ESCUELA POLITÉCNICA

NACIONAL

ESCUELA DE INGENIERÍA

SISTEMA DE DESARROLLO PARA LOS MICROCONTROLADORES PIC

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRONICA Y CONTROL

GERMÁN BOLÍVAR AUCAPIÑA ESCOBAR

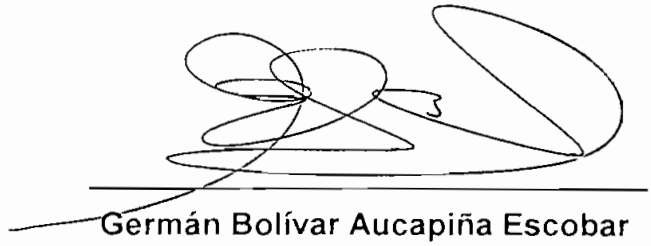
DIRECTOR: DR. LUIS CORRALES

Quito, Enero 2002

DECLARACIÓN

Yo, Germán Bolívar Aucapiña Escobar, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la ley de propiedad intelectual, por su reglamento y por la normativa institucional vigente.



Germán Bolívar Aucapiña Escobar

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Germán Bolívar Aucapiña Escobar, bajo mi supervisión.

Dr. Luis Corrales

DIRECTOR DE PROYECTO

AGRADECIMIENTO

AGRADEZCO A TODAS LAS PERSONAS QUE DE UNA U OTRA MANERA COLABORARON EN EL DESARROLLO DE ESTA TESIS, EN ESPECIAL AL DR. LUIS CORRALES.

DEDICATORIA

DEDICO EL TRABAJO REALIZADO EN ESTA TESIS, A DIOS Y A MIS PADRES, QUE ME HAN ALENTADO DURANTE TODA LA CARRERA PARA ALCANZAR ESTA META TAN ANELADA.

CONTENIDO

CAPITULO I 1

1.1 Introducción 1

1.2 El PIC16F84A y PIC16C84 3

1.2.1 Características Generales 3

1.2.2 Pines y funciones 6

1.2.3 Puertos del microcontrolador 7

1.2.4 El oscilador externo 12

1.2.5 Reset 14

1.2.6 Arquitectura 18

1.2.7 Memoria de programa 19

1.2.8 Registros (Memoria RAM) 20

1.2.9 Registro de trabajo W 32

1.2.10 Pila (Stack) 33

1.2.11 Características especiales 33

1.2.12 El conjunto de instrucciones 42

1.3 El PIC16C71 44

1.3.1 Características especiales 44

1.3.2 Pines y funciones 46

13.3 Puertos del microcontrolador 47

1.3.4 El oscilador externo 52

1.3.5 Reset 54

1.3.6 Arquitectura 57

1.3.7 Memoria de programa 59

1.3.8 Registros (Memoria RAM) 60

1.3.9 Registro de trabajo 71

1.3.10 Pila (Stack) 71

1.3.11 Características especiales 72

1.3.12 El conjunto de instrucciones 84

1.4 El PIC12C50X 87

1.4.1 Características especiales 87

1.4.2 Arquitectura 88

1.4.3 Descripción de pines 89
1.4.4 Organización de la memoria de programa 91
1.4.5 Organización de la memoria de datos 92
1.4.6 Registros de función especial 93
1.4.7 Características especiales 97
CAPITULOII 103
2.1 Introducción 103
2.2 Instalación 103
2.3 Configuración del modo de desarrollo 105
2.4 Creación de un proyecto 106
2.5 Edición del código fuente 109
2.6 Ensamblaje de un archivo 114
2.7 Simulación 115
2.7.1 Otras ventanas de observación 116
2.7.2 Puntos de interrupción 116
2.7.3 Estímulos 117
2.7.4 Opciones varias de simulación 121
CAPITULO III 123
3.1 Configuración general 123
3.2 Diagrama de bloques 124
3.3 Diseño circuital 125
3.3.1 Fuente de alimentación 126
3.3.2 Módulo de conexiones de microcontroladores 130
3.3.3 Módulo de leds 132
3.3.4 Módulo de displays de 7 segmentos 133
3.3.5 Módulo de display de cristal líquido 133
3.3.6 Módulo de dipswitch 136
3.3.7 Módulo de teclado matricial 137
3.3.8 Módulo de pulsadores 138
3.3.9 Módulo de comunicaciones seriales RS-232 138
3.3.10 Módulo de comunicaciones seriales RS-422/485 139
3.3.11 Módulo de memoria serial 141
3.3.12 Módulo de entrada de señales analógicas 144

3.3.13 Módulo de potenciómetro digital 145
3.3.14 Módulo de zumbador piezoeléctrico 147
3.3.15 Módulo de salida PWM 148
3.3.16 Módulo de reloj de tiempo real 149
3.3.17 Módulo de relé 150
3.3.18 Módulo de programador en circuito 151
CAPÍTULO IV 154
4.1 Prácticas con el PIC16F84A 154
4.1.1 Manejo de leds, pulsantes y dipswitch 155
4.1.2 Manejo de displays de 7 segmentos 165
4.1.3 Multiplexaje de teclados y displays 168
4.1.4 Manejo de módulo LCD 181
4.1.5 Comunicación serial 191
4.1.6 Comunicación I^2C 198
4.2 Prácticas con el PIC16C71 208
4.3 Prácticas con el PIC12C508 214
CAPÍTULO V 217
5.1 Conclusiones 217
5.2 Recomendaciones 218
BIBLIOGRAFÍA 219
ANEXOS 220
Descripción de instrucciones
Presupuestos
Diagrama del circuito impreso
Diagrama frontal del impreso
Hojas técnicas de la memoria serial
Hojas técnicas del reloj de tiempo real
Hojas técnicas del módulo LCD
Hojas técnicas del potenciómetro digital
Hojas técnicas del integrado 75176 (RS 422/488)
Hojas técnicas del sensor de temperatura LM35

RESUMEN

El objetivo de este proyecto fue diseñar y construir un sistema de desarrollo para los microcontroladores PIC de la familia enana y media. El sistema de desarrollo originalmente propuesto debía trabajar con el microcontrolador PIC16C84. Pero, ya que este debe ser concebido como un sistema didáctico, se optó por usar una variante de este microcontrolador que es el PIC16F84A, cuya diferencia importante es que tiene memoria de programa tipo Flash en vez de EEPROM, que facilita los procesos de lectura y escritura de programas. También la arquitectura debía ser similar a la del sistema de desarrollo Microlab existente en el Laboratorio de Control, para los microcontroladores INTEL; sin embargo para que el sistema trabaje con varios modelos de micros además del PIC16F84A, como son los PIC16C7X, PIC12C50X y los nuevos PIC16F87X, y todas las versiones compatibles pin a pin, entonces se optó por realizar un diseño de arquitectura abierta, lo que permite la experimentación de varios modelos de micros con una amplia variedad de periféricos. El sistema fue probado durante la realización de prácticas de laboratorio que pueden hacerse con el sistema de desarrollo. Las prácticas cubrieron la mayoría de tareas que se pueden hacer con un microcontrolador como: manejo de pulsantes, de leds, de teclado matricial, dipswitch, displays, módulos LCD, memorias seriales, potenciómetros digitales, reloj calendario, comunicaciones RS 232 / 422 / 485, señales analógicas, salidas PWM, relé y zumbador piezoeléctrico.

Esta versatilidad conseguida demostró que el sistema cumple con su objetivo de proveer una herramienta de desarrollo de proyectos con PICs.

PRESENTACIÓN

Los modernos microcontroladores PIC han revolucionado la automatización, gracias a sus enormes ventajas en características y precios, y sus aplicaciones están limitadas únicamente por la imaginación. Cada día es más frecuente el uso de estos dispositivos para el control de equipos a todo nivel, pero existen muchos campos en los que esta innovación tecnológica no se ha aplicado.

El presente proyecto de titulación describe el diseño y construcción de un sistema de desarrollo para los microcontroladores PIC, de las familias media y enana; lo que permitirá el estudio de estos micros, que poseen integrados todos los recursos que precisan las aplicaciones comerciales de nivel medio y alto.

Para esto en el primer capítulo se realiza una descripción detallada de los microcontroladores PIC16F84A, PIC16C7X y PIC12C50X; que son los más representativos de estas familias. En el segundo capítulo se ha realizado una explicación del manejo del programa MPLAB de microchip, el cual permite la escritura y simulación de los programas para estos micros. En el tercer capítulo se desarrolla el diseño del circuito con todos los módulos necesarios para la experimentación; aquí se incluye un programador en circuito, que facilita la escritura y borrado de programas. Además de los zócalos necesarios para los micros antes mencionados, se ha implementado dos zócalos adicionales para los micros de 28 y 40 pines de la familia PIC16F87X, lo que permitirá la experimentación con estos nuevos microcontroladores que poseen integrados varios elementos y representa un espectacular avance con respecto al PIC16F84A. En el cuarto capítulo se desarrollan varias prácticas como ejemplo del manejo del sistema de desarrollo, estas prácticas han sido diseñadas para permitir un avance progresivo en el manejo de estos micros. Además el sistema posee arquitectura abierta lo que permite desarrollar prácticas planteadas por el usuario. En el quinto capítulo se indica las conclusiones del desarrollo de este proyecto y las recomendaciones para el manejo de la misma.

También se ha adicionado en los anexos la explicación del funcionamiento de las instrucciones de estos microcontroladores, el presupuesto, los diagramas para la fabricación del circuito impreso, y las hojas técnicas de los elementos utilizados en el sistema.

CAPITULO I
LOS MICROCONTROLADORES PIC

1.1 INTRODUCCION

Los microcontroladores resumen en un solo circuito integrado más de 30 años de tecnología electrónica, estos chips contienen la misma estructura de un microprocesador de un computador personal, con la diferencia de que estos son sistemas cerrados de prestaciones limitadas a una aplicación exclusiva.

Las grandes ventajas de los microcontroladores para el control de sistemas son: alto desempeño, eficiencia, facilidad de desarrollo y un excelente factor costo-beneficio. El avance tecnológico ha logrado reducir por lo expuesto, el tamaño sin afectar negativamente su rendimiento, en los últimos años los microcontroladores se han difundido notoriamente en el mundo, y se prevee que para este siglo se tenga casi todos los elementos que nos rodean manejados con estos chips.

Las amplias posibilidades de aplicación de estos elementos, que se pueden considerar ilimitadas, exigen la necesidad de aprender a manejarlos, razón por la cual esta tesis propone diseñar y construir un sistema de desarrollo para estudiar paso a paso, mediante ejercicios prácticos, la estructura y características de los microcontroladores.

La familia de microcontroladores PIC de Microchip Technology Inc. ha dominado en los últimos años el mercado, pues tiene ventajas frente a otras familias de microcontroladores como son: Alta velocidad, bajo costo, bajo consumo de potencia, excelente rendimiento y sobre todo, una gran variedad de modelos que permite encontrar el más adecuado para cada aplicación. Además, existen microcontroladores que tienen incluidos en su arquitectura interna dispositivos tales como conversores análogo digitales, memorias EEPROM, puertos de comunicación, osciladores, circuitos de reset al encendido, circuitos de vigilancia de funcionamiento, resistencias pull up, entre otros que permiten simplificar el diseño y la circuitería de los proyectos.

El sistema de desarrollo originalmente propuesto debía trabajar con el microcontrolador PIC16C84 que posee además de las características generales

de los PIC una memoria EEPROM de programas y de datos. Pero, ya que este debe ser concevido como un sistema didáctico, se ha optado por usar una variante de este microcontrolador que es el PIC16F84A cuya diferencia importante es que tiene memoria de programa tipo Flash en vez de EEPROM, de mejor tecnología en los procesos de escritura/borrado. Estos microcontroladores presentan una gran facilidad en el desarrollo de prácticas y prototipos ya que no se requiere de borrado con luz ultravioleta, como en las versiones EPROM, sino que permite reprogramarlos nuevamente realizando un rápido borrado eléctrico.

Una gran cantidad de aplicaciones requieren manejar señales analógicas, por lo que también es muy importante aprender a utilizar microcontroladores que faciliten este trabajo, para esto se ha incluido el PIC16C71, el cual posee un convertidor analógico digital de cuatro canales.

Otro microcontrolador muy interesante que se incluye en este proyecto es el PIC12C508 que es el primer microcontrolador de 8 pines del mundo. Este presenta enormes facilidades y ventajas para ser utilizado en aplicaciones donde nunca antes se había considerado estos dispositivos.

Además el sistema incorpora los zócalos para trabajar con los nuevos microcontroladores PIC16F87X, los cuales suponen un espectacular avance con respecto a los PIC16F84X, que además de disponer de mayor capacidad de memoria y funcionar a más velocidad con mejor rendimiento, estos microcontroladores poseen incorporados todos los recursos que precisan las aplicaciones comerciales de nivel medio y alto.

Las personas que deseen aprender el manejo de los microcontroladores PIC encontrarán en este sistema de desarrollo una herramienta muy útil y de simple manejo, que incorpora circuitos y prácticas para experimentación con periféricos tales como: teclados, pulsantes, dipswitchs, diodos led, displays de 7 segmentos, display de cristal líquido, pulsadores, relé, zumbador piezoeléctrico, salidas optoacopladas para control de cargas de corriente alterna, entradas analógicas, memoria serial, potenciometro digital, reloj de tiempo real y puertos de comunicaciones seriales RS232, 422 y 485. Obteniendo con esto un conocimiento teórico-práctico en muy corto tiempo, que permita a los estudiantes incorporarse a cientos de ingenieros, que trabajan actualmente en el desarrollo de proyectos basados en microcontroladores.

También se incorporan aspectos sobre el manejo del programa MPLAB que explica la forma de crear un proyecto asociándolo a un archivo fuente para realizar su simulación muy útil en el desarrollo de proyectos; también se explica la forma de usar los comandos generales y la simulación de estímulos externos.

1.2 El PIC16F84A Y PIC16C84

A continuación se explican las características más relevantes de los PIC de las familias PIC16F8X y PIC16C84. Al terminar este numeral el lector tendrá un conocimiento adecuado sobre estos dispositivos.

1.2.1 CARACTERÍSTICAS GENERALES

Los microcontroladores PIC16F8X y PIC16C84 son un grupo extendido de la familia de los mocrocontroladores PIC16CXX de 8 bits. Están incluidos los siguientes microcontroladores:

- PIC16F83
- PIC16F84
- PIC16C84
- PIC16F84A
- PIC16CR83
- PIC16CR84
- Y los de voltaje extendido (PIC16LF8X, PIC16LCR8X)

Estos microcontroladores tienen en común las siguientes características generales:

- Sólo 18 pines, de los cuales 13 son de entrada/salida, con control individual de dirección y capacidad para manejar leds directamente.
- Conjunto de instrucciones RISC (set de instrucciones reducidas) de sólo 35 elementos.
- Memoria de programa de entre 512 y 1024 posiciones, de 14 bits cada una.

- Registros de propósito general de 8 bits cada uno tipo SRAM.

- Frecuencia de oscilador máxima de 10Mhz ó 20Mhz (400ns ó 200ns por instrucción) según la Tabla 1.1 siguiente:

Microcontrolador	Memoria de programa (palabras)	Memoria RAM (bytes)	Memoria EEPROM (bytes)	Frecuencia Máxima (Mhz)
PIC16F83	512 Flash	36	64	10
PIC16F84	1K Flash	68	64	10
PIC16C84	1K EEPROM	36	64	10
PIC16F84A	1K Flash	68	64	20
PIC16CR83	512 ROM	36	64	10
PIC16CR84	1K ROM	68	64	10

Tabla 1.1 Características de memoria y frecuencia

- 15 registros de función especial de hardware.

- Pila de 8 niveles.

- Cuatro fuentes de interrupción:
  - Externa en el pin RB0/INT.
  - Rebase del temporizador TMR0.
  - Interrupción por cambio en los pines PORTB<7:4>.
  - Finalización de escritura en la EEPROM.

- Capacidad de corriente para manejar LEDs directamente.

- Temporizador/Contador de 8 bits con preescalador programable de 8 bits.

- Circuito de vigilancia (Watchdog Timer) incorporado.

- Cuatro opciones para el oscilador.

- Seguridad para protección de código de programa.

- Modo de consumo de baja corriente.
- Programación en paralelo o en serie con sólo dos líneas.
- 1000 ciclos de borrado/escritura para la memoria de programa Flash.
- 10'000.000 de ciclos de borrado/escritura para la memoria de datos EEPROM. excepto en el PIC16F84A que tiene un máximo de 1'000.000
- Reset al encendido mediante dos temporizadores:
  - Temporizador de encendido de 72ms (Power-up Timer).
  - Temporizador de encendido del oscilador (OST).
- Tecnología CMOS de alta velocidad y bajo consumo de potencia.
- Rango de operación de voltaje entre 2.0V a 6.0V, excepto el PIC16F84A cuyo rango está entre 2.0V a 5.5V.

Las diferencias de estos microcontroladores se detallan en la Tabla 1.2.

El PIC16F84A tiene la ventaja de poseer memoria de programa tipo FLASH que en un laboratorio posibilita un rápido desarrollo de prácticas.

Una característica importante es la incorporación de una memoria de datos EEPROM de 64 posiciones, cada una de 8 bits con un tiempo escritura de 4ms. Existen muchas aplicaciones en donde se requiere poder almacenar datos con el fin de que estos permanezcan a disposición para leerlos o modificarlos, aún cuando se presenten cortes de energía o se desconecte el sistema. Esto se solucionaba con el uso de memorias externas tipo RAM alimentadas con baterías de litio o memorias paralelas EEPROM, para lo que era necesario el uso de líneas de control, de datos, de direcciones y espacio adicional.

Este microcontrolador tiene también la capacidad de trabajar a alta velocidad con respecto al resto de su familia, para aplicaciones en donde ello se requiera.

Diferencias	PIC16C84	PIC16F83/F84	PIC16CR83/ CR84	PIC16F84A
Tamaño de la memoria de programa	1k x 14	512 x 14 / 1k x 14	512 x 14 / 1k x 14	1k x 14
Tamaño de la memoria de Datos	36 x 8	36 x 8 / 68 x 8	36 x 8 / 68 x 8	68 x 8
Rango de Voltaje	2.0V - 6.0V (-40°C a +85°C)	2.0V - 6.0V (-40°C a +85°C)	2.0V - 6.0V (-40°C a +85°C)	2.0V - 5.5V (-40°C a +125°C)
Máxima frecuencia de operación	10MHz	10MHz	10MHz	20MHz
Corriente de alimentación IDD	IDD (typ) = 60µA IDD (max) = 400µA (LP osc, FOSC = 32kHz, VDD = 2.0V, WDT inhabilitado)	IDD (typ) = 15µA IDD (max) = 45µA (LP osc, FOSC = 32kHz, VDD = 2.0V, WDT inhabilitado)	IDD (typ) = 15µA IDD (max) = 45µA (LP osc, FOSC = 32kHz, VDD = 2.0V, WDT inhabilitado)	IDD (typ) = 15µA IDD (max) = 45µA (LP osc, FOSC = 32kHz, VDD = 2.0V, WDT inhabilitado)
Corriente en baja Potencia IPD	IPD (typ) = 26µA IPD (max) = 100µA (VDD = 2.0V, WDT inhabilitado, industrial)	IPD (typ) = 0.4µA IPD (max) = 9µA (VDD = 2.0V, WDT inhabilitado, industrial)	IPD (typ) = 0.4µA IPD (max) = 6µA (VDD = 2.0V, WDT inhabilitado, industrial)	IPD (typ) = 0.4µA IPD (max) = 9µA (VDD = 2.0V, WDT inhabilitado, industrial)
Nivel de Entrada de voltaje bajo VIL	VIL (max) = 0.2VDD (Osc1, modo RC)	VIL (max) = 0.1VDD (Osc1, modo RC)	VIL (max) = 0.1VDD (Osc1, modo RC)	VIL (max) = 0.1VDD (Osc1, modo RC)
Nivel Alto de entrada de voltaje VIH	VIH (min) = 0.36VDD (Puertos TTL, I/O 4.5V ≤ VDD ≤ 5.5V)	VIH (min) = 2.4V (Puertos TTL, I/O 4.5V ≤ VDD ≤ 5.5V)	VIH (min) = 2.4V (Puertos TTL, I/O 4.5V ≤ VDD ≤ 5.5V)	VIH (min) = 2.4V (Puertos TTL, I/O 4.5V ≤ VDD ≤ 5.5V)
Tiempo del ciclo de borrado/escritura de la memoria de Datos EEPROM	TDEW (typ) = 10ms TDEW (max) = 20ms	TDEW (typ) = 10ms TDEW (max) = 20ms	TDEW (typ) = 10ms TDEW (max) = 20ms	TDEW (typ) = 4ms TDEW (max) = 10ms

TABLA1.2 Tabla de diferencias entre los microcontroladores PIC16C84, PIC16F83/F84, PIC16CR83/CR84, PIC16F84A

1.2.2 PINES Y FUNCIONES:

El PIC 16F84A es un microcontrolador de Microchip Technology fabricado con tecnología CMOS, su consumo de potencia es muy bajo y además es completamente estático, esto quiere decir que su reloj puede detenerse y los datos de la memoria no se pierden.

El diagrama de pines con sus funciones y numeración se observa en la Figura 1.1

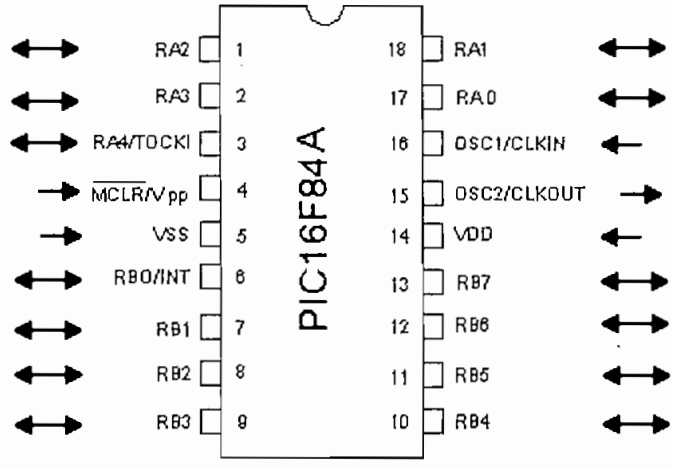


Figura 1.1. Diagrama de pines del PIC16F84A

El encapsulado más común para el microcontrolador es el DIP (Dual In-line Pin) de 18 pines, propio para usarlo en experimentación. La referencia completa es 16F84A-04/P. Sin embargo, hay otros tipos de encapsulado que se pueden utilizar según el diseño y la aplicación que se quiere realizar. Por ejemplo, el encapsulado tipo surface mount (montaje superficial) tiene un reducido tamaño y bajo costo, que lo hace propio para producciones en serie o para utilizarlo en lugares de espacio muy reducido. La Figura 1.2 muestra los tipos de empaque que puede tener el integrado.

Figura 1.2. Tipos de encapsulado

1.2.3 PUERTOS DEL MÍCROCONTROLADOR

Los puertos son el puente entre el microcontrolador y el mundo exterior. Son líneas digitales que trabajan entre cero y cinco voltios y se pueden configurar como entradas o como salidas.

El PIC16F84A tiene dos puertos. El puerto A con 5 líneas y el puerto B con 8 líneas (Figura 1.3.) Cada pin se puede configurar como entrada o como salida independientemente programando un par de registros diseñados para tal fin. En ese registro un "0" configura el pin del puerto correspondiente como salida y un "1" lo configura como entrada.

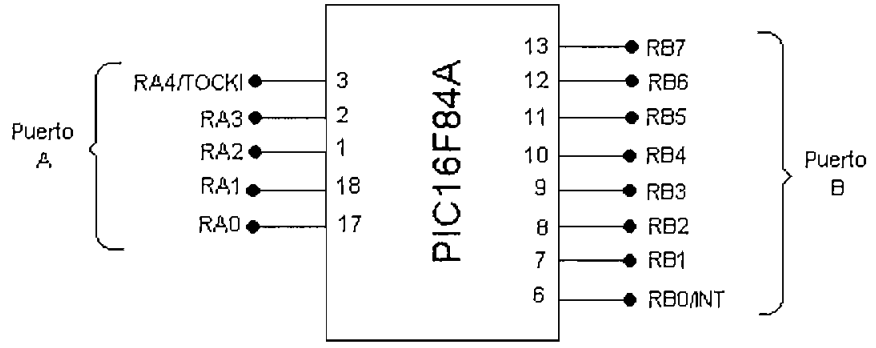


Figura 1.3. Puertos del PIC16F84A

Los pines RA0:RA3 del puerto A tiene Buffers de entrada tipo TTL y drivers de salida tipo CMOS como se indica en la Figura 1.4.

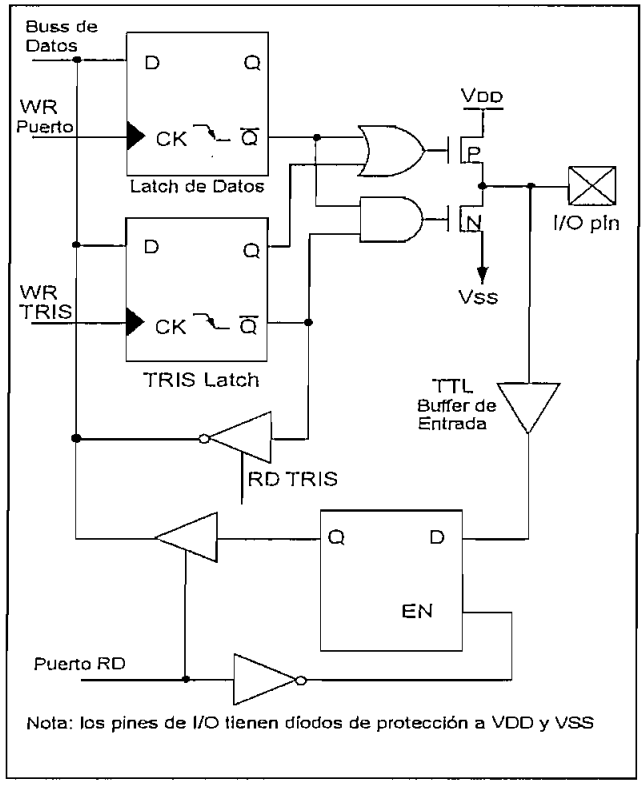


Fig 1.4 Diagrama de bloques de los pines RA3:RA0

El pin RA4/TOCKI del puerto A se muestra en la Figura 1.5 y puede ser configurado como un pin de entrada/salida o como entrada del temporizador/contador. Cuando este pin se programa como entrada digital, funciona como un disparador Schmitt (Schmitt trigger). Puede reconocer señales un poco distorsionadas y llevarlas a niveles lógicos (cero y cinco voltios). Cuando se usa como salida digital se comporta como colector abierto, por lo tanto, se debe poner una resistencia de pull-up (resistencia externa conectada a un nivel de cinco voltios). Como salida, la lógica es inversa: un "0" escrito al pin del puerto entrega en el pin un "1" lógico. Además, como salida no puede manejar cargas como fuente, sólo en el modo sumidero.

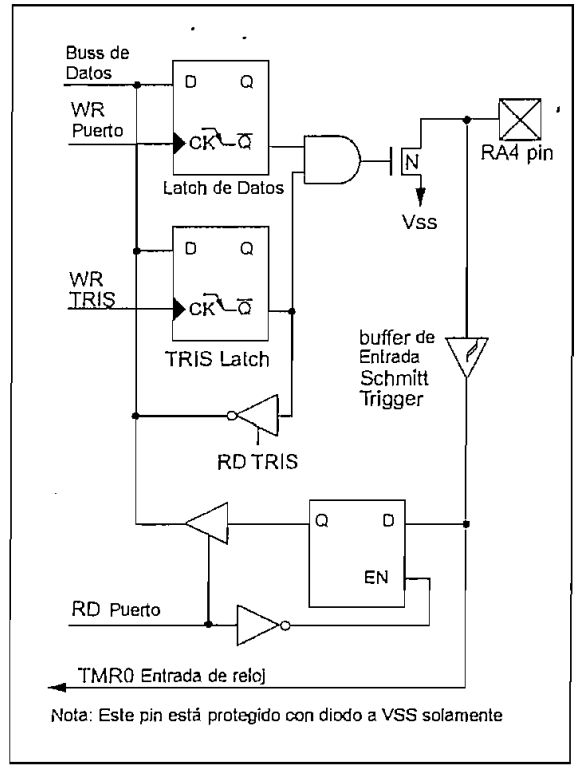


Figura 1.5 Diagrama de Bloques del PIN RA4

El puerto B se muestra en la Figura 1.6 y tiene internamente unas resistencias de pull-up conectadas a sus pines (sirven para fijar el pin a un nivel de cinco voltios), su uso puede ser habilitado o inhabilitado bajo control del programa.

Todas las resistencias pull-up se conectan o se desconectan a la vez, usando el bit llamado RBPU que se encuentra en el registro llamado OPTION.

La resistencia pull-up es desconectada automáticamente en un pin si este se programa como salida.

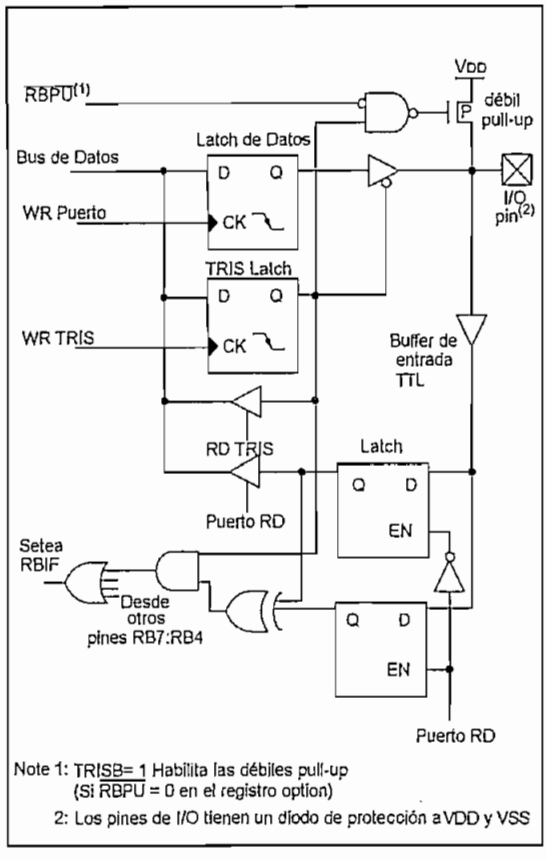


Figura 1.6 Diagrama de bloque de los pines RB7:RB4

Los cuatro pines RB7:RB4 se muestran en la Figura 1.6 y pueden generar una interrupción al detectar un cambio en sus pines. Esto se configura por software utilizando unos bits de los registros INTCON y OPTION. Solo los pines configurados como entradas pueden generar dicha interrupción. El valor de los pines de entrada es comparado con el valor almacenado de la última lectura del puerto B. Esta interrupción puede despertar al microcontrolador del modo SLEEP. Se debe borrar la bandera de la interrupción en la rutina de atención a esta, de cualquiera de las siguientes formas:

a) Leyendo (o escribiendo) sobre el puerto B. Esto finalizará la condición de diferencia.

b) Borrando la bandera bit RBIF.

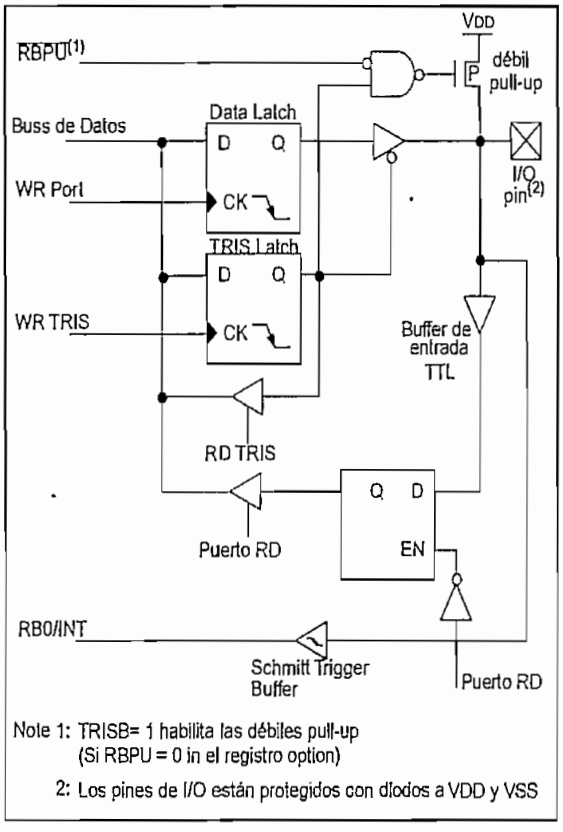


Figura 1.7 Diagrama de Bloque de los pines RB3:RB0

Como este dispositivo es de tecnología CMOS, todos los pines deben estar conectados a alguna parte, nunca dejarlos al aire porque se pueden dañar el integrado.

Los pines que no se estén usando se deben conectar a la fuente de alimentación de +5V, como se muestra en la Figura 1.8.

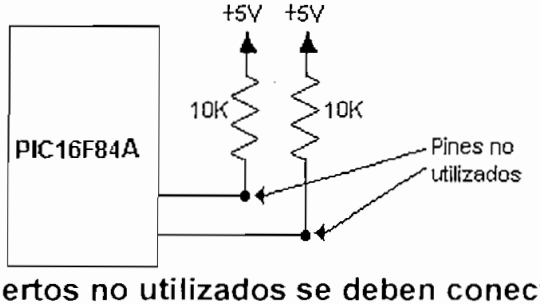


Figura 1.8 Los puertos no utilizados se deben conectar a la fuente

La máxima capacidad de corriente de cada uno de los pines de los puertos en modo sumidero (sink) es de 25 mA y en modo fuente (source) es de 20 mA. La máxima capacidad de corriente total de los puertos es:

Modos	PUERTO A	PUERTO B
Sumidero	80 mA	150mA
Fuente	50 mA	100 mA

Tabla 1.3 Corrientes de consumo de los puertos

El consumo de corriente del microcontrolador depende del voltaje de operación, la frecuencia y de las cargas que tengan sus pines. Para un reloj de 4 MHz el consumo es de aproximadamente 3 mA; aunque este se puede reducir a 40 microamperios cuando se está en el modo sleep (en este modo el micro se detiene y disminuye el consumo de potencia). Se sale de ese estado, cuando se produce alguna condición especial, como por ejemplo una interrupción.

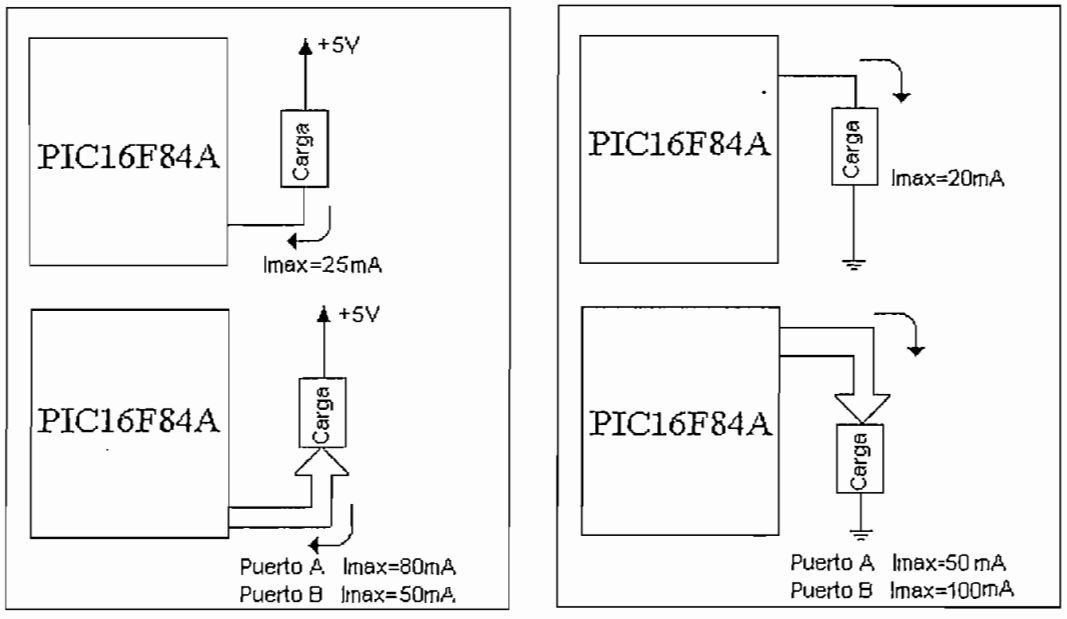


Figura 1.9 Capacidad de corriente del PIC16F84A

1.2.4 El oscilador externo

Todo microcontrolador requiere un circuito externo generador de pulsos que le indique la velocidad a la que debe trabajar. Este circuito, que se conoce como oscilador o reloj, es muy simple pero de vital importancia para el buen funcionamiento del sistema. El PIC16F84A, puede utilizar cuatro tipos de osciladores diferentes, programando los bits de configuración (FOSC1 y FOSC0) para seleccionar uno de estos osciladores. Los tipos de osciladores son:

- **RC**. Oscilador con resistencia y condensador.
- **XT**. Cristal / Resonador
- **HS**. Cristal de alta velocidad / Resonador.
- **LP**. Cristal para baja frecuencia y bajo consumo de potencia.

En el momento de programar o "quemar" el microcontrolador se debe especificar que tipo de oscilador se usa. Esto se hace a través de unos fusibles llamados: "fusibles de configuración".

Para los tipos de cristal XT, LP o HS estos se conectan a los pines de OSC1/CLKIN y OSC2/CLKOUT.

El tipo de oscilador que se sugiere para las prácticas es el cristal de 4 MHz, porque garantiza mayor precisión y un buen arranque del microcontrolador. Internamente esta frecuencia es dividida por cuatro, lo que hace que la frecuencia efectiva de trabajo sea de 1 MHz, por lo que cada instrucción se ejecuta en un microsegundo. El cristal debe ir acompañado de dos condensadores y se conecta como se muestra en la Figura 1.10.

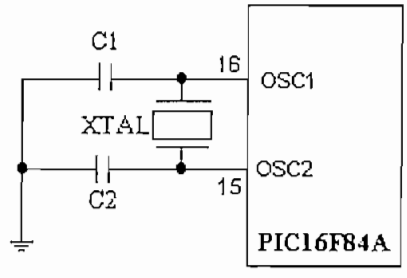


Figura 1.10 Conexión de un oscilador a cristal

Dependiendo de la aplicación, se pueden utilizar cristales de otras frecuencias, por ejemplo se usa el cristal de 3,579545 MHz porque es muy económico, el de 32,768 KHz cuando se necesita crear bases de tiempo de segundo muy precisas. El límite de velocidad en estos microcontroladores es de 20 MHz. Los valores de los condensadores C1 y C2 recomendados para trabajar con osciladores de cristal, se indican en la tabla 1.4 siguiente:

Tipo	Frecuencia	OSC1/C1	OSC2/C2
LP	32 Khz	68 - 100 pF	68 - 100 pF
	200 Khz	15 - 33 pF	15 - 33 pF
XT	100 Khz	100 – 150 pF	100 – 150 pF
	2 Mhz	15 – 33 pF	15 – 33 pF
	4 Mhz	15 – 33 pF	15 – 33 pF
HS	4 Mhz	15 – 33 pF	15 – 33 pF
	10 Mhz	15 – 33 pF	15 – 33 pF

Tabla 1.4 Selección de capacitores para osciladores de cristal

Si no se requiere mucha precisión en el oscilador y se quiere economizar dinero, se puede utilizar una resistencia y un condensador, como se muestra en la Figura 1.11. Los valores recomendados de Rext y Cext son: $5K\Omega \leq Rext \leq 100K\Omega$ y $Cext > 20pF$.

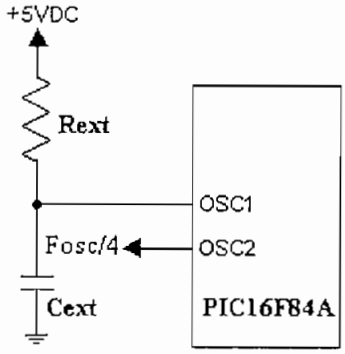


Figura 1.11. Conexión de un oscilador RC

1.2.5 Reset

En los microcontroladores se requiere un pin de reset para reiniciar el funcionamiento del sistema cuando sea necesario, ya sea por una falla que se presente o porque así se halla diseñado el sistema. El pin de reset en los PIC es llamado MCLR (master clear). El PIC 16F84A admite diferentes tipos de reset:

- Al encendido (Power On Reset)
- Pulso en el pin MCLR durante operación normal
- Pulso en el pin MCLR durante el modo de bajo consumo (modo sleep)

• El rebase del conteo del circuito de vigilancia (watchdog) durante operación normal.

• El rebase del conteo del circuito de vigilancia (watchdog) durante el modo de bajo consumo (sleep)

En la Figura 1.12 se muestra el diagrama de bloques del circuito de reset. El MCLR tiene un filtro para ignorar pulsos pequeños, por lo que estos pulsos de reset deben ser grandes.

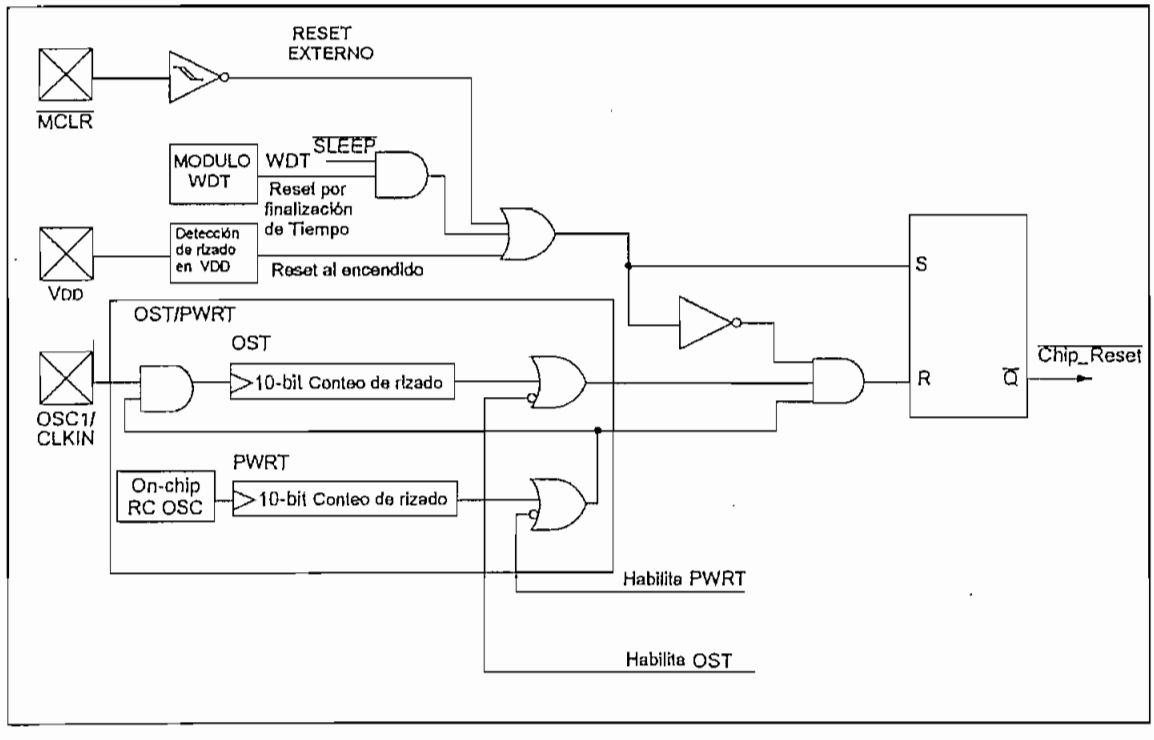


Figura 1.12 Diagrama de bloques del circuito de RESET

Los registros de la memoria RAM del microcontrolador son afectados en distinta forma según el tipo de reset. En la Tabla 1.5 se indican las condiciones de reset para el contador de programa (PC) y el registro STATUS.

Condición	Contador de programa	Registro STATUS
Reset al encendido	000h	0001 1xxx
MCLR Reset durante operación normal	000h	000u uuuu
MCLR Reset durante el modo SLEEP	000h	0001 0uuu
Reset de WDT (en operación normal)	000h	0000 1uuu
Despertar con WDT	PC + 1	uuu0 0uuu
Despertar por interrupción del SLEEP	PC + 1	uuu1 0uuu

Nota: u =no cambia, x =desconocido; Cuando el microcontrolador sale de una condición de reset debido a una interrupción y el bit GIE está en 1 lógico, el contador de programa es cargado con el vector de interrupción.

Tabla 1.5 Condición de reset para el contador de programa y el STATUS

En la Tabla 1.6 de describe los estados de reset para todos los registros.

Registros	Direcciones	Reset al Encendido	Reset en MCLR durante: – Operación normal – SLEEP Reset por el WDT durante operación normal	Al salir de la condición de SLEEP: - Mediante una interrupción - Mediante la finalización del tiempo del WDT
W	—	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF	00h	---- ----	---- ----	---- ----
TMR0	01h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	02h	0000h	0000h	PC + 1(2)
STATUS	03h	0001 1xxx	000q quuu(3)	uuuq quuu(3)
FSR	04h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA(4)	05h	---x xxxx	---u uuuu	---u uuuu
PORTB(5)	06h	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEDATA	08h	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEADR	09h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCLATH	0Ah	---0 0000	---0 0000	---u uuuu
INTCON	0Bh	0000 000x	0000 000u	uuuu uuuu(1)
INDF	80h	---- ----	---- ----	---- ----
OPTION_REG	81h	1111 1111	1111 1111	uuuu uuuu
PCL	82h	0000h	0000h	PC + 1
STATUS	83h	0001 1xxx	000q quuu(3)	uuuq quuu(3)
FSR	84h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TRISA	85h	---1 1111	---1 1111	---u uuuu
TRISB	86h	1111 1111	1111 1111	uuuu uuuu
EECON1	88h	---0 x000	---0 q000	---0 uuuu
EECON2	89h	---- ----	---- ----	---- ----
PCLATH	8Ah	---0 0000	---0 0000	---u uuuu
INTCON	8Bh	0000 000x	0000 000u	uuuu uuuu(1)

Tabla 1.6 Condiciones de reset para todos los registros.

Nota: u =no combia; x = desconocido; - =bit no implementado y se lee con "0";
(1) Uno o más bits del INTCON serán afectados (a causa de despertarse del estado de reposo)
(2) Cuando el microcontrolador sale de una condición de reset debido a una interrupción y el bit GIE está en 1 lógico, el contador de programa es cargado con el vector de interrupción.
(3) En la tabla 1.5 lista las condiciones de reset para cada condición específica.

El reset al encendido se consigue gracias a dos temporizadores.

El primero es el PWRT (Power-Up Timer. Temporizador de encendido) que provee un retardo fijo de 72 ms (nominal) en el encendido únicamente, diseñado para mantener el dispositivo en reset mientras la fuente se estabiliza; este retardo se consigue mediante un oscilador RC interno, y puede ser inhabilitado mediante el bit de configuración PWRET.

El segundo de ellos es el OST (Oscillator Start- Up Timer: Temporizador de encendido del oscilador), que provee un retardo de 1024 ciclos de oscilación, orientado a mantener el microcontrolador en reset hasta que el oscilador del cristal se estabilice. Este temporizador inicia al terminarse el tiempo del temporizador de encendido PWRT. Este retardo es habilitado solo con los osciladores tipo XT, LP y HS.

Para utilizar estos temporizadores, sólo basta con conectar el pin MCLR a la fuente de alimentación a través de una resistencia, evitándose utilizar las tradicionales redes RC externas en el pin de reset.

El reset por MCLR se consigue llevando momentáneamente este pin a un estado lógico bajo, mientras que el watchdog WDT produce el reset cuando su temporizador rebasa la cuenta, o sea que pasa de 0FFh a 00h. Cuando se quiere tener control sobre el reset del sistema se puede conectar un botón como se muestra en la Figura 1.13.

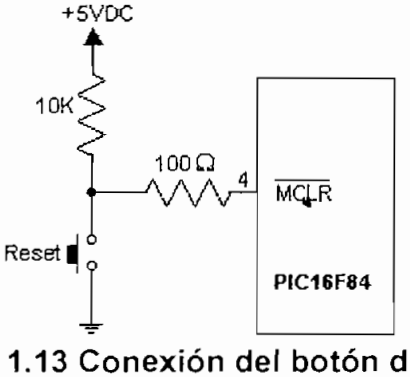


Figura 1.13 Conexión del botón de reset

1.2.6 Arquitectura

Este término se refiere a los bloques funcionales internos que conforman el microcontrolador y la forma en que están conectados, por ejemplo la memoria FLASH (de programa), la memoria RAM (de datos), los puertos, la lógica de control que permite que todo el conjunto funcione, etc. La Figura 1.14 muestra la arquitectura general del PIC16F84A, en ella se pueden apreciar los diferentes bloques que lo componen y la forma en que se conectan. Se muestra la conexión de los puertos, las memorias de datos y de programa, los bloques especiales como el watchdog, los temporizadores de arranque, el oscilador, etc.

Todos los elementos se conectan entre sí por medio de buses. Un bus es un conjunto de líneas que transportan información entre dos o más módulos. Vale la pena destacar que el PIC16F84A tiene un bloque de memoria de datos y registros (RAM) de 68 bytes, una memoria EEPROM de 64 bytes, además del bloque de memoria de programa de 1K de 14bits.

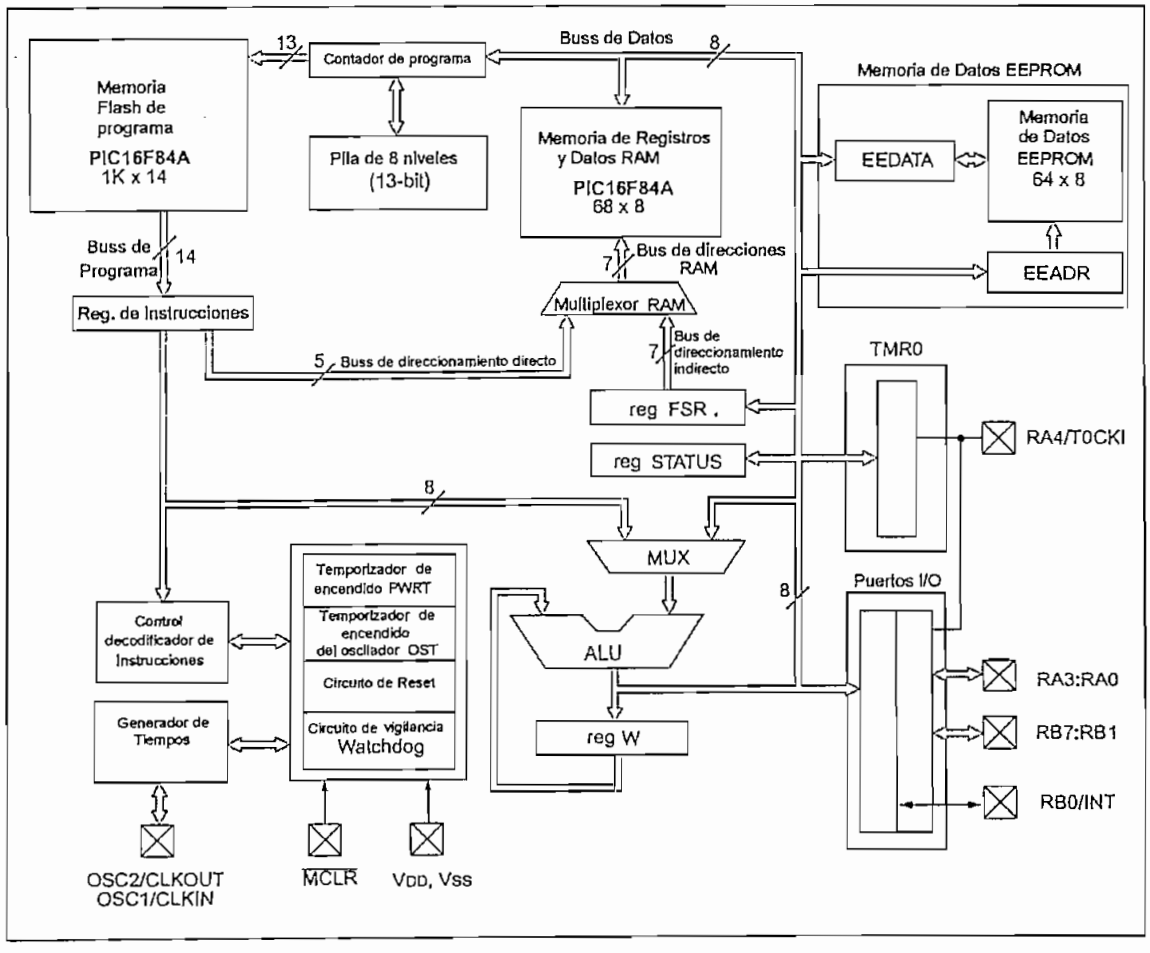


Fig. 1.14 Arquitectura interna del PIC16F84A

El PIC16F84A se basa en la arquitectura Harvard, en la cual el programa y los datos se pueden trabajar desde memorias separadas, lo que posibilita que las instrucciones y los datos posean longitudes diferentes. Esta misma estructura es la que permite la superposición de los ciclos de búsqueda y ejecución de las instrucciones, lo cual se ve reflejado en una mayor velocidad del microcontrolador.

1.2.7 Memoria de programa

Es una memoria de 1 Kbyte de longitud con palabras de 14 bits. Como es del tipo FLASH se puede programarla y borrarla eléctricamente, lo que facilita el desarrollo de los programas y la experimentación. En ella se graba, o almacena el programa o códigos que el microcontrolador debe ejecutar. Dado que el PIC16F84A tiene un contador de programa de 13 bits, tiene una capacidad de direccionamiento de 8K x 14, pero solamente tiene implementado el primer 1K x 14 (0000h hasta 03FFh). Si se direccionan posiciones de memoria superiores a 3FFh se causará un solapamiento con el espacio del primer 1K. En la Figura 1.15 se muestra el mapa de la memoria de programa.

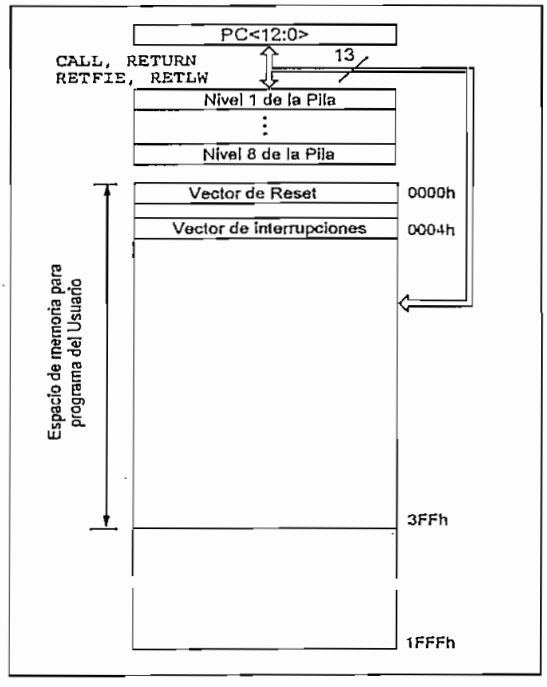


Figura 1.15. Mapa de la memoria de programa

Vector de reset.

Cuando ocurre un reset al microcontrolador, el contador de programa se pone en ceros (000H). Por esta razón, en la primera dirección del programa se debe escribir todo lo relacionado con la iniciación del mismo.

Vector de interrupción.

Cuando el microcontrolador recibe una señal de interrupción, el contador de programa apunta a la dirección 04H de la memoria de programa, por eso, allí se debe escribir toda la programación necesaria para atender dicha interrupción.

1.2.8 Registros (Memoria RAM)

El PIC16F84A puede direccionar 128 posiciones de memoria RAM, pero solo tiene implementados físicamente los primeros 80 (0-4F en hexadecimal).

De estos los primeros 12 son registros que cumplen un propósito especial en el control del microcontrolador y los 68 siguientes son registros de uso general que se pueden usar para guardar los datos temporales de la tarea que se está ejecutando (Figura 1.16).

Los registros están organizados como dos arreglos (páginas) de 128 posiciones de 8 bits cada una (128 x 8).

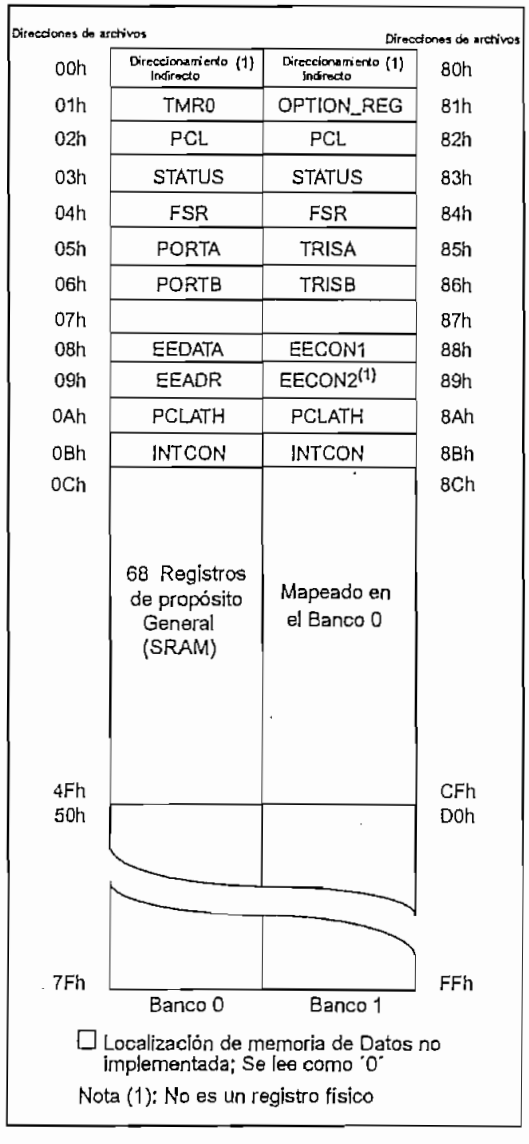


Figura. 1.16 Mapa del archivo de registros PIC16F84A

Archivo de registros de propósito general (GPR)

Cada registro de propósito general tiene 8 bits de ancho y son accesadas mediante direccionamiento directo e indirecto a través del registro selector de registros FSR.

Los GPR direccionados en el banco 1 son mapeados en el banco 0. Como un ejemplo al leer la dirección 0Ch o 8Ch se accesará al mismo GRP.

Registros de función especial.

Los registros de función especial son usados por la CPU y funciones de periféricos para controlar las operaciones del sistema. Estos registros son RAM estática. Los registros de función especial con la identificación de sus bits respectivos son indicados a continuación en la Tabla 1.7.

Dirección	Nombre	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valor despues del reset de encendido	Valor despues de todos los otros reset (Nota 3)
Banco 0											
00h	INDF	Contenido usado por FSR para la dirección de memoria de datos (no es un registro físico)								---- ----	---- ----
01h	TMR0	Reloj de tiempo real / contador de 8 bits								xxxx xxxx	uuuu uuuu
02h	PCL	8 bits bajos del contador de programa PC								0000 0000	0000 0000
03h	STATUS (2)	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx	000q quuu
04h	FSR	Dirección indirecta de memoria de Datos, puntero del INDF								xxxx xxxx	uuuu uuuu
05h	PORTA (4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06h	PORTB (5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07h		Localidad no implementada, se lee con '0'								---- ----	---- ----
08h	EEDATA	Registro de datos de la memória EEPROM								xxxx xxxx	uuuu uuuu
09h	EEADR	Registro de direcciones de la memória EEPROM								xxxx xxxx	uuuu uuuu
0Ah	PCLATH	—	—	—	Buffer de escritura de los 5 bits altos del PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
Bank 1											
80h	INDF	Contenido usado por FSR para la dirección de memoria de datos (no es un registro físico)								---- ----	---- ----
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	8 bits bajos del contador de programa PC								0000 0000	0000 0000
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
84h	FSR	Dirección indirecta de memoria de Datos, puntero del INDF								xxxx xxxx	uuuu uuuu
85h	TRISA	—	—	—	Registro de configuración del puerto A					---1 1111	---1 1111
86h	TRISB	Registro de configuración del puerto B								1111 1111	1111 1111
87h		Localidad no implementada, se lee con '0'								---- ----	---- ----
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	Registro 2 de control de la memoria EEPROM (no es un registro físico)								---- ----	---- ----
0Ah	PCLATH	—	—	—	Buffer de escritura para los 5 bits altos del PC					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

Tabla 1.7 Registros de función especial

00h ó INDF: Registro para direccionamiento indirecto de datos.

Este no es un registro disponible físicamente; utiliza el contenido del FSR y el bit RP0 del registro STATUS para seleccionar indirectamente la memoria de datos o RAM del usuario; la instrucción determinará que se debe realizar con el registro señalado.

01h ó TMR0. Temporizador / contador de 8 bits.

Este registro puede operar como temporizador o como contador. El modo temporizador es seleccionado al escribir un ´1´ en el bit TOSC del registro OPTION. En este caso el registro se incrementará con cada ciclo de instrucción (según el preescalador). Cuando se escribe sobre el registro TMR0, el conteo es inhibido para los dos siguientes ciclos de instrucción. El modo contador es seleccionado al escribir un ´0´ en el bit TOSC del registro OPTION. En este caso el registro se incrementará mediante una señal externa aplicada al pin RA4/TOCKI y puede ser seleccionado para actuar con los flancos de subida o con los de bajada. El flanco de subida o bajada es seleccionado mediante el bit TOSE del registro OPTION. En este caso se requiere que el reloj externo sea sincronizado con la fase del reloj interno (TOSC). El diagrama de bloques del TMR0 se muestra en la Figura 1.17 mostrada a continuación.

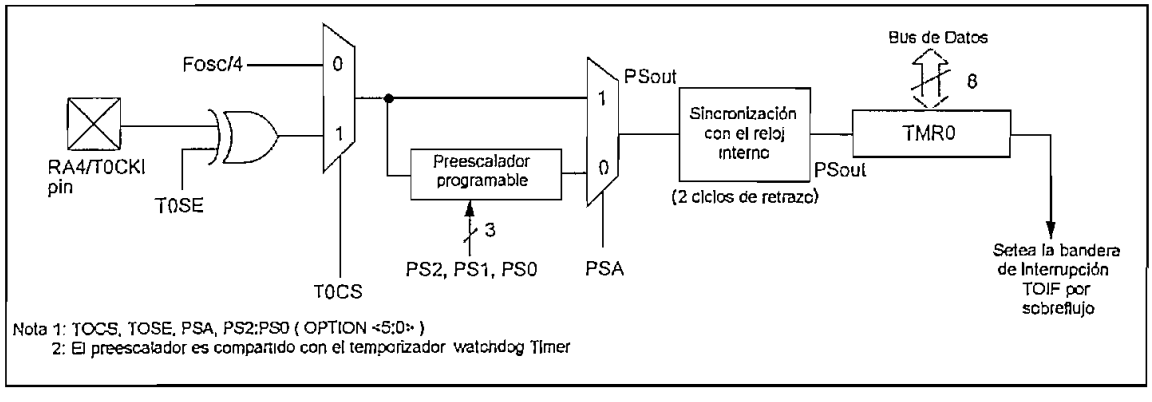


Figura 1.17 Diagrama de bloques del TIMER0

Preescalador.

Un contador de 8 bits sirve para preescalar el temporizador TMR0 o el Wacthdog Timer, este preescalador puede ser usado para uno de los dos temporizadores pero no para los dos. El preescalador no es leíble ni escribible. Un diagrama de bloques del preescalador del TIMER0/WDT se muestra en la Figura 1.18.

Los bits PSA y PS2:PS0 del registro OPTION determinan la asignación y nivel de preescalación.

Con PSA=0 asignará el preescalador al TIMER0 y en este caso el preescalador

toma los valores de 1:2, 1:4, ...1:256.

Con PSA=1 asignará el preescalador al Watchdog Timer y en este caso el preescalador toma los valores de 1:1, 1:2, ...1:128.

Cuando se asigna el preescalador al TIMER0, las instrucciones de escritura sobre el TMR0 borrarán el conteo del preescalador y cuando el preescalador es asignado al WDT la instrucción CLRWDT borrará el conteo del preescalador; pero estas operaciones no cambiarán la asignación dada al preescalador.

Además, la asignación del preescalador es completamente controlado por software. Esto permite que se pueda cambiar la asignación del preescalador durante la ejecución del programa.

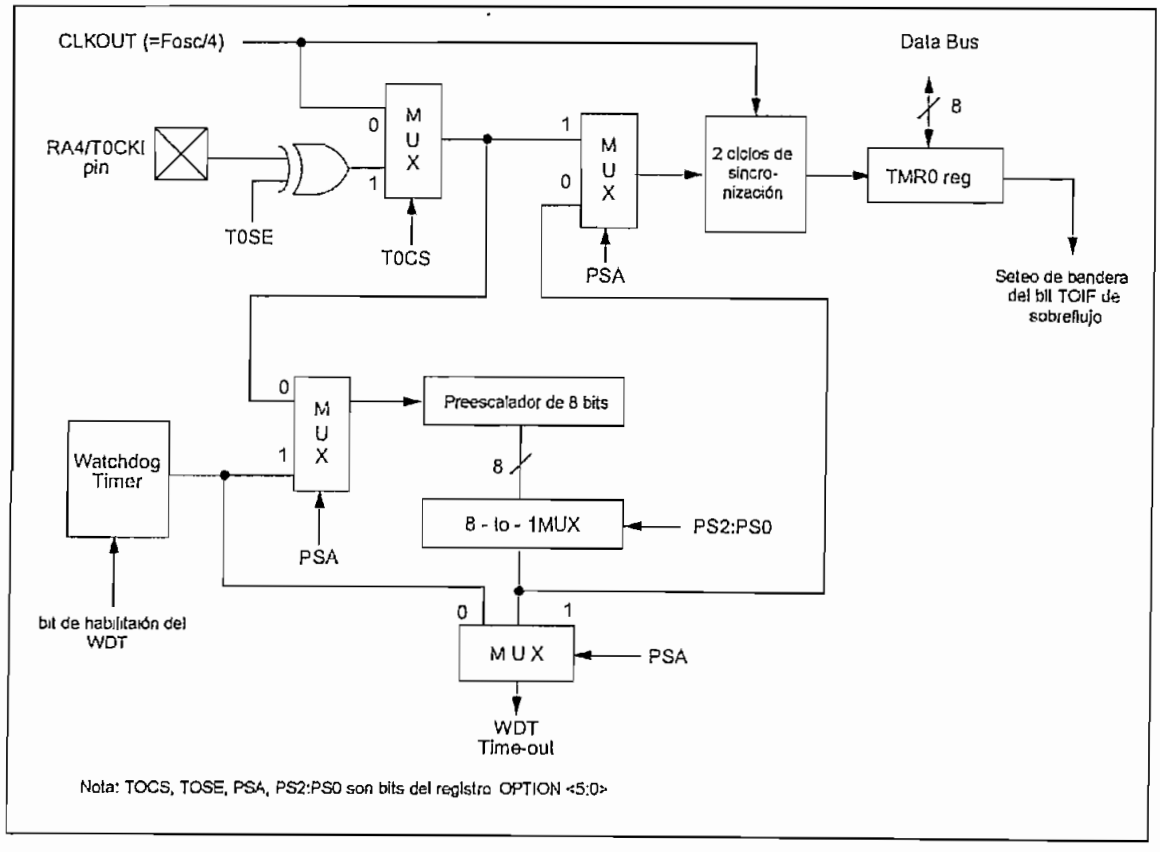


Figura. 1.18 Diagrama de bloques del preescalador del TIMER0/WDT

02h o PCL : Contador de programa.

Se utiliza para direccionar las palabras de 14 bits del programa del usuario que se encuentra almacenado en la memoria ROM; es decir, contiene la dirección de la instrucción a ser ejecutada; este contador de programa es de 13 bits de ancho (Figura 1.19). Sobre el byte bajo (PCL), se puede escribir o leer

directamente, mientras que sobre el byte alto, no. El byte alto se maneja mediante el registro PCLATH (0Ah). A diferencia de los PIC de primera generación, el 16F84A ante una condición de reset inicia el contador de programa con todos sus bits en "cero". Durante la ejecución normal del programa, y dado que todas las instrucciones ocupan sólo una posición de memoria, el contador se incrementa en uno con cada instrucción, a menos que se trate de alguna instrucción de salto.

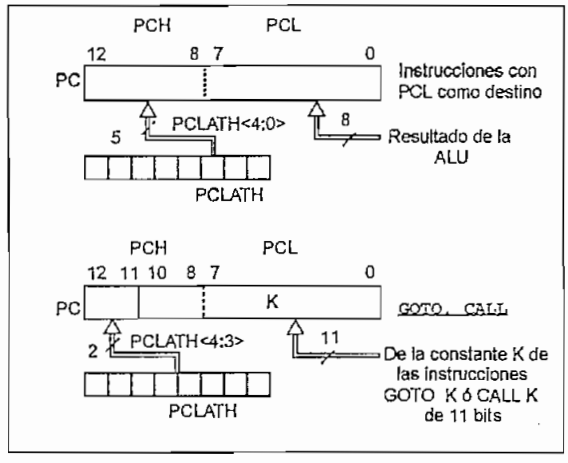


Figura 1.19. Contador de programa (13 bits)

En una instrucción CALL o GOTO, los bits PC <I0:0> se cargan desde el código de operación de la instrucción, mientras que los bits PC <12:11> lo hacen desde el PCLATH<4:3>. Como solamente el primer 1K de memoria está implementado, el código de operación de la instrucción puede contener la dirección destino, eso quiere decir que se pueden hacer saltos y llamados a subrutinas sin necesidad de tener en cuenta la paginación de memoria de programa.

En otras instrucciones donde PCL es el destino, PC<12:8> se carga directamente desde el PCLATH<4:0>, por ejemplo, en el caso de la instrucción ADDWF. Esto se debe tener en cuenta cuando se desea hacer lectura de tablas usando el comando: ADDWF PC, 1; en este caso la tabla debe estar comprendida dentro de un solo bloque de 256 bytes (0-255, 256-511, etc.).

O3h ó STATUS: Registro de estados.

Contiene el estado aritmético de la ALU, la causa del reset y los bits de preselección de página para la memoria de datos. La Figura 1.20 *muestra* los bits correspondientes a este registro. Los bits 5 y 6 (RPO y RP1) son los bits de

selección de página para el direccionamiento directo de la memoria de datos; solamente RPO se usa en el PIC16F84A. RP1 se puede utilizar como un bit de propósito general de lectura/escritura. Los bits TO y PD no se pueden modificar por un proceso de escritura; ellos muestran la condición por la cual se ocasionó el último reset.

Se debe tener en cuenta que al escribir sobre el registro STATUS no se obtendrá posiblemente el resultado esperado, pues sus bits cambian según el resultado de las operaciones, por ejemplo con la instrucción CLRF STATUS se borrarán los tres bits altos y el bit Z se seteará.

Los bits IRP y RP1 no son utilizados en el PIC16F84A y no se recomienda utilizarlos como bits de propósito general, pues a futuro podría generar incompatibilidades al migrar a otros microcontroladores.

El bit C y DC operan como un "llevo" en las operaciones de resta, esto se puede entender al revisar las instrucciones SUBLW y SUBWF.

Cuando el registro STATUS es el destino de una instrucción que afecte a los bits Z, DC o C, entonces la escritura de estos tres bits se inhabilita. Estos bits se actualizarán de acuerdo a la lógica del sistema.

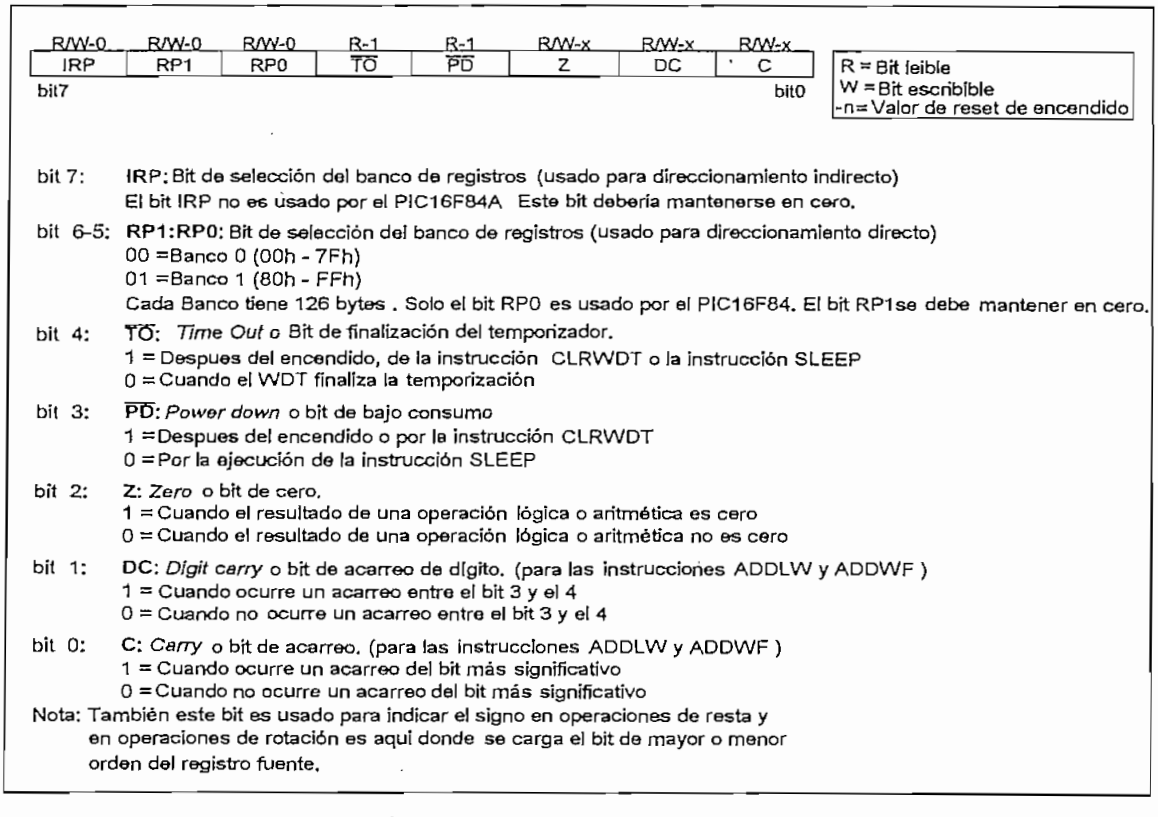


Figura 1.20 Registro de estados

04h o FSR: Registro selector de registros.

En asocio con el registro INDF se utiliza para seleccionar indirectamente los otros registros disponibles, como puntero para direccionar indirectamente la memoria de datos. Si en el programa no se utilizan llamadas indirectas, este registro se puede utilizar como un registro de propósito general. Para entender mejor el funcionamiento del registro INDF, obsérvese el resultado de leer dos registros RAM (05h y 06h) que contienen los valores 10h y 0Ah utilizando direccionamiento indirecto.

- El registro 05 contiene el valor 10h
- El registro 06 contiene el valor 0Ah
- Se carga el valor 05 al registro FSR
- Entonces al leer el registro INDF se obtendrá el valor 10h
- Se incrementa en uno el registro FSR (FSR = 06)
- Entonces al leer el registro INDF ahora se obtendrá el valor 0Ah

Leyendo indirectamente el mismo registro INDF (FSR = 0) se obtendrá 00h. Escribiendo indirectamente en el registro INDF resulta en una operación no efectuada, pues los bits del registro STATUS pueden ser afectados. Otro ejemplo simple de un programa para borrar las localidades de 20h – 2Fh usando direccionamiento indirecto es mostrado a continuación:

```
            movlw   20h     ;inicializa el puntero
            movwf   FSR     ;de la memoria RAM
NEXT        clrf    INDF    ;borra el registro INDF
            incf    FSR     ;incrementa el puntero
            btfss   FSR,4   ;¿ya termino el borrado?
            goto    NEXT    ;NO,  borra el siguiente registro
CONTINUA    .               ;SI, continua con la ejecución del
            .               ;programa restante
```

El bit IRP del registro STATUS funciona como el noveno bit del registro FSR, como se muestra en la Figura 1.21. Sin embargo este bit no es usado en el PIC16F84A.

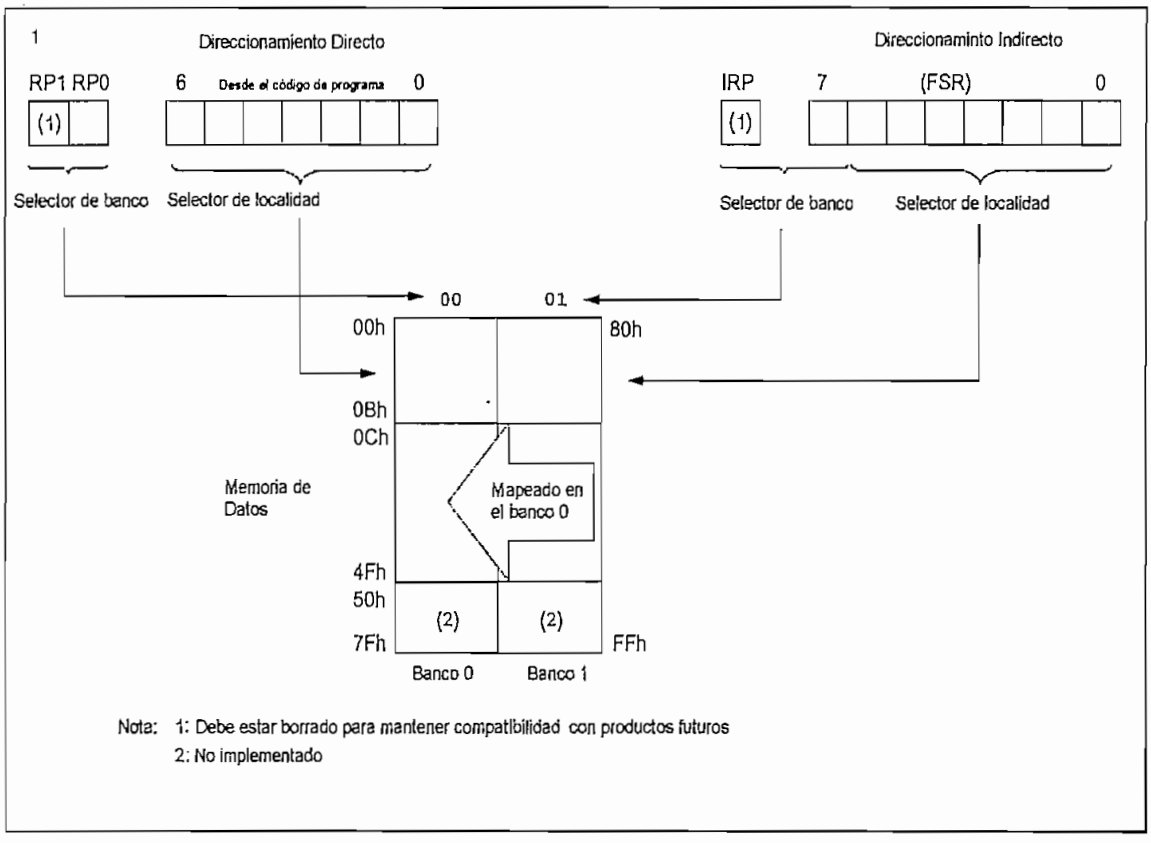


Figura. 1.21 Direccionamiento directo e indirecto

05h o PORTA: Puerto de Entrada/Salida de 5 bits.

Este puerto tiene 5 bits de ancho y al igual que todos sus similares en los PIC, puede leerse o escribirse como si se tratara de un registro cualquiera. El registro que controla el sentido (entrada o salida) de los pines de este puerto, está localizado en la página 1, en la posición 85h y se llama TRISA. Luego de un reset de encendido, estos pines son configurados como entradas y se leen con cero. Al realizar una operación de lectura se lee el estado del pin, y al realizar una operación de escritura se carga en la memoria *(latch)* del puerto.

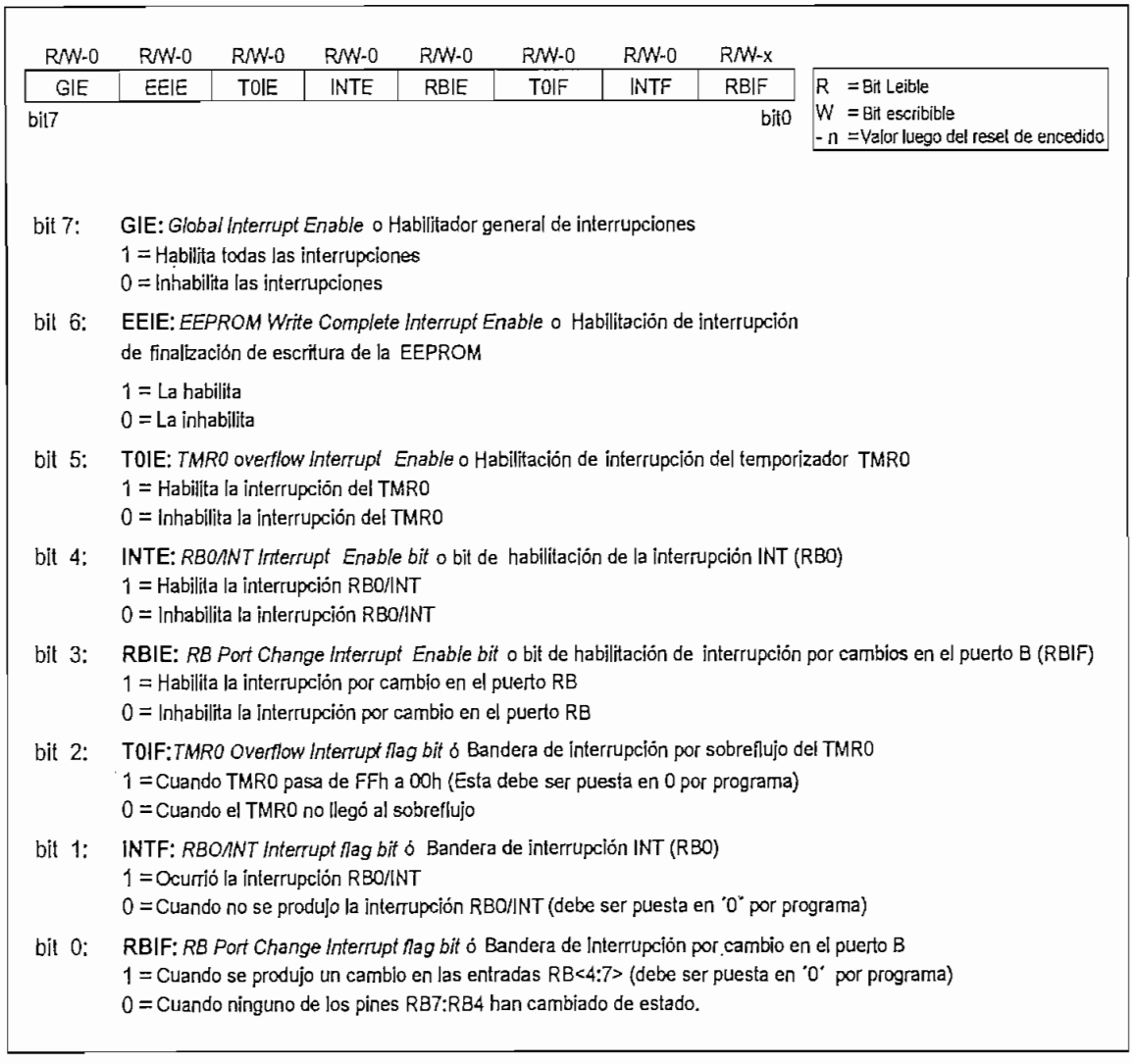
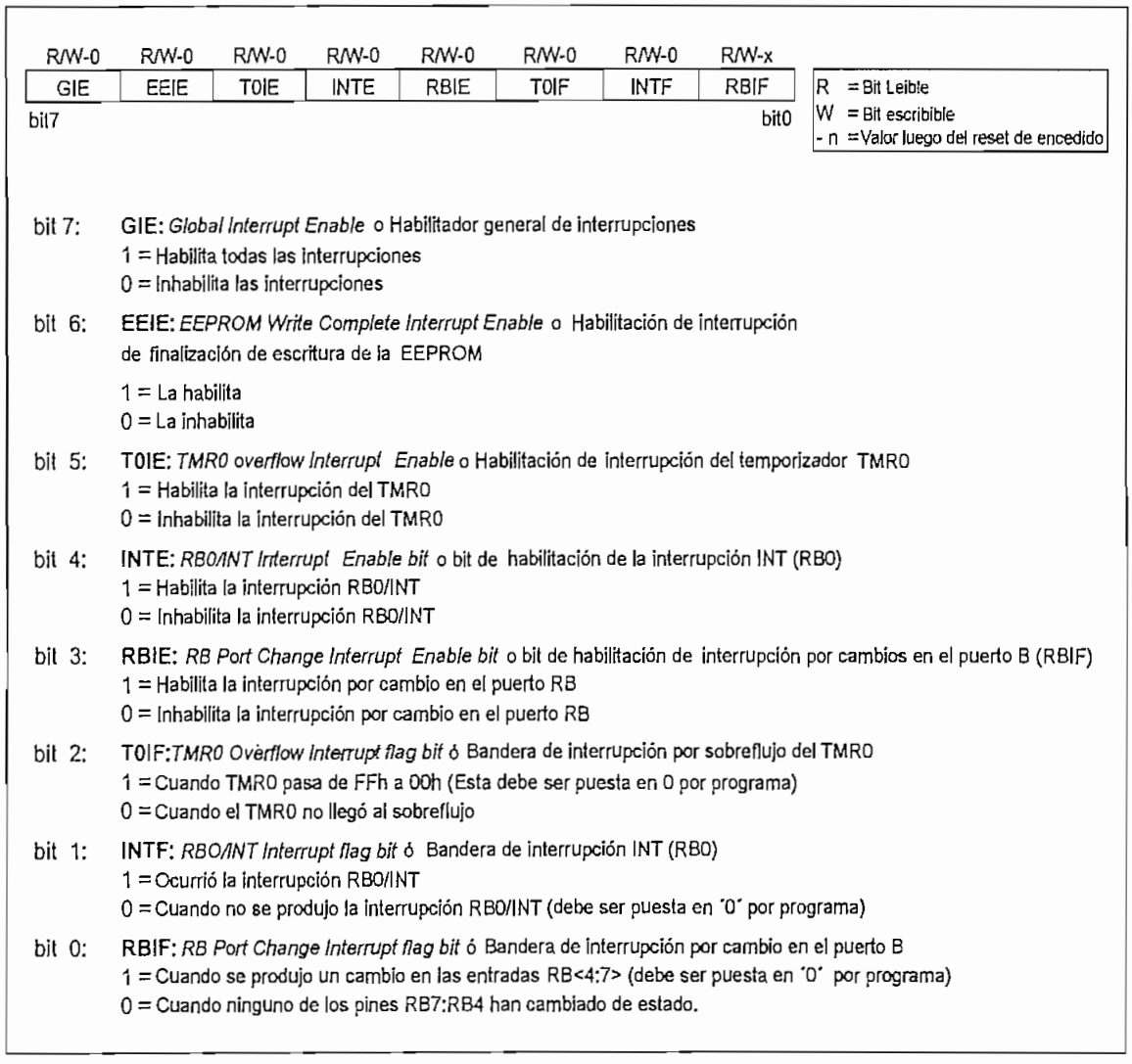


Figura 1.22. Registro INTCON (Direcciones 0Bh, 8Bh)

81h u OPTION: Registro de configuración múltiple.

Este registro es leíble y escribible. Posee varios bits para configurar el preescalador TMR0/WDT, la interrupción externa INT, el temporizador TMR0 y las pull-ups del puerto B. Los bits que contiene y las funciones que realiza este registro se muestran en la Figura 1.23. El preescalador es compartido entre el TMRO y el WDT; su asignación es mutuamente excluyente ya que solamente puede uno de ellos ser preescalado a la vez.



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	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							bit0

R = Bit Leible
W = Bit escribible
- n = Valor luego del reset de encedido

bit 7: **GIE:** *Global Interrupt Enable* o Habilitador general de interrupciones
1 = Habilita todas las interrupciones
0 = Inhabilita las interrupciones

bit 6: **EEIE:** *EEPROM Write Complete Interrupt Enable* o Habilitación de interrupción de finalización de escritura de la EEPROM
1 = La habilita
0 = La inhabilita

bit 5: **T0IE:** *TMR0 overflow Interrupt Enable* o Habilitación de interrupción del temporizador TMR0
1 = Habilita la interrupción del TMR0
0 = Inhabilita la interrupción del TMR0

bit 4: **INTE:** *RB0/INT Interrupt Enable bit* o bit de habilitación de la interrupción INT (RB0)
1 = Habilita la interrupción RB0/INT
0 = Inhabilita la interrupción RB0/INT

bit 3: **RBIE:** *RB Port Change Interrupt Enable bit* o bit de habilitación de interrupción por cambios en el puerto B (RBIF)
1 = Habilita la interrupción por cambio en el puerto RB
0 = Inhabilita la interrupción por cambio en el puerto RB

bit 2: **T0IF:***TMR0 Overflow Interrupt flag bit* ó Bandera de interrupción por sobreflujo del TMR0
1 = Cuando TMR0 pasa de FFh a 00h (Esta debe ser puesta en 0 por programa)
0 = Cuando el TMR0 no llegó al sobreflujo

bit 1: **INTF:** *RB0/INT Interrupt flag bit* ó Bandera de interrupción INT (RB0)
1 = Ocurrió la interrupción RB0/INT
0 = Cuando no se produjo la interrupción RB0/INT (debe ser puesta en '0' por programa)

bit 0: **RBIF:** *RB Port Change Interrupt flag bit* ó Bandera de interrupción por cambio en el puerto B
1 = Cuando se produjo un cambio en las entradas RB<4:7> (debe ser puesta en '0' por programa)
0 = Cuando ninguno de los pines RB7:RB4 han cambiado de estado.

Figura 1.22. Registro INTCON (Direcciones 0Bh, 8Bh)

81h u OPTION: Registro de configuración múltiple.

Este registro es leíble y escribible. Posee varios bits para configurar el preescalador TMR0/WDT, la interrupción externa INT, el temporizador TMR0 y las pull-ups del puerto B. Los bits que contiene y las funciones que realiza este registro se muestran en la Figura 1.23. El preescalador es compartido entre el TMRO y el WDT; su asignación es mutuamente excluyente ya que solamente puede uno de ellos ser preescalado a la vez.

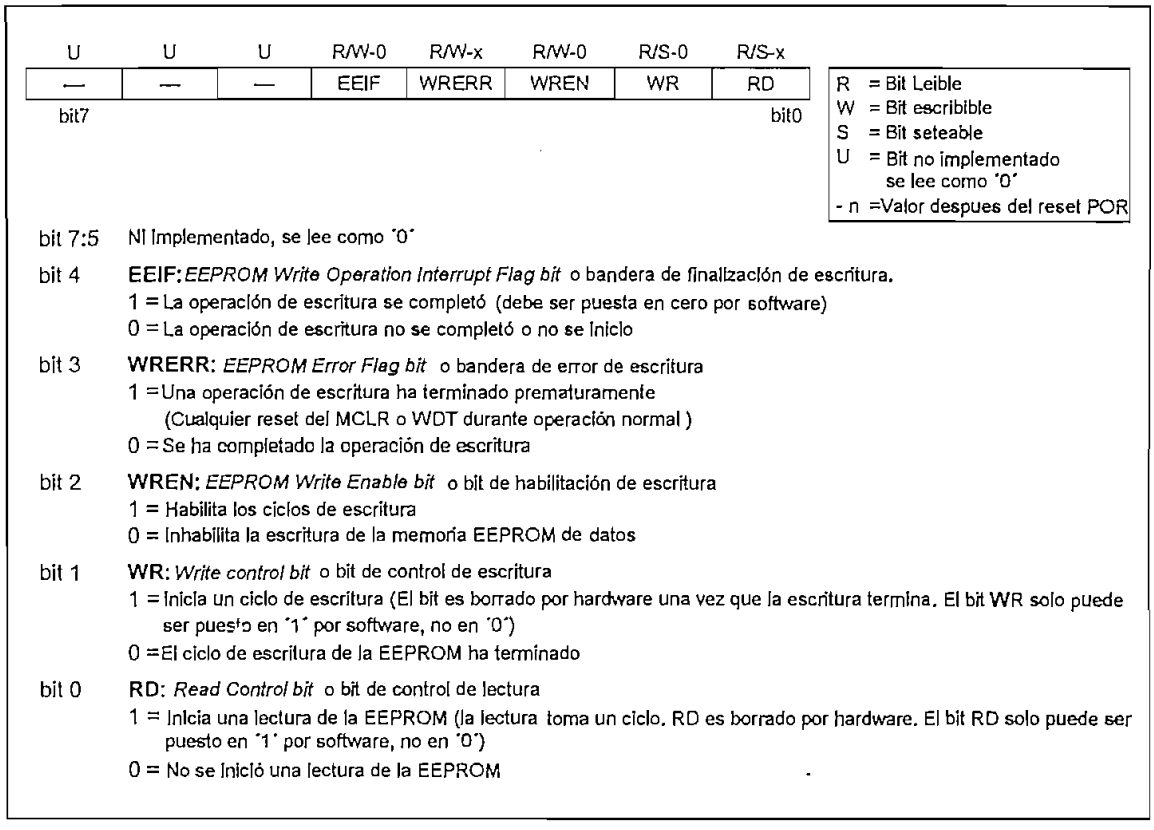


Figura. 1.24 Registro EECON1 (Dirección 88h)

89h o EECON2: Registro auxiliar para control de la memoria EEPROM.

Registro que no está implementado físicamente en el microcontrolador, pero que es necesario en las operaciones de escritura en la EEPROM de datos; ante cualquier intento de lectura se obtendrán "ceros".

OCh a 4Fh: Registros de propósito general.

Estas 68 posiciones están implementadas en la memoria RAM estática, que conforma el área de trabajo del usuario; a ellas también se accede cuando en la página 1 se direccionan las posiciones 8Ch a CFh. Esto se ha diseñado así para evitar un excesivo cambio de páginas en el manejo de la RAM del usuario, agilizando los procesos que se estén llevando a cabo y descomplicando la labor de programación.

1.2.9 Registro de trabajo W.

Este es el registro de trabajo principal, se comporta de manera similar al acumulador en los microprocesadores. Este registro participa en la mayoría de las instrucciones.

1.2.10 Pila (Stack)

Estos registros no forman parte de ningún banco de memoria y no permiten el acceso por parte del usuario. Se usan para guardar el valor del contador de programa cuando se hace un llamado a una subrutina o cuando se atiende una interrupción; luego, cuando el micro regresa a seguir ejecutando su tarea normal, el contador de programa recupera su valor leyéndolo nuevamente desde la pila. El PIC16F84A tiene una pila de 8 niveles, esto significa que se pueden anidar 8 llamados a subrutina sin tener problemas, y al igual que el contador de programa tiene 13 bits de ancho.

1.2.11 Características especiales

Algunos elementos que forman parte de los PIC no se encuentran en micro-controladores de otros fabricantes, y representan alguna ventaja o facilidad a la hora de hacer un diseño. Estas características son muy útiles para lograr equipos de alta calidad y mínimo costo a través de la eliminación de componentes externos, por esto se debe aprender a utilizarlos. A continuación se detallan dichas características:

Circuito de vigilancia (Watchdog Timer WDT)

Su función es restablecer el programa cuando éste se ha perdido por fallas en la programación o por alguna razón externa. Es muy útil cuando se trabaja en ambientes con mucha interferencia o ruido electromagnético. Está conformado por un oscilador RC que se encuentra dentro del microcontrolador. Este oscilador corre de manera independiente al oscilador principal. Cuando se habilita su funcionamiento poniendo un ´1´ en el bit de configuración WDTE , dicho circuito hace que el microcontrolador sufra un reset cada determinado tiempo (que se puede programar entre 18 ms y 2,3 sg mediante el preescalador). Este reset lo puede evitar el usuario mediante una instrucción especial del microcontrolador (**CLRWDT:** borrar el conteo del watchdog), la cual se debe ejecutar antes de que termine el período nominal de dicho temporizador. De esta manera, si el programa se ha salido de su flujo normal, por algún ruido o interferencia externa, el sistema se reiniciará (cuando se acabe el tiempo programado y no se haya borrado el

contador) y el programa puede restablecerse para continuar con su funcionamiento normal. El diagrama de bloques del WDT se puede observar en la figura del preescalador del TIMER0/WDT.

Temporizador de encendido (Power-up Timer)

Este proporciona un reset al microcontrolador en el momento de conectar la fuente de alimentación, lo que garantiza un arranque correcto del sistema. En el momento de grabar el microcontrolador se debe habilitar el fusible de configuración "Power-up Timer", para ello se debe seleccionar la opción "ON". Su tiempo de retardo es de 72 milisegundos.

Modo de bajo consumo (sleep)

Esta característica permite que el microcontrolador entre en un estado pasivo donde consume muy poca potencia; y posteriormente, salir de este estado a operación normal. Cuando entra al modo de bajo consumo y el temporizador del circuito de vigilancia (*Watchgod Timer)* está habilitado, este temporizador se reinicia y empieza el conteo de nuevo, El bit PD es borrado y el bit TO es puesto en ´1´ y se detiene el oscilador principal, deteniendo la ejecución de instrucciones. Los puertos de entrada/salida mantienen la configuración que tenían antes de la ejecución de la instrucción SLEEP.

Para un muy bajo consumo de corriente en el modo SLEEP, se debe colocar a VDD o VSS todos los pines que no tengan drenaje de corriente a un circuito externo desde el pin, y se debe inhabilitar el reloj externo.

Los pines que son entradas de alta impedancia deben ser colocados a un nivel alto o bajo, externamente, para evitar corrientes de oscilación causadas por entradas flotantes. La entrada TOCKI también debe ser conectada a VDD o VSS. También se debe considerar a las resistencias *pull-up* internas del puerto B.

El pin MCLR debe estar colocado a un nivel lógico alto, tomando en cuenta que el Reset del WDT por finalización de temporización no funciona cuando el MCLR está conectado a un nivel lógico bajo.

El microcontrolador sale del modo de bajo consumo SLEEP por una de las siguientes causas:

- Reset externo en el pin MCLR
- Finalización del conteo del temporizador WDT
- Interrupción en el pin RB0/INT, cambio en el puerto RB, o finalización de escritura en la memoria EEPROM.

Los bits TO y PD pueden ser usados para determinar la causa del reset del equipo. El bit PD se pone en ´0´ cuando entra en el modo SLEEP y al salir de este, es colocado en ´1´ . El bit TO es puesto en ´0´ si se finaliza el conteo del temporizador de vigilancia WDT (y causa que salga del modo de reposo SLEEP). Mientras la instrucción SLEEP está siendo ejecutada, la siguiente instrucción (PC+1) es leída. Para que un evento de interrupción saque al sistema del reposo, el correspondiente bit de habilitación de interrupciones (GIE) debe ser puesto en ´1´. Las operaciones posteriores al salir del modo de reposo dependen del bit GIE. Si el bit GIE está en ´0´; es decir, inhabilitadas las interrupciones, entonces el equipo continuará ejecutando la instrucción posterior al SLEEP. Si el bit GIE está en ´1´; es decir, habilitadas las interrupciones, entonces el equipo ejecutará la instrucción siguiente a la instrucción SLEEP y luego salta al vector de interrupción (0004h). En el caso en que la instrucción siguiente al SLEEP no es deseada, esta debe ser una instrucción NOP. En la Figura 1.25 se muestra el funcionamiento de la instrucción SLEEP.

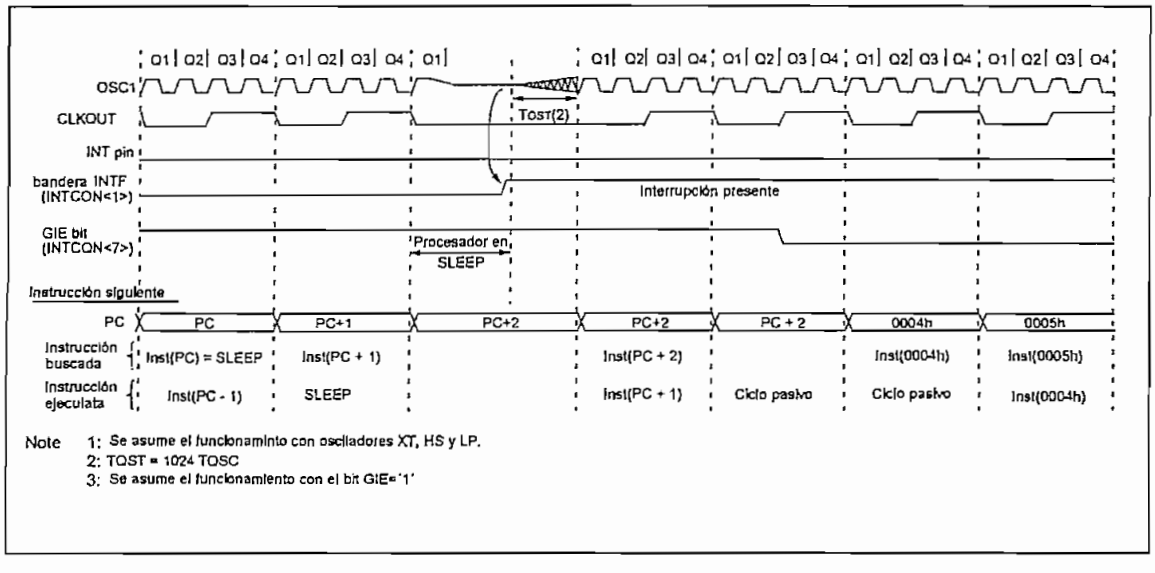


Figura. 1.25 Entrada y salida del modo de bajo consumo a través de una interrupción

Interrupciones

Este microcontrolador incluye el manejo de interrupciones, lo cual representa grandes ventajas. El PIC16F84A posee cuatro fuentes de interrupción:

- Interrupción externa en el pin RB0/INT
- Finalización del temporizador/contador TMRO
- Finalización de escritura en la EEPROM de datos
- Cambio de nivel en los pines RB4 a RB7

El registro 0Bh o INTCON contiene las banderas de las interrupciones INT, cambio en el puerto B y finalización del conteo del TMRO, al igual que el control para habilitar o inhabilitar cada una de las fuentes de interrupción, incluida la de escritura en memoria EEPROM. Sólo la bandera de finalización de la escritura reside en el registro 88h (EECON1<4>)

Si el bit GIE (Global Interrupt Enable) se coloca en 0, inhabilita todas las interrupciones. Cuando una interrupción es atendida, el bit GIE se coloca en 0 automáticamente, para evitar interferencias con otras interrupciones que se pudieran presentar. La dirección de retorno se coloca en la pila y el PC se carga con la dirección 04h. Una vez en la rutina de servicio, la fuente de la interrupción se puede determinar examinando las banderas de interrupción. La bandera respectiva se debe colocar, por software, en cero antes de regresar de la interrupción, para evitar que se vuelva a detectar nuevamente la misma interrupción.

La instrucción RETFIE permite al usuario retornar de la interrupción, a la vez que habilita de nuevo las interrupciones, al colocar el bit GIE en uno. Debe tenerse presente que solamente el contador de programa es puesto en la pila al atenderse la interrupción; por lo tanto, es conveniente que el programador tenga cuidado con el registro de estados y el de trabajo, ya que se pueden producir resultados inesperados si dentro de ella se modifican. En la Figura 1.26 se observa el circuito lógico de atención a interrupciones y de salida del modo de bajo consumo SLEEP.

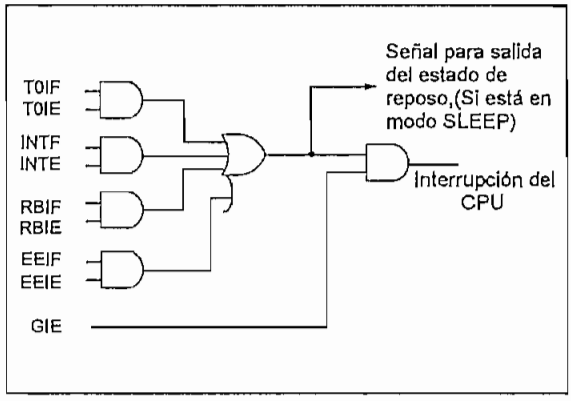


Figura. 1.26 Lógica de interrupciones

- **Interrupción externa.**

Actúa sobre el pin RB0/INT y se puede configurar para activarse con el flanco de subida o el de bajada, de acuerdo al bit INTEDG (OPTION<6>). Cuando se presenta un flanco válido en el pin INT, la bandera INTF (INTCON<1>) se coloca en uno. La interrupción se puede inhabilitar colocando el bit de control INTE (INTCON<4>) en cero. Cuando se atiende la interrupción, a través de la rutina de servicio, INTF se debe colocar en cero antes de regresar al programa principal. La interrupción puede reactivar al microcontrolador después de la instrucción SLEEP, si previamente el bit INTE fue habilitado. El estado del bit GIE decide si el procesador salta al vector de interrupción después al salir del SLEEP.

- **Interrupción por finalización de la temporización.**

La superación del conteo máximo (FFh→00h) en el TMRO colocará el bit TOIF en uno (INTCON<2>). El bit de control respectivo es TOIE (INTCON<5>).

- **Interrupción por cambio en el puerto RB.**

Un cambio en los pines del puerto B <7:4> colocará en uno el bit RBIF (INTCON<0>). El bit de control respectivo es RBIE (INTCON<3>). Para que el cambio sobre el pin de entrada/salida sea reconocido, el ancho del pulso debe ser por lo menos el ancho del tiempo de ciclo de instrucción TCY.

- **Interrupción por finalización de la escritura.**

Cuando la escritura de un dato en la EEPROM finaliza, se coloca en 1 el bit EEIF (EECON1<4>). El bit de control respectivo es EEIE (INTCON<6>).

EEIF (EECON1<4>). El bit de control respectivo es EEIE (INTCON<6>).

Memoria de datos EEPROM

El PIC16F84A tiene una memoria EEPROM de datos de 64 direcciones (0h a 3Fh), de 8 bits cada una, sobre la cual se pueden realizar operaciones de lectura y escritura, esta memoria por ser de tipo EEPROM permite guardar información que no se perderá al quitar la energía al sistema. Este bloque de memoria no se encuentra mapeado en ningún banco, el acceso a esas posiciones se consigue a través de dos registros de la RAM:

- El registro EEADR (posición 09), que debe contener la dirección de la posición de la EEPROM a ser accesada
- El registro EEDATA (posición 08), que contiene el dato de 8 bits que se va a escribir o el que se obtuvo de la última lectura.

Adicionalmente, existen dos registros de control: el EECON 1 (88h), que posee cinco bits que manejan las operaciones de lectura/escritura y el EECON2 (89h), que aunque no es un registro físico, es necesario para realizar las operaciones de escritura.

La lectura toma un ciclo del reloj de instrucciones, mientras que la escritura, por ser controlada por un temporizador incorporado, tiene un tiempo nominal de 10 milisegundos, este tiempo puede variar con la temperatura y el voltaje así como también de chip a chip. Cuando se va a realizar una operación de escritura, automáticamente se hace primero la operación de borrado. El número típico de ciclos de borrado/escritura de la EEPROM de datos es de 1.000.000. y los datos se mantienen almacenados por más de 40 años. Cuando el equipo tiene código de protección, la CPU puede continuar leyendo y escribiendo datos en la memoria EEPROM. Los equipos programadores no pueden accesar a la memoria EEPROM.

- **Lectura de la memoria EEPROM**

Para leer la memoria de datos, el usuario debe escribir la dirección en el

registro EEADR y entonces poner en ´1´ el bit de control RD (EECON1<0>). El dato estará disponible en el siguiente ciclo, en el registro EEDATA; por lo tanto, este puede ser leído en la siguiente instrucción. El registro EEDATA almacenará este valor hasta realizar otra lectura o hasta que en este registro el usuario escriba un nuevo dato (durante una operación de escritura).
El siguiente programa es un ejemplo de una operación de escritura:

```
BCF     STATUS,RP0          ;Banco 0
MOVLW   CONFIG_ADDR ;
MOVWF   EEADR               ;Dirección a ser leída
BSF     STATUS,RP0          ;Banco 1
BSF     EECON1,RD           ;Leer la memoria EEPROM
BCF     STATUS,RP0          ;Banco 0
MOVF    EEDATA,W            ;W = EEDATA
```

- **Escritura de la memoria EEPROM**

Para escribir datos en las localidades de la memoria EEPROM, primero se debe escribir la dirección en el registro EEADR y el dato en el registro EEDATA. Entonces se debe escribir la secuencia siguiente para iniciar la escritura en cada byte:

```
BSF     STATUS,RP0      ;Banco1
BCF     INTCON,GIE      ;Inhabilita interrupciones
BSF     EECON1,WREN     ;Habilita la escritura
MOVLW   55h             ;
MOVWF   EECON2          ;Escribe 55h en el registro
MOVLW   AAh             ;
MOVWF   EECON2          ;Escribe AAh en el registro
BSF     EECON1,WR       ;Setea el bit WR para iniciar escritura
BSF     INTCON,GIE      ;Habilita interrupciones
```

Las líneas en negrilla son obligatorias para iniciar la escritura (Escribir 55h en el registro EECON2, escribir AAh en el registro EECON2, seteo *del bit WR*) para cada byte. El bit WREN en el registro EECON1 debe ser seteado para habilitar la

escritura. Es recomendable que las interrupciones sean inhabilitadas durante la secuencia de este segmento de instrucciones.

Este proceso previene escrituras accidentales de datos en la memoria EEPROM debido a la ejecución de códigos erráticos. El bit WREN se debe tener en cero durante todo el tiempo, excepto cuando se actualizan datos. Además se debe tener en cuenta que el bit WREN no es borrado por hardware.

Después de iniciar secuencia de escritura, el borrado del bit WREN no afecta al ciclo de escritura iniciado. El bit WR (que inicia la escritura),será inhibido para ser seteado a menos que el bit WREN (que habilita la escritura) sea seteado.

Al completar el ciclo de escritura, el bit WR es borrado automáticamente por hardware y la bandera de finalización de escritura (bit EEIF) es seteada. Se puede habilitar esta interrupción o analizar este bit. El bit EEIF debe ser borrado por software.

Bits de configuración

El PIC16F84A posee un byte de configuración. Estos bits se pueden programar para seleccionar varias configuraciones del dispositivo: tipo de oscilador, protección de código, habilitación del circuito de vigilancia y el temporizador de encendido. Los bits se localizan en la posición de memoria 2007h, posición a la cual el usuario sólo tiene acceso durante la programación del microcontrolador.

Cuando se programa la protección de código, el contenido de cada posición de la memoria no se puede leer completamente, de tal manera que el código del programa no se puede reconstruir. Adicionalmente, todas las posiciones de memoria del programa se protegen contra la reprogramación..

Una vez protegido el código, el bit de protección sólo puede ser borrado (puesto en 1) si se borra toda la memoria del programa y la de datos.

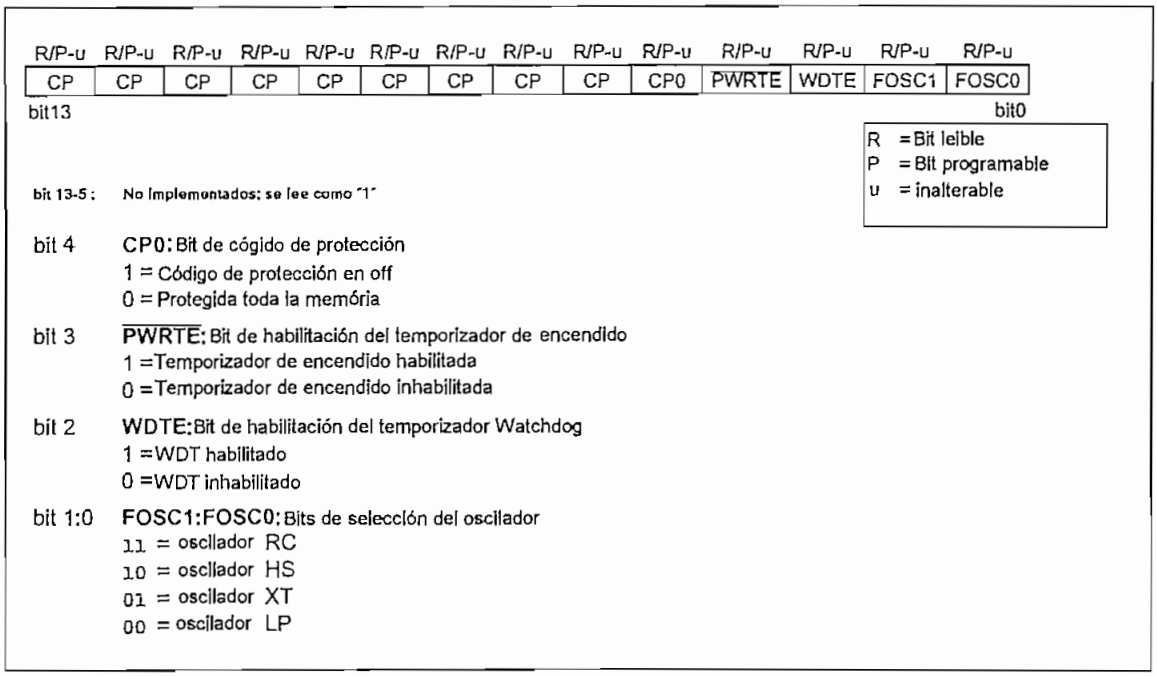


Figura 1.27 Bits de configuración

Las pull-ups internas

Cada uno de los pines del puerto B tiene un elemento pull-up interno débil (250 μA típico); este elemento es automáticamente desconectado cuando el pin se configura como salida. Adicionalmente, el bit RBPU (OPTION<7>) controla todos estos elementos, los cuales están inhabilitados ante una condición de reset. Estos elementos pull-up son especialmente útiles cuando el microcontrolador va a colocarse en el modo de bajo consumo, ya que ayudan a no tener las entradas flotantes, significando una reducción en el consumo de corriente.

Localidades de código de identificación ID

Cuatro localidades de memoria (2000h – 2004h) son diseñadas como localidades para códigos de identificación o chequeo de suma (*checksum*). Estas localidades no son accesibles durante ejecución normal pero pueden ser escritas y leídas durante programación y verificación. Solo se usan los cuatro bits menos significativos.

Programación Serial en circuito

El microcontrolador puede ser programado serialmente al estar montado sobre el circuito final. Esto se lo realiza fácilmente mediante dos líneas de reloj y de datos, y otras líneas para alimentación, tierra y voltaje de programación. Esto permite fabricar equipos que sean fácilmente actualizables con la ultima versión de firmware. El circuito típico de programación serial se muestra en la Figura 1.28.

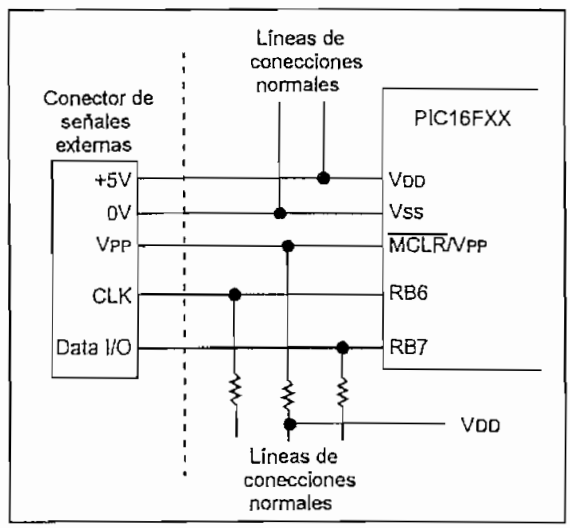


Figura. 1.28 Circuito típico de programación serial en circuito

1.2.12 El conjunto de instrucciones

Estas se clasifican en operaciones orientadas a registros, orientadas al bit y operaciones literales y de control. Cada instrucción es una palabra de 14 bits, dividida en un código de operación (el cual especifica la orden a ejecutar) y uno o más operandos sobre los que se actúa. En la Tabla 1.8 se muestra un resumen de todas las instrucciones la cuál está dividida en operaciones orientadas a bytes, a bits y a operaciones literales y de control. En los anexos se encuentra la lista completa de instrucciones, la cual incluye ejemplos y explicaciones. Como se puede observar allí, en total son 35, las cuales tardan un ciclo de máquina, a excepción de los saltos, que toman dos ciclos.

Nemotécnico		Descripción	Ciclos	Código de operación MSb LSb				Estados Afectados
Operaciones orientadas a registros								
ADDWF	f, d	Sumar W y f	1	00	0111	dfff	ffff	C,DC,Z
ANDWF	f, d	AND entre W y f	1	00	0101	dfff	ffff	Z
CLRF	f	Borrar f	1	00	0001	lfff	ffff	Z
CLRW	-	Borrar W	1	00	0001	0xxx	xxxx	Z
COMF	f, d	Complementar f	1	00	1001	dfff	ffff	Z
DECF	f, d	Decrementar f	1	00	0011	dfff	ffff	Z
DECFSZ	f, d	Decrementar f, salta si es cero	1(2)	00	1011	dfff	ffff	
INCF	f, d	Incrementar f	1	00	1010	dfff	ffff	Z
INCFSZ	f, d	Incrementar f, salta si es cero	1(2)	00	1111	dfff	ffff	
IORWF	f, d	OR entre W y f	1	00	0100	dfff	ffff	Z
MOVF	f, d	Mover f	1	00	1000	dfff	ffff	Z
MOVWF	f	Mover W a f	1	00	0000	lfff	ffff	
NOP	-	No operación	1	00	0000	0xx0	0000	
RLF	f, d	Rotar a la izquierda a través del carry	1	00	1101	dfff	ffff	C
RRF	f, d	Rotar a la derecha a través del carry	1	00	1100	dfff	ffff	C
SUBWF	f, d	Restar W de f (destino= f - W)	1	00	0010	dfff	ffff	C,DC,Z
SWAPF	f, d	Intercambiar f<7:4> ⟷ f<3:0>	1	00	1110	dfff	ffff	
XORWF	f, d	OR exclusiva entre W y f	1	00	0110	dfff	ffff	Z
Operaciones orientadas a Bits								
BCF	f, b	Borrar el bit b de f	1	01	00bb	bfff	ffff	
BSF	f, b	Activar el bit b de f	1	01	01bb	bfff	ffff	
BTFSC	f, b	Probar el bit b de f, saltar si es cero	1 (2)	01	10bb	bfff	ffff	
BTFSS	f, b	Probar el bit b de f, saltar si es uno	1 (2)	01	11bb	bfff	ffff	
Operaciones orientadas a literales y control								
ADDLW	k	Sumar el literal k a W	1	11	111x	kkkk	kkkk	C,DC,Z
ANDLW	k	AND entre k y W	1	11	1001	kkkk	kkkk	Z
CALL	k	Llamar a subrutina k	2	10	0kkk	kkkk	kkkk	
CLRWDT	-	Borrar WDT	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$
GOTO	k	Saltar a la dirección k	2	10	1kkk	kkkk	kkkk	
IORLW	k	OR entre k y W	1	11	1000	kkkk	kkkk	Z
MOVLW	k	Cargar a W con el literal k	1	11	00xx	kkkk	kkkk	
RETFIE	-	Retornar de la interrupción	2	00	0000	0000	1001	
RETLW	k	Retornar de la interrupción y cargar W con K	2	11	01xx	kkkk	kkkk	
RETURN	-	Retornar de subrutina	2	00	0000	0000	1000	
SLEEP	-	Ir al modo de bajo consumo	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$
SUBLW	k	Restar W a k (W = k - W)	1	11	110x	kkkk	kkkk	C,DC,Z
XORLW	k	OR exclusiva entre k y W	1	11	1010	kkkk	kkkk	Z

Nota: Si d = 0 el resultado se almacena en W
Si d = 1 el resultado se almacena en el registro

Tabla 1.8 Resumen de instrucciones

En las instrucciones orientadas a registros, ´f´ representa un registro del archivo de registros, y ´d´ el destino en donde se guardará el resultado. Si d es cero, el resultado es colocado en el registro W. Si d es uno, el resultado es colocado en el registro indicado por la instrucción. En las instrucciones orientadas a bits, ´b´ representa el número de bit ha ser afectado, mientras que ´f´ representa el registro del archivo de registros en el cual está ubicado el bit.

En las instrucciones orientadas a literales y control, ´k´ representa una constante o valor literal de ocho u once bits.

Todas las instrucciones son ejecutadas con un solo ciclo de instrucción, excepto las instrucciones de prueba con resultado verdadero o cuando el contador de programa es cambiado como resultado de la instrucción. En este caso, la instrucción toma dos ciclos de instrucción, en donde la segunda instrucción es ejecutada como un NOP. Una instrucción consiste en cuatro períodos de oscilación. Para un oscilador de 4 Mhz, el tiempo de ejecución de una instrucción normal es de 1usg, mientras que para instrucciones de prueba con resultado verdadero o cuando el contador de programa es cambiado, este tiempo aumenta a 2us.

En la Figura 1.29 se muestra los formatos generales que pueden tener los distintos tipos de instrucciones:

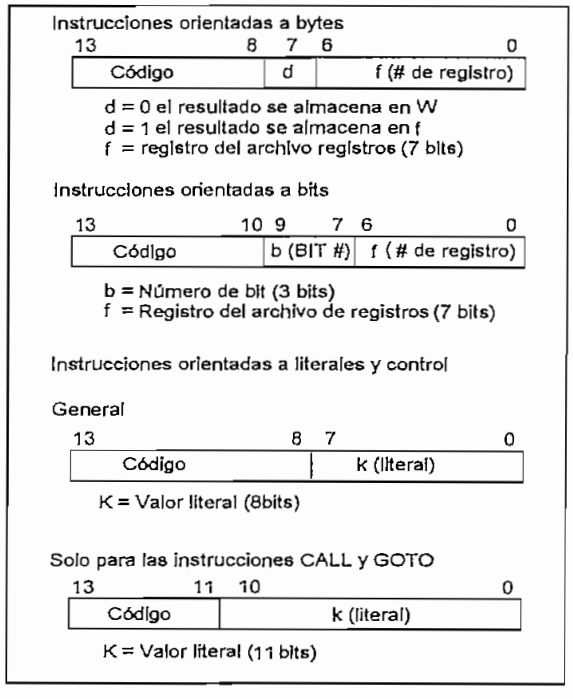


Figura 1.29 Formato general de instrucciones

1.3 EL PIC16C71

1.3.1 Características generales

Este microcontrolador tiene a diferencia del PIC16F84A un convertidor análogo digital interno bastante interesante, que está implementado en vez de la memoria EEPROM interna; esto permite resolver de una manera muy sencilla las

lecturas de señales análogas evitando el uso de buses externos, alambrados, fuentes de alimentación duales, entre otras que aumentaban las dificultades.

El conversor interno tiene las siguientes características básicas:

- Cuatro canales de entrada
- Tiempo de conversión mínimo de 20us.
- Voltaje de referencia interno o externo
- Resolución de 8 bits con precisión de ± 1 LSB.
- Rango de entrada análoga de voltaje desde Vss hasta Vref.

Este microcontrolador incluye las siguientes variantes:

- PIC16C710
- PIC16C71
- PIC16C711
- PIC16C715

Estos microcontroladores tienen en común las siguientes características generales:

- Solo 18 pines, de los cuales 13 son de entrada/salida, con control individual de direcciones y capacidad para manejar leds directamente.
- Conjunto de instrucciones RISC (set de instrucciones reducidas) de solo 35 elementos.
- Todas las instrucciones se ejecutan en un solo ciclo exceptuando las instrucciones de saltos que utilizan dos ciclos.
- Velocidades de operación desde DC hasta 20Mhz.
- Hasta 2K x 14 palabras de memoria de programa y hasta 128 x 8 bytes de memoria de datos (RAM).
- Capacidad de generación de interrupciones.
- Pila de 8 niveles.
- Modos de direccionamiento directo, indirecto y relativo.
- Reset de encendido (POR).
- Temporizador de encendido (PWRT).
- Temporizador de encendido del oscilador (OST).
- Circuito de vigilancia incorporado (Watchdog Timer).
- Seguridad para protección de código de programa.
- Modo de consumo de baja corriente.

- Opciones de oscilador seleccionables.
- Diseño totalmente estático.
- Rango de operación de voltaje de 2,5V a 6V.
- Alto manejo de corriente en modo sumidero y fuente de 25mA.
- Modelos comerciales, industriales y de rango extendido de temperatura.
- Bajo consumo de corriente:
  - < 2 mA @ 5V, 4 Mhz.
  - 15uA típicos @3V, 32 Khz.
  - < 1uA en modo de bajo consumo
- Timer0: temporizador/Contador de 8 bits con preescalador de 8 bits.
- Convertidor análogo a digital multicanal de 8 bits.
- Circuito de detección de falta de energía eléctrica para reset de falla de energía (BOR).
- 13 pines de I/O con control de dirección individual.

A continuación se indica la Tabla 1.9 con las capacidades de cada uno de los modelos de los PIC16C71X.

Características del PIC16C7X	710	71	711	715
Memória de programa (EPROM) x 14 Bits	512	1K	1K	2K
Memória de Datos (bytes) x 8 bits	36	36	68	128
Pines de I/O	13	13	13	13
Temporizadores	1	1	1	1
Canales A/D	4	4	4	4
Programación serial en circuito	Yes	Yes	Yes	Yes
Reset de Falta de energía	Yes	—	Yes	Yes
Fuentes de Interrupción	4	4	4	4

Tabla 1.9 Capacidades de los PIC16C7X

1.3.2 Pines y funciones

El PIC 16C7X es un microcontrolador de Microchip Technology fabricado con tecnología CMOS, su consumo de potencia es muy bajo y además es completamente estático, esto quiere decir que su reloj puede detenerse y los datos de la memoria no se pierden. El diagrama de pines con sus funciones y numeración se observa en la Figura 1.30. El encapsulado más común para el microcontrolador es el DIP (Dual In-line Pin) de 18 pines, propio para usarlo en

experimentación. La referencia completa es 16C71-04/P. Sin embargo, hay otros tipos de encapsulado que se pueden utilizar según el diseño y la aplicación que se quiere realizar. Por ejemplo, el encapsulado tipo surface mount (montaje superficial) tiene un reducido tamaño y bajo costo, que lo hace propio para producciones en serie o para utilizarlo en lugares de espacio muy reducido.

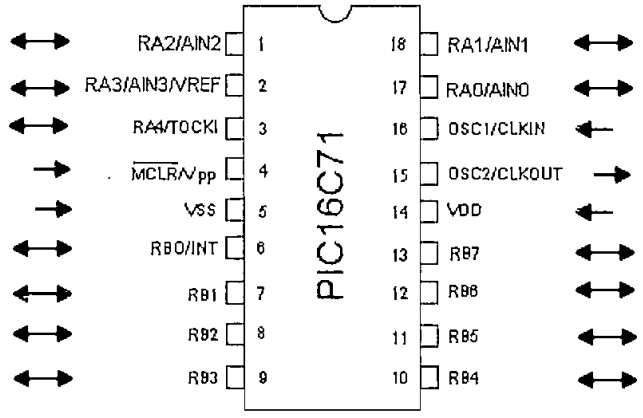


Figura 1.30. Diagrama de pines del PIC16C71

La Figura 1.31 muestra los tipos de empaque que puede tener el integrado.

Figura 1.31. Tipos de encapsulado

1.3.3 Puertos del microcontrolador

Los puertos son el puente entre el microcontrolador y el mundo exterior. Son líneas digitales que trabajan entre cero y cinco voltios y se pueden configurar como entradas o como salidas y algunas de ellas se pueden configurar como entradas analógicas, las cuales aceptan voltajes continuos que pueden variar entre cero y cinco voltios. Estas son las que toman el convertidor análogo a digital.

El PIC16C71 tiene dos puertos. El puerto A con 5 líneas y el puerto B con 8

líneas (Figura 1.32). Cada pin de los puertos se puede configurar como entrada o como salida, independientemente, programando un par de registros diseñados para tal fin. En ese registro un "0" configura el pin del puerto correspondiente como salida y un "1" lo configura como entrada.

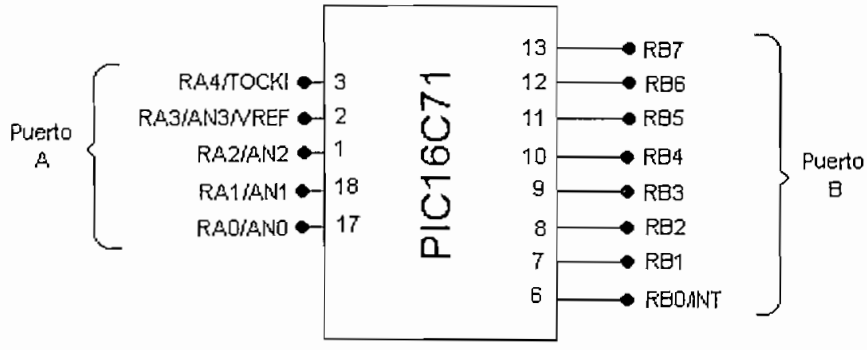


Figura 1.32. Puertos del PIC16C71

Los pines RA0:RA3 del puerto A tiene Buffers de entrada tipo TTL y drivers de salida tipo CMOS como se indica en la Figura 1.33.

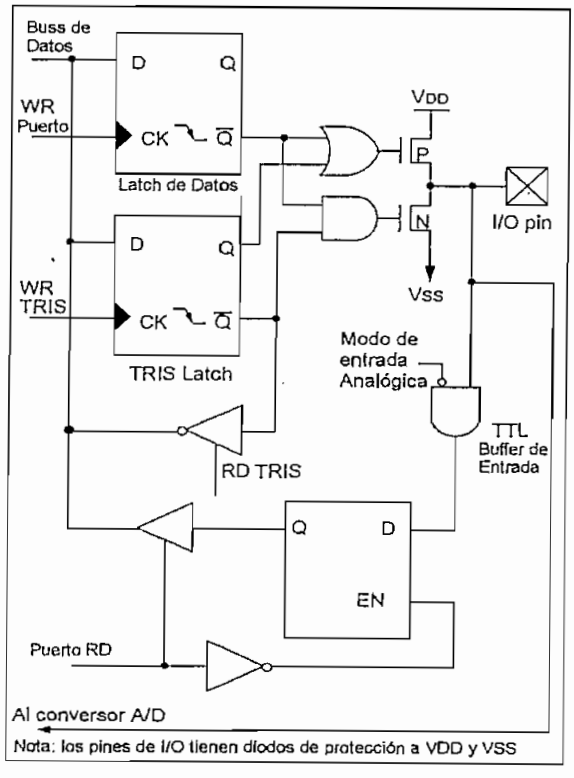


Figura. 1.33 Diagrama de bloques de los pines RA3:RA0

Los pines del puerto A son multiplexados con entradas analógicas y entrada de voltaje de referencia. La operación de cada pin es seleccionada por seteo o borrado de los bits de control en el registro ADCON1 (A/D *Control Register1*). Luego del reset de encendido, estos pines son configurados como entradas analógicas y se lee como ´0´. El registro TRISA controla la dirección de los pines RA, también cuando estos son utilizados como entradas analógicas. El usuario debe asegurarse que el bit en el registro TRISA se mantenga en set cuando este es utilizado como entrada analógica.

El pin RA4/TOCK1 del puerto A puede ser configurado como un pín de entrada salida o como entrada del temporizador contador. Cuando este pín se programa como entrada digital funciona como un disipador schmitt (*schmitt trigger*), puede reconocer señales un poco distorsionadas y llevarlas a niveles lógicos (cero y cinco voltios). Cuando se usa como salida digital se comporta como colector abierto por lo tanto se debe poner una resistencia *pull-up* (resistencia externa conectada a un nivel de cinco voltios). Como salida la lógica es inversa: un ´0´ escrito al pín del puerto entrega a la salida un ´1´ lógico este pín como salida no puede manejar cargas en modo fuente, solo en el modo sumidero. El diagrama de bloques del pin RA4 se muestra en la Figura 1.34 a continuación.

Los cuatro pines RB7:RB4 se muestra en la Figura 1.35 y pueden generar una interrupción al detectar un cambio en sus pines, siempre y cuando estos sean programados como entradas. Esto se configura por software utilizando unos bits de los registros INTCON y OPTION. Solo los pines configurados como entradas pueden generar dicha interrupción. El valor de los pines de entrada es comparado con el valor almacenado de la ultima lectura del puerto B. Esta interrupción puede despertar al microcontrolador del modo SLEEP. Se debe borrar la bandera de la interrupción en la rutina de atención a esta, de cualquiera de las siguientes formas:

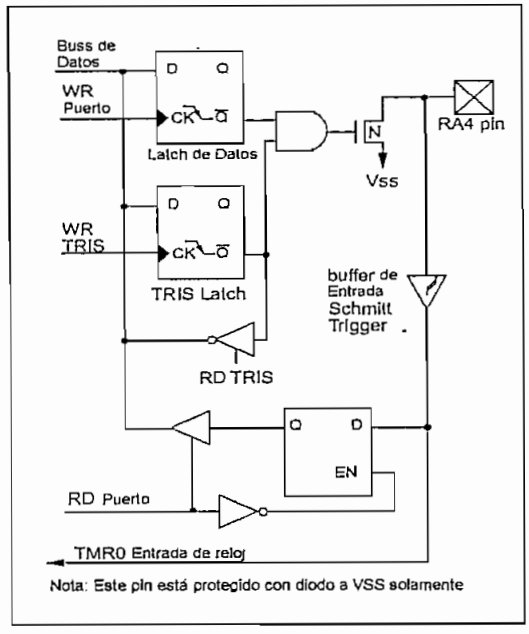


Figura 1.34 Diagrama de bloques del PIN RA4

a) Leyendo (o escribiendo) sobre el puerto B. Esto finalizará la condición de diferencia.

b) Borrando la bandera bit RBIF.

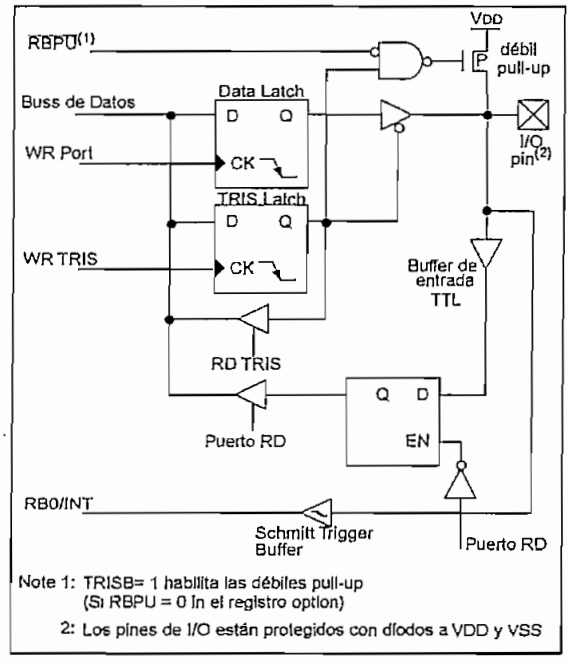


Figura 1.35 Diagrama de Bloque de los pines RB3:RB0

El diagrama de bloques de los pines RB7:RB4 se muestra en la Figura 1.36 a continuaciòn.

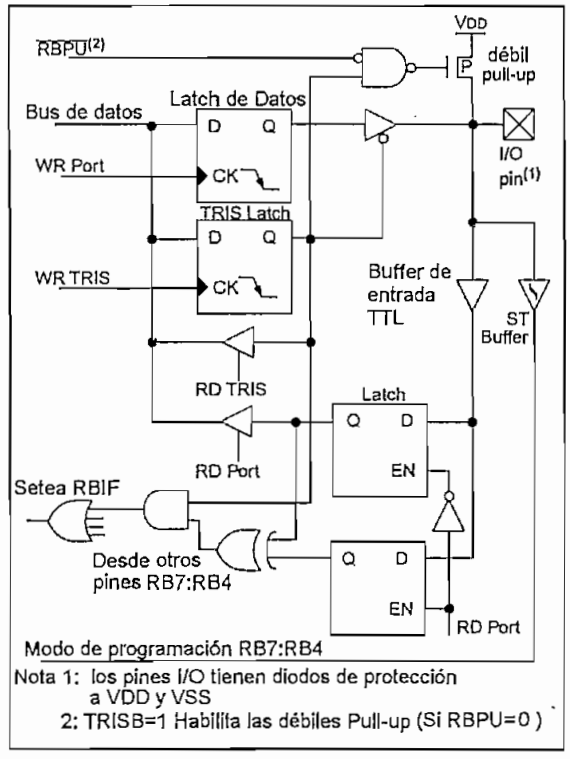


Figura 1.36 Diagrama de Bloques de los pines RB7:RB4 (PIC16C71)

Como este dispositivo es de tecnología CMOS, todos los pines deben estar conectados a alguna parte, nunca dejarlos al aire porque se puede dañar el integrado. Los pines que no se estén utilizando se deben conectar a la fuente de alimentación de más 5v. La máxima capacidad de corriente de cada uno de los pines de los puertos en modo sumidero (sink) es de 25 mA y en modo fuente (source) es de 20 mA. La máxima capacidad de corriente total de los puertos se muestra en la Figura 1.37.

El consumo de corriente del microcontrolador para su funcionamiento depende del voltaje de operación, la frecuencia y de las cargas que tengan sus pines. Para un reloj de 4 MHz el consumo es de aproximadamente 2 mA; aunque este se puede reducir a 40 microamperios cuando se está en el modo sleep (en este modo el micro se detiene y disminuye el consumo de potencia). Se sale de ese estado cuando se produce alguna condición especial como por ejemplo una interrupción.

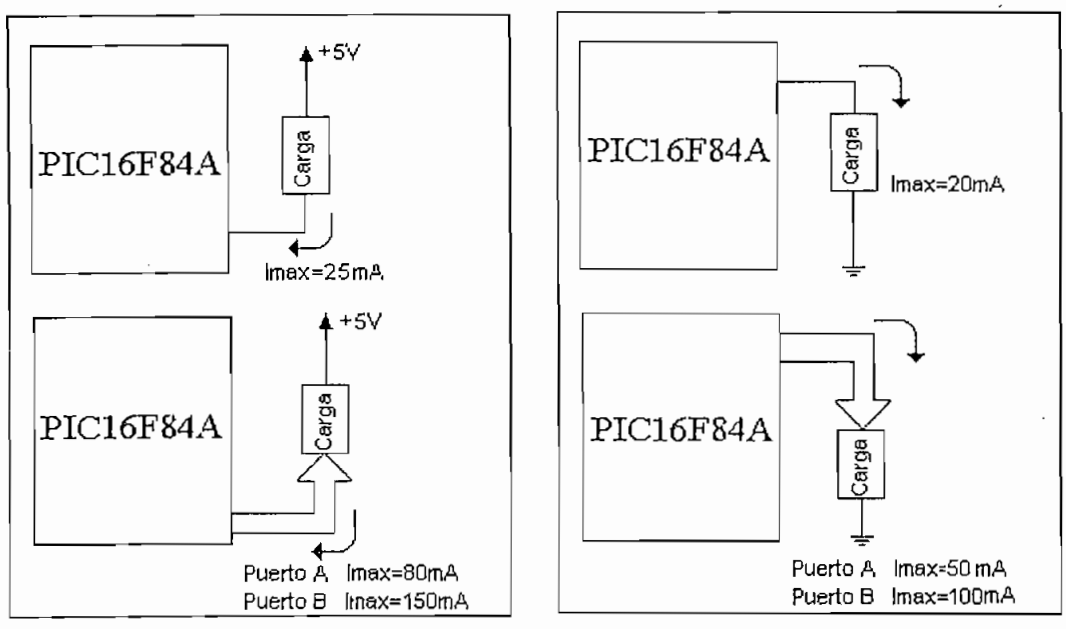


Figura 1.37 Capacidad de corriente del PIC16F84A

1.3.4 El oscilador externo

Al igual que los anteriormente estudiados, el PIC16C71 puede utilizar cuatro tipos de oscilador diferentes programando los bits de configuración (FOSC1 y FOSC0) para seleccinar uno de estos osciladores. Los tipos de osciladores son:

- **RC.** Oscilador con resistencia y condensador.
- **XT.** Cristal / Resonador
- **HS.** Cristal de alta velocidad / Resonador.
- **LP.** Cristal para baja frecuencia y bajo consumo de potencia.

En el momento de programar o "quemar" el microcontrolador se debe especificar que tipo de oscilador se usa. Esto se hace igualmente a través de unos fusibles de configuración.
Para los tipos de cristal XT, LP o HS estos se conectan a los pines de OSC1/CLKIN y OSC2/CLKOUT.

El tipo de oscilador que se sugiere para las prácticas es el cristal de 4 MHz, porque garantiza mayor precisión y un buen arranque del microcontrolador. Inter-

namente esta frecuencia es dividida por cuatro, lo que hace que la frecuencia efectiva de trabajo sea de 1 MHz, por lo que cada instrucción se ejecuta en un microsegundo. El cristal debe ir acompañado de dos condensadores que se conectan como se muestra en la Figura 1.38.

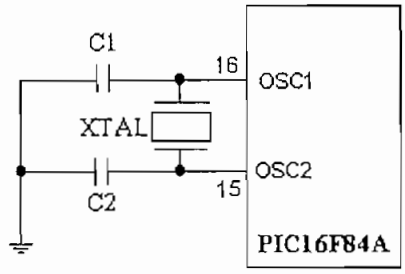


Figura 1.38. Conexión de un oscilador a cristal

Así mismo, dependiendo de la aplicación, se pueden utilizar cristales de otras frecuencias; por ejemplo se usa el cristal de 3,579545 MHz porque es muy económico, el de 32,768 KHz cuando se necesita crear bases de tiempo de un segundo muy precisas. El límite de velocidad en estos microcontroladores es de 20 MHz.

De igual forma, si no se requiere mucha precisión en el oscilador y se quiere economizar dinero, se puede utilizar una resistencia y un condensador, como se muestra en la Figura 1.39. Los valores recomendados de Rext y Cext son: 5KΩ ≤ Rext≤ 100KΩ y Cext > 20pF.

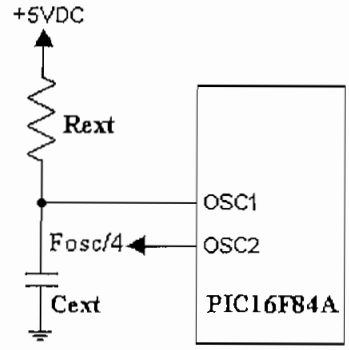


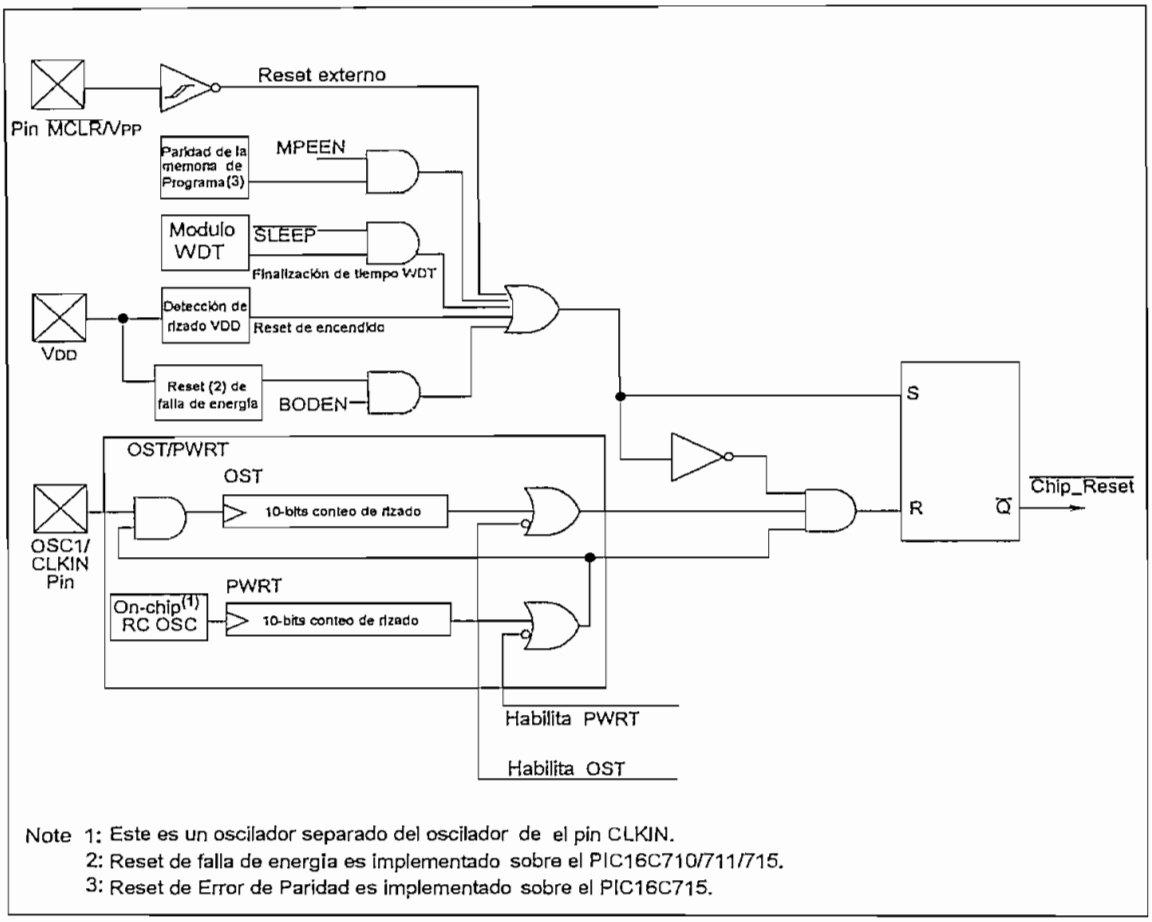
Figura 1.39. Conexión de un oscilador RC

1.3.5 Reset

En los microcontroladores se requiere un pin de reset para reiniciar el funcionamiento del sistema cuando sea necesario, ya sea por una falla que se presente o porque así se halla diseñado el sistema. El pin de reset en los PIC es llamado MCLR (master clear). El PIC 16C71 admite diferentes tipos de reset:

- Al encendido (Power On Reset)
- Pulso en el pin MCLR durante operación normal
- Pulso en el pin MCLR durante el modo de bajo consumo (modo sleep)
- El rebase del conteo del circuito de vigilancia (watchdog) durante operación normal.
- El rebase del conteo del circuito de vigilancia (watchdog) durante el modo de bajo consumo (sleep).

En la Figura 1.40 se muestra un diagrama de bloques del circuito de reset. El MCLR tiene un filtro para ignorar pulsos pequeños, por lo que estos pulsos de reset deben ser grandes.



Note 1: Este es un oscilador separado del oscilador de el pin CLKIN.
2: Reset de falla de energía es implementado sobre el PIC16C710/711/715.
3: Reset de Error de Paridad es implementado sobre el PIC16C715.

Figura. 1.40 Diagrama de bloques del circuito de RESET

Los registros de la memoria RAM del microcontrolador son afectados en distinta forma según el tipo de reset. En la Tabla 1.10 se indican las condiciones de reset para el contador de programa (PC), el registro STATUS y el registro PCON.

Condición	Contador de programa	Registro STATUS	Registro PCON PIC16C710/711
Reset de encendido	000h	0001 1xxx	---- --0x
$\overline{MCLR}$ Reset durante operación normal	000h	000u uuuu	---- --uu
$\overline{MCLR}$ Reset durante reposo	000h	0001 0uuu	---- --uu
Reset WDT	000h	0000 1uuu	---- --uu
Salida del modo de reposo por WDT	PC + 1	uuu0 0uuu	---- --uu
Reset por falla de energía (PIC16C710/711)	000h	0001 1uuu	---- --u0
Interrupción durante modo de reposo	PC + 1(1)	uuu1 0uuu	---- --uu

Simbolos: u=no cambia, x=desconocido, -=no implementado bit se lee como '0'.
Nota 1: Cuando la salida del modo de reposo es debido a una interrupción y el bit GIE está en '1', el PC es cargado con el vector de interruptor (0004h).

Tabla 1.10 Condición de reset para el contador de programa y el registro STATUS

En la Tabla 1.11 de describe los estados de reset para todos los registros.

Registro	Reset al encendido Reset por falla de energía	Reset en MCLR Reset por WDT	Salida del modo de reposo vía WDT o Interrupción
W	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF	N/A	N/A	N/A
TMR0	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000h	0000h	PC + 1[2]
STATUS	0001 1xxx	000q quuu[3]	uuuq quuu[3]
FSR	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	---x 0000	---u 0000	---u uuuu
PORTB	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCLATH	---0 0000	---0 0000	---u uuuu
INTCON	0000 000x	0000 000u	uuuu uuuu[1]
ADRES	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	00-0 0000	00-0 0000	uu-u uuuu
OPTION	1111 1111	1111 1111	uuuu uuuu
TRISA	---1 1111	---1 1111	---u uuuu
TRISB	1111 1111	1111 1111	uuuu uuuu
PCON[4]	---- --0u	---- --uu	---- --uu
ADCON1	---- --00	---- --00	---- --uu

Notas: u =no cambia; x = desconocido; - =bit no implementado y se lee con '0';
El reset por falla de energía no es implementado en el PIC16C71

(1) Uno o más bits del INTCON serán afectados (a causa de despertarse del estado de reposo)
(2) Cuando el microcontrolador sale de una condición de reset debido a una interrupción y el bit GIE está en 1 lógico, el contador de programa es cargado con el vector de interrupción.
(3) En la tabla 1.10 lista las condiciones de reset para cada condición específica.
(4) El registro PCON no es implementado sobre el PIC16C71.

Tabla 1.11 Condiciones de reset para todos los registros.

El reset al encendido se consigue gracias a dos temporizadores. El primero es el PWRT (Power-Up Timer. Temporizador de encendido), que provee un retardo fijo de 72 ms (nominal) en el encendido únicamente, diseñado para mantener el dispositivo en reset mientras la fuente se estabiliza; este retardo se consigue mediante un oscilador RC interno, y puede ser inhabilitado mediante el bit de configuración PWRET. El Segundo de ellos es el OST (Oscillator Start- Up Timer : Temporizador de encendido del oscilador), que provee un retardo de 1024 ciclos de oscilación, orientado a mantener el microcontrolador en reset hasta que el oscilador del cristal se estabilice. Este temporizador inicia al terminarse el tiempo del temporizador de encendido PWRT. Este oscilador es habilitado solo

con los osciladores tipo XT, LP y HS.

Para utilizar estos temporizadores, sólo basta con conectar el pin MCLR a la fuente de alimentación a través de una resistencia, evitándose utilizar las tradicionales redes RC externas en el pin de reset.

El reset por MCLR se consigue llevando momentáneamente este pin a un estado lógico bajo, mientras que el watchdog WDT produce el reset cuando su temporizador rebasa la cuenta, o sea que pasa de 0FFh a 00h. Cuando se quiere tener control sobre el reset del sistema se puede conectar un pulsante para llevar momentáneamente a tierra generando un reset.

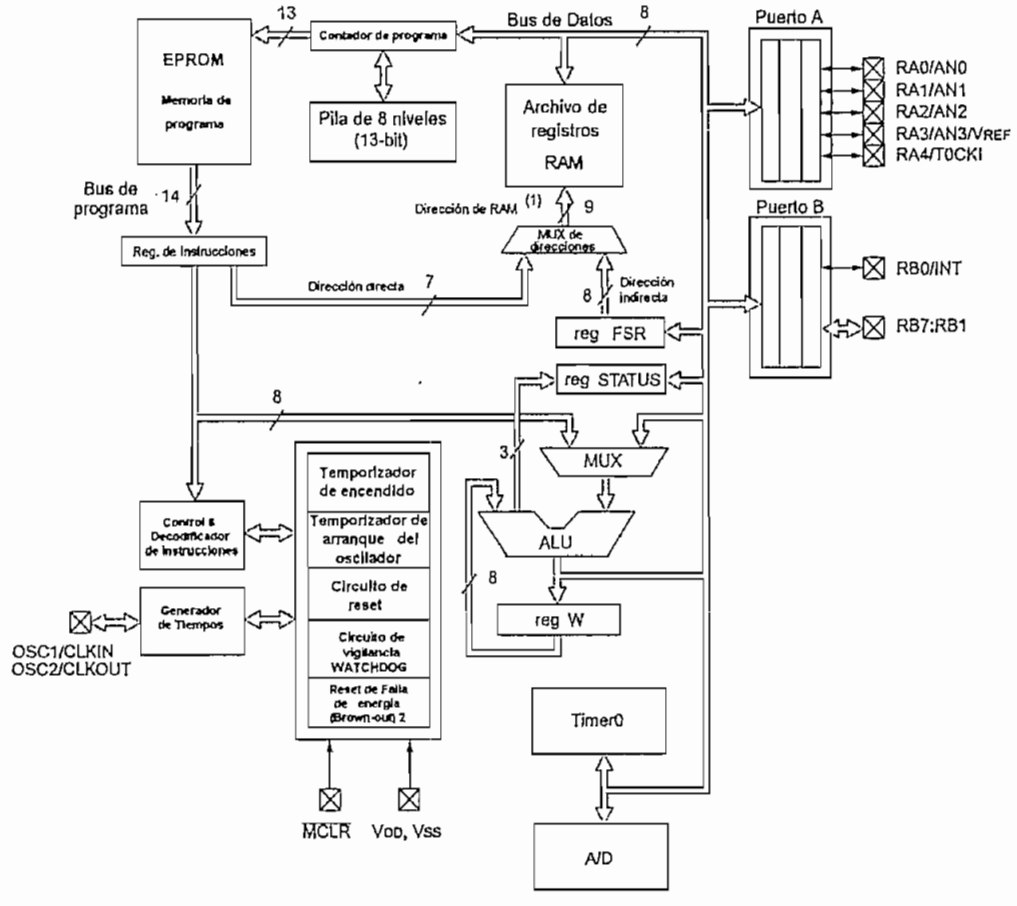
1.3.6 Arquitectura

Este término se refiere a los bloques funcionales internos que conforman el microcontrolador y la forma en que están conectados, por ejemplo la memoria RAM (de datos), los puertos, la lógica de control que permite que todo el conjunto funcione, el convertidor análogo a digital, etc.

La Figura 1.41 muestra la arquitectura general del PIC16C71, en ella se pueden apreciar los diferentes bloques que lo componen y la forma en que se conectan. Se muestra la conexión de los puertos, las memorias de datos y de programa, los bloques especiales como el watchdog, los temporizadores de arranque, el oscilador, etc.

Todos los elementos se conectan entre sí por medio de buses. Un bus es un conjunto de líneas que transportan información entre dos o más módulos. Vale la pena destacar que el PIC16C71 tiene un bloque especial conformado por el convertidor análogo a digital.

Equipo	Memoria de programa	Memoria de Datos
PIC16C710	512 x 14	36 x 8
PIC16C71	1K x 14	36 x 8
PIC16C711	1K x 14	68 x 8
PIC16C715	2K x 14	128 x 8



Nota 1: Los bits de alto orden son direccionados desde el registro STATUS

2: El reset de falla de energía (Brown - out) no está implementado en el PIC16C71.

Figura. 1.41 Arquitectura interna del PIC16C71

El PIC16C71 se basa en la arquitectura Harvard, en la cual el programa y los datos pueden trabajar desde memorias separadas, lo que posibilita que las instrucciones y los datos posean longitudes diferentes. Esta misma estructura es la que permite la superposición de los ciclos de búsqueda y ejecución de las instrucciones, lo cual se ve reflejado en una mayor velocidad del microcontrolador.

1.3.7 Memoria de programa

La familia de los PIC16C71X tiene un contador de programa de 13 bits con capacidad de direccionamiento de hasta 8K x 14 de memoria de programa. La capacidad de memoria de cada versión es listada a continuación en la tabla 1.12:

Equipo	Memória de Programa	Rango de Direcciones
PIC16C710	512 x 14	0000h-01FFh
PIC16C71	1K x 14	0000h-03FFh
PIC16C711	1K x 14	0000h-03FFh
PIC16C715	2K x 14	0000h-07FFh

Tabla 1.12 Capacidad de memoria de los PIC16C71X

Si se direccionan posiciones de memoria superiores al valor máximo se causará un solapamiento con el espacio existente. En la Figura 1.42 se muestra el mapa de la memoria de programa del PIC16C71/711.

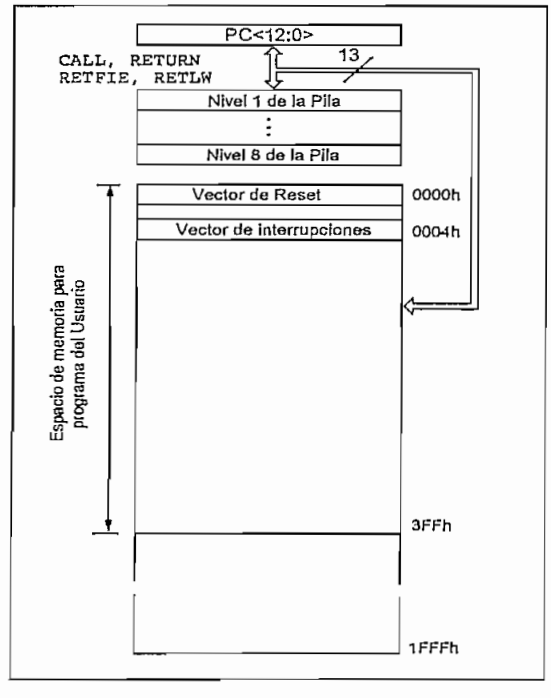


Figura 1.42 Mapa de la memoria de programa del PIC16C71/711.

Vector de reset.

Cuando ocurre un reset al microcontrolador, el contador de programa se pone en ceros (000H). Por esta razón, en la primera dirección del programa se

debe escribir todo lo relacionado con la iniciación del mismo.

Vector de interrupción.

Cuando el microcontrolador recibe una señal de interrupción, el contador de programa apunta a la dirección 04H de la memoria de programa, por eso, allí se debe escribir toda la programación necesaria para atender dicha interrupción.

1.3.8 Registros (Memoria RAM)

El PIC16C71 puede direccionar 128 posiciones de memoria RAM, pero solo tiene implementados físicamente los primeros 48 (0-2F en hexadecimal).

De estos los primeros 12 son registros que cumplen un propósito especial en el control del microcontrolador y los 36 siguientes son registros de uso general que se pueden usar para guardar los datos temporales de la tarea que se está ejecutando (Figura 1.43). Los registros están organizados como dos arreglos (páginas) de 128 posiciones de 8 bits cada una (128 x 8).

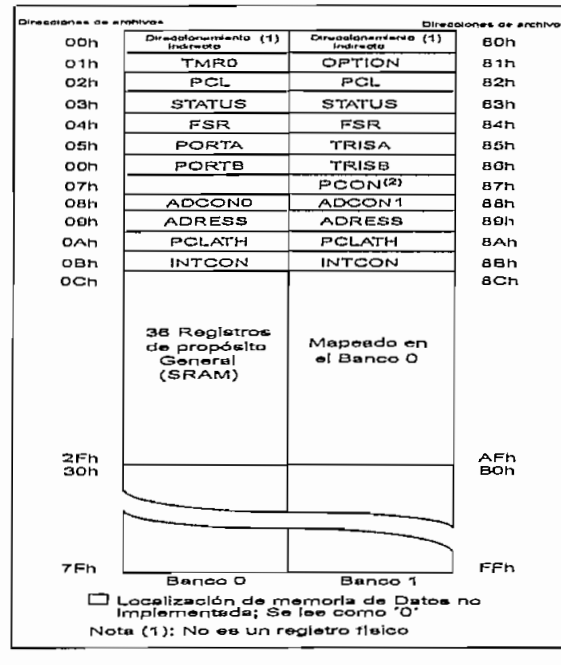


Figura. 1.43 Mapa del archivo de registros PIC16C71/710

Archivo de registros de propósito general (GPR)

Cada registro de propósito general tiene 8 bits de ancho y son accesadas mediante direccionamiento directo e indirecto a través del registro selector de

registros FSR. Los GPR direccionados en el banco 1 son mapeados en el banco 0.

Registros de Función especial.

Los registros de función especial son usados por la CPU y funciones de periféricos para controlar las operaciones del sistema. Estos registros son RAM estática. Los registros de función especial con la identificación de sus bits respectivos son indicados a continuación en la Tabla 1.13

Dirección	Nombre	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valor despues del reset de encendido	Valor despues de todos los otros reset (Nota 3)
Banco 0											
00h[a]	INDF	Contenido usado por FSR para la dirección de memoria de datos (no es un registro físico)								0000 0000	0000 0000
01h	TMR0	Reloj de tiempo real / contador de 8 bits								xxxx xxxx	uuuu uuuu
02h[a]	PCL	8 bits bajos del contador de programa PC								0000 0000	0000 0000
03h[a]	STATUS	IRP[5]	RP1[5]	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
04h[a]	FSR	Dirección Indirecta de memoria de Datos, puntero del INDF								xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	—	PORTA: LATCH DE DATOS CUANDO ESCRIBE; PORTA: PINES CUANDO LEE					---x xxxx	---u uuuu
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07h		Localidad no implementada, se lee con '0'								---- ----	---- ----
08h	ADCON0	ADCS1	ADCS0	(6)	CHS1	CHS0	$\overline{GO/DONE}$	ADIF	ADON	00 0 0000	0000 0000
09h[a]	ADRES	REGISTRO DE RESULTADOS OEL CONVERSOR A/D								xxxx xxxx	uuuu uuuu
0Ah[a,3]	PCLATH	—	—	—	Buffer de escritura de los 5 bits altos del PC					---0 0000	---0 0000
0Bh[a]	INTCON	GIE	ADIE	T0IE	INTE	RBIE	T0IF	$\overline{INTF}$	RBIF	0000 000x	0000 000u
Bank 1											
80h[a]	INDF	Contenido usado por FSR para la dirección de memoria de datos (no es un registro físico)								0000 0000	0000 0000
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h[a]	PCL	8 bits bajos del contador de programa PC								0000 0000	0000 0000
83h[a]	STATUS	IRP[5]	RP1[5]	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
84h[a]	FSR	Dirección indirecta de memoria de Datos, puntero del INDF								xxxx xxxx	uuuu uuuu
85h	TRISA	—	—	—	Registro de configuración del puerto A					---1 1111	---1 1111
86h	TRISB	Registro de configuración del puerto B								1111 1111	1111 1111
87h[4]	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	qq	uu
88h	ADCON1	—	—	—	—	—	—	PCFG1	PCFG0	00	00
89h[a]	ADRES	REGISTRO DE RESULTADOS DEL CONVERSOR A/D								xxxx xxxx	uuuu uuuu
8Ah[a,3]	PCLATH	—	—	—	Buffer de escritura para los 5 bits altos del PC					---0 0000	---0 0000
8Bh[a]	INTCON	GIE	ADIE	T0IE	INTE	RBIE	T0IF	$\overline{INTF}$	RBIF	0000 000x	0000 000u

Símbolos: x=desconocido, u=no cambia, q=valor depende de la condición, -= no implementado se lee como '0'.

Notas 1: Otro reset (no de encendido) incluyendo los reseta externos en $\overline{MCLR}$ y del WDT.

2: El byte alto del contador de programa no es directamente accesible. PCLATH es un registro que contiene los bits PC<12:8> cuyo contenido es transferido a la parte alta del PC.

3: Estos registros pueden ser accesados desde cualquiera de los dos bancos.

4: El registro PCON no está físicamente implementado en el PIC16C71, se lee como '0'.

5: Los bits IRP y RP1 son reservados para los PIC16C710/711/711, siempre se mantienen a estos bits en cero.

6: El bit 6 del ADCON0 es un registro de propósito general de lectura o escritura para el PIC16C710/711 solamente. Para el PIC16C71, este bit no es implementado y se lee como '0'.

Tabla 1.13 Registros de función especial

00h ó INDF: Registro para direccionamiento indirecto de datos.

Este no es un registro disponible físicamente; utiliza el contenido del FSR y el bit RP0 del registro STATUS para seleccionar indirectamente la memoria de

datos o RAM del usuario; la instrucción determinará que se debe realizar con el registro señalado.

0Ih ó TMRO. Temporizador / contador de 8 bits.

Este registro puede operar como temporizador o como contador. El modo temporizador es seleccionado al escribir un ´1´ en el bit TOSC del registro OPTION, en este caso el registro se incrementará con cada ciclo de instrucción (según el preescalador). Cuando se escribe sobre el registro TMR0, el conteo es inhibido para los dos siguientes ciclos de instrucción. El modo contador es seleccionado al escribir un ´0´ en el bit TOSC del registro OPTION, en este caso el registro se incrementará mediante una señal externa aplicada al pin RA4/TOCKI y puede ser seleccionado para actuar con los flancos de subida o con los de bajada. El flanco de subida o bajada es seleccionado mediante el bit TOSE del registro OPTION. En este caso se requiere que el reloj externo sea sincronizado con la fase del reloj interno (TOSC). El diagrama de bloques del TMR0 se muestra en la Figura 1.44 mostrada a continuación.

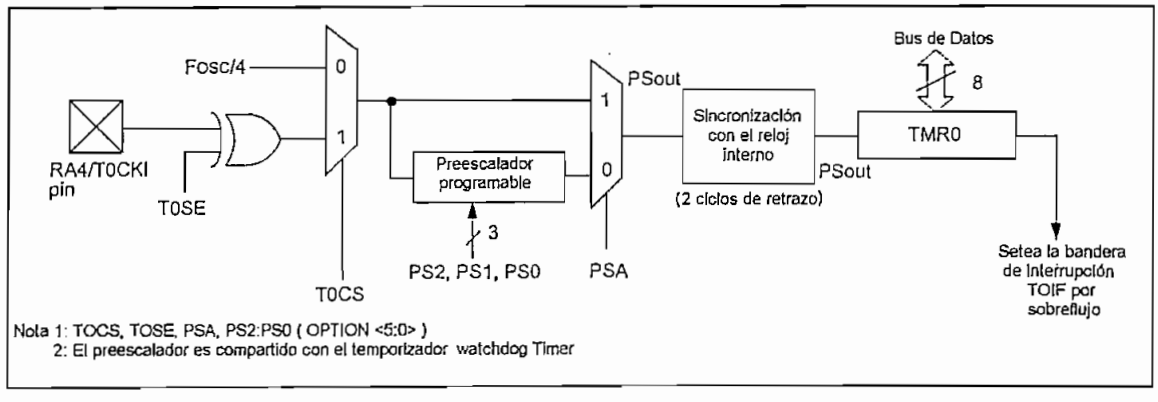


Figura. 1.44 Diagrama de bloques del TIMER0

Preescalador.

Un contador de 8 bits sirve para preescalar el temporizador TMR0 o el Wacthdog Timer, este preescalador puede ser usado para uno de los dos temporizadores pero no para los dos. El preescalador no es leíble ni escribible. Un diagrama de bloques del preescalador del TIMER0/WDT se muestra en la Figura 1.45.

Los bits PSA y PS2:PS0 del registro OPTION determinan la asignación y nivel de preescalación.

Con PSA=0 asignará el preescalador al TIMER0 y en este caso el preescalador toma los valores de 1:2, 1:4, ...1:256.

Con PSA=1 asignará el preescalador al Watchdog Timer y en este caso el preescalador toma los valores de 1:1, 1:2, ...1:128.

Cuando se asigna el preescalador al TIMER0, las instrucciones de escritura sobre el TMR0 borrarán el conteo del preescalador y cuando el preescalador es asignado al WDT la instrucción CLRWDT borrará el conteo del preescalador; pero estas operaciones no cambiarán la asignación dada al preescalador.

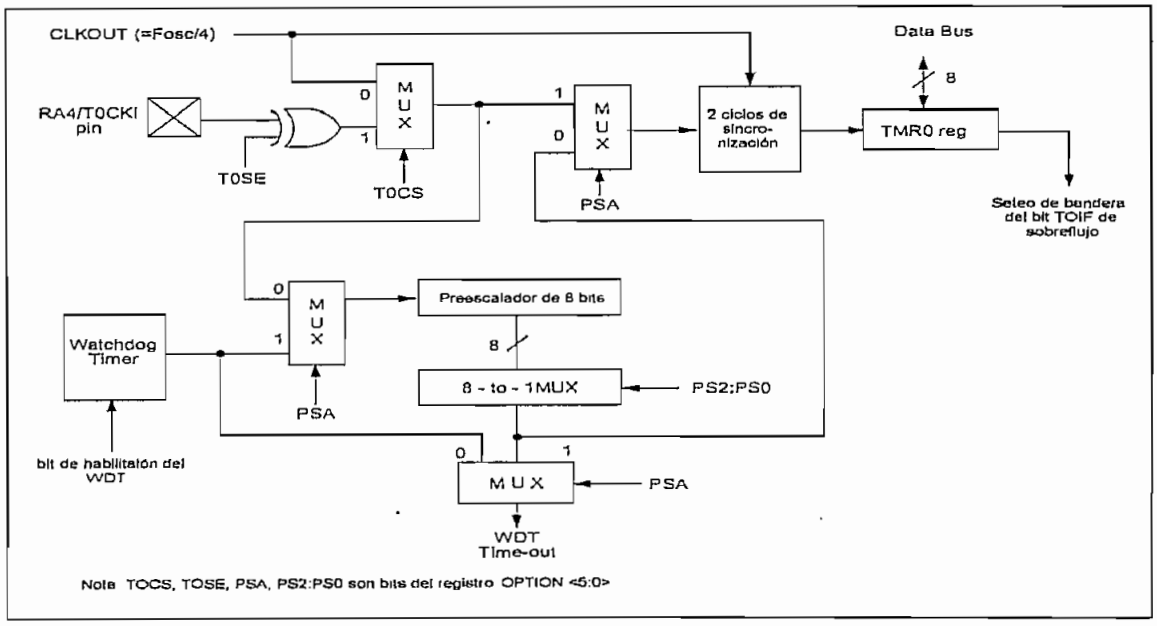


Figura. 1.45 Diagrama de bloques del preescalador del TIMER0/WDT

Además la asignación del preescalador es completamente controlado por software, esto permite que se pueda cambiar la asignación del preescalador durante la ejecución del programa.

02h o PCL : Contador de programa.

Se utiliza para direccionar las palabras de 14 bits del programa del usuario que se encuentra almacenado en la memoria ROM, es decir contiene la dirección de la instrucción a ser ejecutada; este contador de programa es de 13 bits de ancho (Figura 1.46). Sobre el byte bajo (PCL), se puede escribir o leer directamente, mientras que sobre el byte alto, no. El byte alto se maneja mediante el registro PCLATH (0Ah). A diferencia de los PIC de primera generación, el

16C71 ante una condición de reset inicia el contador de programa con todos sus bits en "cero". Durante la ejecución normal del programa, y dado que todas las instrucciones ocupan sólo una posición de memoria, el contador se incrementa en uno con cada instrucción, a menos que se trate de alguna instrucción de salto.

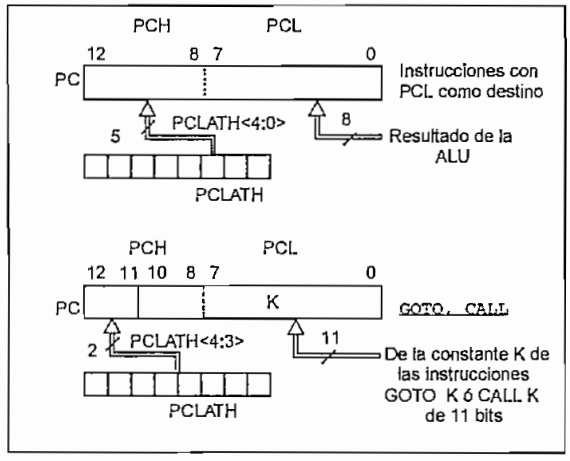


Figura 1.46. Contador de programa (13 bits)

En una instrucción CALL o GOTO, los bits PC.<l0:0> se cargan desde el código de operación de la instrucción, mientras que los bits PC<12:11> lo hacen desde el PCLATH<4:3>. Como solamente el primer 1K de memoria está implementado, el código de operación de la instrucción puede contener la dirección destino, eso quiere decir que se pueden hacer saltos y llamados a subrutinas sin necesidad de tener en cuenta la paginación de memoria de programa.

En otras instrucciones donde PCL es el destino, PC<12:8> se carga directamente desde el PCLATH<4:0>, por ejemplo en el caso de la instrucción ADDWF. Esto se debe tener en cuenta cuando se desea hacer lectura de tablas usando el comando: ADDWF PC, 1; en este caso se debe tener en cuenta que la tabla debe estar comprendida dentro de un solo bloque de 256 bytes (0-255, 256-511, etc.).

O3h ó STATUS: Registro de estados.

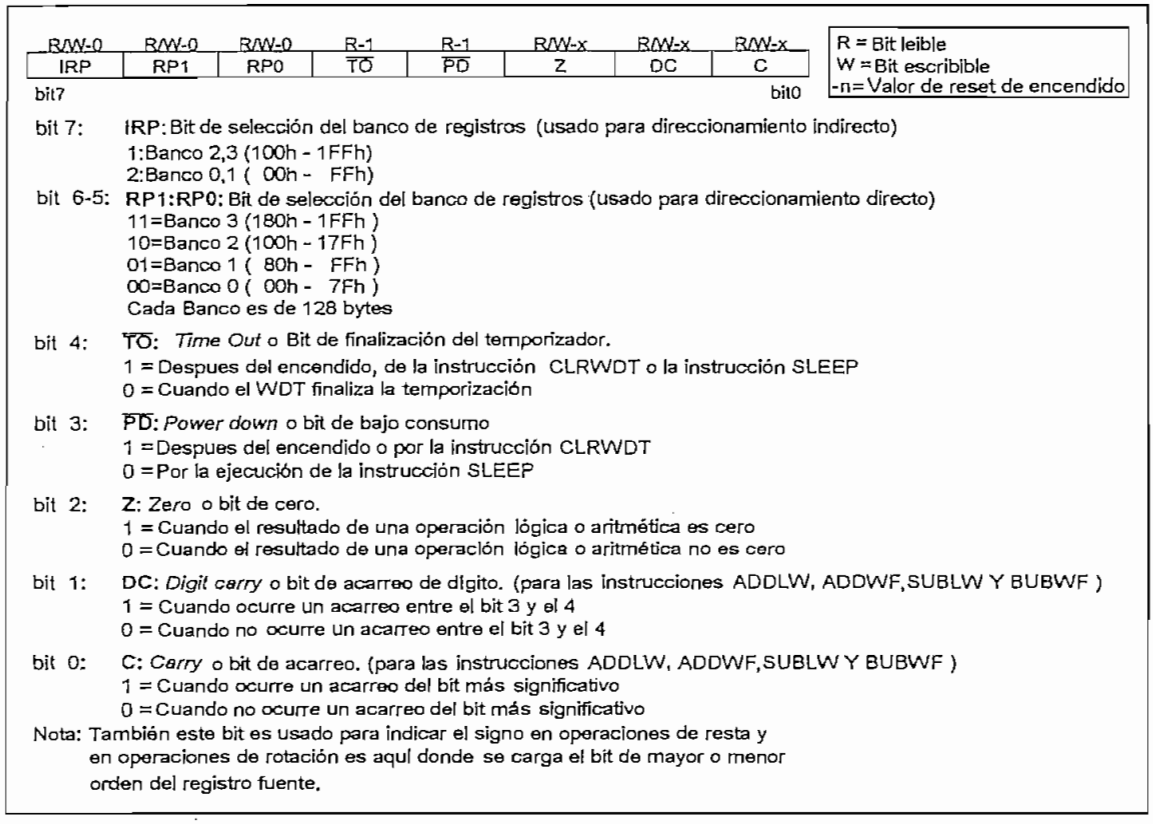
Contiene el estado aritmético de la ALU, la causa del reset y los bits de preselección de página para la memoria de datos. La Figura 1.47 muestra los bits correspondientes a este registro. Los bits 5 y 6 (RPO y RP1) son los bits de selección de página para el direccionamiento directo de la memoria de datos;

solamente RPO se usa en el PIC16C71. RP1 se puede utilizar como un bit de propósito general de lectura/escritura. Los bits TO y PD no se pueden modificar por un proceso de escritura; ellos muestran la condición por la cual se ocasionó el último reset.

Se debe tener en cuenta que al escribir sobre el registro STATUS no se obtendrá posiblemente el resultado esperado, pues sus bits cambian según el resultado de las operaciones, por ejemplo con la instrucción CLRF STATUS se borrarán los tres bits altos y el bit Z se seteará. Los bits IRP y RP1 no son utilizados en el PIC16C71 y no se recomienda utilizarlos como bits de propósito general, pues a futuro podría generar incompatibilidades al migrar a otros microcontroladores.

El bit C y DC operan como un llevo en las operaciones de resta, esto se puede entender al revisar las instrucciones SUBLW y SUBWF.

Cuando el registro STATUS es el destino de una instrucción que afecte a los bits Z, DC o C, entonces la escritura de estos tres bits se inhabilita. Estos bits se actualizarán de acuerdo a la lógica del sistema.



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	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit7							bit0

R = Bit leible
W = Bit escribible
-n= Valor de reset de encendido

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

bit 6-5: **RP1:RP0:** Bit de selección del banco de registros (usado para direccionamiento directo)
11=Banco 3 (180h - 1FFh)
10=Banco 2 (100h - 17Fh)
01=Banco 1 (80h - FFh)
00=Banco 0 (00h - 7Fh)
Cada Banco es de 128 bytes

bit 4: $\overline{TO}$: *Time Out* o Bit de finalización del temporizador.
1 = Despues del encendido, de la instrucción CLRWDT o la instrucción SLEEP
0 = Cuando el WDT finaliza la temporización

bit 3: $\overline{PD}$: *Power down* o bit de bajo consumo
1 = Despues del encendido o por la instrucción CLRWDT
0 = Por la ejecución de la instrucción SLEEP

bit 2: **Z:** *Zero* o bit de cero.
1 = Cuando el resultado de una operación lógica o aritmética es cero
0 = Cuando el resultado de una operación lógica o aritmética no es cero

bit 1: **DC:** *Digit carry* o bit de acarreo de dígito. (para las instrucciones ADDLW, ADDWF,SUBLW Y BUBWF)
1 = Cuando ocurre un acarreo entre el bit 3 y el 4
0 = Cuando no ocurre un acarreo entre el bit 3 y el 4

bit 0: **C:** *Carry* o bit de acarreo. (para las instrucciones ADDLW, ADDWF,SUBLW Y BUBWF)
1 = Cuando ocurre un acarreo del bit más significativo
0 = Cuando no ocurre un acarreo del bit más significativo

Nota: También este bit es usado para indicar el signo en operaciones de resta y en operaciones de rotación es aquí donde se carga el bit de mayor o menor orden del registro fuente.

Figura 1.47 Registro de estados

04h o FSR: Registro selector de registros.

En asocio con el registro IND0, se utiliza para seleccionar indirectamente los otros registros disponibles, como puntero para direccionar indirectamente la memoria de datos. Si en el programa no se utilizan llamadas indirectas, este registro se puede utilizar como un registro de propósito general. Para entender mejor el funcionamiento de este registro se lista un programa simple que borra el contenido de la memoria RAM, empleando direccionamiento indirecto.

- El registro 05 contiene el valor 10h
- El registro 06 contiene el valor 0Ah
- Cargamos el valor 05 al registro FSR
- Entonces al leer el registro INDF se obtendrá el valor 10h
- Incrementamos en uno el registro FSR (FSR = 06)
- Entonces al leer el registro INDF ahora se obtendrá el valor 0Ah

Leyendo indirectamente el mismo registro INDF (FSR = 0) se obtendrá 00h. Escribiendo indirectamente en el registro INDF resulta en una operación no efectuada, pero los bits del registro STATUS pueden ser afectados. El bit IRP del registro STATUS funciona como el noveno bit del registro FSR, como se muestra en la Figura 1.48. Sin embargo este bit no es usado en el PIC16C71.

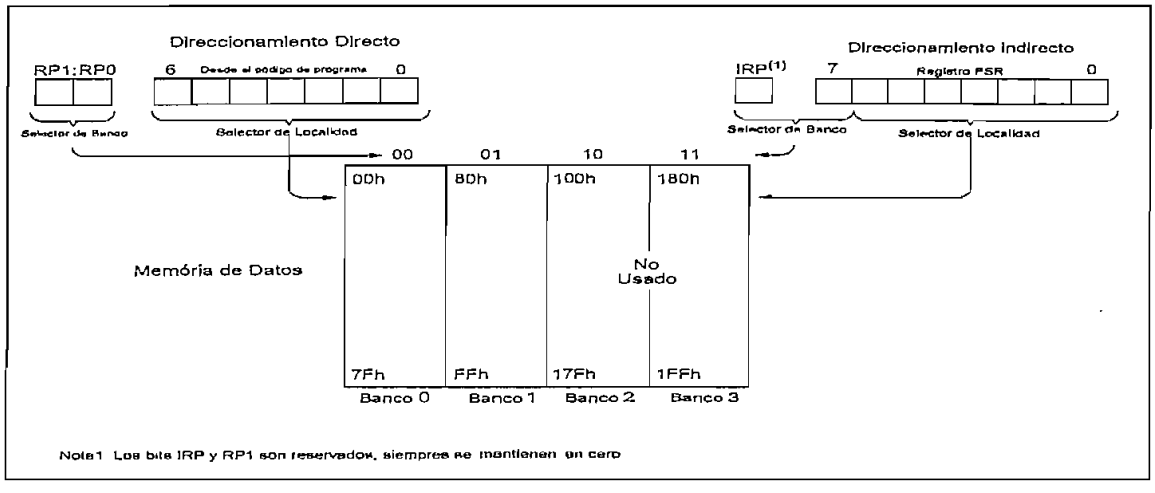


Figura. 1.48 Direccionamiento directo e indirecto

05h o PORTA: Puerto de Entrada/Salida de 5 bits.

Este puerto tiene 5 bits de ancho y al igual que todos sus similares en los PIC, puede leerse o escribirse como si se tratara de un registro cualquiera.

Los pines de este puerto, además de comportarse como entradas y salidas digitales, tienen funciones alternas, las cuales se muestran en la Tabla 1.14 mostrada a continuación.

PIN	FUNCIÓN ALTERNA
RA0 / AIN0	Entrada analógica Canal 0
RA1 / AIN1	Entrada analógica Canal 1
RA2 / AIN2	Entrada analógica Canal 2
RA3 / AIN3 / Vref	Entrada analógica Canal 3 o Voltaje de referencia (Vref)
RA4 / TOCK1	Entrada externa de reloj para contador TMR0

Tabla 1.14 Funciones alternas de los pines del puerto A

Dos bits en el registro ADCON1 (registro 88h) configuran los cuatro pines menos significativos como E/S digitales o entradas analógicas. El registro que controla el sentido (entrada o salida) de los pines de este puerto está localizado en la página 1, en la posición 85h y se llama TRISA, siempre y cuando estos pines estén siendo utilizados como entrada digitales. Luego de un reset de encendido, estos pines son configurados como entradas analógicas y al leerlos se obtiene como resultado cero. Al realizar una operación de lectura se lee el estado del pin, y al realizar una operación de escritura se carga en la memoria (*latch)* del puerto.

El pin RA4/TOCK1 es una entrada Schmitt Trigger y salida de drenaje abierto. Todos los otros pines tienen entradas de nivel TTL y drivers de salida CMOS. Además este pin se multiplexa con el módulo Timer0

06h o PORTB: Puerto de entrada/salida de 8 bits.

Este puerto tiene 8 bits de ancho, al igual que en todos los PIC, este puede leerse o escribirse como si se tratara de un registro cualquiera; algunos de sus pines tienen funciones alternas en la generación de interrupciones. El registro de control para la configuración de la función de sus pines se localiza en la página 1, en la dirección 86h y se llama TRISB.

08h o ADCON0: Registro de control del convertidor.

Este registro, junto con el ADCON1 se dedica al control del convertidor análogo a digital. Los bits que éste contiene y la función que cumplen se muestran en la Figura 1.49 a continuación.

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADCS1	ADCS0	—(1)	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADIF	ADON
bit7							bit0

R = Bit leíble
W = Bit escribible
U = Bit no Implementado, se lee como '0'.
- n= Valor al reset POR.

bit 7-6: **ADCS1:ADCS0:** Bits de selección del reloj del conversor A/D
00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (Reloj derivado desde un oscilador RC)

bit 5: No implementado: se lee como '0'.

bit 4-3: **CHS1:CHS0:** Bit de selección del canal Analógico
00 = canal 0, (RA0/AN0)
01 = canal 1, (RA1/AN1)
10 = canal 2, (RA2/AN2)
11 = canal 3, (RA3/AN3)

bit 2: **GO/$\overline{\text{DONE}}$:** Bit del estado del conversor A/D

SI ADON = 1:
1 = Convertidor A/D en progreso (Seteando este bit se inicia la conversión A/D)
0 = Conversión A/D detenida (Este bit es automáticamente borrado por hardware cuando la conversión es completada)

bit 1: **ADIF:** Bandera de finalización de la conversión A/D
1 = conversión completa (Debe ser borrada por software)
0 = conversión incompleta

bit 0: **ADON:** Bit de encendido del conversor A/D
1 = Módulo del conversor A/D en operación
0 = Módulo del conversor A/D apagado y no consume corriente

Nota 1: El bit 5 del ADECON0 puede ser utilizado como bit de propósito general para los PIC16C710/711 solamente. Para el PIC16C71, este bit no es implementado, y se lee como '0'.

Figura. 1.49 Registro ADCON0 de control del convertidor

09h o ADRES: Registro para el resultado de la conversión.

En esta posición se almacena el resultado digital de la conversión del valor análogo.

OAh o PCLATH: Registro para la parte alta de la dirección.

Este registro es un buffer de los 5 bits altos del contador de programa. El registro alto del contador de programa es llamado PCH y no es leíble ni escribible, El PCLATH es un registro esclavo del PCH (PC<12:8>). El contenido del PCLATH puede ser transferido a la parte alta del contador de programa, pero el contenido del PC<12:8> no se lo puede transferir al PCLATH.

OBh o INTCON: Registro para el control de interrupciones.

Es el encargado del manejo de las interrupciones y contiene los bits que se muestran en la Figura 1.50. Este registro es leíble y escribible, conteniendo los bits de habilitación de todas las fuentes de interrupción, así como las banderas que indican la causa de la interrupción.

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	ADIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							bit0

R = Bit Leible
W = Bit escribible
- n = Valor luego del reset de encendido

bit 7: **GIE:** *Global Interrupt Enable* o Habilitador general de interrupciones
1 = Habilita todas las interrupciones
0 = Inhabilita las interrupciones

bit 6: **ADIE:** Bit de habitación de la interrupción del conversor A/D
1 = La habilita
0 = La Inhabilita

bit 5: **T0IE:** *TMR0 overflow Interrupt Enable* o Habilitación de interrupción del temporizador TMR0
1 = Habilita la interrupción del TMR0
0 = Inhabilita la interrupción del TMR0

bit 4: **INTE:** *RB0/INT Interrupt Enable bit* o bit de habilitación de la interrupción INT (RB0)
1 = Habilita la interrupción RB0/INT
0 = Inhabilita la interrupción RB0/INT

bit 3: **RBIE:** *RB Port Change Interrupt Enable bit* o bit de habilitación de interrupción por cambios en el puerto B (RBIF)
1 = Habilita la interrupción por cambio en el puerto RB
0 = Inhabilita la interrupción por cambio en el puerto RB

bit 2: **T0IF:***TMR0 Overflow Interrupt flag bit* ó Bandera de interrupción por sobreflujo del TMR0
1 = Cuando TMR0 pasa de FFh a 00h (Esta debe ser puesta en 0 por programa)
0 = Cuando el TMR0 no llegó al sobreflujo

bit 1: **INTF:** *RB0/INT Interrupt flag bit* ó Bandera de interrupción INT (RB0)
1 = Ocurrió la interrupción RB0/INT
0 = Cuando no se produjo la interrupción RB0/INT (debe ser puesta en '0' por programa)

bit 0: **RBIF:** *RB Port Change Interrupt flag bit* ó Bandera de interrupción por cambio en el puerto B
1 = Cuando se produjo un cambio en las entradas RB<4:7> (debe ser puesta en '0' por programa)
0 = Cuando ninguno de los pines RB7:RB4 han cambiado de estado.

Figura 1.50 Registro INTCON (Direcciones 0Bh, 8Bh)

81h u OPTION: Registro de configuración múltiple.

Este registro es leíble y escribible. Posee varios bits para configurar el preescalador TMR0/WDT, la interrupción externa INT, el temporizador TMR0 y las pull-ups del puerto B. Los bits que contiene y las funciones que realiza este registro se muestran en la Figura 1.51. El preescalador es compartido entre el MTRO y el WDT; su asignación es mutuamente excluyente ya que solamente puede uno de ellos ser preescalado a la vez.

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit7							bit0

R = Bit lelble
W = Bit escribible
- n = Valor despues del reset de encendido

bit 7: $\overline{\text{RBPU}}$: *PortB Pull-up Enable bit* o bit de habilitación de *Pull-up* del puerto B.
1 = Las pull-ups del puerto B son inhabilitadas
0 = Las pull-ups del puerto B son habilitadas

bit 6: INTEDG: *Interrupt Edge Select bit* o bit selector del flanco de la interrupción INT.
1 = La interrupción se da por el flanco de subida del pin RB0/INT
0 = La interrupción se da por el flanco de bajada del pin RB0/INT

bit 5: T0CS: *TMR0 Clock Source Select bit* o bit de selección de señal de reloj del TMR0.
1 = Transición en el pin RA4/T0CKI (Contador)
0 = Ciclo de instrucciones interno (Temporizador)

bit 4: T0SE: *TMR0 Source Edge Select bit* o bit selector del flanco de la fuente del TMR0
1 = Incremento de transición de alto a bajo en el pin RA4/T0CKI
0 = Incremento en transición de bajo a alto en el pin RA4/T0CKI

bit 3: PSA: *Prescaler Assignment bit* o bit de asignación del preescalador
1 = Preescalador asignado al WDT
0 = Preescalador asignado al TMR0

bit 2-0: PS2:PS0: *Prescaler Rate Select bits* o bits de selección de la proporción del preescalador

Valor de bits	Proporción del TMR0	Proporción 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

Figura. 1.51 Registro OPTION (Dirección 81h)

85h o TRISA: Registro de configuración del puerto A.

Registro de control para el puerto A. Un "cero" en el bit correspondiente al pin lo configura como salida y pone el contenido de la memoria *(latch)* en el pin seleccionado, mientras que un "uno" lo configura como entrada y pone el correspondiente circuito de salida en alta impedancia.

86h o TRISB: Registro de configuración del puerto B.

Orientado hacia el control del puerto B. Son válidas las mismas consideraciones del registro anterior.

88h o ADCON1: Registro auxiliar para el control del convertidor.

Este es el registro se utiliza para la configuración de entradas analógicas y solo destina dos bits para ello; el resto de bits permanecen sin implementar. En la

Figura 1.52 se muestran las funciones de estos bits.

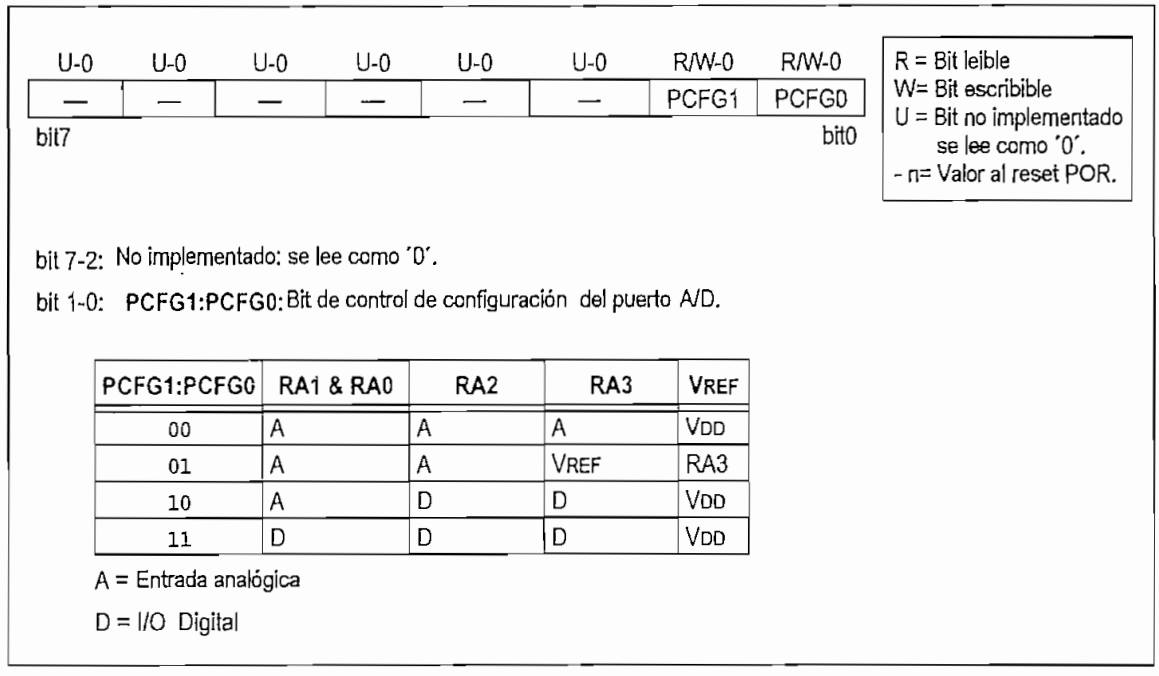


Figura. 1.52 Registro ADCON1 (Dirección 88h)

OCh a 2Fh: Registros de propósito general.

Estas 36 posiciones están implementadas en la memoria RAM estática, que conforma el área de trabajo del usuario; a ellas también se accede cuando en la página 1 se direccionan las posiciones 8Ch a AFh. Esto se ha diseñado así para evitar un excesivo cambio de páginas en el manejo de la RAM del usuario, agilizando los procesos que se estén llevando a cabo y descomplicando la labor de programación.

1.3.9 Registro de trabajo W.

Este es el registro de trabajo principal, se comporta de manera similar al acumulador en los microprocesadores. Este registro participa en la mayoría de las instrucciones.

1.3.10 Pila (Stack)

Estos registros no forman parte de ningún banco de memoria y no permiten el acceso por parte del usuario. Se usan para guardar el valor del contador de programa cuando se hace un llamado a una subrutina o cuando se atiende una

interrupción; luego, cuando el micro regresa a seguir ejecutando su tarea normal, el contador de programa recupera su valor leyéndolo nuevamente desde la pila. El PIC16C71 tiene una pila de 8 niveles, esto significa que se pueden anidar 8 llamados a subrutina sin tener problemas, y al igual que el contador de programa tiene 13 bits de ancho.

1.3.11 Características especiales

Estas características son muy útiles para la lograr equipos de alta calidad y mínimo costo a través de la eliminación de componentes externos, por esto se debe aprender a utilizarlos. A continuación se detallan dichas características:

Convertidor Análogo a Digital

El conversor análogo a digital (A/D) tiene cuatro entradas analógicas multiplexadas a un solo circuito de muestreo, sostenimiento y a un convertidor. Este módulo conversor A/D permite obtener un valor digital de 8 bits de una señal de entrada analógica. El voltaje de referencia del convertidor se puede configurar por software para que sea el voltaje de alimentación del microcontrolador (VDD) o un voltaje externo aplicado al pin RA3/AN3/VREF. En este caso, el valor mínimo que puede tener el voltaje de referencia es de 3 voltios. Este convertidor genera el resultado vía aproximaciones sucesivas el tiempo de conversión está en función del ciclo del oscilador, considerándose que el tiempo mínimo es de 20us.
El conversor A/D tiene una característica única de ser capaz de convertir mientras el equipo está en modo SLEEP. Para operar en este modo, el reloj debe ser derivado desde el oscilador interno RC para el conversor.
El módulo A/D tiene tres registros. Estos registros son:

- Registro de resultados A/D (ADRES)
- Registro 0 de control A/D (ADCON0)
- Registro 1 de control A/D (ADCON1)

El registro ADCON0 se muestra en la Figura 1.49 y controla la operación del módulo A/D. El registro ADCON1, se muestra en la Figura 1.52 y configura las funciones de los pines del puerto como entradas analógicas, RA3 como voltaje de referencia o como entradas o salidas digitales.

La conversión se inicia colocando el bit de control GO/DONE (ADCON0<2>) en ´1´, previa la selección del canal deseado, el correspondiente bit TRIS debe ser seleccionado como entrada y de configurar el registro ADCON1, el cual define los modos en los cuales trabaja el convertidor. Al final de la conversión, el bit GO/DONE se coloca en ´0´ y la bandera de interrupción ADIF del convertidor (ADECON0<1>) en ´1´, almacenándose el resultado de la conversión A/D en el registro ADRES. Estas características permiten que el resultado de la conversión se pueda leer por medio de interrupciones o sin ellas.
El diagrama de bloques del módulo convertidor A/D se muestra en la Figura 1.53 a continuación:

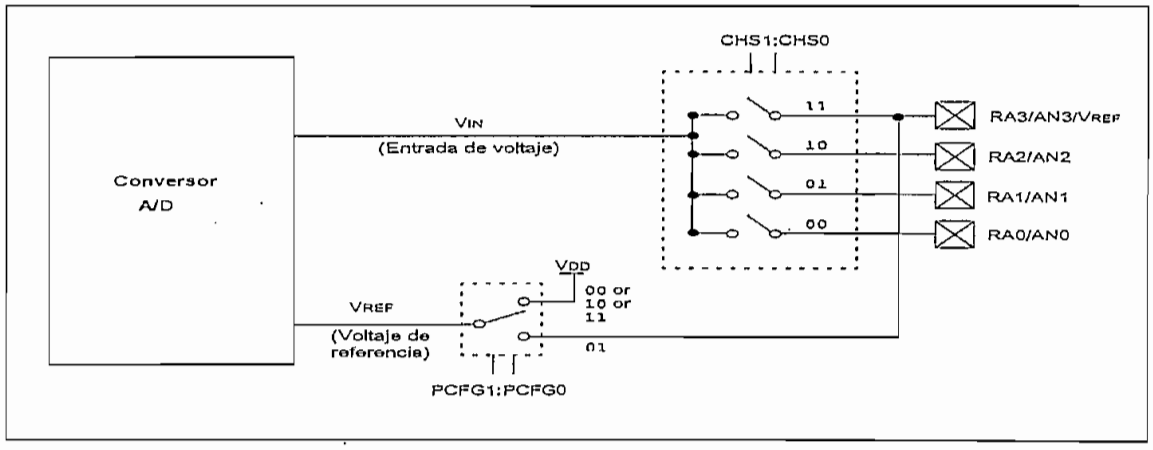


Figura. 1.53 Diagrama de bloques del módulo convertidor A/D

Para realizar la conversión se debe seguir los siguientes pasos:

1 Configuración del módulo A/D:

- Configurar como pines análogos / voltaje de referencia / y entradas - salidas digitales (ADOCON1)
- Seleccionar el canal de entrada A/D (ADCON0)
- Seleccionar el reloj de conversión A/D (ADCON0)
- Encender el módulo A/D (ADCON0)

2 Configurar la interrupción A/D (si se desea utilizarla):

- Borrar el bit ADIF
- Setear el bit ADIE
- Setear el bit GIE

3 Esperar el tiempo de adquisición requerido:

4 Iniciar la conversión:

- Setear el bit GO/DONE (ADCON0)

5 Esperar que se complete la conversión por uno de los dos indicadores:

- Análisis del bit GO/DONE si cambio a "cero". ó
- Espera hasta que se genere la interrupción

6 Lectura del resultado de la conversión en el registro (ADRES), se requiere que se borre el bit ADIF.

7 Para la siguiente conversión, configurar en los pasos 1 y 2 como sea requerido. El tiempo de conversión por bit es definido como TAD. Para iniciar la siguiente conversión se requiere esperara un tiempo mínimo de 2TAD.

El tiempo de adquisición es el requerido para la carga del condensador CHOLD debido a que es necesario que sea cargado completamente al nivel de voltaje del canal de entrada. La Figura 1.54 muestra el modelo del canal de entrada.

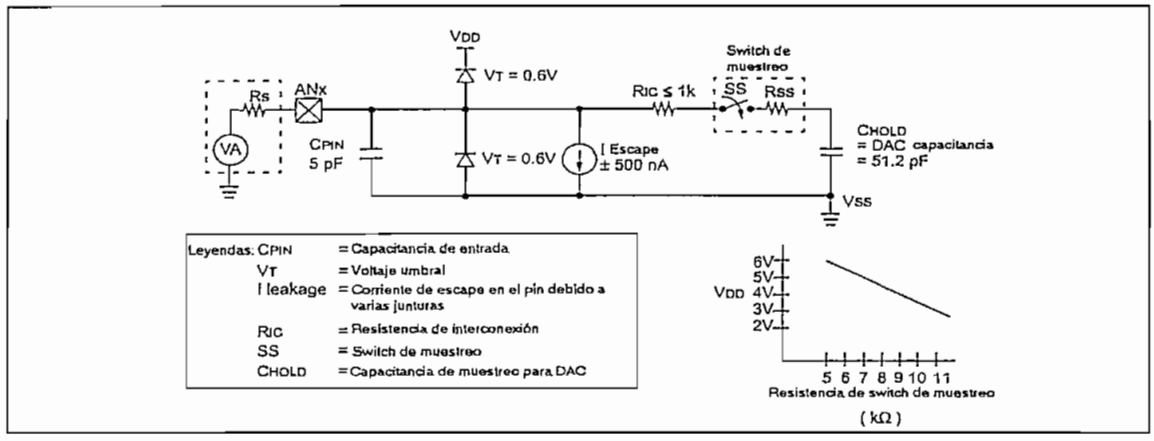


Figura. 1.54 Modelo del canal entrada analógica

La impedancia de la fuente (Rs) y la impedancia del switch de muestreo (Rss) afectan directamente al tiempo requerido para la carga del capacitor CHOLD. La impedancia del switch de muestreo varia en función del voltaje de la fuente VDD. La impedancia de la fuente afecta el voltaje offset (referencia) de la entrada analógica debido a la corriente de escape; el valor máximo recomendado es de 10KΩ. Después de que el canal de entrada ha sido seleccionado se debe

completar la adquisición de la señal antes de que la conversión pueda ser iniciada.

Para calcular el tiempo mínimo de adquisición se debe utilizar las ecuaciones siguientes que tiene un error de ½ LBS (512 pasos) para el coversor A/D que es el máximo permitido.

Tiempo de adquisición:

TACQ = Tiempo de arreglo del amplificador + tiempo de carga del capacitor CHOLD + Tiempo del coeficiente de temperatura.

Tiempo de carga del capacitor CHOLD: TCA= - (51.2 pF) (1 KΩ + Rss + Rs) ln (1/511)

Como ejemplo se muestra a continuación el calculo del tiempo mínimo requerido TACQ, calculado en base de los siguientes parámetros: CHOLD= 51.2 pF, Rs=10KΩ, error de ½ LBS, VDD=5v → Rss =7KΩ., Temperatura máxima aplicada al sistema=50°C., VHOLD = 0 @ T= 0.

TACQ = 5 µs + TCA + [(Temp – 25°C) (0.05 µs /°C)]

TCA = - CHOLD (RIC + Rss +Rs) Ln (1/511)

= - 51.2 pF (1KΩ + 7KΩ + 10KΩ) Ln (0.0020)

= - 51.2 pF (18KΩ) ln (0.0020)

= - 0.921µs (-6.2364)

=5.747µs

TACQ = 5µs + 5.747 µs + [(50°C – 25°C)(0.05µs/°C)]

= 10.747 µs + 1.25 µs

= 11.997µs

La fuente de reloj del conversor A/D es seleccionada por software. El tiempo de la conversión por bit se define como TAD, entonces el conversor A/D requiere 9,5 TAD por los 8 bits. Las cuatro posibles opciones para TAD son:

- 2 TOSC
- 8 TOSC
- 32 TOSC
- oscilador RC interno

- oscilador RC interno

Para una correcta conversión, el reloj del conversor (TAD) debe ser seleccionado para asegurar un mínimo tiempo TAD de 2µs para el PIC16C71 y de 1,6µs para todos los otros PIC16C71X.
La Tabla 1.15 muestra el resultado del tiempo TAD derivado de la frecuencia de operación y de la fuente de reloj seleccionada.

Fuente del reloj de conversor A/D (TAD)		Frecuencia del equipo				
Operación:	ADCS1:ADCS0	20 MHz	16 MHz	4 MHz	1 MHz	333.33 kHz
2TOSC	00	100 ns	125 ns	500 ns	2.0 µs	6 µs
8TOSC	01	400 ns	500 ns	2.0 µs	8.0 µs	24 µs
32TOSC	10	1.6 µs	2.0 µs	8.0 µs	32.0 µs	96 µs
RC	11	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs	2 - 6 µs

Tabla 1.15 TAD vs. Frecuencia de operación para el PIC16C71

Como un ejemplo se muestra a continuación el programa para llevar a cabo la conversión por el pin RA0 (canal 0); los pines del RA son configuradas como entradas analógicas, en este caso el voltaje de referencia (Vref) es el mismo de alimentación VDD del microcontrolador. Se habilita la interrupción A/D y el reloj de conversión seleccionado es FRC.

```
        BSF     STATUS,  RP0   ; Selección del banco 1
        CLRF    ADCON1         ; Configuración de entradas
        BCF     STATUS,  RP0   ; Selección del banco 0
        MOVLW   0xC1           ; Reloj RC, Conversor en ON y
        MOVWF   ADECON0        ; selección del canal 0
        BSF     INTCON,  ADIF  ; Habilita la interrupción A/D
        BSF     INTCON,  GIE   ; Habilita todas las interrupciones
;
;       Asegúrese que el tiempo requerido de muestreo para el canal seleccionado
;       de entrada termine, entonces la conversión puede iniciarse.
;
        BSF     ADCON0, GO     ; Inicia la conversión A/D
        :                      ; El bit ADIF será seteado y el bit
        :                      ; GO/DONE será borrado al completarse
```

El bit GO/DONE no debe ser seteado en la misma instrucción que enciende el conversor A/D. Si se borra el bit GO/DONE durante una conversión causará que se interrumpa la conversión actual y el registro ADRES no actualizará con el valor incompleto de la conversión interrumpida sino que se mantendrá con el valor de la conversión anterior o con el valor escrito en este. Después que la conversión es abortada, se debe esperar un tiempo de 2TAD antes de que la siguiente conversión se pueda iniciar. Después de transcurrido un tiempo de 2TAD, la adquisición es automáticamente iniciada sobre el canal seleccionado.

El conversor A/D puede operar en modo SLEEP, para esto se requiere que la fuente de reloj escogida sea RC (ADCS1:ADCS0= 11). Cuando es seleccionado el reloj RC, el módulo A/D espera un ciclo de instrucción antes de iniciar la conversión. Esto permite que la instrucción SLEEP sea ejecutada, lo cual elimina todos los ruidos de switcheo digital desde el conversor. Cuando se completa la conversión el bit GO/DONE será borrado y el resultado almacenado en el registro ADRES. Si la interrupción A/D es habilitada, el microcontrolador saldrá del modo de reposo. Si la interrupción no está habilitada el conversor A/D se apagará, aunque el bit ADON permanezca seteado.

Cuando la fuente de reloj no es RC, la instrucción SLEEP causará que la conversión presente sea abortada y el módulo A/D se apague, aunque el bit ADON se mantenga seteado.

Circuito de vigilancia (Watchdog Timer WDT)

Su función es restablecer el programa cuando éste se ha perdido por fallas en la programación o por alguna razón externa. Es muy útil cuando se trabaja en ambientes con mucha interferencia o ruido electromagnético. Está conformado por un oscilador RC que se encuentra dentro del microcontrolador. Este oscilador corre de manera independiente al oscilador principal. Cuando se habilita su funcionamiento poniendo un ´1´ en el bit de configuración WDTE, dicho circuito hace que el microcontrolador sufra un reset cada determinado tiempo (que se puede programar entre 18 ms y 2,3 sg mediante el preescalador). Este reset se puede evitar el usuario mediante una instrucción especial del microcontrolador (**CLRWDT**: borrar el conteo del watchdog), la cual se debe ejecutar antes de que termine el período nominal de dicho temporizador. De esta manera, si el programa

se ha salido de su flujo normal, por algún ruido o interferencia externa, el sistema se reiniciará (cuando se acabe el tiempo programado y no se haya borrado el contador) y el programa puede restablecerse para continuar con su funcionamiento normal. El diagrama de bloques del WDT se puede observar en la Figura 1.45.

Temporizador de encendido (Power-up Timer)

Este proporciona un reset al microcontrolador en el momento de conectar la fuente de alimentación, lo que garantiza un arranque correcto del sistema. En el momento de grabar el microcontrolador se debe habilitar el fusible de configuración "Power-up Timer", para ello se debe seleccionar la opción "ON". Su tiempo de retardo es de 72 milisegundos.

Modo de bajo consumo (sleep)

Esta característica permite que el microcontrolador entre en un estado pasivo donde consume muy poca potencia, y posteriormente salir de este estado a operación normal. Si el temporizador del circuito de vigilancia (*Watchgod Timer)* está habilitado este se reinicia y empieza el conteo de nuevo, El bit PD es borrado y el bit TO es puesto en ´1´ y se detiene el oscilador principal, deteniendo la ejecución de instrucciones. Los puertos de entrada/salida mantienen la configuración que tenían antes de la ejecución de la instrucción SLEEP.

Para un muy bajo consumo de corriente en el modo SLEEP, se debe colocar a VDD o VSS todos pines que no tengan drenaje de corriente a un circuito externo desde el pin, se debe inhabilitar el reloj externo y apagar el módulo conversor A/D. Los pines que son entradas de alta impedancia deben ser colocadas alto o bajo externamente para evitar corrientes de oscilación causadas por entradas flotantes. La entrada TOCKI también debe ser conectada a VDD o VSS. También se debe considerar a las resistencias *pull-up* internas del puerto B.

El pin MCLR debe estar colocado a un nivel lógico alto, tomando en cuenta que el Reset del WDT por finalización de temporización no funciona cuando el MCLR está conectado a un nivel lógico bajo.

El microcontrolador sale del modo de bajo consumo SLEEP por una de las siguientes causas:

- Reset externo en el pin MCLR
- Finalización del conteo del temporizador WDT
- Interrupción en el pin RB0/INT, cambio en el puerto RB, o las interrupción de periféricos internos: TMR0 que debe ser programado como contador y el conversor A/D (con la fuente de reloj RC).

Los bits TO y PD pueden ser usados para determinar la causa del reset del equipo. El bit PD que se pone en ´0´ cuando entra en el modo SLEEP, al salir de este es colocado en ´1´ . El bit TO es puesto en ´0´ si se finaliza el conteo del temporizador de vigilancia WDT (y causa que salga del modo de reposo SLEEP). Mientras la instrucción SLEEP está siendo ejecutada, la siguiente instrucción (PC+1) es leída. Para que un evento de interrupción saque al sistema del reposo el correspondiente bit de habilitación de interupciones (GIE) debe ser puesto en ´1´. Las operaciones posteriores al salir del modo de reposo dependen del bit GIE. Si el bit GIE está en ´0´ es decir inhabilitadas las interrupciones, entonces el equipo continuará ejecutando la instrucción posterior al SLEEP. Si el bit GIE está en ´1´ es decir habilitadas las interrupciones, entonces el equipo ejecutará la instrucción siguiente a la instrucción SLEEP y luego salta al vector de interrupción (0004h). En el caso en que la instrucción siguiente al SLEEP no es deseada, esta debe ser una instrucción NOP. En la Figura 1.55 se muestra el funcionamiento de la instrucción SLEEP.

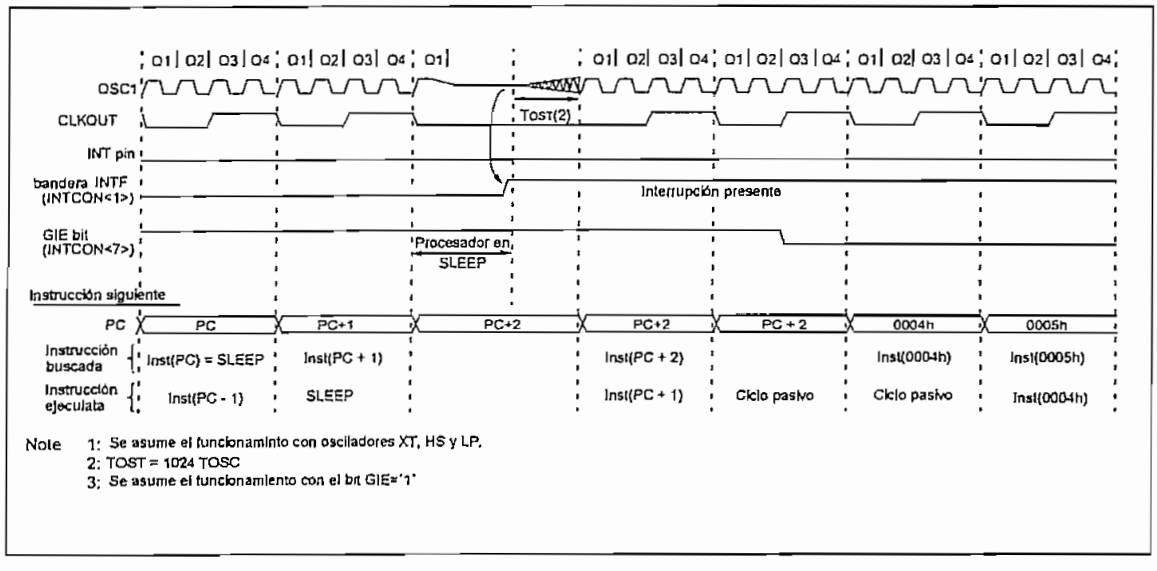


Figura. 1.55 Entrada y salida del modo de bajo consumo a través de una interrupción

Interrupciones

Este microcontrolador incluye el manejo de interrupciones, lo cual representa grandes ventajas. El PIC16C71 posee cuatro fuentes de interrupción:

- Interrupción externa en el pin RB0/INT
- Finalización del temporizador/contador TMRO
- Finalización de la conversión A/D
- Cambio de nivel en los pines RB4 a RB7

El registro 0Bh o INTCON contiene las banderas de las interrupciones INT, cambio en el puerto B y finalización del conteo del TMRO, al igual que el control para habilitar o inhabilitar cada una de las fuentes de interrupción, incluida la de fin de conversión análogo a digital. Sólo la bandera de fin de conversión (bit ADIF) reside en el registro 08h o ADCON0.

Si el bit GIE (Global Interrupt Enable) se coloca en 0, inhabilita todas las interrupciones. Cuando una interrupción es atendida, el bit GIE se coloca en 0 automáticamente para evitar interferencias con otras interrupciones que se pudieran presentar, la dirección de retorno se coloca en la pila y el PC se carga con la dirección 04h. Una vez en la rutina de servicio, la fuente de la interrupción se puede determinar examinando las banderas de interrupción. La bandera respectiva se debe colocar, por software, en cero antes de regresar de la interrupción, para evitar que se vuelva a detectar nuevamente la misma interrupción.

La instrucción RETFIE permite al usuario retornar de la interrupción, a la vez que habilita de nuevo las interrupciones, al colocar el bit GIE en uno. Debe tenerse presente que solamente el contador de programa es puesto en la pila al atenderse la interrupción; por lo tanto, es conveniente que el programador tenga cuidado con el registro de estados y el de trabajo, ya que se pueden producir resultados inesperados si dentro de ella se modifican. En la Figura 1.56 se

observa el circuito lógico de atención a interrupciones y de salida del modo de bajo consumo SLEEP.

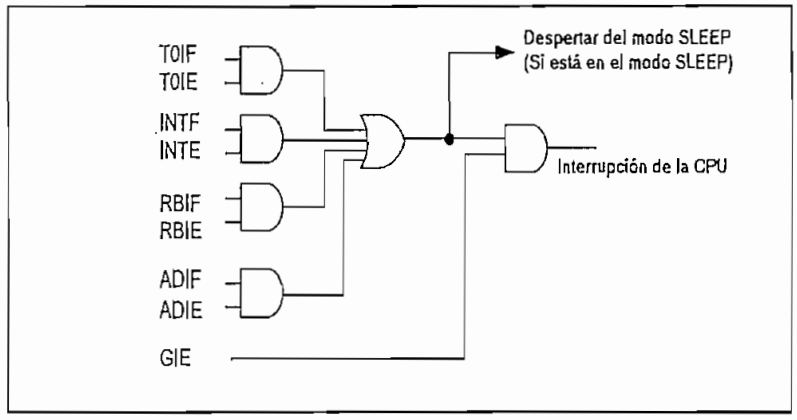


Figura. 1.56 Lógica de interrupciones

- **Interrupción externa.**

Actúa sobre el pin RB0/INT y se puede configurar para activarse con el flanco de subida o el de bajada, de acuerdo al bit INTEDG (OPTION<6>). Cuando se presenta un flanco válido en el pin INT, la bandera INTF (INTCON<1>) se coloca en uno. La interrupción se puede inhabilitar colocando el bit de control INTE (INTCON<4>) en cero. Cuando se atiende la interrupción, a través de la rutina de servicio, INTF se debe colocar en cero antes de regresar al programa principal. La interrupción puede reactivar al microcontrolador después de la instrucción SLEEP, si previamente el bit INTE fue habilitado. El estado del bit GIE decide si el procesador salta al vector de interrupción después al salir del SLEEP.

- **Interrupción por finalización de la temporización.**

La superación del conteo máximo (FFh□00h) en el TMRO colocará el bit TOIF en uno (INTCON<2>). El bit de control respectivo es TOIE (INTCON<5>).

- **Interrupción por cambio en el puerto RB.**

Un cambio en los pines del puerto B <7:4> colocará en uno el bit RBIF (INTCON<0>). El bit de control respectivo es RBIE (INTCON<3>). Para que el cambio sobre el pin de entrada/salida sea reconocido, el ancho del pulso debe ser por lo menos el ancho del tiempo de ciclo de instrucción TCY.

- **Interrupción por conversión A/D.**

El convertidor A/D coloca en uno el bit ADIF (INTCON<1>) al finalizar una conversión. El bit de control respectivo es ADIE (INTCON<6>).

Bits de configuración

El PIC16C71 posee un byte de configuración. Estos bits se pueden programar para seleccionar varias configuraciones del dispositivo: tipo de oscilador, protección de código, habilitación del circuito de vigilancia y el temporizador al encendido. Los bits se localizan en la posición de memoria 2007h, posición a la cual el usuario sólo tiene acceso durante la programación del microcontrolador.

Cuando se programa la protección de código, el contenido de cada posición de la memoria no se puede leer completamente, de tal manera que el código del programa no se puede reconstruir. Adicionalmente, todas las posiciones de memoria del programa se protegen contra la reprogramación.

Una vez protegido el código, el bit de protección sólo puede ser borrado (puesto en 1). si se borra toda la memoria del programa y la de datos.

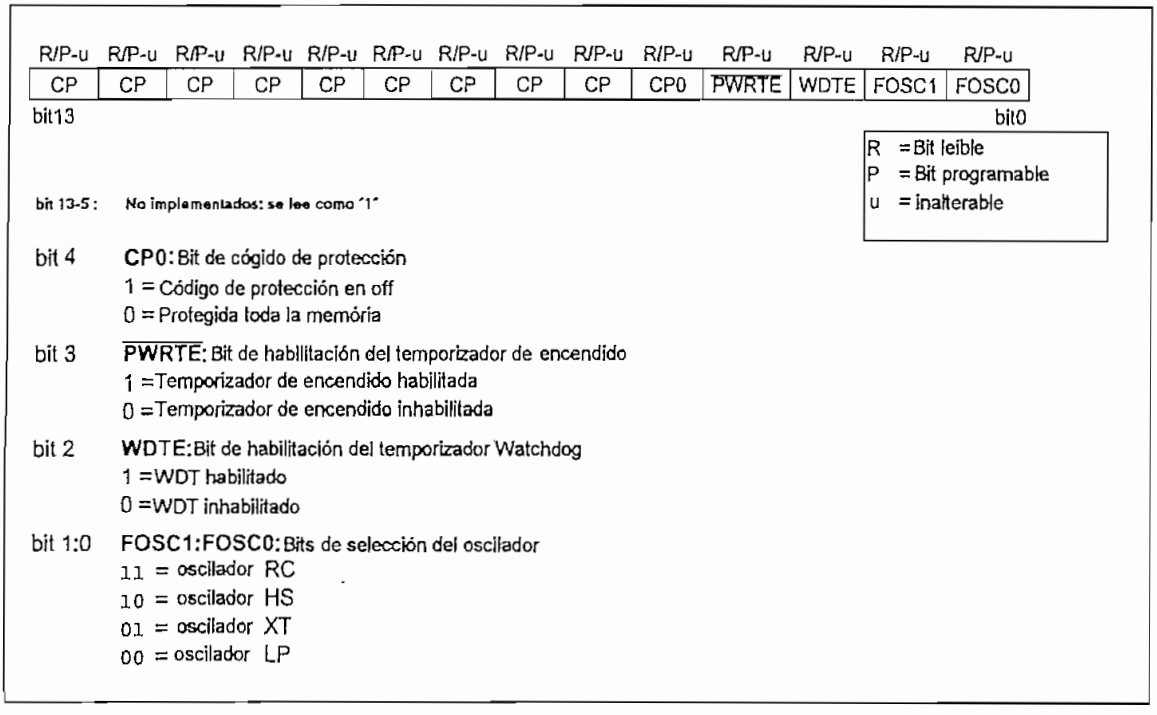


Figura 1.57 Bits de configuración

Las pull-ups internas

Cada uno de los pines del puerto B tiene un débil elemento pull-up interno (250 μA típico); este elemento es automáticamente desconectado cuando el pin se configura como salida. Adicionalmente, el bit RBPU (OPTION<7>) controla todos estos elementos, los cuales están inhabilitados ante una condición de reset. Estos elementos pull-up son especialmente útiles cuando el microcontrolador va a colocarse en el modo de bajo consumo, ya que ayudan a no tener las entradas flotantes, significando una reducción en el consumo de corriente.

Localidades de código de identificación ID

Cuatro localidades de memoria (2000h – 2003h) son diseñadas como localidades para códigos de identificación o chequeo de suma (*checksum*). Estas localidades no son accesibles durante ejecución normal pero pueden ser escritas y leídas durante programación y verificación. Solo los cuatro bits menos significativos son usados.

Programación Serial en circuito

El microcontrolador puede ser programado serialmente al estar montado sobre el circuito final. Esto se lo realiza fácilmente mediante dos líneas de reloj y de datos, y otras líneas para alimentación, tierra y voltaje de programación. Esto permite fabricar equipos que sean fácilmente actualizables con la ultima versión de firmware. El circuito típico de programación serial se muestra en la Figura 1.58.

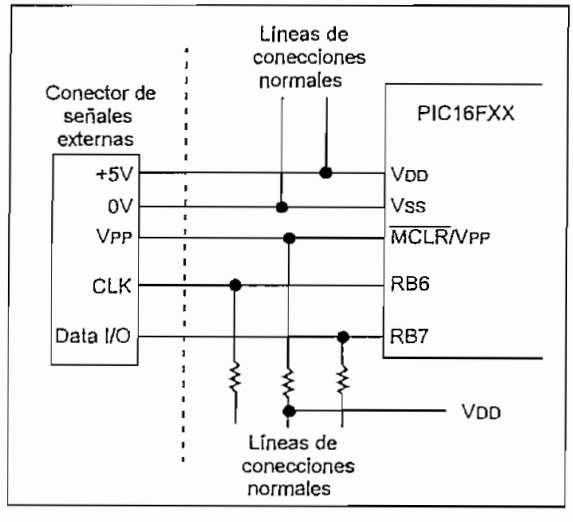


Figura. 1.58 Circuito típico de programación serial en circuito

1.3.12 El conjunto de instrucciones

Estas se clasifican en operaciones orientadas a registros, orientadas al bit y operaciones literales y de control. Cada instrucción es una palabra de 14 bits, dividida en un código de operación (el cual especifica la orden a ejecutar) y uno o más operandos sobre los que se actúa. En la Tabla 1.16 se muestra un resumen de todas las instrucciones la cuál está dividida en operaciones orientadas a bytes, a bits y a operaciones literales y de control.

Nemotécnico		Descripción	Ciclos	Código de operación MSb LSb	Estados Afectados
Operaciones orientadas a registros					
ADDWF	f, d	Sumar W y f	1	00 0111 dfff ffff	C,DC,Z
ANDWF	f, d	AND entre W y f	1	00 0101 dfff ffff	Z
CLRF	f	Borrar f	1	00 0001 lfff ffff	Z
CLRW	-	Borrar W	1	00 0001 0xxx xxxx	Z
COMF	f, d	Complementar f	1	00 1001 dfff ffff	Z
DECF	f, d	Decrementar f	1	00 0011 dfff ffff	Z
DECFSZ	f, d	Decrementar f, salta si es cero	1(2)	00 1011 dfff ffff	
INCF	f, d	Incrementar f	1	00 1010 dfff ffff	Z
INCFSZ	f, d	Incrementar f, salta si es cero	1(2)	00 1111 dfff ffff	
IORWF	f, d	OR entre W y f	1	00 0100 dfff ffff	Z
MOVF	f, d	Mover f	1	00 1000 dfff ffff	Z
MOVWF	f	Mover W a f	1	00 0000 lfff ffff	
NOP	-	No operación	1	00 0000 0xx0 0000	
RLF	f, d	Rotar a la izquierda a través del carry	1	00 1101 dfff ffff	C
RRF	f, d	Rotar a la derecha a través del carry	1	00 1100 dfff ffff	C
SUBWF	f, d	Restar W de f (destino= f - W)	1	00 0010 dfff ffff	C,DC,Z
SWAPF	f, d	Intercambiar f<7:4> ⟷ f<3:0>	1	00 1110 dfff ffff	
XORWF	f, d	OR exclusiva entre W y f	1	00 0110 dfff ffff	Z
Operaciones orientadas a Bits					
BCF	f, b	Borrar el bit b de f	1	01 00bb bfff ffff	
BSF	f, b	Activar el bit b de f	1	01 01bb bfff ffff	
BTFSC	f, b	Probar el bit b de f, saltar si es cero	1 (2)	01 10bb bfff ffff	
BTFSS	f, b	Probar el bit b de f, saltar si es uno	1 (2)	01 11bb bfff ffff	
Operaciones orientadas a literales y control					
ADDLW	k	Sumar el literal k a W	1	11 111x kkkk kkkk	C,DC,Z
ANDLW	k	AND entre k y W	1	11 1001 kkkk kkkk	Z
CALL	k	Llamar a subrutina k	2	10 0kkk kkkk kkkk	
CLRWDT	-	Borrar WDT	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$
GOTO	k	Saltar a la dirección k	2	10 1kkk kkkk kkkk	
IORLW	k	OR entre k y W	1	11 1000 kkkk kkkk	Z
MOVLW	k	Cargar a W con el literal k	1	11 00xx kkkk kkkk	
RETFIE	-	Retornar de la interrupción	2	00 0000 0000 1001	
RETLW	k	Retornar de la interrupción y cargar W con K	2	11 01xx kkkk kkkk	
RETURN	-	Retornar de subrutina	2	00 0000 0000 1000	
SLEEP	-	Ir al modo de bajo consumo	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$
SUBLW	k	Restar W a k (W = k - W)	1	11 110x kkkk kkkk	C,DC,Z
XORLW	k	OR exclusiva entre k y W	1	11 1010 kkkk kkkk	Z

Nota: Si d = 0 el resultado se almacena en W
Si d = 1 el resultado se almacena en el registro

Tabla 1.16 Resumen de instrucciones

En los Anexos se encuentra la lista completa de instrucciones, la cual incluye ejemplos y explicaciones. Como se puede observar allí, en total son 35, las cuales tardan un ciclo de máquina, a excepción de los saltos, que toman dos ciclos.

En las instrucciones orientadas a registros, ´f´ representa un registro del archivo de registros, y ´d´ el destino en donde se guardará el resultado. Si d es cero, el resultado es colocado en el registro W. Si d es uno, el resultado es colocado en el registro indicado por la instrucción.

En las instrucciones orientadas a bits, ´b´ representa el número de bit ha ser afectado, mientras que ´f´ representa el registro del archivo de registros en el cual él está ubicado el bit.

En las instrucciones orientadas a literales y control, ´k´ representa una constante o valor literal de ocho u once bits.

Todas las instrucciones son ejecutadas con un solo ciclo de instrucción, excepto las instrucciones de prueba con resultado verdadero o cuando el contador de programa es cambiado como resultado de la instrucción. En este caso, la instrucción toma dos ciclos de instrucción, en donde la segunda instrucción es ejecutada como un NOP. Una instrucción consiste en cuatro períodos de oscilación.

Para un oscilador de 4 Mhz, el tiempo de ejecución de una instrucción normal es de 1usg, mientras que para instrucciones de prueba con resultado verdadero o cuando el contador de programa es cambiado, este tiempo aumenta a 2usg.

En la Figura 1.59 se muestra los formatos generales que pueden tener los distintos tipos de instrucciones:

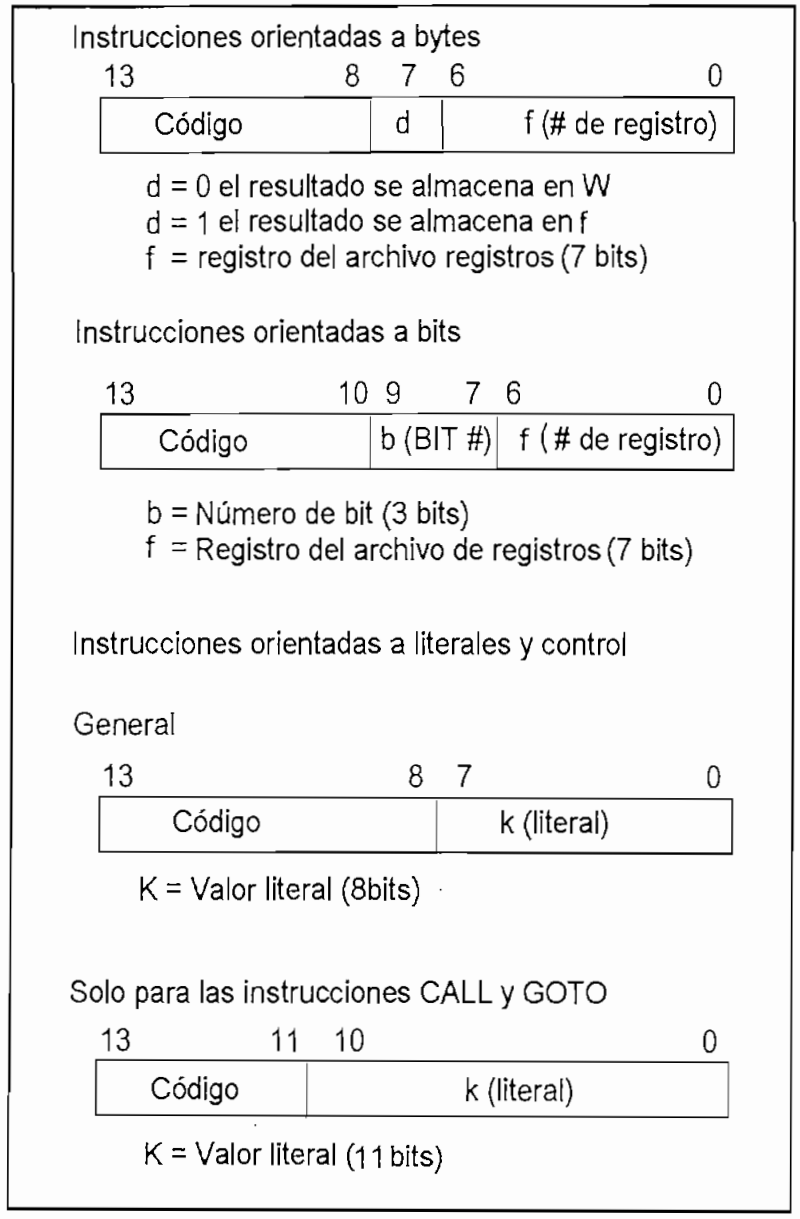


Figura 1.59 Formato general de instrucciones

1.4 El PIC12C50X

1.4.1 Características generales

Estos microcontroladores son ideales para aplicaciones en donde se requiera minimizar el tamaño utilizado por componentes electrónicos permitiendo mayor funcionalidad solucionando así problemas de volumen de dispositivos basados en microcontroladores. Permite aplicarlos en un sinnúmero de casos en donde antes no se consideraba su control como por ejemplo: codificadores/decodificadores en sistemas de control remoto, sistemas de seguridad, receptor de frecuencia, timer, funciones de coprocesador, etc. El PIC12C5O8 y el PICI2C509 son microcontroladores de 8 bits basados en la familia PICI6C5X, su precio y su rendimiento lo ubican como uno de los microcontroladores con mejor relación costo/beneficio.

Una de las mayores ventajas de esta familia es la eliminación de componentes externos ya que posee internamente su circuito de Reset al encender el sistema (PowerOn-Reset), circuito de vigilancia (Watchdog Timer) incorporado, cuatro posibles configuraciones de circuito de reloj, incluyendo un modo de oscilador interno (INTRC).

Principales características del PICI2C50X:

- 8 pines en total, de los cuales 5 se pueden usar como entrada/salida (I/O)
- Unicamente 33 instrucciones.
- Todas las instrucciones se ejecutan en un solo ciclo excepto las de saltos que utilizan dos ciclos.
- Instrucciones de 12 bits de ancho.
- Bus de datos de 8 bits de ancho
- Siete registros de función de hardware.
- Modos de Direcionamiento directo, indirecto y relativo para datos e instrucciones.
- Velocidad de operación de DC hasta 4 MHz. (DC – 1μs)
- Programación serial en circuito.
- Pila de 2 niveles.
- Oscilador interno de 4MHz con calibración programable.
- Temporizador / contador de 8 bits y preescalador programable.

- Pull-up interno en los pines de entrada/salida y en el pin de reset.
- Bajo consumo de potencia
- Reset al encendido (POR)
- Reset temporizado del equipo (DRT)
- Circuito de vigilancia (Watchdog), que incluye su propio oscilador RC interno.
- Código de protección programable.
- 1´000.000 de ciclos de borrado escritura sobre la memoria EEPROM de datos.
- Retención de datos de la memoria EEPROM mayor a 40 años.
- Modo de ahorro de energía SLEEP.
- Salida del modo de reposo por cambio en el pin.
- Opciones del oscilador seleccionables.
  - INTRC: Oscilador interno de 4 Mhz.
  - EXTRC: Oscilador externo RC de bajo costo.
  - XT: Cristal resonador estandar
  - LP: Cristal de baja frecuencia (Ahorro de energía).
- Tecnología CMOS de bajo consumo de potencia y alta velocidad.
- Diseño totalmente estático.
- Amplio rango de voltaje de operación.
- Amplio rango de operación de temperatura.
  - Comercial: 0°C a +70°C
  - Industrial: -40oC a +85oC
  - Extendido:-40oC a +125oC
- Bajo consumo de potencia
  - < 2mA @ 5V, 4Mhz
  - 15µA típicos @ 3V, 32Khz.
  - < 1µA típicos en modo de reposo

1.4.2 Arquitectura.

Los PIC12C50X están también basados en la arquitectura Harvard con sus ventajas ya explicadas para los otros PIC's.

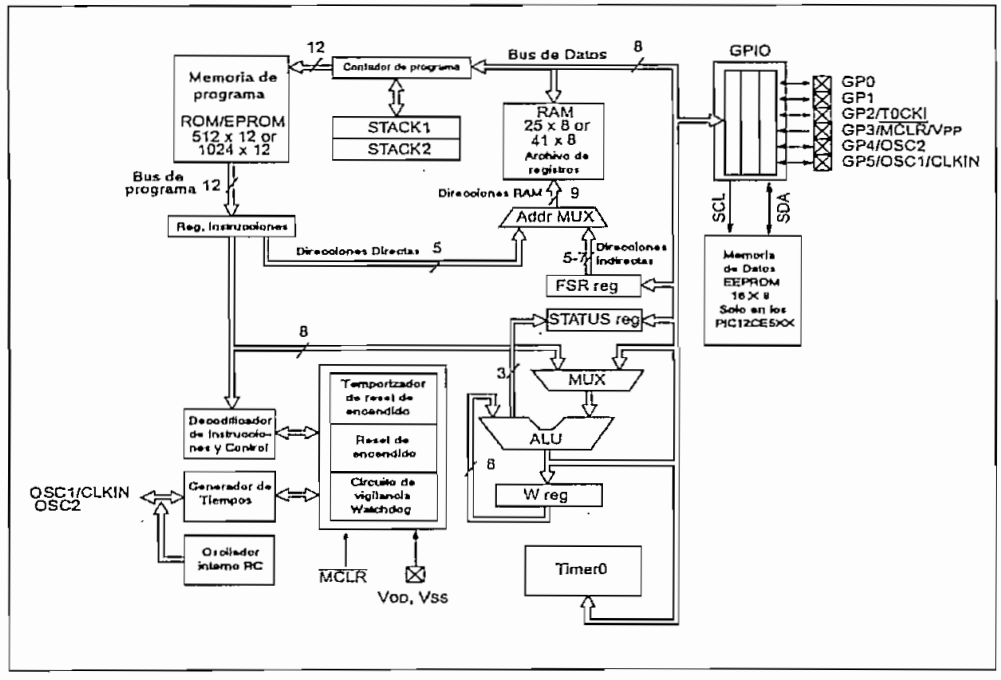


Figura. 1.60 Diagrama de bloques de los PIC12C5XX

La memoria de datos tiene una palabra de 8 bits. El bus y la memoria de programa tienen un ancho de 12 bits, lo que hace posible tener instrucciones poderosas que se ejecutan en un solo ciclo.

La Tabla 1.17 siguiente, lista la memoria de programa (EPROM), memoria de Datos (RAM), memoria ROM, y memoria no volátil (EEPROM) para cada versión.

Equipo:	Memory			
	Programa EPROM	Programa ROM	Datos RAM	Datos EEPROM
PIC12C508	512 x 12		25	
PIC12C509	1024 x 12		41	
PIC12C508A	512 x 12		25	
PIC12C509A	1024 x 12		41	
PIC12CR509A		1024 x 12	41	
PIC12CE518	512 x 12		25 x 8	16 x 8
PIC12CE519	1024 x 12		41 x 8	16 x 8

Tabla 1.17 Memorias de los PIC12C5XX

1.4.3 Descripción de los pines

La Figura 1.61 muestra los pines y su descripción para los microcontroladores PIC12C50X:

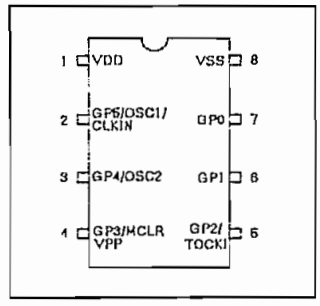


Figura. 1.61 Diagrama de pines del PIC12C50X

GP0: Pin bidireccional de entrada/salida / Datos para programación serial, puede ser programado por software para conectar el pull-up interno, y salida del modo SLEEP por cambio sobre el pín. Este buffer es una entrada tipo Schmitt Trigger cuando es utilizado en modo de programación serial.

GP1: Pin bidireccional de entrada/salida / Señal para programación serial, puede ser programado por software para conectar el pull-up interno, y salida del modo SLEEP por cambio sobre el pín. Este buffer es una entrada tipo Schmitt Trigger cuando es utilizado en modo de programación serial.

GP2/TOCKI: Pin bidireccional de entrada/salida; puede ser programado por software para utilizarlo como TOCKI en el conteo de eventos externos.

GP3/MCLR /Vpp: Pin de entrada solamente. Puede ser configurado por software como MCLR y es activado por un nivel bajo. Cuando se configura en este modo, siempre se activa su pull-up-interno. En el momento de programar el micro, este pin se conecta al voltaje de programación (Vpp).

GP4/OSC2: Pin bidireccional de entrada/salida. En caso detener oscilador con cristal o resonador solo se puede utilizar para tal fin. Si se utiliza el oscilador interno, se puede usar como puerto (I/O).

GP5/OSC1/CLKIN: Pin bidireccional de entrada/salida. Cuando se utiliza un oscilador a cristal o resonador solo se puede utilizar para tal fin. Si se utiliza el oscilador interno, se puede usar como puerto (I/O).

1.4.4 Organización de la memoria de programa

Los PIC12C5XX tienen un contador de programa (PC) de 12 bits con capacidad de direccionamiento de 2K por 12.

Solamente los primeros 512 X 12 (0000h – 01FFh) están implementados para los PIC12C508, PIC12C508A y PIC12CE518; y hasta 1K X12 (0000h – 03FFh) para los PIC12C509, PIC12C509A, PIC12CR509A, y PIC12CE519 (Figura 1.62).

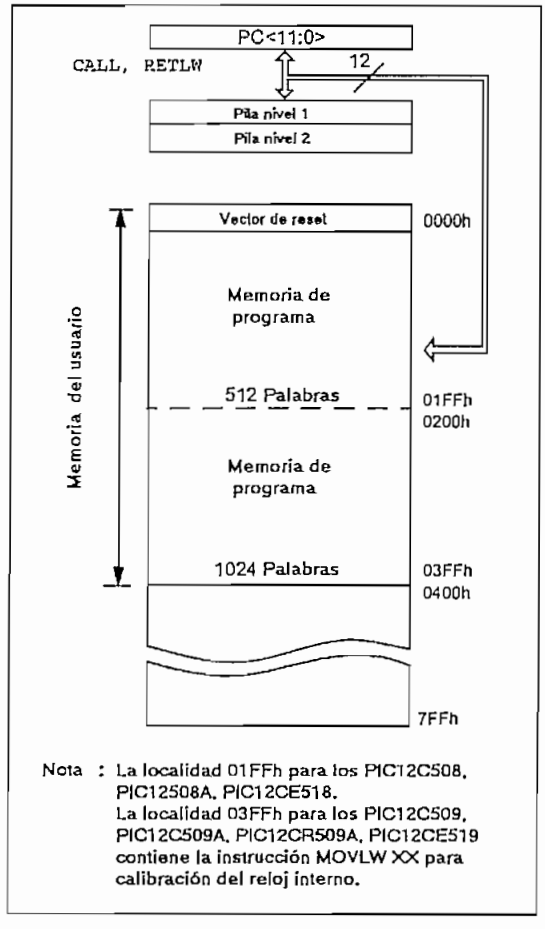


Figura. 1.62 Mapa de la memoria de programa

En el caso del PIC 12C509 se tiene un esquema de paginación de memoria; para accesarlo se debe usar un bit del registro de estado.

El vector de reset está en la dirección 0000h. La posición 01ffh en el P1C12C508 y la 03ffh en el P1C12C509 contiene el valor de calibración para el oscilador interno; este valor nunca podrá ser sobreescrito.

El vector de reset está en la dirección 0000h. La posición 01ffh en el PIC12C508 y la 03ffh en el PIC12C509 contiene el valor de calibración para el oscilador interno; este valor nunca podrá ser sobreescrito.

1.4.5 Organización de la memoria de datos

La memoria de datos está compuesta de registros o bytes de RAM. Estos se dividen en dos grupos: registros de función especial y registros de propósito general.

Para los PIC12C508, PIC12C508A Y PIC12CE518 se tienen 7 registros de función especial y 25 registro de propósito general como se muestra en la Figura 1.63 a continuación:

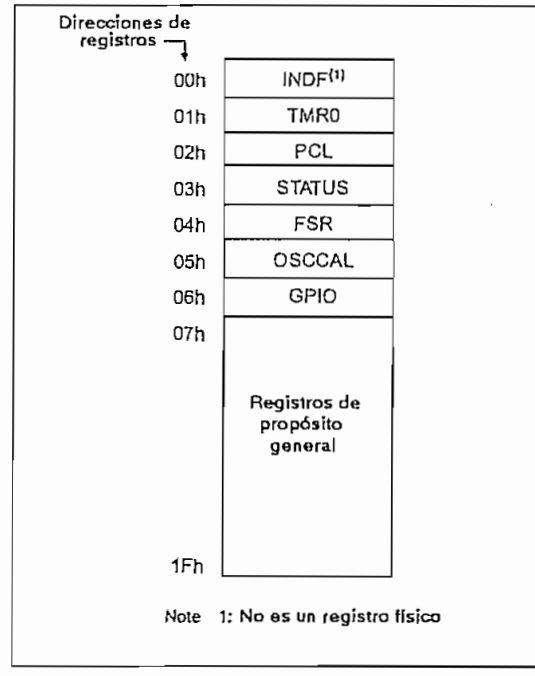


Figura. 1.63. Mapa de la memoria de datos para los PIC12C508, PIC12C508A Y PIC12CE518

Para los PIC12C509, PIC12C509A, PIC12CR509A y PIC12CE519 se tienen 7 registros de función especial, 25 registros de propósito general y 16 registros de propósito general que pueden ser direccionados usando el registro FSR, la Figura 1.64 mostrada a continuación:

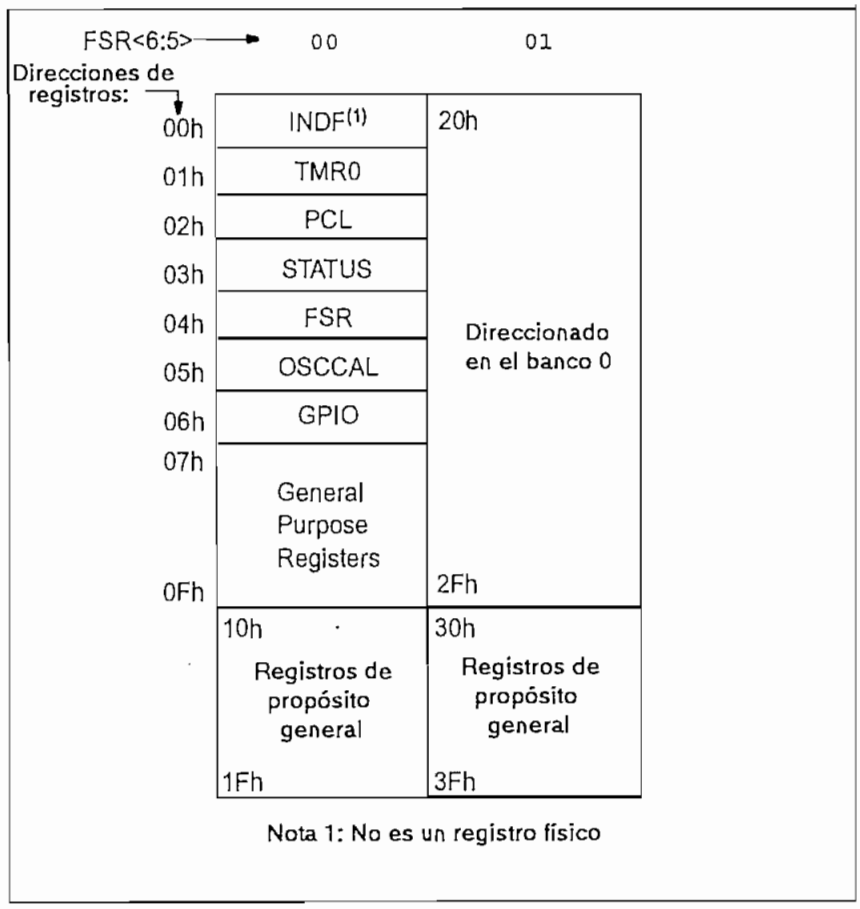


Figura. 1.64. Mapa de la memoria de datos para los PIC12C509, PIC12C509A, PIC12CR509A y PIC12CE519

Los registros de función especial están compuestos por el registro contador y reloj de tiempo real TMR0, el contador de programa (PC), el registro de estado, el puerto GP y el selector de registros FSR. Los registros de propósito general se utilizan para datos e información de control.

1.4.6 Registros de función especial

Los registros de función especial (SFRs) son registros usados por la CPU y funciones de periféricos para controlar la operación del microcontrolador. Son descritos a continuación:

INDF: Registro de direccionamiento indirecto. Este no es un registro que esté implementado físicamente; utiliza el contenido del FSR para direccionar la memoria de datos. Lo que se haga con el dato del registro escogido depende de la instrucción que se utilice.

TMRO: Contador y reloj de tiempo real de 8 bits Este registro se puede configurar por software para que trabaje como temporizador, cuya base de tiempo es el oscilador del micro. El contenido inicial del conteo se puede establecer ya que es un registro que se puede leer y escribir. También se puede configurar como un contador de eventos externos que ocurren en el pin TOCKI. El incremento del conteo se puede afectar con una preescala para lograr un período de tiempo más largo; dicha preescala se ajusta en el registro OPTION. Vale la pena aclarar que la preescala se puede utilizar tanto por el TMRO como por el Watchdog Timer, pero su asignación es mutuamente excluyente.

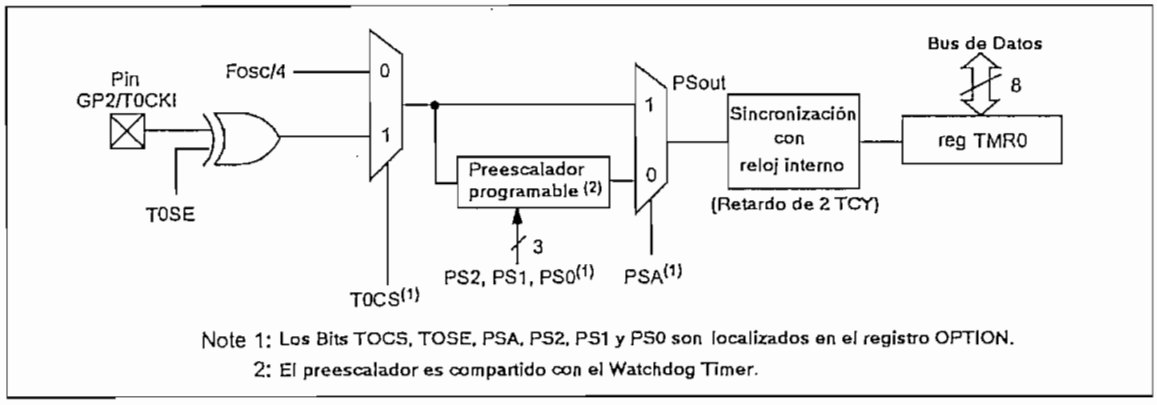


Fig. 1.65 Diagrama de bloques del TIMER0.

PC: Contador de programa. Es utilizado para señalar cual es la instrucción que se va a ejecutar. El PC se incrementa en un paso, a menos que la instrucción lo modifique. Cuando se ejecuta un llamado de subrutina (CALL), el bit 8 del PC se borra, lo que hace que todas las subrutinas deban ser escritas en las primeras 256 posiciones de cada página. Cuando se ejecuta una instrucción de salto (GOTO), el bit 9 del PC se carga con el contenido del bit PAO del registro de estado. Esto hace que si el destino está en otra página de memoria, se deba ajustar primero el bit PA0 del registro de estado con el valor correspondiente.

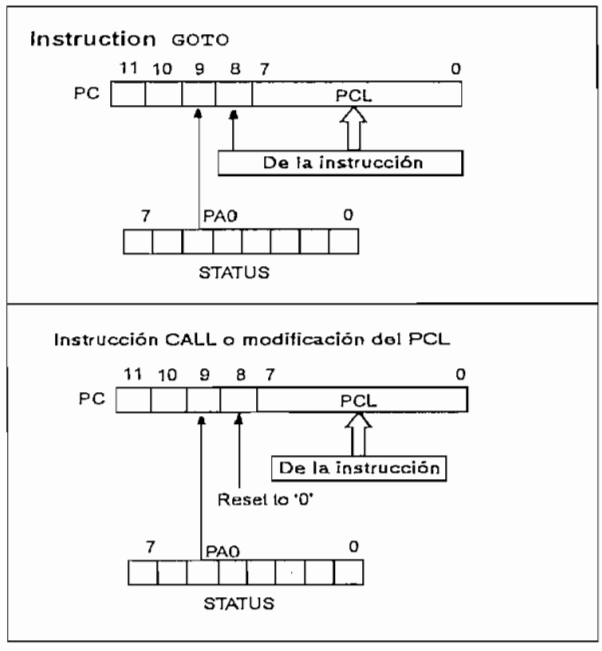
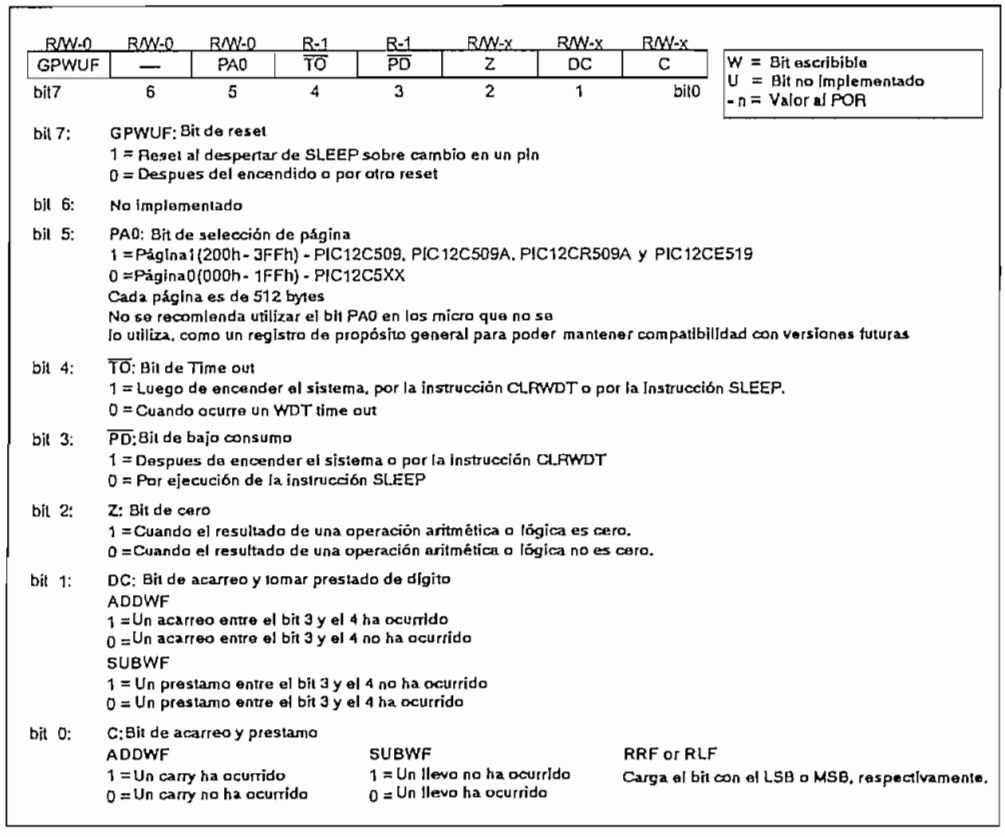


Figura. 1.66 Cargado del PC para instrucciones de saltos de los PIC12C5XX

STATUS: Registro de estado. Contiene el estado aritmético de la ALU, la causa del último Reset y el bit de selección de página de memoria (Figura 1.67).



R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
GPWUF	—	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit7	6	5	4	3	2	1	bit0

W = Bit escribible
U = Bit no implementado
- n = Valor al POR

bit 7: GPWUF: Bit de reset
1 = Reset al despertar de SLEEP sobre cambio en un pin
0 = Despues del encendido o por otro reset

bit 6: No implementado

bit 5: PA0: Bit de selección de página
1 = Página1 (200h - 3FFh) - PIC12C509, PIC12C509A, PIC12CR509A y PIC12CE519
0 = Página0 (000h - 1FFh) - PIC12C5XX
Cada página es de 512 bytes
No se recomienda utilizar el bit PA0 en los micro que no se
lo utiliza, como un registro de propósito general para poder mantener compatibilidad con versiones futuras

bit 4: $\overline{TO}$: Bit de Time out
1 = Luego de encender el sistema, por la instrucción CLRWDT o por la instrucción SLEEP.
0 = Cuando ocurre un WDT time out

bit 3: $\overline{PD}$: Bit de bajo consumo
1 = Despues de encender el sistema o por la instrucción CLRWDT
0 = Por ejecución de la instrucción SLEEP

bit 2: Z: Bit de cero
1 = Cuando el resultado de una operación aritmética o lógica es cero.
0 = Cuando el resultado de una operación aritmética o lógica no es cero.

bit 1: DC: Bit de acarreo y tomar prestado de dígito
ADDWF
1 = Un acarreo entre el bit 3 y el 4 ha ocurrido
0 = Un acarreo entre el bit 3 y el 4 no ha ocurrido
SUBWF
1 = Un prestamo entre el bit 3 y el 4 no ha ocurrido
0 = Un prestamo entre el bit 3 y el 4 ha ocurrido

bit 0: C: Bit de acarreo y prestamo

ADDWF	SUBWF	RRF or RLF
1 = Un carry ha ocurrido	1 = Un llevo no ha ocurrido	Carga el bit con el LSB o MSB, respectivamente.
0 = Un carry no ha ocurrido	0 = Un llevo ha ocurrido	

Figura. 1.67 Registro de estados

FSR: Registro selector de registros. Se utiliza con el registro INDF para apuntar a los otros registros disponibles en el modo de direccionamiento indirecto.

OSCCAL: Registro de calibración del oscilador interno. El PIC12C50X posee un oscilador RC interno de 4 MHz. Una instrucción de calibración de dicho oscilador se debe programar en el tope de la memoria. Este valor, OSCCAL, es programado con una instrucción "MOVLW XX", donde XX es el valor de la calibración, esta instrucción debe ser colocada en última disposición de memoria. Incrementando el valor de calibración se aumenta la frecuencia

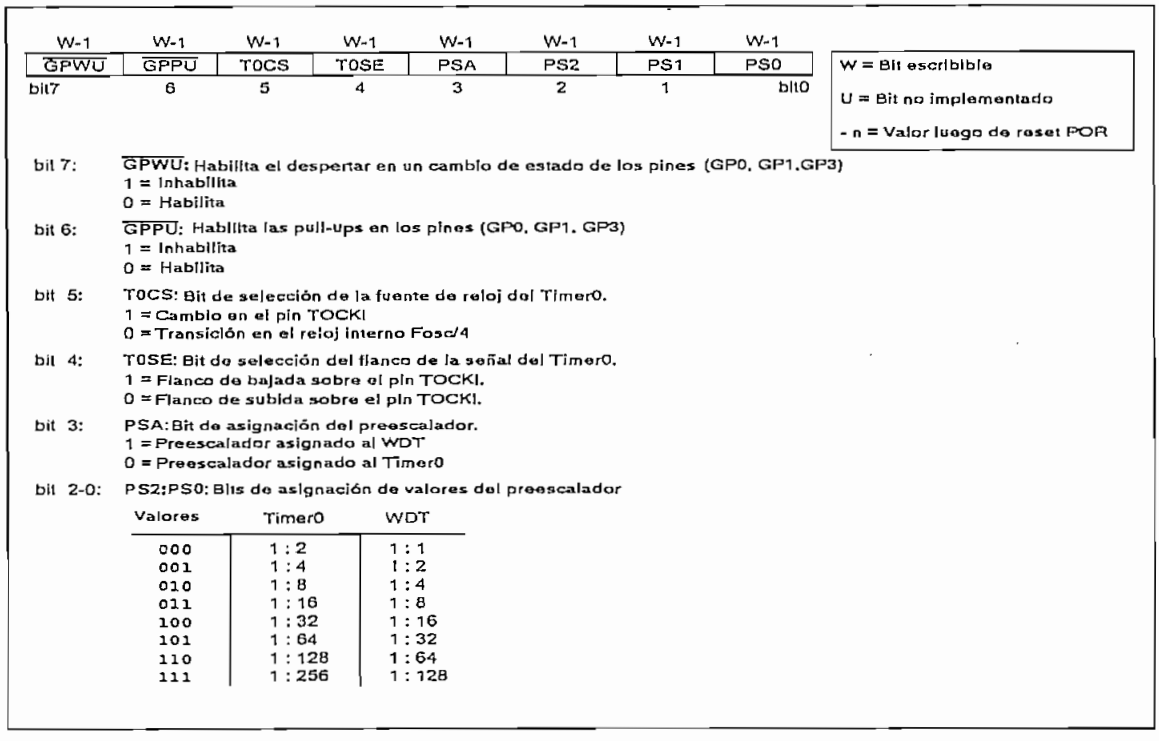
GPIO: Puerto de entrada/salida. Como cualquier otro registro, el GPIO puede ser leído y escrito bajo control del programa. Este es un registro de 8 bits, de los cuales solamente los 6 más bajos están implementados (GP5:GP0), los bits 7 y 6 no están implementados y se lee como ceros, GP0 y GP1 pueden ser configurados para conectar a las resistencias pull-up o como salida de modo de reposo por cambio en el pin, pero estas dos funciones no son seleccionables pin a pin. Si el pin 4 (GP3) es configurado como MCLR, las pull-up están siempre encendidas y la función de salida del modo SLEEP por cambio inhabilitada.

TRIS: Registro de configuración del puerto. Este registro es cargado con el contenido
del registro W al ejecutar la instrucción 'TRIS f''. Un "1"en un bit del registro TRIS configura el bit correspondiente de GPIO como entrada poniéndolo en modo de alta impedancia, un "0" lo configura como salida; a excepción del pin GP3 que solo se puede configurar como entrada y GP2 que puede ser controlado por el registro OPTION Este registro es solo de escritura y luego de una condición de Reset, se coloca todo en unos.

OPTION: Registro de configuración múltiple. Es un registro de 8 bits de sólo escritura (Figura 1.68), el cual contiene varios bits de control para configurar el TMRO y la preescala del Timer0/WDT. La ejecución de una instrucción "OPTION" mueve el contenido de W al registro OPTION. Cuando sucede un Reset todos los bits del registro son puestos a uno.

Si el bit TRIS es puesto en cero, las funciones de despertar del modo SLEEP y pull-up son inhabilitadas para los pines, notando que el registro TRIS tiene prioridad sobre los bits GPPU y GPWU del registro OPTION.

Si el bit TOCS es seteado en uno, GP2 está forzado a ser entrada igual si TRIS GP2='0'.



W-1	W-1	W-1	W-1	W-1	W-1	W-1	W-1
$\overline{GPWU}$	$\overline{GPPU}$	T0CS	T0SE	PSA	PS2	PS1	PS0
bit7	6	5	4	3	2	1	bit0

W = Bit escribible
U = Bit no implementado
- n = Valor luego de reset POR

bit 7: $\overline{GPWU}$: Habilita el despertar en un cambio de estado de los pines (GP0, GP1,GP3)
1 = Inhabilita
0 = Habilita

bit 6: $\overline{GPPU}$: Habilita las pull-ups en los pines (GP0, GP1, GP3)
1 = Inhabilita
0 = Habilita

bit 5: T0CS: Bit de selección de la fuente de reloj del Timer0.
1 = Cambio en el pin TOCKI
0 = Transición en el reloj interno Fosc/4

bit 4: T0SE: Bit de selección del flanco de la señal del Timer0.
1 = Flanco de bajada sobre el pin TOCKI.
0 = Flanco de subida sobre el pin TOCKI.

bit 3: PSA: Bit de asignación del preescalador.
1 = Preescalador asignado al WDT
0 = Preescalador asignado al Timer0

bit 2-0: PS2:PS0: Bits de asignación de valores del preescalador

Valores	Timer0	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

Figura. 1.68. Registro OPTION

1.4.7 Características Especiales

Palabra de configuración para el PIC12C50X.

Este registro no es direccionable durante la operación normal del microcontrolador , sólo se puede utilizar en el momento de programarlo. Tiene un ancho de 12 bits, dos de estos bits son para la selección del tipo de oscilador, uno para habilitar el watch dog timer y uno para habilitar el MCLR (Figura 1.69).

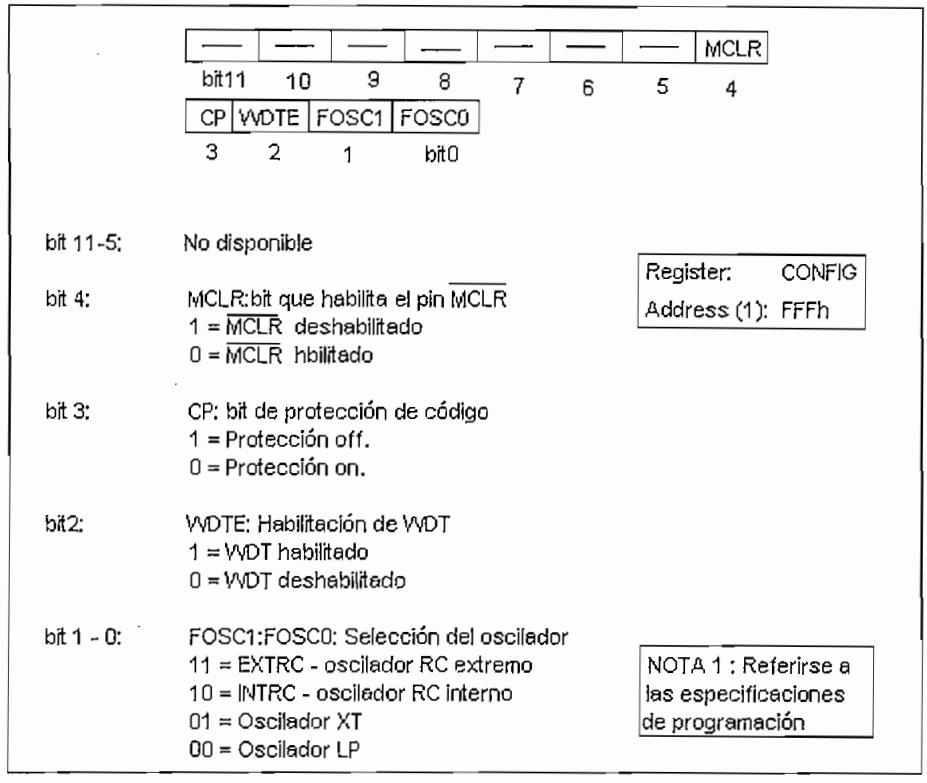


Figura. 1.69. Palabra de configuración del PIC12C50X

Pila o Stack

El PIC12C50X tiene una pila de dos niveles y 12 bits de ancho. Una instrucción "CALL" guarda el valor de stack1 en el stack2 y el valor actual del contador de programa incrementado en uno en el stack1. Si se ejecutan más de dos "CALL", solamente los dos últimos son almacenados.

Una instrucción "RETLW" retorna el valor de stack1 al contador de programa y el valor de stack2 al stack1. Una característica importante es que el registro W se carga con el literal especificado en la instrucción; esto es de gran utilidad cuando se trabaja con tablas usando la memoria de programa.

Watchdog Timer

Este es un oscilador RC interno del micro que corre independientemente del reloj de 4 MHz y no requiere componentes externos. El WDT tiene un período nominal de 18 milisegundos, luego de los cuales el micro sufre un Reset. Para evitar esto, se debe utilizar la instrucción CLRWDT antes de que se complete el tiempo de este oscilador interno. El bit TO (STATUS<4>) será borrado por un reset del WDT.

El tiempo del WDT de 18 milisegundos se puede extender hasta 2 segundos haciendo uso de la preescala, como se vió cuando hablamos del registroOPTION. Esto es, el período del oscilador interno del WDT es dividido por el valor que se le dé a la preescala . Debe tenerse en cuenta que la preescala se puede utilizar con el WDT o con el temporizador TMRO, pero solo con uno a la vez. El WDT se puede habilitar o inhabilitar en el momento de programar el microcontrolador utilizando el fusible correspondiente de la palabra de configuración.

Reset El PIC12C50X tiene varias fuentes de reset:

• La Temporización de reset (DTR) es un dispositivo interno del micro que garantiza un tiempo de reset que depende del oscilador seleccionado según la Tabla 1.18 siguiente:

Configuración del oscilador	Reset POR	Reset Subsecuente
IntRC & ExtRC	18 ms (Típicos)	300 µs (Típicos)
XT & LP	18 ms (Típicos)	18 ms (Típicos)

Tabla 1.18 DRT Períodos de temporización de reset

Esto es muy útil porque evita posibles conflictos que se podrían presentar antes de que se estabilicen las señales. Además, evita tener que utilizar componentes externos en el pin de reset.

- Un nivel bajo en el pin de Master Clear (MCLR) durante operación normal, este pin posee un pull-up interno que evita el uso de elementos externos.
- Un reset causado por el Watchdog Timer, el cual se presenta sino se ha borrado el contador interno antes de completar su período de tiempo.
- Cuando el micro está en modo de bajo consumo por la instrucción SLEEP, el cambio de nivel en un pin programado previamente hace que el micro despierte y sufra un reset.

Configuraciones del oscilador

El PIC12C50X puede ser operado en cuatro modos diferentes de oscilador. El usuario puede escoger a través de dos bits (FOSC1: FOSC0) para seleccionar uno de los modos siguientes:

LP: Cristal de baja potencia

XT: Cristal o resonador cerámico, máximo de 4MHz,

EXTRC: Oscilador externo RC.

INTRC: Oscilador interno de 4MHz. Este no requiere ningún componente externo y nos permite utilizar los dos pines que quedan disponibles como entrada/salida. Esta es una de las ventajas más importantes de esta familia.

Para utilizar el reloj interno se deben tener en cuenta varios aspectos:

1. El oscilador interno provee una frecuencia fija de 4Mh, el oscilador requiere un valor de calibración, dicho número se escribe en la última posición de la memoria de programa, en la forma de una instrucción: MOVLW XX, donde XX corresponde a dicho valor. Si se trata de un microcontrolador en versión OTP *(One Time Programiable:* programable una sola vez), dicha instrucción con el valor de calibración preestablecido en la fábrica ya está escrito, por lo tanto se puede obviar su escritura.

Si se trata de un microcontrolador en versión JW (ventana), dicho valor viene grabado de la misma forma en la última posición: por lo tanto el usuario debe leer el microcontrolador y tomar nota de dicho valor. Esto es necesario ya que en el momento de borrar el microcontrolador con luz ultravioleta para volver a cargar otro programa se borra también el valor de calibración, Por consiguiente, el usuario debe incluir en su programa la instrucción que cargue el valor de calibración en la última posición de memoria.

Por ejemplo si el valor de calibración es 30h, entonces se debe escribir las siguientes líneas de programa:

```
ORG     1FFh    ;escribe la siguiente instrucción en la dirección 1FFh
MOVLW   30h     ;carga el valor de calibración en el registro W
```

Se debe tomar en cuenta que no todos los PIC12C5O8 tienen el mismo valor de calibración.

2. Para cargar el valor de calibración en el registro OSCCAL se debe escribir la siguiente línea al comienzo de todos los programas que se escriban para el PIC12C508:

```
ORG     00h     ;escribe la siguiente instrucción en la dirección 00h
MOVWF   OSCCAL  ;carga el valor de W en el registro OSCCAL
```

Todos estos pasos se deben a la ruta que sigue el contador de programa o PC en el momento en que el microcontrolador sufre un reset , el cual, primero apunta a la última posición de memoria (en este caso 1FFh) y luego a la primera (00h).

Conjunto de Instrucciones

El conjunto de instrucciones es el mismo que se tiene para la familia PIC16C5X,. Debe notarse que solamente se tienen 33 instrucciones, a diferencia del PIC16F84 y del PIC16C71 que tienen 35.

Todas las instrucciones toman un ciclo simple para ser ejecutadas, excepto las de condiciones de prueba cuyo resultado es verdadero que toman dos ciclos de instrucción.

Para el reloj de 4Mhz el tiempo de un ciclo de instrucción es de 1µs y para dos ciclos sería 2µs.

Los formatos que las instrucciones pueden tener son los tres indicados en la Figura 1.70 siguiente:

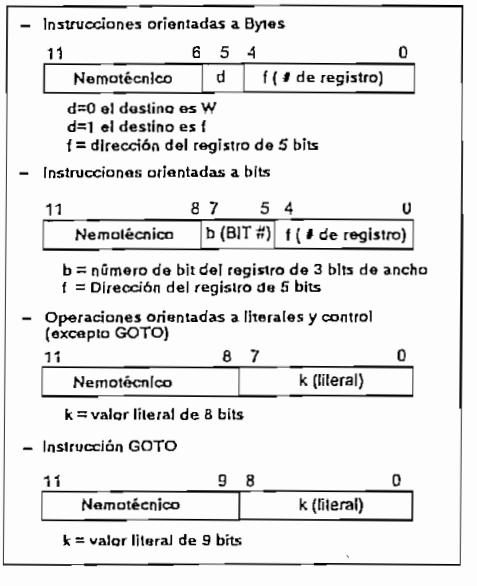


Figura. 1.70 Formatos generales para las instrucciones

A continuación se pueden observar un resumen del conjunto de instrucciones en la Tabla 1.19:

Nemotécnicos		Descripción	Ciclos	Operandos		Estados Afectados
OPERACIONES ORIENTADAS A REGISTROS				MSb	LSb	
ADDWF	f,d	Sumar W y f	1	0001 11df ffff		C,DC,Z
ANDWF	f,d	AND entre W y f	1	0001 01df ffff		Z
CLRF	f	Borrar f	1	0000 011f ffff		Z
CLRW	–	Borrar W	1	0000 0100 0000		Z
COMF	f, d	Complementar f	1	0010 01df ffff		Z
DECF	f, d	Decrementar f	1	0000 11df ffff		Z
DECFSZ	f, d	Decrementar f, salta si es cero	1(2)	0010 11df ffff		Ninguno
INCF	f, d	Incrementar f	1	0010 10df ffff		Z
INCFSZ	f, d	Incrementar f, salta si es cero	1(2)	0011 11df ffff		Ninguno
IORWF	f, d	OR entre W y f	1	0001 00df ffff		Z
MOVF	f, d	Mover f	1	0010 00df ffff		Z
MOVWF	f	Mover W a f	1	0000 001f ffff		Ninguno
NOP	–	No operación	1	0000 0000 0000		Ninguno
RLF	f, d	Rotar a la izquierda a través del carry	1	0011 01df ffff		C
RRF	f, d	Rotar a la derecha a través del carry	1	0011 00df ffff		C
SUBWF	f, d	Restar W de f (destino= f - W)	1	0000 10df ffff		C,DC,Z
SWAPF	f, d	Intercambiar f<7:4> ⟷ f<3:0>	1	0011 10df ffff		Ninguno
XORWF	f, d	OR exclusiva entre W y f	1	0001 10df ffff		Z
OPERACIONES ORIENTADAS A BITS						
BCF	f, b	Borrar el bit b de f	1	0100 bbbf ffff		Ninguno
BSF	f, b	Activar el bit b de f	1	0101 bbbf ffff		Ninguno
BTFSC	f, b	Probar el bit b de f, saltar si es cero	1 (2)	0110 bbbf ffff		Ninguno
BTFSS	f, b	Probar el bit b de f, saltar si es uno	1 (2)	0111 bbbf ffff		Ninguno
OPERACIONES ORIENTADAS A LITERALES Y CONTROL						
ANDLW	k	AND entre k y W	1	1110 kkkk kkkk		Z
CALL	k	Llamar a subrutina k	2	1001 kkkk kkkk		Ninguno
CLRWDT	k	Borrar WDT	1	0000 0000 0100		$\overline{TO}$, $\overline{PD}$
GOTO	k	Saltar a la dirección k	2	101k kkkk kkkk		Ninguno
IORLW	k	OR entre k y W	1	1101 kkkk kkkk		Z
MOVLW	k	Cargar a W con el literal k	1	1100 kkkk kkkk		Ninguno
OPTION	–	Carga el registro OPTION	1	0000 0000 0010		Ninguno
RETLW	k	Retorna de la interrupción y caraga a W con K	2	1000 kkkk kkkk		Ninguno
SLEEP	–	Ir a modo de bajo consumo	1	0000 0000 0011		$\overline{TO}$, $\overline{PD}$
TRIS	f	Carga el registro TRIS	1	0000 0000 0fff		Ninguno
XORLW	k	OR exclusiva entre k y W	1	1111 kkkk kkkk		Z

Tabla 1.19 Resumen de Instrucciones del PIC12C50X

Se ha dado un vistazo detallado de los PICs más representativos y ahora en el capítulo siguiente se va ha realizar una explicación del manejo del programa MPLAB de edición y simulación de programas para estos microcontroladores.

CAPITULO II

SOFTWARE DE DESARROLLO "MPLAB" PARA MICROCONTROLADORES PIC

2.1 INTRODUCCION

El programa MPLAB de microchip es un entorno integrado de desarrollo que permite crear y simular archivos fuente, para los microcontroladores PIC. El Mplab incluye un editor de texto, funciones para crear un proyecto, un programa ensamblador, un simulador interno y una amplia variedad de herramientas que ayudan a depurar los programas en desarrollo. MPLAB puede ser ejecutado bajo Windows 3.1x, 95, 98, NT, y 2000. A continuación se detallarán todas las tareas necesarias para comprender el manejo del paquete MPLAB.

2.2 INSTALACIÓN

El programa puede ser descargado via internert de la página web de microchip cuya dirección es http://www.microchip.com, dentro de la opción de software libre. Estos archivos deben ser desempaquetados y se inicia la instalación ejecutando el archivo Mpl530.exe, luego aparece el menú de la Figura 2.1 que permite seleccionar los componentes del programa Mplab que se desee instalar. Normalmente es aconsejable instalar todo como viene por defecto. El programa de instalación va indicando paso a paso todo lo que se debe realizar para completar la instalación.

Una vez terminada la instalación se puede ejecutar el programa desde la barra de inicio de windows dentro de los programas. Al ejecutarlo aparece la pantalla de la Figura 2.2

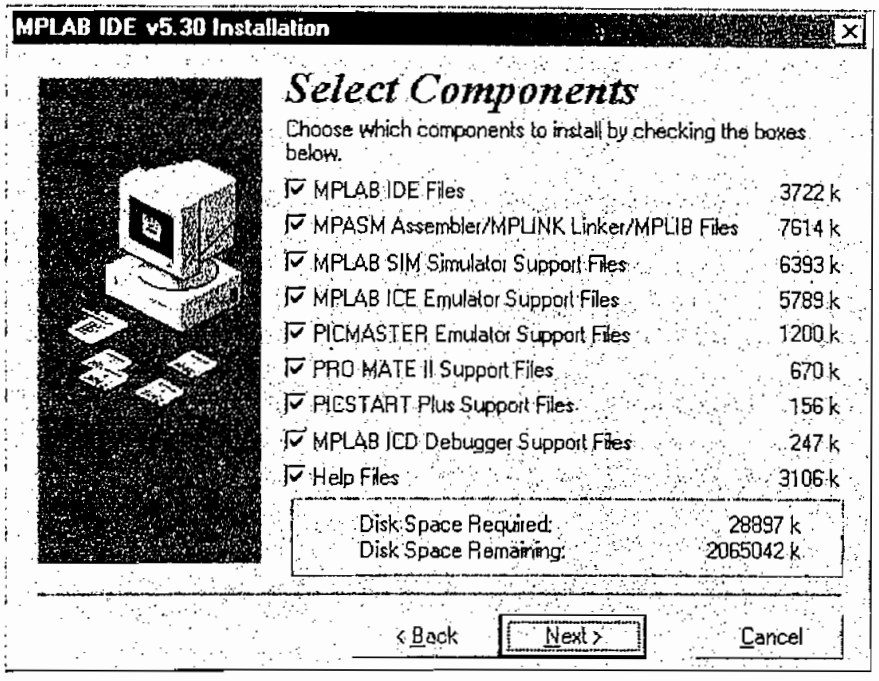


Figura 2.1 Menú de selección de componentes

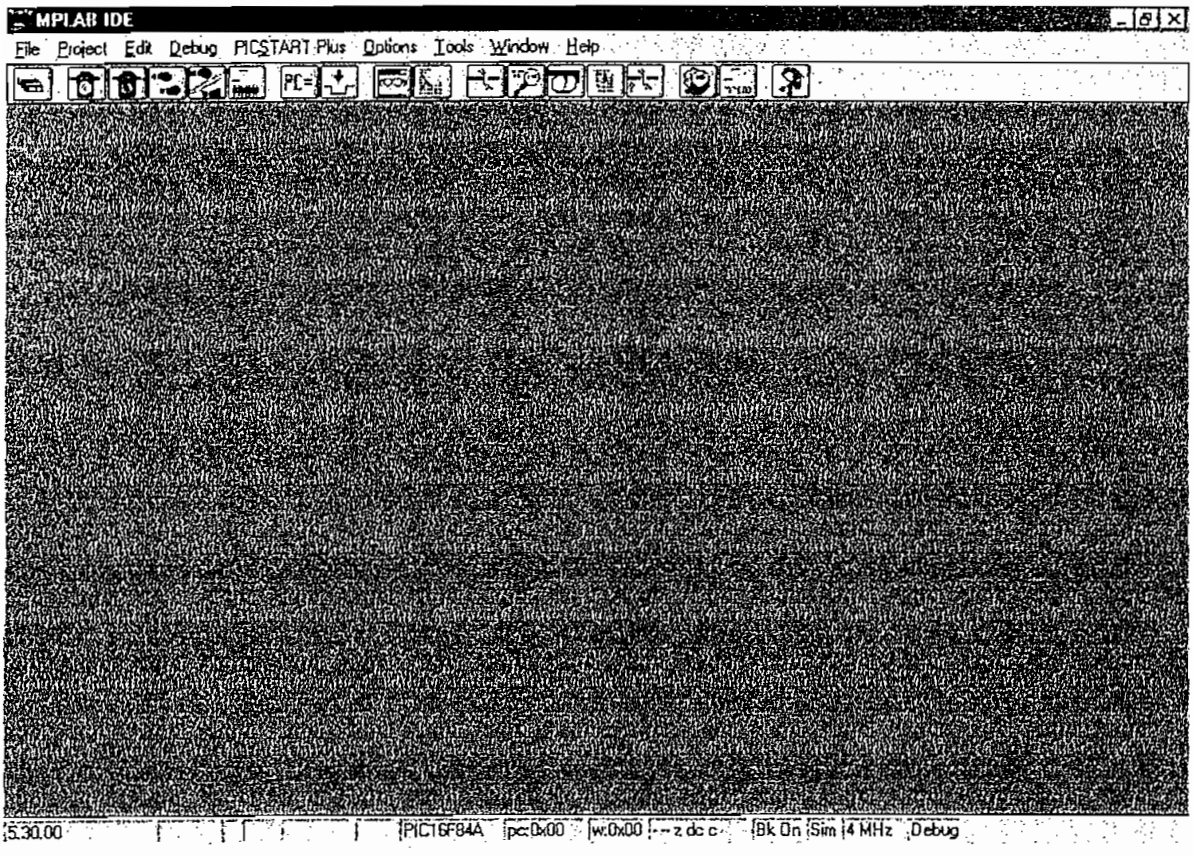


Figura 2.2 Escritorio del MPLAB.

2.3 CONFIGURACIÓN DEL MODO DE DESARROLLO

El escritorio de MPLAB se asemeja al de las aplicaciones de Windows en las que se tiene una barra de herramientas en el margen superior y una barra de estados en el margen inferior. En la barra inferior se puede observar información de la configuración del sistema; por ejemplo, se puede ver el tipo de microcontrolador utilizado y la frecuencia del oscilador.

Una vez ejecutado el programa, se debe configurar el modo de desarrollo, seleccionando en la barra superior el item "Opciónes>modo de desarrollo", entonces aparecerá la pantalla de la Figura 2.3.

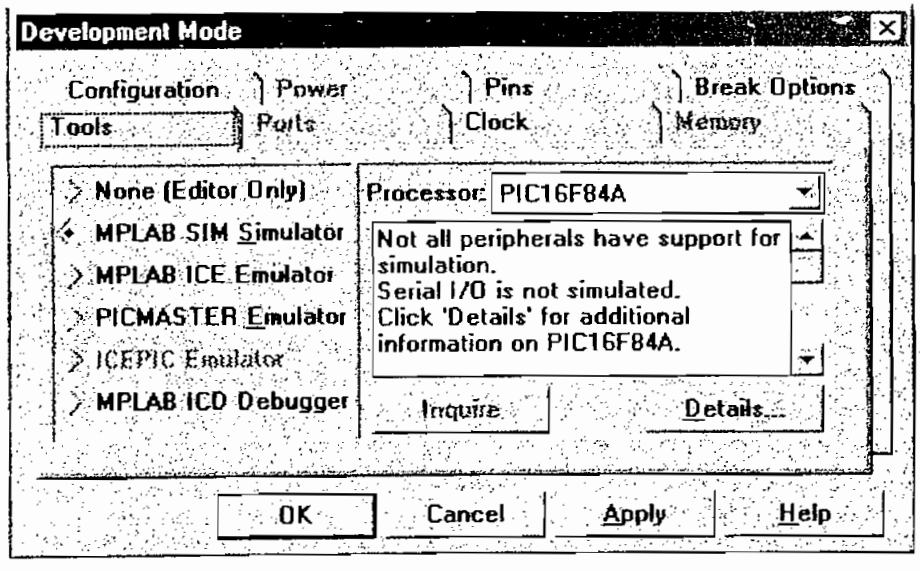


Figura 2.3 Menú de selección del modo de desarrollo

En esta pantalla se debe seleccionar el modo de simulación con un clic en la opción MPLAB SIM para activar el software respectivo. Además se debe escoger el tipo de procesador que se va ha utilizar, en este caso en la mayoría de prácticas se utilizará el PIC16F84A, también se debe indicar la frecuencia del oscilador que en el sistema de desarrollo es de 4Mhz.

2.4 CREACIÓN DE UN PROYECTO

Un proyecto crea en un archivo de extensión .pjt, por ejemplo, led.pjt a través del cual se controlan los siguientes archivos: led.asm, led.hex, led.err, led.lst, led.cod y led.bkx. El archivo de extensión .pjt contiene la información general del proyecto, como las configuraciones de tipo de procesador, de oscilador, del W.D.T., etc. El archivo de extensión .asm debe ser escrito por el usuario y contiene las líneas del programa del proyecto. El archivo de extensión .err es generado al momento de compilar los programas y contiene información de los errores de edición detectados sobre el programa escrito. El archivo de extensión .lst contiene el programa listado por líneas y con las direcciones de ubicación de la memoria de programa, además contiene la tabla de etiquetas y los mensajes de error o advertencia en las líneas correspondientes. Los otros archivos no contienen información útil para el usuario, sino mas bien para el funcionamiento del proyecto.

Antes de crear el proyecto se debe primero crear el archivo del extensión .asm con el programa a simular, para esto se debe seleccionar el ítem archivo>nuevo, de la barra superior del escritorio de MPLAB, entonces aparecerá la ventana de edición del programa sin título. Una vez escrito el programa este debe ser guardado dentro de un directorio, como ejemplo se ha creado el directorio llamado PRUEBA, que guardará todos los archivos para el proyecto. El archivo ya creado se puede observar en la Figura 2.4 a continuación.

c:\prueba\led.asm

```
;Este programa realiza el encendido y apagado secuencial de un led
;con un retardo de 1sg de un estado a otro

        list    p=16F84A        ;Microcontrolador utilizado

status  equ     03h
porta   equ     05h
trisa   equ     85h
lazo1   equ     0ch
lazo2   equ     0dh
lazo3   equ     0eh

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

;*******************Subrutina de retardo de 1sg*********************************

retardo movlw   D'10'   ;El registro lazo1 contiene el número de
        movwf   lazo1   ;veces que repite 100 milisegundos
repite1 movlw   D'100'  ;El registro lazo2 contiene el número de
        movwf   lazo2   ;veces que repite 1 milisegundo
repite2 movlw   D'110'  ;El registro lazo3 contiene el número de
        movwf   lazo3   ;veces que repite los 9 microsegundos
repite3 nop             ;de retardo generados
```

Figura 2.4 Archivo de extensión .asm

Una vez creado el archivo led.asm se procede a crear el proyecto led.pjt, para esto se debe seleccionar en la barra de tareas de MPLAB, el ítem proyecto>nuevo proyecto y aparecerá la ventana de dialogo de la figura 2.5.

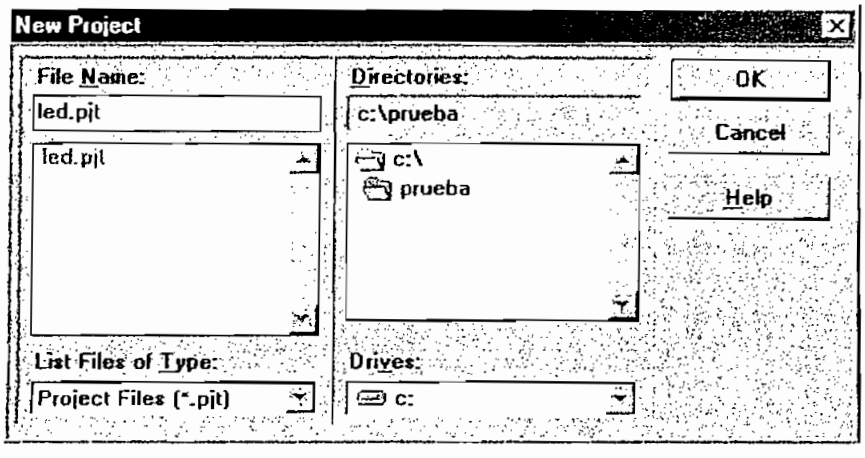


Figura 2.5 Cuadro de creación de un nuevo proyecto

En esta ventana se debe poner el nombre del proyecto, es preferible poner siempre el mismo nombre del archivo .asm, en este caso led.pjt. Hay otro tipo de proyectos que utilizan múltiples archivos con enlaces o "linkers" en los cuales es

posible que los nombres de los archivos sean diferente al del proyecto. Es importante notar que estos dos archivos deben estar dentro del mismo directorio.

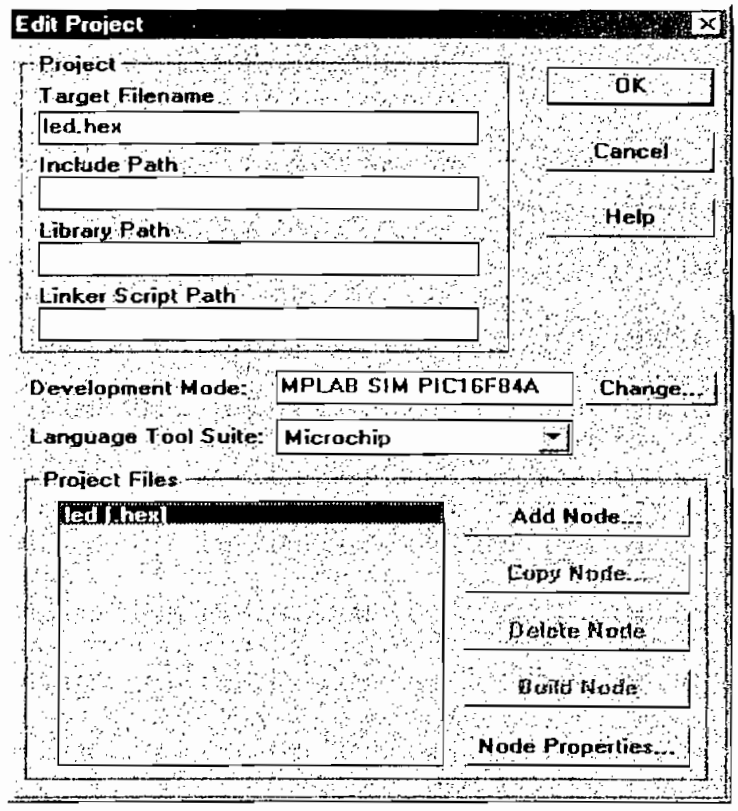


Figura 2.6 Cuadro de dialogo de edición del proyecto

Al presionar el botón OK de este cuadro aparecerá la ventana de la figura 2.6 en la que se puede observar las configuraciones del tipo de procesador realizadas previamente. En este cuadro se debe presionar el botón "adicionar nodo", mediante el cual a través de una ventana estándar de exploración de windows se selecciona el archivo led.asm creado previamente.

En esta ventana, dentro del cuadro de archivos de proyecto, se puede observar el archivo led.hex y al seleccionarlo se puede modificar las propiedades de nodo mediante el cuadro de dialogo de la Figura 2.7

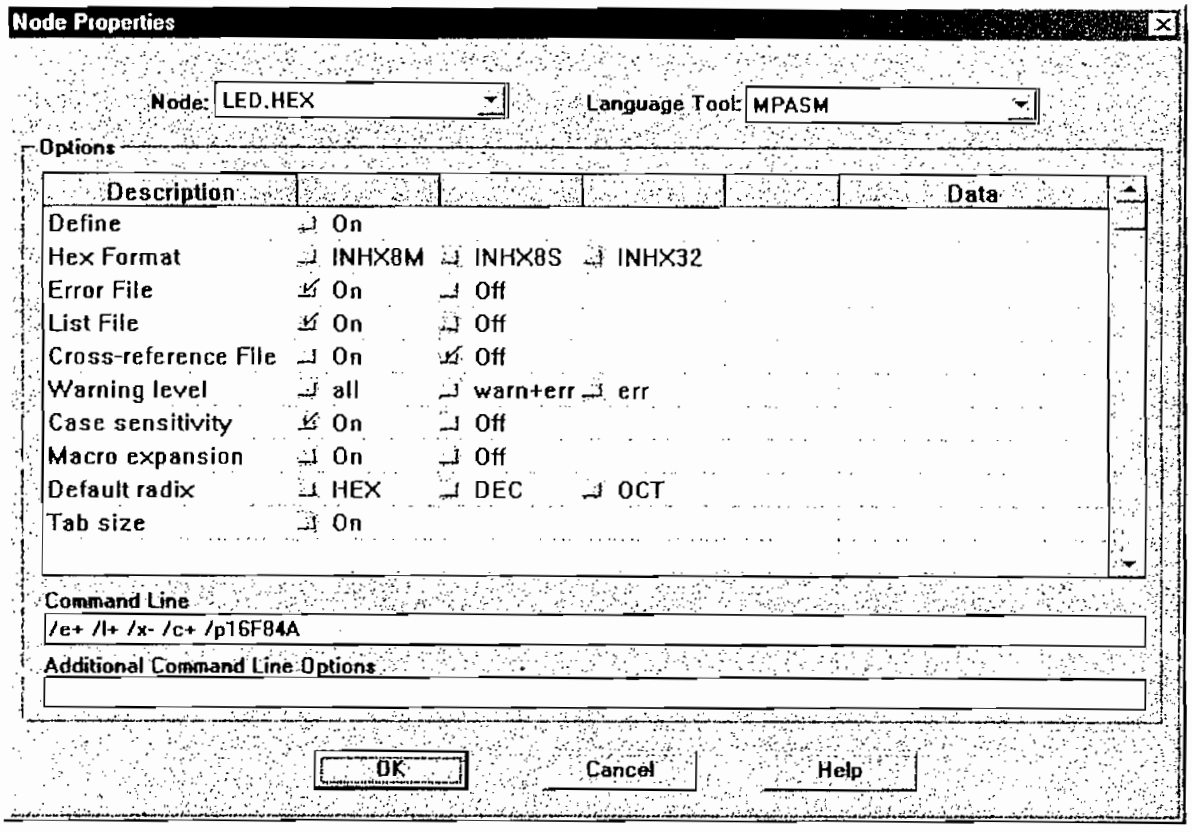


Figura 2.7 Cuadro de propiedades de nodo

Las propiedades de nodo normalmente no requieren ser modificadas salvo en algún caso especial por programadores experimentados.

Si se cambia el nombre del archivo fuente (.asm), también se debe cambiar el nombre del proyecto. Una vez terminado este proceso, se está listo para simular el programa que se almacenará en el microcontrolador PIC para que cumpla una función determinada.

2.5 EDICIÓN DEL CÓDIGO FUENTE

Para iniciar con el uso de microcontroladores PIC en la solución de problemas de control, se requiere por un lado aprender el diseño del diagrama de flujo, y por otro lado la escritura de las líneas de programa así como su simulación. El diseño de los diagramas de flujo se lo analizará en el desarrollo de cada una de las prácticas planteadas en esta tesis. Para la edición del programa se requiere cualquier editor de texto, el software MPLAB de Microchip incluye un editor de texto, esto facilita la escritura del programa dentro del proyecto; en los

pasos citados anteriormente ya se abrió la ventana de edición llamada led.asm.

Para esto se debe tomar en cuenta que en cada línea de programa se puede tener hasta cuatro tipos de información: Etiquetas, nemónicos o comandos, operandos y comentarios. El orden y posición de estos es importante, y se las ubica en columnas. Las etiquetas deben ir en la primera columna; los nemónicos o comandos pueden ir ubicados en la columna dos o siguiente; los operandos van luego de los nemónicos separados por lo menos por un espacio en blanco o en la columna siguiente, múltiples operandos deben ir separados por comas; los comentarios van después de los operandos, nemónicos o etiquetas, estos pueden ir ubicados en cualquier columna; el ancho máximo de cada columna es de 255 caracteres.

Etiquetas:

Son palabras o caracteres definidos por el usuario a los cuales se los asigna un valor numérico con el comando equ o también sirven para indicar la dirección de una línea del programa, en este caso no es necesario definirlo con el comando equ..

El comando equ tiene la siguiente sintaxis:

<Etiqueta> equ <expresión>

Descripción:

El valor de <expresión> es asignado a la etiqueta

Ejemplo:

Cuatro equ 4 ; Asigna el valor numérico 4 a la etiqueta Cuatro

Como ejemplo del uso de etiquetas para indicar la dirección de una línea de programa analicemos el siguiente programa:

Dirección	Etiqueta	nemónico	operando	comentario
00h		goto	inicio	;salta a la línea etiquetada
				;como inicio
01h				
02h				
03h				
04h		goto	subrutina	;salta a la línea etiquetada

```
                                                ;como subrutina
05h     inicio     bsf     status,5             ;pasa al segundo banco
.
.
.
20h     subrutina  movf    ptob,w               ;carga w con ptob
```

En este programa las etiquetas inicio y subrutina toman el valor de la dirección de la línea, entonces inicio = 05h y subrutina=20h. Esto facilita la ejecución de saltos pues no se requiere indicar a que dirección debe saltar, sino tan solo se indica la etiqueta de la línea a la que debe realizar el salto.

Nemónicos o comandos

Nemónicos son las instrucciones de los microcontroladores. Estos comandos van ubicados en la columna dos.

Operandos

Son los valores sobre los que actúan las instrucciones, estos pueden ir colocados en la columna tres; los distintos formatos que estos pueden tener se indican en la Tabla 4.1 siguiente:

Tipo	Sintaxis	Ejemplo
Decimal	D'<digits>'	D'100'
	.<digits>	.100
Hexadecimal	H'<hex_digits>'	H'9f'
	0x<hex_digits>	0x9f
	<digito>h	0ch
		1Ah
Si el valor inicia con una letra se antepone el 0		0A2h
Octal	O'<octal_digits>'	O'777'
Binario	B'<binary_digits>'	B'00111001'
ASCII	A'<character>'	A'C'
	'<character>'	'C'

Tabla 4.1 Formato de operandos

Comentarios

Se ubican en cualquier columna luego de los operandos, estos son ignorados el momento de compilar el programa; sin embargo, son muy útiles en el desarrollo del programa para explicar el objetivo de las instrucciones escritas, esto facilitará también modificaciones futuras.

Además, es importante el uso de los comandos list, org, y end indicados a continuación:

El comando LIST

Selecciona algunas opciones del ensamblador; entre ellas esta la selección del tipo de procesador a utilizar. Esto se indica mediante la variable "p", por ejemplo, para indicar que el microcontrolador es el PIC16F84A se coloca la línea siguiente ubicada en la segunda columna:

Sintaxis: list p=<tipo>

Ejemplo: list p=16F84A

El comando ORG

Indica la dirección de memoria de programa en la cual se escribirán las subsecuentes líneas de programa, este comando tiene la siguiente sintaxis:

[<Etiquetal>] org <expresión>

Ejemplo:

```
Reset     org      0           ;el vector del reset es la línea cero
          goto     inicio      ;salta al inicio del programa
          org      4           ;vector de interrupciones
          goto     subrutina1  ;salta a la subrutina de atención a
                               ;interrupciones
          org      5           ; el programa inicia en la dirección 5
inicio    bsf      status,5    ;inician las instrucciones del programa.
          Movlw    0F0h        ;se carga w con F0h
```

Descripción:

En la dirección de memoria 0 que es el vector de reset, se escribirá la instrucción goto inicio, esta hará que luego de resetear el sistema el programa salte a la línea

5 en donde está la etiqueta inicio; en la línea 4, que es el vector de interrupciones, se escribirá la instrucción goto subrutina1. Esto hará que cuando se produzca una interrupción y el programa salte a la línea etiquetada como subrutina 1, esta será la subrutina de atención a interrupciones. En la línea 5 se escribirá la instrucción bst status,5 que es ya el inicio del programa principal, en la línea 6 la instrucción movwl 0F0h, y así sucesivamente las siguientes instrucciones.

El comando end

Indica la finalización del programa, al igual que todos los comandos deben ser escritos en la segunda columna.

El comando include

Este comando permite adicionar un archivo en el cual se puede listar todas las etiquetas del microcontrolador. Esto facilita la escritura de programas puesto que se puede tener un archivo estándar de todas las etiquetas de los registros de función especial, y sus respectivos bits para cada microcontrolador, la cual puede ser utilizada en todos los programas que se realicen con este modelo de microcontrolador. Este archivo debe tener la extensión .inc y ser ubicada al inicio del programa. El programa MPLAB incluye en sus librerías archivos estándar de etiquetas para todos los modelos de microcontroladores, los cuales pueden ser utilizados en la realización de los programas. Como ejemplo se puede ver en algunas de las prácticas propuestas en esta tesis, el uso de este comando.

Además la estructura de los programas debe seguir el siguiente formato:

1) Comentarios con la descripción del objetivo del programa
2) Indicación del modelo de microcontrolador a utilizar con el comando list
3) Definición de etiquetas con el comando equ
4) Indicación de las direcciones de escritura del programa con el comando org
5) Escritura de subrutinas o tablas
6) Escritura del programa principal
7) Finalización del programa con el comando end

En el caso de que un programa no requiera de subrutinas, entonces se obviará el paso número 6.

Esta estructura se podrá observar en las prácticas planteadas en el capítulo IV .

2.6 ENSAMBLE DEL ARCHIVO FUENTE

Una vez escrito el programa para ensamblarlo se utiliza el item del menú "Proyecto>construir todo" (Proyect>Build All), de este modo se ejecutará el programa MSASM que siempre genera un archivo con extensión .hex de salida con el mismo nombre que el archivo fuente, y esta configuración no puede modificarse. Una vez completado el proceso de ensamblaje, aparecerá la ventana "Resultados de construcción" (Build Results) mostrado en la figura 2.8.

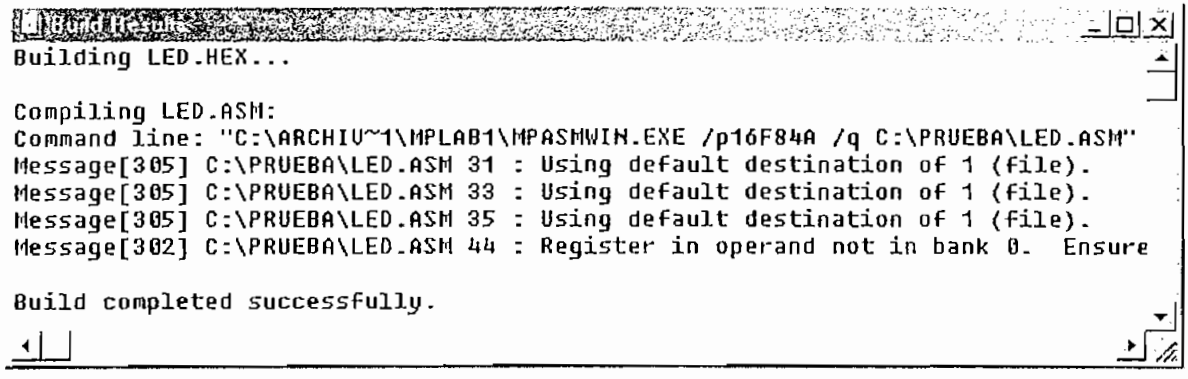


Figura 2.8 Resultado del ensamblaje del código fuente

Se debe notar que al final de esta ventana aparece un mensaje que indica que el ensamblaje se ha completado satisfactoriamente. En caso de existir fallas el mensaje será "construcción fallida" (Build failed), además aparecerán los mensajes de error que al hacer un doble clic sobre estos automáticamente el cursor se ubica sobre la línea que contiene el error en la ventana del código fuente. Una vez corregido todos los errores se debe ensamblar otra vez el archivo fuente utilizando nuevamente el ítem Proyecto>Construir todo. Normalmente este proceso se repite muchas veces hasta corregir todos los errores.

2.7 SIMULACIÓN

Sin duda la herramienta más útil de este paquete es su simulador. Una vez que el ensamblaje se ha completado satisfactoriamente para ponerlo a funcionar paso a paso se debe seleccionar el ítem "Debug>Run>step" y en seguida aparecerá la ventana del código fuente con la primera línea a ser ejecutada sombreada como se observa en la figura 2.9.

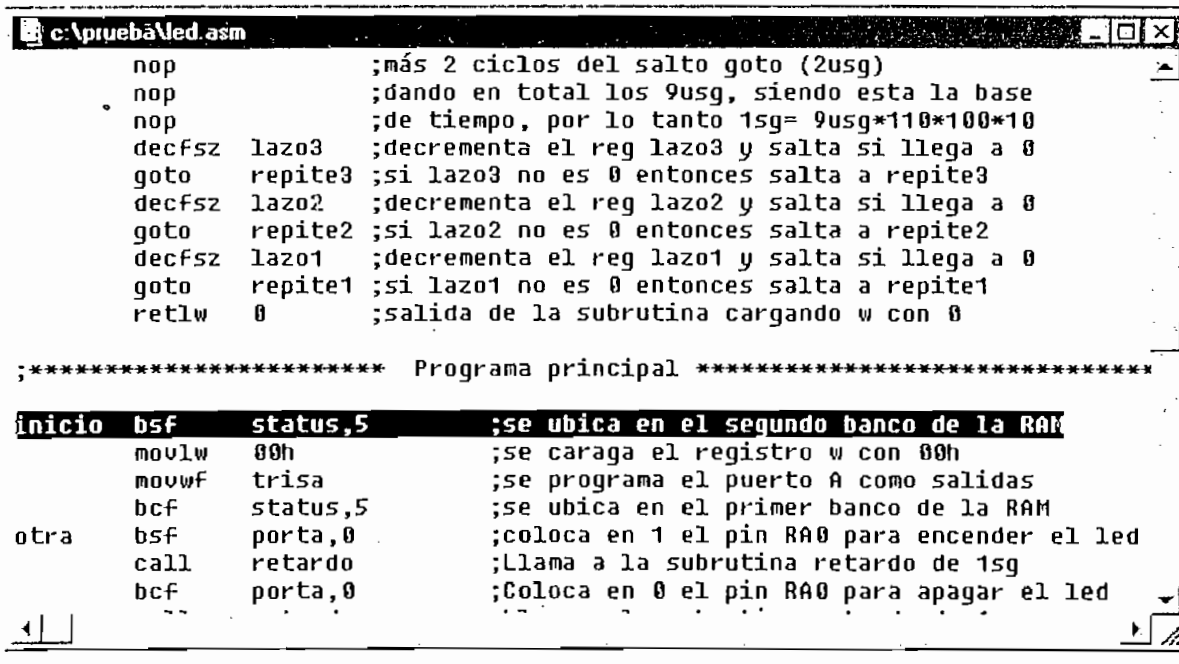


Figura 2.9 Pantalla de inicio de simulación

Para continuar con la simulación paso a paso vasta con pulsar la tecla F7 cada vez, y se verá como avanza la posición de la línea sombreada. Otra forma de ejecutar el programa es mediante el item "Debug>Run>Run" o con la tecla F9, que hará que el programa corra de continuo a la máxima velocidad que pueda el procesador del computador ejecutar las instrucciones, esto depende de las características del PC utilizado. Los colores de la barra de estado cambian para indicar este modo de operación. Para detener la ejecución del programa se bebe seleccionar el ítem "Debug>Run>Halt" (detener) o con la tecla F5. También se puede correr en modo de animación en la que se ejecuta el programa a una velocidad prudente para poder observar los cambios producidos por cada instrucción, esto se hace con el ítem "Debug>Run>Animate" o con las teclas

Control+F9. Si se desea reiniciar la ejecución del programa se puede hacer con el ítem "Debug>Run>Reset" o con la tecla F6. Todas estas funciones y otras más se las puede ejecutar también presionando los botones de la barra que aparece en la parte superior junto con la barra principal.

2.7.1 OTRAS VENTANAS DE OBSERVACIÓN

Existen varias ventanas que permiten visualizar el programa y su ejecución, una de estas es la de registros de función especial, la que permite observar el cambio en todos los registros especiales como el de trabajo W, el contador de programa PC y otros, esta ventana se abre seleccionado el ítem Windows>Special Funtion registers. En todas estas ventanas se puede observar que cuando un valor es alterado su color cambia a rojo, permitiendo notar de mejor manera los cambios producidos. Otra ventana muy útil es la ventana de archivo de registros, la cual permite observar toda la memoria RAM, esta se abre seleccionando el ítem "Windows>file registers" . Además se pueden crear ventanas personalizadas de observación, en las cuales se puede seleccionar los registros a observar y su formato a presentar, esto se hace seleccionando el ítem Windows>Watch Windows>New Watch Windows. Todas estas ventanas abiertas se las puede observar a continuación en la Figura 2.10.

2.7.2 PUNTOS DE INTERRUPCIÓN

En muchas ocasiones es necesario detener la ejecución en un sitio determinado para analizar lo que ha sucedido hasta ese instante y proseguir en el modo paso a paso con las líneas siguientes, para esto simplemente se debe ubicar el puntero del mouse en la línea deseada y con el botón derecho se abre una ventana en la cual se selecciona "Break Point" (punto de parada), entonces la línea cambia de color a rojo. Una vez seleccionados los puntos de parada se ejecuta el programa en modo continuo o de animación y se observará que el programa se detiene automáticamente al llegar a los puntos de parada.

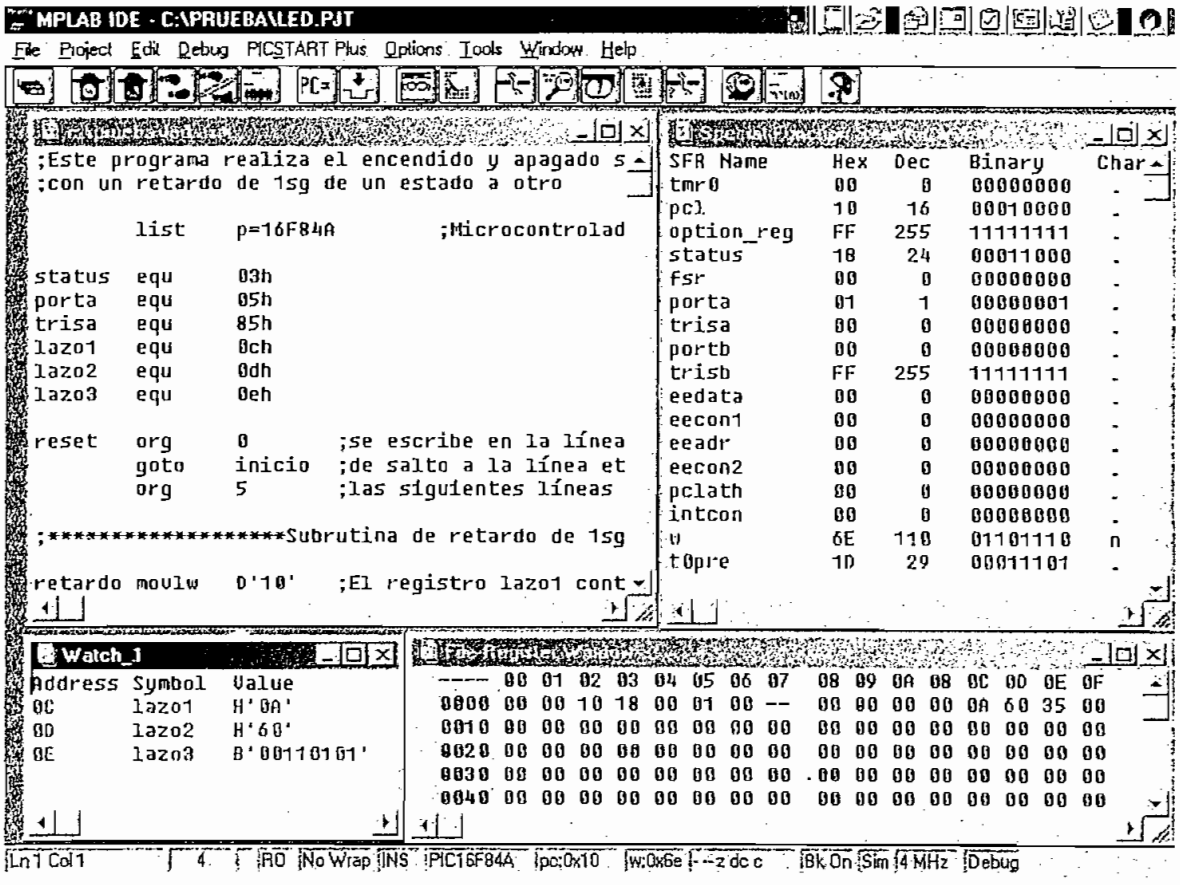


Figura 2.10 Ventanas de observación

2.7.3 ESTÍMULOS

En muchos casos en necesario simular el ingresos de señales externas al microcontrolador, para realizar esto el sistema de desarrollo MPLAB posee varias opciones como son: estímulo asincrónico, archivo de estímulo de Pin, estímulo de reloj y archivo de estímulo.

Estímulo asincrónico: Esta característica provee un cuadro de dialogo para simular el ingreso de un cero o uno lógico en un pin de entrada en cualquier momento. Este cuadro es desplegado seleccionando el item (Debug>Simulator Stimulus>Asynchronous Stimulus) de la barra de herramientas, este cuadro se puede observar en la Figura 2.11 a continuación.

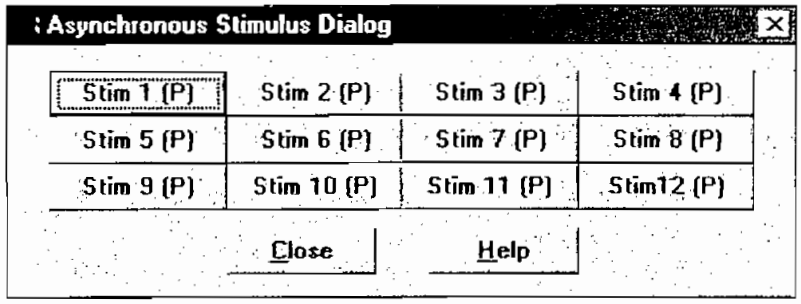


Figura 2.11 Cuadro de estímulos asincrónicos

Cada uno de los botones pueden ser configurados para un de los pines del micro. Al presionar el botón derecho del mouse en estos botones aparece el cuadro de la Figura 2.12 en la cual es preferible seleccionar la opción Toggle, que permite el cambio del estado del pin seleccionado.

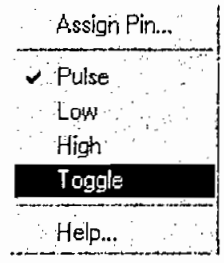


Figura 2.12 Cuadro de selección de modo de operación

En un segundo paso, presionando el botón derecho del mouse sobre el botón (Stim 1 (T)), que ya cambió de P a T en el paso anterior, se debe seleccionar la opción "Assign Pin" para asignar el pin que manejará ese botón, entonces aparecerá el cuadro de la Figura 2.13.

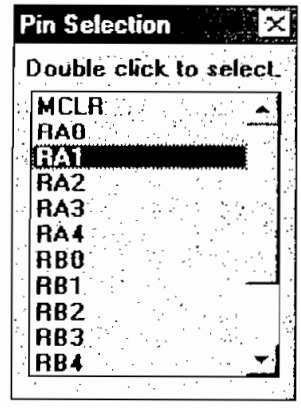


Figura 2.13 Cuadro de asignación del pin

Una vez configurados los botones del cuadro de estímulos asincrónicos, estos pueden ser utilizados en cualquier momento durante la ejecución del programa.

Archivo de estímulos para un pin (Pin Estimulo): Un archivo de estímulo pin está conformado por columnas de unos y ceros de ingreso, que serán aplicadas a los pines cuando el valor de la columna "Cycle" alcance el conteo del reloj del sistema.

Este archivo puede ser editado usando el mismo editor del paquete MPLAB, mediante el item "Archivo>Nuevo archivo" y su formato debe ser como el de la Figura 2.14 en la cual la columna cycle contiene el valor del número de ciclos que pasarán, para que los valores de cada línea de la tabla ingrese a los pines RB0 y RB1.

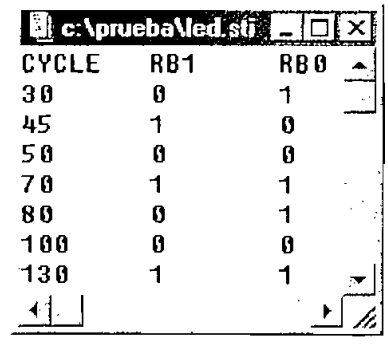


Figura 2.14 Archivo de estímulos para un pin

Para activar esta función es necesario seleccionar el item (Debug>Simulator Stimulus>Pin Stimulus>Enable), que desplegará una ventana estándar de búsqueda de archivos, mediante la cual se selecciona el archivo creado que debe tener la extensión .sti, en este ejemplo se ha llamado led.sti. Entonces, al realizar la ejecución paso a paso del programa se podrá observar el cambio en los pines RB0 y RB1 al llegar a cumplirse el número de ciclos de la tabla.

Para poder observar los ciclos transcurridos se puede abrir la ventana de reloj seleccionando el item (Windows>Stopwatch) cuyo cuadro se observa en la Figura 2.15 a continuación.

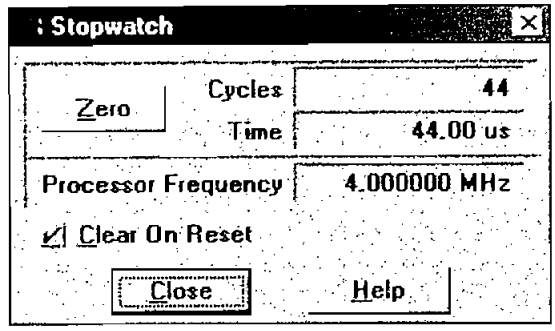


Figura 2.15 Ventana de reloj

Archivo de Estímulos de Registro: Un archivo de estímulos de registro está formado por una columna de los valores que serán enviados a un registro, cuando la dirección de memoria de programa alcance la ubicación establecida en el dialogo Estímulo de Registro. Esto es útil para simular una operación A/D. Para esto se debe crear un archivo como el de la Figura 2.16 cuya extensión debe ser .reg, en este ejemplo se llama led.reg. Este archivo contiene una columna de valores hexadicimales

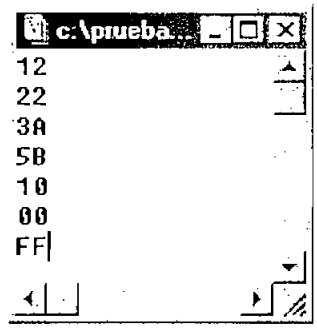


Figura 2.16 Archivo de estímulo de registro

Para activar el funcionamiento de este archivo es necesario seleccionar el item (Debug>Simulator Stimulus>Register Stimulus>Enable) y se abrirá entonces la ventana de la Figura 2.17 en la que se debe seleccionar el registro sobre el cual actuará el archivo, que en este caso se ha elegido el puerto B. También se debe ingresar la dirección de la memoria de programa en la que se ingresará el valor de la tabla y por último en el botón Browse se debe seleccionar el archivo led.reg guardado previamente, entonces se puede ejecutar el programa paso a paso para observar el ingreso de los valores de la tabla del registro seleccionado.

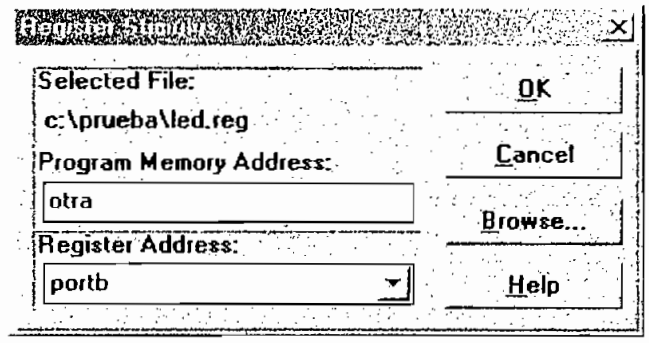


Figura 2.17 Cuadro de activación del estímulo de registro

Estímulo de reloj: El estímulo de reloj genera una onda regular sobre un pin con un ciclo obligatorio, especificado en términos de los ciclos del reloj del procesador. Si se abre el cuadro (Debug>Simulator Stimulus>Clock Stimulus), se podrá ingresar varios estímulos de reloj como se observa en la Figura 2.18

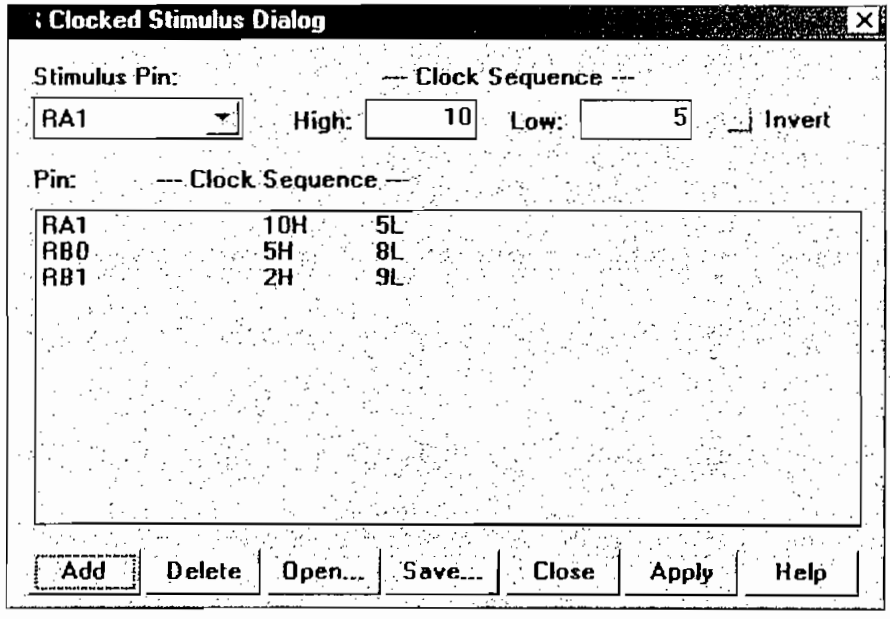


Figura 2.18 Cuadro de estímulos de reloj

2.7.4 OPCIONES VARIAS DE SIMULACIÓN

Cuadro de modificación de valores: una herramienta muy útil es el cuadro de modificación de valores, que permite variar el valor de cualquier registro de la memoria RAM, incluidos los de función especial. Este cuadro se abre seleccionando el item (windows>Modify) y se observa en la Figura 2.19. También

se puede abrir esta ventana mediante un doble clic sobre el registro deseado en cualquiera de las ventanas de observación.

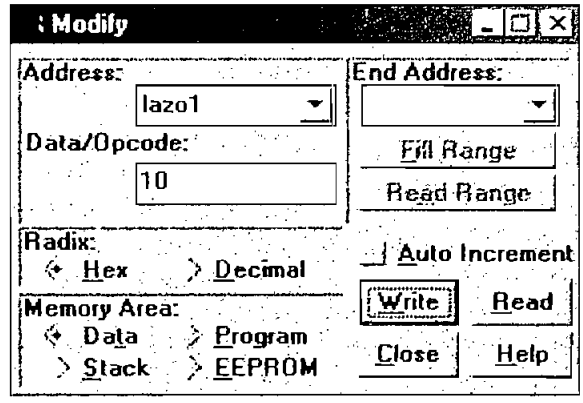


Figura 2.19 Cuadro de modificación de registros

Modificación de contador de programa: mediante el icono de la barra de herramientas cuyo símbolo se observa en la parte superior de la Figura 2.20, se puede modificar el valor del contador de programa.

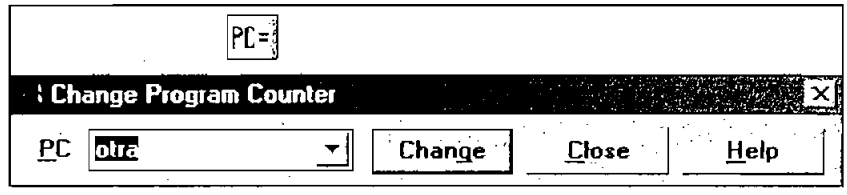


Figura 2.20 Cuadro de modificación del contador de programa

Ejecución de una instrucción: mediante el icono de la barra de herramientas, cuyo símbolo se observa en la parte superior de la Figura 2.21, se puede ejecutar una instrucción en cualquier momento durante el proceso de simulación.

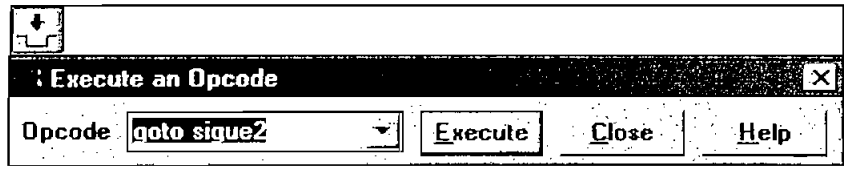


Figura 2.21 Cuadro de ejecución de instrucciones

Existen otras opciones que no son muy utilizadas en el proceso de simulación, estas pueden ser analizadas en caso de ser necesario mediante las ayudas que incluye el paquete MPLAB.

CAPITULO III

DISEÑO Y CONSTRUCCIÓN DEL SISTEMA

3.1 CONFIGURACIÓN GENERAL

El sistema de desarrollo propuesto incluye todos los elementos para la evaluación de aplicaciones basadas en microcontroladores PIC de la gama enana de 8 pines PIC12C50X, y la gama media de 18, 28 y 40 pines PIC16C(F)8X/6X/7X/9X, PIC16F87X; se hará mayor énfasis en el microcontrolador PIC16F84A debido a que posee memoria de programa tipo Flash que permite una rápida escritura y borrado de programas. El sistema cuenta con los periféricos más comunes, zócalos para alojamiento de los varios modelos de microcontroladores con la circuitería básica ya implementada, y un programador en circuito que permite la programación de los microcontroladores sin sacarlos de los zócalos agilitando la depuración de programas.

Con todos estos elementos se puede realizar una gran cantidad de prácticas, gracias a que el sistema tendrá una arquitectura abierta; es decir, no se dejará implementado el cableado de los puertos del microcontrodador hacia los periféricos, de esta manera queda abierta la posibilidad de conectar mediante cables externos cualquiera de los periféricos y en cualquier configuración. Esto permite, por ejemplo, que se pueda realizar prácticas conectando el módulo LCD con interface a 8 bits o a 4 bits.

La implementación del cableado entre los pines del microcontrolador y los de periféricos lo realizará el usuario, mediante un mínimo cableado de fácil colocación.

Los módulos a implementarse son los siguientes:

- Fuente de alimentación regulada de 13.4VDC, 12VDC y 5VDC

- Módulo de conexión de microcontroladores
- Módulo de leds
- Módulo de displays de 7 segmentos
- Módulo de Display de Cristal Líquidos LCD
- Módulo de dipswitch
- Módulo de teclado matricial
- Módulo de pulsadores
- Módulo de comunicaciones seriales RS232
- Módulo de comunicaciones seriales RS-422/485
- Módulo de memoria serial
- Módulo de entrada de señales analógicas
- Modulo de potenciómetro digital
- Módulo zumbador piezoeléctrico
- Módulo de salida PWM
- Módulo de reloj de tiempo real I^2C
- Módulo de relé
- Módulo de programador en circuito

3.2 DIAGRAMA DE BLOQUES

En la Figura 3.1 se presenta el diagrama de bloques del sistema de desarrollo en el cual se pueden identificar los módulos antes descritos:

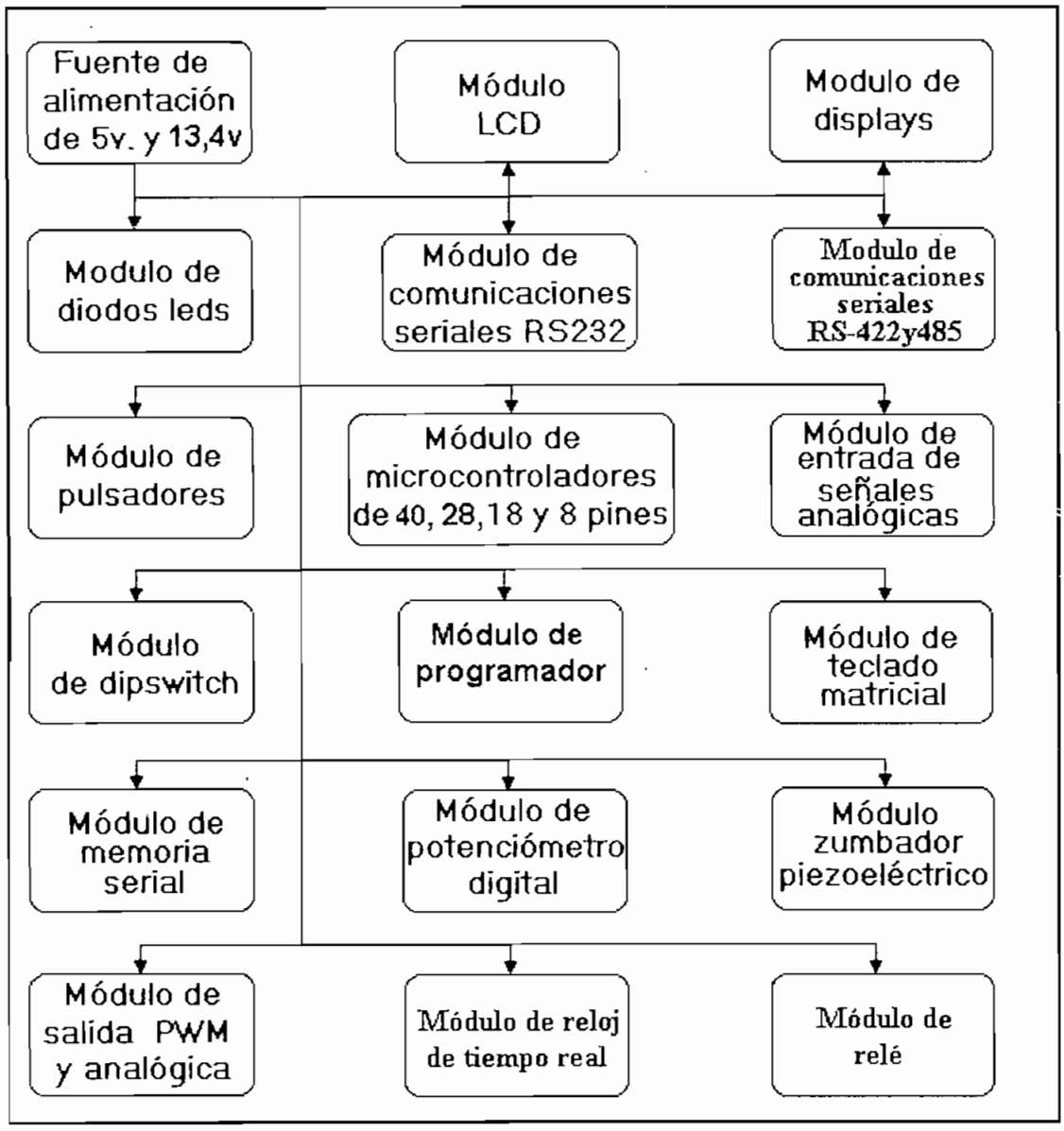


Figura 3.1 Diagrama de bloques del sistema de desarrollo

3.3 DISEÑO CIRCUITAL

El sistema de desarrollo propuesto en este proyecto está formado por módulos independientes en los cuales su circuitería y líneas de alimentación están completamente implementadas, a excepción de los buses de datos y control cuyos terminales quedarán listos para su conexión en borneras redondas para cables tipo telefónico de 0.5mm de diámetro. Esto permite un fácil armado de los circuitos de las practicas y gran versatilidad para la realización de múltiples proyectos, en los cuales se podrán utilizar cualquiera de los módulos según los requerimientos de la práctica a realizar.

Tomando en cuenta que el objetivo de las prácticas no es aprender a armar circuitos, que tampoco son complejos, sino mas bien es realizar las pruebas de los programas que debe ejecutar el microcontrolador, entonces las ventajas del sistema de desarrollo propuesto son optimizar el tiempo de armado de los circuitos y disminuir la probabilidad de cometer errores en las conexiones que incluso pueden provocar la destrucción del microcontrolador y otros periféricos.

A continuación se describe el diseño de cada uno de los módulos que conforman el sistema de desarrollo:

3.3.1 FUENTE DE ALIMENTACIÓN:

La fuente de alimentación debe proveer tres salidas, una de 13.4V regulada requerido como voltaje de programación de los microcontroladores, 12V para energizar un relé u otro dispositivo que lo requiera y otra de 5V también regulada para energizar el microcontrolador y demás periféricos. El uso de reguladores integrados de voltaje garantiza la protección del sistema de desarrollo frente a eventuales cortocircuitos pues estos elementos integran un circuito que limita la corriente a niveles seguros al estar en cortocircuito. Además incluyen circuitos de regulación térmica, un comparador de voltaje, un amplificador de error, un voltaje de referencia , una etapa de salida de referencia, una etapa de salida de potencia y un limitador de corriente.

Alguna de las características más importantes de los reguladores que se utilizarán en la fuente son los siguientes:

- Voltaje de salida de 5V para el regulador de 5V LM7805
- Voltaje de salida de 12V para el regulador de 12V LM7812
- Corriente máxima de salida 1A.
- Regulación típica de línea 0.01%
- Regulación típica de carga 0.1%
- Limitación constante de corriente con respecto a la temperatura
- Máxima disipación de potencia: 15W
- Encapsulado estándar tipo transistor de tres terminales (TO-220).

- Rechazo de rizado 80dB.
- Voltaje de entrada mínimo de 7V y máximo de 35V para el LM7805.
- Voltaje de entrada mínimo de 14V y máximo de 35V para el LM7812.

Para la fuente de 5V se necesitan unos pocos elementos adicionales al regulador. En la Figura 3.2 se puede observar el diagrama esquemático, en donde se puede observar la interconexión de estos elementos.

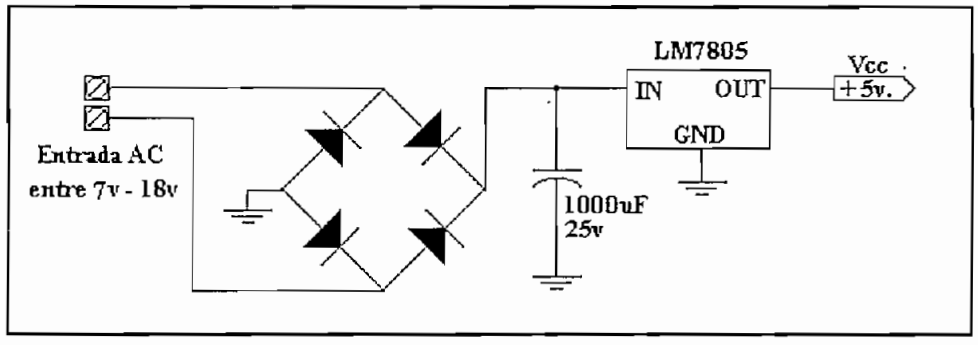


Figura 3.2 Fuente de alimentación regulada de 5V

La fuente de 5V del sistema de desarrollo suministrará una corriente variable según el circuito a implementar y según la Tabla 3.1 de consumo de corrientes.

Elemento	Consumo de corriente (mA)
Circuito de reset	4
Microcontrolador PIC16F84A	1,8
Módulo de leds	63
Módulo de displays	73
Módulo LCD	80
Módulos de comunicaciones	40
Módulo de memoria serial	40
Módulo potenciómetro digital	40
Módulo zumbador piezoeléctrico	10

Tabla 3.1 Consumo de corriente de elementos del sistema de desarrollo

Se puede ver que la corriente normalmente no excederá los 300mA, sin embargo, la fuente de poder será diseñada para entregar una corriente de hasta 400mA. Para esto, debido a que la fuente trabaja mediante un regulador de voltaje, se debe entonces asegurar que el voltaje de entrada nunca sea inferior al voltaje mínimo de ingreso del regulador de 7V, o superior al valor máximo de 35V.

Como se puede observar, el voltaje de ingreso al regulador es generado por un rectificador de onda completa con un filtro capacitivo, cuyo voltaje Vdc está dado por la Ecuación 3.1 indicada a continuación:

$$Vdc = Vp - \frac{Idc}{4fC} - 1.4$$

$$Vdc = \sqrt{2} * Vrms - \frac{4.17 * Idc}{C} - 1.4$$

Ecuación 3.1

Despejando de esta fórmula el voltaje Vrms de entrada al puente rectificador se obtiene la Ecuación 3.2:

$$Vrms = \frac{Vdc + \left(\frac{4.17 * Idc}{C}\right) + 1,4}{\sqrt{2}}$$

Ecuación 3.2

Con esta fórmula se puede encontrar el valores máximo y mínimo de voltaje eficaz con el que se puede alimentar a este circuito.

Para el valor mínimo: Vdc=7V con una corriente máxima de 400mA.

$$Vrms = \frac{7 + \left(\frac{4.17 * 400}{1000}\right) + 1,4}{\sqrt{2}}$$

$$Vrms = 7.12V$$

Para el valor máximo: Vdc=35V con una corriente mínima de 0A

$$Vrms = \frac{35 + \left(\frac{4.17 * 0}{1000}\right) + 1,4}{\sqrt{2}}$$

$$Vrms = 25.73V$$

Por lo tanto se puede energizar el rectificador con una fuente de corriente alterna cuyos valores estén entre 7,12V y 25,73V. Por conveniencias económicas

el voltaje de entrada se limitará a 18v, lo cual permite que el condensador utilizado como filtro de 1000µF sea diseñado para un voltaje de 25V, caso contrario se debería utilizar un condensador de 1000µF a 50v el cual es más voluminoso y caro. Tomando en cuenta que el voltaje de ingresos mínimo es mejor, ya que la diferencia de voltaje entre el valor mínimo y el entregado por el transformador debe ser absorbido por el regulador, provocando calentamiento del disipador, entonces se energizará con una fuente A.C. de 7,5V.

Para la fuente de 13.4V se necesitan unos pocos elementos adicionales al regulador. En la Figura 3.2 se puede observar el diagrama esquemático, en donde se ve que aunque se utiliza un regulador de 12V, se consigue subir a 13,4V mediante la adición de dos diodos en el terminal que conecta a tierra, que suben el nivel de referencia del regulador a 1,4V con lo que sumado a los 12V del regulador se obtienen los 13,4V necesarios como voltaje de programación.

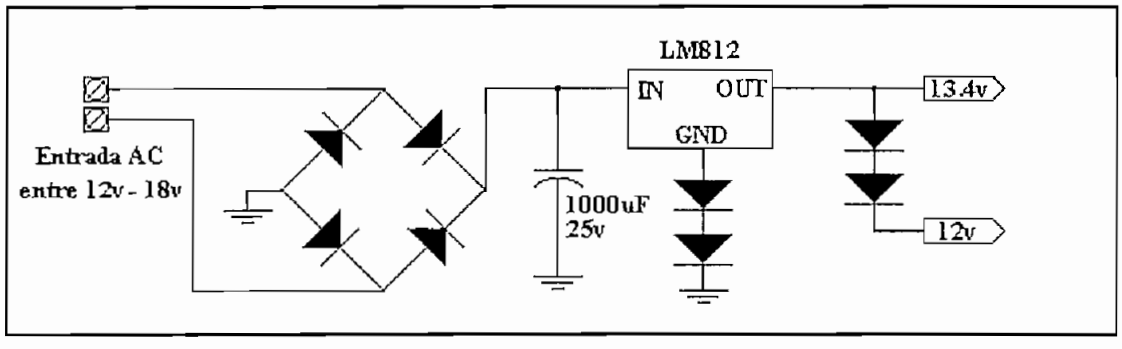


Figura 3.3 Fuente de alimentación regulada de 13,4V y 12V

La fuente de 13.4V del sistema de desarrollo suministrará una corriente muy pequeña para realizar la programación de los microcontroladores, por lo mismo se vio que se podía emplear la misma, mediante dos diodos para reducir a 12V, para energizar el relé incluido en el sistema y otros dispositivos que lo requieran; esta fuente fue diseñada para una corriente máxima de 100mA.

Para esto, debido a que la fuente trabaja mediante un regulador de voltaje, se debe entonces asegurar que el voltaje de entrada nunca sea inferior al voltaje mínimo de ingreso del regulador de 14V, o superior al valor máximo de 35V; tomando en cuenta que la referencia del regulador sube 1,4V debido a los dos diodos, entonces se tiene un voltaje mínimo de 15,4V y un máximo de 36,4V.

Para el valor mínimo: Vdc=15,4V con una corriente máxima de 100mA.

$$Vrms = \frac{15,4 + \left(\frac{4.17 * 100}{1000}\right) + 1,4}{\sqrt{2}}$$

$$Vrms = 12,7V$$

Para el valor máximo: Vdc=36,4v con una corriente mínima de 0A

$$Vrms = \frac{36,4 + \left(\frac{4,17 * 0}{1000}\right) + 1,4}{\sqrt{2}}$$

$$Vrms = 26.72V$$

Por lo tanto se puede energizar el rectificador con una fuente de corriente alterna cuyos valores estén entre 12,7V y 26,72V. Por conveniencias económicas el voltaje de entrada se limita a 18V, lo cual permite que el condensador utilizado como filtro de 1000µF sea diseñado para un voltaje de 25V, caso contrario se debería utilizar un condensador de 1000µF a 50v el cual es más voluminoso y costoso. Tomando en cuenta que el valor más bajo posible es mejor, ya que la diferencia de voltaje entre el valor mínimo y el entregado por el transformador debe ser absorbido por el regulador, provocando calentamiento del disipador, entonces se alimentará con una fuente A.C. de 13v.

Además se incluye un interruptor de encendido general y un fusible de protección de 1A.

3.3.2 MÓDULO DE CONEXIONES DE MICROCONTROLADORES

Este módulo está formado por cuatro zócalos, de 40, 28, 18 y 8 pines para los microcontroladores, PIC16F87x, PIC16F84A, PIC16C71 y PI12C508 o sus versiones pin a pin compatibles. En este circuito se integra también la conexión de la alimentación VDD y VSS a 5v y tierra respectivamente, además se implementa dos circuitos osciladores con cristales de 4MHz. para circuitos en los cuales se requiera tiempos exactos de ejecución de ciclos de instrucción de 1µs, que es muy útil en la mayoría de prácticas, y también otro circuito oscilador con un cristal de 32.768Hz que permite realizar conteo de tiempos exactos necesario en prácticas con temporizadores. La selección de estos dos circuitos osciladores se la realiza muy fácilmente mediante dos jumpers y están conectados solamente a

los zócalos de 40, 28 y 18 pines ya que para los microcontroladores de 8 pines no se requiere circuito de reloj, debido a que lo poseen internamente.

El circuito de este módulo se lo puede observar en la Figura 3.4 a continuación:

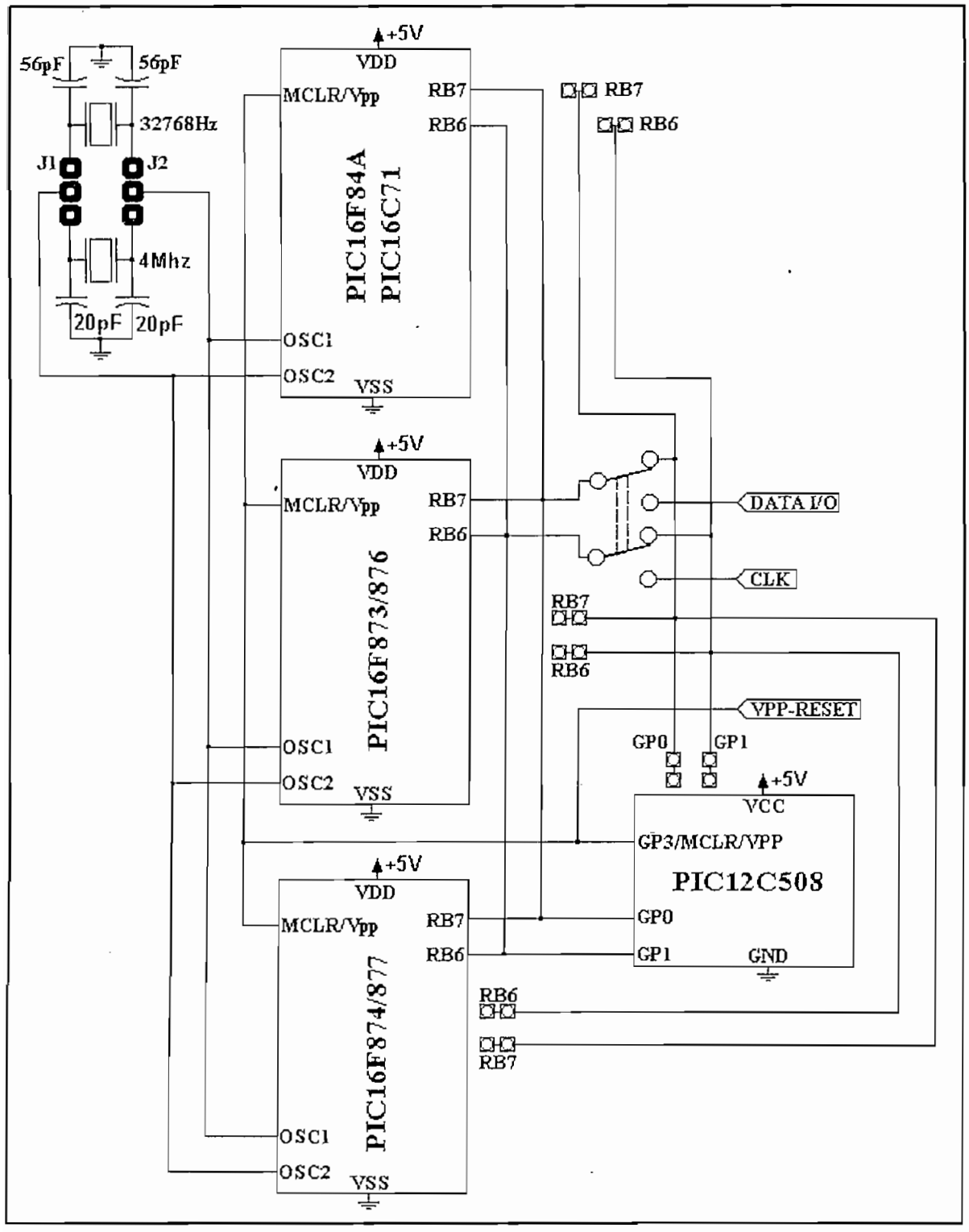


Figura 3.4 Módulo de conexión de microcontroladores

3.3.3 MÓDULO DE LEDS

Este módulo muy sencillo permitirá realizar ejercicios de manejo de diodos leds en las prácticas básicas; está compuesto por un buffer ULN2803 (ECG2018) para no forzar al microcontrolador con corrientes límites, con esto se manejan 8 diodos leds. El valor de la resistencia limitadora de corriente, tomando en cuenta que con una corriente de 7mA se logra un encendido adecuado de los leds y que se produce una caída de voltaje en el bufer de 0,7V, está dada por:

$$R = \frac{(Vdd - Vdiodo)}{I}$$

$$R = \frac{(5-2-0,7)}{0.007}$$

$$R = 328\Omega$$

$$\Rightarrow R = 330\Omega$$

Por lo tanto se colocan 8 resistencias limitadoras de 330Ω. En la Figura 3.5 se muestra el circuito de conexión de estos elementos.

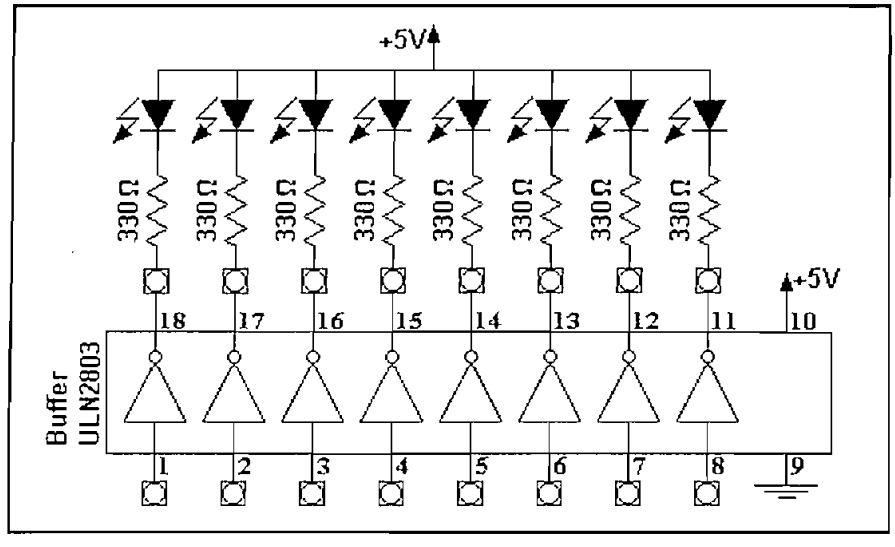


Figura 3.5 Circuito de conexión del módulo de leds

3.3.4 MÓDULO DE DISPLAYS DE 7 SEGMENTOS.

Este módulo permite experimentar con 4 displays de 7 segmentos de cátodo común, conectados sobre un mismo bus de datos y con los cátodos manejados de forma independiente. Para la conversión del dato que entrega el microcontrolador al código de 7 segmentos, se emplea un decodificador 7447; de esta manera se ahorran al menos 4 líneas (solo tres si se utiliza una adicional para manejar el punto decimal). Todos los segmentos de los displays se encuentran conectados a los pines de salida del 7447. Por lo tanto; el dígito que se desea mostrar en ese momento se debe seleccionar mediante la habilitación del display correspondiente. Esto se hace colocando un nivel lógico alto en la base del transistor NPN que maneja el cátodo de dicho display. Los 4 bits del dígito que se va a mostrar se debe conectar desde el microcontrolador a los pines marcados D1 a D4 correspondientes a las bases de los transistores que habilitan los displays. En la Figura 3.6 se muestra el diagrama esquemático de este módulo.

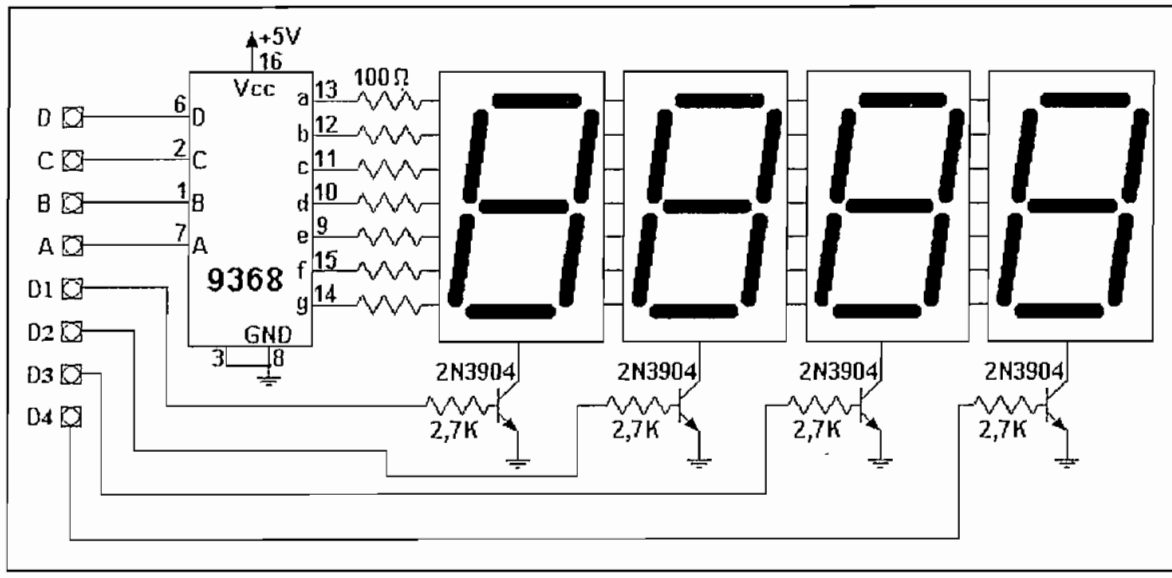


Figura 3.6 Circuito de conexión del módulos de displays

3.3.5 MÓDULO DE DISPLAY DE CRISTAL LÍQUIDO, LCD

Este circuito permite la experimentación con displays de cristal liquido, para visualizar mensajes que comúnmente se requiere en el control de sistemas.

Además, estos elementos presentan muchas ventajas sobre otros medios de visualización, como por ejemplo, los displays de siete segmentos no pueden presentar caracteres alfanuméricos y tienen un alto consumo de corriente mientras que los módulos LCD pueden presentar datos de cualquier tipo con un consumo de potencia mínimo. Por otro lado, un modulo LCD permite minimizar el uso de pines del microcontrolador pues solo se requiere de unas pocas líneas de datos y de control.

Los módulos LCD se encuentran en diferentes presentaciones, por ejemplo 2X16, 2X20, 4X20, 4X40, etc. La forma de utilizarlos y sus interfaces son similares, por eso los conceptos generales se pueden aplicar a cualquier display. Para el sistema de desarrollo se ha escogido un display de 2X16, pues es de costo bajo y es suficiente para las practicas planteadas.

Estos dispositivos tienen distintas configuraciones de sus pines pero sus funciones no cambian, poseen 14 pines numerados de 1 a 14, pueden estar en una sola línea o dos columnas como se muestra en la Figura 3.7 a continuación.

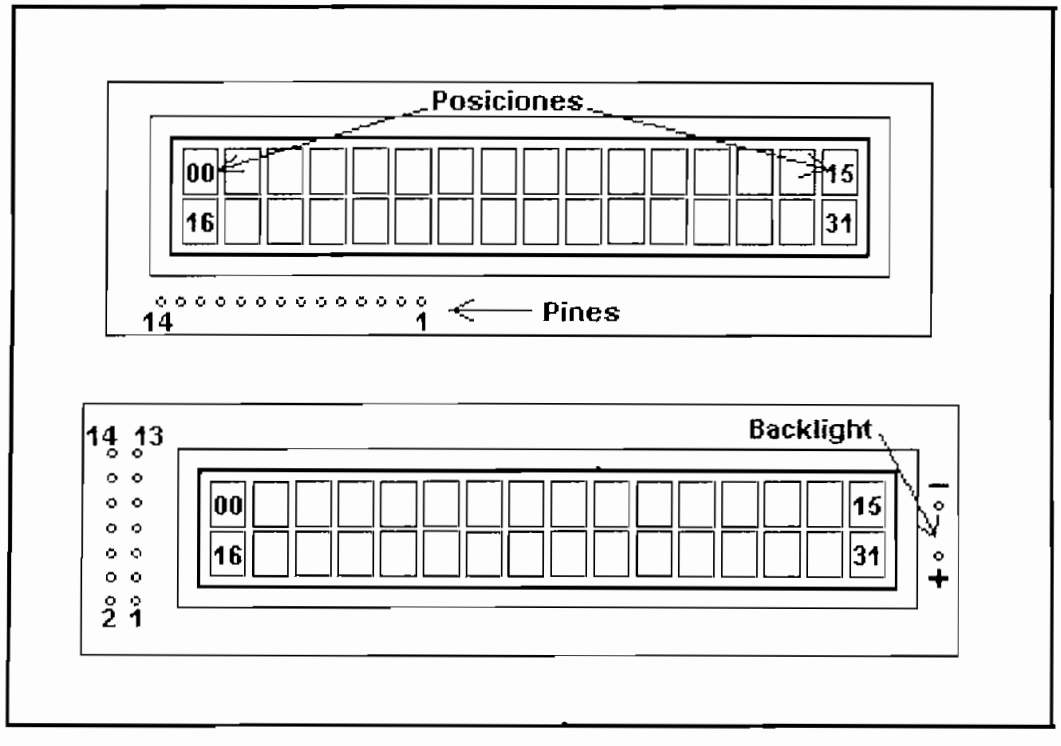


Figura 3.7 Configuración de pines de los módulos LCD

Algunos módulos LCDs para mejorar su visualización poseen luz posterior o "backlight", esta se maneja a través de dos pines que normalmente se conectan

como un diodo. Para mantener la temperatura baja del sistema debido a la luz posterior se debe manejar la alimentación del backlight en forma intermitente a una frecuencia de 60Hz o también se puede limitar mediante una resistencia de 56Ω, 1/2w para alimentar el terminal positivo del backlight.

Los pines de conexión de estos módulos incluyen un bus de datos de 8 bits, un pin de habilitación (E), un pin de selección, que indica que el dato es una instrucción o un carácter del mensaje (RS) y un pin que indica si se va a escribir o a leer en el módulo LCD (R/W). En la Tabla 3.2 se muestra la función de estos pines.

Terminal	Símbolo	Nombre y Función
1	Vss	Tierra, 0v.
2	Vdd	Alimentación +5v.
3	Vo	Ajuste de voltaje de contraste
4	RS	Selección de Dato/ Control
5	R/W	Lectura/ escritura en LCD
6	E	Habilitación
7	D0	D0 Bit menos significativo
8	D1	D1
9	D2	D2
10	D3	D3
11	D4	D4
12	D5	D5
13	D6	D6
14	D7	D7 Bit mas significativo

Tabla 3.2 Funciones de los pines en el módulo LCD

El circuito de conexión del módulo LCD es muy sencillo, y solamente consiste en la conexión de la alimentación en los pines1 y 2; la conexión a un potenciómetro de 5KΩ para ajustar el contraste (Vo) en el pin 3 y la conexión del backlight según la Figura 3.8 a continuación.

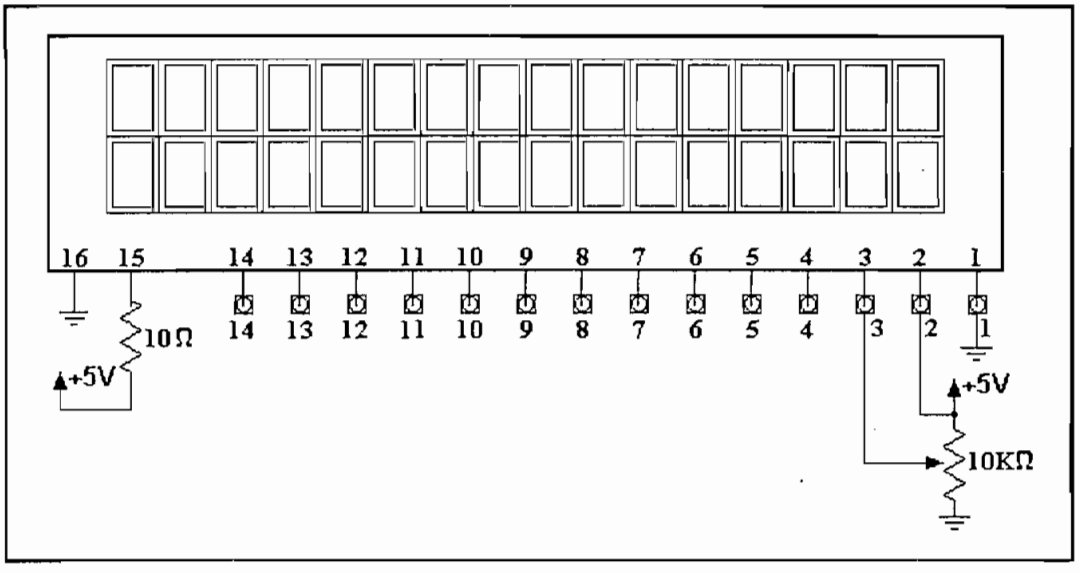


Figura 3.8 Circuito del módulo LCD.

3.3.6 MÓDULO DE DIPSWITCH

Este módulo consta de un dipswitch de 8 elementos, estos interruptores permitirán realizar practicas de lectura de puertos; están identificados desde S1 a S8. Cuando un interruptor está en la posición de OFF; es decir, abierto, el terminal de salida tendrá un nivel lógico alto (+5V) y cuando el interruptor esté en la posición de ON; es decir, cerrado, la salida tendrá un nivel lógico bajo. El circuito puede ser observado en la Figura 3.9 a continuación.

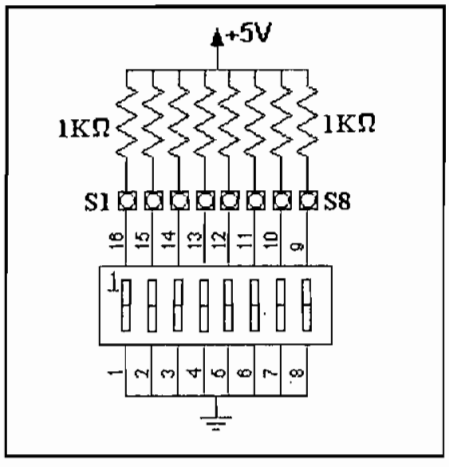


Figura 3.9 Módulo de dipswitch

3.3.7 MÓDULO DE TECLADO MATRICIAL

Este teclado tiene una matriz de 4X4 teclas, con los caracteres en cada tecla del 0 al 9, de A a la D incluidas las teclas * y #. Para las prácticas se puede alterar esta asignación para trabajar en hexadecimal de 0 a F. En el terminal de conexiones del mismo se encuentran marcadas las filas como F0, F4, F8, FC. Las columnas se denominan C0, C1, C2 y C3. La denominación de las filas y las columnas corresponde al primer carácter, ya sea de la fila ó de la columna. Las columnas del teclado están conectadas a través de una resistencia pull-up a +5v. que fija a nivel lógico alto cuando estas no están presionadas. Este teclado es leído por métodos de multiplexaje mediante el cual se analiza cada columna ingresando un nivel lógico bajo en el terminal C de la columna correspondiente y analizando las salidas de las filas F0 a FC para ver si alguna de las teclas están presionadas; esto ya que al presionar una tecla esta conecta el terminal F correspondiente a un nivel lógico bajo. Este proceso se repite en las cuatro columnas secuencialmente.

El circuito a implementar se muestra en la Figura 3.10 mostrado a continuación.

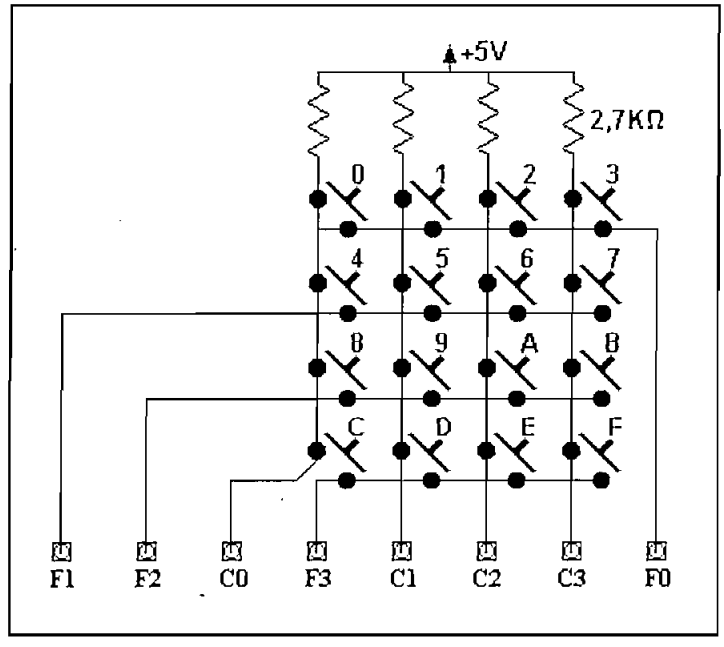


Figura 3.10 Teclado matricial

3.3.8 MÓDULO DE PULSADORES

El sistema incluye dos pulsadores normalmente abiertos, los cuales son también de suma importancia para la realización de prácticas en donde se ejercita la lectura de puertos y se requiera de comandos dados por el usuario. Sus terminales tienen los nombres de S9 y S10 para continuar con la nomenclatura de los dipswitch. En estado de reposo, la lectura de estos elementos tendrá un nivel lógico alto (+5V), mientras que cuando se presionan pasarán a un estado lógico bajo (0V). El circuito se muestra en la Figura 3.11 a continuación.

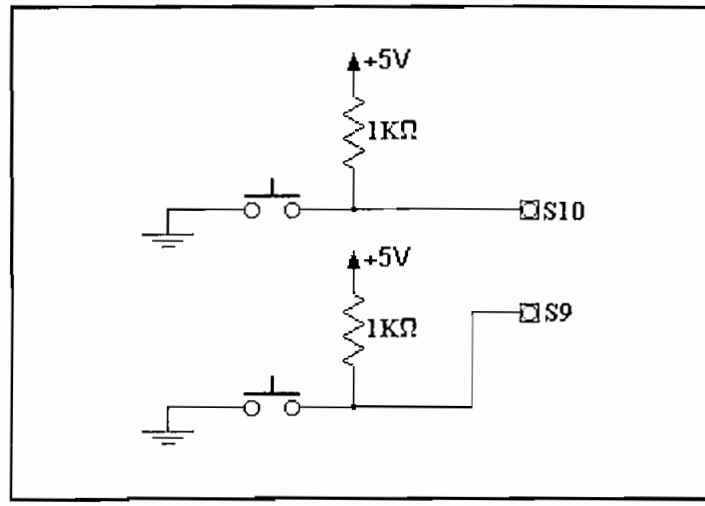


Fig. 3.11 Circuito de pulsadores

3.3.9 MÓDULO DE COMUNICACIONES SERIALES RS232

Uno de los módulos más importantes es la interface serial RS-232 pues es una de la más utilizadas, debido a que toda computadora personal tipo PC tiene al menos uno de estos puertos, en el cual se pueden conectar diferentes clases de dispositivos como mouse, impresoras seriales, etc., y también se lo puede usar para realizar adquisición de datos mediante la conexión con un microcontrolador.

La interface serial permite establecer comunicación entre equipos que estén separados hasta una distancia máxima de 15m (según la norma), pero en la práctica se puede obtener hasta 30 metros. Para implementarla solo se requiere un circuito integrado MAX232 y cinco condensadores de 10uF (tantalio). Los pines

de comunicación se denominan Tx y Rx. El circuito se muestra en la Figura 3.12 a continuación.

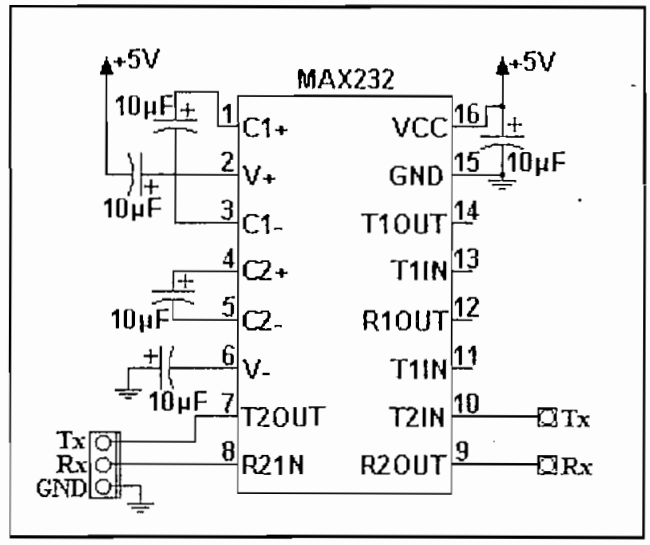


Figura 3.12 Circuito para la interface RS-232

3.3.10 MÓDULO DE COMUNICACIONES SERIALES RS-422/485

Las interfaces seriales RS-422/485 se diseñaron para la conexión física entre computadores y terminales directamente. Estos estándares tienen grandes ventajas con respecto al sistema RS-232 entre ellas se encuentran: la distancia de comunicación se puede extender a 1200 metros, la velocidad puede llegar a 10Mbits por segundo y el número de elementos, llamados también nodos, crece sustancialmente; para el caso de la interface RS-422 pueden estar conectados un transmisor y hasta 10 receptores en un sistema full-duplex, mientras que para la RS-485 pueden estar conectados simultáneamente hasta 32 transmisores/receptores en un sistema half-duplex. Otra ventaja es que este sistema no requiere fuentes duales como en el sistema RS-232, sino que funciona con la fuente de alimentación de 5 voltios, simplificando el diseño.

Ambas utilizan el método de medida diferencial, en la que utiliza dos líneas para la transmisión y dos para la recepción, en cada par de conductores la segunda línea tiene un nivel de voltaje complementario al del primero, y el receptor responde a la diferencia de voltajes entre los dos conductores. Este tipo de líneas de transmisión se llaman balanceadas, permitiendo la eliminación de

ruidos electrostáticos y electromagnéticos común a las dos líneas que se utilizan. El integrado 75176 contiene un transmisor y un receptor y solo dos líneas diferenciales A y B de entrada salida de datos; dos líneas adicionales RE y DE determinan la función que cumple el integrado, permitiendo o inhibiendo la recepción o la transmisión de datos. Como en este integrado el transmisor es habilitado por un estado lógico alto y el receptor por un estado lógico bajo, se puede utilizar la misma señal para controlar la función que cumple el integrado en un momento determinado mediante el terminal llamado control; por lo tanto solo una de las funciones estará habilitada el mismo tiempo. Por lo tanto este circuito está diseñado para permitir una transmisión half-duplex trabajando en la norma RS-485.

El sistema RS-422 establece una comunicación full-duplex para lo cual se requiere 2 líneas adicionales. Esto se consigue con otro integrado 75176 exclusivo para la transmisión mientras que el primero se lo configura como recepción conectando el terminal Control a un estado lógico bajo. De esta forma quedan operando los dos circuitos integrados 75176.

El circuito se puede observar en la Figura 3.13 en el cual como entrada se tienen los terminales Rx, Tx y control que se pueden conectar a los pines del microcontrolador; existen microcontroladores como el PIC16F873 que incorpora un módulo USART (Receptor y transmisor Síncrono y Asíncrono Universal) que simplifican totalmente el proceso de transmisión y recepción de datos. Como salidas se tienen los terminales T-, T+, RT-, RT+ y tierra, a estos se conectan los cables de transmisión de tipo par trenzado (twisted-Pair), el cual consiste de dos conductores aislados que están retorcidos entre si consiguiendo una buena inmunidad al ruido, se mejora con un cable redondo que rodee al par. Hay que tomar en cuenta la necesidad de colocar una resistencia de 120Ω en la entrada de cada receptor ya que las líneas de transmisión deben tener una carga igual a la impedancia característica. El cable par trenzado AWG24 tiene una impedancia característica cercana a los 120 Ω.

Otro aspecto importante es que todos los receptores deben tener a la salida un estado lógico alto, cuando no hay un transmisor activo o éste está desconectado, ya que si cualquiera de la dos condiciones ocurre, el nivel de

entrada que ve el receptor puede ser indefinido, lo cual puede inducir a errores en el proceso de comunicación. Recordar además que las líneas de comunicación serial de los microcontroladores permanecen normalmente en un nivel de voltaje alto, y solo bajan cuando están transmitiendo o recibiendo datos. Esta es la razón por la cual se deben agregar un par de resistencias idénticas a cada una de las líneas de entrada de los receptores.

La línea A tiene una resistencia pull-up y la línea B tiene una resistencia pull-down. En este caso las resistencias de polarización aseguran que el terminal A (+) sea al menos 0.48 voltios más positivo que el terminal B (-); la salida de cada receptor mantendrá un estado lógico alto.

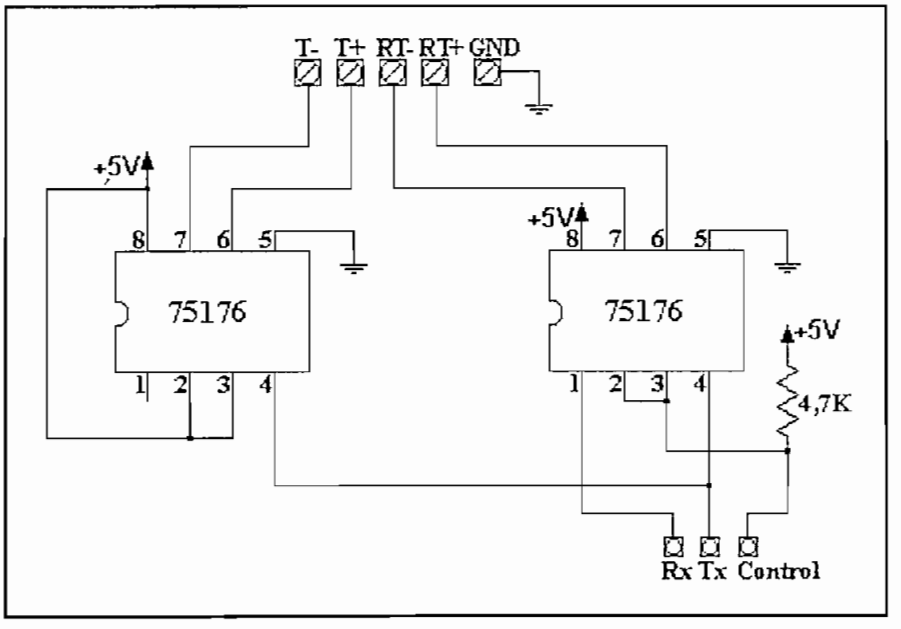


Figura 3.13 Circuito de interface RS-422/485

3.3.11 MÓDULO DE MEMORIA SERIAL

En la mayoría de sistemas de control se necesita almacenar información para su uso posterior, un avance importante en este campo es la incorporación de las memorias EEPROM seriales que utiliza la interface I^2C. Las cuales presentan grandes ventajas si se comparan con otras posibilidades. Entre sus características importantes se destacan:

- Se pueden conectar fácilmente a microcontroladores y microprocesadores, incluso existen microcontroladores que tienen puertos I^2C, simplificando totalmente la comunicación.
- Transferencia de datos de manera serial, lo que permite ahorrar pines del microcontrolador para dedicarlos a otras funciones.
- Este elemento tiene inmunidad al ruido, dado que estos elementos tienen filtros en los pines de comunicación.
- Ocupa un espacio diez veces menor que las memorias que trabajan en paralelo, esto permite ahorrar dinero debido al menor tamaño del circuito impreso.
- El consumo de corriente es mucho menor que en las memorias que trabajan en paralelo, esto las hace ideales para sistemas portátiles que funcionan con baterías.

Aquí se utiliza la memoria serial 24LC04B/P perteneciente al grupo de memorias 24XX. Estas memorias utilizan el bus de 2 hilos para comunicarse con otros dispositivos. Dado que cumplen con el protocolo I^2C, tienen un pin llamado SCL que recibe los pulsos generados por el dispositivo maestro en este caso del microcontrolador y otro llamado SDA que maneja el flujo de datos de forma bidireccional (entrada/salida). Los pines correspondientes a esta memoria se pueden observar en la Figura 3.14 indicada a continuación.

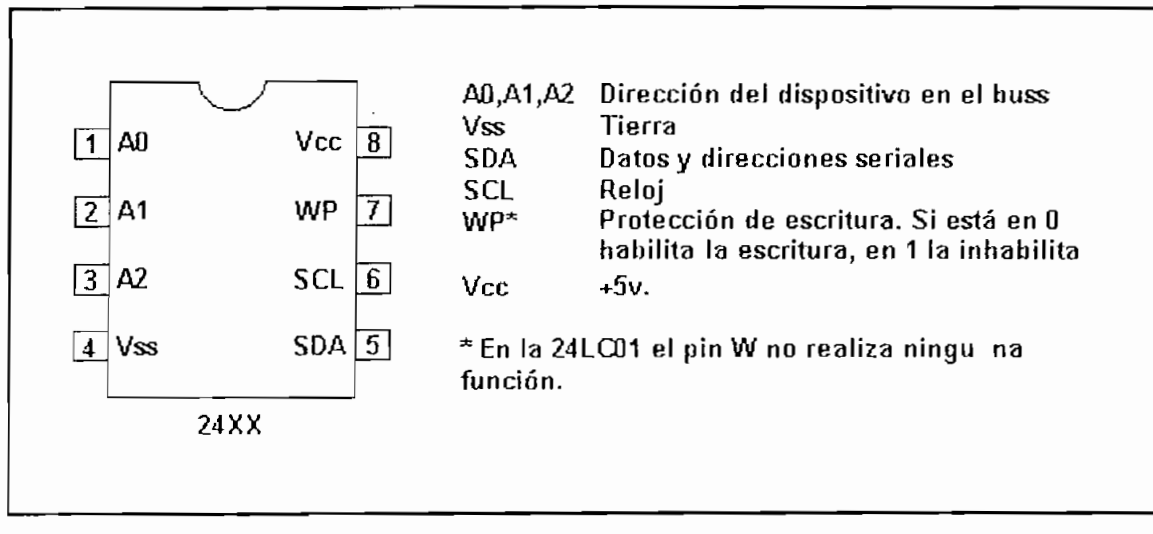


Figura 3.14 Configuración de pines de la memoria 24LCXX

Este dispositivo no requiere de un pin habilitador chip select, ya que en este esquema la transferencia de información solo se puede iniciar cuando el bus esté libre. En este caso, como cada dispositivo tiene su dirección determinada mediante los pines A0, A1 yA2; solamente responderá la memoria cuya dirección coincida con la dirección que va encabezando la trama de información. En la Tabla 3.3 se indica la capacidad de almacenamiento de estos dispositivos y las posibilidades de direccionamiento que tienen.

Referencia	Capacidad en Kbits	Bloques internos	A0	A1	A2	Dispositivos en el bus
24LC01B, 24C01	1	1	1 ó 0	1 ó 0	1 ó 0	8
24LC02B, 24C02	2	1	1 ó 0	1 ó 0	1 ó 0	8
24LC04B, 24C04	4	2	X	1 ó 0	1 ó 0	4
24LC08B	8	4	X	X	1 ó 0	2
24LC16B	16	8	X	X	X	1

Tabla 3.3 Capacidad de memoria y direccionamiento de la memorias 24LCXX

En la Figura 3.15 se muestra el diagrama esquemático del circuito, en este se conecta en el pin WP un switch que permite colocar manualmente en el modo de lectura o lectura/escritura. En este caso los pines de dirección de la memoria se conectan a tierra, estableciendo A0, A1 y A2 como cero lógico, esto se utilizará para transmitir la dirección de la memoria. La resistencia de 4,7KΩ conectada al pin SDA es necesaria dado que dicho pin tiene salida de colector abierto. Los terminales SCL y SDA están a disposición en dos terminales para conectar al microcontrolador.

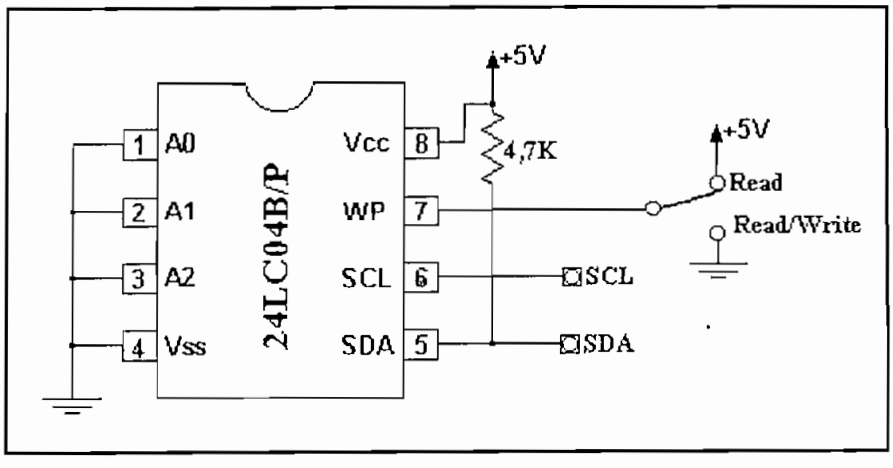


Figura 3.15 Conexión de la memoria serial

3.3.12 MÓDULO DE ENTRADA DE SEÑALES ANALÓGICAS

Una de las características más importantes de los microcontroladores es la incorporación de conversores análogo digitales, simplificando de esta manera toda la circuitería externa que era necesaria para manejar las señales análogas. Por ejemplo el microcontroaldor PIC16C71 posee un conversor A/D de cuatro canales de entrada; para utilizarlo el sistema de desarrollo incluye un potenciómetro conectado entre 0 y 5 voltios que permite simular el ingreso de una señal analógica de sensor en su cursor A1, que puede ser conectada directamente al microcontrolador. Además se incluye un sensor de temperatura LM35, el cual presenta una variación de 10mV/°C y puede trabajar en un rango de –55°C a 150°C, la fuente de alimentación positiva puede estar entre 4 y 30 voltios, siendo su precisión de 0.5°C. La máxima lectura del conversor análogo a digital se obtiene cuando la entrada analógica es igual a 5voltios; de tal manera, que si se hace la división de 5 voltios entre 255 (FFH), se obtiene una relación de 20 mV por unidad. Esto causará que si, por ejemplo, se mide una temperatura de 10°C, en la que el sensor dará 100mV, entonces el conversor entregará (100mV/ 20mV) una lectura de 5, siendo este valor la mitad de la temperatura medida. En este caso una opción es multiplicar internamente en el microcontrolador este valor por dos, o para que sea una lectura directa, se puede multiplicar externamente este valor por dos mediante un circuito amplificador no inversor en cascada de ganancia dos. El sensor se lo conecta mediante un conector de tres terminales para así tener una sonda y medir temperaturas externas al sistema.

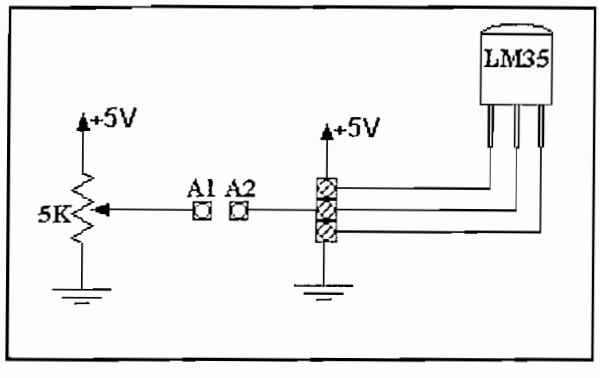


Figura 3.16 Circuito de entradas análogas

3.3.13 MODULO DE POTENCIÓMETRO DIGITAL

Los potenciómetros digitales permiten realizar el control de sistemas en donde se involucre la variación de una resistencia, como control digital de volumen, convertidor digital a análogo, control digital de una fuente, etc. Sus principales ventajas frente a uno mecánico son: mayor precisión, mayor velocidad, menor tamaño, no existe desgaste.

El poteciómetro a utilizar en el sistema de desarrollo es el X9C104/P con un valor total de 100KΩ de la marca Xicor (Figura 3.17), también se consiguen en valores de 10KΩ (X9C103) y 50KΩ (X9503). Este es un potenciómetro de estado sólido no volátil que contiene un arreglo de 99 resistencias y cuyas uniones son accesibles por el cursor o pin central del potenciómetro. La posición del cursor es controlada por los pines CS, INC, y U/D; una de las características más importantes es que posee una memoria no volátil tipo EEPROM que conserva el valor de la resistencia que debe mantener entre sus terminales; esto es de gran utilidad cuando se utilizan sistemas que no utilizan baterías de respaldo.

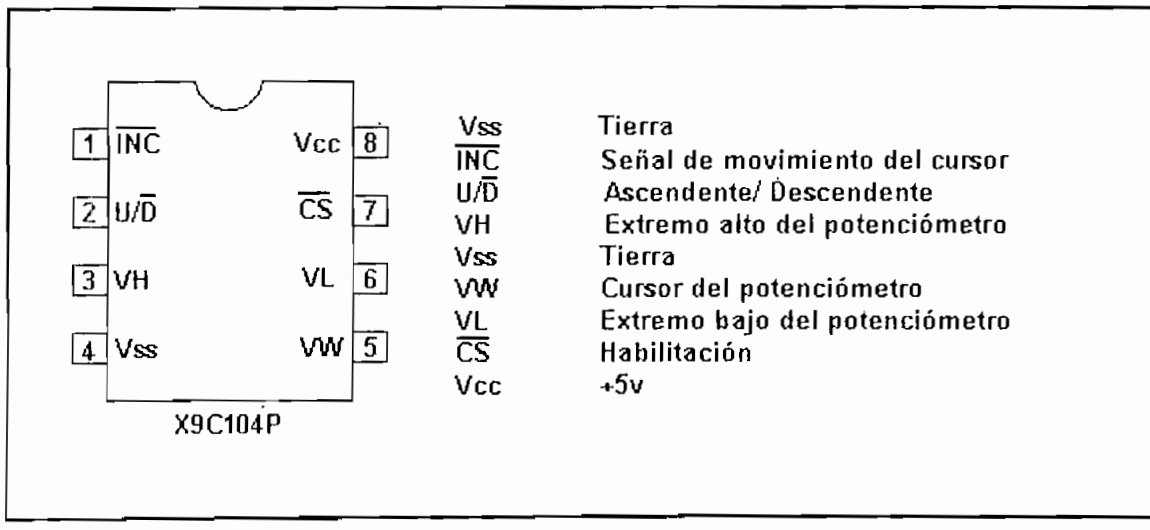


Figura 3.17 Diagrama de pines del potenciómetro digital

El integrado es alimentado con +5V y los extremos del potenciómetro VH y VL se pueden conectar entre ±5V. Las entradas INC, U/D y CS controlan el movimiento del cursor a lo largo del arreglo de resistencias. El pin CS habilita el integrado con un nivel bajo, un flanco de bajada en el pin INC hará que se

incremente o decremente el contador de 7 bits dependiendo de la polaridad del pin U/D. La salida de ese contador es decodificada para seleccionar una de las cien posiciones dentro del arreglo de resistencias.

En la Figura 3.18 se puede observar el diagrama de bloques del potenciómetro digital en la que se distinguen tres bloques: uno de control, contador y decodificador; otro de memoria EEPROM y otro conformado por el arreglo de resistencias de salida. El bloque de control funciona como un contador ascendente descendente (Up/Down); la salida del control se maneja a través de un decodificador de 1/100 seleccionando así una sola salida del cursor.

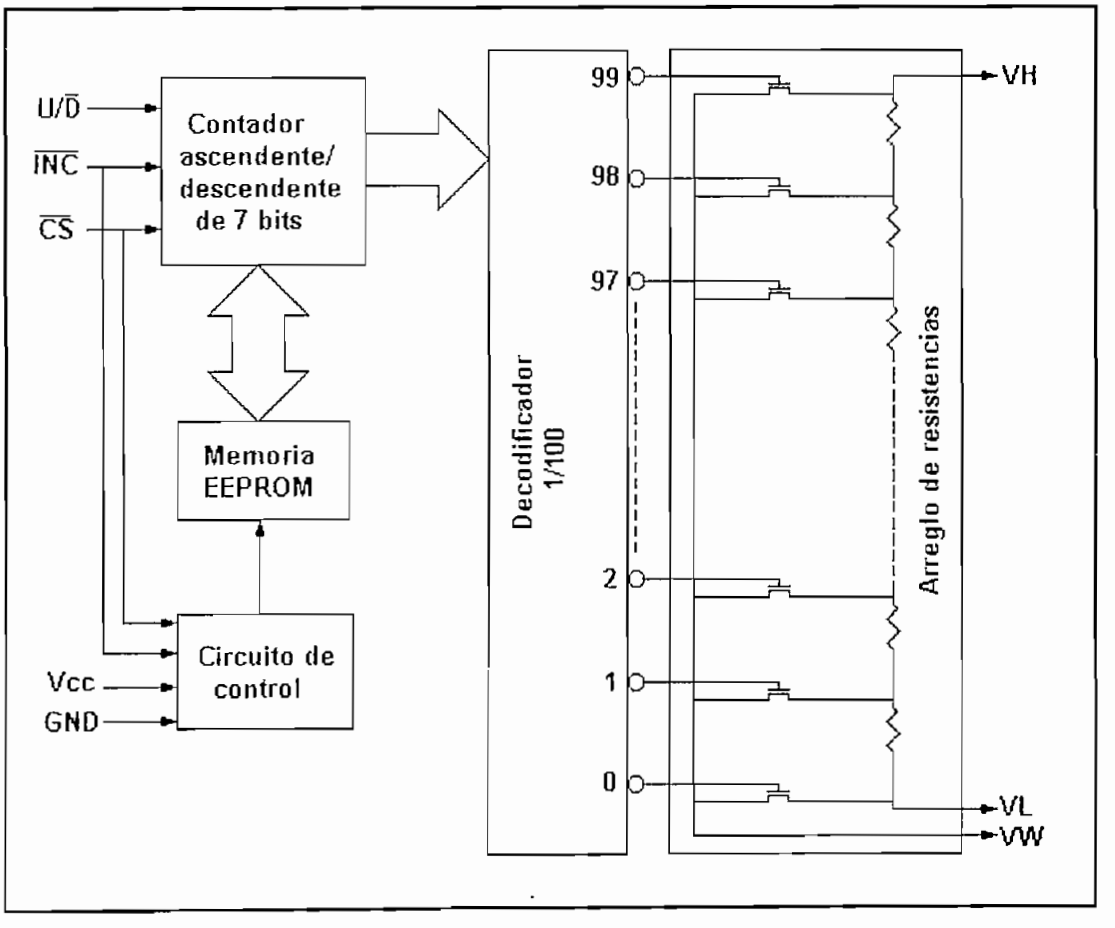


Figura 3.18 Diagrama de bloques del potenciómetro digital

En la Figura 3.19 a continuación se puede observar el circuito del potenciómetro en el cual se conectan los pines de alimentación VL y VH a tierra y 5V, respectivamente y el pin CS a 5v a través de una resistencia de 2,7KΩ, esto se hace para asegurar que la memoria EEPROM no altere su contenido con el ruido

que se genera en el momento del encendido. Entonces se tiene para las señales de entrada a este circuito los terminales CS, U/D e INC; y como salida los terminales Vw y GND.

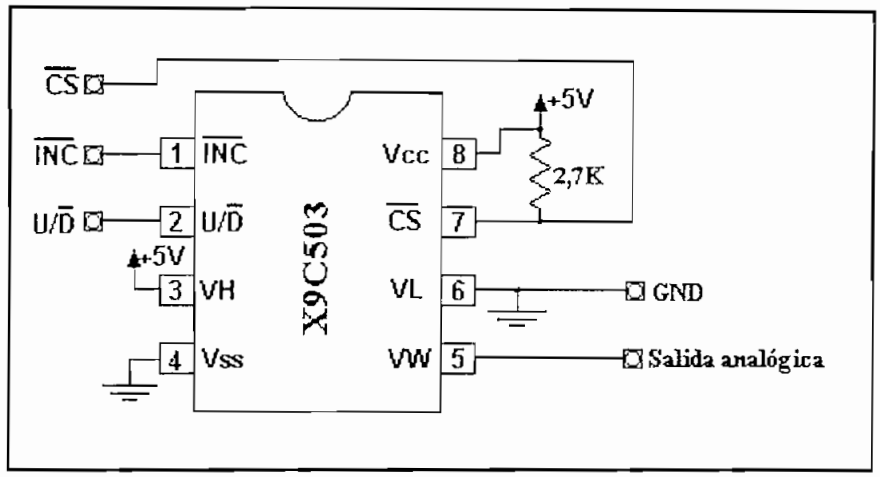


Figura 3.19 Circuito de conexiones del potenciómetro digital

3.3.14 MÓDULO ZUMBADOR PIEZOELÉCTRICO

Los zumbadores son muy utilizados en los sistemas de control como indicadores sonoros, debido a las ventajas que ofrecen sobre los parlantes de bobina móvil. Entre éstas se tiene su bajo consumo de potencia, un volumen reducido, bajo peso, fácil construcción y buena potencia de salida para la mayoría de aplicaciones. Existen dos tipos: los activos y los pasivos. En el primer caso, basta con aplicar un voltaje DC para que emitan sonido, y en el segundo caso, que es el que se incluirá en el sistema de desarrollo, se necesita una señal oscilante.

Para poder realizar la emisión de sonidos de diferentes frecuencias se ha incorporado un zumbador sin circuito de oscilación, el cual puede ser conectado directamente al microcontrolador sin necesidad de un circuito amplificador, ya que este consume una corriente muy pequeña de alrededor de 3mA.

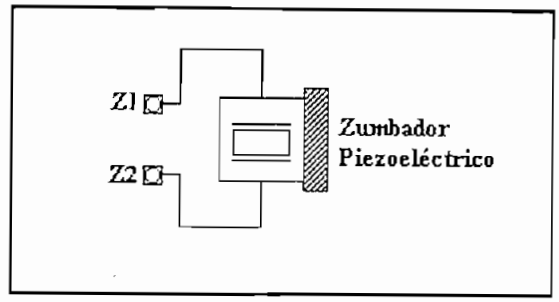


Figura 3.20 Zumbador piezoeléctrico

3.3.15 MÓDULO DE SALIDA PWM

Los microcontroladores poseen módulos de salida PWM, estos sistemas se los utiliza para realizar controles de motores, generación de señales analógicas, etc. Estos módulos generan automáticamente estas señales que entregan en algún pin del micro llamado justamente PWM.

En el sistema se han incorporado dos circuitos; el primero puede controlar la velocidad de un motor DC pequeño y en este caso la salida PWM se amplifica mediante un transistor NPN, también se incluye un diodo de protección del transistor a la salida de circuito, el cual se puede observar en la Figura 3.21 a continuación.

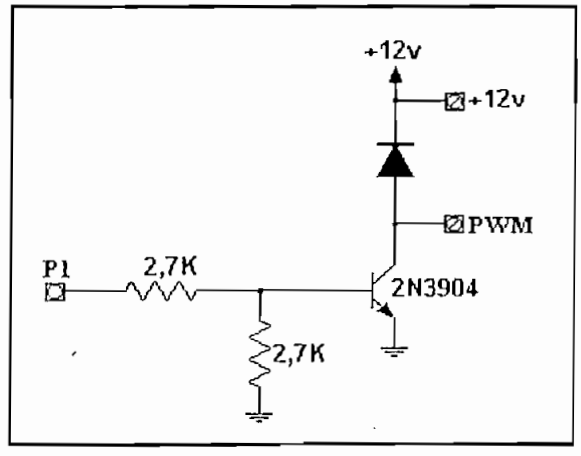


Figura 3.21 Circuito de control de velocidad de un pequeño motor DC

El segundo circuito permite implementar un convertidor digital a analógico mediante un circuito integrador que convierte la salida PWM en un valor DC. El circuito se puede observar en la Figura 3.22 mostrada a continuación.

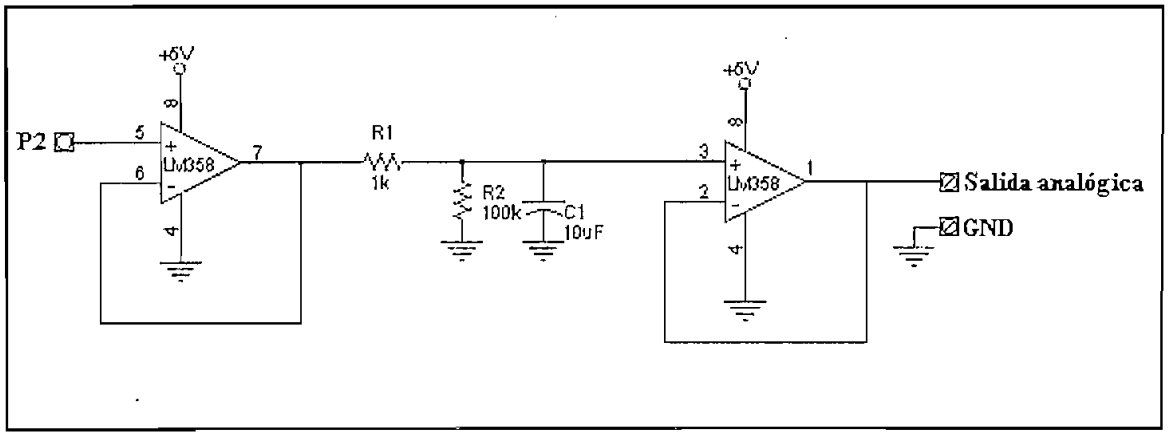


Fig. 3.22 Convertidor de digital a análogo PWM

3.3.16 MÓDULO DE RELOJ DE TIEMPO REAL

Este módulo permite la experimentación con el reloj calendario de tiempo real DS1307, el mismo que cumple la función de conteo de tiempo, en segundos, minutos, horas, día, meses y años, además realiza la corrección de años bisiestos. Sin duda este módulo tiene muchas aplicaciones, por esto se tiene desarrollada una práctica de ejemplo de su manejo, y en esta se hace la descripción completa de la estructura de este integrado.

Para realizar la conexión de este integrado se requieren unos pocos elementos adicionales como son: Una batería de litio de 3V como respaldo en caso de perdida de energía, esta se conecta entre el pin 3 y tierra. Un cristal de cuarzo de 32.768Hz entre los pines 1 y 2. La salida SDA en el pin 5, por ser de drenaje abierto, se requiere la conexión de una resistencia externa pull-up que puede ser de 4,7KΩ. Además para visualizar la salida de onda cuadrada (SQW/OUT) que tiene el integrado en el pin 7, se ha implementado un circuito de activación de un diodo led, a través de un transistor 2N3906, ya que es una salida de colector abierto. Por último se alimenta al circuito con 5V entre los pines 8 y 4.

Hay que notar que el integrado incluye en sus circuitos internos la conmutación de la fuente a la batería, en caso de que el nivel de la fuente Vcc caiga por debajo de 1,25 VBAT.

El circuito implementado se muestra en la Figura 3.23 a continuación.

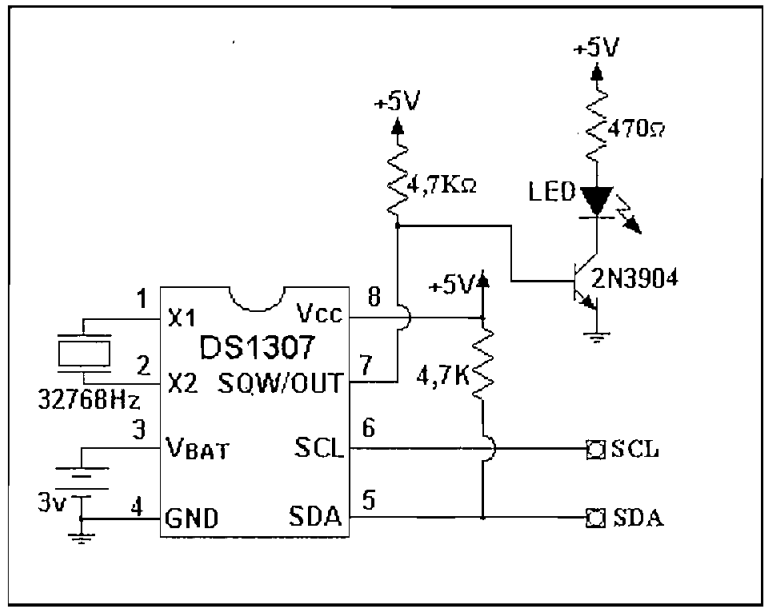


Figura 3.23 Circuito del reloj calendario I^2C

3.3.17 MÓDULO DE RELÉ

Para el manejo de cargas de potencia desde el microcontrolador, se puede emplear un relé. Este se opera a través de un transistor NPN trabajando en corte – saturación, como se muestra en la Figura 3.24; se puede observar que se ha implementado también el diodo de protección en paralelo al transistor, este circuito es activado colocando el terminal K1 a un nivel lógico alto. El cálculo de la resistencia limitadora, se lo realiza a continuación tomando en cuenta que el relé tiene una resistencia de 371Ω.

$$Ic = \frac{V}{R} = \frac{12 - 0.2}{371}$$

$$Ic = 31{,}8mA$$

$$Ib = \frac{Ic}{hfe} = \frac{31.8}{200} = 0.159mA$$

$$Ibsat = 10 * Ib = 1.59mA$$

$$Rb = \frac{V}{Ibsat} = \frac{5 - 0.7}{1.59}$$

$$Rb = 2.7K\Omega$$

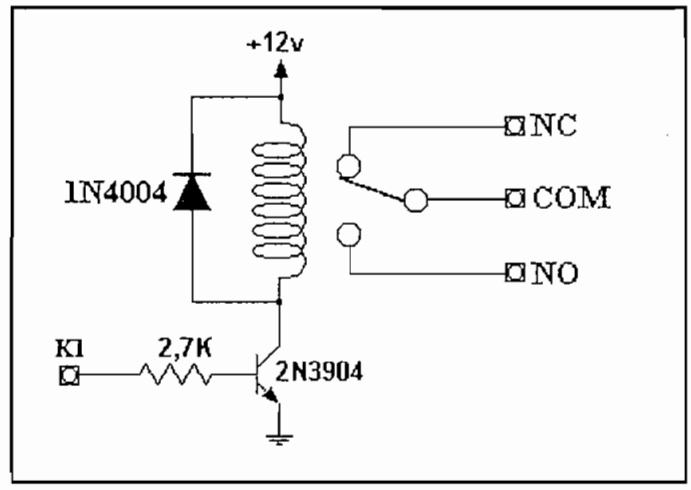


Figura 3.24 Circuito de relé

3.3.18 MÓDULO DE PROGRAMADOR EN CIRCUITO

La mayor parte de modelos de microcontroladores PIC tienen la característica de poder ser programados una vez que están montados en el circuito de aplicación final, esto se lo realiza muy fácilmente a través de programación serial, mediante dos líneas para las señales de reloj y datos; y otras tres adicionales de alimentación (Vcc=+5v), voltaje de programación (Vpp=13.4v) y tierra. Este sistema es utilizado para realizar la programación del microcontrolador con la versión última de firmware, o también para reprogramar el micro con una versión actualizada, sin tener que desmontarlo del circuito armado. Esta capacidad de programar los microcontroladores en circuito se la utiliza también en el sistema de desarrollo para facilitar los procesos de programación al realizar modificaciones en la prueba de programas; así se dispone de un sistema de desarrollo completo; esto también evitará daños por manipulación de los microcontroladores pues no será necesario sacarlos del circuito armado para cargar el programa con un programador externo. El microcontrolador entra al modo de programación/verificación manteniendo en estado bajo (0V) los pines RB6 y RB7 mientras se pasa el pin MCLR del nivel bajo (0V) al nivel alto (13,4V). Luego de esto se envían las señales de reloj y datos por los pines RB6 y RB7 respectivamente, mientras se aplica el voltaje de programación, todas estas señales son generadas por un software instalado en el computador. Este programa no es tema de este proyecto, sin embargo se lo *puede* conseguir

fácilmente. Uno de estos programas es el EPICWIN programer de microEngineering Labs que se incluyen en el libro Curso Avanzado de Microcontroladores PIC de Edison Duque C.

Para esto se ha implementado en el sistema una fuente adicional de 13,4V mediante un regulador de 12V cuya referencia se ha subido 1,4V mediante dos diodos rectificadores. En el pin MLCR/Vpp de los micros deben llegar las señales de reset y de voltaje de programación, para esto se ha diseñado un circuito que permite integrar las dos señales, aislando una de otra mediante diodos rectificadores rápidos ya que los tiempos de programación son extremadamente pequeños, evitando así el uso de un switch para conectar a una de las dos señales. El voltaje de programación es aplicado al pin MCLR / Vpp, esta señal la debe entregar el computador mediante el pin # 5 del conector DB25 del puerto paralelo. Esta señal ingresa al programador por el pin #5 del conector DB9 del sistema y pasa por el buffer ECG7407 que puede manejar niveles de voltaje de hasta 30V, esta señal activa al transistor PNP 2N3906 el cual conecta su colector a 13,4V polarizando directamente al diodo rápido 1N4148, de esta manera llega el voltaje de programación al pin MCLR/Vpp del microcontrolador.

Hay que notar que el voltaje de programación no interfiere con el voltaje Vcc de 5V, al que normalmente se encuentra conectado el pin MCLR del microcontrolador, gracias al segundo diodo rápido 1N4148 en serie con la resistencia pull-up de 4,7KΩ ya que esta se polariza inversamente entre 13,4V y 5V el momento de activarse el transistor 2N3906. Este circuito se une al del reset que, mediante un pulsante, conecta a tierra el pin MCLR, para evitar un cortocircuito de la fuente de 13,4v. en el caso eventual de que se resetee el sistema, al mismo tiempo de activarse la señal de programación. Se han colocado dos resistencias de 100Ω una en serie con el pulsante y otra en el colector del transistor que limitan la corriente en este caso alrededor de 62mA. Además se ha colocado un diodo led que se enciende en el caso de cerrarse el transistor 2N3906 el cual indica la presencia del voltaje de programación. Las señales de reloj y datos deben ingresar a los pines RB6 y RB7, respectivamente, de los microcontroladores de 18, 28, y 40 pines, mientras que para los micros de 8 pines se ingresa por los pines GP1 y GP0. Para evitar interferencia en estos pines, debido a los periféricos conectados en estos, se ha conectado un switch doble de

dos posiciones que permite desconectar estos pines de los periféricos para conectarlos a las líneas de programación. La señal del reloj es enviada desde el computador por el pin #3 del puerto paralelo, ingresa al sistema por el pin #3 del conector DB9. Esta señal pasa por el buffer e ingresa directamente al pin RB6 o al GP1 de los microcontroladores. La señal de datos es enviada desde el computador por el pin #2 del puerto paralelo, e ingresa al sistema por el pin # 2 del conector DB25, esta señal ingresa a los microcontroladores por los pines RB7 y GP0; esta misma señal o los valores de lectura en los procesos de verificación regresan al computador a través de un buffer por el pin #1 del conector DB9 del sistema que conecta al pin # 10 del puerto paralelo del computador. Además de estos pines, la tierra del puerto paralelo del pin #22 se conecta a la tierra del sistema mediante el pin #9 del conector BD9; el resto de pines del puerto paralelo permanece sin conexión.

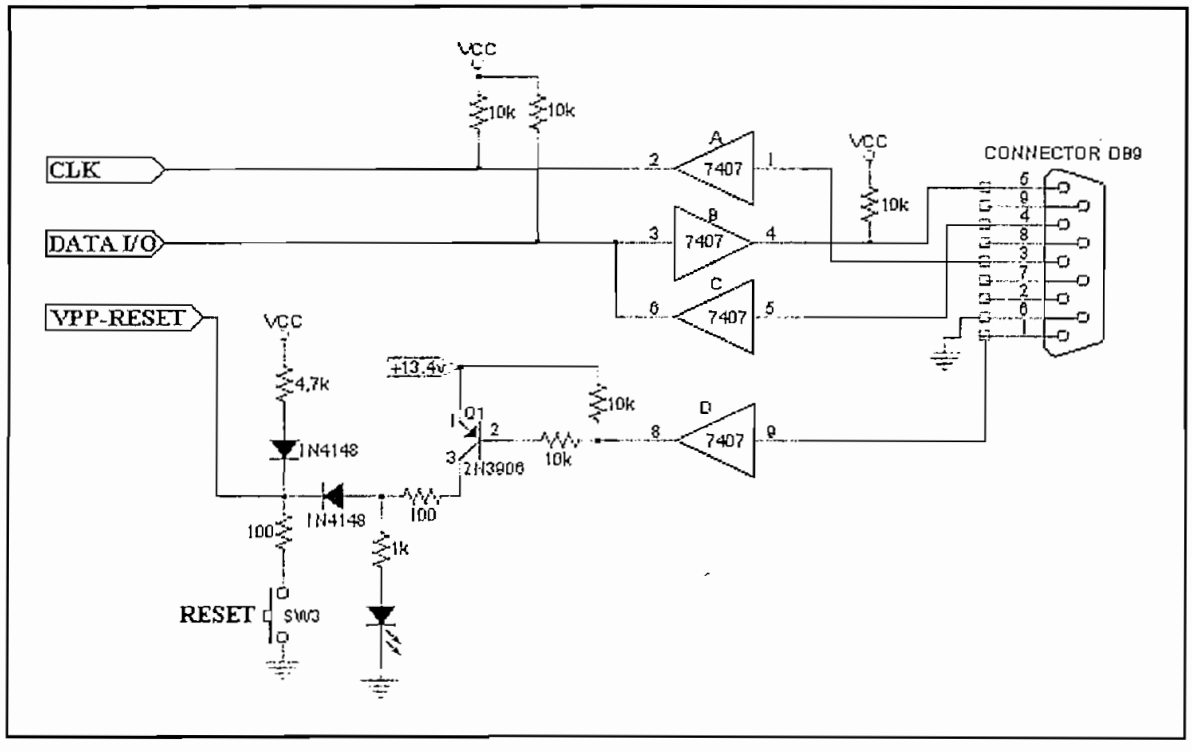


Figura 3.25 Esquema del programador en circuito

A continuación en el siguiente capítulo se van a desarrollar las practicas básicas que se pueden realizar con el sistema aquí diseñado.

CAPITULO IV

DISEÑO Y DESARROLLO DE PRÁCTICAS

4.1 Prácticas con el PIC16F84A.

La forma más efectiva de aprender el manejo de estos microcontroladores es con la realización de prácticas, utilizando para esto el sistema de desarrollo aquí planteado. De esta manera serán utilizados los componentes del sistema de desarrollo al realizar las distintas prácticas planteadas, cada una de las cuales tendrá un objetivo específico sobre el manejo de determinado módulo, o sobre la utilidad de alguna de las características especiales que poseen estos microcontroladores.

A continuación se plantean las prácticas a desarrollarse con el microcontrolador PIC16F84A:

- Manejo de leds, pulsantes y dipswitch
- Manejo de displays de 7 segmentos
- Multiplexaje de teclados y displays
- Manejo del módulo LCD
- Comunicación serial RS-232
- Comunicación I^2C con un reloj calendario

Es importante notar que además de las prácticas planteadas se puede realizar un sinnúmero de prácticas combinando los distintos módulos que incluye el sistema de desarrollo, e incluso se podrían desarrollar proyectos complejos como por ejemplo un control de accesos combinando el teclado, el display, memorias seriales, el reloj calendario y el relé.

4.1.1 MANEJO DE LEDS, PULSANTES Y DIPSWITCH

Práctica 1: Encendido y apagado de un led

Para iniciar de una manera simple y rápida con el manejo de los microcontroladores PIC, se realizará la práctica de control de un diodo led, con esto se analizarán los pasos para el desarrollo de cualquier proyecto. Esta práctica consiste en realizar el encendido y apagado secuencial, de un diodo led con un retardo aproximado de 1sg entre un estado y otro.

Para esto se utilizará el pin RA0 del microcontrolador, con el cristal de 4Mhz. El diodo led a encender será el número 1 del arreglo de leds disponible en el sistema de desarrollo. Debido al Buffer inversor que controla los diodos leds, estos se encienden con un estado alto. El circuito a implementar es el siguiente mostrado en la Figura 4.1.

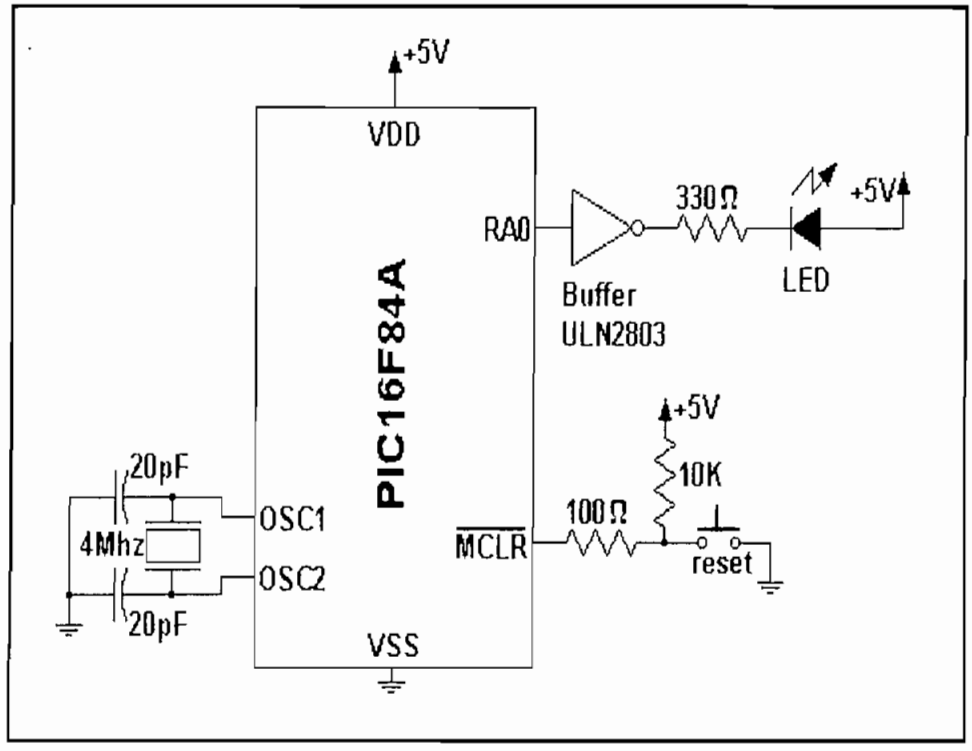


Figura 4.1 Circuito de encendido de un led

El diagrama de flujo correspondiente al ejercicio se muestra *a continuación* en la Figura 4.2.

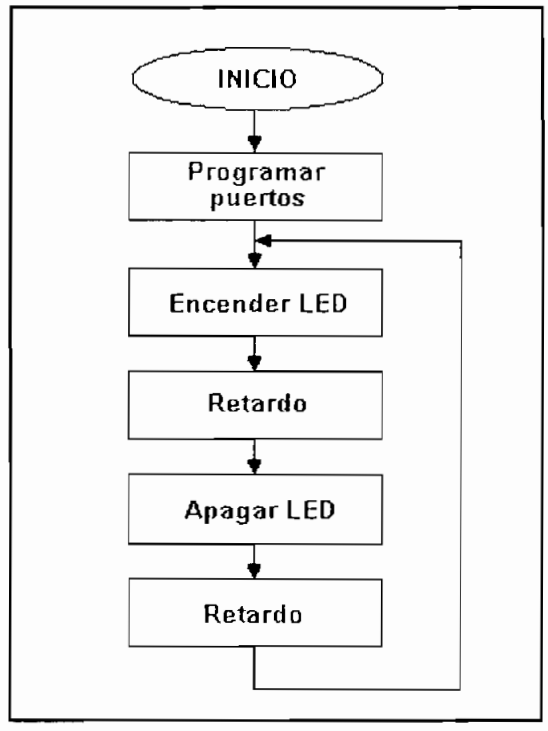


Figura 4.2 Diagrama de flujo del encendido y apagado de un led

En este ejemplo se pueden observar todas las 7 partes de la estructura de un programa, planteadas en el capítulo II. Esto se analiza en el archivo led.asm mostrado en la Figura 4.3; también se explicará el funcionamiento de cada uno de los bloques del diagrama de flujo de la Figura 4.2.

Antes de revisar los bloques del diagrama de flujo, se deben planteas las etiquetas a utilizar en este programa.

Las etiquetas a utilizar son las siguientes:

```
Status      equ   03h
porta       equ   05h
trisa       equ   85h
lazo1       equ   0ch
lazo2       equ   0dh
lazo3       equ   0eh
```

```
;******************Comentarios del objetivo del programa**********************

;Este programa realiza el encendido y apagado secuencial de un led
;con un retardo de 1sg de un estado a otro

;******************Indicación del modelo de microcontrolador******************

        list    p=16F84A        ;Microcontrolador utilizado

;************************Definición de etiquetas*******************************

status  equ     03h
porta   equ     05h
trisa   equ     85h
lazo1   equ     0ch
lazo2   equ     0dh
lazo3   equ     0eh

;**********Indicación de direcciones de escritura del programa*****************

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

;******************Subrutina de retardo de 1sg*********************************

retardo movlw   D'10'   ;El registro lazo1 contiene el número de
        movwf   lazo1   ;veces que repite 100 milisegundos
repite1 movlw   D'100'  ;El registro lazo2 contiene el número de
        movwf   lazo2   ;veces que repite 1 milisegundo
repite2 movlw   D'110'  ;El registro lazo3 contiene el número de
        movwf   lazo3   ;veces que repite los 9 microsegundos
repite3 nop             ;de retardo generados
        nop             ;por los 6 ciclos de las instrucciones nop (6usg)
        nop             ;más 1 ciclo de la instrucción decfsc (1usg)
        nop             ;más 2 ciclos del salto goto (2usg)
        nop             ;dando en total los 9usg, siendo esta la base
        nop             ;de tiempo, por lo tanto 1sg= 9usg*110*100*10
        decfsz  lazo3   ;decrementa el reg lazo3 y salta si llega a 0
        goto    repite3 ;si lazo3 no es 0 entonces salta a repite3
        decfsz  lazo2   ;decrementa el reg lazo2 y salta si llega a 0
        goto    repite2 ;si lazo2 no es 0 entonces salta a repite2
        decfsz  lazo1   ;decrementa el reg lazo1 y salta si llega a 0
        goto    repite1 ;si lazo1 no es 0 entonces salta a repite1
        retlw   0       ;salida de la subrutina cargando w con 0

;************************  Programa principal ********************************
;Programar puertos:
inicio  bsf     status,5        ;se ubica en el segundo banco de la RAM
        movlw   00h             ;se caraga el registro w con 00h
        movwf   trisa           ;se programa el puerto A como salidas
        bcf     status,5        ;se ubica en el primer banco de la RAM
;Encender el led:
otra    bsf     porta,0         ;coloca en 1 el pin RA0 para encender el led
;Retardo:
        call    retardo         ;Llama a la subrutina retardo de 1sg
;Apaga el led:
        bcf     porta,0         ;Coloca en 0 el pin RA0 para apagar el led
;Retardo:
        call    retardo         ;Llama a la subrutina retardo de 1 sg
;Retorno a repetir la secuencia:
        goto    otra            ;salta a otra para repetir la secuencia

;***************************fin del programa*************************************
        end
```

Figura 4.3 Programa de encendido y apagado secuencial de un led

Las etiquetas status, porta y trisa indican la dirección en la memoria RAM de estos registros que son utilizados para configurar y manejar el puerto A; los registros lazo1, lazo2 y lazo3 indican la dirección en la memoria RAM dentro de los registros de propósito general a utilizar para la realización de la subrutina de retardo.

Una vez planteadas las etiquetas, se debe desarrollar el bloque de programación de puertos, en este caso el puerto A se lo debe configurar como salida de datos, para esto se debe escribir ceros en el registro trisa que está ubicado en el segundo banco de la memoria RAM, entonces se cambia al segundo banco colocando un uno en el quinto bit del registro status, con la instrucción bsf status,5 luego se cargan ceros primero en el registro de trabajo W, con la instrucción movlw 00h para cargarlo en el registro trisa de configuración del puerto A mediante la instrucción movwf trisa.

El siguiente bloque a implementar es el que enciende el led, esto se lo realiza con la instrucción bsf porta,0 esta instrucción saca un uno lógico por el pin RA0 del puerto A haciendo que se encienda el diodo led.

El siguiente bloque es el retardo de 1sg. que se lo consigue llamando a la subrutina retardo, mediante la instrucción de llamado de subrutinas call retardo.

El bloque de apagado del led se lo realiza mediante la instrucción bcf porta,0 la cual cambia el estado del pin RA0 de uno a cero lógico, haciendo que se apague el diodo led.

El último bloque de retardo es idéntico al retardo anterior. Para finalizar el diagrama de flujo se debe regresar al punto de encendido del led para repetir el proceso, este sitio está etiquetado con la palabra otra, y esto se realiza con la instrucción goto otra.

La subrutina de retardo se realiza tomando en cuenta que el tiempo requerido para ejecutar una instrucción es de un ciclo de máquina en la mayoría de instrucciones y en los saltos toma dos ciclos, esto con un reloj de 4Mhz se traduce en 1usg o 2usg, mediante el decremento de registros cargados con un cierto valor, que se repite mediante lazos, genera el retardo requerido. El diagrama de flujo se puede observar en la figura 4.4 a continuación.

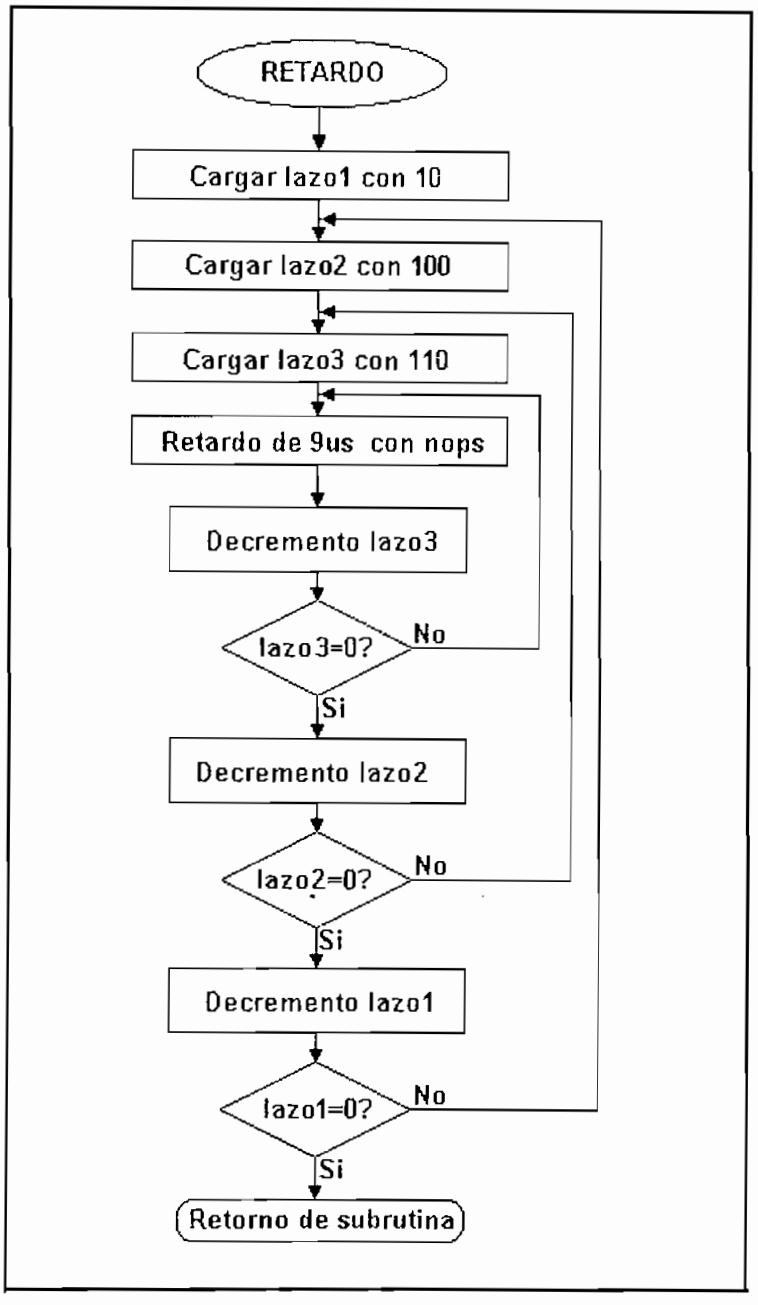


Figura 4.4 Diagrama de flujo del retardo

Práctica 2: Lectura de dipswitch y encendido de leds correspondientes

El objetivo de esta práctica es el uso de los puertos como entradas y como salidas, para esto se desea controlar 4 diodos leds conectados en el puerto A que simplemente indicarán el estado de 4 switchs conectados en el puerto B. El diagrama esquemático se puede observar en la Figura 4.5 a continuación.

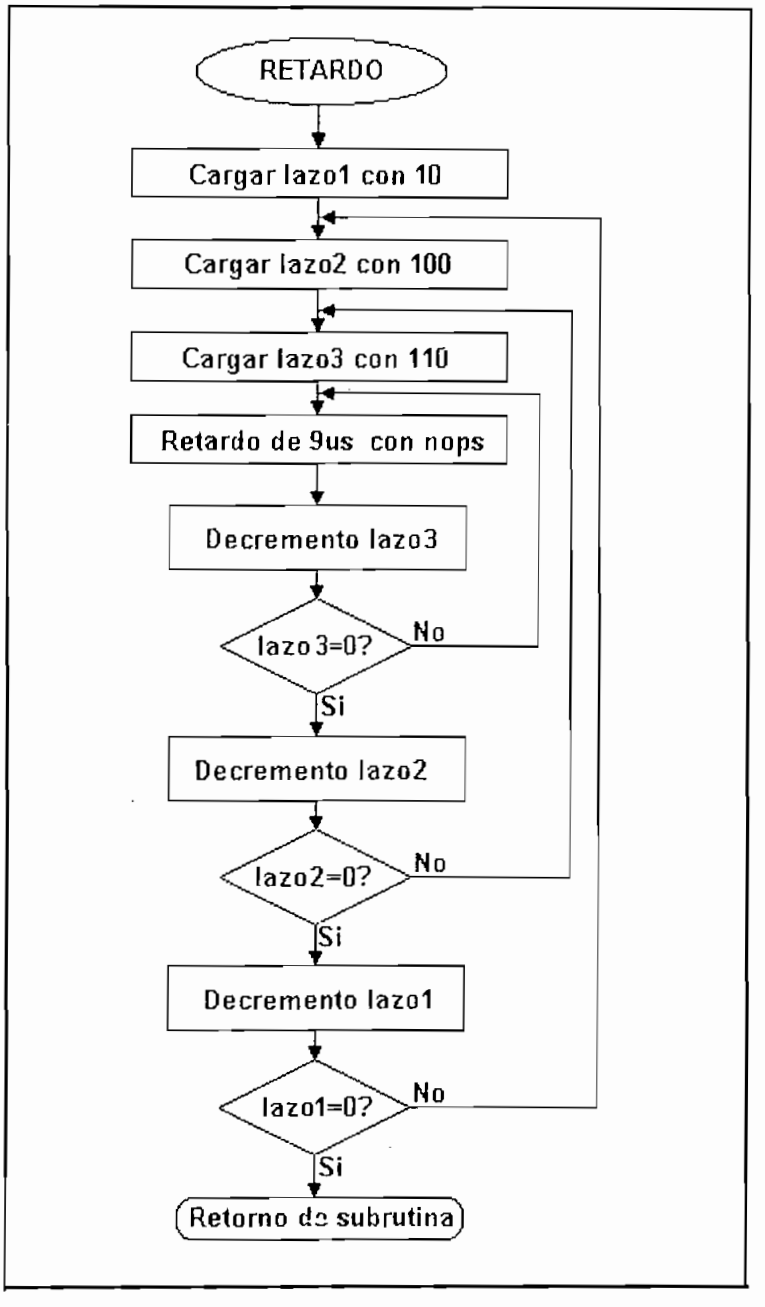


Figura 4.4 Diagrama de flujo del retardo

Práctica 2: Lectura de dipswitch y encendido de leds correspondientes

El objetivo de esta práctica es el uso de los puertos como entradas y como salidas, para esto se desea controlar 4 diodos leds conectados en el puerto A que simplemente indicarán el estado de 4 switchs conectados en el puerto B. El diagrama esquemático se puede observar en la Figura 4.5 a continuación.

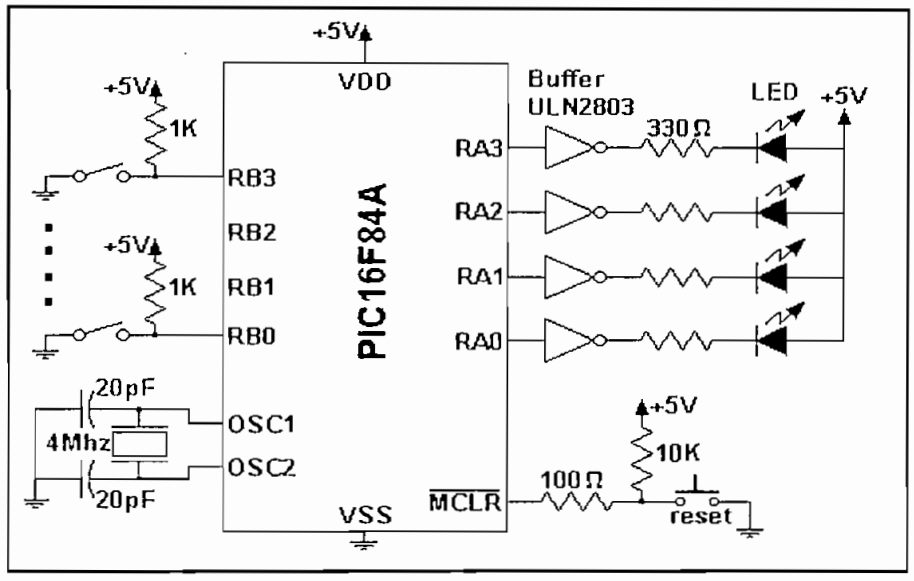


Figura 4.5 Lectura de dipswitch y encendido de leds

En este circuito, los switchs, al estar en posición OFF, sus pines están conectados a través de la resistencia de 1KΩ a 5V, dando un estado lógico alto, en esa posición el led debe apagarse, entonces se requiere invertir el valor leído en el puerto para sacar un estado lógico bajo que apague el diodo. La misma lógica se aplicará en la posición ON. El diagrama de lujo se lo puede observar en la Figura 4.6 y el programa respectivo se lo observa en la Figura 4.7 a continuación.

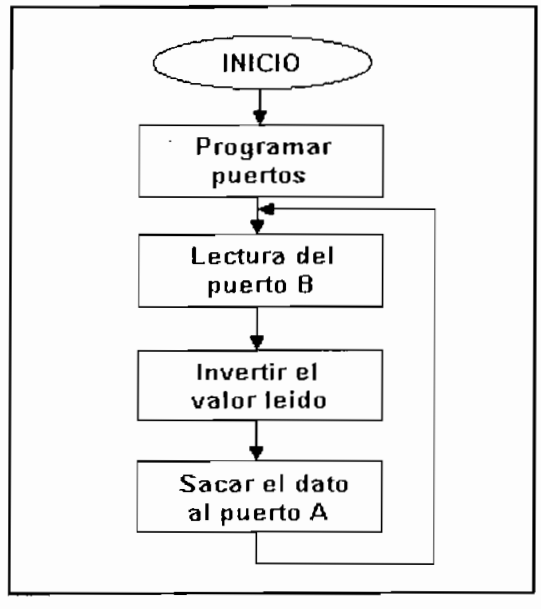


Figura 4.6 Diagrama de flujo para la lectura de dipswitch

```
;Este programa realiza control del encendido y apagado de cuatro leds
;mediante cuatro switchs del bloque de 8 dipswitchs

        list    p=16F84A        ;Microcontrolador utilizado

;Etiquetas

status  equ     03h
porta   equ     05h
portb   equ     06h
trisa   equ     85h
trisb   equ     86h
w       equ     00h

reset   org     0               ;se escribe en la línea 0 la instrucción
        goto    inicio          ;de salto a la línea etiquetada con inicio

        org     5               ;las siguientes líneas se escribirán desde la dirección 5

inicio  bsf     status,5        ;se ubica en el segundo banco de la RAM
        movlw   00h             ;se caraga el registro w con 00h
        movwf   trisa           ;se programa el puerto A como salidas
        movlw   0ffh            ;se caraga el registro w con FFh
        movwf   trisb           ;se programa el puerto B como entradas
        bcf     status,5        ;se ubica en el primer banco de la RAM
ciclo   movf    portb,w         ;el valor del puerto B lo carga en el registro W
        xorlw   0ffh            ;con la operación xor se invierte el valor de W
        movwf   porta           ;pasa el valor de W al puerto A
        goto    ciclo           ;regresa a repetir el proceso
        end                     ;final del programa
```

Figura 4.7 Programa lectura de dipswitch y encendido de leds

Práctica 3: Lectura de teclas y encendido de leds

El uso de teclas en los sistemas digitales tiene el inconveniente de los rebotes de contacto. Para revisar el manejo de estos elementos, se realiza una práctica en la cual se controla cuatro diodos led mediante dos teclas. Al pulsar la tecla1 se encenderán los leds 1 y 2 , y al volver a presionar estos se apagarán; de igual manera funcionará la tecla 2 y los leds 3 y 4. En este caso, en primer lugar el programa debe realizar un monitoreo continuo de las teclas, para ver si estas son o no presionadas, esto se hace con el fin de que no se pierda nunca la acción de la pulsación de una tecla realizada por el usuario. En segundo lugar, para evitar errores por los rebotes o ruidos, una vez que se detecta la pulsación de una tecla, se deja pasar un tiempo de 100ms para entonces comprobar si efectivamente la tecla está presionada, con esto se evita la acción de ruidos. Luego de esto se ejecuta la orden dada por la tecla, ya sea de encender o apagar los diodos correspondientes, y nuevamente se deja pasar un tiempo de 100ms, en este caso para asegurarse que el usuario ha soltado la tecla. Si el usuario no suelta la tecla durante

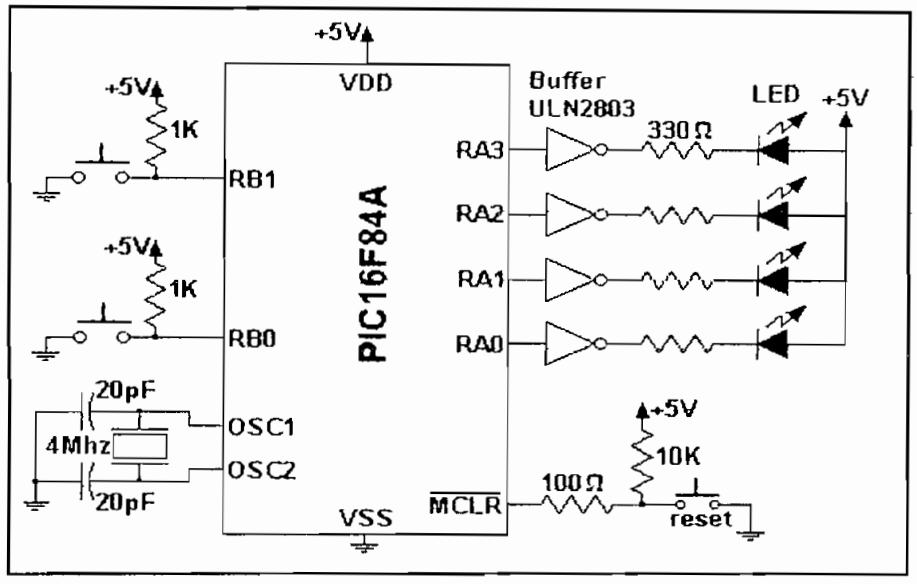


Figura 4.8 Diagrama del circuito de manejo de teclas.

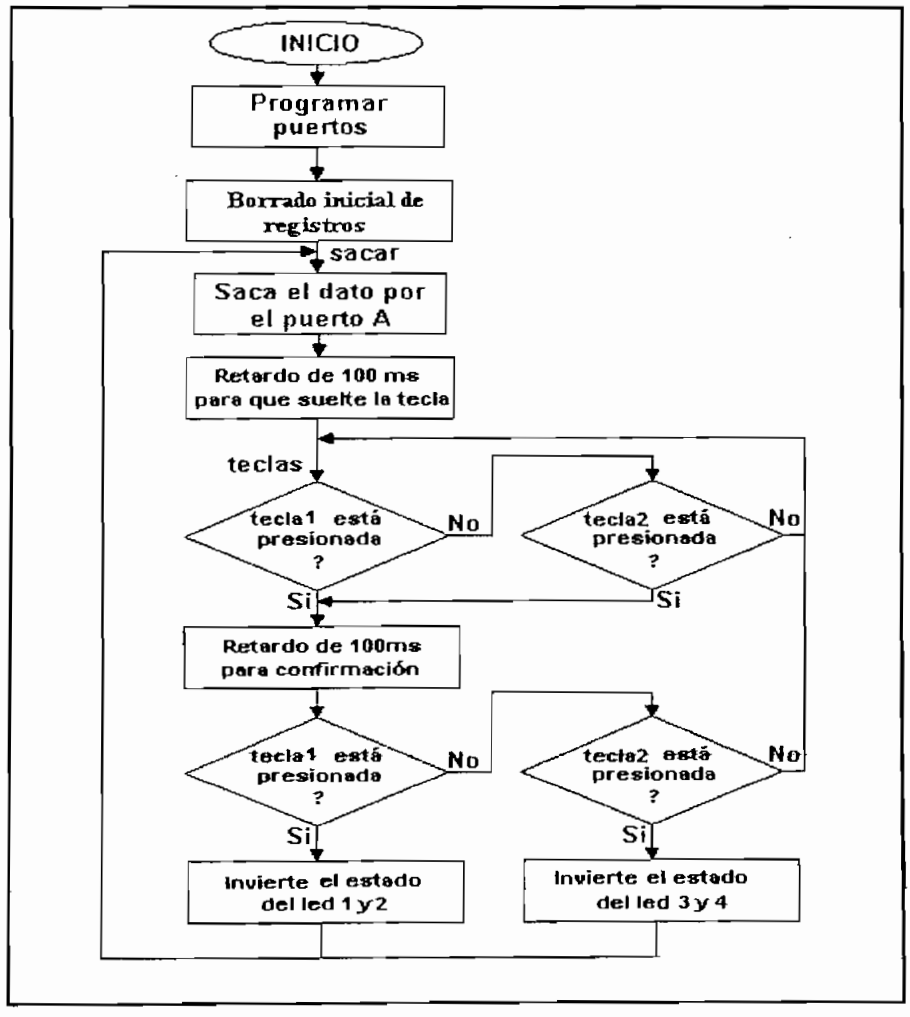


Figura 4.9 Diagrama de flujo de manejo de teclas

300ms el sistema entenderá como dos pulsaciones. En las Figuras 4.8 y 4.9 se muestra el diagrama del circuito y el diagrama de flujo respectivamente.

Para realizar el bloque de cambio de estado de los leds, es interesante el manejo de la operación XOR, la cual invierte el valor de los bits en donde se opera con unos y da como resultado ceros al operar "0 XOR 0", para posteriormente unir estos bits ya invertidos al otro registro mediante la operación OR.

Estas operaciones se pueden observar en la Figura 4.10 y el programa respectivo en la Figura 4.11 a continuación.

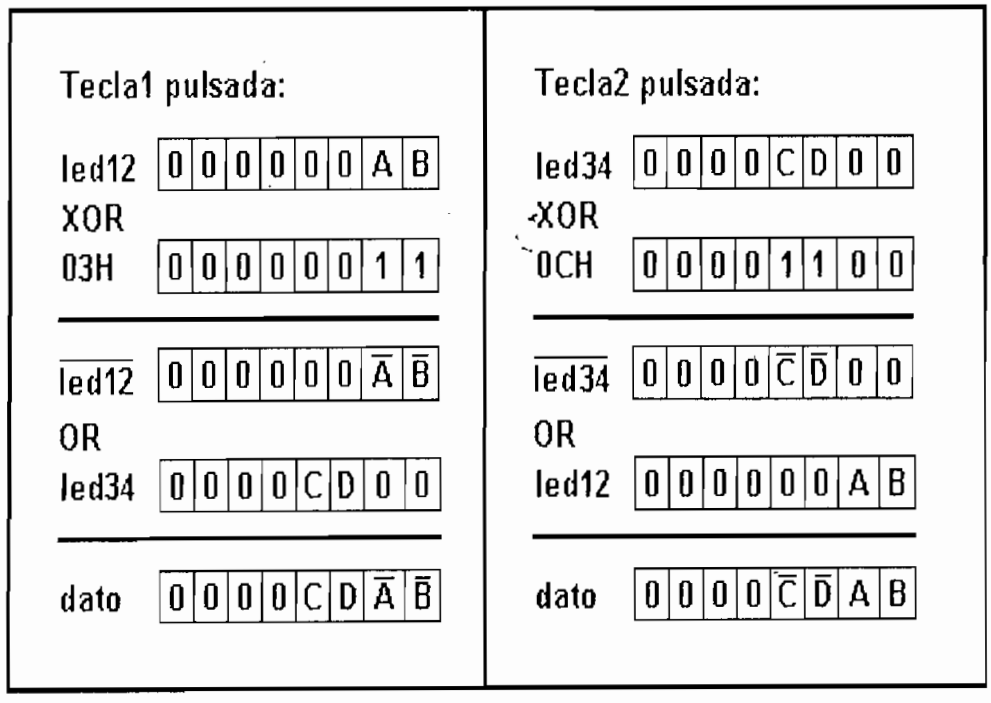


Figura 4.10 Manejo de registros para invertir el estado de leds.

```
;Este programa realiza control del encendido y apagado de cuatro leds
;mediante dos teclas. (Tecla1 controla Leds 1 y 2, y Tecla2 controla
;leds 3 y 4). Al presionar la tecla1 se encienden los leds 1 y 2 y al
;volver a presionar estos se apagan. De igual manera funciona con la
;tecla2 y los diodos 3 y 4.
        list    p=16F84A        ;Microcontrolador utilizado

;Etiquetas

status  equ     03h
porta   equ     05h
portb   equ     06h
trisa   equ     85h
trisb   equ     86h
dato    equ     0ch
led12   equ     0dh
led34   equ     0eh
lazo1   equ     0fh
lazo2   equ     10h
w       equ     00h
r       equ     01h

reset   org     0               ;se escribe en la línea 0 la instrucción
        goto    inicio          ;de salto a la línea etiquetada con inicio

        org     5               ;las siguientes líneas se escribirán desde la dirección 5

;**************Subrutina de retardo de 100 milisegundos*********************
retardo movlw   D'100'  ;El registro lazo1 contiene el número de
        movwf   lazo1   ;veces que repite 1 milisegundo
repite1 movlw   D'110'  ;El registro lazo2 contiene el número de
        movwf   lazo2   ;veces que repite los 9 microsegundos
repite2 nop             ;de retardo generados
        nop             ;por los 6 ciclos de las instrucciones nop (6usg)
        nop             ;más 1 ciclo de la instrucción decfsc (1usg)
        nop             ;más 2 ciclos del salto goto (2usg)
        nop             ;dando en total los 9usg, siendo esta la base
        nop             ;de tiempo, por lo tanto 1sg= 9usg*110*100*10
        decfsz  lazo2   ;decrementa el reg lazo2 y salta si llega a 0
        goto    repite2 ;si lazo2 no es 0 entonces salta a repite2
        decfsz  lazo1   ;decrementa el reg lazo1 y salta si llega a 0
        goto    repite1 ;si lazo1 no es 0 entonces salta a repite
        retlw   0       ;salida de la subrutina cargando w con 0

;******************  Programa principal  **********************************
inicio  bsf     status,5        ;se ubica en el segundo banco de la RAM
        movlw   00h             ;se caraga el registro w con 00h
        movwf   trisa           ;se programa el puerto A como salidas
        movlw   0ffh            ;se caraga el registro w con FFh
        movwf   trisb           ;se programa el puerto B como entradas
        bcf     status,5        ;se ubica en el primer banco de la RAM
ciclo   clrf    led12           ;borrado inicia de los registros
        clrf    led34           ;a utilizar en el programa
        clrf    dato            ;se borra el registro dato para iniciar apagando los leds
sacar   movf    dato,w          ;carga el registro dato a w
        movwf   porta           ;para sacarlo por el puerto A
        call    retardo         ;retardo para que suelte la tecla
teclas  btfss   portb,0         ;pregunta se la tecla1 está presionada
        goto    confm           ;está presionada por lo salta a confirmar
        btfss   portb,1         ;pregunta si la tecla2 está presionada
        goto    confm           ;está presionada por lo salta a confirmar
        goto    teclas          ;no están presionadas, regresa a sequir verificando
confm   call    retardo         ;si está oprimida retarda 100msg para confirmar
        btfss   portb,0         ;pregunta se la tecla1 está presionada
        goto    mod12           ;está presionada por lo salta a modificar leds12
        btfss   portb,1         ;pregunta si la tecla2 está presionada
        goto    mod34           ;está presionada por lo salta a modificar leds34
        goto    teclas          ;no se confirma, regresa a sequir verificando
mod12   movlw   B'00000011'     ;se carga W con FF
        xorwf   led12,w         ;se invierten los bits 0 y 1 del registro led12
        movwf   led12           ;se actualiza led12 con los bits complementados
        iorwf   led34,w         ;unimos los registros led12 con led34
        movwf   dato            ;y cargamos en el registro dato
        goto    sacar           ;salta a sacar el dato por el puerto A
mod34   movlw   B'00001100'     ;se carga W con FF
        xorwf   led34,w         ;se invierten los bits 2 y 3 del registro led12
        movwf   led34           ;se actualiza led12 con los bits complementados
        iorwf   led12,w         ;unimos los registros led12 con led34
        movwf   dato            ;y cargamos en el registro dato
        goto    sacar           ;salta a sacar el dato por el puerto A
        end                     ;final del programa
```

Figura 4.11 Programa de manejo de teclas

4.1.2 MANEJO DE DISPLAYS DE 7 SEGMENTOS

Práctica 4: Manejo de un display de 7 segmentos

El uso de displays es útil en el diseño de controles, mediante el cual se puede observar el estado de una máquina, valores de señales medidas como temperatura, velocidad, el valor del conteo de eventos, etc. En esta práctica, para iniciarse en el uso de estos elementos, se realizará la cuenta decimal de pulsaciones de una tecla, al llegar al valor máximo de nueve se reiniciará en cero. Para esto se utilizará el decodificador 9368 de BCD a 7 segmentos; este decodificador puede mostrar valores hexadecimales; es decir, puede mostrar caracteres de la A hasta la F.

En las Figuras 4.12, 4.13 y 4.14 se observa el circuito, el diagrama de flujo y programa respectivamente.

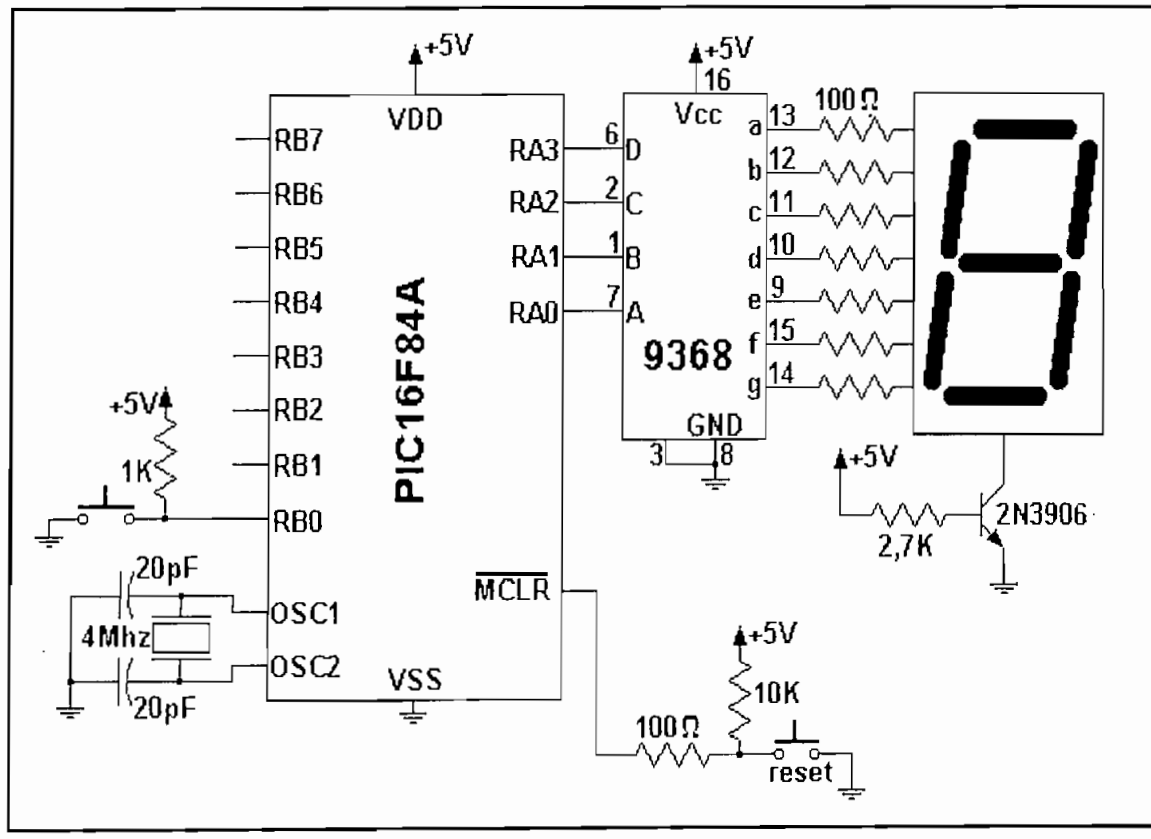


Figura 4.12 Circuito del manejo de un display

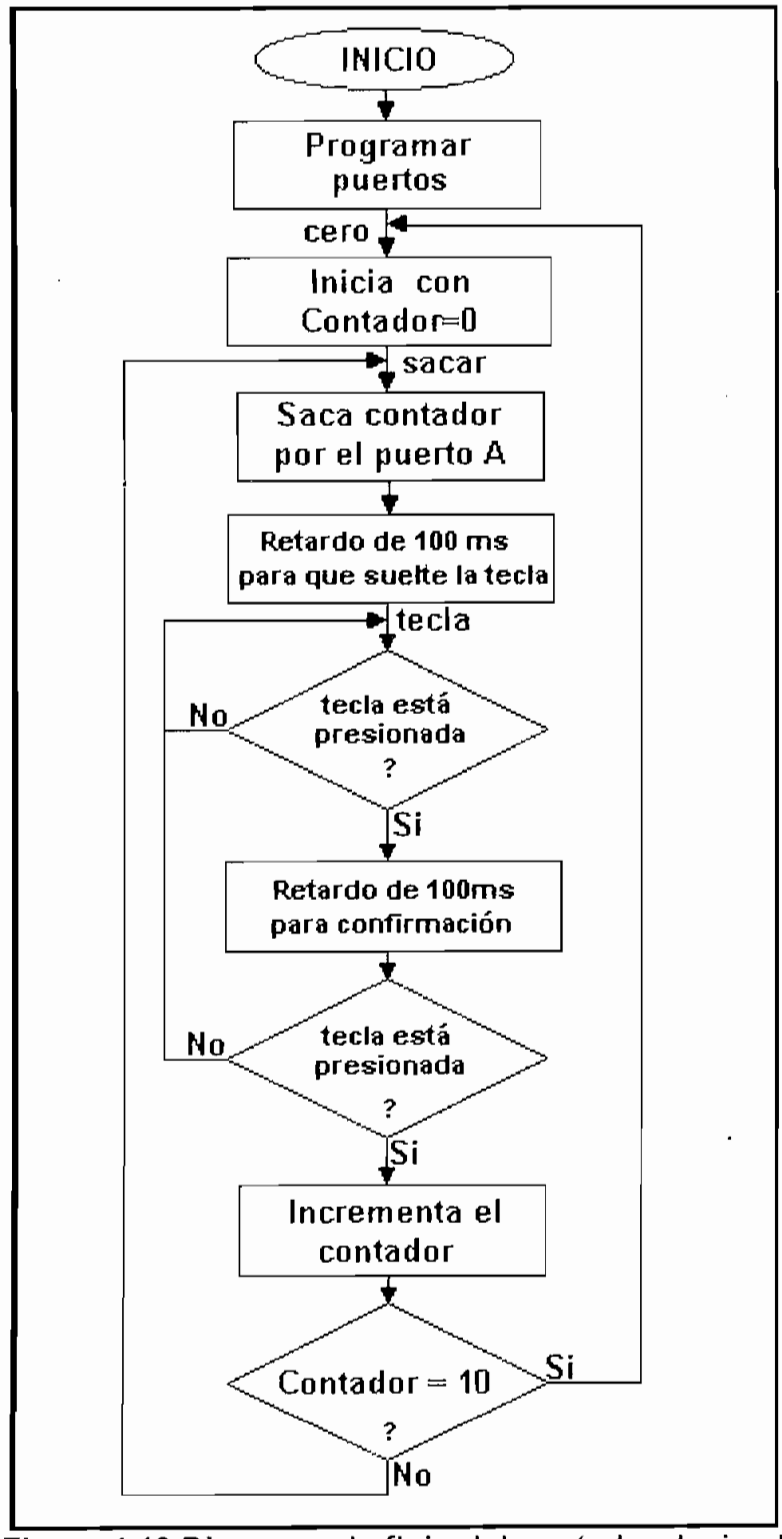


Figura 4.13 Diagrama de flujo del contador decimal

```
;Este programa realiza el conteo de pulsaciones de 0 a 9 y lo muestra
;en un display de 7 segmentos.
        list    p=16F84A        ;Microcontrolador utilizado

;Etiquetas
status  equ     03h             ;registro de estados
porta   equ     05h             ;el puerto A está en la dirección 05 de la RAM
portb   equ     06h             ;el puerto B está en la dirección 06 de la RAM
trisa   equ     85h             ;registro de configuración del puerto A
trisb   equ     86h             ;registro de configuración del puerto B
conta   equ     0ch             ;registro que lleva la cuenta de pulsaciones
lazo1   equ     0dh             ;registro para conteo en la subrutina retardo
lazo2   equ     0eh             ;registro para conteo en la subrutina retardo
w       equ     00h             ;indica que el resultado se guarda en W
r       equ     01h             ;indica que el resultado se guarda en el registro
z       equ     02h             ;bandera de cero en el registro de estados

reset   org     0               ;se escribe en la línea 0 la instrucción
        goto    inicio          ;de salto a la línea etiquetada con inicio

        org     5               ;las sigui. líneas se escribirán desde direc.05h

;**************Subrutina de retardo de 100 milisegundos*******************
retardo movlw   D'100'          ;El registro lazo1 contiene el número de
        movwf   lazo1           ;veces que repite 1 milisegundo
repite1 movlw   D'110'          ;El registro lazo2 contiene el número de
        movwf   lazo2           ;veces que repite los 9 microsegundos
repite2 nop                     ;de retardo generados
        nop                     ;por los 6 ciclos de las instrucciones nop (6usg)
        nop                     ;más 1 ciclo de la instrucción decfsc (1usg)
        nop                     ;más 2 ciclos del salto goto (2usg)
        nop                     ;dando en total los 9usg, siendo esta la base
        nop                     ;de tiempo, por lo tanto 1sg= 9usg*110*100*10
        decfsz  lazo2           ;decrementa el reg lazo2 y salta si llega a 0
        goto    repite2         ;si lazo2 no es 0 entonces salta a repite2
        decfsz  lazo1           ;decrementa el reg lazo1 y salta si llega a 0
        goto    repite1         ;si lazo1 no es 0 entonces salta a repite
        retlw   0               ;salida de la subrutina cargando w con 0
;******************  Programa principal  *********************************
inicio  bsf     status,5        ;se ubica en el segundo banco de la RAM
        movlw   00h             ;se caraga el registro w con 00h
        movwf   trisa           ;se programa el puerto A como salidas
        movlw   0ffh            ;se caraga el registro w con FFh
        movwf   trisb           ;se programa el puerto B como entradas
        bcf     status,5        ;se ubica en el primer banco de la RAM
cero    clrf    conta           ;inicia con el contador en cero
sacar   movf    conta,w         ;carga el registro conta a w
        movwf   porta           ;para sacarlo por el puerto A
        ;call   retardo         ;retardo para que suelte la tecla
teclas  btfsc   portb,0         ;pregunta si la tecla está presionada
        goto    teclas          ;no está presionadas, regresa a sequir verificando
confm   ;call   retardo         ;si está presionada, retardo para confirmar
        btfsc   portb,0         ;pregunta si la tecla está presionada
        goto    teclas          ;no se confirma, regresa a sequir verificando
        incf    conta           ;si lo confirma, incrementa el contador
        movf    conta,w         ;carga W con el valor del conteo
        xorlw   0ah             ;compara con xor si es igual a 10
        btfsc   status,z        ;pregunta si el contador es igual a 10
        goto    cero            ;si es igual el contador se debe poner en 0
        goto    sacar           ;si no llegó a 10 solo salta a sacar conta
        end                     ;final del programa
```

Figura 4.14 Programa del contador decimal

4.1.3 MULTIPLEXAJE DE TECLADOS Y DISPLAYS

Práctica 5: Multiplexaje de displays

La técnica del multiplexaje permite optimizar la cantidad de pines utilizados del microcontrolador, ya que normalmente estos son limitados y se los puede utilizar para otros fines, dando así un mejor aprovechamiento del dispositivo.

El multiplexaje consiste en la transmisión de dos o más señales por un mismo medio conductor en tiempos distintos y secuenciales. Esto se aplica en esta práctica, en la que muestra en cuatro displays un valor de cuatro dígitos. Si se deseara manejar cuatro displays independientes se requeriría de 16 pines, cuatro para cada display, además de 4 decodificadores de BCD a 7 segmentos; sin embargo, con la técnica de multiplexaje se reduce a la mitad el número de pines utilizados y con un solo decodificador.

En el circuito de la Figura 4.15 se puede analizar el funcionamiento de este proceso. Si en el circuito se activa por ejemplo el transistor del display menos significativo, colocando un uno lógico en el pin RA0, entonces se mostrará el dato que sale por el puerto B que debe ser el dígito menos significativo correspondiente a este display, mientras que los otros displays permanecen apagados. Si se saca luego el dato correspondiente al segundo display y se activa el transistor correspondiente, poniendo un uno lógico en el pin RA1, mientras el resto de transistores permanecen apagados, entonces se visualizará el dato del segundo display mientras que el resto de displays permanecen apagados. Si este proceso se lo realiza hasta el display más significativo y luego se repite secuencialmente a alta velocidad entonces se observará como si todos los displays estuvieran encendidos simultáneamente, mostrando así los cuatro dígitos.

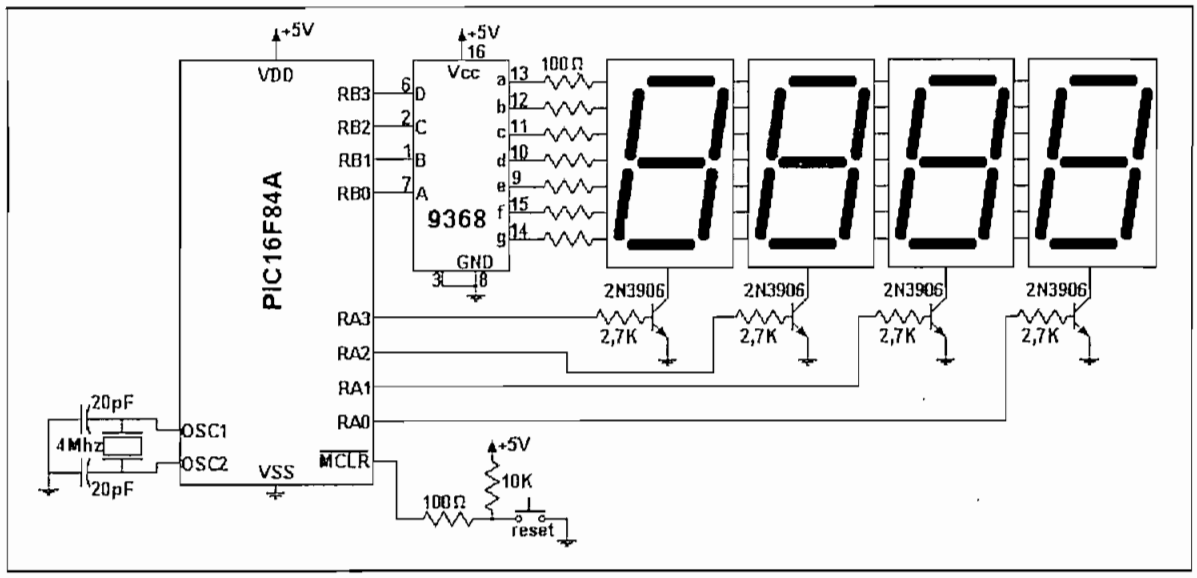


Figura 4.15 Circuito de multiplexaje de displays

Esto es posible gracias a que la retina humana memoriza las imágenes durante varios milisegundos, mientras que los displays se encienden y apagan en pocos microsegundos. Se debe tener cuidado al realizar este proceso en el cambio de un display a otro, ya que pueden aparecer caracteres erróneos en los displays. Para esto se debe seguir la siguiente secuencia:

- Apagar todos los transistores (con ceros en el puerto A)
- Sacar el dígito1 por el puerto B
- Activar el transistor del display 1 (Uno en RA0)
- Generar un retardo para la visualización
- Desactivar el transistor del display 1 (Cero en RA0)
- Sacar el dígito2 por el puerto B
- Activar el transistor del display2 (Uno en RA1)
- Generar un retardo para la visualización
- Desactivar el transistor del display 2 (Cero en RA1)

Este proceso se repite hasta llegar al cuarto dígito, para luego de esto iniciar nuevamente con el dígito uno. Se debe notar que una vez puesto el dato en el puerto B, se realiza el encendido del display correspondiente, el retardo y en seguida el apagado del display, para luego actualizar el dato del dígito del siguiente display en el puerto B, mientras todos los displays están apagados, evitando así que aparezcan datos en displays incorrectos.

La visualización adecuada en los displays depende de la intensidad del brillo de los segmentos y también del tiempo (retardo) de encendido de los displays. Un retardo de visualización adecuado está por los 3 milisegundos, con un brillo dado por las resistencias limitadoras de unos 100Ω.

En la Figura 4.16 y 4.17 se puede observar el diagrama de flujo y también el programa para esta práctica.

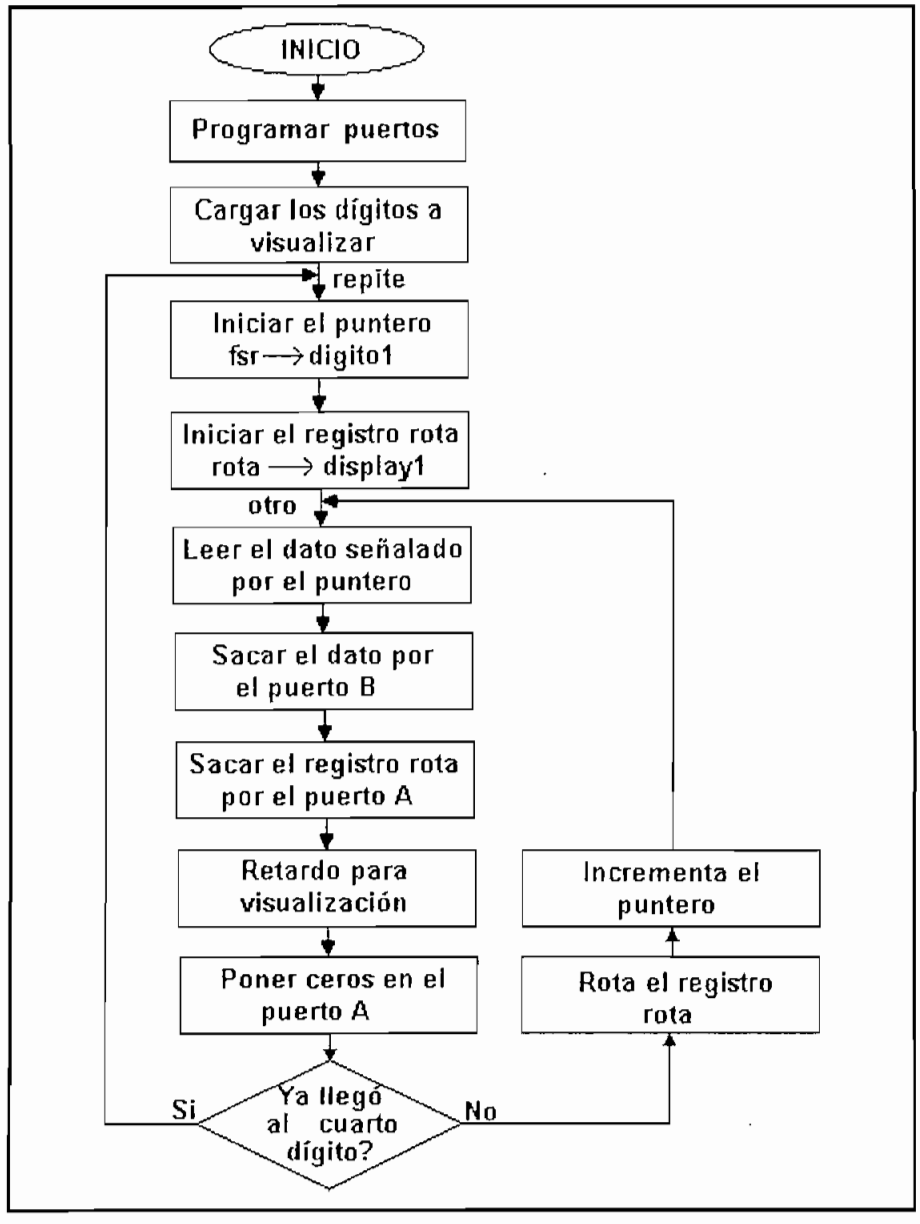


Figura 4.16 Diagrama de flujo del multiplexaje de displays

```
;Este programa realiza el multiplexaje de 4 displays
;para presentar un número de 4 digitos (5F4C)
;almacenado en los registros dig1 a dig4
        list    p=16F84A        ;Microcontrolador utilizado

;Etiquetas
indo    equ     00h             ;registro de direccionamiento indirecto
status  equ     03h             ;registro de estados
Fsr     equ     04h             ;registro selector de registros (puntero)
porta   equ     05h             ;el puerto A está en la dirección 05 de la RAM
portb   equ     06h             ;el puerto B está en la dirección 06 de la RAM
lazo1   equ     0ch             ;registro para conteo en la subrutina retardo
lazo2   equ     0dh             ;registro para conteo en la subrutina retardo
rota    equ     0eh             ;registro que lleva un 1 para activar displays
dig1    equ     0fh             ;dígito 1 menos significativo
dig2    equ     10h             ;dígito 2
dig3    equ     11h             ;dígito 3
dig4    equ     12h             ;dígito 4 más significativo
trisa   equ     85h             ;registro de configuración del puerto A
trisb   equ     86h             ;registro de configuración del puerto B
w       equ     00h             ;indica que el resultado se guarda en W

reset   org     0               ;se escribe en la línea 0 la instrucción
        goto    inicio          ;de salto a la línea etiquetada con inicio

        org     5               ;las sigui. líneas se escribirán desde direc.05h

;**************Subrutina de retardo de 3 milisegundos*********************
retardo movlw   D'3'            ;El registro lazo1 contiene el número de
        movwf   lazo1           ;veces que repite 1 milisegundo
repite1 movlw   D'110'          ;El registro lazo2 contiene el número de
        movwf   lazo2           ;veces que repite los 9 microsegundos
repite2 nop                     ;de retardo generados
        nop                     ;por los 6 ciclos de las instrucciones nop (6usg)
        nop                     ;más 1 ciclo de la instrucción decfsc (1usg)
        nop                     ;más 2 ciclos del salto goto (2usg)
        nop                     ;dando en total los 9usg, siendo esta la base
        nop                     ;de tiempo, por lo tanto 1sg= 9usg*110*100*10
        decfsz  lazo2           ;decrementa el reg lazo2 y salta si llega a 0
        goto    repite2         ;si lazo2 no es 0 entonces salta a repite2
        decfsz  lazo1           ;decrementa el reg lazo1 y salta si llega a 0
        goto    repite1         ;si lazo1 no es 0 entonces salta a repite
        retlw   0               ;salida de la subrutina cargando w con 0

;********************  Programa principal  *********************************
inicio  bsf     status,5        ;se ubica en el segundo banco de la RAM
        movlw   00h             ;se caraga el registro w con 00h
        movwf   trisa           ;se programa el puerto A como salidas
        movwf   trisb           ;se programa el puerto B como salidas
        bcf     status,5        ;se ubica en el primer banco de la RAM
        movlw   0Ch             ;carga W con el dígito Ch
        movwf   dig1            ;carga al registro digito1 con C
        movlw   04h             ;carga W con el dígito 4h
        movwf   dig2            ;carga al registro digito2 con 4
        movlw   0fh             ;carga W con el dígito fh
        movwf   dig3            ;carga al registro digito3 con F
        movlw   05h             ;carga W con el dígito 5h
        movwf   dig4            ;carga al registro digito4 con 5
repite  movlw   dig1            ;Inicia el puntero fsr apuntando a
        movwf   fsr             ;la dirección del digito1
        movlw   B'00000001'     ;carga el registro rota con un uno
        movwf   rota            ;en el bit (display) menos significativo
otro    movf    indo,w          ;lee el dato del registro apuntado por el fsr
        movwf   portb           ;saca el dato por el puerto B
        movf    rota,w          ;lee el registro rota para
        movwf   porta           ;cargarlo el en puerto A
        call    retardo         ;llama al retardo de 3msg para visualización
        clrf    porta           ;saca ceros para deshabilitar los transistores
        btfsc   rota,3          ;pregunto si ya pasaron 4 rotaciones?
        goto    repite          ;salta a repetir el proceso
        rlf     rota            ;rota a la izq. el uno que habilita el display
        incf    fsr             ;apunta al próximo dígito a mostrar
        goto    otro            ;y salta a mostrar el proximo digito
        end                     ;final del programa
```

figura 4.17 Programa del multiplexaje de displays

Práctica 6: Multiplexaje de teclados

Esta práctica consiste en el manejo de un teclado hexadecimal de 16 teclas en la cual al presionar una de las teclas, el valor correspondiente a esta será mostrado en un display de siete segmentos.

Para poder implementar este sistema con un micro PIC16F84A, el cual posee unicamente13 pines de entrada-salida, se aplicará también la técnica de multiplexaje para controlar el teclado. Como se puede observar en la Figura 4.18 el teclado está conformado por una matriz de cuatro filas por cuatro columnas, esta técnica consiste en el análisis de una fila cada vez, en la que se chequea si una de las cuatro teclas del teclado está presionada.

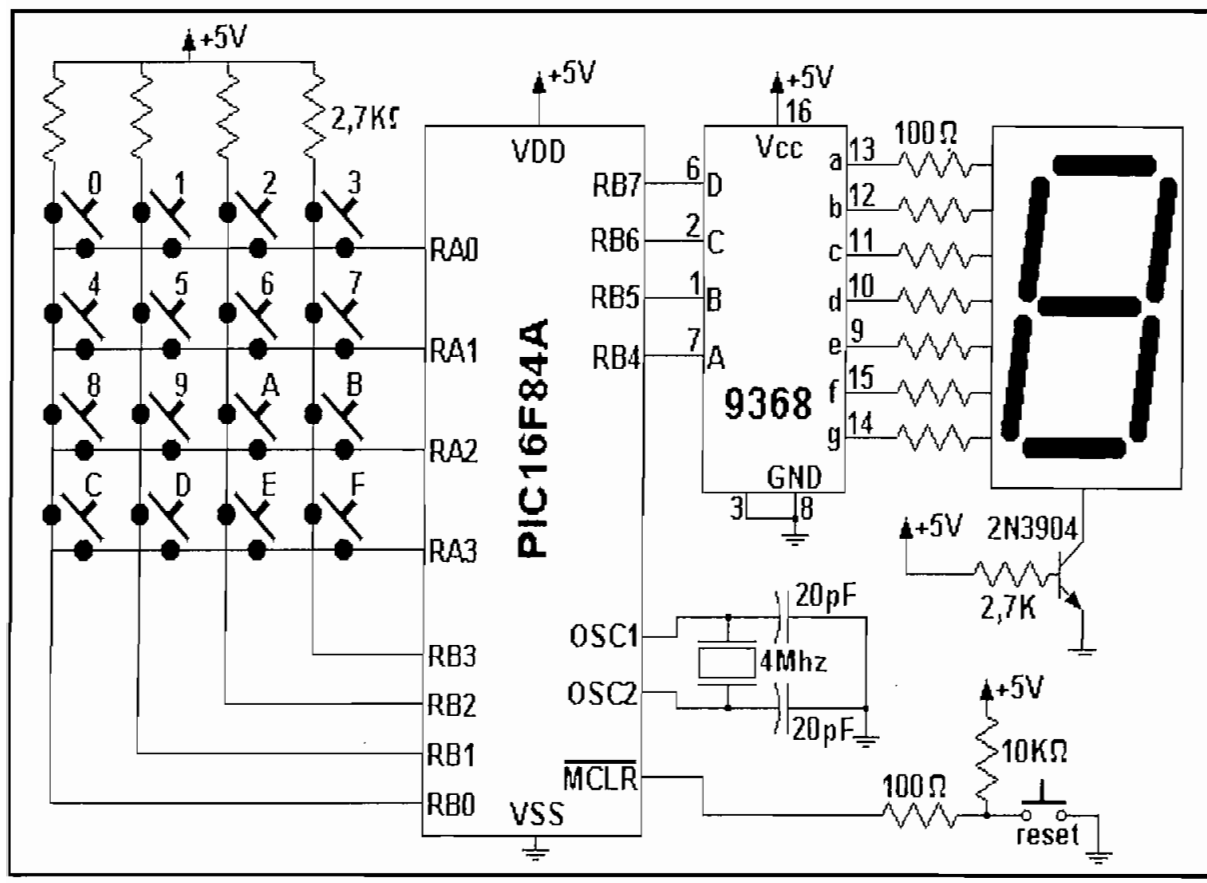


Figura 4.18 Circuito de manejo de un teclado matricial

Para realizar esto se configura al puerto A como salidas y al puerto B como entradas. Como se puede observar, normalmente en los pines del puerto B está ingresando un uno lógico ya que estos pines están conectados por las resistencias pull-up a cinco voltios; sin embargo, esta condición puede cambiar si por ejemplo el pin RA0 es conectado a tierra (RA0 =0L) y si alguna de las teclas de esta fila es presionada, por ejemplo al presionar la tecla 2 en el pin RB2 ingresará un cero lógico; entonces, si se saca un cero lógico en RA0 y al leer el puerto B se ve que ingresa un cero lógico en el pin RB2, entonces se puede concluir que se encuentra presionada la tecla 2.

Para analizar todo el teclado se debe entonces sacar un cero en la fila uno y leer el puerto B, a ver si existe una tecla presionada de esta fila, mientras las otras filas permanecen en estado alto; luego se prosigue poniendo un cero en la segunda fila y de igual manera se lee el puerto B, para ver si hay una tecla presionada de esta fila, y así sucesivamente hasta la cuarta fila, para entonces repetir el proceso iniciando nuevamente por la fila uno.

Este proceso se lo debe realizar a alta velocidad para así no perder nunca la acción de presión de una tecla por parte del usuario. En esta práctica con el cristal de 4Mhz, el análisis de todo el teclado tarda menos de 60us.

También en esta práctica se aplicarán los retardos necesarios para la eliminación de los rebotes de contacto.

El diagrama de flujo y el programa para esta práctica se observan en las Figuras 4.19 y 4.20, respectivamente.

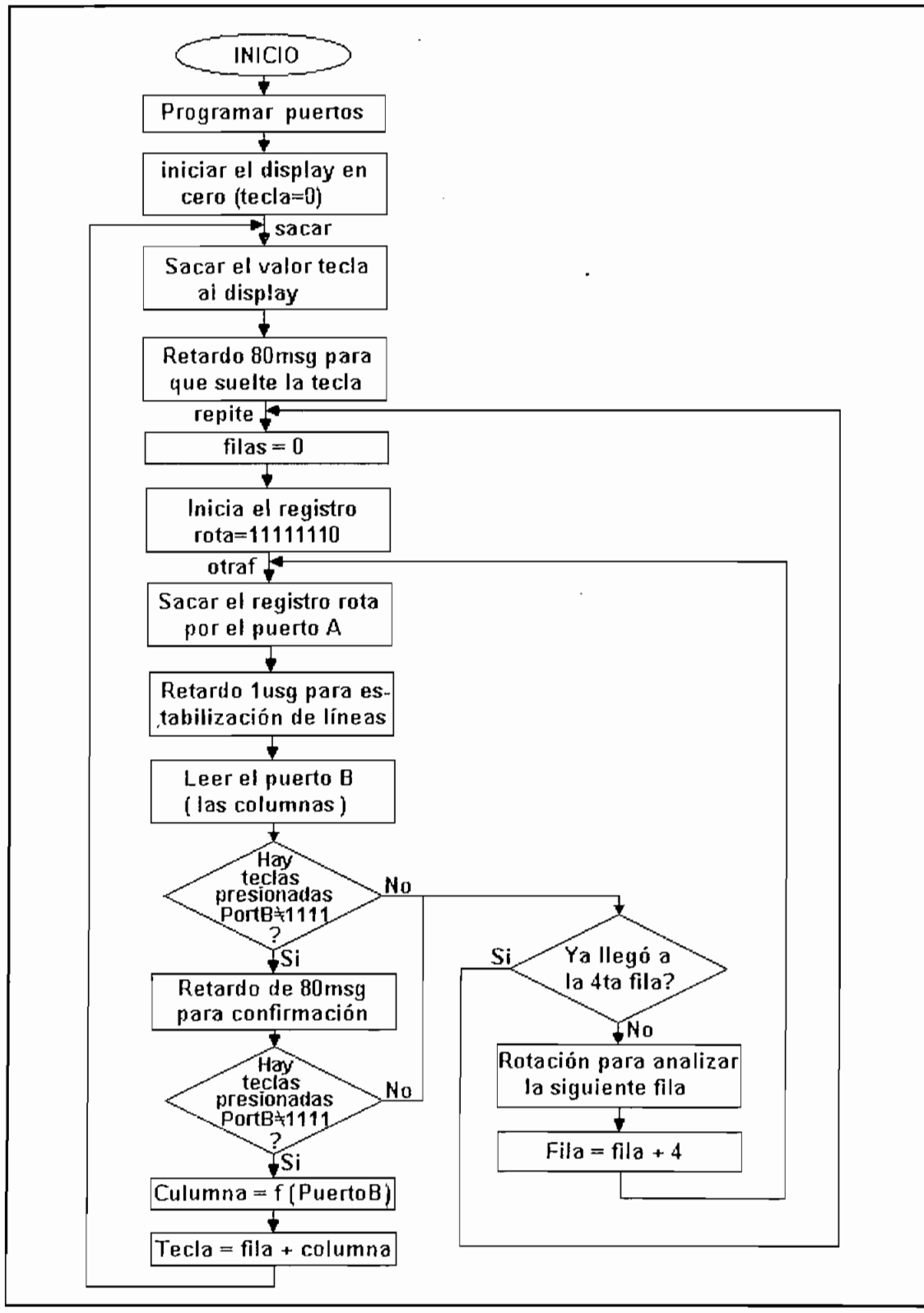


Figura 4.19 Diagrama de flujo del multiplexaje de teclados

```
;Este programa realiza la lecura de un teclado hexadecimal de 16 teclas
;el valor de la tecla presionada es mostrado en un display de 7 segmentos
        list    p=16F84A        ;Microcontrolador utilizado

;Etiquetas
pc      equ     02h             ;registro contador de programa
status  equ     03h             ;registro de estados
fsr     equ     04h             ;registro selector de registros (puntero)
porta   equ     05h             ;el puerto A está en la dirección 05 de la RAM
portb   equ     06h             ;el puerto B está en la dirección 06 de la RAM
lazo1   equ     0ch             ;registro para conteo en la subrutina retardo
lazo2   equ     0dh             ;registro para conteo en la subrutina retardo
rota    equ     0eh             ;registro que lleva un 1 para activar displays
tecla   equ     0fh             ;contiene el valor de la tecla presionada
fila    equ     10h             ;contiene el valor de la fila a analizar
trisa   equ     85h             ;registro de configuración del puerto A
trisb   equ     86h             ;registro de configuración del puerto B
w       equ     00h             ;indica que el resultado se guarda en W
r       equ     01h             ;el resultado se guarda en el registro
z       equ     02h             ;bandera de cero del registro de estados
c       equ     00h             ;bandera de carry del registro de estados
reset   org     0               ;se escribe en la línea 0 la instrucción
        goto    inicio          ;de salto a la línea etiquetada con inicio

        org     5               ;las sigui. líneas se escribirán desde la dir. 5

;**************Subrutina de retardo de 80 milisegundos*******************
retardo movlw   D'80'   ;El registro lazo1 contiene el número de
        movwf   lazo1   ;veces que repite 1 milisegundo
repite1 movlw   D'110'  ;El registro lazo2 contiene el número de
        movwf   lazo2   ;veces que repite los 9 microsegundos
repite2 nop             ;de retardo generados
        nop             ;por los 6 ciclos de las instrucciones nop (6usg)
        nop             ;más 1 ciclo de la instrucción decfsc (1usg)
        nop             ;más 2 ciclos del salto goto (2usg)
        nop             ;dando en total los 9usg, siendo esta la base
        nop             ;de tiempo, por lo tanto 1sg= 9usg*110*100*10
        decfsz  lazo2   ;decrementa el reg lazo2 y salta si llega a 0
        goto    repite2 ;si lazo2 no es 0 entonces salta a repite2
        decfsz  lazo1   ;decrementa el reg lazo1 y salta si llega a 0
        goto    repite1 ;si lazo1 no es 0 entonces salta a repite
        retlw   0       ;salida de la subrutina cargando w con 0

;************************ tabla para hallar el valor de columna **********
tabla   addwf   pc      ;suma W al PC
        nop
        retlw   0       ;columna = 0
        retlw   1       ;columna = 1
        nop
        retlw   2       ;columna = 2
        nop
        nop
        nop
        retlw   3       ;columna = 3

;******************  Programa principal  *********************************
inicio  bsf     status,5        ;se ubica en el segundo banco de la RAM
        movlw   00h             ;se caraga el registro w con 00h
        movwf   trisa           ;se programa el puerto A como salidas
        movlw   0fh             ;se caraga el registro w con 0Fh
        movwf   trisb           ;puerto B (0:3) como entradas; (4:7)como salidas
        bcf     status,5        ;se ubica en el primer banco de la RAM
        clrf    tecla           ;borrado inicia del registro teclas
sacar   swapf   tecla,w         ;intercambio de los 4 bits bajos por los altos
        movwf   portb           ;para cargarlo en la parte alta del puerto B)
        call    retardo         ;retardo para que suelte la tecla
repite  clrf    fila            ;Carga registro fila con el valor de fila1 = 0
        movlw   B'11111110'     ;además notar que: fila2=4; fila3=8; fila4=12
        movwf   rota            ;rota con cero ubicado para analizar fila1
otraf   movf    rota,w          ;carga el registro rota a W para sacarlo por el
        movwf   porta           ;puerto A y poner un cero en una de las filas
        nop                     ;retardo 1usg para que se estabilicen las lineas
        movf    portb,w         ;lee el puerto B (columnas conectadas a este)
        andlw   0fh             ;elimina la parte alta del byte leido
        xorlw   0Fh             ;se invierte la parte baja del byte leido
        btfsc   status,z        ;pregunta si el resultado es cero
        goto    rotar           ;no hay teclas presionadas, salta a rotar
        call    retardo         ;si hay teclas presionadas, retardo de confirma.
```

```
        movf    portb,w         ;lee el puerto B (columnas conectadas a este)
        andlw   0fh             ;elimina la parte alta del byte leido
        xorlw   0Fh             ;se invierte la parte baja del byte leido
        btfsc   status,z        ;pregunta si el resultado es cero
        goto    rotar           ;no hay teclas presionadas, salta a rotar
        call    tabla           ;llama tabla, para hallar el valor de columna
        addwf   fila,w          ;retorna cargando columna en W
        movwf   tecla           ;valor de la tecla= fila + columna
        goto    sacar           ;salta a mostrar tecla en el display
rotar   btfss   rota,3          ;pregunto si ya llegó a la cuarta fila
        goto    repite          ;si llegó: salta a repetir en la fila 1
        bsf     status,c        ;coloca el bit del carry en 1 para rotar
        rlf     rota            ;a la izquierda el cero que va a las filas
        movlw   4               ;carga a W con 4 para
        addwf   fila,r          ;sumarlo al registro fila (fila=fila+4)
        goto    otraf           ;salta analizar otra fila
        end                     ;final del programa
```

Figura 4.20 Programa de multiplexaje de teclados

Práctica 7: Multiplexaje simultáneo de teclados y displays

El manejo simultaneo de teclados y displays es común en aplicaciones de control, en este caso se puede aplicar la técnica de multiplexaje compartiendo cuatro líneas (RB<0:3>) tanto para las filas del teclado como para activar las bases de los transistores que activan los displays, optimizando aún más el número de pines utilizados. En esta práctica, al inicio los displays empezarán con ceros y luego los valores de las teclas presionadas irán rotando en los displays de derecha a izquierda.

El circuito se observa en la Figura 4.21, en este se adicionan inversores para que el cero de la rotación en el puerto B, que habilita las filas, se convierta en un uno, que al rotar active los displays secuencialmente. Estos inversores se pueden eliminar al utilizar displays de ánodo común y transistores PNP, los cuales activarían directamente con el cero de la rotación sin necesidad de invertirlo.

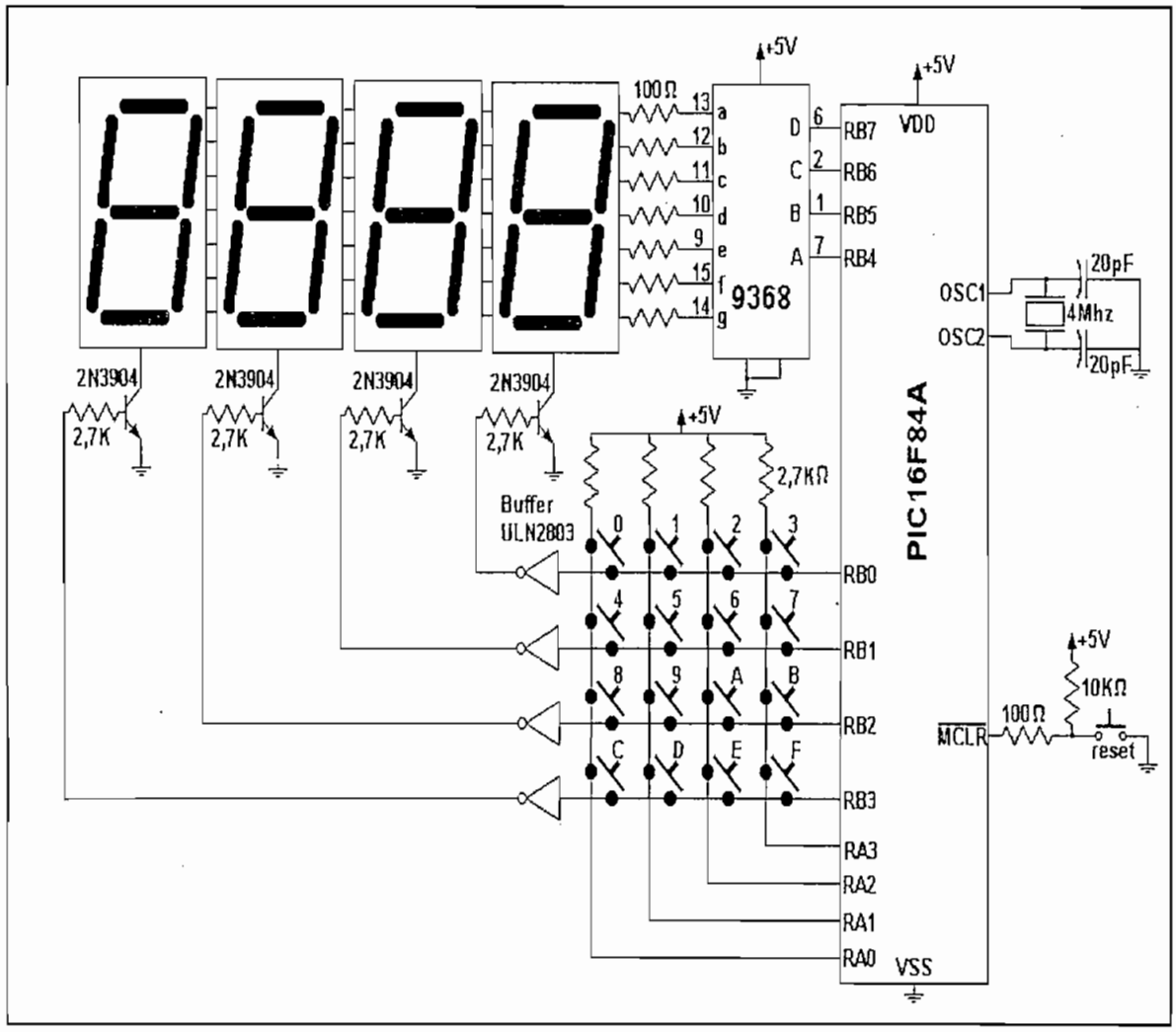


Figura 4.21 Circuito de multiplexaje de teclado y displays

En las Figuras 4.22 y 4.23 se puede observar los diagramas de flujo y el programa que resuelven esta práctica.

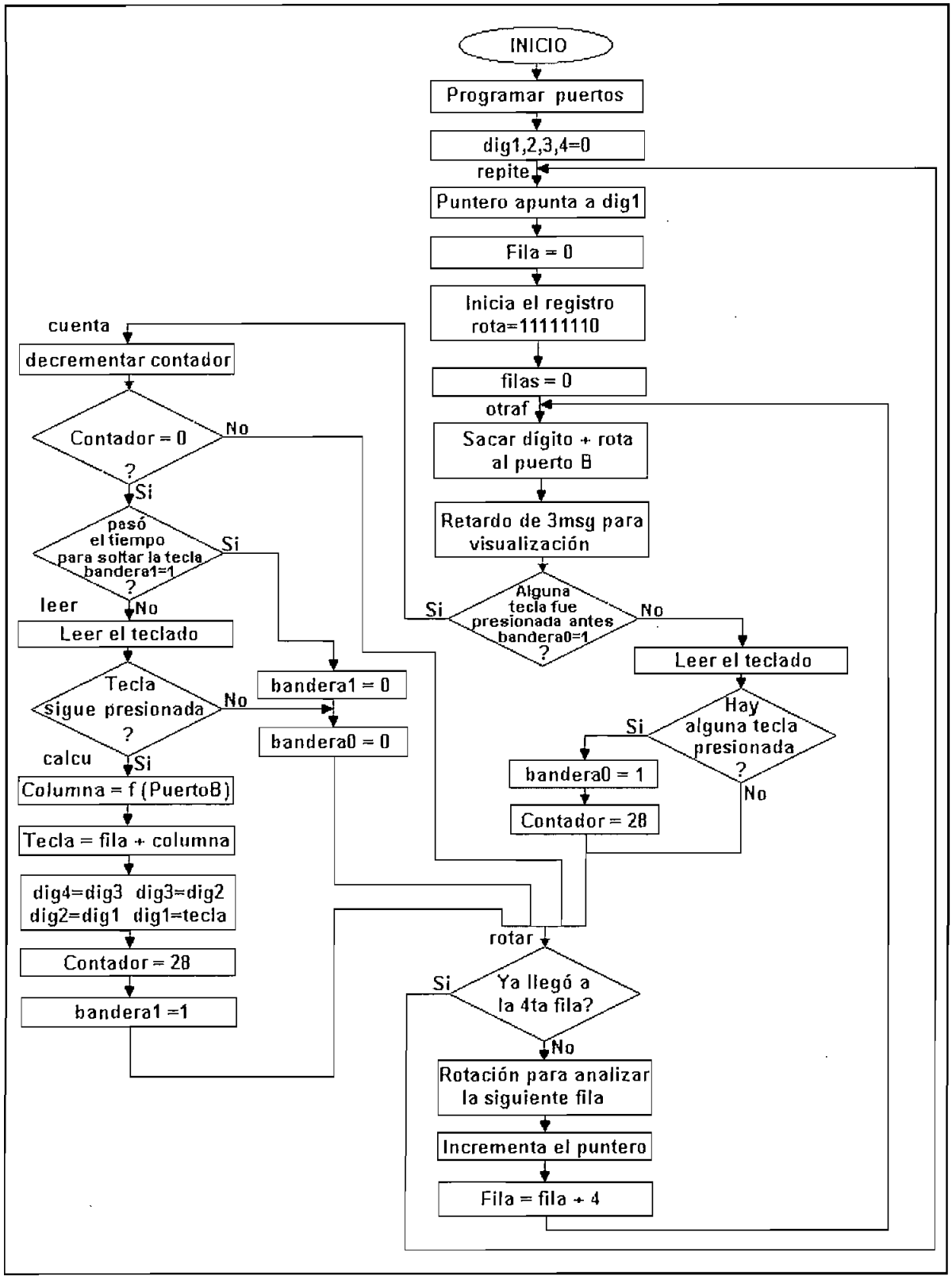


Figura 4.22 Diagrama de flujo del multiplexaje de teclado y displays

```
;Este programa realiza la lectura de un teclado hexadecimal de 16 teclas
;y maneja 4 displays, en los cuales ingresa desde la dercha el valor de
;las teclas presionadas utilizando la técnica de multiplexaje.
        list    p=16F84A        ;Microcontrolador utilizado
;Etiquetas
indo    equ     00h             ;registro de direccionamiento indirecto
pc      equ     02h             ;registro contador de programa
status  equ     03h             ;registro de estados
Fsr     equ     04h             ;registro selector de registros (puntero)
porta   equ     05h             ;el puerto A está en la dirección 05 de la RAM
portb   equ     06h             ;el puerto B está en la dirección 06 de la RAM
lazo1   equ     0ch             ;registro para conteo en la subrutina retardo
lazo2   equ     0dh             ;registro para conteo en la subrutina retardo
rota    equ     0eh             ;registro que lleva un 1 para activar displays
tecla   equ     0Fh             ;contiene el valor de la tecla presionada
fila    equ     10h             ;contiene el valor de la fila a analizar
dig1    equ     11h             ;contiene el valor a mostrar en el display1
dig2    equ     12h             ;contiene el valor a mostrar en el display2
dig3    equ     13h             ;contiene el valor a mostrar en el display3
dig4    equ     14h             ;contiene el valor a mostrar en el display4
bandera equ     15h             ;registro de banderas
conta   equ     16h             ;registro contador
digito  equ     17h             ;registro que almacena el dígito a mostrar
trisa   equ     85h             ;registro de configuración del puerto A
trisb   equ     86h             ;registro de configuración del puerto 8
w       equ     00h             ;indica que el resultado se guarda en W
r       equ     01h             ;el resultado se guarda en el registro
z       equ     02h             ;bandera de cero del registro de estados
c       equ     00h             ;bandera de carry del registro de estados
ban0    equ     00h             ;bandera que indica que hubo una tecla presionada
ban1    equ     01h             ;bandera que indica tiempo a que suelte la tecla
reset   org     0               ;se escribe en la línea 0 la instrucción
        goto    inicio          ;de salto a la línea etiquetada con inicio

        org     5               ;las sigui. líneas se escribirán desde la dir. 5

;**************Subrutina de retardo de 3 milisegundos********************
retardo movlw   D'3'    ;El registro lazo1 contiene el número de
        movwf   lazo1   ;veces que repite 1 milisegundo
repite1 movlw   D'110'  ;El registro lazo2 contiene el número de
        movwf   lazo2   ;veces que repite los 9 microsegundos
repite2 nop             ;de retardo generados
        nop             ;por los 6 ciclos de las instrucciones nop (6usg)
        nop             ;más 1 ciclo de la instrucción decfsc (1usg)
        nop             ;más 2 ciclos del salto goto (2usg)
        nop             ;dando en total los 9usg, siendo esta la base
        nop             ;de tiempo, por lo tanto 1sg= 9usg*110*100*10
        decfsz  lazo2   ;decrementa el reg lazo2 y salta si llega a 0
        goto    repite2 ;si lazo2 no es 0 entonces salta a repite2
        decfsz  lazo1   ;decrementa el reg lazo1 y salta si llega a 0
        goto    repite1 ;si lazo1 no es 0 entonces salta a repite
        retlw   0       ;salida de la subrutina cargando w con 0

;************************ tabla para hallar el valor de columna **********
tabla   addwf   pc      ;suma W al PC
        nop
        retlw   0       ;columna = 0
        retlw   1       ;columna = 1
        nop
        retlw   2       ;columna = 2
        retlw   2       ;columna = 2
        nop
        nop
        nop
        retlw   3       ;columna = 3
```

```
;******************* Programa principal *********************************
inicio  bsf     status,5        ;se ubica en el segundo banco de la RAM
        movlw   0FFh            ;se caraga el registro w con 00h
        movwf   trisa           ;se programa el puerto A como entradas
        movlw   00h             ;se caraga el registro w con 0Fh
        movwf   trisb           ;se programa el puerto B como salidas
        bcf     status,5        ;se ubica en el primer banco de la RAM
        clrf    tecla           ;borrado inicia del registro teclas
        clrf    dig1            ;borrado inicial del dígito1
        clrf    dig2            ;borrado inicial del dígito2
        clrf    dig3            ;borrado inicial del dígito3
        clrf    dig4            ;borrado inicial del dígito4
repite  clrf    fila            ;cargar al registro fila con cero
        movlw   B'11111110'     ;rota con cero ubicado para analizar fila1
        movwf   rota            ;y encender el dígito 1
        movlw   dig1            ;con el registro selector se apunta
        movwf   fsr             ;al dígito1 a mostrar en el display1
otraf   movf    indo,w          ;lee el dato apuntado por el fsr
        movwf   digito          ;carga el dato en el registro dígito
        swapf   digito,w        ;intercambio de los 4 bits bajos por los altos
        iorlw   0fh             ;pone en unos los cuatro bits bajos
        andwf   rota,w          ;une la parte baja de rota con la alta de digt
        movwf   portb           ;y saca esta unión por el puerto B
        call    retardo         ;llama al retardo de 3 milisegundos
        btfsc   bandera,ban0    ;pregunta hubo una tecla presionada antes
        goto    cuenta          ;salta a contar 84msg
        movf    porta,w         ;lee la fila del teclado
        andlw   0fh             ;elimina la parte alta del byte leido
        xorlw   0fh             ;se invierte la parte baja del byte leido
        btfsc   status,z        ;pregunta si el resultado es cero
        goto    rotar           ;sala a rotación
cuenta  decfsz  conta           ;decrementa el contador y salta si es cero
        goto    rotar           ;no es cero salta a rotar
        btfss   bandera,ban1    ;estubo temporizando para que suelte tecla?
        goto    leer            ;salta a leer la bandera
        bcf     bandera,ban1    ;baja la bandera1
        goto    bande0          ;salta a bande0
leer    movf    porta,w         ;lee la fila del teclado
        andlw   0fh             ;elimina la parte alta del byte leido
        xorlw   0fh             ;se invierte la parte baja del byte leido
        btfss   status,z        ;pregunta si el resultado es cero
        goto    calcu           ;sala a calcular el valor de la tecla
bande0  bcf     bandera,ban0    ;baja la bandera0
        goto    rotar           ;salta a rotar
calcu   call    tabla           ;llama tabla, para hallar el valor de columna
        addwf   fila,w          ;retorna cargando columna en W
        movwf   tecla           ;valor de la tecla= fila + columna
        movf    dig3,w          ;leo el valor de digito3
        movwf   dig4            ;y lo cargo en digito4
        movf    dig2,w          ;leo el valor de digito2
        movwf   dig3            ;y lo cargo en digito3
        movf    dig1,w          ;leo el valor de digito1
        movwf   dig2            ;y lo cargo en digito2
        movf    tecla,w         ;leo el valor de tecla
        movwf   dig1            ;y lo cargo en digito1
        movlw   D'28'           ;cargo el contador con 28
        movwf   conta           ;para contar 84msg a que suelte la tecla
        bsf     bandera,ban1    ;activo la bandera1
rotar   btfss   rota,3          ;pregunto si ya llegó a la cuarta fila
        goto    repite          ;si llegó: salta a repetir en la fila 1
        bsf     status,c        ;coloca el bit del carry en 1 para rotar
        rlf     rota            ;a la izquierda el cero que va a las filas
        movlw   4               ;carga a W con 4 para
        addwf   fila,r          ;sumarlo al registro fila (fila=fila+4)
        incf    fsr             ;incrementa el puntero
        goto    otraf           ;salta analizar otra fila
        end                     ;final del programa
```

Figura 4.23 Programa de multiplexaje de teclados y displays

4.1.4 MANEJO DEL MÓDULO LCD

Práctica 8: Manejo del módulo LCD con interface a 8 bits

Los módulos LCD permiten visualizar en su pantalla mensajes alfanuméricos y ASCII, lo que representa una gran ventaja frente a los displays. Además, su manejo es mucho más sencillo ya que son diseñados para conectarlos a microcontroladores, sin la necesidad de hacer multiplexación, ni tablas de caracteres a mostrar. Otra ventaja es un menor consumo de corriente.

Como ya se vio en el capítulo anterior la configuración de los terminales del módulo LCD, entonces se indica la forma de manejarlos. Según la operación a realizar los pines de control E, RS Y R/W deben tener un estado determinado y en el bus de datos debe tener un valor que indique el carácter a mostrar en la pantalla o una instrucción de control. En la Figura 4.24 se observan los tiempos requeridos para el manejo del LCD.

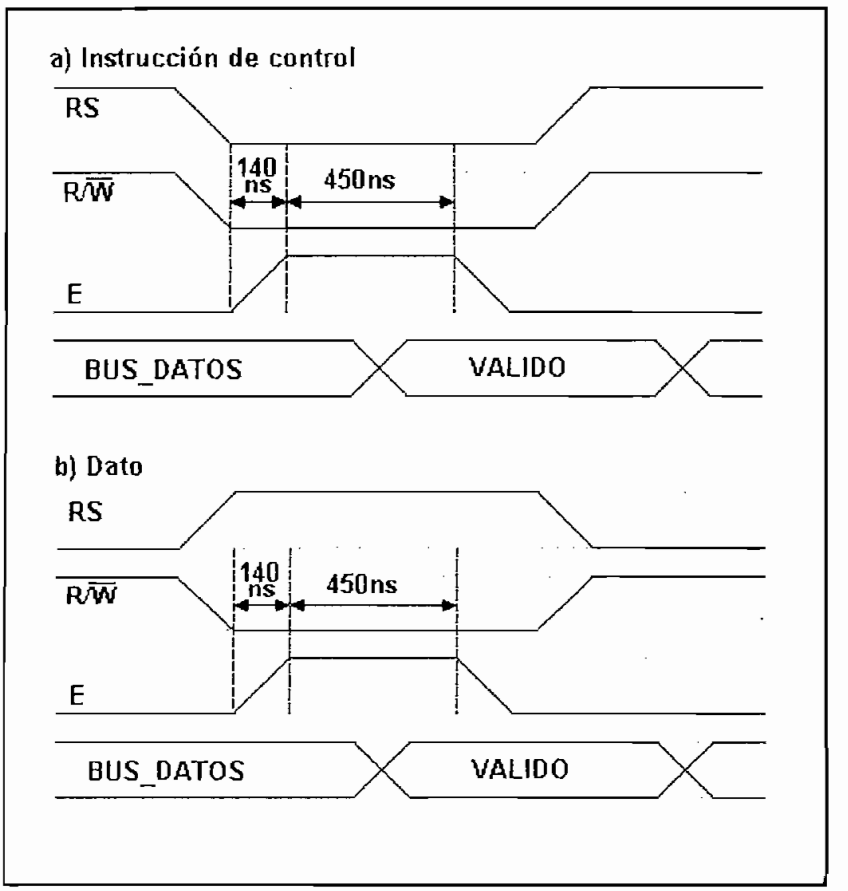


Figura 4.24 Diagrama de tiempos del módulo LCD

El módulo LCD responde a un conjunto especial de instrucciones, estas deben ser enviadas del microcontrolador al módulo, según la operación que se requiera. En la Figura 4.25 se pueden observar estas instrucciones.

Contro y dato / INSTRUCCIONES	Señal de control		DATO/DIRECCIÓN								DESCRIPCIÓN
	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	
Borrar pantalla	0	0	0	0	0	0	0	0	0	1	Limpia todo display y retorna el cursor a la posición de inicio (dirección 0)
Cursor a casa	0	0	0	0	0	0	0	0	1	*	Retorna el cursor a la posición de inicio (dirección 0). También retorna el display, desplazandolo a la posición origina. Los contenidos de la RAM DD permanecen sin cambio.
Seleccionar modo	0	0	0	0	0	0	0	1	RL	S	Configura la dirección de movimiento del cursor si se desplaza o no el display. Esta operación es realizada durante operaciones de lectura escritura
Encender/Apagar pantalla	0	0	0	0	0	0	1	D	C	B	Configura el estado ON/OFF de todo el display (D), el cursor (C) y el parpadeo del caracter en la posición del cursor.
Desplazar Cursor/Pantalla	0	0	0	0	0	1	S/C	R/L	*	*	Mueve el cursor y desplaza el display sin cambiar los contenidos de la RAM DD.
Activar función	0	0	0	0	1	D/L	N	F	*	*	Confugura el tamaño de la Interface (DL), el número de líneas del display (N) y la fuente del caracter (F). N=0, 1 línea N=1, 2 líneas
CG RAM	0	0	0	1	Dirección generador de RAM						Ajusta la dirección del generador de caracteres. El dato CG RAM es enviado y recibido después de este ajuste.
DD RAM	0	0	1	Dirección de datos RAM							Ajusta la direcoión de la RAM DD. La dirección es enviada y recibida despues de este ajuste.
Bandera de ocupado	0	0	BF	AC							Lectura de la bandera Busy Flag. Indicando que operaciones internas son realizadas y lectura de los contenidos del contador de direcciones.
Escritura CG RAM/DD RAM	1	0	Escritura de dato								Escribe datos en la RAM DD o en la RAM CG.
LECTURA CG RAM/DD RAM	1	1	Lectura de Dato								Lectura de datos desde la RAM DD o la RAM CG.

I/D = 1 Incrementa
= 0 Decrementa
S = 1 Desplaza el mensaje en la pantalla
= 0 Mensaje fijo en la pantalla
D = 1 Encender (activar) la pantalla
= 0 Apagar la pantalla (desactivar)

C = 1 Activa cursor
= 0 Desactiva el cursor
B = 1 Parpadea caracterseñalado por el cursor
= 0 No parpadea el caracter
S/C = 1 Desplaza pantalla
= 0 Mueve el cursor

RL = 1 Desplazamiento a la derecha
= 0 Desplazamiento a la izquierda
DL = 1 Datos de 8 bits
= 0 Datos de 4 bits
BF = 1 Durante operación interna del módulo
= 0 Finalizada la operación interna

Figura 4.25 Instrucciones del módulo LCD

Los caracteres que llegan al display se almacenan en la RAM del módulo. Existen posiciones de memoria RAM, cuyos datos se observan en la pantalla y otras que no, las cuales se utilizan para almacenar caracteres que luego se desplazan hacia la parte visible. En la Figura 4.26 se puede observar las dos líneas de caracteres del módulo.

En los anexos adjunto a esta tesis se puede observar la tabla de caracteres ASCII que acepta este display.

La interface entre el microcontrolador y el display de cristal líquido se puede hacer con el bus de datos trabajando a 4 u 8 bits.

Posiciones de memória de la primera fila																	
00H	01H	02H	03H	04H	05H	06H	07H	08H	09H	0AH	0BH	0CH	0DH	0EH	0FH	10H	1FH
	R	E	G	I	O	N		V	I	S	I	B	L	E		Area no visible	
20H	52H	45H	47H	49H	4FH	4EH	20H	56H	49H	53H	49H	42H	4CH	45H	20H		
Códigos ASCII correspondientes a los caracteres																	

Posiciones de memória de la segunda fila																	
40H	41H	42H	43H	44H	45H	46H	47H	48H	49H	4AH	4BH	4CH	4DH	4EH	4FH	50H	5FH
			T	E	S	I	S			E	P	N				Area no visible	
20H	20H	20H	54H	45H	53H	49H	53H	20H	20H	45H	50H	4EH	20H	20H	20H		
Códigos ASCII correspondientes a los caracteres																	

Figura 4.26 Mapa de memoria del módulo LCD

Las señales de control funcionan de la misma forma en ambos casos, al iniciar el sistema se debe establecer la configuración mediante una instrucción adecuada para esto.

Esta práctica consiste en mostrar en el display un texto almacenado en la memoria de programa del microcontrolador utilizando la interface a 8 bits, el cual circula en la línea superior. En la Figura 4.27 se puede observar el circuito a implementar.

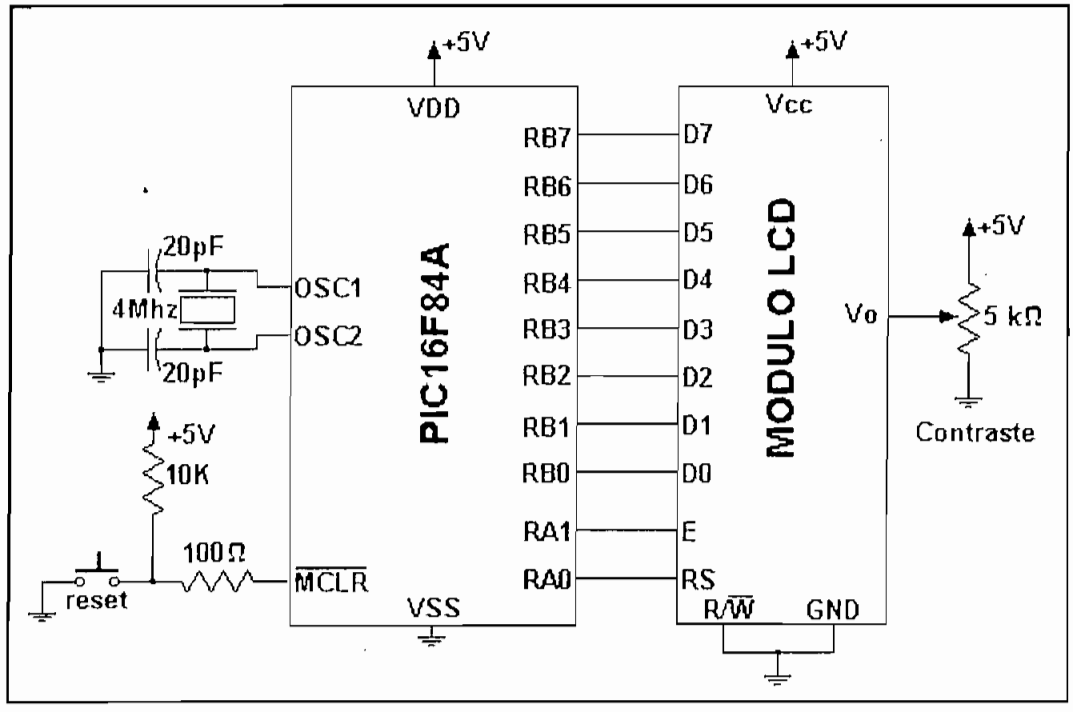


Figura 4.27 Circuito de manejo de un display con interface a 8 bits

En este caso como la operación es solo de escritura, el pin de selección de lectura escritura (R/W) en el display, se conecta a *tierra*. Desde el microcontrolador se envían los datos a 8 bits por el puerto B y se controlan los

pines E y RS mediante RA1 y RA0, para enviar las señales de control. Todas las instrucciones de control se pueden deducir de la Figura 4.25 de la cual se obtiene el valor necesario para configurar el módulo LCD y presentar los caracteres en la forma deseada. Para esta práctica se debe primero enviar las señales de control indicando que la comunicación es a 8 bits y que se empleará solo una línea de caracteres. Utilizando la información de la Figura 4.25 en la fila de activar función, se obtiene el valor 30H. Luego, para indicar que el mensaje se va a desplazar en la pantalla, se debe enviar la instrucción de seleccionar modo obteniéndose 07H, indicando que el mensaje se desplaza hacia la derecha. Luego de configurar esta función, se debe encender el display con la instrucción de encender / apagar la pantalla, enviando el dato 0CH. Una vez encendido el módulo se deben enviar los datos almacenados en una tabla, este proceso se repite continuamente. Las subrutinas control y dato, se encargan de generar las señales y los tiempos necesarios para establecer una comunicación correcta. Los diagramas de flujo, y el programa para esta práctica se pueden observar en las Figuras 4.28 y 4.29 ,respectivamente.

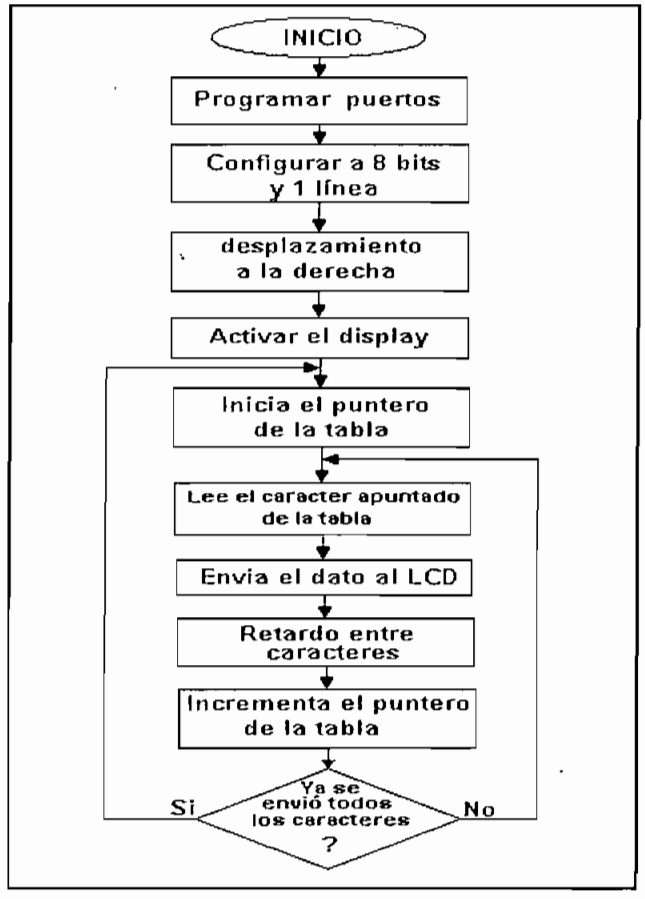


Figura 4.28 Diagrama de flujo para manejo de LCD

```
;Este programa realiza la escritura de un mensaje en el modulo LCD
;el cual se desplaza sobre la primera línea hacia la derecha
        list    p=16F84A        ;Microcontrolador utilizado
;Etiquetas
pc      equ     02h             ;registro contador de programa
status  equ     03h             ;registro de estados
porta   equ     05h             ;el puerto A está en la dirección 05 de la RAM
portb   equ     06h             ;el puerto B está en la dirección 06 de la RAM
lazo1   equ     0ch             ;registro para conteo en la subrutina retardo
lazo2   equ     0dh             ;registro para conteo en la subrutina retardo
puntero equ     0eh             ;registro que lleva un 1 para activar displays
trisa   equ     85h             ;registro de configuración del puerto A
trisb   equ     86h             ;registro de configuración del puerto B
w       equ     00h             ;indica que el resultado se guarda en W
r       equ     01h             ;el resultado se guarda en el registro
z       equ     02h             ;bandera de cero del registro de estados
e       equ     01h             ;bandera que indica que hubo una tecla presionada
rs      equ     00h             ;bandera que indica tiempo a que suelte la tecla

reset   org     0               ;se escribe en la línea 0 la instrucción
        goto    inicio          ;de salto a la línea etiquetada con inicio

        org     5               ;las sigui. líneas se escribirán desde la dir. 5

;************** Subrutina de retardo de 765usg ********************
retardo movlw   0ffh            ;Esta subrutina carga en
        movwf   lazo1           ;lazo1 255, generando un
lazo    decfsz  lazo1           ;retardo de 255 * 3usg
        goto    lazo            ;por las instruc. decfsz y goto
        retlw   0

;************** Subrutina de control y datos ***********************
control bcf     porta,rs        ;pone en cero a la línea RS (señal de control)
        goto    sigue
dato    bsf     porta,rs        ;pone en uno a la línea RS (señal de dato)
sigue   bsf     porta,e         ;habilita el módulo LCD (E=1)
        movwf   portb           ;carga la bus de datos
        call    retardo         ;genera un retardo de 765usg
        bcf     porta,e         ;Deshabilita el módulo LCD (E=0)
        call    retardo         ;genera un retardo para que tome el dato
        retlw   0

;************** tabla con el mensaje a desplazar *********************
tabla   addwf   pc      ;suma W al PC
        retlw   'E'     ;mensaje  a ser mostrado
        retlw   'S'
        retlw   'C'
        retlw   'U'
        retlw   'E'
        retlw   'L'
        retlw   'A'
        retlw   ' '
        retlw   'P'
        retlw   'O'
        retlw   'L'
        retlw   'I'
        retlw   'T'
        retlw   'E'
        retlw   'C'
        retlw   'N'
        retlw   'I'
        retlw   'C'
        retlw   'A'
        retlw   ' '
        retlw   'N'
        retlw   'A'
        retlw   'C'
        retlw   'I'
        retlw   'O'
        retlw   'N'
        retlw   'A'
        retlw   'L'
        retlw   ' '
        retlw   'S'
        retlw   'I'
        retlw   'S'
        retlw   'T'
        retlw   'E'
        retlw   'M'
        retlw   'A'
```

```
        retlw   ' '
        retlw   'D'
        retlw   'E'
        retlw   ' '
        retlw   'D'
        retlw   'E'
        retlw   'S'
        retlw   'A'
        retlw   'R'
        retlw   'R'
        retlw   'O'
        retlw   'L'
        retlw   'L'
        retlw   'O'
        retlw   ' '
        retlw   'P'
        retlw   'A'
        retlw   'R'
        retlw   'A'
        retlw   ' '
        retlw   'L'
        retlw   'O'
        retlw   'S'
        retlw   ' '
        retlw   'M'
        retlw   'I'
        retlw   'C'
        retlw   'R'
        retlw   'O'
        retlw   'C'
        retlw   'O'
        retlw   'N'
        retlw   'T'
        retlw   'R'
        retlw   'O'
        retlw   'L'
        retlw   'A'
        retlw   'D'
        retlw   'O'
        retlw   'R'
        retlw   'E'
        retlw   'S'
        retlw   ' '
        retlw   'P'
        retlw   'I'
        retlw   'C'
        retlw   ' '
        retlw   ' '
        retlw   ' '
        retlw   0
;******************  Programa principal  ********************************
inicio  bsf     status,5        ;se ubica en el segundo banco de la RAM
        movlw   0fch            ;ra0 y ra1 como salidas
        movwf   trisa           ;se programa el puerto A como salidas
        movlw   00h             ;se caraga el registro w con 00h
        movwf   trisb           ;se programa el puerto B como salidas
        bcf     status,5        ;se ubica en el primer banco de la RAM
        movlw   30h             ;configura el display a 8 bits y 1 línea
        call    control
        movlw   07h             ;selecciona modo de desplazamiento a la derecha
        call    control
        movlw   0ch             ;activa el disolau
        call    control
otravez clrf    puntero         ;inicia el puntero en cero
ciclo   movf    puntero,w       ;y lo carga en w para llamar a la tabla
        call    tabla           ;de la que regresa cargado el caracter en
        call    dato            ;w para enviarlo con la subrutina dato
        movlw   D'159'          ;inicia el lazo2 con 159
        movwf   lazo2           ;para generar un lazo de retardo
repite  call    retardo         ;de 247.27msg = 159 * (765+765)
        call    retardo
        decfsz  lazo2
        goto    repite
        incf    puntero         ;incrementa el puntero de la tabla
        movlw   D'85'           ;preguta si ya llegó al valor maximo
        xorwf   puntero,w       ;de la tabla para entonces volver a
        btfss   status,z        ;iniciar en envio del mensaje
        goto    ciclo           ;aún no termina
        goto    otravez         ;ya terminó
        end                     ;final del programa
```

Figura 4.29 Programa de manejo de LCD

Práctica 9: Manejo del módulo LCD con interface a 4 bits

Una alternativa válida para reducir el número de pines necesarios para manejar el módulo LCD es la interface a 4 bits, liberando así 4 pines que pueden ser utilizados para otros fines. En este caso, se debe enviar primero los cuatro bits altos del dato y luego los cuatro bits bajos, para lo cual, al enviar cada uno de los datos de 4 bits, las señales de control deben comportarse de la misma manera como si se tratara de un dato completo de 8 bits.

Como se requieren menos pines, entonces se utilizará solamente 6 pines del puerto B como se observa en la Figura 4.30, y se realizará un programa que permita mostrar un mensaje en el display, utilizando los dos líneas. Este mensaje se mantendrá fijo en la pantalla.

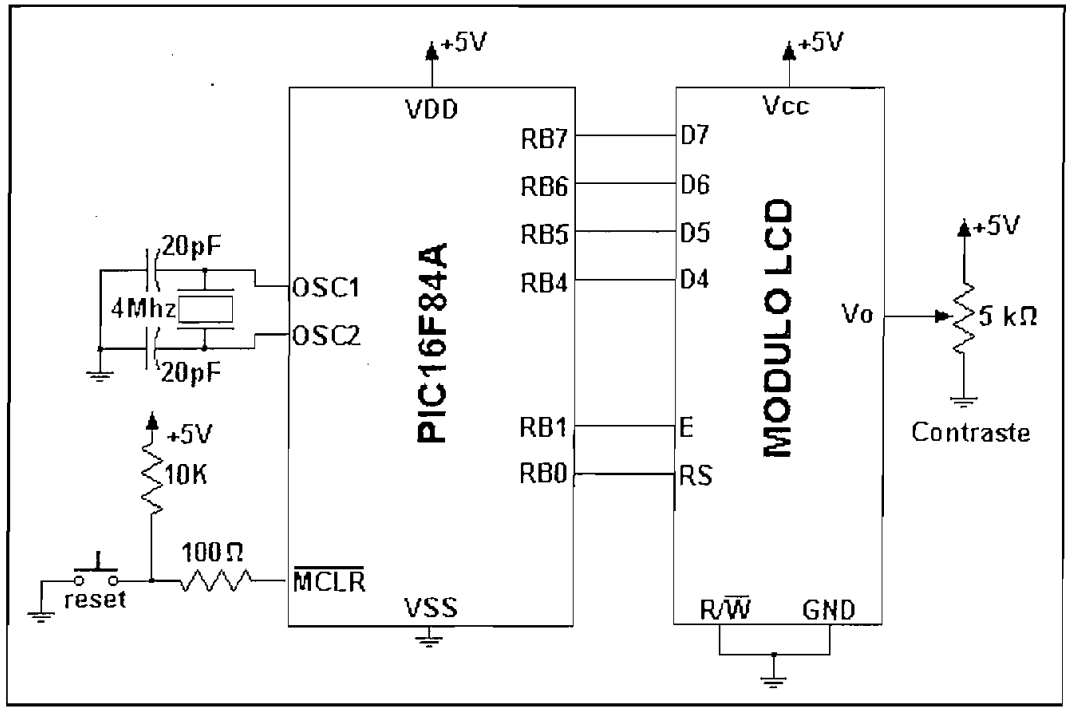


Figura 4.30 Circuito de manejo de un display con interface a 4 bits

Para esta práctica se debe iniciar el módulo LCD indicándole que la comunicación es a 4 bits, esto se lo hace enviando el dato 02H. Luego de esto, para indicar que se utilizarán las dos líneas del display, se envía el dato 28H. Posterior a esto se debe encender el display enviando el dato 0CH y por último, para indicar que el mensaje permanece fijo en la pantalla, se envía 06H. Una vez

configurado el display, se debe proceder a enviar los datos, haciendo previamente un borrado de la pantalla. Los datos son obtenidos de una tabla y para transmitirlos se utilizan las subrutinas dato y control, las cuales envían primero los cuatro bits altos y luego los cuatro bits bajos, generando también los tiempos y las señales de control en los pines RS y E necesarios.

A continuación se puede observar el diagrama de flujo y el programa necesario para esta práctica en las Figuras 4.31 y 4.32, respectivamente.

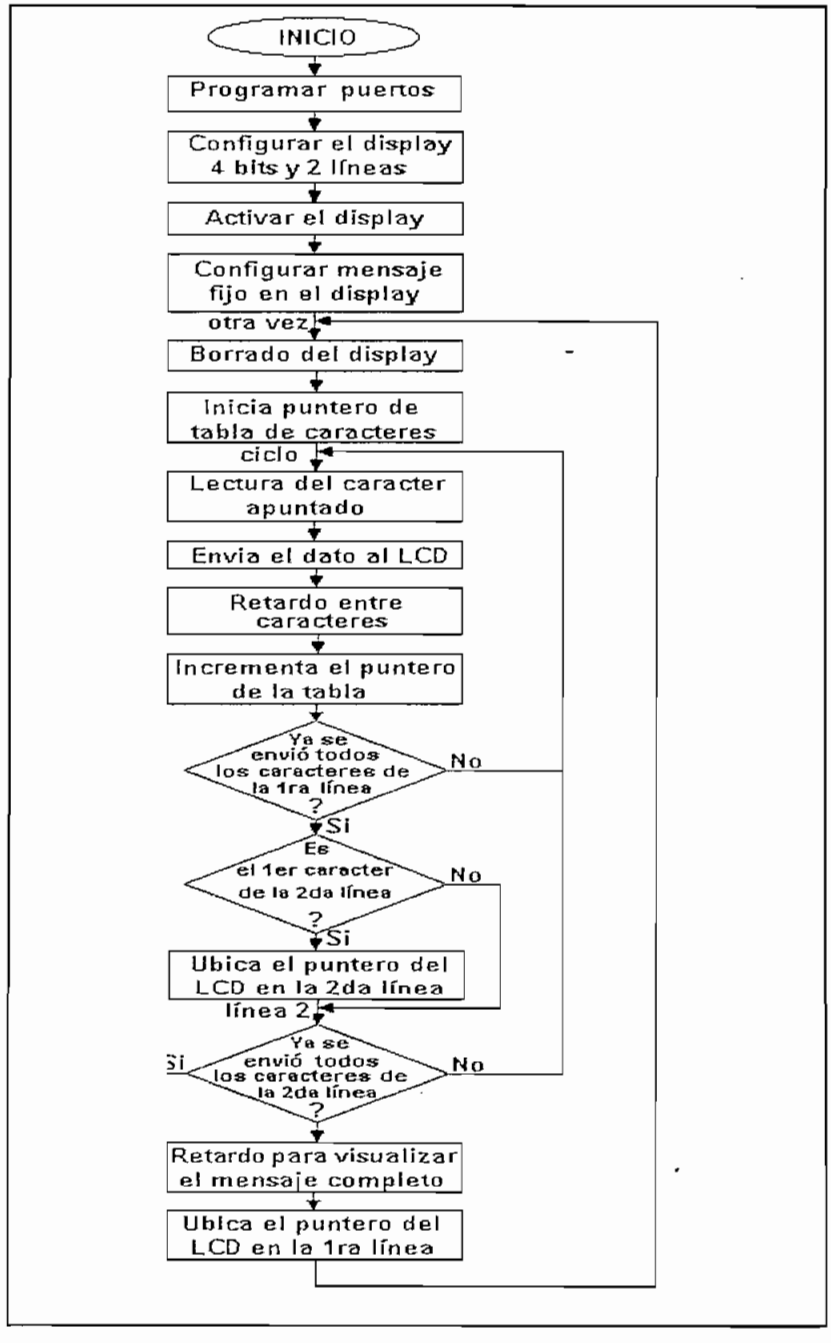


Figura 4.31 Diagrama de flujo para interface a 4 bits del módulo LCD

```
;Este programa realiza la escritura de un mensaje en el modulo LCD
;sobre sus dos líneas, el cual permanece fijo en la pantalla y se
;repite indefinidamente

        list    p=16F84A        ;Microcontrolador utilizado
;Etiquetas
pc      equ     02h             ;registro contador de programa
status  equ     03h             ;registro de estados
porta   equ     05h             ;el puerto A está en la dirección 05 de la RAM
portb   equ     06h             ;el puerto B está en la dirección 06 de la RAM
lazo1   equ     0ch             ;registro para conteo en la subrutina retardo
lazo2   equ     0dh             ;registro para conteo en la subrutina retardo
lazo3   equ     0eh             ;registro que lleva un 1 para activar displays
puntero equ     0fh             ;registro que lleva un 1 para activar displays
caract  equ     10h             ;registro temporal del caracter a mostrar
trisa   equ     85h             ;registro de configuración del puerto A
trisb   equ     86h             ;registro de configuración del puerto B
w       equ     00h             ;indica que el resultado se guarda en W
r       equ     01h             ;el resultado se guarda en el registro
z       equ     02h             ;bandera de cero del registro de estados
e       equ     01h             ;bandera que indica que hubo una tecla presionada
rs      equ     00h             ;bandera que indica tiempo a que suelte la tecla
c       equ     00h             ;bandera de carry
reset   org     0               ;se escribe en la línea 0 la instrucción
        goto    inicio          ;de salto a la línea etiquetada con inicio

        org     5               ;las sigui. líneas se escribirán desde la dir. 5

;*************** Subrutina de retardo de 765usg ********************
retardo movlw   0ffh            ;Esta subrutina carga en
        movwf   lazo1           ;lazo1 255, generando un
lazo    decfsz  lazo1           ;retardo de 255 * 3usg
        goto    lazo            ;por las instruc. decfsz y goto
        retlw   0

;*************** Subrutina de retardo de 195msg ********************
retard1 movlw   07fh            ;inicia el lazo2 con ffh
        movwf   lazo2           ;para generar un lazo de retardo
repite  call    retardo         ;de 195msg = 256 * (765)
        decfsz  lazo2
        goto    repite
        retlw   0
;*************** Subrutina de control y datos ************************
control bcf     portb,rs        ;pone en cero a la línea RS (señal de control)
        goto    sigue
dato    bsf     portb,rs        ;pone en uno a la línea RS (señal de dato)
sigue   bsf     portb,e         ;habilita el módulo LCD (E=1)
        movwf   caract          ;guarda temporalmente el dato en caract
        movlw   0fh             ;pone en cero la
        andwf   portb,r         ;parte alta del puerto B
        movf    caract,w        ;filtra la parte alta del
        andlw   0f0h            ;caracter a mostrar y lo
        iorwf   portb,r         ;une con la parte baja del puerto B
        call    retardo         ;genera un retardo de 765usg
        bcf     portb,e         ;Deshabilita el módulo LCD (E=0)
        call    retardo         ;genera un retardo para que tome el dato
        bsf     portb,e         ;habilita nuevamente el LCD
        movlw   0fh             ;pone en cero la
        andwf   portb,r         ;parte alta del puerto B
        swapf   caract,w        ;intercambio de los 4 bits bajos por los altos
        andlw   0f0h            ;filtra la parte alta del registro caract
        iorwf   portb,r         ;y lo une con la parte baja del puerto B
        call    retardo         ;genera un retardo de 765usg
        bcf     portb,e         ;deshabilita el módulo LCD
        call    retardo         ;genera un retardo para que tome el dato
        retlw   0               ;retorno de la subrutina

;*************** tabla con el mensaje a desplazar ********************
tabla   addwf   pc      ;suma W al PC
        retlw   ' '     ;mensaje  a ser mostrado
        retlw   'T'
        retlw   'E'
        retlw   'S'
        retlw   'I'
        retlw   'S'
        retlw   ':'
        retlw   ' '
        retlw   'P'
        retlw   'R'
        retlw   'A'
        retlw   'C'
        retlw   'T'
        retlw   'I'
```

```
        retlw   'C'
        retlw   'A'
        retlw   'I'
        retlw   'N'
        retlw   'T'
        retlw   'E'
        retlw   'R'
        retlw   'F'
        retlw   'A'
        retlw   'C'
        retlw   'E'
        retlw   ' '
        retlw   '_'
        retlw   '4'
        retlw   'B'
        retlw   'I'
        retlw   'T'
        retlw   'S'
        retlw   0
;********************  Programa principal  *********************************
inicio  bsf     status,5        ;se ubica en el segundo banco de la RAM
        movlw   0FFh            ;ra0 y ra1 como salidas
        movwf   trisa           ;se programa el puerto A como entradas
        movlw   0Ch             ;se caraga el registro w con 00h
        movwf   trisb           ;puerto B como salidas excepto(Rb2:Rb3)
        bcf     status,5        ;se ubica en el primer banco de la RAM
        call    retard1         ;retardo de 159msg para estabilización del LCD
;configuración del LCD
        movlw   02h             ;inicia el display a 4 bits
        call    control
        movlw   28h             ;configura el display a 4 bits y 2 línea
        call    control
        movlw   0ch             ;activa el display
        call    control
        movlw   06h             ;el mensaje permanece fijo
        call    control
otravez movlw   01h             ;limpia el display
        call    control
        clrf    puntero         ;inicia el puntero en cero
ciclo   movf    puntero,w       ;y lo carga en w para llamar a la tabla
        call    tabla           ;de la que regresa cargado el caracter en
        call    dato            ;w para enviarlo con la subrutina dato
        call    retard1         ;retardo entre caracteres
        incf    puntero         ;incrementa el puntero de la tabla
        movlw   D'16'           ;pregunta si el puntero revasó la
        subwf   puntero,w       ;primera fila de la tabla
        btfss   status,c
        goto    ciclo
        movlw   D'16'           ;pregunta si es el primer caracter de la
        xorwf   puntero,w       ;segunda línea para iniciar el puntero de
        btfss   status,z        ;la ram del módulo LCD en la segunda línea
        goto    linea2          ;no es el primer caracter de la 2da línea
        movlw   0c0h            ;ubicar el puntero de la ram
        call    control         ;del módulo LCD en la segunda linea
linea2  movlw   D'32'           ;pregunta si ya terminó la segunda línea
        xorwf   puntero,w       ;para entonces iniciar el puntero de la
        btfss   status,z        ;ram en la primera línea
        goto    ciclo           ;aún no termina
        movlw   0ah             ;inicia el lazo3 con ah
        movwf   lazo3           ;retardo para visualización del mensajé
repte   call    retard1         ;completo de 1,9sg = 256 * (765)*10
        decfsz  lazo3
        goto    repte
        movlw   80h             ;ubica el puntero de la RAM en la 1ra línea
        call    control
        goto    otravez         ;ya terminó y repite todo de nuevo
        end                     ;final del programa
```

Figura 4.32 Programar para interface a 4 bits del módulo LCD

4.1.5 COMUNICACIÓN SERIAL

Práctica 10: Transmisión serial RS-232

En muchas aplicaciones se requiere transmitir y recibir información entre el microcontrolador y una computadora u otro microcontrolador, esto permite realizar controles computarizados, adquisición de datos, etc.

La comunicación serial a diferencia de la paralela presenta la ventaja de requerir un reducido número de líneas, aunque la velocidad disminuye ya que se transmiten bit a bit, con velocidades de hasta 20Kbits. Este sistema también presenta la ventaja de comunicarse a mayor distancia (15m) que la comunicación paralela.

Existen dos formas de comunicación serial: la síncrona y la asíncrona. Para la transmisión síncrona se requiere una línea para transmitir los datos, y otra para generar los pulsos de reloj que indique cuando un valor es válido, la velocidad de transmisión está dada por la duración del pulso de sincronismo. En la comunicación asincrónica, no se requieren los pulsos de reloj, en la cual se aplica un proceso de conteo de tiempos para tomar los datos. La velocidad está definida a un valor determinado e igual tanto para el transmisor como para el receptor.

En la Figura 4.33 se puede observar el formato de un carácter para transmisión serial asíncrona:

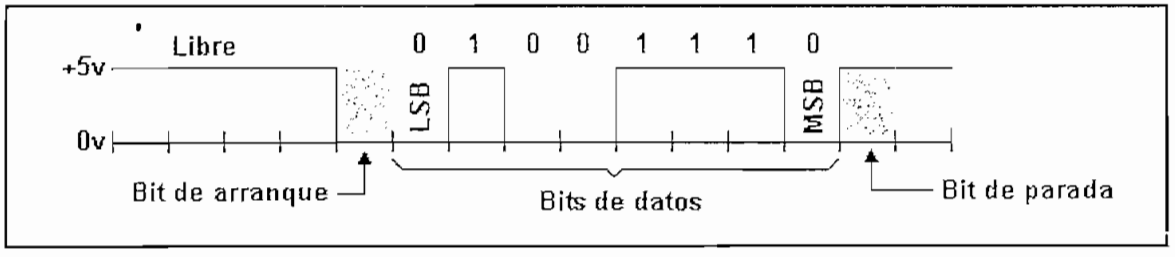


Figura 4.33 Formato de datos para transmisión serial

En esta práctica se estudiará la transmisión asincrónica, en la que la línea de transmisión normalmente se halla en estado de reposo con un nivel lógico alto. Para iniciar la transmisión, se debe enviar un bit de arranque poniendo la

línea de transmisión en un nivel lógico bajo durante el tiempo de un bit, a continuación se envían los bits correspondientes al dato, uno cada vez con un período igual al de un bit, empezando con el bit menos significativo (LSB) hasta el más significativo (MSB). Luego de esto de debe enviar el bit de paridad, si se lo utiliza, y para terminar con uno o dos bits de parada, quedando luego de esto con un estado lógico alto, listo para enviar el siguiente dato.

En este tipo de transmisión, no se tiene una señal de sincronización para saber el inicio de transmisión, entonces para recibir el dato se debe establecer parámetros de velocidad, paridad, número de bits del dato transmitido y de bits de parada, iguales tanto para el transmisor como para el receptor. Entonces el receptor debe estar monitoreando continuamente la línea de transmisión, al momento de detectar un cambio a un nivel bajo debe esperar un tiempo igual a medio bit para leer en el centro del período de duración de un bit, si efectivamente se trata de nivel lógico bajo se interpreta como el bit de arranque; sin embargo, puede tratarse de un ruido en cuyo caso esta lectura no será un nivel bajo. En el caso de confirmar el bit de arranque, se debe proceder a leer los restantes bits, realizando lecturas en períodos de un bit, tomando así el valor del bit en el centro del intervalo de tiempo de cada bit. Los tiempos en los que se realizan las lecturas de datos se puede observar en la Figura 4.34 mostrada a continuación.

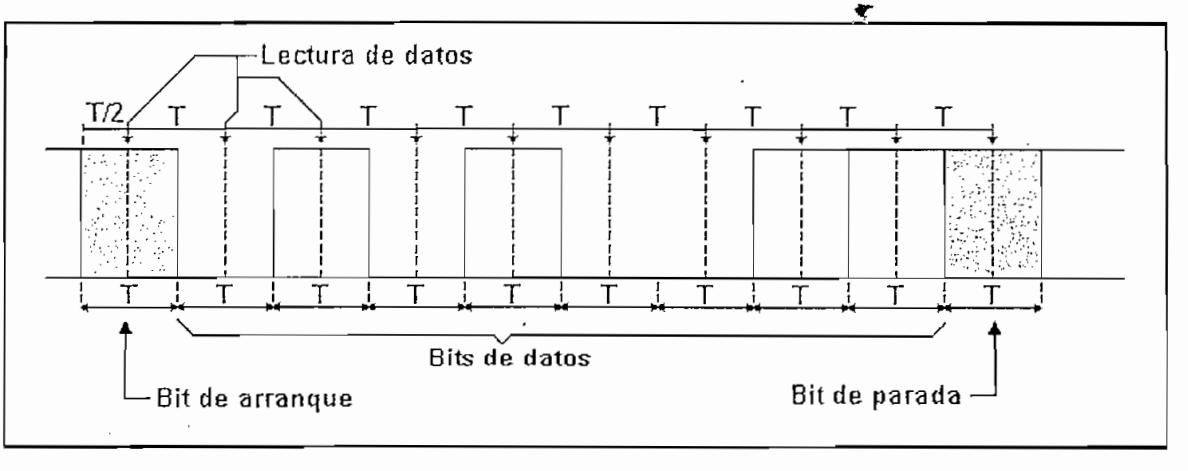


Figura 4.34 períodos de tiempo de lectura de datos

Para realizar la comunicación serial a distancias cortas se pueden utilizar niveles TTL, pero para distancias mayores se debe utilizar circuitos integrados

que permitan aumentar los niveles de voltaje para asegurar fallas por caídas de tensión en los cables y efectos capacitivos.

El circuito más popular para transmisión serial es el MAX232, en el cual un uno lógico es un voltaje comprendido entre –5v. y –15v. en el transmisor y entre –3v. y –25v. en el receptor; en tanto que un cero lógico es un voltaje comprendido entre 5v. y 15v. en el transmisor y entre 3v. y 25v. en el receptor.

Sin embargo existen otras normas como la RS422/485 la cual permite distancias hasta los 1200 metros, velocidades que pueden llegar a los 10Mbits por segundo, pueden trabajar con una fuente de 5v. sin necesidad de fuentes duales y el número de elementos o nodos crece sustancialmente. Para la norma RS-422 pueden estar conectados un transmisor y hasta 10 receptores, mientras que para la norma RS-485 pueden estar conectados simultáneamente hasta 32 transmisores / receptores.

En la práctica presente se desea realizar la transmisión de datos desde el microcontrolador hacia el computador, estos datos serán el resultado de un conteo hexadecimal de 0 a F, el cual es incremetado mediante un pulsador, este valor es mostrado en un display de 7 segmentos a la vez que se lo transmite al computador. En este caso, se realiza solo la transmisión de datos desde el microcontrolador hacia el computador utilizando la línea TX; la configuración de los parámetros de comunicación son los siguientes: velocidad de 1200 bps, datos de 8 bits, sin el bit de paridad y con un bit de parada.

Para recibir los datos en el computador, se puede utilizar el hyperterminal de windows el cual puede ser fácilmente configurado con los mismos parámetros del transmisor, utilizando uno de los puertos seriales existentes en los computadores.

Con el uso del microcontrolador PIC16F84A, es necesario realizar todo el control de la transmisión serial; es decir, establecer las bases de tiempo requeridas, y el envió de los bits del dato uno a uno; sin embargo, existen microcontroladores como los PIC16F87x los cuales poseen puertos de

comunicación serial internos, que permiten transmitir y recibir datos de forma automática.

En la Figura 4.35 se puede observar el circuito de comunicación implementado con el MAX232, en este se utiliza un conector DB9 hembra en el cual se debe hacer puente entre los terminales 7 y 8 y también entre los terminales 1,4 y 6.

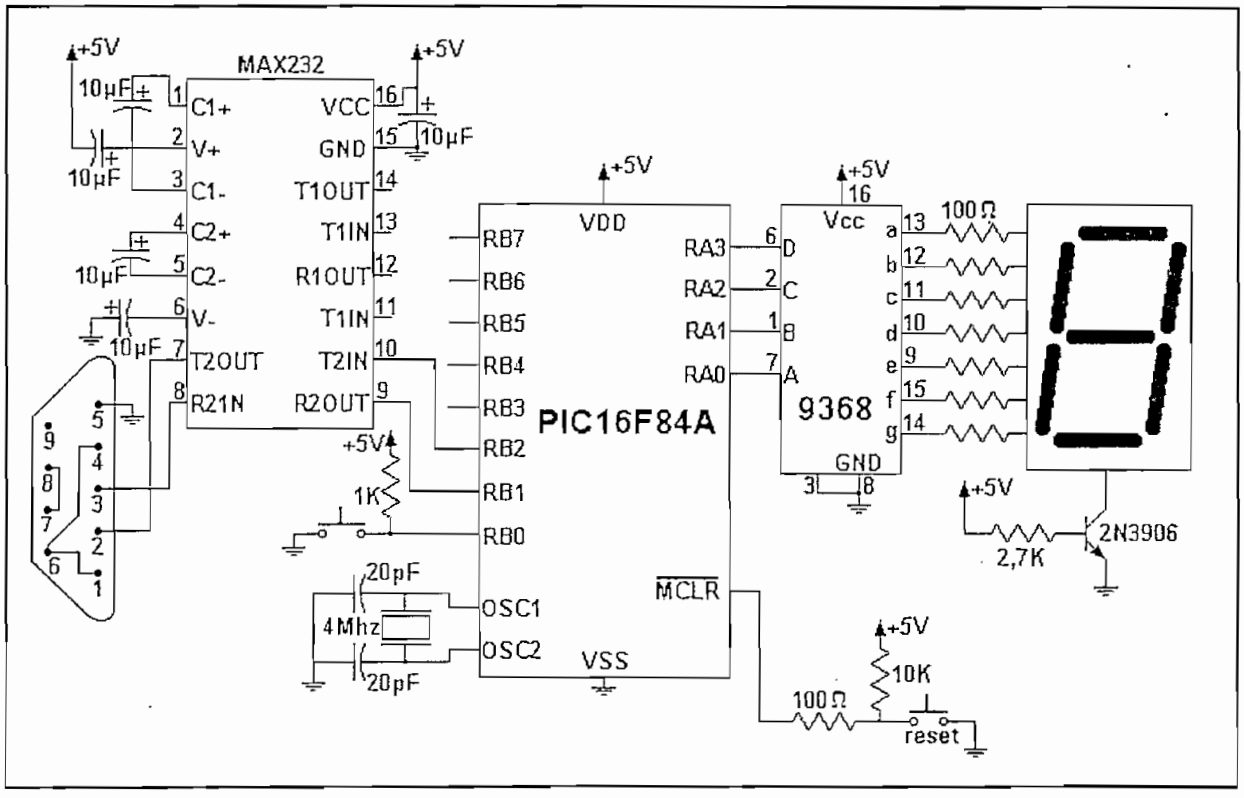


Figura 4.35 Circuito de comunicación serial

En las Figuras 4.36 y 4.37 se puede observar el diagrama de flujo y el programa para esta práctica.

Además en la figura 4.38 se puede observar la pantalla del hyperterminal de windows en la que se han recibido los datos enviados serialmente por el microcontrolador.

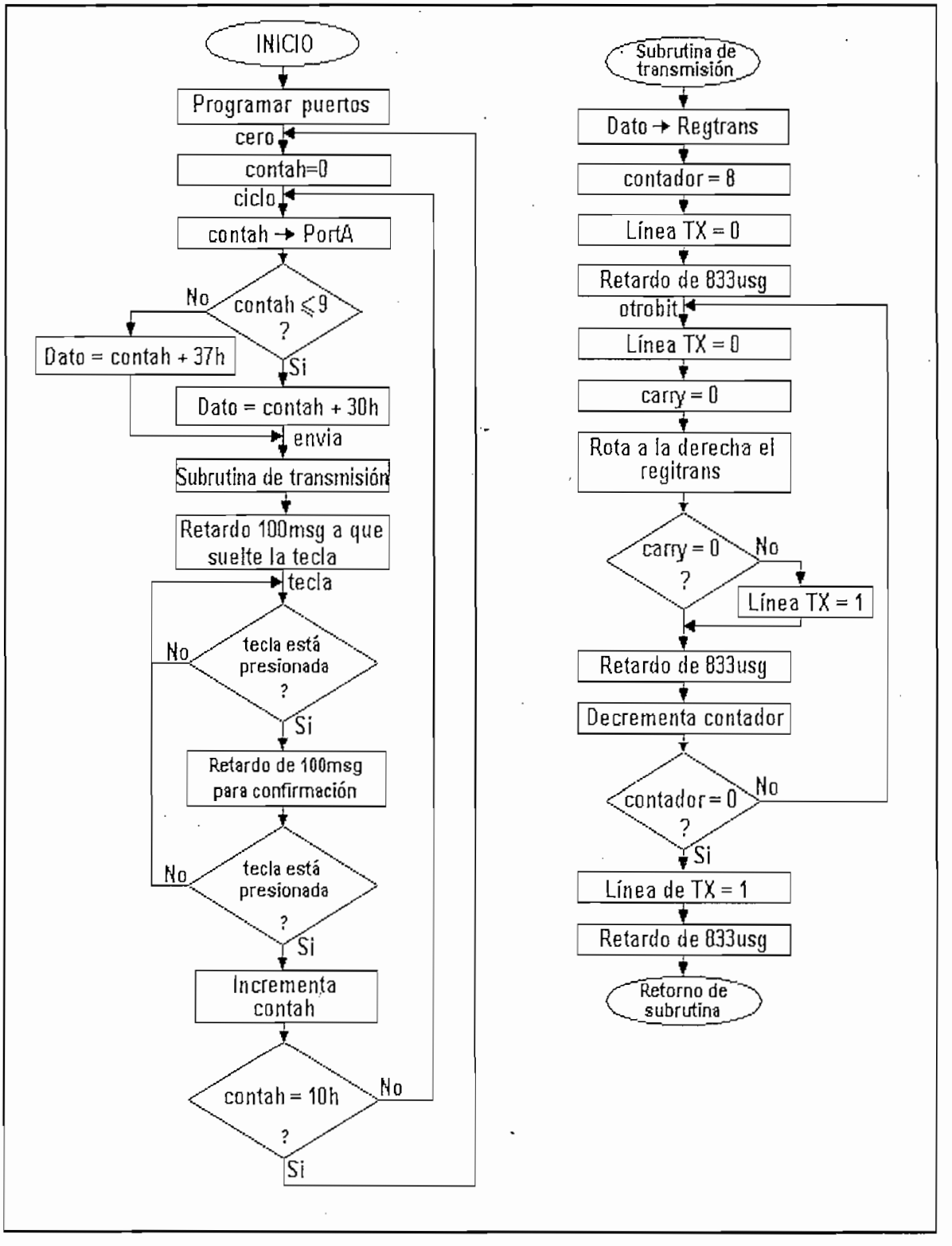


Figura 4.36 Diagrama de flujo para transmisión serial

```
;Esta práctica consiste en un contador hexadecimal de pulsos que
;son enviados al computador mediante comunicación serial RS232
;a 1200 bps, 8 bits de datos, sin paridad y un bit de parada
;Aquí se incluye la librería p16f84a.inc la cual contiene las
;etiquetas de los registros de función especial y sus bits

        List    p=16f84a
        include p16f84a.inc

;Etiquetas de registros de propósito general
regtran equ     0ch             ;registro de transmisión de datos
contab  equ     0dh             ;registro contador de bits
contah  equ     0eh             ;registro de cuenta hexadecimal
lazo    equ     0fh             ;registro para retardo 833usg
lazo1   equ     10h             ;registro para conteo en la subrutina retardo
lazo2   equ     11h             ;registro para conteo en la subrutina retardo
tx      equ     02h             ;bit destinado para transmisión de datos

reset   org     0               ;se escribe en la línea 0 la instrucción
        goto    inicio          ;de salto a la línea etiquetada con inicio

        org     5               ;las sigui. líneas se escribirán desde direc.05h



;***************Subrutina de retardo de 100 milisegundos*********************
retardo movlw   D'100'          ;El registro lazo1 contiene el número de
        movwf   lazo1           ;veces que repite 1 milisegundo
repite1 movlw   D'110'          ;El registro lazo2 contiene el número de
        movwf   lazo2           ;veces que repite los 9 microsegundos
repite2 nop                     ;de retardo generados
        nop                     ;por los 6 ciclos de las instrucciones nop (6usg)
        nop                     ;más 1 ciclo de la instrucción decfsc (1usg)
        nop                     ;más 2 ciclos del salto goto (2usg)
        nop                     ;dando en total los 9usg, siendo esta la base
        nop                     ;de tiempo, por lo tanto 1sg= 9usg*110*100*10
        decfsz  lazo2           ;decrementa el reg lazo2 y salta si llega a 0
        goto    repite2         ;si lazo2 no es 0 entonces salta a repite2
        decfsz  lazo1           ;decrementa el reg lazo1 y salta si llega a 0
        goto    repite1         ;si lazo1 no es 0 entonces salta a repite
        retlw   0               ;salida de la subrutina cargando w con 0

;******************** subrutina de retardo de un bit ************************
unbit   movlw   D'166'          ;Carga al registro lazo con 166
        movwf   lazo            ;para generar un retardo de
vuelta  nop                     ;833usg = 166 * 5usg
        nop                     ;dados por 2 nop + 1 decfsz
        decfsz  lazo            ;+2 del goto = 5 usg
        goto    vuelta
        retlw   0
;******************** subrutina para envio de datos ***************************
enviar  movwf   regtran         ;lleva el contenido de w al registro regtran
        movlw   8               ;carga el número de bits a transmitir
        movwf   contab          ;al registro contador
        bcf     PORTB,tx        ;coloca la línea de transmisión en bajo
        call    unbit           ;generando el bit de arranque
otrobit bcf     PORTB,tx        ;coloca la línea de transmisión en bajo
        bcf     STATUS,C        ;borra el carry para la rotación
        rrf     regtran         ;rota hacia la derecha a través del carry
        btfsc   STATUS,C        ;chequea el bit que ingresó al carry
        bsf     PORTB,tx        ;si es uno coloca la línea en alto
        call .  unbit           ;si es cero se mantiene y llama al retardo
        decfsz  contab          ;decrementa el contador, salta si es cero
        goto    otrobit         ;salta a enviar el siguiente bit
        bsf     PORTB,tx        ;coloca la línea de transmisión en alto
        call    unbit           ;para generar el bit de parada
        retlw   0               ;retorna
```

```
;********************  Programa principal  ********************************
inicio  bsf     STATUS,RP0      ;se ubica en el segundo banco de la RAM
        movlw   00h
        movwf   TRISA           ;se programa el puerto A como salidas
        movlw   0fbh
        movwf   TRISB           ;se prog. el puertoB como salidas, (tx entrada)
        bcf     STATUS,RP0      ;se ubica en el primer banco de la RAM
cero    clrf    contah          ;inicia el contador hexadecimal en cero
ciclo   movf    contah,w        ;el valor del contador pasa al registro W
        movwf   PORTA           ;y lo saca al display
        sublw   9h              ;Para hallar el carracter ASCII
        btfsc   STATUS,C        ;a ser transmitido
        goto    menor           ;se debe suma 30h si el valor está
        movf    contah,w        ;entre 0 y 9, y se debe sumar
        addlw   37h             ;37h si el valor está entre A y F
        goto    envia           ;con la resta veo si es no no mayor a 9
menor   movf    contah,w        ;para sumar el valor adecuado
        addlw   30h             ;le suma 30h para convertirlo en código ASCII
envia   call    enviar          ;envia serialmente el valor del contador
        call    retardo         ;retardo para que suelte la tecla
tecla   btfsc   PORTB,0         ;pregunta si el pulsador está oprimido
        goto    tecla           ;si no está oprimido vuelve a revisar
        call    retardo         ;si está oprimido retarda para confirmar
        btfsc   PORTB,0         ;que la tecla fué presionada
        goto    tecla           ;si no está oprimiendo vuelve a revisar
        incf    contah          ;una vez confirmado incrementa contador
        movf    contah,w        ;compara para ver
        xorlw   10h             ;si el registro conta
        btfsc   STATUS,Z        ;llegó a 10H (sobrepasó a Fh)
        goto    cero            ;si es igual reinicia en cero
        goto    ciclo           ;no llegó a Fh, entonces
        end                     ;continua el ciclo normal
```

Figura 4.37 Programa de transmisión serial

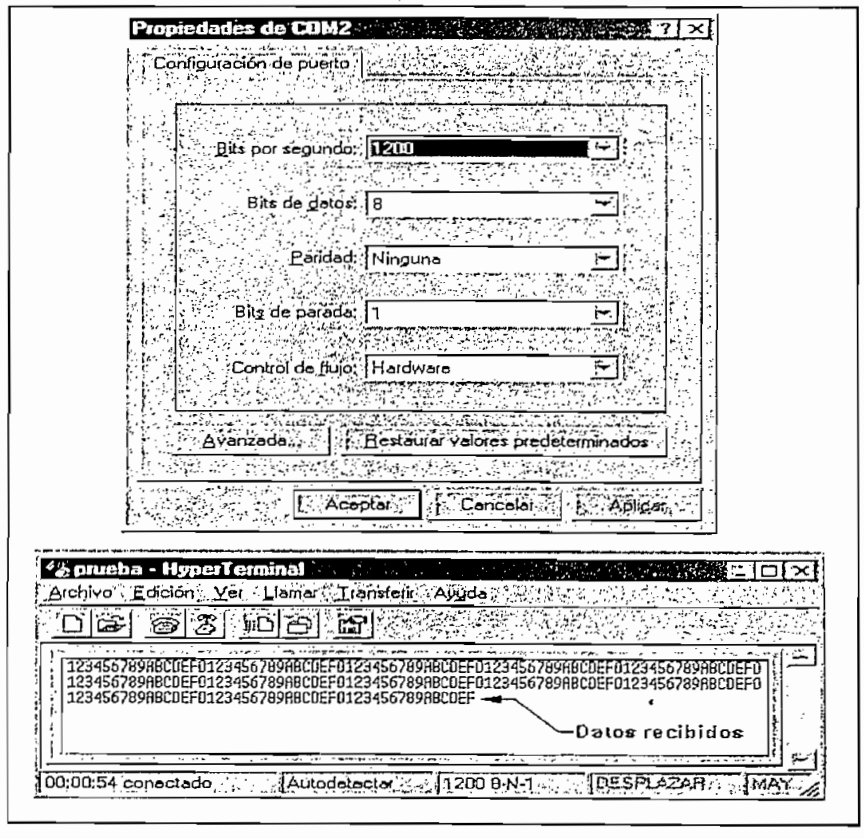


Figura 4.38 Recepción de datos en el Hyperterminal de windows

4.1.6 COMUNICACIÓN I^2C

Práctica 11: Manejo del reloj de tiempo real I^2C DS1307

En muchas aplicaciones se requiere hacer controles o registrar eventos que se presentan en determinada fecha y hora, aunque el conteo de los segundos, minutos, horas, días, meses y años se puede realizar directamente con el microcontrolador, es mejor, para evitarse extensos programas, utilizar un integrado especializado en esta función, lo cual tiene grandes ventajas, como son: Reducido tamaño, bajo costo, facilidad para lectura y escritura, bajo consumo de corriente, un reducido número de componentes externos para un optimo funcionamiento, así como el control de una batería externa que respalda su funcionamiento en caso de falla de energía.

El reloj de tiempo real DS1307 cumple perfectamente con muchas de las necesidades normales en la adquisición y registro de tiempo, siendo un reloj calendario en BCD, cuyas características más destacadas son:

- Reloj de tiempo real que cuenta los segundos, los minutos, las horas, la fecha, el mes, el día de la semana, y el año, con compensación del año bisiesto, válido hasta el año 2100.
- Protocolo I^2C
- 56 bytes de memoria RAM no volátil, para almacenamiento de datos.
- Señal de salida de onda cuadrada programable.
- Circuitería interna de respaldo para la alimentación.
- Bajo consumo de potencia: menor a 500nA en el modo de respaldo, a 25°C.
- Solo 8 pines.

A continuación en la Figura 4.39 se puede observar la disposición de los pines del integrado:

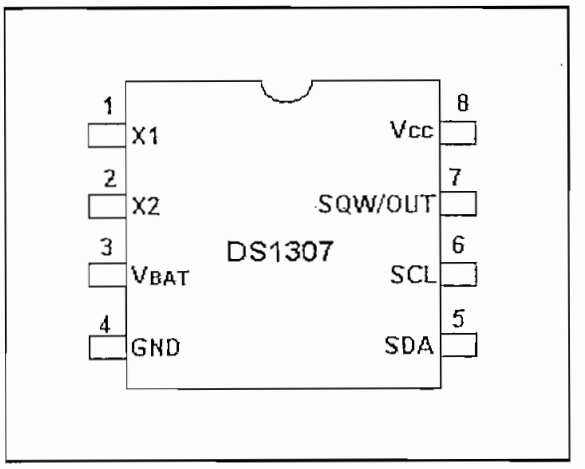


Figura 4.39 Configuración del reloj calendario DS1307

Descripción de los pines:

Vcc, GND: La alimentación DC es proporcionada al circuito a través de estos pines. Vcc es la entrada de +5v, mientras que GND es la referencia.

VBAT: Entrada de alimentación de una pila estándar de litio de 3v. El voltaje debe estar entre 2.5 y 3.5 voltios para una operación apropiada. El fabricante hace cuentas que una pila de litio con una corriente de 35mAh o mayor, podrá respaldar la operación del DS1307 por más de 10 años en ausencia de energía.

SCL: Entrada para sincronizar el movimiento de datos en la interface serial.

SDA: Entrada/Salida de datos para la interface I^2C. Este pin es de drenaje abierto, por lo cual requiere una resistencia pull-up externa.

SQW/OUT: Salida para generar una de cuatro posibles frecuencias de salida: 1Hz, 4KHz, 8KHz, 32KHz. Este pin también es de drenaje abierto, por lo cual requiere una resistencia externa de pull-up.

X1,X2: Conexiones para el cristal de cuarzo estándar de 32768 Hz. La circuitería interna del oscilador está diseñada para operar con un cristal de una capacitancia de 12.5 pf.

Cuando Vcc cae por debajo de 1.25VBAT, la alimentación del circuito integrado es conmutada a VBAT, y la lectura y escritura de los registros de tiempo y la RAM es denegada, pero la función que mantiene el registro del tiempo se conserva y los registros de la RAM permanecen inalterados.

Mapa de direcciones:

En la Figura 4.40 se puede observar el mapa de direcciones para los registros del reloj en tiempo real (RTC) y la memoria RAM del DS1307. Los registros del primero se localizan en las direcciones 00H hasta 07H, mientras que la RAM va desde 08H hasta 3FH.

Como el puntero de direcciones es de solo 6 bits, durante un acceso múltiple, cuando éste alcanza la dirección 3FH, este se traslapa con la dirección 00H, el registro de los segundos RTC.

Reloj y calendario:

La información del tiempo y el calendario está en formato BCD (Binario Codificado Decimal), y es obtenida leyendo los registros apropiados.

El bit 7 del registro 0 es el bit CH (Clock Halt). Cuando este se deja en 1, el oscilador se inhabilita, deteniéndose el funcionamiento del reloj. Cuando se deja en cero, el oscilador es habilitado, y el RTC funciona normalmente

Figura 4.40 Registros del DS1307

Registro de control:

Este registro es utilizado para determinar el funcionamiento del pin de salida SQW/OUT.

OUT: Es el control del nivel de salida cuando la salida de la onda cuadrada está inhabilitada. Si SQWE =0, el nivel en el pin 7 del DS1307 será alto si el bit OUT =1, y será bajo si el bit OUT=0.

SQWE: Este bit cuando se ajusta a 1 habilita la salida del oscilador. La frecuencia de la onda cuadrada dependerá de los bits RS1 y RS0.

RS1, RS0: Estos bits controlan la frecuencia de la onda cuadrada de salida, cuando esta se habilita. En la Tabla 4.1 se muestra las frecuencias que pueden ser seleccionadas con los bits RS1 y RS0.

RS1	RS0	FRECUENCIA DE SALIDA SQW
0	0	1Hz
0	1	4KHz
1	0	8KHz
1	1	32KHz

Tabla 4.1 Frecuencias de salida de la onda cuadrada

Esta práctica consiste en presentar la fecha y hora en un módulo LCD con el uso del integrado DS1307; además, mediante un pulsador, si este se encuentra presionado, entonces actualizará la hora y fecha grabada en el microcontrolador, caso contrario no la actualizará sino que presentará el valor almacenado en el DS1307. Para leer o escribir en el reloj calendario se utiliza el protocolo I^2C, para esto se utilizan las subrutinas LEER y ESCRIBIR las cuales se encargan de generar todas las señales y tiempos requeridos dentro de este protocolo. Se debe notar que en estas instrucciones al inicio se transmite la dirección esclava del integrado con el cual se trabaja, ya que mediante este protocolo pueden estar conectadas varios integrados que se comunique usando I^2C. Para el caso del DS1307 su dirección esclava es 1101000+ el bit que define su operación a realizar, en el caso de escritura es cero y en el caso de lectura es uno.

El circuito implementado se muestra a continuación en la Figura 4.41, se puede ver que la comunicación con el display es realizada a 4 bits.

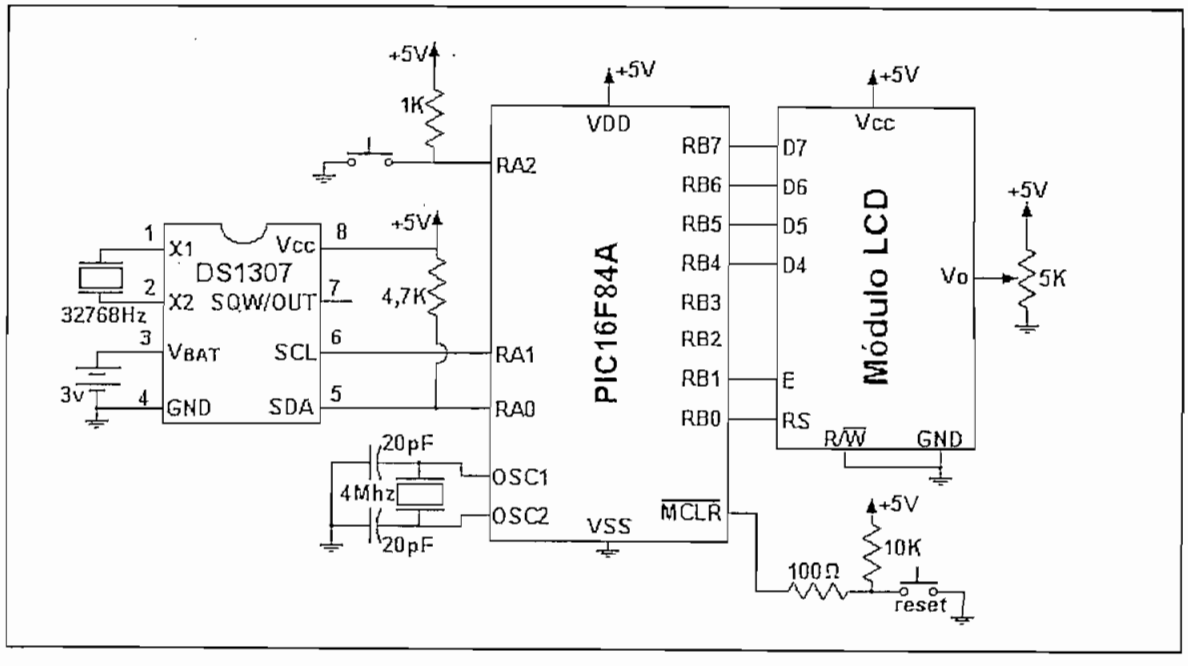


Figura 4.41 Circuito del reloj calendario

```
;Este programa realiza la función de un reloj calendario digital,
;el cual lee la hora actual de un reloj DS1307 mediante el uso del
;protocolo I2C y lo presenta en el módulo LCD.

        List    p=16f84a
        include p16f84a.inc

;Etiquetas de los registros de propósito general
addr    equ     0ch     ;posición de memoria de lectura o escritura
slave   equ     0dh     ;dirección del dispositivo en el bus I2C (1101000)
txbuf   equ     0eh     ;buffer de transmision
eeprom  equ     0fh     ;buffer de bits
rxbuf   equ     10h     ;buffer de recepci¢n
count   equ     11h     ;registo contador de bits
loops2  equ     12h     ;registro utilizado en retardos
loops   equ     13h     ;registro utilizado en retardos
datao   equ     14h     ;registro contiene el dato a transmitir
lazo1   equ     15h     ;registro para conteo en la subrutina retardo
lazo2   equ     16h     ;registro para conteo en la subrutina retardo
lazo3   equ     17h     ;registro utilizado para retardo entre caracteres
puntero equ     19h     ;reg. puntero de la tabla
caract  equ     1bh     ;registro temporal para el dato a enviar al LCD
seg     equ     1ch     ;registro para los segundos
min     equ     1dh     ;registro para los minutos
hor     equ     1eh     ;registro para las horas
dia     equ     1fh     ;registro para los días (lunes a domingo)
fech    equ     20h     ;registro para el día del mes (01-31)
mes     equ     21h     ;registro para los meses (00-12)
año     equ     22h     ;registro para el año (00-99)
conf    equ     23h     ;registro de configuraciones del reloj
tempd   equ     24h     ;registro temporal del mes
di      equ     7       ;bit de entrada desde eeprom
do      equ     6       ;bit de salida para eeprom
sdata   equ     0       ;linea de datos seriales (pin RA0)
sclk    equ     1       ;reloj serial (pin RA1)
RS      equ     00h     ;RB0 maneja la línea de control RS
E       equ     01H     ;RB1 maneja la línea de control E
;*************************************************************
        org     00h     ;Vector de reset
        goto    INICIO
        org     05h
;************************ TABLAS ***************************
tabla   addwf   PCL     ;suma W al PC
        retlw   'R'     ;mensaje  a ser mostrado
        retlw   'E'
        retlw   'L'
        retlw   'O'
        retlw   'J'
        retlw   ' '
        retlw   'C'
        retlw   'A'
        retlw   'L'
        retlw   'E'
        retlw   'N'
        retlw   'D'
        retlw   'A'
        retlw   'R'
        retlw   'I'
        retlw   'O'
        retlw   ' '
        retlw   ' '
        retlw   ' '
        retlw   ' '
        retlw   ' '
        retlw   ' '
        retlw   '/'
        retlw   ' '
        retlw   ' '
        retlw   ':'
        retlw   ' '
        retlw   ' '
        retlw   ' '
        retlw   'M'
        retlw   ' '
        retlw   ' '
        retlw   0
```

```
tabla1  addwf   PCL     ;suma W al PC
        nop
        retlw   'L'     ;mensaje  a ser mostrado
        retlw   'M'     ;en el primer digito
        retlw   'M'     ;del día
        retlw   'J'
        retlw   'V'
        retlw   'S'
        retlw   'D'

tabla2  addwf   PCL     ;suma W al PC
        nop
        retlw   'U'     ;mensaje  a ser mostrado
        retlw   'A'     ;en el segundo digito
        retlw   'I'     ;del día
        retlw   'U'
        retlw   'I'
        retlw   'A'
        retlw   'M'

tabla3  addwf   PCL     ;suma W al PC
        nop
        retlw   'E'     ;mensaje  a ser mostrado
        retlw   'F'     ;en el primer digito
        retlw   'M'     ;del mes
        retlw   'A'
        retlw   'M'
        retlw   'J'
        retlw   'J'
        retlw   'A'
        retlw   'S'
        retlw   'O'
        retlw   'N'
        retlw   'D'
tabla4  addwf   PCL     ;suma W al PC
        nop
        retlw   'N'     ;mensaje  a ser mostrado
        retlw   'E'     ;en el segundo digito
        retlw   'A'     ;del mes
        retlw   'B'
        retlw   'Y'
        retlw   'N'
        retlw   'L'
        retlw   'G'
        retlw   'P'
        retlw   'C'
        retlw   'V'
        retlw   'I'
;****************** SUBRUTINAS PARA EL LCD ************************

;************** Subrutina de retardo de 765usg ********************
retardo movlw   0ffh            ;Esta subrutina carga en
        movwf   lazo1           ;lazo1 255, generando un
lazo    decfsz  lazo1           ;retardo de 255 * 3usg
        goto    lazo            ;por las instruc. decfsz y goto
        retlw   0

;************** Subrutina de retardo de 195msg ********************
retard1 movlw   0ffh            ;inicia el lazo2 con ffh
        movwf   lazo2           ;para generar un lazo de retardo
repite  call    retardo         ;de 391msg = 256 * (765+2)
        call    retardo
        decfsz  lazo2
        goto    repite
        retlw   0
;************** Subrutina de control y datos ***********************
control bcf     PORTB,RS        ;pone en 0 la línea RS (señal de control)
        goto    sigue
dato    bsf     PORTB,RS        ;pone en uno a la línea RS(señal de dato)
sigue   bsf     PORTB,E         ;habilita el módulo LCD (E=1)
        movwF   caract          ;guarda temporalmente el dato en caract
        movlw   0fh             ;pone en cero la
        andwF   PORTB,F         ;parte alta del puerto B
        movf    caract,w        ;filtra la parte alta del
        andlw   0f0h            ;caracter a mostrar y lo
        iorwf   PORTB,F         ;une con la parte baja del puerto B
        call    retardo         ;genera un retardo de 765usg
        bcf     PORTB,E         ;Deshabilita el módulo LCD (E=0)
        call    retardo         ;genera un retardo para que tome el dato
        bsf     PORTB,E         ;habilita nuevamente el LCD
        movlw   0fh             ;pone en cero la
        andwf   PORTB,F         ;parte alta del puerto B
```

```
BITIN                           ;Esta rutina lee un bit que entrega
                                ;la memoria y lo pone en un registro
        bsf     eeprom,di       ;asume que el bit es de nivel alto
        movlw   b'11111101'     ;programa pin de datos como entrada
        tris    PORTA
        bsf     PORTA,sclk      ;sube la linea del reloj
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        btfss   PORTA,sdata     ;pregunta por el estado del pin de datos
        bcf     eeprom,di       ;si es bajo lo pone en ese nivel
        bcf     PORTA,sclk      ;si es alto lo deja como se asumió antes
        retlw   0               ;

TX                              ;Esta rutina se encarga de transmitir
                                ;un byte completo hacia la memoria del reloj
        movlw   .8
        movwf   count           ;el número de bits es 8
TXLP    bcf     eeprom,do       ;asume que el bit a enviar es bajo
        btfsc   txbuf,7         ;consulta el estado real del bit
        bsf     eeprom,do       ;si era alto lo deja con dicho nivel
        call    BITOUT          ;saca el bit por el puerto
        rlf     txbuf,1         ;rota el byte que se está transmitiendo
        decfsz  count           ;pregunta si ya pasaron los 8 bits
        goto    TXLP            ;si no ha terminado sigue transmitiendo
        call    BITIN           ;espera el reconocimiento enviado por
        retlw   0               ;el reloj (ACK)

RX                              ;Esta rutina recibe un byte enviado por
                                ;la memoria y lo entrega en el registro rxbuf
        clrf    rxbuf           ;borra el buffer de entrada
        movlw   .8              ;indica que recibe 8 bits
        movwf   count
        bcf     STATUS,0        ;Borra el carry
RXLP    rlf     rxbuf, F        ;rota a la izquierda
        call    BITIN           ;lee un bit
        btfsc   eeprom,di
        bsf     rxbuf,0         ;si es necesario pone el bit en uno
        decfsz  count           ;pregunta si completo 8 bits
        goto    RXLP            ;sino, recibe otro bit
        bsf     eeprom,do       ;envia el ACK de asentimiento
        call    BITOUT          ;para terminar
        retlw   0
LEER                    ;Esta rutina recibe la direcci¢n que se
                        ;desea LEER y devuelve el dato que tiene grabado
        call    BSTART          ;genera el start bit
        nop                     ;
        nop                     ;
        bcf     slave,0         ;selecciona la memoria
        movf    slave,w         ;y selecciona modo de escritura
        movwf   txbuf           ;
        call    TX              ;envía esos datos a la memoria
        movf    addr,w          ;
        movwf   txbuf           ;envía la posicion de memoria a ser leida
        call    TX              ;
        nop                     ;ahora se selecciona nuevamente la memoria
        nop                     ;y se le indica modo de lectura
        call    BSTART          ;genera start bit
        nop
        nop
        bsf     slave,0         ;indica que se va a LEER
        movf    slave,w         ;selecciona el dispositivo
        movwf   txbuf           ;
        call    TX              ;envia esa información a la memoria
        nop
        call    RX              ;la memoria entrega el byte grabado
                                ;en esa dirección al registro rxbuf
        bsf     eeprom,do       ;envia el ACK de reconocimiento
        call    BITOUT
        call    BSTOP           ;se envia el stop bit para finalizar comunicación
        retlw   0
```

```
ESCRIB                          ;Esta rutina escribe un dato en la posición
                                ;de memoria que se le indique en el registro addr
        call    BSTART          ;genera el start bit
        nop                     ;
        nop                     ;
        bcf     slave,0         ;selecciona la memoria
        movf    slave,w         ;y selecciona modo de escritura
        movwf   txbuf           ;
        call    TX              ;envía esos datos a la memoria
        movf    addr,w          ;
        movwf   txbuf           ;envía la posición de memoria a ser grabada
        call    TX              ;ahora se selecciona nuevamente la memoria
        nop                     ;y se le indica modo de lectura
        nop
        movf    datao,w         ;toma el dato que va a ser grabado
        movwf   txbuf           ;y lo envía
        call    TX
        call    BSTOP
        movlw   .10             ;retardo de 10 ms al escribir
        movwf   loops           ;cada dato
        call    WAIT
        retlw   0
;       org     100h


;       *********   PROGRAMA PRINCIPAL   ********

INICIO  bsf     STATUS,5        ;se ubica en el segundo banco de RAM
        movlw   0ffh            ;se carga el registro W con unos
        movwf   TRISA           ;se programan los pines del puertoA como entradas
        movlw   00h             ;se carga el registro W con 00
        movwf   TRISB           ;se programan los pines del puerto B como salidas
        bcf     STATUS,5        ;se ubica en el primer banco de memoria RAM
        movlw   b'11010000'     ;La direcci¢n A0, A1, y A2 de la memoria
        movwf   slave           ;en el bus I2C es 000
        btfsc   PORTA,2         ;pregunta si hay que igualar el reloj
        goto    noigua          ;cuando no está presionada la tecla salta
        movlw   00h             ;sin actualizar los datos de fecha y hora
        movwf   seg             ;
        movlw   12h             ;
        movwf   min             ;
        movlw   64h             ;carga
        movwf   hor             ;la
        movlw   01h             ;fecha
        movwf   dia             ;y
        movlw   03h             ;hora
        movwf   fech            ;actual
        movlw   12h             ;
        movwf   mes             ;
        movlw   01h             ;
        movwf   año             ;
        movlw   90h             ;
        movwf   conf            ;
;******** Inicia la transmisión hacia el reloj de la fecha y hora ********
        clrf    addr            ;la dirección de escritura inicial en 00h
        movlw   1ch             ;inicia el puntero de la ram
        movwf   FSR             ;con direccionamiento indirecto
sigue1  movf    INDF,w          ;lee el dato apuntado por el FSR
        movwf   datao           ;carga el dato en el registro de transmisión
        call    ESCRIB          ;llama a la subrutina de esritura I2C
        incf    addr            ;incrementa la dirección de escritura ram
        incf    FSR             ;incrementa el puntero de la ram
        btfss   addr,3          ;pregunta si terminó con la tabla
        goto    sigue1          ;si no terminó salta al siguiente

;********** Configuraciones para el LCD ********************************
noigua  call    retard1
        movlw   02h             ;configura comunicación a 4 bits
        call    control
        movlw   28h             ;configura el display a 4 bits y 2 líneas
        call    control
        movlw   0ch             ;activa el display y desactiva el cursor
        call    control
        movlw   06h             ;modo de desplazamiento a la derecha
        call    control
        movlw   01h             ;borrado del display
        call    control
        clrf    puntero         ;inicia en cero el puntero de la tabla
```

```
ciclo   movf    puntero,w
        call    tabla
        call    dato
        movlw   05fh            ;inicia el retardo entre caracteres
        movwf   lazo3
laz3    call    retardo
        decfsz  lazo3
        goto    laz3
        incf    puntero         ;incrementa el puntero de la tabla
        movlw   D'16'           ;pregunta si el puntero revasó la
        subwf   puntero,w       ;primera fila de la tabla
        btfss   STATUS,C
        goto    ciclo
        movlw   D'16'           ;pregunta si es el primer caracter de la
        xorwf   puntero,w       ;segunda línea para iniciar el puntero de
        btfss   STATUS,Z        ;la ram del módulo LCD en la segunda línea
        goto    linea2          ;no es el primer caracter de la 2da línea
        movlw   0c0h            ;ubicar el puntero de la ram
        call    control         ;del módulo LCD en la segunda linea
linea2  movlw   D'32'           ;pregunta si ya terminó la segunda línea
        xorwf   puntero,w       ;para entonces iniciar el puntero de la
        btfss   STATUS,Z        ;ram en la primera línea
        goto    ciclo           ;aún no termina
;************ inicia la lectura de la información del reloj ***************
repite1 clrf    addr            ;la dirección de escritura inicial en 00h
        movlw   1ch             ;inicia el puntero de la ram
        movwf   FSR             ;entonces va leyendo la
sigue2  call    LEER            ;memória del reloj
        movf    rxbuf,w         ;y la ba guardando
        movwf   INDF            ;en la memória ram
        incf    addr            ;del microcontrolador
        incf    FSR
        btfss   addr,3          ;pregunta si ya leyó los 8 bytes iniciales
        goto    sigue2
;Actualización de datos en el LCD
        movlw   0ceh            ;hubica el puntero LCD para decena de seg
        call    control
        swapf   seg,w           ;intercambio los bits altos por los bajos
        andlw   0fh             ;filtro la parte baja del registro
        addlw   30h             ;transforma a ASCII
        call    dato            ;y envía al LCD
        movlw   0cfh            ;hubica el puntero LCD para unidad de seg
        call    control         ;hubica el puntero para unidad de seg
        movf    seg,w           ;se carga seg en w para luego
        andlw   0fh             ;filtrar la parte baja del registro
        addlw   30h             ;transformarla a ASCII
        call    dato            ;y enviarla al LCD
        movlw   0cah            ;hubica el puntero LCD para decena de min
        call    control
        swapf   min,w           ;intercambio los bits altos por los bajos
        andlw   0fh             ;filtro la parte baja del registro
        addlw   30h             ;transforma a ASCII
        call    dato            ;y se envía al LCD
        movlw   0cbh            ;hubica el puntero LCD para unidad de min
        call    control         ;
        movf    min,w           ;se carga minutos a w para
        andlw   0fh             ;filtrar la parte baja del registro
        addlw   30h             ;transformarla a ASCII
        call    dato            ;y enviarla al LCD
        movlw   0c7h            ;hubica el puntero LCD para decena de hor
        call    control
        swapf   hor,w           ;intercambio los bits altos por los bajos
        andlw   01h             ;filtro la parte baja del registro
        addlw   30h             ;transforma a ASCII
        call    dato            ;y se envía al LCD
        movlw   0c8h            ;hubica el puntero LCD para unidad de hor
        call    control
        movf    hor,w           ;se carga las horas en w
        andlw   0fh             ;filtro la parte baja del registro
        addlw   30h             ;transforma a ASCII
        call    dato            ;y se envía al LCD
        movlw   0c2h            ;hubica el puntero LCD para decena de fecha
        call    control
```

```
        andlw   03h             ;filtro la parte baja del registro
        addlw   30h             ;transforma a ASCII
        call    dato            ;y se envía al LCD
        movlw   0c3h            ;hubica el puntero LCD para unidad de fecha
        call    control
        movf    fech,w          ;se carga la fecha en w
        andlw   0fh             ;filtro la parte baja del registro
        addlw   30h             ;transforma a ASCII
        call    dato            ;y se envía al LCD
        movlw   0c0h            ;hubica el puntero LCD primera letra día
        call    control
        movf    dia,w           ;se carga en w el valor del día
        call    tabla1          ;para obtener la letra correspondiente
        call    dato            ;de la tabla1 y enviarla al LCD
        movlw   0c1h            ;hubica el puntero LCD segunda letra dia
        call    control
        movf    dia,w           ;se carga en w el valor del día
        call    tabla2          ;para obtener la letra correspondiente
        call    dato            ;de la tabla2 y enviarla al LCD
        movlw   0c4h            ;hubica el puntero LCD primera letra mes
        call    control
        movf    mes,w           ;se carga el valor del mes en w
        andlw   0fh             ;se filtra el byte bajo
        btfsc   mes,4           ;y se transforma de BCD a hexadecimal
        addlw   d'10'           ;al sumar 10 al byte bajo si la decena
        movwf   tempd           ;del mes en BCD es 1
        call    tabla3          ;la primera letra del mes correspondiente
        call    dato            ;es leida de la tabla y enviada al LCD
        movlw   0c5h            ;hubica el puntero LCD segunda letra mes  -
        call    control
        movf    tempd,w         ;carga en w el valor del mes en hexadecimal
        call    tabla4          ;que fué almacenado en tempd, obteniendo la
        call    dato            ;letra correspondiente de la tabla4, se envía
        movlw   0cch            ;hubica el puntero LCD en la segunda letra
        call    control         ;de AM ó PM
        btfsc   hor,5           ;se revisa el bit que indica AM ó PM
        goto    pm              ;si es 1 salta a pm
        movlw   'A'             ;si es cero envía la letra A de AM
        call    dato
        goto    repite1         ;lectura el reloj y actualización del LCD
pm      movlw   'P'             ;si es uno envía la letra P de PM
        call    dato
        goto    repite1         ;lectura el reloj y actualización del LCD
        end
```

Figura 4.42 Programa para el reloj calendario

4.2 Práctica con el PIC16C71.

La característica más importante de este microcontrolador sin duda es el conversor A/D interno, entonces se realizará una práctica que permita realizar el manejo de una señal análoga, para esto el sistema de desarrollo cuenta con un módulo de entrada de señales analógicas.

Práctica 12: Termómetro digital

Esta práctica consiste en tomar la medida de la temperatura mediante el uso del termistor LM35, y presentarlo en el módulo LCD. El termistor presenta a

su salida una variación de 10mv/°C, con una precisión de 0,5°C, este elemento puede ser energizado entre 4 y 30 voltios. La resolución del conversor de 8 bits es de 5v / 255 (FFH),obteniendo 20mv por unidad; sin embargo, el termistor genera una variación de 10mv/°C, entonces se debe duplicar esta señal para igualar a la resolución del conversor, obteniendo así la lectura directa de la temperatura. Esta duplicación de la señal se la realizará vía software adicionando al valor obtenido de la conversión su mismo valor, este proceso producirá que la lectura presentada en el display sea siempre un valor par, siendo un problema aceptable por tratarse de una práctica. Para corregir este defecto se debería acondicionar la señal mediante amplificadores operacionales en configuración de amplificador no inversor con ganancia 2. Para esta práctica la conexión del módulo LCD se la realizará mediante la interface a 4 bits, reduciendo así el cableado, además se resolverá el programa mediante el uso de la interrupción, generada por el conversor al terminal su proceso, para esto se deben habilitar las interrupciones y escribir la rutina de atención a esta desde la dirección 04H del vector interrupciones. El circuito planteado para solucionar esta práctica se muestra en la Figura 4.43 mostrado a continuación.

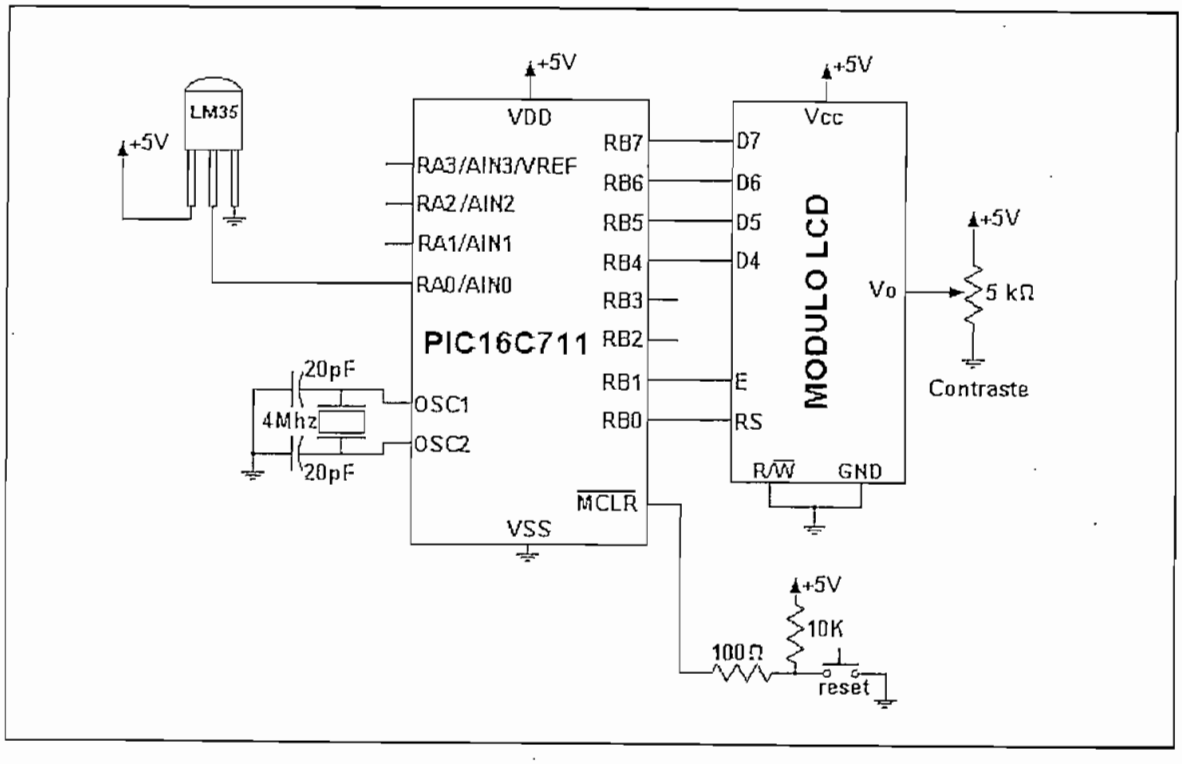


Figura 4.43 Circuito del termómetro digital

El diagrama de flujo del programa y el programa desarrollado se muestran en las Figuras 4.44 y 4.45 a continuación.

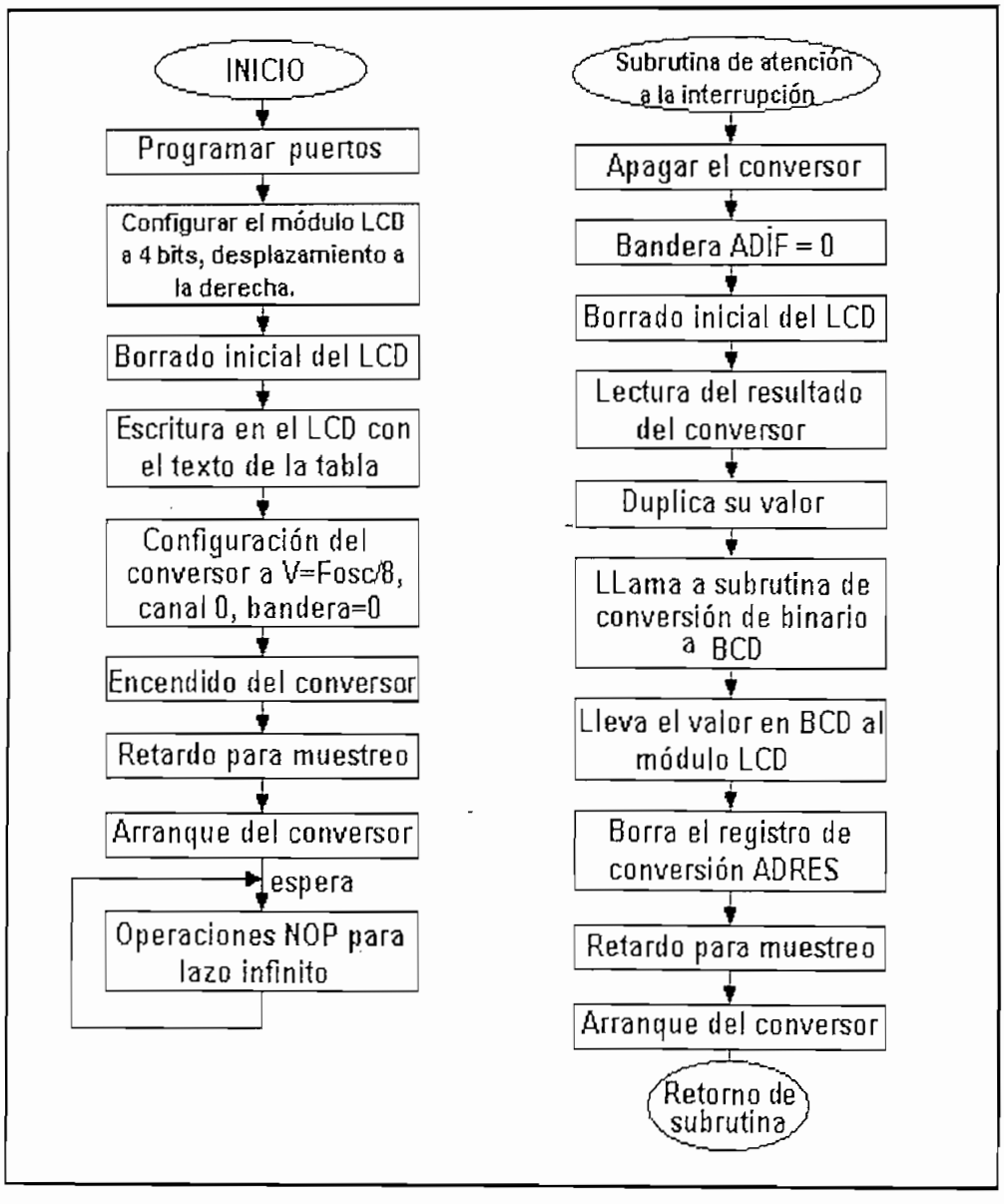


Figura 4.44 Diagrama de flujo del termómetro digital

```
;Esta práctica consiste en realizar un termómetro digital
;utilizando un termistor, cuya señal es digitalizada mediante
;el conversor A/D del PIC16C71, el valor de la temperatura
;es mostrado en el módulo LCD

        List    p=16c71
        include p16c71.inc
;Etiquetas de registros de propósito general
lazo1   equ     0Ch             ;registro para conteo en la subrutina retardo
lazo2   equ     0Dh             ;registro para conteo en la subrutina retardo
lazo3   equ     0eh
lazo4   equ     0Fh
puntero equ     10h
bandera equ     11h
caract  equ     12h             ;registro temporal para el dato a enviar al LCD
unidad  equ     13h             ;contiene la unidad
decena  equ     14h             ;contiene la decena
centena equ     15h             ;contiene la centena
RS      equ     00h             ;RB0 maneja la línea de control RS
E       equ     01H             ;RB1 maneja la línea de control E
reset   org     0               ;se escribe en la línea 0 la instrucción
        goto    inicio          ;de salto a la línea etiquetada con inicio
;************* vector de interrupción ********************
        org     04h             ;vector de interrupción
        bcf     ADCON0,ADON     ;apaga el converidor
        bcf     ADCON0,ADIF     ;pone en cero a la bandera de interrupción
        movf    ADRES,w         ;lee el resultado de la conversión
        movwf   unidad          ;y lo carga el registro unidad
        addwf   unidad,F        ;y duplica el valor de la señal
        call    bcd             ;llama a subrutina para transformar a BCD
        movlw   0cbh            ;hubica el puntero del LCD donde va centena
        call    control
        movf    centena,w       ;carga la centena para enviarla
        addlw   30h             ;transforma a ASCII
        call    dato
        movlw   0cch            ;hubica el puntero del LCD donde va decenas
        call    control
        movf    decena,w        ;carga la decena para enviarla
        addlw   30h             ;transforma a ASCII
        call    dato
        movlw   0cdh            ;hubica el puntero del LCD donde va unidades
        call    control
        movf    unidad,w        ;carga la unidad para enviarla
        addlw   30h             ;transforma a ASCII
        call    dato
        call    retard1         ;llama al retardo1
        clrf    ADRES
conver  bsf     ADCON0,ADON     ;enciende el conversor A/D
        movlw   0Fh             ;Inicia un retardo para muestreo
        movwf   lazo4
laz4    decfsz  lazo4
        goto    laz4
reg     bsf     ADCON0,GO_DONE  ;inicia la conversión
        retfie
;************* Subrutina de conversión de binario de bcd **********
bcd     clrf    decena
        clrf    centena
        movlw   d'100'
otra    subwf   unidad,F        ;resta 100 al valor inicial
        btfss   STATUS,C        ;verifica el carry
        goto    suma            ;si es cero deja de restar
        incf    centena         ;si es uno incrementa centena
        goto    otra            ;vuelve a restar
suma    addwf   unidad          ;sumarle 100
        movlw   d'10'
repit   subwf   unidad          ;resta 10 al valor
        btfss   STATUS,C        ;verifica el carry
        goto    suma1           ;si es cero deja de restar
        incf    decena          ;si es uno incrementa la decena
        goto    repit
suma1   addwf   unidad          ;sumarle 10 al valor
        retlw   0
```

```
;************** Subrutina de retardo de 765usg ********************
retardo movlw   0ffh            ;Esta subrutina carga en
        movwf   lazo1           ;lazo1 255, generando un
lazo    decfsz  lazo1           ;retardo de 255 * 3usg
        goto    lazo            ;por las instruc. decfsz y goto
        retlw   0
;************** Subrutina de retardo de 195msg ********************
retard1 movlw   0ffh            ;inicia el lazo2 con ffh
        movwf   lazo2           ;para generar un lazo de retardo
repite  call    retardo         ;de 391msg = 256 * (765*2)
        call    retardo
        decfsz  lazo2
        goto    repite
        retlw   0
;************** Subrutina de control y datos ***********************
control bcf     PORTB,RS        ;pone en cero a la línea RS (señal de control)
        goto    sigue
dato    bsf     PORTB,RS        ;pone en uno a la línea RS (señal de dato)
sigue   bsf     PORTB,E         ;habilita el módulo LCD (E=1)
        movwf   caract          ;guarda temporalmente el dato en caract
        movlw   0fh             ;pone en cero la
        andwf   PORTB,F         ;parte alta del puerto B
        movf    caract,w        ;filtra la parte alta del
        andlw   0f0h            ;caracter a mostrar y lo
        iorwf   PORTB,F         ;une con la parte baja del puerto B
        call    retardo         ;genera un retardo de 765usg
        bcf     PORTB,E         ;Deshabilita el módulo LCD (E=0)
        call    retardo         ;genera un retardo para que tome el dato
        bsf     PORTB,E         ;habilita nuevamente el LCD
        movlw   0fh             ;pone en cero la
        andwf   PORTB,F         ;parte alta del puerto B
        swapf   caract,w        ;intercambio de los 4 bits bajos por los altos
        andlw   0f0h            ;filtra la parte alta del registro caract
        iorwf   PORTB,F         ;y lo une con la parte baja del puerto B
        call    retardo         ;genera un retardo de 765usg
        bcf     PORTB,E         ;deshabilita el módulo LCD
        call    retardo         ;genera un retardo para que tome el dato
        retlw   0               ;retorno de la subrutina

;************** tabla con el mensaje a desplazar ********************
tabla   addwf   PCL     ;suma W al PC
        retlw   ' '     ;mensaje  a ser mostrado
        retlw   'L'
        retlw   'A'
        retlw   ' '
        retlw   'T'
        retlw   'E'
        retlw   'M'
        retlw   'P'
        retlw   'E'
        retlw   'R'
        retlw   'A'
        retlw   'T'
        retlw   'U'
        retlw   'R'
        retlw   'A'
        retlw   ' '
        retlw   ' '
        retlw   'M'
        retlw   'E'
        retlw   'D'
        retlw   'I'
        retlw   'D'
        retlw   'A'
        retlw   ' '
        retlw   'E'
        retlw   'S'
        retlw   ' '
        retlw   ' '
        retlw   ' '
        retlw   ' '
        retlw   d'223'
        retlw   'C'
        retlw   ' '
        retlw   0
```

```
;******************  Programa principal  **********************************

inicio  bsf     STATUS,RP0      ;se ubica en el segundo banco de la RAM
        movlw   0FFh
        movwf   TRISA           ;se programa el puerto A como entradas
        movlw   0ch
        movwf   TRISB           ;puertoB como salidas [RB0-RB1] y [RB4-RB7]
        BCF     ADCON1,PCFG1    ;programa [RA0-RA3] como entradas analógicas
        BCF     ADCON1,PCFG0    ;y también a Vdd como voltaje de referencia
        bcf     STATUS,RP0      ;se ubica en el primer banco de la RAM
        movlw   0c0h            ;Habilita la interrupción general y
        movwf   INTCON          ;la interrupción del converor A/D
        movlw   02h             ;configura comunicación a 4 bits
        call    control
        movlw   28h             ;configura el display a 4 bits y 2 líneas
        call    control
        movlw   0ch             ;activa el display y desactiva el cursor
        call    control
        movlw   06h             ;modo de desplazamiento a la derecha
        call    control
        movlw   01h             ;borrado del display
        call    control
        clrf    puntero         ;inicia en cero el puntero de la tabla
ciclo   movf    puntero,w
        call    tabla
        call    dato
        movlw   05fh            ;inicia el retardo entre caracteres
        movwf   lazo3
laz3    call    retardo
        decfsz  lazo3
        goto    laz3
        incf    puntero         ;incrementa el puntero de la tabla
        movlw   D'16'           ;pregunta si el puntero revasó la
        subwf   puntero,w       ;primera fila de la tabla
        btfss   STATUS,C
        goto    ciclo
        movlw   D'16'           ;pregunta si es el primer caracter de la
        xorwf   puntero,w       ;segunda línea para iniciar el puntero de
        btfss   STATUS,Z        ;la ram del módulo LCD en la segunda línea
        goto    linea2          ;no es el primer caracter de la 2da línea
        movlw   0c0h            ;ubicar el puntero de la ram
        call    control         ;del módulo LCD en la segunda linea
linea2  movlw   D'32'           ;pregunta si ya terminó la segunda línea
        xorwf   puntero,w       ;para entonces iniciar el puntero de la
        btfss   STATUS,Z        ;ram en la primera línea
        goto    ciclo           ;aún no termina

;****** Inicio de la conversión A/D ******

        movlw   b'01000000'     ;configura el converosr: U=FOSC/8; canal0
        movwf   ADCON0          ;modulo A/D off;bandera ADIF=0;conversión off
        bsf     ADCON0,ADON     ;enciende el conversor A/D
        movlw   0fh             ;Inicia un retardo para muestreo
        movwf   lazo4
lz4     decfsz  lazo4
        goto    lz4
        bsf     ADCON0,GO_DONE  ;inicia la conversión
espera  nop
        nop
        goto    espera
        end
```

Figura 4.45 Programa para el termómetro digital

4.3 Práctica con el PIC12C508.

La característica más importante de este microcontrolador es su tamaño reducido y gran funcionalidad. Para practicar con el manejo de este microcontrolador se realizarán prácticas que permita controles sencillos como el manejo temporizado de un relé.

Práctica 13: Manejo temporizado de un relé

Esta práctica consiste en el encendido secuencial de un relé, cuyo funcionamiento es controlado por un pulsador, mediante el cual, con un pulso inicia su funcionamiento secuencial y con otro pulso lo detiene. En la Figura 4.46 se puede observar el circuito a implementarse.

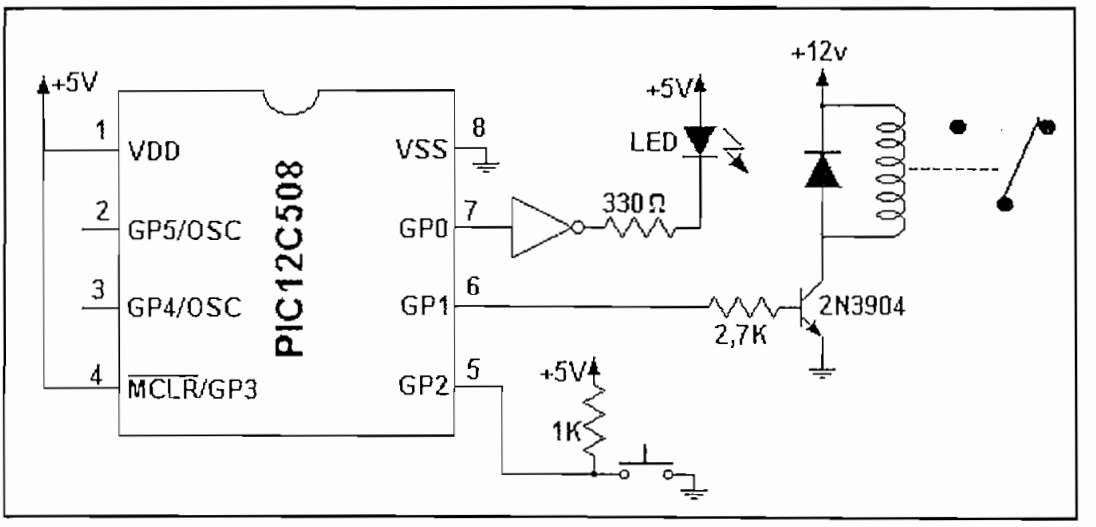


Figura 4.46 Circuito de manejo de un relé temporizado

En las Figuras 4.47 y 4.48 mostradas a continuación, se pueden observar los diagramas de flujo y de programa para esta práctica

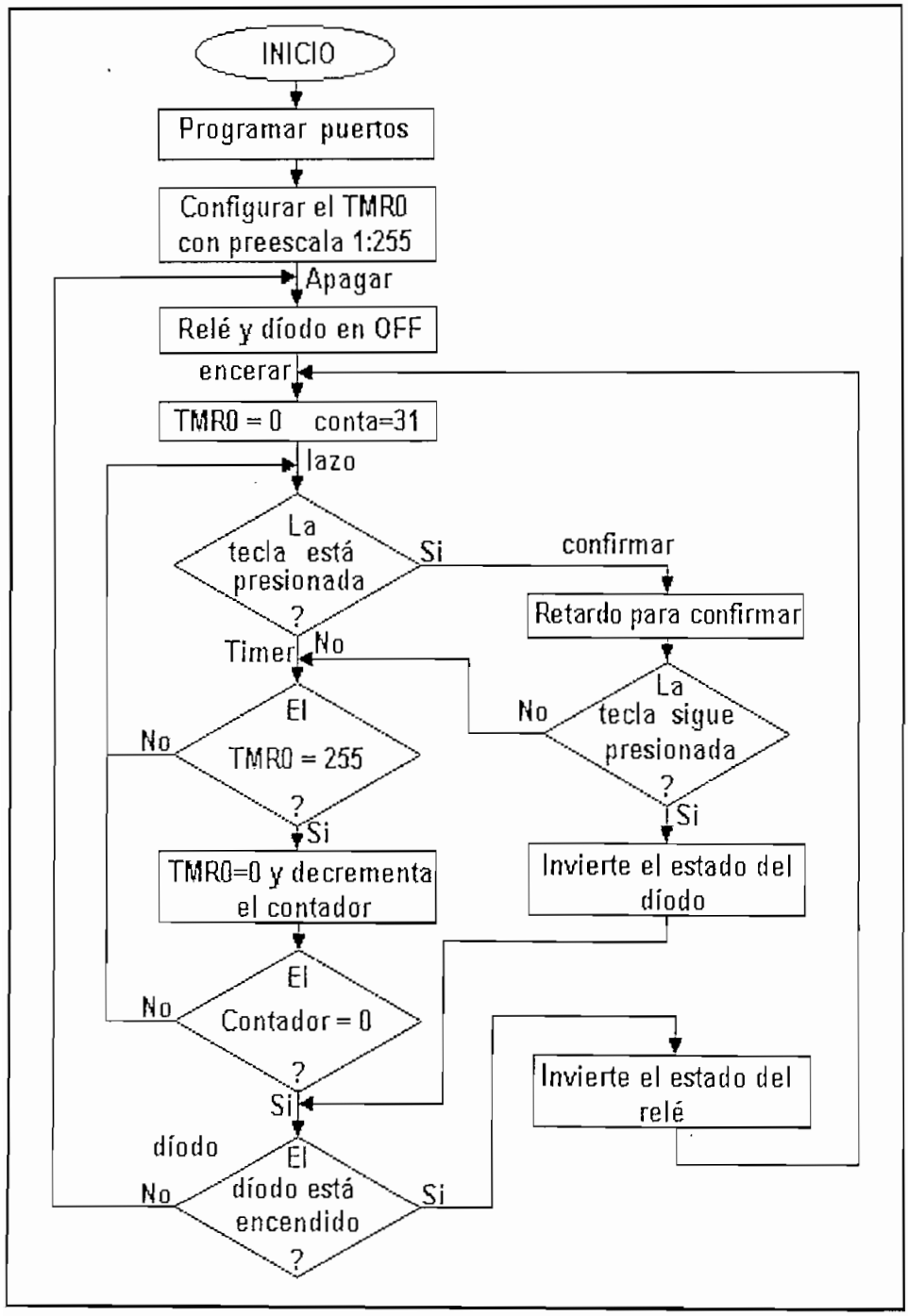


Figura 4.47 Diagrama de flujo del relé temporizado

```
;Esta práctica consiste en realizar el control de un relé temporizado
;secuencialmente. El tiempo de encendido y apagado es ajustado por el
;diseñador el momento de  programar,  con un pulsante se arranca o se
;detiene el funcionamiento del relé. Además un díodo indica el estado
;de funcion del relé.

        List    p=12c508
        include p12c508.inc

lazo1   equ     07h             ;registro utilizado en el retardo
lazo2   equ     08h             ;registro utilizado en el retardo
conta   equ     09h             ;registro contador

        org     1ffh            ;carga valor de calibraci¢n del
        movlw   080h            ;oscilador interno

        org     00h             ;pasa el valor de calibraci¢n
        movwf   OSCCAL          ;al registro osccal
        goto    inicio

;**************Subrutina de retardo de 100 milisegundos*******************
retardo movlw   D'100'          ;El registro lazo1 contiene el número de
        movwf   lazo1           ;veces que repite 1 milisegundo
repite1 movlw   D'118'          ;El registro lazo2 contiene el número de
        movwf   lazo2           ;veces que repite los 9 microsegundos
repite2 nop                     ;de retardo generados
        nop                     ;por los 6 ciclos de las instrucciones nop (6usg)
        nop                     ;más 1 ciclo de la instrucción decfsc (1usg)
        nop                     ;más 2 ciclos del salto goto (2usg)
        nop                     ;dando en total los 9usg, siendo esta la base
        nop                     ;de tiempo, por lo tanto 1sg= 9usg*110*100*10
        decfsz  lazo2           ;decrementa el reg lazo2 y salta si llega a 0
        goto    repite2         ;si lazo2 no es 0 entonces salta a repite2
        decfsz  lazo1           ;decrementa el reg lazo1 y salta si llega a 0
        goto    repite1         ;si lazo1 no es 0 entonces salta a repite
        retlw   0               ;salida de la subrutina cargando w con 0
;*************  programa principal ******************
inicio  movlw   B'11111100'     ;programa puertos
        tris    GPIO            ;deja GP1 como salida para el LED
        movlw   B'11000111'     ;configura TMR0 como temporizador
        option                  ;con preescala de 1:256
apagar  clrf    GPIO            ;apaga el díodo y el relé
encerar clrf    TMR0            ;Inicia en cero el TMR0
        movlw   d'30'           ;inicia el contador con 30
        movwf   conta           ;(este valor varía el tiempo del relé)
lazo    btfss   GPIO,2          ;Pregunta si está presionada la tecla
        goto    confir          ;salta a confirmar
timer   movf    TMR0,w          ;Comparo el temporizador con
        xorlw   D'255'          ;255 para ver si llegó a t=255*255*31=2sg
        btfss   STATUS,Z        ;prueba si fué cero el resultado
        goto    lazo            ;salta a invertir el estado del relé
        clrf    TMR0            ;borra el temporizaoor
        decfsz  conta           ;decrementa el registro contador
        goto    lazo            ;y regresa a lazo si no es cero
diodo   btfss   GPIO,0          ;pregunta el estado del díodo
        goto    apagar          ;salta a apagar el relé
        btfss   GPIO,1          ;pregunta el estado del relé
        goto    salto           ;para invertirlo
        bcf     GPIO,1          ;apaga el relé
        goto    encerar         ;y regresar a encerar
salto   bsf     GPIO,1          ;Enciende el relé
        goto    encerar         ;y regresar a encerar
confir  call    retardo         ;retardo para confirmar
        btfsc   GPIO,2          ;Pregunta si sigue presionada la tecla
        goto    timer           ;si no sigue presionada, salta a timer
        btfss   GPIO,0          ;pregunta el estado del díodo
        goto    salto1          ;para invertirlo
        bcf     GPIO,0          ;Apaga el díodo
        call    retardo         ;retardo para que suelte la tecla
        goto    diodo           ;y salta a díodo
salto1  bsf     GPIO,0          ;Enciende el díodo
        call    retardo         ;retardo para que suelte la tecla
        goto    diodo           ;y salta a díodo
        end
```

Figura 4.48 Programa del relé temporizado

Las prácticas realizadas en este capítulo son una guía para la utilización del sistema de desarrollo; pero el sistema puede ser empleado en el desarrollo de muchas otras prácticas. Además puede ser utilizado para el estudio de los poderosos PIC16F87X.

CAPITULO V
CONCLUCIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

De la experiencia del diseño y construcción del Sistema de Desarrollo y de la implementación de prácticas expuestas en este proyecto, se presentan las siguientes conclusiones:

El equipo está dotado de todos los elementos necesarios para cumplir con su objetivo: permitir la experimentación con los microcontroladores PIC. Esto acelera el proceso de aprendizaje y permite la experimentación con muchos periféricos, que pueden ser estudiados para utilizarlos en aplicaciones reales. Además, la arquitectura abierta permite al usuario controlar las conexión de los puertos del micro, lo cual, además de entrenar al usuario en su manejo, le da versatilidad al sistema.

Como se puede observar en el presupuesto adjunto en los anexos, el costo del sistema de desarrollo resultó conveniente, con respecto a equipos semejantes fabricados en otros países, esto permitirá su implementación en los laboratorios a bajo costo.

La implementación de las prácticas planteadas, demostró la versatilidad y facilidad con que se pueden armar los circuitos, además la introducción del programador evita tener que sacar el micro del sistema para ser programado, minimizando la probabilidad de deterioro por manipulación.

Gracias a su diseño, el sistema permite la experimentación con varios modelos de microcontroladores, los cuales incluyen en su arquitectura muchas características importantes, que pueden ser muy bien explotadas en aplicaciones de control.

De las prácticas realizadas se concluye que el equipo es 100% seguro. Los circuitos de programación funcionan adecuadamente y no tienen interferencia al momento de ejecutar los programas, con los circuitos de los periféricos.

5.2 RECOMENDACIONES

Se sugiere que este equipo, se ponga a disposición de los estudiantes lo antes posible, ya que el avance tecnológico en el campo de los micros es muy rápido y cada vez los fabricantes, sacan nuevos modelos, los cuales tienen mejores prestaciones, sin embargo en caso de ser necesario se podría modificar el diseño de la tarjeta para adaptar el sistema de desarrollo a esos nuevos modelos.

Se recomienda a los usuarios del sistema de desarrollo, realizar un completo estudio teórico de los microcontroladores PIC, y también de los circuitos de cada uno de los módulos del sistema, esto permitirá su manejo adecuado y una mejor utilización de los recursos que incluye el sistema.

En este proyecto se incluye un capítulo de prácticas básicas, en las cuales se encuentra desarrollado todos los pasos de diseño de un proyecto, por esto se recomienda que primero se realice estas prácticas, lo que permitirá que el estudiante se familiarice con el equipo y corrija fácilmente los errores de programación, para luego si realizar otras prácticas.

La aparición reciente de los microcontroladores de la familia PIC16F87x ha constituido un avance importante, y al inicio no fueron considerados como parte de este proyecto; sin embargo, para permitir su estudio se han incluido los zócalos necesarios para estos, entonces queda abierta la posibilidad de practicar con estos micros y se recomienda el diseño de prácticas por parte de los profesores para realizar el estudio de las importantes características que tienen estos modernos microcontroladores.

BIBLIOGRAFIA

1 MICROCHIP DATA SHEET DS35007A, PIC16F84A, Microchip technology,1998

2 MICROCHIP DATA SHEET DS30272A, PIC12C71X, Microchip technology,1997

3 MICROCHIP . DATA SHEET DS40139E, PIC12C508, Microchip technology,1999

4 MICROCHIP DATA SHEET DS30292C, PIC16F87X, Microchip technology,2001

5 MICROCONTROLADORES PIC, José María Angulo, McGRAW-HILL, 2da. Edición,1999

6 CURSO AVANZADO DE MICROCONTROLADORES PIC, Edison Duque, Cekit, 1998

7 MICROCONTROLADORES PIC16F87X, José María Angulo, McGRAW-HILL, 2da. Edición,1999

8 NATIONAL ANALOG AND INTERFACE PRODUCTS DATABOOK, National Semiconductor Corporation,1999

REFERENCIAS

1 http://www.microchip.com

2 http://www.dalsemi.com

3 http://www.xicor.com

4 http://www.natinst.com

5 http://www.national.com

ANEXOS

A continuación se describe el funcionamiento de cada una de las 35 instrucciones de los microcontroladores descritos en esta tesis, eso ayudará a entender la operación que estas cumplen.

ADDLW	Suma un valor literal al contenido del registro W.

Sintaxis: ADDLW k
Operación: (W) + k → (W)
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: C, DC, Z

Ejemplo: ADDLW 15
Antes de la instrucción W = 10
Después de instrucción W = 25

ADDWF	Suma el contenido de un registro al contenido del registro W.

Sintaxis: ADDWF f,d
Operación: (W) + (f) → (destino)
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: C, DC, Z

Ejemplo: ADDWF FSR,0

Antes de la instrucción W = 17 FSR = C2
Después de la instrucción W = D9 FSR = C2

ANDLW	Operación lógica AND entre un literal y el registro W.

Sintaxis: ANDLW k
Operación: (W) AND (k) → (W)
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: Z

Ejemplo: ANDLW 5F

Antes de la instrucción W = A3
Después de la instrucción W = 03

ANDWF Operación lógica AND entre un registro " f " y el registro W.

Sintaxis:	ANDWF f,d
Operación:	(W) AND (f) → (destino)
Ciclos de instrucción:	1
Bits del registro de estados que se afectan:	Z
Ejemplo:	ANDWF FSR,1
	Antes de la instrucción W = 17 FSR = C2
	Después de instrucción W = 17 FSR = 02

BCF Pone en cero el bit " b " del registro " f ".

Sintaxis:	BCF f,b
Operación:	0 → (f < b >)
Ciclos de instrucción:	1
Bits del registro de estados que se afectan:	Ninguno
Ejemplo:	BCF regis,7
	Antes de la instrucción regis = C7
	Después de la instrucción regis = 47

BSF Pone en uno el bit " b " del registro " f ".

Sintaxis:	BSF f,b
Operación:	1 → (f < b >)
Ciclos de instrucción:	1
Bits del registro de estados que se afectan:	Ninguno
Ejemplo:	BSF regis,7
	Antes de la instrucción regis = 0A
	Después de la instrucción regis = 8A

BTFSC Prueba el bit " b " del registro " f " y salta una línea si está en cero

Sintaxis: BTFSC f,b
Operación: salta si (f <b>) = 0
Ciclos de instrucción: 1 (2)
Bits del registro de estados que se afectan: Ninguno

Ejemplo:

```
aquí        BTFSC     regis,0
falso       GOTO      inicio
verdadero   ........
```

Antes de la instrucción: contador de programa = aquí
Después de instrucción:
si el bit 0 del registro regis = 0
    contador de programa = verdadero
si el bit 0 del registro regis = 1
    contador de programa = falso

BTFSS Prueba el bit " b " del registro " f " y salta una línea si está en uno.

Sintaxis: BTFSS f,d
Operación: salta si (f < b >) = 1
Ciclos de instrucción: 1 (2)
Bits del registro de estados que se afectan: Ninguno

Ejemplo :

```
aquí        BTFSS     regis,0
Falso       GOTO      inicio
Verdad
```

Antes de la instrucción: contador de programa = aquí
Después de la instrucción:
si el bit 0 del registro regis = 1
    contador de programa = verdad
si el bit 0 del registro regis = 0
    contador de programa = falso

CALL Llama una subrutina que está ubicada en La posición de memoria o etiqueta " k ".

Sintaxis: CALL k
Operación: (PC) + 1 → pila , k → (PC)

Ciclos de instrucción: 2
Bits del registro de estados que se afectan: Ninguno

Ejemplo: aquí CALL rutina

Antes de la instrucción contador de programa = aquí
Después de instrucción contador de programa = rutina
pila = dirección aquí + 1

CLRF Borra el contenido del registro " f " , lo carga con 00.

Sintaxis: CLRF f
Operación: 00 → (f)
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: Z

Ejemplo: CLRF regis

Antes de la instrucción regis = 5A
Después de instrucción regis = 00

CLRW Borra el contenido del registro W, lo carga con 00

Sintaxis: CLRW
Operación: 00 → (W)
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: Ninguno

Ejemplo: CLRW

Antes de la instrucción W = 5A
Después de la instrucción W = 00

CLRWDT Borra el conteo del Watchdog Timer.

Sintaxis:	CLRWDT
Operación:	00 → WDT
Ciclos de instrucción:	1
Bits del registro de estados Que se afectan:	TO, PD
Ejemplo:	CLRWDT
	Antes de la instrucción contador WDT= ?
	Después de instrucción contador WDT= 00
	TO = 1, PD = 1

COMF Complementa el contenido del registro" f "

Sintaxis:	COMF f,d
Operación:	(f) → (destino)
Ciclos de instrucción:	1
Bits del registro de estados que se afectan:	Z
Ejemplo:	COMF regis,0
	Antes de la instrucción regis = 13, W = ?
	Después de instrucción regis = 13, W = EC

DECF Decrementa el contenido del registro "f ".

Sintaxis:	DECF f,d
Operación:	(f) - 1 → (destino)
Ciclos de instrucción:	1
Bits del registro de estados que se afectan:	Z
Ejemplo:	DECF regis,1
	Antes de la instrucción regis = 13
	Después de la instrucción regis = 12

DECFSZ Decrementa el contenido del registro "f "; si el contenido queda en 00, salta una línea.

Sintaxis: DECFSZ f,d
Operación: (f) - 1 → (destino), salta si el resultado es cero
Ciclos de instrucción: 1 (2)
Bits del registro de estados
Que se afectan: Ninguno

Ejemplo:

```
aquí        DECFSZ    regis,1
            GOTO      ciclo
continúa.....
```

Antes de la instrucción contador de programa = aquí
Después de instrucción regis = regis -1
si regis = 0 , entonces
contador de programa = continua
si regis ≠ 0 , entonces
contador de programa = aquí + 1

GOTO El contador de programa salta a la dirección "k".

Sintaxis: GOTO k
Operación: k → PC
Ciclos de instrucción: 2
Bits del registro de estados
que se afectan: Ninguno

Ejemplo:

```
GOTO      ciclo
```

Antes de la instrucción contador de programa = ?
Después de instrucción contador de programa = ciclo

INCF Incrementa el contenido del registro "f ".

Sintaxis: INCF f,d
Operación: (f) + 1 → (destino)
Ciclos de instrucción: 1
Bits del registro de estados
que se afectan: Z

Ejemplo:

```
INCF      regis,1
```

Antes de la instrucción regis = 24
Después de la instrucción regis = 25

INCFSZ Incrementa el contenido del registro "f ";
si el contenido de "f " queda en 00, salta una línea.

Sintaxis: INCFSZ f,d
Operación: (f)+ 1 →(destino), salta si el resultado es cero
Ciclos de instrucción: 1 (2)
Bits del registro de estados que se afectan: Ninguno

Ejemplo:

```
aquí   DECFSZ   regis,1
       GOTO     ciclo
continúa....
```

Antes de la instrucción — contador de programa = aquí
Después de instrucción — regis = regis +1
si regis = 0 ,entonces
contador de programa = continua
si regis ≠ 0 ,entonces
contador de programa = aquí + 1

IORLW Operación lógica OR entre el registro W y el literal " k ".

Sintaxis: IORLW k
Operación: (W) OR (k) → (W)
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: Z

Ejemplo: IORLW 35

Antes de la instrucción W = 9A
Después de instrucción W = BF

IORWF Operación lógica OR entre el registro W y el registro "f".

Sintaxis: IORWF f,d
Operación: (W) OR (f) → (destino)
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: Z

Ejemplo: IORWF regis,0

Antes de la instrucción regis = 13 , W = 91
Después de la instrucción regis = 13 , W = 93

MOVLW Carga el registro W con el literal "k".

Sintaxis:	MOVLW	k
Operación:	k → (W)	
Ciclos de instrucción:	1	
Bits del registro de estados que se afectan:	Ninguno	
Ejemplo:	MOVLW	5A

Antes de la instrucción	W = ?
Después de instrucción	W =5A

MOVF Mueve el contenido del registro "f".

Sintaxis:	MOVF	f,d
Operación:	(f) → (destino)	
Ciclos de instrucción:	1	
Bits del registro de estados que se afectan:	Z	
Ejemplo:	MOVF	regis,0

Antes de la instrucción	W = ?
Después de instrucción	W = valor guardado en regis

MOVWF Mueve el contenido del registro W al registro "f".

Sintaxis:	MOVWF	f
Operación:	(W) → (f)	
Ciclos de instrucción:	1	
Bits del registro de estados que se afectan:	Ninguno	
Ejemplo:	MOVWF	OPTION

Antes de la instrucción	OPTION = ?,	W = 48
Después de la instrucción	OPTION =48,	W = 48

NOP	No hace nada.

Sintaxis: NOP
Operación: Ninguna
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: Ninguno

Ejemplo: NOP

OPTION	Carga el registro OPTION con el contenido del registro W.

Sintaxis OPTION
Operación: W → OPTION
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: Ninguno

Ejemplo: OPTION

Esta instrucción puede ser obviada, ya que el registro OPTION puede ser manejado como cualquier otro, con la instrucción MOVWF OPTION

RETFIE	Retorno de llamada a interrupción.

Sintaxis: RETFIE
Operación: Pila → contador de programa, 1 → intcon,gie
Ciclos de instrucción: 2
Bits del registro de estados que se afectan: Ninguno

Ejemplo: RETFIE

Antes de la instrucción contador de programa = ?
Después de la instrucción contador de programa = pila

RETLW Retorno cargando el registro W con el literal "k".

Sintaxis: RETLW k
Operación: k → W, pila → contador de programa
Ciclos de instrucción: 2
Bits del registro de estados que se afectan: Ninguno

Ejemplo:

```
        CALL    tabla
        •
        •
tabla   ADDWF   PC,1
        RETLW   k1
        RETLW   k2
        •
        •
        RETLW   kn
```

Antes de la instrucción W = 07
Después de la instrucción W = k7

RETURN Retorno de subrutina.

Sintaxis: RETURN
Operación: pila → contador de programa
Ciclos de instrucción: 2
Bits del registro de estados que se afectan: Ninguno
Ejemplo: RETURN

Después de la subrutina, contador de programa = pila

RLF Rota el contenido del registro "f" a la izquierda, usando el carry

Sintaxis: RLF f,d
Operación: Rota el contenido del registro "f" a la izquierda, a través del Carry. Si d=0, el resultado se guarda en W, si d=1 en "f".
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: C

Ejemplo: RLF regis,0

Antes de la instrucción C = 0, W = ?, regis = 11100110
Después de la instrucción C = 1, W = 11001100
regis=11100110

RRF Rota el contenido del registro "f" a la derecha, usando el carry

Sintaxis: RRF f,d
Operación: Rota el contenido del registro "f" a la derecha a través del carry. Si d=0,el resultado se guarda en W, si d=1 se guarda en "f"
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: C

Ejemplo: RRF regis,0

Antes de la instrucción C = 0, W =?, regis = 11100110
Después de la instrucción C = 0, W = 01110011
regis = 11100110

SLEEP Entra en modo dormido (standby).

Sintaxis: SLEEP
Operación: 00 → WDT, 1 → TO, 0 → PD
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: TO,PD

Ejemplo: SLEEP

SUBLW Resta el contenido del registro W del literal "k " (usando el método de complemento a dos).

Sintaxis: SUBLW k
Operación: k - (W) → (W)
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: C, DC, Z

Ejemplo 1: SUBLW 02

Antes de la instrucción W = 1, C = ?
Después de la instrucción W = 1, C= 1 (positivo)

Ejemplo 2: Antes de la instrucción W = 2, C = ?
Después de la instrucción W = 0, C = 1 (cero)

Ejemplo 3: Antes de la instrucción W = 3, C = ?
Después de la instrucción W = FF, C = 0 (negativo)

SUBWF	Resta el contenido del registro W del registro " f " (usando el método de complemento a dos).

Sintaxis: SUBWF f,d
Operación: (f) - (W) → (destino)
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: C, DC, Z

Ejemplo 1: SUBWF regis,1

Antes de la instrucción regis =3, W =2, C =?
Después de la instrucción regis =1, W =2, C=1(positivo)

Ejemplo 2: Antes de la instrucción regis =2, W =2, C =?
Después de la instrucción regis =0, W =2, C=1(cero)

Ejemplo 3: Antes de la instrucción regis =1, W =2, C =?
Después de la instrucción regis = FF, W =2, C=0(negativo)

SWAPF	Intercambia los cuatro bits altos y los cuatro bits bajos del registro "f".

Sintaxis: SWAPF f,d
Operación: f <3:0 > → d < 7:4 >
f <7:4 > → d < 3:0 >
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: Ninguno

Ejemplo 1: SWAPF regis,0

Antes de la instrucción regis =A5, W =?
Después de la instrucción regis =A5, W =5A

TRIS	Carga el registro TRIS (programación de los puertos como entrada /salida) con el contenido del registro W.

Sintaxis: TRIS f
Operación: W → (registro TRIS del puerto " A ó B")
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: Ninguno

Ejemplo 1: TRIS puertoA

Esta instrucción realmente no es necesaria para configurar los puertos, pero existe para guardar compatibilidad con familias anteriores.

XORLW	Operación lógica XOR entre el registro W y el literal "K".

Sintaxis: XORLW K
Operación: (W) XOR (k) → (W)
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: Z

Ejemplo: XORLW AF

Antes de la instrucción W = B5
Después de la instrucción W = 1A

XORWF	Operación lógica XOR entre el registro W y el registro "f".

Sintaxis: XORWF f,d
Operación: (W) XOR (f) → (destino)
Ciclos de instrucción: 1
Bits del registro de estados que se afectan: Z

Ejemplo: XORWF regis,1

Antes de la instrucción regis = AF, W = B5
Después de la instrucción regis = 1A, W = B5

PRESUPUESTO

PRESUPUESTO

DESCRIPCIÓN	CANT.	PRECIO UNIT.	PRECIO TOTAL
Fuente de alimentación de 5V			
Puente de diodos de 1A	1	0,2	0,2
Bornera de dos terminales	1	0,35	0,35
Condensador de 1000uF 25V	1	0,4	0,4
Regulador de voltaje de 5V (LM7805)	1	0,4	0,4
Disipador pequeño de calor para el LM7805	1	0,3	0,3
Conector redondo tipo zócalo	4	0,01	0,04
Fuente de alimentación de 13,4V Y 12V			
Puente de diodos de 1A	1	0,2	0,2
Bornera de dos terminales	1	0,35	0,35
Condensador de 1000uF 25V	1	0,4	0,4
Regulador de voltaje de 12V (LM7812)	1	0,4	0,4
Conector redondo tipo zócalo	2	0,01	0,02
Diodos rectificadores 1N4004	4	0,06	0,24
Módulo de microcontroladores			
Zócalo de 18 pines redondos	1	1	1
Zócalo de 8 pines redondos	1	0,8	0,8
Zócalo de 28 pines redondos	1	0,88	0,88
Zócalo de 40 pines redondos	1	1,28	1,28
Oscilador de 32.768Hz	1	0,76	0,76
Oscilador de 4MHz	1	1,5	1,5
Condensador de 56pF.	2	0,2	0,4
Condensador de 20pF.	2	0,2	0,4
Jumpers de 3 terminales	2	0,2	0,4
Conectores para Jumpers	2	0,05	0,1
Switch doble de 2 posiciones, 6 terminales	1	0,5	0,5
Conector redondo tipo zócalo	188	0,01	1,88
Módulo de leds			
Diodos leds rojos	8	0,09	0,72
Resistencias de 330ohmios 1/4w	8	0,03	0,24
Buffer ULN2803A (NTE2018)	1	4,14	4,14
Zócalo de 18 pines	1	0,2	0,2
Conector redondo tipo zócalo	24	0,01	0,24
Módulo de displays			
Displays de 7 segmentos de cátodo común	4	1	4
Transistores 2N3904	4	0,08	0,32
Resistencias de 2,7K 1/4w	4	0,03	0,12
Resistencias de 100ohmios 1/4w	7	0,03	0,21
Decodificador BDC a 7 segmentos DM9368M (ECG8368)	1	0,9	0,9
Zócalo de 16 pines	1	0,08	0,08
Conector redondo tipo zócalo	16	0,01	0,16
Módulo LCD			
LCD 2 líneas 16 caracteres	1	19,5	19,5
Potenciómetro cuadrado de 10K	1	0,5	0,5
Resistencia de 56ohmios 1/2W	1	0,05	0,05
Conector redondo tipo zócalo	28	0,01	0,28
Módulo de Dipswitch			
Resistencias de 1K a 1/4w	8	0,03	0,24
Dipswitch	1	0,95	0,95
Conector redondo tipo zócalo	16	0,01	0,16

DESCRIPCIÓN	CANT.	PRECIO UNIT.	PRECIO TOTAL
Módulo de teclado matricial			
Teclado de 16 botones	1	12,5	12,5
Resistencia de 2,7K a 1/4W	4	0,03	0,12
Conector redondo tipo zócalo	16	0,01	0,16
Módulo de pulsadores			
Pulsantes grandes tipo mouse	2	0,3	0,6
Resistencias de 1K a 1/4w	2	0,03	0,06
Conector redondo tipo zócalo	2	0,01	0,02
Módulo interface RS232			
Max232	1	2,3	2,3
Zócalo de 16 pines	1	0,35	0,35
Condensadores de 10uF a 35 vóltios	5	0,15	0,75
Bonera de 3 terminales	1	0,5	0,5
Conector redondo tipo zócalo	2	0,01	0,02
Módulo de interface 422/485			
Integrado SN75176	2	3,8	7,6
Zócalo de 8 pines	2	0,2	0,4
Resistencia de 4,7K a 1/4W	1	0,03	0,03
Bornera de 5 terminales	1	0,9	0,9
Conector redondo tipo zócalo	3	0,01	0,03
Módulo de memoria serial			
Memoria serial 24LC04B/P (4K)	1	1,02	1,02
Zócalo de 8 pines	1	0,2	0,2
Resistencia de 4,7K a 1/4W	1	0,03	0,03
Switch simple de 2 posiciones, 3 terminales	1	0,35	0,35
Conector redondo tipo zócalo	2	0,01	0,02
Módulo de entradas análogas			
Potenciómetro cuadrado de 5K	1	0,5	0,5
Bornera de tres terminales	1	0,35	0,35
Termistor LM35	1	1,54	1,54
Conector redondo tipo zócalo	2	0,01	0,02
Módulo de potenciómetro digital			
Potenciómetro digital X9C104P (100K)	1	5	5
Zócalo de 8 pines	1	0,2	0,2
Resistencia de 2,7K a 1/4W	1	0,03	0,03
Bornera de dos terminales	1	0,35	0,35
Conector redondo tipo zócalo	3	0,01	0,03
Módulo de zumbador piezoeléctrico			
Zumbador	1	0,87	0,87
Conector redondo tipo zócalo	2	0,01	0,02
Salida PWM para controlar un pequeño motor			
Resistencia de 2,7K a 1/4W	2	0,03	0,06
Transistores 2N3904	1	0,08	0,08
Díodo rectificador 1N4004	1	0,06	0,06
Conector redondo tipo zócalo	1	0,01	0,01

DESCRIPCIÓN	CANT.	PRECIO UNIT.	PRECIO TOTAL
Convertidor de digital a análogo PWM			
Amplificador operacional LM358 (ECG 928)	1	0,45	0,45
Zócalo de 8 pines	1	0,2	0,2
Resistencia de 1K 1/4W	1	0,03	0,03
Resistencia de 100K a 1/4w	1	0,03	0,03
Condensador de 10uF 50V	1	0,15	0,15
Bornera de dos terminales	1	0,35	0,35
Conector redondo tipo zócalo	1	0,01	0,01
Módulo de reloj calendario			
Reloj calendario DS1307	1	4,34	4,34
Zócalo de 8 pines	1	0,2	0,2
Oscilador de 32.768Hz	1	0,76	0,76
Batería estandar de lítio	1	0,8	0,8
Porta batería	1	0,7	0,7
Resistencia de 4,7K a 1/4W	2	0,03	0,06
Transistores 2N3904	1	0,08	0,08
Resistencia de 470 ohmios 1/4w	1	0,03	0,03
Díodo led verde	1	0,09	0,09
Conector redondo tipo zócalo	2	0,01	0,02
Módulo de programador en circuito			
Conector DB9 hembra para montaje en impreso	1	2,38	2,38
Buffer para 30V. ECG7407	1	1,2	1,2
Resistencia de 10K a 1/4W	5	0,03	0,15
Resistencia de 1K a 1/4W	1	0,03	0,03
Resistencia de 4,7K a 1/4W	1	0,03	0,03
Resistencia de 100ohmios a 1/4W	2	0,03	0,06
Pulsantes pequeños tipo mouse	1	0,25	0,25
Díodo led rojo	1	0,09	0,09
Módulo de relé			
Relé estandar de 12V 10A	1	0,9	0,9
Díodo rectificador 1N4004	1	0,06	0,06
Transistores 2N3904	1	0,08	0,08
Resistencia 2,7K	1	0,03	0,03
Conectores redondos tipo zócalo	2	0,01	0,02
Bornera de tres terminales	1	0,35	0,35
Accesorios			
Placa de circuito impreso 20 x30	1	60	60
Switch principal con indicador luminoso de encendido	1	0,8	0,8
Porta fusible	1	0,7	0,7
Fusible de 1A	1	0,1	0,1
Tranformador de con salidas de 7,5V y 13V.	1	4	4
Chasis metálico con tablas de acrilico	1	10	10
Pasa cable	1	0,5	0,5
Cable de 2,5m con enchufe	1	1	1
SUMA TOTAL:		**(U.S.D)**	**174,3**

DIAGRAMAS DE CIRCUITO IMPRESO

HOJAS TÉCNICAS

Recommended System Management Alternative: X4043

XICOR*

4K X24C04 512 x 8 Bit

Serial EEPROM

FEATURES

- 2.7V to 5.5V power supply versions
- Low power CMOS
  - —Active read current less than 1 mA
  - —Active write current less than 1.5 mA
- Internally organized 512 x 8
- 2-wire serial interface
  - —Bidirectional data transfer protocol
  - —Schmitt trigger input noise suppression 400kHz across V_{CC} range
- Sixteen byte page write mode
  - —Minimizes total write time per byte
- Self-timed write cycle
  - —Typical write cycle time of 5 ms
- High reliability
  - —Endurance: 1,000,000 cycles
  - —Data retention: 100 years
- 8-pin SOIC

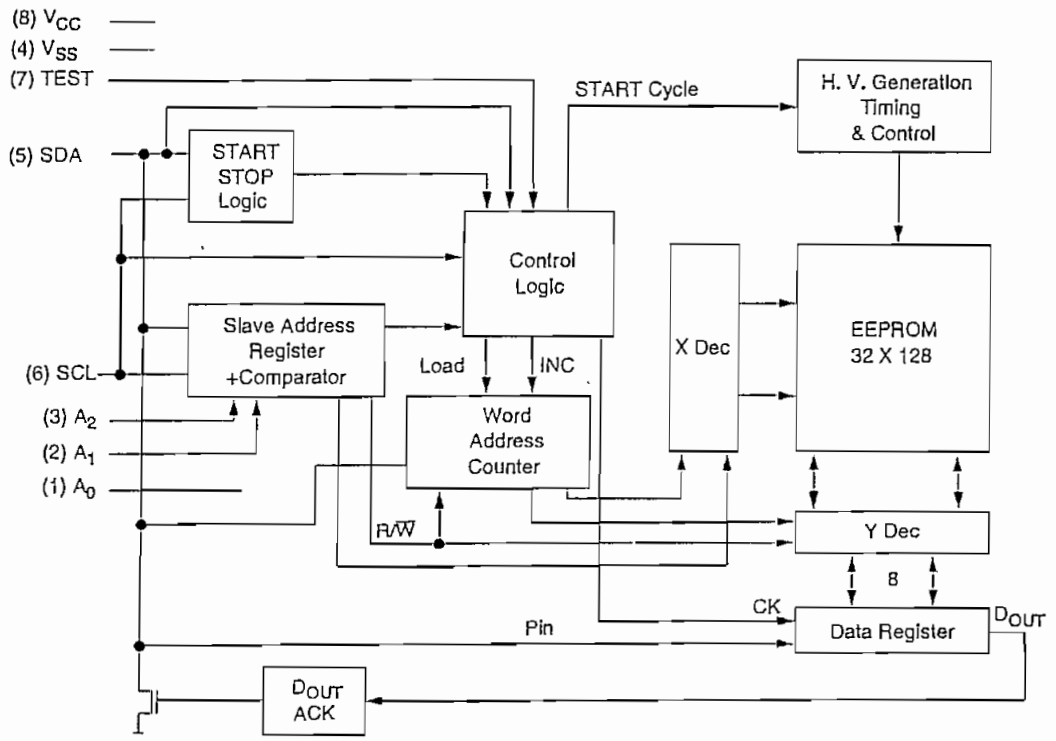
DESCRIPTION

The X24C04 is a CMOS 4096 bit serial EEPROM, internally organized 512 x 8. The X24C04 features a serial interface and software protocol allowing operation on a simple two wire bus.

The X24C04 is fabricated with Xicor's advanced CMOS Textured Poly Floating Gate Technology.

The X24C04 utilizes Xicor's proprietary DirectWrite™ cell, providing a minimum endurance of 1,000,000 cycles and a minimum data retention of 100 years.

BLOCK DIAGRAM



PIN DESCRIPTIONS

Serial Clock (SCL)

The SCL input is used to clock all data into and out of the device.

Serial Data (SDA)

SDA is a bidirectional pin used to transfer data into and out of the device. It is an open drain output and may be wire-ORed with any number of open drain or open collector outputs.

An open drain output requires the use of a pull-up resistor. For selecting typical values, refer to the Pull-Up Resistor selection graph at the end of this data sheet.

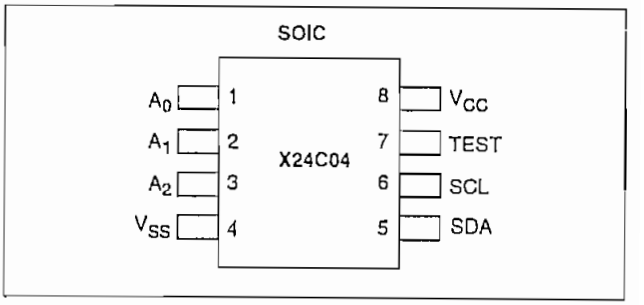
Address (A_0, A_1, A_2)

A0 is a no connect. The Address inputs (A_1, A_2) are used to set the appropriate bits of the seven bit slave address. These inputs can be used static or actively driven. If used statically they must be tied to V_{SS} or V_{CC} as appropriate. If driven they must be driven to V_{SS} or to V_{CC}.

PIN NAMES

Symbol	Description
A_0–A_2	Address Inputs
SDA	Serial Data
SCL	Serial Clock
TEST	Test Input (Hold at V_{SS})
V_{SS}	Ground
V_{CC}	Supply Voltage

PIN CONFIGURATION



DEVICE OPERATION

The X24C04 supports a bidirectional bus oriented protocol. The protocol defines any device that sends data onto the bus as a transmitter, and the receiving device as the receiver. The device controlling the transfer is a master and the device being controlled is the slave. The master will always initiate data transfers, and provide the clock for both transmit and receive operations. Therefore, the X24C04 will be considered a slave in all applications.

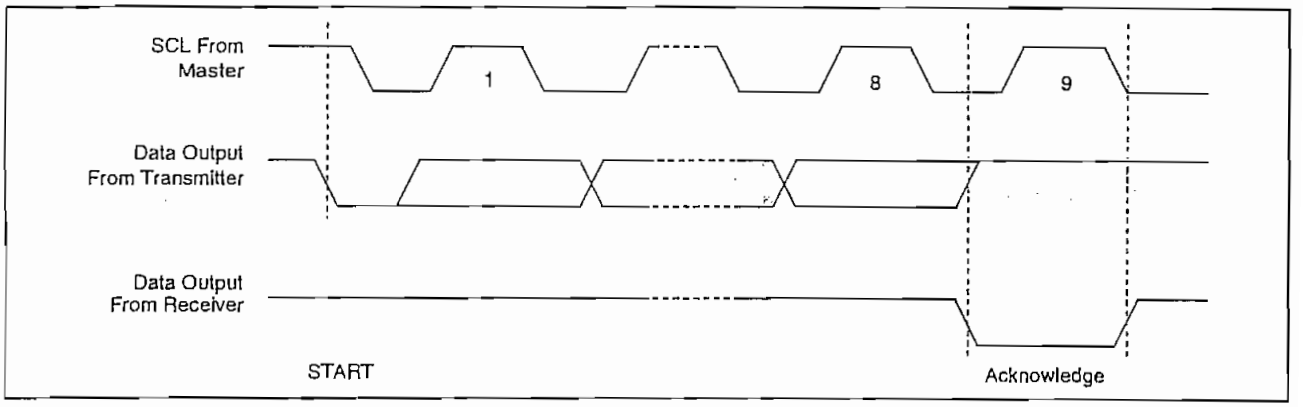
Clock and Data Conventions

Data states on the SDA line can change only during SCL LOW. SDA state changes during SCL HIGH are reserved for indicating start and stop conditions. Refer to Figures 1 and 2.

Start Condition

All commands are preceded by the start condition, which is a HIGH to LOW transition of SDA when SCL is HIGH. The X24C04 continuously monitors the SDA and SCL lines for the start condition and will not respond to any command until this condition has been met.

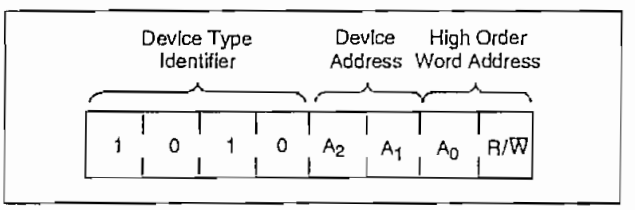
Figure 3. Acknowledge Response From Receiver



DEVICE ADDRESSING

Following a start condition the master must output the address of the slave it is accessing. The most significant four bits of the slave are the device type identifier (see Figure 4). For the X24C04 this is fixed as 1010[B].

Figure 4. Slave Addressing



The next two significant bits address a particular device. A system could have up to four X24C04 devices on the bus (see Figure 10). The four addresses are defined by the state of the A_1 and A_2 inputs.

The next bit of the slave address is an extension of the array's address and is concatenated with the eight bits of address in the word address field, providing direct access to the whole 512 x 8 array.

Note: This bit is part of word address. Not related to device address pin A_0.

The last bit of the slave address defines the operation to be performed. When set to one a read operation is selected, when set to zero a write operation is selected.

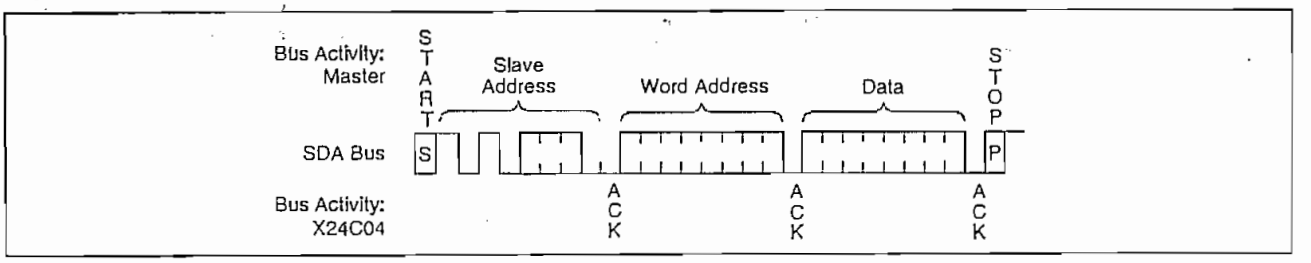
Following the start condition, the X24C04 monitors the SDA bus comparing the slave address being transmitted with its slave address (device type and state of A_1 and A_2 inputs). Upon a correct compare the X24C04 outputs an acknowledge on the SDA line. Depending on the state of the R/W bit, the X24C04 will execute a read or write operation.

WRITE OPERATIONS

Byte Write

For a write operation, the X24C04 requires a second address field. This address field is the word address, comprised of eight bits, providing access to any one of the 512 words of memory. Upon receipt of the word address the X24C04 responds with an acknowledge, and awaits the next eight bits of data, again responding with an acknowledge. The master then terminates the transfer by generating a stop condition, at which time the X24C04 begins the internal write cycle to the nonvolatile memory. While the internal write cycle is in progress the X24C04 inputs are disabled, and the device will not respond to any requests from the master. Refer to Figure 5 for the address, acknowledge and data transfer sequence.

Figure 5. Byte Write

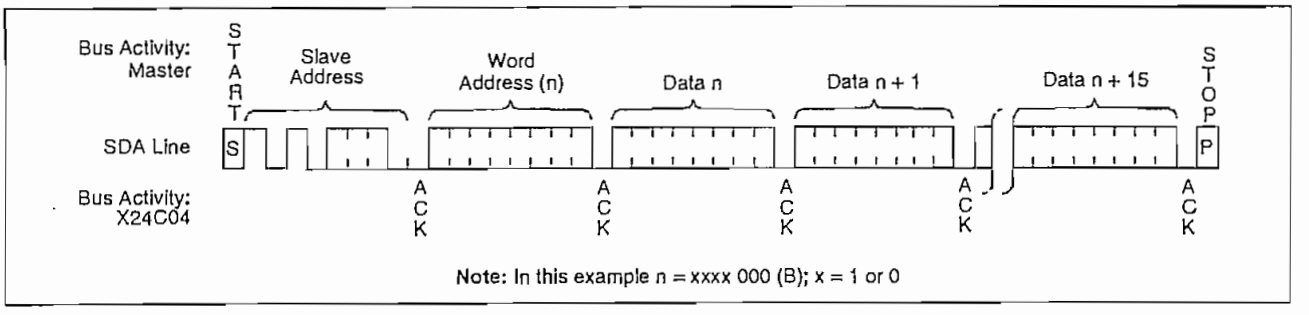


Page Write

The X24C04 is capable of a sixteen byte page write operation. It is initiated in the same manner as the byte write operation, but instead of terminating the write cycle after the first data word is transferred, the master can transmit up to fifteen more words. After the receipt of each word, the X24C04 will respond with an acknowledge.

After the receipt of each word, the four low order address bits are internally incremented by one. The high order five bits of the address remain constant. If the master should transmit more than sixteen words prior to generating the stop condition, the address counter will "roll over" and the previously written data will be overwritten. As with the byte write operation, all inputs are disabled until completion of the internal write cycle. Refer to Figure 6 for the address, acknowledge and data transfer sequence.

Figure 6. Page Write

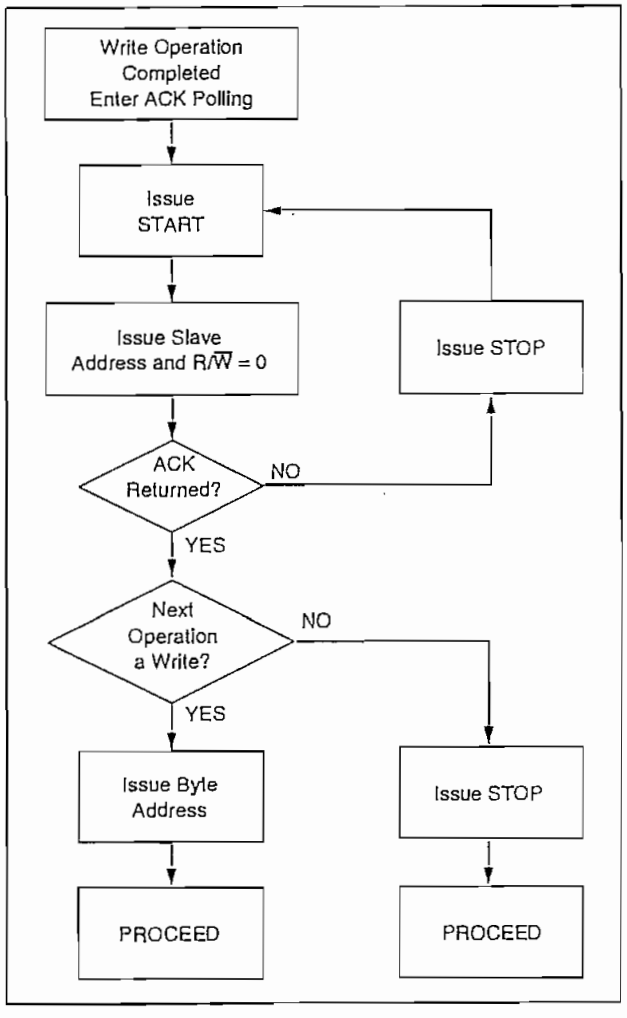


Note: In this example n = xxxx 000 (B); x = 1 or 0

Acknowledge Polling

The disabling of the inputs can be used to take advantage of the typical 5 ms write cycle time. Once the stop condition is issued to indicate the end of the host's write operation, the X24C04 initiates the internal write cycle. ACK polling can be initiated immediately. This involves issuing the start condition, followed by the slave address for a write operation. If the X24C04 is still busy with the write operation no ACK will be returned. If the X24C04 has completed the write operation an ACK will be returned, and the host can then proceed with the next read or write operation. Refer to Flow 1.

Flow 1. ACK Polling Sequence



READ OPERATIONS

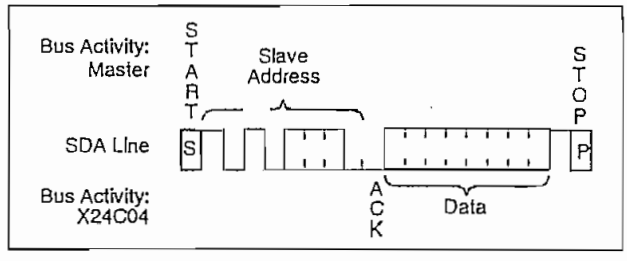
Read operations are initiated in the same manner as write operations with the exception that the R/W bit of the slave address is set to a one. There are three basic read operations: current address read, random read and sequential read.

It should be noted that the ninth clock cycle of the read operation is not a "don't care." To terminate a read operation, the master must either issue a stop condition during the ninth cycle, or hold SDA HIGH during the ninth clock cycle and then issue a stop condition.

Current Address Read

Internally the X24C04 contains an address counter that maintains the address of the last word accessed, incremented by one. Therefore, if the last access (either a read or write) was to address n, the next read operation would access data from address n + 1. Upon receipt of the slave address with the R/W bit set to one, the X24C04 issues an acknowledge and transmits the eight bit word. The read operation is terminated by the master by not responding with an acknowledge, and issuing a stop condition. Refer to Figure 7 for the sequence of address, acknowledge and data transfer.

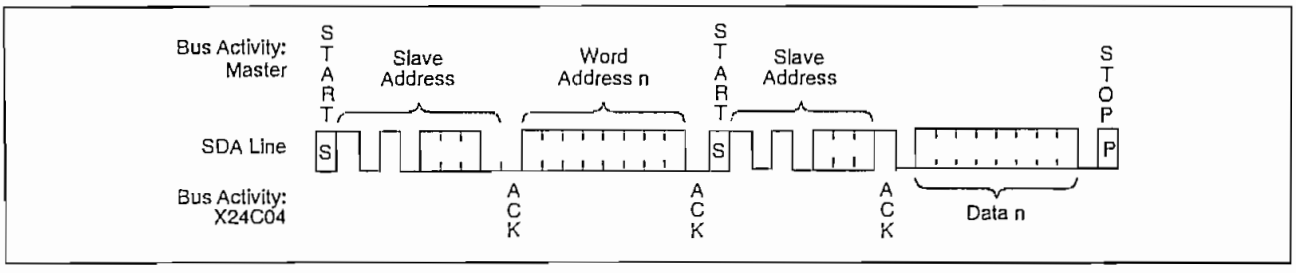
Figure 7. Current Address Read



Random Read

Random read operations allow the master to access any memory location in a random manner. Prior to issuing the slave address with the R/W bit set to one, the master must first perform a "dummy" write operation. The master issues the start condition, and the slave address followed by the word address it is to read. After the word address acknowledge, the master immediately reissues the start condition and the slave address with the R/W bit set to one. This will be followed by an acknowledge from the X24C04 and then by the eight bit word. The read operation is terminated by the master by not responding with an acknowledge, and issuing a stop condition. Refer to Figure 8 for the address, acknowledge and data transfer sequence.

Figure 8. Random Read



Sequential Read

Sequential Read can be initiated as either a current address read or random access read. The first word is transmitted as with the other modes, however, the master now responds with an acknowledge, indicating it requires additional data. The X24C04 continues to output data for each acknowledge received. The read operation is terminated by the master; by not responding with an acknowledge and by issuing a stop condition.

The data output is sequential, with the data from address n followed by the data from n + 1. The address counter for read operations increments all address bits, allowing the entire memory contents to be serially read during one operation. At the end of the address space (address 511), the counter "rolls over" to address 0 and the X24C04 continues to output data for each acknowledge received. Refer to Figure 9 for the address, acknowledge and data transfer sequence.

Figure 9. Sequential Read

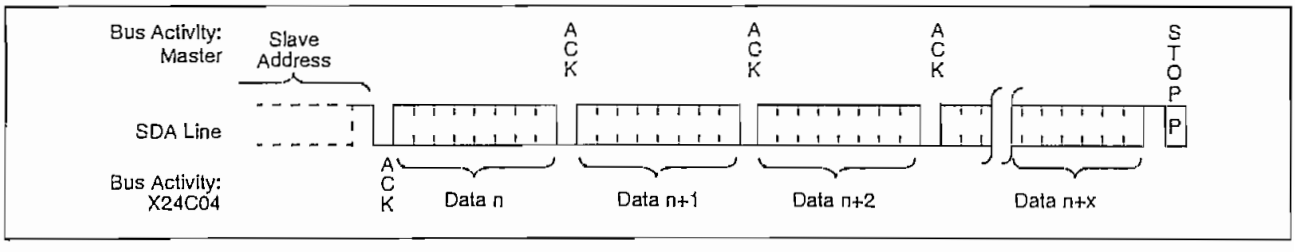
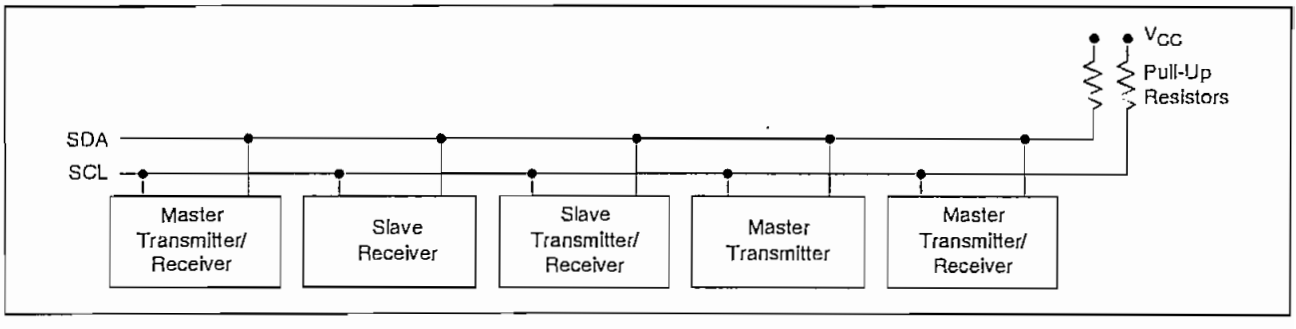


Figure 10. Typical System Configuration



ABSOLUTE MAXIMUM RATINGS

Temperature under bias .....–65 to +135°C
Storage temperature .....–65 to +150°C
Voltage on any pin with
respect to V_{SS} .....–1.0V to +7.0V
D.C. output current ..... 5 mA
Lead temperature (soldering, 10 Seconds) .....300°C

COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only; functional operation of the device (at these or any other conditions above those indicated in the operational sections of this specification) is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED OPERATING CONDITIONS

Temperature	Min.	Max.
Commercial	0°C	70°C
Industrial	–40°C	+85°C

Supply Voltage	Limits
X24C04-2.7	2.7V to 5.5V

D.C. OPERATING CHARACTERISTICS (Over recommended operating conditions unless otherwise specified)

Symbol	Parameter	Limits		Unit	Test Conditions
		Min.	Max.		
I_{CC1}	V_{CC} supply current (Read)		1	mA	SCL = V_{CC} x 0.1/Vd x 0.9 Levels @ 400 kHz, SDA = Open, All Other Inputs = GND or V_{CC} – 0.3V
I_{CC2}	V_{CC} supply current (Write)		1.5		
I_{SB}[1]	V_{CC} standby current		10	µA	SCL = SDA = V_{CC} – 0.3V, All Other Inputs = GND or V_{CC}, V_{CC} = 5.5V
I_{LI}	Input leakage current		10	µA	V_{IN} = GND to V_{CC}
I_{LO}	Output leakage current		10	µA	V_{OUT} = GND to V_{CC}
V_{IL}[2]	Input low voltage	–1.0	V_{CC} x 0.3	V	
V_{IH}[2]	Input high voltage	V_{CC} x 0.7	V_{CC} + 0.5	V	
V_{OL}	Output low voltage		0.4	V	I_{OL} = 3 mA, $V_{CC} \geq$ 2.7V

CAPACITANCE T_A = 25°C, f = 1.0MHz, V_{CC} = 5V

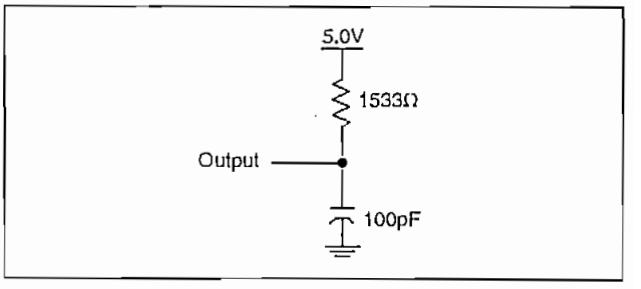
Symbol	Parameter	Max.	Unit	Test Conditions
$C_{I/O}$[3]	Input/output capacitance (SDA)	8	pF	$V_{I/O}$ = 0V
C_{IN}[3]	Input capacitance (A_0, A_1, A_2, SCL)	6	pF	V_{IN} = 0V

Notes: (1) Must perform a stop command prior to measurement.
(2) V_{IL} min. and V_{IH} max. are for reference only and are not tested.
(3) This parameter is periodically sampled and not 100% tested.

A.C. CONDITIONS OF TEST

Input pulse levels	V_{CC} x 0.1 to V_{CC} x 0.9
Input rise and fall times	10 ns
Input and output timing levels	V_{CC} x 0.5

EQUIVALENT A.C. LOAD CIRCUIT



A.C. CHARACTERISTICS (over recommended operating conditions unless otherwise specified)

Read & Write Cycle Limits

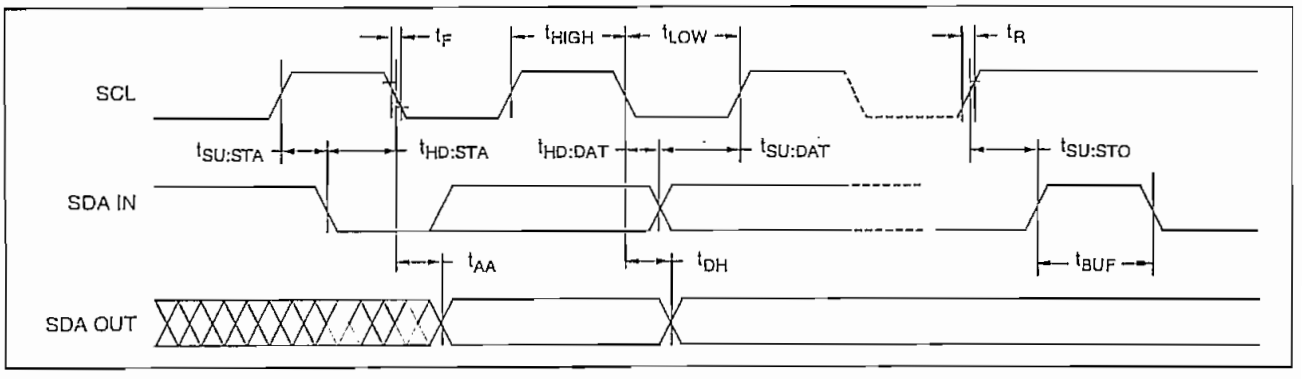
Symbol	Parameter	Min.	Max.	Unit
f_{SCL}	SCL clock frequency	0	400	kHz
T_I	Noise suppression time constant at SCL, SDA inputs	50		ns
td	SCL low to SDA data out valid	0.1	0.9	µs
t_{BUF}	Time the bus must be free before a new transmission can start	1.2		µs
$t_{HD:STA}$	Start condition hold time	0.6		µs
t_{LOW}	Clock low period	1.2		µs
t_{HIGH}	Clock high period	0.6		µs
$t_{SU:STA}$	Start condition setup time (for a repeated start condition)	0.6		µs
$t_{HD:DAT}$	Data In hold time	0		µs
$t_{SU:DAT}$	Data In setup time	100		ns
t_R	SDA and SCL rise time		300	µs
t_F	SDA and SCL fall time		300	ns
$t_{SU:STO}$	Stop condition setup time	0.6		µs
t_{DH}	Data out hold time	50	300	ns

POWER-UP TIMING

Symbol	Parameter	Max.	Unit
t_{PUR}[4]	Power-up to read operation	1	ms
t_{PUW}[4]	Power-up to write operation	5	ms

Note: (4) t_{PUR} and t_{PUW} are the delays required from the time V_{CC} is stable until the specified operation can be initiated. These parameters are periodically sampled and not 100% tested.

Bus Timing



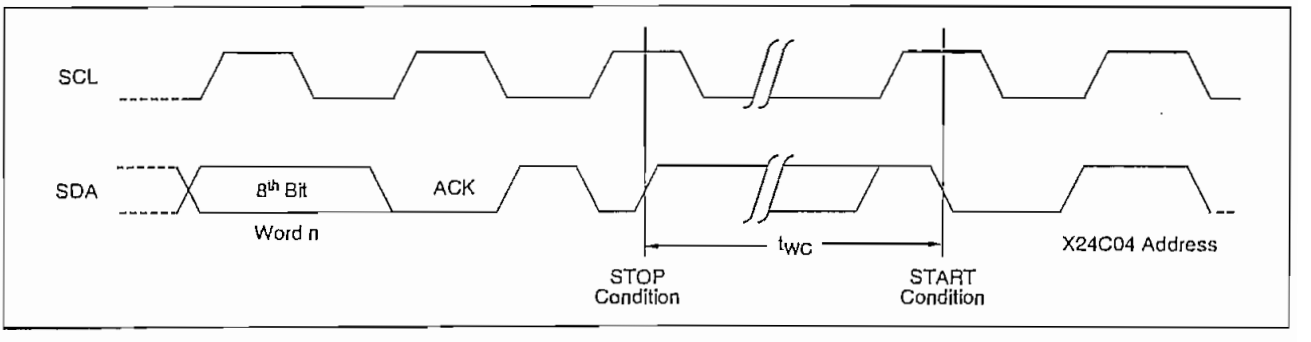
Write Cycle Limits

Symbol	Parameter	Min.	Typ.(5)	Max.	Unit
t_{WC}(6)	Write Cycle Time		5	10	ms

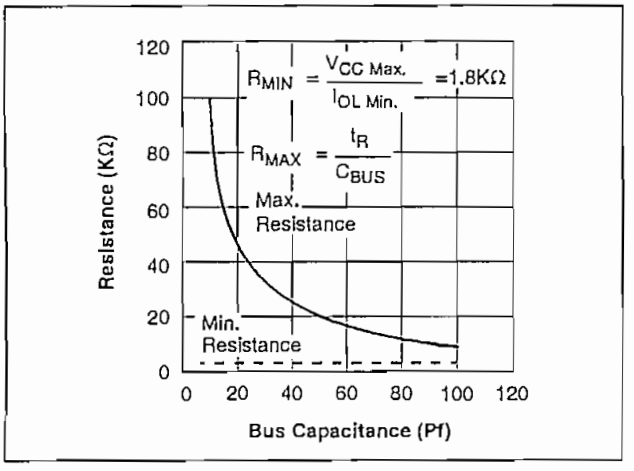
Notes: (5) Typical values are for $T_A = 25°C$ and nominal supply voltage (5V).
(6) t_{WR} is the minimum cycle time to be allowed from the system perspective unless polling techniques are used. It is the maximum time the device requires to automatically complete the internal write operation.

The write cycle time is the time from a valid stop condition of a write sequence to the end of the internal erase/write cycle. During the write cycle, the X24C04 bus interface circuits are disabled, SDA is allowed to remain HIGH, and the device does not respond to its slave address.

Write Cycle Timing



Guidelines for Calculating Typical Values of Bus Pull-Up Resistors

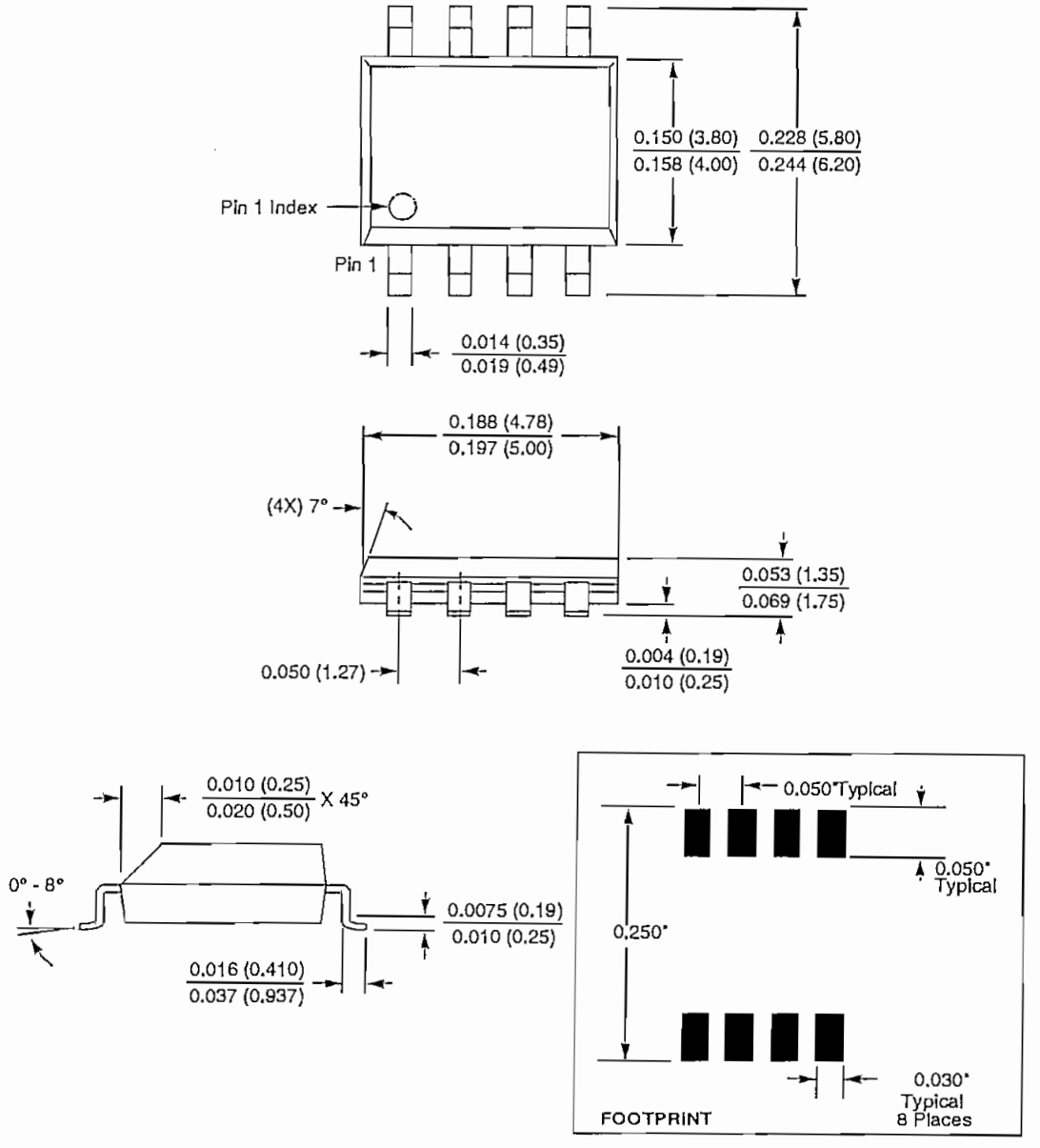


SYMBOL TABLE

WAVEFORM	INPUTS	OUTPUTS
	Must be steady	Will be steady
	May change from Low to High	Will change from Low to High
	May change from High to Low	Will change from High to Low
	Don't Care: Changes Allowed	Changing: State Not Known
	N/A	Center Line is High Impedance

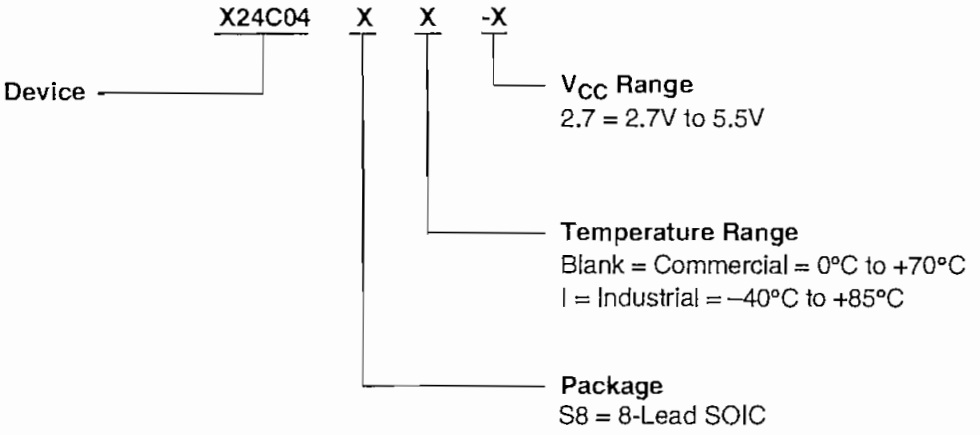
PACKAGING INFORMATION

8-Lead Plastic Small Outline Gull Wing Package Type S

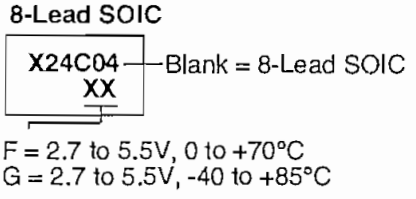


NOTE: ALL DIMENSIONS IN INCHES (IN PARENTHESES IN MILLIMETERS)

Ordering Information



Part Mark Convention



LIMITED WARRANTY

©Xicor, Inc. 2000 Patents Pending

Devices sold by Xicor, Inc. are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Xicor, Inc. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Xicor, Inc. makes no warranty of merchantability or fitness for any purpose. Xicor, Inc. reserves the right to discontinue production and change specifications and prices at any time and without notice.

Xicor, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in a Xicor, Inc. product. No other circuits, patents, or licenses are implied.

TRADEMARK DISCLAIMER:

Xicor and the Xicor logo are registered trademarks of Xicor, Inc. AutoStore, Direct Write, Block Lock, SerialFlash, MPS, and XDCP are also trademarks of Xicor, Inc. All others belong to their respective owners.

U.S. PATENTS

Xicor products are covered by one or more of the following U.S. Patents: 4,326,134; 4,393,481; 4,404,475; 4,450,402; 4,486,769; 4,488,060; 4,520,461; 4,533,846; 4,599,706; 4,617,652; 4,668,932; 4,752,912; 4,829,482; 4,874,967; 4,883,976; 4,980,859; 5,012,132; 5,003,197; 5,023,694; 5,084,667; 5,153,880; 5,153,691; 5,161,137; 5,219,774; 5,270,927; 5,324,676; 5,434,396; 5,544,103; 5,587,573; 5,835,409; 5,977,585. Foreign patents and additional patents pending.

LIFE RELATED POLICY

In situations where semiconductor component failure may endanger life, system designers using this product should design the system with appropriate error detection and correction, redundancy and back-up features to prevent such an occurrence.

Xicor's products are not authorized for use in critical components in life support devices or systems.

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

DALLAS SEMICONDUCTOR

DS1307/DS1308
64 X 8 Serial Real Time Clock

www.dalsemi.com

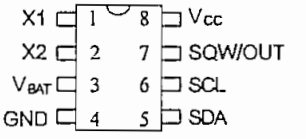
FEATURES

- Real time clock counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap year compensation valid up to 2100
- 56 byte nonvolatile RAM for data storage
- 2-wire serial interface
- Programmable squarewave output signal
- Automatic power-fail detect and switch circuitry
- Consumes less than 500 nA in battery backup mode with oscillator running
- Optional industrial temperature range -40°C to +85°C (IND) available for DS1307 and DS1308
- DS1307 available in 8-pin DIP or SOIC
- DS1308 available in 36-pin SMD BGA (Ball Grid Array)
- DS1308 accuracy is better than ±2 minute/month at 25°C

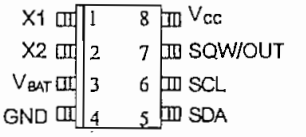
ORDERING INFORMATION

DS1307	Serial Timekeeping Chip; 8-pin DIP
DS1307Z	Serial Timekeeping Chip; 8-pin SOIC (150-mil)
DS1307N	8-pin DIP (IND)
DS1307ZN	8-pin SOIC (IND)
DS1308	36-pin BGA
DS1308N	36-pin BGA (IND)

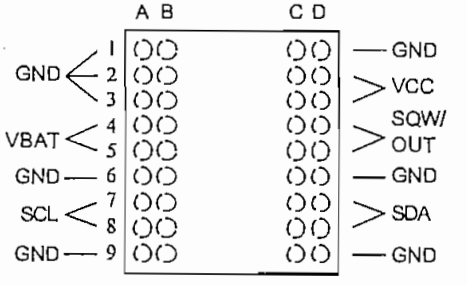
PIN ASSIGNMENT



DS1307 8-Pin DIP (300-mil)



DS1307Z 8-Pin SOIC (150-mil)



DS1308 36-Pin SMD BGA
(TOP VIEW)

PIN DESCRIPTION DS1307/DS1308

V_{CC}	- Primary Power Supply
X1, X2	- 32.768 kHz Crystal Connection
V_{BAT}	- +3 Volt Battery Input
GND	- Ground
SDA	- Serial Data
SCL	- Serial Clock
SQW/OUT	- Square wave/Output Driver

DS1308 PIN IDENTIFIER

V_{CC}	- C2, C3, D2, D3
V_{BAT}	- A4, A5, B4, B5
SDA	- C7, C8, D7, D8
SCL	- A7, A8, B7, B8
SQW/OUT	- C4, C5, D4, D5
GND	- All Remaining Balls

DESCRIPTION

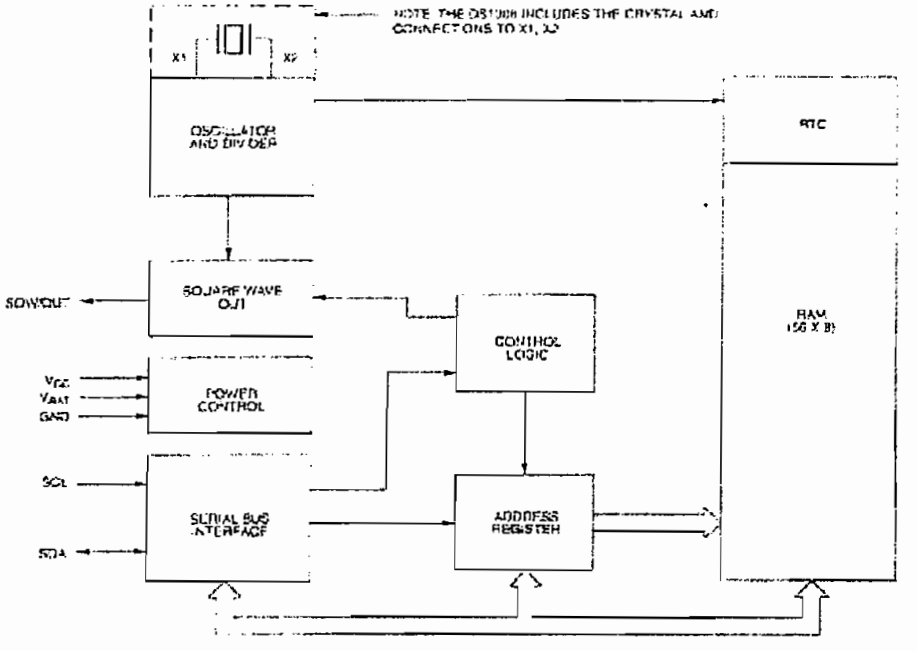
The DS1307 Serial Real Time Clock is a low power, full BCD clock/calendar plus 56 bytes of nonvolatile SRAM. Address and data are transferred serially via a 2-wire bi-directional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with less than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power sense circuit which detects power failures and automatically switches to the battery supply.

The DS1308 incorporates the DS1307 chip with a 32.768 kHz crystal in a surface mountable, 36-pin ball grid array package (BGA). The close proximity of the embedded crystal to the high impedance crystal input pins on the DS1307 minimizes capacitive loading and noise injection problems associated with many other oscillator designs. The total area required for installation is less than that of one United States dime: thus, minimizing PCB space required.

OPERATION

The DS1307/1308 operates as a slave device on the serial bus. Access is obtained by implementing a START condition and providing a device identification code followed by a register address. Subsequent registers can be accessed sequentially until a STOP condition is executed. When V_{CC} falls below 1.25 x V_{BAT} the device terminates an access in progress and resets the device address counter. Inputs to the device will not be recognized at this time to prevent erroneous data from being written to the device from an out of tolerance system. When V_{CC} falls below V_{BAT} the device switches into a low current battery backup mode. Upon power up, the device switches from battery to V_{CC} when V_{CC} is greater than V_{BAT} +0.2V and recognizes inputs when V_{CC} is greater than 1.25 x V_{BAT}. The block diagram in Figure 1 shows the main elements of the Serial Real Time Clock.

DS1307/DS1308 BLOCK DIAGRAM Figure 1



SIGNAL DESCRIPTIONS

V_{CC}, **GND** - DC power is provided to the device on these pins. V_{CC} is the +5 volt input. When 5 volts is applied within normal limits, the device is fully accessible and data can be written and read. When a 3-volt battery is connected to the device and V_{CC} is below 1.25 x V_{BAT}, reads and writes are inhibited. However, the Timekeeping function continues unaffected by the lower input voltage. As V_{CC} falls below V_{BAT} the RAM and timekeeper are switched over to the external power supply (nominal 3.0V DC) at V_{BAT}.

V_{BAT} - Battery input for any standard 3-volt lithium cell or other energy source. Battery voltage must be held between 2.0 and 3.5 volts for proper operation. The nominal write protect trip point voltage at which access to the real time clock and user RAM is denied is set by the internal circuitry as 1.25 x V_{BAT} nominal. A lithium battery with 48 mAhr or greater will back up the DS1307/DS1308 for more than 10 years in the absence of power at 25 degrees C.

SCL (Serial Clock Input) - SCL is used to synchronize data movement on the serial interface.

SDA (Serial Data Input/Output) - SDA is the input/output pin for the 2-wire serial interface. The SDA pin is open drain which requires an external pullup resistor.

SQW/OUT (Square Wave/ Output Driver) - When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square wave frequencies (1 Hz, 4 kHz, 8 kHz, 32 kHz). The SQW/OUT pin is open drain which requires an external pullup resistor.

NOTE: X1, X2 are not applicable for the DS1308 or DS1308N.

X1, X2 - Connections for a standard 32.768 kHz quartz crystal. The internal oscillator circuitry is designed for operation with a crystal having a specified load capacitance (CL) of 12.5 pF.

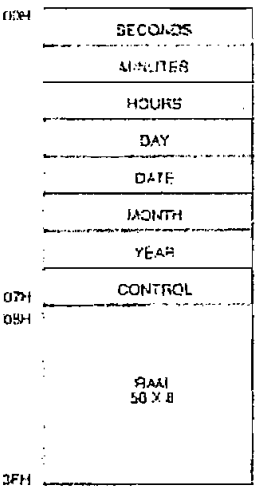
For more information on crystal selection and crystal layout considerations, please consult Application Note 58, "Crystal Considerations with Dallas Real Time Clocks." The DS1307 can also be driven by an external 32.768 kHz oscillator. In this configuration, the X1 pin is connected to the external oscillator signal and the X2 pin is floated.

Please review Application Note 95, "Interfacing the DS1307/DS1308 with a 8051-Compatible Microcontroller" for additional information.

RTC AND RAM ADDRESS MAP

The address map for the RTC and RAM registers of the DS1307/DS1308 is shown in Figure 2. The real time clock registers are located in address locations 00h to 07h. The RAM registers are located in address locations 08h to 3Fh. During a multi-byte access, when the address pointer reaches 3Fh, the end of RAM space, it wraps around to location 00h, the beginning of the clock space.

DS1307 ADDRESS MAP Figure 2



CLOCK AND CALENDAR

The time and calendar information is obtained by reading the appropriate register bytes. The real time clock registers are illustrated in Figure 3. The time and calendar are set or initialized by writing the appropriate register bytes. The contents of the time and calendar registers are in the Binary-Coded Decimal (BCD) format. Bit 7 of Register 0 is the Clock Halt (CH) bit. When this bit is set to a 1, the oscillator is disabled. When cleared to a 0, the oscillator is enabled.

Please note that the initial power on state of all registers is not defined. Therefore it is important to enable the oscillator (CH bit=0) during initial configuration.

The DS1307/DS1308 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10 hour bit (20-23 hours).

DS1307/DS1308 TIMEKEEPER REGISTERS Figure 3

	BIT 7						BIT 0	
00H	CH	10 SECONDS			SECONDS			00–59
	X	10 MINUTES			MINUTES			00–59
	X	12 / 24	10 HR AM/PM	10 HR	HOURS			01–12 00–23
	X	X	X	X	X	DAY		1–7
	X	X	10 DATE		DATE			01–28/29 01–30 01–31
	X	X	X	10 MONTH	MONTH			01–12
	10 YEAR				YEAR			00–99
07H	OUT	X	X	SQWE	X	X	RS1	RS0

CONTROL REGISTER

The DS1307/DS1308 Control Register is used to control the operation of the SQW/OUT pin.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	X	X	SQWE	X	X	RS1	RS0

OUT (Output control): This bit controls the output level of the SQW/OUT pin when the square wave output is disabled. If SQWE=0, the logic level on the SQW/OUT pin is 1 if OUT=1 and is 0 if OUT=0.

SQWE (Square Wave Enable): This bit, when set to a logic 1, will enable the oscillator output. The frequency of the square wave output depends upon the value of the RS0 and RS1 bits.

RS (Rate Select): These bits control the frequency of the square wave output when the square wave output has been enabled. Table 1 lists the square wave frequencies that can be selected with the RS bits.

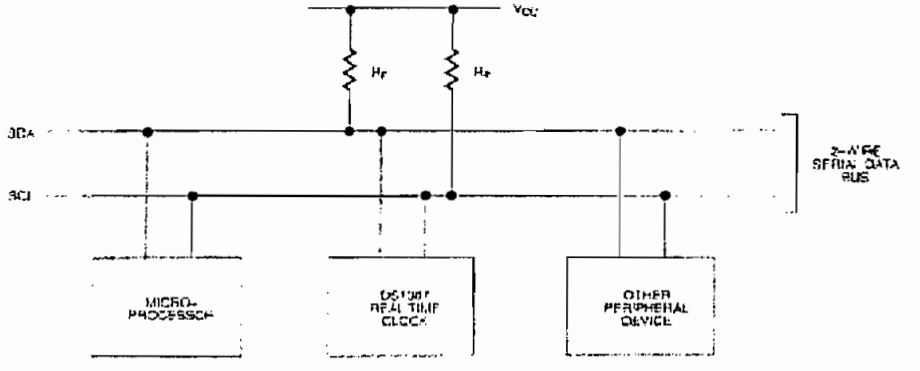
SQUAREWAVE OUTPUT FREQUENCY Table 1

RS1	RS0	SQW OUTPUT FREQUENCY
0	0	1 Hz
0	1	4.096 kHz
1	0	8.192 kHz
1	1	32.768 kHz

2-WIRE SERIAL DATA BUS

The DS1307 supports a bi-directional 2-wire bus and data transmission protocol. A device that sends data onto the bus is defined as a transmitter and a device receiving data as a receiver. The device that controls the message is called a master. The devices that are controlled by the master are referred to as slaves. The bus must be controlled by a master device which generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions. The DS1307/DS1308 operates as a slave on the 2-wire bus. A typical bus configuration using this 2-wire protocol is show in Figure 4.

TYPICAL 2-WIRE BUS CONFIGURATION Figure 4



Figures 5, 6, and 7 detail how data is transferred on the 2-wire bus.

- Data transfer may be initiated only when the bus is not busy.
- During data transfer, the data line must remain stable whenever the clock line is HIGH. Changes in the data line while the clock line is high will be interpreted as control signals.

Accordingly, the following bus conditions have been defined:

Bus not busy: Both data and clock lines remain HIGH.

Start data transfer: A change in the state of the data line, from HIGH to LOW, while the clock is HIGH, defines a START condition.

Stop data transfer: A change in the state of the data line, from LOW to HIGH, while the clock line is HIGH, defines the STOP condition.

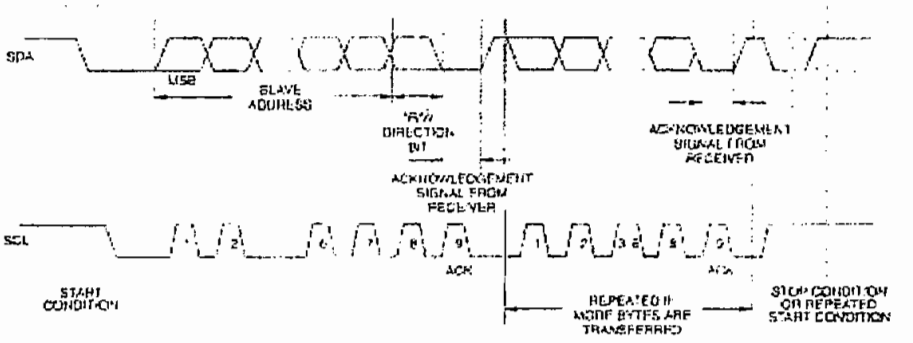
Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the HIGH period of the clock signal. The data on the line must be changed during the LOW period of the clock signal. There is one clock pulse per bit of data.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between START and STOP conditions is not limited, and is determined by the master device. The information is transferred byte-wise and each receiver acknowledges with a ninth bit. Within the 2-wire bus specifications a regular mode (100 kHz clock rate) and a fast mode (400 kHz clock rate) are defined. The DS1307/DS1308 operates in the regular mode (100 kHz) only.

Acknowledge: Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse which is associated with this acknowledge bit.

A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse. Of course, setup and hold times must be taken into account. A master must signal an end of data to the slave by not generating an acknowledge bit on the last byte that has been clocked out of the slave. In this case, the slave must leave the data line HIGH to enable the master to generate the STOP condition.

DATA TRANSFER ON 2-WIRE SERIAL BUS Figure 5



Depending upon the state of the R/$\overline{W}$ bit, two types of data transfer are possible:

1. **Data transfer from a master transmitter to a slave receiver.** The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte. Data is transferred with the most significant bit (MSB) first.

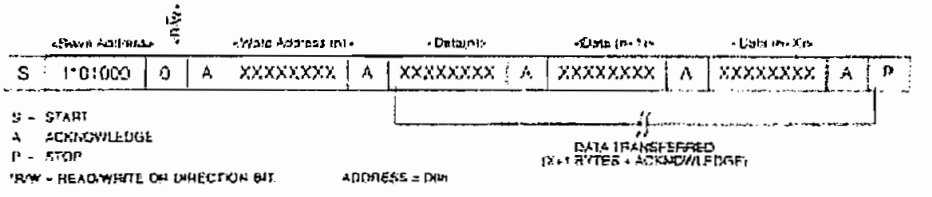
2. **Data transfer from a slave transmitter to a master receiver.** The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. This is followed by the slave transmitting a number of data bytes. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a 'not acknowledge' is returned.

The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the bus will not be released. Data is transferred with the most significant bit (MSB) first.

The DS1307/DS1308 may operate in the following two modes:

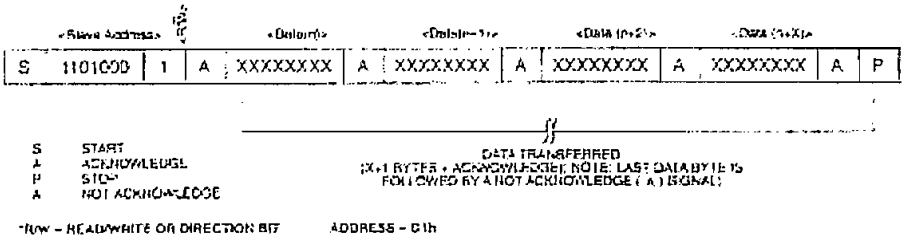
1. **Slave receiver mode (DS1307/DS1308 write mode):** Serial data and clock are received through SDA and SCL. After each byte is received an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and *direction bit (See Figure 6). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7 bit DS1307/DS1308 address, which is 1101000, followed by the *direction bit (R/$\overline{W}$) which, for a write, is a 0. After receiving and decoding the address byte the device outputs an acknowledge on the SDA line. After the DS1307/DS1308 acknowledges the slave address + write bit, the master transmits a register address to the DS1307/DS1308 This will set the register pointer on the DS1307/DS1308. The master will then begin transmitting each byte of data with the DS1307/DS1308 acknowledging each byte received. The master will generate a stop condition to terminate the data write.

DATA WRITE - SLAVE RECEIVER MODE Figure 6



2. **Slave transmitter mode (DS1307/DS1308 read mode):** The first byte is received and handled as in the slave receiver mode. However, in this mode, the *direction bit will indicate that the transfer direction is reversed. Serial data is transmitted on SDA by the DS1307/DS1308 while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (See Figure 7). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7-bit DS1307/DS1308 address, which is 1101000, followed by the *direction bit ($R/\overline{W}$) which, for a read, is a 1. After receiving and decoding the address byte the device inputs an acknowledge on the SDA line. The DS1307/DS1308 then begins to transmit data starting with the register address pointed to by the register pointer. If the register pointer is not written to before the initiation of a read mode the first address that is read is the last one stored in the register pointer. The DS1307/DS1308 must receive a Not Acknowledge to end a read.

DATA READ - SLAVE TRANSMITTER MODE Figure 7



ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground -0.5V to +7.0V
Operating Temperature 0°C to 70°C
Storage Temperature -55°C to +125°C
Soldering Temperature 260°C for 10 seconds (See NOTE 12)

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

The Dallas Semiconductor DS1307/DS1308 is built to the highest quality standards and manufactured for long term reliability. All Dallas Semiconductor devices are made using the same quality materials and manufacturing methods. However, standard versions of the DS1307/DS1308 are not exposed to environmental stresses, such as burn-in, that some industrial applications require. Products which have successfully passed through this series of environmental stresses are marked IND or N, denoting their extended operating temperature and reliability rating. For specific reliability information on this product, please contact the factory at (972) 371-4448.

RECOMMENDED DC OPERATING CONDITIONS (0°C to 70°C)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{CC}	4.5	5.0	5.5	V	1
Logic 1	V_{IH}	2.2		V_{CC}+0.3	V	1
Logic 0	V_{IL}	-0.3		+0.8	V	1
V_{BAT} Battery Voltage	V_{BAT}	2.0		3.5	V	1

DC ELECTRICAL CHARACTERISTICS (0°C to 70°C; V_{CC}=4.5V to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Input Leakage	I_{LI}			1	µA	10
I/O Leakage	I_{LO}			1	µA	11
Logic 0 Output	V_{OL}			0.4	V	2
Active Supply Current	I_{CCA}			1.5	mA	9
Standby Current	I_{CCS}			200	µA	3
Battery Current (OSC ON); SQW/OUT OFF	I_{BAT1}		300	500	nA	4
Battery Current (OSC ON); SQW/OUT ON (32 kHz)	I_{BAT2}		480	800	nA	4

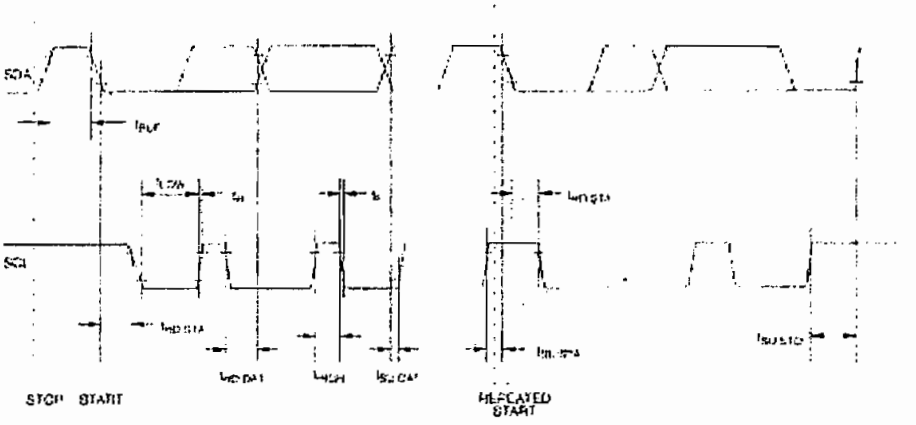
AC ELECTRICAL CHARACTERISTICS (0°C to 70°C; V_{CC}=4.5V to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
SCL Clock Frequency	f_{SCL}	0		100	kHz	
Bus Free Time Between a STOP and START Condition	t_{BUF}	4.7			µs	
Hold Time (Repeated) START Condition	$t_{HD:STA}$	4.0			µs	5
LOW Period of SCL Clock	t_{LOW}	4.7			µs	
HIGH Period of SCL Clock	t_{HIGH}	4.0			µs	
Set-up Time for a Repeated START Condition	$t_{SU:STA}$	4.7			µs	
Data Hold Time	$t_{HD:DAT}$	0			µs	6, 7
Data Set-up Time	$t_{SU:DAT}$	250			ns	
Rise Time of Both SDA and SCL Signals	t_R			1000	ns	
Fall Time of Both SDA and SCL Signals	t_F			300	ns	
Set-up Time for STOP Condition	$t_{SU:STO}$	4.7			µs	
Capacitive Load for each Bus Line	C_B			400	pF	8
I/O Capacitance	$C_{I/O}$		10		pF	
Crystal Specified Load Capacitance			12.5		pF	

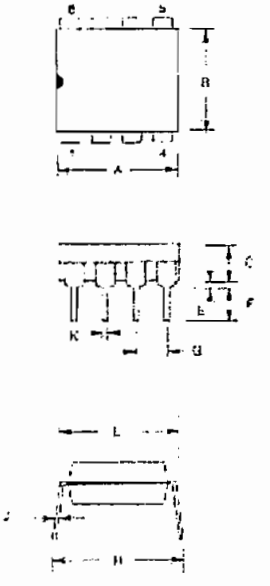
NOTES:

1. All voltages are referenced to ground.
2. Logic zero voltages are specified at a sink current of 5 mA at V_{CC}=4.5V, V_{OL}=GND for capacitive loads.
3. I_{CCS} specified with V_{CC}=5.0V and SDA, SCL=5.0V.
4. V_{CC}=0V, V_{BAT}=3V.
5. After this period, the first clock pulse is generated.
6. A device must internally provide a hold time of at least 300 ns for the SDA signal (referred to the V_{IHMIN} of the SCL signal) in order to bridge the undefined region of the falling edge of SCL.
7. The maximum $t_{HD:DAT}$ has only to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.
8. C_B - total capacitance of one bus line in pF.
9. I_{CCA} - SCL clocking at max frequency = 100 kHz.
10. SCL only.
11. SDA and SQW/OUT
12. The DS1308 is designed to be subjected to no more than two passes through a solder reflow process to limit premature crystal aging effects and maintain a reasonable accuracy of ±2 minutes/month at 25 degrees C (worst case).

TIMING DIAGRAM Figure 8

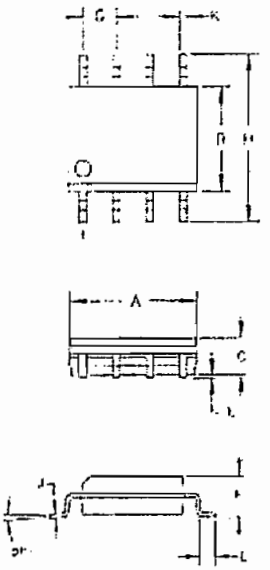


DS1307 64 X 8 SERIAL REAL TIME CLOCK
8-PIN DIP MECHANICAL DIMENSIONS



PKG	8-PIN	
DIM	MIN	MAX
A IN.	0.360	0.400
MM	9.14	10.16
B IN.	0.240	0.260
MM	6.10	6.60
C IN.	0.120	0.140
MM	3.05	3.56
D IN.	0.300	0.325
MM	7.62	8.26
E IN.	0.015	0.040
MM	0.38	1.02
F IN.	0.120	0.140
MM	3.04	3.56
G IN.	0.090	0.110
MM	2.29	2.79
H IN.	0.320	0.370
MM	8.13	9.40
J IN.	0.008	0.012
MM	0.20	0.30
K IN.	0.015	0.021
MM	0.38	0.53

DS1307Z 64 X 8 SERIAL REAL TIME CLOCK
8-PIN SOIC (150-MIL) MECHANICAL DIMENSIONS

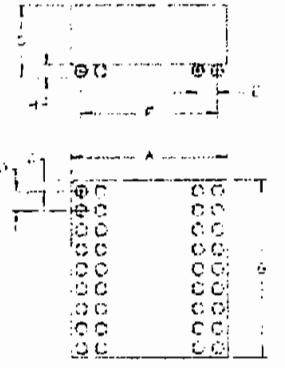


PKG	8-PIN (150 MIL)	
DIM	MIN	MAX
A IN.	0.188	0.196
MM	4.78	4.98
B IN.	0.150	0.158
MM	3.81	4.01
C IN.	0.048	0.062
MM	1.22	1.57
E IN.	0.004	0.010
MM	0.10	0.25
F IN.	0.053	0.069
MM	1.35	1.75
G IN.	0.050 BSC	
MM	1.27 BSC	
H IN.	0.230	0.244
MM	5.84	6.20
J IN.	0.007	0.011
MM	0.18	0.28
K IN.	0.012	0.020
MM	0.30	0.51
L IN.	0.016	0.050
MM	0.41	1.27
phi	0°	8°

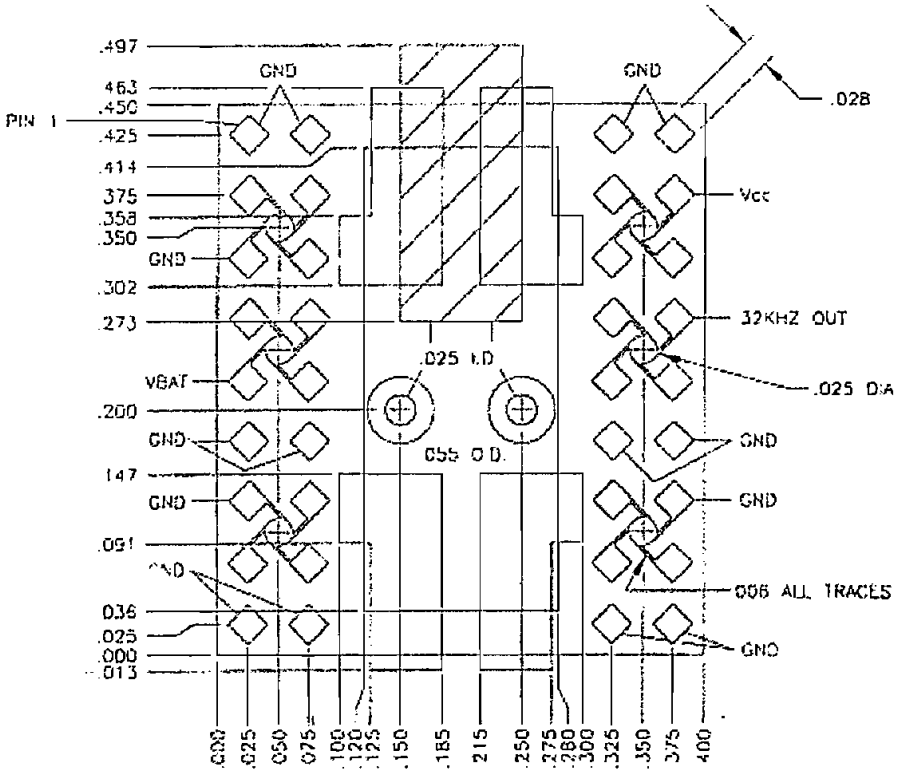
56-G2008-001

DS1308 MECHANICAL DIMENSIONS

PKG	36-PIN BALL GRID	
DIM	MIN	MAX
A IN. MM	0.395	0.405
B IN. MM	0.445	0.455
C IN. MM	0.022	0.028
D IN. MM	0.047	0.053
E IN. MM	0.047	0.053
F IN. MM	0.347	0.353
G IN. MM	0.170	0.190
H IN. MM	0.025	0.030



DS1308 RECOMMENDED LAYOUT LAND PATTERN



INTRODUCTION

Seiko Instruments intelligent dot matrix liquid crystal display modules have on-board controller and LSI drivers, which display alpha numerics, Japanese KATA KANA characters and a wide variety of other symbols in either 5 x 7 dot matrix.

The Internal operation in the KS0066 controller chip is determined by signals sent from the MPU. The signals include: 1) Register select RS Input consisting of Instruction register (IR) when RS = 0 and data register (DR) when RS = 1; 2) Read/write (R/W); 3) Data bus (DB7~ DB0); and 4) Enable strobe (E) depending on the MPU or through an external parallel I/O port. Details on instructions data entry, execution times, etc. are explained in the following sections.

READ AND WRITE TIMING DIAGRAMS AND TABLES

The following timing characteristics are applicable for all of Seiko's LCD dot matrix character modules.

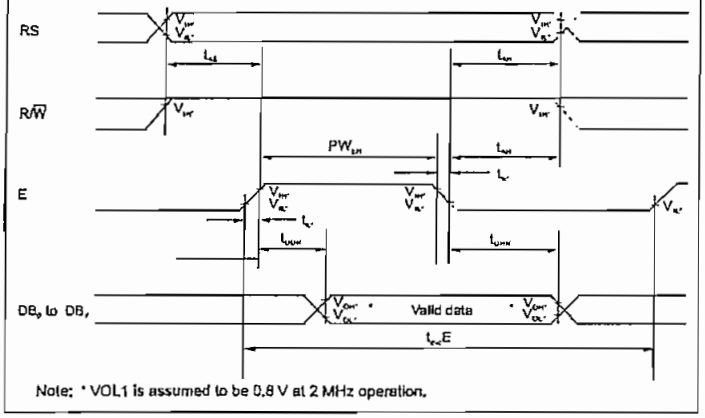
READ TIMING CHARACTERISTICS
V =5.0V±5%, V =0V, T =0°C TO 50°C

Item	Symbol	Standard		Unit
		Min.	Max.	
Enable cycle time	$t_{CYC}E$	500	—	ns
Enable pulse width High Level	PW_{EH}	230	—	ns
Enable rise and fall time	t_{ER}, t_{EF}	—	20	ns
Address setup time RS,R/W—E	t_{AS}	40	—	ns
Address hold time	t_{AH}	10	—	ns
Data delay time	t_{DDR}	—	160	ns
Data hold time	t_H	5	—	ns

WRITE TIMING CHARACTERISTICS
V =5.0V±5%, V =0V, T =0°C TO 50°C

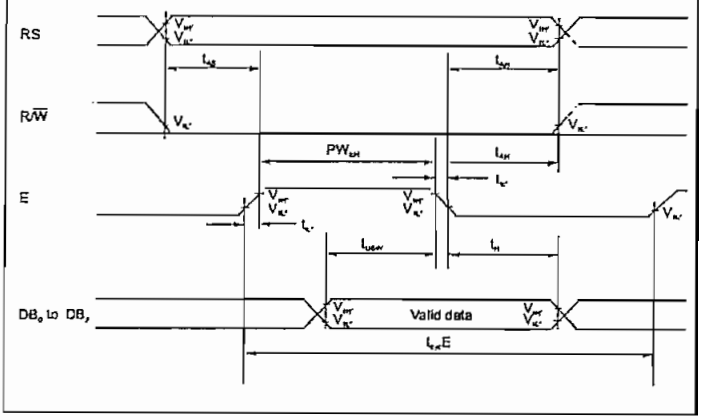
Item	Symbol	Standard		Unit
		Min.	Max.	
Enable cycle time	$t_{CYC}E$	500	—	ns
Enable pulse width High Level	PW_{EH}	230	—	ns
Enable rise and fall time	t_{ER}, t_{EF}	—	20	ns
Address setup time RS,R/W—E	t_{AS}	40	—	ns
Address hold time	t_{AH}	10	—	ns
Data setup time	t_{DSW}	80	—	ns
Data hold time	t_H	10	—	ns

READ OPERATION



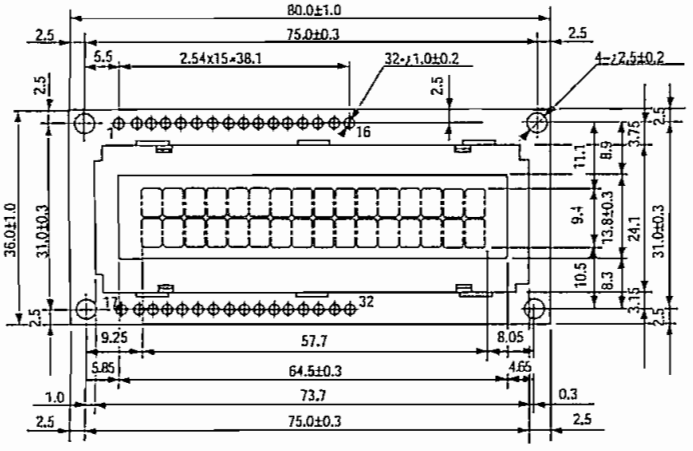
DATA READ FROM MODULE TO MPU

WRITE OPERATION

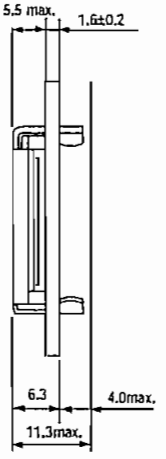


DATA WRITE FROM MPU TO MODULE

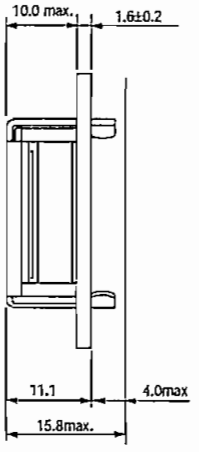
L1682 (2x16) Unit: mm General Tolerance ±0.5 mm

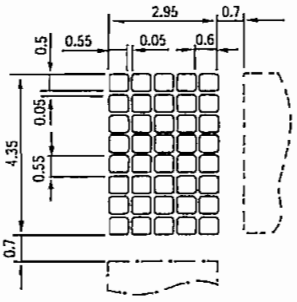


Reflective/EL Backlight



LED Backlight





*LED Powered through pins 15 & 16 or 31 & 32

Pin Functions

No.	No.	Name	Function
1	17	V_{ss}	GND
2	18	V_{cc}	Power supply voltage + 5 V
3	19	V_{lc}	Liquid crystal driving voltage
4	20	RS	L: Instruction code input. H: Data input
5	21	R/W	L: Data write from MPU to LCM. H: Data read from LCM to MPU
6	22	E	Enable
7	23	DB0	Data bus line
8	24	DB1	Data bus line
9	25	DB2	Data bus line
10	26	DB3	Data bus line
11	27	DB4	Data bus line
12	28	DB5	Data bus line
13	29	DB6	Data bus line
14	30	DB7	Data bus line
15	31	V_L*	Anode
16	32	V_c*	Cathode

OPERATING INSTRUCTIONS

INTRODUCTION CODES

Instruction	Set RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution Time (when f_{cp} or f_{osc} is 250 kHz)
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears all display memory and returns the cursor to the home position (Address 0).	82 μs ~ 1.64ms
Return Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (Address 0) shifted to the original position. DD RAM contents remain unchanged.	40 μs ~ 1.6ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies to or not to shift the display. These operations write and read.	40 μs ~ 1.64ms
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	(D) is display ON/OFF control; memory remains unchanged in OFF condition. (C) cursor ON/OFF (B) blinking cursor.	40 μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DD RAM contents.	40 μs
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display lines (N), and character font (F).	40 μs
Set CG RAM Address	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data is sent and received after this setting.	40 μs
Set DD RAM Address	0	0	1	A_{DD}							Sets the DD RAM address. DD RAM data is sent and received after this setting.	40 μs
Read Busy Flag & Address	0	1	BF	AC							Reads Busy Flag (BF) indicating internal operation is being performed and reads address counter contents.	1 μs
Write Data to CG or to DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40 μs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40 μs

* Doesn't matter

DD RAM: Display data RAM
CG RAM: Character generator RAM
A_{CG}: CG RAM address
A_{DD}: DD RAM address corresponds to cursor address
A_C: Address counter used for both DD RAM and CG RAM address

I/D = 1: Increment
I/D = 0: Decrement

S = 1: Display shift
S = 0: No display shift

D = 1: Display ON
D = 0: Display OFF

C = 1: Cursor ON
C = 0: Cursor OFF

B = 1: Blink ON
B = 0: Blink OFF

S/C = 1: Display shift
S/C = 0: Cursor movement

BF = 1: Internal operation in progress
BF = 0: Instruction can be accepted

R/L = 1: Right shift
R/L = 0: Left shift

DL = 1: 8 bits
DL = 0: 4 bits

N = 1: 2 lines (L1671)

F = 0: 5 x 7 dot matrix

Execution times in the above table indicate the minimum values when operating frequency is 250 kHz.

When f_{osc} is 270 kHz: 40μs x 250/250 = 37μs

OPERATING INSTRUCTIONS

INTRODUCTION CODE EXPLANATIONS

The two registers 1) Instruction Register (IR) and the 2) Data Register (DR) In the KS0066 controller chip are directly controlled by the MPU. Control information is temporarily stored In these registers prior to Internal operation start. This allows interface to various types of MPUs which operate at different speeds from that of the KS0066, and allows interface from peripheral control ICs. Internal operations of the KS0066 are determined from the signals sent from the MPU. These signals, including register selection signals (RS), Read/Write (R/W) and data bus signals (DB0 - DB7) are polled instructions.

REGISTER SELECTION

RS	R/W	Operation
0	0	IR selection, IR write. Internal operation: Display clear
0	1	Busy flag (DB7) and address counter (DB0 to DB6) read
1	0	DR selection, DR write. Internal operation: DR to DD RAM or CG RAM
1	1	DR selection, DR read. Internal operation: DD RAM or CG RAM to DR

ADDRESS COUNTER (AC)

The counter specifies an address when data is written into DD RAM or CG RAM and the data stored in DD RAM or CG RAM is read out. If an Address Set Instruction (for DD RAM or CG RAM) is written in the IR, the address information is transferred from the IR to the AC. When display data is written into or read from DD RAM or CG RAM, the AC is automatically incremented or decremented by one according to the Entry Mode Set. The contents of the AC are output to DB0 to DB6; refer to above "Register Selection Table" when RS = 0 and R/W = 1.

CLEAR DISPLAY

	RS	R/W	DB7							DB0
Code	0	0	0	0	0	0	0	0	0	1

Clear all display memory and return the cursor to the home position. In other words, the cursor returns to the first character block on the first line on all 1, 2, and 4 line character modules except L4044. I the above Is entered on E2 (the second controller for lines 3 and 4), the cursor will return to the first character on the third line.

CURSOR HOME

	RS	R/W	DB7							DB0
Code	0	0	0	0	0	0	0	0	1	*

*Doesn't matter

Returns cursor to home position. First line first character blocks on all 1, 2 and 4 line display; except L4044 refer "clear display": (Address 0; A_{DD} "80"). The contents of the DD RAM remain unchanged.

RESTRICTIONS ON EXECUTION OF DISPLAY CLEAR AND CURSOR HOME INSTRUCTIONS

Conditions of use	Restrictions
When executing the Display Clear or Cursor Home instruction when the display is shifted (after execution of Display Shift instruction).	The Cursor Home instruction should be executed again immediately after the Display Clear or Cursor Home instruction is executed. Do not leave an interval of a multiple of $400/f_{osc}$* second after the first execution. • L4052: f_{osc} = 250 kHz • The other modules: f_{osc} = 270 kHz *f_{osc}: Oscillation frequency
When 23_H, 27_H, 63_H, or 67_H is used as a DD RAM address to execute Cursor Home instruction.	Before executing the Cursor Home instruction, the data of the four DD RAM addresses given at the left should be read and saved. After execution, write the data again in DD RAM. (This restriction is necessary to prevent the contents of the DD RAM addresses from being destroyed after the Cursor Home instruction has been executed.)

OPERATING INSTRUCTIONS

ENTRY MODE SET

	RS	R/W	DB7						DB0
Code	0	0	0	0	0	0	0	1	I/D	S

I/D: Increments (I/D = 1) or decrements (I/D = 0) the DD RAM address by one block when writing or reading a character code from DD RAM or CG RAM. The cursor automatically moves to the right when incremented by one or to the left if decremented by one.

S: Shifts the entire display to either the right or left when S = 1 (high). When S = 1 and I/D = 1 the display shifts one position to the left. When S = 1 and 1/D = 0 the display shifts one position to the right. This right or left shift occurs after each data write to DD RAM. Display is not shifted when reading from DD RAM. Display is not shifted when S = 0.

DISPLAY AND CURSOR ON/OFF CONTROL

	RS	R/W	DB7						DB0
Code	0	0	0	0	0	0	1	D	C	B

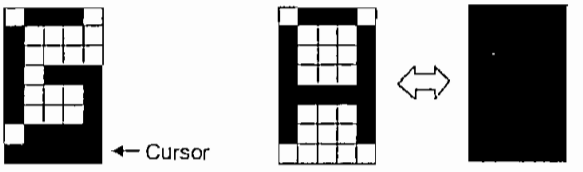
D: Display is turned ON when D = 1 and OFF when D = 0. When display is OFF, display data in DD RAM remains unchanged. Information comes back immediately when D = 1 is entered.

C: Cursor is displayed when C = 1 and not displayed when C = 0. If the cursor disappears, function of I/D etc. does not change during display data write. In a 5 x 7 dot matrix there is an eighth line which functions as the cursor.

B: When B = 1, the character at the cursor position starts blinking. When B = 0 the cursor does not blink. The blink is done by stiching between the all black dot matrix and displayed character at 0.4 seconds intervals. The cursor and the blink can be set at the same time (fosc = 250 kHz).

5 X 7 DOT MATRIX

C = 1 (cursor display) B = 1 (blinking)



CURSOR OR DISPLAY SHIFT

	RS	R/W	DB7						DB0
Code	0	0	0	0	0	1	S/C	R/L	*	*

* Doesn't Matter

Cursor/Display Shift moves the cursor or shifts the display without changing the DD RAM contents.

The cursor position and the AC contents match. This instruction is available for display correction and retrieval because the cursor position or display can be shifted without writing or reading display data. In case of a 2-line display, the cursor is shifted from character block 40 of line 1 to character block 1 of line 2. Displays of lines 1 and 2 are shifted at the same time. In case of a 4-line display, the cursor does not move continuously from line 2 to line 3. The cursor is shifted from character block 40 of line 3 to character block 1 of line 4. Displays of lines 3 and 4 are shifted at the same time. The display pattern of line 2 or 4 is not shifted to line 1 or 3.

S/C	R/L	Operation
0	0	The cursor position is shifted to the left (the AC decrements one)
0	1	The cursor position is shifted to the right (the AC increments one)
1	0	The entire display is shifted to the left with the cursor
1	1	The entire display is shifted to the right with the cursor

OPERATING INSTRUCTIONS

FUNCTION SET

	RS	R/W	DB7							DB0
Code	0	0	0	0	1	DL	N	F	*	*

* Doesn't Matter

Function Set sets the interface data length, the number of display lines and the character font.

DL: Interface data length

When DL = 1, the data length is set at 8 bits (DB7 to DB0).

When DL = 0, the data length is set at 4 bits (DB7 to DB4).

The upper 4 bits are transferred first, then the lower 4 bits follow.

N: Number of display lines

F: Sets character font

N	F	Number of display lines	Character font	Duty Cycle	LCD Module
1	0	2	5 x 7 dot matrix	1/16	All character LCD modules

The Function Set instruction must be executed prior to all other instructions except for Busy Flag/Address Read. If another instruction is executed first, no function instruction except changing the interface data lenght can be executied.

CG RAM ADDRESS SET

	RS	R/W	DB7							DB0
Code	0	0	0	1	A	A	A	A	A	A
				← Upper bit						Lower bit →

CG RAM addresses, expressed as binary AAAAAA, are set to the AC. Then data in CG RAM is written from or read to the MPU.

DD RAM ADDRESS SET

	RS	R/W	DB7							DB0
Code	0	0	1	A	A	A	A	A	A	A
				← Upper bit						Lower bit →

DD RAM addresses expressed as binary AAAAAA are set to the AC. Then data in DD RAM is written from or read to the MPU.

BUSY FLAG/ADDRESS READ

	RS	R/W	DB7							DB0
Code	0	0	BF	A	A	A	A	A	A	A
				← Upper bit						Lower bit →

The BF signal can be read to verify if the controller is indicating that the module is working on a current instruction.

When BF = 1, the module is working internally and the next instruction cannot be accepted until the BF value becomes 0.

When BF = 0, the next instruction can be accepted.

Therefore, make sure that BF = 0 before writing the next instruction. The AC values of binary AAAAAA are read out at the same time as reading the busy flag. The AC addresses are used for both CG RAM and DD RAM but the address set before execution of the instruction determines which address is to be used.

DATA WRITE TO CG RAM OR DD RAM

	RS	R/W	DB7							DB0
Code	1	0	D	D	D	D	D	D	D	D
				← Upper bit						Lower bit →

Binary eight-bit data DDDDDDDD is written into CG RAM or DD RAM. The CG RAM Address Set instruction or the DD RAM Address Set instruction before this instruction selects either RAM. After the write operation, the address and display shift are determined by the entry mode setting.

DATA READ TO CG RAM OR DD RAM

	RS	R/W	DB7							DB0
Code	1	1	D	D	D	D	D	D	D	D
				← Upper bit						Lower bit →

Binary eight-bit data DDDDDDDD is read from CG RAM or DD RAM. The CG RAM Address Set instruction or the DD RAM Address Set instruction before this instruction selects either RAM. In addition, either instruction is executed immediately before this instruction. If no Address Set instruction is executed before a read instruction, the first data read becomes invalid. If read instructions are executed consecutively, data is normally read from the second time. However, if the cursor is shifted by the Cursor Shift Instruction when reading DD RAM, there is no need to execute an address set instruction because the Cursor Shift Instruction does this.

After the read operation, the address is automatically incremented or decremented by one according to the entry mode, but the display is not shifted.

Note: The AC is automatically incremented or decremented by one according to the entry mode after a write instruction is excecuted to write data in CG RAM or DD RAM. However, the data of the RAM selected by the AC are not read out even if a read instruction is executed immediately afterwards.

5 x 7 + CURSOR

Relationships between CG RAM addresses and character codes (DD RAM) and character patterns (CG RAM data), (5 x 7 dot matrix).5 X 7 Table

Character code (DD RAM data)								CG RAM address						Character pattern (CG RAM data)								
7	6	5	4	3	2	1	0	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
Upper bit					Lower bit			Upper bit			Lower bit			Upper bit					Lower bit			
0	0	0	0	*	0	0	0	0	0	0	0	0	0	*	*	*	1	1	1	1	0	
											0	0	1				1	0	0	0	1	Example of character pattern (R)
											0	1	0				1	0	0	0	1	
											0	1	1				1	1	1	1	0	
											1	0	0				1	0	1	0	0	
											1	0	1				1	0	0	1	0	
											1	1	0				1	0	0	0	1	
											1	1	1	*	*	*	0	0	0	0	0	Cursor position
0	0	0	0	*	0	0	1	0	0	1	0	0	0	*	*	*	1	0	0	0	1	
											0	0	1				0	1	0	1	0	Example of character pattern (¥)
											0	1	0				1	1	1	1	1	
											0	1	1				0	0	1	0	0	
											1	0	0				1	1	1	1	1	
											1	0	1				0	0	1	0	0	
											1	1	0				0	0	1	0	0	
											1	1	1	*	*	*	0	0	0	0	0	
											0	0	0	*	*	*						
											0	0	1									
0	0	0	0	*	1	1	1	1	1	1												
											1	0	0									
											1	0	1									
											1	1	0									
											1	1	1	*	*	*						

NOTES:

- In CG RAM data, 1 corresponds to Selection and 0 to Non-selection on the display.
- Character code bits 0 to 2 and CG RAM address bits 3 to 5 correspond with each other (three bits, eight types).
- CG RAM address bits 0 to 2 specify a line posltion for a character pattern. Line 8 of a character pattern is the cursor position where the logical sum of the cursor and CG RAM data Is displayed. Set the data of line 8 to 0 to display the cursor. If the data is charged to 1, one bit lights, regardless of the cursor.
- The character pattern column position corresponds to CG RAM data bits 0 to 4 and bit 4 comes to the left end. CG RAM data bits 5 to 7 are not displayed but can be used as general data RAM.
- When reading a character pattern from CG RAM, set to 0 all of character code bits 4 to 7. Bits 0 to 2 determine which pattern will be read out. Since bit 3 is not valid, 00_H and 08_H select the same character.

PROGRAMMING THE CHARACTER GENERATOR RAM (CG RAM)

The character generator RAM (CG RAM) allows the user to create up to eight custom 5 x 7 characters + cursor (5 x 8). Once programmed, the custom characters or symbols are accessed exactly as if they were in ROM. However since the RAM is a volatile memory, power must be continually maintained. Otherwise, the custom characters/symbols must be programmed into non-volatile external ROM and sent to the display after each display initialization. All dots in the 5 x 8 dot matrix can be programmed, which includes the cursor position.

The modules RAM are divided into two parts: data display RAM (DD RAM) and custom character generator RAM (CG RAM). This is not to be confused programming the custom character generator RAM with the 192 character generator ROM. The CG RAM is located between hex 40 and 7F and is contiguous. Locations 40 thru 47 hold the first custom character (5 x 8), 48 thru 4F hold the second custom character, 50 thru 57 hold the third CG, and so forth to 78 thru 7F for the eighth CG character/symbol.

If during initialization the display was programmed to automatically increment, then only the single initial address, 40, need be sent. Consecutive row data will automatically appear at 41, 42, etc. until the completed character is formed. All eight custom CG characters can be programmed in 64 consecutive "writes" after sending the single initial 40 address.

The CG RAM is 8 bits wide, although only the right-most 5-bits are used for a custom CG character row. The left-most dot of programming the CG RAM character corresponds to D4 in the most significant nibble (XXXD4) of the data bus code, with the remaining 4 dots in the row corresponding to the least significant nibble (D3 thru D0), D0 being the right-most dot. Thus, hex 1F equals all dots on and hex 00 equals all dots off. Examples include hex 15 (10101) equal to 3 dots on the hex 0A (01010) equal 2 dots on. In each case the key 5-bits of the 8-bit code program one row of a custom CG character. When all 7 or 8 rows are programmed, the character is complete. A graphic example is shown below:

RS	R/W	Data	Display	Description
0	0	40	—	addresses 1st row, 1st CG character
1	0	11	* *	result of 11, 1st row
1	0	0A	* *	result of 0A, 2nd row
1	0	1F	*****	result of 1F, 3rd row
1	0	04	*	result of 04, 4th row
1	0	1F	*****	result of 1F, 5th row
1	0	04	*	result of 04, 6th row
1	0	04	*	result of 04, 7th row
1	0	00	—	result of 00, 8th row (cursor position)
1	0	15	***	1st row, 2nd CG character. **Note:** Addressing not now required; hex 48 is next in the sequence.

OPERATING INSTRUCTIONS

ADDRESS LOCATIONS

1) L1671-SERIES (16 characters x 1 line)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	80	81	82	83	84	85	86	87	C0	C1	C2	C3	C4	C5	C6	C7

NOTE: L1671 series is initialized as a 2 line display, because of the absence of an LCD driver. You must address character no. 9 as you would the first position on the 2nd line which is (C0).

2) L1672-SERIES (16 characters x 2 lines)
L1682-SERIES
L1692-SERIES

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

3) L1634-SERIES (16 characters x 4 lines)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
Line 3	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
Line 4	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF

4) L2032-SERIES (20 characters x 2 lines)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3

5) L2034-SERIES (20 characters x 4 lines)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3
Line 3	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
Line 4	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7

6) L2462-SERIES (24 characters x 2 lines)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7

7) L4052-SERIES (40 characters x 2 lines)
L4044-SERIES (40 characters x 4 lines)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7
Line 3	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
Line 4	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7

Note: Address locations on lines 1 & 2 are controlled by enabling E1. Address locations on lines 3 & 4 are controlled by enabling E2.

CHARACTER FONT CODES (5 x 7 DOT MATRIX)

Upper 4 Bit Hexadecimal

Lower 4 Bit Hexadecimal

Upper 4 bits / Lower 4 bits	0000 (0)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1010 (A)	1011 (B)	1100 (C)	1101 (D)	1110 (E)	1111 (F)
×××× 0000 (0)	CG RAM (1)												
×××× 0001 (1)	(2)												
×××× 0010 (2)	(3)												
×××× 0011 (3)	(4)												
×××× 0100 (4)	(5)												
×××× 0101 (5)	(6)												
×××× 0110 (6)	(7)												
×××× 0111 (7)	(8)												
×××× 1000 (8)	(1)												
×××× 1001 (9)	(2)												
×××× 1010 (A)	(3)												
×××× 1011 (B)	(4)												
×××× 1100 (C)	(5)												
×××× 1101 (D)	(6)												
×××× 1110 (E)	(7)												
×××× 1111 (F)	(8)												

OPERATING INSTRUCTIONS

EXAMPLES OF 8-BIT AND 4-BIT DATA TRANSFER OPERATION

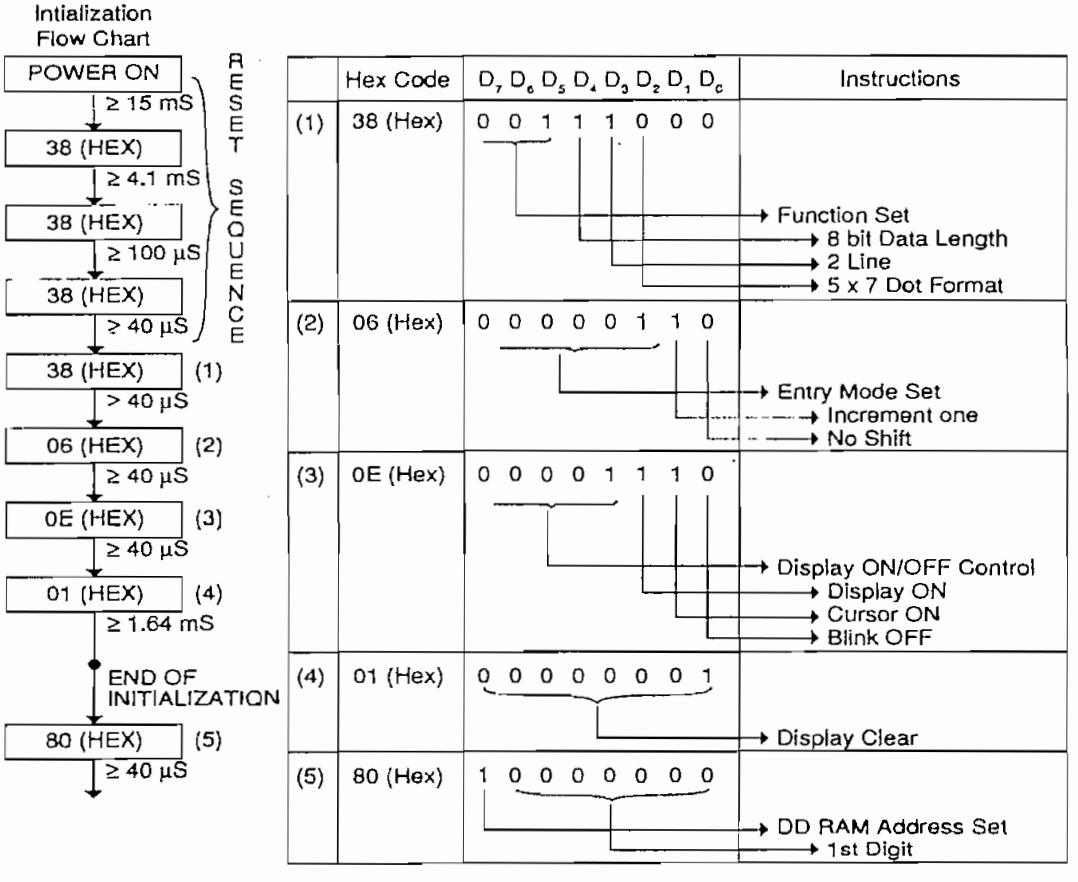
DISPLAY INITIALIZATION

Each time the module is turned on or reset, an initialization procedure must be executed. The procedure consists of sending a sequence of hex codes from the microprocessor or parallel I/O port. The initialization sequence turns on the cursor, clears the display, and sets the module onto an auto-increment mode.

The initial hex code 30, 34, or 38 is sent two or more times to ensure the module enters the 8-bit or 4-bit data mode. All the initialization sequences are performed under the condition of Register Select (RS) = 0 (low) and Read/Write (R/W) = 0 (low).

The 4-bit data bus microcontroller may operate the display module by sending the initialization sequence in 4-bit format. Since 4-bit operation requires the data to be sent twice over the higher 4-bit bus lines (D4-D7), memory requirements are doubled.

A. EXAMPLE FOR THE MODULE WITH 5 x 7 Character Format Under 8-Bit Data Transfer

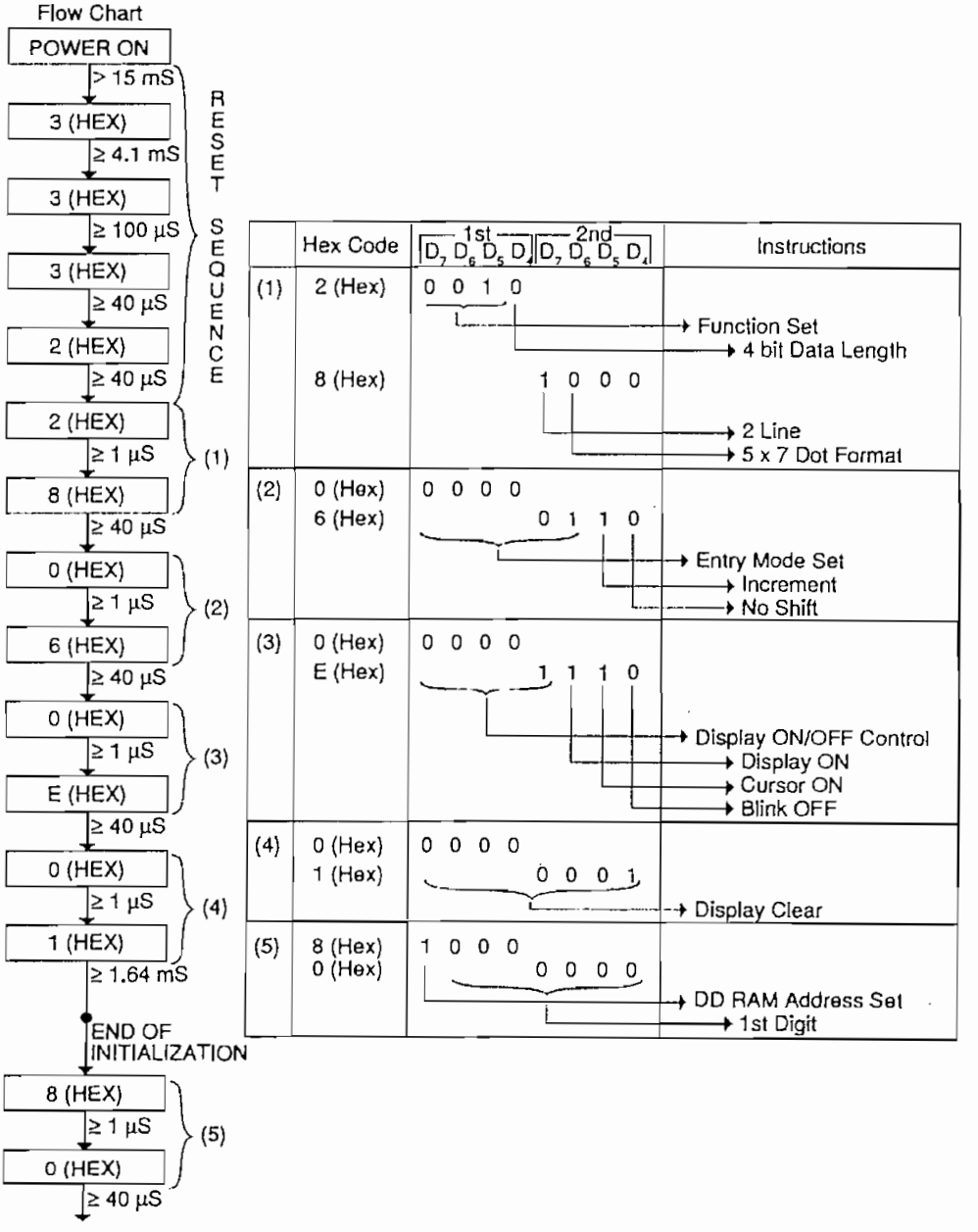


Note:

1) Both RS and R/W terminals shall be "0" in this sequence.
2) RS, R/W and Data are latched at the falling edge of the Enable signal, (falling edge is typically 10nSec; Max: 20nSec).
3) L4044 has t obe initialized on E1 and E2 respectively.

EXAMPLES OF 8-BIT AND 4-BIT DATA TRANSFER OPERATION

B. EXAMPLE FOR THE MODULE WITH 5 x 7 Character Format Under 4-Bit Data Transfer



Note:
1) Both RS and R/W terminals shall be "0" in this sequence.
2) RS, R/W and Data are latched at the falling edge of the Enable signal,
3) Enable signal has to be sent after every 4-bit Data transfer.

OPERATING INSTRUCTIONS

1) Display "THIS IS SEIKO LCD MODULE" on L1672.
   - Set RS = R/W = 0 (low), then send hex codes 38, 38, 06, 0E, 01, 80. (80 is the home position of the DD RAM)
   - Set RS = 1 and R/W = 0, then send hex codes 54, 48, 49 ,53, 20, 49, 53, 20, 53 ,45,49, 4B, 4F.
   - Set RS = R/W = 0, then send hex code C0 to start from beginning of the second line.
   - Set RS = 1 and R/W = 0, then send hex codes 4C, 43, 44, 20, 4D, 44, 55, 4C, 45.

2) Display "ONE", "TWO", "THREE", and "FOUR" on each line of L4044.
   - Set RS = R/W = E2 = 0, E1 = 1, then send hex codes 38, 38, 06, 0E, 01, 80.
   - Set RS = R/W = E1 = 0, E2 = 1, then send hex codes 38, 06, 0C, 01.
   - Set RS = E1 = 1, R/W = E2 = 0, then send hex codes 4F, 4E, 45.
   - Set RS = R/W = E2 = 0, E1 = 1, then send hex code C0.
   - Set RS = E1 = 1, R/W = E2 = 0, then send hex codes 54, 57, 4F.
   - Set RS = R/W = E2 = 0, E1 = 1, then send hex code 0C.
   - Set RS = R/W = E1 = 0, E2 = 1, then send hex codes 0E, 80.
   - Set RS = E2 = 1, R/W = E1 = 0, then send hex codes 45, 48, 52, 45, 45.
   - Set RS = R/W = E1 = 0, E2 = 1, then send hex code C0.
   - Set RS = E2 = 1, R/W = E1 = 0, then send hex code 46, 4F, 55, 52.

2) Display "L1671 LCD MODULE" on L1671. (Special case in the LCD modules)
   - Set RS = R/W = 0, then send hex codes 38, 38, 06, 0E, 01, 80.
   - Set RS = 1 and R/W = 0, then send hex codes 38, 06, 0C, 01.
   - Set RS = R/W = 0, then send hex code C0 to start from 9th character.
   - Set RS = 1 and R/W = 0, then send hex codes 44, 20, 4D, 4F, 44, 55, 4C, 45.

INTERFACE PROBLEMS AND POSSIBLE SOLUTIONS

Although the following problems and possible solutions are not all inclusive, they do represent the most common problems experienced not only by the first-time user, but also experienced users. If the user is experiencing problems, please review all of the following information. If the user still has problems, please call **Seiko Instruments** in Torrance, California at (310) 517-7771.

SYMPTOMS

1) Display is blank after power ON and initialization: Check 1- 6.
2) Wrong information being displayed: Check 3,4,6,7,8,9.
3) Symptoms same as 2, except multiple components are tied to the data bus: Check 8,9,10.
4) ICs become HOT: Check 1,11,12,13.
5) Cannot enter information to the 2nd line or lines 3 and 4 of the 4-line display: Check 4,14,15.

POSSIBLE SOLUTION(S)

1) Check +5 VDC and ground lines and connections.
2) A variable resistor or fixed resistor must be used on the V_{LC} pin for all LCD modules. V_{LC} voltage range is: 0~.7 volts (ref: Contrast Adjustment Circuit).
3) Data is being transmitted too fast:
   - Wait 4.5 mS after Power ON, or until V_{DD} reaches 4.5 volts. Wait more than 15 mS after V_{DD} reaches 4.5 volts.
   - Allow 1.6mS, after entering hexadecimal 01 or 02 at the end of the initialization sequence, then enter data.
   - Time interval between other data entries should be 50uS or greater.
4) Failure to properly initialize the display: Check initialization examples for either 4 or 8-bit. Make sure to enter first hexadecimal entry at least twice in the initializa tion sequence. This sets the LCD to either a 1 or 2 line display.
5) LCD input assumed to be configured as an IC. (This is not correct.)
6) Check the time interval on the falling edge of the enable pulse. Should not exceed 20nS (typical is 10nS).
7) Enable pulse width is shorter than 230nS.
8) More than one external bus being selected: Check data bus connection.
9) Signal levels are too low: Insure that V(IH & OH) is more than 2.4 volts.
10) All data bus components do not have TTL type outputs.
11) V_{DD} and V_{SS} pins are reversed.
12) Too much voltage on V_{DD} - (Max. 7VDC).
13) Load being put on data lines, when power is OFF on the V_{DD} pin.
14) Check address locations for the first position on the second line for each 2 lines (ref: Address Location Chart).
15) L4044 has two controllers: E1 for lines 1 & 2 and E2 for lines 3 & 4. Initialization must be done for E1 and the same for E2.

Terminal Voltages ±5V, 100 Taps

X9C102/103/104/503

Digitally Controlled Potentiometer (XDCP™)

FEATURES

- Solid-state potentiometer
- 3-wire serial interface
- 100 wiper tap points
  - —Wiper position stored in nonvolatile memory and recalled on power-up
- 99 resistive elements
  - —Temperature compensated
  - —End to end resistance, ±20%
  - —Terminal voltages, ±5V
- Low power CMOS
  - —V_{CC} = 5V
  - —Active current, 3mA max.
  - —Standby current, 500µA max.
- High reliability
  - —Endurance, 100,000 data changes per bit
  - —Register data retention, 100 years
- X9C102 = 1 kΩ
- X9C103 = 10 kΩ
- X9C503 = 50 kΩ
- X9C104 = 100 kΩ
- Packages
  - —8-lead SOIC and DIP

DESCRIPTION

The X9Cxxx are Xicor digitally controlled (XDCP) potentiometers. The device consists of a resistor array, wiper switches, a control section, and nonvolatile memory. The wiper position is controlled by a three-wire interface.

The potentiometer is implemented by a resistor array composed of 99 resistive elements and a wiper switching network. Between each element and at either end are tap points accessible to the wiper terminal. The position of the wiper element is controlled by the $\overline{CS}$, $U/\overline{D}$, and $\overline{INC}$ inputs. The position of the wiper can be stored in nonvolatile memory and then be recalled upon a subsequent power-up operation.

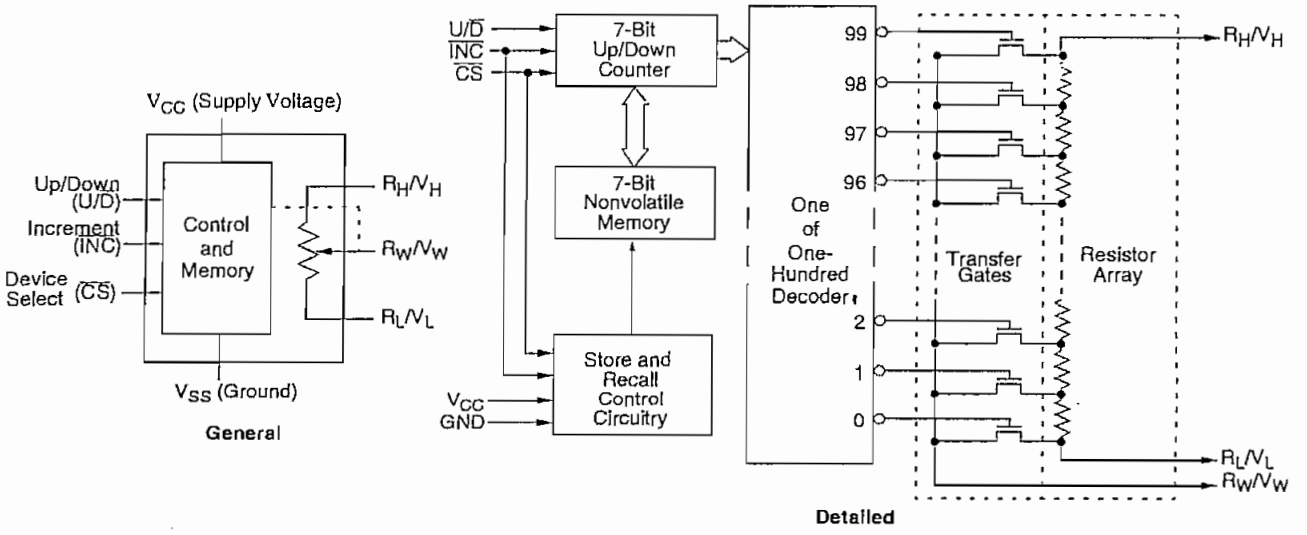
The device can be used as a three-terminal potentiometer or as a two-terminal variable resistor in a wide variety of applications including:

– control

– parameter adjustments

– signal processing

BLOCK DIAGRAM



XDCP is a trademark of Xicor, Inc.

PIN DESCRIPTIONS

R_H/V_H and R_L/V_L

The high (V_H/R_H) and low (V_L/R_L) terminals of the X9C102/103/104/503 are equivalent to the fixed terminals of a mechanical potentiometer. The minimum voltage is –5V and the maximum is +5V. The terminology of V_H/R_H and V_L/R_L references the relative position of the terminal in relation to wiper movement direction selected by the U/$\overline{D}$ input and not the voltage potential on the terminal.

R_W/V_W

V_W/R_W is the wiper terminal, and is equivalent to the movable terminal of a mechanical potentiometer. The position of the wiper within the array is determined by the control inputs. The wiper terminal series resistance is typically 40Ω.

Up/Down (U/$\overline{D}$)

The U/$\overline{D}$ input controls the direction of the wiper movement and whether the counter is incremented or decremented.

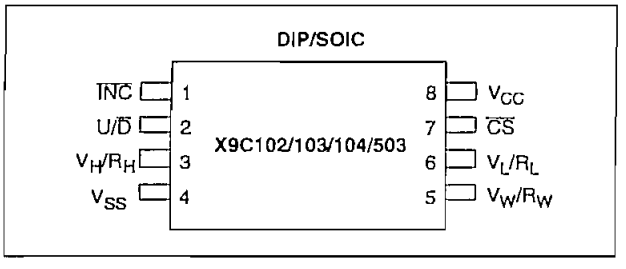
Increment ($\overline{INC}$)

The $\overline{INC}$ input is negative-edge triggered. Toggling $\overline{INC}$ will move the wiper and either increment or decrement the counter in the direction indicated by the logic level on the U/$\overline{D}$ input.

Chip Select ($\overline{CS}$)

The device is selected when the $\overline{CS}$ input is LOW. The current counter value is stored in nonvolatile memory when $\overline{CS}$ is returned HIGH while the $\overline{INC}$ input is also HIGH. After the store operation is complete the X9C102/103/104/503 device will be placed in the low power standby mode until the device is selected once again.

PIN CONFIGURATION



PIN NAMES

Symbol	Description
V_H/R_H	High Terminal
V_W/R_W	Wiper Terminal
V_L/R_L	Low Terminal
V_{SS}	Ground
V_{CC}	Supply Voltage
U/$\overline{D}$	Up/Down Control Input
$\overline{INC}$	Increment Control Input
$\overline{CS}$	Chip Select Control Input
NC	No Connection

PRINCIPLES OF OPERATION

There are three sections of the X9Cxxx: the input control, counter and decode section; the nonvolatile memory; and the resistor array. The input control section operates just like an up/down counter. The output of this counter is decoded to turn on a single electronic switch connecting a point on the resistor array to the wiper output. Under the proper conditions the contents of the counter can be stored in nonvolatile memory and retained for future use. The resistor array is comprised of 99 individual resistors connected in series. At either end of the array and between each resistor is an electronic switch that transfers the potential at that point to the wiper.

The wiper, when at either fixed terminal, acts like its mechanical equivalent and does not move beyond the last position. That is, the counter does not wrap around when clocked to either extreme.

The electronic switches on the device operate in a "make before break" mode when the wiper changes tap positions. If the wiper is moved several positions, multiple taps are connected to the wiper for t_{IW} ($\overline{INC}$ to V_W/R_W change). The R_{TOTAL} value for the device can temporarily be reduced by a significant amount if the wiper is moved several positions.

When the device is powered-down, the last wiper position stored will be maintained in the nonvolatile memory. When power is restored, the contents of the memory are recalled and the wiper is set to the value last stored.

INSTRUCTIONS AND PROGRAMMING

The $\overline{INC}$, U/$\overline{D}$ and $\overline{CS}$ inputs control the movement of the wiper along the resistor array. With $\overline{CS}$ set LOW the device is selected and enabled to respond to the U/$\overline{D}$ and $\overline{INC}$ inputs. HIGH to LOW transitions on $\overline{INC}$ will increment or decrement (depending on the state of the U/$\overline{D}$ input) a seven-bit counter. The output of this counter is decoded to select one of one-hundred wiper positions along the resistive array.

The value of the counter is stored in nonvolatile memory whenever $\overline{CS}$ transitions HIGH while the $\overline{INC}$ input is also HIGH.

The system may select the X9Cxxx, move the wiper, and deselect the device without having to store the latest wiper position in nonvolatile memory. After the wiper movement is performed as described above and once the new position is reached, the system must keep $\overline{INC}$ LOW while taking $\overline{CS}$ HIGH. The new wiper position will be maintained until changed by the system or until a power-down/up cycle recalled the previously stored data.

This procedure allows the system to always power-up to a preset value stored in nonvolatile memory; then during system operation minor adjustments could be made. The adjustments might be based on user preference: system parameter changes due to temperature drift, etc...

The state of U/$\overline{D}$ may be changed while $\overline{CS}$ remains LOW. This allows the host system to enable the device and then move the wiper up and down until the proper *trim* is attained.

MODE SELECTION

$\overline{CS}$	$\overline{INC}$	U/$\overline{D}$	Mode
L	↓	H	Wiper Up
L	↓	L	Wiper Down
↑	H	X	Store Wiper Position
H	X	X	Standby Current
↑	L	X	No Store, Return to Standby

SYMBOL TABLE

WAVEFORM	INPUTS	OUTPUTS
———	Must be steady	Will be steady
////	May change from Low to High	Will change from Low to High
\\\\	May change from High to Low	Will change from High to Low
XXXX	Don't Care: Changes Allowed	Changing: State Not Known
⋈	N/A	Center Line is High Impedance

ABSOLUTE MAXIMUM RATINGS

Temperature under bias .....–65°C to +135°C
Storage temperature .....–65°C to +150°C
Voltage on $\overline{CS}$, $\overline{INC}$, U/$\overline{D}$ and V_{CC}
with respect to V_{SS}.....–1V to +7V
Voltage on V_H/R_H and V_L/R_L
referenced to V_{SS} .....–8V to +8V
$\Delta V = |V_H/R_H - V_L/R_L|$
X9C102 ..... 4V
X9C103, X9C503, and X9C104..... 10V
Lead temperature (soldering, 10 seconds)..... +300°C
I_W (10 seconds) 8.8mA

COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only; functional operation of the device (at these or any other conditions above those listed in the operational sections of this specification) is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED OPERATING CONDITIONS

Temperature	Min.	Max.
Commercial	0°C	+70°C
Industrial	–40°C	+85°C
Military	–55°C	+125°C

Supply Voltage (V_{CC})	Limits
X9C102/103/104/503	5V ±10%

POTENTIOMETER CHARACTERISTICS (Over recommended operating conditions unless otherwise stated.)

Symbol	Parameter	Limits			Unit	Test Conditions/Notes
		Min.	Typ.	Max.		
R_{TOTAL}	End to end resistance variation	–20		+20	%	
$V_{VH/RH}$	V_H terminal voltage	–5		+5	V	
$V_{VL/RL}$	V_L terminal voltage	–5		+5	V	
	Power rating			16	mW	X9C102
	Power rating			10	mW	X9C103/104/503
I_W	Wiper current	-4.4		4.4	mA	
R_W	Wiper resistance		40	100	Ω	Wiper Current = ±1mA
	Noise		–120		dBV	Ref. 1kHz
	Resolution		1		%	
	Absolute linearity(1)	–1		+1	M(3)	$V_{W(n)(actual)} - V_{W(n)(expected)}$
	Relative linearity(2)	–0.2		+0.2	MI(3)	$V_{W(n+1)(actual)} - [V_{W(n)} + MI]$
	RTOTAL temperature coefficient		±300		ppm/°C	X9C103/503/104
	RTOTAL temperature coefficient		±600		ppm/°C	X9C102
	Ratiometric temperature coefficient			±20	ppm/°C	
$C_H/C_L/C_W$	Potentiometer capacitances		10/10/25		pF	See Circuit #3, Macro Model

Notes: (1) Absolute linearity is utilized to determine actual wiper voltage versus expected voltage = $[V_{W(n)(actual)} - V_{W(n)(expected)}] = \pm 1$ MI Maximum.
(2) Relative linearity is a measure of the error in step size between taps = $V_{W(n+1)} - [V_{W(n)} + MI] = +0.2$ MI.
(3) 1 MI = Minimum Increment = $R_{TOT}/99$
(4) Typical values are for $T_A = 25°C$ and nominal supply voltage.
(5) This parameter is not 100% tested.

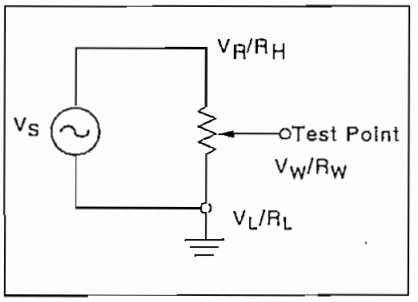
D.C. OPERATING CHARACTERISTICS (Over recommended operating conditions unless otherwise specified.)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.(4)	Max.		
I_{CC}	V_{CC} active current		1	3	mA	$\overline{CS}$ = V_{IL}, U/$\overline{D}$ = V_{IL} or V_{IH} and $\overline{INC}$ = 0.4V to 2.4V @ max. t_{CYC}
I_{SB}	Standby supply current		200	500	µA	$\overline{CS}$ = V_{CC} – 0.3V, U/$\overline{D}$ and $\overline{INC}$ = V_{SS} or V_{CC} – 0.3V
I_{LI}	$\overline{CS}$, $\overline{INC}$, U/$\overline{D}$ input leakage current			±10	µA	V_{IN} = V_{SS} to V_{CC}
V_{IH}	$\overline{CS}$, $\overline{INC}$, U/$\overline{D}$ input HIGH voltage	2		V_{CC} + 1	V	
V_{IL}	$\overline{CS}$, $\overline{INC}$, U/$\overline{D}$ input LOW voltage	–1		0.8	V	
C_{IN}(2)	$\overline{CS}$, $\overline{INC}$, U/$\overline{D}$ input capacitance			10	pF	V_{CC} = 5V, V_{IN} = V_{SS}, T_A = 25°C, f = 1MHz

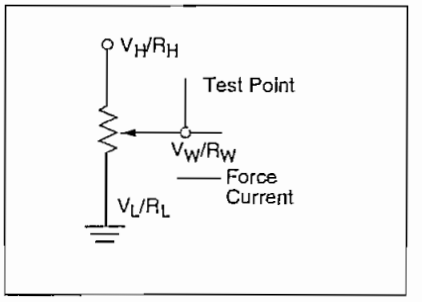
ENDURANCE AND DATA RETENTION

Parameter	Min.	Unit
Minimum endurance	100,000	Data changes per bit per register
Data retention	100	years

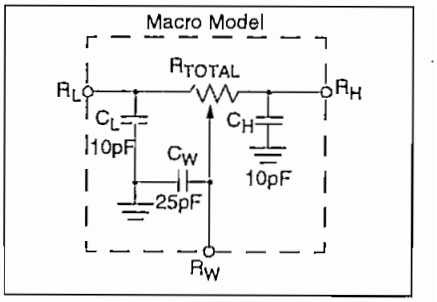
Test Circuit #1



Test Circuit #2



Test Circuit #3



A.C. CONDITIONS OF TEST

Input pulse levels	0V to 3V
Input rise and fall times	10ns
Input reference levels	1.5V

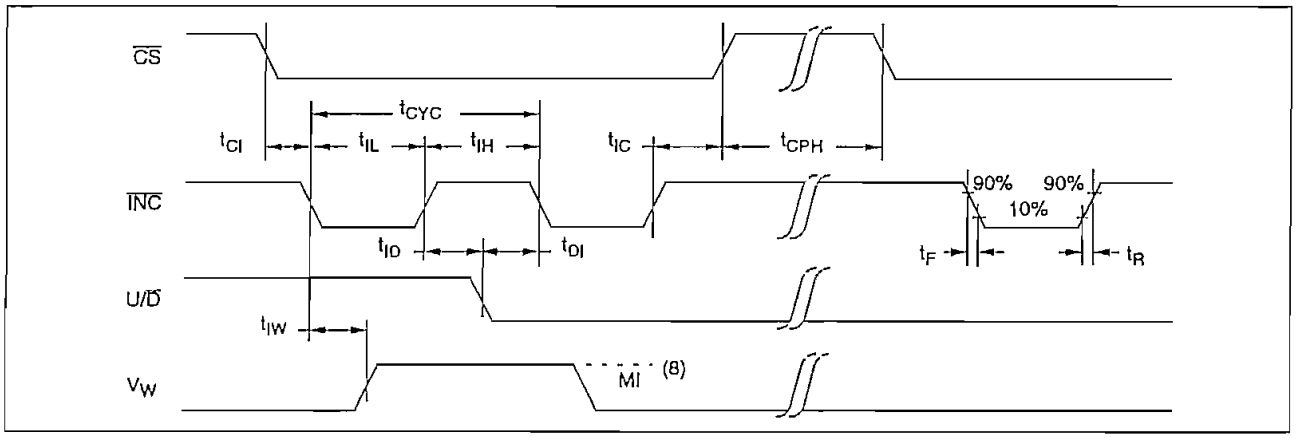
A.C. OPERATING CHARACTERISTICS (Over recommended operating conditions unless otherwise specified)

Symbol	Parameter	Limits			Unit
		Min.	Typ.(6)	Max.	
t_{CI}	$\overline{CS}$ to $\overline{INC}$ setup	100			ns
t_{ID}	$\overline{INC}$ HIGH to U/$\overline{D}$ change	100			ns
t_{DI}	U/$\overline{D}$ to $\overline{INC}$ setup	2.9			µs
t_{IL}	$\overline{INC}$ LOW period	1			µs
t_{IH}	$\overline{INC}$ HIGH period	1			µs
t_{IC}	$\overline{INC}$ inactive to $\overline{CS}$ inactive	1			µs
t_{CPH}	$\overline{CS}$ deselect time (STORE)	20			ms
t_{CPH}	$\overline{CS}$ deselect time (NO STORE)	100			ns
t_{IW}	$\overline{INC}$ to $V_{W/RW}$ change		100	500	µs
t_{CYC}	$\overline{INC}$ cycle time	4			µs
t_R, t_F(7)	$\overline{INC}$ input rise and fall time			500	µs
t_{PU}(7)	Power up to wiper stable			500	µs
t_R V_{CC}(7)	V_{CC} power-up rate	0.2		50	V/ms

POWER UP AND DOWN REQUIREMENTS

There are no restrictions on the sequencing of bias supplies V_{CC}, V+, and V- provided that all three supplies reach their final values within 1msec of each other. At all times, voltages on the potentiometer pins must be less than V+ and more than V-. The recall of the wiper position from nonvolatile memory is not in effect until all supplies reach their final value. The V_{CC} ramp rate spec is always in effect.

A.C. TIMING



Notes: (6) Typical values are for T_A = 25°C and nominal supply voltage.
(7) This parameter is periodically sampled and not 100% tested.
(8) MI in the A.C. timing diagram refers to the minimum incremental change in the V_W output due to a change in the wiper position.

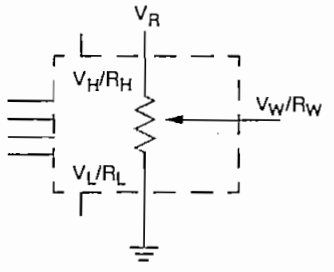
PERFORMANCE CHARACTERISTICS

Contact the factory for more information.

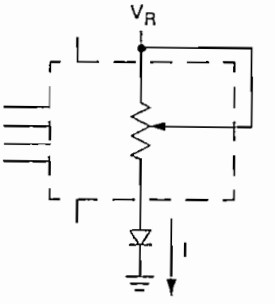
APPLICATIONS INFORMATION

Electronic digitally controlled (XCDP) potentiometers provide three powerful application advantages; (1) the variability and reliability of a solid-state potentiometer, (2) the flexibility of computer-based digital controls, and (3) the retentivity of nonvolatile memory used for the storage of multiple potentiometer settings or data.

Basic Configurations of Electronic Potentiometers



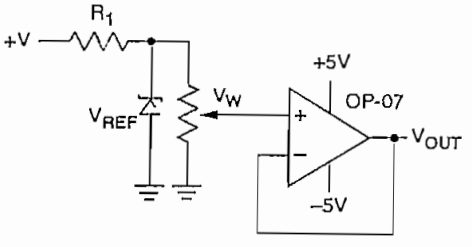
Three terminal potentiometer;
variable voltage divider



Two terminal variable resistor;
variable current

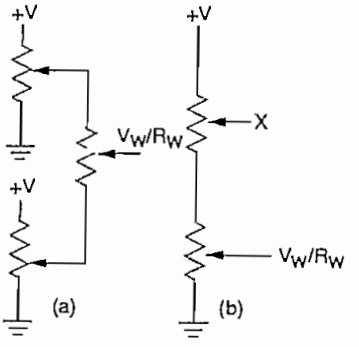
Basic Circuits

Buffered Reference Voltage

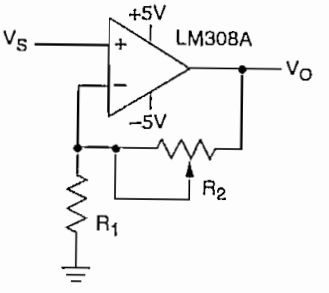


$V_{OUT} = V_W/R_W$

Cascading Techniques

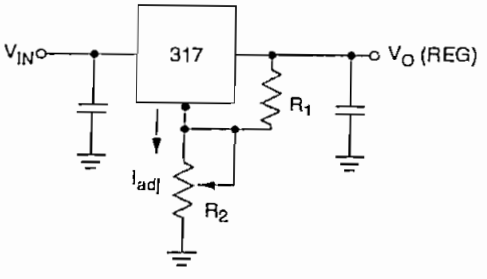


Noninverting Amplifier



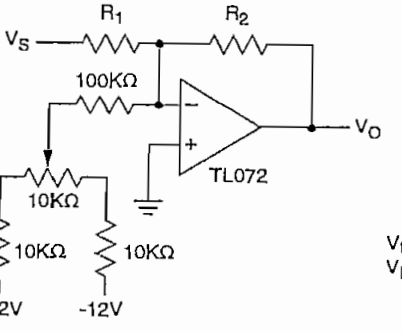
$V_O = (1+R_2/R_1)V_S$

Voltage Regulator

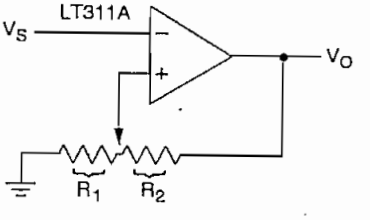


V_O (REG) = 1.25V $(1+R_2/R_1)+I_{adj} R_2$

Offset Voltage Adjustment



Comparator with Hysteresis

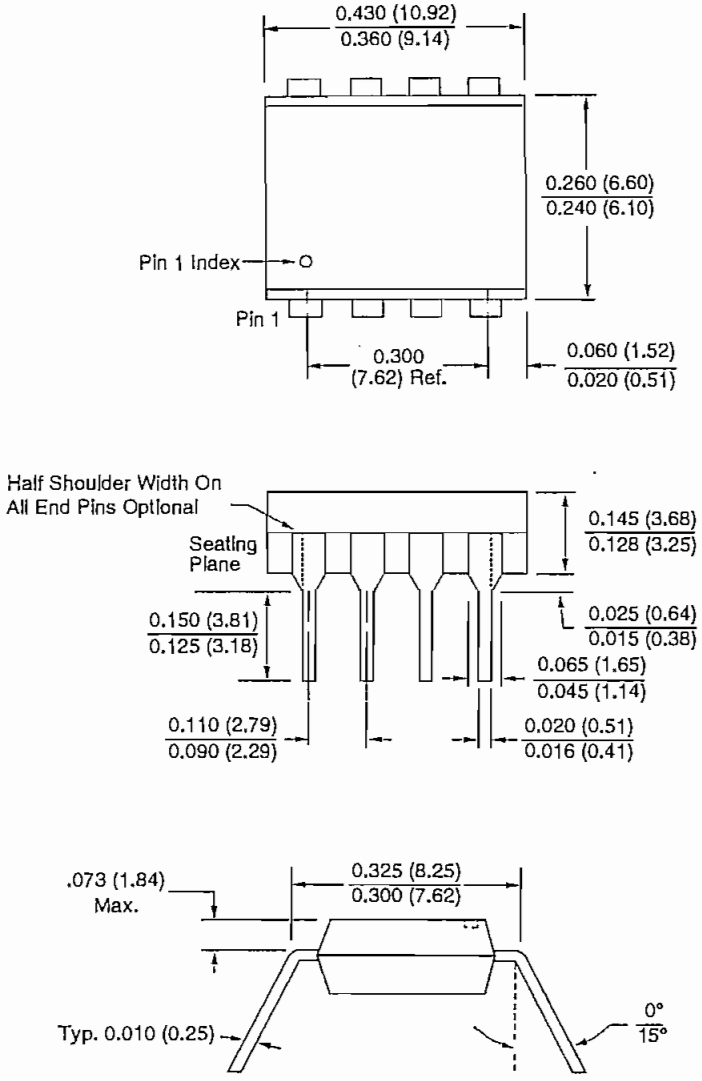


$V_{UL} = \{R_1/(R_1+R_2)\}\ V_O(max)$
$V_{LL} = \{R_1/(R_1+R_2)\}\ V_O(min)$

(for additional circuits see AN115)

PACKAGING INFORMATION

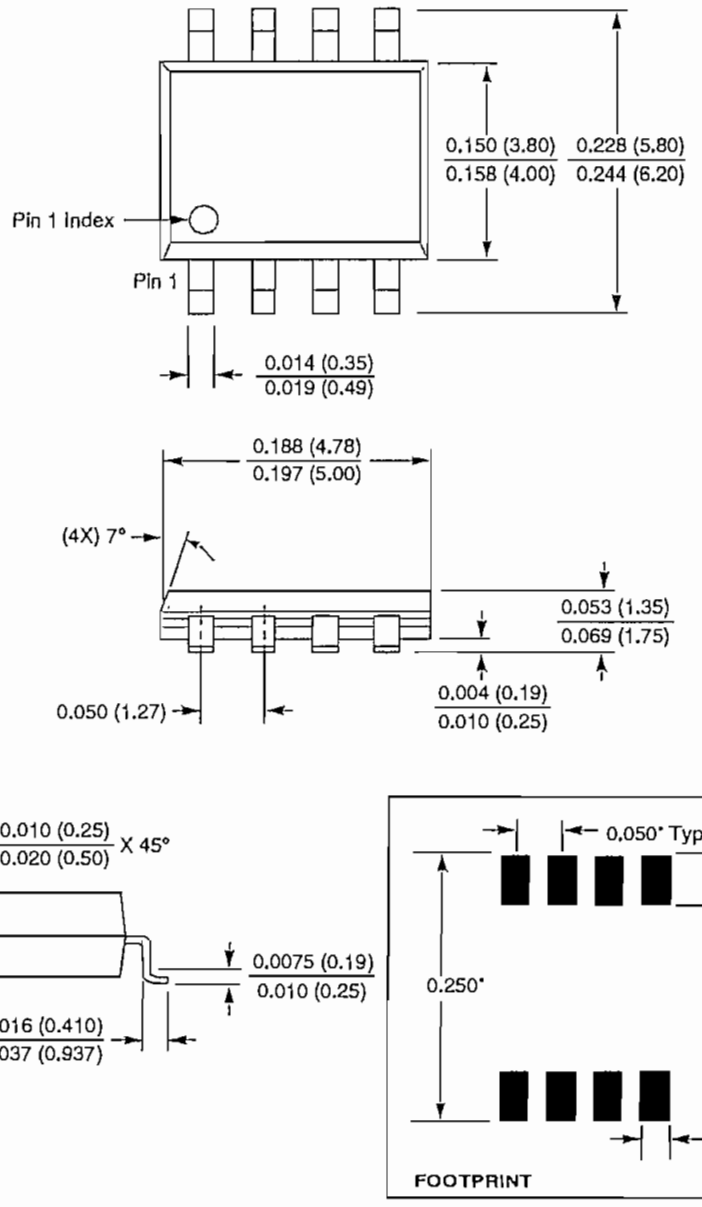
8-Lead Plastic Dual In-Line Package Type P



NOTE:
1. ALL DIMENSIONS IN INCHES (IN PARENTHESES IN MILLIMETERS)
2. PACKAGE DIMENSIONS EXCLUDE MOLDING FLASH

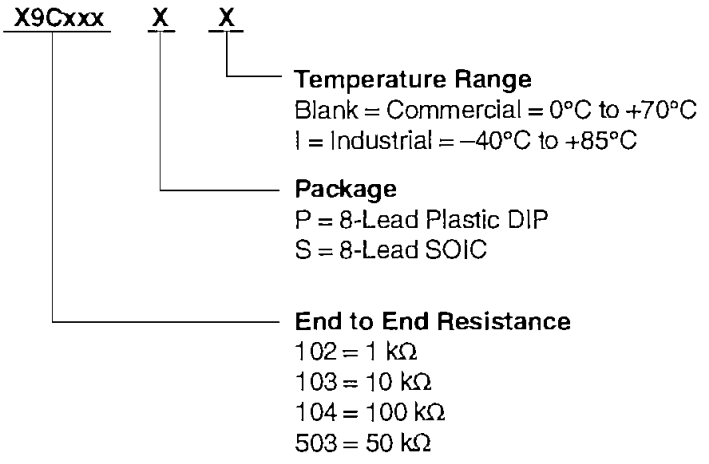
PACKAGING INFORMATION

8-Lead Plastic Small Outline Gull Wing Package Type S



NOTE: ALL DIMENSIONS IN INCHES (IN PARENTHESES IN MILLIMETERS)

Ordering Information



Physical Characteristics

Marking Includes:

– Manufacturer's Trademark

– Resistance Value or Code

– Date Code

LIMITED WARRANTY

©Xicor, Inc. 2000 Patents Pending

Devices sold by Xicor, Inc. are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Xicor, Inc. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Xicor, Inc. makes no warranty of merchantability or fitness for any purpose. Xicor, Inc. reserves the right to discontinue production and change specifications and prices at any time and without notice.

Xicor, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in a Xicor, Inc. product. No other circuits, patents, or licenses are implied.

TRADEMARK DISCLAIMER:

Xicor and the Xicor logo are registered trademarks of Xicor, Inc. AutoStore, Direct Write, Block Lock, SerialFlash, MPS, and XDCP are also trademarks of Xicor, Inc. All others belong to their respective owners.

U.S. PATENTS

Xicor products are covered by one or more of the following U.S. Patents: 4,326,134; 4,393,481; 4,404,475; 4,450,402; 4,486,769; 4,488,060; 4,520,461; 4,533,846; 4,599,706; 4,617,652; 4,668,932; 4,752,912; 4,829,482; 4,874,967; 4,883,976; 4,980,859; 5,012,132; 5,003,197; 5,023,694; 5,084,667; 5,153,880; 5,153,691; 5,161,137; 5,219,774; 5,270,927; 5,324,676; 5,434,396; 5,544,103; 5,587,573; 5,835,409; 5,977,585. Foreign patents and additional patents pending.

LIFE RELATED POLICY

In situations where semiconductor component failure may endanger life, system designers using this product should design the system with appropriate error detection and correction, redundancy and back-up features to prevent such an occurrence.

Xicor's products are not authorized for use in critical components in life support devices or systems.

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage, V_{CC}	7V
Control Input Voltages	7V
Driver Input Voltage	7V
Driver Output Voltages	+15V/ −10V
Receiver Input Voltages (DS75176B)	+15V/ −10V
Receiver Output Voltage	5.5V
Continuous Power Dissipation @ 25°C	
for M Package	675 mW (Note 5)
for N Package	900 mW (Note 4)
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 4 seconds)	260°C

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage, V_{CC}	4.75	5.25	V
Voltage at Any Bus Terminal (Separate or Common Mode)	−7	+12	V
Operating Free Air Temperature T_A			
DS75176B	0	+70	°C
DS75176BT	−40	+85	°C
Differential Input Voltage, VID (Note 6)	−12	+12	V

Electrical Characteristics (Notes 2, 3)

0°C ≤ T_A ≤ 70°C, 4.75V < V_{CC} < 5.25V unless otherwise specified

Symbol	Parameter		Conditions		Min	Typ	Max	Units
V_{OD1}	Differential Driver Output Voltage (Unloaded)		I_O = 0				5	V
V_{OD2}	Differential Driver Output Voltage (with Load)		(*Figure 1*)	R = 50Ω; (RS-422) (Note 7)	2			V
				R = 27Ω; (RS-485)	1.5			V
ΔV_{OD}	Change in Magnitude of Driver Differential Output Voltage For Complementary Output States		(*Figure 1*)	R = 27Ω			0.2	V
V_{OC}	Driver Common Mode Output Voltage						3.0	V
$\Delta\lvert V_{OC}\rvert$	Change in Magnitude of Driver Common Mode Output Voltage For Complementary Output States						0.2	V
V_{IH}	Input High Voltage	DI, DE, $\overline{RE}$, E			2			V
V_{IL}	Input Low Voltage						0.8	
V_{CL}	Input Clamp Voltage			I_{IN} = −18 mA			−1.5	
I_{IL}	Input Low Current			V_{IL} = 0.4V			−200	µA
I_{IH}	Input High Current			V_{IH} = 2.4V			20	µA
I_{IN}	Input Current	DO/RI, $\overline{DO}/\overline{RI}$	V_{CC} = 0V or 5.25V	V_{IN} = 12V			+1.0	mA
			DE = 0V	V_{IN} = −7V			−0.8	mA
V_{TH}	Differential Input Threshold Voltage for Receiver		−7V ≤ V_{CM} ≤ +12V		−0.2		+0.2	V
ΔV_{TH}	Receiver Input Hysteresis		V_{CM} = 0V			70		mV
V_{OH}	Receiver Output High Voltage		I_{OH} = −400 µA		2.7			V
V_{OL}	Output Low Voltage	RO	I_{OL} = 16 mA (Note 7)				0.5	V
I_{OZR}	OFF-State (High Impedance) Output Current at Receiver		V_{CC} = Max 0.4V ≤ V_O ≤ 2.4V				±20	µA
R_{IN}	Receiver Input Resistance		−7V ≤ V_{CM} ≤ +12V		12			kΩ
I_{CC}	Supply Current		No Load	Driver Outputs Enabled			55	mA
			(Note 7)	Driver Outputs Disabled			35	mA
I_{OSD}	Driver Short-Circuit Output Current		V_O = −7V (Note 7)				−250	mA
			V_O = +12V (Note 7)				+250	mA

Electrical Characteristics (Notes 2, 3) (Continued)

$0°C \le T_A \le 70°C$, $4.75V < V_{CC} < 5.25V$ unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ	Max	Units
I_{OSR}	Receiver Short-Circuit Output Current	$V_O = 0V$	−15		−85	mA

Note 1: "Absolute Maximum Ratings" are those beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The tables of "Electrical Characteristics" provide conditions for actual device operation.

Note 2: All currents into device pins are positive; all currents out of device pins are negative. All voltages are referenced to device ground unless otherwise specified.

Note 3: All typicals are given for $V_{CC} = 5V$ and $T_A = 25°C$.

Note 4: Derate linearly at 5.56 mW/°C to 650 mW at 70°C.

Note 5: Derate linearly 6.11 mW/°C to 400 mW at 70°C.

Note 6: Differential - Input/Output bus voltage is measured at the noninverting terminal A with respect to the inverting terminal B.

Note 7: All worst case parameters for which note 7 is applied, must be increased by 10% for DS75176BT. The other parameters remain valid for $-40°C < T_A < +85°C$.

Switching Characteristics

$V_{CC} = 5.0V$, $T_A = 25°C$

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{PLH}	Driver Input to Output	$R_{LDIFF} = 60\Omega$		12	22	ns
t_{PHL}	Driver Input to Output	$C_{L1} = C_{L2} = 100$ pF		17	22	ns
t_r	Driver Rise Time	$R_{LDIFF} = 60\Omega$			18	ns
t_f	Driver Fall Time	$C_{L1} = C_{L2} = 100$ pF (*Figure 3* and *Figure 5*)			18	ns
t_{ZH}	Driver Enable to Output High	$C_L = 100$ pF (*Figure 4* and *Figure 6*) S1 Open		29	100	ns
t_{ZL}	Driver Enable to Output Low	$C_L = 100$ pF (*Figure 4* and *Figure 6*) S2 Open		31	60	ns
t_{LZ}	Driver Disable Time from Low	$C_L = 15$ pF (*Figure 4* and *Figure 6*) S2 Open		13	30	ns
t_{HZ}	Driver Disable Time from High	$C_L = 15$ pF (*Figure 4* and *Figure 6*) S1 Open		19	200	ns
t_{PLH}	Receiver Input to Output	$C_L = 15$ pF (*Figure 2* and *Figure 7*) S1 and S2 Closed		30	37	ns
t_{PHL}	Receiver Input to Output			32	37	ns
t_{ZL}	Receiver Enable to Output Low	$C_L = 15$ pF (*Figure 2* and *Figure 8*) S2 Open		15	20	ns
t_{ZH}	Receiver Enable to Output High	$C_L = 15$ pF (*Figure 2* and *Figure 8*) S1 Open		11	20	ns
t_{LZ}	Receiver Disable from Low	$C_L = 15$ pF (*Figure 2* and *Figure 8*) S2 Open		28	32	ns
t_{HZ}	Receiver Disable from High	$C_L = 15$ pF (*Figure 2* and *Figure 8*) S1 Open		13	35	ns

AC Test Circuits

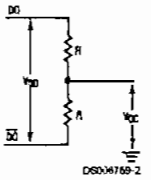
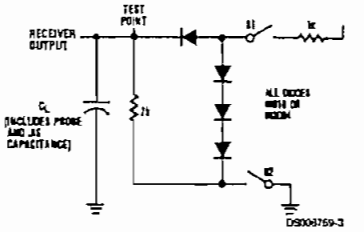


FIGURE 1.



Note: S1 and S2 of load circuit are closed except as otherwise mentioned.

FIGURE 2.

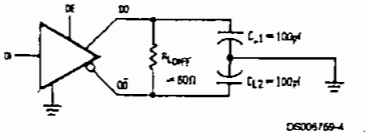
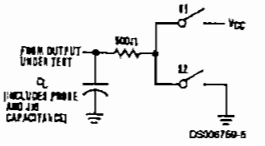


FIGURE 3.



Note: Unless otherwise specified the switches are closed.

FIGURE 4.

Switching Time Waveforms

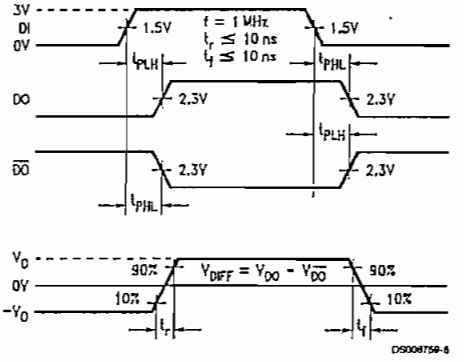


FIGURE 5. Driver Propagation Delays and Transition Times

Switching Time Waveforms (Continued)

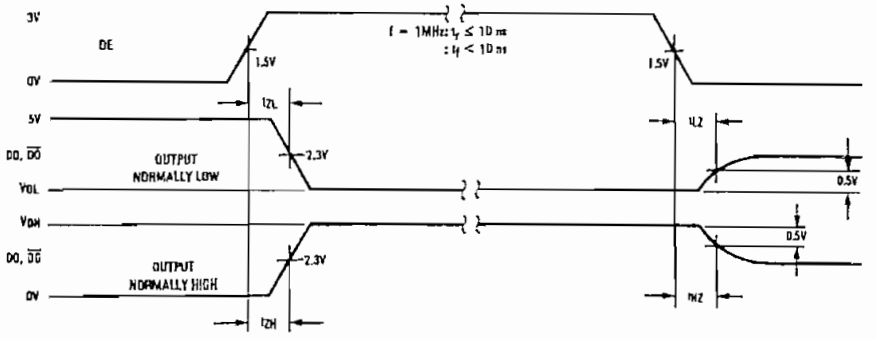
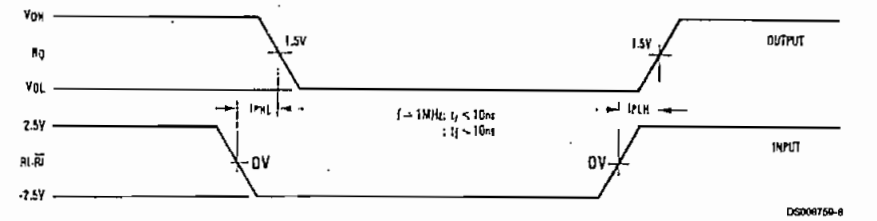


FIGURE 6. Driver Enable and Disable Times



Note: Differential input voltage may may be realized by grounding $\overline{RI}$ and pulsing RI between +2.5V and −2.5V

FIGURE 7. Receiver Propagation Delays

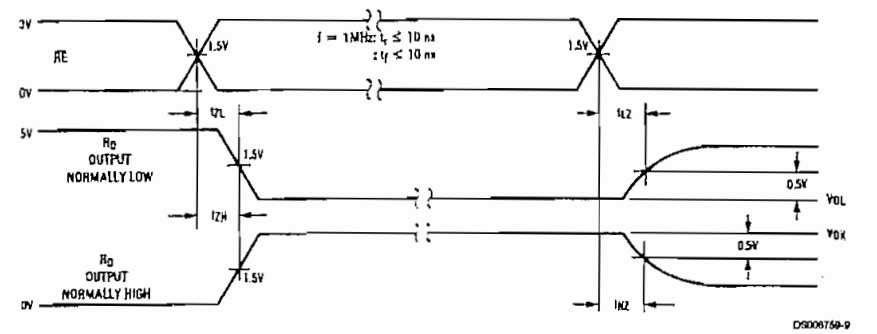


FIGURE 8. Receiver Enable and Disable Times

Function Tables

DS75176B Transmitting

Inputs			Line	Outputs	
$\overline{RE}$	DE	DI	Condition	$\overline{DO}$	DO
X	1	1	No Fault	0	1
X	1	0	No Fault	1	0
X	0	X	X	Z	Z
X	1	X	Fault	Z	Z

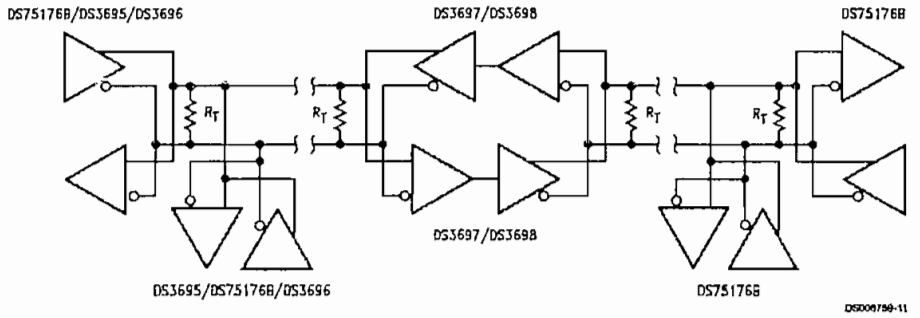
Function Tables (Continued)

DS75176B Receiving

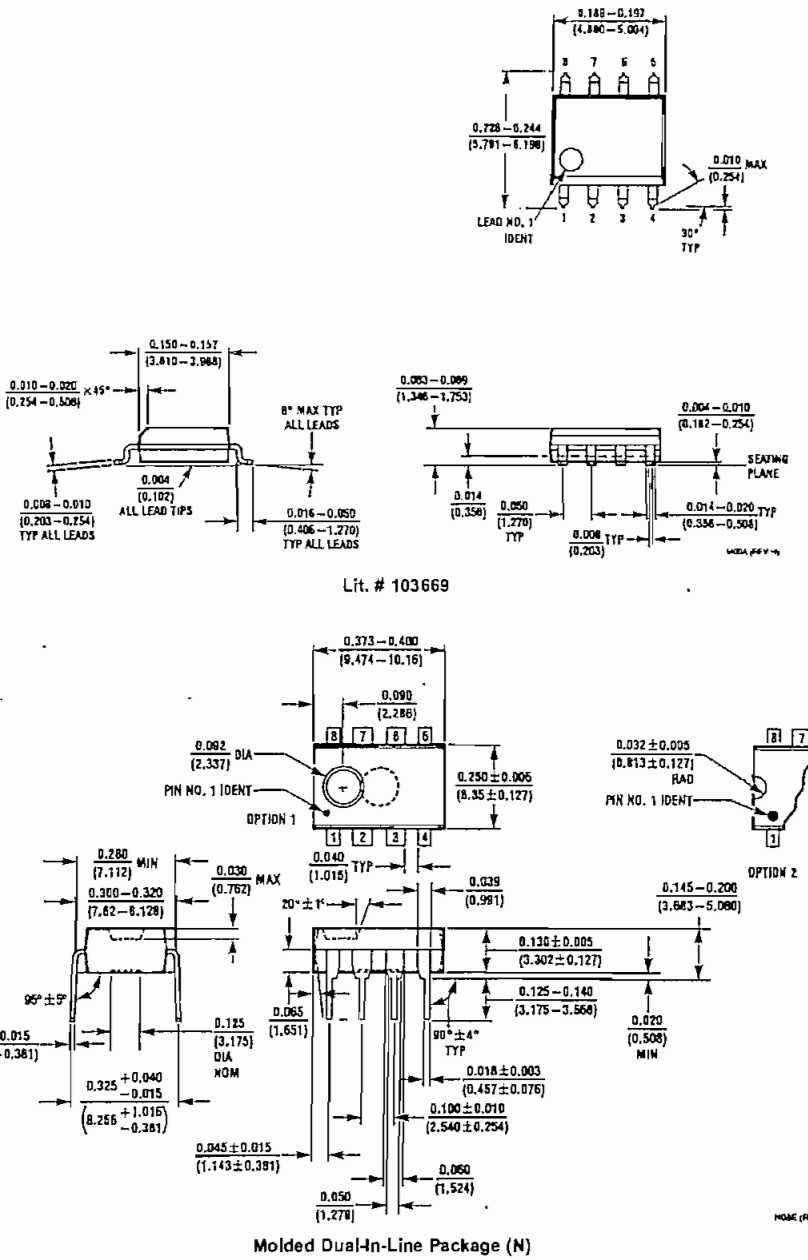
Inputs			Outputs
$\overline{RE}$	DE	RI-$\overline{RI}$	RO
0	0	≥ +0.2V	1
0	0	≤ −0.2V	0
0	0	Inputs Open**	1
1	0	X	Z

X — Don't care condition
Z — High Impedance state
Fault — Improper line conditions causing excessive power dissipation in the driver, such as shorts or bus contention situations
**This is a fail safe condition

Typical Application



Physical Dimensions inches (millimeters) unless otherwise noted



Lit. # 103669

Molded Dual-In-Line Package (N)
Order Number DS75176BN or DS75176BTN
NS Package Number N08E

National Semiconductor

September 1997

LM35/LM35A/LM35C/LM35CA/LM35D Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

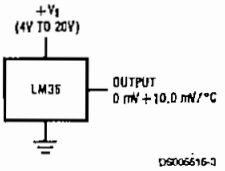
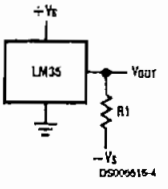


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)



Choose $R_1 = -V_S/50$ µA
V_{OUT} = +1,500 mV at +150°C
= +250 mV at +25°C
= −550 mV at −55°C

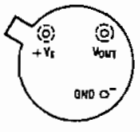
FIGURE 2. Full-Range Centigrade Temperature Sensor

TRI-STATE® is a registered trademark of National Semiconductor Corporation.

© 1999 National Semiconductor Corporation

Connection Diagrams

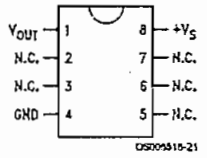
TO-46
Metal Can Package*



*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

SO-8
Small Outline Molded Package



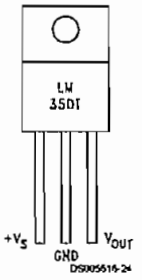
N.C. = No Connection

Top View
Order Number LM35DM
See NS Package Number M08A

TO-92
Plastic Package

Order Number LM35CZ,
LM35CAZ or LM35DZ
See NS Package Number Z03A

TO-220
Plastic Package*



*Tab is connected to the negative pin (GND).
Note: The LM35DT pinout is different than the discontinued LM35DP.

Order Number LM35DT
See NS Package Number TA03F

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.:	
TO-46 Package,	−60˚C to +180˚C
TO-92 Package,	−60˚C to +150˚C
SO-8 Package,	−65˚C to +150˚C
TO-220 Package,	−65˚C to +150˚C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300˚C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260˚C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215˚C
Infrared (15 seconds)	220˚C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55˚C to +150˚C
LM35C, LM35CA	−40˚C to +110˚C
LM35D	0˚C to +100˚C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	T_A=+25˚C	±0.2	±0.5		±0.2	±0.5		˚C
	T_A=−10˚C	±0.3			±0.3		±1.0	˚C
	T_A=T_{MAX}	±0.4	±1.0		±0.4	±1.0		˚C
	T_A=T_{MIN}	±0.4	±1.0		±0.4		±1.5	˚C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	±0.18		±0.35	±0.15		±0.3	˚C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	+10.0	+9.9, +10.1		+10.0		+9.9, +10.1	mV/˚C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25˚C	±0.4	±1.0		±0.4	±1.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	±0.5		±3.0	±0.5		±3.0	mV/mA
Line Regulation (Note 3)	T_A=+25˚C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	±0.02		±0.1	±0.02		±0.1	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25˚C	56	67		56	67		µA
	V_S=+5V	105		131	91		114	µA
	V_S=+30V, +25˚C	56.2	68		56.2	68		µA
	V_S=+30V	105.5		133	91.5		116	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25˚C	0.2	1.0		0.2	1.0		µA
	4V≤V_S≤30V	0.5		2.0	0.5		2.0	µA
Temperature Coefficient of Quiescent Current		+0.39		+0.5	+0.39		+0.5	µA/˚C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	˚C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			˚C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35			LM35C, LM35D			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy,	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		°C
LM35, LM35C	T_A=−10°C	±0.5			±0.5		±1.5	°C
(Note 7)	T_A=T_{MAX}	±0.8	±1.5		±0.8		±1.5	°C
	T_A=T_{MIN}	±0.8		±1.5	±0.8		±2.0	°C
Accuracy, LM35D	T_A=+25°C				±0.6	±1.5		°C
(Note 7)	T_A=T_{MAX}				±0.9		±2.0	°C
	T_A=T_{MIN}				±0.9		±2.0	°C
Nonlinearity	T_{MIN}≤T_A≤T_{MAX}	**±0.3**		**±0.5**	**±0.2**		**±0.5**	°C
(Note 8)								
Sensor Gain	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.8,**		**+10.0**		**+9.8,**	mV/°C
(Average Slope)			**+10.2**				**+10.2**	
Load Regulation	T_A=+25°C	±0.4	±2.0		±0.4	±2.0		mV/mA
(Note 3) 0≤I_L≤1 mA	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±5.0**	**±0.5**		**±5.0**	mV/mA
Line Regulation	T_A=+25°C	±0.01	±0.1		±0.01	±0.1		mV/V
(Note 3)	4V≤V_S≤30V	**±0.02**		**±0.2**	**±0.02**		**±0.2**	mV/V
Quiescent Current	V_S=+5V, +25°C	56	80		56	80		µA
(Note 9)	V_S=+5V	**105**		**158**	**91**		**138**	µA
	V_S=+30V, +25°C	56.2	82		56.2	82		µA
	V_S=+30V	**105.5**		**161**	**91.5**		**141**	µA
Change of	4V≤V_S≤30V, +25°C	0.2	2.0		0.2	2.0		µA
Quiescent Current	4V≤V_S≤30V	**0.5**		**3.0**	**0.5**		**3.0**	µA
(Note 3)								
Temperature		**+0.39**		**+0.7**	**+0.39**		**+0.7**	µA/°C
Coefficient of								
Quiescent Current								
Minimum Temperature	In circuit of	+1.5		+2.0	+1.5		+2.0	°C
for Rated Accuracy	*Figure 1*, I_L=0							
Long Term Stability	T_J=T_{MAX}, for	±0.08			±0.08			°C
	1000 hours							

Note 1: Unless otherwise noted, these specifications apply: −55°C≤T_J≤+150°C for the LM35 and LM35A; −40°≤T_J≤+110°C for the LM35C and LM35CA; and 0°≤T_J≤+100°C for the LM35D. V_S=+5Vdc and I_{LOAD}=50 µA, in the circuit of *Figure 2*. These specifications also apply from +2°C to T_{MAX} in the circuit of *Figure 1*. Specifications in boldface apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400°C/W, junction to ambient, and 24°C/W junction to case. Thermal resistance of the TO-92 package is 180°C/W junction to ambient. Thermal resistance of the small outline molded package is 220°C/W junction to ambient. Thermal resistance of the TO-220 package is 90°C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Note 3: Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

Note 4: Tested Limits are guaranteed and 100% tested in production.

Note 5: Design Limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 6: Specifications in boldface apply over the full rated temperature range.

Note 7: Accuracy is defined as the error between the output voltage and 10mV/°C times the device's case temperature, at specified conditions of voltage, current, and temperature (expressed in °C).

Note 8: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the device's rated temperature range.

Note 9: Quiescent current is defined in the circuit of *Figure 1*.

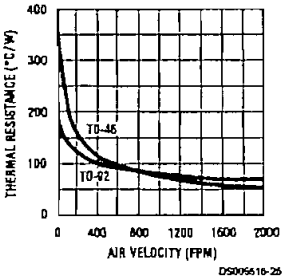
Note 10: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.

Note 11: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Note 12: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

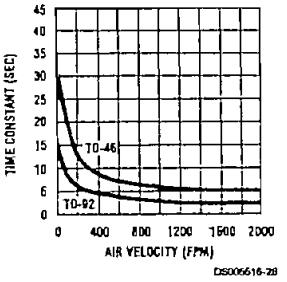
Typical Performance Characteristics

Thermal Resistance
Junction to Air



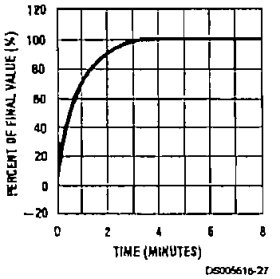
DS005616-25

Thermal Time Constant



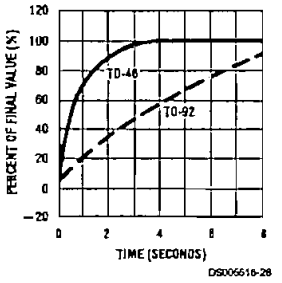
DS005616-28

Thermal Response
in Still Air



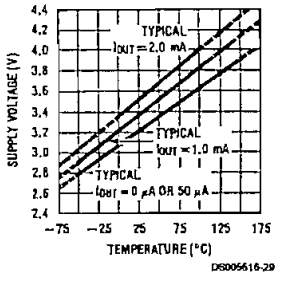
DS005616-27

Thermal Response in
Stirred Oil Bath



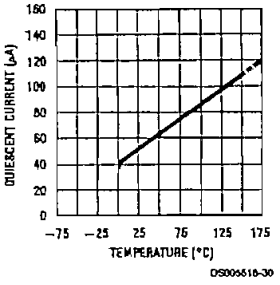
DS005616-26

Minimum Supply
Voltage vs. Temperature



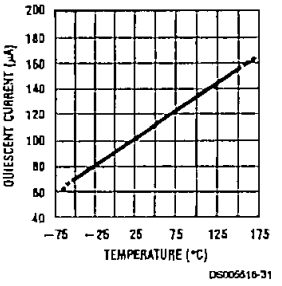
DS005616-29

Quiescent Current
vs. Temperature
(In Circuit of *Figure 1.*)



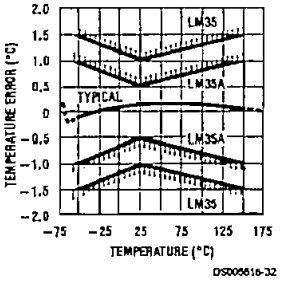
DS005616-30

Quiescent Current
vs. Temperature
(In Circuit of *Figure 2.*)



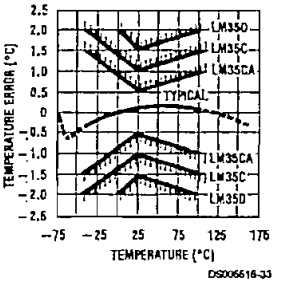
DS005616-31

Accuracy vs. Temperature
(Guaranteed)



DS005616-32

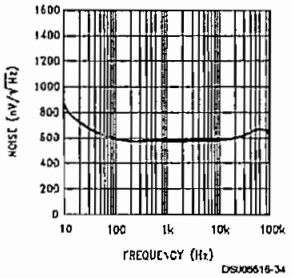
Accuracy vs. Temperature
(Guaranteed)



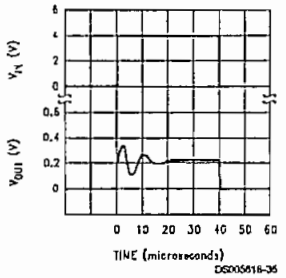
DS005616-33

Typical Performance Characteristics (Continued)

Noise Voltage



Start-Up Response



Applications

The LM35 can be applied easily in the same way as other integrated-circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01°C of the surface temperature.

This presumes that the ambient air temperature is almost the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. This is especially true for the TO-92 plastic package, where the copper leads are the principal thermal path to carry heat into the device, so its temperature might be closer to the air temperature than to the surface temperature.

To minimize this problem, be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the LM35 die's temperature will not be affected by the air temperature.

The TO-46 metal package can also be soldered to a metal surface or pipe without damage. Of course, in that case the V− terminal of the circuit will be grounded to that metal. Alternatively, the LM35 can be mounted inside a sealed-end metal tube, and can then be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true if the circuit may operate at cold temperatures where condensation can occur. Printed-circuit coatings and varnishes such as Humiseal and epoxy paints or dips are often used to insure that moisture cannot corrode the LM35 or its connections.

These devices are sometimes soldered to a small light-weight heat fin, to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand, a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

Temperature Rise of LM35 Due To Self-heating (Thermal Resistance, θ_{JA})

	TO-46, no heat sink	TO-46*, small heat fin	TO-92, no heat sink	TO-92**, small heat fin	SO-8 no heat sink	SO-8** small heat fin	TO-220 no heat sink
Still air	400°C/W	100°C/W	180°C/W	140°C/W	220°C/W	110°C/W	90°C/W
Moving air	100°C/W	40°C/W	90°C/W	70°C/W	105°C/W	90°C/W	26°C/W
Still oil	100°C/W	40°C/W	90°C/W	70°C/W			
Stirred oil	50°C/W	30°C/W	45°C/W	40°C/W			
(Clamped to metal, Infinite heat sink)	(24°C/W)				(55°C/W)		

*Wakefield type 201, or 1" disc of 0.020" sheet brass, soldered to case, or similar.
**TO-92 and SO-8 packages glued and leads soldered to 1" square of 1/16" printed circuit board with 2 oz. foil or similar.

Typical Applications

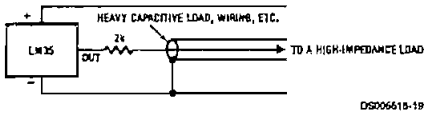


FIGURE 3. LM35 with Decoupling from Capacitive Load

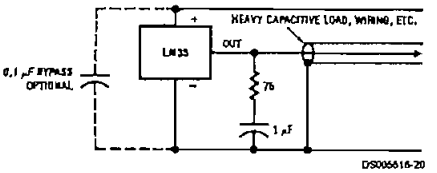


FIGURE 4. LM35 with R-C Damper

CAPACITIVE LOADS

Like most micropower circuits, the LM35 has a limited ability to drive heavy capacitive loads. The LM35 by itself is able to drive 50 pf without special precautions. If heavier loads are anticipated, it is easy to isolate or decouple the load with a resistor; see *Figure 3*. Or you can improve the tolerance of capacitance with a series R-C damper from output to ground; see *Figure 4*.

When the LM35 is applied with a 200Ω load resistor as shown in *Figure 5*, *Figure 6* or *Figure 8* it is relatively immune to wiring capacitance because the capacitance forms a bypass from ground to input, not on the output. However, as with any linear circuit connected to wires in a hostile environment, its performance can be affected adversely by intense electromagnetic sources such as relays, radio transmitters, motors with arcing brushes, SCR transients, etc, as its wiring can act as a receiving antenna and its internal junctions can act as rectifiers. For best results in such cases, a bypass capacitor from V_{IN} to ground and a series R-C damper such as 75Ω in series with 0.2 or 1 μF from output to ground are often useful. These are shown in *Figure 13*, *Figure 14*, and *Figure 16*.

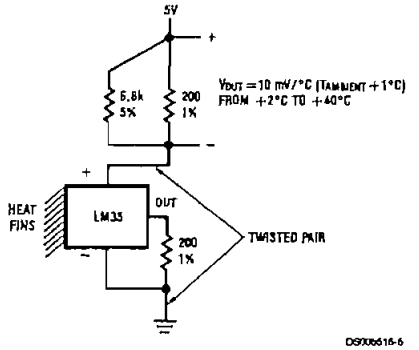


FIGURE 5. Two-Wire Remote Temperature Sensor (Grounded Sensor)

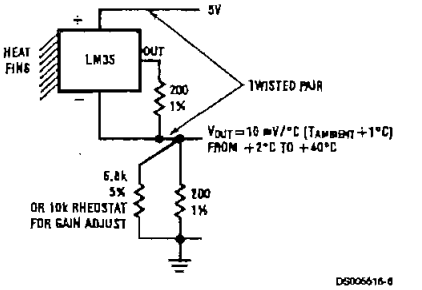


FIGURE 6. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

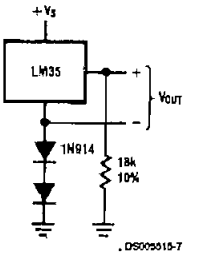


FIGURE 7. Temperature Sensor, Single Supply, −55° to +150°C

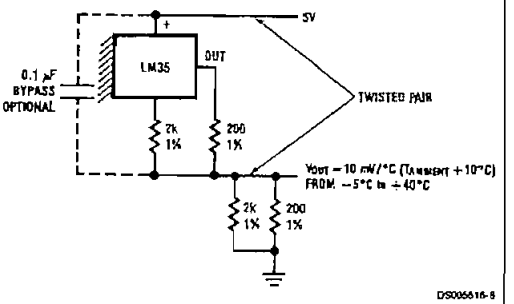


FIGURE 8. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

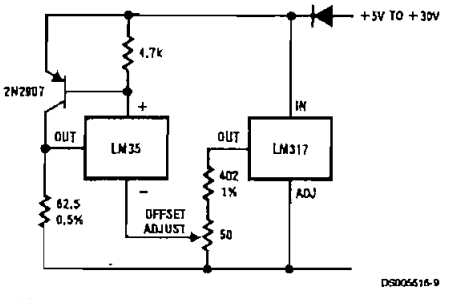


FIGURE 9. 4-To-20 mA Current Source (0°C to +100°C)

Typical Applications (Continued)

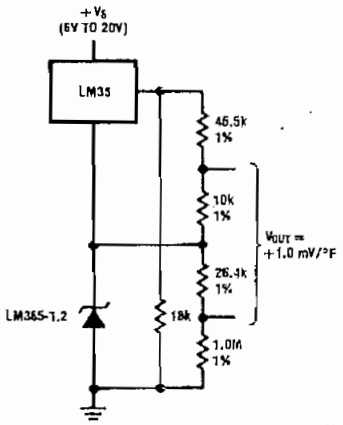


FIGURE 10. Fahrenheit Thermometer

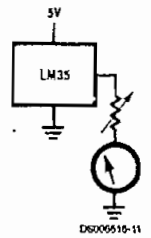


FIGURE 11. Centigrade Thermometer (Analog Meter)

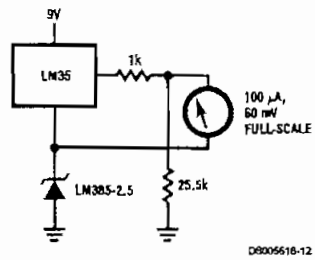


FIGURE 12. Fahrenheit ThermometerExpanded Scale Thermometer
(50° to 80° Fahrenheit, for Example Shown)

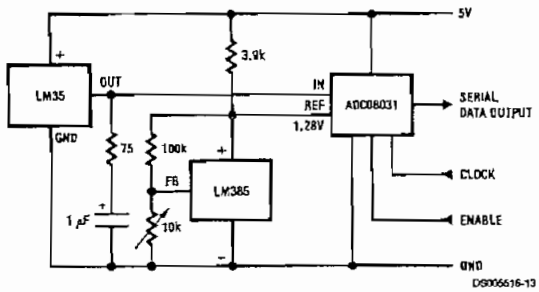


FIGURE 13. Temperature To Digital Converter (Serial Output) (+128°C Full Scale)

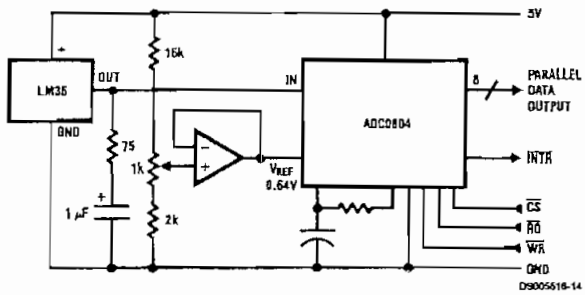
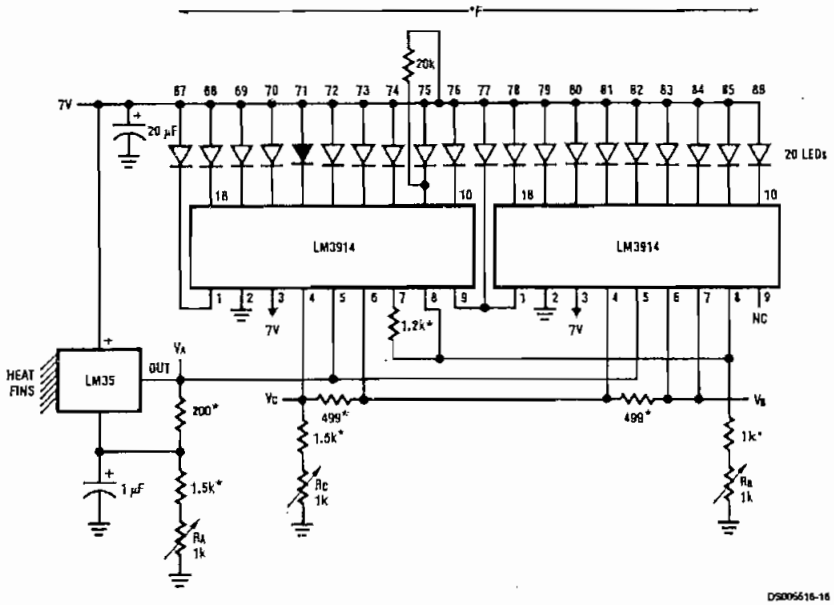


FIGURE 14. Temperature To Digital Converter (Parallel TRI-STATE™ Outputs for Standard Data Bus to µP Interface) (128°C Full Scale)

Typical Applications (Continued)



*=1% or 2% film resistor
Trim R_B for V_B=3.075V
Trim R_C for V_C=1.955V
Trim R_A for V_A=0.075V + 100mV/°C x $T_{ambient}$
Example, V_A=2.275V at 22°C

FIGURE 15. Bar-Graph Temperature Display (Dot Mode)

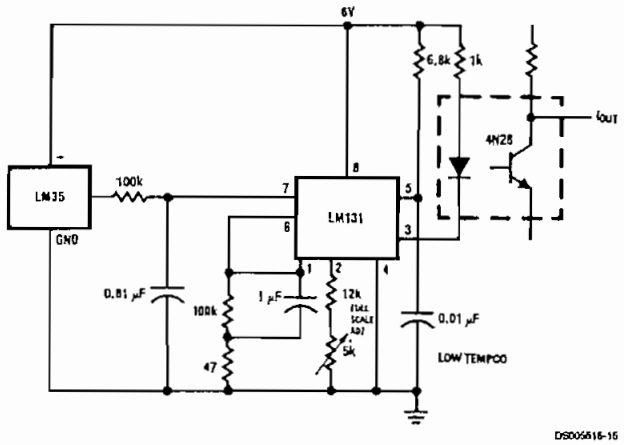
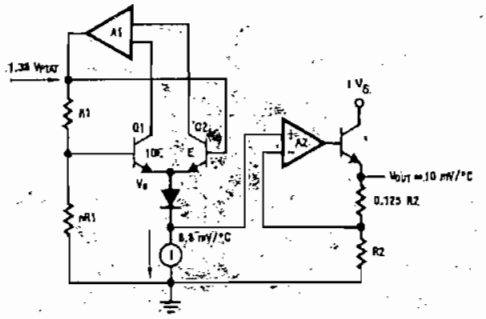


FIGURE 16. LM35 With Voltage-To-Frequency Converter And Isolated Output (2°C to +150°C; 20 Hz to 1500 Hz)

Block Diagram



DS005618-23