ESCUELA POLITECNICA NACIONAL
FACULTAD DE INGENIERIA ELECTRICA

"DISEÑO E IMPLEMENTACION DE UN EQUIPO DIDACTICO BASADO EN MICROPROCESADOR PARA EL AREA DE SISTEMAS DIGITALES"

TESIS PREVIA LA OBTENCION DEL TITULO DE INGENIERO EN LA ESPECIALIZACION DE ELECTRONICA Y TELECOMUNICACIONES

CARLOS GUALBERTO BADILLO SALAZAR

MARZO 1998

Certifico que el presente trabajo ha sido elaborado en su totalidad por el Sr. Carlos G. Badillo Salazar.

Ing. Jaime Velarde
Director

DEDICATORIA

A mis padres.

AGRADECIMIENTO.

Agradezco a DIOS sin cuya presencia y apoyo ninguna labor se completa, a todas las personas que de una u otra manera han contribuido para la realización de esta tesis y un agradecimiento especial al Ing. Jaime Velarde por su paciencia y acertada dirección del presente trabajo.

INDICE.

CAPITULO III.

SOFTWARE.

CAPITULO IV

RESULTADOS EXPERIMENTALES.

PROLOGO.

El equipo implementado en el presente proyecto tiene como propósito fundamental su utilización en el laboratorio de sistemas digitales, como una ayuda para la formación de los estudiantes, de modo que los resultados de las prácticas puedan ser visualizados con mayor facilidad, al mismo tiempo que se proporcionan una serie de señales digitales que resulten de utilidad en este tipo de prácticas.

El proyecto se basa en las capacidades de un microcontrolador y enfatiza en todo momento en el manejo de las señales de entrada salida como la forma de interactuar con el medio externo, siempre considerando señales digitales. Se ha aprovechado las capacidades de multiplexación por software que brinda un microcontrolador para el manejo de matrices de leds, que conforman el visualizador del equipo, logrando de este modo incrementar la disponibilidad de caracteres que se pueden desplegar.

Básicamente la descripción del sistema en forma General se la realiza en el primer capítulo, en el cual se explican cada una de las funciones de las que dispone el estudiante al momento de utilizar el prototipo implementado. Se incluye en este caso la información teórica pertinente al tipo de función que se necesita implementar, en sus partes más relevantes.

En el capítulo II se puede encontrar una descripción más detallada de la forma en que se ha implementado cada una de las funciones previstas desde el punto de vista del Hardware, incluyendo la información técnica más relevante de aquellos dispositivos y elementos electrónicos utilizados en cada etapa del prototipo.

La operación de todo el equipo depende de la programación que se realice en el microcontrolador y es en el capítulo III, en el que se detalla la forma como se concibió cada una de las rutinas en lenguaje de máquina que permiten la operación del dispositivo, cada una de las rutinas y subrutinas trabaja como un bloque independiente ligadas todas por medio de lo que se

constituye en el programa principal. La interacción del usuario se realiza prioritariamente a través del teclado y los códigos generados por este indican cual de las funciones y por tanto rutinas entra en operación en determinado momento.

El capítulo IV por otra parte presenta los resultados obtenidos al realizar las diferentes pruebas con el prototipo, determinando en forma práctica, las frecuencias de barrido de las matrices, frecuencias de señales que se pueden generar, especificaciones de las limitaciones propias del prototipo, especificandose en general los limites de las señales con las que puede trabajar el equipo.

La experiencia lograda al desarrollar el prototipo origina una serie de conclusiones respecto a la forma de desarrollar equipos basados en microprocesadores, así como recomendaciones que se pueden aplicar al desarrollo de proyectos de similares características, todas estas se aparecen en el capítulo V de el presente trabajo.

INTRODUCCION.

La necesidad de contar con un medio de interacción más amigable entre los circuitos de prueba Implementados en los laboratorios en base a elementos digitales y los estudiantes, de modo que los resultados de este tipo de circuitos de prueba puedan ser analizados de una manera más rápida origina el planteamiento del presente proyecto. Para la implementación de un dispositivo electrónico que pueda satisfacer los requerimientos que se originan de el tipo de aplicación planteada se elige el utilizar las bondades de los sistemas basados en microprocesadores, en cuanto a versatilidad y capacidad de procesamiento.

Cualquier computador digital de propósito general consiste de tres componentes principales: la Unidad Central de Procesamiento (CPU), la memoria de datos y la memoria de programa y el sistema de entrada salida. La Unidad Central de Procesamiento controla el flujo de información entre los diferentes componentes del computador además de realizar las operaciones digitales; la mayor parte del procesamiento se realiza en la Unidad Aritmética y Lógica dentro de la Unidad Central de Procesamiento.

Un microprocesador esta conformado por una Unidad Central de Procesamiento construida en un solo dispositivo semiconductor o circuito integrado; el microprocesador es un dispositivo de propósito General que se puede usar en múltiples aplicaciones,

En función de la aplicación que se quiera implementar en base a un microprocesador se seleccionan los dispositivos de memoria y de entrada salida que e interconectan para formar lo que se denomina microcomputador.

Un Microcontrolador por otra parte es una computadora completa construida en un solo circuito integrado que contiene los subsistemas de memoria y de entrada salida, convirtiéndose de esta forma en un dispositivo especializados que puede incrementar las capacidades de procesamiento

digital en aplicaciones de control.

Los microcontroladores son usados normalmente como como dispositivos embebidos dentro de una aplicación especializada, por ejemplo se los usa en sistemas de encendido de automóviles, controles de exposición y enfoque de cámaras fotográficas, estos implican la existencia en un mismo circuito integrado de facilidades tales como puertos seriales, puertos de entrada/salida paralelos, temporizadores, contadores, control de interrupciones, memoria de acceso aleatorio, memoria solo de escritura, etc.

Las aplicaciones de control embebidas sirven también para distinguir a los microcontroladores de los dispositivos de propósito general o microprocesadores. Los sistemas embebidos en general requieren de capacidades de operación en tiempo real y multitarea.

La operación en tiempo real implica que el microcontrolador debe estar en capacidad de recibir y procesar las señales desde el medio ambiente exterior cuando estas están disponibles, o lo que es lo mismo el medio externo en ningún momento deberá esperar para que el controlador este disponible. Además microcontrolador debe suministrar las señales de salida con la rapidez suficiente cuando estas son requeridas, el resto del sistema no debe sufrir retrasos debido a espera de estas señales de control, de modo que el microcontrolador no se constituya en un cuello de botella para el flujo del proceso.

En cuanto a las capacidades de multitarea, se refiere a la capacidad de realizar varias funciones en forma simultanea o cuasi-simultanea, en general debido que el controlador deberá estar monitoreando algunas variables a la vez, con la capacidad de responder en forma apropiada a los cambios que se produzcan en cada una de ellas.

CAPITULO I:

DESCRIPCION DEL SISTEMA

La idea central del presente proyecto, es aprovechar las ventajas que proporciona el uso de microcontroladores en la creación de un equipo que resulte de utilidad en un laboratorio, a la vez que facilite la comprensión del funcionamiento de los sistemas digitales y de los sistemas basados en microprocesadores.

Resulta necesario en todo momento tener presente que el dispositivo en cuestión se lo diseña e implementa con la finalidad específica de destinarlo a un uso didáctico y como complemento a las prácticas de laboratorio del área de sistemas digitales, lo cual determina la selección de alternativas que a primera vista podrían resultar inadecuadas o simplemente no las más óptimas.

El equipo ha sido pensado para cumplir varias funciones que resultan de utilidad en el campo digital; algunas de las cuales son: generación de secuencias lógicas, generación de señales de reloj, captación de datos binarios, manejo de cargas A.C., las mismas que se describen a continuación.

1.1.- ADQUISICION DE DATOS Y PRESENTACION DE RESULTADOS.

Si un sistema de tipo digital se construye con propósitos de prueba, es de primordial importancia el disponer de los medios para observar los resultados, sean finales o intermedios que se obtienen con la interconexión de los diferentes circuitos integrados y/o componentes electrónicos discretos, de modo que se corrobore su correcto funcionamiento o se agilite la detección de los posibles errores cometidos y por lo mismo de su corrección.

Basados en este criterio el dispositivo esta en capacidad de tomar señales digitales desde los circuitos armados en las prácticas de laboratorio. El valor de estos datos binarios, al momento de ingresar al prototipo para su procesamiento, se los puede observar mediante dos leds que indican el estado de la señal entrante, del modo siguiente:

1L	Led rojo encendido
0L	Led verde encendido

Una vez aceptados estos datos pueden ser usados en tres formas diferentes.

a) El estado de los LEDs puede constituirse en el resultado final que desea observarse y por lo tanto no se realizara ningún proceso adicional con estas señales, en otras palabras la representación mediante los estados de los leds de la etapa de captura de datos es suficiente para los fines de la experimentación en curso.

b) Desde un código binario pueden ser convertidos a formatos BCD o Hexadecimal, y de este modo aparecer en un display propio del equipo en una presentación mas amigable para los usuarios, facilitando al mismo tiempo familiarizar al estudiante con la presentación simultanea de las señales en forma binaria y Hexadecimal o BCD.

c) Se los puede utilizar como las señales que determinan alguna forma de control para las cargas de A.C. que puede manejar el equipo. Estas señales provendrían desde sensores o al menos podrían simular tales situaciones.

1.1.1.- EL TECLADO.

En sistemas basados en microprocesadores la forma más común de suministrar al sistema las informaciones requeridas en cuanto a selecciones básicas, control de programas e introducción de variables es mediante un teclado.

El teclado se constituye en la vía de comunicación entre el medio externo y el equipo. El usuario determinara el tipo de tratamiento que recibirán los datos capturados por el equipo o el tipo de función que este debe ejecutar, al ingresar un código preestablecido mediante el teclado.

Si bien se podría utilizar un teclado tan simple como tres teclas las cuales cada vez que son presionadas generen en la pantalla de visualización varias opciones de las cuales el usuario selecciona la más adecuada a sus necesidades, para efectos didácticos resulta más apropiado el uso de un teclado compuesto por un total de 20 teclas, de modo que los datos sean suministrados en forma de códigos hexadecimales lo cual sirve para familiarizar al usuario con este tipo de codificación. En consecuencia las primeras 16 teclas del arreglo corresponderán a los valores comprendidos entre 0 y F h.

La cuatro teclas adicionales constituyen lo que puede denominarse como el bloque de control del equipo, a cada una de ellas le corresponderá un código hexadecimal específico, de modo que al recibirlo el equipo le da el tratamiento propio de un comando para la realización de una tarea específica según el siguiente cuadro:

NOMBRE DE LA TECLA	CODIGO HEX	DESCRIPCION
FUNC	10H	Selecciona la función a ejecutarse
EXE	11H	Inicia la ejecución de una tarea
SEL	12H	Selección de alternativas disponibles
STOP	13H	Detiene tareas en ejecución y retorna al menú principal

Tabla 1.1. Asignación de funcionalidad a las teclas

1.1.2.- PRESENTACION VISUAL.-

Otro componente clave del dispositivo es la presentación visual mediante la cual el operador examina los resultados y recibe la confirmación de las funciones seleccionadas o datos suministrados. Normalmente se utilizan dispositivos optoelectrónicos que despliegan información alfanumérica sea mediante absorción o emisión de luz.

Varias son las tecnologías actualmente en uso para la creación de tales dispositivos entre las cuales podemos mencionar : elementos fluorescentes al vacío, Plasma de corriente continua, Cristal líquido LCD, arreglos de 7 segmentos,etc.

Los elementos de uso más común para la visualización de información son los displays de 7 segmentos, sin embargo su capacidad para representar caracteres es muy limitada pues solo aquellos caracteres que puedan programarse mediante la combinación de los siete segmentos de Leds en condición de apagado o encendido, podrán ser representados.

Para poder disponer de una variedad más amplia de caracteres se pueden utilizar matrices de LEDs 7x5 adecuadamente dispuestas.

Debido al mayor número de elementos independientes en un visualizador de matriz de puntos, en comparación con los de segmentos, se puede obtener una gama mucho más amplia de caracteres diferentes, los cuales inclusive presentan un mejor aspecto.

En estos visualizadores se utiliza una estructura de filas y columnas como la que se aprecia en la figura 1.1. Sin embargo se incrementa en forma notable el número de elementos individuales emisores de luz que deben ser manejados, basta considerar que en cada arreglo existen 35 leds en lugar de los 7 de el otro tipo de display.

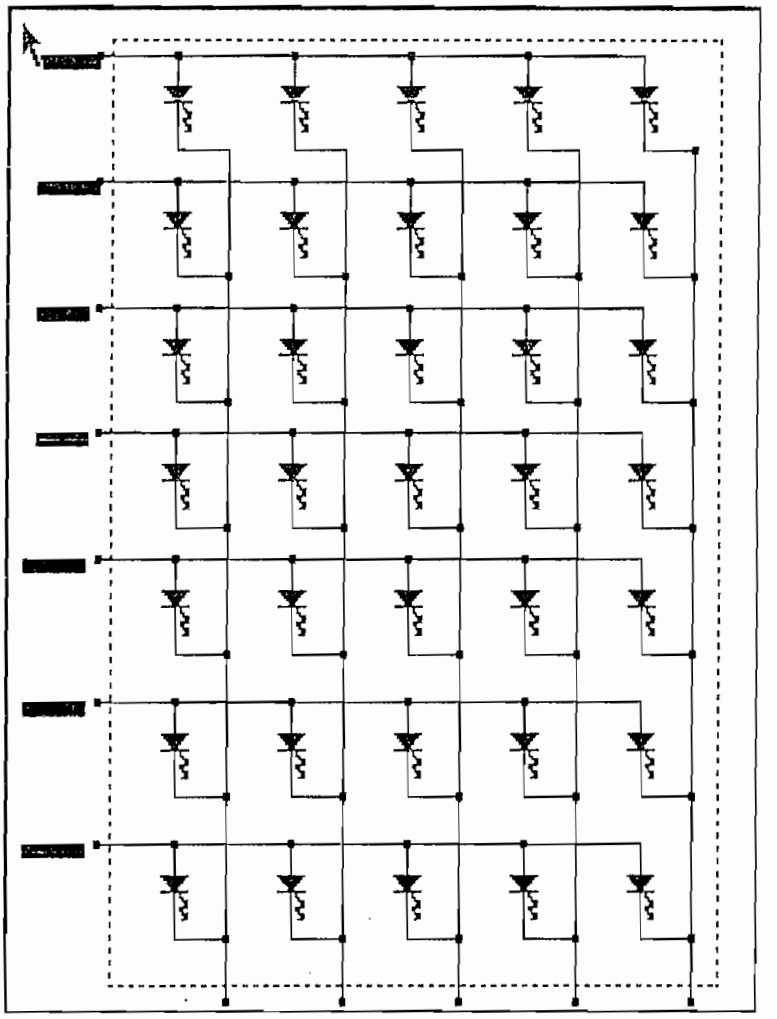


Fig. 1.1. Arreglo de leds de una matriz 7x5.

La generación de los caracteres requiere de una exploración sea de filas o columnas, lo cual implica un multiplexaje de los diferentes elementos del arreglo.

En función de la condición de las señales digitales que alimentan las entradas en paralelo se generan los diferentes caracteres al momento que se cierra el circuito de activación de modo que la corriente puede circular a través de cada uno de los leds y estos pasen a su estado de encendido o apagado. Por ejemplo si necesitamos generar un caracter que representa la letra E, es necesario decidir que tipo de exploración es la que más se adecua a nuestro hardware

y software, si observamos la fig. 1. 2. se puede analizar las implicaciones que tendrá la representación de este caracter.

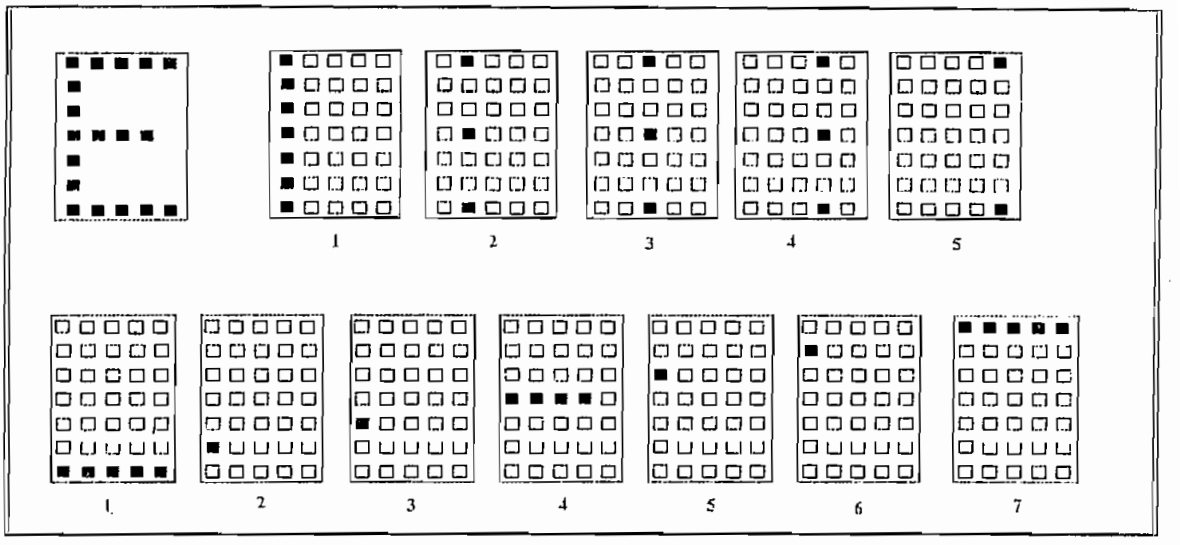


Fig. 1.2. Ejemplo de modalidades de exploración de matrices 7x5

En la parte superior de la figura 1.2 se aprecia la secuencia correspondiente a una exploración horizontal o una exploración por columnas, mientras que en la parte inferior de la misma figura se aprecia una exploración por filas.

Para determinar el tamaño físico de las matrices de LEDs se debe considerar que el equipo esta destinado a uso en laboratorio y con fines didácticos por lo cual se requiere que la visualización de los diferentes datos pueda ser observada a distancia y en forma clara, esta es la razón por la cual se ha preferido usar matrices de LEDs de 40mm x 22mm, siendo también esta la razón para no utilizar dispositivos con tecnología LCD.

1.2.- GENERACION DE SEÑALES DIGITALES DE VARIAS FRECUENCIAS Y SECUENCIAS LOGICAS.

Los sistemas digitales pueden ser clasificados en dos tipos en base a la participación como un elemento del sistema: Secuenciales y Combinacionales.

Los sistemas ***combinacionales*** en los cuales todas las variables de salida dependen exclusivamente del valor actual de las variables de entrada del sistema.

Secuenciales, en los cuales las variables de salida dependen del valor de las variables de entrada en instantes anteriores al actual.

Entonces la diferencia fundamental radica en que un sistema secuencial necesita la intervención del tiempo para la descripción completa del sistema como un medio para memorizar algunos estados de las variables de entrada en función de los cuales se obtienen los resultados.

Por lo tanto para realizar pruebas con circuitos de tipo secuencial se requieren señales de reloj. Es por esta razón el equipo puede proporcionar señales digitales de varias frecuencias.

Estas señales se las puede tomar desde un puerto paralelo con ocho salidas de modo que en el pin correspondiente al bit más significativo se tendrá una señal de la frecuencia seleccionada, mientras que en los restantes se podrá obtener esta señal dividida en pasos de dos.

Los sistemas combinacionales en cambio para ser probados pueden requerir de secuencias lógicas, las mismas que se suministran en forma de ocho bits en paralelo, de los cuales el usuario escoge las que necesita para alimentar el circuito digital bajo prueba.

Estas secuencias podrán ser usadas de dos maneras:

a) La secuencia después de una señal de inicio aparece en forma continua y a una frecuencia determinada, pudiendo detenerla al presionar una tecla para regresar a la pantalla principal de selección de función.
Además se podrá detener la generación en un punto dado mediante la selección de otra de las teclas, e inclusive reinicializar la generación a partir de 0000h, nuevamente esta selección se la hace por teclado.

b) La secuencia se ejecuta paso a paso, para que la salida pase al siguiente valor en el orden de secuencia es necesario que una tecla específica sea presionada.

En esta modalidad es posible también reinicializar la generación de secuencias a su primer valor mediante la selección de una de las teclas.

1.3.- RELOJ SINCRONIZADO CON LA RED.-

Una necesidad que surge con frecuencia en la implementación de sistemas electrónicos es la medición y/o sincronización de eventos que existen en el tiempo tales como ciclos de servicio, velocidad, o frecuencia de trenes de pulsos. Si se analiza como están constituidas estas variables , se puede deducir que su análisis involucra la medición de funciones básicas, tales como anchos de pulso y períodos, mediciones que pueden ser tan simples o complejas como el diseñador quiera programarlas.

Si la aplicación que se esta desarrollando es de tal naturaleza que toda la capacidad de procesamiento del microcontrolador puede dedicarse a la tarea de medición del tiempo, la forma más simple es crear un lazo de temporización dedicado por software, descartando la participación de las interrupciones. Esta opción puede ser usada únicamente cuando todas las otras operaciones del sistema pueden hacerse a un lado por la duración de la función relacionada con el tiempo.

La clave en el lazo de medición consiste en limpiar el registro temporizador e inicializar algunas variables, tales como punteros y contadores, debido a que el proceso de medición de señales periódicas externas normalmente consiste en colectar un número de muestras para su manipulación posterior. El programa de lazo por tanto incrementa un registro y sigue observando el estado de un bit de entrada. Una vez detectada la transición de la señal de entrada, esta almacena la cuenta e inicia una nueva medición, el proceso continua mientras el numero de muestras deseado es recogido.

Una desventaja de esta opción es que la resolución del temporizador se degrada por el tiempo requerido para ejecutar el lazo de medición, el cual consiste de múltiples instrucciones.

El otro método de medición de eventos periódicos, lo puede usar un microcontrolador que dispone de temporizadores internos. En este caso el temporizador es inicializado en cero y el microcontrolador monitorea que ocurra el cambio de estado en un bit de entrada, de forma similar al caso anterior, la cuenta del temporizador es almacenada cuando se detecta el cambio de estado y antes de que el temporizador sea reinicializado y la siguiente muestra pueda ser tomada. En este caso se puede obtener una mejor resolución debido a que el temporizador puede usar una señal de reloj de más alta frecuencia que cuando se usa el método de temporizador por software.

Aprovechando algunas de las características de los microcontroladores antes expuestas se ha planteado la creación de un reloj sincronizado con la red publica de suministro de energía eléctrica resultará de gran ayuda para realizar mediciones de frecuencia y varias pruebas para comprobación de los efectos que tales situaciones producen.

Para tal efecto se partirá de un sistema de detección de cruces por cero del voltaje de línea el cual activará a una de la interrupciones externas del microprocesador, esta será la señal

que active la generación de uno de los flancos de una señal digital que se constituirá en un reloj que permita la temporización de un circuito digital.

1.4.- ENTRADAS SALIDAS PARA COMUNICACIONES SERIAL Y PARALELA.

Un sistema basado en microprocesador necesita establecer comunicación con otros sistemas que pueden constituirse en periféricos que añaden mayores capacidades de procesamiento o funciones desarrolladas con posterioridad al sistema original.

La comunicación tanto serial como paralela son los medios estandarizados a través de los cuales el microprocesador con sus buses tanto de datos como de control puede comunicarse con dispositivos periféricos o con otros equipos.

Específicamente a la circuiteria encargada de generar las señales digitales se la denomina interface y esta envía y recibe las señales digitales a través de un medio físico generalmente un cable.

Resulta evidente la necesidad de una estandarización de las normas para la transmisión de datos, de lo contrario equipos de distinto origen no estarían en capacidad de comunicarse, de ahí surgen los dos tipos de interfaces de uso más común que son serial y paralela, para las cuales se han dictado varias normas por los diferentes organismos internacionales de estandarización.

1.4.1.- COMUNICACION SERIAL.-

La forma más común de comunicación o transmisión serial de datos es la norma EIA-RS232C definida por la Electronic Industry Association, la cual corresponde a la norma V.24 del CCITT.

Según esta norma en una comunicación serial existen dos tipos de dispositivos, como se aprecia en la figura 1.3.

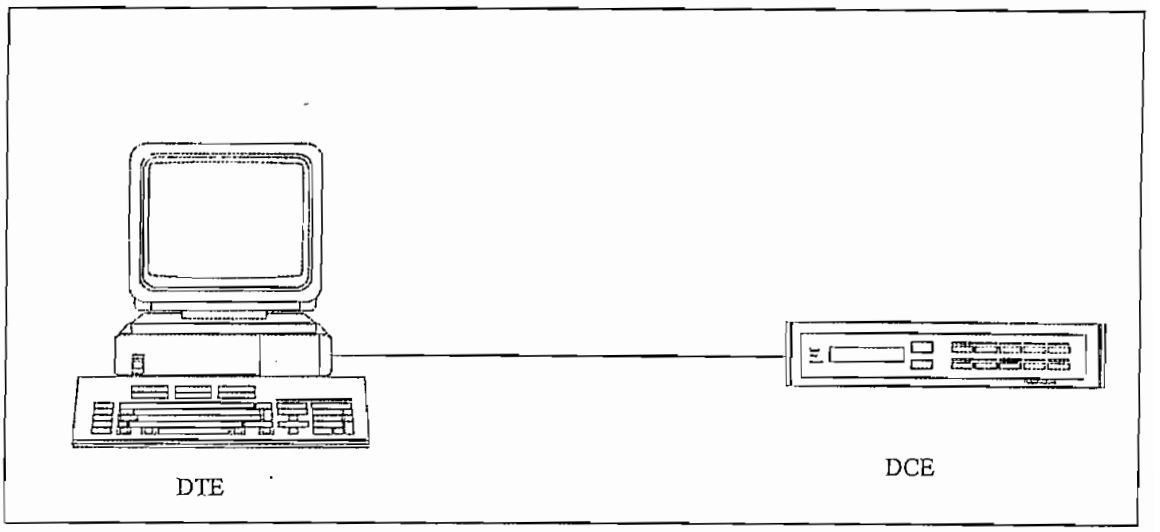


Fig. 1.3. Conexión entre DTE y DCE propia de la norma RS-232.

Cada uno de los cuales cumple una función específica como se detalla a continuación:

DTE es el **Data terminal Equipment**, típicamente un computador o un terminal de computador, el cual controla el proceso de comunicación y se constituye en la fuente y destino de los datos.

DCE es el **Data comunication equipment**, normalmente un modem, que es un equipo que adapta la señal al medio por el cual se va a transmitir la información. Constituido por dispositivos de control, regeneradores y convertidores de señal.

El estándar RS-232 aplicable al intercambio serial de datos binarios entre Equipos terminales de datos DTEs y equipos de comunicación de datos DCEs, especifica entre otras las siguientes características:

- Características de las Señales Eléctricas
- Características mecánicas de la interface
- Descripción funcional de los circuitos de intercambio

Este estándar es aplicable para la transmisión de datos con tazas comprendidas entre 0 y 20.000 bits por segundo como limite superior, y con distancias entre dispositivos que no pueden exceder de los 15 metros.

Se lo usa normalmente para el intercambio de datos, señales de reloj, y señales de control cuando se usa en conjunción con equipo electrónico, cada uno de los cuales tiene una sola señal de retorno común (signal ground), que pueden ser interconectados al punto de interface, y no se la puede aplicar donde se requiere de aislamiento eléctrico entre los equipos a conectados en los extremos opuestos de la interface.

La transferencia de datos binarios se la puede realizar tanto en modalidad sincrónica como asincrónica.

Las características eléctricas que tiene que cumplir los dispositivos bajo este estándar de interface son:

- Las salidas deben tolerar un corto circuito sin sufrir daños.

- La impedancia de salida debe ser mayor o igual a 300Ω con o sin fuente de alimentación.

- La impedancia de entrada debe estar entre 3KΩ y 7KΩ.

Mecánicamente esta interface utiliza un conector tipo "D" de 25 pines, de los cuales 20 están asignados a señales RS232C, 3 están reservados y los 2 restantes no tienen asignación.

La configuración más sencilla de esta interface es aquella en la que se utilizan dos conectores de 25 pines conectados hilo a hilo, utilizada siempre que se conecten los dispositivos específicos para los que fue pensada originalmente esta norma, esto es una conexión entre un DTE y un DCE. Existen sin embargo muchas variantes según el tipo de dispositivos que participan en el intercambio de datos.

En este proyecto en particular los equipos que se va a conectar mediante comunicación serial se encontraran normalmente en el mismo sitio y no en localidades remotos, por lo que se requiere de un tipo de conexión con características especiales.

Comúnmente terminales, impresoras, computadores incorporan un puerto serial RS 232 seteados de modo que actúan como DTEs y por lo tanto esperan responder e interactuar con las señales enviadas por un DCE a otro extremo del cable, al mismo tiempo que suministrará señales tales como RTS (Request to Send) y DTR (Data terminal Ready) que son las salidas normales de un DTE. Por lo tanto considerando que existen 4 categorías de señales como parte de esta norma, datos, control, temporización y referencia; estas señales deben estar presentes en una conexión local de equipos, siendo la alternativa el cruzar estas señales de modo que se pueda "emular" el medio ambiente de comunicaciones como si el modem y las líneas telefónicas estuvieran presentes.

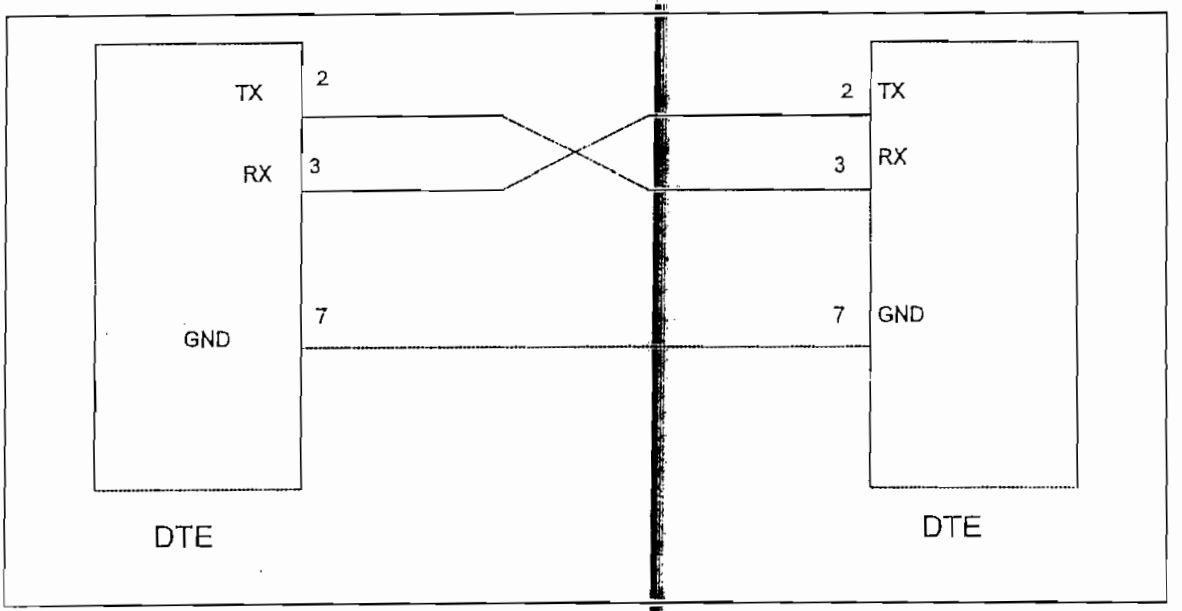


Fig. 1.4. Conexión mínima de la norma RS-232

Debido a que este tipo de conexiones cruzadas se las utiliza cuando el modem no esta presente, se la conoce con el nombre de ***Null-modem***.

La señal que sirve como referencia para las señales binarias, conocida como Tierra de señal, que corresponde al pin 7 debe estar presente para diferenciar los valores de las señales recibidas.

Los Datos en una comunicación bajo la norma RS 232, son transmitidos sobre el Pin 2 del puerto de salida de un DTE y recibidos desde otro sobre el pin 3, por lo tanto para tener una adecuada transmisión y recepción entre los dos equipos deberán estar cruzadas las líneas 2 y 3, teniendo de este modo el más simple de los cables null-modem, a pesar de lo cual tiene el 50% de las señales tenemos tal como se aprecia en la figura 1.4, faltando únicamente las de control y temporización.

La señales que originalmente las provee un DCE, como DSR (Data Set Ready), CTS (Clear to Send) y DCD (Data Carrier Detect) deben ser emuladas con las señales disponibles del DTE esto es con RTS y DTR.

La señal DTR (Pin20) es provista ordinariamente por el DTE para indicar que el terminal esta operativo y es utilizado también para mantener la conexión en un ambiente de dial-up. La indicación de que se ha establecido una línea entre DTE y DCE normalmente la da la señal de DSR (Pin 6). Tan pronto como DSR esta activa podemos asumir que tanto el modem como la línea están disponibles para la transmisión de datos; si por el contrario no existe la señal DSR la transmisión no es posible.

Las funciones de control de flujo requieren de 3 líneas adicionales. La señal RTS (Pin 4) normalmente se origina en el DTE y para permitir la transmisión de datos la señal CTS debe

recibirse en el (Pin 5) del mismo DTE, esto se logra mediante un puente entre los pines 4 y 5 de modo que cuando el DTE envía el pedido de envío, recibe la repuesta de CTS.

Pero adicionalmente se requiere de la confirmación de que existe el camino o medio de transmisión disponible para la recepción de los datos, la cual es esperada con la señal DCD (Pin 8), nuevamente esto se satisface al conectar la señal de RTS a la entrada de DCE de el otro DTE; con lo cual se tiene una configuración Null-modem para transmisión en modo Half-duplex o Full-duplex, como se observa en la figura 1.5.

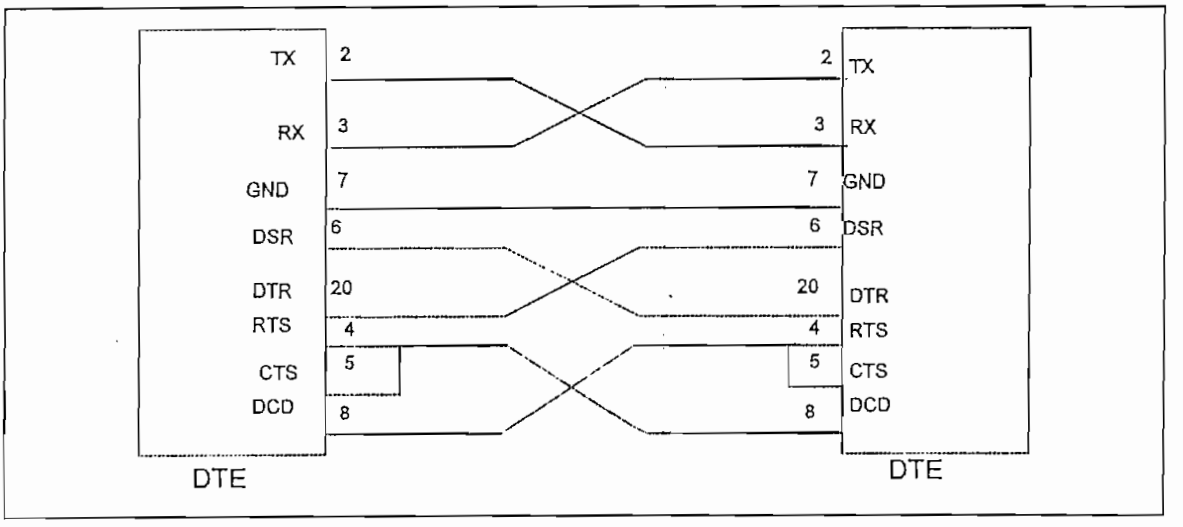


Fig 1.5. Conexión Null Módem incluyendo señales de temporización.

Se incluye un puerto de comunicación serial en el presente proyecto de modo que se puedan realizar transferencias de archivos desde un computador personal para su almacenamiento en la zona de memoria de datos, desde la cual posteriormente podrá ser ejecutada como una rutina más del programa del equipo.

1.4.2.- COMUNICACION PARALELA.

Para la clasificación de las interfaces paralelas se parte de dos características, la primera de las cuales la constituye el número de bits transferidos simultáneamente y la segunda es en base al número de señales de hanshake usadas durante la transferencia de datos.

El número de bits más comunes a ser transferidos es 8, 16 32, 64 bits, siendo la más extendida en su uso aquella que transfiere 8 bits a la vez, debido a que el código de caracteres más común es el ASCII el cual requiere de al menos 7 bits para representar los diferentes caracteres de uso normal, de ahí que la gran mayoría de fabricantes de periféricos y en especial de impresoras utilizan la interface paralela que transfiere 8 bits a la vez.

La segunda forma de clasificación se basa en el número de señales de handshake utilizadas durante la transferencia de datos, existiendo interfaces que requieren de 0, 1, 2, y 3 hilos para tal efecto.

Para nuestro caso en realidad no será necesario usar señales de Hanshake dado que los puertos paralelos serán destinados al manejo de las matrices que conforman el elemento de visualización del prototipo; para la adquisición de datos binarios para el procesamiento por parte del equipo y como salidas a través de las cuales el prototipo suministra señales digitales compatibles con TTL para las diferentes pruebas de sistemas digitales

1.5.- MANEJO DE CARGAS A.C.

Un sistema basado en microprocesador puede realizar labores de control en forma muy eficiente debido a su capacidad de procesar a gran velocidad las señales de entrada que determinan la respuesta del equipo.

Las señales digitales que ingresan a través de un puerto, son procesadas según las necesidades y el tipo del control que deba realizarse y desde el software se determina en que instante y cual de las salidas del sistema debe actuar.

Estas señales de entrada generalmente provienen de sensores, sean estos de nivel, de proximidad inductivos o capacitivos, termocuplas, etc., los cuales proporcionan señales discretas de dos estados, on/off, ó señales continuas que requieren de dispositivos tales como conversores analógico digitales para adaptarse al manejo de un sistema con microprocesador.

En el presente proyecto solo dos cargas de A.C podrán ser manejadas, principalmente con fines demostrativos.

Para que los microcontroladores estén en capacidad de comandar cargas alimentadas con ac, deben actuar a través de otros elementos que se constituyen en las interfaces que actúan sobre dispositivos de mando tales como relés o tiristores.

Cuando se conmutan cargas de ac desde un microcontrolador se debe garantizar que los fenómenos inherentes a la línea de ac en ningún momento lleguen a dañar a los elementos de la lógica digital esto implica garantizar un aislamiento eléctrico entre los componentes digitales y aquella que usa corriente alterna; la mejor forma de lograrlo es utilizando aislamiento óptico

Al combinar en una misma cápsula un LED como fuente emisora de luz, normalmente infrarroja, fabricado en Galio-Arsénico que emite en el rango de 900 a 940 nm, y un detector óptico de estado solido, que puede ser un fototransistor, un fotodarlington, un fototriac, un fotoSCR, etc; se conforma un dispositivo denominado optoaislador u optoacoplador, en el cual la luz originada en el LED atraviesa un medio transparente que puede ser incluso aire, para llegar hasta el fotodetector, este arreglo posibilita el acoplamiento entre circuitos electrónicos totalmente independientes y aislados entre si. Este aislamiento puede estar entre 2000 y 3750 Voltios o aun mayores.

En función de las características de aislamiento y capacitancia entre la entrada y salida del optoacoplador este proporciona protección a los circuitos de control que pueden estar operando a niveles TTL, con respecto a los voltajes elevados que actúan en las etapas de potencia.

La figura 1.6 muestra un optoaislador, que se puede adquirir comercialmente:

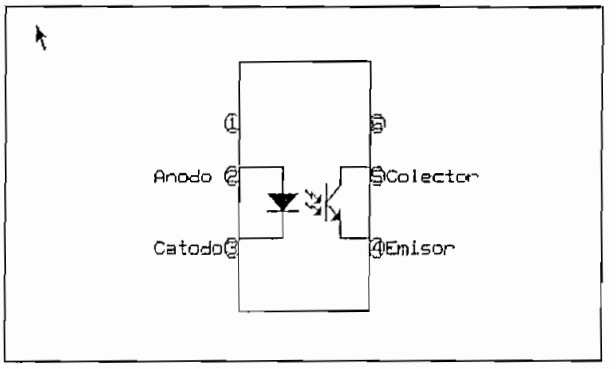


Fig. 1.6. Especificación comercial de un optoacoplador.

Al circular corriente por el LED, este emite fotones los cuales inciden en la base fotosensible del transistor y genera una corriente capaz de llevar al transistor a un estado de saturación .

CAPITULO II

HARDWARE

2.1.- CONFIGURACION BASICA DEL EQUIPO

El equipo desarrollado está en capacidad de satisfacer todos los requerimientos especificados en el primer capítulo por lo tanto precisa de una configuración que pueda ser aprovechada al máximo por el programa en lenguaje de máquina que se encarga de la operación de todo el dispositivo. Para poder cumplir con todas las funciones el prototipo requiere una configuración como la que se puede apreciar en el diagrama de bloques que aparece en la figura 2.1:

2.1.1 .- LA UNIDAD CENTRAL.

Para la creación de la unidad central que se encarga de todas las transformaciones, generación de señales y control de todo el equipo se ha seleccionado un microcontrolador de la familia MCS-51 de INTEL, algunas de cuyas características más relevantes son:

- CPU de 8 bits
- Circuiteria para oscilador incorporada
- 32 líneas de entrada/salida
- Capacidad para direccionar 64 K bytes de memoria de programa
- Capacidad para direccionar 64 K Bytes de memoria de datos
- Incorpora 2 Temporizadores/contadores de 16 bits
- 5 fuentes de interrupción
- Procesador Booleano
- Puerto para comunicación serial Full Duplex

Precisamente varias de estas características y en especial la facilidad para la comunicación serial

han sido determinantes para la selección de dicho dispositivo. La interface serial se implementa mediante el uso del C.I. MAX 232 tal como se explicara más adelante.

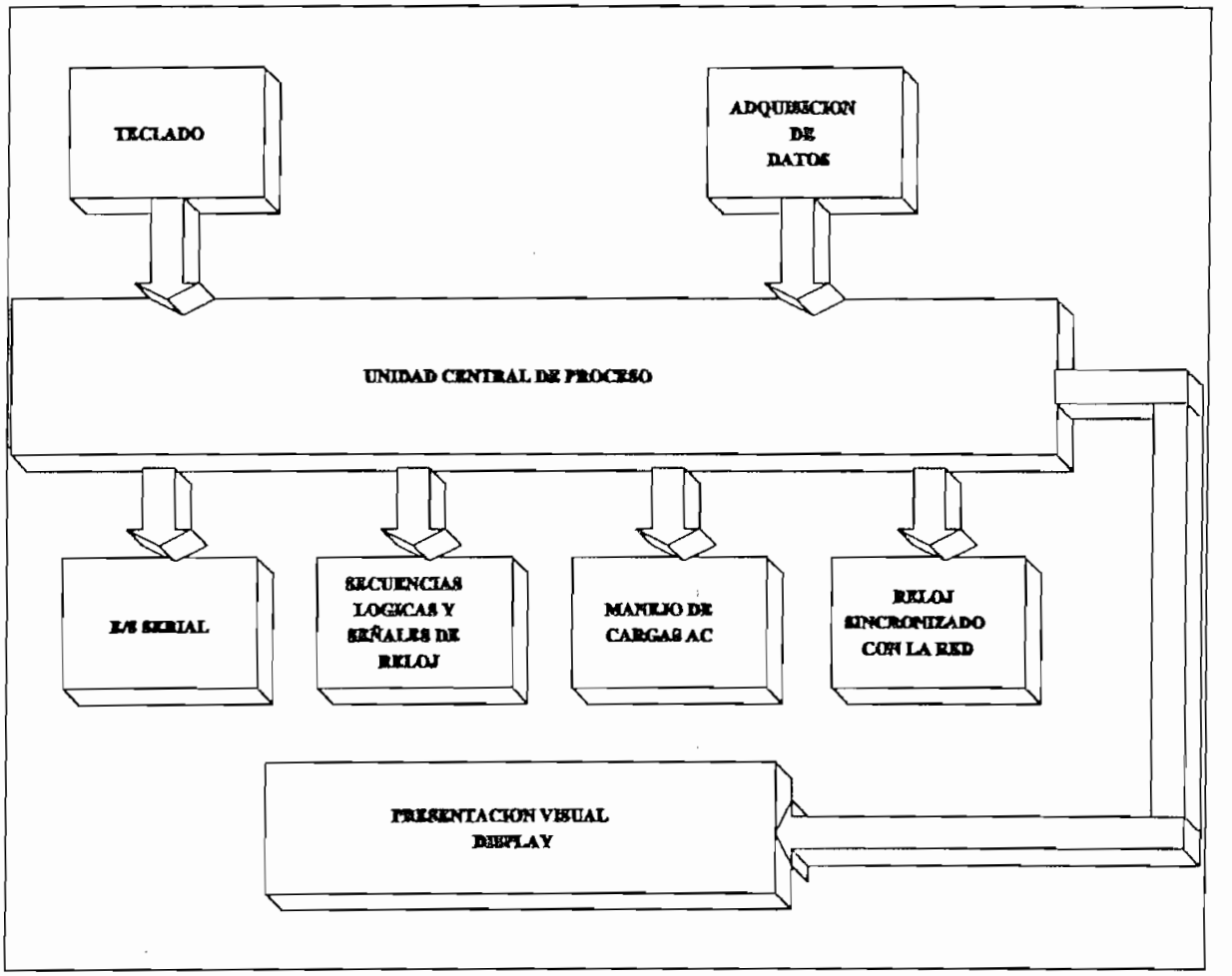


Fig 2.1. **DIAGRAMA DE BLOQUES DE EL EQUIPO.**

2.1.1.1.- CPU, constituida por el microcontrolador 8031, esta seleción implica que toda la memoria de programa sea externa y por lo tanto el pin EA (External Acces) estará permanentemente conectado a tierra obligando a que cualquiera de las instrucciones del programa sean traídas desde la memoria externa de programa.

2.1.1.2.- Oscilador. Se puede utilizar cualquier cristal entre 3.5 y 8 Mhz de acuerdo con las especificaciones del microcontrolador 8031AH, se conectan los extremos del cristal a las patillas 18 y 19 de acuerdo con las especificaciones del manual. En el presente proyecto el cristal tiene

una frecuencia de 7.15909 Mhz por lo tanto para la instrucción que requiere de un ciclo de máquina o lo que es lo mismo doce períodos de oscilación del reloj se tendrá un tiempo de 1.68 Micro segundos de duración.

Los capacitores asociados al circuito del Oscilador son de 30 Picofaradios, se debe considerar que valores pequeños disminuirán el tiempo de inicialización del oscilador, y por el contrario capacitores de mayor valor incrementaran la estabilidad del mismo; por lo tanto el valor de los capacitores no es realmente critico, se debe buscar siempre un compromiso entre las dos características expuestas del circuito de oscilación. La forma exacta de conexión de los diferentes elementos que conforman la etapa de reloj se las aprecia en la Fig. 2.2..

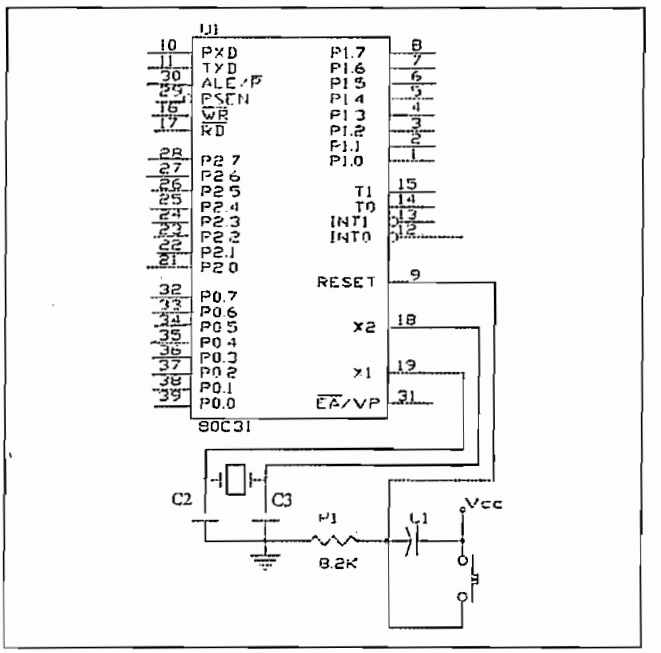


Fig. 2.2. Circuito de Oscilador y Reset

2.1.1.3.- RESET.- De acuerdo con el manual de el microcontrolador (Anexo) un valor alto en el pin 9 del circuito integrado con una duración superior a 2 ciclos de máquina reinicializa al

mismo. Para la construcción de este proyecto se utilizaron los valores de resistencia y capacitor que aparecen en los manuales del microcontrolador 8031, los mismos que son 8.3 K ohmios y 10 Microfaradios.

La forma de interconectar cada uno de los elementos para formar el circuito de reset aparecen en forma explícita en la Fig. 2.2.

2.1.1.4.- Decodificación de direcciones.-En la figura 2.3 se aprecia las diferentes conexiones de lo que se constituiría en el bloque principal del equipo, se puede notar la presencia de un decodificador 74LS138 denominado U5, las salidas de este circuito integrado están conectadas a las entradas de habilitación de los diferentes circuitos integrados, en especial de los PPI, que conforman los puertos paralelos de entrada salida y de los que se detalla más adelante.

Esta forma de conexión permite tratar estos puertos paralelos como direcciones de memoria de modo que las operaciones de recepción de datos digitales o suministro de señales después de procesamiento se las maneja mediante instrucciones de lectura/escritura a memoria. Las entradas del decodificador están conectadas al bus de direcciones inmediatamente después del latch U2 encargado del manejo del byte menos significativo. Estas conexiones se aprecian en la fig. 2.3.

La señal de habilitación G del C.I. 74LS138 esta conectada a la salida más significativa del byte superior del bus de direcciones o lo que es lo mismo al pin P2.7, con lo cual esta etapa de decodificación solo esta activa si se direcciona cualquier posición que sea mayor o igual a 8000h; como consecuencia directa los puertos paralelos suministrados por los C.I. 8255 se los accesa mediante estas direcciones, por ejemplo para el primer PPI, las direcciones son:

Puerto A	8000H	Puerto B	8001H
Puerto C	8002H	Control	8003H

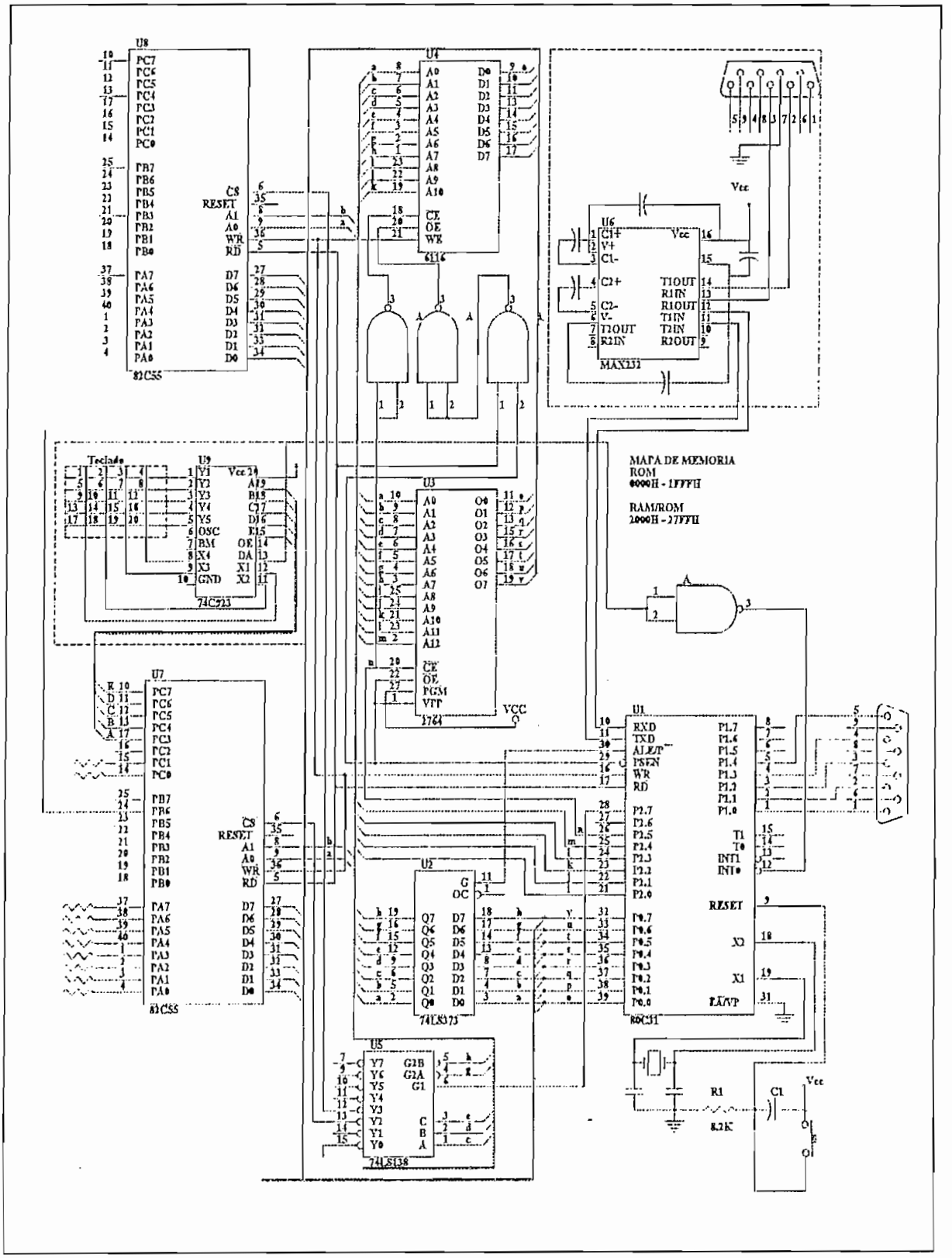


Fig.2.3 Unidad Central incluyendo Memoria, Puerto Serial y Puerto Paralelos

La serie MCS-51 posee espacios separados de direccionamiento para la memoria de programa y memoria de datos. La memoria de programa puede ser de hasta 64 Kbytes, de los cuales los 4 Kbytes inferiores pueden residir en el encapsulado, siendo esta la característica que distingue a los diferentes elementos de la familia como se aprecia en el cuadro :

Nro. de Serie	Memoria de Programa Interna	Memoria de Datos Interna
8031	Ninguna	128 Bytes
8051	4 Kbytes de ROM	128 Bytes
8751	4 Kbytes de EPROM	128 Bytes
8032	Ninguna	256 Bytes
8052	8 Kbytes de ROM	256 Bytes

Tabla 2.1. Versiones de la familia MCS-51

Por su parte la memoria de datos puede consistir de hasta 64 Kbytes de memoria externa, en conjunción con los Registros de Función Especial y los 128 Bytes de Memoria interna ó 256 Bytes en el caso del 8032/8052.

Las 32 líneas de entrada/salida del MCS-51 conforman cuatro puertos bidireccionales, cada uno de los cuales consiste de un buffer de entrada, un driver de salida y de un latch que es uno de los registros especiales nombrados desde P0 hasta P3.

Para el acceso a memoria externa se utilizan los drivers de salida de los puertos P0 y P2, además del buffer de entrada del puerto P0.

Las salidas del puerto P0 multiplexan en tiempo el byte menos significativo de la dirección de

memoria externa y el byte de datos que esta siendo enviado o captado. Cuando se trata de una dirección de hasta 16 bits el byte más significativo de la dirección aparece el las salidas del puerto P2.

Cada uno de los pines del puerto P3 además de constituir líneas de entrada/salida cumplen funciones alternas especiales, las cuales son activadas cuando el bit correspondiente en el registro de funciones especiales contiene un 1L.

Estas funciones alternativas son:

P3.0	RXD Entrada del puerto serial
P3.1	TXD Salida del puerto serial
P3.2	INT0 Interrupción externa
P3.3	INT1 Interrupción externa
P3.4	T0 Temporizador/contador externo 0
P3.5	T1 Temporizador/contador externo 1
P3.6	WR Señal de escritura para memoria externa de datos
P3.7	RD Señal de lectura para memoria externa de datos.

2.1.2.- DIRECCIONAMIENTO Y ACCESO A MEMORIA.

El microcontrolador seleccionado es el 8031, por lo tanto la memoria de programa estará constituida en su totalidad por dispositivos externos. El dispositivo físico que se utiliza en el presente proyecto para el almacenamiento del programa es una EPROM 2764, con lo cual disponemos de 8 Kbytes de memoria, donde residirán las rutinas que determinan el funcionamiento del equipo.

Se añade una RAM 6116, la misma que puede actuar como memoria de programa o como memoria de datos. La selección del modo de trabajo se la realiza desde software, es decir que

esta zona de memoria solo puede ser accesada bajo selección del usuario.

Al contar con esta configuración, los programas en Assembler pueden ser transferidos desde un computador personal a través de comunicación serial y se los almacena en la RAM mientras esta actúa como memoria de datos; luego bajo control de una rutina y con las señales de habilitación adecuadas pasa a trabajar como memoria de programa.

Esto agilitará la depuración de programas en lenguaje de máquina al poder realizar cambios sin necesidad reprogramar varias veces la EPROM, solo una vez que el programa cumple a cabalidad con sus requerimientos se lo almacenara en el dispositivo definitivo.

Por tener el byte menos significativo del bus de direcciones multiplexado en tiempo con las líneas de datos resulta indispensable el uso de un medio para almacenamiento temporal (latch) externo de las direcciones, durante el acceso a memoria externa.

El dispositivo más adecuado para tal propósito es el C.I. 74LS373, que es un registro de 8 bits con salidas de tres estados.

Este circuito TTL esta constituido por 8 flip flops tipo D y es transparente, esto significa que mientras el pin de habilitación (C) esta en alto las salidas Q siguen a los datos en los pines de entrada D.

Cuando la habilitación cambia a bajo las salidas retienen los datos que fueron seteados; la siguiente tabla aclara dicho funcionamiento:

De donde se desprende que en nuestro circuito la entrada de habilitación deberá estar conectada permanentemente a tierra, con lo cual el control de el almacenamiento temporal (latch) lo determina únicamente la señal de reloj.

Habilitación C	Reloj	Entrada D	Salida Q
L	↑	H	H
L	↑	L	L
L	L	X	Qo
L	X	X	Z

Tabla 2.2. Señales del Latch 74LS373

Esta señal de temporización esta prevista en los microcontroladores de la familia MC-51 a través del pin 30 y se llama **ALE (Address Latch Enable)**, la misma que se activa dos veces en cada ciclo de máquina aún cuando la instrucción no involucre una captura desde memoria externa. Unicamente cuando se realiza un acceso a memoria externa de datos mediante la instrucción MOVX, que utiliza 2 ciclos de máquina, se pierde la primera señal de ALE del segundo ciclo.

2.1.3.- DISTRIBUCION DE MEMORIA.-

La memoria del equipo esta constituida por una zona de ROM propiamente dicha y una zona en la cual el dispositivo físico realmente es una RAM, la cual mediante un proceso adecuado de selección actúa como memoria de programa. El cuadro siguiente resume en forma más clara esta distribución de memoria:

Dirección	Tipo de memoria
0000H - 1FFFH	ROM (2764)
2000H - 27FFH	RAM/ROM (6116)

Tabla 2-3. Distribución de memoria.

La señal de habilitación para la memoria de programa externa ha sido prevista en la familia MCS-51 como **PSEN** (Program Store Enable) la cual se activa dos veces durante cada ciclo de máquina excepto al utilizar la instrucción MOVX.

Para la activación del C.I. 2764 (EPROM) se usa una combinación de la dirección A_{13} denominada como **n** de acuerdo a lo que se puede apreciar en la fig 2.3 (o la salida del puerto 2 especificada como P2.5) y de la señal PSEN. Esto es mientras $A_{13} = 0L$ o lo que es lo mismo cualquier dirección inferior a 2000H la entrada CE de la EPROM permanece en bajo y por lo mismo en condición de habilitación. La señal PSEN se conecta a la entrada OE de la memoria de programa y valida las salidas cada vez que se ejecuta un acceso a esa zona de memoria.

Para el caso de la memoria de datos se habilita las salidas por una señal que es una combinación de las señales PSEN y RD, las dos pasan a través de una compuerta NAND 74LS00 y por una segunda compuerta NAND que tiene interconectadas entre si sus entradas y conectadas a la señal de salida de la compuerta anterior con lo cual actúa como un inversor.

Esto implica que cuando cualquiera de las señales PSEN o RD pasan a 0L la salida conectada a las entradas de OE de la memoria de datos tiene un valor de 0L y por tanto están habilitadas sus salidas.

La memoria de datos además tiene su dirección inicial en el valor 2000H por lo tanto únicamente cuando se cumple la condición de la señal $A_{13} = 1L$ la entrada CE de la memoria de datos que previamente pasa por una compuerta encargada de invertir su valor, tendrá el 0L que necesita para habilitar a ese circuito integrado.

Para ejecutar el programa que se descarga en la RAM como si fuese memoria de programa bastara con ejecutar un salto a la dirección 2000H, con lo cual la señal PSEN se encargara de activar esa zona de memoria durante la fase de captura de los códigos en base a las conexiones que se observan en la figura 2.3.

2.1.4.- EXPANSION DE ENTRADAS/SALIDAS.

Si bien los microcontroladores de la familia MCS-51 incorporan cuatro puertos de entrada salida, en algunas aplicaciones estos pueden resultar insuficientes, en el presente proyecto se hace presente tal situación y por lo tanto resulta indispensable realizar una expansión del número de puertos lo cual se logra mediante el uso de C.I. 82C55A de INTEL cuyas características más relevantes para la presente aplicación se resumen a continuación, información completa respecto a los mismos se puede consultar en los anexos correspondientes.

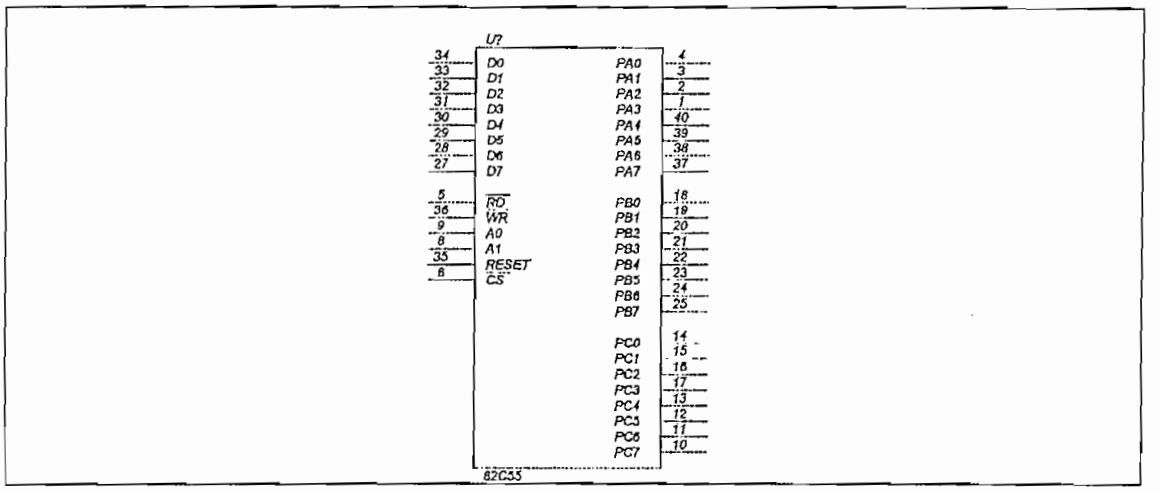


Fig. 2.4. Distribución de pines de el C.I. 8255

El C.I. 82C55A, que aparece en la fig. 2.4. es un dispositivo programable de entrada/ salida de propósito general, especificado por sus fabricantes como una Interface Programable de Periféricos (ProgrammablePeripheral Interface PPI), diseñado para ser usado con microprocesadores INTEL, especialmente para el microprocesador 8085, sin embargo debido a su versatilidad se lo puede emplear con una gran variedad de microprocesadores, como el dispositivo más adecuado para el manejo de un número significativo de líneas de entrada/salida.

Según se aprecia en la figura 2.4, en su presentación de 40 pines, tres bloques de 8 pines

conforman un total de 24 líneas de entrada/ salida ó tres puertos paralelos programables de 8 bits, distribuidos de modo que al puerto A le corresponden PA0 - PA7, al puerto B las líneas PB0 - PB7 y PC0 - PC7 para el puerto C, este ultimo sin embargo puede dividirse en dos sub grupos PC0-PC3 y PC4-PC7 que pueden actuar como puertos individuales de 4 bits o asociarse, de modo que los 4 bits más significativos del puerto C y las líneas del puerto A por una parte y los menos significativos de C conjuntamente con el puerto B llegan a constituirse en grupos de 12 líneas de entrada /salida.

Además de los pines de entrada /salida existen líneas de control y las 8 líneas de datos bidireccionales D0- D7 que son el medio de integración del C.I. 8255 al bus de datos del sistema basado en microprocesador, la distribución interna de los componentes de este circuito integrado aparece en la fig. 2.5.

El C.I. 8255 tiene tres modos de operación, uno de estos modos debe ser seleccionado como primer paso inmediatamente después de que se aplica la alimentación eléctrica al circuito integrado puesto que todas las líneas del bus de datos y los puertos se encuentran en estado de alta impedancia o con valores flotantes, para esta inicialización y para el resto de operación se utiliza la palabra de control que se carga en un registro de 8 bits denominado Registro de control.

En el primer modo, llamado modo 0, cada uno de los puertos, esto es A, B ó C, actúan como entradas o salidas básicas; cuando actúan como salida se dispone de las funcionalidad de Latch en cada uno, es decir el dato actual se mantiene sin alteración mientras no se reciba el siguiente dato, para el caso de entradas por el contrario la operación de lectura debe ser simultanea con la operación de lectura del puerto. En este modo son posibles 16 configuraciones de entrada/salida dependiendo de la combinación de los bits de la palabra de control.

En el segundo modo o modo 1, cada grupo puede ser programado para que existan 8 líneas de entrada/ salida de los puertos A o B y de las 4 linceas restantes que corresponde a la subdivisión del puerto C, 3 son usadas para señales de control de interrupciones y señales de handhake,

quedando aun disponibles dos líneas del puerto C para que actúen como entrada/salida normales. A esta forma de operación se la conoce también como entrada/salida con confirmación o validación, y tanto las entradas como las salidas disponen de la funcionalidad de latch.

El tercer modo de operación o modo 2 las líneas del puerto A actúan como un bus bidireccional para la entrada o salida de datos y 5 líneas del puerto C (PC3 a PC7) proporcionan las señales de handshake con los dispositivos periféricos al el conectados. Las tres líneas sobrantes del puerto C pueden constituirse en líneas auxiliares del puerto B o como entradas/salidas normales, el puerto B puede estar configurado como en el modo 0 o el modo 1. Para esta modalidad de operación la opción de latch se aplica tanto a las entradas como a las salidas.

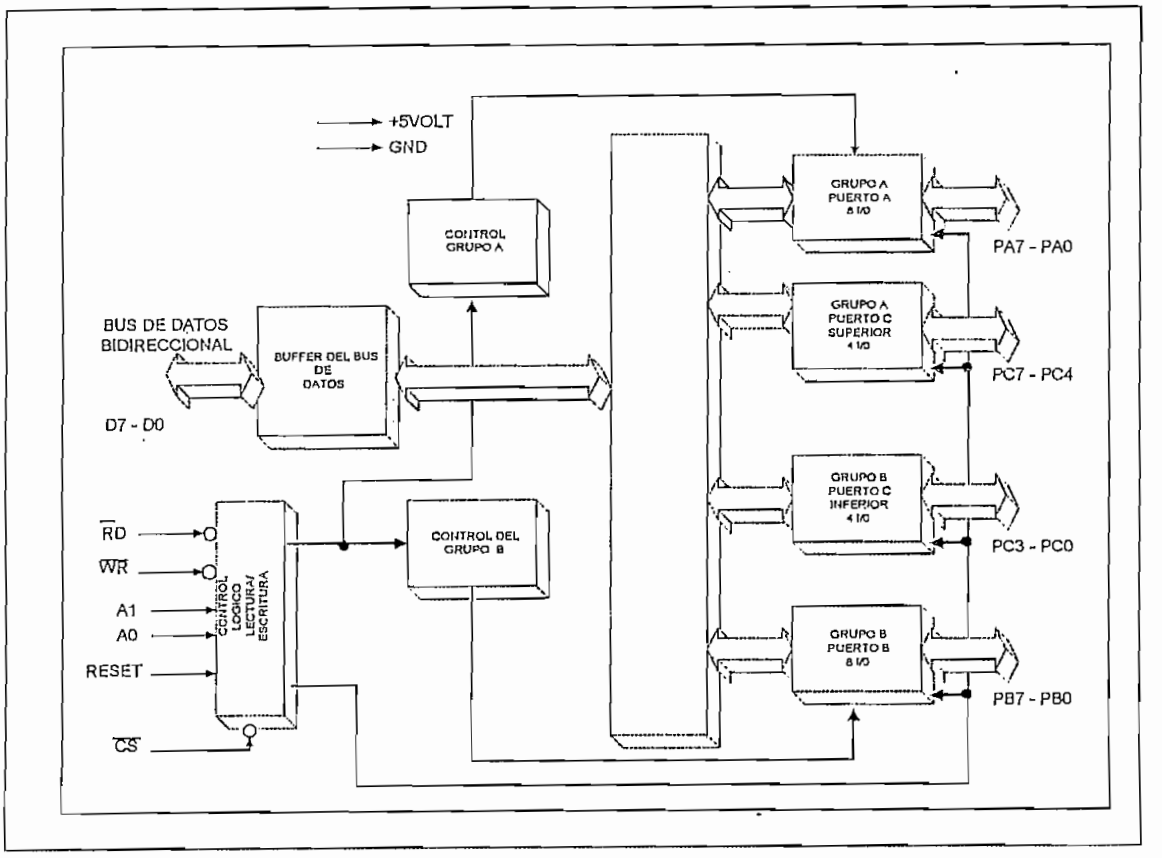


Fig. 2.5. Diagrama de bloques de un PPI 8255.

La configuración funcional de los dispositivo 82C55A se hace a través del software del sistema de modo que no se requiere de lógica adicional para la conexión de periféricos a través de este dispositivo.

El circuito integrado dispone de un ***BUFFER PARA EL BUS DE DATOS***, este buffer tri-state es utilizado para interconectarse con el bus de datos del sistema. Los datos son transmitidos o recibidos por el buffer siempre que se ejecuta una instrucción de entrada o salida en la CPU. Además se transfieren a través del bus de datos las palabras de control y la información del estado del dispositivo.

Las funciones de ***Lectura/Escritura y la lógica de Control*** se encargan de manejar las transferencias tanto internas como externas de las palabras de Control y estado, así como también de la transferencia de datos.

Para el control del dispositivo se utilizan las siguientes señales:

CS (Chip Select) una señal de 0L en este pin habilita las comunicaciones entre el 82C55A y la unidad central de proceso.

RD (Read) se activa con 0L, y permite que la CPU lea información desde el 82C55A, este puede enviar a través del bus de datos tanto información de su estado o datos propiamente dichos.

WR (Write) una señal de 0L en este pin permite a la CPU para que los datos o palabras de control transferidas desde la CPU sean aceptadas.

$\mathbf{A_0}$ y $\mathbf{A_1}$ son dos entradas que en conjunción con las señales de lectura escritura se utilizan para seleccionar uno de los tres puertos o uno de los registros de control. Normalmente están conectados a las líneas menos significativas del bus de direcciones de la CPU.

La selección se la realiza de la siguiente manera:

A1	A0	RD	WR	CS	*OPERACION DE ENTRADA READ*
0	0	0	1	0	PUERTO A ==> BUS DE DATOS
0	1	0	1	0	PUERTO B ==> BUS DE DATOS
1	0	0	1	0	PUERTO C ==> BUS DE DATOS
					OPERACION DE SALIDA WRITE
0	0	1	0	0	BUS DE DATOS ==> PUERTO A
0	1	1	0	0	BUS DE DATOS ==> PUERTO B
1	0	1	0	0	BUS DE DATOS ==> PUERTO C
1	1	1	0	0	BUS DE DATOS ==> CONTROL
X	X	X	X	1	BUS DE DATOS ==> 3-STATE
1	1	0	1	0	CONDICION ILEGAL
X	X	1	1	0	BUS DE DATOS ==> 3-STATE

Tabla 2.4. Modalidades de operación de el C.I. 8255

RESET.- Una señal alta (1L) en esta entrada elimina el contenido de los registros de control y todos los puertos pasan a su modo de entrada.

En cuanto a la operación del dispositivo, como ya se menciono existen tres modos básicos . La selección del modo en el cual el puerto A y B van a trabajar se realiza independientemente mientras que el puerto C puede ser dividido en dos porciones según los requerimientos que producen las definiciones para los puertos A y B. Todos los registros de salida incluyendo los flip-flops de estado serán reseteados cuando se cambia el modo de operación.

Otra característica que resulta sumamente útil, especialmente para aplicaciones de control es, la posibilidad de seteo y reseteo a nivel de bit, esto es cualquiera de los 8 bits del puerto C puede ser seteado o reseteado mediante una sola instrucción de salida.

En nuestro caso particular el modo de trabajo que más se ajusta a nuestras necesidades es el modo 0, con esta configuración funcional se dispone de operaciones de entrada/salida en cada uno de los tres puertos sin la necesidad de señales de Handshake, los datos simplemente pueden ser escritos o leídos desde el puerto especificado por la instrucción.

Las funciones que se definen al seleccionar el modo 0 de operación son:

- Dos puertos de 8 bits y dos puertos de 4 bits.
- Cualquier puerto puede actuar como entrada o salida.
- Las salidas disponen de almacenamiento temporal (latched)
- Las entradas no tienen almacenamiento temporal.
- Se dispone de 16 configuraciones de entrada/salida en este modo.

Para que cumplir con los requerimientos de los diferentes sistemas se puede seleccionar una de las 16 configuraciones disponibles, para lo cual lo se debe realizar una adecuada conformación de 4 de los bits de la palabra de control como se aprecia tabla 2.5:

A		B		GRUPO A			GRUPO B	
D4	D3	D1	D0	PORTA	PORT C SUPERIOR	#	PORT B	PORT C INFERIOR
0	0	0	0	SALIDA	SALIDA	0	SALIDA	SALIDA
0	0	0	1	SALIDA	SALIDA	1	SALIDA	ENTRADA
0	0	1	0	SALIDA	SALIDA	2	ENTRADA	SALIDA
0	0	1	1	SALIDA	SALIDA	3	ENTRADA	ENTRADA
0	1	0	0	SALIDA	ENTRADA	4	SALIDA	SALIDA
0	1	0	1	SALIDA	ENTRADA	5	SALIDA	ENTRADA
0	1	1	0	SALIDA	ENTRADA	6	ENTRADA	SALIDA
0	1	1	1	SALIDA	ENTRADA	7	ENTRADA	ENTRADA
1	0	0	0	ENTRADA	SALIDA	8	SALIDA	SALIDA
1	0	0	1	ENTRADA	SALIDA	9	SALIDA	ENTRADA
1	0	1	0	ENTRADA	SALIDA	10	ENTRADA	SALIDA
1	0	1	1	ENTRADA	SALIDA	11	ENTRADA	ENTRADA
1	1	0	0	ENTRADA	ENTRADA	12	SALIDA	SALIDA
1	1	0	1	ENTRADA	ENTRADA	13	SALIDA	ENTRADA
1	1	1	0	ENTRADA	ENTRADA	14	ENTRADA	SALIDA
1	1	1	1	ENTRADA	ENTRADA	15	ENTRADA	ENTRADA

Tabla 2.5. Configuraciones posibles para el C.I. 8255

Durante la fase de diseño del prototipo se determino la necesidad de contar con al menos tres

circuitos integrados 8255, de modo que el número de líneas de entrada salida disponibles se eleva a 72 sin contar con los puertos propios del microprocesador.

Para direccionar cada uno de los circuitos integrados 8255 se utilizan las señales menos significativas del bus de direcciones del sistema basado en microprocesador, de modo que a cada a las entradas de control A0 y A1 de cada uno de estos dispositivos se encuentran conectadas las líneas P0.0 y P0.1(a y b según la denominación usada en el esquema principal) del bus de direcciones, y la selección del circuito integrado específico se realiza por medio de la entrada CS de cada uno, esta señal proviene de las salidas de el decodificador 74LS 138 (U5) como resultado de la combinación adecuada de las señales P0.2, P0.3y P0.4 (c, d y e) del bus de direcciones del prototipo.

La señal de activación del circuito integrado decodificador esta conectada a la señal más significativa del bus de direcciones del microtontrolador 8031 o lo que es lo mismo P2.7, esto implica que únicamente cuando se direcciona un valor igual o superior a 8000h se estarán seleccionando los puertos de entrada salida y los registros de control de los circuitos integrados 8255.

Por ejemplo si se necesita acceder al registro de control del circuito integrado 8255 al que se le ha denominado U7 (fig. 2.3) y que controla mediante los puertos A y B los transistores de barrido de columnas de las matrices, las señales que se necesitan son:

P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
8 H				**0 H**				**0 H**				**B H**			

Tabla 2.6. Ejemplo de palabra de control para el PPI 8255

Las señales de RD y WR se conectan directamente a las señales correspondientes del microcontrolador 8031 al igual que la entrada de reset que se conecta al reset general del prototipo

2.2.- CREACION DE UN DISPLAY DE DATOS USANDO MATRICES DE LEDS DE 7X5.

Utilizando tecnología de Semiconductores se han desarrollado fuentes de Energía luminosa llamados LEDs (Ligth Emithing Diode o Diodos Emisores de Luz); al recombinarse un electrón y un hueco dentro de este tipo de material se produce emisión de luz.

El color de la luz producida por estos componentes de estado solido, es decir la longitud de onda dentro del espectro visible depende del material utilizado, el cual determina la cantidad de energía que se libera durante la recombinación al pasar un electrón de la banda de conducción a la banda de valencia.

La frecuencia de la luz emitida esta dada por la ecuación:

$$h\nu = E_g$$

donde Eg es la separación energética del Semiconductor
h es la constante de Plank (6.626×10^{-34} J/S)

Para que la emisión de luz se produzca la banda de conducción debe estar poblada por muchos electrones y la banda de valencia con huecos, esto se logra mediante un circuito de polarización que determina la corriente que circula, esto produce un efecto tal que: "Conforme el valor de la corriente aumenta a través de la unión PN, más electrones y huecos serán inyectados en la banda prohibida. Su movimiento ocasiona un efecto secundario que

aumenta el número de portadores disponibles para la recombinación y por ello se eleva la eficiencia del proceso de emisión" (1)

Por eso la salida de potencia del diodo es aproximadamente lineal con respecto a la corriente y disminuye al incrementarse la temperatura ambiente.

Para que el LED opere en un punto especificado de corriente deberá conectarse usando una resistencia limitadora en serie con el mismo tal como se aprecia en la figura 2.6:
Como ya se menciono en el primer capitulo para el presente proyecto se utilizaran matrices de LEDs de 7x5 con un tamaño 40X22mm.

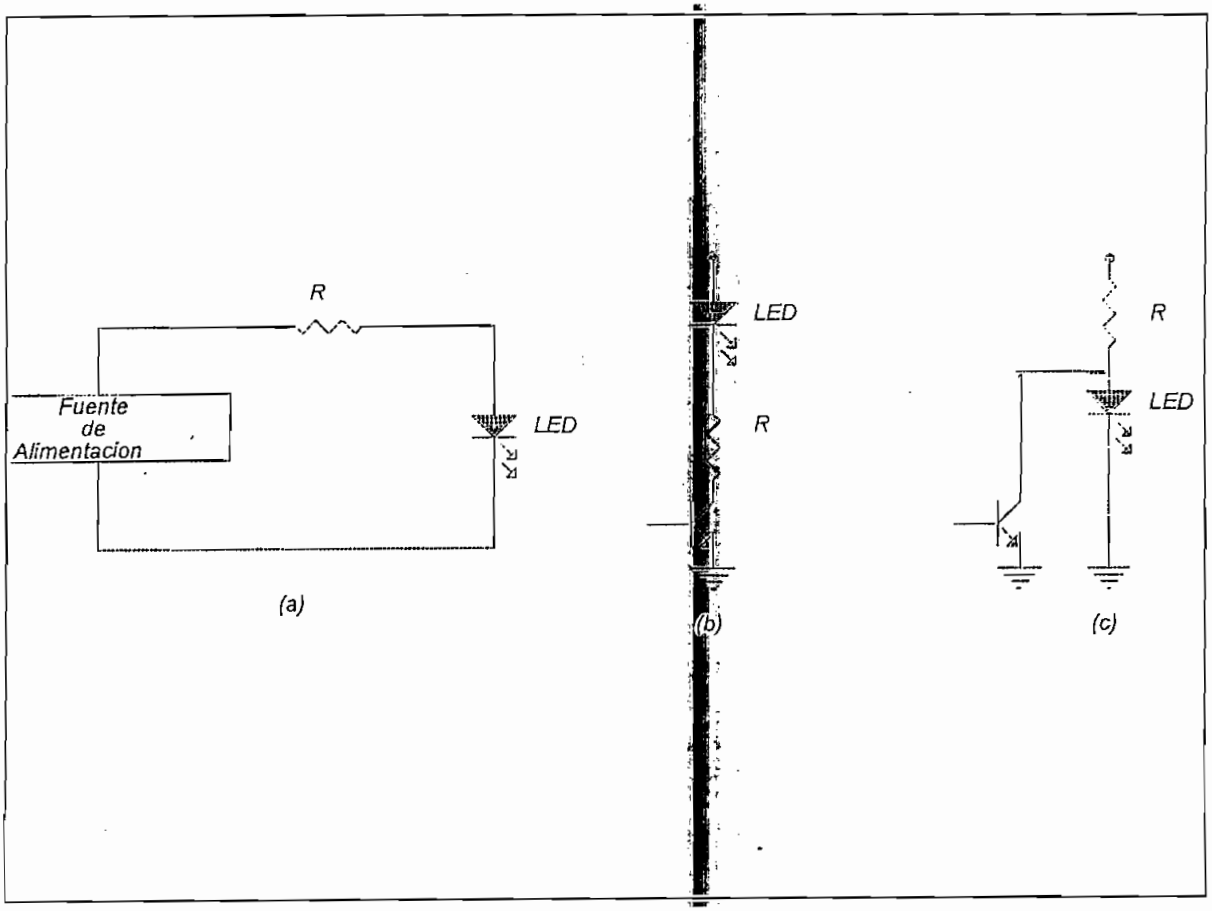


Fig. 2.6 Circuitos de ejemplo de polarización de leds.

Cada caracter o dígito se forma por la combinación adecuada de Leds encendidos y apagados,

y es el microprocesador el encargado de proporcionar esta información, para lo cual dentro del software que constituye el sistema operativo del microprocesador existirán las codificaciones correspondientes a cada columna del arreglo, esta será proporcionada a la matriz a través de uno de los puertos implementados con los C.I. 82C55 y consistirán de palabras de 7 bits correspondientes a cada uno de los leds de la columna, mediante otro puerto y con palabras de 5 bits se determinara cual de las 5 columnas será habilitada.

Como puede verse se esta empleando una técnica de multiplexaje en parte por software y utilizando el hardware mínimo de modo que se han reducido tanto el numero de componentes como de interconexiones que es una de las ventajas de un sistema basado en microprocesadores.

En la figura 2.7 se aprecia una conexión de una de las 40 columnas que conforman el display, en la misma se puede apreciar que la condición de encendido o apagado de cada uno de los 7 leds depende de las líneas conectadas inmediatamente después de la resistencia limitadora, las cuales vienen del un puerto de un C.I. 82C55 y tienen dos condiciones posibles:

Si la línea tiene un nivel alto 1L el único camino posible para la corriente es a través del LED y a ese momento si la línea que esta conectada a la base del transistor tiene un valor alto (existe un pulso de activación) tal que el transistor pase a una condición de saturación el circuito se cierra y el LED se enciende.

Para generara los caracteres el microprocesador a través de los periféricos correspondientes deberá sincronizar un conjunto de pulsos secuenciales para la habilitación de cada una de las columnas que conforman la matriz con los datos respectivos suministrados a cada uno de los 7 LEDs en dicha columna.

La velocidad de renovación o frecuencia de multiplexaje puede expresarse por $f=1/T$ en donde T es el tiempo que tarda un ciclo completo de visualización.

La sincronización se refiere a la simultaneidad que debe existir entre la presencia de los 7 bits de datos y el pulso de activación de la columna correspondiente, al mismo tiempo sin embargo las 10 columnas reciben los mismos datos, pero únicamente una de estas 10 columnas tendrá el pulso de activación y solo sus LEDs se prenderán.

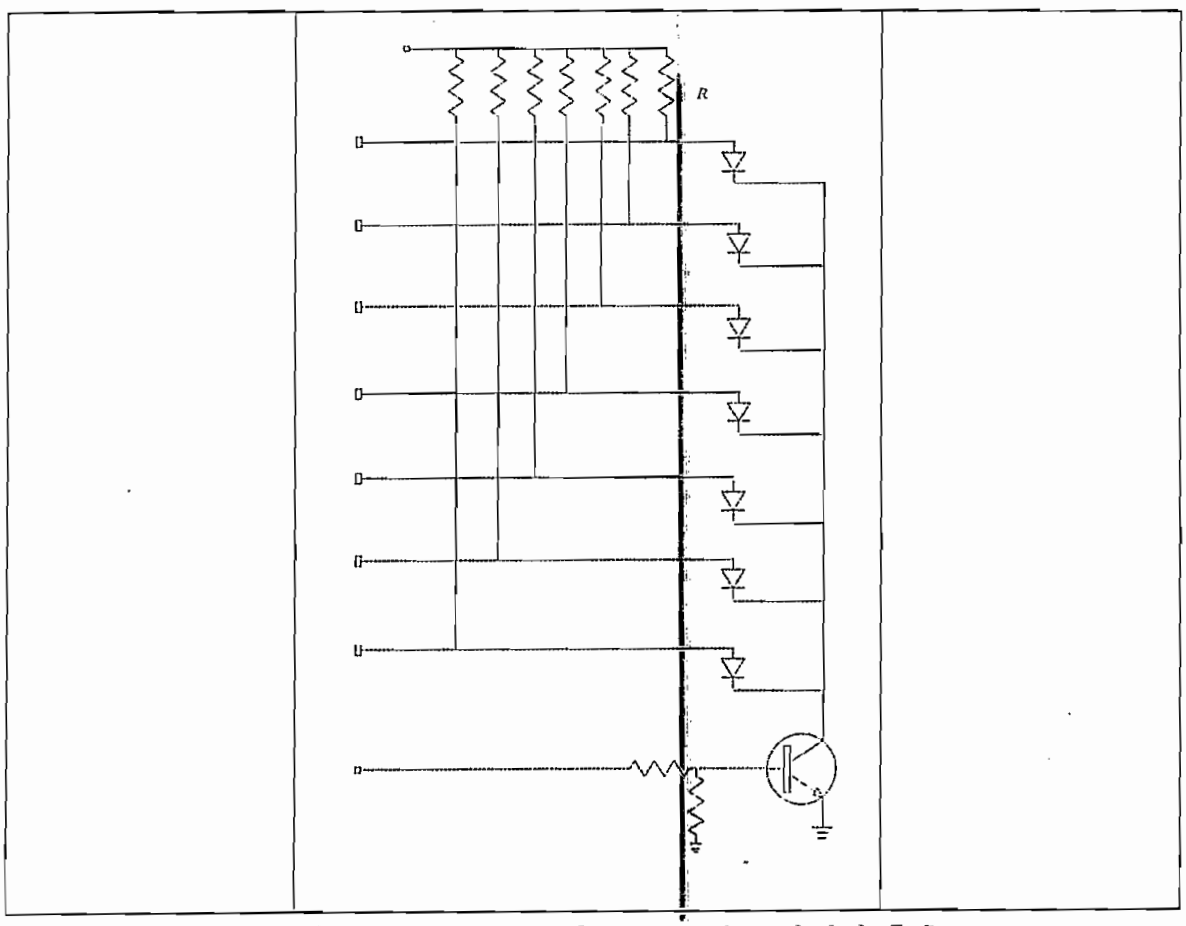


Fig. 2.7. Representación de una columna de las matrices de leds 7x5.

Por lo tanto deberá existir una frecuencia tal de activación de cada uno de estos transistores que el observador vea el caracter formado y no note la discontinuidad en la activación de cada una de las columnas.

A fin de reducir el número de componentes se ha procedido a realizar un arreglo con las matrices de modo que se unen dos matrices para ser manejadas desde un mismo puerto que suministra los datos para el encendido de los LEDs, esto significa que existirán 7 resistencias limitadoras de corriente en lugar de 14, pero se necesitaran de 10 transistores de activación en lugar de 5, se ha llegado a este arreglo buscando un compromiso entre facilidad de lectura, simplificación de hardware y software.

Puesto que los LEDs correspondientes a cada una de las 40 columnas permanecen apagadas la mayor parte del tiempo, de utilizarse la corriente de polarización directa que se usa en un circuito de polarización continuo, su brillo disminuye notablemente por lo tanto para compensar tal situación es necesario incrementar la corriente que circula por el LED, este incremento se basa el **factor de servicio**, que se define como el cociente entre el tiempo que el LED permanece encendido y el que dura apagado, expresado como:

Factor de Servicio (en porcentaje)=1/N x 100

En nuestro caso el factor de servicio resulta=1/40 =2.5%

El pulso de corriente I_P necesario para equiparar a una corriente de polarización de directa I_F de 5 mA será:

I_P=(I_F/factor de servicio)=5mA/0.025=200mA

Para tener esta corriente el valor de la resistencia limitadora será:

$$R=(V_{CC} - V_F)/I_F$$

$$R=(5-1.4)/200mA = 18 \text{ Ohmios}$$

Otro factor que influye en el brillo de los LEDs que conforman la matriz es el ancho del pulso de activación o la duración del mismo, esto en nuestro caso esta controlado por software produciendo retardos en el barrido de cada columna y esto altera el valor de la resistencia R.

2.3.- EL TECLADO PARA SELECCION DE FUNCIONES.

El medio de comunicación principal entre el usuario y el equipo esta constituido por un teclado, esto es el periférico de entrada esta formado por un conjunto de pulsadores de modo que a cada caracter, función o instrucción le corresponde un pulsador determinado.

Cuando hablamos de teclados, tanto la distribución como el número de teclas que lo conforman es muy variable, siendo la aplicación la que determina un desarrollo específico, en nuestro caso el teclado está conformado por 20 teclas asociadas cada una a un pulsador mecánico que forma parte de un arreglo de tipo matricial de tal modo que el número de teclas es igual al número de intersecciones, dando como resultado un total de 9 conductores de salida, distribuidos en cuatro filas y cinco columnas, de modo que al presionar una tecla se establece un contacto con un circuito impreso y se produce un cierre de contactos entre una fila y una columna permitiendo determinar cual de las teclas ha sido presionada.

Estas señales podrían alimentarse directamente a nueve líneas de entrada del sistema basado en microprocesador, el cual mediante software se encargaría de eliminar el rebote y determinar la frecuencia de barrido para el arreglo, pero resulta más práctico utilizar un C.I. como 74C923 que es un codificador de teclado que proporciona una codificación de 4 bits de la tecla presionada al microcontrolador encargándose además de proporcionar la frecuencia de barrido

y la eliminación del rebote. Esta forma de conexión aparece en la fig. 2.8. en la parte correspondiente a la interconexión teclado circuito integrado.

Entre las características más relevantes del Circuito Integrado 74C923 tenemos:

Dentro de cada C.I. 74C923 se ha implementado toda la lógica necesaria para conectar un arreglo matricial de hasta 20 teclas a un sistema digital, mediante dos capacitores externos se realiza el control tanto del período de atenuación de rebote así como la frecuencia de exploración.

Cualquier arreglo de tipo matricial que contenga hasta 4 filas y 5 columnas puede ser conectado al C.I.74C923. Mientras ninguna de las teclas es presionada las entradas correspondientes a las filas se mantiene en alto mediante pull-ups internos y las salidas de las columnas van pasando secuencialmente a un estado bajo. Estas salidas son de tipo de drenaje abierto, y permanecen en un nivel bajo el 25% del tiempo, el resto están en un estado de apagado.

La tasa de exploración de las columnas esta controlada por el capacitor conectado a la entrada del oscilador, el cual consiste de un oscilador Schmitt trigger, un contador de 2 bits y un decodificador de 2 a 4 bits.

Cuando una de las teclas es presionada, 0 por ejemplo, nada ocurrirá mientras la entrada X1 esta en "apagado", puesto que Y1 se mantiene en alto, únicamente cuando la columna X1 es explorada pasa a un nivel bajo haciendo que también Y1 pase a un nivel bajo, esto deshabilita el contador y X1 permanece en bajo, al mismo tiempo que Y1 pasa a nivel bajo.

El circuito de eliminación de rebote inicia su temporización bloqueando las entradas Y, el

código de la tecla se obtiene como una combinación de las salidas del contador y una codificación de las entradas Y.

Una vez que el tiempo necesario para la eliminación del rebote ha transcurrido, la codificación es almacenada y una salida correspondiente a Dato disponible DAV pasa a un nivel alto.

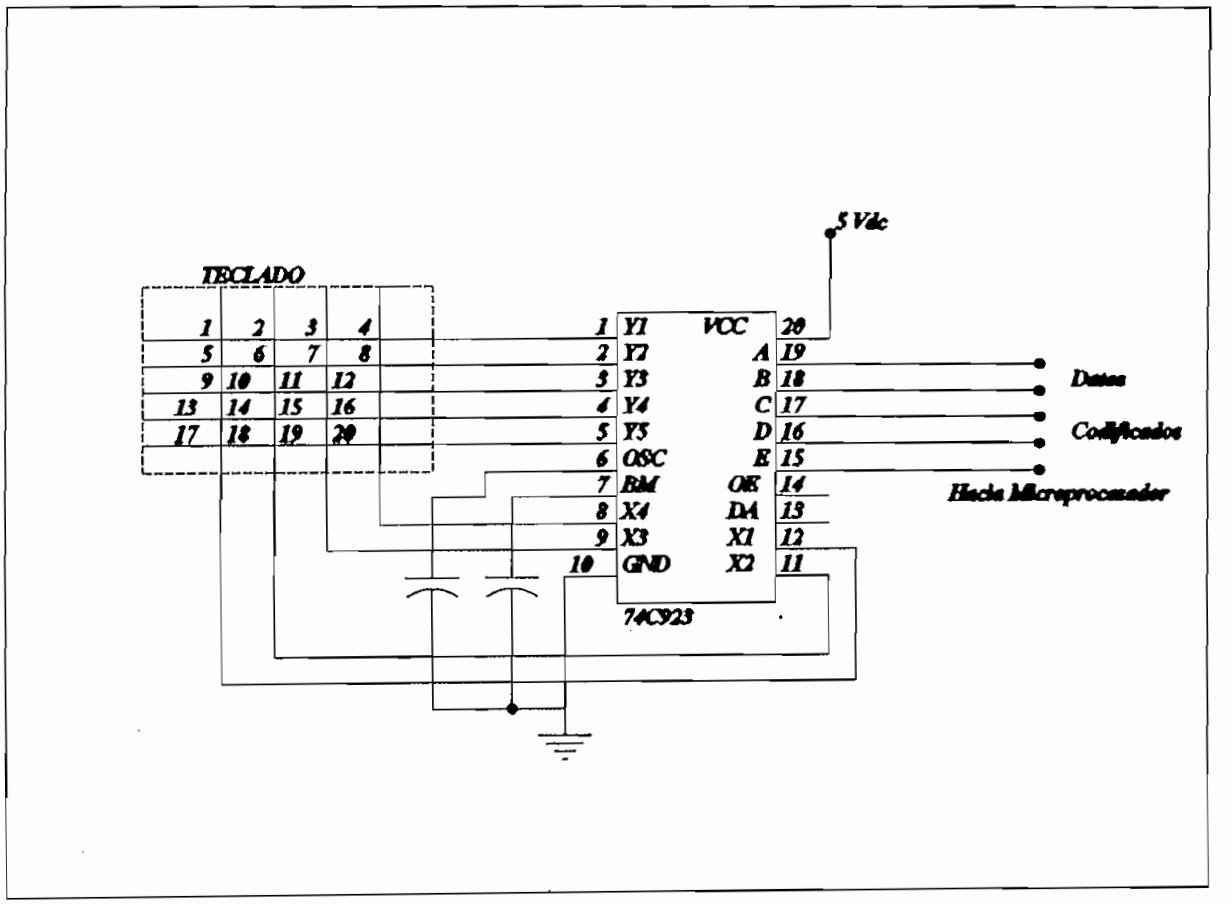


Fig. 2.8 **Teclado y su circuito de codificación**

Precisamente esta señal la podemos aprovechar para notificar al microcontrolador que una de las teclas ha sido presionada y que el usuario requiere atención, para lo cual se conecta esta señal a la entrada correspondiente a la interrupción externa 0 del C.I. 8031, por lo tanto cada vez que una tecla es presionada la interrupción externa cero se activa y la rutina de atención de a esta

interrupción toma el dato del codificador de tecladò a través de las 4 líneas del puerto C superior y la línea correspondiente al dígito más significativo del puerto C inferior, ingresan los 4 bits correspondientes a la codificación proporcionada por el C.I. 74C923.

Como puede verse en forma gráfica en la figura 2.3, se han aprovechado las líneas que aun no habían sido utilizadas del puerto C del circuito integrado 8255 denominado U7.

2.4.- ADQUISICION DE DATOS E INTERFACES.

Existen varios tipos de datos o variables que deben ingresar al microprocesador para su tratamiento, el primero serán los las instrucciones o datos que pueden ser intercambiados entre el prototipo y un Computador personal en cuyo caso el equipo deberá estar en capacidad de comunicarse utilizando la interface serial estandarizada de la que usualmente dispone un computador personal.

El segundo tipo de datos que debe manejar el prototipo son aquellos de tipo digital que provienen desde circuitos digitales externos o aquellas señales que nuestro equipo debe proporcionar a circuitos digitales que están bajo experimentación en laboratorio normalmente este tipo de señales se las suministra en formato de salida paralelo o lo que es lo mismo los datos se entregaran o capturaran en bloques de 8 o 16 en forma simultanea.

2.4.1.- INTERFACE SERIAL.

La comunicación o intercambio de datos entre dispositivos electrónicos implica el envío recepción de niveles lógicos a través de líneas de transmisión (Cables) y normalmente requiere de la adaptación de estas señales lógicas a niveles de voltaje adecuados. Al trabajar con circuitos lógicos basados en microprocesadores, estamos hablando de niveles con los que trabaja la familia TTL y CMOS, las cuales asumen que el Cero Lógico es igual a cero voltios y Uno Lógico

se representa por 5 voltios.

En realidad se tratan de rangos puesto que cualquier voltaje inferior a 0.8 Voltios se interpretará como cero lógico y en forma similar cualquier voltaje entre 2 voltios y 5 voltios indicaran uno lógico.

Si la comunicación se establecerá entre dispositivos ubicados a pequeña distancia, la transferencia de bits se la puede realizar conectando directamente el transmisor y receptor, y se utilizaran los niveles de 0 y 5 voltios respectivamente, sin embargo si la distancia es superior a los 2 metros, factores tales como el ancho de banda característico de la línea de transmisión, la atenuación y la velocidad con la que se realiza la transferencia, influyen en la representación digital de la información de manera notable.

Esto a originado que se defina diferentes "Interfaces" a fin de solucionar este tipo de inconvenientes, la más popular en el caso de comunicaciones seriales es la denominada RS-232, la cual en esencia implica un cambio de niveles de las señales de entrada/salida de los niveles de 0 y 5 voltios a dos niveles de voltaje que sean mas resistentes a los efectos del medio de transmisión y de mayor magnitud, estos niveles son uno positivo (+V) para representar el cero lógico y uno negativo (-V) en el caso de uno lógico.

La interface EIA RS232C es en la practica un conjunto de señales agrupadas en un protocolo estandarizado para los Estados Unidos por la Electronic Industries Association EIA y a nivel internacional por la norma V.24 del CCITT, para la conexión entre un DCE (Data Communications equipment) y un DTE (Data Terminal Equipment).

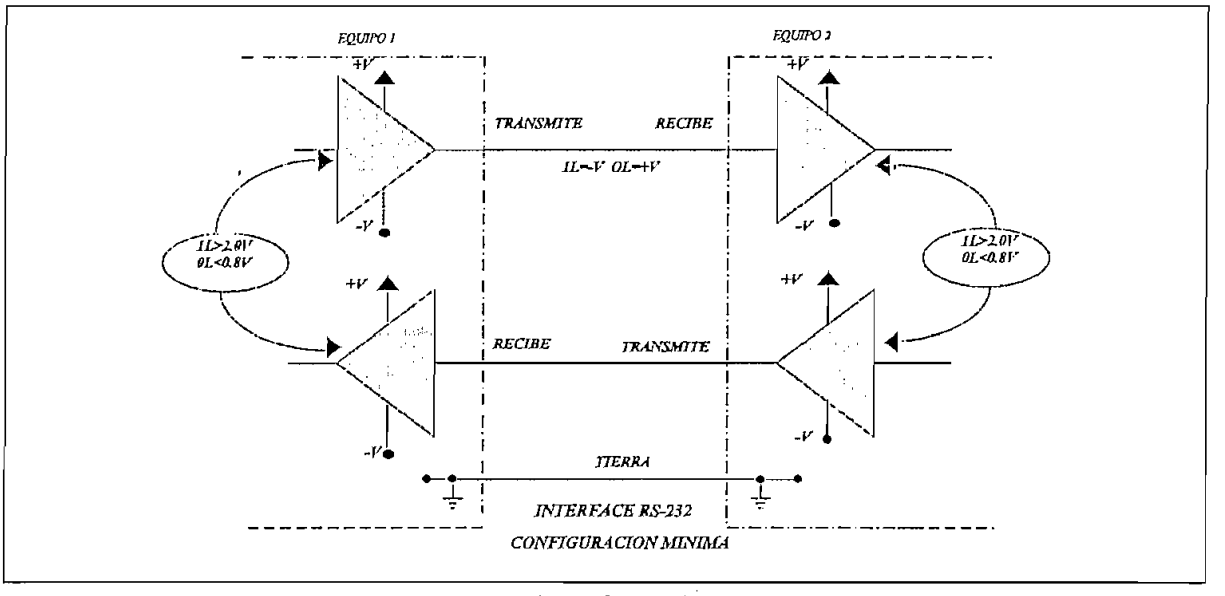


Fig. 2.9. Drivers usados en comunicación serial.

El cable especificado por la EIA esta formado por 25 conductores sobre cada uno de los cuales viaja una señal identificada por un número y un nombre cuya función esta especificada en el estándar. En el presente proyecto sin embargo se utiliza un conector DB9 que es una de las variantes de esta especificación.

Lo niveles de voltaje al ser positivos y negativos mantiene una simetría respecto a tierra y son de al menos +3 Voltios para 0L y - 3 voltios para 1L, sin embargo de que en realidad se trata de un rango de voltajes de activación los cuales se describen a continuación:

VOLTAJE DE LA SEÑAL DE ENTRADA VIN	ESTADO DE LA SEÑAL
3.0 < Vin < 25.0 V	Espacio, On, 0
-3.0 < Vin < 3.0 V	Sin señal, Línea abierta
-3.0 < Vin < -25.0 V	Marca, Off, 1

Tabla. 2.8 Especificaciones eléctricas de la norma RS-232

Resulta inusual un sistema que use las señales sobre los 25 conductores, una interface típica usa aproximadamente la mitad de las funciones disponibles, los restantes normalmente son ignorados.

La interface serial denominada **EIA RS-232 ó CCITT V24/V28**, fue introducida en 1962 y por tanto es anterior al aparecimiento de la lógica de transistor a transistor, por lo cual los niveles de voltaje utilizados en un puerto RS-232-C, tal como se aprecia en el cuadro anterior, difieren de los niveles TTL manejados por el Microcontrolador en su puerto serial, esto evidencia la necesidad de una etapa de interface que adapte las señales RS-232 recibidas desde un pórtico serial de un computador personal a niveles TTL y viceversa, existen para tal efecto varios circuitos integrados de los denominados drivers/receivers tales como los SN75188 y SN75189.

Realmente son las fuentes de alimentación a las cuales se conectan los circuitos de la interface los que determinan los niveles de voltaje siendo los más comunes los que van de ±12 V a ±15 V.

Dos elementos son necesarios para formar una interface RS-232, el primero es el circuito transmisor que se encarga de transformar la señal de voltaje relativamente bajo entregada por el dispositivo lógico hasta niveles de voltaje más altos requeridos por la línea de transmisión y el segundo lo constituye una etapa de recepción, que se ocupa de adaptar los niveles de recibidos hasta aquellos que espera un dispositivo TTL o CMOS.

Uno de los principales problemas que afecta a este tipo de transmisión de datos es el ruido captado por la línea de transmisión. De resultar en un factor importante de alteración se puede utilizar un condensador externo en paralelo con la salida del circuito.

Tal como ya se menciono los circuitos integrados llamados line drivers y line receivers son los adecuados para la implementación de una interface estandarizada, sin embargo esta solución implica la necesidad de una fuente con salidas de doble polaridad, esto es voltajes positivos y

negativos tal como se aprecia en el esquema lógico

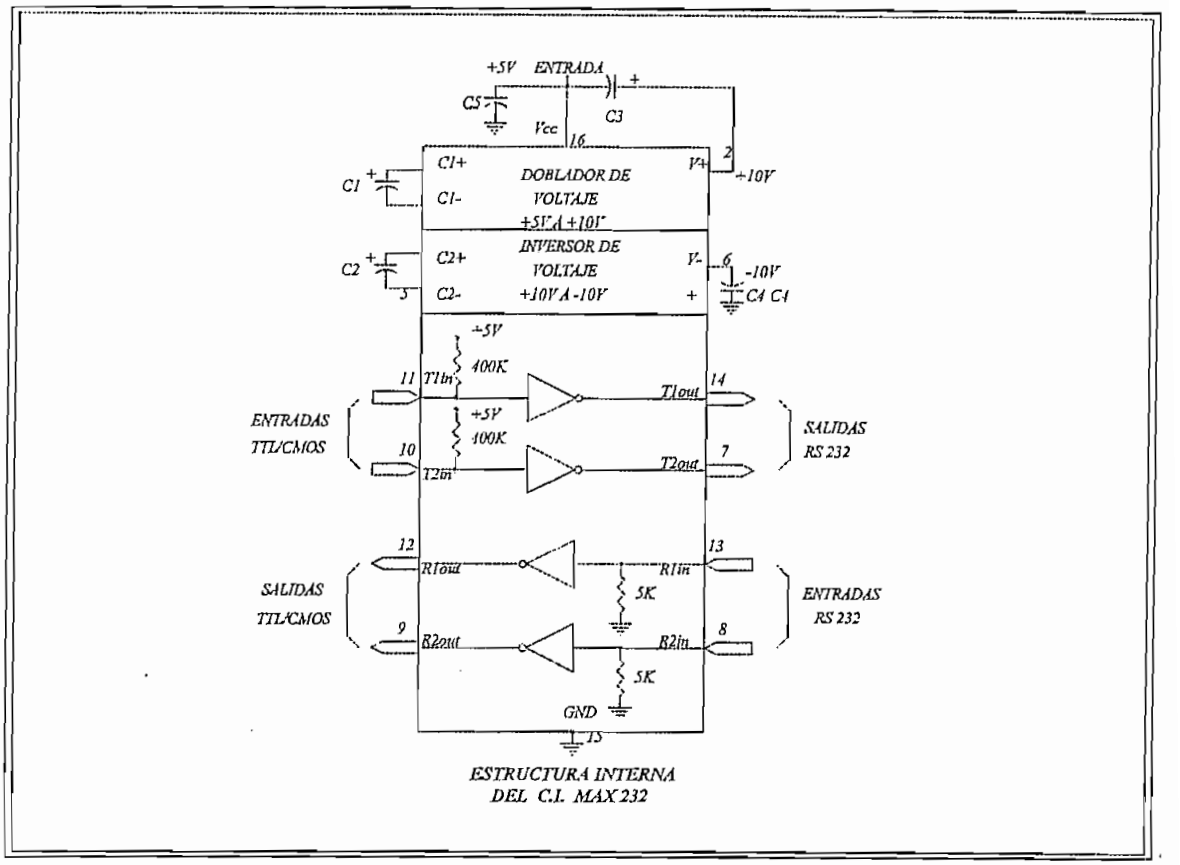


Fig. 2.10. Estructura de un C.I. MAX 232

En el mercado sin embargo existe el circuito integrado MAX232 de la línea de drivers/receivers de MAXIM, el cual incluye driver y receiver en un solo paquete y ha sido desarrollado para aquellas aplicaciones de comunicaciones RS-232 o V24/V28 en las que se procura no utilizar una fuente adicional de ±12 Voltios. Estos drivers RS232 tienen convertidores de voltaje internos los cuales transforman las señales de entrada de +5 V a niveles de salida de ±10 V necesarios para generar los niveles de salida RS232.

Este circuito integrado necesita de 4 capacitores externos para su operación, el valor de los cuales se determina desde las especificaciones del fabricante que aparecen en las hojas del anexo

correspondiente y son los siguientes.

DENOMINACION	VALOR	VOLTAJE
C1	10µF	
C2	10µF	6.3 V
C3	10µF	6.3 V
C4	10µF	6.3 V
C5	10µF	6.3 V

Tabla 2.9. Valores de capacitores del circuito Max 232

Basados en la figura 2.8 en la cual se aprecia la estructura interna del C.I: MAX 232, existe un duplicador de voltaje de +5 V a +10 V y un inversor de Voltaje que permite obtener -10 V

.En la primera etapa se usa el condensador C1 para doblar los +5 V de entrada hasta +10 V sobre el condensador C3 en la salida positiva de V+, Mientras que el segundo convertidor usa el condensador C2 para invertir de +10V a -10V en el condensador C4 de la salida negativa.

En la figura 2.11. se aprecia la conexión de los capacitores en las entradas respectivas y se observa además como las dos líneas, esto es TX y RX que salen desde el microcontrolador llegan hasta el C.I. MAX232.

En el presente proyecto no se ha previsto la transmisión o recepción de señales para el control de flujo de datos o para temporización, y el modo de trabajo del puerto será asincrónico por lo que tampoco están presentes señales de temporización; tratándose por tanto de el modo de transmisión serial más simple.

2.4.2.- INTERFACE PARALELA COMO ENTRADA/SALIDA DE SEÑALES DIGITALES.

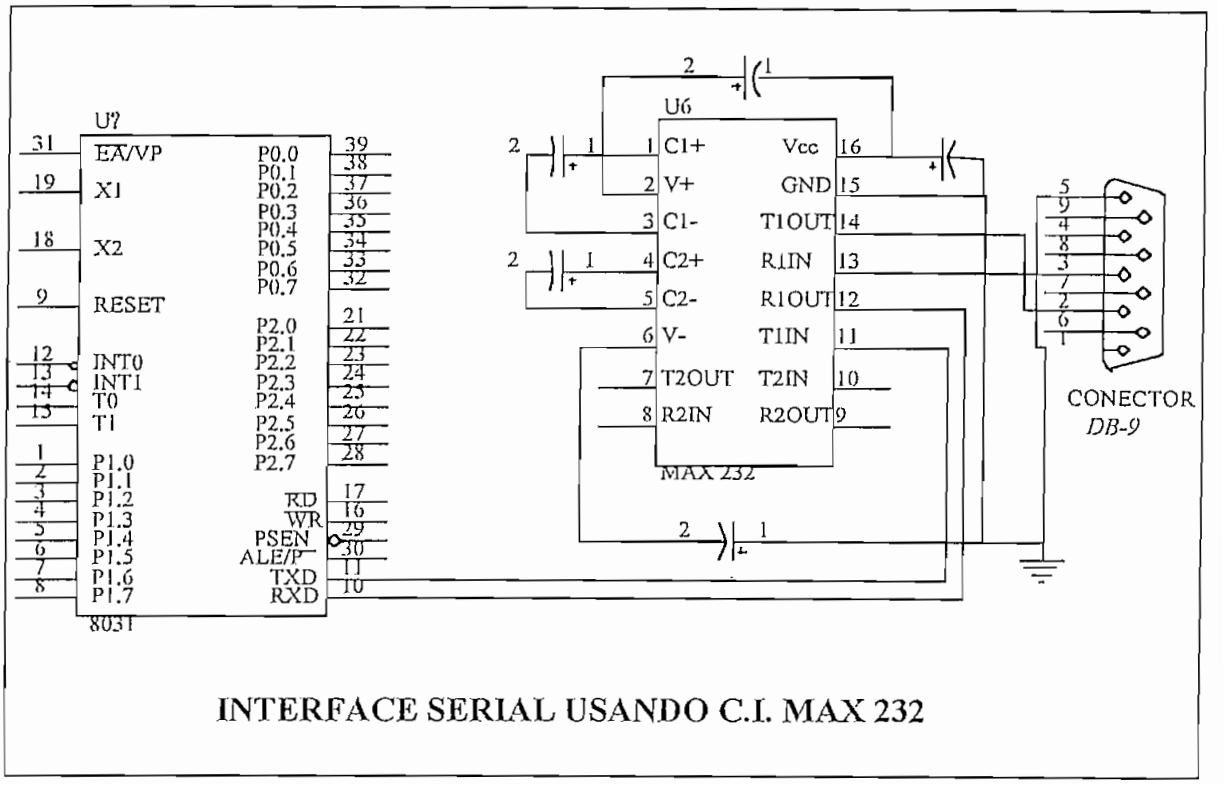


Fig. 2.11. Circuiteria necesaria para la comunicación serial.

Los circuitos integrados 8255 se han utilizado como el medio idóneo para incrementar el numero de lineas de entrada salida que estan en capacidad de tomar las señales digitales en forma simultanea para su posterior procesamiento por parte del prototipo, al mismo tiempo que se constituyen en el medio a través del cual se pueden entregar resultados o señales digitales en formato paralelo hacia el medio externo.

2.4.2.1.- INTERFACES PARA ENTRADA DE SEÑALES DIGITALES.

Para la adquisición de datos se utiliza uno de los puertos de entrada salida que se crearon

mediante la utilización de C.I.82C55A, específicamente aquel rotulado en el esquema como U11, de este integrado se toman los puertos A y B para conformar un puerto de entrada de 16 bits a través del cual se toman los datos binarios desde cualquier circuito digital que se haya implementado.

Los datos binarios activan al ingresar en el sistema uno de los LEDs para indicar su estado esto es 0_L ó 1_L esto sirve como un medio de comprobación visual primario de los datos que se van ingresando.

Por lo tanto cada una de las líneas de ingreso utiliza una interface constituida de la siguiente manera:

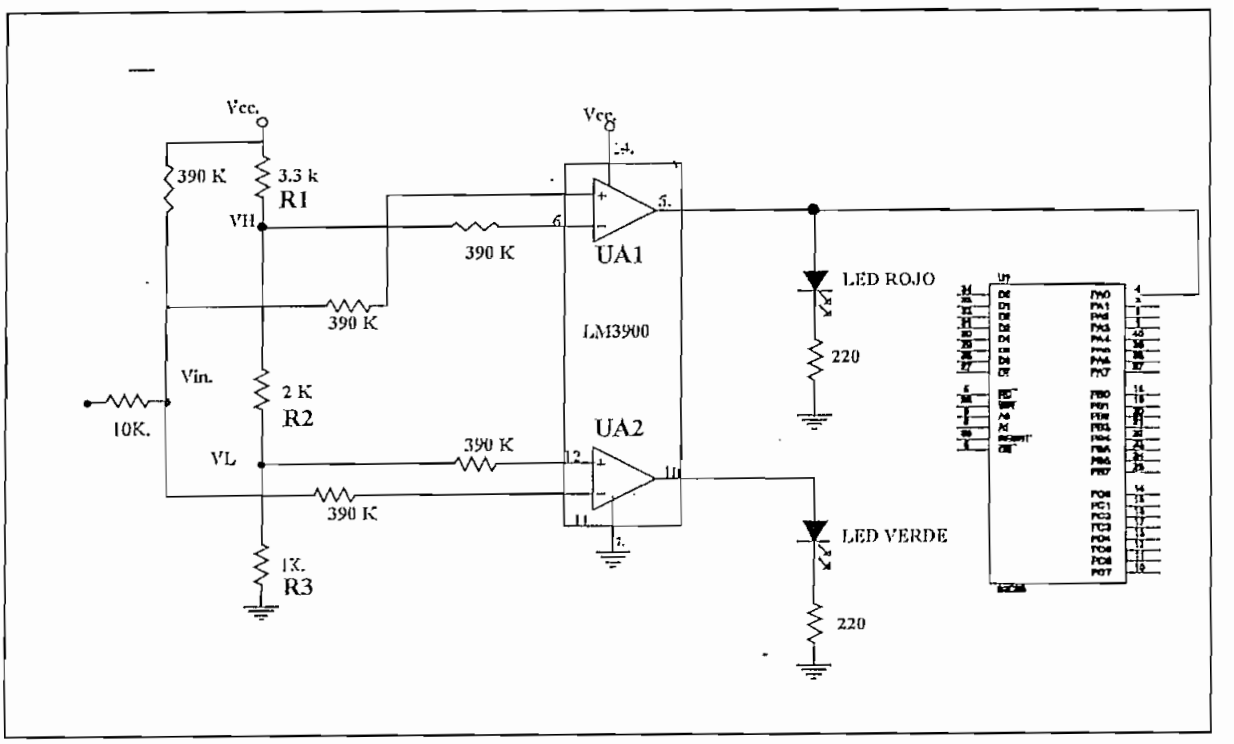


Fig. 2.12. Circuiteria para las interfaces de entrada

Como puede apreciarse en la figura 2.12., la determinación del estado de la señal de entrada se realiza

mediante un comparador de ventana implementado en base a amplificadores operacionales.

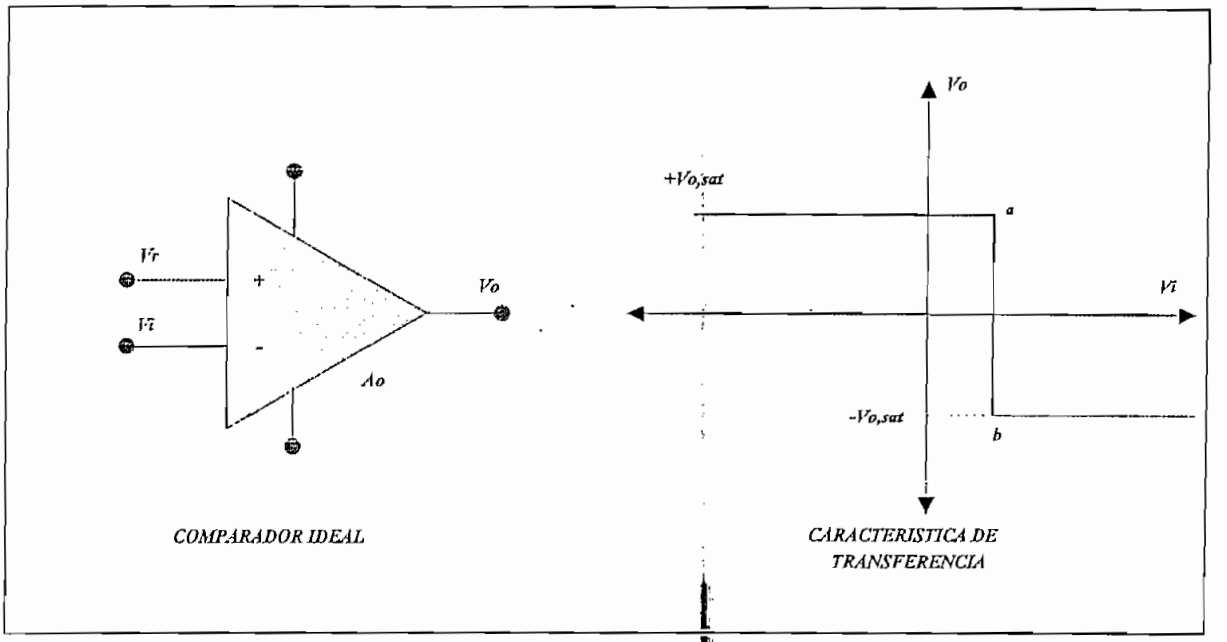


Fig. 2.13. Característica de un comparador ideal

A fin de clarificar el funcionamiento es conveniente recordar la forma de operación de los comparadores de este tipo

Los circuitos de comparación permiten determinar el valor de señales por comparación con el valor de una señal de referencia, en la figura 2.12., puede apreciarse un comparador ideal y su respectiva función de transferencia.

Si la señal se alimenta a la terminal de inversión del amplificador operacional se tiene un comparador con dicha característica, y por el otro lado si se cambia el orden de conexión de Vi y Vr se puede obtener un comparador de no inversión.

De la figura 2.13 en la que se aprecia la característica de transferencia se puede resumir las condiciones de operación en el siguiente cuadro:

VOLTAJE DE SALIDA		
$v_o =$	$V_{O,sat}$	para $v_i < V_R$
	0	para $v_i = V_R$
	$- V_{O,sat}$	para $v_i > V_R$

Tabla 2.10. Voltajes de salida de un comparador de ventana.

Como se puede apreciar el amplificador operacional para el caso del comparador esta operando en condición de ciclo abierto, de donde la pendiente de la recta ab es infinita, en el caso de un amplificador operacional práctico la pendiente es igual a A_o.

El valor de los voltajes de salida esta determinado por los voltajes utilizados para la alimentación de el amplificador operacional, siendo su valor de $V_{o,sat}$ ó $-V_{o,sat}$ según el valor del voltaje a la entrada.

Dado que en el presente proyecto el interés esta en determinar dos estados de las señales que ingresan al prototipo, es necesario definir los voltajes de referencia que se utilizaran para realizar la comparación, estos voltajes corresponderán a valores de 0L y 1L, de acuerdó a la siguiente convención:

$$0Volt =< \mathbf{0}_L =< 0.7Volt \qquad 2.4Volt =< \mathbf{1}_L =< 5Volt$$

El circuito comparador de ventana que aparece en la fig. 2.12., sirve para tomar la muestra de la señal que esta ingresando al prototipo **Vin** y utiliza dos señales $\mathbf{V_H}$ y $\mathbf{V_L}$ como referencias para realizar la comparación.

En el circuito implementado se utiliza un divisor de tensión conformado por R1, R2 y R3, para obtener los voltajes de referencia a partir de Vcc.

Cuando la señal que ingresa como Vin tiene la condición $V_H =< V_{in} =< Vcc$ la salida del amplificador operacional UA1 pasa a condición alta y por lo tanto alimenta al led rojo el cual se enciende señalizando una condición de uno lógico a la entrada de esa línea del prototipo, por otra parte cuando la condición que se presenta es $0Volt =< V_{in} =< V_L$, es la salida del amplificador operacional UA2 la que pasa a su nivel alto y alimenta al led verde, indicando la presencia de un cero lógico.

Cualquier señal de entrada Vin cuyo valor este entre V_L y V_H no produce ningún efecto sobre los amplificadores y por lo tanto ambos leds permanecen apagados.

La señal de salida del amplificador operacional que activa a led rojo se la conecta además a el pin correspondiente de el circuito integrado 8255 de modo que se constituye en la entrada para el procesamiento de la señal en el microcontrolador, existiendo por tanto un circuito de este tipo por cada una de las 16 líneas de entrada de las que dispone el equipo.

Este tipo de conexión permite por otra parte que las señales de salida que el prototipo provea a través de el puerto implementado mediante este circuito integrado activen el led rojo cuando su valor es alto (1 lógico) indicando visualmente el valor de la señal suministrada, aspecto que resulta sumamente útil por ejemplo al momento de generar secuencias lógicas.

2.5.- EL RELOJ SINCRONIZADO CON LA RED.

Para la implementación de un reloj sincronizado con la red de alimentación comercial nos valemos de una de las entradas de interrupción de las que dispone el microcontrolador 8051, de modo que a esta entrada esta conectado una línea que viene desde la electrónica encargada de la detección del cruce por cero de la señal de A.C. de la cual se desea obtener la sincronización

de la señal digital de salida.

Para la detección del cruce por cero, se toman muestras de la señal de corriente alterna inmediatamente después del puente rectificador de onda completa, de modo que se necesita intercalar un diodo entre esta salida y el capacitor encargado de la eliminación del rizado de la señal rectificada.

Esta señal se la conecta a un diodo zener de 3.3 voltios y simultáneamente a la base de un transistor NPN, el colector de este transistor a través de una resistencia limitadora esta conectado a Vcc, mientras que el Emisor esta conectado a tierra.

Como puede apreciarse en la figura2.15, durante la mayor parte de la duración de los ciclos de la señal sinusoidal, el valor de Vz es suficiente para mantener saturada la base de Q1 y por lo tanto el voltaje en el colector del transistor Vc será de 0 Voltios, únicamente durante el corto período de tiempo en que la señal rectificada de onda completa desciende hasta 0 Voltios ó un valor ligeramente superior la señal en la base del transistor es insuficiente para saturarlo, y por tanto el valor de Vc, será igual a Vcc. esto implica que la señal Vc. tendrá la forma de un tren de pulsos que coinciden con los tiempos en los cuales la señal de entrada de AC. pasa por cero.

La señal Vc. se conecta a la entrada de la interrupción externa 1 del microcontrolador 8031, INT1, y la rutina de atención de esta interrupción será la encargada de sacar una señal cuadrada a través de los pines del puerto P1 del microcontrolador 8031, generandose de esta forma una señal digital sincronizada con la red de A.C. sin embargo su frecuencia será el doble de la de la red comercial, esto es en el caso ideal 120 Hz.

En la figura 2.15 se puede apreciar las formas de onda que se obtienen en los diferentes puntos de prueba de l circuito encargado de la detección de cruces por cero y de activación de la entrada de la interrupción externa 1 del microcontrolador 8031.

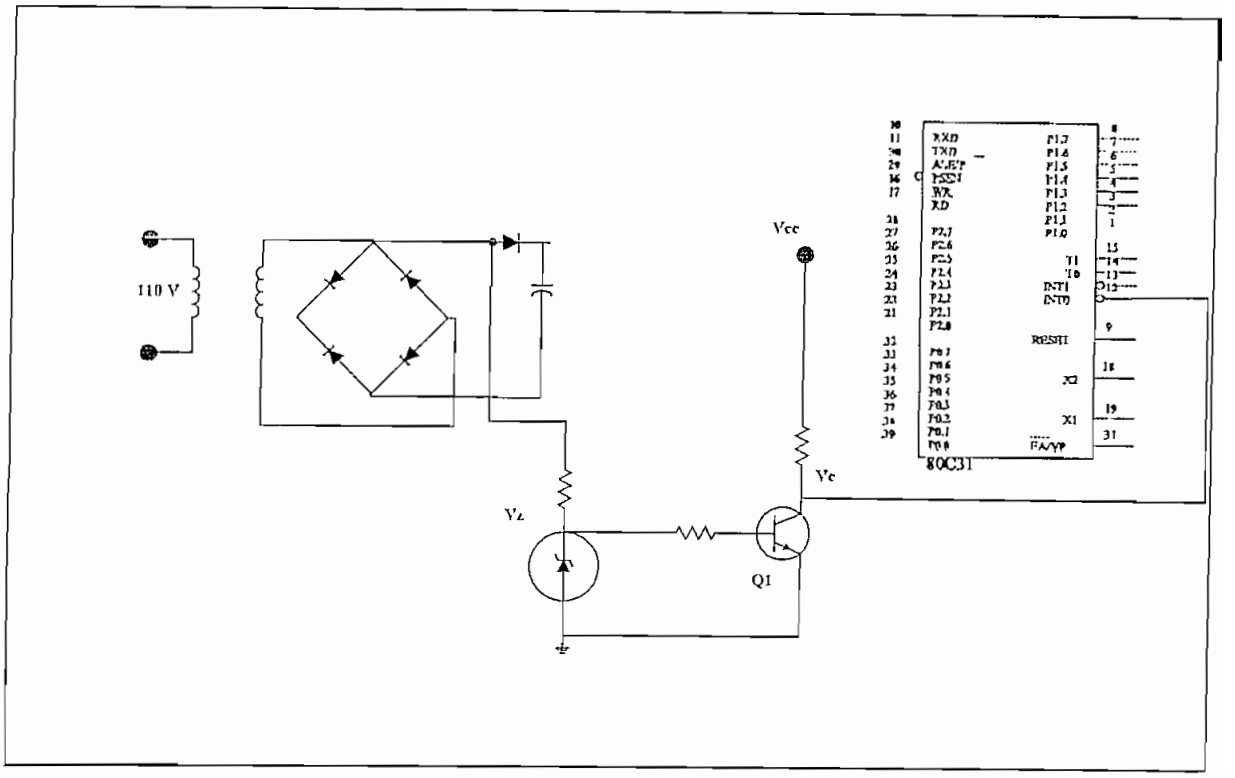


Fig. 2.14. Circuito de detección de cruces por cero

La primera corresponde a la señal de corriente alterna una vez rectificada y muestreada inmediatamente antes del diodo de aislamiento. La segunda es la señal en el extremo del diodo Zener y por tanto su valor máximo es igual a 3.3 Voltios esto es la especificación del Zener utilizado. La tercera señal es un tren de pulsos de duración muy corta y amplitud igual a Vcc, que se obtiene en el colector del transistor y a su vez se conecta al pin correspondiente a la interrupción externa INT1 del microcontrolador y por tanto es la que activa la generación de la señal de reloj digital sincronizada con la frecuencia de la señal de AC que alimenta a todo el prototipo.

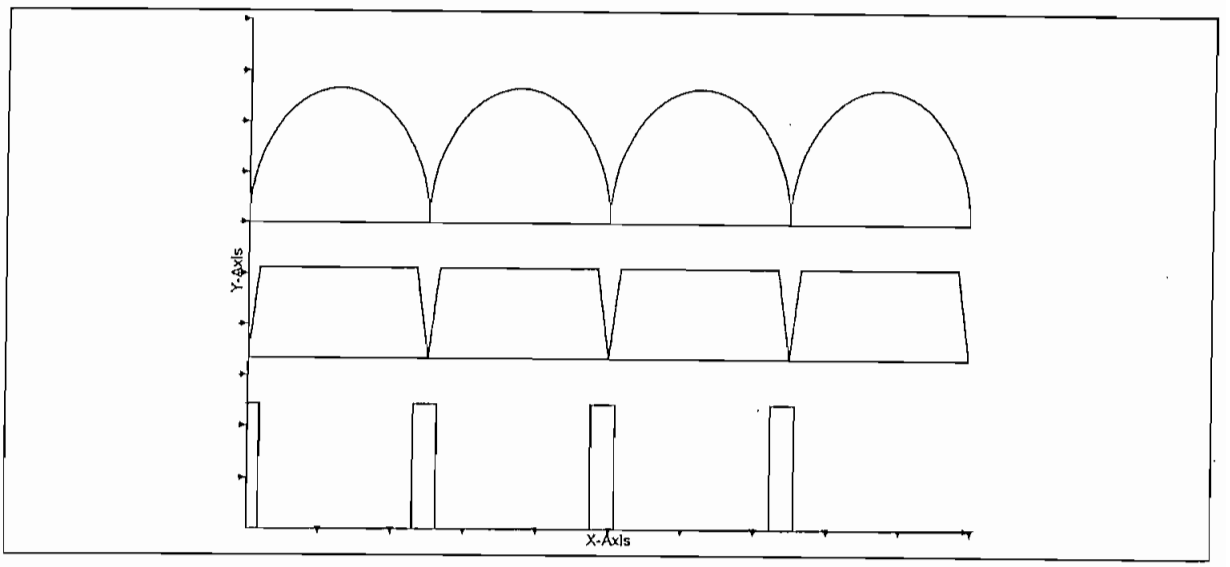


Fig. 2.15. Señales del circuito de detección de cruces por cero.

2.6.- MANEJO DE CARGAS A.C.

Para el manejo de cargas A.C. se utilizan Triacs , si bien es posible utilizar otros dispositivos como transistores de Potencia, relés electromecánicos o relés de estado sólido; pero esta se constituye en una buena oportunidad para familiarizar a los estudiantes con dispositivos electrónicos de control.

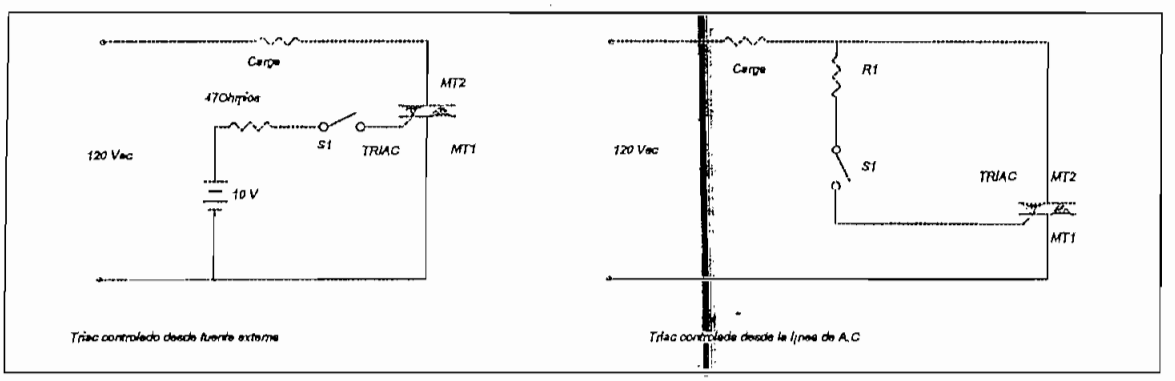


Fig. 2.16. Circuitos de disparo de TRIACS.

Un Triac puede efectuar varios tipos de tareas de control sin embargo una de las formas más usadas es en reemplazo de relés mecánicos en funciones de control de tipo encendido-apagado (on-off), la circuiteria de control para este tipo de aplicaciones es muy simple, consistiendo de una fuente para la señal de compuerta y algún tipo de conmutador de pequeña señal, sea mecánico o eléctrico. La señal de compuerta puede obtenerse desde una fuente separada o directamente desde el voltaje de la línea MT2 del Triac.

Para evitar fallas debidas a la corriente de suministro durante el encendido, la corriente de compuerta I_{GT} deberá ser de 5 a 10 veces la corriente máxima de especificación de la compuerta.

La activación de un tiristor requiere que se satisfagan las especificaciones de energía de compuerta, para lo cual existen varias opciones. En general las compuertas deben ser manejadas con la fuerza y rapidez suficiente.

Usualmente esto significa que se deberá suministrar una corriente de compuerta de al menos tres veces el valor especificado de la corriente de compuerta de encendido mediante un pulso con duración mínima de un microsegundo y máximo diez microsegundos.

La compuerta puede ser manejada también por una fuente de DC con un valor similar al promedio de los valores especificados para la compuerta.

Algunos de los métodos de manejo de compuerta incluyen:

- Manejo directo desde familias lógicas de transistores
- Manejadores por Opto Triacs
- Transistores Unijuntura (UJTs) o transistores unijuntura programables (PUTs).
- Switchs bilaterales de Silicon (SBSs)
- Diacs

En nuestro caso la mas adecuada por los requerimientos de aislamiento eléctrico resulta ser el manejo mediante aisladores/acopladores ópticos.

Un aislador optoelectrónico combina un dispositivo emisor de luz y un fotodetector en un mismo paquete opaco que provee protección contra la luz del medio ambiente.

Puesto que no existe una conexión eléctrica entre la entrada y salida, una señal puede pasar a través del acoplador únicamente en una dirección.

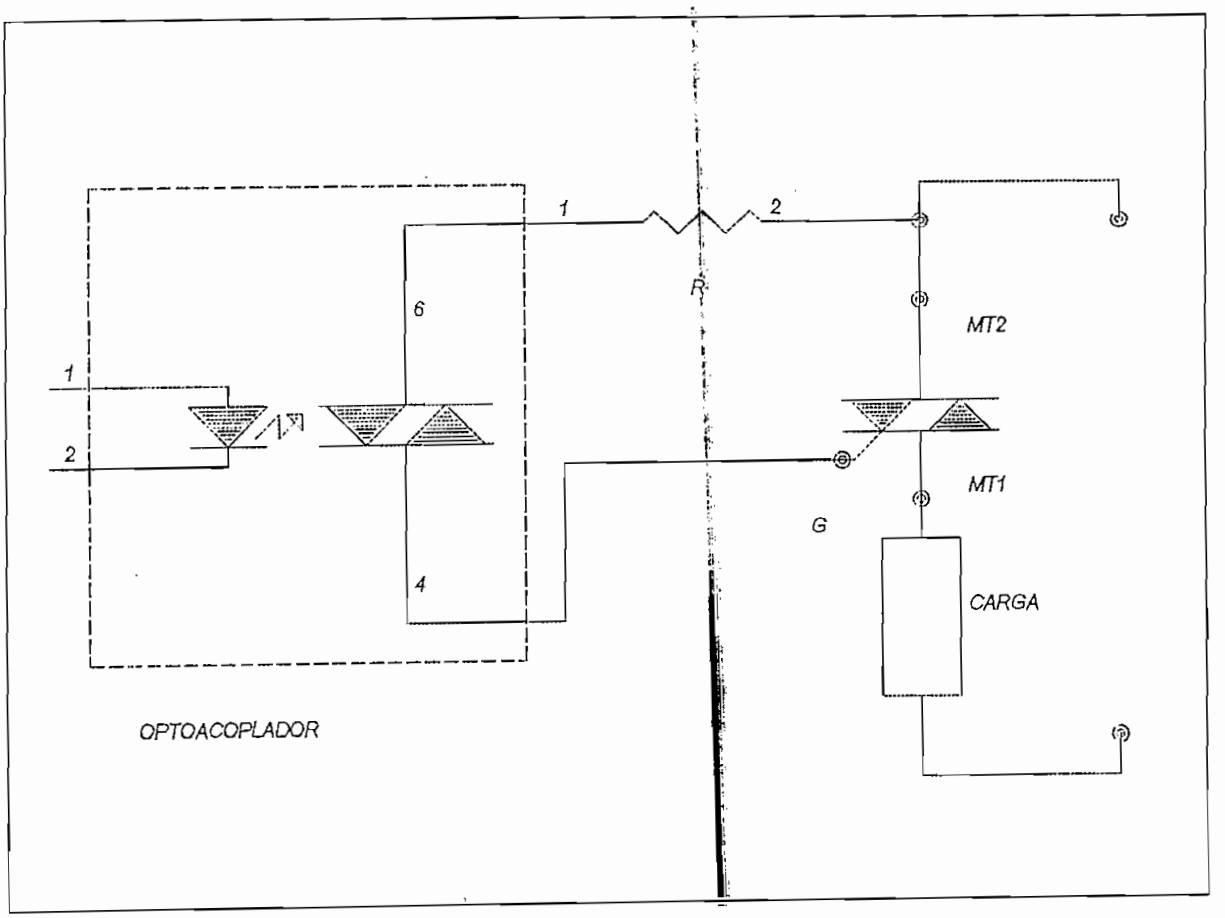


Fig. 2.17. Disparador de TRIAC con optoacoplador.

Debido a que el optoacoplador provee de protección y aislamiento entre la circuiteria de entrada

y las condiciones que se presentan en los circuitos de salida, los optoacopladores son ampliamente utilizados como parte de la lógica de control de circuitos de alto voltaje. En la figura 2.17., un fototriac maneja a un triac de alto poder, el fototriac es sensible a la luz y pasa a su estado de encendido por una cierta densidad especifica de luz incidente la cual es suministrada por el LED y que a su vez es función de la corriente que circula por el LED.

La capacidad de corriente de salida del foto-triac esta típicamente alrededor de los 100 mA si es continua, pasando a 1A como valor pico.

El valor de la resistencia R que aparece en la figura 2.17., esta determinada por la capacidad de corriente que maneja el Foto-Triac. Por ejemplo si se utiliza un foto triac con una especificación de 1 A como corriente máxima, el valor de R mínimo considerando un voltaje de línea de 120 V, se lo puede calcular en la siguiente forma:

$$V_{ACmax} = 120 * 1.4142 = 170 \text{ V}$$

$$R = 170V/1A = 170 \text{ ohmios}$$

En la siguiente tabla se pueden observar las especificaciones de algunos de los optoacopladores mas comunes

TIPO DE DISPOSITIVO	MAXIMA CORRIENTE REQUERIDA POR EL LED (mA)
MOC3009	30
MOC3010	15
MOC3011	10
MOC3020	30
MOC3021	15
MOC3030	30
MOC3031	15

Tabla 2.11. Especificaciones de optoacopladores comerciales.

Los sistemas basados en microprocesadores son capaces de controlar cargas alimentadas por AC cuando están adecuadamente conectados con tiristores.

Los puertos de salida de los microprocesadores ó microcontroladores disponen generalmente de salidas compatibles con niveles TTL con capacidades de suministro y drenaje de corriente significativamente menores que un dispositivo TTL estándar.

Un microcontrolador con salidas compatibles con TTL como el MC-8031 de INTEL presenta las siguientes especificaciones.

$I_{OH} = 80$ uA $V_{OH} = 2.4$ V
$I_{OL} = 1.6$ mA $V_{OL} = 0.45$ V
$Vcc = 5$ V

El cual claramente resulta inadecuado para el manejo de optoacopladores con requerimientos de corriente de activación como los que aparecen en la tabla.

La solución mas práctica en estos casos es utilizar uno de los elementos estándar 7400 para crear la interface que maneje a los optoacopladores tal como aparece en la figura 2.18

La forma de operación es la siguiente cuando la señal de 1L provista por una de las salidas del microcontrolador se combina con la otra entrada de la compuerta NAND que esta fijada a un valor alto, se obtiene como resultado un valor de 0L que esta conectado a uno de los pines del optoacoplador, por lo tanto el led contenido en ese dispositivo se enciende, la luz emitida dispara el fototriac, puesto que al mismo tiempo el triac principal no esta encendido, entre MT2 y la compuerta tenemos un circuito abierto.

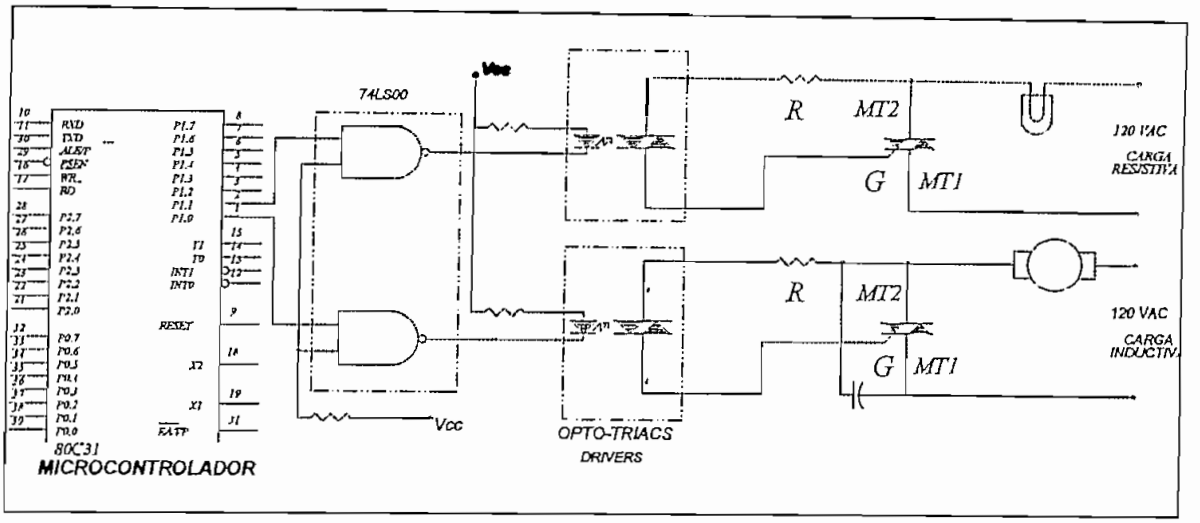


Fig. 2. 18. Circuito de manejo de cargas de AC desde un microcontrolador.

La señal de alimentación de AC causa que la corriente fluya a través de R, el foto triac , la compuerta, la juntura Compuerta-MT1 y la carga, la corriente a través de la Compuerta y MT1 disparan el triac principal, el cual corta entonces la corriente a través de el foto-Triac .

Por lo tanto el disparo del TRIAC de la etapa de potencia depende del encendido del led del optoacoplador que es comandada por la etapa digital.

CAPITULO III.

SOFTWARE

3.1.- DESCRIPCION GENERAL DEL SOFTWARE.

El equipo ha sido diseñado teniendo como elemento fundamental a uno de los miembros de la familia de microcontroladores MCS-51, por lo tanto los programas que determinan su funcionamiento se desarrollan en el lenguaje de máquina correspondiente.

Para su elaboración utilizamos el Sistema Integrado para el Desarrollo de Software SIDES. Este contiene un editor de texto mediante el cual podemos escribir el programa fuente utilizando los mnemónicos propios del microcontrolador.

Para ensamblar el programa fuente disponemos del Meta Cross Assembler C16, con lo cual se obtienen los códigos de máquina que deben transferirse a la memoria de programa del equipo.

Mediante el uso de el AVSIM51 se realiza la simulación de las diferentes rutinas para de este modo realizar una depuración del los programas hasta que estos satisfagan todas las expectativas.

El software desarrollado debe controlar el ingreso de los datos al microprocesador, procesar estos datos, ejecutar los comandos determinados por el usuario, emitir las señales resultantes y establecer las comunicaciones para el intercambio de datos de los cuales debe estar dotado el dispositivo.

Debido a la forma que se ha establecido para la comunicación usuario-equipo y equipo-computador personal resulta esclarecedor un estudio previo de la forma en que se manejan las interrupciones y el funcionamiento del puerto serial de los que dispone el microprocesador.

3.1.1.- INTERRUPCIONES

Los miembros de la familia MCS-51 disponen de cinco fuentes de interrupción, a excepción del 8032/8052 que posee seis.

Dos registros de función especial, **IE** e **IP** son los encargados de controlar y monitorear la habilitación y prioridad de las interrupciones, respectivamente.

La asignación de bits del registro IE aparece a continuación:

MSB LSB

EA	X	X	ES	ET1	EX1	ET0	EX0

Fig. 3.1. Registro de habilitación de interrupciones.

EA.- Habilitación general de las fuentes de interrupción.
ES.- Habilita o deshabilita la interrupción del pórtico serial
ET1.- Controla la habilitación de la interrupción del Timer 1.
EX1.- Controla la habilitación de la interrupción externa 1.
ET0.- Controla la habilitación de la interrupción del Timer 0.
EX0.- Controla la habilitación de la interrupción externa 0.

Cada una de las interrupciones puede ser habilitada (1L) o deshabilitada (0L) bajo control de los bits respectivos. Se dispone además de una habilitación general mediante **EA**, si esta no ha sido seteada todas la interrupciones permanecen deshabilitadas, aunque individualmente su estado sea de habilitación.

La asignación de bits del registro **IP** es la siguiente:

MSB LSB

X	X	X	PS	PT1	PX1	PT0	PX0

Fig. 3.2, Registro de control de prioridades de interrupciones.

PS.- Define la prioridad de la interrupción Serial.

PT1.- Define la prioridad de la interrupción debida al Timer 1

PX1.- Define la prioridad de la interrupción externa 1

PT0.- Define la prioridad de la interrupción debida al Timer 0

PX0.- Define la prioridad de la interrupción externa 0

Las interrupciones cuyos bits sean 1_L resultan ser las de mayor prioridad. Pero puede darse el caso en que todos los bits contengan 1_L, el caso inverso ó un caso en que dos interrupciones de igual prioridad asignada piden atención, entonces la resolución hace uso de la siguiente tabla:

Fuente	Prioridad
IE0	Mayor prioridad
TF0	
IE1	
TF1	
RI+TI	Menor prioridad

Tabla 3.1. Orden de Prioridad de la Interrupciones.

Un tercer registro interviene conjuntamente con las interrupciones es **TCON** , y su distribución de bits aparece a continuación:

MSB LSB

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Fig. 3.3. Registro de control de temporizadores.

Los bits IT0 be IT1 de este registro determinan si las interrupciones externas INT0 e INT1 son activadas por flanco o por estado; mientras que los bits IE0 e IE1 actúan como banderas de las interrupciones, las mismas que pasan a 0_L al momento en que se atiende a dicha llamada, esto siempre y cuando las interrupciones externas estén trabajando en modo de activación por flanco; pues si son activadas por estado, la fuente de la interrupción mantiene el control de las banderas.

Los Timers 0 y 1 generan las interrupciones cuando los valores correspondientes en los registros sobrepasan el valor máximo para el cual fueron definidos según el modo de trabajo seleccionado, en ese momento los bits **TF0** y **TF1** pasan a 1_L excepto para el timer 0 trabajando en modo 3. Estas banderas son limpiadas al activarse las rutinas de servicio que atienden a estas interrupciones.

Por su parte la interrupción generada por el puerto serial hace uso de las banderas **RI** y **TI**, seteadas mediante una lógica OR, dejando a cargo de la rutina de servicio la determinación del origen específico de la interrupción, la cual adicionalmente deberá borrar las banderas.

El software esta en capacidad de generar o cancelar cualquiera de las interrupciones, desde el momento en que puede pasar a 1_L o 0L todas las banderas de control de las mismas.

Durante la ejecución de una rutina de servicio puede producirse una nueva interrupción solo si la prioridad de la fuente que solicita atención es mayor a la actual, de lo contrario no será tomada en cuenta hasta finalizar la tarea en proceso.

Cuando el procesador reconoce un pedido de interrupción, desde hardware se genera la

instrucción LCALL a la rutina de servicio correspondiente y en algunos casos simultáneamente la bandera de la interrupción pasa a 0_L, pero en otros casos esto debe hacerse por software.

Al ejecutar la instrucción de salto de 16 bits, automáticamente se resguarda el contenido del PC dentro del stack, pero resulta indispensable que el acumulador, el **PSW** y cualquier otro registro conteniendo información relevante sean salvados mediante el uso de la instrucción **PUSH**, y posteriormente deberán ser rescatados con las correspondientes instrucciones **POP**.

Una vez resguardado su contenido el PC es cargado con la dirección del vector de la interrupción que generó el pedido. Las localidades de memoria utilizados por estos vectores se aprecia en la siguiente tabla:

Fuente	Direcc. del Vector
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI+TI	0023H

Tabla 3.2. Tabla de vectorización de interrupciones.

La ejecución de una rutina de servicio a una interrupción termina al encontrar la instrucción RETI, momento en cual se recarga el PC desde el stack y se continua con la ejecución del programa previo.

Como ya se menciono las fuentes de interrupción externa pueden ser programadas para su activación por transición o por estado.

Si **ITx** = 0_L la interrupción externa x es disparada por un nivel bajo en el pin INTx.

Si **ITx** = 1_L la interrupción externa x se activa por transición. En este modo si en muestras

sucesivas del pin INTx aparece en un ciclo en nivel alto y en el siguiente se tiene un nivel bajo, la bandera de pedido de interrupción IEx en el registro TCON pasa a 1_L activandose el pedido de interrupción.

Puesto que los pines de interrupción externa son muestreados una vez por ciclo, la señal de entrada deberá mantenerse en alto o bajo por al menos 12 períodos del oscilador para que resulte adecuadamente interpretada.

3.2.- DISEÑO DE LOS ALGORITMOS.

"Conceptualmente, un sistema basado en microcontrolador típico lee señales o datos desde sus puertos, ejecuta las instrucciones apropiadas de manipulación de datos dependiendo de los valores de las señales (período de ejecución) y finalmente emite los resultados (datos procesados) como señales de salida." [1]

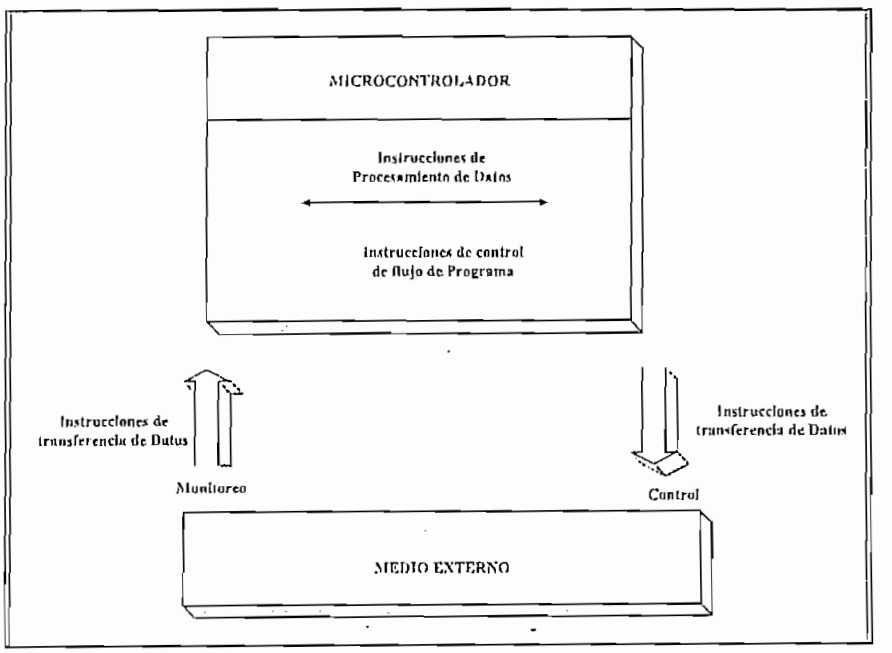


Fig.3.4. Interrelación entre el Microcontrolador y el medio.

[1] Programing and interfacing the 8051 Microcontroller, Cap 2, Pag. 34

Las instrucciones del microcontrolador pueden ser separadas en instrucciones de transferencia de datos , instrucciones de procesamiento o manipulación de datos e instrucciones de control de flujo, en base a estas instrucciones y su adecuada utilización que se elaboran los algoritmos que están en capacidad de proveer los resultados esperados de un sistema basado en microcontrolador, esta forma de trabajo se aprecia en la figura 3.4.

Dependiendo de la aplicación que se desarrolle el programa resultante podrá tener un mayor o menor grado de complejidad, relacionándose también el tamaño del programa resultante con las diferentes funcionalidades que debe proveer.

La programación resultara mucho más fluida y agil si se divide el algoritmo general en pequeñas unidades dedicadas a una funcionalidad específica.

Si los programas se dividen en pequeños módulos, la codificación resulta mucho más sencilla, facilitándose al mismo tiempo la detección y corrección de cualquier tipo de errores. Esta forma de diseñó de los algoritmos mantiene similitudes con la forma en la que se diseña el hardware con bloques independientes que trabajan en forma independiente entre si pero bajo la coordinación del microcontrolador.

Cada uno de los módulos tiene entradas y salidas específicas que deben mantener la concordancia con el resto de bloques de modo que la totalidad de la programación mantenga una consistencia adecuada. Una vez comprobados por separado cada una de estas unidades se integran formando el programa general.

Adicionalmente es muy común que una de los módulos independientes o subprogramas pueda ser utilizado por varias de las rutinas del programa principal, dependiendo entonces los resultados de que las entradas para ese módulo sean especificadas adecuadamente.

Normalmente estos módulos independientes se identifican por nombres o etiquetas que son utilizadas como el medio para llamarlos desde el programa principal en el momento que se los requiere.

Cuando un programa debe ser modificado, la programación modular facilita el actuar únicamente en la zona del programa que requiere de los cambios o de ser el caso se puede añadir un bloque nuevo de programa sin que toda la programación completa deba ser analizada nuevamente.

Precisamente esta técnica de programación modular se aplica para la creación del software encargado de la operación de el prototipo construido en el desarrollo del presente proyecto.

3.3.- PROGRAMA PRINCIPAL.

El programa principal viene a constituirse en el "Sistema Operativo" del equipo y es el que determina cual de las funciones debe entrar en operación, la secuencia de ejecución del mismo puede observarse en el diagrama de flujo que aparece en la figura 3.5.

Dentro de este se realiza la inicialización de los diferentes registros de control del microprocesador que determinan su funcionamiento. Entre estas tenemos la habilitación de las interrupciones, medida necesaria desde el momento que el ingreso de los comandos se realiza a través del teclado el cual envía una señal que interactua con la interrupción externa 0, es la rutina de servicio de esta interrupción la encargada de leer el código ingresado y se la trata en forma más detallada en apartado correspondiente.

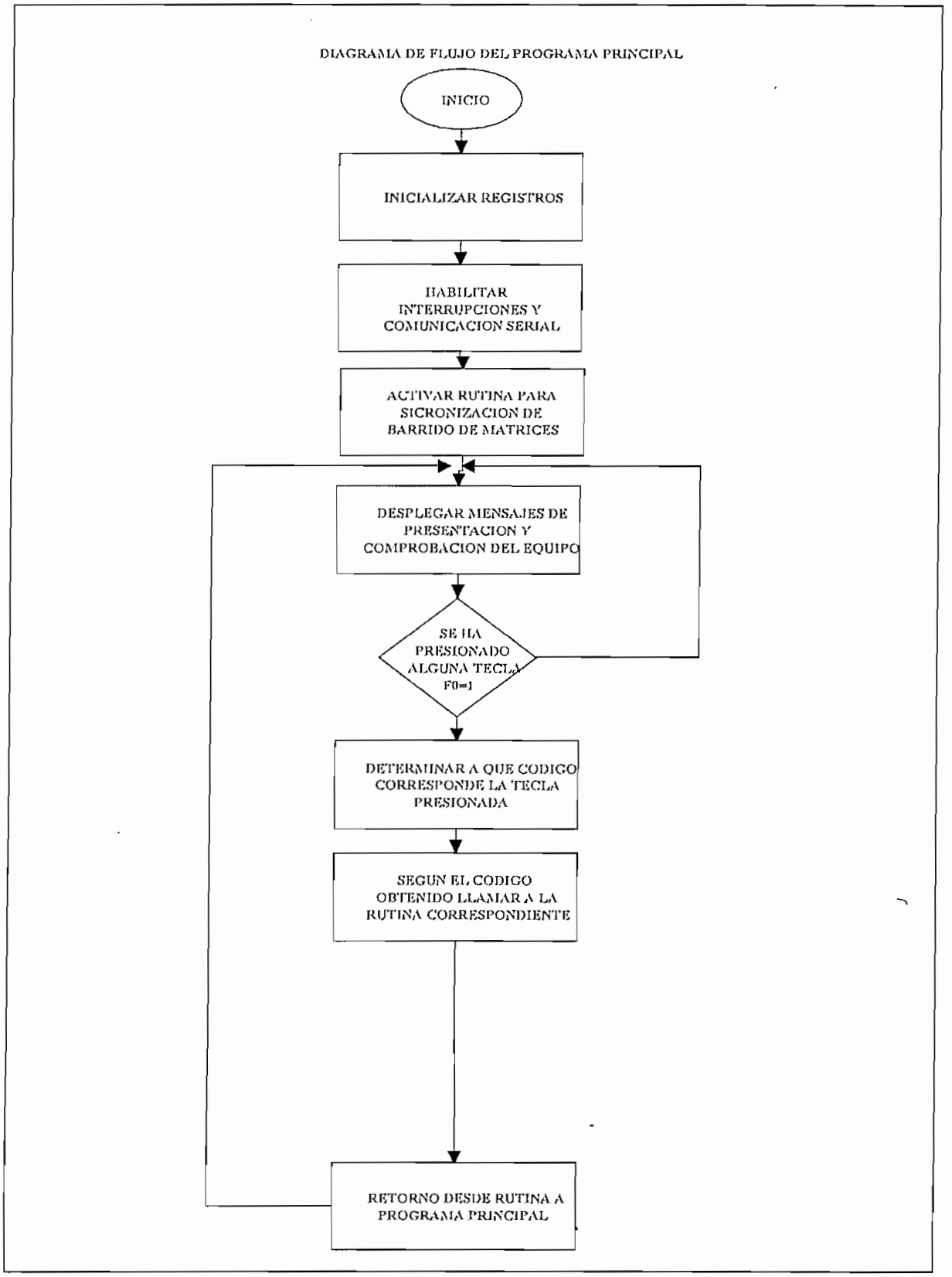


Fig 3.5. Diagrama de Flujo del programa principal

Al encender el equipo aparecen en pantalla una serie de mensajes que tienen doble finalidad; en primer lugar realizan una presentación del sistema y además sirven para comprobar el adecuado funcionamiento de las matrices de leds que conforman el visualizador del dispositivo.

Cuando mensaje LAB_DIGI aparece el equipo esta listo para que el usuario decida cual de las funciones quiere usar. Esta selección se la realiza mediante el uso del teclado.

Según el código de la tecla presionada y mediante el uso de una tabla de selección el programa principal ordena la ejecución de la rutina correspondiente.

Las funciones incluidas en el prototipo están asociadas a las teclas comprendidas entre 00H y 09H, según se detalla en la tabla 3.3.

TECLA	FUNCION
0	Transformación de binario a Hexadecimal
1	Transformación de Binario a BCD
2	Generación de secuencia
3	Generación de secuencias lógicas
4	Reserva
5	Reserva
6	Reloj sincronizado con la red A.C.
7	Comunicación Serial/ Ejecución desde RAM como ROM
8	Leer / modificar RAM
9	Control cargas A.C.

Tabla 3.3. Asignación de funciones a teclas.

A fin de determinar cual de estas teclas a sido elegida se utiliza una tabla y la selección para ejecución se la realiza mediante saltos a rutinas que se inician en etiquetas correspondientes a estas opciones.

Mediante la utilización de la rutina de atención a la interrupción externa 0, se obtiene como retorno en el registro 2 del Banco 0, la codificación hexadecimal de la tecla presionada, este código se lo transfiere al acumulador.

El registro **DPTR** se lo utiliza como puntero y se lo inicializa con la primera dirección de una tabla que comprende 20 localidades de memoria, correspondientes a las 20 teclas disponibles. En cada una de estas localidades de memoria se almacena el resultado de substraer la dirección en la cual se inicia la rutina asociada con cada uno de los códigos de tecla de función definidos menos la dirección inicial de la tabla, con lo cual se logra computar el destino del salto que se deberá efectuar para alcanzar el inicio de la rutina según la función seleccionada por el usuario.

Mediante la utilización de la instrucción MOVC, que permite cargar códigos o constantes desde localidades en memoria de programa, se transfiere al acumulador la longitud del salto, correspondiente a cada función, como puede apreciarse esta longitud estará limitada a 127 localidades hacia adelante.

El salto se efectúa utilizando el acumulador como puntero y el DPTR como dirección inicial del mismo.

Debido a la limitación en la distancia desde el origen al destino del salto, se han asociado a cada etiqueta de la función seleccionada un instrucción de salto incondicional, mediante el cual se redirecciona a la zona de memoria que contiene todos los códigos de la rutina seleccionada por el usuario mediante el teclado.

Algunas etiquetas se las considera de reserva para futuras expansiones de software, por lo cual únicamente se efectúa un salto de regreso al menú de selección de funciones cuando el código de tecla seleccionado corresponde a una de estas etiquetas.

Dentro de cada rutina existe un bloque de programa para determinar en primer lugar si una tecla

ha sido presionada, esto se logra mediante el uso de la bandera **F0**, la cual para todo el programa se la ha definido como indicación de tecla presionada. Si la tecla STOP ha sido seleccionada, la ejecución de dicha rutina se suspende y el sistema regresa al programa principal, específicamente al menú de selección de funciones.

3.4.- DESCRIPCION DE LAS SUBRUTINAS.

3.4.1.- SUBRUTINA PARA ATENCION A INTERRUPCION EXTERNA 0.

Como ya se menciono esta rutina es la encargada de atender al pedido de atención originado por la interrupción externa de más alta prioridad.

Al recibir en la línea **INT0** la señal de activación adecuada se termina la ejecución de la instrucción en curso e inmediatamente se ejecuta un salto incondicional hasta la rutina de servicio de esa interrupción, almacenandose automáticamente el contenido del contador del programa en el stack, tal como puede apreciarse en la figura 3.6.

La rutina de servicio en primer lugar resguarda el contenido del **PSW** y del Acumulador, cambiando inmediatamente al banco de registros 3, con el propósito de utilizar el ultimo registro de este banco para almacenar el código correspondiente a la tecla presionada.

Según se explicó en el capitulo correspondiente el código de la tecla presionada ingresa a través de uno de los puertos paralelos creados mediante la utilización de un Circuito Integrado PPI 82C55, específicamente se utilizan los 4 bits correspondientes al puerto C superior de U7 (Capitulo II, fig. 2.3) y el bit más significativo de los 4 correspondientes a C inferior.

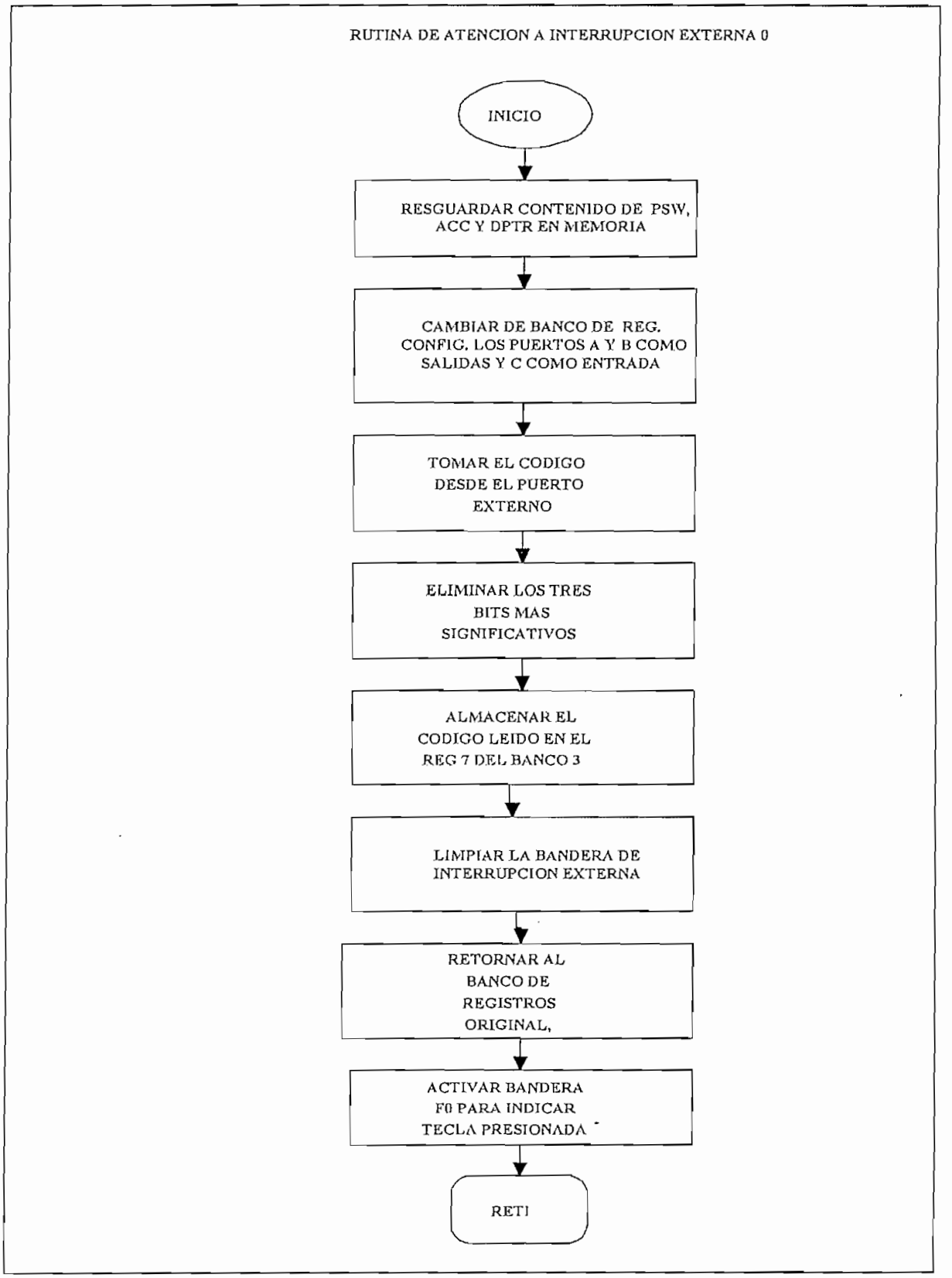


Fig. 3.6. Diagrama de flujo de rutina de atención a la interrupción EXT0

Por lo tanto antes de que la lectura del dato pueda efectuarse el microcontrolador debe configurar mediante el envío de los comandos correspondientes (Los valores en formato hexadecimal aparecen en la tabla 2.5 , Capítulos II), los puertos del C.I. 82C55 signado como U7 de modo que los puertos A y B actúen como salidas y el puerto C actué como entrada, (Capitulo II, fig. 2.3).

Entonces, mediante una instrucción que trata al puerto como una localidad de memoria externa, se toma el código de la tecla presionada y se lo carga en el acumulador.

Se debe tener presente que la codificación esta contenido únicamente en los 5 bits más significativos del acumulador, por lo tanto los tres bits menos significativos deben ser desechados para obtener el valor real del código, esto se realiza mediante la rotación hacia la derecha del acumulador.

El código hexadecimal resultante se almacena en el registro 7 del Banco 3 para preservarlo para su tratamiento posterior tal como se puede observar en el diagrama de flujo de la figura 3.6.

Antes de retornar al programa en ejecución, se borra la bandera de la interrupción para dejarla habilitada para su operación normal, se reconfiguran los puertos de U7, todos como salidas, se retorna al Banco de registros 0 y recupera los valores de PSW y el Acumulador.

3.4.2.- SUBRUTINA PARA DETERMINAR CUAL DE LAS TECLAS SE HA PRESIONADO.

Esta en realidad es una pequeña subrutina que se invoca prácticamente en todas las rutinas del presente proyecto, pues complementa a la rutina de atención a la interrupción externa 0, de modo que, en el momento que se presiona cualquiera de las teclas el código de la misma es recuperada para su tratamiento.

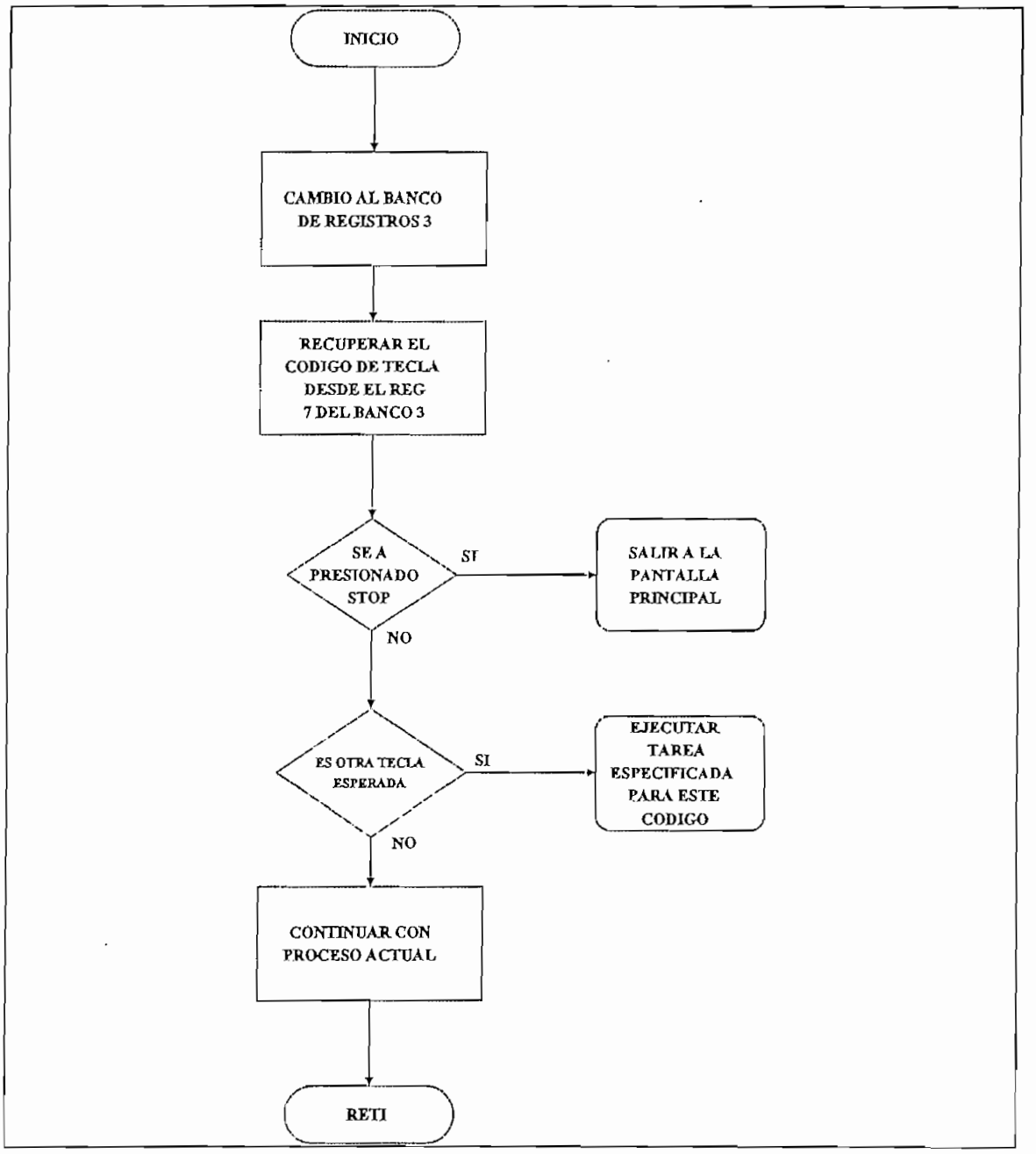


Fig. 3.7. Diagrama de flujo del bloque de comprobación de tecla presionada.

En primer lugar se procede a realizar un cambio al banco de registros 3, que es el utilizado en la rutina de atención a la interrupción; posteriormente se recupera en el Acumulador el código que la rutina de atención a la interrupción ha retornado en el registro R7, adicionalmente antes de retornar se desactiva o encera la bandera de tecla presionada, quedando lista para una nueva

recepción de códigos de tecla.

Tal como se aprecia en el diagrama de flujo que aparece en la figura 3.7, la ultima operación es regresar al banco de Registros 0. Dentro de las diferentes rutinas existe un pequeño bloque de programa que permite realizar la comprobación necesaria para asegurarse que se tome la acción adecuada si una tecla se presiona, en especial si la tecla corresponde a STOP, que se usa para salir desde las diferentes rutinas al programa principal.

Si no se trata de la tecla STOP, sino de otra con un código que es esperado por la rutina en ejecución, se toman las acciones especificas para esa rutina.

De presionarse una tecla con un código que no tiene significado en es rutina, simplemente se continua con la ejecución normal del programa.

3.4.3.- SUBRUTINA PARA DESPLEGAR MENSAJES EN EL DISPLAY.

Esta rutina se encarga de desplegar los resultados o los mensajes indicativos de las funciones del equipo en la pantalla conformada por el arreglo de las matrices de LEDS, esto implica tomar los códigos almacenados en RAM interna correspondientes a los caracteres que deben aparecer en el display y desplegarlos en cada una de las matrices en su posición adecuada.

Este control es más bien desde el punto de vista de posicionamiento específico de los caracteres, mientras que el control de la salida a través de los puertos y el barrido de las columnas se lo realiza mediante el llamado a una subrutina llamada OUTMX, que se explicará en la siguiente sección.

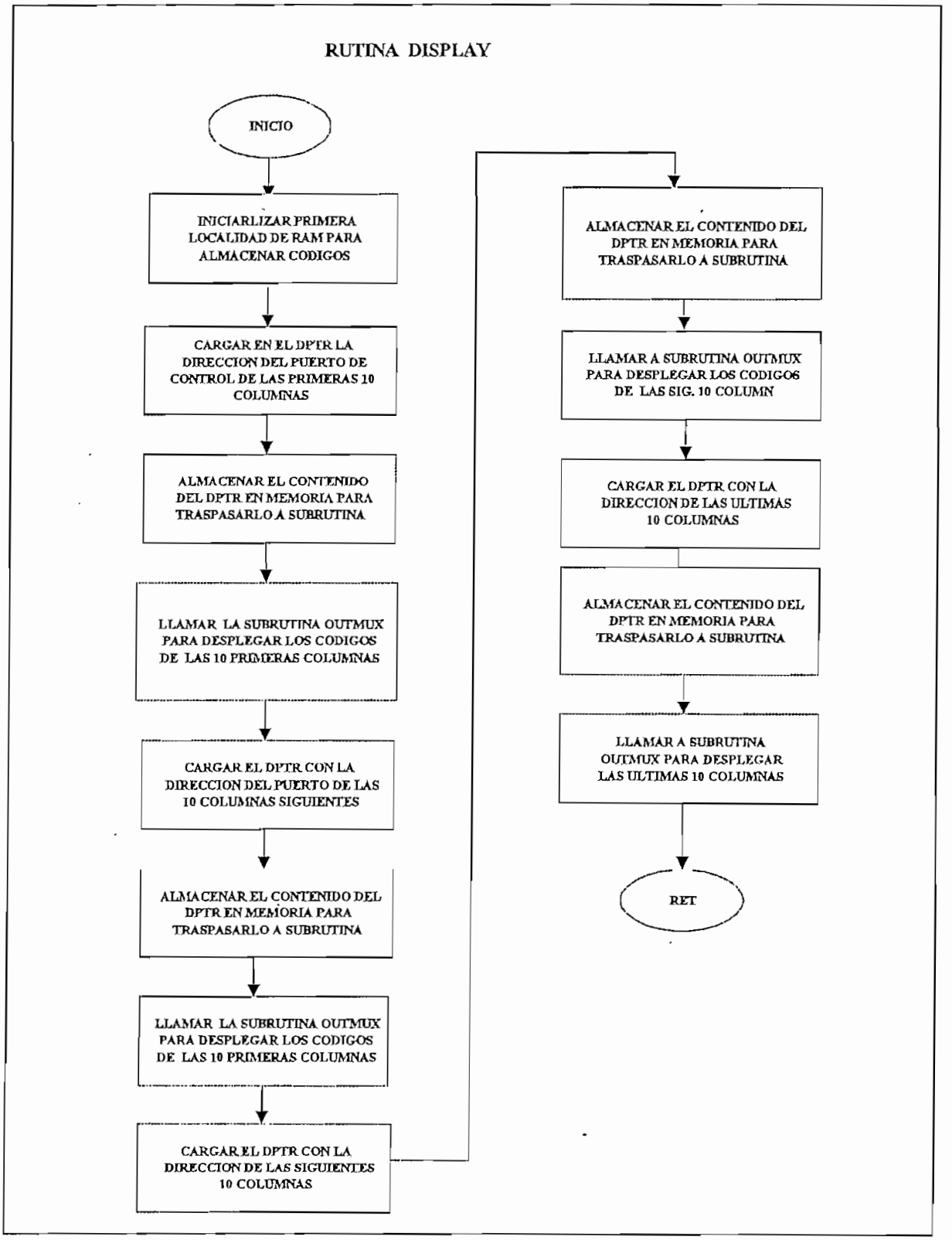


Fig. 3.8. Diagrama de flujo de rutina para desplegar resultados en display.

La rutina Display utiliza el DPTR como puntero debido a que deberán ser direccionados los puertos de salida que desde el punto de vista de el microcontrolador corresponden a localidades de memoria externa.

Los dígitos se manejan en grupos de dos, esto implica un total de 10 códigos correspondientes a diez columnas o lo que es igual 2 matrices, que como ya se explico en capitulo II es el número de columnas que se barren en base al hardware implementado.

Por lo tanto la rutina realiza un proceso que se repite en cuatro ocasiones para desplegar el mensaje total en el display, tal como se aprecia en el diagrama de flujo de la rutina DISPLAY.

3.4.4.- SUBRUTINA DE CONTROL DEL VISUALIZADOR DE MATRICES.

El microcontrolador maneja las 8 matrices de LEDs de 7x5 que conforman el visualizador del equipo a través de los puerto de un C.I. 82C55, de modo que para la formación de cada caracter se envía en forma secuencial palabras de 7 bits, cada uno de los cuales corresponde a cada uno de los 7 leds de una columna de la matriz, luego el byte con la combinación adecuada de 5 bits se encarga de activar a esa columna en particular dando como resultado un barrido por columnas que muestra al caracter como fijo en la matriz.

Los códigos para la presentación de los caracteres en las matrices se los obtiene de las tablas correspondientes que han sido almacenadas en la memoria de programa (EPROM). Pero debido a que se deberá mostrar en el arreglo de matrices hasta un total de 8 caracteres en una combinación diferente según se requiera, se procede a organizar y utilizar la memoria de datos interna del microcontrolador en la forma que aparece en la Figura 3.9.

7FH 58H	STACK DE MEMORIA RAM INTERNA DISPONIBLE DEL MICROCONTROLADOR 8051
53H	DIGITO 8
4EH	DIGITO 7
49H	DIGITO 6
44H	DIGITO 5
3FH	DIGITO 4
3AH	DIGITO 3
35H	DIGITO 2
30H	DIGITO 1
2FH 18H	ZONA DE MEMORIA ACCESIBLE A NIVEL DE BITS.
18H	BANCO 3
10H	BANCO 2
08H	BANCO 1
00H	BANCO 0

Fig. 3.9. Distribución de la memoria RAM interna, con espacio separado para manejo del Display.

Es decir, el espacio de memoria comprendido entre 30H y 57H inclusive, se lo reserva para almacenamiento de códigos para la generación y presentación de los caracteres en cada una de las 8 matrices que conforman el visualizador del equipo.

Cada vez que un mensaje de presentación ó un resultado deba aparecer en la pantalla esta rutina de codificación se encargan de almacenar en la localidad de memoria asignada a cada dígito, los códigos obtenidos desde memoria de programa, de modo que se forma en memoria RAM interna la palabra que debe aparecer en la pantalla.

En el diagrama de flujo se observa la rutina correspondiente:

Para esta rutina se utiliza el registro R1 como puntero, por lo tanto en primer lugar se lo inicializa con la dirección de inicio de la zona de RAM apartada para los códigos de los caracteres.

Los códigos correspondientes a cada una de las columnas utilizan como interface paralela, los puertos de los C.I. 82C55A signados como U7 y U8, que corresponden a las localidades de memoria externa rotuladas como PBU7, PAU8, PBU8 y PCU8, cada uno de los cuales maneja un subarreglo conformado por dos matrices, mientras que las señales para la activación de cada una de las 10 columnas de este subarreglo utilizan como interface el Puerto correspondiente a la dirección de memoria externa rotulada como PAU7 y las dos líneas menos significativas de PCU7.

Entonces el software debe controlar cuatro arreglos de 7 filas y 10 columnas

El DPTR es el puntero para las direcciones de 16 bits por lo tanto se lo inicializa con la dirección correspondiente a la matriz que esta en el extremo derecho del arreglo, mediante una subrutina se almacena esta dirección para su posterior recuperación, se invoca entonces a la subrutina encargada de sacar lo códigos de las columnas y el barrido de las columnas.

Esta subrutina llamada **OUTMX**, inicializa R2 como contador para el barrido de las 8 primeras columnas del arreglo, R3 contendrá el valor inicial del patrón de barrido para activación de las

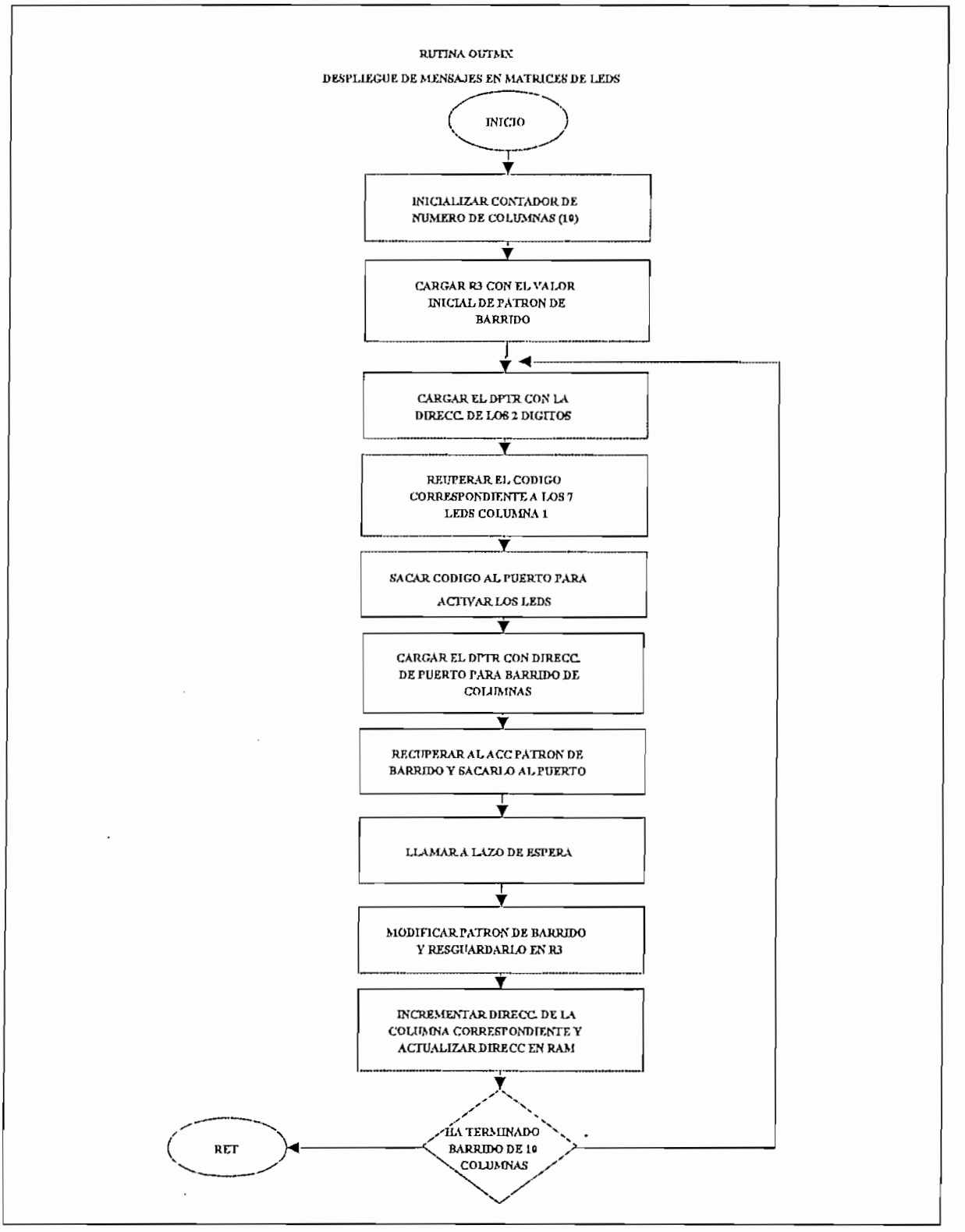


Fig. 3.10. Diagrama de flujo para barrido de matrices.

columnas.

Entonces se empieza un lazo de repetición que se inicia con la recuperación de la dirección de 16 bits y su almacenamiento en DPTR, esto resulta necesario puesto que dentro del mismo lazo el DPTR se lo utiliza tanto para la dirección correspondiente a los 7 bits para control de los LEDs de la columna, como para la dirección a través de la cual sale el patrón de activación de las columnas.

Se utiliza entonces el puntero R1 para recuperar desde RAM interna del microcontrolador, el código de control de los 7 leds de la columna y se lo envía a la dirección de memoria externa contenida en DPTR, utilizando como almacenamiento temporal al acumulador.

Inmediatamente se carga en DPTR la dirección correspondiente al puerto de salida del patrón de barrido de las columnas, se recupera el patrón de modo que la columna que se activa coincida o se sincronize con aquella en la que esta el código de control de los Leds y se lo saca a través del puerto PAU7 del CI 8255.

Se requiere que la columna permanezca activada un cierto período de tiempo de modo que el usuario no llegue a notar que se esta utilizando un proceso de barrido, sino que el caracter aparezca como fijo.Para eso se utiliza un lazo de espera por software..

El proceso se repite por 8 ocasiones, esto es el número de líneas de salida del puerto paralelo del C.I. 82C55 denominado PAU7, sin embargo son 10 las columnas que deben ser barridas, por lo tanto y debido a que se han utilizado las dos líneas menos significativas de PCU7,antes de continuar con el barrido se debe proceder a cargar en el puntero DPTR, la nueva dirección del puerto de salida y se debe reinicializar el contenido del registro que proporciona el patrón de barrido de columnas

3.4.5.- SUBRUTINA PARA ALMACENAR EN RAM CODIFICACION PARA MANEJO DE MATRICES.

Son en realidad dos las rutinas encargadas de tomar los códigos correspondientes a cada una de las columnas del arreglo desde memoria de programa y transferirlas hasta la zona de RAM interna reservada para tal propósito, sin embargo las dos se diferencia únicamente en el valor de los contadores, esto es una transferirá los 40 códigos de una sola vez, mientras que la otra solo llena el espacio correspondiente a una matriz o lo que es lo mismo 5 códigos.

Se utiliza el registro R1 como puntero, en el cual se carga la dirección inicial de la zona de RAM destinada al almacenamiento de los códigos, mientras que R2 contendrá el número de códigos a ser transferidos desde la memoria de programa a la RAM interna esto es 40 ó 5, se inicializa además el registro R3 con 00h como contador.

Previa a la llamada a esta subrutina se carga el DPTR con la dirección inicial del segmento de memoria de programa en el cual esta contenida la codificación correspondiente a los caracteres o resultados que deben ser almacenados en RAM interna para posteriormente ser desplegados en pantalla.

El acumulador en este caso actúa como puntero de la instrucción MOVC y se carga previa la ejecución de esta con el valor que se ha preservado en el registro R3, una vez ejecutada la instrucción, el acumulador contiene el valor del código obtenido desde memoria de programa y mediante direccionamiento indirecto usando el registro R1 se lo transfiere a la localidad de RAM interna correspondiente.

Se apunta entonces mediante R1 a la siguiente localidad de RAM interna, se incrementa además el valor del puntero preservado en el registro R3 y se decrementa el contador de el número de códigos, esto es el registro R2, que determina cuantas veces se repite el proceso anterior, según se deban tomar 5 códigos correspondientes a un solo caracter o lo que es lo mismo una sola matriz ó 40 códigos de las 40 columnas de las 8 matrices o caracteres.

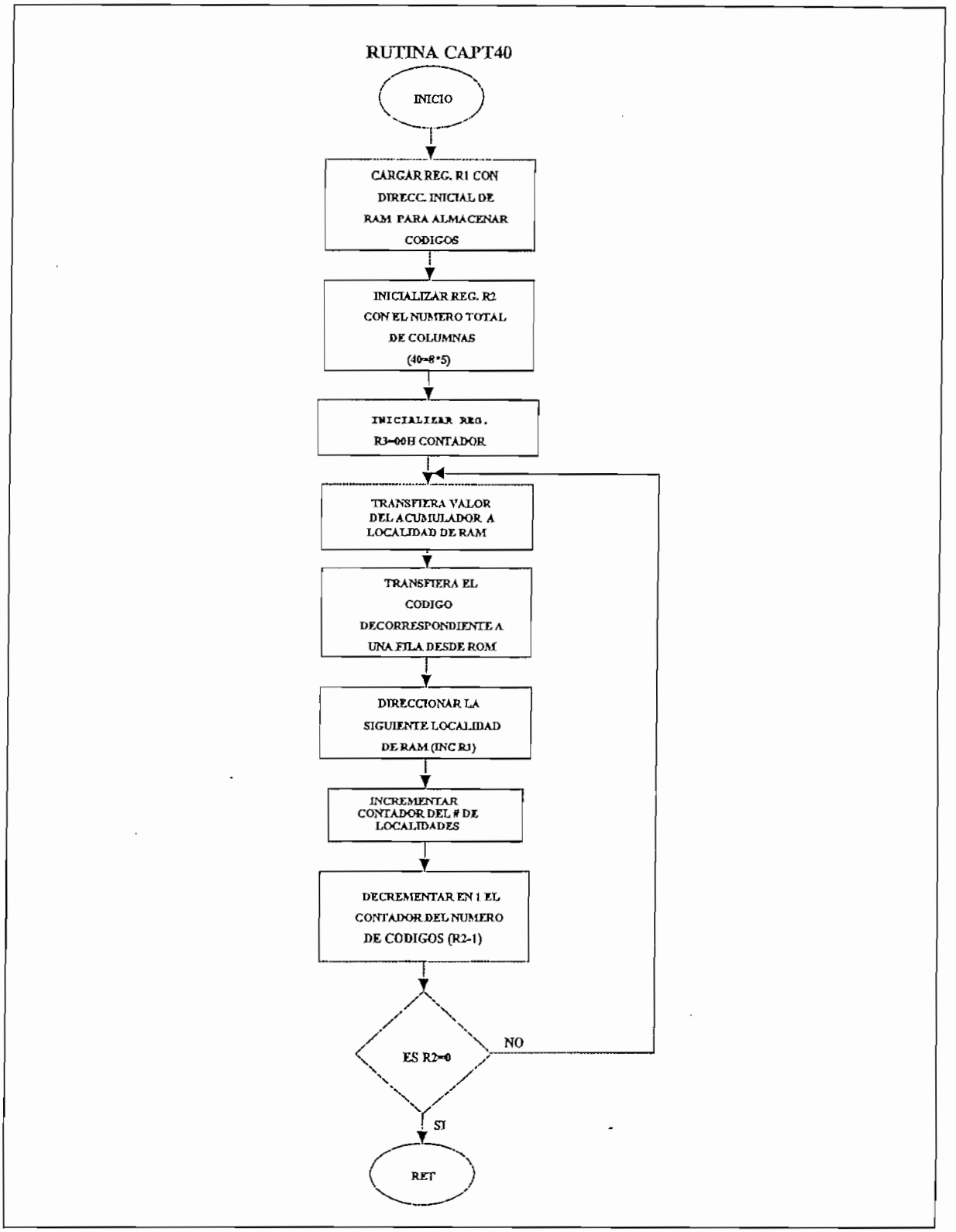


Fig. 3.11. Transferencia a RAM de la codificación de Caracteres.

Todo este proceso se lo aprecia claramente en el diagrama de flujo CAPT40 de la figura 3.11

3.4.6.- SUBRUTINA COD_MESJ.

La subrutina anterior parte de la base de tener una tabla codificada del mensaje completo, esto es se dispondria de los 40 codigos necesarios para la activación de cada uno de los leds de las columnas de las matrices, definidas en un formato como el siguiente:

```
M_BCD:   DB   3EH,51H,49H,45H,3EH
         DB   3EH,51H,49H,45H,3EH
         DB   3EH,51H,49H,45H,3EH
         DB   3EH,51H,49H,45H,3EH
         DB   3EH,51H,49H,45H,3EH
         DB   3EH,51H,49H,45H,3EH
         DB   22H,41H,41H,41H,3EH
         DB   36H,49H,49H,49H,7FH
```

Existe sin embargo otra forma de codificar los mensajes la cual consiste en obtenr estos codigos a partir de los caracteres ASCII del mensaje, sin embargo esta forma necesita de un subrutina complementaria que se encargue de la obtención de esta codificación, la cual se ha denominado ALMSCII y se detalla más adelante.

El formato de especificación del mensaje en ese caso se lo programa como sigue:

```
M_OPP1:   DB   " NOICPO"
```

Como puede notarse los caracteres van de derecha a izquierda, esto es simplemente por la forma en que se programo el almacenamiento en memoria Ram interna de los códigos para activación de las matrices.

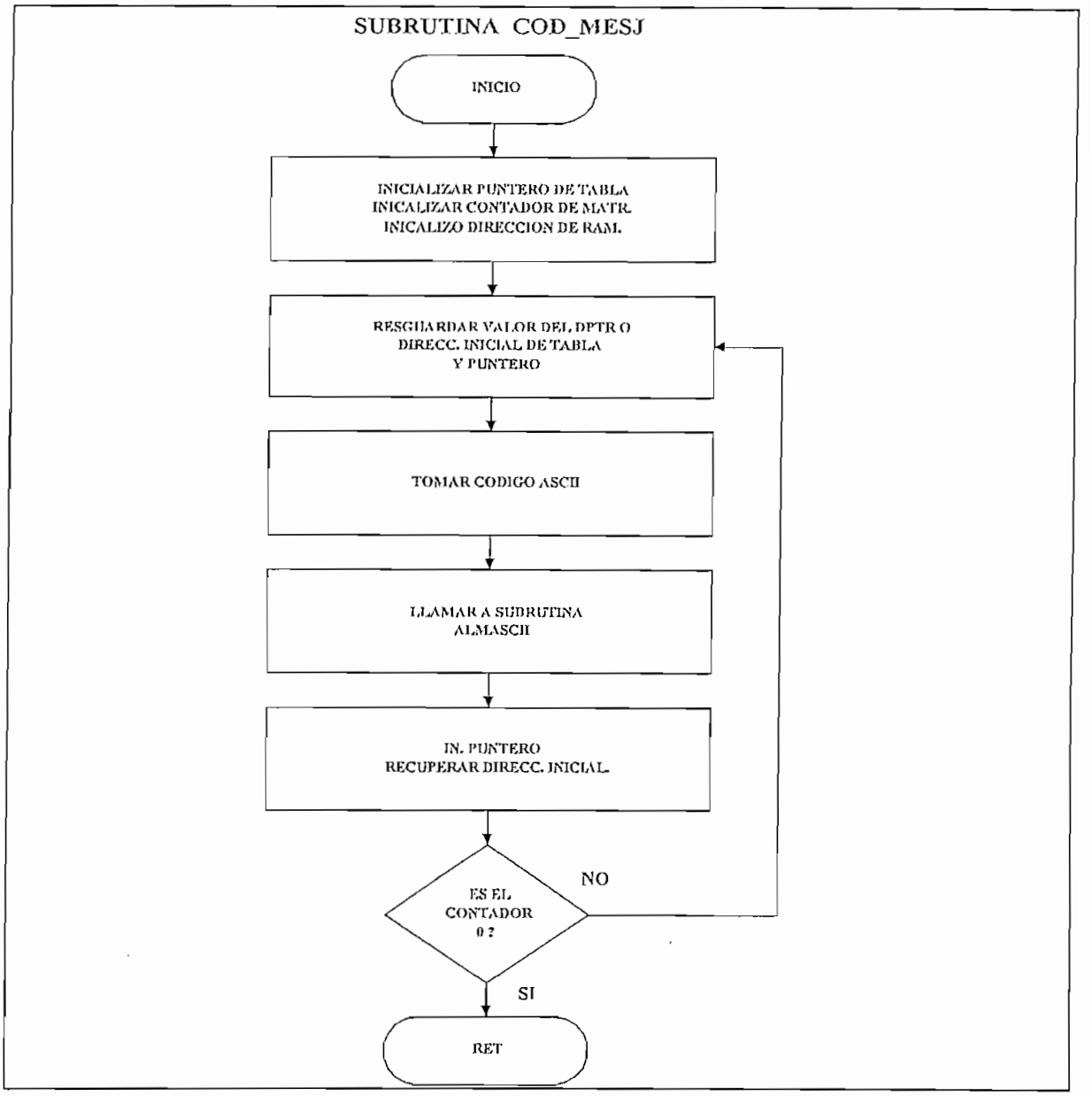


Fig. 3.12. Diagrama de flujo para codificación de caracteres ASCII.

En cuanto a la subrutina actual, y de acuerdo al diagrama de flujo de la figura 3.12, el primer paso consiste en inicalizar el puntero de la tabla, contador de matrices, dirección inical de RAM para almacenamiento de códigos; se resguarda entonces la el contenido del DPTR que corresponde a la dirección inicial de la tabla.

Una vez tomado el caracter ASCII del mensaje se llama a la subruitna ALMSCII para obtener los codigos adecuados para el manejo de las matrices, y se repite este proceso mientras el

contador del número de matrices llega a su valor 0.
Simultaneamente con la obtención del código se realiza su almacenamiento en la locaridad de RAM interna que le corresponde, para su posterior despliegue.

Como se menciono se hace uso de la subrutina ALMASCII para la codificación de los caracteres ASCII, pero además a lo largo del programa se deben codificar además lo valores hexadecimales, para lo cual se usa un subrutina exactamente igual a la de los caracteres ASCII, por lo tanto y dado que son muy similares explicaremos en primer lugar aquella encargada de los valores hexadecimales a continuación.

3.4.7.- SUBRUTINA ALMRAM

Esta subrutina permite obtener la codificación correspondiente a cada uno de los números hexadecimales, necesaria para que en el arreglo de matrices aparezca la representación correcta de dicho número, por ejemplo si el número hexadecimal contenido en un nibble (4bits) es AH, los códigos correspondientes a los 7 leds de cada una de las 5 columnas de aquella matriz en la que deba aparecer representado este dígito deben ser adecuadamente almacenados en las localidades de RAM interna apartadas para tal efecto.

Para obtener esta codificación se utiliza un procedimiento similar al que utiliza el programa principal a fin de dilucidar cual de las funciones debe ser ejecutada, en base al código de la tecla presionada, esto es mediante una tabla definida en la memoria de programa, en esta tabla se han definido 15 localidades de memoria correspondientes a cada uno de los dígitos hexadecimales y cada una contiene el resultado de restar la dirección de la etiqueta asociada a cada dígito menos la dirección inicial de la tabla, o lo que es lo mismo el resultado indica la longitud del salto hasta alcanzar esa etiqueta en particular.

Utilizando como índice el valor hexadecimal del cual se quiere obtener la codificación, transferido hacia el acumulador y como puntero el DPTR inicializado con la dirección de la primera localidad de memoria ocupada por la tabla, se utiliza la instrucción MOVC para obtener

como resultado en el acumulador la longitud del salto necesario para alcanzar la etiqueta mediante un salto cuyo origen es la primera dirección de la tabla. Como se observa en la figura 3.12.

A cada una de las etiquetas correspondientes a los dígitos hexadecimales se encuentran asociadas dos líneas de instrucciones, la primera inicializa el DPTR con la dirección inicial de las 5 localidades de memoria que contienen los códigos de cada uno de los 15 dígitos hexadecimales, y la segunda es una llamada a la subrutina Llenar, que permite transferir estos códigos a la localidad de memoria interna correspondiente.

3.4.8.- SUBRUTINA ALMASCII.

Esta subrutina es del todo similar a la anterior, con la salvedad que permite obtener la codificación de las letras del alfabeto y no los números hexadecimales, sin embargo se basa exactamente en la misma utilización de una tabla en la cual se han definido los códigos correspondientes a cada caracter y su selección basada en saltos con índices calculados en base a la instrucción MOVC y como índice el valor a codificar. Se incluyen caracteres especiales como el blanco.

Este proceso se aprecia en el diagrama de flujo de la figura 3.11, que para todo efecto es similar a la rutina anterior y por este motivo solo se incluye uno de los dos.

3.4.9.- SUBRUTINA PARA TRANSFORMAR DE BINARIO A HEXADECIMAL.

Cuando esta función ha sido seleccionada, el prototipo debe leer el dato binario de hasta 16 dígitos que este en ese momento presente en el puerto paralelo implementado mediante la utilización de un CI 82C55, y transformarlo a formato Hexadecimal, para desplegar este resultado en la pantalla del equipo.

Los microcontroladores de la familia 8051 de INTEL, manejan los datos tanto en forma binaria como Hexadecimal, esto implica que el hecho de transforma un dato leído desde el puerto de

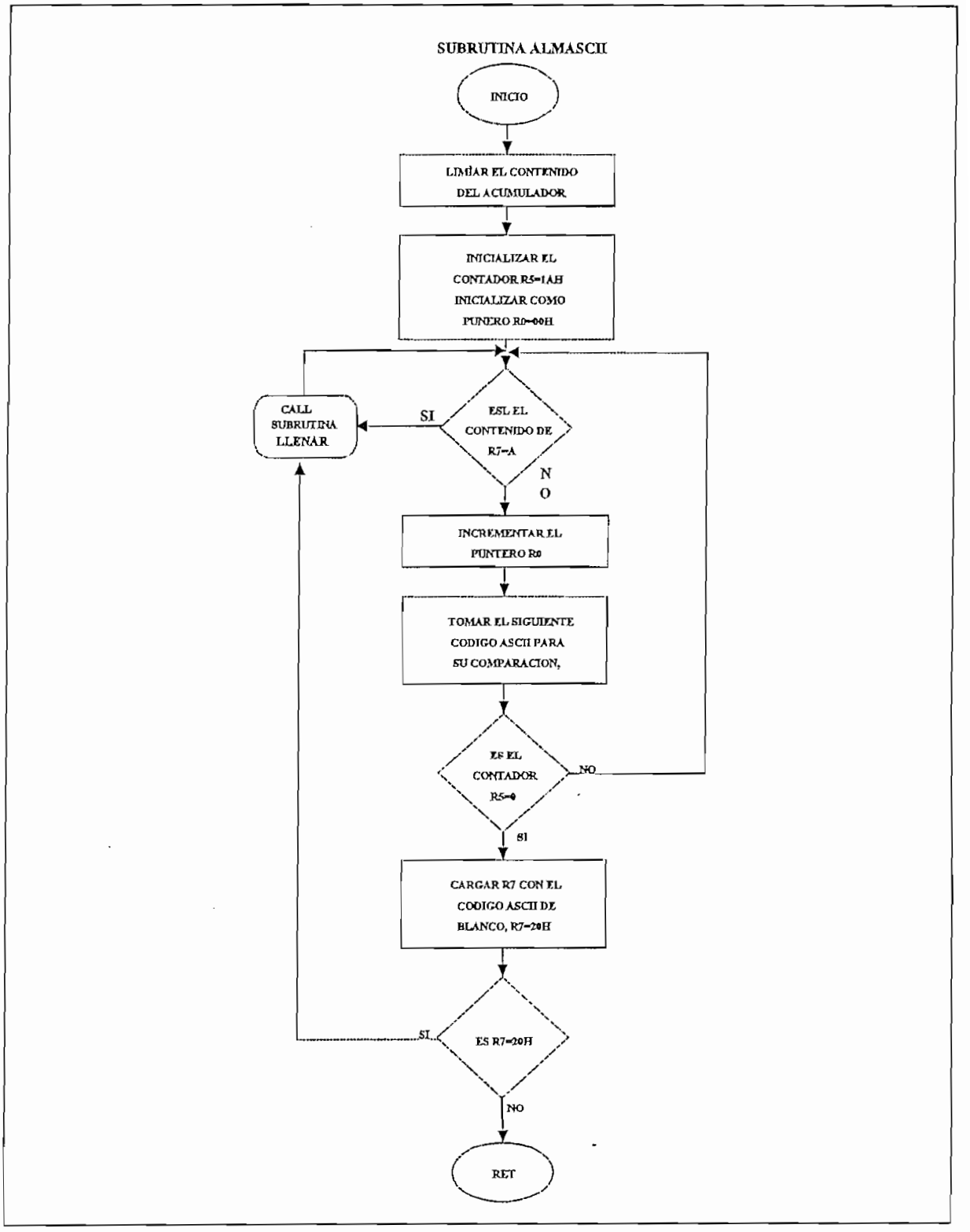


Fig. 3. 13. Diagrama de flujo para codificación ASCII.

entrada en forma Binaria sea una labor relativamente sencilla.

Básicamente solo se realiza la determinación de la codificación correspondiente a cada dígito para que en la pantalla de matrices aparezca el caracter adecuado.

Cuando se ha seleccionado esta opción desde el programa principal, en primer lugar se despliega el mensaje de presentación correspondiente, inmediatamente se verifica cual de las teclas se ha presionado para decidir si se continua con la ejecución de la rutina o se retorna al menú principalLos datos se los va tomando en grupos de cuatro bits con la finalidad obtener los códigos adecuados desde la memoria de programa y almacenarlos en RAM, esta operación se repite hasta obtener el valor de los 4 dígitos posibles.

Una vez que se ha obtenido los códigos correspondientes, se procede a realizar una llamada a la rutina que permite desplegar este resultado en el arreglo de matrices.

El siguiente paso es verificar si se ha presionado alguna tecla mientras se efectuaba el proceso antes descrito, de no haberse producido tal evento se repite el proceso para el siguiente grupo de bits que se toma desde el puerto paralelo.

Si por el contrario se ha presionado una tecla, se debe determinar el código de la misma en función de este se decide si retorna al programa principal o se repite otra vez la captura de un nuevo dato y el despliegue de resultados.

En cierto sentido mediante la selección de una tecla mientras se efectúa la transformación y despliegue de resultados se puede lograr una ejecución paso a paso, aunque los datos que se capturan a través del puerto paralelo se lo realice en forma aleatoria.

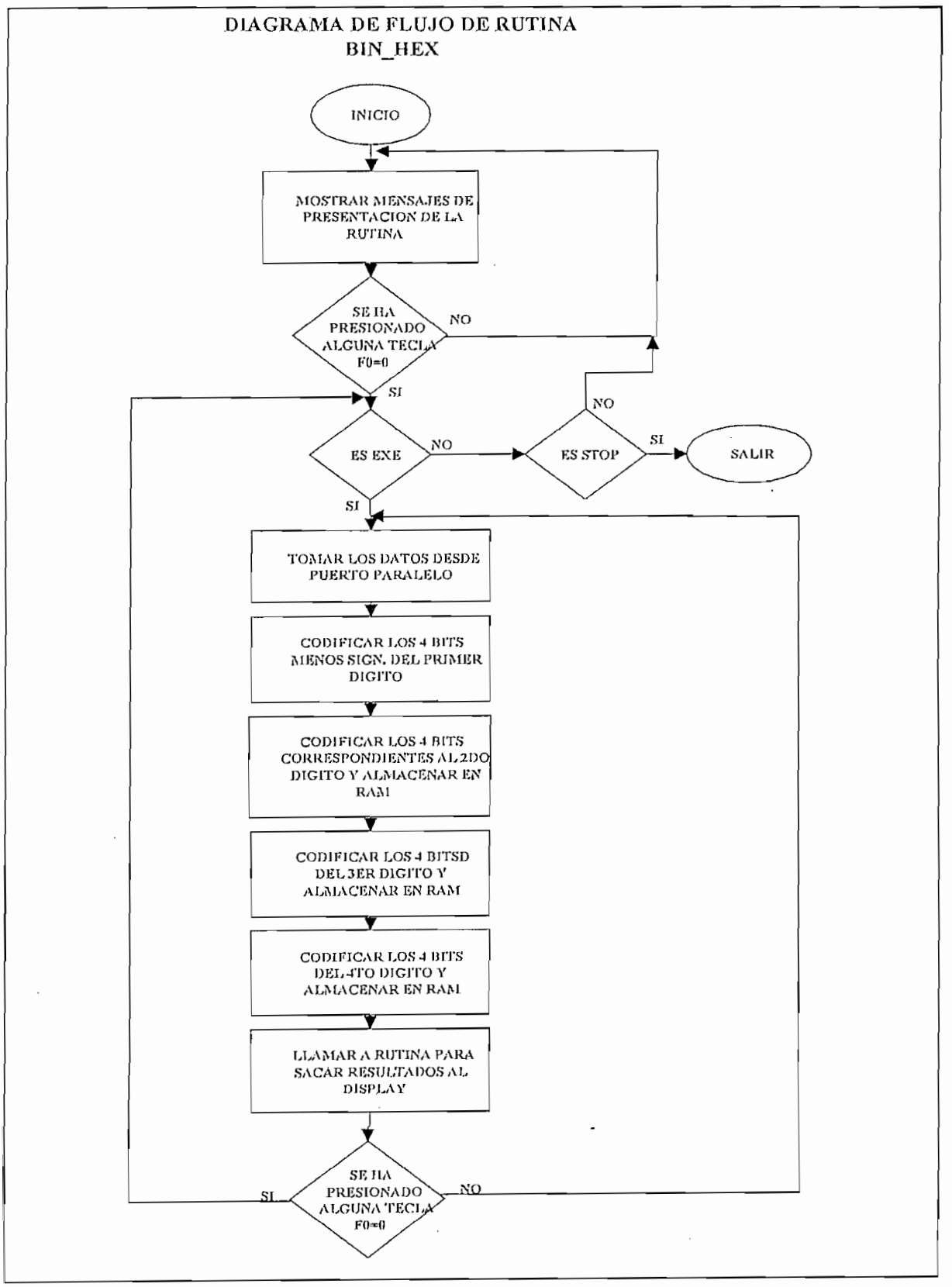


Fig. 3. 14. Diagrama de flujo de codificación Hexadecimal.

3.4.10.-SUBRUTINA PARA TRANSFORMAR DE BINARIO A BCD.

Basados en la consideración de que las entradas digitales pueden ser de 16 bits, la rutina debe estar en capacidad de transformar numeros de dos bytes a su respectiva representación BCD.

El numero más alto que puede representarse mediante 16 bits en representación BCD es 65535 que en hexadecimal corresponde a FFFFH.

Una rutina muy conocida para realizar la transformación de hexadecimal a BCD es aquella en la que se realizan divisiones sucesivas del número que se quiere transformar para determinar el número de centenas decenas y unidades.

Sin embargo al tratarse de una transformación de números de 16 bits el utilizar este proceso de divisiones resulta poco practico, por lo tanto se utilizará restas sucesivas para ir determinando el número de decenas de mil, unidades de mil, centenas, decenas y unidades que contiene el número representado en hexadecimal hasta obtener su representación en BCD.

Inicialmentes se toma los valores leidos desde los puertos paralelos de acceso y se los almacena en el rgistro R6 el byte menos significativo y el el registro R7 el byte más significativo, se utiliza los registros R5 y R4 para almacenar el valor de 10.000 (2710 en hexadecimal) que nos permitira determinar por restas sucesivas el número de veces que esta cantidad esta contenida en el dato que se debe trasnformar, en este caso R4 contiene el byte menos significativo y R5 el más significativo.

Para realizar la resta sucesiva y dado que se trata de un proceso que se repetira para la determinación del número de decenas de mil, unidades de mil, etc, se utiliza una subrutina denominada DIV_SUB, sobre la cual tratremos más tarde.

El resultado de determinar el número de decenas de mil que el dato binario contine retona desde

la subrutina mencionada en el registro B, y este valor en hexadecimal se lo almacena directamente en la localidadd de RAM interna destinada al digito denominado DIGI4 (la numeración se inicia con DIGI0), mediante la rutina ALMRAM; al mismo tiempo la subrutina DIV_SUB devuelve el valor del residuo de la resta en los registros R6 y R7.

Para determinar el número de unidades de mil se procede de la misma manera, siendo el primer paso cargar los registros R4 y R5 con el valor 1.000 (03E8H), y se almacena el resultado obtenido en la localidad de RAM interna correspondiente a las unidades de mil.

Nuevamente para el caso de las centenas se inicaliza los registros con el valor de 100 para utilizar la subrutina encargada de realizar las restas sucesivas hasta determinar el número de veces que el valor 100 se repite en el dato bajo analisis, este resultado se o almacena en la localidad de RAM que corresponde a las centenas o lo que da lo mismo aquel cuyo posición de memoria interna se lo ha rotulado como DIGI2.

El proceso se repite hasta determinar el número completo y entonces se despliega en pantalla el resultado tomando la codificación desde las localidades de RAM y usando la rutina DSPLAY para desplegarlas en pantalla.

El formato en que se despliga el número obetnido permite mostrar hasta 5 digitos con lo cual se puede representar el mayor número esto es 65535.

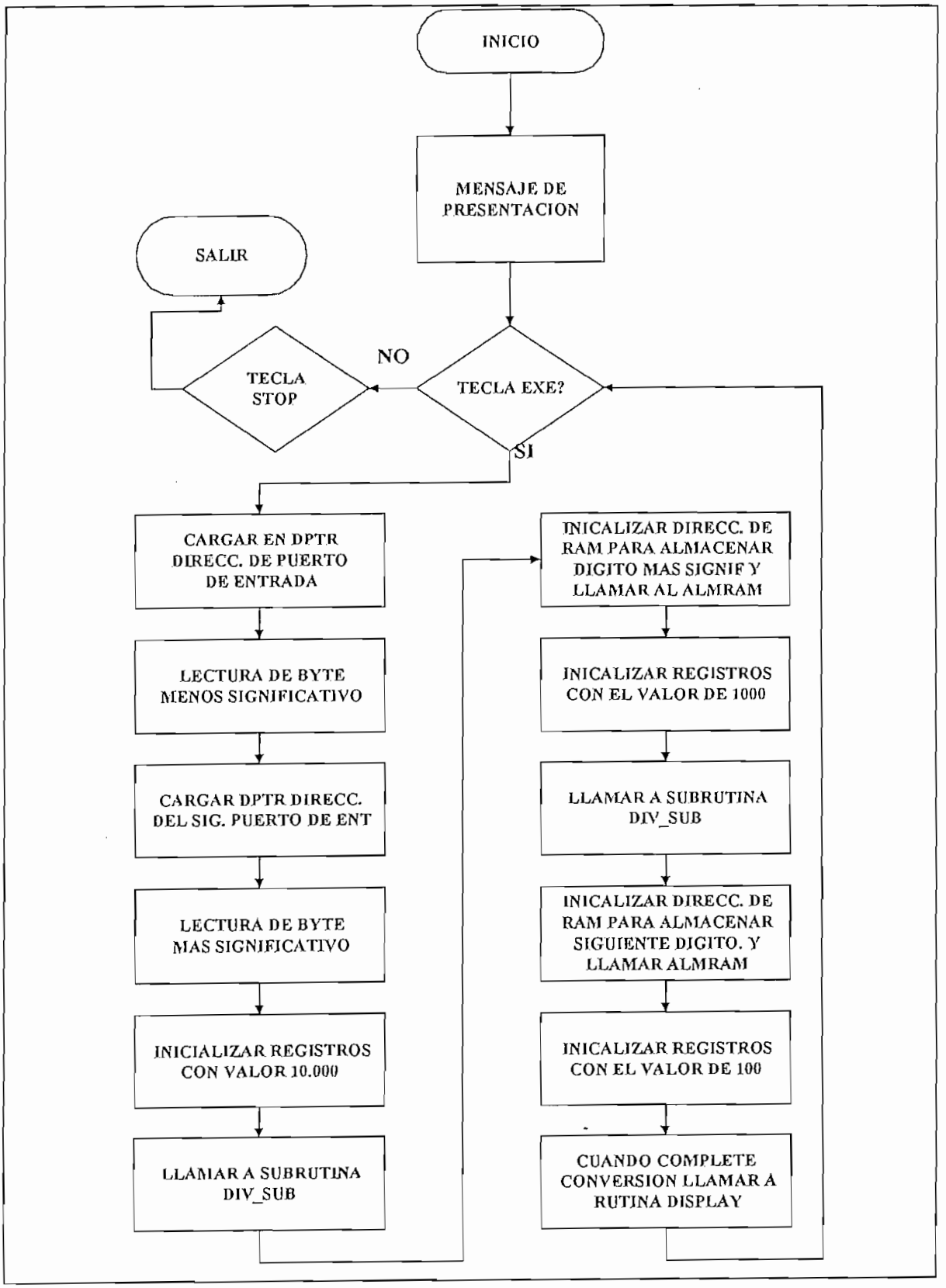


Fig. 3.15. Diagrama de flujo de transformacion de Binario a BCD

3.4.11.- SUBRUTINA DIV_SUB.

Esta subrutina complemeta a la anterior para la transformación de numeros en representación binaria a su representación en BCD.

El primer paso consiste en inicalizar el contador de número de repeticiones con un valor de 0, específicamente se utiliza el registro **B** para este propósito.

Se realiza entonces la operación de resta entre el registro R6 menos el registro R4, los que continen los bytes menos significativos del valor constante y del número bajo trasnformación e inmediatamente se restan entre si los bytes más significativos.

Se incrementa el contador en 1 y se verifica si se ha generado un carry como consecuencia de la operación de resta, de no estar presente la bandera de carry se repite la operación de resta con los mismos registros; por el contrario si se detecta la presencia de carry significa que el número contenido es menor al que se esta usando como constante por lo que se corrige el valor de conteo mediante una operación de decremento del registro contador (B), se recupera entonces el valor del residuo de la operación mediante operacion de suma con carry entre los registros R4, R5, R6 y R7 respectivammente.

Como dato de retorno de esta subrutina se tiene en el registro B el número de veces que se repite la constante en el dato analizado y el valor del residuo de la resta en los registros R6 y R7.

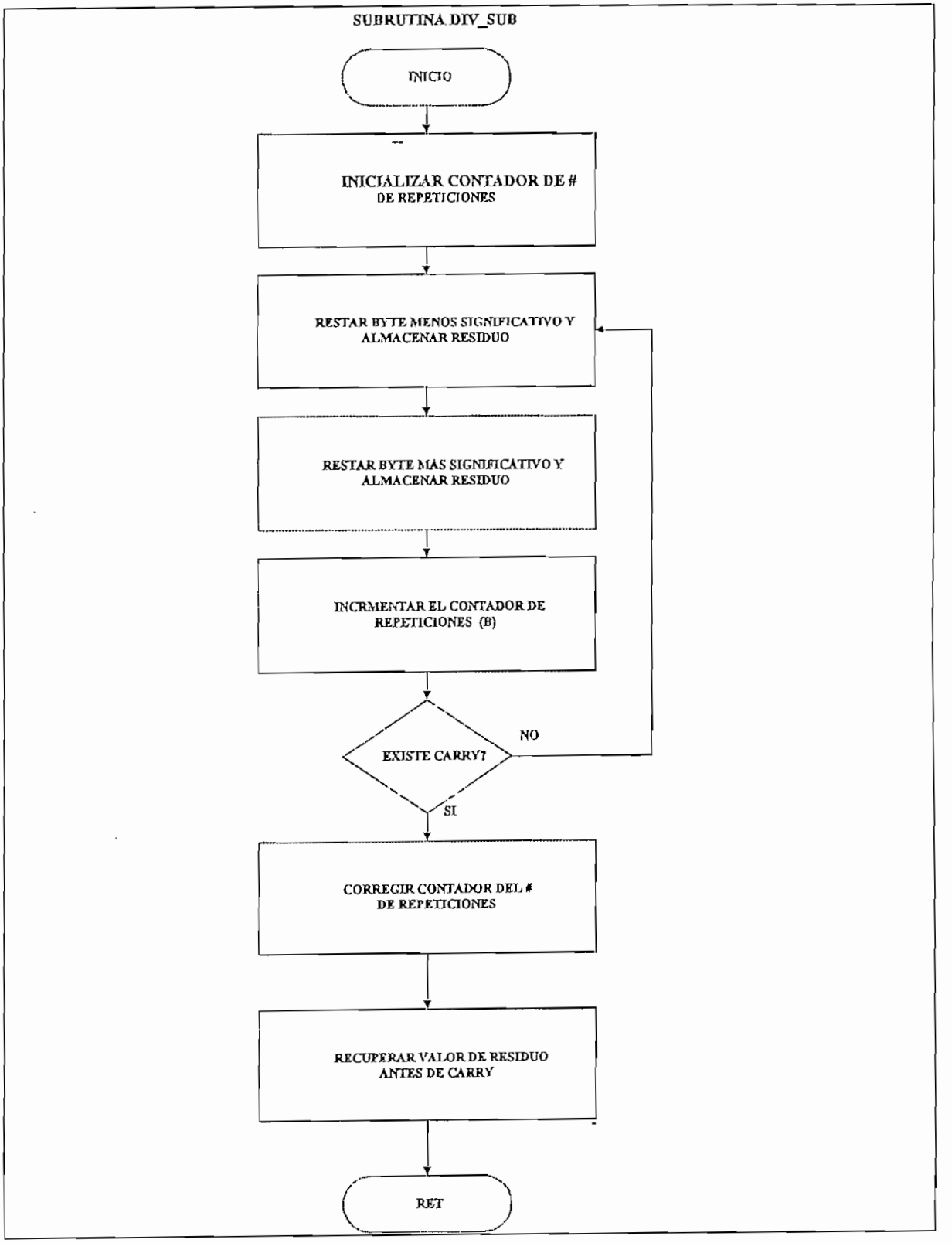


Fig. 3. 16. Diagrama de flujo de rutina DIV_SUB

3.4.12.- SUBRUTINA PARA TRANSFORMAR DE ASCII A BCD.

La codificación ASCII ó *American Standar Code for Information Interchange* es como su nombre lo indica un sistema de codificación mediante la asignación de códigos numéricos a las letras y símbolos; por lo tanto para realizar la transformación desde la representación en dicho código a la representación hexadecimal se debe establecer claramente cual es la forma de asignación numérica utilizada.

De la tabla de codificación ASCII que se adjunta en los anexos puede apreciarse que los códigos correspondientes a los números comprendidos entre 0 y 9 equivalen a la codificación numérica

entre 30H y 39H respectivamente, en cambio que los valores entre "A" y "F" están representados por 41H hasta 46H respectivamente.

Basados en esta forma de codificación, para realizar la transformación de ASCII a hexadecimal se ha procedido de la siguiente forma:

Se almacena el dato correspondiente al código ASCII en uno de los registros y en el Acumulador; de este último se eliminan los cuatro bits menos significativos, con lo cual el nible más significativo solo puede tener dos valores, esto es puede ser 30H, si el código corresponde a uno de los valores entre 0 y 9 (30H...·39H) ó 40H si el código corresponde a un dato entre "A" y "F".

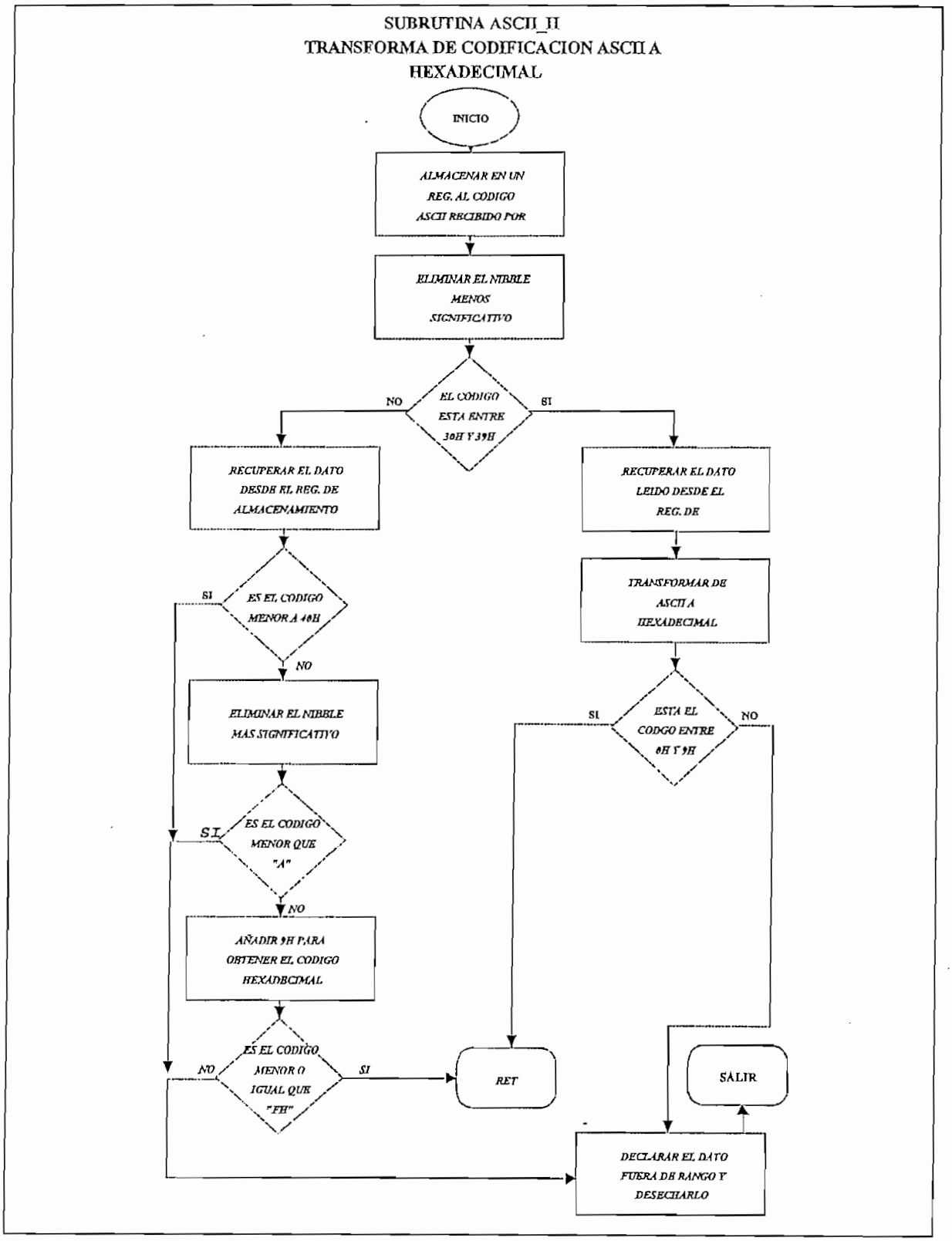


Fig 3.17. Diagrama de flujo para la transformación de ASCII a Hexadecimal

La verificación en la presente rutina se realiza en primer lugar para la codificación correspondiente a valores entre 0 y 9, mediante una comparación con el valor de referencia 30H, de ser este el caso, se recupera el valor original desde el registro donde se lo resguardo al inicio del proceso y se determina el valor en hexadecimal simplemente eliminando los cuatro bits más significativos.

Por otra parte si se trata de un valor comprendido dentro del rango de 40H, se debe comprobar que se encuentre en el grupo correspondiente a las codificaciones para "A" hasta "F", pues de lo contrario, se tratará de un valor ilegal, en cuyo caso el equipo debe sacar un menaje acorde.

Para comprobar que no se trata de un valor mayor a la representación de "F" se eliminan los 4 bits más significativos del byte, una vez que se estamos seguros que este valor es 40H; restando 06H del valor sobrante se verifica si es mayor, pues se activará la bandera del carry, indicando que el valor es mayor a 46H y por tanto ilegal dentro de nuestro rango.

Si por el contrario el dato es correcto, basta con sumarle 9H para obtener la representación hexadecimal pertinente.

Se realiza una verificación adicional para asegurarnos que el valor no es inferior a 0AH, para lo cual se resta el resultado obtenido menos el valor de comprobación (0AH); y en forma similar al caso anterior, si la bandera del carry se activara el dato resultaría ilegal.

3.4.13.- SUBRUTINA PARA RECEPCION DE DATOS TRANSFERIDOS DESDE UN PUERTO SERIAL.

Con la finalidad de establecer la comunicación serial entre el dispositivo y un computador personal se utiliza el puerto full-duplex que para tal efecto ha sido incorporado en el diseño del microprocesador.

La característica Full duplex del puerto serial implica que los datos pueden ser transmitidos y recibidos en forma simultanea. Los datos son recibidos mediante un buffer lo cual significa que se puede iniciar la recepción de un segundo byte aun antes de que el primer byte haya sido leído desde el registro de recepción, sin embargo si la lectura del primer byte no se ha concluido antes del tiempo necesario para completar la segunda recepción, uno de los dos bytes puede perderse de ahí la importancia de configurar adecuadamente la velocidad de transferencia de los datos.

Según el manual del fabricante del Microcontrolador el puerto serial del mismo puede operar en uno de los siguientes modos:

MODO 0.- Los datos seriales son transferidos a una tasa fija de 1/12 de la frecuencia del oscilador. Con un formato de 8 bits, iniciando por el menos significativo.

MODO 1.- En este modo el formato utilizado consta de 10 bits distribuidos en: un bit de inicio, 8 bits de datos, en primer lugar el menos significativo y 1 bit de parada. La tasa de transmisión es variable.

MODO 2.- En este caso 11 bits son transmitidos a través de TX o recibidos a través de RX; un bit de parada (O), 8 bits de datos, iniciando por el menos significativo, el noveno bit es programable y el último es el bit de parada (1), el bit número 9 puede ser utilizado para control de paridad. La tasa de transferencia puede ser programada a 1/32 o 1/64 de la frecuencia del oscilador.

MODO 3.- En este modo el número y utilización de los bits es exactamente los mismos que en el modo 2 y se diferencia de este en que la tasa de transferencia es variable.

Una instrucción que usa como registro de destino a SBUF inicia la transmisión en cualquiera de los cuatro modos, mientras que para iniciar la recepción en modo 0 se debe tener la condición de RI=0 y REN=1. En los tres modos restantes la recepción se inicia por el bit de arranque, siempre y cuando RI=1.

El registro que permite controlar el puerto serial y monitorear su estatus es el registro de función especial **SCON**, como puede apreciarse en la figura:

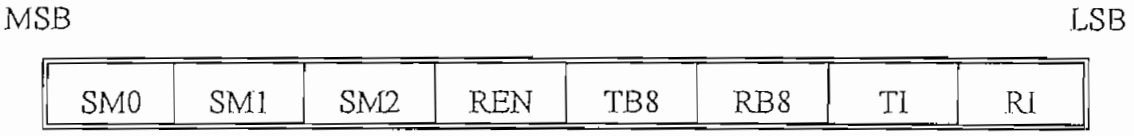


Fig. 3.18. Configuración de registro SCON.

Este registro contiene los bits para seleccionar el modo de operación, así como el sitio para almacenamiento del 9no bit de transmisión o recepción (TB8 y RB8), además de los bits de habilitación de la interrupción del puerto serial.

El control del modo de operación y concomitantemente de la velocidad de transmisión para el puerto serial se lo especifica mediante los bits SM0 y SM1 del registro SCON, en la siguiente forma:

VELOCIDAD DE TRANSMISION	MODO	SM0	SM1
$f_{osc}/12$	0	0	0
Variable	1	0	1
$f_{osc}/32$ ó $f_{osc}/64$	2	1	0
Variable	3	1	1

Tabla. 3.4. Modos de operación de puerto Serial.

Los restantes bits del Registro SCON tienen las siguientes funciones:

- **SM2.-** En los modos 2 y 3 habilita la comunicación multiprocesador.

- **REN.-** En 1L habilita la recepción serial, en 0L deshabilita la recepción; las dos opciones son

seleccionadas por software.

- **TB8.-** Se constituye en el 9no bit de datos que puede ser transmitido en los modos 2 y 3.
- **RB8.-** Es el 9no bit de datos que puede ser recibido en los modos 2 y 3.

- **TI.-** Es la bandera de interrupción en transmisión, seteada por hardware al final del 8vo bit en modo 0 ó al inicio del bit de parada en el resto de modos. En cualquiera de los modos debe ser reseteada por software.

- **RI.-** Es la bandera de interrupción en recepción. Seteada por hardware al final del 8vo bit en modo 0 ó durante el bit de parada en el resto de modos. También debe ser reseteada por software.

De acuerdo con el propósito del presente proyecto una tasa de transferencia fija resulta inadecuada, por lo tanto el modo de operación para el puerto serial deberá seleccionarse entre los modos 1 y 3. De estos dos se ha elegido el modo 1, esto es 1 bit de arranque, 8 bits de datos y un bit de parada, dando un total de 10 bits.

Cuando se trabaja en los modos 1 o 3, la velocidad de transferencia de datos a través del puerto serial está determinada por la tasa de sobreflujo del Timer1 y el valor de SMOD mediante la siguiente relación:

$$Velocidad=\frac{2^{SMOD}}{32}*RelaciónparadesbordamientodelTimer1$$

Ecuación 3.1

Siendo el Timer 1 el que determina el ritmo binario resulta relevante el realizar un breve análisis de los registros especiales que controlan a los temporizadores del microprocesador y que son los siguientes:

GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer 1				**Timer 0**			

Fig. 3.19. Rgistro de control de Temporizadorcs.

El microcontrolador 8051 incorpora dos registros temporizadores/contadores. Actuando como temporizador el registro se incrementa con cada ciclo de maquina por lo cual se podría pensar en el como un contador de ciclos de maquina. Y puesto que cada ciclo de máquina consiste de 12 períodos de oscilador, la tasa de conteo por lo tanto es de 1/12 la frecuencia del oscilador.

Cuando funciona como contador el registro se incrementa en cada transición de 1 a 0 lógicos en la entrada externa, en el correspondiente pin.

Las funciones de cada bit del registro **TMOD** se describe a continuación.

- **GATE.-** Habilita la entrada exterior.

- **C/T.-** Selector para operación como temporizador o contador

- **M1 y M2.-** Determinan el modo de operación según la siguiente tabla de opciones.

MODO	M1	M0	DESCRIPCION
Modo 0	0	0	Temporizador/Contador de 13 bits
Modo 1	0	1	Temporizador/Contador de 16 bits
Modo 2	1	0	Temporizador/Contador de 8bits con autorecarga
Modo 3	1	1	Varios contadores

Tabla. 3.5. Control de modo de trabajo de los temporizadores.

A continuación se expone una breve síntesis de cada modo de operación:

MODO 0.- En este caso cada uno de los temporizadores, es configurado como un registro de 13 bits , compuesto por los 8 bits de TH1 (TH0) y lo 5 bits menos significativos de TL1 (TL0), pudiendo ignorarse los tres bits más significativos de TL1 (TL0).

En este modo de operación los dos temporizadores actúan exactamente igual. Cuando la cuenta produce que todos los valores cambien de 1L a 0L la bandera de la interrupción del temporizador TF es activada (1L).

MODO 1.- La definición de los registros temporizadores en este caso utiliza todos los bits de TH y TL, esto da un total de 16 bits para cada uno de los temporizadores, el resto de la operación es exactamente la misma que en el modo 0.

MODO 2.- En este caso el registro del temporizador se configura como un contador de 8 bits TL1 (TL0) con recarga automática. El desborde (Un valor superior a 255) de TL1 (TL0) activa la bandera TF1 (TF0) y automáticamente el valor contenido en TH1 (TH0) es recargado en TL1 (TL0), sin que se altere el valor de TH1 (TH0) que se inicializa por software.

La operación es exactamente la misma para los contadores/temporizadores 0 y 1.

MODO 3.- Este modo esta previsto para aplicaciones que requieran de un contador ó temporizador adicional. Con el timer 0 trabajando en modo 3 un microcontrolador 8051 aparece como si tuviera tres temporizadores/contadores.

El Temporizador 1 en modo 3 mantiene sus características de conteo, como si TR1=0. Mientras que el temporizador 0 establece TH0 y TL0 como dos contadores independientes.

TL0 utiliza los bits de control del temporizador 0 esto es GATE, TR0, INT0 y TF0. TH0 por otra parte esta enclavado en la función de temporizador y utiliza los bits TR1 y TF1, por lo tanto TH0

ahora controla la interrupción del timer 1.

Según el resumen de los modos de operación cuando se utiliza el Timer 1 como generador del ritmo Binario para el puerto serial, las condiciones iniciales del mismo deben ser las primeras sentencias de la subrutina correspondiente.

El puerto serial debe ser configurado según los requerimientos del proceso de transmisión que se vaya a utilizar, para el presente proyecto y basándonos en las diferentes opciones discutidas en párrafos anteriores se ha utilizado la configuración siguiente:

- Se habilita el temporizador 1 para el conteo mediante TR1=1L.

- El puerto serial trabajara en modo 1, esto es un bit de inicio, 8 bits de datos, 1 bit de parada y con tasa de transmisión variable; por lo tanto el registro SCON se carga con el valor hexadecimal 50H.

- El ritmo binario para la comunicación serial es suministrada por el Temporizador 1, para tal efecto el registro TMOD se inicializa con el valor hexadecimal 20H, definiendo el trabajo del temporizador en Modo 2.

- La tasa de transferencia como ya se menciono se define por la ecuación 3.1, la cual en forma desarrollada puede apreciarse a continuación:

$$VELOCIDAD = \frac{2^{SMOD}}{32} * \frac{FRECUENCIADELOSCILADOR}{12*(256 - TH1)}$$

En la ecuación si la frecuencia del oscilador es conocida, el parámetro que define la tasa de transferencia es el valor con el cual se inicializa el registro TH del temporizador 1, este valor se recarga en forma automática cuando el temporizador trabaja en modo 2.

Por lo tanto para trabajar a 2400 bits por segundo y con un oscilador de 7.15909 MHZ, se inicializan TH1 y TL1 con el valor hexadecimal 0F8H.

El formato de la comunicación serial será aquel que utiliza el SIDES en su opción programar EPROMS .

3.4.14.- SUBRUTINA PARA MANEJO DE RAM Y COMUNICACION CON PC.

Para descargar un programa desde una computadora personal al equipo se utiliza el formato especificado por INTEL, según el cual los archivos .HEX tienen un conformación similar a la que se puede apreciar en el siguiente ejemplo:

:06*0000*00020030020004E78
:03000B00020050A0
:0300130002007474
:03001B000200766A
:0300230002007860
:2000300075815F758900758810758CFF758A00D2AFD2A97521057520007522058OFE80FE8D
:20005000B200D52113B201752105D5220BB202752205D523037523058520907 58CFF758A69
:0A007000003280FE80FE80FE80FE5C
:00000001FF

En primer lugar el símbolo : corresponde al la codificación en ASCII 3AH, que es código que indica inicio de una cadena de datos a ser transmitidos.
Los siguientes dos dígitos 06, en el caso de nuestro ejemplo, indican el número de datos que contiene esa cadena que se transmitirá.

los siguientes cuatro dígitos, en nuestro caso 0000 indican la dirección de memoria en la que

están almacenados los datos.

Los dos dígitos siguientes 00, en nuestro ejemplo son los que indican si se trata de una cadena intermedia de datos o por el contrario si en su lugar se encuentran los dígitos 01 significan fin de transmisión.

La finalización de la transmisión se corrobora si a continuación se recibe el código FF, tal como se puede apreciar en la ultima fila de nuestro archivo de ejemplo.

Después de los dígitos 00 vienen los datos propiamente dichos, y los últimos dos dígitos corresponden al valor para la verificación del checksum.

Para la implementación de esta función una vez seleccionada desde la pantalla principal aparece el mensaje de presentación, el cual permanece mientras no se presiona ninguna tecla, si la tecla presionada es EXE, el programa se redirecciona para su ejecución a partir de 2000h esto es desde RAM.

Por el contrario si cualquier otra tecla se ha presionado el equipo pasa un estado en el cual puede iniciar la recepción de los datos transmitidos a traves del puerto serial para lo cual se configura en primer lugar el puerto y luego el temporizador encargado de proveer la señal de reloj a 2400 baudios, tal como se aprecia en el diagrama de flujo de la fig. 3. 19 y continua en el diagrama de flujo de la figura 3.20.

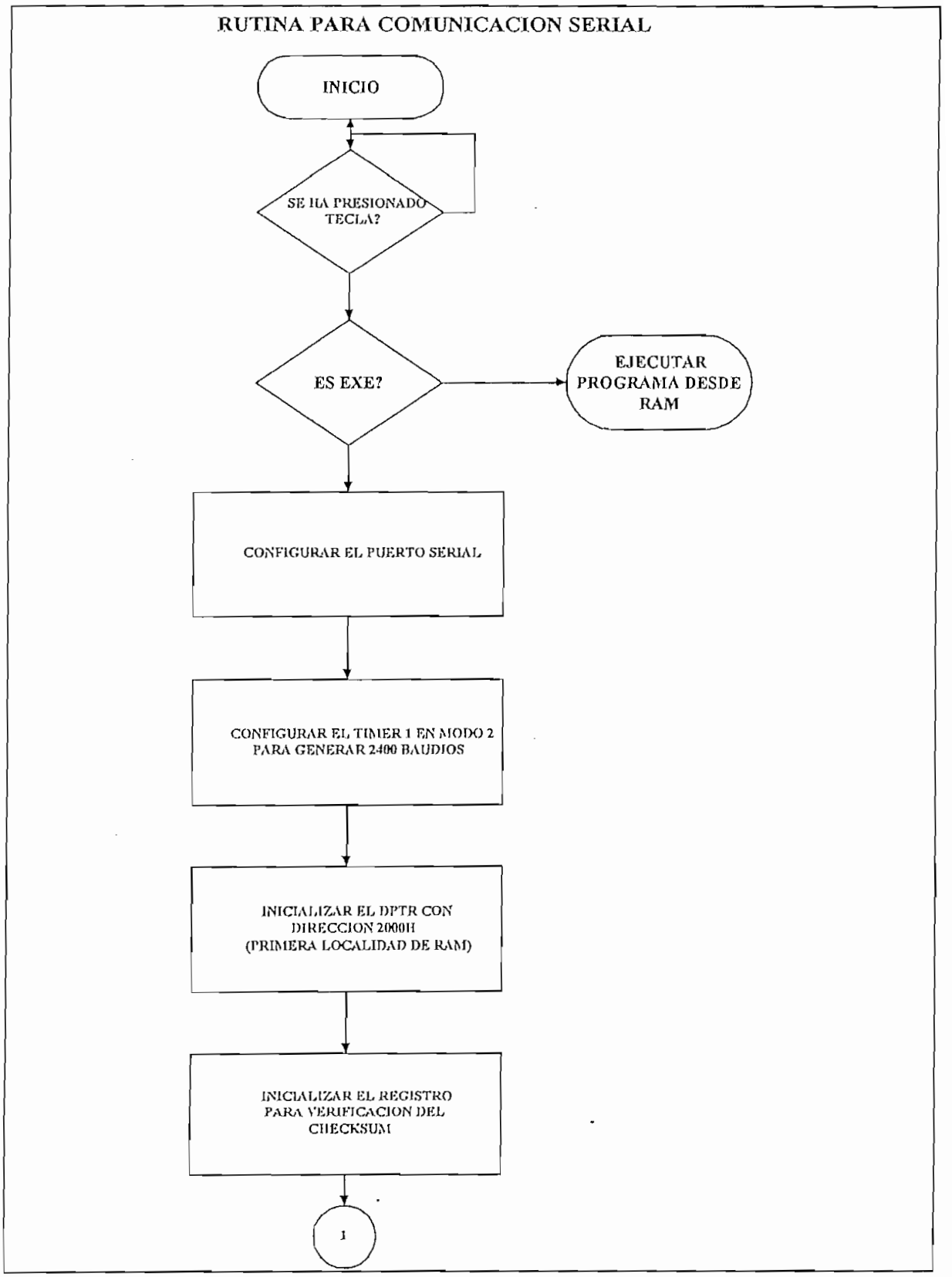


Fig. 3.20. Diagrama de Flujo de comunicación serial.

La recepción de los datos con codificación ASCII y su posterior almacenamiento como datos hexadecimales se inicia precisamente con un lazo de espera durante el cual el microcontrolador toma los datos de SBUF cada vez que la bandera RI se activa, para de inmediato determinar si se trata del código 3AH que indica inicio de transmisión, además se resguarda en un registro el valor para la comprobación del Checksum, en nuestro caso esto se lo hace en el registro R7 del Banco 2.

Si el código recibido coincide con 3AH se produce una llamada a la Subrutina CONVERT, encargada de recibir el dato serial y convertirlo a su valor Hexadecimal para posteriormente retornarlo en el registro R4. En este caso el segundo dato recibido corresponde al número de datos que contiene esa cadena por lo que se almacena este valor en un contador R5.

Se ejecuta entonces la recepción de los datos correspondientes a la dirección de memoria, los cuales son desechados, no sin antes haberlos considerado para el incremento del registro encargado del control de checksum. Estos datos se los ignora en el presente proyecto debido a que el ensamblador crea los códigos de modo que ocupan las localidades de memoria en forma secuencial.

Los siguientes dos dígitos que se reciben se verifican para comprobar si se trata de el fin de la transmisión o de una línea intermedia, si corresponde al fin de transmisión es decir 01H, se produce un salto a una rutina que comprueba si se recibe la cadena FFh para dar por terminada la recepción en forma normal o si existe un error, en cuyo caso se despliega un mensaje de error.

SI se trata de una línea intermedia los datos son recibidos, transformados de ASCII a hexadecimal y almacenados en la localidad de memoria indicada por el DPTR, mientras el contador de número de datos llega a su valor de 0. Cuando esto se produce se verifica el checksum y de no ser correcto se desplegara un mensaje de error.

Si el checksum esta correcto el programa salta al inicio par recibir la siguiente transmisión tal como se aprecia en el diagrama de flujo de la figura 3.21.

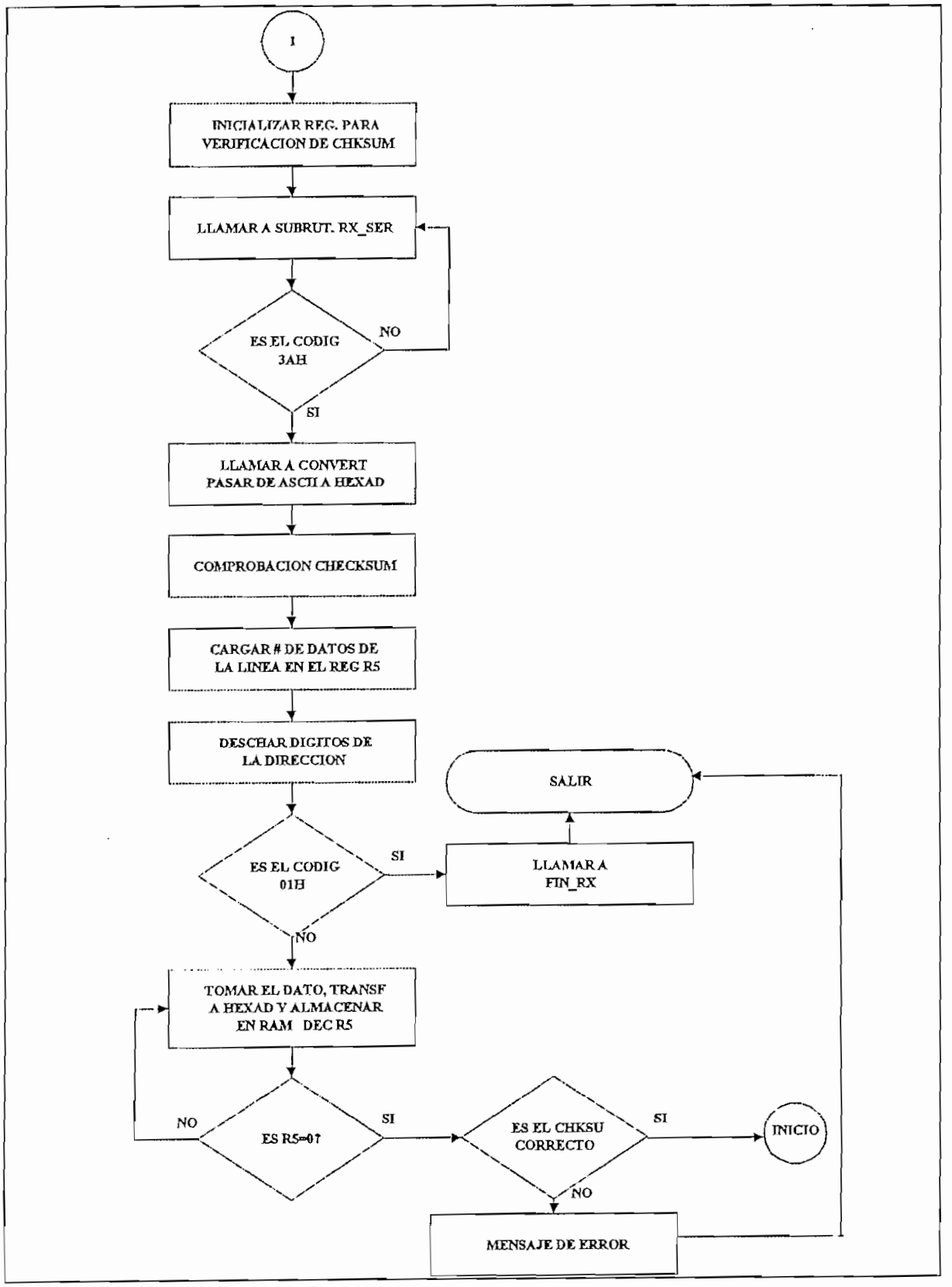


Fig. 3.21. Comunicación Serial Continuación

3.4.15.- SUBRUTINA CONVERT.

Esta subrutina se encarga de tomar el dato desde el puerto serial y convertirlo a formato

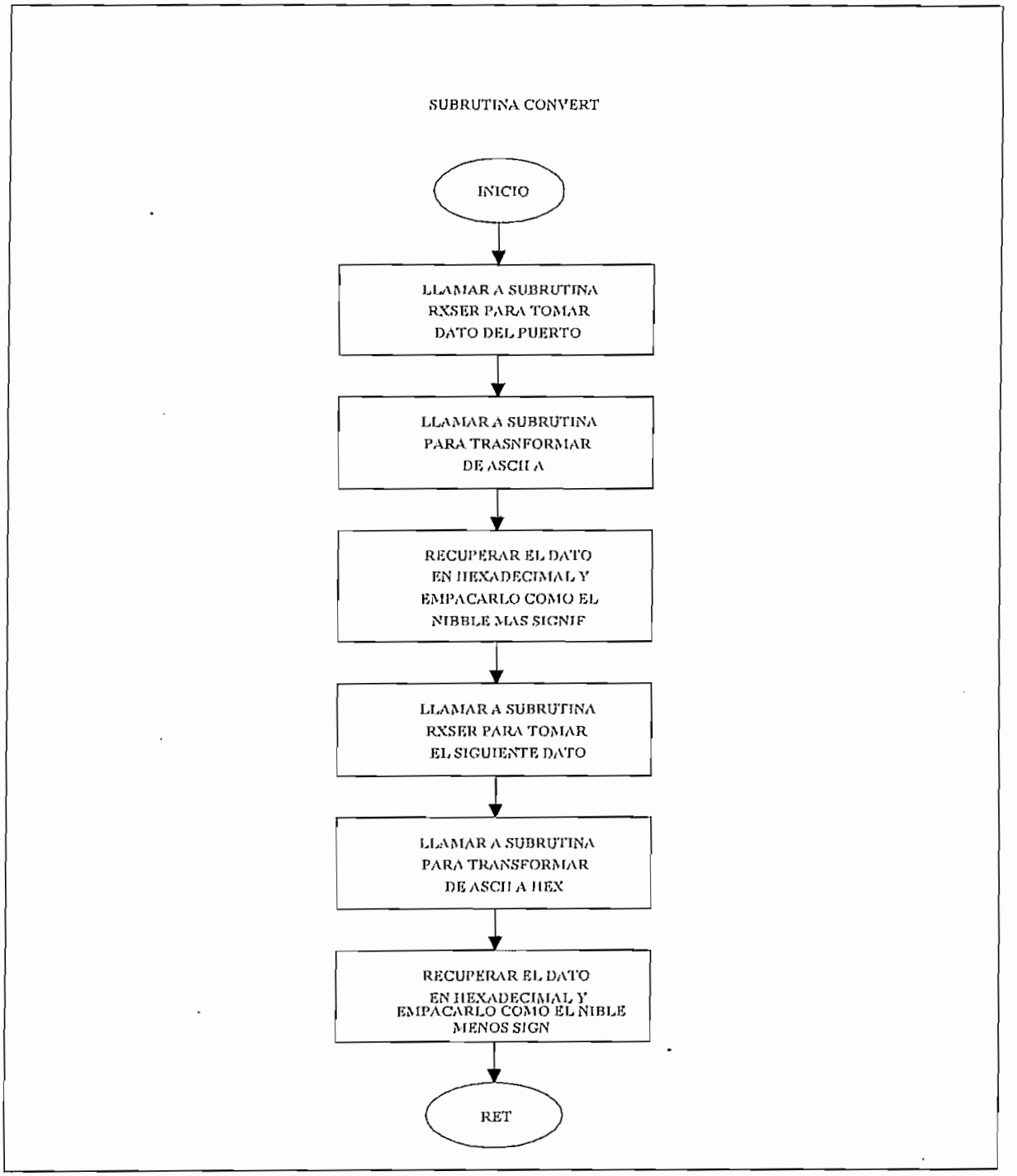


Fig. 3.22. **Diagrama de flujo de subrutina CONVERT.**

hexadecimal, mediante un nuevo llamado a otra subrutina que se encarga de realizar la conversión de ASCII as Hexadecimal.

Posteriormente se encarga de reordenar el dato para su almacenamiento en un byte (8 bits), esto último se requiere desde el momento en que la transformación a hexadecimal retorna la representación usando los 4 bits normales para este tipo de representación.
El diagrama de flujo que aparece en la figura 3.21., nos indica que se trata de un proceso bastante sencillo gracias a que se utilizan otras subrutinas.

3.4.16.- SUBRUTINA RX_SER.-

Una transición desde 1L a 0L producida por el bit de arranque de un caracter entrante, sobre la línea de recepción RX del microcontrolador, inicia el proceso de recepción por el puerto serial.

La subrutina RX_SER se mantiene en un lazo de espera mientras no se culmine la recepción del caracter, lo cual es señalizado mediante la bandera RI.Una vez que la bandera RI se activa, la subrutina deshabilita la recepción serial mediante un valor de 0L en el bit REN del registro SCON.

El dato recibido en SBUF, es pasado entonces al acumulador, después de lo cual se borra la bandera de recepción serial RI y se habilita la recepción serial mediante REN = 1L, quedando listo para la recepción del siguiente dato.

El diagrama de flujo de la figura 3.23, muestra exactamente la subrutina descrita.

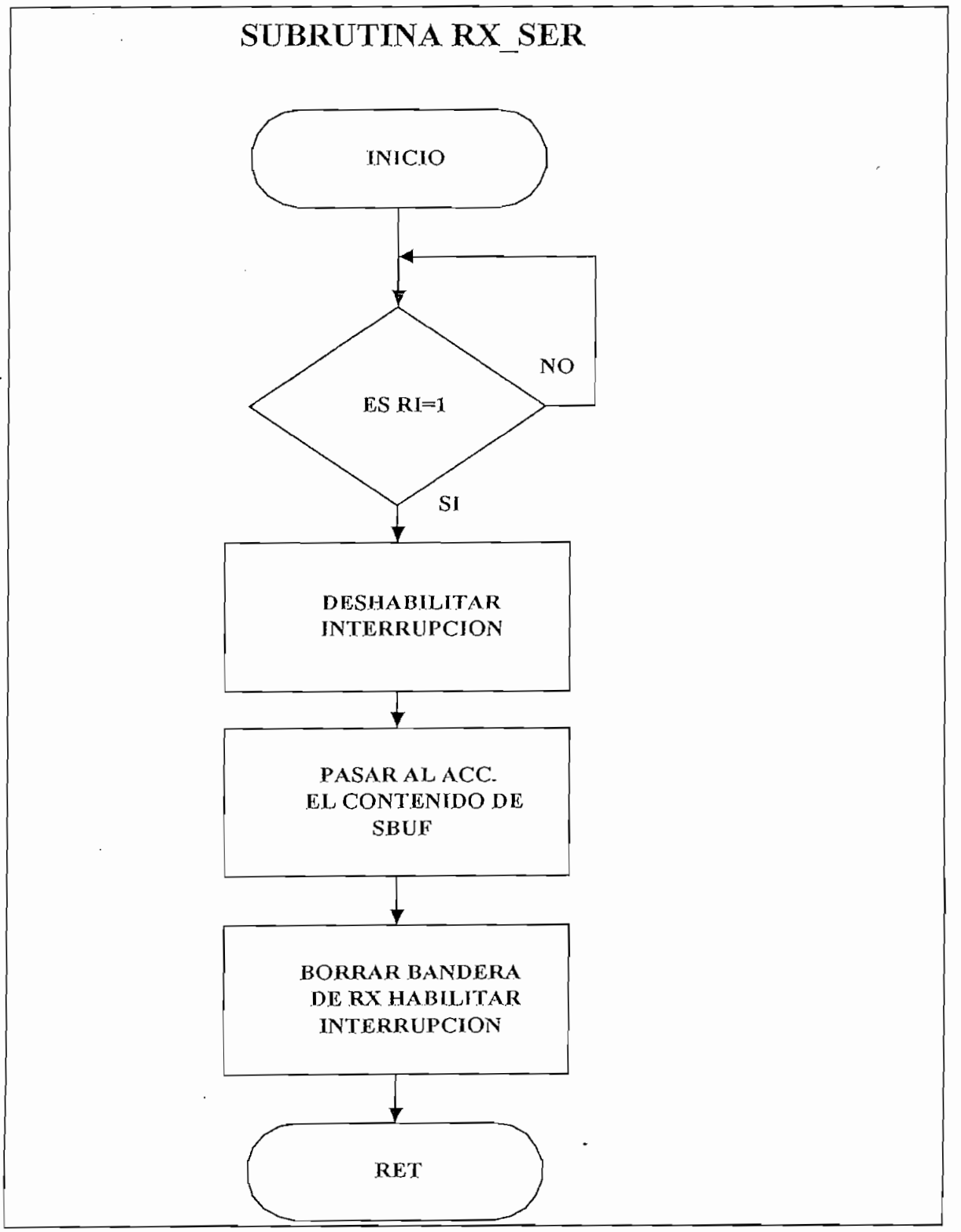


Fig. 3.23. Diagrama de Flujo de la RX Serial.

3.4.17.- SUBRUTINA FIN_RX.

Es una pequeña subrutina que se encarga de comprobar en las operaciones de trasnmisión serial para las comunicaciones del microcontrolador, que los datos se reciben correctamente de

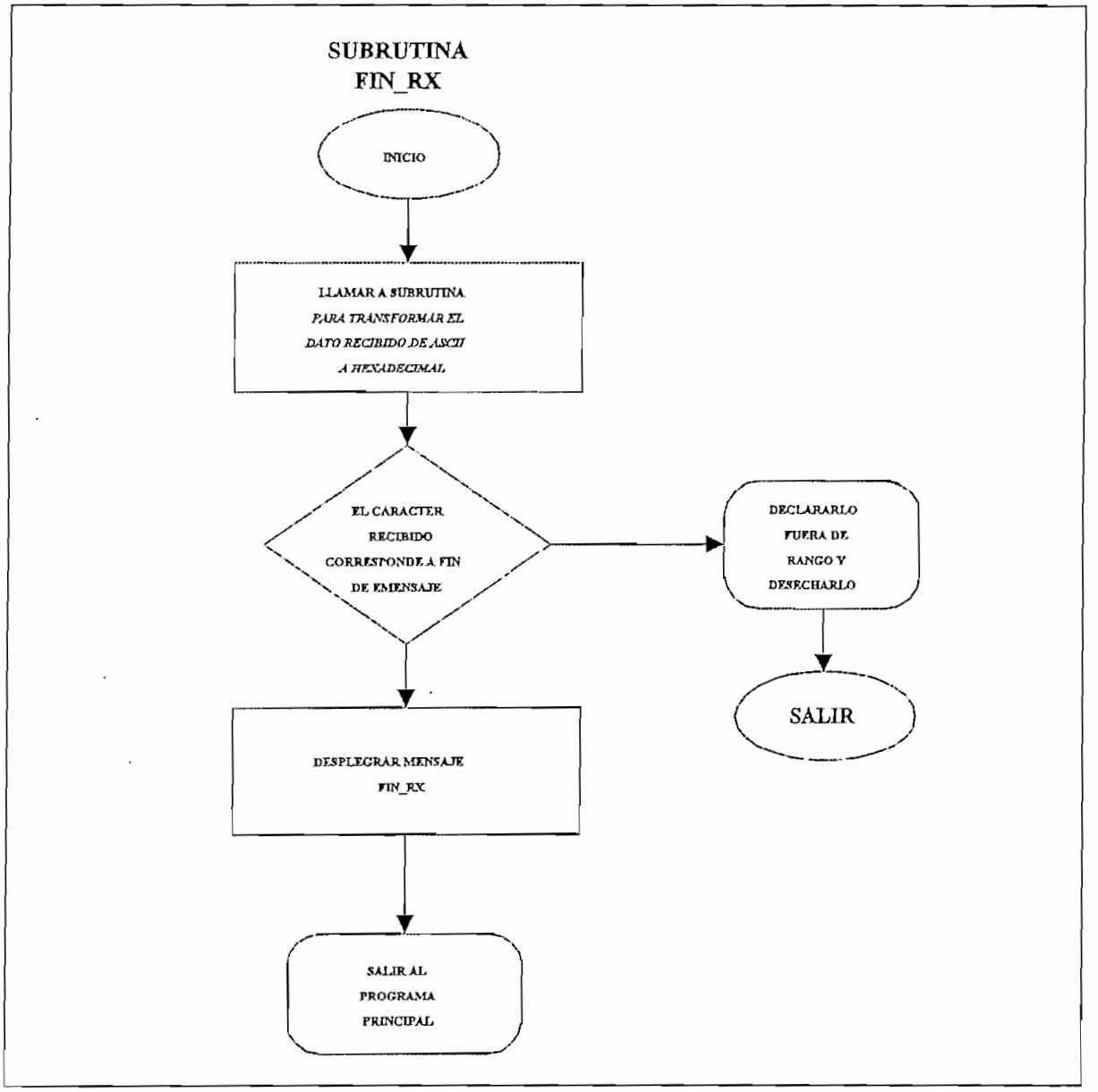


Fig. 3.24. Diagrama de flujo FIN_RX.

lo contrario debe desplegar un mensaje de datos ilegal, que advertira al usuario que existe algún tipo de problema en los datos que se estan recibiendo.

En la figura 3.24. se aprecia el proceso que se aplica para esta comprobación, notandose además que se utilizan llamados a otras subrutinas como el medio de actuar sobre los datos recibidos para su comprobación.

3.4.18.- SUBRUTINA PARA GENERACION DE RELOJ SINCRONIZADO CON LA SEÑAL AC.

Para la implementación de esta opción del prototipo se aprovecha de la segunda interrupción externa de la que disponen los microcontroladores de la familia MCS51, como se recordara la interrupción externa o, se la utiliza para la detección de tecla presionada.

En forma muy sucinta para generar la señal de reloj sincronizada con la red se seguira el siguiente procedimiento:

Se toma una muestra de señal AC a partir de la fuente de alimentación del equipo, específicamente de la salida del puente rectificador de onda completa, y mediante un diodo se aisla del capacitor de la misma fuente.

Esta señal se la alimenta a un detector de cruces por cero implementada en base a un diodo Zener y un transistor, etapa que adicionalmente esta formada por los elementos necesarios para transformar la señal a niveles manejados por la familia TTL, tal como se explico en el capitulo II.

Esta señal se alimenta a la entrada del microprocesador correspondiente a la interrupción externa 1 es decir INT1. En otras palabras el tren de pulsos generado por la circuiteria encargada de la detección de los cruces por cero de la señal de corriente alterna.

Se programan los registros de control de habilitación de la interrupción externa 1 de modo que esta se active por flanco descendente y no por estado, ya que esta forma de trabajo se ajusta más a nuestras necesidades.

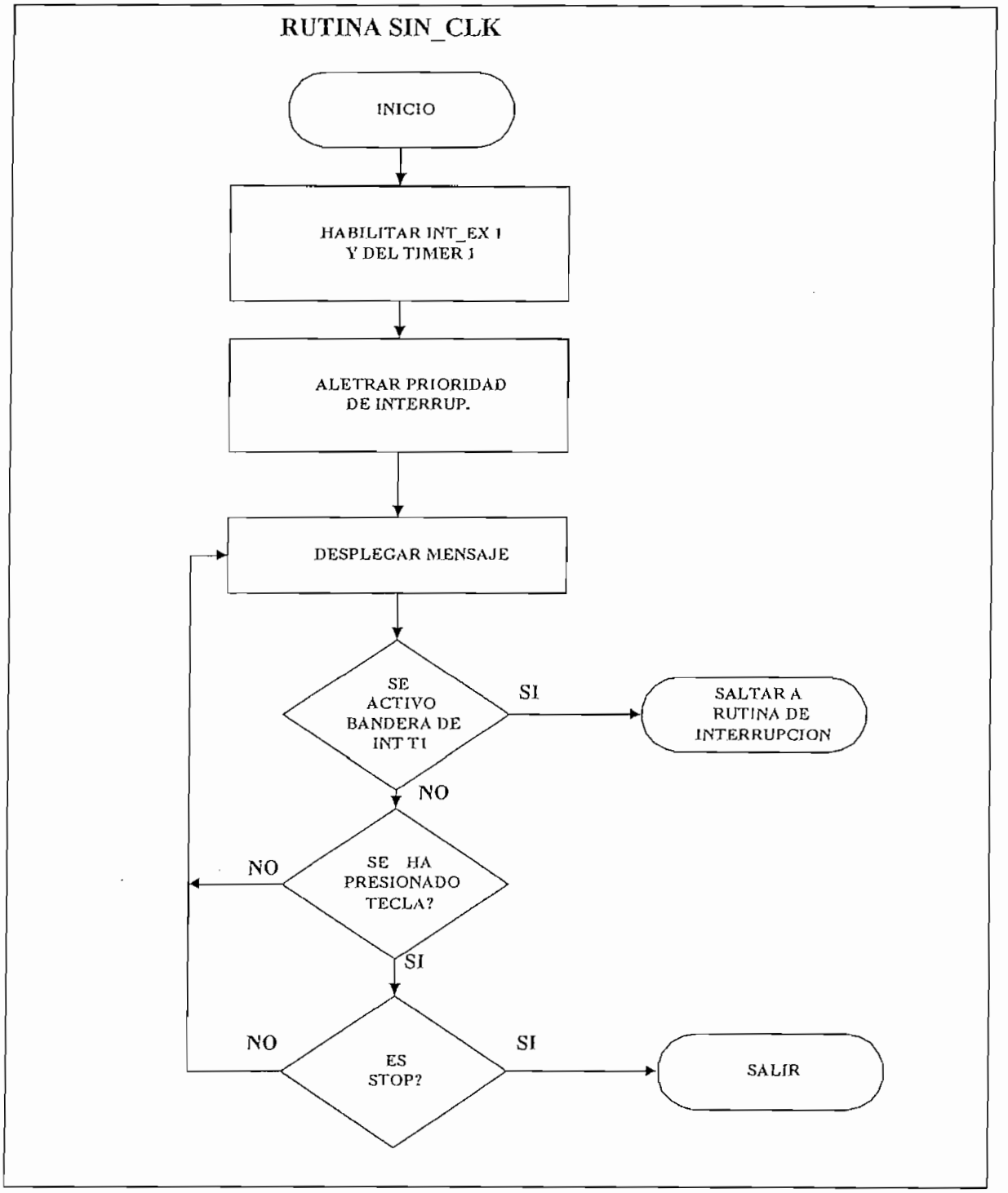


Fig. 3. 25. Diagrama de flujo para generar reloj sincronizado con la red AC.

Una vez que se ha detectado el flanco en el pin de ingreso el programa automáticamente salta a la rutina de atención a la interrupción la cual se encarga de sacar a través del puerto 1 pin 0 la señal digital que se constituye en la señal de reloj sincronizada con la red, la frecuencia de esta

señal, en todo momento será del doble de la frecuencia de la señal de la red comercial, dado que el número de cruces por cero se detecta en la señal rectificada.

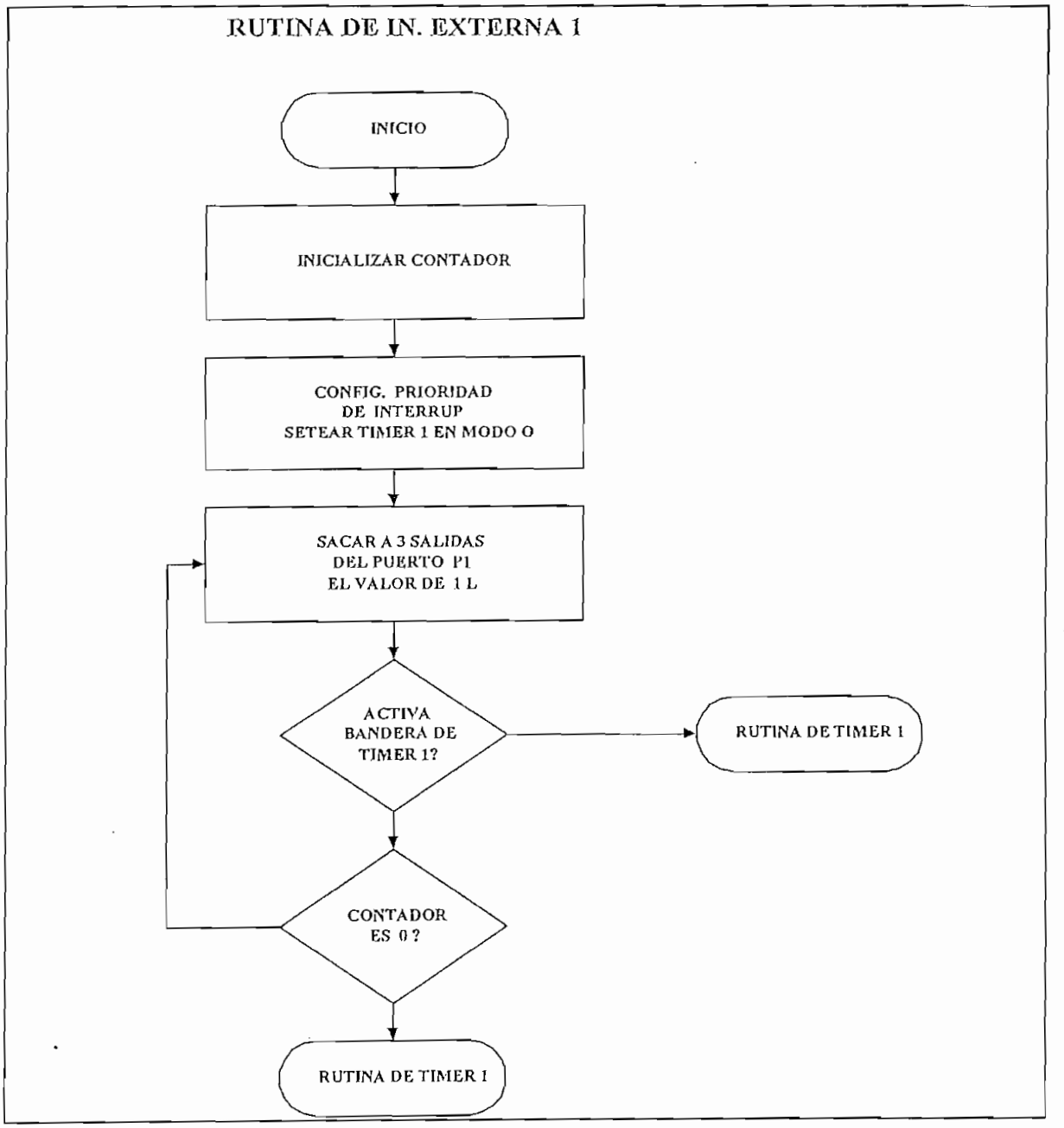


Fig. 3.26. Diagrama de flujo rutina de interrpción externa 1.

3.4.19. - SUBRUTINA DE INTERRUPCION EXTERNA 1.

Esta subrutina hace parte de la función para generar un reloj sincronizado con la red A.C. tal

como se explico en el apartado anterior.

Es necesario cambiar el orden de prioridad de las interrupciones, de modo que al interrupción del temporizador 1, según se aprecia en la figura 3.26. , utilizada para la temporización de la señal de reloj que se genera actúe de forma adecuada.

Se utiliza el temporizador 1 en modo 0.para generar la mitad del ciclo de la señal de reloj en valor bajo o 0 lógico.

Esta rutina saca a los bits menos significativos del puerto 1 un valor de 1 lógico de forma sincronizada con la detección de los cruces por cero de la muestra de la señal rectificada.

Cuando la bandera de interrupción del temporizador 1 se activa la ejecución del programa salta a la rutina de atención a la interrupción del temporizador 1, encargada de proporcionar la zona de la señal de reloj con valor bajo, tal como se aprecia en el diagrama de flujo de la figura 3.26.

3.4.20.- SUBRUTINA PARA GENERAR FRECUENCIAS.

Como se menciono en los capítulos anteriores, una de las opciones que esta en capacidad de proporciona el prototipo es la generación de frecuencias, estas señales saldrán a través de el puerto 1 del microcontrolador

Una vez que se ha seleccionado la función de generación de frecuencias desde el teclado, el programa principal direcciona hacia la subrutina encargada de tal proceso, la primera etapa de la cual se encarga de desplegar un mensaje de presentación mientras no se oprima la tecla ejecutar.

Para esta rutina se utilizan varios contadores, los cuales han sido creados en base a localidades de memoria que se especifican a continuación:

Contador	Dirección de Memoria
fz	20 H
Cont 1	21 H
Cont 2	22 H
Cont 3	23 H
Cont 4	24 H

Tabla. 3.6. Definicion de contadores en localidades de memoria.

El contador rotulado como **fz** es el encargado de sacar los valores de la señal de frecuencia.

Para generar la frecuencia nos valemos de el timer 0 del microcontrolador 8031, el cual debe ser configurado para trabajar en modo 0. y cargado con el valor inicial de TH0=FFH y TL0=00H, además se habilita la interrupción del temporizador 0.

Una vez que el temporizadorr supera el valor de conteo establecido se activa la bandera de sobrepasamiento del temporizador 0 esto es TF0, en el registro de control TCON. e inmediatamente salta a la rutina de atención a la interrupción correspondiente.

Esta rutina en primer lugar complementa el valor anterior que tenía el bit menos significativo de el registro rotulado como fz, decrementa comprueba el contador 1, si este no es igual a cero salta a la sentencia que saca al bit menos significativo del puerto 1 el nuevo valor de fz, este proceso se repite hasta que el contador 1 alcanza su valor de 0, en cuyo caso se complementa el siguiente bit de fz y se lo saca al puerto 1.

Esta forma de trabajo implica que la frecuencia en el bit 2 de fz sea diez veces menor a la del bit menos significativo de fz. De modo que cada uno de los bits de P1 varia en relación de frecuencia en un factor de 10 con respecto al siguiente en el orden de significación.

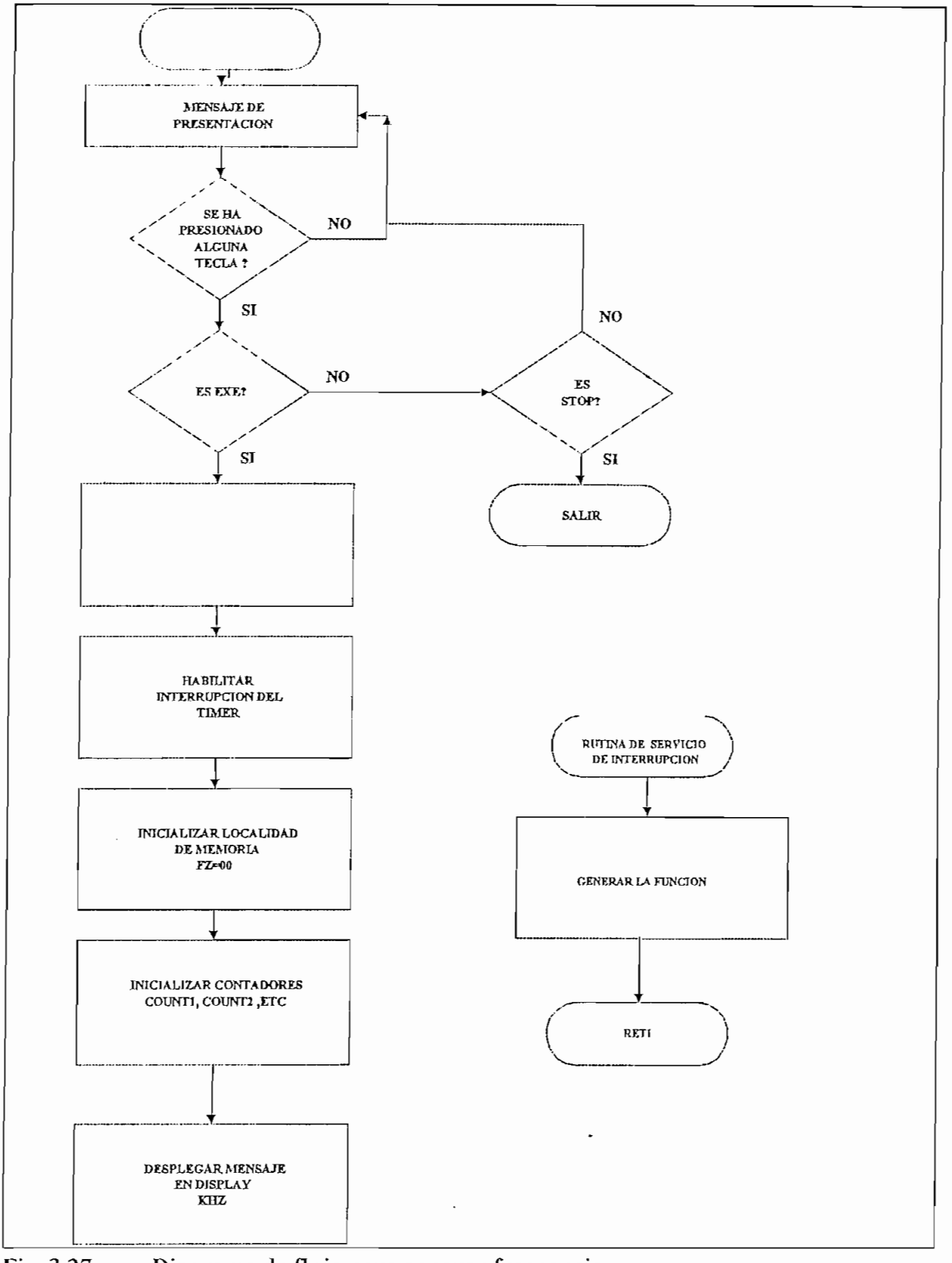


Fig. 3.27. Diagrama de flujo para generar frecuencia.

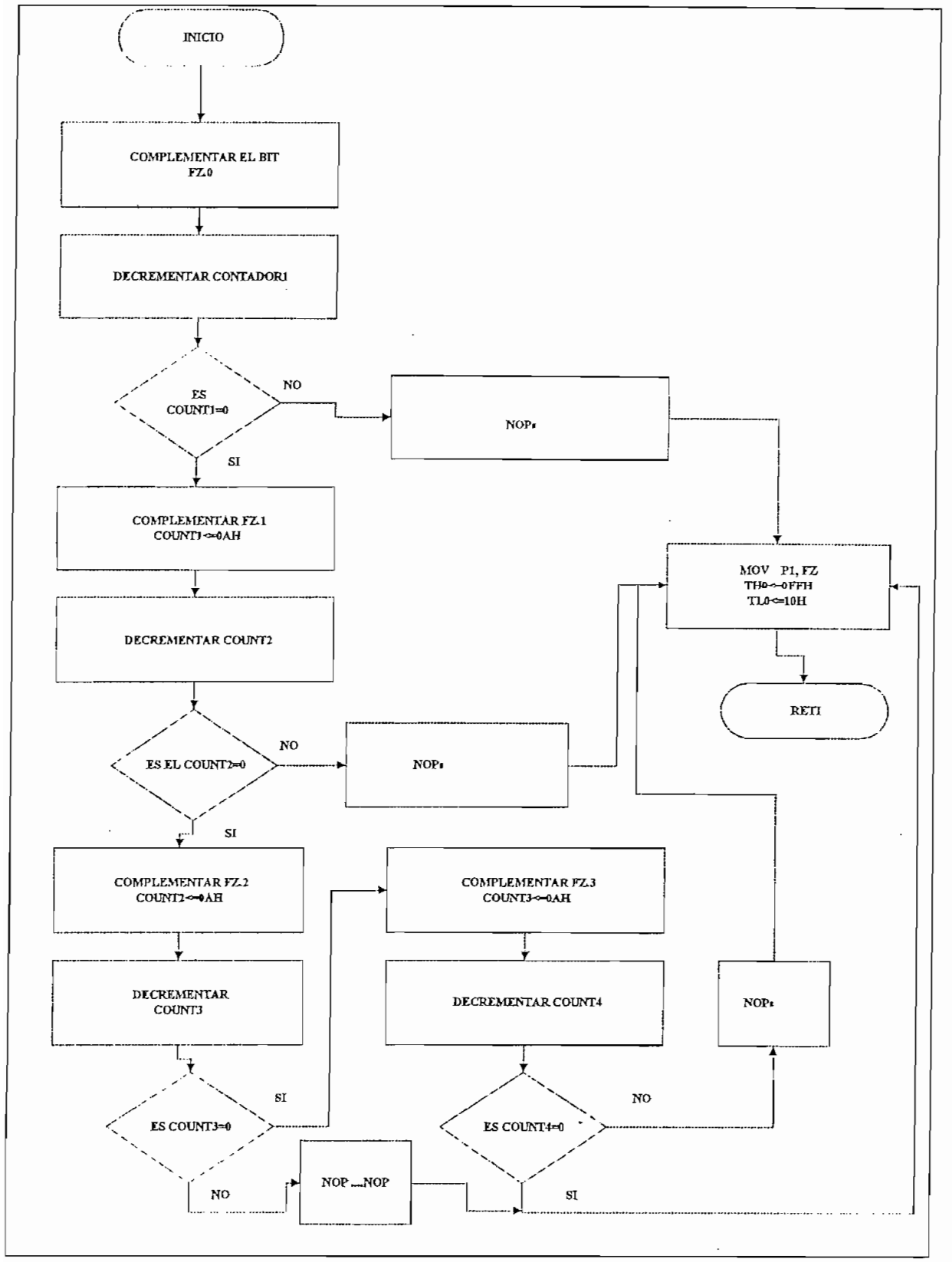


Fig. 3.28. Diagrama de flujo de rutina de interrupción del timer 0.

3.4.21.- SUBRUTINA PARA LECTURA DE CONTENIDO DE RAM.

La ejecución de esta rutina permite al usuario desplegar en el display la dirección de memoria de datos externa (RAM externa) y el contenido de la misma en siguiente formato **0000=FFH** en el cual los 4 dígitos más significativos corresponden a la dirección de memoria, mientras que su contenido aparece precedido por el signo de igualdad y finalmente la indicación de que se trata de un valor hexadecimal.

Al igual que las otras rutinas seleccionadas como funciones desde el menú principal, en primer lugar despliega los mensajes de presentación correspondientes, esta presentación permanece mientras no se presione la tecla EXE que es la que ordena la ejecución propiamente dicha.

Una vez iniciada la ejecución de la rutina, esta permite avanzar a través de las localidades de memoria paso a paso con solo presionar la tecla EXE ó por el contrario se puede retroceder a la localidad anterior mediante la tecla SEL.

Se procede entonces a tratar los datos obtenidos separándolos en nibbles (4 bits) empezando por el menos significativo, del cual se obtiene la codificación en Hexadecimal adecuada para que su valor pueda aparecer en el display, para lo cual se usa una subrutina ALMRAM la que además se encarga de almacenar la codificación resultante en la localidad de RAM interna correspondiente al dígito que se esta tratando; este procedimiento se repite para el segundo dígito del valor en hexadecimal contenido en la localidad de memoria externa.

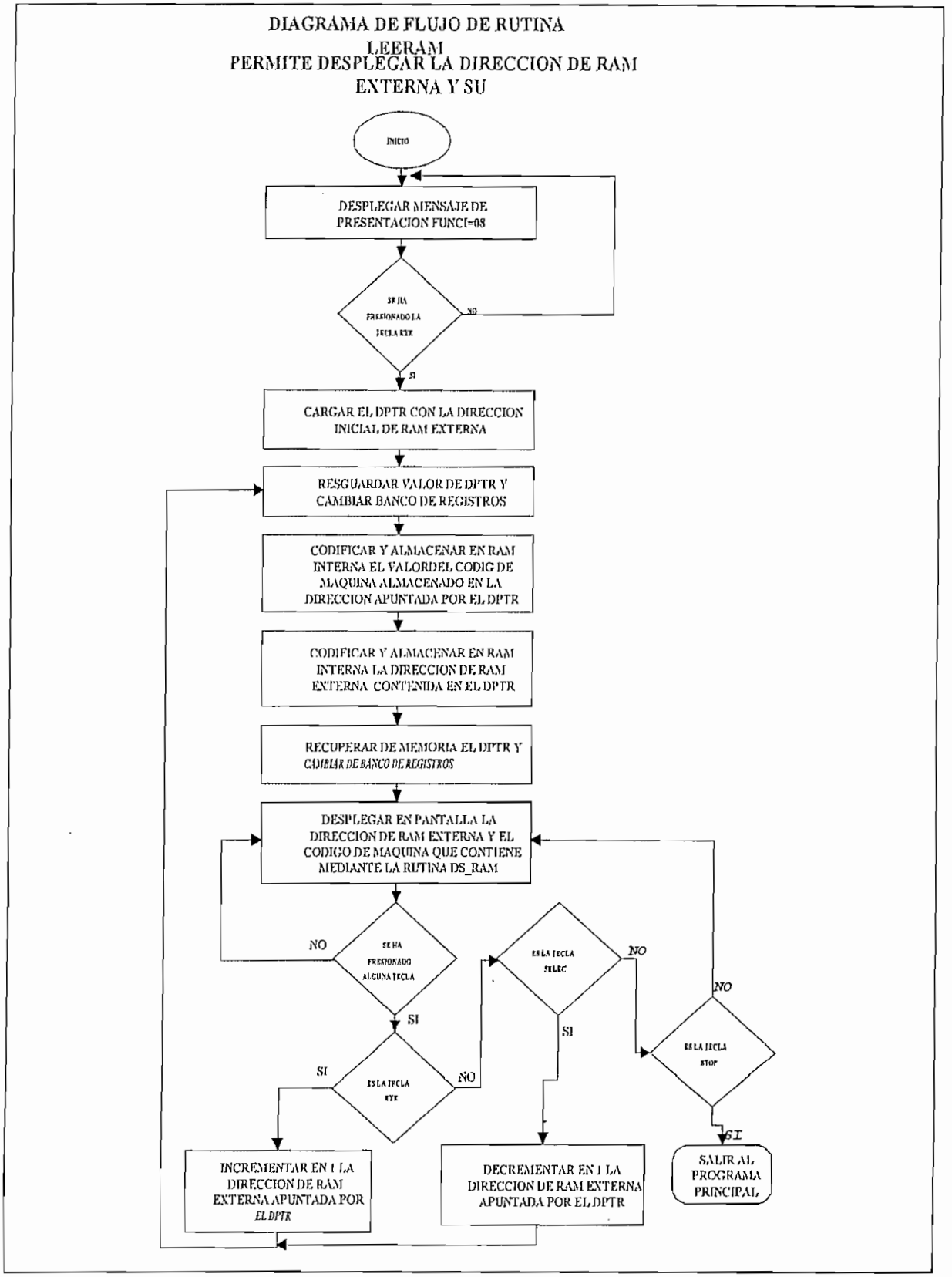


Fig. 3. 29. Diagrama de flujo para lectura de contenido de RAM.

En cuanto a la Dirección de RAM externa, este valor se recupera desde los registros R2 y R3, procediendo a realizar un proceso similar al descrito, de modo que una vez que se ha repetido por un total de 6 ocasiones este proceso(incluyendo las dos del contenido), las localidades de RAM interna destinadas a almacenar los códigos para el despliegue de resultados en el display, guardan la codificación necesaria para que en el arreglo de matrices aparezca la dirección de RAM y su contenido.

Se recupera desde la memoria donde se resguardo el valor de la dirección de RAM contenido originalmente por el DPTR y se lo transfiere esta vez a los registros R0 y R1.

Se utiliza en este punto la rutina que controla al display y los resultados se exponen en el mismo, permaneciendo inalterados mientras no se vuelva a presionar ninguna tecla; cuando esto sucede se realiza un proceso a fin de determinar de que tecla se trata; de ser la tecla STOP, se detiene la ejecución y se regresa al programa principal.

Por otra parte si se trata de la tecla EXE se ejecuta una subrutina que incrementa en uno la dirección contenida en el DPTR y se produce un salto incondicional al inicio del proceso de codificación de valores y despliegue de resultados.

En caso contrario, esto es si la tecla presionada ha sido SEL, la dirección contenida en el DPTR se decrementa en 1 y también se produce un salto al inicio de la rutina. En este caso, al no existir una instrucción que permita decrementar directamente el DPTR, se utiliza una substracción desde el valor contenido en el byte menos significativo del DPTR, es decir DPL, y una vez efectuada esta operación, se recarga con el resultado este registro.

Este procedimiento se aprecia mejor en el diagrama de Flujo de la figura 3.29.

3.4.22.- SUBRUTINA PARA GENERACION DE SECUENCIAS LOGICAS.

Como ya se menciono en el capitulo I los sistemas digitales de tipo Combinacional requieren de secuencias lógicas para probar su funcionamiento esta necesidad la satisface en equipo mediante el procedimiento que se aprecia en el diagrama de flujo correspondiente a la figura 3.30.

Para la salida de las secuencias lógicas se utiliza el puerto C de el PPI 8255 denominado como U10, por lo tanto el primer paso consiste en configurar el puerto como salida mediante la sentencia correspondiente.

La subrutina una vez seleccionada mediante el teclado saca a la pantalla el mensaje para que el usuario realice la selección de las dos posibilidades disponibles, esto es que la generación de las secuencias se realice paso a paso cada vez que se presiona la tecla EXE o si se genera la secuencia lógica en forma automático a una frecuencia especifica; en cualquiera de las dos formas se puede detener la generación de la secuencia mediante la tecla stop.

Para generar la secuencia se utiliza uno de los registros del Banco 0 actuando como contador, en cada incremento de dicho registro se genera una combinación de los 8 bits correspondientes a cada uno de los pasos de la secuencia lógica comprendidos entre 00H y FFH simultáneamente en el arreglo de matrices que conforman el visualizador del prototipo aparece la secuencia correspondiente en Binario, esto implica la codificación de cada uno de los 8 bits para que aparezca el valor correspondiente en cada una de las 8 matrices que conforman el arreglo.

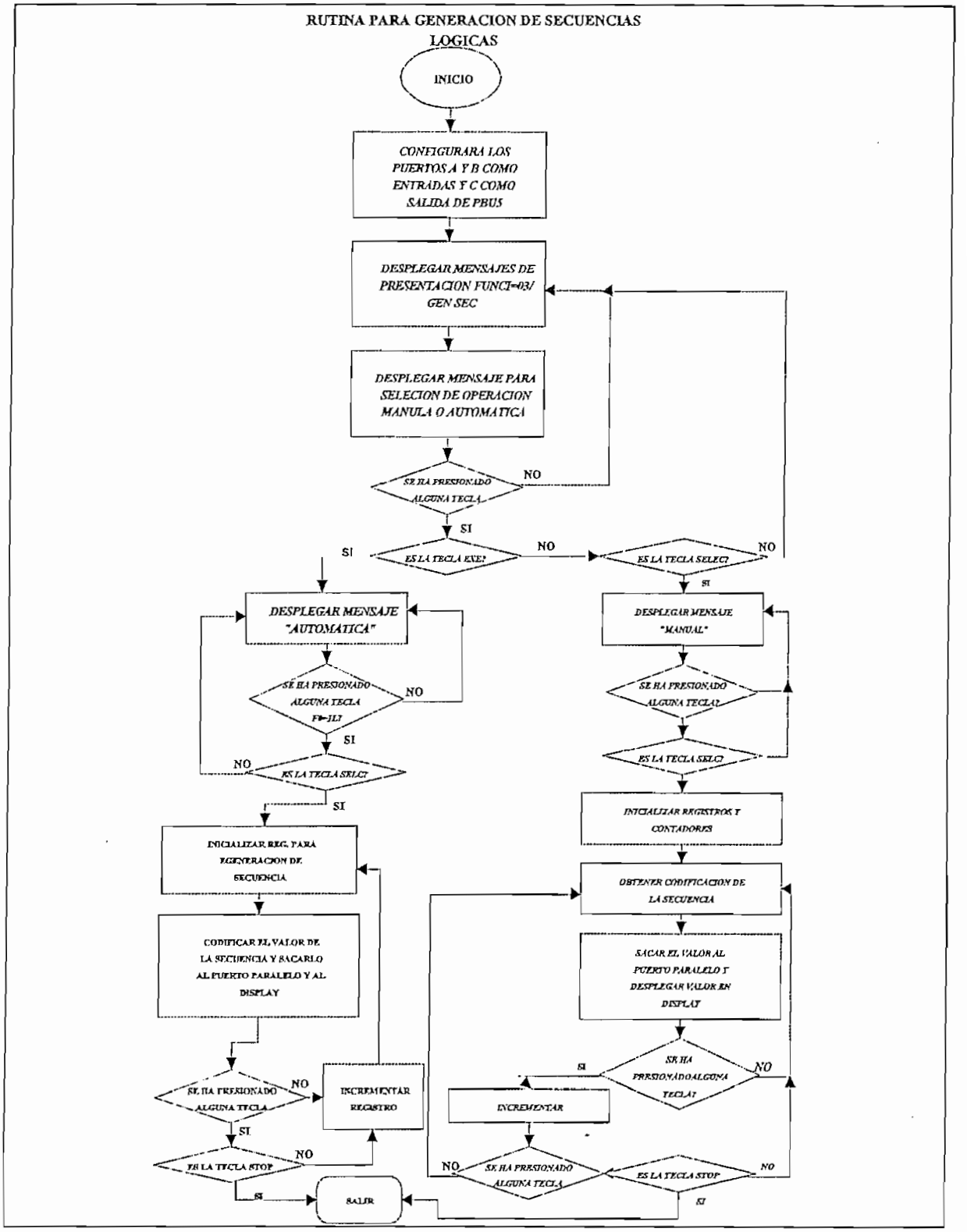


Fig. 3.30. Diagrama de flujo para generación de secuencias lógicas.

Esta codificación se la realiza de la siguiente manera:

El valor del contador se lo pasa al Acumulador, se realiza una rotación con carry hacia la derecha y en función de si existe o no carry se produce un salto a la tabla de codificación correspondiente, es decir.

- Si la bandera del carry contiene un valor de 1L se reduce un salto a una sentencia que carga en el DPTR la dirección correspondiente al inicio de la tabla de codificación que permite que en la matriz correspondiente aparezca el numero 1.

- Si la bandera del carry es 0L se realiza el salto de modo que se carga la codificación correspondiente para que aparezca el 0 en el dígito correspondiente.

Previo a cada uno de los incrementos en el contador se realiza una verificación de la bandera de tecla presionada, si esta ha sido activada se verifica si el código ingresado corresponde a STOP en cuyo caso se detiene la ejecución y se retorna al programa principal, de lo contrario continua con la generación de la secuencia.

Como puede apreciarse en el diagrama de flujo, quizas se esta la rutina con mayor número de decisiones a tomar en función de la tecla que ha sido seleccionada por el usuario.

3.4.23.- SUBRUTINA PARA MANEJO DE CARGAS A.C.

El control de cargas alimentadas por corriente alterna se lo podrá realizar de acuerdo con la programación que se desee, para efectos de este proyecto se desarrolla una rutina que permite operar de la siguiente forma:

Al seleccionar esta función desde el teclado, se despliega en el display un mensaje de

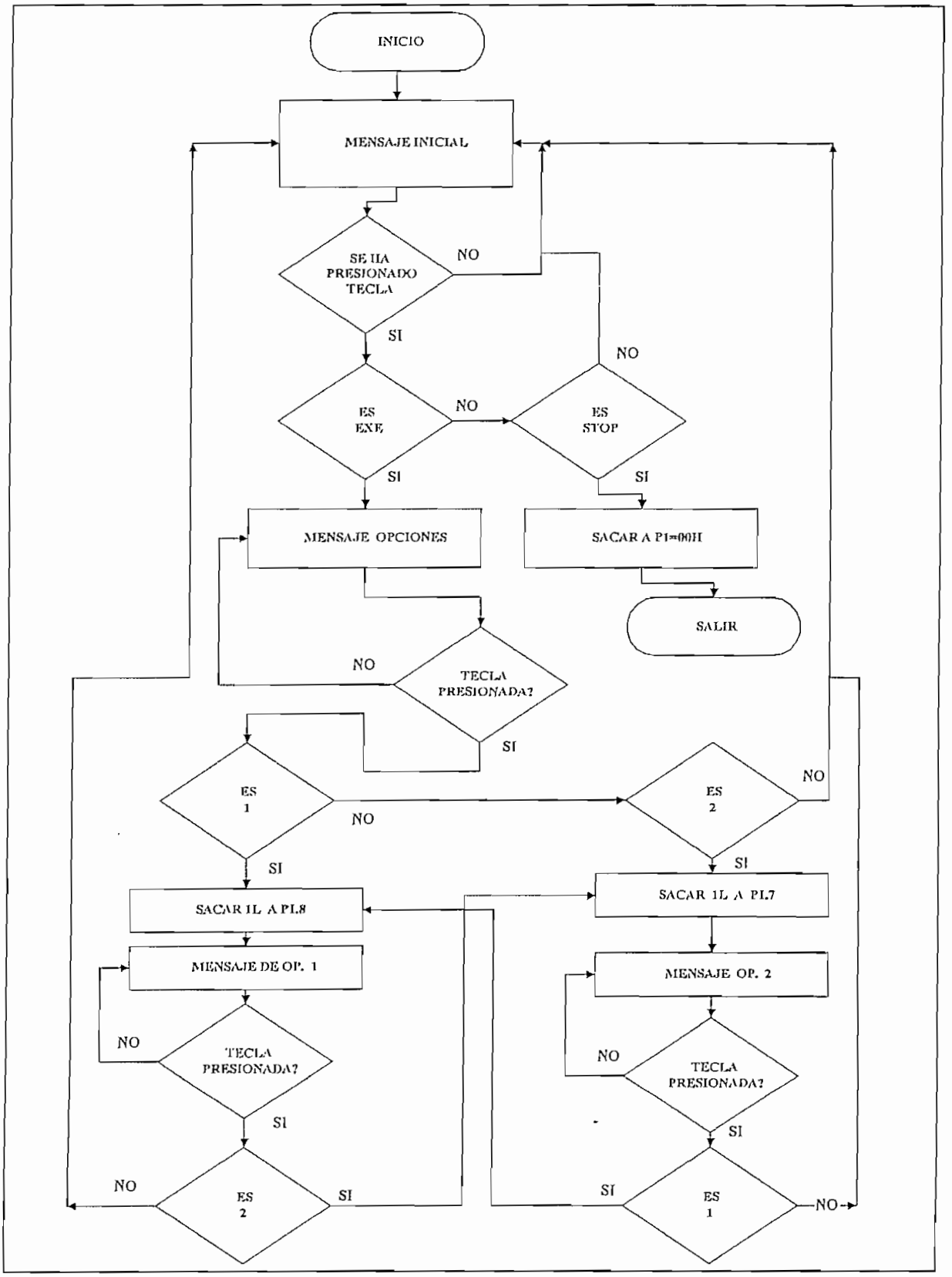


Fig. 3.31 . Diagrama de flujo para control de cargas AC.

presentación de la rutina, el cual se mantendrá en pantalla mientras no se realice una selección mediante teclado.

Si se presiona la tecla STOP, se retorna al programa principal, si se presiona cualquier otra tecla que no sea EXE, el programa se mantiene presentando el mensaje en pantalla.

Por el contrario al presionar la tecla EXE, aparece un nuevo mensaje que invita a seleccionar una de las dos opciones, que en realidad corresponden a la activación de una de las dos cargas conectadas a los terminales de los TRIACS.

Si se presiona en ese momento la tecla 1, se activara o pasara a estado 1 lógico la salida más significativa del puerto P1 y se despliega un mensaje indicando que se ha seleccionado esa opción, este valor provocara el disparo del TRIAC 1 y por tanto la carga conectada a esos terminales, se activara. Permaneciendo en ese estado mientras no se presione una tecla adecuada.

Si se presiona otra tecla y esta corresponde a 2, se activara o pasara a 1 lógico la penúltima salida del puerto 1, es decir P1.7, nuevamente en este caso se energizará la carga conectada a esa parte de la etapa de potencia.

Si se ha seleccionado cualquier otra tecla, el programa retornara al mensaje inicial desde el cual se puede seleccionar la tecla STOP, para apagar las cargas energizadas.

La operación puede repetirse en orden inverso si en primer lugar se presiona la tecla 2, pero de forma totalmente similar, tal como se puede apreciar si se observa el diagrama de flujo en la figura 3.31 .

Como se menciono este es solo un programa de ejemplo pudiendo desarrollarse rutinas que activen la etapa de potencia de muy diversas maneras.

3.5.- LISTADO DE PROGRAMAS.

El listado que contiene toda la programación realizada para el funcionamiento del prototipo se lo presenta en un anexo, debido a que ocupa aproximadamente 40 páginas, este anexo se lo entrega junto con el volumen empastado.

CAPITULO IV.

RESULTADOS EXPERIMENTALES.

4.1.- Pruebas de Laboratorio.

El procedimiento de pruebas que se utiliza en el desarrollo de proyectos que implican la programación de microcontroladores normalmente se inicia con un proceso de simulación de las rutinas, para lo cual existen una serie de productos de software que permiten realizar la simulación con mayor o menor precisión.

En el presente proyecto se utilizó el simulador AVSIM51. que es un programa diseñado para simular completamente la familia de microprocesadores 8051. La arquitectura completa de la CPU puede ser simulada, incluyendo temporizadores, interrupciones y puertos; lo que permite depurar los códigos que utilicen cualquiera de las características del microprocesador, facilitando en forma notable el proceso de aislamiento de errores en la programación. El programa muestra en la pantalla de simulación la condición de todos los registros, banderas, puertos y áreas de memoria seleccionables por el usuario, durante la ejecución del programa bajo prueba.

Existen varias razones para utilizar la simulación como la primera fase de pruebas, entre estas tenemos:

La disminución del tiempo para el desarrollo de un proyecto al utilizar la simulación como el medio para depurar los errores en la programación tales como efectos no deseados de ciertas instrucciones sobre registros no previstos y que afectan a la respuesta del microcontrolador.

Al disponerse de un medio para observar los efectos de las modalidades de direccionamiento, cambios en el contenido de registros y zonas de memoria, que se producen al introducir nuevos

códigos como parte de las rutinas, se pueden detectar y eliminar errores que alteran los resultados esperados

Cada una de las rutinas puede probarse y depurarse por separado antes de ser incluida en el programa principal, sin necesidad de programar y borrar en repetidas ocasiones los dispositivos de memoria del prototipo en desarrollo, además se pueden analizar los efectos al incluir esta rutina en el programa general.

Cada una de las rutinas y subrutinas de las que consta el software del prototipo fueron sometidas a la simulación de modo que aquellos problemas originados por lazos infinitos, saltos a direcciones no especificadas, etc. pudieron ser eliminados

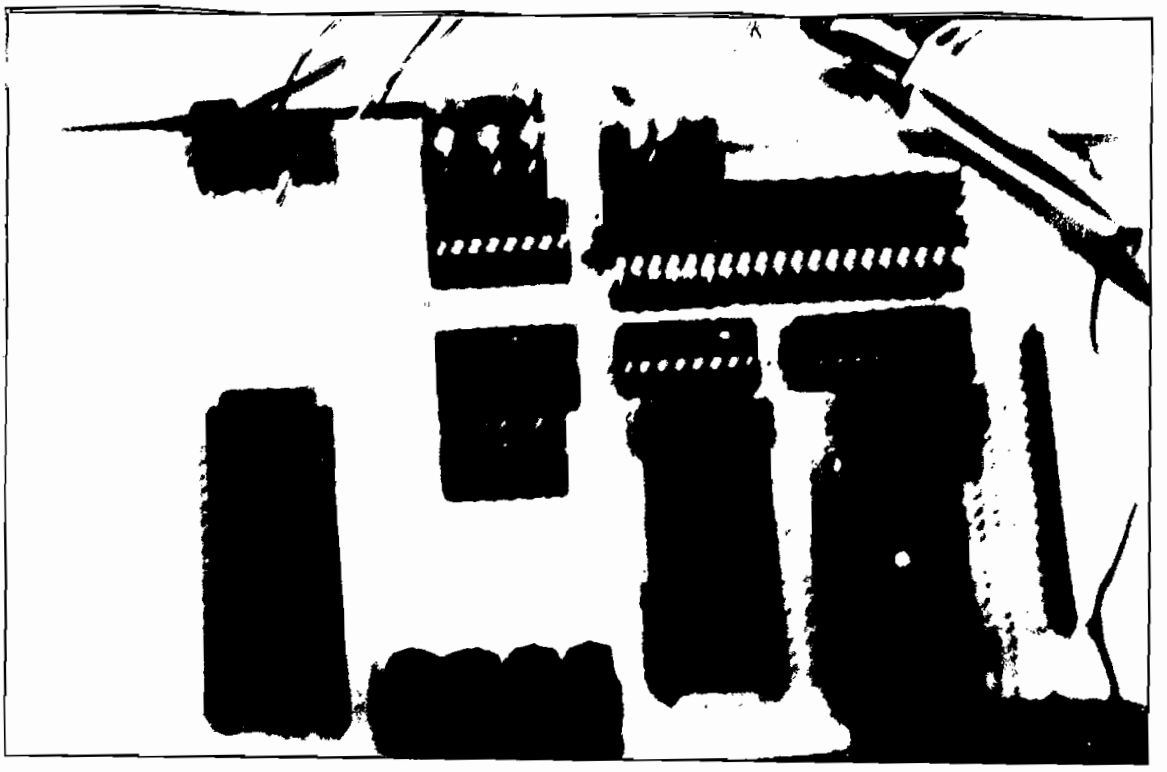
Una vez superada la etapa descrita la simulación de las diferentes funciones previstas para el equipo permitió comprobar que los resultados que se obtienen bajo las diferentes condiciones y de acuerdo a los parámetros ingresados concuerdan plenamente con lo esperado.

La simulación sin embargo tiene una gran limitación cuando se trata de bloques de temporización, en general y a pesar de utilizar un computador de alta velocidad, cualquier rutina que implique lazos de espera por software o conteo de ciclos de máquina por medio de los temporizadores tiene una duración demasiado larga, por lo que en general este tipo de rutinas se las prueba con valores de los contadores especificados intencionalmente bajos.

Para verificar el correcto funcionamiento de el prototipo se realizaron varias pruebas y comprobaciones:

En primer lugar y dado que la presentación de resultados depende en gran medida de los datos desplegados en el display, se considero importante determinar la frecuencia con la cual se barre a las columnas del display.

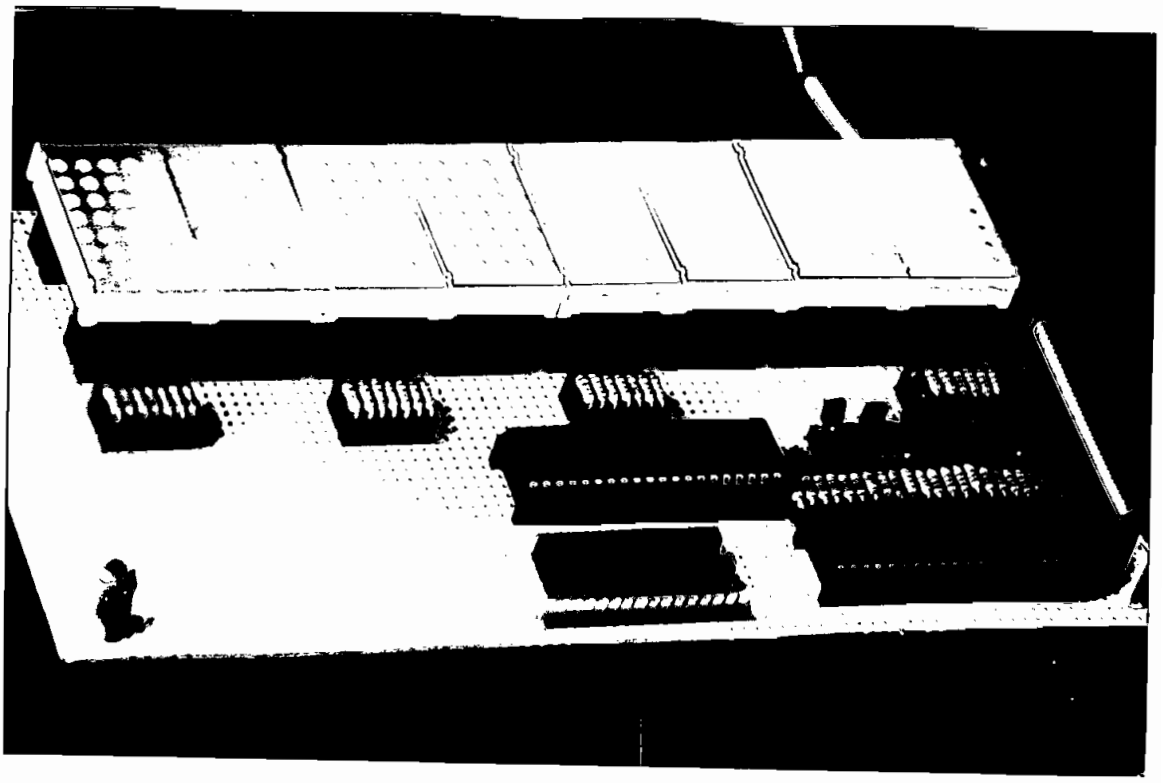
Mediante el uso de un osciloscopio y conectando la punta de prueba a la base de uno de los transistores que se encarga de activar una columna cualquiera, se pudo verificar que la frecuencia con la que se repiten los pulsos esta alrededor de los 380 Hz, y por supuesto depende directamente de la frecuencia del cristal que se utiliza para la temporización del microcontrolador.



Fotografia 1. Placa de la Unidad Central de proceso.

La placa correspondiente a la unidad central de procesoaparece en la fotografia 1:

En la siguiente fotografía puede apreciarse la placa en la cual se ha armado el display y sus componentes relacionados, claramente se aprecian los dos circuitos integrados 8255, así como los bancos de resistencias y los 10 transistores encargados del barrido de las columnas.



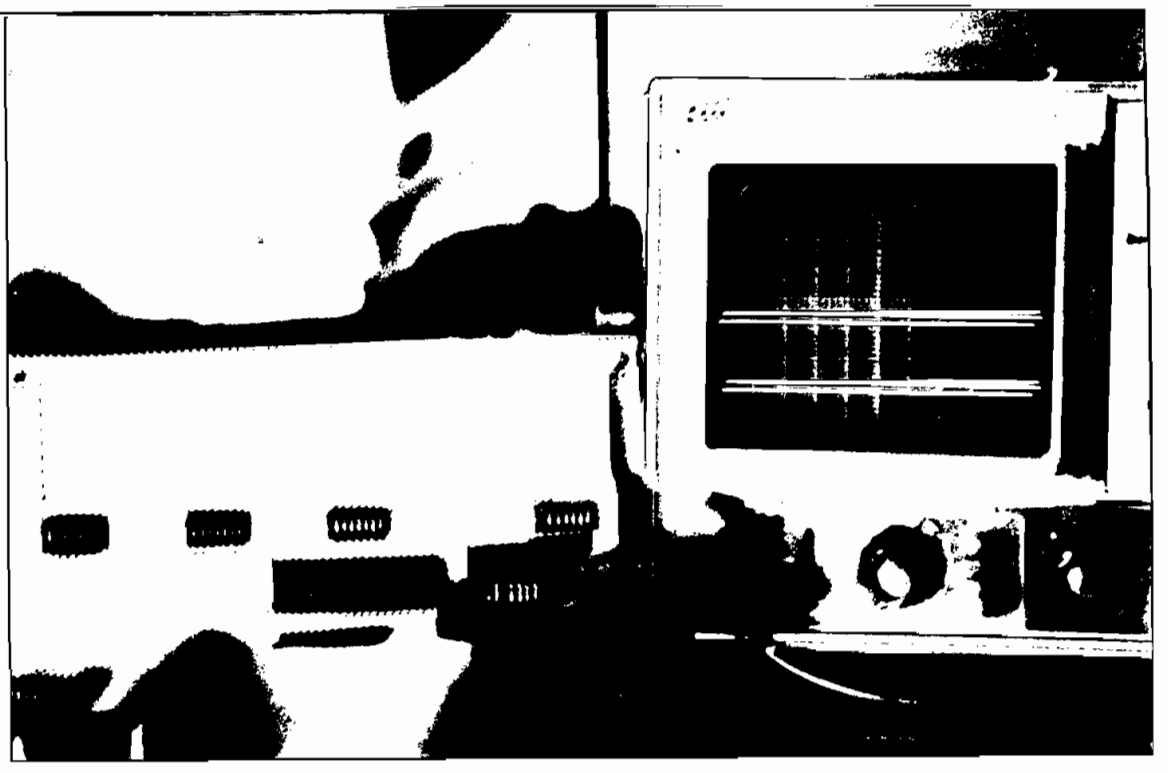
Fotografia 2 Tarjeta de display.

Una vez que el equipo ha realizado el proceso de arranque normal, es posible necesario presionar la tecla FUN, para pasar a la pantalla de selección de funciones, desde la cual se puede avanzar a cada una de la funciones previstas, esta pantalla se puede apreciar en la fotografía 3.

4.1.1.- Pruebas de Generacion de Frecuencias:

Utilizando un osciloscopio se realizan pruebas de verificación para comprobar la exactitud de la frecuencia de las señales generadas.

Como ya se menciono el equipo provee de una señal de frecuencia de alrededor de 10 KHZ la cual se puede tomar a través de uno de l os pines del puerto 1, normalmente P1.0, y se pueden tomar señales divididas por 10 en los siguientes 3 pines.



Fotografia 3 Mensaje en pantalla para selección de funciones.

La comprobación de esta función se la realizo mediante Osciloscopio, determinadose que se logran valores muy aproximados a los 10 Khz, no se logra una exactitud absoluta debido a que el cristal utilizado tiene una frecuencia de 7.15909 Mhz y por lo tanto cada ciclo de máquina o 12 períodos de oscilación del reloj tiene una duración de 1.6762 Microsegundos.

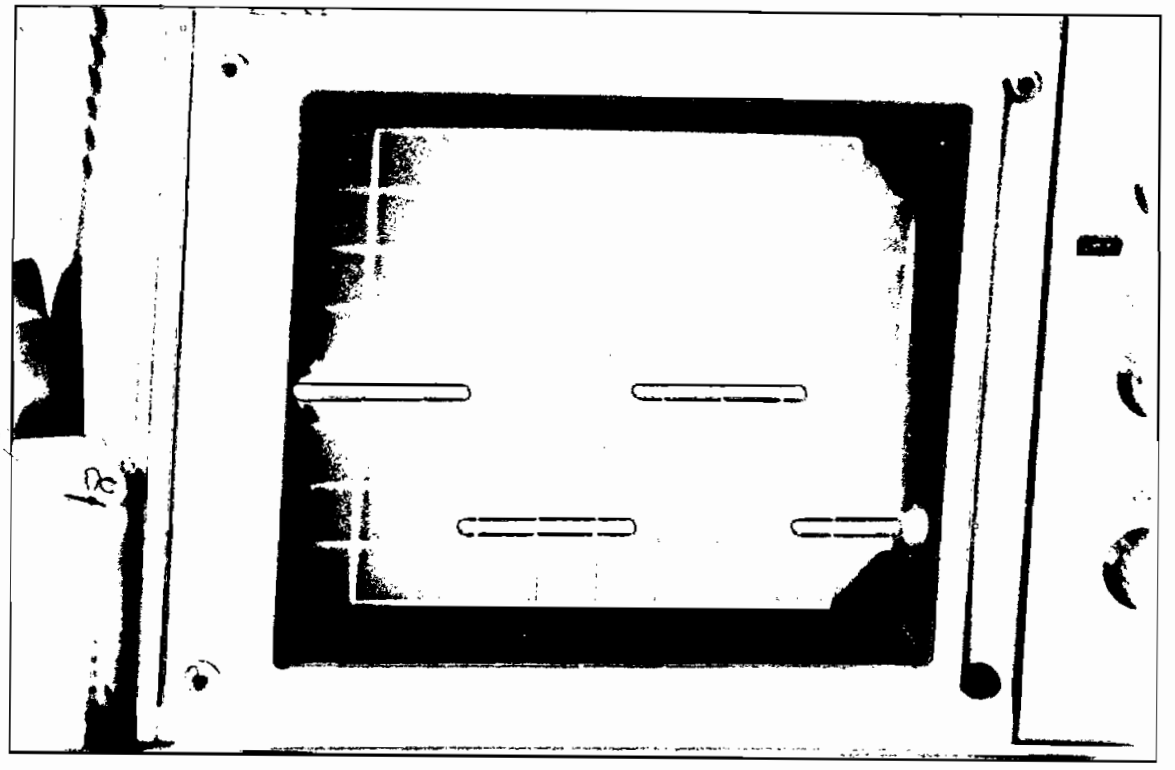
Para tener una señal de 10 Khz, o lo que es lo mismo un período de 100 microsegundos, se necesitaría realizar una programación tal que se ejecutaran 59. 5 ciclos de máquina; de ahí se desprende que la señal obtenida no sea exacta al 100 %.

Sin embargo lo que si se pudo comprobar es que la relación de 10 a 1 entre las señales generadas si se logra en base a los diferentes contadores. .

También es necesario anotar que se debió utilizar sentencias **NOP** para compensar los saltos entre los diferentes lazos. Una de las frecuencias generadas en su fase de comprobación

mediante el osciloscopio se aprecia en la siguiente fotografía 4.

La frecuencia que se consigue generar depende de el valor de recarga del temporizador 0 específicamente de el valor de TL0, así se pudo comprobar que con un valor de TL0 =00H, se logra una frecuencia de alrededor de los 380 Hz.



Fotografía 4 Muestra de una frecuencia generada por el equipo.

4.1.2.- Pruebas de Generación de Secuencias:

Para la comprobación de esta función se utiliza tanto un osciloscopio, como la inspección visual a través de los LEDS de salida. Una vez conectado el osciloscopio se determinó el correcto funcionamiento de esta función, se comprobó además que las alternativas para detener la generación automática en un valor determinado mediante la tecla 2 funciona en forma eficiente.

Mediante la selección de la tecla 1 durante la ejecución del proceso de generación de secuencias

se puede reinicializar desde el valor 00000000, posibilidad que se usa tanto en la opción manual como en la automática y que pudo comprobarse funciona adecuadamente

Fotografía 5. Muestra de mensaje durante la Generación de secuencias lógicas.

En la fotografía 5 se aprecia uno de los valores de estas secuencias desplegado en la pantalla del equipo mientras se realizaba la comprobación del funcionamiento adecuado de esta forma de operación del equipo.

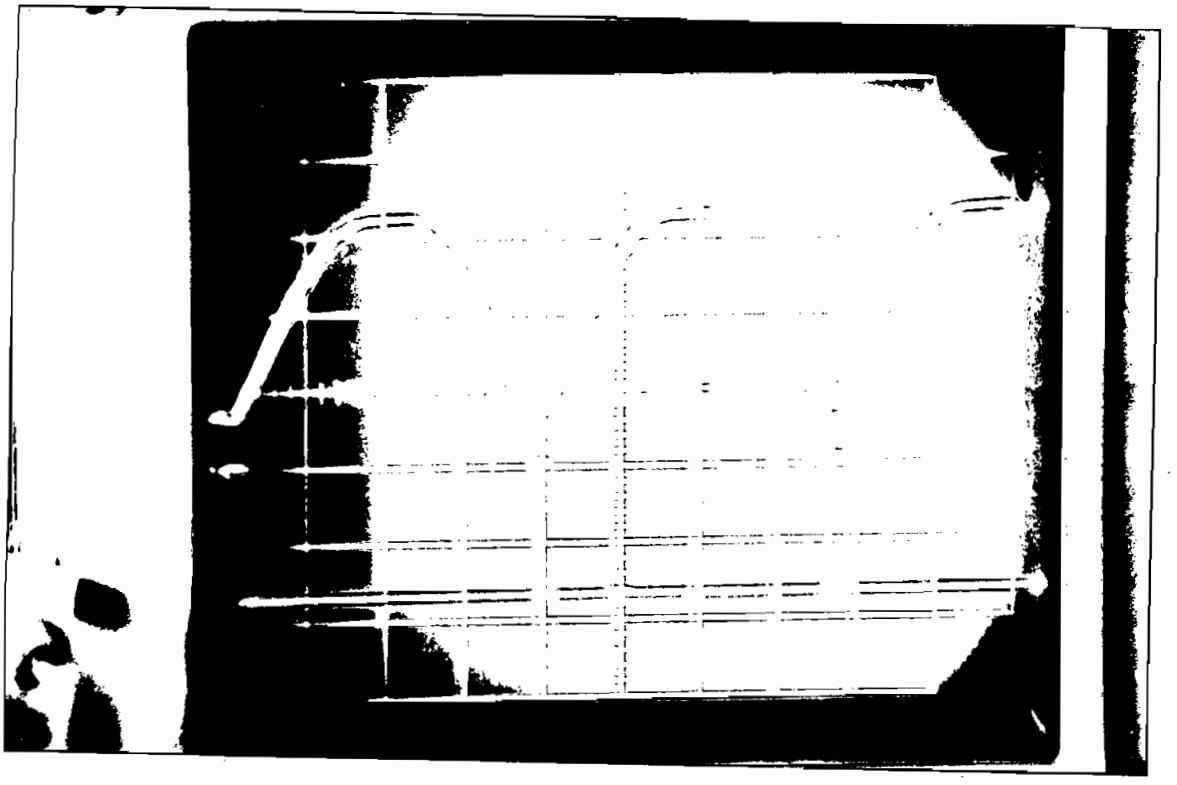
4.1.3.- Pruebas de Reloj sincronizado con la red.

Nuevamente en este caso el instrumento indispensable para la comprobación de las señales es el osciloscopio

En primer lugar se toman muestras de la señal a la salida del diodo intercalado después del rectificador de media onda comprobándose que la señal rectificada de onda completa se la toma

en forma correcta como muestra para alimentarla al diodo Zener que se encarga de transformar a niveles TTL la señal muestreada.

Esta señal se conecta a la base el un transistor el cual permanece saturado la mayor parte del tiempo, excepto cuando el valor de la señal alterna es muy cercana a cero, con lo cual se detectan los cruces por cero de la señal, tal como aparecen en la siguiente fotografía (6):



Fotografía 6. Señales de muestra y pulsos para la interrupción externa 1.

Mediante el Osciloscopio se comprobó esta señal y su correcta sincronización con la muestra de señal rectificada, tal como puede apreciarse en la siguiente fotografía, en la parte superior aparece la señal rectificada de media onda, mientras que en la parte inferior aparecen los pulsos que están conectados a la entrada de la interrupción externa 1, que es la encargada de la detección de la señal entrante y la generación de la señal correspondiente.

Estos pulsos se conectan a la entrada de la interrupción externa 1 del Microcontrolador, el cual

en respuesta ejecuta la rutina de atención a esa interrupción.

La rutina se encarga de sacar la señal digital a través de las salidas P1.1, P1.2 del microcontrolador, verificándose que la señal coincide en el tiempo y en frecuencia con la señal de AC, que en todo caso es del doble de la frecuencia de la red comercial, esto es alrededor de los 120 Hz. debido a la forma en que se han tomado las muestras.

En definitiva la señal digital sincronizada con la entrada se la consigue de la forma esperada. Es necesario indicar que dado que la señal de muestra origina un señal que alimenta a la interrupción externa 1 para evitar que en todo momento se este alimentando a la interrupción se usa un Switch que permite el paso de la señal de muestra cuando esta conectado y la bloquea en el otro estado.

4.2.- Pruebas de Aplicación.

4.2.-1.- Pruebas de Conversión de Binario a Hexadecimal.

Para realizar la comprobación del correcto funcionamiento de esta función se implementa mediante un Banco de Dip Switches de modo que existan 16 entradas que pueden tener los dos estados 0L o 1L, en función del estado de cada dip switch.

Con todos los dip switchs colocados a tierra el valor capturado por el equipo y una vez procesado lo despliega en pantalla como HEX=0000, simultáneamente todos los leds de color verde están encendidos, indicando que cada uno tiene un valor de 0 lógico como entrada; al respecto es necesario aclarar que mientras no se conecten valores TTL válidos, esto es cero o uno lógico en las entradas los leds se mantiene apagados.

A medida que los dip switches se cambia de posición el valor mostrado a través de los leds y en la pantalla varia de acuerdo al valor que corresponde, esto permitió comprobar el correcto

funcionamiento de de esta modalidad de trabajo

Otra prueba se realiza mediante la utilización de un circuito contador armado mediante un circuito integrado 74LS 191 y alimentado como señal de reloj mediante una onda generada por un temporizador 555, las salidas de este contador se conectan a las entradas del equipo con lo cual se verifica en forma dinámica el funcionamiento de la conversión desde datos binarios a hexadecimal, durante las pruebas los leds cambia de acuerdo al valor de las entradas.

Si bien la frecuencia de conteo que puede capturarse esta limitada por el tiempo que necesita el equipo para realizar los procesos de captura, transformación y despliegue de resultados, esta frecuencia normalmente será mucho mayor a aquella otra frecuencia que limita este proceso, y que es la determinada por la utilidad de los datos que pueden observarse por parte de los usuarios.

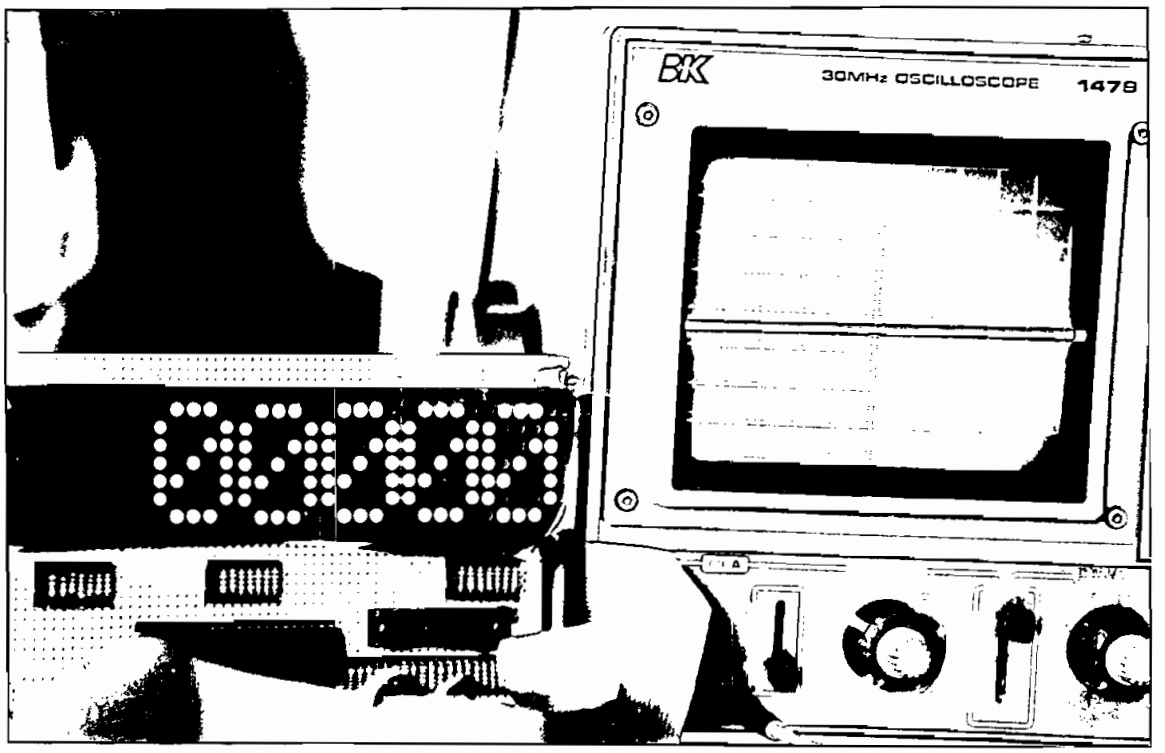
En otras palabras no resulta de utilidad realizar la captura y transformación, si los resultados no pueden apreciarse adecuadamente en la pantalla de modo que resulten útiles.

4.2.2.- Pruebas de Conversión de Binario a BCD.

Las misma pruebas que se describieron en apartado anterior sirvieron para comprobar la operación correcta de esta función, con resultados positivos.

La diferencia por supuesto esta en el formato con que se despliega en la pantalla los valores, en la fotografía 7 se puede apreciar el valor inicial, cuando todas las entradas están en cero lógico.

Por el contrario cuando todas las entradas están en uno lógico, el valor más alto que se despliega es 65535.



Fotografia 7. **Formato en el que aparecen los datos en representación BCD.**

4.2.3.- Pruebas de Control de cargas A.C.

Teniendo presente en todo momento que se trata de un equipo con finalidades demostrativas las cargas de corriente alterna que se usan para las pruebas se limitaran a un consumo máximo de 2 Amperios, en base a esto se utilizo una lámpara de 30 vatios y un radio como cargas de corriente alterna para estas pruebas.

Una vez conectadas, al presionar el teclado de acuerdo al orden establecido por el programa de ejemplo se consigue en forma eficiente el encendido y apagado de las cargas con lo que se comprueba el correcto funcionamiento de los optoacopladores y los TRIACS de la etapa de potencia.

4.2.4.- Pruebas de Comunicación Serial y lectura de RAM.

Para el caso de la comunicación serial es necesario tener muy en cuenta el valor con el que debe recargarse el temporizador que suministra la frecuencia de reloj, como prueba se altero en un dígito este valor con lo cual el microprocesador a pesar de recibir las señales físicas de la transmisión, no puede interpretarlas correctamente desde el programa encargado de tal función con lo cual se frustra la comunicación.

Mediante osciloscopio se realizo un seguimiento de las señales de la comunicación serial comprobrandose que la etapa de interface usando el C.I. MAX 232 realiza en forma eficiente el cambio de niveles desde la norma RS 232 propiamente dicha a niveles TTL usados por el microcontrolador.

Para el caso de la ejecución desde RAM del programa, se descargo un programa que activa (1L) la salida P1.0 por un intervalo de tiempo y la desactiva por otro período y comprobando mediante el osciloscopio su correcto funcionamiento.

La comprobación de la función de lectura de RAM se la realiza descargando un programa pequeño en la zona de memoria RAM desde un computador personal y posteriormente utilizando la rutina que permite leer el contenido de esta zona de memoria y desplegar su valor en el display. Basta entonces contar con una copia de el programa en hexadecimal para realizar la comprobación de que los códigos contenidos en la memoria de datos coincide con los códigos del programa. Simultáneamente entonces se ha comprobado que la rutina que permite la lectura de esta zona de memoria funciona según lo esperado.

Es necesario recalcar que se puede avanzar paso a paso en la lectura del contenido de la memoria y también retroceder, sin embargo dado que no existe la sentencia para decrementar el DPTR, que es el registro que se usa como puntero para esta rutina, solo se ha logrado un número limitado de pasos hacia atrás en la lectura de la memoria de datos.

Además el programa permite modificar los valores hexadecimales contenidos en las localidades de memoria a partir de la dirección 2000H, para esto se presiona la tecla 9, e inmediatamente desde el teclado se introduce el valor hexadecimal necesario, resulta pertinente sin embargo aclarar que se deben digitar los dos valores, por ejemplo, si la localidad 2010 contiene el valor AFH y se lo quiere cambiar por ABH 2010=AF

Será necesario presionar las dos teclas en orden primero A y luego B, después la lectura de los datos continua normalmente, y para realizar un nuevo cambio se deberá presionar la tecla 9 como paso inicial, toda esta forma de operación pudo comprobarse en forma satisfactoria.

4.3.- EVALUACIÓN DE RESULTADOS.

En primer lugar es necesario recalcar que como el desarrollo de cualquier prototipo cada una de las funciones se las va probando para verificar su adecuado funcionamiento y en la etapa final lo que se verifica es que la interacción entre diferentes rutinas resulte armoniosa, esto es no se produzcan alteraciones debido a valores en variables que afecten a otras rutinas, durante el presente proyecto todos estos pasos se completaron en forma exitosa.

Como evaluación de las pruebas y los resultados obtenidos podemos expresar que se han logrado cumplir con los objetivos planteados en cuanto a funcionalidades del equipo, determinándose por supuesto que existirán limitaciones propias de las capacidades de procesamiento del hardware y posiblemente en algún momento de la forma de programación usada.

Sin embargo dadas las bondades de un sistema basado en microprocesador este es susceptible de someterse a un proceso de optimización que permita mejorar su desempeño, o en muchos casos modificar a nivel de software para obtener más provecho con nuevas funcionalidades en base al hardware existente

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES.

A lo largo del presente trabajo se han presentado los aspectos mas destacados involucrados en el diseño e implementación del equipo didáctico basado en el microcontrolador de la familia MCS 51, a nivel de prototipo la evaluación de su funcionamiento satisface las expectativas planteadas en cuanto a contar con un equipo que permita realizar la captura de señales digitales binarias y presentación de los resultados en formatos hexadecimal o BCD, es decir con formatos que agilitan la verificación de resultados, suministrando además señales digitales a nivel de frecuencias y secuencias lógicas para pruebas de los circuitos digitales implementados, se dispone además de un medio para interactuar con elementos eléctricos que se alimentan con señales A.C., desde dispositivos que manejan señales digitales.

Los dispositivos de visualización basados en diodos emisores de luz (LEDs), se los utiliza normalmente como los medios idóneos para presentación de resultados en equipos electrónicos, y normalmente implican el uso de dispositivos tales como decodificadores ASCII, drivers específicos, circuitos integrados para la multiplexación, dispositivos de memoria para generar los caracteres; en el presente proyecto sin embargo la mayor parte de la funcionalidad de esos dispositivos ha sido substituido de forma exitosa por la capacidad de procesamiento del microcontrolador y el software correspondiente, la generación de caracteres se la realiza por software y por tanto la variedad de los mismos es muy amplia.

Un diseño cuyo núcleo central lo constituya un microcontrolador, memoria de estado solido tanto permanente para el programa como aleatoria para los datos, que utiliza como medio de interacción con los usuarios un teclado simple y un display de una sola línea con la capacidad suficiente para generar una amplia variedad de caracteres alfanuméricos, puede ser empleado en una gran variedad de aplicaciones no solo de tipo específico sino como en el presente

proyecto en el que se solicita del prototipo al menos 6 funcionalidades distintas, con ventajas en cuanto a costo respecto a equipos de propósito general como computadores personales.

El prototipo implementado utiliza como medio adecuado para incrementar el número de líneas de entrada/salida el PPI 8255, considerando los 4 puertos de entrada /salida propios del microcontrolador de la familia MCS-51y los puertos adicionales suministrados por los C.I. 8255, el número total de entradas salidas que se manejan en el presente proyecto es de 56. La interacción directa con el mundo digital una vez descontadas todas aquellas señales que se utilizan para direccionamiento de dispositivos de memoria y manejo de display, aun cuenta con un total de 32 entradas salidas que pueden ser aprovechadas para la implementación de funciones adicionales para este prototipo, sin necesidad de actuar sobre el hardware, bastara por tanto con realizar las rutinas adecuadas e incluirlas al programa actual.

El programa que controla todas y cada una de las funciones especificadas para el prototipo ha sido elaborado en lenguaje assembler, a pesar de que en el mercado existen compiladores que permitirían escribir el programa en un lenguaje de alto nivel como Basic, Pascal o C, para luego mediante el compilador generar los códigos de maquina que espera el microcontrolador. Sin embargo es conocido que un programa compilado resulta normalmente más largo y con tiempos de ejecución mayores, que aquel que se escribe directamente en el lenguaje de máquina específico del microcontrolador, por lo tanto cuando la rapidez de ejecución del código y el espacio de almacenamiento de programa disponible son factores fundamentales resulta más practico el uso de el Asembler como lenguaje de programación.

El llevar a la etapa de prototipo un diseño basado en microprocesador requiere la utilización de una serie de herramientas y equipos electrónicos, empezando por multimetro y punta lógica hasta llegar a osciloscopios y analizadores lógicos dependiendo de la complejidad del diseño y los resultados que se espere obtener. Así mismo es necesario disponer de equipos de programación de memorias EPROM y por supuesto se requiere de un computador personal con

el correspondientes software de desarrollo, depuración y simulación de acuerdo con la familia de microprocesadores con la que se trabaja. En lo referente al software en la actualidad existe una gran variedad de programas de distribución gratuita a los que es posible accesar a través de INTERNET y que comúnmente se los denomina Shareware que pueden facilitar esta parte de la tarea desarrollo, sin embargo se puede apreciar que la disponibilidad o no de los instrumentos adecuados influyen directamente en el período necesario para el desarrollo de un proyecto basado en microprocesador.

Sin embargo las ventajas que se obtienen al utilizar un microcontrolador como la base para el desarrollo de un proyecto resultan evidentes en todo momento, en especial la característica que se pone en mayor relevancia es la flexibilidad de un equipo electrónico basado en microprocesador, la capacidad de controlar e interactuar con un elevado número de entradas salidas como medio de comunicación con el mundo externo permiten variar los usos del mismo únicamente en base a cambios en la programación y no a nivel del hardware, esto elimina los limites a las aplicaciones en los que se puede emplear.

El objetivo principal planteado al enunciar este proyecto esto es contar con un sistema de ayuda didáctica para la mejor comprensión del funcionamiento de elementos de la electrónica digital se ha cumplido en su totalidad, disponiendo a demás de una herramienta para familiarizarse con técnicas de manejo de elementos de visualización mediante multiplexación por software.

A pesar de tratarse de un microcontrolador de 8 bits con el uso de rutinas adecuadas se puede realizar la manipulación de información que requiere de 16 bits para su representación, estas rutinas normalmente involucran a registro DPTR que es el único que maneja 16 bits, sin embargo y tal como se evidencia en la rutina encargada de presentar en el display el contenido de la memoria de acceso aleatorio las opciones para decrementar tal registro no son parte de las instrucciones propias de la familia MCS-51, por lo que en este caso es necesario operar sobre el registro DPTR, como dos registros de 8 bits independientes, sin embargo de lo cual no es posible obtener opciones de retorno largos.

La ventaja que presenta un microcontrolador que dispone de un puerto de comunicaciones incluido dentro de su estructura en muchas ocasiones no aparece como evidente, sin embargo si se analiza con mayor detenimiento lo complicado que pueden resultar los procesos de comunicación entre dispositivos electrónicos la opinión puede cambiar. Si una buena parte de la capacidad del procesador debe distraerse manejando las tareas de comunicación parte del potencial del equipo que esta basado en un microprocesador disminuye. Por otra parte si se toma en consideración el tamaño de los programas involucrados con las comunicaciones en el caso de microprocesadores generales este será mucho mayor en comparación con los requeridos en microprocesadores que cuenta este medio como parte integrante del circuito integrado.

Aquellas tareas que deben ser manejadas en tiempo real o lo que es lo mismo en los casos que la velocidad con la que el dispositivo puede responder ante una petición externa es un factor primordial, se evidencia de inmediato que la forma más adecuada de abordar este tipo de necesidades se basa en la utilización de las interrupciones, en el presente proyecto en especial el monitoreo continuo del teclado se constituye en un buen ejemplo de esta afirmación, el programa principal y varias de las rutinas pueden ser interrumpidas al presionar una de las teclas al menos de forma momentánea, mientras se verifica la rutina de atención para determinar si el programa debe continuar en su proceso en curso o debe realizar una tarea diferente, por lo tanto al interacción se ha realizado en forma inmediata. Este tipo de interacción puede resultar además muy útil en las opciones de control de cargas de A.C.

Considerando el hecho de que el equipo cuenta con una memoria RAM a la que puede descargarse a través del puerto serial de un computador personal un programa en lenguaje de máquina para posteriormente utilizar esta memoria como el espacio de almacenamiento del programa a ejecutarse, el prototipo ayudara al desarrollo de nuevas aplicaciones y prácticas, las cuales pueden estar relacionadas en especial con presentaciones visuales de tipo secuencial

aprovenchando la presencia de las matrices de LEDs como la base del display, inclusive, el hecho de poder modificar directamente el contenido de la memoria RAM a través del teclado del prototipo permite crear mensajes con la adecuada codificación hexadecimal de los datos suministrados.

BIBLIOGRAFIA.

1.- YERALAN S., AHLUWALIA A., **Programing and Interfacing the 8051 Microcontroller**, Adison-Wesley Publishing Company, New York, 1996.

2.- GONZALES VAZQUEZ J.A., **Introducción a los Microcontroladores. Hardware,Software y Aplicaciones**, McGraw-Hill/Interamericana. S.A., Madrid, 1992.

3.- MARTINEZ SANCHEZ V. A., **Desarrollo y Programación de Sistemas Digitales. Familia de microprocesadores INTEL MCS 51**, Adison-Wesley Iberoamericana. RA-MA, Madrid 1993.

4.- ARTHUR B. W., **Microprocesadores, Dispositivos periféricos, Optolectrónicos y de Interfaz**, McGraw-Hill, México, 1989.

5.- AXELSON Jan, **The Microcontroller Idea Book**, International Thomson Publishing, Madison USA. 1994.

6.- AVOCET Systems Inc., **AVSIM51 8051 Family Users Manual**, Rocport, Maine, 1986

7.- WHARTON J., **Microcontrolles Aplications**, Literature Departament Intel Corporation., Santa Clara. CA. 1980.

8.- INTEL Corporation, Component Data Catalog, Literature Departament Intel Corp., Santa Clara. CA., 1981.

9.- VELARDE J., BUITRON O., **Microcontroladores**, INCAITEL, Quito, 1995.

10.- MOTOROLA INC.,**Thyristor Device Data**, Technical Information Center, U.S.A. 1985.

11.- Sitios de INTERNET.

http://www.cera2.com/micro.htm

http://www.civil.mtu.edu/chipdir

http://www.circellar.com

http://www.dalsemi.com

http://www.gernsback.com

http://www.intel.com/embedded/051/index.html

http://www.intel.com/design/usb/

http://www.lvr.com/

http://www.ece.orst.edu/serv/8051

http://www.tu-bs.de/studenten/akafunk/pr8051

http://www.ece.orst.edu/~paul/8051

http://www.semiconductors.philips.com/ps/philips17.html

http://www.ee.latrobe.edu.au/postgrad/steve/8051.html

http://www.usb.org/

ANEXO 1
ESQUEMA DEL CIRCUITO ELECTRONICO

MCS® 51 MICROCONTROLLER FAMILY USER'S MANUAL

ORDER NO.: 272383-002
FEBRUARY 1994

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

MDS is an ordering code only and is not used as a product name or trademark of Intel Corporation.

Intel Corporation and Intel's FASTPATH are not affiliated with Kinetics, a division of Excelan, Inc. or its FASTPATH trademark or products.

*Other brands and names are the property of their respective owners.

Additional copies of this document or other Intel literature may be obtained from:

Intel Corporation
Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

MCS® 51 MICROCONTROLLER FAMILY USER'S MANUAL

CONTENTS PAGE

MCS® 51 Family of Microcontrollers Architectural Overview

1

MCS® 51 FAMILY OF MICROCONTROLLERS ARCHITECTURAL OVERVIEW

CONTENTS PAGE

INTRODUCTION

The 8051 is the original member of the MCS®-51 family, and is the core for all MCS-51 devices. The features of the 8051 core are:

- 8-bit CPU optimized for control applications
- Extensive Boolean processing (single-bit logic) capabilities
- 64K Program Memory address space
- 64K Data Memory address space
- 4K bytes of on-chip Program Memory
- 128 bytes of on-chip Data RAM
- 32 bidirectional and individually addressable I/O lines
- Two 16-bit timer/counters
- Full duplex UART
- 6-source/5-vector interrupt structure with two priority levels
- On-chip clock oscillator

The basic architectural structure of this 8051 core is shown in Figure 1.

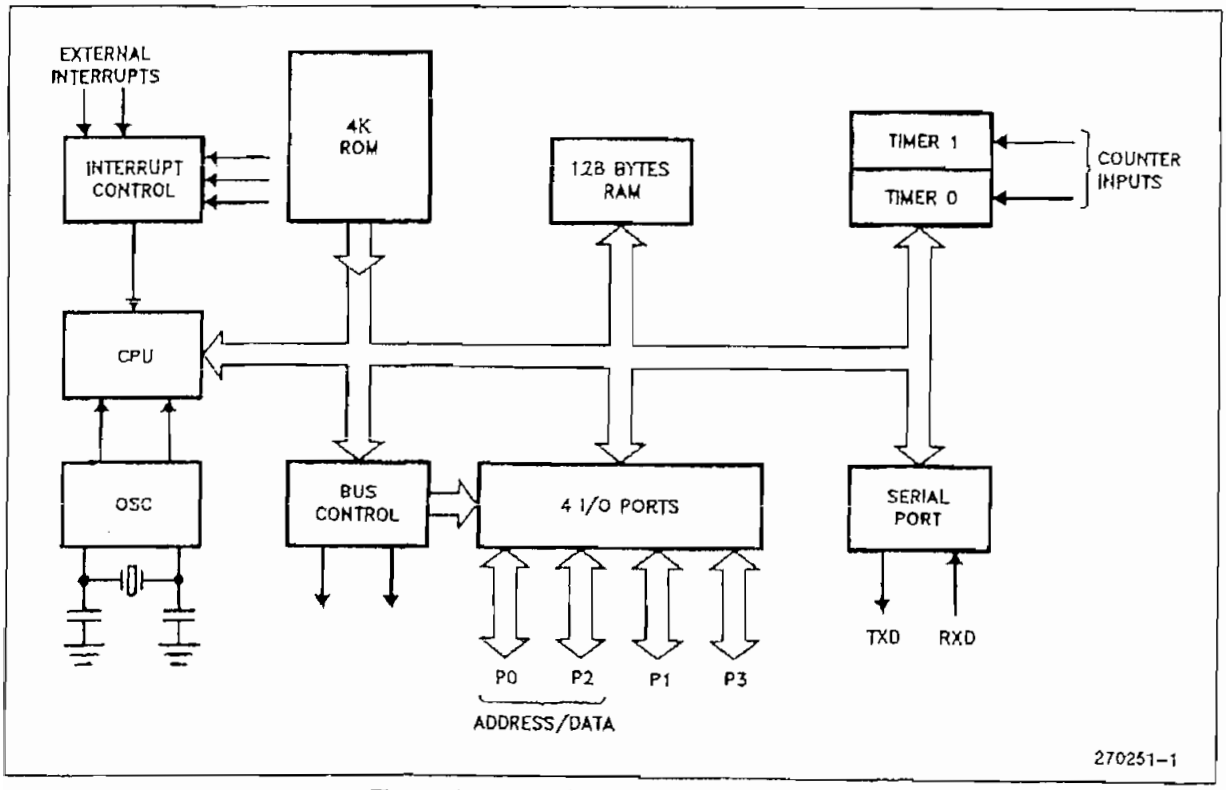


Figure 1. Block Diagram of the 8051 Core

Table 1. The MCS® 51 Family of Microcontrollers

DEVICE	ROM/EPROM (bytes)	Register RAM (bytes)	Speed (MHz)	I/O Pins	Timer/ Counters	UART	Interrupt Sources	PCA Channels	A/D Channels	SEP	GSC	DMA Channels	Lock Bits	Power Down & Idle Modes
8051 Product Line														
8031AH	ROMLESS	128	12	32	2	1	5	0	0	0	0	0	-	-
8051AH	4K ROM	128	12	32	2	1	5	0	0	0	0	0	0	-
8051AHP	4K ROM	128	12	32	2	1	5	0	0	0	0	0	P	-
8751H	4K EPROM	128	12	32	2	1	5	0	0	0	0	0	1	-
8751BH	4K EPROM	128	12	32	2	1	5	0	0	0	0	0	2	-
8052 Product Line														
8032AH	ROMLESS	256	12	32	3	1	6	0	0	0	0	0	-	-
8052AH	8K ROM	256	12	32	3	1	6	0	0	0	0	0	0	-
8752BH	8K EPROM	256	12	32	3	1	6	0	0	0	0	0	2	-
80C51 Product Line														
80C31BH	ROMLESS	128	12,16	32	2	1	5	0	0	0	0	0	-	Yes
80C51BH	4K ROM	128	12,16	32	2	1	5	0	0	0	0	0	0	Yes
80C51BHP	4K ROM	128	12,16	32	2	1	5	0	0	0	0	0	P	Yes
87C51	4K EPROM	128	12,16,20,24	32	2	1	5	0	0	0	0	0	3	Yes
8XC52/54/58 Product Line														
80C32	ROMLESS	256	12,16,20,24	32	3	1	6	0	0	0	0	0	-	Yes
80C52	8K ROM	256	12,16,20,24	32	3	1	6	0	0	0	0	0	1*	Yes
87C52	8K EPROM	256	12,16,20,24	32	3	1	6	0	0	0	0	0	3	Yes
80C54	16K ROM	256	12,16,20,24	32	3	1	6	0	0	0	0	0	1	Yes
87C54	16K EPROM	256	12,16,20,24	32	3	1	6	0	0	0	0	0	3	Yes
80C58	32K ROM	256	12,16,20,24	32	3	1	6	0	0	0	0	0	1	Yes
87C58	32K EPROM	256	12,16,20,24	32	3	1	6	0	0	0	0	0	3	Yes
8XL52/54/58 Product Line														
80L52	8K ROM	256	12,16,20*	32	3	1	6	0	0	0	0	0	1	Yes
87L52	8K OTP ROM	256	12,16,20*	32	3	1	6	0	0	0	0	0	3	Yes
80L54	16K ROM	256	12,16,20*	32	3	1	6	0	0	0	0	0	1	Yes
87L54	16K OTP ROM	256	12,16,20*	32	3	1	6	0	0	0	0	0	3	Yes
80L58	32K ROM	256	12,16,20*	32	3	1	6	0	0	0	0	0	1	Yes
87L58	32K OTP ROM	256	12,16,20*	32	3	1	6	0	0	0	0	0	3	Yes

Table 1. The MCS® 51 Family of Microcontrollers

DEVICE	ROM/EPROM (bytes)	Register RAM (bytes)	Speed (MHz)	I/O Pins	Timer/ Counters	UART	Interrupt Sources	PCA Channels	A/D Channels	SEP	GSC	DMA Channels	Lock Bits	Power Down & Idle Modes
8XC51FA/FB/FC Product Line														
80C51FA	ROMLESS	256	12,16	32	3	1	7	5	0	0	0	0	-	Yes
83C51FA	8K ROM	256	12,16	32	3	1	7	5	0	0	0	0	0	Yes
87C51FA	8K EPROM	256	12,16,20,24	32	3	1	7	5	0	0	0	0	3	Yes
83C51FB	16K ROM	256	12,16,20,24	32	3	1	7	5	0	0	0	0	1	Yes
87C51FB	16K EPROM	256	12,16,20,24	32	3	1	7	5	0	0	0	0	3	Yes
83C51FC	32K ROM	256	12,16,20,24	32	3	1	7	5	0	0	0	0	1	Yes
87C51FC	32K EPROM	256	12,16,20,24	32	3	1	7	5	0	0	0	0	3	Yes
8XL51FA/FB/FC Product Line														
80L51FA	ROMLESS	256	12,16,20*	32	3	1	7	5	0	0	0	0	-	Yes
83L51FA	8K ROM	256	12,16,20*	32	3	1	7	5	0	0	0	0	1	Yes
87L51FA	8K OTP ROM	256	12,16,20*	32	3	1	7	5	0	0	0	0	3	Yes
83L51FB	16K ROM	256	12,16,20*	32	3	1	7	5	0	0	0	0	1	Yes
87L51FB	16K OTP ROM	256	12,16,20*	32	3	1	7	5	0	0	0	0	3	Yes
83L51FC	32K ROM	256	12,16,20*	32	3	1	7	5	0	0	0	0	1	Yes
87L51FC	32K OTP ROM	256	12,16,20*	32	3	1	7	5	0	0	0	0	3	Yes
8XC51GX Product Line														
80C51GB	ROMLESS	256	12,16	48	3	1	15	10	8	1	0	0	-	Yes
83C51GB	8K ROM	256	12,16	48	3	1	15	10	8	1	0	0	1	Yes
87C51GB	8K EPROM	256	12,16	48	3	1	15	10	8	1	0	0	3	Yes
8XC152 Product Line*														
80C152JA	ROMLESS	256	16.5	40	2	1	11	0	0	1	1	2	-	Yes
80C152JB	ROMLESS	256	16.5	56	2	1	11	0	0	1	1	2	-	Yes
83C152JA	8K ROM	256	16.5	40	2	1	11	0	0	1	1	2	0	Yes
8XC51SL Product Line*														
80C51SL-BG	ROMLESS	256	16	24	2	1	10	0	4	0	1	0	-	Yes
81C51SL-BG	8K *ROM	256	16	24	2	1	10	0	4	0	1	0	0	Yes
83C51SL-BG	8K ROM	256	16	24	2	1	10	0	4	0	1	0	0	Yes
80C51SLAH	ROMLESS	256	16	24	2	1	10	0	4	0	1	0	-	Yes
81C51SLAH	16K *ROM	256	16	24	2	1	10	0	4	0	1	0	0	Yes
83C51SLAH	16K ROM	256	16	24	2	1	10	0	4	0	1	0	0	Yes
87C51SLAH	16K EPROM	256	16	24	2	1	10	0	4	0	1	0	0	Yes
80C51SLAL	ROMLESS	256	16	24	2	1	10	0	4	0	1	0	-	Yes
81C51SLAL	16K *ROM	256	16	24	2	1	10	0	4	0	1	0	0	Yes
83C51SLAL	16K ROM	256	16	24	2	1	10	0	4	0	1	0	0	Yes
87C51SLAL	16K EPROM	256	16	24	2	1	10	0	4	0	1	0	0	Yes

ROM/OTP ROM/EPROM (bytes): *ROM = SystemSoft Standard BIOS
Speed (Mhz): 24! = 24 MHz Internal-only operation
20* = 20MHz Available for Commerical Temperature Range Only
Lock Bits: 1* = 1 Lock Bit for 20MHz & 24MHz parts, no Lock Bit for 12 & 16MHz parts
P = Program verification disabled, external memory access limited to 4K
8XC152 Product Line* = Communication Controller
8XC51SL Product Line* = Keyboard Controller

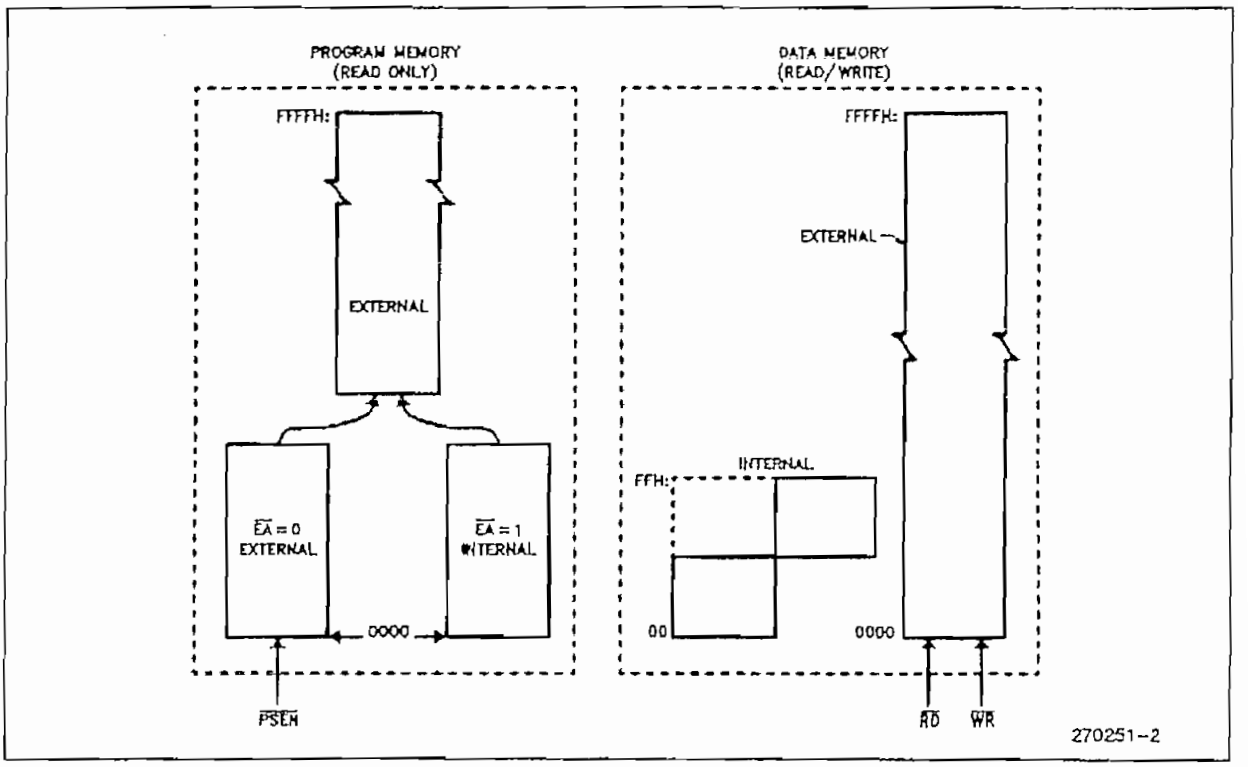


Figure 2. MCS®-51 Memory Structure

CHMOS Devices

Functionally, the CHMOS devices (designated with "C" in the middle of the device name) are all fully compatible with the 8051, but being CMOS, draw less current than an HMOS counterpart. To further exploit the power savings available in CMOS circuitry, two reduced power modes are added:

- Software-invoked Idle Mode, during which the CPU is turned off while the RAM and other on-chip peripherals continue operating. In this mode, current draw is reduced to about 15% of the current drawn when the device is fully active.
- Software-invoked Power Down Mode, during which all on-chip activities are suspended. The on-chip RAM continues to hold its data. In this mode the device typically draws less than 10 μA.

Although the 80C51BH is functionally compatible with its HMOS counterpart, specific differences between the two types of devices must be considered in the design of an application circuit if one wishes to ensure complete interchangeability between the HMOS and CHMOS devices. These considerations are discussed in the Application Note AP-252, "Designing with the 80C51BH".

For more information on the individual devices and features listed in Table 1, refer to the Hardware Descriptions and Data Sheets of the specific device.

MEMORY ORGANIZATION IN MCS®-51 DEVICES

Logical Separation of Program and Data Memory

All MCS-51 devices have separate address spaces for Program and Data Memory, as shown in Figure 2. The logical separation of Program and Data Memory allows the Data Memory to be accessed by 8-bit addresses, which can be more quickly stored and manipulated by an 8-bit CPU. Nevertheless, 16-bit Data Memory addresses can also be generated through the DPTR register.

Program Memory can only be read, not written to. There can be up to 64K bytes of Program Memory. In the ROM and EPROM versions of these devices the lowest 4K, 8K or 16K bytes of Program Memory are provided on-chip. Refer to Table 1 for the amount of on-chip ROM (or EPROM) on each device. In the ROMless versions all Program Memory is external. The read strobe for external Program Memory is the signal $\overline{PSEN}$ (Program Store Enable).

Data Memory occupies a separate address space from Program Memory. Up to 64K bytes of external RAM can be addressed in the external Data Memory space. The CPU generates read and write signals, $\overline{RD}$ and $\overline{WR}$, as needed during external Data Memory accesses.

External Program Memory and external Data Memory may be combined if desired by applying the $\overline{RD}$ and $\overline{PSEN}$ signals to the inputs of an AND gate and using the output of the gate as the read strobe to the external Program/Data memory.

Program Memory

Figure 3 shows a map of the lower part of the Program Memory. After reset, the CPU begins execution from location 0000H.

As shown in Figure 3, each interrupt is assigned a fixed location in Program Memory. The interrupt causes the CPU to jump to that location, where it commences execution of the service routine. External Interrupt 0, for example, is assigned to location 0003H. If External Interrupt 0 is going to be used, its service routine must begin at location 0003H. If the interrupt is not going to be used, its service location is available as general purpose Program Memory.

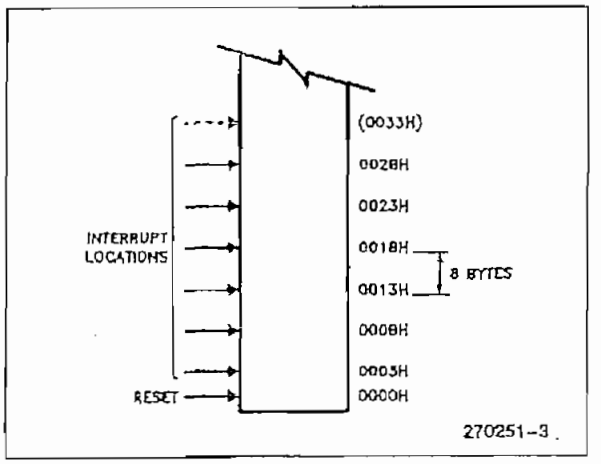


Figure 3. MCS®-51 Program Memory

The interrupt service locations are spaced at 8-byte intervals: 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1, 001BH for Timer 1, etc. If an interrupt service routine is short enough (as is often the case in control applications), it can reside entirely within that 8-byte interval. Longer service routines can use a jump instruction to skip over subsequent interrupt locations, if other interrupts are in use.

The lowest 4K (or 8K or 16K) bytes of Program Memory can be either in the on-chip ROM or in an external ROM. This selection is made by strapping the $\overline{EA}$ (External Access) pin to either V_{CC} or V_{SS}.

In the 4K byte ROM devices, if the $\overline{EA}$ pin is strapped to V_{CC}, then program fetches to addresses 0000H through 0FFFH are directed to the internal ROM. Program fetches to addresses 1000H through FFFFH are directed to external ROM.

In the 8K byte ROM devices, $\overline{EA} = V_{CC}$ selects addresses 0000H through 1FFFH to be internal, and addresses 2000H through FFFFH to be external.

In the 16K byte ROM devices, $\overline{EA} = V_{CC}$ selects addresses 0000H through 3FFFH to be internal, and addresses 4000H through FFFFH to be external.

If the $\overline{EA}$ pin is strapped to V_{SS}, then all program fetches are directed to external ROM. The ROMless parts must have this pin externally strapped to V_{SS} to enable them to execute properly.

The read strobe to external ROM, $\overline{PSEN}$, is used for all external program fetches. $\overline{PSEN}$ is not activated for internal program fetches.

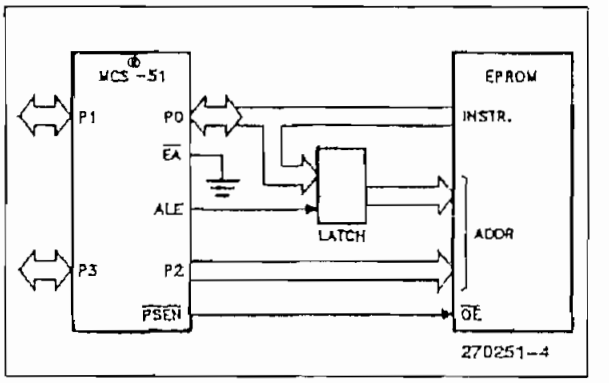


Figure 4. Executing from External Program Memory

The hardware configuration for external program execution is shown in Figure 4. Note that 16 I/O lines (Ports 0 and 2) are dedicated to bus functions during external Program Memory fetches. Port 0 (P0 in Figure 4) serves as a multiplexed address/data bus. It emits the low byte of the Program Counter (PCL) as an address, and then goes into a float state awaiting the arrival of the code byte from the Program Memory. During the time that the low byte of the Program Counter is valid on P0, the signal ALE (Address Latch Enable) clocks this byte into an address latch. Meanwhile, Port 2 (P2 in Figure 4) emits the high byte of the Program Counter (PCH). Then $\overline{PSEN}$ strobes the EPROM and the code byte is read into the microcontroller.

Program Memory addresses are always 16 bits wide, even though the actual amount of Program Memory used may be less than 64K bytes. External program execution sacrifices two of the 8-bit ports, P0 and P2, to the function of addressing the Program Memory.

Data Memory

The right half of Figure 2 shows the internal and external Data Memory spaces available to the MCS-51 user.

Figure 5 shows a hardware configuration for accessing up to 2K bytes of external RAM. The CPU in this case is executing from internal ROM. Port 0 serves as a multiplexed address/data bus to the RAM, and 3 lines of Port 2 are being used to page the RAM. The CPU generates $\overline{RD}$ and $\overline{WR}$ signals as needed during external RAM accesses.

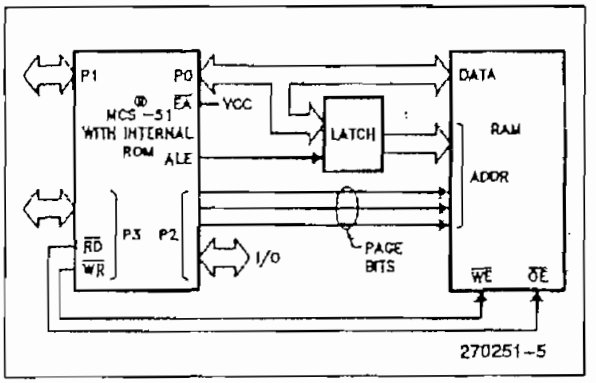


Figure 5. Accessing External Data Memory. If the Program Memory is Internal, the Other Bits of P2 are Available as I/O.

There can be up to 64K bytes of external Data Memory. External Data Memory addresses can be either 1 or 2 bytes wide. One-byte addresses are often used in conjunction with one or more other I/O lines to page the RAM, as shown in Figure 5. Two-byte addresses can also be used, in which case the high address byte is emitted at Port 2.

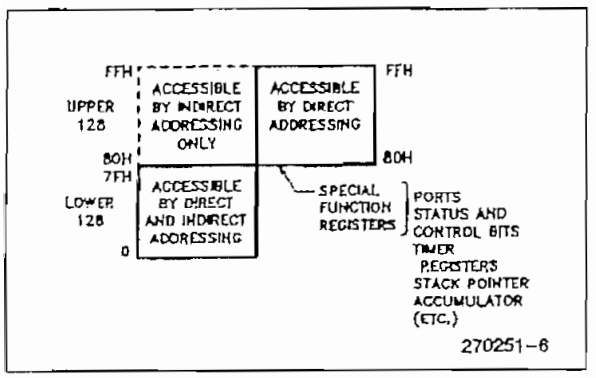


Figure 6. Internal Data Memory

Internal Data Memory is mapped in Figure 6. The memory space is shown divided into three blocks, which are generally referred to as the Lower 128, the Upper 128, and SFR space.

Internal Data Memory addresses are always one byte wide, which implies an address space of only 256 bytes. However, the addressing modes for internal RAM can in fact accommodate 384 bytes, using a simple trick. Direct addresses higher than 7FH access one memory space, and indirect addresses higher than 7FH access a different memory space. Thus Figure 6 shows the Upper 128 and SFR space occupying the same block of addresses, 80H through FFH, although they are physically separate entities.

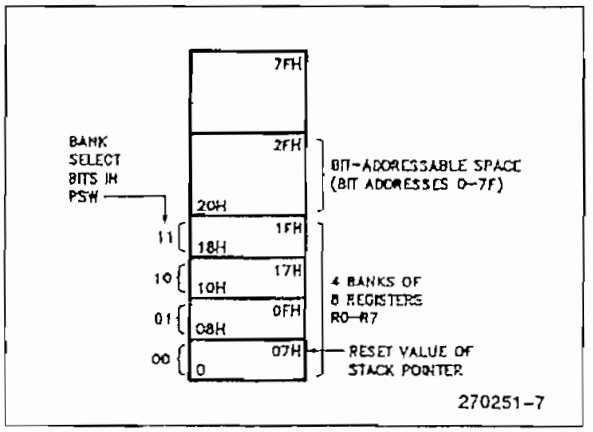


Figure 7. The Lower 128 Bytes of Internal RAM

The Lower 128 bytes of RAM are present in all MCS-51 devices as mapped in Figure 7. The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing.

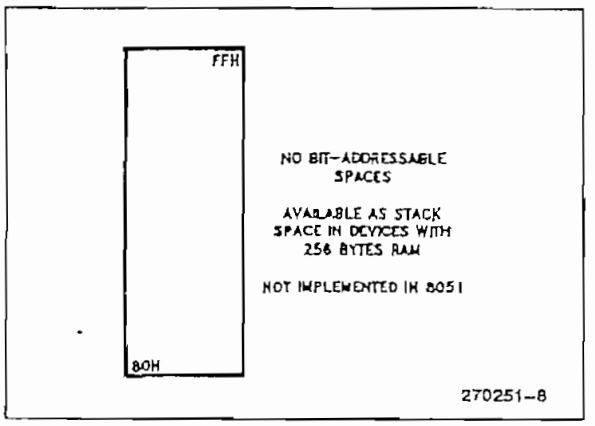


Figure 8. The Upper 128 Bytes of Internal RAM

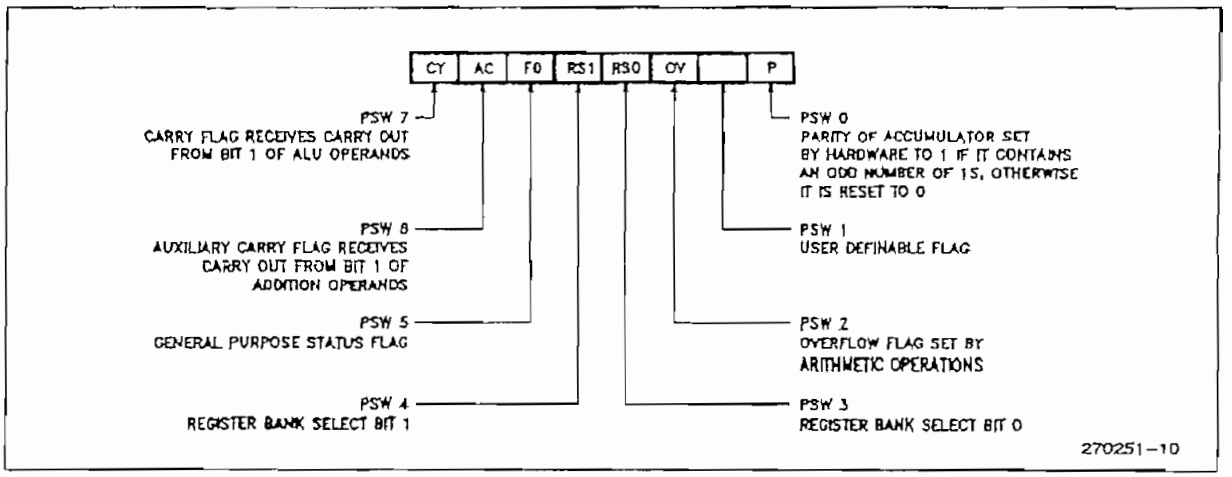


Figure 10. PSW (Program Status Word) Register in MCS®-51 Devices

The next 16 bytes above the register banks form a block of bit-addressable memory space. The MCS-51 instruction set includes a wide selection of single-bit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00H through 7FH.

All of the bytes in the Lower 128 can be accessed by either direct or indirect addressing. The Upper 128 (Figure 8) can only be accessed by indirect addressing. The Upper 128 bytes of RAM are not implemented in the 8051, but are in the devices with 256 bytes of RAM. (See Table 1).

Figure 9 gives a brief look at the Special Function Register (SFR) space. SFRs include the Port latches, timers, peripheral controls, etc. These registers can only be accessed by direct addressing. In general, all MCS-51 microcontrollers have the same SFRs as the 8051, and at the same addresses in SFR space. However, enhancements to the 8051 have additional SFRs that are not present in the 8051, nor perhaps in other proliferations of the family.

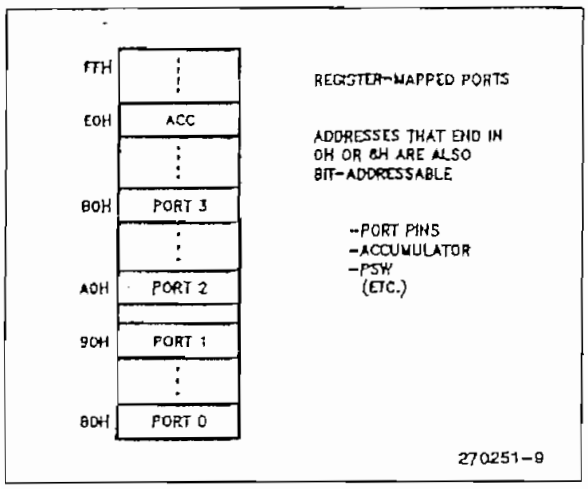


Figure 9. SFR Space

Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 000B. The bit addresses in this area are 80H through FFH.

THE MCS®-51 INSTRUCTION SET

All members of the MCS-51 family execute the same instruction set. The MCS-51 instruction set is optimized for 8-bit control applications. It provides a variety of fast addressing modes for accessing the internal RAM to facilitate byte operations on small data structures. The instruction set provides extensive support for one-bit variables as a separate data type, allowing direct bit manipulation in control and logic systems that require Boolean processing.

An overview of the MCS-51 instruction set is presented below, with a brief description of how certain instructions might be used. References to "the assembler" in this discussion are to Intel's MCS-51 Macro Assembler, ASM51. More detailed information on the instruction set can be found in the MCS-51 Macro Assembler User's Guide (Order No. 9800937 for ISIS Systems, Order No. 122752 for DOS Systems).

Program Status Word

The Program Status Word (PSW) contains several status bits that reflect the current state of the CPU. The PSW, shown in Figure 10, resides in SFR space. It contains the Carry bit, the Auxiliary Carry (for BCD operations), the two register bank select bits, the Overflow flag, a Parity bit, and two user-definable status flags.

The Carry bit, other than serving the functions of a Carry bit in arithmetic operations, also serves as the "Accumulator" for a number of Boolean operations.

The bits RS0 and RS1 are used to select one of the four register banks shown in Figure 7. A number of instructions refer to these RAM locations as R0 through R7. The selection of which of the four banks is being referred to is made on the basis of the bits RS0 and RS1 at execution time.

The Parity bit reflects the number of 1s in the Accumulator: P = 1 if the Accumulator contains an odd number of 1s, and P = 0 if the Accumulator contains an even number of 1s. Thus the number of 1s in the Accumulator plus P is always even.

Two bits in the PSW are uncommitted and may be used as general purpose status flags.

Addressing Modes

The addressing modes in the MCS-51 instruction set are as follows:

DIRECT ADDRESSING

In direct addressing the operand is specified by an 8-bit address field in the instruction. Only internal Data RAM and SFRs can be directly addressed.

INDIRECT ADDRESSING

In indirect addressing the instruction specifies a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed.

The address register for 8-bit addresses can be R0 or R1 of the selected register bank, or the Stack Pointer. The address register for 16-bit addresses can only be the 16-bit "data pointer" register, DPTR.

REGISTER INSTRUCTIONS

The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the opcode of the instruction. Instructions that access the registers this way are code efficient, since this mode eliminates an address byte. When the instruction is executed, one of the eight registers in the selected bank is accessed. One of four banks is selected at execution time by the two bank select bits in the PSW.

REGISTER-SPECIFIC INSTRUCTIONS

Some instructions are specific to a certain register. For example, some instructions always operate on the Accumulator, or Data Pointer, etc., so no address byte is needed to point to it. The opcode itself does that. Instructions that refer to the Accumulator as A assemble as accumulator-specific opcodes.

IMMEDIATE CONSTANTS

The value of a constant can follow the opcode in Program Memory. For example,

```
MOV A, #100
```

loads the Accumulator with the decimal number 100. The same number could be specified in hex digits as 64H.

INDEXED ADDRESSING

Only Program Memory can be accessed with indexed addressing, and it can only be read. This addressing mode is intended for reading look-up tables in Program Memory. A 16-bit base register (either DPTR or the Program Counter) points to the base of the table, and the Accumulator is set up with the table entry number. The address of the table entry in Program Memory is formed by adding the Accumulator data to the base pointer.

Another type of indexed addressing is used in the "case jump" instruction. In this case the destination address of a jump instruction is computed as the sum of the base pointer and the Accumulator data.

Arithmetic Instructions

The menu of arithmetic instructions is listed in Table 2. The table indicates the addressing modes that can be used with each instruction to access the <byte> operand. For example, the ADD A, <byte> instruction can be written as:

```
ADD   A,7FH    (direct addressing)
ADD   A,@R0    (indirect addressing)
ADD   A,R7     (register addressing)
ADD   A,#127   (immediate constant)
```

The execution times listed in Table 2 assume a 12 MHz clock frequency. All of the arithmetic instructions execute in 1 μs except the INC DPTR instruction, which takes 2 μs, and the Multiply and Divide instructions, which take 4 μs.

Note that any byte in the internal Data Memory space can be incremented or decremented without going through the Accumulator.

One of the INC instructions operates on the 16-bit Data Pointer. The Data Pointer is used to generate 16-bit addresses for external memory, so being able to increment it in one 16-bit operation is a useful feature.

The MUL AB instruction multiplies the Accumulator by the data in the B register and puts the 16-bit product into the concatenated B and Accumulator registers.

Table 2. A List of the MCS®-51 Arithmetic Instructions

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
ADD A,<byte>	A = A + <byte>	X	X	X	X	1
ADDC A,<byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A,<byte>	A = A − <byte> − C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A − 1	Accumulator only				1
DEC <byte>	<byte> = <byte> − 1	X	X	X		1
MUL AB	B:A = B x A	ACC and B only				4
DIV AB	A = Int [A/B] B = Mod [A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

The DIV AB instruction divides the Accumulator by the data in the B register and leaves the 8-bit quotient in the Accumulator, and the 8-bit remainder in the B register.

Oddly enough, DIV AB finds less use in arithmetic "divide" routines than in radix conversions and programmable shift operations. An example of the use of DIV AB in a radix conversion will be given later. In shift operations, dividing a number by 2^n shifts its n bits to the right. Using DIV AB to perform the division completes the shift in 4 μs and leaves the B register holding the bits that were shifted out.

The DA A instruction is for BCD arithmetic operations. In BCD arithmetic, ADD and ADDC instructions should always be followed by a DA A operation, to ensure that the result is also in BCD. Note that DA A will not convert a binary number to BCD. The DA A operation produces a meaningful result only as the second step in the addition of two BCD bytes.

Table 3. A List of the MCS®-51 Logical Instructions

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
ANL A,<byte>	A = A .AND. <byte>	X	X	X	X	1
ANL <byte>,A	<byte> = <byte> .AND. A	X				1
ANL <byte>,#data	<byte> = <byte> .AND. #data	X				2
ORL A,<byte>	A = A .OR. <byte>	X	X	X	X	1
ORL <byte>,A	<byte> = <byte> .OR. A	X				1
ORL <byte>,#data	<byte> = <byte> .OR. #data	X				2
XRL A,<byte>	A = A .XOR. <byte>	X	X	X	X	1
XRL <byte>,A	<byte> = <byte> .XOR. A	X				1
XRL <byte>,#data	<byte> = <byte> .XOR. #data	X				2
CRL A	A = 00H	Accumulator only				1
CPL A	A = .NOT. A	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1

Logical Instructions

Table 3 shows the list of MCS-51 logical instructions. The instructions that perform Boolean operations (AND, OR, Exclusive OR, NOT) on bytes perform the operation on a bit-by-bit basis. That is, if the Accumulator contains 00110101B and <byte> contains 01010011B, then

```
ANL     A,<byte>
```

will leave the Accumulator holding 00010001B.

The addressing modes that can be used to access the <byte> operand are listed in Table 3. Thus, the ANL A,<byte> instruction may take any of the forms

```
ANL     A,7FH       (direct addressing)
ANL     A,@R1       (indirect addressing)
ANL     A,R6        (register addressing)
ANL     A,#53H      (immediate constant)
```

All of the logical instructions that are Accumulator-specific execute in 1μs (using a 12 MHz clock). The others take 2 μs.

Note that Boolean operations can be performed on any byte in the lower 128 internal Data Memory space or the SFR space using direct addressing, without having to use the Accumulator. The XRL <byte>, #data instruction, for example, offers a quick and easy way to invert port bits, as in

```
XRL     P1,#0FFH
```

If the operation is in response to an interrupt, not using the Accumulator saves the time and effort to stack it in the service routine.

The Rotate instructions (RL A, RLC A, etc.) shift the Accumulator 1 bit to the left or right. For a left rotation, the MSB rolls into the LSB position. For a right rotation, the LSB rolls into the MSB position.

The SWAP A instruction interchanges the high and low nibbles within the Accumulator. This is a useful operation in BCD manipulations. For example, if the Accumulator contains a binary number which is known to be less than 100, it can be quickly converted to BCD by the following code:

```
MOV     B,#10
DIV     AB
SWAP    A
ADD     A,B
```

Dividing the number by 10 leaves the tens digit in the low nibble of the Accumulator, and the ones digit in the B register. The SWAP and ADD instructions move the tens digit to the high nibble of the Accumulator, and the ones digit to the low nibble.

Data Transfers

INTERNAL RAM

Table 4 shows the menu of instructions that are available for moving data around within the internal memory spaces, and the addressing modes that can be used with each one. With a 12 MHz clock, all of these instructions execute in either 1 or 2 μs.

The MOV <dest>, <src> instruction allows data to be transferred between any two internal RAM or SFR locations without going through the Accumulator. Remember the Upper 128 byes of data RAM can be accessed only by indirect addressing, and SFR space only by direct addressing.

Note that in all MCS-51 devices, the stack resides in on-chip RAM, and grows upwards. The PUSH instruction first increments the Stack Pointer (SP), then copies the byte into the stack. PUSH and POP use only direct addressing to identify the byte being saved or restored,

Table 4. A List of the MCS®-51 Data Transfer Instructions that Access Internal Data Memory Space

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
MOV A,<src>	A = <src>	X	X	X	X	1
MOV <dest>,A	<dest> = A	X	X	X		1
MOV <dest>, <src>	<dest> = <src>	X	X	X	X	2
MOV DPTR,#data16	DPTR = 16-bit immediate constant.				X	2
PUSH <src>	INC SP : MOV "@SP",<src>	X				2
POP <dest>	MOV <dest>, "@SP" : DEC SP	X				2
XCH A,<byte>	ACC and <byte> exchange data	X	X	X		1
XCHD A,@Ri	ACC and @RI exchange low nibbles		X			1

but the stack itself is accessed by indirect addressing using the SP register. This means the stack can go into the Upper 128, if they are implemented, but not into SFR space.

In devices that do not implement the Upper 128, if the SP points to the Upper 128, PUSHed bytes are lost, and POPped bytes are indeterminate.

The Data Transfer instructions include a 16-bit MOV that can be used to initialize the Data Pointer (DPTR) for look-up tables in Program Memory, or for 16-bit external Data Memory accesses.

The XCH A, <byte> instruction causes the Accumulator and addressed byte to exchange data. The XCHD A,@Ri instruction is similar, but only the low nibbles are involved in the exchange.

To see how XCH and XCHD can be used to facilitate data manipulations, consider first the problem of shifting an 8-digit BCD number two digits to the right. Figure 11 shows how this can be done using direct MOVs, and for comparison how it can be done using XCH instructions. To aid in understanding how the code works, the contents of the registers that are holding the BCD number and the content of the Accumulator are shown alongside each instruction to indicate their status after the instruction has been executed.

		2A	2B	2C	2D	2E	ACC
MOV	A,2EH	00	12	34	56	78	78
MOV	2EH,2DH	00	12	34	56	56	78
MOV	2DH,2CH	00	12	34	34	56	78
MOV	2CH,2BH	00	12	12	34	56	78
MOV	2BH,#0	00	00	12	34	56	78

(a) Using direct MOVs: 14 bytes, 9 μs

		2A	2B	2C	2D	2E	ACC
CLR	A	00	12	34	56	78	00
XCH	A,2BH	00	00	34	56	78	12
XCH	A,2CH	00	00	12	56	78	34
XCH	A,2DH	00	00	12	34	78	56
XCH	A,2EH	00	00	12	34	56	78

(b) Using XCHs: 9 bytes, 5 μs

Figure 11. Shifting a BCD Number Two Digits to the Right

After the routine has been executed, the Accumulator contains the two digits that were shifted out on the right. Doing the routine with direct MOVs uses 14 code bytes and 9 μs of execution time (assuming a 12 MHz clock). The same operation with XCHs uses less code and executes almost twice as fast.

To right-shift by an odd number of digits, a one-digit shift must be executed. Figure 12 shows a sample of code that will right-shift a BCD number one digit, using the XCHD instruction. Again, the contents of the registers holding the number and of the Accumulator are shown alongside each instruction.

			2A	2B	2C	2D	2E	ACC
	MOV	R1,#2EH	00	12	34	56	78	XX
	MOV	R0,#2DH	00	12	34	56	78	XX
	loop for R1 = 2EH:							
LOOP:	MOV	A,@R1	00	12	34	56	78	78
	XCHD	A,@R0	00	12	34	58	78	76
	SWAP	A	00	12	34	58	78	67
	MOV	@R1,A	00	12	34	58	67	67
	DEC	R1	00	12	34	58	67	67
	DEC	R0	00	12	34	58	67	67
	CJNE	R1,#2AH,LOOP						
	loop for R1 = 2DH:		00	12	38	45	67	45
	loop for R1 = 2CH:		00	18	23	45	67	23
	loop for R1 = 2BH:		08	01	23	45	67	01
	CLR	A	08	01	23	45	67	00
	XCH	A,2AH	00	01	23	45	67	08

Figure 12. Shifting a BCD Number One Digit to the Right

First, pointers R1 and R0 are set up to point to the two bytes containing the last four BCD digits. Then a loop is executed which leaves the last byte, location 2EH, holding the last two digits of the shifted number. The pointers are decremented, and the loop is repeated for location 2DH. The CJNE instruction (Compare and Jump if Not Equal) is a loop control that will be described later.

The loop is executed from LOOP to CJNE for R1 = 2EH, 2DH, 2CH and 2BH. At that point the digit that was originally shifted out on the right has propagated to location 2AH. Since that location should be left with 0s, the lost digit is moved to the Accumulator.

EXTERNAL RAM

Table 5 shows a list of the Data Transfer instructions that access external Data Memory. Only indirect addressing can be used. The choice is whether to use a one-byte address, @Ri, where Ri can be either R0 or R1 of the selected register bank, or a two-byte address, @DPTR. The disadvantage to using 16-bit addresses if only a few K bytes of external RAM are involved is that 16-bit addresses use all 8 bits of Port 2 as address bus. On the other hand, 8-bit addresses allow one to address a few K bytes of RAM, as shown in Figure 5, without having to sacrifice all of Port 2.

All of these instructions execute in 2 μs, with a 12 MHz clock.

Table 5. A List of the MCS®-51 Data Transfer Instructions that Access External Data Memory Space

Address Width	Mnemonic	Operation	Execution Time (μs)
8 bits	MOVX A,@Ri	Read external RAM @Ri	2
8 bits	MOVX @Ri,A	Write external RAM @Ri	2
16 bits	MOVX A,@DPTR	Read external RAM @DPTR	2
16 bits	MOVX @DPTR,A	Write external RAM @DPTR	2

Note that in all external Data RAM accesses, the Accumulator is always either the destination or source of the data.

The read and write strobes to external RAM are activated only during the execution of a MOVX instruction. Normally these signals are inactive, and in fact if they're not going to be used at all, their pins are available as extra I/O lines. More about that later.

LOOKUP TABLES

Table 6 shows the two instructions that are available for reading lookup tables in Program Memory. Since these instructions access only Program Memory, the lookup tables can only be read, not updated. The mnemonic is MOVC for "move constant".

If the table access is to external Program Memory, then the read strobe is $\overline{PSEN}$.

Table 6. The MCS®-51 Lookup Table Read Instructions

Mnemonic	Operation	Execution Time (μs)
MOVC A,@A+DPTR	Read Pgm Memory at (A+DPTR)	2
MOVC A,@A+PC	Read Pgm Memory at (A+PC)	2

The first MOVC instruction in Table 6 can accommodate a table of up to 256 entries, numbered 0 through 255. The number of the desired entry is loaded into the Accumulator, and the Data Pointer is set up to point to beginning of the table. Then

```
MOVC    A,@A+DPTR
```

copies the desired table entry into the Accumulator.

The other MOVC instruction works the same way, except the Program Counter (PC) is used as the table base, and the table is accessed through a subroutine. First the number of the desired entry is loaded into the Accumulator, and the subroutine is called:

```
MOV     A,ENTRY_NUMBER
CALL    TABLE
```

The subroutine "TABLE" would look like this:

```
TABLE:  MOVC    A,@A+PC
        RET
```

The table itself immediately follows the RET (return) instruction in Program Memory. This type of table can have up to 255 entries, numbered 1 through 255. Number 0 can not be used, because at the time the MOVC instruction is executed, the PC contains the address of the RET instruction. An entry numbered 0 would be the RET opcode itself.

Boolean Instructions

MCS-51 devices contain a complete Boolean (single-bit) processor. The internal RAM contains 128 addressable bits, and the SFR space can support up to 128 other addressable bits. All of the port lines are bit-addressable, and each one can be treated as a separate single-bit port. The instructions that access these bits are not just conditional branches, but a complete menu of move, set, clear, complement, OR, and AND instructions. These kinds of bit operations are not easily obtained in other architectures with any amount of byte-oriented software.

Table 7. A List of the MCS®-51 Boolean Instructions

Mnemonic		Operation	Execution Time (μs)
ANL	C,bit	C = C .AND. bit	2
ANL	C,/bit	C = C .AND. .NOT. bit	2
ORL	C,bit	C = C .OR. bit	2
ORL	C,/bit	C = C .OR. .NOT. bit	2
MOV	C,bit	C = bit	1
MOV	bit,C	bit = C	2
CLR	C	C = 0	1
CLR	bit	bit = 0	1
SETB	C	C = 1	1
SETB	bit	bit = 1	1
CPL	C	C = .NOT. C	1
CPL	bit	bit = .NOT. bit	1
JC	rel	Jump if C = 1	2
JNC	rel	Jump if C = 0	2
JB	bit,rel	Jump if bit = 1	2
JNB	bit,rel	Jump if bit = 0	2
JBC	bit,rel	Jump if bit = 1; CLR bit	2

The instruction set for the Boolean processor is shown in Table 7. All bit accesses are by direct addressing. Bit addresses 00H through 7FH are in the Lower 128, and bit addresses 80H through FFH are in SFR space.

Note how easily an internal flag can be moved to a port pin:

```
MOV     C,FLAG
MOV     P1.0,C
```

In this example, FLAG is the name of any addressable bit in the Lower 128 or SFR space. An I/O line (the LSB of Port 1, in this case) is set or cleared depending on whether the flag bit is 1 or 0.

The Carry bit in the PSW is used as the single-bit Accumulator of the Boolean processor. Bit instructions that refer to the Carry bit as C assemble as Carry-specific instructions (CLR C, etc). The Carry bit also has a direct address, since it resides in the PSW register, which is bit-addressable.

Note that the Boolean instruction set includes ANL and ORL operations, but not the XRL (Exclusive OR) operation. An XRL operation is simple to implement in software. Suppose, for example, it is required to form the Exclusive OR of two bits:

C = bit1 .XRL. bit2

The software to do that could be as follows:

```
        MOV     C,bit1
        JNB     bit2,OVER
        CPL     C
OVER:   (continue)
```

First, bit1 is moved to the Carry. If bit2 = 0, then C now contains the correct result. That is, bit1 .XRL. bit2 = bit1 if bit2 = 0. On the other hand, if bit2 = 1 C now contains the complement of the correct result. It need only be inverted (CPL C) to complete the operation.

This code uses the JNB instruction, one of a series of bit-test instructions which execute a jump if the addressed bit is set (JC, JB, JBC) or if the addressed bit is not set (JNC, JNB). In the above case, bit2 is being tested, and if bit2 = 0 the CPL C instruction is jumped over.

JBC executes the jump if the addressed bit is set, and also clears the bit. Thus a flag can be tested and cleared in one operation.

All the PSW bits are directly addressable, so the Parity bit, or the general purpose flags, for example, are also available to the bit-test instructions.

RELATIVE OFFSET

The destination address for these jumps is specified to the assembler by a label or by an actual address in Program Memory. However, the destination address assembles to a **relative** offset byte. This is a signed (two's complement) offset byte which is added to the PC in two's complement arithmetic if the jump is executed.

The range of the jump is therefore −128 to +127 Program Memory bytes relative to the first byte following the instruction.

Jump Instructions

Table 8 shows the list of unconditional jumps.

Table 8. Unconditional Jumps in MCS®-51 Devices

Mnemonic	Operation	Execution Time (μs)
JMP addr	Jump to addr	2
JMP @A+DPTR	Jump to A+DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

The Table lists a single "JMP addr" instruction, but in fact there are three—SJMP, LJMP and AJMP—which differ in the format of the destination address. JMP is a generic mnemonic which can be used if the programmer does not care which way the jump is encoded.

The SJMP instruction encodes the destination address as a relative offset, as described above. The instruction is 2 bytes long, consisting of the opcode and the relative offset byte. The jump distance is limited to a range of −128 to +127 bytes relative to the instruction following the SJMP.

The LJMP instruction encodes the destination address as a 16-bit constant. The instruction is 3 bytes long, consisting of the opcode and two address bytes. The destination address can be anywhere in the 64K Program Memory space.

The AJMP instruction encodes the destination address as an 11-bit constant. The instruction is 2 bytes long, consisting of the opcode, which itself contains 3 of the 11 address bits, followed by another byte containing the low 8 bits of the destination address. When the instruction is executed, these 11 bits are simply substituted for the low 11 bits in the PC. The high 5 bits stay the same. Hence the destination has to be within the same 2K block as the instruction following the AJMP.

In all cases the programmer specifies the destination address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the destination address into the correct format for the given instruction. If the format required by the instruction will not support the distance to the specified destination address, a "Destination out of range" message is written into the List file.

The JMP @A+DPTR instruction supports case jumps. The destination address is computed at execution time as the sum of the 16-bit DPTR register and the Accumulator. Typically, DPTR is set up with the address of a jump table, and the Accumulator is given an index to the table. In a 5-way branch, for example, an integer 0 through 4 is loaded into the Accumulator. The code to be executed might be as follows:

```
MOV     DPTR,#JUMP_TABLE
MOV     A,INDEX_NUMBER
RL      A
JMP     @A+DPTR
```

The RL A instruction converts the index number (0 through 4) to an even number on the range 0 through 8, because each entry in the jump table is 2 bytes long:

```
JUMP_TABLE:
        AJMP    CASE_0
        AJMP    CASE_1
        AJMP    CASE_2
        AJMP    CASE_3
        AJMP    CASE_4
```

Table 8 shows a single "CALL addr" instruction, but there are two of them—LCALL and ACALL—which differ in the format in which the subroutine address is given to the CPU. CALL is a generic mnemonic which can be used if the programmer does not care which way the address is encoded.

The LCALL instruction uses the 16-bit address format, and the subroutine can be anywhere in the 64K Program Memory space. The ACALL instruction uses the 11-bit format, and the subroutine must be in the same 2K block as the instruction following the ACALL.

In any case the programmer specifies the subroutine address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the address into the correct format for the given instructions.

Subroutines should end with a RET instruction, which returns execution to the instruction following the CALL.

RETI is used to return from an interrupt service routine. The only difference between RET and RETI is that RETI tells the interrupt control system that the interrupt in progress is done. If there is no interrupt in progress at the time RETI is executed, then the RETI is functionally identical to RET.

Table 9 shows the list of conditional jumps available to the MCS-51 user. All of these jumps specify the destination address by the relative offset method, and so are limited to a jump distance of −128 to +127 bytes from the instruction following the conditional jump instruction. Important to note, however, the user specifies to the assembler the actual destination address the same way as the other jumps: as a label or a 16-bit constant.

Table 9. Conditional Jumps in MCS®-51 Devices

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
JZ rel	Jump if A = 0	Accumulator only				2
JNZ rel	Jump if A ≠ 0	Accumulator only				2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNE A,<byte>,rel	Jump if A ≠ <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X		2

There is no Zero bit in the PSW. The JZ and JNZ instructions test the Accumulator data for that condition.

The DJNZ instruction (Decrement and Jump if Not Zero) is for loop control. To execute a loop N times, load a counter byte with N and terminate the loop with a DJNZ to the beginning of the loop, as shown below for N = 10:

```
        MOV     COUNTER,#10
LOOP:   (begin loop)
          .
          .
          .
        (end loop)
        DJNZ    COUNTER,LOOP
        (continue)
```

The CJNE instruction (Compare and Jump if Not Equal) can also be used for loop control as in Figure 12. Two bytes are specified in the operand field of the instruction. The jump is executed only if the two bytes are not equal. In the example of Figure 12, the two bytes were the data in R1 and the constant 2AH. The initial data in R1 was 2EH. Every time the loop was executed, R1 was decremented, and the looping was to continue until the R1 data reached 2AH.

Another application of this instruction is in "greater than, less than" comparisons. The two bytes in the operand field are taken as unsigned integers. If the first is less than the second, then the Carry bit is set (1). If the first is greater than or equal to the second, then the Carry bit is cleared.

CPU TIMING

All MCS-51 microcontrollers have an on-chip oscillator which can be used if desired as the clock source for the CPU. To use the on-chip oscillator, connect a crystal or ceramic resonator between the XTAL1 and XTAL2 pins of the microcontroller, and capacitors to ground as shown in Figure 13.

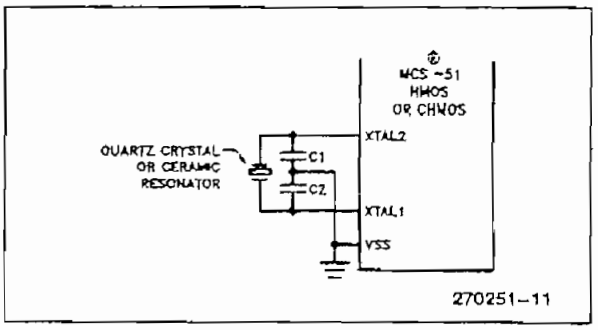


Figure 13. Using the On-Chip Oscillator

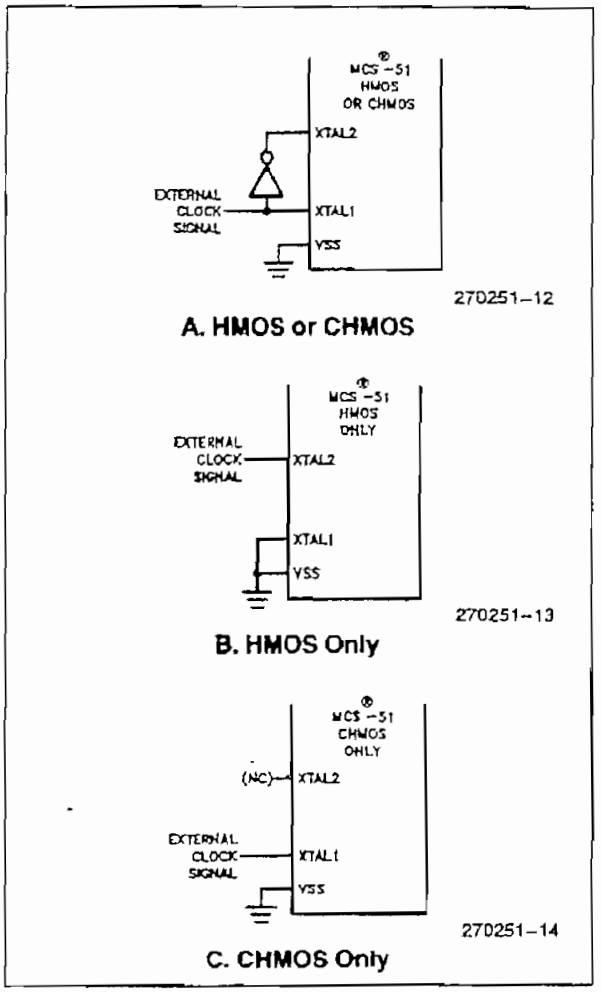


Figure 14. Using an External Clock

Examples of how to drive the clock with an external oscillator are shown in Figure 14. Note that in the HMOS devices (8051, etc.) the signal at the XTAL2 pin actually drives the internal clock generator. In the CHMOS devices (80C51BH, etc.) the signal at the XTAL1 pin drives the internal clock generator. If only one pin is going to be driven with the external oscillator signal, make sure it is the right pin.

The internal clock generator defines the sequence of states that make up the MCS-51 machine cycle.

Machine Cycles

A machine cycle consists of a sequence of 6 states, numbered S1 through S6. Each state time lasts for two oscillator periods. Thus a machine cycle takes 12 oscillator periods or 1 μs if the oscillator frequency is 12 MHz.

Each state is divided into a Phase 1 half and a Phase 2 half. Figure 15 shows the fetch/execute sequences in

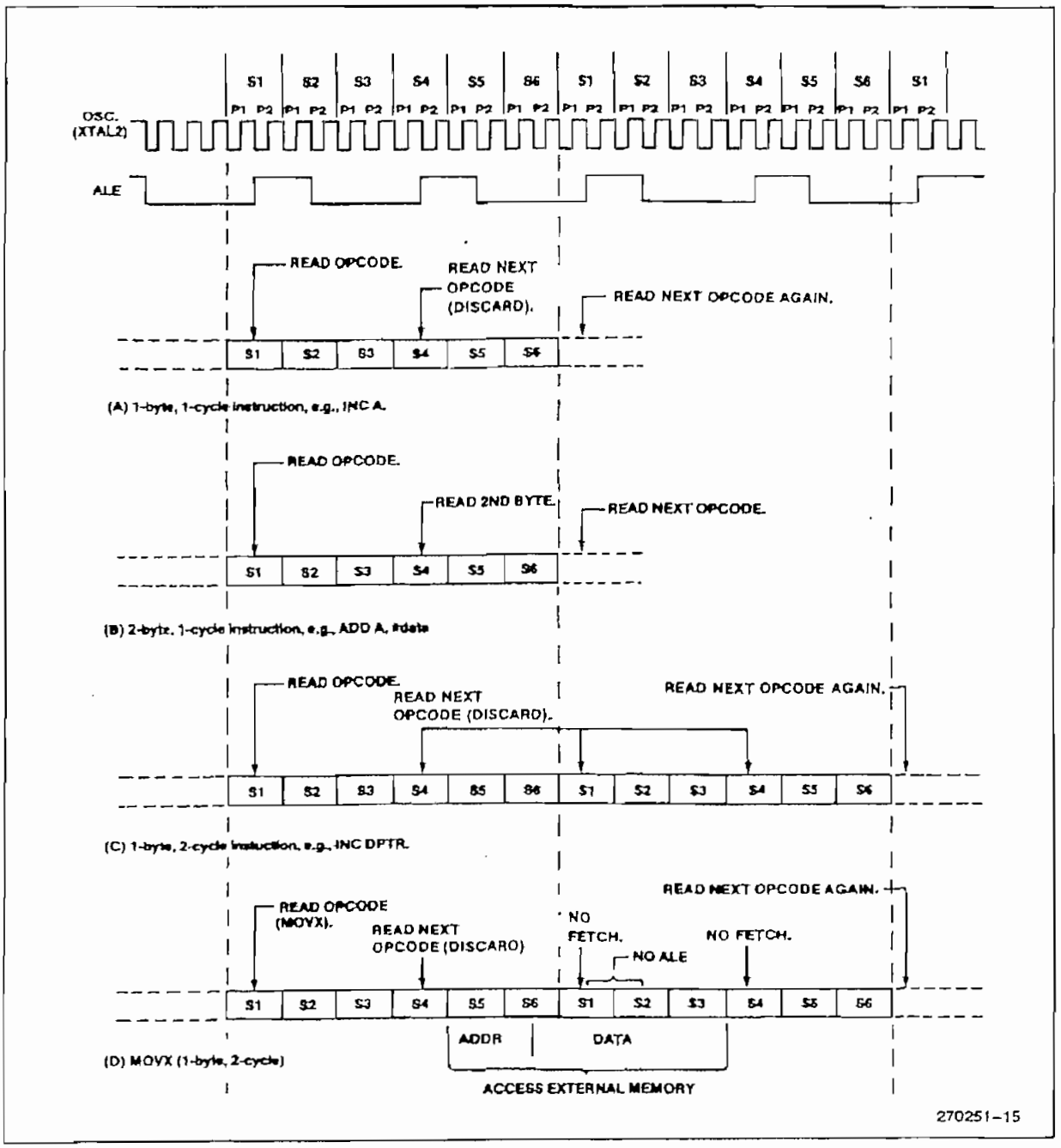


Figure 15. State Sequences in MCS®-51 Devices

states and phases for various kinds of instructions. Normally two program fetches are generated during each machine cycle, even if the instruction being executed doesn't require it. If the instruction being executed doesn't need more code bytes, the CPU simply ignores the extra fetch, and the Program Counter is not incremented.

Execution of a one-cycle instruction (Figure 15A and B) begins during State 1 of the machine cycle, when the opcode is latched into the Instruction Register. A second fetch occurs during S4 of the same machine cycle. Execution is complete at the end of State 6 of this machine cycle.

The MOVX instructions take two machine cycles to execute. No program fetch is generated during the second cycle of a MOVX instruction. This is the only time program fetches are skipped. The fetch/execute sequence for MOVX instructions is shown in Figure 15(D).

The fetch/execute sequences are the same whether the Program Memory is internal or external to the chip. Execution times do not depend on whether the Program Memory is internal or external.

Figure 16 shows the signals and timing involved in program fetches when the Program Memory is external. If Program Memory is external, then the Program Memory read strobe $\overline{PSEN}$ is normally activated twice per machine cycle, as shown in Figure 16(A).

If an access to external Data Memory occurs, as shown in Figure 16(B), two $\overline{PSEN}$s are skipped, because the address and data bus are being used for the Data Memory access.

Note that a Data Memory bus cycle takes twice as much time as a Program Memory bus cycle. Figure 16 shows the relative timing of the addresses being emitted at Ports 0 and 2, and of ALE and $\overline{PSEN}$. ALE is used to latch the low address byte from P0 into the address latch.

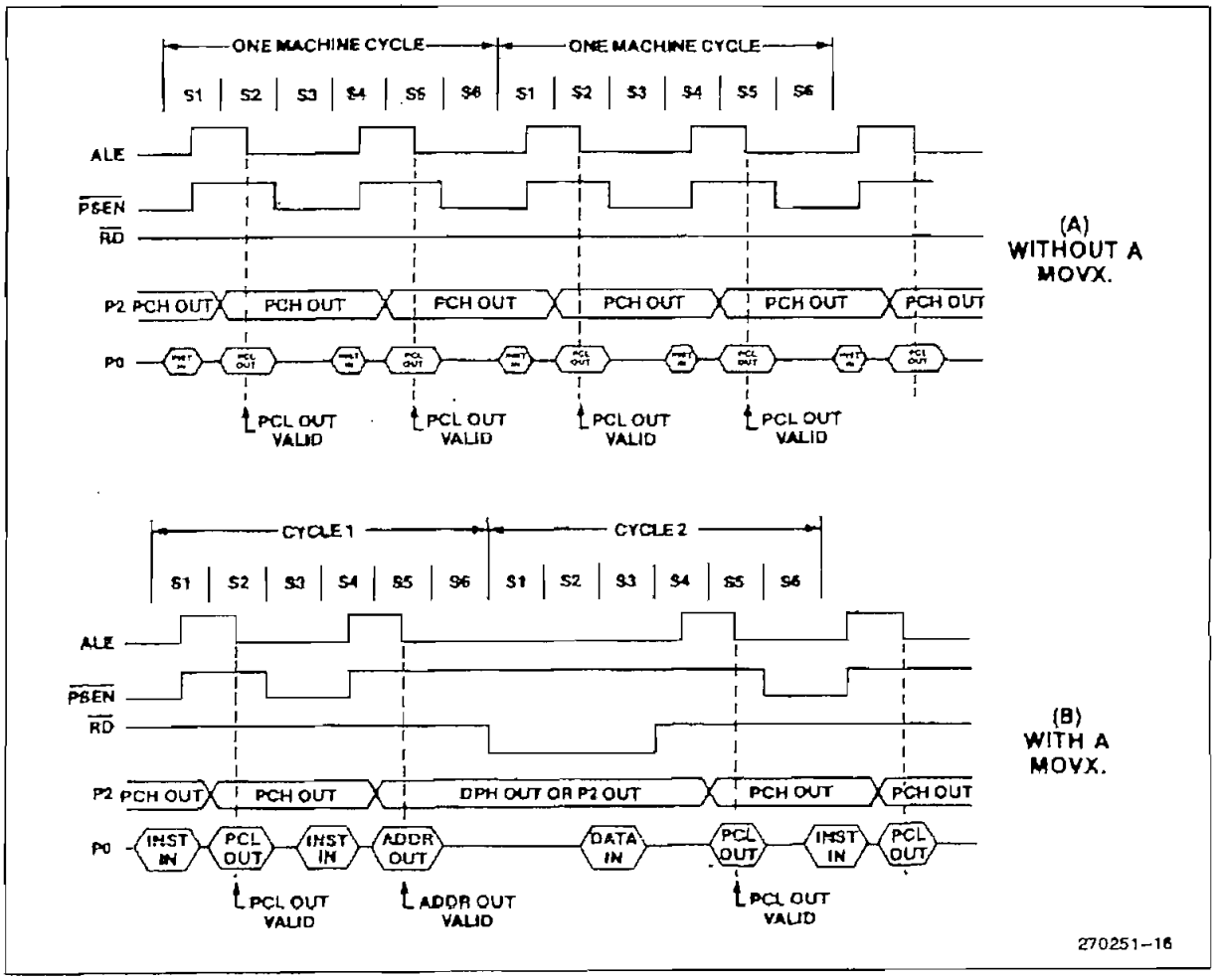


Figure 16. Bus Cycles in MCS®-51 Devices Executing from External Program Memory

When the CPU is executing from internal Program Memory, $\overline{PSEN}$ is not activated, and program addresses are not emitted. However, ALE continues to be activated twice per machine cycle and so is available as a clock output signal. Note, however, that one ALE is skipped during the execution of the MOVX instruction.

Interrupt Structure

The 8051 core provides 5 interrupt sources: 2 external interrupts, 2 timer interrupts, and the serial port interrupt. What follows is an overview of the interrupt structure for the 8051. Other MCS-51 devices have additional interrupt sources and vectors as shown in Table 1. Refer to the appropriate chapters on other devices for further information on their interrupts.

INTERRUPT ENABLES

Each of the interrupt sources can be individually enabled or disabled by setting or clearing a bit in the SFR named IE (Interrupt Enable). This register also contains a global disable bit, which can be cleared to disable all interrupts at once. Figure 17 shows the IE register for the 8051.

(MSB)							(LSB)
EA	—	—	ES	ET1	EX1	ET0	EX0

Enable bit = 1 enables the interrupt.
Enable bit = 0 disables it.

Symbol	Position	Function
EA	IE.7	disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	reserved*
—	IE.5	reserved*
ES	IE.4	Serial Port Interrupt enable bit.
ET1	IE.3	Timer 1 Overflow Interrupt enable bit.
EX1	IE.2	External Interrupt 1 enable bit.
ET0	IE.1	Timer 0 Overflow Interrupt enable bit.
EX0	IE.0	External Interrupt 0 enable bit.

*These reserved bits are used in other MCS-51 devices.

Figure 17. IE (Interrupt Enable) Register in the 8051

INTERRUPT PRIORITIES

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in the SFR named IP (Interrupt Priority). Figure 18 shows the IP register in the 8051.

A low-priority interrupt can be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

If two interrupt requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence.

Figure 19 shows, for the 8051, how the IE and IP registers and the polling sequence work to determine which if any interrupt will be serviced.

(MSB)							(LSB)
—	—	—	PS	PT1	PX1	PT0	PX0

Priority bit = 1 assigns high priority.
Priority bit = 0 assigns low priority.

Symbol	Position	Function
—	IP.7	reserved*
—	IP.6	reserved*
—	IP.5	reserved*
PS	IP.4	Serial Port interrupt priority bit.
PT1	IP.3	Timer 1 interrupt priority bit.
PX1	IP.2	External Interrupt 1 priority bit.
PT0	IP.1	Timer 0 interrupt priority bit.
PX0	IP.0	External Interrupt 0 priority bit.

*These reserved bits are used in other MCS-51 devices.

Figure 18. IP (Interrupt Priority) Register in the 8051

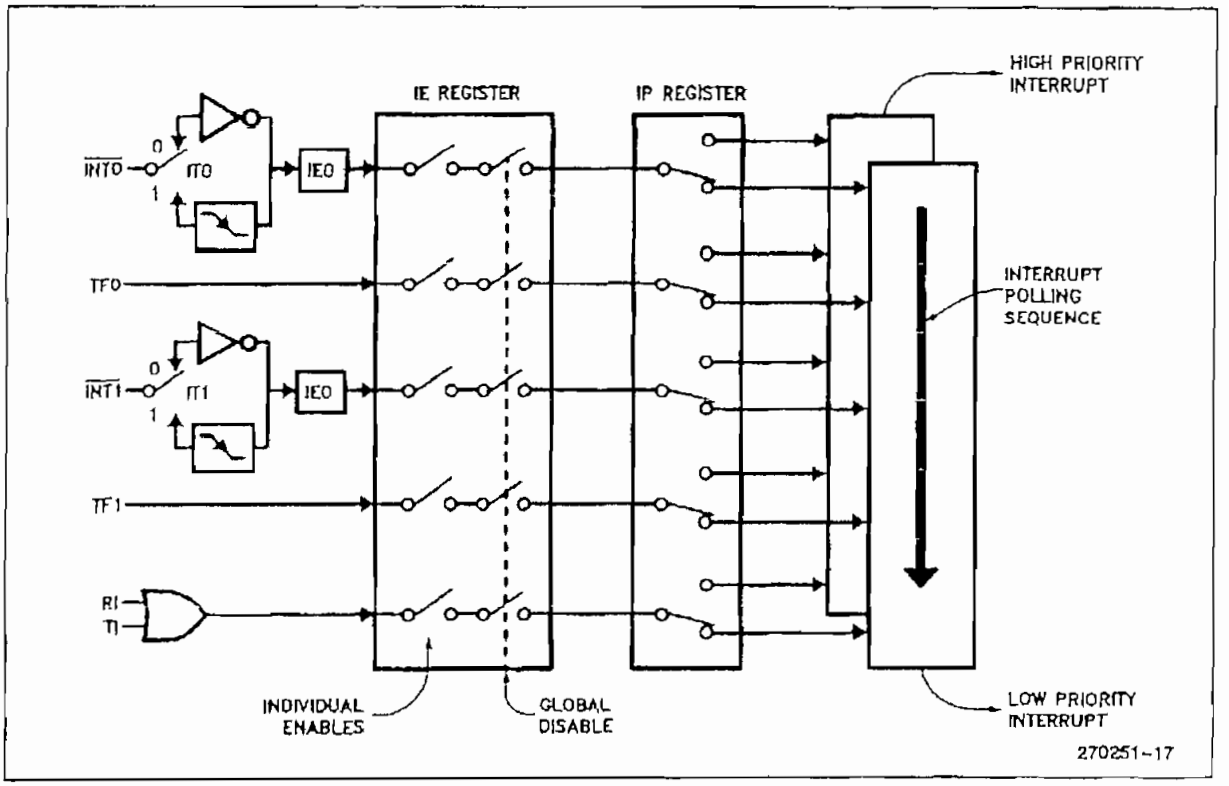


Figure 19. 8051 Interrupt Control System

In operation, all the interrupt flags are latched into the interrupt control system during State 5 of every machine cycle. The samples are polled during the following machine cycle. If the flag for an enabled interrupt is found to be set (1), the interrupt system generates an LCALL to the appropriate location in Program Memory, unless some other condition blocks the interrupt. Several conditions can block an interrupt, among them that an interrupt of equal or higher priority level is already in progress.

The hardware-generated LCALL causes the contents of the Program Counter to be pushed onto the stack, and reloads the PC with the beginning address of the service routine. As previously noted (Figure 3), the service routine for each interrupt begins at a fixed location.

Only the Program Counter is automatically pushed onto the stack, not the PSW or any other register. Having only the PC be automatically saved allows the programmer to decide how much time to spend saving which other registers. This enhances the interrupt response time, albeit at the expense of increasing the programmer's burden of responsibility. As a result, many interrupt functions that are typical in control applications—toggling a port pin, for example, or reloading a timer, or unloading a serial buffer—can often be completed in less time than it takes other architectures to commence them.

SIMULATING A THIRD PRIORITY LEVEL IN SOFTWARE

Some applications require more than the two priority levels that are provided by on-chip hardware in MCS-51 devices. In these cases, relatively simple software can be written to produce the same effect as a third priority level.

First, interrupts that are to have higher priority than 1 are assigned to priority 1 in the IP (Interrupt Priority) register. The service routines for priority 1 interrupts that are supposed to be interruptible by "priority 2" interrupts are written to include the following code:

```
        PUSH    IE
        MOV     IE,#MASK
        CALL    LABEL
        .......
(execute service routine)
        .......
        POP     IE
        RET
LABEL:  RETI
```

As soon as any priority 1 interrupt is acknowledged, the IE (Interrupt Enable) register is re-defined so as to disable all but "priority 2" interrupts. Then, a CALL to LABEL executes the RETI instruction, which clears the priority 1 interrupt-in-progress flip-flop. At this point any priority 1 interrupt that is enabled can be serviced, but only "priority 2" interrupts are enabled.

POPping IE restores the original enable byte. Then a normal RET (rather than another RETI) is used to terminate the service routine. The additional software adds 10 μs (at 12 MHz) to priority 1 interrupts.

ADDITIONAL REFERENCES

The following application notes are found in the *Embedded Control Applications* handbook. (Order Number: 270648)

1. AP-69 "An Introduction to the Intel MCS®-51 Single-Chip Microcomputer Family"
2. AP-70 "Using the Intel MCS®-51 Boolean Processing Capabilities"

8051, 8052 and 80C51 Hardware Description

CONTENTS

CONTENTS

intel.

8051, 8052 AND 80C51 HARDWARE DESCRIPTION

INTRODUCTION

This chapter presents a comprehensive description of the on-chip hardware features of the MCS®-51 microcontrollers. Included in this description are

- The port drivers and how they function both as ports and, for Ports 0 and 2, in bus operations
- The Timer/Counters
- The Serial Interface
- The Interrupt System
- Reset
- The Reduced Power Modes in the CHMOS devices
- The EPROM versions of the 8051AH, 8052AH and 80C51BH

The devices under consideration are listed in Table 1. As it becomes unwieldy to be constantly referring to each of these devices by their individual names, we will adopt a convention of referring to them generically as 8051s and 8052s, unless a specific member of the group is being referred to, in which case it will be specifically named. The "8051s" include the 8051AH, 80C51BH, and their ROMless and EPROM versions. The "8052s" are the 8052AH, 8032AH and 8752BH.

Figure 1 shows a functional block diagram of the 8051s and 8052s.

Table 1. The MCS-51 Family of Microcontrollers

Device Name	ROMless Version	EPROM Version	ROM Bytes	RAM Bytes	16-bit Timers	Ckt Type
8051AH	8031AH	8751H, 8751BH	4K	128	2	HMOS
8052AH	8032AH	8752BH	8K	256	3	HMOS
80C51BH	80C31BH	87C51	4K	128	2	CHMOS

Special Function Registers

A map of the on-chip memory area called SFR (Special Function Register) space is shown in Figure 2. SFRs marked by parentheses are resident in the 8052s but not in the 8051s.

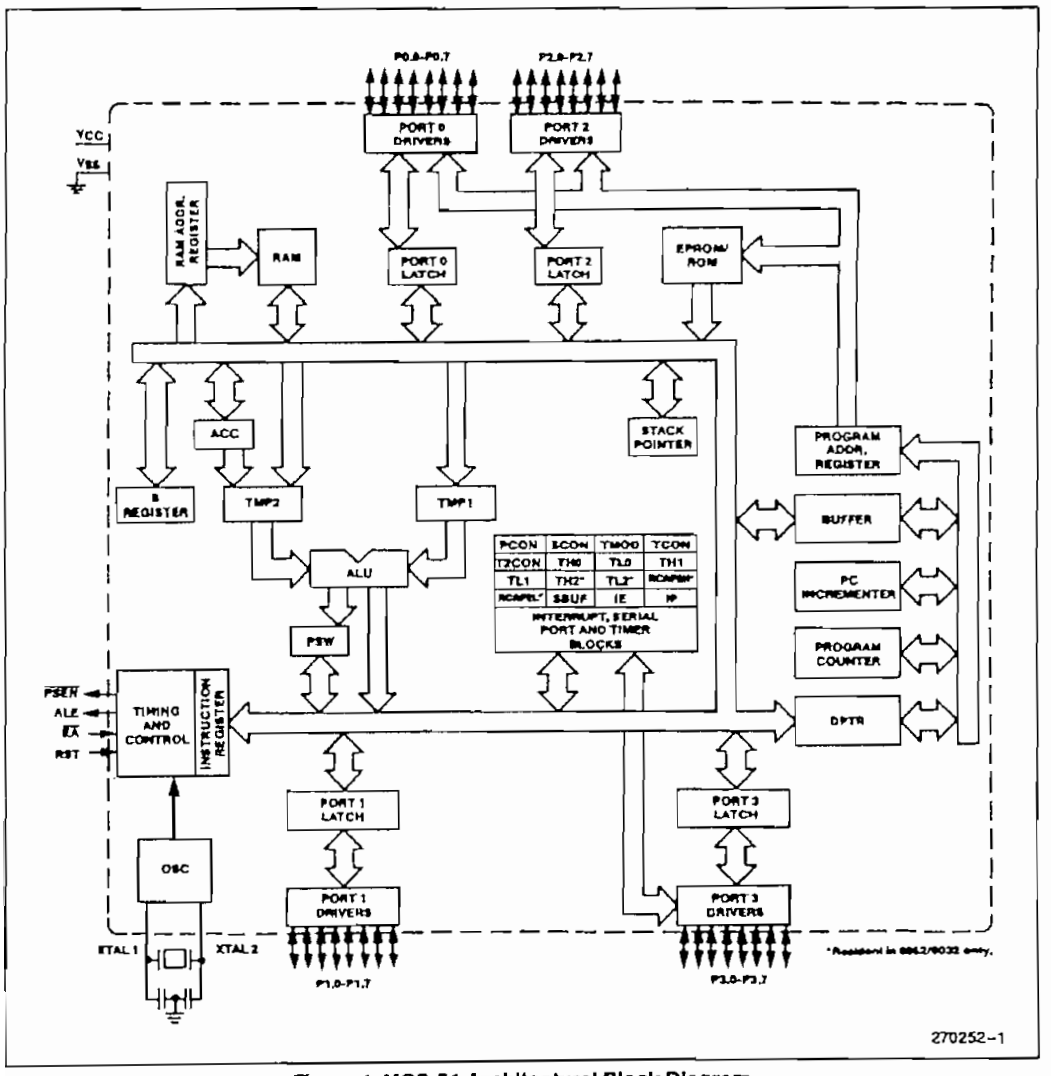


Figure 1. MCS-51 Architectural Block Diagram

	8 Bytes								
F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	(T2CON)		(RCAP2L)	(RCAP2H)	(TL2)	(TH2)			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

Figure 2. SFR Map. (. . .) Indicates Resident in 8052s, not in 8051s

Note that not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

User software should not write 1s to these unimplemented locations, since they may be used in future MCS-51 products to invoke new features. In that case the reset or inactive values of the new bits will always be 0, and their active values will be 1.

The functions of the SFRs are outlined below.

ACCUMULATOR

ACC is the Accumulator register. The mnemonics for Accumulator-Specific instructions, however, refer to the Accumulator simply as A.

B REGISTER

The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

PROGRAM STATUS WORD

The PSW register contains program status information as detailed in Figure 3.

STACK POINTER

The Stack Pointer Register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

DATA POINTER

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

PORTS 0 TO 3

P0, P1, P2 and P3 are the SFR latches of Ports 0, 1, 2 and 3, respectively.

SERIAL DATA BUFFER

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

TIMER REGISTERS

Register pairs (TH0, TL0), (TH1, TL1), and (TH2, TL2) are the 16-bit Counting registers for Timer/Counters 0, 1, and 2, respectively.

CAPTURE REGISTERS

The register pair (RCAP2H, RCAP2L) are the Capture registers for the Timer 2 "Capture Mode." In this mode, in response to a transition at the 8052's T2EX pin, TH2 and TL2 are copied into RCAP2H and RCAP2L. Timer 2 also has a 16-bit auto-reload mode, and RCAP2H and RCAP2L hold the reload value for this mode. More about Timer 2's features in a later section.

CONTROL REGISTERS

Special Function Registers IP, IE, TMOD, TCON, T2CON, SCON, and PCON contain control and status bits for the interrupt system, the Timer/Counters, and the serial port. They are described in later sections.

(MSB)							(LSB)
CY	AC	F0	RS1	RS0	OV	—	P

Symbol	Position	Name and Significance
CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary Carry flag. (For BCD operations.)
F0	PSW.5	Flag 0 (Available to the user for general purposes.)
RS1	PSW.4	Register bank select control bits 1 &
RS0	PSW.3	0. Set/cleared by software to determine working register bank (see Note).
OV	PSW.2	Overflow flag.
—	PSW.1	User definable flag.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the Accumulator, i.e., even parity.

NOTE:
The contents of (RS1, RS0) enable the working register banks as follows:

(0,0)—Bank 0 (00H–07H)
(0,1)—Bank 1 (08H–0FH)
(1,0)—Bank 2 (10H–17H)
(1,1)—Bank 3 (18H–1FH)

Figure 3. PSW: Program Status Word Register

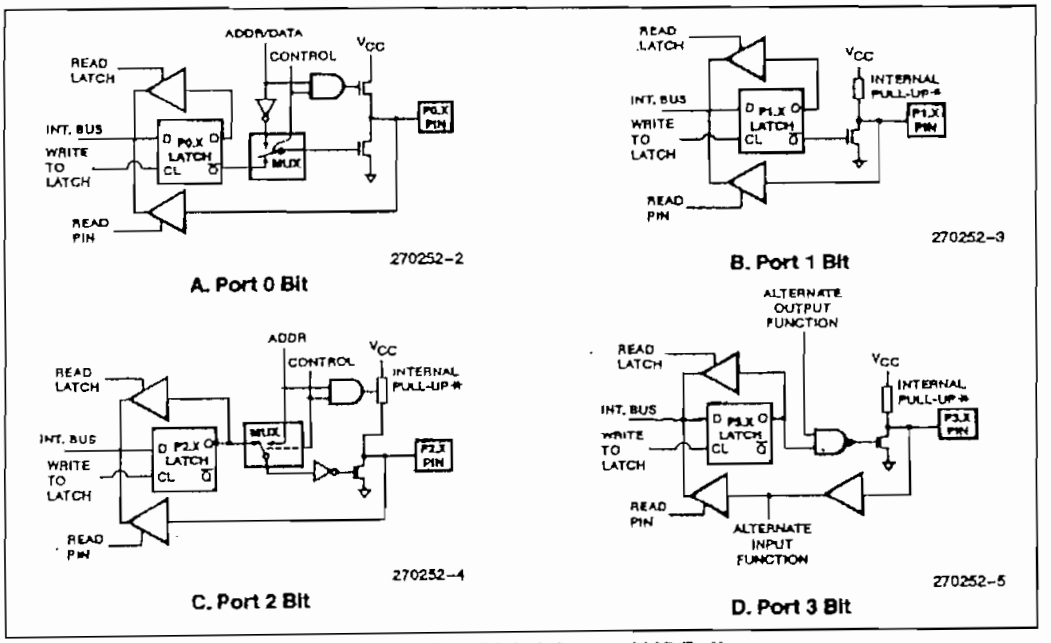


Figure 4. 8051 Port Bit Latches and I/O Buffers
*See Figure 5 for details of the internal pullup.

PORT STRUCTURES AND OPERATION

All four ports in the 8051 are bidirectional. Each consists of a latch (Special Function Registers P0 through P3), an output driver, and an input buffer.

The output drivers of Ports 0 and 2, and the input buffers of Port 0, are used in accesses to external memory. In this application, Port 0 outputs the low byte of the external memory address, time-multiplexed with the byte being written or read. Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. Otherwise the Port 2 pins continue to emit the P2 SFR content.

All the Port 3 pins, and (in the 8052) two Port 1 pins are multifunctional. They are not only port pins, but also serve the functions of various special features as listed on the following page.

Port Pin	Alternate Function
*P1.0	T2 (Timer/Counter 2 external input)
*P1.1	T2EX (Timer/Counter 2 Capture/Reload trigger)
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt)
P3.3	$\overline{INT1}$ (external interrupt)
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input)
P3.6	$\overline{WR}$ (external Data Memory write strobe)
P3.7	$\overline{RD}$ (external Data Memory read strobe)

*P1.0 and P1.1 serve these alternate functions only on the 8052.

The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise the port pin is stuck at 0.

I/O Configurations

Figure 4 shows a functional diagram of a typical bit latch and I/O buffer in each of the four ports. The bit latch (one bit in the port's SFR) is represented as a Type D flip-flop, which will clock in a value from the internal bus in response to a "write to latch" signal from the CPU. The Q output of the flip-flop is placed on the internal bus in response to a "read latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read pin" signal from the CPU. Some instructions that read a port activate the "read latch" signal, and others activate the "read pin" signal. More about that later.

As shown in Figure 4, the output drivers of Ports 0 and 2 are switchable to an internal ADDR and ADDR/DATA bus by an internal CONTROL signal for use in external memory accesses. During external memory accesses, the P2 SFR remains unchanged, but the P0 SFR gets 1s written to it.

Also shown in Figure 4, is that if a P3 bit latch contains a 1, then the output level is controlled by the signal labeled "alternate output function." The actual P3.X pin level is always available to the pin's alternate input function, if any.

Ports 1, 2, and 3 have internal pullups. Port 0 has open drain outputs. Each I/O line can be independently used as an input or an output. (Ports 0 and 2 may not be used as general purpose I/O when being used as the ADDR/DATA BUS). To be used as an input, the port bit latch must contain a 1, which turns off the output driver FET. Then, for Ports 1, 2, and 3, the pin is pulled high by the internal pullup, but can be pulled low by an external source.

Port 0 differs in not having internal pullups. The pullup FET in the P0 output driver (see Figure 4) is used only when the Port is emitting 1s during external memory accesses. Otherwise the pullup FET is off. Consequently P0 lines that are being used as output port lines are open drain. Writing a 1 to the bit latch leaves both output FETs off, so the pin floats. In that condition it can be used a high-impedance input.

Because Ports 1, 2, and 3 have fixed internal pullups they are sometimes called "quasi-bidirectional" ports. When configured as inputs they pull high and will source current (I_{IL}, in the data sheets) when externally pulled low. Port 0, on the other hand, is considered "true" bidirectional, because when configured as an input it floats.

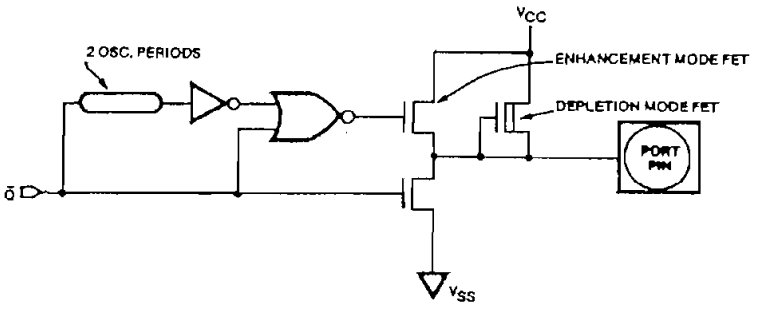
All the port latches in the 8051 have 1s written to them by the reset function. If a 0 is subsequently written to a port latch, it can be reconfigured as an input by writing a 1 to it.

Writing to a Port

In the execution of an instruction that changes the value in a port latch, the new value arrives at the latch during S6P2 of the final cycle of the instruction. However, port latches are in fact sampled by their output buffers only during Phase 1 of any clock period. (During Phase 2 the output buffer holds the value it saw during the previous Phase 1). Consequently, the new value in the port latch won't actually appear at the output pin until the next Phase 1, which will be at S1P1 of the next machine cycle. See Figure 39 in the Internal Timing section.

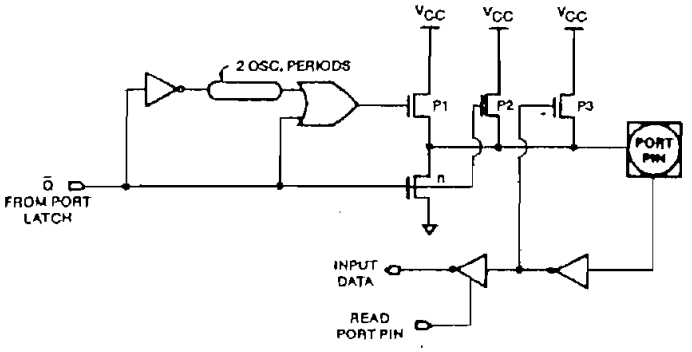
If the change requires a 0-to-1 transition in Port 1, 2, or 3, an additional pullup is turned on during S1P1 and S1P2 of the cycle in which the transition occurs. This is done to increase the transition speed. The extra pullup can source about 100 times the current that the normal pullup can. It should be noted that the internal pullups are field-effect transistors, not linear resistors. The pullup arrangements are shown in Figure 5.

In HMOS versions of the 8051, the fixed part of the pullup is a depletion-mode transistor with the gate wired to the source. This transistor will allow the pin to source about 0.25 mA when shorted to ground. In parallel with the fixed pullup is an enhancement-mode transistor, which is activated during S1 whenever the port bit does a 0-to-1 transition. During this interval, if the port pin is shorted to ground, this extra transistor will allow the pin to source an additional 30 mA.



270252-8

A. HMOS Configuration. The enhancement mode transistor is turned on for 2 osc. periods after Q makes a 0-to-1 transition.



270252-7

B. CHMOS Configuration. pFET 1 is turned on for 2 osc. periods after Q makes a 0-to-1 transition. During this time, pFET 1 also turns on pFET 3 through the inverter to form a latch which holds the 1. pFET 2 is also on.

Figure 5. Ports 1 And 3 HMOS And CHMOS Internal Pullup Configurations. Port 2 Is Similar Except That It Holds The Strong Pullup On While Emitting 1s That Are Address Bits. (See Text, "Accessing External Memory".)

In the CHMOS versions, the pullup consists of three pFETs. It should be noted that an n-channel FET (nFET) is turned on when a logical 1 is applied to its gate, and is turned off when a logical 0 is applied to its gate. A p-channel FET (pFET) is the opposite: it is on when its gate sees a 0, and off when its gate sees a 1.

pFET1 in Figure 5 is the transistor that is turned on for 2 oscillator periods after a 0-to-1 transition in the port latch. While it's on, it turns on pFET3 (a weak pullup), through the inverter. This inverter and pFET form a latch which hold the 1.

Note that if the pin is emitting a 1, a negative glitch on the pin from some external source can turn off pFET3, causing the pin to go into a float state. pFET2 is a very weak pullup which is on whenever the nFET is off, in traditional CMOS style. It's only about $^1/_{10}$ the strength of pFET3. Its function is to restore a 1 to the pin in the event the pin *had* a 1 and lost it to a glitch.

Port Loading and Interfacing

The output buffers of Ports 1, 2, and 3 can each drive 4 LS TTL inputs. These ports on HMOS versions can be driven in a normal manner by any TTL or NMOS circuit. Both HMOS and CHMOS pins can be driven by open-collector and open-drain outputs, but note that 0-to-1 transitions will not be fast. In the HMOS device, if the pin is driven by an open-collector output, a 0-to-1 transition will have to be driven by the relatively weak depletion mode FET in Figure 5(A). In the CHMOS device, an input 0 turns off pullup pFET3, leaving only the very weak pullup pFET2 to drive the transition.

In external bus mode, Port 0 output buffers can each drive 8 LS TTL inputs. As port pins, they require external pullups to drive any inputs.

Read-Modify-Write Feature

Some instructions that read a port read the latch and others read the pin. Which ones do which? The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the latch. These are called "read-modify-write" instructions. The instructions listed below are read-modify-write instructions. When the destination operand is a port, or a port bit, these instructions read the latch rather than the pin:

ANL	(logical AND, e.g., ANL P1, A)
ORL	(logical OR, e.g., ORL P2, A)
XRL	(logical EX-OR, e.g., XRL P3, A)
JBC	(jump if bit = 1 and clear bit, e.g., JBC P1.1, LABEL)
CPL	(complement bit, e.g., CPL P3.0)
INC	(increment, e.g., INC P2)
DEC	(decrement, e.g., DEC P2)
DJNZ	(decrement and jump if not zero, e.g., DJNZ P3, LABEL)
MOV, PX.Y, C	(move carry bit to bit Y of Port X)
CLR PX.Y	(clear bit Y of Port X)
SETB PX.Y	(set bit Y of Port X)

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch.

The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of a transistor. When a 1 is written to the bit, the transistor is turned on. If the CPU then reads the same port bit at the pin rather than the latch, it will read the base voltage of the transistor and interpret it as a 0. Reading the latch rather than the pin will return the correct value of 1.

ACCESSING EXTERNAL MEMORY

Accesses to external memory are of two types: accesses to external Program Memory and accesses to external Data Memory. Accesses to external Program Memory use signal $\overline{PSEN}$ (program store enable) as the read strobe. Accesses to external Data Memory use $\overline{RD}$ or $\overline{WR}$ (alternate functions of P3.7 and P3.6) to strobe the memory. Refer to Figures 36 through 38 in the Internal Timing section.

Fetches from external Program Memory always use a 16-bit address. Accesses to external Data Memory can use either a 16-bit address (MOVX @DPTR) or an 8-bit address (MOVX @Ri).

Whenever a 16-bit address is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or write cycle. Note that the Port 2 drivers use the strong pullups during the entire time that they are emitting address bits that are 1s. This is during the execution of a MOVX @DPTR instruction. During this time the Port 2 latch (the Special Function Register) does not have to contain 1s, and the contents of the Port 2 SFR are not modified. If the external memory cycle is not immediately followed by another external memory cycle, the undisturbed contents of the Port 2 SFR will reappear in the next cycle.

If an 8-bit address is being used (MOVX @Ri), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. This will facilitate paging.

In any case, the low byte of the address is time-multiplexed with the data byte on Port 0. The ADDR/DATA signal drives both FETs in the Port 0 output buffers. Thus, in this application the Port 0 pins are not open-drain outputs, and do not require external pullups. Signal ALE (Address Latch Enable) should be used to capture the address byte into an external latch. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before $\overline{WR}$ is activated, and remains there until after $\overline{WR}$ is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe is deactivated.

During any access to external memory, the CPU writes 0FFH to the Port 0 latch (the Special Function Register), thus obliterating whatever information the Port 0 SFR may have been holding. If the user writes to Port 0 during an external memory fetch, the incoming code byte is corrupted. Therefore, do not write to Port 0 if external program memory is used.

External Program Memory is accessed under two conditions:

1) Whenever signal $\overline{EA}$ is active; or
2) Whenever the program counter (PC) contains a number that is larger than 0FFFH (1FFFH for the 8052).

This requires that the ROMless versions have $\overline{EA}$ wired low to enable the lower 4K (8K for the 8032) program bytes to be fetched from external memory.

When the CPU is executing out of external Program Memory, all 8 bits of Port 2 are dedicated to an output function and may not be used for general purpose I/O. During external program fetches they output the high byte of the PC. During this time the Port 2 drivers use the strong pullups to emit PC bits that are 1s.

TIMER/COUNTERS

The 8051 has two 16-bit Timer/Counter registers: Timer 0 and Timer 1. The 8052 has these two plus one

more: Timer 2. All three can be configured to operate either as timers or event counters.

In the "Timer" function, the register is incremented every machine cycle. Thus, one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is $1/12$ of the oscillator frequency.

In the "Counter" function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0, T1 or (in the 8052) T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is $1/24$ of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full machine cycle.

In addition to the "Timer" or "Counter" selection, Timer 0 and Timer 1 have four operating modes from which to select. Timer 2, in the 8052, has three modes of operation: "Capture," "Auto-Reload" and "baud rate generator."

Timer 0 and Timer 1

These Timer/Counters are present in both the 8051 and the 8052. The "Timer" or "Counter" function is selected by control bits C/$\overline{T}$ in the Special Function Register TMOD (Figure 6). These two Timer/Counters have four operating modes, which are selected by bit-pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for both Timer/Counters. Mode 3 is different. The four operating modes are described in the following text.

MODE 0

Either Timer in Mode 0 is an 8-bit Counter with a divide-by-32 prescaler. This 13-bit timer is MCS-48 compatible. Figure 7 shows the Mode 0 operation as it applies to Timer 1.

In this mode, the Timer register is configured as a 13-Bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TF1. The counted input is enabled to the Timer when TR1 = 1 and either GATE = 0 or $\overline{INT1}$ = 1. (Setting GATE = 1 allows the Timer to be controlled by external input $\overline{INT1}$, to facilitate pulse width measurements.) TR1 is a control bit in the Special Function Register TCON (Figure 8). GATE is in TMOD.

The 13-Bit register consists of all 8 bits of TH1 and the lower 5 bits of TL1. The upper 3 bits of TL1 are indeterminate and should be ignored. Setting the run flag (TR1) does not clear the registers.

Mode 0 operation is the same for Timer 0 as for Timer 1. Substitute TR0, TF0 and $\overline{INT0}$ for the corresponding Timer 1 signals in Figure 7. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

MODE 1

Mode 1 is the same as Mode 0, except that the Timer register is being run with all 16 bits.

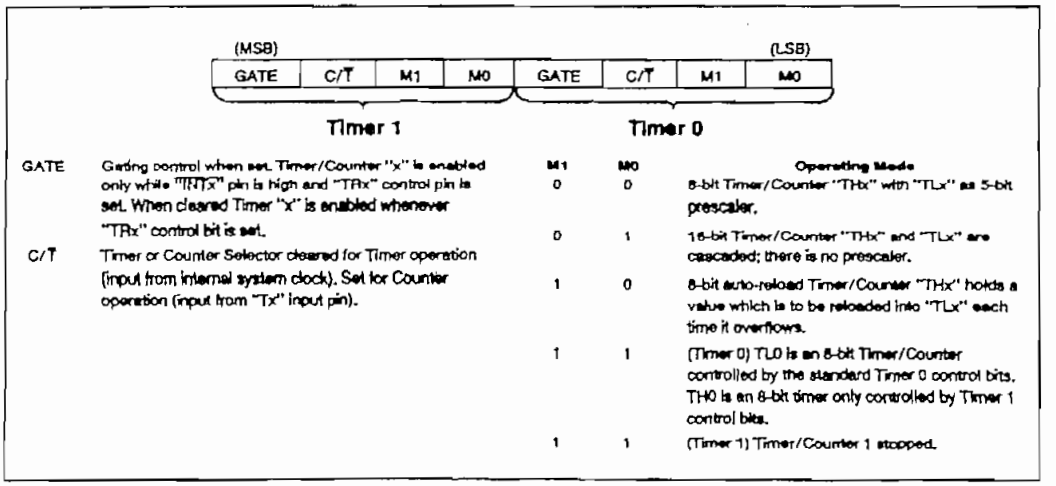


Figure 6. TMOD: Timer/Counter Mode Control Register

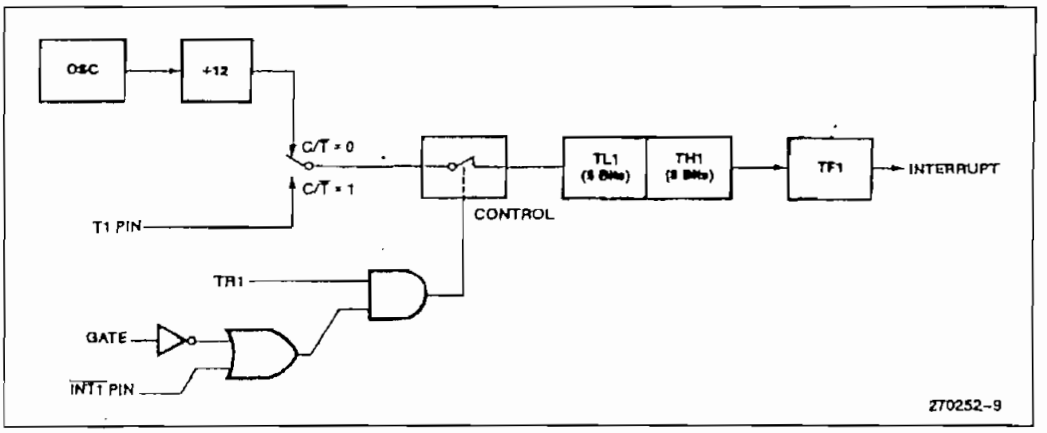


Figure 7. Timer/Counter 1 Mode 0: 13-Bit Counter

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Symbol	Position	Name and Significance
TF1	TCON.7	Timer 1 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
TF0	TCON.5	Timer 0 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

Figure 8.TCON: Timer/Counter Control Register

MODE 2

Mode 2 configures the Timer register as an 8-bit Counter (TL1) with automatic reload, as shown in Figure 9. Overflow from TL1 not only sets TF1, but also reloads TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged.

Mode 2 operation is the same for Timer/Counter 0.

MODE 3

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1 = 0.

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in Figure 10. TL0 uses the Timer 0 control bits: C/$\overline{T}$, GATE, TR0, $\overline{INT0}$, and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus, TH0 now controls the "Timer 1" interrupt.

Mode 3 is provided for applications requiring an extra 8-bit timer or counter. With Timer 0 in Mode 3, an 8051 can look like it has three Timer/Counters, and an 8052, like it has four. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3, or can still be used by the serial port as a baud rate generator, or in fact, in any application not requiring an interrupt.

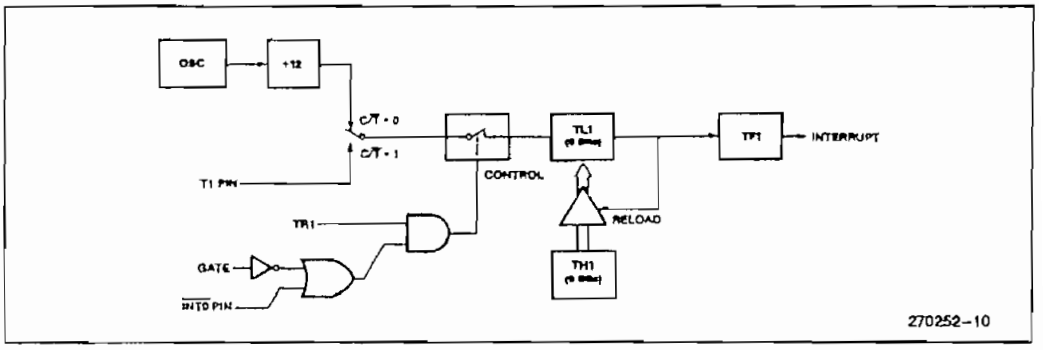


Figure 9. Timer/Counter 1 Mode 2: 8-Bit Auto-Reload

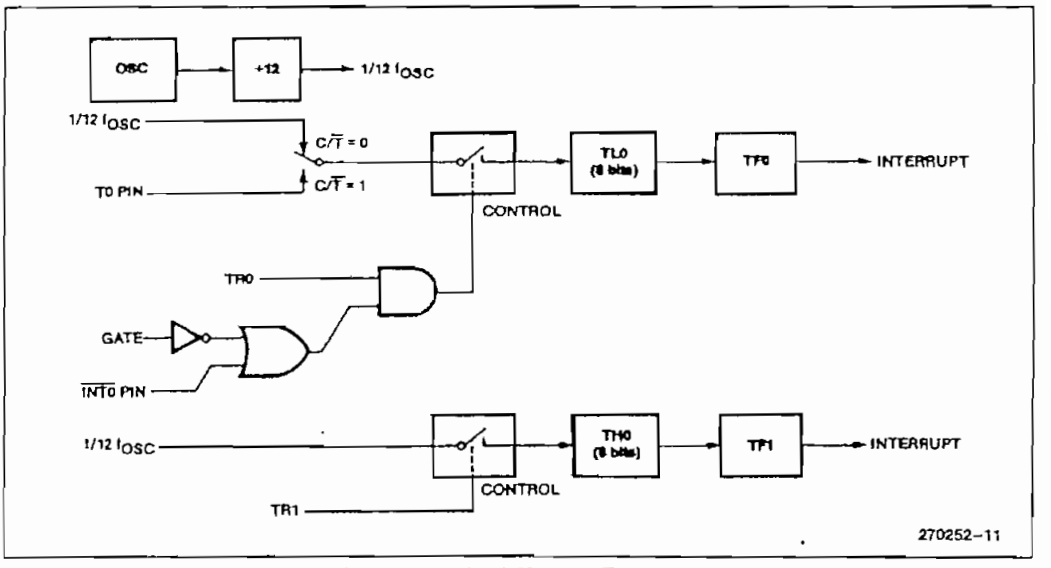


Figure 10. Timer/Counter 0 Mode 3: Two 8-Bit Counters

Timer 2

Timer 2 is a 16-bit Timer/Counter which is present only in the 8052. Like Timers 0 and 1, it can operate either as a timer or as an event counter. This is selected by bit C/$\overline{T2}$ in the Special Function Register T2CON (Figure 11). It has three operating modes: "capture," "auto-load" and "baud rate generator," which are selected by bits in T2CON as shown in Table 2.

Table 2. Timer 2 Operating Modes

RCLK + TCLK	CP/$\overline{RL2}$	TR2	Mode
0	0	1	16-bit Auto-Reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(off)

(MSB) (LSB)

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

Symbol	Position	Name and Significance
TF2	T2CON.7	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	T2CON.6	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.
RCLK	T2CON.5	Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	T2CON.4	Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	T2CON.3	Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	T2CON.2	Start/stop control for Timer 2. A logic 1 starts the timer.
C/$\overline{T2}$	T2CON.1	Timer or counter select. (Timer 2) 0 = Internal timer (OSC/12) 1 = External event counter (falling edge triggered).
CP/$\overline{RL2}$	T2CON.0	Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Figure 11. T2CON: Timer/Counter 2 Control Register

In the Capture Mode there are two options which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then Timer 2 is a 16-bit timer or counter which upon overflowing sets bit TF2, the Timer 2 overflow bit, which can be used to generate an interrupt. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX causes the current value in the Timer 2 registers, TL2 and TH2, to be captured into registers RCAP2L and RCAP2H, respectively. (RCAP2L and RCAP2H are new Special Function Registers in the 8052.) In addition, the transition at T2EX causes bit EXF2 in T2CON to be set, and EXF2, like TF2, can generate an interrupt.

The Capture Mode is illustrated in Figure 12.

In the auto-reload mode there are again two options, which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then when Timer 2 rolls over it not only sets TF2 but also causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2L and RCAP2H, which are preset by software. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX will also trigger the 16-bit reload and set EXF2.

The auto-reload mode is illustrated in Figure 13.

The baud rate generator mode is selected by RCLK = 1 and/or TCLK = 1. It will be described in conjunction with the serial port.

SERIAL INTERFACE

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register. (However, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost). The serial port receive and transmit registers are both accessed at Special Function Register SBUF. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

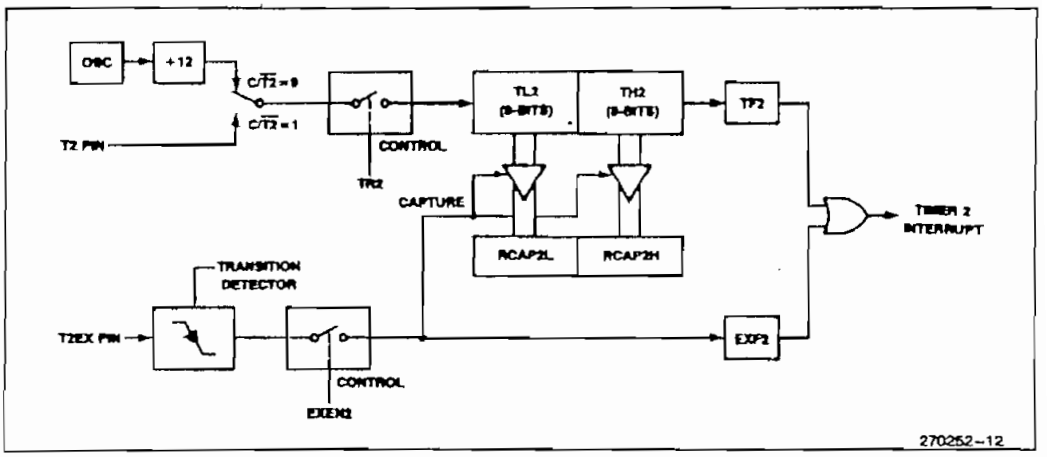


Figure 12. Timer 2 in Capture Mode

The serial port can operate in 4 modes:

Mode 0: Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at 1/12 the oscillator frequency.

Mode 1: 10 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in Special Function Register SCON. The baud rate is variable.

Mode 2: 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On Transmit, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8. On receive, the 9th data bit goes into RB8 in Special Functon Register SCON, while the stop bit is ignored. The baud rate is programmable to either $1/_{32}$ or $1/_{64}$ the oscillator frequency.

Mode 3: 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit and a stop bit (1). In fact, Mode 3 is the same as Mode 2 in all respects except the baud rate. The baud rate in Mode 3 is variable.

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit if REN = 1.

Multiprocessor Communications

Modes 2 and 3 have a special provision for multiprocessor communications. In these modes, 9 data bits are received. The 9th one goes into RB8. Then comes a stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8 = 1. This feature is enabled by setting bit SM2 in SCON. A way to use this feature in multiprocessor systems is as follows.

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2 = 1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and see if it is being addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that weren't being addressed leave their SM2s set and go on about their business, ignoring the coming data bytes.

SM2 has no effect in Mode 0, and in Mode 1 can be used to check the validity of the stop bit. In a Mode 1 reception, if SM2 = 1, the receive interrupt will not be activated unless a valid stop bit is received.

Serial Port Control Register

The serial port control and status register is the Special Function Register SCON, shown in Figure 14. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8), and the serial port interrupt bits (TI and RI).

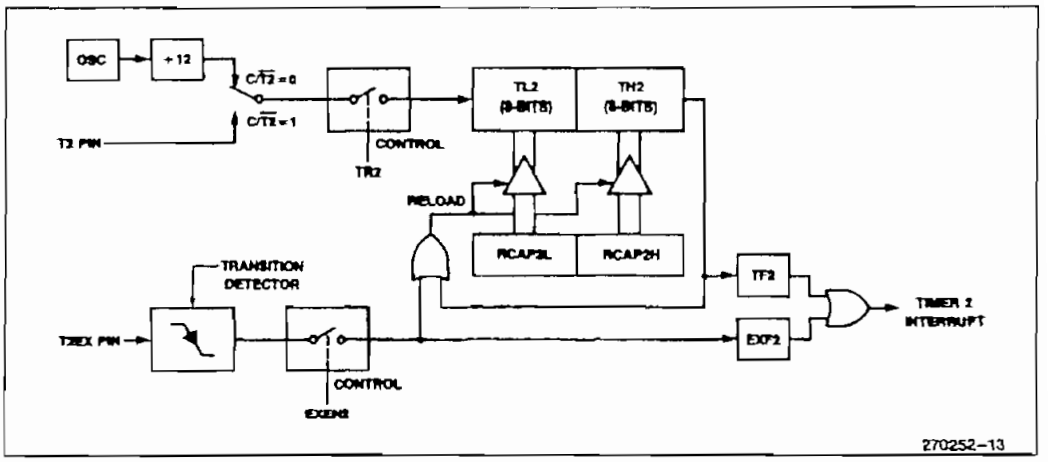


Figure 13. Timer 2 in Auto-Reload Mode

(MSB)							(LSB)
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Where SM0, SM1 specify the serial port mode, as follows:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	shift register	$f_{osc.}/12$
0	1	1	8-bit UART	variable
1	0	2	9-bit UART	$f_{osc.}/64$ or $f_{osc.}/32$
1	1	3	9-bit UART	variable

- SM2 enables the multiprocessor communication feature in Modes 2 and 3. In Mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In Mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In Mode 0, SM2 should be 0.
- REN enables serial reception. Set by software to enable reception. Clear by software to disable reception.
- TB8 is the 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.
- RB8 in Modes 2 and 3, is the 9th data bit that was received. In Mode 1, if SM2 = 0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.
- TI is transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.
- RI is receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.

Figure 14. SCON: Serial Port Control Register

Baud Rates

The baud rate in Mode 0 is fixed:

$$\text{Mode 0 Baud Rate} = \frac{\text{Oscillator Frequency}}{12}$$

The baud rate in Mode 2 depends on the value of bit SMOD in Special Function Register PCON. If SMOD = 0 (which is the value on reset), the baud rate is $1/64$ the oscillator frequency. If SMOD = 1, the baud rate is $1/32$ the oscillator frequency.

$$\text{Mode 2 Baud Rate} = \frac{2^{\text{SMOD}}}{64} \times (\text{Oscillator Frequency})$$

In the 8051, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate. In the 8052, these baud rates can be determined by Timer 1, or by Timer 2, or by both (one for transmit and the other for receive).

Using Timer 1 to Generate Baud Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows:

$$\text{Modes 1, 3 Baud Rate} = \frac{2^{SMOD}}{32} \times (\text{Timer 1 Overflow Rate})$$

The Timer 1 interrupt should be disabled in this application. The Timer itself can be configured for either "timer" or "counter" operation, and in any of its 3 running modes. In the most typical applications, it is configured for "timer" operation, in the auto-reload mode (high nibble of TMOD = 0010B). In that case, the baud rate is given by the formula

$$\text{Modes 1, 3 Baud Rate} = \frac{2^{SMOD}}{32} \times \frac{\text{Oscillator Frequency}}{12x\,[256 - (TH1)]}$$

One can achieve very low baud rates with Timer 1 by leaving the Timer 1 interrupt enabled, and configuring the Timer to run as a 16-bit timer (high nibble of TMOD = 0001B), and using the Timer 1 interrupt to do a 16-bit software reload.

Figure 15 lists various commonly used baud rates and how they can be obtained from Timer 1.

Baud Rate	f_{OSC}	SMOD	Timer 1		
			C/T	Mode	Reload Value
Mode 0 Max: 1 MHZ	12 MHZ	X	X	X	X
Mode 2 Max: 375K	12 MHZ	1	X	X	X
Modes 1, 3: 62.5K	12 MHZ	1	0	2	FFH
19.2K	11.059 MHZ	1	0	2	FDH
9.6K	11.059 MHZ	0	0	2	FDH
4.8K	11.059 MHZ	0	0	2	FAH
2.4K	11.059 MHZ	0	0	2	F4H
1.2K	11.059 MHZ	0	0	2	E8H
137.5	11.986 MHZ	0	0	2	1DH
110	6 MHZ	0	0	2	72H
110	12 MHZ	0	0	1	FEEBH

Figure 15. Timer 1 Generated Commonly Used Baud Rates

Using Timer 2 to Generate Baud Rates

In the 8052, Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Figure 11). Note then the baud rates for transmit and receive can be simultaneously different. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 16.

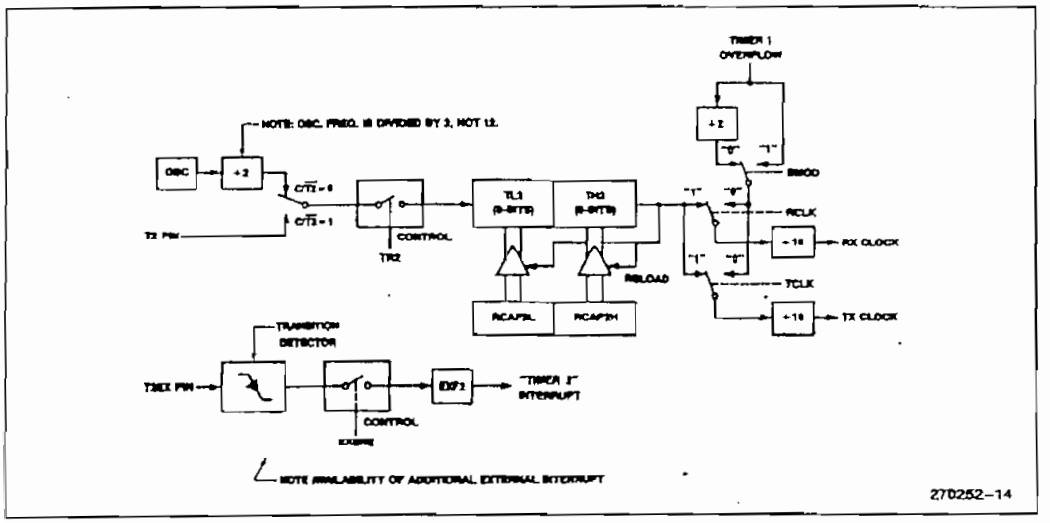


Figure 16. Timer 2 in Baud Rate Generator Mode

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

Now, the baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate as follows:

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either "timer" or "counter" operation. In the most typical applications, it is configured for "timer" operation (C/T2 = 0). "Timer" operation is a little different for Timer 2 when it's being used as a baud rate generator. Normally, as a timer it would increment every machine cycle (thus at $1/12$ the oscillator frequency). As a baud rate generator, however, it increments every state time (thus at $1/2$ the oscillator frequency). In that case the baud rate is given by the formula

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - (\text{RCAP2H, RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 16. This Figure is valid only if RCLK + TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Therefore, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt, if desired.

It should be noted that when Timer 2 is running (TR2 = 1) in "timer" function in the baud rate generator mode, one should not try to read or write TH2 or TL2. Under these conditions the Timer is being incremented every state time, and the results of a read or write may not be accurate. The RCAP registers may be read, but shouldn't be written to, because a write might overlap a reload and cause write and/or reload errors. Turn the Timer off (clear TR2) before accessing the Timer 2 or RCAP registers, in this case.

More About Mode 0

Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at $1/12$ the oscillator frequency.

Figure 17 shows a simplified functional diagram of the serial port in Mode 0, and associated timing.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal at S6P2 also loads a 1 into the 9th position of the transmit shift register and tells the TX Control block to commence a transmission. The internal timing is such that one full machine cycle will elapse between "write to SBUF," and activation of SEND.

SEND enables the output of the shift register to the alternate output function line of P3.0, and also enables SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK is low during S3, S4, and S5 of every machine cycle, and high during S6, S1 and S2. At S6P2 of every machine cycle in which SEND is active, the contents of the transmit shift register are shifted to the right one position.

As data bits shift out to the right, zeroes come in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position, is just to the left of the MSB, and all positions to the left of that contain zeroes. This condition flags the TX Control block to do one last shift and then deactivate SEND and set TI. Both of these actions occur at S1P1 of the 10th machine cycle after "write to SBUF."

Reception is initiated by the condition REN = 1 and RI = 0. At S6P2 of the next machine cycle, the RX Control unit writes the bits 11111110 to the receive shift register, and in the next clock phase activates RECEIVE.

RECEIVE enables SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK makes transitions at S3P1 and S6P1 of every machine cycle. At S6P2 of every machine cycle in which RECEIVE is active, the contents of the receive shift register are shifted to the left one position. The value that comes in from the right is the value that was sampled at the P3.0 pin at S5P2 of the same machine cycle.

As data bits come in from the right, 1s shift out to the left. When the 0 that was initially loaded into the rightmost position arrives at the leftmost position in the shift register, it flags the RX Control block to do one last shift and load SBUF. At S1P1 of the 10th machine cycle after the write to SCON that cleared RI, RECEIVE is cleared and RI is set.

More About Mode 1

Ten bits are transmitted (through TXD), or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in SCON. In the 8051 the baud rate is determined by the Timer 1 overflow rate. In the 8052 it is determined either by the Timer 1 overflow rate, or the Timer 2 overflow rate, or both (one for transmit and the other for receive).

Figure 18 shows a simplified functional diagram of the serial port in Mode 1, and associated timings for transmit receive.

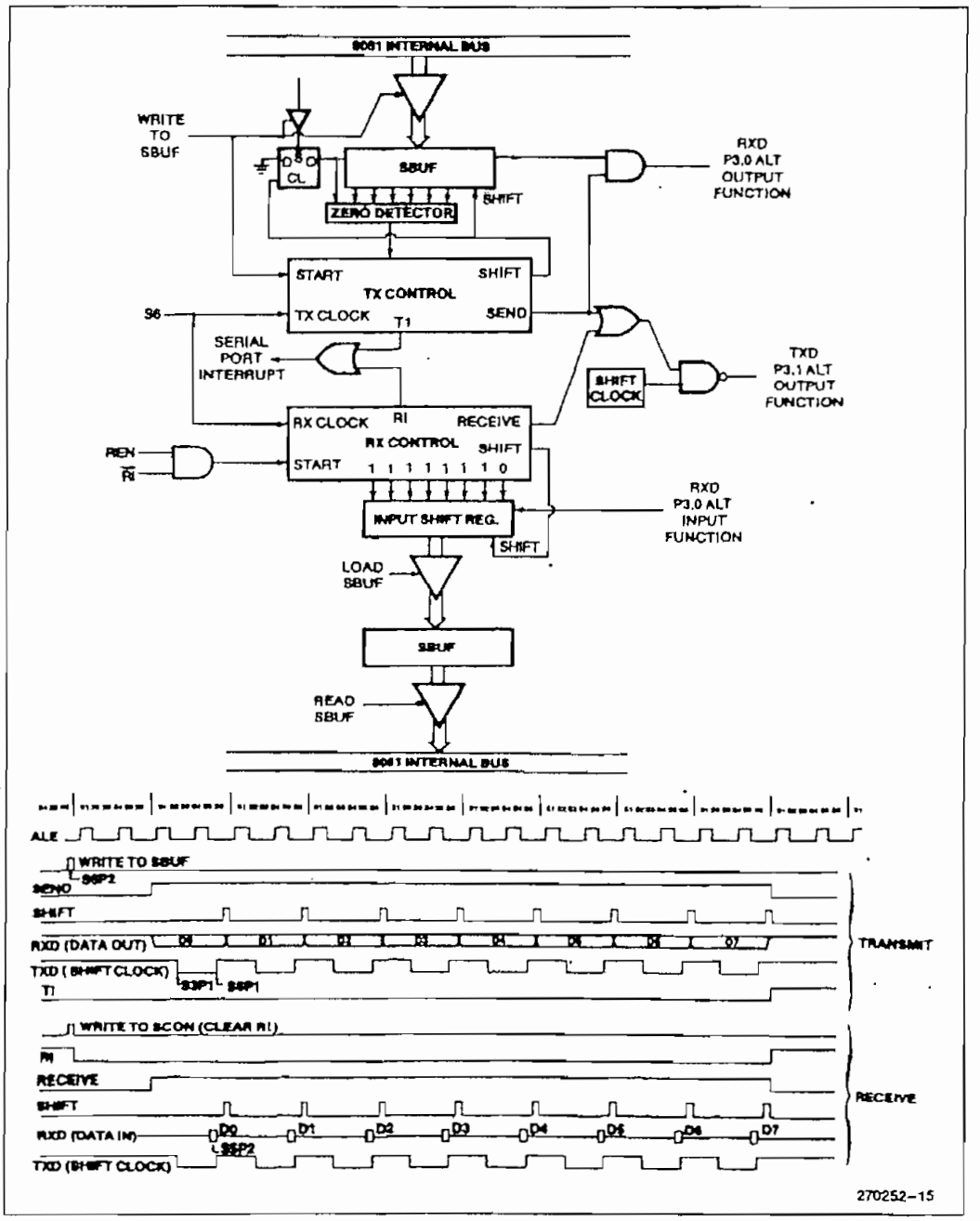


Figure 17. Serial Port Mode 0

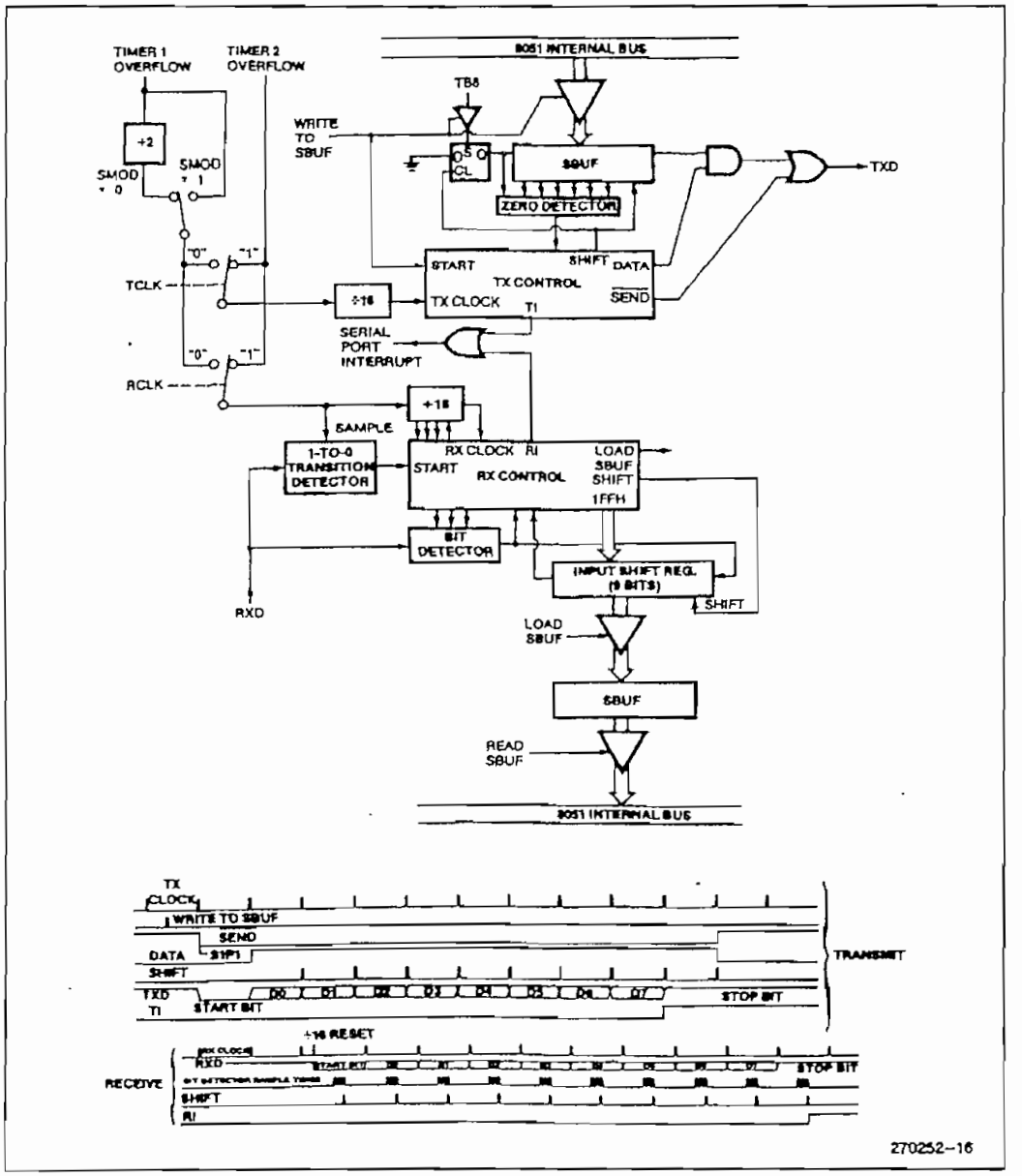


Figure 18. Serial Port Mode 1. TCLK, RCLK and Timer 2 are Present in the 8052/8032 Only.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads a 1 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission actually commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal).

The transmission begins with activation of $\overline{\text{SEND}}$, which puts the start bit at TXD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TXD. The first shift pulse occurs one bit time after that.

As data bits shift out to the right, zeroes are clocked in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position is just to the left of the MSB, and all positions to the left of that contain zeroes. This condition flags the TX Control unit to do one last shift and then deactivate $\overline{\text{SEND}}$ and set TI. This occurs at the 10th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written into the input shift register. Resetting the divide-by-16 counter aligns its rollovers with the boundaries of the incoming bit times.

The 16 states of the counter divide each bit time into 16ths. At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register, (which in mode 1 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated.

1) RI = 0, and
2) Either SM2 = 0, or the received stop bit = 1

If either of these two conditions is not met, the received frame is irretrievably lost. If both conditions are met, the stop bit goes into RB8, the 8 data bits go into SBUF, and RI is activated. At this time, whether the above conditions are met or not, the unit goes back to looking for a 1-to-0 transition in RXD.

More About Modes 2 and 3

Eleven bits are transmitted (through TXD), or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmit, the 9th data bit (TB8) can be assigned the value of 0 or 1. On receive, the 9th data bit goes into RB8 in SCON. The baud rate is programmable to either $1/_{32}$ or $1/_{64}$ the oscillator frequency in Mode 2. Mode 3 may have a variable baud rate generated from either Timer 1 or 2 depending on the state of TCLK and RCLK.

Figures 19 and 20 show a functional diagram of the serial port in Modes 2 and 3. The receive portion is exactly the same as in Mode 1. The transmit portion differs from Mode 1 only in the 9th bit of the transmit shift register.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads TB8 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal.)

The transmission begins with activation of SEND, which puts the start bit at TXD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TXD. The first shift pulse occurs one bit time after that. The first shift clocks a 1 (the stop bit) into the 9th bit position of the shift register. Thereafter, only zeroes are clocked in. Thus, as data bits shift out to the right, zeroes are clocked in from the left. When TB8 is at the output position of the shift register, then the stop bit is just to the left of TB8, and all positions to the left of that contain zeroes. This condition flags the TX Control unit to do one last shift and then deactivate SEND and set TI. This occurs at the 11th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written to the input shift register.

At the 7th, 8th and 9th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

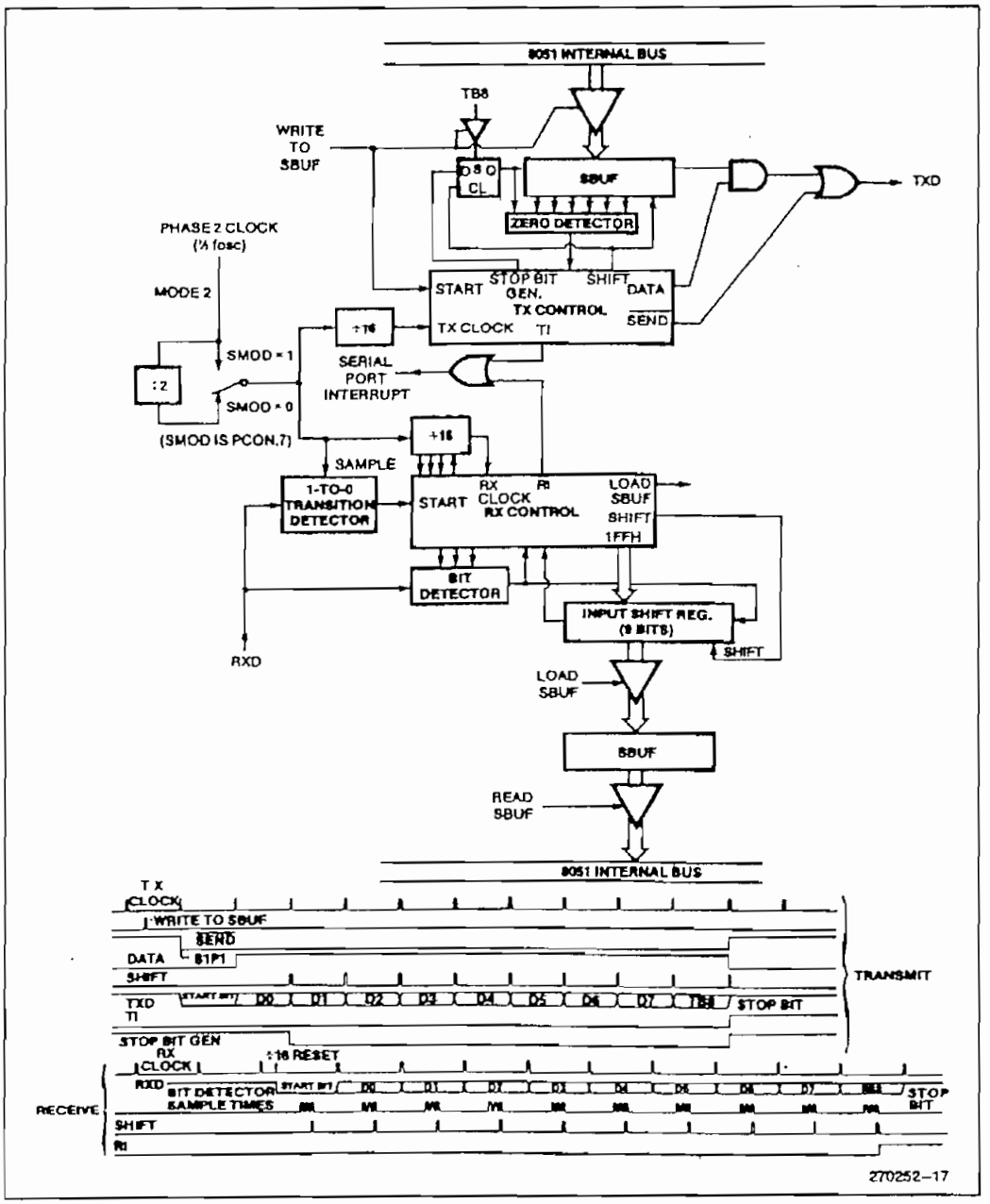


Figure 19. Serial Port Mode 2

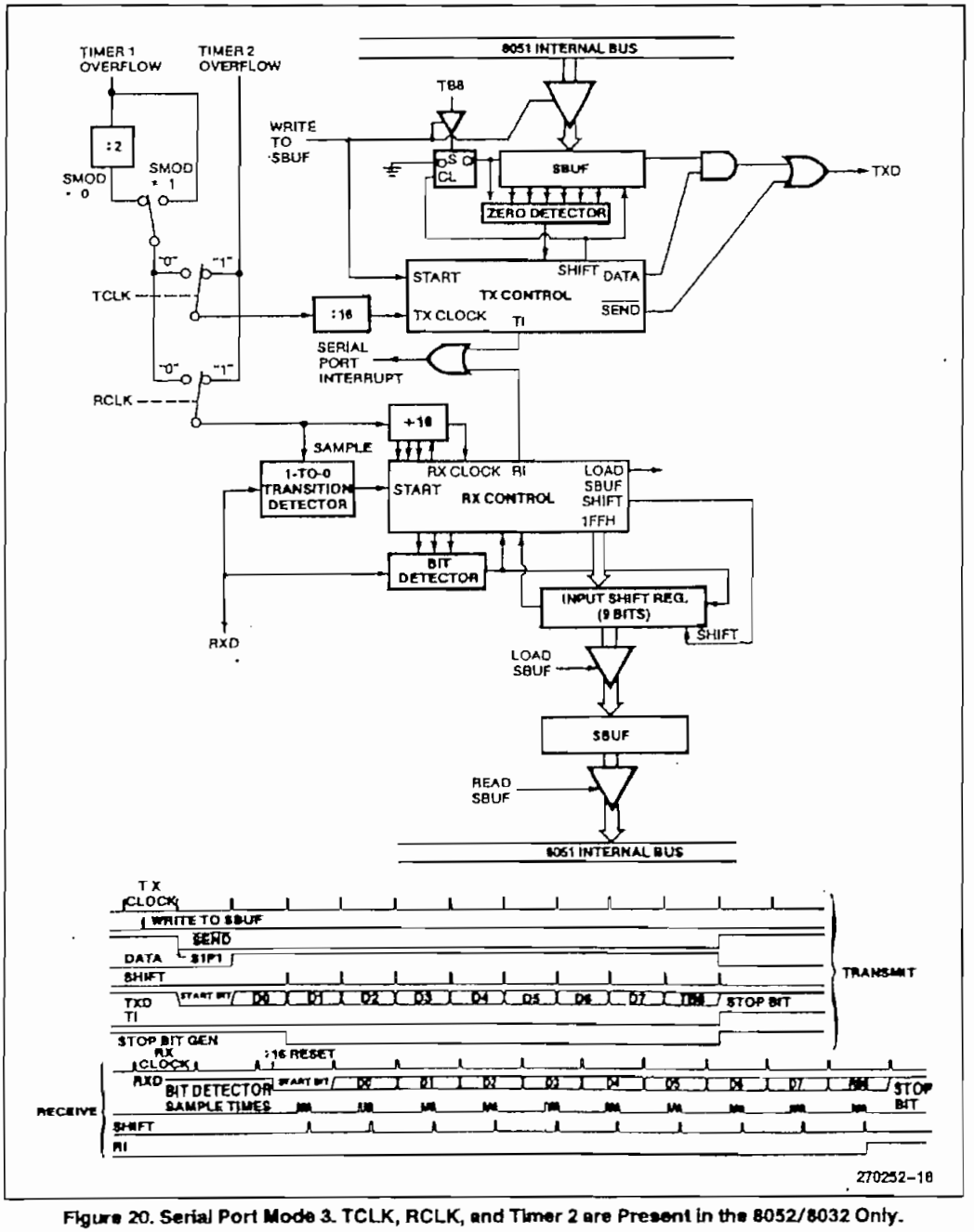


Figure 20. Serial Port Mode 3. TCLK, RCLK, and Timer 2 are Present in the 8052/8032 Only.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in Modes 2 and 3 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

1) RI = 0, and
2) Either SM2 = 0 or the received 9th data bit = 1

If either of these conditions is not met, the received frame is irretrievably lost, and RI is not set. If both conditions are met, the received 9th data bit goes into RB8, and the first 8 data bits go into SBUF. One bit time later, whether the above conditions were met or not, the unit goes back to looking for a 1-to-0 transition at the RXD input.

Note that the value of the received stop bit is irrelevant to SBUF, RB8, or RI.

INTERRUPTS

The 8051 provides 5 interrupt sources. The 8052 provides 6. These are shown in Figure 21.

The External Interrupts $\overline{INT0}$ and $\overline{INT1}$ can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in Register TCON. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to only if the interrupt was transition-activated. If the interrupt was level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

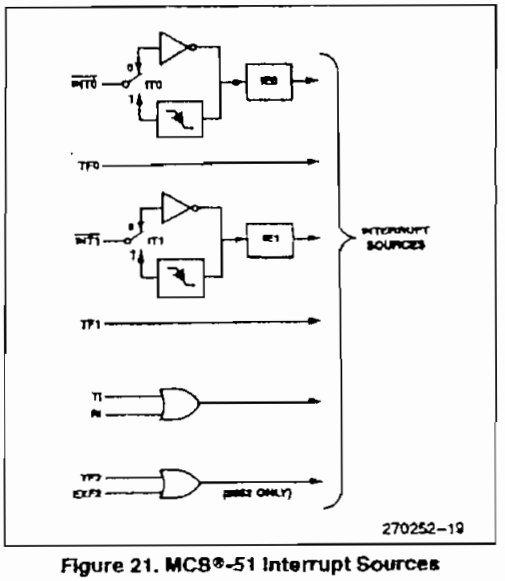


Figure 21. MCS®-51 Interrupt Sources

The Timer 0 and Timer 1 Interrupts are generated by TF0 and TF1, which are set by a rollover in their respective Timer/Counter registers (except see Timer 0 in Mode 3). When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The Serial Port Interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine will normally have to determine whether it was RI or TI that generated the interrupt, and the bit will have to be cleared in software.

In the 8052, the Timer 2 Interrupt is generated by the logical OR of TF2 and EXF2. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and the bit will have to be cleared in software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

(MSB)							(LSB)
EA	—	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.
Enable Bit = 0 disables it.

Symbol	Position	Function
EA	IE.7	disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	Serial Port interrupt enable bit.
ET1	IE.3	Timer 1 interrupt enable bit.
EX1	IE.2	External interrupt 1 enable bit.
ET0	IE.1	Timer 0 interrupt enable bit.
EX0	IE.0	External interrupt 0 enable bit.

User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

Figure 22. IE: Interrupt Enable Register

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE (Figure 22). IE contains also a global disable bit, EA, which disables all interrupts at once.

Note in Figure 22 that bit position IE.6 is unimplemented. In the 8051s, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future MCS-51 products.

Priority Level Structure

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in Special Function Register IP (Figure 23). A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

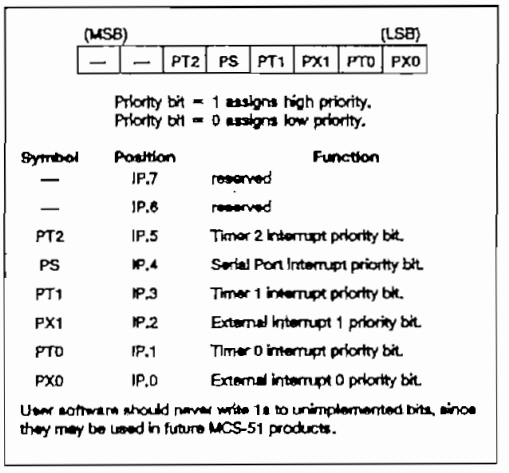


Figure 23. IP: Interrupt Priority Register

If two requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence, as follows:

	Source	Priority Within Level
1.	IE0	(highest)
2.	TF0	
3.	IE1	
4.	TF1	
5.	RI + TI	
6.	TF2 + EXF2	(lowest)

Note that the "priority within level" structure is only used to resolve *simultaneous requests of the same priority level.*

The IP register contains a number of unimplemented bits. IP.7 and IP.6 are vacant in the 8052s, and in the 8051s these and IP.5 are vacant. User software should not write 1s to these bit positions, since they may be used in future MCS-51 products.

How Interrupts Are Handled

The interrupt flags are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. The 8052's Timer 2 interrupt cycle is different, as described in the Response Time Section. If one of the flags was in a set condition at S5P2 of the preceding cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of equal or higher priority level is already in progress.
2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.
3. The instruction in progress is RETI or any write to the IE or IP registers.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be

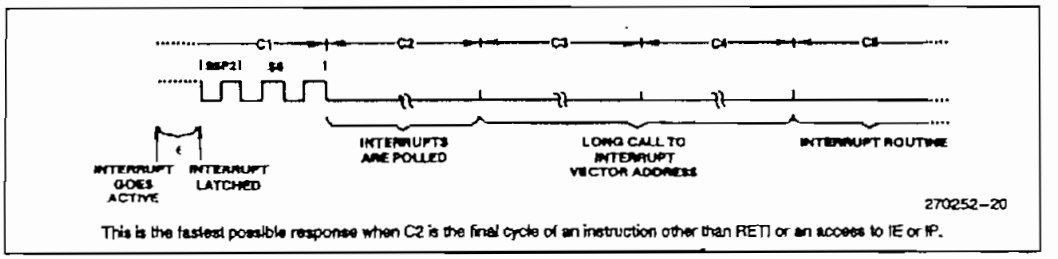


Figure 24. Interrupt Response Timing Diagram

completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any access to IE or IP, then at least *one more* instruction will be executed before any interrupt is vectored to.

The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at S5P2 of the previous machine cycle. Note then that if an interrupt flag is active but not being responded to for one of the above conditions, and is not *still* active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The polling cycle/LCALL sequence is illustrated in Figure 24.

Note that if an interrupt of higher priority level goes active prior to S5P2 of the machine cycle labeled C3 in Figure 24, then in accordance with the above rules it will be vectored to during C5 and C6, without any instruction of the lower priority routine having been executed.

Thus the processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate servicing routine. In some cases it also clears the flag that generated the interrupt, and in other cases it doesn't. It never clears the Serial Port or Timer 2 flags. This has to be done in the user's software. It clears an external interrupt flag (IE0 or IE1) only if it was transition-activated. The hardware-generated LCALL pushes the contents of the Program Counter onto the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to, as shown below.

Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI + TI	0023H
TF2 + EXF2	002BH

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the top two bytes from the stack and reloads the Program Counter. Execution of the interrupted program continues from where it left off.

Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking an interrupt was still in progress.

External Interrupts

The external sources can be programmed to be level-activated or transition-activated by setting or clearing bit IT1 or IT0 in Register TCON. If ITx = 0, external interrupt x is triggered by a detected low at the $\overline{INTx}$ pin. If ITx = 1, external interrupt x is edge-triggered. In this mode if successive samples of the $\overline{INTx}$ pin show a high in one cycle and a low in the next cycle, interrupt request flag IEx in TCON is set. Flag bit IEx then requests the interrupt.

Since the external interrupt pins are sampled once each machine cycle, an input high or low should hold for at least 12 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one machine cycle, and then hold it low for at least one machine cycle to ensure that the transition is seen so that interrupt request flag IEx will be set. IEx will be automatically cleared by the CPU when the service routine is called.

If the external interrupt is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated.

Response Time

The $\overline{INT0}$ and $\overline{INT1}$ levels are inverted and latched into the interrupt flags IE0 and IE1 at S5P2 of every machine cycle. Similarly, the Timer 2 flag EXF2 and the Serial Port flags RI and TI are set at S5P2. The values are not actually polled by the circuitry until the next machine cycle.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag TF2 is set at S2P2 and is polled in the same cycle in which the timer overflows.

If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine. Figure 24 shows interrupt response timings.

A longer response time would result if the request is blocked by one of the 3 previously listed conditions. If an interrupt of equal or higher priority level is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine. If the instruction in progress is not in its final cycle, the additional wait time cannot be more than 3 cycles, since the longest instructions (MUL and DIV) are only 4

cycles long, and if the instruction in progress is RETI or an access to IE or IP, the additional wait time cannot be more than 5 cycles (a maximum of one more cycle to complete the instruction in progress, plus 4 cycles to complete the next instruction if the instruction is MUL or DIV).

Thus, in a single-interrupt system, the response time is always more than 3 cycles and less than 9 cycles.

SINGLE-STEP OPERATION

The 8051 interrupt structure allows single-step execution with very little software overhead. As previously noted, an interrupt request will not be responded to while an interrupt of equal priority level is still in progress, nor will it be responded to after RETI until at least one other instruction has been executed. Thus, once an interrupt routine has been entered, it cannot be re-entered until at least one instruction of the interrupted program is executed. One way to use this feature for single-step operation is to program one of the external interrupts (say, $\overline{INT0}$) to be level-activated. The service routine for the interrupt will terminate with the following code:

```
JNB   P3.2,$  ;Wait Here Till INT0 Goes High
JB    P3.2,$  ;Now Wait Here Till it Goes Low
RETI          :Go Back and Execute One Instruction
```

Now if the $\overline{INT0}$ pin, which is also the P3.2 pin, is held normally low, the CPU will go right into the External Interrupt 0 routine and stay there until $\overline{INT0}$ is pulsed (from low to high to low). Then it will execute RETI, go back to the task program, execute one instruction, and immediately re-enter the External Interrupt 0 routine to await the next pulsing of P3.2. One step of the task program is executed each time P3.2 is pulsed.

RESET

The reset input is the RST pin, which is the input to a Schmitt Trigger.

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), *while the oscillator is running*. The CPU responds by generating an internal reset, with the timing shown in Figure 25.

The external reset signal is asynchronous to the internal clock. The RST pin is sampled during State 5 Phase 2 of every machine cycle. The port pins will maintain their current activities for 19 oscillator periods after a logic 1 has been sampled at the RST pin; that is, for 19 to 31 oscillator periods after the external reset signal has been applied to the RST pin.

While the RST pin is high, ALE and PSEN are weakly pulled high. After RST is pulled low, it will take 1 to 2 machine cycles for ALE and PSEN to start clocking. For this reason, other devices can not be synchronized to the internal timings of the 8051.

Driving the ALE and $\overline{PSEN}$ pins to 0 while reset is active could cause the device to go into an indeterminate state.

The internal reset algorithm writes 0s to all the SFRs except the port latches, the Stack Pointer, and SBUF. The port latches are initialized to FFH, the Stack Pointer to 07H, and SBUF is indeterminate. Table 3 lists the SFRs and their reset values.

The internal RAM is not affected by reset. On power up the RAM content is indeterminate.

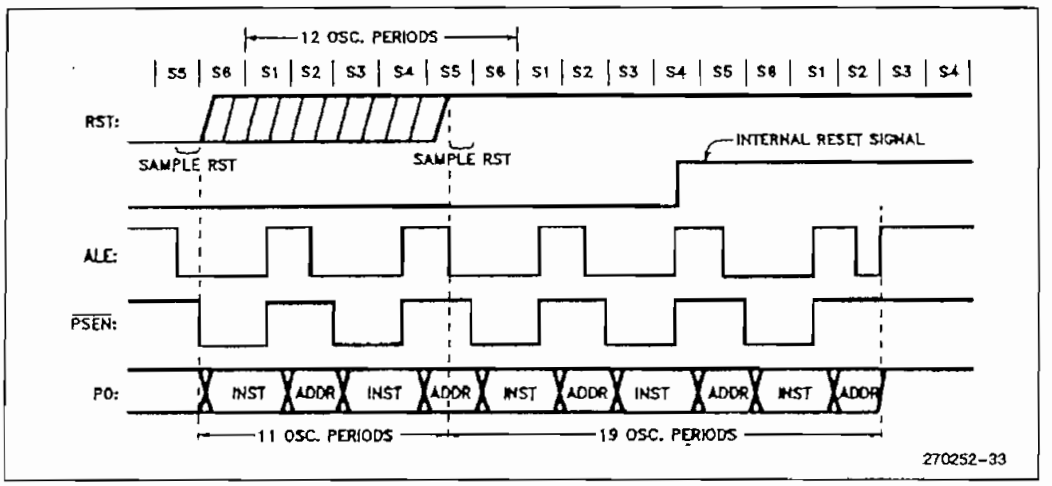


Figure 25. Reset Timing

Table 3. Reset Values of the SFRs

SFR Name	Reset Value
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0–P3	FFH
IP (8051)	XXX00000B
IP (8052)	XX000000B
IE (8051)	0XX00000B
IE (8052)	0X000000B
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
TH2 (8052)	00H
TL2 (8052)	00H
RCAP2H (8052)	00H
RCAP2L (8052)	00H
SCON	00H
SBUF	Indeterminate
PCON (HMOS)	0XXXXXXXB
PCON (CHMOS)	0XXX0000B

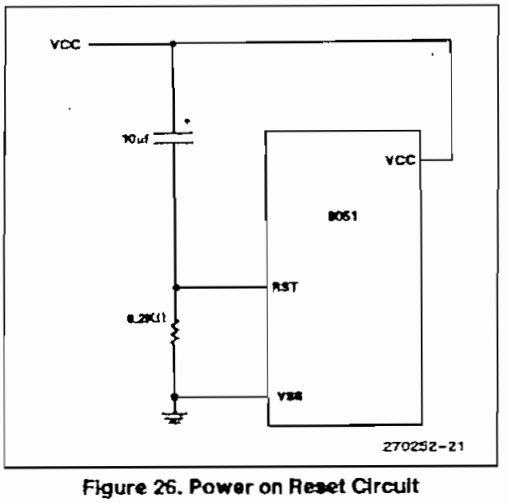


Figure 26. Power on Reset Circuit

POWER-ON RESET

For HMOS devices when V_{CC} is turned on an automatic reset can be obtained by connecting the RST pin to V_{CC} through a 10 μF capacitor and to V_{SS} through an 8.2 KΩ resistor (Figure 26). The CHMOS devices do not require this resistor although its presence does no harm. In fact, for CHMOS devices the external resistor can be removed because they have an internal pulldown on the RST pin. The capacitor value could then be reduced to 1 μF.

When power is turned on, the circuit holds the RST pin high for an amount of time that depends on the capacitor value and the rate at which it charges. To ensure a valid reset the RST pin must be held high long enough to allow the oscillator to start up plus two machine cycles.

On power up, V_{CC} should rise within approximately ten milliseconds. The oscillator start-up time will depend on the oscillator frequency. For a 10 MHz crystal, the start-up time is typically 1 ms. For a 1 MHz crystal, the start-up time is typically 10 ms.

With the given circuit, reducing V_{CC} quickly to 0 causes the RST pin voltage to momentarily fall below 0V. However, this voltage is internally limited and will not harm the device.

NOTE:
The port pins will be in a random state until the oscillator has started and the internal reset algorithm has written 1s to them.

Powering up the device without a valid reset could cause the CPU to start executing instructions from an indeterminate location. This is because the SFRs, specifically the Program Counter, may not get properly initialized.

POWER-SAVING MODES OF OPERATION

For applications where power consumption is critical the CHMOS version provides power reduced modes of operation as a standard feature. The power down mode in HMOS devices is no longer a standard feature and is being phased out.

CHMOS Power Reduction Modes

CHMOS versions have two power-reducing modes, Idle and Power Down. The input through which back-up power is supplied during these operations is VCC. Figure 27 shows the internal circuitry which implements these features. In the Idle mode (IDL = 1), the oscillator continues to run and the Interrupt, Serial Port, and Timer blocks continue to be clocked, but the

clock signal is gated off to the CPU. In Power Down (PD = 1), the oscillator is frozen. The Idle and Power Down modes are activated by setting bits in Special Function Register PCON. The address of this register is 87H. Figure 26 details its contents.

In the HMOS devices the PCON register only contains SMOD. The other four bits are implemented only in the CHMOS devices. User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

IDLE MODE

An instruction that sets PCON.0 causes that to be the last instruction executed before going into the Idle mode. In the Idle mode, the internal clock signal is gated off to the CPU, but not to the Interrupt, Timer, and Serial Port functions. The CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle. The port pins hold the logical states they had at the time Idle was activated. ALE and $\overline{PSEN}$ hold at logic high levels.

There are two ways to terminate the Idle. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating the Idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that put the device into Idle.

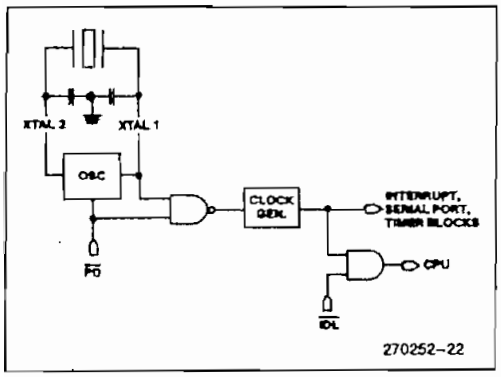


Figure 27. Idle and Power Down Hardware

(MSB)							(LSB)
SMOD	-	-	-	GF1	GF0	PD	IDL

Symbol	Position	Name and Function
SMOD	PCON.7	Double Baud rate bit. When set to a 1 and Timer 1 is used to generate baud rate, and the Serial Port is used in modes 1, 2, or 3.
—	PCON.6	(Reserved)
—	PCON.5	(Reserved)
—	PCON.4	(Reserved)
GF1	PCON.3	General-purpose flag bit.
GF0	PCON.2	General-purpose flag bit.
PD	PCON.1	Power Down bit. Setting this bit activates power down operation.
IDL	PCON.0	Idle mode bit. Setting this bit activates idle mode operation.

If 1s are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XXX0000).
In the HMOS devices the PCON register only contains SMOD. The other four bits are implemented only in the CHMOS devices. User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

Figure 28. PCON: Power Control Register

The flag bits GF0 and GF1 can be used to give an indication if an interrupt occurred during normal operation or during an Idle. For example, an instruction that activates Idle can also set one or both flag bits. When Idle is terminated by an interrupt, the interrupt service routine can examine the flag bits.

The other way of terminating the Idle mode is with a hardware reset. Since the clock oscillator is still running, the hardware reset needs to be held active for only two machine cycles (24 oscillator periods) to complete the reset.

The signal at the RST pin clears the IDL bit directly and asynchronously. At this time the CPU resumes program execution from where it left off; that is, at the instruction following the one that invoked the Idle Mode. As shown in Figure 25, two or three machine cycles of program execution may take place before the internal reset algorithm takes control. On-chip hardware inhibits access to the internal RAM during this time, but access to the port pins is not inhibited. To eliminate the possibility of unexpected outputs at the port pins, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external Data RAM.

POWER DOWN MODE

An instruction that sets PCON.1 causes that to be the last instruction executed before going into the Power Down mode. In the Power Down mode, the on-chip oscillator is stopped. With the clock frozen, all func-

Table 4. EPROM Versions of the 8051 and 8052

Device Name	EPROM Version	EPROM Bytes	Ckt Type	VPP	Time Required to Program Entire Array
8051AH	8751H/8751BH	4K	HMOS	21.0V/12.75V	4 minutes
80C51BH	87C51	4K	CHMOS	12.75V	13 seconds
8052AH	8752BH	8K	HMOS	12.75V	26 seconds

tions are stopped, but the on-chip RAM and Special Function Registers are held. The port pins output the values held by their respective SFRs. ALE and $\overline{PSEN}$ output lows.

The only exit from Power Down for the 80C51 is a hardware reset. Reset redefines all the SFRs, but does not change the on-chip RAM.

In the Power Down mode of operation, VCC can be reduced to as low as 2V. Care must be taken, however, to ensure that VCC is not reduced before the Power Down mode is invoked, and that VCC is restored to its normal operating level, before the Power Down mode is terminated. The reset that terminates Power Down also frees the oscillator. The reset should not be activated before VCC is restored to its normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize (normally less than 10 msec).

EPROM VERSIONS

The EPROM versions of these devices are listed in Table 4. The 8751H programs at VPP = 21V using one 50 msec $\overline{PROG}$ pulse per byte programmed. This results in a total programming time (4K bytes) of approximately 4 minutes.

The 8751BH, 8752BH and 87C51 use the faster "Quick-Pulse" programming™ algorithm. These devices program at VPP = 12.75V using a series of twenty-five 100 μs $\overline{PROG}$ pulses per byte programmed. This results in a total programming time of approximately 26 seconds for the 8752BH (8 Kbytes) and 13 seconds for the 87C51 (4 Kbytes).

Detailed procedures for programming and verifying each device are given in the data sheets.

Exposure to Light

It is good practice to cover the EPROM window with an opaque label when the device is in operation. This is not so much to protect the EPROM array from inadvertent erasure, but to protect the RAM and other on-chip logic. Allowing light to impinge on the silicon die while the device is operating can cause logical malfunction.

Program Memory Locks

In some microcontroller applications it is desirable that the Program Memory be secure from software piracy. Intel has responded to this need by implementing a Program Memory locking scheme in some of the MCS-51 devices. While it is impossible for anyone to guarantee absolute security against all levels of technological sophistication, the Program Memory locks in the MCS-51 devices will present a substantial barrier against illegal readout of protected software.

One Lock Bit Scheme on 8751H

The 8751H contains a lock bit which, once programmed, denies electrical access by any external means to the on-chip Program Memory. The effect of this lock bit is that 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 external Program Memory*. Erasing the EPROM array deactivates the lock bit and restores the device's full functionality. It can then be re-programmed.

The procedure for programming the lock bit is detailed in the 8751H data sheet.

Two Program Memory Lock Schemes

The 8751BH, 8752BH and 87C51 contain two Program Memory locking schemes: Encrypted Verify and Lock Bits.

Encryption Array: Within the EPROM is an array of encryption bytes that are initially unprogrammed (all 1's). The user can program the array to encrypt the code bytes during EPROM verification. The verification procedure sequentially XNORs each code byte with one of the key bytes. When the last key byte in the array is reached, the verify routine starts over with the first byte of the array for the next code byte. If the key bytes are unprogrammed, the XNOR process leaves the code byte unchanged. With the key bytes programmed, the code bytes are encrypted and can be read correctly only if the key bytes are known in their proper order. Table 6 lists the number of encryption bytes available on the various products.

When using the encryption array, one important factor should be considered. If a code byte has the value

0FFH, verifying the byte will produce the encryption byte value. If a large block of code is left unprogrammed, a verification routine will display the encryption array contents. For this reason all unused code bytes should be programmed with some value other than 0FFH, and not all of them the same value. This will ensure maximum program protection.

Program Lock Bits: Also included in the Program Lock scheme are Lock Bits which can be enabled to provide varying degrees of protection. Table 5 lists the Lock Bits and their corresponding effect on the microcontroller. Refer to Table 6 for the Lock Bits available on the various products.

Erasing the EPROM also erases the Encryption Array and the Lock Bits, returning the part to full functionality.

Table 5. Program Lock Bits and their Features

Program Lock Bits				Protection Type
	LB1	LB2	LB3	
1	U	U	U	No program lock features enabled. (Code verify will still be encrypted by the encryption array if programmed.)
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the EPROM is disabled.
3	P	P	U	Same as 2, also verify is disabled.
4	P	P	P	Same as 3, also external execution is disabled.

P-Programmed
U-Unprogrammed

Any other combination of the Lock Bits is not defined.

Table 6. Program Protection

Device	Lock Bits	Encrypt Array
8751BH	LB1, LB2	32 Bytes
8752BH	LB1, LB2	32 Bytes
87C51	LB1, LB2, LB3	64 Bytes

When Lock Bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of $\overline{EA}$ be in agreement with the current logic level at that pin in order for the device to function properly.

ROM PROTECTION

The 8051AHP and 80C51BHP are ROM Protected versions of the 8051AH and 80C51BH, respectively. To incorporate this Protection Feature, program verification has been disabled and external memory accesses have been limited to 4K. Refer to the data sheets on these parts for more information.

ONCE™ Mode

The ONCE ("on-circuit emulation") mode facilitates testing and debugging of systems using the device without the device having to be removed from the circuit. The ONCE mode is invoked by:

1. Pull ALE low while the device is in reset and $\overline{PSEN}$ is high;
2. Hold ALE low as RST is deactivated.

While the device is in ONCE mode, the Port 0 pins go into a float state, and the other port pins and ALE and $\overline{PSEN}$ are weakly pulled high. The oscillator circuit remains active. While the device is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored after a normal reset is applied.

THE ON-CHIP OSCILLATORS

HMOS Versions

The on-chip oscillator circuitry for the HMOS (HMOS-I and HMOS-II) members of the MCS-51 family is a single stage linear inverter (Figure 29), intended for use as a crystal-controlled, positive reactance oscillator (Figure 30). In this application the crystal is operated in its fundamental response mode as an inductive reactance in parallel resonance with capacitance external to the crystal.

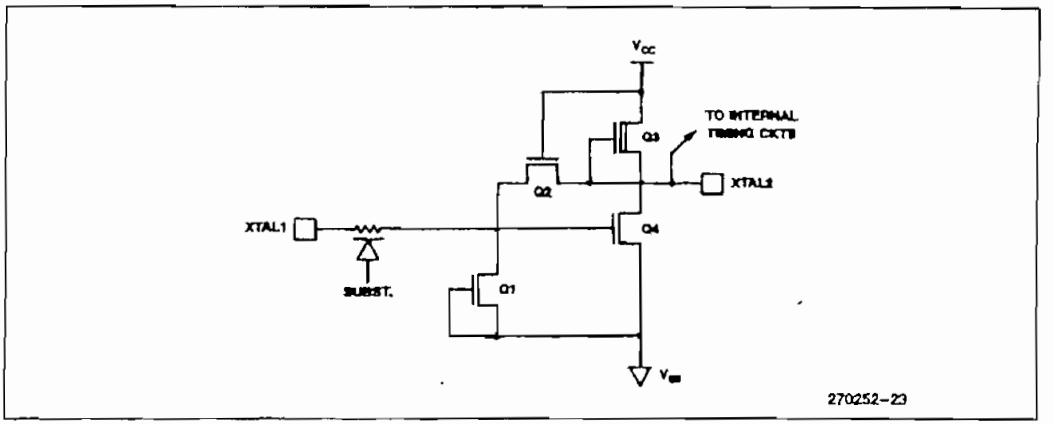


Figure 29. On-Chip Oscillator Circuitry in the HMOS Versions of the MCS®-51 Family

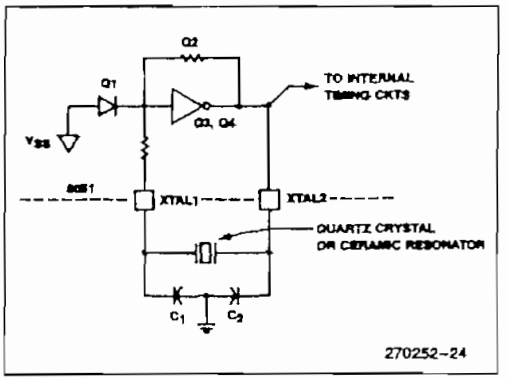


Figure 30. Using the HMOS On-Chip Oscillator

The crystal specifications and capacitance values (C1 and C2 in Figure 30) are not critical. 30 pF can be used in these positions at any frequency with good quality crystals. A ceramic resonator can be used in place of the crystal in cost-sensitive applications. When a ceramic resonator is used, C1 and C2 are normally selected to be of somewhat higher values, typically, 47 pF. The manufacturer of the ceramic resonator should be consulted for recommendations on the values of these capacitors.

In general, crystals used with these devices typically have the following specifications:

ESR (Equivalent Series Resistance)	see Figure 31
C_O (Shunt Capacitance)	7.0 pF max.
C_L (Load Capacitance)	30 pF ±3 pF
Drive Level	1 mW

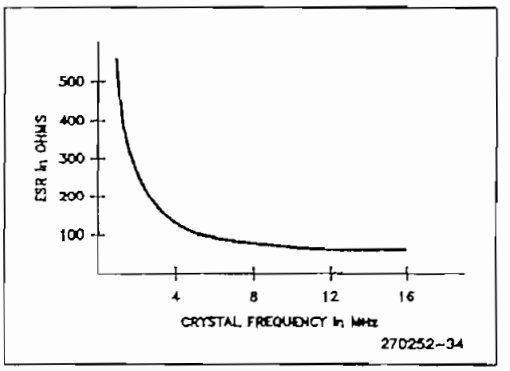


Figure 31. ESR vs Frequency

Frequency, tolerance and temperature range are determined by the system requirements.

A more in-depth discussion of crystal specifications, ceramic resonators, and the selection of values for C1 and C2 can be found in Application Note AP-155, "Oscillators for Microcontrollers," which is included in the *Embedded Applications Handbook*.

To drive the HMOS parts with an external clock source, apply the external clock signal to XTAL2, and ground XTAL1, as shown in Figure 32. A pullup resistor may be used (to increase noise margin), but is optional if VOH of the driving gate exceeds the VIH MIN specification of XTAL2.

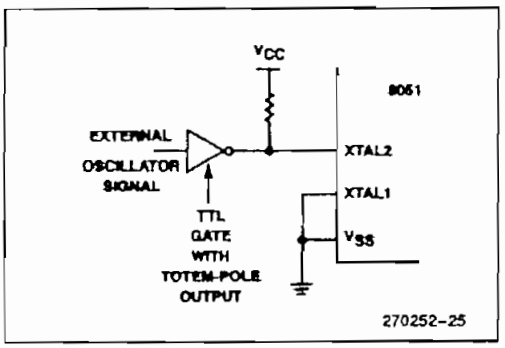


Figure 32. Driving the HMOS MCS®-51 Parts with an External Clock Source

CHMOS Versions

The on-chip oscillator circuitry for the 80C51BH, shown in Figure 33, consists of a single stage linear inverter intended for use as a crystal-controlled, positive reactance oscillator in the same manner as the HMOS parts. However, there are some important differences.

One difference is that the 80C51BH is able to turn off its oscillator under software control (by writing a 1 to the PD bit in PCON). Another difference is that in the 80C51BH the internal clocking circuitry is driven by the signal at XTAL1, whereas in the HMOS versions it is by the signal at XTAL2.

The feedback resistor R_f in Figure 33 consists of paralleled n- and p- channel FETs controlled by the PD bit, such that R_f is opened when PD = 1. The diodes D1 and D2, which act as clamps to VCC and VSS, are parasitic to the R_f FETs.

The oscillator can be used with the same external components as the HMOS versions, as shown in Figure 34. Typically, C1 = C2 = 30 pF when the feedback element is a quartz crystal, and C1 = C2 = 47 pF when a ceramic resonator is used.

To drive the CHMOS parts with an external clock source, apply the external clock signal to XTAL1, and leave XTAL2 float, as shown in Figure 35.

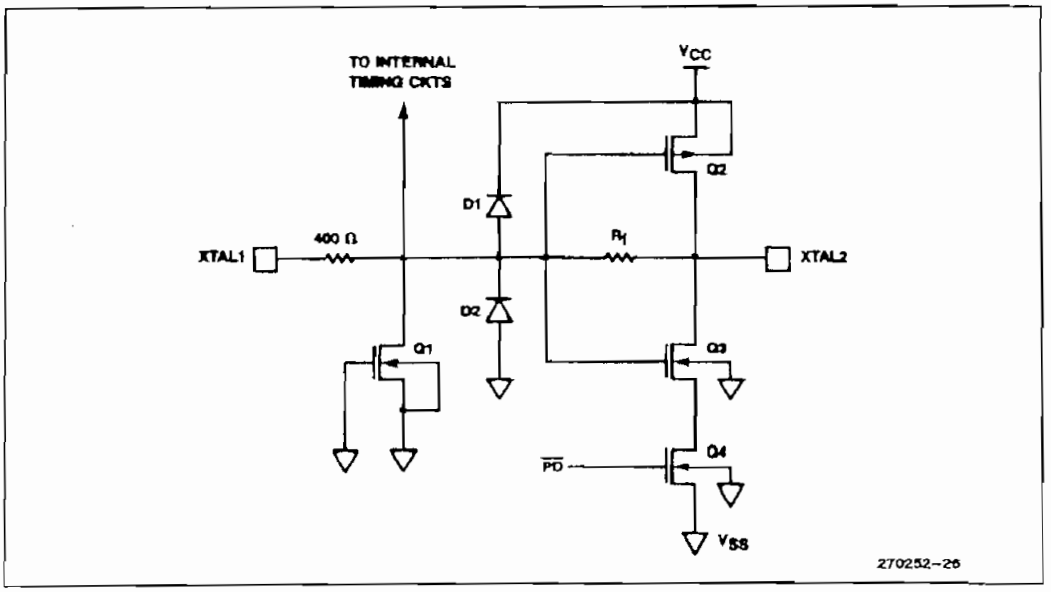


Figure 33. On-Chip Oscillator Circuitry in the CHMOS Versions of the MCS®-51 Family

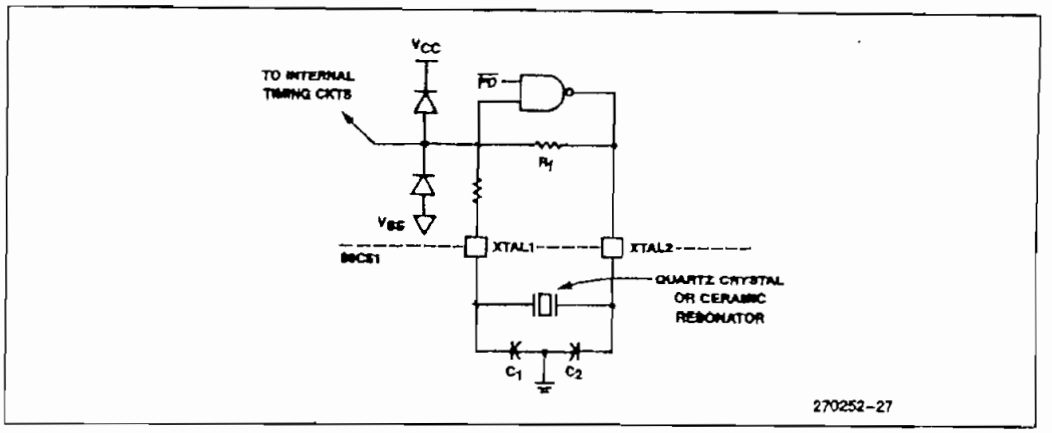


Figure 34. Using the CHMOS On-Chip Oscillator

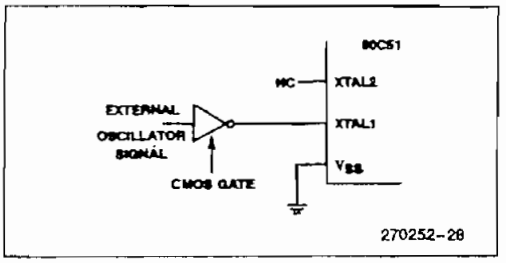


Figure 35. Driving the CHMOS MCS®-51 Parts with an External Clock Source

The reason for this change from the way the HMOS part is driven can be seen by comparing Figures 29 and 33. In the HMOS devices the internal timing circuits are driven by the signal at XTAL2. In the CHMOS devices the internal timing circuits are driven by the signal at XTAL1.

INTERNAL TIMING

Figures 36 through 39 show when the various strobe and port signals are clocked internally. The figures do not show rise and fall times of the signals, nor do they show propagation delays between the XTAL signal and events at other pins.

Rise and fall times are dependent on the external loading that each pin must drive. They are often taken to be something in the neighborhood of 10 nsec, measured between 0.8V and 2.0V.

Propagation delays are different for different pins. For a given pin they vary with pin loading, temperature, VCC, and manufacturing lot. If the XTAL waveform is taken as the timing reference, prop delays may vary from 25 to 125 nsec.

The AC Timings section of the data sheets do not reference any timing to the XTAL waveform. Rather, they relate the critical edges of control and input signals to each other. The timings published in the data sheets include the effects of propagation delays under the specified test conditions.

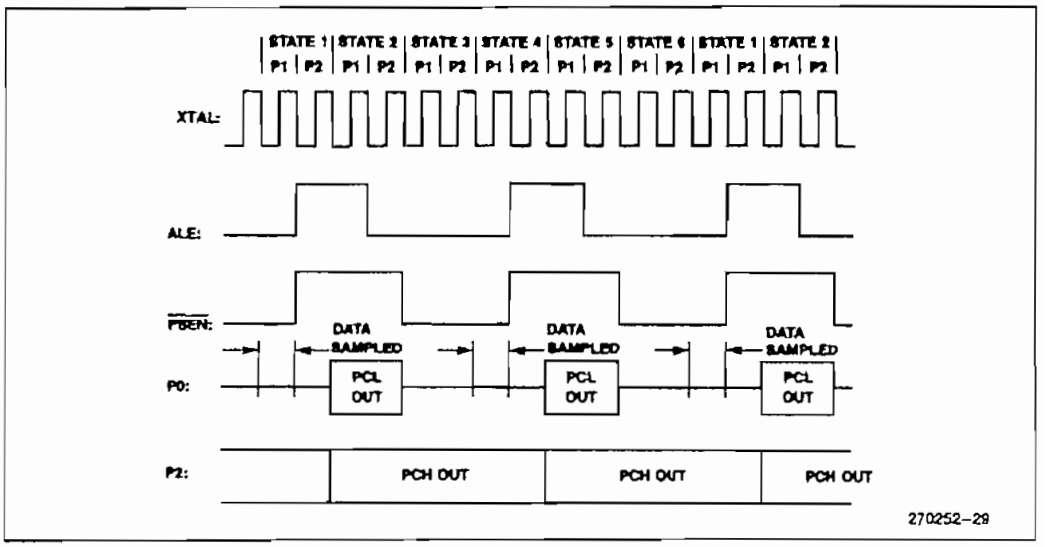


Figure 36. External Program Memory Fetches

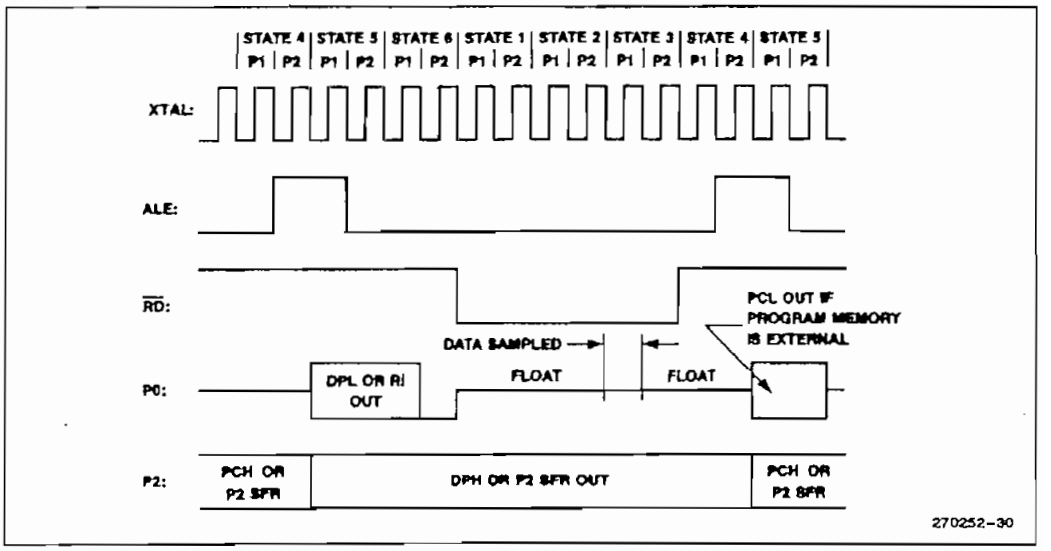


Figure 37. External Data Memory Read Cycle

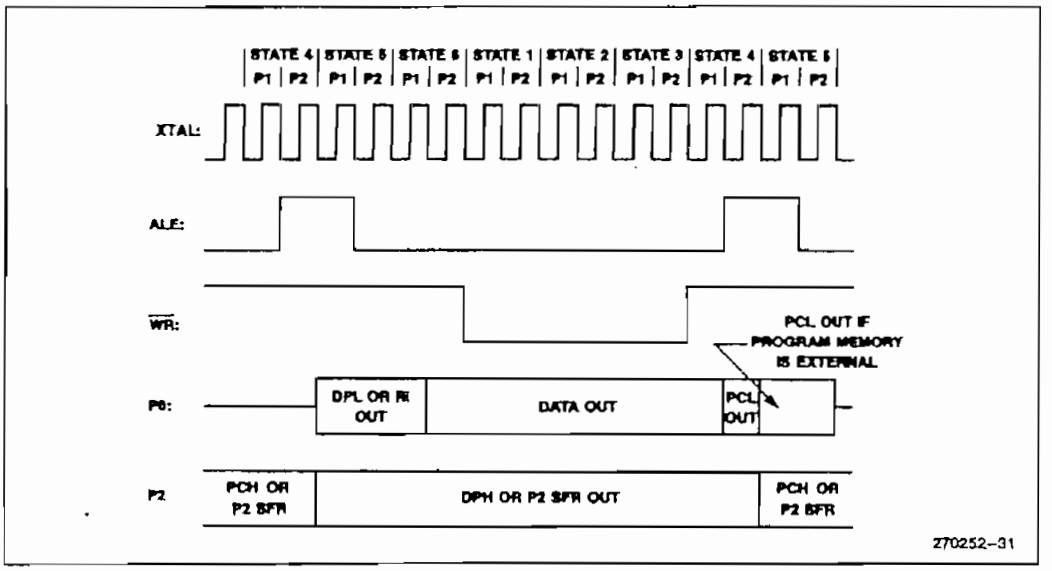


Figure 38. External Data Memory Write Cycle

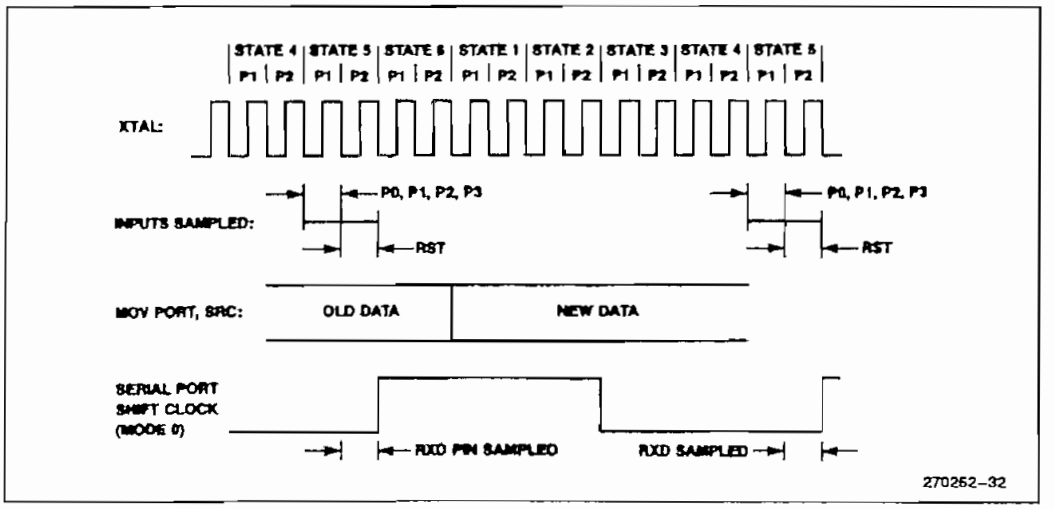


Figure 39. Port Operation

ADDITIONAL REFERENCES

The following application notes and articles are found in the *Embedded Applications* handbook. (Order Number: 270648)

1. AP-125 "Designing Microcontroller Systems for Electrically Noisy Environments".
2. AP-155 "Oscillators for Microcontrollers".
3. AP-252 "Designing with the 80C51BH".
4. AR-517 "Using the 8051 Microcontroller with Resonant Transducers".

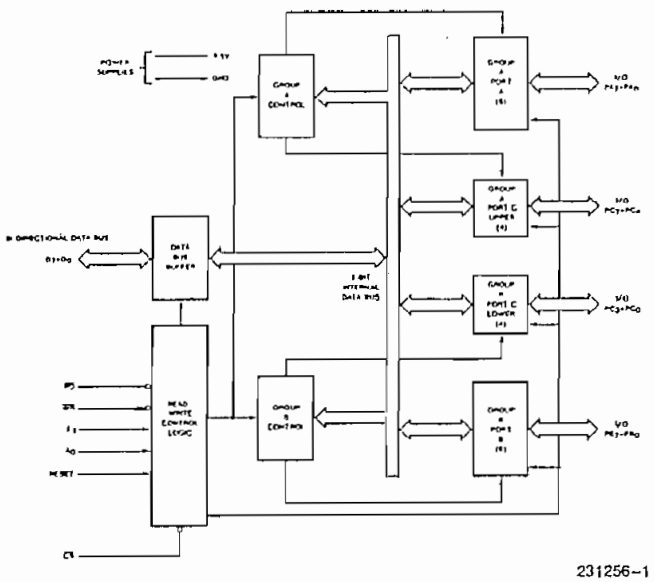
82C55A
CHMOS PROGRAMMABLE PERIPHERAL INTERFACE

- Compatible with all Intel and Most Other Microprocessors
- High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188
- 24 Programmable I/O Pins
- Low Power CHMOS
- Completely TTL Compatible
- Control Word Read-Back Capability
- Direct Bit Set/Reset Capability
- 2.5 mA DC Drive Capability on all I/O Port Outputs
- Available in 40-Pin DIP and 44-Pin PLCC
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel 82C55A is a high-performance, CHMOS version of the industry standard 8255A general purpose programmable I/O device which is designed for use with all Intel and most other microprocessors. It provides 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. The 82C55A is pin compatible with the NMOS 8255A and 8255A-5.

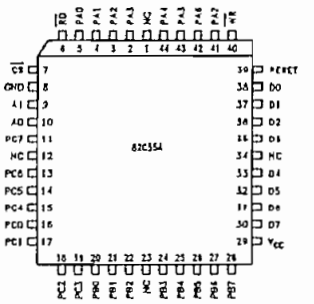
In MODE 0, each group of 12 I/O pins may be programmed in sets of 4 and 8 to be inputs or outputs. In MODE 1, each group may be programmed to have 8 lines of input or output. 3 of the remaining 4 pins are used for handshaking and interrupt control signals. MODE 2 is a strobed bi-directional bus configuration.

The 82C55A is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent NMOS product. The 82C55A is available in 40-pin DIP and 44-pin plastic leaded chip carrier (PLCC) packages.

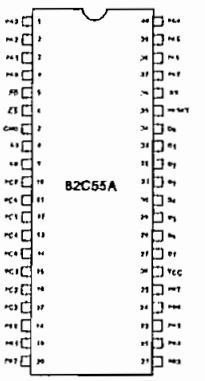


231256-1

Figure 1. 82C55A Block Diagram



231256-31



231256-2

Figure 2. 82C55A Pinout

Diagrams are for pin reference only. Package sizes are not to scale.

Table 1. Pin Description

Symbol	Pin Number Dip	Pin Number PLCC	Type	Name and Function
PA_{3-0}	1–4	2–5	I/O	**PORT A, PINS 0–3:** Lower nibble of an 8-bit data output latch/buffer and an 8-bit data input latch.
$\overline{RD}$	5	6	I	**READ CONTROL:** This input is low during CPU read operations.
$\overline{CS}$	6	7	I	**CHIP SELECT:** A low on this input enables the 82C55A to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and WR are ignored otherwise.
GND	7	8		**System Ground**
A_{1-0}	8–9	9–10	I	**ADDRESS:** These input signals, in conjunction $\overline{RD}$ and $\overline{WR}$, control the selection of one of the three ports or the control word registers.

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Input Operation (Read)
0	0	0	1	0	Port A - Data Bus
0	1	0	1	0	Port B - Data Bus
1	0	0	1	0	Port C - Data Bus
1	1	0	1	0	Control Word - Data Bus
					Output Operation (Write)
0	0	1	0	0	Data Bus - Port A
0	1	1	0	0	Data Bus - Port B
1	0	1	0	0	Data Bus - Port C
1	1	1	0	0	Data Bus - Control
					Disable Function
X	X	X	X	1	Data Bus - 3 - State
X	X	1	1	0	Data Bus - 3 - State

PC_{7-4}	10–13	11,13–15	I/O	**PORT C, PINS 4–7:** Upper nibble of an 8-bit data output latch/buffer and an 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.
PC_{0-3}	14–17	16–19	I/O	**PORT C, PINS 0–3:** Lower nibble of Port C.
PB_{0-7}	18–25	20–22, 24–28	I/O	**PORT B, PINS 0–7:** An 8-bit data output latch/buffer and an 8-bit data input buffer.
V_{CC}	26	29		**SYSTEM POWER:** + 5V Power Supply.
D_{7-0}	27–34	30–33, 35–38	I/O	**DATA BUS:** Bi-directional, tri-state data bus lines, connected to system data bus.
RESET	35	39	I	**RESET:** A high on this input clears the control register and all ports are set to the input mode.
$\overline{WR}$	36	40	I	**WRITE CONTROL:** This input is low during CPU write operations.
PA_{7-4}	37–40	41–44	I/O	**PORT A, PINS 4–7:** Upper nibble of an 8-bit data output latch/buffer and an 8-bit data input latch.
NC		1, 12, 23, 34		No Connect

82C55A FUNCTIONAL DESCRIPTION

General

The 82C55A is a programmable peripheral interface device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 82C55A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 82C55A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 82C55A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 82C55A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A - Port A and Port C upper (C7–C4)
Control Group B - Port B and Port C lower (C3–C0)

The control word register can be both written and read as shown in the address decode table in the pin descriptions. Figure 6 shows the control word format for both Read and Write operations. When the control word is read, bit D7 will always be a logic "1", as this implies control word mode information.

Ports A, B, and C

The 82C55A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 82C55A.

Port A. One 8-bit data output latch/buffer and one 8-bit input latch buffer. Both "pull-up" and "pull-down" bus hold devices are present on Port A.

Port B. One 8-bit data input/output latch/buffer. Only "pull-up" bus hold devices are present on Port B.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B. Only "pull-up" bus hold devices are present on Port C.

See Figure 4 for the bus-hold circuit configuration for Port A, B, and C.

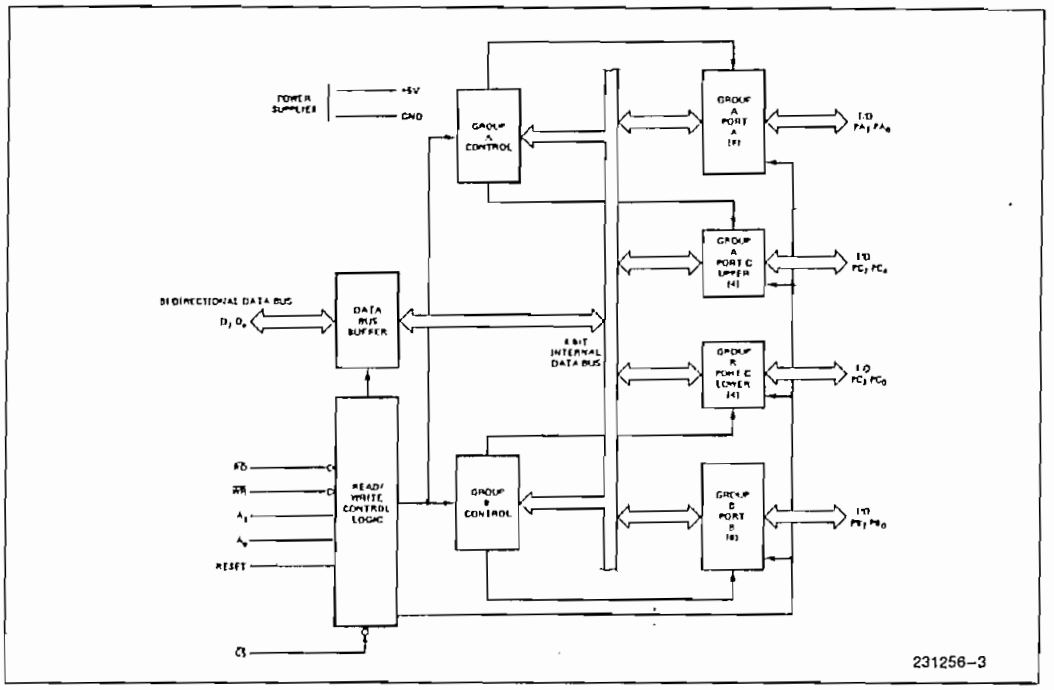
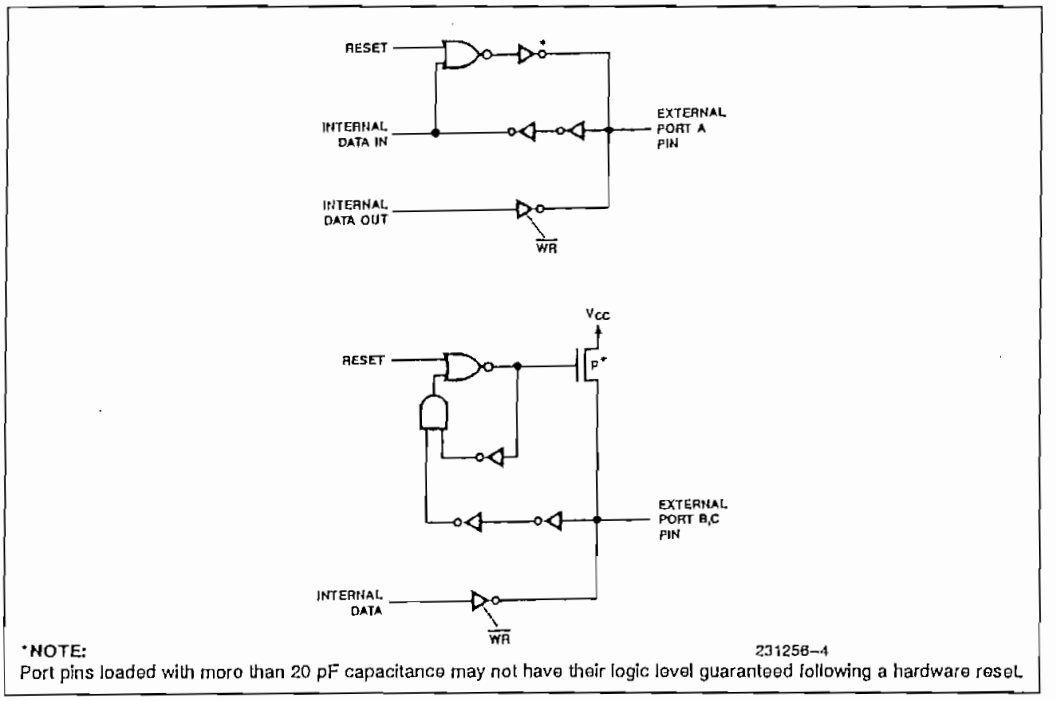


Figure 3. 82C55A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions



*NOTE:
Port pins loaded with more than 20 pF capacitance may not have their logic level guaranteed following a hardware reset.

Figure 4. Port A, B, C, Bus-hold Configuration

82C55A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0 — Basic input/output
Mode 1 — Strobed Input/output
Mode 2 — Bi-directional Bus

When the reset input goes "high" all ports will be set to the input mode with all 24 port lines held at a logic "one" level by the internal bus hold devices (see Figure 4 Note). After the reset is removed the 82C55A can remain in the input mode with no additional initialization required. This eliminates the need for pullup or pulldown devices in "all CMOS" designs. During the execution of the system program, any of the other modes may be selected by using a single output instruction. This allows a single 82C55A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

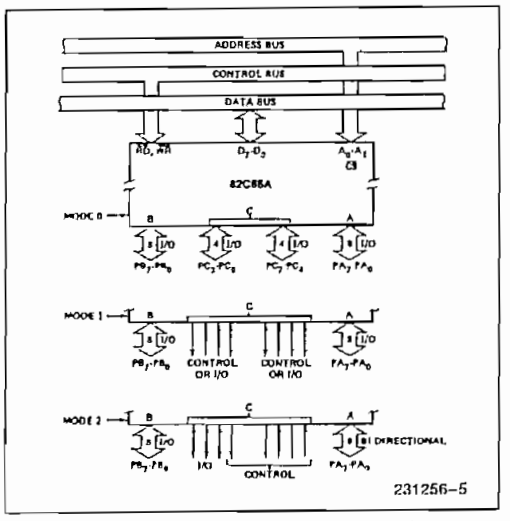


Figure 5. Basic Mode Definitions and Bus Interface

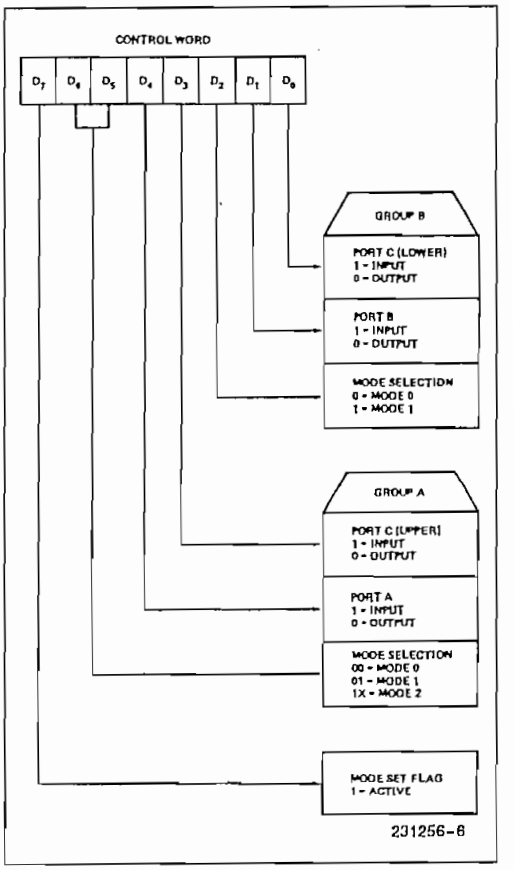


Figure 6. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 82C55A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

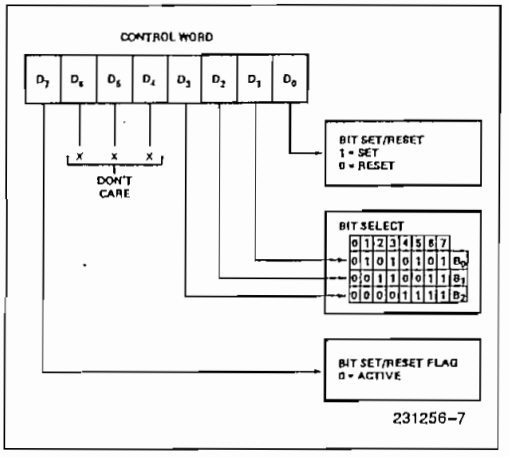


Figure 7. Bit Set/Reset Format

Interrupt Control Functions

When the 82C55A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET)—INTE is SET—Interrupt enable
(BIT-RESET)—INTE is RESET—Interrupt disable

Note:
All Mask flip-flops are automatically reset during mode selection and device Reset.

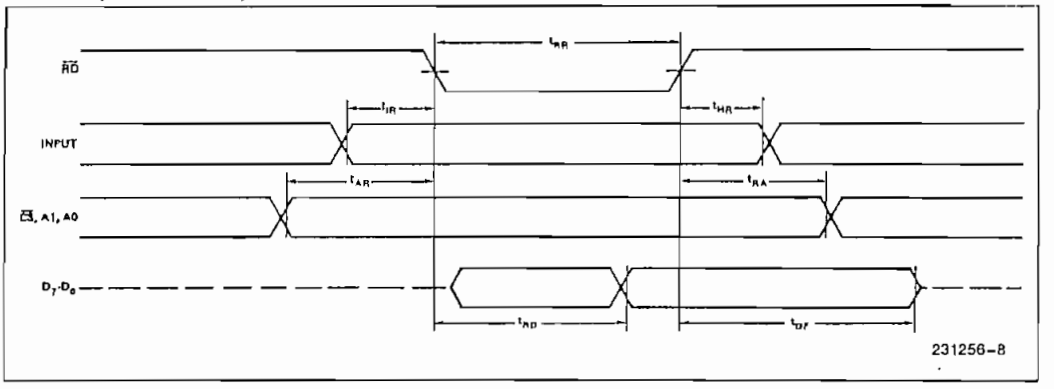
Operating Modes

Mode 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

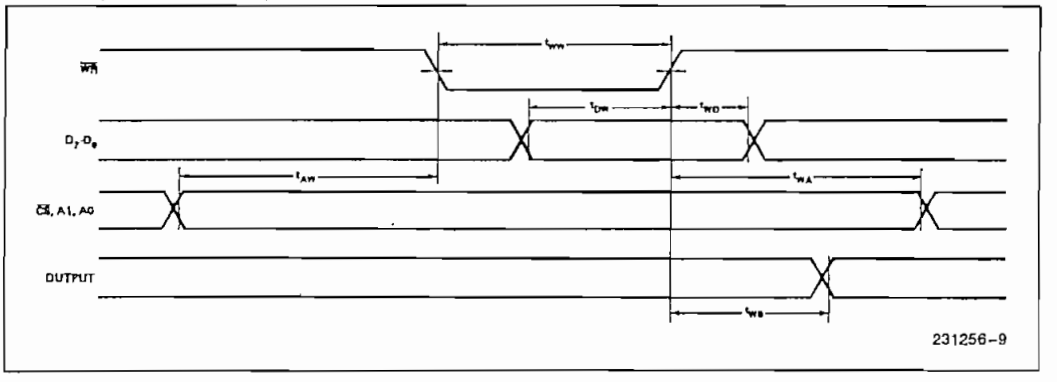
Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 (BASIC INPUT)



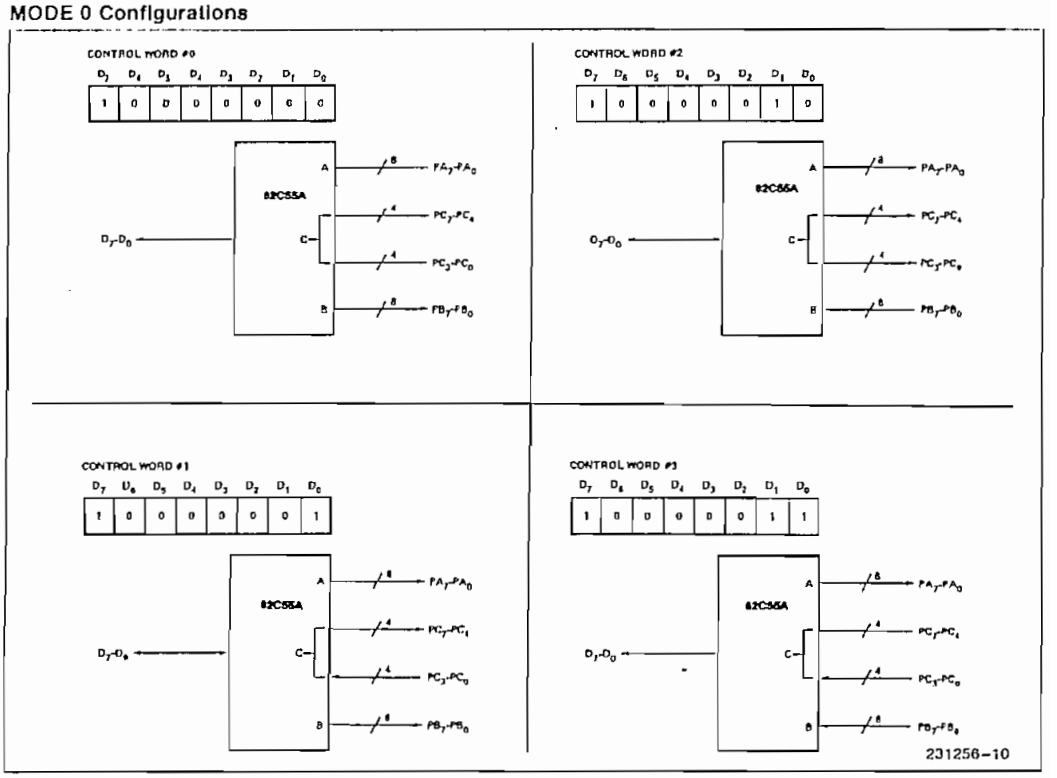
MODE 0 (BASIC OUTPUT)



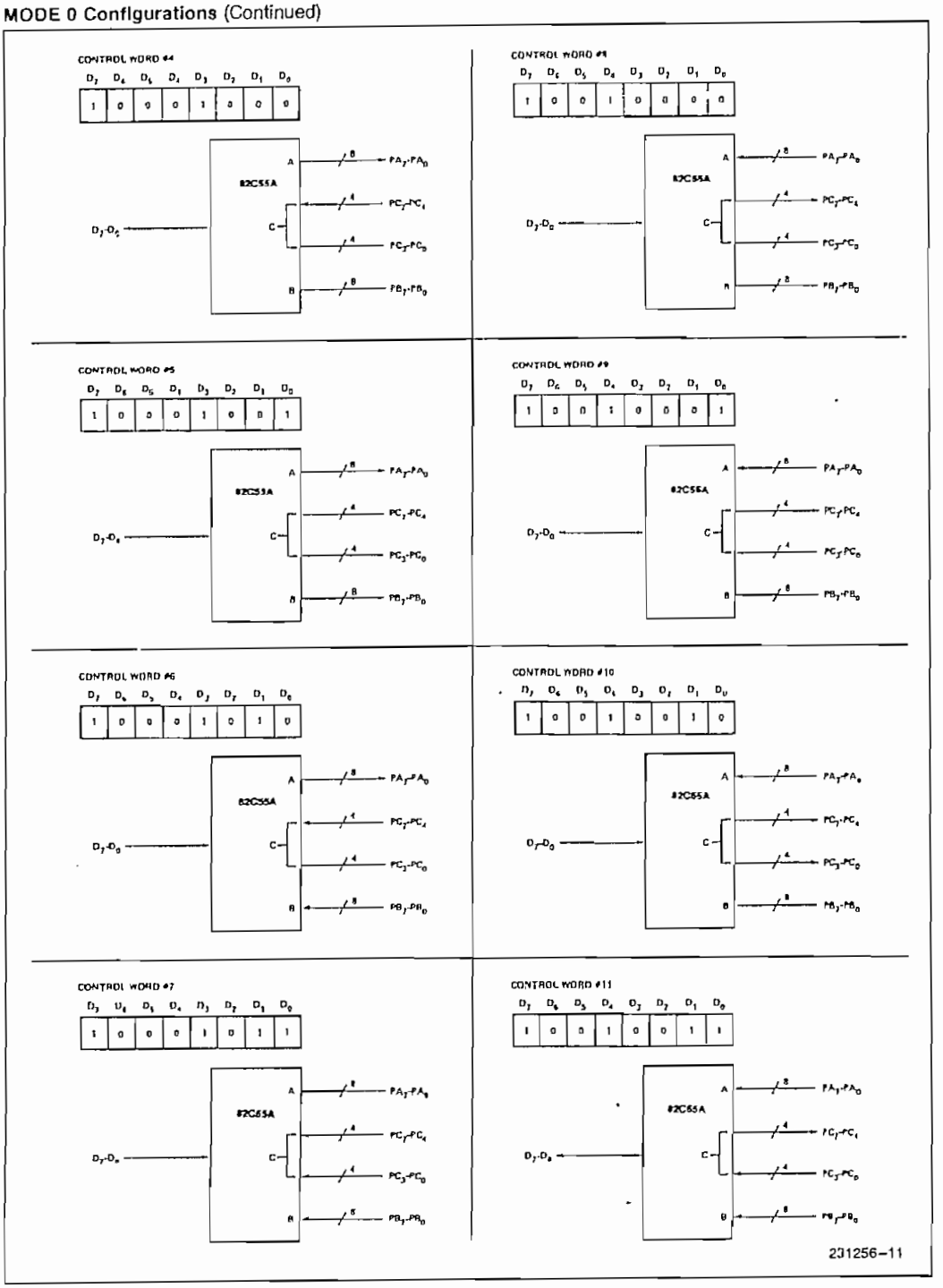
MODE 0 Port Definition

A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

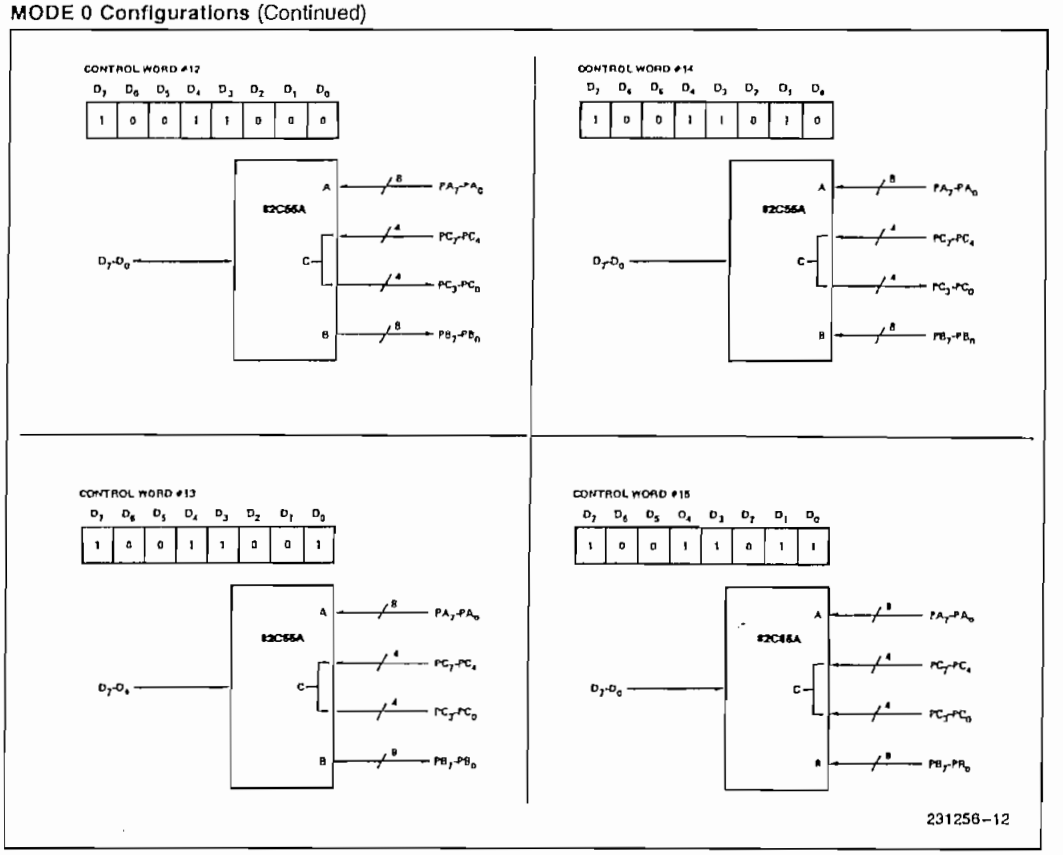
MODE 0 Configurations



MODE 0 Configurations (Continued)



MODE 0 Configurations (Continued)



Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, Port A and Port B use the lines on Port C to generate or accept these "handshaking" signals.

Mode 1 Basic functional Definitions:

- Two Groups (Group A and Group B).
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Input Control Signal Definition

$\overline{STB}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by $\overline{STB}$ input being low and is reset by the rising edge of the $\overline{RD}$ input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{STB}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{RD}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC_4.

INTE B

Controlled by bit set/reset of PC_2.

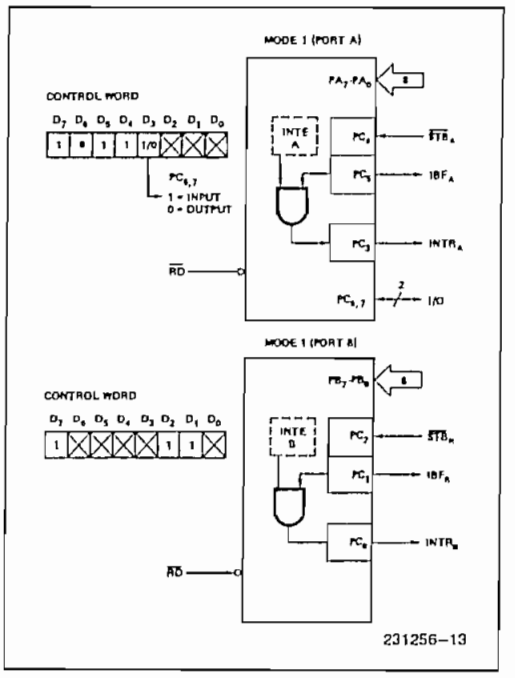


Figure 8. MODE 1 Input

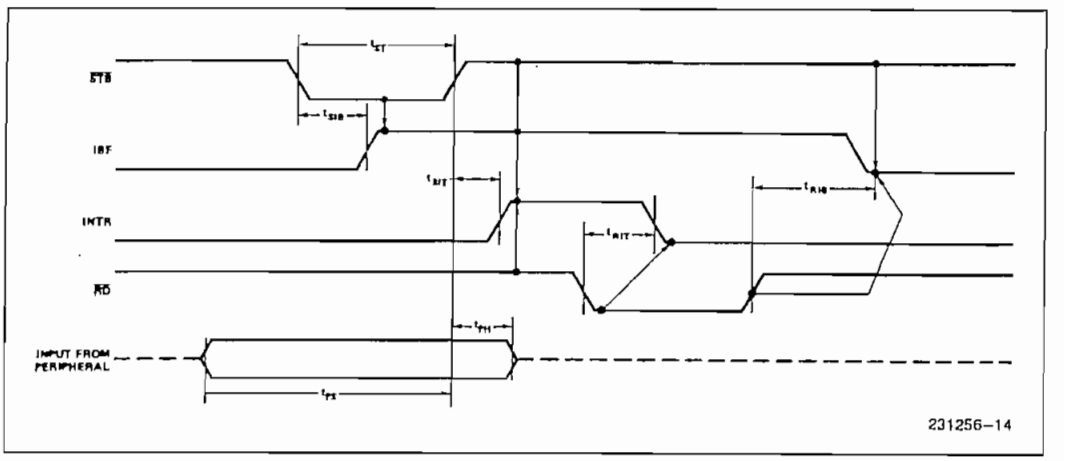


Figure 9. MODE 1 (Strobed Input)

Output Control Signal Definition

$\overline{OBF}$ (Output Buffer Full F/F). The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to the specified port. The $\overline{OBF}$ F/F will be set by the rising edge of the $\overline{WR}$ input and reset by $\overline{ACK}$ input being low.

$\overline{ACK}$ (Acknowledge Input). A "low" on this input informs the 82C55A that the data from Port A or Port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request). A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when $\overline{ACK}$ is a "one", $\overline{OBF}$ is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{WR}$.

INTE A

Controlled by bit set/reset of PC_6.

INTE B

Controlled by bit set/reset of PC_2.

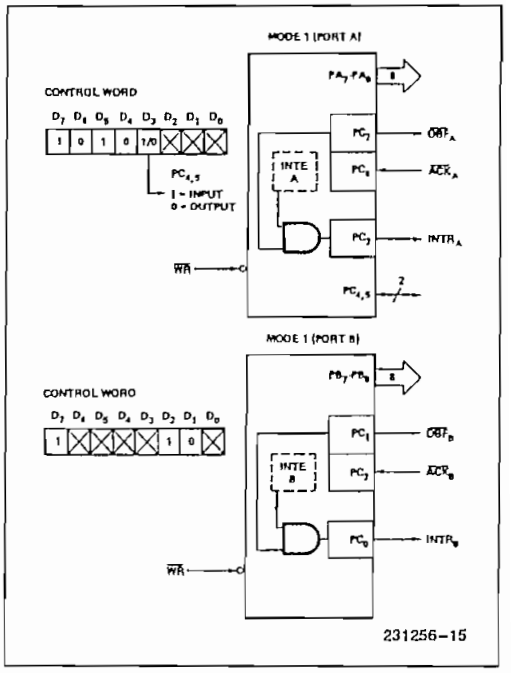


Figure 10. MODE 1 Output

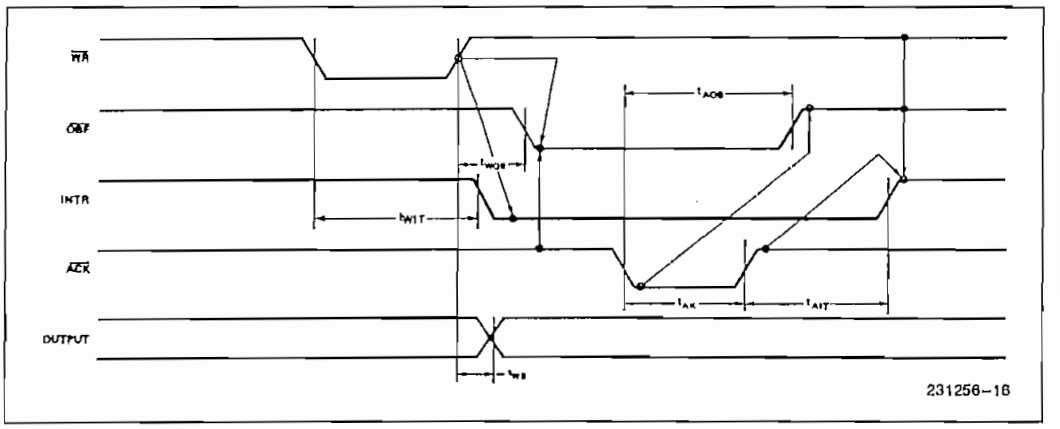


Figure 11. MODE 1 (Strobed Output)

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

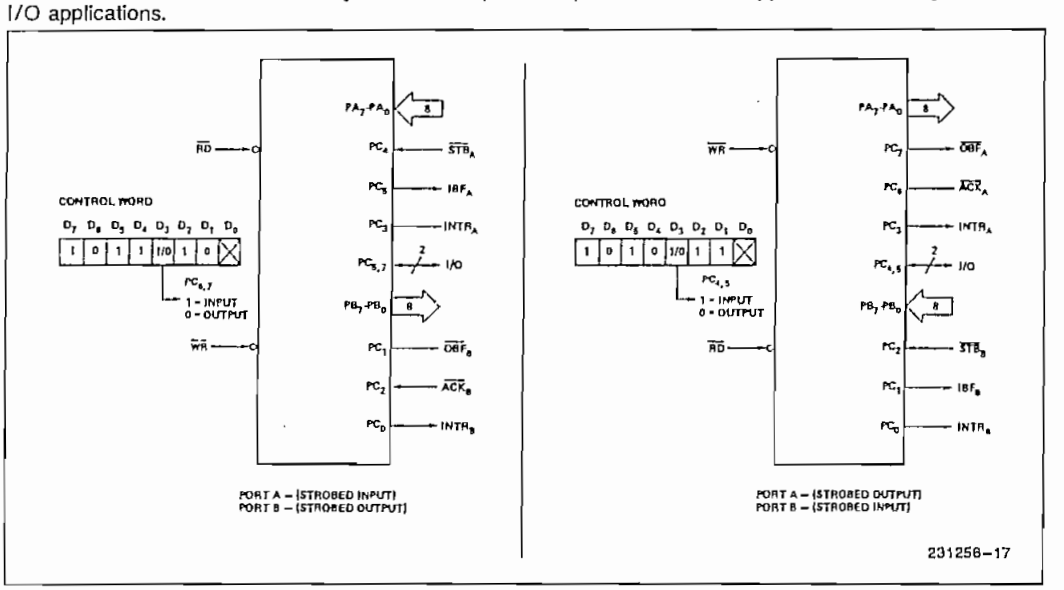


Figure 12. Combinations of MODE 1

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A only.
- One 8-bit, bi-directional bus port (Port A) and a 5-bit control port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for input or output operations.

Output Operations

$\overline{OBF}$ (Output Buffer Full). The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to port A.

$\overline{ACK}$ (Acknowledge). A "low" on this input enables the tri-state output buffer of Port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with $\overline{OBF}$). Controlled by bit set/reset of PC_6.

Input Operations

$\overline{STB}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC_4.

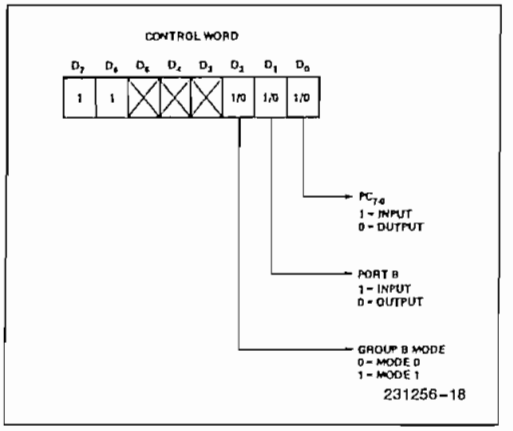


Figure 13. MODE Control Word

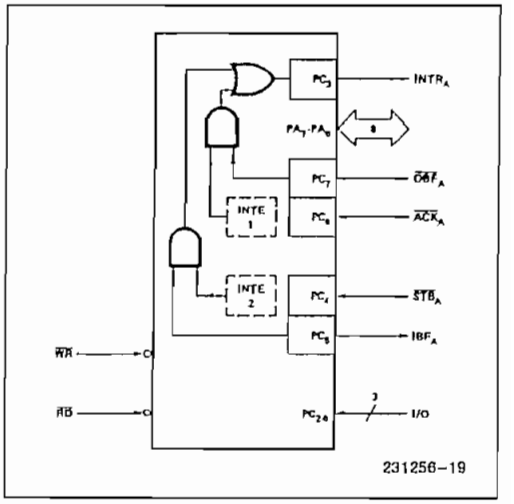


Figure 14. MODE 2

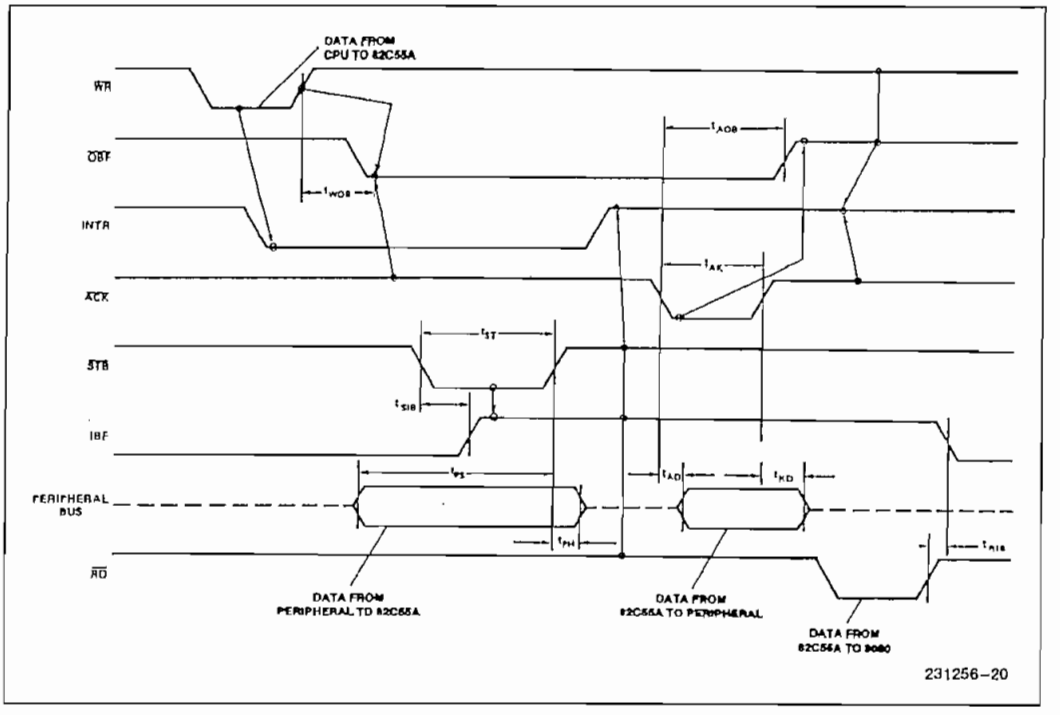


Figure 15. MODE 2 (Bidirectional)

NOTE:
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$, and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

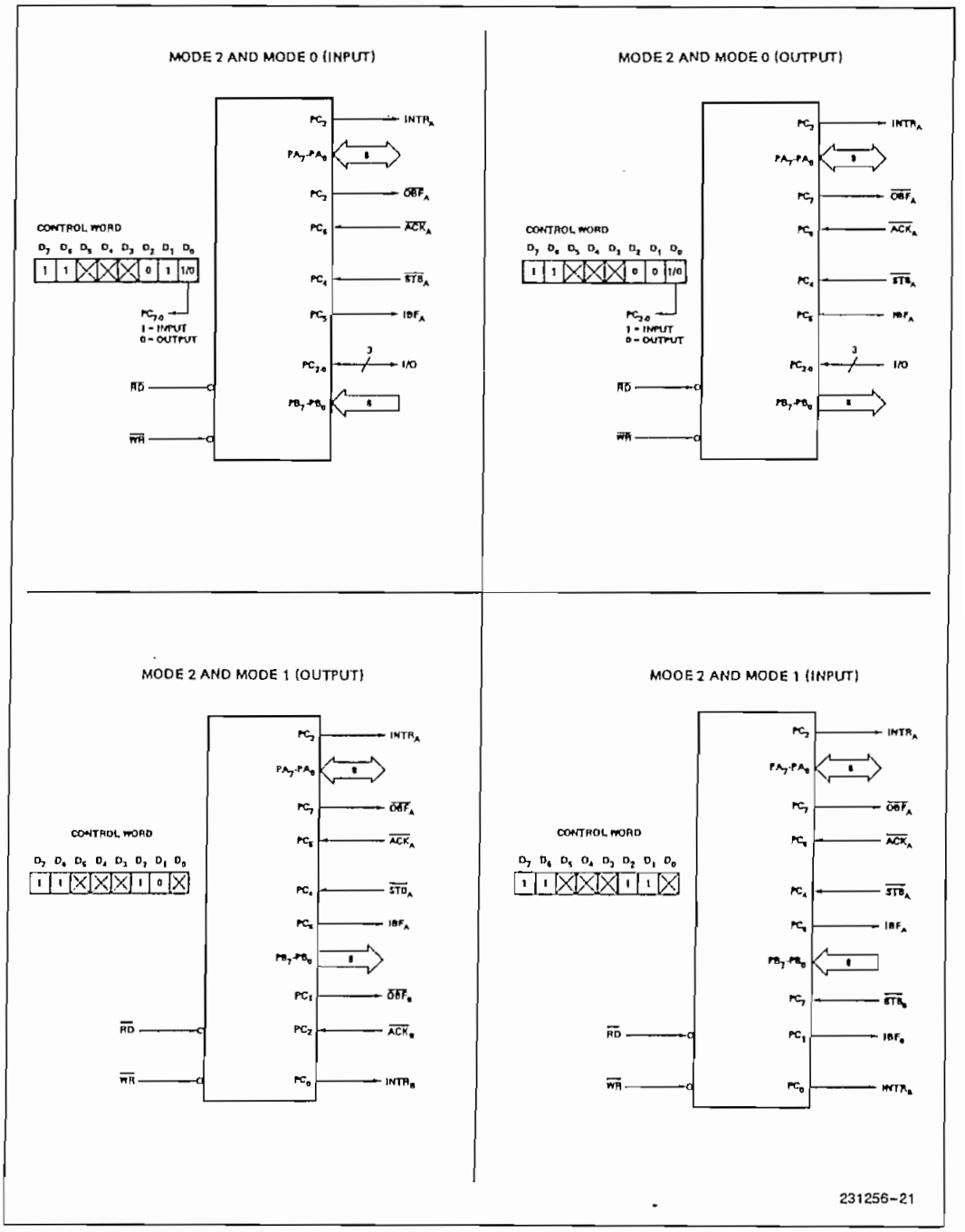


Figure 16. MODE 1/4 Combinations

Mode Definition Summary

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA_0	IN	OUT	IN	OUT	↔	
PA_1	IN	OUT	IN	OUT	↔	
PA_2	IN	OUT	IN	OUT	↔	
PA_3	IN	OUT	IN	OUT	↔	
PA_4	IN	OUT	IN	OUT	↔	
PA_5	IN	OUT	IN	OUT	↔	
PA_6	IN	OUT	IN	OUT	↔	
PA_7	IN	OUT	IN	OUT	↔	
PB_0	IN	OUT	IN	OUT	—	MODE 0 OR MODE 1 ONLY
PB_1	IN	OUT	IN	OUT	—	
PB_2	IN	OUT	IN	OUT	—	
PB_3	IN	OUT	IN	OUT	—	
PB_4	IN	OUT	IN	OUT	—	
PB_5	IN	OUT	IN	OUT	—	
PB_6	IN	OUT	IN	OUT	—	
PB_7	IN	OUT	IN	OUT	—	
PC_0	IN	OUT	$INTR_B$	$INTR_B$	I/O	
PC_1	IN	OUT	IBF_B	$\overline{OBF}_B$	I/O	
PC_2	IN	OUT	$\overline{STB}_B$	$\overline{ACK}_B$	I/O	
PC_3	IN	OUT	$INTR_A$	$INTR_A$	$INTR_A$	
PC_4	IN	OUT	$\overline{STB}_A$	I/O	$\overline{STB}_A$	
PC_5	IN	OUT	IBF_A	I/O	IBF_A	
PC_6	IN	OUT	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$	
PC_7	IN	OUT	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$	

Special Mode Combination Considerations

There are several combinations of modes possible. For any combination, some or all of the Port C lines are used for control or status. The remaining bits are either inputs or outputs as defined by a "Set Mode" command.

During a read of Port C, the state of all the Port C lines, except the $\overline{ACK}$ and $\overline{STB}$ lines, will be placed on the data bus. In place of the $\overline{ACK}$ and $\overline{STB}$ line states, flag status will appear on the data bus in the PC2, PC4, and PC6 bit positions as illustrated by Figure 18.

Through a "Write Port C" command, only the Port C pins programmed as outputs in a Mode 0 group can be written. No other pins can be affected by a "Write Port C" command, nor can the interrupt enable flags be accessed. To write to any Port C output programmed as an output in a Mode 1 group or to change an interrupt enable flag, the "Set/Reset Port C Bit" command must be used.

With a "Set/Reset Port C Bit" command, any Port C line programmed as an output (including INTR, IBF and $\overline{OBF}$) can be written, or an interrupt enable flag can be either set or reset. Port C lines programmed as inputs, including $\overline{ACK}$ and $\overline{STB}$ lines, associated with Port C are not affected by a "Set/Reset Port C Bit" command. Writing to the corresponding Port C bit positions of the $\overline{ACK}$ and $\overline{STB}$ lines with the "Set/Reset Port C Bit" command will affect the Group A and Group B interrupt enable flags, as illustrated in Figure 18.

Current Drive Capability

Any output on Port A, B or C can sink or source 2.5 mA. This feature allows the 82C55A to directly drive Darlington type drivers and high-voltage displays that require such sink or source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 82C55A is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

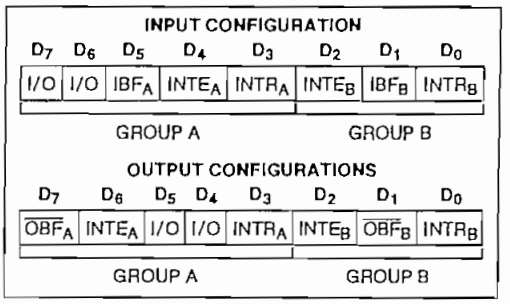


Figure 17a. MODE 1 Status Word Format

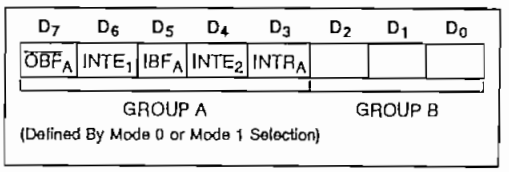


Figure 17b. MODE 2 Status Word Format

Interrupt Enable Flag	Position	Alternate Port C Pin Signal (Mode)
INTE B	PC2	$\overline{ACK}_B$ (Output Mode 1) or $\overline{STB}_B$ (Input Mode 1)
INTE A2	PC4	$\overline{STB}_A$ (Input Mode 1 or Mode 2)
INTE A1	PC6	$\overline{ACK}_A$ (Output Mode 1 or Mode 2

Figure 18. Interrupt Enable Flags in Modes 1 and 2

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias....0°C to + 70°C
Storage Temperature .....− 65°C to + 150°C
Supply Voltage .....− 0.5 to + 8.0V
Operating Voltage .....+ 4V to + 7V
Voltage on any Input.....GND−2V to + 6.5V
Voltage on any Output ..GND−0.5V to V_{CC} + 0.5V
Power Dissipation .....1 Watt

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ±10%, GND = 0V (T_A = −40°C to +85°C for Extended Temperture)

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	3.0 V_{CC} − 0.4		V V	I_{OH} = −2.5 mA I_{OH} = −100 µA
I_{IL}	Input Leakage Current		±1	µA	V_{IN} = V_{CC} to 0V (Note 1)
I_{OFL}	Output Float Leakage Current		±10	µA	V_{IN} = V_{CC} to 0V (Note 2)
I_{DAR}	Darlington Drive Current	±2.5	(Note 4)	mA	Ports A, B, C R_{ext} = 500Ω V_{ext} = 1.7V
I_{PHL}	Port Hold Low Leakage Current	+50	+300	µA	V_{OUT} = 1.0V Port A only
I_{PHH}	Port Hold High Leakage Current	−50	−300	µA	V_{OUT} = 3.0V Ports A, B, C
I_{PHLO}	Port Hold Low Overdrive Current	−350		µA	V_{OUT} = 0.8V
I_{PHHO}	Port Hold High Overdrive Current	+350		µA	V_{OUT} = 3.0V
I_{CC}	V_{CC} Supply Current		10	mA	(Note 3)
I_{CCSB}	V_{CC} Supply Current-Standby		10	µA	V_{CC} = 5.5V V_{IN} = V_{CC} or GND Port Conditions If I/P = Open/High O/P = Open Only With Data Bus = High/Low $\overline{CS}$ = High Reset = Low Pure Inputs = Low/High

NOTES:
1. Pins A_1, A_0, $\overline{CS}$, $\overline{WR}$, $\overline{RD}$, Reset
2. Data Bus; Ports B, C.
3. Outputs open.
4. Limit output current to 4.0 mA.

CAPACITANCE

T_A = 25°C, V_{CC} = GND = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		10	pF	Unmeasured pins returned to GND f_c = 1 MHz(5)
$C_{I/O}$	I/O Capacitance		20	pF	

NOTE:
5. Sampled not 100% tested.

A.C. CHARACTERISTICS

T_A = 0° to 70°C, V_{CC} = +5V ±10%, GND = 0V

T_A = −40°C to +85°C for Extended Temperature

BUS PARAMETERS

READ CYCLE

Symbol	Parameter	82C55A-2		Units	Test Conditions
		Min	Max		
t_{AR}	Address Stable Before $\overline{RD}\downarrow$	0		ns	
t_{RA}	Address Hold Time After $\overline{RD}\uparrow$	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	150		ns	
t_{RD}	Data Delay from $\overline{RD}\downarrow$		120	ns	
t_{DF}	$\overline{RD}\uparrow$ to Data Floating	10	75	ns	
t_{RV}	Recovery Time between $\overline{RD}/\overline{WR}$	200		ns	

WRITE CYCLE

Symbol	Parameter	82C55A-2		Units	Test Conditions
		Min	Max		
t_{AW}	Address Stable Before $\overline{WR}\downarrow$	0		ns	
t_{WA}	Address Hold Time After $\overline{WR}\uparrow$	20		ns	Ports A & B
		20		ns	Port C
t_{WW}	$\overline{WR}$ Pulse Width	100		ns	
t_{DW}	Data Setup Time Before $\overline{WR}\uparrow$	100		ns	
t_{WD}	Data Hold Time After $\overline{WR}\uparrow$	30		ns	Ports A & B
		30		ns	Port C

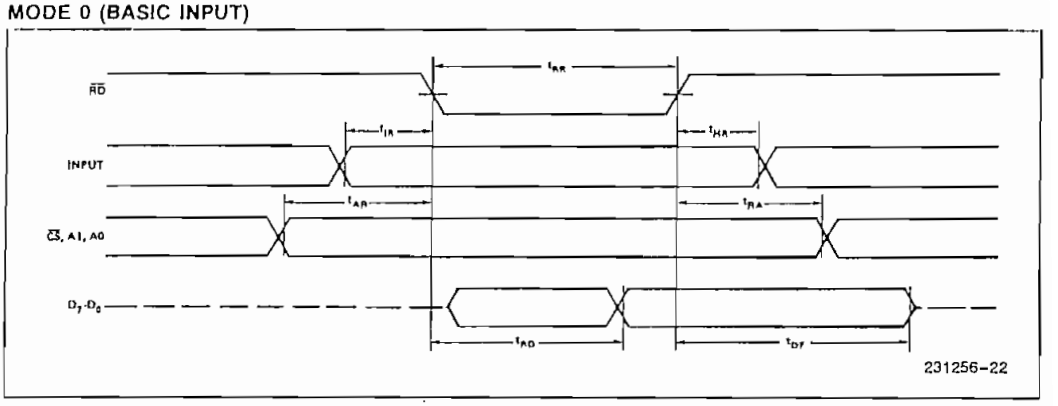
OTHER TIMINGS

Symbol	Parameter	82C55A-2		Units Conditions	Test
		Min	Max		
t_{WB}	$\overline{WR}$ = 1 to Output		350	ns	
t_{IR}	Peripheral Data Before $\overline{RD}$	0		ns	
t_{HR}	Peripheral Data After $\overline{RD}$	0		ns	
t_{AK}	$\overline{ACK}$ Pulse Width	200		ns	
t_{ST}	$\overline{STB}$ Pulse Width	100		ns	
t_{PS}	Per. Data Before $\overline{STB}$ High	20		ns	
t_{PH}	Per. Data After $\overline{STB}$ High	50		ns	
t_{AD}	$\overline{ACK}$ = 0 to Output		175	ns	
t_{KD}	$\overline{ACK}$ = 1 to Output Float	20	250	ns	
t_{WOB}	$\overline{WR}$ = 1 to $\overline{OBF}$ = 0		150	ns	
t_{AOB}	$\overline{ACK}$ = 0 to $\overline{OBF}$ = 1		150	ns	
t_{SIB}	$\overline{STB}$ = 0 to IBF = 1		150	ns	
t_{RIB}	$\overline{RD}$ = 1 to IBF = 0		150	ns	
t_{RIT}	$\overline{RD}$ = 0 to INTR = 0		200	ns	
t_{SIT}	$\overline{STB}$ = 1 to INTR = 1		150	ns	
t_{AIT}	$\overline{ACK}$ = 1 to INTR = 1		150	ns	
t_{WIT}	$\overline{WR}$ = 0 to INTR = 0		200	ns	see note 1
t_{RES}	Reset Pulse Width	500		ns	see note 2

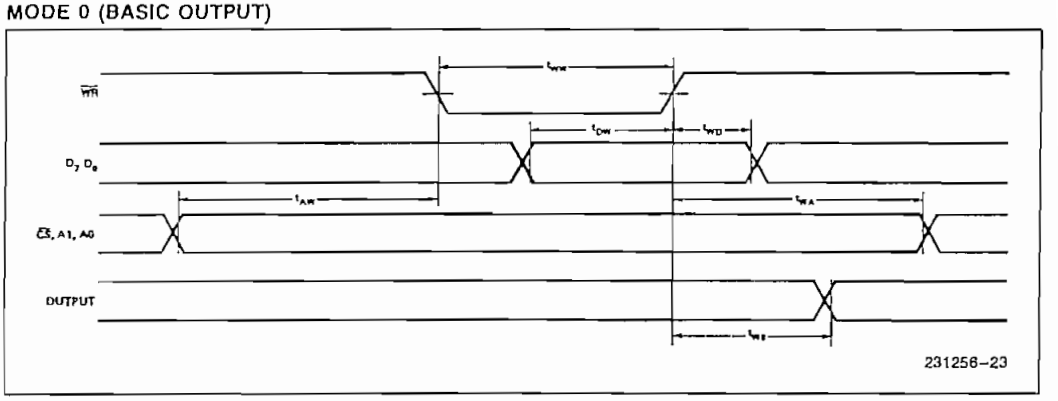
NOTE:
1. INTR ↑ may occur as early as $\overline{WR}$ ↓.
2. Pulse width of initial Reset pulse after power on must be at least 50 μSec. Subsequent Reset pulses may be 500 ns minimum. The output Ports A, B, or C may glitch low during the reset pulse but all port pins will be held at a logic "one" level after the reset pulse.

WAVEFORMS

MODE 0 (BASIC INPUT)

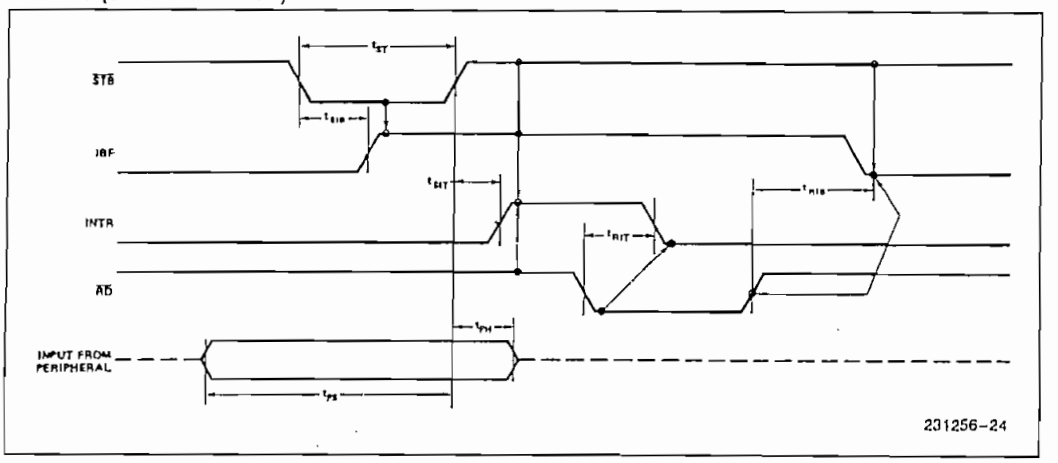


MODE 0 (BASIC OUTPUT)

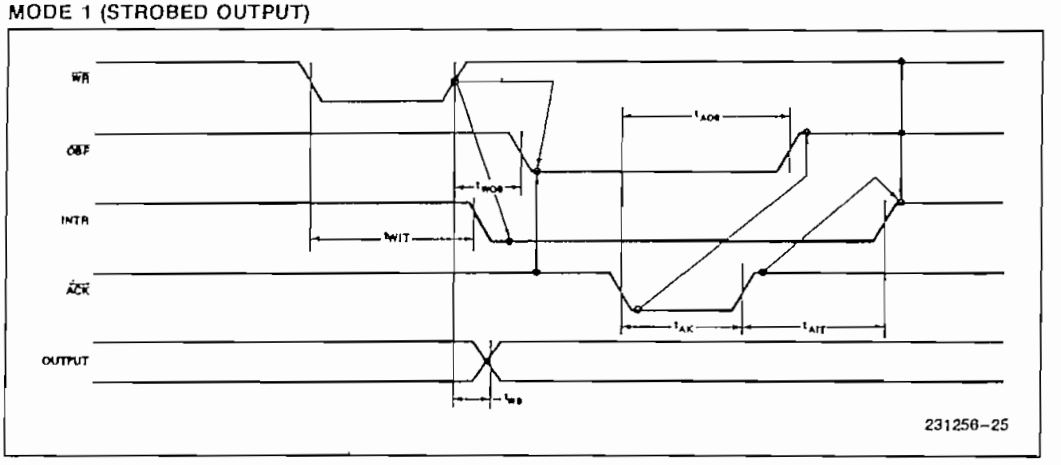


WAVEFORMS (Continued)

MODE 1 (STROBED INPUT)

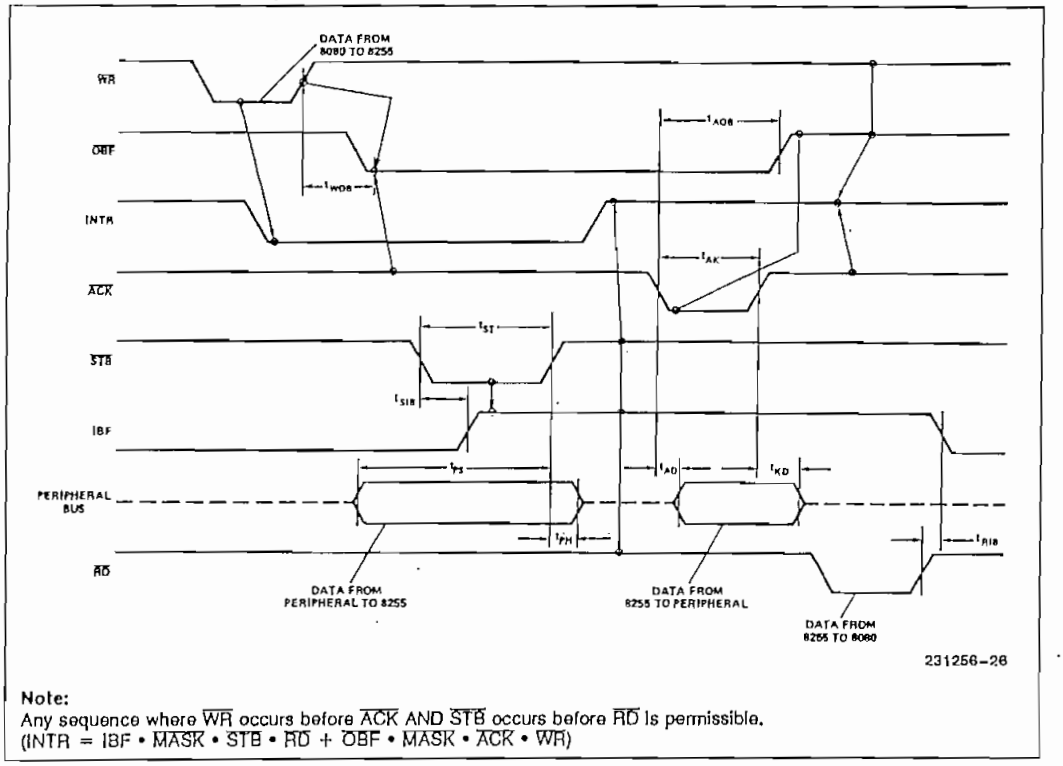


MODE 1 (STROBED OUTPUT)



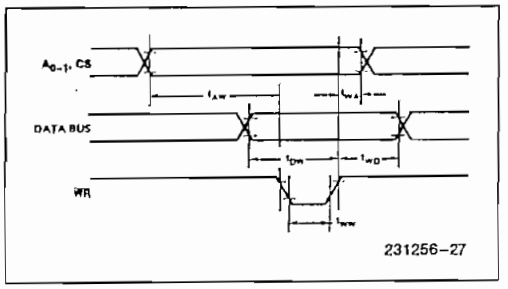
WAVEFORMS (Continued)

MODE 2 (BIDIRECTIONAL)

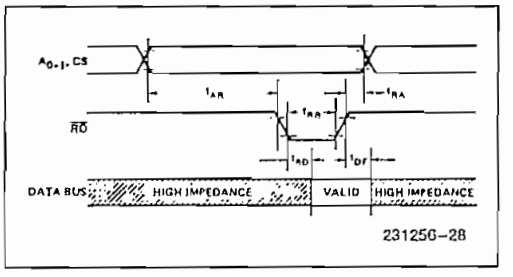


Note:
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ AND $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • MASK • STB • RD + OBF • MASK • ACK • WR)

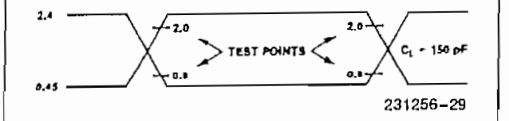
WRITE TIMING



READ TIMING

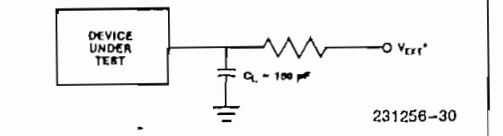


A.C. TESTING INPUT, OUTPUT WAVEFORM



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 0.8 For A Logic 0.

A.C. TESTING LOAD CIRCUIT



*V_{EXT} Is Set At Various Voltages During Testing To Guarantee The Specification. C_L Includes Jig Capacitance.

MM74C920, MM54C921/MM74C921
tic Silicon Gate CMOS RAMs

iption

C920 256 x 4 random access read/
ıfactured using silicon gate CMOS
put is the same polarity as data
s store address inputs, CES and
l is specifically designed to operate
TTL power supplies. All inputs and
patible.

4C921 is identical to the MM54C920/
ata inputs are internally connected
ıumber of package leads is thereby

coding as well as 2-chip select func-
and TRI-STATE® outputs allow easy
ıimum of external components. Ver-
satility plus high speed and low power make these RAMs ideal elements for use in microprocessor, minicomputer, as well as main frame memory applications.

Features

- 256 x 4-bit organization
- Access time
  - 250 ns max. MM74C920, MM74C921
  - 275 ns max. MM54C920, MM54C921
  - 300 ns max. MM74C920–3, MM74C921–3
- TRI-STATE outputs
- Low power
- On-chip registers
- Single 5 V supply
- Data retained with V_{CC} as low as 2V

See page 4–15
for detailed
specifications

MM54C922/MM74C922 16-Key Encoder
MM54C923/MM74C923 20-Key Encoder

General Description

These CMOS key encoders provide all the necessary logic to fully encode an array of SPST switches. The keyboard scan can be implemented by either an external clock or external capacitor. These encoders also have on-chip pull-up devices which permit switches with up to 50 kΩ on resistance to be used. No diodes in the switch array are needed to eliminate ghost switches. The internal debounce circuit needs only a single external capacitor and can be defeated by omitting the capacitor. A Data Available output goes to a high level when a valid keyboard entry has been made. The Data Available output returns to a low level when the entered key is released, even if another key is depressed. The Data Available will return high to indicate acceptance of the new key after a normal debounce period; this two key roll over is provided between any two switches.

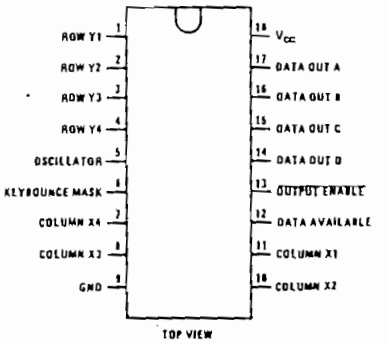
An internal register remembers the last key pressed even after the key is released. The TRI-STATE® outputs provide for easy expansion and bus operation and are LPTTL compatible.

Features

- 50 kΩ maximum switch on resistance
- On or off chip clock
- On chip row pull-up devices
- 2 key roll-over
- Keybounce elimination with single capacitor
- Last key register at outputs
- TRI-STATE outputs LPTTL compatible
- Wide supply range 3V to 15V
- Low power consumption

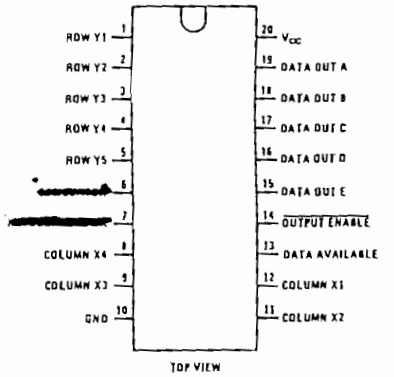
Connection Diagrams

Dual-In-Line Package

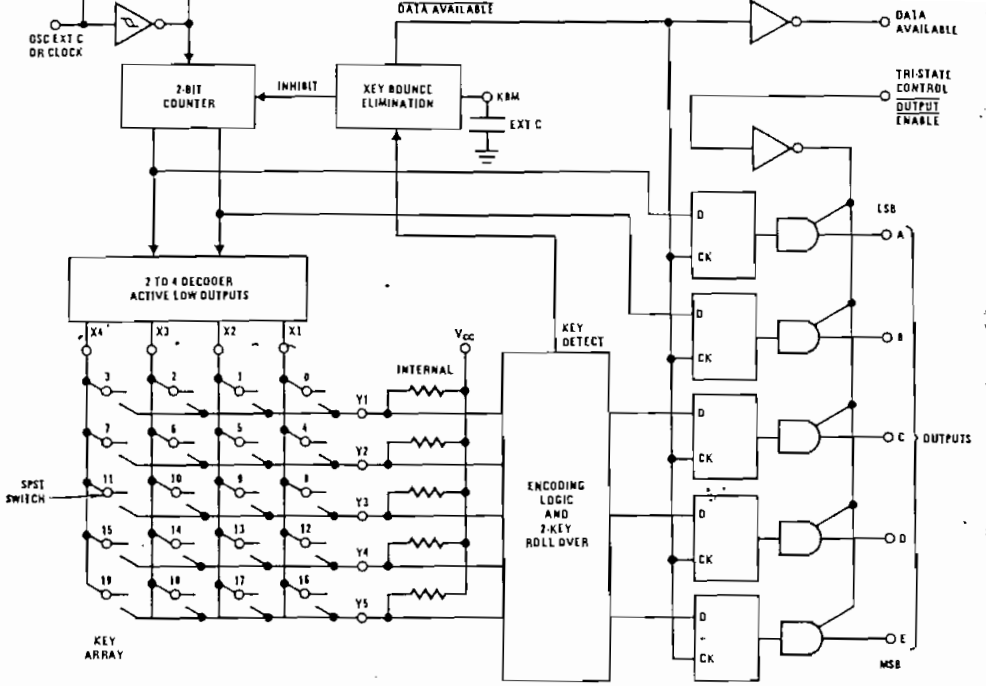


Order Number MM54C922N
or MM74C922N
See Package 20

Dual-In-Line Package



Order Number MM54C923N
or MM74C923N
See Package 20A



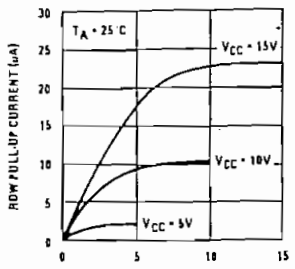
Truth Table

SWITCH POSITION	0 Y1,X1	1 Y1,X2	2 Y1,X3	3 Y1,X4	4 Y2,X1	5 Y2,X2	6 Y2,X3	7 Y2,X4	8 Y3,X1	9 Y3,X2	10 Y3,X3	11 Y3,X4	12 Y4,X1	13 Y4,X2	14 Y4,X3	15 Y4,X4	16 Y5*,X1	17 Y5*,X2	18 Y5*,X3	19 Y5*,X4
DATA OUT A	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
DATA OUT B	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
DATA OUT C	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
DATA OUT D	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
DATA OUT E*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

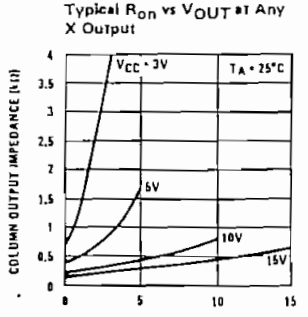
*Omit for MM54C922/MM74C922

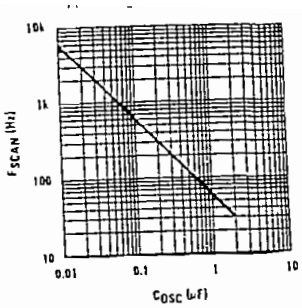
Typical Performance Characteristics

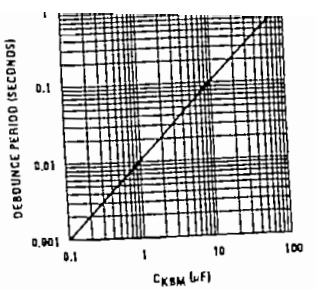
Typical I_{rp} vs V_{IN} at Any Y Input



Typical R_{on} vs V_{OUT} at Any X Output

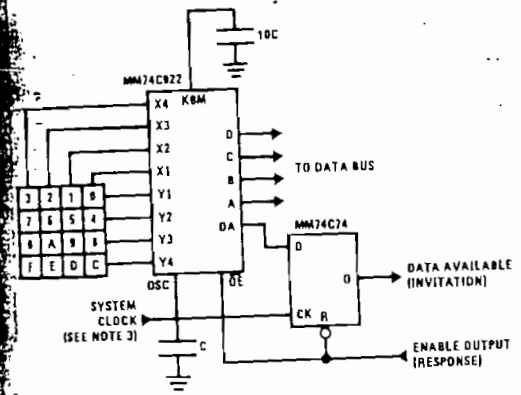




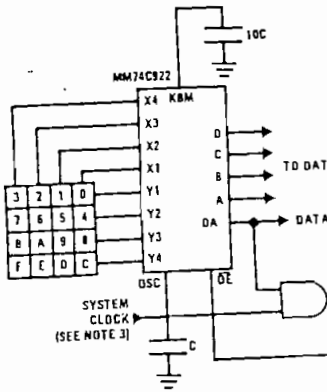


Typical Applications

Synchronous Handshake (MM74C922)

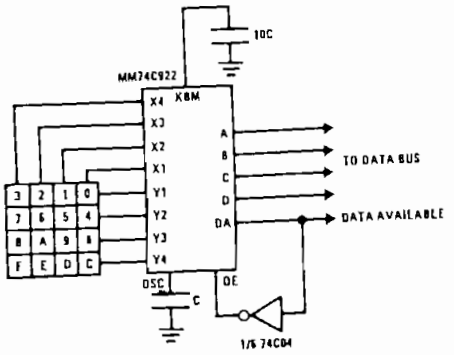


Synchronous Data Entry Onto Bus (MM74



Outputs are enabled when valid entry is ma
go into TRI-STATE when key is released.

Asynchronous Data Entry Onto Bus (MM74C922)



Outputs are in TRI-STATE until key is pressed, then data is placed on bus.
When key is released, outputs return to TRI-STATE.

Keyboard Suppliers

Mini Key Series KL
Digitran Company
Pasadena, California

Computronics Engineering
7235 Hollywood Blvd
Hollywood, California 90046

...omitting the capacitor at osc. and driving osc. directly if th

MM54C922, MM54C923	−55°C to +125°C
MM74C922, MM74C923	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C

V_{CC}	
Lead Temperature (Soldering, 10 seconds)	300°C

DC Electrical Characteristics Min/Max limits apply across temperature range unless otherwise specified

PARAMETER		CONDITIONS	MIN	TYP	MAX	UNITS
CMOS TO CMOS						
V_{T+}	Positive-Going Threshold Voltage at Osc and KBM Inputs	V_{CC} = 5V, $I_{IN} \geq 0.7$ mA	3	3.6	4.3	V
		V_{CC} = 10V, $I_{IN} \geq 1.4$ mA	6	6.8	8.6	V
		V_{CC} = 15V, $I_{IN} \geq 2.1$ mA	9	10	12.9	V
V_{T-}	Negative-Going Threshold Voltage at Osc and KBM Inputs	V_{CC} = 5V, $I_{IN} \geq 0.7$ mA	0.7	1.4	2	V
		V_{CC} = 10V, $I_{IN} \geq 1.4$ mA	1.4	3.2	4	V
		V_{CC} = 15V, $I_{IN} \geq 2.1$ mA	2.1	5	6	V
$V_{IN(1)}$	Logical "1" Input Voltage, Except Osc and KBM Inputs	V_{CC} = 5V,	3.5	4.5		V
		V_{CC} = 10V,	8	9		V
		V_{CC} = 15V,	12.5	13.5		V
$V_{IN(0)}$	Logical "0" Input Voltage, Except Osc and KBM Inputs	V_{CC} = 5V,		0.5	1.5	V
		V_{CC} = 10V,		1	2	V
		V_{CC} = 15V,		1.5	2.5	V
I_{rp}	Row Pull-Up Current at Y1, Y2, Y3, Y4 and Y5 Inputs	V_{CC} = 5V, V_{IN} = 0.1 V_{CC}		−2	−5	μA
		V_{CC} = 10V		−10	−20	μA
		V_{CC} = 15V		−22	−45	μA
$V_{OUT(1)}$	Logical "1" Output Voltage	V_{CC} = 5V, I_O = −10μA	4.5			V
		V_{CC} = 10V, I_O = −10μA	9			V
		V_{CC} = 15V, I_O = −10μA	13.5			V
$V_{OUT(0)}$	Logical "0" Output Voltage	V_{CC} = 5V, I_O = 10μA			0.5	V
		V_{CC} = 10V, I_O = 10μA			1	V
		V_{CC} = 15V, I_O = 10μA			1.5	V
R_{on}	Column "ON" Resistance at X1, X2, X3 and X4 Outputs	V_{CC} = 5V, V_O = 0.5V		500	1400	Ω
		V_{CC} = 10V, V_O = 1V		300	700	Ω
		V_{CC} = 15V, V_O = 1.5V		200	500	Ω
I_{CC}	Supply Current	V_{CC} = 5V, Osc at 0V		0.55	1.1	mA
		V_{CC} = 10V		1.1	1.9	mA
		V_{CC} = 15V		1.7	2.6	mA
$I_{IN(1)}$	Logical "1" Input Current at $\overline{\text{Output Enable}}$	V_{CC} = 15V, V_{IN} = 15V		0.005	1.0	μA
$I_{IN(0)}$	Logical "0" Input Current at $\overline{\text{Output Enable}}$	V_{CC} = 15V, V_{IN} = 0V	−1.0	−0.005		μA
CMOS/LPTTL INTERFACE						
$V_{IN(1)}$	Logical "1" Input Voltage, Except Osc and KBM Inputs	54C, V_{CC} = 4.5V	V_{CC}−1.5			V
		74C, V_{CC} = 4.75V	V_{CC}−1.5			V
$V_{IN(0)}$	Logical "0" Input Voltage, Except Osc and KBM Inputs	54C, V_{CC} = 4.5V			0.8	V
		74C, V_{CC} = 4.75V			0.8	V
$V_{OUT(1)}$	Logical "1" Output Voltage	54C, V_{CC} = 4.5V, I_O = −360μA	2.4			V
		74C, V_{CC} = 4.75V, I_O = −360μA	2.4			V
$V_{OUT(0)}$	Logical "0" Output Voltage	54C, V_{CC} = 4.5V,			0.4	V
OUTPUT DRIVE (See Note 1)						
I_{SOURCE}	Output Source Current (P-Channel)	V_{CC} = 5V, V_{OUT} = 0V, T_A = 25°C	−1.75	−3.3		
I_{SOURCE}	Output Source Current (P-Channel)	V_{CC} = 10V, V_{OUT} = 0V, T_A = 25°C	−8	−15		
I_{SINK}	Output Sink Current (N-Channel)	V_{CC} = 5V, V_{OUT} = V_{CC}, T_A = 25°C	1.75	3.6		
I_{SINK}	Output Sink Current (N-Channel)	V_{CC} = 10V, V_{OUT} = V_{CC}, T_A = 25°C	8	16		

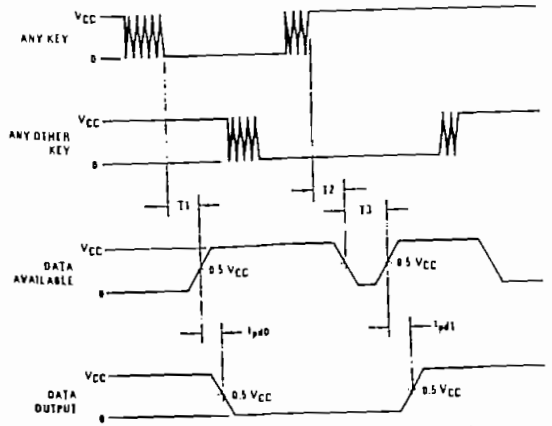
AC Electrical Characteristics T_A = 25°C, C_L = 50 pF, unless otherwise noted

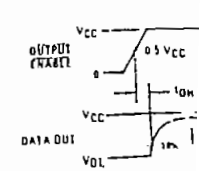
PARAMETER		CONDITIONS	MIN	TYP
t_{pd0}, t_{pd1}	Propagation Delay Time to Logical "0" or Logical "1" from D.A.	C_L = 50 pF, *(Figure 1)*		
		V_{CC} = 5V		60
		V_{CC} = 10V		35
		V_{CC} = 15V		25
t_{0H}, t_{1H}	Propagation Delay Time from Logical "0" or Logical "1" into High Impedance State	R_L = 10k, C_L = 10pF *(Figure 2)*		
		V_{CC} = 5V R_L = 10k		80
		V_{CC} = 10V C_L = 10 pF		65
		V_{CC} = 15V		50
t_{H0}, t_{H1}	Propagation Delay Time from High Impedance State to a Logical "0" or Logical "1"	R_L = 10k, C_L = 50 pF, *(Figure 2)*		
		V_{CC} = 5V R_L = 10k		100
		V_{CC} = 10V C_L = 50 pF		55
		V_{CC} = 15V		40
C_{IN}	Input Capacitance	Any Input, (Note 2)		5
C_{OUT}	TRI-STATE Output Capacitance	Any Output, (Note 2)		10

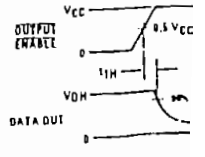
Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. Temperature Range" they are not meant to imply that the devices should be operated at these limits. The table of "Ele provides conditions for actual device operation.

Note 2: Capacitance is guaranteed by periodic testing.

Switching Time Waveforms







$T1 = T2 = RC$, $T3 = 0.7\,RC$ where R = 10k and C is external

−55°C to +125°C
−40°C to +85°C
−65°C to +150°C

V_{CC}
Lead Temperature (Soldering, 10 seconds)

haracteristics Min/Max limits apply across temperature range unless otherwise spec

TER	CONDITIONS	MIN	TYP	MAX	UNI
Threshold Voltage at	V_{CC} = 5V, $I_{IN} \geq 0.7$ mA	3	3.6	4.3	
nputs	V_{CC} = 10V, $I_{IN} \geq 1.4$ mA	6	6.8	8.6	
	V_{CC} = 15V, $I_{IN} \geq 2.1$ mA	9	10	12.9	
Threshold Voltage at	V_{CC} = 5V, $I_{IN} \geq 0.7$ mA	0.7	1.4	2	
Inputs	V_{CC} = 10V, $I_{IN} \geq 1.4$ mA	1.4	3.2	4	
	V_{CC} = 15V, $I_{IN} \geq 2.1$ mA	2.1	5	6	
out Voltage, Except	V_{CC} = 5V,	3.5	4.5		
nputs	V_{CC} = 10V,	8	9		
	V_{CC} = 15V,	12.5	13.5		
out Voltage, Except	V_{CC} = 5V,		0.5	1.5	
Inputs	V_{CC} = 10V,		1	2	
	V_{CC} = 15V,		1.5	2.5	
Current at Y1, Y2, Y3,	V_{CC} = 5V, V_{IN} = 0.1 V_{CC}		−2	−5	
outs	V_{CC} = 10V		−10	−20	
	V_{CC} = 15V		−22	−45	
utput Voltage	V_{CC} = 5V, I_O = −10μA	4.5			
	V_{CC} = 10V, I_O = −10μA	9			
	V_{CC} = 15V, I_O = −10μA	13.5			
utput Voltage	V_{CC} = 5V, I_O = 10μA			0.5	
	V_{CC} = 10V, I_O = 10μA			1	
	V_{CC} = 15V, I_O = 10μA			1.5	
" Resistance at	V_{CC} = 5V, V_O = 0.5V		500	1400	
nd X4 Outputs	V_{CC} = 10V, V_O = 1V		300	700	
	V_{CC} = 15V, V_O = 1.5V		200	500	
nt	V_{CC} = 5V, Osc at 0V		0.55	1.1	
	V_{CC} = 10V		1.1	1.9	
	V_{CC} = 15V		1.7	2.6	
nput Current at e	V_{CC} = 15V, V_{IN} = 15V		0.005	1.0	
nput Current at e	V_{CC} = 15V, V_{IN} = 0V	−1.0	−0.005		
ACE					
nput Voltage, Except	54C, V_{CC} = 4.5V	V_{CC}−1.5			
Inputs.	74C, V_{CC} = 4.75V	V_{CC}−1.5			
nput Voltage, Except	54C, V_{CC} = 4.5V			0.8	
Inputs	74C, V_{CC} = 4.75V			0.8	
utput Voltage	54C, V_{CC} = 4.5V, I_O = −360μA	2.4			
	74C, V_{CC} = 4.75V, I_O = −360μA	2.4			
	54C, V_{CC} = 4.5V,			0.4	

UTPUT DRIVE (See 54C/74C Family Characteristics)					
OURCE Output Source Current (P-Channel)	V_{CC} = 5V, V_{OUT} = 0V, T_A = 25°C	−1.75	−3.3		mA
OURCE Output Source Current (P-Channel)	V_{CC} = 10V, V_{OUT} = 0V, T_A = 25°C	−8	−15		mA
INK Output Sink Current (N-Channel)	V_{CC} = 5V, V_{OUT} = V_{CC}, T_A = 25°C	1.75	3.6		mA
INK Output Sink Current (N-Channel)	V_{CC} = 10V, V_{OUT} = V_{CC}, T_A = 25°C	8	16		mA

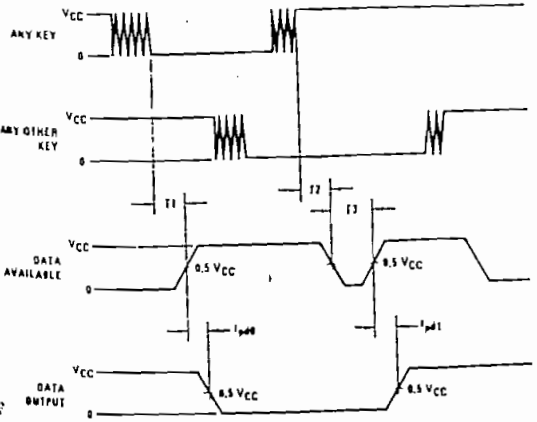
AC Electrical Characteristics T_A = 25°C, C_L = 50 pF, unless otherwise noted

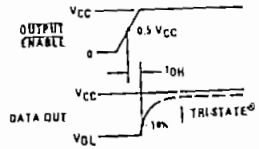
PARAMETER		CONDITIONS	MIN	TYP	MAX	UNITS
t_{pd0},t_{pd1}	Propagation Delay Time to Logical "0" or Logical "1" from D.A.	C_L = 50 pF, *(Figure 1)*				
		V_{CC} = 5V		60	150	ns
		V_{CC} = 10V		35	80	ns
		V_{CC} = 15V		25	60	ns
t_{0H},t_{1H}	Propagation Delay Time from Logical "0" or Logical "1" into High Impedance State	R_L = 10k, C_L = 10pF *(Figure 2)*				
		V_{CC} = 5V R_L = 10k		80	200	ns
		V_{CC} = 10V C_L = 10 pF		65	150	ns
		V_{CC} = 15V		50	110	ns
t_{H0},t_{H1}	Propagation Delay Time from High Impedance State to a Logical "0" or Logical "1"	R_L = 10k, C_L = 50 pF, *(Figure 2)*				
		V_{CC} = 5V R_L = 10k		100	250	ns
		V_{CC} = 10V C_L = 50 pF		55	125	ns
		V_{CC} = 15V		40	90	ns
C_{IN}	Input Capacitance	Any Input, (Note 2)		5	7.5	pF
C_{OUT}	TRI-STATE Output Capacitance	Any Output, (Note 2)		10		pF

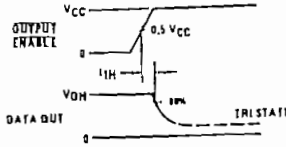
Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. Except for "Operating Temperature Range" they are not meant to imply that the devices should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Capacitance is guaranteed by periodic testing.

Switching Time Waveforms

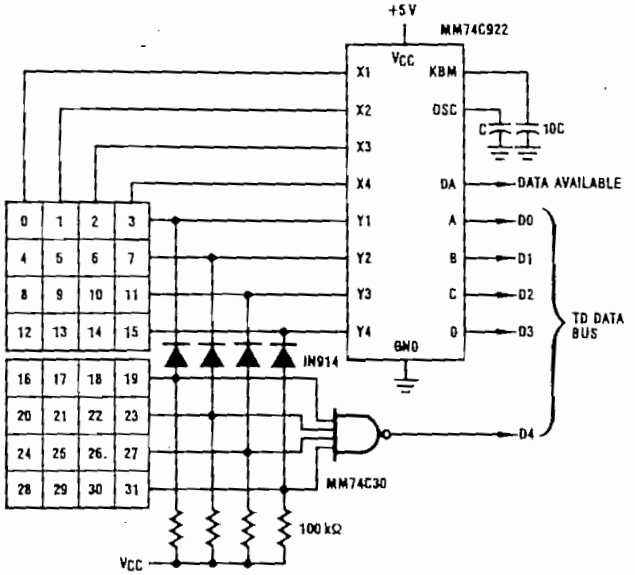






T1 = T2 = 0.6 RC where R = 10k and C is external

Expansion to 32 Key Encoder (MM74C922)



Theory of Operation

The MM74C922/MM74C923 Keyboard Encoders implement all the logic necessary to interface a 16 or 20 SPST key switch matrix to a digital system. The encoder will convert a key switch closer to a 4(MM74C922) or 5(MM74C923) bit nibble. The designer can control both the keyboard scan rate and the key debounce period by altering the oscillator capacitor, C_{OSE}, and the key bounce mask capacitor, C_{MSK}. Thus, the MM74C922/MM47C923's performance can be optimized for many keyboards.

The keyboard encoders connect to a switch matrix that is 4 rows by 4 columns (MM74C922) or 5 rows by 4 columns (MM74C923). When no keys are depressed, the row inputs are pulled high by internal pull-ups and the column outputs sequentially output a logic "0". These outputs are open drain and are therefore low for 25% of the time and otherwise off. The column scan rate is controlled by the oscillator input, which consists of a Schmitt trigger oscillator, a 2-bit counter, and a 2-4-bit decoder.

When a key is depressed, key 0, for example, nothing will happen when the X1 input is off, since Y1 will remain high. When the X1 column is scanned, X1 goes low and [illegible] counter and keeps X1 timing and locks out the other Y inputs. The key code to be outputted is a combination of the frozen counter value and the decoded Y inputs. Once the key bounce circuit times out, the data is latched, and the Data Available (DAV) output goes high.

If, during the key closure the switch bounces, Y1 input will go high again, restarting the scan and resetting the key bounce circuitry. The key may bounce several times but as soon as the switch stays low for a debounce period, the closure is assumed valid and the data is latched.

A key may also bounce when it is released. To ensure that the encoder does not recognize this bounce as another key closure, the debounce circuit must time out before another closure is recognized.

The two key roll over feature can be illustrated by assuming a key is depressed, and then a second key is depressed. Since all scanning has stopped, and all other Y inputs are disabled, the second key is not recognized until the first key is lifted and the key bounce circuitry has reset.

The output latches feed TRI-STATE®, which are enabled [illegible] ($\overline{OE}$) input is taken low.

MM74C925, MM74C926, MM[illegible]
Counters with Multiplexed 7-Segment Output Driver

General Description

These CMOS counters consist of a 4-digit counter, an internal output latch, NPN output sourcing drivers for a 7-segment display, and an internal multiplexing circuitry with four multiplexing outputs. The multiplexing circuit has its own free-running oscillator, and requires no external clock. The counters advance on negative edge of clock. A high signal on the Reset input will reset the counter to zero, and reset the carry-out low. A low signal on the Latch Enable input will latch the number in the counters into the internal output latches. A high signal on Display Select input will select the number in the counter to be displayed; a low level signal on the Display Select will select the number in the output latch to be displayed.

The MM74C925 is a 4-decade counter and has Latch Enable, Clock and Reset inputs.

The MM74C926 is like the MM74C925 except that it has a display select and a carry-out used for cascading counters. The carry-out signal goes high at 6000, goes back low at 0000.

The MM74C927 is like the MM74C926 except the second most significant digit divides by 6 rather than 10. Thus, if the clock input frequency is 10 Hz, the display would read tenths of seconds and minutes (i.e., 9:59.9).

The MM74C928 is like the MM74C926 except the most significant digit divides by 2 rather than 10 and the carry-out is an overflow indicator [illegible] 2000, and it goes back low only whe[n] reset. Thus, this is a 3 1/2-digit counter.

Features

- Wide supply voltage range
- Guaranteed noise margin
- High noise immunity
- High segment sourcing current @ V_{CC} –
- Internal multiplexing circuitry

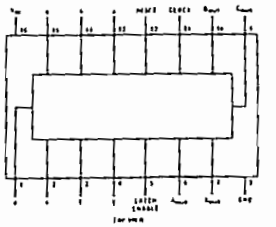
Design Considerations

Segment resistors are desirable to [illegible] dissipation and chip heating. The DS[illegible] good digit driver when it is desired [illegible] displays. When using this driver with [illegible] room temperature, the display can b[illegible] segment resistors to full illumination. [illegible] caution in this mode however, to prev[illegible] the device by using too high a supp[illegible] operating at high ambient temperatures.

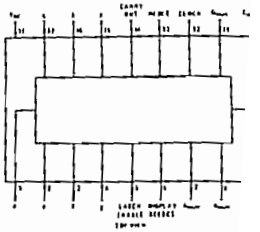
The input protection circuitry consists [illegible] and a diode to ground. Thus input [illegible] V_{CC} will not be clamped. This input sig[illegible] allowed to exceed 15V.

Connection Diagram

Dual-In-Line Package
MM74C925



Dual-In-Line Package
MM74C926, MM74C927 and MM74[illegible]



Functional Description

Reset — Asynchronous, active high

Display Select — High, displays output of counter
Low, displays output of latch

Latch Enable — High, flow through condition
Low, latch condition

Segment Output — Current sourc[illegible]
$V_{OUT} = V_C$
Also, sink ca[illegible]
loads

Digit Output — Current sour[illegible]
$V_{OUT} = 1.7$[illegible]
bility = 2 LTT[illegible]

Carry-out — 2 LTTL l[illegible]

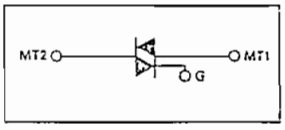
Triacs
Silicon Bidirectional Triode Thyristors

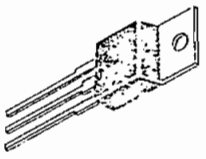
MAC15
MAC15A

TRIACs
15 AMPERES RMS
200 thru 800 VOLTS

... designed primarily for full-wave ac control applications, such as solid-state relays, motor controls, heating controls and power supplies; or wherever full-wave silicon gate controlled solid-state devices are needed. Triac type thyristors switch from a blocking to a conducting state for either polarity of applied anode voltage with positive or negative gate triggering.

- Blocking Voltage to 800 Volts
- All Diffused and Glass Passivated Junctions for Greater Parameter Uniformity and Stability
- Small, Rugged, Thermowatt Construction for Low Thermal Resistance, High Heat Dissipation and Durability
- Gate Triggering Guaranteed in Two Modes (MAC15)
  Four Modes (MAC15A)





CASE 221A-02
TO-220AB

3

MAXIMUM RATINGS

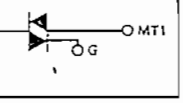
Rating	Symbol	Value	Unit
Peak Repetitive Off-State Voltage (T_J = −40 to 125°C) MAC15-4, MAC15A-4 MAC15-6, MAC15A-6 MAC15-8, MAC15A-8 MAC15-10, MAC15A-10	V_{DRM}	200 400 600 800	Volts
Peak Gate Voltage	V_{GM}	10	Volts
On-State Current RMS Full Cycle Sine Wave 50 to 60 Hz (T_C = +90°C)	$I_{T(RMS)}$	15	Amps
Circuit Fusing	I^2t	93	A^2s
Peak Surge Current (One Full Cycle, 60 Hz, T_C = +80°C) Preceded and followed by rated current)	I_{TSM}	150	Amps
Peak Gate Power (T_C = +80°C, Pulse Width = 2 μs)	P_{GM}	20	Watts
Average Gate Power (T_C = +80°C, t = 8.3 ms)	$P_{G(AV)}$	0.5	Watt
Peak Gate Current	I_{GM}	2	Amps
Operating Junction Temperature Range	T_J	−40 to +125	°C
Storage Temperature Range	T_{stg}	−40 to +150	°C

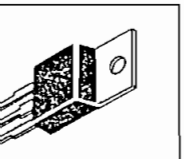
THERMAL CHARACTERISTICS

Characteristic	Symbol	Max	Unit
Thermal Resistance, Junction to Case	$R_{\theta JC}$	2	°C/W

AC15
AC15A

TRIACs
MPERES RMS
ru 800 VOLTS





SE 221A-02
TO-220AB

	Unit
	Volts
	Volts
	Amps
	A^2s
	Amps
	Watts
	Watt
	Amps
125	°C
150	°C

	Unit
	°C/W

ELECTRICAL CHARACTERISTICS (T_C = 25°C, and either polarity of MT2 to MT1 Voltage, unless otherwise noted.)

Characteristic	Symbol	Min	Typ	Max	Unit
Peak Forward or Reverse Blocking Current (Rated V_{DRM} or V_{RRM}, gate open) T_J = 25°C	I_{DRM}, I_{RRM}	—	—	10	μA
T_J = 125°C		—	—	2	mA
Peak On-State Voltage (I_{TM} = 21 A Peak; Pulse Width = 1 to 2 ms, Duty Cycle ≤ 2%)	V_{TM}	—	1.3	1.6	Volts
Peak Gate Trigger Current (V_D = 12 Vdc, R_L = 100 Ohms) (Minimum Gate Pulse Width = 2 μs)	I_{GTM}				mA
MT2(+), G(+) MAC15, MAC15A		—	—	50	
MT2(+), G(−) MAC15A		—	—	75	
MT2(−), G(−) MAC15, MAC15A		—	—	50	
MT2(−), G(+) MAC15A		—	—	75	
Peak Gate Trigger Voltage (V_D = 12 Vdc, R_L = 100 Ohms) (Minimum Gate Pulse Width = 2 μs)	V_{GTM}				Volts
MT2(+), G(+) MAC15, MAC15A		—	0.9	2	
MT2(+), G(−) MAC15A		—	0.9	2.5	
MT2(−), G(−) MAC15, MAC15A		—	1.1	2	
MT2(−), G(+) MAC15A		—	1.4	2.5	
(V_D = Rated V_{DRM}, R_L = 10 k Ohms, T_J = 110°C)					
MT2(+), G(+); MT2(−), G(−) MAC15, MAC15A		0.2	—	—	
MT2(+), G(−); MT2(−), G(+) MAC15A		0.2	—	—	
Holding Current (Either Direction) (V_D = 12 Vdc, Gate Open) (I_T = 200 mA)	I_H	—	6	40	mA
Turn-On Time (V_D = Rated V_{DRM}, I_{TM} = 17 A) (I_{GT} = 120 mA, Rise Time = 0.1 μs, Pulse Width = 2 μs)	t_{gt}	—	1.5	2	μs
Critical Rate of Rise of Commutation Voltage (V_D = Rated V_{DRM}, I_{TM} = 21 A, Commutating di/dt = 8 A/ms, Gate Unenergized, T_C = 80°C)	dv/dt(c)	—	5	—	V/μs

FIGURE 1 — RMS CURRENT DERATING

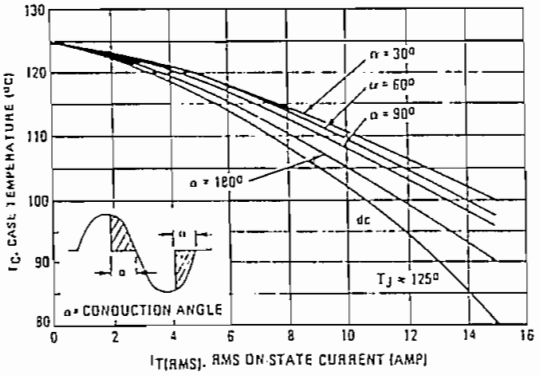


FIGURE 2 — ON-STATE POWER DISSIPATION

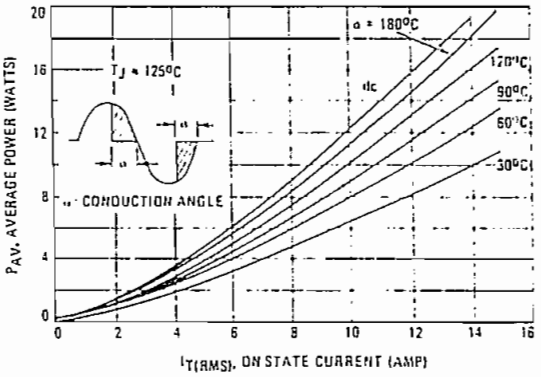


FIGURE 3 — TYPICAL GATE TRIGGER VOLTAGE

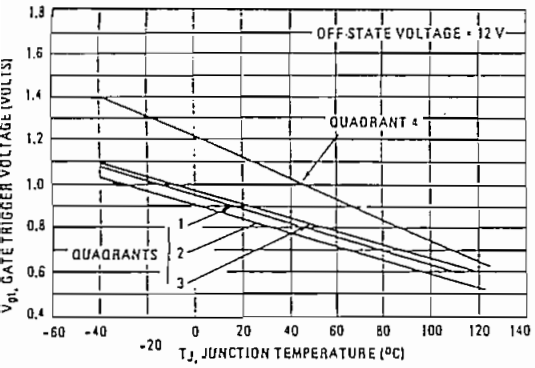
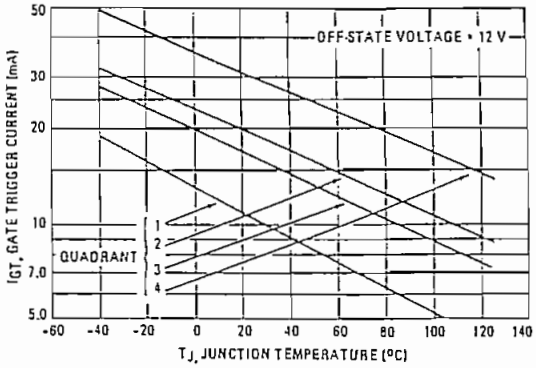


FIGURE 4 — TYPICAL GATE TRIGGER CURRENT



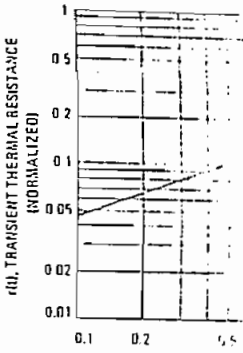


FIGURE 5 — ON-STATE CHARACTERISTICS

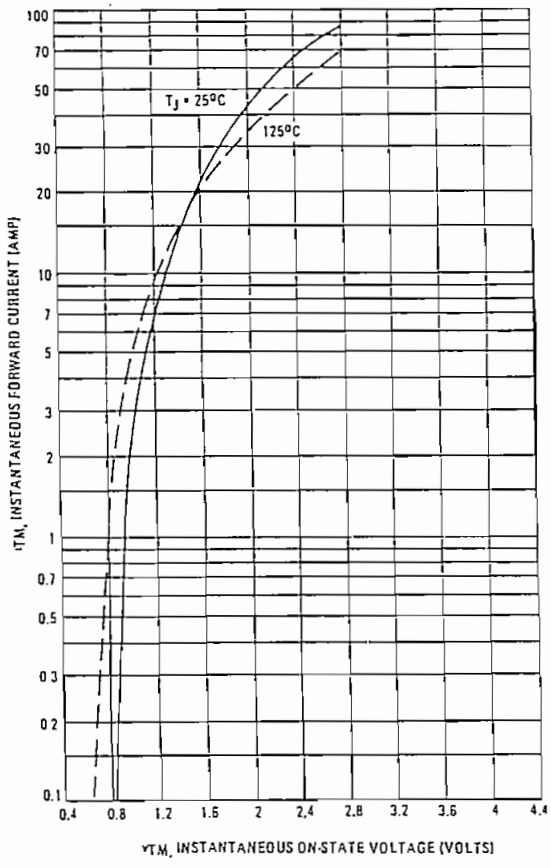


FIGURE 6 — TYPICAL HOLDING CURRENT

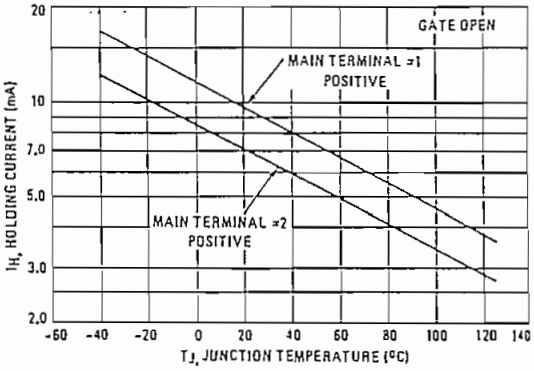
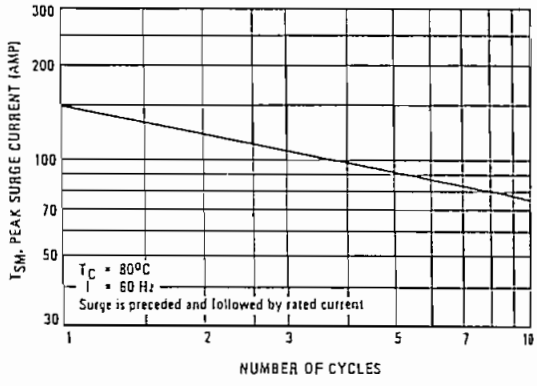


FIGURE 7 — MAXIMUM NON-REPETITIVE SURGE CURRENT



Opto Coupler

Zero Voltage Crossing Optically Isolated Triac Driver

MOC3030
MOC3031

250 VOLTS

CASE 730-01

This device consists of a gallium arsenide infrared emitting diode optically coupled to a monolithic silicon detector performing the function of a Zero Voltage crossing bilateral triac driver.

They are designed for use with a triac in the interface of logic systems to equipment powered from 115 Vac lines, such as teletypewriters, CRTs, printers, motors, solenoids and consumer appliances, etc.

- Simplifies Logic Control of 110 Vac Power
- Zero Voltage Crossing
- High Breakdown Voltage: V_{DRM} = 250 V Min
- High Isolation Voltage: V_{ISO} = 7500 V Min
- Small, Economical, 6-Pin DIP Package
- Same Pin Configuration as MOC3010/3011
- UL Recognized, File No. E54915
- dv/dt of 100 V/μs Typ

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
INFRARED EMITTING DIODE MAXIMUM RATINGS			
Reverse Voltage	V_R	3	Volts
Forward Current — Continuous	I_F	50	mA
Total Power Dissipation @ T_A = 25°C Negligible Power in Output Driver Derate above 25°C	P_D	 120 1.33	 mW mW/°C
OUTPUT DRIVER MAXIMUM RATINGS			
Off-State Output Terminal Voltage	V_{DRM}	250	Volts
On-State RMS Current (Full Cycle, 50 to 60 Hz) T_A = 25°C T_A = 85°C	$I_{T(RMS)}$	100 50	mA mA
Peak Nonrepetitive Surge Current (PW = 10 ms)	I_{TSM}	1.2	A
Total Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	300 4	mW mW/°C
TOTAL DEVICE MAXIMUM RATINGS			
Isolation Surge Voltage (1) (Peak ac Voltage, 60 Hz, 5 Second Duration)	V_{ISO}	7500	Vac
Total Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	330 4.4	mW mW/°C
Junction Temperature Range	T_J	−40 to +100	°C
Ambient Operating Temperature Range	T_A	−40 to +85	°C
Storage Temperature Range	T_{stg}	−40 to +150	°C
Soldering Temperature (10 s)	—	260	°C

(1) Isolation surge voltage, V_{ISO}, is an internal device dielectric breakdown rating.

ELECTRICAL CH...

LED CHARACTER...

Reverse Leaka... (V_R = 3 V)

Forward Volta... (I_F = 30 mA...

DETECTOR CH...

Peak Blockin... (Rated V_{DR}...

Peak On-State ... (I_{TM} = 100 ...

Critical Rate of ...

COUPLED CHA...

LED Trigger Cur... (Main Termi...

Holding Current

ZERO CROSS...

Inhibit Voltage (I_F = Rated ... will not tri...

Leakage in Inhi... (I_F = Rated ...

Notes: 1. Test vo... tag...

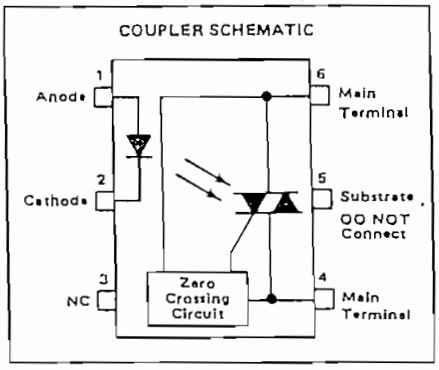
2. All device... I_{FT} (30 mA...

ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted.)

Characteristic	Symbol	Min	Typ	Max	Unit
LED CHARACTERISTICS					
Reverse Leakage Current (V_R = 3 V)	I_R	—	0.05	100	μA
Forward Voltage (I_F = 30 mA)	V_F	—	1.3	1.5	Volts
DETECTOR CHARACTERISTICS (I_F = 0 unless otherwise noted.)					
Peak Blocking Current, Either Direction (Rated V_{DRM}, Note 1)	I_{DRM}	—	10	100	nA
Peak On-State Voltage, Either Direction (I_{TM} = 100 mA Peak)	V_{TM}	—	1.8	3	Volts
Critical Rate of Rise of Off-State Voltage	dv/dt	—	100	—	V/μs
COUPLED CHARACTERISTICS					
LED Trigger Current, Current Required to Latch Output (Main Terminal Voltage = 3 V, Note 2) MOC3030 MOC3031	I_{FT}	— —	— —	30 15	mA
Holding Current, Either Direction	I_H	—	100	—	μA
ZERO CROSSING CHARACTERISTICS					
Inhibit Voltage (I_F = Rated I_{FT}, MT1–MT2 Voltage above which device will not trigger)	V_{IH}	—	15	25	Volts
Leakage in Inhibited State (I_F = Rated I_{FT}, Rated V_{DRM}, Off State)	I_R	—	100	200	μA

Notes: 1. Test voltage must be applied within dv/dt rating.

2. All devices are guaranteed to trigger at an I_F value less than or equal to max I_{FT}. Therefore, recommended operating I_F lies between max I_{FT} (30 mA for MOC3030, 15 mA for MOC3031) and absolute max I_F (50 mA).



TYPICAL ELECTRICAL CHARACTERISTICS
$T_A = 25°C$

FIGURE 1 – ON-STATE CHARACTERISTICS

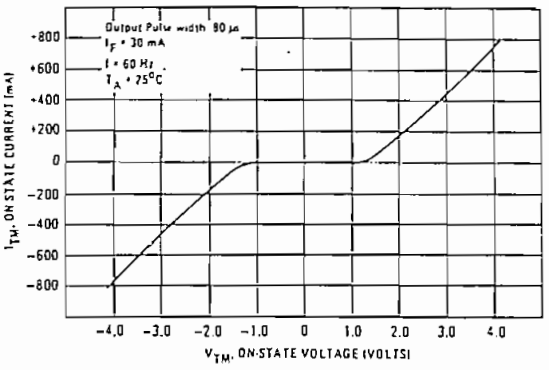


FIGURE 2 – TRIGGER CURRENT versus TEMPERATURE

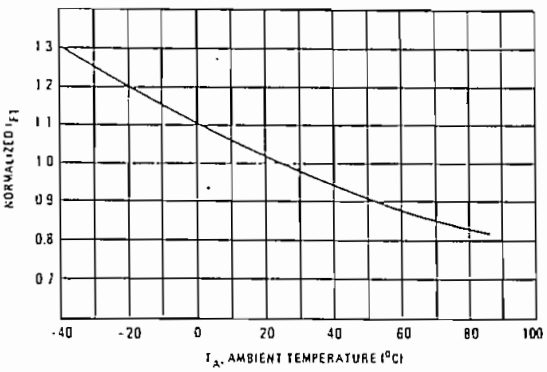
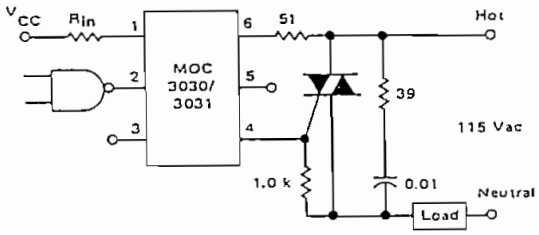
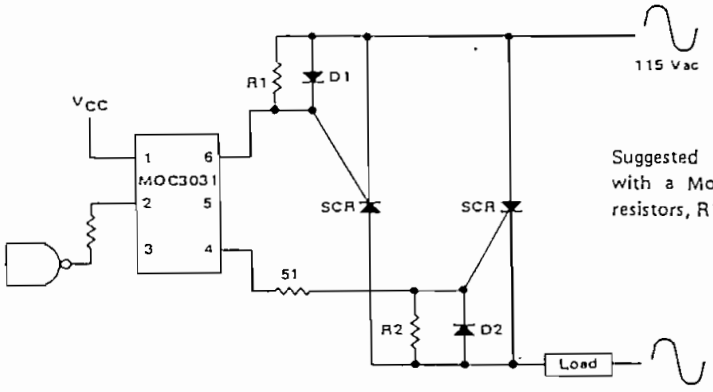


FIGURE 3 – HOT-LINE SWITCHING APPLICATION CIRCUIT



Typical circuit for use when hot line switching is required. In this circuit the "hot" side of the line is switched and the load connected to the cold or neutral side. The load may be connected to either the neutral or hot line.
R_{in} is calculated so that I_F is equal to the rated I_{FT} of the part, 15 mA for the MOC3031 or 30 mA for the MOC3030. The 39 ohm resistor and 0.01 µF capacitor are for snubbing of the triac and may or may not be necessary depending upon the particular triac and load used.

FIGURE 4 – INVERSE-PARALLEL SCR DRIVER CIRCUIT



Suggested method of firing two, back-to-back SCR's, with a Motorola triac driver. Diodes can be 1N4001; resistors, R1 and R2, are optional 1 k ohm.