10 20

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA DE CONTROL DE PROPOSITO GENERAL Y UNA APLICACION DE ADQUISICION DE DATOS PARA EL CONTROL DE PERSONAL

T E S I S

QUE PARA OBTENER EL TITULO DE
INGENIERO MECANICO ELECTRICISTA
P R E S E N T A N

FERNANDO ARAMBULA COSIO
EDGAR MANDUJANO ESCOBAR
ENRIQUE MARTINEZ FLORES

Director de Tesis:
ING. MAURICIO GARCIA ESTEBAN

MEXICO, D. F. 1990

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

CAPITULO I

DISEÑO CONCEPTUAL

I.1.- ANTECEDENTES

Este trabajo consiste en el desarrollo de un equipo para adquirir y almacenar datos digitales en un medio no volátil, con capacidad para controlar dispositivos externos y para comunicarse con otros equipos similares o con una computadora PC. Surge del interés por parte de la sección de electrónica del Centro de Instrumentos en contar con un equipo de automatización de bajo costo, tecnológicamente factible, portátil y de aplicación general para emplearlo como un módulo funcional en el desarrollo de este tipo de sistemas.

Algunos de los campos donde puede aplicarse son: la adquisición de datos en tiempo real en la medición de procesos químicos, llevando el registro de variables como temperatura, ph, oxigeno disuelto, etc. En el caso de que el instrumento que se utilize para realizar las mediciones anteriores sea del tipo analógico, es necesario realizar una conversión analógico a digital para que el equipo pueda procesar los datos que se generen, aunque existen equipos comerciales que evalúan éstas y otras variables y que cuentan con salida de datos en forma digital. Debido a que el equipo cuenta con medios para habilitar actuadores puede emplearse en el control de variables en algún proceso y puede comunicarse con otros equipos similares para implementar un control distribuido.

La aplicación que se dá al equipo en este trabajo es en la captura de datos desde un reloj checador. Esta aplicación surgió de la petición por parte de la misma Universidad al Centro de Instrumentos para desarrollar un sistema automático que contabilice las horas trabajadas por el personal de una

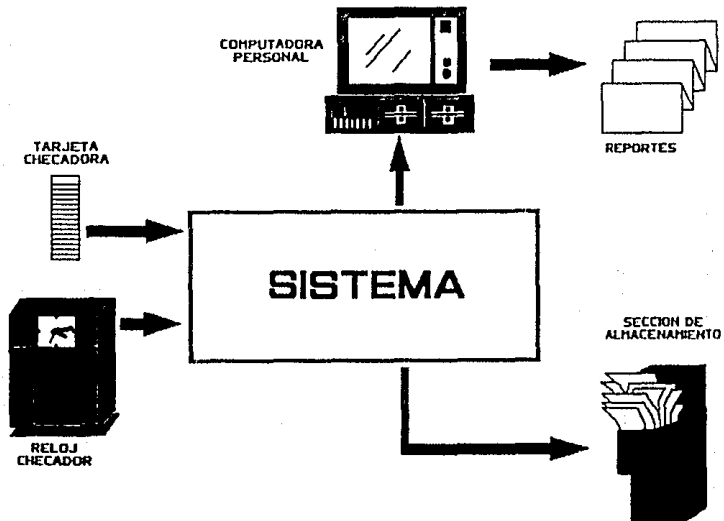


FIG. I.1

dependencia. Esta tarea se hace manualmente en base a las horas marcadas en la tarjeta de checar para calcular los descuentos por retardos y/o el pago de tiempo extra de cada persona. Puede verse que cuando es grande el número de empleados en la dependencia, la tarea anterior consume mucho tiempo e induce a errores, que la implantación del sistema automático reducirá a un mínimo. Un esquema básico de este proyecto se muestra en la fig.I.1.

I.2.- DISEÑO PRELIMINAR

Durante esta etapa determinamos junto con los solicitantes las especificaciones básicas de operación del equipo, para aplicarse en el sistema de captura de datos desde el reloj checador, así como los medios con que debe contar para ser un equipo de aplicación general.

Se solicitó que el sistema de adquisición no interfiriera con el proceso convencional de checar, en el que cada empleado tiene asignada una tarjeta de registro en la que el reloj checador imprime las horas de entrada y de salida durante 15

dias, de este modo la persona cuenta con un medio de comprobación. Por lo anterior, se conservó el uso de la tarjeta de checar tradicional y del reloj checador convencional. Debe ser capaz de almacenar por lo menos, la información correspondiente a 500 personas durante 7 dias mínimo. Se consideraron 500 personas basándonos en la cantidad de personal que labora en la mayoría de las dependencias de la U.N.A.M., y los 7 dias son el periodo mínimo que se consideró adecuado para vaciar la información, ya que un periodo menor seria ineficiente por el tiempo dedicado a ello. Y un periodo mayor, aunque más eficiente, implica una mayor capacidad de almacenamiento y por ende, mayor costo; lo que contribuiría a que el producto final sea menos competitivo.

Enfocado para cubrir la necesidad ya planteada, el proyecto debe ser capaz de registrar la hora y fecha en que se da un evento, siendo este evento, la entrada o salida de cada usuario; entendiéndose por usuario a cada persona que checa su tarjeta. Tiene que almacenar la información de manera confiable, además de poder comunicarse con una microcomputadora tipo PC para que la información se procese en una base de datos, con el fin de tener un archivo de control de horas trabajadas por el personal, de donde se pueden generar los reportes requeridos, como pueden ser número de retardos, faltas y cálculo de tiempo extra.

Como sistema debe ser amigable al usuario, es decir, de operación sencilla y proporcionando toda la información necesaria para que este se de cuenta de la operación que se está realizando, por otra parte basamos el diseño principalmente en componentes de fácil adquisición en el mercado Mexicano, para disminuir el costo y los problemas de producción por falta de proveedores nacionales.

La opción que proponemos cuenta con una gran capacidad de direccionamiento de memoria , con dispositivos periféricos de entrada/salida que permiten una comunicación via puerto serie ó paralelo con una microcomputadora, cuenta también con un controlador de teclado y despliegue que permite la interacción con el usuario, así como un controlador de interrupciones que atiende en forma jerarquizada las peticiones de atención y las envía al microprocesador encargado de controlar el sistema, lo que le da gran diversidad de aplicaciones en distintas áreas que dependen cada vez mas de lo práctico y seguro del procesamiento digital de datos. Todo implementado en un equipo de tamaño reducido y muy funcional. La figura I.2 muestra el diagrama a bloques de la operación de este proyecto en la aplicación planteada.

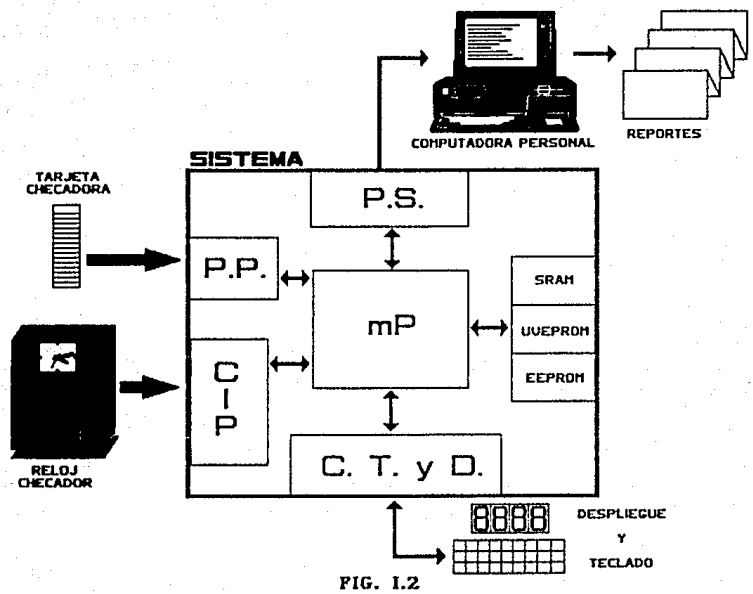

COMPUTADORA PERSONAL
REPORTES
SISTEMA
TARJETA CHECADORA
P.S.
P.P.
mP
SRAM
UVEPROM
EEPROM
C I P
C. T. y D.
RELOJ CHECADOR
DESPLIEGUE Y TECLADO

FIG. I.2

CAPITULO II

ESPECIFICACIONES DE DISEÑO

Introducción.

En este capitulo hablaremos de las características particulares que se le dieron al sistema de adquisición para cumplir con los requerimientos del proyecto, tales como la forma de distinguir un usuario de otro, flujo que se le da a la información generada, donde se guarda y que capacidad se tiene para almacenarla, así como características básicas de funcionamiento y diseño que le permitan un buen desempeño en el campo de aplicación.

Se buscó la manera de que a partir de la misma tarjeta de checar se distinguiera a todos y cada uno de los usuarios. Así, una vez identificados, sabrá en donde almacenar los datos que genere en el instante que se interactúe con él, y mostrar la información correspondiente para que pueda ser verificada.

Por otra parte, considerando el número de usuarios, la información que se guarda y el número de dias a cubrir se determina la cantidad de memoria a utilizar, la que deberá cumplir con ciertos requerimientos de seguridad con el objeto de no perder información ya almacenada.

En cuanto a características funcionales, se considera que una comunicación fácil y explícita entre el personal y el sistema es fundamental para su aceptación por los usuarios y para su

confiable operación, por lo que contará con un despliegue que muestre toda la información que el usuario requiera, con un teclado numérico y de funciones especiales, así como con un conector para comunicarse con una microcomputadora tipo PC mediante el puerto serie y que la información almacenada pueda ser procesada; además de que el diseño de su gabinete permita buena visibilidad, funcionalidad y seguridad.

II.1.- METODO DE ADQUISICION.

Esta parte tiene especial atención, ya que debíamos obtener la información para identificar a cada usuario a partir de las tarjetas de checar estándar y por otro lado se buscó que quien adquiriera este sistema tuviera la posibilidad de marcar el mismo en las tarjetas las claves de identificación de cada usuario, lo que da gran flexibilidad en el uso del equipo. Se implementó un medio de identificación que no requiere de tecnologías de costo elevado (como en el caso de las tarjetas con banda magnética), para fijarse en las tarjetas de cartoncillo, consiste en sensar un número en código marcado en la tarjeta de checar utilizando un detector óptico que nos permite, con mínimos cambios en dicha tarjeta, registrar con seguridad el número de identificación de cada usuario.

Para esta aplicación se utilizan dos sensores ópticos, uno dedicado a enviar pulsos de referencia al sistema y el otro asignado a detectar el bit correspondiente del código en binario natural del número de identificación.

Dicho código se marcará a lo largo de los lados mas grandes de la tarjeta; hecho a partir de dos hileras de marcas obscuras y espacios claros, siendo una la señal de referencia y otra la de datos, correspondiendo una barra obscura a un "1" lógico y un espacio claro a un "0" lógico.

La forma de leer el número de identificación es la siguiente: al deslizar la tarjeta en el reloj para realizar la acción de checar, un sensor detecta una primera franja obscura en la hilera de marcas de referencia y manda una interrupción al sistema con la cual se le indica que lea el bit que el otro sensor esté enviando, al seguir deslizando detecta la siguiente franja obscura y lee el bit correspondiente, así sucesivamente hasta que se leen los 9 bits que forman el número de identificación. Al deslizarse la tarjeta hacia afuera del reloj se realiza la misma acción, es decir, se vuelve a leer el número de identificación y se compara con el que se obtuvo primero, de ser iguales, el sistema muestra durante unos segundos el número decimal correspondiente en un sistema de despliegue para que pueda ser verificado por el usuario y la información corrrespondiente a ese instante sea almacenada en las localidades de memoria adecuadas. De no coincidir las lecturas, se deshecha el número adquirido y se espera a que se vuelva a introducir la tarjeta.

II.2.- MANEJO DE LOS DATOS ADQUIRIDOS

Para llevar un control claro y completo, la información que debe almacenarse para cada usuario es: hora de entrada y salida por turno, y fecha.

El esquema que manejamos para lograrlo fué el de utilizar el número de identificación como un apuntador hacia una localidad de memoria cuya ubicación está determinada por las condiciones que se tengan, que pueden ser: entrada o salida, turno (mañana, tarde o extraordinario) y el número de día.

El número de día se lleva en un contador que se incrementa diariamente (cada 24 horas) y se inicializa en el primer día cada vez que se vacía la información almacenada en el equipo. Este contador es equivalente a la fecha si se toma nota de la misma cada vez que se vacía la memoria.

En la localidad determinada por las condiciones de entrada/salida, turno, número de día, se almacena la hora vigente en el reloj de tiempo real interno, del sistema y en el byte siguiente los minutos.

El reloj interno de tiempo real se genera a partir de una señal de 1 Hz que se atiende como interrupción actualizando contadores que llevan la cuenta del número de horas, minutos y segundos en localidades de memoria. A continuación se detalla el método de almacenamiento:

El número de identificación en la tarjeta de checar se codifica en binario natural con 9 bits, así tenemos 512 posibles números que son:

BINARIO		HEXADECIMAL		DECIMAL
0 0000 0000	ó	000H	ó	000
0 0000 0001		001H		001
. ...		....		...
. ...		....		...
1 1111 1111		1FFH		512

Asignamos a cada usuario un bloque de memoria donde se almacenan sus datos, necesitando entonces 512 bloques. Considerando posibles retrasos en el periodo de 7 días para vaciar la información acumulada por el equipo, se instaló capacidad para capturar la información durante 10 días, esto proporciona tres días extras para vaciarla, cargándola en la PC, sin que se interrumpa el proceso de captura. Entonces cada bloque asignado a un usuario consta de 10 días; y en la sección de memoria correspondiente a un día, el equipo tiene el espacio necesario para almacenar las horas de entrada y salida de tres turnos. En el registro de entrada o salida utilizamos 2 bytes como se mencionó, uno contiene la hora y otro los minutos, como son tres turnos y cada uno tiene hora de entrada y de salida necesitamos , entonces, 12 bytes por día. En las figuras II.1.a

y II.1.b se ilustra el método descrito.

Nótese que sólo se almacenan la hora y minutos, mientras el resto de la información está determinado por la posición que ocupan dentro del esquema anterior.

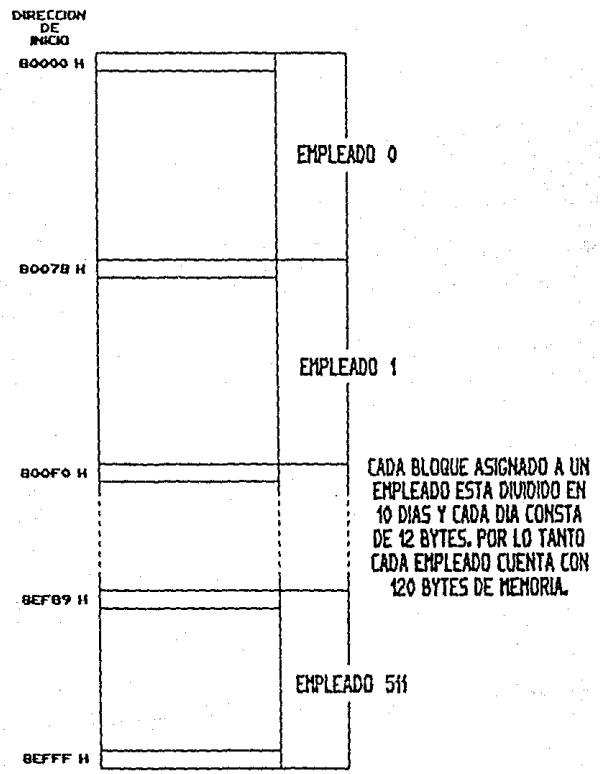


FIG. II.1.a

II.3.- TIPO Y CAPACIDAD DE MEMORIA DE ALMACENAMIENTO.

Para tener una máxima protección, los datos capturados se almacenaron en memorias de tipo EEPROM. Seleccionamos en particular la 2864A por sus características eléctricas y de

temporización. Es una memoria de 8 Kbytes x 8 bits que opera con sólo una fuente de 5 volts, con tiempos de acceso de lectura de 250 ns y de escritura muy similares a los de una SRAM. El ciclo de escritura no volátil se temporiza internamente manteniendo la dirección y el dato, dejando así, libre el bus del sistema durante el período de escritura. La 2864A incorpora un ciclo de borrado por byte, automático y transparente al diseñador, durante la operación de escritura por byte, completando el ciclo de borrado y escritura en un máximo de 10 ms. Cuenta con tres mecanismos de protección contra falsas

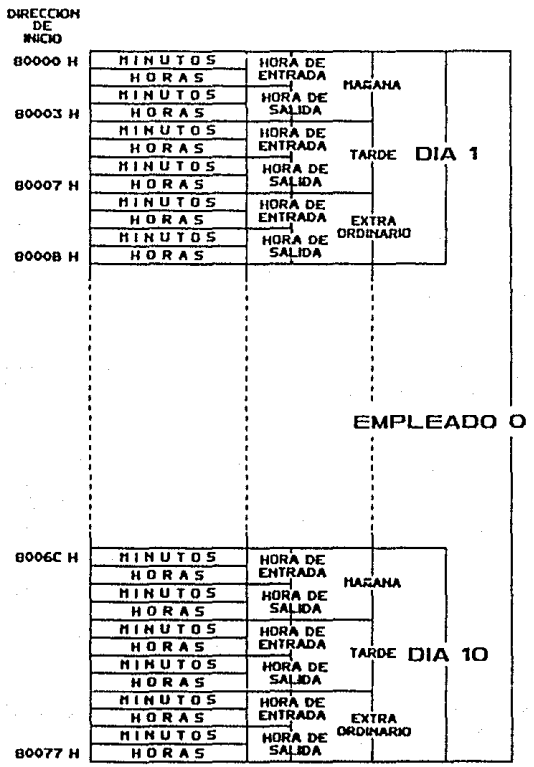


FIG. II.1.b

señales de escritura generadas por transitorios en la alimentación.

La capacidad de memoria instalada está en función del número de usuarios así como del manejo de datos descrito en el inciso

anterior, entonces:

TOTAL DE BYTES = BYTES POR DIA X NUMERO DE DIAS X NUMERO DE USUARIOS

TOTAL DE BYTES = 12 X 10 X 512 = 61144 = 60 KBYTES

II.4.- ESPECIFICACIONES FUNCIONALES

Básicamente se buscó que este sistema sea de fácil manejo, capaz de ser operado por personal no especializado y con ello cumplir cabalmente con uno de sus principales objetivos. Para ello, deberá realizar sin ningún problema las siguientes funciones:

-Mostrar permanentemente en un sistema de despliegue toda la información que el usuario requiere como es la fecha, la hora, el turno y si se trata de entrada/salida.

-Permitir al usuario seleccionar las condiciones (entrada/salida y el turno), en las que desea checar.

-Al realizar la acción de checar con la tarjeta, por medio del sensor, tiene que capturar y comprobar el número en código marcado en ella.

-Decodificar y desplegar unos segundos el número equivalente en decimal, correspondiente al número de usuario para que este lo compruebe.

-Direccionar las localidades de memoria adecuadas y almacenar en ellas los datos que correspondan en ese momento.

El teclado debe tener las siguientes caracteristicas:

-Estar claramente señalizado.

-Contar con funciones para actualizar hora y fecha así como para "checar" por medio de este en caso de situaciones irregulares.

-Capturar e interpretar correctamente la función solicitada, salvando la información que no se debe perder.

-Llevar acabo la función, colocándose en un estado de espera para recibir los datos que correspondan, y que se deberán dar en el orden adecuado.

-Desplegar la información según se le vaya dando para comprobar que es correcta.

Finalmente, las especificaciones de la envolvente ó chasis consisten en que:

-Sea lo más reducido posible, con el objeto de que requiera poco espacio para instalarse.
-La distribución de los elementos internos sea tal que se tenga fácil acceso a los puntos de diagnóstico eléctrico, y a los diferentes componentes del equipo.

-Buena presentación, pues de ello depende la actitud que tenga el personal hacia su uso y cuidado.

-Resistente, es decir, elaborarse a partir de un material adecuado que le permita durar mucho tiempo aún cuando esté en una parte de mucha actividad de personal.

-Diseño sencillo, que facilite la producción en serie.

CAPITULO III

DISEÑO ELECTRONICO

Introducción.

Con base a los planteamientos del proyecto de buscar un sistema económico, eficiente y con el soporte necesario para el desarrollo de la programación nos llevó a elegir el microprocesador 8088, el cual nos dá la ventaja de poder manejar datos de 16 bits y a la vez utilizar lógica de 8 bits en su electrónica; considerando además su variada y extensa familia de soporte diseñada para optimizar las tareas del microprocesador y de la cual también se seleccionaron sólo los elementos necesarios para la implementación de este sistema de propósito general.

Bajo los mismos lineamientos se realizó su lógica de decodificación de memorias y de dispositivos periféricos que nos permite, en determinado momento, el crecimiento del diseño según las necesidades de aplicación a través de ranuras de expansión tipo PC.

En cuanto a la aplicación especifica que aquí se le dá, se describe el desarrollo de un sensor óptico necesario para registrar el número del usuario a través de marcas en código binario natural que se ponen en la tarjeta de cada uno, y con lo cual el sistema es capaz de distinguirlos para direccionar y almacenar en las localidades de memoria corespondientes.

Por último, se describe el funcionamiento electrónico del

sistema a partir de sus señales de control, de direcciones y de los datos generados por el microprocesador.

III.1.- Selección del microprocesador.

Los criterios básicos que empleamos para seleccionar el microprocesador son:

- Contexto de operación.
- Frecuencia de trabajo.
- Resolución.
- Periféricos.
- Capacidad de direccionamiento de memoria.
- Definir el grado de paralelismo (etapas de pipeline) a desarrollar en el sistema.
- Definir la arquitectura a emplear.
- Ubicar las características anteriores en alguna familia de microprocesadores comerciales.
- Ubicar el microprocesador específico a emplear.

III.1.a.- Contexto de operación.

Nuestro contexto de operación lo constituye el capturar datos de las tarjetas de checar mediante la interface óptica y la comunicación con la PC para su proceso. El campo donde se instale el equipo puede ser cualquier centro de trabajo e independientemente de la actividad que ahí se desarrolle, no debe estar sujeto a condiciones extremas de: temperatura, suciedad, humedad, ruido eléctrico, etc. Ya que el diseñar para operar en condiciones extremas eleva el costo de los dispositivos que se empleen y consideramos que en la mayoría de las aplicaciones, puede instalarse el equipo en puntos con condiciones menos severas. Aunque buscaremos darle una buena protección en condiciones normales de operación.

III.1.b.- Frecuencia de trabajo.

La frecuencia de trabajo del micropreocesador no es un factor critico en este sistema, debido a que no se le va a enfocar hacia el procesamiento masivo de información ni al muestreo de eventos muy rápidos. Sin embargo no debe ser tan baja que limite su capacidad de adquisición de datos en los procesos que se mencionaron en el capítulo I.

III.1.c.- Resolución (número de bits que se manejan en el proceso)

En el manejo de datos determinamos 9 bits mínimo para distiguir 512 números de identificación diferentes, entonces el utilizar un microprocesador que opere datos de 16 bits nos facilita el manejo de información.

III.1.d.- Periféricos

El sistema debe incluir los dispositivos periféricos adecuados para interactuar con el campo de aplicación de manera eficiente ajustándose a alguno de los estándares de comunicación vigentes para microcomputadoras tipo PC, vía puerto serie en formato RS-232 o mediante un puerto paralelo; así como a las características eléctricas de acoplamiento entre ellos.

Debe tener un dispositivo dedicado a recibir datos desde un teclado y con capacidad para controlar un despliegue.

Tendrá que contar con medios para recibir peticiones de atención por parte de dispositivos externos al sistema. Resumiendo, las tareas que van a realizar los dispositivos periféricos son las siguientes:

- establecer la comunicación en modo serie en formato RS-232 ;
- controlar un puerto paralelo para transmitir y recibir los datos necesarios en esta aplicación ;
- recibir y jerarquizar para que se atiendan por orden de prioridad las peticiones de atención por parte de dispositivos externos al sistema;
- decodificar y explorar un teclado numérico y de funciones especiales;
- habilitar permanentemente un despliegue que muestre la información necesaria al usuario e indique la función que se esté realizando;
- generar la señal de referencia para llevar un reloj de tiempo real.

Existen dispositivos especializados programables que realizan éstas tareas con sólo inicializarlos en el modo de operación deseado (Vea el anexo A).

III.1.e.- Capacidad de direccionamiento de memoria.

Se determinó anteriormente una capacidad de almacenamiento de 80 Kbytes en EEPROM, esto más las localidades de RAM y UVEPROM nos fijan un mínimo de 90 Kbytes de capacidad de direccionamiento, se pueden obtener comercialmente capacidades de 64 Kbytes y 1 Mbyte como opción siguiente, seleccionar una capacidad de direccionamiento de 1 Mbyte cubre nuestras necesidades y nos da la opción de incrementar la decodificación de memoria mediante tarjetas adicionales conecatdas a las lineas de datos, direcciones y control del sistema mediante "slots" o ranuras de expansión.

III.1.f.- Grado de paralelismo (etapas de pipeline) del sistema.

Normalmente, para ejecutar una instrucción un microprocesador lo hace en cuatro etapas principales: "fetch" de la instrucción (IF), durante la que "lee" de memoria el código de la instrucción a ejecutar, decodificación de la instrucción (ID)

en la que identifica la operación a realizar, fetch de los operandos (OF),si es necesario, ejecución de la instrucción (IE) decodificada. En un microprocesador que no maneje el esquema de "pipeline" las cuatro etapas anteriores deben completarse antes de que pueda ejecutarse la instrucción siguiente, mientras que en un microprocesador con etapas en "pipeline" las instrucciones se ejecutan traslapando las etapas mencionadas como se ilustra en la fig.III.1. Donde se muestran 4 etapas de pipeline IF, ID, OF e IE. Puede observarse en los diagramas la ventaja del pipeline en cuanto al tiempo de ejecución de un grupo de instrucciones.

Hasta ahora se ha descrito el "pipeline" de las 4 etapas en la ejecución de una instrucción y debe mencionarse que en algunos sistemas la etapa de ejecución se divide a su vez en múltiples

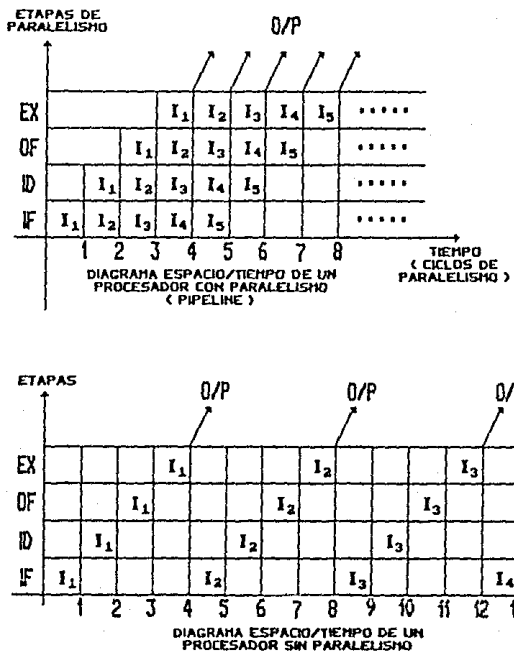


FIG. III.1

etapas que pueden estar traslapadas con las de instrucciones siguientes, logrando así un mayor número de etapas en pipeline y por lo tanto una ejecución mas eficiente.

En nuestro diseño no son necesarias un gran número de etapas

en pipeline, debido a que, como se mencionó anteriormente el tiempo de ejecución de las instrucciones no es un factor crítico.

III.1.g.- Definición de la arquitectura.

En general los sistemas digitales de cómputo pueden clasificarse en cuatro categorías de acuerdo al número de cadenas de instrucciones y de datos. Este esquema de clasificación fue propuesto por Michael J. Flynn. El proceso escencial de computación consiste en la ejecución de una cadena de instrucciones sobre una cadena de datos. Una cadena de instrucciones es una secuencia de éstas en el orden que van siendo ejecutadas por el microprocesador, una cadena de datos es la secuencia de ellos, incluyendo entradas, resultados parciales y resultados que son utilizados por las instrucciones. En ésta clasificación la organización del sistema de cómputo se caracteriza por la redundancia del "hardware" que se proporciona para atender las secuencias de instrucciones y de datos. Así tenemos cuatro posibles arquitecturas:

- Una cadena de instrucciones-una cadena de datos.
  (Single Instruction Stream- single Data stream SISD)

- Una cadena de instrucciones - múltiples cadenas de datos.
  (Single Instruction stream - Multiple Data stream SIMD)

- Múltiples cadenas de instrucciones - una cadena de datos.
  (Multiple Instruction stream - Single Data stream MISD)

- Múltiples cadenas de instrucciones - múltiples cadenas de datos.
  (Multiple Instruction stream - Multiple Data stream MIMD)

La clasificación anterior se ilustra en la figura III.2 y figura III.3. Depende de la redundancia de eventos simultáneos en los componentes del sistema. En los diagramas, instrucciones y datos son alimentados desde los módulos de memoria. Las instrucciones son decodificadas por la unidad de control que envía la instrucción decodificada a la unidad de proceso para su ejecución. Cadenas de datos fluyen entre los procesadores y la memoria bidireccionalmente. Cada cadena de instrucciones decodificadas se genera por una unidad de control independiente.

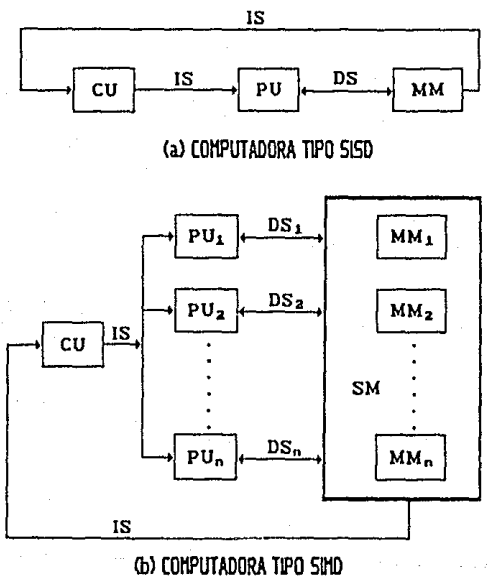
Basaremos nuestro diseño en la arquitectura SISD. Esta organización, mostrada en la fig.III.2 se utiliza en la mayoría de sistemas con un sólo microprocesador. Las instrucciones se ejecutan secuencialmente, aunque pueden ser traslapadas (etapas de pipeline) en su ejecución. La mayoría de sistemas SISD, con un sólo microprocesador manejan etapas en pipeline

III.1.h - Ubicación de la Familia del Microprocesador a Emplear.

Un aspecto importante para este punto está determinado por

la infraestructura con la que se cuenta para el desarrollo del proyecto. Para este caso, el Centro de Instrumentos tiene amplios recursos para el uso de la familia Intel que van desde el software compuesto por programas ensamblador, desensamblador, debugger, editores y procesadores de palabras, etc., hasta el equipo necesario como lo son microcomputadoras, programador de UVEPROM y borrador, analizador de estados lógicos, manuales y toda la información en general necesaria; además de experiencia en el manejo de ésta familia de microprocesadores y sus dispositivos de soporte.

Por su parte, la familia de soporte Intel es de las más extensas y poderosas en cuanto al desahogo de tareas del procesador, además de considerar que casi todos los periféricos existentes para los microprocesadores de 8 bits (de esta familia) son compatibles.



(a) COMPUTADORA TIPO SISD

(b) COMPUTADORA TIPO SIMD

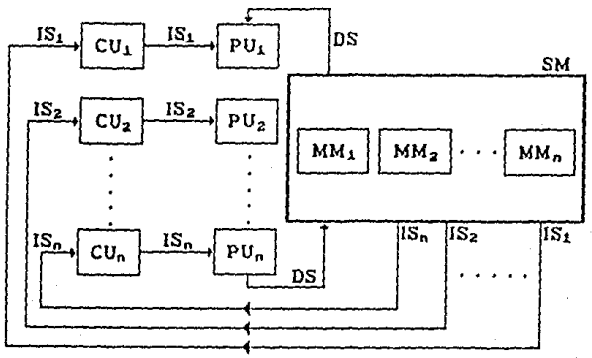
FIG. III.2

III.1.i.- Ubicación del Microprocesador.

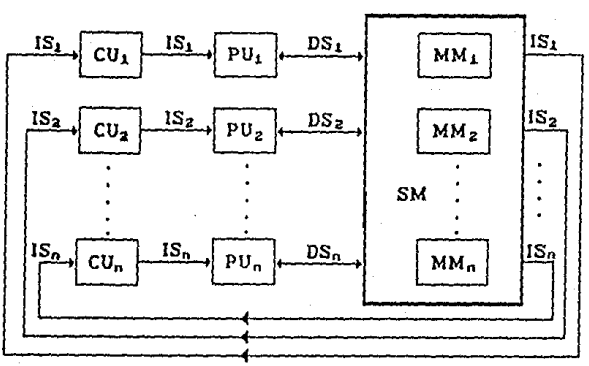
Dentro de la familia Intel se tienen varias opciones; sin embargo, el más adecuado dadas las necesidades planteadas consideramos que es el microprocesador 8088, ya que es en el que

se basa la arquitectura de las microcomputadoras tipo PC cada vez más empleadas en el desempeño de distintas actividades, pero principalmente por permitirnos el direccionamiento de hasta 1 Mbyte de memoria, factor importante tratándose de un sistema de adquisición y almacenamiento de datos.

Otras alternativas eran el microprocesador 8085 (antecesor del 8088), que si bien las instrucciones a nivel ensamblador son casi iguales no nos permite el direccionamiento de la cantidad de



(c) COMPUTADORA TIPO MISD



(d) COMPUTADORA TIPO MIMD

FIG. III.3

memoria necesaria; y por el otro lado estaba el microprocesador 8086 cuya única diferencia con el 8088 es que el segundo maneja un bus de datos externo de sólo 8 bits y en el primero es de 16 bits y otras pequeñas variantes derivadas de la anterior. Por lo pronto, consideramos que para los fines de este sistema un bus de

datos de 8 bits cubre éstos y otros requerimientos aún más exigentes en los que sólo se tendría que adecuar la programación.

En cuanto al grado de paralelismo este no es determinante, sin embargo, al comparar microprocesadores encontramos algunos que para nuestros propósitos son muy sofisticados. Por su parte, el microprocesador 8088 resulta interesante porque es una mezcla de dos filosofías de funcionamiento, una en la que se basa la familia Motorola y otra en la que se basa Zilog, ya que tiene la característica de que su unidad de ejecución está basada en el diseño microprogramado y la unidad de interface al canal funciona a partir de un diseño de lógica aleatoria, lo cual permite un balance entre facilidad de diseño y rapidez, característica de la ejecución de las instrucciones de microprocesadores pipeline de propósito general de la familia Intel. Algunas características del microprocesador 8088 son las siguientes:

-Bus de datos de 8 bits.
-Arquitectura interna de 16 bits.
-Capacidad de direccionamiento directo de memoria de 1 Mbyte.
-Operaciones simétricas con 14 palabras por registros de 16 bits.
-Modos de direccionamiento con 24 operandos.
-Operaciones por byte, palabra y en bloques.
-Aritmética en binario ó decimal con 8 ó 16 bits signada o no signada, incluyendo multiplicación y división.

En base a lo anterior, y considerando para el diseño una arquitectura SISD, sencilla pero enfocada a cubrir diversas aplicaciones, el microprocesador propuesto cumple ampliamente con los requerimientos ya planteados.

La información completa sobre el microprocesador 8088 se puede consultar en el anexo A.

III.2.- Selección de los dispositivos periféricos.

En el inciso III.1.d se estableció que el equipo debe contar con los siguientes dispositivos periféricos:

- Puerto serie.
- Puerto paralelo.
- Controlador de teclado y despliegue.
- Controlador de interrupciones.
- Timer programable.

De la familia de dispositivos periféricos de Intel seleccionamos los siguientes:

Puerto paralelo 8155, con las siguientes características:

- Totalmente compatible con microprocesadores 8088.
- Memoria RAM de 256 palabras x 8bits.
- Operación completamente estática.

- Latch interno de direcciones.
- 2 puerto de E/S programables de 8 bits.
- 1 puerto de E/S programable de 6 bits.
- Contador programable binario de 14 bits (timer programable).
- Bus de datos y direcciones multiplexado.

Puerto serie 8251, con las siguientes características:

-Transmisor/receptor universal síncrono/asíncrono.
-Operación con una amplia variedad de formatos de comunicación serial.
-Fué diseñado para cubrir un amplio rango de requerimientos en los modos de transmisión síncrono, asíncrono e isosíncrono.
-En el modo síncrono opera con 5, 6, 7 u 8 bits por caracter.
-Puede transmitir con los tres formatos en modo duplex, medio, y duplex completo.
-Las señales de interface se pueden dividir en dos grupos: un grupo referido al CPU y otro referido a los dispositivos.

Controlador de interrupciones 8259:

-8 niveles de prioridad en la atención a interrupciones.
-Expandible a 64 niveles.
-Modos de interrupción programables.
-Capacidad de mascarar interrupciones.
-Funcionamiento estático, no requiere entrada de reloj.

Controlador de teclado y despliegue 8279:

-Operación simultánea de teclado y despliegue.
-Modo de entrada controlado.
-Teclado FIFO de 8 caracteres.
-Circuitos de sensado de teclado antirebotes.
-Un display numérico de 16 dígitos, o dos de 8.
-Despliegue con entrada izquierda o derecha.
-Modos de operación programables desde el CPU.
-Muestreo secuencial del teclado.
-Tiempo de muestreo programable.
-Salida de interrupción para indicar un dato de entrada por el teclado.
-Capacidad para manejar despliegues LED, de cristal líquido, etc.

III.3.- Lógica de decodificación de memoria y de dispositivos periféricos.

El tamaño del mapa de memoria está determinado por la capacidad de direccionamiento del microprocesador 8088, que es de un megabyte, desde la dirección 00000H hasta la FFFFFH, este espacio está dividido lógicamente por el microprocesador en segmentos de código, datos, extra y stack, como se explica en el anexo A . Ciertas localidades están reservadas para funciones

especificas, como también se explica en el anexo A, las localidades de la dirección FFFF0H a FFFFFH deben contener un salto hacia el principio de la rutina de inicialización (rutina de reset) del sistema. Las localidades de la 00000H a 003FFH están reservadas para los apuntadores hacia las rutinas de atención a interrupción. De acuerdo a la distribución anterior nuestro mapa de memoria contiene localidades de memoria RAM al

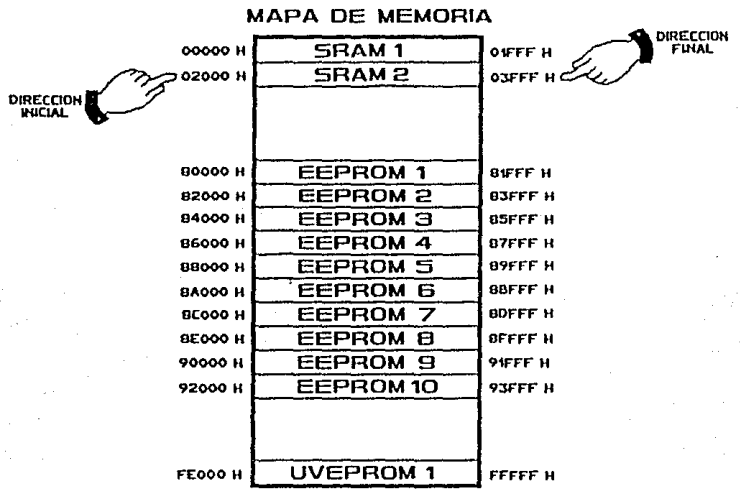


FIG. III.4

principio, donde se cargan el stack y los apuntadores a interrupción. En la última parte del mapa se encuentran localidades de UVEPROM que contienen el programa monitor, las rutinas de servicio e inicialización así como el apuntador hacia la rutina de reset. Quedando en medio del mapa el área de almacenamiento en localidades de EEPROM. Con base en experiencias anteriores en el Centro de Instrumentos, se determinó que 16 Kbytes de RAM y 8 Kbytes de UVEPROM serían suficientes para cubrir las necesidades de la mayoría de aplicaciones. La cantidad de EEPROM está dada por el manejo de datos que hicimos y que como se mencionó es de 60 Kbytes, para contar con recursos adicionales decodificamos 80 Kbytes de EEPROM. El mapa de memoria se implementó con memorias de 8 Kbytes cada una en el arreglo mostrado en la fig.III.4.

En la decodificación de memoria, primero dividimos el mapa en cuatro bloques de 250 Kbytes cada uno, conectando la lineas A_{18} y A_{19} del bus de direcciones a un decodificador de 2:4, en el primer bloque se encuentra la memoria RAM, en el tercero y cuarto se encuentran la EEPROM y la UVEPROM respectivamente.

Para direccionar dentro del bloque de RAM conectamos las lineas de direcciones A_{13} y A_{14} a un decodificador 2:4 del que utilizamos las salidas Y_0 y Y_1 para habilitar a la memoria correspondiente. En el direccionamiento de la EEPROM utilizamos las lineas A_{13}-A_{16} en un decodificador 4:16 del que utilizamos 10 salidas para habilitar cada una de las memorias. El bloque de UVEPROM se decodifica con las lineas A_{13} y A_{14} conectadas a un decodificador 2:4. Conjuntamente con las lineas de dirección se utiliza la linea IO/M para habilitar el mapa de memoria o de periféricos. La decodificación anterior puede observarse en el diagrama esquemático del sistema anexo C, al final del trabajo.

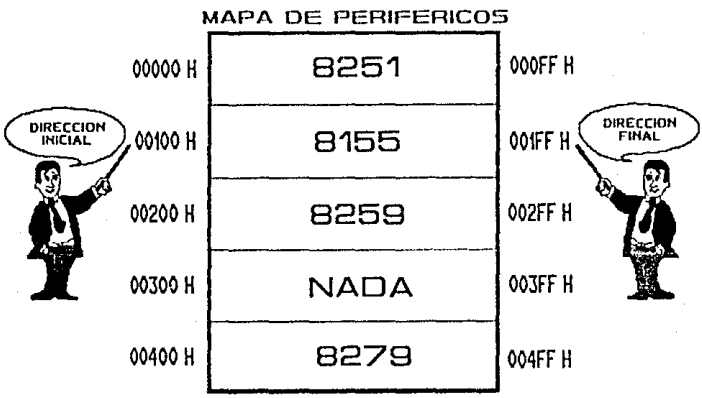


FIG. III.5

El mapa de periféricos en el 8088 puede ser de hasta 64 Kbytes. En nuestra decodificación el puerto serie 8251 ocupa las direcciones 0000H a 00FFH, el puerto paralelo 8155 se direcciona desde 0100H hasta 01FFH, el controlador de interrupciones 8259 se encuentra desde la localidad 0200H hasta la 02FFH y el controlador de teclado 8279 desde la localidad 0400H hasta la localidad 04FFH. La distribución anterior se repite durante todo el mapa de periféricos del 8088 como se muestra en la fig.III.5

Para direccionar el mapa anterior utilizamos las lineas A_8, A_9 y A_{10} conectadas a un decodificador 3:8 del que se utilizan, la salida Y_0 para habilitar al puerto serie, Y_1 para el puerto paralelo, Y_2 para el controlador de interrupciones 8259 y Y_4 para el controlador de teclado y despliegue 8279 como se muestra en el diagrama esquemático (anexo C), nuevamente utilizando la linea IO/M se distingue al mapa de puertos del de memoria.

III.4 - DISEÑO DEL DETECTOR OPTICO.

III.4.a.- Especificaciones de Operación.

Este sistema basa su diseño en la aplicación específica que se le va a dar, sin alterar la forma sencilla con la que el usuario interactúa con un reloj checador, conservando el uso de la tarjeta de checar tradicional para cumplir, además, con un requisito administrativo de la misma Universidad.

Como se mencionó, el sistema reconocerá a cada usuario por medio de un número en código que será incluido en áreas poco utilizadas de la tarjeta, como son las orillas de sus costados más largos; de tal forma que cuando el usuario introduzca su tarjeta al reloj para registrar su hora de entrada y/o salida el código será leido, y se hará una segunda lectura al sacar la tarjeta con el propósito de comparar los números detectados; que de ser iguales, el sistema mostrará el número correspondiente y emitirá una señal audible indicando que se está realizando un registro, procediendo a almacenar la información generada en ese instante. De no cumplirse la igualdad, se deberá intentar introduciendo la tarjeta de nuevo.

En cuanto a la elaboración del código, éste se deberá realizar a partir de marcas relativamente pequeñas (más adelante se especifican las medidas adecuadas) con una tinta ó un material fácil de conseguir en una oficina; cubriendo así otro de los objetivos del proyecto consistente en no perforar las tarjetas para la detección del número.

Por otra parte, la presencia y funcionamiento del sensor no deberá afectar en ningún momento la velocidad con la que el usuario está acostumbrado a meter y sacar la tarjeta del reloj, pues de otro modo el sistema será impráctico. Para que ésto no suceda, el alineamiento entre los sensores y las marcas en la tarjeta de checar no debe ser crítico para su correcta detección, es decir, debe permitirse cierto juego para que se deslize sin atorarse.

Finalmente, la detección deberá continuar realizándose correctamente a pesar del deterioro normal que sufran las tarjetas durante su uso. Mientras se sustituyen las tarjetas maltratadas se puede utilizar la opción de registrar entradas y/o salidas mediante el teclado.

Haciendo una recopilación de las condiciones de operación que debe cubrir el sensado del número en código para no modificar los hábitos de los usuarios al registrar sus entradas y/o salidas, podemos mencionar las siguientes:

-Se mantiene el uso de la tarjeta de "checar" tradicional.
-El código del número no debe interferir con los requisitos que debe cubrir la tarjeta con fines administrativos.
-La tarjeta no deberá perforarse.

-La tinta para marcar el código debe ser fácil de conseguir en una oficina ó centro de trabajo.
-La rapidez con la que los usuarios registran sus entradas y/o salidas no se debe afectar.
-La tarjeta debe seguir funcionando aún cuando se ensucie ó arrugue dentro de un límite razonable.

III.4.b.- Determinación del Tipo de Sensor.

Establecidas las condiciones de operación anteriores, se plantea un medio de sensado óptico para el reconocimiento de las marcas que forman el número en código de cada empleado.

Al investigar las posibilidades que tenemos si utilizamos un medio de detección óptico, nos encontramos que existen básicamente dos métodos de sensado:

Métodos de sensado óptico { - Por endidura ó canal
                           { - Por reflejo

El método de sensado por endidura ó canal consiste en un elemento emisor (LED) y un elemento receptor (fototransistor), enfrente uno del otro en el mismo eje óptico, no necesariamente contenidos en mismo encapsulado, de tal forma que les permite detectar la presencia de un objeto que pase por el claro que hay entre ellos, como se ve en la figura III.6.

No siempre van en un mismo encapsulado, pues se tiene la opción de que se pueden conseguir por separado los diodos emisores y los fototransistores receptores, quedando como único relativo inconveniente, el tener que alinearlos en el mismo eje óptico, dejando a criterio del diseñador el seleccionar la alternativa que más le convenga dependiendo de la aplicación.

El sensado por reflejo consiste en un elemento emisor (LED) y un elemento receptor (fototransistor), dispuestos en un encapsulado, de tal forma que están dirigidos a un mismo punto para sensar la presencia de un objeto, utilizando la luz que éste refleja al pasar enfrente del emisor y que es capturada por el receptor, como se aprecia en la figura III.7.

III.4.c - Selección del Tipo de Sensor.

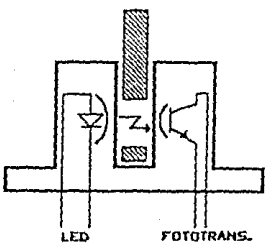
En primera instancia, se consideró la posibilidad de realizar la lectura del número en código por medio del sensado de tipo reflectivo, por ser el método que más se apega a las condiciones de operación, y para lo cual se diseñaron circuitos de acondicionamiento de señal y se realizaron múltiples pruebas.

La experiencia que obtuvimos con este tipo de sensado y específicamente para esta aplicación, nos llevó a intentar con el

método de sensado por endidura ó canal, lo cual no era de nuestro completo agrado pues implicaba el tener que perforar la tarjerta de checar; haciendo que ya no se cumpliera con todas las condiciones, sin embargo, sería más eficaz.

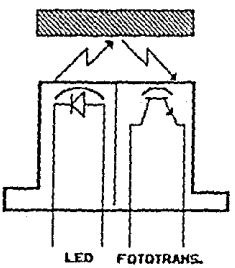
Para iniciar con este método se eligieron los sensores EE-SM3 de la marca OMRON, debido a sus principales características:

-Es el más apropiado para manejar circuitos CMOS.
-Su configuración de salida es de fotodarlington.
-Sus ventanas de emisión y recepción son de las más chicas (0.03" por lado), lo que nos dá una alta resolución y nos permite hacer perforaciones pequeñas.
-Puede observarse la luz infraroja emitida por el LED.



SENSADO POR RANURA

FIG. III.6



SENSADO POR REFLEJO

FIG. III.7

-Existe en dos tamaños de encapsulado, uno con postes altos (EE-SM3) y otro con postes bajos (EE-SM3B), como se muestran en las hojas de características en el anexo D.
-Es económico.

III.4.d - Caracterización de los Sensores.

Como se mencionó en el capitulo anterior, se van a utilizar dos detectores, uno para el registro de interrupciones y otro para el registro del número del usuario, con el fin de que al deslizar la tarjeta y se detecte una interrupción indique al sistema que "lea" el dato que el otro sensor le mande en ese instante. Puesto que el número de usuario está dado por un código de nueve bits, se tiene un igual número de marcas de interrupciones.

Para el empleo de los sensores elegidos se utilizó el mismo circuito que ya se había diseñado para el método de sensado por reflejo, sólo que se volvieron a recalcular los valores de los componentes para que no se sobrepasaran los valores de corriente

y voltaje especificados en las hojas de características, que se pueden consultar en el anexo D.

Dicho circuito nos permite variar la intensidad de luz infrarroja emitida por el LED en base al ajuste de corriente, así como la sensibilidad de recepción del fototransistor.

En los inicios de las pruebas para determinar el tamaño más adecuado de las perforaciones, nos dimos cuenta que no hacía falta perforar las tarjetas, pues el haz de luz infrarojo del LED atraviesa las tarjetas de cartoncillo ó cartulina, siempre y cuando éstas no sean muy gruesas ni de un color obscuro.

Con ésta observación, se pensó en formar el código de las tarjetas a partir de marcas oscuras que interrumpieran el paso de la luz infrarroja, y con ello ya no se tendrían que perforar, por lo que se procedió a caracterizar a un grupo de sensores ópticos, del mismo tipo, de acuerdo a las nuevas condiciones de funcionamiento que se tendrían, con el fin de determinar si podrían ser confiables o no. Dichas condiciones serían ahora:

-Sensor sin tarjeta.
-Sensor con tarjeta.
-Sensor con tarjeta y con marca oscura.

De acuerdo a lo que nos especifican las hojas de características de los sensores, que en el peor de los casos, para una corriente de 3 mA a través del diodo emisor (LED) tendríamos una corriente de salida mínima en el fototransistor de 1,5 mA; se reconfiguró el circuito complementario al receptor con el fin de hacer pasar por él una corriente de 2 mA fijando la del emisor (LED) a 5 mA constantes.

En el receptor se fijó una corriente de colector pequeña, de 2 mA, con el propósito de que se necesite poca corriente de base (es decir, poca luz) para encenderlo, así, aún con la tarjeta, la pequeña cantidad de luz que logre pasar a través de ella sea suficiente para hacerlo conducir; y cuando se interponga una marca oscura se apague, debido a la obstrucción total del paso de la luz infraroja.

El diagrama del circuito de prueba para la caracterización de los sensores es el de la figura III.8.

Donde: @ $V_{satQ1} = 1$ V

$$Rc = \frac{Vcc - V_{satQ1}}{Ic} = \frac{12 - 1}{2 \times 10^{-3}}$$

$Rc = 5.5\ k\Omega$

utilizando un valor comercial: $Rc = 5.6\ k\Omega$

La tarjeta utilizada fué similar a la tradicional que se

emplea para registrar las horas de entrada y/o salida, hecha de cartulina en color amarillo claro y a la cual se le puso una marca oscura con tinta china.

El grupo de sensores caracterizado estaba formado por cinco del modelo EE-SM3 y cinco del modelo EE-SM3B, cuyas características eléctricas son iguales, la única variante está en

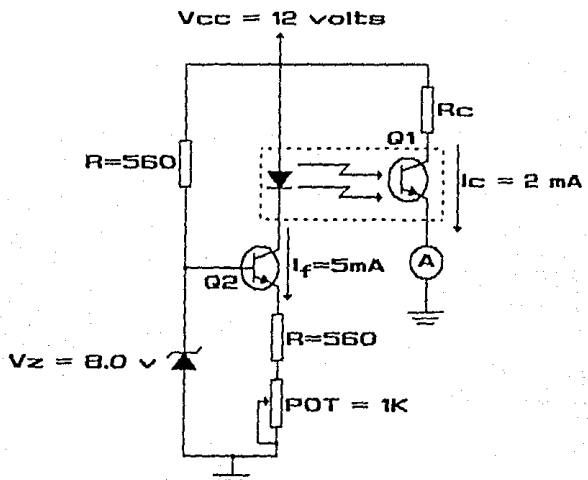


FIG. III.8

su encapsulado; el primero tiene postes más largos que el segundo (0.51" para el SM3 y 0.28" para el SM3B).

Los resultados obtenidos se presentan en la tabla III.1:

CONDICIONES												
Emisor	LED	5	5	5	5	5	5	5	5	5	5	mA
Receptor	Sin tarj.	2	2	2	2	2	2	2	2	2	2	mA
	Con tarj.	20	10	40	60	50	70	60	70	80	60	uA
	Tarj. con marca	1	5	2	3	2	4	3	6	4	5	uA

TABLA III.1

Como se observa en la tabla de resultados anterior, el cambio de la intensidad de corriente entre las condiciones de sin tarjeta y con tarjeta es considerable, no así el que hay entre con tarjeta y con tarjeta con marca. Y es éste el rango en el que hay que definir bien los estados a partir de los niveles de corriente de los sensores, para que no presenten datos falsos, pues puede suceder que el receptor deje de conducir con la sola presencia de la tarjeta o que "no vea" las marcas oscuras.

Otro aspecto importante, considerando que son dos los sensores que trabajan simultáneamente, es la variación que presentan entre ellos para las mismas condiciones, lo que nos obliga a diseñar un circuito complementario al receptor que permita un ajuste de corrientes para compensar sus variantes y asegurar un buen funcionamiento para cualquier sensor de este tipo y modelo.

III.4.e.- Diseño de los Circuitos de Sensado.

Los circuitos complementarios a cada sensor son iguales, requisito para que el sistema sea práctico en su funcionamiento y mantenimiento; la única variante es el modelo de los sensores, pues se utiliza uno de postes largos y uno de postes cortos, pero como ya se mencionó, sus características eléctricas son las mismas.

En la figura III.9 se muestra la configuración del circuito diseñado, complementario a los sensores y conformador de la señal emitida por ellos.

Como se aprecia en la figura III.10, la parte del diodo emisor del sensor está constituida por una fuente de corriente variable con un rango de 4.6 mA a 13 mA, aproximadamente; determinado principalmente para que no se sobrepase la corriente de 15 mA indicada como la máxima en el LED infrarrojo.

Considerando el circuito de la fig. III.10 :

donde: Q2 = 2N2222A
R1 = R3 = 560Ω
R2 = 1 kΩ

Tenemos que los extremos del rango de corriente están determinados por:

- si el cursor del trimpot está en la parte superior:

$$I_{Fmáx} = \frac{V_Z - V_{BE}}{R1} = \frac{8 - 0.7}{560} = 13 \text{ mA}$$

- si el cursor del trimpot está en la parte inferior:

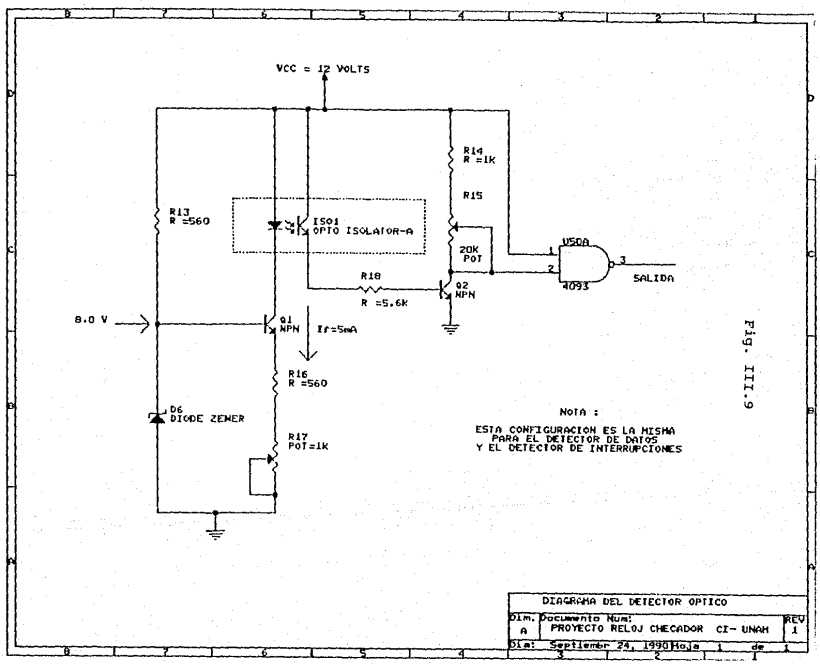

VCC = 12 VOLTS
R14
R =1k
R15
R13
R =560
IS01
OPTO ISOLATOR-A
20K
POT
U50A
4093
SALIDA
R18
R =5.6k
Q2
NPN
8.0 V
Q1
NPN
If=5mA
R16
R =560
D6
DIODE ZENER
R17
POT=1k
Nota :
ESTA CONFIGURACION ES LA MISMA
PARA EL DETECTOR DE DATOS
Y EL DETECTOR DE INTERRUPCIONES
DIAGRAMA DEL DETECTOR OPTICO
Dim. A
Documento Num: PROYECTO RELOJ CHECADOR CI- UNAM
REV 1
Dia: September 24, 1990 Hoja 1 de 1

Fig. III.9

$$I_{Fmin} = \frac{V_Z - V_{BE}}{R1 + R2} = \frac{8 - 0.7}{560 + 1000} = 4.68 \text{ mA}$$

El circuito asociado al fototransistor receptor se muestra en la figura III.11.

Se observa que el funcionamiento del circuito consiste en obtener la respuesta del receptor por medio de su emisor a través de una resistencia que limita la corriente de base del transistor

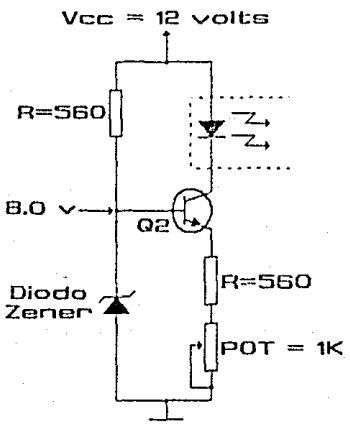


FIG. III.10

de switcheo siguiente, cuya corriente de colector está determinada por una resistencia y un potenciómetro en serie del tipo trimpot de 20 vueltas, y del cual a su vez se obtiene su respuesta en configuración de emisor común, alimentando una de las entradas de una compuerta NAND tipo Schmitt Trigger y teniendo su otra entrada conectada a Vcc, lo que elimina el efecto inversor de la señal de la etapa anterior y evita que se presenten voltajes intermedios a los niveles lógicos (0V y 12V).

En este circuito es importante el ajuste de sensibilidad por medio del trimpot, para compensar las variaciones en las características eléctricas de cada sensor, pues como lo pudimos observar, al realizar la caracterización de los sensores, las especificaciones cambian de un sensor a otro aún siendo del mismo tipo y modelo, y en condiciones ambientales similares.

Para determinar el valor de los componentes del circuito nos basamos en el que ya se había utilizado para la caracterización de los sensores, es decir, de la tabla III.1 consideramos que el valor más crítico de corriente cuando se tiene tarjeta es el menor, o sea, 20 uA y para el cual los transistores deben seguir

saturados sin presentar un cambio en su respuesta. Así, si la resistencia POT está en corto, la resistencia Rc limita el valor de Ic, que no deberá ser menor de 1 mA, pues corresponde, debido a la h_{FE}=50 del transistor a la mínima corriente de base permitible para seguir en saturación, es decir:

$$\frac{Ic}{h_{FE}} = I_{Bsat}$$

$$\frac{1\ mA}{50} = 20\ uA$$

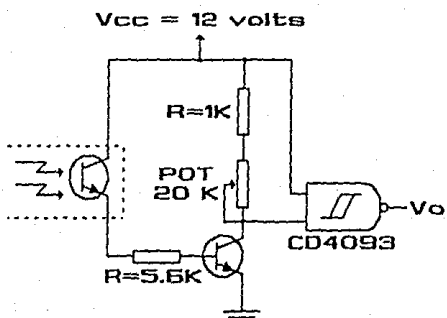


FIG. III.11

ya que de otro modo se saturaría con valores de corriente menores a 20 uA.

Debido a estas condiciones de funcionamiento se calcularon los elementos como sigue:

fijamos: I_B = 2 mA e Ic = 1 mA

de las especificaciones tenemos: $V_{CEsatQ1}$ = 0.9 V

$V_{CEsatQ3}$ = 1 V ===> @ Ic = 1 mA, h_{FE} = 50

Por lo que:

$$R_B = \frac{Vcc - V_{CEsatQ1} - V_{BEQ3}}{I_B} = \frac{12 - 0.9 - 0.7}{2 \times 10^{-3}}$$

R_B = 5.2 kΩ

utilizando un valor comercial: R_B = 5.6 kΩ

y:

$$Rc = \frac{Vcc - V_{CEQ3}}{Ic} \approx \frac{12 - 1}{1 \times 10^{-3}} = 11 \text{ k}\Omega$$

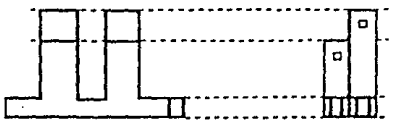
utilizando un valor comercial: Rc = 12 kΩ

El trimpot de 20 k se incluyó para compensar las variaciones que presentan los detectores, es decir, al sumarle a la resistencia Rc (de 1 kohm) una parte del potenciómetro (tipo trimpot de 20 kohms), la corriente Ic tiende a ser proporcionalmente más pequeña en la medida que Rc+POT aumente, con lo cual se necesitará una corriente de base menor para saturar el transistor de switcheo; permitiendo con ello graduar la sensibilidad del receptor. Si la resistencia POT está en corto, la resistencia Rc limita el valor de Ic, que no debe ser menor de 1 mA pues corresponde, debido a la h_{FE} (=50) del transistor, al mínimo permitible para seguir en saturación (1mA / h_{FE} = 2 uA), ya que de otro modo se saturaría para valores de corriente menores de 2 uA.

Por último, llegamos a que para hacer las marcas del dato e interrupciones se necesita de preferencia tinta china ó un plumón negro opaco con el objeto de que no permita el paso de la luz infrarroja, y que las marcas de interrupciones deben estar separadas un mínimo de 7 mm. y tengan 1.5 mm. de ancho para compensar la histéresis en la respuesta de los sensores, mientras las marcas de los datos son mínimo de 3 mm. de ancho por 3 mm. de alto.

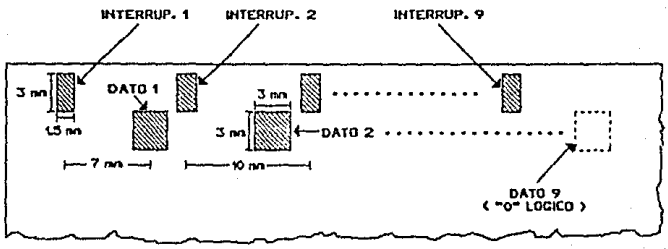
Por la disposición física de los detectores para registrar las interrupciones y los datos, como se observa en la figura III.12, la línea de interrupciones va por arriba de la de los datos, como se muestra en la figura III.13.

Para este caso, observamos que una separación de 10 mm. entre interrupciones es suficiente para que el dato sea leído con seguridad metiendo y sacando la tarjeta a velocidad normal.



POSICION DE LOS DETECTORES

FIG. III.12



POSICION DE LAS LINEAS DE INTERRUPCION Y DATOS

FIG. III.13

III.5.- Teoría de operación del sistema.

Conceptos básicos de operación del microprocesador 8088.

En este diseño se configuró al 8088 en modo mínimo (vea el anexo A), que está optimizado para aplicaciones pequeñas y medianas. El microprocesador mantiene su capacidad de direccionamiento de memoria de 1 Mbyte, así como los 64 Kbytes para direccionar dispositivos periféricos. Proporciona directamente todas las señales de control de bus (DT*/R, DEN*, ALE, M/IO*, RD*, WR*, INTA*), así como un mecanismo sencillo (HOLD, HLDA) para ceder el control del bus a otro procesador (controladores DMA).

El 8088 es un microprocesador con arquitectura interna de 16 bits y un bus de datos externo de 8 bits, 1 Mbyte de capacidad para direccionar memoria y 64 Kbytes de espacio para direccionar dispositivos periféricos. Interactúa con el resto del sistema mediante un bus de 20 bits multiplexado con direcciones, datos y señales de estado. La escritura de datos o la lectura de comandos en memoria las realiza ejecutando ciclos de bus ,el ciclo de bus mínimo consiste en 4 periodos de reloj, llamados estados T, durante el primer estado T (T1), el microprocesador presenta una dirección en el bus multiplexado datos/direc/estado, en el siguiente estado, T2, retira la dirección del bus y pone en tercer estado los 8 bits menos significativos (datos), para una

operación de lectura o bien presenta un dato en estos bits para realizar una escritura. Los comandos RD*, WR* e INTA* se envían siempre durante T2, también se mandan señales de estado del microprocesador a través de los cuatro bits más significativos del bus de datos/direc/estado.

Durante el estado T3, el microprocesador continua proporcionando información sobre su estado y escribiendo o leyendo datos por medio de los ocho bits menos significativos del bus. En caso de que la memoria ó dispositivo de E/S seleccionado no sea capaz de realizar la transferencia a la frecuencia de operación del microprocesador, es necesario que dicho dispositivo indique que no está listo mediante la señal NOT READY, para que inserte ciclos de reloj adicionales (ciclos de espera) a T3. La señal NOT READY debe presentarse al microprocesador al principio de T3. Durante los ciclos de espera se ejecutan las mismas acciones que en T3. El CPU captura los datos en el bus durante el último TW o al final de T3 si no se insertaron ciclos de espera. El ciclo de bus es un evento asincrono para los dispositivos externos al microprocesador en el que el único control que tienen sobre él es la posibilidad de insertar ciclos de espera. El 8088 sólo ejecuta ciclos de bus cuando deben ser transferidos operandos o datos hacia o desde memoria o dispositivos de E/S, cuando no realiza ciclos de bus ejecuta ciclos de reposo en los que mantiene el estado del ciclo de bus anterior, es decir continua presentando datos si el ciclo fué de escritura o mantiene en tercer estado el bus de datos, si el ciclo fué de lectura.

Configuración de la tarjeta principal.

- Demultiplexaje del bus de datos/direcciones/estado.

Como las memorias y dispositivos de E/S que contiene la tarjeta requieren de direcciones estables durante todo el ciclo de bus, es necesario "sujetar" la dirección que presenta el microprocesador durante el estado T1 del ciclo de bus, para lograr esto empleamos tres circuitos 74LS373, cada uno es un "latch" de 8 bits, habilitados por la señal ALE del 8088, cuando ALE esta en alto la dirección presente en las entradas de los 74LS373 se propaga hacia sus salidas, en el flanco de bajada de la señal ALE se sujeta y se mantiene durante todo el ciclo de bus. Con este esquema el bus multiplexado del 8088 se separa en un bus de 8 bits para datos y otro de 20 bits, para direcciones, al que se conectan todos los dispositivos periféricos y de memoria de la tarjeta, como puede observarse en el diagrama esquemático (anexo C), nótese que el bus de datos está todavía multiplexado con direcciones, por esto es necesario que todos los dispositivos periféricos y de memoria cuenten con un "buffer" de tres estados a la salida, habilitado por la señal RD* del microprocesador, para así garantizar que los dispositivos periféricos no interferirán con el bus de direcciones. En el mismo diagrama se observa también que el puerto paralelo 8155 se conectó directamente al bus multiplexado del 8088, esto es debido a que este dispositivo hace él mismo el demultiplexaje del bus,

sincronizado por la señal ALE.

La tarjeta cuenta con tres ranuras de expansión (slots) completamente compatibles con el bus estándard de PC, para lograr esto fue necesario generar las señales MEMW*, MEMR*, I/OW*, I/OR*, se hizo mediante las señles RD*, WR* e IO/M*, conectadas a un decodificador y una compuerta AND como se muestra en el diagrama esquemático, a estas ranuras llegan también las señales de control del 8088.

La totalidad de memoria EEPROM y su lógica de decodificación se instaló en una tarjeta adicional, conectada a través de una de las ranuras de expansión, así se tiene la posibilidad de modificar la capacidad de memoria cambiando dicha tarjeta.

- Señal de reloj del equipo.

El 8088 requiere de una señal de reloj con tiempos de subida y de bajada de 10 nS. máx., con un voltaje bajo de -0.5V a 0.6V y un voltaje alto de 3.9V a VCC+1V. La máxima frecuencia de reloj del 8088 es de 5 Mhz. Como contiene localidades de memoria dinámicas es necesaria una frecuencia mínima de 2 Mhz para mantener el estado del microprocesador. Conforme la frecuencia del sistema se aproxima a la máxima frecuencia de operación , el ciclo de trabajo de la señal debe aproximarse más a un 0.33% para satisfacer los requerimientos mínimos de tiempo en nivel alto y en nivel bajo. Para generar nuestra señal de reloj con los requerimientos anteriores empleamos el circuito integrado 8284 de Intel en la configuración mostrada en el diagrama esquemático. La frecuencia de oscilación del cristal debe ser tres veces la de operación que se desee tener en la salida, en este caso utilizamos un cristal de 12 Mhz., para que el 8088 trabaje con una frecuencia de 4 Mhz. En caso de que en el medio de implementación (circuito impreso, tableta prototipo, etc), se tengan capacitancias parásitas entre las terminales X1 y X2 de más de 10 pf.(sin incluir la capacitancia propia del 8284), es necesario conectar las resistencias R1 y R2 mostradas para asegurar el buen funcionamiento del oscilador interno del dispositivo, aunque es recomendable limitar la capacitancia entre dichas terminales a menos de 10 pf., para evitar desviaciones en la frecuencia de operación.

- Reloj horario del sistema.

Para llevar el reloj en tiempo real del sistema, generamos pulsos cada segundo que se atienden como interrupciones de la mayor jerarquía, la rutina de atención se encarga de incrementar los contadores apropiados, para llevar la cuenta del número de segundos, minutos y horas que han transcurrido desde que se inicializó el sistema. Para generar la frecuencia de 1Hz. dividimos la frecuencia de 1200 Hz. que proporciona el circuito MC 14411, primero entre 5 en un contador binario 7490 y posteriormente entre 240 en el timer interno del puerto paralelo 8155.

-Captura de los datos enviados por el detector óptico.

Cada vez que se inserta una tarjeta de checar en el equipo, el detector óptico envía nueve pulsos de referencia y simultáneamente, asociado a cada uno de ellos, el dato correspondiente es decir un "1" o un "0". Las señales de referencia se atienden como interrupciones al conectarlas a la entrada IR1 del circuito controlador de interrupciones 8259, mientras que la señal de datos se conecta en el bit menos significativo del puerto de entrada B del puerto paralelo 8155. La rutina de atención asociada se encarga de leer el dato presente en el puerto para determinar si se trata de un "1" o un "0", e ir formando el número de identificación del usuario, como se explicó en el capítulo II.

En el resto de los bits del puerto B se encuentran conectados los tres bits menos significativos de un contador de 8 bits, que indica al sistema en que turno se insertó la tarjeta de checar y si se trata de entrada o salida, los pulsos que incrementan la cuenta de este contador se generan cada vez que el usuario oprime un botón con el que selecciona el turno en el que quiere checar y si quiere checar entrada o salida , el microprocesador está programado para desplegar continuamente las condiciones anteriores (turno, entrada o salida), así el usuario sólo tiene que oprimir el botón en repetidas ocasiones hasta tener las condiciones que desee, como se hace en los relojes checadores convencionales.

Cada vez que se captura un número de identificación válido se activa un zumbador durante dos segundos, esto se logra activando el bit menos significativo del puerto de salida A del puerto paralelo 8155. Este bit se conectó a la base de un transistor que funciona como switch saturado para encender el zumbador, transcurridos los dos segundos se desactiva por programación el bit menos significativo del puerto A con lo que se interrumpe la alimentación al zumbador.

-Manejo de interrupciones.

El resto de las terminales de petición de interrupción del circuito 8259 se encuentran conectadas a las ranuras de expansión de la tarjeta principal, de esta manera una tarjeta adicional tiene la posibilidad de interactuar con la tarjeta principal mediante interrupciones.

-Comunicación serie.

El circuito UART 8251 de Intel se encarga de realizar la comunicación serie hacia la computadora PC. A través de un circuito 1489 se hace el acoplamiento de niveles RS-232 a niveles TTL, para recibir, y a través de un circuito 1488 se convierten niveles TTL a RS-232 para transmitir. La frecuencia de transmisión la proporciona un circuito MC 14411 de Motorola diseñado para generar diferentes frecuencias de salida utilizando una red de división de frecuencia, alimentada por un oscilador controlado por cristal. El circuito proporciona una dirección de dos bits para seleccionar uno de cuatro múltiplos de frecuencia

de salida, las principales características de este circuito son:

- fuente de alimentación única de 5 V DC;
- puede entregar 16 diferentes frecuencias de salida;
- el ciclo de trabajo de la señal de salida es del 50%;
- salidas compatibles con niveles TTL,y
- oscilador interno controlado por cristal.

-Funcionamiento del teclado y despliegue del sistema.

El dispositivo encargado de decodificar el teclado y de controlar el despliegue del equipo es el controlador programable de teclado y despliegue 8279 de Intel, instalado en la tarjeta principal (U6). El teclado consta de 16 teclas del tipo "pushbutton", en un arreglo matricial implementadas en una tarjeta separada junto con los 18 caracteres de siete segmentos que forman el despliegue del sistema. Esta tarjeta también contiene la lógica de decodificación necesaria para que el 8279 realize sus funciones .

-Funcionamiento del teclado.

Programamos al 8279 en modo de exploración con teclado codificado (vea el anexo A). A través de sus cuatro lineas de "scan" el 8279 presenta una cuenta progresiva del 0 al 15 en binario. Conectamos los tres bits menos significativos de las líneas de "scan" a un decodificador 3:8, así al ir contando el 8279 se habilita la salida correspondiente en el decodificador . Como el teclado sólo es de 16 teclas utilizamos las cuatro salidas menos significativas del decodificador conectadas a los renglones de el teclado matricial, mientras que a las columnas conectamos las cuatro lineas de retorno menos significativas del 8279 . Así el dispositivo sabe cual fué la tecla que se oprimió dependiendo de en que renglón y en que columna se encuentra.

-Funcionamiento del despliegue.

Conectamos las cuatro lineas de "scan" a un decodificador 4:16, cada salida del decodificador enciende dos caracteres de siete segmentos a través de un transistor que proporciona la amplificación de corriente necesaria para que el caracter opere. El puerto A de salida de datos del 8279 (vea el anexo A) se conectó a un decodificador BCD a siete segmentos, con las salidas conectadas a nueve despliegues de siete segmentos de la tarjeta. El puerto B se conectó a otro decodificador BCD a siete segmentos con las salidas conectadas a otros nueve caracteres de la tarjeta (como se muestra en el diagrama esquemático, anexo C). También conectamos tres caracteres de siete segmentos a los puertos A y B a través de buffers de corriente con el fin de desplegar símbolos no hexadecimales.

El 8279 actualiza el despliegue al incrementar la cuenta en sus lineas de "scan", con lo que se va actualizando cada vez un par de caracteres de siete segmentos, uno con el dato presente en

el puerto A y otro con el dato del puerto B.

-Fuente de alimentación.

Las especificaciones de la fuente de alimentación del sistema son:

-una salida de +5.0 Vdc @ 2.0A ;
-una salida de +12Vdc @ 0.5A, y
-una salida de -12Vdc @0.5A.

Utilizamos un transformador de alimentación con entrada de 127V AC y los siguientes voltajes de salida:

- 30 Vca. @ 1A, con tap central y
- 7.5Vca. @ 2A.

El voltaje una vez rectificado y filtrado se alimenta a reguladores integrados de las series 78XX y 79XX, como puede observarse en el diagrama esquemático. Todo se implementó en una tarjeta aparte de la principal, con el fin de poder emplear otra fuente de alimentación siempre que cumpla con los requerimientos del equipo.

-Diseño del circuito impreso principal.

Al principio del desarrollo de este proyecto implementamos un pequeño sistema, consistente en un 8088 dispositivos de memoria y un puerto serie 8251, en tabletas de experimentación. En la evaluación de este prototipo tuvimos muchos problemas con falsos contactos en las tabletas por lo que decidimos continuar el proyecto en el circuito impreso principal.

Se diseñó una tarjeta de doble cara por la alta densidad de componentes en ella. Para minimizar las capacitancias parásitas, la mayoría de las pistas en una cara son horizontales mientras que en la otra son verticales en su mayor parte. Las líneas de tierra y polarización son lo más anchas posible para reducir los efectos resistivos. Las trayectorias que unen los diferentes puntos de conexión son lo más cortas posible para disminuir efectos inductivos.

En los dispositivos periféricos empleamos capacitores de desacoplo en sus terminales de polarización, para que no se propague al resto de la tarjeta, a través de las líneas de polarización, el ruido generado por los flancos de transición en sus señales de salida.

CAPITULO IV

DESARROLLO DE LA PROGRAMACION

Introducción.

En este capítulo describiremos la metodología que seguimos para realizar la programación del sistema, así como las funciones que realiza cada rutina desarrollada y los parámetros necesarios para su correcto funcionamiento.

Primero realizamos las rutinas de inicialización de los dispositivos periféricos, poniendo especial atención en el puerto serie 8251. Mediante el analizador de estados lógicos comprobamos el correcto funcionamiento de la lógica de decodificación de memoria y puertos de E/S de la tarjeta principal. Una vez que el puerto serie estuvo operando elaboramos un programa monitor para interactuar a través de él con una terminal; este programa cuenta con funciones como: leer y modificar localidades de memoria, leer y escribir datos y comandos en el puerto paralelo 8155, correr programas almacenados en memoria RAM; así tenemos en la terminal un medio para verificar la correcta ejecución de la rutina en que se esté trabajando, apoyados además en el uso de un analizador de estados lógicos, en el que se puede observar el estado de las lineas de datos y direcciones en condiciones predeterminadas.

Cuando todos los dispositivos periféricos estuvieron operando empezamos el desarrollo del programa principal y sus rutinas de apoyo. La programación está hecha de tal modo que el programa principal realiza sus funciones llamando a subrutinas

encargadas de tareas específicas, esto con el fin de tener un programa más estructurado para su mayor comprensión y fácil mantenimiento.

Para ilustrar el funcionamiento de las rutinas y del programa principal mostramos sus diagramas de flujo junto con una descripción de las funciones que realizan, así como con su código en lenguaje ensamblador de 8088 con algunos comentarios para facilitar su comprensión. Las herramientas que empleamos en el desarrollo de la programación son el programa MASM 8088 versión 2.0, la tarjeta EPROM WRITER V-5.3 para grabar UVEPROMS, una computadora PC, una lámpara de luz ultravioleta e instrumentos de medición como : analizador de estados lógicos, multímetro y osciliscopio.

IV.1.- PROGRAMACION DEL SISTEMA

IV.1.A.- Programa principal.

Este programa es el encargado de poner en marcha el sistema después de un RESET, inicializando los periféricos, cargando en las localidades de memoria asignadas los valores iniciales de las variables que se van a emplear, despliega en la terminal (si se encuentra conectada) un letero de bienvenida y un "promt" (>) que indica la espera de algún comando. Llama al programa monitor que es el que se ocupa de interactuar con la terminal, como se mencionó antes, y de actualizar el despliegue del equipo, como se explicará en el inciso siguiente. A continuación se definen todas las variables que se emplean en el programa principal:

DIRECCION DE INICIO DEL STACK.

```
            sptack = 004FH        ; valor del STACK POINTER
            sstack = 0040H        ; valor del SATCK SEGMENT
Dirección de inicio del STACK     ; 0400H + 004F = 044FH
```

DIRECCION DEL PUERTO SERIE 8251.

```
            UART1 = 01H
            UART0 = 00H
```

DIRECCIONES DE MEMORIA.

Direcciones de inicio de mensajes:

```
            DSGEN = 0FF80H        ;segmento de datos en el
                                  que se encuentran los
                                  letreros que despliega el
                                  sistema en la terminal.
```

Letrero de ERROR:

```
            SIERRO = 0443H  ;dirección de inicio del letrero
            COERRO = 0011H  ;número de caracteres del letrero
```

Menú del sistema:

```
SIMENU = 022EH  ;dirección de inicio del MENU
COMENU = 00BBH  ;número de caracteres del MENU
```

Letrero de bienvenida:

```
SILET = 0200H   ;dirección de inicio
COLET = 002DH   ;número de caracteres
```

DATOS PARA EL MOVIMIENTO DE CADENAS DE VECTORES DE INTERRUPCION.

Programa principal:

```
SIVEC2 = 0000H  ;dirección física de origen del
DSVEC2 = 0FF00H ;vector FF000H

DIVEC2 = 000CH  ;dirección física de destino del
ESVEC2 = 0000H  ;vector 000CH

COVEC2 = 0004H  ;número de bytes por transferir.
```

Programa monitor:

```
SIVEC = 0004H   ;dirección física de origen
DSVEC = 0FF00H  ;del vector FF004h

DIVEC = 0080H   ;dirección física de destino
ESVEC = 0000H   ;del vector 0080H

OVEC = 0008H    ;número de bytes por transferir
```

Direcciones de memoria para almacenamiento de datos:

```
M001 = 0201H    ;en éstas localidades se
M002 = 0202H    ;almacenan datos recibidos
M003 = 0203H    ;por el puerto paralelo B
```

Las siguientes localidades se utilizan como contadores en los que se lleva el número de:

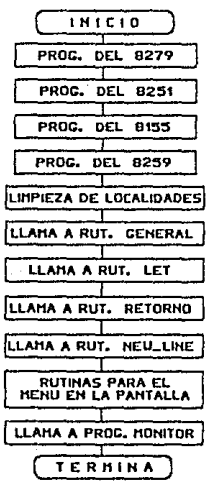
```
SEG = 0450H     ;segundos
MIN = 0451H     ;minutos
HOR = 0452H     ;horas
YEAR = 0453H    ;años
MES = 0454H     ;meses
DIA = 0455H     ;dias
BIS = 0456H     ;años bisiestos
CODIA = 0F001H  ;dias transcurridos a partir
                ;de la ultima transferencia de
                ;datos
```

Las siguientes se utilizan como localidades de control en la ejecución del programa principal:

```
CEROS = 045BH   ;almacena el número de ceros
                ;registrados por el detector
                ;óptico
UNOS  = 045CH   ;almacena el número de unos
                ;registrados por el detector
                ;óptico
DATO  = 045DH   ;almacena el número de
                ;identificación capturado
OTAD  = 0457H   ;almacena el número de
                ;identificación en la verificación
TESD  = 0480H   ;almacena las condiciones
                ;de número de turno, entrada/salida
                ;presentes en el equipo
```

El diagrama de bloques es el siguiente:

DIAGRAMA DE BLOQUES DEL PROGRAMA PRINCIPAL.



Los siguientes son contadores de control que se emplean en diversas rutinas:

```
CONT  = 0459H
CONT2 = 045AH
CONGE = 045FH
```

```
LOC    = 0500H
LOC1   = 0502H
LOC2   = 0503H
CONBO  = 0461H
```

Variables que se refieren al código ASCII:

```
PROMT1 = 003FH ;código ASCII de "?"
SPAA   = 0020H ;código ASCII de un espacio
               ;en blanco
CR     = 000DH ;código ASCII de un retorno de
               ;carro

LF     = 000AH ;código ASCII de un salto de
               ;renglón
```

El programa en ensamblador es el siguiente:

```
;-------------------
; PROGRAMA PRINCIPAL
;-------------------
MAIN:           MOV             BX,SSTACK
                MOV             SS,BX
                MOV             BX,SPTACK
                MOV             SP,BX

                ;PROGRAMACION DE PERIFERICOS

                ;INICIALIZACION DEL 8279
                MOV             DX,401H       ;Modo de teclado y
                MOV             AL,0AH        ;despliegue
                OUT             DX,AL
                NOP
                MOV             DX,401H       ;Ajuste   de   la
                MOV             AL,3FH        ;frecuencia interna
                OUT             DX,AL         ;del reloj
                NOP
                MOV             DX,401H       ;Borra   la   RAM
                MOV             AL,0CDH       ;interna    del
                OUT             DX,AL         ;circuito
                NOP
                MOV             DX,401H       ;Lee la  RAM  como
                MOV             AL,40H        ;FIFO
                OUT             DX,AL
                NOP
                MOV             DX,400H       ;Lee las ocho loca-
                MOV             CX,0AH        ;lidades de la me-
ATI:            IN              AL,DX         ;moria
                LOOP            ATI
                NOP

                ;PROGRAMACION DEL PUERTO SERIE 8251
                MOV             AL,7AH
                OUT             01H,AL
```

```
NOP
MOV             AL,37H
OUT             01H,AL
NOP

;PROGRAMACION DEL PUERTO PARALELO 8155
MOV             DX,0100H
MOV             AL,0C1H
OUT             DX,AL
NOP
MOV             DX,0104H
MOV             AX,4078H
OUT             DX,AX
NOP

;PROGRAMACION DEL CONTROLADOR DE INTERRUPCIONES
;8259
MOV             DX,0200H
MOV             AL,017H
OUT             DX,AL
NOP
NOP
MOV             DX,0201H
MOV             AL,020H
OUT             DX,AL
NOP
NOP
MOV             DX,201H
MOV             AL,03H
OUT             DX,AL
NOP
NOP
MOV             DX,201H
MOV             AL,00H
OUT             DX,AL

;SE LIMPIAN LOCALIDADES PARA LA FECHA Y EL RELOJ
MOV             CX,00H
MOV             DS,CX
MOV             AL,00H
MOV             DS:[SEG],AL
MOV             DS:[MIN],AL
MOV             DS:[HOR],AL
MOV             DS:[CONT],AL
MOV             DS:[CONT2],AL
MOV             DS:[CONBO],AL
MOV             DS:[TESD],AL
MOV             DS:[UNOS],AL
MOV             DS:[CEROS],AL
MOV             AX,00
MOV             DS:[DATO],AX
MOV             DS:[OTAD],AX
MOV             AL,5AH
MOV             DS:[YEAR],AL
MOV             AL,01H
```

```
                MOV             DS:[MES],AL
                MOV             DS:[DIA],AL
                MOV             AL,5CH
                MOV             DS:[BIS],AL
                NOP
                MOV             SI,SILET1       ;Movimiento de vec-
                MOV             CX,COLET1       ;tores de letrero
                CALL            GENERAL         ;de bienvenida
                CALL            LET             ;Letrero de bienve-
COM0:           MOV             SI,SIVEC        ;nida
                MOV             BX,DSVEC
                MOV             DS,BX
                MOV             DI,DIVEC        ;Movimiento de vec-
                MOV             BX,ESVEC        ;tores en la rutina
                MOV             ES,BX           ;principal
                MOV             CX,COVEC
                CLD
FD:             MOVSB                           ;Malla para el ma-
                LOOP            FD              ;nejo de vectores
                STI                             ;de interrupción
                CALL            RETORNO
                CALL            NEW_LINE
                MOV             SI,SIMENU
                MOV             CX,COMENU       ;Envia el menú de
                CALL            GENERAL         ;acciones a la
                CALL            LET             ;pantalla
                CALL            RETORNO
                CALL            MONITOR
                HLT
                NOP
MAIN_PROG       ENDP
PROG_CODE       ENDS
                END             MAIN_PROG
```

IV.1.B.- PROGRAMA MONITOR

Este programa realiza las siguientes funciones:

-Carga los vectores que apuntan a las rutinas de atención a interrupción en el espacio de memoria RAM dedicado a ello en el 8088.

-Lee el registro de estado del controlador de teclado 8279 (vea el anexo A), en caso de que este haya recibido un comando válido lo indicará en este registro y el programa monitor decodificará el comando para ejecutarlo mediante la rutina "Teclado", de lo contrario continua con el siguiente evento.

-Actualiza el despliegue del equipo con los datos que se encuentren en las localidades de almacenamiento de datos del 8279.

-Lee el registro de estado del puerto serie 8251 mediante la rutina actual, en caso de que indique que recibió un dato de la terminal (AP=054), despliega un letrero de bienvenida en la misma, decodifica el dato recibido y ejecuta el comando mediante las subrutinas que se muestran en diagrama de bloques.

-Las acciones anteriores se repiten a partir del segundo inciso, es decir, el programa monitor continuamente polea al puerto serie y al controlador de teclado en espera de algún comando válido.

Comandos del programa monitor:

>PA XX .- Envía un byte XX de dos caracteres en código hexadecimal al puerto de salida A del puerto paralelo 8155 . El sistema regresa al modo monitor una vez concluída esta acción.

>C .- Lee los datos presentes en el puerto de entrada B del puerto paralelo 8155 y los despliega en la terminal conectada al sistema.

>M XXXXX .- Muestra el contenido de la localidad de memoria indicada por el número XXXXX, el cual corresponde a cinco dígitos en código hexadecimal que el sistema organiza de la siguiente manera toma los primeros cuatro dígitos y los coloca en su segmento de datos, posteriormente toma el último dígito y lo coloca en el registro SI, finalmente lee el valor de la localidad de memoria direccionada por la suma de ambos registros y lo presenta en la pantalla como se muestra en la fig.IV.1

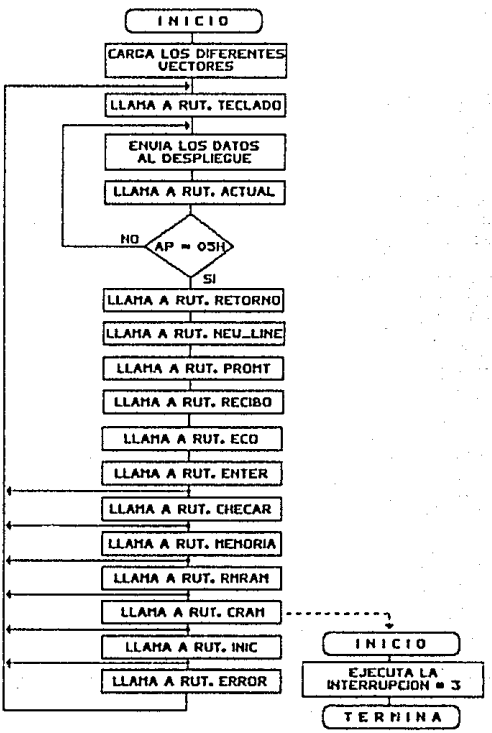
Es posible desplazarse a la dirección de memoria anterior ó a la siguiente, mediante las teclas "." y "LINE FEED" de la terminal.

```
0000:0000 FF
    |     |  |
    |     |  DATO
    |     REGISTRO SI
SEGMENTO
```

Fig. IV.1 Formato para la visualización de memoria.

El diagrama de bloques es el siguiente:

DIAGRAMA DE BLOQUES DEL PROGRAMA MONITOR.



>T XXXXX .- Este comando se utiliza para indicar al sistema que transmita un archivo almacenado a partir de la dirección XXXXX.

>R .- Ejecuta un programa almacenado a partir de la dirección 00470H de la memoria RAM del sistema.

>S .- Inicializa el contador de dias del sistema a 1. Este comando debe enviarse desde la PC cada vez que se vacia la información almacenada en el equipo.

El programa en ensamblador es el siguiente:

```
;-----------------
; PROGRAMA MONITOR
;-----------------
MONITOR         PROC            NEAR
                MOV             SI,SIVEC2     ;Movimiento de vec-
                MOV             BX,DSVEC2     ;tores de interrup-
                MOV             DS,BX         ;ción para el moni-
                MOV             DI,DIVEC2     ;tor.
                MOV             BX,ESVEC2
                MOV             ES,BX
                MOV             CX,COVEC2
HC1:            MOVSB
                LOOP            HC1
                STI
PRINCI:         CALL            TECLADO
                MOV             DX,102H       ;Lee las condiciones
                IN              AL,DX         ;de turno y E/S del
                AND             AL,0FCH       ;puerto paralelo B.
                CMP             AL,4
                JNZ             ES1
                MOV             AL,88H        ;Actualiza el des-
                MOV             DX,401H       ;pliegue con las
                OUT             DX,AL         ;condiciones leídas
                MOV             AL,0F1H       ;del puerto B.
                MOV             DX,400H
                OUT             DX,AL
                MOV             DX,401H
                MOV             AL,9CH
                OUT             DX,AL
                MOV             DX,400H
                MOV             AL,97H
                OUT             DX,AL
                MOV             AL,45H
                OUT             DX,AL
                MOV             AL,87H
                OUT             DX,AL
                JMP             ANT
ES1:            CMP             AL,8
                JNZ             ES2
                MOV             DX,401H
                MOV             AL,88H
                OUT             DX,AL
                MOV             AL,0F1H
                MOV             DX,400H
                OUT             DX,AL
                MOV             DX,401H
                MOV             AL,9CH
                OUT             DX,AL
                MOV             DX,400H
```

```
        MOV     AL,0D6H
        OUT     DX,AL
        MOV     AL,77H
        OUT     DX,AL
        MOV     AL,83H
        OUT     DX,AL
        JMP     ANT
ES2:    CMP     AL,0CH
        JNZ     ES3
        MOV     DX,401H
        MOV     AL,88H
        OUT     DX,AL
        MOV     DX,400H
        MOV     AL,0F2H
        OUT     DX,AL
        MOV     DX,401H
        MOV     AL,9CH
        OUT     DX,AL
        MOV     DX,400H
        MOV     AL,97H
        OUT     DX,AL
        MOV     AL,45H
        OUT     DX,AL
        MOV     AL,87H
        OUT     DX,AL
        JMP     ANT
ES3:    MP      AL,10H
        JNZ     ES4
        MOV     DX,401H
        MOV     AL,88H
        OUT     DX,AL
        MOV     DX,400H
        MOV     AL,0F2H
        OUT     DX,AL
        MOV     DX,401H
        MOV     AL,9CH
        OUT     DX,AL
        MOV     DX,400H
        MOV     AL,0D6H
        OUT     DX,AL
        MOV     AL,77H
        OUT     DX,AL
        MOV     AL,83H
        OUT     DX,AL
        JMP     ANT
ES4:    CMP     AL,14H
        JNZ     ES5
        MOV     DX,401H
        MOV     AL,88H
        OUT     DX,AL
        MOV     DX,400H
        MOV     AL,0F3H
        OUT     DX,AL
        MOV     DX,401H
        MOV     AL,9CH
```

```
                OUT             DX,AL
                MOV             DX,400H
                MOV             AL,97H
                OUT             DX,AL
                MOV             AL,45H
                OUT             DX,AL
                MOV             AL,87H
                OUT             DX,AL
                JMP             ANT
ES5:            MOV             DX,401H
                MOV             AL,88H
                OUT             DX,AL
                MOV             DX,400H
                MOV             AL,0F3H
                OUT             DX,AL
                MOV             DX,401H
                MOV             AL,9CH
                OUT             DX,AL
                MOV             DX,400H
                MOV             AL,0D6H
                OUT             DX,AL
                MOV             AL,77H
                OUT             DX,AL
                MOV             AL,83H
                OUT             DX,AL
ANT:            CALL            ACTUAL
                MOV             CX,0FFFFH       ;Estas instruccio-
J:              NOP                             ;nes introducen un
                NOP                             ;retardo para actua-
                LOOPNZ          J               ;lizar el despliegue.
                IN              AL,UART1        ;Se interroga al
                AND             AL,02           ;puerto serie en
                CMP             AL,02           ;busca de datos.Si
                JZ              MON1            ;no se encuentran va
                JMP             PRINCI          ;a interrogar al te-
MON1:           CALL            RETORNO         ;clado.
                CALL            NEW_LINE
                CALL            PROMT
                CALL            RECIBO          ;Se almacena en AL
                IN              AL,UART0        ;el dato del puerto
                MOV             BL,AL           ;serie y se muestra
                CALL            ECO             ;en la terminal.

                ;SE DETERMINA MEDIANTE COMPARACIONES EL COMANDO
                ;QUE SE RECIBIO, PARA SU EJECUCION.
K:              CMP             BL,50H          ;Compara con P.
                JNZ             A
                CALL            ENTER
                JMP             PRINCI
A:              CMP             BL,43H          ;Compara con C.
                JNZ             B
                CALL            CHECAR
                JMP             PRINCI
B:              CMP             BL,4DH          ;Compara con M.
                JNZ             C
```

```
                CALL        MEMORIA
                JMP         PRINCI
C:              CMP         BL,54H          ;Compara con T.
                JNZ         D
                CALL        RMRAM
                JMP         PRINCI
D:              CMP         BL,52H          ;Compara con R.
                JNZ         G
                CALL        CRAM
                JMP         PRINCI
G:              CMP         BL,53H          ;Compara con S.
                JNZ         H
                CALL        INIC
                JMP         PRINCI
H:              CALL        ERROR           ;Llama al mensaje
                JMP         PRINCI          ;de error en espera
                RET                         ;de otro dato.
MONITOR         ENDP
```

IV.1.C.- SUBRUTINAS

Primeramente se muestran las subrutinas que envian y reciben caracteres desde la terminal. Estas rutinas destruyen ó afectan el contenido del registro AL.

```
;-------------------------------------------------------
; RUTINA QUE BORRA EL CONTENIDO DE LOS REGISTROS DX Y BX
;-------------------------------------------------------
BORRA           PROC            NEAR
                MOV             DH,00H  ;Elima el contenido de
                MOV             DL,00H  ;los registros DX y BX
                MOV             BL,00H  ;haciéndolos ceros.
                MOV             BH,00H
                RET
BORRA           ENDP


;--------------------------------
; RUTINA DE POSICION DE LETREROS
;--------------------------------
GENERAL         PROC            NEAR
                PUSH            BX
                MOV             BX,DSGEN;Coloca el DS en la
                MOV             DS,BX   ;dirección de  los
                POP             BX      ;letreros.
                RET
GENERAL         ENDP


;----------------
; RUTINA DE ENVIO
;----------------
ENVIO           PROC            NEAR    ;Verifica el estado
EN1:            IN              AL,UART1 ;del buffer de trans-
                AND             AL,04H  ;misión del USART y
                CMP             AL,04H  ;en el momento que se
                JNZ             EN1     ;encuentre vacío re-
                RET                     ;gresa.
ENVIO           ENDP


;--------------------
; RUTINA DE RECEPCION
;--------------------
RECIBO           PROC            NEAR    ;Verifica si el  buffer
REC1:           IN              AL,UART1 ;del USART ha recibido
                AND             AL,02H  ;algún dato, de ser así
                CMP             AL,02H  ;regresa.
                JNZ             REC1
                RET
RECIBO          ENDP
```

```
;--------------
; RUTINA DE ECO
;--------------
ECO             PROC            NEAR        ;Envía por el USART
                OUT             UART0,AL    ;un caracter y re -
                CALL            ENVIO       ;gresa en busca de
                RET                         ;otro.
ECO             ENDP


;---------------------------------------
; ENVIO DE UN RETORNO DE CARRO A PANTALLA
;---------------------------------------
RETORNO         PROC            NEAR
                MOV             AL,CR   ;Envia   un   caracter
                CALL            ECO     ;carrier return a la ter-
                RET                     ;minal.
RETORNO         ENDP


;---------------------------
; ENVIA UN ESPACIO A PANTALLA
;---------------------------
SPACE           PROC            NEAR
                MOV             AL,SPAA ;Envía un espacio a
                CALL            ECO     ;pantalla.
                RET
SPACE           ENDP


;--------------------
; ABRE UNA LINEA NUEVA
;--------------------
NEW_LINE        PROC            NEAR
                MOV             AL,LF   ;Envía una línea nueva
                CALL            ECO     ;a pantalla.
                RET
NEW_LINE        ENDP


;-----------------------------------
; ENVIA UN IDENTIFICADOR ">" A PANTALLA
;-----------------------------------
PROMT           PROC            NEAR
                CALL            RETORNO
                CALL            NEW_LINE
                MOV             AL,3EH
                CALL            ECO
                RET
PROMT           ENDP
```

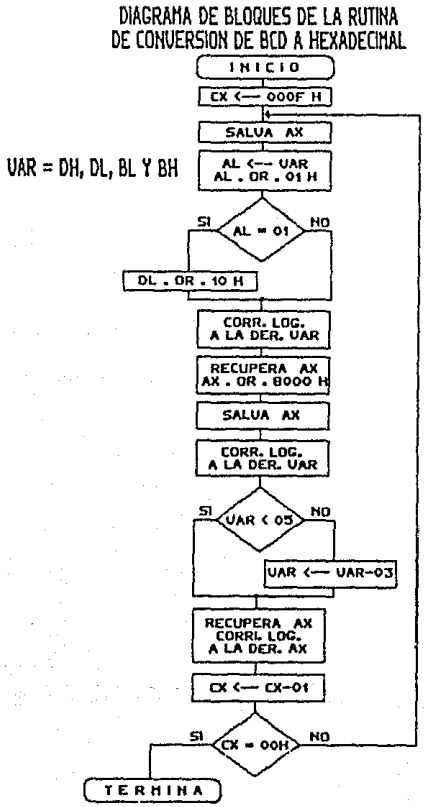
```
;-------------------------------------
; ENVIA UN IDENTIFICADOR "?" A PANTALLA
;-------------------------------------
PROM1           PROC            NEAR
                MOV             AL,PROMT1
                CALL            ECO
                RET
PROM1           ENDP


;-----------------------------------------------------------
; RUTINA QUE SE ENCARGA DEL ENVIO DE UN LETRERO A PANTALLA
;-----------------------------------------------------------
LET             PROC            NEAR
ET1:            MOV             AL,[SI] ;Toma el contenido de la
                CALL            ECO     ;memoria y lo envia a
                INC             SI      ;pantalla las veces que
                LOOP            ET1     ;el tamaño del letrero
                RET                     ;lo indique.
LET             ENDP


;-----------------
; MENSAJE DE ERROR
;-----------------
ERROR           PROC            NEAR
                CALL            RETORNO     ;Apunta a la direc-
                CALL            NEW_LINE    ;ción del letrero de
                MOV             SI,SIERRO   ;error para enviarlo
                MOV             CX,COERRO   ;a pantalla.
                CALL            GENERAL
                CALL            LET
                MOV             CX,0FFFFH
                RET
ERROR           ENDP
```

SUBRUTINA DE CONVERSION DE CODIGO BCD A HEXADECIMAL.

Para realizar la conversión de código BCD a código hexadecimal es necesario tomar en cuenta que se realizará la operación con respecto a cuatro dígitos (utilizando un registro de 8 bits para cada uno), los valores que serán convertidos deberán encontrarse ubicados del más significativo al menos significativo en el siguiente orden de registros DH, DL, BL y BH. El algoritmo se basa en generar un corrimiento a la derecha de los bits de dichos registros y de existir un 1 lógico en el bit



menos significativo del registro BH, antes del corrimiento, este se almacenará en el bit más significativo del acumulador AX, realizándose a continuación una comparación de los registros utilizados con respecto al número cinco, de ser mayores o iguales a este valor, se efectuará una sustracción de tres unidades

(únicamente en aquellos registros que hayan cumplido con esta condición), una vez que este paso se realizó se procede a generar un corrimiento a la derecha del registro AX y se regresa a realizar el algoritmo un total de 16 veces, obteniendo el resultado de la operación en el registro AX.

Debe notarse que el valor en código BCD estará contenido en los cuatro bits menos significativos de cada uno de los registros en que se almacena el valor que se desea convertir y por lo tanto se realiza una pequeña rutina que permite hacer los corrimientos necesarios aún encontrándose los cuatro bits más significativos del siguiente registro de por medio.

El diagrama de bloques se muestra en la hoja anterior.

Programa en ensamblador:

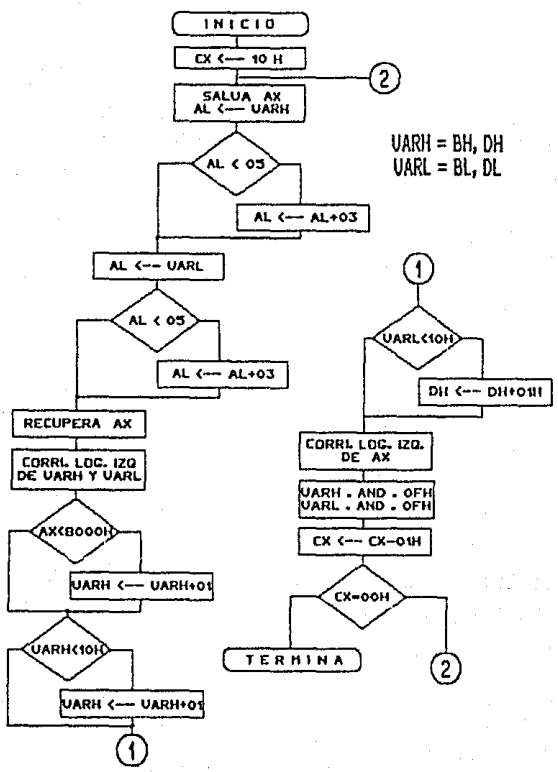
```
BCD_HEX     PROC    NEAR
            MOV     CX,0FH
EMPE:       PUSH    AX
            MOV     AL,DH
            AND     AL,01
            CMP     AL,01
            JNE     EB1
            OR      DL,10H
EB1:        SHR     DH,01
            MOV     AL,DL
            AND     AL,01
            CMP     AL,01
            JNE     EB2
            OR      BL,10H
EB2:        SHR     DL,01
            MOV     AL,BL
            AND     AL,01
            CMP     AL,01
            JNE     EB3
            OR      BH,10H
EB3:        SHR     BL,01
            MOV     AL,BH
            AND     AL,01
            CMP     AL,01
            JNE     EB4
            POP     AX
            OR      AX,8000H
            PUSH    AX
EB4:        SHR     BH,01
            CMP     BH,05
            JB      EB5
            SUB     BH,03
EB5:        CMP     BL,05
            JB      EB6
            SUB     BL,03
EB6:        CMP     DL,05
            JB      EB7
            SUB     DL,03
```

```
EB7:            CMP             DH,05
                JB              EB8
                SUB             DH,03
EB8:            POP             AX
                SHR             AX,01
                LOOP            EMPE
                RET
BCD_HEX         ENDP
```

SUBRUTINA DE CONVERSION DE HEXADECIMAL A BCD.

Esta rutina realiza la conversión de código hexadecimal a código BCD utilizando el algoritmo contrario al que se emplea en la conversión de código BCD a hexadecimal, es decir, el dato que se ha de convertir deberá estar en el registro AX y el resultado se encontrará dentro de los registros DH, DL, BL y BH estando ordenados desde el más significativoa hasta el menos significativo, en el orden en que se indican.

DIAGRAMA DE BLOQUES DE LA RUTINA DE CONVERSION DE HEXADECIMAL A BCD



El procedimiento que se emplea es el siguiente: primero se compara el valor de los registros (que al inicio de la conversión se encuentran en cero) con el número cinco en base hexadecimal, y en el caso de que estos cumplan la condición de ser mayores o iguales se le adiciona un número 3, una vez realizada esta

comparación con los cuatro registros ocupados se efectúa un corrimiento a la izquierda de cada uno de los registros incluyendo el AX y en el caso de que se encuentre un 1 lógico en el bit más significativo del registro AX este pasará a formar parte del registro BH, de la misma manera se prueba el primer bit de la parte más significativa del registro BH (bit 5) y en caso de ser un 1 lógico este se guardará en el siguiente registro (BL), de la misma manera se realiza este paso con los demás registros y se repite la secuencia un total de 16 veces. Destruye el contenido de los registros AX, BX y CX.

El diagrama de bloques se muestra en la hoja anterior.

Programa en ensamblador:

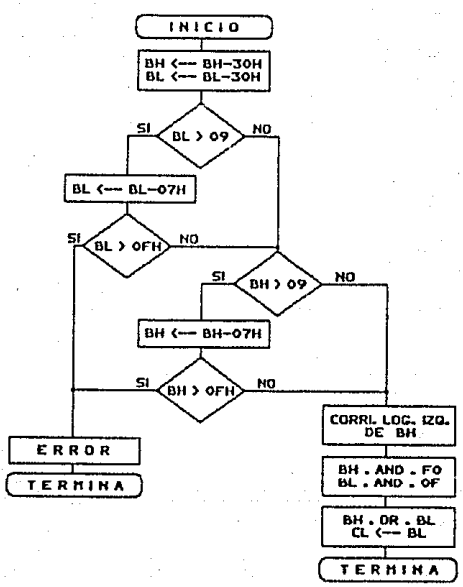
```
HEX_BCD         PROC            NEAR
                CALL            BORRA
                MOV             CX,10H
ETE:            PUSH            AX
                MOV             AL,BH
                CMP             AL,05H
                JB              ET2
                ADD             AL,03H
ET2:            MOV             BH,AL
                MOV             AL,BL
                CMP             AL,05H
                JB              ET3
                ADD             AL,03H
ET3:            MOV             BL,AL
                MOV             AL,DL
                CMP             AL,05H
                JB              ET5
                ADD             AL,03H
ET5:            MOV             DL,AL
                MOV             AL,DH
                CMP             AL,05H
                JB              ET6
                ADD             AL,03H
ET6:            MOV             DH,AL
                POP             AX
                ROL             BH,01
                ROL             BL,01
                ROL             DL,01
                ROL             DH,01
                CMP             AX,8000H
                JB              ETA
                ADD             BH,01H
ETA:            CMP             BH,10H
                JB              ETB
                ADD             BL,01H
ETB:            CMP             BL,10H
                JB              ETC
                ADD             DL,01H
ETC:            CMP             DL,10H
                JB              ETD
```

```
                ADD             DH,01H
ETD:            SHL             AX,01
                AND             BH,0FH
                AND             BL,0FH
                AND             DL,0FH
                AND             DH,0FH
                LOOP            ETE
                RET
HEX_BCD         ENDP
```

SUBRUTINA DE CONVERSION ASCII A HEXADECIMAL.

Obtiene dos dígitos en código hexadecimal a partir de un par de caracteres en ASCII mediante la siguiente secuencia. Primero, para los dos caracteres ASCII se realiza la sustracción con un número 30 en base hexadecimal y verifica si se trata de un número comprendido entre el cero y el nueve, si no es así se les resta un 7 en base hexadecimal. Los resultados de la sustracción se comparan con F base hexadecimal y si alguno es mayor se ejecuta la rutina de error, en caso de que ambos cumplan con las condiciones se encontrarán representados como dos bytes comprendidos cada uno entre 0 y F. En el segundo paso se realizan los corrimientos y las operaciones lógicas para que los dos números obtenidos formen un sólo byte.

DIAGRAMA DE BLOQUES DE LA RUTINA
DE CONVERSION DE ASCII A HEXADECIMAL



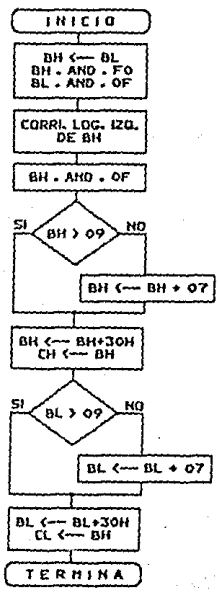
Programa en ensamblador:

```
AII_HEX     PROC    NEAR
            SUB     BH,30H
            SUB     BL,30H
            CMP     BL,0AH
            JB      E1
            SUB     BL,07H
            CMP     BL,10H
            JAE     E2
E1:         CMP     BH,0AH
            JB      E3
            SUB     BH,07H
            CMP     BH,10H
            JAE     E2
E3:         SHL     BH,1
            SHL     BH,1
            SHL     BH,1
            SHL     BH,1
            AND     BH,0F0H
            AND     BL,0FH
            OR      BL,BH
            MOV     CL,BL
            RET
E2:         CALL    ERROR
            RET
AII_HEX     ENDP
```

SUBRUTINA DE CONVERSION DE HEXADECIMAL A ASCII.

Esta rutina obtiene dos caracteres ASCII a partir de dos digitos hexadecimales utilizando el algoritmo inverso de la rutina anterior los resultados se almacenan en el registro CX

DIAGRAMA DE BLOQUES DE LA RUTINA
DE CONVERSION DE HEXADECIMAL A ASCII



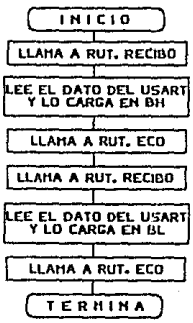
Programa en ensamblador:

```
HEX_AII         PROC            NEAR
                MOV             BH,BL
                AND             BH,0F0H
                AND             BL,0FH
                SHR             BH,1
                SHR             BH,1
                SHR             BH,1
                SHR             BH,1
                AND             BH,0FH
                CMP             BH,0AH
                JB              HE1
                ADD             BH,07H
HE1:            ADD             BH,30H
                MOV             CH,BH
                CMP             BL,0AH
                JB              HE2
                ADD             BL,07H
HE2:            ADD             BL,30H
                MOV             CL,BL
                RET
HEX_AII         ENDP
```

SUBRUTINA QUE CAPTURA DOS CARACTERES ASCII

Esta rutina acepta dos caracteres ASCII desde la terminal a través del puerto serie y los almacena en el registro BX. Se pierde el contenido de BX.

DIAGRAMA DE BLOQUES DE LA RUTINA " INT_REG "



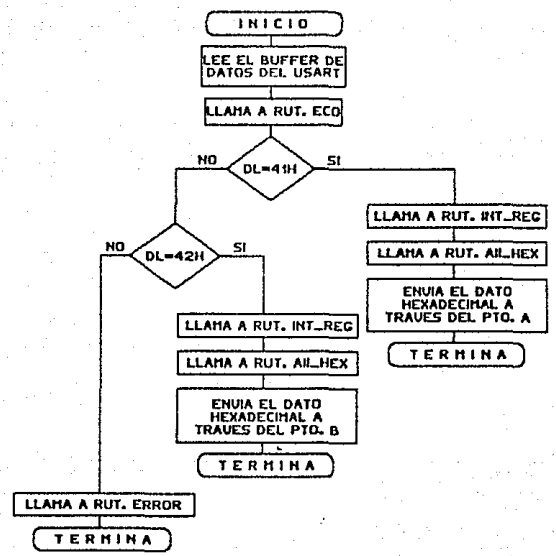
Programa en ensamblador:

```
INT_REG     PROC    NEAR
            CALL    RECIBO
            IN      AL,UART0
            MOV     BH,AL
            CALL    ECO
            CALL    RECIBO
            IN      AL,UART0
            MOV     BL,AL
            CALL    ECO
            RET
INT_REG     ENDP
```

SUBRUTINA PARA EL ENVIO DE DATOS POR EL PUERTO PARALELO

Esta rutina decodifica el puerto paralelo de referencia (A o B) y acepta dos caractres en ASCII convirtiéndolos en dígitos hexadecimales para enviarlos por el puerto seleccionado.

DIAGRAMA DE BLOQUES DE LA RUTINA " ENTER "



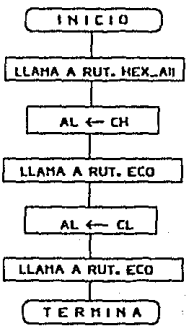
Programa en ensamblador:

```
ENTER       PROC        NEAR
            CALL        RECIBO
            IN          AL,UART0
            MOV         DL,AL
            CALL        ECO
            CMP         DL,41H
            JNZ         E6
            CALL        INT_REG
            CALL        AII_HEX
            MOV         DX,101H
            MOV         AL,CL
            OUT         DX,AL
            RET
E6:         CMP         DL,42H
            JNZ         E7
            CALL        INT_REG
            CALL        AII_HEX
            MOV         DX,102H
            MOV         AL,CL
            OUT         DX,AL
            RET
E7:         CALL        ERROR
            RET
ENTER       ENDP
```

SUBRUTINA DIR_SA

Esta subrutina despliega en pantalla dos caracteres ASCII correspondientes a dos digitos hexadecimales.

DIAGRAMA DE BLOQUES DE LA RUTINA DIR_SA



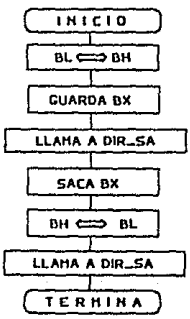
Programa en ensamblador:

```
DIR_SA          PROC    NEAR
                                CALL    HEX_AII
                                MOV     AL,CH
                                CALL    ECO
                                MOV     AL,CL
                                CALL    ECO
                                RET
DIR_SA          ENDP
```

SUBRUTINA DIR_SAC

Despliega en pantalla cuatro caracteres ASCII correspondientes a dígitos hexadecimales.

DIAGRAMA DE BLOQUES DE LA RUTINA DIR_SAC



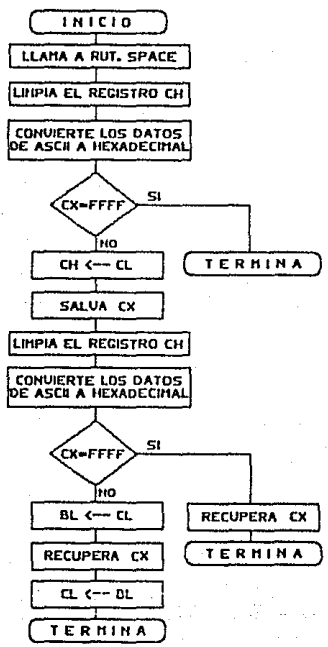
Programa en ensamblador:

```
DIR_SAC         PROC            NEAR
                XCHG            BL,BH
                PUSH            BX
                CALL            DIR_SA
                POP             BX
                XCHG            BH,BL
                CALL            DIR_SA
                RET
DIR_SAC         ENDP
```

SUBRUTINA DIREC

Esta subrutina convierte los cinco caracteres ASCII que se envían de la terminal en un digito hexadecimal que se utiliza para direccionar una localidad de memoria.

DIAGRAMA DE BLOQUES DE LA RUTINA " DIREC "



Programa en ensamblador:

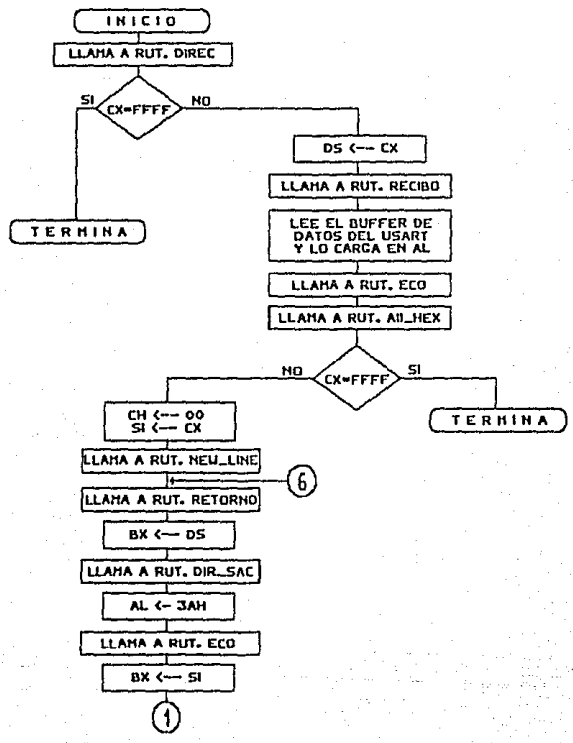
```
DIREC           PROC            NEAR
                CALL            SPACE
                MOV             CH,00
                CALL            INT_REG
                CALL            AII_HEX
                CMP             CX,0FFFFH
                JNZ             CA1
                RET
CA1:            MOV             CH,CL
                PUSH            CX
                MOV             CH,00
                CALL            INT_REG
                CALL            AII_HEX
                CMP             CX,0FFFFH
                JNZ             CA2
                POP             CX
                RET
CA2:            MOV             BL,CL
                POP             CX
                MOV             CL,BL
                RET
DIREC           ENDP
```

SUBRUTINA MEMORIA.

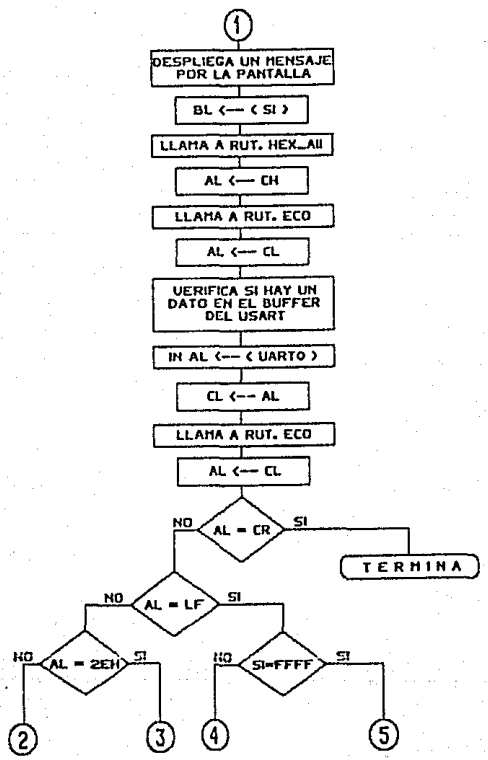
Esta subrutina permite observar el contenido de una localidad de memoria y si se desea, modificar su valor.

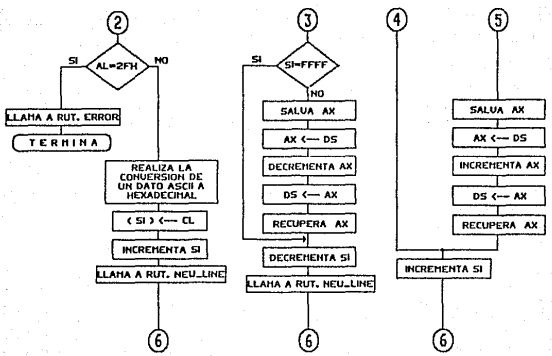
En el diagrama de flujo la condición CX=FFFFH, se da cuando existe un error que termina con la ejecución del programa. El diagrama de bloques se muestra en las páginas siguientes:

RUTINA MEMORIA (PARTE 1)



RUTINA MEMORIA (PARTE 2)





RUTINA MEMORIA (PARTE 3)

Programa en esamblador:

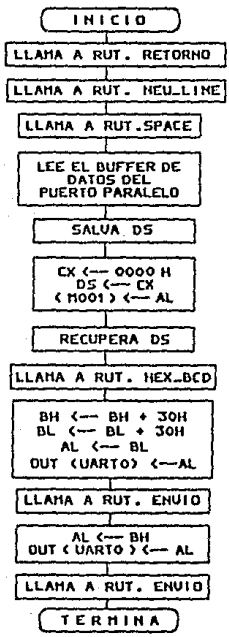
```
MEMORIA         PROC            NEAR
                CALL            DIREC           ;Recibe de la termi-
                CMP             CX,0FFFFH       ;nal la dirección de
                JNZ             CB7             ;la localidad de me-
                RET                             ;moria que se desea
CB7:            MOV             DS,CX           ;observar.
                CALL            RECIBO
                IN              AL,UART0
                MOV             BL,AL
                MOV             BH,30H
                CALL            ECO
                CALL            AII_HEX
                CMP             CX,0FFFFH
                JNZ             CA3
                RET
CA3:            MOV             CH,00H
                MOV             SI,CX
                CALL            NEW_LINE
CA4:            CALL            RETORNO
                MOV             BX,DS
                CALL            DIR_SAC
                MOV             AL,3AH
                CALL            ECO
                MOV             BX,SI
                CALL            DIR_SAC
                CALL            SPACE
                MOV             BL,[SI]
                CALL            HEX_AII
                MOV             AL,CH
                CALL            ECO
                MOV             AL,CL
                CALL            ECO
                CALL            RECIBO          ;Las siguientes ins-
                IN              AL,UART0        ;trucciones decodi-
                MOV             CL,AL           ;fican mediante el
                CALL            ECO             ;comando que se en-
                MOV             AL,CL           ;vió desde la termi-
                CMP             AL,CR           ;nal, para ejecutar-
                JNZ             CA5             ;lo.
                RET
CA5:            CMP             AL,LF
                JNZ             CA6
                CMP             SI,0FFFFH
                JNZ             CC1
                PUSH            AX
                MOV             AX,DS
                INC             AX
                MOV             DS,AX
                POP             AX
```

```
CC1:        INC     SI
            JMP     CA4
CA6:        CMP     AL,2EH
            JNZ     CB1
            CMP     SI,0000H
            JNZ     CB2
            PUSH    AX
            MOV     AX,DS
            DEC     AX
            MOV     DS,AX
            POP     AX
CB2:        DEC     SI
            CALL    NEW_LINE
            JMP     CA4
CB1:        CMP     AL,2FH
            JNZ     CA7
            CALL    SPACE
            CALL    INT_REG
            CALL    AII_HEX
            MOV     [SI],CL
            INC     SI
            CALL    NEW_LINE
            JMP     CA4
CA7:        CALL    ERROR
            RET
MEMORIA     ENDP
```

SUBRUTINA CHECAR

Mediante esta subrutina se reciben datos por el puerto B, del puerto paralelo 8155, los almacena en memoria en la localidad M001 y los despliega en la pantalla de la terminal.

DIAGRAMA DE BLOQUES DE LA RUTINA CHECAR

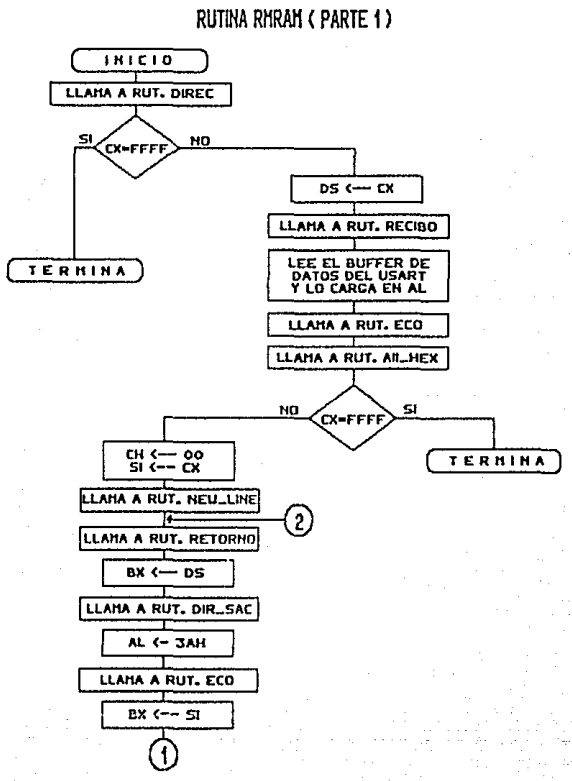


Programa en ensamblador:

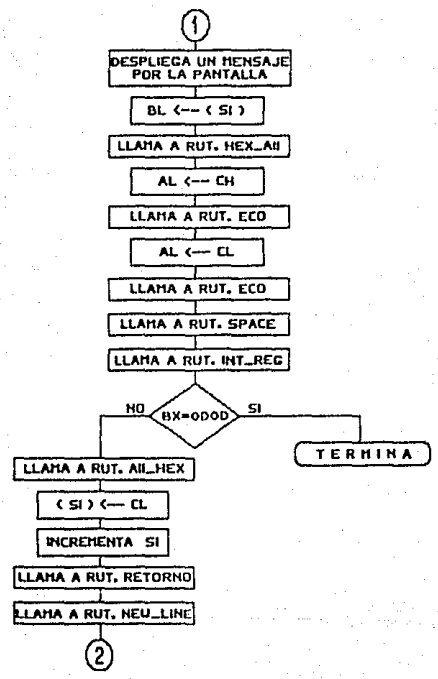
```
CHECAR          PROC            NEAR
                CALL            RETORNO
                CALL            NEW_LINE
                CALL            SPACE
                MOV             DX,102H
                IN              AL,DX
                PUSH            DS
                MOV             CX,0000H
                MOV             DS,CX
                MOV             DS:[M001],AL
                POP             DS
                CALL            HEX_BCD
                ADD             BH,30H          ;Estas instrucciones
                ADD             BL,30H          ;convierten los da-
                MOV             AL,BL           ;tos hexadecimales
                OUT             UART0,AL        ;en ASCII para que
                CALL            ENVIO           ;se muestren en  la
                MOV             AL,BH           ;terminal.
                OUT             UART0,AL
                CALL            ENVIO
                RET
CHECAR          ENDP
```

SUBRUTINA RMRAM.

Recibe archivos por el puerto serie y los carga en la memoria RAM del equipo. Se muestra el diagrama de bloques en las páginas siguientes:



RUTINA RHRAM (PARTE 2)



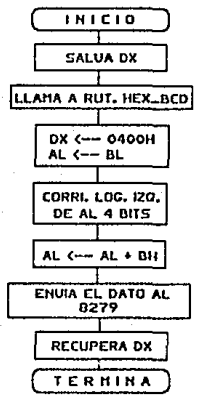
Programa en ensamblador:

```
RMRAM   PROC    NEAR
        CALL    DIREC
        CMP     CX,0FFFFH
        JNZ     FER
        RET
FER:    MOV     DS,CX
        CALL    RECIBO
        IN      AL,UART0
        MOV     BL,AL
        MOV     BH,30H
        CALL    ECO
        CALL    AII_HEX
        CMP     CX,0FFFFH
        JNZ     ENR
        RET
ENR:    MOV     CH,00H
        MOV     SI,CX
        CALL    NEW_LINE
EDG:    CALL    RETORNO
        MOV     BX,DS
        CALL    DIR_SAC
        MOV     AL,3AH
        CALL    ECO
        MOV     BX,SI
        CALL    DIR_SAC
        CALL    SPACE
        MOV     BL,[SI]
        CALL    HEX_AII
        MOV     AL,CH
        CALL    ECO
        MOV     AL,CL
        CALL    ECO
        CALL    SPACE
        CALL    INT_REG
        CMP     BX,0D0DH
        JNZ     ROS
        RET
ROS:    CALL    AII_HEX
        MOV     [SI],CL
        INC     SI
        CALL    RETORNO
        CALL    NEW_LINE
        JMP     EDG
        RET
RMRAM   ENDP
```

SUBRUTINA DISPLAY.

Se encarga de realizar las conversiones y corrimientos necesarios para enviar dígitos hexadecimales al controlador de teclado y despliegue 8279. Convierte un número hexadecimal en dos caracteres BCD y los envía al 8279 mediante el registro AL. A continuación se muestra el listado, en lenguaje ensamblador, de esta rutina:

DIAGRAMA DE BLOQUES DE LA RUTINA DISPLAY



Programa en ensamblador:

```
DISPLAY     PROC    NEAR
            PUSH    DX
            CALL    HEX_BCD
            MOV     DX,400H
            MOV     AL,BL
            ROL     AL,01
            ROL     AL,01
            ROL     AL,01
            ROL     AL,01
            ADD     AL,BH
            OUT     DX,AL
            POP     DX
            RET
DISPLAY     ENDP
```

SUBRUTINA CRAM

Es la encargada de ejecutar los programas que se carguen en la memoria RAM del sistema, lo hace invocando una interrupción por software que obliga al microprocesador a continuar la ejecución a partir de la dirección 0457H de la memoria RAM, en esta localidad debe empezar el programa que se desee ejecutar.

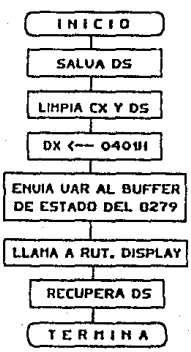
Programa en ensamblador:

```
CRAM            PROC            NEAR
                INT             3
                RET
CRAM            ENDP
```

SUBRUTINA ACTUAL.

Se ocupa de actualizar el despliegue de la hora y fecha del equipo con la información que se encuentre en las localidades de memoria destinadas a este fin. En el diagrama de bloques, 401H es la dirección (en hexadecimal) del registro de control de estado del 8279 y VAR es la palabra de control con la que se programa al dispositivo para escribir en el despliegue del equipo de adquisición de datos. (Vea el apédice A)

DIAGRAMA DE BLOQUES DE LA RUTINA ACTUAL



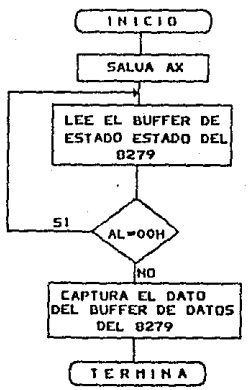
Programa en ensamblador:

```
ACTUAL          PROC            NEAR
                PUSH            DS
                MOV             CX,0000H
                MOV             DS,CX
                NOP
                MOV             DX,401H
                MOV             AL,81H
                OUT             DX,AL
                NOP
                MOV             AL,DS:[DIA]
                CALL            DISPLAY
                NOP
                MOV             AL,82H
                OUT             DX,AL
                MOV             AL,DS:[MES]
                CALL            DISPLAY
                NOP
                MOV             AL,83H
                OUT             DX,AL
                MOV             AL,DS:[YEAR]
                CALL            DISPLAY
                NOP
                MOV             AL,85H
                OUT             DX,AL
                MOV             AL,DS:[HOR]
                CALL            DISPLAY
                NOP
                MOV             AL,86H
                OUT             DX,AL
                MOV             AL,DS:[MIN]
                CALL            DISPLAY
                NOP
                MOV             AL,87H
                OUT             DX,AL
                MOV             AL,DS:[SEG]
                CALL            DISPLAY
                NOP
                POP             DS
                RET
ACTUAL          ENDP
```

SUBRUTINA RETEC

Se encarga de interrogar repetidamente al controlador de teclado 8279 en busca de algún dato presente en su buffer de salida, en caso de existir lee el dato y lo carga en el registro AL. La condición AL = 00H indica que el dispositivo no ha recibido ningún dato desde el teclado.

DIAGRAMA DE BLOQUES DE LA RUTINA RETEC



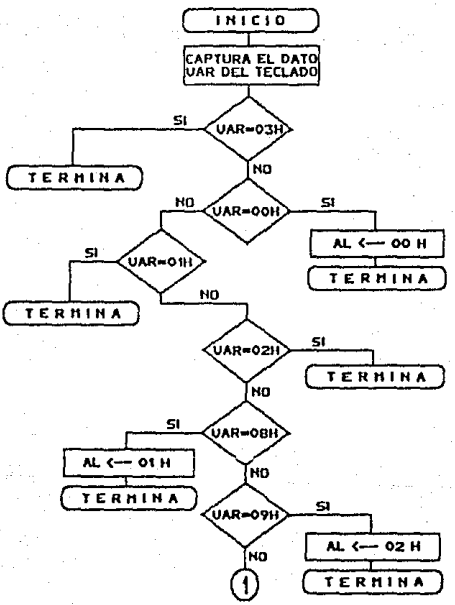
Programa en ensamblador:

```
RETEC       PROC        NEAR
            PUSH        DX
ID0:        MOV         DX,401H
            IN          AL,DX
            AND         AL,07H
            CMP         AL,00
            JZ          ID0
            MOV         DX,400H
            IN          AL,DX
            POP         DX
            RET
RETEC       ENDP
```

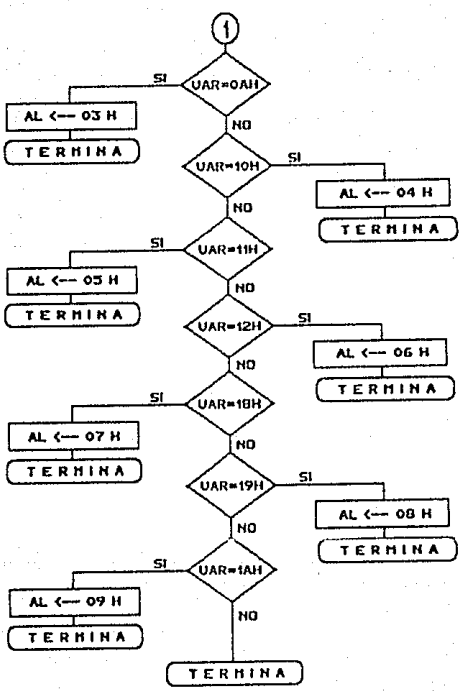
SUBRUTINA IDEN

Determina mediante comparaciones el dato que se recibió desde el teclado y almacena su caracter correspondiente en el registro AL. En las páginas siguientes se muestra su diagrama de bloques, en él, VAR es el valor que el circuito 8279 asigna a cada una de las teclas que forman el teclado, para distinguirlas.

RUTINA IDEN (PARTE 1)



RUTINA IDEN (PARTE 2)



Programa en ensamblador:

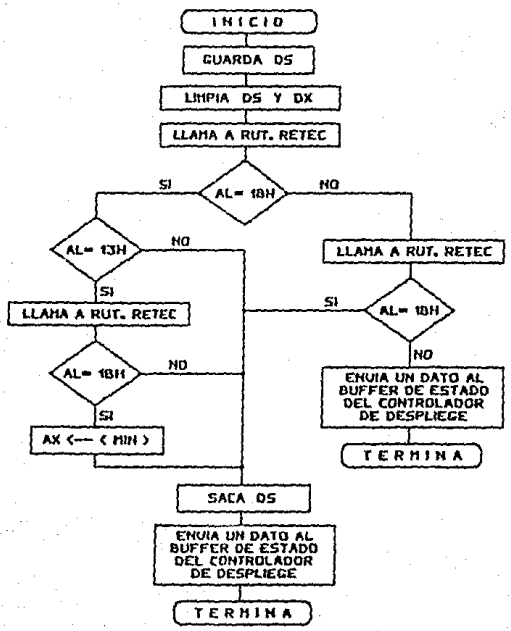
```
IDEN        PROC        NEAR
            CALL        RETEC
            CMP         AL,03
            JZ          ID11
            CMP         AL,00
            JNZ         I1
            MOV         AL,00
            RET
I1:         CMP         AL,01
            JNZ         I2
            RET
I2:         CMP         AL,02
            JNZ         ID1
            RET
ID1:        CMP         AL,08
            JNZ         ID2
            MOV         AL,01
            RET
ID2:        CMP         AL,09
            JNZ         ID3
            MOV         AL,02
            RET
ID3:        CMP         AL,0AH
            JNZ         ID4
            MOV         AL,03
            RET
ID4:        CMP         AL,10H
            JNZ         ID5
            MOV         AL,04
            RET
ID5:        CMP         AL,11H
            JNZ         ID6
            MOV         AL,05
            RET
ID6:        CMP         AL,12H
            JNZ         ID7
            MOV         AL,06
            RET
ID7:        CMP         AL,18H
            JNZ         ID8
            MOV         AL,07
            RET
ID8:        CMP         AL,19H
            JNZ         ID9
            MOV         AL,08
            RET
ID9:        CMP         AL,1AH
            JNZ         ID10
            MOV         AL,09
            RET
```

```
ID10:           CMP             AL,1BH
                JNZ             ID11
                RET
ID11:           RET
IDEN            ENDP
```

SUBRUTINA AUX

Esta subrutina determina mediante comparaciones, contra un valor almacenado en AL, si se desea checar una entrada o una salida a través del teclado, es una rutina auxiliar que se emplea en la subrutina TECLADO.

DIAGRAMA DE BLOQUES DE LA RUTINA AUX



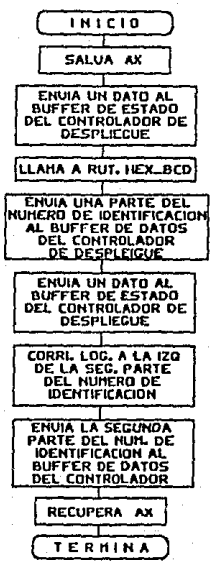
Programa en ensamblador:

```
AUX             PROC            NEAR
                PUSH            DS
                MOV             DX,00
                MOV             DS,DX
                CALL            RETEC
                CMP             AL,1BH
                JNZ             SAL
                CALL            RETEC
                CMP             AL,1BH
                JNZ             QUIT1
                MOV             AX,DS:[MIN]
                POP             DS
                MOV             DS:[BX],AX
                MOV             DX,401H
                MOV             AL,0CDH
                OUT             DX,AL
                RET
SAL:            CMP             AL,13H
                JNZ             QUIT1
                ADD             BX,02
                CALL            RETEC
                CMP             AL,1BH
                JNZ             QUIT1
                MOV             AX,DS:[MIN]
QUIT1:          POP             DS
                MOV             DS:[BX],AX
                MOV             DX,401H
                MOV             AL,0CDH
                OUT             DX,AL
                RET
AUX             ENDP
```

SUBRUTINA DENUM

Se encarga de desplegar el número de identificación del usuario que haya checado su tarjeta . Convirtiendo el número hexadecimal que envía el detector óptico en dígitos BCD para que sean desplegados en caracteres de siete segmentos.

DIAGRAMA DE BLOQUES DE LA RUTINA DENUM



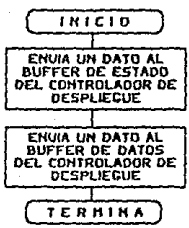
Programa en ensamblador:

```
DENUM           PROC            NEAR
                PUSH            AX
                MOV             DX,401H
                MOV             AL,8AH
                OUT             DX,AL
                NOP
                POP             AX
                PUSH            AX
                CALL            HEX_BCD
                MOV             AL,DL
                MOV             DX,400H
                OUT             DX,AL
                NOP
                MOV             DX,401H
                MOV             AL,8BH
                OUT             DX,AL
                NOP
                MOV             DX,400H
                MOV             AL,BL
                ROL             AL,1
                ROL             AL,1
                ROL             AL,1
                ROL             AL,1
                ADD             AL,BH
                OUT             DX,AL
                NOP
                POP             AX
                RET
DENUM           ENDP
```

SUBRUTINA OVER

Esta subrutina despliega una indicación de memoria llena en los mismos dígitos en que aparece el número del empleado cuando alguien trata de "checar" en éstas condiciones.

DIAGRAMA DE BLOQUES DE LA RUTINA OVER



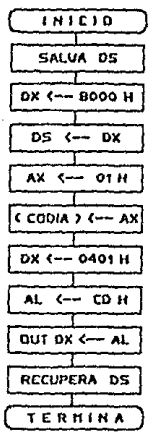
Programa en ensamblador:

```
OVER            PROC            NEAR
                MOV             DX,401H
                MOV             AL,8AH
                OUT             DX,AL
                MOV             DX,400H
                MOV             AL,0BCH
                OUT             DX,AL
                MOV             DX,401H
                MOV             AL,08BH
                OUT             DX,AL
                MOV             DX,400H
                MOV             AL,0BCH
                OUT             DX,AL
                RET
OVER            ENDP
```

SUBRUTINA INIC

Inicializa la localidad donde se lleva el contador de días transcurridos (CODIA).

DIAGRAMA DE BLOQUES DE LA RUTINA INIC



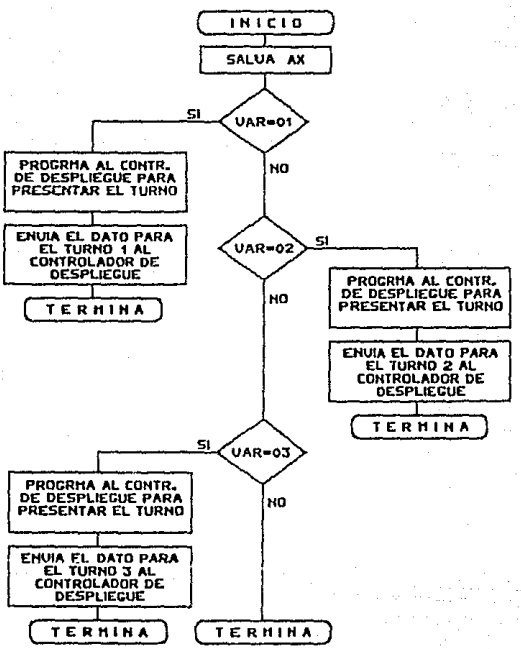
Programa en ensamblador:

```
INIC            PROC            NEAR
                PUSH            DS
                MOV             DX,8000H
                MOV             DS,DX
                MOV             AX,01
                MOV             DS:[CODIA],AX
                MOV             DX,401H
                MOV             AL,0CDH
                OUT             DX,AL
                POP             DS
                RET
INIC            ENDP
```

SUBRUTINA DETUR.

Esta es una subrutina auxiliar que se encarga de efectuar las conversiones y corrimientos necesarios para desplegar el número de turno que el equipo esté checando.

DIAGRAMA DE BLOQUES DE LA RUTINA DETUR

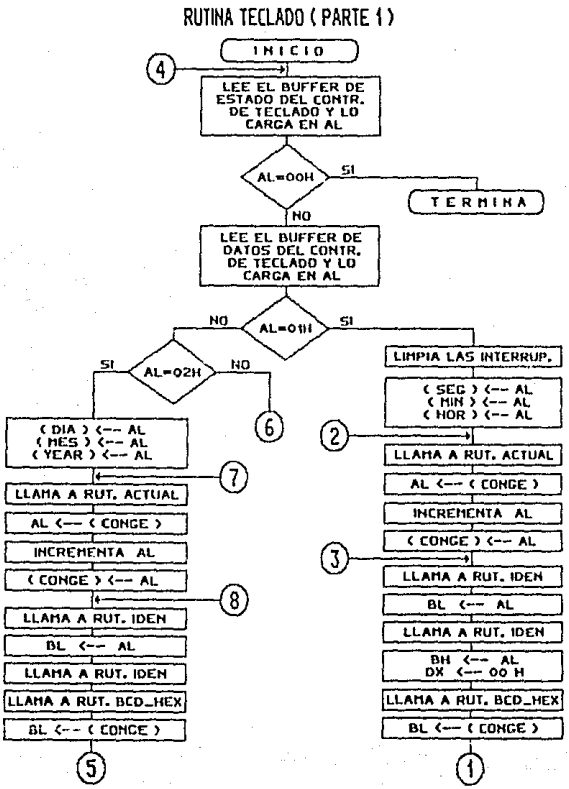


Programa en ensamblador:

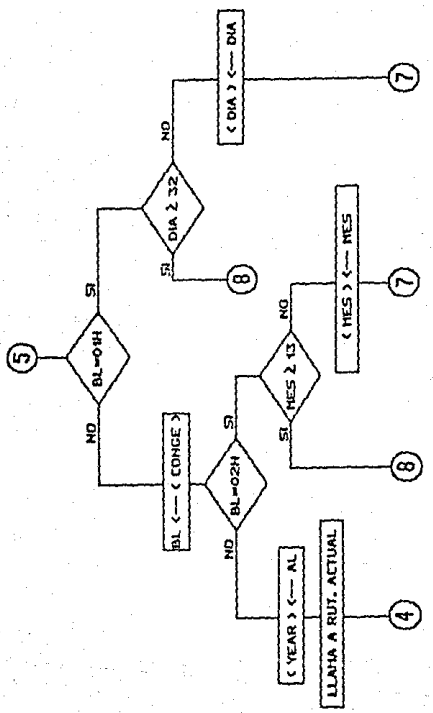
```
DETUR           PROC            NEAR
                PUSH            AX
                CMP             AL,1BH
                JNZ             DET1
                MOV             DX,401H
                MOV             AL,88H
                OUT             DX,AL
                MOV             DX,400H
                MOV             AL,0F1H
                OUT             DX,AL
                POP             AX
                RET
DET1:           CMP             AL,13H
                JNZ             DET2
                MOV             DX,401H
                MOV             AL,88H
                OUT             DX,AL
                MOV             DX,400H
                MOV             AL,0F2H
                OUT             DX,AL
                POP             AX
                RET
DET2:           CMP             AL,0BH
                JNZ             DET3
                MOV             DX,401H
                MOV             AL,88H
                OUT             DX,AL
                MOV             DX,400H
                MOV             AL,0F3H
                OUT             DX,AL
DET3:           POP             AX
                RET
DETUR           ENDP
```

SUBRUTINA TECLADO

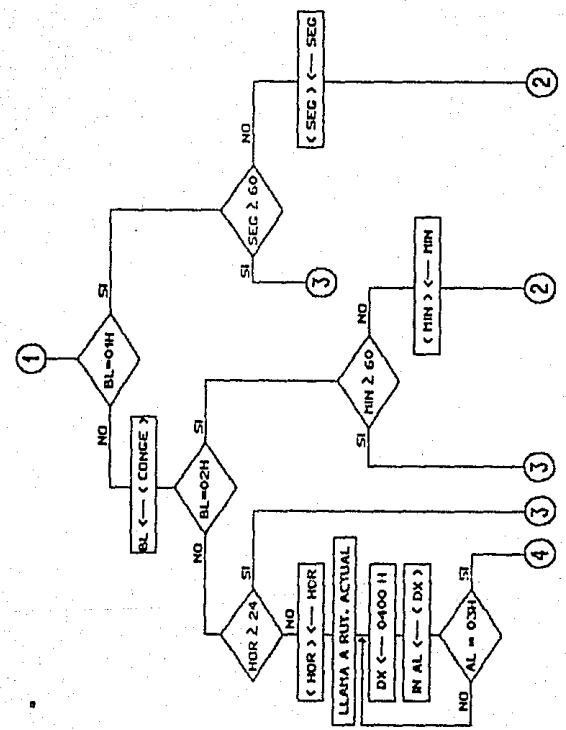
Se encarga de decodificar, mediante comparaciones, y ejecutar, todos los comandos que se reciben desde el teclado, tales como : modificar hora y fecha, checar entrada o salida a través del teclado.



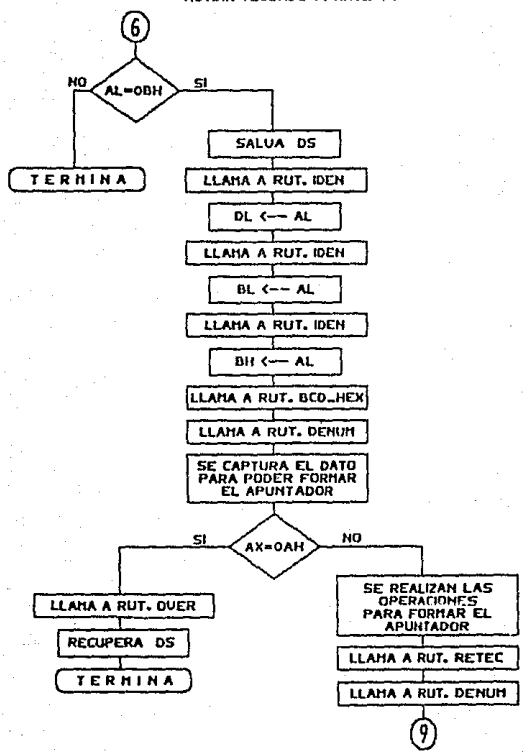
RUTINA TECLADO (PARTE 2)

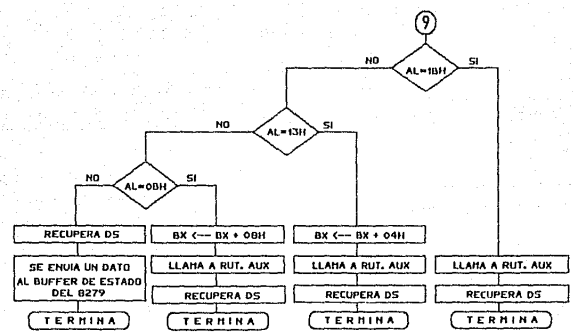


RUTINA TECLADO (PARTE 3)



RUTINA TECLADO (PARTE 4)





RUTINA TECLADO (PARTE 5)

Programa en ensamblador:

```
TECLADO         PROC            NEAR
QUIT:           STI
                MOV             DX,401H
                IN              AL,DX
                AND             AL,07
                CMP             AL,00
                JNZ             TE1
                RET
TE1:            MOV             BX,0000H
                MOV             DS,BX
                MOV             DS:[CONGE],BL
                MOV             DX,400H
                IN              AL,DX
                CMP             AL,01           ;En este bloque se
                JNZ             MOFE            ;modifica la hora
                CLI                             ;(hora, minutos, se-
                MOV             AL,00           ;gundos).
                MOV             DS:[SEG],AL
                MOV             DS:[MIN],AL
                MOV             DS:[HOR],AL
ACT:            CALL            ACTUAL
                MOV             AL,DS:[CONGE]
                INC             AL
                MOV             DS:[CONGE],AL
TONTO:          CALL            IDEN
                MOV             BL,AL
                CALL            IDEN
                MOV             BH,AL
                MOV             DX,00
                CALL            BCD_HEX
                MOV             BL,DS:[CONGE]
                CMP             BL,01
                JNZ             TEMN
                CMP             AL,3CH
                JGE             TONTO
                MOV             DS:[SEG],AL
                JMP             ACT
TEMN:           MOV             BL,DS:[CONGE]
                CMP             BL,02
                JNZ             TEHR
                CMP             AL,3CH
                JGE             TONTO
                MOV             DS:[MIN],AL
                JMP             ACT
TEHR:           CMP             AL,18H
                JGE             TONTO
                MOV             DS:[HOR],AL
                CALL            ACTUAL
TE2:            MOV             DX,400H
                IN              AL,DX
```

```
                CMP             AL,03
                JZ              QUIT
                JMP             TE2
MOFE:           CMP             AL,02           ;En este bloque se
                JNZ             CHEC            ;modifica la fecha
                MOV             AL,00           ;(dia, mes, año).
                MOV             DS:[DIA],AL
                MOV             DS:[MES],AL
                MOV             DS:[YEAR],AL
MOF1:           CALL            ACTUAL
                MOV             AL,DS:[CONGE]
                INC             AL
                MOV             DS:[CONGE],AL
TONTO1:         CALL            IDEN
                MOV             BL,AL
                CALL            IDEN
                MOV             BH,AL
                MOV             DX,00
                CALL            BCD_HEX
                MOV             BL,DS:[CONGE]
                CMP             BL,01
                JNZ             MES1
                CMP             AL,20H
                JGE             TONTO1
                MOV             DS:[DIA],AL
                JMP             MOF1
MES1:           MOV             BL,DS:[CONGE]
                CMP             BL,02
                JNZ             AN
                CMP             AL,0DH
                JGE             TONTO1
                MOV             DS:[MES],AL
                JMP             MOF1
AN:             MOV             DS:[YEAR],AL
                CALL            ACTUAL
                JMP             QUIT
CHEC:           CMP             AL,0BH          ;En este bloque se
                JNZ             SET             ;"checa" por medio
                PUSH            DS              ;del teclado.
                CALL            IDEN
                MOV             DL,AL
                CALL            IDEN
                MOV             BL,AL
                CALL            IDEN
                MOV             BH,AL
                MOV             DH,00
                CALL            BCD_HEX
                CALL            DENUM
                MOV             BX,78H
                MUL             BX
                MOV             BX,AX
                MOV             DX,8000H
                MOV             DS,DX
                MOV             AX,DS:[CODIA]
                CMP             AX,0AH
```

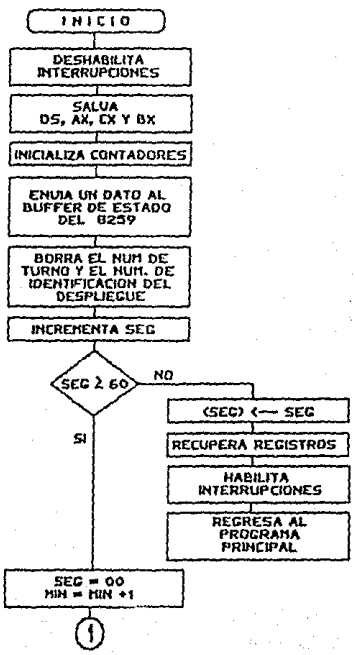
```
                JG              OV
                MOV             AX,0CH
                MOV             DX,DS:[CODIA]
                MUL             DX
                SUB             AX,0CH
                ADD             BX,AX
                CALL            RETEC
                CALL            DETUR
                CMP             AL,1BH
                JNZ             TARDE
                CALL            AUX
                POP             DS
                RET
TARDE:          CMP             AL,13H
                JNZ             EXT
                ADD             BX,04
                CALL            AUX
                POP             DS
                RET
EXT:            CMP             AL,0BH
                JNZ             TIUQ
                ADD             BX,08
                CALL            AUX
                POP             DS
SET:            RET
TIUQ:           POP             DS
                MOV             DX,401H
                MOV             AL,0CDH
                OUT             DX,AL
                RET
OV:             CALL            OVER
                POP             DS
                RET
TECLADO         ENDP
```

IV.1.D.- RUTINAS DE ATENCION A INTERRUPCION

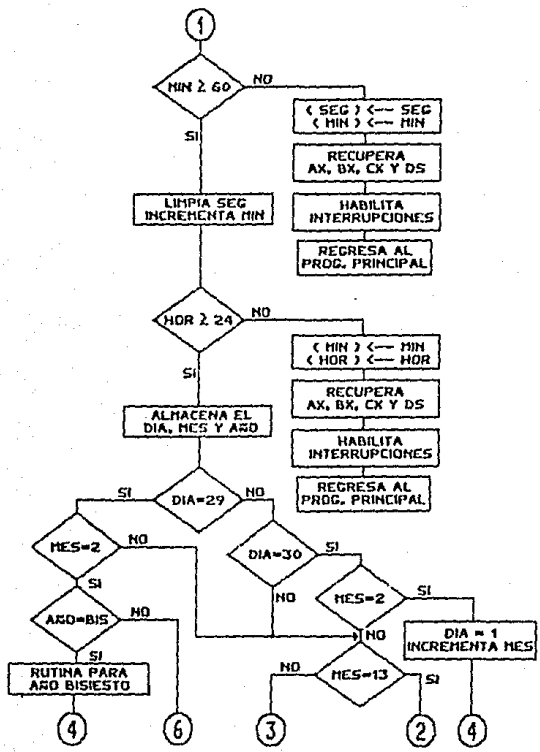
RUTINA DE ATENCION A LA INTERRUPCION DEL TIMER

Es la rutina de mayor jerarquía. Se encarga de actualizar los segundos y a partir de ahí los minutos, horas, días, mes y año considerando los años bisiestos; también borra el número de usuario y deshabilita, después de dos segundos, el zumbador que indica al usuario que su número de identificación es válido.

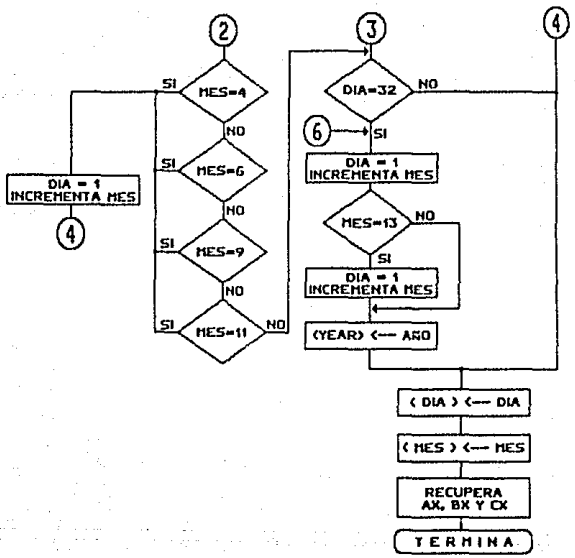
RUTINA DE INTERRUPCION DEL TIMER (PARTE 1)



RUTINA DE INTERRUPCION DEL TIMER (PARTE 2)



RUTINA DE INTERRUPCION DEL TIMER (PARTE 3)



Programa en ensamblador:

```
        CLI
        NOP
        PUSH    DS
        PUSH    AX
        PUSH    CX
        PUSH    BX
        MOV     CX,0000H
        MOV     DS,CX
        MOV     AL,DS:[CONBO]
        INC     AL
        MOV     DS:[CONBO],AL
        CMP     AL,2
        JNZ     TI
        MOV     DX,201H
        MOV     AL,00
        OUT     DX,AL
TI:     CMP     AL,3
        JNZ     TIM1
        MOV     DX,401H
        MOV     AL,88H
        OUT     DX,AL
        MOV     DX,400H
        MOV     AL,0FFH
        OUT     DX,AL
        MOV     DX,401H
        MOV     AL,8AH
        OUT     DX,AL
        MOV     DX,400H
        MOV     AL,0FFH
        OUT     DX,AL
        MOV     DX,401H
        MOV     AL,8BH
        OUT     DX,AL
        MOV     DX,400H
        MOV     AL,0FFH
        OUT     DX,AL
TIM1:   MOV     AL,DS:[SEG]
        NOP
        INC     AL
        CMP     AL,3CH
        JGE     MN
        NOP
        MOV     DS:[SEG],AL
        POP     BX
        POP     CX
        POP     AX
        POP     DS
        STI
        IRET
MN:     MOV     AL,00H
```

```
        MOV     DS:[SEG],AL
        MOV     AL,DS:[MIN]
        INC     AL
        CMP     AL,3CH
        JGE     HR
        MOV     DS:[MIN],AL
        POP     BX
        POP     CX
        POP     AX
        POP     DS
        STI
        IRET
HR:     MOV     AL,00
        MOV     DS:[MIN],AL
        MOV     AL,DS:[HOR]
        INC     AL
        CMP     AL,18H
        JGE     DAY
        MOV     DS:[HOR],AL
        POP     BX
        POP     CX
        POP     AX
        POP     DS
        STI
        IRET
DAY:    MOV     AL,00H
        MOV     DS:[HOR],AL
        MOV     AL,DS:[DIA]
        INC     AL
        PUSH    DS
        MOV     DX,8000H
        MOV     DS,DX
        MOV     DX,DS:[CODIA]
        INC     DX
        MOV     DS:[CODIA],DX
        POP     DS
        MOV     AH,DS:[MES]
        CMP     AL,1DH
        JNZ     RE1
        CMP     AH,02H
        JNZ     RE2
        MOV     BL,DS:[YEAR]
        MOV     BH,DS:[BIS]
        CMP     BL,BH
        JNZ     RE3
        ADD     BH,04H
        CMP     BH,64H
        JNZ     RE5
        MOV     BH,00H
RE5:    MOV     DS:[BIS],BH
        JMP     RE4
RE1:    CMP     AL,1EH
        JNZ     RE2
        CMP     AH,02H
        JZ      RE3
```

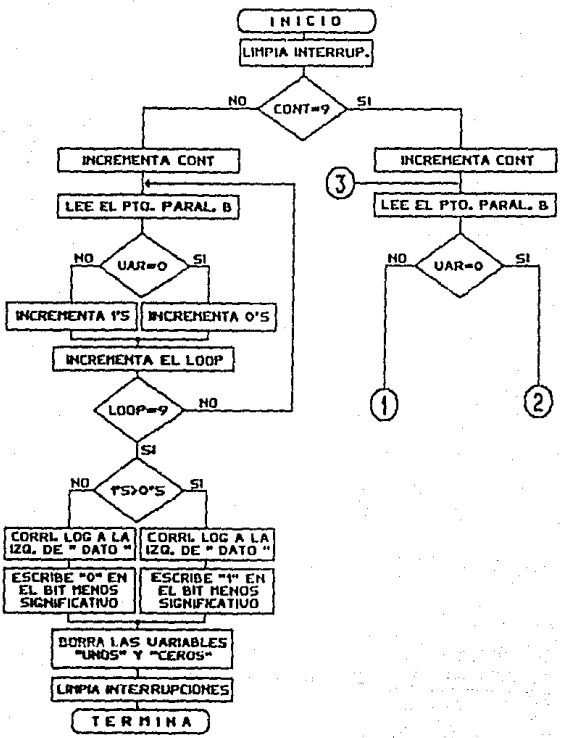
```
RE2:    CMP     AL,1FH
        JNZ     RE6
        CMP     AH,04H
        JZ      RE3
        CMP     AH,06H
        JZ      RE3
        CMP     AH,09H
        JZ      RE3
        CMP     AH,0BH
        JZ      RE3
RE6:    CMP     AL,20H
        JNZ     RE4
RE3:    MOV     AL,01H
        INC     AH
        CMP     AH,0DH
        JNZ     RE4
        MOV     AH,01H
        MOV     BL,DS:[YEAR]
        INC     BL
        CMP     BL,64H
        JNZ     RE7
        MOV     BL,00H
RE7:    MOV     DS:[YEAR],BL
RE4:    MOV     DS:[MES],AH
        MOV     DS:[DIA],AL
        POP     BX
        POP     CX
        POP     AX
        POP     DS
        STI
        IRET
```

RUTINA DE ATENCION AL DETECTOR OPTICO

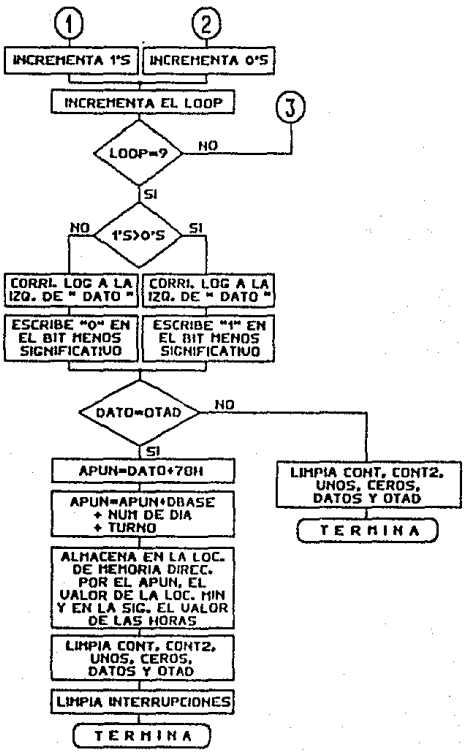
Cada vez que se recibe una interrupción desde el detector óptico se transfiere el control a esta rutina, que se encarga de formar el número de identificación de cada usuario leyendo del puerto paralelo B cada uno de los nueve bits que lo forman. Lo anterior lo hace dos veces, una al introducir la tarjeta almacenando el resultado en la localidad DATO y otra al retirarla almacenando el resultado en la localidad OTAD; posteriomente comprueba si los dos resultados corresponden, de ser así, se despliega el número leído y se manda un dato al puerto paralelo A que activa un zumbador; de lo contrario se limpian las localidades DATO y OTAD.

Con el dato válido se forma un apuntador hacia una localidad de memoria EEPROM donde se almacena el número de minutos del reloj y en la localidad siguiente el número de horas. Finalmente se limpian todas las localidades de memoria empleadas en el proceso de adquisición (DATO, OTAD, CONT, CONT2, UNOS, CEROS, TESD, CONBO y CODIA) y se regresa el control al programa principal. En el reset se limpian las localidades anteriores dejando al sistema listo para atender interrupciones. En las páginas siguientes se muestra el diagrama de bloques.

RUTINA DE ATENCION DEL DETECTOR OPTICO (PARTE 1)



RUTINA DE ATENCION DEL DETECTOR OPTICO (PARTE 2)



Programa en ensamblador:

```
                CLI
                PUSH    CX
                PUSH    DX
                PUSH    AX
                PUSH    DS
                MOV     DX,00H
                MOV     DS,DX
                MOV     AL,DS:[CONT]
                CMP     AL,09H
                JGE     RUT2            ;Este bloque lee  un
                INC     AL              ;dato de 9 bits.
                MOV     DS:[CONT],AL
                MOV     CX,09H
ANA:            MOV     DX,102H
                IN      AL,DX
                MOV     DS:[TESD],AL
                AND     AL,01
                CMP     AL,00
                JNZ     OTI
                MOV     AL,DS:[CEROS]
                INC     AL
                MOV     DS:[CEROS],AL
                LOOP    ANA
                JMP     DE1
OTI:            MOV     AL,DS:[UNOS]
                INC     AL
                MOV     DS:[UNOS],AL
                LOOP    ANA
DE1:            MOV     AL,DS:[CEROS]
                CMP     AL,DS:[UNOS]
                JG      ATE
                MOV     AX,DS:[DATO]
                SHL     AX,01H
                ADD     AX,01H
                MOV     DS:[DATO],AX
                MOV     AL,00H
                MOV     DS:[CEROS],AL
                MOV     DS:[UNOS],AL
                POP     DS
                POP     AX
                POP     DX
                POP     CX
                STI
                IRET
ATE:            MOV     AX,DS:[DATO]
                SHL     AX,01
                MOV     DS:[DATO],AX
                MOV     AL,00H
                MOV     DS:[CEROS],AL
                MOV     DS:[UNOS],AL
```

```
                POP             DS
                POP             AX
                POP             DX
                POP             CX
                STI
                IRET
RUT2:           MOV             AL,DS:[CONT2];Este bloque lee el
                INC             AL            ;mismo dato otra
                MOV             DS:[CONT2],AL;vez para  su com-
                MOV             CX,09H        ;probación.
ANA1:           MOV             DX,102H
                IN              AL,DX
                MOV             DS:[TESD],AL
                AND             AL,01
                CMP             AL,00
                JNZ             OT2
                MOV             AL,DS:[CEROS]
                INC             AL
                MOV             DS:[CEROS],AL
                LOOP            ANA1
                JMP             DE2
OT2:            MOV             AL,DS:[UNOS]
                INC             AL
                MOV             DS:[UNOS],AL
                LOOP            ANA1
DE2:            MOV             AL,DS:[CEROS]
                CMP             AL,DS:[UNOS]
                JG              ITA
                MOV             AX,DS:[OTAD]
                SHR             AX,01
                ADD             AX,100H
                MOV             DS:[OTAD],AX
                MOV             AL,00H
                MOV             DS:[UNOS],AL
                MOV             DS:[CEROS],AL
                MOV             AL,DS:[CONT2]
                CMP             AL,09
                JZ              FIN
                POP             DS
                POP             AX
                POP             DX
                POP             CX
                STI
                IRET
ITA:            MOV             AX,DS:[OTAD]
                SHR             AX,01
                MOV             DS:[OTAD],AX
                MOV             AL,00H
                MOV             DS:[UNOS],AL
                MOV             DS:[CEROS],AL
                MOV             AL,DS:[CONT2]
                CMP             AL,09
                JZ              FIN
                POP             DS
                POP             AX
```

```
                POP             DX
                POP             CX
                STI
                IRET
ERR:            MOV             DX,401H
                MOV             AL,0CDH
                OUT             DX,AL
                MOV             AX,00
                MOV             DS,AX
                MOV             DS:[CONT],AL
                MOV             DS:[CONT2],AL
                MOV             DS:[CONBO],AL
                MOV             DS:[DATO],AX
                MOV             DS:[OTAD],AX
                POP             DS
                POP             AX
                POP             DX
                POP             CX
                STI
                IRET
FIN:            MOV             DX,201H
                MOV             AL,2
                OUT             DX,AL
                MOV             AX,DS:[DATO]
                CMP             AX,DS:[OTAD]
                JNZ             ERR
                PUSH            BX
                PUSH            DS
                MOV             DX,8000H
                MOV             DS,DX
                MOV             BX,DS:[CODIA]
                CMP             BX,0AH
                JG              OV1
                CALL            DENUM
                MOV             CX,0FFFH
F3:             NOP
                LOOP            F3
                MOV             BX,78H
                MUL             BX
                MOV             BX,AX
                MOV             AX,DS:[CODIA]
                MOV             DX,0CH
                MUL             DX
                SUB             AX,0CH
                ADD             BX,AX
                POP             DS
                MOV             AL,DS:[TESD]
                AND             AL,0FCH
                CMP             AL,18H
                JNZ             EXTE
                ADD             BX,0AH
                MOV             AL,0BH
                JMP             CAR
EXTE:           CMP             AL,14H
                JNZ             TARS
```

```
                ADD             BX,8
                MOV             AL,0BH
                JMP             CAR
TARS:           CMP             AL,10H
                JNZ             TARE
                ADD             BX,6
                MOV             AL,13H
                JMP             CAR
TARE:           CMP             AL,0CH
                JNZ             MANS
                ADD             BX,4
                MOV             AL,13H
                JMP             CAR
MANS:           CMP             AL,8
                JNZ             MAE
                ADD             BX,2
                MOV             AL,1BH
                JMP             CAR
MAE:            CMP             AL,4
                JNZ             ERR1
                MOV             AL,1BH
                JMP             CAR
OV1:            CALL            OVER
                MOV             AX,00
                MOV             DS,AX
                MOV             DS:[CONT],AL
                MOV             DS:[CONT2],AL
                MOV             DS:[CONBO],AL
                MOV             DS:[DATO],AX
                MOV             DS:[OTAD],AX
                POP             DS
                POP             BX
                POP             DS
                POP             AX
                POP             DX
                POP             CX
                STI
                IRET
CAR:            MOV             CX,0FFFH
CAR1:           NOP
                LOOP            CAR1
                CALL            DETUR
                MOV             AX,DS:[MIN]
                MOV             DX,8000H
                MOV             DS,DX
                MOV             DS:[BX],AX
                MOV             AX,00
                MOV             DS,AX
                MOV             DS:[CONBO],AL
                MOV             DS:[CONT],AL
                MOV             DS:[CONT2],AL
                MOV             DS:[DATO],AX
                MOV             DS:[OTAD],AX
                POP             BX
                POP             DS
```

```
                POP     AX
                POP     DX
                POP     CX
                STI
                IRET
ERR1:           JMP     ERROR
```

CAPITULO V

DISEÑO DE LA ENVOLVENTE

Introducción.

En el proceso de desarrollo y construcción de equipos electrónicos, el diseño y elaboración de la envolvente debe ser hecha bajo ciertos criterios técnicos y funcionales que permitan al usuario la mayor facilidad en la instalación, operación y mantenimiento del equipo.

Sin embargo, los criterios técnicos y funcionales no son los únicos que se deben de tomar en cuenta para el diseño de la envolvente, sino también la complejidad en la manufactura y el costo de los materiales empleados.

Si solamente consideramos los primeros dos criterios, se tendría un diseño con una alta funcionalidad pero tendríamos un producto con un elevado costo de producción. Pero si el diseño se basara en obtener una envolvente con un bajo costo de producción, entonces caeriamos en un producto de baja calidad y/o funcionalidad.

Por otra parte, la forma y calidad de los acabados dada a la envolvente, tendrá una repercución importante en la introducción a nivel comercial de éste equipo. Es claro que un equipo electrónico elaborado con tecnologica moderna no tendrá una buena aceptación si su envolvente presenta características de diseño

obsoleto y poco funcional.

En relación a nuestro proyecto, debido a que se trató de un prototipo, se diseño la envolvente utilizando los criterios técnicos y de funcionalidad y en cierto modo se buscó una simplicidad en la manufactura.

V.1 .- DISEÑO.

Para el diseño de la envolvente se tomaron en cuenta las características del sistema indicadas en el capítulo I. Estas características indican que, el sistema sea capaz de proporcionar información al usuario a través de un despliegue y en caso necesario poder introducir información a través de un teclado, el sistema debe aceptar las tarjetas checadoras para leer el número de empleado y finalmente debe comunicarse con una computadora personal por medio de un conector DB-25.

Por otra parte, la envolvente debe de contener a todos los elementos del sistema (tarjeta madre, tarjeta de memoria, tarjeta de despliegue-teclado, la fuente de alimentación y los detectores ópticos) y poder ser instalado en la pared a una determinada altura para el fácil acceso de los empleados al momento de llegar a checar su tarjeta, son características no especificadas pero que el sistema tiene que cumplir por funcionalidad del mismo.

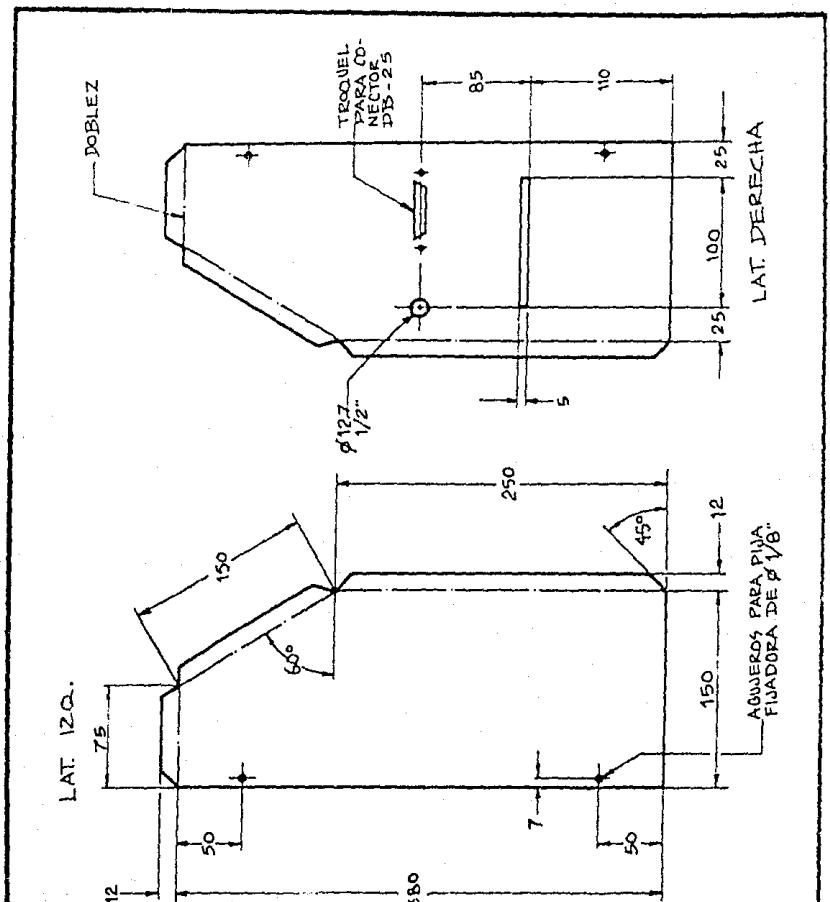
Por lo anterior se pensó en una envolvente en dos partes:

- Una es la tapa o frente que es fácilmente removible para el fácil mantenimiento preventivo y correctivo del sistema. La guía para la tarjeta checadora, el push boton selector de turno y entrada/salida así como los detectores ópticos de código, se encuentran colocados instaladosen ésta parte de la envolvente. También cuenta con las aberturas necesarias para el despliegue y el teclado.

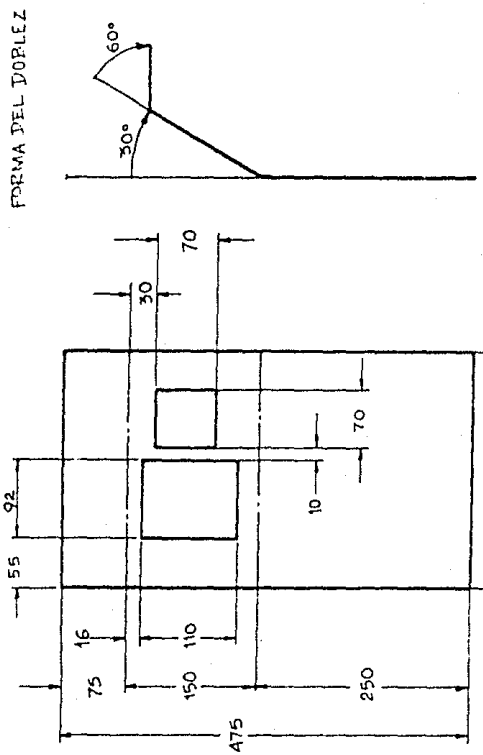
- Y una parte trasera, que se encontrará fija a la pared, y en donde se encuentran montadas el resto de los elementos del sistema

Debido al equipo y material disponible, se decidió utilizar lámina de calibre 18 para toda la envolvente mecánica.

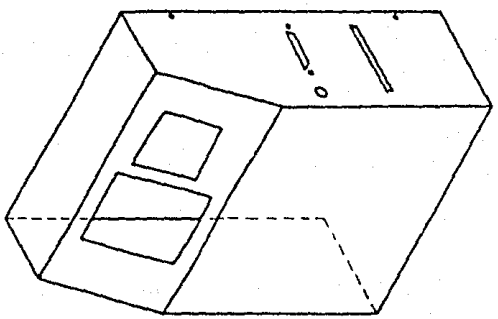
En las páginas siguientes, se presenta un bosquejo general del diseño realizado y el diseño en detalle de las partes involucradas.



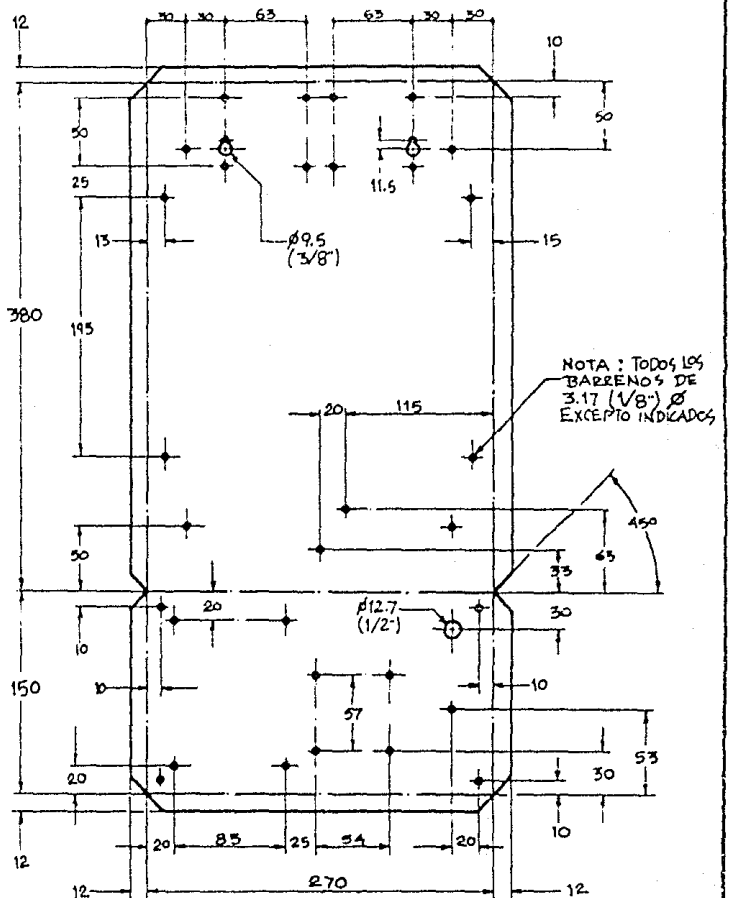
CENTRO DE INSTRUMENTOS		ESC. 1:33	ACOT. mm.
		FECHA: SEPT. 90	DIB. Nº
NOMBRE: LATERALES IZQ. Y DERECHA	PROYECTO: RELOJ CHECADOR	DIBUJO: G. García	
MATERIAL: LAMINA DE FIERRO CAL. N. 18		DISEÑO: EDGAR MANDUJANO	
CANTIDAD: 1 C/U	DEPTO. SEC. TECNICA	REVISO: MAURICIO GARCIA	



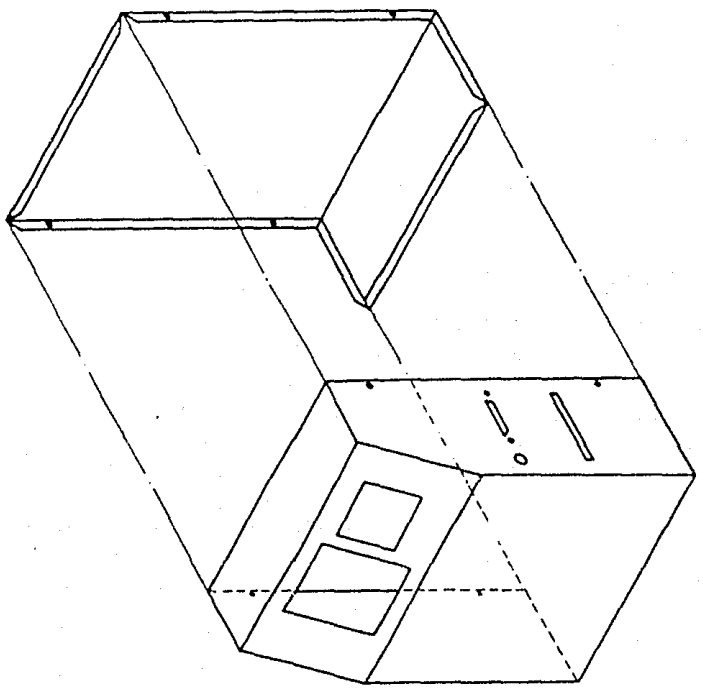
CENTRO DE INSTRUMENTOS		ESC. 1:50	
		ACOT. m.m.	
		FECHA: SEPT. 90	DIB. Nº
NOMBRE: TAPA SUPERIOR	PROYECTO: RELOJ CHECADOR	DIBUJO: G. García	
MATERIAL: LAMINA DE FIERRO CAL. No 18		DISEÑO: EDGAR MANDUJANO	
CANTIDAD: 1	DEPTO. SEC. TECNICA	REVISO: MAURICIO GARCIA	



CENTRO DE INSTRUMENTOS		ESC. 1:50	
		ACOT. SIN	
		FECHA: SEPT. 90	DIB. Nº
NOMBRE: ISOMETRICO, TAPA SUPERIOR Y LATERALES	PROYECTO: RELOJ CHECADOR	DIBUJO: G. García	
MATERIAL: INDICADO		DISEÑO: EDGAR MANDUJANO	
CANTIDAD: INDICADO	DEPTO: SEC. TECNICA	REVISO: MAURICIO GARCIA	

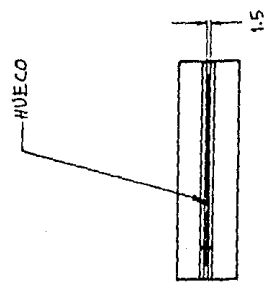


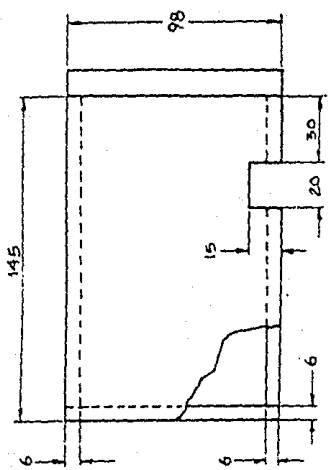
CENTRO DE INSTRUMENTOS		ESC 1:33	ACOT. mm.
		FECHA: SEPT. 90	DIB. Nº
NOMBRE: PARTE TRASERA E INFERIOR	PROYECTO: RELOJ CHECADOR	DIBUJO: G. G. G	DISEÑO: EDGAR MANDUJANO
MATERIAL: LAMINA DE FIERRO CAL. No 18			
CANTIDAD: 1	DEPTO. SEC. TEC.		REVISO: MAURICIO GARCIA

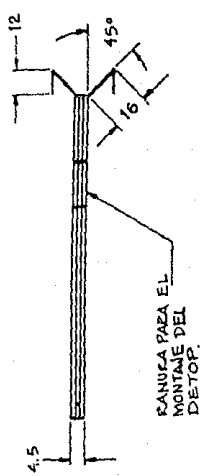


CENTRO DE INSTRUMENTOS		ESC. 1:50	
		ACOT. SIN	
		FECHA: SEPT. 90	DIB. Nº
NOMBRE: ISOMETRICO DE LA UNION DE LA ENVOLVENTE	PROYECTO: RELOJ CHECADOR	DIBUJO: G.G.G.	
MATERIAL: INDICADO		DISEÑO: EDGAR MANDUJANO	
CANTIDAD: INDICADO	DEPTO. SECRETARIA TECNICA	REVISO: MAURICIO GARCIA	

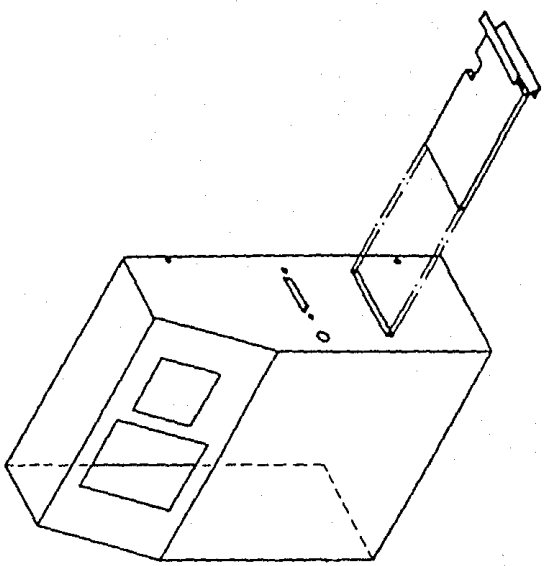
SOPORTE Y ENTRADA PARA LA TARJETA CHECADORA.



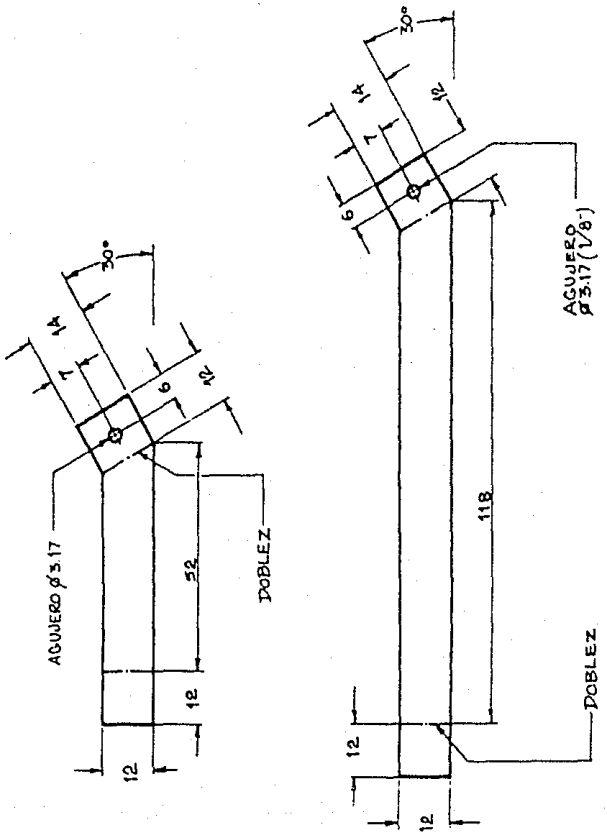




CENTRO DE INSTRUMENTOS		ESC. 1:2	
		ACOT. mm.	
NOMBRE: INDICADO	PROYECTO: RELOJ CHECADOR	FECHA: SEPT. 90	DIB. Nº
		DIBUJO: C.G.G.	
MATERIAL: LÁMINA DE FIERRO CAL. No 18		DISEÑO: EDGAR MANDUJANO	
CANTIDAD: 1	DEPTO. SEC. TECNICA	REVISO: MAURICIO GARCIA	

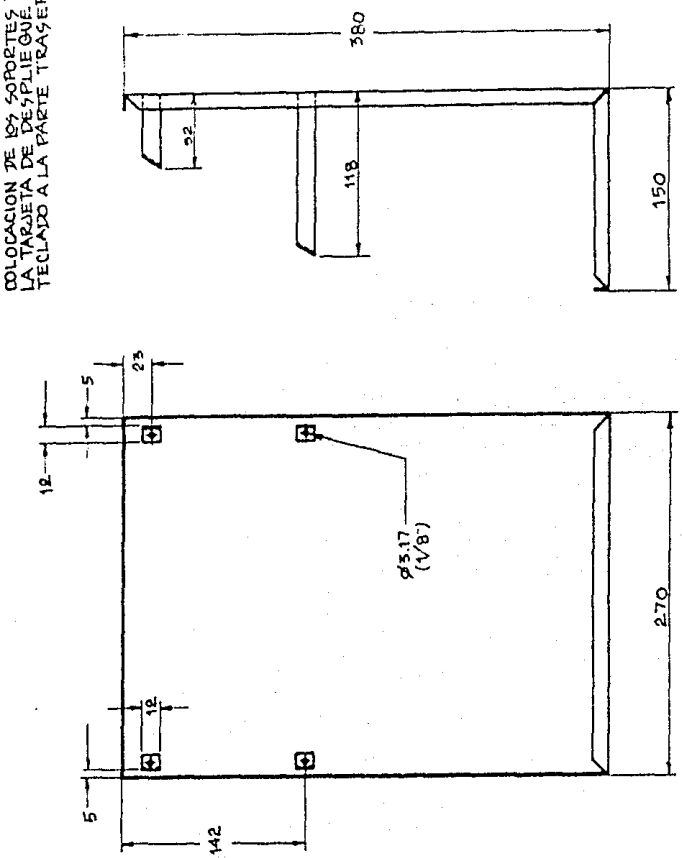


CENTRO DE INSTRUMENTOS		ESC. 1:50	
		ACOT. SIN	
		FECHA: SEPT. 90	DIB. Nº
NOMBRE: ISOMETRICO DE LA COLOCACION DEL SOPORTE PARA LA TARJETA	PROYECTO: RELOJ CHECADOR	DIBUJO: G.G.G.	
MATERIAL: INDICADO		DISEÑO: EDGAR MANDUJANO	
CANTIDAD: INDICADO	DEPTO. SECRETARIA TECNICA	REVISO: MAURICIO GARCIA	

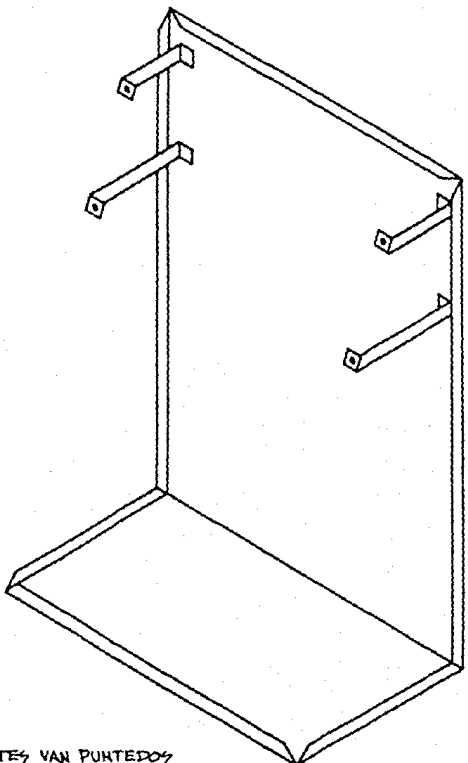


CENTRO DE INSTRUMENTOS		ESC 1:1	
		FECHA: SEPT. 90	ACOT. mm.
NOMBRE: SOPORTE PARA TECLADO Y DESPLIEGUE IZQ. Y DER.	PROYECTO: RELOJ CHECADOR	DIBUJO: G.G.G.	DIB. Nº
MATERIAL: LAMINA DE FIERRO CAL. No 18		DISEÑO: EDGAR MANDUJANO	
CANTIDAD: 2 JUEGOS IZQ. Y DER.	DEPTO. SECRETARIA TECNICA	REVISO: MAURICIO GARCIA	

COLOCACION DE LOS SOPORTES PARA LA TARJETA DE DESPLIEGUE Y TECLADO A LA PARTE TRASERA.

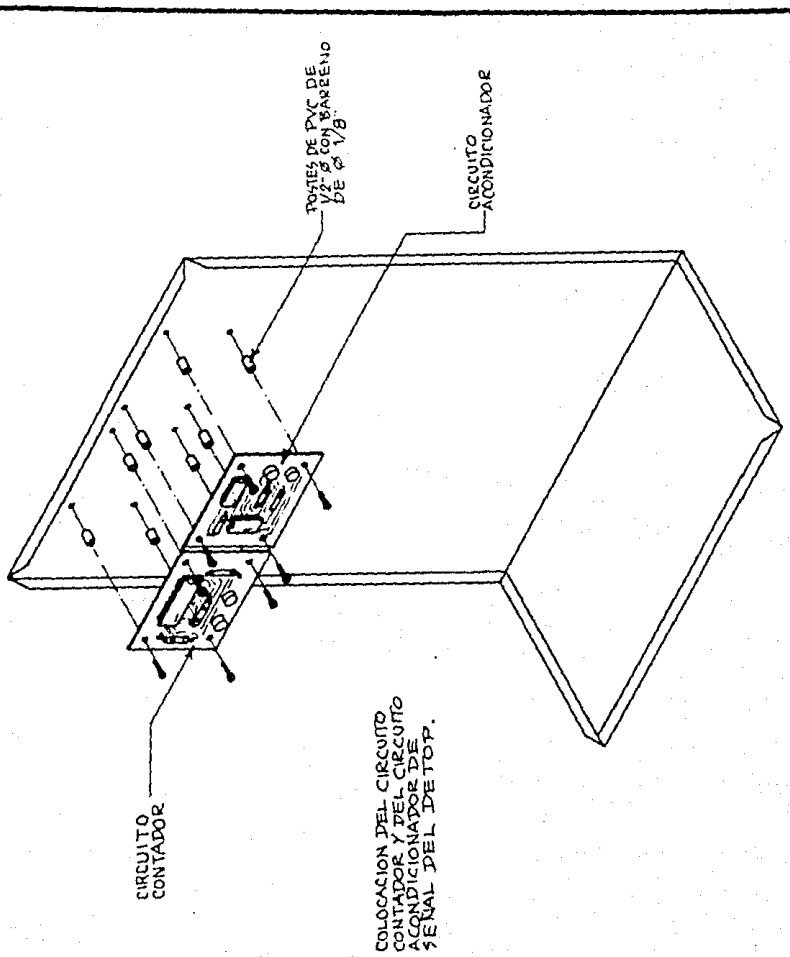


CENTRO DE INSTRUMENTOS		ESC. 1:33	ACOT.
		FECHA: SEPT. 90	DIB. Nº
NOMBRE: INDICADO	PROYECTO: RELOJ CHECADOR	DIBUJO: G.G.G.	
MATERIAL: INDICADO		DISEÑO: EDGAR MANDUJANO	
CANTIDAD: 1	DEPTO. SEC. TECNICA	REVISO: MAURICIO GARCIA	



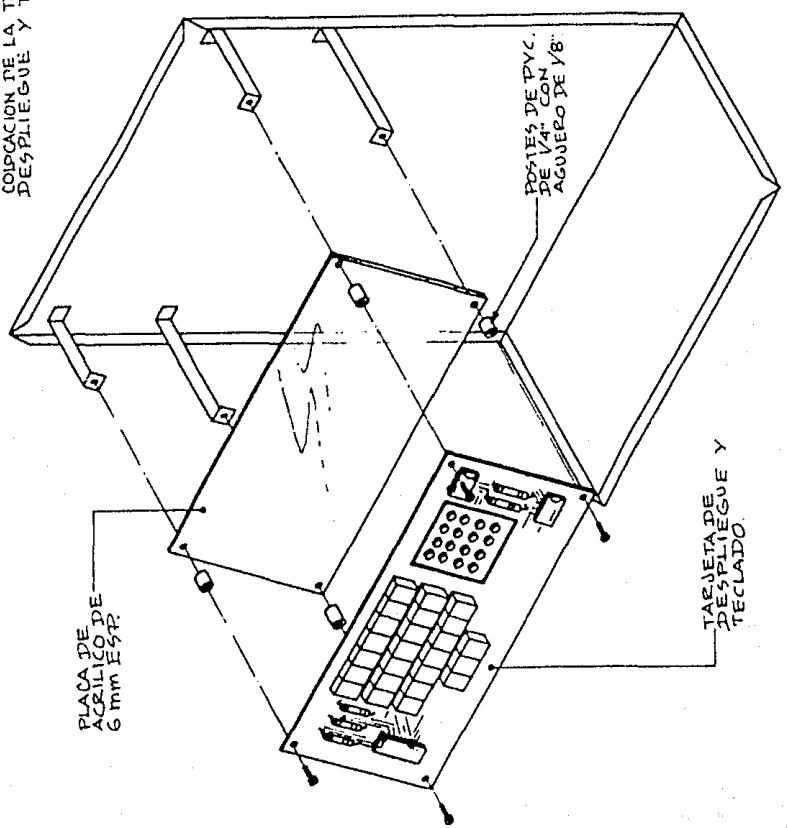
NOTA :
LOS SOPORTES VAN PUNTEDOS
A LA ENVOLVENTE.

CENTRO DE INSTRUMENTOS		ESC. SIN	
		ACOT. SIN	
		FECHA: SEPT. 90	DIB. Nº
NOMBRE: ISOMETRICO PARTE TRASERA E INFERIOR CON SOPORTES	PROYECTO: RELOJ CHECADOR	DIBUJO: G.G.G.	
MATERIAL: INDICADO		DISEÑO: EDGAR MANDUJANO	
CANTIDAD: 1	DEPTO. SEC. TECNICA	REVISO: MAURICIO GARCIA	

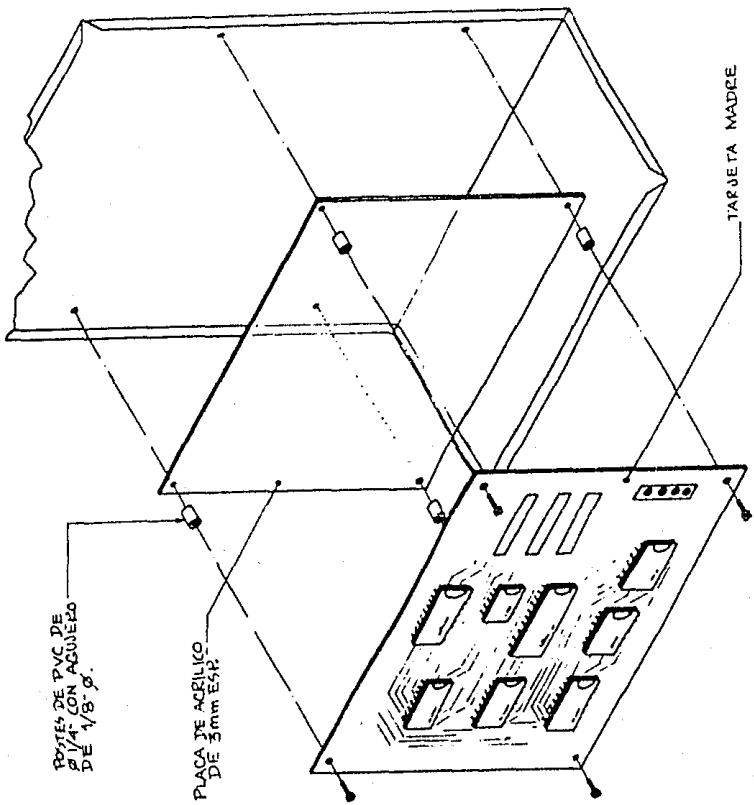


CENTRO DE INSTRUMENTOS		ESC SIN	ACOT. SIN
		FECHA: SEPT. 90	DIB. Nº
NOMBRE: INDICADO	PROYECTO: RELOJ CHECADOR	DIBUJO: G.G.G	
MATERIAL: ———		DISEÑO: EDGAR MANDUJANO	
CANTIDAD: 1	DEPTO. SECRETARIA TECNICA	REVISO: MAURICIO GARCIA	

COLOCACION DE LA TARJETA DE DESPLIEGUE Y TECLADO.



CENTRO DE INSTRUMENTOS		ESC. S/N	
		ACOT. S/N	
		FECHA: SEPT. 90	DIB. Nº
NOMBRE: INDICADO	PROYECTO: RELOJ CHECADOR	DIBUJO: G. G. G.	
MATERIAL: ———		DISEÑO: EDGAR MANDUJANO	
CANTIDAD: ———	DEPTO. SECRETARIA TECNICA	REVISO: MAURICIO GARCIA	



CENTRO DE INSTRUMENTOS

ESC. S/N	ACOT. S/N
FECHA: SEPT. 90	DIB. Nº
DIBUJO: G.G.G.	
DISEÑO: EDGAR MANDUJANO	
REVISO: MAURICIO GARCIA	

NOMBRE: COLOCACION DE LA TARJETA MADRE

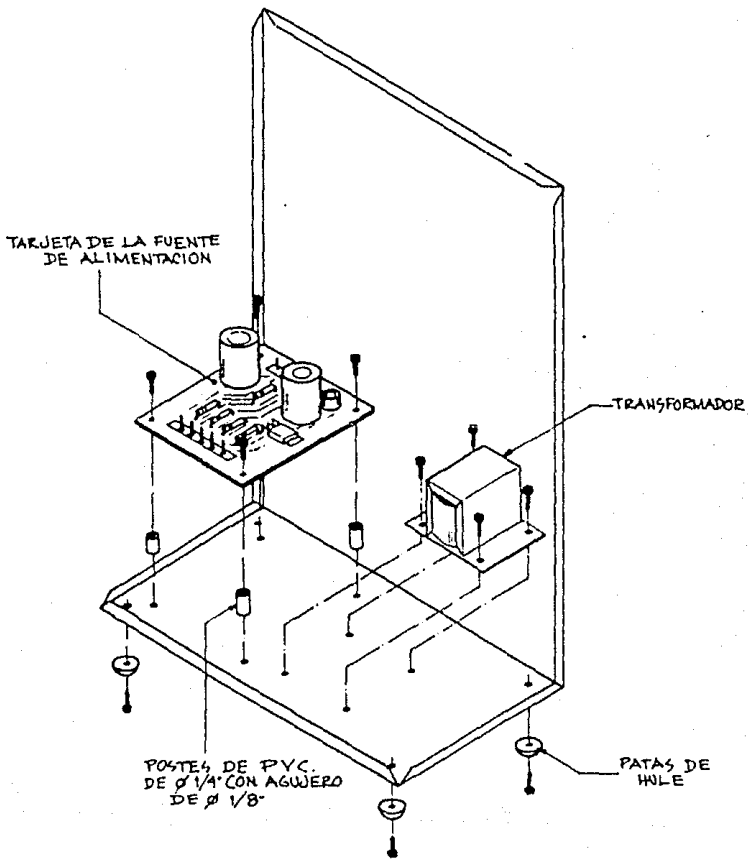
MATERIAL: _____

CANTIDAD: _____

PROYECTO: RELOJ CHECADOR

DEPTO. SECRETARIA TECNICA

UBICACION DEL TRANSFORMADOR
Y DE LA FUENTE DE ALIMENTACION



CENTRO DE INSTRUMENTOS		ESC. SIN	
		ACOT. SIN	
	FECHA: SEPT. 90	DIB. Nº	
NOMBRE: INDICADO	PROYECTO: RELOJ CHECADOR	DIBUJO: G.G.G.	
MATERIAL: _____		DISEÑO: EDGAR MANDUJANO	
CANTIDAD: _____	DEPTO. SECRETARIA TECNICA	REVISO: MAURICIO GARCIA	

CAPITULO VI

OBSERVACIONES Y CONCLUSIONES.

Durante el desarrollo de este proyecto nos dimos cuenta de la gran flexibilidad de un sistema controlado por un microprocesador, ya que una vez que se cuenta con la arquitectura básica, sólo es necesario realizar los programas apropiados para realizar nuevas funciones o modificar las existentes.

Distinguimos que las fallas debidas a programación siguen, en la mayoría de los casos, una secuencia definida y repetible, aunque puede darse el caso de alguna falla en la programación que genere comportamientos erráticos diversos en el equipo como cuando no se tiene un manejo correcto del "stack". En ocasiones fué necesario insertar retardos en la ejecución de algún programa que interactuaba con dispositivos periféricos para dar tiempo a estos de realizar su tarea.

En cuanto a la electrónica, el diseño del circuito impreso, cuando se tiene una alta densidad de dispositivos montados en él, es de suma importancia, aunque, utilizando algunas técnicas sencillas, como las que mencionamos en el inciso III.5, se obtienen resultados satisfactorios. El uso extensivo del analizador de estados lógicos y del osciloscopio facilitan en gran medida la localización y corrección de fallas en estos equipos, ya que el primero permite observar el comportamiento de las señales en condiciones preestablecidas y en el segundo se puede observar la forma de la señales que están presentes en las líneas del circuito impreso.

Actualmente contamos con un prototipo del equipo desarrollado en este trabajo, así como con la programación necesaria para su operación en la aplicación planteada. Este prototipo ha estado en evaluación en laboratorio durante aproximadamente dos meses y no ha presentado fallas significativas en su operación, aunque todavía es necesaria una etapa final de puesta a punto en cuanto a la velocidad de respuesta del lector óptico se refiere para su posible comercialización

En general, el sistema dá una solución de bajo costo a la necesidad planteada realizando satisfactoriamente todas las funciones para las que fué diseñado, con un equipo funcional de fácil operación e instalación y un acabado que, como comprobamos en el VI Congreso de la Sociedad Mexicana de Instrumentación, llama la atención de la gente. Recuérdese que la aplicación desarrollada sólo es una de las múltiples que puede tener un sistema de adquisición de datos como el que se implementó.

La siguiente etapa de este proyecto es la de transferencia tecnológica hacia la industria privada para lo cual se están haciendo las gestiones necesarias con el Centro de Innovación Tecnológica, organismo encargado de ésta tarea en la Universidad.

Por último, los autores esperamos que el presente trabajo pueda ayudar en algo a quien se inicie en el desarrollo de sistemas basados en microprocesadores especialmente el Intel 8086/8088.

ANEXO A

ESTUDIO DEL MICROPROCESADOR Y SUS PERIFERICOS.

Introducción.

Los circuitos integrados se clasifican en dos categorías generales : LINEALES Y DIGITALES. Los circuitos integrados lineales operan con señales continuas para producir funciones electrónicas tales como amplificadores y comparadores de voltaje. Los circuitos digitales, operan con señales binarias y se hacen de compuertas digitales binarias interconectadas.

A medida que se mejora la tecnología de los circuitos integrados, el número de compuertas que pueden encapsularse en un dado de silicón, ha aumentado considerablemente. La forma de diferenciar aquellos circuitos integrados que tengan unas pocas compuertas, con las que tienen cientos de compuertas, es referir al circuito como un elemento de pequeña, mediana o gran integración. Por lo anterior tenemos circuitos SSI, MSI, LSI y VLSI.

Unas pocas compuertas en un circuito constituyen un SSI (pequeña escala de integración). Para poder clasificar a un elementos como MSI (mediana escala de integración), el circuito debe de cumplir una función lógica completa y tener una complejidad de 10 a 100 compuertas. Un elemento LSI (alta escala de integración) cumple con una función lógica con más de 100 compuertas. Finalmente encontramos los circuitos VLSI (muy alta escala de integración) que contienen miles de compuertas en un

solo circuito integrado.

Un microprocesador es un circuito VLSI o un grupo de circuitos los cuales realizan las operaciones aritméticas, lógicas y de control para las instrucciones y las secuencias que se le indiquen. Un microprocesador consiste de un ALU, acumuladores y registros, los cuales mantienen datos temporales para operaciones rápidas, el control del decodificador-secuenciador interpreta los comandos del programa y envía ordenes detalladas, llamadas microinstrucciones, a cada uno de los elementos del CPU para que realicen la operación indicada.

El microprocesador 8088 es un circuito integrado fabricado con tecnología de semiconductor de metal-óxido de alto funcionamiento (HMOS) con aproximadamente 29,000 transistores. El 8088 sólo se diferencia del 8086 en el canal de datos, que es de 8 bits externos y 16 bits internos. La arquitectura interna y el software son similares y son totalmente compatibles.

La tendencia de los fabricantes a producir versiones de 8 bits intenta simplificar el paso de los usuarios hacia los 16 bits, aprovechando los desarrollos que ya existen en hardware y software para los microprocesadores anteriores. Es por ello que la razón de la existencia del 8088 con un canal de 8 bits es establecer una continuidad entre el 8086 que tiene 16 bits en su canal de datos y los anteriores microprocesadores elaborados por INTEL. Una ventaja es que el 8088 puede reemplazar a uno de los primeros microprocesadores de 8 bits en un sistema ya existente, aumentando su capacidad en cuanto al conjunto de operaciones que realiza y la capacidad de direccionamiento de la memoria.

En este estudio del microprocesador 8088, primeramente se hablará de su arquitectura interna, y con ello de los registros internos y sus unidades básica. Posteriormente se mencionarán las terminales del 8088 y las señales que manejan, además de los modos de operación y los principales ciclos que realiza. Otros puntos a trarar serán los modos de direccionamiento y los tipos de interrupciones para que, finalmente, se vea el set o conjunto de instrucciones del microprocesador.

1.- ARQUITECTURA INTERNA

El 8088 está dividido en dos unidades básicas que son la **UNIDAD DE EJECUCION (EU)** y la **UNIDAD DE INTERFACE DEL CANAL (BIU)**. Ambas unidades operan asincronamente para darle un mecanismo de captura y de ejecución de instrucciones al mismo tiempo.

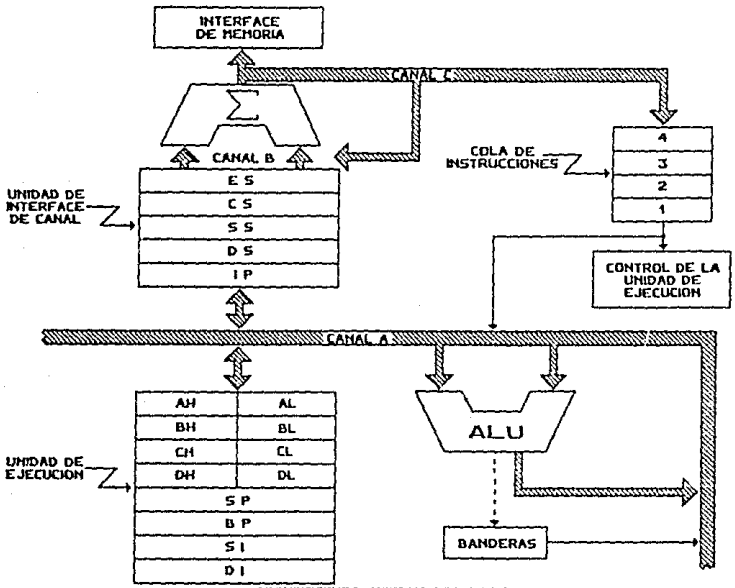
La BIU se encarga de efectuar todas las operaciones relacionadas con los canales, como son transferencia de datos entre el microprocesador y la memoria ó dispositivos de entrada/salida (E/S), la captura de instrucciones mediante el método llamado "por cola" (stack), relocalización de direcciones y control del canal del sistema. La BIU está formada por los registros de segmento, los registros de comunicación interna, el apuntador de instrucciones, la cola de instrucciones, un sumador el cual se usa para generar la dirección física que es enviada por el canal de direcciones; y la lógica de control del canal.

La BIU usa un mecanismo conocido como COLA DE INSTRUCCIONES para implementar una arquitectura " PIPELINE ". Esta cola permite la precaptura de hasta 4 bytes de código de operación. Siempre que exista uno ó más bytes vacíos en la cola y que al mismo tiempo la EU no este solicitando una lectura ó escritura de operandos desde la memoria ó desde algún dispositivo de E/S, la BIU está libre para la precaptura de la próxima instrucción secuencial; la estructura de la cola permite que lo primero que entra es lo primero que sale, esto es, que la primera instrucción que entra es la primera en ser atendida. A este tipo de estructura se le conoce como FIFO (First Input, First Output). Si la cola está llena y la EU no está solicitando acceso a operandos de la memoria, la BIU realiza ciclos de canal. Más aún, si la BIU esta ocupada en el proceso de captura de una instrucción cuando la EU solicita una lectura o escritura de operandos, la BIU primero completa el ciclo de canal de captura de instrucción antes de iniciar el ciclo de lectura/escritura de datos.

La EU es responsable de decodificar y ejecutar todas las instrucciones, y está formada por un ALU, un registro de banderas, de 8 registros de propósito general, registros temporales y la lógica de control de la cola de instrucciones. Esta unidad no tiene conexión directa con el canal del sistema pues las instrucciones son obtenidas de la cola de instrucciones que le proporciona la BIU; cuando una instrucción requiere de que un dato sea leído de ó almacenado en memoria ó dispositivo periférico, la EU solicita a la BIU que realice los ciclos de canal de lectura o escritura. Durante la ejecución de una instrucción, la EU verifica el registro de las banderas de estado y control, actualizándolas en base a los resultados de la ejecución de dicha instrucción. Si la cola de instrucciones está vacía, la EU espera a que el próximo byte de instrucción sea capturado y recorrido a la parte alta de la cola.

Cuando la EU ejecuta una INSTRUCCION DE SALTO, la BIU automáticamente borra la cola de instrucciones y comienza a

capturar instrucciones desde la nueva localidad. Finalmente, todas las instrucciones manejadas por la EU son de 16 bits, por lo que es necesario que la BIU realice una relocalización de éstas para poder tener acceso a todo el espacio de direcciones.



ARQUITECTURA INTERNA DEL 8088

Las dos unidades pueden funcionar independientemente la una de la otra en muchas condiciones. El resultado es que el tiempo empleado para la busqueda de instrucciones es muy pequeño ya que esta búsqueda se efectúa mientras la EU procesa instrucciones.

REGISTROS DE PROPOSITO GENERAL.

Dentro de la unidad de ejecución (EU) se encuentran 8 registros de 16 bits, que están divididos en dos conjuntos de 4 registros cada uno :

GRUPO HL. El cual contiene cuatro registros de datos, que son :

- El registro acumulador (AX),
- El registro base (BX),
- El registro contador (CX), y
- El registro de datos (DX).

GRUPO PI. Este grupo está fromado por dos registros apuntadores y dos registros de indice y son :

- El registro apuntador del stack (SP),
- El registro apuntador de base (BP),
- El registro Indice de fuente (SI), y
- El registro Indice de destino (DI).

Los registros de datos también pueden ser direccionados separadamente como registros de 8 bits y pueden servir como acumuladores secundarios en la mayoría de las operaciones.

REGISTROS DE SEGMENTO

Para poder direccionar un Megabyte de memoria en segmentos de 64 Kbytes, el microprocesador tiene que hacer uso de los registros de segmento, los cuales se localizan dentro de la Unidad de Interface del Canal (BIU). Estos registros de 16 bits son :

- El registro de segmento de código (CS),
- El registro de segmento de datos (DS),
- El registro de segmento del stack (SS), y
- El registro de segmento extra (ES).

El microprocesador tiene acceso directo a estos cuatro registros, dentro de los cuales están contenidas las **DIRECCIONES BASE DE CADA SEGMENTO** .

APUNTADOR DE INSTRUCCION

El apuntador de instrucción (IP) es un registro de 16 bits que identifica la localidad de memoria en la que se encuentra la próxima instrucción que será capturada. Es similar a un contador de programa (PC), sin embargo, el apuntador de instrucción contiene un desplazamiento. Este desplazamiento es combinado con el contenido del registro de segmento de código para generar la dirección física de la instrucción.

REGISTRO DE BANDERAS Y CONTROL

Este registro tiene 16 bits y se encuentra en la Unidad de Ejecución (EU) y solo 9 de sus bits están implementados. Seis de estos bits representan las banderas de estado.

15																1
	X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

Estas banderas de estado son:

- Acarreo (CF),
- Paridad (PF),
- Acarreo auxiliar (AF),
- Cero (ZF),
- Signo (SF), y
- Sobreflujo (OF).

Los otros tres bits son banderas de control :

TRAMPA (TF) : Si esta bandera presenta un 1 lógico, el microprocesador entra automáticamente en el modo paso a paso. En este modo se genera una interrupción interna cada véz que se ejecuta una instrucción. Esta característica se puede utilizar para corregir programas.

HABILITACION DE INTERRUPCIONES (IF) : Encendiendo esta bandera, el microprocesador tiene la capacidad para atender interrupciones mascarables pero si está apagada no atenderá interrupciones. Esta bandera no tiene efecto sobre las interrupciones no mascarables.

DIRECCION (DF) : Si esta bandera está encendida, provoca que los apuntadores de las instrucciones de manejo de las cadenas (SI y DI) son decrementadas automáticamente. Si la bandera está apagada, los apuntadores son incrementados.

ORGANIZACION DE LA MEMORIA

El microprocesador 8088 posee 20 bits de direcciones de memoria. La memoria está organizada como un arreglo lineal de un Megabyte, direccionado desde 00000_H hasta $FFFFF_H$. La memoria está lógicamente dividida en segmentos de código, datos, datos extra y cola (stack) de 64K cada uno.

Todas las referencias a la memoria son hechas relativas a una dirección base contenida en un registro de segmento de alta velocidad.

Los tipos de segmento son seleccionados basándose en las direcciones necesitadas en el programa. Los operandos de 16 bits pueden ser localizados en direcciones pares ó nones. Para operandos de direcciones y datos, el byte menos significativo de la palabra está almacenada en la localidad inferior (en valor) y el byte más significativo está en la siguiente localidad. La BIU ejecuta automáticamente dos ciclos de Fetch ó escritura para operandos de 16 bits.

Ciertas localidades de la memoria están reservadas para operaciones específicas del CPU. Las localidades direccionadas desde la $FFFF0_H$ hasta la $FFFFF_H$ están reservadas para operaciones, incluyendo el salto a la rutina de inicialización del sistema. Después del RESET, siempre empieza a ejecutar la rutina localizada en $FFFF0_H$. Las localidades de la 00000_H hasta la $003FF_H$ están reservadas para operaciones de interrupción.

2.- TERMINALES DEL 8088.

El microprocesador 8088 está encapsulado en un paquete de 40 terminales, de las cuales algunas manejan doble señal, dependiendo del modo de operación en el que esté trabajando el microprocesador.

El canal de datos tiene una amplitud de 8 bits y se encuentra multiplexado con el canal de direcciones (AD_0 - AD_7), entendiendo por multiplexado el uso del mismo conjunto de líneas para enviar grupos de señales diferentes. El 8088 tiene un canal de direcciones de 20 bits que le permite una capacidad de direccionamiento de 1 Megabyte de memoria, además puede direccionar 64 Kbytes de puertos de entrada/salida. Este microprocesador trabaja con un reloj de 5 MHz.

MODOS DE OPERACION.

NOTA : TODAS LAS SEÑALES SIGUIENTES QUE TENGAN UN ASTERISCO (*) SIGNIFICARA QUE SON NEGADAS.

El 8088 cuenta con dos modos de operación y son conocidos como el modo de sistema mínimo y modo de sistema máximo. El modo de sistema mínimo se selecciona cuando un 1 lógico es aplicado a la terminal de entrada MN/MX* (terminal 33), cambiando a un 0 lógico esta misma terminal se habrá seleccionado el modo máximo de operación.

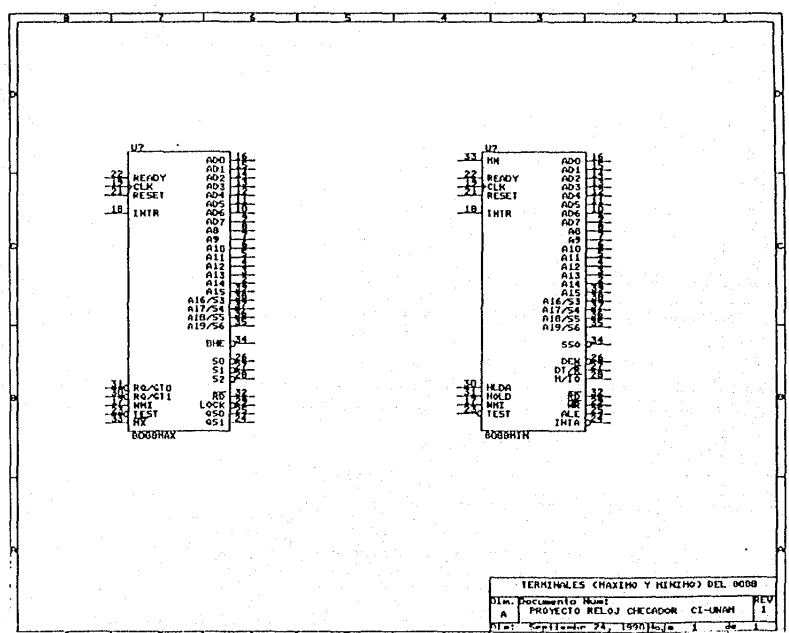
El modo de sistema mínimo es utilizado para trabajar en sistemas pequeños, donde el microprocesador proporciona todas las líneas de control de los canales para el control de la memoria y dispositivos periféricos y el modo de sistema máximo se utiliza en sistemas grandes con más de un microprocesador, en donde es necesario tener un controlador de canal. Las señales que son comunes para ambos modos y aquellas que son únicas en cada modo son las siguientes :

SEÑALES COMUNES.

AD_0-AD_7 (bidireccional, 3 estados).
Canal de direcciones/datos : Estas líneas constituyen el canal multiplexado de direcciones de memoria/periféricos (en T1) y datos (en T2, T3, Tw y T4).

A_{15}-A_8 (salida, 3 estados).
Canal de direcciones : Estas líneas proporcionan los bits del 8 al 15 para completar el ciclo de canal (T1-T4).

A_{19}/S_6,A_{18}/S_5,A_{17}/S_4,A_{16}/S_3 (salidas, 3 estados).
Direcciones/estados : Durante T1, estas son las cuatro líneas de dirección más significativas para operaciones en la memoria. Durante operaciones de E/S, estas líneas están en un estado BAJO. Durante operaciones de memoria y E/S, la información de estado está disponible en estas líneas durante T2, T3, Tw y T4. Los bits S_3 y S_4 juntos forman un código binario de dos bits que identifican cual de los registros de segmento fué usado para


U7
8088MAX
U7
8088MIN
TERMINALES (MAXIMO Y MINIMO) DEL 8088
PROYECTO RELOJ CHECADOR CI-UNAM

generar la dirección física que se envió por el canal de direcciones durante el ciclo de canal actual. Dicho código es el siguiente :

S_4	S_3	REGISTRO DE SEGMENTO
0	0	EXTRA.
0	1	STACK.
1	0	CODIGO.
1	1	DATOS.

La línea S_5 muestra el estado de la bandera de interrupciones. El estado de la línea S_6 es siempre a un nivel de 0 lógico.

RD* (salida, 3 estados).
Lectura : Esta señal es utilizada por el microprocesador en el ciclo de lectura de memoria o de periféricos de E/S, dependiendo de la señal IO/M* ó S_2. RD* es activa baja durante T2, T3 y Tw en cualquier ciclo de lectura.

READY (entrada).
Listo : Es un reconocimiento desde un dispositivo direccionado (memoria o E/S) para indicar que está completa la transferencia de datos. La señal RDY desde memoria ó E/S está sincronizada con el generador del reloj 8284.

INTR (entrada).
Solicitud de interrupción : Esta señal es muestreada durante el último ciclo de reloj de cada instrucción para saber si el microprocesador entra en un estado de interrupción. Una subrutina es vectorizada por medio de un vector de interrupciones localizado en una tabla en la memoria . Puede ser internamente mascarable por software reinicializando el bit habilitador de interrupción.

TEST* (entrada).
TEST : Esta entrada es examinada por la instrucción "wait and test". Si la entrada TEST* está baja, continua la ejecución, por el otro lado el microprocesador espera en un estado ocioso. Esta entrada es sincronizada internamente durante cada ciclo de reloj en el flanco positivo del mismo.

NMI (entrada).
Interrupción no mascarable : Esta señal, junto con INTR, forman el conjunto de señales no mascarables. NMI es una señal por flanco; con un flanco negativo se inicia la interrupción y con un flanco positivo se indica el final de la interrupción.

RESET (entrada).
Reset : Esta señal altera la cola de interrupciones y ciertos registros (los indicadores, el apuntador de instrucción y los diferentes registros de segmento). Esta señal debe permanecer ALTA por lo menos cuatro ciclos de reloj.

CLK (entrada).
Reloj : Es el reloj del sistema y controlador del canal. Este es asimétrico con un ciclo de trabajo de un tercio (33 %) para proporcionar una óptima sincronía interna.

VCC
Vcc : Es de +5 volts de DC.

GND
Tierra : Es el nivel de referencia del sistema.

MN/MX* (entrada).
Mínimo/Máximo : Como ya se mencionó, indica en que modo de operación está el microprocesador.

SEÑALES DEL 8088 EN EL MODO MINIMO.

IO/M* (salida, 3 estados).
Línea de estado : Es utilizada para diferenciar entre un acceso a memoria y un acceso a un periférico de E/S. IO/M* puede ser valida en el T4 precediendo a un ciclo de canal.

WR* (salida, 3 estados).
Escritura : Es utilizada por el microprocesador para indicar una escritura en memoria o periféricos, dependiendo del estado de la señal IO/M*. WR* es activa para T2, T3 y Tw de cualquier ciclo de escritura.

INTA* (salida).
INTA : Es utilizada como un habilitador de lectura de ciclos de reconocimiento de interrupción.

ALE (salida).
Habilitación de captura de direcciones : Es proporcionada por el microprocesador para capturar las direcciones en un latch de direcciones.

DT/R* (salida, 3 estados).
Transmisión/recepción de datos : Es necesario en un sistema mínimo para usar el canal transceptor de datos. Esta es usada como un control de dirección en el flujo de datos a travéz del transceptor.

DEN* (salida, 3 estados).
Habilitación de datos : Es proporcionada como un habilitador de salida para el canal transceptor de datos. DEN* es activa BAJA durante cada acceso a memoria ó E/S y para ciclos INTA*.

HOLD, HOLDA (entrada, salida).
HOLD : El 8088 tiene una entrada de solicitud HOLD y una salida de reconocimiento HOLDA. Después de recibir una entrada ALTA en HOLD, el 8088 completa la ejecución de la instrucción en su ciclo de canal actual antes de entrar en un estado de HOLD y manda una salida ALTA en la terminal HOLDA.

SS0* (salida).

Línea de estado : Esta línea se combina con IO/M* y DT/R* para decodificar el presente ciclo de estado en el canal. Esta combinación es la siguiente :

IO/M*	DT/R*	SS0*	CICLO DE CANAL
0	0	0	ACCESO AL SEGMENTO DE CODIGO.
0	0	1	LECTURA DE MEMORIA.
0	1	0	ESCRITURA DE MEMORIA.
0	1	1	NO OPERA.
1	0	0	RECONOCIMIENTO DE INTERRUPCIONES.
1	0	1	LECTURA DE E/S.
1	1	0	ESCRITURA DE E/S.
1	1	1	"HALT".

SEÑALES DEL 8088 PARA EL MODO MAXIMO.

S2*, S1*, S0* (salidas, 3 estados).

Señales de estado : Estas señales permiten informar el tipo de operación que está realizando el microprocesador. Los códigos de estado proporcionados son los siguientes :

S2*	S1*	S0*	FUNCION
0	0	0	Reconocimiento de instrucción.
0	0	1	Lectura de E/S.
0	1	0	Escritura de E/S.
0	1	1	Paro.
1	0	0	Búsqueda de instrucción.
1	0	1	Lectura de memoria.
1	1	0	Escritura de memoria.
1	1	1	No hay ciclo de canal.

RQ*/GT0*, RQ*/GT1* (entrada/salida).

Request/Grant : Estas señales son usadas por otro controlador del canal para forzar al microprocesador a "soltar" el canal local y para que termine el ciclo de canal presente del microprocesador.

LOCK* (salida, 3 estados).

Lock : Con esta señal se le indica a otros controladores que NO utilicen el canal del sistema si LOCK* está en un nivel BAJO.

QS_1, QS_0 (salidas).

Cola de estado : Proporciona el estado de la cola de instrucciones . La cola de estado es válida durante el ciclo de reloj posterior en que se utiliza la operación de cola. El código binario de estas señales es el siguiente :

QS_1	QS_0	FUNCION.
0	0	No operación.
0	1	Primer byte del código de operación desde la cola.
1	0	La cola está vacía.
1	1	Subsiguiente byte desde la cola.

Después de hablar sobre las terminales del 8088, es conveniente mencionar los diferentes ciclos que ocurren cuando en determinadas terminales se les aplica una señal válida.

CICLO DE RESET.

Cuando se tiene un nivel alto en la terminal de RESET, el 8088 entra en un estado de RESET. Cuando esto ocurre se realizan los siguientes eventos :

1) El registro de banderas es borrado.
2) El contenido de registro de segmento de código es puesto en $FFFF_H$.
3) Los registros de segmentos de datos, stack y extra, así como el apuntador de instrucción son puestos a ceros.
4) La primera instrucción ejecutada después del RESET es aquella que se encuentra en la localidad de memoria $FFFF0_H$.

ESTADO DE HOLD.

El estado de HOLD del 8088 se usa para habilitar la lógica de acceso directo de memoria y para deshabilitar al microprocesador cuando más de un CPU accesa el mismo canal del sistema en una configuración multiCPU.

En el modo mínimo, el 8088 tiene una entrada de solicitud HOLD y una salida de reconocimiento HOLDA. El HOLD debe de estar estable cuando el reloj (CLK) está haciendo una transición de bajo a alto. Ya después de recibir una entrada alta en HOLD, el 8088 completa la ejecución de la instrucción en su ciclo de canal actual antes de entrar en un estado de HOLD y manda una salida alta en la terminal HOLDA; el 8088 muestrea la entrada HOLD en la transición bajo a alto de la señal de reloj.

El estado de HOLD permanece hasta que la entrada de HOLD baja otra véz.

ESTADO DE HALT.

Después de la ejecución de la instrucción de HALT el 8088 entra en un estado de HALT : los datos definidos son enviados al canal de datos/direcciones. Ningún ciclo de canal puede ser ejecutado mientras el 8088 esté en un estado de HALT, este estado se termina con un RESET o una interrupción.

ESTADOS DE ESPERA.

En cualquier ciclo de canal es posible insertar uno o más ciclos de espera entre T3 y T4. Para extender un ciclo de canal con periodos de espera, la lógica externa debe de mandar una señal baja a la terminal READY durante el T2 del ciclo de canal que va a ser extendido. La entrada READY del 8088 debe ser sincronizada con el flanco de bajada del reloj al final de T2; la entrada READY sincronizada, es proporcionada por el circuito 8284 generador de reloj. La lógica externa normalmente envia una señal READY asíncrona al dispositivo 8284 cuya salida es una señal READY sincronizada con el 8088. Los periodos de reloj de espera continuan siendo insertados en el ciclo de canal hasta que la señal READY pasa a ALTO otra véz. Todos los niveles de las señales de salida son mantenidos en la duración del estado de espera.

Cuando el 8088 tiene un estado de espera iniciado por programación, entonces la lógica externa debe terminarlo por medio de la señal externa TEST. Después de que la instrucción WAIT es ejecutada, el 8088 genera una secuencia de periodos de reloj ociosos. Esta secuencia permanece hasta que la lógica externa envia una señal baja a la entrada TEST.

3.- MODOS DE DIRECCIONAMIENTO

Cuando el 8088 ejecuta una instrucción, realiza una operación especifica sobre los datos. Estos datos pueden ser parte de la instrucción, estar en uno de los registros internos del 8088, estar almacenados en una localidad de memoria, o estar en un puerto de E/S. Para accesar estos diferentes tipos de operandos, el 8088 está provisto con varios modos de direccionamiento. Estos modos reciben varios nombres, así tenemos:

- Direccionamiento de registros.
- Direccionamiento inmediato.
- Direccionamiento directo.
- Direccionamiento de registro indirecto.
- Direccionamiento de base.
- Direccionamiento indexado.
- Direccionamiento de base indexado.
- Direccionamiento de filas.
- Direccionamiento de puertos.

Para obtener una dirección real del operando se deben de sumar (la suma se realiza internamente en el microprocesador) cuatro cantidades :

1) Una dirección de segmento (el contenido del registro de segmento es multiplicado por 10_H antes de ser utilizado para generar la dirección real).
2) Una dirección base.
3) Un índice.
4) Un desplazamiento.

A continuación se explican brevemente cada uno de estos modos de direccionamiento.

DIRECCIONAMIENTO DE REGISTRO.

En este modo, el operando está contenido en uno de los registros de propósito general especificados dentro del código de operación.

DIRECCIONAMIENTO INMEDIATO.

En este modo de direccionamiento uno de los operandos es especificado en uno ó dos bytes siguientes al código de operación.

DIRECCIONAMIENTO DIRECTO.

En este modo, la dirección del operando en memoria es epecificada como la suma del contenido de registro de segmento de datos y un desplazamiento de 16 bits contenido en los dos bytes siguientes al código de operación.

DIRECCIONAMIENTO DE RGISTRO INDIRECTO.

La dirección del operando en memoria es especificada por el contenido de un registro base o un registro índice.

DIRECCIONAMIENTO DE BASE.

En este modo, la dirección del operando es especificada como la suma del contenido del registro BX o BP y un desplazamiento opcional. Si el registro especificado es el BP entonces el operando será obtenido del segmento usado como stack.

DIRECCIONAMIENTO INDEXADO.

En este modo la dirección del operando en memoria es especificada por el contenido de uno de los registros de índice (SI o DI). Este modo tiene la opción de especificar un desplazamienot de 16 bits en los dos bytes siguientes al código de operación, al cual es sumado al registro de índice especificado.

DIRECCIONAMIENTO DE BASE INDEXADO.

En este modo la dirección es especificada como la suma del contenido de un registro base (BX o BP), el contenido de un registro de índice (SI o DI) y un desplazamiento de 16 bits.

DIRECCIONAMIENTO DE FILAS.

Entendemos por filas una serie de bytes o palabras de datos que residen en localidades de memoria consecutivas. De esta forma podemos mover, combinar, almacenar o buscar bloques de datos.

DIRECCIONAMIENTO DE PUERTOS.

Como su nombre lo indica, con este tipo de direccionamiento podemos habilitar un puerto cualquiera de una forma directa o indirecta.

4.- INTERRUPCIONES.

El microprocesador 8088 permite que las interrupciones sean originadas en una de las tres formas posibles :

a) Por medio de programación o dentro de la lógica de un programa.
b) Por medio de una lógica externa como una interrupción NO MASCARABLE.
c) Por medio de una lógica externa como una interrupción MASCARABLE.

En el caso de que dos o más de los tres tipos de interrupciones ocurran simultáneamente; las interrupciones generadas por programa tienen la más alta prioridad y las interrupciones mascarables tienen la más baja prioridad.

El concepto de interrupción por programa se usa para manejar una variedad de problemas que surgen a raíz de la secuencia del programa normal. La interrupción por programa se refiera a la trasferencia de control de un programa que está trabajando normalmente a otro programa de servicio como resultado de una condición dada.

Las interrupciones por programa pueden ocurrir :

1) Siguiendo un intento de dividir por cero. Una solicitud especial de interrupción ocurrirá en este caso.
2) Después de la ejecución de la instrucción INT.
3) Después de la ejecución de la instrucción INTO, si la bandera de sobreflujo está en un estado de 1 lógico.

Una solicitud de interrupción no mascarable es iniciada cuando la lógica externa transmite una transición de BAJO a ALTO en la terminal NMI del microprocesador 8088.

Una solicitud de interrupción mascarable será generada cuando la lógica externa transmite un nivel ALTO en la terminal INTR del 8088.

Relacionada con el proceso de interrupciones está una tabla de vectores que puede tener hasta 1024 bytes de longitud, ocupando las direcciones de memoria desde la 00000_H hasta la $003FE_H$ Esta tabla de vectores consiste de hasta 256 vectores de cuatro bytes cada uno. Cada vector contiene dos direcciones de 16 bits cada una de ellas, que serán cargadas en el registro de segmento de código y en el apuntador de instrucción.

Algunos de estos vectores sirven para interrupciones específicas. Otros vectores están reservados por INTEL y deben de ser evitados para poder tener una compatibilidad con los productos de INTEL. El resto de la tabla, 224 vectores, esta disponible para interrupciones externas mascarables.

Cuando una **INTERRUPCION POR PROGRAMA** es reconocida, se realizan los siguientes pasos :

1) El registro de banderas es guardado en el stack y el Stack Pointer es decrementado en dos.
2) Las banderas de interrupción y de prueba (I y T respectivamente) son puestas en 0 lógico; esto deshabilita las interrupciones mascarables y el modo de paso simple.
3) El contenido del registro de segmento de código es guardado en el stack y el Stack Pointer es nuevamente decrementado en dos.
4) El nuevo contenido del registro de segmento de código es tomado de la localidad apropiada del vector de interrupciones. Con excepción de la instrucción INT, las interrupciones generadas por programación tienen un vector especial como se puede ver en la tabla de vectores de interrupciones. La instrucción INT permite que cualquiera de los 256 vectores sea seleccionado; la opción automáticamente selecciona el vector 3.
5) El contenido del apuntador de instrucción es guardado en el stack; el Stack Pointer es decrementado en dos.
6) El nuevo contenido del apuntador de instrucción es tomado del vector de interrupción apropiado.

Cuando una **INTERRUPCION NO MASCARABLE** es reconocida, se realizan los siguientes eventos :

1) El contenido del registro de banderas es guardado en el stack y es Stack Pointer es decrementado en dos.
2) Las banderas de interrupción y prueba son puestas en 0 lógico, esto deshabilita las interrupciones no mascarables y el modo de paso simple.
3) El contenido del segmento de código y del apuntador de instrucción son guardados en el stack y los nuevos valores son tomados del vector de interrupción número dos.

Cuando una **INTERRUPCION MASCARABLE** es reconocida, ocurren los siguientes pasos :

1) Dos ciclos de canal de reconocimiento de interrupción son ejecutados por la BIU. Un ciclo de canal de reconocimiento de interrupción es similar al ciclo de canal de lectura de memoria, con la excepción de que el pulso de reconoicimiento de interrupción reemplaza al pulso de lectura de memoria.
2) El dispositivo externo debe de enviar un byte de datos sobre las líneas AD_0-AD_7 en respuesta al segundo ciclo de canal de reconocimientro de interrupción. Este byte de datos es interpretado como un apuntador hacia el vector de interrupción. Multiplicando este valor de 8 bits por 4 se crea la dirección de inicio del vector de interrupción.
3) El registro de banderas es guardado en el stack.
4) Las banderas de interrupción y prueba son puestas en 0 lógico.
5) EL contenido del registro de segmento de código es tomado del vector de interrupciones identificado en el paso 2.
6) El contenido en el apuntador de instrucción es guardado

en el stack.

7) El nuevo contenido en el apuntador de instrucción es tomado del vector de interrupción especificado en el paso 2.

8) La primera instrucción de la rutina de interrupción es capturada usando el nuevo apuntador de instrucción y el nuevo segmento de código.

En la siguiente página se presenta la tabla de vectores de interrupción.

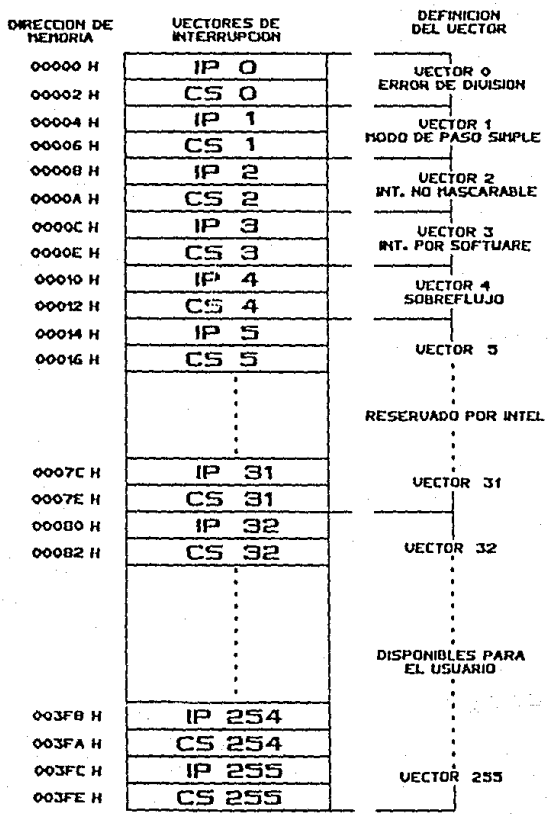


TABLA DE VECTORES DE INTERRUPCION

5.- SET DE INSTRUCCIONES.

En general, las instrucciones del microprocesador 8086/88 se pueden clasificar de la siguiente manera :

1.- Instrucciones de transferencia de datos.
2.- Instrucciones aritméticas.
3.- Instrucciones de operaciones lógicas.
4.- Instrucciones de desplazamientos y corrimientos.
5.- Instrucciones para el control del registro de banderas.
6.- Instrucciones de comparación.
7.- Instrucciones de salto.
8.- Instrucciones para el manejo de subrutinas.
9.- Instrucciones para guardar y recuperar.
10.- Instrucciones de malla.
11.- Instrucciones para el manejo de strings.
12.- Instrucciones de interrupción.

INSTRUCCIONES DE TRANSFERENCIA DE DATOS.

Son las instrucciones encargadas de mover datos de un sitio a otro, ya sea de la memoria, registro interno ó dispositivo de entrada/salida. Este grupo incluye instrucciones para mover un byte o palabra, intercambiar un byte o palabra, trasladar un byte, cargar una dirección efectiva y cargar un segmento extra.

INSTRUCCIONES ARITMETICAS.

El microprocesador 8088 cuenta con un completo conjunto de instrucciones aritméticas y de complemento. Estas instrucciones sirven para sumar, restar, multiplicar ó dividir. Estas operaciones pueden realizarse sobre números expresados en una variedad de formatos numéricos. Estos formatos pueden ser: bytes o palabras signadas o no signadas, bytes de números decimales empaquetados o números ASCII.

INSTRUCCIONES DE OPERACION LOGICA.

El 8088 tiene instrucciones para realizar operaciones lógicas AND, OR, XOR (OR exclusiva) y NOT.

INSTRUCCIONES DE DESPLAZAMIENTO Y CORRIMIENTO.

Las instrucciones de corrimiento del 8088 pueden realizar dos tipos básicos de corrimientos: corrimientos lógicos y corrimientos aritméticos y a la vez, cada una de estas operaciones pueden hacerse hacia la derecha o hacia la izquierda.

INSTRUCCIONES PARA EL CONTROL DEL REGISTRO DE BANDERAS.

Como se vió anteriormente, el 8088 cuenta con un registro de banderas que nos sirve para monitorear el estado de las instrucciones que se están ejecutando como para el control de opciones disponibles en su operación . El 8088 cuenta con un grupo de instrucciones las cuales, cuando son ejecutadas, afectan

directamente al registro de banderas.

INSTRUCCIONES DE COMPARACION.

Existe una instrucción incluida en el conjunto de instrucciones del 8088, la cual puede ser usada para comparar dos números de 8 bits o dos de 16 bits. El resultado de la comparación es reflejado por cambios en seis de las banderas del 8088. El proceso de comparación es básicamente una operación de substracción en donde el operando fuente se substrae del operando destino; el resultado de la substracción NO es guardado, sin embargo las banderas son afectadas de acuerdo al resultado. Los operandos pueden estar almacenados tanto en localidades de memoria así como en registros internos del miroprocesador, o bien ser especificados como parte de la instrucción.

INSTRUCCIONES DE SALTO.

El propósito de una instrucción de salto es el de alterar la secuencia de ejecución de las instrucciones de un programa. En el microprocesador 8088, el registro desegmento de código y el apuntador de instrucción indican la próxima instrucción que será ejecutada. Entonces una instrucción de salto implica la alteración del contenido de estos registros. De esta manera, la ejecución del programa continua en una dirección diferente a la de la próxima instrucción secuiencial, es decir, ocurre un salto a otro parte del programa.

El 8088 cuenta con dos tipos diferentes de instrucciones de salto; el salto incondicional y el salto condicional. En el salto incondicional no se requiere que exista un estado particular para que el salto ocurra. Esto es, cuando la instrucción es ejecutada, el salto el salto siempre tiene lugar para cambiar la secuencia de ejecución. Por otra parte, en las instrucciones de salto condicional, las condiciones de estado que existen en el momento en que la instrucción de salto es ejecutada decide si el salto ocurre o no. Es decir, si la condición o condiciones se cumplen entonces el salto se realiza; de otra forma la ejecución continua con la próxima instrucción secuencial del programa.

Existen dos tipos de saltos incondicionales. La primera, llamado salto intrasegmento, esta limitada a direcciones dentro del segmento de código actual. Este tipo de salto es realizado tan solo modificando el valor del apuntador de instrucción. La otra clase de salto, el salto intersegmento, permite saltar de un segmento de código a otro. La realización de este tipo de salto requiere la modificación del contenido del registro de segmento de código así como del apuntador de instrucción

INSTRUCCIONES PARA EL MANEJO DE SUBRUTINAS.

El conjunto de instrucciones del microprocesador 8088 cuenta con dos instrucciones básicas para el manejo de subrutinas. Estas son: la llamada (CALL) y el retorno (RET). Juntas proporcinan el mecanismo para llamar una subrutina y para regrasar el control al

programa principal cuando la subrutina se ha realizado. La instrucción de llamada permite dos tipos de operación: la llamada intrasegmento y la llamada intersegmento.

INSTRUCCIONES PARA GUARDAR Y RECUPERAR.

Cuando se trabaja con subrutinas, frecuentemente encontramos la necesidad de guardar el contenido de ciertos registros o parámetros del programa principal. Estas valores son salvados al guardarlos en el stack. De esta manera su contenido se mantiene intacto durante la ejecución de la subrutina. Antes del regresar al programa principal los valores guardados deber ser recuperados. Esto se logra al tomar los valores del satck y regresarlos a sus registros originales. La instrucción que se usa para guardar parámetros en el stack es la instrucción PUSH y la que se usa para recuperarlos es la instrucción POP.

INSTRUCCIONES DE MALLA.

El microprocesador tiene tres instrucciones especificamente diseñadas para implementar operaciones de malla. Estas instrucciones pueden ser usadas en lugar de ciertas instrucciones de salto condicionado y dan al programador una forma fácil de escribir secuencias de malla.

INSTRUCCIONES PARA EL MANEJO DE STRINGS.

El 8086 está equipado con instrucciones especiales para el manejo de operaciones con "string". Por "string" entendemos una serie de bytes o palabras de datos que residen en localidades consecutivas de memoria. Estas instrucciones permiten al programadorpoder implementar operaciones tales como mover datos de un bloque de memoria hacia otro bloque en cualquier parte de la memoria. Otra aplicación es poder comparar dos grupos de datos para poder determinar si son iguales o no, o bien el buscar un grupo de datos almacenados en memoria buscandolos por su valor especifico.

INSTRUCCIONES DE INTERRUPCION.

Algunas instrucciones del 8088 son utilizadas para el procesamiento de interrupciones. Por ejemplo STI y CLI permiten manipular la bandera de interrupción mediante programación y de este modo habilitar o no las interrupciones.

Las últimas dos instrucciones asociadas con la interface de interrupciones son HLT y WAIT, estas instrucciones permiten sincronizar al 8088 con un evento en la electrónica externa. Cuando la instrucción HLT es ejecutada, el 8088 suspende su operación y entra en un estado ocioso y permanece así hasta que una interrupción externa o un RESET ocurre. Cuando esto sucede el 8088 reasume su operación con la correspondiente subruitna de servicio. Si la instrucción WAIT ocurre, el 8088 verifica el nivel lógico de la entrada TEST; si estáen un 1 lógico el 8088va a un estado ocioso, mientras está en este estado el 8088 continua

verificando el nivel lógico de la entrada TEST buscando la transición a un cero lógico y cuando esto acurre la ejecución continúa con la siguiente instrucción secuencial en el programa.

DISPOSITIVOS PERIFERICOS.

Introducción.

Un dispositivo de entrada/salida es un componente LSI que provee el enlace de interconexión entre un microprocesador y el mundo externo. Cuando está en modo de salida de datos, el dispositivo recibe información binaria del canal de datos a la tasa de transmsión y modo de transferencia del microprocesador y la tramsmite de esa manera.

La mayoria de los dispositivos de entrada/salida pueden ser programados para acomodar una gran variedad de combinaciones de modo de operación. El microprocesador, por medio de instrucciones de programa, transfiere un byte al registro de control dentro del dispositivo. Esta información de control coloca al dispositivo en uno de los modos posibles disponibles. Cambiando el byte de control es posible cambiar las caracteristicas de operación del dispositivo. Por ello, a menudo se les llama programables.

Las instrucciones que transfieren la información de control a un dispositivo programable son incluidas en el programa y pueden iniciar la interconexión para un modo particular de operación.

Los fabricantes de microprocesadores complementan sus productos con un conjunto de dispositivos periféricos integrados adecuados para la comunicación entre el microprocesador y el mundo exterior.

Un microprocesador tiene un límite para el número de terminales que puede tener un circuito integrado. No hay suficientes terminales en el CI para que el microprocesador puede suministrar canales separados para comunicarse con la memoria y los dispositivos de entrada/salida. Invariablemente todos los microprocesadores utilizan un sistema de canal común para seleccionar palabras de memoria y dispositivos periféricos. El canal del microprocesador no distingue entre un dato de una memoria y un dato de un dispositivo periférico. Es responsabilidad del usuario, por medio del instrucciones en el programa, especificar la dirección apropiada que seleccione uno u otro. Existen dos maneras de hacer esto: uno es el llamado I/O por mapa de memoria y el otro es el I/O aislado.

En el método por mapa de memoria, el microprocesador trata al registro del dispositivo periférico como una parte del sistema de memoria; por ello, las direcciones utilizadas para estos registros no pueden ser usadas para la memoria, reduciendo el espacio de la misma. La organización de I/O por mapa de memoria es conveniente para sistemas que no necesiten espacio disponible de memoria de las líneas del canal de direcciones.

Con la organización de I/O aislado, el microprocesador

especifica en sí mismo cuando la dirección en el canal de direcciones es para una memoria o un dispositivo periférico. Esto se hace por medio de una o dos líneas de control adicionales que presentan las terminales del microprocesador. En la organización de I/O aislado, el microprocesador debe entregar instrucciones diferentes de entrada y salida y cada una de ellas debe de asociarse con una dirección. Cuando el microprocesador busca y decodifica el código de operación de ina instrucción de entraday salida, éste lee la dirección asociada con la instrucción y la coloca en el canal de direcciones.

Debido a que nuestro sistema cuenta con dispositivos periféricos del tipo serie y paralelo solo nos dedicaremos a este tipo de dispositivos. Un dispositivo de entrada/salida (I/O) puede transferir información binaria en serie o en paralelo. En la transmisión en paralelo, cada bit de informació usa una línea separada de manera que los N bits de un ítem pueden ser transmitidos simultáneamente.

En la transmisión en serie, los bits de una palabra son tramsmitidos en secuencia, bit por bit a través de una sola línea. La transmisión en paralelo es mucho más rápida pero requiere de muchas líneas. Esta se usa donde las distancias son cortas y la velocidad es importante. La transmisión en serie es lenta pero menos costosa ya que solamente requiere de una línea.

A continuación se presentan los estudios de los dispositivos periféricos utilizados en nuestro sistema, como son el 8155 que es un dispositivo con tres puerts paralelos (dos de 8 bits y uno de 6 bits), el 8251 que se trata de un puerto tipo serie síncrono/asíncrono, el 8279 que es un controlador de interrupciones programable y el 8259 que es un controlador de despliegue y teclado.

EL 8155H. PUERTO PARALELO.

El 8155H es una memoria RAM estática (HMOS) de 256 x 8 bits con puertos de entrada/salida y un TIMER. entre otras características de este circuito es que contiene un latch interno de direcciones, dos puertos de entrada/salida programables de 8 bits y un puerto programable de 6 bits. Además posee un contador/timer binario programable de 14 bits y los canales de direcciones y de datos están multiplexados.

El 8155H tiene un tiempo máximo de acceso de 330 nseg con lo que es compatible con el 8085 y el 8088CPU con reloj de 5 MHz.

Con respecto a los puertos de entrada/salida, se trata de puertos de propósito general. Uno de los tres puertos (puerto C)* puede ser programado para señales de estado con lo que se permitiría a los otros dos puertos operar en el modo de protocolo (HANDSHAKE).

El contador/timer programable puede proporcionar una señal cuadrada o una señal de pulsos para un sistema CPU.

PROGRAMACION DEL REGISTRO COMANDO

El registro comando consiste de 8 latches, uno por cada bit. Cuatro de los ocho bits (0-3) definen el modo de los puertos, dos de los bits (4-5) habilitan o deshabilitan la interrupción de los puertos cuando el puerto C actua como puerto de control y los últimos dos bits (6-7) son para el timer.

El contenido del registro comando puede ser alterado en cualquier momento usando la dirección de entrada/salida (I/O) XXXXX000 durante la operación de escritura, con el CHIP ENABLE (CE) activado e IO/M* en 1 lógico.

La asignatura de los bits del registro comando es la siguiente :

```
B0: PA= Define PA 0-7  \   ENTRADA=1
B1: PB= Define PB 0-7  /   SALIDA =0
          /  00= ALTERNATIVA 1
B2: PC1  /   11=      "       2
B3: PC2  \   01=      "       3
          \  10=      "       4
B4: IEA=HABILITA LA INTERRUPCION DEL PA  /  1=HABILITA
B5: IEB=   "       "       "          "  PB  \  0=DESHABILITA
                                  /  00= NOP
B6: TM1   COMANDO DE TIMER       /   01= ALTO
B7: TM2                          \   10= ALTO DESPUES TC
                                  \  11= ARRANCA

Donde :    PA  >>---> Puerto A.
           PB  >>---> Puerto B.
           PC  >>---> Puerto C.
```

NOTA : Las alternativas del comando del timer se explicarán más adelante.

LECTURA DEL REGISTRO DE ESTADO.

El registro de estado está constituido por siete latches, uno por cada bit; seis (0-5) son para el estado de los puertos y uno (6) para el estado del timer.

El estado del timer y la sección I/O puede ser revisada mediante la lectura del registro de estado. La asignatura del

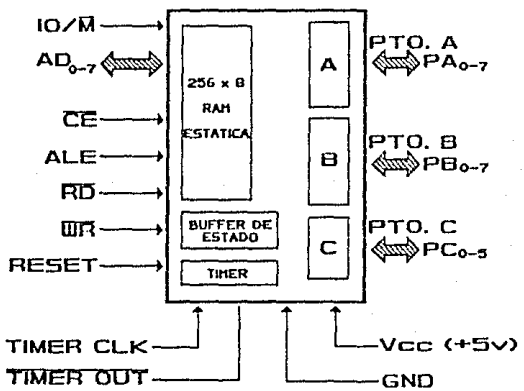


DIAGRAMA DE BLOQUES DEL 8155H

registro de estado es la siguiente :

AD_0: INTR A = SOLICITUD DE INTERRUPCION DEL PUERTO A.
AD_1: A BF = BUFFER LLENO/VACIO DEL PA (ENTRADA/SALIDA).
AD_2: INTE A = HABILITADOR DE INTERRUPCION DEL PA.
AD_3: INTR B = SOLICITUD DE INTERRUPCION DEL PUERTO B.
AD_4: B BF = BUFFER LLENO/VACIO DEL PB.
AD_5: INTE B = HABILITADOR DE INTERRUPCION DEL PB.
AD_6: TIMER = INTERRUPCION DEL TIMER.
AD_7: NO IMPORTA.

SECCION DE ENTRADA / SALIDA .

La sección de entrada/salidaconsiste de cinco registros :

REGISTRO COMANDO/ESTADO (C/S).
Ambos registros están asignados a la dirección XXXXX000. La dirección del registro C/S sirve para un doble propósito.

Cuando el registro C/S es seleccionado durante la operación de escritura (WRITE). Cuando C/S es elegido durante la operación

de lectura (READ), la información de estado de los puertos I/O y del timer pasa a las terminales de dirección/dato.

REGISTRO PA (PUERTO A).
Este registro puede ser programado como puerto de entrada o de salida dependiendo del estado del contenido del registro C/S. Dependiendo del comando, el puerto puede operar en modo básico o modo strobed. La dirección de este registro es XXXXX001. Las terminales asignadas en relación a este registro son PA_0-PA_7.

REGISTRO PB (PUERTO B).
Este registro es similar al registro PA pero en relación al puerto B. Las terminales asociadas a este registro son PB_0-PB_7. La dirección de este registro es XXXXX010.

REGISTRO PC (PUERTO C).
Este registro tiene la dirección XXXXX011 y solo contiene 6 bits. Los 6 bits pueden ser programados para funcionar como puerto de entrada, puerto de salida o como señales de control de los puertos A y B. Cuando PC_0-PC_5 es usado como puerto de control, tres de los bits son asignados al puerto A y tres al puerto B. El primer bit es una interrupción. El segundo indica si el buffer se encuentra lleno o vacio y el tercero es una terminal de entrada para aceptar la habilitación para el modo de entrada strobed.

Cuando el 8155H es inicializado (se activa el RESET), las salidas son limpiadas y los tres puertos se colocan en el modo de entrada.

SECCION DEL TIMER.

El timer es un contador hacia abajo de 14 bits que cuenta los pulsos del TIMER IN y proporciona una señal cuadrada o un pulso cuando la terminal de conteo (TC) es tocada.

El timer tiene la dirección de I/O XXXXX100 para el byte menos significativo del registro y la dirección de I/O XXXXX101 para el byte más significativo del registro.

Para programar el timer, el registro contador de tiempo es cargado primeramente, un byte a la véz, para seleccionar las direcciones del timer. Los bits del 0 al 13 del registro de conteo especifican el tiempo del próximo conteo y los bits 14 y 15 de este registro especifican el modo de salida del timer.

La asignación del registro de conteo es el siguiente:

B_0-B_7 : T_0 A T_7 =PARTE MENOS SIGNIFICATIVA DEL REGISTRO.

B_8-B_{13} : T_8 A T_{13} =PARTE MAS SIGNIFICATIVA DEL REGISTRO.

B_{14}-B_{15}: M_1 Y M_2 =MODO DEL TIMER.

0 0 :SEÑAL CUADRADA SIMPLE.

B_{14}-B_{15}: M_1 Y M_2 =MODO DEL TIMER.

0	1	:SEÑAL CUADRADA CONTINUA.
1	0	:PULSO SIMPLE A LA TERMINAL DE CONTEO.
1	1	:SEÑAL DE PULSOS CONTINUOS.

Los bits 6 y 7 (TM1 y TM2) del registro comando son utlilizados para poner en marcha o parar el contador. Hay cuatro maneras de combinación:

TM1	TM2	FUNCION
0	0	NOP:No afecta la operación del contador.
0	1	PARAR:No afecta el contador si el timer no está corriendo y detiene el conteo si el timer está corriendo.
1	0	PARA DESPUES DEL TC:Para el conteo después de que se presenta una señal en TC.
1	1	ARRANCA:Carga el modo y el registro de conteo y pone en marcha el contador después de la carga.Si el timer está corriendo, arranca en el nuevo modo inmediatamente después de que se presenta una señal en TC.

Note que mientras el contador está contando, se puede cargar un nuevo conteo y modo en los registros de conteo de tiempo. Antes del nuevo modo y conteo a usar por el contador, se debe de mandar un comando de arranque al contador.

El contador del 8155H no es inicializado en ningún modo en particular o conteo cuando ocurre un RESET, pero el RESET detiene el conteo. Por lo tanto, el conteo no puede seguir después de un RESET hasta que el comando de arranque es enviado por el registro C/S.

EL 8251. RECEPTOR/TRANSMISOR SINCRONO/ASINCRONO

DESCRIPCION POR BLOQUES.

El 8251 está formado de cinco secciones las cuales se comunican mediante un bus interno de datos, dichas secciones son:

- El receptor,
- El transmisor,
- Modem de control,
- Control de lectura/escritura, y
- el buffer de entrada/salida.

El buffer de entrada/salida está dividido en tres partes: el buffer de estado, el buffer de transmisión de datos/comando y el buffer receptor de datos.

RECEPTOR.

Acepta datos en serie en el pin RxD y los convierte en paralelo de acuerdo al formato adecuado. Cuando el 8251 está en el modo asíncrono y listo para aceptar un caracter va a un nivel bajo en la línea RxD. Cuando "ve" un nivel bajo, asume que éste es un bit de START y habilita un contador interno. A la cuenta de medio o de un bit, la línea RxD se muestrea de nuevo, si la línea es aún baja un bit de START tiene probabilidad de ser recibido y el 8251 porocede a ensamblar el caractes, pero si ésta es alta entoces está ocurriendo un pulso de ruido en la liinea o el receptor es habilitado a la mitad de la transmisión de un caracter.

Cuando opera con menos de 8 bits , los caracteres son debidamente justificados, y RxRDY es mantenida para indicar que un caracter es aprovechado.

En el modo sincrono el receptor admite el número especificado de bits de datos y los transfiere al registro buffer receptor, activando RxRDY. Por lo tanto el receptor al azar bits de datos en catacteres que deberán significar la sincronización del receptor y el transmisor manteniendo los caracteres limitadores en la corriente de datos en serie.

Esta sincronización es ejecutada en el modo HUNT. El 8251 en el modo HUNT cambia en datos un bit a la vez en la línea RxD. Después de cada bit es recibido. el registro recibido es comparado con el registro que contiene el caracter SYN. Si los dos registros no son iguales, el 8251 cambia un bit y repite la comparación. Cuando los registros comparados son iguales, el 8251 termina el modo HUNT y activa la línea de SYNDET para indicar que la sincronización ha sido ejecutada. Si el USART es programado para operar con dos caracteres SYN, el proceso es como ya se describió, excepto de que los dos caracteres contiguos provenientes de la línea deben compararse con los caracteres SYN almacenados antes de que la sincronización sea declarada. Si el USART es programado para aceptar sincronización externa, la

terminal SYNDET es usada como una entrada para sincronizar el receptor.

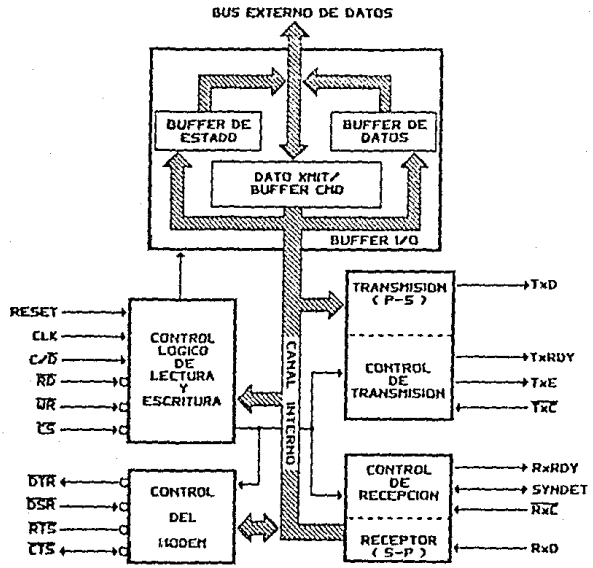


DIAGRAMA DE BLOQUES DEL 8251

TRANSMISOR.

El transmisor acepta daots en paralelo provenientes del procesador, añadiendo la información ordenadamente, seriandola y transmitiéndola por la terminal TxD.

En el modo asincrono, el transmisor siempre añade un bit de START, y dependiendo de cómo esté programada la unidad, ésta añade una paridad opcional par ó impar y 1, 1 1/2 ó 2 bits de STOP.

En el modo sincrono, no son generados bits extras por el transmisor a menos de que la computadora omita un caracter al mandarlo al USART. Si el USART está listo para transmitir un caracter y el nuevo caracter no está siendo suministrado por la computadora, el USART transmitirá un caracter SYN. Si el USART está operando en modo SYN doble, ambos caracteres serán transmitidos antes de que el mensaje pueda ser resumido. El USART no genera caracteres SYN hasta que el software tenga suministrado el último caracter. Los caracteres SYN transmitidos por el USART son especificados por el software durante le procedimiento de

inicialización. En los modos síncronos y asíncronos, la transmisión es inhibida hasta que TxE es habilitada y CTS* es confirmada.

Una característica adicional del transmisor es la habilidad de transmitir un BREAK, el cual es un período de SPACE continuo en la línea de comunicación y es usado en la comunicación FULL DUPLEX para interrumpir la terminal de transmisión. El 8251 transmitirá una condición BREAK de 3 bits de largo cuando el registro de comando es activado.

MODEM DE CONTROL.

El modem de control se encarga de la generación de RTS* y la recepción de CTS*. En adición, una salida de propósito general y una entrada de propósito general son suministradas. La salida es llamada DTR* y la entrada es DSR*.

DTR (Data Terminal Ready) es normalmente asignado al modem, indicando que la terminal está lista para comunicarse y DSR (Data Set Ready) es una señal proveniente del modem indicando que éste está listo para la comunicación.

CONTROL DE LECTURA/ESCRITURA.

La lógica de leer/escribir decodifica señales de control dentro del canal correspondiente, las cuales activan o desactivan el canal interno del USART y controlan el canal externo de E/S (DB0-DB7). La tabla de verdad para estas operaciones es la siguiente :

CE	C/D*	READ*	WRITE*	FUNCION
0	0	0	1	EL CPU LEE DATOS DEL USART.
0	1	0	1	EL CPU LEE ESTADO DEL USART.
0	0	1	0	EL CPU ESCRIBE DATOS AL USART.
0	1	1	0	EL CPU ESCRIBE COMANDOS AL USART.
1	X	X	X	CANAL DE USART EN TRES ESTADOS.

BUFFER DE ENTRADA/SALIDA.

El buffer de E/S contiene el buffer de STATUS, el buffer RECEIVE DATA y el XMIT DATA/CMD. Hay dos registros los cuales almacenan datos para transferirlos al CPU (STATUS y RECEIVE DATA) y hay un solo registro que almacena datos para ser transferidos al USART.

SEÑALES DE INTERFACE.

Las señales de interface del 8251 (USART) pueden ser divididas en dos grupos : un grupo referido al CPU y otr grupo referido al dispositivo. Las señales referidas al CPU son designadas para optimizar la unión del 8251 al sistema MCS-80. Las señales referidas a un dispositivo son producidas para

interface de un modem u otro dispositivo.

FORMATOS DE COMUNICACION

La comunicación en serie, ya sea en una unión o encadenamiento de datos o con un periférico local ocurre en uno de los dos formatos básicos: asíncrono o síncrono. Son similares en cuanto ambos requieren ajustar u ordenar información para que sea añadida a los datos y habilitar la detección del caracter hasta el final de la reccepción. La mayor diferencia entre los dos formatos es que el asíncrono requiere ajustar información para ser "sumada" a cada caracter, mientras que el síncrono "suma" información ordenandola en bloques de datos o mensajes; por lo tanto el formato sincrono es más eficiente pero requiere una decodificación más compleja, siendo típico encontrarlo en encadenamientos de datos de alta velocidad mientras que el asíncrono se usa para líneas de baja velocidad.

El formato asíncrono empieza por añadir y transmitir un bit de inicio (START) y uno o más bits de alto (STOP) al final del mensaje cuando ha sido transmitido. El bit de START es un cero lógico u espacio ; el bit de STOP es un uno lógico o marca. El bit de START llama al receptor convocando un caracter y permite que éste y el transmisor sean sincronizados.

Uno o más bits de STOP son añadidos al final del caracter para asegurar que el próximo bit de START del siguiente caracter causará la transmisión en la línea de comunicación un cambio que avise al receptor de un próximo caracter.

El formato síncrono, en lugar de sumar bits para cada caracter, graba interiormente grupos de caracteres y añade caracteres ordenados para la grabación. Los caracteres ordenados son conocidos como caracteres SYN y son usados por el receptor para determinar donde están los caracteres limitadores en un arreglo de bits.

El formato síncrono requiere de un bits de START precedente para cada caracter y un solo bit de STOP de éste, siendo utilizados 10*n bits para transmitir "n" caracteres, y en el modo síncrono son utilizados 8N+16 bits.

En adición a las diferencias entre los formatos podemos mencionar que el formato síncrono es mucho más eficiente que el formato asíncrono en la transmisión de mensajes largos, también existen diferencias entre el tipo de modems que puedan ser usados. Los modems asíncronos típicamente emplean técnicas FSK (Frecuency Shift Keying) la cual simplemente genera un tono de audio por una marca y otro por un espacio. El modem receptor detecta estos tonos en la línea telefónica, convirtiéndolos en señales lógicas.

Los modems síncronos alimentan regulando información para la terminal y requiere datos para ser presentados en sincronia con la información regulada.

El modo sincrono del 8251 opera con caracteres de 5, 6, 7 u 8 bits. La paridad par o impar puede ser opcionalmente añadida y checada. La sincronía puede ser ejecutada externamente o internamente mediante la detección de los caracteres SYN. Dicha detección está basada en uno o dos caracteres, los cuales pueden ser o no iguales. El único o doble caracter SYN son insertados automáticamente dentro de la corriente de datos. La generación automática de caracteres SYN se requiere para prevenir la pérdida de sincronización.

En el modo asíncrono el 8251 opera con los mismos datos y estructuras de paridad como hace en el modo síncrono, y aparte de generar un bit de START, añade 1, 1 1/2 ó 2 bits de STOP que acomodados adecuadamente son checados en el receptor y el estado de las banderas se activa si hay un error.

MODOS DE SELECCION.

Como el USART 8251 es capaz de operar en dos modos (síncrono ó asíncrono), éstos son seleccionados mediante una serie de salidas de control a el USART. Estas salidas del control de modo deben ocurrir entre el tiempo en que el USART es inicializado y el tiempo en que es utilizado para transferir datos. Por lo tanto, el USART necesita esta información para estructurar su lógica interna esencial para completar la inicialización antes de que un intento sea hecho para transferir datos.

La primera operación que debe de ocurrir después de una inicialización es es que la carga del registro de control de modo mediante una salida de control (C/D*=1, RD*=1, WR*=0 y CS*=0). El formato de la instrucción de control de modo se muestra más adelante.

En dicho diagrama, se observa que la instrucción puede ser considerada como cuatro campos de 2 bits. El primer campo de dos bits (D_1 y D_0) determina si el USART estáoperando en el modo síncrono ó asíncrono. En el modo asíncrono este campo también controla el factor de escala del reloj.

El segundo campo (D_3 y D_2) determina el número de bits de datos en el caracter y el tercer campo (D_4 y D_5) controlan la generación de paridad. El bit de paridad es añadido a los bits de datos y no es considerado como como parte de ellos cuando activa alto la longitud del caracter.

El siguiente es el formato de introducción de modo de operación :

D_1 D_0 : FACTOR DE TASA DE VELOCIDAD.

0 0 = SYN MODE
0 1 = ASYN x1
1 0 = ASYN x16
1 1 = ASYN x64

D_3 D_2 : LONGITUD DEL CARACTER.

0	0	= 5 BITS
0	1	= 6 BITS
1	0	= 7 BITS
1	1	= 8 BITS

D_5 D_4 : CONTROL DE PARIDAD.

X	0	= NO PARAIDAD
0	1	= PARIDAD NON
1	1	= PARIDAD PAR

D_7 D_6 : FRAMING CONTROL.

NO ASYN(D_1D_0=XX)

0	0	= NO VALIDO
0	1	= 1 BIT DE STOP
1	0	= 1 1/2 BITS DE STOP
1	1	= 2 BITS DE STOP

CONTROL DE SINCRONIA.

SI (D_1D_0=00)

X	0	= SINCRONIA INTERNA
X	1	= SINCRONIA EXTERNA
0	X	= DOBLE CARCTER DE SIN.
1	X	= UN CARACTER DE SIN.

El último campo (D_7 y D_6) tiene dos significados, dependiendo en cual operación esté y del modo de operación. Para el modo asíncrono (D_1D_0=00), éste controla el número de bits de STOP transmitidos con el caracter. Puesto que el receptor siempre opera con un solo bit de STOP, D_7 y D_6 sólo controlan el transmisor. En el mod síncrono éste campo controla el proceso de sincronización.

Después de cargar la instrucción de modo de caracter (ó caracteres) SYN apropiados deben ser cargados si el modo síncrono es especificado. El ó los caracteres SYN son cargados con la última instrucción de salida de control usada para cargar la instrucciódе modo. El USART determina, de la instrucción de modo, si uno ó dos caracteres SYN son requeridos y usa la salida de control para cargar los caracteres SYN hasta que el número requerido es cargado.

La instrucción de comando controla la operación del USART dentro de la estructura de trabajo básica establecida por la instrucción de modo. El formato de la instrucción de comando se muestra a continuación :

D_0 : TxEN : TRANSMIT ENABLE.
0 = ENABLE.
1 = DISANABLE.

D_1 : DTR : DATA TERMINAL READY.
UN ALTO FORZA A LA SALIDA DTR A UN CERO.

D_2 : RxE : RECEPTOR HABILITADO.
0 = DESHABILITADO RxRDY.
1 = HABILITADO RxRDY.

D_3 : SBRK : SEND BREAK CHARACTER.
1 = FORZA A TxD A UN BAJO.
0 = OPERACION NORMAL.

D_4 : ER : ERROR RESET.
RESETEA TODAS LAS BANDERAS DE ERROR.
(PE, OE, FE)

D_5 : RTS : REQUEST TO SEND.
UN REQUEST TO SEND EN ALTO FORZA A LA SALIDA RTS* A CERO.

D_6 : IR : RESET INTERNO.
UN RESET INTERNO EN ALTO REGRESA AL 8251 AL FORMATO DE MODO DE INSTRUCCION.

D_7 : EH : ENTER HUNT MODE.
1 = ENABLE SEARCH FOR SYN CHARACTERS.

El formato de la lectura del registro de estado se muestra en la página siguiente :

D_0 : TxRDY : TRANSMITTER READY. ESTA SALIDA INDICA AL CPU QUE EL USART ESTA LISTO PARA UN CARACTER DE DATOS O COMANDO. PUEDE SER UTILIZADO COMO UNA INTERRUPCION EN EL SISTEMA O PARA OPERACIONES DE POLEO.

D_1 : RxRDY : RECEVER READY. ESTA SALIDA PASA A UN ALTO PARA INDICAR QUE EL 8251 A RECIBIDO UN CARACTER SOBRE SU ENTRADA SERIE Y ESTA LISTO PARA TRANSFERIRLO AL CPU.

D_2 : TxE : UNA SALIDA ALTA SOBRE ESTA SALIDA INDICA QUE EL CONVERTIDOR PARALELO A SERIE EN EL TRANSMISOR ESTA VACIO.

D_3 : PE : ERROR DE PARIDAD. LA BANDERA PE SE ACTIVA CUANDO ES DETECTADO UN ERROR DE PARIDAD. ES RESETEADA POR EL BIT ER DE LA INSTRUCCION DE COMANDO. " PE " NO PUEDE INHIBIR LA OPERACION DEL 8251.

D_4 : OE : ERROR DE SOBRECORRIDA. LA BANDERA OE SE ACTIVA CUANDO EL CPU NO PUEDE LEER UN CARACTER ANTES DE QUE EL SIGUIENTE LLEGUE A ESTAR DISPONIBLE. ES RESETEADO POR EL BIT ER DE LA INSTRUCCION DE COMANDO. OE NO PUEDE INHIBIR LA OPERACION DEL 8251; SIN EMBARGO, EL CARACTER SOBRECORRIDO PREVIAMENTE SE PIERDE.

D_5 : FE : FRAMING ERROR (SOLO ASINCRONO). LA BANDERA FE SE ACTIVA CUANDO UN BIT DE STOP VALIDA NO ES DETECTADO AL FINAL DE CADA CARACTER. ES RESETEADO POR MEDIO DEL BIT ER DE LA INSTRUCCION DE COMANDO. FE NO PUEDE INHIBIR LA OPERACION DEL 8251.

D_6 : SYNDET : SYNCH DETECT. ESTA LINEA SE USA SOLO EN EL MODO SINCRONO. CUANDO ESTA EN EL MODO SINCRONO INTERNO, EL USART USA SYNDET COMO UNA SALIDA PARA INDICAR QUE EL DISPOSITIVO A DETECTADO LOS CARACTERES DE SINCRONIA REQUERIDOS.

D_7 : DSR : DEFINICIONES IGUALES A LAS TERMINALES DE ENTRADA/SALIDA EXCEPTO QUE TxRDY NO ESTA CONDICIONADA POR TxEN O CTS*.

El registro recibido es comparado con el registro que contiene el caracter SYN. Si los dos registros no son iguales, el 8251 cambia un bit y repite la comparación.

EL .8259. CONTROLADOR PROGRAMABLE DE INTERRUPCIONES.

El 8259 se encarga del tratamiento de las interrupciones de hasta 8 dispositivos, pudiendo formar una red de hasta 8 8259 para poder controlar hasta 64 periféricos.

En forma general, el 8259 recibe una línea de petición del canal de un máximo de 8 periféricos. El controlador ha recibido, previamente, una mascara de interrupciones de 8 bits, procedente del microprocesador, que le indica las interrupciones que debe de atender.

Cuando el 8259 admite una interrupción, se lo comunica al microprocesador con la activación de la línea INTR. El microprocesador reconoce la interrupción con la señal INTA y, entoces, el controlador de interrupciones envia un byte al microprocesador, con el tipo de interrupción que debe ser ejecutada.

La operación de este circuito puede ser programada bajo el control del programa monitor y ofrece una gran variedad de modos de operación. El circuito puede dividirse en ocho secciones:

- El buffer de datos,
- La lógica de control de lectura/escritura,
- La lógica de control,
- El registro en servicio,
- El registro de solicitud de interrupción,
- El selector de prioridad,
- El registro de la mascara de interrupciones, y
- El buffer de cascada/comparador.

El selector de prioridad determina las prioridades de los bits del registro IRR. El buffer de datos es del tipo 3-estados bidireccional y se usa como enlace entre el 8259 y el canal de datos del sistema. Las palabras de control y de información de estado son transferidas a través de este buffer. La lógica de control de lectura y escritura permite la transferencia de información (palabras de comando y estado). Este bloque contiene los registros de comandos de inicialización y los registros de comando operacionales.

Las señales de interrupción en las líneas de entrada IR son manejadas por los registros en cascada, el registro de solicitud de interrupción IRR y el registro de servicio ISR. El registro de solicitud de interrupción se usa para almacenar todos los nivels de interrupción, los cuales estan solicitando servicio y el registro de servicio se usa para almacenar todos los niveles de interrupción los cuales son atendidos.

El registro de mascara de interrupción almacena los bits de las líneas de interrupción que están enmascaradas. El buffer de cascada/comparador almacena y compara los niveles de interrupción asignados a todos los circuitos 8259 utilizados en el sistema. Las tres líneas asociadas (CAS_0 - CAS_2) son salidas cuando el

8259 es utilizado como maestro y son entradas cuando es utilizado como esclavo.

Como se mencionó, la forma en la cual trabaja el 8259 queda determinada de acuerdo a la forma en que se programe. Existen dos

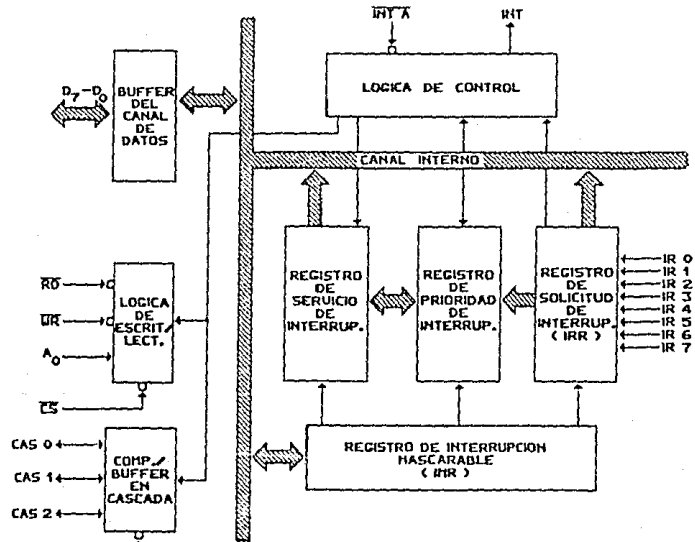


DIAGRAMA DE BLOQUES DEL 8259

tipos de comandos que se utilizan para este propósito, estos son los comandos de inicio (ICW) y los comandos operacionales (OCW).

Los cuatro comandos de inicio se usan para almacenar información en los cuatro registros internos del 8259 y poder iniciar su operación normal. Por otra parte los tres comandos operacionales permiten al 8259 iniciar variaciones en los modos de operación básicos definidos por los comandos de inicio. Los comandos operacionales pueden ser escritos en cualquier momento después del inicio. A continuación se muestra un diagrama de flujo con la secuencia de programación para el 8259.

EL 8279 : INTERFACE DE TECLADO/DISPLAY PROGRAMABLE.

DIAGRAMA DE BLOQUES.

El circuito Intel 8279 es un dispositivo de E/S de propósito general programable para teclado y display. La parte del teclado posee una interface analizadora para una matriz de 64 contactos (ó teclas) ó un arreglo de sensores. La parte del display tiene una interface analizadora para LED's, focos incandecentes y otras tecnologías populares. El 8279 tiene una sección de memoria RAM de 16 x 8 bits para el display, la cual puede ser organizada en dos de 16 x 4 bits. La memoria RAM puede ser cargada o leída por el CPU.

A continuación se presenta el diagrama de bloques interno del 8279.

DESCRIPCION DE LAS TERMINALES.

DB_0 - DB_7 (entrada/salida).
Canal de datos bidireccional : Todos los datos y comandos entre el CPU y el 8279 son transmitidos en éstas líneas.

CLK (entrada).
Reloj : El reloj proveniente del sistema es usado para generarla señal interna.

RESET (entrada).
Reset : Una señal alta en ésta terminal inicializa el 8279, después de lo cual queda en el siguiente modo :
1) Display con 16 caracteres de 8 bits en entrada o registro izquierdo.
2) Codificación del análisis del teclado con dos teclas de paro.

CS* (entrada).
Selección del chip : Una señal baja en ésta terminal habilita las funciones de interface para recibir o tranmitir.

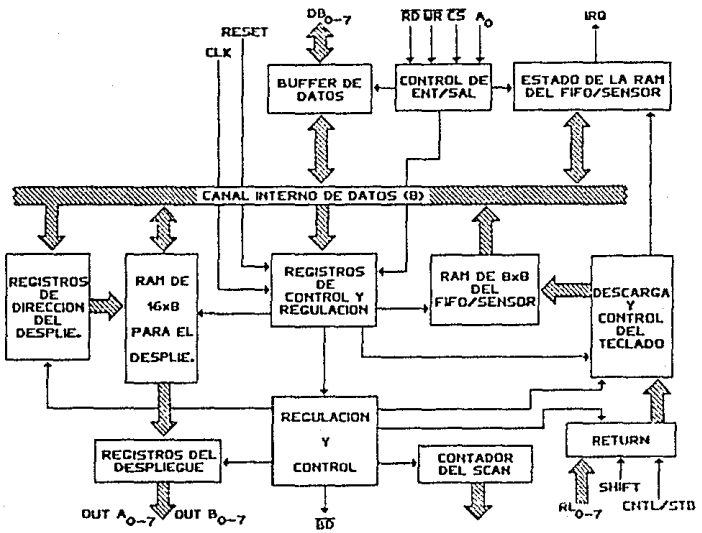
A_0 (entrada).
Buffer de direcciones : Un alto en esta terminal indica que las señales dentro o fuera son interpretadas como un comando o estado. Un bajo indica que son datos.

RD*, WR* (entrada).
Leer y escribir Entrada/Salida : Estas señales habilitan al buffer de datos para mandarlos al canal externo o recibirlos de ahí.

IRQ (salida).
Permiso de interrupción : En el modo de teclado, la línea de interrupción es alta cuando hay datos en el FIFO/sensor de la RAM. La línea de interrupción va a bajo con cada leída del FIFO/sensor de la RAM y regresa a alto si todavía hay información en la RAM. En el modo sensor, la línea de interrupción va a alto si se detecta un cambio en un sensor.

Vss y Vcc (entrada).
Tierra y alimentación, respectivamente.

DIAGRAMA DE BLOQUES DEL 8279



SL_0 - SL_3 (salida).
Líneas analizadoras : Son usadas para buscar teclas o sensores en la matriz y los dígitos del display. Estas líneas pueden ser codificadas (1 de 16) o decodificadas (1 de 4).

RL_0 - RL_3 (entrada).
Líneas de retorno : Estas entradas son conectadas a las líneas analizadoras a través de teclas o sensores. Mantienen un nivel alto hasta que un switch se cierra provocando un bajo. También sirven como entrada de 8 bits en el modo de entrada "strobed".

SHIFT (entrada).
Shift (cambio) : El estado de la entrada es almacenado con la posición de la tecla cerrada en los modos de análisis del teclado.

CNTL/STB (entrada).
Modo de entrada Control/Strobed : Para modos de teclado esta línea es usada como una entrada de control y almacena un estado deseado en una "cerradura". Es también la línea que suministra

los datos dentro de la FIFO en el modo de entrada "strobed". Mantiene un estado alto hasta que la "cerradura" o switch se hace bajo.

OUT A_0 - OUT A_3 y OUT B_0 - OUT B_3 (salidas).

Salidas : Estos dos puertos son las salidas para los registros de refresco del display de 16 x 4. Los datos provenientes de éstas salidas son sincronizados por las líneas analizadoras (SL_0 - SL_3) para multiplexar los dígitos del display. Los dos puertos de 4 bits pueden ser borrados independientemente. Estos dos puertos pueden también ser considerados como un puerto de 8 bits.

BD* (salida).

Borrado de display : Esta salida es usada para borrar el display durante el switcheo de un dígito o por el comando de borrado del display.

DESCRIPCION FUNCIONAL.

El 8279 tiene dos secciones: el teclado y el display. La sección de display controla los elementos alfanuméricos o un banco de luces indicadoras. Así el CPU es relevado del análisis del teclado o del refresco del display.

El 8279 está diseñado para conectarse directamente al canal del microprocesador. El CPU puede programar todos los modos de operación del 8279. Estos modos incluyen :

MODO DE ENTRADA :

- Teclado analizado. Con codificación (de 8x8 interruptores en el teclado) o decodificación (de 4x8 interruptores en el teclado) de las líneas analizadoras. La presión de una tecla genera la codificación de 6 bits correspondientes a su posición. Posición, cambio y estado de control son almacenados en la FIFO.

- Matriz de sensores analizadores. Con codificación (de una matriz de 8x8 switches) o decodificación (de una matriz de 4x8 switches) de líneas analizadoras. El estado de las teclas (abierto o cerrado) es almacenado en la RAM direccionado por el CPU.

- Entrada "strobed". Datos en la línea de retorno son tranferidos a la FIFO durante el "strobe" de la línea de control.

MODOS DE SALIDA :

-Multiplexado de displays de 8 o 16 caracteres que pueden ser organizados como dos de 4 bits o uno solo de 8 bits (B_0=D_0, A_3=D_7).

-Formatos de display de entrada derecha o de entrada izquierda.

Otras características del 8279 son :

- El modo de programación lo da el CPU.

- Reloj preescalado.

- Saloda para interrumpir la señal del CPU cuando hay datos del teclado o del sensor.

- Un FIFO de 8 bits para almacenar información del teclado.

- 16 bytes internos de la RAM del display para refrescarlo. Esta RAM también puede ser leída por el CPU.

PRINCIPIOS DE OPERACION.

CONTROL DE E/S Y BUFFERS DE DATOS.

La sección de control de entrada/salida usa las líneas CS*, A0, RD* y WR* para controlar el flujo de datos para y desde los diversos registros internos y buffers. El flujo de datos para y desde el 8279 es habilitado por el CS*. El tipo de información, dada o recibida por el CPU, es identificada por A0. Un uno lógico significa que la información es un comando o estado. Un cero lógico significa que la información es un dato. RD* y WR* determinan la dirección del flujo de datos a través de los buffers. Cuando el chip no es seleccionado (CS* = 1), el dispositivo está en un estado de alta impedancia.

CONTROL, TIEMPO DE REGISTROS Y TIEMPO DE CONTROL.

Estos registros almacenan los modos de teclado y display, y otras condiciones de operación programadas por el CPU. Los modos son programados presentando los comandos apropiados en las líneas de datos con A_0= 1 y mandando un WR*. El comando es mantenido en el flanco de subida de WR*. El comando es entonces decodificado y la función apropiada es activada. El control de tiempo contiene la cadena contadora de tiempo básico. El primer contador es uno entre N preescalas que puede ser programado para generar una frecuencia interna de 100 KHz los cuales dan un tiempo de análisis del teclado de 5.1 mseg. y un tiempo de "debounce" de 10.3 mseg.

CONTADOR ANALIZADOR.

El contador analizador tiene dos modos :

Modo codificador. En este modo se suministra un contador binario que debe ser decodificado externamente para suministrar las líneas analizadoras al teclado y display.

Modo de decodificador. En este modo el contador analizador decodifica los dos bits menos significativos y suministra una decodificación de uno de cuatro análisis. Cuando el teclado está en análisis de decodificación, siempre que esté el display. Esto

significa que sólo cuatro primeros caracteres en la RAM del display son mostrados.

En el modo codificador, las líneas analizadoras son salidas que se activan en alto. En el modo decodificador, las líneas analizadoras son salidas que se activan en bajo.

BUFFERS DE RETORNO, TECLADO DE ANTIRREBOTE Y CONTROL.

Las ocho líneas de retorno son mantenidas por los buffers del mismo nombre. En modo de teclado éstas líneas son analizadas. Si el circuito antirebote detecta un switch cerrado, esperando alrededor de 10 mseg. para cerciorarse que el switch permanece cerrado, si esto ocurre, la dirección del switch en la matriz más el estado de SHIFT y CONTROL son transferidos a la FIFO.

FIFO/SENSOR DE RAM Y ESTADOS.

Este bloque es una RAM de 8x8 de doble función. En el modo de teclado o entrada "strobed" funciona como una FIFO. La FIFO guardan el curso del númerode caracteres de memoria y sis está llena o vacía.

En el modo de matriz sensora analizadora IRQ es alta si un cambio en el sensor es detectado.

REGISTRO DE DIRECCIONES DEL DISPLAY Y DISPLAY DE LA RAM.

Los registros de direcciones del display contienen las direcciones de la palabra que está siendo escrita o leída por el CPU y los dos "nibbles" de 4 bits que están siendo mostrados. La lectura/escritura direccionadas son programadas por el comando del CPU, pudiendo ser activado el auto-incremento después de cada lectura o escritura. El display de la RAM puede ser directamente leído por el CPU después de que el modo correcto y las direcciones sean activadas.

Los nibbles A y B pueden ser inicializados independientemente o como palabra, de acuerdo al modo que sea activado por el CPU.

OPERACIONES DE SOFTWARE.

COMANDOS DEL 8279.

Los siguientes comandos programan los modos de operación del 8279. Los comandos son enviados por el canal de datos con $\overline{CS}$ bajo y A_0 alto y son cargados al 8279 en el flanco de subida de $\overline{WR}$.

Activación del modo de teclado/display.

Código : 0 0 0 D D K K K <---<< LSB

Donde D D es el modo de display y K K K es el modo de

teclado.

```
    D  D

    0  0  Display de 8  caracteres de 8 bits - Entrada izquierda.
#   0  1  Display de 16 caracteres de 8 bits - Entrada izquierda.
    1  0  Diaplay de 8  caracteres de 8 bits - Entrada derecha.
    1  1  Display de 16 caracteres de 8 bits - Entrada derecha.

    K  K  K

#   0  0  0  Teclado analizador codificador - Dos teclas de paro.
    0  0  1  Teclado  analizador  decodificador - Dos teclas de
               paro.
    0  1  0  Teclado analizador codificador -N teclas "rollover"
    0  1  1 Teclado analizador   decodificador   -   N teclas
               "rollover".
    1  0  0  Matriz sensora analizadora codificadora.
    1  0  1  Matriz sensora analizadora decodificadora.
    1   1  0 Entrada  "strobed",  codificación  del  display
               analizado.
    1  1  1 Entrada  "strobed", decodoficación  del  display
               analizado.

#  >>----> Default después de analizarse.
```

Programación del reloj.

Código : 0 0 1 P P P P P

Todo conteo y multiplexado de señales para el 8279 son generados por un contador interno. Este contador divide el reloj externo (terminal 3) por un entero programable. Los bits PPPPP determinan el valor de éste entero el cual tiene un rango desde 2 hasta 31.

Si la terminal 3 del 8279 está siendo controlada por una señal de 2 MHz y PPPPP se formó con 10100 para dividir el reloj por 20, se tiene una frecuencia de operación de 100 KHz.

Lectura del FIFO/sensor de la RAM.

Código : 0 1 1 A1 X A A A

X >>----> No importa.

El CPU activa al 8279 para leer al FIFO/Sensor de la RAM por medio de éste comando. En el modo de teclado analizador de auto-incremento de la bandera A_1 y los bits de dirección son irrelevantes. El 8279 automáticamente manejará el canal de datos para cada lectura subsecuente (A_0 = 0) en la misma secuencia en la cual el primer dato entró en la FIFO.

En el modo de sensado de matriz, los bits de direcciones de la RAM AAA seleccionan una de las 8 líneas del sensor de la RAM.

Si la bandera A_1 se activa (A_1 = 1), cada lectura sucesiva será originaria de la linea subsecuente de el sensor de la RAM.

Lectura del display de la RAM.

Código : 0 1 1 A1 A A A A <---<< LSB

El CPU activa al 8279 para lIer el display de la RAM por medio de éste comando. Los bits de direcciones AAAA seleccionan una de las 16 líneas del display de la RAM. Si la bandera A1 es activada (A_1=1), la dirección de la línea será incrementada después de cada lectura o escritura siguiente del display de la RAM. Por lo cual el mismo contador es usado para ambos, lectura y escritura.

Escritura en el display de la RAM.

Código : 1 0 0 A1 A A A A

El CPU activa el 8279 para escribir en el display de la RAM por medio de éste comando. Después de escribir el comando con A_0=1, todas la escrituras subsecuentes con A_0=0 estarán en el display de la RAM.

Limpiar (clear).

Código : 1 1 0 C_D C_D C_D C_F C_A

Los bits C_D son indispensables en éste comando para limpiar todas las líneas del display de la RAM según el código de borrado siguiente :

C_D C_D C_D

0	X	Todos ceros (X= No importa)
1	0	AB= Hex 20 (0010 0000)
1	1	Todos unos

Habilita la limpieza del display cuando = 1
(o por C_A = 1)

Si el bit C_F es mantenido en un estado alto, el estado de la FIFO es limpiado y la línea de salida de interrupciones es inicializada. También, el apuntador del sensor de la RAM es situado en la línea 0.

El bit C_A limpia todos los bits; tiene el efecto de combinación de C_D y C_F, éste usa el código de C_D para limpiar el display de la RAM y también limpia el estado de la FIFO.

Fin de interrupción/activación del modo de error.

Código : 1 1 1 E X X X X

X >>---> No importa.

En los modos de sensado de matriz éste comando baja la línea IRQ y habilita además la escritura dentro de la RAM. Para el modo de N teclas "rollover" si el bit E se programa como "1" el chip operará en el modo de error especial.

Palabra de estado.

La palabra de estado contiene el estado de la FIFO, error y señales no aprovechadas por el display. Esta palabra es leída por el CPU cuando A_0 es alto y CS^* y RD^* son bajas.

Lectura de datos.

Los datos son leídos cuando A_0, CS^* y WR^* son bajos. La fuente de los datos es especificada por la lectura de la FIFO o comandos de lectura del display.

Escritura de datos.

Los datos son escritos con A_0, CS^* y WR^* bajos, siempre se escriben en el display de la RAM. La dirección es especificada por la última lectura del display o escritura.

CONSIDERACIONES DE INTERFACE.

MODO DE TECLADO ANALIZADOR, 2 TECLAS DE PARO.

Hay tres posibles combinaciones de sondiciones que pueden ocurrir durante el análisis. Cuando una tecla es presionada, la lógica de antirebote es activada. Otras teclas presionadas son consideradas para los próximos dos análisis.
Si la FIFO fué vaciada, IRQ será activada para indicar al CPU que se puede entrar a la FIFO. Si la FIFO fué llenada, la tecla no será considerada y la bandera de error será activada. Una tecla es llevada a la FIFO cada vez que se presiona, no importa en que forma muchas teclas se presionen o en que orden. Si dos teclas son presionadas dentro del ciclo de antirebote, esto se considera una presión simultánea.

MODO DE ANALISIS DE TECLADO, N TECLAS "ROLLOVER".

Con N teclas "rollover", cada presión de tecla es tratada independientemente de las otras. Cuando una tecla es oprimida, el circuito antirebote espera dos análisis de teclado y checa si aún está abajo. Si es así, la tecla es llevada dentro de la FIFO. Si ocurre una presión simultánea, las teclas son reconocidas y suministradas de acuerdo al orden en que se realiza el análisis del teclado.

ANALISIS DEL TECLADO - MODO ESPECIAL DE ERROR.

Para el modo de teclas "rollover", el usuario puede programar un modo especial de error.Este se da por el comando de "Fin de interrupción/activación del modo de error". El ciclo antirebote y el chequeo de validez de la tecla es como en el modo

de N teclas. Si durante un sólo ciclo de rebote dós teclas son presionadas, ésto se considera una presión múltiple, y activa la bandera de error. Esta bandera puede ser leida en éste modo por la palabra de estado de la FIFO. La bandera de error es inicializada con el comando CLEAR normal con $C_F = 1$.

MODO DE SENSADO DE MATRIZ.

En éste modo, la lógica de antirebote es inhibida. El estado del switch sensor es llevado al sensor de la RAM. En este camino el sensor de la RAM guarda una imagen del estado de los switches en el sensor de la matriz.

La línea IRQ va a alto si cualquier cambio en el valor del sensor es detectado al final de un análisis del sensor de la matriz. La línea IRQ es limpiada por la primera operación de lectura de datos si la bandera de auto-incremento es puesta a cero, o por el comando de Fin de interrupción si la bandera de auto-icremento es puesto a uno.

FORMATO DE DATOS.

En el modo de teclado analizado, el caracter suministrado dentro de la FIFO corresponde a la posición del switch en el teclado más el estado de CNTL y SHIFT. CNTL es el MSB (bit más significativo) del caracter y SHIFT es el próximo bit más significativo. Los siguientes tres bits son el contador e indican la línea en que la tecla fué suministrada. Los últimos tres bits son del contador de columna e indican en cual línea de retorno fué conectada la tecla.

MSB			LSB
CNTL	SHIFT	SCAN	RETURN

En el modo de sensado de matriz, los datos en la línea de retorno son suministrados directamente en la línea del sensor de la RAM que corresponde a la línea en la matriz que está siendo analizada. Las entradas SHIFT y CNTL son ignoradas en este modo. Ocho puertos con entrada multiplexada pueden ser conectados a la línea de retorno y analizados por el 8279.

MSB							LSB
RL7	RL6	RL5	RL4	RL3	RL2	RL1	RL0

En el modo de entrada "strobed", los datos también son suministrados a la FIFO provenientes de la línea de retorno.

DISPLAY.

Entrada izquierda.

El modo de entrada izquierda es el formato del display más simple en cada posición del display corresponde directamente a un byte (o nibble) en el display de la RAM. La dirección 00 en la RAM es el caracter del display más a la izquierda y la dirección 15 (o dirección 7 en display de 8 caracteres) es el caracter más a la derecha del display.

Entrada derecha.

La primera entrada es ubicada en el caracters más a la derecha del display. La entrada siguiente es también ubicada en el caracter más a la derecha y el display es cambiado a la izquierda un caracter.

Auto-incremento.

En el modo de entrada izquierda, el auto-incremento causa que la dirección donde el CPU escribirá sea incrementada por uno y el caracter aparezca en la próxima localización.

Formatos de display de 8/16 caracteres.

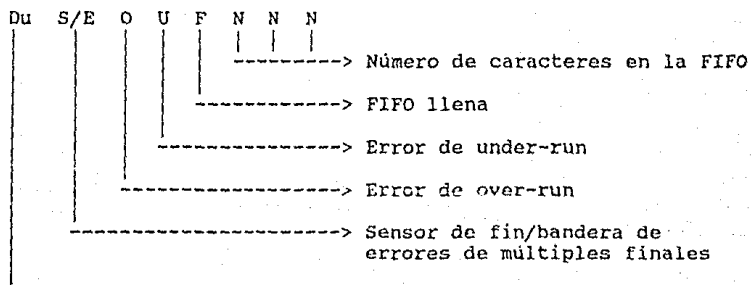
Si el modo de display es puesto para 8 caracteres, el ciclo de trabajo es doble para un display de 16 caracteres.

Estado de la FIFO.

El estado de la FIFO es utilizado en los modos de teclado y de entrada "strobed" para indicar el número de caracteres en la FIFO y para indicar donde está ocurriendo un error. Hay dos tipos de posibles errores : over-run y under-run. El over-run ocurre cuando la entrada de algún caracter, dentro de la FIFO que se encuentra llena, es intentada. El under-run ocurre cuando el CPU intenta leer la FIFO vacía.

En el modo de error especial, el bit S/E muestra la bandera de error y sirve como una indicación de donde un error, de finalización múltiple, está ocurriendo.

Palabra de estado de la FIFO.



ANEXO B.

LISTA DE PARTES Y LOCALIZACION DE COMPONENTES.

CIRCUITOS INTEGRADOS .

U1	>>---> 8088-2 Microprocesador INTEL en modo minimo.
U2	>>---> 8284 Generador de reloj.
U3	>>---> 8155H Puerto paralelo programable (3).
U4	>>---> 8251 Receptor/Transmisor Sincrono/Asincrono.
U5	>>---> 8259 Controlador programable de interrupciones.
U6	>>---> 8279 Controlador programable de despliegue y teclado.
U7	>>---> MC14411
U8-U10	>>---> 74LS373 Latches D transparentes 3-estados.
U11-U15	>>---> 74LS139 Decodificadores de 2 a 4.
U16	>>---> 74LS48 Decodificador BCD/7segmentos.
U17	>>---> 74LS138
U18-U19	>>---> MC6264 Memoria SRAM de 8k x 8 bits.
U20	>>---> MC2764 Memoria UVEPROM de 8k x 8 bits.
U21-U30	>>---> MC2864 Memoria EEPROM de 8k x 8 bits.
U31	>>---> 7490 Contador por década.
U32	>>---> 7408 Compuerta AND de 2 entradas.
U33	>>---> 1489
U34	>>---> 1488
U35	>>---> LM7812 Reg. de voltaje de +12 V @ 1.5 Amps.
U36	>>---> LM7912 Reg. de voltaje de -12 V @ 1.5 Amps.
U37	>>---> LM7805 Reg. de voltaje de + 5 V @ 3.0 Amps.
U38	>>---> 74154 Decodificador de 4 a 16.
U40	>>---> 74LS122 One shot.
U41	>>---> 74LS191 Contador de 4 bits.

U42 >>---> 74LS10 Compuerta NAND de 3 entradas.
U43-U44 >>---> 74LS47 Decodificador BCD/7 segmentos.
U45 >>---> 74LS138 Decodificador de 3 a 8.
U46 >>---> 74154
U47-U49 >>---> MH03798
U50 >>---> CD4093 Compuerta NAND Schmmitt Trigger de 2 entradas.
U70 >>---> 74LS240

TRANSISTORES .

Q1-Q1A >>---> 2N2222 A.
Q2-Q2A >>---> 2N2222 A.
Q20-Q28 >>---> EM4238.

DIODOS .

D1-D6 >>---> 1N4001
D7 >>---> Diodo Zener de 8 volts.
D8-D9 >>---> 1N4001 para el zumbador.

DETECTORS OPTICOS .

ISO1 >>---> EE-SM3 Marca ONROM.
ISO2 >>---> EE-SM3B Marca ONROM.

DESPLIEGUES .

U51-U68 >>---> HDSP-5501 Despliegues de 7 segmentos cátodo común.

RESISTENCIAS .

R1 >>---> 10 Kohms.
R2 >>---> 100 Kohms.
R3-R4 >>---> 680 Ohms.
R5-R7 >>---> 330 Ohms.
R8 >>---> 10 Kohms.
R10-R11 >>---> 10 Kohms.
R12 >>---> 150 Kohms.
R13 >>---> 560 Ohms.
R14 >>---> 1 Kohms.
R15 >>---> POT de 20 Kohms.
R16 >>---> 560 Ohms.
R17 >>---> 1 Kohms.
R18 >>---> POT de 5.6 Kohms.
R20-R28 >>---> 8.2 Kohms.
R30-R38 >>---> 330 Ohms.

CAPACITORES .

C1,C4,C7 >>---> 0.33 pF @ 50 V.
C2,C5,C8 >>---> 0.1 mF @ 16 V.
C3,C6,C9 >>---> 1 mF @ 63 V.

C10 >>----> 1 mF @ 63 V.
C11 >>----> 15 mF @ 16 V.
C12 >>----> 2.2 mF @ 63 V.
C16 >>----> 26 pF @ 50 V.

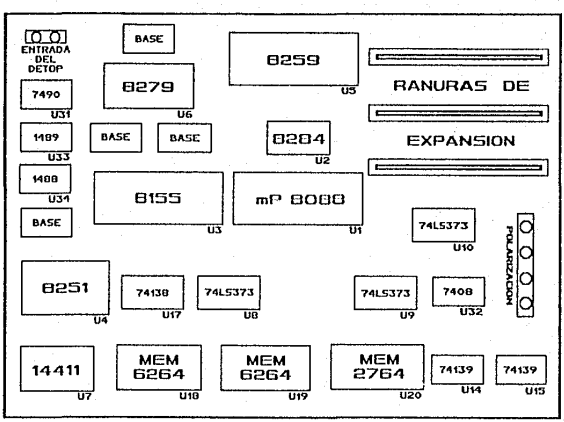
CRISTALES .

X1 >>----> 21.73 MHz.
X2 >>----> 1.84 MHz.

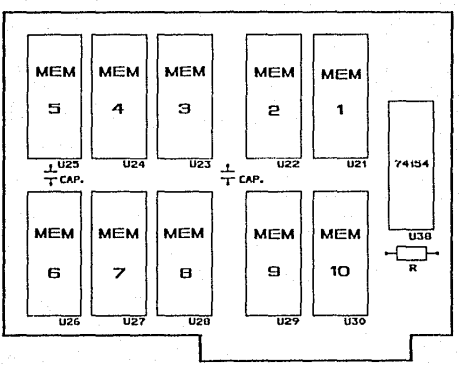
MISCELANEA .

T1 >>----> Transformador de 127/ +12 V @ 0.5 Amp.
127/ -12 V @ 0.5 Amp.
127/ + 5 V @ 2 Amps.
FUSIBLE >>----> Fusible de 127 @ 2 Amps.
SW1 >>----> Push bottom normalmente abierto.
SW2 >>----> interruptor tipo botón normalmente abierto.
CN >>----> Conector DB-25 hembra.
ZMB >>----> Zumbador de 1.5 V @ 10 mAmp.
SW11-SW26 >>----> Interruptores push bottom tipo tecla para formar el teclado matricial.

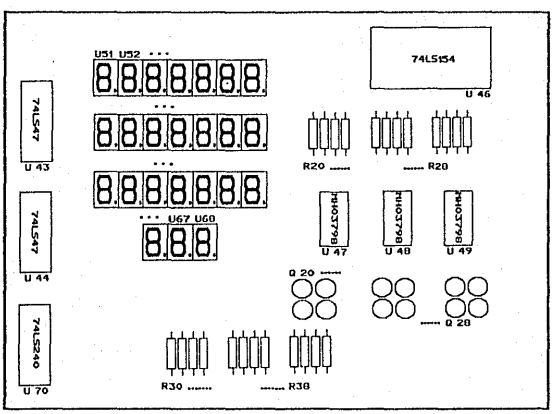
A continuación se muestra la localización de los circuitos integrados en la tarjeta madre, en la tarjeta de memoria y en la tarjeta de despliegue y teclado.



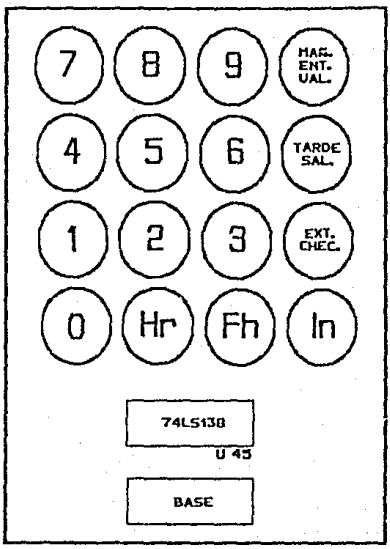
COMPONENTES DE LA TARJETA MADRE



UBICACION DE LOS COMPONENTES
EN LA TARJETA DE MEMORIA



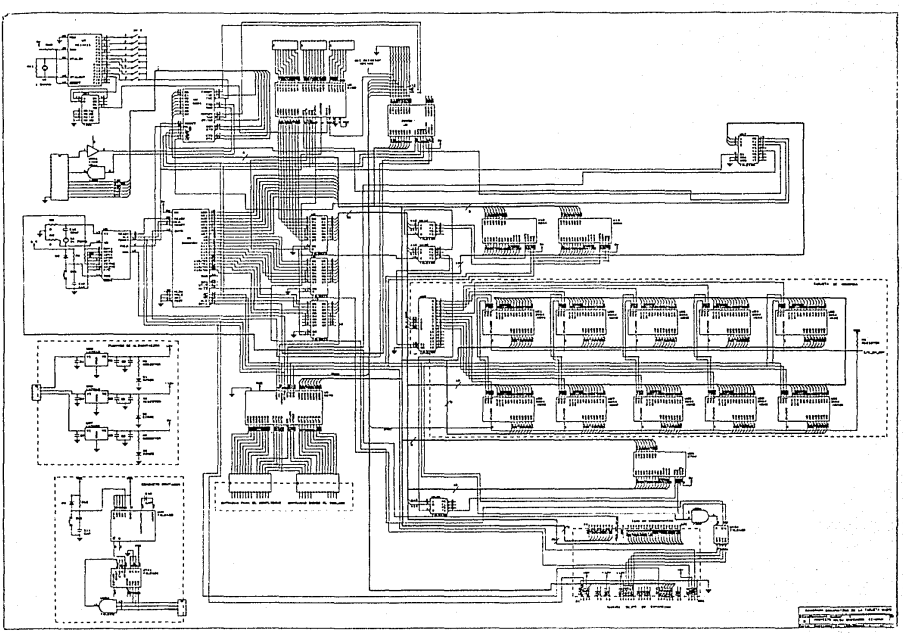
DISTRIBUCION DE COMPONENTES EN LA TARJETA DE DESPLIEGUE.

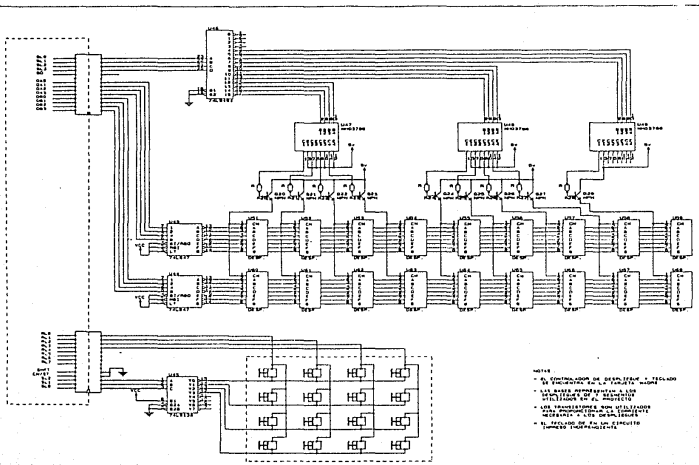


DISTRIBUCION EN LA TARJETA DE TECLADO.

ANEXO C

DIAGRAMAS ESQUEMATICOS





NOTAS

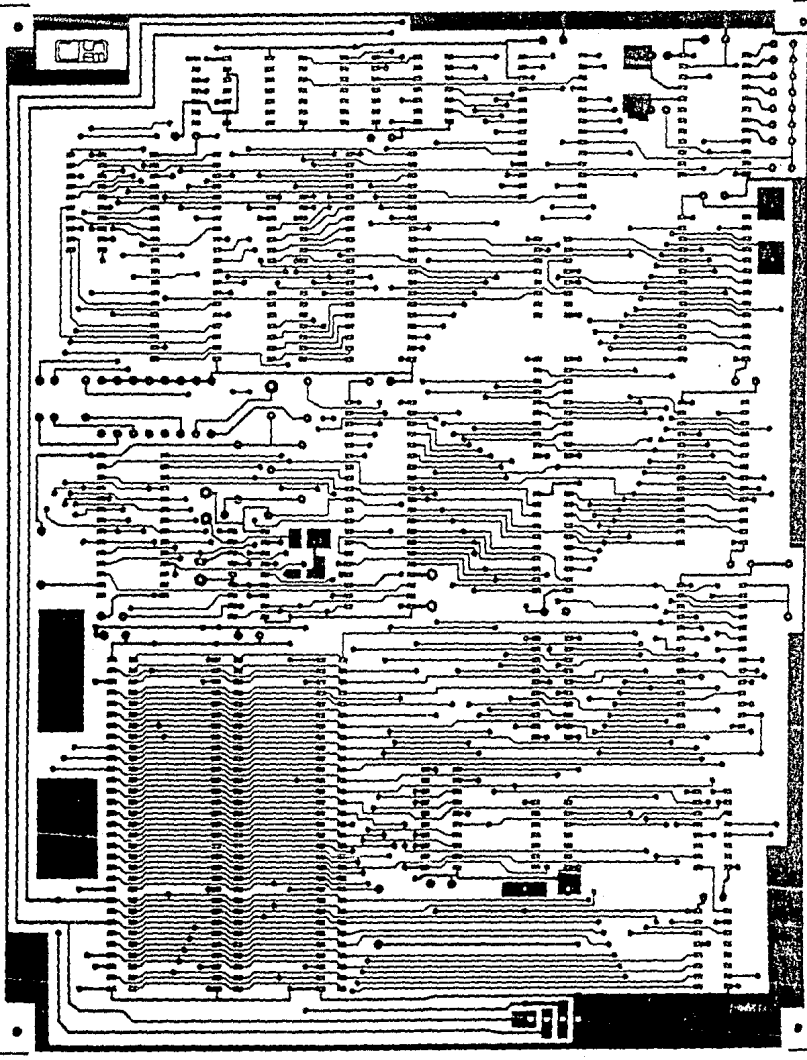
- EL CONTROLADOR DE DESPLIEGUE Y TECLADO SE ENCUENTRA EN LA TARJETA MADRE
- LAS BASES REPRESENTAN A LOS TRANSISTORES QUE SE ENCUENTRAN UTILIZADOS EN EL PROYECTO
- LOS TRANSISTORES SON UTILIZADOS COMO INTERRUPTORES PARA ACTIVAR LOS DISPLAYS Y LOS LEDS INDIVIDUALES
- EL TECLADO SE ENCUENTRA EN UN CIRCUITO IMPRESO INDEPENDIENTE

	DISEÑO ESQUEMATICO TARJETA DE DESPLIEGUE Y TECLADO	
	PROYECTO "RELOJ CRONOMETRO ESTACIONAL"	

ANEXO D

MASCARILLAS DE CIRCUITOS IMPRESOS

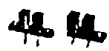
LADO DE SOLDADURA



LADO DE COMPONENTES

ANEXO E

HOJAS DE ESPECIFICACIONES DEL DETECTOR OPTICO

Classification	Appearance	Model	Sensing method	Output configuration	Sensing distance (mm)	Aperture size (mm)	Terminal type	Remarks	Page
Photomicrosensor (transmissive)		EE-SX1042	Slot type	Photomicrosensor	5.0	0.5	PCB	• Up to 12 mm optical axis clearance and 5 mm beam width • Perfect for printer ribbon detection	80
Photomicrosensor (transmissive)		EE-SM3	Slot type	Photomicrosensor	3.0	3.0 dia. 0.5	PCB	• Red LED for easy monitoring of operation • Low drawing current, high output • Low-cost type for general purposes	84
		EE-SM3B							
		EE-SJ3W-B							88
		EE-SK3W-B							
Photomicrosensor (reflective)		EE-SB5	Reflective type	Photomicrosensor	5.0*		Solder	• Equipped with visible light cutting filter • High efficiency and output • Low-cost type for general purposes	92
		EE-SD5-B					PCB		
		EE-SF5					Solder	• Equipped with visible light cutting filter • High efficiency and output • Low-cost type for general purposes	96
		EE-SF5-B							
		EE-SY101			1.0**		PCB	• Subminiature type 4 mm dia. (EE-SY101) • With holder for easy mounting (EE-SY102)	100
		EE-SY102							
		EE-SY110			5.0*			• Compact Reflective Photomicrosensor • Low-profile, 3 mm(d) x 4.8 mm tall • Low cost	104
		EE-SY120/121/122			1.0**			• Phototransistor output • Thin and compact • Oblivious to external interference light • Mounts directly on PCB	108
		EE-SY148			5.0*			• Low-profile facilitate stacking • Phototransistor output • Low cost • High-sensitivity reflective sensor	112
Photo IC output type		EE-SY310	Reflective type	DARK-ON	5.0*	—	PCB	• Low cost • Automatic dip soldering and immersion cleaning possible	116
		EE-SY410		LIGHT-ON					
Photodarlington transistor output type		EE-SY201	Reflective type	Photodarlington transistor	4.0*	—	PCB	• Low drawing current, high output • Red LED	120
Photoreflector (transmissive)		EE-SMR3-1 EE-SMR1-1	Reflective type	Phototransistor	1.0*	—	PCB	• Easy installation in limited spaces • Extended life and higher reliability	124

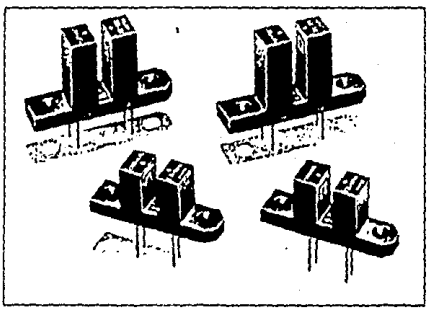
* The object to be sensed: white paper with a reflection factor of 90%.
** The object to be sensed: Al vacuum evaporated glass

OMRON

PHOTOMICROSENSOR

EE-SM3

- Low cost Slot Type with Low Driving Input and High Transmission Efficiency
- Only 3 mA required as the forward current of an LED (1.5 mA min. output at 3 mA input)
- Best suited to drive CMOS IC
- Two versions with different slot depths available (13 mm for Model EE-SM3 and 7 mm for Model EE-SM3B)



Ordering Information

Sensing method	Slot type	
Slot width	3.0 mm	
Slot depth	13 mm	7 mm
Aperture size	0.8 × 0.8 mm	
Output configuration	Photodarlington transistor	
Terminal type	PC board	
No. of terminals	4	
Model	EE-SM3	EE-SM3B

Specifications

Absolute Maximum Ratings (Ta = 25°C)

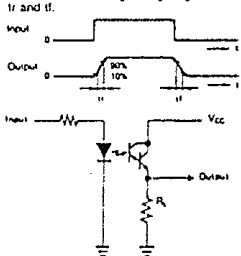
Item		Symbol	Value
Input	Forward current*	I_F	15 mA
	Reverse voltage	V_R	4 V
Output	Collector-emitter voltage	V_{CEO}	24 V
	Collector current	I_C	20 mA
	Collector power dissipation*	P_C	75 mW
Operating temperature**		Topr	−20° to 60°C
Storage temperature		Tstg	−20° to 80°C

* Refer to "Temperature rating diagram" when ambient temperature is not at 25°C.
** Without freezing or condensation.

Electrical Characteristics (Ta = 25°C)

Item		Symbol	Limits	Test Conditions
Input	Forward voltage	V_F	2 V typ. 2.6 V max.	I_F = 15 mA
	Reverse current	I_R	5 μA max.	V_R = 4 V
Output	Dark current	I_D	20 nA typ. 250 nA max.	V_{CE} = 10 V 0 lx
Input/ output (coupled)	Light current (Collector current)	I_L	1.5 to 120 mA	I_F = 3 mA V_{CE} = 10 V
	Collector-emitter saturation voltage	V_{CE} (sat)	0.9 V typ.	I_F = 3 mA I_L = 0.5 mA
	Rise time (See Note.)	t_r	180 μs typ	V_{CC} = 5V R_L = 0.1 kΩ I_L = 10 mA
	Fall time (See Note.)	t_f	60 μs typ.	

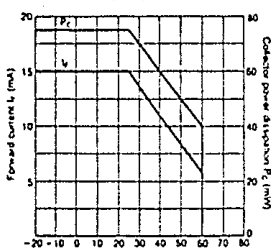
Note
Refer to the following timing diagram for tr and tf.



Engineering Data

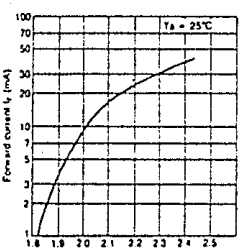
Note: The following are typical values (except for "Temperature rating diagram"). Actual values vary.

Temperature rating diagram*

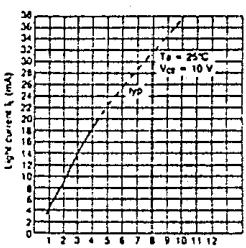


*Avoid use beyond the absolute maximum ratings

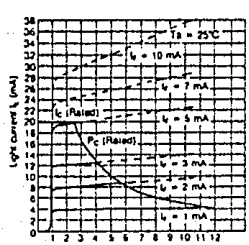
Input characteristics



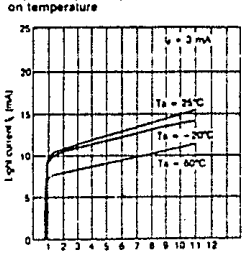
Input/output characteristics



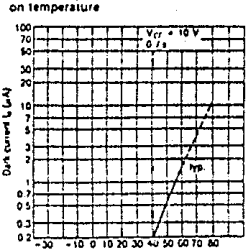
Output characteristics



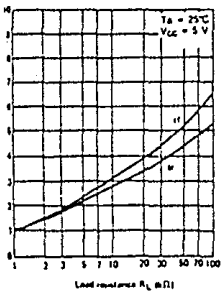
Dependency of output characteristic on temperature



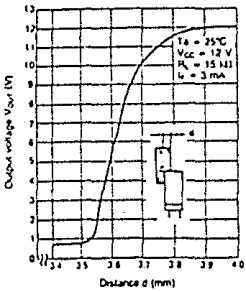
Dependency of dark current on temperature



itching characteristics

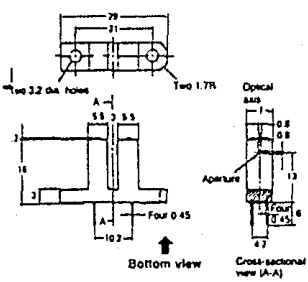


Sensing position characteristics
EE-SM3

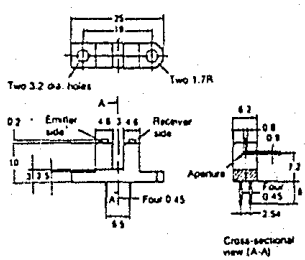


)imensions

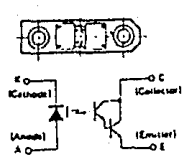
!E-SM3



EE-SM3B



'erminal arrangement
bottom view)



B I B L I O G R A F I A

Osborne Adam, Kane Gerry.
Osborne 16-bit, Microprocessor Handbook.
Mc GRAW-HILL, 1981.

Morgan Christopher L., Waite Mitchell
Introducción al microprocesador 8086/8088
Mc GRAW-HILL, MEXICO, 1984.

Bradley David J.
Assembly language programming for the IBM personal computer.
PRENTICE-HALL, 1984

IBM Personal Computer Hardware Reference Library.
Technical Reference TX., Abril 1984.

Omron Tateisi Electronics Co.
Photomicrosensor Manual.
1988.

Motorola Semiconductors Data Library CMOS.
Volumen V, Serie B, 1976.

Intel Corporation.
Microprocessors & Peripherals Handbook
Volúmen 1 y II
1988

Motorola
Linear Interface Integrated Circuits.
1986.

National Semiconductor Corp.
TTL Data Book.
1987.

National Semiconductor Corp.
Discrete Semiconductor Products Databook
1989.

National Semiconductor Corp.
MOS Memory Databook
1984.

Exel Microelectronics Inc.
Memory Databook
Abril, 1987

Texas Instruments
Optoelectronics Databook
1982.

Microsoft
Macroassambler V 5.0, Guía de usuario
1988.

Tesis Profesional
Contreras Nieto, Jaime
Garcia Peña, Adolfo
"Desarrollo de un sistema electronico para
la adquisición, recuperación y procesamiento
de datos de demanda de energía eléctrica y
tiempo de interrupción".
1989.