UNIVERSIDAD DE MONTERREY

DIVISION DE INGENIERIA Y CIENCIAS
NATURALES Y EXACTAS

UNIVERSIDAD
DE MONTERREY

SISTEMA DE COMUNICACION RESIDENTE EN MEMORIA

REPORTE DEL PROGRAMA DE
EVALUACION FINAL

QUE PRESENTA:
RAUL GARCIA RODRIGUEZ

EN OPCION AL TITULO DE
INGENIERO EN COMPUTACION ADMINISTRATIVA
Y DE PRODUCCION

SAN PEDRO GARZA GARCIA, N. L. MAYO 1991

A mis padres.

En realidad, los méritos que pudiera tener este trabajo les pertenecen más a ellos que a mí.

Agradecimientos.

Es muy difícil agradecer en forma pública toda la ayuda a lo largo de mi carrera y en el desarrollo de este trabajo, sin dejar de olvidar algunos nombres, sin embargo , algunos de ellos me es muy difícil olvidar :

A mi asesor : Ing. Emilio Villarreal, por todo su apoyo para la realización de este trabajo, y su comprensión para conmigo.

A mi familia : Por toda su comprensión.

Al Ing. Rubén Marcos M. : Por toda su ayuda para finalizar este trabajo.

En general, gracias a todos mis amigos que de una u otra manera estuvieron pendientes de mi, durante todo el trayecto de mi carrera.

El precio que se paga por seguir una profesión o vocación es el reconocimiento íntimo de su lado feo.

James Baldwin

INDICE

Organización del presente documento.

Debido a la propia naturaleza del sistema realizado, el presente documento fué organizado de tal forma que el lector pueda entender el funcionamiento del sistema en el nivel que él asi lo requiera, para lograr esto, el libro está organizado de la siguiente manera :

En la introducción se expone la importancia que ha venido teniendo en los últimos años el uso de la información, asi como también, el crecimiento generado en la industria de la computación para eficientar los sistemas de información y de comunicación.

En el capítulo I, se explica detalladamente el proyecto realizado, dando a conocer las partes del sistema y su funcionamiento.

El capítulo II muestra un análisis general del software de comunicación existente en el mercado.

Los capítulos III al V muestran los principios bajo los cuales se realizó el diseño y programación del proyecto, exponiendo desde los conceptos fundamentales, hasta el diseño de las rutinas necesarias en cada caso para el funcionamiento del sistema.

Introducción.

- El perfil de una nueva sociedad de información.

La sociedad de información tuvo sus principios en 1956 y 1957, dos años de la década que encarnó el poder industrial norteamericano.

El de 1956 fué un año de prosperidad, de productividad y crecimiento industrial para los norteamericanos, sin embargo, por primera vez en la historia de este pais, los empleados que ocupaban puestos técnicos, administrativos y de oficina sobrepasaban en número a los de la clase obrera. Estados Unidos, la nación industrial, le cedía el paso a una nueva sociedad en la que, por primera en la historia, la gente empezó a trabajar mas que con bienes de capital, con información.

El año siguiente, 1957, marcó el incio de la universalización de la revolución de la información. Los rursos lanzaron el Sputnik, de cual, su importancia estribaba mas como iniciador de la era de las comunicaciones, que como de la era del espacio.

En una sociedad industrial, el recurso estratégico es el capital, sin embargo, en nuestra nueva sociedad el recurso estratégico es la información.

Actualmente vivimos en un época en la que florece la creatividad empresarial, esto se debe, a que las empresas de hoy en día están comprometidas con sigo mismas y con la sociedad a la que pertenecen, a mantenerse activas en un mercado que cada día es mas competido y mas inovador.

La nueva competitividad empresarial, genera cambios en las estratégias de las empresas, lo que viene a producir cambios en la propia estructura de la empresa y en el trabajo de los ejecutivos.

Los ejecutivos de hoy en día, están obligados a buscar y encontrar nuevos caminos para permanecer competitivos. Para lograr esto, deben de buscar la obtención de los recursos críticos con los cuales competir. El recurso "información", dado los cambios tecnológicos y de pensamiento, no pude quedar olvidado, sino por el contrario, aprovechado al máximo en beneficio de los propios interéses que el ejecutivo persigue.

La informática, proceso automatizado de la información, es la heramienta clave para realmente hacer de la información un arma poderosa de competitividad.

Existen sin embargo, muchas opciones para configurar la herramienta de informática que logre optimizar los procesos de información de las empresas, está variedad de opciones pueden llegar a traducirse en verdaderos problemas de confusión para la empresa que desea obtener el equipo que se adapte a sus necesiades.

Debido a lo anterior, fueron creados en las empresas el departamento de informática, el cual carga con la responsabilidad de planear, organizar y mantener los sistemas de información de la empresa, de modo que esta se beneficie.

Situándonos en nuestro país, estos departamentos de informática deben de estar concientes de la inminente apertura de fornteras con los Estados Unidos, y de el grado de competencia que tendrán que afrontar las empresas de las cuales son ellos el departamento clave si aceptamos la importancia que tiene hoy en día la información como recurso crítico.

Debemos pués, revisar las tendencias mundiales que de uno u otro modo afectan el desempeño de nuestro trabajo, aprenidendo a usar de manera adecuada la tecnología en informática, pero sobre todo, aprender a construir nuestra propia tecnología, que se adapte a nuestras necesiades específicas, y aún más, que reduzca el lapso en el cual, podamos exportar a otros paises tecnología en informática, tal y como la importamos hoy en día.

- Comunicación y computadoras : Dos caminos paralelos.

En los 75 años que siguieron a la introducción del teléfono en la sociedad, se estableció una compleja red de sistemas de telecomunicaciones para conectar entre sí los distintos puntos del planeta.

La primera unión entre dispositivos de cómputo y de comunicación ocurrió en 1940, cuando el doctor George Stibitz utilizó líneas telegráficas para enviar datos desde el DartMouth College, en New Hampshire, a una calculadora situada en Bell Laboratories en la ciudad de Neva York. Pero no fué sino hasta finales de la década de los 50 cuando se inició en serio la conexión entre la computación y la comunicación.

Una de las primeras aplicaciones a gran escala en los negocios fué el sistema SABRE de reservación de pasajeros que desarrolló American Airlines a finales de la década de 1950 y a principios de la siguiente. Se conectaron cientos de terminales dispersas a un centro de proceso. Desde entonces se han usado cada vez mas las comunicaciones.

Hoy en día, las minis y macrocomputadoras están en constante comunicación con terminales remotas. Además, con los aditamentos apropiados, la mayor parte de las computadoras personales pueden utilizar canales de telecomunicación para enlazarse con sistemas de bancos, universidades, centros de información, etc.

Hace 20 años, la capacidad de cómputo era proporcionada por los proveedores de computadoras, pero las compañías de telecomunicaciones proporcionaban los servicios de comunicación. Las empresas de comunicaciones utilizaban algunos dispositivos de comunicación de mensajes controlados por computadora para mejorar sus servicios y los proveedores de computadoras ofrecían paquetes de comunicación limitados para vender servicios de proceso de datos.

- Enlaces entre computadoras.

Las microcomputadoras permiten a las personas procesar datos en forma individual y las macrocomputadoras contienen miles de millones de bytes de datos que son importantes para una organización. Por ello, los usuarios de micros han buscado la forma de tener acceso a los datos de las macrocomputadoras, extraer la información que necesiten y almacenar estos datos en sus computadoras personales, donde pueden manipularlos y analizarlos mediante programas de aplicación apropiados. Es posible también que los usuarios de computadoras personales tengan archivos con información que puede ser de interés para otras personas, la cual se podría enviar a una base de datos de macrocomputadoras con el fin de que se distribuya más ampliamente. O bien, el usuario de computadora personal quizá deseara enviar datos a la localidad donde se encuentra la macrocomputadora a fin de que se imprima eficientemente o se le procese de alguna otra forma con la ayuda de impresoras de alta velocidad y otros recursos de la máquina central. Los enlaces entre micro y macrocomputadoras consisten en el equipo y los programas de comunicación que permiten a las computadoras personales conectarse con sistemas más grandes con el fin de lograr estos objetivos.

En la actualidad se dispone de tal variedad de productos de enlace que puede provocar confusión. Algunos son tarjetas de circuitos que se insertan en las ranuras de expansión de las computadoras personales. estás tarjetas y los programas correspondientes permiten a las micros "hablar" con los sistemas mas grandes como si fueran temrinales de macrocomputadoras. Estos productos de emulación de terminales permiten transferencias sencillas entre los archivos de las computadoras personales y algunos archivos de aplicación de las macrocomputadoras, pero casi siempre exigen a los usuarios de la computadora personal familiarizarse con los procedimientos de acceso de la macrocomputadora.

Otros productos enlazan íntimamente programas de macrocomputadoras de ciertos proveedores o de casas de programación para computadoras personales. Por ejemplo, un programa de enlace puede permitir a los usuarios de la computadora personal copiar datos escritos para un paquete específico de base de datos de macrocomputadora.

Revisando los puntos anteriores, podemos afirmar, que antes de seleccionar los productos de comunicación para computadoras, es necesario realizar un estudio minusiuoso de las necesidades que se tienen y de las posibles soluciones que se puedan adaptar correctamente para cubrir estos requerimientos.

Capítulo I

Sistema de Comunicación Residente en Memoria.

TSRMITE

1.1.- Objetivos del Proyecto

Objetivo General del Proyecto : Se desarrollará un sistema computacional que provea las herramientas necesarias, para la transmisión de archivos entre computadoras, por medio del puerto serial.

Objetivos Específicos :

- El sistema ha desarrollar, será implantado en forma residente en la memoria de la computadora, de tal forma que permita la transmisión de archvios, mientras la máquina pueda ser utilizada en otras aplicaciones.

- El sistema ha desarrollar permitirá la transmisión de archivos entre computadoras locales.

- Se plasmarán los principios de diseño y operación del sistema, de modo que al finalizarlo, se cuente con la información necesaria para facilitar la segunda etapa del proyecto, la cual consistirá en permitir la comunicación con computadoras remotas a través del uso de modems.

1.2.- Justificación del lenguaje Turbo C.

La creación y puesta en venta del lenguaje C en el mercado, marcó el inicio de una nueva etapa para los programadores, ya que desde este lenguaje es posible diseñar sistemas que puedan interacturar en forma directa con los dispostivos de la máquina, y aunque con el paso del tiempo otros compiladores ofrecen ahora estas caqracterísticas, es desde el lenguaje C donde estas funciones se pueden realizar fácilmente. Como ventaja adicional, el lenguaje C ofrece reducciones en el código de los programas desarrollados en comparación con otros compiladores, acceso y manejo directo de la memoria,y una serie de utilerías estándar, que se pueden accesar fácilmente desde el programa principal. No en balde una gran cantidad de software que circula hoy en día en el mercado, fué programado en lenguaje C, desde aplicaciones pequeñas hasta sistemas operativos.

Para el presente proyecto se decidió utilizar el lenguaje Turbo C de Borland, el cual ofrece una gran cantidad de utilerías, a parte de las estándar, lo que facilitó en gran parte el desarrollo del sistema TSRMITE.

1.3.- Módulos del Sistema TSRMITE.

El sistema desarrollado lleva por nombre TSRMITE y cuenta con los siguientes módulos :

Módulo principal : Este módulo es el que se carga desde el sistrema operativo y contiene los siguientes módulos. Al instalarse permanece residente en la memoria de la computadora y es activado mediante la digitalización de una combinación de teclas. Desde este módulo se pueden llamar a los módulos de transmisión y recepción según lo requiera el usuario.

Al activarse este módulo se suspende la ejecución de la aplicación actual. Adémas, guarda el estado de la pantalla (la cual deberá estar en modo texto) y presenta un menú de opciones. Para volver a la aplicación original, se deberá ejecutar la opción de salir del sistema, aun y aunque cualquiera de los demás módulos haya sido llamado.

Módulo de transmisión : Este módulo es llamado desde el módulo principal y tiene como finalidad establecer la comunicación con la computadora destino para la transmisión de archivos. Por medio de un menú de opciones, desde este módulo es posible comenzar una nueva transmisión, o bien, cancelar una que este sucediendo. Este módulo no podrá ser activado si el módulo de recepción aún se encuentra trabajando. En caso de que la transmisión haya empezado sin problemas, el usuario podrá salirse del sistema a ejecutrar otra aplicación mientras la transmisión aún esta sucediendo.

Módulo de recepción : Este módulo es llamaado desde el módulo principal y tiene como finalidad permitir la recepción de archivos desde la computadora destino. Desde este módulo es posible comenzar una transmisión o bien cancelar la que este ocurriendo en ese momento. Este módulo no podrá ser ejecutado si el módulo de transmisión aún este activo. Al comenzar la comunicación, el usuario podrá salir del sistema a ejecutar otras aplicaiones, mientras la transmisión es llevada a término.

1.4 .- Filosofía del diseño de TSRMITE.

Para diseñar un sistema TSRMITE, fué necesario implementar una serie de menajadores de interrupciones de la máquina con el objetivo de lograr que el sistema permaneciera residente en memoria.

Una vez instalado en la memoria, cuando el sistema es activado y la transmisión es iniciada, los módulos de transmisión y recepción, instalan un manejador de interrupciones del reloj para permitir la transmisión de lsoa rchivos en forma **background**. Los detalles del diseño y la implementación de estas interrupciones, serán explicadas con mas detlle en los siguientes capítulos del presente documento.

1.5.- Alcances y limitaciones del sitema.

El sistema desarrollado permite la transmisión de archivos tanto en forma texto como binarios. Esta diseñado para realizar la transmisión o recepción, solo entre dos computadoras locales, enlazadas a través del puerto serial, mediante un cable.

1.7.- Requermientos del Sistema.

Para lograr el correcto funcionamiento del sistema TSRMITE, es necesario tomar en cuenta los siguientes requerimientos.

Requerimientos de Hardware.

(2) Computadoras IBM o compatibles módelo XT o superior, con 640k de memoria principal como mínimo.

(2) Puertos seriales, instalados correctamente en cada una de las computadoras.

(1) Cable de a lo menos tres hilos, configurado para comunicación NULL MODEM.

Requerimientos de software.

Las dos máquinas deberán estar corriendo bajo MS-DOS versión 2.0 o mas reciente. El ROM BIOS de las dos computadoras debe tener fecha de desarrollo de 1985 o mas reciente.

En las dos máquinas debe estar instalado el sistema TSRMITE.

CAPITULO II

Análisis del software de comunicación en el mercado.

Según los expertos del mercado de la computación, el problema en las empresas en cuanto a el manejo de la información, no reside tanto en su elaboración sino mas bien en su accesibilidad. De ahí la razón de que en los últimos años se han estado comercializando una serie de paquetes de comunicación, los cuales permitan a las empresas corregir estos problemas.

Muchos programas de comunicación están dediacados a realizar una sola tarea, tal como transmisión de archivos, acceso a macrocomputadoras, y operaciones remotas entre computadoras personales.

La siguiente es una lista de los paquetes de comunicación considerados como los mejores del mercado, en cuanto a desarrollo, versatilidad y facilidad de uso :

- BLAST PC.
- COM-AND.
- Telix.
- Unicom.
- PereLine.
- Crosstalk.
- ProComm Plus.
- MTEZ.
- SmartCom.
- Profesional YAM.
- Mirror III.
- WinComm.
- Crosstalk (Windows).
- HyperAcces/5.

Estos paquetes estan provsitos de herramientas para lograr facilitar la comunicación, tales como : Directorio telefónico, uso de **paswords** , etc.

Por otro lado, permiten la conexión con macrocomputadoras, minis y otras computadoras personales, utlizando diferentes protocolos de comunicación dependiendo del paquete en particular.

- Transmisión **background**.

Las nuevas versiones de estos paquetes, empiezan ya a tener opciones que permitan las operación **background**, es decir, a permitir la transferencia de información mientras el usuario pueda ejecutar otras aplicaciones. Sin embargo, muchas de esas versiones apenas estan llegando a los usuarios y la mayoría aún no han sido probados.

CAPITULO III

INTERRUPCIONES EN LA FAMILIA DE PROCESADORES 8086

3.1.- Concepto de interrupción.

Una interrupción en la familia de procesadores 8086 es un evento que temporalmnente suspende la ejecución del programa actual y le da el control a una rutina de servcio para realizar los procesos requeridos.

Una interrupción es una indicación al microprocesador de que se necesita atención inmediata. La familia de microprocesadores pueden responder a interrupciones de hardware y software. Un periférico de hardware puede generar una señal de interrupción, la cual es procesada por el controlador programable de interrupciones (PIC) y enviada al microprocesador; desde software, la instrucción INT (En ensamblador) genera una interrupción. En ambos casos, el microprocesador interrumpe su proceso actual y ejecuta una subrutina residente en memoria llamada manejador de interrupción. Después de que el manejador de interrupción ha realizado su tarea, el microprocesador regresa a seguir con su proceso en el punto donde se quedó cuando la interrupción fué generada.

La familia de procesadores 8086 soportan 256 interrupciones, cada una de ellas identificada por un número entre 00Fh y FFh (255 en decimal). La dirección de los 256 manejadores de interrupciones son almacenados en una tabla de vectores de interrupción que comienza en la dirección 0000:0000h (esto es, en el comienzo de la memoria disponible). Cada vector de interrupción tiene un tamaño de 4 bytes, asi que para localizar la dirección de cualquier manejador de interrupciones es necesario multiplicar el número del manejador por cuatro. Es posible reemplazar un manejador de interupción existente por uno nuevo, almacenando la nueva dirección del manejador en el vector de interrupción apropiado.

3.2.- Vectores de interrupción.

Como regla general, las interrupciones de la familia de las PC pueden ser divididas en seis categorías :

1) Interrupciones del microprocesador.
2) Interrupciones de Hardware.
3) Interrupciones de Software.
4) Interrupciones del DOS.
5) Interrupciones de BASIC.
6) Interrupciones de uso de general.

1) Interrupciones del microprocesador: También llamadas interrupciones lógicas, fueron diseñadas para el funcionamiento mismo del microprocesador. Cuatro de ellas (interrupciones 00h, 01h, 03h, 04h) son generadas por el propio microprocesador y otra (002h) es activada por una señal generada por ciertos periféricos de hardware, tal como el coprocesador matemático 8087.

2) Interrupciones de Hardware : Son construidas en el hardware de la computadora. En las computadoras PC, XT y PS/2 modelos 25 y 30, son usados los números de interrupción que van desde el 08h al 0Fh. En las computadoras AT y PS/2 modelos 50, 60 y 80, son usados los números de interrupción del 08h al 0Fh y del 70h al 77h.

Cada una de las interrupciones de hardware está asociada con un periférico en particular. Cada periférico que necesite de la atención del procesador envía una "interrupción de petición", IRQ (Interrupt Request), hacia el controlador de interrupciones 8259A, el cual organiza las interrupciones para darles servicio. Cuando el controlador decide que una interrupción puede ser atendida, este manda una "interrupción de reconocimiento" hacia el periférico, deshabilita todas las interrupciones, y genera una interrupción.

En respuesta de una interrución de hardware en particular, el procesador busca la dirección de la rutina de servicio de la interrupción en la tabla de vectores de interrupción (IVT, Interrupt Vector Table). Esta tabla ocupa las primeras 256 palabras-dobles (1024 bytes) de la memoria. Cada registro contiene la dirección de una rutina de servicio de interrupción (ISR, Interrupt Service Routine). El procesador salva el estado de los registros actual y empieza la ejecución de la ISR.

La ISR realiza la tarea requerida para servir la interrupción. En algún momento, la ISR envía un mensaje de final de interrupción (EOI, End Of Interrupt) hacia el controlador 8259, indicando que el procesador está listo para aceptar otra petición de servicio. EL controlador de interrupciones no reconocerá ninguna interrupción de ningun periférico, independientemente de su nivel de prioridad, hasta que no reciba el EOI. Después de que la ISR termina su tarea ejecuta una instrucción de IRET(Interrupt RETurn), la cual restaura los registros del procesador.

3) Interrupciones de Software : Incorporadas en el diseño de la PC, son parte de los programas del ROM BIOS (Read Only Memory, Basic Input Output System). Las rutinas del ROM BIOS son invocadas por estas interrupciones que no pueden ser modificadas, pero los vectores que apuntan a estas pueden ser cambiados de tal forma que apunten hacia diferentes rutinas. Los números de interupción son del 10h a la 1Fh y del 40h al 5Fh.

La instrucción INT de los procesadores 80x8x provee el mecanismo de las interrupciones de software. El procesador trata a las interrupciones de hardware y software de la misma manera. La ejecución de una instrucción INT causa la transferencia del control hacia la ISR especificada en el operando de la instrucción. Por ejemplo, la interrupción 60h invoca a la ISR que está alamcenada en la dirección 180h (4*60h) de la tabla de vectores de interrupción. El controlador de interrupciones no es involucrado, y por lo tanto, el software incluído en la ISR no tiene que enviar un EOI hacia el controlador. El DOS usa una vasta variedad de interrupciones de software.

4) Interrupciones del DOS : Siempre presentes cuando es usado el DOS, muchos programas y lenguajes de programación usan los servicios provistos por DOS a través de sus interrupciones, para el manejo de operaciones básicas, especialmente operaciones de entrada y salida en discos. Los números de interrupciónes del DOS van del 20h al 3Fh.

5) Interrupciones del BASIC : Son asignadas por el lenguaje BASIC y se encuentran disponibles simpre que BASIC esté usándose. Los números de interrupción son 80h al F0h.

6) Interrupciones de uso general : Están disponibles para usarse temporalmente por los programadores. Los números de interrupción son el 60h al 66h.

La mayoría de los vectores de interrupción usados por el ROM BIOS, DOS y BASIC contienen la dirección del manejador de interrupción correspondiente. Sin embargo algunos vectores de interrupción apuntan hacia tablas de información. Por ejemplo, la interrupción 1Eh contiene la dirección de una tabla de parámetros de inicialización del drive; el vector de interrupción 1Fh apunta hacia la tabla de parámetros de bits usados por el ROM BIOS para desplegar caracteres de texto; y la interrupción 41h y 46h apuntan a la tabla de parámetros del disco duro. Estos vectores de interrupción son usados por conveniencia, no para propósitos de interrupciones.

Ejemplos de interrupciones de Hardware.

La PC usa un canal del chip counter/timer 8253 para hacer una petición de interrupción 18.2 veces por segundo. El controlador 8259A genera la interrupción 8h en respuesta de este requerimiento. Esta interrupción de reloj tiene el nivel de prioridad mas alto y por lo tanto su ejecución precedrá a cualquier otra mientras que el procesador no haya deshabilitado todas las interrupciones mediante la instrucción CLI (CLear Interrupts).

Es el ROM BIOS el que normalmente atiende este requerimiento. Después de actualizar la hora y el día y realizar otras tareas, el ROM-BIOS ejecuta la instrucción INT 1ch, la cual solo tiene como objetivo, ejecutar la instrucción IRET.

Otro ejemplo de una interrupción de hardware es la que es generada por el teclado. Cada vez que el usuario presiona una tecla, el teclado causa una interrupción que cede el control hacia la rutina almacenada en la dirección 09h de la tabla del vector de interrupciones. La rutina de servicio para esta interrupción, almacena la tecla presionada y realiza otra tarea del teclado.

La figura III.1 muestra las interrupcions principales y sus vectores de localización. Estas son las interrupciones más usadas por los programadores.

FIGURA 3.1

INTERRUPCION	DIRECCION	USO
00H	0000	GENERADA POR EL CPU CUANDO REALIZA UNA DIVISION ENTRE CERO
01H	0004	USADA PARA IRSE PASO POR PASO EN UN PROGRAMA (COMO EL DEBUG)
02H	0008	INTERRUPCION NO DOCUMENTADA
03H	000C	USADA PARA PONER BREAK POINS EN PROGRAMAS
04H	0010	GENERADA CUNADO UN RESULTADO ARITMETICO ES OVERFLOW
05H	0014	INVOCA LA RUTINA DE IMPRIMIR PANTALLA DEL ROM BIOS
08H	0020	GENERADA POR EL RELOJ DE HARDWARE
09H	0024	GENERADA POR UNA ACCION DEL TECLADO
0EH	0038	SEÑAL DE ATENCION DEL DISKET
0FH	003C	USADA EN CONTROL DE IMPRECION
10H	0040	INVOCA UN DESPLIEGE DEL VIDEO DELROM BIOS
11H	0044	INVOCA LA LISTA DE EQUIPO DEL ROM BIOS
12H	0048	INVOCA EL TAMAÑO DE MEMORIA DEL ROM BIOS
13H	004C	INVOCA UN DISCO DEL ROM BIOS
14H	0050	INVOCA COMUNICACION DEL ROM BIOS
15H	0054	INVOCA SERVICIOS DEL SISTEMA DEL ROM BIOS
16H	0058	INVOCA SERVICIOS ESTANDARES DEL TECLADO DEL ROM BIOS

INTERRUPCION	DIRECCION	USO
17H	005C	INVOCA SERVICIOS DE IMPRESION DEL ROM BIOS
18H	0060	ACTIVA EL LENGUAJE ROM BASIC
19H	0064	INVOCA LA RUTINA DE COMIENZO DEL ROM BIOS
1AH	0068	INVOCA LOS SERVICIOS DE TIEMPO Y FECHA DEL ROM BIOS
1BH	006C	INTERRUPCION DEL ROM BIOS POR EL CTRL-BREAK
1CH	0070	INTERRUPCION GENERADA POR CADA PULSO DEL RELOJ
1DH	0074	APUNTAN A TABLAS DE LOS PARAMETROS DEL CONTROL DE VIDEO
1EH	0078	APUNTA A LOS PARAMETROS DE LAS TABLAS DE LOS DISKETS

INTERRUPCIONES MAS IMPORTANTES USADAS EN LAS COMPUTADORAS PERSONALES DE LA FAMILIA IBM

3.3.- Cambios en los vectores de interrupciones.

El mayor interés de la programación en cuanto a los vectores de interrupción no reside en leer éstos, sino más bien en modificarlos de tal forma que apunten hacia una nueva rutina manejadora de interrupciones. Para hacer esto se debe escribir una rutina que realice una función diferente de las que desarrollan los manejadores de interrupciones del DOS y el ROM BIOS, almacenar la rutina en RAM y por último asignar la dirección de la rutina en una interrupción existente en la tabla.

Un vector de interrupción puede ser modificado byte por byte en lenguaje ensamblador o usando instrucciones de otro lenguaje de programación como la instrucción POKE en BASIC, en algunos casos, esto puede ser peligroso si ocurre una interrupción cuando el cambio en el vector no ha sido terminado. Existen dos métodos para modificar un vector minimizando el riesgo de que otra interrupción ocurra mientras se está haciendo el cambio : Suspendiendo las interrupciones durtante el proceso, o usando una interrupción del DOS especialmente diseñada para cambiar vectores.

El primer método requiere el uso del lenguaje ensamblador para suspender las interrupciones mientras se modifica el vector de interrupción. Se puede usar la instrucción CLI (Clear Interrupts), la cual suspende todas las interrupciones hasta que sea ejecutada la instrucción STI (Set Interrupts). Deshabilitando las interrupciones temporalmente, con CLI, se asegura que no pueda ocurrir ninguna interrupción mientras se actualiza un vector de interrupción.

El segundo método para actualizar un vector interrupción consiste en usar la interrupciones del DOS (21h), específicamente la función 25h, la cual fué diseñada para este propósito. Existen dos ventajas muy importantes para usar las interrupciones del DOS. Una ventaja es que el DOS se toma la tarea de almacenar el vector en su lugar de la manera mas segura posible. La otra ventaja es mas difícil de asimilar. Usar las funciones del DOS para actualizar un vector de interrupción en lugar de hacerlo directamente es el único método que puede reducir el riesgo de que un programa sea incompatible con nuevas máquinas o nuevos ambientes del Sistema Operativo.

3.4.- Diseño del manejador de interrupciones desde Turbo C.

El mecanismo de interrupción es uno de los más importantes recursos en una máquina trabajando bajo DOS. existen dos maneras en las cuales los programas pueden ser beneficiados por éstos mecanismos. Primero, el hardware constantemente genera interrupciones que permiten a los programas desarrollar tareas subsecuentes a la interrupción que se está generando. Segundo, el Sistema Operativo invoca una gran cantidad de interrupciones de software que permiten instalar manejadores de interrupciones para salidas por teclas-clave (break-key) y condiciones de error. Los manejadores de interrupción son tradicionalmente escritos en lenguaje ensamblador, sin embargo, usando algunas de las características avanzadas de TURBO C es posible escribir óptimos manejadores de interrupciones que realicen diversas tareas.

Pese a las ventajas de utilizar los servicios del DOS para instalar los maneadores de interrpción, los realizados en el desarrollo del sistema TSRMITE están realizados directamente, es decir, sin utilizar este servicio. Las razones para hacerlo de esta manera, consisten en que los manejadores de interrupción del sistema realizado, o bien fueron diseñados para generar el programa residente en memoria (Ver Capítulo IV), o son instalados desde este tipo de programas. En el Capítulo IV, se detallará sobre los riesgos de utilizar interrupciones del DOS, desde un progrma residente en memoria. La segunda razón es que el lenguaje Turbo C, provee las herramientas suficientes para suspender todas las interrupciones de la máquina, mientras el manejador está siendo instalado.

Como ya se dijo en el párrfo anterior, el sistema TSRMITE, instala una serie de manejadores de interrupción. Cabe mencionar, que las rutinas de instalación de menajadores de interrupción fueron desarroladas fuera del programa principal del sistema y son llamadas cuando se requiere su uso, en forma de librerías.

En la figura III.2 se puede analizar la rutina en pseudocódigo C, utilizada para instalar manejadores de interrupción.

```
Instala-M ( * Rutina-Usuario)
{

    Rutina-Original = getvect (INT);

    setvect = (INT, Rutina-Nueva);
}

Desinstala-M ()
{

    setvect(INT, Rutina-Original);

 }

Rutina-Nueva ()
{

   (* Rutina-Original);

    (* Rutina-Usuario);

}
```

FIGURA III.2 Manejador de interrupción.

Este pseudocódigo, consta de tres rutinas, las cuales realizan sus propias actividades, tal y como se describen a continuación.

Instala-M : Esta rutina es llamada desde el código principal, para instalar el nuevo manejador de interrupción. LLeva como parámetro único, la dirección de la rutina de usuario a instalar en el nuevo majneador. Realiza las siguientes actividades :

- Salva, en una variable especial, la dirección de la rutina del manejador original, mediante la función getvect, esta última lleva como parámetro el número de interrupción requerida (INT).

- Instala en el vector de interrupción a Rutina-Nueva, mediante la función setvect, la cual lleva como parámetros; el número de interrupción para la cual debe intalar el manejador, y la rutina a instalar, en este caso Rutina-Nueva.

Desinstala-M : Esta rutina es llamada desde el código principal, con el objetivo de desinstalar la rutina de usuario en el vector de interrupción adecuada, e instalar la rutina del manejador original. El uso de esta rutina es de vital importancia, ya que es indispensable reinstalar las rutinas originales de los manejadores de interrupción, cuando ya no sean requeridos, o bien al finalizar la ejecución del programa, ya que el no hacerlo puede causar el mal funcionamiento de nuevas aplicaciones y del sistema operativo mismo.

Rutina-Nueva : Esta rutina es llamada por la propia librería, para ser instalada como el nuevo manejador de una interrupción específica. Realiza dos funciones : Primero, ejecuta la rutina original de la interrupción, es muy importante que esto se realice debido a que el usuario debe de permitir a la máquina hacer las actividades correspondientes a la interrupción, de otro modo se causarían graves probleamas en el buen fucnionamiento del sistema operativo. Por último, se ejecuta la rutina diseñada por el usuario y regresa hacia donde fué llamada.

Como ya se mencionó anteriormente, estas rutinas fueron diseñadas dentro de una sola rutina de librerías, para ser llamadas desde los programas de aplicación.

La figura III.3 muestra el pseudocódigo-C, de una aplicación, la cual utiliza esta librería.

La filosofía general de trabajo del programa de aplicación es la siguientes :

1 - Se realizan las inicializaciones pertinentes de acuerdo a la aplicación.

2 - Se instala una rutina de usuario como manejador de una interrupción en particular.

3 - Se realizan procesos subsecuentes de la aplicación.

```
main ()
{

    Inicia-1 ();

    Instala-M (Mi-Rutina) ;

    proceso-1 ();

    Desinstala-M ();
}

Mi-Rutina ();
{

    disable ();

    condiciones ();

    enable ();

    aplicacion();

    return ;
}
```

FIGURA III.3 Aplicación de un manejador de interrupción.

4 - Se desinstala la rutina de usuario y se resatablece la original antes de terminar el programa.

La rutina de usuario realiza los siguientes procesos :

1 - Se desactivan todas las interrupciones de la máquina.

2 - Se condiciona la ejecución de la rutina al valor de ciertos parámetros, dependiendo de la aplicación diseñada. Estas condiciones son de dos tipos :

De semáforo : El uso de este tipo de condición es indispensable, ya que evita que se llame de nuevo a la rutina antes de que esta termine su ejecución.

Da aplicación : Este tipo de condiciones, son diseñadas de acuerdo a la aplicación requerida, ya que es posible que el usuario no desee que se ejecute la rutina, estrictamente el número de veces que sucede la interrupción.

3 - Se activan todas las interrupciones.

4 - Se realiza la aplicación correspondiente.

5 - Regresa el control de ejecución a la rutina principal después de desactivar los semáforos de condición.

Las rutinas de manejadores de interrupción programadas en el sistema TSRMITE trabajan de acuerdo a lo que se ha descrito anteriormente, distribuyendo estas, en las diferente librerías del sistema, dependiendo de los requerimientos propios del sistema.

CAPITULO IV

Programas Residentes en Menoria.
(Terminate and Stay Resident Programs).

4.1.- Introducción a los programas residentes en memoria.

En el simple concepto aparente, la creación de un programa residente en memoria (TSR), es actualmente una de las mas difíciles tareas en el ambiente de las computadoras personales. Los riesgos sin embargo, están balanceados con el profesionalismo de los resultados que se pueden obtener al utilizar este tipo de recursos.

Por naturaleza propia, un progrma TSR está íntimamente relacionado con el hardware y el sistema operativo en el cual se ha diseñado, por lo siguiente, todo el material expuesto en este documento en cuanto programas TSR se refiere, es aplicado específicamente a la línea de computadoras personales IBM y comptaibles, corriendo bajo el sistema operativo DOS. Aún mas, el diseño del modulo TSR del sistema TSRMITE, está especificado para programarse en Turbo C, aunque se prodría adecuar para otros compiladores.

Intrínseco al desarrollo y uso de un programa TSR es la modificación de la tabla de vectores de interrupción (Ver Capítulo III), por lo que al programar dicha subrutina, se tiene que tener un grado óptimo de precaución, ya que un error, podría degenerar en una serie de problemas graves en la información almacenada en el disco duro de la máquina.

Todos los programas TSR empiezan su ciclo de vida como cualquier programa ordinario. Cuando el programa termina, deja una parte de su código en la memoria. El código que se ejecuta primero es llamado "código de inicialización", y el que se deja en memoria es conocido como "código residente". La tarea principal del código de inicialización consiste en preparar el código residente para su uso posterior. No existen restricciones a cerca de lo que pueda hacer el código de inicialización, pero la programación del código residente puede ser problemática.

Los programas TSR's pueden ser agrupados en tres categorías. Los miembros del primer grupo no tienen interfase con el usuario con su parte residente. una vez cargados, éstos programas permanecen residentes en memoria, realizando su tarea sin hacer cualquier tipo de petición al BIOS. El programa ASSIGN, utilería del DOS, es uno de estos, su parte residente monitorea y redirecciona los requerimientos de disco de un drive hacia otro.

Los miembros del segundo grupo, dejan su parte residente en memoria sin hacer ninguna tarea hasta que sean actvados a petición del usuario. Normalmente esta petición se realiza a través de una tecla o de la combinación de varias, esta combinación es conocida con el nombre de "Hot Key". Nuevamente, el código residente no hace ningún tipo de petición al BIOS, pueden obtener sin embargo cualquier servicio del DOS, como leer un archivo, solo durante la inicialización.

Una pequeña base de datos de una guía telefónica, puede caer en esta categoría. El código de inicialización podría leer los registros de una archivo de teléfonos en la memoria. En respuesta a la digitalización de cierta combinación de teclas, el código residente podría salvar la pantalla actual, pedirá al usuario por un o mas nombres, y desplegar el número telefónico. Cuando no haya mas nombres que buscar, el TSR podría restaurar la pantalla original y desactivarse a sí mismo.

El grupo final de programas TSR's son los que realizan peticiones asincrónicas al BIOS. Estas peticiones puden ser accesadas mediante la digitalización de un Hot Key, o bien mediante alguna otra interrupción de hardware. (Por ejemplo el Pulsador de Tiempos). El código residente no necesariamente tendrá interfases con el usuario. El Programa PRINT, utilería del DOS, cae dentro de esta categoría. Este tipo de programas, son difíciles de escribir debido a que el DOS básicamente es un sistema operativo para un solo programa y un solo usuario. Muchos de los servicios necesarios para la creación de este tipo de programas están indocumentados y se requiere un conocimiento profundo sobre el funcionamiento del sistema operativo.

4.2.- Principios de un programa TSR.

Un programa TSR, se desarrolla utilizando la interrupción 21h del DOS, función 31h, la cual causa que el programa regrese el control al sistema operativo, mientras que premanece en una porción de la memoria que no es usada por el DOS, de esta manera, el programa puede ser instantáneamente activado sin tener que ser cargado de nuevo. Uno de los ejemplos mas conocidos de programas TSR es el SideKick de Borland.

La mayoría de los programas TSR son activados a través de una interrupción, la cual puede ser generada de muchas maneras. Las interrupciones mas comunes son : interupción de <Imprime Pantalla>, de reloj, o el telcado. Para los programas TSR, que despliegan en pantalla una serie de opciones de menú, las interupciones mas comunmente usadas son las de teclado y de <Imprime Pantalla>, ya que permiten al usuario activar el programa oprimiendo una secuencia clave de teclas (Hot Key).

4.3.- Interupciones. DOS vs BIOS.

Los progrmadores frecuentemente contemplan al DOS como un sistema operativo no recursivo, esto es, mientras está siendo usado por un programa, el DOS no puede ser usado por un segundo programa. (esto explica por qué el DOS no permite multitasking).

El BIOS (Basic Input Output System) instalado en el ROM de las PC's permite cierta recursividad. Por ejemplo, la interupción 16h (entrada por teclado), puede ser usada en programas TSR, sin efectos secundarios. La única manera de asegurar que el uso de otra interupción no causa ningún efecto anormal, es la experimentación. Si una función no puede ser usada, fácilmente podrá detectarse ya que la "caída del sistema" será inminente, en la mayoría de los casos.

4.4.- Diseño general de un programa TSR.

Tal y como cualquier programa de el DOS, las aplicaciones TSR pueden ser archivos .COM o bien .EXE. El sistema operativo carga todos los programas de la misma manera, Cada programa cuenta con su propio PSP (Program Segment Prefix), su código y datos. La diferencia entre un TSR y una aplicación estándar reside en que el TSR tiene que realizar una serie de tareas para prepararse así mismo con el fin de reactivarse tiempo después.

Para convertir el sistema TSRMITE en un programa TSR, se siguió la misma filosofía observada en el diseño de los manejadores de interrupción, es decir, se programó un código en Turbo C, el cual contiene dos rutinas para el manejo de aplicaciones TSR, este código está fuera del programa principal del sistema y es utlizado por éste en forma de librería.

La librería cuenta con las funciones siguientes :

TSRInstall y TSRInDos.

La función TSRInstall tiene el siguiente formato :

TSRInstall (*FUNCION, HOTKEY, CODIGO)

la cual realiza las siguientes actividades :

1 - Checa para asegurase que el programa aún no se ha instalado en la memoria. Cada programa instalado por esta rutina debe ser identificado por un valor único designado en el parámetro CODIGO.

2 - Instala los manejadores de interrupción apropiados de manera que cuando el usuario oprima la combinación de teclas especificada en el parámetro HOTKEY, la función dada por el parámetro FUNCION reciba el control inmediato.

3 - Termina el programa, dejando el código de la función indicada residente en la memoria.

Una vez que la función TSRInstall termina en el programa principal, la aplicación instalada no debe de regresar el control, si esto sucede es debido a que a ocurrido un error en el sistema y el programa no será instalado en la memoria. En este caso, una de los siguientes códigos de error es regresado al programa principal:

INSTALADO : Este código de error indica que un TSR contiene el mismo código de identificación que uno que ya a sido instalado.

NOINT : Este valor significa que no existen vectores de interrupción de usuario libres. Los vectores de interrupción 60h al 67h, son designados disponibles para los porgramas de usuarios. El primero de esos vectores que se encuentre libre será usado para contener el código de identificación de un TSR. Si todos los vectores están ocupados, será regresado un código de NOINT.

ERRORDOS : la función que instala las aplicaciones TSR requiere que se este usando el DOS versión 2.0 o una más actualizada. Si la versión es menor que 2.0 se regresará este código de error.

ERROR : Este código de error es regresado si el proceso de instalación falla, debido a que la aplicación TSR regresa el control al programa principal y ninguno de los errores anteriores ha sido detectado.

La función TsrInDos tiene el siguiente formato :

int TsrInDos (void)

Esta función regresa un valor diferente de cero si cualquier servicio del MS-DOS está activo cuando la aplicación TSR recibe el control, y regresa un cero si el MS-DOS no está activo. El propósito de esta interrupción es determinar cuando las funciones del MS-DOS que se encuantran en el sistema pueden ser llamadas. Específicamente, si TSRInDos regresa un valor de 0 (cero), el sistema puede llamar directamente cualquier servicio del MS-DOS o bien llamar a cualquier función de librería que invoque los servicios del DOS. Las funciones de librería de Turbo C, que emplean estos servicos están definidas en el encabezado **conio.h** .

En muchos aspectos, un programa destinado a ser TSR puede ser escrito como cualquier otro. Sin embargo, existen tres reglas especiales que se deben de tomar en cuenta en el desarrollo de programas residentes en memoria.

La primera regla es que el programa debe de ser compilado usando los modelos de memoria: **small**, **medium** o **tiny**. El módulo está diseñado para trabajar con programas que usen modelos de memoria pequeños; si se convierte el módulo para trabajar con un modelo de memoria **large** es muy posible que suceda algún problema.

La segunda regla es que cuando el sistema escriba sobre la pantalla, se debe de guardar y restaurar el estado completo del video. Las funciones en el módulo TSR atomáticamente guardan y restauran el estado total del programa suspendido. Pero el trabajo de interactuar con la pantalla es dejado a la aplicación.

Para lograr esto, es preferible que se utilicen rutinas especiales que pueden ser llamadas por la propia aplicación, estas rutinas cubren tareas importantes como la de mantener un stack de pantallas, para facilitar el guardado y restauración de las mismas.

La tercera regla es que la aplicación TSR debe siempre regresar el control al finalizar su trabajo, mediante la instrucción **return** hacia la rutina donde fué instalado. El sistema no puede usar instrucciones **exit** para terminar un programa. La razón es que esta instrucción fué implementada para terminar un programa no-residente y regresar al DOS. Si se usara esta función se abortaría tanto la función TSR como la aplicación que fué suspendida antes de que se cediera el control.

4.5.- Estrategia general de las rutinas TSR.

La tarea fundamental requerida para instalar un TSR es cargar el programa en un área de memoria protegida en la cual no puedan usar otras aplicaciones. Afortunadamente el DOS provee un servicio que permite a un programa terminar su ejecución protegiendo la memoria en la que quedó almacenado. El MS-DOS reserva esta área de memoria removiendo a esta de la lista de bloques de memoria libres. El módulo TSR accesa este servicio del DOS a través de una función de librería de Turbo C llamada **keep**. La única complicación al usar esta función es que se debe especificar en uno de sus parámetros, el tamaño de memoria que se necesita para alocarlo en memoria principal. Los detalles de cómo se resolvió este problema serán aclarados más adelante.

La siguiente tarea requerida para crear un TSR es establecer ciertos puntos para permitir la activación del programa mediante una combinación de teclas (Hot Key). La estretegia seguida para lograr esto, fué el remplazo del manejador de interrupción original del teclado con una función de interrupción de Turbo C.

Cada vez que una tecla es presionada, la función de interrupción recibe el control; esta función primero llama a la función original, y luego checa si la combinación de teclas presinada corresponde a la declarada para activar la rutina TSR.

Si la combinación previamente establecida es presionada, la rutina desarrolla otros chequeos y dependiendo del resultado de éstos últimos se decidirá si se ejecuta la aplicación TSR.

Otro importante aspecto a tomar en consideración, es evitar la activación de el programa residente en el momento en que una interrupción está en proceso. El momento mas peligroso para activar una rutina TSR, es cuando el BIOS está desarrollando una actividad en el disco. Esto último puede ocasionar que se almacene información erronea en el FAT (File Allocation Table) del disco con las subsecuentes consecuencias graves que puede traer esto.

Para prevenir lo anterior. el módulo TSR, también instala una rutina de manejador de interrupción para la interrupción 13h.

La función Turbo C, creada para remplazar la rutina original de la interrupción 13h, primero "prende" una bandera informando que una función para disco se está realizando, después cede el control a la rutina original de la interrupción y cuando el control es regresado, "apaga" la bandera.

De manera que, antes de que la rutina manejadora del teclado, active la función TSR, checa el estado de la bandera de la rutina de interrupción 13h y regresa el control inmediatamente si el valor de la bandera revela que se está realizando una actividad en el disco.

Por otro lado, una aplicación TSR no puede ser activada cuando un servicio del MS-DOS está activo. El peligro de suspender un servicio del MS-DOS reside en que el propio TSR puede invocar una de las funciones del sistema operativo, lo que se traducirá en una corrupción del stack usado por la invocación original del DOS. En general, un servicio del MS-DOS no pude ser interrumpido en un punto arbitrario y el código del stack usado por otro proceso.

Para atacar este problema, el MS-DOS provee una solución parcial. En cualquier momento en el que un servicio del MS-DOS se active, asigna una valor diferente de cero a una bandera interna; la dirección de esta bandera es regresada por la función 34h del DOS, que a pesar de que no está docuentada es muy usada para el desarrollo de este tipo de aplicaciones.

De manera que en adición de checar una posible actividad del disco el manejador de interrupción del teclado, necesita chacar la bandera que indica el estado de uso actual del DOS, y regresar el control inmediatamente si cualquiera de esas banderas contienen un valor diferente de cero.

Esta técnica para detectar interrupciones del DOS en progreso, tiene sin embargo un problema serio. Mientras el intérprete de comandos (COMMAND.COM) está esperando la entrada de información por parte del usuario, la bandera que indica el uso del DOS contiene el valor de 1, debido a que el DOS usa uno de sus propios servicios para leer caracteres. Mientras el DOS está esperando por algún comando en particular, se está relativamente disponible para activar la aplicación TSR, de hecho, todas las funciones del DOS pueden ser usadas en este momento excepto las que están en el rango de 01h al 0Ch. De modo que la aplicación TSR, necesita una manera de enterarse que aunque la bandera del DOS indica que el sistema operativo está en servicio, probablemente solo está esperando leer caracteres por medio del intérprete de comandos.

Para resolver el problema anterior, se toma ventaja de la siguiente situación : Cuando el DOS espera una orden desde el intérprete de comandos, continuamente invoca la interrupción 28h (la cual es usada para activar procesos **background** tales como un spool para la impresora). De tal suerte, que la simple indicación de la ocurrencia de la interrupoción 28h es suficiente indicación (independientemente del estado de la bandera de servicios del DOS) para determinar que es posible activar la aplicación TSR.

Aplicando este principio en el sistema, el módulo TSR instala un último manejador de interrupción, el cual intercepta la interrupción 28h y provee una segunda puerta para activar la aplicación TSR.

Resumiendo, existen dos mecanismos paralelos para activar la aplicación TSR cuando la combinación de teclas predeterminada ha sucedido. Primero, la aplicación TSR puede ser activada a través de la rutina manejador de interrupción del teclado, la cual , para decidir si activa la aplicación, debe de checar el estado de la bandera que indica la actividad de un servicio de disco y la bandera que indica que una interrupción del DOS está sucediendo. Segundo, la aplicación TSR puede ser activada a través del manejador de interrupción 28h, la cual para decidir si activa la aplicación solo debe de checar la bandera que indica si está sucediendo un servicio de disco.

Se puede determinar el camino por el cual una aplicación TSR fué activada, esto es importante cuando desde la aplicación TSR se requiere de los servicios 01h al 0Ch del DOS, debido a que esas funciones utlizan el mismo stack que el DOS estaba utilizando antes de ser suspendido.

Una vez que el módulo TSR ha determinado que es posible activar la aplicación residente en memoria, la tarea que resta es la de salvar el estado actual del programa interrumpido y llamar a la aplicación TSR que quedó especificada en el momento de la instalación, el estado del programa interrumpido, debe de ser restaurado antes de regresarle el control del procesador.

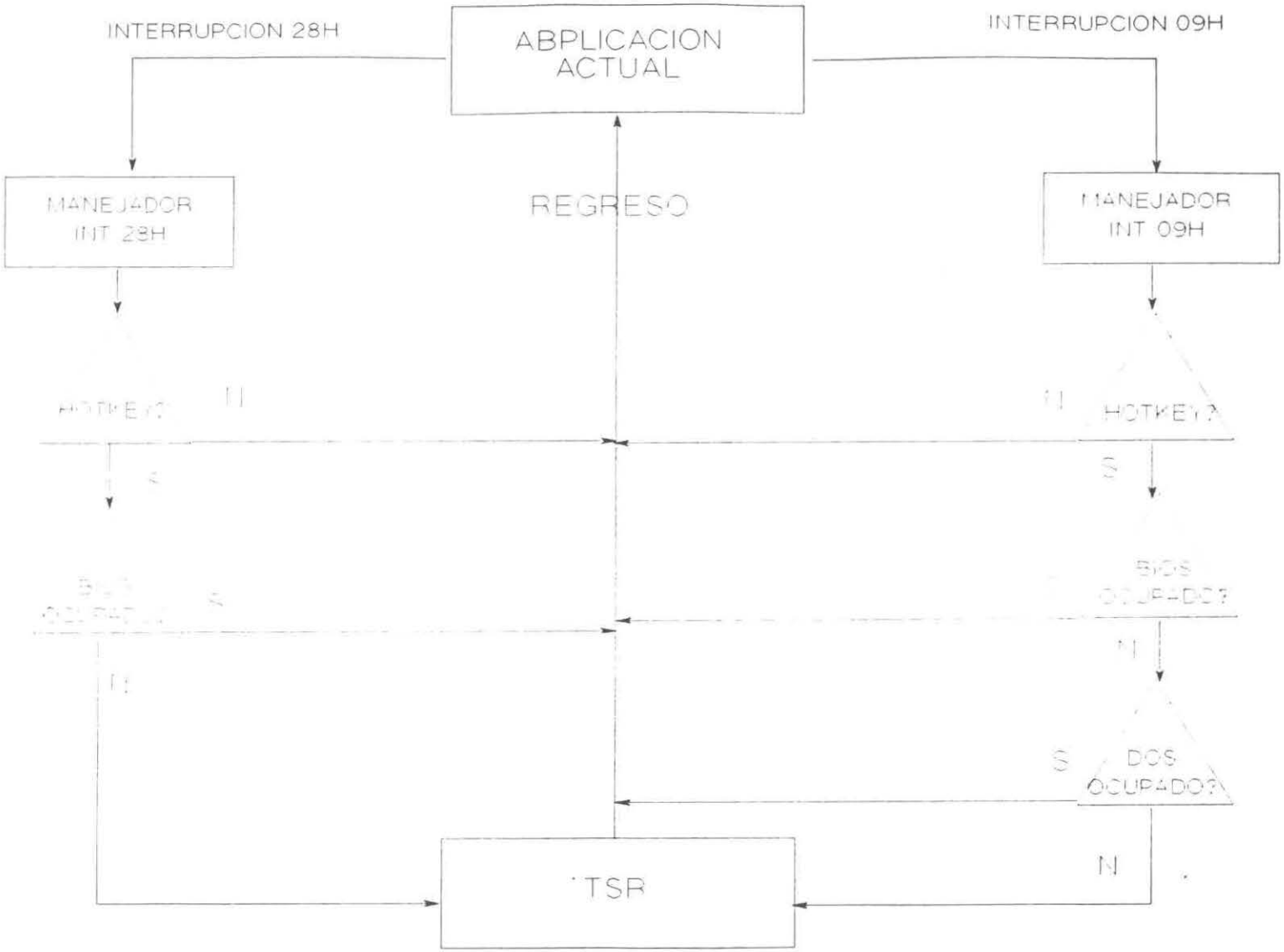

FIGURA IV.1 ACCESO PARALELO A UNA RUTINA TSR

4.6.- Aspectos de implementación.

La función de instalación TSRInstall, empieza por checar si la aplicación TSR ya ha sido instalada en la memoria, buscando a través de las interrupciones de usuario (Números 60h al 67h) el número de identificación contenido en el parámetro Código. Si el código no es encontrado, esto indica que la aplicación aún no ha sido instalada, entonces se procede a localizar el valor del código en el primer vector de interrupción libre (Igual a cero) dentro de este rango. La función de instalación entonces checa que se este trabajando con una versión del DOS 2.0 a mas reciente, debido a que las versiones anteriores no soportan las funciones requeridas para instalar un programa residente en memoria.

Si todo va bien hasta este punto, la función de instalación guarda la siguiente información importante, la cual será requerida cuando la aplicación TSR sea activada :

. La dirección de la apicación TSR es guardada en el contenido de un apuntador.

. Los valores del segmento actual de la dirección de transferencia de disco (DTA) son salvados, de manera que puedna ser restaurados después de que la aplicación TSR reciba el control.

. La dirección de la bandera que indiica si está sucediendo un servicio del DOS es obtenida realizando el servicio 34h de la interrupción del DOS (21h), y es guradada en un apuntador.

. Finalmente, la función de instalación salva la dirección de las funciones de interrupción originales para que los tres manejadores de interrupción diseñados puedan ser instalados.

Una vez que toda la información crítica ha sido guardada, la función TSRInstall instla los nuevos manejadores para las interrupciones 28h, 13h, 09h. La última tarea desarrollada por la rutina de instalación, es la de calcular el tamaño del programa y llamar a la función **keep** para terminar el programa mientras deja el código necesario residente en la memoria. El primer parámetro asignado en la función **keep** asigna un cero a la bandera de **errorlevel** del DOS, indicando con esto la terminación normal de un programa. El segundo parámetro especifica el tamaño del block de memoria que va a ser protegido. Este tamaño debe de ser dado en párrafos de 16 bytes, y es obtenido por medio de la resta de dos variables predefinidas de Turbo C, **_heapbase** y **_psp**. La varible **_heapbase** es un apuntador hacia el primer byte de memoria debajo del final del segmento del stack, y marca el final de la memoria que normalmente es ocupada por un programa de Turbo C. la variable predefinida **_psp** la dirección del segmento del "**program segment prefix**" (PSP), el cual es un registro que contiene información del programa y que siempre está localizado al principio de el programa en memoria. De manera que el número total de párrafos asignados a un progrma, puede ser obtenido por medio de la resta del contenido de estos dos apuntadores de direcciones.

La función **NewInt13** es el nuevo manejador de la interrupción 13h. esta función simplemente asigna un valor de 1 (uno) en la bandera **InBios** indicando que el BIOS está ocupado en una actividad de disco, llama entonces al manejador de interrupción original, y cuando le es regresado el control, asigna un valor de 0 (cero) a la bandera.

La función NewInt09 es el nuevo manejador de la interrupción de el teclado (09h). Esta función primero llama al manejador original de la interrupción, cuando éste último le regresa el control, desactiva las interrupciones con el objeto de no ser interrumpido en la mitad del proceso en el que checa y "prende" la bandera denominada **Busy**. Esta bandera sirve para prevenir que una aplicación sea llamada cuando todavía permanece activa. la función **NewInt09** llama entonces a una función del tecaldo predefinida **KbdGetShift** para determinar si la combinación de teclas adecuada fué presionada, si esto ha sucedido, la función checa si actualmente se lleva a cabo actividad en el disco mediante la revisión del estado de la bandera **InBios**, y si existe una interrupción del DOS en proceso, mediante la revisión del estado de la bandera **InDOS**.

Si el estado de las banderas **InBios** e **InDOS** es favorable para activar la aplicación (contienen valor 0), se llama a la función **Activate**, y cuando ésta última regresa el control, se "apaga" la bandera **Busy** y el manejador de interrupción regresa el control.

La función **NewInt28** es el nuevo manejador de la interrupción 28h y realiza exactamente las mismas tareas que la función **NewInt09**, excepto que no checa si el MS-DOS se está utlizando, ya que la respuesta negativa es obvia.

La función **Activate** puede ser llamada ya sea por la función **NewInt09** a bien por **NewInt28**, cuando uno de esos manejadores de interrupción han detectado que la combinación de teclas predefinida (Hot Key) ha sido presionada y que es posible invocar a aplicación TSR.

Cuando es llamada, la función **Acivate**, salva la dirección de transferencia de disco (DTA) y se cambia a la dirección que fué salvada en el momento de la instalación. Después, antes de llamar a la aplicación TSR, la función **Activate** informa al sistema operativo que un nuevo proceso está activo asignando a un registro del DOS de el "**program segment prefix**" el valor correcto del progama C. El sistema operativo por medio de la dirección del "**program segment prefix**". Para lograr lo anterior, existen dos funciones no documentadas para manipular el registro interno del acutal PSP. La función 51h regeresa el PSP actual, y la función 50h lo modifica. El PSP original es guardado en una variable, y la nueva dirección es obtenida de la variable predefinida de Turbo C **_psp**. Una ventaja adicional obtenida de cambiar el PSP, reside en que el MS-DOS guarda los manejadores de archivos en el PSP actual, y es posible que no exista suficiente memoria disponibla para abrir mas manejadores en el PSP actual. Es importante aclarar que debido a que este proceso requiere de dos servicios del DOS (50h y 51h), éstos servicios pueden ser ejecutados siempre y cuando el DOS no se encuentre activo en ese momento, de esta forma el PSP solo será cambiado cuando la bandera **InDOS** contenga un valor de 0 (cero).

Finalmente, la aplicación TSR es llamada a través de un apuntador de función que fue previamente definido durante la instalación. Cuando el control es regresado a la función Activate, esta última restaura todos los valores guardados : el PSP, el DTA, y el stack. Acto seguido, regresa el control al manejador de interrupción de donde fué llamada (**NewInt9** o **NewInt28**).

La función **TSRInDOS** simplemente regresa el valor actual de la bandera **InDos**.

CAPITULO V

Transmisión serial asincrónica de información.

5.1.- Comunicación asincrónica.

La comunicación de información a través de la computadora está basada en la transimisón de bytes de información de un equipo a otro. Si se cuenta con 8 líneas entre los dos equipos, se puede asignar cada línea a un bit de información correspondiente, de manera que se pueda enviar byte por byte, esto es conocido como "transferencia paralela". El puerto paralelo de la computadora trabaja de esta forma.

Por otro lado, si se cuenta con una línea, la transmisión de cada byte, tendrá que hacerse bit pot bit en forma serial. Mas aún, es posible enviar la información sincrónicamente de modo que cada byte es enviado en lapso fijo predeterminado, o bien, asincrónicamente en donde el tiempo transcurrido entre un envío y otro no sea necesariamente uniforme.

La comunicación serial es mucho mas barata ya que requiere menos líneas de datos, además, el modo de transmisión asincrónico tiene mucho mas demanda debido a que no necesita de hardware especial entre el transmisor y el receptor.

En resumen, la comunicación serial asincrónica es la solución general a las necesiades de comunicación, debido a su bajo costo y al bajo nivel de complejidad requerido en el hardware. Por supuesto, en este modo de transmisión de información se debe tener en mente la necesidad de convertir cada byte de información en series de bits, indicando el principio y el final de cada byte transmitido. La figura V.1 ilustra el concepto de comunicación serial asincrónica.

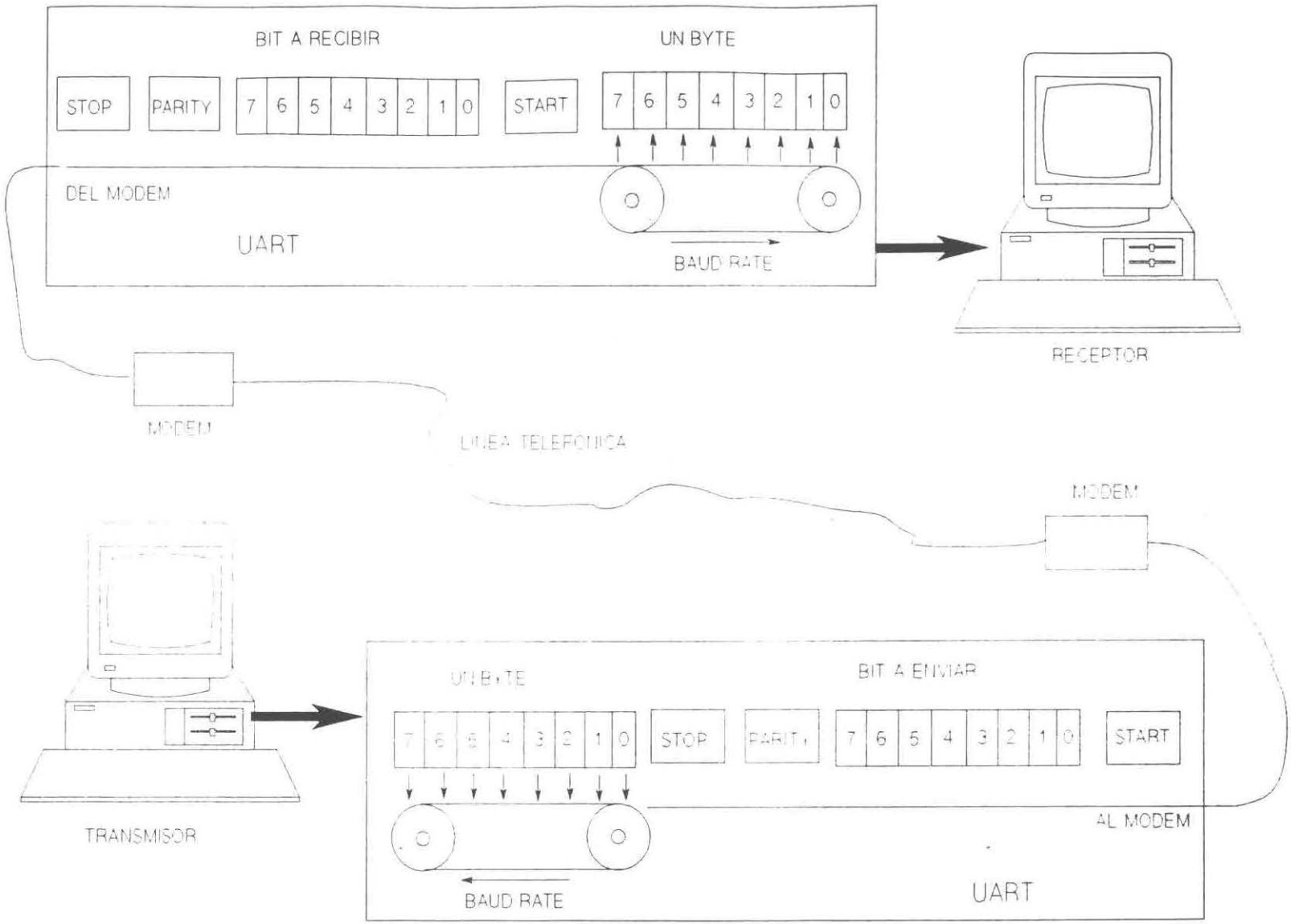

FIGURA V.1 COMUNICACION SERIAL ASINCRONICA

5.2.- Uso del Puerto Serial.

Quizá ningún otro concepto causa tanto interés en los programadores, como el puerto serial asincrónico. A diferencia del simple uso de un puerto paralelo, el uso del puerto serial puede traer como consecuencia secundaria una larga lista de errores de transmisión. Aún y con estos problemas, el puerto serial, es usado con bastante regularidad debido a que es el modo mas barato de enlazar dos aparatos que están separados por unos cuantos metros.

La comunicación serial asincrónica, es llevada a cabo en la computadora mediante el uso del puerto serial, en los sistemas MS-DOS es conocido como adaptador serial o adaptador serial asincrónico. Este adaptador está basado en el Intel 8250 UART (Universal Asynchronous Reciver Trasmitter), el cual cuenta con un RS-232C para hacer la conexión, y es el encargado de la conversión de cada byte en los bits correspondientes para efectuar la transmisión y viceversa cuando se trata de recepción de la información.

Cada byte de información trasmitido por el puerto serial, esta compuesto de las siguiente secuencia de señales :

1. Un bit de inicio (Start Bit).
2. 8 bits de información (7 en algunos casos).
3. bit de paridad (Opcional.)
4. Uno o dis bits de detención. (Stops bits).

Entre la transmisión de cada byte, cualquier cantidad de tiempo puede transcurrir. El bit de incio señala el comienzo de la transmisión de un nuevo byte. Los bits de información son entonces transmitidos, seguidos por un bit de paridad opcional. Finalmente los bits de detención son transmitidos, señalando con esto que la transmisión a concluido.

El bit de paridad, es utilizado para constatar que la información ha sido comunicada correctamente.

La velocidad de transmisión de cada bit es deniminada "Baud Rate", y es asociada generalmente como una relación de bits/segundos.

Cuando se desea transferir archivos de texto por el puerto serial, solo son necesarios siete bits de información, debido a que ninguna letra o signo de puntuación requiere de ocho bits para su representación. Enviando solo siete bits de información, se incrementa la velocidad en la cual un archivo es transmitido. Los problemas empiezan cuando se desea mandar un archivo que se encuentra guradado en disco en forma de no-texto, tal es el caso de un programa ejecutable.

Todos los archivos de programa y algunos archivos de información contienen información que usa todos los ocho bits de un byte. Para transmitir esta clase de archivos se debe de enviar los ocho bits completos. Por esta razón el sistema TSRMITE, esta preparado para transmitir ocho bits por byte. Sin embargo, existe un pequeño problema cuando se transmiten archivos binarios : la marca de EOF (End Of File), no puede ser usada como señal de fin de archivo. Para resolver este problema, el número de bytes que ocupa el archivo a transmitirse es contabilizado y puesto como parámetro de la comunicación, para que el receptor sepa cuantos bytes va a recibir.

5.3.- El estándar RS-232C.

El RS-232C es un estándar publicado en 1969 por la EIA (Electronic Industries Association). Las letras RS son la siglas de "Recommended Standard" y el 232 es el número de identificación para ese estándar en particular. La letra C desinga la última revisión hecha al estándar RS-232. El propósito de este estándar es la definición de las características eléctricas para la interfase de equipos DTE (Data Terminal Equipment) y los DCE (Data Communications Equipment. Para los usuarios de la IBM, éstos términos se refieren a la computadora y al modem respectivamente.

5.4- Asignación de los pins en el RS-232C.

La implementación física de el RS-232C se muestra en la figura V.1. Esta figura muestra la asignación de los pins usada en el diseño de el conector DB-25, normalmente utilizado en cada uno de los cabos del cable de comunicación asincrónica.

Aún y aunque la asingación de pins fué definida bajo el estándar RS-232C, el diseño actual del conector es controlado por la ISO (International Standard Organization). De manera que se supone que el conector hembra será usado en los modems u otros aparatos, y el conector macho en el puerto serial, pero no todos los vendedores de equipo obedecen siempre este estándar. Mas aún, muchos de los pins no son utilizados actualmente en las aplicaciones de comunicación.

A continuación, se muestra la utlización de los pins que se usan con mas frecuencia en la comunicación de datos :

- Transmit Data (TD, Pin 2)

Las señales en este pin son transmitidas desde la PC hacia el equipo destino. El puerto serial mantiene este circuito apagado cuando no se está transmitiendo información.

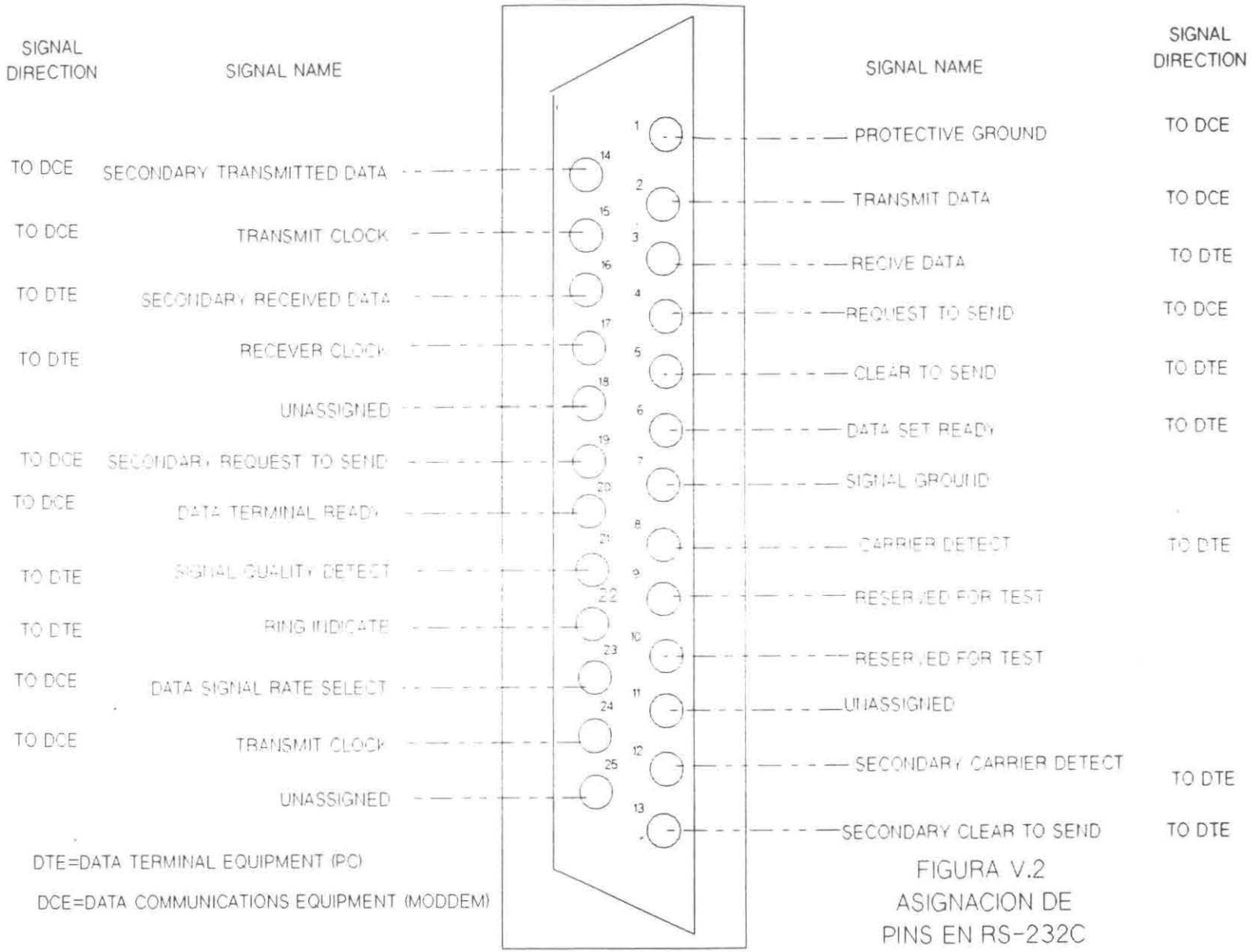

FIGURA V.2
ASIGNACION DE
PINS EN RS-232C

- Recive Data (RD, Pin 2)

Las señales en este pin son transmitidas desde el equipo destino hacia la PC. este circuito también es mantenido apagado, mientras no se este transmitiendo información.

- Request to Send (RTS, Pin 4)

Este circuito es usado para enviar una señal hacia el equipo destino pidiendo con esto una señal de CLEAR para poder enviar infomación en el pin 2. esta señal es usada en combinación con el circuito Clear To Send para controlar el flujo de información entre ambos equipos.

- Clear to Send (CTS, pin 5)

Este circuito es usado por el equipo destino para indicarle a la PC que se encuentra listo para recibir información. Cuando este circuito está apagado, el equipo receptor esta indicando a la PC que aun no está listo para recibir información.

- Data Set Ready (DSR, Pin 6)

Cuando este circuito está prendido, es una señal para indicar a la PC que el modem está concectado correctamente a la línea telefónica.

- Signal Ground (SG, Pin 7)

Este circuito sirve como señal de referencia para todos los demás circuitos. Para las demás señales siempre está prendido.

- Data Terminal Ready (DTR, Pin 20)

La PC prende este circuito cuando está lista para comunicarse con el equipo destino.

5.5.- Configuranción del Cable.

Las señales del RS-232C son trasmitidas hacia y desde el puerto serial a través de un cable de comunicación. El cable puede ser redondo o plano. La forma del cable no hace ninguna diferencia en la habilidad de soportar la comunicación, pero los tipos de conexiones provistos en cada uno de los cabos del cable o el número de cables conectados a los pins en el conector hacen la diferencia. La mayoría de los puertos seriales tienen un conector macho DB-25, los cuales requieren de un cable con un conector hembra.

El puerto serial de las máquinas AT son una excepción a la regla del DB-25. El puerto serial estándar de IBM para máquinas AT tiene un conector macho DB-9. Muchos proveedores hacen cables que convierten a este puerto serial en un conector DB-25 para que pueda ser usado correctamente en la conexión con otras máquinas.

Las interfases de cables en su mayoría están diseñadas para conectar computadoras y modems, sin embargo, cuando se requiere comunicar dos computadoras sin un modem de por medio, tal es el caso del sistema TSRMITE, se necesitan hacer ciertos ajustes en la conexión de los cables.

5.6.- Comunicación sin Modem.

Cuando se pretende comunicar dos computadoras locales por medio del puerto serial, es necesario diseñar una interfase denominada "NULL MODEM", de hecho, este tipo de interfase no incluye modem en absoluto, en términos simples, es un cable o un set de conectores diseñados para eliminar la necesidad del modem. La interfase NULL MODEM hace que la computadora opere como si se estubiera comunicando con un modem. En la figura V.3 se puede observar que la computadora y el modem envían y reciben información por líneas compatibles, y las dos computadoras están esperando para enviar y recibir información por la misma línea. La figura V.4 muestra la solución a este problema mediante el uso de una interfase NULL MODEM. Cruzando los circuitos RD y TD, la microcomputadora A está "escuchando" por la línea que la computadora B, utiliza para "hablar" y viceversa. Existe una gran variedad de formas de conectar los cables, de manera que dos computadoras locales puedan establecer comunicación. La conexión mostrada en la figura V.5, dicha conexión es usada para trabajar con una computadora PC y la mayoría de las microcomputadoras del mercado.

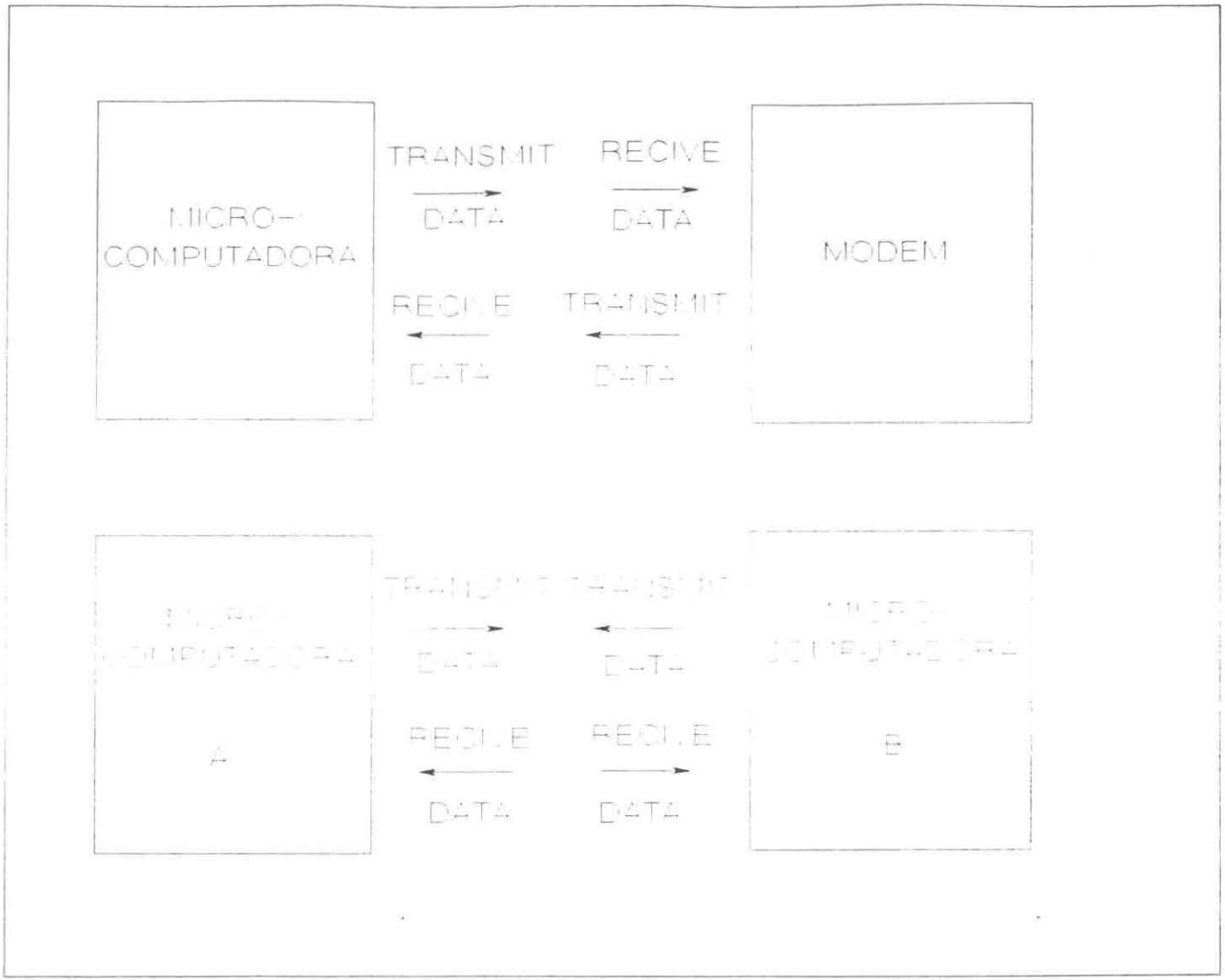

FIGURA V.3 INTERFASE MICROCOMPUTADORA-MODEM

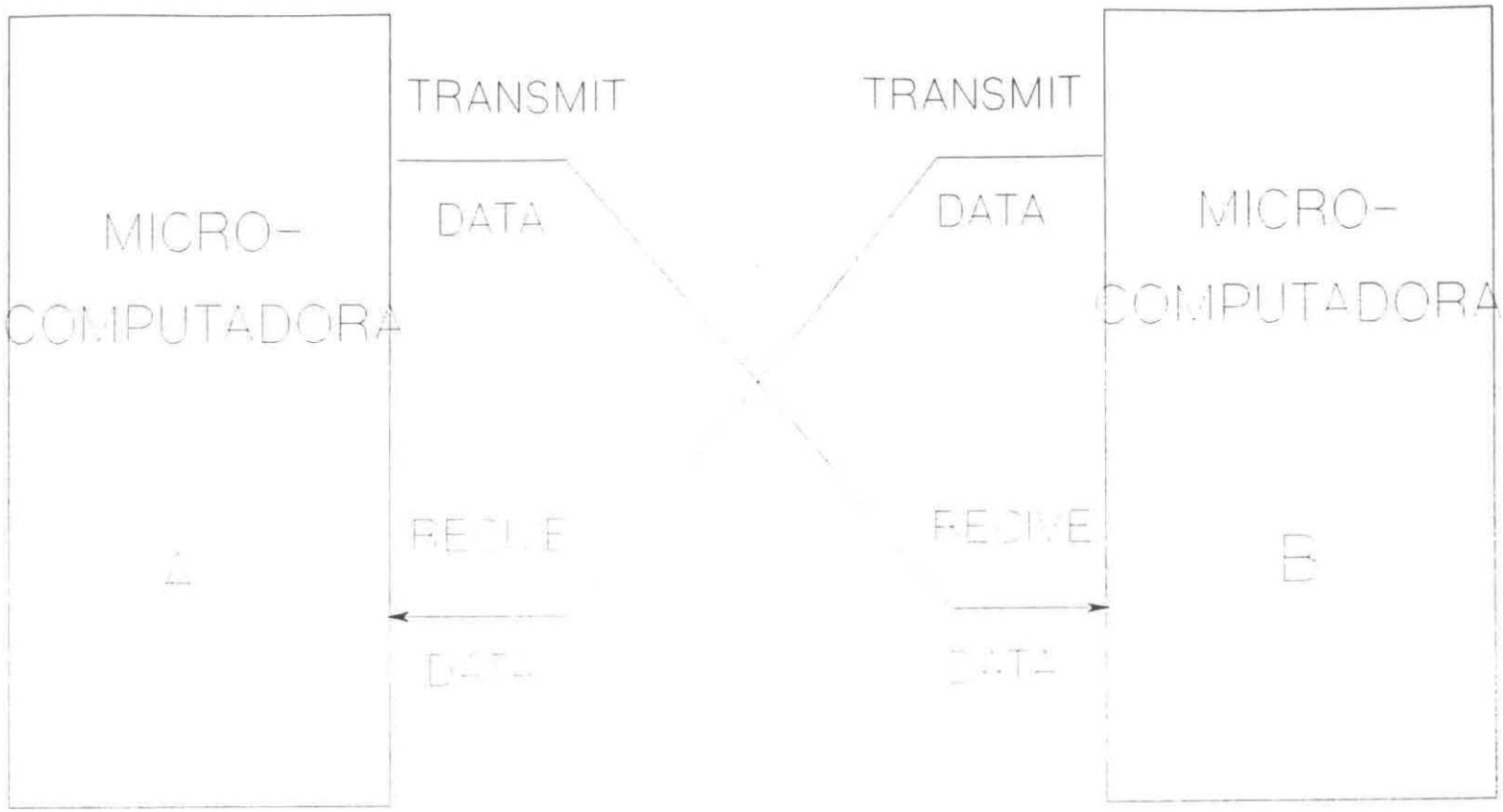

FIGURA V.4 INTERFASE NULL MODEM

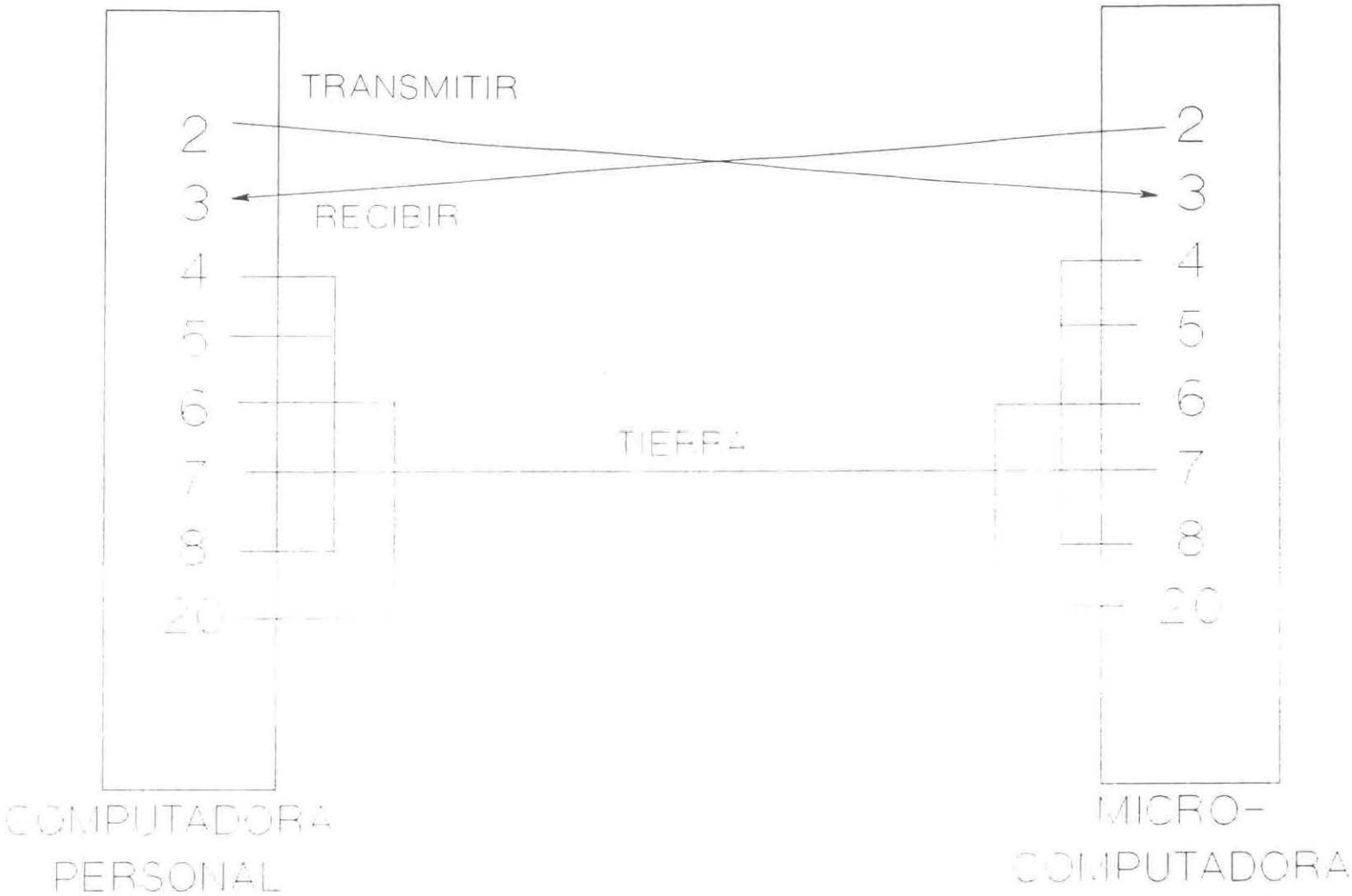

FIGURA V.5 DIAGRAMA DEL NULL MODEM

Cuando se realiza una conexión directa computadora-computadora por medio de la interfase NULL MODEM, la información puede ser transimida rápidamente y además se elimina la posibilidad de que sucedan muchos tipos de errores asociados con la transmisión de datos usando la línea telefónica.

5.7.- Programación del puerto serial.

Existen varias formas de accesar el puerto serial en los sistemas MS-DOS. Se puede controlar el puerto serial a través de un "Installable Device Diriver" que desarrolle las tareas de entrada y salida con el puerto serial. La segunda forma de accesar el puerto serial, es mediante la instalación de un programa TSR que utilice los servicios de interrupción del BIOS (14h) para accesar el puerto serial. El tercer método consiste en desarrollar una aplicación que incluya un manejador de interrupción para el puerto serial.

El sistema TSRMITE utliza el segundo método de acceso al puerto serial. De manera que a continuación se explica como se utiliza el BIOS para realizar la comunicación.

5.8.- Acceso del puerto serial a través del BIOS

El puerto serial de la PC o compatible puede ser accesado a través del DOS, a través del BIOS, o bien, controlando directamente el software de la máquina. El acceso del puerto serial a través del DOS generalmente no es una buena solución debido a que el DOS no provee de ninguna función para obtener el estado del puerto, sólo provee aquellas utilizadas para leer o escribir el puerto. Aun y aunque el acceso directo del hardware del puerto puede ser utilizado, esto no es necesario, ya que se pueden implantar sistemas de gran desarrollo accesando las interrupciones del BIOS.

Son cuatro los servicios del ROM BIOS utilizados para accesar el puerto. Estos servicios están incluídos en la interrupción 14h.

5.9.- Inicialización del puerto.

Antes de utilizar el puerto serial, es necesario inicializarlo con los parámetros necesarios.

Los parámetros de inicialización del puerto serial, son los siguientes :

1 - Baud Rate (Velocidad de transmisión BAUDS)

2 - Tamaño del caracter.

3 - Paridad.

4 - Stop Bits.

El servicio 0 de la interrupción 14h es utilizado para inicializar el puerto serial. Al igual que en otras interupciones del BIOS, el registro AH es usado para indicar el número del servicio. El registro AL indica los parámetros de inicialización, los cuales deben ser empaquetados en un solo byte como se muestra en la figura V.6.

La velocidad de transmisión es establecida como se muestra en la tabla VI-a. La paridad se muestra en la tabla VI-b.

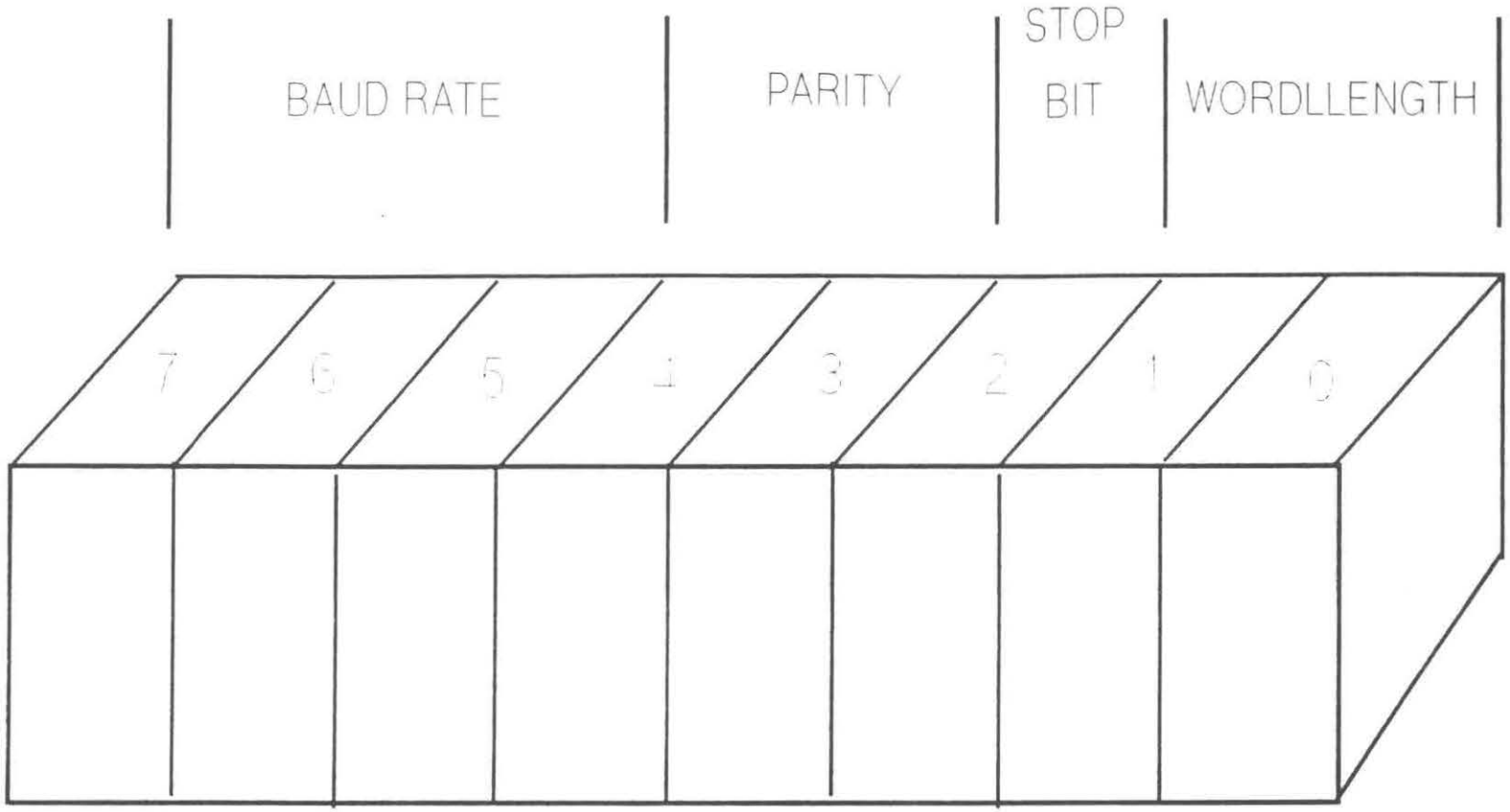

FIGURA V.6 PARAMETROS DE COMUNICACION EMPACADOS EN UN BYTE

TABLA V-a

BAUD	CONVINACION
9600	1 1 1
4800	1 1 0
2400	1 0 1
1200	1 0 0
600	0 1 1
300	0 1 0
150	0 0 1
110	0 0 0

TABLA V-b

PARIDAD	CONVINACION
SIN PARIDAD	0 0 O 1 0
IMPAR	0 1
PAR	1 1

El número de "stop bits" es determinado por el bit 2 del byte de inicialización. Si el bit 2 es "1", se usarán dos "stop bits", de cualquier otra forma se utilizará un solo "stop bit". Finalmente el número de bits por caracter, es establecido en el bit 1 y bit 0 del byte de inicialización. De las cuatro posibles combinaciones, sólo dos son válidas. Si los bit 1 y 0 contienen la combinación "10", se utilizarán 7 bits por caracter. Si la combinación es "11", se utilizarán 8 bits por caracter.

5.A.- Transmisión de Bytes.

El servicio 1 de la interrupción 14h transmite un byte a través del puerto serial especificado en el registro DX. El byte que se enviará debe estar contenido en el registro AL. El status de la transmisión es regresado en el registro AH.

Si el bit 7 del registro AH es igual a 1, esto significa que ha ocurrido un error en la transmisión. Para determinar la causa del error, se debe leer el status del puerto.

5.B.- Chequeo del status del puerto.

El servicio 3 de la interrupción 14h es usado para obtener el status del puerto. El número que se va a checar debe ser especificado en el registro DX. Después de generar la interrupción, los registros AH y AL contienen el status del puerto, empaquetado en dos bytes, tal y como se muestra en la Tabla V-C.

Como se puede observar, la mayoría de la información regresada en ambos registros se aplica únicamente para comunicación vía modem. Sin embargo, existe una condición que es importante : "Data Ready". Por medio del chequeo de esta condición, se puede determinar cuándo un byte de información ha sido recibido por un puerto y está listo para ser leído.

TABLA V-c

LINEA DE STATUS (AH)

SIGNIFICADO CUANDO ESTA	BIT
DATA READY	0
OVERRUN ERROR	1
PARITY ERROR	2
FRAMING ERROR	3
BREAK-DETECT ERROR	4
TRANSFER HOLDING REGISTER EMPTY	5
TRANSFER SHIFT REGISTER EMPTY	6
TIME-OUT ERROR	7

STATUS DE MODEM (AL)

SIGNIFICADO CUANDO ESTA	BIT
CHANGE IN CLEAR-TO-SEND	0
CHANGE IN DATA-SET-READY	1
TRAILING-EDGE RING DETECTOR	2
CHANGE IN LINE SIGNAL	3
CLEAR-TO-SEND	4
DATA-SET-READY	5
RING INDICATOR	6
LINE SIGNAL DETECTED	7

5.D.- Recepción de Bytes.

El servicio 2 de la interrupción 14h es usado para leer un byte de información de el puerto serial. Nuevamente, el puerto serial es especificado en el registro DX. Una vez generada la interrupción, el caracter leído se encuentra en el registro AL. De igual forma que cuando se trasmite un caracter, el bit 7 del registro AH es utilizado para indicar si la función se realizó sin problemas o bien, si existe algún error de recepción.

En general, estas son las funciones provistas por el BIOS para trabajar con el puerto serial.

5.E.- Transmisión de archivos.

La manera idónea de trasnmitir información a través del puerto serial es monitoreando el status de la señal CTS (Clear Tto Send) de el puerto receptor. No debe enviarse información hasta que el CTS indique que esta libre. De esta manera cuando un protocolo de hardware es utilizado, la rutina de transmisión en pseudo-c, se implementa de la siguiente manera :

```
do
 {

  while(not CTS) espera;

  envia (byte);

 } while(bytes-por-mandar);
```

Aunque esta es la forma stándard general de usar el puerto serial, cuando este puerto es accesado por medio de las interrupciones del BIOS (caso de TSRMITE), es posible implementar un protocolo de comunicación por software, el cual trabaja de la siguiente manera : La computadora que esta transmitiendo envía el primer byte de información y espera que la computadora receptora regrese un byte de reconocimiento. Una vez que el reconociemiento es recibido, el transmisor envía el siguiente byte y espera de nuevo por el reconocimiento. Este proceso continua hasta que la transmisión ha sido concluida. En pseudo-C, las rutinas de transmisión y recepción son implementadas de la siguiente manera :

```
Transmision ()
{

        while (bytes-por-mandar) {

                envia(byte);

                espera();
        }
}
```

```
Recepcion()
{
    do {
        recibe_byte ();
        envia(reconocimiento);
    } while(bytes-por-leer);
}
```

De esta forma, el transmisor nunca enviará bytes al receptor hasta que este último haya leido el anterior, no importando las diferencias de velocidad de operación de las dos computadoras.

El único defecto de este tipo de protocolo, reside en que su efectividad reduce la velocidad de transmisión, ya que dos bytes serán enviados por cada byte de información.

- Rutinas de transmisión.

La primera rutina necesaria, es una función que transmita un archivo a través del puerto serial. En general, esta rutina debe de abrir el archivo que va a ser transmitido, contar el número de bytes que contiene, transmitir el número de bytes, y finalmente enviar el archivo.

Esta rutina debe de contener las siguientes funciones :

Envío del nombre : Esta función tiene dos propósitos. Primero, estabilizar la comunicación enviando un caracter especial hasta que el receptor envie un reconocimiento, que indique que la transmisión puede empezar, y segundo, para enviar el nombre del archivo que será transmitido.

Chequeo-Puerto : Esta función regresa el estado del puerto serial, dependiendo de el valor de este estado, es posible determinar cuando se enviará el siguiente byte de información.

Espera-Por : Esta función tiene el objetivo de hacer esperar a la rutina de transmisión por un reconocimiento que le indique que puede seguir la comunicación.

La rutina de recepción de un archivo es exactamente la oposición a la de transmisión. Primero, la rutina debe esperar por la señal inicio de la comunicación, Acto seguido, responde envíando una señal de reconocimiento, indicando que la transmisión puede ser iniciada. Entonces lee del puerto el nombre del archivo, seguido del número de bytes que lo forman. Finalmente el archivo es leido, considerando que por cada byte recibido, enviará un byte de reconocimiento.

Tanto la rutina de recepción como de transmisión que se encuentran implementadas en el sistema TSR fueron diseñadas dentro de manejadores de interrupción del pulsador de tiiempos (Ver Capítulo III), esto es lo que finalmente permite que la transmisión se lleve a cabo, mientras se permita al usuario ejecutar otras aplicaciones.

BIBLIOGRAFIA

DVORAK JOHN C., ANIS NICK
Guide to Desktop Telecomunications.
McGraw-Hill, 1990, USA.

DUNCAN RAY.
Advanced MSDOS Programming.
Microsoft PRESS, 1988. USA.

JORDAN LARRY, BRUCE CHURCHILL.
Comunications and Networking. 3ed.
BRADY, 1989, USA.

THE WAITE GROUP.
MS-DOS Developer´s Guide 2ed.
Howard W. SAMS & Company, 1989, USA.

NORTON PETER, RICHARD WILTON
The IBM PC & PS/2.
Microsoft PRESS, 1988, USA.