ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

TESIS DE GRADO

Previa la obtención del Título de:

Ingeniero en Electrónica y Telecomunicaciones

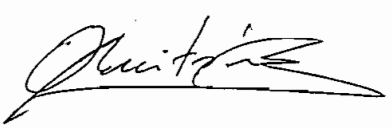
Temporizador programable controlado por un microcontrolador

FRANCISCO XAVIER TOLEDO RIVADENEIRA

QUITO, AGOSTO DE 1995

Certifico que la presente tesis fue realizada en su totalidad por el Señor

FRANCISCO XAVIER TOLEDO RIVADENEIRA



ING. OSWALDO BUITRON

DIRECTOR DE TESIS

AGRADECIMIENTO

A mis padres por el apoyo que a lo largo de toda mi vida me han brindado.

A mis hermanos por estar siempre a mi lado.

Al Ing. Oswaldo Buitrón que constantemente me impulso para lograr tan ansiado objetivo.

A mis amigos que de una u otra manera colaboraron para que este trabajo haya alcanzado los frutos deseados.

Al Ing. Fausto Ayarza.

A Dios por haberme dado la vida.

DEDICATORIA

A mis padres que amor, sacrificio siempre me han apoyado para culminar con éxitos mis metas.

A mis hermanos, a mis amigos.

A todos los que se maravillan de cada nuevo día.

INDICE

CAPITULO I

CAPITULO II

CAPITULO III

C A P I T U L O I

G E N E R A L I D A D E S

La necesidad de que distintos procesos tengan como referencia al tiempo, ha determinado la conveniencia de contar con dispositivos temporizadores y que los mismos, dadas las múltiples ventajas, sean ejecutados por medio de circuitos electrónicos.

En este capítulo se trata de enfocar los aspectos fundamentales de esta clase de circuitos, su definición, de acuerdo con la función que cumplen, algunos criterios de selección, las características del temporizador a diseñarse en el presente trabajo; sus alternativas de diseño, y una conclusión de estas posibilidades.

1.1.- CARACTERISTICAS GENERALES DE LOS TEMPORIZADORES

Existen muchos circuitos cuya principal aplicación es la de temporizar. Entre los principales se pueden destacar, los circuitos osciladores, los circuitos de pulsos y conmutación y los de temporización

programables.

Un circuito oscilador electrónico, normalmente se lo define como un dispositivo que convierte una entrada dc a una salida variable en el tiempo. Los osciladores tienen un elemento de su circuito que puede variar para producir diferentes frecuencias; f = 1/t.

Un oscilador debe tener retroalimentación positiva, la cual servirá de entrada para el mismo. Si la ganancia en lazo y el ángulo de fase son los adecuados aparecerá una señal a la salida sin que se aplique ninguna señal a la entrada. El oscilador no crea energía, simplemente la transforma de una fuente dc en energía ac.

Un oscilador debe tener elementos que determinen su frecuencia, generalmente elementos pasivos, un mecanismo que limite su amplitud, y una suficiente ganancia de lazo cerrado para compensar las pérdidas en el circuito.

Los circuitos de pulsos y conmutación son circuitos de temporización que utilizan elementos electrónicos que en la actualidad debido a la tecnología digital se

comportan como conmutadores, es decir, que durante un cierto tiempo actúan como un interruptor abierto que no permite el paso de corriente y en otro como un interruptor cerrado.

Existe una gran variedad de circuitos de pulsos y conmutación entre los que podemos mencionar los circuitos multivibradores en sus distintas formas de operación.

Entre los circuitos que realizan estas funciones se encuentran circuitos integrados como el TIMER 555, el monoestable SN74121 o Flip - Flops.

Los circuitos de temporización programables generan cambios de estado en períodos específicos de tiempo previamente programados en los elementos del circuito. La programación de la temporización puede ser en tiempo virtual o en tiempo real. El presente trabajo trata un sistema de temporización programable en tiempo real el cual controlará diferentes equipos hasta en períodos de una semana.

1.2.- APLICACIONES DE LOS TEMPORIZADORES

Los circuitos osciladores y los circuitos de pulsos y conmutación tiene un campo de utilización amplio y variado en diseños electrónicos, en tanto que, los circuitos de temporización programables normalmente son utilizados en control de eventos ya sea en tiempo virtual o en tiempo real.

El temporizador programable a diseñarse trata de enfocar el área de programación en tiempo real a través de un diseño específico, el cual permitirá controlar diferentes equipos en su activado y desactivado.

De acuerdo con esta característica es de interés mencionar algunas aplicaciones en las cuales el sistema podrá ser de utilidad. En el campo industrial, podría ser utilizado para el control en la operación de máquinas en intervalos de tiempo, evitando el encendido y apagado en forma manual, el cual será automático de acuerdo con la programación establecida.

En el caso de un banco, el sistema podría ser usado en seguridad, por ejemplo: abrir una bóveda sólo a cierta hora y cerrarla a otra hora conveniente en forma

automática o por medio de una clave.

En sistemas de redes de procesamiento de datos, el sistema podrá ser usado para activar y desactivar estaciones remotas, controladores de terminales, procesadores de comunicaciones con lo que se lograría una optimización de los recursos.

Se lo podría utilizar en hospitales como equipo de señalización para suministrar a los pacientes los medicamentos a la hora exacta.

En labores del hogar, activando o desactivando diferentes electrodomésticos de acuerdo con la programación que realice el ama de casa.

1.3.- CARACTERISTICAS DEL SISTEMA A DISEÑARSE.

Los requerimientos funcionales de diseño que debe cumplir el temporizador programable posibilitará la operación para el control de encendido y apagado de 4 interruptores o de relés, en períodos de hasta una semana y con resolución de aproximadamente 1 segundo. Para cada interruptor se podrá programar 4 instantes de encendido y apagado. El circuito poseerá un reloj en

tiempo real, cuyo resultado podrá ser visualizado en un display del tipo LCD.

El sistema tendrá las siguientes características:

a) POSIBILIDAD DE IGUALACION DEL RELOJ DE TIEMPO REAL.- Permitirá que el sistema funcione de acuerdo con una referencia de tiempo real y que la misma pueda ser modificada.

b) POSIBILIDAD DE SEÑALIZACION DE ENCENDIDO Y APAGADO DE UN INTERRUPTOR.- Esta característica permitirá al sistema emitir alguna forma de señalización, de manera que el usuario se informe de la ocurrencia de un evento en ese tiempo.

c) POSIBILIDAD DE QUE A PARTIR DE UN TIEMPO DADO SE ACTIVE DETERMINADO INTERRUPTOR, Y QUE EL MISMO SE DESCONECTE DESPUES DE UN TIEMPO DADO.- Esta posibilidad es la más usual en sistemas de control ya que se requiere que un interruptor se active a una hora fija y se desactive a otra hora de acuerdo con una programación establecida.

d) POSIBILIDAD DE DARLE UNA ORDEN PARA QUE EL

FUNCIONAMIENTO SEA PERIODICO.- Algunas veces se requiere que un determinado equipo se active o se desactive todos los días de la semana a determinadas horas, para lo cual sería necesario que se programe esta posibilidad, en una sola ocasión.

e) POSIBILIDAD DE QUE DESPUES DE HABER DADO UNA ORDEN DE FUNCIONAMIENTO PERIODICO SE PUEDAN PROGRAMAR EXCLUYENTES.- Muchas veces es conveniente tener en cuenta que exista en el calendario días de vacaciones o no laborables en los cuales los equipos no deben realizar ninguna labor por lo que, el sistema deberá excluir estos días del funcionamiento periódico normal.

f) POSIBILIDAD DE OBSERVAR COMO ESTAN PROGRAMADOS LOS INTERRUPTORES.- Si se olvida como están programados los interruptores el sistema debe ser capaz de recordarle al usuario el estado de cada uno de los interruptores en todos los días de la semana, para lo cual es necesario realizar una acción de chequeo y visualización de la información almacenada.

g) POSIBILIDAD DE BORRAR LO PROGRAMADO PARA TODOS LOS INTERRUPTORES.- Algunas ocasiones es indispensable

borrar toda acción programada para los interruptores, con el objeto de ajustarlos a una nueva realidad que satisfaga otros requerimientos.

h) POSIBILIDAD DE BORRAR LO PROGRAMADO PARA UN INTERRUPTOR.- Borrar tan sólo lo concerniente a un interruptor en especial y si es necesario reprogramarlo.

i) POSIBILIDAD DE BORRAR LO PROGRAMADO PARA UN DIA.- Si se desea borrar lo programado para un día de la semana en particular el sistema debe poseer esta capacidad y además la de realizar una reprogramación para ese día.

j) POSIBILIDAD DE ACTIVAR O DESACTIVAR UN INTERRUPTOR DIRECTAMENTE.- Cuando se requiera activar o desactivar un interruptor en forma inmediata y directa, el sistema debe proveer acceso directo a los interruptores.

Además, es necesario considerar la posibilidad de agregar otras características dependiendo de las alternativas de diseño que se tome para la solución del sistema propuesto.

1.4.- ALTERNATIVAS DE SOLUCION

Existen varias metodologías para resolver el problema. Algunas ya se las comercializa en el mercado pero su costo es elevado.

Como primera opción tenemos el uso de un computador personal, el cual podría servir como cerebro para el control de una gran cantidad de procesos. Se podría desarrollar un programa en cualquier lenguaje, el cual quede como residente. El control se lo podría realizar a través del pórtico paralelo, el cual actuará sobre relés capaces de manejar una mayor cantidad de corriente.

El problema que existe es que esta solución es muy costosa, ya que se necesitaría un equipo confiable y que se mantenga en funcionamiento durante todo el tiempo. Otro inconveniente es la dificultad de su transporte debido a su volumen.

En el mercado existen equipos llamados Controladores Lógicos Programables, pero su costo es todavía alto. Además utilizan programas dedicados y se requiere de un equipo adicional para el desarrollo de nuevas

aplicaciones.

Como otra alternativa se tiene el uso de un microcontrolador por medio del cual existe la posibilidad de manejar diversos estados, valiéndose de la facilidades que presta por tener integrados puertos de entrada y salida.

Entre los microcontroladores se tiene la posible utilización del microcontrolador DS5000T. Este es un microcontrolador compatible con el estándar industrial i51 tanto en su set de instrucciones como en su configuración de pines. Tiene todas las características del i8051 y adicionalmente las siguientes:

- Batería de litio interna que preserva la memoria y las funciones de reloj en ausencia de energía.
- 32 kbytes de memoria incluidos en el chip que pueden ser utilizados como ROM y/o RAM.
- Reloj calendario en tiempo real incluído en el chip, lo cual permite el registro de eventos relacionados con fecha y hora.

Debido a las características con las que cuenta el

microcontrolador permite que se puedan realizar sistemas con poca circuitería adicional, lo que redunda en una simplicidad del hardware requerido para una aplicación determinada.

Esta alternativa no se descarta para su posterior uso, pero prevalece el inconveniente de su costo en comparación con la familia del MCS-51.

Por último se tiene una alternativa parecida a la anterior, pero utilizando un microcontrolador de la familia del MCS-51, con el apoyo de circuitos LSI adicionales.

El diseño actual de los equipos se lo realiza en base al apoyo de varios microcontroladores dedicados para mejorar en velocidad de procesamiento y con el objeto de no recargar a uno sólo de ellos. A esta filosofía se la conoce como cliente servidor.

1.5.- CONCLUSIONES

Realizando un análisis del equipo a diseñarse se puede concluir que debe cumplir con las siguientes características:

- De preferencia,portátil
- Su costo conveniente
- Versátil y flexible a cambios que se puedan requerir a futuro.

Por estos motivos la solución que se va a utilizar es la que involucra el uso de un microcontrolador. Entre las ventajas con las que se cuenta se tiene:

- la posibilidad de múltiples puertos de entrada/salida programables
- capacidad de almacenamiento de datos
- capacidad de comunicación a través de un puerto serial con diversos dispositivos
- manejo de interrupciones y temporizadores.

Por la facilidad de comunicación serial se utilizará un interfaz con un computador personal con el objeto de descargar la información con lo cual la programación será más versátil y ahorrará parte del trabajo al microcontrolador.

El desarrollo de la tecnología actual hace posible que se pueda encontrar solución a determinados problemas, mediante circuitería digital en base a un microcontrolador, aplicación que permite mayor

flexibilidad y adaptación a los cambios.

Se eligió el microcontrolador de la familia del MCS-51 ya que presenta grandes ventajas respecto a un microprocesador (pórticos, memoria incorporados) y es el mayor familiaridad.

C A P I T U L O I I

B R E V E E S T U D I O T E O R I C O

Una vez que se han definido las características del temporizador ha ser diseñado, y de haber concluído que la solución más adecuada para su implantación es desarrollarlo como un circuito programable; en este capítulo se realiza un estudio de los diversos componentes que conforman el temporizador y su interrelación.

2.1 DIAGRAMA DE BLOQUES DEL TEMPORIZADOR PROGRAMABLE.

El sistema digital a ser diseñado, será entonces programable y constará básicamente de las tres unidades principales que se detallan a continuación:

- Unidad Central de Proceso (CPU).
- Unidad de Almacenamiento o Memoria (M).
- Unidad de Entrada/Salida de la información.

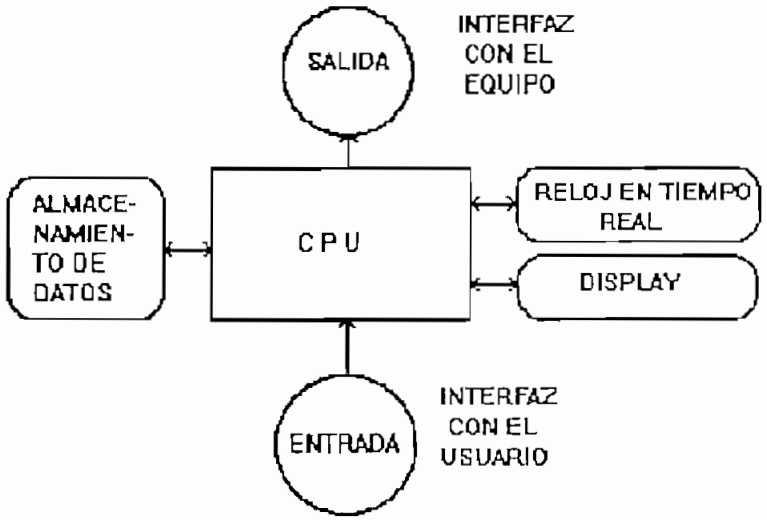


Fig 2.1 Diagrama de bloques del temporizador

La etapa o unidad fundamental del sistema, constituye la CPU como puede apreciarse en la Fig. 2.1 donde se muestra un diagrama de bloques del temporizador a diseñarse.

De acuerdo con el diagrama de bloques indicado se plantean los siguientes requerimientos que a su vez se constituirían en especificaciones:

- Dispositivo de Entrada: Interfaz serial con el computador.
- Dispositivo de Procesamiento: CPU + reloj en tiempo real
- Dispositivo de Salida: Display + Interfaz de relés

Cada uno de los dispositivos de Entrada/Salida requiere

de un interfaz para poder acoplarse a la unidad central de procesamiento, que en nuestro caso se trata de un microcontrolador.

2.1.1 DESCRIPCION GENERAL

El temporizador programable permitirá programar el encendido y apagado de 4 interruptores, en períodos de hasta una semana y con resolución de aproximadamente un segundo. Para cada interruptor se podrá programar 4 momentos de encendido y apagado. Para el propósito antes indicado, el sistema deberá poseer un reloj de tiempo real.

Los dispositivos de entrada/salida se encargarán de proporcionar la vía de comunicación entre el sistema y el usuario.

Mediante el interfaz serial de la computadora y del equipo de temporización se podrá establecer una comunicación para programar el estado de los interruptores cada vez que sea necesario.

Se consultará continuamente al reloj en tiempo real y esa información se la mostrará en un display de cristal

líquido con lo cual se tendrá la hora actualizada para el usuario.

Se debe disponer de la suficiente cantidad de memoria para el almacenamiento de datos y el almacenamiento del programa de control que será el encargado de administrar los recursos físicos del sistema, con la ayuda de la CPU que es el cerebro del temporizador.

2.2 CIRCUITOS DE ENTRADA Y SALIDA DE LA INFORMACION

Los circuitos de entrada son los encargados de llevar la información del usuario al sistema temporizador, y los circuitos de salida, por el contrario, llevarán la información del temporizador al usuario o a los dispositivos que se desea controlar.

2.2.1 CIRCUITO DE ENTRADA

EL circuito para entrada de los datos es el pórtico serial. Existe un interfaz gráfico en el computador que permite que la información sea ingresada correctamente y corregida cada vez que el usuario crea conveniente. Cabe resaltar que la única forma de ingresar datos es mediante la ayuda de un computador.

Con la utilización de este elemento adicional se disminuye el trabajo del microcontrolador en cuanto a las subrutinas de ordenamiento de datos, lo cual se va a realizar en el computador. Una vez ordenados los mismos se transferirá los datos al temporizador, utilizando el pórtico serial del microcontrolador.

A continuación en la Fig 2.2 se presenta un diagrama que ilustrra la conexión del computador al circuito de entrada del temporizador.

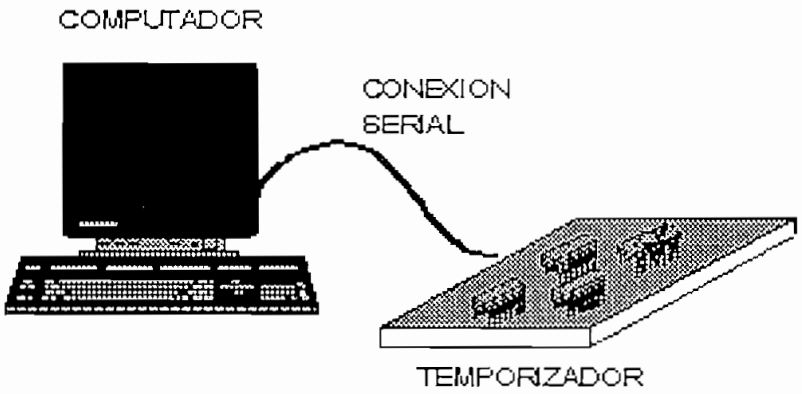


Fig 2.2 Conexión del PC al circuito de entrada

2.2.2 CIRCUITOS DE SALIDA DE LA INFORMACION

El circuito de salida está compuesto por un display de cristal líquido, el cual está compuesto por un arreglo de caracteres compuesto por 2 filas por 16 columnas en el cual se puede desplegar toda la información necesaria. Para nuestro caso se podrá observar continuamente el reloj en tiempo real, la programación de cada uno de los interruptores, entre otras opciones que podrán ser incluidas en caso de considerarse necesario. En la figura 2.3 se puede observar un diagrama que corresponde al circuito de salida al display.

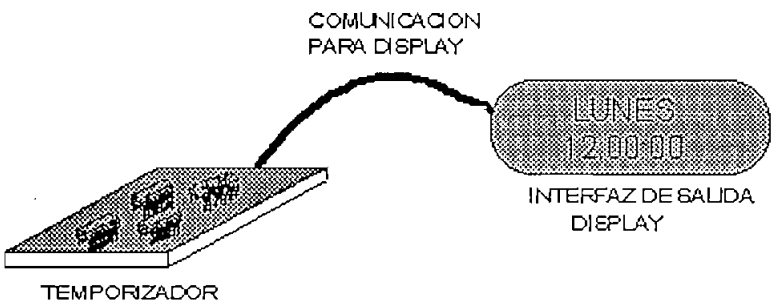


Fig 2.3 Circuito de salida al display

Otro circuito de salida es el interfaz para el control de los interruptores. En este caso se utilizará un circuito compuesto por optoacopladores, los cuales serán los

encargados de manejar los relés de salida.

2.3 MICROCONTROLADOR Y CIRCUITOS ADICIONALES A UTILIZARSE.

Una vez que existe una concepción del circuito se procederá a la explicación de los componentes que integran el temporizador.

2.3.1 MICROCONTROLADOR

Para la implantación del temporizador se utilizará, como ya se indicó en el capítulo I y ya que existe un conocimiento más profundo de la familia del MCS51, el microntrolador i8031. Posteriormente y para fines comerciales se podría realizar el mismo con el microcontrolador i8751, para ahorrarse circuitería y espacio físico.

Se eligió la utilización del i8031 debido a que existe la posibilidad de utilizar una memoria EPROM para realizar los programas de prueba. En el presente trabajo se utilizó una memoria NVRAM (RAM no volátil), ya que existe la posibilidad de programar la memoria mediante la ayuda de un computador personal, facilitando el desarrollo del

trabajo y evitándose la adquisición de un grabador y un borrador de EPROM.

2.3.2 MEMORIA

De acuerdo a las especificaciones indicadas, la unidad de memoria del temporizador programable está formada por dos partes: la primera para almacenamiento del programa monitor o de control y la segunda para el almacenamiento de los datos tanto de la programación de los interruptores así como de datos que se requieren para el normal funcionamiento de el programa de control.

Para el desarrollo del presente trabajo se utilizó memorias del tipo NVRAM tanto para el almacenamiento del programa de control como para el de los datos que determinan el funcionamiento de los interruptores.

2.3.2.1 CARACTERISTICAS DE LA NVRAM

La NVRAM que se utilizó es la MK48Z02B-15 del fabricante THOMPSON. Tiene las siguientes características:

- 11 años de vida en las peores condiciones a 70°C.
- Retención de datos en ausencia de energía.

- Seguridad en los datos por medio de protección automática de escritura durante pérdidas de energía.
- Compatibilidad en pines y funcionalidad con RAMs estáticas de 2K x 8 bytes y con EEPROMs.
- Lectura/Escritura sólo de +5V.
- Ciclos de escritura convencionales de SRAM.
- Consumo de 5.5 mW en standby.
- Tiempo de ciclo de lectura igual al tiempo de ciclo de escritura.
- Aviso de batería baja.

El MK48Z02/03/12/13 es una RAM no volátil estática, de 16384 bits, organizada en 2K x 8 utilizando tecnología CMOS y una fuente integral de energía de Litio. El consumo de energía para retener los datos es tan pequeña que una celda miniatura de Litio incorporada en el empaquetado provee de una fuente de energía para retener los datos. Es pin a pin compatible con la EPROM 2716 y con una EEPROM 2K x 8. No se necesita de circuitería adicional para conectar a un microprocesador.

El valor de Vcc se monitorea constantemente. Si decae el nivel de Vcc, la RAM automáticamente activará la deselección por falla de energía (power-fail deselect). Existe una ventana para el valor de V_{PFD} , el cual está

entre un máximo de 4.75 V y un mínimo de 4.5 V para la serie MK48Z02/03. Esto provee de seguridad bastante alta para los datos.

Tiene un 1% de probabilidad de falla de batería en 11 años de vida y un 50% en 12 años.

La memorias con las que se cuenta fueron fabricadas en 1994.

2.3.3 CONTROLADOR PARALELO DE ENTRADA / SALIDA

Para la activación del circuito de salida se hará uso de cuatro pines del pórtico P1 del microcontrolador. Para tener más control en el encendido y apagado de los relés se dispone de DIP SWITCH's, los cuales ingresan a una compuerta NAND en conjunto con la salida de los pines de P1. Con estos se podrá deshabilitar automáticamente una salida en caso de emergencia.

2.3.4 DISPLAY

El circuito de salida de la información consiste en un display de 16 caracteres por 2 líneas. El display consta de un controlador LSI HD44780.

2.3.4.1 INICIALIZACION POR CIRCUITO DE RESET INTERNO

Este se inicia automáticamente (resetea) cuando se energiza utilizando el circuito de reset interno. La bandera busy (BF) se la mantiene en un estado de ocupado antes de que termine la inicialización (BF=1); el estado de ocupado se da 10 ms después de que Vcc alcance 4.5V. Los siguientes pasos se cumplen durante la etapa de inicialización:

1) Borrar Display
2) Seteo de función
   DL = 0: interface de datos de 8 bits
   N = 0: 1 línea de display
   F = 0: caracteres de matriz 5x7
3) Control de display ON/OFF
   D = 0: display OFF
   C = 0: cursor OFF
   B = 0: titileo OFF
4) Seteo de modo de entrada
   I/D = 1: +1(incremento)
   S = 0: no desplazamiento

2.34.2 INICIALIZACION POR INSTRUCCIONES

Si no se consiguen las condiciones de alimentación para la correcta operación del circuito de reset interno, se requiere la inicialización por instrucciones. Como se utiliza el interface de 4 bits el procedimiento es el siguiente:

1) Encendido

2) Esperar más de 15 ms antes de que Vcc alcance 4.5V

3) No se puede chequear BF antes de esta intrucción

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	1	1

4) Esperar más de 4.1 ms.

5) No se puede chequear BF antes de esta intrucción

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	1	1

6) Esperar más de 100 us

7) No se puede chequear BF antes de esta intrucción

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	1	1

8)

RS	R/W	DB7	DB6	DB5	DB4	
0	0	0	0	1	0	I
0	0	0	0	1	0	
0	0	N	F	*	*	II
0	0	0	0	0	0	
0	0	1	0	0	0	III
0	0	0	0	0	0	
0	0	0	0	0	1	IV
0	0	0	0	0	0	
0	0	0	1	I/D	S	V

I. Setea el modo de trabajo a 4 bits. Esta instrucción indica al controlador del display aceptar y enviar datos en dos transferencias de 4 bits para todas las transferencias y transacciones posteriores. Esta es la única instrucción de 4 bits que reconoce el módulo.

II. Corresponde al seteo de función, configurando el modo de trabajo a 4 bits, 2 filas de trabajo (N = 1) y caracteres correspondientes a una matriz de 5 x 7 puntos.

Es importante anotar que, después de esta programación no se puede modificar el formato del display.

III. Se apaga el display con los datos retenidos en la RAM integrada en el circuito de display.

IV. La RAM de datos del display con el número 20H, el

contador de direcciones se pone en cero. El cursor vuelve la primera posición.

V. Seteo del modo de entrada, se incrementa la dirección de entrada (I/D = 1) y el display no se desplaza cuando se escribe un dato a la RAM generadora de caracteres.

Se pone en uno el valor de RS para mandar una dirección y en cero cuando se envia un dato.

2.4 CIRCUITO DE RELOJ DE TIEMPO REAL

El circuito de reloj en tiempo real que se utilizará es el MM58167 de la NATIONAL SEMICONDUCTOR. Es un circuito CMOS que funciona como reloj calendario en tiempo real en un sistema de bus orientado para microprocesadores. El dispositivo incluye un contador direccionable, retenedores direccionables para funciones de alarma, y 2 salidas de interrupciones. Una entrada de bajo consumo (power-down) permite al chip desconectarse del mundo exterior para una operación standby y así consumir menos energía. El oscilador que utiliza es de 32768 Hz.

2.4.1 CARACTERISTICAS DEL RELOJ

- Compatible con un microprocesador.
- Contadores de milésimas de segundos, centésimas de segundos, decenas de segundos, segundos, minutos, horas, día de la semana, día del mes y mes con sus correspondientes retenedores para funciones de alarma.
- Salida de interrupción mascarable con 8 posibles fuentes.
    - comparación entre retenedor y contador
    - cada decena de segundo
    - cada segundo
    - cada minuto
    - cada hora
    - cada día
    - cada semana
    - cada mes
- Modo de bajo consumo de energía que deshabilita todas las salidas excepto las salidas de interrupción la cual ocurre en una comparación entre contador y retenedor. No es lo mismo que una salida de interrupción mascarable.
- Estado no importa en los retenedores.
- Bit de status que indica desbordamiento durante una

lectura.

- Cristal de referencia de 32768 Hz, con sólo un capacitor de sintonización de entrada y capacitor externo de carga.
- Calendario de cuatro años.

2.4.2 DESCRIPCION FUNCIONAL

El circuito incluye contadores direccionables en tiempo real y retenedores direccionables, cada uno desde milésimas de segundo hasta meses. Tanto el contador como el retenedor están divididos en bytes de cuatro bits cada uno. Cuando se envía una dirección al bus de direcciones, 2 bytes aparecen en el bus de datos de E/S. Los datos, en BCD, pueden ser transferidos desde y hacia los contadores vía el bus de datos de E/S así cada set de 2 bytes (1 palabra) pueden ser accesados independientemente. Si un bit no es utilizado como es el caso de los días de la semana, en el ciclo de escritura no es reconocido este bit y en el ciclo de lectura se lo lleva a un nivel Vss la salida correspondiente. Si se introduce un dato no válido en el contador durante el ciclo de escritura, tomará 4 conteos (4 meses en el caso del contador de meses) para restaurar un dato BCD permitido al contador durante un conteo normal.

Los retenedores leerán y escribirán todos los 4 bits de cada byte. Cada una de las palabras del contador y del retenedor pueden ser inicializadas con las direcciones y los datos apropiados. La inicialización del contador es una función de escritura. Los retenedores pueden ser programados para compararse con los contadores todo el tiempo escribiendo un segundo en los dos bits más significativos de cada retenedor, lo que establece un estado no importa en el retenedor. Este estado de no importa es programado a nivel de byte.

Después de una lectura de un contador en tiempo real se da un bit de status. Si durante un ciclo de lectura el reloj se desborda, la lectura del dato puede ser inválida. Durante una lectura si el reloj se desborda, el bit de status se pone en alto. Este bit aparece en D0 cuando se lee, D1 a D7 serán ceros.

Existe un estado de GO para sincronizar el reloj con el tiempo real. Se inicializan las milésimas, centésimas, decenas y segundos. Después de igualar los demás datos, se puede mandar un pulso de lectura para resetear todos los contadores antes mencionados.

Existe un segundo comando especial el cual habilita la

salida de interrupción de standby. Es la única entrada o salida habilitada durante la condición de power-down o modo de standby. La condición de bajo consumo se da cuando la entrada de power-down se va a un nivel bajo. En este modo las salidas son tres estados y las entradas son ignoradas. La interrupción de standby se habilita escribiendo un 1 en la línea D0 seleccionada previamente en la dirección de interrupción standby.

La salida de interrupción está controlada por el registro de interrupción de status (8 bits) y el registro de interrupción de control (8 bits). La única vía para deshabilitar la salida de la interrupción es escribiendo ceros en el registro de control o habilitar la entrada de power-down.

El bus de E/S es controlado por las líneas de read, write, ready y chip select. Durante un ciclo de lectura (*RD = 0, *WR = 1, *CS = 0, RDY = 0) el dato en el bus de E/S es el dato contenido en el contador o retenedor direccionado. Durante un ciclo de escritura (*RD = 1, *WR = 0, *CS = 0, RDY = 0) el dato en el bus es retenido en el contador o retenedor direccionado.

Al inicio de cada ciclo de lectura o escritura la señal

de RDY se va a un nivel bajo y se mantiene en ese nivel hasta que el reloj haya puesto un dato válido en el bus o hasta que el haya completado de retener datos en una escritura. La línea de chip select es usada para habilitar o deshabilitar las salidas del dispositivo. Cuando el chip está seleccionado el dispositivo llevará el bus de E/S a una lectura o utilizará el bus de E/S como una entrada para una escritura. Cuando el chip no está seleccionado el bus no será afectado.

La base de tiempo del reloj es un oscilador controlado por un cristal de 32768 Hz. Internamente se cuenta con un inversor de alta ganancia, una red de retardo RC, y una resistencia de polarización. Para sintonizar el oscilador una lectura constante se puede dar en uno de los contadores de alta frecuencia. Por ejemplo, una lectura constante en el contador de milésimas de segundo pondrá una señal de 500 Hz en la línea de dirección D4 del bus. El período varía lentamente debido a la deshabilitación de los retenedores durante el desarrollo del conteo.

C A P I T U L O I I I

DISEÑO DEL TEMPORIZADOR PROGRAMABLE

El temporizador cuenta con 2 módulos principales. El primero es el interfaz gráfico con el usuario, el cual tiene menúes y ventanas para facilitar la programación y el segundo es el temporizador en si con el microcontrolador, el reloj, el display, los relés.

Además se utilizan los pórticos serial y paralelo del computador para descargar la información del interfaz y para programar la NVRAM respectivamente.

3.1 CONSIDERACIONES GENERALES DE DISEÑO

Todos los sistemas basados en microprocesadores o microcontroladores se diseñan en base al concepto de "bus". En el caso del i8031, se tiene tres buses principales:

- Bus de direccionamiento: 16 líneas
- Bus de datos bidireccional: 8 líneas
- Bus de control: 18 líneas

Para el caso de un microcontrolador y en especial el i8031 posee una RAM interna para almacenamiento de datos, pórticos paralelos y serial con los que se facilita el desarrollo circuital de una aplicación.

La ventaja de esta técnica de organización circuital la podemos resumir en:

- Admite incrementar las capacidades del sistema con otros elementos.
- La interconexión entre elementos es sencilla.
- Todos los elementos comparten la información del bus pero sólo uno de ellos puede recibir o enviar datos en un instante dado.
- La selección del elemento lo realiza la CPU a través del bus de direccionamiento y del bus de control.
- Uso de la tecnología tres estados que admite la posibilidad de "un tercer estado lógico", que permite compartir de las líneas de comunicación y por tanto de la información.

3.1.1 EL BUS DE DATOS

El bus de datos del microcontrolador i8031 es de 8

líneas, tres estados, bidireccional. Entre los elementos del sistema que son bidireccionales está la memoria RAM de almacenamiento de datos, es decir que puede ser fuente o destino de la información. El microcontrolador actúa como controlador del sistema; así la RAM es fuente, es decir si el microcontrolador desea leer información, por el bus de control envía una señal que habilite la memoria RAM (requerimiento de memoria) y además una señal de lectura RD, las salidas de la RAM aparecen entonces en el bus de datos de acuerdo a la posición dada por el bus de direccionamiento y el microcontrolador lee la información. Las mismas fases de este proceso toman lugar si el microcontrolador desea leer información de la memoria EPROM o de un puerto de entrada del controlador paralelo ya que son elementos fuentes de información del temporizador programable.

Si el microcontrolador desea almacenar o escribir un dato en la memoria RAM, en este caso la memoria esta siendo utilizada como destino de la información, el microcontrolador coloca el dato en el bus de datos, luego genera las señales de control necesarias, en este caso, el impulso para habilitar la memoria RAM, una señal de escritura en el dispositivo WR, la dirección

donde se va almacenar el dato está dada por el bus de direccionamiento y se carga el dato en esta posición. Las mismas operaciones se ejecutan si el dispositivo es un puerto de salida del controlador paralelo, con la diferencia que el impulso de requerimiento de memoria es reemplazado por un impulso de requerimiento de entrada / salida por ser elementos de destino de la información.

Se debe agregar, que la comunicación de datos entre las diferentes partes del temporizador se la realiza a través del microcontrolador. Esto es, si se desea transferir datos desde una puerta de entrada a memoria, el microcontrolador lee el dato de la puerta de entrada, lo almacena en una memoria intermedia, y después lo graba en la memoria. En conclusión, el bus de datos permite la transferencia de información en la presente aplicación y es compartido por todos los elementos del sistema.

3.1.2 EL BUS DE DIRECCIONES

En el literal 3.1.1 se analizó la forma en que comparten el bus de datos varios elementos del sistema, se trata en esta parte de estudiar una técnica que

permita seleccionar al microcontrolador el periférico adecuado en el instante que tiene que comunicarse a través del bus de datos. El bus de direcciones y el bus de control cumplen esta función.

La técnica que se ha escogido para el temporizador es la que se denomina PAGINAMIENTO DE MEMORIA en la cual cada posición de memoria o puerto de entrada / salida responde a una dirección en especial.

El bus de direcciones del i8031 está compuesto de 16 líneas denotadas desde A0 hasta A15 que permiten direccionar 65536 posiciones de memoria. Antes de realizar cualquier transferencia por el bus de datos, el microcontrolador debe proporcionar la dirección del elemento que debe enviar o recibir la información. Esto es, el microcontrolador selecciona algún elemento del temporizador de acuerdo a la instrucción que se encuentre con la ayuda de las señales de control.

3.1.3 EL BUS DE CONTROL

El proceso de seleccionar una posición específica y el manejo del bus de datos se coordina mediante el bus de control. El bus de control que posee el i8031 es de 2

líneas para la comunicación con la EPROM, 6 líneas para interrupciones externas (2 externas, 2 temporizadores, 2 seriales), 2 líneas para interactuar con RAM EXTERNA y 8 líneas adicionales correspondientes a un puerto de entrada/salida.

La ventaja de la utilización de un microcontrolador en lugar de un microprocesador es la disminución de circuitería y la versatilidad de la utilización de sus pórticos de entrada y salida.

Con el uso de un microcontrolador i8751 se ahorra aún más el cableado pero para el desarrollo de prototipos no es muy recomendable, su uso radicaría en sistemas definitivos.

3.1.4 DECODIFICADOR DE DIRECCIONES

Es un concepto importante dentro del presente trabajo y se lo define como un generador de señales que permite seleccionar a un elemento determinado, de acuerdo con el mapa de memoria establecido de los componentes del sistema cuando hay una dirección dada por el bus de direcciones.

Este concepto se ilustra en los diseños de los circuitos decodificadores de direcciones para memoria y entrada / salida.

Con el PAGINAMIENTO DE MEMORIA se trata a todos los elementos externos como RAM EXTERNA con lo cual su acceso se facilita.

3.2 DISEÑO UTILIZANDO EL MICROCONTROLADOR

El microcontrolador i8031 es el elemento del temporizador que controla toda la operación del sistema, constituye por lo tanto el cerebro de la aplicación y realizará las siguientes funciones:

- Provee las señales de control y tiempo para todos los elementos del temporizador.
- Ejecuta la fase de traída de datos e instrucciones de memoria.
- Decodifica las instrucciones
- Desarrolla las operaciones aritméticas y lógicas de acuerdo a la ejecución de una instrucción.
- Transfiere datos a/o de los dispositivos de entrada/salida.
- Responde a las señales de control generados por

equipos de entrada/salida.

- Genera las señales de control para lectura o escritura de datos en memoria o entrada/salida.

Se debe anotar, que la lógica interna del microcontrolador permite realizar estas funciones, y que, no se puede llegar a ésta externamente, sino mediante programas que darán las respectivas instrucciones para que funcione con las caracteristicas del temporizador programable. De aquí se desprende la gran ventaja del diseño utilizando microcontrolador, pues para cambiar de aplicación sólo se requiere modificar los programas o levemente su hardware.

3.2.1 DISEÑO DEL CIRCUITO DECODIFICADOR DE DIRECCIONES

Para accesar a los diversos dispositivos adicionales al microcontrolador se utiliza el integrado 74LS139 el cual es un decodificador dual 2 a 4. Uno de ellos se lo usa para los dispositivos de lectura y el otro para los de escritura. Como dispositivos de lectura se tiene la RAM EXTERNA y el RELOJ en tiempo real; y como de escritura el DISPLAY, la RAM EXTERNA y el RELOJ en tiempo real.

3.2.3 CIRCUITO DE RELOJ PARA EL MICROCONTROLADOR i8031.

Se utiliza como reloj en tiempo real el integrado MM58167AN con las características indicadas en el numeral 2.4. La dirección base de memoria es la 1000H. A partir de esta dirección se puede accesar a las demás opciones siguiendo las intrucciones del manual.

3.3 DISEÑO DE LOS CIRCUITOS DE INTERFAZ

Se describirá 4 interfaces:

1) Pórtico paralelo del computador para la grabación de la NVRAM.
2) Pórtico serial del computador para transferir la información al temporizador desde el PC.
3) Grabación de la NVRAM utilizando el pórtico paralelo y un circuito adicional.
4) A los relés para controlar las cargas de corriente alterna.

3.3.1 PORTICO PARALELO

El puerto paralelo también llamado puerto de la impresora, está diseñado de forma que permita la

conexión de impresoras paralelas, pero también es usado como un puerto de entrada/salida genérico. Este ofrece 3 registros que poseen las siguientes características:

dirección	No de bits	entrada	salida	lectura	escritura
378H	8		x	x	x
379H	5	x		x	
37AH	4	x	x	x	x

En la dirección 378H existe un registro de 8 bits, usado para la salida, pudiendo ser también leído. En la dirección 379H existe una entrada de 5 bits. En la dirección 37AH existe un registro de 4 bits, cuasibidireccional, el cual puede ser configurado para entrada o salida (colector abierto). Una de las entradas que están en la dirección 379H, pueden provocar una interrupción, cuya habilitación es controlada por software.

Todos los pines de entrada/salida están disponibles en un conector tipo D de 25 pines, en la parte trasera del computador (DB25).

El puerto paralelo normalmente está en la dirección 378H, pero también puede ser configurado para ocupar la dirección 278H. Existe una placa que ofrece un

adaptador de video monocromático y puerto paralelo, llamado "Monochrome Display and Printer Adapter" (MD&PA), en la cual el puerto paralelo se encuentra en la dirección 3BCH. La tarjeta MD&PA es muy rara, pero esa dirección sigue siendo usada en los PS de IBM y en las COMPAQ. Así cuando se trabaja en el puerto paralelo, se debe chequear la dirección de ese puerto. Se dice que en el 90% de los casos estaría en la 378H. La tabla siguiente muestra la correspondencia de las posibles direcciones. En este estudio será usada la dirección 378H.

BASE	ALTERNATIVO	MD&PA
378H	278H	3BCH
379H	279H	3BDH
37AH	27AH	3BEH

3.3.1.1 PUERTO DE SALIDA DE 8 BITS (378H)

El siguiente cuadro muestra como los registros están conectados a los pines del conector:

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
PINES	9	8	7	6	5	4	3	2

Una instrucción de salida (OUT) escribe directamente en

los pines del conector. Al escribir un bit, resulta un nivel TTL alto en la salida. Este registro también puede ser leído con una instrucción entrada (IN), esto permite verificar si los datos están siendo correctamente transferidos. Notar que esto no funciona como entrada, ya que lee la salida del 373.

Se puede colocar al pin *OE del 373 en Vcc, con eso el puerto se toma una entrada de 8 bits. Eso implicaría una modificación de la placa.

3.3.1.2 PUERTO DE ENTRADA DE 5 BITS (379H)

El siguiente cuadro indica como este registro está ligado a los pines del conector:

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
PINES	11(L)	10	12	13	15	-	-	-

(L) indica una inversión

Una lectura desde esta dirección con una instrucción IN, refleja el estado inmediato de estos pines. El pin 10 puede ser utilizado para provocar una interrupción IRQ7. Eso ocurrirá cuando hubiere una transición de bajo a alto en este pin. Es necesario que el bit 4 del

registro 37AH esté en 1.

3.3.1.3 PUERTO BIDIRECCIONAL DE 4 BITS (37AH)

La tabla siguiente indica como este registro está ligado a los pines del conector:

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
PINES	-	-	-	habIRQ7	17(L)	16(H)	14(L)	1(L)
RESET	-	-	-	0	1	0	1	1

(L) indica que el pin es invertido

El bit 4 es utilizado para controlar la habilitación del IRQ7. Cuando este bit es colocado en 1, la interrupción puede ocurrir. Estas salidas son accionadas por colector abierto. Esto les permite trabajar tanto como entrada o como salida. Ellas están conectadas a Vcc por resistores de 4.7 k (pull-up), pueden recibir hasta 7 mA y todavía mantener un nivel bajo de 0.8 V.

Para funcionar como salida, se debe usar la instrucción OUT, para escribir en los pines.

3.3.2 PORTICO SERIAL

Los canales del PC son totalmente programables. Un generador de baudios programables permite una tasa de operación de 50 a 9600 baudios. Se puede transmitir caracteres con 5, 6, 7 u 8 bits con 1, 1 1/2 y 2 bits de parada. El sistema priorizador de interrupciones controla las interrupciones de transmisión, recepción, error y línea de status. Exiten recursos para diagnóstico a través de funciones "loop back" para transmisión/recepción y para señales de entrada/salida.

El corazón de la interfaz serial es el CI 8250, o su equivalente funcional. Con este chip se consigue otras características adicionales:

- Bufferización doble que elimina la necesidad de una sincronización precisa.
- Entrada independiente para el reloj de recepción.
- Funciones para control de modem.

Los diferentes modos de operación son seleccionados a través de la programación del 8250. Este chip necesita 8 direcciones consecutivas. La interfaz COM1 tiene su 8250 ubicada en la dirección 3F8H y la COM2 usa la

dirección 2F8H.

Existen funciones del BIOS que se pueden usar para programar el 8250, pero se obtienen mejores ventajas cuando se controla directamente. En la siguiente tabla se muestra las direcciones de los registros:

COM2	COM1	REG.	NOMBRE DE LOS REGISTROS	DLAB
2F8	3F8	TxB	Búffer de transmisión	0 (Write)
2F8	3F8	RxB	Búffer de recepción	0 (Read)
2F8	3F8	DLL	Latch para divisor (LSB)	1
2F9	3F9	DLM	Latch para divisor (MSB)	1
2F9	3F9	IER	Reg. habilitador de interrupción	0
2FA	3FA	IIR	Reg. identificador de interrupciones	
2FB	3FB	LCR	Reg. controlador de líneas	
2FC	3FC	MCR	Reg. controlador de modem	
2FD	3FD	LSR	Reg. de status de línea	
2FE	3FE	MSR	Reg. de status de modem	

A continuación se describirá algunos de los registros anteriormente citados:

TxB (Búffer de transmisión)

En este registro se debe escribir el byte a ser transmitido por el 8250. Se accesa, con escritura, a

este registro cuando el bit DLAB = 0 (bit 7 de LCR)

RxB (Búffer de recepción)

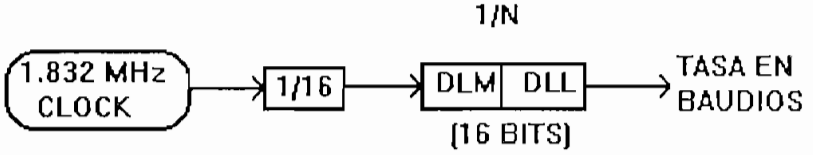
En este registro se puede leer el byte que llega por el canal serial. Al igual que el anterior se accesa, con lectura, cuando DLAB = 0 (bit 7 de LCR)

DLL (Latch Divisor para LSB)

Aquí se escribe el byte menos significativo del número (16 bits), por el cual se quiere dividir el reloj de entrada.

DLM (Latch Divisor para MSB)

Aquí se escribe el byte más significativo del número por el cual se desea dividir el clock de entrada. Con estos dos registros se programa la tasa de baudios (DLAB = 1). En la figura 3.1 se presenta un diagrama del divisor de frecuencia.



3.1 Diagrama de Bloques del divisor de frecuencia del pórtico serial

$$N = \frac{1843200}{16 * Baudios} = \frac{115200}{Baudios}$$

LCR (Registrador de Control de Línea)

Se especifica el formato de comunicación asincrónica.

A continuación se describen la función de cada bit:

D7	D6	D5	D4	D3	D2	D1	D0
DLAB	Set Break	Stick Parity	EPS	PEN	STB	WSL1	WSL0

DLAB:

Acceso al Divisor
0 - No acceso
1 - Acceso habilitado

EPS:

Paridad par 0 - deshabilita
1 - habilita

PEN:

Paridad 0 - deshabilita
1 - habilita

STB:

Bits de Parada

WSL1, WSL0:

00 5 bits
01 6 bits
10 7 bits
11 8 bits

LSR (Registro de Estado de Línea)

Este es un registro de 8 bits en el que se encuentra la información correspondiente a la transferencia de datos. De este registro son de interés dos bits:

D0 (DR)

Indica que fue recibido un dato, es decir, que existe un dato listo en el búffer. Este bit va a "1" siempre que se complete la recepción de un caracter y la transferencia para el búffer. Va a "0" cuando el procesador lee el dato recibido o cuando se escribe en esa dirección con el bit DR = 0.

D5 (THRE)

Indica que el registro de transmisión está vacío. El 8250 está listo para aceptar un nuevo caracter para transmitir. Va a "1" cuando el byte es transferido del registro de transmisión para el registro de desplazamiento. Este bit va a "0" cuando se escribe en el registro de transmisión.

3.3.3 INTERFAZ PARA GRABACION DE LA NVRAM

Debido a la dificultad que presenta el proceso de grabar y regrabar una memoria EPROM para su utilización en proyectos basados en microcontroladores, se ha diseñado un interfaz de grabación en una RAM no

volátil.

De entre las características de esta RAM y de acuerdo con los datos entregados por el fabricante, dispone de celdas de litio con una capacidad de respaldo de 11 años en ausencia de energía.

El diagrama de tiempos para poder realizar las operaciones de lectura y escritura en RAM son las que se muestran en las figuras 3.2 y 3.3

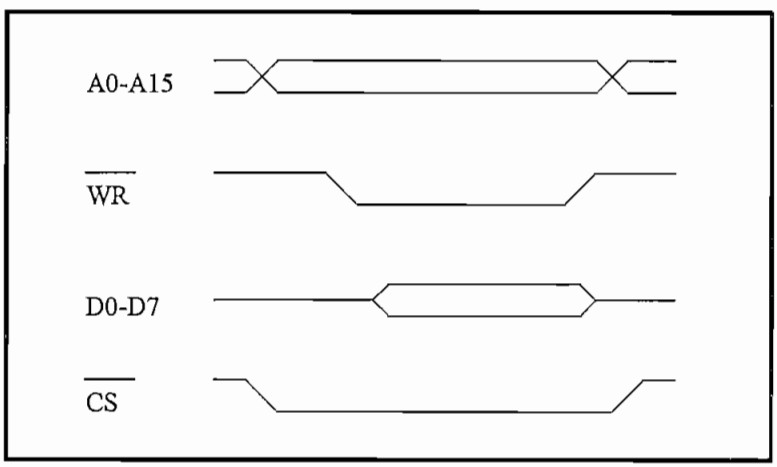


Fig 3.2 Ciclo de Escritura en RAM

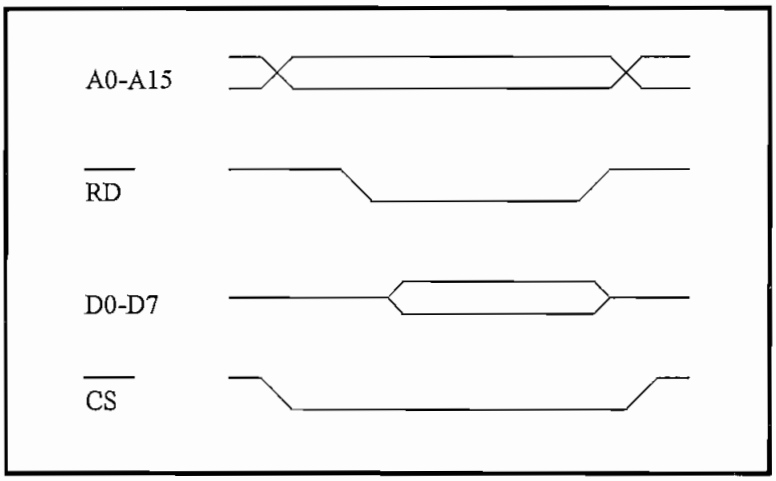


Fig 3.3 Ciclo de Lectura en RAM

Tanto las señales de datos como las de lectura y escritura serán manejadas por el puerto paralelo del computador que se utilice, para grabar datos y programar en el dispositivo de memoria del equipo; lo descrito puede verse en mayor detalle en el anexo 3 donde se presenta el circuito utilizado para el propósito señalado.

La dirección que se usará para el envio de datos es 0x378 (0x278 ó 0x3BC) que corresponde a los pines 2 al 9 del conector DB25. Este es un puerto exclusivamente de salida.

Las señales de habilitación se las genera en base al puerto cuya dirección es 0x37A (0x27A ó 0x3BD) que corresponde a los pines 1, 14, 16 y 17 del conector

DB25. Todos los pines a excepción del 16 son activados en bajo. La posición dentro del byte (enviado por el puerto) que ocupan los bits mencionados se muestra a continuación:

bit	7	6	5	4	3	2	1	0
pin	x	x	x	x	17	16	14	1

Debido a que se dispone de un bus de 8 bits y se debe accesar a direcciones de 11 bits, se requieren de 2 retenedores, mientras que el dato, siendo de 8 bits sólo require la presencia de un búffer.

En el anexo 3 se presenta el circuito para la grabación de la NVRAM, el mismo que cumple con la asignación de pines del puerto bidireccional en relación a las diferentes señales, como se muestra a continuación:

señal	B	*WE	G2	*OE	G1
pin	17	17	16	14	1

Siendo:

Señal	Descripción
B	Habilitación de todo búffer
*WE	Habilitación de escritura de la NVRAM
*OE	Habilitación de lectura de la NVRAM
G1	Habilitación del latch de dirección LSB (8 bits)
G2	Habilitación del latch de dirección MSB (3 bits)

A continuación se muestran las secuencias de instrucciones que se deben seguir para realizar las operaciones de escritura y lectura.

Escritura

B	*WE	*OE	G1	G2	Acción
1	1	1	0	0	Todo deshabilitado
1	1	1	1	0	Habilitación latch 1
LSB addr					Captura LSB addr
1	1	1	0	0	Todo deshabilitado
1	1	1	0	1	Hablitación latch 2
MSB addr					Captura MSB addr
1	1	1	0	0	Todo deshabilitado
0	0	1	0	0	Hablitación escritura
dato					
1	1	1	0	0	Todo deshabilitado

Lectura

B	*WE	*OE	G1	G2	Acción
1	1	1	0	0	Todo deshabilitado
1	1	1	1	0	Habilitación latch 1
LSB addr					Captura LSB addr
1	1	1	0	0	Todo deshabilitado
1	1	1	0	1	Hablitación latch 2
MSB addr					Captura MSB addr
1	1	1	0	0	Todo deshabilitado
1	1	0	0	0	Habilitación lectura

La lectura es necesaria para la comprobación de los datos grabados.

El programa desarrollado en lenguaje C para la grabación de las memorias NVRAM es el que se presenta en el anexo 1 del presente trabajo, al igual que el esquema del circuito utilizado.

3.3.4 INTERFAZ A LOS RELES

Se utiliza una compuerta NAND en la que ingresa el pin del pórtico del microcontrolador y una línea de control externa, formada por un dip switch y una resistencia. Con esto se puede desconectar la salida en casos de emergencia.

Hay que tener en cuenta que los relés se activan en bajo. A continuación, en la figura 3.4 consta un esquema del circuito.

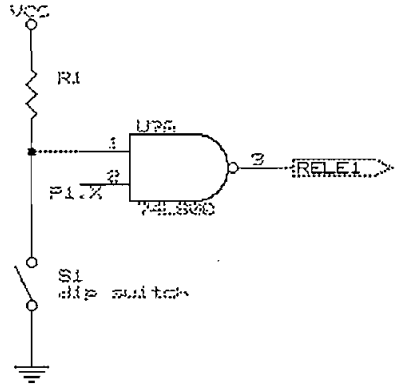


Figura 3.4 Circuito para controlar los relés

3.4 PROGRAMACION DE LA MEMORIA EPROM

Para el desarrollo del temporizador se utilizó en su lugar una memoria NVRAM con el interfaz descrito anteriormente

3.4.1.- TIPOS DE DATOS

De acuerdo a las caracteristicas analizadas en la sección 1.3 los tipos de datos que soporta el temporizador son:

a.- DESACTIVADO DIRECTO

b.- ACTIVACION A DETERMINADA HORA Y DESACTIVACION A OTRA HORA

c.- FUNCION 60 MINUTOS

d.- FUNCION PERIODICA

e.- EXCLUYENTES

3.4.2.- TIPOS DE DATOS DE CONTROL

a.- POSIBILIDAD DE VER LO PROGRAMADO PARA LOS INTERRUPTORES

b.- BORRAR LO PROGRAMADO PARA UN DIA

c.- BORRAR LO PROGRAMADO PARA UN INTERRUPTOR

d.- BORRAR LO PROGRAMADO PARA TODA LA SEMANA

3.4.3 TECNICA DE DISENO TOP-DOWN

Diseñar un programa es frecuentemente una tarea difícil, no hay un completo set de reglas que definan el diseño; por el contrario, el diseño del programa es un proceso creativo.

Una forma de enfrentar el diseño es establecer el algoritmo mediante la división de la una tarea en otras subtareas y estas subtareas en otras más pequeñas,

luego reemplazar estas pequeñas subtareas por subtareas muy pequeñas, etc. Generalmente, estas subtareas son tan pequeñas que son fáciles de ser implantadas en algún lenguaje de programación.

Este método es considerado como el más eficiente para diseño y sus principios serán utilizados en el diseño del programa de control del temporizador programable.

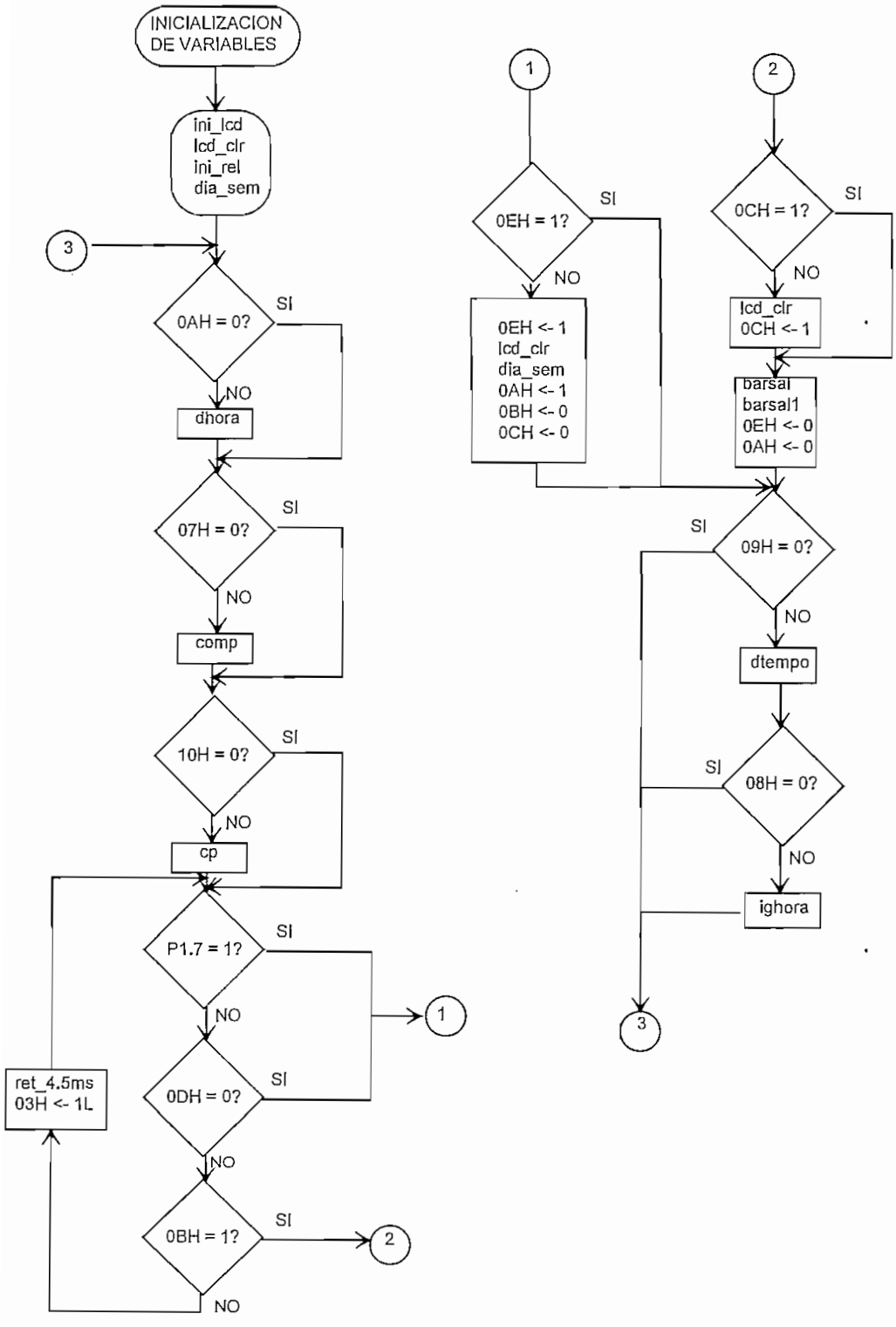
3.4.4 DIAGRAMAS DE FLUJO

Para la mejor comprensión del programa se presentan algunos diagramas de flujo.

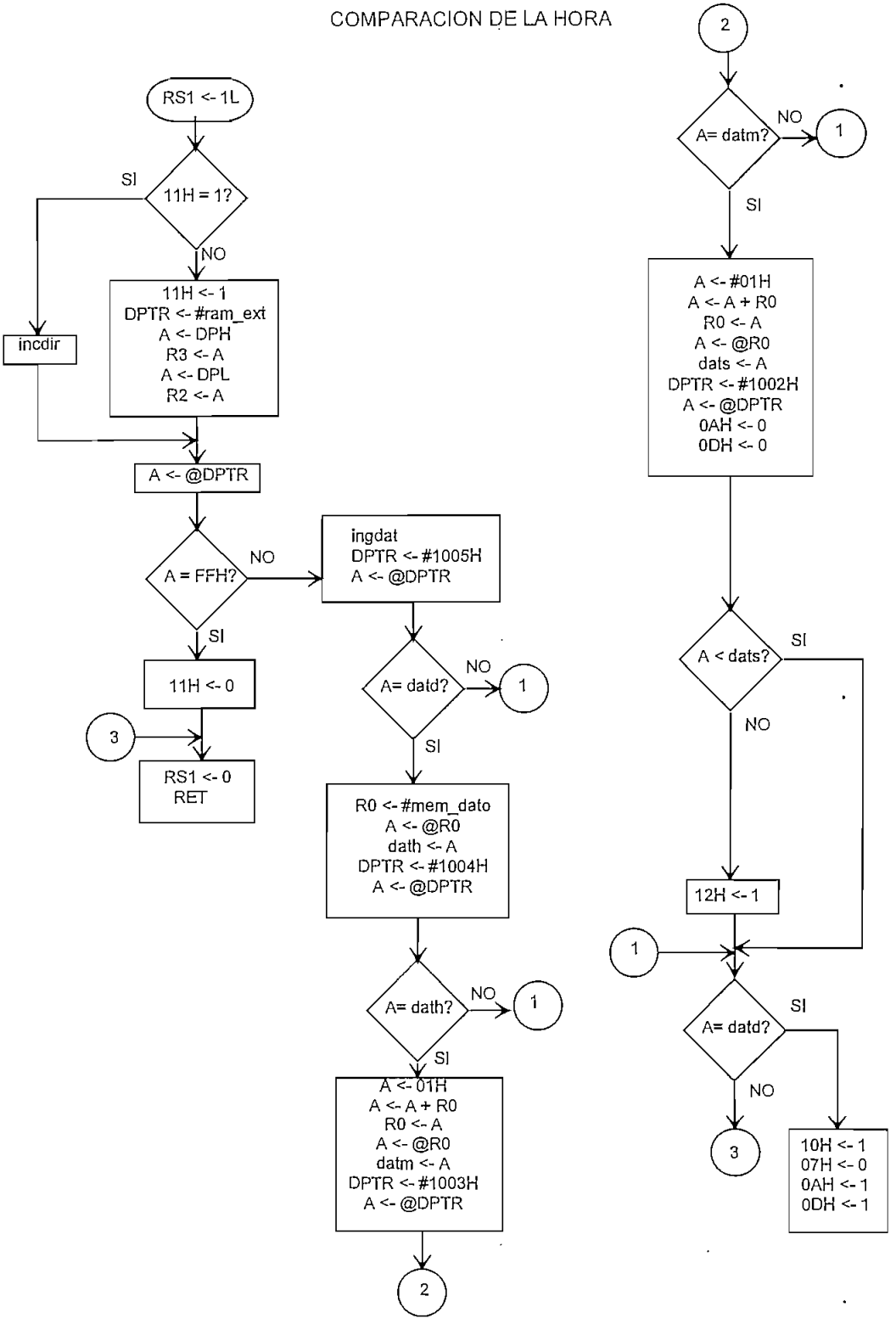
Existen algunas subrutinas que son muy pequeñas o simplemente son lineales y son fáciles de comprender.

No están especificadas todas las subrutinas pero, si la mayoría. En el anexo 1, se adjunta el listado del programa del microcontrolador con lo cual se puede analizar más detenidamente.

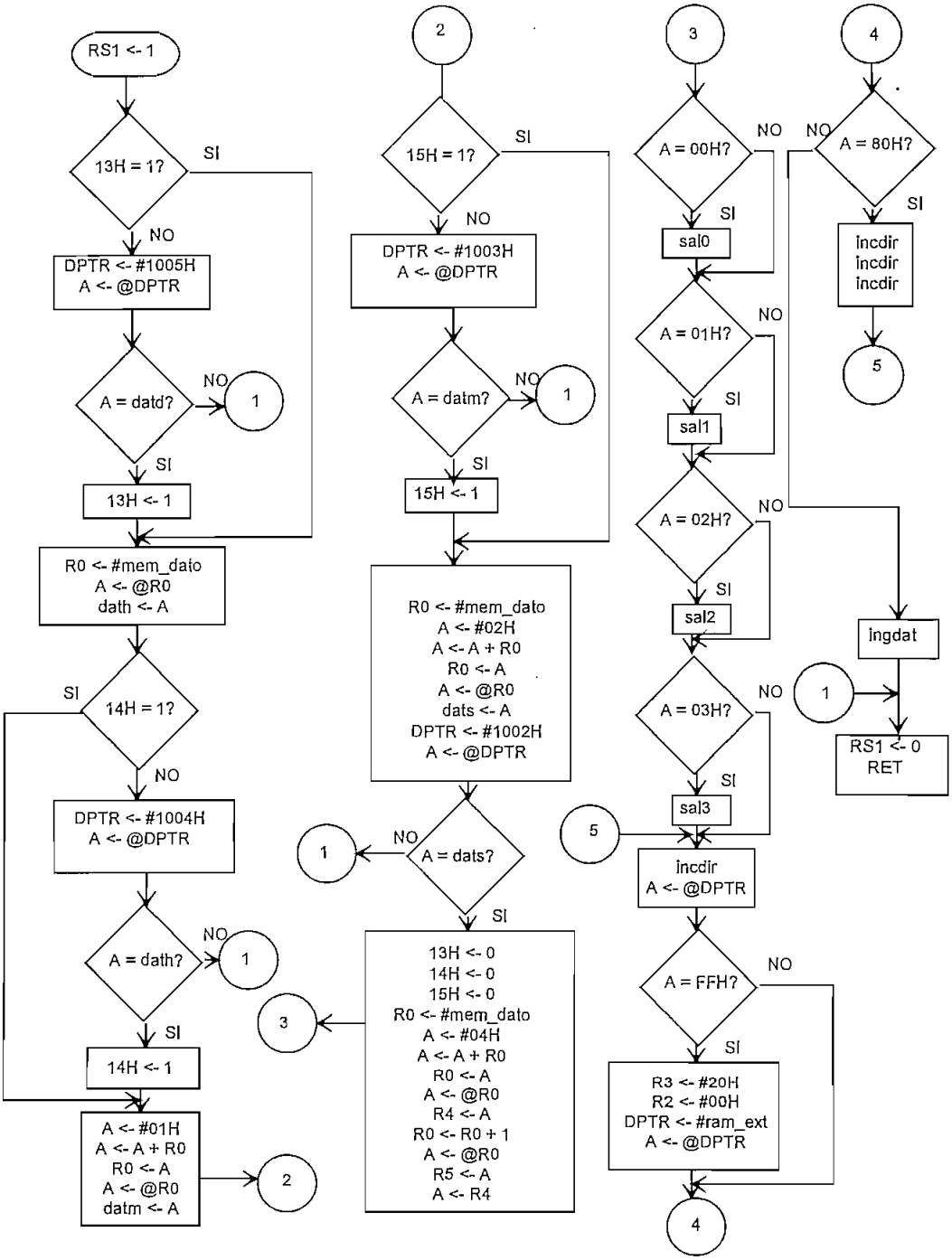
PROGRAMA PRINCIPAL



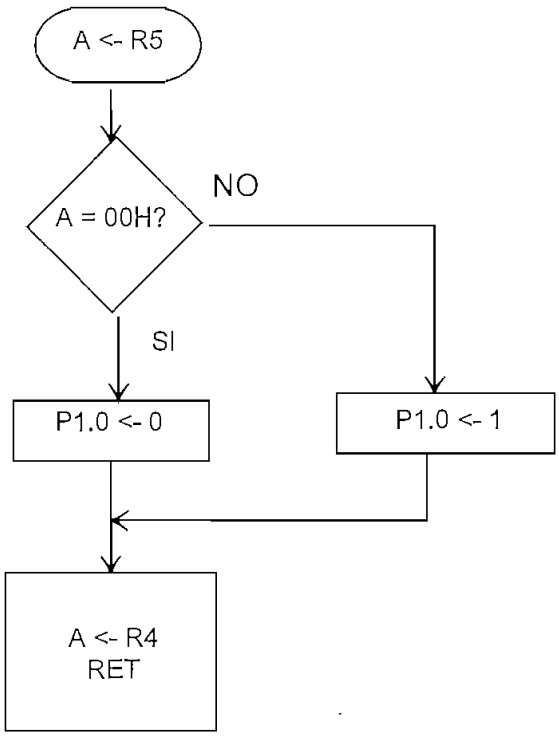
COMPARACION DE LA HORA



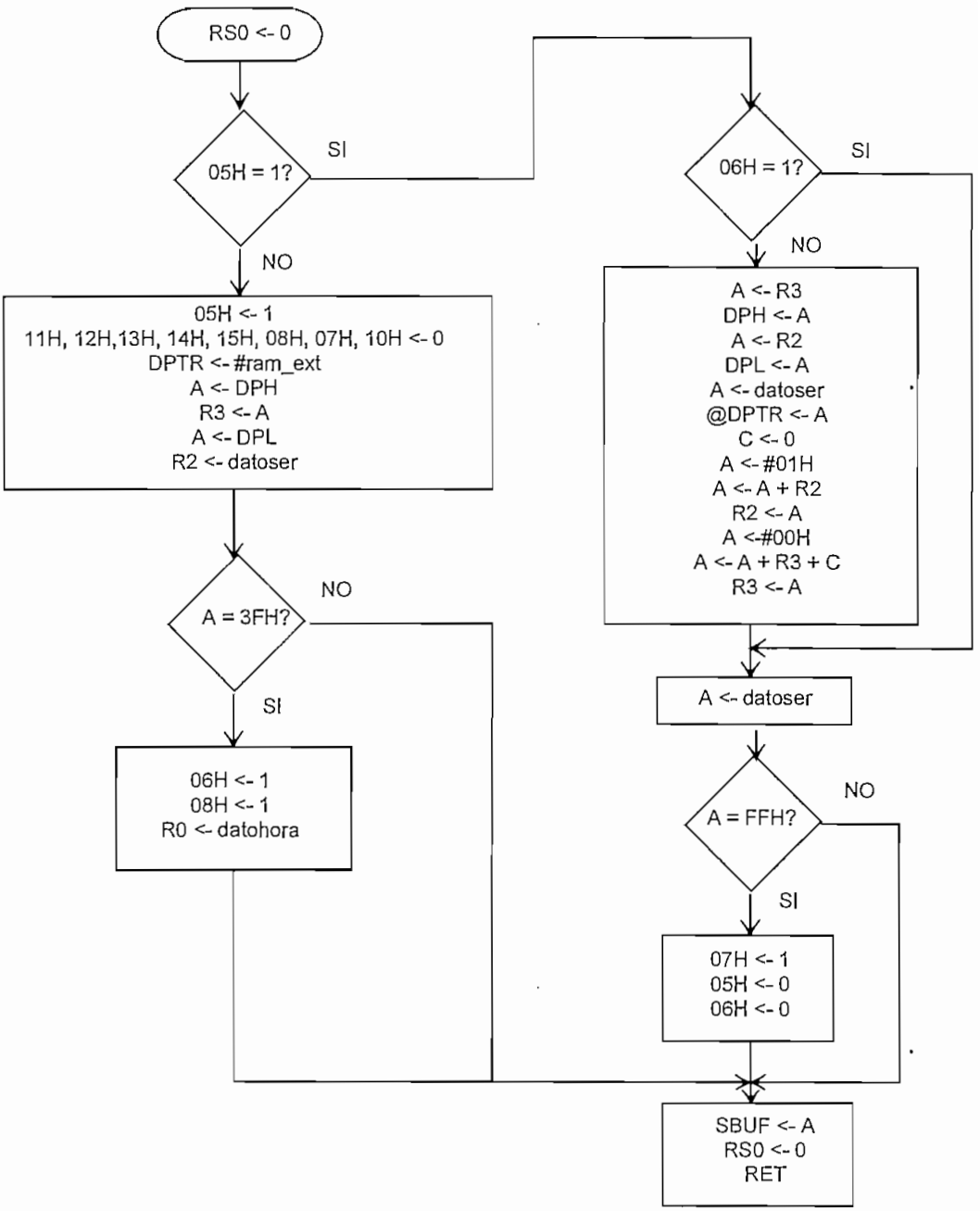
COMPARACION REGISTRO POR REGISTRO



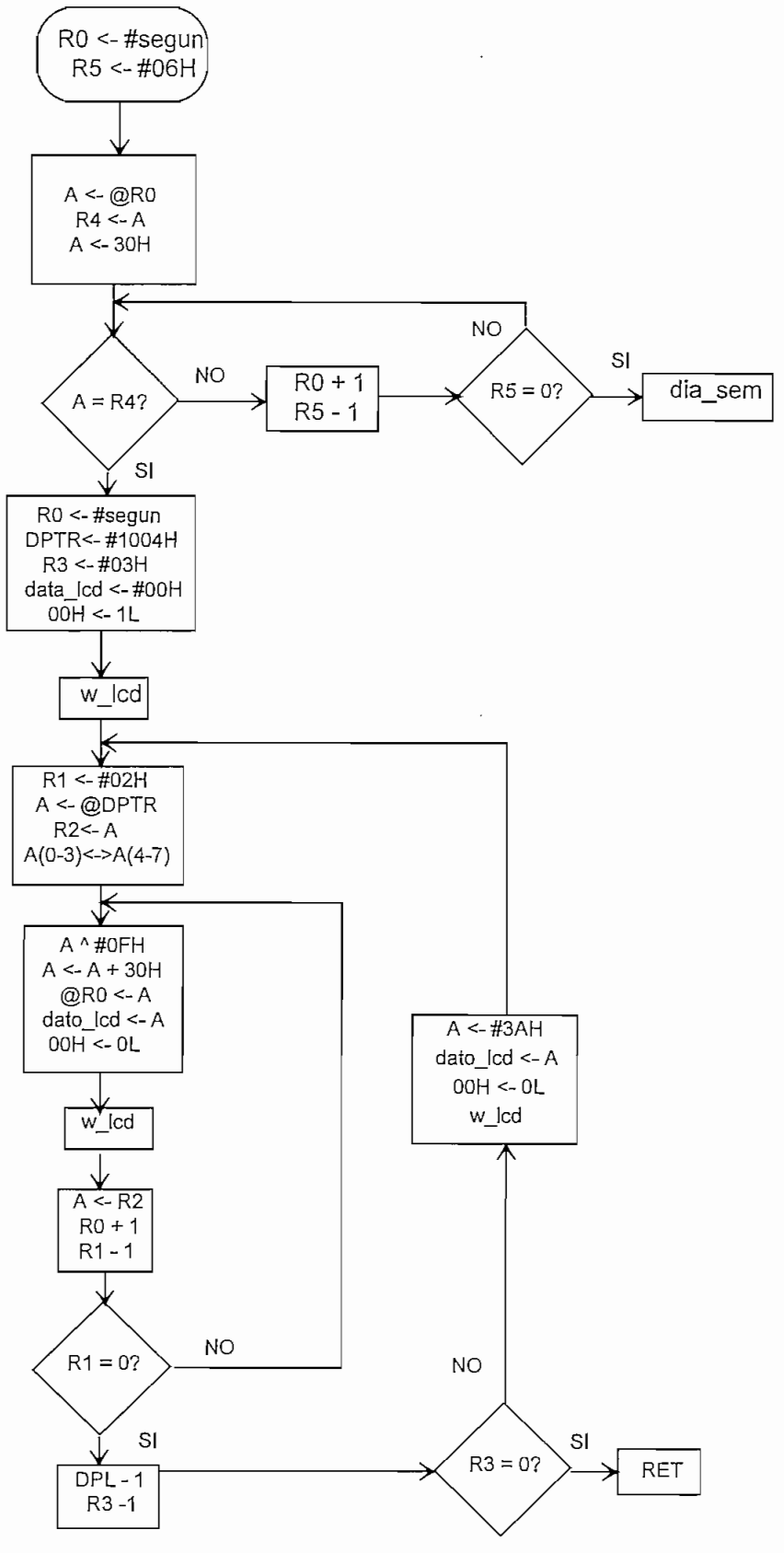
SALIDA 0 (IGUAL PARA SALIDA 1, SALIDA 2 Y SALIDA 3
SINO QUE AFECTA OTRO PIN DEL PORTICO)



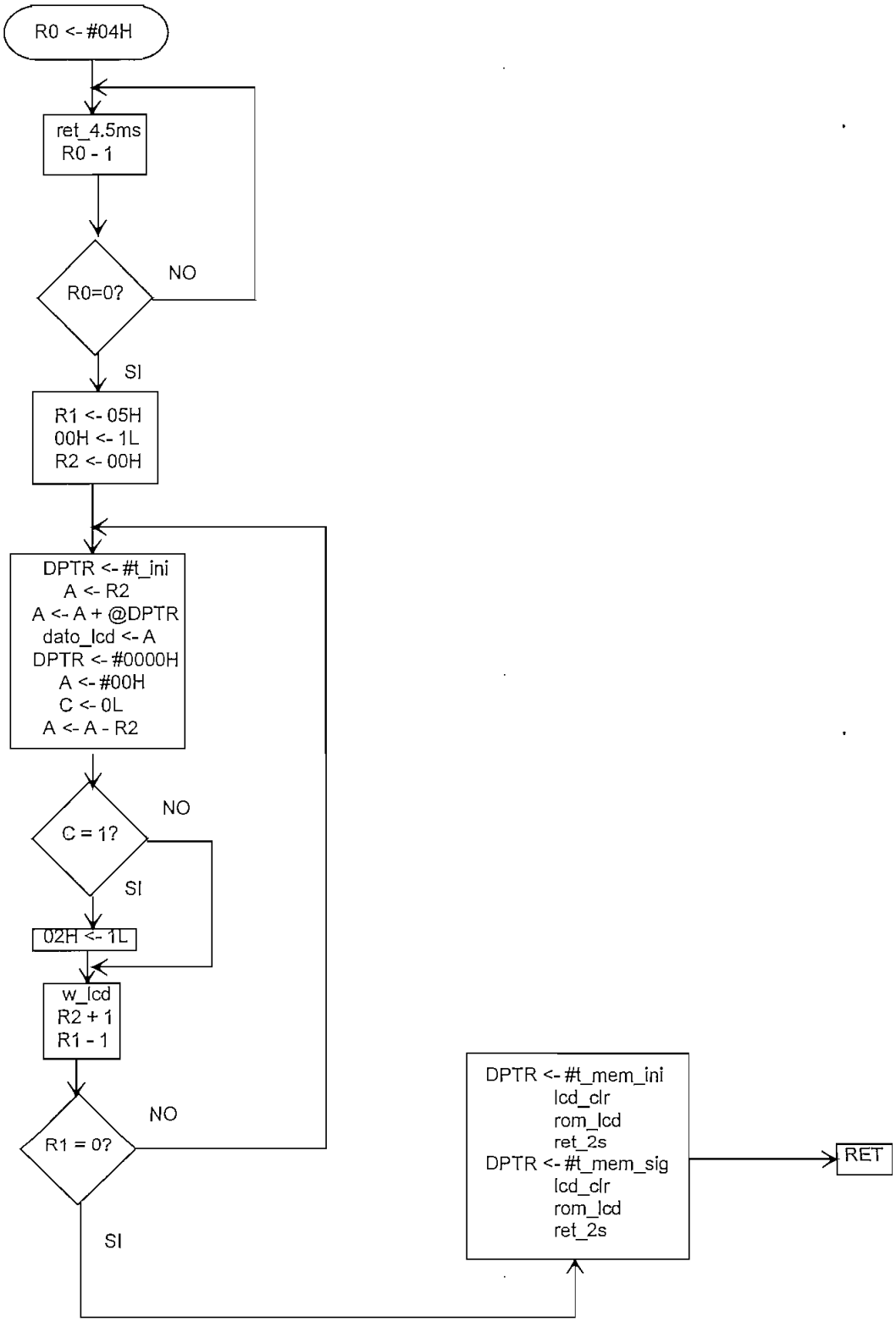
INGRESO DE DATOS LUEGO DE INTERRUPCION SERIAL



MOSTRAR HORA EN EL DISPLAY



INICIALIZACION DEL DISPLAY



BORRAR DISPLAY

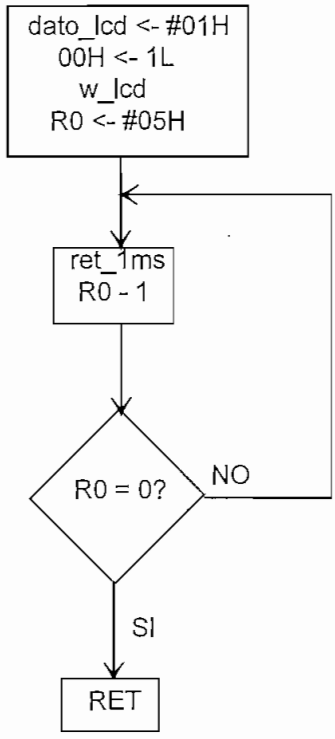
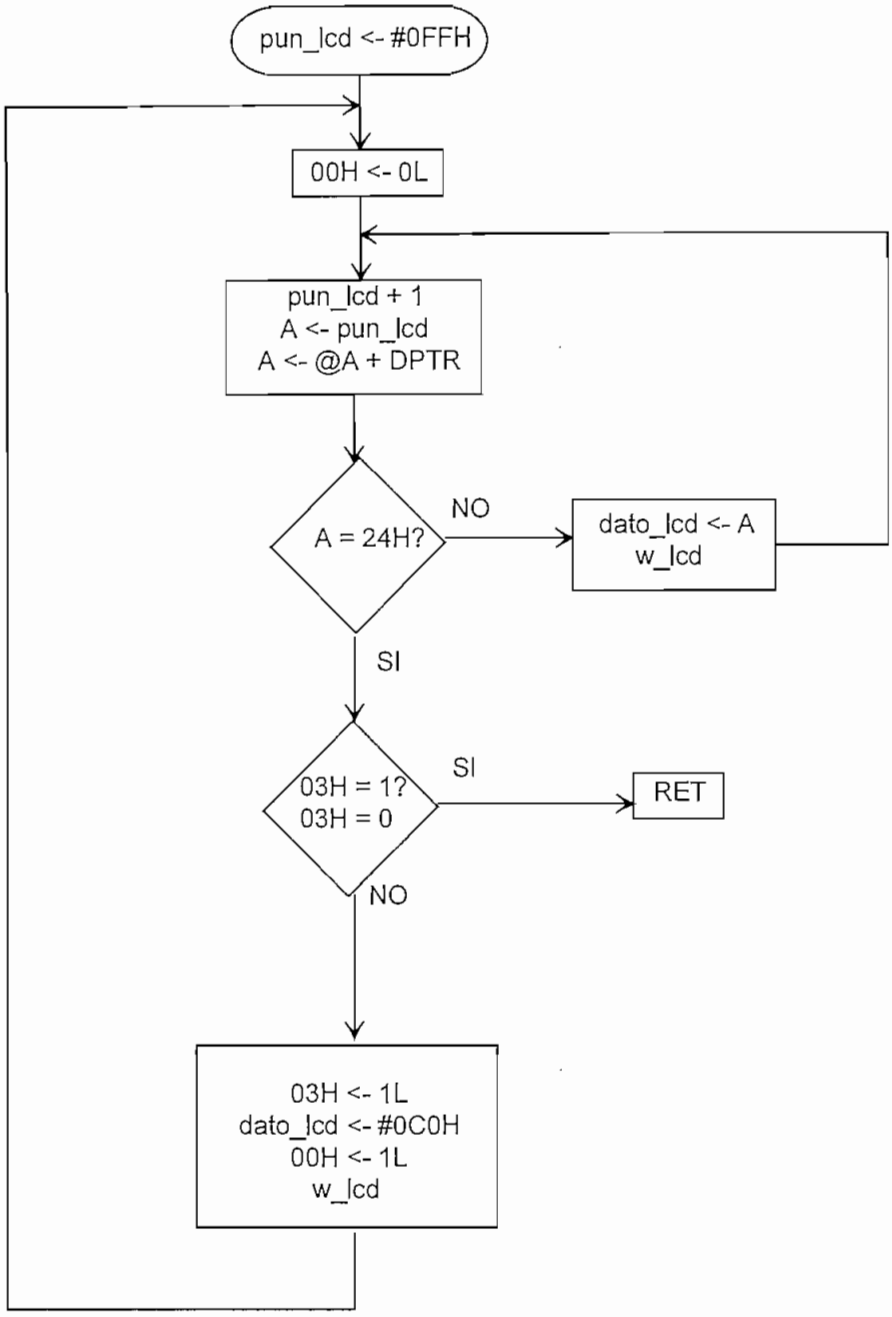
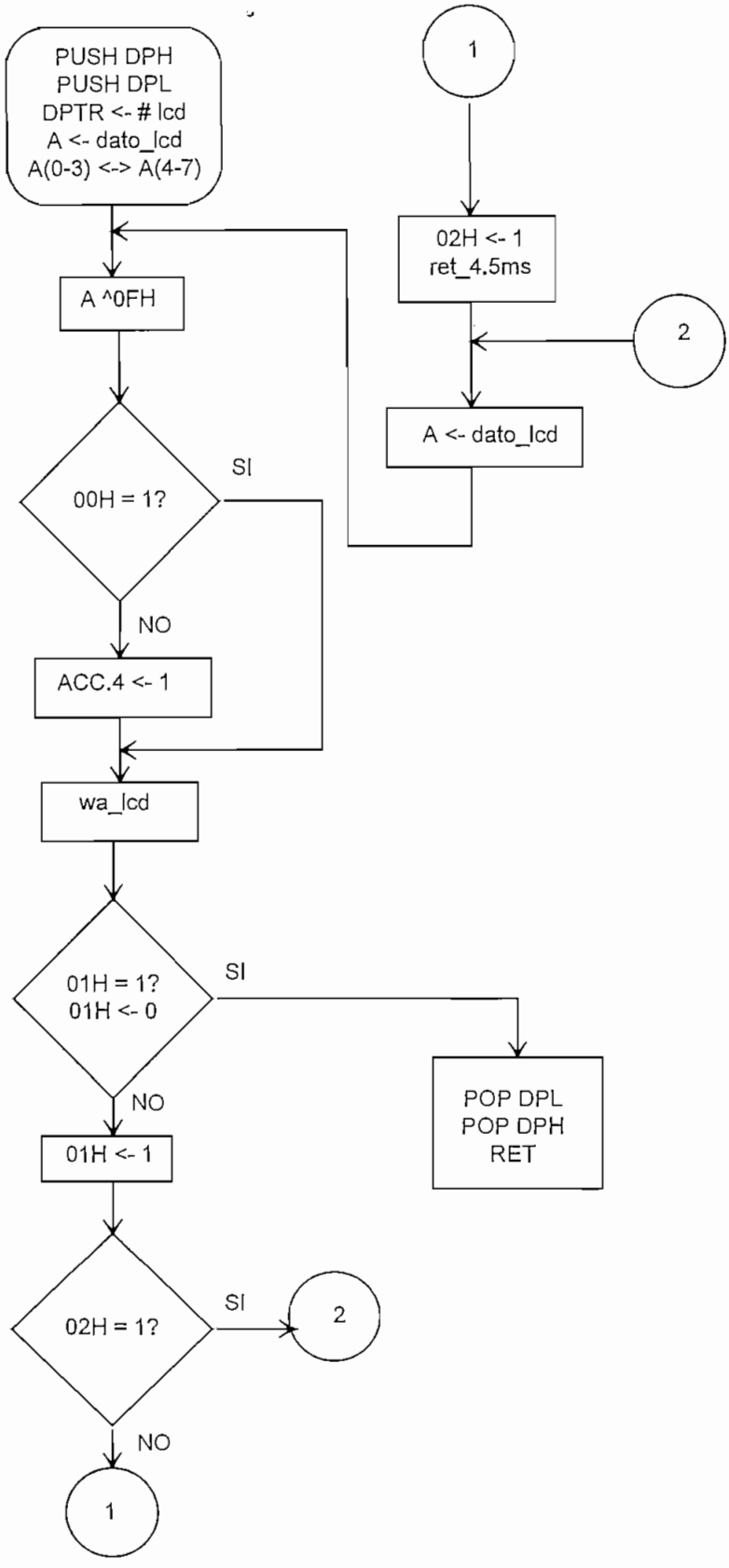


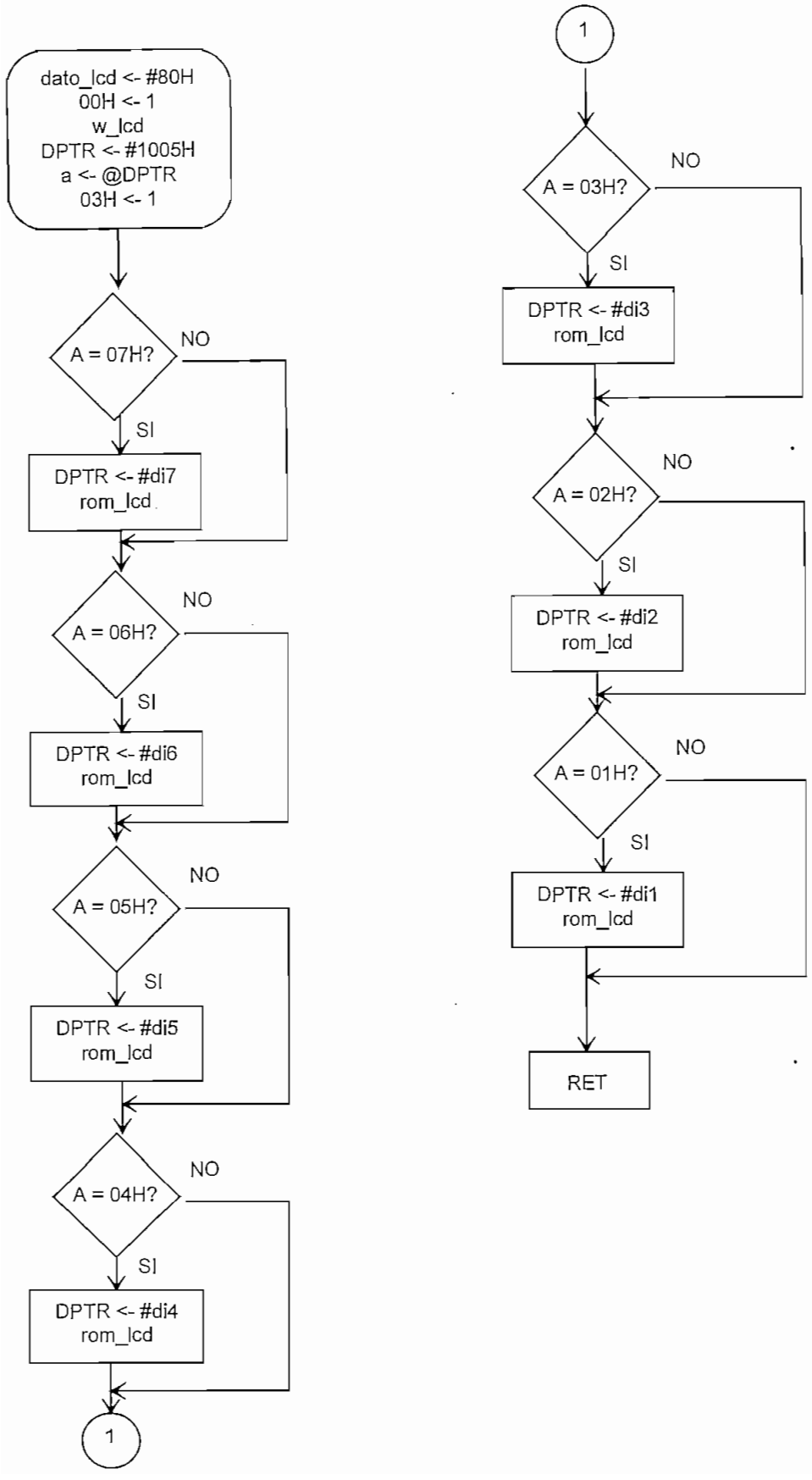
TABLA DE ROM


pun_lcd <- #0FFH
00H <- 0L
pun_lcd + 1
A <- pun_lcd
A <- @A + DPTR
A = 24H?
NO
dato_lcd <- A
w_lcd
SI
03H = 1?
03H = 0
SI
RET
NO
03H <- 1L
dato_lcd <- #0C0H
00H <- 1L
w_lcd

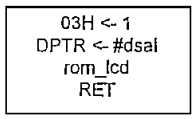
ESCRIBE EN DISPLAY CONTENIDO DE dato_lcd



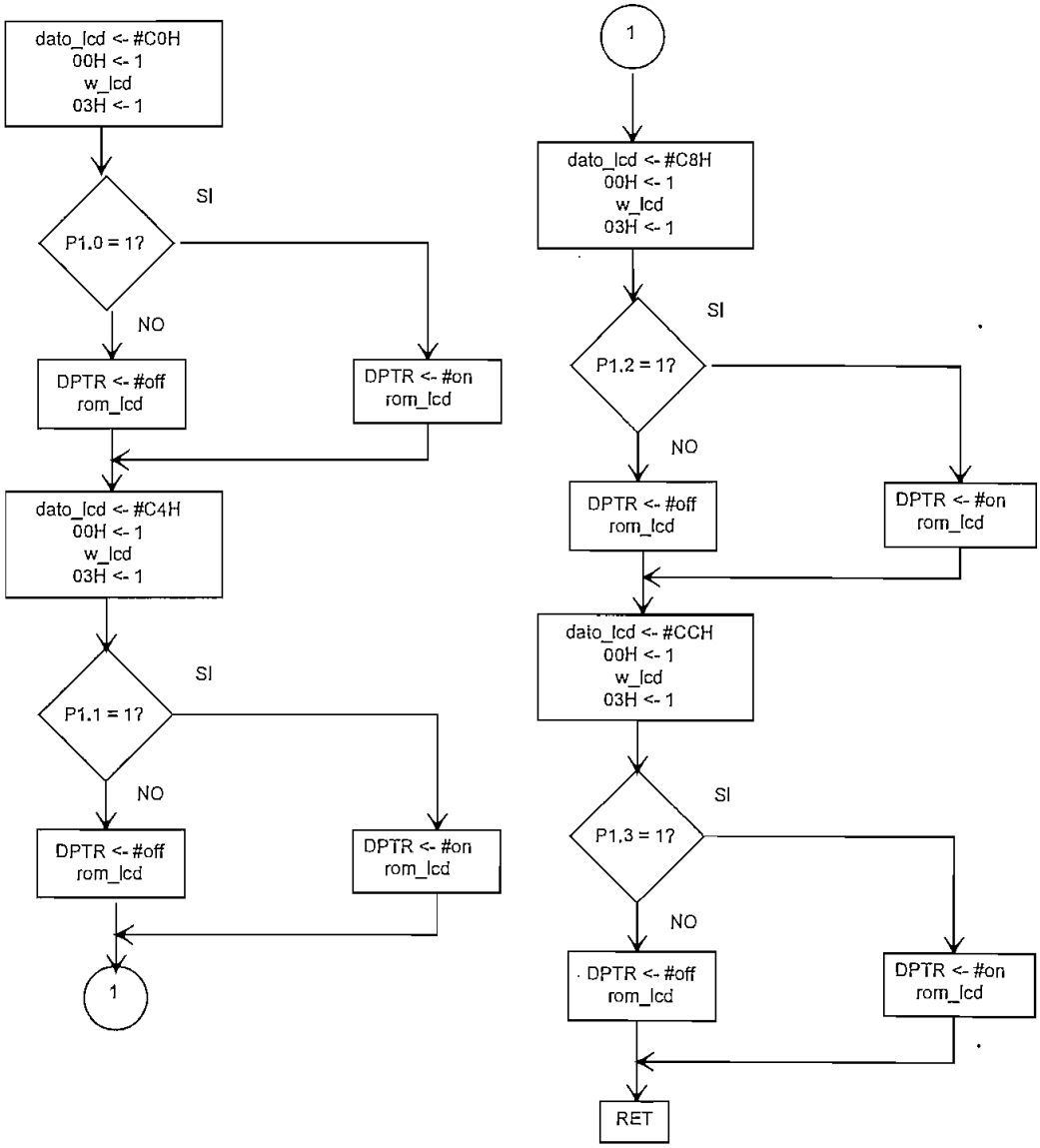
COMPROBACION DEL DIA DE LA SEMANA



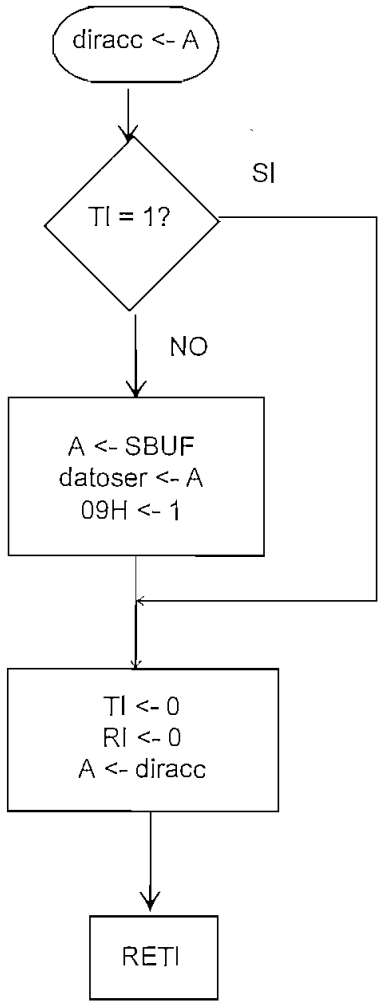
DESPLEGAR ESTADO DE SALIDAS (1era FILA)



DESPLEGAR ESTADO DE SALIDAS (2da FILA)



SUBRUTINA DE INTERRUPCION SERIAL



3.4.5 MAPA DE MEMORIA DE DATOS

El método utilizado es el de paginamiento, que como su nombre lo indica consiste en dividir la memoria externa disponible en páginas para controlar a los diversos elementos adicionales.

Las direcciones utilizadas son las que constan en la tabla siguiente.

DISPOSITIVO	DIRECCION
DISPLAY	0000H
RELOJ EN TIEMPO REAL	1000H
RAM EXTERNA	2000H

En lo referente a la memoria interna se tiene el siguiente mapa de memoria.

Variable	Dirección	Comentarios
STACK	30H	Cambio del Stack Pointer
datd	56H	Dato del día a ser comparado
dath	57H	Dato de la hora a se comparada
datm	58H	Dato de los minutos a ser comparados
dats	59H	Dato de los segundos a ser comparados
mem_dato	5AH	Dato Registro individual(hora)
	5BH	minutos

	5CH	segundos
	5DH	día
	5EH	salida
	5FH	estado
datohora	60H	dato para igualación reloj (día)
	61H	hora
	62H	minutos
	63H	segundos
datoser	70H	localidad para dato serial
pun_lcd	71H	puntero de localidad en ROM
dato_lcd	72H	dato para entrar en diaplay
diracc	73H	posición para guardar acumulador
segun	74H	actualización del reloj(decenas hora)
	75H	unidades hora
	76H	decenas minutos
	77H	unidades minutos
	78H	decenas segundos
	79H	unidades segundos

En cuanto a las banderas se tiene el siguiente mapa.

Bandera	Comentario
00H	0 = display despliega datos;1 = direcciones
01H	Control para inicialización del display
02H	Control para inicialización display
03H	Desplegar tabla de ROM
04H	Utilizada en Timer0
05H	Control en ingreso datos

06H	Diferenciación de datos(igualación de hora)
07H	Comparación inicial de registros
08H	Igualación de reloj
09H	Ingreso datos seriales
0AH	Control despliegue de hora
0BH	Comprobación de P1.7
0CH	Control paso de hora a barrido
0DH	Control despliegue de barrido de salidas
0EH	Control paso de barrido a hora
10H	Comparación registro por registro
11H	Control en comparación inicial
12H	Fin de comparación inicial
13H	Control comparación registros (día)
14H	Control comparación registros (hora)
15H	Control comparación registros (minutos)

La codificación que se realiza por registro es la siguiente:

b7	b6	b5	b4	b3	b2	b1	b0
decenas de hora				unidades de hora			
decenas de minuto				unidades de minuto			
decenas de segundo				unidades de segundo			
X	estado	salida			día		

Cada registro ocupa 4 localidades de memoria de 8 bits cada una, comenzando desde la localidad 2000H que corresponde a la RAM externa. Por ejemplo, las decenas

de hora tiene 4 bits al igual que las unidades de hora. El bit inicial de la cuarta localidad tiene un estado de no importa.

Cuando un registro no ha sido programado, se codifica con el primero de los cuatro bytes como 80H.

3.5 PROGRAMACION EN EL COMPUTADOR DE LA COMUNICACION SERIAL

En el numeral 3.3.2 se explicó el funcionamiento del pórtico serial del computador.

Se inicializa el pórtico ya sea con la dirección 3F8H ó con la dirección 2F8H.

Al registro LCR se lo inicializa con el número 80H para habilitar el acceso al divisor. Se introduce el valor de 000CH para setear la velocidad a 9600 baudios.

Luego se deshabilita el divisor y se habilita la comunicación serial con los siguientes parámetros:

- 8 bits de datos
- paridad deshabilitada

- 1 bit de parada.

Con el registro MCR se permite hacer comprobaciones, ya que se puede realizar un loop a nivel del 8250.

El registro LSR permite comprobar que se transmitan y se reciban los datos.

Los datos previamente son ordenados y luego codificados para ser transmitidos. La codificación se la puede observar en el numeral anterior.

La transmisión de la hora se la realiza directamente sin codificación. Estos datos se los almacena a partir de la dirección 60H (datohora).

Para la comprobación de la transmisión serial se compara el dato enviado del computador al microcontrolador con el dato enviado del microcontrolador al computador.

CAPITULO IV

RESULTADOS EXPERIMENTALES

4.1 FUNCIONAMIENTO DE LA PROGRAMACION

El sistema consta, de algunos componentes como el reloj en tiempo real, la memoria NVRAM de programa y la que almacena los datos y otros.

El análisis de los resultados obtenidos se puede resumir así:

El corazón del circuito constituye el microcontrolador.

En el programa residente se realizan las siguientes operaciones:

1) Inicialización de los parámetros como:
   - Timer 0, temporizador de 16 bits para realizar la función de crear retardos.
   - Timer 1, temporizador de 8 bits con autorecarga para utilizarlo en la generación de velocidad para el pórtico serial para que trabaje a la velocidad de 9600 baudios.
   - Habilitación de interrupciones: serial, timer 0, timer 1.
   - Inicialización de banderas usadas.

2) Inicialización del display, borrado del display.

3) Comprobación del día que será posteriormente desplegado en el display.

4) Se crea un lazo en el cual se observa constantemente la hora y el día actualizados.

Serialmente se puede variar la programación de los eventos y modificar la hora.

Una vez realizada cualesquiera de estas operaciones se procede a la comparación de los datos guardados con la hora real.

Cuando se logra igualdad en día, hora y minutos ocurre una comparación de segundos y se sitúa el puntero en el inmediato superior. Posteriormente se realiza a una comparación de registro por registro.

Al darse una igualdad total, el microcontrolador actúa sobre los relés por medio de su pórtico de salida P1, específicamente de P1.0 a P1.3.

5) Existe la posibilidad de realizar una inspección de las salidas mediante un dip switch conectado al pin P1.7.

Todo lo descrito anteriormente corresponde al programa que maneja el microcontrolador.

Este sistema posee además un interfaz gráfico para el ingreso de datos.

Con este interfaz gráfico se puede efectuar las siguientes operaciones:

1) Ingreso de los registros con un máximo de 224 correspondientes a 7 días de la semana, 4 salidas, 8 estados por día y por salida.

2) Modificación de cualesquiera de los registros.

3) Chequeo de cualquier registro ya sea individual o de todo el conjunto.

4) Ordenación de los registros de menor a mayor y posteriormente transmisión serial de los mismos al microcontrolador.

La ordenación se realiza de la siguiente manera:

1 corresponde a Domingo
2 corresponde a Lunes
3 corresponde a Martes
4 corresponde a Miércoles

5 corresponde a Jueves

6 corresponde a Viernes

7 corresponde a Sábado

Se crea un número de la siguiente forma:

dhhmmss

Estos números son ordenados cronológicamente de menor a mayor. Pueden existir registros en blanco los cuales se colocarán al inicio de la pila de comparación.

5) Para la hora existe la posibilidad de ingresor los datos, verificarlos y transmitirlos.

6) El interfaz cuenta con facilidades adicionales como manejo del mouse, minimización y maximización de ventanas, y de un sistema de menúes.

4.2 COMPROBACION DE CADA UNO DE LOS BLOQUES DEL SISTEMA

El sistema consta de los siguientes elementos:

1) El interfaz gráfico.

Los datos se los ingresa por registro. Cada registro tiene 6

campos con restricciones cada uno:

- día de la semana
- hora (0 - 23)
- minutos (0 - 59)
- segundos (0 - 59)
- salida (0 - 3)
- estado (0 - 1)

Los datos son ordenados en forma ascendente, codificados y enviados para almacenarse en la memoria NVRAM externa del microcontrolador.

2) El microcontrolador.

La información es descargada a través del pórtico serial del computador al pórtico serial del microcontrolador.

En la transmisión se discrimina mediante una comparación; si se trata de eventos del temporizador o si se va a igualar al reloj. Para la comunicación serial se utiliza un conversor de RS232 a TTL y viceversa.

El computador trabaja con el estándar RS232 y el microcontrolador con niveles TTL.

3) La memoria de programa.

Se utiliza un microcontrolador para trabajo con EPROM externa. En lugar de la EPROM se utiliza una NVRAM, la cual reemplaza eficazmente a la EPROM.

La programación de la NVRAM se la realiza por medio del pórtico paralelo del computador.

La explicación del circuito programador de NVRAM consta en el anexo correspondiente.

4) El reloj en tiempo real.

Se lo maneja como memoria externa. A partir de la dirección 1000H se puede accesar al reloj y a partir de esta dirección a cualquier función que permita el mismo.

5) La RAM externa.

Es una NVRAM con el fin de mantener los datos programados. La dirección en la que comienza la NVRAM es la 2000H.

6) El display.

Representa uno de los dos dispositivos de salida del

temporizador. En él se ve la hora y el estado actual de las salidas.

Tanto el cuarto, quinto y sexto elementos están controlados por un decodificador de direcciones conectado al bus de direcciones con lo cual se habilita cada uno de estos como si fueran memoria externa. El Paginamiento de Memoria funciona aplicando este principio.

7) El pórtico de salida con el interfaz a los relés.

Se dispone de un dip switch para disponer de un mejor control de las salidas en casos de emergencia.

Los relés están controlados por optoacopladores. Para activar los relés se debe presentar un estado bajo, correspondiente a un cero lógico. Los relés manejan cargas a 110 V. Para la comprobación se acoplan 4 focos que simulan a las cargas.

8) Un circuito de control del display.

Este circuito está compuesto por un dip-switch el cual ingresa al pin P1.7 y permite desplegar en el display el estado actual de las salidas.

Debido a la facilidad de grabar datos por medio del pórtico serial las pruebas para la comprobación del correcto funcionamiento se han simplifican notablemente.

Para la verificación del funcionamiento del circuito se cargaron datos correspondientes a diferentes eventos, es decir, conectar y desconectar las cargas, en este caso los focos.

Una vez realizada la programación se iguala la hora a un valor próximo a los eventos. Su comportamiento fue exitoso.

La limitación que presenta este temporizador radica en el algoritmo para localizar el primer evento, ya que realiza un barrido total y puede no encontrarlo si se coloca a una hora cercana (menor a 10 segundos), del primer evento.

4.3 RESULTADO CON EL SISTEMA TOTAL.

La comprobación de la operación del circuito resulta sencilla, ya que sólo se necesita realizar la programación para una sola salida, dos, tres o las cuatro; es decir, se puede variar la programación de eventos, así como la igualación del reloj.

Para la comprobación del sistema total se procedió a pruebas iniciales con pocos datos, para introducir paulatinamente una mayor cantidad de ellos. Al realizar las pruebas iniciales se

detectó la limitación del número de eventos a programarse; para ampliar su campo de acción se realizaron cambios en el software del microcontrolador.

Las rutinas de desplegar datos en el display son muy lentas, por lo que retrasan la comparación; por tal motivo existe un instante en el cual se procede a la comparación sin realizar refresco de pantalla.

Al incrementar la posibilidad de realizar un barrido en cualquier momento del estado de las salidas, se crea una subrutina bastante pesada ya que se despliega en tiempo real el estado de las mismas, lo que implica un refresco constante del display.

Para la comparación inicial se vería como una limitación si la hora del reloj en tiempo real está próximo a la efectivización del primer evento.

Para los sucesivos eventos las rutinas del display no afectan porque la comparación la realiza con un sólo registro.

4.4 CONCLUSIONES

El sistema se ha desarrollado con una concepción de bloques para que se facilite cualquier nueva implantación. Inicialmente puede controlar a cuatro salidas, pero se podría ampliar a muchas más.

Además, con el reloj en tiempo real se puede llevar un registro de períodos superiores a una semana.

La versatilidad de utilizar un microcontrolador permite disponer de mayores opciones con el mismo hardware o con aumentos no muy significativos.

Como es prototipo de laboratorio es factible introducir mejoras al diseño para poder comercializarlo en un futuro.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 ANALISIS ECONOMICO DEL SISTEMA

En si el corazón del circuito constituye el microcontrolador. Existen elementos adicionales los cuales son: un reloj en tiempo real, dos NVRAM (una para memoria del programa y otra para almacenamiento de datos), un display por el cual se despliega la información al usuario, un integrado que permite la comunicación entre el pórtico serial del computador y el pórtico serial del microcontrolador, un interfaz compuesto por optoacopladores y relés para manejar las cargas y elementos adicionales como resistencias, condensadores, dip switch's, compuertas, buffers. Cabe resaltar que la placa de interfaz para manejar las cargas, la cual está compuesta por los optoacopladores y los relés provino de una tesis anterior desarrollada por el Ingeniero José Puebla.

Para obtener un precio referencial del circuito implantado se desarrolla una descripción detallada de los materiales:

ITEM	DESCRIPCION	CANTIDAD	PRECIO UNITARIO	PRECIO TOTAL
1	Microcontrolador MCS-8031	1	3.25	3.25
2	Display de cristal líquido	1	25.00	25.00
3	Reloj en tiempo real	1	9.95	9.95
4	NVRAM MK48Z02B-15	2	9.95	19.90
5	MAX232	1	1.95	1.95
6	PC BOARD	1	5.00	5.00
7	Resistencias de 1/4 W	6	0.10	0.60
8	Capacitores electrolíticos	6	0.15	0.90
9	Capacitores cerámicos	4	0.10	0.40
10	Potenciómetro	1	0.25	0.25
11	Cristal de 7.3728 MHz	1	1.09	1.09
12	Cristal de 32 MHz	1	1.25	1.25
13	Botón de reset	1	0.39	0.39
14	Dip switch de 8 posiciones	1	0.89	0.89
15	74LS00	1	0.29	0.29
16	74LS04	1	0.29	0.29
17	74LS08	1	0.29	0.29
18	74LS139	1	0.39	0.39
19	74LS373	2	0.49	0.98
20	Cable plano	1	2.00	2.00
21	Zócalo de 14 pines	6	0.79	4.74
22	Zócalo de 16 pines	4	0.85	3.40
23	Zócalo de 20 pines	3	0.89	2.67
24	Zócalo de 24 pines	3	0.99	2.97
25	Zócalo de 40 pines	2	1.49	2.98
26	Zócalo desmontable de 24 pines	1	10.95	10.95
27	Conector de 14 posiciones de 2 filas	1	0.59	0.59
28	Conector de 34 posiciones doble fila	2	0.69	1.38
29	50 pines para conexión	1	1.19	1.19
30	Conector de 9 pines para cable plano	1	1.15	1.15
31	Fuente de poder	1	32.00	32.00
32	Rollo de cable de wire wrap	2	3.00	6.00
	TOTAL			145.08

5.2 CONCLUSIONES DEL ANALISIS ECONOMICO

Al valor de 145.08 dólares se le debería incrementar: un porcentaje por diseño, el costo del chasis o gabinete (que no se considera en el presente trabajo, ya que el alcance del circuito fue experimental de laboratorio) y otros costos directos e indirectos, que aparecen en una construcción en serie.

Los precios que son referenciales de los Estados Unidos se debería agregar las cantidades debidas a trámites de importación y flete, equivalentes a un 17 % adicional.

Sin embargo, si se realizaría una producción en serie los costos podrían reducirse. Considerando todos estos parámetros, si al precio obtenido se le agrega un 30 % con el propósito de obtener un valor final aproximado sería de 190 dólares. Este precio llama la atención para que en efecto se estudie en mayor detalle la posibilidad de su comercialización.

5.3 RECOMENDACIONES GENERALES PARA EL USO DEL SISTEMA

Por la magnitud de los datos a compararse se debe tener un margen de 10 segundos entre el evento a realizarse y la igualación del reloj en tiempo real.

Siempre se debe desconectar los dip switch que controlan las

salidas antes de conectar el temporizador, para evitar conexiones de las salidas y provocar efectos indeseados.

Este circuito que corresponde a un prototipo de laboratorio se ha construído mediante la técnica de wire wrap. Para el caso que se decida la construcción como prototipo de producción o para alguna aplicación en particular, se adjuntan los planos del circuito impreso en el anexo 5.

5.4 CONCLUSIONES FINALES

Este sistema constituye un temporizador versátil en cuanto a su programación y manejo.

El interfaz gráfico permite al usuario ingresar los datos con facilidad y de manera amigable.

La concepción del diseño permite repartir la carga entre el computador y el microcontrolador. Es más simple para el computador realizar ciertas operaciones ya que se trabaja con lenguaje de alto nivel, mientras que para el microcontrolador las operaciones pueden resultar más pesadas y requerirse de muchas intrucciones para realizarlas. Esta filosofía de programación se la aplica a diversos programas de computación con lo que se evita que se recarge sobre el host todo el peso del procesamiento, ya que ciertas operaciones se pueden realizar por

otros computadores más pequeños. Por este motivo se habla de la metodología down sizing.

La utilización de un microcontrolador permite tener un sistema modular en el cual se pueden introducir variaciones, sin afectar el hardware requerido o con pequeñas variaciones.

Otra facilidad del presente trabajo es el uso del grabador de NVRAM utilizando el pórtico paralelo del computador. Al respecto cabe destacar que en los computadores en los que se tuvo un poco de dificultad para utilizar el circuito son los COMPAQ. Sin embargo fue de gran utilidad en el desarrollo del prototipo, ya que se evita la utilización de memorias EPROM con sus respetivos grabadores y borradores.

En relación a la programación, es importante anotar que el lenguaje C presenta versatilidad en su uso con librerías preestablecidas, las mismas que permiten ahorrar parte del trabajo menos importante para conseguir el objetivo, en este caso de la tesis; el trabajo al que se hace referencia corresponde básicamente a la fachada que tiene el programa, es decir, el interfaz gráfico.

Se sugiere la enseñanza de lenguajes de programación más nuevos en la Facultad para así poder desarrollar trabajos mejores.

Con un temporizador se ahorra trabajos tediosos para los seres humanos, pudiendo dedicar sus esfuerzos a activadades más productivas en las que verdaderamente se requiera su presencia.

Otro punto importante es la posibilidad de la utilización de elementos de otras tesis que pueden servir como complemento y aliviar de cierta manera el trabajo, con lo cual se puede volcar los esfuerzos a tópicos diferentes.

ANEXOS

ANEXO 1. PROGRAMA EN EL MICROCONTROLADOR

ANEXO 2. PROGRAMA EN EL COMPUTADOR

ANEXO 3. PROGRAMA PARA GRABACION DE LA NVRAM Y DIAGRAMA DEL CIRCUITO PARA LA GRABACION

ANEXO 4. COPIA DE LOS MANUALES DE LOS ELEMENTOS UTILIZADOS

ANEXO 5. DIAGRAMAS EN ORCAD Y EN TANGO DEL TEMPORIZADOR

ANEXO 6. MANUAL DEL USUARIO

ANEXO 1

ANEXO 1

PROGRAMA DEL MICROCONTROLADOR

El programa se presenta a continuación corresponde al programa de control del equipo y se encuentra residente en la NVRAM que reemplaza a la EPROM.

```
;-------------------------------------------------
;       VARIABLES
;-------------------------------------------------

datd            EQU     56H
dath            EQU     57H
datm            EQU     58H
dats            EQU     59H
mem_dato        EQU     5AH
datohora        EQU     60H
datoser         EQU     70H
pun_lcd         EQU     71H
dato_lcd        EQU     72H
diracc          EQU     73H
segun           EQU     74H
stack           EQU     30H

;-------------------------------------------------
;       DIRECCIONES DE RAM EXTERNA
;-------------------------------------------------

lcd             EQU     0000H
reloj           EQU     1000H
ram_ext         EQU     2000H

;-------------------------------------------------
;       VECTORES DE INTERRUPCION
;-------------------------------------------------

        ORG     00H
        LJMP    30H

        ORG     0BH
        LJMP    INTT0

        ORG     23H
```

```
        LJMP    SERIAL

;-------------------------------------------------
;         INICIALIZACION DEL MICROCONTROLADOR
;-------------------------------------------------

        ORG     30H
INICIO: MOV     SP,#stack
        MOV     IE,#10010010B   ;serial habilitado, timer0
habilitado
        MOV     PCON,#00000000B ;
        MOV     TMOD,#00100001B ;timer0 16 bits, timer1 8 bits
recarga
        MOV     SCON,#01010000B ;
        MOV     IP,#00010000B   ;prioridad de la interrupcion
serial
        MOV     TH1,#0FEH
        MOV     TL1,#0FEH
        SETB    TR1
        MOV     20H,#00H
        MOV     21H,#00H
        MOV     22H,#00H
        SETB    07H
        SETB    0AH
        SETB    0DH
        SETB    0EH
        CLR     P1.0
        CLR     P1.1
        CLR     P1.2
        CLR     P1.3

;-------------------------------------------------
;         PROGRAMA
;-------------------------------------------------
        LCALL   ini_lcd
        LCALL   lcd_clr
        LCALL   ini_rel
        LCALL   dia_sem

lazo1:  JNB     0AH,lazo11
        LCALL   dhora
lazo11: JNB     07H,lazo2
        LCALL   comp
lazo2:  JNB     10H,lazo12
        LCALL   cp
lazo12: JB      P1.7,lazo14
        JNB     0DH,lazo14
        JB      0BH,lazo13
        LCALL   ret_4.5ms
        SETB    0BH
        SJMP    lazo12
```

```
lazo13: JB      0CH,lazo15
        LCALL   lcd_clr
        SETB    0CH
lazo15: LCALL   barsal
        LCALL   barsal1
        CLR     0EH
        CLR     0AH
        SJMP    lazo3
lazo14: JB      0EH,lazo3
        SETB    0EH
        LCALL   lcd_clr
        LCALL   dia_sem
        SETB    0AH
        CLR     0BH
        CLR     0CH
lazo3:  JNB     09H,lazo1
        CLR     09H
        LCALL   dtempo

        JNB     08H,lazo1
        LCALL   ighora

        SJMP    lazo1

;-----------------------------------------------
;       COMPARACION DE LA HORA
;-----------------------------------------------
comp:   SETB    RS1

        JB      11H,comp1
        SETB    11H
```

```
        MOV     DPTR,#ram_ext
        MOV     A,DPH
        MOV     R3,A
        MOV     A,DPL
        MOV     R2,A
        SJMP    comp2

comp1:  LCALL   incdir

comp2:  MOVX    A,@DPTR
        CJNE    A,#0FFH,comp3
        CLR     11H
        LJMP    fincomp

comp3:  LCALL   ingdat

        MOV     DPTR,#1005H
        MOVX    A,@DPTR
        CJNE    A,datd,comp4
        MOV     R0,#mem_dato
        MOV     A,@R0
        MOV     dath,A
        MOV     DPTR,#1004H
        MOVX    A,@DPTR
        CJNE    A,dath,comp4
        MOV     A,#01H
        ADD     A,R0
        MOV     R0,A
        MOV     A,@R0
        MOV     datm,A
        MOV     DPTR,#1003H
        MOVX    A,@DPTR
        CJNE    A,datm,comp4
        MOV     A,#01H
        ADD     A,R0
        MOV     R0,A
        MOV     A,@R0
        MOV     dats,A
        MOV     DPTR,#1002H
        MOVX    A,@DPTR

        CLR     0AH
        CLR     0DH

        CJNE    A,dats,comp41
comp41: JC      comp4
        SETB    12H

comp4:  JB      12H,comp5
        LJMP    fincomp
```

```
comp5:  SETB    10H
        CLR     07H
        SETB    0AH
        SETB    0DH

fincomp:CLR     RS1
        RET

;-------------------------------------------------
;   COMPARACION DE LA HORA REGISTRO POR REGISTRO
;-------------------------------------------------
cp:     SETB    RS1

        JB      13H,cp1
        MOV     DPTR,#1005H
        MOVX    A,@DPTR
        CJNE    A,datd,cp0
        SETB    13H
        SJMP    cp1

cp0:    LJMP    fincp

cp1:    MOV     R0,#mem_dato
        MOV     A,@R0
        MOV     dath,A

        JB      14H,cp2
        MOV     DPTR,#1004H
        MOVX    A,@DPTR
        CJNE    A,dath,fincp
        SETB    14H

cp2:    MOV     A,#01H
        ADD     A,R0
        MOV     R0,A
        MOV     A,@R0
        MOV     datm,A

        JB      15H,cp3
        MOV     DPTR,#1003H
        MOVX    A,@DPTR
        CJNE    A,datm,fincp
        SETB    15H

cp3:    MOV     R0,#mem_dato
        MOV     A,#02H
        ADD     A,R0
        MOV     R0,A
        MOV     A,@R0
        MOV     dats,A
```

```
        MOV     DPTR,#1002H
        MOVX    A,@DPTR
        CJNE    A,dats,fincp

        CLR     13H
        CLR     14H
        CLR     15H
        MOV     R0,#mem_dato
        MOV     A,#04H
        ADD     A,R0
        MOV     R0,A
        MOV     A,@R0
        MOV     R4,A

        INC     R0
        MOV     A,@R0
        MOV     R5,A
        MOV     A,R4

salida0:
        CJNE    A,#00H,salida1
        LCALL   sal0
salida1:
        CJNE    A,#01H,salida2
        LCALL   sal1
salida2:
        CJNE    A,#02H,salida3
        LCALL   sal2
salida3:
        CJNE    A,#03H,cp4
        LCALL   sal3

cp4:    LCALL   incdir
        MOVX    A,@DPTR
        CJNE    A,#0FFH,cp5
        MOV     R3,#20H
        MOV     R2,#00H
        MOV     DPTR,#ram_ext
        MOVX     A,@DPTR
cp5:    CJNE    A,#80H,cp6
        LCALL   incdir
        LCALL   incdir
        LCALL   incdir
        SJMP    cp4
cp6:    LCALL   ingdat
fincp:  CLR     RS1
        RET

;-------------------------------------------------
;       INGRESO DATOS PARA COMPARACION
;-------------------------------------------------
```

```
ingdat: MOV     R0,#mem_dato
        MOV     @R0,A           ;HORA
        LCALL   incdir
        INC     R0
        MOVX    A,@DPTR
        MOV     @R0,A           ;MINUTOS
        LCALL   incdir
        INC     R0
        MOVX    A,@DPTR
        MOV     @R0,A           ;SEGUNDOS
        LCALL   incdir
        INC     R0
        MOVX    A,@DPTR
        MOV     R4,A
        ANL     A,#00000111B
        MOV     @R0,A           ;DIA
        MOV     A,R4
        ANL     A,#00111000B
        RR      A
        RR      A
        RR      A
        INC     R0
        MOV     @R0,A           ;SALIDA
        MOV     A,R4
        ANL     A,#01000000B
        RL      A
        RL      A
        INC     R0
        MOV     @R0,A           ;ESTADO

        CLR     C
        MOV     A,#03H
        MOV     R0,#mem_dato
        ADD     A,R0
        MOV     R0,A
        MOV     A,@R0
        MOV     datd,A

        RET

;-------------------------------------------------
;       SALIDA 0
;-------------------------------------------------
sal0:   MOV     A,R5
        CJNE    A,#00H,sal01
        CLR     P1.0
        SJMP    sal02
sal01:  SETB    P1.0
sal02:  MOV     A,R4
        RET
```

```
;-------------------------------------------------
;       SALIDA 1
;-------------------------------------------------
sal1:   MOV     A,R5
        CJNE    A,#00H,sal11
        CLR     P1.1
        SJMP    sal12
sal11:  SETB    P1.1
sal12:  MOV     A,R4
        RET

;-------------------------------------------------
;       SALIDA 2
;-------------------------------------------------
sal2:   MOV     A,R5
        CJNE    A,#00H,sal21
        CLR     P1.2
        SJMP    sal22
sal21:  SETB    P1.2
sal22:  MOV     A,R4
        RET

;-------------------------------------------------
;       SALIDA 3
;-------------------------------------------------
sal3:   MOV     A,R5
        CJNE    A,#00H,sal31
        CLR     P1.3
        SJMP    sal32
sal31:  SETB    P1.3
sal32:  MOV     A,R4
        RET

;-------------------------------------------------
;       INCREMENTAR DIRECCION
;-------------------------------------------------
incdir: MOV     DPL,R2
        MOV     DPH,R3
        INC     DPTR
        MOV     R2,DPL
        MOV     R3,DPH
        RET

;-------------------------------------------------
;       IGUALACION DE LA HORA
;-------------------------------------------------
ighora: MOV     A,#datohora
        MOV     R1,A
        MOV     DPTR,#1005H
        MOV     A,@R1
        MOVX    @DPTR,A
```

```
        INC     R1
        MOV     A,@R1
        MOV     DPTR,#1004H
        MOVX    @DPTR,A
        INC     R1
        MOV     A,@R1
        MOV     DPTR,#1003H
        MOVX    @DPTR,A
        INC     R1
        MOV     A,@R1
        MOV     DPTR,#1002H
        MOVX    @DPTR,A

        LCALL   dia_sem

        RET

;-------------------------------------------------
;       INGRESO DATOS
;-------------------------------------------------
dtempo: SETB    RS0

ser1:   JB      05H,ser2
        SETB    05H

        CLR     11H
        CLR     12H
        CLR     13H
        CLR     14H
        CLR     15H
        CLR     08H                     ;PARA IGUALAR HORA
        CLR     07H
        CLR     10H
        MOV     DPTR,#ram_ext
        MOV     A,DPH
        MOV     R3,A
        MOV     A,DPL
        MOV     R2,A
        MOV     A,datoser
        CJNE    A,#3FH,ser6
        SETB    06H
        SETB    08H
        MOV     R0,#datohora
        SJMP    ser6
ser2:   JB      06H,serX
        MOV     A,R3
        MOV     DPH,A
        MOV     A,R2
        MOV     DPL,A
        MOV     A,datoser
```

```
        MOVX    @DPTR,A
        CLR     A
        MOVX    A,@DPTR
        MOV     P1,A
        CLR     C
        MOV     A,#01H
        ADD     A;R2
        MOV     R2,A
        CLR     A
        ADDC    A,R3
        MOV     R3,A
        SJMP    serXX
serX:   MOV     A,datoser
        MOV     @R0,A
        INC     R0
serXX:  MOV     A,datoser
        CJNE    A,#0FFH,ser6
        SETB    07H
        CLR     05H
        CLR     06H

ser6:   MOV     SBUF,A
        CLR     RS0
        RET

;-----------------------------------------------
;       MOSTRAR HORA EN DISPLAY
;-----------------------------------------------
dhora:  MOV     R0,#segun
        MOV     R5,#06H
sig1:   MOV     A,@R0
        MOV     R4,A
        MOV     A,#30H
        CJNE    A,R4,prg0
        INC     R0
        DJNZ    R5,sig1
        LCALL   dia_sem

prg0:   MOV     R0,#segun
        MOV     DPTR,#1004H
        MOV     R3,#03H
        MOV     dato_lcd,#0C0H
        SETB    00H
        LCALL   w_lcd
        SJMP    prg2
prg1:   MOV     A,#3AH
        MOV     dato_lcd,A
        CLR     00H
        LCALL   w_lcd
prg2:   MOV     R1,#02H
        MOVX    A,@DPTR
```

```
        MOV     R2,A
        SWAP    A
prg3:   ANL     A,#0FH
        ADD     A,#30H
        MOV     @R0,A
        MOV     dato_lcd,A
        CLR     00H
        LCALL   w_lcd
        MOV     A,R2
        INC     R0
        DJNZ    R1,prg3
        DEC     DPL
        DJNZ    R3,prg1
        RET

;-------------------------------------------------
;       INICIALIZACION DEL DISPLAY
;-------------------------------------------------
ini_lcd:MOV     R0,#04H

lcd1:   ACALL   ret_4.5ms
        DJNZ    R0,lcd1

        MOV     R1,#05H
        SETB    00H
        MOV     R2,#00H


lcd2:   MOV     DPTR,#t_ini
        MOV     A,R2
        MOVC    A,@A+DPTR
        MOV     dato_lcd,A
        MOV     DPTR,#lcd
        MOV     A,#00H
        CLR     C
        SUBB    A,R2
        JNC     lcd3
        SETB    02H

lcd3:   LCALL   w_lcd
        INC     R2
        DJNZ    R1,lcd2

        MOV     DPTR,#t_mem_ini
        LCALL   lcd_clr
        LCALL   rom_lcd
        LCALL   ret_2s

        MOV     DPTR,#t_mem_sig
        LCALL   lcd_clr
        LCALL   rom_lcd
```

```
        LCALL   ret_2s

        RET

;-----------------------------------------------
;       BORRAR EL DISPLAY
;-----------------------------------------------
lcd_clr:MOV     dato_lcd,#01H
        SETB    00H
        LCALL   w_lcd

        MOV     R0,#05H
clr1:   ACALL   ret_1ms
        DJNZ    R0,clr1
        RET

;-----------------------------------------------
;       TABLA DE ROM
;-----------------------------------------------
rom_lcd:MOV     pun_lcd,#0FFH
rom1:   CLR     00H
rom2:   INC     pun_lcd
        MOV     A,pun_lcd
        MOVC    A,@A+DPTR
        CJNE    A,#24H,rom3
        JBC     03H,rom4
        SETB    03H
        MOV     dato_lcd,#0C0H
        SETB    00H
        LCALL   w_lcd
        SJMP    rom1

rom3:   MOV     dato_lcd,A
        LCALL   w_lcd
        SJMP    rom2

rom4:   RET

;-----------------------------------------------
;       ESCRIBE EN DISPLAY CONTENIDO DE dato_lcd
;-----------------------------------------------
w_lcd:  PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#lcd

        MOV     A,dato_lcd
        SWAP    A
w_lcd1: ANL     A,#0FH
        JB      00H,w_lcd2
        SETB    ACC.4
```

```
w_lcd2: ACALL   wa_lcd
        JBC     01H,w_lcd3
        SETB    01H
        JB      02H,w_lcd4
        SETB    02H
        ACALL   ret_4.5ms

w_lcd4: MOV     A,dato_lcd
        SJMP    w_lcd1

w_lcd3: POP     DPL
        POP     DPH
        RET

;-----------------------------------------------
;       MOSTRAR UN DATO EN EL DISPLAY
;-----------------------------------------------
wa_lcd: MOVX    @DPTR,A
        SETB    ACC.6
        MOVX    @DPTR,A
        LCALL   ret_1ms
        CLR     ACC.6
        MOVX    @DPTR,A
        RET

;-----------------------------------------------
;       COMPROBACION DEL DIA DE LA SEMANA
;-----------------------------------------------
dia_sem:MOV     dato_lcd,#80H
        SETB    00H
        LCALL   w_lcd
        MOV     DPTR,#1005H
        MOVX    A,@DPTR
        SETB    03H
        CJNE    A,#07H,dia1
        MOV     DPTR,#di7
        LCALL   rom_lcd
dia1:   CJNE    A,#06H,dia2
        MOV     DPTR,#di6
        LCALL   rom_lcd
dia2:   CJNE    A,#05H,dia3
        MOV     DPTR,#di5
        LCALL   rom_lcd
dia3:   CJNE    A,#04H,dia4
        MOV     DPTR,#di4
        LCALL   rom_lcd
dia4:   CJNE    A,#03H,dia5
        MOV     DPTR,#di3
        LCALL   rom_lcd
dia5:   CJNE    A,#02H,dia6
        MOV     DPTR,#di2
```

```
        LCALL   rom_lcd
dia6:   CJNE    A,#01H,finds
        MOV     DPTR,#di1
        LCALL   rom_lcd
finds:  RET

;-------------------------------------------------
;       DESPLEGAR ESTADO DE SALIDAS (1era FILA)
;-------------------------------------------------
barsal: SETB    03H
        MOV     DPTR,#dsal
        LCALL   rom_lcd
        RET

;-------------------------------------------------
;       DESPLEGAR ESTADO DE SALIDAS (2da FILA)
;-------------------------------------------------
barsal1:MOV     dato_lcd,#0C0H
        SETB    00H
        LCALL   w_lcd
        SETB    03H
        JB      P1.0,bar1
        MOV     DPTR,#off
        LCALL   rom_lcd
        SJMP    bar2
bar1:   MOV     DPTR,#on
        LCALL   rom_lcd
bar2:   MOV     dato_lcd,#0C4H
        SETB    00H
        LCALL   w_lcd
        SETB    03H
        JB      P1.1,bar3
        MOV     DPTR,#off
        LCALL   rom_lcd
        SJMP    bar4
bar3:   MOV     DPTR,#on
        LCALL   rom_lcd
bar4:   MOV     dato_lcd,#0C8H
        SETB    00H
        LCALL   w_lcd
        SETB    03H
        JB      P1.2,bar5
        MOV     DPTR,#off
        LCALL   rom_lcd
        SJMP    bar6
bar5:   MOV     DPTR,#on
        LCALL   rom_lcd
bar6:   MOV     dato_lcd,#0CCH
        SETB    00H
        LCALL   w_lcd
        SETB    03H
```

```
        JB      P1.2,bar7
        MOV     DPTR,#off
        LCALL   rom_lcd
        SJMP    bar8
bar7:   MOV     DPTR,#on
        LCALL   rom_lcd
bar8:   RET

;-----------------------------------------------
;       RETARDO DE 1 ms
;-----------------------------------------------
ret_1ms:MOV     TL0,#0BEH
        MOV     TH0,#0FAH
        CLR     04H
        SETB    TR0
ms1_1:  JNB     04H,ms1_1
        RET

;-----------------------------------------------
;       RETARDO DE 4.5 ms
;-----------------------------------------------
ret_4.5ms:      MOV     TL0,#072H
                MOV     TH0,#0CFH
                CLR     04H
                SETB    TR0
ms5_1:          JNB     04H,ms5_1
                RET

;-----------------------------------------------
;       RETARDO DE 2 s
;-----------------------------------------------
ret_2s: MOV     R7,#12H
s2_1:   MOV     TL0,#00H
        MOV     TH0,#01H
        CLR     04H
        SETB    TR0
s2_2:   JNB     04H,s2_2
        DJNZ    R7,s2_1
        RET

;-----------------------------------------------
;       INTERRUPCION TIMER 0
;-----------------------------------------------
INTT0:  SETB    04H
        CLR     TR0
        RETI

;-----------------------------------------------
;       INTERRUPCION SERIAL
;-----------------------------------------------
```

```
SERIAL: MOV     diracc,A
        JB      TI,FINSERIAL
        MOV     A,SBUF
        MOV     datoser,A
        SETB    09H

FINSERIAL:
        CLR     TI
        CLR     RI
        MOV     A,diracc
        RETI

;-------------------------------------------------
;       TABLAS
;-------------------------------------------------
t_ini:  DB      33H,32H
        DB      28H
        DB      0CH
        DB      06H

t_mem_ini:      DB      '....FRANCISCO...$'
                DB      '.....TOLEDO.....$'

t_mem_sig:      DB      '..TEMPORIZADOR..$'
                DB      '.. PROGRAMABLE..$'

di1:            DB      'DOMINGO        $'
di2:            DB      'LUNES          $'
di3:            DB      'MARTES         $'
di4:            DB      'MIERCOLES      $'
di5:            DB      'JUEVES         $'
di6:            DB      'VIERNES        $'
di7:            DB      'SABADO         $'

dsal:           DB      'S0  S1  S2  S3$'

on:             DB      'ON  $'
off:            DB      'OFF $'

espvac:         DB      '       $'

END
```

ANEXO 2

ANEXO 2

PROGRAMA DEL MICROCONTROLADOR

El programa para el interfaz gráfico fue desarrollado aprovechando las librerías de C contenidas en el programa CSCAPE.

A continuación se presenta el listado del programa:

```
#include <stdio.h>
#include <time.h>          /*  for popdecl.h  */
#include <dos.h>
#include <io.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>

#include "cscape.h"
#include "ostdlib.h"       /*   for exit(), otolower()   */
#include "framer.h"
#include "scancode.h"
#include "dpref.h"
#include "ogldecl.h"
#include "popdecl.h"       /*  for opc_View  */
#include "oakalloc.h"      /*   for omalloc of icon title string
*/
#include "oaktag.h"        /*   tags for omalloc */

OSTATIC void         about_box(void);
OSTATIC sed_type     background(void);
OSTATIC void         loop_principal(void);
OSTATIC sed_type     fun_MakeSed(sed_type *headsed, int id);
OSTATIC sed_type     fun_DeleteSed(sed_type *headsed, sed_type
sed);
OSTATIC int          consulta(int reg);
OSTATIC int          tx_datos(void);
OSTATIC int          tx_hora(void);
OSTATIC spc_func     (spc_FunCommand);
OSTATIC aux_func     (icwin_FunAux);
OSTATIC boolean      do_command(sed_type sed, int scancode, int
*codep);
OSTATIC void         chequeo(int k);
OSTATIC int          seleccion();
```

```
OSTATIC   sed_type  ig_h(sed_type *headsed);
OSTATIC void        pre_igu(void);
OSTATIC void        t_registros(void);
OSTATIC int         chk(int i);


/* Comandos */

#define TRUE   1
#define FALSE  0
#define V_INF  11
#define V_VPR  12
#define V_MIN  13
#define V_MAX  14
#define D_ING  15
#define D_MOD  16
#define D_CHE  17
#define D_TRA  18
#define D_TOD  19
#define H_IGU  20
#define H_TRA  21
#define H_VER  22
#define S_ESC  23
#define SELEC  24

typedef struct
{
     char dia[10];
     int  hora, minutos, segundos, estado, salida, intdia;
} registro;

typedef struct
{
     char dia[10];
     int  hora, minutos, segundos, intdia;
} i_hora;

struct frame_def menu_bar[]={

{ "@VENTANA",          FNULL,    0},
{ "@Información (F1)",   FNULL,    V_INF},
{ "@Minimizar   (F7)",   FNULL,    V_MIN},
{ "M@aximizar   (F8)",   FNULL,    V_MAX},
{ FRAME_END },
{ "@DATOS",            FNULL,    0},
{ "@Ingreso     (F2)",   FNULL,  D_ING},
{ "@Modificación(F3)",   FNULL,    D_MOD},
{ "@Chequeo     (F4)",   FNULL,    D_CHE},
{ "@Ver. los Reg.   ",   FNULL,    D_TOD},
{ "@Transmisión (F5)",   FNULL,    D_TRA},
{ FRAME_END },
```

```
{ "@HORA",           FNULL,    0},
{ "@Igualación  (F6)",   FNULL,    H_IGU},
{ "@Verificación    ",  FNULL,  H_VER},
{ "@Transmisión     ",   FNULL,    H_TRA},
{ FRAME_END },
{ "@SALIR (ESC)",    FNULL,    S_ESC},
{ FRAME_END },
{ FRAME_END }
};

/* Variables globales */

i_hora         igh[1];
registro  miregistro[225];
unsigned int   COM, LSR, LCR, MCR;
int       lu = 0, ma = 0, mi = 0, ju = 0, vi = 0, sa = 0, dom
= 0;
sed_type  sedact = NULL;
sed_type  menubar = NULL;
pmap_type icon_pmap = NULL;
char      *dia_sem;

/* Atributos de la ventana */

byte background_attr, selected_attr, regular_attr;

int main(int argc, char *argv[])
{
     sed_type  back;
     FILE *fp;

     if ((argc > 1 && *(argv[argc-1]) == 't')
     || !disp_Init(def_ModeGraphics, FNULL)) {
          /* modo de texto */
          disp_Init(def_ModeText, FNULL);
     }
     else {
          ogl_Init(def_OGL);
     }

     background_attr = regular_attr = 0x87;
     selected_attr = 0x78;

/* Inicializa bdcua para gráficos y texto */

     bdcua_InitCombo();

/* Enciende manejador de archivos .pcx */

     pmap_IoInit();
```

```
     if((fp = fopen("demofm.icn", "rb")) != NULL){
          opc_Message(NULL, NULL, 0, "Escuela Politécnica
Nacional");
          delay(1000);
          icon_pmap = pmap_Load(fp, def_PmapFilePCX);
          fclose(fp);
          if(icon_pmap != NULL){
               opc_Message(NULL, "Temporizador programable",
0,
                              "    Francisco Toledo");
               delay(2000);
               pmap_DispLoad(icon_pmap);
          }
          opc_Message(NULL, NULL, 0, NULL);
     }
     background_attr = regular_attr = 0x07;
     selected_attr = 0x70;
     disp_MapMono(disp_GetColors() <= 2L );

/* Habilitación del mouse */

     hard_InitMouse();
     sedwin_MouseInit();

     back = background();
     loop_principal();

     sed_Close(menubar);
     sed_Close(back);

     disp_Close();

     exit(0);
     return(0);
}

static sed_type background()
{
     register  int i;
     menu_type       menu;
     sed_type  sed;

     menu = menu_Open();

     for (i = 0; i < disp_GetHeight(); i++){
          menu_Printf(menu, "@[%d,\260]\n", disp_GetWidth());
     }
     menu_Flush(menu);

     sed = sed_Open(menu);
     sed_SetShadowAttr(sed, 0x08);
```

```
     sed_Repaint(sed);

     menubar = frame_Open(menu_bar, bd_1, background_attr,
selected_attr, background_attr);
     frame_Repaint(menubar);

     return(sed);
}

static void loop_principal()
{
     sed_type  headsed = NULL;
     sed_type  next;
     int       fldno;
     byte      reg, sel, bck;
     unsigned char  bandera = TRUE, ighora = FALSE, modband =
FALSE;
     int       i = 0, j = 0, dcont = 0, mod = 0, cont = 1;
     menu_type menu;
     FILE      *arch;

     arch = fopen("regis.dat", "r");

     for (i = 0;i < 225; i++)
     {
          fscanf(arch, "%s %d %d %d %d %d %d\n",
          miregistro[i].dia,
          &miregistro[i].intdia,
          &miregistro[i].hora,
          &miregistro[i].minutos,
          &miregistro[i].segundos,
          &miregistro[i].salida,
          &miregistro[i].estado);
     }
     fclose(arch);

     i = 0;
     sedact = fun_MakeSed(&headsed, i);

     while(bandera == TRUE ){
          switch(sed_Go(sedact)){
               case S_ESC:
                    bandera = FALSE;
                    break;

               case V_INF:
                    about_box();
                    break;

               case V_MIN:
                    if (win_IsIconified(sedact)) {
```

```
icwin_DeIconify(win_GetIconWin(sedact));
                    }
                    else {
                         win_Iconify(sedact);
                    }
                    break;

               case V_MAX:
                    if (!win_IsIconified(sedact)) {
                         win_Maximize(sedact);
                    }
                    break;

               case D_ING:
                    ighora = FALSE;
                    if (cont < 225)
                    {
                         if (miregistro[i].hora < 24 &&
miregistro[i].hora >= 0 &&
                         miregistro[i].minutos < 60 &&
miregistro[i].minutos >= 0 &&
                         miregistro[i].segundos < 60 &&
miregistro[i].segundos >= 0 &&
                         miregistro[i].salida < 4 &&
miregistro[i].salida >= 0 &&
                         miregistro[i].estado < 2 &&
miregistro[i].estado >= 0 &&
                         miregistro[i].dia != "(null)")
                         {
                              if (!modband)
                              {
                                   i++;
                                   cont++;
                              }
                              sedact = fun_DeleteSed(&headsed,
sedact);
                              sedact = fun_MakeSed(&headsed,
i);
                         }
                         else
                         {
                              ighora = FALSE;
                              sedact = fun_DeleteSed(&headsed,
sedact);
                              sedact = fun_MakeSed(&headsed,
i);
                         }
                    }
                    else
                    {
```

```
                    opc_View(NULL, NULL, 0," No puede
ingresar más datos");
                    }
                    modband = FALSE;
                    break;

               case D_MOD:
                    modband = TRUE;
                    ighora = FALSE;
                    mod = consulta(j);
                    if (mod < 225)
                    {
                         sedact = fun_DeleteSed(&headsed,
sedact);
                         sedact = fun_MakeSed(&headsed, mod);
                    }
                    else
                    {
                         opc_View(NULL, NULL, 0,"El registro
no existe");
                    }
                    break;

               case D_CHE:
                    j = consulta(j);
                    if (j < 225) chequeo(j);
                    else
                    {
                         opc_View(NULL, NULL, 0," Registro no
disponible");
                         j = 0;
                    }
                    break;

               case D_TRA:
                    tx_datos();
                    break;

               case D_TOD:
                    t_registros();
                    break;

               case H_IGU:
                    sedact = fun_DeleteSed(&headsed, sedact);
                    sedact = ig_h(&headsed);
                    ighora = TRUE;
                    break;

               case H_TRA:
                    if (igh[0].dia != NULL &&
                    igh[0].hora < 24 && igh[0].dia >= 0 &&
```

```
                    igh[0].minutos < 60 && igh[0].minutos >= 0
&&
                    igh[0].segundos < 60 && igh[0].segundos >=
0)
                    {
                         tx_hora();
                         break;
                    }
                    else
                    {
                         opc_View(NULL, NULL, 0,"Ingrese
nuevamente la hora");
                         break;
                    }

               case H_VER:
                    pre_igu();
                    break;

               case SELEC:
                    dcont = seleccion();
                    if (ighora)
                    {
                         igh[0].intdia = dcont;
                    }
                    else
                    {
                         if (modband)
                         {
                              miregistro[mod].intdia = dcont;
                         }
                         else
                         {
                              miregistro[i].intdia = dcont;
                         }
                    }
                    break;

               default:
                    break;
          }
     }
     arch = fopen("regis.dat", "w");
     for (i = 0;i < 225; i++)
     {
          fprintf(arch, "%s %d %d %d %d %d %d\n",
          miregistro[i].dia,
          miregistro[i].intdia,
          miregistro[i].hora,
          miregistro[i].minutos,
          miregistro[i].segundos,
```

```
            miregistro[i].salida,
            miregistro[i].estado);
      }
      fclose(arch);
}

static sed_type fun_MakeSed(sed_type *headsed, int id)
{
      menu_type menu;
      sed_type  sed;
      char      titulo[22];
      char      *icontitle;


      if ((menu = menu_Open()) == NULL) {
            return(NULL);
      }
      menu_Printf(menu, " Día:   @fd[##########]\n",
miregistro[id].dia, &string_funcs);
      menu_Printf(menu, " Hora:         @f[###]\n",
&miregistro[id].hora, &int_funcs);
      menu_Printf(menu, " Minutos:      @f[###]\n",
&miregistro[id].minutos, &int_funcs);
      menu_Printf(menu, " Segundos:     @f[###]\n",
&miregistro[id].segundos, &int_funcs);
      menu_Printf(menu, " Salida:        @f[##]\n",
&miregistro[id].salida, &int_funcs, (1, 4));
      menu_Printf(menu, " Estado:        @f[##]\n",
&miregistro[id].estado, &int_funcs, (0, 1));
      menu_Flush(menu);

      if ((sed = sed_Open(menu)) == NULL) {
            menu_Destroy(menu);
            return(NULL);
      }

      sed_SetHeight(sed, 8);
      sed_SetWidth(sed, 28);

      sed_SetAux(sed, aux_Top);
      sed_SetSpecial(sed, spc_FunCommand);

      if (!sed_SetBorder(sed, bd_cua)) {
            sed_Close(sed);
            return(NULL);
      }
      sprintf(titulo, "Ingreso de datos %d", id);
      sed_SetBorderColor(sed, background_attr);
      sed_SetBorderTitle(sed, titulo);
      sed_SetBorderFeature(sed, BD_TITLE | BD_MOVE | BD_RESIZE
| BD_MAXMIN | BD_SYSBOX |
```

```
     BD_VTSHRINK | BD_HIDEHANDLES);

     win_SetIconAux(sed, icwin_FunAux);
     win_SetIconPmap(sed, icon_pmap);

     if ((icontitle = (char *)omalloc(OA_NOTAG, sizeof(char) *
10)) != NULL) {
          sprintf(icontitle, "Icono");
          win_SetIconText(sed, icontitle);
     }

     sed_SetPosition(sed, 10, 20);
     sed_SetShadow(sed, 1);
     sed_SetShadowAttr(sed, 0x08);
     sed_SetColors(sed, background_attr, regular_attr,
selected_attr);
     sed_SetMouse(sed, winmou_All);
     win_SetMouseFeature(sed, MOUF_TRACK);

     if (*headsed == NULL) {
          *headsed = sed;
     }

     sed_Repaint(sed);
     return(sed);
}

static sed_type fun_DeleteSed(sed_type *headsed, sed_type sed)
{
     sed_type  curr = NULL;
     sed_type  last, next;
     char      *icontitle;

     for (last = next = *headsed; next != NULL; next =
(sed_type) sed_GetData(next)) {
          if (sed == next) {

               if (sed == *headsed) {
                    *headsed = (sed_type) sed_GetData(sed);
                    curr = *headsed;
               }
               else {
                    sed_SetData(last, sed_GetData(sed));
                    if ((curr = (sed_type) sed_GetData(sed))
== NULL) {
                         curr = *headsed;
                    }
               }

               if ((icontitle = win_GetIconText(sed)) != NULL)
{
```

```
                    ofree(OA_NOTAG, (VOID *)icontitle);
                    win_SetIconText(sed, NULL);
               }
               sed_Close(sed);
               break;
          }

          last = next;
     }
     return(curr);
}

static int consulta(int reg)
{
     menu_type menu;
     sed_type  sed;
     char      titulo[22];
     char      *icontitle;

     if ((menu = menu_Open()) == NULL) {
          return(NULL);
     }
     menu_Printf(menu, " Registro:    @f[####]\n", &reg,
&int_funcs);
     menu_Flush(menu);

     if ((sed = sed_Open(menu)) == NULL) {
          menu_Destroy(menu);
          return(NULL);
     }

     sed_SetHeight(sed, 2);
     sed_SetWidth(sed, 28);

     if (!sed_SetBorder(sed, bd_cua)) {
          sed_Close(sed);
          return(NULL);
     }
     sprintf(titulo, "Consulta");
     sed_SetBorderColor(sed, background_attr);
     sed_SetBorderTitle(sed, titulo);
     sed_SetBorderFeature(sed, BD_TITLE | BD_DOUBLELINE);

     win_SetIconAux(sed, icwin_FunAux);
     win_SetIconPmap(sed, icon_pmap);

     if ((icontitle = (char *)omalloc(OA_NOTAG, sizeof(char) *
10)) != NULL) {
          sprintf(icontitle, "Consulta");
          win_SetIconText(sed, icontitle);
     }
```

```
     sed_SetPosition(sed, 6, 20);
     sed_SetShadow(sed, 1);
     sed_SetShadowAttr(sed, 0x08);
     sed_SetColors(sed, background_attr, regular_attr,
selected_attr);
     sed_SetMouse(sed, winmou_All);
     win_SetMouseFeature(sed, MOUF_GREEDY);

     sed_Repaint(sed);
     sed_Go(sed);

     sed_Close(sed);
     return(reg);
}

void chequeo(int k)
{
     menu_type menu;
     sed_type  sed;
     char      titulo[22];

     menu = menu_Open();

     menu_Printf(menu, "\n\n      Dia    : %s\n",
miregistro[k].dia);
     menu_Printf(menu, "      Hora   : %2d:%2d:%2d\n",
miregistro[k].hora, miregistro[k].minutos,
miregistro[k].segundos);
     menu_Printf(menu, "      Salida :       %2d\n",
miregistro[k].salida);
     menu_Printf(menu, "      Estado :       %2d\n\n",
miregistro[k].estado);
     menu_Printf(menu, "Pulse cualquier tecla para
continuar");
     menu_Flush(menu);

     sed = sed_Open(menu);

     if (!sed_SetBorder(sed, bd_cua)) {
          sed_Close(sed);
     }
     sprintf(titulo, "Registro %d", k);
     sed_SetBorderColor(sed, background_attr);
     sed_SetBorderTitle(sed, titulo);
     sed_SetBorderFeature(sed, BD_TITLE | BD_DOUBLELINE);

     sed_SetHeight(sed, 10);
     sed_SetWidth(sed, 40);

     sed_SetPosition(sed, 10, 15);
     sed_SetShadow(sed, 2);
```

```
     sed_SetShadowAttr(sed, 0x08);
     sed_SetColors(sed, background_attr, regular_attr,
selected_attr);

     sed_Repaint(sed);
     sed_Go(sed);

     getch();

     sed_Close(sed);
}

seleccion()
{
     menu_type menu;
     sed_type  sed;
     int       ret, fila, columna;
     char      titulo[22];
     unsigned char  banddia = TRUE;

     menu = menu_Open();

     menu_Printf(menu, "Días de la semana\n\n");
     menu_Printf(menu, "@fd2[Lunes]\n", NULL, &gmenu_funcs,
NULL, "2");
     menu_Printf(menu, "@fd2[Martes]\n", NULL, &gmenu_funcs,
NULL, "3");
     menu_Printf(menu, "@fd2[Miércoles]\n", NULL,
&gmenu_funcs, NULL, "4");
     menu_Printf(menu, "@fd2[Jueves]\n", NULL, &gmenu_funcs,
NULL, "5");
     menu_Printf(menu, "@fd2[Viernes]\n", NULL, &gmenu_funcs,
NULL, "6");
     menu_Printf(menu, "@fd2[Sábado]\n", NULL, &gmenu_funcs,
NULL, "7");
     menu_Printf(menu, "@fd2[Domingo]\n", NULL, &gmenu_funcs,
NULL, "1");


     sed = sed_Open(menu);

     if (!sed_SetBorder(sed, bd_cua)) {
          sed_Close(sed);
     }

     sed_SetBorderColor(sed, background_attr);
     sed_SetBorderFeature(sed, BD_DOUBLELINE);

     sed_GetPosition(sedact, &fila, &columna);
     sed_SetPosition(sed, fila+3, columna+10);
```

```
sed_SetMouse(sed, winmou_All);
win_SetMouseFeature(sed, MOUF_GREEDY);

sed_Repaint(sed);
ret = sed_Go(sed);
switch(ret){
     case 2:
          if (lu < 32)
          {
               dia_sem = "Lunes     ";
               lu++;
               banddia = TRUE;
               break;
          }
          else
          {
               banddia = FALSE;
               break;
          }
     case 3:
          if (ma < 32)
          {
               dia_sem = "Martes    ";
               ma++;
               banddia = TRUE;
               break;
          }
          else
          {
               banddia = FALSE;
               break;
          }
     case 4:
          if (mi < 32)
          {
               dia_sem = "Miércoles";
               mi++;
               banddia = TRUE;
               break;
          }
          else
          {
               banddia = FALSE;
               break;
          }
     case 5:
          if (ju < 32)
          {
               dia_sem = "Jueves    ";
               ju++;
               banddia = TRUE;
```

```
                    break;
               }
               else
               {
                    banddia = FALSE;
                    break;
               }
          case 6:
               if (vi < 32)
               {
                    dia_sem = "Viernes   ";
                    vi++;
                    banddia = TRUE;
                    break;
               }
               else
               {
                    banddia = FALSE;
                    break;
               }
          case 7:
               if (sa < 32)
               {
                    dia_sem = "Sábado    ";
                    sa++;
                    banddia = TRUE;
                    break;
               }
               else
               {
                    banddia = FALSE;
                    break;
               }
          case 1:
               if (dom < 32)
               {
                    dia_sem = "Domingo   ";
                    dom++;
                    banddia = TRUE;
                    break;
               }
               else
               {
                    banddia = FALSE;
                    break;
               }
          default:
               break;
     }
     if (banddia == TRUE)
     {
```

```
            sed_SetCurrRecord(sedact, dia_sem);
            sed_UpdateCurrField(sedact);
      }
      else
      {
            opc_View(NULL, "Intente nuevamente", 0," No hay como
realizar más programaciones en ese día");
      }
      sed_Close(sed);
      return(ret);
}

static sed_type ig_h(sed_type *headsed)
{
      menu_type menu;
      sed_type  sed;
      char      titulo[22];
      char      *icontitle;

      menu = menu_Open();

      menu_Printf(menu, "Día:   @fd[##########]\n", igh[0].dia,
&string_funcs);
      menu_Printf(menu, "Hora:         @f[###]\n",
&igh[0].hora, &int_funcs);
      menu_Printf(menu, "Minutos:      @f[###]\n",
&igh[0].minutos, &int_funcs);
      menu_Printf(menu, "Segundos:     @f[###]\n",
&igh[0].segundos, &int_funcs);

      sed = sed_Open(menu);

      if (!sed_SetBorder(sed, bd_cua)) {
            sed_Close(sed);
      }
      sprintf(titulo, "Igualación de la hora");
      sed_SetBorderColor(sed, background_attr);
      sed_SetBorderTitle(sed, titulo);
      sed_SetBorderFeature(sed, BD_TITLE | BD_MOVE | BD_RESIZE
| BD_MAXMIN | BD_SYSBOX |
      BD_VTSHRINK | BD_HIDEHANDLES);

      sed_SetAux(sed, aux_Top);
      sed_SetSpecial(sed, spc_FunCommand);

      sed_SetMouse(sed, winmou_All);
      win_SetMouseFeature(sed, MOUF_TRACK);

      sed_SetHeight(sed, 4);
      sed_SetWidth(sed, 40);
```

```
     win_SetIconAux(sed, icwin_FunAux);
     win_SetIconPmap(sed, icon_pmap);

     if ((icontitle = (char *)omalloc(OA_NOTAG, sizeof(char) *
10)) != NULL) {
          sprintf(icontitle, "Icono");
          win_SetIconText(sed, icontitle);
     }

     sed_SetPosition(sed, 6, 20);
     sed_SetShadow(sed, 1);
     sed_SetShadowAttr(sed, 0x08);
     sed_SetColors(sed, background_attr, regular_attr,
selected_attr);

     if (*headsed == NULL) {
          *headsed = sed;
     }


     sed_Repaint(sed);
     return(sed);
}

void pre_igu(void)
{
     menu_type menu;
     sed_type  sed;
     char      titulo[22];

     menu = menu_Open();

     menu_Printf(menu, "\n\n      Dia    : %s\n", igh[0].dia);
     menu_Printf(menu, "      Hora   : %2d:%2d:%2d\n\n",
igh[0].hora, igh[0].minutos, igh[0].segundos);
     menu_Printf(menu, "Pulse cualquier tecla para
continuar");
     menu_Flush(menu);

     sed = sed_Open(menu);

     if (!sed_SetBorder(sed, bd_cua)) {
          sed_Close(sed);
     }
     sed_SetBorderFeature(sed, BD_DOUBLELINE);

     sed_SetHeight(sed, 10);
     sed_SetWidth(sed, 40);

     sed_SetPosition(sed, 10, 15);
     sed_SetShadow(sed, 2);
```

```
     sed_SetShadowAttr(sed, 0x08);
     sed_SetColors(sed, background_attr, regular_attr,
selected_attr);
     sed_Repaint(sed);
     sed_Go(sed);
     getch();
     sed_Close(sed);
}

void t_registros(void)
{
     menu_type menu;
     sed_type  sed;
     char      titulo[22];
     int       i, j;

     j = 0;
     do
     {
     menu = menu_Open();

     menu_Printf(menu, "Registro   Día            Hora
Salida          Estado    Dia(#)\n");

     for (i = j;i < j + 25;i++)
     {
          menu_Printf(menu,
"\t%2d\t\t%9s\t%2d:%2d:%2d\t%d\t\t\t%d\t\t%d\n", i,
                    miregistro[i].dia,
                    miregistro[i].hora,
                    miregistro[i].minutos,
                    miregistro[i].segundos,
                    miregistro[i].salida,
                    miregistro[i].estado,
                    miregistro[i].intdia);
     }
     menu_Printf(menu, "\t\tPresione cualquier tecla para
continuar");

     sed = sed_Open(menu);

     if (!sed_SetBorder(sed, bd_cua)) {
          sed_Close(sed);
     }
     sed_SetBorderFeature(sed, BD_DOUBLELINE);

     sed_SetPosition(sed, 2, 5);

     sed_SetShadow(sed, 1);
     sed_SetShadowAttr(sed, 0x08);
     sed_SetColors(sed, background_attr, regular_attr,
```

```
selected_attr);
     sed_Repaint(sed);
     sed_Go(sed);
     getch();
     sed_Close(sed);
     j = j + 25;
     } while (j < 225);
}


static boolean do_command(sed_type sed, int scancode, int
*command_codep)
{
     menu_type      menu;
     int       row, col;
     mev_struct     mev;                    /* a mouse event
structure */
     byte      reg, sel, bck;

     /* set current sed to our sed */
     sedact = sed;

     switch(scancode) {
     case ESC:
          *command_codep = S_ESC;
          return(TRUE);

     case FN1:
          *command_codep = V_INF;
          return(TRUE);

     case FN2:
          *command_codep = D_ING;
          return(TRUE);

     case FN3:
          *command_codep = D_MOD;
          return(TRUE);

     case FN4:
          *command_codep = D_CHE;
          return(TRUE);

     case FN5:
          *command_codep = D_TRA;
          return(TRUE);

     case FN6:
          *command_codep = H_IGU;
          return(TRUE);
```

```
     case FN7:
          *command_codep = V_MIN;
          return(TRUE);

     case FN8:
          *command_codep = V_MAX;
          return(TRUE);

     case FN9:
          *command_codep = SELEC;
          return(TRUE);

     case FN10:
          sed_SetNextWin(sed, menubar);
          return(TRUE);

     case MOU_CLICK:
          *command_codep = SELEC;
          break;

     case ENTER:
          return(TRUE);

     case KEY_SYSBOX:
          /* click on CUA border SYSBOX button */
     case KEY_DSYSBOX:
     {
          static char *win_sysbox_menu[] = {
               " Minimize (F7) ",
               " Maximize (F8) ",
               NULL
          };

          opc_type  pc;
          int       ret;
          opcoord        left, top, right, bot;

          if ((pc = opc_Open()) == NULL) {
               return(TRUE);
          }
          bord_GetPixSides(sed, &left, &top, &right, &bot);
          left -= bord_GetXmin(sed);
          top  -= bord_GetYmin(sed);
          opc_SetPixRow(pc, -top);
          opc_SetPixCol(pc, -left);
          opc_SetBorderFeature(pc, opc_GetBorderFeature(pc) &
~BD_MOVE);
          opc_SetBorderFeature(pc, opc_GetBorderFeature(pc) &
~BD_TITLE);

          ret = opc_DoStrList(pc, win_sysbox_menu);
```

```
            opc_Close(pc);

            switch (ret) {
            case 1:
                 *command_codep = V_MIN;
                 break;
            case 2:
                 *command_codep = V_MAX;
                 break;
            }    /*   end of SYSBOX menu choice switch */

            return(TRUE);

      }    /*   end of KEY_SYSBOX block */
}


      return(FALSE);
}

static boolean spc_FunCommand(sed_type sed, int scancode)
{
      int command_code;

      command_code = 0;
      if (do_command(sed, scancode, &command_code)) {
            if (command_code != 0) {
                 sed_SetBaton(sed, command_code);
            }
            sed_ToggleExit(sed);
            return(TRUE);
      }
      else {
            return(FALSE);
      }
}

static int icwin_FunAux(win_type win, int msg, VOID *indata,
VOID *outdata)
{
      switch(msg) {
      case WINA_GO:
            return(icwin_defgo(win));

      case WINA_EVENT:
            switch(*((int *) indata)) {
            case FN10:
                 win_SetNextWin(win, menubar);
                 *((int *) outdata) = MOU_THERE;
                 return(BOB_ABORT);
```

```
            case ENTER:
                 break;

            default:

                 *((int *) outdata) = MOU_HERE;

                 return(do_command(icwin_GetClient(win),
                      *((int *) indata), (int *) outdata) ?
BOB_ABORT : 1);
            }
            break;     /*    (end of WINA_EVENT case) */

      default:
            break;
      }
      return(1);
}


static void about_box(void)
{
      opc_View(NULL, "Temporizador Programable EPN 1995", 0,
                  " Diseñado por Francisco Toledo \n
Interfaz gráfico");
}

void inicia(int com)
{
      if (com) COM=0x3F8;
      else COM=0x2F8;

      LSR=COM+5;
      LCR=COM+3;
      MCR=COM+4;

      outportb(LCR,0x80);
      outportb(COM+1,0);
      outportb(COM,0x0C);
      outportb(LCR,0x07);
      outportb(MCR,0x03);
      inportb(COM);
}

unsigned char recibe(int a)
{
      int         b;
      unsigned char  error;

      while(!(inportb(LSR)&0x20));
      outportb(COM,a);
```

```
     while(!(inportb(LSR)&0x01));
     b = inport(COM);
     b = b & 0x0ff;
     if (b==a)
     {
          error = TRUE;
          return(b);
     }
     else
     {
          opc_Message(NULL, NULL, 0, "Error en la
transmisión");
          error = FALSE;
          return (error);
     }
}

int chk(int i)

{
  if(i==0x41||i==0x61)i=0xa;
  if(i==0x42||i==0x62)i=0xb;
  if(i==0x43||i==0x63)i=0xc;
  if(i==0x44||i==0x64)i=0xd;
  if(i==0x45||i==0x65)i=0xe;
  if(i==0x46||i==0x66)i=0xf;
  if(i==' ')i=0x0;
  if(i>=0x30 && i<=0x39)i=i-0x30;
  return(i);
}

int tx_datos(void)
{
     menu_type menu;
     sed_type  sed, headsed = NULL;
     int       com, i, j, k;
     unsigned char  a, error, titulo[22], dtx[10];
     FILE           *arch;
     int       t_ar[4];
     float          intdia, hora, min, seg, x, b[225];
     char      xdia[10];
     int       xhora, xminutos, xsegundos, xestado, xsalida,
xintdia;

     miregistro[225].dia[0] = '\0';
     miregistro[225].intdia = 0;
     miregistro[225].hora = 0;
     miregistro[225].minutos = 0;
     miregistro[225].segundos = 0;
     miregistro[225].salida = 0;
     miregistro[225].estado = 0;
```

```
     sedact = fun_DeleteSed(&headsed, sedact);
     sedact = fun_MakeSed(&headsed, 225);

     com = 1;

     menu = menu_Open();
     menu_Printf(menu, "           COM : @f[##]", &com,
&int_funcs);
     menu_Flush(menu);
     sed = sed_Open(menu);
     sed_SetPosition(sed, 4, 20);
     sed_SetHeight(sed, 1);
     sed_SetWidth(sed, 26);

     if (!sed_SetBorder(sed, bd_cua)) {
          sed_Close(sed);
          return(NULL);
     }
     sprintf(titulo, "Ingrese el pórtico serial");
     sed_SetBorderColor(sed, background_attr);
     sed_SetBorderTitle(sed, titulo);
     sed_SetBorderFeature(sed, BD_TITLE | BD_MOVE | BD_RESIZE
| BD_VTSHRINK | BD_HIDEHANDLES);

     sed_SetShadow(sed, 1);
     sed_SetShadowAttr(sed, 0x08);
     sed_SetColors(sed, background_attr, regular_attr,
selected_attr);
     sed_SetMouse(sed, winmou_All);
     win_SetMouseFeature(sed, MOUF_GREEDY);
     sed_Repaint(sed);
     sed_Go(sed);

     sed_Close(sed);

     inicia(com);
     while(TRUE)
     {
        arch = fopen("d_tempo.dat", "w");

        opc_Message(NULL, NULL, 0, "Ordenando datos para
transmitir");
        fprintf(arch, "%2x\n", 0x00);

        for (i = 0; i < 225; i++)
        {
          seg = miregistro[i].segundos;
          min = miregistro[i].minutos * 100.;
          hora = miregistro[i].hora * 10000.;
          intdia = miregistro[i].intdia * 1000000.;
          b[i] = intdia + hora + min + seg;
```

```
        }
        for (i = 0; i < 224; i++)
        {
          for ( k = i; k >= 0; k--)
          {
               if (b[k] > b[k+1])
               {
                    x = b[k];
                    strcpy(xdia, miregistro[k].dia);
                    xintdia = miregistro[k].intdia;
                    xhora = miregistro[k].hora;
                    xminutos = miregistro[k].minutos;
                    xsegundos = miregistro[k].segundos;
                    xestado = miregistro[k].estado;
                    xsalida = miregistro[k].salida;
                    b[k] = b[k+1];
                    strcpy(miregistro[k].dia,
miregistro[k+1].dia);
                    miregistro[k].intdia =
miregistro[k+1].intdia;
                    miregistro[k].hora = miregistro[k+1].hora;
                    miregistro[k].minutos =
miregistro[k+1].minutos;
                    miregistro[k].segundos =
miregistro[k+1].segundos;
                    miregistro[k].estado =
miregistro[k+1].estado;
                    miregistro[k].salida =
miregistro[k+1].salida;
                    b[k+1] = x;
                    strcpy(miregistro[k+1].dia, xdia);
                    miregistro[k+1].intdia = xintdia;
                    miregistro[k+1].hora = xhora;
                    miregistro[k+1].minutos = xminutos;
                    miregistro[k+1].segundos = xsegundos;
                    miregistro[k+1].estado = xestado;
                    miregistro[k+1].salida = xsalida;
               }
          }
        }


        for (i = 0; i < 225; i++)
        {
          if (miregistro[i].intdia == 0)
          {
               t_ar[0] = 80;
          }
          else
          {
               t_ar[0] = miregistro[i].hora;
```

```
            }
            t_ar[1] = miregistro[i].minutos;
            t_ar[2] = miregistro[i].segundos;
            t_ar[3] = miregistro[i].estado * 64 +
miregistro[i].salida * 8 + miregistro[i].intdia;
            fprintf(arch, "%2d%2d%2d%2x\n", t_ar[0], t_ar[1],
t_ar[2], t_ar[3]);
          }
          fprintf(arch, "%2x", 0xff);

          fclose(arch);

          while(TRUE)
          {
            if ((arch=fopen("d_tempo.dat","r"))!=NULL)
            {
                 fseek(arch, 0, SEEK_SET);
                 {
                      opc_Message(NULL, NULL, 0, "Transmitiendo
archivo");
                      while(!feof(arch))
                      {
                           k = 0;
                           fgets(dtx, 10, arch);
                           do
                           {
                                i = (int) *(dtx+k);
                                i = chk(i);
                                j = (int) *(dtx+k+1);
                                j = chk(j);
                                i = i * 0x10 + j;
                                k = k + 2;
                                delay(50);
                                recibe(i);
                           }while(k<(strlen(dtx)-1));
                      }
                      fclose(arch);
                      opc_Message(NULL, "Fin de Transmisión", 0,
"Pulse cualquier tecla para continuar");
                      getch();
                      opc_Message(NULL, NULL, 0, NULL);
                      return (FALSE);
                 }
            }
          }
      }
}


int tx_hora(void)
{
```

```
     menu_type      menu;
     sed_type  sed;
     int       com, i, j, k;
     unsigned char  a, error, titulo[22], dtx[10];
     FILE           *arch;

     com = 1;

     menu = menu_Open();
     menu_Printf(menu, "           COM : @f[##]", &com,
&int_funcs);
     menu_Flush(menu);
     sed = sed_Open(menu);
     sed_SetPosition(sed, 4, 20);
     sed_SetHeight(sed, 1);
     sed_SetWidth(sed, 26);

     if (!sed_SetBorder(sed, bd_cua)) {
          sed_Close(sed);
          return(NULL);
     }
     sprintf(titulo, "Ingrese el pórtico serial");
     sed_SetBorderColor(sed, background_attr);
     sed_SetBorderTitle(sed, titulo);
     sed_SetBorderFeature(sed, BD_TITLE | BD_MOVE | BD_RESIZE
| BD_VTSHRINK | BD_HIDEHANDLES);

     sed_SetShadow(sed, 1);
     sed_SetShadowAttr(sed, 0x08);
     sed_SetColors(sed, background_attr, regular_attr,
selected_attr);
     sed_SetMouse(sed, winmou_All);
     win_SetMouseFeature(sed, MOUF_GREEDY);
     sed_Repaint(sed);
     sed_Go(sed);

     sed_Close(sed);

     inicia(com);
     while(TRUE)
     {
          arch = fopen("ig_hora.dat", "w");

          fprintf(arch, "%2x\n", 0x3f);
          fprintf(arch, "%2d%2d%2d%2d\n", igh[0].intdia,
               igh[0].hora,
               igh[0].minutos,
               igh[0].segundos);
          fprintf(arch, "%2x", 0xff);

          fclose(arch);
```

```
          if ((arch=fopen("ig_hora.dat","r"))!=NULL)
          {
               fseek(arch, 0, SEEK_SET);
               opc_Message(NULL, NULL, 0, "Transmitiendo
archivo");
               {
                    while(!feof(arch))
                    {
                         k = 0;
                         fgets(dtx, 10, arch);
                         do
                         {
                              i = (int) *(dtx+k);
                              i = chk(i);
                              j = (int) *(dtx+k+1);
                              j = chk(j);
                              i = i * 0x10 + j;
                              if (i!=0x3f && i!=0xff)
                              {
                                   i = (int) *(dtx+k);
                                   i = chk(i);
                                   i = i * 0x10 + j;
                              }
                              k = k + 2;
                              delay(50);
                              recibe(i);
                         }while(k<(strlen(dtx)-1));
                    }
                    opc_Message(NULL, "Fin de Transmisión", 0,
"Pulse cualquier tecla para continuar");
                    getch();
                    opc_Message(NULL, NULL, 0, NULL);
                    fclose(arch);
                    return (FALSE);
               }
          }
     }
}
```

ANEXO 3

ANEXO 3

PROGRAMA PARA LA GRABACION DE LA NVRAM

Para utilizar este programa se debe ingresar la dirección del pórtico dependiendo del computador que se ocupe.

En el prompt de DOS se debe introducir la siguiente instrucción:

> WX1 <DIRECCION DEL PORTICO>

A continuación tenemos el programa desarrollado en lenguaje C:

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>

#define tiempo 10

formato(char dato[]);
write(int i,int a[]);
int out,in,outin;
int chequeo(int);

main(int argc,char *argv[])
{
 char dato[80], nombre[20], hex0, hex1, hex2;
 int h0, h1, h2;
 FILE *programa;

 clrscr();
 hex0=*(argv[1]);
 hex1=*(argv[1]+1);
 hex2=*(argv[1]+2);
 h0=(int)hex0;
 h0=chequeo(h0);
 h1=(int)hex1;
 h1=chequeo(h1);
 h2=(int)hex2;
 h2=chequeo(h2);
```

```
 out=h0*0x100+h1*0x10+h2;
 in=out+1;
 outin=out+2;
 printf("\nIngrese el nombre del archivo ");
 gets(nombre);
 if ((programa=fopen(nombre,"rb"))!=NULL)
 {
  fseek(programa, 0, SEEK_SET);
  {
   while(1)
   {
    fgets(dato,80,programa);
    if(*(dato+8)!=0x31)
    formato(dato);
     else
     {
      fclose(programa);
      exit(0);
     }
    }
  }
 }
 return(0);
}

/*Esta subrutina permite el chequeo de los datos para
convertirlos en valores hexadecimales*/

int chequeo(int i)
{
  if(i==0x41)i=0xa;
  if(i==0x42)i=0xb;
  if(i==0x43)i=0xc;
  if(i==0x44)i=0xd;
  if(i==0x45)i=0xe;
  if(i==0x46)i=0xf;
  if(i>=0x30 && i<=0x39)i=i-0x30;
  return(i);
}

/* Esta subrutina permite obtener la dirección y el dato para
luego enviarlos tanto a pantalla como a su grabación en la
memoria NVRAM*/

formato(char dato[80])

{
  int i,j,k,a1,a2,a3,a4,dir,a[4],add,qq;

  k=1;
  do
  {
```

```
 /*Desechamos el primer caracter que corresponde a ":". Los
siguientes datos se los coge de dos en dos para su posterior
tratamiento*/

    i=(int) *(dato+k);
    i=chequeo(i);
    j=(int) *(dato+k+1);
    j=chequeo(j);
    if(k!=3)i=i*0x10+j;

/*En el cuarto caracter comienza la dirección compuesta por 2
bytes, es decir, cuatro valores hexadecimales*/

    if(k==3)
    {
      a1=(int) *(dato+k);
      a1=chequeo(a1);
      a2=(int) *(dato+k+1);
      a2=chequeo(a2);
      a3=(int) *(dato+k+2);
      a3=chequeo(a3);
      a4=(int) *(dato+k+3);
      a4=chequeo(a4);
      dir=a1*0x10*0x10*0x10+a2*0x10*0x10+a3*0x10+a4;
      k=k+2;
    }
     k=k+2;

/*Los datos a ser grabados se encuentran a partir del noveno
caracter.  En esta subrutina se saca tanto el valor de la
dirección y el dato hexadecimal a grabar*/

    if(k>9)
    {
      printf("add = %x dato = %x\t",dir,i);
      add=dir;
      a[0]=dir/(0x10*0x10*0x10);
      dir=dir-a[0]*0x10*0x10*0x10;
      a[1]=dir/(0x10*0x10);
      dir=dir-a[1]*0x10*0x10;
      a[2]=dir/0x10;
      dir=dir-a[2]*0x10;
      a[3]=dir;
      dir=dir-a[3];
      add+=0x01;
      dir=add;
      printf("MSB = %x LSB = %x  ",a[0]*0x10+a[1]
,a[2]*0x10+a[3]);
      write(i,a);
    }
```

```
  } while(k<=(strlen(dato)-6));
  return 0;
}

/*Esta subrutina corresponde a la grabación en la NVRAM como se
indica en el documento anterior*/

write(int i,int a[4])

{
  int lsb,msb;

  lsb=a[2]*0x10+a[3];
  msb=a[0]*0x10+a[1];
  outportb(outin,0x01);// Todo deshabilitado
  delay(tiempo);
  outportb(outin,0x00);// Habilitacion latch 1
  delay(tiempo);
  outportb(out,lsb);//     add LSB
  outportb(outin,0x01);// Todo deshabilitado
  delay(tiempo);
  outportb(outin,0x05);// Habilitacion latch 2
  delay(tiempo);
  outportb(out,msb);//  add MSB
  delay(tiempo);
  outportb(outin,0x01);// Todo deshabilitado
  delay(tiempo);
  outportb(outin,0x09);// Habilitacion de escritura
  delay(tiempo);
   outportb(out,i);//     dato
  delay(tiempo);//getchar();
  outportb(outin,0x01);// Todo deshabilitado
  return 0;
}
```

Como ya se indicó la lectura de los datos es importante para la comprobación de los datos que se han grabado. El siguiente programa es el que se utiliza para la lectura.

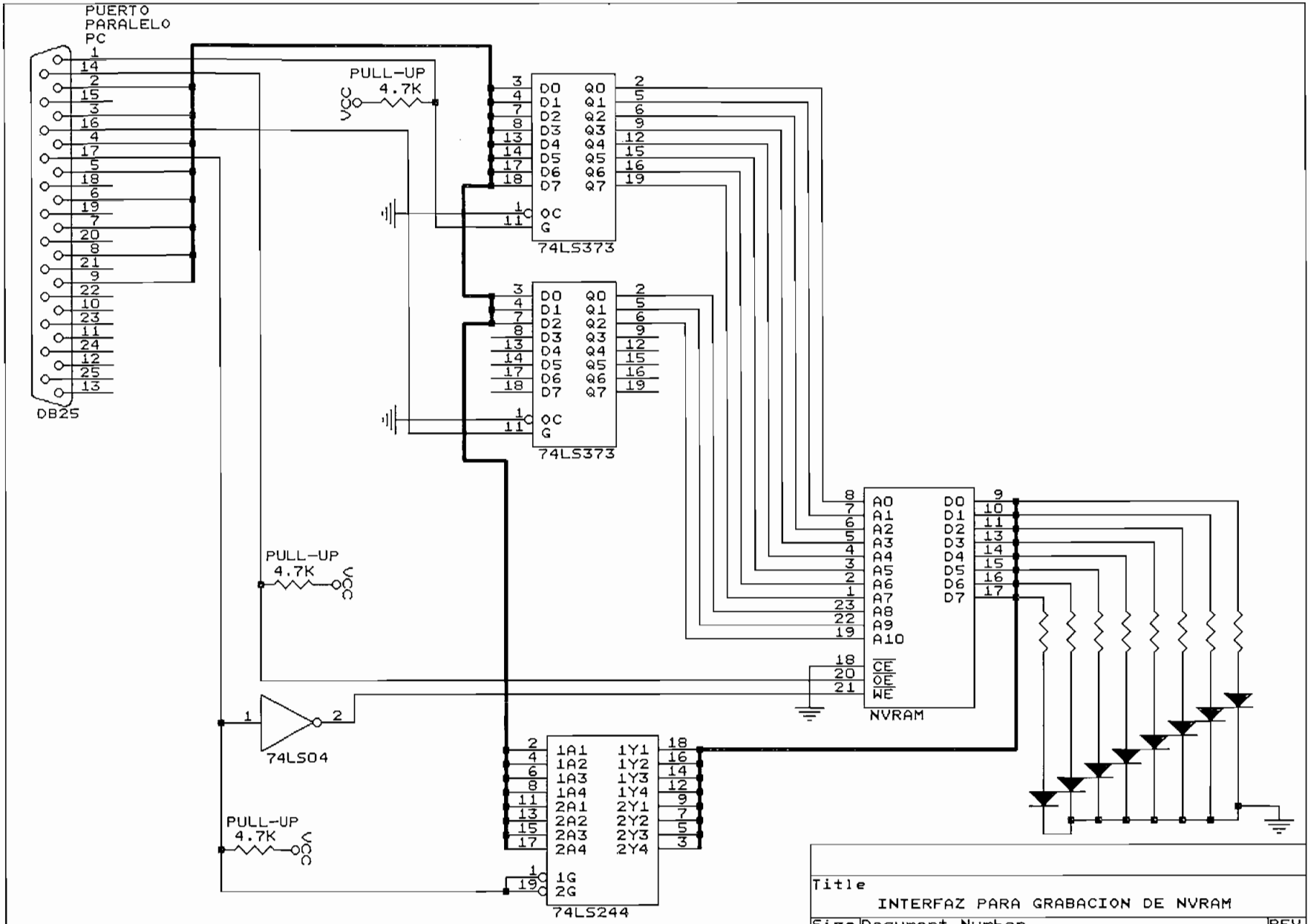
Al igual que en el anterior programa se debe ingresar de la siguiente manera:

> RX1 <DIRECCION DEL PORTICO>

El programa es el siguiente:

```
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <iostream.h>

#define tiempo 100
int out,in,outin;
int chequeo(int);

main(int argc,char *argv[])
{
 char c, hex0, hex1, hex2;
 int i, j, h0, h1, h2;

 clrscr();
 hex0=*(argv[1]);
 hex1=*(argv[1]+1);
 hex2=*(argv[1]+2);
 h0=(int)hex0;
 h0=chequeo(h0);
 h1=(int)hex1;
 h1=chequeo(h1);
 h2=(int)hex2;
 h2=chequeo(h2);
 out=h0*0x100+hl*0x10+h2;
 in=out+1;
 outin=out+2;
 do
 {
  printf("LSB = ");
  scanf("%x",&i);
  printf("MSB = ");
  scanf("%x",&j);
  outportb(outin,0x00);// *WE=1 G=1 *OE=1
  outportb(out,i);//     add LSB
  outportb(outin,0x01);// *WE=1 G=0 *OE=1
  outportb(outin,0x05);// *WE=1 G=0 *OE=0
  outportb(out,j);//     add MSB
  outportb(outin,0x01);// *WE=1 G=0 *OE=1
  outportb(outin,0x03);// *WE=1 G=0 *OE=0
  delay(tiempo);
  cout << "Desea consultar otro valor? ";
  cin >> c ;
  } while (c!='n' && c!='N');
  clrscr();
  return(0);
 }

 int chequeo(int i)
```

```
{
  if(i==0x41)i=0xa;
  if(i==0x42)i=0xb;
  if(i==0x43)i=0xc;
  if(i==0x44)i=0xd;
  if(i==0x45)i=0xe;
  if(i==0x46)i=0xf;
  if(i>=0x30 && i<=0x39)i=i-0x30;
  return(i);
}
```


PUERTO
PARALELO
PC
DB25
PULL-UP
4.7K
Vcc
74LS373
74LS373
PULL-UP
4.7K
Vcc
NVRAM
74LS04
PULL-UP
4.7K
Vcc
74LS244
Title
INTERFAZ PARA GRABACION DE NVRAM
Size
A
Document Number
1
REV

ANEXO 4

THOMSON COMPONENTS MOSTEK

MK48Z02/03/12/13(B)-12/15/20/25

2K x 8 ZEROPOWER® RAM

MEMORY COMPONENTS

FEATURES

- ☐ Predicted worst case battery life of 11 years @ 70°C
- ☐ Data retention in the absence of power
- ☐ Data security provided by automatic write protection during power failure
- ☐ Pin and functional compatibility with 2K × 8 Byte Wide Static RAMs (MK48Z02/12)
- ☐ Pin and function compatible with 2K × 8 EEPROMs (MK48Z03/13)
- ☐ +5 Volt only Read/Write
- ☐ Conventional SRAM write cycles
- ☐ Full CMOS-440 mW active; 5.5 mW standby
- ☐ 24-Pin Dual in Line package, JEDEC pinouts
- ☐ Read-cycle time equals write-cycle time
- ☐ Low-Battery Warning
- ☐ Two power-fail deselect trip points available
  MK48Z02/03 $4.75V \geq V_{PFD} \geq 4.50V$
  MK48Z12/13 $4.50V \geq V_{PFD} \geq 4.20V$

Part Number	Access Time	R/W Cycle Time
MK48ZXX-12	120 ns	120 ns
MK48ZXX-15	150 ns	150 ns
MK48ZXX-20	200 ns	200 ns
MK48ZXX-25	250 ns	250 ns

TRUTH TABLE (MK48Z02/12)

V_{CC}	$\overline{E}$	$\overline{G}$	$\overline{W}$	MODE	DQ
$<V_{CC}$ (Max) $>V_{CC}$ (Min)	V_{IH} V_{IL} V_{IL} V_{IL}	X X V_{IL} V_{IH}	X V_{IL} V_{IH} V_{IH}	Deselect Write Read Read	High-Z D_{IN} D_{OUT} High-Z
$<V_{PFD}$ (Min) $>V_{SO}$	X	X	X	Power-Fail Deselect	High-Z
$\leq V_{SO}$	X	X	X	Battery Back-up	High-Z

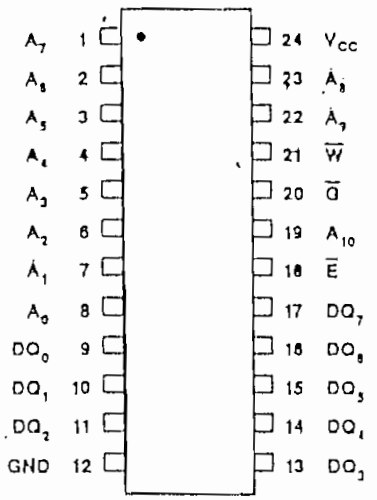


Figure 1. Pin Connections

PIN NAMES

A_0 - A_{10}	Address Inputs	V_{CC}	System Power (+5 V)
$\overline{E}$	Chip Enable	$\overline{W}$	Write Enable
GND	Ground	$\overline{G}$	Output Enable
DQ_0—DQ_7 Data In/Data Out			

TRUTH TABLE (MK48Z03/13)

V_{CC}	$\overline{E}$	$\overline{G}$	$\overline{W}$	MODE	DQ
$<V_{CC}$ (Max) $>V_{CC}$ (Min)	V_{IH} V_{IL} V_{IL} V_{IL} V_{IL}	X V_{IH} V_{IL} V_{IH} V_{IL}	X V_{IL} V_{IH} V_{IH} V_{IL}	Deselect Write Read Read Read	High-Z D_{IN} D_{OUT} High-Z High-Z
$<V_{PFD}$ (Min) $>V_{SO}$	X	X	X	Power-Fail Deselect	High-Z
$\leq V_{SO}$	X	X	X	Battery Back-up	High-Z

DESCRIPTION

The MK48Z02/03/12/13 is a 16,384-bit, Non-Volatile Static RAM, organized 2K x 8 using CMOS and an integral Lithium energy source. The ZEROPOWER™ RAM has the characteristics of a CMOS static RAM, with the important added benefit of data being retained in the absence of power. Data retention current is so small that a miniature Lithium cell contained within the package provides an energy source to preserve data. Low current drain has been attained by the use of a full CMOS memory cell, novel analog support circuitry, and carefully controlled junction leakage by an all implanted CMOS process. Safeguards against inadvertent d loss have been incorporated to maintain data integ in the uncertain operating environment associated power-up and power-down transients. The ZEROP ER RAM can replace existing 2K x 8 static RAM, dire conforming to the popular Byte Wide 24-pin DIP p age (JEDEC). MK48Z02/03/12/13 also matches the ning of 2716 EPROM and 2K x 8 EEPROM. Like o static RAMs, there is no limit to the number of write cles that can be performed. Since the access time, cycle, and write cycle are less than 250 ns and req only +5 volts, no additional support circuitry is nee for interface to a microprocessor.

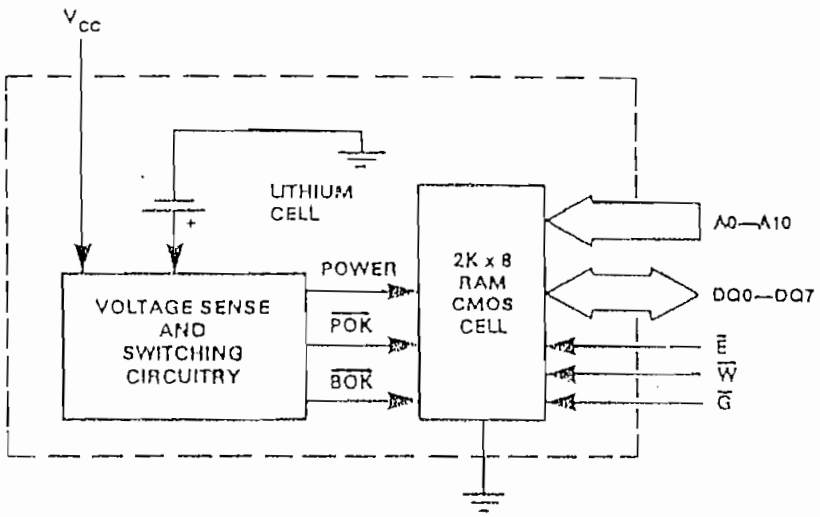


Figure 2. Block Diagram

OPERATION

Read Mode

The MK48Z02/03/12/13 is in the Read Mode whenever $\overline{W}$ (Write Enable) is high and $\overline{E}$ (Chip Enable) is low, providing a ripple-through access of data from eight of 16,384 locations in the static storage array. Thus, the unique address specified by the 11 Address Inputs (A_n) defines which one of 2,048 bytes of data is to be accessed.

Valid data will be available to the eight data Output Drivers within t_{AA} after the last address input signal is stable, providing that the $\overline{E}$ and $\overline{G}$ access times are satisfied. If $\overline{E}$ or $\overline{G}$ access times are not met, data access will be measured from the limiting parameter (t_{CEA} or t_{OEA}), rather than the address. The state of the eight Data I/O signals is controlled by the $\overline{E}$ and $\overline{G}$ control signals. The data lines may be in an indeterminate state between t_{OH} and t_{AA}, but the data lines will always have valid data at t_{AA}.

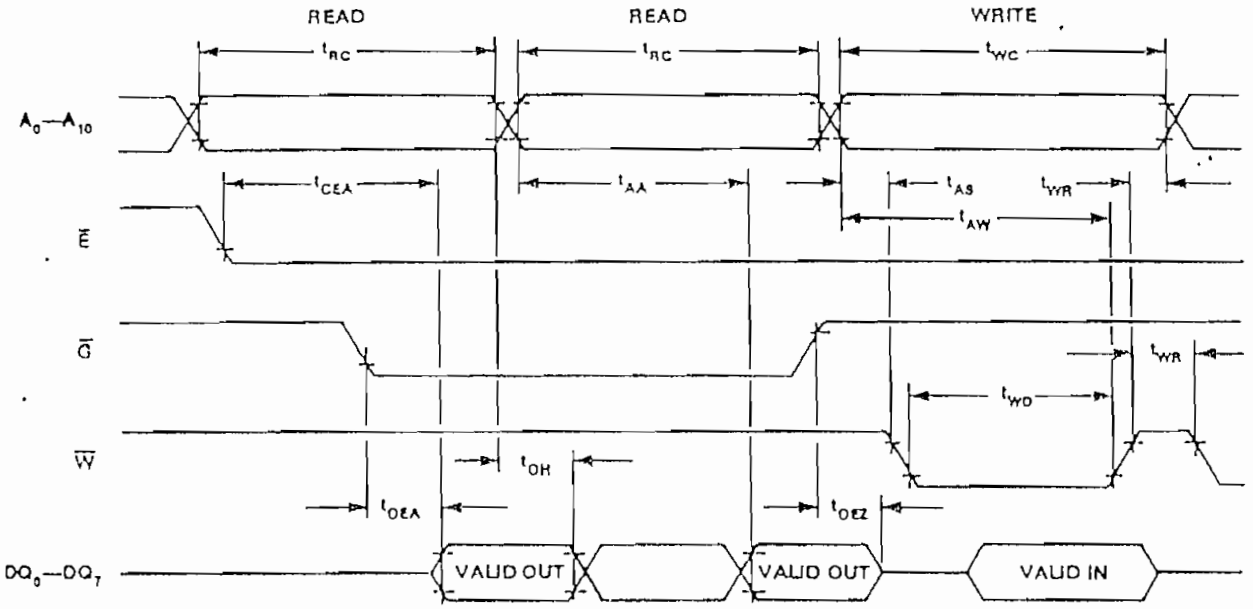


Figure 3. Read-Read-Write Timing

AC ELECTRICAL CHARACTERISTICS (READ CYCLE TIMING)

($0°C \le T_A \le 70°C$) (V_{CC} (Max) $\ge V_{CC} \ge V_{CC}$ (Min))

		MK48ZXX-12		MK48ZXX-15		MK48ZXX-20		MK48ZXX-25			
SYM	PARAMETER	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	UNITS	NOTES
t_{RC}	Read Cycle Time	120		150		200		250		ns	
t_{AA}	Address Access Time		120		150		200		250	ns	1
t_{CEA}	Chip Enable Access Time		120		150		200		250	ns	1
t_{OEA}	Output Enable Access Time		75		75		80		90	ns	1
t_{CEZ}	Chip Enable HI to High-Z		30		35		40		50	ns	
t_{OEZ}	Output Enable HI to High-Z		30		35		40		50	ns	
t_{OH}	Valid Data Out Hold Time	15		15		15		15		ns	1

NOTE

1. Measured using the Output Load Diagram shown in Figure 8.

WRITE MODE

The MK48Z02/12 is in Write Mode whenever the $\overline{W}$ and $\overline{E}$ inputs are held low. The MK48Z03/13 requires $\overline{G}$ to be held high in addition to $\overline{W}$ and $\overline{E}$ being held low. The start of a Write is referenced to the latter occurring falling edge of either W or $\overline{E}$, or the rising edge of $\overline{G}$ (MK48Z03/13). A Write is terminated by the earlier rising edge of $\overline{W}$ or $\overline{E}$, or the falling edge of $\overline{G}$ (MK48Z03/13). The addresses must be held valid throughout the cycle. $\overline{W}$ or $\overline{E}$ must return high, or $\overline{G}$ must return low (MK48Z03/13) for a minimum of t_{WR} prior to the initiation of another Read or Write Cycle. Data-in must be valid for t_{DS} prior to the End of Write and remain valid for t_{DH} afterward.

Some processors thrash producing spurious Write Cycles during power-up, despite application of a power-on reset. The MK48Z03/13 allow a user to easily overcome this problem by holding $\overline{G}$ low with the power-on reset signal. MK48Z02/12 users should force $\overline{W}$ or $\overline{E}$ high during power-up to protect memory after V_{CC} reaches V_{CC} (min) but before the processor stablizes.

The MK48Z02/12 $\overline{G}$ input is a DON'T CARE in the write mode. $\overline{G}$ can be tied low and two-wire RAM control can be implemented. A low on $\overline{W}$ will disable the outputs t_{WEZ} after $\overline{W}$ falls. Take care to avoid bus contention when operating with two-wire control.

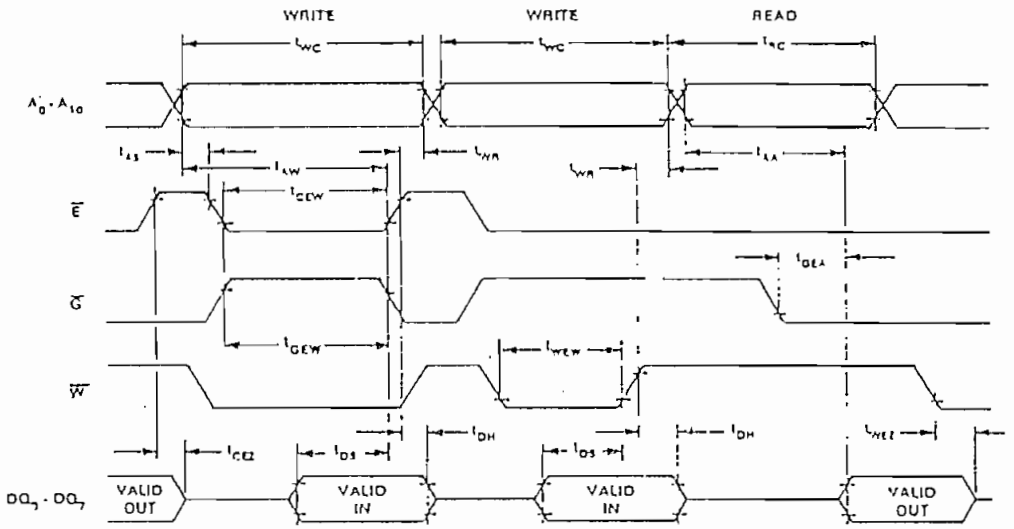


Figure 4. Write-Write-Read Timing

AC ELECTRICAL CHARACTERISTICS (WRITE CYCLE TIMING)

($0°C \le T_A \le 70°C$) (V_{CC} (Max) $\ge V_{CC} \ge V_{CC}$ (Min))

		MK48ZXX-12		MK48ZXX-15		MK48ZXX-20		MK48ZXX-25			
SYM	PARAMETER	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	UNITS	NOTES
t_{WC}	Write Cycle Time	120		150		200		250		ns	
t_{AS}	Address Setup Time	0		0		0		0		ns	
t_{AW}	Address Valid to End of Write	120		150		200		250		ns	
t_{CEW}	Chip Enable to End of Write	75		90		120		160		ns	
t_{WEW}	Write Enable to End of Write	75		90		120		160		ns	
t_{WR}	Write Recovery Time	10		10		10		10		ns	
t_{DS}	Data Setup Time	35		40		60		100		ns	
t_{DH}	Data Hold Time	0		0		0		0		ns	
t_{WEZ}	Write Enable Low to High-Z		40		50		60		80	ns	
t_{GEW}	$\overline{G}$ to End of Write	75		90		120		160		ns	1

NOTE:
1. Applies to MK48Z03/13 only.

DATA RETENTION MODE

With V_{CC} applied, the MK48Z02/12/03/13 operates as a conventional BYTEWIDE static ram. However, V_{CC} is being constantly monitored. Should the supply voltage decay, the RAM will automatically power-fail deselect, write protecting itself when V_{CC} falls within the V_{PFD} (max), V_{PFD} (min) window. The MK48Z02/03 has a V_{PFD} (max) -V_{PFD} (min) window of 4.75 volts to 4.5 volts, providing very high data security, particularly when all of the other system components are specified to 5.0 volts plus and minus 10%. The MK48Z12/13 has a V_{PFD} (max) -V_{PFD} (min) window of 4.5 volts to 4.2 volts, allowing users constrained to a 10% power supply specification to use the device.

Note: A mid-write cycle power failure may corrupt data at the current address location, but does not jeopardize the rest of the RAM's content. At voltages below V_{PFD} (min), the user can be assured the memory will be in a write protected state, provided the V_{CC} fall time does not exceed t_F. The MK48Z02/12/03/13 may respond to transient noise spikes that reach into the deselect window if they should occur during the time the device is sampling V_{CC}. Therefore decoupling of power supply lines is recommended.

The power switching circuit connects external V_{CC} to the RAM and disconnects the battery when V_{CC} rises above V_{SO}. As V_{CC} rises the battery voltage is checked. If the voltage is too low, an internal Battery Not OK ($\overline{BOK}$) flag will be set. The $\overline{BOK}$ flag can be checked after power up. If the $\overline{BOK}$ flag is set, the first write attempted will be blocked. The flag is automatically cleared after first write, and normal RAM operation resumes. Figure 5 illustrates how a $\overline{BOK}$ check routine could be structured.

Normal RAM operation can resume t_{REC} after V_{CC} exceeds V_{PFD} (Max). Caution should be taken to keep $\overline{E}$ or $\overline{W}$ high or $\overline{G}$ low (MK48Z03/13) as V_{CC} rises past V_{PFD} (Min) as some systems may perform inadvertent write cycles after V_{CC} rises but before normal system operation begins.

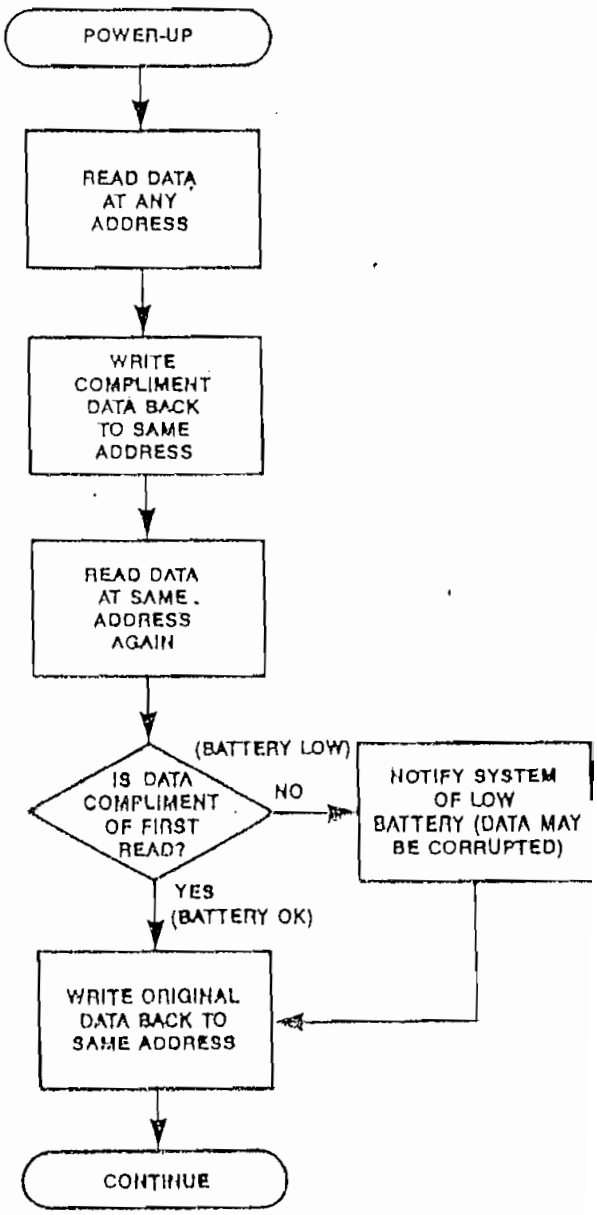


Figure 5. Checking the $\overline{BOK}$ Flag Status

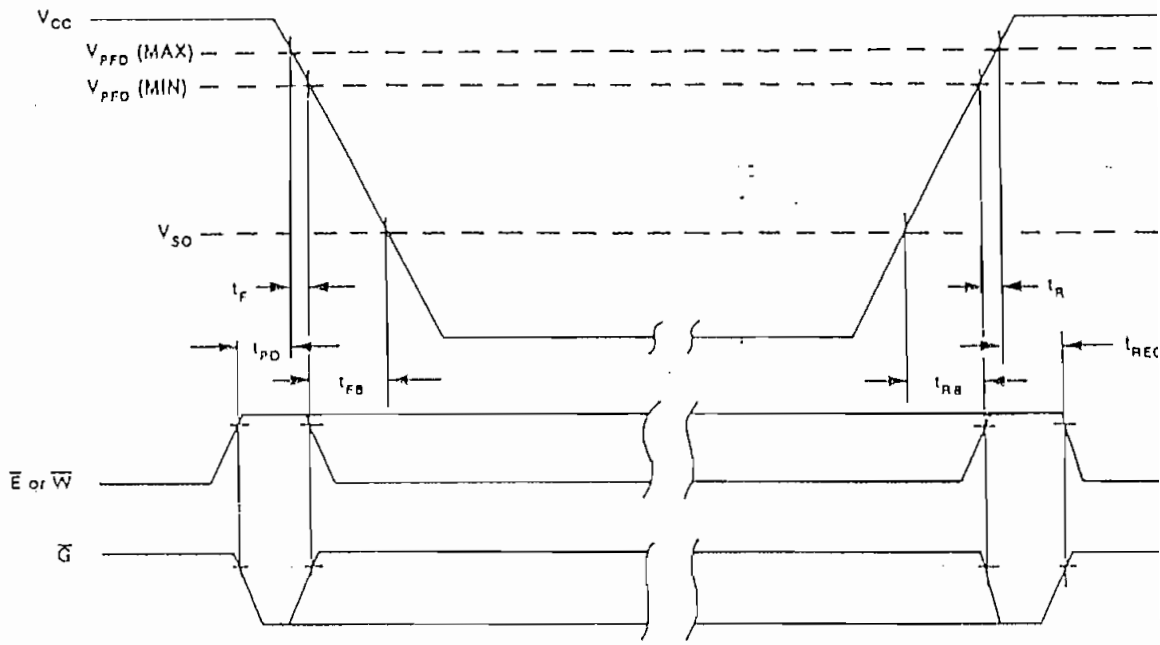


Figure 6. Power-Down/Power-Up Timing

DC ELECTRICAL CHARACTERISTICS (POWER-DOWN/POWER-UP TRIP POINT VOLTAGES)
($0°C \leq T_A \leq +70°C$)

SYM	PARAMETER	MIN	TYP	MAX	UNITS	NOT
V_{PFD}	Power-fail Deselect Voltage (MK48Z02/03)	4.50	4.6	4.75	V	
V_{PFD}	Power-fail Deselect Voltage (MK48Z12/13)	4.20	4.3	4.50	V	
V_{SO}	Battery Back-up Switchover Voltage		3		V	

AC ELECTRICAL CHARACTERISTICS (POWER-DOWN/POWER-UP TIMING)
($0°C \leq T_A \leq +70°C$)

SYM	PARAMETER	MIN	MAX	UNITS	NOT
t_{PD}	$\overline{E}$ or $\overline{W}$ at V_{IH} or $\overline{G}$ at V_{IL} before Power Down	0		ns	
t_F	V_{PFD} (Max) to V_{PFD} (Min) V_{CC} Fall Time	300		µs	
t_{FB}	V_{PFD} (Min) to V_{SO} V_{CC} Fall Time	10		µs	
t_{RB}	V_{SO} to V_{PFD} (Min) V_{CC} Rise Time	1		µs	
t_R	V_{PFD} (Min) to V_{PFD} (Max) V_{CC} Rise Time	0		µs	
t_{REC}	$\overline{E}$ or $\overline{W}$ at V_{IH} or $\overline{G}$ at V_{IL} after Power Up	2		ms	

NOTES:
1. All voltages referenced to GND.
2. V_{PFD} (Max) to V_{PFD} (Min) fall times of less t_F may result in deselection/write protection not occurring until 50 µs after V_{CC} passes V_{PFD} (Min). V_{PFD} (Max) to (Min) fall times of less than 10 µs may cause corruption of RAM data.
3. V_{PFD} (Min) to V_{SO} fall times of less than t_{FB} may cause corruption of RAM data.

CAUTION
Negative undershoots below −0.3 volts are not allowed on any pin while in Battery Back-up mode.

DATA RETENTION TIME

About Figure 7

Figure 7 illustrates how expected MK48Z02/03/12/13 battery life is influenced by temperature. The life of the battery is controlled by temperature and is virtually independent of the percentage of time the MK48Z02/03/12/13 spends in battery back-up mode.

Battery life predictions presented in Figure 7 are extrapolated from temperature accelerated life-test data collected in over 100 million device hours of continuing bare cell and encapsulated cell battery testing by Thomson - Mostek. Obviously, temperature accelerated testing cannot identify non-temperature dependent failure mechanisms. However, in view of the fact that no random cell failures have been recorded in any of Thomson - Mostek's ongoing battery testing since it began in 1982, we believe the likelihood of such failure mechanisms surfacing is extremely poor. For the purpose of this testing, a cell failure is defined as the inability of a cell stabilized at 25°C to produce a 2.0 volt closed-circuit voltage across a 250K ohm load resistance.

> A Special Note: The summary presented in Figure 7 represents a conservative analysis of the data presently available. While Thomson - Mostek is most likely in possession of the largest collection of battery life data of this kind in the world, the results presented should not be considered absolute or final; they can be expected to change as yet more data becomes available. We believe that future read-points of life tests presently under way and improvements in the battery technology itself will result in a continuing improvement of these figures.

Two end of life curves are presented in Figure 7. They are labeled "Average ($t_{50\%}$)" and "($t_{1\%}$)". These terms relate to the probability that a given number of failures will have accumulated by a particular point in time. If, for example, expected life at 70°C is at issue, Figure 7 indicates that a particular MK48Z02/03/12/13 has a 1% chance of having a battery failure 11 years into its life and a 50% chance of failure at the 12 year mark. Conversely, given a sample of devices, 1% of them can be expected to experience battery failure within 11 years; 50% of them can be expected to fail within 20 years.

The $t_{1\%}$ figure represents the practical onset of wear-out, and is therefore suitable for use in what would normally be though of as a worst-case analysis. The $t_{50\%}$ figure represents "normal" or "average" life. It is, therefore, accurate to say that the average device will last "$t_{50\%}$".

Battery life is defined as beginning on the date of manufacture. Each MK48Z02/03/12/13 is marked with a four digit manufacturing date code in the form YYWW (Example: 8502 = 1985, week 2).

Calculating Predicted Battery Life

As Figure 7 indicates, the predicted life of the battery in the MK48Z02/03/12/13 is a function of temperature. The back-up current required by the memory matrix in the MK48Z02/03/12/13 is so low that it has negligible influence on battery life.

Because predicted battery life is dependent upon application controlled variables, only the user can estimate predicted battery life in a given design. As long as ambient temperature is held reasonably constant, expected life can be read directly from Figure 7. If the MK48Z02/03/12/13 spends an appreciable amount of time at a variety of temperatures, the following equations should be used to estimate battery life.

$$\text{Predicted Battery Life} = \frac{1}{[(TA_1/TT)/BL_1)]+[(TA_2/TT)/BL_2]+\ldots+[(TA_n/TT)/BL_n)]}$$

Where TA_1, TA_2, TA_n = Time at Ambient Temperature 1, 2, etc.

TT = Total Time = $TA_1 + TA_2 + \ldots + TA_n$

BL_1, BL_2, BL_n = Predicted Battery Lifetime at Temp 1, Temp 2, etc. (see Figure 7).

EXAMPLE PREDICTED BATTERY LIFE CALCULATION

A cash register/terminal operates in an environment where the MK48Z02/03/12/13 is exposed to temperatures of 30°C (86°F) or less for 3066 hrs/yr; temperatures greater than 25°C, but less than 40°C (104°F), for 5256 hrs/yr; and temperatures greater than 40°C, but less than 70°C (158°F), for the remaining 438 hrs/yr.

Reading predicted typical life values from Figure 7; BL_1 = 456 yrs., BL_2 = 175 yrs., BL_3 = 11.4 yrs.

Total Time (TT) = 8760 hrs./yr. TA_1 = 3066 hrs./yr. TA_2 = 5256 hrs./yr. TA_3 = 438 hrs./yr.

$$\text{Predicted Typical Battery Life} \geq \frac{1}{[(3066/8760)/456]+[(5256/8760)/175]+[(438/8760)/11.4]}$$

$$\geq 116.5 \text{ yrs.}$$

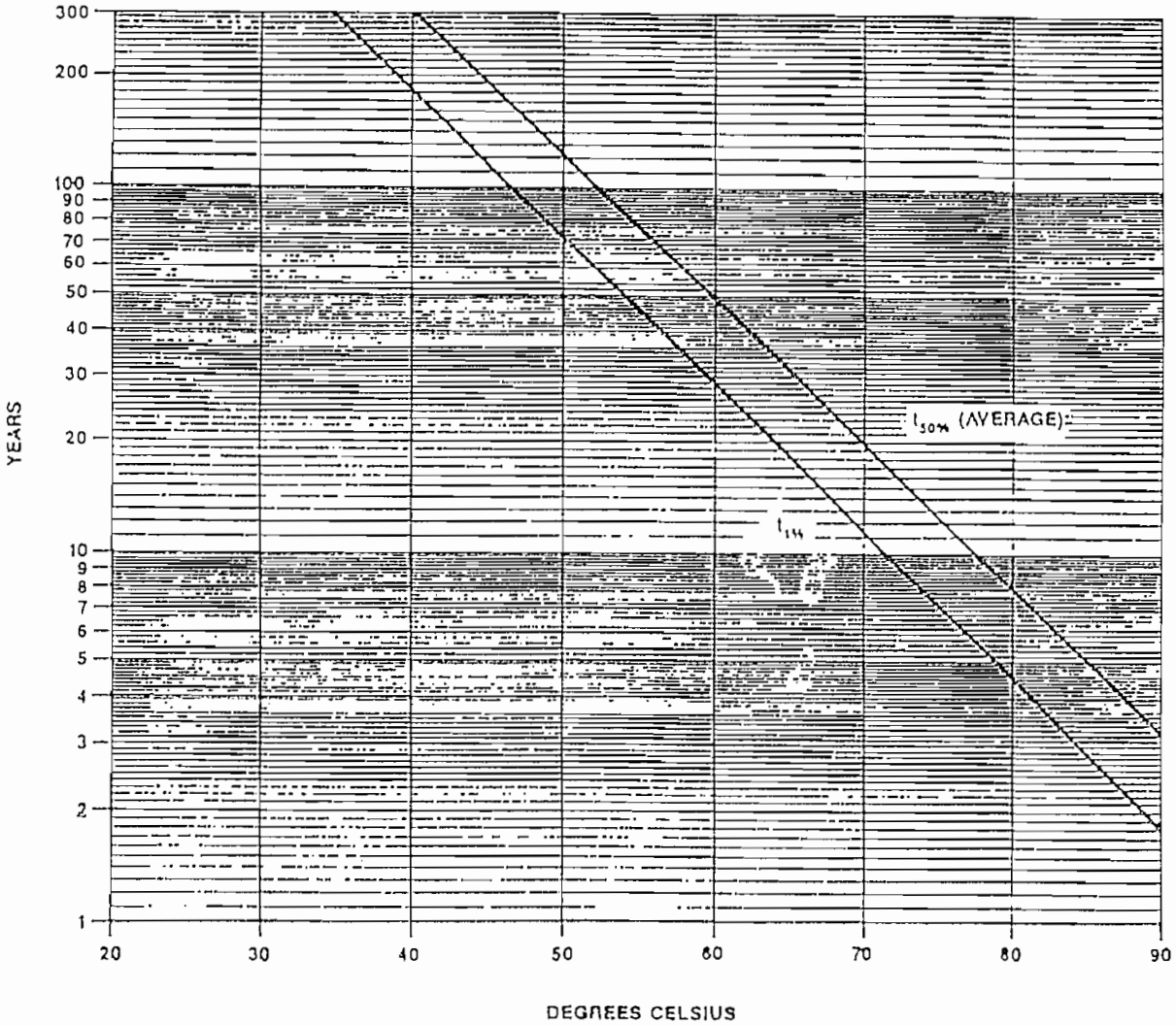


Figure 7. MK48Z02/03/12/13 Predicted Battery Storage Life vs Temperature

ABSOLUTE MAXIMUM RATINGS*

Voltage On Any Pin Relative To GND . . . . . −0.3 V to +7.0
Ambient Operating (V_{CC} On) Temperature (T_A) . . . . . 0°C to +70
Ambient Storage (V_{CC} Off) Temperature . . . . . −40°C to +85
Total Device Power Dissipation . . . . . 1 W
Output Current Per Pin . . . . . 20 r

*Stresses greater than those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functio operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to abso maximum rating conditions for extended periods of time may affect reliability.

CAUTION: Under no conditions can the "Absolute Maximum Rating" for the voltage on any pin be exceeded since it will cause permanent damage. Specific do not perform the "standard" continuity test on any input or output pin, i.e do not force these pins below −0.3 V DC.

RECOMMENDED DC OPERATING CONDITIONS
($0°C \le T_A \le 70°C$)

SYM	PARAMETER	MIN	MAX	UNITS	NOTES
V_{CC}	Supply Voltage (MK48Z02/03)	4.75	5.50	V	1
V_{CC}	Supply Voltage (MK48Z12/13)	4.50	5.50	V	1
GND	Supply Voltage	0	0	V	1
V_{IH}	Logic "1" Voltage All Inputs	2.2	V_{CC} + 0.3 V	V	1
V_{IL}	Logic "0" Voltage All Inputs	−0.3	0.8	V	1,2

DC ELECTRICAL CHARACTERISTICS
($0°C \le T_A \le +70°C$) (V_{CC} (max) $\ge V_{CC} \ge V_{CC}$ (min))

SYM	PARAMETER	MIN	MAX	UNITS	NOTES
I_{CC1}	Average V_{CC} Power Supply Current		80	mA	3
I_{CC2}	TTL Standby Current ($\overline{E}$ = V_{IH})		3	mA	
I_{CC3}	CMOS Standby Current ($\overline{E} \ge V_{CC}$ −0.2 V)		1	mA	
I_{IL}	Input Leakage Current (Any Input)	−1	+1	μA	4
I_{OL}	Output Leakage Current	−5	+5	μA	4
V_{OH}	Output Logic "1" Voltage (I_{OUT} = −1.0 mA)	2.4		V	
V_{OL}	Output Logic "0" Voltage (I_{OUT} = 2.1 mA)		0.4	V	

CAPACITANCE (T_A = 25°C)

SYM	PARAMETER	MAX	NOTE
C_I	Capacitance on all pins (except D/Q)	7 pF	5
$C_{D/Q}$	Capacitance on D/Q pins	10 pF	4,5

NOTES
1. All voltages referenced to GND.
2. Negative spikes of −1.0 volts allowed for up to 10 ns once per cycle.
3. I_{CC1} measured with outputs open.
4. Measured with GND $\le V_I \le V_{CC}$ and outputs deselected.
5. Effective capacitance calculated from the equation $C = \frac{|\Delta I|}{\Delta V}$ with ΔV = 3 volts and power supply at nominal level.

AC TEST CONDITIONS

Input Levels:	0.6 V to 2.4 V
Transition Times:	5 ns
Input and Output Timing Reference Levels	0.8 V or 2.2 V
Ambient Temperature	0°C to 70°C
V_{CC} (MK48Z02/03)	4.75 V to 5.5 V
V_{CC} (MK48Z12/13)	4.5 V to 5.5 V

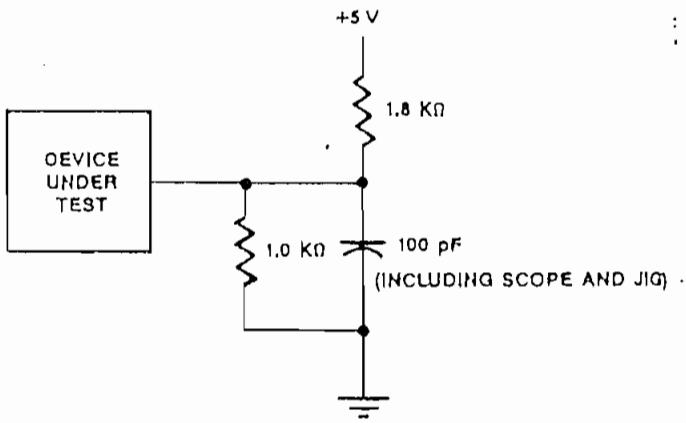


Figure 8. Output Load Diagram

ORDERING INFORMATION

MK48Z X X B -XX

DEVICE FAMILY

V_{CC} RANGE
- 0 +10%/−5%
- 1 +10%/−10%

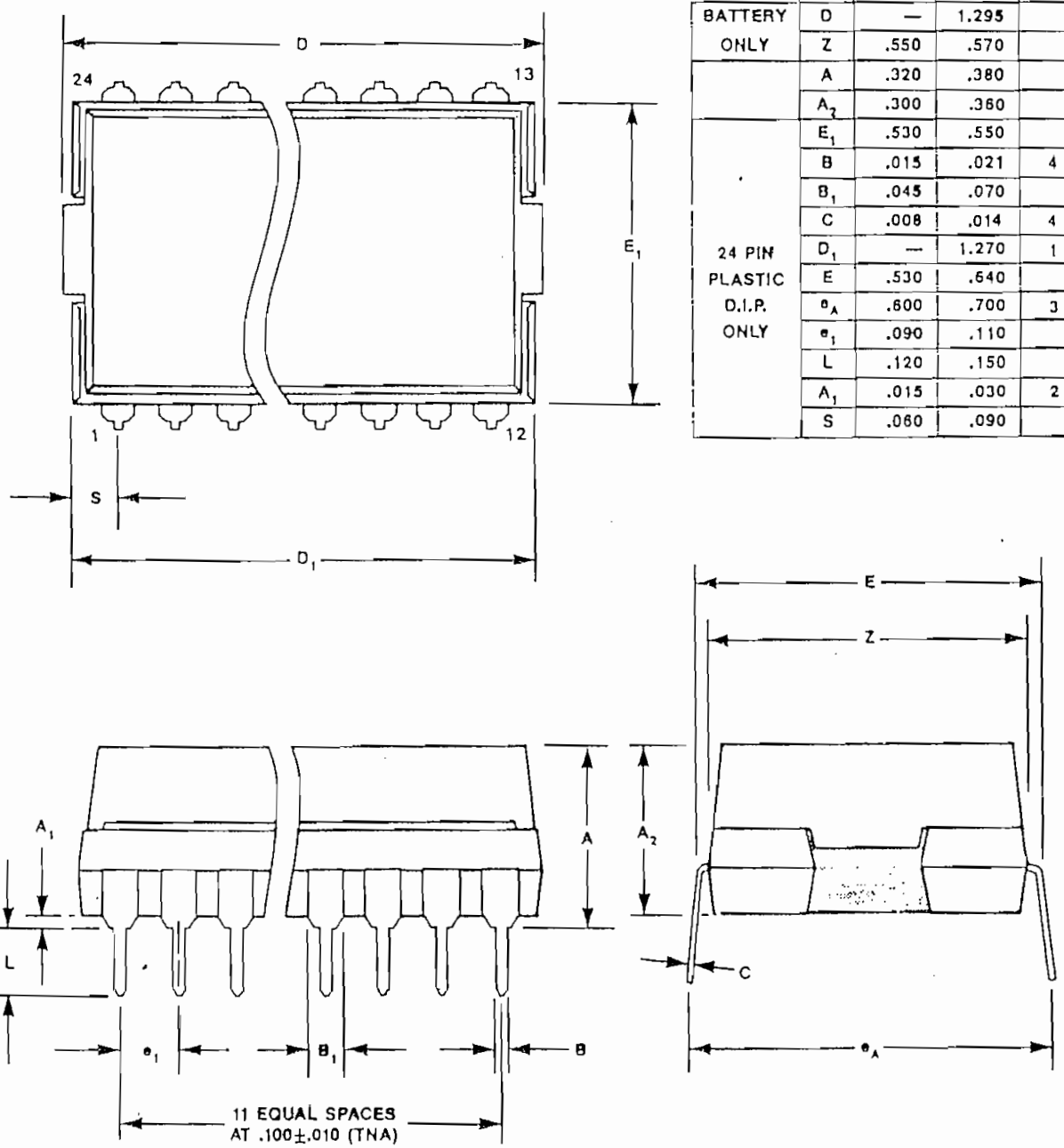
$\overline{G}$ FUNCTION
- 2 $\overline{G}$ DON'T CARE ON WRITE
- 3 $\overline{G}$ HIGH FOR WRITE

PACKAGE
- B PLASTIC WITH BATTERY TOP HAT

SPEED
- -12 120 NS ACCESS TIME
- -15 150 NS ACCESS TIME
- -20 200 NS ACCESS TIME
- -25 250 NS ACCESS TIME

PACKAGE DESCRIPTION
8 Package
24 Pin

	DIM.	INCHES MIN	INCHES MAX	NOTES
BATTERY ONLY	D	—	1.295	
	Z	.550	.570	
	A	.320	.380	
	A_2	.300	.360	
	E_1	.530	.550	
	B	.015	.021	4
	B_1	.045	.070	
	C	.008	.014	4
24 PIN PLASTIC D.I.P. ONLY	D_1	—	1.270	1
	E	.530	.640	
	e_A	.600	.700	3
	e_1	.090	.110	
	L	.120	.150	
	A_1	.015	.030	2
	S	.060	.090	



NOTES:
1. Overall length includes .010 in. flash on either end of the package.
2. Package standoff to be measured per JEDEC requirements.
3. Measured from centerline to centerline at lead tips.
4. When the solder lead finish is specified, the maximum limit shall be increased by .003 in.

DMC16207

• Display Format (16 character × 2 line) • Display Fonts(5×8dots) • Driving Method(1/16D)

ABSOLUTE MAXIMUM RATINGS

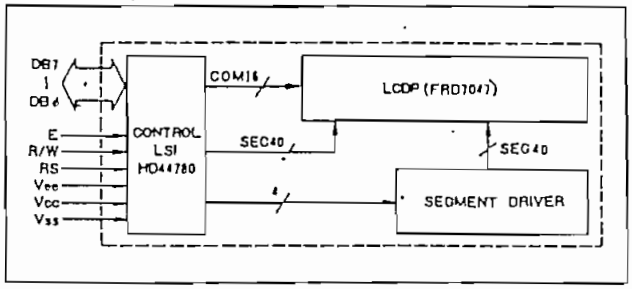
Item	Symbol	Test Condition	Standard Value			Unit
			min.	typ.	max.	
Power Supply Voltage for Logic	Vcc–Vss	—	0	—	7	V
Power Supply Voltage for LCD Drive	Vcc–Vee	—	0	—	13.5	V
Input Voltage	VI	—	Vss	—	Vcc	V
Operating Temperature	Ta	—	0	—	+50	℃
Storage Temperature	Tstg	—	−20	—	+70	℃

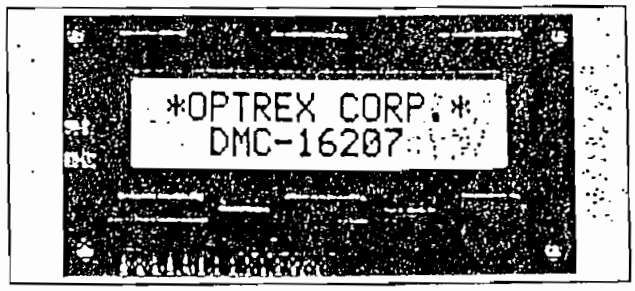
ELECTRICAL CHARACTERISTICS

Item	Symbol	Test Condition	Standard Value			Unit
			min.	typ.	max.	
Input "High" Voltage	VIH	—	2.2	—	Vcc	V
Input "Low" Voltage	VIL	—	−0.3	—	0.6	V
Output "High" Voltage	VOH	−IOH=0.205mA	2.4	—	—	V
Output "Low" Voltage	VOL	IOH=1.2mA	—	—	0.4	V
Power Supply Current	Icc	Vcc=5.0V	—	0.5	2.0	mA

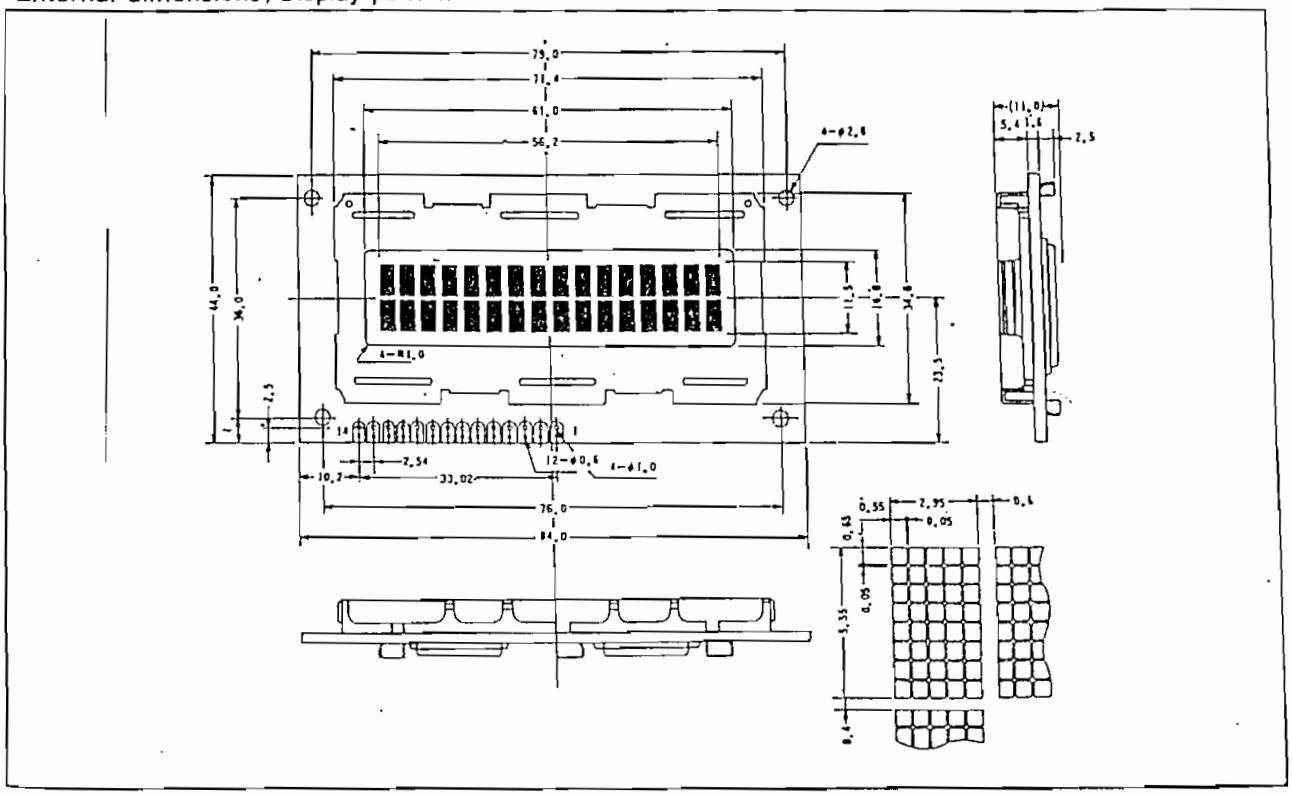
※Vcc=5.0V±5%, Ta=25℃

Block diagram

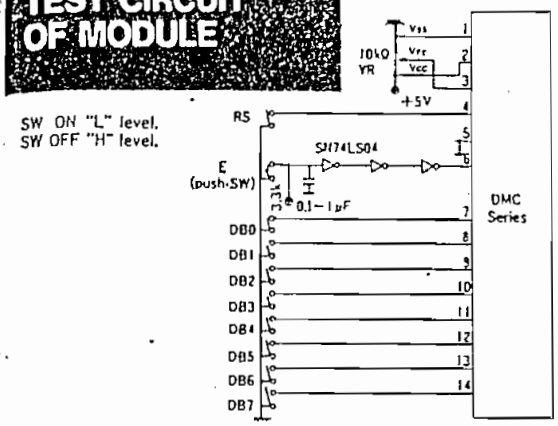




External dimensions / Display pattern

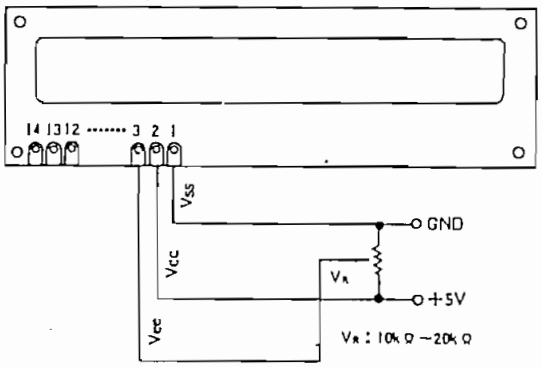


TEST CIRCUIT OF MODULE

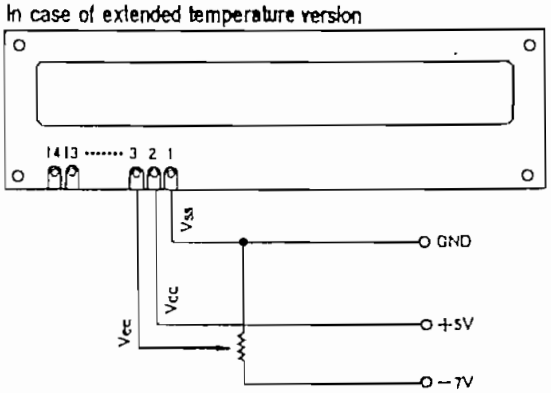


EXAMPLE OF POWER SUPPLY

DMC Module



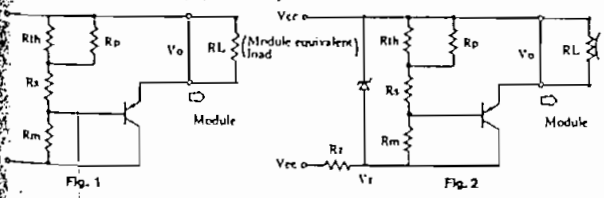
In case of extended temperature version



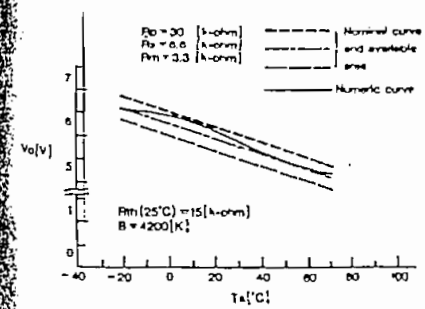
*NOTE:When the voltage of Vee is different from the recommended voltage, the viewing angle may be changed.

●Examples of Temperature Compensation Circuits for Extended Temp Type.(Only for reference)

(A) 1/8Duty—1/4Bias

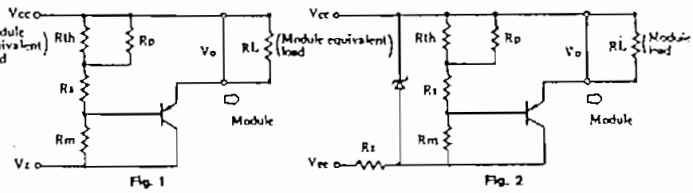


Thermistor:Rth(25℃) = 15[k-ohm], B = 4200[K]
Resistors :Rp = 30[k-ohm], Rs = 6.8[k-ohm], Rm = 3.3[k-ohm]
Transistor :PNP Type
Vcc:+5v, Vss:0v (Logic Supply)
Vz :−8[V](−7.8 to −8.2[V])
Vee<Vz[V], Rz = (Vz−Vee)/5[k-ohm]

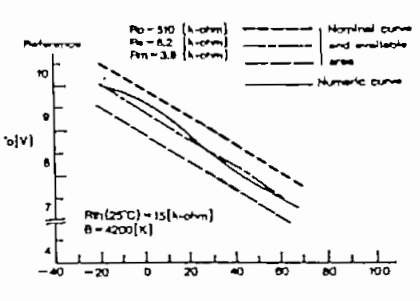


Ta[℃]	Vo[V]
−20	6.56
−10	6.50
0	6.40
10	6.26
20	6.09
30	5.88
40	5.67
50	5.47
60	5.29
70	5.15

(B) 1/16Duty—1/5Bias



Thermistor:Rth(25℃) = 15[k-ohm], B = 4200[K]
Resistors :Rp = 510[k-ohm], Rs = 8.2[k-ohm], Rm = 3.9[k-ohm]
Transistor :PNP Type
Vcc:+5v, Vss:0v (Logic Supply)
Vz :−11[V](−10.725 to −11.275[V])
Vee<Vz[V], Rz = (Vz−Vee)/5[k-ohm]



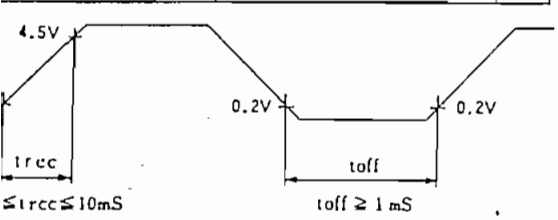
Ta[℃]	Vo[V]
−20	10.01
−10	9.84
0	9.60
10	9.28
20	8.88
30	8.49
40	8.11
50	7.79
60	7.53
70	7.33

*Specifications are subject to change without notice.

OWER SUPPLY RESET

internal reset circuit will not be correctly operated, the following power supply condition is not satisfied. his case, please perform initial setting according to nstruction.

Item	Symbol	Measuring Condition	Standard Value min.	typ.	max.	Unit
Supply Rise Time	trcc	———	0.1	—	10	mS
Supply OFF Time	toff	———	1	—	—	mS



≤trcc≤10mS toff ≥ 1 mS

item toff defines the time when the power supply is off, when the power ply shuts down momentarily or repeats on-off state.

FUNCTION

alizing by Internal Reset Circuit

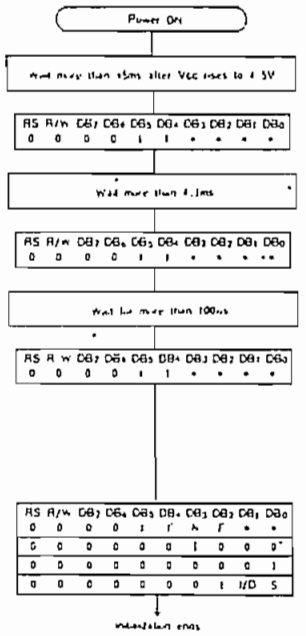
D44780 automatically initializes (resets) when power is turned ng the internal reset circuit. The following instructions are ted in initialization. The busy flag (BF) is kept in busy state itialization ends. (BF=1) The busy state is 10ms after Vcc o 4.5V.

- play clear
- ction ser
- =1 : 8bit long interface data
- 0 : 1-line display F=0 : 5×7dot character font
- lay ON/OFF control
- =0 : Display OFF C=0 : Cursor OFF B=0 : Blink OFF
- mode set
- D=1 : +1(increment) S=0 : No shift

n conditions in "Power Supply Conditions Using Internal Reset Circuit" not met, the internal reset circuit with not operate normally and initialization not be performed. In this case initialize by MPU according to "Initializing by uction".

● Initializing by Instruction

If the power supply conditions for correctly operating the internal reset circuit are not met, initialization by instruction is required. Use the following procedure for initialization.



INSTRUCTIONS

Instruction	Code										Description
	RS	R/W	DB 7	DB 6	DB 5	DB 4	DB 3	DB 2	DB 1	DB 0	
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0).
Cursor At Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (Address 0). Also returns the display being shifted to the original position. DDRAM contents remain unchanged.
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.
Display On/Off Control	0	0	0	0	0	0	1	0	C	B	Sets ON/OFF of all display (D) cursor ON /OFF (C), and blink of cursor position character (B).
Cursor/Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DDRAM contents.
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL) number of display lines (L) and character font (F).
CGRAM Address Set	0	0	0	1	ACG						Sets the CGRAM address. CGRAM data is sent and received after this setting.
DDRAM Address Set	0	0	1	ADD							Sets the DDRAM address. DDRAM data is sent and received after this setting
Busy Flag/ Address Read	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.
CGRAM/DDRAM Data Write	1	0	WRITE DATA								Writes data into DDRAM or CGRAM.
CGRAM/DDRAM Data Read	1	1	READ DATA								Reads data from DDRAM or CGRAM.

Code	Description	Execute Time (max.)
I/D =1 : Increment I/D =0 : Decrement S =1 : With display shift S/C=1 : Display shift S/C=0 : Cursor movement R/L=1 : Shift to the right R/L=0 : Shift to the left DL =1 : 8-bit DL =0 : 4-bit N =1 : 1/16Duty N =0 : 1/8Duty, 1/11Duty F =1 : 5×10dots F =0 : 5×7dots BF =1 : Internal operation is being performed BF =0 : Instruction acceptable	DDRAM : Display Data RAM CGRAM : Character Generator RAM ACG : CGRAM Address ADD : DDRAM Address Corresponds to cursor address. AC : Address Counter, used for both DDRAM and CGRAM ※ : Invalid	fcp or fosc=250kHz However, when frequency cha. execution time also changes Ex When fcp or fosc=270kHz, $40\mu S \times \frac{250}{270} = 37\mu S$

FONT TABLE (5 × 11bits)

Lower 4-bit \ Upper 4-bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
××××0000	CG RAM (1)												
××××0001	(2)												
××××0010	(3)												
××××0011	(4)												
××××0100	(5)												
××××0101	(6)												
××××0110	(7)												
××××0111	(8)												
××××1000	(1)												
××××1001	(2)												
××××1010	(3)												
××××1011	(4)												
××××1100	(5)												
××××1101	(6)												
××××1110	(7)												
××××1111	(8)												

3 RAM : Character pattern area can be rewritten by program.

PRELIMINARY

MCS®-51
8-BIT CONTROL-ORIENTED MICROCOMPUTERS
8031/8051
8031AH/8051AH
8032AH/8052AH
8751H/8751H-8

- High Performance HMOS Process
- Internal Timers/Event Counters
- 2-Level Interrupt Priority Structure
- 32 I/O Lines (Four 8-Bit Ports)
- 64K Program Memory Space
- Security Feature Protects EPROM Parts Against Software Piracy
- Boolean Processor
- Bit-Addressable RAM
- Programmable Full Duplex Serial Channel
- 111 Instructions (64 Single-Cycle)
- 64K Data Memory Space

The MCS®-51 products are optimized for control applications. Byte-processing and numerical operations on small data structures are facilitated by a variety of fast addressing modes for accessing the internal RAM. The instruction set provides a convenient menu of 8-bit arithmetic instructions, including multiply and divide instructions. Extensive on-chip support is provided for one-bit variables as a separate data type, allowing direct bit manipulation and testing in control and logic systems that require Boolean processing.

The 8051 is the original member of the MCS-51 family. The 8051AH is identical to the 8051, but it is fabricated with HMOS II technology.

The 8751H is an EPROM version of the 8051AH; that is, the on-chip Program Memory can be electrically programmed, and can be erased by exposure to ultraviolet light. It is fully compatible with its predecessor, the 8751-8, but incorporates two new features: a Program Memory Security bit that can be used to protect the EPROM against unauthorized read-out, and a programmable baud rate modification bit (SMOD). The 8751H-8 is identical to the 8751H but only operates up to 8 MHz.

The 8052AH is an enhanced version of the 8051AH. It is backwards compatible with the 8051AH and is fabricated with HMOS II technology. The 8052AH enhancements are listed in the table below. Also refer to this table for the ROM, ROMless, and EPROM versions of each product.

Device	Internal Memory		Timers/ Event Counters	Interrupts
	Program	Data		
8052AH	8K x 8 ROM	256 x 8 RAM	3 x 16-Bit	6
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8051	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8032AH	none	256 x 8 RAM	3 x 16-Bit	6
8031AH	none	128 x 8 RAM	2 x 16-Bit	5
8031	none	128 x 8 RAM	2 x 16-Bit	5
8751H	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5
8751H-8	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5

October 1988
Order Number: 270048-004

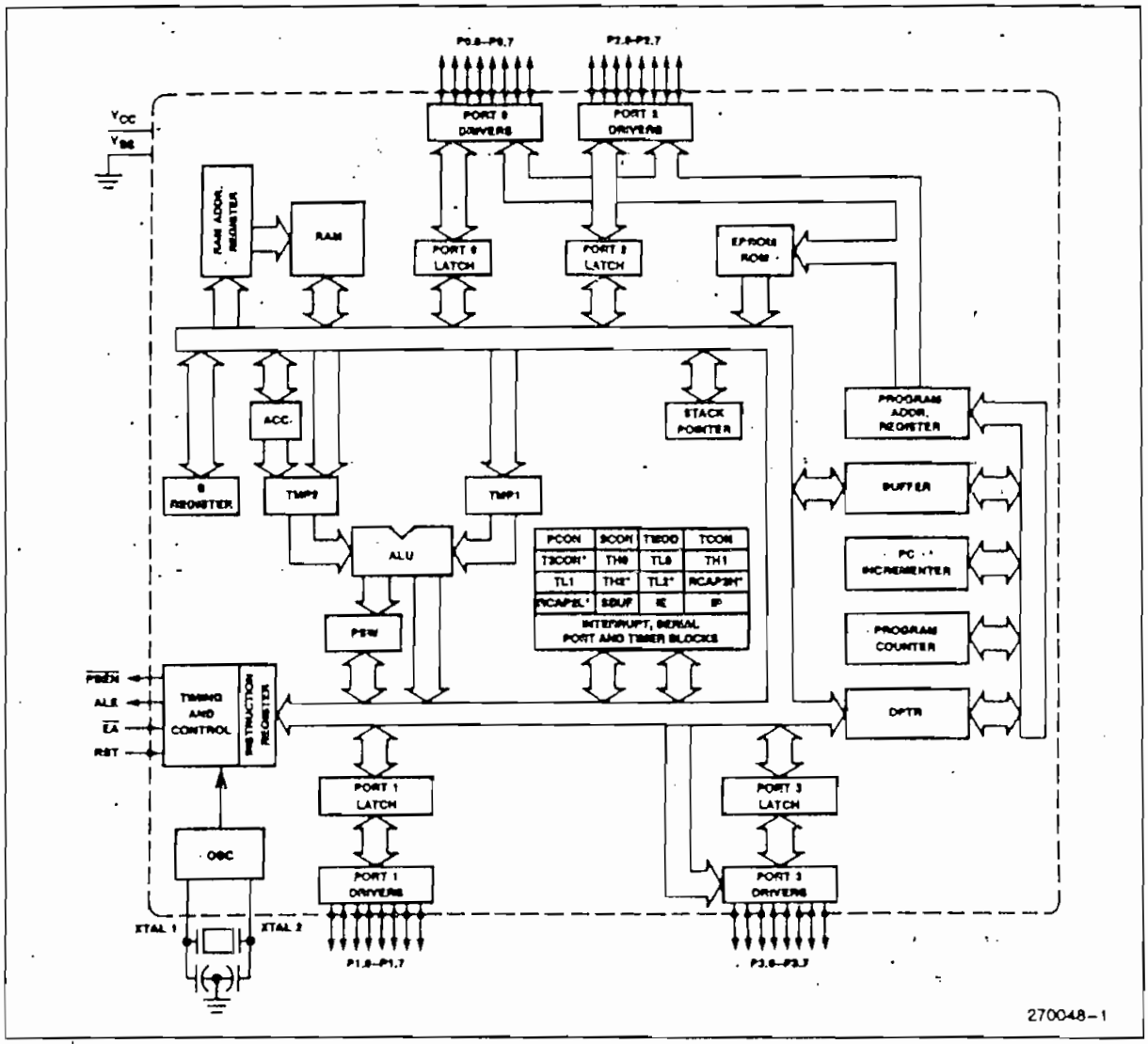


Figure 1. MCS®-51 Block Diagram

PACKAGES

Part	Prefix	Package Type
8051AH/ 8031AH	P	40-Pin Plastic DIP
	D	40-Pin CERDIP
	N	44-Pin PLCC
8052AH/ 8032AH	P	40-Pin Plastic DIP
	D	40-Pin CERDIP
	N	44-Pin PLCC
8751H/ 8751H-8	D	40-Pin CERDIP
	R	44-Pin LCC

PIN DESCRIPTIONS

V_{CC}: Supply voltage.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs.

Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s and can source and sink 8 LS TTL inputs.

Port 0 also receives the code bytes during programming of the EPROM parts, and outputs the code bytes during program verification of the ROM and EPROM parts. External pullups are required during program verification.

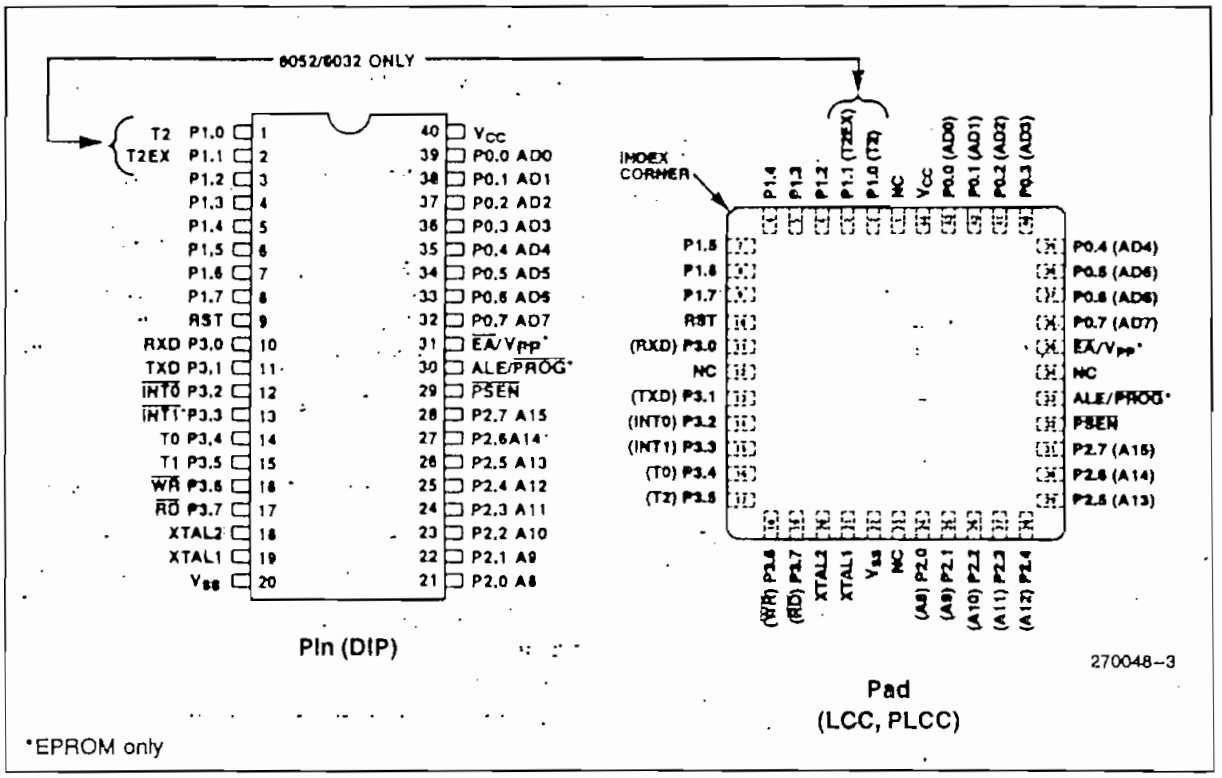


Figure 2. MCS®-51 Connections

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL inputs. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during programming of the EPROM parts and during program verification of the ROM and EPROM parts.

In the 8032AH and 8052AH, Port 1 pins P1.0 and P1.1 also serve the T2 and T2EX functions, respectively.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits during programming of the EPROM parts and during program verification of the ROM and EPROM parts.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternative Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during programming of the EPROM parts.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

$\overline{PSEN}$: Program Store Enable is the read strobe to external Program Memory.

When the device is executing code from external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external Data Memory.

$\overline{EA}$/V_{PP}: External Access enable $\overline{EA}$ must be strapped to V_{SS} in order to enable any MCS-51 device to fetch code from external Program memory locations starting at 0000H up to FFFFH. $\overline{EA}$ must be strapped to V_{CC} for internal program execution.

Note, however, that if the Security Bit in the EPROM devices is programmed, the device will not fetch code from any location in external Program Memory.

This pin also receives the 21V programming supply voltage (VPP) during programming of the EPROM parts.

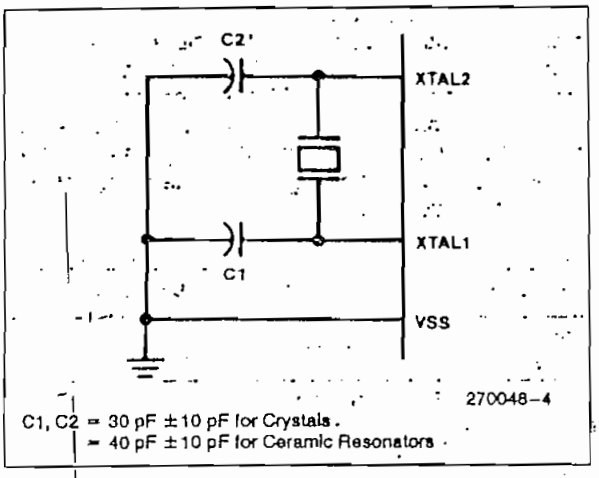


Figure 3. Oscillator Connections

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

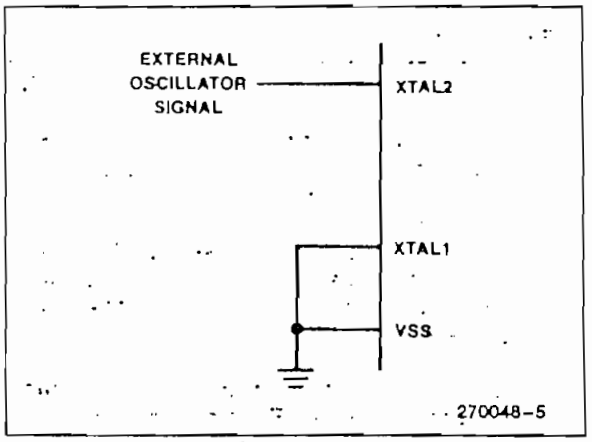


Figure 4. External Drive Configuration

DESIGN CONSIDERATIONS

If an 8751BH or 8752BH may replace an 8751H in a future design, the user should carefully compare both data sheets for DC or AC Characteristic differences. Note that the V_{IH} and I_{IH} specifications for the $\overline{EA}$ pin differ significantly between the devices.

Exposure to light when the EPROM device is in operation may cause logic errors. For this reason, it is suggested that an opaque label be placed over the window when the die is exposed to ambient light.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on $\overline{EA}/V_{PP}$ Pin to V_{SS} ...−0.5V to +21.5V
Voltage on Any Other Pin to V_{SS} ....−0.5V to +7V
Power Dissipation.....1.5W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and 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.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C; V_{CC} = 5V ±10%; V_{SS} = 0V

Symbol	Parameter		Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage (Except $\overline{EA}$ Pin of 8751H & 8751H-8)		−0.5	0.8	V	
V_{IL1}	Input Low Voltage to $\overline{EA}$ Pin of 8751H & 8751H-8		0	0.7	V	
V_{IH}	Input High Voltage (Except XTAL2, RST)		2.0	V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage to XTAL2, RST		2.5	V_{CC} + 0.5	V	XTAL1 = V_{SS}
V_{OL}	Output Low Voltage (Ports 1, 2, 3)*			0.45	V	I_{OL} = 1.6 mA
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{PSEN}$)*					
		8751H, 8751H-8		0.60 0.45	V V	I_{OL} = 3.2 mA I_{OL} = 2.4 mA
		All Others		0.45	V	I_{OL} = 3.2 mA
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, $\overline{PSEN}$)		2.4		V	I_{OH} = −80 µA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)		2.4		V	I_{OH} = −400 µA
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3, RST) 8032AH, 8052AH All Others			−800 −500	µA µA	V_{IN} = 0.45V V_{IN} = 0.45V
I_{IL1}	Logical 0 Input Current to $\overline{EA}$ Pin of 8751H & 8751H-8 Only			−15	mA	V_{IN} = 0.45V
I_{IL2}	Logical 0 Input Current (XTAL2)			−3.2	mA	V_{IN} = 0.45V
I_{LI}	Input Leakage Current (Port 0) 8751H & 8751H-8 All Others			 ±100 ±10	 µA µA	 0.45 ≤ V_{IN} ≤ V_{CC} 0.45 ≤ V_{IN} ≤ V_{CC}
I_{IH}	Logical 1 Input Current to $\overline{EA}$ Pin of 8751H & 8751H-8			500	µA	V_{IN} = 2.4V
I_{IH1}	Input Current to RST to Activate Reset			500	µA	V_{IN} < (V_{CC} − 1.5V)
I_{CC}	Power Supply Current: 8031/8051 8031AH/8051AH 8032AH/8052AH 8751H/8751H-8			 160 125 175 250	 mA mA mA mA	 All Outputs Disconnected; $\overline{EA}$ = V_{CC}
C_{IO}	Pin Capacitance			10	pF	Test freq = 1 MHz

***NOTE:**
Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

A.C. CHARACTERISTICS

T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V;
Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF;
Load Capacitance for All Other Outputs = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL−40		ns
TLLAX	Address Hold after ALE Low	48		TCLCL−35		ns
TLLIV	ALE Low to Valid Instr In 8751H All Others		 183 233		 4TCLCL−150 4TCLCL−100	 ns ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	58		TCLCL−25		ns
TPLPH	$\overline{PSEN}$ Pulse Width 8751H All Others	 190 215		 3TCLCL−60 3TCLCL−35		 ns ns
TPLIV	$\overline{PSEN}$ Low to Valid Instr In 8751H All Others		 100 125		 3TCLCL−150 3TCLCL−125	 ns ns
TPXIX	Input Instr Hold after $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float after $\overline{PSEN}$		63		TCLCL−20	ns
TPXAV	$\overline{PSEN}$ to Address Valid	75		TCLCL−8		ns
TAVIV	Address to Valid Instr In 8751H All Others		 267 302		 5TCLCL−150 5TCLCL−115	 ns ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		20		20	ns
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL−100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL−100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		252		5TCLCL−165	ns
TRHDX	Data Hold after $\overline{RD}$	0		0		ns
TRHDZ	Data Float after $\overline{RD}$		97		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL−150	ns
TAVDV	Address to Valid Data In		585		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition 8751H All Others	 13 23		 TCLCL−70 TCLCL−60		 ns ns
TQVWH	Data Valid to $\overline{WR}$ High	433		7TCLCL−150		ns
TWHQX	Data Hold after $\overline{WR}$	33		TCLCL−50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		20		20	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High 8751H All Others	 33 43	 133 123	 TCLCL−50 TCLCL−40	 TCLCL+50 TCLCL+40	 ns ns

NOTE:
*This table does not include the 8751-8 A.C. characteristics (see next page).

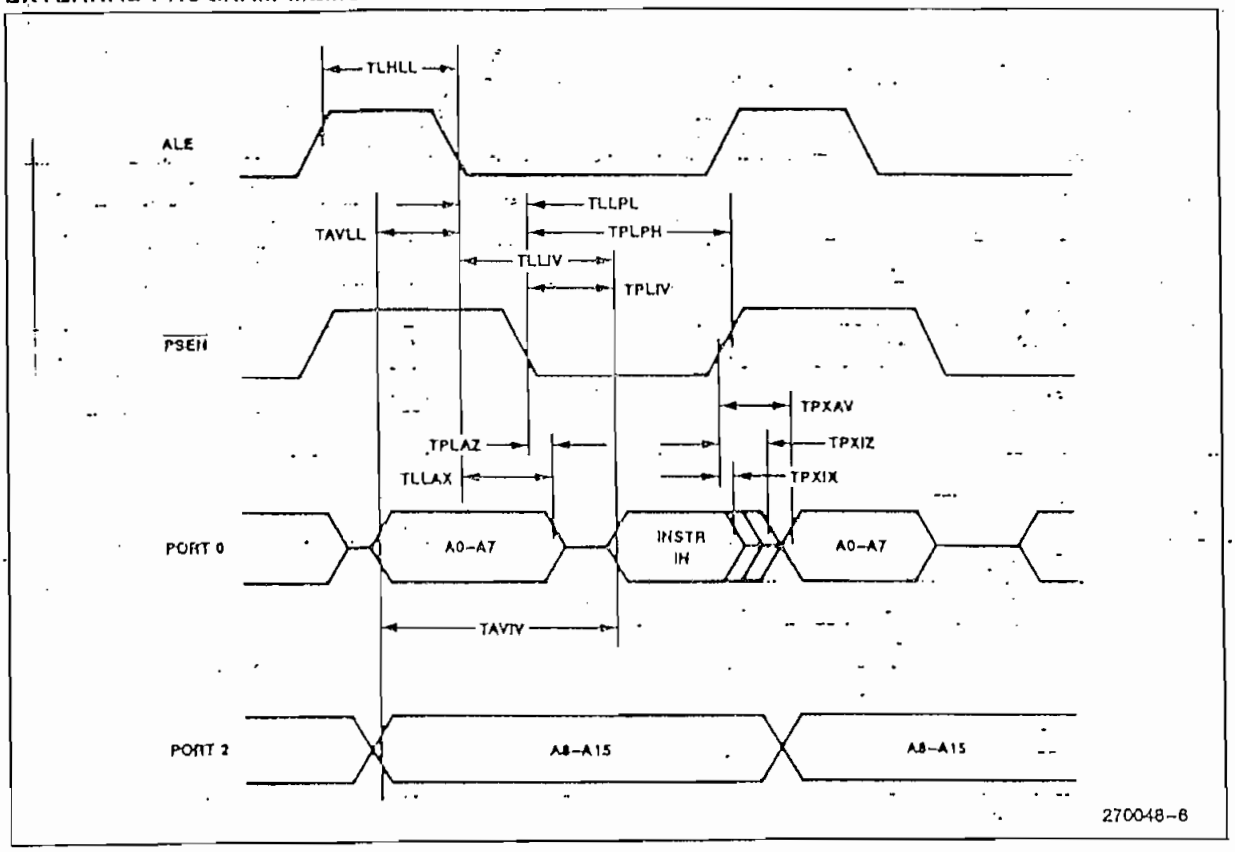
This Table is only for the 8751H-8

A.C. CHARACTERISTICS

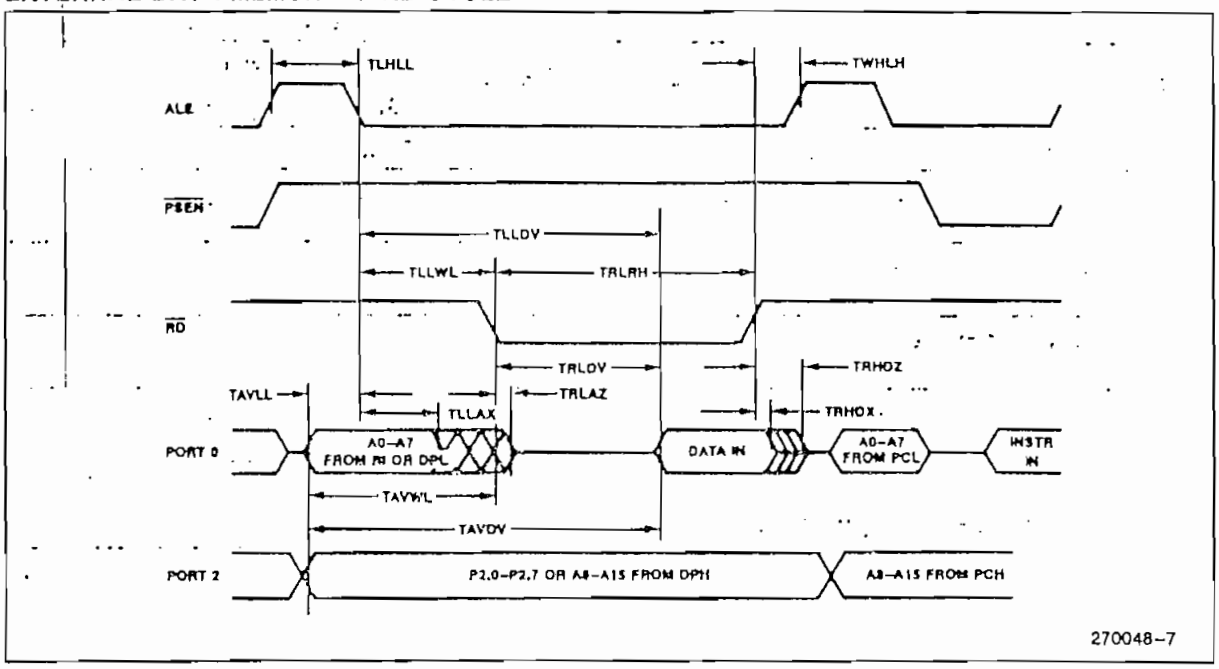
T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V;
Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF;
Load Capacitance for All Other Outputs = 80 pF

Symbol	Parameter	8 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	8.0	MHz
TLHLL	ALE Pulse Width	210		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	85		TCLCL−40		ns
TLLAX	Address Hold after ALE Low	90		TCLCL−35		ns
TLLIV	ALE Low to Valid Instr In		350		4TCLCL−150	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	100		TCLCL−25		ns
TPLPH	$\overline{PSEN}$ Pulse Width	315		3TCLCL−60		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instr In		225		3TCLCL−150	ns
TPXIX	Input Instr Hold after $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float after $\overline{PSEN}$		105		TCLCL−20	ns
TPXAV	$\overline{PSEN}$ to Address Valid	117		TCLCL−8		ns
TAVIV	Address to Valid Instr In		475		5TCLCL−150	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		20		20	ns
TRLRH	$\overline{RD}$ Pulse Width	650		6TCLCL−100		ns
TWLWH	$\overline{WR}$ Pulse Width	650		6TCLCL−100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		460		5TCLCL−165	ns
TRHDX	Data Hold after $\overline{RD}$	0		0		ns
TRHDZ	Data Float after $\overline{RD}$		180		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		850		8TCLCL−150	ns
TAVDV	Address to Valid Data In		960		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	325	425	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	370		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition	55		TCLCL−70		ns
TQVWH	Data Valid to $\overline{WR}$ High	725		7TCLCL−150		ns
TWHQX	Data Hold after $\overline{WR}$	75		TCLCL−50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		20		20	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	75	175	TCLCL−50	TCLCL+50	ns

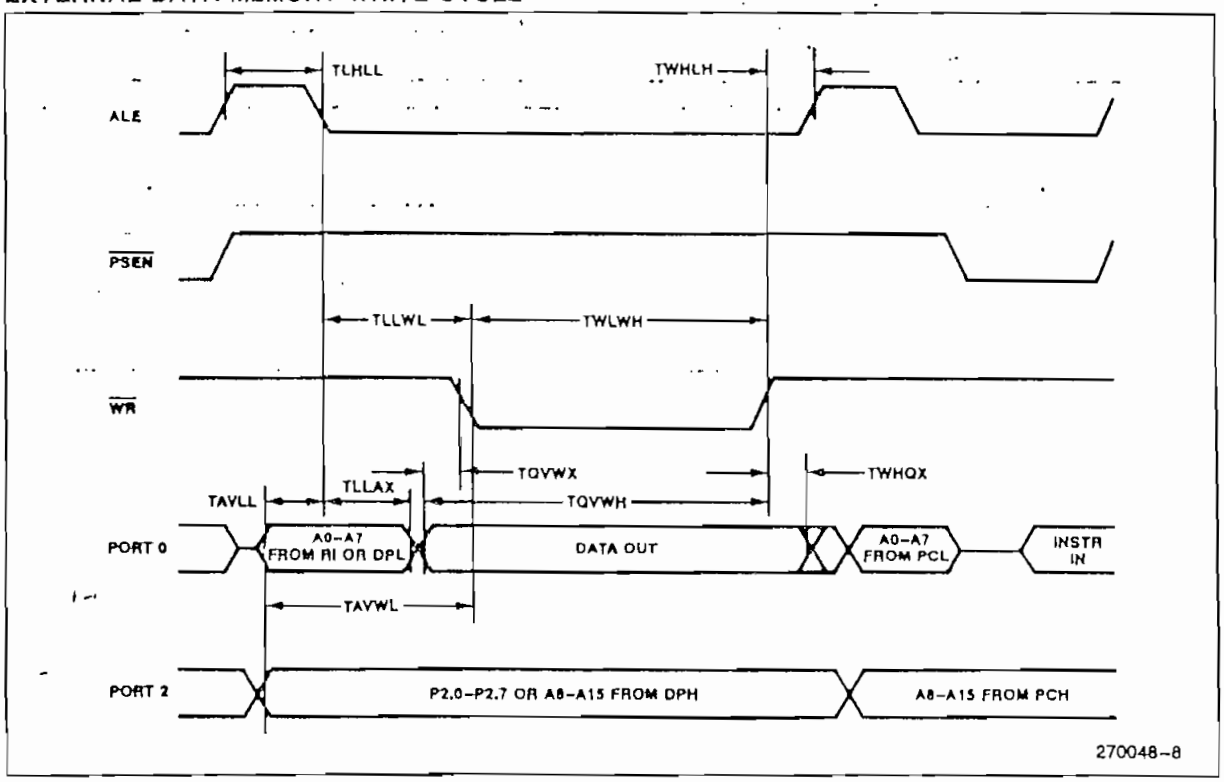
EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE

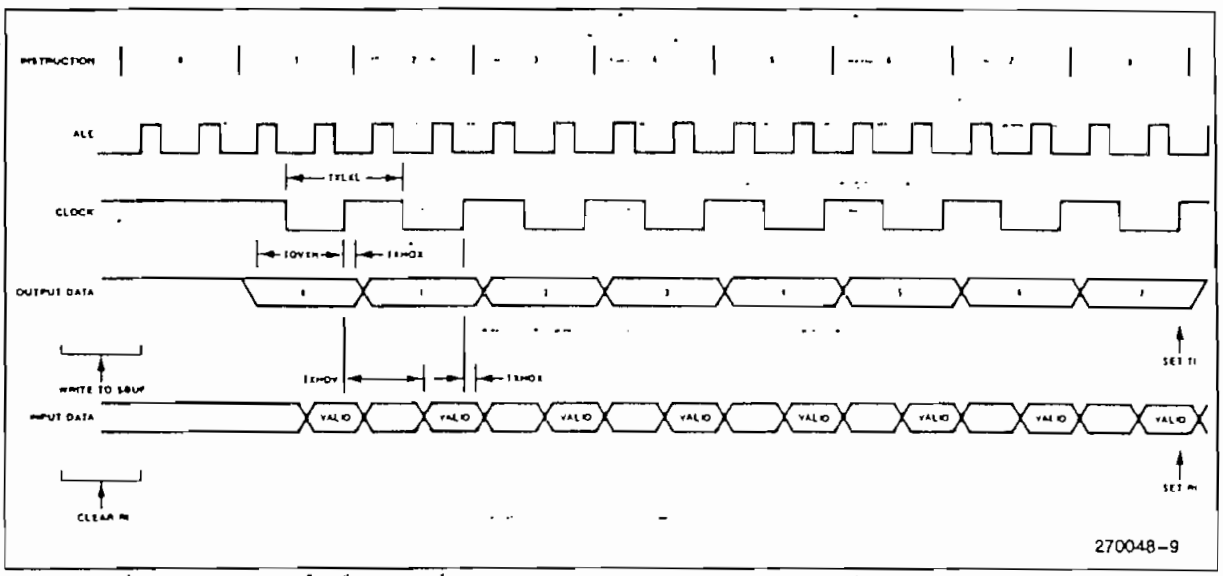


SERIAL PORT TIMING—SHIFT REGISTER MODE

Test Conditions: T_A = 0°C to 70°C; VCC = 5V ±10%; VSS = 0V; Load Capacitance = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL−133		ns
TXHQX	Output Data Hold after Clock Rising Edge	50		2TCLCL−117		ns
TXHDX	Input Data Hold after Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL−133	ns

SHIFT REGISTER TIMING WAVEFORMS

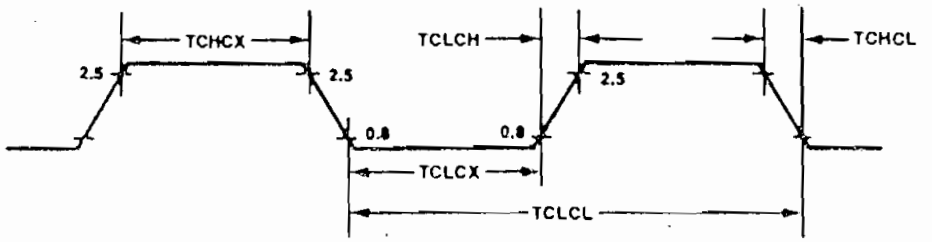


270048-9

EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency (except 8751H-8) 8751H-8	3.5 3.5	12 8	MHz MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

EXTERNAL CLOCK DRIVE WAVEFORM



270048-10

A.C. TESTING INPUT, OUTPUT WAVEFORM

270048-11

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

EPROM CHARACTERISTICS

Table 3. EPROM Programming Modes

Mode	RST	PSEN	ALE	EA	P2.7	P2.6	P2.5	P2.4
Program	1	0	0*	VPP	1	0	X	X
Inhibit	1	0	1	X	1	0	X	X
Verify	1	0	1	1	0	0	X	X
Security Set	1	0	0*	VPP	1	1	X	X

NOTE:
"1" = logic high for that pin
"0" = logic low for that pin
"X" = "don't care"
"VPP" = +21V ±0.5V
*ALE is pulsed low for 50 ms.

Programming the EPROM

To be programmed, the part must be running with a 4 to 6 MHz oscillator. (The reason the oscillator needs to be running is that the internal bus is being used to transfer address and program data to appropriate internal registers.) The address of an EPROM location to be programmed is applied to Port 1 and pins P2.0–P2.3 of Port 2, while the code byte to be programmed into that location is applied to Port 0. The other Port 2 pins, and RST, PSEN, and EA should be held at the "Program" levels indicated in Table 3. ALE is pulsed low for 50 ms to program the code byte into the addressed EPROM location. The setup is shown in Figure 5.

Normally EA is held at a logic high until just before ALE is to be pulsed. Then EA is raised to +21V, ALE is pulsed, and then EA is returned to a logic high. Waveforms and detailed timing specifications are shown in later sections of this data sheet.

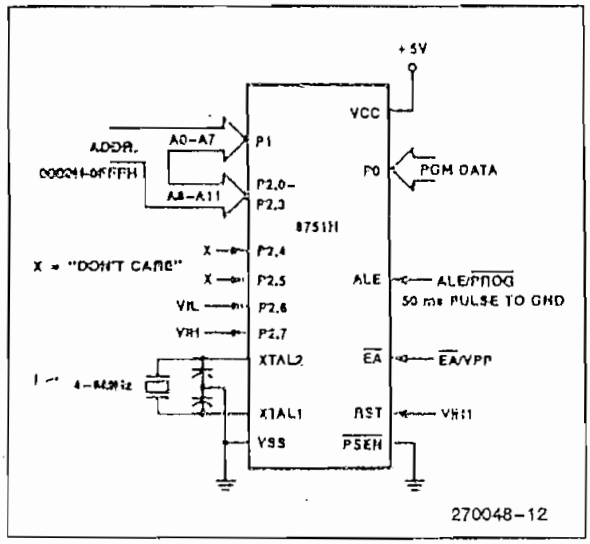


Figure 5. Programming Configuration

Note that the EA/VPP pin must not be allowed to go above the maximum specified VPP level of 21.5V for any amount of time. Even a narrow glitch above that voltage level can cause permanent damage to the device. The VPP source should be well regulated and free of glitches.

Program Verification

If the Security Bit has not been programmed, the on-chip Program Memory can be read out for verification purposes, if desired, either during or after the programming operation. The address of the Program Memory location to be read is applied to Port 1 and pins P2.0–P2.3. The other pins should be held at the "Verify" levels indicated in Table 3. The contents of the addressed location will come out on Port 0. External pullups are required on Port 0 for this operation.

The setup, which is shown in Figure 6, is the same as for programming the EPROM except that pin P2.7 is held at a logic low, or may be used as an active-low read strobe.

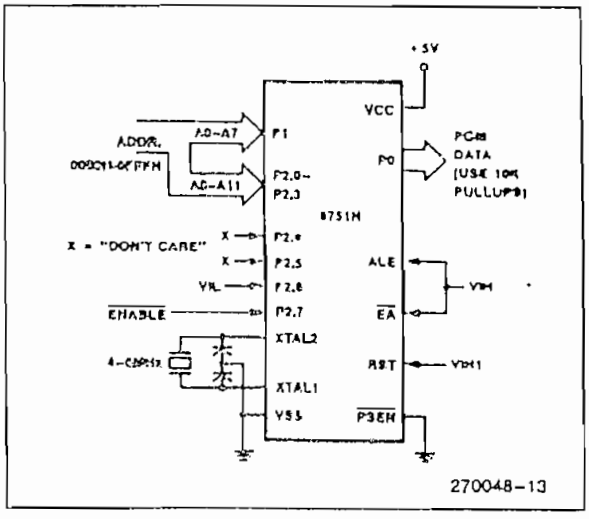


Figure 6. Program Verification

EPROM Security

The security feature consists of a "locking" bit which when programmed denies electrical access by any external means to the on-chip Program Memory. The bit is programmed as shown in Figure 7. The setup and procedure are the same as for normal EPROM programming, except that P2.6 is held at a logic high. Port 0, Port 1, and pins P2.0–P2.3 may be in any state. The other pins should be held at the "Security" levels indicated in Table 3.

Once the Security Bit has been programmed, it can be cleared only by full erasure of the Program Memory. While it is programmed, the internal Program Memory can not be read out, the device can not be further programmed, and it **can not execute out of external program memory**. Erasing the EPROM, thus clearing the Security Bit, restores the device's full functionality. It can then be reprogrammed.

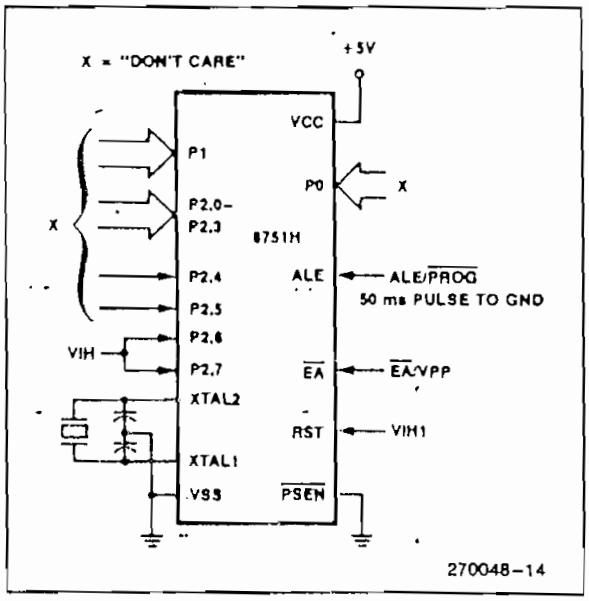


Figure 7. Programming the Security Bit

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room-level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at least 15 W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000 μW/cm^2 rating for 20 to 30 minutes, at a distance of about 1 inch, should be sufficient.

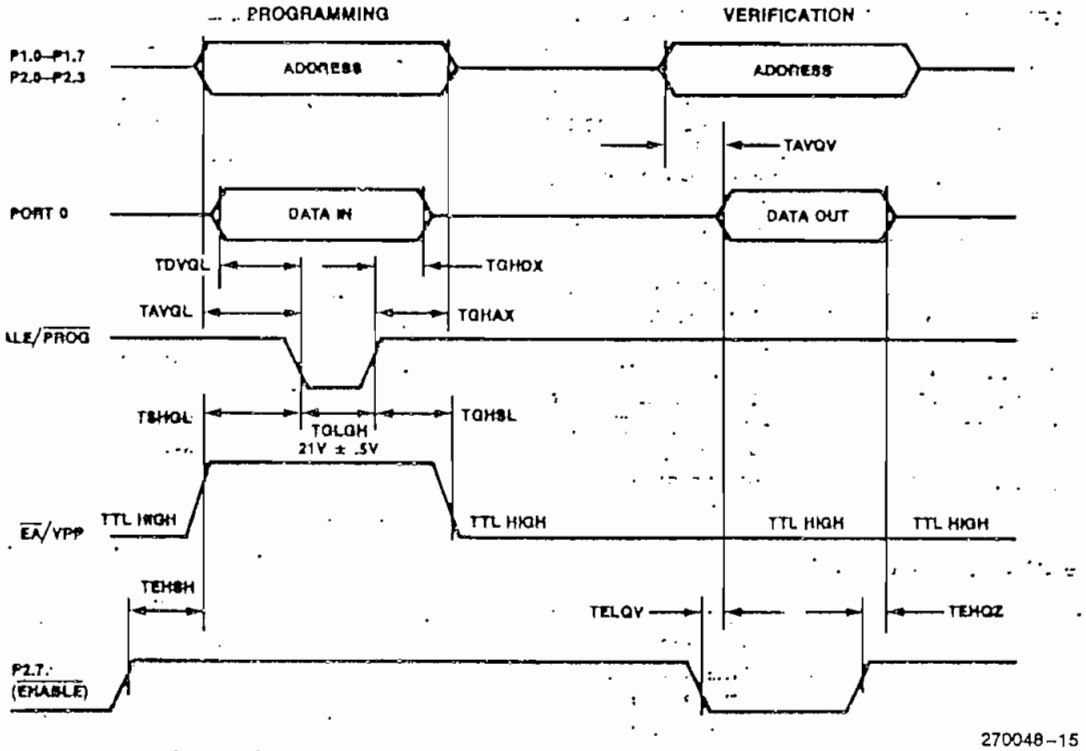
Erasure leaves the array in an all 1s state.

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_A = 21°C to 27°C; VCC = 5V ±10%; VSS = 0V

Symbol	Parameter	Min	Max	Units
VPP	Programming Supply Voltage	20.5	21.5	V
IPP	Programming Supply Current		30	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{PROG}$ Low	48TCLCL		
TGHAX	Address Hold after $\overline{PROG}$	48TCLCL		
TDVGL	Data Setup to $\overline{PROG}$ Low	48TCLCL		
TGHDX	Data Hold after $\overline{PROG}$	48TCLCL		
TEHSH	P2.7 ($\overline{ENABLE}$) High to VPP	48TCLCL		
TSHGL	VPP Setup to $\overline{PROG}$ Low	10		µs
TGHSL	VPP Hold after $\overline{PROG}$	10		µs
TGLGH	$\overline{PROG}$ Width	45	55	ms
TAVQV	Address to Data Valid		48TCLCL	
TELQV	$\overline{ENABLE}$ Low to Data Valid		48TCLCL	
TEHQZ	Data Float after $\overline{ENABLE}$	0	48TCLCL	

EPROM PROGRAMMING AND VERIFICATION WAVEFORMS



DATA SHEET REVISION SUMMARY

The following are the key differences between this and the -003 version of this data sheet:

1. Introduction was expanded to include product descriptions.
2. Package table was added.
3. Design Considerations added.
4. Test Conditions for I_{IL1} and I_{IH} specifications added to the DC Characteristics.
5. Data Sheet Revision Summary added.

MAXIM

+5V Powered RS-232 Drivers/Receivers

General Description

Maxim's family of line drivers/receivers are intended for all RS-232 and V.28/V.24 communications interfaces, and in particular, for those applications where ±12V is not available. The MAX230, MAX236, MAX240 and MAX241 are particularly useful in battery powered systems since their low power shutdown mode reduces power dissipation to less than 5µW. The MAX233 and MAX235 use no external components and are recommended for applications where printed circuit board space is critical.

All members of the family except the MAX231 and MAX239 need only a single +5V supply for operation. The RS-232 drivers/receivers have on-board charge pump voltage converters which convert the +5V input power to the ±10V needed to generate the RS-232 output levels. The MAX231 and MAX239, designed to operate from +5V and +12V, contain a +12V to −12V charge pump voltage converter.

Since nearly all RS-232 applications need both line drivers and receivers, the family includes both receivers and drivers in one package. The wide variety of RS-232 applications require differing numbers of drivers and receivers. Maxim offers a wide selection of RS-232 driver/receiver combinations in order to minimize the package count (see table below).

Both the receivers and the line drivers (transmitters) meet all EIA RS-232C and CCITT V.28 specifications.

Features

- Operates from Single 5V Power Supply (+5V and +12V — MAX231 and MAX239)
- Meets All RS-232C and V.28 Specifications
- Multiple Drivers and Receivers
- Onboard DC-DC Converters
- ±9V Output Swing with +5V Supply
- Low Power Shutdown — <1µA (typ)
- 3-State TTL/CMOS Receiver Outputs
- ±30V Receiver Input Levels

Applications

Computers
Peripherals
Modems
Printers
Instruments

Selection Table

Part Number	Power Supply Voltage	No. of RS-232 Drivers	No. of RS-232 Receivers	External Components	Low Power Shutdown /TTL 3-State	No. of Pins
MAX230	+5V	5	0	4 capacitors	Yes/No	20
MAX231	+5V and +7.5V to 13.2V	2	2	2 capacitors	No/No	14
MAX232	+5V	2	2	4 capacitors	No/No	16
MAX233	+5V	2	2	None	No/No	20
MAX234	+5V	4	0	4 capacitors	No/No	16
MAX235	+5V	5	5	None	Yes/Yes	24
MAX236	+5V	4	3	4 capacitors	Yes/Yes	24
MAX237	+5V	5	3	4 capacitors	No/No	24
MAX238	+5V	4	4	4 capacitors	No/No	24
MAX239	+5V and +7.5V to 13.2V	3	5	2 capacitors	No/Yes	24
MAX240	+5V	5	5	4 capacitors	Yes/Yes	44 (Flatpak)
MAX241	+5V	4	5	4 capacitors	Yes/Yes	28 (Small Outline)

*Patent Pending

ABSOLUTE MAXIMUM RATINGS

V_{CC} ..... −0.3V to +6V
V^+ ..... (V_{CC} − 0.3V) to +14V
V^- ..... +0.3V to −14V
Input Voltages
T_{IN} ..... −0.3 to (V_{CC} + 0.3V)
R_{IN} ..... ±30V
Output Voltages
T_{OUT} ..... (V^+ + 0.3V) to (V^- − 0.3V)
R_{OUT} ..... −0.3V to (V_{CC} + 0.3V)

Short Circuit Duration
T_{OUT} ..... continuous
Power Dissipation
CERDIP 675mW
(derate 9.5mW/°C above +70°C)
Plastic DIP 375mW
(derate 7mW/°C above +70°C)
Small Outline (SO) 375mW
(derate 7mW/°C above +70°C)
Lead Temperature (soldering 10 seconds) ..... +300°C
Storage Temperature ..... −65°C to +160°C

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

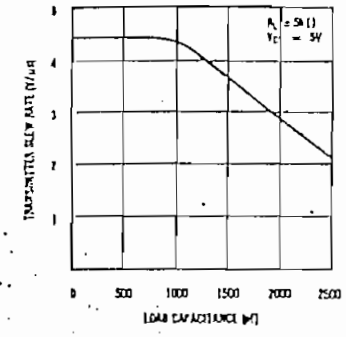
(MAX232, 234, 236, 237, 238, 240, 241 V_{CC} = 5V ±10%; MAX233, 235 V_{CC} = 5V ±5% C1-C4 = 1.0µF; MAX231, 239 V_{CC} = 5V ±10%, V^+ = 7.5V to 13.2V; T_A = Operating Temperature Range, Figures 3-14, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
Output Voltage Swing	All Transmitter Outputs loaded with 3kΩ to Ground		±5	±9		V
V_{CC} Power Supply Current	No load, T_A = +25°C MAX232-MAX233			5	10	mA
	MAX230, MAX234-238, MAX240-MAX241			7	15	
	MAX231, MAX239			0.4	1	
V^+ Power Supply Current	No load, MAX231 and MAX239 only	MAX231		1.8	5	mA
		MAX239		5	15	
Shutdown Supply Current	Figure 1, T_A = +25°C			1	10	µA
Input Logic Threshold Low	T_{IN}, $\overline{EN}$, Shutdown				0.8	V
Input Logic Threshold High	T_{IN}		2.0			V
	$\overline{EN}$, Shutdown		2.4			
Logic Pullup Current	T_{IN} = 0V			15	200	µA
RS-232 Input Voltage Operating Range			-30		+30	V
RS-232 Input Threshold Low	V_{CC} = 5V, T_A = +25°C (MAX231, 239 V^+ = 0V)		0.8	1.2		V
RS-232 Input Threshold High	V_{CC} = 5V, T_A = +25°C (MAX231, 239 V^+ = 12V)			1.7	2.4	V
RS-232 Input Hysteresis	V_{CC} = 5V		0.2	0.5	1.0	V
RS-232 Input Resistance	T_A = +25°C, V_{CC} = 5V		3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 1.6mA (MAX231-233, I_{OUT} = 3.2mA)				0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = 1.0mA		3.5			V
TTL/CMOS Output Leakage Current	$\overline{EN}$ = V_{CC}, 0V ≤ R_{OUT} ≤ V_{CC}			0.05	±10	µA
Output Enable Time (Figure 2)	MAX235, MAX236, MAX239, MAX240, MAX241			400		ns
Output Disable Time (Figure 2)	MAX235, MAX236, MAX239, MAX240, MAX241			250		ns
Propagation Delay	RS-232 to TTL			0.5		µs
Instantaneous Slew Rate	C_L = 10pF, R_L = 3-7kΩ, T_A = +25°C (Note 1)				30	V/µs
Transition Region Slew Rate	R_L = 3kΩ, C_L = 2500pF, Measured from +3V to -3V or -3V to +3V			3		V/µs
Output Resistance	V_{CC} = V^+ = V^- = 0V, V_{OUT} = ±2V		300			Ω
RS-232 Output Short Circuit Current				±10		mA

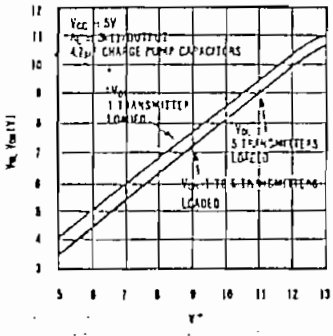
Note 1: Sample tested

Typical Operating Characteristics

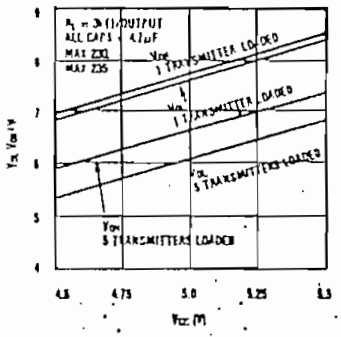
TRANSMITTER SLEW RATE vs. LOAD CAPACITANCE



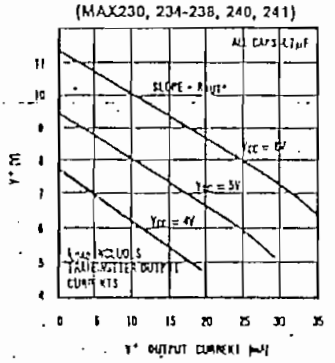
MAX239 TRANSMITTER OUTPUT VOLTAGE vs. V+ VOLTAGE



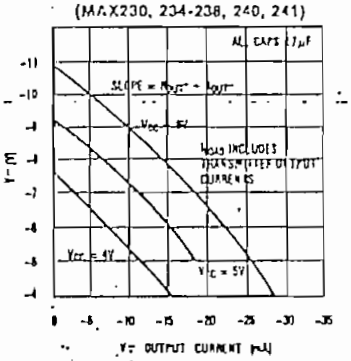
TRANSMITTER OUTPUT VOLTAGE vs. V_{CC} VOLTAGE



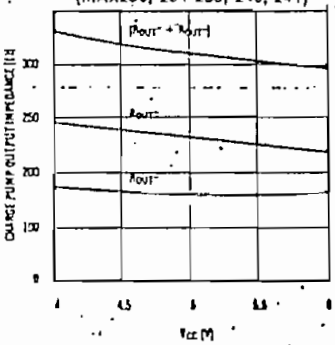
V+ SUPPLY VOLTAGE vs. LOAD CURRENT (MAX230, 234-238, 240, 241)



V- SUPPLY VOLTAGE vs. LOAD CURRENT (MAX230, 234-238, 240, 241)



CHARGE PUMP OUTPUT IMPEDANCE vs. V_{CC} (MAX230, 234-238, 240, 241)



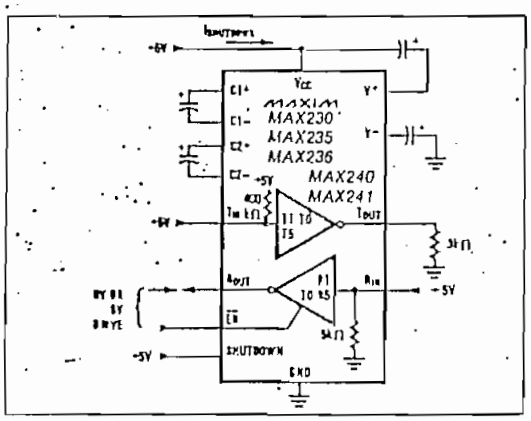


Figure 1. Shutdown Current Test Circuit

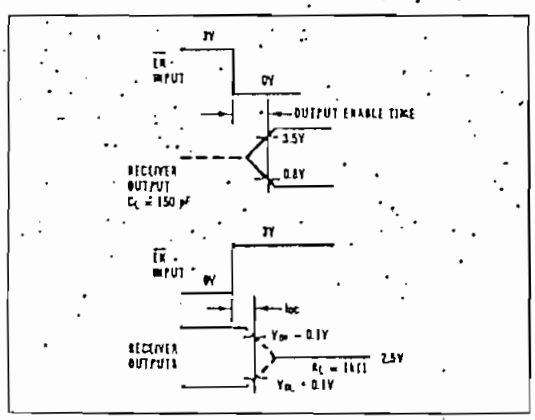
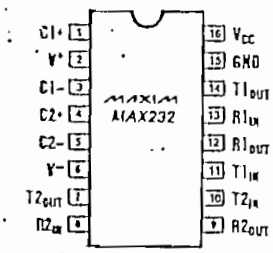


Figure 2. Receiver Output Enable and Disable Timing



16 Lead Small Outline
also available.

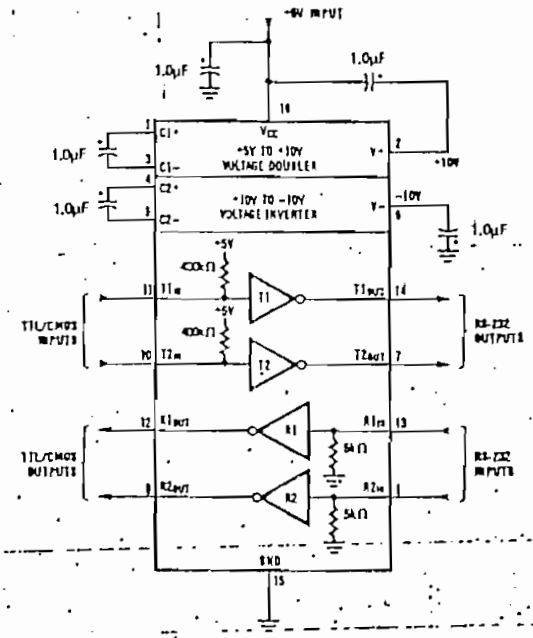
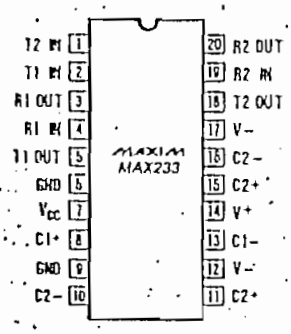


Figure 5. MAX232 Typical Operating Circuit



Small Outline Not Available

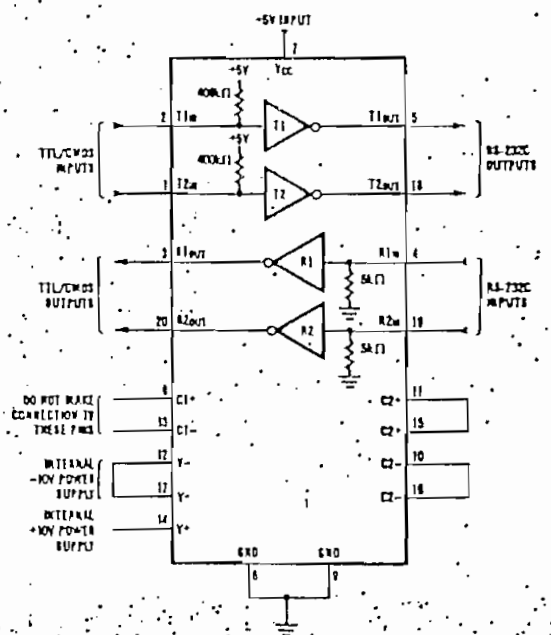


Figure 6. MAX233 Typical Operating Circuit.

V Powered
232 Drivers/Receivers

Typical Applications

3 through 14 show typical applications. The or values are non-critical. Reducing the capa- C1 and C2 to 1µF will slightly increase the nce of the charge pump, lowering the RS-232 utput voltages by about 100mV. Lower values nd C4 increase the ripple on the V+ and V−

ower supply input to the device has a very fast rise (as would occur if a PCB were to be into a card cage with power already on), use ple RC filter shown in Figure 15. This bypass is not needed if the V_{CC} rate-of-rise is below

ivers and drivers are inverting. The ENable of the MAX235, MAX236, MAX239, MAX240 X241 enables the receiver TTL/CMOS outputs is at a low level, and places the TTL/CMOS of the receivers into a high impedance state is a high level.

e Shutdown control of the MAX230, MAX235, , MAX240 and MAX241 is at a logic 1 the pump is turned off, the receiver outputs are the high impedance state, V+ is pulled down − is pulled up to ground, and the transmitter are disabled. The supply current drops to 10µA.

Detailed Description

owing sections provide supplementary infor- lor those designers with non-standard appli- and for those with interest in the internal n of the devices.

ices consist of 3 sections: the transmitters, ivers, and the charge pump DC-DC voltage r.

+5V to ±10V Dual Charge Pump Voltage Converter

All but the MAX231 and MAX239 convert +5V to ±10V. This conversion is performed by two charge pump voltage converters. The first uses capacitor C1 to double the +5V to +10V, storing the +10V on the V+ output filter capacitor, C3. The second charge pump voltage converter uses capacitor C2 to invert the +10V to −10V, storing the −10V on the V− output filter capacitor, C4. The equivalent circuit of the charge pump section is shown in Figure 16.

A small amount of power may be drawn from the V+ and V− outputs to power external circuitry. Two Typical Operating Characteristics graphs show typical output voltage versus load current for the MAX230, 234–238, and 1241. Transmitter output current is included in these plots. The MAX231-233, which are not shown in the graphs, supply less output current, and are limited to 1 or 2mA of excess output load current.

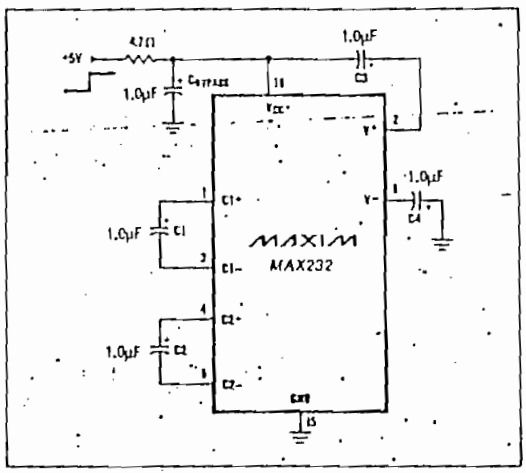
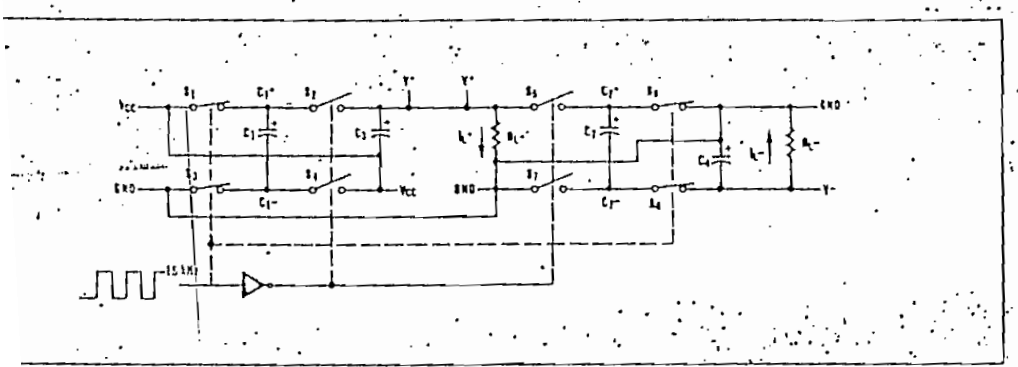


Figure 15. Protection from High $\frac{dV}{dT}$



arge Pump Diagram.

For applications needing only the +5V to ±10V charge pump voltage converter, the MAX680 is available.

The capacitor values for C1 through C4 are noncritical. At the 30kHz (MAX231-MAX233, 60kHz otherwise) typical switching frequency of the voltage converter, a 1µF capacitor has approximately 10Ω impedance and replacing the 4.7µF and 10µF capacitors shown in the typical applications with 1µF for C1 and C2 will increase the output impedance of the V^+ output by about 10Ω and the output impedance of V^- by about 20Ω. Lowering the value of C3 and C4 increases the ripple on the V^+ and V^- outputs. Where operation to the upper temperature limit is not required, or V_{CC} will not go below 4.75V, C1 and C2 can be 1µF, and C3 and C4 can be 1µF per output channel (1µF if one transmitter is used, 5µF if five transmitters are used).

There are parasitic diodes which become forward biased if V^+ goes below V_{CC} or V^- goes above ground. When in the shutdown mode (MAX230, MAX235, MAX236, MAX240 and MAX241 only), V^+ is internally connected to V_{CC} by a 1kΩ pulldown, and V^- is internally connected to ground via a 1kΩ pullup.

The MAX233 and MAX235 contain all charge pump components, including the capacitors, and operate with NO external components.

The MAX231 and MAX239 include only the V^+ to V^- charge pump, and are intended for applications which have a +5V supply and either a +12V ± 10% supply or a 7.5V to 13.2V battery voltage. When operating with V^+ greater than 8.0V, both capacitors can be 1µF.

Driver (Transmitter) Section

The transmitters or line drivers are inverting level translators which convert the CMOS or TTL input levels to RS-232 or V.28 voltage levels. With +5V V_{CC}, the typical output voltage swing is ±9V when loaded with the nominal 5kΩ input resistance of an RS-232 receiver. The output swing is guaranteed to meet the RS-232/V.28 specification of ±5V minimum output swing under the worst case conditions of all transmitters driving the 3kΩ minimum allowable load impedance, V_{CC} = 4.5V, and maximum operating ambient temperature. The open circuit output voltage swing is from (V^+ - 0.6V) to V^-.

The input thresholds are both CMOS and TTL compatible, with a logic threshold of about 25% of V_{CC}. The inputs of unused drivers sections can be left unconnected; an internal 400kΩ input pullup resistor to V_{CC} will pull the inputs high, forcing the unused transmitter outputs low. The input pullup resistors source about 12µA, and the driver inputs should be driven high or open circuited to minimize power supply current in the shutdown mode.

When in the low power shutdown mode, the driver outputs are turned off and their leakage current is less than 1µA with the driver output pulled to ground. The driver output leakage remains less than 1µA, even if the transmitter output is backdriven between 0V and (V_{CC} + 6V). Below –0.5V the transmitter is diode clamped to ground with 1kΩ series impedance. The transmitter is also zener clamped to approximately V_{CC} + 6V, with a series impedance of 1kΩ. As required by the RS232 and V.28, the slew rate is limited to less than 30V/µs. This limits the maximum usable baud rate to 19,200 baud.

Receiver Section

All but the MAX230 and MAX234 contain RS-232/V.28 receivers. These receivers convert the ±5V to ±15V RS-232 signals to 5V TTL/CMOS outputs. Since the RS-232C/V.28 specifications define a voltage level greater than +3V as a 0, the receivers are inverting. Maxim has set the guaranteed input thresholds of the receivers to 0.8V minimum and 2.4V maximum, which are significantly tighter than the -3.0V minimum and +3.0V maximum required by the RS-232 and V.28 specifications. This allows the receivers to respond both to RS-232/V.28 levels and TTL level inputs. The receivers are protected against input overvoltage up to ±30V.

The 0.8V guaranteed lower threshold is important to ensure that the receivers will have a logic 1 output if the receiver is not being driven because the equipment containing the line driver is turned off or disconnected, or if the connecting cable has an open circuit or short circuit. In other words, the receiver implements Type 1 interpretation of fault conditions (§7 of V.28, §2.5 of RS-232C). While a 0V or even a –3V receiver threshold would be acceptable for the data lines, these lower thresholds would not give proper indication on the control lines such as DTR and DSR. The receivers, on the other hand, have a full 0.8V noise margin for detecting the power-down or cable-disconnected states.

The receivers have a hysteresis of approximately 0.5V, with a minimum guaranteed hysteresis of 200mV. This aids in obtaining clean output transitions, even with slow rise and fall time input signals with moderate amounts of noise and ringing. The propagation delays of the receivers are 350ns for negative-going input signals, and 650ns for positive-going input signals (see Typical Characteristics graphs).

The MAX239 has a receiver 3-state control line, and the MAX235, MAX236, MAX240 and MAX241 have both a receiver 3-state control line and a low power shutdown control. The receiver TTL/CMOS outputs are in a high impedance 3-state mode whenever the 3-state $\overline{\text{EN}}$able line is high, and are also high impedance whenever the Shutdown control line is high.

+5V Powered RS-232 Drivers/Receivers

Review of EIA Standard RS-232-C and CCITT Recommendations V.28 and V.24

The most common serial interface between electronic equipment is the "RS232" interface. This serial interface has been found to be particularly useful for the interface between units made by different manufacturers since the voltage levels are defined by the EIA Standard RS-232-C and CCITT Recommendation V.28. The RS-232 specification also contains signal circuit definitions and connector pin assignments, while CCITT circuit definitions are contained in a separate document, Recommendation V.24. Originally intended to interface modems to computers and terminals, these standards have many signals which are not used for computer-to-computer or computer-to-peripheral communication.

Serial interfaces can be used with a variety of transmission formats. The most popular by far is the asynchronous format, generally at one of the standard baud rates of 300, 600, 1200, etc. The maximum recommended baud rate for RS-232 and V.28 is 20,000 baud, and the fastest commonly used baud rate is 19,200 baud. Asynchronous serial links use a variety of combinations of the number of data bits, what type (if any) of parity bit, and the number of stop bits. A typical combination is 7 data bits, even parity, and 1 stop bit.

RS232/V.28 physical links are also suitable for synchronous transmission protocols. These higher level protocols often use the standard RS-232C/V.28 voltage levels. Note that one type of physical link (such as RS-232/V.28 voltage levels) can be used for a variety of higher level protocols. Table 2 summarizes the voltage levels and other requirements of V.28 and RS-232.

Comparison of RS-232C/V.28 with other Standards

The other two most common serial interface specifications are the EIA RS423 and RS422/RS485 (CCITT recommendations V.10 and V.11). While the RS-232 or V.28/V.24 interface is the most common interface for communication between equipment made by different manufacturers, the RS423/V.10 interface and RS422/V.11 interfaces can operate at higher baud rates. In addition, the RS485 interface can be used for low cost local area networks.

The RS423 and V.10 interfaces are unbalanced or "single-ended" interfaces which use a differential receiver. This standard is intended for data signaling rates up to 100 kbit/s (100 kilobaud). It achieves this higher baud rate through more precise requirements on the waveshape of the transmitters and through the use of differential receivers to compensate for ground potential variations beween the transmitting and receiving equipment. With certain limitations, this interface is compatible with RS-232 and V.28. The limitations are:

1) less than 20,000 baud rate,
2) maximum cable lengths determined by RS-232 performance,
3) RS423/V.10 DTE and DCE signal return paths must be connected to the the RS232/V.28 signal ground,
4) the RS-232 transmitter output voltages must be limited to ±12V, or additional protection must be provided for the RS423/V.10 receivers, and
5) not all RS232/V.28 receivers will show proper power-off detection of V.10 transmitter outputs.

Maxim's MAX230 and MAX232-MAX238, MAX240 and MAX241 meet restrictions 4 and 5 over the entire range of recommended operating conditions. The MAX231 and MAX239 meet restrictions 4 and 5 provided that the V⁻ voltage is 12.5V or less.

The RS422, RS485, and V.11 interfaces are balanced double-current interchanges suitable for baud rates up to 10 Mbit/s. These interfaces are not compatible with RS-232 or V.28 voltage levels.

Application Hints

Operation at High Baud Rates

V.28 states that "the time required for the signal to pass through the transition region during a change in state shall not exceed 1 millisecond or 3 percent of the nominal element period on the interchange circuit, whichever is less." RS-232C allows the transition time to be 4 percent of the duration of a signal element. At 19,200 baud, the "nominal element period" is approximately 50µs, of which 3 percent is 1.5µs. Since the transition region is from −3V to +3V, this means the V.28 slew rate would ideally be faster than 6V/1.5µs = 4V/µs at 19.2 kbaud and 2V/µs at 9600 baud. The RS-232 requirement is equivalent to 3V/µs at 19.2 kbaud, 1.5V/µs at 9600 baud, etc. The slew rate of the MAX230 series devices is about 3V/µs with the maximum recommended load of 2500pF. In practice, the effect of less than optimum slew rate is a distortion of the recovered data, where the 1's and 0's no longer have equal width. This distortion generally has negligible effect and the devices can be reliably used for 19.2 kbaud serial links when the cable capacitance is kept below 2500pF. With very low capacitance loading, the MAX230 and MAX234-239, MAX240 and MAX241 may even be used at 38.4 kbaud, since the typical slew rate is 5V/µs when loaded with 500pF in parallel with 5kΩ. Under no circumstance will the

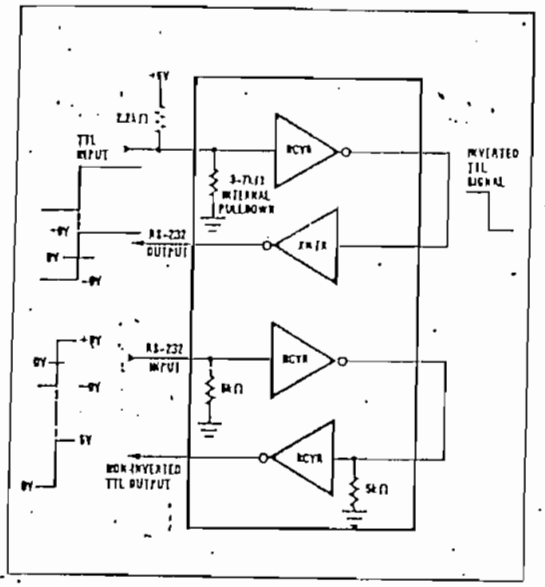


Figure 17. Non-inverting RS-232 Transmitters and Receivers.

slew rate exceed the RS-232/V.28 maximum spec of 30V/µs and, unlike the 1488 driver, no external compensation capacitors are needed under any load condition.

Driving Long Cables

The RS-232 standard states that "The use of short cables (each less than approximately 50 feet or 15 meters) is recommended; however, longer cables are permissible, provided that the load capacitance . . . does not exceed 2500pF."

Baud rate and cable length can be traded off: use lower baud rates for long cables, use short cables if high baud rates are desired. For both long cables and high baud rates, use RS422/V.11. The maximum cable length for a given baud rate is determined by several factors, including the capacitance per meter of cable, the slew rate of the driver under high capacitive loading, the receiver threshold and hysteresis, and the acceptable bit error rate. The receivers have 0.5V of hysteresis, and the drivers are designed such that the slew rate reduction caused by capacitive loading is minimized (see Typical Characteristics).

Non-Inverting Drivers and Receivers

Occasionally a non-inverting driver or receiver is needed instead of the inverting drivers and receivers of the family. Simply use one of the receivers as a TTL/CMOS inverter to get the desired operation (Figure 17). If the logic output driving the receiver input has less than 1mA of output source capability, then add the 2.2kΩ pullup resistor.

The receiver TTL outputs can directly drive the input of another receiver to form a non-inverting RS-232 receiver.

Protection for Shorts to ±15V Supplies

All driver outputs except on the MAX231, MAX232 and MAX233 are protected against short circuits to ±15V, which is the maximum allowable loaded output voltage of an RS-232/V.28 transmitter. The MAX231, MAX232, and MAX233 can be protected against short circuits to ±15V power supplies by the addition of a series 220Ω resistor in each output. This protection is not needed to protect against short circuits to most RS-232 transmitters such as the 1488, since they have an internal short circuit current limit of 12mA.

The power dissipation of the MAX230 and MAX234-MAX239, MAX240 and MAX241 is about 200mW with all transmitters shorted to ±15V.

Isolated RS-232 Interfaces

RS-232 and V.28 specifications require a common ground connection between the two units communicating via the RS-232/V.28 interface. In some cases, there may be large differences in ground potential between the two units, and in other cases it may be desired to avoid ground loop currents by isolating the two grounds. In other cases, a computer or control system must be protected against accidental connection of the RS-232/V.28 signal lines to 110/220VAC power lines. Figure 18 shows a circuit with this isolation. The power for the MAX233 is generated by a MAX635 DC-DC converter. When the MAX635 regulates point "A" to -5V, the isolated output at point "B" will be semi-regulated to +5V. The two optocouplers maintain isolation between the system ground and the RS-232 ground while transferring the data across the isolation barrier. While this circuit will not withstand 110VAC between the RS-232 ground and either the receiver or transmitter lines, the voltage difference between the two grounds is only limited by the optocoupler and DC-DC converter transformer breakdown ratings.

+5V Powered RS-232 Drivers/Receivers

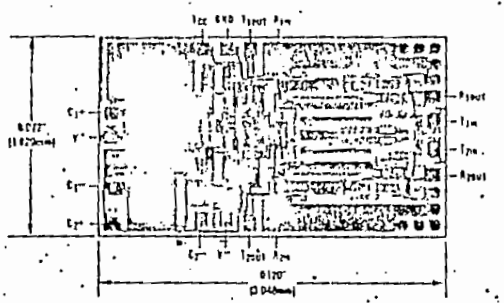
Table 1. Circuits Commonly Used for RS-232C and V.24 Asynchronous Interfaces

PIN	CIRCUIT	
1	Protective Ground	Connect to Earth Ground
2	Transmit Data (TD)	Data from DTE
3	Receive Data (RD)	Data from DCE
4	Request To Send (RTS)	Handshake from DTE
5	Clear to Send (CTS)	Handshake from DCE
6	Data Set ready (DSR)	Handshake from DCE
7	Signal Ground	Reference Point for Signals
8	Received Line Signal Detector (sometimes called Carrier Detect, DCD)	Handshake from DCE
11	Printer Busy Signal	Handshake from Printer
20	Data Terminal Ready	Handshake from DTE
22	Ring Indicator	Handshake from DCE

Table 2. Summary of RS-232C and V.28 Electrical Specifications

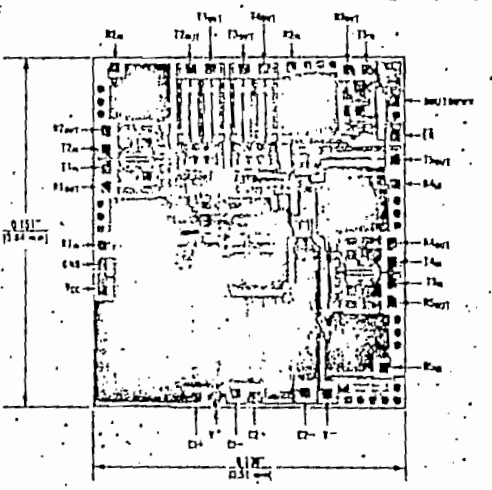
PARAMETER	SPECIFICATION	COMMENTS
Driver Output Voltage		
0 level	+5V to +15V	With 3-7kΩ load
1 level	-5V to -15V	With 3-7kΩ load
Max. output	±25V Max.	No Load
Receiver Input Thresholds (data and clock signals)		
0 level	+3V to +25V	
1 level	-3V to -25V	
Receiver Thresholds RTS, DSR, DTR		
On level	+3V to +25V	
Off level	Open Circuit or -3V to -25V	Detects Power Off Condition at Driver
Receiver Input Resistance	3kΩ to 7kΩ	
Driver Output Resistance, power off condition	300Ω Min.	V_{OUT} < ±2V
Driver Slew Rate	30V/µs Max.	3kΩ < R_L < 7kΩ; 0pF < C_L < 2500pF
Signalling Rate	Up to 20kbits/sec.	
Cable Length	50'/15 m. Recommended Max. Length	Longer cables permissible, if C_{LOAD} ≤ 2500pF

Chip Topography



MAX231, MAX232 and MAX233

Note: Connect substrate to V+.



MAX230 and MAX234-239, MAX240, MAX241

Notes:
1. Shutdown pin of MAX234, MAX237, MAX238, MAX239, MAX240 and MAX241 are internally connected to ground.
2. Connect substrate to V+.

National Semiconductor

Electronic Data Processing

MM58167 Microprocessor Compatible Real Time Clock

General Description

The MM58167 is a low threshold metal-gate CMOS circuit that functions as a real time clock calendar in bus-oriented microprocessor systems. The device includes an addressable counter, addressable latch for alarm-type functions, and 2 interrupt outputs. A power-down input allows the chip to be disabled from the outside world for standby low power operation. The time base is generated from a 32,768 Hz crystal-controlled oscillator.

Features

- Microprocessor compatible
- Thousandths of seconds, hundredths of seconds, tenths of seconds, seconds, minutes, hours, day of the week, day of the month, and month counters with corresponding latches for alarm-type functions
- Interrupt output (maskable) with 8 possible interrupt signals:
  - Latch and counter comparison
  - Every tenth of a second
  - Every second
  - Every minute
  - Every hour
  - Every day
  - Every week
  - Every month
- Power-down mode that disables all outputs except for an interrupt output that occurs on a counter latch comparison. This is not the same as the maskable interrupt output
- Don't care states in the latches
- Status bit to indicate clock rollover during a read
- 32,768 Hz crystal reference, with only the input tuning capacitor and load capacitor needed externally
- Four year calendar

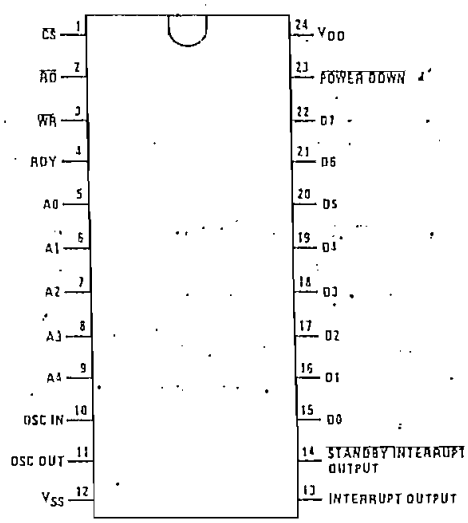
Functional Description

The MM58167 is a microprocessor oriented real time clock. The circuit includes addressable real time counters and addressable latches, each for thousandths of seconds through months. The counter and latch are divided into bytes of 4 bits each. When addressed, 2 bytes will appear on the data I/O bus. The data, in binary coded decimal, can be transferred to and from the counters via the data I/O bus so that each set of 2 bytes (1 word) can be accessed independently as grouped in Table I.

If either of the bytes in the above 8-bit counter words do not legally reach 4-bit lengths (e.g., day of the week uses only the 3 least significant bits) the unused bits will be unrecognized during a write and held at V_{SS} during a read. If any illegal data is entered into the counters during a write cycle, it may take up to 4 clocks (4 months in the case of the month counter) to restore legal BCD data to the counter during normal counting. The latches will read and write all 4 bits per byte. Each of the counter and latch words can be reset with the appropriate address and data inputs. The counter reset is a write function. The latches can be programmed to compare with the counters at all times by writing 1's into the 2 most significant bits of each latch, thus establishing a don't care state in the latch. The don't care state is programmable on the byte level, i.e., tens of hours can contain a don't care state, yet unit hours can contain a valid code necessary for a comparison.

Connection Diagram

Dual-In-Line Package



TOP VIEW

Order Number MM58167N
See Package 22

Absolute Maximum Ratings

Voltage at All Inputs and Outputs	V_{DD} + 0.3 to V_{SS} − 0.3
Operating Temperature	−25°C to +85°C
Storage Temperature	−65°C to +150°C
$V_{DD} - V_{SS}$	6V
Lead Temperature (Soldering, 10 seconds)	300°C

Electrical Characteristics $T_A = -25°C$ to $+85°C$, $V_{SS} = 0V$

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage					
V_{DD}	Outputs Enabled	4.0		5.5	V
V_{DD} (Note 1)	Power Down Mode	2.0		5.5	V
Supply Current					
I_{DD}, Static	Outputs TRI-STATE, f_{IN} = DC, V_{DD} = 5.5V			10	μA
I_{DD}, Dynamic	Outputs TRI-STATE, f_{IN} = 32 kHz, V_{DD} = 5.5V, $V_{IH} \geq V_{DD} - 0.3V$, $V_{IL} \leq V_{SS} + 0.3V$			20	μA
I_{DD}, Dynamic	Outputs TRI-STATE, f_{IN} = 32 kHz, V_{DD} = 5.5V, V_{IH} = 2.0V, V_{IL} = 0.8V			12	mA
Input Voltage					
Logical Low		0.0		0.8	V
Logical High		2.0		V_{DD}	V
Input Leakage Current	$V_{SS} \leq V_{IN} \leq V_{DD}$			1	μA
Output Impedance	(I/O and Interrupt Output)				
Logical Low	V_{DD} = 4.75V, I_{OL} = 1.6 mA			0.4	V
Logical High	V_{DD} = 4.75V, I_{OH} = −400 μA,	2.4			V
	I_{OH} = −10 μA	0.8 V_{DD}			V
TRI-STATE®	V_{OUT} = 0V,			−1	μA
	V_{OUT} = V_{DD}			1	μA
Output Impedance	(Ready and Standby Interrupt Output)				
Logical Low, Sink	V_{DD} = 4.75V, I_{OL} = 1.6 mA			0.4	V
Logical High, Leakage	$V_{OUT} \leq V_{OD}$			10	μA

Note 1: To insure that no illegal data is read from or written into the chip during power up, the power down input should be enabled only after all other lines (Read, Write, Chip Select, and Data Bus) are valid.

Functional Description (Continued)

TABLE I

COUNTER ADDRESSED	UNITS D0	D1	D2	D3	MAX USED BCD CODE	TENS D4	D5	D6	D7	MAX USED BCD CODE
Ten Thousandths of a Second	0	0	0	0	0	I/O	I/O	I/O	I/O	9
Tenths and Hundredths of Seconds	I/O	I/O	I/O	I/O	9	I/O	I/O	I/O	I/O	9
Seconds	I/O	I/O	I/O	I/O	9	I/O	I/O	I/O	0	5
Minutes	I/O	I/O	I/O	I/O	9	I/O	I/O	I/O	0	5
Hours	I/O	I/O	I/O	I/O	9	I/O	I/O	0	0	2
Day of the Week	I/O	I/O	I/O	0	7	0	0	0	0	0
Day of the Month	I/O	I/O	I/O	I/O	9	I/O	I/O	0	0	3
Month	I/O	I/O	I/O	I/O	9	I/O	0	0	0	1

Functional Description (Continued)

TABLE II. ADDRESS CODES AND FUNCTIONS

A4	A3	A2	A1	A0	FUNCTION
0	0	0	0	0	Counter — Thousandths of Seconds
0	0	0	0	1	Counter — Hundredths and Tenths of Seconds
0	0	0	1	0	Counter — Seconds
0	0	0	1	1	Counter — Minutes
0	0	1	0	0	Counter — Hours
0	0	1	0	1	Counter — Day of the Week
0	0	1	1	0	Counter — Day of the Month
0	0	1	1	1	Counter — Months
0	1	0	0	0	Latches — Thousandths of Seconds
0	1	0	0	1	Latches — Hundredths and Tenths of Seconds
0	1	0	1	0	Latches — Seconds
0	1	0	1	1	Latches — Minutes
0	1	1	0	0	Latches — Hours
0	1	1	0	1	Latches — Day of the Week
0	1	1	1	0	Latches — Day of the Month
0	1	1	1	1	Latches — Months
1	0	0	0	0	Interrupt Status Register
1	0	0	0	1	Interrupt Control Register
1	0	0	1	0	Counter Reset
1	0	0	1	1	Latch Reset
1	0	1	0	0	Status Bit
1	0	1	0	1	"GO" Command
1	0	1	1	0	Standby Interrupt
1	1	1	1	1	Test Mode

All others unused.

TABLE III. COUNTER AND LATCH RESET FORMAT

D0	D1	D2	D3	D4	D5	D6	D7	COUNTER OR LATCH RESET
1	0	0	0	0	0	0	0	Thousandths of Seconds
0	1	0	0	0	0	0	0	Hundredths and Tenths of Seconds
0	0	1	0	0	0	0	0	Seconds
0	0	0	1	0	0	0	0	Minutes
0	0	0	0	1	0	0	0	Hours
0	0	0	0	0	1	0	0	Days of the Week
0	0	0	0	0	0	1	0	Days of the Month
0	0	0	0	0	0	0	1	Months

FOR COUNTER RESET A4—A0 MUST BE 10010
FOR LATCH RESET A4—A0 MUST BE 10011

Functional Description (Continued)

Following a read of any real time counter a status bit read should be done. If during a counter read cycle the clock rolls over, the data read out could be invalid. Thus, during a read if the clock rolls over the status bit will be set. The status bit will appear on D0 when read, D1 through D7 will be zeros.

To synchronize the clock with real time a "GO" command exists which can be used to reset the thousandths of seconds, hundredths and tenths of seconds, and seconds counters. After setting the lower frequency counters (minutes through months), the appropriate address and a write pulse can be sent to reset all counters mentioned above. This allows the clock to be started at an exactly known time. It can also be used as a stopwatch function. The "GO" command is the start and a counter read is the stop point. The clock does not stop during or following a read, so each read would be a split time.

A second special command will enable the standby interrupt output. The standby interrupt output is the only input or output enabled during the power down or standby mode. Power down occurs when the power down input goes to a logical zero level. In this mode the outputs are TRI-STATED and the inputs ignored regardless of the state of the chip select. The standby interrupt is enabled by writing a 1 on the D0 line with the standby interrupt address selected. On the next counter-latch comparison the open drain output device turns on, sinking current. The output will be turned on immediately upon writing a 1 on D0 if the comparison occurred before the write, yet is still in effect. To disable the output a zero on D0 is written at the standby interrupt address. The write cycles must occur during normal operation, but the output can become active during power down. This feature can be used to turn the power back on during a power down mode (see *Figure 4* for a typical application). Refer to Tables II and III for the address input codes and functions and for the counter and latch reset format.

The interrupt output is controlled by the interrupt status register (8 bits) and the interrupt control register (8 bits). The status register contains the present state of the comparator (compares the counters and latches) and the outputs (1 bit each) of the tenths of seconds, seconds, minutes, hours, week, day of the month, and month counters *(Figure 1)*. The interrupt status register can only be read. The interrupt control register is a mask register that regulates which of the 8 bits in the status register goes out as an interrupt. The control register cannot be read from. A 1 is written into the control register to select the appropriate interrupt output. If more than a single 1 exists in the control register each selected bit will come out as an interrupt. This will appear as an interrupt occurring at the highest frequency selected. The interrupt is acknowledged by addressing and reading the status register. Once acknowledged the interrupt output and status register are reset. The only way to disable the interrupt output is to write all 0's into the control register or to enable the power down input.

The I/O bus is controlled by the read, write, ready and chip select lines. During a read cycle ($\overline{RD}$ = 0, $\overline{WR}$ = 1, $\overline{CS}$ = 0, RDY = 0) the data on the I/O bus is the data contained in the addressed counter or latch. During a write cycle ($\overline{RD}$ = 1, $\overline{WR}$ = 0, $\overline{CS}$ = 0, RDY = 0) the data on the I/O bus is latched into the addressed counter or latch. At the start of each read or write cycle the RDY signal goes low and will remain low until the clock has placed valid data on the bus or until it has completed latching data in on a write. The chip select line is used to enable or disable the device outputs. When the chip is selected the device will drive the I/O bus for a read or use the I/O bus as an input for a write. The I/O bus will not be affected when the chip is deselected. The outputs driving the bus will go to the TRI-STATE or high impedance state. The chip will not respond to any inputs when deselected. Refer to *Figures 2 and 3* for read and write cycle timing.

The clock's time base is a 32,768 crystal controlled oscillator. Externally, the crystal, the input tuning capacitor, and the output load capacitor are required. Included internally are a high gain inverter, an RC delay, and the bias resistor. To tune the oscillator a constant read can be done on one of the higher frequency counters. For example, a constant read of the thousandths of seconds counter will place an average 500 Hz signal on the D4 bus line. The period varies slightly due to disable of latches during counter roll.

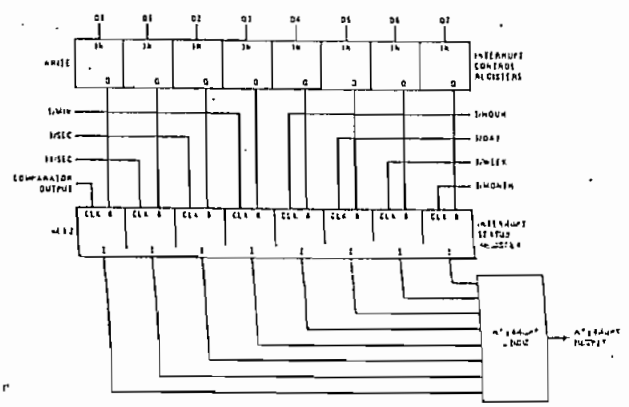


FIGURE 1. Interrupt Register Format

Read Cycle Timing Characteristics $T_A = -25°C$ to $+85°C$, $V_{DD} = 4.0V$ to $5.5V$, $V_{SS} = 0V$

	PARAMETER	MIN	TYP	MAX	UNITS
t_{AR}	Address Bus Valid to Read Strobe	100			ns
t_{CSR}	Chip Select ON to Read Strobe	0			ns
t_{RRY}	Read Strobe to Ready Strobe			150	ns
t_{RYD}	Ready Strobe to Data Valid			800	ns
t_{AD}	Address Bus Valid to Data Valid			1050	ns
t_{RH}	Data Hold Time from Trailing Edge of Read Strobe	0			ns
t_{HZ}	Trailing Edge of Read Strobe to TRI-STATE Mode			250	ns
t_{RYH}	Read Hold Time After Ready Strobe	0			ns
t_{RA}	Address Bus Hold Time from Trailing Edge of Read Strobe	50			ns

Data bus loading is 100 pF
Ready output loading is 50 pF
Input and output AC timing levels are:
Logical "1" = 2.0V
Logical "0" = 0.8V

Write Cycle Timing Characteristics $T_A = -25°C$ to $+85°C$, $V_{DD} = 4.0V$ to $5.5V$, $V_{SS} = 0V$

	PARAMETER	MIN	TYP	MAX	UNITS
t_{AW}	Address Valid to Write Strobe	100			ns
t_{CSW}	Chip Select ON to Write Strobe	0			ns
t_{DN}	Data Valid Before Write Strobe	100			ns
t_{WRY}	Write Strobe to Ready Strobe			150	ns
t_{RY}	Ready Strobe Width			800	ns
t_{RYH}	Write Hold Time After Ready Strobe	0			ns
t_{WD}	Data Hold Time After Write Strobe	110			ns
t_{WA}	Address Hold Time After Write Strobe	50			ns

Data bus loading is 100 pF
Ready output loading is 50 pF
Input and output AC timing levels are:
Logical "1" = 2.0V
Logical "0" = 0.8V

Switching Time Waveforms

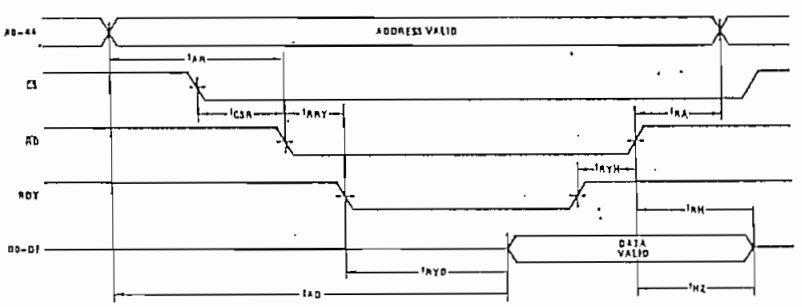


FIGURE 2. Read Cycle Waveforms

Switching Time Waveforms (Continued)

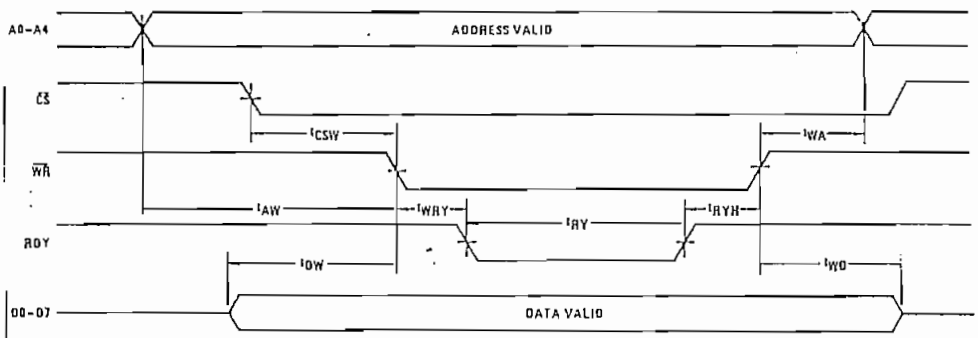


FIGURE 3. Write Cycle Waveforms

Typical Application

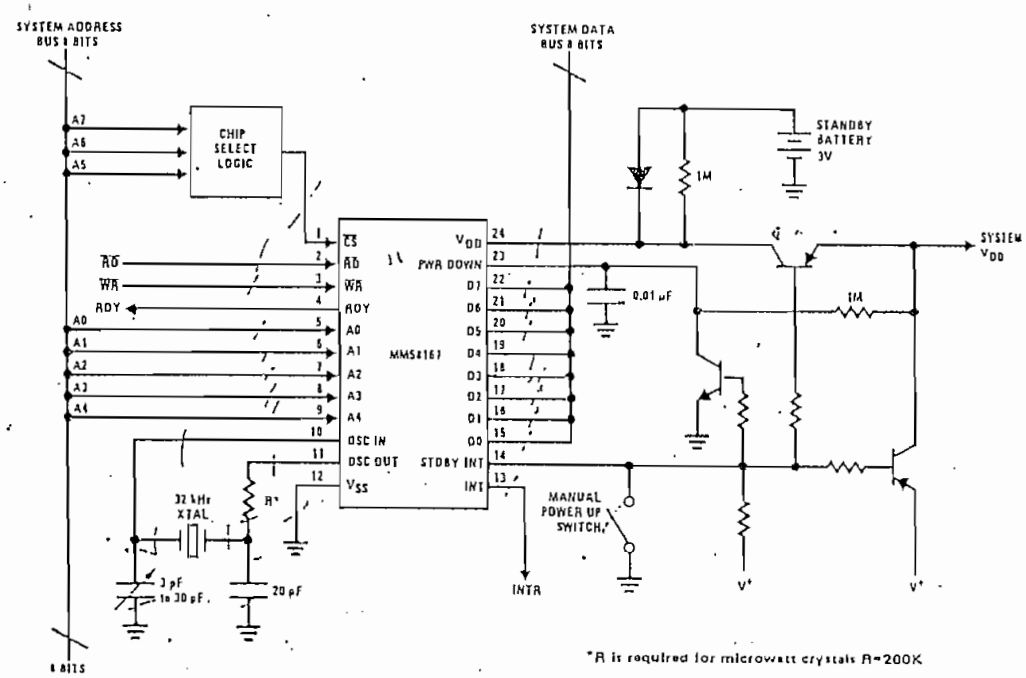
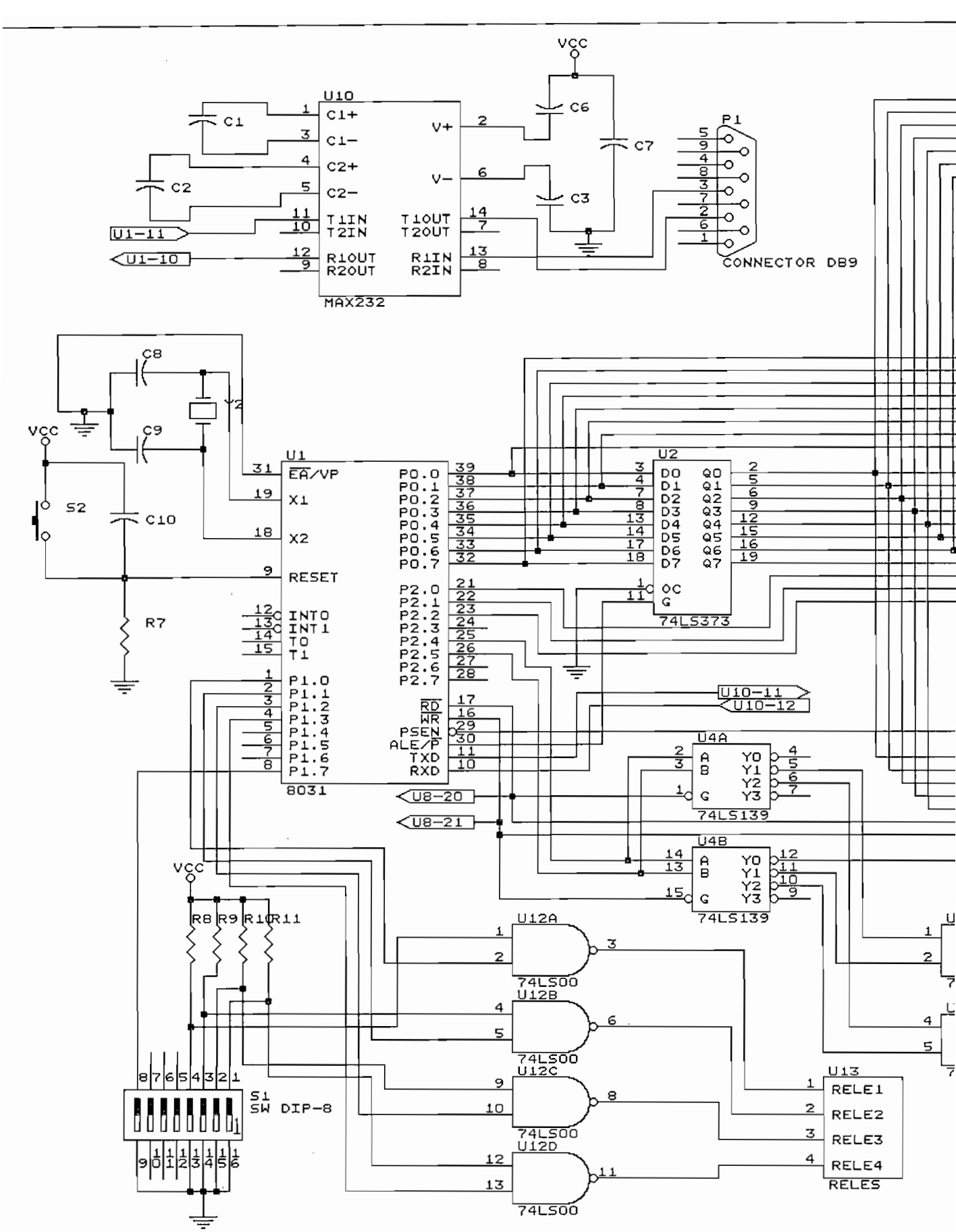
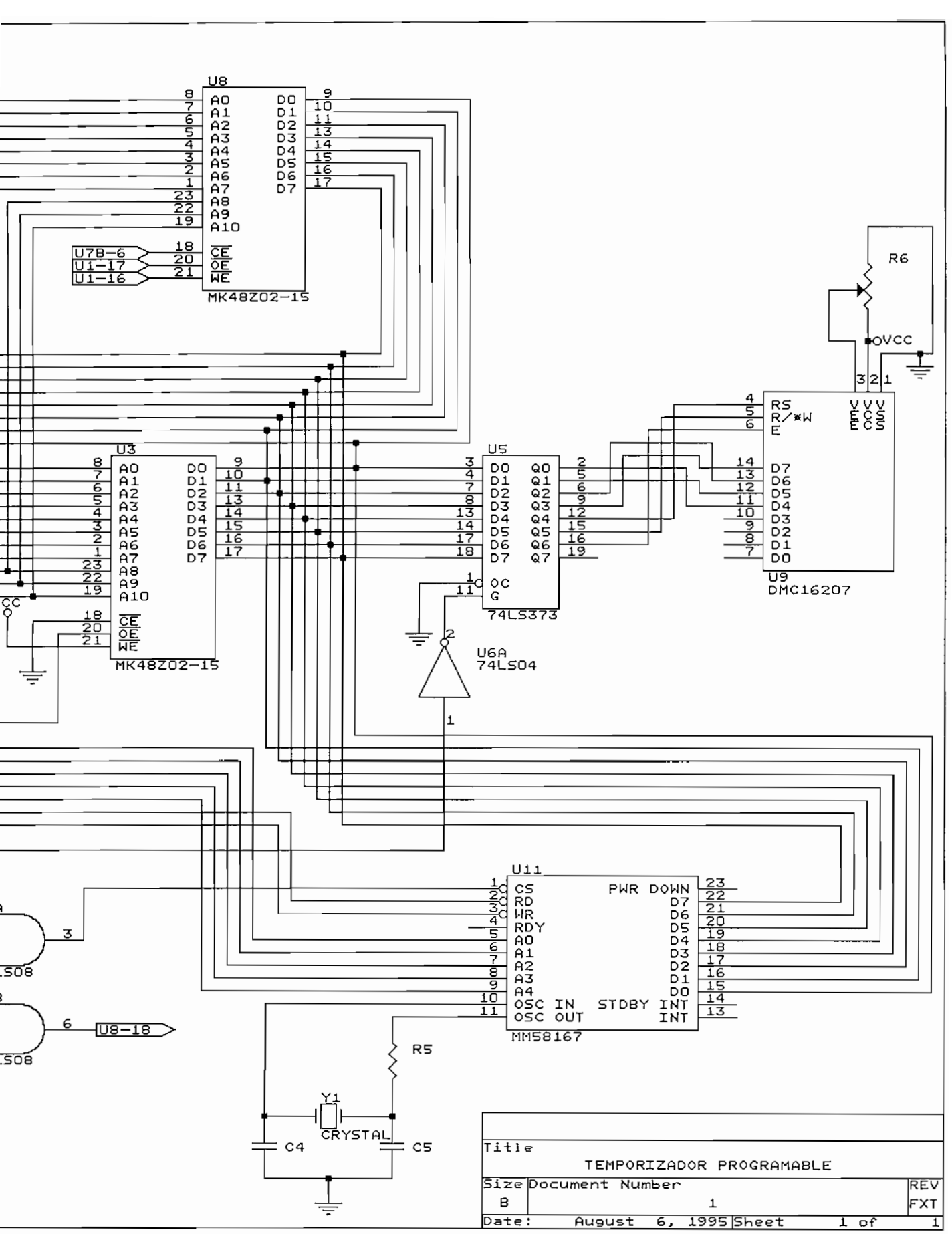
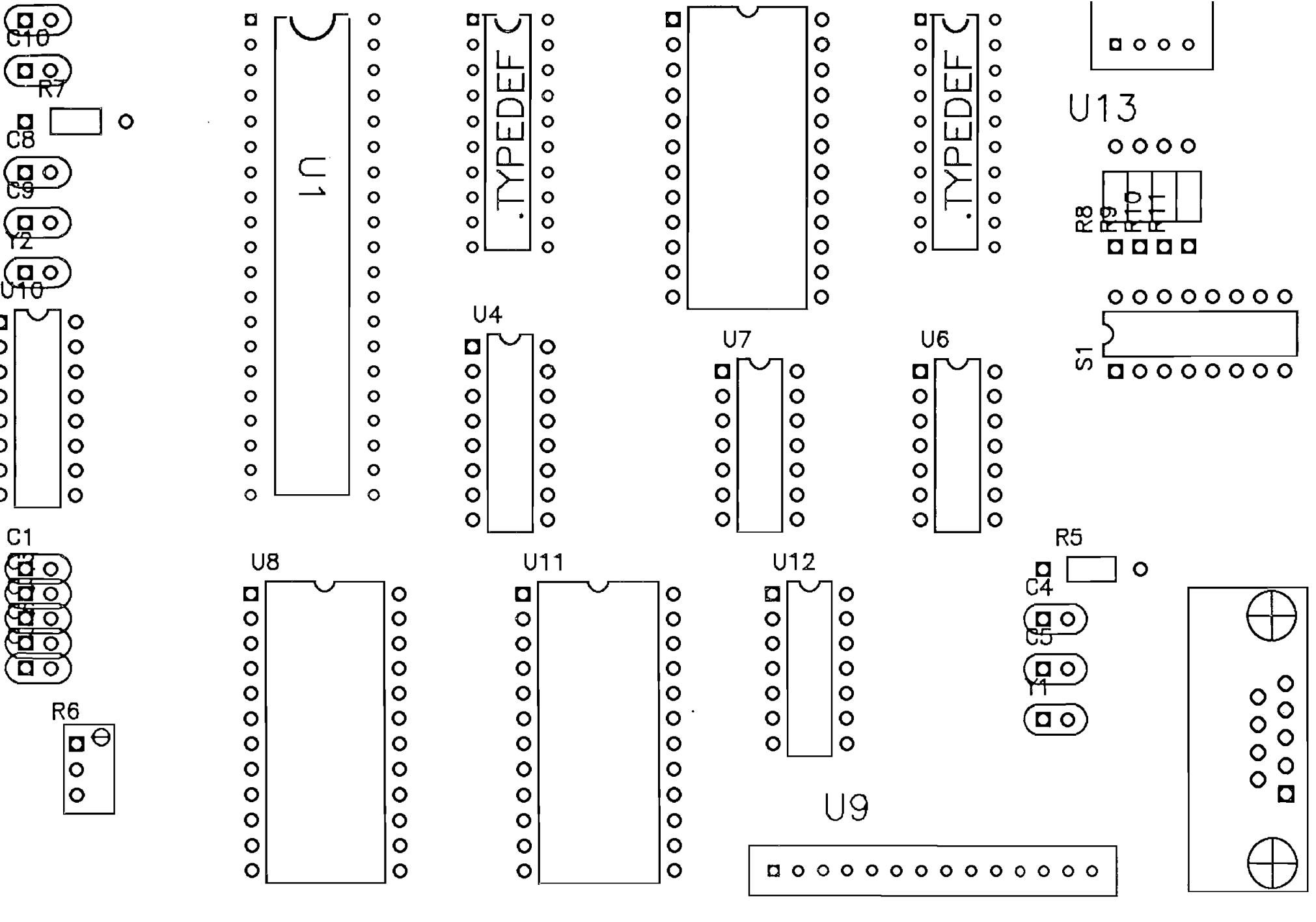


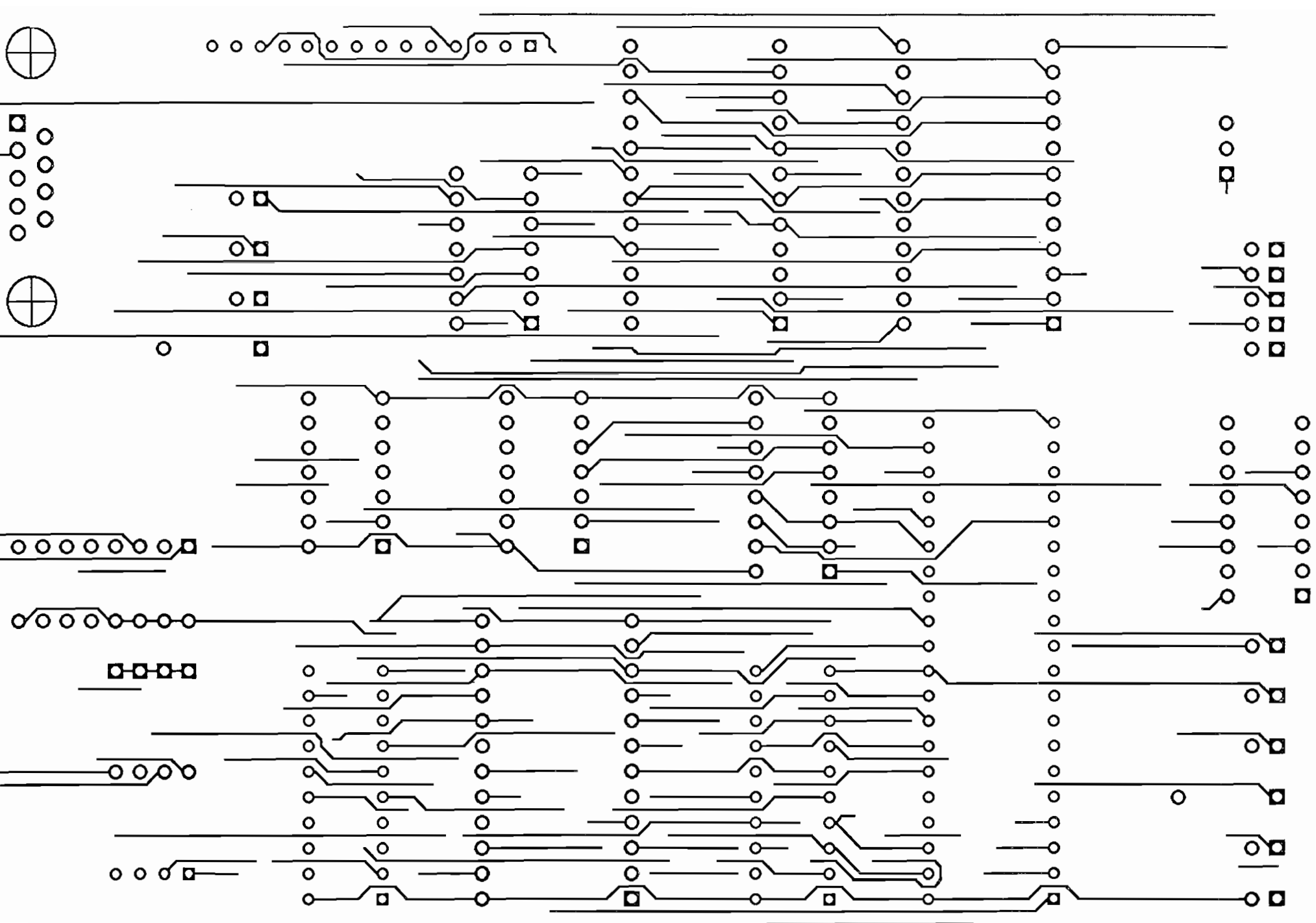
FIGURE 4. Standby Interrupt is Enabled (ON) for Normal Operation and Disabled for Standby Operation

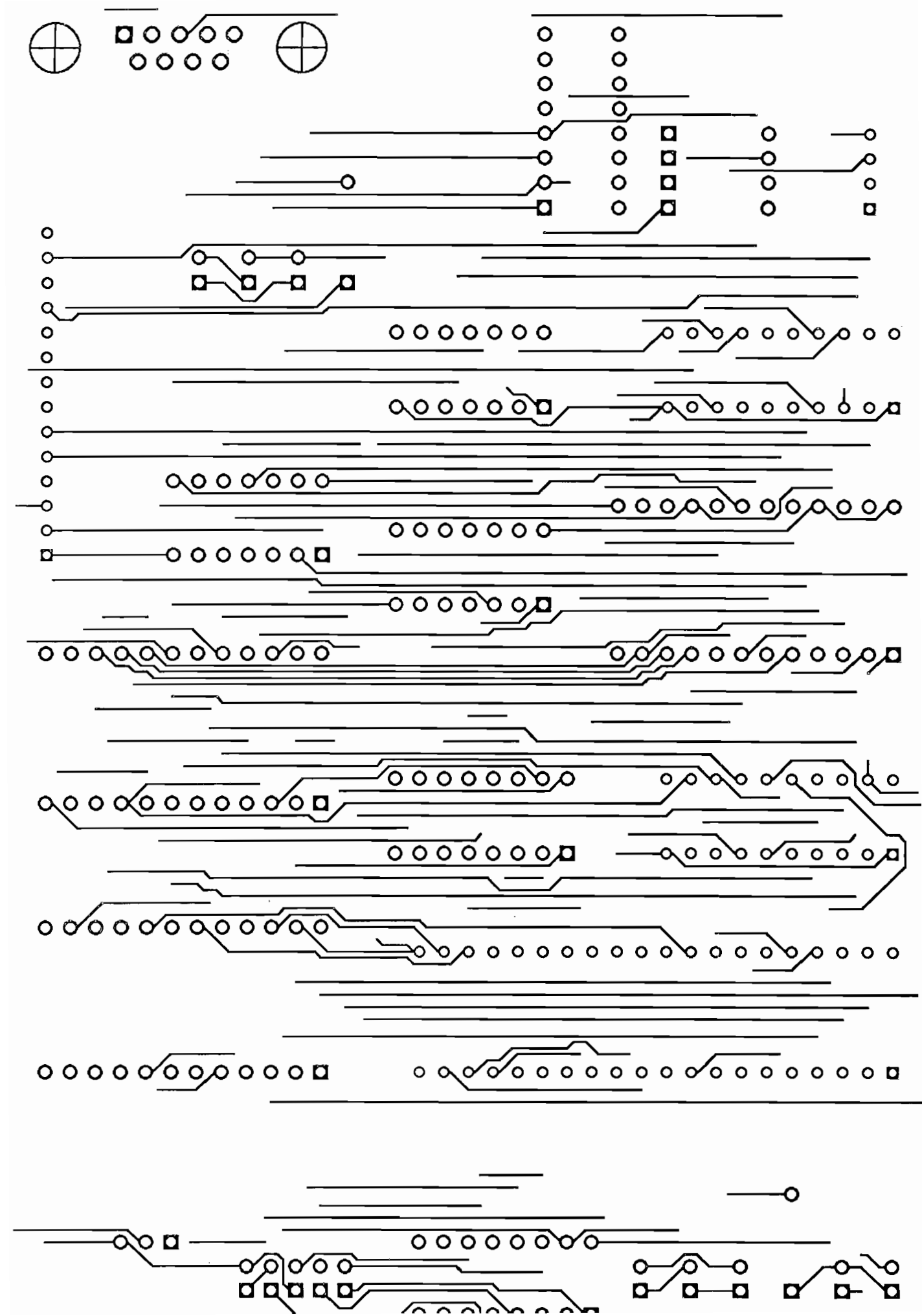
ANEXO 5

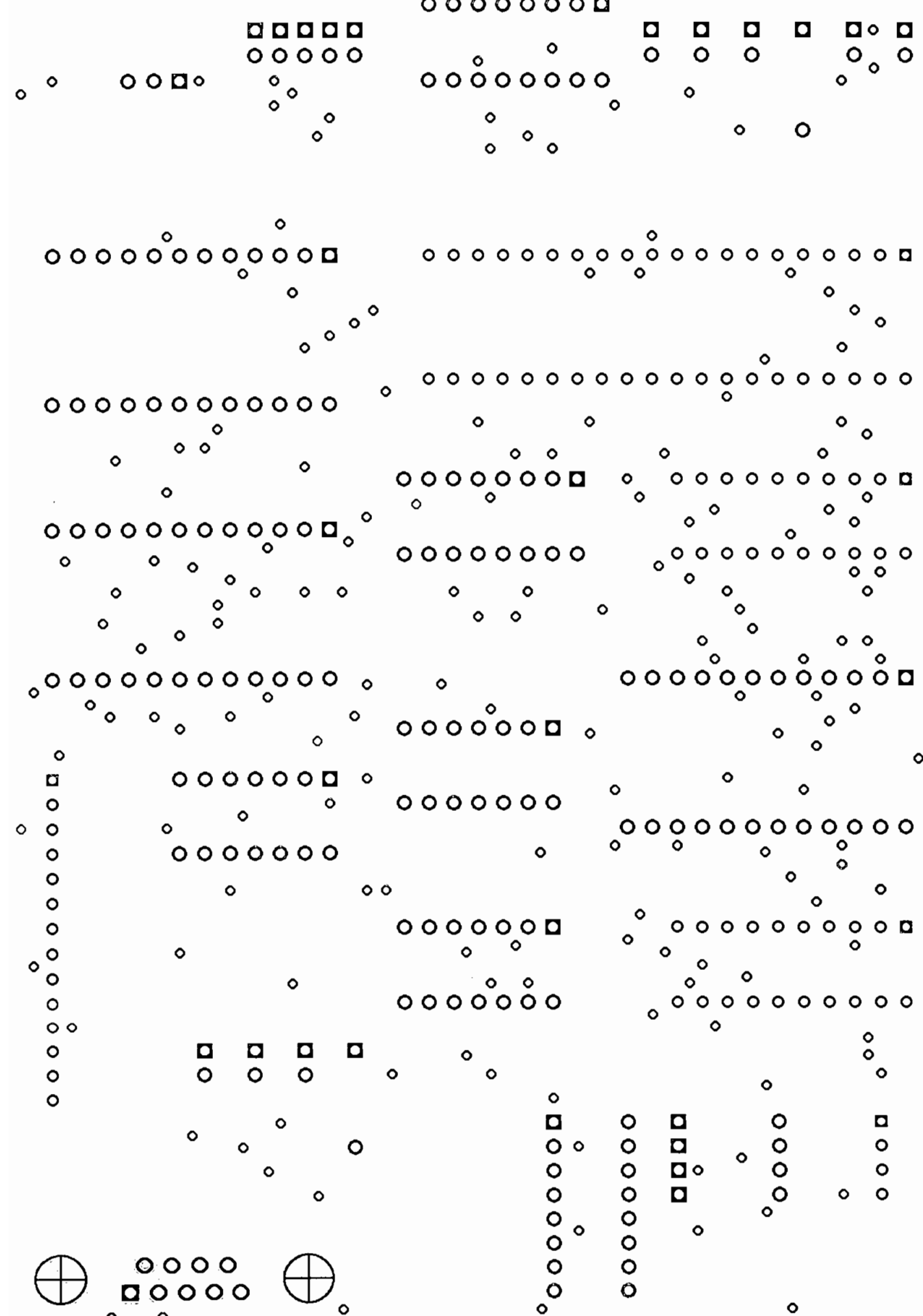

VCC
U10
C1+
C1-
C2+
C2-
T1IN
T2IN
R1OUT
R2OUT
V+
V-
T1OUT
T2OUT
R1IN
R2IN
MAX232
C1
C2
C6
C7
C3
U1-11
U1-10
P1
CONNECTOR DB9
C8
C9
Y2
S2
C10
R7
U1
EA/VP
X1
X2
RESET
INT0
INT1
T0
T1
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
RD
WR
PSEN
ALE/P
TXD
RXD
8031
U2
D0
D1
D2
D3
D4
D5
D6
D7
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
OC
G
74LS373
U10-11
U10-12
U4A
U4B
A
B
G
Y0
Y1
Y2
Y3
74LS139
U8-20
U8-21
R8
R9
R10
R11
S1
SW DIP-8
U12A
U12B
U12C
U12D
74LS00
U13
RELE1
RELE2
RELE3
RELE4
RELES

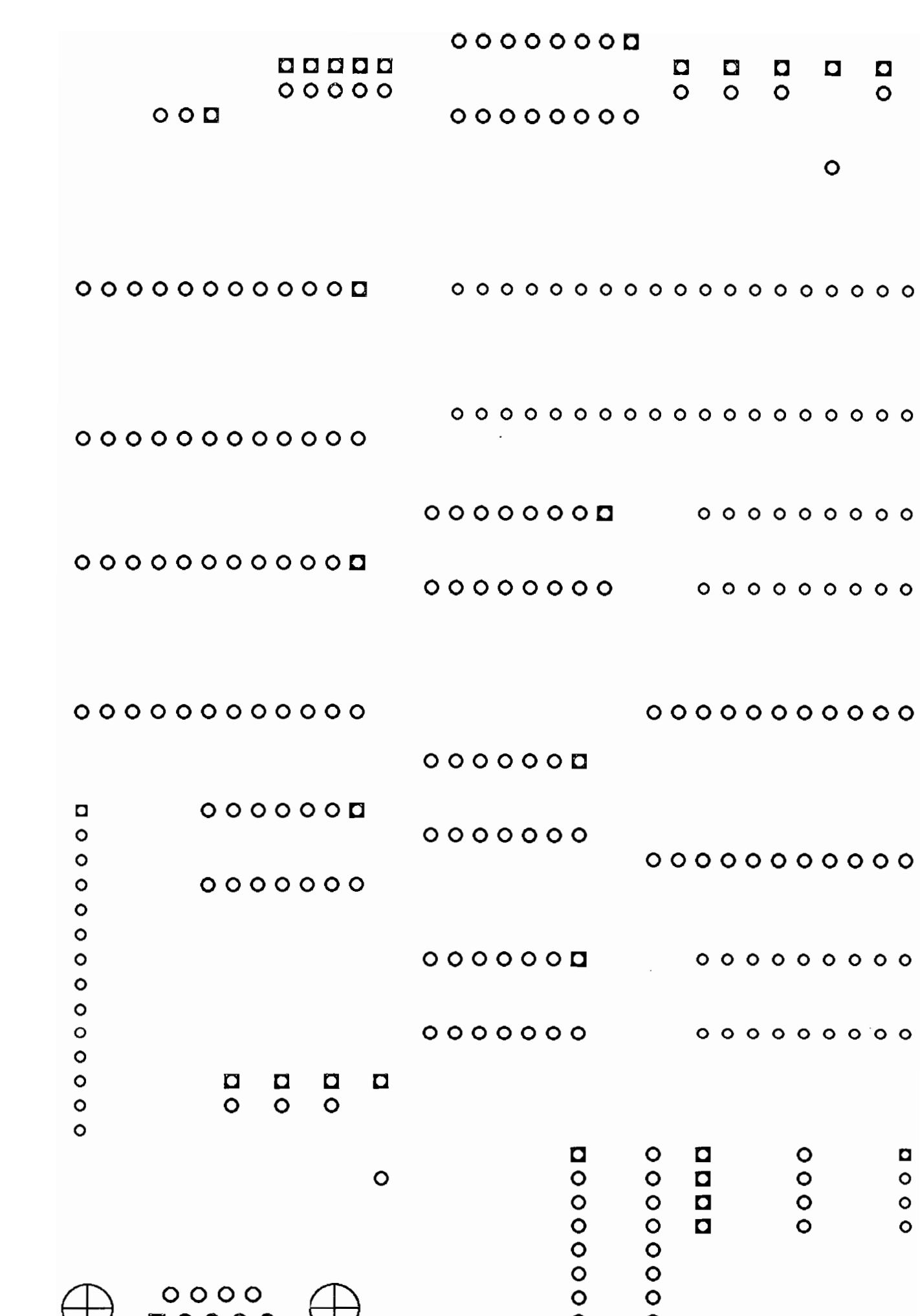

U8
MK48Z02-15
U7B-6
U1-17
U1-16
U3
MK48Z02-15
U5
74LS373
U6A
74LS04
R6
VCC
U9
DMC16207
U11
MM58167
PWR DOWN
STDBY INT
OSC IN
OSC OUT
R5
Y1
CRYSTAL
C4
C5
U8-18
Title
TEMPORIZADOR PROGRAMABLE
Size
B
Document Number
1
REV
FXT
Date: August 6, 1995
Sheet 1 of 1


C10
R7
C8
C9
Y2
U10
U1
.TYPEDEF
.TYPEDEF
U13
R8
R9
R10
R11
S1
U4
U7
U6
C1
C3
R6
U8
U11
U12
R5
C4
C5
Y1
U9

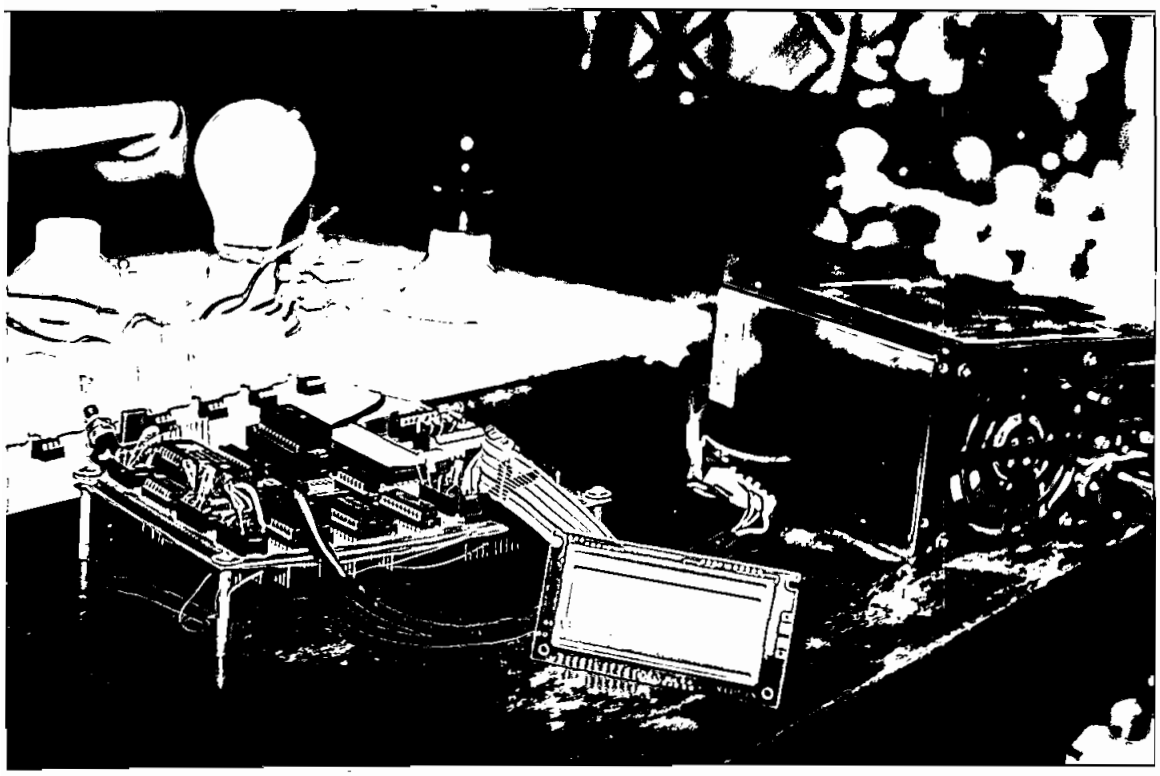








FOTOGRAFIA DEL TEMPORIZADOR PROGRAMABLE



ANEXO 6

MANUAL DEL USUARIO

El temporizador cuenta con un interfaz gráfico mediante el cual se puede controlar su funcionamiento.

El programa se puede utilizar en cualquier PC.

Los archivos que se necesitan para su funcionamiento son los manejadores de gráficos, dependiendo del computador con el que se cuente (extensión BGI) y el programa ejecutable TEMPO.EXE.

Se crean además tres archivos con extensión DAT, los cuales son:

1) IG_HORA.DAT en el que se encuentra la información de la igualación del reloj en tiempo real.
2) D_TEMPO.DAT en que se encuentra la programación de los eventos a ser descargados en el temporizador.
3) REGIS.DAT en el que se encuentra la misma programación de los eventos pero no codificados para ser recuperados en el programa del PC.

Inicialmente se debe digitar la siguiente instrucción en el prompt del computador:

> tempo

Posteriormente aparecerán en la pantalla mensajes iniciales del programa del temporizador, para presentarse la pantalla de la figura 1.

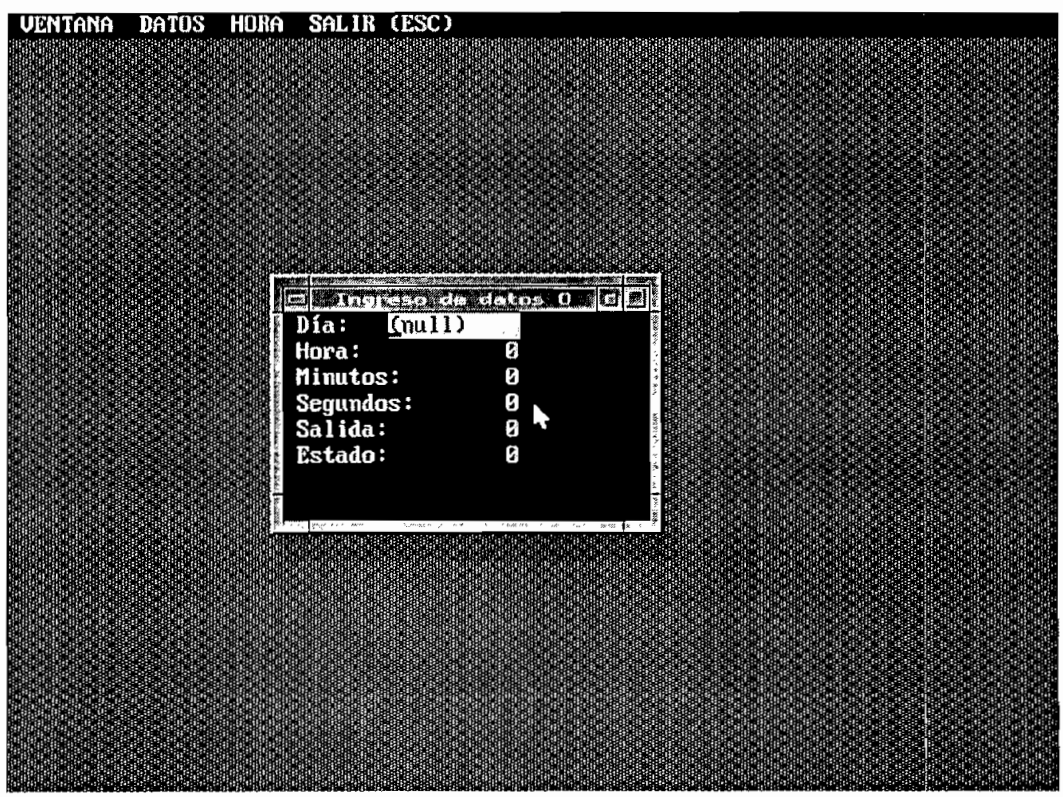


Figura 1. Pantalla inicial del interfaz gráfico

Este interfaz proporciona las siguientes facilidades:

- Permite el manejo del mouse para poder pasar de un campo a otro. Si no se dispone de un mouse para su manejo existen teclas que también posibilitan realizar funciones similares.

- Con la tecla F10 se puede ingresar al menú donde se dispone de las opciones de: Ventana, Datos, Hora y Salir (ESC).

Cada una de estas opciones contenidas en el menú principal posee submenúes. Con la tecla ESC se puede salir de los niveles inferiores hacia los superiores.

- A todas las ventanas se las puede minimizar, maximizar, mover, cambiar de tamaño.

El procedimiento a seguirse para el ingreso de datos es el siguiente:

1) Como se puede apreciar en la figura 1 de este manual, existe una subventana dentro de la principal en la cual se puede realizar el ingreso de varios campos. El primero corresponde al día. Mediante la tecla F9 se despliega una subventana donde se elige el día en el cual se desea que ocurra el evento. Esto se aprecia en la figura 2. Los demás campos corresponden a la hora, minutos, segundos, salida y estado.

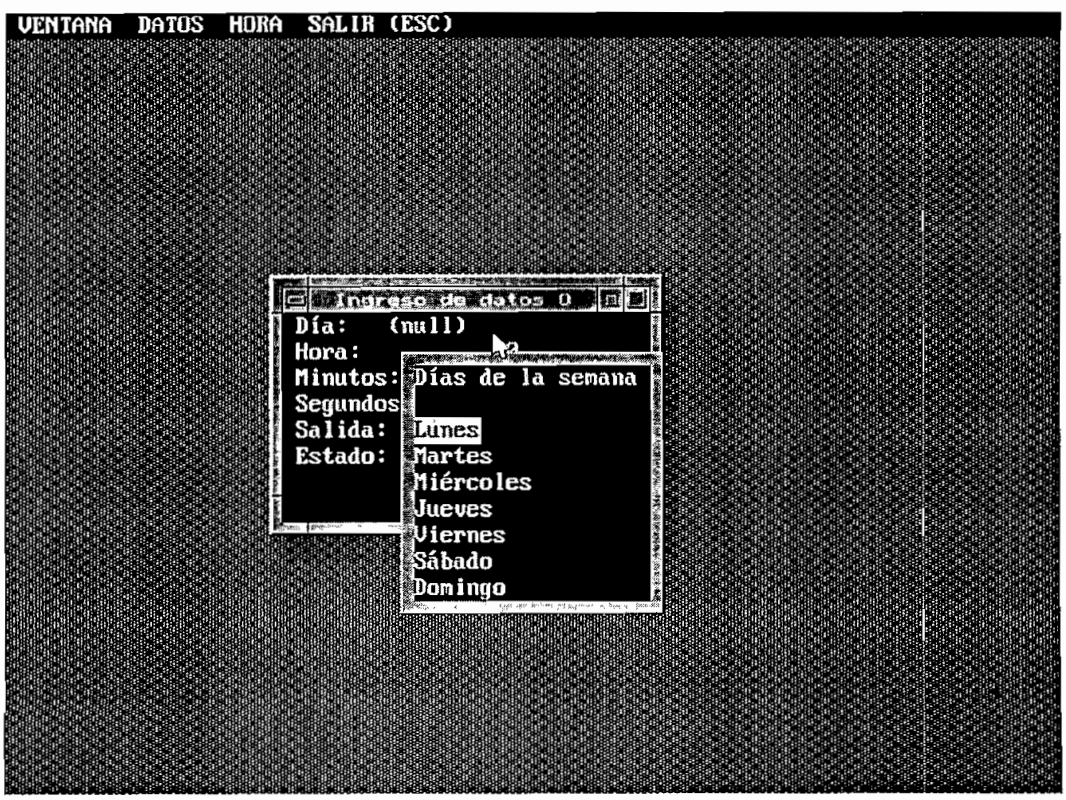


Figura 2. Ingreso del campo de día de la semana

Cada uno de estos campos tiene las siguientes restricciones:

hora de 0 a 23

minutos de 0 a 59

segundos de 0 a 59

salida de 0 a 3

estado de 0 a 1

donde el término *salida* corresponde al relé que se desea controlar y el término *estado* si el relé debe activarse o desactivarse.

Existen 224 registros que pueden ser ingresados, lo que corresponde al los previstos por la tesis, pudiéndose incrementar éstos en un futuro.

2) Una vez ingresado un registro se puede accesar al siguiente presionando la tecla F2 o ingresando a la opción Datos - Ingreso como se puede apreciar en la figura 3.

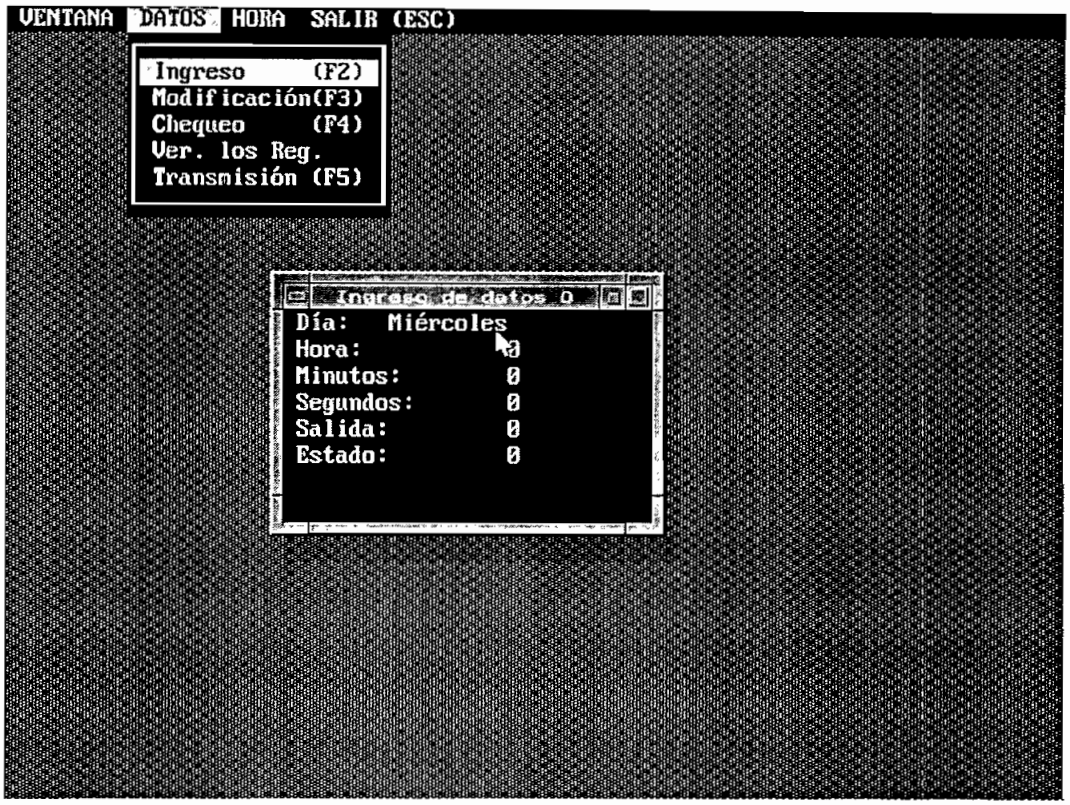


Figura 3. Submenú para ingreso de datos

3) De acuerdo con lo que se ilustra en la figura 3 existe la posibilidad de modificar y chequear cada uno de los registros ya ingresados mediante la pulsación de las teclas F3 y F4 respectivamente.

Además el interfaz permite la visualización de todos los registros, proceso que se ilustra en la figura 4.

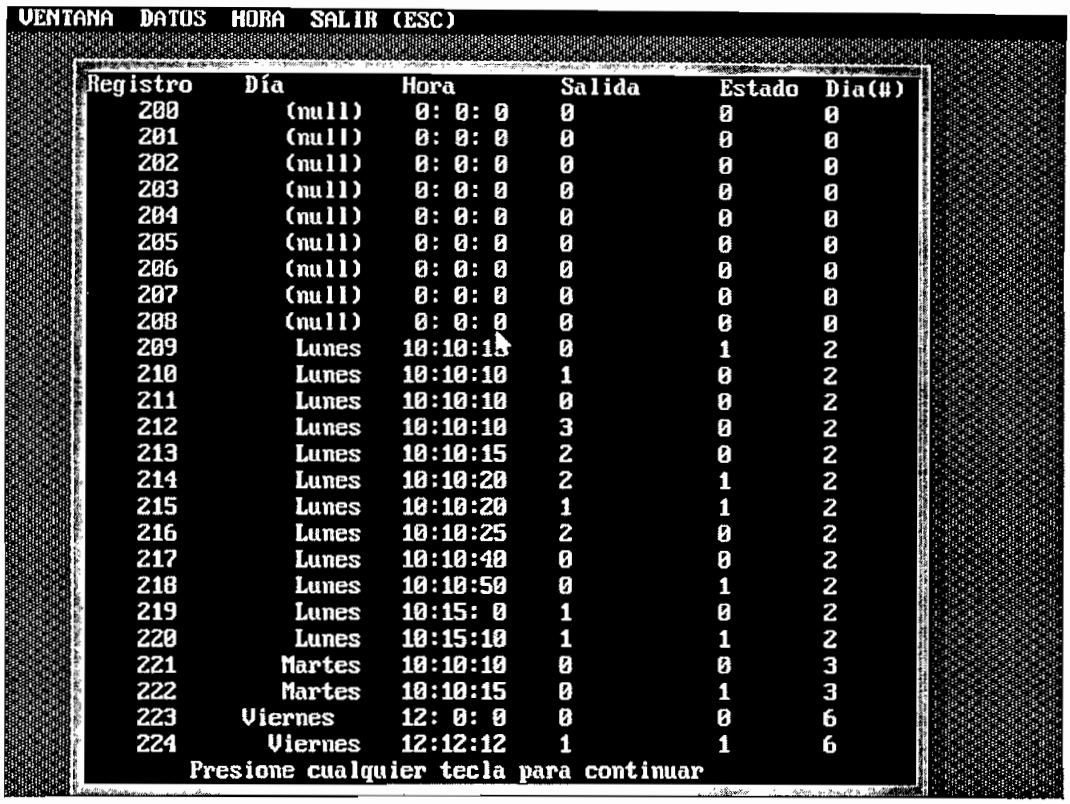


Figura 4. Visualización de todos los registros

5) Una vez que se hayan introducido todos los datos se los puede descargar al temporizador mediante la interfaz serial presionando la tecla F5 o Datos - Transmisión.

Para la comunicación serial, en primer lugar se tiene que definir el pórtico que se va a utilizar; el pórtico es por defecto el COM1, pudiendo elegir el COM2 si fuese necesario. Una vez realizada esta operación, el

computador ordenará los datos y los enviará al temporizador para almacenarlos en la NVRAM, como se ilustra en la figura 5.

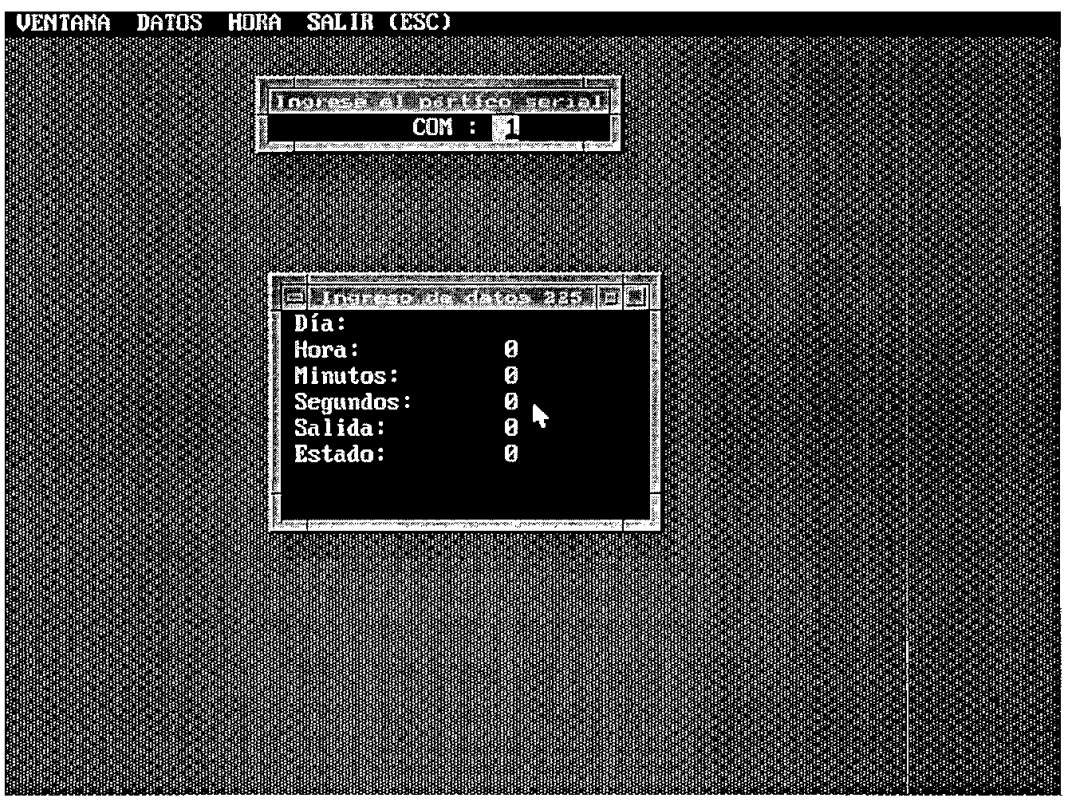


Figura 5. Inicio de la transmisión serial

6) De manera similar se ingresa la hora para igualar el reloj en tiempo real. Mediante la tecla F6 o con el ingreso al menú principal con las opciones **Hora - Ingreso** se procederá a introducir los datos necesarios, como se aprecia en la figura 6. Una vez terminada la

programación de la hora se transmite serialmente al temporizador con las opciones Hora - Transmisión pudiéndose elegir de igual manera que en el caso anterior el pórtico a utilizarse.

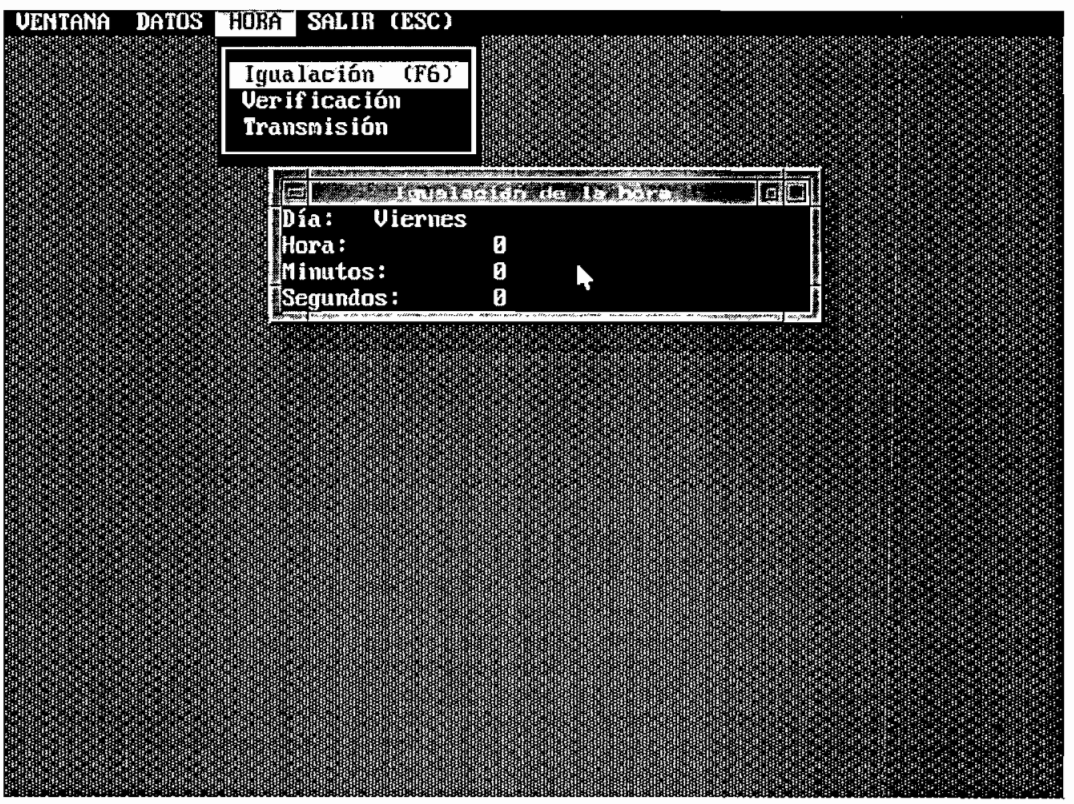


Figura 6. Ventana para igualación de la hora

El ingreso del día de la semana es similar al de los datos descrito anteriormente, mediante la tecla F9.

Todos los pasos descritos son los necesarios a seguir para la programación del temporizador.

Adicionalmente, el interfaz posee la opción de Ventana y de Salir. En la primera se tiene tres opciones que son Información (F1), Minimizar (F7) y Maximizar (F8), como se ilustra en la figura 7.

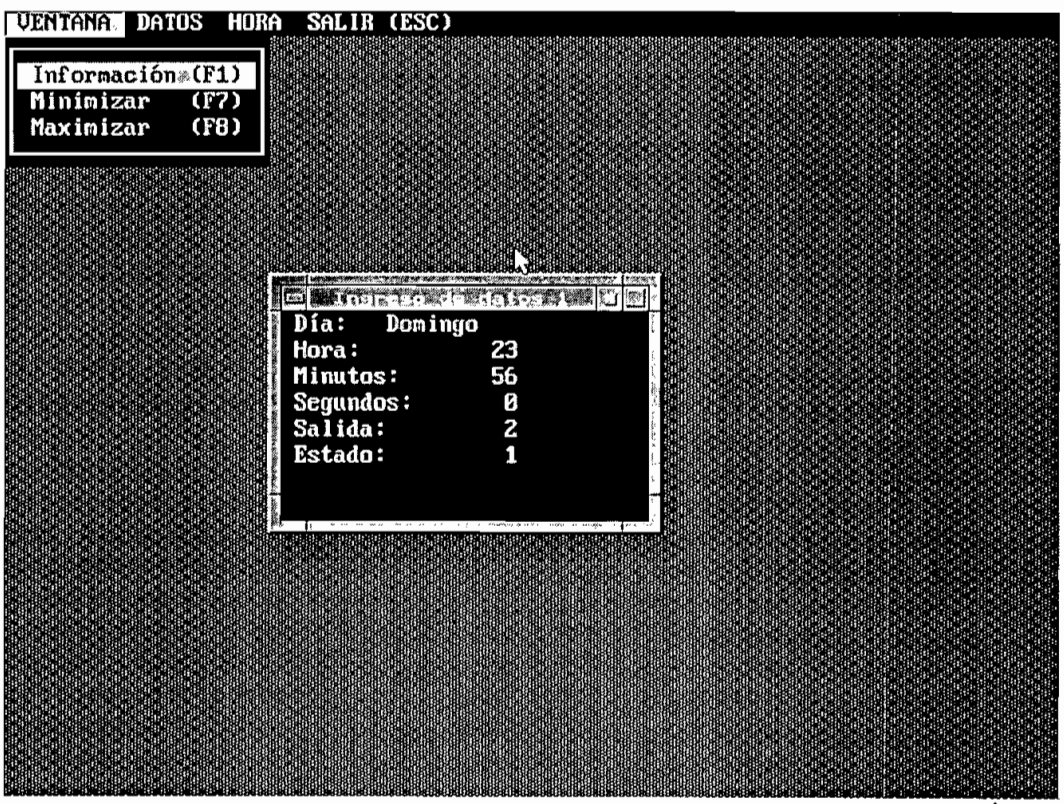


Figura 7. Submenú Ventana

Por último la opción salir permite terminar el programa y salir al sistema operativo.

Como facilidades adicionales para la operación del temporizador existe un botón que sirve para resetear todo el circuito en caso de funcionamiento erróneo. Además se dispone de 4 dip switch para desconectar las salidas que manejan las cargas en caso de requerirse. Estos corresponden a los dip switch 1, 2, 3 y 4. El dip switch 8 puesto en ON sirve para realizar un barrido de las salidas en ese instante específico y mostrarlo en el display.

BIBLIOGRAFIA

1.- GONZALEZ JOSE ADOLFO, Introducción a los microcontroladores, McGraw-Hill, Madrid, 1992.

2.- ING. M.SC. RICARDO ZELENOVSKY, El PC para ingenieros electrónicos, ESPE, Quito, 1994.

3.- FOLKES DELROY- BANDA HUGO, Programming in C, The University of Dundee, 1990.

4.- MANDADO ENRIQUE, Sistemas electrónicos digitales, Publicaciones Marcombo, México, 1987.

5.- LIANT SOFTWARE CORPORATION, The C-scape Interface Managment System Manual, Massachusetts, 1992.

6.- LIANT SOFTWARE CORPORATION, Function Reference, Vol I - II, Massachusetts, 1992.

7.- LIANT SOFTWARE CORPORATION, Oakland Windowing Library Manual, Massachusetts, 1992.

8.- MORENO CESAR - GONZALEZ FABIO, Depurador y kit de desarrollo para el microcontrolador DS5000T, EPN, Quito, 1991.

9.- BUBAN PETER - MALVINO ALBERT - SCHMITT MARSHALL, Electricidad y electrónica, aplicaciones prácticas, McGraw - Hill, México, 1990.

10.- VELARDE JAIME, Apuntes de microcontroladores I, EPN, 1992.