UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA
ESCUELA DE INGENIERIA ELECTRICA

"DISEÑO Y CONSTRUCCION DE UN SISTEMA CONTROLADOR DE MATRICES DE PUNTOS, PARA UN TABLERO ELECTRONICO DE ANUNCIOS EN LA ESCUELA DE INGENIERIA ELECTRICA DE LA UNIVERSIDAD DE EL SALVADOR"

TRABAJO DE GRADUACION PRESENTADO POR:

WALTER FREDI ESCOBAR PORTILLO
OSCAR RENE LOPEZ FUENTES

PARA OPTAR AL TITULO DE:

INGENIERO ELECTRICISTA

JUNIO DE 1994

SAN SALVADOR, EL SALVADOR, CENTRO AMERICA

UNIVERSIDAD DE EL SALVADOR

RECTOR:

DR. FABIO CASTILLO FIGUEROA

SECRETARIO GENERAL:

LIC. MIRNA ANTONIETA PERLA DE ANAYA

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO:

ING. JOAQUIN ALBERTO VANEGAS AGUILAR

SECRETARIO:

ING. JOSE RIGOBERTO MURILLO CAMPOS

ESCUELA DE INGENIERIA ELECTRICA

DIRECTOR:

ING. SALVADOR DE JESUS GERMAN

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE INGENIERIA ELECTRICA

Trabajo de graduación previo a la opción al grado de:

INGENIERO ELECTRICISTA

Título:"DISEÑO Y CONSTRUCCION DE UN SISTEMA CONTROLADOR DE MATRICES DE PUNTOS, PARA UN TABLERO ELECTRONICO DE ANUNCIOS EN LA ESCUELA DE INGENIERIA ELECTRICA DE LA UNIVERSIDAD DE EL SALVADOR"

Presentado por:

WALTER FREDI ESCOBAR PORTILLO
OSCAR RENE LOPEZ FUENTES

Trabajo de Graduación aprobado por:

Coordinador:

ING. RICARDO ERNESTO CORTEZ

Asesor:

ING. HECTOR DOMPILIO ESCOBAR

San Salvador, junio de 1994.

ACTA DE CONSTANCIA DE NOTA Y DEFENSA FINAL

En esta fecha, 8 de JUNIO de 1994, en el local de Sala de Lectura de la Escuela de Ing. Eléctrica a las 7:00 horas, con la presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1- Ing. Salvador de J. German — Director — Firma

2- _____

3- _____

ESCUELA DE INGENIERIA ELECTRICA
FACULTAD DE INGENIERIA
Y ARQUITECTURA
Universidad de El Salvador

Y con el Honorable Jurado de evaluación integrado por las personas siguientes:

1- Ing. Ramón Portillo

2- Ing. Carlos García Bayón

3- Ing. Roberto Carlos Alvarenga

4- _____

5- _____

6- _____

Se efectuó la defensa final reglamentaria del Trabajo de Graduación: "DISEÑO Y CONSTRUCCION DE UN SISTEMA CONTROLADOR DE MATRICES DE PUNTOS, PARA UN TABLERO ELECTRONICO DE ANUNCIOS EN LA ESCUELA DE INGENIERIA ELECTRICA DE LA UNIVERSIDAD DE EL SALVADOR"

a cargo del (los) Br(es): Walter Fredi Escobar Portillo
Oscar René López Fuentes

Habiendo obtenido el presente trabajo una nota final, global de 8.6

(ocho punto Seis)

TRABAJO DEDICADO A:

DIOS NUESTRO SEÑOR Por haberme ayudado a afrontar y superar todas las situaciones difíciles que se presentaron a lo largo de todos mis estudios.

MI MADRE Inés Portillo de Escobar por todo su sacrificio ,Y por todo el apoyo incondicional que me brindó. Por su esfuerzo culmino hoy esta carrera.

A ellos GRACIAS.

WALTER FREDI ESCOBAR PORTILLO

AGRADECIMIENTOS A:

Dios por darme la sabiduria y fortaleza necesaria para culminar mis estudios.

Mis padres Jacinto Roque Fuentes y María Ermelinda López por haberme apoyado y creer siempre en mí.

Mis hermanos Maritza,Roque y Daniel que siempre estuvieron a mi lado.

A mis amigos y compañeros con los cuales comparti muchos momentos de alegría y preocupación durante todos mis estudios.

A ellos dedico éste triunfo.

OSCAR RENE LOPEZ FUENTES

PREFACIO

El avance extraordinario de las técnicas de integración ha producido en los últimos años un desarrollo espectacular de las técnicas digitales, añadiendo a los conceptos clásicos el de programabilidad.

La creación de circuitos integrados cada vez mas pequeños y potentes ha permitido llegar a componentes muy versátiles, como el microprocesador, y no es una exageración afirmar que éste ha revolucionado la industria de la electrónica y ha tenido un gran impacto en muchos aspectos de nuestras vidas.

La integración a gran escala (LSI) ha reducido tan considerablemente el tamaño y el costo de las computadoras que los diseñadores consideran utilizar rutinariamente el poder y la versatilidad del microprocesador en una amplia variedad de aplicaciones.

Los sistemas de visualización digital que se basan en microprocesadores, están teniendo un gran auge en los países desarrollados, pues, es común ver letreros luminosos especiales en los edificios, centros comerciales, hoteles, etc. Aún en nuestro país ya se observa alguna incursión de esta tecnología. La tendencia en la actualidad encamina a la digitalización de todos los sistemas que incluyen cálculos y representaciones visuales. Consecuentemente la información adquiere un perfil más elegante en cuanto a su procesamiento y presentación, ya que pueden incorporárseles efectos especiales que optimizen la comunicación.

El desarrollo de proyectos encaminados al diseño y construcción de presentadores de gran tamaño, crea muchas expectativas en cuanto a que muchos desean conocer las técnicas y procedimientos empleados en la elaboración de tan atractivo producto.En éste sentido nuestro trabajo servirá en parte para satisfacer algunas de esas inquietudes.

El objetivo de éste trabajo es diseñar y construir un presentador de caracteres,electrónico y programable,que pueda ser utilizado como medio de información dentro de la Escuela de Ingenieria Eléctrica

RESUMEN DEL TRABAJO

En la actualidad los sistemas dedicados a la transmisión y presentación de información han evolucionado rapidamennte , ésto se debe a que la electrónica ha tenido grandes avances,especialmente la electrónica digital.

Aprovechando éstos avances ,el presente trabajo tiene por objetivo utilizar las nuevas tecnicas en el diseño de sistemas,para crear un rótulo electrónico capaz de presentar información útil a la Escuela de Ingeniería Electrica.

El trabajo consiste en el diseño y construcción basado en microprocesadores de un rótulo electrónico que muestre mensajes con corrimiento;el proyecto también incluye un medidor de temperatura cuya lectura se multiplexará con el mensaje.De la misma forma que el mensaje y la temperatura el rótulo mostrará la fecha actual y la hora local.El microprocesador que se ha seleccionado para ésta aplicación es el Z80-A.

Toda la información que muestra el presentador esta adornada con efectos especiales que hacen agradable la observación y lectura de los mensajes.

El presentador puede ser programado desde un teclado a larga distancia ya que el diseño incluye un sistema de transmisión de datos en serie.

El desarrollo de éste trabajo se ha estructurado en tres capitulos.

El capitulo I esta dedicado a la recopilación de información acerca de los típos de tecnologías de visualizadores más utilizadas,sus ventajas y desventajas así como a los modos de direccionamiento de cada una.

En el capitulo II se abordan las implicaciones de los sistemas basados en microprocesadores , se hacen también comparaciones entre micros existentes para determinar cual es el más adecuado a nuestra aplicación.

Finalmente en el capitulo III se detalla el diseño del sistema en sus diferentes partes ,incluyendo en cada una la explicación del software utilizado.

Al final del documento, en la seccion de anexos se incorporan todos los listados de programas en ensamblador que soportan el funcionamiento del sistema.

TABLA DE CONTENIDOS

LISTA DE FIGURAS

Figura — Página

CAPITULO I

TEORIA SOBRE VISUALIZADORES

Introducción

El uso creciente de visualizadores en la vida común, ha obligado al hombre a mejorar y crear nuevas técnicas en el diseño de visualizadores. En la actualidad existen una gran variedad de éstos, es por ésta razón, necesario conocer sus principales caracteristicas a fín de hacer una elección adecuada del indicador para una aplicación específica.

En este primer capítulo se describe los tipos de visualizadores más comunes, incluyendo las ventajas y desventajas que presentan cada uno de ellos.

Luego se describen algunas técnicas empleadas para menejar los visualizadores, con especial énfasis a la tecnología de los microprocesadores, que ha llegado a ser la más eficiente.

1.1 TIPOS DE VISUALIZADORES

En la actualidad existen diferentes tipos de visualizadores, la mayoría de estos son fáciles de entender, pero se requiere muchas veces de algún detalle acerca de los circuitos necesarios para interconectarlos con un sistema digital.

Visualizadores fluorescentes

Estos visualizadores han llegado a ser muy populares en los últimos años, debido a su gran brillantez y facilidad para cambiar de color (a través de un filtro de luz). La desventaja del visualizador fluorescente es que requiere de tres diferentes tipos de voltaje: un voltaje de filamento, un voltaje de gría negativo y un voltaje de ánodo de 15 a 30V. Además la vida útil de estos visualizadores es muy corta. La luz emitida por estos visualizadores es azul-verde en vez de la típica luz roja del LED.

Este visualizador opera aplicandole una señal de 15 a 30V a cada ánodo que será iluminado. Al mismo tiempo un voltaje de gría debe ser cambiado de un voltaje negativo a un voltaje ligeramente positivo. Este procedimiento causa que el tubo comience a conducir y los electrones choquen en el ánodo, excitándolo, y de esta forma produzcan luz.

Puesto que este dispositivo requiere un voltaje de gría negativo para encender y apagar un segmento, el circuito de interface debe contener una fuente negativa. Cuando está disponible, esto no reprosenta ningún problema en el diseño del visualizador, pero cuando no lo está, el diseño involucra un costo adicional.

Visualizador en cristal líquido.

El visualizador en cristal líquido o LCD, es el más común en calculadoras y relojes digitales, debido a que comunmente se presentan de una manera conveniente para tal propósito. La ventaja de este display es su gran visibilidad en lugares bien iluminados en donde otros dispositivos son extremadamente difíciles de ver. Otra cualidad de estos visualizadores es que necesitan una extremadamente baja potencia para funcionar, lo cual es ideal en la mayoria de aplicaciones.

Este tipo de visualizador es el más difícil de usar, puesto que requiere de un voltaje de excitación AC, en vez de un voltaje de excitación DC. Este voltaje AC debe estar entre los 30Hz y 1000Hz, y entre los niveles lógicos normales TTL.

Visualizadores a base de LEDs.

El dispositivo más común y más simple para emitir luz, es el diodo emisor de luz o LED. La figura 1.1. ilustra la típica interface digital a LED. Note cómo un estandar componente TTL maneja el LED. La mayoría de LEDs requieren una corriente aproximadamente de 10 mA para producir su brillantez completa.

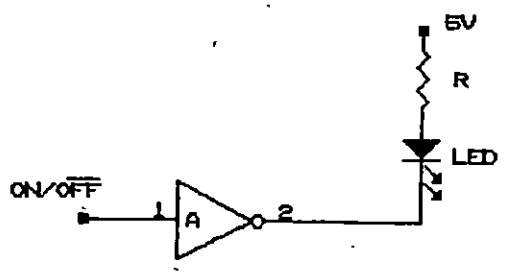


Figura 1.1 LED manejado por circuiteria digital.

Una compuerta de baja potencia 74LSXX no es capaz de producir suficiente corriente, para manejar un LED, ya que el valor de corriente que puede producir es alrededor de los 4 mA. Por lo tanto se debe tener cuidado si la interface no es capaz de suministrar la corriente necesaria para manejar el visualizador.

La figura 1.2 ilustra un visualizador de 7 segmentos en ánodo común en donde su estructura interna no es mas que un simple LED para cada segmento. La interface que maneja este arreglo de LEDs puede ser la misma de la figura 1.1.

En muchos casos los visualizadores son multiplexados y pueden requerir más corriente. En tales casos es común encontrar drivers de alta corriente formados por transistores o pares darlington.

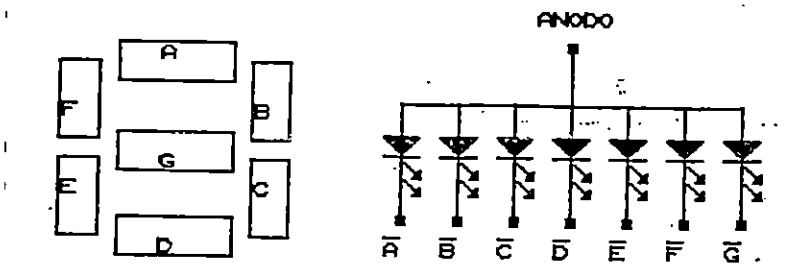


Figura 1.2 Display de 7 segmentos formado por LEDs

Debido a la necesidad de generar caracteres alfanuméricos de buena definición , los LEDs se agrupan en matrices de puntos tal como se muestra en la figura 1.3.

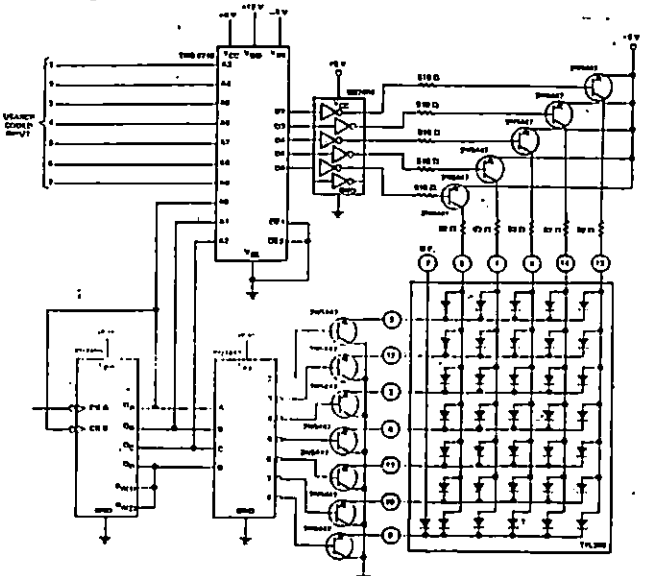


Figura 1.3 Arreglo de LEDs formando una matriz de puntos con su interfaz de potencia.

Este arreglo de LEDs es manejado por una etapa transistorizada dónde se deberá tomar en cuenta, que al manejar más caracteres la brillantez de los LED disminuirá, por lo tanto esta etapa deberá suministrar más corriente para mantener una buena brillantez.

1.2 TECNICAS EMPLEADAS PARA MANEJAR LOS VISUALIZADORES.

Obviamente, los visualizadores no serían nada sin algún método para activarlos de forma que visualizen precisamente la información que se desea y de forma inequívoca.

La aplicación concreta del usuario determinará el tipo de formato a utilizar, dependiendo de si se desea usar sólo caracteres numéricos (0-9), alfanuméricos parciales ó hexadecimales (0-9, A-F), ó bien todos los caracteres alfanuméricos completos.

Para la formación de los caracteres existen a su vez diferentes formatos según las tecnologías, pero se pueden reducir a: formato de 7 segmentos, de 9 segmentos, matriz de puntos y de caracteres ya formados.

La técnica para decodificar y direccionar los múltiples tipos de visualizadores es bastante similar, y aún más dentro de la misma tecnología. Por ejemplo, cualquier visualizador de 7 segmentos puede ser codificado y decodificado exactamente como cualquier otro. Las diferencias se estrablecerán en cuanto a la forma de diseñar el circuito a causa de las diferentes alimentaciones o requerimientos de tensiones y corrientes que pueda tener un visualizador con respecto a otro. Son precisamente éstas diferencias y sus costos asociados los que deberán ser evaluados para escoger entre varias alternativas posibles.

El manejo de varios caracteres en unos rótulos puede resolverse por medio de una decodificación y un direccionamiento hasta el infinito. No obstante, la economía impone el márgen de 4 a 6 dígitos o unidades sueltas del mismo circuito en tiempo compartido o por medio de multiplexado.

Multiplexado.

La multiplexación en el caso de los visualizadores digitales consiste en la utilización de uno o varios circuitos que serán comunes a todas las unidades sueltas de visualización o dígitos. La elección de cualquier técnica de multiplexado se basa preferentemente en motivos económicos, aunque también se utiliza en algunos casos unicamente para

reducir las interconexiones entre la lógica exterior y el visualizador. La norma general es efectuar una multiplexación a partir de cuatro dígitos o bien en unidades sueltas compuestas de matrices de puntos. En estos últimos el número de puntos suele ser de 28 (4x7) ó 35 (5x7) lo que viene a ser como cuatro o cinco visualizadores sencillos de 7 segmentos respectivamente.

Un ejemplo de multiplexado se muestra en la figura 1.4.

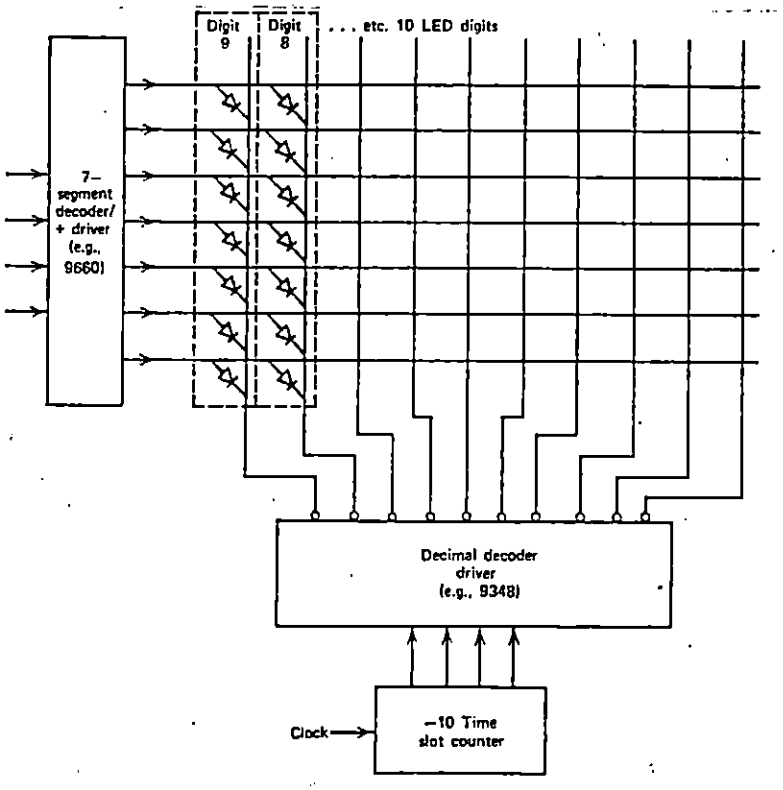


Figura 1.4 Ejemplo del método de multiplexado.

Visualizadores manejados por microprocesadores

En la actualidad existen muchas compañias que fabrican visualizadores. Estas compañias proporcian al consumidor una gran variedad de modelos, los cuales van desde modelos sencillos hasta muy complejos, en el cual se incorporan muchos efectos visuales y en los que además se mejora su forma de programación. La mayoria de estos modelos están manejados por microprocesadores o por una microcomputadora.

La idea principal de estos sistemas es la de controlar el visualizador (receptor) por medio de un sistema a base de microprocesador (fuente). Este esquema se muestra en la figura 1.5. en donde se puede observar a nivel de bloques cómo el sistema fuente proporciona la información que se mostrará en el receptor (visualizador).

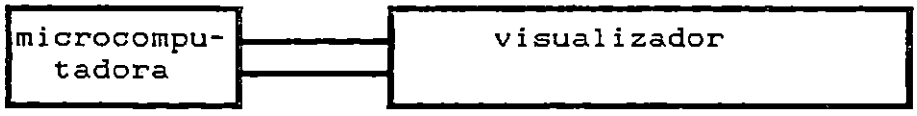


Figura 1.5 Visualizador manejado a base de microprocesadores

CONCLUSIONES DEL CAPITULO I

El emplear una de las tecnologías de presentación de información con que se cuenta en la actualidad dependerá específicamente de la aplicación particular que se le pretenda dar al dispositivo que se está diseñando, y un tanto así también de la inversión que se este dispuesto a hacer .

Debido a que en el país es muy difícil tener acceso a todas éstas tecnicas, es necesario evaluar cual de las tecnologías disponibles es posible llevar a cabo. El empleo de LEDs, para el diseño del visualizador resulta ser la solución más factible.

REFERENCIAS BIBLIOGRAFICAS

1. Ciarcia Steve. Construya Una Microcomputadora Basada En El Z-80. Editorial Byte Books/MacGraw Hill, 1986.

2. Mandado E., Tassis E. Diseño De Sistemas Digitales Con Microprocesadores. Publicaciones Marcombo, Boixareu editores.

3. Revista mundo Electrónico. No 56,junio,1982. Boixareu editores.

CAPITULO II

SISTEMAS BASADOS EN MICROPROCESADORES

Introducción

Con el constante desarrollo que se ha venido dando en la microelectrónica, en la actualidad se ha llegado a escalas de integración muy grandes, tales como la LSI. Estos niveles de integración han permitido crear dispositivos versatiles y programables, tales como el microprocesador .

En este capitulo se estudian los diferentes tipos de microprocesadores, ventajas y desventajas que presentan, además se recomienda que microprocesador puede utilizarse en nuestra aplicación. Luego de seleccionar el microprocesador se estudian los conceptos basicos relacionados a dicho componente. También se muestra la forma de relacionarlo con el mundo exterior y la forma en que el microprocesador maneja la memoria.

2.1 CONCEPTOS BASICOS

Un sistema basado en microprocesadores, generalmente está constituido por un microprocesador, que se organiza en la forma que describe en el diagrama de la figura 2.1 en la que se notan 5 unidades básicas, a saber 1) la unidad de entrada 2) las unidades de control 3) unidades de aritmética, contenidas en el microprocesador 4) la unidad de memoria y 5) la unidad de salida.

El microprocesador controla todas las unidades del sistema empleando la líneas de control mostradas a la izquierda de la figura 2.1. El bus de direcciones (16 conductores paralelos) selecciona un lugar en memoria, puerto de acceso o puerto de salida; el bus de datos (8 conductores paralelos), ubicado a la derecha de la figura, constituye una vía de dos sentidos para la transferencia de datos, ya sea hacia la unidad de procesamiento o desde la misma unidad. Es importante hacer notar que el microprocesador (CPU) puede enviar o recibir datos provenientes de la memoria haciendo uso del bus de datos. Si

un programa debe ser almacenado permanentemente, se le coloca en un dispositivo de memoria llamado ROM (memoria de sólo lectura). El ROM es generalmente un chip de memoria (CI) permanentemente programado. Para el almacenamiento temporal se emplea un dispositivo de circuito integrado RWM (memoria de lectura y escritura). Es práctica común llamar a la RWM memoria de lectura/estritura (RAM) . Los usuarios de las microcomputadoras almacenan en la sección de memoria RAM, junto con los datos, los programas de carácter temporal. En la figura 2.1 se muestran separadas las secciones de memoria RAM y ROM, debido a que por lo general son dos circuitos integrados que están separados.

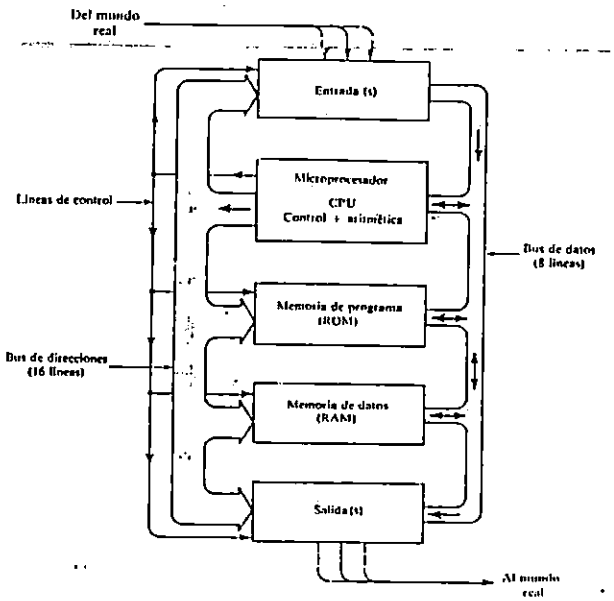


Figura 2.1 Organización de un sistema basado en microprocesadores.

El sistema mostrado en la figura 2.1 representa la organización de una microcomputadora, que también representa el modelo de cualquier sistema basado en microprocesadores. La mayoría de ellas está formada así, como mínimo, además de tener otras unidades. En aras de mayor claridad, en los diagramas de bloques se acostrumbra omitir la necesaria fuente de potencia, el reloj y algunas líneas de retroalimentación de la unidad microprocesadora.

2.2 MICROPROCESADORES QUE EXISTEN Y CUAL USAR.

En la actualidad existen muchos tipos de microprocesadores, los cuales se diferencian principalmente

por el tamaño del dato, capacidad de direccionamiento a memoria, velocidad, manejo de memoria virtual, etc.

La integración a gran escala ha permitido que los microprocesadores alcancen grandes capacidades, llamandose éstos de una generación avanzada, teniendo su principal aplicación en sistemas multiusuarios. Algunos de estos microprocesadores son:

68000	: Motorola
Z800	: Zilog
80286/386/486	: Intel

Los microprocesadores que antecedieron a esta generación, se utilizan actualmente en sistemas más simples, en dónde no es necesario software muy complejo, entre los más comunes tenemos:

8080/8085	: Intel
8088/8086	: Intel
6800	: Motorola
6502	: Tecnología MOS
Z80-A	: Zilog.

Debido a su estructura mas simple, y su bajo costo (en comparación a los microprocesadores avanzados), es necesario evaluar estos últimos 5 microprocesadores para decidir cuál de ellos es más aconsejable en nuestra aplicación.

Como consecuencia de la gran difusión de cada dispositivo, la documentación y el software puede obtenerse fácilmente. La disponibilidad de software compatibles con el 8080/8085 es la más grande y el costo es bajo, pero su complejidad circuital es también la mayor de todos los microprocesadores antes citados. El 8080A, aunque se describe como "una computadora de una sola pastilla", cuenta con varios excitadores-controladores externos y dispositivos de apoyo. Su configuración mínima está constituida por 3 pastillas integradas, como se muestra en la figura 2.2.

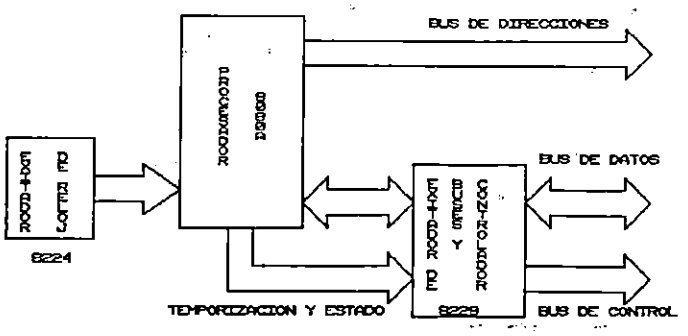


Figura 2.2 Configuración mínima del 8080A.

Un aspecto que debe evaluarse, junto con las especificaciones del hardware, es la forma de implementar el software en el sistema. La fig. 2.3 ilustra la estructura interna de los microprocesadores considerados aquí. Todos incluyen un acumulador, un registro índice o apuntador, algunos registros de próposito general, un stack pointer y contador de programa. El Numero de registros internos varía grandemente de uno a otro, generalmente un mayor número de registros, hacen al microprocesador más flexible y más fácil para escribir el software . En términos de utilización de memoria se mejora la velocidad y disminuye los costos

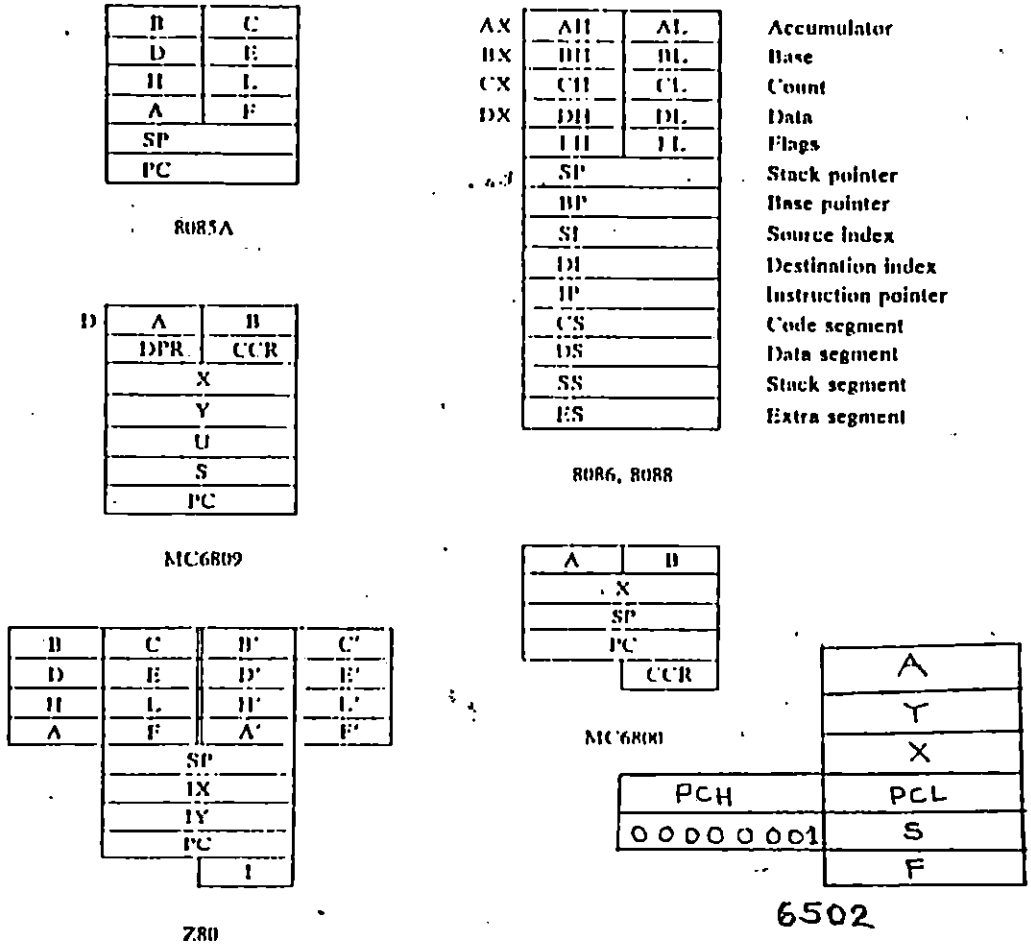


Figura 2.3 Modelos de programación internos, para el microprocesador 8080/8085, 8088/8086, MC6800, MC6809, Z80-A y el 6502.

Lo mejor de ambos mundos (software y hardware) está incorporado dentro del Z80-A. No solamente ejecuta el juego completo de instrucciones del 8080A, 6502 y del 6800, sinó que también tiene instrucciones adicionales que sirven para hacerle un procesador poderoso. En la figura 2.4 se ilustra

la estructura de buses del Z80-A. El Z80-A es algo más caro que los otros procesadores citados, sinembargo su circuiteria externa reducida da lugar a una compensación efectiva de los costos.

Así mismo, la facilidad de interconexión del Z80-A le hace ser la elección natural cuando se construye un sistema con microprocesadores, a partir de nada ya existente.

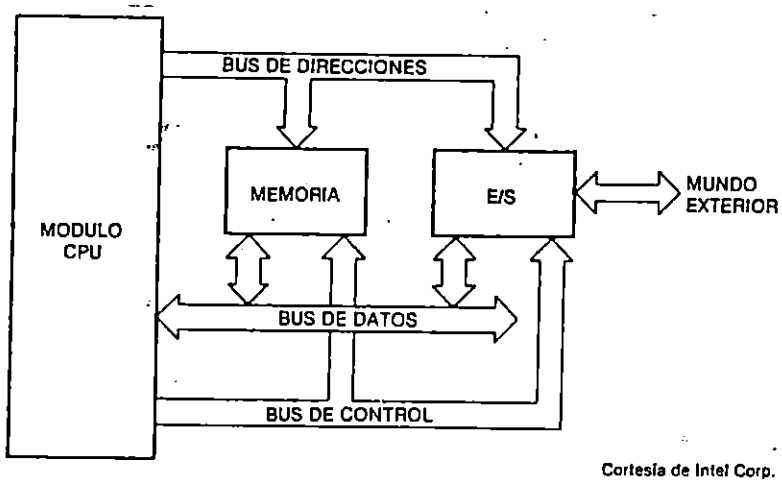


Figura 2.4 Configuración de buses del Z80-A

El Z80-A es superado únicamente por el 8088/8086, que es de una tecnología más reciente, y por lo tanto más eficiente . Este microprocesador debe tomarse en cuenta si se desea realizar proyectos de mayor complejidad.

2.2.1 FORMAS DE INTERFAZAR UN MICROPROCESADOR

La mayoría de los microprocesadores tienen poco valor funcional en sí mismos. La mayoría de ellos no contienen una memoria substancial y pocos tienen puertos de entrada y salida que se conectan directamente a los dispositivos periféricos. Los microprocesadores operan como parte de un sistema. La interconexión, o liga de las partes dentro de este sistema, se conoce como interfazado. Generalmente, una

interfaz es un limite compartido entre dos o más dispositivos; esto implica compartir información.

Un papel importante en el interfazado de dispositivos lo realizan los buses de dirección, datos y control. El interfazado se refiere a la sincronización y transmisión de los datos desde y hacia la CPU; por lo tanto, el software y el hardware deben ser estudiados.

2.2.2 CONEXION Y SINCRONIZACION CON EL MUNDO EXTERIOR

La utilidad de cualquier sistema, se basa, en cómo éste puede responder a un estímulo dado. Los sistemas basados en microprocesadores pueden comunicarse con el mundo exterior mediante operaciones de entrada y salida.

Una operación de entrada o salida es el acto de transferencia de datos desde o hacia un dispositivo periférico seleccionado. El microprocesador es el foco de todas esas operaciones; por lo tanto, una entrada significa que los datos fluyen a la CPU y una salida significa que salen de ella. Aquellas posiciones en donde los datos entran o salen, suelen ser llamados puertos de entrada o salida

Existen dos formas de manejar los puertos en un sistema basado en microprocesadores:

1- Memoria aislada de puertos
2- Mapeo de memoria para puertos.

La primera técnica utiliza las intrucciones IN y OUT para transferir datos hacia los puertos de E/S. La instrucción de salida se representa por el mnemónico OUT en los programas de lenguaje ensamblador, mientras que los instrucción de entrada utiliza el mnemónico IN. Estas instrucciones estan acompañadas de un byte, que corresponden al número de puerto o dispositivo. Para el microprocesador 8080/8085 o el Z80-A es de hasta 256 . La dirección de puerto viene de las ocho líneas de dirección menos significativas (A0-A7). La utilización de la instrucción OUT genera una señal especial llamada escritura de E/S (I/O W), así tambien la utilización de la instrucción IN genera una señal llamada lectura de E/S (I/O R). Ambas señales son activas en bajo para el Z80-A..

La transferencia de datos que se realiza mediante esta técnica puede ser iniciada por el dispositivo periférico diciendo: "estoy listo para enviar o recibir datos". Se emplean interrupciones en un dispositivo periférico para iniciar la acción de la CPU. Recuerde que cuando ésta última

recibe una requisición de interrupción termina la instrucción que está ejecutando y salta a una rutina de servicio de interrupciones en la memoria del programa. Esta rutina puede incluir operaciones de entrada y salida.

La segunda técnica maneja los puertos de E/S como posiciones de memoria, lo que obliga a que las lineas de direcciones sean decodificadas para seleccionar una dirección de entrada o salida exacta. Las señales de control usuales de escritura (WR) y de lectura (RD) sirven también para introducir o sacar datos. Cualquier instrucción de acceso a la memoria puede ser útil para introducir o sacar datos mediante la técnica de mapeo de memoria de E/S.

Esta técnica es probablemente la más común y puede utilizarse con cualquier microprocesador. La técnica de aislamiento de E/S sólo puede aplicarse con microprocesadores que tienen instrucciones IN y OUT por separado y salidas de control de escritura y lectura especiales de entrada/salida.

La E/S del mapeo de memoria se emplea en los microprocesadores comunes, tales como el Motorola 6800 y la tecnología MOS 6502. La E/S aislada (también conocida como la E/S de canal de datos) es utilizada por microprocesadores como el Intel 8080/8085 y el Zilog Z80-A.

Hasta el momento hemos supuesto que cuando el programa dirige a la CPU para dar entrada desde un puerto era válido que había datos disponible en esa posición. Pero podría no ocurrir esto, dado que los dispositivos periféricos (tales como un teclado) no actúan a la misma velocidad que la CPU. Por esta razón existen varias técnicas de sincronización de transferencia de datos. Estas técnicas son el polling y las interrupciones.

La técnica de polling también se conoce como E/S programada. Es el método más sencillo para sincronizar la entrada/salida y se utiliza en aplicaciones dedicadas pequeñas. La idea del polling es introducir o sacar datos en forma repetitiva utilizando en el programa una iteración . Podría construirse una interfaz de polling extremadamente sencilla para leer inicialmente el interruptor de entrada y entonces escribir esa condición en un indicador de salida LED. En éste ejemplo la iteración de lectura-escritura-lectura-escritura, sucesivamente, debería ser continua. La CPU está realizando un polling de entrada y actualizando la salida con intervalos de pocos microsegundos.

Con más frecuencia la iteración de polling pueden ser realizada para varios dispositivos. La CPU preguntaría al primer dispositivo si necesita servicio. Si el servicio se

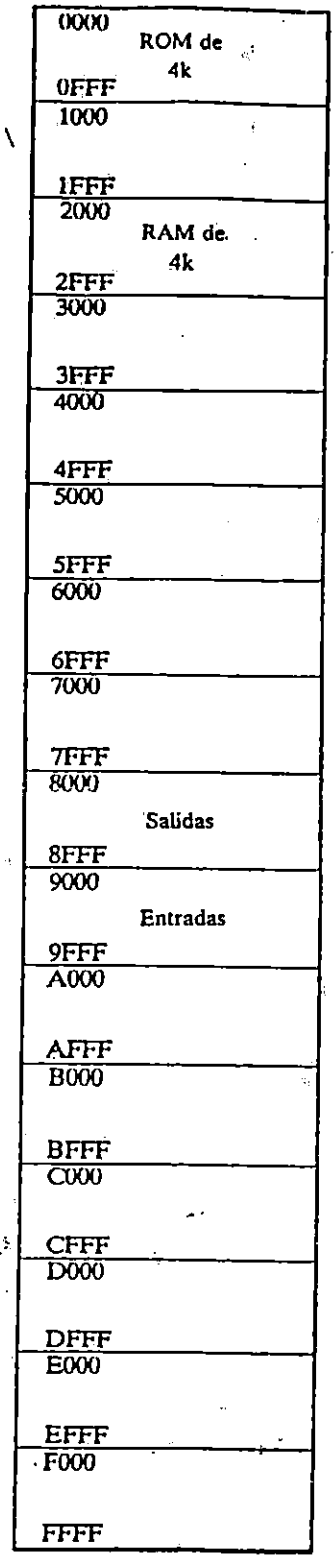
indicara con una condición de bandera de verdad, la CPU le proporcionaría servicio a ese dispositivo. Si aquello no ocurriera por una bandera falsa de condición, la CPU iría hacia un segundo dispositivo de E/S, sucesivamente.

El segundo método utiliza una línea de requerimientos y de interrupción para notificar al microprocesador cuando los datos están listos para ser tranferidos a la CPU. Luego que el microprocesador ha reconocido la interrupción, la CPU terminará la instrucción actual, almacenará el registro actual y el contador del programa con su contenido en relación con la fila, y brincaría hacia una rutina de servicio de interrupción especial. Después de hacerse cargo de la interrupción, la CPU regresará a la ejecución en el programa principal.

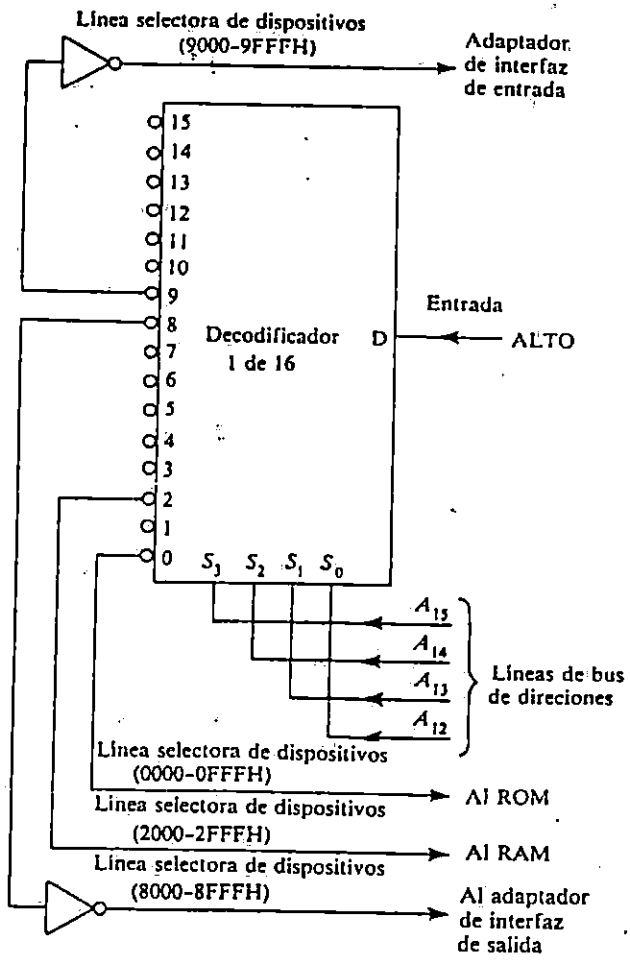
2.2.3 DECODIFICACION DE DIRECCIONES

En los sistemas basados en microprocesadores es necesario hacer una decodificación de direcciones, la cual garantiza que solo un dispositivo ha sido seleccionado

Con la decodificación de direcciones (memoria) podemos construir un mapa de memoria tal como se muestra en la figura 2.5a. En este mapa puede observarse que el total de memoria (64 kbytes) se ha dividido en segmentos de 4 kbytes. El primer segmento se utiliza para una ROM de 4K, mientras que el tercero se utiliza par memoria RAM. Los segmentos 9 y 10 son utilizados para puertos de salida y entrada, respectivamente.



a)



b)

Fig 2.5 a) Mapa de memoria b) Decodificador de memoria

La figura 2.5b muestra una forma sencilla de decodificar éstas direcciones. El circuito emplea un decodificador de 1 a 16 para seleccionar los dispositivos .

También existe otro método con el cual seleccionamos dispositivos. El método emplea compuertas NAND para generar los pulsos de selección de chip, como se muestra en la figura 2.6.

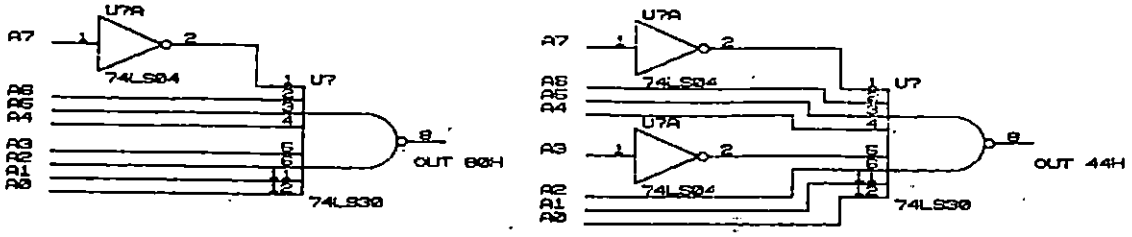


Figura 2.6 Decodificación por compuertas

En la figura 2.6, se muestra cómo el dispositivo que maneja la compuerta NAND, solo responderá a una dirección especifica, 80h y 44h.

CONCLUSIONES DEL CAPITULO II

Es difícil seleccionar un microprocesador para una aplicación específica sin tomar en cuenta los aspectos del hardware del sistema a diseñar. Una vez el hardware es especificado, es necesario considerar o análizar el software que se utilizará en el diseño particular.

Uno de los microprocesadores que satisface ésta aplicación es el Z80-A, pues es de fácil manejo tanto en su software como en el hardware que necesita para el funcionamiento.

REFERENCIAS BIBLIOGRAFICAS

1. L. Tokheim. Fundamentos De Los Microprocesadores. Serie Shaum, 1982.

2. Ciarcia steve. Construya Una Microcomputadora Basada En El Z80. Editorial Byte Books/MacGraw Hill, 1986.

3. Nichols C. ,Nichols Elizabeth A. ,Rony Peter R. El Microprocesador Z-80,Programación e Interfaces . Publicaciones Alfaomega-Marcombo.

CAPITULO III

DISEÑO DE UN ROTULO ELECTRONICO UTILIZANDO EL MICROPROCESADOR Z80-A

Introducción

Los sistemas basados en microprocesadores suelen estar asociados al manejo de grandes cantidades de información y a operaciones de alta velocidad,sinembargo su uso no está limitado a éste tipo de operación.

El caso que nos ocupa aquí es el diseño de un sistema de información (Rótulo) utilizando el microprocesador Z80-A,cuya complejidad puede considerarse en un término medio.

Este sistema se plantea en un principio como un circuito básico al que se le pueden ir incorporando nuevos elementos hasta obtener el sistema deseado.

A éste sistema básico se le adicionan periféricos tales como una pantalla de presentación y un teclado, desde el cual se tendrá el control.

Con respecto al display se hacen consideraciones en cuanto al tamaño adecuado de los caracteres ,modo de direccionamiento de la pantalla,y dimensiones del rótulo en general.También se hacen algunas consideraciones técnicas en función de mejorar la presentación.

Una etapa de sensoreo de la temperatura ambiente es diseñada e incorporada al sistema a traves de un puerto de entrada.

El software que se utiliza para la presentacion de la información en pantalla es abordada con algun detalle en éste capítulo,en fin todo lo relacionado al diseño del sistema está contemplado en éste capítulo.

3.1 UNIDAD CENTRAL DEL SISTEMA

Una impresión inicial que tenemos por ahora sobre los sistemas de computadora,es que la cpu,aunque crítica, es solamente una parte de un sistema en el cual una gran cantidad de componentes deben interaccionar en armonia y sincronización.

No es fácil imaginar que un solo componente realiza funciones de entradas de datos,proceso,y presentacion de información.Todas las operaciones son complementadas por diversos elementos que actuan de forma interactiva teniendo una finalidad común,la de echar en marcha un sistema.

Este sistema aunque sencillo presenta rasgos semejantes al de una computadora dónde la parte inteligente la constituye el microprocesador.Este elemento es en principio el encargado de coordinar todas las las funciones importantes ya sean éstas de decisión o de ejecución.

Muchos circuitos digitales disponibles, son incapaces de realizar tareas sin el auxilio de componentes externos,tal es el caso del microprocesador Z80-A que aunque poderoso con su set de instrucciones muy variado,es un elemento inoperante sin el concurso de circuitos externos adicionales.

En primera instancia se nos viene a la mente que el microprocesador ejecuta programas ;es evidente que el primer elemento necesario para la operación de un sistema con microprocesador es,la memoria,pues sin éste recurso no podrian guardarse programas que le ordenen realizar tareas específicas. Naturalmente, se trata de una parte muy importante en cualquier sistema de computadora .Tanto las instrucciones de programación como los datos deben ser almacenados en el tiempo adecuado de tal forma que la computadora pueda realizar su función.

Aún cuando el procesador central del Z80-A tiene una gran cantidad de registros de almacenamiento de 8 bits,éstos pueden ser utilizados solamente en la manipulación temporal de datos y no pueden almacenar instrucciones de programa. Las instrucciones de programa deben ser almacenadas en elementos de memoria exterior.

Ciertamente este sistema incluye un bloque de memoria la que esta compuesta por un paquete de ROM para almacenar todos los programas que soportan el funcionamiento del sistema;y un paquete de RAM para guardar datos, que son resultado de operaciones intermedias y para la programación que haga el usuario.

En un inicio la CPU Z80-A es provista de un puerto de entrada y un puerto de salida,estos circuitos, incorporados al bus de datos del Z80-A ,permiten la comunicación de información y datos entre los elementos de memoria o el microprocesador y el usuario o sistemas externos .Un puerto de entrada será empleado más adelante para proveer datos provenientes del sensoreo de la temperatura ambiente ,o ya sea para adaptarle al sistema una entrada de datos desde teclado.

El puerto físico de salida se le provee en principio como un medio de acceso a la información que procesa la CPU Z80-A,ésto significa que en cualquier instante se puede ampliar éste esquema inicial y adaptarle una etapa de visualización que permita la observación de los resultados obtenidos en las diferentes tareas que realize el sistema.Desde luego en este sistema es indispensable una etapa de presentación.

Todo lo que se ha escrito en lo que va de éste capitulo se resume en el esquema de la figura 3.1.Esta configuración constituye un sistema sino mínimo ,muy básico que puede ser implementado utilizando el Z80-A .

A este pequeño sistema ya es posible ordenarle mediante un programa, que realize funciones de entrada de datos,procesamiento ,y salida de los resultados;o simplemente procesamiento y salida de información.

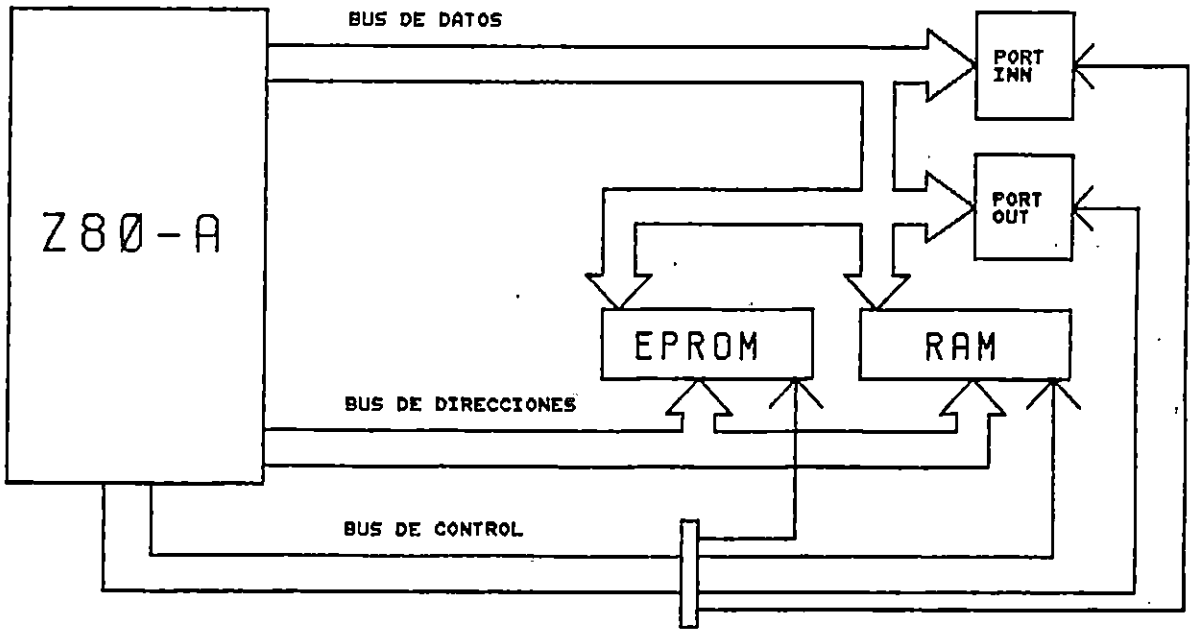


Figura 3.1 Sistema básico utilizando el Z80-A

Las memorias semiconductoras seleccionadas para éste diseño son: EPROM / TMS 2732A con capacidad de 4K y palabra de 8 bits, y RAM / HM 6116 con capacidad de 2K y palabra de 8 bits.A nuestra consideración éstos bloques de memoria son más que suficiente para la aplicación que aquí nos ocupa; o en todo caso si el software es mucho ,solo es cuestión de seleccionar una de mayor capacidad de alojamiento de datos.

Las hojas de especificación de éstos componentes son incorporadas en la sección de anexos de éste documento.

A éste bloque inicial se le provee de otro elemento circuital que le imprime al sistema seguridad en su función. Manejadores de corriente (buffers) son incorporados en el bus de datos y en el bus de direcciones (observe la fig. 3.2),para protección del microprocesador.El chip utilizado para esta finalidad es el 74LS245.

Los buses de datos y direcciones actuan en la mayoria de los casos sobre la entrada de muchos dispositivos en paralelo,todos los cuales consumen alguna potencia de entrada,dichos buses deben tener una corriente de salida que satisfaga la demanda de carga.Además cada pin del microprocesador Z80-A solo puede manejar una carga TTL y en algun instante de error podrían haber dos o más cargas actuando y en consecuencia dañarlo.

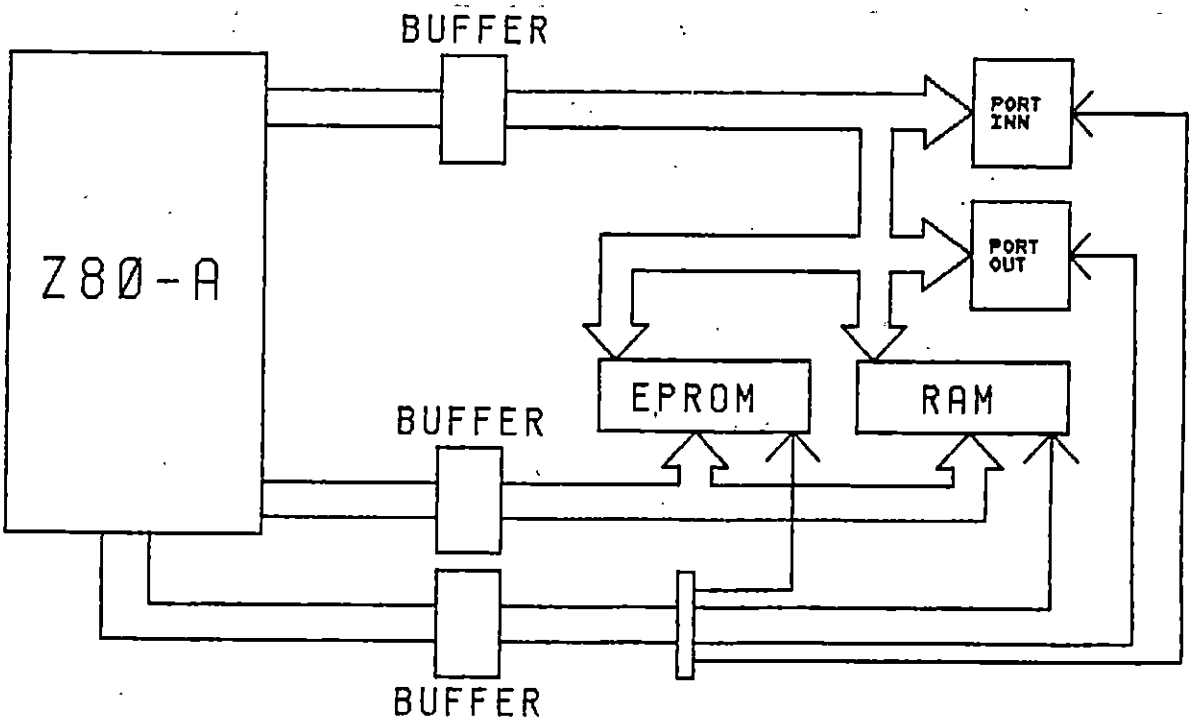
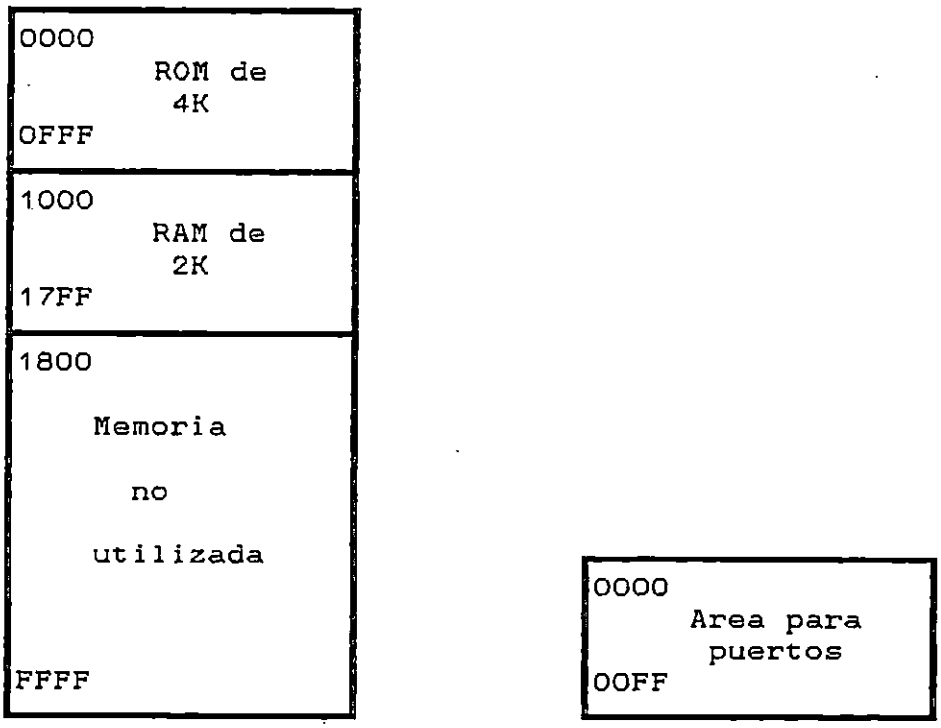


fig 3.2 Esquema de protección al Z80-A

Hecho ésto, el aspecto de la carga de los buses ya no es de cuidado,pudiendo hacerse pruebas y operar el sistema en un ambiente más confiable.

Todo sistema basado en microprocesadores maneja dispositivos externos, como los que acabamos de mencionar. Para poder manejar dichos dispositivos (memorias ROM,y RAM, puertos de entrada y de salida), es necesario distribuirlos dentro de la capacidad de direccionamiento a memoria que tiene el microprocesador. A la distribución de dispositivos en el total de memoria accesible se le conoce como MAPA DE MEMORIA. En la figura 3.3, se muestra el mapa de memoria empleado en nuestro sistema. En ésta figura observamos que el microprocesador Z80-A es capaz de manejar la memoria y los puertos de forma independiente. Esto significa que además de los 64k bytes de memoria que direcciona el Z80-A, puede direccionar 256 bytes de memoria adicionales que pueden utilizarse como puertos de entrada o como puertos de salida.



Fugura 3.3 Mapa de memoria del sistema
a) área de memoria, b) área de puertos para Z80-A

Los bloques de memoria que se muestran en la figura 3.3 son divididos en otros segmentos, los que son utilizados para alojar las diferentes partes de la programación del sistema. La figura 3.4 muestra la utilización que se le ha dado a los diferentes segmentos que forman los bloques de memoria ROM y RAM.

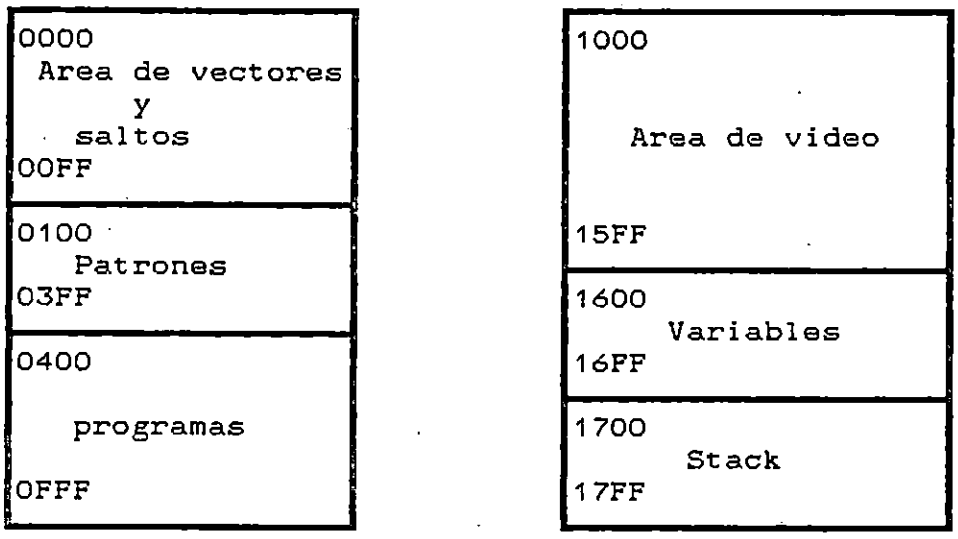


Figura 3.4 Distribución de la memoria. a) ROM, b) RAM

La figura 3.4 muestra que la dirección 0000 a 00FF (página 0) se ha utilizado para saltos, éstas posiciones son utilizados para reconocer las interrupciones INT, NMI e interrupción de inicialización. En las siguientes direcciones (0100 a 03FF) se ubican los patrones que definen los caracteres en el visualizador. Las direcciones finales de ROM son utilizados para alojar los programas y rutinas que utiliza el microprocesador.

La memoria RAM es utilizada principalmente para alojar la información a visualizar, por tanto del total de memoria (2K byte) 3/4 partes son utilizadas para éste propósito; la cuarta parte es dedicada para uso de variables de los programas y para el STACK.

Otro componente importante en el hardware del sistema es el circuito de reset,arreglo que proporciona la inicialización correcta del sistema.La no consideración de éste circuito acarreará muchos problemas a cualquiera que pretenda diseñar un sistema basado en el microprocesador Z80-A.

Una entrada de reset suele ser manual,automática ,o una combinación de ambas formas .La fig.3.5 muestra la comfiguración que utilizamos como inicializador del sistema.

El circuito de reset entra a funcionar cada vez que se enciende el sistema ,activa el pin de inicialización del Z80-A y el Z80-A a su vez direcciona la posición de memoria donde debe estar la primera instrucción de programa a ejecutar.El Z80-A al ser inicializado correctamente arranca direccionando la memoria en la posición 00; la no inclusión del inicializador ,provoca que el Z80-A direccione al azar la posición de memoria inicial y por lo tanto realiza funciones indeseadas e inprevistas.

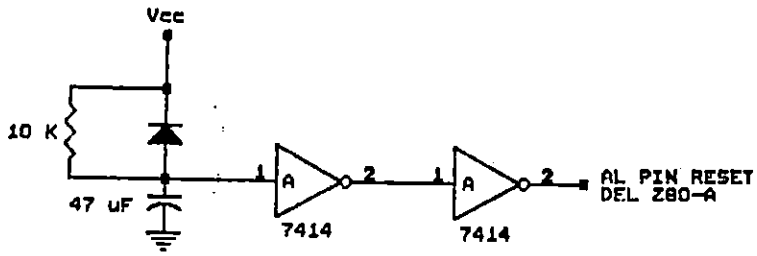


figura 3.5 Circuito inicializador del sistema

Debe también tenerse cuidado con las entradas del Z80-A,los pines que no sean utilizados deben ser puestos en su estado lógico inactivo,ya que en la mayoria de los casos el no desactivarlos interfiere con el buen funcionamienro del circuito, ya que le introducen inestabilidad especialmente cuando se está trabajando a alta frecuencia.Por ejemplo el pin WAIT del Z80-A es una línea de entrada la cual podría no ser utilizada y en dicho caso debe ser puesta a su estado inactivo.

3.2 ETAPA DE VISUALIZACION

Uno de los objetivos principales del sistema que se esta diseñando es la presentación de información,éste atributo indica que el diseño debe incluir una etapa de visualización que sea capaz de mostrar en forma clara información pertinente.

Aqui se pretende diseñar un display de gran tamaño en el cual sean visible muchos caracteres y formen mensajes de

gran significado y que en general ofrezca información útil a los usuarios.

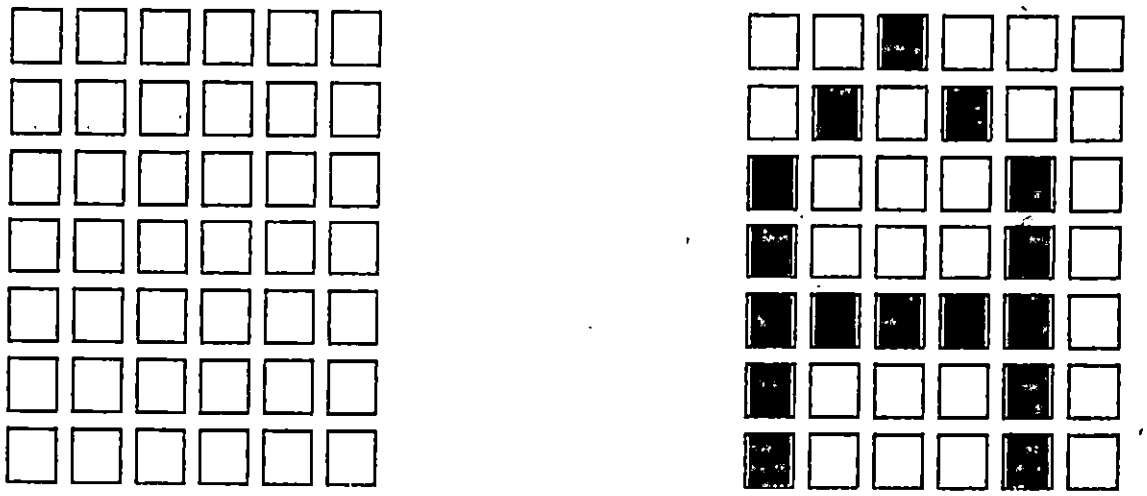
El display será diseñado en base a un arreglo matricial de puntos (leds) y será capaz de mostrar simultaneamente 10 caracteres alfanumericos .Cada caracter será formado por una matriz de 7 filas y 6 columnas incluida una columna de puntos para separación entre caracteres.

3.2.1 FORMA DEL DISPLAY Y CAPACIDAD DE ALOJAMIENTO DE CARACTERES

Un esquema de visualización en siete segmentos es una alternativa muy aceptable en sistemas que requieren presentación de caracteres numéricos,sinembargo en sistemas muy interactivos con el usuario como se pretende que sea éste ,el método de presentación debe ser más general y versátil.

La presentación de caracteres en base a matrices de puntos ofrece la posibilidad de generar cualquier caracter alfanumérico aunque su direccionamiento es de mayor complejidad que el requerido para otras técnicas.

Una matriz de puntos es como la mostrada en la fig.3.6a y un caracter se forma como en la figura 3.6b.Esta matriz consta de 35 puntos y en la realidad puede estar formada por bombillos incandescentes,leds,u otros dispositivos luminosos afines con la aplicación.



a) b)

Figura 3.6 Formación de caracteres
a)Matriz de puntos b)Caracter formado

El esquema de presentación utilizado en éste sistema es un arreglo matricial de puntos luminosos,en el cual la matriz básica incorporada,es como la de la figura 3.6a y esta constituida por 35 leds de color rojo arreglados en 7 filas y 5 columnas.

Si disponemos consecutivamente 10 matrices como la mostrada en la figura 3.6a formamos un display en el que se pueden visualizar palabras o mensajes de hasta 10 caracteres.Esta idea se muestra en la figura 3.7 dónde también se puede advertir la incorporación de columnas de leds entre cada par de matrices para separacion de los caracteres.

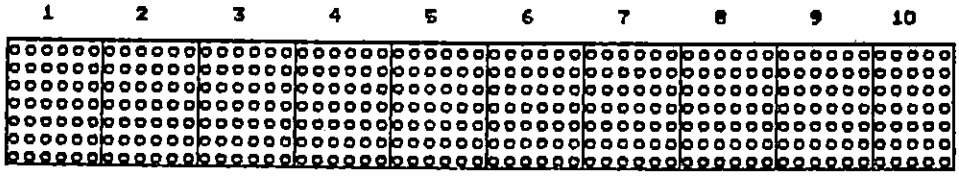


Figura 3.7 Matriz de leds para mostrar 10 caracteres

Cada punto de la matriz está realmente constituido por un led geometricamente redondo, tal y como se observa en la figura anterior.

3.2.2 DIMENSIONES DEL PRESENTADOR

Hay muchas consideraciones que deben hacerse antes de decidir sobre el tamaño del display que se deba incorporar a un sistema en particular que lo requiera

El tamaño de los caracteres esta determinado por las condiciones normales de interacción entre el operario o usuario y el dispositivo, o por la aplicación particular del sistema .

El tamaño del caracter ofrecido por el presentador que hemos diseñado,es tal que los mensajes pueden leerse perfectamente a una distancia promedio de 15 metros, ésta distancia es tomada como un elemento inicial de diseño y así mediante un proceso experimental de observación ,determinamos el tamaño adecuado.

Debemos mencionar que los leds utilizados tienen un diámetro de 5 mm y también fueron tomados como dato inicial del diseño.

Las medidas de dimensión reales de un caracter mostrado por el presentador ,son las especificadas en la figura 3.8 (no esta a escala).

La separación entre los emisores de luz depende del diámetro del led ,y del requerimiento en cuanto a tamaño y definición del caracter; un caracter puede ser grande pero su definición muy mala si el diámetro de los leds es pequeño pues la separación entre los puntos sería considerable.

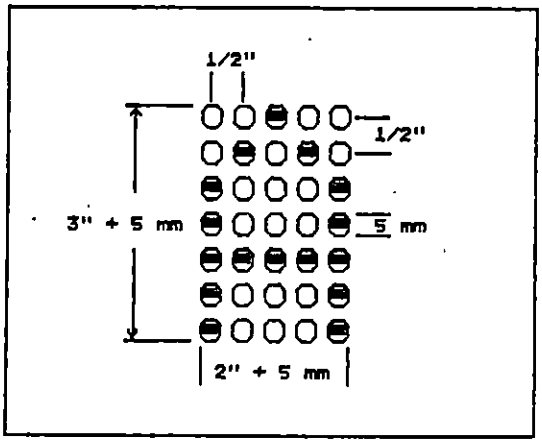


Figura 3.8 Dimensiones del tipo de caracter mostrado por el presentador

Las dimensiones de la pantalla donde se formarán hasta 10 caracteres están especificadas en el esquema de la figura 3.9.Estas dimensiones forman el área efectiva de presentación del dispositivo, y está formada por 413 leds,arreglados como matriz de 7 filas y 59 columnas. 9 columnas de emisores de luz son utilizados como separadores de caracteres. El dibujo de la figura 3.9 no esta a escala pero las dimensiones especificadas en el, son las reales del diseño.

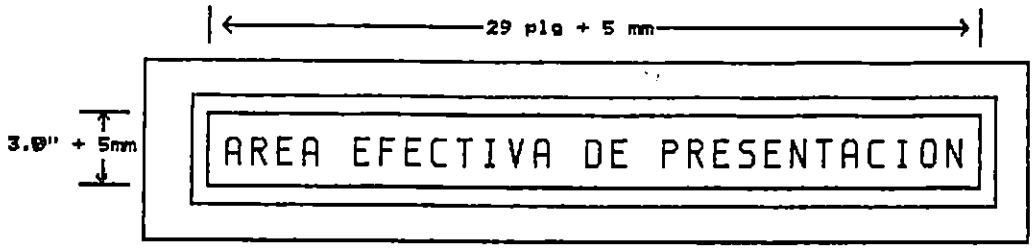


Figura 3.9 Area efectiva de presentación

3.2.3 DECODIFICACION Y MANEJO DEL DISPLAY

La figura 3.10 muestra el arreglo de leds así como las conexiones hechas para formar la matriz-display.

Las columnas son manejadas por drenadores de corriente (IC 2011) que manejan hasta 600 mA por canal .El IC 2011 es un chip inversor de 7 canales ,compatible con dispositivos TTL y que ofrece al exterior 7 pines para el control (switcheo) y 7 pines para conectar la carga a cada uno de los canales respectivos.

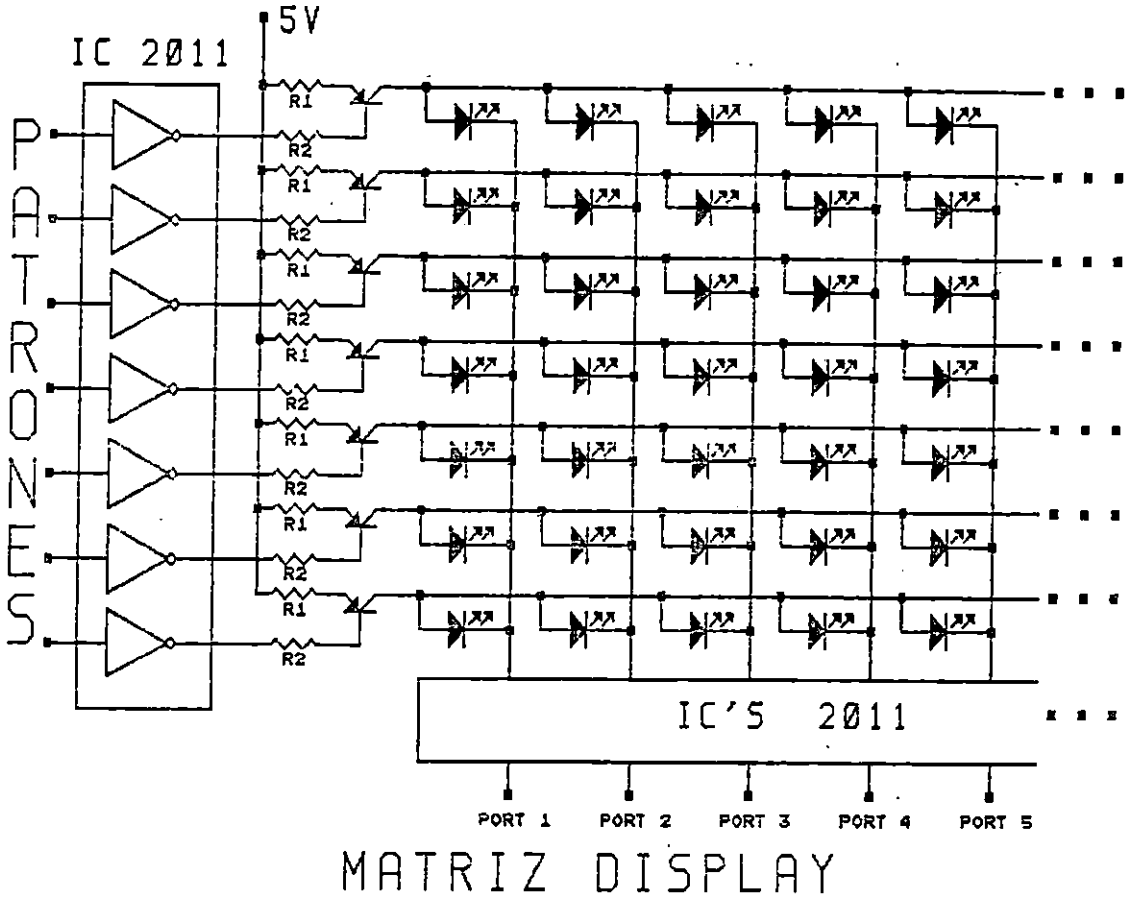


fig.3.10:Matriz display con sus manejadores de potencia

Las filas al igual que las columnas no pueden ser manejadas directamente por circuiteria común,es necesario ,montar una pequeña interface de potencia constituida por 7 transistores PNP manejados a su vez por inversores amplificadores (un IC 2011),los cuales responden a la excitación directa de los bits patrones que están presentes en los canales de un puerto de salida.Los transistores de ésta interface son puestos a funcionar como interruptores lógicos para lo cual se hacen operar en dos puntos estratégicos : CORTE Y SATURACION.

Para una mejor comprensión del esquema utilizado en el manejo de las filas observe la fig. 3.11,ahí cuando el bit patron es un cero (0) el inversor 2011 pone un uno (1) a su salida y ésto obliga a que Q1 se ponga en corte y en consecuencia la fila queda deshabilitada; por otro lado si el bit patron es un uno (1) y con la adecuada selección de las resistencias , Q1 es llevado a la saturación y así la fila queda activada.El transistor utilizado en la interface es el ECG 159 que maneja una corriente máxima de 1 amperio y puede trabajar a una frecuencia de hasta 200 Mhz.

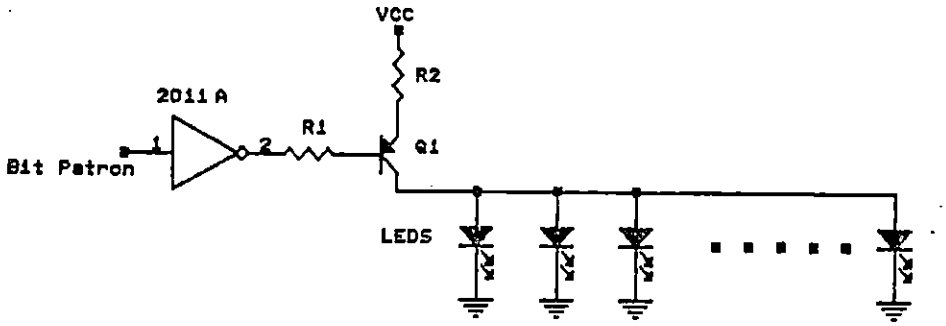


figura 3.11 Manejo de las filas en el display

3.2.4 ALGUNAS CONSIDERACIONES TECNICAS

La circuiteria de manejo y de control del display estará dispuesta en el presentador mismo. La tableta display y la tarjeta de control estarán depositadas dentro de un gabinete como el mostrado en la fig. 3.12. La fuente que alimenta este dispositivo también estará dispuesta dentro del gabinete.

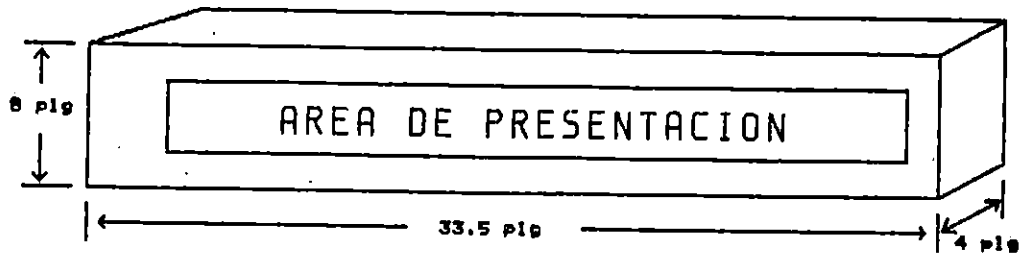


figura 3.12 Dimensiones del gabinete.

La caja está pintada de color negro para ofrecer un fondo oscuro que permite un mayor realze del área de presentación.También el área de presentación está cubierta por una tapadera de acrílico pintada de color oscuro en tonalidad suave,lo cual permite que solamente sean observados los leds seleccionados por decodificación;ésto favorece la definición de caracteres y refleja una sensación de mayor brillantes .Además el material acrílico le imprime a los puntos brillantes una cierta distorsión ,que hace que éstos se vean más grandes y más continuos produciendose un efecto de agrandamiento del caracter.

3.2.5 SOFTWARE EMPLEADO PARA LA VISUALIZACION DE LA INFORMACION

El manejar un visualizador es una tarea que implica que un sistema esté enviando información.

Una manera muy eficiente de manejar un visualizador puede lograrse empleando la técnica de multiplexado. Esta técnica consiste en enviar una fracción de la información en un pequeño lapso de tiempo (fracción de segundo). Luego enviar otra fracción de la información, en un tiempo posterior, y así sucesivamente hasta completar la información

En base a éste principio se diseñó un programa que realiza ésta función. La tarea del programa consiste en simular la técnica de multiplexado, lo cual implica presentar fracciones de caracteres en diferentes periodos de tiempo, para nuestro caso cada caracter esta constituido por 5 fracciones o patrones ,tal como se muestra en la fig 3.13.

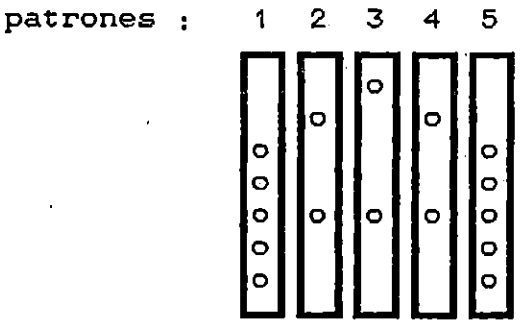


Figura 3.13 Esquema que muestra los patrones que forman un caracter en matriz de puntos.

Esto quiere decir que para formar un caracter necesitamos suministrar 5 patrones al visualizador.Estos patrones son almacenados en ROM en forma consecutiva para cada caracter y a un espacio (byte) por medio entre caracter y caracter. Ver anexo C para lista completa de patrones utilizados en el programa.

El objetivo principal del visualizador es poder mostrar la información que ha sido almacenada en un área de memoria RAM. Esta área de memoria es llamada área de video. En el área de video se almacena la información en forma de códigos ó valores hexa que representan a cada caracter, es decir que cada letra, número ó símbolo tiene su equivalente hexadecimal, ejemplo de esto es 12h representa la Q ,16h representa la T, etc.

Como la información que se mostrará, se almacena en posiciones de memoria consecutivas, el programa lee cada caracter de dicha área y envia sus patrones correspondientes al visualizador. Esto se muestra en la figura 3.14.

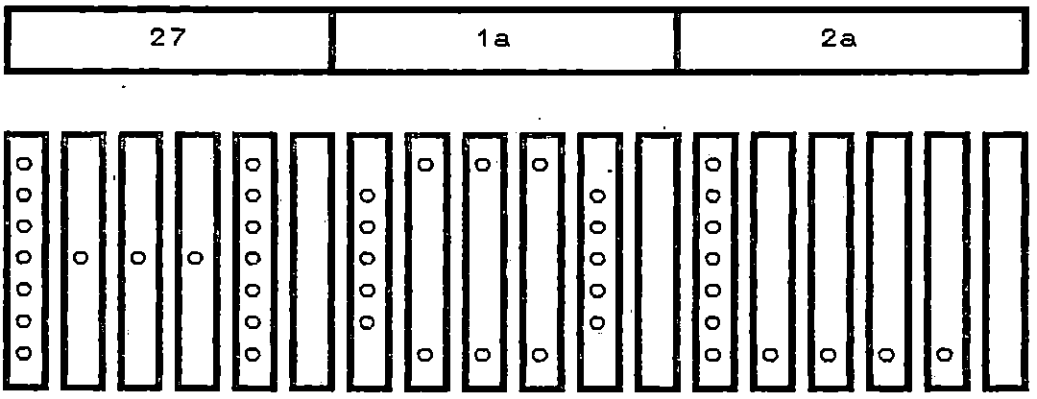


Figura 3.14 Representación de la información en códigos hexadecimales y patrones.

Debido a que cada código en el área de video es formado por 5+1 patrones (el +1 corresponde al espacio en blanco entre caracter y caracter) son necesarios 60 patrones para obtener 10 caracteres visibles en el visualizador. Esto significa que el programa proporciona 60 patrones para llenar una pantalla, ésto se hace por medio de la multiplexación constante, para así obtener el efecto de un mensaje estático.

En la figura 3.15, se muestra el flujograma del programa empleado para mostrar la información que se ha almacenado en el área de video. Aquí se puede observar como existen tres lazos dentro del flujograma. El más interno se repite 6 veces, lo cual se debe a que es el número de patrones que se necesitan para formar un caracter, el siguiente más externo es utilizado para suministrar los 60 patrones al visualizador, esto quiere decir que mientras no se hayan proporcionado los 60 patrones al visualizador, el programa no puede efectuar otras tareas. Al finalizar éste ultimo lazo, el programa es inicializado mediante el uso del lazo mas externo y con ésto el programa es obligado a leer el primer dato del área de videó para que se muestre en la primera posición de la pantalla. Para manejar éstas dos variables, en el programa se emplea el par de registros HL para apuntar a la dirección del caracter en el área de video que se mostrará en pantalla. Tambíen se utiliza el registro D para indicar la posición ó columna donde el patrón se presentará en el display, es decir si "D" es igual a 5 el patron que suministra el programa servirá para la columna 5.

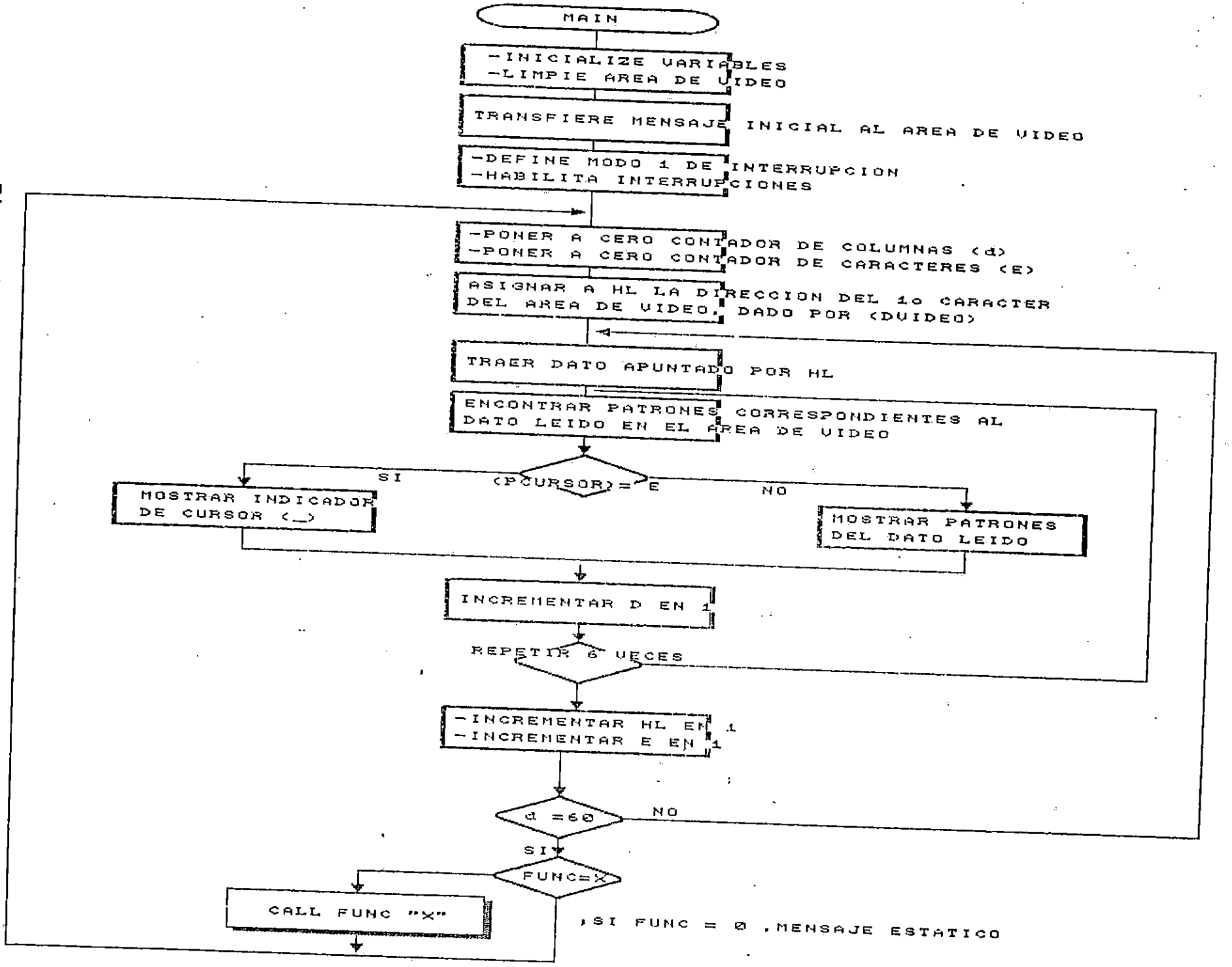


Figura 3.15 Flujoglama de la rutina principal

El control de las columnas es manejado por el programa mediante el uso de las instrucciones de escritura en puertos. Con éstas instrucciones las columnas son manejadas como puertos y el dato enviado al puerto constituye el patrón que formará el caracter.

El diseño de un visualizador requiere de una programación, la cual constituirá el mensaje a mostrar. Esta programación exige que el usuario pueda observar los datos que tecléa, para ello el programa indíca la posición de donde el dato se escribirá en pantalla. Esto se logra mediante un "cursor".

Para lograr éste objetivo se utiliza una variable llamada "PCURSOR", la cual contiene la posición de dónde el cursor se aparecerá. Otra variable necesaria para mostrar el cursor es el registro "E". Esta variable indica cual caracter se mostrará . Si el registro E es igual a la variable PCURSOR se muestra el cursor.

Como parte adicional al sistema se suministran efectos especiales a los mensajes, los cuales dan mayor realse a la presentación. Estos efectos son obtenidos mediante el uso de funciones.
Las funciones incorporadas al visualizador son:

1. Corrimiento horizontal
2. Corrimiento vertical pausado
3. Efecto de caida libre con empaquetado
4. Efecto de desvanecimiento

El uso de cualquiera de las funciones dentro del programa es logrado por medio de una variable llamada "FUNC". El valor de ésta variable determina cual de la funciones se ejecutará. Por ejemplo si al final de completar los 60 pratrones FUNC es igual a 1 se producirá un corrimiento horizontal.

El detalle de las rutinas o funciones de efectos especiales se muestran en la sección de anexos.

3.3 ETAPA PARA SENSAR LA TEMPERATURA

Los fenómenos físicos de interes a la ingeniería generalmente no pueden ser cuantificados en forma directa,sus variaciones son detectadas mediante elementos denominados transductores (sensores) los cuales describen el fenómeno físico generando señales proporcionales o no proporcionales que son facilmente medibles y de mayor versatilidad para el acondicionamiento e interpretación.

Para sensar la temperatuta ambiente, utilizamos un IC transductor que genera corriente eléctrica en proporción directa con la variación de la temperatura.Este transductor es el AD-592 (fig.3.16),un IC de tres (dos) terminales que al conectarse a una fuente de DC de entre 4 a 30 voltios impulsa una corriente eléctrica de 1 microamperio por cada grado centígrado en que se incremente la temperatura;por otro lado si la temperatura disminuye en 1 grado centígrado la corriente impulsada disminuirá en en un microamperio.

El AD 592 puede ser utilizado en aplicaciones entre -25ºC y 105ºC.Particularmente es usado en aplicaciones de sensoreo remoto.

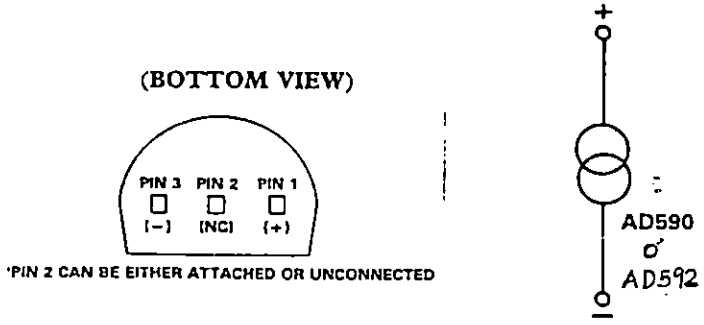


figura 3.16 Diagramas del AD 592.

El AD 592 es un ideal sustituto de las viejas tecnologías de transductores de temperatura como termistores,diodos,RTDS,termocuplas,etc.

3.3.1 CONSIDERACIONES SOBRE EL AD 592

1- Excelente linealidad.
2- Dado que el AD 592 es una fuente de corriente dependiente de la temperatura,es inmune a los voltajes de ruido.
3- Rango de temperatura de operación : -25ºC a 105ºC.
4- Fuente de voltaje de operación : +4V a 30 Voltios.
5- A 0ºC el AD 592 genera 273.15 microamperios.
6- Entrada:Temperatura / Salida:Corriente.

El dato de temperatura debe ser enviado hacia un medio visual y atraves del bus de datos de la CPU Z80-A.Este dato debe ser acondicionado a un código digital que pueda ser leido atraves de un puerto para introducirlo a la parte inteligente del sistema para luego ser enviado hacia el display en códigos conocidos.

Esta conversión de códigos es realizada atraves de un ADC el cual acepta a su entrada ,niveles de tensión de un rango específico los que son traducidos a una palabra binaria de 8 bits que puede ser interpretada por cualquier componente digital que sea compatible con el ADC.

Un pequeño inconveniente se advierte en la utilización del ADC que se ha seleccionado pues el sensor de temperatura es un generador de corriente y el ADC acondiciona voltajes.No es nada serio en verdad ,la solución se obtiene, haciendo circular la corriente del transductor atraves de una resistencia de valor estratégico ,y así se porvoca un voltaje que servirá como entrada al ADC.

El ADC que se ha utilizado como interface entre el sensor y el microprocesador es el AD 7820.Este es un convertidor A/D de palabra de 8 bits y alta velocidad de conversión ;éste componente funciona a Vcc=5V y se comfigura como un puerto de entrada que será leido periodicamente por el Z80 A.

El AD 7820 tiene dos pines importantes nominados V+ y V- ,estos pines sirven para marcar el rango de voltajes analógicos dentro del cual se estará haciendo la conversión.

Como el AD 7820 tiene 8 bits de salida ,puede realizar 256 conteos binarios que serán completados dentro del rango V+,V- seleccionado,así:
El rango de voltajes analógicos seleccionados para hacer la conversión es dividido en 256 incrementos de magnitud identica,siendo cada uno equivalente a un conteo digital.Para nuestro caso el AD 7820 no acepta niveles de señal de más de 5 voltios por lo que hemos escogido como límite superior de la conversión (V+),una señal de 5 voltios,y como límite inferior (V-) a 2.44 voltios.

Esta selección no ha sido una arbitrariedad sino mas bién una decisión acomodada de tal forma que cada conteo digital equivalga a 10 mV analógicos.

Por tanto:

$$(V+ - V-)/256 = (5.00 - 2.44)/256 = 0.01 = 10 \text{ mV/conteo}.$$

Un esquema que muestra la relación existente entre la CPU Z80-A/el ADC,y el AD 592 se muestra en la figura 3.17.

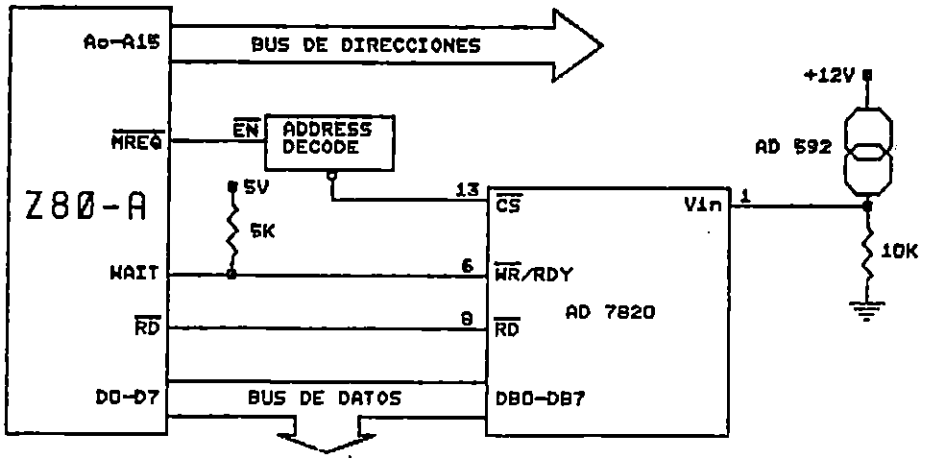


Figura 3.17 Interconexión entre AD 592,ADC,y CPU Z80-A

Ahora podemos seleccionar el valor del resistor por el cual debe circular la corriente del transductor.

De la hoja de datos del AD 592 sabemos que este produce aproximadamente 273 microamperios a 0ºC,y si un microamperio circula por una R=10K tenemos:

V = I R = 1 E-6 X 10 E 3 = 0.01 Voltios = 10 mV.

Esto implica que por cada microamperio que produzca el tranductor se generan 10 mVoltios.

Luego a 0ºC:

V = I R = 273 E-6 X 10 E 3 = 2.73 Voltios.

De todo esto podemos concluir que: el cero digital seleccionado en el convertidor es equivalente a 2.44 voltios; y que la temperatura real 0ºC es equivalente a 2.73 voltios y está desviada en 29 conteos del cero digital.Este error generado es corregido por software en el programa de presentación de la temperatura.

3.3.2 SOFTWARE EMPLEADO PARA MOSTRAR LA TEMPERATURA

Una función que se ha incluido en el diseño del rótulo eléctronico es la presentación de la temperatura ambiente. Esto es logrado externamente por medio de un convertidor ADC y un adecuado sensor de temperatura.

La presentacion de la temperatura, también requiere el uso de un pequeño programa, el cual se muestra en la figura 3.18. Este programa se encarga de leer el valor presente en el convertidor ADC para luego convertirlo en un valor decimal. Estos valores a su vez son enviados al área de video, en donde son utilizados por la rutina principal, para mostrar la temperatura.

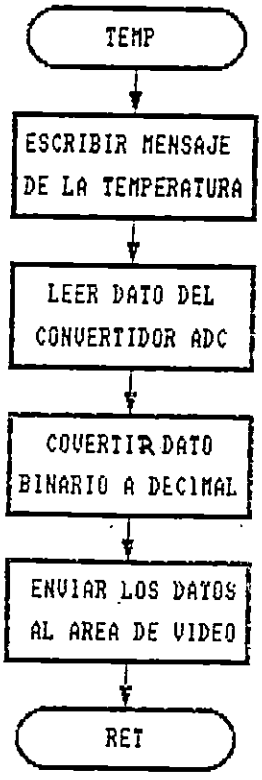


figura 3.18 Flujograma empleado para mostrar temperatura.

3.4 GENERADOR DE RELOJ DE TIEMPO REAL

Convertir este sistema en un reloj de tiempo real requiere el uso de un esquema de interrupción, el cual puede ser muy simple, tal como se muestra en la figura 3.19. Este sistema de interrupción está formado por un oscilador que produce una frecuencia de 1Hz, el cual es conectado a una entrada de interrupción del microprocesador (NMI).

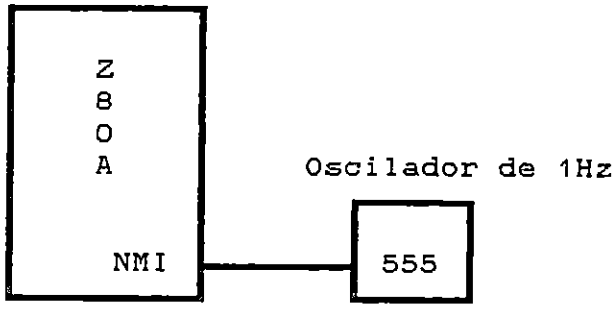


Figura 3.19 Entrada de interrupción manejada por un oscilador de 1 Hz, para crear un reloj de tiempo real.

El microprocesador Z80-A cuenta con 2 entradas de interrupción: La entrada de interrupción mascarable (INT) y la entrada de interrupción no mascarable (NMI). La diferencia principal entre estas dos entradas de interrupción es la capacidad que tiene la CPU para ignorarla o no.

La entrada de interrupción enmascarable (INT) puede ser ignorada por la CPU mediante software, mientras que la entrada de interrupción no enmascarable (NMI) no puede ser ignorada bajo ninguna circunstancia.

Esta cualidad de la entrada de interrupción no mascarable es muy importante en el diseño del reloj de tiempo real, ya que la actualización del tiempo debe realizarse a iguales intervalos de tiempo. Debido a ésto la rutina de actualización de tiempo es manejado por una entrada de interrupcion NMI.

Debe tenerse en cuenta que la solicitud de una interrupción NMI, ejecuta una rutina de servicio, la cual debe comenzar en la posición 66H de memoria.

3.4.1 SOFTWARE GENERADOR DEL RELOJ

El tiempo de este reloj es almacenado en 4 posiciones de memoria RAM. Estas contienen el tiempo que se programó en la incialización del sistema y el cual se encuentra en el formato BCD. La primera posición contiene los segundos y funciona como un MOD 60, la siguiente posición contiene los minutos y funciona de igual manera. El tercer contador es un MOD 12 en el cual se almacena la hora, en el formato de 12 horas. Debido a éste formato de 12 horas se emplea un cuarto contador, el cual indica si es AM ó PM, por tanto solo tiene 2 condiciones 1 para AM y 2 para PM.

```
TIME:   DB   00     ;Contador de segundos
        DB   00     ;Contador de minutos
        DB   12     ;Contador de horas
        DB   02     ;Indicador de medio-dia
```

El sistema además incluye la opción de almacenar la fecha.

```
FECHA:  DB   01     ;Dia
        DB   01     ;Mes
        DB   94     ;Año
```

3.4.2 RUTINA DE ACTUALIZACION

Puesto que hay 4 contadores que deben ser periodicamente actualizados, es conveniente desarrollar una rutina que pueda manejar a cada uno de ellos. Dicha rutina se muestra en la figura 3.20. A está rutina se le debe proporcionar el módulo de contador a ser actualizado y su localización en memoria. Para nuestro caso el par de registros HL, contiene la dirección del contador y el registro B el módulo del contador a ser incrementado. B le indica a la rutina cuándo limpiar el contador. Está información debe ser proporcionada junto con el programa que lo invoca, para el caso la rutina de servicio de interrupción.

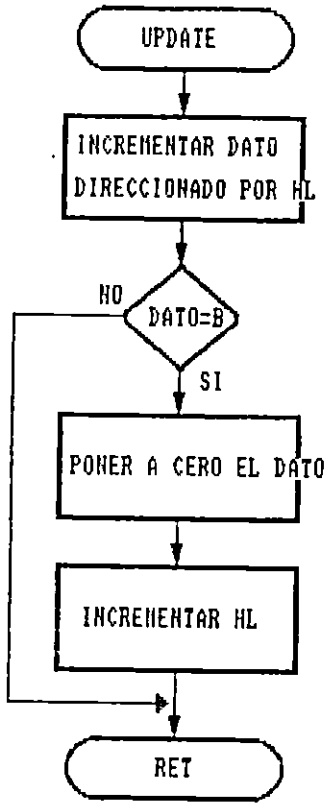


Figura 3.20 Rutina de actualización del tiempo

En la rutina de UPDATE el contador es incrementado en 1. Este valor es comparado con el conteo terminal almacenado en B, si es mayor que B ejecuta un retorno de interrupción, pero si alcanza el valor final, el contador es limpiado y la dirección almacenada en HL es incrementada a la siguiente dirección del contador.

3.4.3 RUTINA DE SERVICIO DE INTERRUPCION

La rutina de servicio de interrupción es mostrada mediante el flujograma de la figura 3.21. Esta debe actualizar el reloj BCD almacenado en RAM.

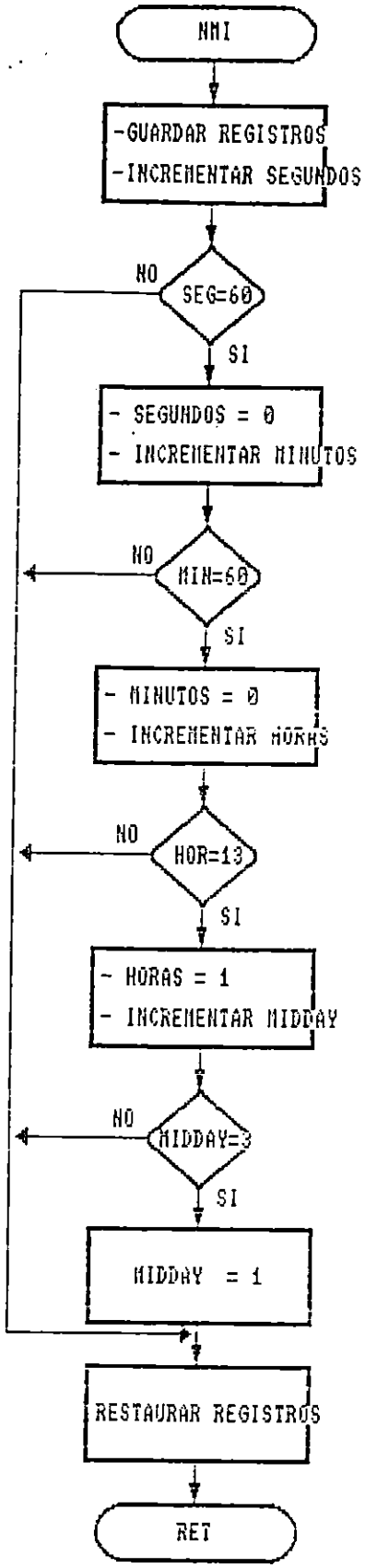


Figura 3.21 Flujoglama de la rutina del servicio de interrupción

Como puede observarse en la figura 3.21 el valor de cada posición de memoria, conteniendo los segundos, minutos y horas del dia actual, se incrementan en 1 en el momento que se excede un valor limite. Este valor corresponde a la máxima cuenta permitida para los segundos, minutos y horas.

Al final del flujograma se maneja una variable llamada MIDDAY ,en la que se almacena la parte del dia en que nos encontramos (1 para AM y 2 para PM). Esta variable puede ser eliminada si el formato del tiempo es de 24 horas.

3.4.4 PROGRAMACION Y VISUALIZACION DE LA HORA

La información presente en las posiciones de memoria llamadas "contadores de tiempo" puede manipularse tanto para la programación como para la lectura. Estas dos funciones son llevadas a cabo mediante las rutinas: "STOREH" y "HORA".

La rutina STOREH se encarga de almacenar los datos que se encuentran el el área de video y que han sido programados mediante el teclado.

Para almacenar éstos datos correctamente debe tenerse en cuenta que los datos en el área de video y los datos en los contadores de tiempo se encuentran en diferentes formatos. Esto quiere decir que 2 posiciones de memoria consecutiva en el área de video corresponden a un solo byte en el área de los contadores de tiempo. La figura 3.22 muestra los dos tipos de formatos usados para la presentación de la hora

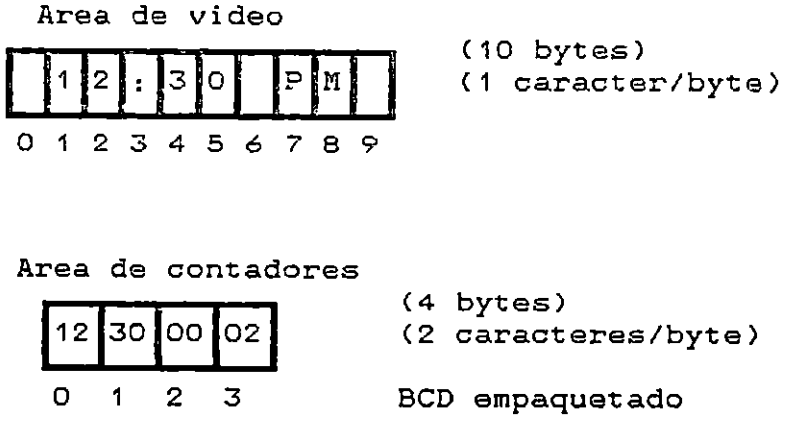


Figura 3.22 Formatos empleados en el manejo de datos de video y contadores

La rutina STOREH se muestra en la figura 3.23. En la cual observamos, que la rutina primeramente empaqueta dos valores del área de video, que corresponden al valor de la hora. A continuación se empaquetan los bytes correspondientes a los minutos y finalmente se almacena el valor de 1 ó 2 en la posición de memoria MIDDAY, el cual depende de si es AM (1) ó PM (2).

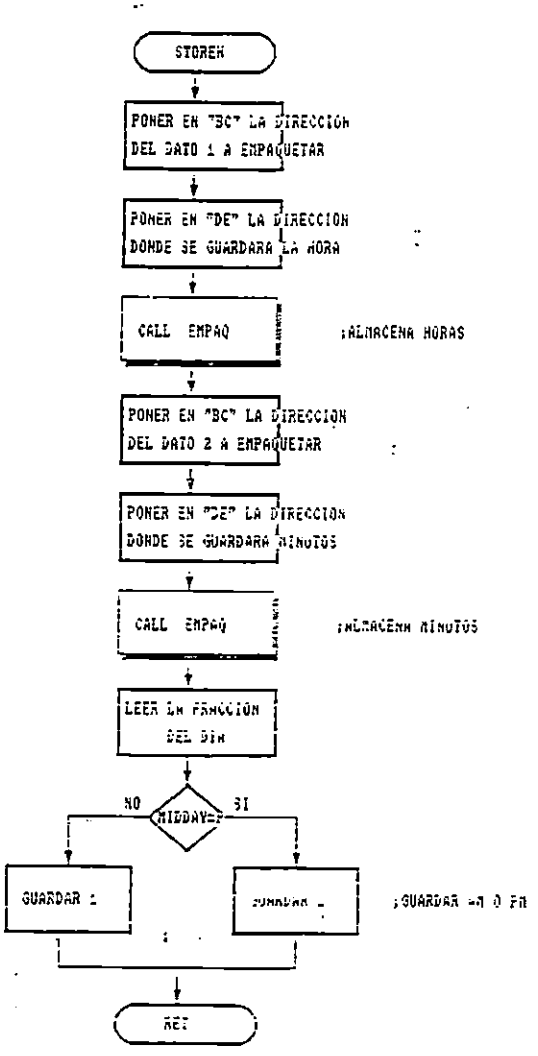


Figura 3.23 Rutina empleada para almacenar la hora.

La rutina que se encarga de empaquetar los datos del área de video se muestra en la figura 3.24. Esta rutina utiliza 2 registros, los cuales indican la dirección del dato fuente y la dirección del dato destino. La rutina se basa en unir dos datos consecutivos apuntados por la dirección fuente mediante una operación OR, luego de esto se almacena el resultado en la dirección destino.

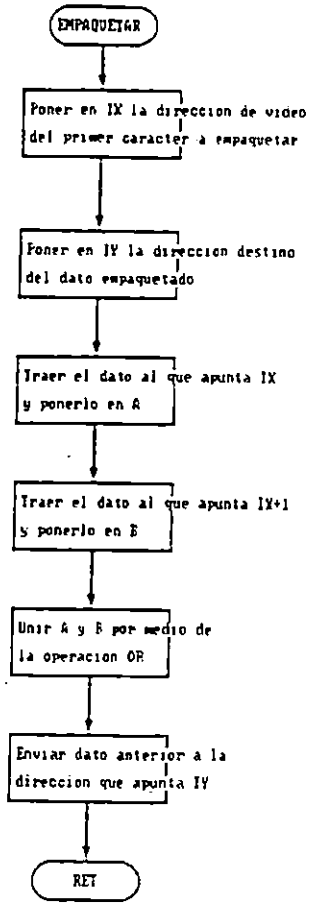


Figura 3.24 Rutina de empaquetado.

El proceso contrario de almacenar la hora está representado por la rutina llamada "HORA". Dicha rutina se muestra en la figura 3.25. Esta rutina se encarga de leer los valores que se encuentran en los "contadores del tiempo" y enviarlos al área de memoria destinada a video. Para esta función se utilizan los registros índices IX e IY. IX apunta a la dirección de memoria de donde se leerá el valor del tiempo. Primeramente este valor se inicializa con la dirección de segundos. IY apunta a la dirección de video donde se presentará el valor leído .

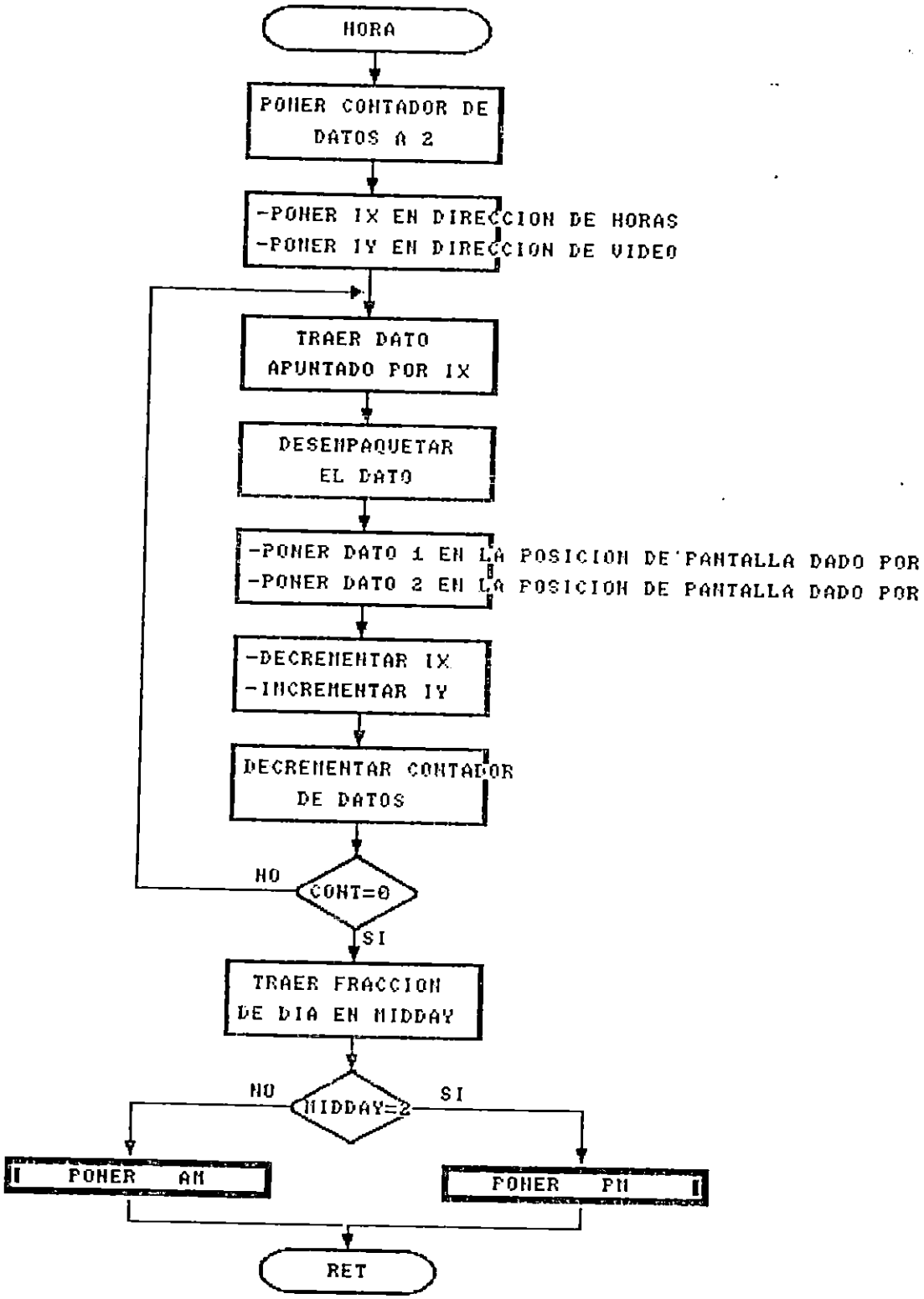


Figura 3.25 Rutina utilizada para presentar la hora

Para realizar esta transferencia de datos, de una posición de memoria (contadores de tiempo) a otra (área de video) es necesario realizar un desempaquetado de datos, ésto se debe a que el formato de dichos datos son diferentes como se mencionó en la rutina STOREH.

Con el desempaquetado de datos, obtenemos 2 valores, los cuales son enviados a posiciones consecutivas, hasta el área de video. Este proceso es repetido 2 veces, mostrando así los valores de hora y minutos. Como parte final de esta rutina el valor que se encuentra en la variable MIDDAY determina si se escribe AM ó PM en el área de video

3.4.5 PROGRAMACION Y VISUALIZACION DE LA FECHA

De igual manera que la hora la fecha puede programarse ó leerse, mediante el uso de las rutinas "STOREF" y "FECHA".

Con la rutina STOREF almacenamos los datos que representan la fecha y que se encuentran en el área de video. Estos datos fueron programados mediante teclado, en el formato que se muestra en la figura 3.26.

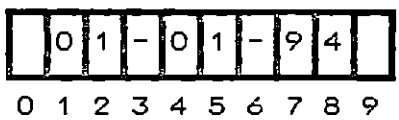


Figura 3.26 Formato empleado para almacenar la fecha

Para almacenar éstos datos la rutina STOREF empaqueta en pares, los datos, primeramente los dos bytes correspondientes al dia, seguidamente los bytes equivalentes al mes y por último los bytes que representan el año; ésta rutina se muestra en la figura 3.27.

Para lograr empaquetar éstos datos se utiliza la rutina EMPAQ que fue utilizada en la presentación de la hora

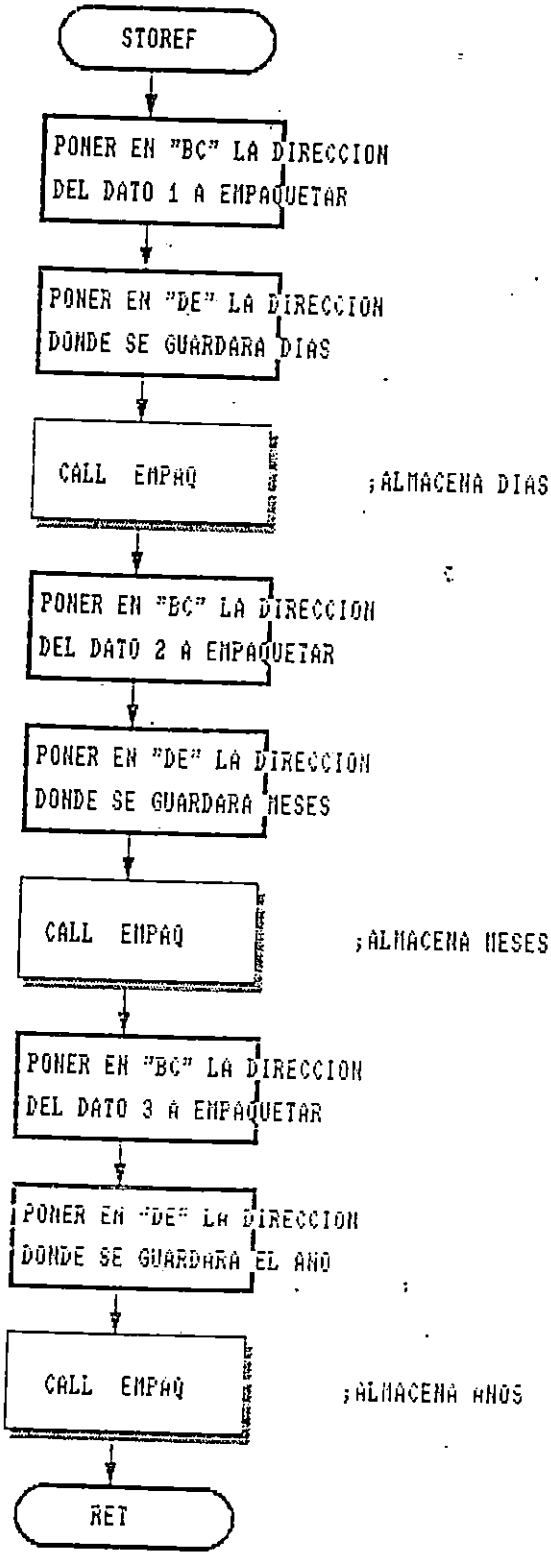


Figura 3.27 Rutina utilizada para almacenar la fecha

Como se dijo anteriormente, a la rutina de empaquetado debe suministrarsele la dirección fuente y la dirección destino de los datos. La figura 3.27 muestra que tres operaciones de empaquetado son necesarias para almacenar la fecha: Primero el almacenamiento del día, luego el almacenamiento del mes y por último el almacenamiento del año.

Los valores programados para la fecha pueden ser visualizados mediante la rutina llamada "FECHA",como se muestra en la figura 3.28. Esta rutina utiliza el registro indice IX para apuntar a una dirección que corresponde al dia, mes o año que hemos almacenado mediante la rutina STOREF. También se utiliza el registro indice IY el cual apunta a la dirección de video donde el dato correspondiente de la fecha será mostrado.

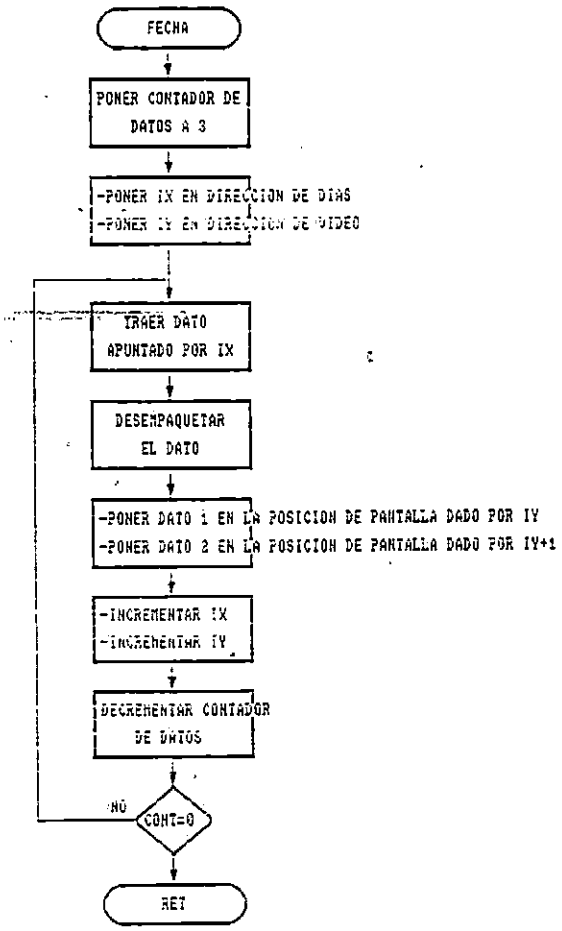


Figura 3.28 Rutina utilizada para mostrar la fecha

Con el registro IX inicializado con la dirección que apunta al valor del dia e IY apuntando a la dirección de video en donde se mostrará la información, se procede a desempaquetar los datos para luego enviar los 2 nuevos datos a posiciones de memoria consecutivas . Luego se incrementa el valor de IX , para apuntar a la dirección del mes con lo que volvemos a repetir el proceso de desempaquetado y de enviar los 2 nuevos nuevos datos a video del mes; se repite el proceso para enviar el valor del año, terminando así ésta rutina

3.5 ENTRADA DE DATOS AL SISTEMA

El provisionar un medio que permita tener control sobre la operación de un sistema es una consideración inevitable a la hora de hacer un diseño.Esto es indiscutiblemente cierto pues cualquier aparato sin un medio de control,no tendria utilidad reconocible.

El control de nuestro sistema de información será ejecutado desde un teclado muy sencillo,desde el cual,el operador podrá suministrar ordenes y datos al sistema de procesamiento para que se realizen funciones deseadas.

El teclado esta compuesto por las letras del alfabeto en mayusculas ,por los numeros decimales del 0 al 9 ,y por los signos de puntuación;además el teclado incluye una tecla que habilita una segunda función con la cual se tiene acceso al alfabeto en letras minúsculas y a otros símbolos importantes.También se pueden definir comandos exclusivos tanto para el proceso de programación como para la ejecución de funciones en el visualizador .

Nuestro diseño consta de 54 teclas en forma de matriz,compuesta por 4 filas y 14 columnas ,donde cada intersección corresponde a un caracter o a una función.Cada caracter puede ser seleccionado oprimiendo una especie de pulsador que pone eléctricamente en contacto la fila con la columna correspondiente a cada caracter seleccionado.
Para una mejor comprensión observese la fig.3.29 donde se muestra el diagrama eléctrico que genera el código de tecla.

El teclado sirve un código binario que corresponde a la posición matricial ocupada por la tecla seleccionada;dicho código es generado por un contador de 6 bits que en un conteo normal total sondea todas las teclas disponibles ,este sondeo lo realiza con el auxilio de un decodificador de 2 a 4 líneas que barre las 4 filas de la matriz del téclado;y un multiplexor de 16 entradas que sondea las columnas corespondientes a la matriz del teclado.

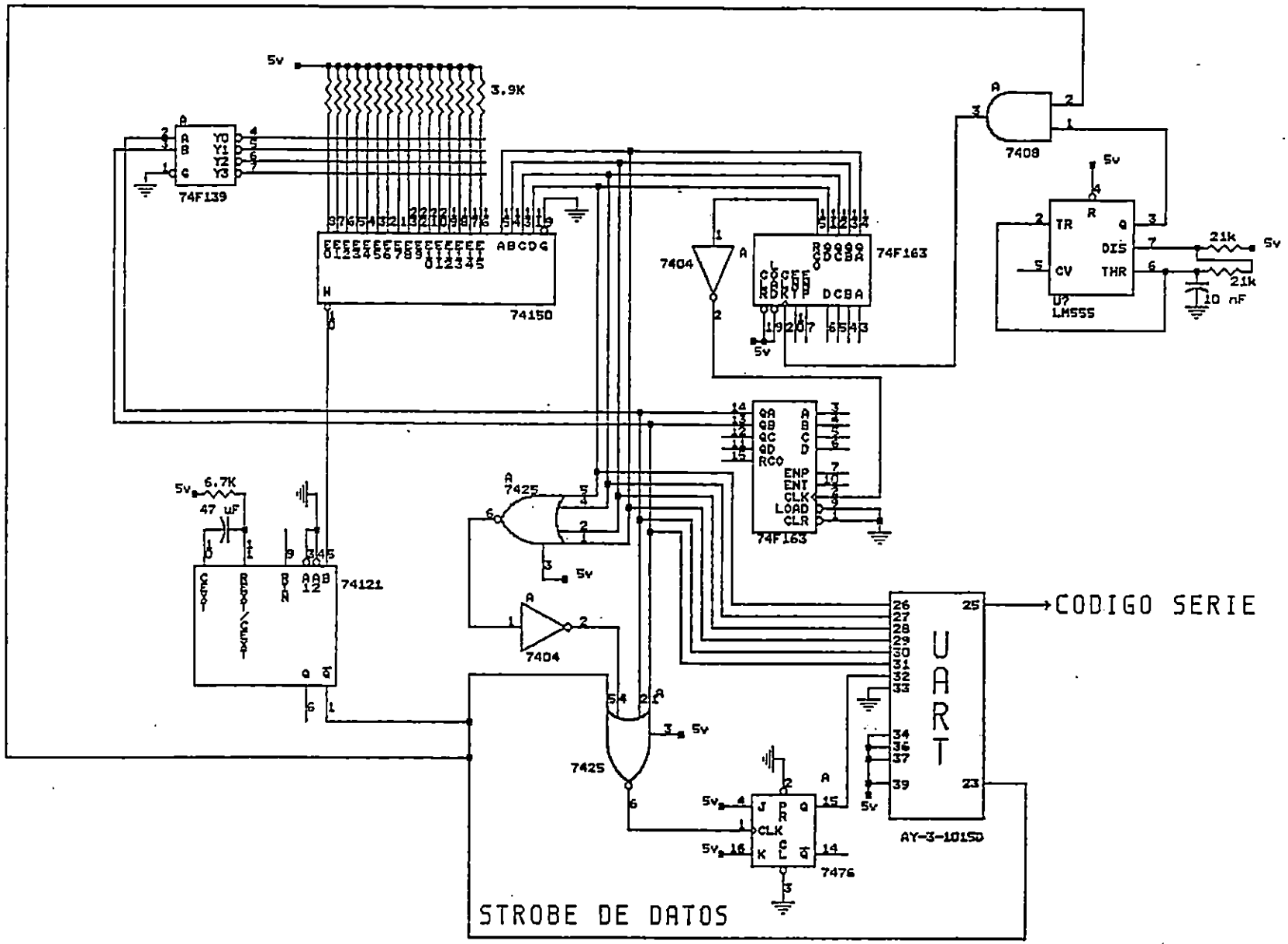


Figura 3.29 CIRCUITO CODIFICADOR DE TECLADO

Los 4 bits menos significativos del contador scan ,sirven como entradas de selección al multiplexor y,las 2 líneas de bits más significativos son conectadas a las entradas de selección del decodificador excudriñador de filas .

Cuando se activa el teclado para introducir algun mensaje ,el contador empieza su conteo y pone a funcionar al multiplexor y al decodificador para que detecten si hay pulsación de teclas.

Si asumimos que el contador esta límpio,podemos decir que el conteo empieza en 000000,lo cual hace que el decoder active la línea horizontal superior de la matriz ;al mismo tiempo que el multiplexor transmite hacia la salida el dato correspondiente a la columna que esta más a la izquierda .Mientras el decoder mantiene activada la línea horizontal superior ,el mux va transmitiendo uno a uno los datos encontrados en cada una de sus entradas(columnas de la matriz).Cuando el MUX ha transmitido hacia su salida todos los datos presentes en sus entradas,el contador hace que en el siguiente conteo el decodificador active la segunda linea horizontal,al mismo tiempo que el MUX recicla a transmitir el dato encontrado en la columna de más a la izquierda y así sucesivamente se excudriñan todas las posiciones del teclado.Como no se ha pulsado ninguna tecla,el MUX mantiene en su salida un nivel bajo.

Un detalle que se debe mencionar es que este barrido se hace a una velocidad tal que se pueda vencer la agilidad de nuestros dedos y así pueda detectarse cuando se ha pulsado una tecla.

Cuando se pulsa una tecla,el MUX que generalmente tenia en todas sus entradas el mismo estado ,detecta un estado diferente en la columna correspondiente a la tecla pulsada,transmitiendo éste dato a la salida generando un pulso que activa un ONE SHOT que a su vez genera otro pulso que detiene por un tiempo al contador excudriñador en el conteo correspondiente a la tecla pulsada.Este conteo en el que se detuvo el contador corresponderia al código que se genera cuando se pulsa una tecla .

Durante el tiempo que dura el pulso del ONE SHOT, el código del caracter es leído.Cuando el pulso del ONE SHOT termina ,el contador prosigue su conteo excudriñador y el teclado está lísto para recibir la pulsación de la siguiente tecla.

3.5.1 SISTEMA TRANSMISOR DE DATOS

El esquema que aquí utilizamos para transmitir datos(Observe figura 3.31),requiere la aplicación de 2 unidades UART ;una en cada extremo de la línea .La transmisión que hemos diseñado desplaza códigos a distancias de entre 10 a 20 metros a un hilo de par trenzado unico pudiendo alcanzar distancias mayores dependiendo de la velocidad de transmisión,de la calidad de conductor de la línea y otros.

Se necesita enviar códigos desde el teclado hasta la CPU del sistema .

La transmisión de datos real sigue el formato en serie asíncrono ilustrado en la fig. 3.30.Cuando no se está transmitiendo ,la línea de datos se situa en una marca de trabajo (nivel 1) a la espera de una señal "strobe" por pulsación de tecla.Esta ultima señal es un impulso positivo que indica que se ha pulsado una tecla y que un código binario de esa tecla está disponible para ser transmitido.La señal "strobe",que esta unida a la strobe de datos del UART emisor ,hace que los datos en binario se carguen en una memoria intermedia (de la uart) en paralelo e inicia el ciclo de transmisión.

La salida en serie del UART emisor hará entonces una transición desde "1" a "0" .Este bit de marca indica el principio de una palabra transmitida en serie .Despues del bit de comienzo siguen 7 bits de datos ,teniendo cada bit de datos la longitud de un periodo de reloj.A la conclusión de los bits de datos ,un bit de paridad y un bit de parada se hacen salir por el UART emisor para indicar el final de transmisión. Si se pulsa otra tecla el proceso se repite a si mismo.

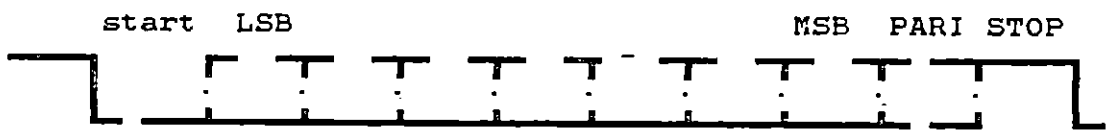


Figura 3.30 Esquema de transmisión de datos

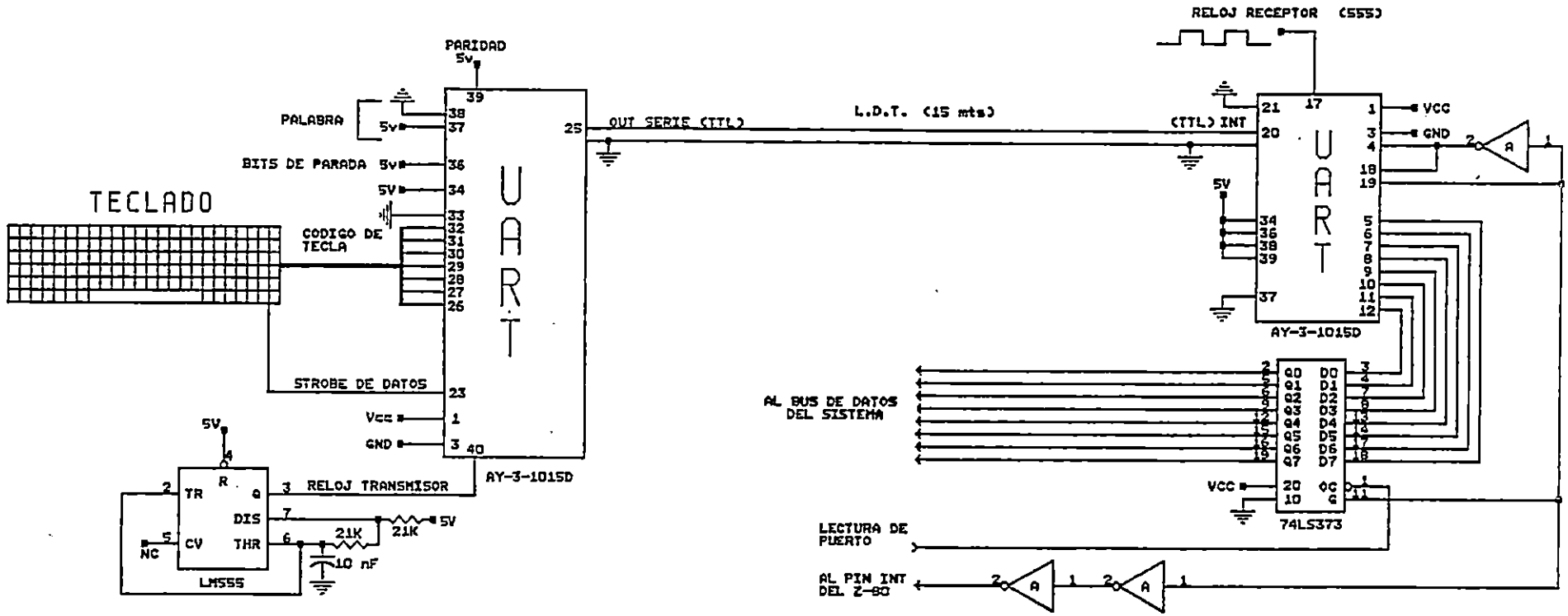


Figura 3.31 CIRCUITO PARA TRANSMISION DE DATOS

En el extremo de recepción el UART esta controlando continuamente la línea de entrada en serie para detectar el bit de comienzo.A su ocurrencia ,los 7 bits de datos se deslizan a un registro y se comprueba la paridad .A la terminación de la introducción en serie ,una salida que significa datos disponibles se establece por el UART (pin 19 del AY-3-1015D) y esta señal es empleada como impulso de interrupción a la cpu Z80-A .El UART receptor no procesará entradas en serie adicionales a no ser que se acuse recibo del indicador de datos disponibles y se seleccione por strobe la línea de reset de datos disponibles .La transmisión real puede incluir o excluir paridad,tener 1 o 2 bits de parada y los datos pueden ser palabras de 5 a 8 bits.Estas opciones son seleccionables por terminales.

3.5.2 SOFTWARE UTILIZADO PARA LA ENTRADA DE DATOS (LECTURA DE TECLADO)

El manejar un teclado es tarea muy común en los sistemas basados en microprocesadores.En la actualidad existen 2 métodos que manejan muy eficientemente un teclado.El primero de ellos involucra el sensoreo constante del teclado por parte de la CPU ,mediante un software dedicado a éste propósito. El segundo método y que además es el utilizado en nuestro sistema es el que se realiza mediante interrupciones,en dónde el teclado además de proporcionar un código de tecla ,proporciona una señal de dato listo que le indica a la CPU cuando se debe efectuar una operación de lectura .Esto es muy ventajoso por que se libera al microprocesador de estar sensando el teclado y así se pueda utilizar ese tiempo para realizar tareas más importantes.

Debido a que el proceso de lectura del teclado se lleva a cabo mediante el proceso de interrupciones ,es necesario diseñar una rutina que se ejecute cada vez que una tecla sea presionada .Debe tenerse en cuenta que al utilizar la línea de interrupción INT del Z80-A,el servicio de interrupción debe residir en la dirección de memoria 38H,esto se debe a que el microprocesador esta configurado en el modo 1 para el manejo de interupciones.

El Z80-A posee además 2 modos adicionales para el manejo de las interrupcines enmascarables ,éstos modos son denominados modo 0 y modo 1 de interrupciones;pero debido a la facilidad del manejo se utilizó el modo 1.

En la figura 3.32 se muestra el flujograma de la rutina de servicio de interrupción INT que se ejecuta cada vez que se presiona una tecla.

Uno de los pasos que se efectua primeramente en ésta rutina es guardar el estado actual de la CPU antes de servir a la interrupción,pues de ésta manera al terminar el servicio de la interrupción puede regresarse al programa principal sin ningun problema.

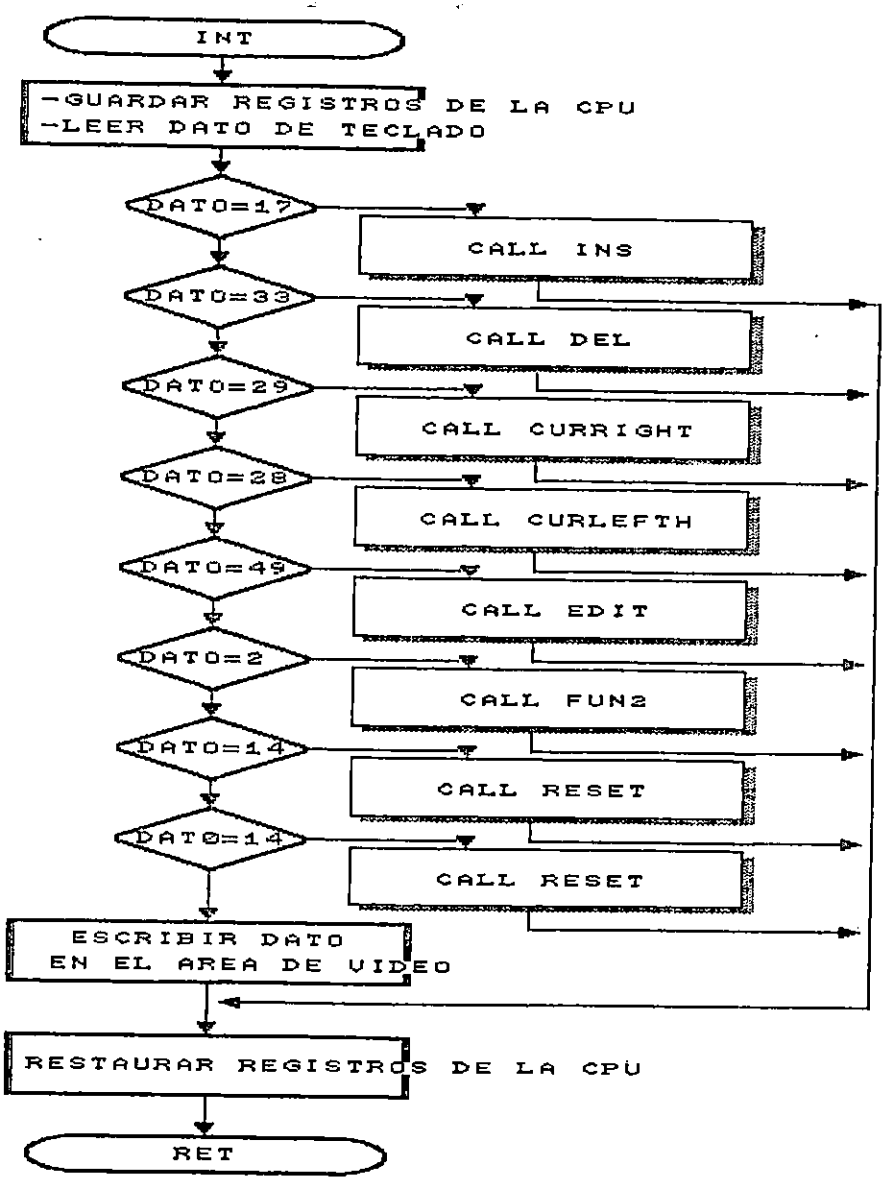


Figura 3.32 Rutina de servicio de interrupción

Como puede observarse en la figura 3.32 ,la rutina de servicio de interrupción es capaz de detectar si la tecla que se ha pulsado es una tecla de función;de ser ésto cierto se ejecuta la rutina correspondiente a dicha función,en caso contrario al tecla presionada sirve como dato y su valor es almacenado en RAM.

Nuestro sistema cuenta con funciones que hecen al usuario sentir que está manejando un pequeño editor de texto (Insert,Delete,Backspace,CurLeft,CurRigth,Edit,Reset,etc) Estas rutinas hacen fácil la tarea de edición y corrección de la información que mostrará el presentador.El software empleado para realizar éstas tareas puede consultarse en los anexos de éste documento.

Ya que el sistema almacena en RAM la información que se programa,ésta permanece intacta mientras el sistema no se desenergize o mientras no se de un reset desde teclado.El contar con una tecla EDIT resulta ventajoso cuando se desea hacer correcciones a la información preprogramada;con ésta función podemos modificar cualquier parte de la información que maneja el visualizador (hora ,fecha , y mensaje),simplemente debe moverse el cursor a la posición del caracter o caracteres que se deseen cambiar y luego teclear el nuevo o los nuevos caractacteres que sustituirán a los anteriores;finalmente teclear ENTER para al macenar las modificaciones .

Debe tenerse en cuenta que si no se desea cambiar alguna de las partes de la información (Hora,Fecha,y Mensaje),basta teclear ENTER para no modificar la información.

La interacción funcional de las diferentes partes que constituyen el sistema y que han sido desarrolldas a lo largo de éste capítulo, se esquematiza en el diagrama de la figura 3.33.Ahí puede observarse la disposición final de los componentes del sistema de presentación de información que hemos diseñado

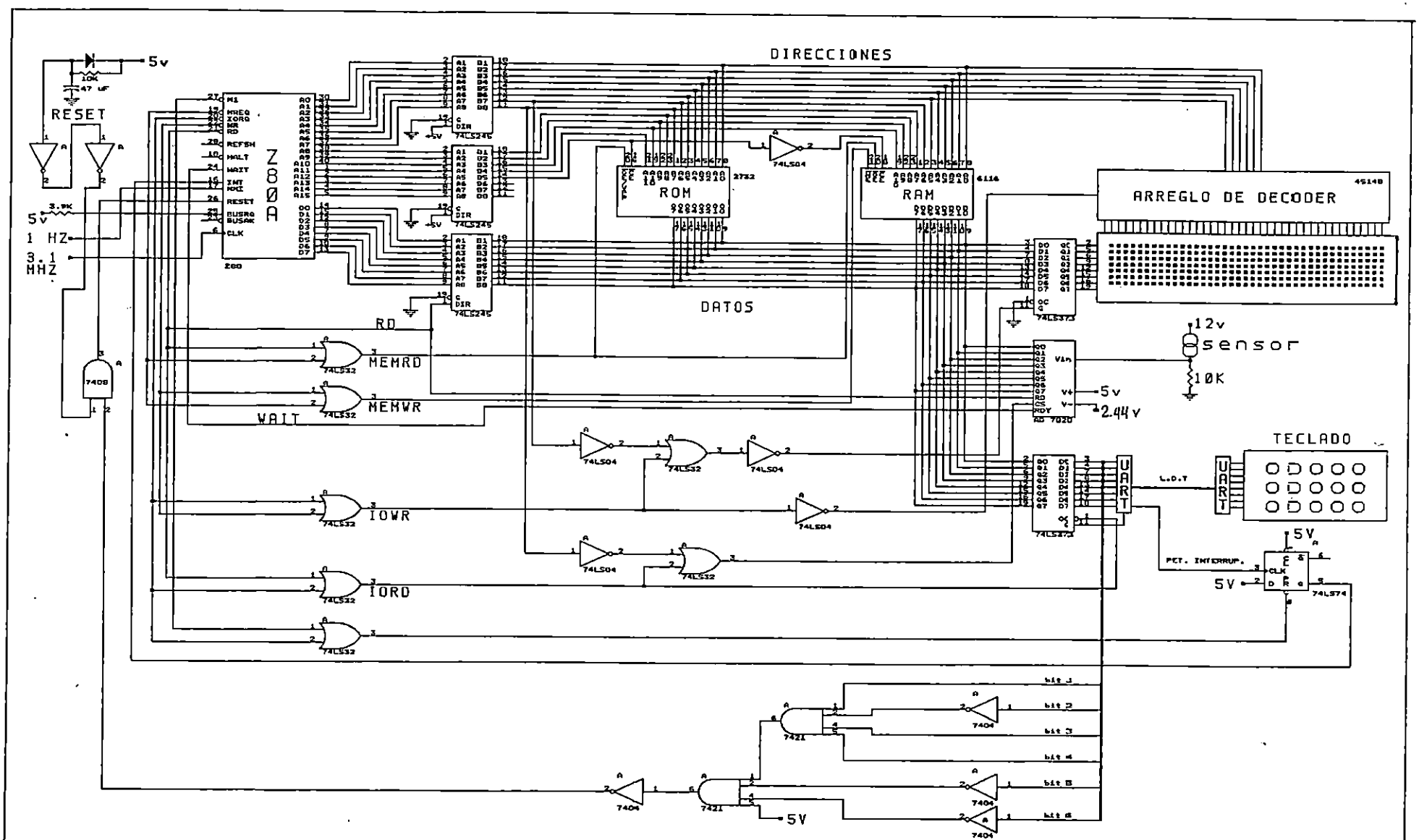


Figura 3.33 DIAGRAMA TOTAL DEL SISTEMA DE PRESENTACION

3.6 MANUAL DE USUARIO

Introduccion

Esta guía tiene como objetivo principal ,ofrecer al usuario una pequeña orientación de lo que es el sistema e indicarle los pasos que debe seguir para su programación .No se necesita ningun conocimiento especial para la utilización del equipo que aquí hemos diseñado.

3.6.1 Lo que ofrece el sistema

Este es un sistema que presenta información,esta diseñado para poder mostrar la hora actual,la fecha,la temperatura ambiente,y un mensaje a gusto del usuario.

La información es presentada en la pantalla por el método de multiplexación en el tiempo es decir,una a continuación de la otra.

A excepción de la temperatura, el resto de la información puede ser alterada por el usuario;la temperatura es un parámetro que solo depende de las condiciones del medio y así el sistema mismo modifica su presentación de forma automática.

Para la programación del sistema se dispone de un teclado donde el usuario puede seleccionar : Letras mayusculas,signos de puntuación,numeros decimales,símbolos importantes ,y teclas de función.

3.6.2 Explicación del teclado

La disposición de teclas del teclado se muestra en la figura 3.34.

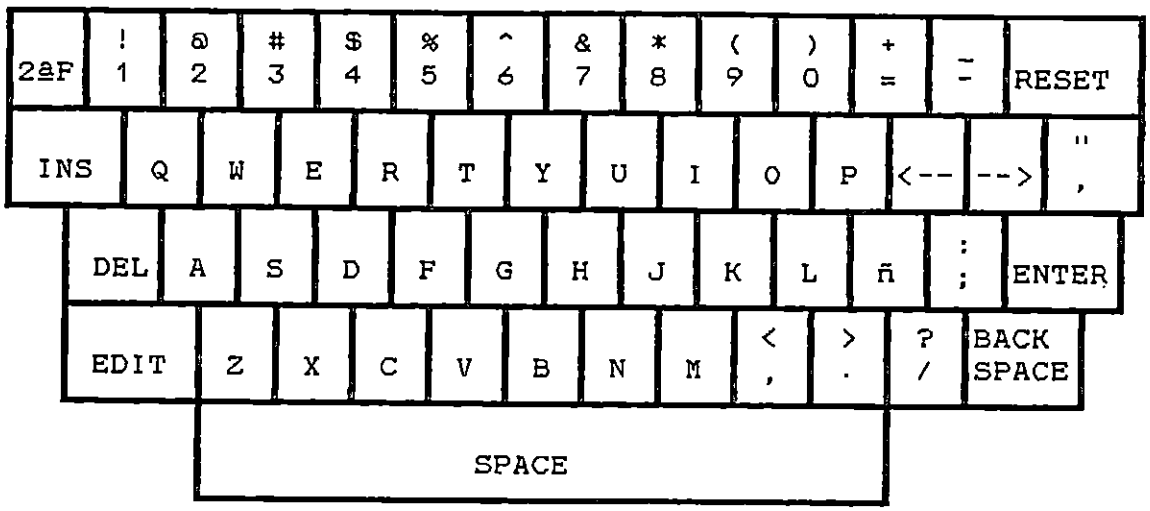


Figura 3.34 Teclado del sistema.

A excepción de las teclas de función,el resto de teclas tienen una doble función .La segunda función es obtenida presionando la tecla marcada con " 2ªF " ;cuando esto ocurra encenderá un indicador LED dispuesto en el teclado,el cual nos dará la certeza de que estamos en segunda función.Para volver a la primera función ,presione nuevamente la misma tecla.

En la opción de primera función son seleccionables :Los numeros decimales,las letras mayusculas ,y los símbolos marcados en la parte baja de cada tecla.

En segunda función son seleccionables :Los símbolos marcados en la parte alta de cada tecla,y las letras en minuscula.

Para las teclas de función como: RESET,CURSORES,ENTER,etc,es indiferente la función en la que se este.

3.6.3 Programación del sistema

Para la programación del sistema se ha elaborado un programa muy interactivo con el usuario ,dicho programa le indica a éste lo que debe ir haciendo y le muestra además el formato en el que debe introducir los datos.

Secuencia de programación

Al encender el sistema ,este corre un mensaje en pantalla que dice: PROGRAMAR HORA ,y seguidamente muestra el formato de como se debe introducir dicho dato.

El formato de introducción de la HORA es el siguiente:

XX:YY ZZ

Donde:

XX: Horas
YY: Minutos
ZZ: AM o PM.

Cuando ya se ha programado la hora presione la tecla ENTER y esto hará que se guarde lo programado,y aparecerá un nuevo mensaje en pantalla que dirá: PROGRAMAR FECHA,y se muestra el formato de como introducirla.

El formato de introducción de la fecha es el siguiente:

XX-YY-ZZ

Donde:

XX:Día del mes
YY:Mes del año
ZZ:Año.

Un ejemplo de como se debe introducir la fecha (12 de mayo de 1994) se muestra a continuación.

12-05-94

Cuando se ha programado la fecha presione la tecla ENTER y esto hará que se muestre en pantalla un nuevo mensaje que dice: PROGRAMAR MENSAJE .Cuando este mensaje halla desaparecido ,el cursor se posicionará a la izquierda de la pantalla y,esto indicará al usuario que puede empezar a digitar su mensaje.El tamaño del mensaje puede ser de hasta 1536 caracteres (600H).

Cuando termine de programar el mensaje presione la tecla ENTER y esto hará que la parte del mensaje que aún se visualiza en pantalla corra hacia la izquierda y cuando desaparezca ,se activará la rutina de presentación de toda la información programada.

La secuencia de presentación en pantalla, de la información programada es la siguiente: Primeramente aparece la hora con el efecto de caida vertical y

empaquetamiento,posteriormente corre hacia la izquierda hasta desaparecer; el segundo dato que aparece es la fecha y ésto se hace con el efecto de corrimiento vertical pausado; la tercera información que aparece es la lectura de la temperatura ambiente y ésta se muestra con el efecto de aparecimiento y desaparecimiento por patrones (filas); por ultimo aparecerá el mensaje que usted digitó,presentandose éste con el efecto de corrimiento horizontal.Al desaparecer el ultimo caracter del mensaje ,la secuencia de presentación se repite.

Si usted quiere modificar algun mensaje aún cuando el sistema este haciendo su presentación ,presione la tecla EDIT y ésto detendrá la presentación y en pantalla y aparecerá el mensaje PROGRAMAR HORA ,si usted no desea cambiar la hora solo presione ENTER y hágalo de nuevo hasta que aparezca la información que usted desea modificar.Presine ENTER para volver a la rutina de presentación normal

Si usted lo desea, puede inicializar el sistema desde el teclado presionando la tecla RESET,esto le permite a usted reiniciar la programación .

Para programar o reprogramar su mensaje usted tiene acceso a las siguientes teclas de función:

--->	:Mueve el cursor sobre las letras hacia la derecha
<---	:Mueve el cursor sobre letras hacia la izquierda
BACK SPACE	:Borrar la ultima letra escrita
2ªF	:Selección de segunda función
EDIT	:Detiene la presentación para reprogramar
RESET	:Inicializar el sistema desde el teclado
INS	:Insertar caracteres en los mensajes
DEL	:Borrar caracteres de cualquier parte del mensaje

CONCLUSIONES DEL CAPITULO III

El diseño de un sistema digital utilizando tecnología LSI representa una tarea muy factible desde el punto de vista del hardware si lo comparamos con otras técnicas, no obstante se requiere de un conocimiento más profundo sobre la teoría de procesamiento y control de la información digital, al mismo tiempo que obliga al diseñador a introducirse en el ambiente de los lenguajes de programación de bajo nivel.

REFERENCIAS BIBLIOGRAFICAS

1- Ciarcia Steve. Construya Una Microcomputadora Basada En El Z-80. Editorial Byte Books/MacGraw Hill, 1986.

2- Nichols C. ,Nichols Elizabeth A. ,Rony Peter R. El Microprocesador Z-80. Programacion e Interfaces Publicaciones Alfaomega-Marcombo.

3- Mandado E. ,Tassis E. Diseño De Sistemas Digitales Con Microprocesadores.Publicaciones Marcombo,Boixareu editores.

4- Tokheim L. Fundamentos De Los Microprocesadores.Serie Schaum/Macgraw Hill, 1982.

CONCLUSIONES GENERALES Y RECOMENDACIONES

CONCLUSIONES

La conclusión más clara a la que hemos llegado al término de éste trabajo es que para diseñar sistemas digitales de gran versatilidad y gran capacidad funcional, es necesario utilizar componentes de gran poder, es imposible lograr grandes metas utilizando solamente circuitos SSI y MSI;especificamente nos referimos a que es necesario la utilización de microprocesadores en el diseño de éstos sistemas. La capacidad y poder de los microprocesadores se pone de manifiesto al comparar un pequeño set de instrucciones formando un programa, con lo complicado que resulta obtener determinada respuesta de un sistema sin la utilización del recurso moderno de la programación.

RECOMENDACIONES

La programación de sistemas basados en microprocesadores, tal como el que hemos diseñado aquí, resulta ser muy tediosa sino se dispone de una técnica de programación adecuada. El hecho de manejar un lenguaje de bajo nivel, tal como un ensamblador, hace que el programador cometa muchos errores.

Las pruebas de los programas utilizados son efectudas en memorias ROM, resultando muy complicado y no muy aconsejable pues tienen que estarse borrando y reprogramando constantemente .A manera de recomendación presentamos la idea de diseñar una interface entre una PC y el sistema ,para facilitar la programación. Esta interface debe ser capaz de traducir un lenguaje de alto nivel(BASIC, PASCAL ó LENGUAJE C) a un lenguaje de bajo nivel (ensablador),mediante el uso de software y hardware, y de ésta manera hacer el trabajo más eficientemente.

El diseño de software en los sistemas programables requiere el uso de microprocesadores muy flexibles en cuanto a su set de instrucciones. El microprocesador Z80-A a pesar del gran número de registros y su set de instrucciones o mnemónicos, resulta ser un poco rígido en el manejo de las instrucciones. Un microprocesador que resulta ser muy eficiente en tareas que involucran mucho software es el 8088/8086. Si se desea realizar trabajos de la misma índole que el que aquí presentamos, o crear nuevos proyectos relacionados a éste, se recomienda utilizar éste microprocesador.

ANEXOS

ANEXO A

HOJAS DE DATOS DE COMPONENTES

CMOS STATIC RAMS 16K (2K x 8 BIT)

IDT6116S
IDT6116L

MILITARY, INDUSTRIAL, AND COMMERCIAL TEMPERATURE RANGES **SEPTEMBER 1984**

FEATURES:

- High-speed address/chip select access time
  - Military and Industrial
    - 55/50, 70/65,90/90, 120/120, 150/150 ns (max.)
  - Commercial
    - 55/50, 70/70, 90/90, 120/120, 150/150 ns (max.)
- Low-power operation
  - —IDT6116S
    - Active: 180mW (typ.)
    - Standby: 100μW (typ.)
  - —IDT6116L
    - Active: 160mW (typ.)
    - Standby: 20μW (typ.)
- Battery backup operation — 2V data retention voltage
- Produced with advanced CEMOS™ II high-performance technology
- CEMOS II process virtually eliminates alpha particle soft-error rates (with no organic die coatings)
- Single 5V (±10%) power supply
- Input and output directly TTL-compatible
- Three-state output
- Static operation: no clocks or refresh required
- Standard 24-pin DIP, 24-pin THINDIP, 28-pin and 32-pin LCC, or 24-Lead Flatpack
- Pin compatible with standard 16K static RAM and EPROM
- Military product 100% screened to MIL-STD-883, Class B

DESCRIPTION:

The IDT6116 is a 16,384-bit high-speed static RAM organized as 2K x 8. It is fabricated using IDT's high-performance, high-reliability technology — CEMOS™ II. This state-of-the-art technology, combined with innovative circuit design techniques, provides a cost-effective alternative to bipolar and fast NMOS memories.

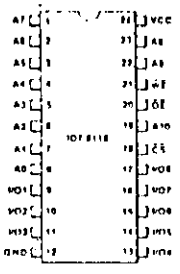
Address access times at 55 ns with chip select times as fast as 50 ns are available with maximum power consumption of only 550mW. The circuit also offers a reduced power standby mode. When $\overline{CS}$ goes high, the circuit will automatically go to, and remain in, a standby power mode as long as $\overline{CS}$ remains high. In the standby mode, the low power device consumes less than 20μW typically. This capability provides significant system level power and cooling savings. Both versions also offer a battery backup data retention capability where the circuit typically consumes only 1μW to 4μW operating off of a 2V battery.

All inputs and outputs of the IDT6116 are TTL-compatible and operation is from a single 5V supply, simplifying system designs. Fully static asynchronous circuitry is used, requiring no clocks or refreshing for operation, providing equal access and cycle times for ease of use.

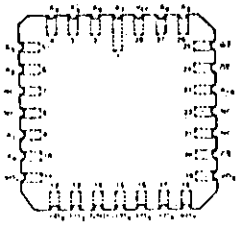
The IDT6116 is packaged in either a 24-pin, 600 and 300 mil-DIP or 32-pin and 28-pin leadless chip carriers, providing high board-level packing densities.

The IDT6116 Military RAM is 100% processed in compliance to the test methods of MIL-STD-883, Method 5004, making it ideally suited to military temperature applications demanding the highest level of performance and reliability.

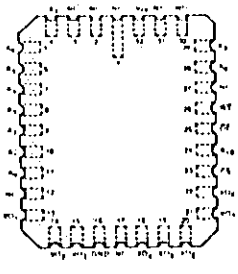
PIN CONFIGURATIONS



24 PIN SIDEBRAZE
TOP VIEW

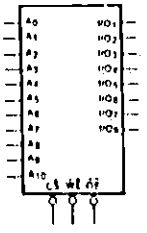


28 PIN LCC
TOP VIEW



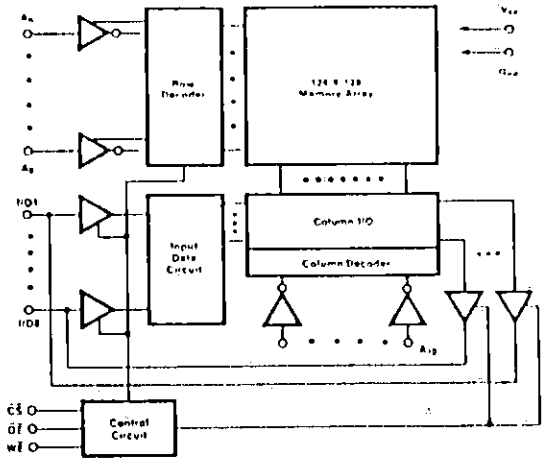
32 PIN LCC
TOP VIEW

LOGIC SYMBOL



A_0-A_{10}	ADDRESS	$\overline{WE}$	WRITE ENABLE
I/O1-I/O8	DATA INPUT/OUTPUT	$\overline{OE}$	OUTPUT ENABLE
$\overline{CS}$	CHIP SELECT	GND	GROUND
V_{CC}	POWER		

FUNCTIONAL BLOCK DIAGRAM



CEMOS is a trademark of Integrated Device Technology, Inc.

Integrated Device Technology

MEMORY

Integrated Device Technology, Inc.
3236 Scott Blvd., Santa Clara, CA 95051
Telephone: (408) 727-6116 • TWX 910-338-2070

P2732A
32K (4K x 8) PRODUCTION EPROM

- 200 ns (P2732A-2) Maximum Access Time . . . HMOS*-E Technology
- Compatible with High-Speed 8 MHz iAPX 186 . . . Zero WAIT State
- Two Line Control
- Compatible with 12 MHz 8051 Family
- Industry Standard Pinout . . . JEDEC Approved
- Low Active Current . . . 100 mA Max.
- inteligent Identifier™ Mode
- Fast 20 ms Programming Time
- TTL Compatible

The Intel P2732A is a 5V-only, Electrically Programmable Read-Only Memory in a plastic package. One time programmable, it has been designed for high volume production environments where a programmable memory is required for flexibility. The standard P2732A access time is 250 ns with speed selection (P2732A-2) available at 200 ns. The access time is compatible with high performance microprocessors such as the 8 MHz iAPX 186. In these systems, the P2732A allows the microprocessor to operate without the addition of WAIT states.

The P2732A is ideal for volume production environments where inventory and lead time risks occur for program codes. Inventoried in the unprogrammed state, the P2732A is programmed quickly and efficiently when the need to change code arises. Costs incurred for new ROM masks or obsoleted ROM inventories are avoided. The tight package dimensional controls, inherent non-erasability, and high reliability of the P2732A make it the ideal component for these production applications.

Using Intel's HMOS*-E technology, low power consumption combined with high speed data access are achieved. The maximum P2732A active current is 100 mA, while standby is only 35mA. The standby mode is selected by applying a TTL-high signal to the $\overline{CE}$ input.

PIN NAMES

A_0-A_{11}	ADDRESSES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}/V_{PP}$	OUTPUT ENABLE/V_{PP}
O_0-O_7	OUTPUTS

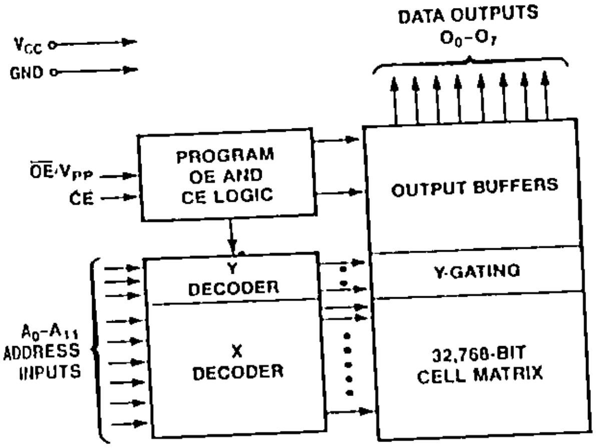


Figure 1. Block Diagram

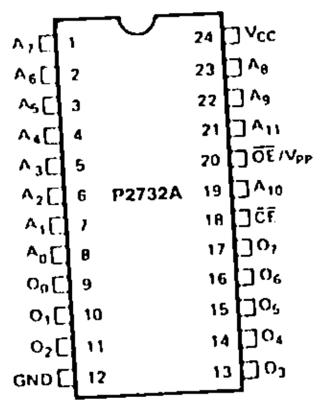


Figure 2. Pin Configuration

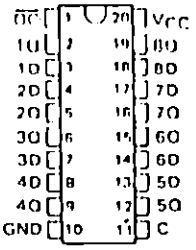
*HMOS is a patented process of Intel Corporation.

OCTOBER 1983
ORDER NUMBER: 230724-002

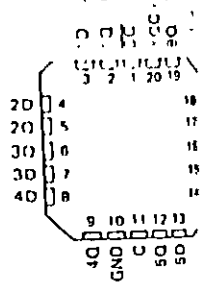
SN54HC373, SN74HC373
OCTAL D-TYPE TRANSPARENT LATCHES WITH 3-STATE OUTPUTS

- 8 High-Current Latches In a Single Package
- High-Current 3-State True Outputs Can Drive up to 15 LSTTL Loads
- Full Parallel Access for Loading
- Package Options: Ceramic Chip Carriers, Surface Mount Small Outline, and Plastic or Ceramic DIPs
- Dependable Texas Instruments Quality and Reliability

SN54HC373 . . . J PACKAGE
SN74HC373 . . . N OR D (SOIC) PACKAGE
(TOP VIEW)



SN54HC373 . . . FK PACK
(TOP VIEW)



description

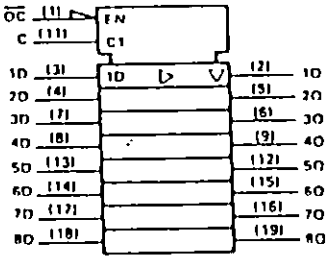
These 8-bit latches feature three-state outputs designed specifically for driving highly capacitive or relatively low-impedance loads. They are particularly suitable for implementing buffer registers, I/O ports, bidirectional bus drivers, and working registers.

The eight latches of the 'HC373 are transparent D-type latches. While the enable (C) is high the Q outputs will follow the data (D) inputs. When the enable is taken low, the Q outputs will be latched at the levels that were set up at the D inputs.

An output-control input ($\overline{OC}$) can be used to place the eight outputs in either a normal logic state (high or low logic levels) or a high-impedance state. In the high-impedance state the outputs neither load nor drive the bus lines significantly. The high-impedance third state and increased drive provide the capability to drive the bus lines in a bus-organized system without need for interface or pull-up components.

The output control $\overline{OC}$ does not affect the internal operations of the latches. Old data can be retained or new data can be entered while the outputs are off.

The SN54HC373 is characterized for operation over the full military temperature range of −55°C to 125°C. The SN74HC373 is characterized for operation from −40°C to 85°C.



FUNCTION TABLE (EACH LATCH)

INPUTS			OUTPUT
$\overline{OC}$	ENABLE C	D	Q
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

TEXAS INSTRUMENTS
POST OFFICE BOX 225012 • DALLAS TEXAS 75265

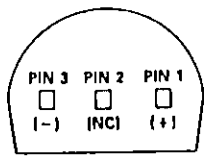
ANALOG DEVICES

Low Cost, Precision IC Temperature Transducer

AD592*

ATURES
- **gh Precalibrated Accuracy: 0.5°C max @ 25°C**
- **cellent Linearity: 0.15°C max (0 to +70°C)**
- **ide Operating Temperature Range: −25°C to +105°C**
- **ngle Supply Operation: +4V to +30V**
- **cellent Repeatability and Stability**
- **gh Level Output: 1μA/K**
- **wo Terminal Monolithic IC: Temperature In/ Current Out**
- **inimal Self-Heating Errors**

CONNECTING DIAGRAM (BOTTOM VIEW)



*PIN 2 CAN BE EITHER ATTACHED OR UNCONNECTED

ODUCT DESCRIPTION

e AD592 is a two terminal monolithic integrated circuit temature transducer that provides an output current proportional absolute temperature. For a wide range of supply voltages the nsducer acts as a high impedance temperature dependent rrent source of 1μA/K. Improved design and laser wafer mming of the IC's thin film resistors allows the AD592 to ieve absolute accuracy levels and nonlinearity errors previously tainable at a comparable price.

e AD592 can be employed in applications between −25°C d +105°C where conventional temperature sensors (i.e., thermistor, RTD, thermocouple, diode) are currently being used. e inherent low cost of a monolithic integrated circuit in a astic package, combined with a low total parts count in any en application, make the AD592 the most cost effective temperature transducer currently available. Expensive linearization rcuitry, precision voltage references, bridge components, resistance measuring circuitry and cold junction compensation are t required with the AD592.

ypical application areas include; appliance temperature sensing, tomotive temperature measurement and control, HVAC (heating/ventilating/air conditioning) system monitoring, industrial mperature control, thermocouple cold junction compensation, ard-level electronics temperature diagnostics, temperature adout options in instrumentation, and temperature correction rcuitry for precision electronics. Particularly useful in remote nsing applications, the AD592 is immune to voltage drops and oltage noise over long lines due to its high impedance current tput. AD592s can easily be multiplexed; the signal current an be switched by a CMOS multiplexer or the supply voltage an be enabled with a tri-state logic gate.

The AD592 is available in three performance grades; the AD592AN, AD592BNand AD592CN. All devices are packaged in a plastic TO-92 case rated from −45°C to +125°C. Performance is specified from −25°C to +105°C. AD592 chips are also available, contact the factory for details.

PRODUCT HIGHLIGHTS

1. With a single supply (4V to 30V) the AD592 offers 0.5°C temperature measurement accuracy.
2. A wide operating temperature range (−25°C to +105°C) and highly linear output make the AD592 an ideal substitute for older, more limited sensor technologies (i.e., thermistors, RTDs, diodes, thermocouples).
3. The AD592 is electrically rugged; supply irregularities and variations or reverse voltages up to 20V will not damage the device.
4. Because the AD592 is a temperature dependent current source, it is immune to voltage noise pickup and IR drops in the signal leads when used remotely.
5. The high output impedance of the AD592 provides greater than 0.5°C/V rejection of supply voltage drift and ripple.
6. Laser wafer trimming and temperature testing insures that AD592 units are easily interchangeable.
7. Initial system accuracy will not degrade significantly over time. The AD592 has proven long term performance and repeatability advantages inherent in integrated circuit design and construction.

TO-92 (N)

LINEAR

Analog Devices

TWO TECHNOLOGY WAY • P.O. BOX 280 • NORWOOD, MA 02062 • (617) 329-4700

MAXIM

CMOS High Speed 8 Bit A/D Converter with Reference and Track/Hold Function

General Description

The MAX150/AD7820 is a high speed, microprocessor compatible, 8 bit analog to digital converter which uses a half-flash technique to achieve a conversion time of 1.34 μs. The converter has a 0V to +5V analog input range and uses a single +5V supply.

A built-in track-and-hold function is included, eliminating the need for an external track-and-hold for input slew rates up to 100mV/μs. The MAX150 also provides an on-chip 2.5 V reference output, making it a complete analog to digital converter.

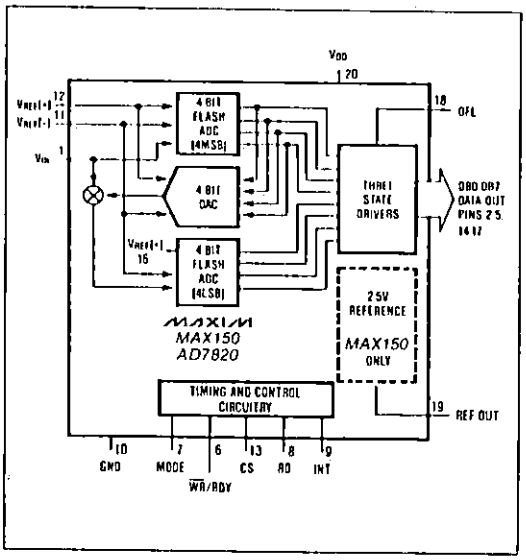
The A/Ds easily interface with microprocessors by appearing as a memory location or I/O port without the need for external interfacing logic. The data outputs use latched, three-state buffer circuitry to allow direct connection to a microprocessor data bus or system input port. An over-flow output is also provided for cascading devices to achieve higher resolution.

The AD7820 is pin compatible with Analog Devices' AD7820. The MAX150 is also compatible with the AD7820 but also includes an internal 2.5V reference.

Applications

Digital Signal Processing
High Speed Data Acquisition
Telecommunications
High Speed Servo Loops
Audio Systems

Functional Block Diagram



Features

- Fast Conversion Time: 1.34μs Max.
- Built-In Track-and-Hold Function
- No Adjustment Required
- No External Clock
- Single +5V Supply
- Easy Interface To Microprocessors
- Internal 2.5V Reference (MAX150 only)

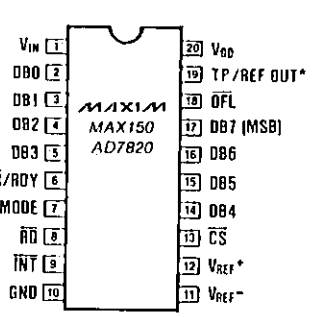
Ordering Information

PART	TEMP. RANGE	PACKAGE†	ERROR
MAX150ACPP	0°C to +70°C	Plastic DIP	±½ LSB
MAX150BCPP	0°C to +70°C	Plastic DIP	±1 LSB
MAX150BC/D	0°C to +70°C	Dice*	±1 LSB
MAX150ACWP	0°C to +70°C	Small Outline	±½ LSB
MAX150BCWP	0°C to +70°C	Small Outline	±1 LSB
MAX150AEPP	-40°C to +85°C	Plastic DIP	±½ LSB
MAX150BEPP	-40°C to +85°C	Plastic DIP	±1 LSB
MAX150AEWP	-40°C to +85°C	Small Outline	±½ LSB
MAX150BEWP	-40°C to +85°C	Small Outline	±1 LSB
MAX150AMJP	-55°C to +125°C	CERDIP	±½ LSB
MAX150BMJP	-55°C to +125°C	CERDIP	±1 LSB

† *All devices — 20 lead packages*
* *Consult factory for dice specifications.*
Ordering Information continued on last page

Pin Configuration

Top View



*REF OUT — MAX150 only.

CMOS High Speed 8 Bit A/D Converter with Reference and Track/Hold Function

ABSOLUTE MAXIMUM RATINGS

Supply Voltage, V_{DD} to GND ..... 0V, +10V
Voltage at any other pins
(Pins 1-9, 11-19) ..... GND - 0.3V, V_{DD} +0.3V
Output current (Pin 19) 30mA
Power Dissipation (Any Package) to 75°C 450mW
Derate Above +75°C by ..... 6mW/°C

Operating Temperature Ranges
MAX150XCXX, AD7820LN/KN/LCWP/KCWP ... 0°C to +70°C
AD7820BQ/CQ ..... -25°C to +85°C
MAX150XEXX ..... -40°C to +85°C
MAX150XMXX, AD7820TQ/UQ ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (Soldering 10 seconds) ..... +300°C

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

ELECTRICAL CHARACTERISTICS

(V_{DD} = +5V, V_{REF}^+ = +5V, V_{REF}^- = GND, RD-MODE, T_A = T_{MIN} to T_{MAX}, unless otherwise noted)

PARAMETER	SYMBOL	CONDITIONS	MIN.	TYP.	MAX.	UNITS
ACCURACY						
Resolution			8			bits
Total Unadjusted Error (Note 1)		MAX150A, AD7820L/C/U			±1/2	LSB
		MAX150B, AD7820K/B/T			±1	
No Missing Codes Resolution			8			bits
REFERENCE INPUT						
Reference Resistance		T_A = +25°C	1.4	2.2	4.0	kΩ
		T_A = T_{MIN} to T_{MAX}	1.25		4.0	
V_{REF}^+ Input Voltage Range			V_{REF}^-		V_{DD} + 0.1	V
V_{REF}^- Input Voltage Range			GND - 0.1		V_{REF}^+	V
REFERENCE OUTPUT MAX150 ONLY (Note 2)						
Output Voltage	REF OUT	T_A = +25°C	2.47	2.50	2.53	V
Load Regulation		I_L = 0 to 10mA T_A = +25°C		-6	-10	mV
Power Supply Sensitivity		V_{DD} ±5% T_A = +25°C		±1	±3	mV
Temperature Drift (Note 3)		MAX150XC T_A = 0°C to +70°C		40	70	ppm/°C
		MAX150XE T_A = -40°C to +85°C		40	70	
		MAX150XM T_A = -55°C to +125°C		60	100	
Output Noise				200		µV/rms
Capacitive Load					0.01	µF
ANALOG INPUT						
Analog Input Voltage Range	V_{INR}		GND - 0.1		V_{DD} + 0.1	V
Analog Input Capacitance	C_{VIN}			45		pF
Analog Input Current	I_{VIN}	V_{IN} = 0V to +5V T_A = +25°C			±0.3	µA
		T_A = T_{MIN} to T_{MAX}			±3	
Slew Rate, Tracking (Note 4)	SR			0.2	0.1	V/µs
LOGIC INPUTS						
Input HIGH Voltage	V_{INH}	$\overline{CS}$, $\overline{WR}$, $\overline{RD}$: MAX150	2.0			V
		AD7820	2.4			
		MODE	3.5			
Input LOW Voltage	V_{INL}	$\overline{CS}$, $\overline{WR}$, $\overline{RD}$			0.8	V
		MODE			1.5	
Input High Current	I_{INH}	$\overline{CS}$, $\overline{RD}$: T_A = +25°C			0.3	µA
		T_{MIN} to T_{MAX}			1	
		$\overline{WR}$; T_A = +25°C			0.3	
		T_{MIN} to T_{MAX}			3	
		MODE: T_A = +25°C		50	150	
		T_{MIN} to T_{MAX}			200	

Note 1: Total unadjusted error includes offset, full-scale and linearity errors.
Note 2: Specified with no external load unless otherwise noted.
Note 3: Temperature drift is defined as change in output voltage from +25°C to T_{MIN} or T_{MAX} divided by (25 - T_{MIN}) or (T_{MAX} - 25)
Note 4: Sample tested at +25°C by Quality Assurance to ensure compliance

MAXIM

CMOS High Speed 8 Bit A/D Converter with Reference and Track/Hold Function

ELECTRICAL CHARACTERISTICS (continued)

(V_{DD} = +5V, V_{REF}^+ = +5V, V_{REF}^- = GND, RD-MODE, T_A = T_{MIN} to T_{MAX}, unless otherwise noted)

PARAMETER	SYMBOL	CONDITIONS	MIN.	TYP.	MAX.	UNITS
LOGIC INPUTS (continued)						
Input Low Current	I_{INL}	$\overline{CS}$, $\overline{RD}$, $\overline{WR}$, MODE T_A = +25°C T_{MIN} to T_{MAX}			-0.3 -1	µA
Input Capacitance (Note 5)	C_{IN}	$\overline{CS}$, $\overline{RD}$, $\overline{WR}$, MODE		5	8	pF
LOGIC OUTPUTS						
Output HIGH Voltage	V_{OH}	DB0-DB7, $\overline{OFL}$, $\overline{INT}$ V_{DD} = +4.75V I_{OUT} = -360µA V_{DD} = +4.75V I_{OUT} = -10µA	4.0 4.5			V
Output LOW Voltage	V_{OL}	DB0-DB7, $\overline{OFL}$, $\overline{INT}$, RDY V_{DD} = +4.75V I_{OUT} = 1.6mA			0.4	V
Three-state Output Current		DB0-DB7, RDY T_A = +25°C T_{MIN} to T_{MAX}			±0.3 ±3	µA
Output Source Current	I_{SRC}	DB0-DB7, $\overline{OFL}$, $\overline{INT}$, V_{OUT} = 0	-10	-25		mA
Output Sink Current	I_{SINK}	DB0-DB7, $\overline{OFL}$, $\overline{INT}$, RDY; V_{OUT} = V_{DD}	15	40		mA
Output Capacitance (Note 5)	C_{OUT}	DB0-DB7, $\overline{OFL}$, $\overline{INT}$, RDY		5	8	pF
POWER SUPPLY						
Supply Voltage	V_{DD}	+5V ±5% for specified performance	4.75		5.25	V
Supply Current	I_{DD}	$\overline{CS}$ = $\overline{WR}$ = $\overline{RD}$ = 0 T_A = +25°C T_{MIN} to T_{MAX}		5	10 15	mA
Power Dissipation		$\overline{CS}$ = $\overline{WR}$ = $\overline{RD}$ = 0		25		mW
Power Supply Sensitivity	PSS	V_{DD} = ±5%		±1/16	±1/4	LSB

Note 5: Guaranteed by design.

Pin Description

PIN	NAME	FUNCTION
1	V_{IN}	Analog input; range = GND < V_{IN} < V_{DD}.
2	DB0	Three-state data output, bit 0 (LSB).
3	DB1	Three-state data output, bit 1.
4	DB2	Three-state data output, bit 2.
5	DB3	Three-state data output, bit 3.
6	$\overline{WR}$/RDY	WRITE control input/READY status output. See Digital Interface section.
7	MODE	Mode selection input. This input is internally pulled low with a 50µA current source. RD Mode: MODE low/open. WR-RD Mode: MODE high.
8	$\overline{RD}$	READ input. $\overline{RD}$ must be low to access data. See Digital Interface section.
9	$\overline{INT}$	INTERRUPT output. $\overline{INT}$ going low indicates the completion of a conversion. See Digital Interface section.
10	GND	Ground.

PIN	NAME	FUNCTION
11	V_{REF}^-	Lower limit of reference span. Sets the zero code voltage. Range: GND to V_{REF}^+
12	V_{REF}^+	Upper limit of reference span. Sets the Full Scale input voltage. Range: V_{REF}^- to V_{DD}.
13	$\overline{CS}$	CHIP-SELECT input. $\overline{CS}$ must be low for the device to recognize $\overline{WR}$ or $\overline{RD}$ inputs
14	DB4	Three-state data output, bit 4.
15	DB5	Three-state data output, bit 5.
16	DB6	Three-state data output, bit 6.
17	DB7	Three-state data output, bit 7 (MSB).
18	$\overline{OFL}$	Overflow Output. If the analog input is greater than V_{REF}^+, $\overline{OFL}$ will be high at the end of the conversion. It can be used to cascade two or more devices to increase resolution.
19	TP REF OUT	Test pin for AD7820. Do not connect pin 19 for AD7820. 2.5V Internal reference output for MAX150 only.
20	V_{DD}	Power supply voltage, +5V.

STANDARD MICROSYSTEMS CORPORATION

We keep ahead of our competition so you can keep ahead of yours.

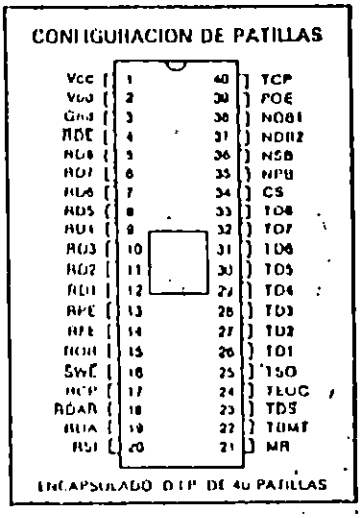
COM 2502
COM 2017
COM 2502/H
COM 2017/H

TRANSMISOR/RECEPTOR ASINCRONO UNIVERSAL (UART)

CARACTERISTICAS

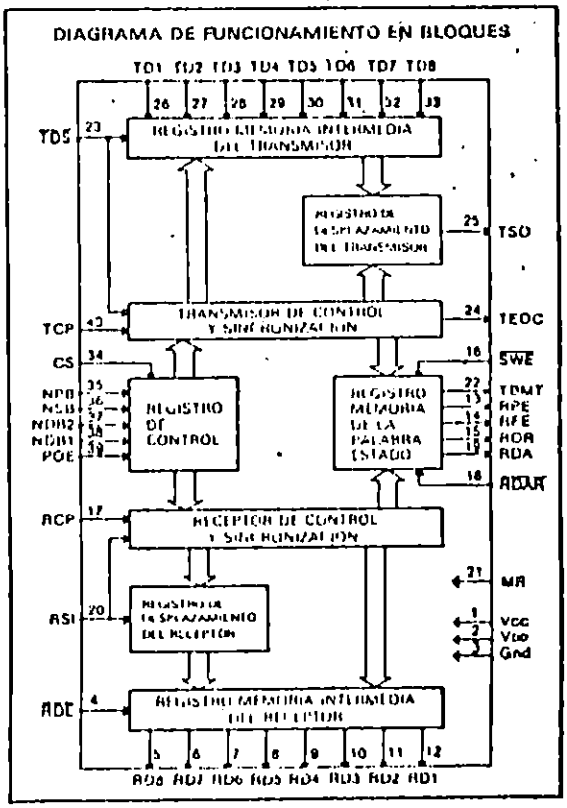
- Compatibilidad directa con TTL; no se requieren circuitos de interfaz.
- Funcionamiento completo en dúplex o semidúplex; puede recibir y transmitir simultáneamente a diferentes velocidades de transmisión de baudios.
- Completamente separado por doble partida, lo que elimina la necesidad de sincronización externa exacta.
- Verificación del bit de arranque, que disminuye la tasa de errores.
- Completamente programable; longitud de la palabra de datos, modo de paridad, número de bits de parada, uno, uno y medio o dos.
- Funcionamiento a alta velocidad; 40 kilobaudios, referencias de 200 ns.
- Reinicialización principal («maestra»). Pone a cero todas las salidas de estado.
- Salidas triestados, diseñadas para una estructura tipo bus.
- Bajo consumo; requisitos mínimos de alimentación.
- Entrada protegida, lo que elimina problemas de manejo.
- Encapsulado DIP cerámico o plástico, para inserción fácil en la tarjeta.

CONFIGURACION DE PATILLAS



ENCAPSULADO D.I.P. DE 40 PATILLAS

DIAGRAMA DE FUNCIONAMIENTO EN BLOQUES

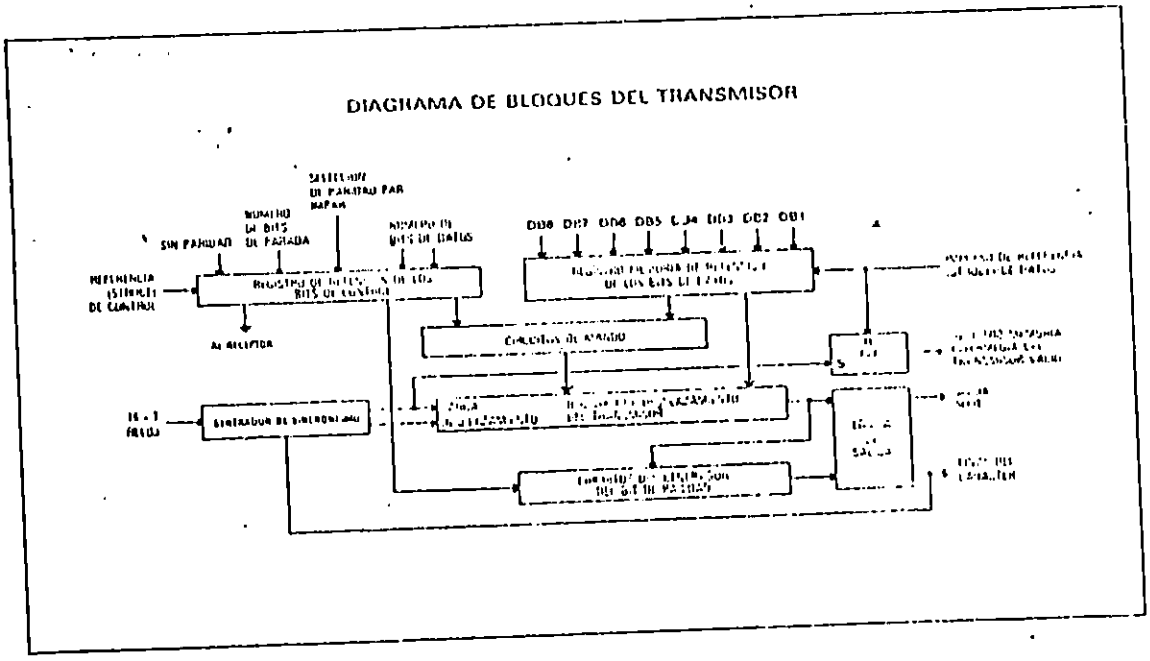


DESCRIPCION GENERAL

El transmisor/receptor asíncrono universal es un circuito monolítico MOS/LSI que ejecuta todas las funciones de recepción y transmisión asociadas con las comunicaciones asíncronas de datos. Este circuito se fabrica usando el procedimiento canal-P de óxido nitruro a baja tensión de SMC. El modo dúplex, la velocidad de transmisión en la longitud de la palabra de datos, el modo de paridad y el número de bits de parada son programables independientemente mediante la aplicación de controles del exterior. Puede haber 5, 6, 7 u 8 bits de datos, paridad par/impar o no paridad, y 1 ó 2 bits de parada, o 1,5 bits de parada cuando se utiliza el código de 5 bits de las COM 2017 o COM 2017/H. El UART puede trabajar en cualquiera de los modos dúplex o semidúplex. Estas características programables proporcionan al usuario la posibilidad de interconexión («interfaz») con todos los periféricos asíncronos.

DESCRIPCION DEL FUNCIONAMIENTO COMO TRANSMISOR

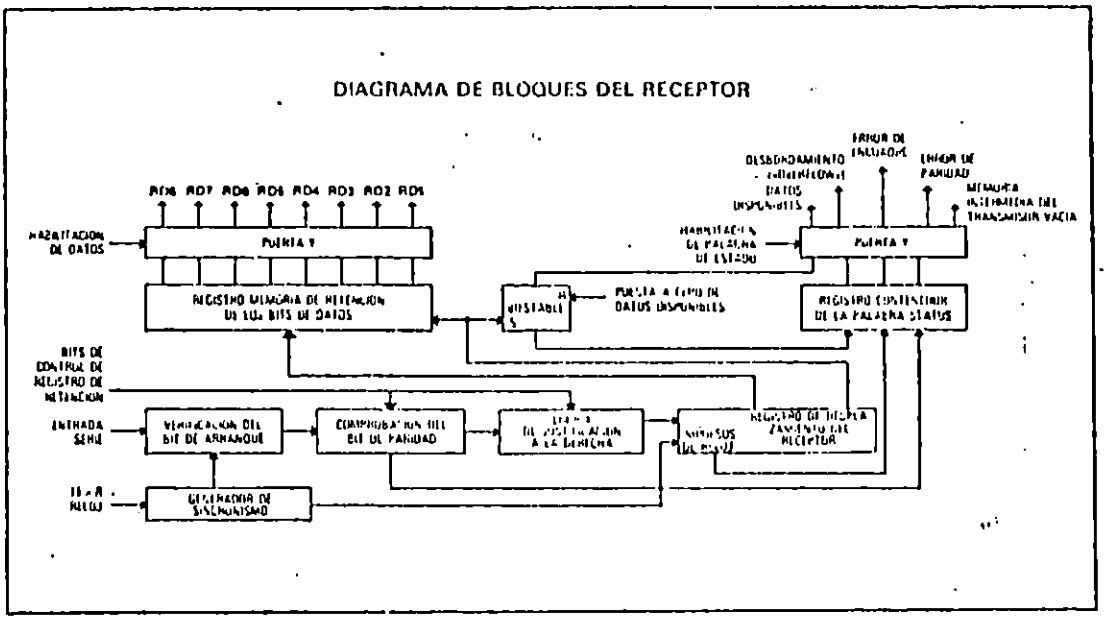
Al encenderlo para ponerlo en marcha, se aplica un tren de impulsos de reloj cuya frecuencia es 16 veces la velocidad de transmisión en baudios deseada y se pulsa la reinicialización (reset) maestra. Bajo estas condiciones TBMT, TEOC y TSO están todas a nivel alto (la línea es activada).
Cuando TBMT y TEOC están a nivel alto, los bits de control pueden estar a uno. Después que esto ocurre los bits de datos pueden estar a uno. Es normal que los bits de control sean introducidos en el transmisor por el impulso de referencia antes que los bits de datos. Sin embargo, y mientras que no se incumplan las especificaciones de anchura mínima del impulso, TDS y CS pueden ocurrir simultáneamente. Una vez que el impulso de referencia de datos (TDS) ha ocurrido, la señal TMBT se va a nivel bajo, indicando que el registro memoria de los bits de datos está lleno y no puede recibir más información.
Si el registro de desplazamiento del transmisor está transmitiendo los datos cargados previamente, la señal TBMT permanece a nivel bajo. Si el registro de desplazamiento del transmisor está vacío, o cuando ha terminado de transmitir el carácter previo, los datos del registro memoria pasan inmediatamente al registro de desplazamiento del transmisor y comienza la transmisión de datos. TSO va a nivel bajo (el bit de arranque). TEOC va a nivel bajo, TBMT va a nivel alto indicando que los datos del registro memoria han sido cargados en el registro de desplazamiento del transmisor y que el primero está disponible para recibir datos nuevos.
Si en este momento se cargan nuevos datos en el registro memoria, TBMT vuelve a nivel bajo y permanece en este estado hasta que se completa la transmisión en curso. Se dispone del tiempo que ocupa la transmisión completa de un carácter para cargar el siguiente carácter sin perder la velocidad de transmisión. Esta es la ventaja de poseer dos registros intermedios.
La transmisión de datos ocurre de una forma ordenada: bit de arranque, bits de datos, bit de paridad (si se ha seleccionado) y bit(s) de parada. Cuando el último bit de parada ha sido puesto en la línea, TEOC pasa a nivel alto por la duración de un bit. Si TBMT está a nivel bajo, la transmisión comienza inmediatamente. Si TBMT está a nivel alto, el transmisor queda en reposo completo y, si así se desea, se pueden cargar nuevos bits de control antes de la siguiente transmisión de datos.



DESCRIPCION DEL FUNCIONAMIENTO COMO RECEPTOR

Al encenderlo para ponerlo en marcha, se aplica un tren de impulsos de reloj cuya frecuencia es 16 veces la velocidad de transmisión en baudios deseada y se pulsa la reinicialización (reset) maestra. La señal de datos disponibles (RDA) está ahora a nivel bajo. Hay solo un conjunto de bits de control para transmisor y receptor.
La recepción de datos comienza cuando la línea de entrada en serie pasa de marca (nivel alto) a espacio (nivel bajo). Si la línea RSI permanece en espacio durante la duración de medio bit, es que se ha verificado un bit de arranque verdadero. Si la línea retornara a la condición de marca (trabajo) antes de que transcurriera el tiempo correspondiente a medio bit, el proceso de verificación del bit de arranque comenzaría de nuevo. Para empezar la verificación del bit de arranque tiene que ocurrir en la línea una transición de marca a espacio. Una vez que se ha verificado la recepción del bit de arranque, tiene lugar la recepción de los datos de forma ordenada: bit de arranque recibido y verificado, bits de datos recibidos, bit de paridad recibido (si se ha seleccionado) y bit(s) de parada recibidos.
Si el bit de paridad transmitido no coincide con el recibido, se pone a uno el biestable de error de paridad del registro memoria que contiene la palabra estado, indicando un error de paridad. Sin embargo, si no se ha seleccionado modo de paridad, el biestable de error de paridad se mantiene a cero incondicionalmente, inhibiendo la indicación pertinente. Si no se recibe el bit de parada, debido al encuadre inapropiado del carácter, se pone a uno el biestable de error de encuadre, indicando el error pertinente.
Una vez que se ha recibido un carácter completo, la circuitería lógica interna comprueba el estado de la señal datos disponibles (RDA). Si en este momento la señal presenta un nivel alto (el receptor ha leído todo el carácter precedente no ha sido leído y se pone a uno el biestable de desbordamiento (overflow). La única manera de que el receptor advierta que ha recibido los datos es poner a cero la señal de datos disponibles.

En este momento la salida de RDA se va a nivel alto indicando que todas las salidas pueden ser examinadas. El registro de desplazamiento del receptor está disponible ahora para empezar a recibir el próximo carácter. Dado la existencia de dos registros intermedios, se dispone del tiempo de recepción de un carácter completo para retirar el recibido.

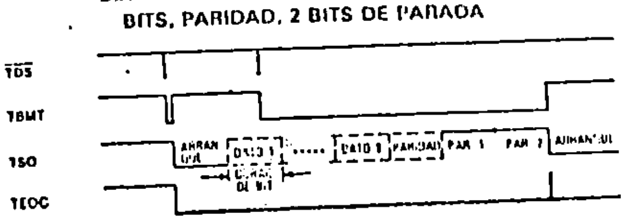


DESCRIPCION DE LAS FUNCIONES DE LA PATILLA

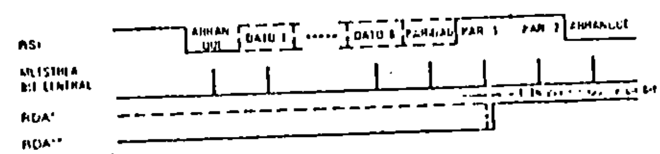
PATILLA NUMERO	SIMBOLO	NOMBRE	FUNCION
1	V_{CC}	Fuente de alimentación	+5 voltios
2	V_{DD}	Fuente de alimentación	−12 voltios
3	GND	Tierra	Tierra
4	$\overline{RDE}$	Hab. para los datos recibidos	Una entrada de nivel bajo autoriza las salidas (RD8-RD1) del registro memoria del receptor.
5-12	RD8-RD1	Dalidas de datos del receptor	Son las 8 salidas triestados habilitadas por $\overline{RDE}$. Las líneas de datos que no se usan, de acuerdo con la selección de NDB1 y NDB2, presentan la salida a nivel bajo y los caracteres recibidos están justificados a la derecha, es decir, que el bit menos significativo aparece en la salida RD1.
13	RPE	Error de paridad en el receptor	La salida triestado (habilitada por $\overline{SWE}$) está a nivel alto si el bit de paridad del carácter recibido no coincide con la paridad seleccionada.
14	RFE	Error de encuadre del receptor	La salida triestado (habilitada por $\overline{SWE}$) está a nivel alto si el carácter recibido no lleva un bit de parada válido
15	ROR	Desbordamiento del receptor	La salida triestado (habilitada por $\overline{SWE}$) está a nivel alto si el carácter recibido anteriormente no fue leído (la salida RDA no se puso a cero) antes de que el caracter en curso sea transferido al registro memoria del receptor.
16	$\overline{SWE}$	Hab. para la palabra de status	Una entrada a nivel bajo habilita las salidas (RPE, RFE, ROR, RDA y a TBMT) del registro memoria de la palabra de status
17	RCP	Reloj del receptor	Esta entrada es un tren de impulsos de reloj cuya frecuencia es 16 veces (16×) la velocidad de transmisión en baudios deseada.
18	RDAR	Puesta a cero de datos del receptor disponibles	Una entrada de nivel bajo pone a cero la salida RDA.
19	RDA	Datos del receptor disponible	La salida triestado (habilitada por $\overline{SWE}$) está a nivel alto cuando se ha recibido un carácter completo y se lo ha transferido al registro memoria del receptor.
20	RSI	Entrada serie del receptor	Esta entrada acepta la corriente de entrada de bits en serie. Se requiere una transición de nivel alto (marca) a nivel bajo (espacio) para comenzar la recepción de datos.
21	MR	Reinicialización maestra	Esta entrada debe conectarse a nivel alto después de haber encendido el circuito. Ello pone las TSO, TEOC y TBMT a nivel alto, y las RDA, DPE, RFE y ROR a nivel bajo.

PATILLA NUMERO	SIMBOLO	NOMBRE	FUNCION
22	TBMT	Memoria del transmisor vacío	Esta salida triestado (habilitada por SWE) está a nivel alto cuando el registro memoria del transmisor puede ser cargado con información nueva.
23	$\overline{TDS}$	Impulso de referencia de datos en el transmisor	Un impulso de referencia de nivel bajo introduce los bits de datos en el registro memoria del transmisor.
24	TEOC	Final del carácter en el transmisor	Esta señal aparece como un nivel alto cada vez que se completa la transmisión de un carácter. Permanece a este nivel hasta el principio de la transmisión del carácter siguiente, o durante la mitad de un período de reloj (TCP) en el caso de transmisión continua.
25	TSO	Salida en serie del transmisor	Esta salida suministra en serie la totalidad del carácter transmitido. La TSO permanece a nivel alto mientras que no se está transmitiendo datos.
26-33	TD1-TD8	Entradas de datos al transmisor	Hay 8 líneas de entrada de datos (referenciados por TDS) disponibles. Las líneas no utilizadas, de acuerdo con la selección hecha por NDB1 y NDB2, pueden estar en cualquier estado lógico. El bit menos significativo (LSB) debe estar situado siempre en TD1.
34	CS	Impulso de referencia de control	Una entrada de nivel alto llega a los bits de control (NDB1, NDB2, NSB, POE y NPB) del registro de retención de bits de control. Esta señal debe estar referenciada o cableada a un nivel alto.
35	NPB	Sin bit de paridad	Una entrada de nivel alto elimina la transmisión del bit de paridad, los bits de parada siguen inmediatamente al último bit de datos. Además, el receptor requiere que los bits de parada sigan inmediatamente al último de datos. También se fuerza a la salida RPE a tomar un valor de nivel bajo. Ver patilla 39, POE.
36	NSB	Número de bits de parada	Esta entrada selecciona el número de bits de parada. Una entrada de nivel bajo selecciona 1 bit de parada; una de nivel alto selecciona 2 bits. La selección de 2 bits, cuando se ha programado una palabra de 5 bits, origina 1,5 bits de parada en el COM 2017 o en el COM 2017 H.
37-38	NDB2, NDB1	Número de bits de datos por carácter	Estas dos entradas se decodifican internamente para seleccionar 5, 6, 7 u 8 bits de datos por carácter, según la siguiente tabla de verdad: NDB2 — NDB1 — bits de datos/carácter L — L — 5 L — H — 6 H — L — 7 H — H — 8
39	POE	Selección de paridad par/impar	El nivel lógico de esta entrada, en conjunción con la de NPB, determina el modo de paridad para receptor y transmisor, de acuerdo con la siguiente tabla de verdad: NPB — POE — MODO L — L — Paridad impar L — H — Paridad impar H — X — Sin paridad no afecta
40	TCP	Reloj de transmisor	Esta entrada es un tren de impulsos de reloj cuya frecuencia es 16 veces (16 ×) la velocidad de transmisión en baudios que se desea.

SINCRONIZACION DEL TRANSMISOR - 8 BITS, PARIDAD, 2 BITS DE PARADA

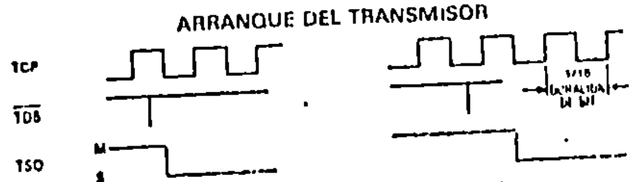


RECEPCION DEL TRANSMISOR - 8 BITS, PARIDAD, 2 BITS DE PARADA



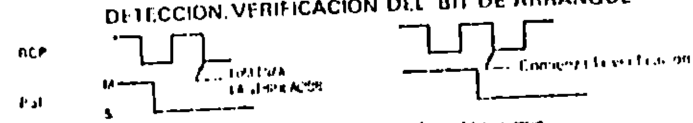
* La línea RDA no fue puesta a cero previamente (RDR = nivel alto)
** La línea RDA fue puesta a cero previamente (RDR = nivel bajo)

ARRANQUE DEL TRANSMISOR



A la iniciación de la transmisión de datos, o cuando las líneas no se utilizan, el TDS ... la transmisión del bit de arranque se ... en la línea TSO a la transición de nivel alto a nivel bajo del reloj TCP, inmediatamente después del flanco posterior de TDS.

DETECCION, VERIFICACION DEL BIT DE ARRANQUE



Si la línea RSI ... durante la mitad del tiempo de 1 bit, ... un bit de arranque ... la verificación del bit de arranque ...

ANEXO B

CIRCUITO EMPLEADO PARA CORRER PROGRAMAS PASO A PASO

IMPLEMENTACION DE UN CIRCUITO PASO A PASO EN EL SISTEMA.

La función de paso a paso ó paso único se realiza utilizando las señales de control generadas por el Z80-A durante la ejecución del programa. Las dos señales particulares de interés son M1 y WAIT. M1 es una salida y WAIT es una entrada. Como se muestra en la figura B.1., M1 va a un nivel lógico "0" al comienzo de cada ciclo de búsqueda de instrucción. La genereación de la señal M1 significa que el sistema ha completado una instrucción y está comenzando con la siguiente. El objetivo es detener el microprocesador antes de que ejecute ésta siguiente instrucción.

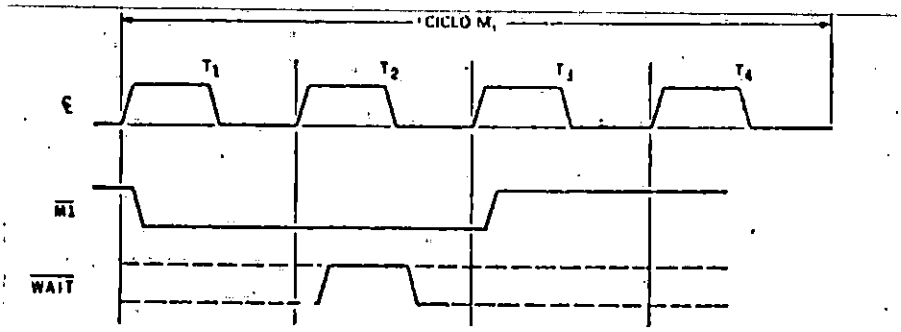


Figura B.1 Cronograma para la busqueda de código de operación (M1) de instrucción.

La entrada WAIT de Z80-A hace justamente eso. Un nivel lógico "0" aplicado a esta entrada suspenderá la ejecución del progrrama del sistema y lo mantendra indefinidamente en el ciclo M1. Durante T2, el procesador central muestrea la línea de entrada WAIT con el flanco posterior del reloj. Si, en éste tiempo, WAIT está a un nivel lógico "0", un estado adicional de espera será introducido y la línea será muestreada de nuevo. El procesador central se mantendrá en esta modalidad hasta que WAIT pase a un nivel lógico "1". Se debe observar que no se trata de una orden de parada del sistema.

El circuito de la figura B.2 nos permite controlar el estado WAIT y ejecutar solamente una instrucción con cada pulsación del botón. La salida del flip-flop (La entrada WAIT), es normalmente baja, dando lugar a una espera indefinida. Cuando el botón es pulsado, un pulso único con supresión de rebotes sincroniza al flip-flop que es disparado por el flanco y solamente está relacionado con el flanco de entrada. Al pulsar el botón se activa el flip-flop y se eleva el nivel de la línea WAIT. Ya que no se le dice que espere ,el procesador central ejecuta la instrucción a plena velocidad del reloj. Como está a punto de empezar el

siguiente ciclo de búsqueda de instrucción, M1 pasa a nivel bajo como antes y detiene el circuito controlable.

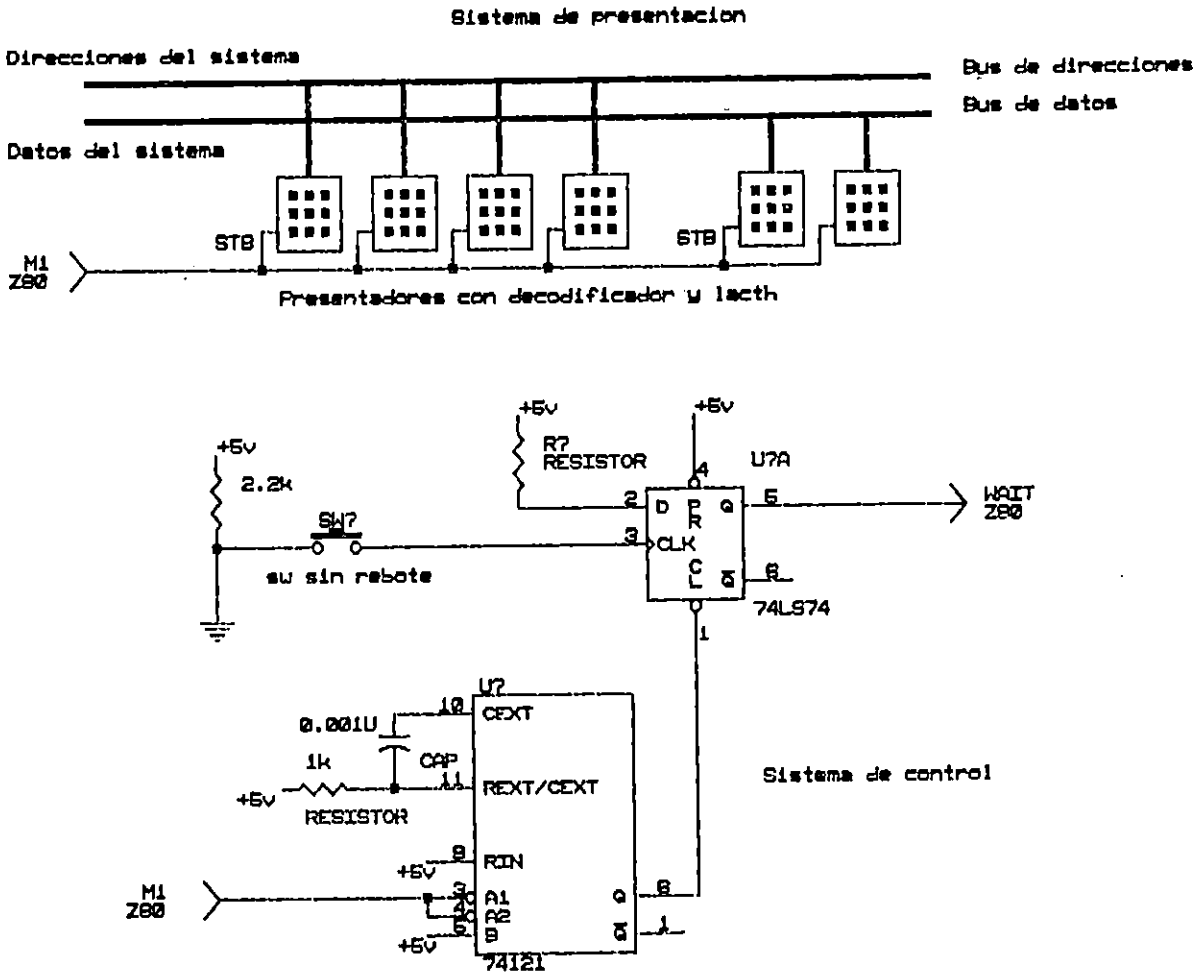


Figura B.2 Circuito de paso simple de instrucción

En la figura B.2 puede observarse tanto las direcciones como los datos de un programa particular. Esto es muy importante en la depuración de programas ,ya que se puede seguir el flujo de las instrucciones y corroborar la ejecución del programa.

Un detalle muy importante del circuito de paso único es que éste se detiene en cada operación M1, lo cual no siempre concuerda con el inicio de cada instrucción. Esto se debe a que el Z80-A cuenta con instrucciones que realizan dos ciclos M1 (estos códigos de operación comienzan siempre con

CB,DD,ED o FD), por tanto debe tenerse presente a la hora de seguir la secuencia de un programa.

ANEXO C

CODIGOS DE PATRONES Y PATRONES DE CARACTERES EMPLEADOS EN EL SISTEMA

```
patrones    db    ffh,ffh,ffh,ffh,ffh,00      FUNC     DEC     HEX
            db    ffh,ffh,ffh,ffh,ffh,00
            db    00h,21h,7fh,01h,00h,00      ;1       ,2
            db    27h,49h,49h,49h,31h,00      ;2       ,3
            db    22h,41h,49h,49h,36h,00      ;3       ,4
            db    0ch,14h,24h,7fh,04h,00      ;4       ,5
            db    72h,51h,51h,51h,4eh,00      ;5       ,6
            db    1eh,29h,49h,49h,46h,00      ;6       ,7
            db    40h,47h,48h,50h,60h,00      ;7       ,8
            db    36h,49h,49h,49h,36h,00      ;8       ,9
            db    31h,49h,49h,4ah,3ch,00      ;9       ,10     ,a
            db    3eh,45h,49h,51h,3eh,00      ;0       ,11     ,b
            db    14h,14h,14h,14h,14h,00      ;=       ,12     ,c
            db    08h,08h,08h,08h,08h,00      ;-       ,13     ,d
            db    20h,10h,08h,04h,02h,00      ;\       ,14     ,e
            db    ffh,ffh,ffh,ffh,ffh,00      ;        ;15     ,f
            db    ffh,ffh,ffh,ffh,ffh,00      ;        ;16     ,10
            db    ffh,ffh,ffh,ffh,ffh,00      ;        ;17     ,11
            db    3eh,41h,45h,42h,3dh,00      ;Q       ,18     ,12
            db    7fh,02h,0ch,02h,7fh,00      ;W       ,19     ,13
            db    7fh,49h,49h,49h,41h,00      ;E       ,20     ,14
            db    7fh,48h,4ch,4ah,31h,00      ;R       ,21     ,15
            db    40h,40h,7fh,40h,40h,00      ;T       ,22     ,16
            db    60h,10h,0fh,10h,60h,00      ;Y       ,23     ,17
            db    7eh,01h,01h,01h,7eh,00      ;U       ,24     ,18
            db    00h,41h,7fh,41h,00h,00      ;I       ,25     ,19
            db    3eh,41h,41h,41h,3eh,00      ;O       ,26     ,1a
            db    7fh,48h,48h,48h,30h,00      ;P       ,27     ,1b
            db    00h,00h,7fh,41h,41h,00      ;[       ,28     ,1c
            db    41h,41h,7fh,00h,00h,00      ;]       ,29     ,1d
            db    00h,00h,70h,00h,00h,00      ;'       ,30     ,1e
            db    00h,30h,48h,30h,00h,00      ;o       ,31     ,1f
            db    ffh,ffh,ffh,ffh,ffh,00      ;        ,32     ,20
            db    ffh,ffh,ffh,ffh,ffh,00      ;        ,33     ,21
            db    1fh,24h,44h,24h,1fh,00      ;A       ,34     ,22
            db    31h,49h,49h,49h,46h,00      ;S       ,35     ,23
            db    7fh,41h,41h,22h,1ch,00      ;D       ,36     ,24
            db    7fh,48h,48h,48h,40h,00      ;F       ,37     ,25
            db    3eh,41h,41h,49h,4eh,00      ;G       ,38     ,26
            db    7fh,08h,08h,08h,7fh,00      ;H       ,39     ,27
            db    02h,01h,41h,7eh,40h,00      ;J       ,40     ,28
            db    7fh,08h,14h,22h,41h,00      ;K       ,41     ,29
            db    7fh,01h,01h,01h,01h,00      ;L       ,42     ,2a
            db    5fh,48h,44h,42h,5fh,00      ;Ñ       ,43     ,2b
            db    00h,6bh,6eh,00h,00h,00      ;";"     ,44     ,2c
            db    ffh,ffh,ffh,ffh,ffh,00      ;Enter   ,45     ,2d
            db    ffh,ffh,ffh,ffh,ffh,00      ;        ,46     ,2e
            db    ffh,ffh,ffh,ffh,ffh,00      ;        ,47     ,2f
            db    ffh,ffh,ffh,ffh,ffh,00      ;        ,48     ,30
            db    ffh,ffh,ffh,ffh,ffh,00      ;        ,49     ,31
            db    43h,45h,49h,51h,61h,00      ;Z       ,50     ,32
            db    63h,14h,08h,14h,63h,00      ;X       ,51     ,33
            db    3eh,41h,41h,41h,22h,00      ;C       ,52     ,34
            db    70h,0ch,03h,0ch,70h,00      ;V       ,53     ,35
```

```
db    7fh,49h,49h,49h,36h,00      ;B      ,54     ,36
db    7fh,20h,10h,08h,7fh,00      ;N      ,55     ,37
db    7fh,20h,18h,20h,7fh,00      ;M      ,56     ,38
db    00h,0bh,0eh,00h,00h,00      ;,      ,57     ,39
db    00h,03h,03h,00h,00h,00      ;.      ,58     ,3a
db    02h,04h,08h,10h,20h,00      ;/      ,59     ,3b
db    ffh,ffh,ffh,ffh,ffh,00      ;Shift  ,60     ,3c
db    00h,00h,00h,00h,00h,00      ;space  ,61     ,3d
db    ffh,ffh,ffh,ffh,ffh,00      ;
db    ffh,ffh,ffh,ffh,ffh,00      ;       ,63     ,3f
db    ffh,ffh,ffh,ffh,ffh,00      ;       ,64     ,40
db    ffh,ffh,ffh,ffh,ffh,00      ;               ,41
db    00h,00h,79h,00h,00h,00      ;!      ,66     ,42
db    26h,49h,4fh,41h,3eh,00      ;@      ,67     ,43
db    14h,7fh,14h,7fh,14h,00      ;#      ,68     ,44
db    12h,2ah,3bh,2ah,24h,00      ;$      ,69     ,45
db    62h,64h,08h,13h,23h,00      ;%      ,70     ,46
db    20h,40h,40h,40h,20h,00      ;^      ,71     ,47
db    06h,39h,4dh,32h,05h,00      ;&      ,72     ,48
db    14h,08h,3eh,08h,14h,00      ;*      ,73     ,49
db    00h,1ch,22h,41h,00h,00      ;(      ,74     ,4a
db    00h,41h,22h,1ch,00h,00      ;)      ,75     ,4b
db    08h,08h,3eh,08h,08h,00      ;+      ,76     ,4c
db    01h,01h,01h,01h,01h,00      ;_      ,77     ,4d
db    00h,00h,77h,00h,00h,00      ;|      ,78     ,4e
db    ffh,ffh,ffh,ffh,ffh,00      ;       ;79     ,4f
db    ffh,ffh,ffh,ffh,ffh,00      ;       ;80     ,50
db    ffh,ffh,ffh,ffh,ffh,00      ;       ;81     ,51
db    08h,14h,14h,14h,1fh,00      ;q      ;82     ,52
db    1fh,02h,04h,02h,1fh,00      ;w      ;83     ,53
db    0eh,15h,15h,15h,0ch,00      ;e      ;84     ,54
db    1fh,08h,10h,10h,10h,00      ;r      ;85     ,55
db    10h,7eh,11h,11h,02h,00      ;t      ;86     ,56
db    18h,05h,05h,05h,1eh,00      ;y      ;87     ,57
db    1eh,01h,01h,01h,1fh,00      ;u      ;88     ,58
db    00h,11h,5fh,01h,00h,00      ;i      ;89     ,59
db    0eh,11h,11h,11h,0eh,00      ;o      ;90     ,5a
db    1fh,14h,14h,14h,08h,00      ;p      ;91     ,5b
db    00h,00h,7fh,41h,41h,00      ;[      ,92     ,5c
db    41h,41h,7fh,00h,00h,00      ;]      ,93     ,5d
db    00h,00h,70h,00h,00h,00      ;'      ,94     ,5e
db    ffh,ffh,ffh,ffh,ffh,00      ;       ,95     ,5f
db    ffh,ffh,ffh,ffh,ffh,00      ;       ,96     ,60
db    ffh,ffh,ffh,ffh,ffh,00      ;       ,97     ,61
db    02h,15h,15h,15h,0fh,00      ;a      ,98     ,62
db    08h,15h,15h,15h,02h,00      ;s      ,99     ,63
db    0eh,11h,11h,11h,7fh,00      ;d      ,100    ,64
db    08h,3fh,48h,48h,40h,00      ;f      ,101    ,65
db    08h,15h,15h,15h,1eh,00      ;g      ,102    ,66
db    3fh,10h,10h,10h,0fh,00      ;h      ,103    ,67
db    01h,01h,11h,5eh,00h,00      ;j      ,104    ,68
db    7fh,04h,0ah,11h,00h,00      ;k      ,105    ,69
db    00h,41h,3fh,01h,00h,00      ;l      ,106    ,6a
db    5fh,50h,50h,50h,0fh,00      ;ñ      ,107    ,6b
db    00h,36h,36h,00h,00h,00      ;       ,108    ,6c
```

```
db      00h,70h,00h,70h,00h,00          ;"      ,109    ,6d
db      ffh,ffh,ffh,ffh,ffh,00          ;       ,110    ,6e
db      ffh,ffh,ffh,ffh,ffh,00          ;       ,111    ,6f
db      ffh,ffh,ffh,ffh,ffh,00          ;       ,112    ,70
db      ffh,ffh,ffh,ffh,ffh,00          ;       ,113    ,71
db      11h,13h,15h,19h,11h,00          ;z      ,114    ,72
db      11h,0ah,04h,0ah,11h,00          ;x      ,115    ,73
db      0eh,11h,11h,11h,11h,00          ;c      ,116    ,74
db      18h,06h,01h,06h,18h,00          ;v      ,117    ,75
db      7fh,11h,11h,11h,0eh,00          ;b      ,118    ,76
db      1fh,10h,10h,10h,0fh,00          ;n      ,119    ,77
db      0fh,10h,0fh,10h,0fh,00          ;m      ,120    ,78
db      08h,14h,22h,41h,00h,00          ;<      ,121    ,79
db      41h,22h,14h,08h,00h,00          ;>      ,122    ,7a
db      20h,40h,45h,48h,30h,00          ;?      ,123    ,7b
db      ffh,ffh,ffh,ffh,ffh,00          ;Shift  ,124    ,7c
db      00h,00h,00h,00h,00h,00          ;space  ,125    ,7d
```

ANEXO D

COSTO DEL SISTEMA

COSTO DE COMPONENTES

DISPLAY

Cant.	No del componente	Precio/unidad	Precio/total
1	GABINETE	$ 25.0	$ 25.0
4	NTE 4514B	$ 0.79	$ 3.16
10	NTE 2011	$ 1.460	$ 10.46
1	NTE 74LS139	$ 0.39	$ 0.39
420	LED rojos (5mm)	$ 0.07	$ 29.4
7	2N22	$ 0.12	$ 0.84
14	RESISTENCIAS 1/4 W	$ 1.0	$ 1.0
			$ 70.25

TECLADO

Cant.	No del componente	Precio/unidad	Precio/total
1	TECLADO	$ 12.0	$ 12.0
1	GABINETE	$ 12.0	$ 12.0
1	NTE 74LS150	$ 1.25	$ 1.25
1	NTE 74LS139	$ 0.39	$ 0.39
2	NTE 74LS163	$ 0.49	$ 0.98
1	NTE 74LS121	$ 0.29	$ 0.29
1	NTE 74LS08	$ 0.29	$ 0.29
1	NTE 74LS14	$ 0.39	$ 0.39
2	NE 555	$ 0.89	$ 1.78
1	NTE 74LS25	$ 0.39	$ 0.39
1	NTE 74LS76	$ 0.39	$ 0.39
1	AY-3-1015D (UART)	$ 4.95	$ 4.95
20	RESITENCIAS 1/4W	$ 1.0	$ 1.0
3	CAPACITORES	$ 0.12	$ 0.36
			$ 36.46

SISTEMA DE CONTROL

Cant.	No del componente	Precio/unidad	Precio/total
1	CPU 280A	$ 1.75	$ 1.75
1	ROM 2732A	$ 4.95	$ 4.95
1	RAM 6116	$ 2.25	$ 2.25
3	NTE 74LS245	$ 0.69	$ 2.07
2	NTE 74LS373	$ 0.69	$ 1.38
2	NTE 74LS32	$ 0.25	$ 0.5
1	NTE 74LS75	$ 0.29	$ 0.29

2	NTE 74LS14	$ 0.39	$ 0.78
1	NTE 74LS08	$ 0.29	$ 0.29
1	NTE 74LS21	$ 0.29	$ 0.29
1	NTE 74LS123	$ 0.49	$ 0.49
1	AY-3-1015D (UART)	$ 4.95	$ 4.95
1	AD 7820	$ 2.0	$ 2.0
2	NE 555	$ 0.89	$ 1.78
1	XTAL 3.1MHz	$ 2.0	$ 2.0
			$ 25.77

SUBTOTAL	$ 70.25
	$ 36.46
	$ 25.77
	$ 132.48
GASTOS VARIOS	$ 30.0
TOTAL	$ 132.0
	$ 30.0
	$ 162.0

ANEXO E

LISTADO DE PROGRAMAS

```
; *********     Programa principal   ********************************
;
            ld    sp,stack
;
            ld    a,1                   ;Correr mensaje de Rom y luego pare
            ld    (func),a              ;programa 1
            ld    (prog),a
            ld    a,0
            ld    (pcursor),a           ;No mostrar cursor
            ld    (rot1),a              ;No rotar patrones
            ld    (rot2),a
            ld    hl,video
            ld    (dvideo),hl           ;Comensar en video + 10
            ld    hl,video10
            ld    (ddato),hl
            ld    (max),hl              ;Direccion del ultimo dato
            ld    a,20                  ;Velocidad de corrimiento
            ld    (veloc),a
            ld    a,6                   ;Numero de patrones por caracter
            ld    (Npatrones),a
;
            ld    bc,1                ;Limpiar area de mensajes
            ld    hl,5ffh
            ld    de,video
            and   0                   ;Limpiar bandera de carri
            ld    a,61
blanco:     ld    (de),a
            inc   de
            sbc   hl,bc
            jr    nz,blanco
;
            ld    hl,mensaje1
            ld    de,video10
            ld    bc,14
            ldir
            ld    hl,24                 ;Tamano del buffer
            ld    (tbuffer),hl
;
            ld    hl,time               ;Datos iniciales de tiempo
            ld    de,counters
            ld    bc,7
            ldir
;
            im1                         ;Modo de interrupcion 1, RST 38H
            ei                          ;Habilitar interrupciones
;
inicio:     ld    d,0                   ;Contador de columnas
            ld    e,0                   ;Contador de caractores
            ld    hl,(dvideo)
salto1:     push  hl
            inc   e
            ld    a,(hl)                ;Busqueda de patron
            sla   a
            ld    b,a
            sla   a
            ld    h,0
            jr    nc,salto2
            ld    h,1
```

```
salto2:     add   a,b
            jr    nc,salto3
            inc   h
salto3:     ld    l,a
            ld    bc,patrones
            add   hl,bc                    ;Direccion del patron 1
            ld    a,(func)
            cp    5
            jr    z,salto10                ;Ignore estos patrones
            cp    3
            jr    z,salto10
;
            ld    b,6
            ld    a,e
            cp    1                        ;Es primer caracter ?
            jr    nz,salto5
            ld    a,(Npatrones)            ;Sacar  N patrones
            ld    b,a
            cp    6
            jr    z,salto5
            sub   a,6
salto4:     inc   hl                       ;Ignorar primeros patrones
            inc   a
            jr    nz,salto4
salto5:     ld    a,(rot1)                 ;Rotacion vertical 1 ?
            ld    c,a
            cp    0
            jr    z,salto6
            ld    a,(hl)
rote1:      sla   a
            dec   c
            jr    nz,rote1
            jr    salto8
salto1x:    jr    salto1
aux:        jr    inicio
salto6:     ld    a,(rot2)                 ;Rotacion vertical 2 ?
            ld    c,a
            cp    0
            jr    z,salto7
            ld    a,(hl)
rote2:      sra   a
            dec   c
            jr    nz,rote2
            jr    salto8
salto7:     ld    a,(pcursor)
            cp    e
            ld    a,(hl)
            jr    nz,salto8
            ld    a,01h                    ;Poner cursor (_)
salto8:     ld    c,7fh
            out   (c),a
            ld    c,d
            out   (c),a
            inc   d
            inc   hl
            dec   b
            ld    a,d
            cp    28
            jr    nz,salto9
            add   a,4
            ld    d,a
```

```
salto9:     cp      63                  ;Ultima columna
            jr      z,salto10
            ld      a,b
            cp      0
            jr      nz,salto5           ;Fin de caracter
            pop     hl
            inc     hl
            jr      salto1x
salto10:    pop     hl                  ;Limpiar stack
            ld      a,(func)            ;Comienza desplazamiento ?
            cp      1                   ;Si es cero no desplase
            call    z,func1             ;Desplazamiento horizontal
            cp      3
            call    z,func3             ;Caida vertical
            cp      4
            call    z,func4             ;Desplazamiento vertical
            cp      5
            call    z,func5             ;Desvanecimiento
            jr      aux
```

```
;****** Rutina de desplazamiento horizontal, funcion 1 ********
;
func1:      push  af
            ld    a,(veloc)
            dec   a
            ld    (veloc),a
            jr    nz,finx
            ld    a,15
            ld    (veloc),a
            ld    a,(Npatrones)
            dec   a
            ld    (Npatrones),a
            jr    nz,finx
            ld    a,6
            ld    (Npatrones),a
            ld    hl,(dvideo)
            inc   hl
            ld    (dvideo),hl
            and   0                      ;Limpia bandera de carri
            ld    bc,1
            ld    hl,(tbuffer)
            sbc   hl,bc
            ld    (tbuffer),hl
            jr    nz,finx                ;Termino de desplazar mensaje ?
            ld    a,(prog)               ;Que desplaza, hora,fecha,mensaje
            cp    1
            jr    nz,cont1
            call  cls
            call  hora
            ld    hl,1014h
            ld    (max),hl
            ld    a,0
            ld    (func),a
            ld    hl,video10                 ;Inicialize variables
            ld    (dvideo),hl
            ld    (ddato),hl
            ld    a,1
            ld    (pcursor),a
            jr    finx
cont1:      cp    2
            jr    nz,cont2
            call  cls
            call  fecha
            ld    a,0
            ld    (func),a
            ld    hl,video10                 ;Inicialize variables
            ld    (dvideo),hl
            ld    (ddato),hl
            ld    a,1
            ld    (pcursor),a
finx:       jr    fin
```

```
cont2:      cp    3
            jr    nz,cont3
            call  cls
            ld    hl,bvideo                ;Recupera mensaje escrito
            ld    (dvideo),hl
            ld    (ddato),hl
            ld    a,1
            ld    (pcursor),a
            ld    a,0
            ld    (func),a                 ;Detiene corrimiento de mensaje
            jr    fin
cont3:      cp    4
            jr    nz,cont4
            call  cls
            call  hora
            ld    hl,video10
            ld    (dvideo),hl
            ld    (ddato),hl
            ld    a,3
            ld    (func),a
            jr    fin
cont4:      cp    5
            jr    nz,cont5
            call  cls
            call  fecha
            ld    hl,video10
            ld    (dvideo),hl
            ld    a,4
            ld    (func),a
            ld    a,1
            ld    (part),a
            ld    a,7
            ld    (rot1),a
            ld    a,0
            ld    (rot2),a
            jr    fin
cont5:      call  cls
            call  hora
            ld    a,3
            ld    (func),a
fin:        pop   af
            ret
```

```
;****** Rutina para generar funcion 2 ***************
;       "caida vertical"
;
func3:      push  af
            ld    a,1
            ld    (1600h),a
            ld    (1602h),a
            ld    a,0
            ld    (stop),a              ;Teclado desactivado
            ld    e,64
            ld    ix,00
;
inicio:     ld    d,0
            ld    hl,video10
salto2:     push  hl
            ld    a,(hl)
            sla   a                     ;Busqueda del patron
            ld    b,a
            sla   a
            ld    h,0
            jr    nc,sig1
            ld    h,1
sig1:       add   a,b
            jr    nc,sig2
            inc   h
sig2:       ld    l,a
            ld    bc,patrones
            add   hl,bc
;
            ld    b,6
salto3:     ld    c,7fh
            ld    a,(1602h)
            cp    1
            jr    z,one
            cp    2
            jr    z,two
            cp    3
            jr    z,three
            cp    4
            jr    z,four
            cp    5
            jr    z,five
            cp    6
            jr    z,six
            cp    7
            jr    z,seven
;
one:        ld    a,(hl)
            and   0
            jr    salida
salto21:    jr    salto2
salto31:    jr    salto3
princi:     jr    inicio
two:        ld    a,(hl)
            and   1
            jr    salida
three:      ld    a,(hl)
            and   3
            jr    salida
```

```
four:       ld    a,(hl)
            and   7
            jr    salida
five:       ld    a,(hl)
            and   15
            jr    salida
six:        ld    a,(hl)
            and   31
            jr    salida
seven:      ld    a,(hl)
            and   63
;
salida:     ld    (1604h),a
            ld    a,(1600h)
            cp    7
            jr    z,siete
            cp    6
            jr    z,seis
            cp    5
            jr    z,cinco
            cp    4
            jr    z,cuatro
            cp    3
            jr    z,tres
            cp    2
            jr    z,dos
            ld    a,(hl)
            rrc   a
            rrc   a
            jr    comun
inter:      jr    salto21
salt:       jr    salto31
;
siete:      ld    a,(hl)
            jr    comun
seis:       ld    a,(hl)
            rlc   a
            jr    comun
cinco:      ld    a,(hl)
            rlc   a
            rlc   a
            jr    comun
cuatro:     ld    a,(hl)
            rlc   a
            rlc   a
            rlc   a
            jr    comun
tres:       ld    a,(hl)
            rrc   a
            rrc   a
            rrc   a
            rrc   a
            jr    comun
inic:       jr    princi
dos:        ld    a,(hl)
            rrc   a
            rrc   a
            rrc   a
```

```
comun:  and   e
        ld    (1612h),a
        ld    a,e
        ld    (1606h),a
        ld    a,(1604h)
        ld    e,a
        ld    a,(1612h)
        or    e
        out   (c),a
        ld    a,(1606h)
        ld    e,a
        inc   hl
        dec   b
        ld    c,d
        out   (c),a
        inc   d
        ld    a,d
        cp    28
        jr    nz,siga
        add   a,4
        ld    d,a
siga:   ld    a,d
        cp    63
        jr    z,fin
        ld    a,b
        cp    0
        jr    nz,salt
        pop   hl
        inc   hl
        jr    inter
fin:    pop   hl
        ld    a,(stop)          ;Salir si hay stop
        cp    1
        jr    nz,cont
        pop   af
        ret
cont:   inc   ix
        ld    (1608h),ix
        ld    a,(1608h)
        cp    5
        jr    nz,inic
        ld    ix,00
        rrc   e
;
        ld    a,(1600h)
        inc   a
        ld    (1600h),a
        cp    8
        jr    z,reset
        jr    inic
;
reset:  ld    e,64
        ld    a,(1602h)
        inc   a
        ld    (1602h),a
        ld    (1600h),a
        cp    8
        jr    nz,inic
```

```
ld      a,1
ld      (func),a
ld      a,5
ld      (prog),a
ld      a,200
ld      (veloc),a
ld      hl,10
ld      (tbuffer),hl
ld      hl,video10
ld      (dvideo),hl
pop     af
ret
```

```
;******* Rutina de corrimiento vertical funcion 3 *****
;
vert:       push    af
            ld      a,(veloc)
            dec     a
            ld      (veloc),a
            jr      nz,fin
            ld      a,15
            ld         (veloc),a                    ;Velocidad de
corrimiento
            ld      a,(part)
            cp      2
            jr      z,sig
            ld      a,(rot1)
            dec     a
            ld      (rot1),a
            cp      0
            jr      nz,fin
            ld      a,2
            ld      (part),a
            ld      a,255
            ld      (veloc),a
            jr      fin
sig:        ld      a,(rot2)
            inc     a
            ld      (rot2),a
            cp      7
            jr      nz,fin
            ld      a,0
            ld      (rot2),a
            call    cls
            call    temp
            ld      a,5
            ld      (func),a
fin:        pop     af
            ret
            ld     hl,video10
            ld     (dvideo),hl
            pop    af
            ret
```

```
;******** Funcion 5 ***********
;
func5:      push   af
            ld     hl,300
            ld     (1610h),hl
            ld     a,0
            ld     (1602h),a
            ld     (1612h),a
            ld     (stop),a
            ld     e,0
            ld     ix,0
inicio:     ld     d,0
            ld     hl,video10
salto2:     push   hl
            ld     a,(hl)
            sla    a                ;Busqueda del patron
            ld     b,a
            sla    a
            ld     h,0
            jr     nc,sig1
            ld     h,1
sig1:       add    a,b
            jr     nc,sig2
            inc    h
sig2:       ld     l,a
            ld     bc,patrones
            add    hl,bc
;
            ld     b,6
salto3:     ld     c,7fh
            ld     a,(hl)
            and    e
            out    (c),a
            inc    hl
            dec    b
            ld     c,d
            out    (c),a
            inc    d
            ld     a,d
            cp     28
            jr     nz,siga
            add    a,4
            ld     d,a
siga:       ld     a,d
            cp     63
            jr     z,fin
            ld     a,b
            cp     0
            jr     nz,salto3
            pop    hl
            inc    hl
            jr     salto2
iniciox:    jr     inicio
fin:        pop    hl
            ld     a,(stop)
            cp     1
            jr     nz,cont
            pop    af
            ret
```

```
cont:   ld    a,(1612h)
        cp    0
        jr    z,cont2
        and   0                    ;limpiar carri
        ld    bc,1
        ld    hl,(1610h)
        sbc   hl,bc
        ld    (1610h),hl
        jr    nz,inicio
        ld    a,0
        ld    (1612h),a
cont2:  inc   ix
        ld    (1600h),ix
        ld    a,(1602h)
        cp    1
        jr    z,desap
        ld    a,(1600h)
        cp    30
        jr    z,pat1
        cp    60
        jr    z,pat2
        cp    90
        jr    z,pat3
        cp    120
        jr    z,pat4
        cp    150
        jr    z,pat5
        cp    180
        jr    z,pat6
        cp    210
        jr    z,pat7
        cp    255
        jr    z,regres
        jr    inter
inter:  jr    inicio
pat1:   ld    a,e
        add   a,1
        ld    e,a
        jr    inter
pat2:   ld    a,e
        add   a,2
        ld    e,a
        jr    inter
pat3:   ld    a,e
        add   a,4
        ld    e,a
        jr    inter
pat4:   ld    a,e
        add   a,8
        ld    e,a
        jr    inter
pat5:   ld    a,e
        add   a,16
        ld    e,a
        jr    inter
pat6:   ld    a,e
        add   a,32
        ld    e,a
        jr    inter
```

```
pat7:       ld      a,e
            add     a,64
            ld      e,a
            jr      inter
regres:     ld      a,1
            ld      (1612h),a
            ld      ix,00
            ld      a,1
            ld      (1602h),a
            jr      inter
;
desap:      ld      a,(1600h)
            cp      30
            jr      z,rote
            cp      60
            jr      z,rote
            cp      90
            jr      z,rote
            cp      120
            jr      z,rote
            cp      150
            jr      z,rote
            cp      180
            jr      z,rote
            cp      210
            jr      z,rote
            cp      255
            jr      z,comien
            jr      inter
;
rote:       sla     e
            jr      inter
;
comien:     ld      a,1
            ld      (func),a
            ld      a,6
            ld      (prog),a
            and     0                   ;Limpiar carri
            ld      bc,10
            ld      hl,bvideo
            sbc     hl,bc
            ld      (dvideo),hl          ;Direccion del  mensaje a
mostrar
            ld      hl,(buffer)
            add     hl,bc
            ld      (tbuffer),hl
            pop     af
            ret
```

```
;******* Rutina de Interrupción para el servicio de hora *********
;
nmi:        push  hl
            push  bc
            push  af
            ld    b,60h
            ld    hl,counters
            call  update        ;segundos
            jr    nz,done
            call  update        ;minutos
            jr    nz,done
            ld    b,13h
            call  update        ;horas
            jr    z,cont
            ld    a,(hl)
            cp    12h
            jr    z,cont1
            jr    done
cont2:      dec   hl
            call  update        ;Poner 1 hora "correccion de la cero horas
            inc   hl
            jr    done
cont1:      inc   hl
            ld    b,3
            call  update        ;Fraccion de dia
done:       pop   af
            pop   bc
            pop   hl
            retn
;
;******* Rutina para desplegar fecha ,que se encuentra en los contadores
;
fecha:       ld     ix,counters
             ld     iy,video10
             ld     c,3
start:       ld     a,(ix+4)            ;Traer dia
             srl    a
             srl    a
             srl    a
             srl    a
             inc    a
             cp     1
             jr     nz,sig
             ld     a,11
sig:         ld     (iy+1),a
             ld     a,(ix+4)
             and    0fh
             inc    a
             cp     1
             jr     nz,sig2
             ld     a,11
sig2:        ld     (iy+2),a
             dec    c
             ret    z                   ;Terminar si c = 0
             ld     (iy+3),13           ;Poner '-' entre numeros
             inc    ix
             inc    iy
             inc    iy
             inc    iy
             jr     start
```

```
;******Rutina para desplegar hora de los contadores *********
;
Hora:       ld    ix,counters
            ld    iy,video10
            ld    c,2
next:       ld    a,(ix+2)        ;Traer hora actual
            srl   a
            srl   a
            srl   a
            srl   a
            inc   a
            cp    1               ;Si primer digito es cero poner spc
            jr    nz,salto
            ld    a,c
            cp    2
            ld    a,11
            jr    nz,salto        ;Es c =/ de cero
            ld    a,61            ;Poner espacio en blanco
salto:      ld    (iy+1),a        ;Imprimir en segunda posicion de pantalla
            ld    a,(ix+2)
            and   0fh
            inc   a
            cp    1
            jr    nz,sig
            ld    a,11
sig:        ld    (iy+2),a        ;Imprimir en tercera posicion de pantalla
            dec   c
            jr    z,meridiano
            ld    (iy+3),108      ;pone ":" entre hora y minutos
            dec   ix
            inc   iy
            inc   iy
            inc   iy
            jr    next
meridiano:  ld    a,(ix+4)
            cp    1                         ;AM o PM  ?
            jr    nz,pm
am:         ld    (iy+4),34   ;A
            ld    (iy+5),56   ;M
            jr    siga
pm:         ld    (iy+4),27   ;P
            ld    (iy+5),56   ;M
siga:       ret
;
;*******  Rutina para actualizar contadores de hora y fecha
;
update:     ld    a,(hl)
            inc   a
            daa
            ld    (hl),a
            sub   b
            ret   nz
            ld    (hl),a
            inc   hl
            ret
```

```
;****** Rutina para guardar hora de video en contadores ********
;
storeh:     ld      bc,1                ;Empaquetar posicion 1 y 2 de video
            ld      de,2                ;en counters 3
            call    empaq
            ld      bc,4
            ld      de,1
            call    empaq
            ld      ix,video10
            ld      iy,counters
            ld      a,(ix+7)
            cp      27                  ;Es PM?
            jr      nz,AM
            ld      (iy+3),2            ;Pongale PM
            jr      siga
AM:         ld      (iy+3),1            ;Pongale AM
siga:       ret
;
;*******   Rutina para guardar fecha de video en contadores ********
;
storef:     ld      bc,1
            ld      de,4
            call    empaq
            ld      bc,4
            ld      de,5
            call    empaq
            ld      bc,7
            ld      de,6
            call    empaq
            ret
;
;   ******** Rutina de empaquetado **********
;
empaq:      ld      ix,video10
            ld      iy,counters
            add     ix,bc               ;Sumar el valor de BC y DE dado al inici
            add     iy,de               ;de la rutina a video y a counters
            ld      a,(ix)
            dec     a                   ;Correccion del dato
            cp      10
            jr      nz,cont1
            ld      a,0
            jr      cont2
cont1:      cp      60
            jr      nz,cont2
            ld      a,0
cont2:      sla     a
            sla     a
            sla     a
            sla     a
            ld      b,(ix+1)
            dec     b                   ;Correccion del dato
            ld      c,a
            ld      a,b
            cp      10
            jr      nz,sig2
            ld      b,0
sig2:       ld      a,c
            or      a,b
            ld      (iy),a
            ret
```

```
;****** Rutina de Interrupcion de teclado
;
inicio:     push  af
            push  bc
            push  de
            push  ix
            push  iy
            in    a,(00)              ;Lectura de dato
            ld    b,a
            cp    28                  ;Cursor izquierdo ?
            jr    nz,salto1
            call  CurLefth
            jr    salir
salto1:     cp    92
            jr    nz,salto2
            call  CurLefth
            jr    salir
salto2:     cp    29                  ;Cursor derecho ?
            jr    nz,salto3
            call  CurRight
            jr    salir
salto3:     cp    93
            jr    nz,salto4
            call  CurRight
            jr    salir
salto4:     cp    60                  ;BackSpace ?
            jr    nz,salto5
            call  BackSpc
            jr    salir
salto5:     cp    124
            jr    nz,salto6
            call  BackSpc
            jr    salir
salto6:     cp    45                  ;Enter ?
            jr    nz,salto7
            call  Enter
            jr    salir
salto7:     cp    109
            jr    nz,salto8
            call  Enter
salir       jr    salirx
salto8:     cp    49                  ;Edit ?
            jr    nz,salto9
            call  edit
            jr    salirx
salto9:     cp    113
            jr    nz,salto10
            call  edit
            jr    salirx
salto10:    cp    33                  ;Delete ?
            jr    nz,salto11
            call  delete
            jr    salirx
salto11:    cp    97
            jr    nz,salto12
            call  delete
            jr    salirx
```

```
salto12:    cp    17                    ;Insert ?
            jr    nz,salto13
            call  insert
            jr    salirx
salto13:    cp    81
            jr    nz,salto14
            call  insert
            jr    salirx
salto14:    cp    14                    ;Reset ?
            jr    z,salirx
            cp    78
            jr    z,salirx
            cp    1                     ;Caps  ?
            jr    z,salirx
            cp    65
            jr    z,salirx
;
salto16:    ld    hl,(ddato)            ;Escritura de dato leido
            ld    (hl),b
            call  CurRight
salirx:     pop   iy
            pop   ix
            pop   de
            pop   bc
            pop   af
            ei
            reti
```

```
;******** Rutina para funcion delete **********
;
delete:     ld  hl,(max)          ;Restar (max) - (ddato)
            ld  de,(ddato)
            and 0                 ;Limpiar carri
            sbc hl,de             ;a = l - e
            ld  b,h
            ld  c,l               ;Tamaño del buffer a mover
            ld  hl,(ddato)
            inc hl                ;Direccion fuente
            ld  de,(ddato)        ;Direccion destino
            ldir                  ;Mover el bloque de datos
            ld  hl,(max)
            dec hl
            ld  (max),hl          ;Decremente tamaño del buffer
            ld  (hl),61           ;Borre ultimo caracter
            ret
;
; ****** Rutina para funcion insert **************
;
insert:     ld  hl,(max)          ;Restar (max) - (ddato)
            ld  de,(ddato)
            and 0                 ;Limpiar carri
            sbc hl,de             ;a = l - e
            inc hl                ;a = a + 1
            ld  b,h
            ld  c,l               ;Tamaño del buffer a mover
            ld  hl,(max)          ;Direccion fuente
            ld  de,(max)
            inc de                ;Direccion destino
            lddr                  ;Mover el bloque de datos
            ld  hl,(ddato)
            ld  (hl),61           ;Borre dato del cursor
            ld  hl,(max)
            inc hl
            ld  (max),hl          ;Incremento direccion maxima de datos
            ret
;
;******** Rutina  de tecla de retroceso (Back space)
;
BackSpc:      ld   hl,(ddato)
              ld   (hl),61          ;Borre caracter actual
              ld   de,video10       ;Es la direccion del dato igual-
              and  0                ;a la direccion inicial de video
              sbc  hl,de
              jr   z,salir3         ;Si es = no retroceder
              ld   hl,(ddato)
              dec  hl
              ld   (ddato),hl
              ld   a,(pcursor)
              dec  a
              ld   (pcursor),a
              cp   0                ;Posicion izquierda?
              jr   nz,salir3
              inc  a
              ld   (pcursor),a
              ld   hl,(dvideo)
              dec  hl
              ld   (dvideo),hl
salir3:       ret
```

```
;******* Rutina para mover cursor a la izquierda
;
CurLefth:       ld   de,video10        ;Esta en posicion izquierda de buffer ?
                ld   hl,(ddato)
                and  0                 ;limpiar carri
                sbc  hl,de
                jr   z,salir1          ;Si es = no retroceder
                ld   hl,(ddato)        ;Incremente direccion del dato
                dec  hl
                ld   (ddato),hl
                ld   a,(pcursor)       ;Incremente posicion del cursor
                dec  a
                ld   (pcursor),a
                cp   0
                jr   nz,salir1
                inc  a                   ;Poner en posicion 1 el cursor
                ld   (pcursor),a
                ld   hl,(dvideo)
                dec  hl
                ld   (dvideo),hl
salir1:         ret
;
;******** Rutina para mover cursor a la derecha
;
CurRight:       ld   hl,(ddato)
                inc  hl
                ld   (ddato),hl
                and  0                 ;Limpiar carri
                ld   de,(max)
                sbc  hl,de
                jr   c,sig             ;Si (ddato) > (max) ==> (max) = (ddato)
                ld   hl,(ddato)
                ld   (max),hl
sig:            ld   a,(pcursor)
                inc  a
                ld   (pcursor),a
                cp   11
                jr   nz,salir2
                dec  a
                ld   (pcursor),a       ;Poner en posicion 10 el cursor
                ld   hl,(dvideo)
                inc  hl
                ld   (dvideo),hl
salir2:         ret
```

```
;*******Rutina de terminacion de mensaje (Enter) ********
;
Enter:      ld    a,(prog)
            cp    1
            jr    nz,next1
            call  storeh          ;Guardar hora
            call  cls
            ld    hl,mensaje2     ;Escribir mensaje
            ld    de,video10
            ld    bc,15
            ldir
            ld    a,25
            ld    (tbuffer),a
            ld    a,2             ;Siguiente entrada fecha
            ld    (prog),a
            ld    a,1
            ld    (func),a
            ld    a,0             ;Elimine cursor
            ld    (pcursor),a
            ld    hl,video
            ld    (dvideo),hl     ;Inicializa (dvideo)
            jr    salir
next1:      cp    2
            jr    nz,next2
            call  storef          ;Guardar fecha
            call  cls
            ld    hl,mensaje3     ;Escribir mensaje3
            ld    de,video10
            ld    bc,17
            ldir
            ld    a,27
            ld    (tbuffer),a
            ld    a,1
            ld    (func),a
            ld    a,3             ;Siguiente entrada,mensaje
            ld    (prog),a
            ld    a,0
            ld    (pcursor),a
            ld    hl,video
            ld    (dvideo),hl
            jr    salir
next2:      ld    hl,(max)        ;max-video
            ld    de,bvideo
            and   0               ;limpiar carri
            sbc   hl,de
            ld    (buffer),hl      ;Tamano del mensaje
            ld    a,(pcursor)     ;Desplazamiento inicial
            ld    (tbuffer),a
            ld    a,0
            ld    (pcursor),a     ;Desaparecer cursor
            ld    a,4
            ld    (prog),a
            ld    a,1             ;Corrimiento de mensajes
            ld    (func),a
salir:      ret
```

```
;****** Rutina para detener corrimiento ***************
;
Edit:           ld    a,1
                ld    (stop),a
                call  cls
                ld    hl,mensaje1
                ld    de,video10
                ld    bc,14
                ldir
                ld    a,24
                ld    (tbuffer),a
                ld    a,1
                ld    (func),a          ;Detener corrimiento
                ld    (prog),a
                ld    hl,video
                ld    (dvideo),hl
                ld    (ddato),hl
                ret
```