión del
                goto  CHECATX    ;dato fue completada
                return

ESPERA          btfss PIR1,5     ;Espera a recibir una señal de
                goto  ESPERA     ;ENTER desde el teclado para
                movlw 0D         ;iniciar una acción
                subwf RCREG,w
                btfss STATUS,Z
                goto  ESPERA
                return

FRENO           btfsc PORTC,2    ;Revisa cuál era el sentido en el
                goto  FRENO_D    ;que giraba el motor
                goto  FRENO_I
FRENO_D         bcf   PORTC,2    ;Cambia el sentido para frenar y
                goto  FRENO_T    ;va a la subrutina de tiempo
FRENO_I         bsf   PORTC,2    ;Cambia el sentido para frenar y
FRENO_T         bcf   INTCON,2   ;va a la subrutina de tiempo
                movlw 0F8        ;Realiza un pequeño retardo de
                movwf TMR0       ;2ms mientras se aplica la señal

FRENO_T_1       btfss INTCON,2   ;de inversión de polaridad para
                goto  FRENO_T_1  ;frenar al motor
                bcf   PORTC,5
                return

            END
```

3.7. Aplicaciones del sistema

El objetivo inicial de este proyecto ha sido el de crear una base para realizar una actualización de las prácticas del laboratorio de control digital del área de ingeniería electrónica, sin embargo la utilidad del proyecto puede extenderse a problemas más reales que suelen presentarse en la industria. La versatilidad de los microcontroladores aunada a la flexibilidad del diseño permitiría controlar motores de mayor capacidad realizando algunas modificaciones, principalmente sustituyendo los elementos de potencia por otros de mayor capacidad, pues el control que realiza la parte digital cambiaría muy poco. Este es debido a que es muy sencillo retirar el microcontrolador del circuito de control, hacer las modificaciones necesarias en el programa principal y volver a montar el microcontrolador en su sitio. Otra ventaja es la disponibilidad de un puerto de 8 bits que puede utilizarse, como se mencionó anteriormente, para diversos objetivos pero principalmente puede utilizarse para conectar ciertos elementos que sirvan para realizar una evaluación de las parte del sistema total. Todo lo anterior es hablando de forma general.

Dentro de los distintos usos a los que podría destinarse este proyecto pueden mencionarse:

- Control de rotación de antenas de recepción satelital.
- Manejo de brazos robóticos a nivel industrial que utilicen, ya sea motores de CD o motores de pasos.
- Control de movimiento de una banda transportadora.
- Operación de sistemas de vigilancia cuyas cámaras de seguridad necesiten ser rotadas.
- Sistemas de impresión donde el papel o medio imprimible necesite ser colocado en una posición determinada para realizar la operación de impresión.
- Etc.

Conclusiones

Originalmente este proyecto fue concebido como un sistema de control de posición para servomotores por medio de una interfase con la PC el cuál sería implementado como una práctica de laboratorio correspondiente a la materia de Control Digital sin embargo, durante el desarrollo se presentaron situaciones que provocaron que se modificara el enfoque del proyecto, por mencionar algunas están el elevado costo de los servomotores, la carencia de estos en el laboratorio de la facultad de modo que varios equipos de alumnos pudieran realizar sus prácticas de manera simultánea, utilizar en la interfase la menor cantidad de elementos o resaltar la versatilidad de los microcontroladores. Así que para resolver todos estos aspectos se realizaron cambios durante el desarrollo como el cambio a un motor de CD que resulta mucho más económico y del que se disponen de varias unidades en el laboratorio de la facultad, se seleccionaron los microcontroladores PIC de la marca 'Microchip' los cuales no son muy costosos, existe una gran variedad de ellos con distintas características y la documentación al respecto, como son las hojas técnicas, y el software de programación son ofrecidos de manera gratuita por el mismo fabricante.

Como se ha mencionado el la parte final de este trabajo, además de la aplicación original, se puede realizar una variada implementación del proyecto en otros campos prácticos de la industria adaptándolo a la situación indicada. Todo esto da por resultado un proyecto didáctico que demuestra la adaptabilidad de los microcontroladores que puede ser implementado sin demasiadas complicaciones por los alumnos después de haber recibido una explicación general de su funcionamiento.

APÉNDICE A - Glosario

- **µC**. Abreviatura de microcontrolador
- **µP**. Abreviatura de microprocesador
- **ALU**. Siglas de Arithmetic Logic Unit, Unidad Lógico-Aritmética.
- **BRG.** Baud Rate Generator, Generador de la Velocidad de Transmisión.
- **CISC**. Siglas de Complex Instruction Set Computer, Computadora de Juego de Instrucciones Complejo.
- **Control dedicado**. Dispositivo capaz de encenderse o de hacer un reset a sí mismo.
- **Conversor A/D**. Conversor Analógico a Digital.
- **Conversor D/A**. Conversor Digital a Analógico.
- **CPU**. Siglas de Central Processing Unit., Unidad Central de Procesamiento.
- **EEPROM**. Siglas de Electrically Erasable and Programmable ROM, ROM Eléctricamente Borrable y Programable.
- **EPROM**. Siglas de Electrically Programmable ROM, ROM Eléctricamente Programable.
- **Embobinado de armadura**. Consiste de bobinas aisladas, tanto entre sí como del núcleo de la armadura. Estas bobinas están incrustadas en ranuras en el núcleo de la armadura y están conectadas eléctricamente con el conmutador, el cual, debido a la rotación del eje, da los cambios de conexión necesarios para el proceso de conmutación.
- **Embobinado de campo**. Los devanados de campo, que consisten de algunas vueltas o espiras de conductor en el caso de un campo serie, o de muchas vueltas de alambre delgado en el caso de un campo en paralelo o derivación. En esencia, las bobinas de campo son electroimanes cuyos amperes-vuelta proveen la fuerza magnetomotriz adecuada para producir el flujo necesario para generar una fuerza electromotriz.
- **FSR**. Siglas de File Select Register, Registro Selector de Archivos.
- **Full Duplex**. Con el funcionamiento full dúplex puede haber transmisiones en ambas direcciones al mismo tiempo. En ocasiones, a los sistemas full dúplex se les llama simultáneos de dos sentidos. Un sistema telefónico normal es un ejemplo de funcionamiento full dúplex.
- **GPR**. Siglas de General Purpose Register, Registro de Propósito General.
- **Half Duplex**. En este tipo de funcionamiento, las transmisiones se pueden hacer en ambas direcciones, pero no al mismo tiempo. Las líneas half dúplex también se llaman líneas de dos sentidos alternas, o líneas de uno de dos sentidos. La radio de banda civil es un ejemplo de transmisión half dúplex.
- **I²C**. Inter-Integrated Circuit, Circuito de Intercomunicación Integrado.
- **I/O**. Siglas de In/Out, Entradas/Salidas.
- **IRP**. Siglas de Indirect Register Pointer, Indicador de Registros Indirectos.
- **ICSP**. Siglas de In-Circuit Serial Programming, Programación Serial In-Circuit.
- **LCD**. Siglas de Liquid Crystal Display. Pantalla de cristal líquido.
- **LIFO**. Siglas de Last Instruction First Out, la Ultima Instrucción que entra al Stack es la Primera en Salir.
- **LST**. Siglas de Long Scale Integrated, Alta Escala de Integración.

- **Máscara**. Proceso de fabricación de circuitos integrados en el que por medio de una contaminación selectiva de capas de silicio con óxido y la difusión o implantación de iones es posible grabar agujeros de contacto y patrones de disposición de delgadas películas en capas para satisfacer requerimientos de dimensiones, propiedades eléctricas y espesor a fin de obtener el mayor rendimiento.
- **Nibbles**. Conjuntos de cuatro bits.
- **OTP**. Siglas de One Time Programmable, Programable Una Vez.
- **PC**. Siglas de Program Counter, Contador de Programa.
- **PSP**. Siglas de Parallel Slave Port, Puerto Paralelo Esclavo.
- **PWM**. Siglas de Pulse Wide Modulation, Modulación por Ancho de Pulsos.
- **RAM**. Random Access Memory
- **RISC**. Siglas de Reduced Instrucción Set Computer, Computadora de Juego de Instrucciones Reducido.
- **ROM**. Siglas de Read Only Memory, Memoria de Sólo Lectura.
- **RSR.** Siglas de Receive Shift Register, Registro de Corrimiento para la Recepción.
- **SFR**. Siglas de Special Function Register, Registro de Función Especial.
- **SPI**. Siglas de Serial Peripherical Interphase, Interfase Periférica Serial.
- **SSP**. Synchronous Serial Port. Puerto Serial Síncrono.
- **Snack**. Pila. Zona aislada de la memoria.
- **TSR.** Transmit Shift Register, Registro de Corrimiento para la Transmisión.
- **USART**. Siglas de Universal Synchronous Asynchronous Receiver Transmitter, Receptor Transmisor Síncrono Asíncrono Universal.
- **USB**. Siglas de Universal Serial Bus, Bus Serial Universal.

APÉNDICE B - Conjunto de instrucciones del PIC16F877A

Mnemónico, Operandos		Descripción	Ciclos	14-Bit Opcode				Status Afectado	Notas
				MSb			LSB		
OPERACIONES DE REGISTROS ORIENTADAS A BYTES									
ADDWF	f, d	Suma W y f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W con f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Limpia f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Limpia W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complementa f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrementa f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrementa f, Salta si 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Incrementa f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Incrementa f, Salta si 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	OR Inclusiva de W con f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Mueve f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Mueve W a f	1	00	0000	lfff	ffff		
NOP	-	No Operación	1	00	0000	0xx0	0000		
RLF	f, d	Rota a la izquierda f usando el Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rota a la derecha f usando el Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Resta W de f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Intercambia los conjuntos de 4 bit en f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	OR Exclusiva W con f	1	00	0110	dfff	ffff	Z	1,2
OPERACIONES DE REGISTROS ORIENTADAS A BITS									
BCF	f, b	Apaga un Bit de f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Enciende un Bit de f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Prueba un Bit de f, Salta si cero	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Prueba un Bit de f, Salta si uno	1 (2)	01	11bb	bfff	ffff		3
OPERACIONES CON LITERALES Y DE CONTROL									
ADDLW	k	Suma una Literal y W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND una Literal con W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Llama la subrutina.	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Limpia el Watchdog Timer	1	00	0000	0110	0100	TO,PD	
GOTO	k	Ve a la dirección	2	10	1kkk	kkkk	kkkk		
IORLW	k	OR Inclusiva de una Literal con W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Mueve una Literal a W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Regresa de una interrupción	2	00	0000	0000	1001		
RETLW	k	Regresa con una Literal en W	2	11	01xx	kkkk	kkkk		
RETURN	-	Regresa de la Subrutina.	2	00	0000	0000	1000		
SLEEP	-	Ir al modo de espera	1	00	0000	0110	0011	TO,PD	
SUBLW	k	Resta W de la Literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	OR Exclusiva de una Literal con W	1	11	1010	kkkk	kkkk	Z	

Nota 1: Cuando un registro de I/O es modificado como una function del mismo (ejm. MOVF PORTB,1), el valor usado será el valor presente en los pines. Por ejemplo, si el dato del latch es '1' para configurar un pin de entrada y es puesto en bajo por un dispositivo externo, el dato se escribirá como un '0'.

2: Si esta instrucción es ejecutada en el registro del TMR0 (y, donde se aplica d=1), el prescaler será limpiado si es asignado al módulo del Timer0.

3: Si el Contador de Programa (PC) es modificado, o una prueba condicionada es verdadera, la instrucción requiere de dos ciclos. El segundo ciclo es ejecutado como un NOP.

Descripción de las Instrucciones

ADDLW **Suma una Literal y W**

Sintaxis: [*etiqueta*] ADDLW k
Operandos: 0 ≤k ≤255
Operación: (W) + k → (W)
Status Afectado: C, DC, Z
Descripción: El contenido del registro W es sumado a una literal de 8 bits ‘k’ y el resultado es almacenado en el registro W.

ADDWF **Suma W y f**

Sintaxis: [*etiqueta*] ADDWF f, d
Operandos: 0 ≤ f ≤127
d ∈ [0,1]
Operación: (W) + (f) → (destino)
Status Afectado: C, DC, Z
Descripción: Suma el contenido del registro W con l registro ‘f’. Si ‘d’ es 0, el resultado es almacenado en el registro W. Si ’d’ es 1, el resultado es almacenado en el registro ’f’.

ANDLW **AND una Literal con W**

Sintaxis: [*etiqueta*] ANDLW k
Operandos: 0 ≤k ≤255
Operación: (W) .AND. (k) → (W)
Status Afectado: Z
Descripción: Se realiza una AND lógica del contenido de W con una literal de 8 bits ‘k’. El resultado es almacenado en el registro W.

ANDWF **AND W con f**

Sintaxis: [*etiqueta*] ANDWF f, d
Operandos: 0 ≤f ≤127
d ∈ [0,1]
Operación: (W) .AND. (f) → (destino)
Status Afectado: Z
Descripción: Se realize una AND lógica del registro W con el registro 'f'. Si 'd' is 0, el resultado es almacenado en el registro W. Si 'd' is 1, el resultado es almacenado en el registro 'f'.

BCF **Apaga un Bit de f**

Sintaxis: [*etiqueta*] BCF f, b
Operandos: 0 ≤f ≤127 0 ≤b ≤7
Operación: 0 → (f<b>)
Status Afectado: Ninguno
Descripción: El Bit 'b' en el registro 'f' es puesto en cero (apagado).

BSF **Enciende un Bit de f**

Sintaxis: [*etiqueta*] BSF f, b
Operandos: 0 ≤f ≤127 0 ≤b ≤7
Operación: 1 → (f<b>)
Status Afectado: Ninguno
Descripción: El Bit 'b' en el registro 'f' es puesto en uno (encendido).

BTFSS **Prueba un Bit de f, Salta si Uno**

Sintaxis: [*etiqueta*] BTFSS f, b
Operandos: 0 ≤f ≤127 0 ≤b < 7
Operación: skip if (f<b>) = 1
Status Afectado: Ninguno
Descripción: Si el bit 'b' en el registro 'f' is '0', la siguiente instrucción es ejecutada. Si el bit 'b' is '1', entonces la siguiente instrucción es ignorada y un NOP es ejecutado en su lugar, haciendo a esta una instrucción de 2TCY.

BTFSC **Prueba un bit de f, Salta si cero**

Sintaxis: [*etiqueta*] BTFSC f, b
Operandos: 0 ≤f ≤127 0 ≤b ≤7
Operación: skip if (f<b>) = 0
Status Afectado: Ninguno
Descripción: Si el bit 'b' en el registro 'f' es '1', la siguiente instrucción es ejecutada. Si el bit 'b' es '0', entonces la siguiente instrucción es ignorada y un NOP es ejecutado en su lugar, haciendo a esta una instrucción de 2TCY.

CALL	**Llama la Subrutina**
Sintaxis:	[*etiqueta*] CALL k
Operandos:	0 ≤k ≤2047
Operación:	(PC) + 1→TOS, k →PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Afectado:	Ninguno
Descripción:	Llama la subrutina. Primero, la dirección de regreso (PC+1) es empujada dentro del stack. Los once bits inmediatos de la dirección son cargados en los bits del PC <10:0>. Los bits más altos del PC se cargan desde PCLATH. CALL es una instrucción de dos ciclos.

CLRF	**Limpia f**
Sintaxis:	[*etiqueta*] CLRF f
Operandos:	0 ≤f ≤127
Operación:	00h → (f) 1 →Z
Status Afectado:	Z
Descripción:	El contenido del registro ‘f’ es limpiado (puesto en cero) y el bit Z es encendido.

CLRW	**Limpia W**
Sintaxis:	[*etiqueta*] CLRW
Operandos:	Ninguno
Operación:	00h → (W) 1 →Z
Status Afectado:	Z
Descripción:	El registro W es limpiado (puesto en cero). El bit cero (Z) es encendido.

CLRWDT	**Limpia Watchdog Timer**
Sintaxis:	[*etiqueta*] CLRWDT
Operandos:	Ninguno
Operación:	00h →WDT 0 →WDT prescaler, 1 →TO 1 →PD TO, PD
Status Afectado:	
Descripción:	La instrucción CLRWDT da un reset al Watchdog Timer. Esto también da un reset al prescaler del WDT. Los bits del registro Status TO and PD se encienden.

COMF	**Complementa f**
Sintaxis:	[*etiqueta*] COMF f, d
Operandos:	0 ≤f ≤127 d ∈ [0,1]
Operación:	(f) → (destino)
Status Afectado:	Z
Descripción:	El contenido del registro ‘f’ es complementado. Si ’d’ es 0, el resultado es almacenado en W. Si ’d’ s 1, el resultado es almacenado en el registro ’f’.

DECF	**Decrementa f**
Sintaxis:	[*etiqueta*] DECF f, d
Operandos:	0 ≤f ≤127 d ∈ [0, 1]
Operación:	(f) - 1 → (destino)
Status Afectado:	Z
Descripción:	Decrementa el registro ’f’. Si ’d’ es 0 el resultado es almacenado en el registro W. Si ’d’ es 1, el resultado es almacenado en el registro ’f’.

DECFSZ	**Decrementa f, Salta si Cero**
Sintaxis:	[*etiqueta*] DECFSZ f, d
Operandos:	0 ≤f ≤127 d ∈[0,1]
Operación:	(f) - 1 → (destino); salta si el resultado = 0
Status Afectado:	Ninguno
Descripción:	El contenido del registro ‘f’ es decrementado. Si ’d’ es 0, el resultado es guardado en el registro W. Si ’d’ es 1, el resultado es guardado en el registro ’f’. Si el resultado es 1, la siguiente instrucción es ejecutada. Si el resultado es 0, entonces un NOP es ejecutado en su lugar, haciendo esta una instrucción de 2TCY.

GOTO	**Rama no condicionada**
Sintaxis:	[*etiqueta*] GOTO k
Operandos:	0 ≤k ≤2047
Operación:	k →PC<10:0> PCLATH<4:3> → PC<12:11>
Status Afectado:	Ninguno
Descripción:	GOTO es una rama no condicionada. Los once bits inmediatos de la dirección son cargados en los bits del PC <10:0>. Los bits más altos del PC se cargan desde PCLATH<4:3>. GOTO es una instrucción de dos ciclos.

INCF	**Incrementa f**
Sintaxis:	[*etiqueta*] INCF f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	(f) + 1 → (destino)
Status Afectado:	Z
Descripción:	El contenido del registro ’f’ es incrementado. Si ’d’ es 0, el resultado es guardado en el registro W. Si ’d’ es 1, el resultado es guardado en el registro ’f’.

INCFSZ	**Incrementa f, Salta si Cero**
Sintaxis:	[*etiqueta*] INCFSZ f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0, 1]$
Operación:	(f) + 1 → (destino), salta si el resultado = 0
Status Afectado:	Ninguno
Descripción:	El contenido del registro ’f’ es incrementado. Si ’d’ es 0, the result is placed in the W register. If ’d’ is 1, the result is placed back in register ’f’. If the result is 1, the next instruction is executed. If the result is 0, a NOP is executed instead, making it a 2TCY instruction.

IORLW	**OR Inclusiva de una Literal con W**
Sintaxis:	[*etiqueta*] IORLW k
Operandos:	$0 \leq k \leq 255$
Operación:	(W) .OR. k →(W)
Status Afectado:	Z
Descripción:	Se aplica una OR lógica al contenido del registro W con la literal de 8 bits ‘k’. El resultado es guardado en el registro W.

IORWF	**OR Inclusiva de W con f**
Sintaxis:	[*etiqueta*] IORWF f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	(W) .OR. (f) → (destino)
Status Afectado:	Z
Descripción:	Se aplica una OR lógica al registro W con el registro 'f'. Si 'd' es 0 el resultado es guardado en el registro W. Si 'd' es 1 el resultado es guardado en el registro 'f'.

MOVF	**Mueve f**
Sintaxis:	[*etiqueta*] MOVF f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0, 1]$
Operación:	(f) → (destino)
Status Afectado:	Z
Descripción:	El contenido del registro ‘f’ es movido al destino dependiendo del status de ‘d’. Si d = 0, el destino es el registro W. Si d = 1, el destino es el registro ‘f’ mismo. d = 1 es usado para probar registros, usando el bit de bandera Z del registro STATUS.

MOVLW	**Mueve una Literal a W**
Sintaxis:	[*etiqueta*] MOVLW k
Operandos:	$0 \leq k \leq 255$
Operación:	k → (W)
Status Afectado:	Ninguno
Descripción:	La literal de 8 bits ‘k’ es cargada en el registro W. Los no importa serán tomados como ceros.

MOVWF	**Mueve W a f**
Sintaxis:	[*etiqueta*] MOVWF f
Operandos:	$0 \leq f \leq 127$
Operación:	(W) → (f)
Status Afectado:	Ninguno
Descripción:	Mueve el dato desde el registro W al registro ‘f’.

NOP	**No Operación**
Sintaxis:	[*etiqueta*] NOP
Operandos:	Ninguno
Operación:	No Operación
Status Afectado:	Ninguno
Descripción:	No se realiza ninguna operación durante un ciclo.

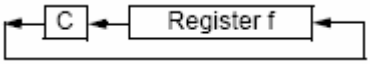
RETFIE	**Regresa de la Interrupción**
Sintaxis:	[*etiqueta*] RETFIE
Operandos:	Ninguno
Operación:	TOS →PC, 1 →GIE
Status Afectado:	Ninguno

RETLW **Regresa con una Literal en W**

Sintaxis: [*etiqueta*] RETLW k
Operandos: $0 \leq k \leq 255$
Operación: $k \rightarrow (W)$; TOS $\rightarrow$PC
Status Afectado: Ninguno
Descripción: El registro W es cargado con la literas de 8 bits ‘k’. El contador de programa es cargado con la dirección que está en la parte superior del stack (la dirección de regreso). Esta es una instrucción de dos ciclos.

RLF **Rota a la Izquierda usando el Carry**

Sintaxis: [*etiqueta*] RLF f, d
Operandos: $0 \leq f \leq 127$ d $\in$[0,1]
Operación: Ver descripción abajo
Status Afectado: C
Descripción: El contenido del registro ‘f’ es rotado un bit a la izquierda usando la bandera del carry. Si ’d’ es 0, el resultado es guardado en el registro W. Si ’d’ es 1, el resultadi es guardado en el registro ‘f’.

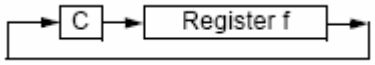


RETURN **Regresa de la Subrutina**

Sintaxis: [*etiqueta*] RETURN
Operandos: Ninguno
Operación: TOS $\rightarrow$PC
Status Afectado: Ninguno
Descripción: Regresa de la subrutina. El snack recibe un POP y la parte superior del snack es cargada en el contador de programa. Esta es una instrucción de dos ciclos.

RRF **Rota a la Derecha usando el Carry**

Sintaxis: [*etiqueta*] RRF f, d
Operandos: $0 \leq f \leq 127$ d $\in$[0, 1]
Operación: Ver descripción abajo
Status Afectado: C
Descripción: El contenido del registro ‘f’ es rotado un bit a la derecha usando la bandera del Carry. Si ’d’ es 0, el resultado es guardado en el registro W. Si ’d’ es 1, el resultado es guardado en el registro ‘f’.



SLEEP **Ir al modo de espera**

Sintaxis: [*etiqueta*] SLEEP
Operandos: Ninguno
Operación: 00h $\rightarrow$ WDT,
0 $\rightarrow$WDT prescaler,
1 $\rightarrow$TO,
0 $\rightarrow$PD
Status Afectado: TO, PD
Descripción: El bit de apagado de registro STATUS, PD es apagado (puesto en ‘0’). El bit TO del registro STATUS es encendido (puesto en’1’). El Watchdog Timer y su prescaler son limpiados. El procesador es puesto en el modo de “espera” con el oscilador detenido.

SUBLW **Resta W de la Literal**

Sintaxis: [*etiqueta*] SUBLW k
Operandos: $0 \leq k \leq 255$
Operación: k - (W) $\rightarrow$ (W)
Status Afectado: C, DC, Z
Descripción: El registro W es restado (método de complemento a 2) de la literal de 8 bits ‘k’. El resultado es guardado en el registro W.

SUBWF **Resta W de f**

Sintaxis: [*etiqueta*] SUBWF f, d
Operandos: $0 \leq f \leq 127$ d $\in$[0,1]
Operación: (f) - (W) $\rightarrow$ (destino)
Status Afectado: C, DC, Z
Descripción: Resta (método de complemento a 2) el registro W del registro ‘f’'. Si 'd' es 0, el resultado es almacenado en el registro W. Si 'd' es 1, el resultado es almacenado en el registro 'f'.

SWAPF	**Intercambia los nibbles en f**
Sintaxis:	[*etiqueta*] SWAPF f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	(f<3:0>) → (destino<7:4>), (f<7:4>) → (destino<3:0>)
Status Afectado:	Ninguno
Descripción:	Los nibbles superior e inferior del registro ‘f’ son intercambiados. Si ‘d’ es 0, el resultado es guardado en el registro W. Si ’d’ es 1, el resultado es guardado en el registro ‘f’.

XORLW	**OR Exclusiva de una Literal con W**
Sintaxis:	[*etiqueta*] XORLW k
Operandos:	$0 \leq k \leq 255$
Operación:	(W) .XOR. k → (W)
Status Afectado:	Z
Descripción:	Se aplica una XOR lógica al contenido del registro W con la literal de 8 bits ‘k’. El resultado es guardado en el registro W.

XORWF	**OR Exclusiva de W con f**
Sintaxis:	[*etiqueta*] XORWF f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0, 1]$
Operación:	(W) .XOR. (f) → (destino)
Status Afectado:	Z
Descripción:	Se aplica una OR Exclusiva lógica al contenido del registro W con el rgistro ‘f’. Si 'd' es 0, el resultado es almacenado en el registro W. Si 'd' es 1, el resultado es almacenado en el registro ‘f’.

APÉNDICE C – Códigos ASCII

Tabla ASCII – Formato de caracteres estándares

ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
0	0	NUL	16	10	DLE	32	20	(espacio)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?
ASCII	**Hex**	**Símbolo**	**ASCII**	**Hex**	**Símbolo**	**ASCII**	**Hex**	**Símbolo**	**ASCII**	**Hex**	**Símbolo**
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	□

Tabla ASCII – Códigos de control

ASCII	Hex	Símbolo	Tipo	Descripción
0	0	NUL		Null – nulo
1	1	SOH	CC	Start of Heading - inicio de cabecera
2	2	STX	CC	Start of Text - inicio de texto
3	3	ETX	CC	End of Text - fin de texto
4	4	EOT	CC	End of Transmission - fin de transmisión
5	5	ENQ	CC	Enquiry - solicitud de información
6	6	ACK	CC	Acknowledge – confirmación
7	7	BEL		Bell - señal audible / tono de atención
8	8	BS	FE	Backspace – retroceso
9	9	TAB	FE	Horizontal Tabulation - tabulado horizontal
10	A	LF	FE	Line Feed - avance de línea
11	B	VT	FE	Vertical Tabulation - tabulado vertical
12	C	FF	FE	Form Feed - avance de página
13	D	CR	FE	Carriage Return - Retorno de carro, iniciar nueva linea
14	E	SO		Shift Out - terminar modo mayúsculas
15	F	SI		Shift In - iniciar modo mayúsculas
16	10	DLE	CC	Data Link Escape – escape del enlace de datos
17	11	DC1		Device Control 1 - control de dispositivo 1
18	12	DC2		Device Control 2 - control de dispositivo 2
19	13	DC3		Device Control 3 - control de dispositivo 3
20	14	DC4		Device Control 4 - control de dispositivo 4
21	15	NAK	CC	Negative Acknowledge - confirmación negativa
22	16	SYN	CC	Synchronous Idle - sincronización de la comunicación
23	17	ETB	CC	End of Transmission Block - fin de bloque de transmisión
24	18	CAN		Cancel – cancelar
25	19	EM		End of Medium - fin del medio (cinta/disco/papel)
26	1A	SUB		Substitute – reemplazar
27	1B	ESC		Escape
28	1C	FS	IS	File Separator - separador de archivos
29	1D	GS	IS	Group Separator - separador de grupo
30	1E	RS	IS	Record Separator - separador de registro
31	1F	US	IS	Unit Separator - separador de unidad

Tipo	Descripción
CC	Communication Control – control de comunicación
FE	Format Effector - manipulador de formato
IS	Information Separator - separador de información

APÉNDICE D – Bibliografía y Directorio de Internet

Bibliografía

~ Microcontroladores PIC -Diseño práctico de aplicaciones-
Angulo Usastegui, José Ma.
Angulo Martínez,Ignacio
3º Edición
Edit. McGraw Hill

~ PIC16F87XA Data Sheet
Microchip Technology Inc.

~ PIC micro Mid-Range MCU Family Reference Manual
Microchip Technology Inc.

~ Máquinas Eléctricas Rotativas y Transformadores
Richardson, Donald V.
Caisse, Arthur J.
4º Edición
Edit. Prentice Hall

~ Máquinas eléctricas y transformadores
Kosow, Irving L.
2º Edición
Edit. Prentice Hall

~ Máquinas electromecánicas y electromagnéticas
Matsch, Leander W.
Edit. Alfaomega

~ Motores eléctricos
Ferrer, Ricardo.
7º Edición
Juan Broger Editor.

~ Diseño y tecnología de circuitos integrados
Morant, Martin J.
Edit. Addison-Wesley Iberoamericana

Directorio de Internet

- Creaturoides - Robots Animatronics -
  http://www.creaturoides.com/puentesh.htm

- ETS Universidad de Granada
  http://www-etsi2.ugr.es/alumnos/mlii/

- IC-Prog Software
  http://www.ic-prog.com

- Microchip Inc.
  http://www.microchip.com

- Electrónica, Programación y Seguridad
  http://www.pablin.com.ar/

- Sfriswolker's Homepeich
  http://usuarios.lycos.es/sfriswolker/

- Webopedia: Online Computer Dictionary for Computer and Internet Terms and Definitio
  http://www.webopedia.com

FAIRCHILD SEMICONDUCTOR™

September 1983
Revised February 1999

MM74HC14
Hex Inverting Schmitt Trigger

General Description

The MM74HC14 utilizes advanced silicon-gate CMOS technology to achieve the low power dissipation and high noise immunity of standard CMOS, as well as the capability to drive 10 LS-TTL loads.

The 74HC logic family is functionally and pinout compatible with the standard 74LS logic family. All inputs are protected from damage due to static discharge by internal diode clamps to V_{CC} and ground.

Features

- Typical propagation delay: 13 ns
- Wide power supply range: 2–6V
- Low quiescent current: 20 μA maximum (74HC Series)
- Low input current: 1 μA maximum
- Fanout of 10 LS-TTL loads
- Typical hysteresis voltage: 0.9V at V_{CC} = 4.5V

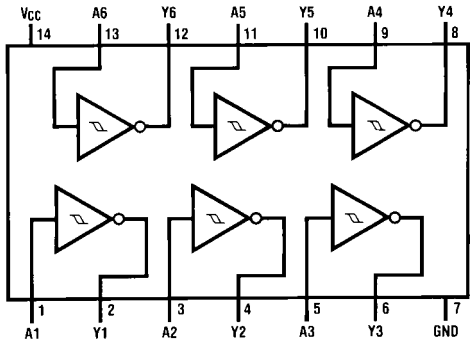
Ordering Code:

Order Number	Package Number	Package Description
MM74HC14M	M14A	14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-120, 0.150" Narrow
MM74HC14SJ	M14D	14-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
MM74HC14MTC	MTC14	14-Lead Thin Shrink Small Outline Package (TSSOP), JEDEC MO-153, 4.4mm Wide
MM74HC14N	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

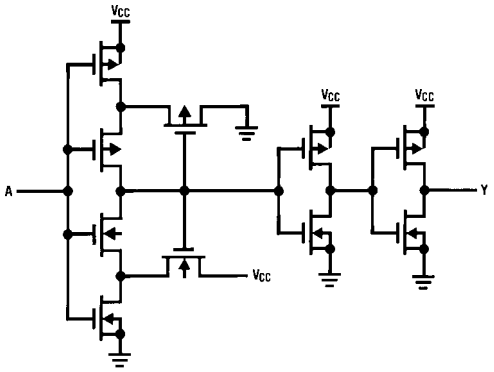
Connection Diagram

Pin Assignments for DIP, SOIC, SOP and TSSOP



Top View

Logic Diagram



Absolute Maximum Ratings (Note 1)

(Note 2)

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−1.5 to V_{CC} +1.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC} +0.5V
Clamp Diode Current (I_{IK}, I_{OK})	±20 mA
DC Output Current, per pin (I_{OUT})	±25 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±50 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D)	
(Note 3)	600 mW
S.O. Package only	500 mW
Lead Temperature (T_L)	
(Soldering 10 seconds)	260°C

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage (V_{CC})	2	6	V
DC Input or Output Voltage (V_{IN}, V_{OUT})	0	V_{CC}	V
Operating Temperature Range (T_A)	−40	+85	°C

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: −12 mW/°C from 65°C to 85°C.

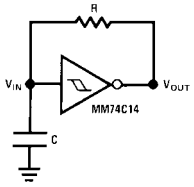
DC Electrical Characteristics (Note 4)

Symbol	Parameter	Conditions	V_{CC}	T_A = 25°C	T_A = −40 to 85°C	T_A = −55 to 125°C	Units
				Typ	Guaranteed Limits		
V_{T+}	Positive Going	Minimum	2.0V	1.2	1.0	1.0	1.0	V
	Threshold Voltage		4.5V	2.7	2.0	2.0	2.0	V
			6.0V	3.2	3.0	3.0	3.0	V
		Maximum	2.0V	1.2	1.5	1.5	1.5	V
			4.5V	2.7	3.15	3.15	3.15	V
			6.0V	3.2	4.2	4.2	4.2	V
V_{T-}	Negative Going	Minimum	2.0V	0.7	0.3	0.3	0.3	V
	Threshold Voltage		4.5V	1.8	0.9	0.9	0.9	V
			6.0V	2.2	1.2	1.2	1.2	V
		Maximum	2.0V	0.7	1.0	1.0	1.0	V
			4.5V	1.8	2.2	2.2	2.2	V
			6.0V	2.2	3.0	3.0	3.0	V
V_H	Hysteresis Voltage	Minimum	2.0V	0.5	0.2	0.2	0.2	V
			4.5V	0.9	0.4	0.4	0.4	V
			6.0V	1.0	0.5	0.5	0.5	V
		Maximum	2.0V	0.5	1.0	1.0	1.0	V
			4.5V	0.9	1.4	1.4	1.4	V
			6.0V	1.0	1.5	1.5	1.5	V
V_{OH}	Minimum HIGH Level	V_{IN} = V_{IL}	2.0V	2.0	1.9	1.9	1.9	V
	Output Voltage	$\|I_{OUT}\|$ = 20 μA	4.5V	4.5	4.4	4.4	4.4	V
			6.0V	6.0	5.9	5.9	5.9	V
		V_{IN} = V_{IL}						
		$\|I_{OUT}\|$ = 4.0 mA	4.5V	4.2	3.98	3.84	3.7	V
		$\|I_{OUT}\|$ = 5.2 mA	6.0V	5.7	5.48	5.34	5.2	V
V_{OL}	Maximum LOW Level	V_{IN} = V_{IH}	2.0V	0	0.1	0.1	0.1	V
	Output Voltage	$\|I_{OUT}\|$ = 20 μA	4.5V	0	0.1	0.1	0.1	V
			6.0V	0	0.1	0.1	0.1	V
		V_{IN} = V_{IH}						
		$\|I_{OUT}\|$ = 4.0 mA	4.5V	0.2	0.26	0.33	0.4	V
		$\|I_{OUT}\|$ = 5.2 mA	6.0V	0.2	0.26	0.33	0.4	V
I_{IN}	Maximum Input Current	V_{IN} = V_{CC} or GND	6.0V		±0.1	±1.0	±1.0	μA
I_{CC}	Maximum Quiescent	V_{IN} = V_{CC} or GND	6.0V		2.0	20	40	μA
	Supply Current	I_{OUT} = 0 μA						

Note 4: For a power supply of 5V ±10% the worst case output voltages (V_{OH}, and V_{OL}) occur for HC at 4.5V. Thus the 4.5V values should be used when designing with this supply. Worst case V_{IH} and V_{IL} occur at V_{CC} = 5.5V and 4.5V respectively. (The V_{IH} value at 5.5V is 3.85V.) The worst case leakage current (I_{IN}, I_{CC}, and I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.

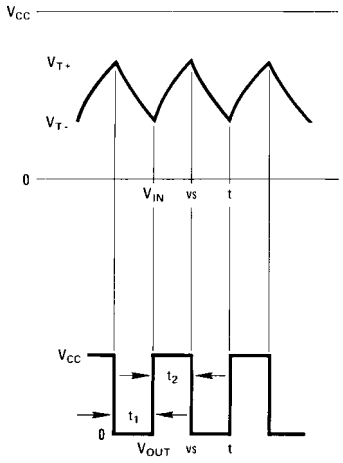
Typical Applications

Low Power Oscillator



$$t_1 \approx RC \ln \frac{V_{T+}}{V_{T-}}$$

$$t_2 \approx RC \ln \frac{V_{CC} - V_{T-}}{V_{CC} - V_{T+}}$$



$$f \approx \frac{1}{RC \ln \frac{V_{T+}(V_{CC} - V_{T-})}{V_{T-}(V_{CC} - V_{T+})}}$$

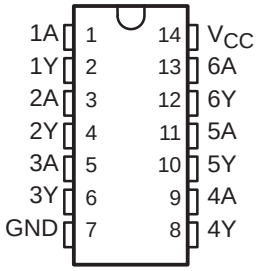
Note: The equations assume $t_1 + t_2 >> t_{pd0} + t_{pd1}$

- **Dependable Texas Instruments Quality and Reliability**

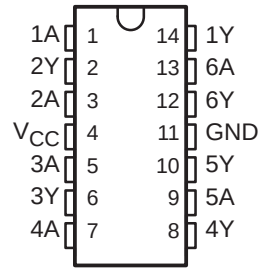
description/ordering information

These devices contain six independent inverters.

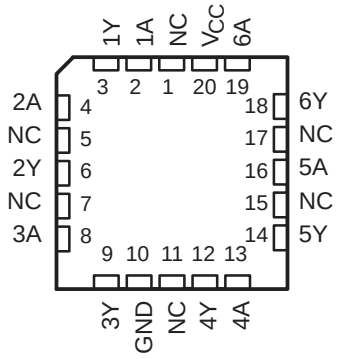
SN5404 . . . J PACKAGE
SN54LS04, SN54S04 . . . J OR W PACKAGE
SN7404, SN74S04 . . . D, N, OR NS PACKAGE
SN74LS04 . . . D, DB, N, OR NS PACKAGE
(TOP VIEW)



SN5404 . . . W PACKAGE
(TOP VIEW)



SN54LS04, SN54S04 . . . FK PACKAGE
(TOP VIEW)



NC – No internal connection

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP – N	Tube	SN7404N	SN7404N
		Tube	SN74LS04N	SN74LS04N
		Tube	SN74S04N	SN74S04N
	SOIC – D	Tube	SN7404D	7404
		Tape and reel	SN7404DR	
		Tube	SN74LS04D	LS04
		Tape and reel	SN74LS04DR	
		Tube	SN74S04D	S04
		Tape and reel	SN74S04DR	
	SOP – NS	Tape and reel	SN7404NSR	SN7404
		Tape and reel	SN74LS04NSR	74LS04
		Tape and reel	SN74S04NSR	74S04
	SSOP – DB	Tape and reel	SN74LS04DBR	LS04
–55°C to 125°C	CDIP – J	Tube	SN5404J	SN5404J
		Tube	SNJ5404J	SNJ5404J
		Tube	SN54LS04J	SN54LS04J
		Tube	SN54S04J	SN54S04J
		Tube	SNJ54LS04J	SNJ54LS04J
		Tube	SNJ54S04J	SNJ54S04J
	CFP – W	Tube	SNJ5404W	SNJ5404W
		Tube	SNJ54LS04W	SNJ54LS04W
		Tube	SNJ54S04W	SNJ54S04W
	LCCC – FK	Tube	SNJ54LS04FK	SNJ54LS04FK
		Tube	SNJ54S04FK	SNJ54S04FK

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

FUNCTION TABLE
(each inverter)

INPUT A	OUTPUT Y
H	L
L	H

bsolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage, V_{CC} (see Note 1) . . . . . 7 V
Input voltage, V_I: ’04, ’S04 . . . . . 5.5 V
’LS04 . . . . . 7 V
Package thermal impedance, θ_{JA} (see Note 2): D package . . . . . 86°C/W
DB package . . . . . 96°C/W
N package . . . . . 80°C/W
NS package . . . . . 76°C/W
Storage temperature range, T_{stg} . . . . . −65°C to 150°C

† Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. This are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. Voltage values are with respect to network ground terminal.
2. The package thermal impedance is calculated in accordance with JESD 51-7.

recommended operating conditions (see Note 3)

		SN5404			SN7404			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High-level input voltage	2			2			V
V_{IL}	Low-level input voltage			0.8			0.8	V
I_{OH}	High-level output current			−0.4			−0.4	mA
I_{OL}	Low-level output current			16			16	mA
T_A	Operating free-air temperature	−55		125	0		70	°C

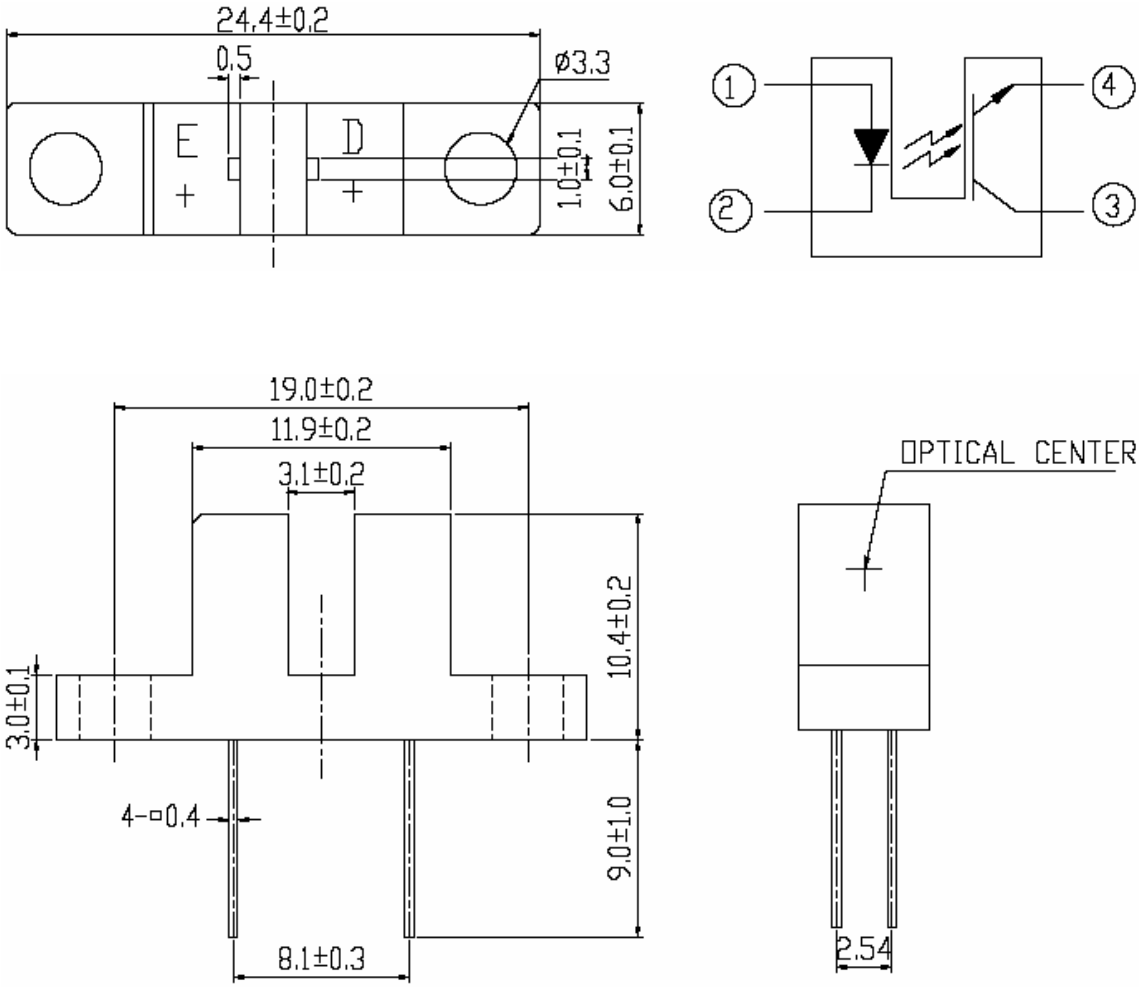
NOTE 3: All unused inputs of the device must be held at V_{CC} or GND to ensure proper device operation. Refer to the TI application report, *Implications of Slow or Floating CMOS Inputs*, literature number SCBA004.

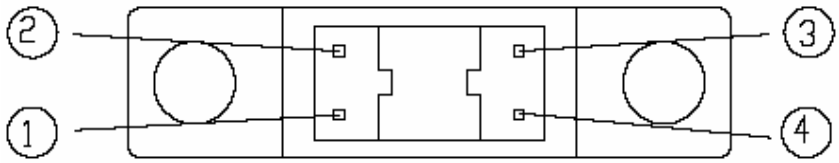
electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS‡			SN5404			SN7404			UNIT
				MIN	TYP§	MAX	MIN	TYP§	MAX	
V_{IK}	V_{CC} = MIN,	I_I = −12 mA				−1.5			−1.5	V
V_{OH}	V_{CC} = MIN,	V_{IL} = 0.8 V,	I_{OH} = −0.4 mA	2.4	3.4		2.4	3.4		V
V_{OL}	V_{CC} = MIN,	V_{IH} = 2 V,	I_{OL} = 16 mA		0.2	0.4		0.2	0.4	V
I_I	V_{CC} = MAX,	V_I = 5.5 V				1			1	mA
I_{IH}	V_{CC} = MAX,	V_I = 2.4 V				40			40	µA
I_{IL}	V_{CC} = MAX,	V_I = 0.4 V				−1.6			−1.6	mA
I_{OS}¶	V_{CC} = MAX			−20		−55	−18		−55	mA
I_{CCH}	V_{CC} = MAX,	V_I = 0 V			6	12		6	12	mA
I_{CCL}	V_{CC} = MAX,	V_I = 4.5 V			18	33		18	33	mA

‡ For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
§ All typical values are at V_{CC} = 5 V, T_A = 25°C.
¶ Not more than one output should be shorted at a time.

- Package Dimensions :





(1) Anode
(2) Cathode
(3) Collector
(4) Emitter

*Notes:

1. All dimensions are in millimeter.
2. General Tolerance: ±0.2mm
3. Lead spacing is measured where the lead emerge from the package.
4. Above specification may be changed without notice. EVERLIGHT will reserve authority on material change for above specification.
5. When using this product , please observe the absolute maximum ratings and the instructions for use outlined in these specification sheets. EVERLIGHT assumes no responsibility for any damage resulting from use of the product which does not comply with the absolute maximum ratings and the instructions included in these specification sheets.

■ Descriptions:

The ITR8102(Slot Optical Switch) is a gallium arsenide infrared emitting diode which is coupled with
a silicon photo transistor in a plastic housing. The packaging system is designed to optimizes the
mechanical resolution, coupling efficiency, and insulates ambient light. The slot in the housing a
provides a means of interrupting the signal with printer, scanner, cop ier, or other opaque material,
switching the output from an ”ON” to ”OFF” state.

■ Features:

- Wide gap between light emitter and detector(3.1mm)
- High sensing accuracy
- PWB mounting type package Pb free

■ Applications:

- Copier
- Printer
- Facsimile
- Ticket vending machine
- Opto-electronic switch

■ Absolute Maximum Ratings (Ta=25 ºC)

Parameter		Symbol	Ratings	Unit
Input	Power Dissipation at(or below) 25ºC Free Air Temperature	Pd	75	mW
	Reverse Voltage	VR	5	V
	Forward Current	I F	50	mA
	Peak Forward Current Pulse width. 100µs, Duty cycle=1%	IFP	1	A
Output	Collector Power Dissipation	PC	75	mW
	Collector Current	I C	20	mA
	Collector-Emitter Voltage	VCEO	30	V
	Emitter-Collector Voltage	VECO	5	V
Operating Temperature		Topr	-25~+85	ºC
Storage Temperature		Tstg	-40~+85	ºC
Lead Soldering Temperature (1/16 inch from body for 5 seconds)		Tsol	260	ºC

■ Electro-Optical Characteristics (Ta=25ºC)

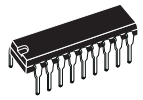
Parameter		Symbol	Min.	Typ.	Max.	Unit	Condition
Input	Forward Voltage	V_F	-	1.2	1.6	V	I_F=20mA
	Reverse Current	I_R	-	-	10	µA	V_R=5V
	Peak Wavelength	λ_P	-	940	-	nm	I_F=20mA
	View Angle	2 1/2	-	60	-	Deg	I_F=20mA
Output	Collector Dark Current	I_{CEO}	-	-	100	nA	V_{CE}=10V
Transfer Characteristic	C-E Saturation Voltage	V_{CE}(sat)	-	-	0.4	V	I_C=0.5mA I_F=20mA
	Collector Curren	$I_{C(ON)}$	0.9	4	15	mA	V_{CE}=5V I_F=20mA
	Rise time	t_r	-	20	-	µsec	V_{CE}=5V I_C=1mA R_L=1K.
	Fall time	t_f	-	20	-	µsec	

L6201
L6202 - L6203

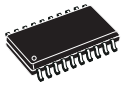
DMOS FULL BRIDGE DRIVER

- SUPPLY VOLTAGE UP TO 48V
- 5A MAX PEAK CURRENT (2A max. for L6201)
- TOTAL RMS CURRENT UP TO
  L6201: 1A; L6202: 1.5A; L6203/L6201PS: 4A
- $R_{DS\ (ON)}$ 0.3 Ω (typical value at 25 °C)
- CROSS CONDUCTION PROTECTION
- TTL COMPATIBLE DRIVE
- OPERATING FREQUENCY UP TO 100 KHz
- THERMAL SHUTDOWN
- INTERNAL LOGIC SUPPLY
- HIGH EFFICIENCY

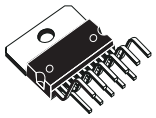
MULTIPOWER BCD TECHNOLOGY



Powerdip 12+3+3



SO20 (12+4+4)



Multiwatt11

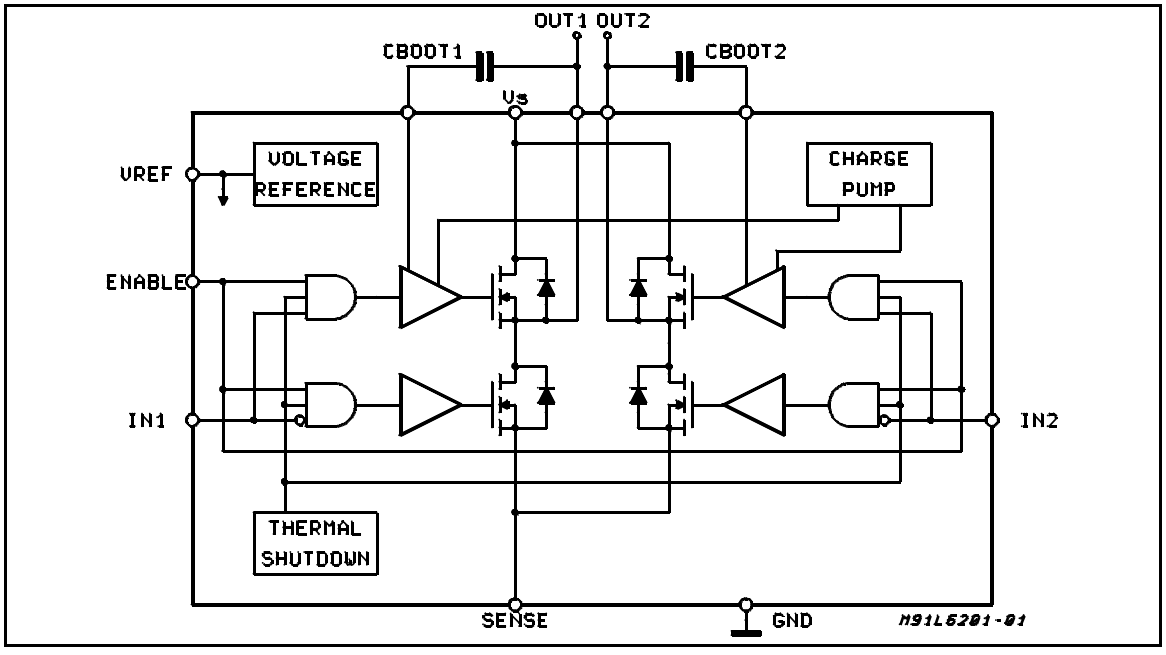
PowerSO20

ORDERING NUMBERS:
L6201 (SO20)
L6201PS (PowerSO20)
L6202 (Powerdip18)
L6203 (Multiwatt)

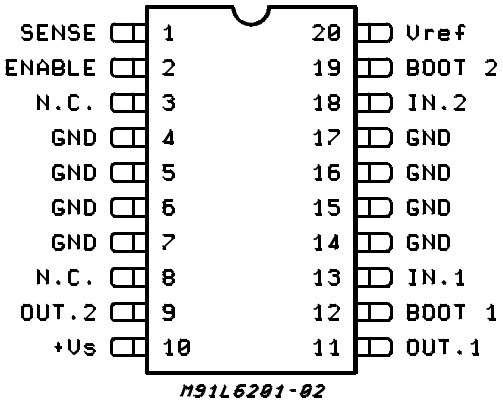
DESCRIPTION

The I.C. is a full bridge driver for motor control applications realized in Multipower-BCD technology which combines isolated DMOS power transistors with CMOS and Bipolar circuits on the same chip. By using mixed technology it has been possible to optimize the logic circuitry and the power stage to achieve the best possible performance. The DMOS output transistors can operate at supply voltages up to 42V and efficiently at high switching speeds. All the logic inputs are TTL, CMOS and μC compatible. Each channel (half-bridge) of the device is controlled by a separate logic input, while a common enable controls both channels. The I.C. is mounted in three different packages.

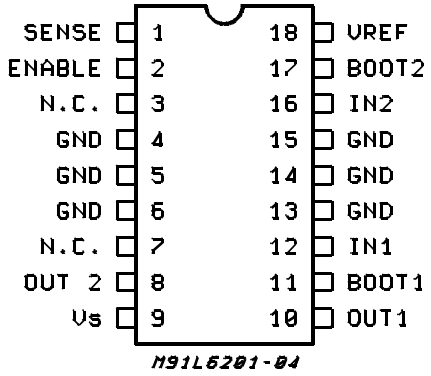
BLOCK DIAGRAM



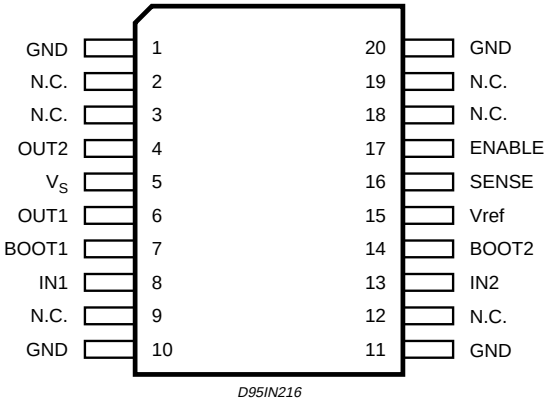
PIN CONNECTIONS (Top view)



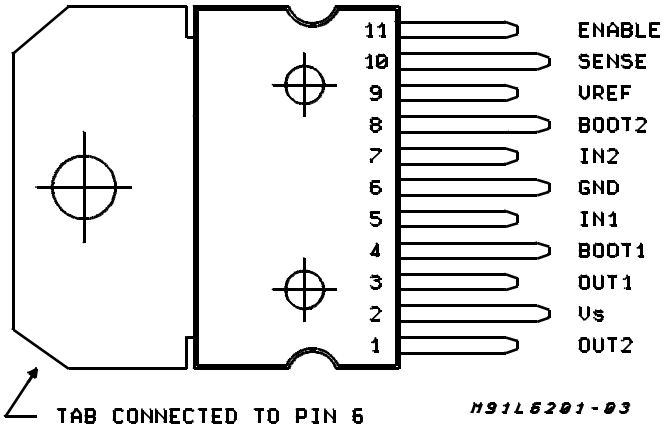
SO20



POWERDIP



PowerSO20



MULTIWATT11

PINS FUNCTIONS

Device				Name	Function
L6201	L6201PS	L6202	L6203		
1	16	1	10	SENSE	A resistor R_{sense} connected to this pin provides feedback for motor current control.
2	17	2	11	ENABLE	When a logic high is present on this pin the DMOS POWER transistors are enabled to be selectively driven by IN1 and IN2.
3	2,3,9,12, 18,19	3		N.C.	Not Connected
4,5	–	4	6	GND	Common Ground Terminal
–	1, 10	5		GND	Common Ground Terminal
6,7	–	6		GND	Common Ground Terminal
8	–	7		N.C.	Not Connected
9	4	8	1	OUT2	Ouput of 2nd Half Bridge
10	5	9	2	V_s	Supply Voltage
11	6	10	3	OUT1	Output of first Half Bridge
12	7	11	4	BOOT1	A boostrap capacitor connected to this pin ensures efficient driving of the upper POWER DMOS transistor.
13	8	12	5	IN1	Digital Input from the Motor Controller
14,15	–	13	6	GND	Common Ground Terminal
–	11, 20	14		GND	Common Ground Terminal
16,17	–	15		GND	Common Ground Terminal
18	13	16	7	IN2	Digital Input from the Motor Controller
19	14	17	8	BOOT2	A boostrap capacitor connected to this pin ensures efficient driving of the upper POWER DMOS transistor.
20	15	18	9	V_{ref}	Internal voltage reference. A capacitor from this pin to GND is recommended. The internal Ref. Voltage can source out a current of 2mA max.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Power Supply	52	V
V_{OD}	Differential Output Voltage (between Out1 and Out2)	60	V
V_{IN}, V_{EN}	Input or Enable Voltage	– 0.3 to + 7	V
I_o	Pulsed Output Current for L6201PS/L6202/L6203 (Note 1) – Non Repetitive (< 1 ms) for L6201 for L6201PS/L6202/L6203 DC Output Current for L6201 (Note 1)	5 5 10 1	A A A A
V_{sense}	Sensing Voltage	– 1 to + 4	V
V_b	Boostrap Peak Voltage	60	V
P_{tot}	Total Power Dissipation: T_{pins} = 90°C for L6201 for L6202 T_{case} = 90°C for L6201PS/L6203 T_{amb} = 70°C for L6201 (Note 2) for L6202 (Note 2) for L6201PS/L6203 (Note 2)	 4 5 20 0.9 1.3 2.3	 W W W W W W
T_{stg}, T_j	Storage and Junction Temperature	– 40 to + 150	°C

Note 1: Pulse width limited only by junction temperature and transient thermal impedance (see thermal characteristics)
Note 2: Mounted on board with minimized dissipating copper area.

THERMAL DATA

Symbol	Parameter		Value				Unit
			L6201	L6201PS	L6202	L6203	
$Rt_{h\ j\text{-}pins}$	Thermal Resistance Junction-pins	max	15	–	12	–	°C/W
$Rt_{h\ j\text{-}case}$	Thermal Resistance Junction Case	max.	–	–	–	3	
$Rt_{h\ j\text{-}amb}$	Thermal Resistance Junction-ambient	max.	85	13 (*)	60	35	

(*) Mounted on aluminium substrate.

ELECTRICAL CHARACTERISTICS (Refer to the Test Circuits; T_j = 25°C, V_S = 42V, V_{sens} = 0, unless otherwise specified).

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_s	Supply Voltage		12	36	48	V
V_{ref}	Reference Voltage	I_{REF} = 2mA		13.5		V
I_{REF}	Output Current				2	mA
I_s	Quiescent Supply Current	EN = H V_{IN} = L EN = H V_{IN} = H I_L = 0 EN = L (Fig. 1,2,3)		10 10 8	15 15 15	mA mA mA
f_c	Commutation Frequency (*)			30	100	KHz
T_j	Thermal Shutdown			150		°C
T_d	Dead Time Protection			100		ns

TRANSISTORS

OFF						
I_{DSS}	Leakage Current	Fig. 11 V_s = 52 V			1	mA
ON						
R_{DS}	On Resistance	Fig. 4,5		0.3	0.55	Ω
$V_{DS(ON)}$	Drain Source Voltage	Fig. 9 I_{DS} = 1A **L6201** I_{DS} = 1.2A **L6202** I_{DS} = 3A **L6201PS/03**		 0.3 0.36 0.9		 V V V
V_{sens}	Sensing Voltage		– 1		4	V

SOURCE DRAIN DIODE

V_{sd}	Forward ON Voltage	Fig. 6a and b I_{SD} = 1A **L6201** EN = L I_{SD} = 1.2A **L6202** EN = L I_{SD} = 3A **L6201PS/03** EN = L		 0.9 (**) 0.9 (**) 1.35(**)		 V V V
t_{rr}	Reverse Recovery Time	$\frac{dif}{dt}$ = 25 A/µs I_F = 1A L6201 I_F = 1.2A L6202 I_F = 3A L6203		300		ns
t_{fr}	Forward Recovery Time			200		ns

LOGIC LEVELS

$V_{IN\ L}$, $V_{EN\ L}$	Input Low Voltage		– 0.3		0.8	V
$V_{IN\ H}$, $V_{EN\ H}$	Input High Voltage		2		7	V
$I_{IN\ L}$, $I_{EN\ L}$	Input Low Current	V_{IN}, V_{EN} = L			–10	µA
$I_{IN\ H}$, $I_{EN\ H}$	Input High Current	V_{IN}, V_{EN} = H		30		µA

ELECTRICAL CHARACTERISTICS (Continued)
LOGIC CONTROL TO POWER DRIVE TIMING

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
t_1 (V_i)	Source Current Turn-off Delay	Fig. 12		300		ns
t_2 (V_i)	Source Current Fall Time	Fig. 12		200		ns
t_3 (V_i)	Source Current Turn-on Delay	Fig. 12		400		ns
t_4 (V_i)	Source Current Rise Time	Fig. 12		200		ns
t_5 (V_i)	Sink Current Turn-off Delay	Fig. 13		300		ns
t_6 (V_i)	Sink Current Fall Time	Fig. 13		200		ns
t_7 (V_i)	Sink Current Turn-on Delay	Fig. 13		400		ns
t_8 (V_i)	Sink Current Rise Time	Fig. 13		200		ns

(*) Limited by power dissipation
(**) In synchronous rectification the drain-source voltage drop VDS is shown in fig. 4 (L6202/03); typical value for the L6201 is of 0.3V.

Figure 1: Typical Normalized I_S vs. T_j

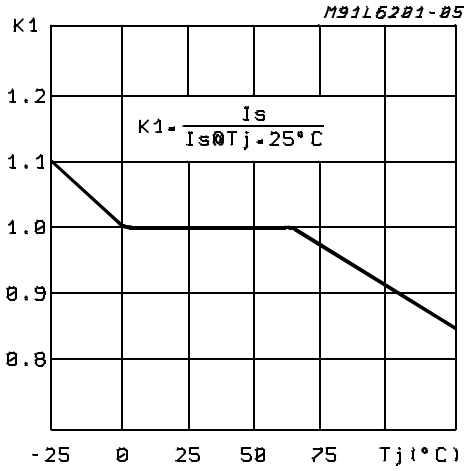


Figure 2: Typical Normalized Quiescent Current vs. Frequency

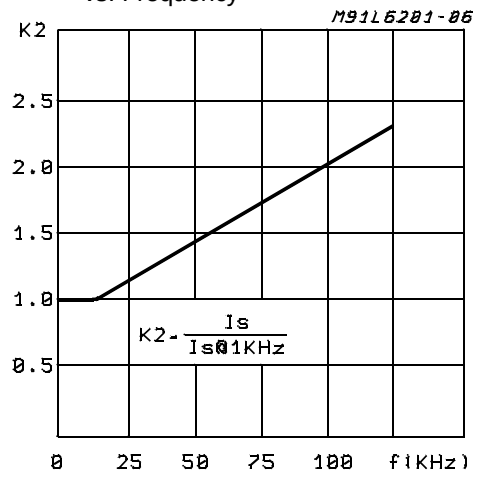


Figure 3: Typical Normalized I_S vs. V_S

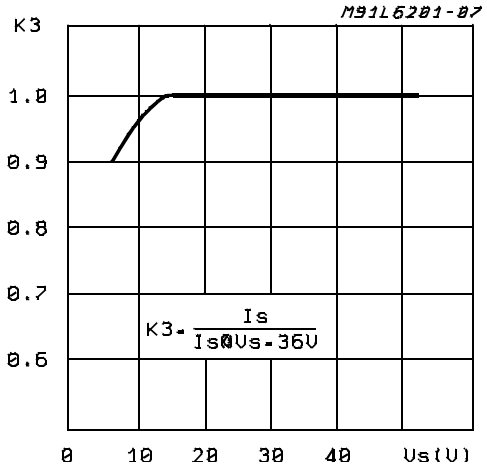
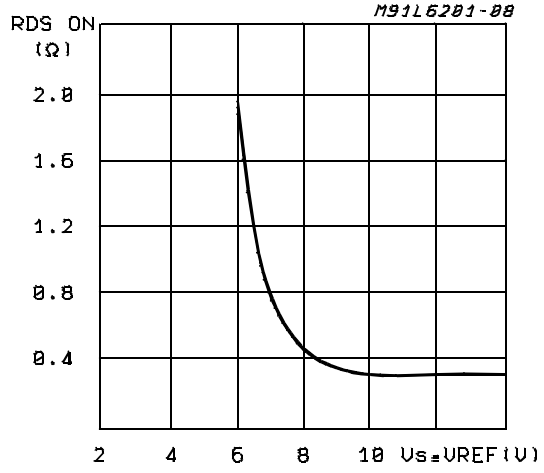


Figure 4: Typical $R_{DS\ (ON)}$ vs. V_S ~ V_{ref}



CIRCUIT DESCRIPTION

The L6201/1PS/2/3 is a monolithic full bridge switching motor driver realized in the new Multipower-BCD technology which allows the integration of multiple, isolated DMOS power transistors plus mixed CMOS/bipolar control circuits. In this way it has been possible to make all the control inputs TTL, CMOS and μC compatible and eliminate the necessity of external MOS drive components. The Logic Drive is shown in table 1.

Table 1

Inputs			Output Mosfets (*)
	IN1	IN2	
V_{EN} = H	L	L	Sink 1, Sink 2
	L	H	Sink 1, Source 2
	H	L	Source 1, Sink 2
	H	H	Source 1, Source 2
V_{EN} = L	X	X	All transistors turned oFF

L = Low H = High X = DON't care
(*) Numbers referred to INPUT1 or INPUT2 controlled output stages

Although the device guarantees the absence of cross-conduction, the presence of the intrinsic diodes in the POWER DMOS structure causes the generation of current spikes on the sensing terminals. This is due to charge-discharge phenomena in the capacitors C1 & C2 associated with the drain source junctions (fig. 14). When the output switches from high to low, a current spike is generated associated with the capacitor C1. On the low-to-high transition a spike of the same polarity is generated by C2, preceded by a spike of the opposite polarity due to the charging of the input capacity of the lower POWER DMOS transistor (fig. 15).

Figure 14: Intrinsic Structures in the POWER DMOS Transistors

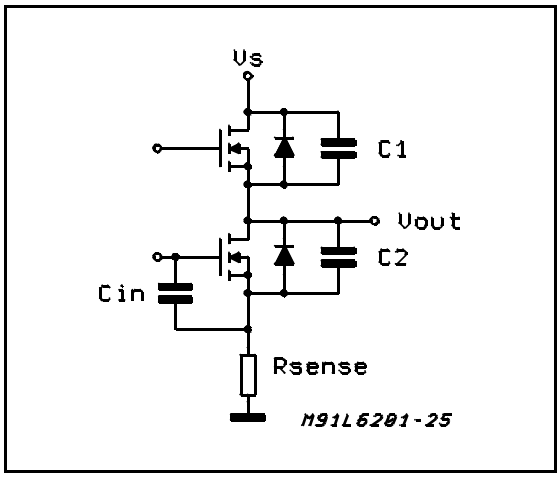
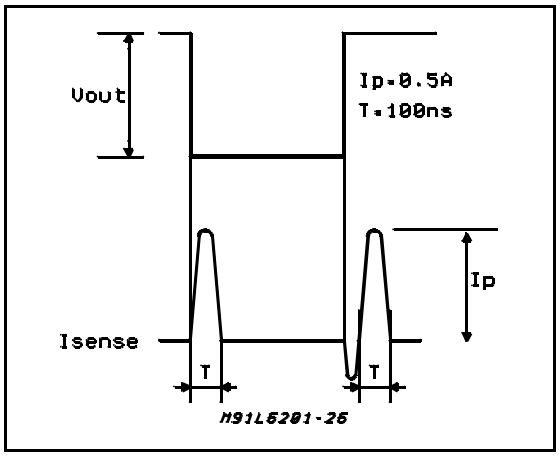


Figure 15: Current Typical Spikes on the Sensing Pin



TRANSISTOR OPERATION

ON State

When one of the POWER DMOS transistor is ON it can be considered as a resistor $R_{DS\ (ON)}$ throughout the recommended operating range. In this condition the dissipated power is given by :

$$P_{ON} = R_{DS\ (ON)} \cdot I_{DS}^2 \text{ (RMS)}$$

The low $R_{DS\ (ON)}$ of the Multipower-BCD process can provide high currents with low power dissipation.

OFF State

When one of the POWER DMOS transistor is OFF the V_{DS} voltage is equal to the supply voltage and only the leakage current I_{DSS} flows. The power dissipation during this period is given by :

$$P_{OFF} = V_S \cdot I_{DSS}$$

The power dissipation is very low and is negligible in comparison to that dissipated in the ON STATE.

Transitions

As already seen above the transistors have an intrinsic diode between their source and drain that can operate as a fast freewheeling diode in switched mode applications. During recirculation with the ENABLE input high, the voltage drop across the transistor is $R_{DS\ (ON)} \cdot I_D$ and when it reaches the diode forward voltage it is clamped. When the ENABLE input is low, the POWER MOS is OFF and the diode carries all of the recirculation current. The power dissipated in the transitional times in the cycle depends upon the voltage-current waveforms and in the driving mode. (see Fig. 7ab and Fig. 8abc).

$$P_{trans.} = I_{DS}(t) \cdot V_{DS}(t)$$

National Semiconductor

April 2003

LM340/LM78XX Series 3-Terminal Positive Regulators

General Description

The LM140/LM340A/LM340/LM78XXC monolithic 3-terminal positive voltage regulators employ internal current-limiting, thermal shutdown and safe-area compensation, making them essentially indestructible. If adequate heat sinking is provided, they can deliver over 1.0A output current. They are intended as fixed voltage regulators in a wide range of applications including local (on-card) regulation for elimination of noise and distribution problems associated with single-point regulation. In addition to use as fixed voltage regulators, these devices can be used with external components to obtain adjustable output voltages and currents.

Considerable effort was expended to make the entire series of regulators easy to use and minimize the number of external components. It is not necessary to bypass the output, although this does improve transient response. Input bypassing is needed only if the regulator is located far from the filter capacitor of the power supply.

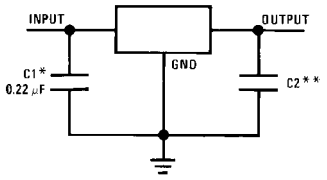
The 5V, 12V, and 15V regulator options are available in the steel TO-3 power package. The LM340A/LM340/LM78XXC series is available in the TO-220 plastic power package, and the LM340-5.0 is available in the SOT-223 package, as well as the LM340-5.0 and LM340-12 in the surface-mount TO-263 package.

Features

- Complete specifications at 1A load
- Output voltage tolerances of ±2% at T_j = 25˚C and ±4% over the temperature range (LM340A)
- Line regulation of 0.01% of V_{OUT}/V of ΔV_{IN} at 1A load (LM340A)
- Load regulation of 0.3% of V_{OUT}/A (LM340A)
- Internal thermal overload protection
- Internal short-circuit current limit
- Output transistor safe area protection
- P^+ Product Enhancement tested

Typical Applications

Fixed Output Regulator

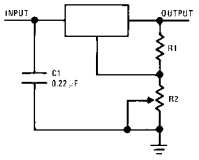


00778101

*Required if the regulator is located far from the power supply filter.

**Although no output capacitor is needed for stability, it does help transient response. (If needed, use 0.1 µF, ceramic disc).

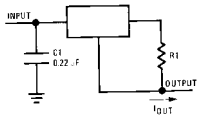
Adjustable Output Regulator



00778102

V_{OUT} = 5V + (5V/R1 + I_Q) R2 5V/R1 > 3 I_Q,
load regulation (L_r) ≈ [(R1 + R2)/R1] (L_r of LM340-5).

Current Regulator

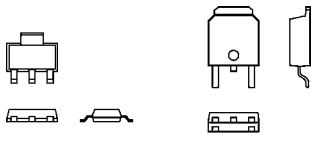


00778103

$$I_{OUT} = \frac{V2\text{-}3}{R1} + I_Q$$

ΔI_Q = 1.3 mA over line and load changes.

Comparison between SOT-223 and D-Pak (TO-252) Packages



00778138

Scale 1:1

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

(Note 5)

DC Input Voltage

All Devices except LM7824/LM7824C	35V
LM7824/LM7824C	40V
Internal Power Dissipation (Note 2)	Internally Limited
Maximum Junction Temperature	150˚C
Storage Temperature Range	−65˚C to +150˚C
Lead Temperature (Soldering, 10 sec.)	
TO-3 Package (K)	300˚C
TO-220 Package (T), TO-263 Package (S)	230˚C
ESD Susceptibility (Note 3)	2 kV

Operating Conditions (Note 1)

Temperature Range (T_A) (Note 2)	
LM140A, LM140	−55˚C to +125˚C
LM340A, LM340, LM7800	0˚C to +125˚C

LM340A Electrical Characteristics

I_{OUT} = 1A, −55˚C ≤ T_J ≤ +150˚C (LM140A), or 0˚C ≤ T_J ≤ + 125˚C (LM340A) unless otherwise specified (Note 4)

Symbol	Output Voltage		5V			12V			15V			Units
	Input Voltage (unless otherwise noted)		10V			19V			23V			
	Parameter	Conditions	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
V_O	Output Voltage	T_J = 25˚C	4.9	5	5.1	11.75	12	12.25	14.7	15	15.3	V
		P_D ≤ 15W, 5 mA ≤ I_O ≤ 1A	4.8		5.2	11.5		12.5	14.4		15.6	V
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}	(7.5 ≤ V_{IN} ≤ 20)			(14.8 ≤ V_{IN} ≤ 27)			(17.9 ≤ V_{IN} ≤ 30)			V
ΔV_O	Line Regulation	I_O = 500 mA			10			18			22	mV
		ΔV_{IN}	(7.5 ≤ V_{IN} ≤ 20)			(14.8 ≤ V_{IN} ≤ 27)			(17.9 ≤ V_{IN} ≤ 30)			V
		T_J = 25˚C		3	10		4	18		4	22	mV
		ΔV_{IN}	(7.5 ≤ V_{IN} ≤ 20)			(14.5 ≤ V_{IN} ≤ 27)			(17.5 ≤ V_{IN} ≤ 30)			V
		T_J = 25˚C			4			9			10	mV
		Over Temperature			12			30			30	mV
		ΔV_{IN}	(8 ≤ V_{IN} ≤ 12)			(16 ≤ V_{IN} ≤ 22)			(20 ≤ V_{IN} ≤ 26)			V
ΔV_O	Load Regulation	T_J = 25˚C, 5 mA ≤ I_O ≤ 1.5A		10	25		12	32		12	35	mV
		T_J = 25˚C, 250 mA ≤ I_O ≤ 750 mA			15			19			21	mV
		Over Temperature, 5 mA ≤ I_O ≤ 1A			25			60			75	mV
I_Q	Quiescent Current	T_J = 25˚C			6			6			6	mA
		Over Temperature			6.5			6.5			6.5	mA
ΔI_Q	Quiescent Current Change	5 mA ≤ I_O ≤ 1A		0.5			0.5			0.5		mA
		T_J = 25˚C, I_O = 1A			0.8			0.8			0.8	mA
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}	(7.5 ≤ V_{IN} ≤ 20)			(14.8 ≤ V_{IN} ≤ 27)			(17.9 ≤ V_{IN} ≤ 30)			V
		I_O = 500 mA			0.8			0.8			0.8	mA
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}	(8 ≤ V_{IN} ≤ 25)			(15 ≤ V_{IN} ≤ 30)			(17.9 ≤ V_{IN} ≤ 30)			V
V_N	Output Noise Voltage	T_A = 25˚C, 10 Hz ≤ f ≤ 100 kHz		40			75			90		µV
$\frac{\Delta V_{IN}}{\Delta V_{OUT}}$	Ripple Rejection	T_J = 25˚C, f = 120 Hz, I_O = 1A	68	80		61	72		60	70		dB
		or f = 120 Hz, I_O = 500 mA,	68			61			60			dB
		Over Temperature, V_{MIN} ≤ V_{IN} ≤ V_{MAX}	(8 ≤ V_{IN} ≤ 18)			(15 ≤ V_{IN} ≤ 25)			(18.5 ≤ V_{IN} ≤ 28.5)			V
R_O	Dropout Voltage	T_J = 25˚C, I_O = 1A		2.0			2.0			2.0		V
	Output Resistance	f = 1 kHz		8			18			19		mΩ

LM340A Electrical Characteristics (Continued)

I_{OUT} = 1A, −55˚C ≤ T_J ≤ +150˚C (LM140A), or 0˚C ≤ T_J ≤ + 125˚C (LM340A) unless otherwise specified (Note 4)

Symbol	Output Voltage		5V			12V			15V			Units
	Input Voltage (unless otherwise noted)		10V			19V			23V			
	Parameter	Conditions	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
	Short-Circuit Current	T_J = 25˚C		2.1			1.5			1.2		A
	Peak Output Current	T_J = 25˚C		2.4			2.4			2.4		A
	Average TC of V_O	Min, T_J = 0˚C, I_O = 5 mA		−0.6			−1.5			−1.8		mV/˚C
V_{IN}	Input Voltage Required to Maintain Line Regulation	T_J = 25˚C	7.5			14.5			17.5			V

LM140 Electrical Characteristics (Note 4)

−55˚C ≤ T_J ≤ +150˚C unless otherwise specified

Symbol	Output Voltage			5V			12V			15V			Units
	Input Voltage (unless otherwise noted)			10V			19V			23V			
	Parameter	Conditions		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
V_O	Output Voltage	T_J = 25˚C, 5 mA ≤ I_O ≤ 1A		4.8	5	5.2	11.5	12	12.5	14.4	15	15.6	V
		P_D ≤ 15W, 5 mA ≤ I_O ≤ 1A V_{MIN} ≤ V_{IN} ≤ V_{MAX}		4.75 (8 ≤ V_{IN} ≤ 20)		5.25	11.4 (15.5 ≤ V_{IN} ≤ 27)		12.6	14.25 (18.5 ≤ V_{IN} ≤ 30)		15.75	V V
ΔV_O	Line Regulation	I_O = 500 mA	T_J = 25˚C ΔV_{IN}		3 (7 ≤ V_{IN} ≤ 25)	50		4 (14.5 ≤ V_{IN} ≤ 30)	120		4 (17.5 ≤ V_{IN} ≤ 30)	150	mV V
			−55˚C ≤ T_J ≤ +150˚C ΔV_{IN}			50 (8 ≤ V_{IN} ≤ 20)			120 (15 ≤ V_{IN} ≤ 27)			150 (18.5 ≤ V_{IN} ≤ 30)	mV V
		I_O ≤ 1A	T_J = 25˚C ΔV_{IN}			50 (7.5 ≤ V_{IN} ≤ 20)			120 (14.6 ≤ V_{IN} ≤ 27)			150 (17.7 ≤ V_{IN} ≤ 30)	mV V
			−55˚C ≤ T_J ≤ +150˚C ΔV_{IN}			25 (8 ≤ V_{IN} ≤ 12)			60 (16 ≤ V_{IN} ≤ 22)			75 (20 ≤ V_{IN} ≤ 26)	mV V
ΔV_O	Load Regulation	T_J = 25˚C	5 mA ≤ I_O ≤ 1.5A 250 mA ≤ I_P ≤ 750 mA		10 25	50		12 60	120		12 75	150	mV mV
		−55˚C ≤ T_J ≤ +150˚C, 5 mA ≤ I_O ≤ 1A				50			120			150	mV
I_Q	Quiescent Current	I_O ≤ 1A	T_J = 25˚C			6			6			6	mA
			−55˚C ≤ T_J ≤ +150˚C			7			7			7	mA
ΔI_Q	Quiescent Current Change	5 mA ≤ I_O ≤ 1A			0.5			0.5			0.5		mA
		T_J = 25˚C, I_O ≤ 1A V_{MIN} ≤ V_{IN} ≤ V_{MAX}				0.8 (8 ≤ V_{IN} ≤ 20)			0.8 (15 ≤ V_{IN} ≤ 27)			0.8 (18.5 ≤ V_{IN} ≤ 30)	mA V
		I_O = 500 mA, −55˚C ≤ T_J ≤ +150˚C V_{MIN} ≤ V_{IN} ≤ V_{MAX}				0.8 (8 ≤ V_{IN} ≤ 25)			0.8 (15 ≤ V_{IN} ≤ 30)			0.8 (18.5 ≤ V_{IN} ≤ 30)	mA V
V_N	Output Noise Voltage	T_A = 25˚C, 10 Hz ≤ f ≤ 100 kHz			40			75			90		µV

Application Hints

The LM340/LM78XX series is designed with thermal protection, output short-circuit protection and output transistor safe area protection. However, as with *any* IC regulator, it becomes necessary to take precautions to assure that the regulator is not inadvertently damaged. The following describes possible misapplications and methods to prevent damage to the regulator.

SHORTING THE REGULATOR INPUT

When using large capacitors at the output of these regulators, a protection diode connected input to output (*Figure 1*) may be required if the input is shorted to ground. Without the protection diode, an input short will cause the input to rapidly approach ground potential, while the output remains near the initial V_{OUT}because of the stored charge in the large output capacitor. The capacitor will then discharge through a large internal input to output diode and parasitic transistors. If the energy released by the capacitor is large enough, this diode, low current metal and the regulator will be destroyed. The fast diode in *Figure 1* will shunt most of the capacitors discharge current around the regulator. Generally no protection diode is required for values of output capacitance ≤ 10 µF.

RAISING THE OUTPUT VOLTAGE ABOVE THE INPUT VOLTAGE

Since the output of the device does not sink current, forcing the output high can cause damage to internal low current paths in a manner similar to that just described in the “Shorting the Regulator Input” section.

REGULATOR FLOATING GROUND (*Figure 2*)

When the ground pin alone becomes disconnected, the output approaches the unregulated input, causing possible damage to other circuits connected to V_{OUT}. If ground is reconnected with power “ON”, damage may also occur to the regulator. This fault is most likely to occur when plugging in regulators or modules with on card regulators into powered up sockets. Power should be turned off first, thermal limit ceases operating, or ground should be connected first if power must be left on.

TRANSIENT VOLTAGES

If transients exceed the maximum rated input voltage of the device, or reach more than 0.8V below ground and have sufficient energy, they will damage the regulator. The solution is to use a large input capacitor, a series input breakdown diode, a choke, a transient suppressor or a combination of these.

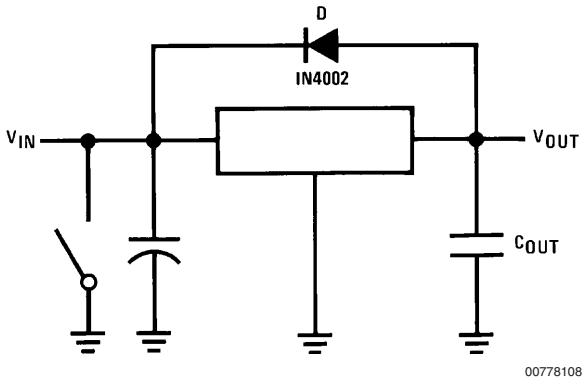


FIGURE 1. Input Short

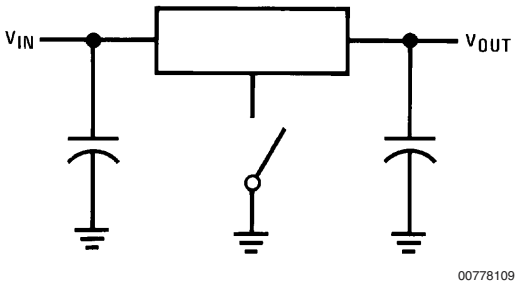


FIGURE 2. Regulator Floating Ground

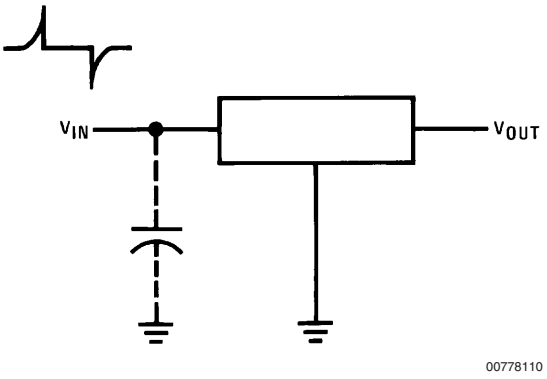


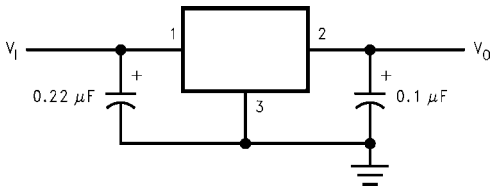
FIGURE 3. Transients

When a value for $\theta_{(H-A)}$ is found using the equation shown, a heatsink must be selected that has *a value that is less than or equal to this number.*

$\theta_{(H-A)}$ is specified numerically by the heatsink manufacturer in this catalog, or shown in a curve that plots temperature rise vs power dissipation for the heatsink.

Typical Applications

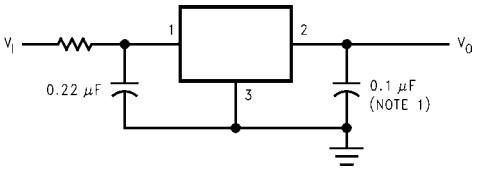
Fixed Output Regulator



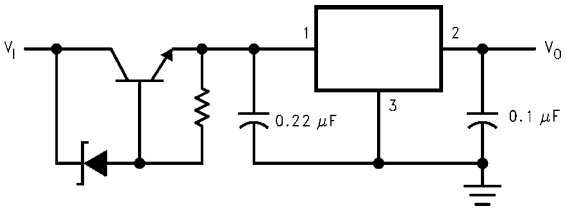
00778113

Note: Bypass capacitors are recommended for optimum stability and transient response, and should be located as close as possible to the regulator.

High Input Voltage Circuits

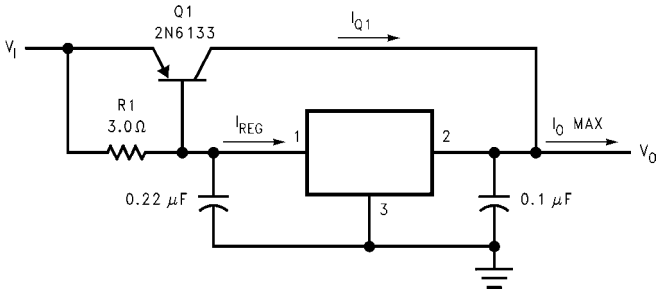


00778114



00778115

High Current Voltage Regulator



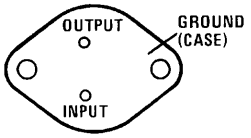
00778116

$$\beta(Q1) \geq \frac{I_{O\ Max}}{I_{REG\ Max}}$$

$$R1 = \frac{0.9}{I_{REG}} = \frac{\beta(Q1)\ V_{BE(Q1)}}{I_{REG\ Max}\ (\beta + 1) - I_{O\ Max}}$$

Connection Diagrams and Ordering Information

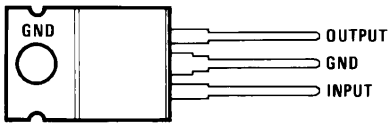
TO-3 Metal Can Package (K)



00778111

Bottom View
Steel Package Order Numbers:
LM140K-5.0 LM140K-12 LM140K-15
LM340K-12 LM340K-15
LM340K-5.0
See Package Number K02A
LM140K-5.0/883 LM140K-12/883 LM140K-15/883
See Package Number K02C

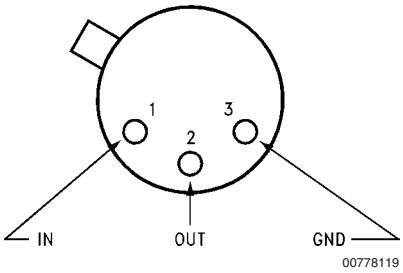
TO-220 Power Package (T)



00778112

Top View
Plastic Package Order Numbers:
LM340AT-5.0 LM340T-5.0
LM340T-12 LM340T-15
LM7805CT LM7812CT
LM7815CT LM7808CT
See Package Number T03B

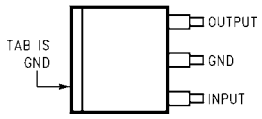
TO-39 Metal Can Package (H)



00778119

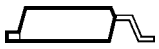
Top View
Metal Can Order Numbers†:
LM140H-5.0/883 LM140H-6.0/883
LM140H-8.0/883 LM140H-12/883
LM140H-15/883 LM140H-24/883
See Package Number H03A

TO-263 Surface-Mount Package (S)



00778120

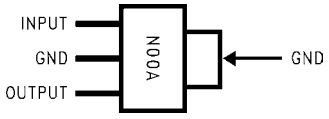
Top View



00778121

Side View
Surface-Mount Package Order Numbers:
LM340S-5.0 LM340S-12
See Package Number TS3B

3-Lead SOT-223
(Front View)
Order Number LM340MP-5.0
Package Marked NO0A
See Package Number MA04A



00778143

†The specifications for the LM140H/883 devices are not contained in this datasheet. If specifications for these devices are required, contact the National Semiconductor Sales Office/Distributors.

19-4323; Rev 9; 5/00

MAXIM

+5V-Powered, Multichannel RS-232 Drivers/Receivers

General Description

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where ±12V is not available.

These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5µW. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Applications

Portable Computers

Low-Power Modems

Interface Translation

Battery-Powered RS-232 Systems

Multidrop RS-232 Networks

Features

Superior to Bipolar

- **Operate from Single +5V Power Supply (+5V and +12V—MAX231/MAX239)**
- **Low-Power Receive Mode in Shutdown (MAX223/MAX242)**
- **Meet All EIA/TIA-232E and V.28 Specifications**
- **Multiple Drivers and Receivers**
- **3-State Driver and Receiver Outputs**
- **Open-Line Detection (MAX243)**

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX220CPE	0°C to +70°C	16 Plastic DIP
MAX220CSE	0°C to +70°C	16 Narrow SO
MAX220CWE	0°C to +70°C	16 Wide SO
MAX220C/D	0°C to +70°C	Dice*
MAX220EPE	-40°C to +85°C	16 Plastic DIP
MAX220ESE	-40°C to +85°C	16 Narrow SO
MAX220EWE	-40°C to +85°C	16 Wide SO
MAX220EJE	-40°C to +85°C	16 CERDIP
MAX220MJE	-55°C to +125°C	16 CERDIP

Ordering Information continued at end of data sheet.

**Contact factory for dice specifications.*

Selection Table

Part Number	Power Supply (V)	No. of RS-232 Drivers/Rx	No. of Ext. Caps	Nominal Cap. Value (µF)	SHDN & Three-State	Rx Active in SHDN	Data Rate (kbps)	Features
MAX220	+5	2/2	4	0.1	No	—	120	Ultra-low-power, industry-standard pinout
MAX222	+5	2/2	4	0.1	Yes	—	200	Low-power shutdown
MAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✔	120	MAX241 and receivers active in shutdown
MAX225	+5	5/5	0	—	Yes	✔	120	Available in SO
MAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes	—	120	5 drivers with shutdown
MAX231 (MAX201)	+5 and +7.5 to +13.2	2/2	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; same functions as MAX232
MAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No	—	120 (64)	Industry standard
MAX232A	+5	2/2	4	0.1	No	—	200	Higher slew rate, small caps
MAX233 (MAX203)	+5	2/2	0	—	No	—	120	No external caps
MAX233A	+5	2/2	0	—	No	—	200	No external caps, high slew rate
MAX234 (MAX204)	+5	4/0	4	1.0 (0.1)	No	—	120	Replaces 1488
MAX235 (MAX205)	+5	5/5	0	—	Yes	—	120	No external caps
MAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes	—	120	Shutdown, three state
MAX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No	—	120	Complements IBM PC serial port
MAX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No	—	120	Replaces 1488 and 1489
MAX239 (MAX209)	+5 and +7.5 to +13.2	3/5	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; single-package solution for IBM PC serial port
MAX240	+5	5/5	4	1.0	Yes	—	120	DIP or flatpack package
MAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes	—	120	Complete IBM PC serial port
MAX242	+5	2/2	4	0.1	Yes	✔	200	Separate shutdown and enable
MAX243	+5	2/2	4	0.1	No	—	200	Open-line detection simplifies cabling
MAX244	+5	8/10	4	1.0	No	—	120	High slew rate
MAX245	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, two shutdown modes
MAX246	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, three shutdown modes
MAX247	+5	8/9	0	—	Yes	✔	120	High slew rate, int. caps, nine operating modes
MAX248	+5	8/8	4	1.0	Yes	✔	120	High slew rate, selective half-chip enables
MAX249	+5	6/10	4	1.0	Yes	✔	120	Available in quad flatpack package

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC}) .....-0.3V to +6V
Input Voltages
T_{IN}.....-0.3V to (V_{CC} - 0.3V)
R_{IN} (Except MAX220) .....±30V
R_{IN} (MAX220).....±25V
T_{OUT} (Except MAX220) (Note 1) .....±15V
T_{OUT} (MAX220).....±13.2V
Output Voltages
T_{OUT}.....±15V
R_{OUT}.....-0.3V to (V_{CC} + 0.3V)
Driver/Receiver Output Short Circuited to GND.....Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C)....842mW
18-Pin Plastic DIP (derate 11.11mW/°C above +70°C)....889mW
20-Pin Plastic DIP (derate 8.00mW/°C above +70°C) ..440mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) ...696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C).....762mW
18-Pin Wide SO (derate 9.52mW/°C above +70°C).....762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C)....800mW
20-Pin SSOP (derate 8.00mW/°C above +70°C) 640mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C).....800mW
18-Pin CERDIP (derate 10.53mW/°C above +70°C).....842mW
Operating Temperature Ranges
MAX2_ _AC_ _, MAX2_ _C_ _.....0°C to +70°C
MAX2_ _AE_ _, MAX2_ _E_ _ .....-40°C to +85°C
MAX2_ _AM_ _, MAX2_ _M_ _.....-55°C to +125°C
Storage Temperature Range .....-65°C to +160°C
Lead Temperature (soldering, 10sec) .....+300°C

Note 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.
Note 2: For the MAX220, V+ and V- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High	All devices except MAX220		2	1.4		V
	MAX220: V_{CC} = 5.0V		2.4			
Logic Pull-Up/Input Current	All except MAX220, normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown, MAX220			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate				200	116	kb/s
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±22		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V_{CC} = 5V, no hysteresis in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243 (continued)

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
TTL/CMOS Output Leakage Current	$\overline{SHDN}$ = V_{CC} or $\overline{EN}$ = V_{CC} ($\overline{SHDN}$ = 0V for MAX222), 0V ≤ V_{OUT} ≤ V_{CC}			±0.05	±10	µA
$\overline{EN}$ Input Threshold Low	MAX242			1.4	0.8	V
$\overline{EN}$ Input Threshold High	MAX242		2.0	1.4		V
Operating Supply Voltage			4.5		5.5	V
V_{CC} Supply Current ($\overline{SHDN}$ = V_{CC}), Figures 5, 6, 11, 19	No load	MAX220		0.5	2	mA
		MAX222/232A/233A/242/243		4	10	
	3kΩ load both inputs	MAX220		12		
		MAX222/232A/233A/242/243		15		
Shutdown Supply Current	MAX222/242	T_A = +25°C		0.1	10	µA
		T_A = 0°C to +70°C		2	50	
		T_A = -40°C to +85°C		2	50	
		T_A = -55°C to +125°C		35	100	
$\overline{SHDN}$ Input Leakage Current	MAX222/242				±1	µA
$\overline{SHDN}$ Threshold Low	MAX222/242			1.4	0.8	V
$\overline{SHDN}$ Threshold High	MAX222/242		2.0	1.4		V
Transition Slew Rate	C_L = 50pF to 2500pF, R_L = 3kΩ to 7kΩ, V_{CC} = 5V, T_A = +25°C, measured from +3V to -3V or -3V to +3V	MAX222/232A/233A/242/243	6	12	30	V/µs
		MAX220	1.5	3	30	
Transmitter Propagation Delay TLL to RS-232 (normal operation), Figure 1	t_{PHLT}	MAX222/232A/233A/242/243		1.3	3.5	µs
		MAX220		4	10	
	t_{PLHT}	MAX222/232A/233A/242/243		1.5	3.5	
		MAX220		5	10	
Receiver Propagation Delay RS-232 to TLL (normal operation), Figure 2	t_{PHLR}	MAX222/232A/233A/242/243		0.5	1	µs
		MAX220		0.6	3	
	t_{PLHR}	MAX222/232A/233A/242/243		0.6	1	
		MAX220		0.8	3	
Receiver Propagation Delay RS-232 to TLL (shutdown), Figure 2	t_{PHLS}	MAX242		0.5	10	µs
	t_{PLHS}	MAX242		2.5	10	
Receiver-Output Enable Time, Figure 3	t_{ER}	MAX242		125	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}	MAX242		160	500	ns
Transmitter-Output Enable Time ($\overline{SHDN}$ goes high), Figure 4	t_{ET}	MAX222/242, 0.1µF caps (includes charge-pump start-up)		250		µs
Transmitter-Output Disable Time ($\overline{SHDN}$ goes low), Figure 4	t_{DT}	MAX222/242, 0.1µF caps		600		ns
Transmitter + to - Propagation Delay Difference (normal operation)	t_{PHLT} - t_{PLHT}	MAX222/232A/233A/242/243		300		ns
		MAX220		2000		
Receiver + to - Propagation Delay Difference (normal operation)	t_{PHLR} - t_{PLHR}	MAX222/232A/233A/242/243		100		ns
		MAX220		225		

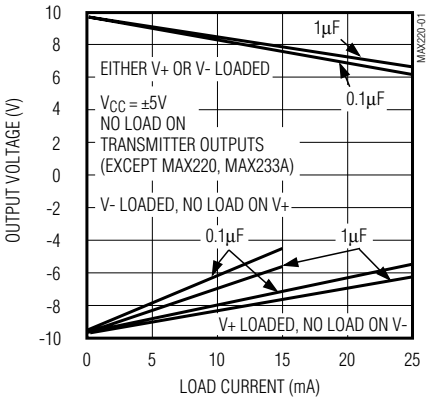
Note 3: MAX243 $R2_{OUT}$ is guaranteed to be low when $R2_{IN}$ is ≥ 0V or is floating.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

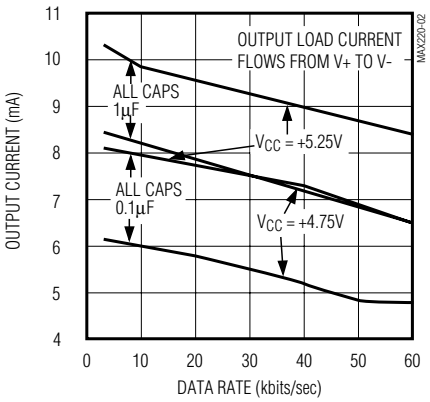
Typical Operating Characteristics

MAX220/MAX222/MAX232A/MAX233A/MAX242/MAX243

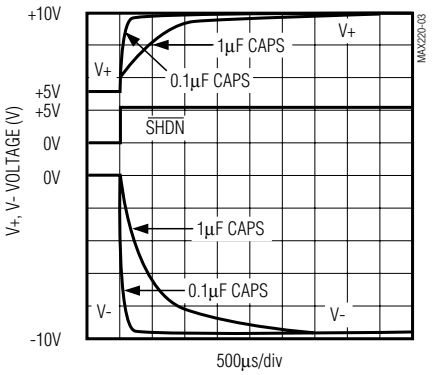
OUTPUT VOLTAGE vs. LOAD CURRENT

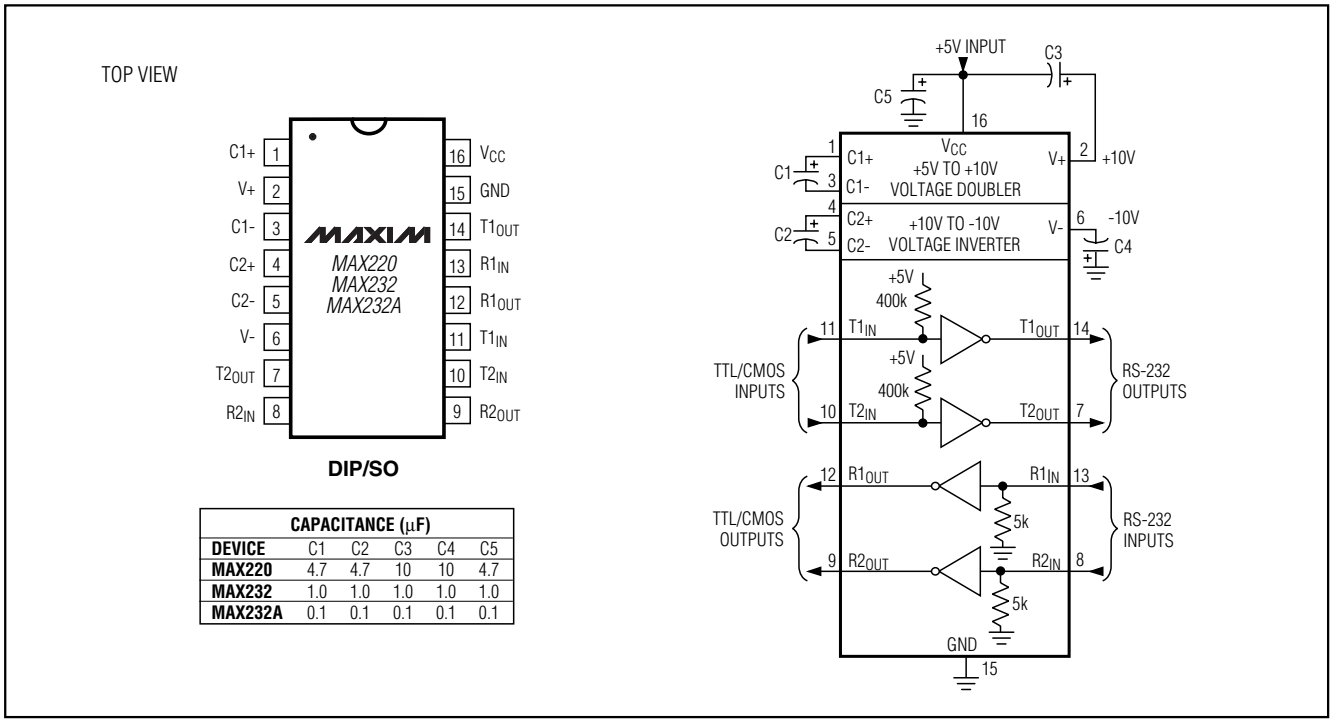


AVAILABLE OUTPUT CURRENT vs. DATA RATE



MAX222/MAX242 ON-TIME EXITING SHUTDOWN





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

Figure 5. MAX220/MAX232/MAX232A Pin Configuration and Typical Operating Circuit

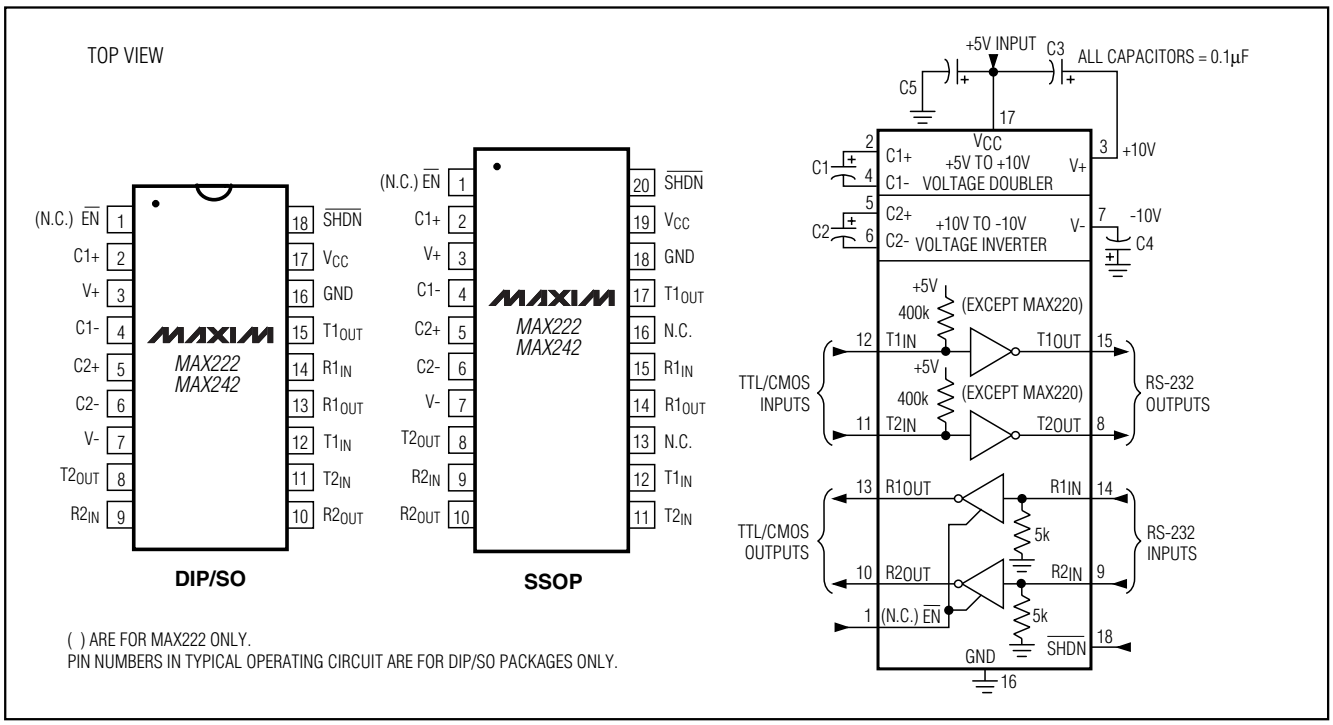


Figure 6. MAX222/MAX242 Pin Configurations and Typical Operating Circuit