UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SOFTWARE CONTROLADOR DE UNA RED DE ANILLO PARA PC's

FALLA DE ORIGEN

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N :

JOSE BOGAR PEREZ GUTIERREZ

DORA ALEJANDRA TIPACAMU RIOS

DIRECTOR DE TESIS: ING. GABRIEL CASTILLO HERNANDEZ

MEXICO, D. F. 1995

Nuestros más sinceros agradecimientos:

Al Ing. Gabriel Castillo Hernández
Por proponer el tema de tesis, dirigirnos y asesorarnos durante la realización de ésta.

Al Instituto de Física
Por las facilidades otorgadas para la edición e impresión de nuestro trabajo.

Al Dr. J. Guadalupe Pérez Ramírez
Por sus valiosos consejos para la redacción de este trabajo.

A mis padres y a mi hermano
Quienes durante toda mi vida me han apoyado y brindado todo su amor.
Aleli.

A Bogar
Quien me ayudó a ser feliz en la Facultad.

A mis padres
Por su amor, confianza, apoyo y paciencia.
Bogar

A mis hermanos
Por permitirme usar la computadora cuando la utilizaban para jugar.

A Ale
Por su paciencia en esos momentos de desesperación.

A Nuestros amigos
Por esos momentos...

INDICE

1. INTRODUCCION

En nuestros días las redes de computadoras han adquirido una gran importancia, teniendo por lo tanto un gran desarrollo. Estas han aumentado la productividad de las empresas, debido a su capacidad de permitir el intercambio de información entre computadoras locales e incluso entre computadoras de diferentes empresas localizadas en cualquier parte del mundo. Esta tendencia permite el flujo de información casi instantáneo entre los diferentes departamentos de una organización. Para establecer este flujo de información dentro de la organización se implementan diferentes tipos de redes. Dentro de éstas, una de las más importantes y que ha tomado mayor auge es la red de área local. Para la implementación de este tipo de red, las organizaciones de gran tamaño adquieren equipo especial (hardware y software) a altos costos.

Para organizaciones más pequeñas, el invertir en este equipo significa exceder su presupuesto destinado para estos fines. Pero el no invertir en equipo para la implementación de una red local, puede resultar en una baja de productividad, lo cual generaría pérdidas para la organización.

Ante este panorama, surgió la opción de desarrollar software que utilice el hardware básico de una PC, evitando el gasto de hardware adicional. Dado el hardware disponible la topología en anillo de paso de estafeta (Token Ring) resultó ser la más conveniente, ya que no es necesario equipo adicional para su implementación.

Los puertos seriales de la PC son el canal de flujo de información entre PC's. Así, el software desarrollado controla los puertos seriales, permitiendo que la topología en anillo funcione de acuerdo a sus requerimientos.

El objetivo de ésta tesis es desarrollar software controlador de hardware, permitiendo así profundizar en el conocimiento de la arquitectura de la PC, en el desarrollo de redes, así como en el desarrollo de manejadores de dispositivos.

2. BASES TEORICAS

En este capítulo se presentan los conceptos necesarios para el desarrollo del manejador de dispositivos, las rutinas de atención de interrupciones, así como el protocolo, y los programas de comunicación.

2.1 REDES DE COMPUTADORAS

Gracias a las capacidades de las redes de computadoras, hoy en día es posible reservar cuartos en hoteles, boletos para cualquier evento, realizar operaciones financieras de todo tipo, pedir cualquier tipo de información a cualquier persona conectada a una red en todo el mundo. Un ejemplo típico es la red internacional INTERNET donde podemos consultar miles de temas en tan solo unos segundos. Se calcula que en todo el mundo están conectados 30 millones de computadoras de todos tipos, desde supercomputadoras hasta computadoras personales.

Podemos definir una red de computadoras como un grupo de computadoras conectadas entre si utilizando diversos medios con el fin de intercambiar información. El tipo de información que circula a través de una red puede ser desde un simple archivo de texto hasta audio y video. Esto es posible, gracias al avance en las comunicaciones por medio de satélites y fibra óptica.

TOPOLOGIAS DE RED

A la forma en la que está configurada una red de computadoras se le conoce como *topología*. La topología es en sí la forma física en la que se encuentran conectados los elementos de la red.

Para establecer la topología de una red el diseñador debe plantearse los siguientes objetivos:

- Proporcionar máxima confiabilidad, para garantizar la recepción de todo el tráfico.

- Encontrar el camino más conveniente para el tráfico, considerando que este sea el más económico y confiable dentro de la red.
- Proporcionar al usuario final un tiempo de respuesta aceptable.

La confiabilidad en la red se refiere a la capacidad que tiene la misma de transportar los datos de una computadora a otra sin que presente errores. Esto también incluye la capacidad de recuperación de datos perdidos, ya sea por un fallo en el canal de comunicación, o en alguna de las máquinas que conforman la red. La confiabilidad debe enfocarse también, al mantenimiento preventivo y correctivo de la red. Así cuando se presenta algún error en cualquiera de los componentes de la red, el protocolo debe corregirlo y aislarlo para que no afecte al resto de la red.

Para encontrar el camino más económico posible, es necesario: primero, minimizar la longitud real del canal que une los componentes, lo que implica que la información viaje a través del menor número de componentes intermedios. Y segundo proporcionar a cada actividad, un canal especial y económico, de acuerdo a sus necesidades. Por ejemplo, transmitir datos de baja prioridad por un canal de baja velocidad, como la línea telefónica, lo que resulta más barato, que si se emplea un canal de alta velocidad que transmite vía satélite.

Para reducir al mínimo el tiempo de respuesta hay que acortar el tiempo de retraso entre la transmisión y recepción de los datos de una computadora a otra. Además, la velocidad y la eficacia con la que los datos viajan deberá ser lo más elevado posible.

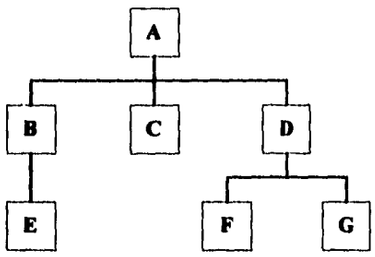
Topologías de red más comunes

- Topología jerárquica o árbol.
- Topología horizontal. (bus)
- Topología en estrella.
- Topología en anillo.
- Topología en malla.

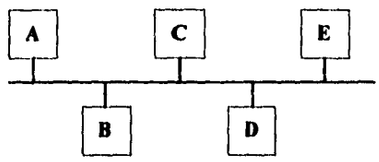
La estructura de las topologías mencionadas, se muestran en la figura 2.1.1. Esta nos permite observar las diferencias que existen entre cada topología.

LA TOPOLOGIA ANILLO

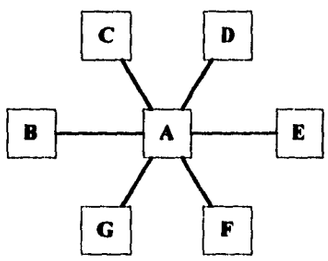
Dentro de las topologías utilizadas por redes locales de computadoras (LAN), la mejor alternativa es la topología en anillo. Esta topología ha disfrutado de gran popularidad en Europa y ahora en Estados Unidos donde se ha utilizado desde 1985, cuando IBM hizo la introducción de esta topología de anillo al mercado. Se le llama así por su aspecto circular del flujo de datos. La organización en anillo resulta muy interesante ya que sólo en raras ocasiones se presentan problemas de embotellamiento, mismas que son comunes en topologías en estrella o en árbol. El problema más importante es que todos los componentes del anillo se encuentran conectados a un mismo canal, así, si falla el canal entre dos nodos, toda la red se ve interrumpida. Para esto algunos fabricantes han incluido en los componentes de la red, un canal de seguridad o el uso de un conmutador que redirige la información, evitando pérdida de información, o un fallo en toda la red.



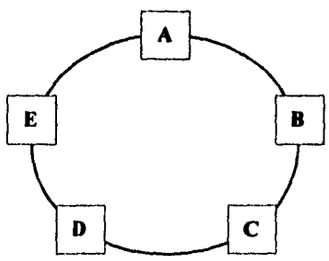
Topología Jerárquica o de árbol



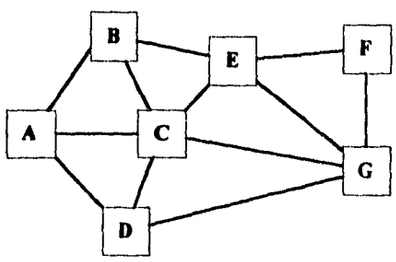
Topología Horizontal o en bus



Topología en estrella



Topología en anillo



Topología en malla

Fig. 2.1.1. Topologías de red

CARACTERISTICAS DE LA TOPOLOGIA EN ANILLO

Formada por un cierto número de computadoras, la topología en anillo se conecta de una computadora a otra a través de un cable unidireccional. Los datos transmitidos son enviados bit a bit por todo el anillo y viajan hasta alcanzar su destino, lo que implica que cada computadora recibe un dato y luego lo retransmite.

Para que una red de comunicaciones implementada con esta topología trabaje de manera correcta son necesarias algunas funciones en cada computadora, dichas funciones son la inserción y recepción de datos, así como la remoción de los mismos.

Los datos que viajan a través de la red, son organizados en paquetes de información, para evitar su pérdida. Los paquetes de información cuentan con un campo de identificación con el objeto de que cada computadora analice si la información que acaba de llegarle esta destinada a ella o no. En el caso afirmativo el paquete de información se copia. De lo contrario se retransmite. Dada la topología de la red los paquetes de información siempre se encontrarán recorriendo todo el anillo hasta que una de sus computadoras los copie, de esta forma se verán suprimidos del anillo.

Una gran variedad de estrategias pueden ser usadas para determinar cuando y como los paquetes de información deben ser insertados en el anillo. A estas estrategias se les llama *protocolos de control.*

Cada computadora dentro de la red cuenta con dos funciones importantes: la primera, contribuir al buen funcionamiento del anillo pasando a la siguiente computadora toda la información que le llega, y segundo, permitir a las computadoras que se encuentran en comunicación enviar y recibir datos. De acuerdo con lo anterior se contemplan dos estados:

- El estado de repetición.
- El estado de transmisión.

- ***Estado de Repetición***

En este estado cada bit que es recibido se retransmite con un pequeño retraso, el cual es necesario para permitir a la computadora realizar ciertas funciones durante ese tiempo. Generalmente el tiempo de retraso es el equivalente al tiempo que se emplea para enviar un bit de información. Las funciones que realiza cada computadora consisten en analizar la información, verificar si se trata de un paquete destinado a ella y, en caso de no ser así retransmitirlo.

- ***Estado de Transmisión***

Cuando se posee información que será enviada y cuando, de acuerdo a los protocolos de control se tiene el permiso para hacerlo, se inicia el estado de transmisión. En este estado la computadora recibe bits y los retransmite a su destino.

Durante el periodo de transmisión los bits que son recibidos pueden presentarse de diversas formas y de acuerdo a ello serán tratados. En algunos protocolos de control existe la posibilidad de que más de un paquete de información circule a través del anillo al mismo tiempo. Si la máquina que transmite información recibe bits que no han sido enviados por ella, los almacenará hasta que pueda retransmitirlos.

Para conectar cada nodo de la red (entendiendo como nodo cada computadora) a la red misma, puede utilizarse cable coaxial, fibra óptica ó par trenzado.

La razón por la que el anillo fue aceptado tardíamente en algunos países, se debió al número de problemas que esta topología presenta. Entre ellos, cuando uno de los nodos es deshabilitado la red entera también lo estará. Instalar un nuevo nodo requiere de introducirle los identificadores de los nodos adyacentes para que pueda establecer comunicación dentro de la red. Debido a que la red está cerrada, cuando existe información circulando a través de ella son necesarios para evitar errores, programas destinados a eliminar y respaldar dicha información. Otros problemas que se presentan con más frecuencia en este tipo de topología es el número limitado de repetidores que pueden conectarse a la red. Este problema se

suscita al aumentar el número de computadoras, los problemas de mantenimiento, retrasos en la transmisión y recepción de información se ven multiplicados. Un límite razonable para una red con topología en anillo es aproximadamente de unos 100 repetidores.

A través del tiempo se ha experimentado con una variedad de algoritmos de control de la topología en anillo. La técnica Paso de estafeta (*Token Ring)* es de las más importantes y quizá una de las más viejas técnicas de control. Esta técnica está basada en el uso de un pequeño paquete llamado estafeta (*token)*, que circula por todo el anillo. Al inicializar la red, la estafeta también se inicializa y se lanza por la red con la etiqueta de *estafeta libre.* Cuando una estación o nodo de la red requiera transmitir, tendrá que esperar hasta detectar la estafeta pasando por ella. Entonces ésta será modificada de estafeta libre a *estafeta ocupada.* Por su parte la estación libera la estafeta a la red e inmediatamente después la información. Si en el momento en que la estafeta se encuentra etiquetada como ocupado otra estación desea transmitir tendrá que esperar hasta que detecte una etiquetada como libre. Para que la estación que utilizó la estafeta pueda fijar a ésta como libre deberá cumplirse lo siguiente:

- La estación ha terminado por completo la transmisión de su paquete de información.
- La *estafeta ocupada* ha regresado a la estación que la usó.

El Paso de estafeta (*token ring)* comparte muchas ventajas con el Paso de testigo en bus (*token bus).* Por ejemplo, su principal ventaja es el control del tráfico a través de la red. Ya sea que las estaciones transmitan información por medio de la estafeta o bien por medio de prioridades, donde las estaciones de mayor prioridad tienen ventaja sobre las otras para recibir la estafeta. La principal desventaja de la topología del paso de estafeta es el requerimiento de la estafeta misma. Esto se debe a que se debe proporcionar mantenimiento constante. Algunos problemas que se presentan con el uso de la estafeta, es la duplicidad y la pérdida de la misma. Por ello una de las estaciones debe ser fijada como monitor para verificar si es necesario insertar una nueva estafeta o si no existe más de una en la red.

2.2 COMUNICACION SERIAL

Una de las interfases más conocidas y utilizadas, es el puerto de comunicación serial. Este tipo de interfase es utilizada para conectar equipos periféricos, por ejemplo: terminales, modems, otras computadoras e impresoras. El puerto de comunicación serial, incluso permite realizar transferencias de bloques de datos entre el equipo conectado y la computadora. En los siguientes párrafos se hablará de los tipos de interfase serial que existen, así como la definición de puerto.

COMUNICACION SINCRONA Y COMUNICACION ASINCRONA

Para que la comunicación entre dos computadoras, se realice, es indispensable que se notifiquen entre ellas que son capaces de realizarla. Deben disponer de un mecanismo con la que ambas lleven el control de la transmisión. Estas actividades, forman parte de un protocolo de comunicación y se les conoce como sincronización.

Para conseguir la sincronización entre los sistemas, se emplean dos convenios de organización o de formato en los datos: uno es el formato asíncrono, en que cada byte o caracter, incluye señales de sincronización al inicio y al final (arranque y parada). El segundo convenio de organización es la transmisión síncrona.

La ***comunicación síncrona*** emplea canales separados de reloj o códigos autosincronizados (Fig. 2.2.1). En los formatos síncronos se suprimen las señales de arranque y parada que acompañan a cada caracter. Las señales que son utilizadas por este tipo de formato se les llama bytes de sincronización o banderas. Algunos problemas se presentan , cuando un mensaje de gran longitud y sin bits de arranque/parada es enviado, debido a que el receptor puede desplazarse con respecto a la señal. Para evitar problemas como el expuesto, los sistemas envían repetidamente los caracteres de sincronización, previniendo la desincronización. (Fig. 2.2.2)

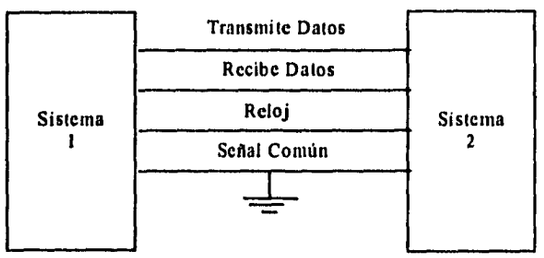


Fig. 2.2.1 Comunicación síncrona

La comunicación síncrona inicia cuando el sistema transmisor envía un caracter de sincronización al sistema receptor. El receptor lee el caracter y una vez que lo ha identificado, da inicio a la lectura de caracteres procedentes de la línea de transmisión, hasta que concluya el bloque de datos, o bien hasta que se presentan problemas de sincronización entre sistemas.

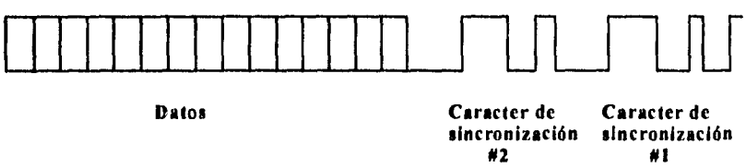


Fig. 2.2.2 Caracteres de sincronización

La comunicación síncrona es utilizada principalmente en aplicaciones que requieren alta velocidad durante la transmisión.

La ***comunicación asíncrona*** elimina la necesidad de la señal del reloj, por ello los sistemas en comunicación se encuentran conectados por tres líneas: la de transmisión, la de recepción de datos y la señal común.(Fig. 2.2.3)

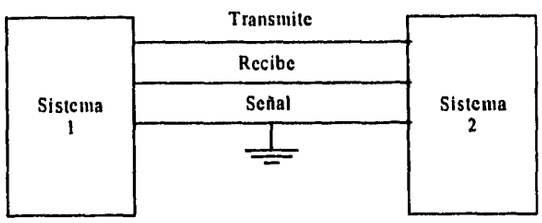


Fig. 2.2.3 Comunicación asíncrona

La transmisión de los datos se ejecuta enviando caracter por caracter. El sistema receptor examina cada uno de los caracteres y busca los bits de sincronización que se localizan al inicio y al final de cada caracter.

En el bloque de transmisión[1], el primer bit que se transmite o se recibe es el ***bit de arranque*** (Start Bit) seguido del bit menos significativo del caracter y de sus 6 bits restantes, el ***bit de paridad*** y finalmente el ***bit de parada*** (Stop Bit). Dependiendo del esquema de comunicación que se use, pueden existir 1, 1½ o 2 bits de parada en cada caracter.(Fig. 2.2.4)

La misión de los bits es primero avisar al receptor que está llegando un dato y segundo, darle tiempo suficiente para que realice las funciones de sincronía necesarias, antes de que llegue el siguiente caracter.

El bit de arranque indica al receptor que los siguientes 8 bits que recibirá constituyen un byte, es decir un caracter. Para identificar al bit de arranque y de parada se utilizan niveles lógicos de voltaje, permitiendo que existan únicamente dos valores: 0 y 1.

[1]El bloque de transmisión es cada caracter incluyendo los bits de sincronización.

El ***bit de arranque*** corresponde al nivel 0 lógico y el de ***parada*** al 1 lógico, proporcionando al receptor la posibilidad de saber que el caracter ha sido enviado. La diferencia entre ambos niveles lógicos permite al receptor identificar cuando empieza un nuevo bloque de transmisión.

El ***bit de paridad*** es utilizado para la detección de errores en la transmisión de datos. El objeto de este bit es indicar pérdida de información. La paridad puede definirse como par o impar. Cuando el receptor lee el caracter y cada uno de los bits que lo conforman, si encuentra el bit de paridad encendido, verifica de que tipo es y luego si se trata de paridad par cuenta el número de 1's contenidos en el caracter, el resultado debe coincidir con el tipo de paridad escogida, es decir obtendrá un número de 1's par o impar. Si no coincide el resultado con el tipo de paridad se detectará un error.

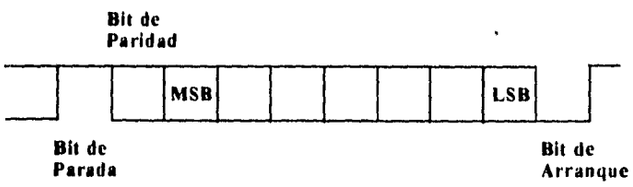


Fig. 2.2.4 Bits de sincronización

La comunicación asíncrona es bastante utilizada por los sistemas de comunicación, dada la economía y sencillez, de las interfases que la emplean.

PUERTOS DE COMUNICACION

Un puerto de comunicación tiene por objetivo enlazar un canal de comunicaciones con una computadora y proporcionar las funciones de movimiento de datos desde y hacia el dispositivo, con el que la máquina o computadora se comunica.

Los puertos de comunicaciones generalmente son pequeños microprocesadores que cuentan con su propio reloj, registros, memoria e incluso CPU. La capacidad e inteligencia de los puertos varía según su diseño y tipo de interfase.

Los puertos de comunicación básicos de cualquier sistema PC son los puertos serial y paralelo. Los puertos seriales son utilizados principalmente por dispositivos que se comunican con el sistema, bidireccionalmente, como son los modems, los mouse, los scanners, los digitalizadores y otras computadoras. A los puertos de comunicación se conoce también como: adaptador de comunicaciones, puerto serie, tarjeta serie. Por su parte los puertos paralelos son usados en impresoras y operan unidireccionalmente, aunque en ocasiones pueden operarse bidireccionalmente con la ayuda de software especial.

En esencia la diferencia entre los puertos paralelo y serial, es la forma en la que envían o reciben los datos. En el paralelo los bits de cada caracter, se transmiten todos al mismo tiempo; mientras que en el serial, los bits se transmiten uno por uno.

EL UART Y USART

La comunicación vía puerto serial es muy utilizada en el mundo de las comunicaciones, por ello se han desarrollado dispositivos electrónicos como los siguientes: el UART (*Transmisor/Receptor de comunicaciones asíncronas*) y el USART (*Transmisor/Receptor de comunicaciones síncronas/asíncronas*).

El UART 8250 es el dispositivo encargado de controlar la comunicación serial en las tarjetas seriales, que encontramos en las computadoras IBM y compatibles.

Conociendo que la Unidad Central de Proceso (CPU) procesa los datos en paralelo, es importante mencionar que el UART y el USART tienen la habilidad de hacer las conversiones necesarias para la transmisión de datos con el CPU. La primera conversión que se efectúa, se presenta cuando el dispositivo recibe del CPU los datos a

ser enviados (conversión paralelo a serie), y la segunda cuando los datos son enviados al CPU (serie a paralelo).

Igualmente, el UART y el USART poseen la capacidad de identificar cada caracter que se transmite, así como cada uno de los bits de sincronización (arranque, parada, y paridad). Cuando se utiliza el bit de paridad, éste permite que los dispositivos detecten errores en el momento de la transmisión.

A continuación se describirá en detalle el UART 8250, dado que es el dispositivo que se utilizó en el presente trabajo.

El UART 8250 cuenta con cuatro interfases:

La del ***microprocesador***, la de ***transmisión***, la de ***recepción*** y la del ***protocolo,*** mismas que se muestran en el diagrama de bloques de la figura 2.2.5.

INTERFASE DEL MICROPROCESADOR.

La interfase consiste en un bus de datos bidireccional de 8 bits (D_0 - D_7) y 3 líneas de control CS(Chip select), RD (Read) y WR (Write).

El microprocesador y el UART 8250 se conectan por el bus de datos. Como se mencionó, el bus trabaja en dos direcciones; en la primera, se leen caracteres del receptor y en la segunda, se envían caracteres por el transmisor. Además, entre el UART 8250 y el microprocesador circula información, por el bus de datos, referente a las instrucciones de modo de operación, instrucciones de comandos de operación y de estado.

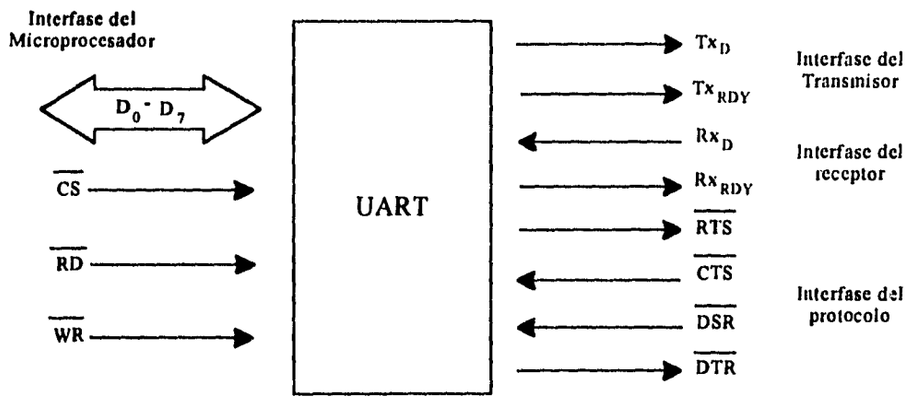


Fig. 2.2.5 Diagrama de bloques del UART

El UART 8250 se programa a través de software, empleando diferentes modos de operación. Para ello, dentro del UART 8250, existen diversos registros de control. En estos registros, se fijan los parámetros que permiten obtener el modo de operación elegido. Los parámetros a programar, son referentes a cada caracter que se va a transmitir o se va a recibir, por ejemplo: tamaño del caracter, tipo de paridad y 1, 1½ ó 2 bits de parada. (Stop bits).

En el UART 8250, también se programan otros eventos como interrupciones y velocidad de transmisión e incluso, puede verificarse el estado en que se encuentran los registros de control, así como los errores que puedan presentarse.

INTERFASE DEL TRANSMISOR Y DEL RECEPTOR

La interfase del transmisor se compone de 2 líneas (Figura 2.2.5): la línea de Transmisión de datos (Tx_d) y la línea Transmisión Lista (Tx_{rdy}). Por Tx_d fluyen los caracteres que se transmitirán, y esta línea a su vez se conecta a la línea Rx_d del otro sistema, el sistema encargado de recibir los datos. Generalmente el UART 8250 puede

mantener un sólo caracter en el registro de transmisión durante la transmisión y recepción de datos, por eso el microprocesador debe enviar alguna señal cuando la transmisión del caracter se ha completado. Para este propósito existe la línea Tx_{rdy}. Cuando la transmisión de un caracter ha terminado, activa la línea y ésta devuelve la señal al microprocesador como si se tratara de una interrupción, así rápidamente es atendida y se envía un nuevo caracter a ser transmitido.

La sección del receptor es similar a la del transmisor. Sin embargo la línea de recepción de datos Rx_d, como su nombre lo indica permite la entrada de los datos que se están enviando desde el transmisor del sistema con el que se comunica. La línea de salida Rx_{rdy} es utilizada como una interrupción del microprocesador, porque la línea avisa al microprocesador que un caracter se recibirá.

INTERFASE DEL PROTOCOLO

Con las señales que confonman esta interfase **RTS** (Request to send), **DSR** (Data Set Ready), **DTR** (Data Terminal Ready) y **CTS** (Clear to Send) pueden implementarse diferentes protocolos de comunicación asíncrona. Un protocolo se define cuando dos sistemas que se comunican envían señales que controlan dicha comunicación, de tal forma que siempre saben cuando están listos para transmitir o recibir.

PROGRAMACION DEL UART 8250

Para la inicialización y configuración del UART 8250 es necesario conocer cada uno de sus registros de control. Para accesar cada uno de ellos, se realizan escrituras o lecturas a los puertos de entrada/salida de la computadora, a los cuales se encuentran conectados en COM1 y el COM2. Las direcciones de los registros del COM1 son: de la 3F8H a la 3FFH. Y las direcciones, para el COM2 son: de la 2F8H a la 2FFH.

La tabla 2.2.1 describe individualmente cada uno de los registros.

NOMBRE	ABREVIATURA	DIRECCION	E/S
Registro de Mantenimiento de Transmisión	THR	xF8H	S
Registro Buffer de Recepción	RBR	xF8H	E
Latch del divisor (DLL)	DLL	xF8H	S
Latch del divisor (DLM)	DLM	xF9H	S
R. de habilitación de interrupción	IER	xF9H	S
Registro de control de línea	LCR	xFBH	S
Registro de la línea de estado	LSR	xFDH	E

Tabla 2.2.1 Registros del UART 8250

REGISTRO DE MANTENIMIENTO DE TRANSMISION (THR)

Este registro contiene el siguiente caracter a ser transmitido, el primer bit del caracter que se envía es el bit 0. Para utilizar este registro el bit 7 del Registro de control de línea (LCR) debe encontrarse en 0.

REGISTRO BUFFER DE RECEPCION (RBR)

Este registro contiene el caracter que se acaba de recibir. Como la comunicación es serial, el caracter se transmite bit a bit llegando primero el bit menos significativo esto es el bit 0. También se necesita que el bit 7 del Registro de control de línea (LCR) este en 0.

LATCH DEL DIVISOR (DLL)

El Latch del divisor (DLL) es el byte menos significativo del divisor que se usa para calcular la velocidad de transmisión (Bauds). La dirección de memoria en que se localiza es la xF8 y es un registro de lectura/escritura. Para el uso de este registro, el bit 7 (DLAB) del registro de control de línea debe estar en 1.

LATCH DEL DIVISOR (DLM)

En este registro se encuentran los 8 bits más significativos del divisor para calcular la velocidad de transmisión. Y complementa al DLL. Para obtener el divisor dependiendo de la velocidad de transmisión se utiliza la siguiente fórmula:

$$\text{Divisor} = \frac{115\,200}{\text{Velocidad de Transmisión}}$$

Para el uso de este registro también el bit 7 del Registro de control de línea (LCR) debe estar en 1. La tabla 2.2.2 proporciona los valores de los registros que contienen el byte menos y mas significativos del divisor, de acuerdo a la velocidad en Bauds que se utilizará.

Bauds	DLM	DLL
50	08H	00H
75	06H	00H
110	04H	17H
134.5	03H	59H
150	03H	00H
300	01H	80H
600	00H	C0H
1200	00H	60H
1800	00H	40H
2000	00H	3AH
2400	00H	30H
3600	00H	20H
4800	00H	18H
7200	00H	10H
9600	00H	0CH

Tabla 2.2.2 Velocidad de transmisión.

REGISTRO HABILITACION DE INTERRUPCION (IER)

El PIC 8259 (Dispositivo controlador de interrupciones) permite cuatro tipos de interrupciones que se habilitan y deshabilitan independientemente, lo que permite que más de una interrupción sea habilitada a la vez. Los diferentes tipos de interrupciones se indican en los 4 bits menos significativos de este registro.(Tabla 2.2.3)

Dependiendo de la interrupción se fija el correspondiente bit en 1 para activarla o bien en 0 para desactivarla, logrando así que cuando se presente alguno de los estados que se pretenden se genere una interrupción. Por ejemplo: si el bit 0 está en 1, significa que cada vez que se reciba un dato en el registro de recepción se generará una interrupción.

Este registro comparte la dirección del Latch del divisor (DLM) (xF9), pero el Latch del divisor (DLM) necesita el DLAB del Registro de control de línea (LCR) en 1, para el uso del registro de habilitación de interrupción, el bit 7 del LCR permanece en 0

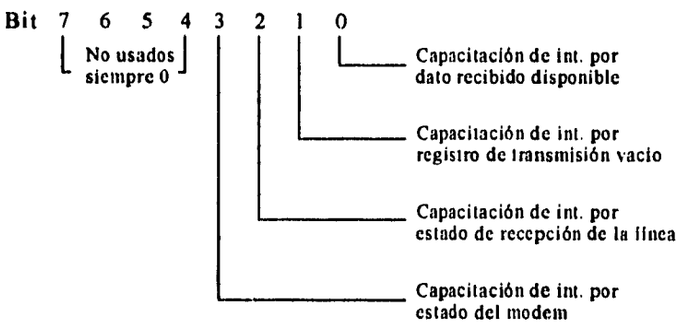


Tabla 2.2.3 Registro de habilitación de interrupción

REGISTRO DE CONTROL DE LÍNEA (LCR)

Como ya se ha mencionado el bit más significativo de éste registro debe mantenerse en 1 cuando se fija la velocidad de transmisión en el divisor. En el resto de los bits del registro se define y almacena lo relacionado al protocolo de comunicación. La función de cada bit se muestra en la tabla 2.2.4.

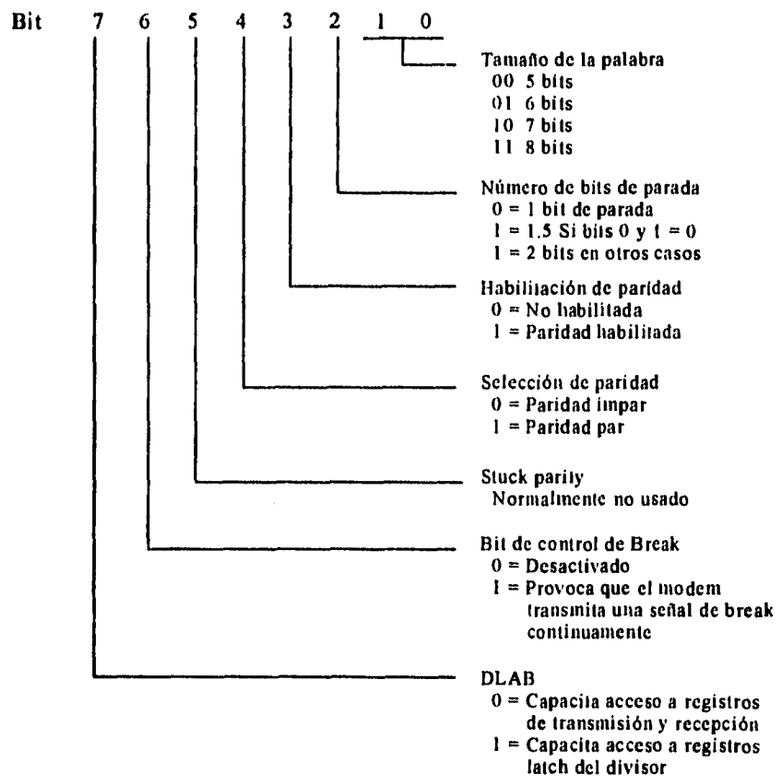


Tabla 2.2.4 Registro de control de línea

REGISTRO DE ESTADO DE LA LÍNEA (LSR)

El registro de estado proporciona información referente a los resultados de la transferencia de datos y de las condiciones de error que le corresponden.(Tabla 2.2.5)

Datos Listos (Data Ready). Cuando tiene 1 como valor indica que se ha recibido un caracter y que está contenido en el registro buffer de recepción. Este bit es puesto en 0 cuando el dato se ha leído.

Error de sobreescritura (Overrun error). Se activa cuando el caracter del registro buffer de recepción no fue leído antes de recibir al siguiente caracter, lo que significa que el caracter anterior ha sido destruido. Después de leído el registro de estado de la línea el bit de error de sobreescritura se fija en 0.

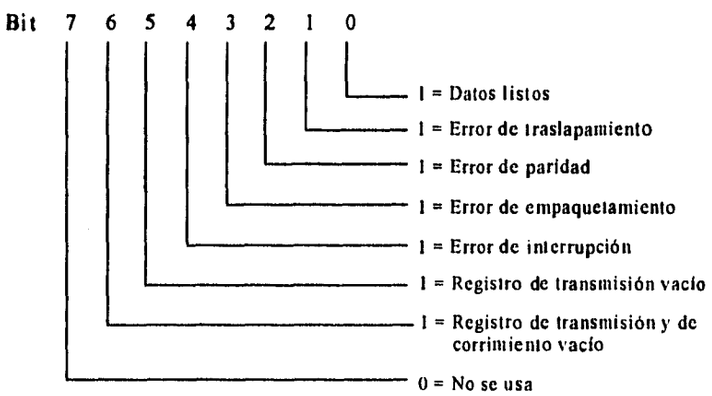


Tabla 2.2.5 Registro de estado de la línea

Error de paridad (Parity error). Este error se presenta cuando al verificar la paridad que contiene el caracter recibido ésta no coincide con la establecida en el LCR. Al leer el contenido del registro de estado este bit vuelve al valor de 0.

Error de construcción (Framming error). El error es delectado después de verificar el caracter que se ha recibido, se pone en 1 si registra que le falta un bit de parada válido. El bit se pone en 0 después de leer el registro de estado.

Interrupción del Break (Break interrupt error). Indica que se ha recibido una señal de break y no indica cuando finaliza la señal. Al igual que los bits anteriores fija el valor de 0 al leer este registro.

Registro de mantenimiento de transmisión vacío (Transmitter holding register empty). Cuando el UART 8250 está preparado para aceptar un nuevo caracter para transmisión este bit es encendido. Se apaga al cargar el registro de mantenimiento de transmisión.

Registro de desplazamiento de transmisión vacío. (Transmitter holding and shift register empty). Se enciende cuando el último caracter en el registro de desplazamiento de transmisión se ha transmitido. Se pone en 0 al transferir un dato del registro de mantenimiento al registro de desplazamiento de transmisión.

INTERFASE RS-232C

La interfase RS-232C (Reference Standard number 232 revision C) se introdujo al mercado en 1969 como un estándar para asegurar la compatibilidad entre equipos de diversos fabricantes. La C que aparece en la denominación de la interfase refiere a la cuarta versión, aprobada en 1981.

Las interfases de nivel físico, en general, se utilizan para conectar dispositivos de usuario al circuito de comunicaciones. Dado esto, la interfase RS-232C, es útil para la implementación de comunicación serial asíncrona en dispositivos como impresoras, modems, teclados y terminales.

En las especificaciones de la interfase, se describen cuatro funciones de la misma:

- Definición de las señales de control que atraviesan la interfase.
- Movimiento de los datos de usuario a través de la interfase
- Transmisión de las señales de tiempos necesarios para sincronizar flujo de datos.
- Conformación de las características eléctricas de la interfase.

La interfase RS-232 transmite los datos por medio de cambios en los niveles de voltaje, así un 0 binario se registra cuando los niveles de voltaje se encuentran comprendidos entre +3V y +12V, mientras que un 1 binario está representado por los niveles entre los -3V y -12V. La longitud del cable depende de las características eléctricas del mismo, aunque algunos fabricantes prohiben longitudes superiores a los 16 metros. Para cubrir sin problemas esta distancia, el transmisor debe proporcionar la corriente necesaria a la línea y así el receptor trabajará sin errores.

El estándar RS-232 es una interfase de hardware con salidas de 9 o de 25 pines. En la Figura 2.2.6 se lista cada uno de los 25 pines con su respectiva función. Para la interfase de 25 pines se utiliza un conector hembra que se conoce como **DB25** y para la de 9, al conector se le llama **DB9**. No existen diferencias de función entre ambos conectores, ya que una gran parte de los pines de DB25 no son utilizados. En la Tabla 2.2.6 se listan las señales utilizadas en una comunicación asíncrona de ambos conectores DB9 y DB25.

Para la implementación de una comunicación asíncrona es necesario el uso de dos señales de control localizadas en los pines 4 y 5 y estas son REQUEST TO SEND y CLEAR TO SEND.

Nombre de la señal	Conector DB9 No. de Pin	Conector DB25 No. de Pin
DCD (Data Carrier Detect)	1	8
RX (Receive Data)	2	3
TX (Transmit Data)	3	2
DTR (Data Terminal Ready)	4	20
GND (Signal Ground)	5	7
DSR (Data Set Ready)	6	6
RTS (Request to send)	7	4
CTS (Clear to send)	8	5
RI (Ring Indicator)	9	22

Tabla 2.2.6 Señales de la interfase RS-232

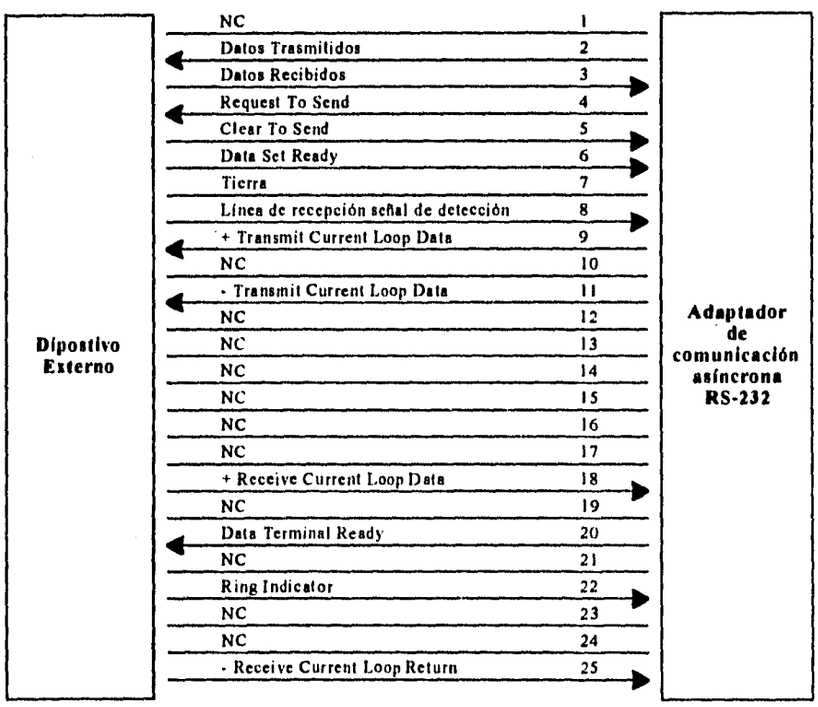


Fig. 2.2.6 Interfase RS-232

COMUNICACION SIMPLEX, HALF-DUPLEX, Y FULL DUPLEX

Aplicaciones de la vida real requieren de diversas formas de implementación para comunicar equipos. Por ejemplo una impresora conectada a una computadora solo necesita de información que fluye en un solo sentido (*unidireccionalmente*), es decir la impresora únicamente recibe datos de la computadora y ésta no necesita regresarle información, a este tipo de implementación se le denomina ***Comunicación Simplex***.

Otro tipo de dispositivos como terminales con su respectivo teclado envían y reciben datos del procesador esto puede llevarse realizarse por medio de una sola línea de comunicación, a esto se le conoce como ***Comunicación Half-Duplex***. Finalmente cuando es necesaria grandes velocidades o un muy buen funcionamiento durante la comunicación se utilizan dos líneas de comunicación separadas, una de transmisión y otra de recepción. A este tipo de implementación se le llama ***Comunicación Full-Duplex***. (Fig. 2.2.7)

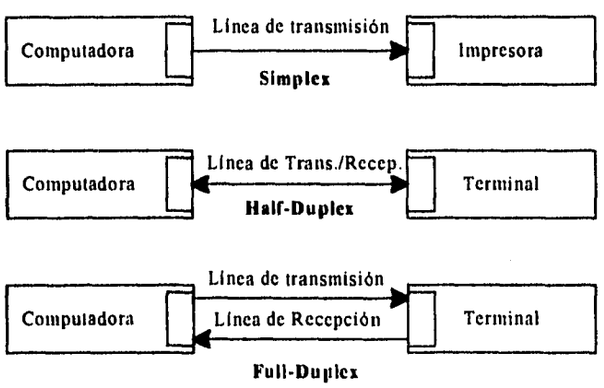


Fig. 2.2.7 Tipos de Comunicación.

2.3 INTERRUPCIONES

TIPOS DE INTERRUPCIONES

Las interrupciones proporcionan la ayuda necesaria para transformar el ambiente en el que se trabaja a otro totalmente diferente y después de un tiempo, regresar al anterior sin ningún problema, ni modificación. La transferencia del control inicia cuando se presenta un evento, ya sea interno del procesador, o externo del hardware. Por ejemplo cuando un dispositivo interno envía alguna señal indicando que necesita ser atendido, el procesador suspende lo que realiza en el momento y pasa el control a una rutina especial que ejecuta lo requerido por el dispositivo. A esta rutina especial se le llama ***rutina de atención de interrupción***. Una vez que se ha terminado la ejecución de la rutina de interrupción el control regresa al punto exacto en donde el procesador fue interrumpido. El procesador es capaz de implementar hasta 256 interrupciones. Las interrupciones se clasifican en 5 grupos: ***Interrupciones externas de hardware, interrupciones de software, interrupciones internas, la interrupción no mascarable y la interrupción de reset.***

El usuario define la función de las interrupciones de hardware, de software y las no mascarables, mientras que las internas y las de reset están dedicadas a funciones del sistema.

Por otro lado es muy importante especificar que además las interrupciones están clasificadas por la prioridad que tienen, esto significa que habrá interrupciones de mayor prioridad que serán atendidas de inmediato, mientras que las de menor prioridad tendrán que esperar a que una interrupción de mayor prioridad termine de ser atendida. Así las interrupciones de mayor prioridad son las internas, luego las no mascarables, las de software y las de menor prioridad son las interrupciones externas de hardware. Cada interrupción se identifica gracias a un número que puede encontrarse desde 0 hasta 255. El 0 corresponde a la interrupción de mayor prioridad y el 255 a la de menor.

Con esto puede decirse que una rutina de interrupción que se inicia puede ser interrumpida únicamente por otra de mayor prioridad, de lo contrario trabajará hasta terminar.

VECTOR DE INTERRUPCIONES

El vector de interrupciones es utilizado para almacenar los apuntadores de dirección que indican donde se localizan cada una de las rutinas de servicio de interrupción. La figura 2.3.1 muestra el mapa en memoria del vector de interrupciones, este mapa contiene 256 vectores o apuntadores de dirección. El apuntador de dirección corresponde a la primera instrucción de la rutina de interrupción de cada una de las 256 interrupciones. El vector de interrupciones inicia en la localidad de memoria 00000 y termina en la 003FEH, lo que representa el primer 1K bytes de la memoria RAM de la computadora.

Para representar los apuntadores de dirección de las rutinas de interrupción se necesitan 2 palabras (4 bytes) de memoria que se almacenan en direcciones pares. La palabra más significativa del apuntador corresponde al segmento de memoria en el que la rutina se encuentra, a esta palabra se le nombra dirección base. La palabra menos significativa corresponde al offset y ésta es la primera instrucción de la rutina que se localiza en el segmento definido por la dirección base.

INSTRUCCIONES PARA EL USO DE INTERRUPCIONES

Algunas de las instrucciones para el uso de interrupciones se listan en la tabla 2.3.1. Las primeras instrucciones de la tabla 2.3.1 son: **CLI** (deshabilita bandera de interrupción) y **STI** (Habilita bandera de interrupción) que permiten la manipulación de las banderas de interrupción (IF) a través de software. El STI (Set Interrupt) activa la entrada de interrupciones externas. Por otro lado la ejecución de CLI desactiva éstas mismas. La siguiente instrucción en la lista, es INT (Interrupción) usado para inicializar el vector

correspondiente que inicia la rutina de interrupción. Por ejemplo si se quiere ejecutar la interrupción 50, la instrucción se escribiría como sigue: INT 50H. Primero en el vector de interrupciones se localiza el vector 50, se lee el offset que se carga al IP (Apuntador de instrucciones) y el segmento cargado al CS (Code Segment), se calcula la dirección física y se leen las instrucciones de la memoria (ciclo de fetch).

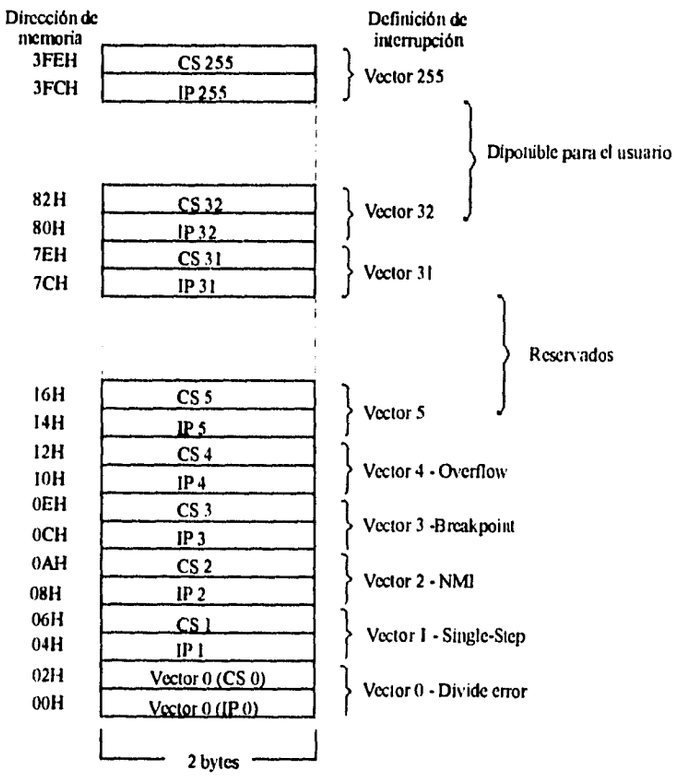


Fig. 2.3.1 Vector de interrupciones

Otra instrucción de las listadas en la tabla. Es la instrucción IRET (Retorno de rutina de atención de interrupción), esta instrucción se localiza al final de cada rutina de atención de interrupción. La instrucción IRET termina la ejecución de la rutina de interrupción y recupera los valores que almacenó en la pila al generarse la interrupción. Estos valores son el IP (Apuntador de instrucción), el CS (Code Segment) y las banderas (Desbordamiento (OF), Dirección (DF), Interrupción (IF), Signo (SF), Cero (ZF), Acarreo auxiliar (AF), Paridad (PF) y Acarreo (CF)). Una vez que se han recuperado todos los valores almacenados en la pila, se tiene todo lo necesario para que el IP (Apuntador de instrucciones) regrese al lugar en donde se propició la interrupción y se continúe con el programa.

Abreviatura afectadas	**Significado**	**Formato**	**Operación**	**Banderas**
CLI	Desactiva bandera de interrupción	CLI	0 → (IF)	IF
STI	Activa bandera de interrupción	STI	1 → (IF)	IF
INT n	Interrupción n de software	INT n	(Flags)-→((SP)-2) 0 → TF,IF (CS) → ((SP)-4) (2+4•n) → (CS) (IP) → ((SP)-6) (4 • n) → (CS)	TF,IF
IRET	retorno de interrupción	IRET	((SP)) → (IP) ((SP)+2) → (CS) ((SP)+4) → (Flags) (SP)+6 → (SP)	AH
INTO	Interrupción	INTO	INT 4	TF,IF
HLT	Paro (Halt)	HLT	Espera una int. externa.	Ninguna
WAIT	Espera	WAIT	Espera a la señal TEST para activarse.	Ninguna

Tabla 2.3.1 Instrucciones para el uso de interrupciones

INTERRUPCIONES EXTERNAS

El microprocesador 8088 posee dos líneas físicas, diseñadas para detectar una petición de interrupción. Las líneas son las siguientes: la línea INTR (Petición de interrupción) y NMI (interrupciones no mascarables). La línea INTR es manejada por el dispositivo controlador de interrupciones (PIC 8259). Los dispositivos externos que accesan al procesador por medio de interrupciones, primero son conectador al PIC 8259.

El término mascarable (según IBM) es utilizado para designar a las interrupciones externas, generadas en el controlador de interrupciones (PIC 8259). La habilidad que se tiene para inicializar una ***interrupción mascarable***, depende directamente de encender las banderas de interrupción (IF) o de apagarlas, cuando se enciende alguno de los bits correspondientes al registro de banderas se dice que se está haciendo una máscara en el momento de activar la interrupción. Por medio del software, esto puede realizarse usando la instrucción STI y CLI.

Las ***interrupciones no mascarables*** son las interrupciones que llegan al microprocesador a través de la línea NMI. Las interrupciones no mascarables son todas aquellas, que no se originan en el controlador de interrupciones y que no son afectadas por la bandera de interrupción del procesador.

CONTROLADOR DE INTERRUPCIONES 8259

Una computadora requiere de dispositivos de entrada y de salida como el teclado, el monitor, sensores y otros componentes para recibir atención por parte del sistema, lo cual podría llevarse gran parte del tiempo de procesamiento del sistema. El método más común para darle atención y servicio a estos dispositivos es el de "poleo", con el cual tenemos que estar verificando continuamente a cada uno de los dispositivos durante la ejecución del programa principal para determinar si requieren de alguna atención por parte del procesador central.

Este método es muy sencillo de implementar pero no es lo más eficiente, debido a que tenemos que preguntarle a cada dispositivo si requiere atención y detener la ejecución del programa principal cada vez que se haga una inspección a los dispositivos conectados, además el dispositivo que requiere de atención puede quedarse esperando una respuesta por más tiempo debido a que tiene que esperar a que el procesador central le pregunte si no requiere atención.

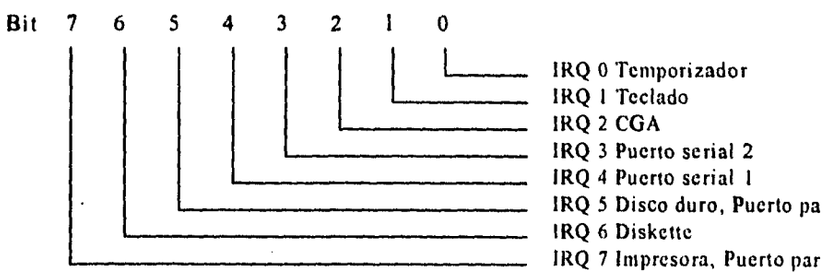
El método más comúnmente utilizado por su eficiencia para la atención de dispositivos periféricos es el de interrupciones. Este método permite al procesador estar ejecutando un programa y cuando algún dispositivo requiera de atención del procesador, es avisado para atender al dispositivo que requiere de su atención; esta señal que avisará al procesador que un dispositivo requiere de atención es llamada interrupción, la cual causará que el procesador deje de hacer en ese momento lo que estaba haciendo para atender al dispositivo.

Como el sistema puede tener conectados varios dispositivos a la vez, es necesario administrar las interrupciones que mandan cuando requieren atención del procesador, aquí es donde el controlador de interrupciones es importante.

El controlador de interrupciones puede manejar desde 8 hasta 64 dispositivos cuando varios controladores de interrupciones son conectados en cascada y el controlador de interrupciones principal se encuentra conectado a la línea INTR del 8086, cuando algún dispositivo requiere atención produce una interrupción que es recibida por el controlador de interrupciones y éste interrumpe al procesador indicándole cual fue el dispositivo responsable de la interrupción, esto es determinado dependiendo de como estén conectados los dispositivos periféricos al controlador de interrupciones.

En la tabla 2.3.2 se muestra la máscara del controlador de interrupciones que indica que dispositivos tiene conectado, con esta máscara se pueden habilitar o deshabilitar las interrupciones producidas por algunos dispositivos, esta máscara se encuentra en el puerto 21h por lo cual es posible accesarla mediante las instrucciones de ensamblador IN y OUT. En la siguiente tabla podemos ver a que corresponden cada bit del puerto 21h en las computadoras IBM y compatibles.

Linea de entrada	Código de Int.	Dirección de V.I.	Dispositivo externo
IRQ 0	08H	0020H	Temporizador
IRQ 1	09H	0024H	Teclado
IRQ 2	0AH	0028H	Adaptador Gráfico de Colores CGA
IRQ 3	0BH	002CH	Puerto Serial 2 y 4
IRQ 4	0CH	0030H	Puerto Serial 1 y 3
IRQ 5	0DH	0034H	Disco duro en XT Puerto paralelo 2 en AT
IRQ 6	0EH	0038H	Diskette
IRQ 7	0FH	003CH	Impresora puerto paralelo 1



Bit = 0 si la interrupción está habilitada
Bit = 1 si la interrupción está deshabilitada

Tabla 2.3.2 Máscara del controlador de interrupciones

Cuando el dispositivo ha terminado su operación debe de indicar al controlador de interrupciones que puede aceptar una nueva interrupción de cualquier otro dispositivo, por lo cual es necesario mandarle una señal conocida como fin de interrupción (EOI); esto se hace mandando el valor de 20H al puerto 20H, con esto el controlador de interrupciones libera la interrupción que estaba atendiendo y se prepara para recibir una nueva interrupción, la instrucción sería la siguiente:

```
MOV     AL,20H  ; código de Fin de Interrupción
OUT     20H,AL  ; Al puerto 20H
```

Este par de instrucciones se localizan generalmente al final de las Rutinas de Atención de Interrupciones (ISR).

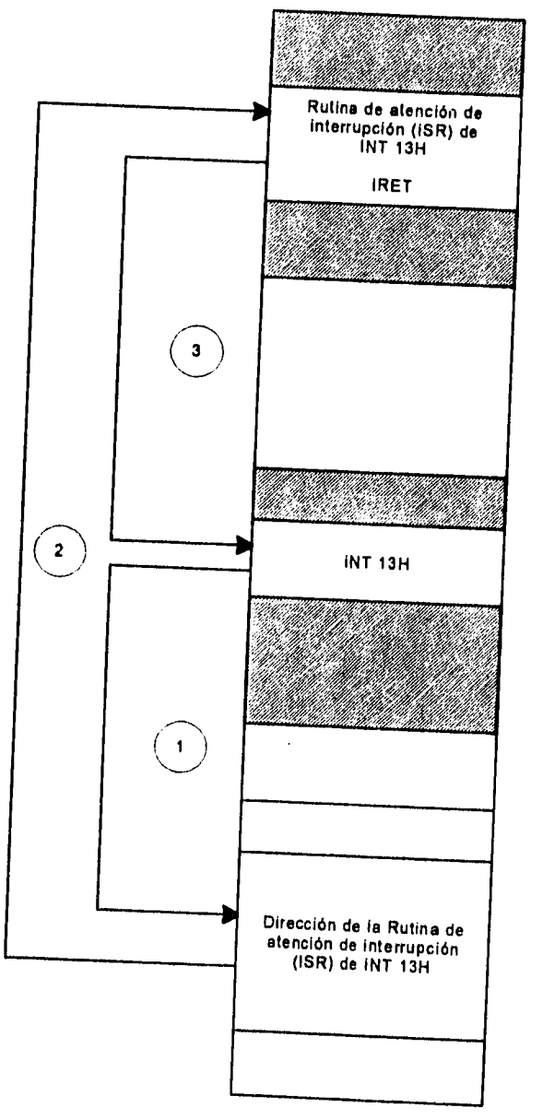
RUTINA DE ATENCIÓN DE INTERRUPCION

Cuando algún dispositivo interno o externo necesita la atención del CPU, manda un señal que es conocida como interrupción misma que es atendida por un controlador de interrupciones (Programable Interrupt Controller PIC). Al momento de generarse una interrupción el controlador de interrupciones se encarga de ignorar el acceso al procesador por interrupciones de menor prioridad. Esto, hasta que el procesador termine de atender la interrupción. Una vez que se le ha comunicado al procesador de la presencia de una interrupción, este termina de ejecutar la instrucción que este realizando en el momento de la interrupción, salva la banderas de estado del procesador y llama a una rutina que se va a encargar de atender a la interrupción. Esta rutina es llamada Rutina de Atención de Interrupción (ISR-Interrupt Service Routine). La dirección de esta rutina se encuentra almacenada en el Vector de Interrupciones localizado en la dirección 0000:0000 de la memoria. Debido a que la dirección de la rutina de atención de interrupción está compuesta por Offset y Segmento, ocupa 4 bytes de memoria, por lo tanto la dirección que debe leer el procesador para determinar la dirección de la rutina de atención de interrupción, es segmento 0000 y el offset se calcula multiplicando el número de la interrupción por cuatro,

(ej. INT 13h, 13h x 4h = 4Ch por lo tanto la dirección de la rutina de atención de interrupción de INT 13h en el Vector de Interrupciones es la 0000:004C); figura 2.3.2.

La rutina de atención de interrupción se ejecuta en el momento que se produzca la interrupción. La interrupción puede ser originada por el disco duro, el teclado, los discos blandos, el puerto serial, el puerto paralelo o cualquier dispositivo que en un momento dado requiera atención del procesador. Cuando se produce una interrupción el procesador debe dejar de hacer lo que estaba haciendo en ese momento, por lo tanto es necesario salvar el estado de registros y banderas, así como la dirección de retorno para que se sepa en donde había sido interrumpido y retomar su tarea anterior.

La rutina de atención de interrupción salva el estado de los registros en la pila y al finalizar la ejecución de la rutina de atención de interrupción retorna los registros a sus valores previos a la interrupción e indica al controlador de interrupciones que ha terminado la interrupción y que puede atender una nueva interrupción; esto se hace mandando un fin de interrupción (EOI) a los puertos de controlador de interrupciones. Finalmente la rutina de atención de interrupción se termina con una instrucción IRET, para indicarle al procesador que puede continuar con su trabajo.



1

La ejecución de la INT 13h hace que el microprocesador salve su estado actual y salte a la función que se encuentra en la dirección almacenada en el vector de interrupciones correspondiente a la interrupción número 13H

2

El microprocesador ejecuta el código que maneja la interrupción número 13H (en este caso es la encargada del manejo de los discos).

3

Cuando es ejecutada la intrucción IRET de la rutina de atención de interrupción, el control es regresado a la instrucción siguiente a la INT 13H del programa

Fig. 2.3.2 Secuencia de una interrupción

2.4 MANEJADORES DE DISPOSITIVOS (DEVICE DRIVERS)

Un manejador de dispositivo es la parte del sistema operativo responsable de controlar y establecer comunicación con el *hardware*. Esta parte representa el nivel más bajo del sistema operativo, permitiendo al resto de los niveles trabajar con independencia total del hardware. En otras palabras, un manejador de dispositivo mantiene una relación directa con el núcleo (*Kernel*) del sistema operativo y proporciona la información de los diversos periféricos que se encuentran conectados a la unidad central de proceso. La relación que un manejador de dispositivo mantiene con el núcleo del sistema operativo, es análoga a la que el sistema operativo mantiene con los programas de aplicación.

En un principio, los manejadores de dispositivos residían en el código del sistema operativo. Esto significaba una gran desventaja ya que era prácticamente imposible modificarlos e instalar nuevo hardware. Pero a partir de la versión 2.0 del sistema operativo (MS-DOS) los manejadores de dispositivos se convirtieron en código enteramente modificable proporcionando al usuario mayor flexibilidad, y la posibilidad de instalar todo el hardware a su disposición.

El sistema operativo (MS-DOS) y el manejador establecen su comunicación por medio de llamadas a funciones y estructuras de datos simples. Conocer los principios de la comunicación permite al programador de leguaje ensamblador, desarrollar un manejador de dispositivo para controlar cualquier dispositivo través del sistema operativo. El desarrollo de manejadores no puede realizarse en lenguajes de alto nivel.

Cuando un sistema es inicializado, los manejadores (drivers) NUL, CON, AUX, PRN, así como los manejadores del disco duro y de los discos flexibles son cargados e instalados en memoria. Una vez que se encuentran en memoria los manejadores se conectan unos con otros en una especie de cadena. Para indicarle al sistema operativo (MS-DOS) que se instalará un nuevo manejador, es necesario incluir en el archivo Config.sys una línea como la que sigue:

DEVICE = ANSI:SYS

Donde ansi.sys es el manejador a instalar.

Al inicializar el sistema, el contenido del archivo CONFIG.SYS es leído y evaluado. Si después de inicializar los manejadores antes mencionados, y que llamaremos manejadores del sistema operativo, existe una línea como la anterior entonces el manejador será incluido en la cadena de manejadores instalados. La cadena de manejadores se liga por medio de apuntadores, que el sistema operativo sigue para encontrar el manejador a instalar durante la inicialización. El nuevo manejador que se incorpora a la cadena es instalado inmediatamente después del manejador NUL.

Los manejadores que se instalan durante la inicialización, no son únicamente manejadores del sistema operativo. El usuario del sistema puede crear nuevos manejadores, ya sea que estos, sustituyan las operaciones que ejecutan los manejadores del sistema operativo, ó que realicen operaciones que cubran necesidades propias del usuario. Dado que, el primer manejador que se instala, después del NUL, es el manejador que el usuario define en la línea del config.sys. Si este manejador sustituye a uno de los del sistema operativo, cuando el sistema operativo recorre la cadena de manejadores, el primero que encuentra y ejecuta es el instalado por el usuario, y el antiguo manejador es ignorado. En la figura 2.4.1, se muestra como los manejadores se van ligando para formar la cadena de manejadores.

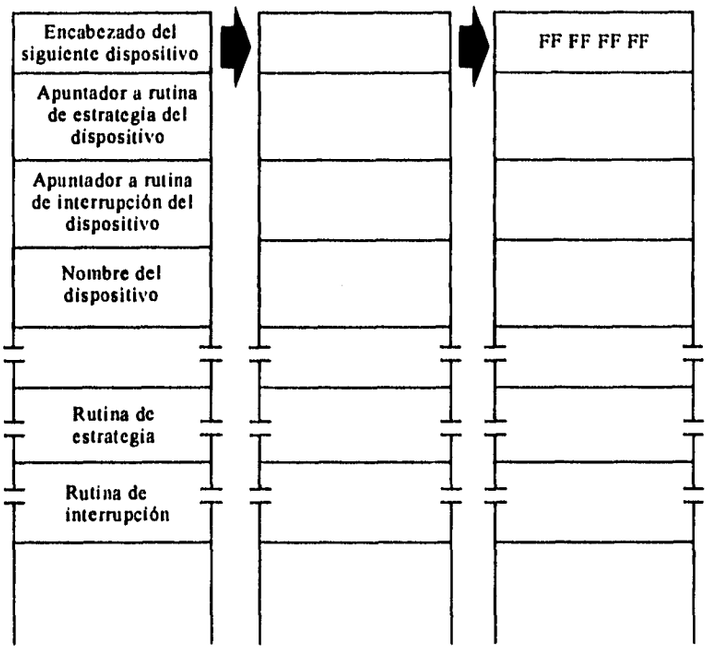


Fig. 2.4.1 Instalación de manejadores de dispositivos

No todos los manejadores del sistema operativo pueden ser remplazados por nuevos. El NUL es un ejemplo de ello. Este manejador siempre se encuentra a la cabeza de las cadena de manejadores. Si por alguna razón se coloca un nuevo manejador NUL, el sistema operativo lo ignora e instala el original. Lo mismo se presenta para los manejadores que controlan los discos flexibles y los discos duros. La razón para ello es que estos tienen especificadores de dispositivos y no nombres como el resto de los manejadores.

Los manejadores que controlan dispositivos de almacenamiento se dividen en dos tipos de manejadores de dispositivos:

- ***Tipo caracter***
- ***Tipo Bloque***

- ***Tipo Caracter***

Estos manejadores se comunican con dispositivos periféricos, tales como el teclado, el monitor, la impresora o el modem, y lo hacen enviando y recibiendo un byte a la vez. Cada manejador tipo caracter posee un nombre lógico que puede contener desde uno hasta 8 caracteres, este nombre es utilizado por un programa de aplicación para abrir el dispositivo que fue programado. Lo que significa que un manejador de dispositivo únicamente puede controlar un solo dispositivo.

- ***Tipo Bloque***

Usualmente controlan el acceso de información aleatoria y masiva de dispositivos tales como los discos duros, y los discos flexibles. Pero también permiten el acceso de dispositivos como cintas magnéticas. Estos manejadores transmiten bloques de información pero no caracter por caracter, incluso pueden transmitirse varios bloques cuando se ejecuta una llamada a una función. El tamaño de los bloques difiere dependiendo del tipo de dispositivo para el que fue diseñado el manejador.

Los manejadores tipo bloque no necesitan de nombres lógicos como los de tipo caracter, en lugar de esto, el sistema operativo les asigna una letra consecutiva (a, b,....z) de acuerdo al número de dispositivos que se encuentran conectados e instalados. Los manejadores tipo bloque consideran a cada dispositivo como una unidad, dentro de cada manejador. Un manejador tipo bloque tiene la capacidad de controlar al mismo tiempo varios dispositivos o unidades. Incluso puede dividir un dispositivo en varias unidades lógicas.

Por ejemplo, cuando un disco duro se divide en dos unidades lógicas, con los nombres C y D. Cada una de las unidades lógicas debe tener una FAT (File Allocation Table apéndice A) y un directorio raíz. Para que cada una de las unidades lógicas se manifiesten como discos duros independientes y con su propia organización. Y con el propósito de evitar errores.

Los manejadores de bloque siempre leen y escriben el número exacto de bloques definidos, a menos que un error ocurra.

Estructura de un manejador de dispositivos

Independientemente de las diferencias que existen entre los manejadores tipo caracter y los de bloque, ambos constan de la siguiente estructura:

- Cabecera (Device Header)
- Rutina de estrategia (Strategy Routine)
- Rutina de interrupción (Interrupt Routine)

- ***Cabecera (Device Header)***

Esta parte se localiza al inicio de cada manejador de dispositivo y contiene la información necesaria para que el DOS implemente el manejador.(tabla 2.4.1)

Los primeros dos campos son la liga con el siguiente manejador de la cadena (offset y segmento). Las localidades de memoria en los que estos campos se alojaran deben ser reservados por el programador, pero DOS los reserva cuando el manejador es instalado en el sistema. El siguiente campo son los atributos de las banderas para el dispositivo (tabla 2.4.2), los cuales especifican, entre otras cosas si el manejador es de tipo caracter o de bloque.

Finalmente los últimos campos corresponden a las direcciones (offset) de la rutina de estrategia y de interrupción, así como el número de dispositivos conectados al manejador (Bloque) o bien el nombre lógico del manejador cuando se trata de manejador tipo caracter.

Byte Offset	Descripción	Tamaño
00H	Dirección del offset del siguiente manejador	1 palabra
02H	Dirección del segmento del siguiente manejador	1 palabra
04H	Atributo	1 palabra
06H	Offset de la rutina de estrategia	1 palabra
08H	Offset de la rutina de interrupción	1 palabra
0AH	Nombre del *manejador* (tipo caracter) ó Número de dispositivos usados por el de bloque.	8 bytes

Tabla 2.4.1 Cabecera del manejador de dispositivo.

- ***Rutina de estrategia (Strategy routine)***

El sistema operativo hace uso de esta rutina en dos ocasiones, primero, al inicializar el manejador y después cada vez que el dispositivo realiza llamadas de entrada/salida. La dirección de la estructura de datos que contiene la información referente a la operación a ser ejecutada, conocida como encabezado de petición (Request Header), se obtiene cuando el sistema operativo (DOS) lo pasa a la rutina de estrategia por medio de los registros ES:BX. Dicha dirección es almacenada y el control regresa al sistema operativo inmediatamente, para permitir que la rutina de interrupción atienda la operación que se requiere. Así, el único trabajo que realiza la rutina de estrategia es salvar la dirección del encabezado de petición y no ejecuta ninguna operación de entrada/salida. Además debe tomarse en cuenta que dentro de la rutina de estrategia no puede utilizarse ninguna llamada a las funciones de la interrupción 21h.

Bit	Significado
15	= 1 Si es un manejador tipo caracter. = 0 Si es un manejador de bloque.
14	= 1 Si IOCTL, lectura y escritura esta activado.
13	Para manejadores tipo bloque = 1 Si el BPB (Bloque de parámetros del BIOS) será utilizado para determinar las características del medio. = 0 Si el byte identificador del medio será utilizado.
12	Para manejadores tipo caracter = 1 Si el manejador utiliza el comando Salida hasta que se encuentra ocupado.
11	= 1 Si el manejador permite el uso de los comandos Dispositivo abierto, dispositivo cerrado y medio removible.
7-10	*Reservado* (siempre en 0)
6	= 1 Si utiliza los comandos Obtención de dispositivo lógico y Configuración de dispositivo lógico.
5	*Reservado* (siempre en 0)
4	= 1 Si se implementa la interrupción 29h.
3	= 1 Si se usa el dispositivo actual del reloj.
2	= 1 Si se usa el dispositivo NUL actual.
1	Para manejadores tipo bloque = 1 Si el manejador direcciona sectores de 32 bits Para manejadores tipo caracter = 1 Si es un dispositivo estándar de salida.
0	= 1 Si es un dispositivo estándar de entrada.

Tabla 2.4.2 Palabra de atributos de la cabecera. Para los manejadores tipo bloque, únicamente los bits 1,6, 11, y del 13 al 15 tienen significado, los demás deberán estar en cero.

El Bloque de petición esta formado de al menos 13 bytes y estos forman un bloque de parámetros necesarios para ejecutar cualquier operación o función deseada. Después de ejecutar la operación, información adicional es agregada al Bloque de petición, de acuerdo al tipo de operación o función que se haya utilizado. El Bloque de petición es, entonces el enlace que existe entre el sistema operativo y el manejador, ya que a través de él, el sistema operativo (MS-DOS) se comunica, primero proporcionando al manejador los parámetros requeridos para la función y segundo, leyendo en el Bloque de petición la información que el manejador arroja después de su ejecución.

El componente más importante del Bloque de petición es el número de comando o de función que se necesita, pero no son menos importantes el número de sectores a leer o escribir, el estado etc. Tabla 2.4.3.

Byte Offset	Descripción	Tamaño
00H	Tamaño del bloque de datos en bytes	1 byte
01H	Número de dispositivo en comunicación	1 byte
02H	Número de comando	1 byte
03H	Palabra de Estado	1 palabra
05H	Reservado	8 bytes
0DH	Descripción del medio	1 byte
0EH	Dirección del Buffer (Offset)	1 palabra
10H	Dirección del Buffer (Segmento)	1 palabra
12H	Número de sectores	1 palabra
14H	Sector inicial	1 palabra

Tabla 2.4.3 Estructura del encabezado de petición.

- ***Rutina de interrupción***

Esta es la última y más compleja parte del manejador. Es llamada, por el sistema operativo (MS-DOS), inmediatamente después de que la rutina de estrategia ha concluido y es propiamente la que permite que el manejador funcione. Cuando una función de entrada/salida ha sido completada, la rutina de interrupción utiliza el campo de estado en el Bloque de petición para informar al núcleo (Kernel) del sistema operativo como ha terminado la operación o función ejecutada, además informa sobre el número de sectores transferidos, entre otras.

La rutina de interrupción realiza las operaciones siguientes:

1. Cubrir las necesidades de las llamadas del sistema operativo (Comandos).
2. Salvar todos los registros del procesador en la pila que resultan afectados y modificados durante la ejecución de cada rutina de comando. Extraer

del Bloque de petición el código de la función o comando deseado e incluir una tabla que permita ejecutar la rutina correspondiente al código leído.

3. Almacenar el código de estado y del error en el Bloque de petición y restablecer el valor de los registros.

Campo de Estado

Cada vez que la rutina de un comando ha terminado, en este campo se especifica si ha ocurrido un error o no. Pero independientemente de los errores detectados, el bit 8 de la palabra de estado debe activarse, después de ejecutar cualquier comando, incluso cuando se trata de un comando de no-ejecución (DUMMY), es decir cuando la rutina de ese comando no realiza ninguna actividad.

Bit(s)	Descripción
15	Error
12-14	Reservado
9	Ocupado (Busy)
8	Terminado(Done)
0-7	Código del error

Tabla 2.4.4 Valores que se obtienen de la palabra de estado del Bloque de petición.

COMANDOS

Un manejador de dispositivo puede hacer uso de un total de 20 rutinas de comandos diferentes. Dentro de los comandos posibles existen algunos que son utilizados para manejador de dispositivos de tipo caracter, de bloque o para ambos. En la tabla 2.4.6 se listan todos los comandos y su función así como el tipo de manejadores que los utilizan. Posteriormente se describirá cada uno de los comandos que se usan únicamente en manejadores tipo bloque.

Código	Descripción
00H	Violación de protección de escritura
01H	Unidad desconocida
02H	Dispositivo no listo
03H	Comando desconocido
04H	Error de datos (CRC)
05H	Longitud no válida
06H	Error de búsqueda
07H	Medio desconocido
08H	Sector no encontrado
09H	Impresora sin papel
0AH	Error de escritura
0BH	Error de lectura
0CH	Falla general
0DH-0EH	Reservado
0FH	Cambio de medio ilegal

Tabla 2.4.5 Código de errores que se localizan en los bits 0-7 de la palabra de estado del Bloque de petición.

Comando 0 Inicialización

Este comando es llamado durante el proceso de arranque del sistema para inicializar el manejador de dispositivo a instalar. En él pueden involucrarse requerimientos de hardware que se necesiten, fijar variables internas así como, los vectores de interrupciones. Dado que el sistema operativo no se encuentra totalmente inicializado cuando este comando se ejecuta, no se pueden utilizar todas las funciones de la interrupción 21h, únicamente las funciones de la 01 a la 0CH y la 30H.

La rutina de inicialización debe comprender diversas actividades, que proporcionarán tanto al sistema operativo(MS-DOS) como a las otras rutinas información necesaria para que el manejador se ejecute correctamente (Tablas 2.4.8 y 2.4.9).

Entre esas actividades encontramos las siguientes:

- Obtener el nombre o el valor del manejador. Si se trata de un manejador tipo bloque transformar el valor obtenido en su respectiva letra (0=A, 1=B, 2=C, etc.).

Número de comando	Función	Caracter	Bloque
0	Inicialización	X	X
1	Verificación del medio		X
2	Construcción del BPB		X
3	Lectura IOCTL	X	X
4	Lectura	X	X
5	Lectura no destructiva	X	
6	Estado de entrada	X	
7	Limpieza de buffers de entrada	X	
8	Escritura	X	X
9	Escritura con verificación		X
10	Estado de salida	X	
11	Limpieza de buffers de salida	X	
12	Escritura IOCTL	X	X
13	Dispositivo abierto	X	X
14	Dispositivo cerrado	X	X
15	Medio removible		X
16	Salida hasta que se encuentre ocupado	X	
19	Género IOCTL	X	X
23	Obtención de dispositivo lógico		X
24	Configuración de dispositivo lógico		X

Tabla 2.4.6 Comandos

- Indicar al sistema operativo si se ha ejecutado correctamente el comando, es decir el estado en el terminó el comando.
- Proporcionar, para manejadores tipo bloque, al sistema operativo el número de dispositivos lógicos que soporta. A pesar de que esta información se encuentra en la cabecera del manejador, el sistema operativo no inspecciona en él y por ello es necesario especificárselo en esta rutina.

- Dirigir de la parte más alta de la memoria que ocupa y usa el manejador (Break address), para que a partir de ahí el sistema operativo pueda instalar otro posible manejador.
- Pasar, si el manejador es tipo bloque, al sistema operativo la dirección del arreglo que contiene las direcciones del Bloque de parámetros del BIOS (BPB) de cada dispositivo lógico. La dirección del Bloque de parámetros del BIOS se encuentra en dos palabras, la primera es el offset, y la segunda el segmento. Cuando el manejador se compone de más de un dispositivo, las primeras dos palabras corresponden al primer dispositivo, las segundas al segundo dispositivo y así hasta el último dispositivo instalado.

Byte(s)	Contenido
00-01H	Bytes por sector
02H	Sectores por unidad
03H-04H	Número de sectores reservados (empezando en sector 0)
05H	Número de FAT (Fat Allocation Table)
06H-07H	Máximo número de entradas en directorio raiz
08H-09H	Número de sectores total en el medio
0AH	Byte del medio
0BH-0CH	Número de sectores ocupado por una sola FAT
0DH-0EH	Sectores por pista (versiones 3.0 en adelante)
0FH-10H	Número de cabezas(versiones 3.0 en adelante)
11H-12H	Número de sectores ocultos (versiones 3.0 en adelante)
13H-14H	Palabra de número de sectores ocultos
15H-18H	Si los bytes 8 y 9 son cero, Número total de sectores en el medio(versión 4)
19H-1EH	Reservado, debe estar en cero (versión 4)

Tabla 2.4.7 Estructura del Bloque de parámetros del BIOS (BPB).

El Bloque de parámetros del BIOS es un bloque de datos que contiene información que describe al dispositivo lógico. Durante el proceso de inicialización el sistema operativo busca en el Bloque de parámetros del BIOS ciertos datos, como por ejemplo el sector más

grande que existe en el dispositivo, para poder definir el tamaño del buffer que usará (véase apéndice A).

El código de la rutina de inicialización usualmente se localiza al final del manejador, con la idea de que después de utilizarse, la memoria que ocupa, pueda liberarse y sea utilizada nuevamente.

Para llamar al comando de inicialización, el sistema operativo lo hace por medio del Bloque de petición mismo que proporciona los siguientes parámetros:

Bloque de petición offset	Tamaño	Función
2	Byte	Número de comando = 0
18	Dword	Apunta al primer caracter después del igual en la línea del Config.sys
22	Byte	Número de dispositivo (0=A, 1=B, etc)

Tabla 2.4.8

Los parámetros que el manejador tiene que proporcionar al sistema operativo:

Bloque de petición offset	Tamaño	Función
3	Word	Estado
13	Byte	Número de unidades (únicamente tipo bloque)
14	Dword	Dirección de memoria libre (Break Address)
18	Dword	Apuntador al arreglo de BPB(únicamente tipo bloque)

Tabla 2.4.9

Comando 01 Verificación del Medio (Media Check)

Este comando sólo se usa con manejadores tipo bloque y para determinar si el medio o disco ha sido removido. Generalmente esta función es usada cuando se examina el directorio de un disco. El sistema operativo pasa a la rutina del comando el byte con la información que identifica el disco correspondiente al manejador instalado. La rutina proporciona el código que indica si el disco ha cambiado desde la última transferencia o acceso.

Código	Significado
0F0H	3.5",2 lados, 18 sectores
0F8H	Disco Duro
0F9H	3.5", 2 lados, 9 sectores
0F9H	5.25", 2 lados,15 sectores
0FCH	5.25", 1 lado, 9 sectores
0FDH	5.25", 2 lado, 9 sectores
0FEH	5.25", 1 lado, 8 sectores
0FFH	5.25", 2 lado, 8 sectores

Tabla 2.4.10 Estructura del Bloque de parámetros del BIOS (BPB).

Cuando el disco nunca cambia, como en el caso de un disco duro, se le indica a la rutina, para evitar que lea la FAT del disco cada vez que el sistema operativo realiza un acceso. Esto permite al sistema, un mejor y más rápido desempeño.

El sistema interpreta el resultado que el comando arroja de la siguiente manera:

- Cuando el disco no ha cambiado, el sistema procede al acceso del disco.
- Si el disco ha cambiado, el sistema operativo invalida todos los buffers asociados a la unidad, hace una llamada a la rutina de construcción del Bloque de parámetros del BIOS, lee la FAT y el directorio del disco.
- En el caso de que el estado de la función sea desconocido, entonces el sistema operativo procederá de acuerdo al contenido de los buffers internos. Si estos

contienen información el sistema operativo asume que el disco no ha sido cambiado y procede a escribir en el disco, corriendo el riesgo de destruir la estructura del disco si este fue cambiado. Pero si los buffers están vacíos, el sistema supone que el disco ha cambiado y ejecuta lo correspondiente a esta acción.

Para llamar al comando:

Bloque de petición offset	Tamaño	Función
1	Byte	Código de la unidad
2	Byte	Número de comando = 1
13	Byte	Byte que describe al medio

Tabla 2.4.11

Los parámetros que el manejador tiene que proporcionar al sistema operativo:

Bloque de petición offset	Tamaño	Función
3	Word	Estado
14	Byte	Código de cambio de medio -1 Si el disco ha cambiado 0 Si no se conoce el estado del disco 1 Si el disco no ha cambiado
15	Dword	Apuntador a la etiqueta del volumen previo. Bit 11 de la Cabecera del manejador activado.

Tabla 2.4.12

Comando 02 Construcción del Bloque de parámetros del BIOS (BPB)

Este comando se usa únicamente en manejadores tipo bloque. El sistema operativo llama al comando cuando en el comando de verificación del medio se indica que se ha

cambiado el disco. Si el manejador tiene que atender a un comando como éste, primero lee el sector reservado ó de carga (Boot) del disco para accesar al Bloque de parámetros del BIOS, y devolverle al sistema operativo, su dirección. El sistema operativo utiliza el Bloque de parámetros del BIOS para calcular donde se encuentran la FAT y el directorio del disco.

Para llamar al comando:

Bloque de petición offset	Tamaño	Función
1	Byte	Número de dispositivo
2	Byte	Número de comando = 2
3	Byte	Byte que describe al medio
14	Dword	Dirección del Buffer que contiene la FAT

Tabla 2.4.13

Los parámetros que el manejador tiene que proporcionar al sistema operativo

Bloque de petición offset	Tamaño	Función
3	Word	Estado
18	DWord	Dirección del BPB

Tabla 2.4.14

Comando 04 Lectura

DOS solicita este comando cuando necesita leer datos del disco. El sistema operativo pasará al manejador de dispositivos el número de sectores a leer, el sector inicial, y la dirección de memoria en donde los datos serán colocados. Antes de hacer la llamada a esta rutina el sistema operativo calcula los sectores que necesita utilizando la FAT y el directorio del disco.

Si durante la lectura se detecta un error, la rutina del comando debe indicarlo por medio de la palabra de estado. Pero también reporta el número de sectores que fueron transferidos con éxito, antes de que el error se presentara.

Cuando el dispositivo, para el que se construye un manejador, es un disco duro, el primer sector puede ser un número desde el cero hasta el máximo número de sectores del disco. El máximo número de sectores corresponde al inicio de la partición del disco. Por otro lado para los discos flexibles el sector inicial equivale siempre al sector reservado o de carga (Boot).

Finalmente es importante señalar que el sector inicial que el sistema operativo proporciona al manejador, tendrá que ser traducido por el manejador. De esta traducción obtiene: número de cabeza, la pista y el sector de la unidad física que se está utilizando.

Para llamar al comando:

Bloque de petición offset	Tamaño	Función
1	Byte	Número de dispositivo
2	Byte	Número de comando = 4
13	Byte	Byte que describe al medio
14	DWord	Dirección del Buffer en la que los datos serán leidos
18	Word	Número de sectores a ser leidos
20	Word	Primer sector a ser leído o contiene un -1 si el número de sector es de 32 bits
26	DWord	Primer sector (32 bits)

Tabla 2.4.15

Los parámetros que el manejador tiene que proporcionar al sistema operativo:

Bloque de petición offset	Tamaño	Función
3	Word	Estado
18	Word	Número de sectores transferidos
22	DWord	Apuntador a la etiqueta del volumen, si se presenta el error 0FH

Tabla 2.4.16

Comando 08 Escritura

Este comando indica al manejador que se escribirán uno o más sectores en el disco. La rutina transfiere los datos del buffer en memoria donde se encuentran al dispositivo al que se escribirán. De la misma forma que en el comando de lectura el sistema operativo pasa al manejador el sector inicial, el número de sectores a escribir y la dirección del buffer que contiene los datos a escribir.

Al igual que en la rutina de lectura, en este comando de escritura, el manejador tendrá que traducir la dirección lógica del sector a la dirección física del dispositivo. Y tendrá que reportarse el número de sectores transferidos con éxito cuando se presenta un error.

Los parámetros que se requieren para llamar al comando son prácticamente los mismos que para la lectura (tabla 2.4.13), únicamente hay que tomar en cuenta que el número de comando es el 08, y que el primer sector, el número de sectores y el buffer son para la escritura.

Comando 0FH Medio Removible

El comando indica si el medio puede cambiar o no. El comando sólo puede usarse cuando el bit 11 de la palabra de atributo de la cabecera del manejador se encuentra en 1.

Para llamar al comando:

Bloque de petición offset	Tamaño	Función
1	Byte	Número de dispositivo
2	Byte	Número de comando = 15 (0FH)

Tabla 2.4.17

Los parámetros que el manejador tiene que proporcionar al sistema operativo:

Bloque de petición offset	Tamaño	Función
3	Word	Estado

Tabla 2.4.18

ESTRUCTURA DEL MANEJADOR

Los manejadores deben seguir siempre la estructura de un archivo .COM. Ya que si se siguiera el formato de un archivo .EXE, no sería posible instalarse como se hace, porque los archivos que permiten cargar archivos .exe, aún no se instalan cuando un manejador debe instalarse. El manejador no debe declarar el segmento de una pila y seguir las restricciones para los archivos .COM. Debido a que los manejadores se pueden instalar en cualquier parte de la memoria, debe tenerse especial cuidado al organizar la ubicación del manejador en la memoria física.

Después de describir cada una de las partes que forman a un manejador, podemos implementar uno que cubra las características que deseamos. Para resumir, a continuación se

presenta el código en ensamblador que generalmente se encuentra en un manejador. Aumentándole el código correspondiente a cada comando, el manejador es funcional.

```
;*************************************************************************
;       Driver.asm
;*************************************************************************
        code segment
                assume cs:code, ds:code, es:code
                org 0

;*******Constantes*********************************************************
Maxcom equ  16      ;Máximo número de comandos
;*************************************************************************

;*******Cabecera del manejador*********************************************
        dw -1,-1            ;Conexión con siguiente manejador
        dw 8000h            ;Palabra de atributo
        dw estrategia       ;Offset de la rutina de estrategia
        dw interrupción     ;Offset de la rutina de interrupción
        db nombre           ;Nombre del manejador

;*************************************************************************

;******* Almacenar dirección del encabezado de petición ********************
        db_ptr dd     ?     ;Apuntador para la dirección del encabezado de petición
;*************************************************************************
```

```
;******* Tabla de comandos ************************************************
        Tab_cmd     dw      Inicialización              ;Comando 0
                    dw      Verificación del medio      ;Comando 1
                    dw      Construcción BPB            ;Comando 2
                    dw      Lectura directa             ;Comando 3
                    dw      Lectura                     ;Comando 4
                    dw      Lectura no destructiva      ;Comando 5
                    dw      Estado de entrada           ;Comando 6
                    dw      Limpia buffers de entrada   ;Comando 7
                    dw      Escritura                   ;Comando 8
                    dw      Escritura con verificación  ;Comando 9
                    dw      Estado de salida            ;Comando 10
                    dw      Limpia de buffers de salida ;Comando 11
                    dw      Escritura directa           ;Comando 12
                    dw      Dispositivo abierto         ;Comando 13
                    dw      Dispositivo cerrado         ;Comando 14
                    dw      Medio removible             ;Comando 15
                    dw      Salida hasta ocupado        ;Comando 16
;***************************************************************************

;********* Rutina de Estrategia*********************************************
;Rutina que es llamada por el MS-Dos. Salva el apuntador al encabezado de petición
;***************************************************************************
strat proc far
        mov word ptr cs:[db_ptr],bx
        mov word ptr cs:[db_ptr+2],es
ret
```

```
;******** Rutina de interrupción***********************************************
;Rutina que es llamada por el MS-Dos inmediatamente después de llamar la rutina de
estrategia.
;******************************************************************************
intr proc far
        push ax             ;Salva los registros
        push bx
        push cx
        push dx
        push ds
        push es
        push di
        push si
        push bp
        push cs             ; El data segment toma el valor del code segment
        pop ds

        les di,[db_ptr]     ; Carga en es:di el encabezado de petición
        mov bl,es:[di+2]    ; Verifica si el existe el comando que solicita
        xor bh,bh
        cmp bx,Maxcom
        jle int1            ;Salta si el código del comando es correcto
        call error          ;Se llama la rutina de error cuando el código del comando
                            ;no existe.
        jmp int2
int1:   shl bx,1            ;Calcula el valor dentro de la tabla de comandos
        call word ptr[bx+Tab_cmd]       ;Salta a la rutina
```

```
        les di,[db_ptr]             ;Dirección del encabezado de petición
int2:   or ax,0100h                 ;activa el bit de done de la palabra de estado
        mov es:[di+3],ax            ;almacena el valor de la palabra en el request
                                    ;header

        pop bp                      ;Recupera registros
        pop si
        pop di
        pop es
        pop ds
        pop dx
        pop cx
        pop bx
        pop ax
ret                                 ;Regresa al DOS.
;*************************************************************************
*
;******** Rutinas de los Comandos ***********************************************
;       Las rutinas, a continuación descritas, únicamente contienen el código que la indica
;       a la palabra de estado que se realizó con éxito. Para que un manejador sea funcional,
;       cada comando deberá tener código extra, que realice las tareas necesarias.
;*************************************************************************
*
MediaChk proc near                  ;Comando 1= Verificación del medio
        xor ax,ax
        ret
MediaChk endp
```

```
OutBusy proc near                ;Comando 16= Salida hasta ocupado
        xor ax,ax
        ret
OutBusy endp
Error proc near                  ;Rutina que fija en la palabra de estado un
        mov ax,8003              ;error,  se activa bit de error y de comando
                                 ;desconocido
        ret
Error endp

;*************************************************************************
;El comando de inicialización se realiza únicamente una vez durante la ejecución del
;manejador por lo que se coloca a esta altura del manejador, para que una vez que se ha
;ejecutado, el espacio en memoria que ocupa, pueda utilizarse nuevamente.
;*************************************************************************
Init proc near                   ;Comando 0=Inicialización
        xor ax,ax
        ret
Init endp
code ends
     end
```

3. DESARROLLO DE LA RED

En este capítulo, trataremos en forma, el desarrollo propio de la red. Primeramente abordaremos de que manera están conectadas fisicamente las computadoras y que tipo de conectores podemos usar así como los pines de estos que debemos conectar para un buen funcionamiento. Esto es importante, debido a que cada computadora puede tener diferentes tipos de conectores (DB 9 y DB 25), por lo tanto es necesario conocer la configuración de cada uno de ellos. Para poder establecer la comunicación entre computadoras, se desarrolló un protocolo de comunicaciones, podemos decir que este, fue desarrollado tomando como base el funcionamiento teórico de un protocolo de paso de estafeta, cabe mencionar que no son iguales, pero realizan las mismas funciones de operación, dada la implementación del software. Se desglosará la Rutina de Atención de Interrupciones y el Manejador de Dispositivos en todos los procedimientos que los componen para un mejor entendimiento, así como su mutua interacción. Para esta tarea contamos con diagramas de flujo que nos permiten visualizar de una manera gráfica el seguimiento de cada una de las etapas de los programas.

3.1 CONEXIONES

Las computadoras que integran la red están conectadas por medio de los puertos seriales, figura 3.1.1. Cada computadora debe contar con dos puertos seriales, los puertos que utilizamos durante la implementación de la red son los puertos COM1 y COM2, que corresponden a las direcciones 3F8H y 2F8H respectivamente.

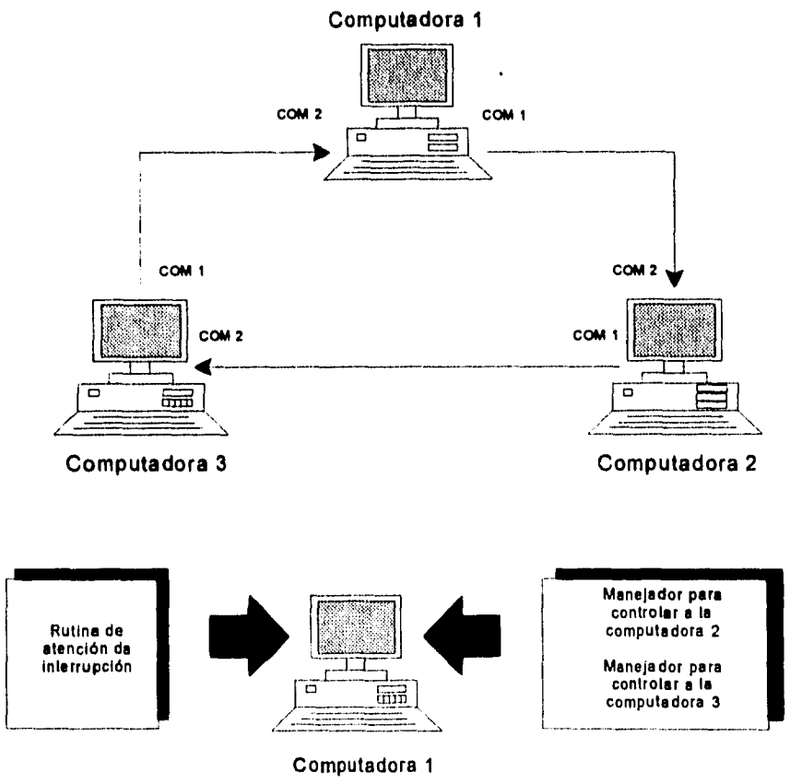


Fig. 3.1.1 Conexión de las computadoras

En la figura 3.1.1 se tienen conectadas tres computadoras y observamos que cada una de ellas tiene instalado una rutina de atención de interrupción y el número de manejadores respectivo. En este caso el número de manejadores en cada computadora es de dos, esto se debe a que cada computadora tendrá que controlar remotamente a dos computadoras más.

Cada puerto serial tiene una función bien definida, no se puede alterar la dirección del flujo de datos dentro de la red, ya que implicaría cambiar las rutina de atención de interrupciones de cada puerto. La información de petición de acceso de la computadora origen siempre viajará desde el puerto COM1 al COM2, mientras que la información que viaja en respuesta a la petición de acceso viaja desde el COM2 al COM1.

Para establecer la comunicación entre las dos computadoras por medio de los puertos seriales fue necesario hacer un cable con el cual conectarlas, figura 3.1.2. El tipo de conector con el cual cuentan los cables depende del tipo de conector que tengan las computadoras ya sea DB9 o DB25. El siguiente diagrama nos muestra como están conectados los cables para cada tipo de conector.

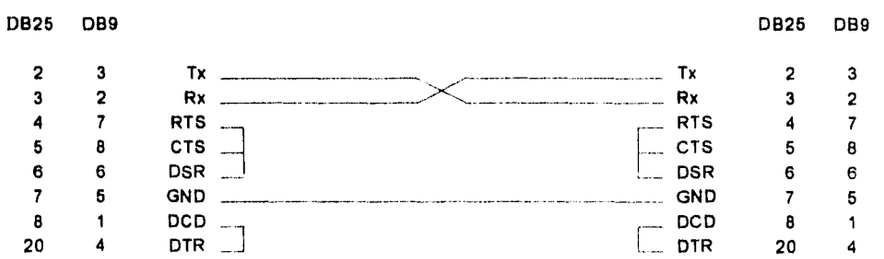


Fig. 3.1.2 Conexión entre conectores

3.2 ESTRUCTURA DEL PROTOCOLO DE COMUNICACION

Para poder entablar una comunicación ordenada es necesario controlarla mediante un protocolo que norme la forma de comunicarse entre máquinas. Por ello se diseñó y desarrolló el protocolo de la siguiente manera.

En la figura 3.2.1 podemos observar gráficamente el protocolo desarrollado. Como primer paso cuando la computadora origen quiere hacer un acceso al disco de otra computadora manda un paquete, que llamamos encabezado, de cuatro bytes. El primer byte contiene el identificador de la computadora destino. El segundo byte contiene el identificador de la computadora origen. Los dos bytes restantes contienen el número de bytes que seguirán al encabezado. Si el encabezado no corresponde a la computadora que lo ha tomado es retransmitido por la red para que la siguiente computadora lo analice. Si el byte corresponde a la computadora destino, es tomado y dependiendo del nivel de la rutina de atención de interrupción en el que nos encontremos es la acción que se ejecuta.

Cuando una computadora origen quiere iniciar la comunicación con una computadora destino manda una señal como petición de atención, esta señal es un caracter *ACK*; Cuando este caracter es recibido en la rutina de atención de interrupción de la computadora destino, se verifica si se está en condiciones de atender la petición, si es el caso se retorna un caracter de *RACK*. En caso contrario, no se enviará nada por estar ocupada y la computadora origen esperará tres segundos para volver a hacer la petición de atención hasta tres veces. Una vez enviado la señal *RACK*, la computadora origen enviará un paquete (frame) que contiene el tipo de operación que desea realizar (lectura o escritura), el primer sector y el número de sectores a operar. Una vez que la rutina de atención de interrupción ha recibido el paquete entonces lo analiza y determina si se trata de una operación de lectura o de escritura. Si es una operación de lectura la rutina de atención de interrupción extrae la información referente a las direcciones lógicas del disco y las pasa como parámetros a la rutina encargada de ejecutar la lectura. Cuando se realiza la operación se verifica la bandera de acarreo (carry) para detectar un posible error, si hubo error es mandado una caracter de regreso a la computadora origen *NSACK* que indica que la operación no tuvo

éxito, por lo cual la computadora origen esperará un pequeño lapso de tiempo para mandarle a la rutina de atención de interrupción un caracter de *BELL* indicándole que vuelva a intentar la operación de lectura. Si la lectura fue exitosa la rutina de atención de interrupción manda un caracter *SACK* indicándole que se prepare para recibir el buffer que contiene la información que leyó.

En caso de que la operación a realizar sea una escritura, la computadora destino debe prepararse para recibir el contenido del almacenamiento temporal (buffer) que contiene la información que va a escribir en el disco, despues de recibir el paquete que contiene el tipo de operación.

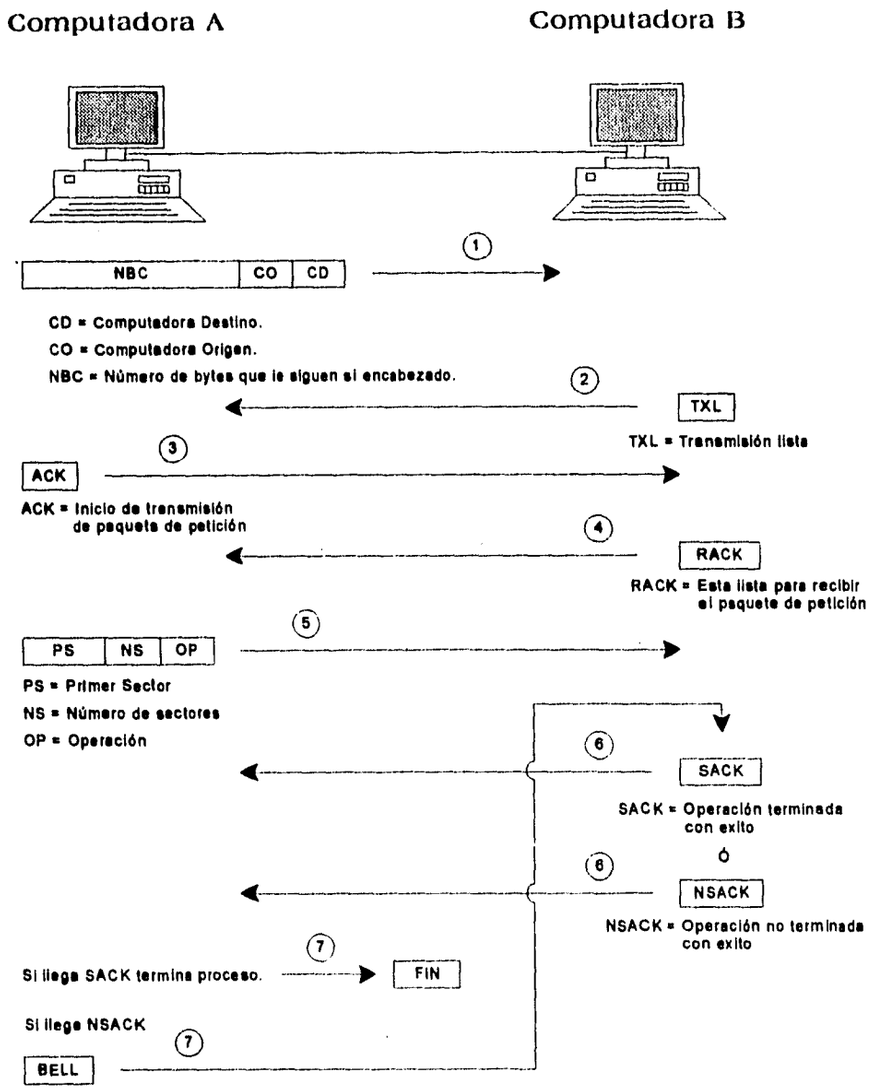


Fig. 3.2.1 Protocolo de comunicación

3.3 RUTINA DE ATENCION DE INTERRUPCIONES PARA EL PUERTO SERIAL

El uso de interrupciones es la base del presente trabajo, ya que cada vez que un caracter llega a cualquiera de los dos puertos produce una interrupción y son activadas las rutinas de atención de interrupción, lo cual permite a la información viajar a través de la red y pueda llegar a cualquier computadora.

En este caso la interrupción será originada por cualquiera de los puertos seriales el COM1 o el COM2. Desarrollamos rutinas de atención interrupción para cada uno de los puertos. Para el COM1 la rutina únicamente se encarga de retransmitir cada bit que le llegue. La rutina para el COM2 tiene básicamente tres funciones:

- Controlar el flujo de la estafeta (Token) a través de la red.
- Retransmitir información que no es para la estación.
- Realizar operaciones de lectura y escritura hacia el disco duro.

RUTINA DE ATENCIÓN DE INTERRUPCION COM 1

Cuando una interrupción es provocada por el puerto serial 1 (COM 1), figura 3.3.1, significa que la computadora está sirviendo de paso a través de la cual únicamente va a circular información que no es para ella y será puesta nuevamente en la línea para que sea tomada por la siguiente computadora. En esta función de repetición, cada byte que entra por el puerto es retransmitido sin ningún análisis ni tratamiento. De esta manera el flujo de información es tan rápido que la computadora que está sirviendo de repetidor no sufre ninguna alteración en su velocidad de proceso, por lo cual resulta imperceptible para el usuario que por su computadora están circulando cadenas de bytes.

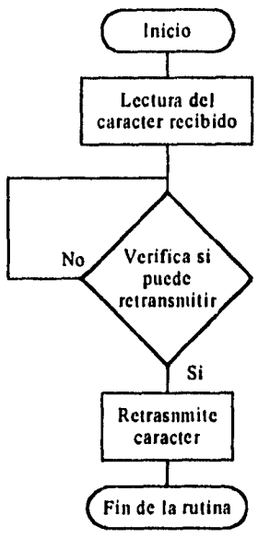


Fig. 3.3.1 Rutina de atención de interrupción COM 1

RUTINA DE ATENCION DE INTERRUPCION COM 2

Dado las tres funciones básicas que realiza la rutina de atención de interrupción cuando una interrupción es generada mediante el puerto serial 2 (COM 2), debe ser capaz de saber que es lo que va a realizar, para ello se ha dotado a la rutina de atención de interrupción de banderas de estado que le indican la función que debe realizar.

Control de la estafeta: Esta sección de la rutina de atención de interrupción se encarga de recibir, retransmitir, analizar e inicializar contadores para el tiempo de estancia de la estafeta en cada computadora, figura 3.3.2. La estafeta es un caracter con un valor constante (en este caso 26), cuando un caracter provoca una interrupción al puerto se analiza si este corresponde al valor de la estafeta, si es el caso, se inicializa un contador con el valor de 18 que corresponde aproximandamente a un segundo como veremos más adelante y se activa una bandera para

indicarle al manejador de dispositivos (driver) de la computadora que se encuentra analizando el caracter que la estafeta está presente y que podría intentar transmitir información. Cada vez que sea llamada la rutina del temporizador (TIMER) aproximadamente 18.2 veces por segundo el contador será decrementado en una unidad, hasta que llegue a cero, esto ocurrirá en un segundo aproximandamente, en este momento la rutina se encargará de desactivar la bandera que indica la presencia de la estafeta y de retransmitir la estafeta a la siguiente computadora.

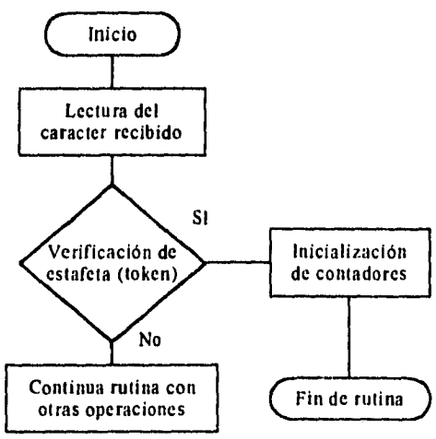


Fig. 3.3.2 Control de la estafeta

Retransmisión de información: Cada vez que una computadora quiere hablar con otra computadora envía un paquete de información que contiene 4 caracteres (bytes), figura 3.3.3, el primer caracter corresponde al identificador de la computadora con la que se quiere establecer comunicación (computadora destino), el segundo caracter contiene el identificador de la computadora que quiere hablar (computadora origen) y los dos caracteres restantes indican el número de caracteres que seguirán a este paquete en el momento en que se establezca comunicación con la computadora destino. Cuando este paquete haya sido recibido se analizará el

primer caracter para determinar su destino, si este caracter coincide con el identificador de la computadora que lo analiza ésta regresará un caracter de recibido y atenderá la petición, en caso contrario retransmitirá el paquete a la siguiente computadora sin ninguna modificación, cargará un contador de paso con el número de caracteres que le seguirán al paquete y activará una bandera que indicará que opera en modo de paso, es decir que retransmitirá todo caracter que le llegue por ese mismo puerto hasta que el contador de paso llegue a cero.

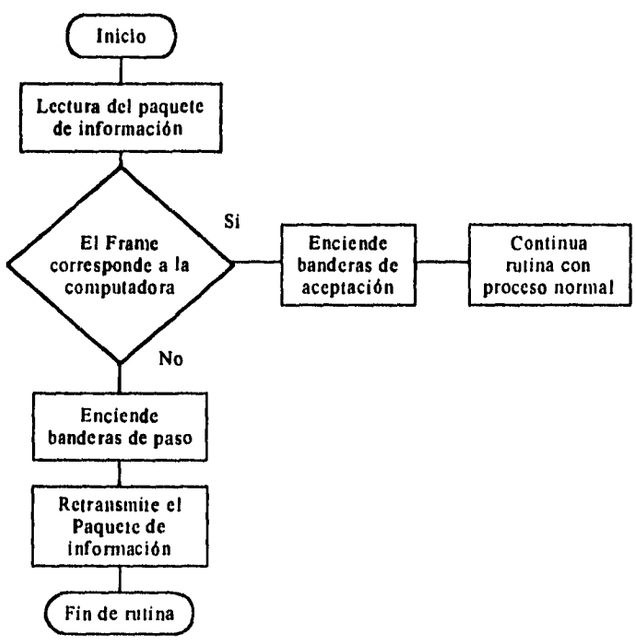


Fig. 3.3.3 Retransmisión de información

Lectura y escritura hacia el disco duro: Cuando un manejador de dispositivos (device driver) desea hacer uso del disco duro de alguna otra computadora para una operación de lectura o de escritura, arma un paquete de ocho bytes que contiene el tipo de operación que desea realizar, el primer sector que desea leer y cuantos sectores del disco desea leer o escribir a partir del primer sector que indicó anteriormente. Cuando este paquete es recibido por la rutina de atención de interrupción es analizado para extraerle la información anterior para pasarla a una rutina que se encargará de convertirla a cabezas, cilindros y sectores que sean necesarios para que pueda efectuarse una lectura y escritura transparente, figura 3.3.4. Si la operación que se realizó fue una lectura, la información leída es colocada en un almacenamiento temporal (buffer) para posteriormente transmitirla de regreso a quien la solicitó. Si se trata de una escritura, la información que se desea escribir es recibida por el puerto y colocada en un almacenamiento temporal para posteriormente escribirla en el disco duro de acuerdo con los parámetros de primer sector y número de sectores a escribir contenidos en el paquete.

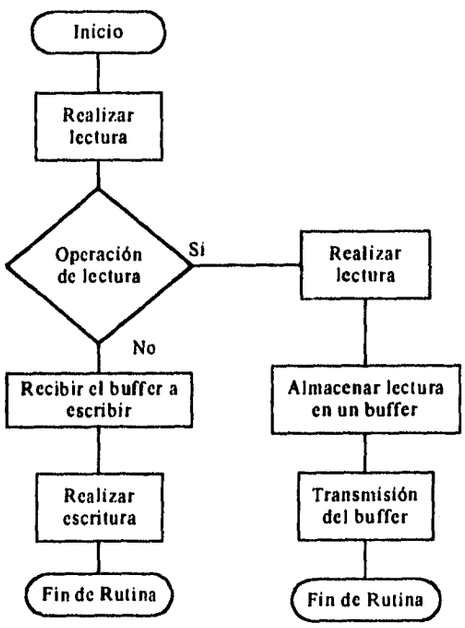


Fig. 3.3.4 Lectura y escritura de una disco duro

PROGRAMA RESIDENTE

Este programa se encargará de inicializar los puertos seriales con los parámetros adecuados, habilitar al controlador de interrupciones, para que pueda producir interrupciones cada vez que llegue un dato por alguno de los dos puertos, y finalmente instalar las direcciones en el vector de interrupciones de las nuevas rutinas de atención de interrupción para los puertos seriales (Interrupcion 0Ch para COM 1, 0Bh para COM 2.), para el temporizador (TIMER) (Interrupcion 08h.) y para el disco duro (Interrupcion 13h.). Dentro de este programa, figura 3.3.5, se encuetran las rutina de atención de interrupción que controlan los puertos seriales, el temporizador y el disco duro.

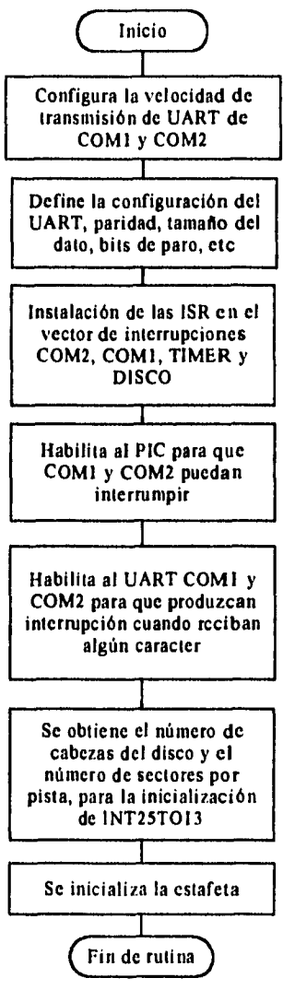


Fig. 3.3.5 Programa residente

RUTINA TIMER

Esta rutina es la encargada de controlar el tiempo de estancia de la estafeta en cada computadora, figura 3.3.6. Esta rutina se ejecuta automáticamente 18.2 veces por segundo aproximadamente. Cada vez que se activa la rutina lo primero que hace es ejecutar la antigua rutina de interrupción, salva el estado de las banderas y de todos los registros del procesador, verifica si la estafeta se encuentra presente, si es así, decrementa su contador y sale de la rutina. Si este ha llegado a cero, apaga la bandera que indica la presencia de la estafeta y la retransmite a la siguiente computadora. Si la estafeta no se encuentra presente en la computadora la rutina solamente recupera el estado de los registros y terminará su ejecución.

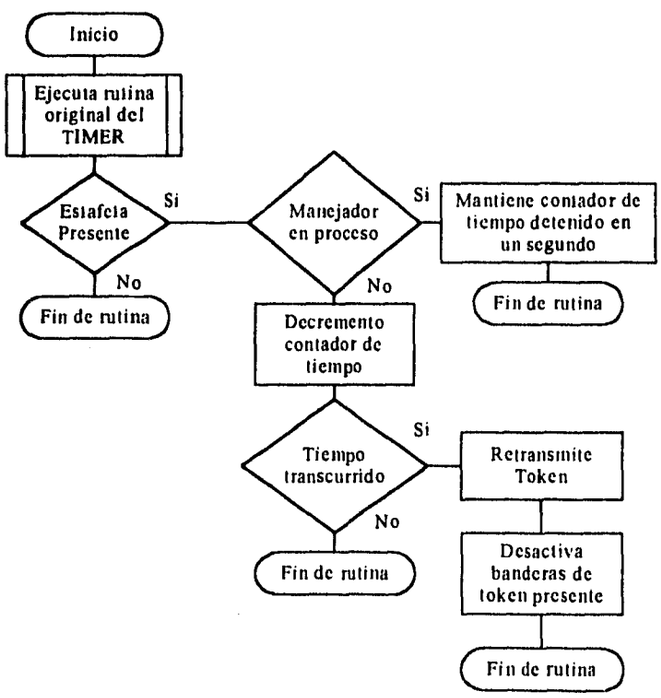


Fig. 3.3.6 Rutina timer

RUTINA INT25TO13

Debido a que la única manera accesar el disco duro desde una rutina de interrupción es por medio de la interrupción 13H porque es una función incluida en el BIOS y por lo tanto no entra en conflicto con la rutina de atención de interrupción, figura 3.3.7. Los parámetros requeridos por la interrupción 13H son: el número del cilindro, el número de cabeza, el primer sector sobre el cilindro y cuantos sectores a partir del primer sector van a leer, si es una operación de lectura, se necesita la dirección del almacenamiento temporal donde se guardará la información, y si la operación es de escritura, la dirección del almacenamiento temporal contiene la información que se va a escribir en el disco. Cuando el driver quiere realizar alguna operación ya sea de escritura o de lectura, los parámetros a enviar los pasa considerando la organización del disco de una manera lineal, vamos a llamarlas direcciónes lógicas, es decir únicamente nos manda el primer sector que quiere leer y cuantos sectores a partir de ese sector sin considerar en que cilíndro se encuentran esos sectores ni sobre que cara de disco están localizados para saber con que cabeza se tiene que leer, por lo cual requerimos hacer cálculos para transformar esas direcciones lógicas a direcciones fisicas. Para realizar esta operación fue creada la rutina.

Primero calculamos el cilindro sobre el cual se localiza el sector inicial haciendo uso de la siguiente fórmula:

$$Cilindro = \frac{\mathrm{Pr}\, iSec}{SecTrack * TotCab}$$

Una vez que tenemos el cilindro podemos calcular la cabeza que corresponde a ese cilindro empleando la siguiente fórmula:

$$Cabeza = \left(\frac{\mathrm{Pr}\, iSec}{SecTrack} + 1\right) - (Cilindro * TotCab)$$

Finalmente calculamos el número de sector fisico inicial del cilindro por medio de la siguiente fórmula:

$$Sector = \left[\mathrm{mod}\left(\frac{\mathrm{Pr}\, iSec}{SecTrack}\right)\right] + 1$$

Ya que se han obtenido las direcciones físicas, estas se pasan como parámetros a la interrupción 13H de la siguiente manera:

- **AH** el tipo de operación a realizar 02h para lectura y 03h para escritura.
- **AL** contiene el número de sectores a leer o escribir.
- **CH** especifica el número de cilindro.
- **CL** especifica el número del sector incial.
- **DH** especifica la cabeza.
- **DL** especifica el disco que se va a utilizar (80h para el primer disco duro).
- **ES:BX** especifica la dirección del almacenamiento temporal (buffer) de donde va a dejar la información en caso de lectura o en donde la va a tomar en caso de escritura.

Cuando se realiza la interrupción 13H ésta regresa el estado de la operación, si la bandera de *CARRY* ha sido encendida es porque ha ocurrido algún error. Si este es el caso, el error será codificado en **AH** de lo contrario la bandera de *CARRY* no se prenderá y **AH** contendrá el valor de cero y **AL** el número de sectores que se operaron.

Debido a que cada cilindro tiene un número determinado de sectores y de que cada cabeza sólo puede accesar cierto número de cilindros, no se pueden efectuar operaciones de un gran número de sectores sin ajustar el número de cilindro y el número de cabeza, por lo tanto la rutina se encarga de controlar el número de cabeza y de cilindro para realizar las lectura correctas con las direcciones físicas.

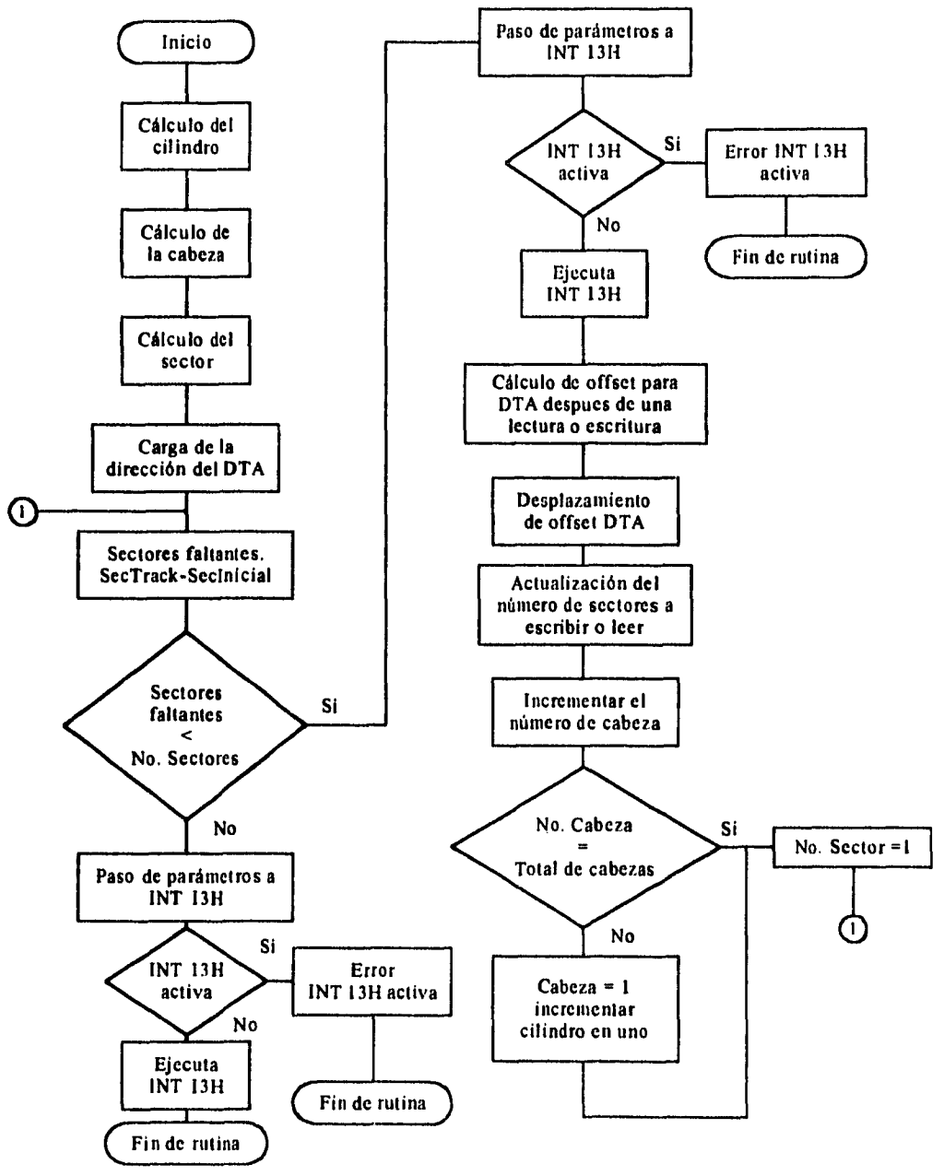


Fig. 3.3.7 Rutina int25to13

RUTINA DISK_INT

Algunas operaciones dentro de la computadora son dependientes del tiempo, es decir, sino se ejecutan en un cierto tiempo puede terminarse en un bloqueo del sistema, o en un mal funcionamiento del mismo. Una de estas operaciones son los accesos a disco los cuales no pueden ser interrumpidos mientras están en proceso, por lo tanto, debido a que una petición de acceso al disco puede llegar en cualquier momento, debe de encenderse una bandera que indique que en ese momento el disco está siendo accesado, de manera que cuando el programa quiera hacer uso del disco, verifique la bandera, y se asegure que puede realizar alguna operación, figura 3.3.8. Una vez terminado el acceso, la bandera debe ser apagada. Esta bandera es controlada desde la rutina de atención de interrupción de la interrupción 13H que es la interrupción encargada del manejo de los discos blandos y del disco duro en su más bajo nivel.

Fig. 3.3.8 Rutina Disk_int

RUTINA INI_TIEMPO

En algunos procedimientos es necesario esperar una respuesta de alguna petición, o hacer que algunas banderas duren encendidas durante algún tiempo. Por ello es necesario llevar un control del tiempo real. La rutina INI_TIEMPO, figura 3.3.9, se encarga de inicializar un contador con el valor reportado por la interrupción 1AH. El valor reportado por esta interrupción se encuentra en un contador interno que se incrementa en uno a razón de 18.2 veces por segundo. Esta es una forma cómoda de llevar contadores de tiempo real con una precisión de 1/18 de segundo aproximadamente.

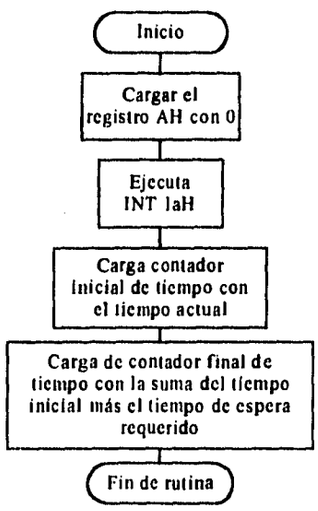


Fig. 3.3.9 Rutina Ini_tiempo

RUTINA RS232_INT_COM2

Las figuras 3.3.10 y 3.3.11 muestran el procedimiento completo de la rutina de atención de interrupción del COM2, es decir, se integran todos los procedimientos que se explicaron anteriormente.

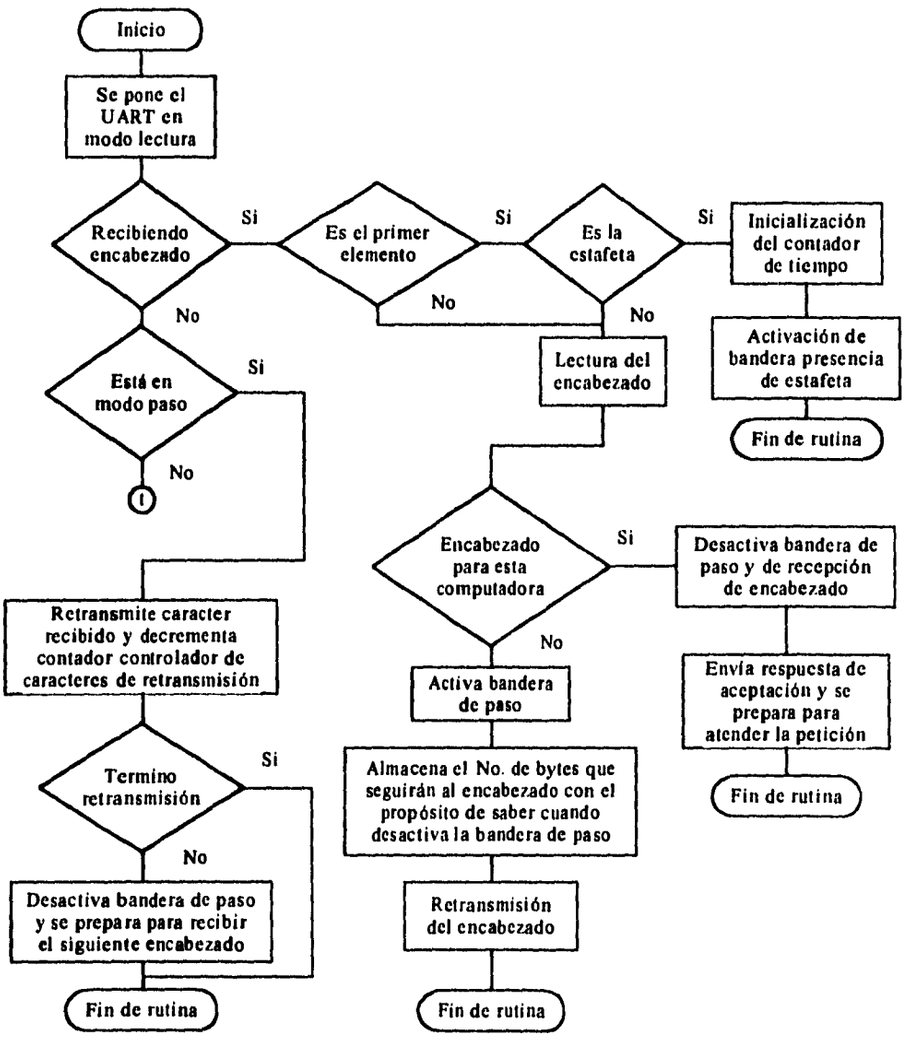


Fig. 3.3.10 Rutina de atención de interrupción

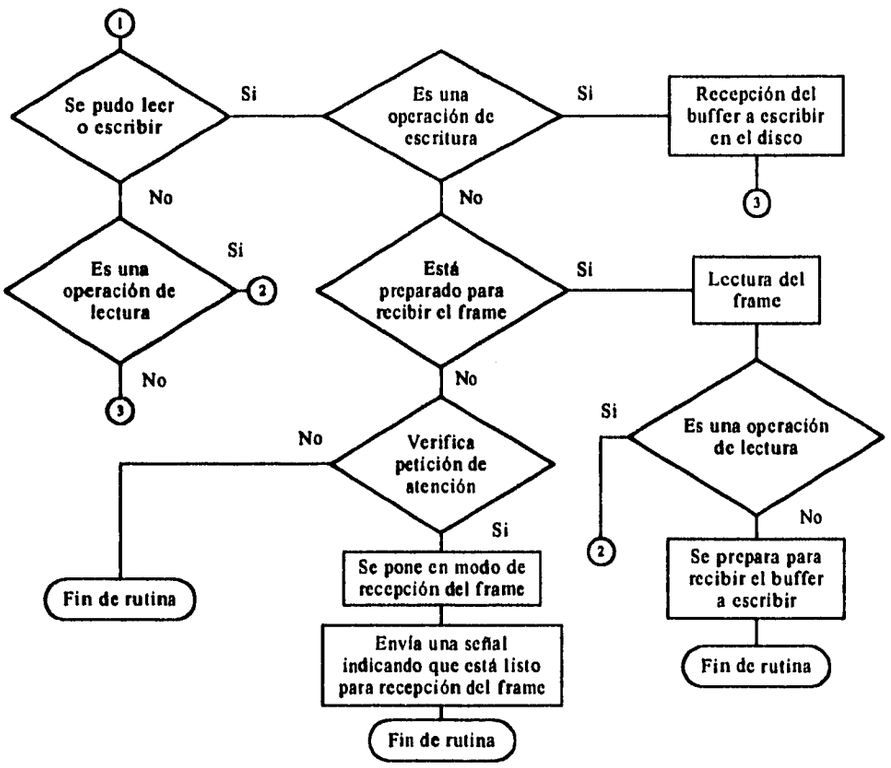


Fig. 3.3.11(a) Rutina de atención de interrupción

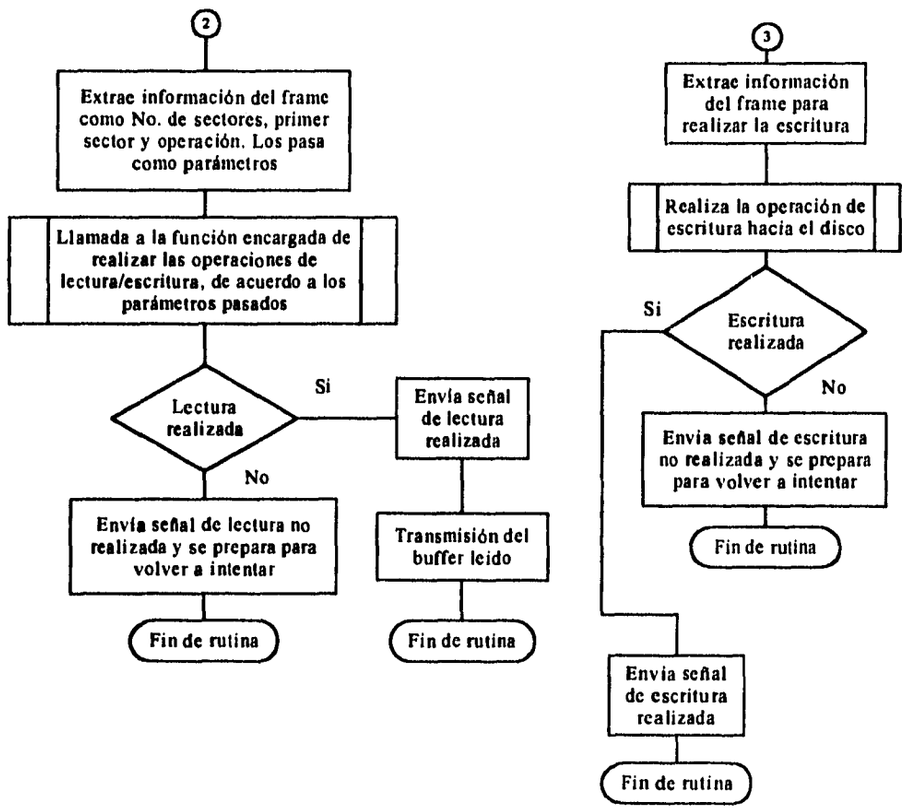


Fig. 3.3.11 (b) Rutina de atención de interrupción

3.4 DESARROLLO DEL MANEJADOR DE DISPOSITIVO (DEVICE DRIVER)

El manejador de dipositivo representa una de las partes más importantes en el desarrollo de la red. Esta importancia se debe a que el manejador desarrollado permite al usuario hacer accesos al disco duro de la computadora con la que entabla comunicación. Por lo que el manejador es el controlador del disco duro remoto. Esto permite disponer del disco duro como si estuviera instalado en la computadora que utilizamos. Por ésto, el usuario se dirige al disco duro remoto como a cualquier otro disco.

Algunas de las características, que debe tener el manejador desarrollado son: definir al manejador como uno tipo bloque y cubrir con las especificaciones y estructura de todo manejador. (Sección 2.4)

El manejador se define como un manejador tipo bloque porque el acceso del disco duro remoto, se efectúa a través de bloques de información.

La estructura de un manejador debe contar con una serie de rutinas, tales como la rutina de interrupción, la rutina de estrategia, y las rutinas de los comandos. Las rutinas de los comandos incluyen otras rutinas y procedimientos que cubren las necesidades requeridas por la red en anillo. Es decir, son la parte de comunicaciones del manejador. Esta parte se encarga de enviar señales, vía el puerto serial, para iniciar y establecer comunicación con la computadora destino que se quiere trabajar. Las rutinas y procedimientos de esta parte del manejador efectúan tareas como las siguientes:

- Comunicarse con la computadora que posee el disco duro al que se quiere tener accesos,
- Obtener la información necesaria del disco duro en cuestión, y
- Procesar la información obtenida para proporcionársela a los comandos del manejador, que el sistema operativo solicita.

A continuación, se describe cada uno de los módulos y secciones que forman a nuestro manejador. Primero se describirá en detalle cada sección relacionada con las definiciones, las

constantes y los atributos del manejador. Posteriormente se describen las rutinas de estrategia e interrupción, así como los comandos utilizados y las rutinas de los mismos.

DIRECTIVAS

Cuando el programador desea moverse por el segmento de datos, y fijar su programa en una zona específica de la memoria, se utiliza la directiva ORG (origen). Esta pseudo instrucción[1] informa al ensamblador que la instrucción u operando en la línea siguiente debe ser colocada en una localización de memoria especificada por el argumento o número siguiente a ORG.

Los manejadores de dispositivos, siempre se instalan en la primera localidad del segmento (offset 0000h). Así que el argumento de la directiva ORG es un cero.

CONSTANTES DEFINIDAS EN EL MANEJADOR

En esta sección del programa, se declaran las constantes que son usadas a lo largo de la programación del manejador. Se definen dos tipos de constantes. Las primeras, que llamaremos constantes del encabezado de petición (request header), son las que guardan una dirección predeterminada (offset) que permite identificar las funciones del encabezado de petición. Esto significa que por ejemplo, para localizar el número de comando, la constante deberá ser igual a 2. Ya que en el encabezado de petición el número de comando se encuentra 2 bytes después de la dirección inicial del encabezado de petición.

Las segundas constantes, que nombramos constantes de comunicación, son todas las utilizadas por las rutinas de comunicación. Algunas constantes de comunicación, definen las direcciones de los Registros del circuito integrado UART 8250 (Universal Asyncronous Receiver/Transmiter) descritos en la sección 2.2. Y que se utilizan durante la comunicación entre las computadoras. Otras constantes de comunicación establecen los valores que se necesitan en el protocolo de comunicación.

[1]Una pseudo instrucción no es una instrucción de máquina sino una instrucción que se dirije al ensamblador, para que la utilice durante el ensamblado del programa.

CABECERA DEL MANEJADOR

Como se dió a conocer en la sección 2.4, en la cabecera del manejador se especifica la conexión con el siguiente manejador, el atributo del manejador, el apuntador de la rutina de estrategia, el apuntador de la rutina de interrupción y el espacio en memoria necesario para que se almacene el nombre del manejador.

. Para la conexión con el siguiente manejador se declara un valor de -1 como valor inicial en el segmento (offset) y -1 como segmento. Lo que en hexadecimal se representaría con el valor FFFFH para el valor inicial en el segmento y FFFFH para el segmento. Estos valores le indican al sistema operativo que el manejador únicamente manejará un solo dispositivo y que no existe ninguna conexión con otro dispositivo.

El espacio en memoria que se reserva para el nombre del manejador, en nuestro caso, será llenado por una sola letra que corresponde al número del manejador que se instala. Por ejemplo, en una computadora se encuentran instalados tres manejadores tipo bloque, que controlan discos flexibles y duros; uno para un disco flexible de 3 1/2", otro para un disco flexible de 5 1/4" y uno más para un disco duro.

Al primer manejador se le asigna el número 0 ó letra "A", y el segundo manejador el número 1 ó letra "B" y finalmente, al tercero el número 2 ó letra "C". Por lo que, si se instala un nuevo manejador tipo bloque, el número y letra asignados serían: el 3 y la letra D. Esta última letra que corresponde al último manejador que se instaló, es la que se almacena en el espacio reservado, para el nombre del manejador. Si se instala otro manejador, además del asignado a la letra D, entonces el último manejador instalado tomará como nombre, la letra E.

Las computadoras componentes de la red en anillo desarrollada, contienen un número variable de manejadores. Este número varía de acuerdo al número de computadoras instaladas en la red. Por ejemplo, si la red está compuesta de tres nodos, a cada computadora ó nodo se le instalan dos manejadores. Cada manejador controla, a cada una de las computadoras conectadas al nodo, donde los manejadores se encuentran instalados.

Así, para hacer referencia a cada nodo de la red, no hay más que teclear la letra asignada en el momento de la instalación del manejador.

TABLA DE COMANDOS

La tabla de comandos que se lista en esta parte del manejador, contiene los 16 comandos utilizados por el mismo. La tabla esta compuesta en cada uno de sus renglones de la dirección (offset) de cada función que atiende las peticiones del sistema operativo. De los 16 comandos permitidos, en nuestro manejador, se han implementado únicamente cinco. Los comandos que no poseen implementación propia, utilizan el código de alguno de los cinco, eligiéndolo de acuerdo a si cubren o no sus necesidades. Los 5 comandos implementados son los siguientes: inicialización (*Comando 0*), verificación del medio (*Comando 1*), construcción del bloque de parámetros del BIOS (*Comando 2*), lectura (*Comando 4*), escritura (*Comando 8*), medio removible (*Comando 15*).

Con la implementación del código de los comandos mencionados, es suficiente para el buen funcionamiento del manejador. Ya que estos comandos, son los comandos básicos para los manejadores que controlan al disco duro bajo el sistema operativo MS-DOS.

ESTRUCTURA, CONTENIDO Y APUNTADOR DEL BLOQUE DE PARAMETROS DEL BIOS

En este módulo del manejador, se define la estructura del bloque de parámetros del BIOS (Sección 2.4). Se guardan, de acuerdo a la estructura definida en la tabla 2.4.7 (Estructura del Bloque de parámetros del BIOS), las localidades de memoria necesarias, para almacenar el contenido del bloque de parámetros del BIOS del disco duro, para el que está hecho el manejador. Valores como: los bytes por sector, el número de cabezas, el número de sectores ocultos, entre muchos otros, son leídos del disco duro y luego almacenados en las localidades de memoria.

El apuntador del bloque de parámetros del BIOS, siempre tendrá la dirección en la que se localiza la estructura de éste mismo. Y por ello en futuras referencias el acceso a éste se realiza a través del apuntador.

RUTINA DE ESTRATEGIA Y RUTINA DE INTERRUPCION

En la figura 3.4.1 se ejemplifica, la comunicación que existe entre el sistema operativo y el manejador de dispositivo (Device Driver). Cuando el sistema operativo hace acceso al manejador, primero, se ejecuta la rutina de estrategia e inmediatamente después la rutina de interrupción. La rutina de estrategia salva la dirección del encabezado de petición en una variable definida en el sistema. Por su parte la rutina de interrupción, es la rutina medular del manejador, ya que desde ella se ejecutan todos los comandos posibles implementados. También verifica si el comando que el sistema operativo solicita es válido, si lo es, lo ejecuta. Al término de la rutina de interrupción, el manejador devuelve al sistema operativo, a través del encabezado de petición, los resultados obtenidos después del acceso que se ha realizado.

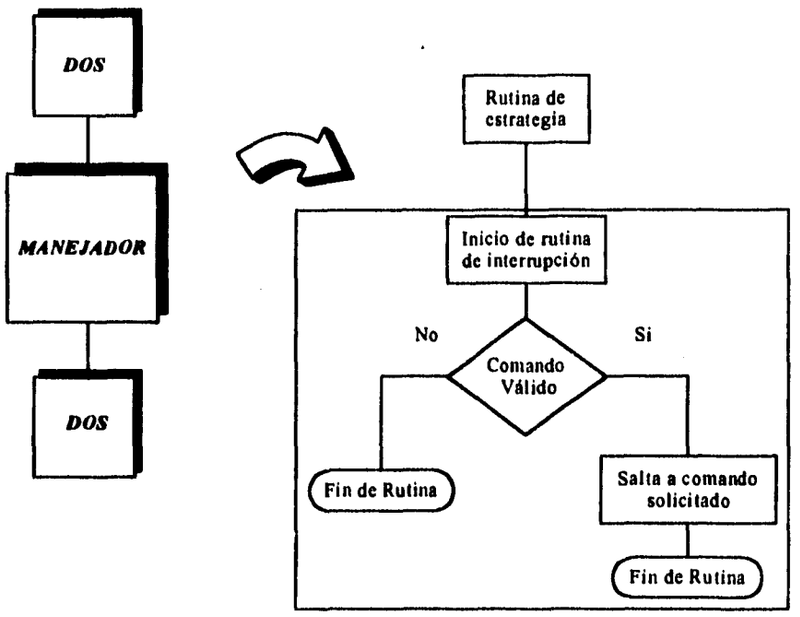


Fig. 3.4.1 Rutina de estrategia y Rutina de interrupción

RUTINA DE INTERRUPCION

La rutina de interrupción como se ha mencionado en varias ocasiones, realiza sus actividades, inmediatamente después de que la rutina de estrategia ha salvado, la dirección del encabezado de petición.

Dentro de la rutina de interrupción, encontramos algunas instrucciones que controlan aspectos relacionados con la red en anillo que se ha desarrollado. Estas instrucciones están orientadas a cubrir las características de la topología en anillo.

En la figura 3.4.2 se describe en forma general cuales son las actividades que la rutina de interrupción realiza. Como primer punto, la rutina de interrupción guarda todos los registros del

procesador en la pila, ya que dentro de la rutina pueden ser modificados, lo que provocaría errores futuros en el sistema. Posteriormente, se obtiene la dirección del encabezado de petición a través de la variable que la almacena. Esta dirección se asigna a los registros ES:DI. Una vez que estos registros contienen la dirección del encabezado de petición, son utilizados como vía de comunicación con el sistema operativo. Por ejemplo para la siguiente línea de código:

```
mov   bl, es:[di + cmd_fld]
```

donde:

bl es la parte menos significativa del registro de propósito general BX.

cmd_fld es una constante del encabezado de petición, definida de la siguiente forma:

```
cmd_fld     equ   2
```

La línea de código mostrada, mueve al registro **bl,** el contenido de la dirección del encabezado de petición más 2 bytes. Esto significa que el contenido de esa dirección pertenece al número de comando que el sistema operativo quiere ejecutar (Tabla 2.4.3).

La rutina de interrupción debe corroborar que el número de comando que el sistema operativo solicita es uno de los comandos existentes. Cuando el comando no existe, en la palabra de estado del encabezado de petición, se indica que ha ocurrido un error, activando el bit 15 de la palabra de estado, y en los bits del 0 al 7, estableciendo el valor del código de error correspondiente a *comando desconocido* (consultar Tablas 2.4.4 y 2.4.5).

Cuando el comando solicitado se encuentra dentro de la tabla de comandos, se ejecuta la rutina del comando correspondiente. Al finalizar, se guarda en el registro **ax**, el estado en el que terminó la rutina (error, éxito, etc.). Además, cuando el control retorna a la rutina de interrupción, la palabra de control debe contener el estado que la rutina del comando arrojó y debe especificarsele que ésta se ha ejecutado, encendiendo el bit 8 de la misma palabra de estado.

Continuando con la figura 3.4.2, si el comando solicitado es el de inicialización (comando 0). Se procede a ejecutar la rutina que atiende a este comando. En caso contrario, entonces, se

verifican las banderas que controlan las necesidades de la red, las que se explican más adelante. Una vez que el comando solicitado termina sus funciones, se llena la palabra de estado y se recuperan los valores de los registros almacenados, al principio, en la pila. Con esto último el manejador termina su labor y regresa el control al sistema operativo.

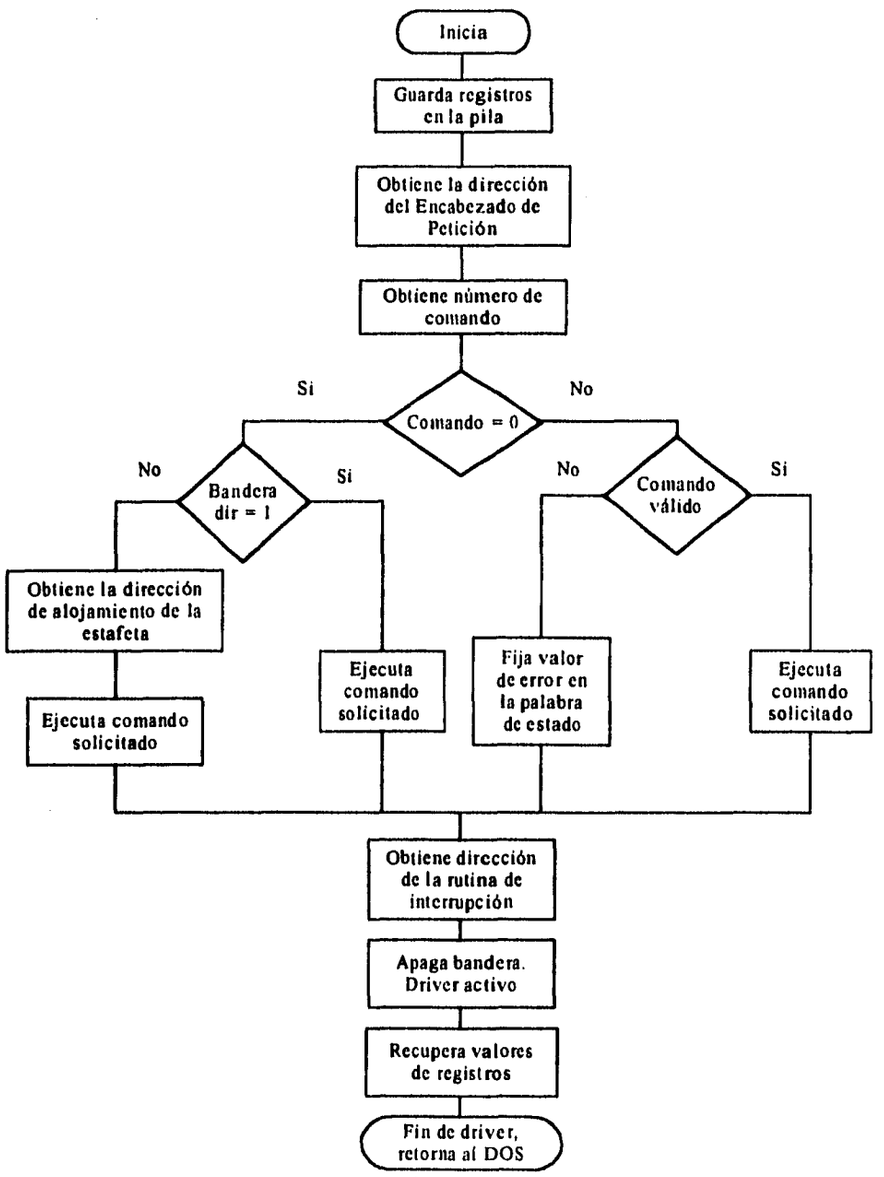


Fig. 3.4.2 Rutina de Interrupción

PROCESOS DENTRO DE LA RUTINA DE INTERRUPCION QUE CUBREN LAS NECESIDADES DE LA RED

La rutina de atención de interrupciones (ISR) del COM2 tiene dos banderas que pueden ser accesadas desde el manejador. Para ello, el manejador utiliza la dirección de la memoria en la que se encuentran almacenados. Las banderas que se encuentran en la rutina de atención de interrupciones tienen una doble función. Por una parte al manejador, le permiten conocer el momento en el que la computadora tiene en su poder la estafeta (Bandera *token*), y por otra, permiten a la rutina de atención de interrupciones conocer cuando el manejador está activo o ejecutando alguno de sus comandos (Bandera *driver activo*).

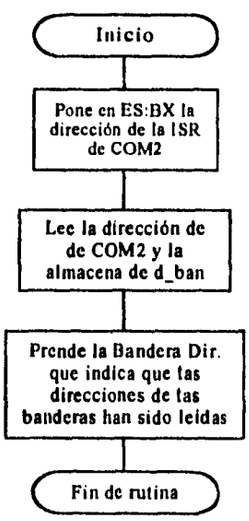


Fig. 3.4.3 Procedimiento de banderas

La dirección de la memoria, en donde se almacenan las banderas, se obtiene por medio del procedimiento de banderas. Mismo que se describe en la figura 3.4.3. Como se observa en la figura, durante el procedimiento se obtiene la dirección de las banderas y se indica en una bandera más (*dir*) que éstas ya han sido leídas, evitando que sean leídas nuevamente en procesos posteriores.

El comando de inicialización (comando 0) no necesita comunicarse con otras máquinas durante su proceso. Por esto y siguiendo con el orden de la figura 3.4.2, tiene que verificarse si el comando que se ejecutará es el comando 0. Si no lo es, entonces, la bandera *dir* es verificada. Cuando la bandera no está encendida, se hace una llamada al procedimiento de banderas. Finalmente, si la bandera *dir* está encendida se hace la llamada a la rutina del comando correspondiente.

Antes de que la rutina de interrupción restaure todos los registros almacenados en la pila, ésta, realiza algunas tareas como la de apagar la bandera de *driver activo*, ésto indica al ISR que la actividad en el manejador ha finalizado. Para apagar esta bandera, primero se verifica si la bandera *token* se encuentra activa, ya que si ésta no lo está, es imposible que el manejador se encuentre haciendo accesos remotos.

COMANDOS

Las rutinas de los comandos que se han implementado para el manejador se localizan después de la rutina de estrategia y la rutina de interrupción. Para ejecutar cada uno de ellos es necesario, obtener en la rutina de interrupción el número de comando y luego saltar a la tabla de comandos, desde la cual se accesa a cada uno de los comandos implementados.

En la tabla de comandos, encontramos todos los comandos posibles en un manejador, incluso aquellos que pertenecen a los manejadores de tipo caracter. En la tabla se incluyen todos los comandos con la finalidad de evitar errores. Por ejemplo el sistema operativo podría llegar a pedir un comando no implementado. Si esto sucede, se localiza el comando dentro de la tabla de

comandos y luego se salta a una rutina que simplemente desactiva los bits de petición del encabezado de petición, como en el caso de la rutina de no ejecución (dummy).

Todos los comandos que pertenecen a los manejadores tipo caracter, desempeñan el código correspondiente a la rutina de no ejecución.

Las rutinas de los comandos de lectura y escritura, se utilizan en varias ocasiones y por diversos comandos, estos últimos están relacionados con algún tipo de lectura o de escritura. Esto se ejemplifica con comandos como: la lectura directa (comando 3) y la escritura con verirficación (comando 9).

RUTINA DE INICIALIZACION (COMANDO 0)

La rutina de inicialización (Figura 3.4.4), como su nombre la describe, inicializa algunos de los parámetros del manejador. Para el manejador que hemos implementado, es necesario, además, inicializar los puertos de comunicación por los que viaja la información que se transmite y recibe.

Las funciones que se realizan para establecer los parámetros del manejador durante la rutina inicialización son los siguientes:

- Asignar al manejador su letra de identificación. Después de haber obtenido del *encabezado de petición* el número de dispositivo. (Tabla 2.4.8).
- Determinar el tamaño del manejador. Este en realidad se calcula para proporcionarle al encabezado de petición la dirección de memoria libre (Break address). Además para encontrar el valor del segmento de la dirección es necesario calcular el número de párrafos, que el programa del manejador ocupa. Para ello se obtiene el tamaño del programa en bytes, para después dividirlo entre 16, obteniéndose así, el número de párrafos correspondientes.
- Indicar el número de dispositivos que el manejador puede controlar.
- Obtener la dirección del apuntador de la estructura del bloque de parámetros del BIOS. Esta estructura contiene la información necesaria para el uso del disco.

- Inicializar los puertos de comunicaciones. Considerando: la velocidad de transmisión, la paridad y el tamaño de la palabra (Tabla 2.2.4) La máxima velocidad[2] aceptada es de 38400 Bauds. Esto se obtuvo después de realizar varias pruebas de operación normal de la red. Porque al exceder la velocidad de 38400 Bauds se presentaron problemas de desincronización entre los componentes de la red, provocando la pérdida de datos.

Una vez que se han realizado cada una de las funciones anteriores, se almacena en una variable el valor que indica que la rutina ha concluido sin errores. La variable guardará el valor hasta que durante la rutina de interrupción sea leída y transferida al encabezado de petición. **(Tabla 2.4.4)**

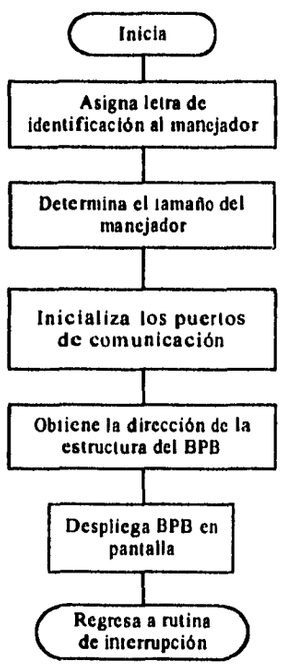


Fig. 3.4.4 Rutina de Inicialización

[2] La Velocidad de transmisión puede ser variada por el usuario durante la ejecución del programa de configuración de la red.

RUTINA DE VERIFICACION DEL MEDIO(Comando 1)

El sistema operativo hace uso de este comando cada vez que se ejecuta una lectura o escritura, para estar seguro de que el medio o disco con el que se trabaja no ha cambiado desde el último acceso. Para el caso del manejador que hemos desarrollado, sabemos que el disco nunca cambiará, ya que trabajamos con un disco duro. Por esto, cuando se ejecuta la rutina se especifica al encabezado de petición que el medio no ha cambiado. Por medio de la siguiente línea de código se indica al encabezado de petición lo requerido:

```
mov  byte ptr    es:[di+cambia],1
```

en donde:

es:di es la dirección del encabezado de petición.

cambia es el desplazamiento en el encabezado de petición, en donde se especifica si el medio ha cambiado o no (Tabla 2.4.12).

Para finalizar la rutina de verificación del medio se limpia la palabra de estado (Tabla 2.4.4) para borrar el bit de ocupado e indicar al sistema operativo que el comando se ha ejecutado.

RUTINA DE CONSTRUCCION DEL BPB (Comando 2)

Cuando el sistema operativo hace una llamada a ésta rutina, es porque necesita el bloque de parámetros del BIOS del dispositivo controlado por el manejador. Generalmente se requiere de este comando cada vez que el comando de verificación del medio indica que el medio ha cambiado. Para el manejador que se ha desarrollado lo anterior nunca sucede, porque en el comando 1 siempre se indica que el medio no cambia.

Aún así, el sistema operativo llega a necesitar de este comando, por lo que cada vez que es llamado proporciona la dirección del bloque de parámetros del BIOS al apuntador designado. Finalmente se borra el bit de ocupado de la palabra de estado y se regresa el control a la rutina de interrupción.

RUTINA DE LECTURA Y ESCRITURA (Comando 4 y 8)

El sistema operativo requiere del comando de lectura o de escritura, cada vez que necesita leer datos o escribir datos en el disco duro remoto. Para el comando de lectura es necesario especificar al sistema operativo, cual es el número de sectores a transferir, así como el primer sector a ser leído. Los datos que se obtengan son escritos en la parte de memoria que el sistema asigna para este trabajo.

Para el comando de escritura, la información a ser escrita se encuentra almacenada en la localidad de memoria, misma que el sistema operativo asigna.

En la rutina, el primer paso a seguir, es verificar la presencia de la estafeta, porque es necesario para poder hacer cualquier petición al disco remoto, cuando la estafeta no esta presente el manejador espera hasta dos minutos para detectarla, en caso contrario se indica que se ha presentado un error.

Una vez tomada la estafeta, el manejador saltará a la rutina encargada de la operación de escritura o lectura (**lec_esc**); esta rutina toma el contenido de la variable ***oper*** y lo analiza, si el contenido de esta variable es un dos se trata de una operación de lectura, y si el contenido es igual a tres la operación a realizar es una escritura. La rutina **lec_esc** (Fig. 3.4.5) fue desarrollada con el fin de extraer la información referente al primer sector a operar y el número de sectores a operar, para pasarla a otro procedimiento (***sec1***) que se encargará de fragmentarla para que la petición sea atendida de sector en sector.

La rutina **lec_esc** controla el número de sectores a leer debido a que la máxima petición de sectores que se puede manejar con un solo segmento son 128, ya que cada sector contiene 512 bytes lo cual nos da un total de 64Kb que es lo máximo que podemos manejar con un solo apuntador de segmento, en caso de que se presente una petición mayor a 128 sectores, la rutina se encargará de fragmentar la petición para que pueda ser manejada correctamente debido a que se tiene que actualizar el apuntador de segmento sumándole 4096 párrafos de 16 bytes cada uno lo que provoca que el apuntador de segmento se coloque en el siguiente segmento consecutivo.

Una vez terminado su trabajo verifica la bandera de acarreo para determinar la presencia de un error, si ésta está encendida es llenada la palabra de estado con un error en caso contrario se llena la palabra de estado con el valor correspondiente a que todo está correcto.

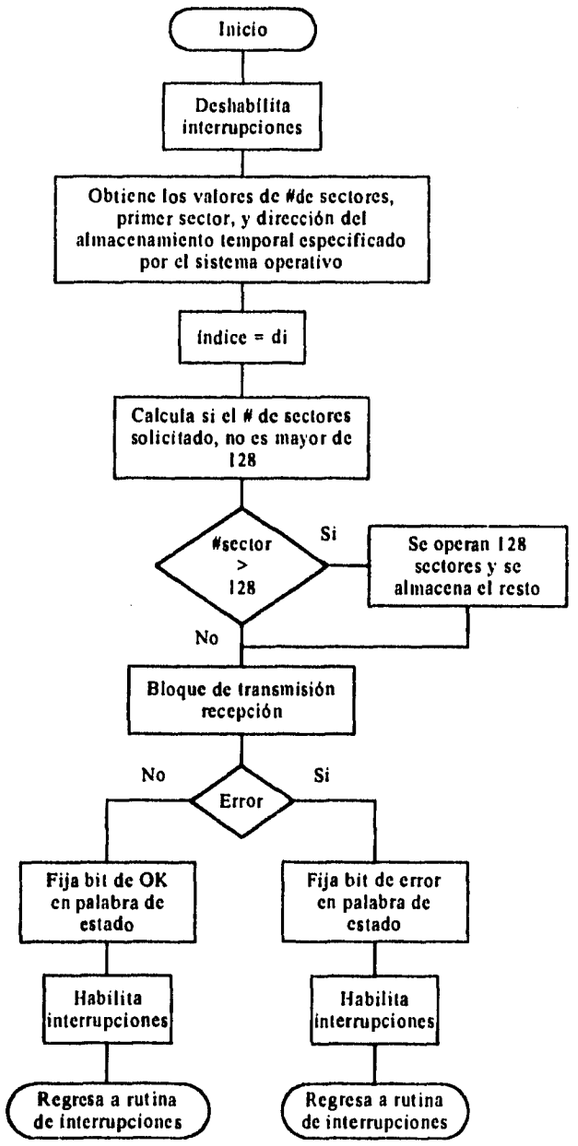


Fig. 3.4.5 Procedimiento lec_esc

Procedimiento SECI

Este procedimiento (Fig. 3.4.6) es el encargado de fragmentar una petición de varios sectores en peticiones de un solo sector para que puedan ser manejados de una manera más eficiente por la computadora que está siendo accesada. Cuando la rutina es llamada, los valores de primer sector y número de sectores se encuentran localizadas en dos variables: ***PriSec*** y ***NumSec*** respectivamente, las cuales nos ayudarán en la fragmentación. Durante el primer ciclo de petición a la rutina **Ack_Envia** se le pasan como parámetros el valor original de ***PriSec*** y siempre un sector para operar.

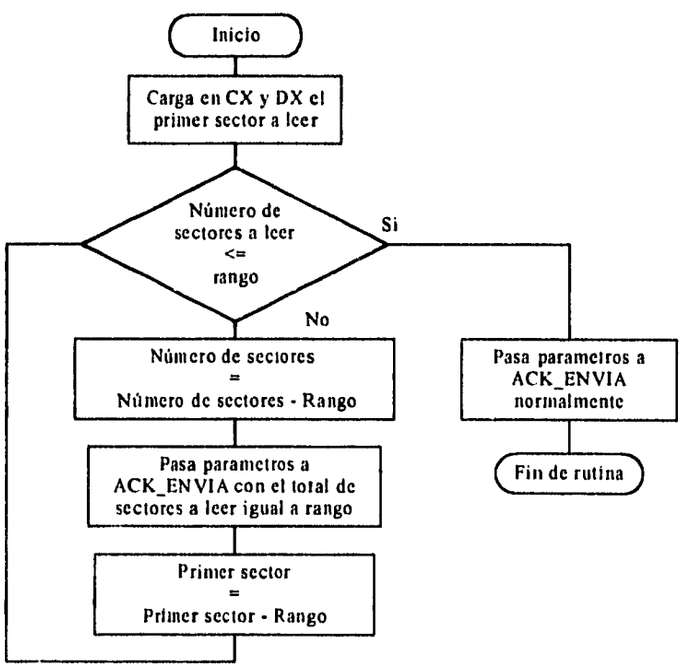


Fig. 3.4.6 Procedimiento seci

Durante los siguientes ciclos el valor de ***PriSec*** se incrementará en uno hasta el final, mientras que ***NumSec*** se decrementará en uno hasta llegar a cero indicando este resultado que se ha transmitido la petición completa.

Procedimiento Ack_Envía

Este procedimiento (Fig. 3.4.7) es el más importante dentro del desarrollo del manejador de dispositivos. Este procedimiento requiere que se le proporcione el primer sector, el número de sectores a operar y la operación a realizar (lectura o escritura). En el caso de una lectura se necesita saber la dirección del buffer donde se van a colocar los datos leídos y en caso de una escritura es necesario saber la dirección del buffer donde se encuentra almacenada la información a escribir en el disco.

Una vez que se cuenta con toda esta información, el procedimineto se encarga de armar un paquete con la siguiente información para que sea tomada por la computadora destino y ésta pueda atender la petición correctamente: el paquete lleva la operación a realizar, para una lectura será un dos y en el caso de una escritura será un tres, enseguida el primer sector a operar y finalmente el número de sectores que serán operados.

Ya que se ha terminado de armar el paquete el siguiente paso es transmitirlo, para esto es necesario indicarle a la computadora destino que se prepare para recibir el paquete. Para esta tarea es mandada una señal llamada *ACK* a la computadora destino, y se esperará una respuesta de ésta llamada *RACK*, esta señal será esperada por espacio de tres segundos. Si al término de este tiempo no se tiene respuesta, se vuelve a mandar la señal *ACK* y se repite el procedimiento hasta un número máximo de tres intentos para obtener la respuesta. Si la respuesta nunca es recibida, se salta directamente al final de la rutina **lec_esc**, señalando que ha ocurrido un error de transmisión, y finalizando así, la actividad del manejador.

Una vez que se tiene la respuesta de *RACK*, se transmite el paquete byte por byte. Cuando ya se ha transmitido el paquete se analiza para ver que operación está en cuestión. Si la operación

Durante los siguientes ciclos el valor de ***PriSec*** se incrementará en uno hasta el final, mientras que ***NumSec*** se decrementará en uno hasta llegar a cero indicando este resultado que se ha transmitido la petición completa.

Procedimiento Ack_Envía

Este procedimiento (Fig. 3.4.7) es el más importante dentro del desarrollo del manejador de dispositivos. Este procedimiento requiere que se le proporcione el primer sector, el número de sectores a operar y la operación a realizar (lectura o escritura). En el caso de una lectura se necesita saber la dirección del buffer donde se van a colocar los datos leídos y en caso de una escritura es necesario saber la dirección del buffer donde se encuentra almacenada la información a escribir en el disco.

Una vez que se cuenta con toda esta información, el procedimineto se encarga de armar un paquete con la siguiente información para que sea tomada por la computadora destino y ésta pueda atender la petición correctamente: el paquete lleva la operación a realizar, para una lectura será un dos y en el caso de una escritura será un tres, enseguida el primer sector a operar y finalmente el número de sectores que serán operados.

Ya que se ha terminado de armar el paquete el siguiente paso es transmitirlo, para esto es necesario indicarle a la computadora destino que se prepare para recibir el paquete. Para esta tarea es mandada una señal llamada *ACK* a la computadora destino, y se esperará una respuesta de ésta llamada *RACK*, esta señal será esperada por espacio de tres segundos. Si al término de este tiempo no se tiene respuesta, se vuelve a mandar la señal *ACK* y se repite el procedimiento hasta un número máximo de tres intentos para obtener la respuesta. Si la respuesta nunca es recibida, se salta directamente al final de la rutina **lec_esc**, señalando que ha ocurrido un error de transmisión, y finalizando asi, la actividad del manejador.

Una vez que se tiene la respuesta de *RACK*, se transmite el paquete byte por byte. Cuando ya se ha transmitido el paquete se analiza para ver que operación está en cuestión. Si la operación

es una lectura, se esperará el resultado de la petición de lectura, cuando la computadora destino ha realizado la lectura manda un señal llamada *SACK*, en caso contrario manda una señal *NSACK*.

Si la señal *SACK* es recibida se procede a recibir la información leída, y se termina con el procedimiento, apagando la bandera de acarreo para indicar el éxito de la operación. En caso de que se haya recibido una señal *NSACK* se esperará un lapso de 0.5 seg para mandar una señal llamada *BELL* a la computadora destino con el fin de reintentar la operación, hasta que sea obtenido un *SACK*. En caso de no obtener ninguna respuesta durante cinco segundos se indicará un error en la operación encendiendo la bandera de acarreo.

Si la operación es una escritura el procedimiento es el mismo hasta el momento de mandar el paquete de petición, pero después de esto el procedimiento toma otro camino. Se envía la información contenida en el almacenamiento temporal (buffer) de escritura, su tamaño que es de 512 bytes correspondientes a un solo sector. Una vez enviado el almacenamiento temporal, al igual que para una lectura, se espera la respuesta acerca de si pudo escribir la información en el disco la computadora destino con las señales *SACK* o *NSACK*, repitiéndose el procedimiento descrito en el párrafo anterior. Con la única diferencia que para la escritura no es necesario recibir información, puesto que la tarea en este comando es escribir información remotamente.

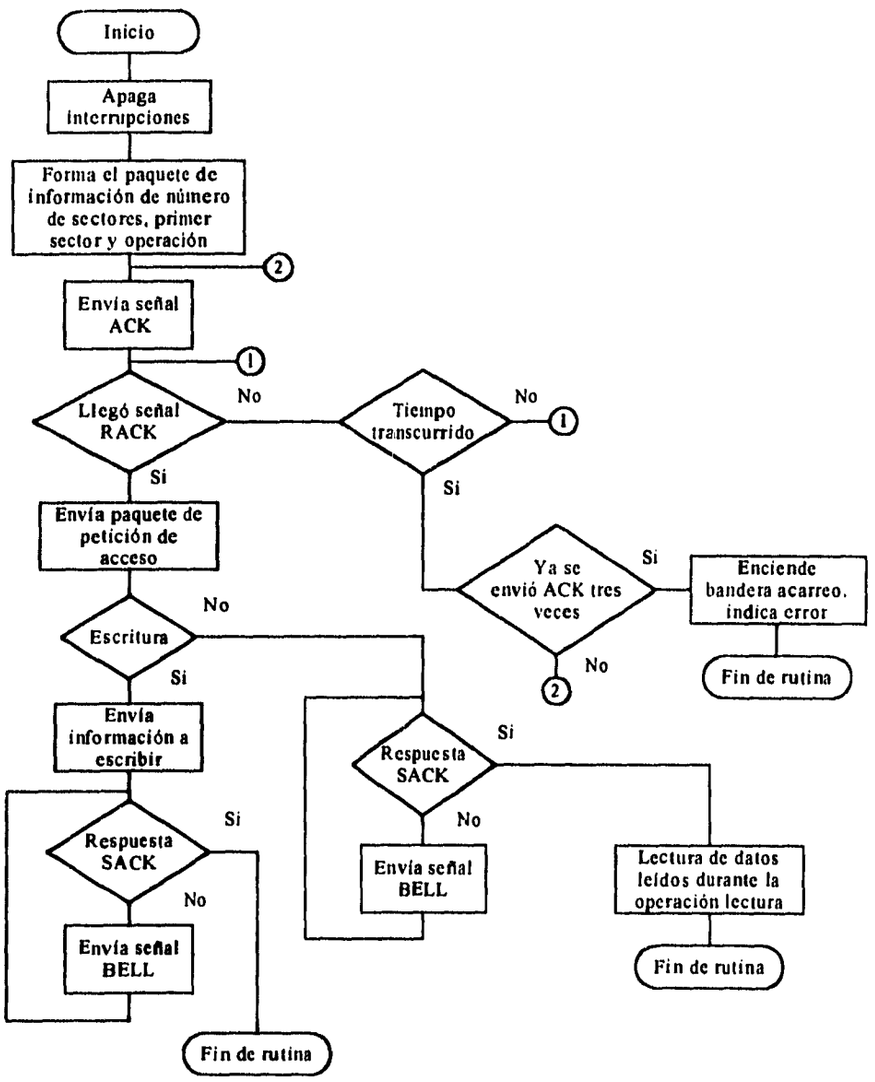


Fig. 3.4.7 Procedimiento Ack_envia

Procedimiento Token_Act

Este procedimiento (Fig. 3.4.8) es utilizado para determinar la presencia de la estafeta (Bandera *token*) en la computadora y poder hacer alguna petición a cualquier otra computadora. El procedimiento lee el contenido de la bandera que indica la presencia de la estafeta en la computadora, si el valor encontrado en la bandera es igual a uno indica que la estafeta está presente y se tiene permiso para realizar cualquier operación, por lo tanto activa una bandera indicando que el manejador va a realizar alguna operación (Bandera *driver activo*), esto se hace con el fin de indicarle al control de la estafeta que no la libere hasta que el manejador termine su tarea.

En caso de que la estafeta no se encuentre presente, el manejador esperará durante un lapso de dos minutos. Si al término de estos dos minutos la estafeta no se hace presente, se indicará un error señalando que la computadora que se quiere accesar está ocupada.

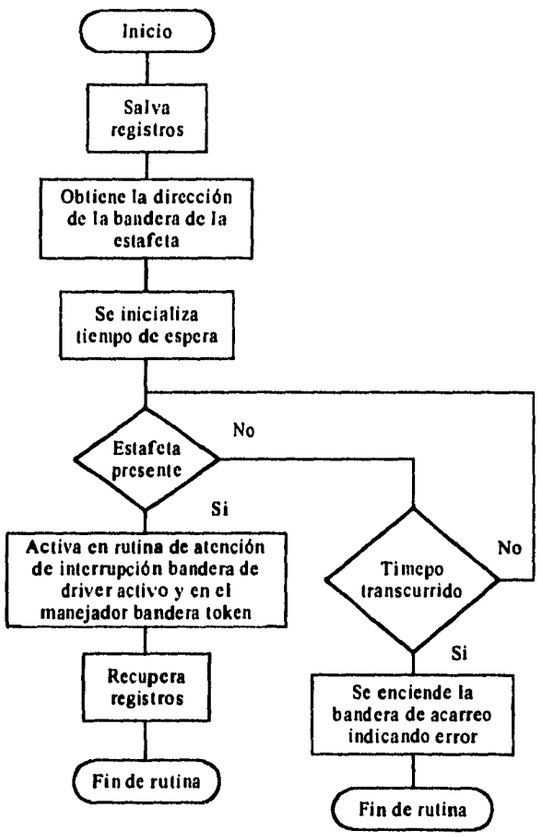


Fig. 3.4.8 Procedimiento Token_act

Procedimiento Enviar_ide

Antes de que alguna señal o algún paquete de información sea transmitido, debe de indicarse hacia donde va dirigido, por lo tanto es necesario mandar un paquete llamado encabezado que contenga información referente al destino de la información, quien manda la información y de cuantos bytes consiste la información que se va a enviar. De llenar este paquete con la información correcta y de enviarlo se encarga este procedimiento. (Fig. 3.4.9)

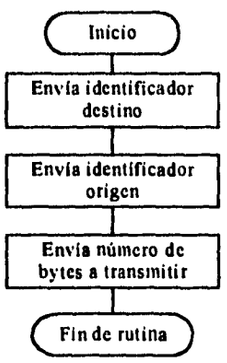


Fig. 3.4.9 Procedimiento Enviar_ide

Procedimiento Espera_Respuesta

Este procedimiento (Fig. 3.4.10) únicamente se encarga de esperar la respuesta a la petición de atención de operación del manejador a la computadora destino. Siempre es llamado inmediatamente despues de haber enviado el encabezado mediante el procedimiento **Enviar_ide**.

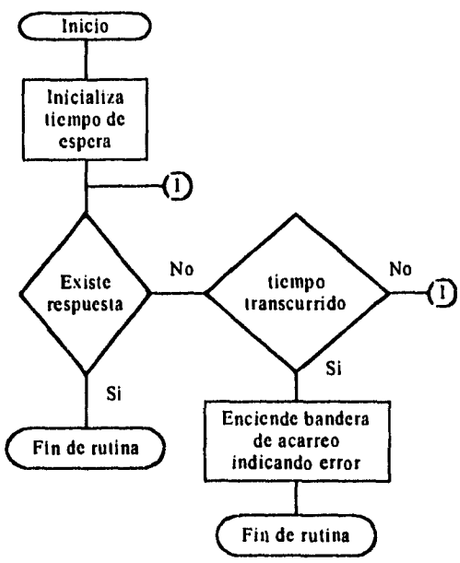


Fig. 3.4.10 Procedimiento espera_respuesta

Procedimiento Deshabilita y Habilita

Cuando el manejador se encuentra activado (Fig. 3.4.11), los datos que llegan por medio de los puertos originan interrupciones que son atendidas por la rutina de atención de interrupción y no por el manejador, esto provoca que los datos no lleguen al manejador. Debido a que en este procedimiento se utiliza la técnica de poleo del puerto, los datos son capturados de una manera más lenta.

Por este motivo es necesario desactivar las líneas que permiten producir interrupciones a los puertos seriales cuando reciben un dato. Esta desactivación (Fig. 3.4.12) se logra escribiendo directamente al controlador de interrupciones al puerto 21h un 1 en los bits 3 y 4 del puerto que corresponden al COM2 y COM1 respectivamente. Para activar nuevamente las interrupciones se escribe nuevamente al puerto 21h pero ahora se ponen 0 en los bits 3 y 4.

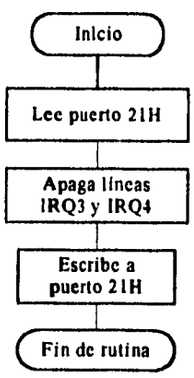


Fig. 3.4.11 Procedimiento deshabilita

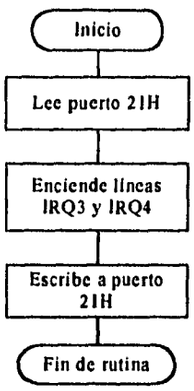


Fig. 3.4.12 Procedimiento Habilita

Rutina Ini_Tiempo

En algunos procedimientos es necesario esperar una respuesta de alguna petición, por lo tanto es necesario utilizar una rutina que inicialice contadores de tiempo real, como la utilizada en la rutina de atención de interrupción; como esta ya fue explicada en la rutina de atención de interrupción no se entrará en más detalle.(Fig. 3.4.13)

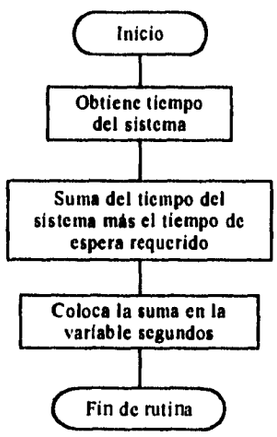


Fig. 3.4.13 Rutina Ini_tiempo

Procedimiento Estado

Cuando se va a transmitir o recibir un caracter por medio del puerto serial con el método de poleo del puerto, es necesario saber si algún caracter ha llegado. Esto se consigue mediante la lectura de la palabra de estado del puerto serial, en donde un bit de esta palabra es encendido cuando esto ocurre. Debido a la necesidad de leer constantemente este puerto se desarrolló este procedimiento para hacerlo más entendible y más fácil de manejar. (Fig. 3.4.14)

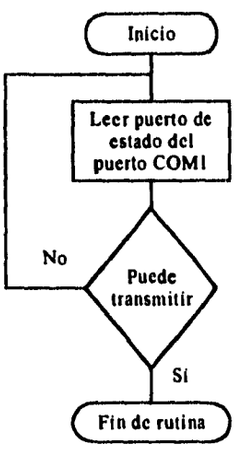


Fig. 3.4.14 Procedimiento de Estado

RUTINA DE MEDIO REMOVIBLE

El comando de medio removible indica si el disco, es un medio que puede cambiar o no. Para indicar que el medio no es removible se enciende el bit de ocupado de la palabra de estado del encabezado de petición. Cuando el medio es removible entonces el bit de ocupado permanece apagado.

Puesto que el disco duro remoto es un medio no removible, en la implementación de la rutina de este comando, se utilizó el siguiente código:

```
no_rem  proc near
        mov  ax,20
        ret
no_rem  endp
```

El valor 20 enciende el bit correspondiente al bit de ocupado, este valor se almacena en el registro *ax*, el cual durante la rutina de interrupción, se transfiere a la palabra de estado del encabezado de petición.

RUTINA DE NO EJECUCION (DUMMY)

La rutina de no ejecución es una rutina de protección para comandos no implementados en el manejador. Durante la ejecución de esta rutina únicamente se desactiva el bit que indica que un comando esta en operación, es decir apaga el bit de ocupado. El código para la rutina de no ejecución es el siguiente:

```
dummy  proc near
       xor  ax,ax
       ret
dummy  endp.
```

El comando de no ejecución es el último de los implementados y con esto, finalizamos lo referente a los comandos. La implementación de únicamente cinco comandos no significa que en un futuro no puedan aumentarse comandos a este manejador. El implementar los anteriores es debido a que con ellos fue suficiente para obtener los resultados deseados.

4. ALCANCES Y LIMITACIONES DE LA RED DESARROLLADA PRESENTACION DE RESULTADOS Y COSTOS

La red desarrollada puede operar en casi cualquier computadora personal compatible con IBM, desde la más antiguas hasta las más modernas, la restricción más fuerte es impuesta por la compatibilidad del BIOS el cual debe ser compatible con el de IBM, debido a que todas las operaciones implementadas en los programas fueron hechas con funciones de bajo nivel (BIOS).

A continuación mostramos algunos de los requerimiento deseables de las computadoras que serán conectadas a la red.

Los requerimiento de las computadoras para trabajar con esta red son:

- Computadora IBM compatible 8086 o superior.
- Monitor indistinto.
- Memoria mínimo 640 KB.
- Dos puertos seriales.
- Sistema operativo MS-DOS 4.0 en adelante.

Este tipo de red fue diseñada y desarrollada para un ambiente pequeño, la limitación es debido a las velocidades de transmisión que se pudieron alcanzar. El conectar un gran número de computadoras implicaría que la estafeta tardara más tiempo en llegar a cada una de las computadoras, y por lo tanto éstas tendrán que esperar más tiempo para transmitir una petición de acceso. La máxima velocidad a la que hemos podido transmitir sin errores por medio del puerto serial ha sido de 28,800 bauds (el transmitir 1 MB de información a la velocidad de 28,800 bauds nos llevaría 4 minutos con 51 segundos aproximadamente, si intentáramos transmitir un archivo de 1 MB por medio de un módem a través de la línea telefónica normal tendríamos que hacerlo a una velocidad de 14,400 bauds nos llevaría 9 minutos con 42 segundos), aunque se pudieron alcanzar velocidades mayores se presentaron problemas, por lo tanto no es recomendable trabajar a altas velocidades.

Los manejadores de dispositivos y las rutinas de atención de interrupciones fueron desarrolladas de acuerdo a las características del sistema operativo MS-DOS por lo cual únicamente se puede trabajar bajo esta plataforma. Si alguno de los usuarios en un momento dado quisiera trabajar bajo WINDOWS, este nodo de la red quedaría desactivado, y por lo tanto la red interrumpida, debido a que WINDOWS quita de la memoria todos los manejadores de dispositivos y rutinas de atención de interrupciones protegidas por funciones del sistema operativo MS-DOS para instalar los suyos. Esto es, si quisiéramos que esta red operara bajo ambiente WINDOWS, los manejadores de dispositivos y las rutinas de atención de interrupciones tendrían que ser desarrolladas de acuerdo a las características de la plataforma WINDOWS.

Una limitación de este desarrollo es el relacionado con la memoria disponible en las computadoras. Cada computadora debe tener un manejador de dispositivos por cada computadora que esté conectada a la red, por lo tanto vamos a ocupar más memoria alojando manejadores, esto implica que mientras más computadoras tengamos conectadas en la red, más memoria ocuparemos para tener más manejadores de dispositivos funcionando en cada computadora. La rutina de interrupción solo será instalada una sola vez por computadora debido a que únicamente controlará las funciones de esa computadora, lo cual no representa un gran consumo de memoria.

La versión del sistema operativo MS-DOS es importante debido a la estructura de algunos elementos utilizados en este trabajo, como lo son, el sector de arranque (boot) del disco duro, el tamaño de los discos duros que pueden manejar y la manera en que el sistema operativo maneje y controle los manejadores de dispositivos (device drivers). Muchos de estos cambios se han dado desde la versión 4.0 del sistema operativo, por lo tanto es necesario usar esta versión o una más reciente.

En una computadora IBM compatible podemos tener como máximo cuatro puertos seriales, el COM1, COM2, COM3 y COM4. Estos puertos funcionan con únicamente dos líneas del controlador de interrupciones , la IRQ3 y la IRQ4, el COM1 y el COM3 comparten la línea IRQ4 y el COM2 y el COM4 comparten la línea IRQ3. Debido a esta limitante no podemos conectar un ratón al COM3 o al COM4 porque tenemos que compartir la línea de interrupción del controlador de interrupciones con el manejador y la rutina de atención de interrupción que controlan la red, lo cual en un momento dado puede dar problemas, debido a que dos de estos puertos quieran hacer uso de la interrupción al mismo tiempo, esto derivaría en un paro del sistema, la caída de un nodo y por lo tanto la caída de toda la red.

El hecho de utilizar manejadores de dispositivos, nos permite ver a los discos duros de las otras computadoras como si fueran discos duros instalados en nuestra computadora, por lo tanto podemos realizar cualquier operación en ellos, como leer, escribir, y ejecutar programas, lo cual nos permite navegar a través de cualquier directorio y visualizar cualquier información, esto puede verse desde dos puntos de vista. El poder tener acceso libre y transparente a cualquier información puede representar una ventaja, el poder plantear políticas de organización para la red, y por otro lado se tiene el problema de que no existe ningún tipo de restricción a la información.

En cuanto los costos de hardware para esta implementación, son muy reducidos, lo único que se requiere para la implementación de la red es que se tenga dos puertos seriales y un cable por medio del cual se van a conectar las computadoras, hoy en día, las computadoras en el mercado usualmente traen dos puertos seriales, por lo tanto no se tiene que hacer ese gasto.

5. CONCLUSIONES

Después de realizar el presente trabajo, concluimos lo siguiente:

Comprender claramente el funcionamiento de los manejadores de dispositivos nos permite la configuración y el mantenimiento óptimo de las computadoras de escritorio, quitando, de alguna manera, el carácter mágico-místico que pueden significar los manejadores de dispositivos.

La experiencia en el desarrollo de los manejadores de dispositivos es muy importante, ya que en nuestro país no se elabora software de desarrollo. Por esto si México desea perfilarse como un país desarrollado tendrá, necesariamente, que dejar de ser un país usuario de software e ingresar en el área de desarrollo. Este es un pequeño esfuerzo para alcanzar este objetivo.

El área de redes es, en este momento un campo fértil para la implementación y desarrollo de sistemas. El comprender claramente los aspectos y problemas que implican los programas de comunicación entre computadoras es de vital importancia para el logro de este objetivo.

El conocimiento adquirido permitirá que en un futuro, se pueda desarrollar e implementar software controlador de nuevos dispositivos o plantear mejoras a los ya existentes, debido a que el funcionamiento de los manejadores de dispositivos ha quedado comprendido es posible emigrar a otras plataformas, como windows, unix, etc., con relativa facilidad.

Si bien es cierto que el software aquí presentado es sencillo en su manejo también es cierto que requirió de tiempo de investigación y pruebas, que a menudo resultaban frustrantes. Una razón de esto es que no existía bibliografía suficientemente documentada o actualizada tanto en el área de manejador de dispositivos como en el de programación de sistemas y MS-DOS. La presente tesis, es nuestra contribución a este grave problema.

El tema no está agotado y se requiere de más investigación y más pruebas con el fin de perfeccionar el sistema y ahondar en el desarrollo del manejador de dispositivos.

También es cierto que el manejador de dispositivos actual presenta algunos problemas, que no ha sido posible corregir, principalmente por la falta de información. Los problemas que requieren ser resueltos son los siguientes:

- Un mecanismo que garantice el correcto funcionamiento del sistema aún después de que una computadora de la red haya sido apagada.
- En algunas ocasiones, si dos computadoras accesan simultáneamente el mismo programa una de ellas se bloquea. Este problema ha sido muy difícil de resolver y aún no se encuentra una justificación de este.
- El funcionamiento del sistema bajo ambiente Windows es tema de otra tesis. El sistema actual no es capaz de soportarlo. Habría sido, imposible tratar de abordar los dos problemas en conjunto: el desarrollo del manejador de dispositivos y el ambiente Windows.
- Cuando dos computadoras de diferente velocidad se comunican, la velocidad de transmisión de la red debe ser disminuída, de modo que la más lenta es la que establece las velocidades máximas de comunicación.

Sin embargo y a pesar de estos problemas, el presente trabajo es una buena plataforma para continuar o empezar el desarrollo de otros manejadores de dispositivos, o incursionar en el desarrollo de programas de comunicaciones.

Finalmente, se obtuvo un software de comunicación de muy bajo costo y pocos requerimientos, el cual puede ser implantado en empresas cuyos requerimientos de cómputo no justifique la inversión en tarjetas, software e instalación de redes comerciales

También es cierto que el manejador de dispositivos actual presenta algunos problemas, que no ha sido posible corregir, principalmente por la falta de información. Los problemas que requieren ser resueltos son los siguientes:

- Un mecanismo que garantice el correcto funcionamiento del sistema aún después de que una computadora de la red haya sido apagada.
- En algunas ocasiones, si dos computadoras accesan simultáneamente el mismo programa una de ellas se bloquea. Este problema ha sido muy difícil de resolver y aún no se encuentra una justificación de este.
- El funcionamiento del sistema bajo ambiente Windows es tema de otra tesis. El sistema actual no es capaz de soportarlo. Habría sido, imposible tratar de abordar los dos problemas en conjunto: el desarrollo del manejador de dispositivos y el ambiente Windows.
- Cuando dos computadoras de diferente velocidad se comunican, la velocidad de transmisión de la red debe ser disminuída, de modo que la más lenta es la que establece las velocidades máximas de comunicación.

Sin embargo y a pesar de estos problemas, el presente trabajo es una buena plataforma para continuar o empezar el desarrollo de otros manejadores de dispositivos, o incursionar en el desarrollo de programas de comunicaciones.

Finalmente, se obtuvo un software de comunicación de muy bajo costo y pocos requerimientos, el cual puede ser implantado en empresas cuyos requerimientos de cómputo no justifique la inversión en tarjetas, software e instalación de redes comerciales

6. BIBLIOGRAFIA

- **Turbo Assembler Bible.**

  Syck Gary.

  The Waite Group's SAMS, Division of Macmillan Computer Publishing; Carmel Indiana, 1991.

- **Programmer's Reference Manual for IBM Personal Computers**

  Armbrust Steven, Forgeron Ted.

  Homewood, Illinois.: Dow Jones-Irwin; 1986.

- **PC Magazine DOS 5 Techniques and Utilities**

  Prosise Jeff.

  Ziff-Davis; Emeryville California; 1991.

- **Assembly Language Tools and Techniques for the IBM Microcomputers**

  Sánchez Julio.

  Prentice Hall; Englewood Cliffs, New Jersey; 1990.

- **The 8088 Microprocesor, programming, interfacing, software, hardware and applications**

  Singh Avtar, Triebel Walter A.

  Regents/Prentice Hall; Englewood Cliffs, New Jersey; 1991.

- **Wrinting MS-DOS Device Drivers**.

  Lai Robert S.

  Addison-Wesley; Reading, Masachusetts; 1992

- **Guía a las comunicaciones de IBM**

  Kruglinski David.

  Osborne/McGraw-Hill; Madrid; México;1985

- **Redes de computadoras Protocolos, Normas e Interfaces**

  Black Uyless

  Macrobit; Prentice Hall; Madrid; México; 1989

- **Advanced MS-DOS programming**

  Duncan Ray

  Microsoft; Redmond, Washington; 1988

APENDICE A

ESTRUCTURA DEL DISCO DURO

Para poder tener acceso al disco duro mediante un manejador de dispositivos, es necesario conocer la organización física y lógica del disco. El manejador de dispositivos al momento de su inicialización necesita conocer la organización lógica del disco duro para poder hacer sus peticiones de acceso correctamente.

ORGANIZACION LOGICA DEL DISCO DURO

Dentro de la organización lógica de un disco duro (Figura 1) tenemos cuatro secciones que serán descritas a continuación en detalle.

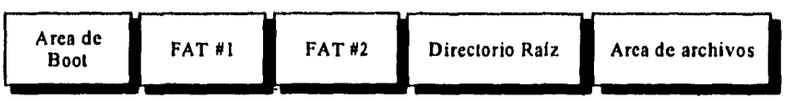


Figura 1. Organización lógica del disco duro.

1. El sector de arranque (Boot). Este es el primer sector que vamos a encontrar en un disco, contiene información necesaria para su operación y para la instalación del sistema operativo; figura 2. El primer byte que podemos localizar en el sector de arranque es un salto (E9 XXXX si se trata de un salto largo de 16 bits o EB XX90 si se trata de un salto corto de 8 bits, seguido de un 90 que indica una instrucción NOP-No operación) a la rutina localizada en el BIOS que se encarga de localizar el archivo del sistema operativo dentro del disco e instalarlo. Los siguientes ocho bytes contienen el nombre del fabricante del disco o el nombre y versión del sistema operativo con el cual se le dio el formato. Los bytes restantes son conocidos como el Bloque de Parámetros del BIOS (Bios Parameter Block BPB), en el Bloque de Parámetros del BIOS está codificada la información acerca de la organización física del disco, esta información debe ser leída por el manejador de dispositivos para que pueda calcular correctamente las

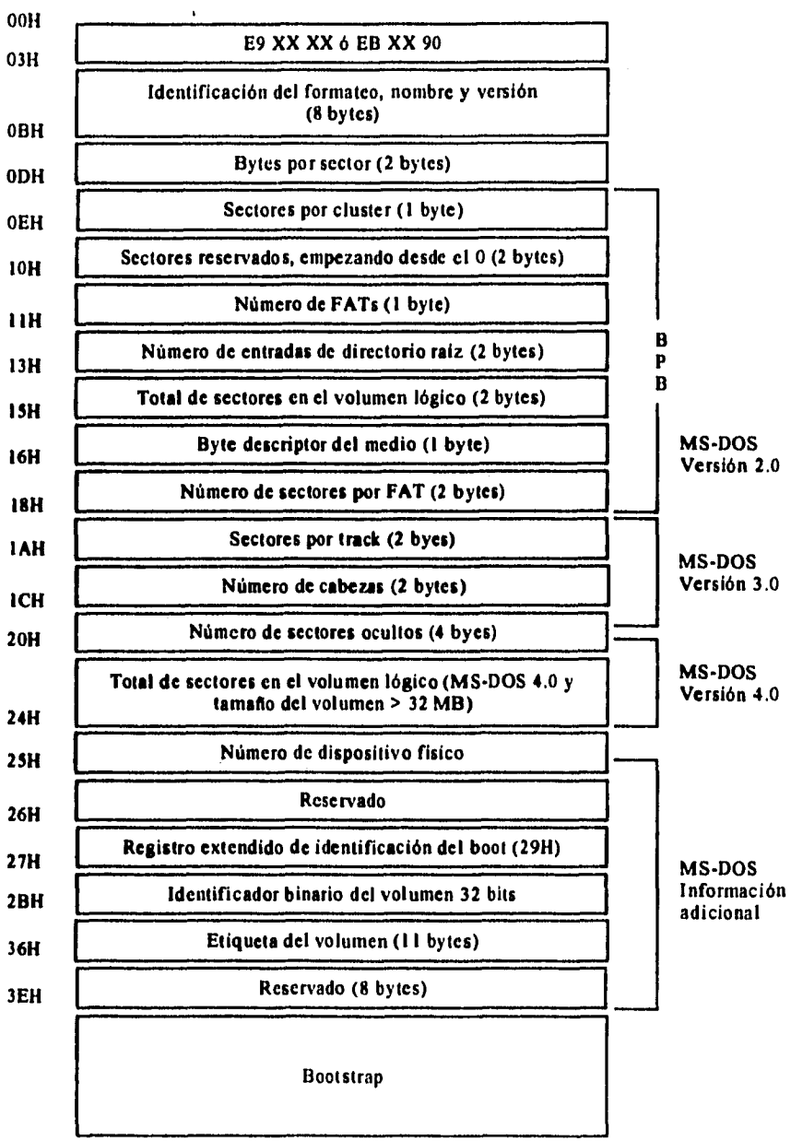


Figura 2. Estructura del Sector de Arranque (Boot) del disco duro.

direcciones fisicas de un sector lógico dado; así mismo el sistema operativo toma la información de Bloque de Parámetros del BIOS para calcular las direcciones físicas de la Tabla de Ubicación de Archivos (FAT), el directorio raíz y el área de archivos.

2. La Tabla de Ubicación de Archivos (FAT-File Allocation Table). La Tabla de Ubicación de Archivos mostrada en la figura 3, es una tabla que lleva un registro de las direcciones de los grupos de sectores (cluster) que componen a un determinado archivo, esta tabla es necesaria debido a que el almacenamiento de archivos es de manera aleatoria y se debe tener una manera de conectar los pedazos en los cuales son divididos los archivos cuando son almacenados. Como veremos más adelante un campo del registro que identifica a cualquier archivo dentro de la zona del directorio raíz, se refiere al grupo de sectores de inicio de este archivo. Esta información es utilizada por el sistema operativo para determinar por medio de la Tabla de Ubicación de Archivos en que grupos de sectores se encuentra el resto del archivo.

En cada localidad de la Tabla de Ubicación de Archivos vamos a encontrar un número que corresponde al siguiente grupo de sectores del archivo. Esto se repite hasta llegar a una localidad con el número **FFF8-FFFFH** que corresponde al final de archivo. Cuando en una localidad de la Tabla de Ubicación de Archivos encontremos el número **0000H** indica que el grupo de sectores no ha sido ocupado aún. Cuando encontramos el número **FFF7H** significa que este grupo de sectores está dañado y que por lo tanto no podrá ser utilizado. El número de sectores que conforman al grupo de sectores depende de cada formato del medio de almacenamiento, por ejemplo un disco de 5.25" con capacidad de 360 KB tiene 2 sectores por grupo de sectores, mientras que en un disco duro para computadora PC/AT tiene 4 sectores por grupo de sectores.

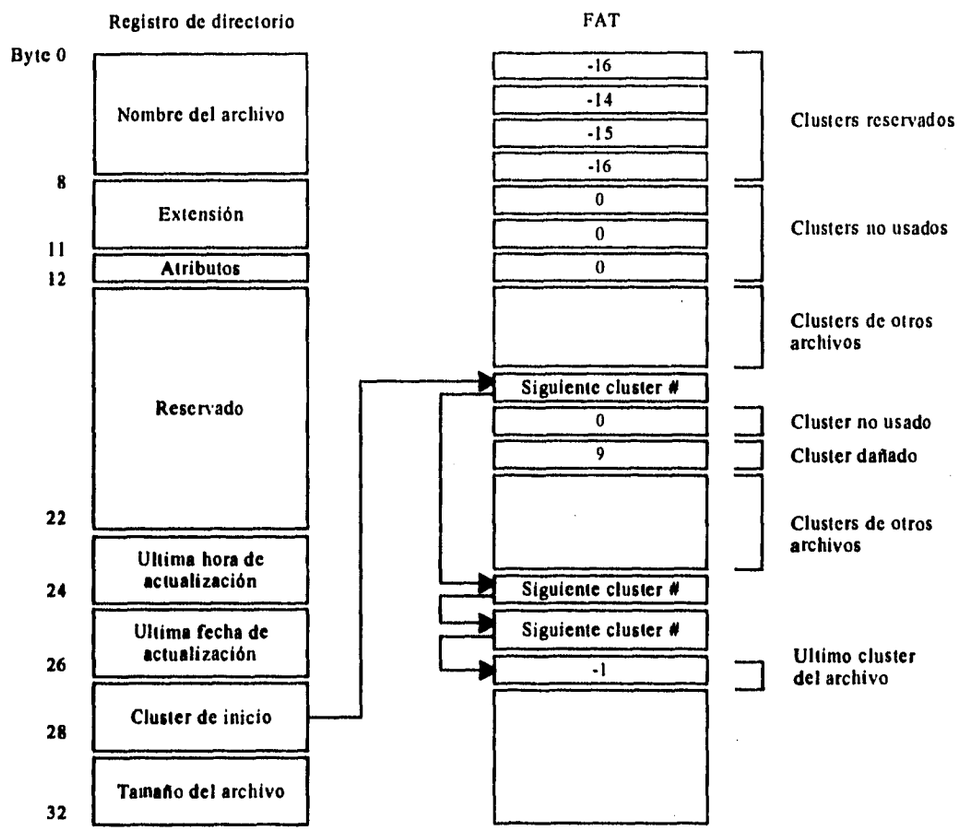


Figura 3. Directorio raíz relacionado con la Tabla de Ubicación de Archivos

3. El directorio raíz. En el disco inmediatamente después de la Tabla de Ubicación de Archivos encontramos la zona que corresponde al directorio raíz, su tamaño es determinado durante el proceso de formato. En esta zona tenemos un registro por cada archivo almacenado en disco y cada registro contiene campos que nos dan información de cada archivo; figura 4. En el directorio raíz el primer campo de ocho bytes contiene el nombre del archivo, el segundo campo contiene la extensión y es de tres bytes, el tercer campo que es de un solo byte contiene los atributos del archivo. El quinto campo está reservado para uso del sistema operativo, los campos sexto y séptimo, cada uno de dos bytes de longitud, contienen la hora y la fecha a la cual el archivo fue modificado por última vez respectivamente. El octavo campo, de dos bytes de longitud, contiene el grupo de sectores inicial. Finalmente el noveno campo, de cuatro bytes de longitud, contiene el tamaño del archivo en bytes.

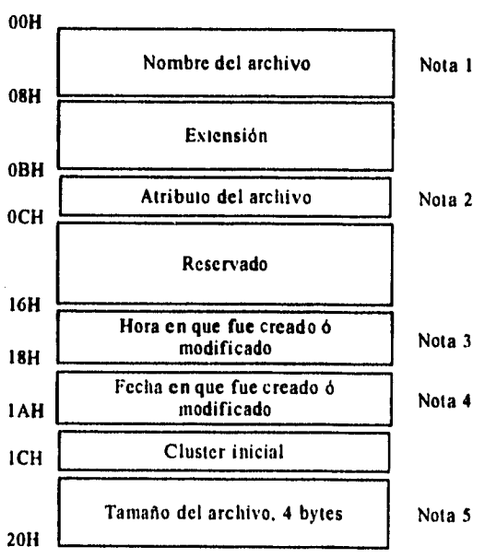


Figura 4. Estructura de un registro de archivo en el directorio raíz.

En algunos de los campos la información está codificada de modo que tiene un significado especial, a continuación se indica como debe ser interpretada esta información.

- Nota 1. El primer caracter del nombre del directorio puede ser cualquiera de los siguientes. Tabla 1.

Valor	*Significado*
00H	Localidad del directorio que nunca se ha usado
05H	El primer caracter del nombre del archivo es un E5H
E5H	El archivo ha sido borrado.

Tabla 1.

- Nota 2. El byte de atributo codifica la información de la siguiente manera. Tabla 2.

Bit	*Significado*
0	Solo lectura, todo intento por escribir en el o borrarlo será invalido.
1	Archivo oculto, será excluido de las búsquedas normales.
2	Archivo de sistema, será excluido de las búsquedas normales.
3	Etiqueta de volumen, solo existe en el directorio raíz.
4	Directorio, excluido de las búsquedas normales.
5	Bit de archivo, encendido cuando un archivo es modificado
6	Reservado
7	Reservado

Tabla 2.

- Nota 3. El campo de la hora codifica su información de la siguiente manera. Tabla 3.

Bits	*Contenido*
00H-04H	Número de segundos en incrementos de 2 segundos (0-29, que corresponden a 0-58
05H-0AH	Número de minutos (0-59)
0BH-0FH	Número de horas (0-23)

Tabla 3.

- Nota 4. El campo de la fecha es codificado de la siguiente manera. Tabla 4.

Bits	*Contenido*
00H-04H	Día del mes (1-31)
05H-08H	Mes (1-12)
09H-0FH	Año (relativo a 1980)

Tabla 4.

- Nota 5. El tamaño del archivo es almacenado en cuatro bytes, los primeros dos corresponden a la parte baja y los dos finales a la parte alta.

4. El área de archivos. La última sección de organización del disco duro es el área de archivos, está inmediatamente después del directorio raíz, esta zona es donde se encuentran los grupos de sectores que forman los archivos, cada grupo de sectores obtiene un valor determinado por su estado. Este estado puede se de ocupado, vacío, inválido (generalmente por defectos en el medio), y de fin de archivo.

APENDICE B

INSTALACION DE LA RED

Para la instalación de la red, se han desarrollado dos programas. El primero de ellos se encarga de leer el contenido del Bloque de Parámetros del BIOS de cada computadora. Su objetivo es reconocer la estructura del disco duro, obteniendo los parámetros de este, como: el número de cabezas, el número de sectores, el número de pistas, el tamaño de los sectores, el tamaño de los clusters, y en que posiciones se encuentran localizados el directorio raíz y las FATs. Los parámetros obtenidos se almacenan en un archivo especificado por el usuario. Este programa debe ser ejecutado en todas y cada una de la computadoras que vayan a ser integradas a la red, debido a que para ser reconocidas y poder ser accesadas las demás computadoras deben de tener conocimiento de la estructura de su disco duro. El nombre de este programa es BOOT.EXE

Una vez que se ha leído y almacenado la información del Bloque de Parámetros del BIOS en un archivo, este es leído por el segundo programa llamado CONFIG.EXE. Este se encarga de pasarle la información a cada manejador de dispositivos y a la rutina de atención de interrupciones la cual controla el flujo de información a través de los puertos seriales. Es decir, si se tienen tres computadoras conectadas en red cada una tiene instalados dos manejadores de dispositivos, y cada uno de estos debe contener información sobre la estructura del disco duro de la computadora a la cual va hacer acceso. Para lograr lo anterior es necesario leer el archivo de Bloque de Parámetros del BIOS correspondiente a esa computadora y escribir la información en el manejador de dispositivos y en la rutina de atención de interrupciones de los puertos seriales.

Ejemplo de instalación:

Supongamos que tenemos tres computadoras llamadas A, B y C. Para instalar la red, como primer paso tenemos que ejecutar el programa BOOT.EXE en cada computadora para generar un archivo con su Bloque de parámetros del BIOS (ARCHA.BPB, ARCHB.BPB y ARCHC.BPB). Una vez que se han generado estos archivos, tendrán que ser copiados en todas las computadoras.

Como segundo paso, ejecutaremos el programa CONFIG.EXE y le daremos como entradas el nombre del archivo que contiene información del Bloque de parámetros del BIOS de la computadora a la cual va ha hacer acceso, el nombre del manejador de dispositivos que va a controlar a esa computadora, y finalmente el nombre de la rutina de atención de interrupciones. Este segundo paso se tiene que repetir para cada manejador de dispositivos, (n-1 veces, donde n es el número total de computadoras conectadas a la red). Una vez realizado esto, la instalación de la red queda concluida, y no será necesario hacer modificaciones a menos que el número de computadoras en la red se incremente.