19
2ej

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Escuela Nacional de Estudios Profesionales

"ARAGON"

INGENIERIA EN COMPUTACION

# COMUNICACION AUTOMATICA ENTRE COMPUTADORAS PERSONALES

## TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE

Ingeniero en Computación

P R E S E N T A:

Silvestre López Abundio

Asesor de Tesis: Ing. David J. González Maxinez

San Juan de Aragón, Edo. de Méx., 1993.


**UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso**

# PROLOGO

El objetivo de este trabajo es la presentación del diseño de un sistema de comunicación entre computadoras personales con carácter automático. El servicio de transferencia de información es desarrollado en lenguaje 'C', considerando además a la línea telefónica común para el canal de enlace y al MODEM como interfaz entre la computadora y la línea.

Aunado a ese objetivo se ha considerado la utilidad que puede brindar este sistema a la Universidad Nacional Autónoma de México, (denominado SACCP, Sistema Automatico de Comunicación entre Computadoras Personales). Tanto por su uso en los servicios que presta la administración, como en la impartición de cátedra al ser considerado como un escrito de apoyo en asignaturas como Programación Estructurada y Características de Lenguaje, Sistemas Operativos, Comunicaciones Digitales, Diseño de sistemas Digitales y Redes de Computadoras, dentro de la Carrera de Ingeniero en Computación que se imparte en la Escuela Nacional de Estudios Profesionales Unidad Aragón.

El contenido de cada capítulo es consecutivo, sin embargo, cada uno de ellos puede ser tratado en forma independiente. El Capítulo Uno, toca los puntos referentes a los sistemas actuales y anteriores, sus ventajas, desventajas y normatividades, finalizando con un panorama de lo que se busca alcanzar entorno a los enlaces entre computadoras.

En el Capítulo Dos, se marcan las características específicas de la aplicación, para esto, a) Se proporciona una descripción y un breve análisis del Control Remoto y Correo Electrónico; b) Se delimitan y explican los rasgos de la propuesta.

El seudocódigo, algoritmos, diagramas de flujo y listados de programas son expuestos y documentados en el Capítulo Tres. La implantación, funcionamiento y descripción de los puntos críticos del sistema se encuentran en el Capítulo Cuatro.

Los apartados: Perspectivas y Conclusiones, brindan un resumen de lo alcanzado y de lo que se puede continuar haciendo en esta área; y como apoyo auxiliar, al final se han ubicado un conjunto de apéndices para la consulta de tópicos inherentes en esta obra, que no fueron tratados con profundidad, pero sí manejados en el diseño y desarrollo de este trabajo.

S. L. A.

México 1993

# INDICE GENERAL

PROLOGO..... 9

INTRODUCCION.....13

CAPITULO 1
ANTECEDENTES DE LA COMUNICACION ENTRE COMPUTADORAS

1.1 Introducción.....17
1.2 Breve historia de enlaces entre computadoras.....17
1.3 Enlaces entre computadoras.....19
1.4 Facilidades y limitantes de los sistemas multiusuario.....23
1.5 Modelo OSI.....25
1.6 Hacia la comunicación automática.....28

CAPITULO 2
DESCRIPCION DEL SISTEMA PROPUESTO

2.1 Introducción.....33
2.2 Factores de Software y Hardware.....33
2.3 Correo Electrónico y Control Remoto.....36
2.4 Análisis de las circunstancias actuales.....37
2.5 Descripción del sistema.....41

CAPITULO 3
DESARROLLO DEL SISTEMA

3.1 Introducción.....46
3.2 Seudocódigo.....46
3.3 Diagrama de Flujo General.....48

3.4 Módulos y funciones.....49
3.5 Listado de programas.....50
SACCP.C .....51
ARCHIVOS.C .....54
TSR_POP.C .....58
RECIBE.C .....73
ENVIA.C .....80
ENLACE.H .....90
APOYO.H .....91
LCBBS.H .....97

CAPITULO 4
PUNTOS IMPORTANTES DEL DISEÑO

4.1 Introducción.....103
4.2 Compilación y liga de programas.....103
4.3 Intrumentación.....105
4.4 Funcionamiento.....106
4.5 Mantenimiento.....107

PERSPECTIVAS.....111

CONCLUSIONES.....115

APENDICES
A --> Clubs de PC's .....119
B --> Microprocesadores.....121
C --> Elementos y selección de un equipo de cómputo.....123
D --> Topologías de Redes y características.....126
E --> Utilerías de comunicación para C
(COMMUNICATION TOOLBOX FOR MS AND TURBO C).....129
F --> Términos manejados en la obra.....144
G --> Manejo del MODEM.....147

ESCRITOS DE APOYO.....155

# INTRODUCCION

El presente trabajo muestra el desarrollo de un sistema denominado Comunicación Automática entre Computadoras Personales, éste tiene como finalidad transmitir información entre diferentes usuarios periódicamente, reportando esta actividad y sin que el usuario ocupe tiempo en: Realizar enlace, seleccionar archivo(s) y supervisar la transferencia paso por paso mientras se lleva a cabo la transferencia.

Actualmente existen sistemas de comunicación con computadoras que permiten el intercambio de datos con las ventajas mencionadas en el párrafo anterior, pero requieren de un canal permanentemente libre o dedicado, así como de hardware y software de aplicación especial, lo cual implica un alto costo en su mantenimiento e instalación. En consecuencia adquieren un carácter privado y exclusivo dado que sólo la mediana y gran empresa tienen acceso a una red con dichas características. Aunado a ésto, el uso y acceso es reestringido por los propietarios, razón por la que su comportamiento queda fuera del control de los usuarios.

Estas circunstancias hacen ver la necesidad de tener un sistema alterno flexible y accesible a usuarios independientes sin descuidar los rasgos de eficiencia que se ven reflejados por la velocidad, grado de automatización y transparencia hacia el usuario.

Los rasgos específicos del sistema propuesto son los siguientes:

I) Accesible tanto a grandes como pequeños usuarios,

a) Bajo costo.
b) Fácil manejo.
c) Utilización de un reducido espacio de memoria.
d) Uso de PC's, MODEM's y línea telefónica.
e) Implantación modular en lenguaje 'C'.

II) Liberación de las tareas de transferencia,

a) Carga del software específico de comunicación.
b) Inserción de número telefónico a marcar.
c) Selección de modo: Recepción o Transmisión.
d) Selección de archivo.
e) Espera por inicio y fin de transacción (envío o recepción).
f) Descarga y salida del software utilizado.

La idea general del diseño consiste en la implantación de un conjunto de programas en C dedicados a:

a) Definir una lista de archivos a enviar y direcciones de envío.
b) Realizar marcaciones para envío.
c) Atender llamadas telefónicas para recepción. Y,
d) Actualizar una bitácora de transacciones.

Se contempla también un módulo residente que permite el acceso al modo automático, regreso a la línea de comandos del sistema operativo y descarga del mismo, así como rutinas de envío, recepción y arranque del sistema de comunicación.

En otro aspecto, y dando fin a esta introducción, no queda sino esperar que el presente trabajo sea útil en el continuo diseño de aplicaciones dentro del área de la Comunicación y la Computación.

S. L. A.

# COMUNICACION AUTOMATICA ENTRE COMPUTADORAS PERSONALES

# CAPITULO 1

ANTECEDENTES DE LA COMUNICACION ENTRE COMPUTADORAS

1.1 Introducción.

1.2 Breve historia de enlaces entre computadoras.

1.3 Enlaces entre computadoras.

1.4 Facilidades y limitaciones de los sistemas multiusuario.

1.5 Modelo OSI.

1.6 Hacia la comunicación automática.

# 1 ANTECEDENTES DE LA COMUNICACION ENTRE COMPUTADORAS

## 1.1 INTRODUCCION

La creciente cantidad de información ha suscitado el establecimiento de tecnologías para su transmisión, procesamiento, almacenamiento y anális dentro de la diferentes épocas de las sociedades humanas. Por ejemplo, los griegos hace 2000 años hacían uso de pergaminos para grabar y transportar información de un poblado a otro. Mas reciente y actual, es el uso de tambores por tribus de África y Brasil para el envío de mensajes.

Existen otros ejemplos, pero en lugar de dar su descripción, pasemos a través de este capítulo al tratado de los factores, la historia y el presente de la computadora en relación con la transmisión de información.

## 1.2 BREVE HISTORIA DE ENLACES ENTRE COMPUTADORAS

La historia en este campo no ha sido lineal, se ha dado en base a avances esporádicos, pero en general puede afirmarse que en los primeros sistemas multiusuario la computadora central contenía toda la información y la compartía en base a las solicitudes desde cada terminal, éstas consistían de un teclado y un tubo de rayos catódicos para el desplegado de datos y resultados. Posteriormente aparecieron modificaciones a este tipo de sistemas, pero siempre utilizando un canal dedicado para la transmisión.

El primer sistema multiusuario es sin duda la máquina de Stibitz denominado Calculador de Complejos. Este sistema realizaba operaciones aritméticas con números complejos y podía accesarse desde tres terminales locales. Además, este equipo fue también pionero en teleprocesamiento ya que en 1940 se conectó una máquina en Nueva York a una terminal (Teletipo) del Colegio de Dartmouth en Nuevo Hampshire (342 km de distancia).

En Diciembre de 1969, surgió la primera red experimental llamada *ARPANET*, desarrollada por la Agencia de Proyectos e Investigaciones Avanzadas *ARPA* del Departamento de Defensa de los Estados Unidos esta red contaba con 4 nodos y conectaba 100 computadoras ubicadas en varios estados de ese país. Muchos de los conocimientos actuales sobre redes son resultado directo del proyecto *ARPANET*, por ello la terminología actual de redes de computadora conserva algunos conceptos ideados para esta primera red.

En 1973, la compañía Xerox desarrolla una red de gestión de archivos en base a sus equipos instalados en Estados Unidos, esta red fue pionera de la red Ethernet que hoy conocemos.

En 1974, comienza a funcionar la red pública *TRANSPAC* de Francia que conecta a cientos de equipos en todo el país *TRANSPAC* fue una de las primeras redes publicas.

En 1981, México pone en marcha su red pública *TELEPAC* para ofrecer servicios de transmisión de datos en todo el país.

Finalmente en ese mismo año la aparición de las computadoras personales marca un cambio definitivo en la informática y comienzan a desarrollarse las primeras redes locales de microcomputadoras y computadoras personales.

Ante estos contínuos avances de la Computación y las Telecomunicaciones, la Organización Internacional de Normas *ISO* y la Unión Internacional de Telecomunicaciones *UIT* ) a través del Comité Consultivo de Telefonía y Telegrafía *CCITT* deciden establecer las primeras normas para la conectividad de equipos en redes de computadoras con lo que quedan establecidas las bases fundamentales de las redes de computo actuales.

## 1.3 ENLACES ENTRE COMPUTADORAS

Las redes locales y sistemas multiusuario basados en minicomputadoras, hoy por hoy siguen utilizando el esquema original, aunque claro, la velocidad, capacidad de almacenamiento y dimensiones físicas han sido mejoradas. Las topologías de red han seguido una analogía biológica, buscando un mejor servicio cada vez más indispensable.

Los sistemas multiusuario basados en minicomputadoras penden su funcionamiento de un procesador central capaz de servir a diferentes usuarios a la vez, en su estación de trabajo, dedicando intervalos de tiempo en base a prioridades para dar respuesta a sus solicitudes de proceso. La terminal sólo puede realizar operaciones de bajo nivel para la transmisión y recepción de peticiones o resultados, dejando toda la carga de trabajo al procesador central, quien además administra otros recursos como impresoras y monitores.

Del otro lado tenemos las redes de computadora de área local y amplia, (basadas en computadoras personales) las cuales se han caracterizado por permitir procesar información en cada estación de trabajo compartiendo todos los recursos del sistema y controlando al mismo tiempo los periféricos.

En general un sistema multiusuario tiene como finalidad:

a) Compartir información
b) Compartir periféricos

} Utilización óptima de los recursos.

Ambos esquemas multiusuario (Basado en una minicomputadora o en computadoras personales), tienen otras características que las asemejan y distinguen:

- Compatibilidad con otros sistemas tanto en hardware como en software.

- El Sistema Operativo.

- El tipo o forma de conexión (Circular, lineal, distribuida, etc.).

- Número de estaciones de trabajo que soporta, en este rubro se incluyen periféricos como impresoras, discos duros, lectoras de cinta, etc.

- Protocolos de comunicación.

- Tipo de canal de comunicación.

- Capacidad de almacenamiento para procesamiento. Esto es, el espacio de RAM disponible para el procesamiento de información y funcionamiento de lo paquetes tanto de comunicacion como de trabajo.

- La modularidad, que define la capacidad de expansión o sustitución de elementos.

- La velocidad de transmisión y de respuesta.

Estas características, entre otras, son las que permiten dislucidar que es mejor en un momento dado, si un sistema multiusuario basado en minicomputadora, uno basado en computadoras personales o uno mixto. (Las figuras 1.1 y 1.2, ilustran estos arreglos generales).

Sin embargo, estos esquemas dejan fuera a un miembro que día a día extiende su número: El usuario independiente.

Si, efectivamente todos los sistemas mencionados anteriormente son sistemas de compañías, teniendo utilidad sólo dentro de las mismas. Además dichos sistemas multiusuario no satisfacen del todo los deseos del usuario directo, ni del departamento de costos y finanzas, ya que la adquisición de estos sistemas resulta a elevado costo; y de que el cliente queda atado a su distribuidor quien eleva los precios al ser el único capaz de brindar mantenimiento y accesorios al sistema. A pesar de lo anterior, la implantación de estos sistemas ocacionan una reducción de costos administrativos a largo y mediano plazo.

# SISTEMA MULTIUSUARIO

## BASADO EN UNA MINICOMPUTADORA

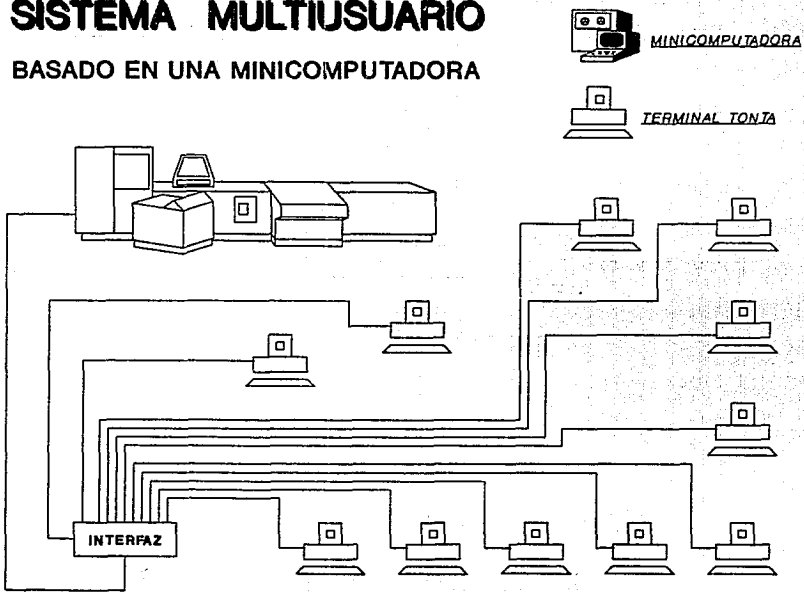



FIGURA 1.1

# SISTEMA MULTIUSUARIO

## BASADO EN COMPUTADORAS PERSONALES

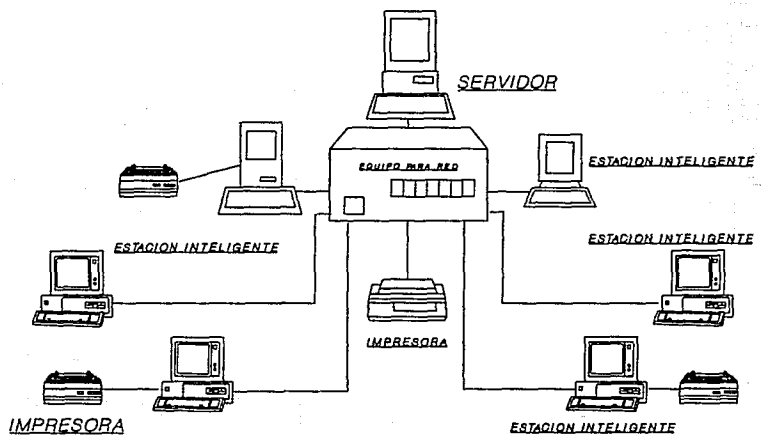



FIGURA 1.2

## 1.4 FACILIDADES Y LIMITANTES DE LOS SISTEMAS MULTIUSARIO

En relación al punto anterior, a continuación se listan las principales características de los sistemas multiusuario, y que son una facilidad(F) o una limitante(L) de ellos.

MULTIUSUARIO BASADO EN UNA MINICOMPUTADORA:

L - Sistema operativo particular de la computadora.

L - Terminales de trabajo tontas, incapaces de procesar información remitiendose a comunicar peticiones y resultados.

L - El área física de aplicación no es mayor a un kilómetro, aunque a últimas fechas han aparecido sistemas capaces de enlazarce intercontinentalmente, mediante la utilización de accesorios como: REPETIDORES, GATEWAYs, RUTEADORES y BRIDGEs, y claro del uso de satélites.

L - Hardware compatible exclusivo de la marca.

L - Uso de software ajeno a la marca del fabricante.

L - Uso de canales dedicados.

L - El mantenimiento a estos sistemas es costoso, requieren de áreas especiales para funcionamiento, y para el soporte técnico se está limitado al distribuidor autorizado.

F - Aparición de nuevos sistemas operativos compatibles con el SO propio, (como es el caso del DOS, Novell, Xenix, etc.).

F - Adquisición integral de un equipo de cómputo, se compra todo el sistema, no requiere el armado de diferentes productos que [illegible] compatibles.

MULTIUSUARIO BASADO EN COMPUTADORAS PERSONALES
(REDES DE COMPUTADORAS):

L - Existencia de sistemas operativos particulares a la arquitectura de conexión de la red. (Artisoft, D-link, LAN_tastic, LANsmart, etc.).

L - Requiere de hardware y software exclusivo para cada topología.

L - El área física de trabajo es menor a un kilómetro (Si es una LAN).

L - Uso de canales dedicados.

L - Uso de protocolos no estandares o simplemente diferentes, aunque existe una gran diversidad de los mismos, estos son en la mayoría de los casos incompatibles por lo cual se requiere de "gateway's" o "bridge's" para realizar el acoplamiento.

F - Sistema operativo versátil y rasgos de compatibilidad entre los diferentes tipos de DOS. (OS, MS, XENIX, DR). Lo cual da portabilidad a las aplicaciones.

F - Terminales inteligentes (capaces de realizar procesamientos en la estación sin requerir del ordenador central o servidor), o tontas a opción.

F - Amplias posibilidades de expansión y mantenimiento con el fabricante, distribuidor autorizado o algún otro tipo de soporte técnico.

F - Existencia de diferentes configuraciones seleccionables en base a las necesidades especificas a resolver.

En el apéndice D, se encuentran las caracteristicas y configuraciones básicas de este tipo de sistemas.

## 1.5 MODELO OSI

Un aspecto importante en la computación y la comunicación, es la estandarización del hardware y software para todo sistema. Afortunadamente ya existe una tendencia hacia la estandarización, la organización internacional de estandares (International Stantard Organization, ISO), establece niveles (áreas o capas) de desarrollo para utilerias tanto fisicas como lógicas en torno a los enlaces entre computadoras.

Se maneja la palabra afortunadamente, porque tal estandarización permite tanto a grandes como a pequeños usuarios tener la facilidad de encontrar equipo como aditivo o reemplazo compatible, no importando la marca, sino tan sólo las caracteristicas del estandar buscadas.

Actualmente no se ha logrado que todas las compañias sigan estas normas, pero dia a dia algunas de ellas comienzan a integrarse a esta normatividad conocida como modelo OSI, ya que ésta también les permite mantener una compatibilidad con sus productos en sus diferentes versiones y modelos.

La primeras ventajas para el usuario que se presentan al seguir este modelo son:

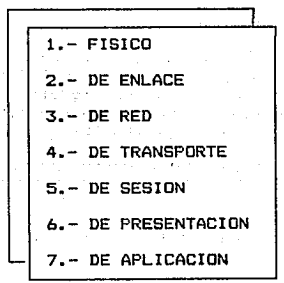
- Independencia del fabricante ya que al contar con equipo de cómputo compatible el usuario puede recurrir a cualquier otro fabricante que ofrezca productos normalizados

- Compatibilidad completa con los nuevos equipos y versiones de software y hardware que aparezcan en el mercado

- Facilidad de expansión del sistema con una basta gama de opciones en los accesorios

En torno al diseño de herramientas o elementos de software y hardware tenemos la modularidad, ya que cada nueva sección que se desarrolla puede partir de alguna de las capas que establece el modelo, sin importar que existe arriba o abajo de la capa donde se ubique la aplicación generada.

El modelo consta de siete áreas que aunque hacen referencia a diferentes puntos están mutuamente relacionadas. Los niveles o áreas se ilustran a continuación:




M O D E L O

O S I

Nivel 1 Fisico

Hace referencia a las características mecánicas, eléctricas y electrónicas de los periféricos y conexiones usados para el transporte de la información.

Nivel 2 Enlace

Se refiere a las características con las cuales se transporta la información en forma de bits. Dentro de esto se pide se siga el formato de Control de Alto Nivel para el Enlace de Datos (High Level Data Link Control, HDLC). En sí, especifica las características del protocolo de comunicación.

En terminos generales sugiere manejar una bandera de inicio, un segmento de control, un segmento para destinatario, un segmento para información transmitida, un segmento destinado a la paridad y otro como bandera para señalar el fin de cada paquete a transmitir.

Nivel 3 Red

Bajo este nivel se dan las características en que se está transmitiendo e interpretando la información, tiene un carácter más lógico que físico, en contraste con el nivel anterior.

Nivel 4 Transporte

En este nivel se caracteriza al software básico (sistema operativo y/o software de aplicación), bajo el cual se estará llevando a cabo la comunicación.

Nivel 5 Sesión

En el se describen los rasgos que tiene la comunicación en torno a la sincronización, el control y su establecimiento. Una sesión es un estado de enlace entre un usuario y el sistema.

Nivel 6 Presentación

Señala los formatos con los cuales los usuarios interactúan con el sistema, buscando la transparencia y sencillez para el usuario.

Nivel 7 Aplicación

Contempla el conjunto de programas y procesos que dan servicio a los usuarios. Entre los más comunes actualmente está el Correo Electrónico y los manejadores de base de datos, así como una gran diversidad de paquetes orientados a la graficación, reconocimiento de patrones y control de textos.

El punto fundamental que determina la importancia de seguir esta concepción es la de permitir ver dentro de un panorama más claro y sencillo cada uno de los elementos de cualquier sistema de red ya que de esta forma se localiza con mayor prontitud en que nivel se tienen problemas o que nivel específico se puede optimizar en forma independiente.

## 1.6 HACIA LA COMUNICACION AUTOMATICA

La automatización tiene sus primeros pasos desde que el hombre se dió cuenta que provocando ciertas condiciones, un número de eventos secuenciales ocurrirían a continuación; por ejemplo al orillar venados a un barranco, éstos al seguir su carrera, caerían y morirían. Las condición provocada sería la caída libre, y los eventos provocados caer y morir.

Definitivamente es un caso poco detallado en automatización, hoy en día el nivel de sofisticación es mucho mayor; ahora se tiene la producción en serie, muestra de tal eficiencia.

Automatizar implica hacer que los eventos se realicen por si mismos o con la mínima intervención humana. Existe una diversidad de niveles y aplicaciones de la automatización, niveles y tipos que se ven justificados por alguna(s) de las siguientes razones:

- Eliminar las tareas peligrosas y/o penosas haciendo ejecutar a la máquina las tareas humanas complejas, poco gratas o repetitivas.

- Mejorar la productividad, sometiendo a la máquina a criterios de producción, de rendimiento o calidad.

- Controlar una producción variable, facilitando al hombre pasar de una producción a otra.

- Reforzar la seguridad, vigilando y controlando las instalaciones y máquinas.

Concretando la automatización tiene como fin obtener soluciones a problemas de tipo teórico y/o práctico en beneficio del hombre a través de la ejecución de tareas, sin la intervención directa del hombre.

Tomando como base tales premisas el hombre puede hablar hoy de los viajes e investigación espacial, del control de termoeléctricas, de la producción en serie, de semáforos inteligentes y equipos de vida artificial, entre una serie de aplicaciones que ya son una realidad, y que continuamente se buscan mejorar.

Ahora bien, en todos estos casos se requiere de la transmisión de señales de un subsistema a otro para su interrelación, es por ello que podemos hablar ya de una comunicación automática intríseca entre cada elemento, misma que se está perfeccionando con propuestas que son rechazadas, mejoradas o que dan pauta a nuevos métodos y técnicas.

La comunicación automática entre los seres humanos ya no radica en la leyenda, el cuento y los trovadores, es tanta la información que es necesario un almacenamiento masivo y un control eficaz en torno a su transmisión.

Hablar de la automatización de la comunicación es hablar de la misma existencia del hombre. El decir comunicación automática, es hacer énfasis en la busqueda de mecanismos que permitan intercambiar datos y señales entre distintos elementos que no necesariamente sean del tipo humano, pero que le sirvan a éste.

Comentar algo más acerca de tal situación sería entrar a un juicio de que tan eficiente es la comunicación, y al por qué le podemos manejar como una comunicación automática. En lugar de ello resta mencionar que la eficiencia de ésta se ve reflejada y limitada por el uso que le demos, y que su optimización se está aumentando por el uso y aplicación de las computadoras.

La comunicación automática entre computadoras hace referencia al conjuto de software y hardware que permite enviar información tangible y útil al ser humano de una estación de trabajo a otra con la mínima intervención humana.

Bajo tal filosofía se sigue la presentación del sistema, el cual se ubica dentro de las capas 4, 5 y 6 del Modelo OSI, y se puede revisar en los siguientes capítulos.

*COMUNICACION AUTOMATICA ENTRE*

*COMPUTADORAS PERSONALES*

# CAPITULO 2

DESCRIPCION DEL SISTEMA PROPUESTO

2.1 Introducción.

2.2 Factores de Software y Hardware.

2.3 Correo Electrónico y Control Remoto.

2.4 Análisis de las circunstancias actuales.

2.5 Descripción del sistema.

# 2 DESCRIPCION DEL SISTEMA PROPUESTO

## 2.1 INTRODUCCION

Este capítulo está conformado en base a dos aspectos: Uno, el referente al marco teórico y práctico donde se ubica el sistema, y otro, en donde se describe propiamente.

El marco, queda integrado por los productos similares (o relacionados) existentes y su análisis tomando como referencia los servicios que prestan. En torno a la descripción se detalla cada una de las funciones del sistema propuesto.

## 2.2 FACTORES DE SOFTWARE Y HARDWARE

Dentro de los elementos de Software (S) y Hardware (H), tenemos los ilustrados en las figuras 2.1 y 2.2, en la página siguiente.

Cada uno de ellos tiene una un papel especializado y un nivel de importancia. Por el lado del S, el Sistema Operativo (SO) y los Lenguajes de Programación (LP), tienen los dos niveles más altos, y de los dos el SO se encuentra arriba. El SO, marca hoy una línea de productos de S y H como alternativas de uso una vez que se ha seleccionado.

En este caso, se seleccionó DOS (MS vers. 3.01 o posterior), en consecuencia, el programa ejecutable puede correrse sobre cualquier arquitectura que soporte dicha versión, y específicamente en cualquier computadora compatible con el estandar de IBM.

# ELEMENTOS DE SOFTWARE

PANORAMA GLOBAL

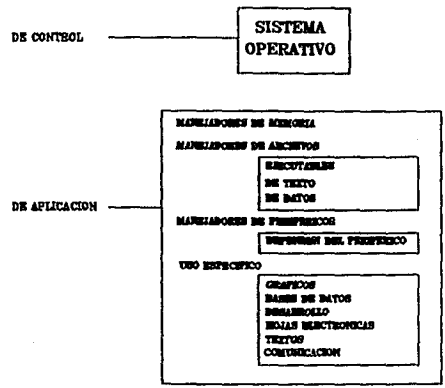



FIGURA 5.1

# ELEMENTOS DE HARDWARE

PANORAMA GLOBAL

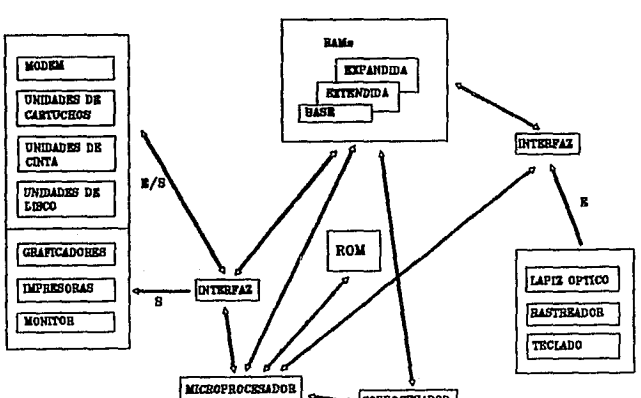



FIGURA 5.2

Los LP, permiten interactuar más directamente con la computadora y por ello juegan un papel importante, el Lenguaje C, es uno de los más difundidos y aceptados por su modularidad y facilidad de manejo tanto de instrucciones como de macroinstrucciones.

El DOS, fue diseñado para aprovechar los recursos del 8088, microprocesador que ha ido evolucionado hasta llegar al 80586. Tómese como muestra comparativa que el 8088 trabaja entre 4 y 8 Mhz, y el 80586 en el orden de los 50 Mhz. Lo importante de mencionar todo esto es que gracias al auge de esta familia de microprocesadores, tanto el DOS, como el Lenguaje C, coexisten y trabajan en conjunto en un sin fin de aplicaciones. Por ejemplo, los productos XENIX, FOXPRO, ORACLE, DBASE, CLIPPER, entre otros, fueron implementados en Lenguaje C, para trabajar sobre plataformas con DOS.

Lo anterior, no implica que la última palabra sea DOS, ni C, ni las PC's basada en la familia 80xxx, no, pero hoy son una realidad y por su difusión un patrón presente en el diseño y desarrollo de cualquier proyecto, tanto de software como de hardware.

Por último, en el lado del H, es también importante resaltar dentro de este trabajo la existencia y utilidad del MODEM y la Línea Telefónica. El primero, se maneja como una caja negra que funciona como interfaz entre la computadora y el canal de comunicación que es la línea telefónica. Esta combinación permite manejar una velocidad de 2400 baudios, se puede incrementar ésta a 9600, mas ello implica un alto costo en la adquisición del MODEM. Los MODEM de 300 a 2400 baudios tienen un precio al público que va de los 100,000 a los 500,000 pesos, y los que permiten el manejo de velocidaddes mayores y diferentes estandares, tienen un precio entre los 900,000 y los 3,000,000 de pesos.

El objetivo de tratar estos elementos, no es definirlos, sino tan sólo presentarlos como partes fundamentales en el desarrollo de la aplicación. Información en torno a la selección de equipos de cómputo y rasgos generales de microprocesadores se pueder encontrar en los apéndices C y B respectivamente.

## 2.3 CORREO ELECTRONICO Y CONTROL REMOTO

El correo electrónico es una utilería que permite enviar y recibir información entre PC's. Con ella el usuario no puede realizar otra actividad a parte de la de esperar el enlace y efectuar la transmisión. Los sistemas de este tipo requieren entre 40 y 150 Kb de memoria RAM de base, algunos además requieren del uso de la expandida. Además es indispensable que las computadoras sean sincronizadas por los usuarios en cada punto, y obviamente que ambos usuarios tengan las mismas características de comunicación (velocidad de transmisión, protocolo, paridad, y banderas de control), en caso contrario la comunicación será imposible. Otras características de estos paquetes son las siguientes:

- Está orientado su uso dentro de una red, y a usuarios con acceso a MODEM y línea telefónica.

- Las terminales quedan atadas durante el enlace a esa actividad.

- El tipo de correo electrónico depende del equipo que se tenga, los sistemas multiusuario suelen incluir este servicio, pero es un servicio propio del sistema específico.

- Si el enlace es remoto, implica un tiempo largo de ejecución, dado que su nivel de prioridad dentro de una red por lo regular es de los más bajos.

- Esta destinado al envío de pequeños escritos.

Apesar de no ser algo novedoso a últimas fechas está tomando cierta difusión, a futuro quiza desaparezca el correo postal. Aunque no se debe olvidar que su uso esta orientado a redes, lo cual le agrega sus ventajas y desventajas.

El Control Remoto es un paquete que permite manejar una PC a distancia (pc esclava), desde otra (pc maestra), como si fuera la misma. Por ejemplo se puede accesar al disco duro y mandar el contenido de un archivo de la esclava, a un dispositivo de salida de la maestra.

En estos sistemas, la PC maestra coordina todas las operaciones de la esclava provocando que una vez enlazada, a menos que se reinicialice la esclava, quedará atada a la maestra hasta que sea liberada.

En el caso del control remoto, mas que transferencia de información su uso tiene concreción en el aprovechamiento de sistemas propios los cuales no se pueden transportar, mas sí se pueden controlar a distancia desde otro pequeño o sencillo, pero portable, teniendose así las ventajas del equipo fijo. En general el Control Remoto hace que la esclava reciba las instrucciones desde el teclado de la maestra y los resultados sean enviados a los dispositivos que le indiquen, por default, la pantalla de la maestra.

Para su funcionamiento, se requiere de la línea telefónica, el MODEM, y que ambos equipos (pc maestra y esclava) se encuentren ejecutando el Control Remoto específico. La maestra llama a la esclava y ésta deshabilita su teclado, por ello en caso de algun error o deseo de terminar con el acceso remoto es necesario: a) La liberé la maestra (Considerando que no hubo errores), b) Reinicializar a la esclava.

## 2.4 ANALISIS DE LAS CIRCUNSTANCIAS ACTUALES

Existen dos factores básicos y suficientes que señalan las características que debe tener un sistema de comunicación:

La u t i l i d a d

El c o s t o

La utilidad, todos buscamos acrecentarla y resulta más económico y posible para un usuario individual contar con un sistema de bajo precio que le permita realizar operaciones análogas (en torno a la transmisión de información) a las que podría ejecutar en una terminal (tonta o inteligente) conectada a un sistema multiusuario, que contar con un sistema de este tipo.

La idea central es transferir información y siguiendo la filosofía *beneficio/costo*, resulta obvio que a una empresa mediana o grande le es conveniente tener un sistema multiusuario, pero no resulta igual para un usuario independiente.

A manera de ilustrar los costos de algunos sistemas multiusuario se incluye a continuación una lista de precios de Agosto de 1990*...

"Escala general de precios. Para propósitos de comparación, en números redondos, precios de red para sistemas de diez usuarios (con NetWare 386 en los LANs y SCO Unix en los sistemas Unix).

# US$21,000: Sistema Unix basado en la IBM PS/2 Modelo 80.
# US$24,000: LAN sólo de ZEOS con un servidor 386 de 25 MHz con 5 Mb de RAM y disco duro de 400 Mb, ocho 386SXs, una 386 de 20 MHz con un disco duro de 20 Mb, una 386 de 25 Mhz con un disco duro de 80 Mb.
# US$25,000: DEC MicroVAX 3100 con la mayor cantidad de periféricos de terceros fabricantes posibles.
# US$31,000: Micro VAX 3100 con todos los equipos de DEC.
# US$38,000: Sistema Unix basado en la Compaq Systempro.
*# US$47,000: LAN basado en el modelo 80.
*# US$64,000: LAN basado en la Systempro.
# US$96,000: LAN basado en sistemas Compaq.
# US$60,000: Configuración SCO Unix basada en la Systempro # US$100,000: LAN totalmente de Zeos con un servidor 386 de 25 Mhz con 8 Mb de RAM y 800 Mb en disco duro, más 50 nodos 386SX.
# US$128,000: MicroVAX 3800 con todos los periféricos de terceros fabricantes posibles.
# US$160,000: MicroVAX 3800 con todos los equipos de DEC.
# US$226,00: Systempro con 50 estaciones Compaq Deskpro 286 (monitor monocromático, sin discos duros).
# US$282,000: Systempro con 25 estaciones Deskpro 286 y 25 estaciones Deskpro 386/20."

*NOTA: Esta información fue extraida de la revista PC Magazine en español de Agosto de 1990 (Vol. 1, No. 5,) "¿ Pueden las LANs vencer a las Minis ?", pags. 75-85. Las opciones con '*' incluyen estaciones de trabajo del tipo 286 y 386, cinco y cinco.

De la relación anterior, se observa que para un usuario independiente, el cual busca intercambiar información y quiza compartir periféricos, cualquiera de estas opciones por muy ventajosa o económica que sea, resulta un gasto excesivo y no productivo.

Si en lugar de ello optará por conectarse a una red de usuarios personales (clubs), ya existente (en el apéndice A se da una lista de ellos) requeriría como mínimo de un MODEM, una línea telefónica y un paquete de software que sólo le permitirá entrar a una red específica, pagando una cuota por ello y quedando limitado a las reglas que ésta le imponga, como:

- Para bajar información sólo en horarios nocturnos, después de la 1:30 de la mañana y hasta las 5:30.
- No se permite intercambiar información únicamente entre dos usuarios.
- No se permite subir al sistema paquetes comerciales.
- La información que se carga al sistema queda sujeta al juicio de los administradores.
- Todos los usuarios tienen una clave que les permite tener cierto nivel de servicio. Esta clave y el nivel queda determinado por los administradores del sistema.

Estas políticas son en el caso de que la red sea un club público de usuarios (BBS), ya que al hablar de las redes en compañías específicas el acceso y servicios están más limitados y orientados a un fin que es el de la compañía y no el del usuario.

Los clubs intentan solucionar el problema, sin embargo no tardan mucho en volverse elitistas, favoreciendo sólo a los usuarios que puedan brindar información, esto es a los usuarios con conocimiento de computación y dejando fuera a los usuarios incipientes y/o de bajos recursos.

Otra opción son las redes de universidades o centros culturales que prestan un servicio al público en general, pero uno de los primeros problemas que presentan es el horario en que una persona puede conectarse, a parte del costo por clave y la alta demanda de este servicio. Entre las instituciones que cuentan con un servicio de este tipo son: La UNAM, el IPN, el CONACyT, la Biblioteca Nacional de México y la UAM. Para la consulta de información y con tiempo de sobra, cualquiera de estos lugares es una buena alternativa.

Hasta este punto, los sistemas multiusuario, los BBS, y los productos de correo electrónico y control remoto, permiten la transferencia de información, pero en todos ellos el problema común es la accesibilidad, definida por las políticas de una organización y/o por su costo.

Concretando, estos sistemas no brindan el servicio que requiere un usuario común:

- Intercambiar información en el momento y con las condiciones que él establezca o tenga.

Han aparecido una serie de utilerías del tipo correo electrónico entre los que se encuentran los siguientes:

- Blast Profesional V 10.5.1
- Carbon Copy Plus V 6.0
- Central Point Commute V 1.1
- Close-Up V 4.0
- ReachOut V 2.1
- Remote$^2$ V 6.01
- TakeOver Lite V 2.0
- PCAnyWhere

de acuerdo a la revista PC Magazine en Español (vol. 1 # 3), el costo de ellos oscila entre los 250 y 700 Dolares.

Se podría comprar alguno de ellos para cada PC en donde se deseará utilizar, se requieren al menos de dos licencias para transferir información, a parte del MODEM y el uso de la línea telefónica. Aunado a ello, siempre que se lleve a cabo una tranferencia se tendrían los siguientes problemas:

- Espera por enlace.
- Frecuentes caídas de la línea telefónica implicando tener que limpiar archivos incompletos y reintentar.
- Incompatibilidad de MODEM's.
- Incompatibilidad de protocolos, sólo si se utilizan diferentes versiones o paquetes de comunicación.
- Pérdida de tiempo del usuario, principalmente por la existencia de errores, y la supervisión de las operaciones de marcación, espera por respuesta, selección de archivo, etc.

Y es precisamente aquí donde entra el sistema que se describe acontinuación, en la solución de estos problemas a través de la automatización de tranferencia de archivos.

## 2.5 DESCRIPCION DEL SISTEMA

Para su funcionamiento se requiere de al menos dos PC's compatibles con monitor de cualquier tipo, espacio en RAM de 640 KB, SO vers 3.01 o posterior. Un MODEM para cada PC y la línea telefónica común.

La figura 2.3 (página siguiente) es un esquema de conectividad del Sistema de Comunicación Automática Entre Computadoras Personales. La descripción de los módulos principales es la que sigue:

Proceso de Comunicación. Con él, la computadora atiende peticiones de otras computadoras para llevar a cabo algún intercambio de información, en caso de no sensar alguna llamada y no tener que enviar, seguirá sensando hasta que sea desactivado. También se encarga de checar lo que recibe a través del MODEM, así como de enviar la información correspondiente a otra(s) computadora(s) señalada(s).

Las características de funcionamiento son:

- Ejecusión en el momento que el usuario lo solicite.
- Bitácora de operaciones (Envíos, Recepciones).
- Control de destinos y archivos de envío.
- Control de envíos y recepciones.
- Interfaz para auxilio al usuario para definir archivos y destinos.

En si, llama a programas orientados a la recepción y transmisión de información mientras el usuario no solicite el uso de los recursos.

Procesos del usuario. Los programas del usuario se ejecutan independientemente a los relacionados con el sistema de comunicación. Se entra al modo automático con F12 y se regresa a DOS presionando cualquier tecla. Si se presiona 'u', además de regresar, se descarga el TSR.

COMUNICACION AUTOMATICA

ENTRE COMPUTADORAS PERSONALES

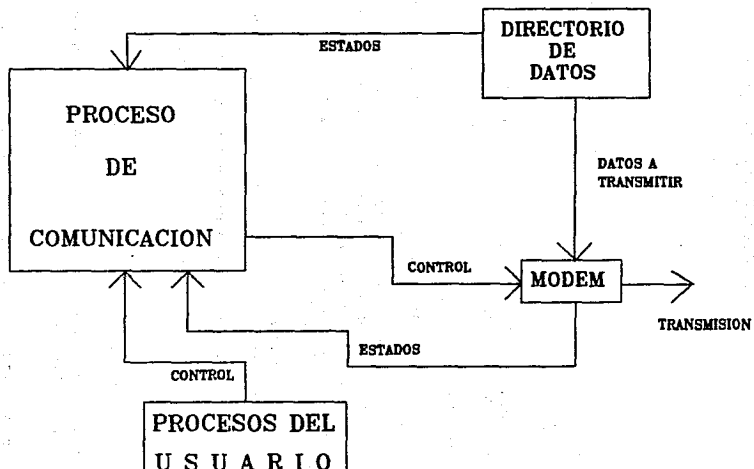



FIGURA 2.3

Esta sección, no tiene por objeto explicar a fondo el funcionamiento del sistema, sino tan solo describir sus elementos, y su tarea principal que puede resumirse como sigue:

> El sistema se encarga de establecer los enlaces, realizar el intercabio de archivos -ya sea llamando o siendo llamado-, permite seguir trabajando al usuario normalmente, y cuando éste decide llevar a cabo la tranferencia, presiona F12 y el modo automático comenzará.

No debe olvidarse las características y ventajas que se buscan, el decir establecer comunicación y realizar la transacción, suena sencillo, pero eso es precisamente por el grado de automatización que se está alcanzando.

# COMUNICACION AUTOMATICA ENTRE COMPUTADORAS PERSONALES

# CAPITULO 3

DESARROLLO DEL SISTEMA

3.1 Introducción.

3.2 Seudocódigo.

3.3 Diagrama de flujo general.

3.4 Módulos y funciones.

3.5 Listado de programas.

# 3 DESARROLLO DEL SISTEMA

## 3.1 INTRODUCCION

El contenido de este capítulo consiste en la descripción detallada de módulos, funcionamiento y listado de los programas integrantes del sistema. Cada una de las formas representativas se dá como un punto independiente, por lo que se puede pasar directamente a cada una de las secciones (listados, diagrama, seudocódigo o módulos y funciones), o se pueden seguir una a una de acuerdo a la secuencia establecida.

## 3.2 SEUDOCODIGO

```
INICIO:
     - Opción.
     - Actualización de archivos y destinos.
     - Carga TSR de comunicación.
     - Fin.

FIN:
     - Liberación del sistema.


ACTUALIZACION DE ARCHIVOS Y DESTINOS:
     - Checa existencia de archivos.dat
          - No existe:
               - Crea archivos.dat
     - Inserción.
     - Modificación.
```

CARGA TSR DE COMUNICACION:
- Levanta TSR.
- Pasa control a DOS y sensa F12 para ejecutar TSR. (LLAMADA y ENVIO, una vez que se presionó F12)

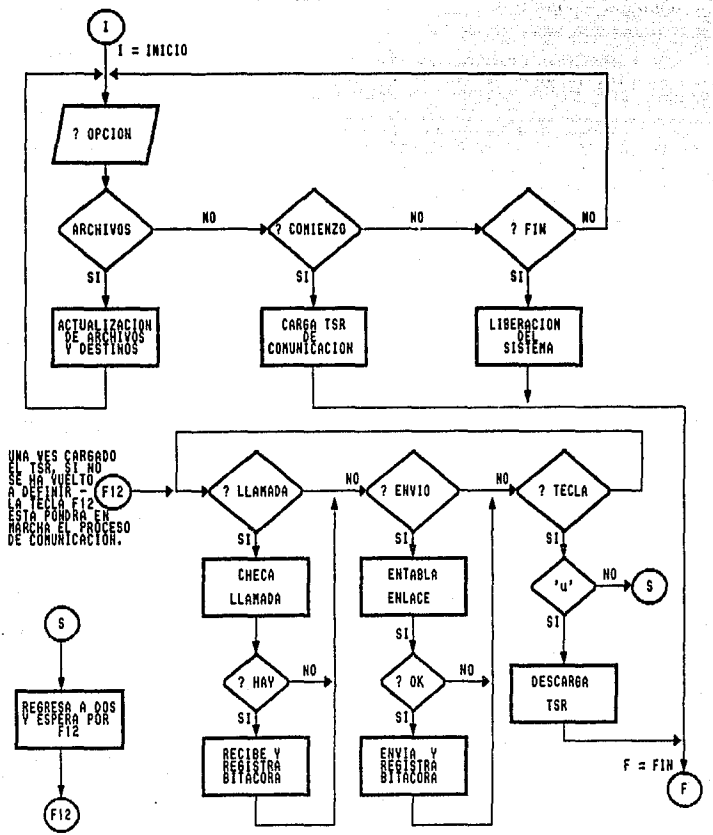
Una vez activado el TSR con F12, realiza continuamente:

LLAMADA:
- Checa llamada,
  - Hay:
    - Recibe.
    - Registra Bitácora.
    - Regresa nivel.
  - No hay:
    - Regresa nivel.

ENVIO:
- Checa si hay envío,
  - Hay:
    - Entabla enlace.
      - Hay:
        - Envia.
        - Registra Bitácora.
        - Regresa nivel.
      - No hay:
        - Regresa nivel.
  - No hay:
    - Regresa nivel

SENSA TECLA (si se presionó alguna tecla):
- Se presionó 'u', baja TSR.
- Regresa a la línea de comandos de DOS.

## 3.3 DIAGRAMA DE FLUJO GENERAL



## 3.4 MODULOS Y FUNCIONES

OPCION. Función que se encarga de aceptar una petición del usuario, para pasar al manejo de los archivos a transmitir, al cargado del TSR y salida, o solamente salir.

CARGA TSR DE COMUNICACION. Este modulo se encuentra latente una vez que es activado. Se encarga de atender las peticiones del usuario en torno a la comunicación, checa si hay algo que enviar y realiza las operaciones pertinentes, así mismo, checa si llama alguna otra computadora remota y le atiende. Para esto, invoca a dos programas correspondientes a la recepción y al envío de archivos. Ejecuta repetitivamente estos mientras no se presione alguna tecla; en tal caso, si es una 'u' regresa a DOS y baja el TSR, si es cualquier otra regresa a DOS, pero el TSR queda en memoria y se puede volver a ejecutar con F12.

ACTUALIZACION DE ARCHIVOS Y DESTINOS. Módulo que permite la administración de archivos y destinos de envío, cuenta con inserción y borrado de registros, con estas opciones se pueden llevar a cabo las altas, bajas, cambios y consultas en forma no directa, pero se ahorra código y agiliza su operación.

LIBERACION DEL SISTEMA. Módulo que se encarga de bajar el programa principal y grabar los actuales valores de archivo para las subsecuentes transacciones de envío.

RECEPCION. Módulo especializado para recibir archivos provenientes de otra computadora. Incluye sensado de llamada de la otra computadora con el mismo sistema. Se encuentra en RECIBE.C, uno de los programas que se ejecutan desde el TSR.

ENVIO. Módulo especializado para efectuar la transferencia de archivos a otra(s) computadora(s), el destino está definido en base al número telefónico del destinatario, que es almacenado en 'archivos.dat'. Tiene un submódulo que se encarga de entablar la comunicación, en el diagrama y aquí se ha manejado independientemente debido a la importancia que le reviste en el proceso de la comunicación automática.

ENTABLA ENLACE. Dentro de este módulo se llevan las operaciones de marcación y espera por enlace vía telefónica a otra computadora. Se encuentra en el programa ENVIA.C, que corresponde al módulo ENVIO, el cual es llamado desde el TSR.

REGISTRA BITACORA. Luego de cada recepción o envío, esta rutina se encarga de registrar la transaccion, escribiendo que archivo fue transmitido, si fue exitosa la operación, y en que hora fue realizada; la hora es tomada del sistema de cada usuario; la rutina es utilizada tanto en ENVIA.C como RECIBE.C, despues de realizada alguna transferencia y el registro se hace en el archivo BITACORA.DAT.

DESCARGA TSR. Este proceso es llevado a fin por el mismo TSR. Se ha presentado en un bloque separado para hacer más explicita su ubicacion, el TSR es descargado cuando durante su ejecucion se presiona 'u', cuando termina el proceso actual, se baja el TSR y pasa el control a la línea de comandos de DOS. La presión de cualquier otra tecla ocasiona pasar a DOS, pero sin descargarlo.

## 3.5 LISTADO DE PROGRAMAS

Esta sección queda integrada por los programas fuente, de los módulos que componen el sistema. Cada uno de ellos incluye su documentación dentro del listado. La información de la utilería de comunicación de LiteComm se encuentra en el apéndice E.

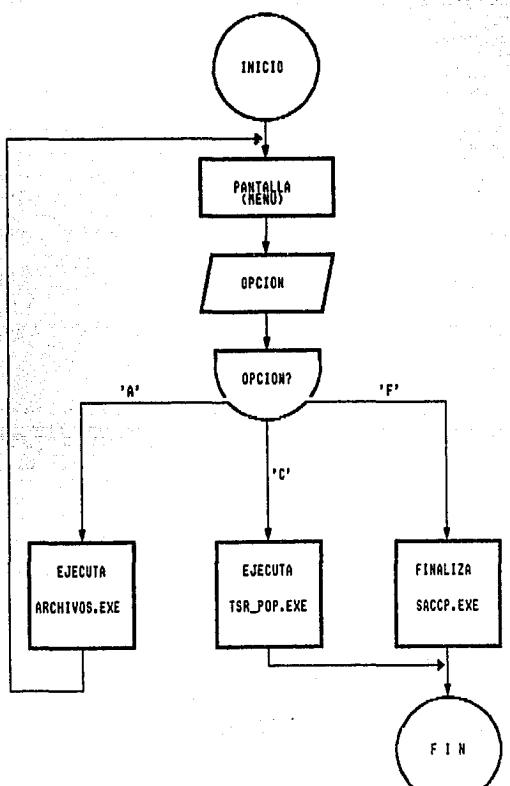
La descripción general de cada listado es la siguiente:

(1) SACCP.C Arranque del sitema, llama a 2 y 3.
(2) ARCHIVOS.C Permite el control del archivo de envíos y destinos (archivos.dat).
(3) TSRPOP.C Carga y descarga el módulo residente, TSR, que controla el modo automático, llama a 4 y 5.
(4) RECIBE.C Checa, atiende llamadas para recibir archivos y actualiza bitácora.
(5) ENVIA.C Checa lista de envíos (archivo.dat), establece enlaces, envía y actualiza bitácora.

Librerías definidas para el sistema:

(6) ENLACE.H Parámetros, constantes y variables del sistema.
(7) APOYO.H Lectura de campos y desplegado de mensajes.
(8) LCBBS.H Funciones de comunicación.

DIAGRAMA DE FLUJO (PROGRAMA: SACCP.C)



```
/* ************************************************************************/
/*                  Programa principal: BACF.C                            */
/* ************************************************************************/

#include <ctype.h>
#include <process.h>
#include <stdio.h>
#include "apoyo.h"

#define comienzo system("tsr_pop.exe");
#define limpia   system("exit");
#define graba    printf("\n Respaldo de trabajos pendientes ");
#define avisa    printf("\n Sistema Cargado");
#define archivos system("archivos.exe");

void pantalla(void);
char opcion='F';
main()

{
cls;
do
   {
    pantalla();
    while ( strchr( "ACF", opcion=toupper(getch())  ) == NULL  ) {};
    gotoxy(20,20);
    switch(opcion)
        {
        case 'A' :{cls;archivos;          break;}
        case 'C' :{cls;limpia  ;comienzo;break;}
        case 'F' :{cls;graba   ;          break;}
        }
    cls;
    }
while(opcion != 'F' && opcion != 'C');
exit(0);
}
void pantalla(void)
{
int x=5,y=2;
 int i,c,b;
 i=2;c=7;b=1;
```

```
 xy_txt(x+2,y,
"S I S T E M A  D E  C O M U N I C A C I O N  A U T O M A T I C A",i,c,b);
 xy_txt(x+6,y+2,"                    E N T R E",i,c,b);
 xy_txt(x+2,y+4,
"C O M P U T A D O R A S  P E R S O N A L E S",i,c,b);
i=0;x=x+10;b=1;y=y+4;
 xy_txt(x,y+8,"A ---> Actualizacion de archivos y destinos",i,c,b);
 xy_txt(x,y+10,"C ---> Comienzo del Proceso Automatico",i,c,b);
 xy_txt(x,y+12,"F ---> Fin",i,c,b);
 xy_txt(27,24,"S. L. A. I. C.  1992",2,7,0);
 xy_txt(x,y+14,"Opcion ----> [ ]\b\b",1,7,0);
}
```

DIAGRAMA DE FLUJO (PROGRAMA: ARCHIVOS.C)

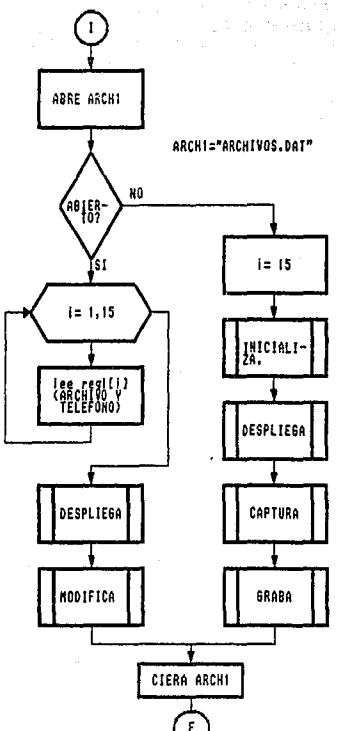
INICIALIZA: LIMPIA reg[i].

DESPLIEGA: MUESTRA EN PANTALLA -- MODO TEXTO LOS VALORES DE reg[i].

MODIFICA: PERMITE CAMBIAR EL VA-- LOR UNO A UNO DE reg[i].

CAPTURA: ACEPTA LOS VALORES DE -- reg[i], PARA GENERAR ARCHI.

GRABA: ALMACENA EN DISCO LOS VA-- LORES DE reg[i].

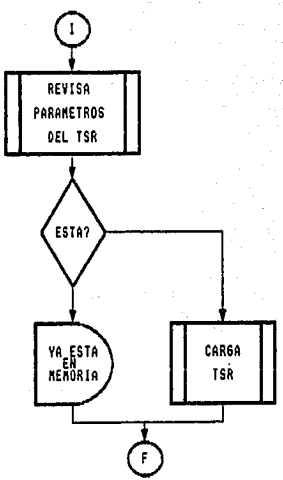


```
/* ##########################################################################
   ARCHIVOS.C
   SE ENCARGA DEL MANEJO DE NOMBRES DE ARCHIVO Y TELEFONOS A LOS
   CUALES SE ENVIARAN. LA SENCILLEZ DE ESTE MODODULO ES COMPEN--
   SADA POR SU VELOCIDAD, Y POR SU POCO ESPACIO DE MEMORIA NECE-
   SARIO PARA SU EJECUCION.
########################################################################## */

#include <stdio.h>
#include <stdio.h>
#include "apoyo.h"
#include "enlace.h"

#define cont_inicial "1 "

void captura(void);
void inicializa(void);
void despliega(void);
void modifica(void);
void graba(void);

struct registro  regl[15];
int i=0,z=0;
char resp;
FILE *fl;

/* actualizacion de archivo de comunicaciones */
main()
{
cls;
i=0;
if ( (fl = fopen(archi,"r")) != NULL )
        {
        while( eof(fl)!=0 && i<15)
                {
                fscanf(fl,"%s%s",regl[i].arcnivo,regl[i].telefono);
                i++;
                }
        despliega();
        modifica();
        }
```

```
    else
        {
        /* No existe el archivo */
        i=15;
        inicializa();
        despliega();
        captura();
        graba();
        };
        fclose(f1);
}
void captura()
{
int cont=0;
resp='S';
while (resp == 'S' && cont <15)
        {
         xy_txt(7,14,"Nombre de archivo: ",1,7,0);
         xy_txt(7,18,"Telefono: ",1,7,0);
         lee_cadena(26,14,12,0,15,regi[cont].arcnivo);
         lee_entero(17,18,10,0,15,regi[cont].telefono);
         cont++;
         xy_txt(8,22,"Desea continuar (S/N)  \b\b");
         while( strchr( "SN",resp=toupper(getch()) ) == NULL ) {};
         cls;
         despliega();
        }
}
void inicializa(void)
{
for(z=0;z<i;z++)
                {
                strcpy(regi[z].archivo,arch_vacio);
                strcpy(regi[z].telefono,tel_vacio);
                }
}
void despliega(void)
{
int pos_x=45,pos_y=5;
xy_txt(5,3,"SISTEMA DE COMUNICACION",2,0,15);
xy_txt(5,4,"         ENTRE          ",2,0,15);
xy_txt(5,5,"COMPUTADORAS PERSONALES",2,0,15);
xy_txt(pos_x,pos_y,"TELEFONO     ARCHIVO",2,0,15);
```

```
for(z=0;z<i;z++)
                {
                gotoxy(pos_x-6,  z+pos_y+1);printf("%2d--)",z+1);
                xy_txt(pos_x,    z+pos_y+1, ( regi[z].telefono ), 1,0,15);
                xy_txt(pos_x+13, z+pos_y+1, ( regi[z].archivo ), 1,0,15);
                }
}
void modifica(void)
{
int cont=0;
char cont_tempo[3]=cont_inicial;
resp='S';
while (resp == 'S')
        {
        xy_txt(5,10,"No. de registro a modificar: ",1,7,0);
        xy_txt(5,14,"Nombre de archivo: ",1,7,0);
        xy_txt(5,18,"Telefono: ",1,7,0);
        while( 0)=cont || cont>15)
                {
                lee_entero(34,10,2,0,15,cont_tempo);
                cont=atoi(cont_tempo);
                }
        lee_cadena(24,14,12,0,15,regi[cont-1].archivo);
        lee_entero(15,18,10,0,15,regi[cont-1].telefono);
        cont=0;
        xy_txt(8,22,"Desea continuar [S/N]  \b\b");
        while(  strchr( "SN",resp=toupper(getch()) ) == NULL ) ();
        cls;
        despliega();
        }
        graba();
}
void graba(void)
{
f1=fopen(archi,"w");
        for (z=0;z<i;z++)
            fprintf(f1,"%s %s \n",regi[z].archivo,regi[z].telefono);
}
```

DIAGRAMA DE FLUJO (PROGRAMA: TSR_POP.C)




CARGA: INVOCA A RUTINAS PARA EL CONTROL DE LA PARTE RESIDENTE,

- MODO DE DESPLEGADO ACTUAL (TEXTO O GRAFICO),
- RESPALDO DE LA ACTUAL ASIGNACION DE F12 Y MODIFICACION PARA ACTIVACION DEL TSR,
- ACTIVACION DE UN MANEJADOR DE INTERRUPCIONES, ANTES DE PASAR A DOS, LAS INTERCEPTA EL SISTEMA PARA NO ABORTAR SU EJECUSION (Ctrl-C, Ctrl-Break, ACCESO A DISCO).
- COLOCACION EN MEMORIA DE RUTINAS Y PRINCIPAL,
  - EL PRINCIPAL ACTIVA RUTINAS PARA DETECCION DE MODO, RESPALDO DEL MISMO, EJECUSION ALTERNADA DE -- LOS PROGRAMAS DE COMUNICACION ENVIA.C Y RECIBE.C
  - RUTINAS PARA CONTROL DE TECLADO, AL SENSAR UNA -- TECLA DIFERENTE DE 'u', REGRESA A DOS, CON LAS -- CARACTERISTICAS DE ARRANQUE DE LA PARTE RESIDENTE.
  - RUTINAS PARA DESCARGA DE LA PARTE RESIDENTE Y REGRESO A DOS CON EL AMBIENTE DE ARRANQUE.

```
/* ************************************************************************
   MODULO TRS_POP.C
   Se encarga de montar, desmontar  un TSR,  que llama a 2 programas
   definidos en ENLACE.H (RECIBE.EXE Y ENVIA.EXE), una vez que se ha
   levantado el TSR, los programas se ejeutan continuamente al pre--
   sionar F12, hasta presionar otra tecla. Si es 'u', descarga la --
   parte residente, si es cualquier otra regresa a DOS.
** ************************************************************************/
#undef DEBUG
#include <dos.h>
#include <bios.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include  "enlace.h"
typedef char boolean;
#define TRUE 1
#define FALSE 0

/* Declaracion de vectores de interrrupcion ***************************/
#define INT_TIMER          0x6     /* Tiempo,reloj                    */
#define INT_KEY            0x9     /* Teclado, sensado de             */
#define INT_VIDEO          0x10    /* Video, informacion              */
#define INT_DISK           0x13    /* I/O discos,                     */
#define INT_BREAK          0x1B    /* Manejador de salida  ESC        */
#define INT_CONTROL_C      0x23    /* Manejador de ^C                 */
#define INT_CRITICAL       0x24    /* Manejador de errores criticos   */
#define INT_IDLE           0x28    /* DOS Idle interrupt              */
#define INT_DOS            0x21    /* Llamadas al dos                 */

/* Codigos de funcion para interrupcion de video **********************/
#define VIDEO_SET_CURSOR_SIZE      0x1       /* Define tamano del cursor  */
#define VIDEO_SET_CURSOR_POSITION  0x2       /* Posiciona al  cursor      */
#define VIDEO_GET_INFO             0x3       /* Proporciona tamano y posi-*/
                                             /* cion del cursor           */
/* Posiciones para almacenamiento del modo de video */
#define MODE_SEG 0x40
#define MODE_OFFSET 0x49
/* Codigos de funcion de interrupcions del dos (INT_DOS) **************/
#define DOS_GET_DOS_BUSY 0x34        /* Proporciona direccion de banderas */
                                     /* de dos ocupadas                   */
#define DOS_SET_PSP        0x50      /* Coloca la direccion actual de PSP */
#define DOS_GET_PSP        0x51      /* Proporciona la direccion actual - */
                                     /* del PSP (Segmento de Programa --- */
                                     /* Prefijado                         */
```

```
#define DOS_LIST_ADDRESS 0x52          /* Proporciona la direccion de lista */
                                       /* del DOS                           */

/* Puertos de hardware necesarios ***********************************/
#define KEYBOARD_DATA       0x60       /* Teclado                           */
#define KEYBOARD_STATUS 0x61           /* Estatus del puerto de teclado     */

/* Posiciones donde el BIOS almacena los valores del Shift***********/
#define KEY_FLAGS       0x417   /* Posicion banderas                        */
#define K_RIGHT             (1<<0)     /* Shift de la derecha   presionada */
#define K_LEFT              (1<<1)     /* Shift de la izquierda presionada */
#define K_CONTROL           (1<<2)     /* Control presionado               */
#define K_ALT               (1<<3)     /* Alt     presionado               */
#define K_SLOCK             (1<<4)     /* Scroll lock activo               */
#define K_NLOCK             (1<<5)     /* Num lock    activo               */
#define K_CAPS              (1<<6)     /* Caps lock   activo               */
#define K_INSER             (1<<7)     /* Insert      activo               */

/* Estructuras de los registros pasados a una rutina de interrupcion */
#define INTERRUPT_REGS int bp,int di,int si,int ds,int es,\
        int dx,int cx,int bx,int ax,int ip,int cs,int flags

static unsigned scancode = 0x58; /* Para el control de la tecla de    */
static unsigned keymask = 0;     /* funcion F12                       */
unsigned _stklen = (2 * 1024);   /* Valor para el calculo del tamano  */
                                 /* del programa a almacenar          */
unsigned _heaplen = 2;           /* Valor default para el calculo     */
/* unsigned int _stklen = (16 *1024);/* Incremento del stack          */

/* El siguiente valor es usado para ver si ya ha sido cargado el TSR */
static unsigned long int magic_number = 0x54535258L;

/* Verdadera,Si el proceso de descarga de la rutina fue satisfactorio*/
static boolean unloading = FALSE;
static unsigned char far *video_mode;   /* Apuntador a la posicion   */
                        /* de almacenamiento del modo de video */
/* Las siguientes tres interrrupciones son grabadas solo cuando la  */
 * porcion del TSR cargada esta ejecutandose, para que no se salga  */
 * el programa ******************************************************/
static void interrupt tsr_critical(INTERRUPT_REGS);
static void interrupt tsr_break(void);
#define tsr_control_c tsr_break
void interrupt (*norm_break)(void);
void interrupt (*norm_control_c)(void);
void interrupt (*norm_critical)(void);
```

```
static char far *dosbusy;/* Indica cuando estan ocupadas las banderas*/
                         /* del DOS */
static int diskflag = 0;/* Ayuda al control de disco, si no es 0 esta*/
                         /* siendo utilizada alguna unidad */
struct mcb {
    char flag;           /* 'M' -- bloque normal 'Z' -- ultimo bloque */
    unsigned int owner; /* PSP propietario del bloque                 */
    unsigned int size;           /* Size of the block in paragraphs   */
    unsigned char not_used[3];  /* Control Espacio libre,amortiguador*/
    char name[8];                /* Nombre de quien usa los recursos  */
};

static struct mcb far *mcb_start;    /* Variables auxiliares para el */
static boolean hotkey_found = FALSE; /* arranque y alto del TSR      */

static boolean running = FALSE; /* Verdad, cuando el TSR esta activo */

/* vectores de interrupcion cargados permanentemente para el control */
/* del TSR cuando se carga ***************************************/
void interrupt tsr_timer(void);                 /* Reloj             */
void interrupt tsr_idle(void);                  /* Dos IDEL          */
void interrupt tsr_keyboard(void);              /* Tecla presionada  */
void interrupt tsr_disk(INTERRUPT_REGS);/* Uso de disco              */

/* Lugares para almacenar las interrupciones anteriores **************/
void interrupt (*norm_timer)(void);
void interrupt (*norm_keyboard)(void);
void interrupt (*norm_idle)(void);
void interrupt (*norm_disk)(void);

/*void llamada(void);
void envios(void);*/

/* Vectores de interrupcion almacenados durante el cargado del TSR ***/
struct hook_vect {
    int  vect;      /* Numero del vector de interrupcion a interceptar */
    void interrupt ((*tsr_vect)(void); /* Apuntador a la nueva rutina */
                                       /* de interrupcion             */
    void interrupt ((*old_vect)(void);/* Almacenamiento de la rutina */
                                       /* anterior                    */
} hooks[] = {
    /*vector****TSR***************anterior estado ********************/
    {INT_TIMER, tsr_timer,          &norm_timer},
    {INT_IDLE,    tsr_idle,         &norm_idle},
    {INT_KEY,   tsr_keyboard,     &norm_keyboard},
```

```
#pragma warn -sus  /* Desactiva el apuntador de amonestaciones      */
/* Turbo-C maneja la rutina de interrupcion con registro en diferente*/
/* forma a la rutina normal de interrupcion  ************************/
    (INT_DISK,     tsr_disk,          &nora_disk),
#pragma warn .sus  /*Activa amonestaciones para el resto del programa*/
    (-1,           NULL,         NULL)/*Fin de la lista de indicadores*/
};




/*****************************************************************************
 * Contexto --) Estructura para guardar toda la informacion necesaria       *
 *              para la toma de acciones.                                    *
 *                                                                           *
 * Informacion almacenada --)                                                *
 *      segmentos de pila: stack_register, PSP, DTA                          *
 *      Manejadores de interrupcion:(break, control_c, error critico)        *
 *                                                                           *
 * Informacion no almacenada --)                                             *
 *      Registros: Son almacenados sobre la interrupcion de pila             *
 *      Registros de punto flotante: No salvados. Se asume que el ---        *
 *              programa no hace uso de ellos.                               *
 *      Datos de pantalla: Se graba la pantalla actal antes ejecutar-        *
 *              se las operaciones del TSR.                                  *
 *****************************************************************************/
struct context

          /* Localizacion de segmentos, pila y registros (SS, SP) */
 unsigned int stack_segment, stack_register;
 unsigned int psp;              /* Segmento Prefijado de Programa PSP */
/* DTA(Direccion de Transferencia de datos) Donde DOS lee/escribe    */
 char far *dta;
 /*Manejadores de interrupcion*************************************/
 void interrupt (*int_break)(void);          /* Break              */
 void interrupt (*int_control_c)();          /* Control-C          */
 void interrupt (*int_crit)(INTERRUPT_REGS);/* Error Critico       */
 unsigned int cursor_size;                   /* Tamano cursor      */
 unsigned int cursor_loc;                    /* Posicion cursor    */
 unsigned char video_page;                   /* No. de pagina de video*/
 boolean control_break;                      /* Bandera de rompimiento*/
};
```

```
/* Contexto para este programa (TSR)*************************************/
static struct context tsr_context = {
    0, 0,                           /* Segmentos de pila y registro */
    0,                              /* PSP                          */
    0,                              /* DTA                          */
    tsr_break,                      /* Break                        */
    tsr_control_c,                  /* Control-C                    */
    tsr_critical,                   /* Error critico                */
    0x0B0C,                         /* Tamano cursor                */
    0x0102,                         /* Posicion cursor              */
    0,                              /* Pagina de video              */
    0,                              /* Bandera de rompimiento       */
};



/* Contexto del programa que llama al TSR */
static struct context normal_context;



/* Carga-descarga del TSR ***********************************************/
static void do_unload(void)
{
 struct hook_vect *hook;        /* Actual interrupcion considerada   */
 struct mcb far *current_mcb = mcb_start;/*Segmento de memoria actual*/

 for (hook = hooks; hook->vect != -1; hook++)
  {
  setvect(hook->vect, *hook->old_vect);
  }
 while (1)
  {
  if (current_mcb->owner == _psp)  freemem(FP_SEG(current_mcb)+1);
  if (current_mcb->flag  != 'M')   break;
  current_mcb = MK_FP(FP_SEG(current_mcb) + current_mcb->size + 1, 0);
  }
 magic_number = 0;                          /* Elimina el numero */
}
```

```
/*************************************************************************
 * unload_ok: VERDADERO, si se permite descargar                         *
 * Regresa                                                               *
 *      True -- podemos descarga                                         *
 *      False -- no se descargara                                        *
 * Checa si el vector de interrupciones que se sobre escribio ha sido    *
 * utilizado por algun otro proceso, si es asi, no podemos reestable-    *
 * cerle los valores, por ello tampoco podemos bajar el TSR              *
 *************************************************************************/
static boolean unload_ok(void)
{
 struct hook_vect *cur_hook;      /* actual interrupcion considerada */
 for (cur_hook = hooks; cur_hook->vect != -1; cur_hook++)
  {
  if (getvect(cur_hook->vect) !=  cur_hook->tsr_vect)  return (FALSE);
  }
 return (TRUE);
}

/*drawbox -- pinta una caja utilizando linea doble x1,y1 a x2,y2 */
static void drawbox(int x1, int y1, int x2, int y2)
{
 int x, y; /* Posicion actual */
 gotoxy(x1+1, y1);          /* Pinta parte superior linea horizontal */
 for (x = 1; x < x2; x++) putch(0xCD);

 gotoxy(x1+1, y2);          /* Pinta parte inferior linea horizontal */
 for (x = 1; x < x2; x++) putch(0xCD);
 for (y = 1; y < y2; y++)   /* Pinta lineas laterales                */
   {
    gotoxy(x1, y);  putch(0xBA);
    gotoxy(x2, y);  putch(0xBA);
    }
 gotoxy(x1, y1); putch(0xC9); /* Pinta esquina superior izquierda   */
 gotoxy(x1, y2); putch(0xC8); /* Pinta esquina inferior izquierda   */
 gotoxy(x2, y1); putch(0xBB); /* Pinta esquina superior derecha     */
 gotoxy(x2, y2); putch(0xBC); /* Pinta esquina inferior derecha     */
}
/* Estas son las constantes que definen la caja ********************/
#define X1        20
#define Y1        10
#define X2        34
#define Y2        13
/* Nota: Se ha salvado una linea extra para que no cambie la pantalla *
 *       cuando se escribe un caracter en la ultima linea             */
```

```
#define X_WIDTH (X2-X1+1)
#define Y_WIDTH (Y2-Y1+1)
/* 2 bytes por caracter (1 atributo/1 caracter) */
static char far window_save [(X_WIDTH * Y_WIDTH * 2)];



/*************************************************************************
 * tsr_main -- cuerpo de la rutina que coloca al TSR                     *
 *                                                                       *
 * Despliega una ventana con "COMUNICANDO" y efectua la operacion de     *
 * comunicion. Si se presiona "u" al terminar trata de descargar.        *
 *************************************************************************/

static void tsr_main(void)
{
 int ch;
 int band;
 (void)gettext(X1, Y1, X2, Y2, window_save);
 window(X1, Y1, X2, Y2);

 clrscr();
 drawbox(1, 1, X_WIDTH, Y_WIDTH-1);
 gotoxy(2,2);
 cputs(" COMUNICANDO");

    band=0;
    while(band != 1)
       {
        llamada();
        envios();
        if (kbhit() != 0)
                {
                 ch=bioskey(0);
                 ch &= 0x7F;
                 band=1;
                }
        }
    if (ch=='u') unloading=unload_ok();
 (void)puttext(X1, Y1, X2, Y2, window_save);
}
```

```
/*****************************************************************************
 * Get_Context:  Obtiene el contexto del actual proceso.                     *
 * Parametros:   El contexto.                                                *
 * Nota: Esto hace no manejable la obtencion de los segmentos de pila        *
 *       y registro. Las pilas son dificiles de trabajar con el con-         *
 *       texto, y debe ser salvado desde alto nivel.                         *
 *****************************************************************************/

void get_context(struct context *context)
{
 _AH = DOS_GET_PSP;                  /* Obtiene la direccion del PSP */
 geninterrupt(INT_DOS);
 context->psp = _BX;

 context->dta = getdta();
 context->control_break = getcbrk();

 context->int_crit = getvect(INT_CRITICAL);
 context->int_control_c = getvect(INT_CONTROL_C);
 context->int_break = getvect(INT_BREAK);

 _AH = VIDEO_GET_INFO;
 geninterrupt(INT_VIDEO);
 context->cursor_size = _CX;
 context->cursor_loc = _DX;
 context->video_page = _BH;
}


/*****************************************************************************
 * set_context: Pone a salvo el contexto para los valores dados.             *
 * Parametros:  El contexto a salvar.                                        *
 *                                                                           *
 * Nota:No se salva el apuntador de pila por la dificultad de manejo.        *
 *****************************************************************************/
void set_context(struct context *context)
{
 _AH = DOS_SET_PSP;  /* Pone la direccion del PSP */
 _BX = context->psp;
 geninterrupt(INT_DOS);
 setdta(context->dta);
 setcbrk(context->control_break);
 setvect(INT_CRITICAL, context->int_crit);
 setvect(INT_CONTROL_C, context->int_control_c);
 setvect(INT_BREAK, context->int_break);
```

```
    _AH = VIDEO_SET_CURSOR_SIZE;
    _CX = context->cursor_size;
    geninterrupt(INT_VIDEO);

    _AH = VIDEO_SET_CURSOR_POSITION;
    _DX = context->cursor_loc;
    _BH = context->video_page;
    geninterrupt(INT_VIDEO);
   }

   /* do_it -- salva el contexto y ejecuta el TSR ***********************/
   static void do_it(void)
   {
    static int mode;
    disable();

    normal_context.stack_segment = _SS;
    normal_context.stack_register = _SP;

    get_context(&normal_context);
    set_context(&tsr_context);

    _SS = tsr_context.stack_segment;
    _SP = tsr_context.stack_register;

    running = TRUE;
    hotkey_found = FALSE;

    enable();

    mode = *video_mode; /* Solo en modo texto */

    if (((mode >= 0) && (mode <= 3)) ||(mode == 7)) tsr_main();

    running = FALSE;

    disable();
    if (unloading) do_unload();
    set_context(&normal_context);

    _SP = normal_context.stack_register;
    _SS = normal_context.stack_segment;
    enable();
   }
   #define PSP_MAGIC      0x20CD   /* Con esto comienza cada PSP        */
```

```
/*************************************************************************
 * is_loaded -- Regresa VERDADERO si ya esta cargado.                    *
 *                                                                       *
 * Esta rutina recorre la memoria buscando PSP's, cuando encuentra -- *
 * otro programa hace el magic_numeric igual al lugar en que lo hallo.*
 *                                                                       *
 * Precaucion: Solo trabaja con modelos de memoria TINY, SMALL           *
 *************************************************************************/

static boolean is_loaded(void)
{
 struct mcb far *current_mcb = mcb_start;/*Actual valor en la cadena MCB*/


 int far *block_ptr;        /* Apuntador al actual bloque de memoria */
 unsigned int magic_seg, magic_off;/* Segmento y baja de  no. magico */
 long unsigned int far *magic_ptr; /* Apuntador a numero magico      */

 while (1)
  {
   if ((current_mcb->owner != NULL) &&
       (current_mcb->owner != _psp))
       block_ptr = MK_FP(current_mcb->owner, 0);

       if (*block_ptr == PSP_MAGIC) {
            magic_seg = FP_SEG(block_ptr) + _DS - _psp;
            magic_off = (int)&magic_number;
            magic_ptr = MK_FP(magic_seg, magic_off);
            if (*magic_ptr == magic_number)
                return (TRUE);
       }
   }
   if (current_mcb->flag != 'M')
       break;

   current_mcb =
       MK_FP(FP_SEG(current_mcb) + current_mcb->size + 1, 0);
  }
  return (FALSE);
}
/*************************************************************************
 * tsr_break: Manejador de rompimientoo (BREAK)                          *
 *              Dado que se tiene un TSR, este no debe bloquearce, y     *
 *              por lo cual se ignoran todos los BREAKS                  *
 *************************************************************************/
```

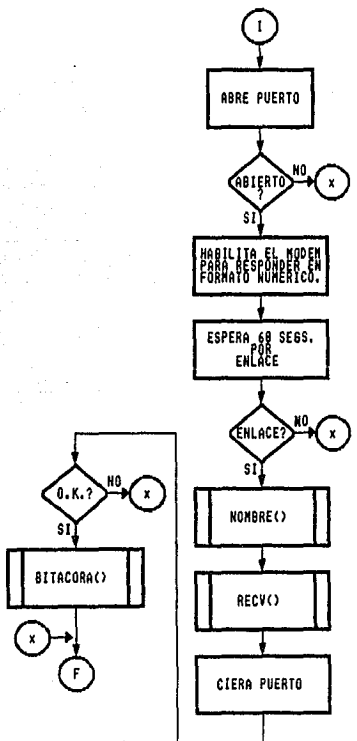
```
static void interrupt tsr_break(void)
{
 return;
}


/*************************************************************************
 * tsr_critical: Manejador de error critico.                             *
 * Regresa: AX = 0    Le dice a DOS que ignore el error.                 *
 *************************************************************************/

#pragma warn -par    /* Nota: No se utilizan todos los parametros    */
static void interrupt tsr_critical(INTERRUPT_REGS)
{
 ax = 0;
}
#pragma warn .par    /* Restablece las amonestaciones ****************/


/*************************************************************************
 * tsr_disk -- Es llamada para cada interrupcion de disco.               *
 *                                                                       *
 * Esta rutina guarda la lista de las interrupcion de disco para         *
 * que no se carge el TSR cuando exista alguna operacion de disco.       *
 *************************************************************************/

#pragma warn -par     /* Nota: No se utilizan todos los parametros   */
static void interrupt tsr_disk(INTERRUPT_REGS)
{
 diskflag++;
 (*norm_disk)();
 ax = _AX;      /* Pasa de regreso todos los registros ****************/
 bx = _BX;
 cx = _CX;
 dx = _DX;
 si = _SI;
 di = _DI;
 es = _ES;
 flags = _FLAGS;
 diskflag--;
}
#pragma warn .par      /* Reestablece las amonestaciones  ************/
```

```
/*************************************************************************
 * tsr_keyboard -- Interrupcion de teclado.                              *
 *                                                                       *
 * Checa si alguna tecla de especial fue presionada y pone las bande-    *
 * ras en base a ello.                                                   *
 *************************************************************************/

static void interrupt tsr_keyboard(void)
{
 static int keyboard_status;        /* Estatus del puerto de teclado */
 static unsigned char far *key_flags = MK_FP(0, KEY_FLAGS);

 if (running == FALSE)
    {
     if ((inportb(KEYBOARD_DATA) == scancode) &&
         ((*key_flags & keymask) == keymask))
         {
          hotkey_found = TRUE;
          /* Prueba el bit mas alto del registro de estados de teclado*/
          keyboard_status = inportb(KEYBOARD_STATUS);
          outportb(KEYBOARD_STATUS, keyboard_status | 0x80);
          outportb(KEYBOARD_STATUS, keyboard_status);
          outportb(0x20, 0x20);       /* Limpia la interrupcion ********/
          return;
         }
     }
 (*norm_keyboard)(); /* La tecla es FIZ */
}
/*************************************************************************
 * tsr_timer -- Rutina de interrupcion de reloj                          *
 * Cuando es tiempo de ejecutar el tsr_main(), esta rutina lo indica */
static void interrupt tsr_timer(void)
{
 (*norm_timer)();
 if (running == FALSE)
    {
    if  ( hotkey_found /* && (*dosbusy == 0)*/)  /*cambios VIP */
       {
         if (diskflag == 0)
            {
             do_it();
            }
         }
     }
}
```

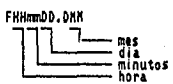
```
/*************************************************************************
 * tsr_idle -- Manejador de Interrupcion IDLE                            *
 *                                                                       *
 * Esta es llamada cuando DOS no esta muy ocupado para que permita a     *
 * los TSR ejecutarse.                                                   *
 *************************************************************************/

static void interrupt tsr_idle(void)
{
 (*norm_idle)();
 if (running == FALSE)
    {
     if (hotkey_found)
         {
          if (diskflag == 0)
             {
              do_it();
             }
         }
    }
}

int main(void)
{
 struct hook_vect *hook;                    /* Vector actual tomado */
 unsigned int far *table_ptr;       /* Apuntador a las tablas de DOS */

 _AH = DOS_GET_DOS_BUSY;
 geninterrupt(INT_DOS);
 dosbusy = MK_FP(_ES, _BX);

 _AH = DOS_LIST_ADDRESS;
 geninterrupt(INT_DOS);
 table_ptr = MK_FP(_ES, _BX);
 /* Entrada -1 = Inicio de cadena de memoria */
 mcb_start = MK_FP(*(table_ptr - 1), 0);
#pragma warn -rch /* El siguiente codigo no debe ser modificado si se */
/* compila correctamente, sin embargo, si la alineacion de estructura */
/* de palabras se usa, dara baja a los procesos y avisara al usuario. */
 if (sizeof(struct mcb) != 16) {
     (void)cputs("Error de Compilacion: No compile con alineacion de palabras\n\r");
     exit (1);
 }
#pragma warn .rch          /* Reestablece amonestaciones */
  video_mode = MK_FP(MODE_SEG, MODE_OFFSET);
```

```
#ifdef DEBUG
    tsr_main();
#else DEBUG
    if (is_loaded() == FALSE) {
        tsr_context.stack_segment = _SS;
        tsr_context.stack_register = _SP;
        tsr_context.psp = _psp;
        tsr_context.dta = getdta();

        for (hook = hooks; hook->vect != -1; hook++) {
            hook->old_vect = getvect(hook->vect);
            setvect(hook->vect, hook->tsr_vect);
        }

        {
         static unsigned keep_size; /* Tamano del oregrama en parrafos */
         keep_size = _SS - _psp + (_SP / 16) + 50;
         keep(0, keep_size);
        }
    }
    (void)cputs("Programa ya cargado\n\r");
#endif DEBUG
    return (0);
}
```

DIAGRAMA DE FLUJO (PROGRAMA: RECIBE.C)



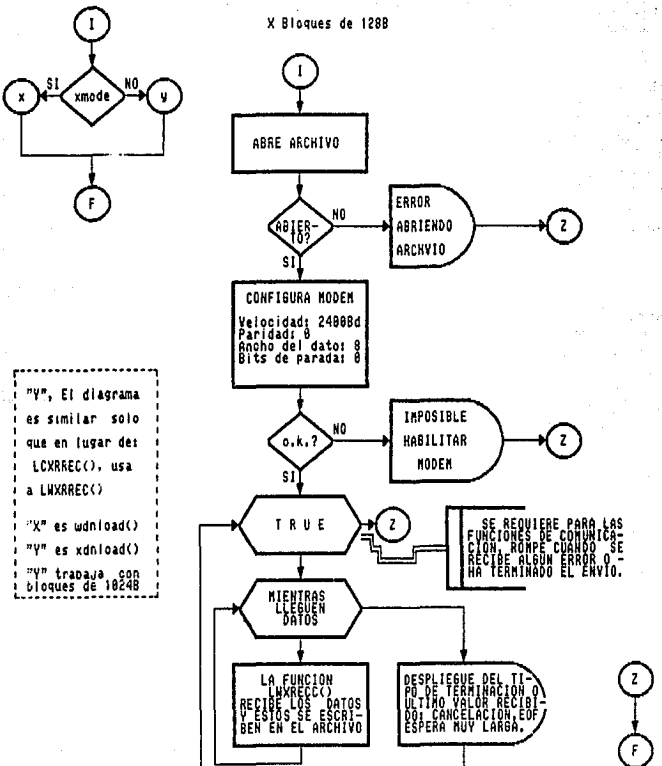

NOMBRE(): Esta funcion -- determina el nombre del - archivo que se va a recibir, las caracteristicas del nombre son:




BITACORA(): Escribe en - ARCH2 (BITACORA.DAT), el nombre del archivo recibido y la fecha de recepcion.

RECV(): Rutina que llama a funciones de comunicacion (iwxrecc, yoxrecc), para recibir el archvio.

DIAGRAMA DE FLUJO (PROGRAMA: RECIBE.C [FUNCION RECV()])



```
/* Modulo recibe.c se encarga de aceptar comunicacion, checa llamada */
/* y una vez que se establece la conexion recibe un archivo dejando- */
/* lo bajo un nombre en base a la hora y fecha de recepcion. Luego,  */
/* actualiza la bitacora en base a lo realizado.                     */
/* Las funciones utilizadas en la comunicacion son de la libreria de */
/* de comunicaciones de LiteComm.                                    */


#include <time.h>
#include <stdlib.h>
#include <dos.h>
#include <stdio.h>
#include <dir.h>
#include <fcntl.h>
#include <sys\stat.h>
#include <io.h>
#include <mem.h>

#include "apoyo.h"
#include "enlace.h"

#include <litecomm.h>
#include <litehcm.h>
#include <litexm.h>
#include "lcbbs.c"

#ifdef __TURBOC__
extern unsigned _stklen = 8192;
#endif

int nocbrk();
void recv(void);
void xdnload(void);
void wdnload(void);

void nombre(void);
int bitacora(void);

int band=0,espera;
char *directorio;
char *dir_back;
char name[13]="F";
int res=2726;
char *captura,cap;
```

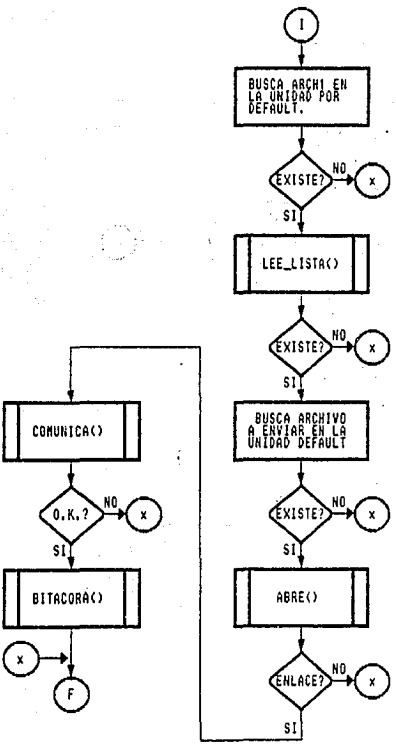
```
void main()
{
 if (coma_opn(puerto,2400,NPARITY,BITS,STOP1,2048,2048,TRUE) == -1)
    {
     urgentmsg("ERROR", "Imposible abrir puerto");
     abort();
    }
 lch_nueres(puerto);
 sleep(60);
 if ( (lc_estat(puerto) & DCD) )
    {
     nombre();
     recvi();      /*se establecio conexion */
     coma_close(puerto,TRUE);
     if (band==1) oitacora();  /* se recibio adecuadamente archivo */
    }
}

voio nombre(voiд)
{
time_t *fechador;
struct tm *fechas;
struct tm *fecha;
char *punto=".D";
int    hh=0,       min=0,       mes=0,    dia=0;
cnar *c_hh=" ", *c_min=" ", *c_mes=" ", *c_dia=" ";

time(fechador);
fechas = localtime(fechador);

hh  = fechas->tm_hour;
min = fechas->tm_min;
dia = fechas->tm_mday;
mes = fechas->tm_mon + 1;

  itoa(hh , c_hh,10);
     strcat(name, c_hh   );
  itoa(min, c_min,10);
     strcat(name, c_min  );
  itoa(dia, c_dia,10);
     strcat(name, c_dia  );
  itoa(mes, c_mes,10);
     strcat(name, punto  );
     strcat(name, c_mes  );
}
```

```
int bitacora(void)
{
time_t *fechador;
char *fecha=NULL;
FILE *f2=0;
char status[9] = "RECIBIDO";
char phone[11]  = "**********";
struct reg_bitacora bitacora_reg;
time(fechador);
fecha = ctime(fechador);
if ( (f2=fopen(arch2,"a")) != NULL)
          {
            fprintf(f2,"%s %s %s %s \n",name,phone,status,fecha);
            fclose(f2);
            printf(" \n Actualiza  Bitacora %s",fecha);
          }
}

void recv(void)
{
 if (xmode) wdnload();   else xdnload();
}

/* realiza envios empaquetando bloques de 128 bytes ********************/
void wdnload(void)
{
int  fd;
unsigned char buf[128];
unsigned char hdshk;
int           mode;
int           result;
if ((fd = open(name,
        (O_WRONLY|O_CREAT|O_TRUNC|O_BINARY),(S_IREAD|S_IWRITE))) == -1)
   {
    urgentmsg("ERROR", "Imposible abrir archivo");
    return;
   }
if (coma_setup(puerto,pbaud,NPARITY,BIT8,pstop) == ERR)
   {
    urgentmsg("ERROR", "Imposible habilitar linea con el MODEM");
    return;
   }
```

```
while (TRUE)
   {
    while ((result = lwxrrec(puerto, buf)) == SUCCESS)
      {
       write(fd, buf, sizeof(buf));              /* Baja el bloque */
      }
    switch(result)
      {
       case DUPSEQ:
                    break;
       case   CAN:
                    urgentmsg("ERROR", "Cancelacion recibida");
                    break;
       case   EOT:
                    urgentmsg("SUCCESS","Terminacion Normal");
                    break;
       case  TOUT:
                    urgentmsg("ERROR", "SOH Se acabo el tiempo");
                    break;
       case RETRIES:
                    urgentmsg("ERROR", strbuf);
                    break;
       default:
              urgentmsg("ERROR", "Error Fatal en Transmision");
              break;
      }
    if ((result != SUCCESS) && (result != DUPSEQ)) break;
  }                              /* while TRUE ********************/
comm_setup(puerto,pbaud,pparity,pbits,pstop);
close(fd);
}

/* realiza envios empaquetando bloques de 1024 bytes ******************/
void xdnload(void)
{
int  fd;
unsigned char buf[1024];
unsigned char hdshk;
int           bsize;
int           hmode;
int           result;
```

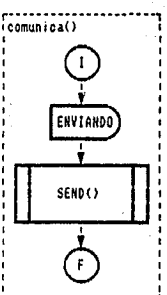
```
if ((fd = open(name,
        (O_WRONLY:O_CREAT:O_TRUNC:O_BINARY),(S_IREAD:S_IWRITE))) == -1)
    {
     urgentmsg("???????", "Imposible abrir archivo");
     return;
    }
if (comm_setup(puerto,pbaud,NPARITY,BIT8,pstop) == ERR)
    {
     urgentmsg("ERROR", "Imposible inicializar MODEM");
     return;
    }
hdshk = CRC;                          /* Usa metodo CRC          */
hmode = RELAXED;                        /* Tiempo normal de conexion */
while (TRUE)
  {
   while ((result = ixxrrec(puerto, buf, &bsize, hmode, &hdshk)) == SUCCESS)
     {
      write(fd, buf, bsize);                  /* Baja el bloque     */
     }
   switch(result)
     {
     case DUPSEQ: break;
     case   CAN:
                  urgentmsg("???????", "Cancelacion desde el solicitante");
                  break;
     case   EOT:
                  urgentmsg("RECIBIDO", "Terminacion normal");
                  band=1;
                  break;
     case  TOUT:
                  urgentmsg("ERROR", "SOH Se acabo el tiempo");
                  break;
     case RETRIES:
                  sprintf(strbuf, "Demasiados intentos: %d",(rec-1));
                  urgentmsg("ERROR", strbuf);
                  break;
     default:
             urgentmsg("ERROR", "Error Fatal en Transmision");
             break;
     }
   if ((result != SUCCESS) && (result != DUPSEQ)) break;
  }                                       /* while TRUE **********/
comm_setup(puerto,pbaud,pparity,pbits,pstop);
close(fd);
}
```

DIAGRAMA DE FLUJO (PROGRAMA: ENVIA.C)




LEE_LISTA(): Abre ARCHI y lee el primer registro, - Si no tiene un valor significativo (telefono y -- archivo), regresa un valor de inexsistente.

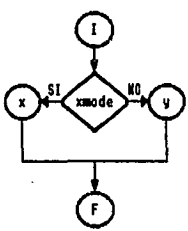
ABRE(): Marca el numero - telefonico, espera por -- respuesta, portadora y -- enlace.

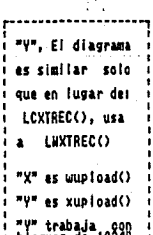


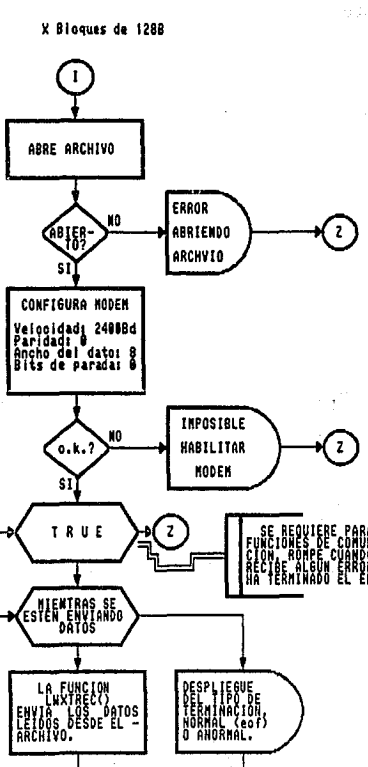

COMUNICA(): Llama a SEND() y esta a su vez a las --- funciones de comunicacion (ioxtrrec, iwtrec), para enviar el archivo a la otra PC. Al terminar, regresa un valor enbase al tipo de envio, si fue --- llevado a cabo bien o no.

BITACORA(): Si el archivo fue enviado adecuadamente se registrara en ARCH2 la hora, y archivo enviado. Tambien actualiza ARCHI, dado que ya se realizo un envio de la lista.

DIAGRAMA DE FLUJO (PROGRAMA: ENVIA.C [FUNCION SEND()])



```
/* Modulo envia.c se encarga de entablar la comunicacion y enviar el */
/* una vez que se establece la conexion, cuando termina actualiza -- */
/* la bitacora con la informacion correspondiente.                   */
/* Las funciones utilizadas en la comunicacion son de la libreria de */
/* de comunicaciones de LiteComm.                                    */


#include <stdio.h>
#include <dos.h>
#include <dir.h>
#include <time.h>
#include <string.h>
#include <fcntl.h>

#include <litecomm.h>
#include <litehce.h>
#include <litexm.h>
#include <lcbbs.h>

#include "enlace.h"
#include "apoyo.h"
#include "lcbbs.c"

struct registro regi[15];
char   strbuf[80];


int  lee_lista(void);  /* LEE EL SIGUIENTE ARCHIVO Y DESTINO */
int  abre();           /* ENTABLA COMUNICACION               */
int  comunica(void);   /* ENVIA EL ARCHIVO                   */
void bitacora(void);   /* ACTUALIZA LA BITACORA              */
void marca(void);
void pega_phone(void);
void send(void);
void xupload(void);
void wupload(void);


int f1;
int f2;
int i;
int band=0;
char *directorio;
char dstr[] = "ATD ";
```

```
main()
{

if ( ( directorio = searchpath(archi) ) != NULL)
   {
    if ( lee_lista() == 1)
       {
        directorio = searchpath(reg1[0].archivo);
        if ( (f1 = fopen(directorio,"r")) != NULL)
           {
            fclose(f1);
            if ( abre() == 1)
               {
                  if ( comunica() == 1) bitacora();
               }
           }
       }
   }
   reset_modem(puerto);
   comm_close(puerto,TRUE);
}



int lee_lista(void)
{
i=0;
if ( (f1 = fopen(directorio,"r")) != NULL )
        {
        while( eof(f1)!=0 && i<15)
                {
                fscanf(f1,"%s%s",reg1[i].archivo,reg1[i].telefono);
                i++;
                }
        fclose(f1);
        }
if (i != 0) return(1); else return(0);
}
```

```
int abre(void)
{
pega_pone();
if (coma_opn(puerto,2400,NPARITY,8118,STOP1,2048,2048,TRUE) == -1)
        {
         urgentesg("ERROR", "Imposible abrir puerto");
         abort();
        }
    else
        {
         if (reset_modem(puerto) != -1)
                {
                 marca();
                 return();
                }
                else urgentesg("ERROR","Imposible controlar MODEM");
        }
}

int comunica(void)
{
printf(" \n Enviando %s  a  %s",reg1[0].archivo,reg1[0].telefono);
send();
return(band);
}

void bitacora(void)
{
time_t *fechador;
int z=0;
char *fecha;
char *status="ENVIADO";
struct reg_bitacora bitacora_reg;

time(fechador);
fecha = ctime(fechador);
strcpy(bitacora_reg.fechas ,"                    ");
strcpy(bitacora_reg.archivo ,"          ");
strcpy(bitacora_reg.telefono,"          ");
strcpy(bitacora_reg.estado  ,"          ");
strcpy(bitacora_reg.fechas  ,fecha          );
strcpy(bitacora_reg.archivo ,reg1[0].archivo );
strcpy(bitacora_reg.telefono,reg1[0].telefono);
strcpy(bitacora_reg.estado  ,status         );
```

```
if ((f2 = fopen(arch2,"a")) != NULL)
        {
        fprintf(f2,"%s %s %s %s \n",bitacora_reg.archivo,
                                    bitacora_reg.telefono,
                                    bitacora_reg.estado,
                                    bitacora_reg.fechas);
        fclose(f2);
        }
directorio = searchpath(arch1);
fi=fopen(directorio,"w");
        for (z=0;z<(i;z++)
            {
             if (z==14)
               fprintf(f1,"%s %s \n",arch_vacio,tel_vacio);
               else
               fprintf(f1,"%s %s \n",regi[z+1].archivo,regi[z+1].telefono);
}


printf(" \n Actualiza  Bitacora %s",fecha);
}
void marca(void)
{
  char *t;
                for(t = dstr; *t; t++)
                        lc_put(puerto, *t);
                delay(500);
}
void pega_phone(void)
{
 strcat(dstr,regi[0].telefono);
 strcat(dstr,"\r");
}

void send(void)
{
 if (xnode)
        wupload();
    else
        xupload();
}
```

```
void xupload(void)
{
 int  fd;
 unsigned char buf[1024];
 unsigned char kopos;
 int toread;
 int tosend;
 unsigned char hoshk;
 int hmode;
 int result;

if ((fd = open(directorio, (O_RDONLY:O_BINARY))) == -1)
    {
     urgentmsg("Error", "Imposible abrir archivo");
     return;
    }

/* ### archivo abierto, cambio a modo de 8 bits para el protocolo xmodem */
if (comm_setup(puerto,pbaud,NPARITY,8ITS,pstop) == ERR)
    {
     urgentmsg("ERROR","Imposible habilitar linea con el MODEM");
     return;
    }
/* Inicio de la transmision */
if (yxmode)
    {
     ymodem = TRUE;
     toread = 1024;
    }
    else
    {
     ymodem = FALSE;
     toread = 128;
    }
while (TRUE)
    {
     memset(buf, 0x1a, sizeof(buf));     /* limpia buffer */
     if ((tosend = read(fd, buf, sizeof(buf))) < 1)  /* EOF o Error */
        {
          lcxtout(puerto);            /* envia fin de archivo */
          urgentmsg("Enviado","Fin de la transmisio");
          band=1;
          break;
        }
```

```
    bpos = buf;
    while (TRUE)
        {
        if (tosend (= 0)
                break;              /* Bloque enviado completamente */
        if (ymodem)
                if (tosend != toread)       /* bloque corto */
                        {
                        toread = 128;       /* largo del bloque */
                        ymodem = FALSE;
                        }
        result = lcxtrec(puerto, bpos);
        switch(result)              /* accion a tomar */
                {
                case SUCCESS:
                                tosend -= toread;
                                bpos += toread;
                                break;
                case    CAN:
                                urgentmsg("Error","Cancelacion recibida");
                                break;
                case RETRIES:
                                urgentmsg("ERROR", strbuf);
                                break;
                default:
                                urgentmsg("ERROR", "Error Fatal en Transmision");
                                break;
                }
        if (result != SUCCESS) break;
        }                   /* while TRUE anidado */
        if (result != SUCCESS) break;
    }                       /* while TRUE externo */
comm_setup(puerto,pbaud,pparity,pbits,pstop);
close(fd);
}

void wupload(void)
{
 int  fd;
 unsigned char buf[128];
 unsigned char hdshk;
 int hmode;
 int result;
 int nrec;
```

```
if ((fd = opendirectorio,(O_RDONLY|O_BINARY))) == -1)
    {
     urgentmsg("ERROR", "Imposible abrir archivo");
     return;
    }
/* El archivo fue abierto, cambio a modo de 8 bits para el protocolo xmodem */
if (comm_setup(puerto,pbaud,NPARITY,BITS,pstop) == ERR)
    {
     urgentmsg("ERROR", "Imposible habilitar linea con el MODEM");
     return;
    }
lc_xoff(puerto,TRUE);                    /* enciende auto xon-xoff */
/* Inicia transmision del archivo */
while (TRUE)
    {
     memset(buf, 0x1a, 128);     /* limpia buffer */
     if (read(fd, buf, sizeof(buf)) < 1)  /* EOF o Error */
           nrec = -1;                /* Determina el fin de archivo */
       else
           nrec = 0;
     result = lwxtrec(puerto, buf, &nrec);
     switch(result)          /* accion a tomar */
       {
         case SUCCESS:
                     if (nrec == -1)  /* EOF Confirmado */
                     urgentmsg("Enviado","Fin de la transmision");
                     band=1;
                     break;
         case     CAN:
                     urgentmsg("ERROR", "Cancelacion recibida");
                     break;
         case RETRIES:
                     urgentmsg("ERROR", strbuf);
                     break;
         case  RESEND:    /* Debe inicializarce la posicion del archvio */
                     lseek(fd, (long)(nrec * sizeof(buf)), SEEK_SET);
                     break;
             default:
                     urgentmsg("ERROR", "Error Fatal en Transmision");
              break;
       }
```

```
    if (result == RESEND) continue;
    if (result == SUCCESS)
          if (nrec == -1)              /* Se encontro EOF */
            break;
          else
            continue;
    break;                          /* algun otro error */
   }                                /* while TRUE externo*/
   lc_xoff(puerto, FALSE);           /* apaga auto xon/xoff */
  comm_setup(puerto,pbaud,pparity,pbits,pstop);
  close(fd);
}
```

```
/*************************************************************************
 MODULO "enlace.h"
 Tipos, constantes y funciones generales del sistema.
*************************************************************************/

#include <process.h>
#include <litecomm.h>

#define arch1       "archivos.dat"
#define arch2       "bitacora.dat"
#define DIR1        "cd c:\\files"
#define arch_vacio  "AAAAAAAAAAA"
#define tel_vacio   "0000000000"
#define puerto      2

struct registro
                {
                char archivo[13];
                char telefono[11];
                };
struct reg_bitacora
                {
                char archivo[13];
                char telefono[11];
                char estado[9];
                char fechas[26];
                };

unsigned pbaud   = 2400;
unsigned pparity = NPARITY;
unsigned pbits   = BIT8;
unsigned pstop   = STOP1;
int      xmode   = 0;
int      yxmode  = 0;
char     strbuf[80];

void envios(void)
{
system("envia.exe");
}

void llamada(void)
{
system("recibe.exe");
}
```

```
/*************************************************************************
 MODULO "apoyo.h"
 Funciones y contantes para desplegado de mensajes y captura de datos
**************************************************************************/

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

#define cls system("cls");

/* int xy_txt(int x,int y,char *t,int i,int c, int b);*/
/* funcion para desplegado de textos
     x posicion en X
     y posicion en Y
     t cadena de texto
     i intensidad 0,1,2; baja, normal, alta
     c color
     b centelleo 0,1; normal, centellando */

int xy_txt(x,y,t,i,c,b)
int x,y,i,c,b;
char t[72];
{
 int d;
 switch(i)
      {
       case 0:lowvideo();break;
       case 1:normvideo();break;
       case 2:highvideo();break;
      }
  if (c != 8) c=0;
  (b == 0)?(d=0):(d=128);
  if ( x>80  || x<=0  ||  y>24  ||  y<=0 ) { y=1;x=1; }
  textcolor(c);
  textbackground(c+d);

  gotoxy(x,y);
  printf("%s",t);
  normvideo();
  textcolor(WHITE);
  textbackground(BLACK);
 }
```

```
char lee_cadena(x,y,largo,txt_color,txt_fondo,car)
int x,y,largo,txt_color,txt_fondo;
char car[100];
{
char ca;
char c1;
int c,cc,band;
 if (x)0 && x<72 && largo)0 && largo<79 && y)0 && y<25 && txt_color)=0 &&
     txt_color(=15 && txt_fondo)=0 && txt_fondo(=15)
     {
     textcolor(txt_color);
     textbackground(txt_fondo);
     c=0;
     band=0;
     gotoxy(x,y);
     printf("%s",car);
     gotoxy(x,y);
     while (band==0)
          {
         gotoxy(x+c,y);
          ca=getche();
          switch(ca)
               {
                 case 27 :{                  /*Escape*/
                           gotoxy(x+c,y);putchar(car[c]);
                           break;
                           }
             /*espacio*/
                 case 32:{
                           car[c]=' ';
                           if (c(largo-1) c++;
                           break;
                           }
              /*return*/
                 case 13:{
                           if (car != NULL) band=1;
                           break;
                           }
              /*borra a la izq*/
                 case 8: {
                           car[c]=' ';
                           gotoxy(x+c,y);printf("%c",car[c]);
                           if (c)0) c-- ;
                           break;
                           }
```

```
                /*Extendido*/
                  case '^': {
                            switch(c1=getch());
                                {
                                  case 'P':{              /*flecha abajo*/
                                           gotoxy(x+c,y);putchar(car[c]);
                                           break;
                                           }
                                  case 'K':{              /*flecha izq. */
                                           gotoxy(x+c,y);putchar(car[c]);
                                           if(c>0) c--;
                                           break;
                                           }
                                  case 'M':{              /*flecha der. */
                                           gotoxy(x+c,y);putchar(car[c]);
                                           if (c<largo-1) c++;
                                           break;
                                           }
                                  case 'H':{              /*flecha arriba*/
                                      gotoxy(x+c,y);putchar(car[c]);
                                           break;
                                           }
                                }
                            break;
                            }

                  default:{
                          if (ca!=11 && ca!=16 && ca!=127 && ca)=28 && ca(=122)
                            {
                            car[c]=ca;
                            if (c<largo-1) c++;
                            }
                          }

                  }

          }
      }
   return(car);
}
```

```
char lee_entero(x,y,largo,txt_color,txt_fondo,car)
int x,y,largo,txt_color,txt_fondo;
char car[100];
{
char ca;
char c1;
int c,cc,band;
 if (x>0 && x<72 && largo>0 && largo<79 && y>0 && y<25 && txt_color>=0 &&
     txt_color<=15 && txt_fondo>=0 && txt_fondo<=15)
     {
     textcolor(txt_color);
     textbackground(txt_fondo);
     c=0;
     band=0;
     gotoxy(x,y);
     printf("%s",car);
     gotoxy(x,y);
     while (band==0)
         {
         gotoxy(x+c,y);
          ca=getche();
          switch(ca)
               {
                 case 27 :{                  /*Escape*/
                           gotoxy(x+c,y);putchar(car[c]);
                           break;
                           }
             /*espacio*/
                 case 32:{
                          car[c]=' ';
                          if (c<largo-1) c++;
                          break;
                          }
              /*return*/
                 case 13:{
                          if (car != NULL) band=1;
                          break;
                          }
              /*borra a la izq*/
                 case 8: {
                          car[c]=' ';
                          gotoxy(x+c,y);printf("%c",car[c]);
                          if (c>0) c-- ;
                          break;
                          }
```

```
            /*Extendido*/
              case '^': {
                        switch(ci=getch()){
                            {
                              case 'P':{                /*flecha abajo*/
                                       gotoxy(x+c,y);putchar(car[c]);
                                       break;
                                       }
                              case 'K':{                /*flecha izq. */
                                        gotoxy(x+c,y);putchar(car[c]);
                                        if(c)0) c--;
                                        break;
                                       }
                              case 'M':{                /*flecha der. */
                                       gotoxy(x+c,y);putchar(car[c]);
                                       if (c(largo-1) c++;
                                       break;
                                       }
                              case 'H':{                /*flecha arriba*/
                                   gotoxy(x+c,y);putchar(car[c]);
                                       break;
                                       }
                            }
                        break;
                        }
                default:{
                        if (ca!=11 && ca!=16 && ca!=127 && ca)=20 && ca(=122)
                          {
                          if (ca(=57 && ca)=48)
                            {
                          car[c]=ca;
                            if (c(largo-1) c++;
                            }
                            else
                            {
                          gotoxy(x+c,y);
                            putchar(car[c]);
                            }
                          }
                        }
                }
        }
    }
 return(car);
}
```

```
void urgentmsg(a,b)
char a[25],b[25];
{
 gotoxy(17,20);
 printf("%s %s",a,b);
}
```

```
/**************************************************************************
   MODULO LCBBS.H
   Funciones de comunicacion para el sistema
**************************************************************************/

#include <stdlib.h>
#include <dos.h>
#include "litecom.h"
#include "litecom.fns"
#include "litehcm.h"
#include "litehcm.fns"
#include "lcbbs.h"

#ifdef M_I86
void sleep();
#endif

/*** Cadena a ser enviadas en la inicializacion del MODEM ***********/
static char MODEMSET0[] = "ATZ\r";
static char MODEMSET1[] = "ATT E0\r";
static char MODEMSET2[] = "ATC1 V0 X1 S0=1 M1\r";

/* Corresponden a:
**       Reset
**       Marcaje con tono y sin eco en la consola
**       Baja portadora, y modo de deteccion de portadora
**       Uso de codigo numerico
**       Modo compatible con Hayes extendido
**       Modo de respuesta al primer Ring
**       Habilita bocina del MODEM
*/

int check_for_call(port)  /* Checa si el telefono esta sonando */
unsigned port;
{
 long  cdtimer;
 int   connval;

/* Para la implementacion de esta funcion se supone la compatibilidad
   del MODEM's con el estandar HAYES, modo de respuesta automatico, -
   uso del codigo numerico y extendido.                              */

 if ( !(lc_mstat(port) & RICHG)) return(0);     /* Falso, sino suena */
/* Si suena, se dan 30 segundos para deteccion de portadora **********/
 cdtimer = new_event(30);
```

```
  while (TRUE)  /* Espera 30 segundos por llamada */
    {
      if (check_event(cdtimer))  /* checa tiempo */
          {
           disconnect(port);      /* Posible error */
           reset_modem(port);
           return(ERR);           /* No pasa nada */
          }
      if (online(port))  break;  /* Checa portadora */
    }

    if ((connval = get_modem_reply(port)) == ERR)
       {
        disconnect(port);
        reset_modem(port);
        return(ERR);              /* No hubo conexion */
       }
    return (connval);
}                           /* check_for_call */


int get_modem_reply(port)
unsigned port;
{
 unsigned char resp[3];
 int i=0;
 int ch;

 resp[0] = 0x00;
 resp[1] = 0x00;
 resp[2] = 0x00;

 while (TRUE)
   {
    if ((ch = wait(port, 1)) == ERR)    /* Tiempo fuera, sin acceso */
       return (ERR);

    if (ch == '\r')  break;             /* Regreso */
    resp[i++] = (unsigned char) ch;
    if (i > 1)  break;
   }
   return(atoi(resp));                  /* Regresa resultado */
}                           /* get_modem_reply */
```

```
void disconnect(port)
unsigned port;
{
 lc_togmdm(port, DTR);          /* Baja DTR tocacionando rompimiento */
 sleep(1);
 lc_setmdm(port, (DTR|RTS));    /* Levanta nuevamente DTR  */
}                               /* disconnect */

int reset_modem(port)
unsigned port;
{
 int mres;
 char *t;

 while(TRUE)
   {
    for(t = MODEMSET0; *t; t++) lc_put(port, *t);
#ifdef M_I86
    sleep(1);
#else
    delay(500);
#endif
    for(t = MODEMSET1; *t; t++) lc_put(port, *t);
#ifdef M_I86
    sleep(1);
#else
    delay(500);
#endif
    purge(port);                /* Limpia buffer de salida */
    for(t = MODEMSET2; *t; t++)
    lc_put(port, *t);
    sleep(1);
    return (get_modem_reply(port));  /* Espera por respuesta */
   }
}                               /* reset_modem */


long new_event(esecs)
unsigned esecs;
{
 long tbase;

 time(&tbase);
 return (tbase + esecs);
}                       /* new_event */
```

```
int check_event(ev)          /* Checa si ya termino el evento */
long ev;
{
 long l;

 time(&l);                   /* Obtiene el tiempo actual */
 return((ev > l) ? 0 : 1);   /* Regresa 0, sino ha terminado */
}                            /* check_event */


#ifdef M_I86
void sleep(secs)
unsigned secs;
{
 long sleeptimer;

 sleeptimer = new_event(secs);
 while( ! check_event(sleeptimer));          /* Espera 30 segundos */
}
#endif
```

# CAPITULO 4

## *COMUNICACION AUTOMATICA ENTRE COMPUTADORAS PERSONALES*

PUNTOS IMPORTANTES DEL DISEÑO

4.1 Introduccón.

4.2 Compilación y liga de programas.

4.3 Instrumentación.

4.4 Funcionamiento.

4.5 Mantenimiento.

# 4 PUNTOS IMPORTANTES DEL DISENO

## 4.1 INTRODUCCION

En esta sección se describen los procedimientos de compilación, liga, funcionamiento y mantenimiento del sistema, para ello se hace referencia a los programas descritos en el capítulo anterior.

No se requiere su lectura para la comprensión de los temas aquí tratados, pero si se desea más información acerca de alguno de los programas, se deberá recurrir al Capítulo Tres.

## 4.2 COMPILACION Y LIGA DE PROGRAMAS

La aplicación ha sido codificada en Turbo C versión 2.0 y la información que sigue supone el uso de éste en la liga y compilación. En caso de utilizar otro compilador de C, se deberá recurrir a la documentación respectiva con el fin de realizar las operaciones análogas siguientes.

Para los programas TSR_POP.C, ARCHIVOS.C y SACCP.C; se procede a compilar dirigiendo la salida a disco con lo que se genera el archivo correspondiente ".EXE".

En el caso de RECIBE.C y ENVIA.C, se declara un archivo ".PRJ" (tipo proyecto) para cada uno de ellos como sigue:

RECIBE.PRJ

```
RECIBE.C
tcomms.lib
tclxms.lib
```

ENVIA.PRJ

```
ENVIA.C
tcomms.lib
tclxms.lib
```

El primer renglón corresponde al archivo fuente a compilar y ligar, los siguientes, a las librerías de comunicación utilizadas. Para generar el ".EXE", se compila a disco el proyecto, no el programa; ésto se logra declarando en "Project" (opción del menú de Turbo C el nombre de archivo ".PRJ" respectivo.

Las librerías son seleccionadas en base al tipo del modelo de compilación de Turbo C (Tiny, Small, Medium, Compact, Large y Huge). El programa TSR_POP.C, es compatible con los modelos Small y Tiny. El uso de las funciones de comunicación (archivo LCBBS.C) es recomendable con el modelo Small, razón por la que se debe compilar los programas ENVIA.C y RECIBE.C, junto con las librerías mostradas en el recuadro anterior para que no se presenten errores de ejecución. En caso contrario, se corre el riesgo de que el sistema entre en un ciclo infinito o se reinicie.

Los archivos ".LIB" y ".H" de ToolBox Communication, deben ser distribuidos en los subdirectorios LIB e INCLUDE correspondientes para su llamado desde Turbo C. (Ver apéndice E).

El algoritmo general para la compilación y liga es el siguiente:

--> Inicio

- Ejecución de Turbo C (TC [RETURN]).
- Carga del programa XXXXXXXX.C deseado.
  Se selecciona y carga con F3 o ALT-FL.
- Selección del archivo tipo proyecto XXXXXXX.PRJ
  Se da el nombre con la opción ALT-P.
- Compilación y liga a disco duro ALT-CB.

-->Fin.

## 4.3 INSTRUMENTACION

Los programas con extensión ".EXE" pueden ser ejecutados sobre un equipo de cómputo con las siguientes características básicas:

- MSDOS versión 3.03 (posterior o compatible).
- CPU con 80286 (subsecuente o compatible).
- Un puerto serial (donde se conecta el MODEM).
- Línea telefónica.
- MODEM (con velocidad de 2400 bps y compatible con Hayes).

Originalmente el sistema está diseñado para trabajar sobre COM2, (COM1, regularmente se utiliza con el MOUSE, impresora(s), juegos, comunicación local, lápiz óptico o algún otro dispositivo), si se desea trabajar sobre COM1 se deberá modificar el valor de PUERTO dentro de ENLACE.H y volver a compilar RECIBE.C y ENVIA.C,.

Un esquema a bloques de la conexión física se muestra a continuación en la figura 4.1,

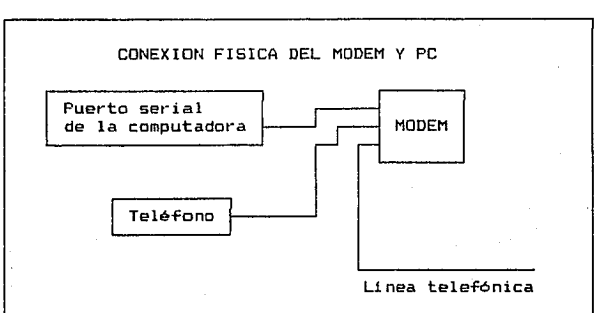



FIGURA 4.1

## 4.4 FUNCIONAMIENTO

Una vez que se ha ejecutado SACCP.EXE, y/o cargado TSR_POP.EXE, al presionar F12 el sistema pasa al modo automático, la PC atenderá llamadas y enviará archivos continuamente. Sólo son enviados los archivos declarados en ARCHIVOS.DAT mediante la opcion A (Actualización de archivos y destinos) de SACCP.EXE,. Si ARCHIVOS.DAT no existe, el sistema sólo atiende llamadas y recibe archivos.

Si durante la ejecución del TSR es presionada 'u', al finalizar el ciclo actual, el TSR es descargado y pasa el control a la línea de comandos de DOS, en caso contrario, sólo regresa a DOS y podrá reactivarse el modo automático al presionar F12. El mensaje Abnormal Termination Program, es una amonestación de DOS, y es provocado porque el TSR es detenido, los programas que llama se cierran, pero el módulo sigue residente en memoria; el DOS no logra distinguir entre si ya terminó o no y por ello envia dicha amonestación, no implica un verdadero error, sino tan sólo un aviso de que no se ha cerrado completamente el proceso anterior dado que existe aún una sección residente.

Al entrar a la opción A de SACCP.EXE, sino existe ARCHIVOS.DAT en el directorio actual, se crea y luego pasa a la actualización de registros (campos: archivo y número telefónico de destino), si ya existe pasa a la actualización del mismo.

No se requiere de la creación de un directorio específico, pero deben estar en el mismo directorio los archivos ejecutables en el momento de su llamado.

Los programas que abren archivos, buscan en toda la unidad lógica los archivos solicitados y toman el primero que encuentran, por lo que es recomendable tener cuidado con la existencia de diferentes versiones de un mismo archivo. La solución a este tipo de problemas es crear un directorio de trabajo en donde además de los ejecutables se encuentren los archivos a enviar y se depositen los archivos recibidos, para esto, siempre, antes de presionar F12 deberá de encontrarse el cursor en el directorio de trabajo definido, de lo contrario se enviará el primer archivo que se localice, y los archivos recibidos serán depositados en el directorio actual.

Los pasos a seguir para el manejo del sistema son:

(a) Ejecución de SACCP.EXE
(b) Definición de lista de archivos y destinos
    - Entrar a la opción A.
    - Insertar y modificar archivos.
    - Salida
        Si ==> (c).
        No ==> (b).
(c) Carga del módulo residente.
    - Selección de la segunda opción.
      Al hacer esto se carga el TSR y se regresa a la línea de comandos de DOS.
(d) Presionar F12 para activar el modo automático.
    - Después de esto,
        Al presionar otra tecla
            Si es 'u' ==> Descarga TSR y pasa a DOS.
            Se no es 'u" ==> Regresa a DOS.
        Si no se presiona otra tecla el sistema continuará en el modo automático.

Es posible cargar el TSR, ejecutando TSR_POP.EXE, al hacerlo, se ubica los pasos en el punto (d).

## 4.5 MANTENIMIENTO

El sistema está compuesto por módulos independientes y son modificables por separado sin afectar las partes restantes.

En el caso de modificaciones a los parámetros de comunicación (velocidad, paridad, tamaño del dato, bits de parada, tamaño del bloque, etc.), debe afectarse simultáneamente a ENVIA.C y RECIBE.C.

Cada programa (o módulo) ha sido listado en el capítulo anterior, es recomendable ver la documentación implícita de cada uno ellos para llevar a cabo cualquier modificación.

Uno de los cambios deseables podría ser el menú de SACCP.EXE y la pantalla de captura de ARCHIVOS.EXE, estos cambios son realizables sin alterar el funcionamiento general del sistema, pero esto provocaría un aumento en el tamaño de la memoria utilizada, estimado en 10Kb, del orden de 10 a 40Kb.

El control y definición de la tecla F12, debe considerarse al manejar simultáneamente y/o diseñar otros programas que lleguen a utilizar esta tecla de función, de no hacerlo, puede provocar que sea modificada su definición y que no llegue a funcionar adecuadamente el sistema.

Otro aspecto que se debe contemplar, es la configuración del MODEM. Es necesario que éste trabaje en respuesta autómatica al primer RING y en modo de respuesta con código numérico. Si es un MODEM compatible con HAYES, esto se logra enviando la sigueinte lista de comandos al MODEM:

| | |
|---|---|
| AT s0=1 | (respuesta automática al primer RING). |
| AT v0 | (modo de respuesta en código numerico). |
| AT v1 | (modo de respuesta en palabras CONNECT, NO CARRY, BUSY, ERROR, SO MUCH RETRIES, etc.). |
| AT &w | (graba los valores modificados en memoria no volatil.). |

Esto también puede lograrse desde programa, pero es recomendable hacerlo fuera de él, ya que ahorra código en nuestros diseños y tiempo en la ejecusión de los mismos.

Más información acerca del MODEM se localiza en el apéndice G.

PERSPECTIVAS

# *COMUNICACION AUTOMATICA ENTRE COMPUTADORAS PERSONALES*

# PERSPECTIVAS

El sistema SACCP (Sistema de Comunicación automática entre Computadoras Personales), es un producto terminado, pero es modificable y pauta de nuevos sistemas relacionados con la comunicación automática. Las necesidades que pretende cubrir son generales, cada entidad puede ampliar, modificar y/o reducir su funcionamiento.

Continuamente en el mercado aparecen nuevos productos, o versiones de ellos, con rasgos similares a los del sistema, pero sus costos siguen siendo altos para usuarios de escasos recursos y el número de éstos aumenta, razón que motiva al uso y perfecionamiento de SACCP.

El hardware requerido para su funcionamiento, sigue siendo un factor ajeno a este trabajo. En la actualidad, el costo de los MODEM's ha tendido a subir y las líneas telefónicas han tenido una mayor difusión, situaciones que llevan a pensar que el uso de estos medios de comunicación no se volverá obsoleto sino por el contrario será más utilizado en el futuro.

Por otro lado, el sistema fue pensado para trabajar en un momento dado con otros canales de comunicación (Por ejemplo, fibra óptica y vía satélite), permitiendo alcanzar una velocidad de transmisión mayor y continuar con su utilización.

A pesar de estar dirigido a usarios independientes, puede ser modificado y utilizado por coorporaciones con más recursos, la comercialización de versiones especificas queda a responsabilidad de los involucrados.

Dos ejemplos generales de su aplicación son:

- La sustitución del correo actual, las cartas se escribirían en una pc y entregarían mediante una copia en disco a la central o buzón (éste sería una terminal) para ser transmitidas por el sistema con una rapidez mayor a su destino.

- Repartición de propaganda o publicaciones a personas específicas, con ello no sólo se ahorra tiempo, sino todo los costos relacionados a la edición y envío de la publicación.

El primer ejemplo abarca únicamente el envío de cartas (o telegramas) de una entidad a otra, documentos cortos y de caracter personal. El segundo, es más ambicioso pues pretende hacer llegar información pública o privada a un gran número de personas sin tener canales dedicados. El análisis de los problemas que se presentarían, no serán discutidos aquí, baste mencionar que con SACCP:

a) Se reducen costos,

b) Se reduce el consumo de papel, y

c) Es más rápida la difusión de información.

***COMUNICACION AUTOMATICA ENTRE***

***COMPUTADORAS PERSONALES***

# CONCLUSIONES

# CONCLUSIONES

La intensión principal ha sido presentar una alternativa viable para el intercambio de información (tomando como unidad de la misma el archivo) y todos aquellos usuarios independientes con tal necesidad, podrán verla satisfecha con este sistema.

Las tareas de marcación, espera por enlace y supervisión de la transmisión se automatizan con él, además de disminuir los costos, dado que se parte del uso de la línea telefónica, y de MODEM's.

Las mejoras presentadas radican especificamente en la liberación del usuario de las siguientes actividades:

- Carga y/o ejecución del software de comunicación.

- Navegación dentro del mismo software para elegir las operaciones de selección de archivo, marcaje, enlace y envio (si transmite); o para elegir las operaciones de aceptación de enlace, declaración del nombre del archivo a recibir y habilitar recepción (si recibe).

- Descargar, o salir del software de comunicación y seguir con las actividades normales, o volver a navegar en él,para realizar las operaciones de recepción o transmisión según se requiera.

- Compra de software especializado en comunicaciones, un costo del orden de $700,000.00 (N$700).

- Supervisar las transacciones en el momento en que ocurren, esto implica atar a la máquina y al usuario a una misma actividad.

Con el sistema propuesto, se requiere correr SACCP.exe, definir la lista de archivos a enviar y su destino (número de teléfono a donde se enviará) y el sistema se encarga de entablar el enlace y realizar las transacciones pertinentes; para ello es necesario que ambos equipos de cómputo estén corriendo el mismo programa SACCP.EXE, estos se encargan del resto, creando además un archivo tipo bitácora donde se registra que archivos han sido enviados o recibidos y en que momento fue realizada la transacción.

Además de los alcances anteriormente mencionados, el sistema sienta bases para la implementación de software de comunicación, da a conocer librerías de C orientadas a esa área y posibles ejemplos de su utilización, por lo que tan sólo resta decir que se espera sirva para el fin que fue diseñado y desarrollado.

Atte,

Silvestre López Abundio

*COMUNICACION AUTOMATICA ENTRE*

*COMPUTADORAS PERSONALES*

# APENDICES

# A P E N D I C E -A-

Listado de algunos teléfonos para conectarse a clubs de usuarios, y solicitar información acerca del acceso y servicio que ofrece. Esta lista fue obtenida el 27 de Agosto de 1991 al enlazarce con ServiNet, D.F.

915-575-4519 ServiNet, D.F
913-621-0315 CompuTel, Guadalajara
915-395-4339 CompuCom, D.F para distribuidores de cómputo)
912-246-0241 Unitel, Puebla
915-669-0097 Chalpulnet, D.F.
915-550-5634 Facultad de Contabilidad y Administración.
915-511-9448 La Maquina del Tiempo (BBS experimental 19-9 Hrs)
915-575-8995 RadioNet D.F. (de las 23:00 a las 8:00)Radiocom.
913-642-5407 TecoTel, Guadalajara, Jal. (24 Hrs.)
915-598-7194 TeleLink (22:00 a 7:00 Hrs.) Telecomunicaciones
918-344-6068 UniRegio, Monterrey, N.L.
914-715-0916 BajioNet, Leon Gto.
915-510-2825 He aquí otro BBS con MNP5
915-521-3825 Data-Bit
913-622-7942 CyberNet Guadalajara (22:00 a 9:00 Hrs.)
915-573-5515 SCIENCE_UVM BBS Unidad del Valle de México
915-570-7037 Departamento de sistemas U.I.A (20:00 a 8:00 Hrs.)
915-546-4093 PRADSA BBS Una nueva opcion en BBS
915-272-5274 GER-NET Experimental de 7-9 pm
914-712-2718 PCNet Leon, Gto.
915-554-5132 La cueva del BBS (21:00 a 7:00)
541-224-1971 Fidocenter- Central de BBS en América Latina
809-751-7728 Mega-D RBBS - Central de Fido en Puerto Rico
598-268-0690 Compuservice -BBS multilinea en Uruguay
915-682-4671 Super BBS
915-560-1883 ¡Construyamos este BBS!

915-677-3446 El ORACULO de 7:00 a 10:00
915-543-6530 Noche mágica (horario nocturno)
915-560-9263 Super BBS'
915-570-0207 Eclipse BBS: Lun,Mar,Jue,Vier 3:00 a 6:30 pm
915-560-1883 experimental de 21:00 a 6:00 Hrs.
913-622-3319 ISAONet Guadalajara, Jal.
915-564-1679 Fimpestel México, D.F.

Entre las opciones más frecuentes que tienen:

- Comentarios.

- Información general.

- Mensajería.

El primero para mejorar el servicio y el segundo cubre rubros de información que pueden ser compartidos, como nuevos virus, vacunas, exposiciones actuales, documentos editados en otros lugares, etc; y la mensajería, está orientada a los nuevos servicios, nuevas reglas y cuotas, así como nuevos horarios.

# A P E N D I C E - B -

(Microprocesadores)

MICROPROCESADORES

| CANAL DE DATOS | MEMORIA RAM | O T R A S CARACTERISTICAS | COMPANIA | EJEMPLO |
|---|---|---|---|---|
| 4 Bits | 4<br>a<br>8<br>Kb | - Programación en ensamblador.<br>- Uso reducido de interrupciones.<br>- Canal de datos unidireccional.<br>- E/S programada. | INTEL | MCS-4<br>4040 |
| 8 Bits | 16<br>a<br>64<br>Kb | - Programación en ensamblador y lenguajes de alto nivel.<br>- Uso extensivo de interrupciones, tanto mascarables como no mascarables.<br>- Existencia y manejo de sistemas operativos propios.<br>- Canal de datos bidireccional. | INTEL<br>ZILOG<br>MOTOROLA<br>MOS-TECH<br>DEC<br>TEXAS | 8008<br>Z80<br>6800<br>6502<br>LSI<br>99000 |

| | | | | |
|---|---|---|---|---|
| | | - Especialización de control y uso de E/S programada.<br>- Velocidad de reloj de 0.2500 a 0.0925 Hz.<br>- Aplicaciones específicas. | | |
| 16,32 Bits | | - Programación en ensamblador y lenguajes de alto nivel.<br>- Uso de SO e interfaces gráficas para el aprovechamientos de los recursos.<br>- Tamaño de la palabra variable.<br>- Control externo del microprocesador.<br>- Velocidad de reloj de 6 a 50 MHz.<br>- Aplicaciones generales y/o en multitarea.<br>- Control de errores.<br>- Uso de coprocesadores.<br>- Uso de SO compatibles. | INTEL<br>MOTOROLA | S-80's<br>68000 |

# A P E N D I C E - C -

(Elementos y Selección de un equipo de cómputo)

## ELEMENTOS DE UN EQUIPO DE COMPUTO

Los elementos específicos que debe tener actualmente un equipo de cómputo son:

Capacidad de RAM mínima de 1Mb.
Disco duro.
Unidades de disco flexibles de alta densidad.
Monitor.
Teclado.
Velocidad de 16 MHz o mayor.
Ratón.
Impresora.
Puertos Seriales.
Puertos Paralelos.
Modem.
Compatibilidad con DOS vers. 3.3 o posterior.

Hoy por hoy los requerimientos de la industria mexicana en torno a equipos de cómputo quedan satifechos por los elementos citados arriba. Estos elementos pueden considerarse como indispensables actualmente. Puede trabajarse con otras configuraciones ásteras, mas ello impicará perdidas en tiempo y disminución de la capacidad de manejo. Pronto, serán otros los elementos que se incluyan en esta lista, como rastreadores, graficadores, lapiz óptico y multimedios; pero en estos momentos, con los componentes mencionados podremos desarrollar o utilizar la mayoría de las aplicaciones de proposito general y algunas de propósito especifico.

## SELECCION DE UN EQUIPO DE COMPUTO

Antes que nada y como en toda meta, ésta debe ser clara, por ello debemos primero definir el uso para el cual se requiere el equipo de cómputo. Esto sólo lo pueden definir en conjunto el usuario y un especialista en computación (Analista de Sistemas).

En general el equipo buscado deberá tener las siguientes cualidades:

COMPATIBILIDAD

ACTUALIDAD

CALIDAD

UTILIDAD

EFICIENCIA

Compatibilidad. Los elementos tanto propios como adicionales no sólo deben de existir, sino deben ser fáciles de conseguir y efectivamente funcionar de acuerdo a los estandares y a la manera deseada. Entre las características de compatibilidad tenemos la velocidad, el SO, los tipos de canales, etc., pero tengase siempre en cuenta que la compatibilidad no es únicamente entre software y software, sino además es hardware-hardware y hardware-software.

Actualidad. Cada uno de los componentes deben ser actuales y actualizables, aunque debe observarse que no sean prototipos a menos que se desee probar el funcionamiento del producto.

Calidad. El buen funcionamiento y uso de los elementos reflejarán la utilidad del equipo, pero esto también será en base a la calidad de los mismos; existen una serie de marcas que presentan una buena opción, pero no pasa mucho tiempo y uno llega a darse cuenta de fallas en la solucion, implicando pérdidas de tiempo y capital. La calidad está dada por el buen funcionamiento del equipo durante toda su vida útil y una forma de guiarse, es el prestigio de la compañía.

Utilidad. Sólo con los resultados y comparación de los mismos podrán ser tangibles las utilidades. Si se han seguido y buscado la cualidades aquí mencionadas las utilidades serán las esperadas.

Eficiencia. Un super-equipo es bueno, pero esto no implica que sea lo óptimo. Un equipo debe satisfacer lo esperado, no debe ser desperdiciado. La eficiencia no radica en pagar más o comprar lo más costoso y de mayor renombre, sino en el nivel de productibidad que dicho equipo nos brinde por su velocidad y confiabilidad.

# A P E N D I C E - D -

## TOPOLOGIAS DE REDES Y CARACTERISTICAS

Entre los diferentes puntos de contemplación de las redes de computadora, tenemos su topología o forma en que se disponen los nodos de la red. Las siguientes son las topologías básicas de las que se puede habla:

a) Jerárquica (Árbol o vertical).
b) Horizontal (Bus).
c) Estrella.
d) Anillo.
e) Malla.
f) Mixta, combinación de las anteriores.

Las características de cada una de ellas se listan a continuación, un esquema de cada una de ellas se da en el gráfico adjunto. (Figura D.1).

a) Jerárquica.

- Control en la rama o elemento superior.
- Aplicación en proceso distribuido.
- Cuellos de botella frecuentes.
- Flexibilidad para añadir nuevos elementos.
- La información pasa de punto en punto hasta llegar al destinatario.
- Fácil detección de nodos en mal funcionamiento.

b) Horizontal.

- La información pasa directamente al nodo destino.
- Un sólo canal de comunicación común.
- Dificultad para aumentar o determinar un equipo en mal funcionamiento.
- Redundancia en el envío de información. Loenviado llega a todos los nodos.
- Flexibilidad para añadir nuevos elementos.
- Aplicación en redes de área local.

c) Estrella.

- Facilidad en el control a partir del nodo central.
- Similar al jerárquico, mas presenta limitación al aplicarse a sistemas de proceso distribuido.
- Toda la información para por el nodo central.
- Dificultad para agregar nuevos nodos.
- Fácil localización de averías.
- Cuellos de botella frecuentes.

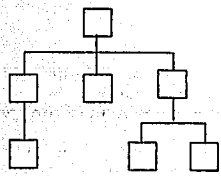
d) Anillo.

- La información pasa por la periferia del anillo de nodo en nodo con un sólo sentido, hasta encontrar el nodo destino.
- Cuellos de botella poco probables.
- Fácil localización de averías.
- Aplicación en proceso en lotes.

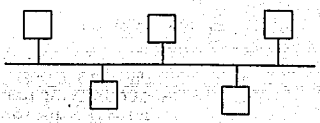
e) Malla.

- Facilidad para implementar procesos distribuidos o centralizados.
- No presenta cuellos de botella.
- Fácil localización de averías.
- Costoso y compleja agregación de nodos.
- La información es ruteada entre los nodos por la ruta más cercana y posible, en caso de una avería se re-ruteará la información hasta llegar a su destino.

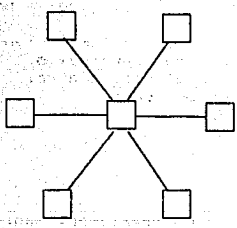
# TOPOLOGIAS DE RED BASICAS



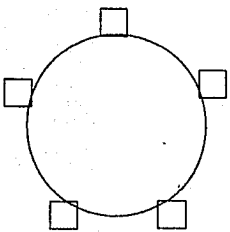
a) JERARQUICA



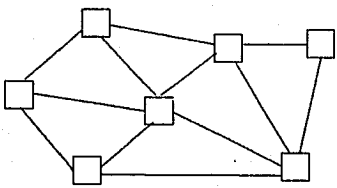
b) HORIZONTAL



c) ESTRELLA



d) ANILLO



e) MALLA

FIGURA D.1

# A P E N D I C E - E -

## (Utilerias de comunicación para C)

UTILERIAS DE COMUNICACION
(Communications ToolBox for MicroSoft and Turbo C)

La presente información, así como las funciones utilizadas en el desarrollo del presente trabajo entorno a la sección de comunicación son un producto de LiteComm (TM), con derecho de autor (Copyright) de 1987,1988,1989. La distribución de archivos es la siguiente, los de extensión ".H" y ".FNS", deben copiarse al subdirectorio INCLUDE, y los ".LIB" al subdirectorio LIB, de Turbo C.

FUNCIONES DE COMUNICACION

portchg

Resumen

```
#include <litecomm.h>
int protchg(port,base,irq,vector)
unsigned port;
unsigned base;
unsigned vector;
char irq;
```

Descripción

Cambia uno o más de los parámetros críticos para COM3 o COM4. Esta función debe ser utilizada antes de que el puerto sea abierto. Para no cambiar alguno de los parámetros y dejar el default, colocar un 0 donde corresponda. Considerese que el vector es un número de vector no una dirección o apuntador.

Valores que regresa

Retorna 0, si la función se llevo a cabo correctamente, -1, si se intenta cambiar otro puerto diferente al 3 ó 4.

comm_opn

Resumen

```
#include <litecomm.h>
int comm_opn(port,baud,parity,datab,stopb,inbufsz,
                outbufsz, raisemdm)
unsigned port;
unsigned baud;
unsigned parity;
unsigned datab;
unsigned stopb;
unsigned inbufsz;
unsigned outbufsz;
```

Descripción

Abre el puerto especificado para utilizarce sobre la interrupción de DOS correspondiente. Opcionalmente levanta las señales de control DTR y RTS del MODEM, si el valor de raisemdm es TRUE. La función permite también definir el tamaño del bloque de control de datos tanto de entrada como de salida, el mínimo valor de inbufsz es de 128 y el de outbufsz es 64.

Si se define adecuadamente la apertura de puerto, la función además indicará la velocidad, paridad y tamaño de la palabra. El consumo de memoria asociado a esta función es de 1.2K sobre el buffer especificado.

Valores que regresa

Si se abrió adecuadamente, regresa el puerto abierto, en caso contrario regresa -1.

comm_close

Resumen

```
#include <litecomm.h>
int comm_close(port,dropmdm)
unsigned port;
```

Descripción

Esta función se utiliza en conjunto con comm_opn, y realiza la operación opuesta.

Valores que regresa

Si ocurre algún error, matiene los estados actuales, si no, regresa -1.

comm_setup

Resumen

```
#include <litecomm.h>
int comm_setup(port,baud,parity,datab,stopb)
unsigned port;
unsigned baud;
unsigned parity;
unsigned datab;
unsigned stopb;
```

Descripción

La función comm_setup maneja algunos de los parámetros manejados en comm_opn. Su utilidad toma forma cuando se modifican los parámetros del puerto al estar éste ya abierto.

Valores que regresa

Si ocurre algún error matiene los estados actuales, si no, regresa -1.

lc_vport

Resumen

```
#include <litecomm.h>
COMM *lc_vport(port)
```

Descripción

lc_vport es una macro que es utilizada internamente para validar el número de puerto especificado y checar si realmente ha sido abierto o sigue abierto.

Valores que regresa

Si el puerto es valido y está abierto, regresa un apuntador al CCB (bloque de control de comunicación). Si hay error regresa NULL.

lc_icnt
lc_ocnt

Resumen

```
#include <litecomm.h>
int lc_icnt(port);
int lc_ocnt(port);
unsigned port;
```

Descripción

Estas funciones podrían ser utilizadas para determinar el número de caracteres que actualmente están entrando o saliendo por el buffer del puerto.

Valores que regresa

Si hay error regresa -1, de lo contrario regresa el número de errores detectado.

lc_mstat

Resumen

```
#include <litecomm.h>
int lc_mstat(port);
unsigned port;
```

Descripción

Puede ser utilizado para determinar el último estado conocido del MODEM, apartir de las señales de manejo.

La señales de manejo consisten de un byte en el cual los bits 4-7 contienen los estados actuales de las señales (como Limpiado para envio CTS) y los bits 0-3 contienen información entorno a la variación de las señales.

Valores que regresa

Si el puerto es válido y está abierto regresa el byte indicando el estado actual, si hubo error regresa -1.

lc_estat

Resumen

```
#include <litecomm.h>
int lc_estat(port)
unsigned port;
```

Descripción

Puede ser utilizado para determinar el último estado de los bits de error del puerto.

Los tipos de error reportados son:

ORUNERR - Falla en la busqueda de caracteres desde el puerto antes del siguiente caracter recibido, implica error en el manejador de interrupción.

PARERR - Uno o más caracteres recibidos con otra paridad, usualmente es provocado por ruido en el canal.

FRMERR - El tamaño de la palabra recibida fue mayor al especificado, usualmente es provocado por ruido en el canal.

lc_getw

Resumen

```
#include <litecomm.h>
int lc_getw(port)
unsigned port;
```

Descripción

Lee un caracter desde el buffer de entrada del puerto serial, espera indefinidamente hasta que llegue un caracter. Si el puerto es desconectado por alguna razón, el sistema queda bloqueado.

Valores que regresa

Regresa el siguiente caracter disponible en el buffer de entrada del puerto serial. Regresa -1 si el puerto no está activo, o es inválido el número de puerto especificado.

lc_setmdm

Resumen

```
#include <litecomm.h>
int lc_setmdm(prot,newset)
unsigned port;
unsigned newset;
```

Descripción

Levanta una o más de las señales del MODEM, newset, son valores predefinidos de las señales a levantar y se encuentran en la librería que se incluye.

Valores que regresa

Regresa 0 si la operación fue realizada adecuadamente, y -1 en caso contrario.

lc_clrmdm

Resumen

```
#include <litecomm.h>
int lc_clrmdm(port,newset)
unsigned port;
unsigned nwset;
```

Descripción

Baja una o varias de las señales del MODEM, es función opuesta a lc_setmdm.

Valores que regresa

Regresa 0 si la operación fue realizada adecuadamente, y -1 en caso contrario.

lc_togmdm

Resumen

```
#include <litecomm.h>
int lc_togmdm(port,newset)
unsigned port;
unsigned newset;
```

Descripción

Invierte el estado actual de las señales del MODEM, se utiliza en conjunción con lc_setmdm y lc_clrmdm.

Valores que regresa

Regresa 0 si la operación fue realizada adecuadamente, y -1 en caso contrario.

lc_xoff

Resumen

```
#include <litecomm.h>
int lc_xoff(port,flag)
unsigned port;
int flag;
```

Descripción

Si la bandera es diferente de cero, la función habilita la funcion de flujo semiautomático. Si es 0 (default) el control de flujo es deshabilidato. Si se habilita, se enviarán fines virtuales de los datos enviados para el control de colisiones.

Valores que regresa

Regresa 0 si la operación fue realizada adecuadamente, y -1 en caso contrario.

lc_gotxoff

Resumen

```
#include <litecomm.h>
int lc_togxoff(port)
unsigned port;
```

Descripción, invierte el valor de las señales XOFF y XON para permitir la continuidad de los enlaces al utilizar el control de flujo.

Valores que regresa

Regresa un valor diferente de 0 si fue detectado un XOFF, 0 si no es XOFF, y regresa -1 en caso de error.

lc_putxoff

Resumen

```
#include <litecomm.h>
lc_putxoff(port)
unsigned port;
```

Descripción

Ver valores que regresa.

Valores que regresa

Regresa un valor diferente de 0, si un XOFF fue enviado al otro disposito, 0 si un XOFF no fue enviado en la última función llamada, y un -1 en caso de error.

lc_get

Resumen

```
#include <litecomm.h>
int lc_get(port)
unsigned port;
```

Descripción

Lee un caracter desde el buffer de entrada del puerto serial.

Valores que regresa

Regresa el siguiente caracter en el buffer de entrada del puerto. Si se ha especificado alguna paridad, ésta tiene que ser removida antes de utilizar el caracter. Regresa -1 si el puerto no esta activo, o si no hay caracteres en el buffer.

lc_peek

Resumen

```
#include <litecomm.h>
int lc_peek(port)
unsigned port;
```

Descripción

Busca el siguiente caracter en el buffer de entrada del puerto serial.

Valores que regresa

El siguiente caracter disponible en el buffer de entrada del puerto, no remueve el caracter del buffer.

lc_put

Resumen

```
#include <litecomm.h>
int lc_put(port,ch)
unsigned port;
char ch;
```

Descripción

Pone un caracter dentro del buffer de salida del puerto serial.

Valores que regresa

Regresa 0 si se realiza adecuadamente. Este hecho no implica que el caracter haya sido enviado. -1, si el puerto no está activo o si no hay espacio en el buffer del puerto.

lc_gets

Resumen

```
#include <litecomm.h>
int lc_gets(port,buff,cnt)
unsigned port;
char *buff;
int cnt;
```

Descripción

Lee una cadena de cnt -o más-, caracteres desde el buffer de entrada del puerto serial y lo deja en buff. Esta función no checa si la cadena es nula. El manejo de paridad implica que se debe eliminarla antes de tomar en cuenta la cadena.

Valores que regresa

El número de caracteres transmitidos a buff, o -1 en caso de error.

lc_puts

Resumen

```
#include <litecomm.h>
int lc_puts(port,buff,cnt)
unsigned port;
char *buff;
int cnt;
```

Descripción

Coloca la cadena de cnt caracteres -o más-, dentro del buffer de salida del puerto.

Valores que regesa

cnt, aunque esto no implica que la cadena haya sido transferida al otro MODEM. Regresa 0 si se presenta al gún error o si el buffer de salida esta lleno.

lc_flush

Resumen

```
#include <litecomm.h>
int lc_tflush(port)
int lc_rflush(port)
int lc_flshtrue(port,ch);
int lc_nflush(port,cnt);
unsigned port;
char ch;
int cnt;
```

Descripción

Las funciones lc_[x]flush remueven caracteres del buffer especificado y los descarta.

lc_tflush() limpia inmediatamente el buffer transmitido.

lc_rflush() limpia inmediatamente el buffer recibido.

lc_flshtrue() limpia cuando se recibe el caracter ch.

lc_nflush() limpia cuando los cnt caracteres se encuentran en el buffer de entrada.

Valores que regresa

lc_flshtrue() no regresa valores.

lc_tflush y lc_nflush regresa 0 si no hubo error y -1 si lo hubo.

lcnflush, el número de caracteres removidos y 0 en caso contrario.

lc_sbrk

Resumen

```
#include <litecomm.h>
int lc_sbrk(port)
int lc_gotbrk(port)
unsigned port;
```

Descripción

lc_sbrk() genera una señal BREAK utilizando características particulares del UART 8250. lc_gotbrk, checa si ha llegado una señal BREAK.

Valores que regresa

lc_gotbrk, un valor diferente de 0 si una señal BREAK fue recibida en el puerto especificado y 0 en caso contrario.

lc_sbreak, 0 si no hubo error, y -1 si el puerto no se encuentra activo.

new_event

Resumen

```
#include "lcbbs.h";
long new_event(sec);
int check_event(evtimer);
unsigned sec;
long evtimer;
```

Descripción

La función new_event crea un 'temporizador de eventos' el cual termina en sec segundos. Sirve para controlar tiempos muertos dentro de las aplicaciones.

La función check_event examina el contenido de un temorizador de eventos creado por new_event con respecto a la hora y fecha actual e indica si ya ha finalizado o no.

Si se utiliza check_event sin haber usado antes new_event, posiblemente el sistema quede trabado.

Valores que regresan

new_event, un valor en segundos a partir del cual el temporizador fue llamado. check_event, regresa un valor diferente de 0 si aún no termina el evento y 0 en caso contrario.

chech_for_call

Resumen

```
#include "lcbbs.h"
int chekc_call(port)
unsigned port;
```

Descripción

Esta función checa el estatus del puerto especificado para determinar si hay alguna llamada y de ser así espera 30 segundos por la portadora y entablar el enlace.

Valores que regresa

Si no esta sonando el telefono 0, si suena y después de 30 segundos no hay conexión -1, y en tcualquier otro caso, un código número de interpretación a jucio del programador.

get_modem_reply

Resumen

```
#include "lcbbs.h"
int get_modem_reply(port)
unsigned port;
```

Descripción

Esta función sirve para obtener respuestas números en lugar de palabras del MODEM después de alguna transacción realizada. Para ello se debe configurar primeramente el MODEM para responder con códigos numéricos en lugar de palabras.

Valores que regresa
Si no hay respuesta del MODEM en 1 segundo, -1, en otro caso el código correspondiente.

reset_modem

Resumen
```
#include "lcbbs.h"
int reset_modem(port)
unsigned port;
```

Descripción
Reinicializa el MODEM con los parámetros de default grabados o configurados en la memoria no volátil del mismo.
Dentro de la configuración se debe considerar
- Supresión del eco de comandos.
- Detección de la portadora.
- Respuesta en código numérico en lugar del de palabras.
- Respuesta automática a los n rings-telefónicos.

Valores que regresa
El código respectivo a la transacción.

disconnect

Resumen
```
#include "lcbbs.h"
void disconnect(port)
unsigned port;
```

Descripción
Forza el rompimiento de enlace bajando la señal DTR momentáneamente. No regresa valores.

## FUNCIONES DE COMUNICACION PARA MODEM's HAYES Y COMPATIBLES

Cuando se utilizan las sigueintes funciones, se deben de incluir el archivo litehcm.h, el cual a su vez incluye al litecomm.h

lch_codeset
lch_dial
lch_fduplex
lch_hduplex
lch_greg
lch_sreg
lch_offcarr
lch_oncarr
lch_pulse
lch_tone
lch_speaker
lch_offecho
lch_onecho
lch_hook
lch_redo
lch_numres
lch_wrdres
lch_codesoff
lch_codeson
_retset
_rettype

Resumen

```
#include <litehcm.h>
int lch_codeset(port,mode)
int lch_dial(port,dstr)
int lch_fduplex(port)
int lch_hduplex(port)
int lch_greg(port,reg)
int lch_sreg(port,reg,value)
int lch_offcarr(prot)
int lch_oncarr(port)
int lch_offecho(port)
int lch_onecho(port)
int lch_hook(port,hmode)
int lch_redo(port)
int lch_numres(port)
int lch_wrdres(port)
int lch_codesoff(port)
int lch_codeson(port)
int lch_pulse(port)
```

```
int lch_tone(port)
int lch_speaker(port,spkmode)
int _retset()
int _rettype()
unsigned prot;
unsigned mode;
char *dstr;
unsigned reg;
int value;
unsigned hmode;
unsigned spkmode;
```

Descripción

Los valores utilizados en conjunción con mode, hmode y spkmode son definidos simbólicamente en el archivo litehcm.h .

lch_codeset, permite cambiar el código que regresa el MODEM después de alguna transacción.

lch_dial, le indica al MODEM marcar el número en dstr, no se requiere agregar ATD o \r, la función los agrega.

lch_hduple:, lchfduplex; pone al MODEM en echo local y remoto respectivamente.

lch_greg, lch_sreg; hacen que el MODEM le de el valor contenido en el S-registro (0-13), regresan -1 en caso de error.
La primera lo trae del MODEM, la segunda pone ese valor en el registro.

lch_offcarr, habilita el la detección de portadora del MODEM, pero deshabilita la señal de portadora. La función lch_oncarr habilita ambas. Cuando lch_ofcarr es utilizada el MODEM puede recibir datos, pero no enviarlos.

lch_offecho, lch_onecho; determinan cuando los commandos enviados al MODEM tienen eco para ser enviados al programa.

lch_hook, permite tener el control de los estados actuales del MODEM.

lch_redo, le indica al MODEM realizar nuevamente la ultima secuencia de comandos.

lcn_wrdres, numres; habilitan la respuestas del MODEM con palabras en ingles o con códigos numericos respectivamente.

lch_speaker, permite el control de la bocina interna del MODEM, si es que cuenta con una .

_retset, regresa el último conocido comando.

_rettype, regresa el último resultado obtenido en la última transacción.

Valores que regresan

Todas regresan -1 si hubo error y 0 en caso contrario.

# A P E N D I C E -F-

(Términos manejados en la obra)

BBS.-(Bulletin Board System se interpreta como Sistema de boletín electrónico). Estas siglas hacen referencia a una especie de club de computación a los cuales una persona puede enlazarce para compartir recursos de cómputo.

Bridges.- Los puentes (bridges) son dispositivos que operan en la capa de enlace de datos (capa 2 de OSI) e interconectan a dos redes de tal manera que los usuarios piensen que están conectados a una sola red. Los bridges son más eficientes que los repetidores en esta tarea, debido a que los primeros envían solamente los datos que son necesarios para la otra red. Esto reduce bastante la cantidad de tráfico que se transmite entre las dos redes, pero requiere que ambas sean de la misma topología( Por ejemplo Ethernet con Ethernet). Esto es, el puente no sólo renova la señal sino codifica y decodifica las señales optimizando su direccionamiento.

Dato.- Elemento del que se parte para generar información. Es todo hecho o evento que ocurre a nuestro alrededor.

Gateway.- Son dispositivos que operan en la capa de aplicación (capa 7 de OSI) e interconectan a redes de diferentes tipos. Realizan la interconexión de la manera más directa posible. Literalmente convierten la salida de los dispositivos de una red en el protocolo que entienden los dispositivos de la otra red. A diferencia de los ruteadores, los *gateways* de hecho traducen un protocolo a otro. Los ruteadores agregan otro nivel de información de dirección de los datos, de modo que estos se puedan enviar a la otra red. Cuando se utiliza un *gateway* para interconectar diferentes redes, o dispositivos de una red se comunican con los de la otra como si fueran de la misma clase.

Información.- La Información es un evento que cobra significado ante nosotros, razón por la cual puede decirse que es todo dato procesado o evento percibido.

Interrupción.- Es una señal que puede ser por origen de software o hardware enviada al procesador de la computadora para que realice una operación correspondiente. Cuando se presenta la interrupción el procesador detiene el proceso actual, salva sus registros, y una vez que termina la rutina solicitada el proceso anterior continua.

Libreria.- En programación, son archivos pertenecientes a lenguajes como C o Pascal, en donde previamente se han definido estructuras, constantes, funciones y módulos para facilitar la tarea del programador. Una vez declarados correctamente pueden ser incluidos en diferentes programas.

Manejador de interrupción.- Es la interrupción que se encarga de manejar la entrada-salidad de las interrupciones especificas.

MAU.- (Multi Access Unit, Unidad de acceso multiple), Dispositivo de hardware que utilizan las redes de anillo para evitar que la red se rompa en caso de faltar o fallar alguna de las estaciones de trabajo. El dispositivo "puentea" la sección de la estación faltante y permite la continuidad de trabajo de la red. Comunmente son de cuatro puertos, los cual le permite "puentear" hasta cuatro estaciónes de trabajo fuera de servicio.

Procesamiento.- Cambio o metabolismo en el cual se parte de condiciones iniciales y se llega a un aprovechamiento de los elementos que intervienen.

Proceso.- Camino, método o conjunto de actividades con el que se logra transformar y aprovechar los recursos de un sistema.

Repetidores.- Los repetidores operan en la capa física (capa 1 de OSI) e interconectan a dos segmentos de la misma red que estén geográficamente separados. Los repetidores generan las señales en la línea de comunicación y son transparentes al usuario, su finalidad es renovar las señales iniciales ya que éstas van siendo atenuadas por la distancia.

Routers.- Los ruteadores son dispositivos que operan en la capa de red (capa 3 de OSI) y pueden interconectar a dos redes diferentes (Por ejemplo Ethernet con Token ring). Sus capacidades van mucho más allá de las de un puente. Debido a que los ruteadores no son transparentes, los usuarios que desean tener acceso a los recursos o a otra red, deben dar una dirección de destino. Esta dirección es un nombre lógico que el ruteador convierte en algo conocido como Dirección Interna de Red. Los ruteadores utilizan la dirección interna para encontrar los destinos siempre que estén en la red. Ya que los ruteadores realmente "hablan" con los dispositivos de la red, cada uno debe soportar los protocolos que utilizan los dispositivos de esa red. Esto hace que los ruteadores sean más costosos, más complejos de establecer y mantener, y más lentos para establecer la conexión que los puentes. Sin embargo, son el producto a utilizar cuando el requerimiento es establecer conexiones de extremo a extremo en cualquier parte de la red, independientemente del tipo de red (Por ejemplo X.25, Ethernet, y Token Ring).

Sistema.- Referencia léxica y semántica al conjunto de elementos que se integran e interactúan para obtener un fin común.

# A P E N D I C E - F -

## ( Manejo del MODEM )

La adquisición de un MODEM, va acompañada de un manual de operación, es en tal documento donde se encontrarán las funciones, especificaciones y forma de configurar esta interfaz.

La información siguiente no pretende sustituir tal documentación, sino dar un panorama general de la instalación, prueba y modificación de registros de MODEM's, bajo la consideración de tratarse de un MODEM compatible con el estandar HAYES.

El MODEM (M), puede ser interno o externo, si es externo, se requiere conectar mediante un cable serial el puerto serial de la PC (COM1, COM2, COM3 COM4) con M.

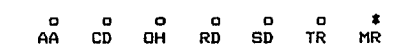
Si es interno, se abrirá el gabinete de la CPU e insertará M, en alguna de las ranuras de expansión libres.

M, debe ser configurado para trabajar con el respectivo puerto serial, para hacer esto, deben modificarse los micro-interruptores de M, en base a las especificaciones señaladas en el manual de operación.

La instalación de M, debe hacerse con el equipo apagado para portección del mismo y del operador. Luego de instalarce se puede encender el equipo, si M es externo habrá que encenderlo en forma independiente, si es interno encenderá simultáneamente.

En caso de observar alguna anomalía, se deberá apagar el equipo, revisar conexiones y configuración. Si no llega a funcionar M: Si es interno, hay que acudir con el vendedor para cambiarlo, si es externo, debe revisarse primero el cable y el puerto, quiza alguno de ellos no sea el adecuado.

Cuando M enciende correctamente, encienden al menos la señal MR, tal como se muestra en el siguiente esquema de un panel de un MODEM externo,




AA -- Auto answer
On, Responde automáticamente.
Off, No responde a una llamada telefónica.
Intermitente, recibiendo llamada, (Ring).

CD -- Carrier Detect
On, Se detecta señal portadora remota.
Off, no se detecta señal portadora.

OH -- Off Hook
On, teléfono descolgado.
Off, teléfono colgado.

RD Receive Data
On, Recibiendo datos.
Off, Sin recibir datos.

SD -- Send Data
On, enviando datos.
Off, sin enviar datos.

TR -- Terminal Ready
On, MODEM listo para recibir datos dede la PC.
Off, Fuera de línea con la PC.

MR -- Modem Ready
On, Encendido.
Off, Apagado.

Si encienden todos los Led's indicadores, implicará un mal funcionamiento. Sólo durante la transferencia es correcto que enciendan todos ellos simultáneamente.

Si M, no es externo,y en general, deben correrse los programas de auxilio (manejo y chequeo de M) que vienen en la compra de M. Pueden utilizarce otros programas compatibles para el chequeo de M como: EZ, QBTTL, SMARTCOM, PROCOMM, CrossTalk o QAMODEM.

Una vez que se tiene establecida la comunicación con M, puede manejarse éste desde la PC con ayuda de un programa específico como los presentados en el Capitulo Tres (ENVIA.C y RECIBE.C) o a través de los programas de manejo de MODEM's.

Es importante tener presente los tipos de registro que pueden modificarse, (y cómo se modifican), para alcanzar un funcionamiento óptimo de M. A continuación y finalizando este apendice se describen algunos de los más reelevantes, así como los comandos los modifican.

| Registro | Rango | Valor Default | Función |
|---|---|---|---|
| S0 | 0-255 rings | 0 | Número de ring antes de contestar llamada. |
| S1 | 0-255 rings | 0 | Contador de rings. |
| S2 | 0-127 ASCII | 43 | Caracter de ESCAPE |
| S3 | 0-127 ASCII | 13 | Caracter de retorno de carro. |
| S4 | 0-127 ASCII | 10 | Caracter de alimentación de línea. |
| S5 | 0-32,127 ASCII | 8 | Caracter de BackEspace. |
| S6 | 0-255 segundos | 2 | Espera por tono. |
| S7 | 1-255 segundos | 30 | Espera por señal de portadora remota. |
| S8 | 0-255 segundos | 2 | Tiempo máximo de pausa. |

| | | | |
|---|---|---|---|
| S9 | 1-255<br>1/10 segundos | 6 | Tiempo de respuesta al detectar portadora. |
| S10 | 1-255<br>1/10 segundos | 14 | Tiempo muerto entre la perdida de portadora y colgado. |

Para modificar alguno de ellos se envia a M, en el modo de comando la instrucción:

AT SX = Z

donde X es el número de registro y Z es el nuevo valor.

Cada uno de ellos puede ser modificado y probar su funcionamiento. En ocaciones, es recomendable delimitar y grabar una configuración diferente a la de fábrica para no tener que reconfigurar M, cada vez que sea encendido; en otras, es preferible cambiar temporalmente los patrones de funcionamiento. Y por estas razones es necesario contar con la información básica descrita en esta sección.

Algunos comandos de interés que ayudan al manejo de M, son los que se listan acontinuación. A todos ellos le antecede 'AT' para su ejecución.

| Comando | Función |
|---|---|
| A | Respuesta manual a llamada telefónica. |
| B0 | Compatibilidad con CCITT V.21/V.22/V.22Bis |
| B1 | Compatibilidad con Bell 103/212A (varía de acuerdo al MODEM). |

| Velocidad | Forma de los datos | Compatibilidad |
|---|---|---|
| 0-300 | Asincrona | Bell 103<br>CCITT V.21 |
| 600 | Asincrona o sincrona | CCITT V.22 |
| 1200 | Asincrona o sincrona<br>o<br>o | CCITT V.22<br>Bell 212A<br>CCITT V.22Bis |
| 2400 | Sincrona | CCITT V.22Bis |

| | |
|---|---|
| D_ | _ número telefónico, parámetros.<br>marca el número telefónico.<br>Parámetros<br>T marcaje por tono.<br>P marcaje por pulso.<br>W espera un segundo por tono.<br>, pausa.<br>; regresa a modo de comandos despes de marcar. |
| DS_ | _ número telefonico 0-9 grabado en el MODEM.<br>Marca el número telefónico. |
| E0<br>E1 | Sin eco de comando.<br>Con eco de comando. |
| +++ | Cambia de modo de datos a modo de comandos. |
| H0<br>H1 | Cuelga teléfono.<br>Descuelga teléfono. |
| I0<br>I1<br>I2 | Despliega código de identificación del MODEM.<br>Prueba de fabricación.<br>Prueba interna del MODEM. |
| L1<br>L2<br>L3 | Volumen bajo de la bocina.<br>Volumen medio de la bocina.<br>Molumen máximo de la bocina. |
| Sr? | r, número de registro.<br>Despliega el valor actual del registro r. |
| Z | Reset. |
| F | Reestablece configuración de fábrica. |
| &C1<br>&C2 | Levanta portadora y la mantiene.<br>Levanta portadora cuando detecta la portadora remota. |
| &Z= | Graba el número telefónico en memoria no volátil. |
| &W | Graba en memoria no volátil la configuración actual. |

| | |
|---|---|
| &T0 | Termina la prueba actual. |
| &T1 | Prueba analógica local. |
| &T3 | Prueba digital local. |
| &T6 | Prueba digital remota. |
| &T7 | Prueba digital remota e interna. |
| &T8 | Prueba analógica remota e interna. |

***COMUNICACION AUTOMATICA ENTRE***

***COMPUTADORAS PERSONALES***

**ESCRITOS DE APOYO**

# BIBLIOGRAFIA Y HEMEROGRAFIA

## BIBLIOGRAFIA

- Introducción al Teleprocesamiento. Martin James. DIANA, México 1975. 311 pags.

- Sistemas de comunicación. Simón Haykin. Interamericana, México 1985. 650 pags.

- Tratamiento automático de la información. Bernardo Jean. AGUILAR, Madrid-España 1973. 489 pags.

- Communications Arquitecture for distribued systems. Cypser, R.J., Addison_Wesley, Mass.-E.U.N.A..

- Programación de microprocesadores 8086/8088 e interfaces. M. en C. Oscar Olmedo A. (et. al.). Centro de Investigación y Estudios Avanzados del IPN., Dpto. de Ing. Elec., Secc. Comp. 1988.

- Digital Network and computer systems. Taylor L. Eooth. Wesley, New York-E.U.N.A., 1978.

- Computer communications. Cole Roberts. MCMILLAN, Londres-G.B., 1982. 200 pags.

- Redes de Computadoras, Protocolos, Normas e Interfaces. Ulysess Black. Macrobit, México 1990.

- Redes de Ordenadores. Andrew S. Tanebaum. Prentice Hall, México 1991.

- Advanced C Programming. Brady, The Peter Norton Programming Series, New York 1992.

- C Primer Plus. Mitchell Wait. Howard W. Sams & Co. Inc. Indiana.

- Manuales de Turbo C++. Borland, E.U., 1991.

- Biblioteca de la informática. Noriega-LIMUSA, México 1990. Tomos I,II y V (Tot. tomos 7).

- Sistemas Operativos (Conceptos y diseño). Milenkovic, Milan. Mc Graw Hill., Madrid, España 1988. Primera edición. 618 pags.

- Diccionario de la informática. P. Guirao. PRISMA, México. 278 pags.

- Tesis profesionales. Mendieta Alatorre Ma. de los Angeles. Porrúa, México 1976. 258 pags.

HEMEROGRAFIA

- PC MAGAZINE (en español) Volúmenes: Abril 1990-Octubre 1992.