87
2
9

UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA DE MONITOREO DE TELEFONOS PUBLICOS "ALCANCIAS"

T E S I S

QUE PARA OBTENER EL TITULO DE
INGENIERO EN COMPUTACION
P R E S E N T A N

ARACELI IVONNE OLEA CRUZ
ALEJANDRO GONZALEZ PALACIOS
ALFONSO RODRIGUEZ CISNEROS
FAUSTINO NEVAREZ LOPEZ

ASESOR DE TESIS:
M. I. JUAN CARLOS ROA BEIZA

1996

A mis padres y hermano, que de ellos he recibido cariño y apoyo incondicional, para seguir creciendo como persona y profesionalmente.

A cada una de las personas que se encuentran o encontraron a mi alrededor, que de cierta manera me apoyaron y aprendí algo de ellas, directa o indirectamente.

Las palabras serían pocas para expresar mi cariño y agradecimiento a cada una de ellas. Sin olvidar a Dios.

Ivonne.

"Si te sientes deprimido, canta.
Si estas triste, ríe.
Si tienes miedo lánzate adelante.
Que no hay tiempo que no llegue;
ni plazo que no se cumpla."

A mis padres:

Josefina y Joel por su gran paciencia para mostrarme el camino cuando era niño y no permitirme olvidarlo cuando era adolescente.
Por su apoyo cariño y comprensión en todos estos años.

A mis hermanos:

Jenny, Jorge, Adrian y Nancy por su compañía cariño paciencia conocimientos y ejemplo.

A mis abuelitos tíos y primos:

Ya que gracias ha todos ustedes puedo formar parte de esta familia tan unida y alegre que somos.

A mis Amigos:

No quise escribir ningún nombre por que afortunadamente tengo muchos amigos y no quiero que se me pase alguno. Pero se que todos ustedes saben cuan importantes son para mi.

A mi país México:

A tus instituciones educativas publicas y maestros entregados a la difícil tarea de la educación que me dieron la oportunidad de estudiar una carrera.

Alejandro

A mis padres:

Que me brindaron la oportunidad de estudiar una carrera y en todo momento me dieron la libertad de decidir pero con la sabia dirección y consejos de ellos. Y por estimularme a pesar del tiempo a terminar la carrera.

Para mi han sido un verdadero ejemplo, puesto que muchos padres solo dan a sus hijos lo que ellos recibieron de sus padres, pero ellos nos dieron lo que ellos nunca tuvieron, definitivamente la carrera que ellos emprendieron como padres lo han hecho y lo siguen haciendo con excelencia.

A mis hermanos:

Que siempre conté con su cariño, estímulo y ayuda en todo momento que la necesitaba, el titulo de hermanos se gana al nacer de los mismos padres pero el hecho de serlo me lo han demostrado toda la vida.

A mi novia:

Por haber soportado todo este tiempo que tuvimos que estar separados, apoyándome con su comprensión y cariño tan especial que toda persona necesita, y por su amor que fue uno de mis estímulos para iniciar este trabajo.

Y sobre todo al Señor

Por darme la oportunidad de la vida misma, y también por darme la oportunidad de encontrar lo mejor que me ha pasado en la vida y es la ocasión de conocer a su hijo Jesús. Y que a pesar de que la ciencia explica las cosas del mundo, el nos explica sabiduría de lo alto.

Alfonso.

"El principio de toda sabiduría es el temor en el Señor" Prov. 1.7.

Si quieres tener algo que nunca antes haz tenido debes hacer algo que nunca haz hecho:
Bill Wilson.

A mi Familia.

A mis Padres.

A mi Padre (Barrigón):

Papá, esta tesis no es para nadie más que para tí. Porque tu comprendes perfectamente lo que significa la Facultad de Ingeniería, lo que significa un examen en ella, lo que es un maestro y lo que es un salón de clase de esta Facultad. Te dedico esta tesis porque en ella se encierran los esfuerzos y el trabajo que se necesitan para tener un título tan valioso como el de ser Ingeniero. Tu esfuerzo es el mío, tu trabajo es el mío, tus ilusiones son las mías, tus alegrías son las mías y también tus penas son mías. Hoy, que tengo el triunfo más importante de mi vida, te digo que este triunfo es tuyo más que mío. Te amo y mucho.

A mi Madre (Turtle):

Mamita, esto no es nada en comparación con lo que tu haz hecho por mí. Tu vida entera, tus lagrimas, tus preocupaciones, tus regaños, tus enseñanzas, tu ejemplo, tu dedicación, tus mortificaciones. Todo eso y más me diste y no pediste nada y yo te he dado muy poco. Esta tesis, no es más que la culminación de una parte de mi vida y sin embargo, es necesario recordar el comienzo, cuando todas las mañanas me llevabas y a medio día me recogías en la escuela. Ayudarme a hacer la tarea y explicarme lo que no entendía era de diario y además lo hacías con mucha paciencia y con un gran amor. De tí, aprendí el tesón y la lucha, a no darme por vencido y para ser sincero a soñar. Toda la carrera la hice con tesón y en muchas ocasiones que sentía que ya no podía, algo invisible me impulsaba y ese algo eras tú. Lo de hoy solo es la culminación de algo que empezaste hace 26 años. Te amo y mucho.

A mis hermanos.

José (Chepo):

Porque a tu examen profesional fue al primero que fui y desde ese momento quise estudiar en C.U. y titularme. Gracias por pegarme tu gusto por la música y las cosas buenas de la vida. Te Amo.

Juan (Chino):

Pongo todas mis energías, fe, karma o lo que sea y haga falta, para que tú también te titules y pronto. Noches de estudio y el interés y respeto por todo lo que significa vida es lo que deseo agradecerte siempre. Te Amo.

Martha (Gordis):

Gordita, recuerdo perfectamente cuando te dictaba y te ayudaba en tus trabajos de la escuela. También recuerdo cuando jugabas conmigo de niño. Gracias por todo lo que me quieres. Te Amo.

Quiero que sepan que ustedes tres fueron mis primeros ídolos.

A mis sobrinos (en orden de aparición):

Paty, Diana, Carlos, Tanía y los que se acumulen. Hijos: recuerden que ustedes deben de hacer en esta vida más de lo que sus abuelos, sus padres y nosotros sus tíos, hemos hecho y haremos en esta vida. Sean siempre positivos y confíen en ustedes mismos. Todo se puede si deberás se quiere.

A mis cuñados (también en orden de aparición):

Javier y Patricia: Porque con su ejemplo y a veces con pláticas, son los que me enseñan que es lo que se debe de hacer y como se debe de hacer. Gracias.

A mi abuela

A mis amigos desde la infancia:

Agustín Millan (Mamut), Guillermo Jaime Saraco (Memo), Sergio Rodríguez (Ruco), Virgilio López (Sangano),

Porque de ustedes he aprendido, he reflexionado, me han ayudado y me han brindado su amistad incondicionalmente.

A mis amigos de la Facultad (nuevamente en orden de aparición):

Rafael Henández Olvera, Alfredo Gutiérrez (Zanahoria), Raúl Medina (El Guero), Enrique Laviño (El Chino), Helaman Saunders (El Gordo), Lorena Herrera (Rangelita), Rodolfo Landa (El Guiri Guiri), Victor López (Vitor), Jaime Martínez Anzures (Mi Hijo el Bobo), Jaime Martínez Espitia (El Bulto), Laura Lezama (aunque me hiciste sufrir), y todos mis demás amigos de la Facultad.

Gracias por compartir todas las experiencias que viví con ustedes en la Facultad. Los exámenes, las horas de estudio (fueron pocas), las horas de sano esparcimiento (fueron muchas), las alegrías y las penurias. Gracias.

A mis amigos del trabajo:

Margarita, Miguel, Chaparro, Germán, Pacheco, Silvia, Rocio, Jesus, Raúl, Angela, Lic. Bonequi.

Por haberme soportado y aguantado como soy en todo este tiempo.

A mis jefes Lic. Manuel Rocha y Ma. Eugenia Vargas.

Por el apoyo para hacer esta tesis.

A mis Jefazos:

Lic. David López Manriquez, Lic. Alejandro Baca Pasos.

Porque ustedes fueron mi guía y un ejemplo digno de ser imitado. Gracias por su liderazgo y su amistad.

A mis maestros:

Porque a ellos les debo todo el conocimiento que tengo sobre este mundo.

A mi Escuela: la Facultad de Ingeniería.

Porque es el corazón de la Universidad.

A mi Universidad: La Universidad Nacional Autónoma de México.

La mejor, la única, la insuperable.

A Dios.

Por dejarme vivir en este mundo y darme la oportunidad de investigar que es la vida.

A todas las personas que conozco y cuyo trato me permite sentirme vivo.

Faustino.

CAPITULO I CONCEPTOS BASICOS.

CAPITULO II PLANTEAMIENTO DEL PROBLEMA Y PROPUESTA DE SOLUCION.

CAPITULO III DISEÑO Y DESARROLLO DEL SISTEMA

APENDICES

CAPITULO I

CONCEPTOS BASICOS

1.1 Centrales Telefónicas

1.1.1 Antecedentes.

Durante un gran lapso de tiempo, la tendencia de las redes de telecomunicaciones han consistido de switcheo, transmisión y equipo auxiliar desarrollados por diferentes fabricantes. De tal manera que muchas tecnologías y diferentes filosofías de sistemas se presentan en estas redes.

Hoy en día las redes de telecomunicaciones están siendo desarrolladas a un paso vertiginoso; switcheo digital y transmisión digital son instaladas e integradas; nuevos servicios y capacidades de manejo son añadidos a los servicios de red y se incrementa rápidamente la base de los clientes.

Todo esto permitió un incremento en la presión en los administradores para mantener un manejo eficiente de redes y de mantener los costos bajo control a un servicio de alto nivel.

El uso de computadoras para asistir en la operación de las redes ha sido la vía por muchos años. Esto sin embargo, ha dado como resultado que en una compañía de telecomunicaciones se tengan diversos equipos, de diferentes fabricantes y muchos sistemas para varios propósitos de manejo.

Los administradores de telecomunicaciones de ahora, están encarando un rápido cambio en las características de las redes, debido al hecho del incremento en el número de switches digitales que son instalados. Esto agiliza en la operación de las redes por el hecho de que los switches digitales están siendo controlados por datos grabados en memoria, los cuales pueden ser

manejados remotamente (ver figura 1.1.1-1). De hecho la mayoría del trabajo operacional en una red digital (arriba del 85%) puede ser ejecutado remotamente. Las administraciones de esta manera han proporcionado una gran libertad en organizar la operación, especialmente en funciones tales como la conexión de subscriptores, y la inserción y modificación de los datos y parámetros que controlan la red y sus servicios.

Para una compañía de telecomunicaciones los problemas que se presentan en administrar estas, además de utilizar diferentes tecnologías se podrán suponer un sin número de dilemas; sin embargo el objeto del presente trabajo solo incluye aquellos que se presentan en el campo de la telefonía publica en especial con la interacción con las redes de telecomunicaciones, o mas en particular con una parte de ella que son las centrales telefónicas, por lo que de ahora en adelante, los conceptos que se manejen en el presente son aquellos que vengan relacionados directamente con estos.

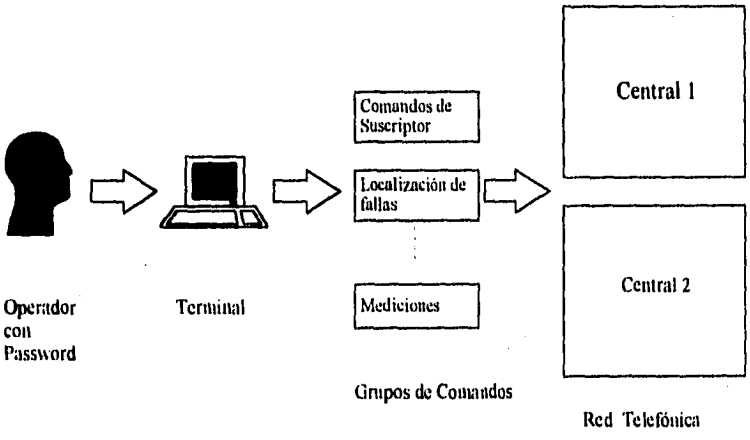


Fig. 1.1.1-1 Manejo Remoto de las Centrales.

A pesar de existir fabricantes, se mencionarán únicamente dos de ellos, analizando primero la central para después entrar al ambiente de comunicaciones de cada uno de estos.

1.1.2. Centrales EXA10.

Definición de EXA10.

El sistema de conmutación telefónica EXA10 es una versión totalmente digital que fue diseñado para la optimización del sistema en su manejo. Lo anterior se logra mediante la modularidad funcional del sistema EXA10.

El manejo acertado de un sistema de control por programa almacenado, requiere además del uso de varios sistemas de software que sirvan de apoyo.

Estos auxiliares del manejo (en el diseño, producción, instalación y todas las fases de la operación) deberán considerarse como parte integral del sistema EXA10.

Una parte presenta el sistema administrativo MOA empleado en el mantenimiento y la operación centralizados, así como algunos ejemplos de como puede organizarse una red administrativa que tenga varios centros operativos. La centralización no solo se refiere al mantenimiento del equipo de la central, sino también a aquellas funciones "en línea" tales como la supervisión y la reparación fuera de la planta, la administración de tráfico, la evaluación de la calidad del servicio y la facturación. Los centros que están "fuera de línea" son, por ejemplo, los almacenes de parte de repuesto, la reparación del hardware, la ingeniería del software y la capacitación.

A pesar de existir fabricantes, se mencionarán únicamente dos de ellos, analizando primero la central para después entrar al ambiente de comunicaciones de cada uno de estos.

1.1.2. Centrales EXA10.

Definición de EXA10.

El sistema de conmutación telefónica EXA10 es una versión totalmente digital que fue diseñado para la optimización del sistema en su manejo. Lo anterior se logra mediante la modularidad funcional del sistema EXA10.

El manejo acertado de un sistema de control por programa almacenado, requiere además del uso de varios sistemas de software que sirvan de apoyo.

Estos auxiliares del manejo (en el diseño, producción, instalación y todas las fases de la operación) deberán considerarse como parte integral del sistema EXA10.

Una parte presenta el sistema administrativo MOA empleado en el mantenimiento y la operación centralizados, así como algunos ejemplos de como puede organizarse una red administrativa que tenga varios centros operativos. La centralización no solo se refiere al mantenimiento del equipo de la central, sino también a aquellas funciones "en línea" tales como la supervisión y la reparación fuera de la planta, la administración de tráfico, la evaluación de la calidad del servicio y la facturación. Los centros que están "fuera de línea" son, por ejemplo, los almacenes de parte de repuesto, la reparación del hardware, la ingeniería del software y la capacitación.

El sistema EXA10 ofrece un máximo de flexibilidad en estas aplicaciones. Flexibilidad para adaptarse y colaborar en un medio ambiente constituido por la transmisión, la señalización, los servicios al abonado y administrativos, todos los cuales estan cambiando continuamente. Las adiciones y modificaciones al sistema se introducen en forma fácil y segura, incluyendo aquellos detalles y equipos que en el presente aún hoy hayan sido definidos.

Características de EXA10.

EXA10 es un sistema de comunicación telefónica que emplea el control mediante programa almacenado. el sistema está diseñado para su utilización como central local, de tránsito y combinada. Para llevar a cabo una central local, se puede utilizar la conmutación distribuida, mediante unidades de abonado remotas (concentradores). El sistema está constituido en su totalidad por grupos de tarjetas de circuito impreso. Las tarjetas del PC (Procesador Central) se colocan en estantes o almacenes. El almacén constituye la unidad básica del sistema para el manejo del hardware; todas las conexiones con otros almacenes y con MDF se realizan mediante cables enchufables. Los almacenes tienen diversos tamaños y se colocan en un bastidor.

El sistema de control es un sistema de procesamiento de datos con dos niveles y tiene una lógica parcialmente distribuida. Un nivel es el constituido por el procesador central y en el otro nivel, existen varios procesadores pequeños, los procesadores regionales, que estan duplicados. Todos los pares de procesadores centrales trabajan en modo síncrono paralelo, en cambio los procesadores regionales trabajan de acuerdo con el método de distribución de carga.

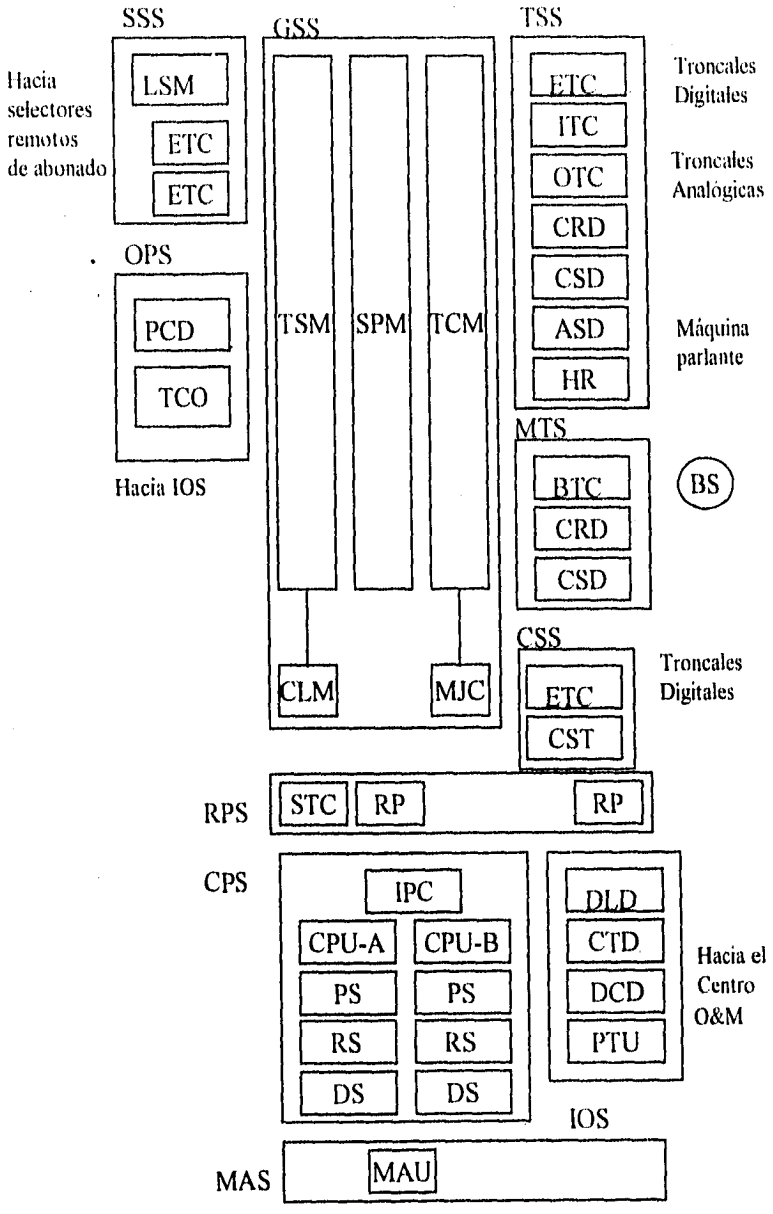


Fig.1.1.2.-1 Estructura Simplificada del Hardware de EXA10.

Mediante la descentralización del sistema de procesamiento de datos ha sido posible crear una estructura que, para las aplicaciones a escala pequeña, resulta económica al tiempo que satisface los requerimientos que impone una elevada capacidad máxima. Para una central local con un par de procesadores, el número máximo de líneas de abonado es de 40,000 considerando un tiempo promedio de ocupación de 100 segundos y un tráfico de 0.10 Erlangs/abonado (calculado para el caso donde se empleen 100% teléfonos con teclado, señalización MFC, etc.).

La figura 1.1.2.-1 es un diagrama a bloques en el que se muestra la estructura de hardware del sistema.

Podrán observarse algunos de los principales subsistemas:

SSS (Subscriber Switch Subsystem, Subsistema del paso de abonado), con un máximo de 16 unidades LSM (ó EM), cada una de las cuales conecta a 128 abonados.

GSS (Group Switch Subsystem, Subsistema de selector de grupo), es un selector digital tiempo-espacio-tiempo. Podrán observarse también el reloj triplicado CLM y el multienlace MJC. Se usa en las comunicaciones colectivas, en el ofrecimiento de troncal, etc.

TSS (Trunk and Signaling Subsystem, Subsistema de troncal y señalización), incluye a los circuitos para la conexión de los circuitos analógicos y digitales así como los dispositivos de señalización de selector de grupo. Los circuitos para las troncales analógicas estan agrupados en almacenes de 16 dispositivos y en los que se incluye la conversión analógica-digital. CSD y CRD se utilizan en la

señalización MFC. ST es la terminal de señal en el caso de señalización en canal común tipo CCITT No 7.

En la figura 1.1.2-1, se muestra CPS (Central Processing Subsytem, Subsistema de procesamiento central), formado por dos unidades de procesamiento de información y cada una de estas, constituida por un almacén de programas, un almacén de datos y un almacén de referencia.

En MAS (Maintenance Subsystem, Subsistema de mantenimiento), se puede observar la unidad de mantenimiento, MAU (Maintenance Unit), que supervisa la operación síncrona paralela de cada par de procesadores.

En IOS (Input/Output Subsystem, Subsistema de entrada - salida), se observan las interfaces de algunos dispositivos I/O típicos: pantalla, DLD; casetera, CTD; DCD permite la conexión entre el enlace de datos y un centro de operación y mantenimiento. PTU es la unidad de prueba del procesador (Processor Test Unit). La característica más notable del sistema EXA10 es su modularidad.

Ejemplo de ello, a nivel de subsistema, es suministro opcional en SSS y GSS de bloques de conmutación ya sea del tipo analógico o bien del tipo digital. Las interfaces del sistema están estandarizadas con lo que, a partir de un conjunto de subsistema que se dispone es posible obtener una gran cantidad de combinaciones para la planeación de las distintas centrales. De acuerdo con la notación:

SSS-A Subsistema de Selector de Abonado Analógico
SSS-D Subsistema de Selector de Abonado Digital
GSS-A Subsistema de Paso de Selección Analógico

GSS-D Subsistema de Paso de Selección Digital

Se tienen las siguientes combinaciones básicas:

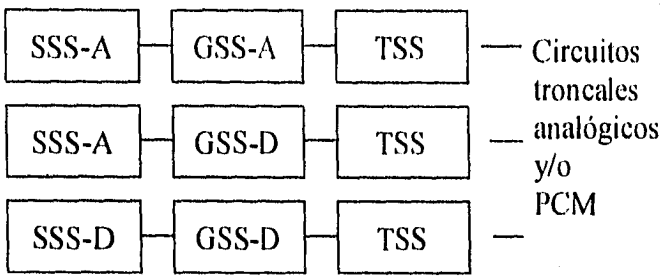


Fig. 1.1.2-2 Combinaciones Básicas de Subsistemas.

En los casos 2 y 3 pueden conectarse con GSS-D las unidades digitales remotas y los concentradores de abonado.

Como ya se mencionó la modularidad funcional se obtiene mediante una estructura de bloques funcionales. En su forma básica el sistema ofrece los servicios de abonado modernos que son normales, así como funciones para la operación y el mantenimiento. Mediante la adición de los módulos correspondientes, se obtiene una amplia gama de servicios adicionales. Se ha definido la frontera existente entre bloques funcionales y subsistemas con el objeto de limitar los factores específicos del mercado a un número reducido de bloques funcionales y subsistemas.

Lo anterior permite el uso de procedimientos sencillos al diseñar ampliaciones, cambios funcionales así como adaptaciones necesarias para satisfacer las diversas necesidades.

Estructura funcional del sistema EXA10.

El rápido desarrollo de la tecnología y la red telefónica impone requisitos muy específicos a un sistema de conmutación a fin de introducir el sistema en forma sencilla en ambientes diversos y cambiantes, la modularidad funcional constituye uno de los requisitos básicos.

De todo sistema SPC se espera una operación confiable. Es un hecho que los requisitos de confiabilidad deberán considerarse no solo para el hardware sino también para el software. Debido a esto la confiabilidad basada en el método de redundancia de hardware se complementa con requisitos para la seguridad en el software.

La capacidad modular es el último de los requisitos básicos y sirve para adaptar el sistema a un amplio margen de tamaños y condiciones de tráfico. El concepto de modularidad funcional se refleja claramente en la estructura funcional del sistema EXA10 que se ilustra en la figura 1.1.2-3. El sistema se construye mediante bloques de configuración agrupados en cuatro niveles jerárquicos.

Nivel funcional APT210.

La división del APT210 en subsistemas (ver figura 1.1.2.-4) obedece a las condiciones y requisitos impuestos para el manejo de tráfico, así como para las funciones de operación y mantenimiento.

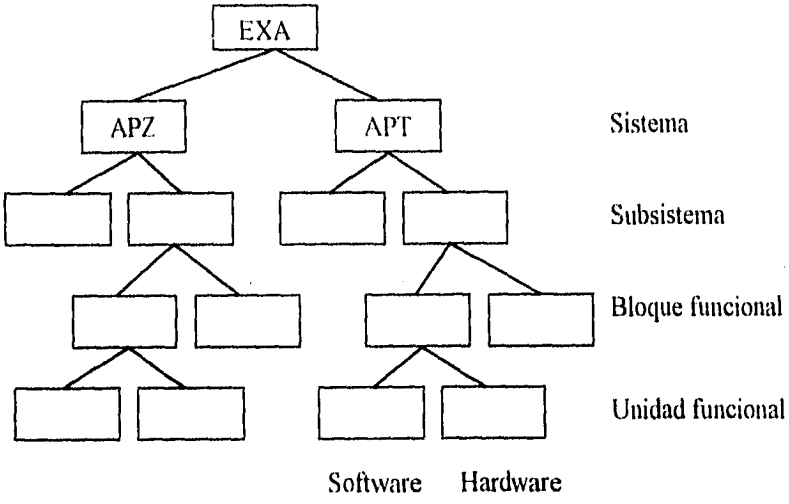


Fig. 1.1.2-3 Niveles Funcionales del Sistema EXA.

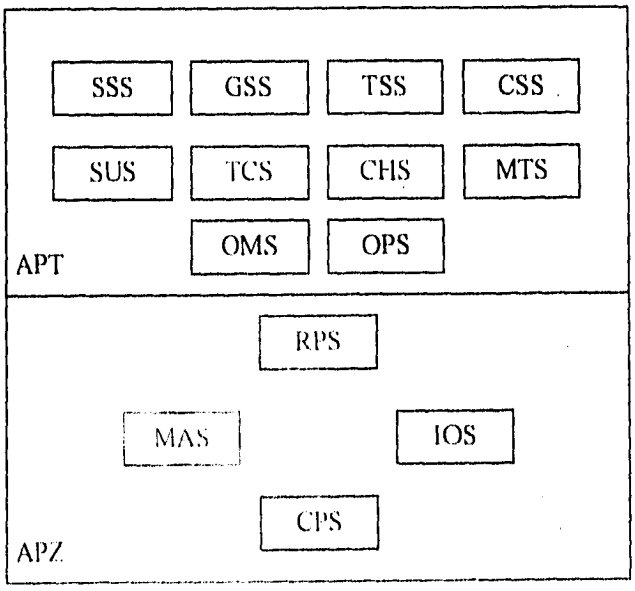


Fig. 1.1.2-4 División del APT210.

Los subsistemas anteriores se llevan a la práctica parcialmente en hardware y parcialmente en software. Las funciones de software se dividen en dos: funciones sencillas, rutinarias, con requerimientos de alta capacidad, y por otro lado, las funciones complejas. Con el fin de alcanzar un alto nivel de optimización en los requisitos para gran capacidad y bajo costo, las funciones complejas se ejecutan en forma centralizada por un subsistema de procesamiento central; las funciones rutinarias que demandan una alta capacidad se ejecutan en una etapa de procesamiento que está dentro del subsistema de procesadores regionales, el cual es un subsistema constituido por varios procesadores pequeños y sencillos. Así pues, el software del APT se divide en software regional y software central.

"APT"
SSS Subsistema de selectores de abonado
GSS Subsistema de selectores de Grupo
TSS Subsistema de Troncal y señalización
CSS Subsistema de Señalización en canal común
SUS Subsistema de servicios al abonado
TCS Subsistema de control de tráfico
CHS Subsistema de Tasación
MTS Subsistema de telefonía móvil
OMS Subsistema de Operación y Mantenimiento.
OPS Subsistema de operador

"APZ"
RPS Subsistema de procesador regional
CPS Subsistema de procesador central

IOS Subsistema de Entrada-Salida

MAS Subsistema de mantenimiento

En la figura 1.1.2.-5 se muestra la distribución del hardware así como del software regional y central correspondientes a los diversos subsistemas que están en APT.

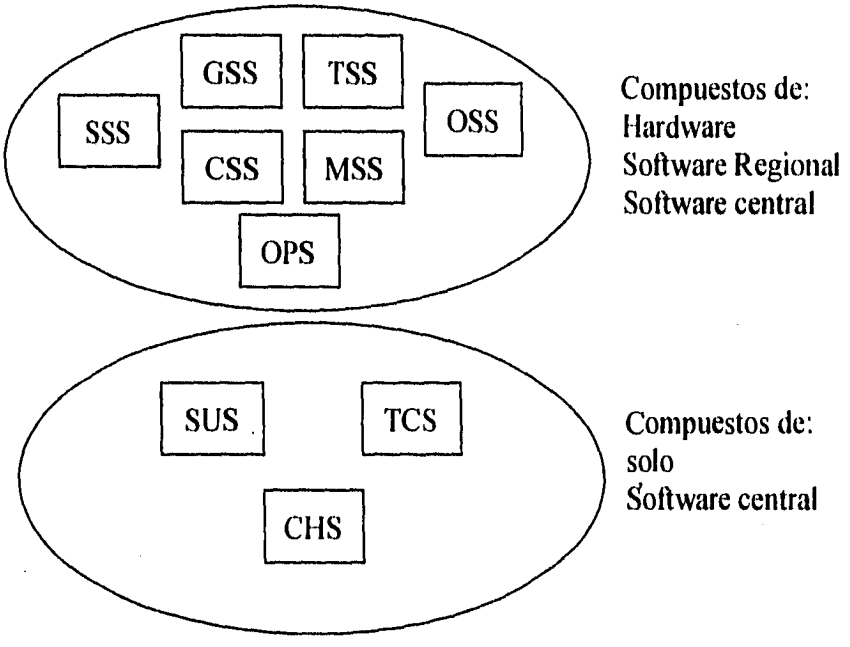


Fig. 1.1.2-5 Distribución de Hardware del ATP.

1.1.3 Sistema MOA.

Definición del MOA.

El MOA es un sistema de manejo de red basado en computadora, diseñado para alcanzar una desempeño universal, para las necesidades del manejo de telecomunicaciones.

Desde el punto de vista básico, las funciones del MOA son tales como ligar conexiones entre las facilidades de la red y el personal operador de la red.

El MOA como sistema puede ser añadido modularmente según las necesidades, en casos donde el equipo monitoreado no contenga la "inteligencia" requerida, el equipo MOA remoto puede ser añadido, en donde las capacidades especiales son necesarias en el lado de la organización, por ejemplo en la presentación gráfica o manejo de formas, también puede ser añadido al sistema. (Ver figura 1.1.3-1).

El MOA asegura que cualquier sistema de su misma familia pueda ser ensamblado con cualquier conjunto de capacidades de MOA, dentro del gabinete del sistema; y de esta forma asegurar que sistemas MOA futuros, basados en tecnologías nuevas, puedan ser compatibles con sistemas existentes.

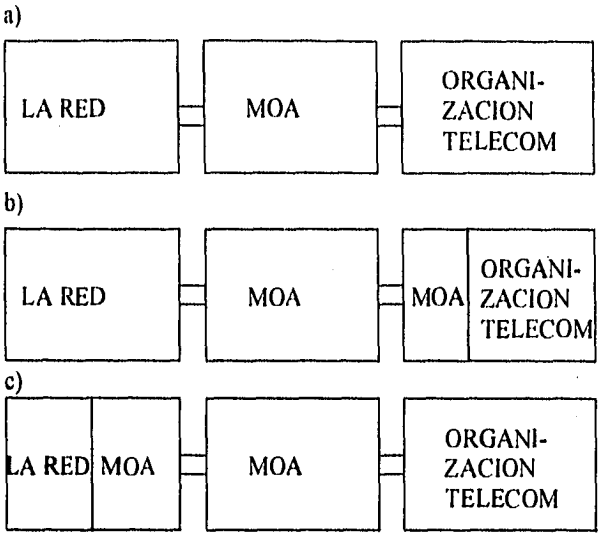


Fig. 1.1.3-1 Capacidades Especiales del MOA.

Dentro del concepto del MOA, hay un rango completo de funciones y productos disponibles, que cubren requerimientos de una administración de telecomunicaciones en manejo de redes. Cualquier instalación dada puede ser configurada para la supervisión y control de red, mantenimiento centralizado, colección de estadísticas de tráfico, cuentas de peaje y locales, equipo y funciones de bases de datos de subscriptores, etc. Todas las funciones pueden ser adaptadas en el mismo complejo computacional, o si se prefiere, en sistemas de computadoras separadas, pero con una tecnología y manejo común.

El MOA maneja áreas importantes de operación de una red como lo son:

- Supervisión
- Planeación

- Control
- Operación
- Mantenimiento
- Cargos y cuentas

Operación remota del MOA.

Las características que tiene la operación remota son las siguientes:

1) La información concerniente a la operación y el mantenimiento de una red puede ser transportada por varios medios de telecomunicaciones, ya sea por modems, fibra óptica, radiofrecuencias, internet. (Ver figura 1.1.3-2).

2) Es posible escribir información en las centrales por medios computacionales, tales como PC's remotas o terminales TTY.

3) Existen sistemas de seguridad para controlar el acceso remoto a los dispositivos de la red de telecomunicaciones, así como a sus funciones.

4) Es posible accesar cualquier equipo de red desde cualquier estación arbitraria de trabajo, provista con acceso autorizado.

5) Tiene operación en tiempo real para todas las tareas.

6) Tiene una interface hombre-máquina amigable.

7) Rápida colección y distribución de datos generados por la red.

8) El transporte de datos es seguro.

9) El trabajo en la operación de un supervisor de la red se reduce únicamente a dar mantenimiento a la base de datos, ligar información y montaje de cintas magnéticas.

10)Provee una solución integral para la operación de redes digitales y analógicas.

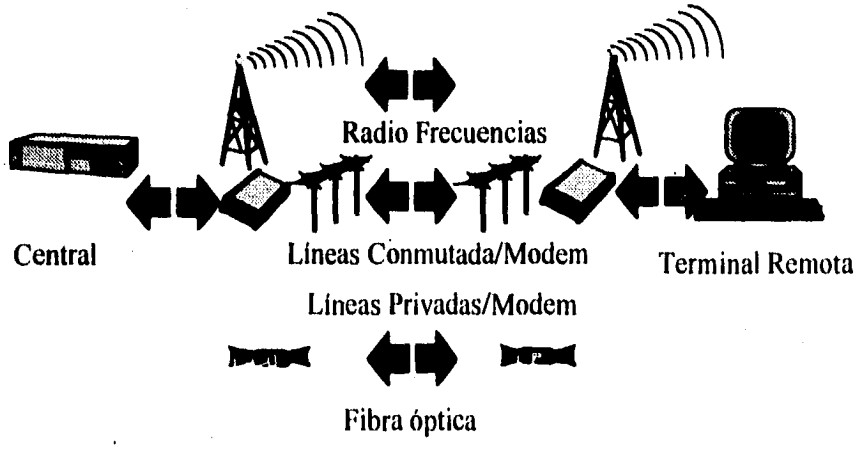


Fig. 1.1.3-2 Medios de Transporte de Unidad Remota.

Estaciones de Trabajo Especializadas.

- Con la filosofía de MOA la administración de las telecomunicaciones pueden ser organizadas en grupos con diferentes tareas (por ejemplo: operaciones orientadas a subscriptores, control de tráfico, administración de switches, etc.).

Otro criterio puede ser el dividir la operación por tareas dentro de grupos dedicados como:

- Tipos de equipo (transmisión, tipo de switches, potencia, etc.).
- Area Geográfica (área de mantenimiento, área de servicio, etc.).

Cambios de datos de los subscriptores.

Ejemplos de actividades son la conexión y desconexión de líneas de subscriptores y cambios de funciones de los subscriptores.

Las categorías de subscriptores usuales son las siguientes:

- Subscriptores de Teléfono regular.
- Subscriptores de Alcancías.
- Subscriptores de PABX.
- Subscriptores de Telex.
- Subscriptores con circuitos privados.

Cambios de datos de los switches.

Las actividades típicas que se incluyen en esta categoría son: cambios de rutas, cambios de dispositivos de entrada/salida, alteraciones en el tamaño de switches, cambios funcionales de switches, colección de estadísticas, y colección de resultados de medición.

Las funciones de soporte especializadas en el MOA, tales como archivos de comandos y bases de datos grabadas en circuitos, reducen tiempos significativamente.

Mantenimiento de Switches.

Mantenimiento de Switches incluyen actividades tales como la Inspección y pruebas, recepción y evaluación de alarmas, localización de fallas y reparaciones. Inspección, así como ciertas pruebas y reparaciones, deben ser ejecutados en sitio, pero las otras actividades pueden ser ejecutadas remotamente.

Las funciones de soporte del MOA tales como registro de fallas automáticas, trabaja en un orden manejable, y el MOA genera sumarios y reportes estadísticos.

Mantenimiento de líneas de subscriptores.

El mantenimiento de las líneas de los subscriptores está generalmente tomando actividades tales como test de rutina de líneas de subscriptores, recepción de alarmas y reporte de fallas, identificación y diagnóstico de fallas, y atención de ordenes de reparación.

Con un soporte para base de datos de subscriptores integrada en el MOA, es posible delinear el manejo de quejas de fallas y mediciones de líneas remotas, así como la creación y distribución de ordenes de reparación.

Las funciones del MOA pueden ser usadas para generar reportes estadísticos definidos.

Las estaciones de trabajo individuales pueden tener asignadas todas las tareas asociadas con este flujo de trabajo, o estaciones separadas pueden ser dedicadas cada una a pasos específicos en la tarea completa.

Equipo de una estación de trabajo.

Una estación de trabajo esta normalmente equipada con un display o terminal TTY y un conector telefónico. Las estaciones de trabajo dedicadas a ciertas tareas pueden requerir una impresora esclava conectada al display de la terminal. Como una alternativa a la impresora esclava, es posible permitir que un número de estaciones de trabajo compartan una impresora rápida. El ruteo de material a esta impresora puede ser decidido por cada operador, o por el supervisor de grupo. En algunos casos, la estación de trabajo puede ser equipada con una impresora de terminal. Esta alternativa es normalmente solo recomendada si va a ser limitado el uso de las estaciones de trabajo.

Funciones de las estaciones de trabajo.

Con las estaciones de trabajo localizadas en muchas oficinas de la compañía que lo opera y dedicada a muchas diferentes tareas, es extremadamente importante el tener un sistema de control de autoridad comprensible. El MOA habilita alojar tareas dedicadas a estaciones de trabajo específicas o alojar password a las tareas. Los passwords son dados por los operadores, quienes estén habilitados a usarlos por la firma, ganando el acceso únicamente a las tareas para las cuales hayan sido entrenados.

El password de acceso habilitará al operador a entrar al MOA, y por lo tanto direccionar y conectarse únicamente a los switches con los que el password está asociado. El password, también habilitará el uso asociado con el MOA y con los switches de comandos.

1.1.4. Central ISS2199.

Definición de ISS2199.

La central ISS2199 es un sistema completamente digital, que utiliza técnicas de optimización de transmisión y conmutación de datos, mediante Modulación de Pulsos Codificados (PCM) también puede dar transmisión de alta calidad, este sistema opera bajo control totalmente distribuido. La potencia de procesamiento y control es escalable, es tanta como sea posible en la práctica y pueden ser colocarlas en terminales, en lugar de concentrarse en una unidad central.

Gracias al advenimiento de microprocesadores poco costosos y sus memorias asociadas de bajo costo, hace posible ahora dividir modularmente el control a través del sistema.

En una configuración de control centralizado, si la función de control no opera, el sistema entero es inoperante. Cuando el control se distribuye en un gran número de módulos individuales, el mal funcionamiento en cualquier módulo solo tendrá un efecto marginal sobre el sistema total, sin reducir el servicio por debajo del límite adaptable.

El control totalmente distribuido y la modularidad ofrecen además otras ventajas:

- Mayor facilidad en la planeación de la red.
- Aprovechamiento de su capacidad.
- Errores significativos solo en software.
- Abatimiento de costos.
- Incrementar la capacidad de procesamiento en proporción a su nuevo tamaño y tráfico.

Una red de telecomunicaciones se considera a salvo de fallas si el funcionamiento indebido de cualquiera de sus elementos operativos no surte ningún efecto, o casi ninguno sobre el sistema global. Esto puede lograrse mediante el control distribuido. Con el control distribuido, en la mayoría de los casos un máximo de solo 60 líneas o 30 troncales, pueden dejar de dar servicio a causa de una sola falla en cualquier momento dado. (Ver figura 1.1.4-1).

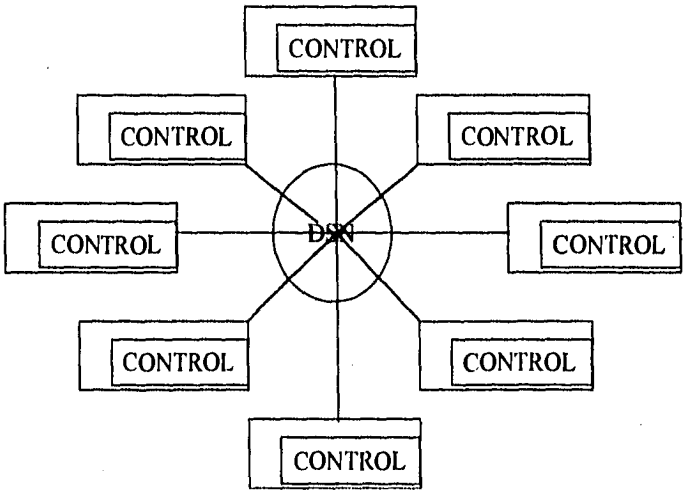


Fig. 1.1.4.-1 Control Totalmente Distribuido.

Para obtener una confiabilidad aún mayor, muchas unidades funcionales de la ISS2199 han sido duplicadas, reduciéndose así el riesgo de inhabilitación del sistema entero casi hasta cero.

Otros aspectos de este concepto de seguridad en cuanto a fallas, es la confiabilidad de las trayectorias mediante la Red Digital de Conmutación basada en un circuito integrado LSI.

Estos circuitos integrados contienen su propia lógica y memoria y permiten el establecimiento de rutas múltiples por toda la Red de Conmutación Digital.

La falla de cualquiera de los circuitos LSI significa simplemente que la trayectoria se le asigna a otra ruta, a través de otro circuito LSI, este circuito LSI (DSE) realiza las 3 funciones siguientes:

- Transmisión de voz y/o datos.
- Selección de Trayectorias.
- Comunicación entre microprocesadores distribuidos.

Uno de los aspectos más importantes del concepto de futuro asegurado de la ISS2199 es la habilidad en la central para mantenerse al día con los desarrollos más recientes tanto de hardware como de software. Cada módulo del hardware de la ITT1440 tiene su propio módulo de software, con una interface fija hacia el resto del sistema. Por tanto para introducir un nuevo módulo de hardware, solo es necesario cambiar su módulo asociado de software. (Ver figura 1.1.4-2).

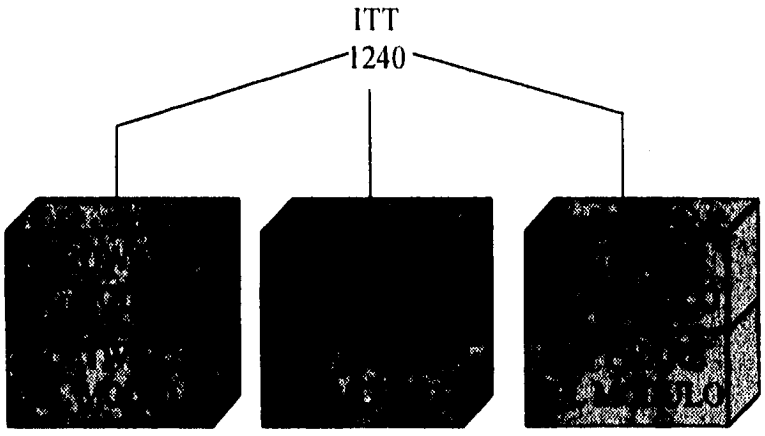


Fig 1.1.4-2 Capas de Software por Módulo.

El software está estructurado para que las reformas no requieran una recomprobación extensa del software ya existente.

La ITT 1440 abarca toda la gama de aplicaciones de controles locales, tandem y de cobro, desde la más pequeña unidad de abonados remotos hasta la central más grande local o de cobro con más de 100,000 líneas ó 60,000 troncales, todas ellas con los mismos módulos de hardware con la misma arquitectura de control distribuido. Además, el tipo más pequeño de central puede expandirse fácil hasta el tamaño más grande, usando los mismos módulos de hardware y software.

La ISS2199 comprende dos productos como sigue:

1) Unidad Digital ISS2199 de Abonados Remotos (RSU):

- PCM, de 32 o de 24 canales.
- Con 6 a 480 líneas.

- Con 1 ó 2 troncales digitales a la central digital matriz ISS2199.
- Con 8 puntos de multiconexión, para la distribución hasta de 480 líneas.

La unidad digital de Abonados Remotos (RSU) funciona con el fin de convertir a digitales las señales analógicas procedentes de los abonados, y para concentrar señales digitales que después se transmiten a una central matriz por conducto de uno o dos enlaces de transmisión PCM.

2) Central Digital ISS2199.

- Local, tándem, Local/Tándem, tarificación, Internacional.
- PCM de 32 o de 24 canales.
- 60 o más de 100,000 líneas
- 120 o mas de 60,000 troncales.
- Más de 25,000 Erlangs de conmutación.
- Más de 750,000 intentos de llamada en horas de máximo tráfico (BHCAs).

Adaptabilidad de ISS2199 a los diferentes sistemas.

Las administraciones telefónicas de todo el mundo se comprometieron y han hecho realidad en la implementación de la IDN para el tráfico de voz. El hardware digital de la IDN, que usa la tecnología de semiconductores, ofrece ventajas muy claras de costo y funcionamiento, respecto del equipo tradicional.

Conforme las centrales digitales vayan sustituyendo cada vez más a las centrales analógicas, serán menos las conversiones de analógicas o digital que resulten indeseables y costosas. El progreso de la IDN trae dos características cruciales para los planes a largo plazo de una administración.

IDN significa canales de comunicación de 64 Kbits/seg. por toda la red, en comparación con un ancho de banda estándar de 3 KHz en las redes analógicas.

Eventualmente, la red digital extenderá la operación digital de bucle de abonado, y este es su primer paso hacia la IDN. La CCITT está estudiando un bucle de abonado de 80 Kbits/seg. que consta de 3 canales:

Uno de 64 Kbits/seg. para voz o datos de alta velocidad.
Uno de 8 Kbits/seg. para señalización con la central, alarmas y telemetría.
Uno independiente de 8 Kbits/seg. para datos de baja velocidad.

[redacted]

[redacted]

[redacted]

1.1.5. Sistema 2199.

Distribución del Sistema 2199.

La implementación del control completamente distribuido implica cierto número de nuevos conceptos que permiten a un gran número de microprocesadores ubicados por toda la central el control tanto de un grupo pequeño de dispositivos terminales como el acceso a la red asociada de dispositivos terminales sin la ayuda de ningún otro procesador.

Los conceptos más importantes son una red de conmutación que puede controlarse desde sus puntos externos sin necesitar un control central para establecer y mantener trayectorias y una estructura de software que permite a cierto número de microprocesadores autónomos cooperar en el manejo de todas las funciones de la central.

Esta estructura de software requiere que cada microprocesador pueda establecer trayectorias en la red de conmutación, tanto para conexiones de terminal como para pasar información de control a otros microprocesadores.

Puesto que no se proporciona ninguna facilidad por separado para la comunicación de datos entre microprocesadores, estos tienen que depender exclusivamente del establecimiento de trayectorias a través de la red de conmutación para completar cualquier función de control deseado.

Es este aislamiento físico de cada microprocesador respecto a los demás lo que le permite a la estructura de control totalmente distribuido evolucionar

eventualmente hasta un microprocesador por línea, para soportar facilidades avanzadas de control de terminales.

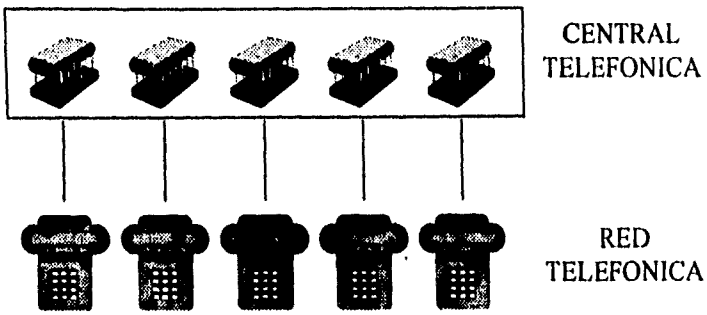


Figura 1.1.5.-1 Evolución a un Microprocesador por Línea.

la red de conmutación digital, o IDN, no solo sustituye a la red de conmutación convencional de control centralizado, sino también a los complejos sistemas con bus de intercomunicación que requiere el control centralizado para comunicarse y controlar uno de los dispositivos individuales en las terminales.

Las funciones principales de la IDN son responder a comandos de los microprocesadores para establecer conexiones entre terminales de abonado o troncal, transmitir digitalmente voz y datos, y transmitir mensajes entre los microprocesadores; cada unidad funcional de la IDN contiene en sí misma toda la lógica necesaria para actuar como una unidad independiente.

La red de conmutación digital, se basa en un solo circuito LSI. Este circuito es llamado Elemento de Conmutación Digital (DSE).

La central Digital ISS2199 (ver figura 1.1.5.-2), tiene aspecto de una variedad de módulos conectados a una red central de conmutación digital. La IDN es el medio empleado para efectuar esta comunicación.

Gran parte de la lógica para la transmisión de voz y datos se aloja fuera de la DSN y dentro de los módulos.

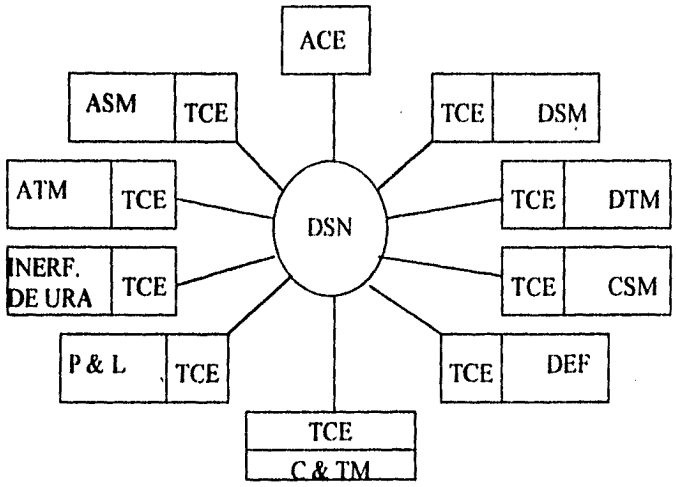


Fig. 1.1.5-2 Módulos de la Central 2199.

Cada módulo del sistema ISS2199 consta de dos partes: la Terminal y el elemento de control. Cada tipo de terminal realiza un servicio distinto; por ejemplo el manejo de líneas analógicas, líneas digitales, troncales analógicas, troncales digitales, etc.

En contraste, lo que hace el elemento de control de cada módulo es controlar la operación de la terminal y conectarla a la DSN. Esto se efectúa mediante

formatos de mensajes estandarizados; es decir, sin importar en que módulo se origina un mensaje, tendrá un formato estándar respecto de la DSN.

El elemento de control de cada módulo, consiste en un interface de terminal, microprocesador y una memoria. Dependiendo de cual módulo se trate, puede haber un volumen mayor o menor de memoria; en todos sus demás aspectos, todos los Elementos de Control de Terminal son exactamente iguales.

Elementos Conmutación Digital (DSE).

Es de particular interés el diseño de la red de conmutación digital o DSN (Digital Switching Network) (ver figura 1.1.5-3) usada en las centrales digitales ISS2199. La DSN se constituye con elementos de conmutación digital (o DSE) idénticos, cada uno de los cuales tiene 16 puertos compuestos por una entrada y una salida independiente de troncales PCM de 32 canales, llevando 16 bits por palabra. El DSE realiza conmutación especial entre puertos y conmutación temporal entre canales permitiendo a cada uno de los 512 (32x16) canales entrantes, conectarse a cualquiera de los 512 canales salientes.

Cada elemento de conmutación incluye circuitos de control y memorias para establecer y mantener una conexión entre todos los canales entrantes, y sus canales salientes asignados.

Para establecer una conexión, una palabra de comando es enviada por un procesador de origen a través de T1 a la DSN; el elemento de conmutación reconoce la palabra como un comando y establece una conexión entre el canal entrante en que se recibió el comando, y el canal saliente cuya dirección estaba adentro de la palabra de comando. Una conexión

establecida puede ser cancelada por otro comando del procesador el cual regresa los canales del elemento de conmutación a un estado desocupado, dejándolo disponible para nuevas conexiones. Además si un elemento de conmutación falla al reconocer un comando, se genera y envía al procesador de origen una señal de no-reconocimiento y un nuevo intento se realiza.

Elementos de Conmutación Digital

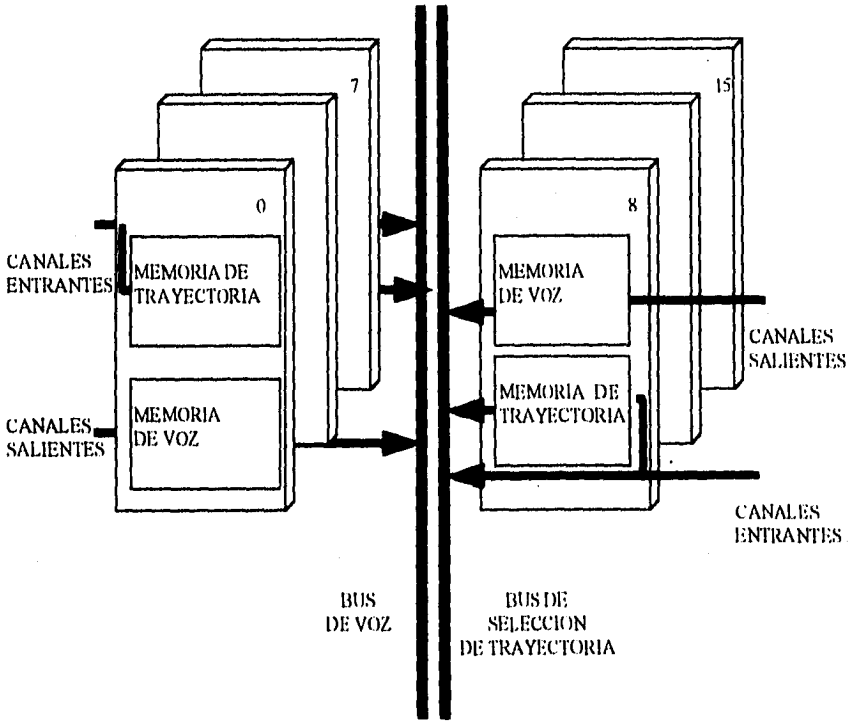


Fig. 1.1.5-3 Elementos de Conmutación Digital.

Red de conmutación digital.

Las funciones principales de la DSN (Red de Conmutación Digital)(ver figura 1.1.5 - 4), son:

a) Responder a los comandos de los microprocesadores para establecer conexiones entre terminales de abonado o troncal.

b) Transmitir digitalmente voz y datos.

c) Transmitir mensajes entre microprocesadores.

Aún cuando toda la DSN se construye con los mismos Elementos de Conmutación Digital (DSE), esta dividida en dos partes diferentes llamados:

1) Conmutador de grupo.

2) Conmutador de acceso (Access Switch).

El conmutador de grupo consiste hasta de 4 canales independientes llamados planos. El número de planos paralelos depende de las necesidades de tráfico, mientras mas planos haya, mas volumen de tráfico puede atender la DSN. Para todos los fines prácticos, 4 planos son suficientes para manejar cualquier requerimiento de tráfico.

Los conmutadores de acceso (AS) Access switch, son llamados así porque dan acceso a los planos del conmutador de grupo, es decir su función primaria de (AS) es distribuir el tráfico que entra a la DSN entre los planos del conmutador de grupo. El conmutador de acceso (AS) aporta los puntos de conexión entre el conmutador de grupo y los diversos módulos de la ISS2199.

Los Conmutadores de Acceso y de Grupo se componen exactamente del mismo Elemento de Conmutación Digital (DSE), la única diferencia estriba en las funciones que desempeñan; por eso es que los conmutadores de acceso (AS)

Red de Conmutación Digital

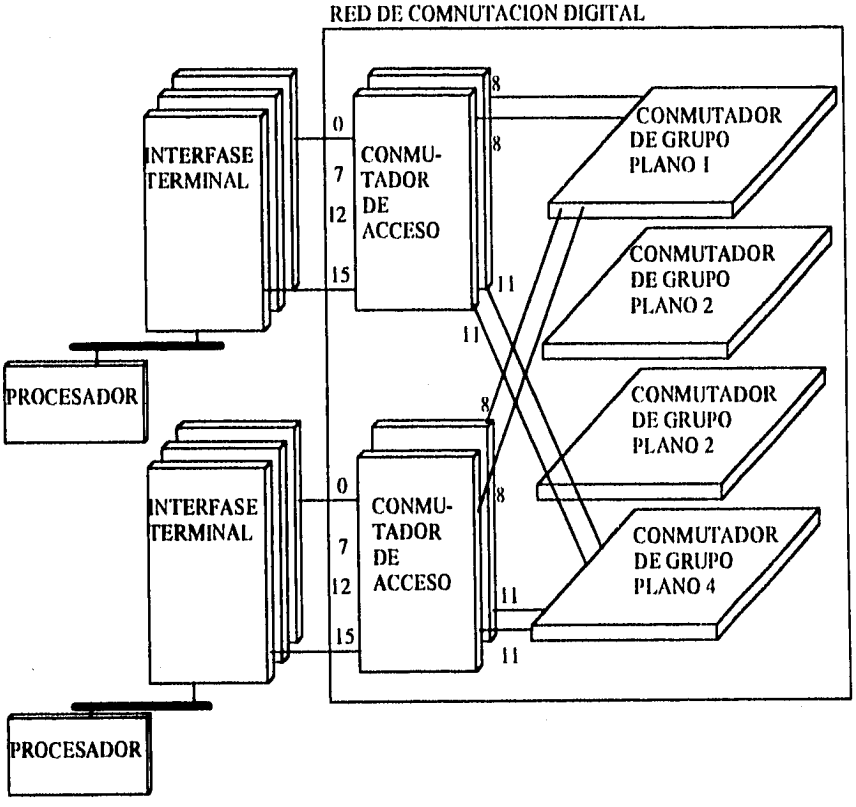


Fig. 1.1.5-4 Red de Conmutación Digital.

siempre se instalan por pares a fin de proporcionar trayectorias duplicadas al Conmutador de Grupo, y a eso se debe también que los (DSE) de conmulador de grupo estén configurados en planos para manejar distintos requerimientos de tráfico.

Elemento de Contol Terminal (TCE).

Cada módulo del sistema ISS2199 consta de 2 partes: la terminal y el elemento de control.

Cada tipo de terminal realiza un servicio distinto, ya sea el manejo de líneas analógicas, líneas digitales, troncales analógicas, troncales digitales, etc.

La función del elemento de control de cada módulo (TCE) (ver figura 1.1.5-5), es controlar la operación de la terminal y conectarla a la DSN.

Esto se efectúa mediante formatos de mensajes estandarizados; o sea, sin importar en que módulo se origina un mensaje, tendrá un formato estandarizado respecto a la DSN.

El elemento de control de cada módulo consiste de una interface de terminal (TERI), un microprocesador (B. Pro) y una memoria (B. Memory). Dependiendo de cual módulo se trate, puede haber un volumen mayor o menor de memoria; en todos sus demás aspectos, todos los elementos de control de terminal (TCE) son exactamente iguales.

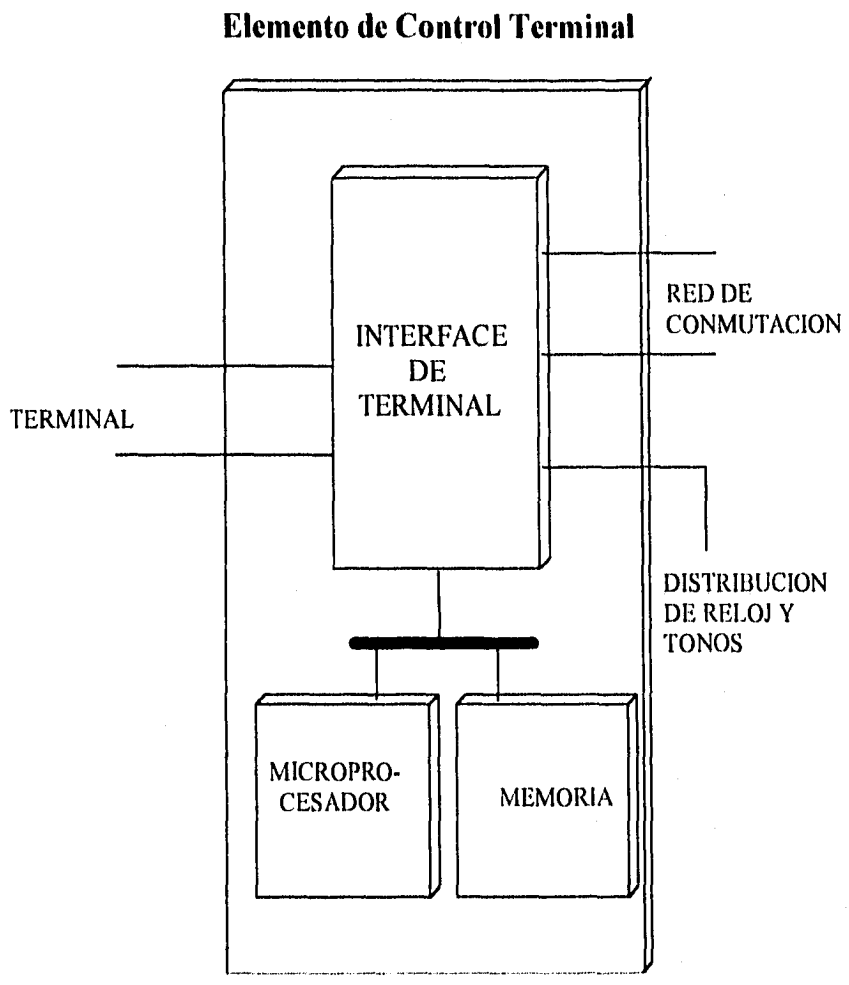


Fig. 1.1.5-5 Elemento de Control Terminal.

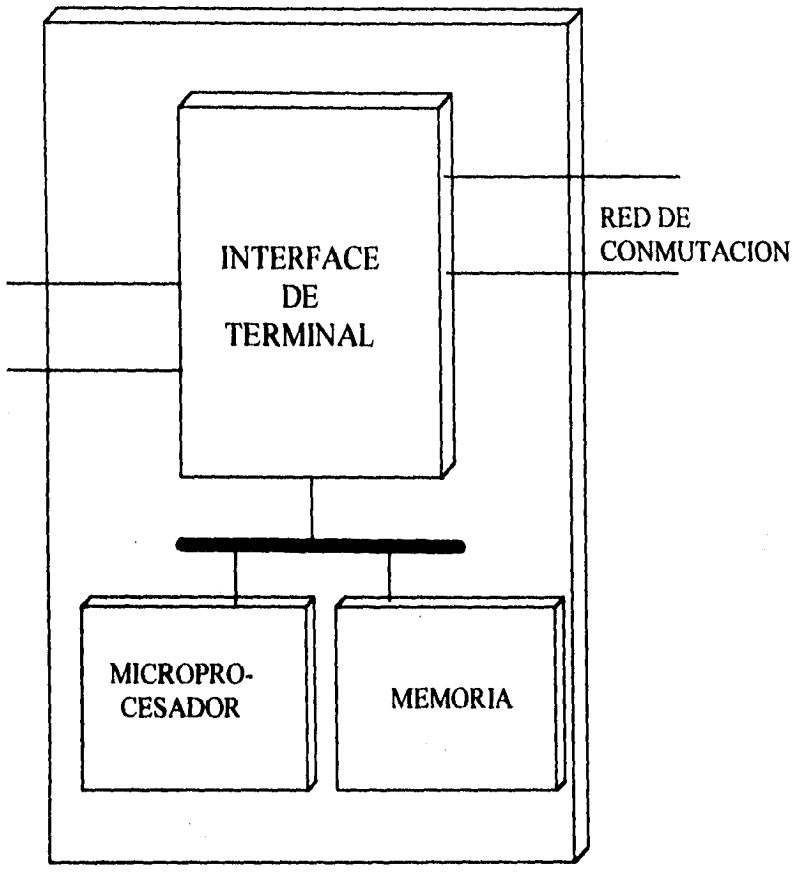


Fig. 1.1.5-6 Elemento de Control Auxiliar.

Elemento de Control Auxiliar (ACE).

El elemento de control auxiliar (ver figura 1.1.5-6) aporta niveles adicionales de control distribuido, por medio de un microprocesador , su memoria asociada y una interface de terminal. A cada uno se le asigna una tarea, como por

ejemplo procesamiento de llamadas, traducciones, reserva, etc. , cargándose con el software adecuado.

Se conectan con la DSN y se comunican a través de ella exactamente de la misma manera que todos los demás elementos de control, pero no se hayan conectados a ninguna terminal. El número de estos procesadores en las centrales grandes es aproximadamente de 1 por cada 480 "líneas equivalentes" en las que una troncal cuenta como 4 líneas equivalentes.

Procesador "B".

La tarjeta del procesador B (BPRO) forma parte de un conjunto de una hasta 4 tarjetas de memoria externa BMEM en la parte procesadora del elemento de control de la central ISS2199.

El procesador puede ser usado en el elemento de control terminal (TCE) ó en el elemento de control auxiliar (ACE) como sea necesario. En el TCE el BPRO esta involucrado en los procesos terminales en el sistema, en este modo el BPRO es usado con una tarjeta de memoria sencilla. En el ACE el BPRO realiza procesos no terminales y al realizar estos procesos cada ACE usa desde una hasta 4 tarjetas de memoria.

La tablilla de la interface terminal (TERI) es también una parte de cada elemento de control. Esta tarjeta TERI proporciona la interface entre procesador y terminales, módulos no terminales especiales, y la red de comunicación digital.

La tarjeta BPRO tiene los siguientes elementos del circuito principal:

- Generador de reloj (8284A).
- Dispositivo microprocesador (8086).
- Controlador de Bus (8288).
- Controlador de interrupciones programable (8259A).
- Temporizador de intervalos programable (8253-5).
- Registro de control del procesador para banderas de errores y comandos (2Kx8).
- Protección de escritura RAM 1Kx1.
- PROM de 8K x 16 para programa.
- Circuitería para temporización de bus de control de terminal.
- 2x8 de PROM para decodificación de direcciones.
- Buffer para el bus de control de terminal (TC).

Memoria B.

La tarjeta de memoria tiene una capacidad de almacenamiento de 128K por 22 bits-palabras, organizados en 2 bancos de 64 K palabras c/u. 16 Bits de cada palabra son bits de datos (en dos bytes) y los restantes 6 son los bits del código Hamming de comprobación de error los cuales permiten la detección y corrección de error sencillo y únicamente la detección de error doble. Estos bits pueden ser inhibidos por el procesador.

El direccionamiento de la memoria es sobre un bus de 3 estados de 18 bits. Las señales de control permiten un acceso a escritura o lectura de una palabra de 16 Bits, del Byte alto de la palabra (Bits 8 a 15) ó bien del byte bajo de la palabra (bits 0 a 7), de acuerdo al modo de acceso empleado.

Los datos son transferidos entre el procesador y la tarjeta de memoria sobre un bus bidireccional tres estados de 16 bits.

Un ciclo de renovación de la memoria es automáticamente insertado entre los accesos del procesador, este ciclo llega cada 15 segundos teniendo prioridad al no ser interrumpidos por un acceso del procesador. Si un acceso del procesador llega a la memoria al mismo tiempo de un ciclo de renovación, un mediador resuelve el problema.

La tarjeta de MEM tiene circuitos de acceso para funciones de prueba.

Interface terminal.

La interface de terminal, proporciona la forma de establecer y mantener trayectorias a través de la red de conmutación. También aporta una etapa de conmutación entre los circuitos terminales de la ISS2199 y el conmutador de acceso, y una función de almacenamiento provisional para los mensajes interprocesadores.

La interface de terminal con la red de conmutación contiene 4 enlaces unidireccionales para voz y datos, terminados en 2 puertos de recepción y 2 puertos de transmisión, la interface de tonos contiene un enlace unidireccional que suministra los tonos a la interface de terminal. Cada enlace lleva los datos en el formato normal de 32 canales de palabras de 16 bits, con el canal 0 y el canal 16 usados para sincronización y señalización. En el caso del enlace para distribución de tonos el canal 16 llevará información de tonos por no ser necesaria la señalización.

Cuando la interface de terminal, esta asociado con un procesador de terminal, se usará la interface con los circuitos terminal, esta interface esta organizada en una forma similar a la de la interface con la red de conmutación descrito anteriormente.

La interface de terminal agrega tonos a la muestras de voz, cuando esto se requiere utilizando las señales recibidas en el puesto de distribución de tonos.

Hay una etapa completa de conmutación entre los puertos de la red de conmutación y las de los circuitos terminales. El interface terminal se concentrará a dos distintos elementos del conmutador de acceso. Un defecto en uno de sus elementos reducirá el número de canales disponible en un 50%, pero los restantes permanecen en operación.

La comunicación entre una interface de terminal y su procesador asociado se efectúa por conducto de 3 buses:

Un bus de control, un bus bidireccional de datos y bus de direccionamiento. Este último se usa para especificar cual localidad de memoria de la interface de terminal será la afectada por una transferencia de datos.

El almacenamiento en la interface de terminal se divide en 2 secciones mayores, siendo una la RAM de paquetes y la otra la RAM de puertos y canales.

Estructura de Software 2199.

Conceptos generales del software.

La estructura del software en módulos discretos permite una flexibilidad bastante considerable en el diseño y la aplicación de la arquitectura de control distribuido de la ITT 2199. Los módulos de software pueden asignarse de acuerdo con el total de recursos de procesamiento disponibles, puede reasignarse en caso de expansión de la central o de que se le agreguen características, y se reasignen automáticamente cuando ocurren fallas de procesador.

La división completa y el uso de lenguaje de alto nivel ofrecen la ventaja de pequeños programas independientes segmentados que son fáciles de diseñar, codificar, probar y mantener. Estos puntos clave dan como resultado un software total que es fácil de entender y es fácilmente modificado para satisfacer nuevas demandas por parte de los clientes o para aprovechar al máximo la tecnología involucrada.

La estructura del software se basa en una división estricta, ordenada como cierto número de niveles funcionales. La estructura funcional del software discreto es una jerarquía. Cada nivel abarca un aspecto progresivamente más detallado del requerimiento total de la central. El nivel mas bajo se asocia directamente con el hardware, mientras que los niveles más altos llevan al cabo funciones telefónicas particulares dentro de la central.

Sistema operativo de los elementos de control.

El sistema operativo de los elementos de control está compuesto por el sistema operativo propiamente dicho y el sistema de control de la base de datos.

Sistema operativo.

Cada elemento de control de la ISS2199 tiene un núcleo de sistema operativo residente. Esto, junto con el hardware del sistema, constituye un procesador virtual para el resto del software.

Las tareas principales de este núcleo, son:

- Programación de los procesos del software.
- Transferencia de los mensajes entre procesos.
- Manejo de la interface de terminal y la red de comunicación digital; es decir, del control de las trayectorias, la comunicación de paquetes y las funciones de mantenimiento de soporte para la DSN y la interface de terminal.
- Proporciona facilidades de sincronía y hora del día.
- Manejo de interruptores, carga de programas no residentes, soporte para la prueba y acciones de recuperación.

Hay otros dos componentes del sistema operativo usados en los elementos de control de periféricos de computadora.

- Manejadores de entrada y salida.
- Cargadores.

Los manejadores de I/O soportan la interface para los periféricos de computadora y la facilidad de comunicación hombre-máquina. El software de arranque se requiere para cargar los diferentes elementos de control con su software específico. Cada elemento de control tiene un programa local de "BOOTSTRAP" en memoria de sola lectura (ROM) el cual es ejecutado cada vez que el elemento requiere cargarse desde el disco o la cinta magnética. Los programas cargadores también se usan para traer programas no residentes a la memoria del procesador.

Debido a que la ISS2199 es un sistema de tiempo real, muchas de las mismas funciones tienen que realizarse simultaneamente para cierto número de usuarios. Por ejemplo, una sola FMM de manejo de llamadas puede requerirse para manejar llamadas múltiples que se translapan en el tiempo. Cada uso que se hace la FMM es designado como "un proceso".

Sistema de control de base de datos.

Los diversos procesadores en la central requieren información relativa a los recursos disponibles del sistema, el plan de numeración de la central, las clases de abonado y los esquemas de señalización, junto con toda la información que tradicionalmente se considera como datos de la central.

En cuanto a las funciones del software la base de datos está distribuida entre los procesadores del sistema. Por ejemplo, los datos que caracterizan la clase de línea (línea de abonado de PABX, etc.) se localizan en el TCE del abonado, mientras que los datos que describen la clase de servicio (aparato de botonera, servicio restringido, etc.) residen en el ACE de control de llamadas asociado con el TCE del abonado.

El sistema de control de la base de datos rescata datos y opera con ellos sin importar cual sea su ubicación, tipos de memoria, ordenes de acceso o forma en que se almacenan los datos.

En la memoria se proporciona almacenamiento de respaldo para todos los datos no dinámicos de la base.

La integridad de la base de datos la salvaguardan programas que realizan verificaciones de validez para todas las actualizaciones solicitadas por abonado o por el personal de la central, ya sean mediante auditorías regulares o cotejando los datos en el procesador con el contenido de la memoria.

Manejadores de dispositivos telefónicos.

La función básica del manejador de dispositivos telefónicos es la de traslación entre señales físicas (eléctricas) en el nivel de hardware y las señales lógicas en el nivel de usuario. Todo acceso al y del dispositivo telefónico, tanto para el manejo de llamadas (lógica de señalización) como para el mantenimiento, se efectúa a través del manejador de dispositivos. Se proporciona un manejador de dispositivos especializados para cada tipo de hardware de terminal; es decir de línea de abonado analógico, troncal digital, emisor, receptor de MF, etc.

Cada manejador de dispositivos, junto con circuitería asociada, aparece como un "dispositivo virtual" para el resto del software del sistema. La ventaja de este enfoque es que los detalles de la implementación del hardware de las terminales son independientes de la lógica de señalización. De esta manera el efecto del cambio en el hardware de una terminal se limita al software del manejador del dispositivo.

Los manejadores específicos de dispositivos telefónicos, residen siempre en los TCEs conectados a las terminales que controlan.

Manejo de llamadas.

Hay cuatro componentes mayores en el software del manejo de llamadas: señalización, control de llamadas, servicio de llamadas, y tarificación. Estos componentes se subdividen en máquinas de mensajes finitos (FMMs) y son cargados en uno o varios elementos de control durante la inicialización del sistema.

Las funciones primarias proporcionadas por señalización son de darle un significado telefónico a todo evento recibido del manejador de dispositivos telefónicos y la de transformar las órdenes telefónicas procedentes del control de llamadas en la debida secuencia de eventos que ha de transmitirse por conducto de la terminal.

Las funciones de señalización dependen de la señalización de línea y de registro, y cada función se implementa en una FMM. Esta partición hace posible las modificaciones sencillas y económicas para aceptar nuevos sistemas de señalización en lo futuro.

El software de control de llamadas aporta el control de alto nivel para el tratamiento de llamadas básicas, así como otras facilidades para llamadas mas complejas. La lógica del control de llamadas se divide entre FMMs que implementan las diferentes fases de una llamada; es decir, preselección, terminación de llamada, y liberación. Cada una de estas etapas puede a su

vez subdividirse en FMMs específicas de una facilidad que se halle activa durante esa fase de la llamada, por ejemplo, preselección de conferencia.

La interface entre señalización y el control de llamadas esta a nivel de lenguaje telefónico, utilizando términos como "toma", "contestación", "liberación", etc. Este lenguaje es independiente del sistema de señalización asociado con una terminal, así que la lógica de control de llamadas esta aislada de la lógica de señalización. El software de control de llamadas se ubica normalmente en los ACEs de control de llamadas.

Las FMMs de servicio de llamadas proveen a las FMMs de control de llamadas, por ejemplo, con información de enrutamiento basada en los dígitos recibidos y la clase de terminal, la identidad del abonado local y las identidades de los recursos telefónicos disponibles, tales como troncales emisores/receptores, etc. La mayoría de las FMMs de servicio de llamadas y sus relaciones de datos asociadas, estan ubicadas en los ACEs del sistema.

Las FMMs de tarificación manejan el control y la generación de los registros para la facturación de las llamadas, que puede ser ya sea en conjunto o detallada por abonado. Luego ya pueden vaciarse los registros en la memoria, para su procesamiento posterior y la facturación a abonado.

Software de mantenimiento y administración.

El software de mantenimiento asegura un alto grado de servicio, en presencia de defectos en el hardware y el software.

El software de mantenimiento explota la naturaleza distribuida del sistema, al distribuir todas las funciones de detección y de recuperación de emergencia, mientras mantiene su tamaño y su complejidad al mínimo. La redundancia adecuada en los elementos de control asegura que el nivel global de servicio se obtenga sin necesidad de duplicar todos los elementos de control.

Cada elemento de control contiene dos niveles de funciones autónomos de recuperación de manejadores de dispositivos, y la recuperación de fallas de procesadores. Los manejadores de dispositivos proporcionan también funciones de aislamiento del dispositivo, de inicialización y de soporte de pruebas, para el TCE de periféricos de computadora.

Los elementos de control tienen suficiente capacidad de verificación de datos para permitir la detección de errores y la inicialización automática.

El TCE periférico de computadora controla y ejecuta la mayoría de las acciones de mayor urgencia del mantenimiento a las fallas, las pruebas de enrutamiento y las auditorías. Además, este TCE está involucrado en la recarga del software, ya sea para un elemento de control o para la inicialización total del sistema.

La propagación de fallas se detiene de manera efectiva por verificaciones del protocolo de comunicaciones, mediante acciones efectivas autónomas en cada elemento de control y con auditorías que se efectúan independientemente en cada elemento de control.

Estructuralmente, el software de mantenimiento es altamente modular, cada FMM realiza una función específica.

El software de administración se reparte entre varias áreas principales, tales como mediciones de tráfico, administración en la red, comunicaciones hombre-máquina, facturación, extensiones, etc. El software de administración también es altamente modular; cada función es implementada en una FMM. La interface con el software del manejo de llamadas es por interface de alto nivel, lo que hace al software de administración enteramente independiente del software del sistema.

Implementación del software.

El software del la ISS2199 se caracteriza por su alta confiabilidad, su inmunidad a la falla total del sistema y su capacidad de aceptar con facilidad y de manera económica la incorporación de nuevos servicios y tecnologías. Con el fin de alcanzar estos objetivos, el grupo técnico encargado del software en la ITT ha aprovechado los métodos y conceptos de diseño más avanzados.

Hardware virtual.

Cada subsistema principal del hardware tiene una porción asociada de software que se conoce como "manejador de dispositivo" conjuntamente en el subsistema de hardware y el manejador de dispositivo se convierte en una forma de máquina virtual conocida con Hardware Virtual. Se escogió esta tecnología porque el manejador de dispositivo se comunica con los otros niveles de software a través de una interface que consiste en un bien definido juego de mensajes en ambas direcciones. Mientras se mantenga esta interface, sin importar que hace el subsistema de hardware ni como puede ser modificado, la comunicación con el resto del sistema será siempre la misma.

El concepto de hardware virtual aporta una o varias capas de software entre el hardware y la interface virtual, independizando así del resto del sistema la implementación específica del hardware.

Por ejemplo, el manejador de dispositivo transforma las ordenes de mensajes del software en señales para control del hardware y reporta los cambios en el estado eléctrico del hardware.

Este enfoque se aplica no solamente al hardware de telefonía, como por ejemplo los circuitos de línea y la Red de Comunicación Digital, sino también al procesador. De esta manera, puede incorporarse un nuevo tipo de procesador adaptando su manejador de dispositivo efectuando cambios menores en el sistema operativo y el software de mantenimiento, y modificando partes del compilador de lenguaje de alto nivel.

Máquina de mensajes finitos.

Una máquina de mensajes finitos (FMM) es una aplicación de la máquina de estado finito, un concepto que se estudia en la teoría de las máquinas secuenciales. Una FMM es un modulo de software dedicado a una función en particular (por ejemplo, señalización entrante de registro R2) la cual tiene las siguientes propiedades:

- Consiste en un grupo bien definido de estados y en las transiciones permitidas entre estos estados.
- Se define en cada estado un cierto número de mensajes de entrada permitidos y un cierto número de mensajes de salida permitidos.

- Para cada combinación específica de estados de FMM y mensaje de entrada recibido, la FMM primeramente toma la acción bien definida, genera luego uno o varios mensajes de salida y después pasa a un nuevo estado específico en el que espera hasta recibirse el siguiente mensaje de entrada, para comenzar el nuevo ciclo. En términos generales, la secuencia principia desde un estado inactivo, al recibirse un mensaje inicial y termina eventualmente con el regreso al estado de inactividad.
- Una FMM opera en modo de tiempo compartido, de manera que cierto número de FMMs puede ser usado simultáneamente.

En la ISS2199, muchas FMMs existen como una partición más de las diversas funciones, y se comunican mediante mensajes entre ellas mismas. Estos mensajes se definen en las primeras etapas del diseño del sistema y constituyen un paso importante en la consecución de la arquitectura modular. El intercambio de mensajes entre FMMs es soportado por el sistema operativo.

Para prevenir la propagación de errores las FMMs reaccionan únicamente a mensajes de formatos aceptables; los mensajes que no están en la forma apropiada son rechazados o ignorados o pasados a un programa manejador de errores.

Para mantener las FMMs independientes unas de otras las FMMs que transmiten no necesitan conocer la identidad de la FMMs receptoras ni su localización física en la memoria. La identidad de las FMM receptoras es determinada por la manejadora de mensajes la cual deriva la información del tipo de mensaje y la fuente. Este arreglo tiene muchas ventajas.

Una es que la misma estructura de software y sus módulos pueden ser usados para diferentes configuraciones del sistema. Aunque todas las FMMs residen únicamente en un procesador o son distribuidos sobre varios procesadores, esto no tiene influencia sobre la misma FMM. Unicamente el manejador de mensajes habrá de ser informado de tal forma que ella mande mensajes al procesador correcto. Otra ventaja ayuda a minimizar el esfuerzo de diseño de la Ingeniería del Cliente. Cada una de las FMMs puede ser programada y probadas independientemente de los otros módulos como un resultado de las interfaces de mensajes rígidamente definidas. Por lo tanto los cambios de una FMM no requieren de volver a probar el paquete de software completo.

Finalmente, este método facilita la extensión on-line del software sin afectación del servicio. Cada FMM puede ser compilada, enlazada y cargada separadamente de una central que está manejando tráfico. Así, las FMMs representan un conjunto de bloques construidos tal que el software de la central digital ISS2199 puede ser arreglado y puesto en conjunto para satisfacer virtualmente cualquier requerimiento. Esto reduce el esfuerzo de ingeniería del Cliente y acorta los tiempos de instalación y entrega. Y también proveen la facilidad para extensiones con poca complicación cuando el sistema crece.

Lenguajes de programación.

El uso de lenguajes de alto nivel en el diseño del sistema elimina la necesidad de que el programador sepa detalles del funcionamiento de la computadora, permitiendo así que las expresiones usadas en las instrucciones de los programas sean similares a las funciones por implementar.

El lenguaje básico utilizado en la ISS2199 es el CHILL, el lenguaje CCITT de alto nivel diseñado especialmente con lenguaje común estándar para las autoridades mundiales en materia de telecomunicación. El CHILL tiene muchas características que mejoran el proceso de desarrollo del software: estructura de software coherente, legibilidad, transparencia, confiabilidad, tiempo breve de desarrollo, código de origen fuente entendible, facilidad de mantenimiento y de corrección de defectos.

Otro instrumento importante para el diseño del software son los lenguajes orientados hacia problemas (POL). Los Pols tienen todos los elementos necesarios para la programación estructurada, pero incluye comandos para la ejecución directa de funciones de conmutación. Los Pols utilizados en la ISS2199 se basan en el CHILL y reúnen las siguientes características:

- Son lo más parecido posible al lenguaje telefónico de ingeniería.
- Son de fácil entendimiento para personal de ingeniería telefónica, que no necesitan ser programadores para poder implementar debidamente las funciones telefónicas.
- El código producido es mantenido fácilmente, tanto durante la fase de su implementación como por toda la vida útil de la central.

1.1.6 Comandos EXA.

Consultar clase de línea.

Nombre del Comando: Subscriber Categories Print
Abreviación: SUBSCP
Sintaxis: SUBSCP:SNB={NoDirectorio};

Función: Este comando permite al usuario observar hasta un máximo de 8 números(separados por &): , observar la(s) categoría(s) de la(s) línea(s) y clases de control remoto de un abonado.

Restricciones: Solo el comando operará si el número deseado tiene un número de directorio válido y no está en un grupo PABX.

Modificar clase de línea.

Nombre del Comando: Subscriber Categories Change

Abreviación: SUBSCC

Sintaxis: SUBSCC:SNB=(NoDirectorio),SCL=(*Parámetros*);

Donde los parámetros se ven en la figura 1.1.6-1.

Función: Este comando permite al usuario adicionar nuevas categorías, remover categorías asignadas, activar/desactivar clases de control remoto o cambiar datos asociados con clases particulares de uno hasta un máximo de 20 abonados.

Restricciones: Este comando no operará si esta agrupado en grupo PABX.

El (los) abonado(s) seleccionado(s) debe(n) ya estar definido(s) en el sistema.

NOMBRE	NUM PARAM	NUM ARG	SIGNIFICADO	VALOR DEL ARGUMENTO
TBR	1	1	Transferencia en ocupado variable o lista = activar facilidad = desactivar facilidad	 TBR-1 TBR-0
CCA	1	1	Abonado ausente = activar facilidad = desactivar facilidad	 CCA-1 CCA-0

Fig. 1.1.6-1 Parámetros del Comando SUBSCC.

NOMBRE	NUM PARAM	NUM ARG	SIGNIFICADO	VALOR DEL ARGUMENTO
ENQ	1	1	Consulta o llamada tripartita = activar Consulta = activar llamada tripartita = desactivar facilidad	 ENQ-1 ENQ-2 ENQ-0
ALS	1	1	Despertador Automático = activar facilidad = desactivar facilidad	 ALS-1 ALS-0
CAW	1	1	Llamada en espera = activar facilidad = desactivar facilidad	 CAW-1 CAW-0
PRI	1	1	Llamada con prioridad = activar facilidad = activar facilidad = desactivar facilidad	 PRI-1 PRI-2 PRI-0
SSI	1	1	Tipo de Marcación = Disco = Teclado y Disco	 SSI-0 SSI-1
ADI	1	1	Marcación Abreviada = activar facilidad = desactivar facilidad	 ADI-2 ADI-0
DDB	1	1	No molestar = activar facilidad = desactivar facilidad	 DDB-1 DDB-0
CBA	1	1	Restricción de salida = activar facilidad = desactivar facilidad	 CBA-1 al 15 CBA-0
CTF	1	1	Transferencia fija de llamada = activar facilidad = desactivar facilidad	 CFT-1 CTF-0
CTP	1	1	Transferencia variable de llamada = activar facilidad = desactivar facilidad	 CTP-1 CTP-0
MCT	1	1	Rastreo de llamada maliciosa = activar facilidad = desactivar facilidad	 MCT-1 MCT-0
CHT	1	1	Tipo de cargo	CHT-0 al 1
LTE	1	1	Prueba de línea	LTE-0 al 3
TOF	1	1	Protección contra of. llamada	TOF-0 al 1

Fig. 1.1.6-1 Parámetros del Comando SUBSCC (Continuación).

NOMBRE	NUM PARAM	NUM ARG	SIGNIFICADO	VALOR DEL ARGUMENTO
OCG	1	1	Cargo de origen	OCG-0 al 15
ORT	1	1	Ruta de origen	ORT-0 al 15
OTS	1	1	Tiempo sup. de origen	OTS-0 al 3
OBA	1	1	Análisis de origen	OBA-0 al 15
TSU	1	1	Tiempo de sup.	TSU-0 al 1
BOC	1	1	Restricción de salida (El abonado no tiene tono de marcar) = activar facilidad = desactivar facilidad	 BOC-1 BOC-0
TBO	1	1	Restricción de salida (El abonado no tiene tono de marcar) = activar facilidad = desactivar facilidad	 TBO-1 TBO-0
BIC	1	1	Restricción de entrada (Número suspendido) = activar facilidad = desactivar facilidad	 BIC-1 BIC-0
TBI	1	1	Restricción de entrada (Número suspendido) = activar facilidad = desactivar facilidad	 TBI-1 TBI-0
ICS	1	1	Intercepción = número cambiado = número suspendido = desactivación de facilidad	 ICS-1 al 3 ICS-4 al 15 ICS-0
ASU	1	1	Abonado ausente fijo = activar facilidad = desactivar facilidad	 ASU-1 al 15 ASU-0
TLI	1	1	Tipo de línea de abonado	TLI-0 al 15
TCL	1	1	Tipo de abonado	TCL-0 al 15
EMA	1	1	Área de Emergencia	EMA-0 al 15
ISE	1	1	Información costo de llamada	ISE-0 al 15
LLE	1	1	Longitud de línea	LLE-0 al 15

Fig. 1.1.6-1 Parámetros del Comando SUBSCC (Continuación).

Consulta de Contadores.

Nombre del Comando: Consulta de contador de llamadas, locales y metropolitanas.
Abreviación: CONT
Sintaxis: CONT:SNB=(NoDirectorio);
Función: Este comando permite al usuario observar hasta un máximo de 8 números(separados por &), y así observar el(los) contador(es) local(es) y metropolitano(s) de la(s) línea(s) especificada(s).
Restricciones: Solo el comando operará si el número deseado tiene un número de directorio válido y no está en un grupo PABX.

Prueba de las condiciones de la línea.

Nombre del Comando: Prueba las condiciones de la línea (mediciones de capacitancia y resistencia.
Abreviación: CONDLI
Sintaxis: CONDLI:SNB=(NoDirectorio),MP=(Tipo de prueba);
Función: Este comando permite al usuario observar las condiciones de capacitancia y de resistencia entre los hilos, hacia tierra o batería, además de observarse desconexiones hacia la central o hacia la red, este comando permite probar hasta un máximo de 8 números (separados por &), y observar la(s) condición(es) de la(s) línea(s) específicada(s).
Tipo de prueba = 5 permite ver información completa.

1.1.7 Comandos Sistema 2199.

Consultar clase de línea.

Nombre del comando: DISPLAY-SINGLE-SUBSCR

Abreviación: 78

Sintaxis: DISPLAY-SINGLE-SUBSCRIBER: {DN=K' *NoDir*}{*Parámetros*}{,} | {;}

Ver parámetros en figura 1.1.7-1

Función: Este comando permite al usuario observar los diferentes medidores, observar clases de línea, y clases de control remoto de un abonado.

Restricciones: Solo el comando operará si el número deseado tiene un número de directorio válido y si no esta ocupado al llamar el teléfono.

Modificar clase de línea.

Nombre del comando: MODIFY-SINGLE-SUBSCR

Abreviación: 77

Sintaxis: MODIFY-SINGLE-SUBSCR: {ABORT} | {DN=K' *NoDir*}{PARÁMETROS}{,} | {;}

Donde las parámetros se ven en la figura 1.1.7-1

Función: Este camando permite al usuario adicionar nuevas clases, remover clases asignadas, activar/desactivar clases de control remoto o cambiar datos asociados con clases particulares de uno hasta un máximo de 20 abonados.

Si un MODIFY múltiple es solicitado, las características especificadas serán aplicadas a todos los abonados especificados.

Usando el parámetro ABORT, un comando de modificación previo puede ser cancelado.

En caso de solicitud múltiple, el usuario puede pedir reportes de resultados intermedios mediante la especificación del parámetro DETAIL.

Restricciones: Debido a las características particulares de algunas categorías, se presenta el problema de compatibilidad entre categorías, por lo que hay que tomar en cuenta que cuando se usan unas categorías, no se pueden usar otras.

Para adicionar la facilidad de mal pagador BADP, y cuando está activa la restricción del originante por clave de software (ORSWK), primero desactive ORSWK.

El(los) abonado(s) seleccionado(s) debe(n) ya estar definido(s) en el sistema.

NOMBRE	NO PARM	NO ARG	SIGNIFICADO	VALOR DEL ARGUMENTO	OBSERVACIONES
ABORT	31		Cancelar comando		
DN	1	1	Número(s) de directorio = K'Número	0 a 9	16 Dígitos Máximo (20) argumentos
TRSTR	23	1	Restricción permanente en llamadas terminales = adicionar facilidad = remover facilidad	 ADD /1 REM /2	
ANNUL	2	1	Anular todas las facilidades de control remoto		
NTCD	13	1	No tarificación al abonado llamado = adicionar facilidad = remover facilidad	 ADD /1 REM /2	
TOBS	22	1	Observación en el destino = observación normal = observación dirigida = remover facilidad	 NORM /1 DIR /2 REM /3	

Fig. 1.1.7-1 Parámetros del Comando MODIFY-SINGLE-SUBSCR.

NOMBRE	NO PARM	NO ARG	SIGNIFICADO	VALOR DEL ARGUMENTO	OBSERVACIONES
TRFI	26		Transferencia fija de llamadas terminales		argu 2 y 3 no dados
		1	= remover transferencia	REM /0	ACT o DACT sólo
			= activar transferencia	ACT /254	después de
			= desactivar transferencia	DACT /255	asignadas
		2	= dígitos del DN que recibirá llamada	K' ...	16 dígitos máximo
		3	= dígitos del DN que recibirá llamada	K' ...	4 dígitos máximo
CBOX	4		Tipo de aparato de alcancías		
		1	= facilidad moneda única	SINGLE /1	
			= facilidad moneda única	SINGLE-S /2	
			= facilidad multi-moneda	MULTI /3	
			= facilidad multi-moneda	MULTI-S /4	
			= remover tipo de aparato	REM /5	
		2	de alcancías = los dígitos de la última identificación son 9999	I9999	
COL	5	1	Clase de línea		
			= línea de datos	DATL /1	
			= línea de mantenimiento	MNTL /2	
			= línea prioritaria	PRTO /3	
			= línea de operador	OPER /4	
			= línea de prueba de troncal	TKTLINE /5	
			= remover tipo de línea	REM /8	
DBLNG	6	1	Facturación detallada		
			= para todo tipo de llamadas	ALL /1	
			= para llamadas de larga distancia nacionales + internacionales	TOLLINT /2	
			= para llamadas internacionales	INTAL /3	
			= para servicios especiales	SVCE /4	
			= remover facilidad	REM /5	

Fig. 1.1.7-1 Parámetros del Comando MODIFY-SINGLE-SUBSCR (Continuación).

NOMBRE	NO PARM	NO ARG	SIGNIFICADO	VALOR DEL ARGUMENTO	OBSERVACIONES
DBO	7	1	Observación de facturación detallada = dirigida = remover facilidad	 DIR /2 REM /4	
HM	8	1	Medidor de casa = Inversor de polaridad = adaptador de 16 Khz = remover facilidad	 POLAR /1 KHZ16 /2 REM /3	(Usado para Ladatel)
BADP	3	1	Mal pagador = adicionar facilidad = remover facilidad	 ADD /1 REM /2	
LNCHAR	9	1	Características de línea = remover facilidad = no permita prueba de línea hacia afuera = híbrido instalado para prueba manual de línea = extensión del bucle = extensión del bucle (vía hilo C) = inversión de polaridad	 REM /1 NOLNTST /2 HYBRID 3 LOOPEXT /4 LOOPEXTC/5 POLAR /7	Hasta 4 argumentos si REM no es dado por el argumento1
OOBS	14	1	Observación en la origen = normal = dirigida = remover facilidad	 NORM /1 DIR /2 REM /3	
TRIPLEX	15	1	Línea triplex = adicionar facilidad = remover facilidad	 ADD /1 REM /2	
ORPT	16	1	Restricción permanente de origen = Tipo de llamada restringida = remover restricción	 SI 34 NONE	

Fig. 1.1.7-1 Parámetros del Comando MODIFY-SINGLE-SUBSCR (Continuación).

NOMBRE	NO PARM	NO ARG	SIGNIFICADO	VALOR DEL ARGUMENTO	OBSERVACIONES
RECALL	18		Registro de llamada		Hasta 4
		1	= remover tipo de rellamada	REM /1	argumentos si REM no es dado por el argumento 1
			= terminación de llamada con abonado ocupado	CCMB /2	
			= llamada en espera	CWTG /3	
			= retención de consulta	CSHLD /4	
			= transferir conversación	CONVTF /5	
			= conferencia 3 vías	CONF3WAY/8	
			= conferencia marcada	DIALCONF	
SSIG	20	1	Señalización de abonado	DLSET /1	Marcación Z
			= tipo de aparato de disco	DLSETX /2	Marcación X
			= aparato de disco	PBSET10 /3	10 Teclas
			= aparato de botonera	PBSET /4	12 Teclas
			= tipo de aparato de botonera	PBSET16 /5	16 Teclas
				CBSET10 /6	Z-10 Teclas
			= aparato de botonera	CBSET /7	Z-12 Teclas
			= aparato combinado	CBSET16 /8	Z-16 Teclas
			= tipo de aparato combinado	CBSETX10 /9	X-10 Teclas
				CBSETX10 /10	X-12 Teclas
			= aparato combinado	CBSETX16 /11	X-16 Teclas
			= aparato combinado		
			= aparato combinado		
			= aparato combinado		
HOTL	24		Línea directa		
		1	= inmediata	DRTD /1	
			= retardada	DLYD /2.	
			= remover facilidad	REM /3	
		2	= K' dígitos del numero llamado	K' ...	
ORSWK	17		Restricción de originante		ACT o DACT sólo
		1	por		después de
			clave de software	SI 34	asignadas
			= Tipo de ramada restringida	NONE /0	
				ACT /2	
			= sin restricción	DACT /3	
			= activar facilidad		
			= desactivar facilidad		

Fig. 1.1.7-1 Parámetros del Comando MODIFY-SINGLE-SUBSCR (Continuación).

NOMBRE	NO PARM	NO ARG	SIGNIFICADO	VALOR DEL ARGUMENTO	OBSERVACIONES
SUBGRP	21		Grupo de abonados para enrutamiento/tarificación		También llamado código de origen (SCO)
		1	= número de grupo	1 a 32767	
KEYNR	25		Número clave para Acceso		4 dígitos, no si REM es dado
		1	ó Activación a Facilidad	ADD /1	
			= adicionar facilidad	REM /2	
			= remover facilidad	MDFY /3	
			= modificar numero clave	K'...	
			= K' número		
LOOS	10		Línea fuera de servicio		
		1	= adicionar facilidad	ADD /1	
			= remover facilidad	REM /2	
REDIR	19		Redirección		
		1	= servicio en ausencia automático	AUTBSE /1	
			= servicio de no molestar	DNSDT /1	
			= reactivar redirección	DACT /5.	
TRVA	27		Transferencia variable		argu 2 y 3 no son dados si DACT es especificado
		1	= abonado	SUBSCR /1	
			= anuncios	ANNM /2	
			= transferencia en ocupado	BYTF /3	
				DACT /5	16 dígitos
		2	= desactivar facilidad	K' ...	máximo
			= dígitos del DN que		
		3	recibirá la llamada	K' ...	4 dígitos máximo
			= dígitos del DN que recibirá la llamada		
WAKE	29		Facilidad de despertador		argu 2 y 3 no son dados si DACT es especificado
		1	= activar facilidad	ACT /4	
			= desactivar facilidad	DACT /5	
		2	= tiempo de despertador	0 a 23	
		3	(hrs)	0 a 59	
			= tiempo de despertador (min)		

Fig. 1.1.7-1 Parámetros del Comando MODIFY-SINGLE-SUBSCR (Continuación).

NOMBRE	NO PARM	NO ARG	SIGNIFICADO	VALOR DEL ARGUMENTO	OBSERVACIONES
MALOC	11		Llamada maliciosa		
		1	= adicionar facilidad	ADD /1	
			= remover facilidad	REM /2	
NOASWT RF	28		Transferencia sin respuesta		
		1	= activar facilidad	ACT /4	
			= desactivar facilidad	DACT /5	
		2	= dígitos del DN que recibirá la llamada	K' ...	16 Dígitos máximo
			= dígitos del DN que recibirá la llamada	K' ...	4 Dígitos máximo

Fig. 1.1.7-1 Parámetros del Comando MODIFY-SINGLE-SUBSCR (Continuación).

MNEMONICO	VAL	SIGNIFICADO
CONT	7	Solo llamadas continentales
LOC	4	Solo llamadas locales
NAT	6	Solo llamadas nacionales
NDA	2	No llamadas permitidas (no para orswk)
SSE	3	Solo llamadas de servicio especial
ZON	5	Solo llamadas de zona

Fig. 1.1.7-2 Restricciones de Origen.

Consulta de Contadores.

Se usa el mismo comando descrito en Consulta de Categorías.

Prueba de las condiciones de la línea.

Nombre del Comando: Display-Line-Status (Prueba las condiciones de la línea mediciones pruebas de capacitancia y resistencia).
Abreviación: 528

Sintaxis: DSIPLAY-LINE-STATUS:DN=K'(NoDirectorio),4=2;

Función: Este comando permite al usuario observar las condiciones de capacitancia y de resistencia entre los hilos, hacia tierra o batería, además de observarse desconexiones hacia la central o hacia la red, este comando permite probar hasta un máximo de 8 números (separados por &), y observar la(s) condicion(es) de la(s) línea(s) especificada(s).

1.1.8 Interpretación de Respuestas MOA.

```
SUBSCP: SNB= 62130000;
SUBSCRIBER DATA
SNB          DEV     DETY    SUT     SCL     MIS
62103409     LI1-0                   SSI-1
```

Fig. 1.1.8-1 Respuesta Típica de Consulta de Clase de Línea.

```
SUBSCC:SNB=62130000,SCL=BOC-1&TBO-1;
 SUBSCC:SNB=62130000,SCL=BOC-1&TBO-1;
END;
EXECUTED
```

Fig. 1.1.8-2 Respuesta Típica de Modificación de Clase de Línea.

```
{3}<CONT:SNB=16170350;
    CALL METER VALUES READING
    INDIVIDUAL
    SNB       A CMVL    CMVS    TYPE
    16170350  4 00000007 00000000
    END
    {3}<
```

Fig. 1.1.8.3 Respuesta Típica de Consulta de Contadores.

```
WO    18-6/COP2/32/0709/014  AT-6    TIME 960430 1809  PAGE   1
{3}<CONDLI:SNB=16170000,MP=5;
    SUBSCRIBER LINE TEST
    DEV          SNB        MP  FCODE
    LI1-38912    16170000   5
    VOL  IRAB  IRABE  IRAB48  PHO  BRK  LOOP  DIAL  KEYSET  BELL
    OK   >500  >500   >500   CON

    {3}:END;
    CONCLUSION OF COMMAND CONDLI
```

Fig. 1.1.8-4 Respuesta típica de Prueba de las Condiciones de Línea.

1.1.9 Interpretación de Respuestas Sistema 2199.

```
SEC=1719.960523 9002
  COM=0078
  COMANDO ENVIADO A EJECUCION
          9000
  RESULTADOS A CONTINUACION

  CORTINEZ        1996-05-23  19:42:03  JU
  SEC=1719.960523 0076
  ADMINISTRACION DE ABONADOS

    VISUALIZAR-ABONADO-INDIVIDUAL                CON EXITO
                              RESULTADO FINAL
    ------------------------------------------------------------
    ND              NE FISICO            CE LOGICO
    63147000        H'727 (1831 ) & 209   H'1BB0(7088 )
    MEDIDORES: 2584   4093   0     1     0     0
    ESPECIFICACIONES:
        SENAB  : AP COMB Z-16
    CLASES DE LINEA :
        CTRLREM : RECAUT

    REPORTE COMPLETO =      0076
```

Fig. 1.1.9-1 Respuesta Típica de Consulta de Clase de Línea.

```
SEC=1199.941123 9002
  COM=0077
  COMANDO ENVIADO A EJECUCION
          9000
  RESULTADOS A CONTINUACION
  SEC=1799.941123 0076
  ADMINISTRACION DE ABONADOS

   MODIFICAR-ABONADO-INDIVIDUAL              CON EXITO
                          RESULTADO FINAL
   ------------------------------------------------------------
   ND            NE FISICO          CE LOGICO
   63233456      H'3 (3 ) & 201     H'16F0(5872 )
   MEDIDORES: 31   129  0    0    0    0
   ESPECIFICACIONES: PREVIAS
       CARACLN:  POLAR
       SENAB  : AP COMB Z-12
    ESPECIFICACIONES: ACTUALES
       CARACLN:  POLAR
       SENAB  : AP COMB Z-12
    CLASES DE LINEA :  PREVIAS
       CTRLREM : INDIV
    CLASES DE LINEA :  ACTUALES
       ALC: INDIVID
       RESTPOR: LOCAL
   REPORTE COMPLETO =     0076
```

Fig. 1.1.9-2 Respuesta Típica de Modificación de Clase de Línea.

```
SEC=0873.960523 9002
COM=0078
COMANDO ENVIADO A EJECUCION
        9000
RESULTADOS A CONTINUACION
SEC=0873.960523 0076
ADMINISTRACION DE ABONADOS

  VISUALIZAR-ABONADO-INDIVIDUAL                CON EXITO
                              RESULTADO FINAL
  ---------------------------------------------------

  ND              NE FISICO            CE LOGICO
  16101692        H'1621(5665 ) & 157    H'2A50(10832)

  MEDIDORES: 188    656   0    3    0    0

  ESPECIFICACIONES:
      SENAB  : AP COMB Z-12

  CLASES DE LINEA :
      MEDPRIV : POLAR
      RESTERM
  REPORTE COMPLETO =      0076
```

Fig. 1.1.9-3 Respuesta Típica de Consulta de Contadores.

SEC=2172.940318 0154

PRUEBA

PRUEBA MANUAL SOLICITADA CON EXITO

CENTRAL = 0	DR= H'0000	NUM=10	ND= 64170009
	POT CD	POT CA	
A-B	000,000 VOLT	000,000 VOLT	
TIERRA-A	000,000 VOLT	000,000 VOLT	
TIERRA-B	000,000 VOLT	000,000 VOLT	
	RES	CAP	
A-B	> 010,000 MOHM	687,019 NF	
TIERRA-A	> 010,000 MOHM	307,578 NF	
TIERRA-B	> 010,000 MOHM	297,021 NF	

REPORTE COMPLETO

Fig. 1.1.9-4 Respuesta Típica de Prueba de las Condiciones de Línea

1.1.10. Análisis de protocolos.

Protocolo de control de enlace.

Los protocolos de control de enlace realizan las siguientes funciones básicas:

- Sincronización de tramas y transparencia, estableciendo la delimitación de mensajes para poder recuperarlos a partir de las secuencias de bits o caracteres recibidos por el circuito físico.
- Control de errores de transmisión, introduciendo redundancia en los mensajes para poder detectar los errores causados por el ruido o interferencia de transmisión.

- Coordinación de la comunicación, mediante reglas que determinan el turno de intervención a través del enlace.
- Compartición del circuito físico, multiplexándolo dinámicamente entre diferentes enlaces lógicos, mediante la inclusión de mensajes, para identificar el remitente y/o el destinatario.
- Recuperación ante fallos, supervisando la comunicación, detectando anomalías e intentando restablecer la situación normal.

Sincronización de trama y transparencia.

Principio y Fin.

Se identifica el principio de la trama con un caracter de principio de trama, PDT, utilizándose para ello los caracteres STX (Start of Text) o SOH (Start of Header) en el caso de que la trama incluya una cabecera con información adicional , cerrándose con un fin de trama, FDT, para lo que se utiliza ETX (End of Text) o ETB (End of Block) para distinguir entre la trama final de un mensaje y tramas intermedias. La trama suele contener otros caracteres de control (CC), por ejemplo para el origen o destino de la trama, que preceden a los datos y se termina con la redundancia en la secuencia de verificación de trama (SVT).

Esto ocurre con las tramas que contienen información del nivel superior, mientras que las tramas que solo contienen datos de control del propio protocolo de enlace se identifican mediante secuencia de uno o varios caracteres de control (CC) diferentes a los de principio y fin.

Este tipo de sincronización de trama está asociada a los denominados protocolos orientados a caracter (BSC de IBM, BMP de ISO, etc...) por construirse

los mensajes de control de protocolo con caracteres de control de la comunicación pertenecientes a un alfabeto determinado (ASCII, EBCDIC,...). En la figura 1.1.10-1 se recoge el alfabeto CCITT no 5 que se compone de un grupo de caracteres alfanuméricos, otros de caracteres de control de impresión (retorno de carro...) y el de caracteres de control de la comunicación (STX,ETX,DLE,etc..).

				b_7	0	0	0	0	1	1	1	1
				b_6	0	0	1	1	0	0	1	1
				b_5	0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1		0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	TC_7	SP	0	@	P	'	p
0	0	0	1	1	TC1	DC_1	!	1	A	Q	a	q
0	0	1	0	2	TC2	DC_2	"	2	B	R	b	r
0	0	1	1	3	TC3	DC_3	#	3	C	S	c	s
0	1	0	0	4	TC4	DC_4	o	4	D	T	d	t
0	1	0	1	5	TC5	TC_8	%	5	E	U	e	u
0	1	1	0	6	TC6	TC9	&	6	F	V	f	v
0	1	1	1	7	BEL	TC10	¤	7	G	W	g	w
1	0	0	0	8	FE0	CAN	(	8	H	X	h	x
1	0	0	1	9	FE1	EM	)	9	I	Y	i	y
1	0	1	0	10	FE2	SUB	*	:	J	Z	j	z
1	0	1	1	11	FE3	ESC	+	;	K	[	k	{
1	1	0	0	12	FE4	IS4	,	<	L	\	l	|
1	1	0	1	13	FE5	IS3	-	=	M	]	m	}
1	1	1	0	14	SO	IS2	.	>	N	^	n	¯
1	1	1	1	15	SI	IS1	/	?	O	_	o	DEL

Fig. 1.1.10-1 Alfabeto CCITT no 5.

Este tipo de sincronización de trama no es transparente es decir, no se admiten datos iguales a los caracteres de control; por ejemplo si un dato a transmitir fuese igual al caracter ETX, en recepción se interpretaría como fin de trama. Esta limitación no es importante cuando el nivel superior no se puede generar esos caracteres de control , por ejemplo en una terminal alfanumérica, usando el mismo código que el protocolo. Pero sí tiene importancia cuando la fuente no cumple esa restricción, por ejemplo un cifrador de datos, un terminal trabajando con código distinto al del protocolo, etc.

Para conseguir una transmisión transparente, los símbolos de control generados por el nivel de enlace, duplican su longitud, componiéndose del caracter del modo no transparente precedido por un caracter común de "escape" del protocolo (DLE-Data Link escape). Y los datos se transmiten tal como se entrega el nivel superior, salvo los eventuales caracteres que coincidan con el "escape" DLE que también se duplican. De tal forma que en recepción, al llegar un DLE se espera la llegada del siguiente; si es otro DLE se elimina uno de ellos y el otro se entrega como dato al nivel superior. Si es un caracter de control se interpreta como tal y se ejecuta la acción pertinente. Si el caracter de control recibido no va precedido de DLE se interpreta como dato del nivel superior. Por ejemplo, si el contenido de los datos a transmitir es:

a9 STX ACK 2 6 DLE ETX 3 t y
la trama transmitida es:
DLE STX a 9 STX ACK 2 6 DLE DLE ETX 3 t y DLE ETX SVT

Obsérvese que aunque con el modo transparente este tipo de sincronización evita restricciones a la fuente, la eficiencia del protocolo disminuye por la intención forzada de caracteres DLE, descendiendo en el peor caso

(secuencia de datos idénticos a DLE) a un 50% de eficiencia (2 bits transmitidos por bit de información).

Otro inconveniente de los protocolos que utilizan este tipo de tramas es que son multiformato, es decir, varía el formato de las tramas según sea de información o de control, lo que complica la realización de los protocolos de procesamiento externo.

1.2 Comunicación Serial.

1.2.1 Clases de Puertos que Poseen las Centrales.

Introducción.

Ocasionalmente, esto puede ser inconveniente o no factible para soportar una conexión de cableado directo entre una PBX y el **host computer** (Computador servidor); pero se realiza el acceso remoto entre ambos usando líneas públicas, privadas o contratando servicios de transmisión.

Es costoso proveer de muchos puertos como terminales de usuarios. Las terminales de usuario están en controversia para un número limitado de puertos de computadora, un usuario puede ser bloqueado del acceso al host, cuando las conexiones a través de PBX están disponibles. Una de PBX data-call processing software (software de llamado de datos) es decidir el flujo para los puertos de las computadoras. Por ejemplo, bloquear llamadas cuando están en el servidor de encolamiento (FIFO, first input -first output), priorizar. Como los puertos son liberados, la cola puede ser actualizada y estar lista para recibir datos. Cuando se esta llamando y este llamado llega al encabezado de la cola y un puerto esta libre, entonces el PBX puede reiniciar el llamado de los datos. En el intermedio, los llamadores están libres y pueden atender otros procesos.

El uso del encolamiento para controlar los recursos provee un acceso fácil al manejo de datos de la red. Las estadísticas de encolamiento, tales como el promedio máximo y mínimo de llamadas de los llamadores, el tiempo promedio de espera en la cola, y el número de llamadas completas o abandonadas, proveen un indicador de la utilización de los recursos. Puede

esto adicionarse al aprovechamiento de recursos de los puertos de las terminales.

Puertos.

IOS contiene una interfaz estándar UB 24 a la que se pueden conectar una máquina de escribir, una pantalla o un modem (puerto serial).

Entre las funciones y equipo que ofrece IOS, se tiene la introducción y salida de datos vía máquina de escribir, una pantalla visual con teclado, así como la entrada y salida orientadas hacia un archivo a través de una cinta magnética. IOS tiene también un equipo de canal de datos que permite ubicar a los dispositivos terminales de entrada salida en posiciones remotas.

Los manejadores de entrada/salida soportan la interfaz para los periféricos de conmutadora y la facilidad de comunicación hombre-máquina. Se requiere del sistema operativo para cargar los diferentes elementos de control como software específico. Cada elemento de control tiene un programa local de "bootstrap en memoria de solo lectura el cual es ejecutado cada vez que el elemento requiere cargarse desde el disco o la cinta magnética.

Los medios más comunes de acceso remoto es a través de las facilidades telefónicas por medio de líneas públicas o líneas conmutadas. El mejor de los medios disponibles hoy en día son diseñados para transmisión analógica telefónica; de ahí que, los módem son usados como interfaz de un puerto de PBX al de una troncal telefónica. Para transmisión de datos en gran escala los cuales pueden ser soportados por la amplitud de banda estándar de voz analógica. (es decir, 2400-4800 baudios), utilizan líneas especiales de teléfono e incrementan substancialmente el costo.

Cada uno de los dispositivos de entrada-salida incluyendo los enlaces de datos, se encuentran conectados físicamente con un procesador regional. Este puede ser conectado con un procesador central.(Ver figura 1.2.1-1).

Un bus de datos es una ruta de alta amplitud de banda que es parte de muchos dispositivos. Estaciones de voz, equipos terminales de datos, o computadoras personales obtienen el acceso a el bus de datos a través de interfaces de bus, los cuales provienen de uno o más puertos. (Ver figura 1.2.-2).

Muchas centrales digitales emplean uno o más buses de datos para concentrar los datos prioritarios a switchear.

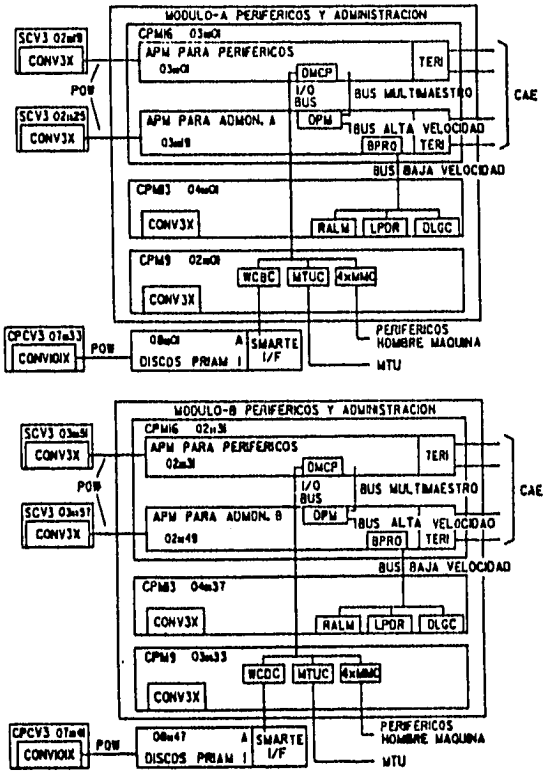


Fig. 1.2.1-1 Distribución del Gabinete de la Central.

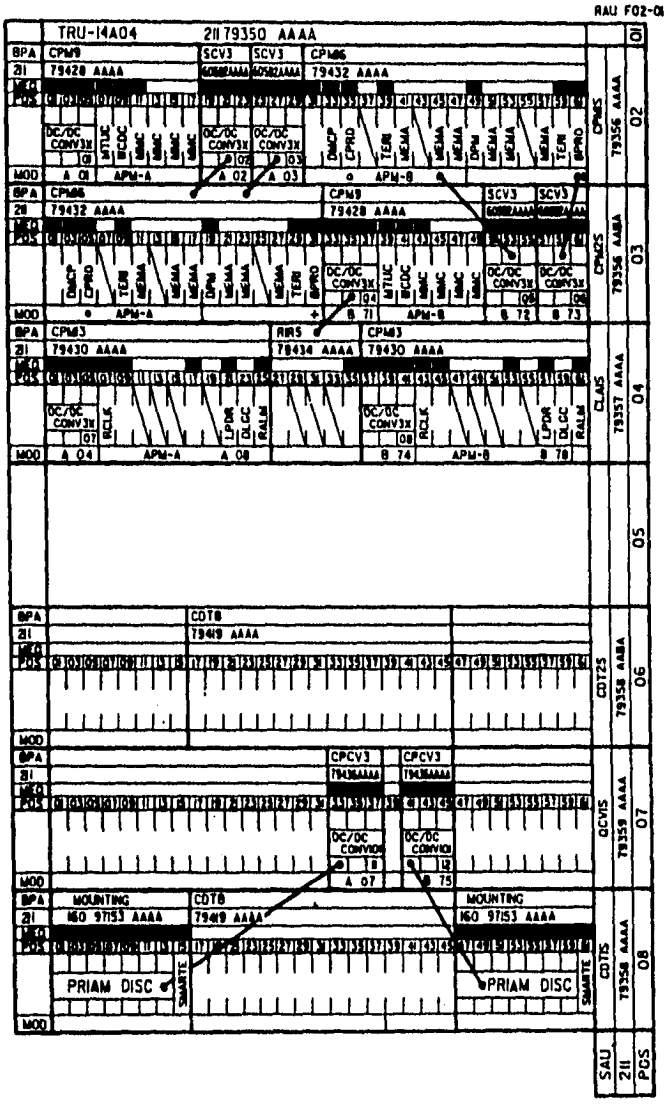


Fig. 1.2.1-2 Distribución de los Puertos.

1.2.2 Interfaz RS-232, RS-422 Y AS-485C.

Introducción.

Los interfaces de nivel físico se utilizan para conectar los dispositivos de usuario a los circuitos de comunicaciones. Para realizar esta importante función, la mayoría de los interfaces de nivel físico describen cuatro atributos. Los atributos eléctricos describen los niveles de tensión (o de corriente) y la temporización de los cambios eléctricos que representan los valores binarios 0 y 1. Los atributos funcionales describen las funciones a ser realizadas por el interfaz físico. Muchos protocolos de nivel físico clasifican esas funciones como control, temporización, datos y tierra. Los atributos mecánicos describen los conectores y los hilos del interfaz. Todos los hilos de control, señalización y datos se encuentran generalmente en un mismo cable y unidos a conectores en ambos extremos del mismo. Los atributos de procedimiento describen la función de los conectores y la secuencia de eventos que es necesario efectuar para la transmisión real de datos por el interfaz.

Interfaz RS-232.

La transmisión serial de datos ha sido utilizada en gran medida como medio eficiente de transmisión de información digital a larga distancia. El convenio de la interfaz RS-232 se desarrollo para estandarizar la interfaz entre el **ETD** (Equipo Terminal de Datos)y los **ETCD** (Equipo Terminal de Comunicación de Datos) utilizando un intercambio de datos a través de datos seriales binarios.

Los ETD y los ETCD se conectan a la vía interfaz estándar RS-232. Un ETD es normalmente un equipo de usuario final, como una terminal o computadora. El

ETCD proporciona al ETD la conexión con el circuito de comunicaciones. La RS-232 describe cuatro funciones del interfaz:

-Definición de las señales de control del interfaz.
- Movimientos de los datos del usuario a través del interfaz.
- Transmisión de señales de reloj para sincronizar el flujo de datos.
- Formación de las características eléctricas reales del interfaz.

RS-232 envía los datos por el interfaz mediante cambios de niveles de tensión. Un 0 binario se representa con una tensión en un rango de +3 a +12 voltios. un 1 binario se representa con un -3 a -12 voltios. La longitud real del cable RS-232 depende de las características eléctricas del cable, aunque los vendedores no permiten una longitud mayor a los 50 pies o su equivalente de 15 metros. El estándar V.28 define un interfaz eléctrico similar a RS-232.

Los circuitos de RS-232 consisten de 25 conexiones (canales).Ver figura 1.2.2-1. No se utilizan los 25 canales. Un interfaz entre dos ETCD normalmente necesita entre cuatro y ocho canales.

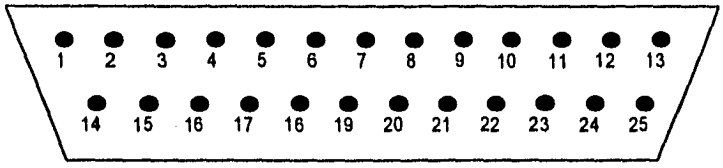
Las funciones de las 25 conexiones como sigue:

Terminal 1 Circuito AA - Tierra de protección: El conductor se conecta eléctricamente al chasis del equipo.

Terminal 7 Circuito AB - Tierra de señal: Tierra común para todos los circuitos. Así se establece la tensión de referencia para las otras líneas. Es un circuito de referencia común.

Terminal 2 Circuito BA - Datos de transmisión: Señales de datos transmitidas desde el ETD hacia el ETCD. Generalmente representan datos de usuario.

Terminal 3 Circuito BB - Datos de recepción: Señales de datos de usuario transmitidas desde un ETCD hacia un ETD.



1. TIERRA DE PROTECCION.
2. DATOS TRANSMITIDOS.
3. DATOS RECIBIDOS.
4. SOLICITUD DE TRANSMISION.
5. PERMISO PARA TRANSMITIR.
6. EQUIPO DATOS PREPARADO.
7. TIERRA DE SEÑAL.
8. DETECCION DE PORTADORA.
9. RESERVADO. NO ASIGNADO.
10. RESERVADO. NO ASIGNADO.
11. RESERVADO. NO ASIGNADO.
12. DETECCION DE PORTADORA.
13. PERMISO PARA TRANSMITIR.
14. DATOS SECUNDARIOS.
15. RELOJ DE TRANSMISION.
16. DATOS SECUNDARIOS.
17. RELOJ DE RECEPCION.
18. NO ASIGNADO.
19. SOLICITUD DE TRANSMISION.
20. TERMINAL DE DATOS PREPARADA.
21. DETECTOR DE CALIDAD DE SEÑAL.
22. TIMBRE INDICADOR.
23. SELECTOR DE VELOCIDAD DE DATOS.
24. RELOJ DE TRANSMISION.
25. NO ASIGNADO.

Fig. 1.2.2-1 Conexiones del Circuito RS-232.

Terminal 4 Circuito CA - Solicitud de envío: Señal de ETD a ETDC. Este circuito sirve para notificar al ETCD que la terminal ó computador tiene datos para transmitir. el circuito CA se utiliza también en las líneas semiduplex para controlar la dirección de la transmisión de datos. La transición de inactivo a activo notifica al ETCD que debe tomar las medidas necesarias para prepararse para la transmisión.

Terminal 5 Circuito CB - Permiso para transmitir: Señal de ETCD que indica al ETD que puede transmitir datos. La señal de permiso puede

activarse tras recibir una señal de portadora precedente del módem remoto. La temporización del CB varía de un módem a otro.

Terminal 6 Circuito CC - Equipo de datos preparado: Señal procedente del ETCD, con lo que indica una de las siguientes condiciones:

a) La máquina esta descolgada, es decir conectada a la línea conmutada.
b) El ETCD ha completado las funciones de sincronización y responde con tonos.

Terminal 20 Circuito CD - Terminal de datos preparados: Señal procedente del ETD que indica que la terminal ó computadora están encendidas, que no se detecta ningún indicio de mal funcionamiento y que no se encuentra en modo de pruebas. Por lo general la línea CD permanecerá activa siempre que el equipo este listo para transmitir o recibir datos. En una configuración conmutada, una señal de timbre del nodo remoto activará generalmente el CD. CD mantiene el canal en condición de conectado.

Terminal 22 Circuito CE - Indicador de timbre: Señal procedente del ETCD que indica que se está recibiendo una señal de llamada por un canal conmutado.

Terminal 8 Circuito CF - Detección de Señal de recepción en línea: Señal procedente del ETCD, con la que se indica que éste esta detectando la señal portadora generada por el módem remoto. Se denomina también Detección de portadora en línea (DCD).

Terminal 21 Circuito CG - Detector de calidad de la señal: Señal procedente del ETCD, con la que se indica que la señal recibida tiene la calidad suficiente para suponer que no ha aparecido ningún error.

Terminal 23 Circuitos CH y CI - Selector de velocidad binaria de la señal: Señales procedentes del ETD y ETCD, respectivamente, que indican la velocidad de señalización de los datos en las máquinas dotadas de velocidad dual. Algunos dispositivos son capaces de transmitir a velocidades binarias variables.

Terminal 24 Circuito DA - Temporización del elemento de señal del transmisor: Señales procedentes del ETD que proporcionan la temporización a las señales de datos que estén siendo transmitidos por el circuito o BA (Datos transmitidos) hacia el ETCD. El ETD se encarga de generar la señal; si es el ETCD el que genera el sincronismo, el circuito empleado es DB.

Terminal 15 Circuito DB - Temporización del elemento de señal del transmisor: Señales procedentes del ETCD que proporcionan la temporización a las señales de datos que estén siendo transmitidos por el circuito o BA (Datos transmitidos) hacia el ETCD. El ETD se encarga de generar la señal; si es el ETD el que genera el sincronismo, el circuito empleado es DA.

Terminal 17 Circuito DD - Temporización del elemento de señal del receptor: Señales procedentes del ETCD que proporcionan al ETD la

temporización necesaria para las señales de datos que estén siendo recibidas por el circuito BB (Datos recibidos).

En RS-232 los niveles de tensión son detectados en el receptor mediante la diferencia relativa de tensión entre el circuito de señal y el circuito de tierra (circuito AB). sin embargo, las estaciones transmisoras y receptora generalmente tienen diferente tierras debido a las diferentes características eléctricas de sus componentes. El voltaje aplicado por el transmisor y el recibido por el receptor pueden ser diferentes. Si la diferencia de potencial es pequeña, no se producirán errores.

Uno de los usos más comunes para RS-232 es la conexión de terminales con computadoras. Esto se realiza a través de modem tanto directa como indirectamente. Otros dispositivos, tales como programadores PROM's e impresoras, han adoptado también los estándares de RS2-232 recientemente.

Interfaz RS-422.

Este circuito es diseñado de modo equilibrado, donde un circuito toma como referencia otro circuito y no una determinada tierra.

El circuito de interfaz de voltaje balanceado normalmente se utiliza para líneas de datos temporizado o control, donde las velocidades de la señal están entre 100 Kbps a 10 Mbps. Las especificaciones del RS-422 no ponen restricciones en la frecuencia de operación mínima o máxima pero sí en la relación de velocidades de transición de un intervalo unitario.

Aunque los circuitos de transmisión simple son normalmente utilizados a bajas frecuencias, la transmisión diferencial en líneas balanceadas pueden ser preferidas bajo las siguientes condiciones:

A. Líneas de interconexión demasiado largas para operación desbalanceada efectiva.

B. Líneas de transmisión expuestas a los niveles de ruido electrostático o electromagnético.

C. Donde se desee una simple inversión de señales (obtenida a cambio de las líneas balanceadas).

Un circuito de interfaz digital balanceado básico consta de 3 partes y un ejemplo está en la figura 1.2.2-2:

A. El generador (G) o manejador de línea de datos.

B. Una línea de transmisión balanceada.

C. Las cargas, donde una carga puede consistir de uno o más receptores (R) y la línea de resistencia terminal de la línea. (R_t)

El tipo de manejador RS-422 tiene una fuente de voltaje de salida balanceada (diferencial) con una impedancia de 100Ω menos. Su salida de voltaje diferencial esta en un rango de 2 V mínimo y como máximo 6 V. Adicionalmente, el voltaje de salida de cualquier salida, con respecto a tierra, no debe exceder a 6 V.

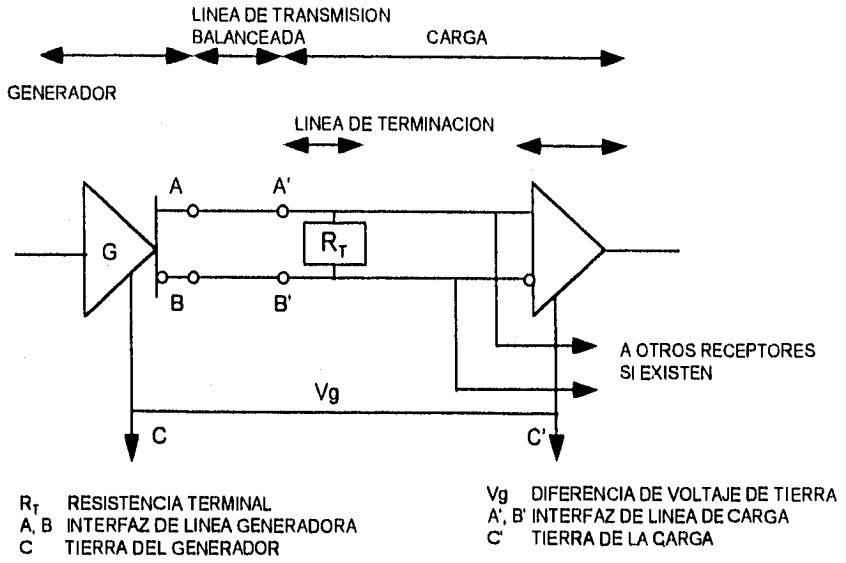


Fig. 1.2.2-2 Interfaz Digital Balanceada.

- El balance de voltaje de salida (V_{OD}) no debe ser menor a 2 V con dos resistencias terminales de 50 Ω (± 1%)en serie entre salidas.

- La diferencia entre la polaridad opuesta del voltaje de salida diferencial debe ser menor a 0.4 V.

- El voltaje de offset de salida del manejador (V_{OS}), medido de la unión de dos resistencias terminales de 50 Ω y la tierra del manejador , no debe exceder a los 3 V(en cualquier polaridad). La magnitud del cambio en (V_{OS}) debe ser menor a 0.4 V para voltajes diferenciales de salida de la polaridad opuesta.

La corriente de salida del manejador, con cualquier salida en corto con tierra, no debe exceder 150 μA. La corriente de fuga en "apagado", con cualquier voltaje entre - 0.25 y 6 V aplicado a cualquier salida, no debe exceder 100 mA.

Los requerimientos de entrada básicos del receptor son como siguen:

- Sensibilidad del umbral de entrada de datos diferencial de ± 200 mV, sobre un rango de modo común (V_{CM}) de -7 a 7 V. Impedancia de entrada mayor o igual a 4 Ω.

- Las características de voltaje-corriente de entrada del receptor deben ser balanceadas de tal forma que su salida permanezca en el estado binario deseado con una entrada diferencial aplicada de 400 mV (a través de 500 Ω ± 1% en cada terminal de entrada)

Interfaz RS-485.

El estándar EIA RS-485, introducido en 1983, es una versión mejorada del RS-422. Incrementa el uso de transmisiones balanceadas en la distribución de datos a varios sistemas, componentes y periféricos sobre las líneas relativamente largas, teniendo la necesidad de múltiples combinaciones manejador/receptor en una línea simple de par trenzado.

El estándar EIA RS-485 toma los requerimientos del RS-422 para la transmisión en líneas balanceadas más las características adicionales para manejadores y receptores múltiples. (Ver figura 1.2.2-3.).

El estándar EIA RS-485 difiere del RS-422 principalmente en las características que permiten comunicaciones multipunto confiables. Para los manejadores, estas características son:

- Un manejador puede controlar hasta 32 cargas unitarias y un total de resistencia en la línea terminal de 60 Ω o más (Una carga unitaria es típicamente un manejador pasivo o un receptor).

- La corriente de fuga de salida del manejador, en "apagado" debe ser de 100 μA o menor con cualquier voltaje entre -7 y 7 V.

- El manejador debe de ser capaz de proporcionar un voltaje de salida diferencial entre 1.5 y 5 V con voltajes de línea a modo común de -7 a 12 V.

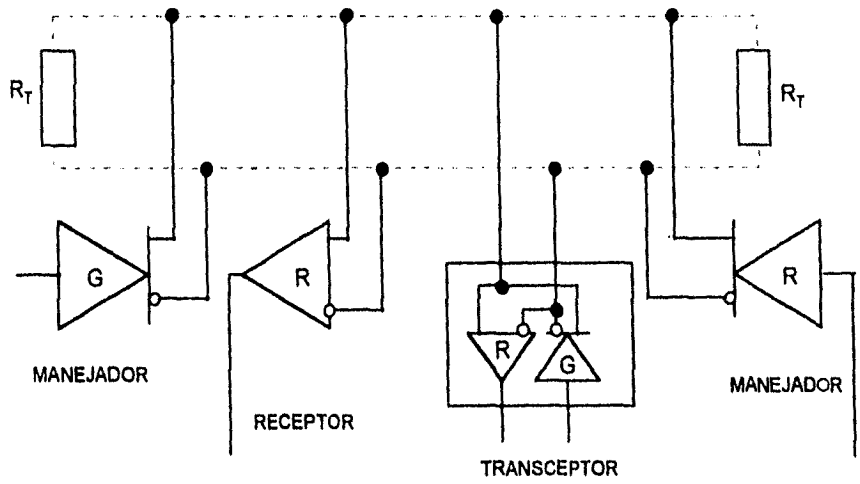


Fig. 1.2.2-3 Interfaz Digital Balanceada Multipunto.

Los manejadores deben tener protección propia contra cualquier contención

(los manejadores múltiples "pelean" por la línea de transmisión al mismo tiempo) esto es, no debe ocurrir daño en el manejador cuando sus salidas están conectadas a una fuente de voltaje de -7 a 12 V, si su estado de salida es 1 binario, 0 binario o pasivo.

Para los receptores las características son :

- Alta resistencia de entrada del receptor, 12 Ω mínimo.
- Un rango de voltaje de entrada a modo común entre -7 y 12 V.
- Sensibilidad de entrada diferencial de ±200 μV sobre un rango de -7 y 12 V.

PARAMETRO	RS232	RS422	RS485
Modo de operación	Terminado sencillo	Diferencial	Diferencial
Número de manejadores y receptores permitido	1 Manejador 1 Receptor	1 Manejador 10 Receptores	32 Manejadores 32 Receptores
Longitud máxima del cable (m)	15	1200	1200
Velocidad máxima de datos (bits por segundo)	20 k	10 M	10 M
Voltaje máximo de modo común (volts)	± 25 V	6V, -0.25V	12V, -7V
Salida del manejador	± 5 V mínimo ± 15 V máximo	± 2 V mínimo	± 1.5 V mínimo
Carga del manejador	3 k (Ω) a 7 (Ω)	100 Ω	60 Ω
Slew rate del manejador	30 V/μ máximo	No aplica	No aplica
Límite de corriente de corto circuito de la sálida del manejador	500 μ A a VCC ó a tierra	150 μ A a tierra	150 μ A a tierra, 250 μ A a 8 ó 12 V.

Fig. 1.2.2. - 4 Características, Ventajas y Desventajas de la Transmisión Síncrona y Asíncrona.

PARAMETRO	RS232	RS422	RS485
Resistencia del manejador (Ω)			
Encendido	No aplica	No aplica	120 Ω
Apagado	300 Ω	60 Ω	120 Ω
Resistencia de la entrada del receptor (Ω)	3 kΩ a 7 kΩ	4 kΩ	12 kΩ
Sensibilidad del receptor	± 3 V	± 200 mV	± 2 mV

Fig. 1.2.2. - 4 Características, Ventajas y Desventajas de la Transmisión Síncrona y Asíncrona. (Continuación).

1.2.3 Características, Ventajas y Desventajas de la Transmisión Síncrona y Asíncrona.

Introducción.

Los sistemas síncronos y asíncronos pueden ser tanto en serie como en paralelo (vectorial). La mayoría de los módem, por utilizar la línea telefónica conmutada, emplean un sistema asíncrono de comunicación.

El sistema síncrono cuya diferencia del asíncrono radica en que tanto el ordenador como el receptor quedan sincronizados, es decir, sus ciclos de lectura escritura de datos son coincidentes. Además los bits se transmiten en grupos llamados tramas.

Características, Ventajas y Desventajas de la Transmisión Asíncrona.

La transmisión de datos asíncrona es comúnmente usada para datos menores a 19 Kbps. Este es el esquema usado en la mayoría de terminales usadas hoy en día. En la transmisión asíncrona las referencias internas de tiempo o relojes, no intervienen en los procesos de transmisión y recepción de datos. El transmisor puede enviar una o mas unidades de datos listos para ser enviados. Cada unidad de dato debe tener un formato. En otras palabras cada unidad de datos debe contener bits de inicio y bits de paro(start bit, stop bit) indicando el principio y fin de cada una de ellas.

Desde solo un manejador de bits son enviados al mismo tiempo. El receptor puede resincronizar este reloj al comienzo de cada transmisión y así el reloj no necesita la precisión que podría requerirse para la sincronización de una gran serie continua de bits. Para ayudar en la detección de errores un bit de paridad puede ser adicionado al final de cada byte. El bit de paridad es seleccionado para hacer un número non o par para ser checado. El chequeo de la paridad puede detectarse con un simple bit de error el cual es usualmente suficiente, dado el pequeño número de bits mandados por cada carácter.

Se requiere de un circuito integrado de interfaz entre la microcomputadora y el dispositivo de entrada/salida. Este circuito integrado desempeña las siguientes funciones:

- Convertir una unidad de datos en bits paralelas de la microcomputadora a datos seriales para transmisión al dispositivo serial.

- Convertir los datos seriales del dispositivo de entrada/salida a una unidad de datos de bits paralelos para transmitirlo a la microcomputadora.

Cada unidad de datos seriales asíncrona puede dividirse en intervalos de tiempo iguales ,llamados intervalos de bit. Un bit de dato puede tener un nivel bajo o alto durante cada intervalo de bit. Un dado de 8 bits tendrá 8 intervalos de bit. Cada bit de datos corresponderá a uno de los 8 bits de intervalos de bit.

El formato para datos seriales asíncronos contienen la siguiente información:

- Un bit con nivel bajo para el inicio.
- De 5 a 8 bits de datos, denotando el dato actual que se esta transfiriendo.
- Un bit de paridad opcional, ya sea de paridad par o impar.
- Uno, uno y medio o dos bits de paro que tengan niveles altos .

El siguiente ejemplo muestra un dato serial asíncrono con un bit bajo de inicio, un bit de paridad impar y un bit de paro.

0	10010100	0	1
Bit de inicio	Datos	Paridad	Bit de paro

En estos sistemas cada dato se envía secuencialmente, precedido por un bit de arranque y seguido por un bit de parada. El tiempo entre dato y dato es variable.

El bit de arranque tiene por misión activar en el equipo receptor la lectura de los datos(bits) enviados. El bit de parada deja al receptor en estado de espera.

Características, Ventajas y Desventajas de la Transmisión Síncrona.

La característica principal de la transmisión serial síncrona de datos es que estos son enviados o recibidos basándose en una señal de reloj. Después de decidir la velocidad de transmisión (baud rate, bits por segundo).El dispositivo transmisor envía un bit del dato en cada pulso del reloj. Para poder interpretar correctamente los datos, el dispositivo receptor debe conocer el inicio y fin de cada unidad de datos. De allí, en una transmisión síncrona el receptor debe reconocer el numero .de unidades de datos a ser transferidos. También el receptor debe de estar sincronizado con los limites del dato.

Generalmente se utilizan uno o dos caracteres de sincronia para indicar el inicio de flujo de datos.

1101 0110	00111
SINCRONIA	Primer bit del dato

La unidad de datos debe de contener un bit de paridad. En cada unidad de datos estos consisten de 5, 6, 7 u 8 bits para datos el resto de ellos son ignorados. Una unidad de datos de 9 bits con 8 bits de datos y 1 de paridad se puede representar como:

XXXX XXXX	P
8 bits de datos	1 bit de paridad.

El bit de paridad puede ser par o impar.

El receptor síncrono espera en un modo de "caza" mientras espera los datos. Tan pronto como el receptor reconoce uno o dos bits de sincronía, basándose en el número de caracteres de sincronía utilizados, empieza a interpretar los datos. En la transmisión síncrona, el transmisor requiere enviar continuamente los datos al receptor. De cualquier forma, sí los datos no están listos para ser transmitidos el receptor enviara caracteres de sincronía hasta que los datos se encuentren listos.

Por ejemplo, un sistema síncrono, donde cada bit corresponde con un ciclo de reloj, se llama frecuencia de bit al inverso del período de la señal del reloj. De esta forma, en una transmisión serie síncrona, el tiempo de un bit incluido en un carácter es múltiplo entero del período de la señal. La sincronización a nivel de bit se realiza reconstituyendo la base de tiempos de la señal a partir de las transiciones (1-0, 0-1). a nivel de carácter se utilizan códigos especiales. Para que las transiciones sean frecuentes, el mensaje de datos a enviar es aleatorio y en recepción se aplica el mismo proceso para recuperarlo.

Hay que tener presente que, dentro de la palabra, los bits se transmiten de forma síncrona. La sincronización entre emisor y receptor es fundamental para que éstos puedan intercambiar información. Esta se produce tanto a nivel de los bits (por coincidencia de la frecuencia nominal de los relojes de emisor receptor) como a nivel de los caracteres (diferenciar un carácter de otro por la secuencia de comienzo y parada). De no producirse la sincronización, el receptor obtendría de la señal recibida datos distintos de los realmente enviados.

1.2.4 Acceso a las Centrales.

Todo acceso al dispositivo telefónico, tanto para el manejo de llamadas (lógica de señalización) como para el mantenimiento del mismo, se efectúa a través del manejador de dispositivos especializados para cada tipo de hardware de terminal, es decir, de línea de abonado analógico, troncal, digital, emisor, receptor de MF, etc. La mayoría de las FMMs asociadas están en el ACEs del sistema de señales lógicas en el nivel de usuario.

Cada manejador de dispositivos, junto con su circulteria asociada, aparece como un "dispositivo virtual" para el resto del software del sistema.

Las centrales telefónicas proporciona una gama muy útil de mediciones de datos en las que pueden modificarse o ampliarse rápidamente. Las características de las líneas telefónicas, los datos de los abonados, los datos de los contadores, las alarmas, etc. se hayan separadas en módulos, asegurando la modularidad y confiabilidad de datos.

Acceso Manual a la Central Telefónica.

Una vez comprobado que el módem responde y la comunicación fuera por línea conmutada hay que marcar al teléfono del modem de la central, por ejemplo:

ATDT123456

En caso de que este ocupado o no responda, se contacta con el personal de la central para verificar que este encendido el modem y reanudar el llamado.

Al responder al llamado que se le hace a la central nos envía la palabra: CONECTED 1200 o cualquier velocidad a la que este trabajando.

Ejemplos de Respuesta de Diferentes Tipos de Centrales.

- MOA

[Backspace]	Pide password de acceso.
[Control + B]	Envía un break y se espera que responda la central.
[Logout]	Desconectarse de la central.

- GIO11

[Enter]	Pide acceso a la central.
[CHP]	Lee los contadores de un teléfono en particular.
[Control + E]	Desconectarse de la central.

- 1540 NAM

[Control + B]	Pide acceso a la central.
[99]	Lee los parámetros generales de un teléfono en particular.
[Control + E]	Desconectarse de la central.

- 1740 GIO11

[Control + B] Pide acceso a la central.

[99] Lee los parámetros generales de un teléfono en particular.

[Control + E] Desconectarse de la central.

Acceso Automático a la Central Telefónica.

Hay que configurar un módulo diseñado para el acceso directo a la central telefónica con los siguientes datos:

Limite Inferior: Nos pide el número telefónico en orden ascendente menor que vayamos a accesar.

Limite Superior: Nos pide el número telefónico en orden ascendente mayor que vayamos a accesar.

Nota: El acceso puede ser a un solo número ó a una serie en particular.

Número de dígitos: Número de dígitos que componen el (los) número(s) a probar, esto es, los números que componen la clave lada mas los que componen el número a probar ejemplo: En la ciudad de Cuernavaca el número de dígitos sería igual a 10, que es compuesto por 4 dígitos de la clave lada (9173) y seis dígitos que corresponden

a los números telefónicos de esta zona en específico y varia de ciudad a ciudad.

Tipo de central: Tipo de central con el que se que va a trabajar.

Password: En algunos casos es requerido que se de un número de acceso a la central el cual es proporcionado por el personal responsable de la central.

Login: Se usa para accesar a ciertas centrales MOA.

ArchivoCFG: Es el archivo que contiene la configuración del modem previamente configurado (ya que dependiendo del modem que se utilice es necesario configurarlo).

No. de módem: En caso de que la central telefónica tenga más de un modem se indicará por cual accesaremos.

Es importante hacer notar que el acceso a los módulos de las centrales está en función y permisos del personal responsables de ella.

1.2.5 Características de los UART I.

Definición de UART I.

En inglés Universal Asynchronous Receiver/Transmisor, que en español significa: Interfaz Universal Asíncrona de Transmisión /Recepción.

Es un subsistema completo e integrado en un chip que convierte datos en paralelo a datos serie, y datos serie en datos en paralelo. La UART determina e

inserta los bits de paridad recibidos, crea el bit de inicio, selecciona e inserta los bits de parada, controla el número de bits por carácter e incluso lo almacena todo para tener tiempo de realizar todas estas tareas. Mucho de este trabajo se realiza con dispositivos externos (interruptores).

La interacción entre el software de comunicaciones y la UART presenta un alto nivel de transparencia. El software le indica a la UART que envíe o reciba caracteres, y que inicie o detenga el flujo de datos (Control de flujo). Ver figura 1.2.5-1.

Características de los UART I.

- Convierte datos paralelos en datos series.
- Convierte datos serie en datos paralelos.
- Formatea los datos para su transmisión.
- Suprime los bits de formato tras la recepción.
- Puede transmitir, recibir y dar servicio de acceso al CPU, todo ello a velocidades distintas de reloj.
- Todas las operaciones son orientadas por bytes. Esto es, hace su interfaz con el sistema de buses de 8 bits al mismo tiempo.
- Todos los registros de escritura pueden ser solamente de escritura; es decir, su contenido puede ser no recuperable.
- El generador del reloj maestro es externo al UART.
- 16 rangos para la transmisión de 50 a 19200 bps están disponibles.
- El baud rate esta bajo control de programa.

UART I Transmisor.

Un byte a ser transmitido es escrito a la dirección del transmisor del UART I, donde es eventualmente presentado a la sección de transmisión. Esta sección consiste principalmente de un registro de cambio, el control lógico para cargar el byte dentro del registro de cambio, y uno o más buffers de transmisión. Ver figura 1.2.5-1.

El transmisor por si mismo consiste de una entrada paralela y una salida serial. Los bits a ser transmitidos son cargados dentro del registro de cambio, luego son cambiados en la transición negativa del reloj de transmisión de datos. Cuando todos los bits han sido cambiados fuera del registro del tranmisor el próximo SDU (Serial Data Unit, Unidad de Datos Serial) es cargado y el proceso se repite. No contando con el bit de comienzo, el registro de cambio puede ser tan grande para acomodar de 7 bits (5N1) a 11 bits (8P2), aunque el posterior es raro.

Los bits de datos pueden ser llamados directamente del sistema de bus de datos, pero más UARTs contienen un buffer de transmisión, o registro de ocupación del transmisor, el cual forma una pequeña cola en la cual los datos pueden ser temporalmente asignados mientras esperan la serialización. Un byte para transmitir puede por consiguiente ser escrito en el registro. Este tipo de manejo de transmisión de datos no solo es conveniente, esto hace la transmisión más eficiente. Porque hay un SDU completo sin el cual el procesador puede reemplazar el buffer, esto es fácil de guardar al registro de cambio perpetuamente ocupado. Ahí pueden ser transmitidos varios buffers, pero muchos UARTs contienen solo uno.

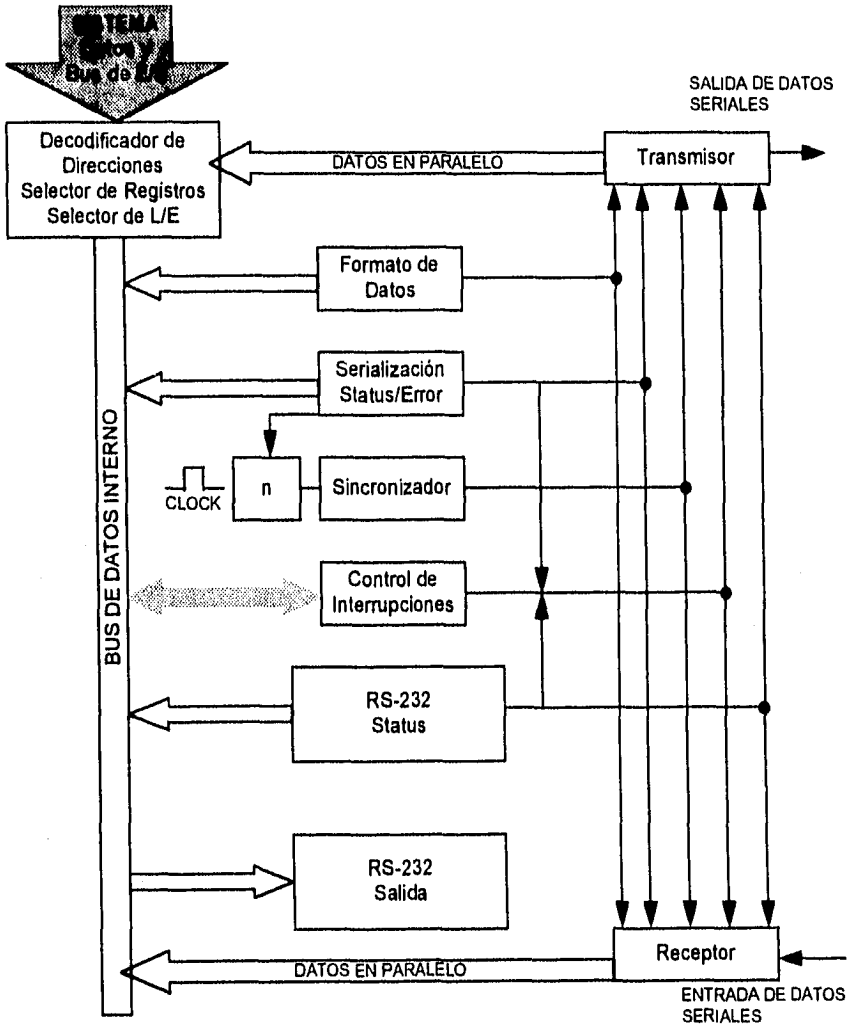


Fig. 1.5.1-1 Diagrama Hipotético de un UART.

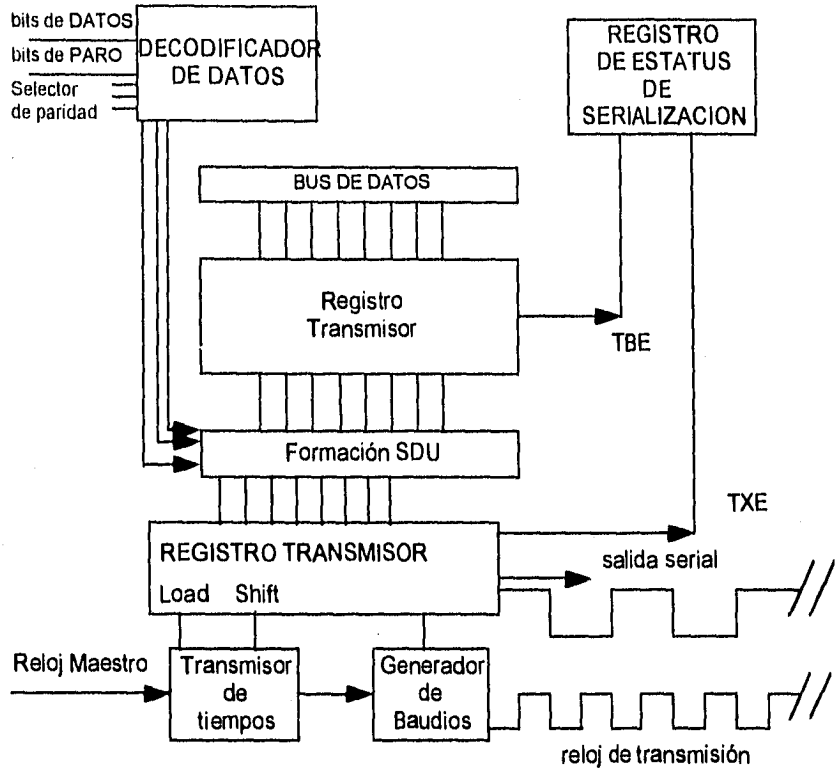


Fig. 1.2.5-2 UART Transmisior.

Status del Transmisor.

Los programas escritos para transmitir datos requieren solamente dos elementos de información acerca de la transmisión: que ambos el buffer de transmisión y el registro de cambio del transmisor estén vacíos. Cada uno de estas condiciones es reportada por medio de un bit en el registro de estatus de serialización. Cuando los datos del buffer son llamados dentro del transmisor, una bandera de buffer transmisor vació (Transmitter Buffer Empty TBE) señala al

UART que puede aceptar otro byte. Si el TBE esta limpio después del final del bit del SDU es cronometrado en el registro de cambio, la bandera de transmisión vacía señala que el registro de cambio esta también vacío.

Durante la transmisión dúplex completa (full duplex), el software necesita solamente probar el valor del bit TBE (en el registro de estado de serialización) para hacer cierto que el buffer esta vacío antes de escribir un byte a el UART. Los modem pueden invertir sus papeles, de transmitir a recibir y viceversa. antes el software podría dar instrucciones de transmisión al modem para comenzar a recibir , esto conociendo no solamente que el buffer esta vacío pero también que el registro de cambio del transmisor esta así mismo vacío.

UART I Receptor.

Un byte es recibido por la lectura de la dirección en la sección de recepción de los UARTs, cuyo trabajo es construir un SDU de bits traído de la línea de entrada serial. El UART debe entonces extraer el byte de datos de la información forrajeada circundante en el SDU de un UART receptor. Tiempo atrás un bit de comienzo es detectado, los bits sucesivos son cambiados dentro del registro de cambio del receptor. Después, el byte es movido dentro de un buffer FIFO. Cada UART moderno tiene un FIFO, pero varia el tamaño de 1 a 5 bytes. Ver figura 1.2.5-3

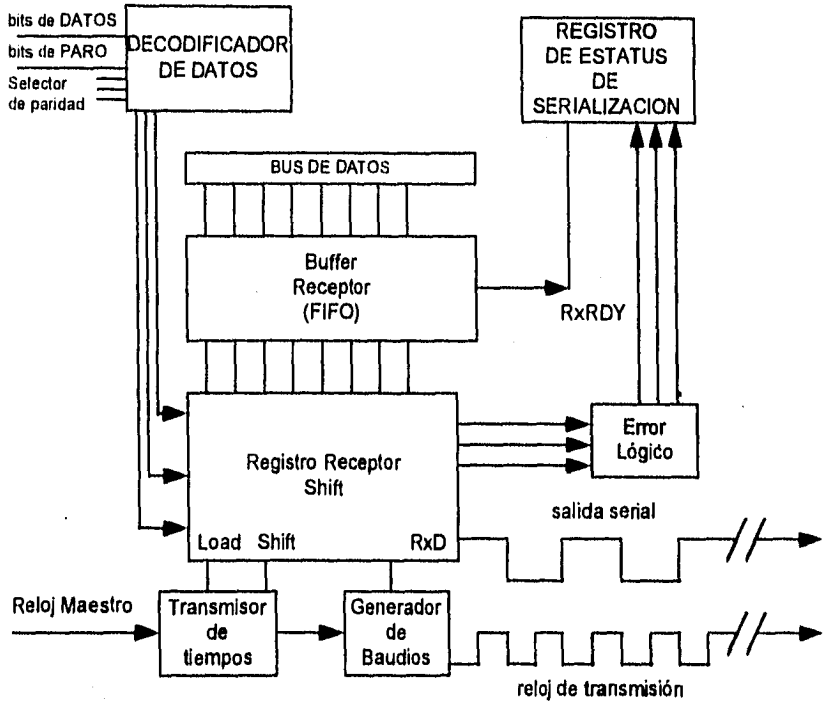


Fig. 1.2.5-3 Interfaz Digital Balanceada.

Status del Receptor.

Cuando un byte ha sido movido dentro de la cola del receptor (FIFO), la bándera RxRDY es puesta en verdadero y permanece en verdadero hasta que todos los elementos de la cola (FIFO) esta vacía. Para cada elemento en la cola (FIFO), hay un registro correspondiente de estado de serialización.

Errores.

El UART no forza la respuesta de un error. En otro tiempo un bit de comienzo es detectado, el UART obliga completamente montar un byte igual en la faz de cataclismo de errores. Esto es, ninguna prevención al software de lectura de resultado de un byte erróneo. Aunque el software no tiene la obligación de prueba de errores, los UARTs proveen inevitablemente un mecanismo para reportarlos. La responsabilidad para probar y manejar errores, sin embargo, siempre es vendido con el software. El término de "error de serialización" aplica a errores en transmisión y recepción de datos.

Errores de Transmisión.

Como puede ser supuesto, pocos errores son posibles en el proceso de transmisión. El resultado de escribir dentro de un buffer de transmisión no vacío es un error de sobreescritura (algunas veces referido como un error de transmisión de ***UnderRun***)

Errores de Recepción.

A diferencia de los procesos de transmisión, un número de problemas pueden presentarse durante la recepción.

- Cuando los bytes llegan más rápido de lo que pueden ser leídos, son puestos en el buffer (FIFO) del receptor: Cuando este buffer se encuentra lleno, este los "sobreescribe", cada nuevo byte sobreescrito en uno anterior y aún no leídos están en el buffer. Esto provoca un error de OverRun.

- Si los valores del bit de paridad no están de acuerdo con el decodificador del programa de usuario en el registro de formato de datos, un error de paridad es reportado.

- Si un bit de Stop invalido es recibido y si todos los bits de datos y el bit de paridad es cero, la línea serial es asumida para ser Falso para un tiempo igual a un SDU, y un Break es reportado.

La transmisión y recepción de datos son operaciones idénticas en velocidad de transmisión y formato de datos. Lo que provoca errores del mismo tipo.

1.3 Modems.

1.3.1 ¿Qué es un Modem y Como Funciona Este?.

El modem es un dispositivo para convertir los pulsos eléctricos ON/OFF generados por una computadora en sonidos audibles, y luego reconvertirlos en impulsos eléctricos ON/OFF al otro extremo de la línea. La información digital utilizada por la computadora es convertida por el modem en señales analógicas para su transmisión por las líneas telefónicas ordinarias. Una vez transmitida la información analógica, esta es convertida de nuevo en información digital. De hecho, la palabra modem es una contracción de modulación/demodulación, términos técnicos usados para describir como trabaja un modem.

Cuando se está transmitiendo información, un modem convierte (o modula) las señales digitales que vienen de la PC en señales analógicas que resultan compatibles con los sistemas analógicos de telefonía. Cuando se esta recibiendo información, el mismo modem convierte (o demodula) las señales analógicas provenientes de la línea telefónica a la que se conectó una computadora remota, en señales digitales cuyo formato es requerido por nuestra PC.

Características de un Modem.

Las características de un modem de llamada pueden ser muy diversas. Los modems de bajo costo tienden a ofrecer menos prestaciones que los modelos de mayor precio.

Todos los modems incluyen componentes comunes, como un transmisor y un receptor. El transmisor modula la señal digital a analógica (tonos y sonido), y el receptor demodula la señal analógica recibida y la convierte de nuevo en digital.

La figura 1.3.1.-1 muestra ambos tipos de señales: analógicas y digitales.

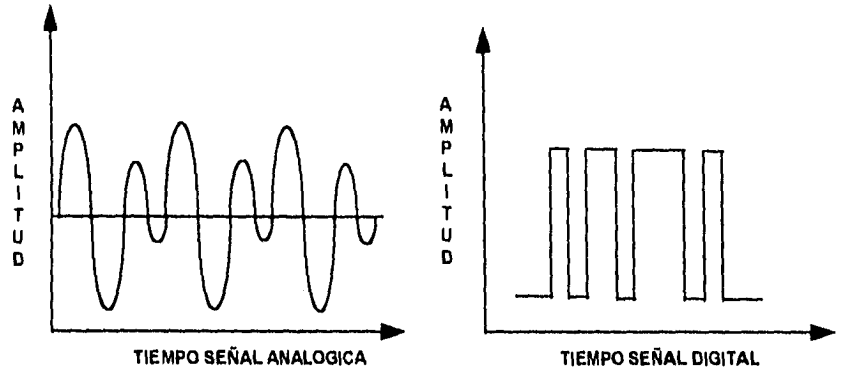


Fig. 1.3.1-1 Señales Analógicas y Digitales.

Señal Portadora y Onda Senoidal.

Cuando dos modems se comunican, intercambian tonos audibles continuos denominados señales portadoras. Cada señal portadora tiene una frecuencia establecida por los fabricantes de modems o un estándar publicado. Si un modem detecta la ausencia de portadora durante un intervalo superior a unos pocos milisegundos, interrumpe la conexión (o el modem cuelga). Es como una conversación entre dos personas; si Juan no escucha nada procedente del otro extremo de la línea (ruido de fondo, música respiración), cuelga. El modem de Juan envía un mensaje de "Portadora Perdida" al usuario.

Las señales portadoras son generadas como ondas senoidales, como veremos en la figura 1.3.1. -2. Las ondas senoidales comienzan con un voltaje cero y suben hasta llegar a un cierto valor positivo, luego vuelven a cero, luego al mismo valor pero negativo y luego a cero. Cuanto más ciclos se produzcan en una unidad de tiempo, mayor será la frecuencia de la señal.

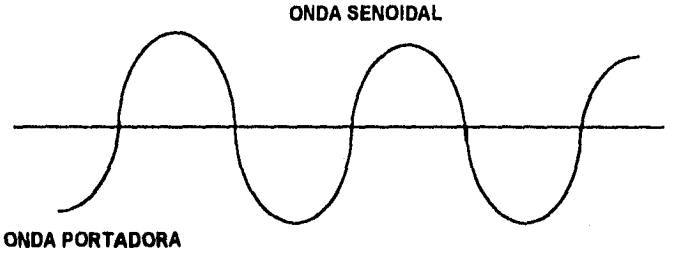


Fig. 1.3.1-2. Representación de una Señal Senoidal.

Transmisión Mediante Modem.

El sistema de numeración binario utiliza 1's y 0's, y permite expresar cada número, letra y símbolo como una secuencia especial de estos dos dígitos. Cada grupo de ocho bits (1's y 0's) representa un byte o carácter. Generalmente, dentro de una computadora la transferencia de información se realiza mediante un bus paralelo. Combinando ocho canales de dos estados, obtenemos un total de 256 posibles combinaciones.

El objetivo de un modem es permitirnos conectar nuestra PC. con otras computadoras utilizando líneas telefónicas estandar. Para poder utilizar las líneas telefónicas, el modem requiere de convertir o trabajar con las señales que maneja, convierte las señales digitales generadas dentro de la PC. en señales analógicas, las cuales son transmitidas a través de las líneas telefónicas

estándar. La necesidad de transformar de un tipo de señal a otro se fundamenta en el hecho de que los teléfonos son dispositivos analógicos, lo que significa que para funcionar correctamente, manejan ondas de sonido de forma continua. La computadora maneja por otro lado señales digitales, las cuales son el resultado de los millones de señales que se generan durante cualquier procesamiento de datos. Estas señales se pueden generalizar a dos diferentes tipos de archivos:

- ASCII
- Binario

Básicamente, un archivo ASCII es aquel en el cual, todos los caracteres son representados por números decimales del 0 al 128. (Por ejemplo la A mayúscula equivale al valor 65 en ASCII). La razón de que se manejen 128 valores es porque 2 elevado a la 7 potencia es igual a 128 (2^7=128), por lo que en realidad es un protocolo de 7 bits. Los valores manejados son los de letras minúsculas y mayúsculas (A a la Z y a a la z), caracteres numéricos, signos de puntuación y unos pocos códigos de formatéo.

Los archivos binarios consisten en extensiones de datos de 8 bits, en los cuales cada bit representa el dígito binario 0 o 1, lo que da lugar a 256 posibles combinaciones. Esto aumenta al doble la cantidad de caracteres que se pueden transmitir. Virtualmente todos los programas ejecutables pueden ser transferidos usando el protocolo binario.

Transmisión de Información Digital Usando Modulación.

Para representar información digital, se necesitan como mínimo dos estados. Estos estados se representan por la alteración de la señal portadora para

representar el dígito binario 0 y/o el dígito binario 1. La modificación de la señal portadora se denomina modulación.

La modulación puede emplear la variación de un grupo cualquiera de estos atributos de la portadora:

Amplitud	Magnitud o nivel de voltaje de pico a pico.
Frecuencia	Número de oscilaciones completas de la señal por unidad de tiempo.
Fase	Posición en que la señal pasa por cero, relativa a la señal anterior.

Los modems utilizan distintas formas de modulación, dependiendo de la velocidad correspondiente. Por ejemplo:

Modulación por Desplazamiento de Frecuencia (FSK). Se utiliza para velocidades inferiores a los 19200 bps. La modulación FSM es una técnica en dos niveles que representa los cambios en el patrón binario de bits mediante cambios en la frecuencia de tonos de audio. Se supone que la línea está en reposo con un valor 1, representado por un tono de una frecuencia determinada. El modem cambia a un tono de otra frecuencia cuando se envía un bit de datos 0 (véase figura 1.3.1-3). Estos cambios de tono causan un efecto musical durante la transmisión.

Modulación por Desplazamiento de Fase (PSK). Esta modifica la fase de una señal, es decir, su sincronización respecto a una referencia fija, para representar cambios en el patrón de bits. Para medir el desplazamiento de fase de la señal recibida y determinar si es 0 o 1, se utiliza un oscilador de referencia.

Modulación por Desplazamiento de Fase Diferencial (DPSK). Se utiliza en modems de 1200 a 19200 bps para PC, y compara el ángulo de fase de la señal recibida con la señal recibida anteriormente. Un cambio de fase se interpreta como un 0 binario, si la fase anterior se interpretaba como 1, y así sucesivamente. Este método no requiere una señal de referencia, y por lo tanto necesita menos circuitería.

Modulación de Amplitud (AM). Es la técnica de modulación más sencilla. Las ondas de amplitud grande se asignan al 1 binario, y las ondas de amplitud pequeña al 0 binario. La AM es muy susceptible a las interferencias de las líneas y en la práctica no se utiliza de forma aislada.

Ancho de Banda.

El ancho de banda es la capacidad de transporte de información de un servicio de transmisión. El ancho de banda define un rango de frecuencias, medido en Hertz (ciclos por segundo), que pueden alcanzarse sin una degradación significativa de la señal. Cuanto mayor sea el rango de frecuencias del modem, mayor será su capacidad para transportar datos. La mayoría de modems utilizan un rango de frecuencias de 300 a 3000 Hz, en el centro del ancho de banda de las líneas telefónicas.

Podemos ver el ancho de banda como la fidelidad. En la radio, las notas muy agudas (altas) y muy graves (bajas) no se reproducen muy bien. En telecomunicaciones, por término medio, las líneas telefónicas no son muy estables en frecuencias altas o bajas. El modem está limitado al centro del ancho de banda, que es la zona más clara y con mayor capacidad de reproducir con exactitud la modulación.

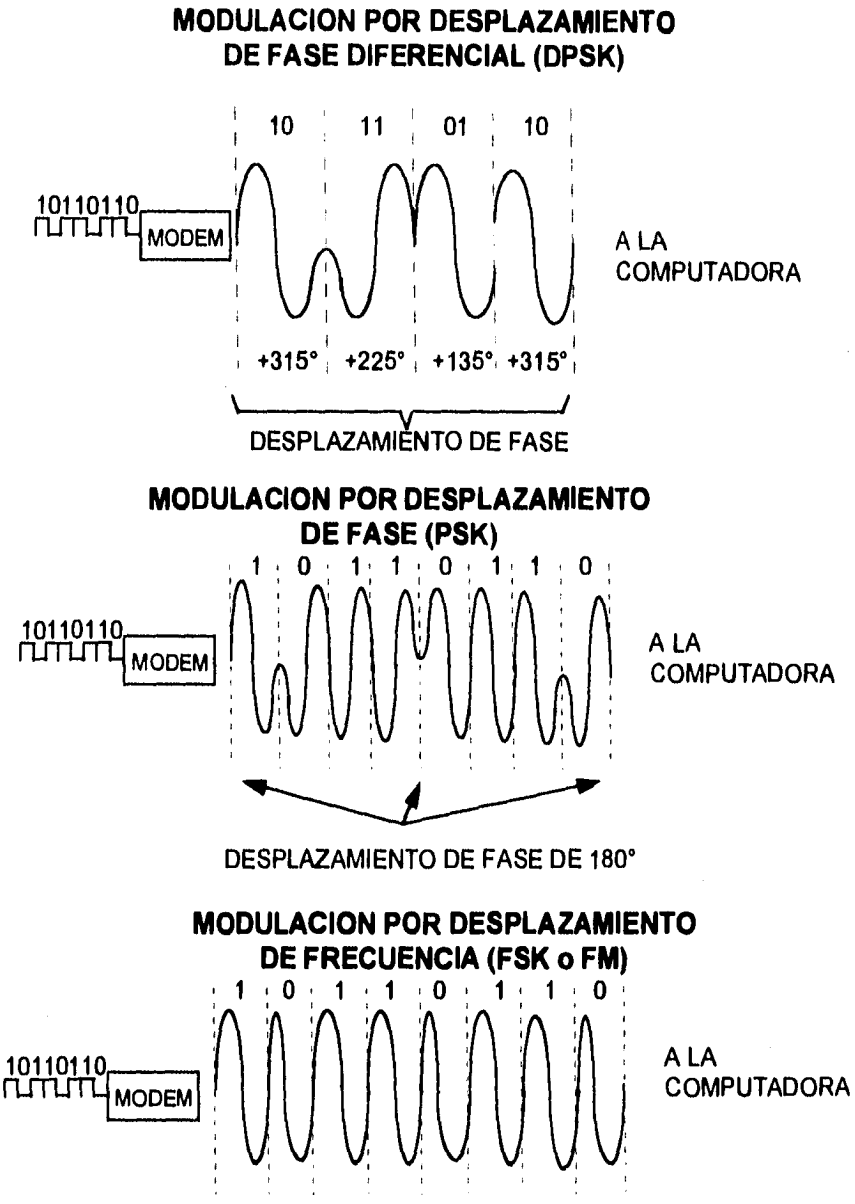


Fig. 1.3.1-3 Diferentes Tipos de Modulación.

Para moverse dentro de los límites del ancho de banda, a menudo los modems utilizan algoritmos sofisticados de codificación de varios bits para comprimir datos todo lo posible en ambas direcciones. Desafortunadamente, la codificación de varios bits también incrementa la pérdida de datos durante los fallos en la línea (como la presencia de electricidad estática o el cruce de modulaciones). El objetivo perseguido en el diseño de modems efectivos es minimizar las perdidas de datos al enviar grandes cantidades de información por un enlace de comunicaciones.

Características de Transmisión: BPS y Baudios.

La primera confusión que aparece en telecomunicaciones es la curiosa combinación de dos palabras: velocidad de transmisión. La velocidad de transmisión, indicada generalmente en bits por segundo, es el término correcto al que nos referimos como "velocidad". Los modems trabajan a una velocidad de transmisión fija, o la velocidad del dispositivo emisor (aunque está restringida a un rango especifico). Algunos modems vienen equipados para trabajar a varias velocidades distintas. La velocidad de transmisión se controla con la configuración de los interruptores de modem, o por ordenes a través de software. La velocidad de transmisión aparece a menudo aunque erróneamente, denominada como la velocidad en baudios.

"Velocidad en baudios" y "velocidad de transmisión" no significan lo mismo aunque a menudo se utilizan como sinónimos. Entonces, ¿donde está la diferencia?. El término "bps" expresa la velocidad de cambios en la señal; es una medida de la velocidad de modulación. Una línea de voz puede soportar 2 400 cambios de señal por segundo (baudios). Los modems de alta velocidad

Bits de Datos y Bits de Paro.

Los bits de datos y paro definen como se organiza y transmite datos durante una sesión en línea. Los bits de datos usados durante una sesión en línea especifican el número de bits individuales que son usados para indicar un carácter sencillo. Esta es la referencia común para el largo de una palabra. La mayoría de los programas comerciales manejan palabras de 7 u 8 bits.

Cuando las computadoras manejan palabras de 7 bits, se encuentran limitadas a mandar letras de texto, números y los signos de puntuación más comunes. Si se incrementa la longitud de la palabra a 8 bits, se permite la transmisión de datos binarios - por ejemplo, archivos de programas ejecutables (COM y EXE). Muchos **mainframe** están limitados para comunicarse en palabras de 7 bits. En el mundo de las PC's, las palabras de 8 bits prevalecen.

Los bits de paro son usados para indicar el fin de un caracter compuesto de múltiples bits de datos. Ellos dicen al modem en la parte final: "Esto es el fin de la actual secuencia de bits. Continua y muestra en pantalla este caracter". O bien la otra alternativa es copiar esto a un disco de archivo. Los bits de paro más comunes son el 1 y el 2.

Paridad.

Para asegurarse que un caracter es recibido correctamente, se añade un bit inmediatamente después de los bits de datos; a este se le denomina bit de paridad. No siempre se usa.

Cuando se usa, la paridad puede ser par o impar. La paridad se verifica anotando el número de 1's incluidos en los bits de datos (los bits de inicio y de parada no se tienen en cuenta).

Si se utiliza paridad par, el caracter tendrá que tener un número par de 1's entre los bits de inicio y final. Si hay un número impar de unos en los bits de datos, el bit de paridad se pone a 1 para que el número de 1's sea par. Si el número de bits ya es par, el bit de paridad será 0, para mantener el número de unos.

Si utilizamos paridad impar, el objetivo es tener un número impar de 1's entre los bits de inicio y parada. El bit de paridad añadido será 1 o 0, según sea necesario, para tener un número impar de 1's en los datos.

Cuando se recibe un caracter, se cuenta el número de 1's y se compara con el bit de paridad, verificando así si algunos datos han cambiado durante la transmisión. Si la paridad no coincide, se indica el error y se puede producir la retransmisión de los datos.

En resumen, para enviar ocho bits de datos, hay, que enviar 11 bits de información (ocho bits de datos, un bit de inicio, un bit de paridad y un bit de parada). Esto implica disminuir en un tercio la capacidad de enviar información.

1.3.2. Tipos de Modem que Existen Actualmente (Características, Ventajas y Desventajas).

Existen cientos de diferentes modelos de modems. Cada uno ofrece una gran variedad de velocidades, estándares, características y estilos.

Tipos de Modems.

Los modems vienen en dos modelos básicos:

- Modems Internos
- Modems Externos.

Modems Internos.

Un modem interno es instalado dentro de la PC insertándolo dentro de un slot de expansión. Estos están alineados por lo general en la parte trasera de las tarjetas madre y son usadas para conectar una gran variedad de tarjetas especializadas como por ejemplo de memoria extra, controladores de disco, controladores de video y mas como se ilustra en la figura 3.1.2-1.

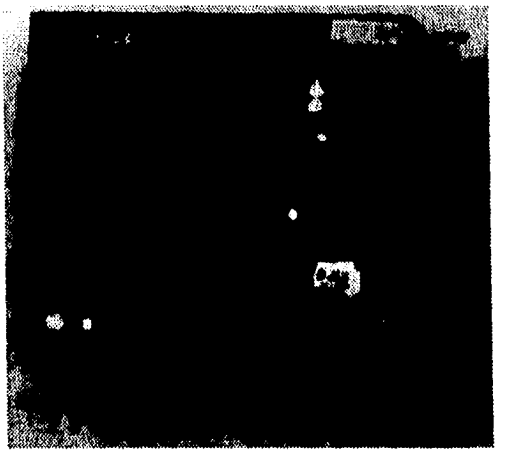


Fig. 1.3.2. - 1 Modem interno.

Ventajas y Desventajas de los Modems Internos.

Esto ahorra espacio del **desktop** y reduce el enredo de cables fuera de la PC. Si se tiene cuatro o menos slots, se debe de considerar una unidad externa. Además, desde que los modems internos se instalan dentro de la PC ellos roban potencia a la fuente de poder e incrementan la temperatura dentro de la PC Esto puede no ser un problema, pero algunas PC's viejas tienen problemas con las fuentes de poder y los sistemas de enfriamiento.

Todos los modems necesitan un puerto RS-232-C para comunicarse con otra PC. En un modem interno el puerto RS-232-C forma parte de la tarjeta madre. Algunos modems internos permiten seleccionar los puertos COM1 o COM2, mientras que otros permiten direccionar el modem al COM3 o COM4. Si se tiene dos o mas dispositivos seriales como un mouse y una impresora serial, esto puede ser una importante ventaja. Usualmente el ratón y la impresora están direccionadas a los puertos 1 o 2.

Debido a que se instalan dentro de tu PC, los modems internos no son portátiles. Esto tiene en si mismo ventajas y desventajas. Para los modems internos es muy poco probable que salgan del escritorio de trabajo lo cual puede ser un factor de seguridad para los manejadores de recursos. Pero en contrapartida se pierde flexibilidad porque no se puede usar más de un modem interno en una PC, y cambiarlo requiere remover la cubierta de la PC. lo cual ya es más complicado. Tal vez la mayor deficiencia en contra de los modems internos es que no tienen un patrón de luces o indicadores de estado. La mayoría (pero no todos) de los modems externos tienen muchas luces en el panel frontal del modem. La típicas luces indicadora señalan **carrier detect, phone line off hock, received data and transmitted data**. Si se llega a experimentar un problema

con el modem, las luces de estado pueden ser una valiosa herramienta de ayuda. Los modems internos nos tienen estas luces o indicadores de estado, lo cual hace más difícil localizar una falla si esta llega a ocurrir. Finalmente, los modems internos cuestan menos que sus contrapartes externas.

Modems Externos.

Un modem externo es por otra parte, una pieza separada del equipo, como la que se puede observar en la figura 1.3.2-2.

Como estos modems no se conectan directamente a la tarjeta madre, un modem externo puede conectarse a la PC a través de un puerto serial estándar. Un modem externo se coloca sobre el escritorio y se conecta al puerto serial de la PC por un cable RS-232-C. Requiere de una fuente de corriente además de un cable conectado a un enchufe de corriente alterna que se enredara atrás de tu PC. La mayoría de los modems son diseñados para colocarse abajo de un teléfono, y esto puede ser o no ser útil y funcional para nosotros. Por ejemplo, el **Hayes Smartmodem** fue diseñado para colocarse debajo de los teléfonos **Bell 500D** o del **2500D**. Desafortunadamente, la mayoría de los teléfonos modernos son un poco mas grandes que esos antiguos teléfonos de la Bell y otros nos se ajustan a las unidades de la Hayes. Las unidades **U.S. Robotics** son más grandes comparadas con las unidades Hayes y casi cualquier teléfono puede colocarse en la parte superior de los Robotics sin ningún problema.

Si se tienen muchas PC's cerca, se puede compartir un modem externo entre las distintas PC's usando un multiplexor para RS-232C. Esto funciona bien si los usuarios solo accesan el modem en contadas ocasiones para compartir la

línea. Si se tiene una LAN se puede usar un LAN communications server para compartir modems en la red.

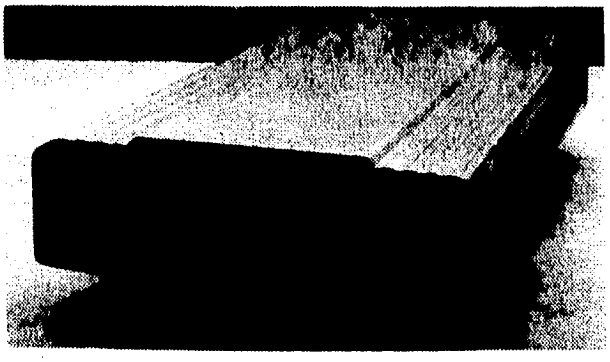


Fig. 1.3.2-2 Modem Externo.

Como un modem externo incluye su cubierta y su fuente de poder, su precio es un poco más elevado que su contraparte interna. Los modems externos deben ser conectados a un puerto serial en la PC por lo que se tiene que considerar el costo del puerto serial y del cable RS-232-C . La mayoría de las computadoras vienen por lo menos, con un puerto serial pero mucha gente usa el puerto empotrado para la conexión del mouse. Si se diera el caso de que tu tuvieras un mouse con conexión al puerto serial, sería necesario instalar un puerto serial adicional a la PC.

1.3.3. Modems Compatibles con las Centrales Telefónicas.

Las líneas telefónicas son cables, y todos los cables tienen resistencia. Los cables permiten circular a los electrones de un punto a otro. La resistencia de los cables se opone al flujo de electrones.
La impedancia es una propiedad física inherente del cable. Su valor depende del ancho del cable respecto de la corriente que circula y la longitud del cable. Al extremo de los cables, en la central telefónica, hay circuitos y conmutadores eléctricos diseñados para permitir la transmisión en estás líneas telefónicas, pero siempre con posibilidades limitadas.

Decibelios.

Uno de los términos usados con más frecuencia en la industria de las telecomunicaciones es el decibelio (DB). El decibelio es una unidad para medir la potencia relativa del sonido, o para expresar la relación de dos magnitudes de potencia acústica o eléctrica. Los decibelios pueden referirse a distintas unidades de medidas; sin embargo, pueden convertirse a voltajes (voltios y milivoltios). La transmisión y recepción de datos se produce a un cierto nivel de DB. El ruido de fondo se mide en DB, así como su respuesta en frecuencia.

Los decibelios fueron diseñados, originalmente, para determinar la tasa de cambio del sonido. En comunicaciones de datos, este uso no es tan consistente. Actualmente, el DB se usa a menudo como punto de referencia (o nivel de voltaje). Sin embargo, como el DB es el lenguaje de la compañía telefónica, estamos obligados a adoptarlo. Para la compañía telefónica, una señal de 2.2 voltios pico a pico será el punto cero de referencia de nuestra aplicación. Representa la máxima señal permitida que puede o debe ser

aplicada a una línea telefónica. A partir de este punto, todos los niveles de tensión serán inferiores. Sin embargo, como nuestro tope está en cero, representamos estos pequeños voltajes - DB en decibelios negativos.

Podemos decir que las perdidas de la línea incluyen la impedancia del "cableado doméstico", añadida a la distancia del conector telefónico de la casa hasta la central telefónica más cercana. El **fixed loss loop** (**FLL**; bucle de perdida fija) es un factor predeterminado creado por la compañía telefónica modificando las pérdidas existentes en la línea y sumando o restando impedancia. De esta forma, se asegura de que la portadora de transmisión del modem llega a la central telefónica a un nivel aceptable.

Las distintas centrales de las empresas telefónicas pueden aceptar distintos estándares. Sin embargo, usando un nivel de transmisión predefinido por el modem, cualquier compañía telefónica puede ajustar el FLL para adaptarse a sus necesidades. Normalmente, el FLL se ajusta de modo que si nuestro modem transmite a -9 DB, la señal llegará correctamente a la central telefónica. Si nuestro modem transmite a un nivel más alto (digamos -8 DB), podemos causar interferencias con otras señales, molestando a la compañía telefónica con una señal demasiado fuerte. Si transmitimos a un nivel inferior (-13 DB o -15 DB), nos arriesgamos a que la señal sea demasiado débil como para ser recibida. La mayoría de modems de entorno PC están diseñados para transmitir -9 DB. Existen muy pocos problemas a este nivel.

La impedancia de las líneas puede cambiar por su longitud (distancia), por su diámetro (ancho del cable), por la temperatura y otras influencias externas (lluvia o pájaros posados en la línea). Si transmitimos en el rango de los -9 DB, ¿a qué nivel recibiremos las señales que nos envíen?. Legalmente, la compañía

telefónica está obligada a garantizar una perdida menor de 16 DB del transmisor al receptor. Por lo tanto, si transmitimos a -9DB, podemos esperar que el modem remoto reciba la señal a -25 DB.

En general, el canal de comunicación que usamos para conectarse con la central es de cable telefónico, por lo que, si el modem cumple con las características descritas en el punto anterior, no habrá ningún problema para enlazarse con la central.

Cada modem o dispositivo similar diseñado para conectarse a una línea telefónica tiene que cumplir los estándares de la FCC. Una de estas normas controla la carga introducida en la línea por el dispositivo. La mayoría de modems y dispositivos relacionados utilizan un transformador no solo para balancear la carga, sino para evitar la circulación de voltajes peligrosos en ambos sentidos.

Velocidades de Transferencia de Series de Datos.

En las comunicaciones seriales, la transferencia de datos es medida en bits por segundo (*bps*). Por otro lado, los promedios de señalización del *Modem*, son medidos en baudios. El promedio de baudios de un modem describe el número total de *eventos de señalización binaria* que se producen cada segundo para transmitir datos binarios a través de una línea telefónica como si fueran sonidos. Este promedio de baudios define la duración del voltaje de la señal que utiliza el modem para representar los eventos de señalización binarias. La siguiente fórmula es una definición matemática exacta del término:

Promedio en Baudios = 1 / Duración de la señal de un Bit.

Por el decrecimiento de la duración de un evento de señalización binaria, algunas veces llamada *tiempo de bit*, es posible incrementar el promedio en baudios. De la otra forma, por el incremento de la duración del evento de señalización binaria, es posible decrementar el promedio en baudios.

Las relaciones entre baudios y bits por segundo dependen de la técnica que el dispositivo de comunicación utilice para codificar los datos. Para comunicaciones de baja-velocidad, el promedio en baudios puede ser igual al promedio de transferencia de datos en bits. En comunicaciones de alta velocidad, se requiere de dispositivos que sobrepongan muchas señales de datos binarios en cada evento de señalización binaria. Esto permite que la transferencia de datos dado en bits exceda el promedio de baudios. Para comunicaciones abajo de los 600 baudios, el promedio de baudios y el promedio de bits son iguales. Para comunicaciones de 600 baudios o más, el promedio de baudios y el promedio de bits son diferentes.

1.3.4. Líneas Privadas y Líneas Conmutadas.

Redes de Comunicaciones de Datos.

Si sólo dos computadoras se quieren conectar y están en el mismo cuarto u oficina, entonces la transmisión se facilita a una conexión punto a punto a través de un cable. Ahora que si están localizadas en diferentes partes de una ciudad, se deben de usar las de redes publicas de comunicaciones. Por lo general, esto significa usar la red de teléfonos públicos conmutados (PSTN) para la cual se requiere de un modem para transmitir los datos. El arreglo general se muestra en la figura I.3.4-1.

Circuitos Públicos.

Cuando los datos se van a transmitir entre dos DTE's en el mismo edificio, es relativamente fácil llevar a cabo la instalación por cable. Típicamente, este puede ser un par blindado o sin blindar, cable coaxial o fibra óptica. En algunos casos, se puede utilizar el radio. Cuando los datos van a ser transmitidos entre dos diferentes oficinas y diferentes edificios, esto solamente puede llevarse a cabo por el uso de microondas o bien mediante un enlace vía satélite, o a través de las líneas de una de las compañías telefónicas. La última solución es muy utilizada; de hecho puede ser a través de líneas de circuitos conmutados o mediante líneas privadas o rentadas.

Los circuitos conmutados pueden utilizarse a través de la red de teléfonos públicos conmutados (PSTN) o mediante una red digital de servicios integrados (ISDN), dependiendo de la disponibilidad y de los recursos que se tengan entre otras muchas cosas. A pesar de que la red analógica PSTN fue diseñada específicamente para comunicaciones de voz, es posible transmitir datos usando modem. En el caso de un ISDN, las llamadas pueden alcanzar un promedio de bits mucho mayor al promedio de una PSTN.

En el caso de las líneas privadas, en algunos casos es necesario rentar líneas de la PSTN - y por lo tanto modems - en la mayoría de los casos, las líneas privadas son ahora totalmente digitales.

Circuitos Analógicos PSTN.

Cuando los datos se van a transmitir a través de las líneas de transmisión PSTN existentes, es necesario el convertir las señales de salida eléctricas de la fuente del DTE en una forma que sea posible transmitirlas por el PSTN. Este fue

diseñada para transmitir voz por lo cual se asumió que era necesario considerar una mezcla de frecuencias que van de los 400 a los 3400 Hz. como se muestra en la figura I.3.4-2.

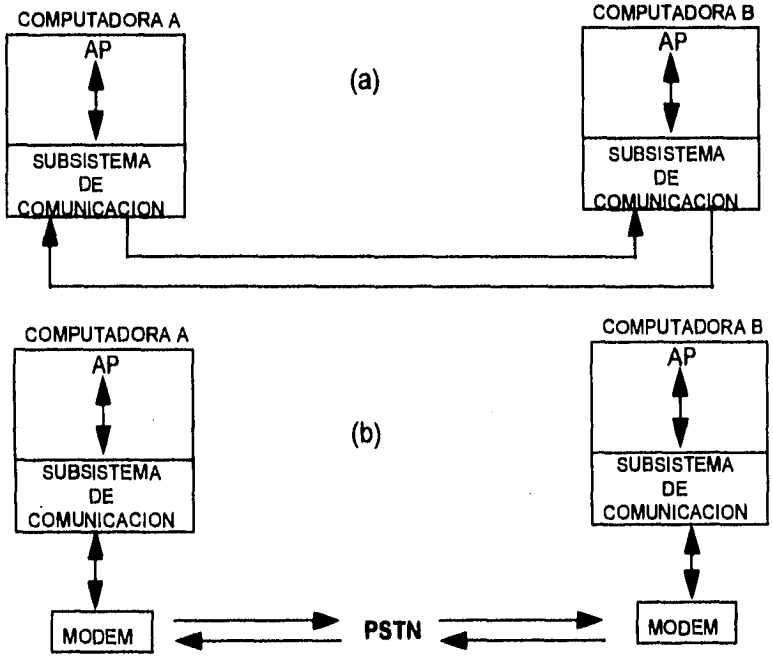


Fig. I.3.4-1 Alternativas de Conexión de Computadora a Computadora. (a) Cable de Conexión Punto a Punto. (b) PSTN + Modem.

El rango de la frecuencia de la señal de un circuito es conocido como ancho de banda. Por lo tanto, el PSTN se dice que tiene un ancho de banda de 400 a 3400 Hz o simplemente de 3000 Hz. Todo esto significa que una línea telefónica no permite transmitir frecuencias muy bajas, por ejemplo, las cadenas de datos que se van a transmitir están hechas de 1's y 0's binarios. Por esta razón, la

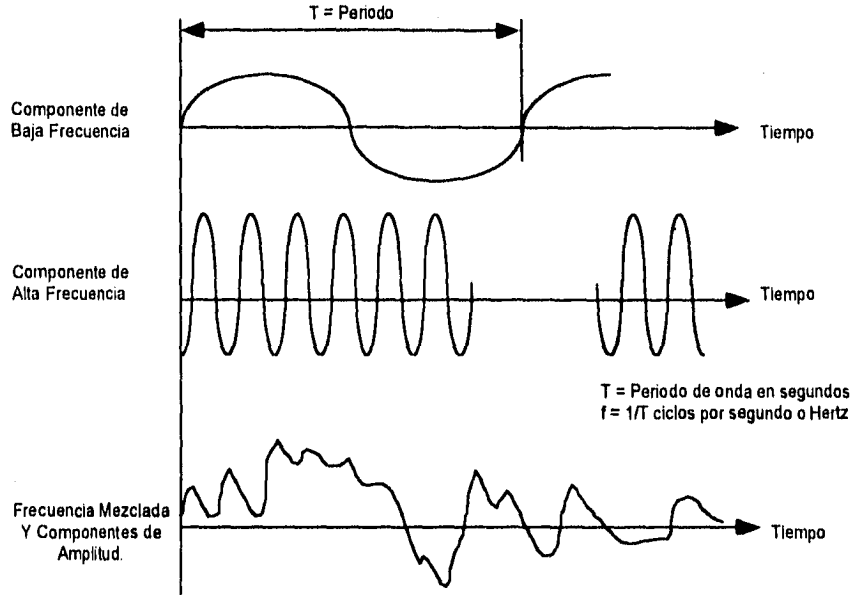


Fig. I.3.4-2. Componentes de Frecuencia de Ondas de Voz.

transmisión de datos no es una cosa tan simple como aplicar dos niveles de voltaje a la línea telefónica desde que la salida de un voltaje puede ser cero para ambos niveles si la transmisión de los datos puede ser de puros unos o de puros ceros. Como sea, es necesario el convertir los datos binarios con una forma compatible con una señal de voz que se manda al extremo de la línea y posteriormente reconvertir esta señal nuevamente a su forma binaria para el receptor. El circuito que ejecuta la primera operación se conoce como modulador, y el circuito que ejecuta la función inversa se conoce como demodulador. Como a cada extremo del cable normalmente se manda y se reciben datos, se necesita de un dispositivo que efectúe ambas operaciones, este dispositivo es el modem del cual ya hablamos anteriormente. Usando modems, los datos pueden ser transmitidos a través de PSTN ya sea por un marcaje normal o por medio de la colocación de una línea conmutada a

través de la red, como una llamada normal a través de un teléfono, o bien por la renta de una línea privada de PTT. Desde que las líneas privadas dejaron a un lado el equipo de conmutación normal en la red y están permanente conectadas, estas solo son justificables económicamente para aplicaciones que tienen un factor de utilización considerablemente elevado. Una ventaja de una línea privada es que por las características de su operación los datos pueden ser cuantificados de manera más precisa que en una línea de red de teléfonos públicos estándar, lo que permite transmitir a velocidades mucho mayores.

Para explicar esto con mayor claridad la modulación y la demodulación son funciones que se realizan por separado. La modulación es un tema que ya se trató anteriormente por lo que pasaremos al tema de la demodulación.

Demodulación.

Para entender como las señales moduladas mostradas en la figura I.3.4-3. son demoduladas es necesario entender algunas propiedades básicas de este tipo de señales. Esta fuera del alcance de este trabajo el deliberar sobre las complejas expresiones matemáticas que producen estas formas de ondas por los diferentes métodos de modulación y los diferentes procesos de demodulación. Algunos de estos efectos se muestran en la figura I.3.4-5. Los siguientes puntos son interpretados de esta figura:

Cuando una señal portadora de frecuencia fija f_c es modulada por una segunda señal fija f_m, un número adicional de componentes de la frecuencia conocidos como límites de banda aparecen la figura 1.3.4-5.(a).

1. Con AM sólo dos límites de banda se producen, en $f_c + f_m$ y $f_c - f_m$, cada una contiene una fracción de la potencia que contiene la portadora. Nótese que los límites de banda son los que contienen la información requerida f_m. Con ambos límites de banda FM y PM muchos límites de banda son producidos como múltiplos de FM de la portadora ($f_c + f_m$, f_c+2f_m, etc.) cuyas amplitudes son calculadas usando las funciones Bessel.

2. Usando un técnica matemática conocida como análisis de Fourier, se sabe que una onda cuadrada - equivalente a una cadena de datos binarios de 1's y 0's - esta hecha de un número infinito de componentes de frecuencia senoidal. Estos forman una frecuencia principal, f_n, que es igual a un medio del promedio de bits en ciclos por segundo o hertz, y múltiplos de esa frecuencia ($3f_n$, $5f_n$, $7f_n$, etc.) conocidas como armónicas, las amplitudes de estas decrecen con el incremento de la frecuencia principal.

3. Cuando un grupo de datos binarios es transmitido, el patrón de bits cambia continuamente. Por esto, la frecuencia principal (y sus armónicas asociadas) también cambian continuamente. De un extremo el flujo de datos puede ser de ondas cuadradas (equivalentes a una transición de 1's y 0's) mientras al otro extremo puede estar una señal de frecuencia 0 (equivalente a una cadena continua de 1's y 0's).

Se puede concluir por lo tanto que la señal producida después de modular una señal portadora senoidal con corrientes de datos binarios, esta hecha de una portadora más un número infinito de frecuencias que contienen la información requerida. Como la mayor potencia esta dentro de la frecuencia portadora del flujo de bits y los límites de la banda primaria de la señal modulada resultante, es posible en la práctica, determinar la transmisión de información mediante la

detección de los límites de las bandas de frecuencias a cada lado de la portadora y asegurarse que esta banda cubre los extremos primarios producidos por la frecuencia de modulación.

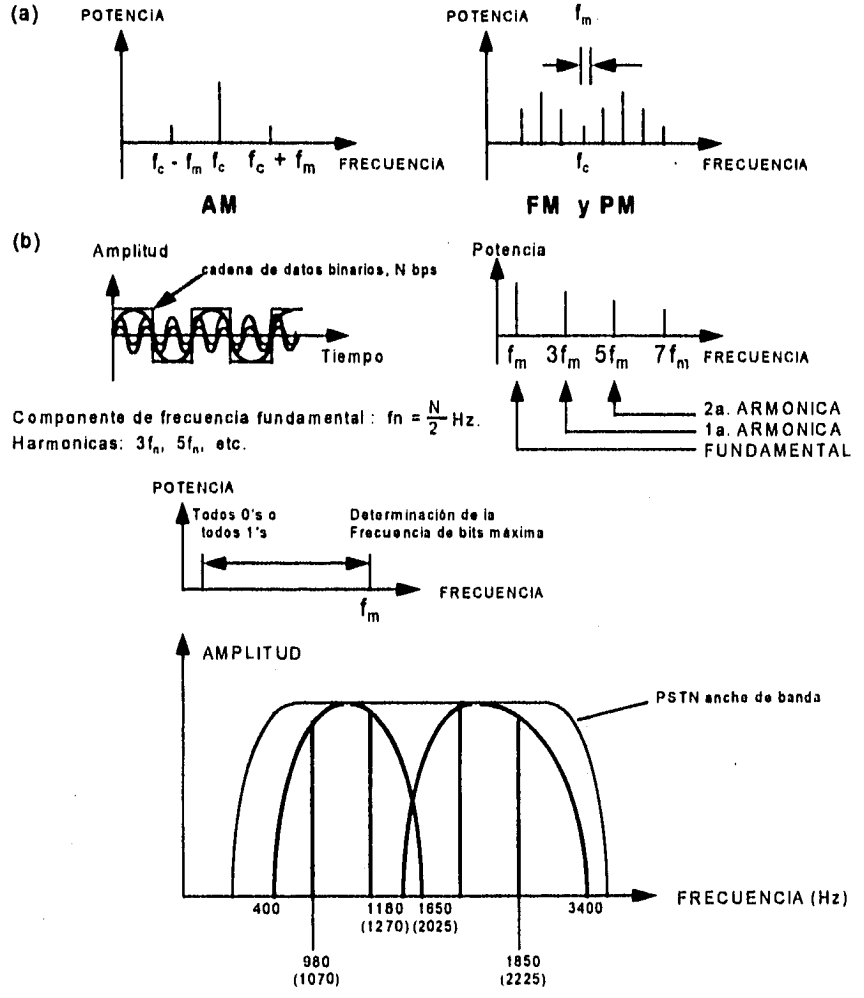


Fig. I.3.4-3. Modulación. (a) Espectro de Potencia; (b) Componentes de Frecuencia de la Onda Cuadrada; (c) Frecuencia Principal de un Conjunto de Datos Binarios.

Por ejemplo una señal modulada, FSK puede considerarse hecha de dos frecuencias portadoras separadas - una para los 0's binarios y otra para los 1's binarios - cada una de las cuales esta en la parte baja y alta de la frecuencia de bits máxima como se muestra en la figura I.4.4 (a). El espectro de frecuencia de este tipo de señal se muestra en la figura I.4.4 (b). Ahora, si el promedio de bits máximo es de 300 bits por segundo, este tiene una frecuencia máxima de 150 Hz. en cada lado de la portadora. Si la frecuencia de separación entre las dos portadoras se selecciona en 200 Hz., esta puede abarcar los límites de banda de cada una de las portadoras. Similarmente, si el promedio de bits máximo es de 1,200 bps, se tiene una frecuencia máxima de 600 Hz. por lo que su frecuencia de separación es del orden de 1000 Hz. Como se puede notar un promedio de 9,600 bps, tiene una frecuencia máxima de 4,800 Hz. lo cual excede el ancho de banda de una línea PSTN. Por lo tanto esta frecuencia de operación no puede ser utilizada con las técnicas de modulación básica.

La figura I.3.4-5 ilustra como un par de canales son obtenidos a partir de un par de cables conectados a una línea PSTN. Los dos canales pueden ser usados para implementar una conexión Full-Duplex de 300 bps entre dos equipos DTE's. Este tipo de modem usa modulaciones FSK con las frecuencias más bajas portándose en una dirección y en la dirección contraria las frecuencias más altas. Las actuales condiciones de transmisión de este tipo de modem varían de un país a otro (y también las redes telefónicas públicas).

Los circuitos digitales privados son utilizados no solamente para conectar dos equipos DTE si no también en la mayoría de las redes de datos privados. Estas redes son utilizadas por organizaciones y empresas que requieren transmitir datos a un gran volumen.

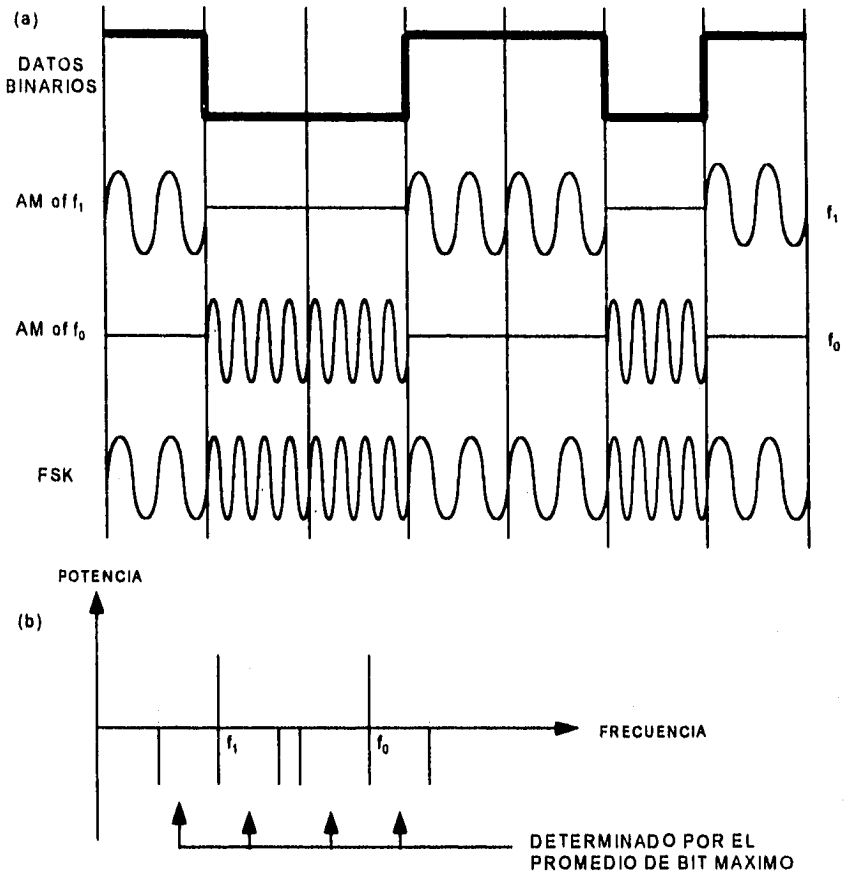


Fig. I.3.4-4. Componentes de Frecuencia FSK: (a) Componentes de la Modulacion FSK; (b) Espectro de Potencia.

Toda la información relativa a llamadas - voz y datos - y que están asociadas con la mayoría de las redes públicas están ahora, transmitiendo en forma digital. Más aún, la forma digital de trabajar está extendiéndose hacia muchos consumidores. La red que resulta de esto es conocida como red digital de servicios integrados ó ISDN donde el usuario puede transmitir datos con voz sin el

uso necesario de un modem. Uno de los resultados de este desarrollo es que ahora es posible rentar circuitos digitales que operan a velocidades que van de los Kbps hasta los Mbps.

Estos circuitos se derivan de aquellos circuitos en red que eran usados para intercambiar información. Cuando usemos estos circuitos es necesario saber como están organizados para estar en la capacidad de aprovecharlos al máximo.

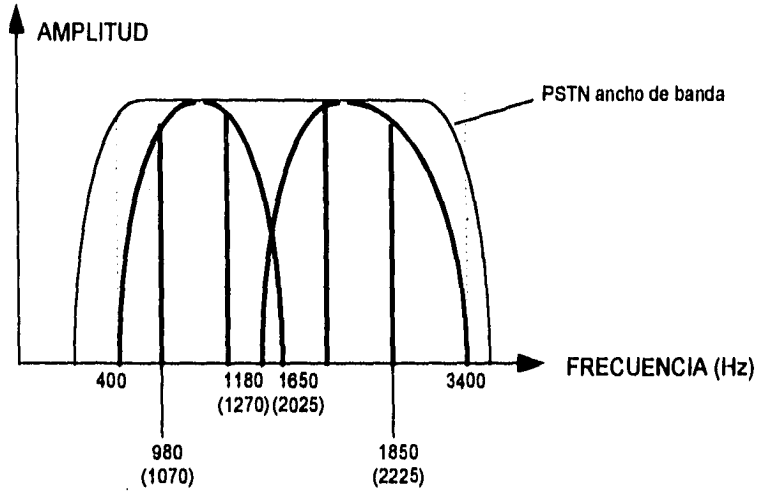


Fig. I.3.4-5. Modem Full-Duplex de 300 bps: (a) Espectro de Frecuencia.

Digitalización.

Se puede concluir del anterior análisis que los circuitos digitales están diseñados para transmitir voz y datos. Como se describió anteriormente, la transmisión en voz está limitada a un ancho de banda máximo y que sea menor a 4 KHz.

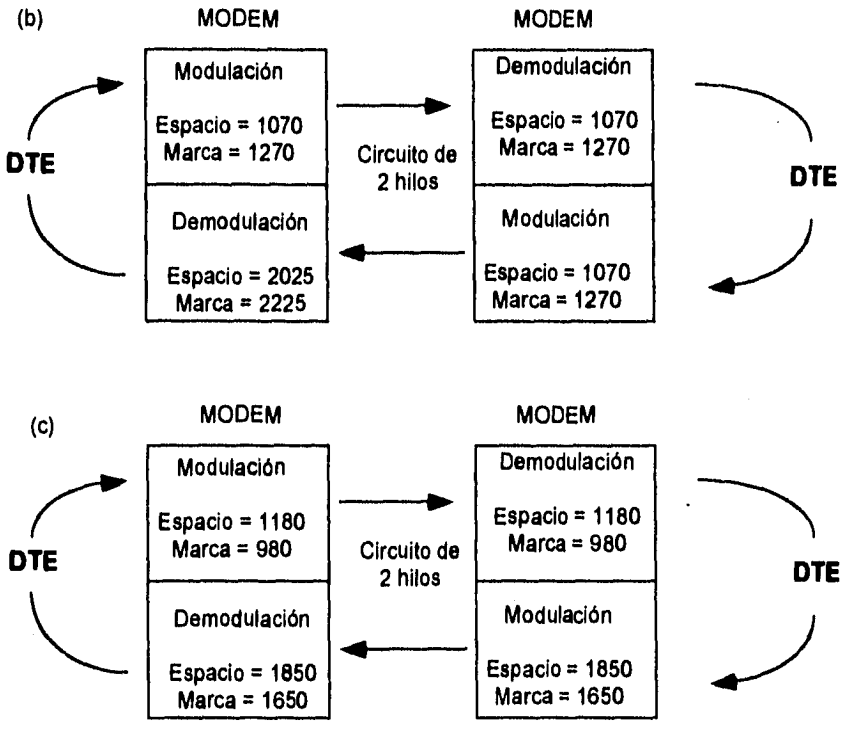


Fig. 1.3.4-5. Modem Full-Duplex de 300 bps: (b) Asignaciones de Frecuencia US (C) Asignaciones de Frecuencia CCITT (Continuación)

Circuitos Digitales Privados.

Para convertir estas señales a su forma digital el teorema de muestreo de Nyquist establece que la amplitud de la onda o señal analógica debe ser muestreada a un mínimo del doble de la frecuencia máxima que tiene la señal analógica. Por lo tanto para convertir una señal de voz de 4 KHz., en una señal digital, esta debe de ser muestreada 8,000 veces en un segundo. El arreglo general de este procedimiento se muestra en la figura 1.3.4 - 6 (a); las partes (b) y (c) de la figura muestran más detalles. En la parte (b) se observa una

frecuencia sencilla (analógica), la cual representa una señal de voz típica, la cual esta compuesta por una mezcla de frecuencias. Como se puede ver, la señal muestreada primero se convierte en una cadena de pulsos, la amplitud de cada uno de los pulsos es igual a la amplitud de la señal analógica original en el instante de muestreo. La señal resultante es conocida como pulso de amplitud modulada o señal PAM.

La señal PAM todavía es analógica porque su amplitud puede variar dentro del rango completo. Para convertirla en una forma digital, es necesario cuantificar cada pulso a su equivalente forma en binario. Ocho dígitos binarios (bits) son usados para cuantificar cada señal PAM, la cual incluye un bit que indica el signo de la señal -positivo o negativo-. Esto significa 256 niveles de uso -desde 8 bits 0 hasta 8 bits 1. La señal digital resultante tiene un promedio de 64 Kbps - lo que es lo mismo a 8,000 muestreos por segundo de cada uno de los 8 bits - y es la unidad mínima para transmitir datos en un circuito digital.

Como se puede deducir de la frecuencia (señal senoidal) en la parte (a), de la figura, el promedio del cambio de la amplitud de la señal varia en diferentes partes del ciclo. En particular los cambios más rápidos en la señal se dan en las amplitudes pequeñas en lugar de las amplitudes más grandes. Como la señal está linealmente cuantificada sobre su amplitud, ocurre un fenómeno conocido como distorsión de cuantización. Para entender esto, la amplitud de la señal PAM primero es clasificada en 8 niveles de prioridad que van a representar los diferentes valores de los pulsos. Con esta técnica se comprimen señales largas y expanden señales pequeñas las cuales se cuantifican. Esto se puede observar en la parte (c) de la figura.

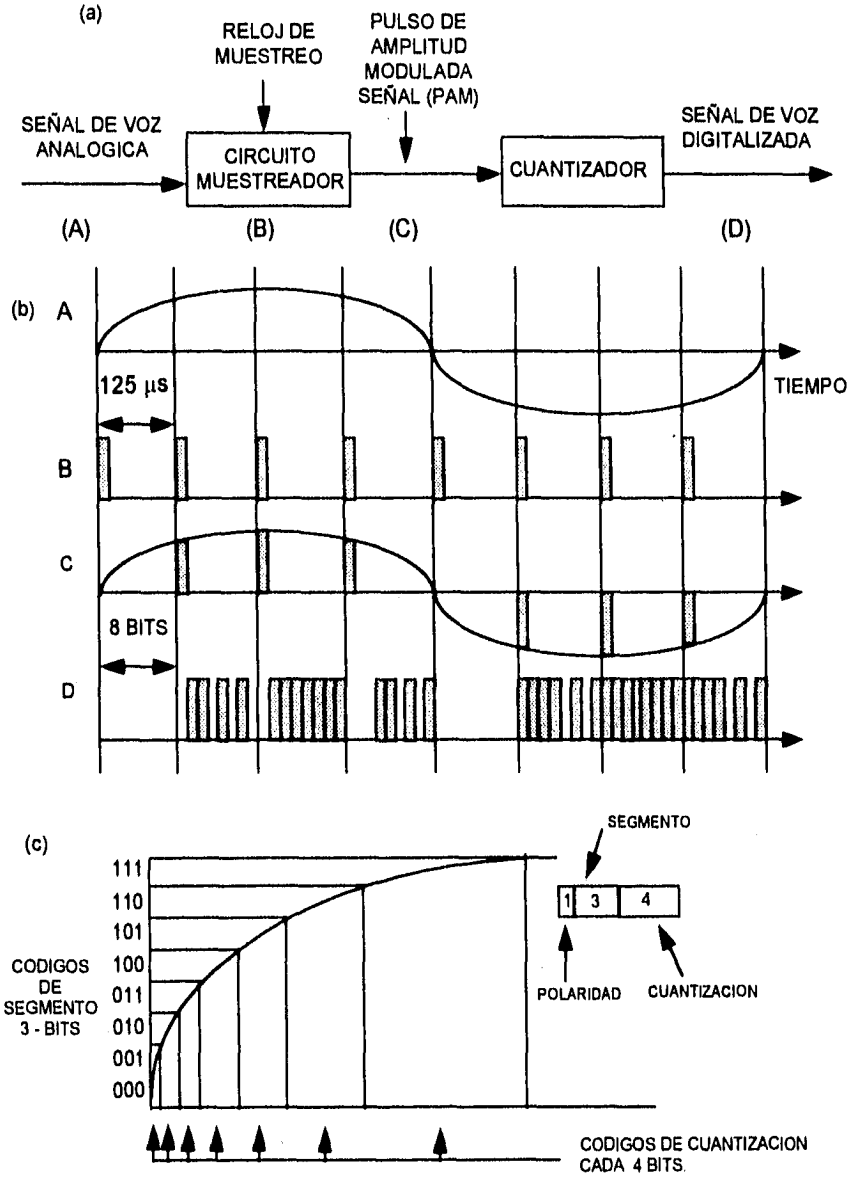


Fig. I.3.4-6. Principios de la Digitalización. (a) Código Esquemático (b) Señales Codificadas (c) Cuantización.

El efecto de cuantificación es aquel que digitaliza el valor (cuantifica) de cada una de las señales PAM, las cuales están hechas de tres partes: Un Bit de polaridad (0 = positivo, 1 = a negativo), tres bits de código de segmento (denotan 8 segmentos) y cuatro bits de cuantización (los cuales dan 16 niveles de cuantización para cada uno de los segmentos). Las leyes que rigen este procedimiento en Estados Unidos y Japón son ligeramente diferentes a las usadas en Europa y otras partes del mundo. Debido a esto la conversión necesita de circuitos que puedan operar en ambos continentes. Afortunadamente esto sólo es necesario en las comunicaciones de voz y no en las de datos.

Multiplexación.

Los circuitos que intercambian ó conmutan múltiples llamadas se les conoce como circuitos conmutados. Esto es logrado por la multiplexación por división del tiempo (TDM) en forma digital. Esto quiere decir que en un equipo TDM, las señales digitales que provienen de múltiples fuentes son asignadas a un tiempo especifico (time slot) en una frecuencia mucho mayor. Como cada señal analógica es muestreada 8,000 veces por segundo, esto produce un muestreo de 8 bits cada 125 microsegundos. El promedio de bits, es una función del número de canales de voz que este porta. En América y Japón 24 canales de voz son agrupados aunque en algunos países se utilizan 30 canales. Esto agrega promedios de 1,544 Mbps y 2,048 Mbps respectivamente, el esquema general de este proceso se ilustra en la figura 1.3.4. - 7 parte (a).

Es necesario incluir bits adicionales (o canales) para otros propósitos. Esto incluye bits para empezar cada **frame** - sincronización del frame - y bits por cada llamada (señalización). Para la sincronización del frame en América se

usa un bit al principio de cada frame with **toggles** (alternados) entre 1 y 0 para frames consecutivos. La información señalizada se lleva en el primer bit de los **slots**. Por lo tanto el promedio de bits es de (24 x 8 + 1) bits/ 125 s = 1.544 Mbps. Estos circuitos son conocidos como DS1 o T1.

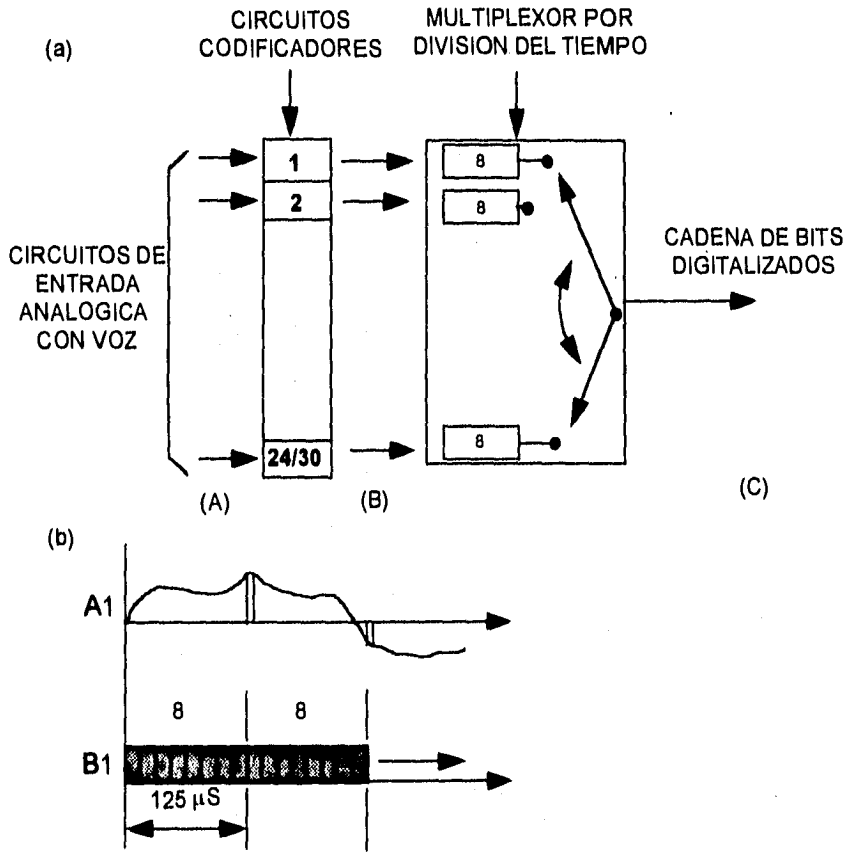


Fig. I.3.4-7. Multiplexación. (a) Esquema TDM; (b) Estructuras Frames

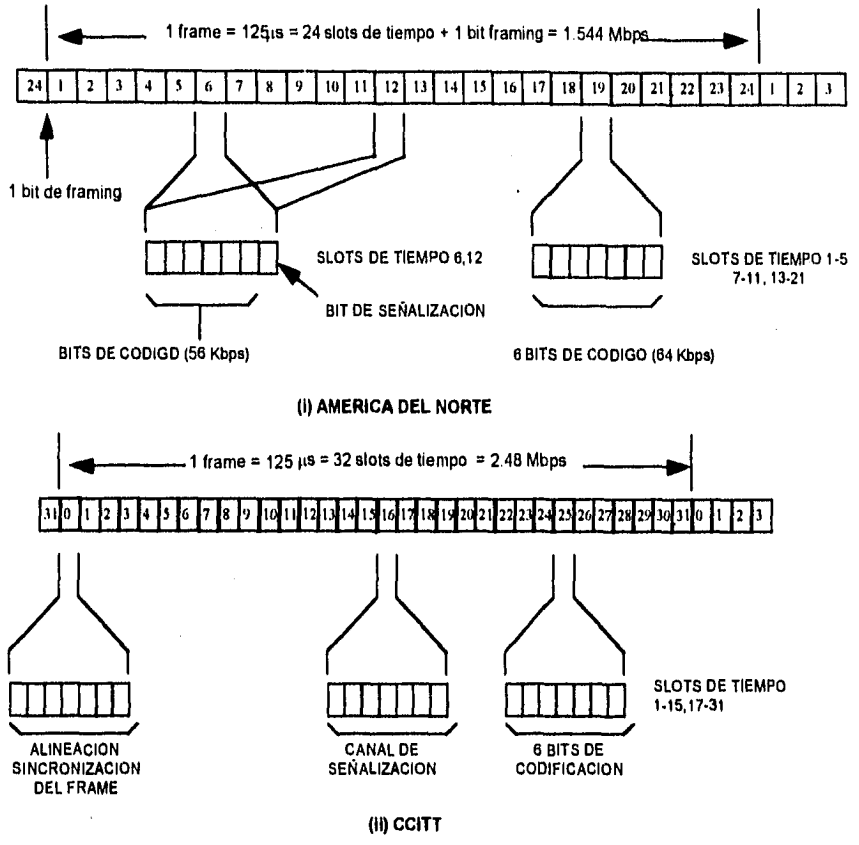


Fig. I.3.4-7. Multiplexación. (b) Estructuras Frames. (Continuación)

En el sistema CCITT el slot de tiempo 0 (cero) es usado para la sincronización de los frame -también es conocido como alineamiento de frame desde que este permite al receptor interpretar los slots de tiempo en cada frame. La señalización es entonces portada a través del slot del tiempo a la que se le agrega un promedio de bits de 32 x 8 bits/125 s = 2.048 Mbps. Este circuito se conoce como E1. En los dos sistemas el promedio de bits más bajo es conocido como T1 / E1.

Los promedios de bits más grandes se logran gracias a la multiplexación entre muchos grupos. El rango de bit promedio de estos dos sistemas junto con sus nombres son los siguientes:

	Estándar	Transmisión (Mbps)	Canales de voz y datos
América:	DS1	1.544	24
	DS1C	3.152	48
	DS2	6.312	96
	DS3	44.736	672
	DS4	274.176	4032
CCITT:	E1	2.048	30
	E2	8.448	120
	E3	34.368	480
	E4	139.264	1920
	E5	565.148	7680

1.4 Teoría de Bases de Datos.

1.4.1. Conceptos Básicos de Bases de Datos.

Un sistema de manejo de base de datos (DBMS) se compone de una serie de datos relacionados entre sí y de un conjunto de archivos interrelacionados y una serie de programas que permiten a varios usuarios tener acceso a estos archivos y modificarlos. Uno de los objetivos principales de un sistema de base de datos es proporcionar a los usuarios una visión abstracta de la información. Es decir, el sistema oculta ciertos detalles relativos a la forma como los datos se almacenan y mantienen.

Los sistemas de base de datos se diseñan para manejar grandes cantidades de información. El manejo de los datos implica tanto la definición de estructuras para el almacenamiento como la creación de mecanismos para manejar la información. Además, el sistema de base de datos debe cuidar la seguridad de la información almacenada en la base datos, previendo caídas del sistema o intentos de acceso no autorizados. Si se va compartir la información entre varios usuarios, el sistema debe evitar posibles resultados anómalos.

Uno de los objetivos principales de una base de datos es proporcionar a los usuarios una visión abstracta de los datos. Es decir, el sistema oculta ciertos detalles relativos a la forma en que se almacenan y mantienen los datos. Esto se logra definiendo tres niveles de abstracción en los que puede considerarse la base de datos: físico, conceptual y de visión.

Para describir la naturaleza de una base de datos, se define el concepto de modelo de datos, que es un conjunto de herramientas conceptuales para

describir los datos, las relaciones entre ellos, su semántica y sus limitantes. Se han propuesto varios modelos diferentes, los cuales se dividen en tres grupos: lógicos basados en objetos, lógicos basados en registros y los modelos físicos de datos.

Las bases de datos cambian con el tiempo al insertar información en ellas y eliminarla. El conjunto de información almacenada en la base de datos en determinado momento se denomina instancia de la base de datos. El diseño general de dicha base se conoce como esquema de la base de datos. La capacidad para modificar una definición de esquema en un nivel sin afectar la definición del esquema en el nivel inmediato superior se denomina independencia de los datos. Existen dos niveles de ésta: independencia física e independencia lógica de los datos.

Un lenguaje de manejo de datos (DML) permite a los usuarios tener acceso a los datos o manejarlos. Existen básicamente dos tipos de DML: de procedimientos (requieren que el usuario especifique cuáles datos necesita y cómo se van a obtener) y sin procedimientos (requieren que el usuario especifique cuáles son los datos que necesita sin especificar la forma de obtención).

Un manejador de base de datos es un módulo de programa que constituye la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicaciones y las consultas que se hacen al sistema. El manejador de base de datos se encarga de interactuar con el manejador de archivos, de conservar la integridad, de garantizar la seguridad, del respaldo y recuperación, y del control de concurrencia.

- Redundancia e inconsistencia de los datos.
- Dificultad para tener acceso a los datos.
- Aislamiento de los datos.
- Usuarios múltiples.
- Problemas de seguridad.
- Problemas de integridad.

Abstracción de la información.

La búsqueda de la eficiencia conduce al diseño de estructuras de datos complejas para representar la información en la base de datos.

Niveles de abstracción en los que puede observarse la base de datos.

- Nivel físico. Este es el nivel más bajo de abstracción, en el que se describe cómo se almacenan realmente los datos. En este nivel se describen en detalle las estructuras de datos complejas del nivel más bajo.

- Nivel conceptual. Este es el siguiente nivel más alto de abstracción, en el que se describe cuáles son los datos reales que están almacenados en la base de datos y qué relaciones existen entre los datos. Este nivel contiene toda la base de datos en términos de unas cuantas estructuras relativamente sencillas. Aunque es posible que la implantación de las estructuras simples del nivel conceptual requiera estructuras complejas en el nivel físico, no es forzoso que el usuario del nivel conceptual se dé cuenta de ello. El nivel conceptual de abstracción lo utilizan los administradores de base de datos; quienes deciden qué información se guarda en la base de datos.

- Nivel de visión. Este es el nivel de abstracción más alto, en el cual se describe solamente una parte de la base de datos. Aunque en el nivel conceptual se utilizan estructuras más simples, todavía queda una forma de complejidad que resulta del gran tamaño de la base de datos. El sistema puede proporcionar muchas vistas diferentes de la misma base de datos.

La interrelación entre estos tres niveles de abstracción se muestran en la figura 1.4.1.-1.

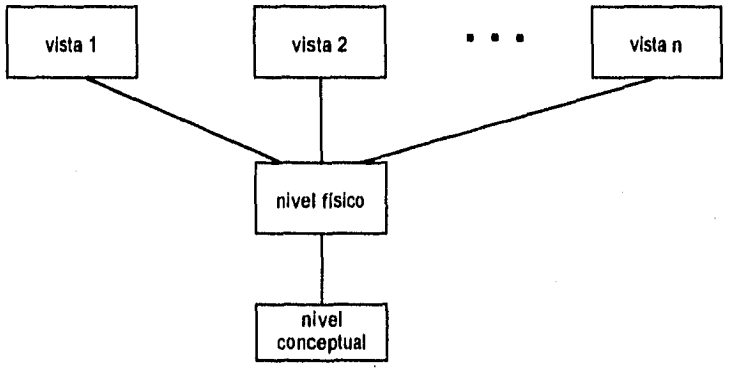


Fig. 1.4.1.-1 Los Tres Niveles de Abstracción de la Información.

Modelos de Datos.

Para describir la estructura de una base de datos es necesario definir el concepto de modelo de datos. Este es un grupo de herramientas conceptuales para describir los datos, sus relaciones, su semántica y sus limitantes. Se han propuesto varios modelos de datos diferentes, los cuales

pueden dividirse en tres grupos: los modelos lógicos basados en objetos y en registros, y los modelos físicos de datos.

- Modelos lógicos basados en objetos.

Los modelos lógicos basados en objetos se utilizan para describir los datos en los niveles conceptual y de visión. Se caracterizan por el hecho de que permiten una estructuración bastante flexible y hacen posible especificar claramente las limitantes de los datos. Algunos de los más conocidos son:

- El modelo entidad-relación
- El modelo binario
- El modelo semántico de datos
- El modelo infológico

Se ha escogido el modelo entidad-relación como representativo de la clase de modelos lógicos basados en objetos. Se eligió éste porque ha tenido bastante aceptación como modelo de datos apropiado para el diseño de bases de datos y porque se utiliza ampliamente en la práctica.

El modelo de datos entidad-relación (E-R) se basa en una percepción de un mundo real que consiste en un conjunto de objetos básicos llamados entidades, y de las relaciones entre estos objetos. Una entidad es un objeto que existe y puede distinguirse de otros. La distinción se logra asociando a cada entidad un conjunto de atributos que describen al objeto. El conjunto de todas las entidades y relaciones del mismo tipo se denomina conjunto de entidades y conjunto de relaciones, respectivamente.

Además de entidades y relaciones, el modelo E-R representa ciertas limitantes que debe cumplir el contenido de una base de datos. Una de estas limitantes importantes es la cardinalidad de mapeo, que expresa el número de entidades con las que puede asociarse otra entidad por medio de un conjunto de relaciones.

La estructura lógica general de una base de datos puede expresarse gráficamente por medio de un diagrama E-R que consta de los siguientes componentes: rectángulos, elipses, rombos y líneas.

Modelos Lógicos Basados en Registros.

Los modelos lógicos basados en registros se utilizan para describir los datos en los niveles conceptual y de visión. A diferencia de los modelos de datos basados en objetos, estos modelos sirven para especificar tanto la estructura lógica general de la base de datos como una descripción en un nivel más alto de la implantación. Sin embargo, no permiten especificar en forma clara las limitantes de los datos.

- Modelo relacional. Los datos y las relaciones entre estos se representan por medio de una serie de tablas, cada una de las cuales tiene varias columnas con nombres únicos.

- Modelo de red. Los datos en el modelo de red se representan por medio de conjuntos de registros y las relaciones entre los datos se representan con ligas, que pueden considerarse como apuntadores. Los registros de la base de datos se organizan en forma de conjuntos de gráficas arbitrarias.

- Modelo jerárquico. El modelo jerárquico es similar al modelo de red en cuanto a que los datos y las relaciones entre los datos se representan por medio de registros y ligas, respectivamente. El modelo jerárquico difiere del de red en que los registros están organizados como conjuntos de árboles en vez de gráficas arbitrarias.

Modelos Físicos de los Datos.

Los modelos físicos de los datos sirven para describir los datos en el nivel más bajo. A diferencia de los modelos lógicos de los datos, son muy pocos los modelos físicos utilizados. Algunos de los más conocidos son:

- El modelo unificador.
- La memoria de cuadros.

Los modelos físicos de los datos capturan aspectos de la implantación de los sistemas de base de datos.

Instancias y Esquemas.

Las bases de datos cambian con el tiempo al insertarse información en la base de datos y eliminarse de ella. El conjunto de información almacenada en la base de datos en cierto momento se denomina una instancia en la base de datos. El diseño general de la base de datos se llama esquema de la base de datos. Los esquemas se alteran muy raras veces, o nunca.

El concepto de un esquema de base de datos corresponde a la idea de definición de tipo en el lenguaje de programación. Una variable de un

determinado tipo tiene un valor específico en un momento dado. Así el concepto del valor de una variable en los lenguajes de programación corresponde al concepto de una instancia del esquema de una base de datos.

Existen varios esquemas en la base de datos, y éstos se dividen de acuerdo con los niveles de abstracción mencionados. En el nivel más bajo se tiene el esquema físico; en el nivel intermedio está el esquema conceptual, mientras que en el nivel más alto existe un subesquema.

Independencia de los Datos.

Se definieron tres niveles de abstracción en los que puede verse la base de datos. La capacidad de modificar una definición de esquema en un nivel sin afectar la definición del esquema en el nivel inmediato superior se denomina independencia de los datos. Existen dos niveles de tal independencia:

Independencia física, que es la capacidad de modificar el esquema físico sin obligar a que se vuelvan a escribir los programas de apilcaciones. En algunas ocasiones son necesarias modificaciones en el nivel físico para mejorar el rendimiento.

Independencia lógica, que es la capacidad de modificar el esquema conceptual sin obligar a que se vuelvan a escribir los programas de aplicaciones. Las modificaciones en el nivel conceptual son necesarias siempre que se altera la estructura lógica de la base de datos.

La independencia lógica de los datos es más difícil de lograr que la independencia física, ya que los programas de aplicaciones dependen en alto grado de la estructura lógica de los datos a los que tienen acceso.

Lenguaje de Definición de Datos.

Un esquema de base de datos se especifica por medio de una serie de definiciones que se expresan en un lenguaje especial llamado lenguaje de definición de datos (DDL, data definition language). El resultado de la compilación de las proposiciones en DDL es un conjunto de tablas que se almacenan en un archivo especial llamado diccionario de datos.

Un diccionario de datos en un archivo que contienen, "datos acerca de los datos". Este archivo se consulta antes de leer o modificar los datos reales en el sistema de base de datos.

La estructura de almacenamiento y los métodos de acceso empleados por el sistema de base de datos se especifican por medio de un conjunto de definiciones de un tipo especial de DDL llamado lenguaje de almacenamiento y definición de los datos. El resultado de la compilación de estas definiciones es una serie de instrucciones que especifican los detalles de implantación de los esquemas de base de datos que normalmente no pueden ver los usuarios.

Lenguaje de Manejo de Datos.

El manejo de los datos consiste en:

- La recuperación de información almacenada en la base de datos.

- La inserción de información nueva en la base de datos.
- La eliminación de información de la base de datos.

En el nivel físico, deben definirse algoritmos que permitan tener acceso a los datos en forma eficiente. En los niveles de abstracción más altos lo importante es la facilidad de uso. El objetivo es lograr una interacción eficiente entre las personas y el sistema.

Un lenguaje de manejo de datos (DML, data manipulation language) permite a los usuarios manejar o tener acceso a los datos que estén organizados por medio del modelo apropiado. Existen básicamente dos tipos de DML.

- De procedimientos, necesitan que el usuario especifique cuáles datos quiere y cómo deben obtenerse.

- Sin procedimientos, requieren que el usuario especifique cuáles datos quiere sin especificar cómo obtenerlo.

Una consulta es una proposición que solicita la recuperación de información. La parte de un DML que implica la recuperación de información se conoce como lenguaje de consultas o lenguaje de manejo de datos como sinónimos.

Manejador de Base de Datos.

Generalmente las bases de datos requieren una gran cantidad de espacio de almacenamiento. Las bases de datos de las empresas comúnmente se miden en términos de gigabytes de información. Un gigabyte equivale a 1 000 megabytes o mil millones de bytes. Puesto que la memoria principal de la

computadora no puede almacenar esta información, se guarda en discos. Los datos se transfieren entre el almacenamiento en disco y la memoria principal, según se requiera. Ya que el movimiento de los datos del disco y al disco es lento comparado con la velocidad de la unidad central de procesamiento de las computadoras, es imperativo que el sistema de base de datos estructure la información de tal manera que se reduzca la necesidad de transferir datos entre el disco y la memoria principal.

El objetivo de un sistema de base datos es simplificar y facilitar el acceso a los datos. Si el tiempo de respuesta para una consulta es demasiado largo, el valor del sistema se reduce. El funcionamiento del sistema depende de la eficiencia de las estructuras de datos utilizados para representar los datos en la base de datos y de qué tan eficientemente pueda operar el sistema con esas estructuras. Como sucede en muchos otros aspectos de los sistemas de cómputo, deben hacerse concesiones, no sólo entre el espacio y el tiempo, sino también entre la eficiencia de un tipo de operación y la de otro.

El manejador de base de datos es responsable de las siguientes tareas:

- Interacción con el manejador de archivos. Los datos sin procesar se almacenan en el disco mediante el sistema de archivos proporcionado normalmente por un sistema operativo convencional. El manejador de base de datos traduce las diferentes proposiciones en DML a comandos de sistema de archivos de bajo nivel. Así, el manejador de base de datos se encarga realmente del almacenamiento, recuperación y actualización de los datos en la base de datos.

- Implantación de la integridad. Los valores de los datos almacenados en la base de datos deben satisfacer ciertos tipos de limitantes de consistencia.

- Puesta en práctica de la seguridad. No es preciso que todos los usuarios de la base de datos tengan acceso a todo su contenido. Es labor del manejador de la base de datos hacer que se cumplan estos requisitos de seguridad.

- Respaldo y recuperación. Un sistema de cómputo, como cualquier otro dispositivo mecánico o eléctrico, está sujeto a fallas. Existen muy diversas causas de estas fallas, entre ellas el aterrizaje las cabezas lectoras de disco, la interrupción del suministro de energía y los errores de software. En cada uno de estos casos se pierde información de la base de datos. Es responsabilidad del manejador de la base de datos detectar estas fallas y restaurar la base de datos al estado que existía antes de presentarse la falla. Esto se logra normalmente iniciando diversos procedimientos de respaldo y recuperación.

- Control de concurrencia. Cuando varios usuarios actualizan la base de datos en forma concurrente, es posible que no se conserve la consistencia de los datos. Es necesario que el sistema controle la interacción entre los usuarios concurrentes; lograr dicho control es una de las tareas del manejador de la base de datos.

Administrador de Base de Datos.

Una de las razones principales para contar con sistemas de manejo de base de datos es tener un control centralizado tanto de los datos como de los programas que tienen acceso a ellos. La persona que tiene este control

centralizado sobre el sistema es el administrador de base de datos (DBA, data base administrator). Las funciones del administrador de base de datos son, entre otras:

- Definición del esquema, es decir, la creación del esquema original de la base de datos. Esto se logra escribiendo una serie de definiciones que el compilador de DDL traduce a un conjunto de tablas que se almacenan permanentemente en el diccionario de datos.

- Definición de la estructura de almacenamiento y del método de acceso, es decir, la creación de las estructuras de almacenamiento y métodos de acceso apropiados. Esto se lleva a cabo escribiendo una serie de definiciones que posteriormente son traducidas por el compilador del lenguaje de almacenamiento y definición de datos.

- Modificación del esquema y de la organización física, ya sea la modificación del esquema de la base de datos o de la descripción de la organización física del almacenamiento. Estos cambios, aunque son relativamente poco frecuentes, se logran escribiendo una serie de definiciones utilizadas, ya sea por el compilador de DDL o por el compilador del lenguaje de almacenamiento y definición de datos para generar modificaciones a las tablas internas apropiadas del sistema.

- Concesión de autorización para el acceso a los datos, es decir, conceder diferentes tipos de autorización para acceso a los datos a los distintos usuarios de la base de datos. Esto permite al administrador de base de datos regular cuáles son las partes de la base de datos a las que van a tener acceso diversos usuarios.

- Especificación de las limitantes de integridad. Estas limitantes se conservan en una estructura especial del sistema que consulta el manejador de base de datos cada vez que se lleva a cabo una actualización en el sistema.

Estructura General del Sistema.

Un sistema de base de datos se divide en módulos que se encargan de cada una de las tareas del sistema en general. Algunas de las funciones del sistema de base de datos pueden ser realizadas por el sistema operativo. En la mayor parte de los casos, el sistema operativo proporciona únicamente los servicios más elementales y la base de datos debe partir de ese fundamento. Así, el diseño de la base de datos debe incluir una consideración de la interfaz entre el sistema de base de datos y el sistema operativo.

Un sistema de base datos consiste en varios componentes funcionales, entre los que se cuentan:

- El manejador de archivos, encargado de asignar espacio en el disco y de las estructuras de datos que se van a emplear para representar la información almacenada en el disco.

- El manejador de base de datos, que constituye la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programa de aplicaciones y las consultas que se hacen al sistema.

- El procesador de consultas, que traduce las proposiciones en lenguaje de consulta a instrucciones de bajo nivel que puede entender el manejador de la base de datos. Además, el procesador de consultas trata de convertir la

solicitud del usuario a una forma equivalente pero más eficiente, encontrando una estrategia adecuada para ejecutar la consulta.

- El precompilador de DML, que convierte la proposiciones en DML incrustadas en un programa de aplicaciones en llamadas normales a procedimientos en el lenguaje huésped. El precompilador debe interactuar con el procesador de consultas para generar el código apropiado.

- El compilador de DDL, que convierte las proposiciones en DDL en un conjunto de tablas que contienen metadatos. Tales tablas se almacenan después en el diccionario de datos.

Además, se requieren varias estructuras de datos como parte de la implantación del sistema físico, incluyendo:

- Archivos de datos, que guardan la base de datos.
- Diccionario de datos, que almacena la información relativa a la estructura de la base de datos. Se usa constantemente, por lo que debe tenerse mucho cuidado de desarrollar un diseño apropiado y una implantación eficiente.
- Indices, que permiten el acceso rápido a elementos de información que contienen valores determinados.

La figura 1.4.1.-2. muestra los componentes y las conexiones entre ellos.

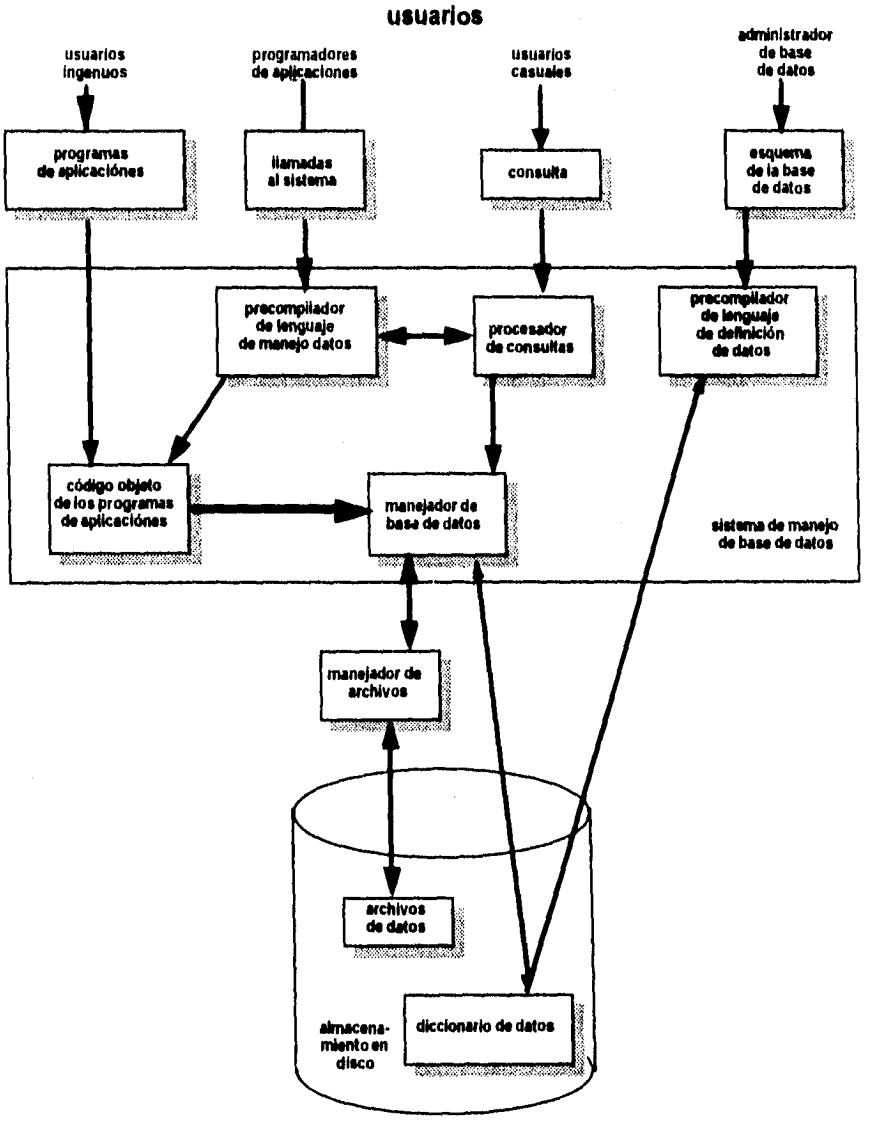


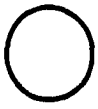
Fig. 1.4.1.-2. Relaciones entre Archivos de Datos, Diccionarios de Datos e Indices.

1.4.2. Diagrama de Flujo de Datos.

La información que se maneja en un software, pasa por ciertos procesos, en los cuales la mayoría de las veces es modificada. El diagrama de flujo de datos (DFD), es una representación gráfica de los flujos de datos y de las funciones o procesos de un sistema, permitiendo al ingeniero de software desarrollar los módulos del ámbito de información y del ámbito funcional.

La notación básica que se usa para crear diagramas de flujo de datos, consta de cuatro elementos que son:

Los flujos de datos son canales, en los cuales fluyen unidades de información cuya estructura conocemos.



Las transformaciones o procesos, son acciones que transforman uno o varios flujos de datos centrales ó varios flujos de datos

Un almacén de datos, es un depósito para los datos al que se accede con retardo temporal. Puede tratarse tanto de ficheros manuales, como de ficheros legibles para la máquina.

Un enlace final, es un enlace del sistema con el exterior, con ellos se representan por ejemplo las introducciones que efectúa el usuario al sistema, así las salidas del sistema, destinadas al usuario.

Al diseñar los diagramas de flujo de datos se deben seguir las siguientes reglas:

- Se documenta exclusivamente el flujo de datos.

- Los flujos de datos, así como las transformaciones deben llevar nombres claros y expresivos.

- Cada una de las transformaciones pueden generar únicamente aquellas salidas para las que también recibe las entradas relevantes.

El diagrama de flujo de datos (DFD) más sencillo, es aquel en el cual el programa se especifica con una sola burbuja, y se dice que este DFD es de nivel 0, mientras más específico sea este diagrama va aumentando de nivel.

1.4.3. Modelo Entidad-Relación.

El modelo de datos de entidad-relación (E-R) se basa en una percepción de un mundo real que consiste en un conjunto de objetos básicos llamados entidades y de relaciones entre estos objetos. Se desarrolló para facilitar el diseño de bases de datos permitiendo especificar un esquema empresarial. Este esquema representa la estructura lógica general de la base de datos.

Entidades y Conjuntos de Entidades.

Una entidad es un objeto que existe y puede distinguirse de otros objetos.

Un conjunto de entidades es un grupo de entidades del mismo tipo. No es necesario que las entidades pertenezcan exclusivamente a un solo conjunto.

Una entidad está representada por un conjunto de atributos.

Un conjunto de relaciones es un grupo de relaciones del mismo tipo. Formalmente es una relación matemática de n>=2 (posiblemente idénticos)

conjuntos de entidades. Si E_1, E_2, ..., E_n son conjuntos de entidades, entonces un conjunto de relaciones R es un subconjunto de $\{(e_1, e_2,...,e_n) \mid e_1 \in E_1, e_2 \in E_2, ..., e_n \in E_n\}$

Limitantes de Mapeo.

Un esquema E-R empresarial puede definir ciertas limitantes con las que deben cumplir los datos contenidos en la base de datos. Una limitante importante es la de las cardinalidades de mapeo que expresan el número de entidades con las que puede asociarse otra entidad mediante una relación.

Las cardinalidades de mapeo son más útiles al describir conjuntos binarios de relaciones, aunque en ocasiones contribuyen a la descripción de conjuntos de relaciones que implican más de dos conjuntos de entidades.

Para un conjunto binario de relaciones R entre los conjuntos de entidades A y B, la cardinalidad de mapeo debe ser una de las siguientes:

- **Una a una.** Una entidad en A está asociada únicamente con una entidad en B, y una entidad en B está asociada sólo con una entidad en A.
- **Una a muchas.** Una entidad en A está relacionada con cualquier número de entidades en B, pero una entidad en B puede asociarse únicamente con una entidad en A.
- **Muchas a una.** Una entidad en A está vinculada únicamente con una entidad en B, pero una entidad en B está relacionada con cualquier número de entidades en A.

- **Muchas a muchas.** Una entidad en A está asociada con cualquier número de entidades en B, y una entidad en B está vinculada con cualquier número de entidades en A.

La figura 1.4.3.-1. muestra las cardinalidades de mapeo mencionadas.

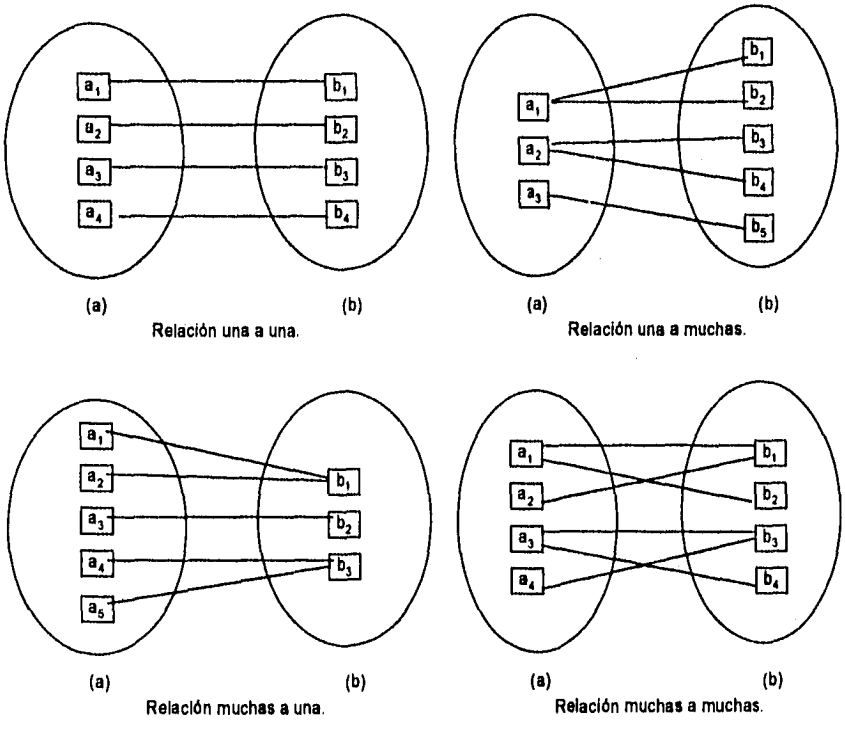


Fig. 1.4.3.-1. Cardinalidades de Mapeo.

La cardinalidad de mapeo apropiada para un conjunto de relaciones determinado dependerá, obviamente, del mundo real que el conjunto de relaciones está modelando.

Las dependencias de existencia constituyen otra clase importante de limitantes. Específicamente, si la existencia de la entidad "x" depende de la existencia de la entidad "y", entonces se dice que "x" es dependiente por existencia de "y". Funcionalmente, esto quiere decir que si se elimina "y", también se eliminará "x". Se dice que la entidad "y" es una entidad dominante y que "x" es una entidad subordinada.

Una tarea muy importante dentro de la modelación de bases de datos consiste en especificar cómo se van a distinguir las entidades y las relaciones. Conceptualmente, las entidades individuales y las relaciones son distintas entre si, pero desde el punto de vista de una base de datos la diferencia entre ellas debe expresarse en términos de sus atributos.

Es posible que existan varios conjuntos de atributos distintos que pudieran servir como llaves candidato.

Se utilizará el término llave primaria para referirse a la llave candidato que elija el diseñador de la base de datos como la forma principal de identificar a las entidades dentro de un conjunto de éstas.

Los conjuntos de relaciones también tienen llaves primarias. Sus llaves primarias se forman tomando todos los atributos que constituyen las llaves primarias de los conjuntos de entidades que definen al conjunto de relaciones.

Diagrama Entidad-Relación.

Un diagrama E-R que se integra con los siguientes componentes, los cuales se muestran en la figura 1.4.4.

- Rectángulos, que representan conjuntos de entidades.
- Elipses, que representan atributos.
- Rombos, que representan conjuntos de relaciones.
- Líneas, que conectan los atributos a los conjuntos de entidades, y los conjuntos de entidades a los conjuntos de relaciones.

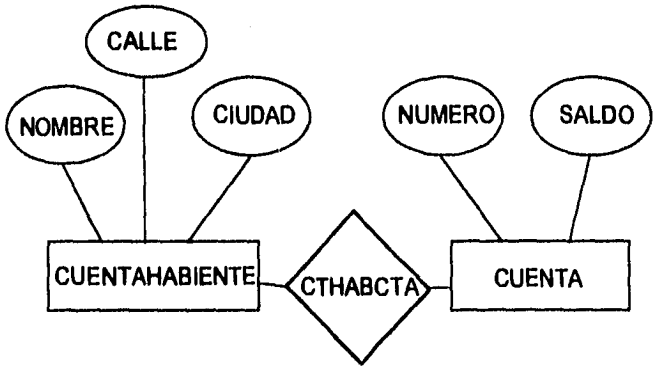


Fig. 1.4.3.-2. Ejemplo de un Diagrama E - R.

Reducción de los Diagramas E-R a Tablas.

Una base de datos que se ajuste a un diagrama E-R puede representarse por medio de un conjunto de tablas. Para cada conjunto de entidades y de relaciones en la base de datos, existe una tabla única que recibe el nombre del conjunto de entidades o de relaciones correspondiente. Cada tabla tiene un número de columnas que, también tiene nombres únicos.

Representación de Conjuntos de Entidades Fuertes.

Sea E un conjunto de entidades fuertes con los atributos descriptivos $a_1, a_2,...,a_n$. Este conjunto de entidades se representa por medio de una tabla denominada E con n columnas diferentes, cada una de las cuales corresponde a uno de los atributos de E.

Representación de Conjuntos de Entidades Débiles.

Sea A un conjunto de entidades débiles con los atributos descriptivos $a_1, a_2, ..., a_n$. Sea "B" el conjunto de entidades fuerte del que depende "A". La llave primaria de "B" incluye los atributos $b_1, b_2, ..., b_n$. El conjunto de entidades "A" se representa mediante una tabla llamada "A" con una columna por cada atributo del conjunto.

$\{a_1, a_2,...,a_i\} \cup \{b_1,b_2,...,b_i\}$

Generalización y Especialización.

- La generalización es el resultado de la unión de dos o más conjuntos de entidades (de bajo nivel) para producir un conjunto de entidades de más alto nivel.

- La especialización es el resultado de tomar un subconjunto de un conjunto de entidades de alto nivel para formar un conjunto de entidades de más bajo nivel.

Agregación.

Una limitación del modelo E-R es que no es posible expresar relaciones entre relaciones. Para ilustrar el porque de la necesidad de una construcción de este tipo, piénsese en una base de datos que describe información acerca de los empleados que trabajan en un proyecto determinado y empleando varias máquinas diferentes para realizar ese trabajo. Empleando las construcciones básicas de modelado E-R, se obtiene el diagrama E-R. Los conjuntos de relaciones trabaja y utiliza pueden combinarse para formar un solo conjunto de relaciones. Sin embargo, si eso se hiciera, no podría verse claramente la estructura lógica de este esquema.

La solución es utilizar la agregación; esto es, una abstracción por medio de la cual las relaciones se tratan como entidades de alto nivel.

1.4.4 Normalización.

El concepto principal, tomado del modelo relacional utilizado en el desarrollo del modelo conceptual, es el proceso de normalización, esto es, el proceso de agrupar a los campos de datos en tablas que representan a las entidades y sus relaciones. La teoría de la normalización está basada en la observación de que un cierto conjunto de relaciones tiene mejores propiedades en un medio de inserción, actualización y supresión, que las que tendrían otros conjuntos de relaciones conteniendo los mismos datos.

La razón de usar el procedimiento de normalización es asegurar que el modelo conceptual de la base de datos funcionará. Esto no significa que una estructura no normalizada no funcionará, sino que puede causar algunos problemas cuando los programadores de aplicación traten de modificar la base de datos. El administrador de la base de datos debe decidir, después de localizar las violaciones provenientes de la normalización, si las modificaciones afectarán la forma en que la base de datos funcionará.

Un modelo de datos no normalizado consiste en registros utilizados por los programas de aplicación. El primer paso de la normalización consiste en transformar los campos de datos a una tabla de dos dimensiones. Lo que se requiere usualmente en este paso es la eliminación de ocurrencias repetidas de campos de datos, de tal manera que se obtenga un archivo fijo. Por ejemplo, si una declaración incluye espacio para el nombre de un empleado, su número, esposa y hasta diez hijos, el resultado será una tabla de 4 por 10, con cuatro columnas y diez renglones. Cada uno de los renglones tendrá el nombre y número del empleado, esposa y el nombre de uno de los hijos. Los diez

renglones tendrán los nombres de los diez hijos. Este es sólo un paso preliminar que hace posible trasladarse a la segunda forma normalizada.

El segundo paso de la normalización es establecer las claves y relacionarlas con los campos de datos. En la primera forma normalizada, el renglón entero de la tabla (cadena) depende de todos los campos de claves. En la segunda forma normalizada, se hace un intento de establecer los campos de datos que están relacionados con alguna parte de la clave completa. Si los campos de datos sólo dependen de una parte de la clave, la clave y los campos conectados a la clave parcial son susceptibles de separarse en registros independientes. La división de la primera tabla normalizada, en una serie de tablas en las que cada campo sólo depende de la clave completa se llama la segunda forma normalizada.

El tercer paso consiste en separar los campos de las segundas relaciones normales que, aunque dependan sólo de una clave, deben tener una existencia independiente en la base de datos. Esto se hace de forma tal que la información sobre estos campos pueda introducirse separadamente a partir de las relaciones en las que se encuentra implicada.

En cada modelo de datos uno o más campos de datos se agrupan para representar entidades y sus relaciones . En los agrupamientos de los campos de datos pueden darse tres tipos generales de problemas, y la eliminación de cada uno de éstos da pie a las tres formas normalizadas de relaciones (tablas). Por lo tanto el proceso de normalización es una disciplina que consiste en agrupar a los campos de datos en un conjunto de relaciones (tablas). La figura 1.4.4.-1. muestra las tres formas normalizadas.

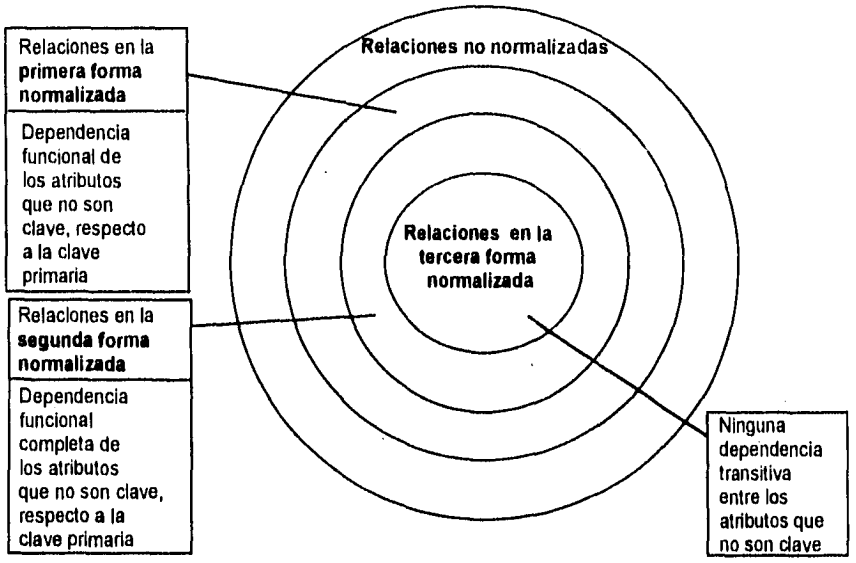


Fig. 1.4.4.-1. Toda Relación que Está en la Primera Forma Normalizada es un Caso Especial de una Relación no Normalizada. Pero no Toda Relación no Normalizada Está en la Primera Forma. Toda Relación que Está en la Segunda Forma Normalizada es un Caso Especial de Relación en la Primera Forma, pero no Viceversa. Toda Relación que Está en la Tercera Forma es un Caso Especial de Relación en la Segunda Forma Normalizada, Pero no Toda Relación en la Segunda Forma Normalizada Está Necesariamente en la Tercera Forma.

Todas las relaciones normalizadas están en la primera forma normal, algunas relaciones de la primera forma normalizada están en la segunda forma y, finalmente, algunas relaciones de la segunda forma normalizada están en la tercera. El proceso de normalización produce las tres formas de relaciones. Las relaciones de la tercera forma representan a las entidades y sus relaciones dentro de una empresa.

La primera, segunda y tercera formas normalizadas proporcionan mejoras sucesivas en las operaciones de inserción, supresión y actualización de la base de datos. Como veremos, el proceso de normalización hace que el diseñador entienda la semántica de los atributos y sus relaciones y como resultado influye en el proceso de consideraciones al hacer el análisis de los datos.

En los párrafos precedentes hemos visto someramente los pasos comprendidos al tomar una estructura no normalizada y convertirla hasta una tercera forma. El resto de esta sección muestra cómo hacer esto.

Considere el ejemplo de la figura 1.4.4.-2. Los campos de datos o atributos son: número, nombre y dirección del paciente, número de licencia y nombre del cirujano, fecha de la cirugía y cirugía, medicina administrada después de la operación y su efecto secundario.

Una columna o conjunto de columnas se llama "clave permitida" cuando sus valores identifican de manera única el renglón de la relación. Dado el número del paciente (1234), el número de licencia del cirujano (243) y la fecha de la cirugía (abril 5, 76), queda determinado de manera única el renglón (cadena) "1234243 abril 5, 1976 Mary Jones 10 Main Street, Rye, N. Y. Charles Field, Extirpación de catarata Tetraciclina Fiebre". Ya que ésta es la única clave permitida se convierte en una clave primaria (se considera que toda relación tiene sólo una clave permitida. Así, la única clave permitida es la clave primaria, y todo atributo que no forma parte de ésta es un atributo que no es clave).

Primera forma normalizada. En la primera forma, una relación es una "tabla". En cada intersección de un renglón (cadena) y en una columna sólo puede

haber un valor en la tabla. Ningún grupo de valores está permitido en las intersecciones.

Considere la figura 1.4.4.-3. Esta es una relación "no normalizada". En los cruces de algunos renglones y columnas se llega a presentar mas de un valor de la clave primaria, los valores de los atributos que no son clave no se pueden determinar de manera única.

Número del paciente	Número de licencia del cirujano	Fecha de la cirugía	Nombre del paciente	Dirección del paciente	Nombre del cirujano	Cirugía	Medicina administrada despues de la operación	Efecto secundario de la medicina
1111	145	Enero 1, 1977	John White	15 New Street, New York, NY.	Beth Little	Extirpación de calculos biliares	Penicilina	Prurito
1111	311	Junio 12, 1977	John White	15 New Street, New York, NY.	Michael Diamond	Extirpación de calculos renales	----------	----------
1234	243	Abril 5, 1976	Mary Jones	10 Main Street Rye, NY.	Charles Field	Extirpación de catarata	Tetraciclina	Flebre
1234	467	Mayo 10, 1977	Mary Jones	10 Main Street Rye, NY.	Patricia Gold	Eliminación de trombosis	----------	----------

Clave primaria

Atributo

Cadena o renglón

Fig. 1.4.4.-2. Representación de Datos Utilizando un Modelo Relacional: "Medio Pseudohospitalario".

Número del paciente	Número de licencia del cirujano	Fecha de la cirugía	Nombre del paciente	Dirección del paciente	Nombre del cirujano	Cirugía	Medicina administrada despues de la operación	Efecto secundario de la medicina
1111	145	Enero 1, 1977	John White	15 New Street, New York, NY.	Beth Little	Extirpación de calculos biliares	Penicilina	Prurito
1111	311	Junio 12, 1977			Michael Diamond	Extirpación de calculos renales	----------	----------
1234	243	Abril 5, 1976	Mary Jones	10 Main Street Rye, NY.	Charles Field	Extirpación de catarata	Tetraciclina	Fiebre
1234	467	Mayo 10, 1977			Patricia Gold	Eliminación de trombosis	----------	----------

Fig. 1.4.4.-3. Esta es una Relación "No Normalizada". En el Cruce de Algunos Renglones y Columnas se Presentan más de un Valor.

Transferimos la relación no normalizada de la figura 1.4.4.-3 a una relación normalizada en la figura 1.4.4.-4. En esta conversión repetimos los valores tomados por el número, nombre y dirección del paciente y, en un caso, los del número de licencia y nombre del cirujano. La figura 1.4.4.-4 representa una relación en la primera forma normalizada. En cada intersección de un renglón (cadena) y una columna, sólo hay un valor en la tabla –no grupos de valores. La figura 1.4.4.-5 muestra la relación de una manera gráfica.

Cuando los valores tomados por el número del paciente, el número de licencia del cirujano y la fecha de cirugía se conocen, también se conocen los valores tomados por el nombre del paciente, la dirección del paciente, el nombre del cirujano, la cirugía, la medicina administrada después de la operación y el efecto secundario de la medicina. Así, la clave primaria está compuesta del número del paciente, el número de licencia del cirujano y la fecha de cirugía. No hay otra clave permitida para esta relación (tabla). Todos los atributos que

no son clave en la relación son funcionalmente dependientes de la clave primaria, quedan determinados de manera única los valores tomados por los atributos que no son clave.

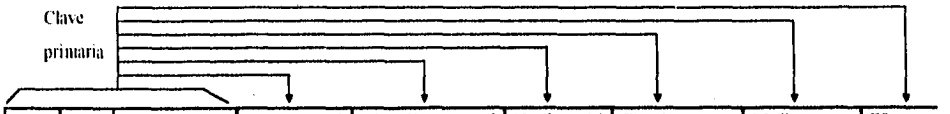
Fallas de almacenamiento de la primera forma normalizada. Las siguientes fallas sobrevendrán si dejamos nuestros datos en la primera forma normalizada (véase la figura 1.4.4.-4).

Falla de inserción. Es posible que a un nuevo paciente no se le haya practicado una intervención quirúrgica en el hospital y que por lo tanto no se le haya asignado ningún cirujano. Como resultado de no habérsele asignado ningún cirujano, la cadena del paciente no se puede introducir, esto es, la información sobre el paciente no se puede almacenar en la primera forma de la figura 1.4.4.-4.

Supongamos que deseáramos insertar los datos del paciente: nombre y dirección. No se debería conocer necesariamente el número de licencia del cirujano ni la fecha de la cirugía. Esto significa que, para identificar de manera única el nombre y la dirección del paciente son superfluos dos componentes de la clave primaria, el número de licencia del cirujano y la fecha de cirugía. Si separamos el nombre del paciente y su dirección, junto con el número del paciente, en una relación (tabla) distinta, como en la figura 1.4.4.-6a, esta falla de inserción se podrá rectificar.

Falla de inserción. Es posible que el hospital haya nombrado a un nuevo cirujano que aún no haya operado a ningún paciente de este hospital. Como consecuencia, en el renglón para este cirujano no habrá ningún valor asignado al número del paciente o a la fecha de la cirugía. Sólo una parte de

la clave primaria está constituida del número de licencia del cirujano; el resto consiste en el número del paciente y en la fecha de la cirugía. Como consecuencia, no se puede introducir una nueva cadena para un nuevo cirujano, es decir, no se puede almacenar la información del cirujano. En este caso, el atributo que no es clave correspondiente al nombre del cirujano, está identificado de manera única con la clave primaria formada por el número del paciente + número de licencia del cirujano + fecha de la cirugía. Pero la identificación única del nombre del cirujano, la única parte necesaria de la clave primaria es el número de licencia del cirujano. Si separamos el nombre y el número de licencia del cirujano, como en la figura 1.4.4.-6b, la falla en la inserción se rectifica.



Número del paciente	Número de licencia del cirujano	Fecha de la cirugía	Nombre del paciente	Dirección del paciente	Nombre del cirujano	Cirugía	Medicina administrada después de la operación	Efecto secundario de medicina
111	145	Enero 1, 1977	John White	15 New Street, Nueva York, N. Y.	Beth Little	Extirpación de cálculos biliares	Penicilina	Prurito
111	311	Junio 12, 1977	John White	15 New Street, Nueva York, N. Y.	Michael Diamond	Extirpación de cálculos renales	--------	-------
1234	243	Abril 5, 1976	Mary Jones	10 Main Street, Rye, N. Y.	Charles Field	Extirpación de catarata	Tetracilina	Fiebre
1234	467	Mayo 10, 1977	Mary Jones	10 Main Street, Rye, N. Y.	Patricia Gold	Eliminación de trombosis	-------	-------
2345	189	Enero 8, 1978	Charles Brown	Dogwood Lane Harrison, N. Y.	David Rosen	Cirugía de corazón abierto	Cefalosporina	-------
4876	145	Noviembre 5, 1977	Hal Kane	55 Boston Post Road. Chester, Conn.	Beth Little	Colecistectomía	Demicilina	-------
5123	145	Mayo 10, 1977	Paul Kosher	Blind Brook Mamaroneck, N. Y.	Beth Little	Extirpación de cálculos biliares	-------	-------
6845	243	Abril 5, 1976	Ann Hood	Hilton Road Larchmont, N. Y.	Charles Field	Reemplazo de córnea	Tetraciclina	Fiebre
6845	243	Diciembre 15, 1976	Ann Hood	Hilton Road Larchmont, N. Y.	Charles Field	Extirpación de catarata	-------	-------

Fig. 1.4.4.-4. Representación de Datos Utilizando un Modelo Relacional. Relación en la Primera Forma Normalizada.

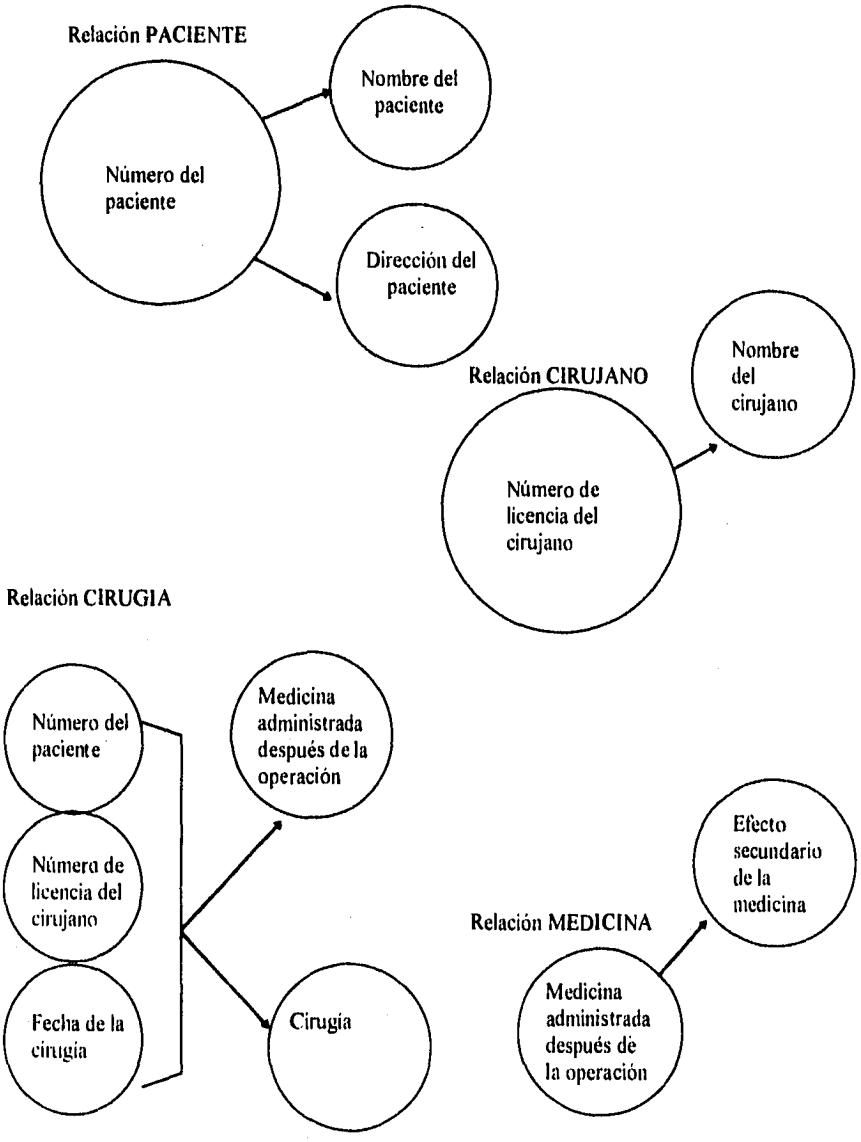


Fig. 1.4.4.-5. Una Relación en la Primera Forma Normalizada.

Estas fallas ocurren debido a que algunas columnas sólo dependen una parte de la clave primaria, mientras que la clave primaria completa es necesaria para identificar de manera única a un renglón completo.

Falla de actualización. Si John White ingresa en el hospital por tercera vez, y si entre la segunda y tercera estancia se cambió de casa, nos gustaría actualizar su dirección en todos los renglones en los que aparece. (Queremos guardar sólo la información actualizada sobre los nombres y direcciones, con el fin de tener consistencia. Suponemos que el hospital no quiere guardar las antiguas direcciones). Este ejemplo muestra la dificultad de actualizar una relación en la primera forma normalizada, debido a que las cadenas en las que se debe reflejar el cambio, varían con el tiempo. Lo peor que podría suceder sería tener algunas cadenas con la dirección antigua y algunas con la nueva.

Esta falla de actualización se puede resolver si la dirección del paciente sólo aparece una vez. Si separamos el nombre del paciente, su dirección y su número, como el la figura 1.4.4.-6a la falla desaparece. La dirección de John White sólo aparecerá una vez en la tabla de PACIENTE, como en la figura 1.4.4.-6a.

Falla de supresión (tipo 1). Supongamos que queremos borrar la información de un paciente después de que él/ella haya muerto. Consideremos que el paciente Charles Brown fallece (véase la figura 1.4.4.-4). Cuando suprimimos la cadena correspondiente a Charles Brown , toda la información relativa al cirujano David Rosen se pierde debido a que esta operación fue la única que él realizo. En algunas aplicaciones, la pérdida de información de este tipo podría tener efectos serios. Puesto que el renglón suprimido podría ser la única fuente de información sobre David Rosen. Para prevenir que algo así ocurra,

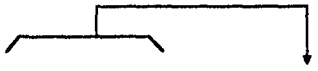
tendríamos que responsabilizar al usuario de la verificación de que la cadena por ser borrada no sea la única fuente de información de esta "categoría".

Tabla PACIENTE

Número del paciente	Nombre del paciente	Dirección del paciente
1111	John White	15 New Street, Nueva York, N. Y.
1234	Mary Jones	10 Main Street, Rye, N. Y.
2345	Charles Brown	Dogwood Lane, Harrison, N. Y.
48776	Hal Kane	55 Boston Post Road, Chester,
5123	Paul Kosher	Conn.
6845	Ann Hood	Blind Brook, Mamaroneck, N. Y.
		Hilton Road, Larchmont, N. Y.

Fig. 1.4.4.-6a. Relación en la Segunda Forma Normalizada. La Clave Primaria es el Número del Paciente. Los Atributos que no son Clave, Nombre y Dirección del Paciente Necesitan de la Clave Primaria Completa para su identificación Unica. El Nombre y la Dirección del Paciente Sólo Aparecen en Esta Relación.

Tabla CIRUJANO



Número de licencia del cirujano	Nombre del cirujano
145	Beth Little
189	David Rosen
243	Charles Field
311	Michael Diamond
467	Patricia Gold

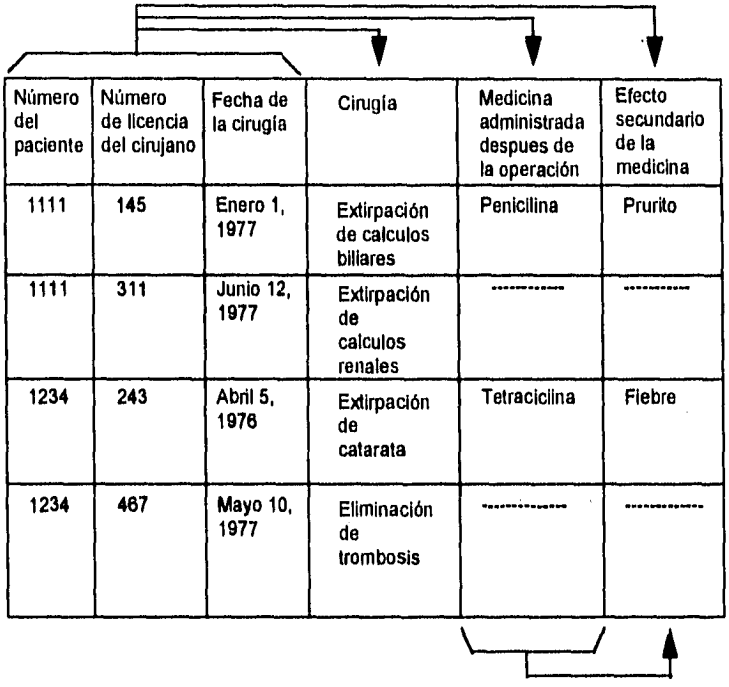
Fig. 1.4.4.-6b. Relación en la Segunda Forma Normalizada. La Clave Primaria es el Número de Licencia del Cirujano. El Atributo que no es Clave, Nombre del Cirujano, Necesita de la Clave Primaria Para su Identificación Unica. El Nombre del Cirujano Sólo Aparece en Esta Relación.

Una posibilidad de resolver un problema de esta naturaleza es separar la información del cirujano que no dependa del paciente y viceversa. Esto se puede lograr teniendo dos relaciones (tablas), como en las figuras 1.4.4.-6a y 1.4.4.-6b, para PACIENTE y CIRUJANO, respectivamente.

Falla de supresión (tipo 2). Otra falla de supresión resulta de un atributo que no es clave, de la relación de la figura 1.4.4.-4, que es funcionalmente dependiente de otro atributo que tampoco sea clave, de la misma relación. El efecto secundario de una medicina es funcionalmente dependiente sólo de la medicina administrada. Es posible que el prurito sea resultante de administrarle otra medicina. Los valores de atributo para esa cadena específica, cambian a otra medicina y a otro efecto secundario. Como consecuencia de este cambio, perdemos la información de que la penicilina produjo prurito. Puesto que esta cadena particular podría constituir la única fuente de información en la base de datos para esa categoría, esta pérdida no es deseable.

Para resolver estas fallas, separaremos la información sobre el paciente en una relación llamada PACIENTE, como en la figura 1.4.4.-6a, y la información del cirujano en una relación llamada CIRUJANO, como en la figura 1.4.4.-6b. La clave primaria en la relación PACIENTE de la figura 1.4.4.-6a es el número del paciente, y la clave primaria en la relación CIRUJANO de la figura 1.4.4.-6b es el número de licencia del cirujano. Los atributos restantes comprenden la relación PACIENTE-Y-CIRUJANO mostrada en la figura 1.4.4.-6c, donde la clave primaria está compuesta del número del paciente, el número de licencia del cirujano y la fecha de cirugía.

Tabla PACIENTE y CIRUJANO



Número del paciente	Número de licencia del cirujano	Fecha de la cirugía	Cirugía	Medicina administrada despues de la operación	Efecto secundario de la medicina
1111	145	Enero 1, 1977	Extirpación de calculos biliares	Penicilina	Prurito
1111	311	Junio 12, 1977	Extirpación de calculos renales	----------	----------
1234	243	Abril 5, 1976	Extirpación de catarata	Tetraciclina	Fiebre
1234	467	Mayo 10, 1977	Eliminación de trombosis	----------	----------

Fig. 1.4.4.-6c. Relación en la Segunda Normalizada. La Clave Primaria es Número de Paciente + Número de Licencia del Cirujano + Fecha de la Cirugía. Todo Atributo que no es Clave (por Ejemplo, Cirugía, Medicina Administrada Después de la Operación y Efecto Secundario de la Medicina) Requieren de la Clave Primaria para su Identificación Unica.

Se dice que las relaciones de las figuras 1.4.4.-6a, 1.4.4.-6b 1.4.4.-6c están en la segunda forma normalizada. Estas tres relaciones para nuestros datos son más deseables que la relación en la primera forma de la figura 1.4.4.-4, debido a que una relación en la segunda forma normalizada elimina algunas de las fallas de almacenamiento encontradas en la relación que está en la primera forma. La figura 1.4.4.-7. representa las tres relaciones de manera gráfica en la segunda forma normalizada.

Relación de la segunda forma normalizada. Se dice que una relación está en la segunda forma normalizada, cuando todo atributo que no sea clave es completamente dependiente de manera funcional de la clave primaria, es decir, todo atributo que no es clave necesita de la clave primaria completa para poder ser identificado de manera única. Inversamente, una relación no está en la segunda forma si existe un atributo que no sea clave que no dependa completamente y de manera funcional de la clave primaria.

Toda relación en la segunda forma normalizada también representa una relación en la primera forma.

Fallas de almacenamiento de la relación en la segunda forma normalizada. Las figuras 1.4.4.-6a, 1.4.4.-6b y 1.4.4.-6c representan tres relaciones en la segunda forma. Algunas de las fallas que se tenían en la primera forma normalizada, como en la figura 1.4.4.-4, se han eliminado en las relaciones que están en la segunda forma.

Inserción. Podemos introducir un nuevo paciente que todavía no ha sido operado en el hospital y al cual no se le ha asignado ningún cirujano, simplemente introduciendo la información del paciente en la relación de la figura 1.4.4.-6a.

Inserción. La información referente a un nuevo cirujano que aún no ha operado a ningún paciénte en el hospital, se puede introducir añadiendo simplemente una nueva cadena en la relación de la figura 1.4.4.-6b.

Supresión (tipo 1). Si Charles Brown fallece, la cadena correspondiente en las relaciones de las figuras 1.4.4.-6a y 1.4.4.-6c se pueden suprimir. La información

referente al cirujano David Rosen permanecerá en la relación de la figura 1.4.4.-6b.

Actualización. Si John White ingresa al hospital por tercera vez y tiene una dirección distinta, el único lugar en que el cambio en la dirección se tendrá que hacer será en la relación de la figura 1.4.4.-6a y no en la de la figura 1.4.4.-6c. Parece que existen algunas fallas de almacenamiento en la relación de la figura 1.4.4.-6c.

Falla de inserción. No podemos introducir el hecho de que una determinada medicina produzca determinados efectos secundarios a menos que está se le suministre a un paciente.

No podemos introducir una cadena en la relación de la figura 1.4.4.-6c hasta que tengamos un paciente que haya sido operado y al cual se le haya suministrado dicha medicina.

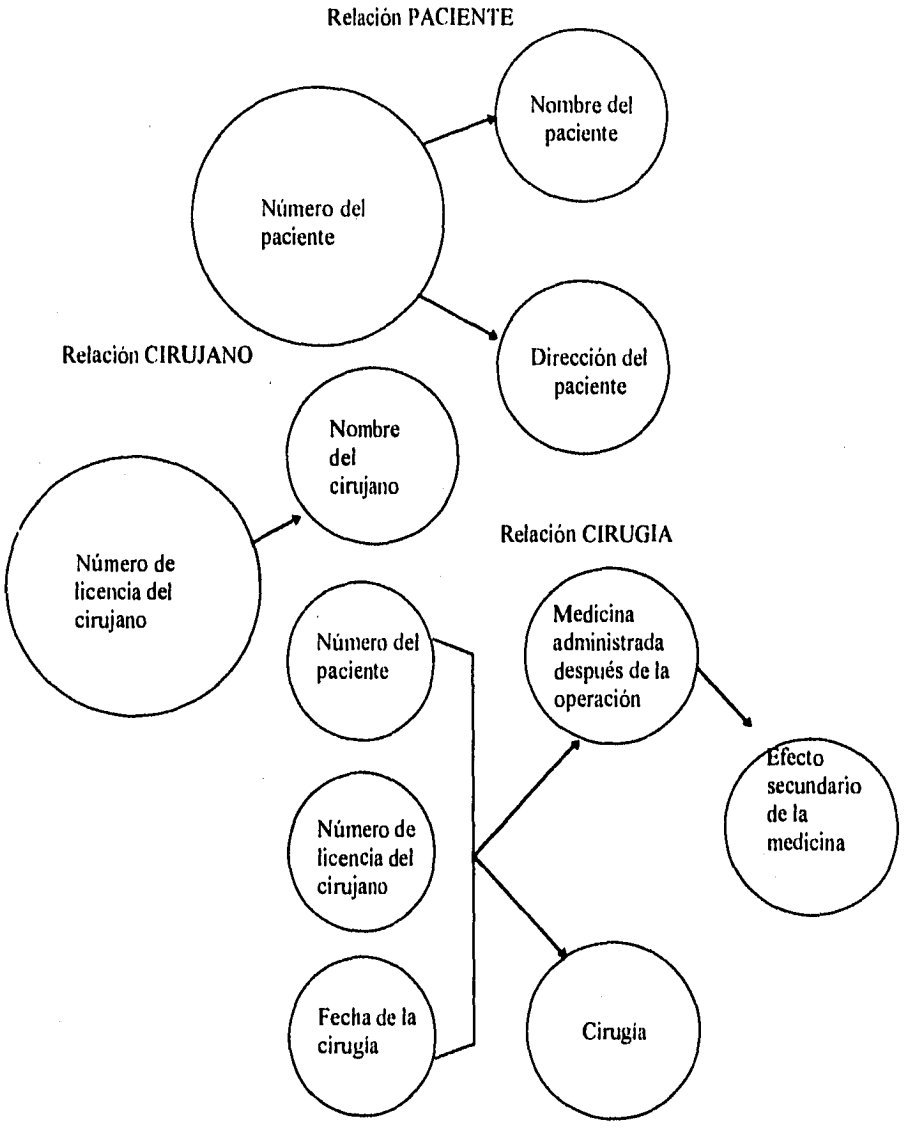


Fig. 1.4.4.-7. Tres Relaciones en la Segunda Forma Normalizada. La Clave Primaria de la Relación PACIENTE es el Número del Paciente. La Clave Primaria de la Relación CIRUJANO es, el Número de Licencia del Cirujano. La Clave Primaria de la Relación CIRUGIA es el Número del Paciente + Número de Licencia del Cirujano + Fecha de la Cirugía.

Falla de supresión. La falla de supresión en la cual un atributo que no es clave de una relación depende funcionalmente de otro atributo que tampoco sea clave permanece. Si John White recibe alguna otra medicina como consecuencia del severo prurito debido a la penicilina y se actualizan los valores de atributo de la medicina administrada y su efecto secundario, perdemos la información de que la penicilina produjo prurito. Puesto que esa cadena particular era la única fuente de información en la base de datos para esa categoría, esta pérdida no es deseable.

Falla de actualización. El efecto secundario de una medicina aparece varias veces en la relación de la figura. 1.4.4.-6c. Si el fabricante de una determinada medicina cambia la fórmula de tal manera que el efecto secundario cambie, nos encontramos con dos posibilidades: 1. Buscar en toda la relación de la figura 1.4.4.-6c y cambiar el valor del efecto secundario cada vez que la medicina se haya administrado o; 2. Trabajar con algunas inconsistencias como consecuencia de cambiar el valor sólo en algunas cadenas. Las fallas de inserción, supresión y actualización son los resultados de la dependencia del atributo que no es clave "efecto secundario de la medicina" de otro atributo que tampoco sea clave, "medicina administrada después de la operación". Este tipo de dependencia se llama "transitiva" (véase la figura 1.4.4.-8).

La solución a estos problemas de la relación de la figura 1.4.4.-6c, es reemplazaria con otras dos relaciones, como en las figuras 1.4.4.-9c y 1.4.4.-9d.

Puesto que no hay fallas de almacenamiento en las relaciones de las figuras 1.4.4.-6a y 1.4.4.-6b, las dejamos intactas en las figuras 1.4,4.-9a y 1.4.4.-9b, respectivamente.

Se debe notar que la conversión de la segunda a la tercera forma normalizada es semejante a la conversión de la primera a la segunda forma, excepto al trabajar con atributos que no son clave. En la conversión de 1FN (primera forma normal) a 2FN (segunda forma normal), observamos las relaciones entre los atributos que son clave y los que no son. Sin embargo, en la conversión de 2FN a 3FN observamos las relaciones entre los atributos que no son clave.

Las relaciones PACIENTE, CIRUJANO, PACIENTE-y-CIRUJANO y MEDICINA, que se muestran en las figuras 1.4.4.-9a, 1.4.4.-9b, 1.4.4.-9c, 1.4.4.-9d, respectivamente, están en la tercera forma normalizada. La figura 1.4.4.-10 muestra estas cuatro relaciones en forma gráfica.

Note, también, que MEDICINA puede ser una clave para contestar preguntas como "¿Quién está tomando o tomó tetraciclina?".

Relación en la tercera forma normalizada. Se dice que una relación está en la tercera forma de normalización si no existe ninguna dependencia funcional transitiva entre los atributos que no son clave. (Cuando un atributo que no es clave se puede determinar con uno o más atributos que tampoco lo sean, se dice que existe una dependencia funcional transitiva entre los dos. El efecto secundario de una medicina se puede determinar a partir de la medicina administrada. Así, el efecto secundario de una medicina depende funcional y transitivamente de la medicina administrada.) Alternativamente, si existen algunos atributos que no siendo clave tengan dependencia funcional entre ellos, la relación no es de la tercera forma.

Como consecuencia de la creación de dos relaciones, como las mostradas en las figuras 1.4.4.-9c 1.4.4.-9d hechas de la relación de la figura 1.4.4.-6c, se eliminan las fallas de almacenamiento.

Inserción. Podemos introducir el hecho de que una medicina determinada tenga determinado efecto secundario en la relación de la figura 1.4.4.-9d, sin que haya administrado la medicina a ningún paciente.

Dependencia transitiva

Número del paciente	Número de licencia del cirujano	Fecha de la cirugía	Cirugía	Medicina administrada despues de la operación	Efecto secundario de la medicina
1111	145	Enero 1, 1977	Extirpación de calculos biliares	Penicilina	Prurito
1111	311	Junio 12, 1977	Extirpación de calculos renales	----------	----------
1234	243	Abril 5, 1976	Extirpación de catarata	Tetraciclina	Fiebre
1234	467	Mayo 10, 1977	Eliminación de trombosis	----------	----------

Fig. 1.4.4.-8. Representación de Datos Utilizando un Modelo Relacional.

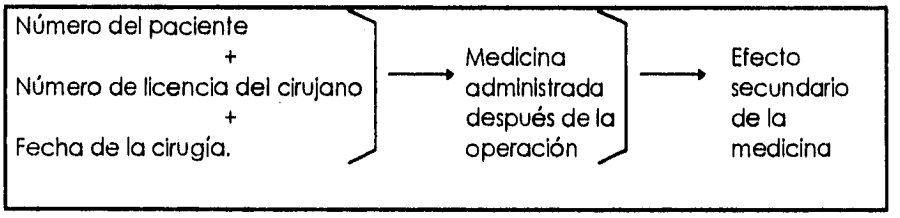


Tabla PACIENTE

Número del paciente	Nombre del paciente	Dirección del paciente
1111	John White	15 New Street, Nueva York, N. Y.
1234	Mary Jones	10 Main Street, Rye, N. Y.
2345	Charles Brown	Dogwood Lane, Harrison, N. Y.
48776	Hal Kane	55 Boston Post Road, Chester,
5123	Paul Kosher	Conn.
6845	Ann Hood	Blind Brook, Mamaroneck, N. Y.
		Hilton Road, Larchmont,.N. Y.

Fig. 1.4.4.-9a. Representación de Datos Utilizando el Modelo Relacional.

Relación en la tercera forma normalizada. La clave primaria es el número del paciente.

Tabla CIRUJANO

Número de licencia del cirujano	Nombre del cirujano
145	Beth Little
189	David Rosen
243	Charles Field
311	Michael Diamond
467	Patricia Gold

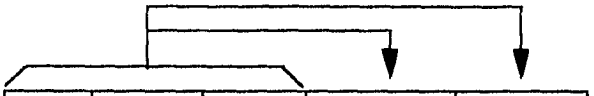
Fig. 1.4.4.-9b. Representación de Datos Utilizando el Modelo Relacional. Relación en la Forma Normalizada. La Clave Primaria es el Número de Licencia del Cirujano.

Supresión. John White recibe alguna otra medicina como resultado de haber presentado un severo prurito debido a la penicilina; el hecho de que haya sufrido ese efecto de la penicilina se puede registrar en la relación de la figura 1.4.4.-9d.

Actualización. El efecto secundario de cada medicina aparecerá sólo una vez en la relación de la figura 1.4.4.-9d, resolviendo así la falla de actualización.

Es importante notar que, en el proceso de reducción de una relación en la primera forma de normalización a un conjunto de relaciones en la tercera forma, no perdimos información. Esto significa que cualquier información que se pueda derivar de la relación en la figura 1.4.4.-4 también se puede derivar de las relaciones en las figuras 1.4.4.-9a, 1.4.4.-9b, 1.4.4.-9c y 1.4.4.-9d. La relación de la figura 1.4.4.-4 se puede construir combinando las relaciones de las figuras 1.4.4.-9a a 1.4.4.-9d. Otro punto que debe enfatizarse es que el proceso de reducción o normalización no es una función de los valores de los datos que aparecen en las relaciones en algún momento determinado: es una función de las relaciones entre los atributos. Para seguir el proceso de normalización, es absolutamente necesario que el diseñador de la base de datos entienda la semántica de la información. Dependiendo de las suposiciones sobre la dependencia funcional entre los atributos, el conjunto de relaciones en la tercera forma normalizada para la empresa será diferente.

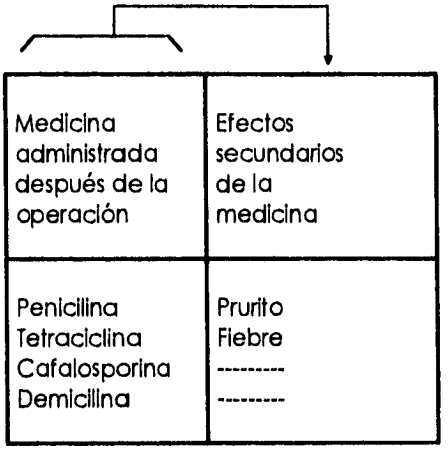
Relación PACIENTE y CIRUJANO



Número del paciente	Número de licencia del cirujano	Fecha de la cirugía	Cirugía	Medicina administrada despues de la operación
1111	145	Enero 1, 1977	Extirpación de calculos biliares	Penicilina
1111	311	Junio 12, 1977	Extirpación de calculos renales	----------
1234	243	Abril 5, 1976	Extirpación de catarata	Tetraciclina
1234	467	Mayo 10, 1977	Eliminación de trombosis	----------

Fig. 1.4.4.-9c. Representación de los Datos Utilizando el Modelo Relacional. Relación en la Tercera Forma Normalizada. La Clave Primaria es el Número del Paciente + Número de Licencia del Cirujano + Fecha de la Cirugía. Los Atributos que no son Clave son Cirugía y Medicina Administrada Después de la Operación.

Relación MEDICINA



Medicina administrada después de la operación	Efectos secundarios de la medicina
Penicilina Tetraciclina Cafalosporina Demicilina	Prurito Fiebre --------- ---------

Fig. 1.4.4.-9d. Representación de Datos Utilizando el Modelo Relacional. Relación en la Tercera Forma Normalizada. La Clave Primaria es Medicina Administrada Después de la Operación.

1. Relación PACIENTE. La clave primaria es el número del paciente.
2. Relación CIRUJANO. La clave primaria es el número de licencia del cirujano.
3. Relación CIRUGIA. La clave primaria es el número del paciente + el número de licencia del cirujano + la fecha de la cirugía.
4. Relación MEDICINA. La clave primaria es la medicina administrada después de la operación.

Como resultado del proceso de normalización, se derivan relaciones en la tercera forma normalizada. Estas relaciones representan a las entidades y sus relaciones y corresponden a la idea intuitiva de entidades: PACIENTE, CIRUJANO, CIRUGÍA y MEDICINA, estas cuatro relaciones representan al modelo conceptual.

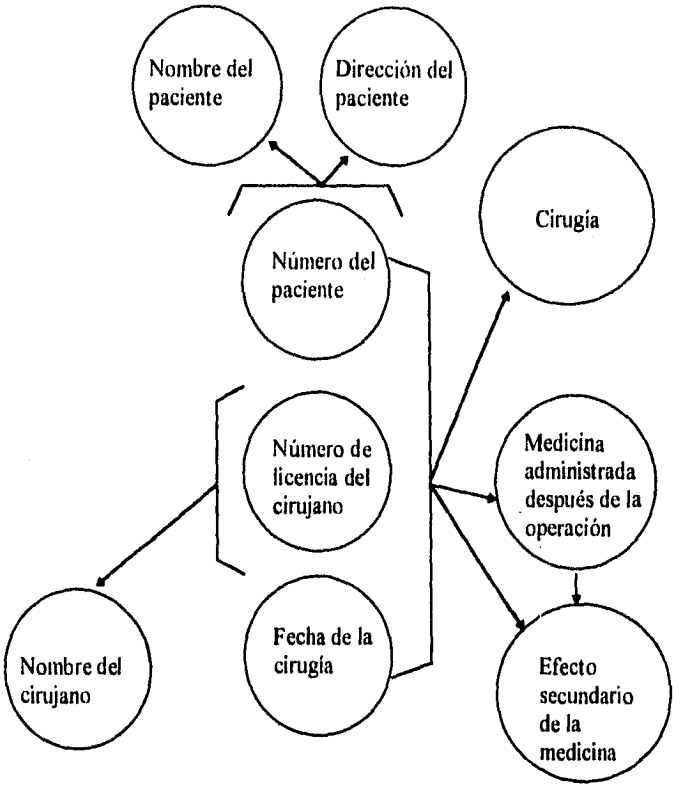


Fig. 1.4.4.-10. Cuatro Relaciones en la Tercera Forma Normalizada.

1.5 Software.

1.5.1. Características, Ventajas y Desventajas de los Diversos Paquetes de Comunicaciones.

SLIP.

El software PC-NFS sobre un puerto serial incluye una vista superficial de **Serial Line Internet Protocol** (SLIP).

El software PC-NFS permite a los usuarios de PC obtener el máximo de un ambiente en red, como es compartir archivos, impresiones, correo electrónico y otros recursos.

El software PC-NFS utiliza protocolos de comunicación llamados en su conjunto **Protocolo Internet (IP).** IP codifica la información de tal forma que pueda ser transmitida a través de distintos enlaces de comunicaciones. Si se conecta la PC a un servidor, utilizando una interconexión serial, el PC_NFS utiliza el SLIP para realizarla.

Un enlace serial entre una PC y un servidor pueda realizarse de dos maneras:

- Una conexión remota, con un modem y línea telefónica.
- Una conexión directa con un puerto serial RS-232.

Una conexión SLIP bajo una PC corriendo el software de PC-NFS conecta un servidor corriendo una implementación SLIP compatible. El servidor proporciona al usuario de PC las mismas aplicaciones y ambientes de red que se

encuentran disponibles, al usuario de PC usando una red *Ethernet* u otra red de área local.

Todos los servicios y aplicaciones del software PC-NFS trabajan sobre una conexión SLIP con algunas limitaciones de velocidad.

Cada red SLIP debe tener su propia dirección y nombre de Internet; cada PC y cada servidor de una red SLIP deben de tener su nombre y dirección en el ambiente, los nombres y direcciones deben de ser diferentes a los de la red.

¿Porqué usar SLIP?.

Una conexión remota SLIP proporciona la flexibilidad y conveniencia de utilizar PC-NFS lejos del servidor. Si la PC se encuentra conectada normalmente a una LAN, quizá se quiera conectar al servidor desde otro lugar y la conexión remota SLIP permite realizarlo.

Además, un cable serial es muchas veces la única manera práctica de conectar una PC a la red y una conexión SLIP permite realizarla mediante este tipo de conexión.

Sistema Sun.

El servidor PC-NFS SLIP esta diseñado solamente para máquinas que utilizan el sistema operativo SunOS. Este software proporciona soporte para Sun-3, Sun-4, y Sparc Station, utilizando el sistema operativo Solaris 2.x. Se debe asegurar que el sistema Sun se utilice como GATEWAY (puerta) tenga 1 megabyte o más espacio libre en disco para colocar el software de distribución, se pueden

instalar más puertos seriales en el sistema Sun para instalar la tercera parte de tarjetas seriales.

- En equipo PC

Una PC necesita MS-DOS 3.3 o posterior, y el software PC-NFS versión 3.5 o posterior. Por lo menos una PC en la red deberá usar el PC-NFS usando una tarjeta Ethernet para poder transferir los archivos de la PC al servidor. La PC requiere un cable RS-232 para conexión directa, o un modem en caso de que sea un acceso remoto.

Modems y Cables.

Para usar NFS y compartir archivos se requiere un modem que sea capaz de corregir errores, si el modem es de menor velocidad de 9600 esta acción es deshabilitada.

Las conexiones de cable y modem utilizan las siguientes velocidades: 1200, 2400, 4800 y 9600 baudios; la velocidad de 9600 proporciona la mejor conexión.

SIMPLE (Programa en C).

Debido a sus características C se considera muy eficaz para realizar un programa que ligue nuestra PC con la central telefónica.

C es un lenguaje de programación de propósito general que ha sido estrechamente asociado con el sistema UNIX en donde fue desarrollado puesto que tanto el sistema como los programas que corren en él están escritos

en lenguaje C. Sin embargo, este lenguaje no esta ligado a ningún sistema operativo ni a ninguna máquina, por lo que es muy utilizado para escribir programas de diversas disciplinas.

C proporciona una variedad de tipos de datos. Además existe una jerarquía de tipos de datos derivados, creados con apuntadores, arreglos, estructuras y uniones. Las expresiones se forman a partir de operadores y operandos; también proporciona un adecuado control de flujo; las funciones pueden regresar valores de tipos básicos, estructuras, uniones o apuntadores. Cualquier función puede ser llamada recursivamente. Las variables locales son normalmente "automáticas", o creadas de nuevo con cada invocación. Las variables pueden ser internas a una función, externas pero conocidas sólo dentro de un archivo fuente, o visibles al programa completo.

Como cualquier otro lenguaje, C tiene sus defectos. Algunos de los operadores tienen la precedencia equivocada; algunos elementos de la sintaxis pueden ser mejores. A pesar de todo, C ha probado ser un lenguaje extremadamente efectivo y expresivo para una amplia variedad de programas de aplicación.

C es conocido como un lenguaje de nivel medio. Nivel medio no tiene un significado negativo. No significa que C sea menos poderoso, manejable o menos desarrollado que un lenguaje de alto nivel. Es considerado de nivel medio, ya que combina elementos de lenguajes de alto nivel con la funcionalidad del ensamblador.

El código C es muy transportable. La portabilidad significa que se puede adaptar el software escrito para un tipo de computadora a otro tipo. El soporte C del estándar ANSI asegura la transportabilidad del código a otros entornos.

Todos los lenguajes de programación de alto nivel soportan el concepto de tipos de datos. Un tipo de dato define un conjunto de valores que una variable puede almacenar con un conjunto de operaciones que la computadora pueda realizar sobre esa variable, los tipos de datos comunes incluyen enteros, caracteres y reales. Aunque C tiene cinco tipos de datos incorporados, no es un lenguaje fuertemente tipado como Pascal o Ada, C siempre permitirá todas las conversiones de tipo. Por ejemplo, en la mayoría de las expresiones se pueden mezclar libremente caracteres y enteros. En general, los compiladores de C realizan una ligera comprobación de error en tiempo de ejecución, como la comprobación de los límites de array o la compatibilidad del tipo de argumento. Estas comprobaciones son responsabilidad del programador. La razón para esta carencia de comprobación en tiempo de ejecución se realiza de una forma más lenta que la ejecución del programa.

Como lenguaje de tipo medio, C permite la manipulación de bits, bytes y direcciones. Esta posibilidad hace de C un lenguaje adecuado para la programación a nivel de sistema donde son comunes estas operaciones.

El programa simple fue desarrollado por Simple Communications Program, y se modificó para adaptarlo a nuestras necesidades ya que se cuenta con el código fuente para diferentes compiladores del lenguaje C como son: Turbo C, Turbo C++, Borland C, Microsoft y otros.

Permite llamar a otro modem; transferir ficheros con diversos protocolos; emular diversas terminales; la posibilidad de responder a la llamada de otro modem; ajustar los parámetros de la línea; programar un proceso de conexión; etc. Es evidentemente un programa robusto y versátil.

Software: Simple.

Requiere: 192K de RAM, Sistema Operativo 2.0 o superior PC, conexión serial.

Comentarios: Programa diseñado para controlar la comunicación entre la PC y la central telefónica a través de modems compatibles.

Emula terminales ANSI y VT100, es fácil de transportar a otras plataformas de desarrollo y se cuenta con el código fuente.

ProComm.

Para un adecuado aprovechamiento de la unidad de comunicaciones, es necesario seleccionar un programa que se adapte de una parte al modem y de otra a nuestros intereses. Entre éstos, uno de los más completos y de propósito general es ProComm. Permite llamar a otro modem; transferir ficheros con diversos protocolos; emular diversas terminales; dar facilidades al usuario para registrar lo que ocurre durante la comunicación; la posibilidad de responder a la llamada de otro modem; ajustar los parámetros de la línea; programar un proceso de conexión; etc. Es evidentemente un programa muy versátil.

El ingenioso proceso de instalación de ProComm le da una variedad de opciones. Se puede escoger su configuración de modem de una lista de casi 100 cadenas de configuración y escoger protocolos prefijados para recepción de un menú de más de 10. Los protocolos de transferencia de archivos incluyen ASCII, CompuServe B Plus, Kermit, Xmodem, ZModem y más. Más tarde, puede cambiar más de 100 parámetros de la sesión, tales como el ajuste de la

velocidad en bits por segundo dentro de un rango de 300 a 115.200 o la selección de uno de ocho, puertos seriales definidos por el usuario.

Los menús expandibles de ProComm añaden otra dimensión al programa, llamando el directorio de llamadas, el menú de emulación de terminal y el servicio de ayuda dependiente del contexto. El menú de comandos le da una lista de las opciones de comunicación, que incluyen el modo de charla y el archivo de registro. Para ahorrar tiempo, puede evitar el menú de comando y ejecutar los comandos en cualquier momento usando las combinaciones de ALT-tecla.

El directorio de llamadas le deja añadir hasta 200 nombres, números y ajustes de sesión para sistemas a los que se llame frecuentemente. Puede crear tantos directorios de llamada como su disco pueda contener. ProComm le deja fijar una lista de números a los que llamará hasta que uno conteste, o puede hacer que siga llamando al mismo número hasta que conecte. El programa emula unas 33 terminales, que incluyen DEC VT52, VT102, VT220 y VT320, así como el IBM 3270 y 3101, ANSI y TTY.

ProComm incluye algunas opciones de seguridad. Aunque no tienen una contraseña para protección contra el acceso local, puede fijar contraseñas cuando use el programa como anfitrión remoto.

El lenguaje de guiones del programa ASpect, ofrece 208 verbos. Se puede usar la opción de aprendizaje del programa para construir un guión a partir de sus golpes de tecla. El editor que se incluye en ProComm puede manejar múltiples archivos sin problema y el mismo incluye una opción en búsqueda y reemplazo.

Además de sus protocolos incluidos. ProComm trabaja con tres protocolos externos definidos por el usuario. Una ventana que muestra el tiempo estimado, el conteo de bytes y el promedio de caracteres por segundo, junto con un gráfico de barra dejan seguir el progreso de la transferencia de archivos. ProComm emite un sonido cuando completa una transferencia.

Como podemos ver ProComm es un programa muy completo, que presenta varias opciones para realizar una comunicación adecuada, además, nos damos cuenta que este nos sirve para el propósito de éste tema. Esta conexión se puede realizar ya que ProComm cuenta con un lenguaje de programación el cuál permite automatizar el proceso de comunicación y ajustarlo a las necesidades requeridas.

Software: ProComm Plus, versión 2.0

Requiere: 192K de RAM, DOS 2.0 o posterior.

Comentarios: Un producto poderoso e intuitivo que presenta menús desplegables. Es un placer usar ProComm Plus. El mismo incluye apoyo para más de 100 modems, un modo de charla, y un directorio de llamadas que contienen hasta 200 nombres, números y ajustes de sesión. Por desgracia, no puede conectarse a las velocidades mas altas en pruebas de ZModem basado en caracteres.

DynaComm.

DynaComm es un poderoso producto de comunicaciones basado en Windows, puede ayudar en cualquier cosa, desde conectarse a un boletín electrónico

(BBS) hasta emular una terminal 3270 para tener acceso a un modem de red. La curva de aprendizaje es baja comparada con cualquier otro producto de Windows. El programa terminal que se incluye con Windows es realmente un subconjunto de DynaComm, también permite emular múltiples sesiones de mainframe con una sola ventana.

DynaComm incluye el apoyo para DDEs y una larga lista de emulación de terminales, junto con un completo lenguaje de guiones que permite hacer cosas con la creación de cajas de diálogo y menús para sus aplicaciones. *Future soft* recientemente añadió *ZModem* al producto.

La instalación no es complicada, pero debido a la compleja variedad de opciones de DynaComm, se debe correr primero el tutor. Las cadenas predefinidas de configuración de modem son limitadas, ya que sólo hay cuatro.

DynaComm muestra un icono de ayuda difícil de ignorar. Todos los iconos de DynaComm están bien diseñados y dan mucha información a simple vista. Esta versión no solamente añade ayuda dependiente del contexto sino también la capacidad para ver archivos del tipo GIF y RLE.

Permite modificar sus parámetros para la sesión seleccionando las opciones apropiadas. Estas le dejan fijar la razón de transmisión desde 110 hasta 19.700 bps, escoger COM1 o COM4 o una conexión a modem en red, ajustar el tamaño de palabra o desactivar la verificación de paridad o la detección de portador. El intercambio de señales maneja tanto control de flujo en software como en hardware, para enviar la información más rápido a los modems que tienen posibilidad de compresión de información.

DynaComm apoya 14 tipos de emulación de terminal. El trabajar con estos estándares comunes para terminales asegura la compatibilidad con cualquier servicio de comunicaciones.

DynaComm permite usar o modificar guiones preconfigurados de muestra y los mismos son útiles para aprender el robusto y complejo lenguaje de guiones de DynaComm. Este tiene su propio editor que permanece activo cuando una ventana de edición que contiene un guión esta activa. El lenguaje permite personalizar completamente las comunicaciones, y controlar sesiones de anfitrión entre PC's o entre PC's y mainframes 3270.

Podemos concluir, que DynaComm también nos proporciona varias opciones para poder realizar la liga entre una central telefónica y una PC, a través de su lenguaje de programación y utilizando sus características de programa de comunicaciones.

Software: DynaComm Asynchronous Edition, Versión 3.00.08

Requiere: 512 de RAM (se recomienda 640K), disco duro, Microsoft Windows 2.x o 3.0.

Comentarios: DynaComm incluye muchas opciones valiosas tales como el acceso a los modems de red y la emulación de terminal 3270, pero el dominarlo lleva tiempo. El versátil lenguaje de guiones puede crear una variedad de aplicaciones, incluyendo juegos.

Btrieve.

Netware Btrieve es un sistema de teclas indexadas para la administración, diseñado para manejo de datos de alto nivel y para una mejor productividad en la programación. Btrieve permite a las aplicaciones insertar, llamar, actualizar o borrar datos por el valor de una tecla o mediante la utilización de métodos de acceso secuencial o aleatorio. Btrieve ha sido implementado como un programa ejecutable, contenido en si mismo, que las aplicaciones pueden accesar mediante una función de llamada.

Btrieve puede correr en una workstation(versión cliente-servidor) o en un servidor (versión servidor-base). La versión de cliente-servidor ejecuta todos los procesos en la workstation. Accesa a todos los archivos mediante funciones de llamadas al sistema operativo. Estas llamadas al sistema operativo son ejecutadas ya sean localmente (para archivos locales) o redirigidas al servidor (para archivos en el servidor).

En la versión Servidor-base el Administrador de Grabación Btrieve se ejecuta en el servidor y un programa residente es ejecutado en la workstation. Existen programas residentes para cada tipo de plataforma que ejecutan generalmente: DOS, OS/2, y para Windows. El programa ejecutable maneja datos de entrada salida I/O entre la workstation y el servidor de la red. El Administrador de Grabación Btrieve que se ejecuta en el servidor maneja los datos I/O con el sistema de archivos.

La versión de Btrieve que corre en un servidor se ejecuta en un servidor Novell Netware. La versión cliente-servidor puede ejecutarse sobre varias plataformas y distintos sistemas operativos como son Windows, DOS y OS/2. Las aplicaciones

y programas para la versión de cliente-servidor de Btrieve podrán ser ejecutadas en la versión de servidor y viceversa.

Además de proveer un sistema de administración de grabaciones, en su versión basada en servidor, para las workstation. Btrieve también acepta llamadas de otras aplicaciones basadas en servidor.

Existen cuatro archivos básicos en el paquete de red de Btrieve, estos programas son:

- BSPXCOM.NLM
- BTRIEVE.NLM
- BTRIEVE RESIDENTE
- BROUTER.NLM

Btrieve.nlm es una librería de las funciones de Btrieve en red, reside en el servidor de archivos. Ambas versiones de Btrieve pueden accesar a Btrieve en red.

La versión basada en cliente se comunica con la versión de Btrieve en servidor haciendo uso del programa residente de Btrieve y de BSPXCOM.

Las Aplicaciones basadas en servidor pueden mandar llamadas a Btrieve al servidor local o a servidores distintos. Para llamadas de Btrieve en el mismo servidor donde se origino la petición se puede usar a Btrieve directamente o hacer uso de BROUTER. Las llamadas a Btrieve hechas desde otro servidor pueden accesar al Btrieve remoto haciendo uso de BROUTER y el BSPXCOM remoto.

Btrieve.nlm

Btrieve.nlm es una librería de funciones de Btrieve, debe de ser cargado en todos los servidores de archivos que almacenan archivos.

BSPXCOM realiza llamadas a Btrieve desde una workstation cliente o mediante un BROUTER remoto. Especialmente, estas llamadas realizan las siguientes tareas:

Realiza todas las I/O de disco para los archivos de Btrieve almacenados en el servidor donde las copias accesibles de Btrieve se encuentran.

Implementa o retira todos los candados de elemento y de archivo, en el servidor donde la copia de Btrieve accesada se encuentra.

Graba las peticiones a Btrieve que resulten en cambios de un archivo.

BSPXCOM.NLM.

Para hacer uso de BSPXCOM.NLM, debe de estar cargado en el servidor de archivos que se accese. BSPXCOM permite que las llamadas de Btrieve que se originen fuera del servidor se comuniquen con Btrieve.

Cuando Btrieve es accesado la llamada puede originarse en: una copia distinta del Btrieve del servidor o bien en una versión diferente de Btrieve, como puede ser la versión basada en cliente de una workstation.

Cuando la petición proviene de otro servidor, BPSXCOM recibe la petición de la otra copia de BROUTER en el otro servidor y realiza las llamadas de función apropiadas en Btrieve para procesar la petición.

Cuando la llamada proviene de una workstation, BPSXCOM recibe la petición del programa residente en la *workstation* y ejecuta las funciones apropiadas para que Btrieve pueda realizar el proceso.

Después de que una petición ha sido procesada, BPSXCOM comprime la petición para que pueda ser transmitida a una copia del programa residente en una *workstation* o a una copia de BROUTER en otro servidor.

BROUTER.NLM.

BROUTER.NLM debe de ser llamada en el servidor. Es una aplicación basada en servidor que permite que cualquier otra aplicación cargada en el servidor se comunique con Btrieve haciendo uso del programa BPSXCOM. BROUTER realiza las siguientes tareas:

Provee acceso convencional basado SPX para poder llamar archivos Btrieve de cualquier otro servidor en la red.

Mantiene un sistema único de identificación de código para cada aplicación para que los candados, transacciones y otros mecanismos de control de acceso puedan trabajar sin conflictos en toda la red.

Programas de Utileria.

Se proporcionan dos programas mas: BCONSOLE.NLM, una utileria de monitoreo de consolas, y BSETUP.NLM, una utileria que permite la instalación y configuración .

Acceso a BTRIEVE.

El programa de Btrieve en red funciona como si fuera una subrutina de una aplicación. Btrieve Netware (que es el nombre del programa en red) da apoyo a los siguientes métodos de acceso a Btrieve:

Una aplicación en una workstation puede accesar Btrieve mediante un programa residente.

Una aplicación de un servidor puede accesar a Btrieve directamente en el mismo servidor al llamar el punto de acceso.

Una aplicación puede accesar Btrieve en otro servidor vía el programa BROUTER.

Estos métodos de acceso son explicados a continuación:

APLICACION CLIENTE: Aplicaciones basadas en cliente corren en una aplicación y se comunican con Btrieve mediante un programa residente. Los siguientes pasos ilustran los flujos de control cuando una aplicación basada en cliente accesa a Btrieve mediante el programa residente:

1. La aplicación llama a Btrieve mediante una función de llamada.

2. El código de interfaz de la aplicación permite la liga con el programa residente.

3. El programa residente compacta la llamada como un mensaje en la red, determina que servidor deberá de recibir la llamada, y manda la llamada al programa.

4. BPSXCOM que se encuentra residente en el servidor.

5. BPSXCOM recibe el mensaje de la red, valida los parámetros, y ejecuta la petición al realizar llamadas de función a Btrieve. Dependiendo de la naturaleza de la petición, esto puede incluir una operación de memoria o una operación I/O a un dispositivo de almacenamiento. BSPXCOM proporciona el resultado de la operación al programa residente en la *workstation*.

6. El programa residente proporciona los datos adecuados y código de status a las variables residentes en memoria de la aplicación que llamó a Btrieve. después regresa el control a la aplicación.

Control de Flujos para una Llamada Btrieve Remota

Los siguientes pasos son los que se siguen para realizar el flujo de control cuando una NLM realiza una llamada remota a Btrieve. Los componentes requeridos en cada servidor para procesar una llamada remota:

1. La aplicación NLM local realiza una llamada a Btrieve (usando una función de llamada) que es llevada por el BROUTER.
2. BROUTER lleva la petición de Btrieve usando el BSPXCOM al servidor remoto.
3. BSPXCOM realiza la petición a Btrieve ejecutando una función de llamada.
4. El Btrieve en el servidor remoto regresa los datos apropiados. Así como su código status al programa BPSXCOM.
5. BPSXCOM regresa los datos y el código de status al BROUTER en el servidor local donde la llamada se originó.
6. BROUTER proporciona estos resultados al NLM que realizó la llamada. Los resultados son depositados en la memoria del NLM designada por los parámetros suministrados por Btrieve. El control es regresado a la aplicación.

Características	SLIP	PROCOMM	DYNACOM	BTRIEVE	SIMPLE
IP	•	•	•	•	
CONEXION REMOTA	•	•	•	•	•
CONEXION SERIAL	•				
SOFTWARE ADICIONAL	•			•	
FLEXIBILIDAD		•	•		•
PROPOSITO ESPECIFICO PARA EL SISTEMA					•
EMULACION DE TERMINALES	•	•	•	•	•
OTROS PROTOCOLOS		•	•	•	•
CODIGO FUENTE					•

Fig. 1.5.1.-1. Sumario de Características del Software de Comunicación.

Conclusión.

A pesar de todas las ventajas y características que nos proporcionan los programas de comunicación como SLIP, PROCOMM, BTRIEVE y DYNACOMM, observamos que la interfaz entre estos y las demás funciones del sistema era muy complicada y por ello fue conveniente utilizar Simple, el cuál será adecuado para satisfacer las necesidades específicas que deseamos, es decir, que se pueda realizar la comunicación entre la central telefónica y una PC, para poder llevar a cabo el monitoreo de los teléfonos públicos.

1.5.2. Características, Ventajas y Desventajas de los Diversos Manejadores de Bases de Datos.

En una base de datos por relación, cada indagación de base de datos produce una serie de uno o más registros. El juego es realmente un tipo de datos complejo (una relación) que está sujeto al rango completo de operaciones de relación. El resultado es que algunas bases de datos de multitabla son "por relación" pero otras no.

Tanto las bases de datos por relación como las de multitablas simples tienen ventajas sobre los productos de bases de datos de archivo único, ya que le permiten a los usuarios manejar con eficiencia problemas de administración de información complejos. En una base de datos de archivo único, toda la información va en una misma tabla o en varias tablas que se agrupan de forma débil. Esto es algo flexible, ya que se esta limitando a un número fijo de elementos de línea por cada registro. Además, cuando se usa el mismo fabricante más de una vez, se debe reingresar la información del fabricante, malgastando tiempo y espacio de disco.

Una ventaja de la técnica de multitabla es la flexibilidad. Puede tener tantos elementos de línea como quiera, y puede añadir reportes o procesos que usen la tabla del fabricante sin tener que modificar su estructura. Las bases de datos de archivo único no pueden adaptarse tan bien a los cambios, ya que no pueden derivarse tan fácilmente las colecciones individuales de información significativa.

Otra ventaja de las multitablas es su facilidad de actualización. Cuando se cambia una dirección, por ejemplo, el cambio de esa dirección se refleja en

cada lugar que esté relacionado. En la base de datos de archivo único, se puede cambiar una dirección en una sola estructura, pero cuando mira hacia atrás no sabe cuál cambio fue el más reciente.

La ventaja de la técnica de archivo único es su simplicidad. Trabaja al estilo de los archivos de gavetas del mundo real, a los que estamos tan acostumbrados, pero al igual que estos, mientras más información se añade, más desorganización existe. Con una base de datos de multitabla, cuando se añade información, no se aumenta la desorganización.

Entre las bases de datos de multitabla hay varios grados de relación. Las que tienen más relación tienen un acceso completo sin procedimiento a la información usando SQL, QBE (indagación con ejemplos) o comandos equivalentes sin procedimiento. Una vez más el término "sin procedimiento" significa que el sistema determina el mejor método para tener acceso al archivo, y no el programador o el usuario. Estos incluyen sistemas tales como SQL y QBE para suplementar sus interfaces nativas "con procedimiento" pero requieren programación de procedimiento para muchas operaciones. Algunos sistemas, como FoxPro, proveen sólo unas pocas opciones de relación, tales como escritores de reportes sin procedimiento y herramientas de indagación, pero requieren programación con procedimiento para todas las otras operaciones. Las bases de datos sin procedimiento ofrecen ventajas de productividad en tanto que los lenguajes de base de datos con procedimiento ofrecen mayor flexibilidad.

Los sistemas de administración de bases de datos en PC vienen en todos los modelos y tamaños. Por ejemplo, servidores de bases de datos de LAN tales como SQL Server y OS2/EE Database Manager se especializan en la

administración y protección de la información para grupos de trabajo. Los sistemas de desarrollo de aplicaciones tales como FoxPro y dBASE IV se especializan en herramientas y lenguajes para construir aplicaciones. Los administradores de bases de datos por relación sofisticados tales como Paradox se especializan en proveer habilidades interactivas y herramientas integradas para aplicaciones. Los administradores de bases de datos de gráficos tales como Superbase 4 ofrecen herramientas para construir aplicaciones que emplean fuentes e imágenes escalables. Además, los sistemas tales como DataEase, Paradox, y dBASE IV hacen también el papel de preprocesadores (fron-end) para servidores de bases de datos, con el fin de compartir armoniosamente la información corporativa.

Para usar un término tradicional de la administración de bases de datos, puede "descomponer" el género de administración de bases de datos en categorías, identificando al usuario de un producto así como su funcionalidad. Estas categorías incluyen:

- Desarrollo de aplicaciones programables para profesionales de programación de bases de datos.
- Administración interactiva de información y facilidades de administración para profesionales de la computación.
- Facilidades interactivas de administración de información para usuarios finales.
- Herramientas de creación de aplicaciones para usuarios finales.
- Preprocesadores (front-ends) para servidores de bases de datos.
- Combinaciones de lo anterior.

Excepto los servidores de bases de datos y los "front-ends", generalmente estas categorías no vienen escritas en la caja del producto, y si bien puede que dos productos parezcan similares a primera vista, pueden estar destinados a dos clases distintas de usuarios. Un fabricante puede decir que produce un sistema que se adapta a todas las necesidades, pero es casi seguro que el producto está orientado hacia un tipo específico de administración de bases de datos.

Algunas características de base de datos por relación, se muestran en la figura 1.5.2.-1.

FACILIDADES DE INDAGACION	DATAEASE	DBASE IV	FOXPRO	PARADOX	CODEBASE
Operadores de relación					
Diferencia (MINUS)		X			X
División (DIVIDEBY)		X			X
Intersección	X	X		X	X
Unión	X	X	X	X	X
Natural		X	X	X	X
Fuera de	X	X		X	X
Producto (TIMES)	X			X	
Proyecto	X			X	
Selección (WHERE)	X	X		X	X
Unión	X	X		X	X
Opciones de Lenguaje de Datos					
Tipo de lenguaje:					
Compilado		X	X		X
Interpretado	X	X	X	X	X
Trabaja con lenguaje SQL integrado		X		X	

Fig. 1.5.2 - 1. Características de las Bases de Datos por Relación.

FACILIDADES DE INDAGACION	DATAEASE	DBASE IV	FOXPRO	PARADOX	CODEBASE
Funciones de Lenguaje de Base de Datos					
Funciones agregadas:					
Promedio	X	X	X	X	X
Conteo	X	X	X	X	X
Fecha y hora	X	X	X	X	X
Financiera	X	X	X	X	X
Lógicas	X	X	X	X	X
Máximo	X	X	X	X	X
Mínimo	X	X	X	X	X
Totales	X	X	X	X	X

Selección del Editor si-X, no en blanco

Fig. 1.5.2 - 1. Características de las Bases de Datos por Relación. (Continuación).

Dataease.

DataEase International Inc., requiere 640K de RAM, DOS 3.1 o posterior, promete ayudarle a crear sistemas sofisticados de bases de datos con un mínimo de esfuerzo y tiempo.

- DataEase se basa mucho en menús y en pantallas que le piden la entrada de datos para realizar su interfaz. El propio programa se construyó usando herramientas similares a las que usará al crear aplicaciones para DataEase. Como con otros programas de bases de datos, se puede construir una

secuencia de comandos para que se ejecuten, pero en su mayoría, DataEase toma una estrategia más al estilo sin procedimiento, mediante la cual el usuario define las operaciones diseñando pantallas y llenando los espacios en blanco.

- El elemento fundamental de una aplicación de DataEase es el modelo, que tiene el doble papel de estructura de tabla de datos y el modelo de entrada de datos. Se crea un modelo pintando encabezamientos y campos en la pantalla. Una tecla de función abre una pantalla. Una tecla de función abre una pantalla de definición de campo, y el nombre del campo está prefijado al del texto inmediatamente adyacente al mismo. Si entra "Name" (nombre) y aprieta F1, estará listo para definir un campo llamado "Name". Después que define el tipo del campo, su longitud, etc, se guardan esos ajustes. Entonces el sistema lo regresa a la pantalla, en la que el nuevo campo definido aparece como un bloque.

- DataEase depende mucho de las opciones de definición de campo para lograr su amplio rango de validación compleja de entrada de datos y sus funciones de edición filtro, que en otros productos requieren código especial para lograrse. Existen las funciones típicas de campos que deben llenarse, valor único y funciones de campo calculadas, pero también puede llegar a definir rangos de valores permitidos, cadenas numéricas con formato (incluyendo número de teléfono, número de seguridad social y formatos exclusivos) y niveles de seguridad para controlar el acceso de lectura y escritura y evitar que se entre información sin autorización.

- También puede especificar la numeración secuencial automática de registros para asegurarse que nunca se usa un número dos veces. Otras

opciones de definición de campo le dejan ver los valores correspondientes en una tabla relacionada, tomar el valor de campos relacionados y simplemente verificar que el valor que entró existe en otra tabla.

- DataEase tiene una útil característica llamada una pantalla de multimodelo, que muestra el contenido de dos o más modelos en la misma pantalla. Primero, se define el modelo "patrón" y entonces un submodelo que muestra la información relacionada de otro modelo. Entonces fija el número mínimo y máximo de líneas que quiere que aparezcan en el submodelo y su tamaño crecerá o se comprimirá, según lo requiera el número total de registros que concuerden. Se puede entrar, editar y borrar registros en el submodelo como lo haría en un modelo regular. Para construir esto en un producto típico como dBASE llevaría horas de programación y páginas de código. Con DataEase, sólo le lleva unos minutos el pintar la pantalla y llenar los modelos. La opción de multimodelos es la única área de DataEase que ofrece una verificación de integridad de referencia automática, pero aún esto es más que lo que otros productos le ofrecen en esta clase. Puede hasta especificar las acciones que quiere que el programa permita o prohiba. Por ejemplo, puede prohibirle la anulación de un registro patrón si hay registros hijos relacionados. Por otro lado, puede hacer que los cambios que se hacen a un patrón se pasen a los hijos, una opción que hace muy fácil el actualizar todas las estructuras relacionadas al mismo tiempo.

Algunas otras de las características de la bases de datos relacionales, se muestran en la figura 1.5.2 - 2.

FUNCIONES DEL LENGUAJE DE BASE DE DATOS	DATAEASE	DBASE IV	FOXPRO	PARADOX	CODEBASE
Funciones matemáticas					
Aritmética	X	X	X	X	X
Numérico	X	X	X	X	X
Transcendentales		X	X	X	X
Trigonométricas	X	X	X	X	X
Funciones de sistemas operativo	X	X	X	X	X
Funciones estadísticas:					
Desviación estándar	X	X	X	X	X
Variable	X	X	X	X	X
Funciones de cadena	X	X	X	X	X
Capacidades de macros					
Grabación automática	X	X	X	X	
Lenguaje de macros	X			X	
Seguridad					
Codificación de la aplicación	X		X	X	X
Codificación de los datos	X	X		X	X
Protección de contraseña:					
Base de datos	X	X			
Diccionario	X				
Usuario	X	X		X	X
Niveles de acceso restringido:					
Bases de datos	X	X		X	X
Diccionario	X			X	X
Campo	X	X		X	X

Selección del Editor si-X, no en blanco

Fig.1.5.2 -2 Características de las Base de Datos Relacionales.

- La técnica se basa en modelos de DataEase hace que otras funciones sean también fáciles de usar. Las relaciones entre los modelos se definen en un modelo. Se puede basar la concordancia en uno o más campos, y si bien puede que sea más lento, se puede hasta definir las relaciones basadas en campos que no tengan índices. Los niveles de seguridad del usuario, el apoyo para impresoras, las funciones para importar datos y la definición personalizada de menús se logran todas llenando modelos. También se pueden elaborar reportes sofisticados sin crear un procedimiento. Los Reportes Rápidos (Quick Reports) de QBE de DataEase se definen llenando una serie de modelos para la selección de registros y campos. Es posible usar una tecla dedicada para salir a un modelo relacionado y especificar allí los campos.

- DataEase puede ajustar automáticamente su reporte en formato de columna con encabezamientos y notas marginales de grupo, o usar la misma composición que su modelo. Si lo prefiere, puede construir su propia composición a partir de cero o a partir de una de las composiciones automáticas.

- Si no puede realizar su tarea usando uno de los procedimientos automatizados, tendrá que usar el Lenguaje de Indagación de DataEase (DQL), que le permite reportar, actualizar, borrar por grupos y automatizar la manipulación de su información. Hay tres modos que puede usar. El modo de bajo nivel, con preguntas, lo guía a través del ensamblaje de las secuencias de comando al preguntar cuáles son los comandos de uso más frecuente. Ya que todas las opciones están en la pantalla al mismo tiempo, nunca se pierde tiempo buscando nombres de campos u opciones de comando.

- El modo de alto nivel trabaja de la misma manera, pero presenta todas las opciones, dándole más acceso a los comandos más poderosos. Finalmente, está el modo de edición de pantalla completa, en el que puede moverse libremente por la pantalla, entrando comandos a medida que los necesita. DataEase verifica automáticamente su código buscando una sintaxis correcta antes de guardarlo. Si bien esto es conveniente, hay que rodear con comentarios las áreas que todavía tengan errores antes de que pueda guardar un trabajo aún no terminado.

- En términos de ejecución, DataEase no fue uno de los productos evaluados más rápidos. En la prueba de 100,000 registros, realmente mostró uno de los tiempos más rápidos (excepto cuando cargaba y daba índices).

- Cuando se considera el tiempo que ahorra en la creación y mantenimiento de un sistema complejo de bases de datos al usar DataEase y se trabaja en instalaciones de único usuario y de red, donde no se tienen que codificar ningún mecanismo de reservación para multiusuarios. DataEase lo maneja automáticamente, incluyendo las actualizaciones automáticas de la pantalla, si así lo quiere. También hay una versión separada del producto que usa la memoria extendida.

dBASE IV.

dBASE IV, versión 1.1, requiere 640K de RAM, DOS 2.1 o posterior, una base de datos con muchas opciones para programadores.

- dBASE IV se administra casi completamente por el Centro de Control, una herramienta de interfaz que le deja escoger entre varios objetos de base de

datos tales como tablas, vistas, modelos, reportes, diseños de etiquetas y hasta programas. El Centro de Control le deja cambiar fácilmente de un módulo a otro sin pasar por capas de menú, pero no le da la habilidad de ver archivos en ventanas múltiples superpuestas como lo hace Foxpro, o aún ver los resultados de una indagación mientras se opera el menú principal como lo hace Paradox. La interfaz pide a gritos el apoyo del ratón. Los Generadores de Reportes y Etiquetas ofrecen interfaces LQVELQO para diseñar etiquetas y reportes.

- El módulo de "Indagación con ejemplos" (QBE) se integra en el Centro de Control y le deja buscar juegos de estructuras, verificando los campos que quiere ver en un modelo de QBE. Se pueden establecer las condiciones de selección entrando simples expresiones y fórmulas en una caja de condición, y recibir las finales correspondientes con unos pocos golpes de teclas QBE. Se pueden establecer las condiciones de selección entrando simples expresiones y fórmulas en una caja de condición, y recibir las filas correspondientes con unos pocos golpes de teclas. Le deja usar "alias" para los nombres de los campos de salida en el resultado. Junto con la capacidad para realizar cálculos, uniones de multitablas, sumarios y ordenamientos, el QBE de dBASE IV también le dejará ejecutar búsquedas parciales y aproximadas.

- El módulo de QBE no realiza realmente una búsqueda, sino que genera un programa en lenguaje de dBASE que usa los comando SET FILTER TO (fije el filtro a) y SET RELATION TO (fije la relación a) para que establezcan las condiciones de búsqueda para obtener los registros y llama a dBASE para que lo corra.

- Algo más, dBASE crea una vista de los registros pedidos y si examina (usando BROWSE) los resultados de la indagación derivados de múltiples tablas, los resultados son de "sólo lectura" y no se pueden modificar. Finalmente, dBASE insiste en que debe hacer disponible un índice que concuerde con el criterio de búsqueda de registro. Si el usuario no crea el índice que concuerde con el criterio de búsqueda del resto. Si el usuario no crea el índice, dBASE IV lo creará cuando se guarde la indagación, un proceso que consume tanto tiempo como espacio de disco. Esto es molesto cuando se crea una indagación pero no cuando se ejecuta. Se convierte en un problema muy serio cuando quiere correr indagaciones específicas en una base de datos extensa.

- En las pruebas de ejecución, los resultados de dBASE la colocaron en segunda en sobrecarga (overhead) de salida (debido al cache de disco), entre las tres mejores en selección, y la más lenta en algunas partes de la prueba de unión de 100,000 registros. En cuando a carga, índices, proyección y reporte, dBASE logró resultados promedios.

- dBASE IV ofrece apoyo automático para multiusuarios, con reservación de archivo y registro, y una facilidad de volver a probar (autoretry) para habilitar registros que se hayan reservado. Hay disponibles refrescos de pantalla de tiempo real mediante la facilidad de autorefresco. También, los índices múltiples de dBASE IV usan un solo archivo de disco para mantener hasta 47 diferentes claves de índice, que se abren y actualizan automáticamente junto con el archivo correspondiente de base de datos.

- A pesar de estas innovaciones, la mayoría de las opciones de dBASE IV solamente la ponen a la par de otras bases de datos. Ciertamente no es tan

fácil de usar como Paradox o tan poderosa como FoxPro. Aunque opera con sólo 450K de RAM, dBASE IV no aprovecha todavía la memoria expandida o extendida.

Foxpro.

Foxpro Software Inc., requiere 420K de RAM, DOS 2.0 o posterior, una base de datos de alto rendimiento, compatible con dBASE y con una elegante interfaz y poderosas facilidades de desarrollo de programas.

- Foxpro Software, ha reemplazado a dBASE como el líder de la industria en innovaciones, potencia y velocidad, una combinación invencible que le ganó a una versión anterior del producto una Selección del Editor. La interfaz de Foxpro, que opera con ratón y trabaja en modo de caracteres, casi cumple con los requisitos de CUA (Accesos Común para el Usuario) con sus menús desplegables, y cajas de diálogos en pantalla, cajas de lista y barras de movimiento. Una útil calculadora en pantalla, un calendario, y hasta una tabla de ASCII completamentan a los menús estándares.

- Foxpro tiene más de 200 mejoras en su dialecto del lenguaje de dBASE, y ofrece opciones poco usuales, como los campos de comentarios de longitud variable. Hay funciones disponibles para realizar operaciones de búsqueda y reemplazo dentro de un campo de texto y realizar E/S de DOS de bajo nivel.

- Con FoxPro, la creación de una nueva aplicación es algo simple. Puede usar Fox-View para diseñar pantallas y generar un código de aplicaciones, FoxCode para diseñar aplicaciones con un lenguaje de plantillas, y FoxDoc para documentar y hacer referencias cruzadas en su código fuente. El

lenguaje de plantillas de FoxCoder le deja crear plantillas reusables para aplicaciones que le evitan pasar por los detalles de ensamblar una aplicación. FoxDoc puede crear reportes con referencias cruzadas en su programa, mostrar un organigrama de su estructura y mantener un diccionario de datos. FoxPro creará un código de lenguaje de programación con cada selección de menú, y al tiempo que se genera, muestra el código en una ventana en el fondo de la pantalla. Es posible ver los resultados de la generación de código, editarlo y probar los resultados inmediatamente. Hay disponibles ventanas adicionales para correr el trazador de errores y cortar y pegar ejemplos de código entre ventanas de código fuente o desde las ventanas de archivos de ayuda hacia su programa.

- FoxPro es también el líder del rendimiento, obteniendo el primer lugar en las pruebas de ejecución de Carga grande y pequeña y de índice, la prueba de proyecto, la de reporte y la de sobrecarga (overhead) de salida. Además, se situó entre las primeras en las pruebas de unión y enlace.

Paradox.

Paradox, versión 3.5, Borland International Inc., requiere 512K de RAM, DOS 2.0 o posterior. Paradox tiene su propia técnica para crear bases de datos, pero una vez que se acostumbre, le provee una impresionante combinación de facilidad, poder y velocidad.

- "El sistema de administración de base de datos del usuario que piensa" Paradox es todavía uno de esos raros programas que es del gusto tanto de los novatos como de los usuarios experimentados. En vez de comenzar con la vista tradicional de campos y registros, generaliza por dBASE y sus

descendientes, los diseñadores de Paradox salieron con una vista que ellos consideraban que se comprendería mejor. El programa usa "tablas" en lugar de archivos o relaciones, y éstas se muestran en la pantalla en filas y columnas, como un rango de base de datos en una hoja de cálculo de Lotus 1-2-3.

- Sin embargo, la similitud no termina aquí. Cuando se hace una indagación, los registros que cumplen con el criterio de selección "caen" al fondo de la pantalla, para crear una tabla temporal llamada "ANSWER" (respuesta). Luego puede usar esta tabla como una fuente de información para reportes o guardarla bajo un nuevo nombre y realizar operaciones adicionales sobre la misma.

- La operación de indagación es tan visual y directa como el resto de Paradox. Como usa su propia versión de "Indagación con Ejemplos", el programa le deja entrar instrucciones condicionales en las columnas de una tabla para definir los criterios de selección. Unas marcas de verificación en las columnas indican cuáles campos y constantes para crear nuevos campos sobre la marcha.

- Y si necesita enlazar dos o más tablas, sólo tiene que colocar los valores que correspondan en las columnas que quiera combinar y el programa hace el resto. No necesita programaciones complicadas, relaciones predefinidas o la creación previa de complicados índices de campos.

- Ya que las repuestas aparecen en una nueva tabla, no existe el riesgo de modificar la información original. El mismo proceso de indagación le permite realizar actualizaciones y borraduras en masa, pero siempre de forma segura,

Paradox mantiene los originales de todas las estructuras modificadas o borradas en una tabla temporal, de forma que puedan anular los cambios rápidamente, si su indagación hiciera algo que no consideró. También hay un rápido acceso a una variedad de opciones sofisticadas para entrada de datos, tales como la verificación de rangos, la validación con búsqueda en otras tablas, y las imágenes de información para el formato de las entradas.

- Se puede crear un "reporte instantáneo" o usar la opción normal de reporte. El programa se basa en una técnica de "banda" para la definición de un reporte, en la que cada banda representa un nivel de agrupamiento. ¿No quiere solamente una lista de nombres y números? Paradox puede crear una amplia variedad de gráficos usando su información y estos incluyen los de barra, escalonados, de línea y los circulares.

- Paradox le da muchas formas de programar, que varían en los grados de complejidad. Los programas de Paradox se llaman guiones, y la forma fácil de crear uno es simplemente grabándolo. El resultado es un archivo de texto que puede editarse con el editor de guiones de Paradox o cualquier otro editor de texto. Luego puede ir a modificarlo o añadirlo a su guión. De la misma manera, puede grabar y reproducir indagaciones.

- Finalmente, puede escribir sus propios procedimientos directamente en PAL, el Lenguaje de Aplicaciones de Paradox. Este es un lenguaje completo que incluye docenas de distintos comandos y opciones. También puede obtener el Paradox Engine (la máquina de Paradox), una biblioteca de funciones en C y procedimientos en Pascal, que puede usar en aplicaciones de usuarios únicos y múltiples.

- Paradox es muy rápido. En las pruebas de ejecución de las PC Magazine Labs, el rendimiento de Paradox estuvo cerca del más rápido en muchas de las operaciones, aún cuando tuvo que dar pasos intermedios con tablas intermedias. Esta velocidad se hizo más evidente en las tareas de 10,000 registros; en las de 100,000 no se situó tan a la delantera del grupo. Sin embargo, aún con aplicaciones muy grandes de base de datos, probablemente encontrará que Paradox es muy rápido. Hasta venció a FoxPro en algunas uniones complejas.

Diferencias Filosóficas.

- Pero el programa tiene algunos problemas. La filosofía en la que se basan las estructuras, es bastante distinta de las herramientas más tradicionales de programación de bases de datos y requieren un tiempo mayor para que las aprenda.

- Esto todavía deja algunas ideas a las que hay que adaptarse. Por ejemplo, si tiene una indagación que ejecute frecuentemente y quiere darle formato en una composición dada para un reporte, tiene que pasar por muchos pasos. Primero debe crear una tabla falsa con una estructura idéntica a la de la tabla ANSWER que resultó de la indagación, y luego crear su formato de reporte para esa tabla. Entonces crea un gulón que abrirá la tabla fuente, ejecuta la indagación, guarda la tabla ANSWER bajo un nuevo nombre, copia el formato del reporte de la tabla falsa a la tabla ANSWER a la que se cambió el nombre, y entonces ejecuta el reporte. Esto no es difícil de hacer una vez que se acostumbre, pero sí hay que acostumbrarse.

- Paradox también tiene capacidades de multiusuario. Puede usarse el mismo producto como una versión individual o como una versión de multiusuario.

Codebase.

Codebase, requiere 640K de RAM, DOS 2.1 o posterior, una base de datos con muchas opciones para programadores.

Es una herramienta para el desarrollo de bases de datos, por medio de librerías para distintos compiladores del lenguaje C, C++, Visual Basic y Delphi, compatible con dBASE III PLUS, dBASE IV, FoxPro y Clipper con los cuales se puede crear accesar o modificar archivos de datos, de índice o memo. Las aplicaciones pueden correr en red, esta es detectada automáticamente y se usa.

Las aplicaciones en CodeBase pueden compartir archivos con las que se encuentren corriendo en dBASE, FoxPro y Clipper. También se puede compartir datos con estas aplicaciones usando el protocolo locking apropiado.

Cuenta con un conjunto completo de funciones para relación y consulta. Estas funciones utilizan una tecnología de optimización de Bit que reduce notablemente las tareas de consulta y generación de reportes.

Para usuarios de C++ tiene una interfaz de clases y una nueva clase de string. La interfaz de clases utiliza la sintaxis de C++. Con la nueva clase para string es muy sencillo manipular texto y campos. Algunas otras características de CodeBase son:

Sorprendente velocidad:

Consulta de 500,000 registros en la base de datos en fracción de segundos, o agregar mas de 1000 registros por segundo a la base de datos es sencillamente rápido.

XBase Compatibilidad:
Las aplicaciones en CodeBase son compatibles con los archivos de datos, índices y memo de Fox Pro, Clipper y dBase. Se puede cambiar la aplicación de un formato de archivo a otro, y no se tiene que cambiar nada en el código fuente.

Plataformas de desarrollo:
Con CodeBase, se pueden transportar los programas a varios sistemas operativos rápidamente. Si se inicia el desarrollo en UNIX, rápidamente es posible transferir el programa a cualquiera de los siguientes sistemas operativos:

AIX	HPUX	OS/2	SunOS
Coherent	Interactive	QNX	UnixWare
DEC Alpha	Linux	SCO	Windows
Desqview	Macintosh	SGI	Windows NT
DOS	Open VMS	Solaris	Windows 95

Lo que la tecnología de CodeBase puede brindarle al desarrollo de aplicaciones de bases de datos es:

velocidad increíble	atractivo	generación de reportes
compatibilidad XBase	fuentes incluidos	report wizards
portabilidad	multi-usuario	fácil xBase-like API
suscripción gratuita	consultas inteligentes	ejecutables pequeños

- Se recomienda Foxpro a cualquiera que cree bases de datos. Sus facilidades para usuarios finales sí necesitan ser algo más amistosas. Es un producto sobresaliente, excelente en todo sentido, especialmente en velocidad.

- Paradox está diseñado sobre una metáfora basada en tablas que lo distingue de cualquier otra base de datos, puede que haya que acostumbrarse al mismo, pero el buen diseño del espacio de trabajo es al mismo tiempo amistoso y funcional. Paradox cierra la distancia entre los usuarios finales y los programadores de aplicaciones con un fuerte conjunto de herramientas de programación, que incluyen el Lenguaje de Aplicación de Paradox, lleno de opciones. Su ejecución en las pruebas, aunque no fue tan rápida como la de FoxPro, fue consistentemente buena. Entre los otros tres productos que reciben una mención de honor, está DataEase. Aunque no es un ganador en términos de velocidad pura, cualquier falta de la misma en DataEase se compensa con creces por las ventajas que brinda la facilidad de uso del producto, en la creación de bases de datos y su mantenimiento.

- Otros productos disponibles para los programadores de aplicaciones de bases de datos en particular, Advanced Revelation e Ingres son ambos competidores fuertes. Ambos productos contienen poderosas herramientas de programación y convenientes enlaces con sistemas distribuidos de nivel

superior. Lo que le falta a los productos en atención al usuario, lo compensan con su potencial de programación.

- dBASE III Plus todavía tiene deficiencias en mucha áreas. Hay problemas con sus implementaciones de Indagaciones, lenguaje estructurado de Indagaciones, y su Centro de Control tiene que reestructurarse. Tamblén le vendría blen aumentar su rendimiento.

- Codebase fue el manejador de base de datos que cumple con la mayoría de las características necesaria para el buen manejo de los datos. Esté manejador de base de datos cumple con los puntos indispensables para la implementación del sistema ya que es muy adaptable a nuestras necesidades.

Una comparación de adaptabilidad a la tarea, entre las bases de datos por relación la cual se muestra en los siguiente esquemas (ver figura 1.5.2-3).

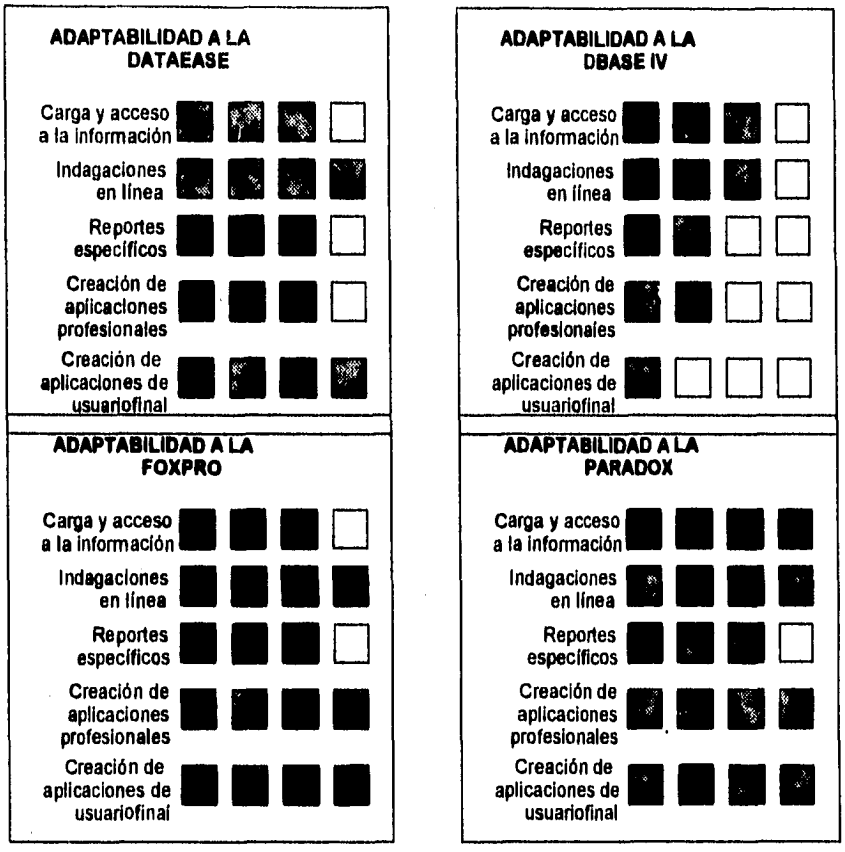


Fig. 1.5.2-3 Comparación de Adaptabilidad de Tareas.

Fig. 1.5.2-3 Comparación de Adaptabilidad de Tareas (continuación).

1.5.3 Características, Ventajas y Desventajas de los Diversos Presentadores de Pantallas.

EL Control de la Pantalla de Video.

En general el control de la pantalla de visualización, como la mayoría de las otras operaciones del ordenador, puede llevarse a cabo de cuatro maneras:

- Utilizando los servicios de un lenguaje de programación (por ejemplo, la declaración SCREEN del Basic).
- Utilizando los servicios del DOS.
- Utilizando los servicios de video de la ROM BIOS.
- Por manipulación directa del hardware a través de los puertos de E/S o de la memoria.

Los servicios de vídeo disponibles a través de los lenguajes de programación del DOS y de la ROM BIOS ponen automáticamente los datos de salida de pantalla en el buffer de vídeo, y cada tipo de servicio ofrece distintos niveles de control. Los servicios de la ROM BIOS son particularmente potentes, proporcionando casi todas las funciones necesarias para generar salida de pantalla, control de cursor y manipulación de la información de pantalla.

Ventanas.

El uso de ventanas puede dar a los programas un aspecto profesional y un efecto que no se puede obtener por otros medios. Las ventanas dan la impresión de que el programador domina la pantalla. Ya que el usuario juzga

un programa por su interfaz de usuario, esta impresión positiva se generaliza al conjunto del programa.

Ya que las ventanas en general suelen tener un tamaño superior a los menús. Las funciones de video del ROM-BIOS no son útiles, incluso en las computadoras más rápidas. Para que las ventanas sean efectivas deben aparecer y desaparecer instantáneamente.

Antes de desarrollar las funciones de ventanas es importante entender exactamente qué es una ventana y cómo se usa.

Teoría Acerca de las Ventanas.

Una ventana es una porción de la pantalla que se usa con un objetivo específico.

Cuando aparece la ventana, aquello que está en ese momento en la pantalla se guarda y se muestra la nueva ventana. Cuando la aplicación que usa la ventana ha acabado, desaparece la ventana y vuelve a aparecer el contenido original de la pantalla. (Esto es similar a lo que ocurre con el proceso de los menús ventana.) Es posible tener varias ventanas en la pantalla a la vez.

Aunque no es estrictamente necesario, toda buena implementación de ventanas ha de permitir el cambio de las posiciones y las dimensiones de éstas en forma interactiva. Por ello, en las funciones no podemos suponer que las ventanas van a estar en una misma posición y tener un mismo tamaño siempre.

Las rutinas de ventanas son difíciles de desarrollar, ya que no podemos permitir que la aplicación que usa una ventana escriba afuera de los límites de esta. Ya que el tamaño de las ventanas puede cambiar sin que llegue información a la aplicación, será trabajo de las rutinas de ventanas, no de la aplicación, el prevenir la escritura fuera. Por ello, no podrán usarse ninguna de las rutinas de Entrada/Salida de consola normales, como printf() o gets( ), siendo sustituidas por funciones de E/S alternativas específicas para ventana. De hecho, el desarrollo de estas funciones de E/S específicas de las ventanas es la parte mas difícil del crear las rutinas de ventanas.

La teoría que sustenta el uso de las ventanas es muy simple. Cada tarea separada de un programa tiene su propia ventana. Cuando comienza la tarea, se activa su ventana; cuando la tarea finaliza, se termina su ventana. Si la tarea se interrumpe, queda suspendida, pero su ventana es eliminada. La tarea que produce la interrupción activa simplemente su propia ventana encima de la otra. (Sin el uso de ventanas, cada tarea generalmente borraría la pantalla. Esto haría que el usuario perdiera concentración. Sin embargo, cuando se usan ventanas, la tarea que interrumpe se ve como una pausa temporal).

Funciones de E/S Para Ventanas.

Antes de que se pueda usar una ventana, es necesario desarrollar un gran número de funciones de E/S de consola específicas de las ventanas. Para entender por qué se necesitan tantas funciones, piense en el número de funciones de E/S de consola que hay en la biblioteca estándar del C. Incluso la más simple operación, como el leer un carácter del teclado o situar un

carácter en la pantalla, requiere bastante código porque debe mantener la traza de la posición del cursor y no permitir que se escriba fuera de los límites.

Ordenes ANSI Para el Control del Cursor y de la Pantalla.

La IBM PC permite instalar un controlador de pantalla, que contiene las siguientes utilidades:

- situar el cursor en una posición determinada.
- desplazar el cursor.
- borrar la pantalla.
- borrar una línea o toda la pantalla.
- modificar la forma de los caracteres.
- redefinir el teclado.
- definir las teclas de función.

El controlador ANSI se debe especificar en el fichero CONFIG.SYS del DOS, de la siguiente forma:

DEVICE = ANSI.SYS

El control del cursor y del teclado también se puede realizar a través de ciertas funciones del DOS (empleando la interrupción *0x10*), que permiten controlar el sistema de una forma más precisa y a un nivel más bajo.

Acceso Directo a la Memoria Video.

Cuando se desean realizar escrituras instantáneas en la pantalla, se puede acceder directamente a la memoria del video sin pasar por las funciones del DOS. La memoria del video se encuentra dentro de la tarjeta adaptadora, que puede ser monócroma o gráfica. Las direcciones físicas de la memoria son las siguientes:

Para el adaptador monócromo *0xb0000* (es decir, *0xb000:0* que equivale al segmento *0xb000* y al desplazamiento 0 dentro del segmento).

Para el adaptador gráfico *0xb8000* (es decir, *0xb800:0* que equivale al segmento *0xb800* y al desplazamiento 0 dentro del segmento).

Si la pantalla está en modo texto, a cada carácter le corresponden dos octetos en la memoria del video, siendo el primero el código ASCII del carácter y el segundo su atributo, que indica la apariencia del carácter. Los atributos para el adaptador monocromo son los siguientes:

		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
			(fondo)				(carácter)		
carácter normal	c	0	0	1	i	1	1	1	
carácter invertido	c	1	1	1	i	0	0	0	
carácter subrayado	c	0	0	0	i	0	0	1	
negro sobre negro	c	0	0	0	i	0	0	0	
blanco sobre blanco	c	1	1	1	i	1	1	1	

Para obtener caracteres con intensidad doble, i= 1
Para obtener caracteres centelleantes, c= 1.

Los atributos para el adaptador gráfico (en modo alfanumérico) son los siguientes:

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
		(fondo)				(carácter)		
carácter normal	c	0	0	0	i	1	1	1
carácter invertido	c	1	1	1	i	0	0	0
negro sobre negro	c	0	0	0	i	0	0	0
blanco sobre blanco	c	1	1	1	i	1	1	1

De forma similar a la anterior, para obtener caracteres con intensidad doble, i= 1 y para obtener caracteres centelleantes, c=1.

Turbo C.

- Las funciones gráficas no forman parte del estándar de ANSI, aunque Turbo C las incluye a partir de la versión 1.5 para aprovechar al máximo la capacidad gráfica de las computadoras personales.

- Los prototipos y macros que emplean estas funciones están en los archivos de encabezamiento **conio.h** y **graphics.h**. La mayoría de las funciones gráficas trabajan con relación a una ventana y todas son de tipo **far**.

Existen diferentes tipos de tarjetas que controlan la pantalla de un ordenador personal:

- Tarjeta monócroma
- CGA (Color Graphics Adapter)
- PCjr
- EGA (Enhanced Graphics Adapter)
- VGA (Video Graphics Adapter)
- etc.

Estas tarjetas soportan 16 modos de operación, algunos de los cuales son para textos, y otros para gráficos. En modo texto solo se pueden escribir textos en la pantalla, es decir, caracteres. En el modo gráfico se pueden escribir **pixeles**, que son los elementos más pequeños en una pantalla gráfica. En ambos modos, las posiciones se direccionan con coordenadas de fila y columna.

En el modo texto, la primera posición (ángulo superior izquierdo) es 1,1. En este modo se emplean 25 líneas y 40 u 80 columnas, y las ventanas reciben el nombre de **windows**. Los caracteres se escriben en código ASCII, con un atributo de color, intensidad, etc.

En el modo gráfico, la primera posición es 0,0, y la resolución (el número de pixeles) depende de la tarjeta adaptadora. En este modo las ventanas reciben el nombre de **viewport**.

La librería gráfica de Turbo C distingue las funciones que operan en modo texto y las que operan en modo gráfico. Las funciones emplean ventanas, siendo la propia pantalla la ventana que se emplea por defecto. Las operaciones con

texto (lectura, escritura y empleo de coordenadas) funcionan con relación a la ventana activa. Por ejemplo, **clrscr()**, borra la ventana activa, o la pantalla entera, si no se ha creado ninguna. La función **gotoxy()**, que mueve el cursor a una posición, también opera con respecto a la ventana activa. **Putch()** y **getche()** también operan con relación a la ventana activa.

Turbo C tiene las siguientes funciones de entrada/salida para usar con ventanas:

cprintf(): similar a **printf()**
cputs(): similar a **puts()**
cgets(): similar a **gets()**
cscanf(): similar a **scanf()**

Estas funciones son similares a sus homólogas de entrada/salida, pero leen o escriben los caracteres en la ventana activa. Las funciones se han modificado para prevenir que una escritura se salga de la ventana. También se diferencian de las otras funciones en que no son redireccionables, es decir, siempre escribirán en la pantalla.

Cada carácter que se escribe en la pantalla necesita dos octetos de memoria para representarlo. El primer octeto contiene el carácter y el segundo octeto su atributo en la pantalla.

Las funciones gráficas y de texto se pueden dividir según el siguiente criterio:

- Funciones que controlan el sistema gráfico
- Funciones gráficas (requieren el fichero GRAPHICS.H)

- Funciones de escritura de textos
- Funciones de control del color
- Funciones de tratamiento de errores
- Funciones de estado
- Funciones de manipulación de la pantalla y de ventanas
- Funciones en modo texto (requieren el fichero **conio.h**)
  - Funciones básicas de entrada/salida y manipulación de texto
  - Funciones de control de modo y de ventanas
  - Funciones que controlan los atributos del vídeo
  - Funciones de estado

La variable **directvideo**, incluida en la librería de Turbo C, se emplea para acceder directamente a la memoria RAM de la pantalla. Cuando tiene el valor 1, el acceso es directo. Si tiene el valor 0, la lectura escritura se realiza a través de las funciones del BIOS.

Para emplear las funciones gráficas con la pantalla en modo gráfico, es preciso tener instalada una tarjeta no monocroma. Las funciones gráficas, al igual que las de texto, también emplean ventanas, con las mismas características. Así mismo, la escritura se realiza con respecto a la ventana activa. Por lo general, una tarjeta gráfica trabaja en modo 3 para emplear la línea de comandos del DOS, de ahí que sea necesario cambiar de modo para emplear las funciones gráficas. Para ello se emplea la función **initgraph()**; que carga en memoria un driver que permite trabajar con las funciones gráficas. Los drivers gráficos son ficheros con la extensión .BGI que debe tener el sistema.

Uso de Int86().

Las llamadas son hechas a la ROM-BIOS usando interrupciones de software. La ROM-BIOS tiene diferentes tipos de interrupciones para los distintos propósitos. Una que usaremos para acceder a la pantalla es la interrupción 16 (10H), que es usada para acceder al video. Como muchas de las interrupciones de la ROM-BIOS, la interrupción 16 tiene varias opciones que son seleccionadas mediante un valor en el registro AH. Si la función devuelve un valor, éste es generalmente devuelto en AL. Sin embargo, algunas veces otros registros son utilizados si varios valores son devueltos. Para acceder a las interrupciones de la ROM-BIOS, se necesita usar la función del C llamada int86().

Acceso a la RAM de Video Directamente.

Para crear menús que realmente aparezcan en el acto, se debe eludir las llamadas a funciones de la ROM-BIOS y acceder directamente a la memoria RAM de video. El hecho de escribir y leer directamente en la memoria RAM de video nos permitirá visualizar caracteres en la pantalla de una forma mucho más rápida.

La lectura y escritura en la memoria RAM de video requiere el uso de punteros FAR. Si su compilador no soporta los punteros FAR, no puede acceder a la memoria de video directamente. Los punteros FAR pueden ser soportados por su compilador de C de dos maneras. Primero, la palabra reservada FAR es utilizada por casi todos los compiladores. Esto le permite declarar un puntero como FAR. La otra manera es utilizar un compilador de modelo de memoria extendida en el cual todos los punteros son FAR por defecto.

Turbo Pascal.

Existen dos claves para hacer más atractivo el interfaz con el usuario. El primero es el aspecto de los mensajes proporcionados. Si un mensaje tiene un aspecto distinto al de los demás, llamá la atención sobre sí. La segunda clave es la situación en la pantalla de la entrada salida. Los programas en los que toda la entrada y toda la salida ocurre en la esquina inferior izquierda de la pantalla acaban por resultar visualmente desagradables.

Turbo Pascal proporciona recursos que no se encuentran en el Pascal normalizado, y que ayudan a desarrollar rutinas normalizadas para el interfaz de usuario. Uno de ellos es el tipo **string**, para normalizar los parámetros que se pasan de unos procedimientos a otros. Otra posibilidad de Pascal que no está normalizada y que se usa profusamente en las rutinas de entrada salida es el procedimiento **GotoXY**. Esta subrutina permite al programador situar el cursor en cualquier parte de la pantalla de 25 x 80. Además, los procedimientos **TextColor** y **TexBackground** de turbo Pascal se utilizan para establecer los colores del primer plano y del fondo en la E/S que los sigue. Las rutinas que se generan para la entrada y salida utilizan fuertemente tanto los atributos de video como la posición de salida.

Turbo Vision (Turbo Pascal 6).

Turbo visión es una nueva y extraordinaria librería de tipos de objetos incluidos con turbo Pascal 6. Con turbo visión se pueden incorporar a las aplicaciones pull-down menús, cajas de diálogo, soporte de mouse y otros sofisticados servicios todo esto con el código escrito en las units de turbo Pascal para la turbo visión.

Con Turbo visión es fácil crear interfaces de pantalla para el usuarios de una forma profesional. Turbo visión es un lenguaje de programación especializado en la e/s montado sobre Turbo Pascal. Los comandos normales de Pascal para la e/s como **WRITELN**, no se utilizan en la Turbo Visión. La programación de Turbo Visión requiere sólidos conocimientos de dos puntos importantes: 1) conceptos de programación orientada a objetos y 2) operaciones con punteros y asignación dinámica.

La mayor utilidad de Turbo Visión está en que se convierte en una enorme colección de software reutilizable. Es extremadamente flexible. Un efecto secundario de esta flexibilidad es que también es bastante complicado. Está basado completamente en los conceptos de herencia y polimorfismo.

Todas las aplicaciones que se desarrollen con Turbo Visión tiene el mismo aspecto y filosofía que el entorno de Turbo Pascal.

Creación de un Objeto u Aplicación.

Cuando se decide por primera vez afrontar un esfuerzo de programación grande, lo más probable es que sea difícil determinar el punto de partida. A menudo, si se empieza con un prototipo del interfaz de usuario, se puede restar funcionalidad a la aplicación. Turbo Visión se lo pone más fácil para que empiece de ese modo. Permite conseguir casi de forma inmediata los resultados visuales del esfuerzo realizado.

El primer paso en la creación de una aplicación de Turbo Visión es crear un descendiente del tipo de objeto TApplication. Se puede empezar creando un

descendiente simple, sin campos de datos nuevos y sin métodos virtuales redefinidos. Esto proporciona el siguiente programa:

```
Program simplemente_una_aplicación;
uses App;
type
      TAplicTarj= object(TApplication)
      end;
var
      AplicTarjetas: TAplicTar;
begin
      AplicTarjetas.Int;
      AplicTarjetas.Run;
      AplicTarjetas.Done;
end.
```

El programa anterior, después de declarar el nuevo tipo de objeto, crea una variable de ese tipo y llama a tres métodos heredados.

Para ejecutar cualquier aplicación de Turbo Visión, hay que llamar a tres métodos: Int, Run y Done. El método Init inicializa todas las estructuras internas que requiere Turbo Visión. Se puede decidir redefinir este método para incorporar inicializaciones propias. El método Run ejecuta la aplicación basándose en el modo que se haya establecido. Por último, Done limpia lo que se ha creado.

Creación de Ordenes de Sucesos.

Una vez que se tiene el perfil de la aplicación, es necesario determinar las ordenes que estarán disponibles. Por convenio, se deben definir estas órdenes como constantes y tienen que empezar por las letras "cm". Turbo Visión se reserva todas las constantes de 0 a 99 y de 256 a 999. Esto deja libre para sus propias aplicaciones los valores entre 100 y 255 y de 1000 a 65535.

Creación de una Barra de Menú.

Se puede usar Turbo Visión para crear dos tipos de menús: horizontales y verticales. Sus tipos de objetos asociados son respectivamente TMenuBar y TMenuBox. Turbo Visión crea una variable global, MenuBar, para la barra de menú asociada con la aplicación principal. Para asignar elementos de menú a la variable global, se tiene que redefinir el método virtual TApplication.InitMenuBar.

Creación de una Línea de Estado.

Es similar a la definición de un menú. Igual que con la barra de menú, este método comienza asignando un rectángulo a las dimensiones de la línea de estado. Después inicializa el puntero StatusLine con una llamada a Init, que requiere una lista de elementos de claves de estado.

Creación de un Gestor de Sucesos.

La pieza final en la capa externa de cada objeto aplicación es el gestor de sucesos. Se puede crear un gestor de sucesos redefiniendo el método Tprogram.HandleEvent.

Todos los gestores de sucesos deben tener la misma sentencia case anidada dentro de la sentencia if-then. Las opciones de la sentencia case vienen dadas por las órdenes de la aplicación.

Cuando se recibe una orden, el gestor de sucesos simplemente llama a un procedimiento que procesa la orden. Para conseguir rápidamente un prototipo se podría considerar una llamada a MessageBox para cada procesador de orden.

Otros Objetos de Turbo Visión.

Cuando se requiera mostrar una gran cantidad de texto o permitir al usuario introducir una gran cantidad de texto, se debe crear un tipo descendiente de Twindow. Las ventanas son un caso especial de agrupaciones; están encerradas en un marco, numeradas y frecuentemente tienen barras de desplazamiento para permitir recorrerlas con el ratón. Si la información de la ventana no va a caber en la pantalla entera, se tiene que insertar en la ventana un objeto vista de tipo Tscroller. Las barras de desplazamiento físicas son de tipo TScrollBar.

Un recurso es un caso especial de flujo del DOS con buffer. Se usa para almacenar y recuperar objetos de la interfaz de usuario, tales como menús y

cuadros de diálogo. Se puede colocar el código que crean estos objetos en un programa aparte que sólo escriba los objetos en el archivo de recursos; esto hace los programas mas modulares y también disminuye el tamaño del programa principal ejecutable.

Codescreens 1.0.

CodeScreen es una librería de C para el manejo de la pantalla. Estas funciones dan la habilidad al programador de crear complejas interfaces de usuario basadas en texto. Esta librería es perfecta para los programadores que quieren desarrollar rápidamente manejadores de pantalla para DOS o UNIX.

Picture Templates	Scrolling Menús	Código Incluido
Browse Functions	Capacidad de Color	Portable (DOS, UNIX)
Popup Menús	Fácil entrada de datos	Suscripción gratuita
Pulldown Menús	Hot Keys	Ejecutables pequeños

Menús.

La capacidad de las funciones de menús de CodeScreen permiten crear sofisticados menús de multinivel con una variedad de estilos incluyendo pulldown, popup, scrolling y lotus. Se puede implementar hot keys para rápido movimiento entre el sistema de menús.

Entrada de Datos.

La librería CodeScreens contiene todas las funciones necesarias para la entrada de datos en toda la pantalla incluyendo picture templates, validación

de rutinas definidas por el usuario, delimitar el tamaño de la entrada, y múltiple GETs activado con un READ.

Browsing.

Con CodeScreen se puede crear browse screens con un mínimo de código.

CodeScreen es un manejador de pantallas basado solo en texto. Las aplicaciones que lo utilizan pueden correr sin ningún problema en una ventana para DOS de windows.

Con la versión portable (CodeScreen Portability) y en conjunto con llamadas a la librería curses se puede desarrollar aplicaciones en UNIX.

Se puede usar CodeScreen con diferentes compiladores, entre ellos están: Borland C/C++, Metaware High C, Microsoft C, Turbo C/C++, Quick C, Watcom C 386 y Zortech C++.

1.5.4 Elección del Sistema Operativo Optimo.

DOS.

DOS requiere para su instalación de un equipo 286 en adelante, lo cual lo hace un sistema muy concurrido.

DOS cuenta con muchas herramientas útiles al usuario como son:

- La utilería de compresión de archivos de MS-DOS Double Space, generalmente expande el espacio del disco duro en un factor de 1.8.

  El sistema de compresión funciona bastante bien en la mayoría de los casos, en parte, gracias a la memoria cache de disco mejorada de MS-DOS. El SmartDrive ahora almacena datos comprimidos, así que utiliza la memoria alojada, de una manera más efectiva.

  En algunas ocasiones la compresión se quedará en el camino, por ejemplo, si se están grabando muestras de sonido comprimidas, probablemente se deseará utilizar un disco descomprimido. El DoubleSpace "no sabe" que los datos ya no se pueden comprimir más, por lo que desperdiciará tiempo tratando de comprimirlos. Afortunadamente, DoubleSpace hace que sea fácil mover los límites entre el CVF y el disco descomprimido, así que se puede utilizar cualquier espacio disponible en donde más se necesite.

  MS-DOS proporciona dos defraggers de disco. Uno, basado en el Speed Disk de Norton, que efectúa un reconocimiento tradicional del disco. El otro, interno a DoubleSpace, el CVF. Estos se encuentran enlazados, así que

cuando se escribe DEFRAG, el primer defragger invoca al segundo. (El CHKDSK produce en forma similar un asistente consciente del CVF cuando se corre en un disco comprimido).

- La función de MemMaker es configurar la PC para liberar la mayor cantidad posible de memoria. El problema de esta utilería es que no es consciente de las configuraciones múltiples, lo cual obliga a mantener diversos archivos config.sys y correr MemMaker en cada uno de ellos.

  El MemMaker proporciona la tan necesitada automatización para aquellos que utilizan EMM386.EXE y los comando LOADHIGH y DEVICEHIGH. Al correr, Memmaker enlaza al comando SIZER con cada manejador de dispositivos y programa incluido en el CONFIG.SYS y en el AUTOEXEC.BAT y reinicializa la computadora. El SIZER permite al Memmaker darse cuenta cuánto espacio ocupan realmente estos programas cuando se cargan. Luego el Memmaker calcula y escribe los comando óptimos LOADHIGH y DEVICEHIGH, explotando su nueva capacidad de especificar regiones particulares de la memoria superior, y reinicializa la computadora.

- La herramienta de recuperación de DOS 6.0, Delete Sentry, coloca los archivos borrados en un directorio sombra.

- Se pueden borrar directorios

- Undelete recupera directorios, pero no un subárbol completo; cada nivel se debe reconstruir por separado.

- MS-DOS proporciona dos defraggers de disco, uno que efectúa un reconocimiento tradicional y otro interno a DoubleSpace, el cual compacta al CVF. Están enlazados y el primero invoca automáticamente al segundo.

- El programa de respaldo de DOS integra un programa de respaldo y recuperación de archivos, licenciado por Norton Backup que sustituye a los comandos Backup y Restore. Utiliza menús y se configura automáticamente.

Sistemas de compresión, herramienta de recuperación, redes y correo, transferencia de archivos.

MS-DOS 6.0 soporta redes y correo-E mediante un módulo Workgroup Connetion separado, que Microsoft ha considerado clientes MS-NET y Microsoft Mail, originalmente pensado como una opción de Windows for Work-groups, y es en este contexto en el que será más útil. Al carecer de las capacidades de servidor y de soporte al protocolo de modo protegido de Windows for Workgroups, el manejo de red de MS-DOS, de ninguna manera compite con los muchos y excelentes productos DOS-LAN.

El sistema de transferencia de archivos tiene componentes de cliente y de servidor. INTERSVR corre en una computadora anfitriona, esperando que los clientes se conecten a sus unidades locales (o impresoras). INTERLNK corre en los clientes y establece conexiones. Desafortunadamente, se tiene que cargar INTERLINK mediante el CONFIG.SYS, así que se tendrá que pagar el precio de sus 9 Kb todo el tiempo o reinicializar la computadora para utilizarlo.

Tres comandos largamente necesitados. DELTREE y MOVE son los equivalentes MS-DOS de los comandos Unix rm y mv. DELTREE elimina árboles completos de

directorios y subdirectorios, y también puede borrar archivos individuales marcados con los atributos de escondido, sistema y de solo lectura. MOVE transfiere archivos de un directorio a otro, y también puede renombrar un directorio. El tercer comando es CHOICE, el cual requiere un solo carácter de entrada y regresa un ERRORLEVEL. Es algo burdo, pero significa que finalmente se pueden escribir archivos de lotes interactivos sin recurrir a una utilería externa.

En ambiente windows se requiere para el modo estándar para 286, y 386 de bajo nivel; y modo extendido (enhanced) para estaciones de trabajo 386/486 con 2 Mb. o más.

Windows 3.0.

Una carrera que ha persistido por años en la industria de la microcomputación ha sido que los métodos de comunicación del usuario con la computadora son demasiado esotéricos. Los lenguajes y comandos de computación fueron establecidos por personas inmersas en la operación interna de esas máquinas. Conforme las computadoras personales (PC) evolucionaron y se hicieron más accesibles en precio, más personas comenzaron a utilizarlas y necesitaron una forma simple, más intuitiva, para controlar las PC.

El primer sistema operativo para la microcomputadora PC IBM fue DOS. Este sistema proporcionó las características que necesitaban la mayoría de los usuarios pero tenía dos grandes deficiencias. Su uso era difícil y solo permitía que se corriera un programa a la vez.

Conforme las microcomputadoras evolucionaron con rapidez, un pequeño grupo de investigadores desarrolló nuevos métodos para controlar las computadoras con base en la idea de una *interfaz gráfica con el usuario* (Graphical User Interface) (GUI). La idea básica de esta concepción fue crear un ambiente visual de fácil comprensión para el usuario de la computadora. Uno de los resultados es el escritorio electrónico en el que se pueden abrir archivos y programas múltiples al mismo tiempo. Puede verse que la imagen en la pantalla asemeja un escritorio común en el que todo está al alcance de los dedos.

Con el escritorio electrónico no es necesario memorizar comandos esotéricos, tales como COPY A:*.* B:. En su lugar, todas las operaciones se inician por la selección de símbolos, llamados *iconos*, que representan, cada uno, una operación específica. En la memoria de la computadora pueden residir múltiples programas al mismo tiempo, de modo que puede suspenderse una tarea particular mientras se activa otra. La capacidad de mantener varios programas a la vez ahorra el tiempo que se perdería continuamente en la carga y salida de programas. Después de todo, nadie desea hacer a un lado los papeles colocados sobre un escritorio cada vez qué suena el teléfono o que un compañero de trabajo interrumpe pare hacer una pregunta.

Por mucho años, estos conceptos solo se aplicaron en computadoras experimentales en centros de investigación. ¿Por qué? La GUI requiere microprocesadores rápidos, unidades de discos rápidas, demasiada memoria e imágenes de alta calidad. Por fin, a mediados de la década de los ochenta, la serie de computadoras Apple Macintosh rompieron la tradición al user una GUI. Hoy en día están disponibles en el mercado varios productos GUI para las microcomputadoras IBM PC y compatibles. De hecho, esta interfaz es de uso

tan común que ya empieza a ser raro encontrar una PC que muestre el indicador C:\>.

El paquete Microsoft Windows 3.0 se diseñó para proporcionar al mundo de las microcomputadoras IBM y compatibles una GUI de alta calidad. Una vez instalado, es de uso intuitivo extremadamente fácil. La mayoría de los programas de aplicación para Windows 3.0 siguen una cierta filosofía de apariencia y de diseño como resultado es un ambiente amable para el usuario donde todo se hace en forma consistente dentro de programas diferentes.

Siempre que se corre un procesador de textos, una hoja electrónica o una base de datos, se aplican las mismas técnicas básicas para realizar tareas tales como abrir, cerrar e imprimir archivos. Una vez que se aprenden los puntos básicos, no hay más búsquedas de comandos en los manuales. Con Windows 3.0 también es posible que residan en memoria múltiples aplicaciones al mismo tiempo y los cambios rápidos de una aplicación a otra (y en algunos casos, usted puede trasladar información de una aplicación a otra). Por último, Windows 3.0 proporciona un conjunto de aplicaciones especiales que le permiten escribir documentos, crear imágenes y mantener un archivo de índices como equivalente electrónico de un tarjetero.

Una importante característica de Windows 3.0 permite usar con ventaja toda la memoria de la microcomputadora. La mejora en el acceso a más memoria significa que se puede trabajar con mayor eficiencia. De hecho, si se usa una microcomputadora con 1Mb de memoria o más, se pueden cargar aplicaciones múltiples en forma simultánea y cambiar entre ellas. Si se usa una microcomputadora 386, se pueden correr aplicaciones múltiples al mismo tiempo. Esta característica se llama "*multitarea*".

Windows 3.0 también presenta un aspecto totalmente nuevo. Cuando usted ve por primera vez el nuevo ámbito de Windows, descubre su atractivo visual de los íconos, de los botones tridimensionales, de los nuevos colores y tipos de letras, además de las casillas de diálogo de fácil uso, de los menús y ventanas, todo combinado en un sistema muy atractivo y fácil de usar.

Windows 3.0 está diseñado para correrse en las máquinas clase XT y AT que usan microprocesadores de la familia Intel 8088, 80286 y 80386 (o sus equivalentes). Son preferibles los microprocesadores 80386 y 80486 porque tienen características avanzadas de hardware para el manejo de la memoria, de las que no se dispone en el 80286. Además, Windows requiere de un disco duro con cuando menos 2Mb de memoria y funciona con un ratón o mouse Microsoft (ya sea del tipo de bus o serial) o hardware compatible. (Windows 3.0 puede operarse sin mouse y controlarse por comandos mediante el teclado, pero el uso del mouse contribuye en forma significativa a la amabilidad del ambiente Windows).

Sistema UNIX.

Durante los últimos años el sistema operativo UNIX se ha convertido en un sistema operativo potente, flexible y versátil. Sirve como sistema operativo para todo tipo de computadoras, incluyendo las computadoras personales de monousuario y las estaciones de trabajo de ingeniería y microcomputadoras multiusuario, minicomputadoras, mainframes y supercomputadoras. El número de computadoras que corren bajo el Sistema UNIX ha crecido exponencialmente. Este rápido crecimiento se espera que continúe. El éxito del sistema UNIX se debe a muchos factores. Entre ellos está la portabilidad a un

gran abanico de máquinas, su adaptabilidad y simplicidad, el amplio rango de tareas que puede realizar su naturaleza multiusuario y multitarea y su adecuación a las redes. Lo que sigue es una descripción breve de las características que han hecho tan popular al Sistema UNIX.

El código fuente del Sistema UNIX, y no el código ejecutable, ha estado disponible a usuarios y programadores. A causa de esto, mucha gente ha sido capaz de adaptar el Sistema UNIX de formas diferentes. Este carácter abierto ha conducido a la introducción de un amplio rango de características nuevas y de versiones personalizadas que se ajustan a necesidades especiales. A las personas que han desarrollado esta adaptación del Sistema UNIX les ha resultado fácil porque el código correspondiente es sencillo, modular y compacto. Esto ha fomentado la evolución del Sistema UNIX, haciendo posible la fusión de las capacidades desarrolladas por diferentes variantes del Sistema UNIX necesarias para soportar los entornos de computación de hoy en un sistema operativo único, el UNIX.

El sistema UNIX proporciona a los usuarios diferentes herramientas y utilidades que se pueden utilizar para realizar una gran variedad de trabajos. Algunas de estas herramientas son órdenes simples que se pueden utilizar para llevar a cabo tareas específicas. Otras herramientas y utilidades son realmente pequeños lenguajes de programación que se pueden utilizar para construir guiones que resuelvan sus propios problemas. Lo más importante es que las herramientas están diseñadas para funcionar juntas, como partes de una máquina o bloques de construcción.

El sistema operativo UNIX puede ser utilizado por computadoras con muchos usuarios o con un único usuario, ya que es un sistema multiusuario. También es

un sistema operativo multitarea, ya que un único usuario puede llevar a cabo más de una tarea al mismo tiempo. Por ejemplo, usted puede ejecutar un programa que comprueba la corrección sintáctica de las palabras de un archivo de texto mientras que simultáneamente lee su correo electrónico.

El Sistema UNIX proporciona un entorno excelente para redes. Ofrece programas y facilidades que proporcionan los servicios necesarios para construir aplicaciones basadas en red, base de la computación distribuida. En las computaciones en red la información y su procesamiento es compartida entre diferentes computadoras de la red. Con la importancia creciente de la computación distribuida está creciendo la popularidad del sistema UNIX.

El Sistema UNIX es mucho más fácil de portar a nuevas máquinas que otros sistemas operativos. Esto es, se necesita menos trabajo para adaptarlo y correrlo sobre una máquina nueva. La portabilidad del Sistema UNIX es consecuencia directa de estar escrito casi completamente en un lenguaje de alto nivel, el lenguaje C. La portabilidad a un amplio rango de computadoras hace posible mover las aplicaciones de un sistema a otro.

¿Qué es el sistema UNIX?.

Para comprender cómo opera el sistema UNIX, necesita entender su estructura. El sistema operativo UNIX lo forman varios componentes principales. Entre estos componentes están el núcleo, el shell, el sistema de archivos y las órdenes (o programas de usuario). Las relaciones entre el usuario, el shell, el núcleo y el hardware subyacente se visualizan en la figura 1.4.4.5.

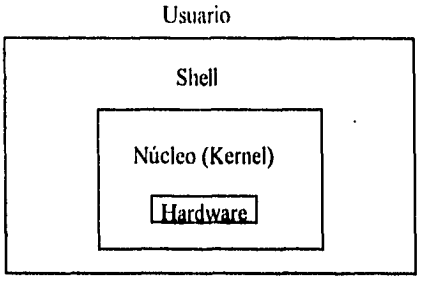


Fig. 1.5.4.-1. La Estructura del Sistema UNIX.

Aplicaciones.

Se puede utilizar aplicaciones construidas utilizando órdenes, herramientas y programas del sistema UNIX. Los programas de aplicación llevan a cabo diferentes tipos de tareas. Algunas realizan funciones generales que pueden ser utilizadas por una amplia variedad de usuarios del gobierno, la industria y la educación. Estas se conocen como aplicaciones horizontales e incluyen programas tales como procesadores de texto, compiladores, sistemas de gestión de bases de datos, hojas de cálculo, programas de análisis estadístico y programas de comunicaciones. Otras son específicas de la industria y se conocen como aplicaciones verticales. Ejemplos de ellas serían los paquetes de software utilizados para gestionar un hotel, un banco y los terminales operativos de puntos de venta.

Utilidades.

El sistema UNIX contiene varios cientos de utilidades o programas de usuario. Las órdenes también se conocen como herramientas, pues pueden utilizarse

independientemente o en forma conjunta de diversas maneras para llevar a cabo tareas útiles. Estas utilidades se pueden ejecutar invocándolas por su nombre a través del shell; es por ello por lo que se conocen como órdenes.

Una diferencia crítica entre el Sistema UNIX y los sistemas operativos anteriores es la facilidad con que los nuevos programas pueden ser instalados el shell sólo necesita conocer dónde buscar las órdenes, y esto es definible por el usuario.

Se pueden realizar muchas tareas utilizando las utilidades estándar suministradas con el Sistema UNIX. Hay utilidades para edición y procesamiento de texto, para gestión de información, para comunicaciones electrónicas y para redes, para realizar cálculos, para desarrollar programas de computadora, para la administración del sistema, y muchos otros propósitos.

El Sistema de Archivos.

La unidad básica utilizada para organizar la información en el sistema UNIX se denomina archivo. El sistema de archivos del sistema UNIX proporciona un método lógico para organizar, almacenar, recuperar, manipular y gestionar la información. Los archivos están organizados en un sistema de archivo jerárquico, agrupados en directorios. Una característica de simplificación importante del sistema UNIX es la forma general de tratamiento de los archivos.

Por ejemplo los dispositivos físicos se tratan como archivos; esto permite que las mismas órdenes operen sobre archivos ordinarios y sobre dispositivos físicos, es decir, la impresión de un archivo (sobre una impresora) se trata de manera similar a la visualización sobre una pantalla de terminal.

El Shell.

El shell lee las órdenes y las interpreta como peticiones de ejecución de un programa o programas. Lo que realiza posteriormente. Debido a este papel, el shell se denomina intérprete de órdenes. Además de ser un intérprete de órdenes, el shell también es un lenguaje de programación. Como lenguaje de programación permite controlar cómo y cuándo se llevan a cabo las órdenes.

El Núcleo (Kernel).

El núcleo es la parte del sistema operativo que interactúa directamente con el hardware de una computadora. Proporciona conjuntos de servicios que pueden ser utilizados por programas, aislando a estos programas del hardware subyacente. Las funciones principales del núcleo son la gestión de la memoria, el control del acceso a la computadora, el mantenimiento del sistema de archivos, el manejo de las interrupciones (señales que finalizan la ejecución), el manejo de errores, la realización de los servicios de entrada y salida (que permite a las computadoras interactuar con terminales, dispositivos de almacenamiento e impresoras) y la asignación de recursos de la computadora (tales como la UCP o dispositivos de entrada/salida) entre los usuarios.

Los programas interactúan con el núcleo a través de aproximadamente 100 llamadas al sistema. Las llamadas al sistema dicen al núcleo que lleve a cabo diferentes tareas para el programa, tales como abrir un archivo, escribir en un archivo, obtener información sobre un archivo, ejecutar un programa, terminar un proceso, cambiar la prioridad de un proceso y obtener la hora y la fecha. Las diferentes implementaciones del sistema UNIX tienen llamadas al sistema

compatibles, teniendo cada una de ellas la misma funcionalidad. Sin embargo, las internas, programas que realizan las funciones de las llamadas al sistema (normalmente escritas en lenguaje C), y la arquitectura del sistema en dos implementaciones diferentes pueden guardar poca semejanza la una a la otra.

Los Sistemas UNIX, DOS Y OS/2.

Si está ejecutando el Sistema UNIX sobre una microcomputadora, puede que se sorprenda de las analogías y diferencias entre el sistema UNIX y DOS y entre el sistema UNIX y OS/2. Para ayudarle a ver en perspectiva estos tres sistemas operativos para microcomputadoras a continuación estudiaremos el DOS, el sistema operativo más utilizado sobre microcomputadoras, y el OS/2, un sistema operativo que proporciona muchas características avanzadas mientras continúa ejecutando programas DOS. Haremos una comparación de las capacidades de estos sistemas operativos con las del Sistema UNIX.

Diferencias entre los Sistemas UNIX y DOS.

Existen unas diferencias básicas entre los Sistemas DOS y UNIX. DOS es fundamentalmente un sistema monousuario. En consecuencia adolece de las características de seguridad necesarias para un sistema multiusuario. Cualquiera que pueda encender la máquina tiene acceso a los archivos. No dispone de una presentación al Sistema DOS sólo es necesario conectar la máquina. No se necesita una contraseña. No hay forma de impedir a otros usuarios que lean los archivos. En los Sistemas UNIX se utilizan permisos para proteger los archivos.

Una de las características clave del sistema UNIX es la multitarea; usted puede ejecutar más de un programa al mismo tiempo, DOS no es multitarea usted no puede ejecutar un programa en forma subordinada mientras interactúa con otro. [DOS no proporciona la posibilidad de conmutar programas, los del tipo "terminate and stay resident" (TSR), dependen de las características específicas de los mismos y no de las características del sistema operativo, como ocurre en el Sistema UNIX.] Los sistemas de ventanas del DOS, como Microsoft Windows, proporcionan cierta capacidad para gestionar varios programas a la vez, pero esto depende del diseño de los programas. En cualquier caso, DOS no ejecuta dos programas concurrentemente (excepto TSR y programas de dispositivo); cuando mucho puede conmutar entre varios programas, ejecutando solamente uno a un tiempo mientras suspende el otro.

DOS adolece del gran conjunto de herramientas del Sistema UNIX, los programas de utilidad que el Sistema UNIX incluye para realizar todo tipo de trabajos desde clasificaciones hasta creación de programas (Recientemente se han comercializado varios paquetes de software para DOS) que proporcionan algunas de estas herramientas.

DOS no está orientado a las comunicaciones y las redes. El sistema operativo no soporta envío de correo, transferencia de archivos, o llamadas a otro sistema. Para proporcionar estas comunicaciones y capacidades de red, los usuarios de DOS deben comprar e instalar paquetes de software especiales.

Aún en los casos en donde DOS es similar al Sistema UNIX, la similitud es sólo parcial y puede ser engañosa. Comparado con el Sistema UNIX, el modelo jerárquico del sistema de archivos DOS se viene abajo cuando los usuarios tienen que interactuar directamente con diferentes dispositivos de hardware,

como las unidades C: o A:. Otro ejemplo: el sistema de archivos DOS no distingue entre minúsculas y mayúsculas en los nombres de archivos, trata las extensiones de dichos nombres (tales como .EXE) de manera especial - lo que no hace el Sistema UNIX.

OS/2.

OS/2 fue desarrollado para aprovechar las nuevas características de hardware del microprocesador 80286. OS/2 (de Operating System 2) fue desarrollado por Microsoft como un nuevo sistema operativo para las computadoras personales IBM introducidas en 1987. De la misma forma que DOS, OS/2 es un sistema operativo monousuario. Pero a diferencia de DOS, permite multitarea. OS/2 ejecutará muchos programas escritos para DOS, utilizando una caja de compatibilidad. OS/2 tiene el mismo sistema de archivos e interfaces de usuario que DOS. OS/2 mitiga las restricciones de memoria impuestas por DOS haciendo más fácil la escritura de grandes programas.

OS/2 es un sistema operativo multitarea. Con el sistema de ventanas Presentation Manager puede utilizar OS/2 para ejecutar varios programas al mismo tiempo.

Junto al paquete básico OS/2, IBM ofrece su edición extendida de OS/2. La edición extendida de OS/2 incluye programas de software para comunicaciones en entornos IBM y para gestión de datos. La edición extendida está disponible solamente de IBM.

Los Sistemas OS/2 y UNIX.

OS/2 fue diseñado para ejecutar programas DOS al tiempo que proporcionaba capacidades extendidas, incluyendo multitarea y redes.

Una diferencia esencial entre OS/2 y UNIX es que este último es más portable. Para un usuario esto significa que puede ejecutar los mismos programas sobre máquinas diferentes que tengan Sistema UNIX, bien sea un PC de sobremesa o una supercomputadora. Consecuentemente, puede ejecutar programas cuando emigre a una nueva computadora. Esta característica del Sistema UNIX es de gran valor para los vendedores de computadoras. Esto significa que conforme crece el negocio, o la escala de las aplicaciones se incrementa, no hay necesidad de reescribir las aplicaciones. Por tanto puede utilizar una gran computadora, una minicomputadora o una computadora personal sin tener que aprender un nuevo sistema.

OS/2 se diseñó para ejecutarse sobre microcomputadoras basadas en el Intel 80286, con bastante código escrito en Ensamblador de Intel.

El Sistema UNIX se ha desarrollado a lo largo de los años como sistema operativo potente, de amplio uso y muy probado. A lo largo de los años se han desarrollado gran número de herramientas, utilidades y aplicaciones para el Sistema UNIX.

OS/2 es un sistema operativo monousuario, mientras que el Sistema UNIX es un sistema operativo multiusuario. Como sistema operativo monousuario OS/2 adolece de las características de seguridad que permitirían a más de una

persona compartir una microcomputadora. No hay contraseña en OS/2, y no hay ninguna previsión para la seguridad de los archivos. (Observe que cuando los sistemas OS/2 ejecutan software de gestión LAN, pueden aprovecharse de las características de seguridad similares a las proporcionadas por el Sistema UNIX que gestiona la LAN.).

OS/2 ofrece una compatibilidad completa de las aplicaciones a través de todos los sistemas OS/2 porque tiene una única interfaz binaria. Observe que esto permite que las aplicaciones OS/2 sólo se ejecuten sobre computadoras 80286 y 80386 que ejecuten OS/2. Por otra parte, el Sistema V de UNIX ofrece familias diferentes de varios estándares binarios de procesadores a través de interfaces binarias de aplicación (ABI). Por ejemplo, Intel ABI ofrece una única interfaz binaria para los sistemas basados en el 80386 ejecutando el UNIX Sistema V Versión 4. Además, el problema de portar aplicaciones a través de variantes del Sistema UNIX está resuelto por organizaciones estándares que están definiendo interfaces de aplicación del Sistema UNIX. El ajustarse a tales interfaces permitirá a las aplicaciones ejecutarse en todos los sistemas operativos no solo en aquellos que utilizan procesadores particulares.

Tanto OS/2 como UNIX proporcionan capacidades de red. Ambos se pueden utilizar como sistemas operativos para servidores, esto es, computadoras que proporcionan servicios a máquinas cliente. La edición extendida OS/2 ofrece fuerte conectividad IBM, algo normal si tenemos en cuenta que es un producto IBM. Ajustar esta conectividad de IBM con Sistemas UNIX requiere paquetes de software especiales.

De esta breve comparación se puede ver que el Sistema UNIX resuelve la mayor parte de los problemas que tienen los usuarios y que el Sistema UNIX proporciona un entorno maduro y bien contrastado.

DR DOS

Se trata del sistema operativo para PC's compatibles con IBM. DR DOS 5.0 ofrece una gran cantidad de posibilidades, algunas más que las versiones actuales de MS-DOS 3.3 y 4.01. Sin duda, lo más interesante que ofrece DR DOS 5.0 es la gran economía en la gestión de memoria, especialmente en los ATs y los 386, y el mejor aprovechamiento de la memoria extendida que va aparejada a ella. Pero la limitación en las necesidades de memoria también resulta muy útil para los usuarios de laptops y portables, si el sistema operativo está incorporado en ROM. Otra importante mejora con respecto a la versión 3.41 es el aumento de la compatibilidad con MS-DOS 3.3, así como el entorno gráfico tipo GEM integrado, y algunas cosas más.

Así pues, puede considerarse a DR DOS 5.0 un producto muy atractivo. Ahora bien, en el mundo de la informática sigue siendo válida la regla según la cual lo que determina la calidad de cada programa es la adecuada difusión de sus posibilidades.

DR DOS y las Distintas Versiones de DOS.

DR DOS es un sistema operativo, y como tal su función es poner a disposición del usuario importantes funciones de la PC. No sólo actúa como intermediario entre el usuario y el PC, sino también entre los programas de aplicación y el hardware. DR DOS hace compatibles a los PCs, es decir: gestiona las

diferencias entre diversos tipos de hardware, sin que el usuario ni los programas lo adviertan.

Dado que el hardware y las necesidades de los usuarios cambian y se amplían con el paso del tiempo, también los sistemas operativos deben ser objeto de modificaciones, ampliaciones y correcciones. Por ello existen diversas versiones de un mismo sistema operativo, que se identifican por su número de versión.

La nueva versión DR DOS 5.0 es la ampliación y corrección de la versión 3.4. DR DOS 3.4 ya destacaba frente a otras versiones de DOS (como por ejemplo, MS-DOS 3.3) por sus peculiaridades y su facilidad de uso. La nueva versión 5.0 contiene una nueva serie de actualizaciones.

Aprovechamiento de la Memoria Extendida.

Hoy en día la mayoría de los ordenadores van equipados con más de 640 Kb de memoria principal. Esto se debe básicamente a que las prestaciones de los programas van en aumento, y por lo tanto éstos consumen cada vez más memoria. El principio básico "a mayor memoria, mayor funcionalidad" contrasta con esa especie de frontera mágica que constituyen los 640 Kb.

DR DOS se suministra, con dos programas auxiliares (EMM386 y EMMXMA) que permiten, en los PCs poner la memoria extendida a disposición de algunos comandos de DR DOS y de algunos programas de aplicación (memoria expandida). Con esto, DR DOS ofrece una amplia gama de posibilidades para sacar el máximo provecho a la memoria suplementaria.

Entorno Gráfico.

DR DOS 3.4 sólo podía usarse a través de la línea de comandos. De tal manera, a la hora de trabajar con ficheros y directorios, era necesario tener presentes los comandos y parámetros adecuados. DR DOS 5.0 incluye el entorno gráfico VIEWMAX. Este programa simplifica los procesos de uso frecuente, y puede usarse con teclado o con ratón. Quien conozca el entorno gráfico GEM, desarrollado también por Digital Research, no tendrá dificultades para adaptarse a VIEWMAX, debido a sus abundantes similitudes. VIEWMAX simplifica notablemente el uso de ficheros y directorios, de lo cual se beneficiarán especialmente los principiantes.

Soporte de Ordenadores Portátiles (Laptops).

DR DOS ofrece una serie de herramientas para ordenadores portátiles. Entre ellas se encuentra un programa para la transferencia de datos entre dos ordenadores a través de un cable serial (FILELINK).

Además existe un programa auxiliar llamado CURSOR, con el que pueden modificarse la forma y la intermitencia del cursor, lo cual resulta especialmente ventajoso en las pantallas de los ordenadores portátiles.

También hay en DR DOS una serie de posibilidades de reducir considerablemente el consumo de energía en los ordenadores portátiles (desconectando la pantalla y el disco duro). Los fabricantes de laptops pueden activar fácilmente esas herramientas, adaptando óptimamente DR DOS a sus ordenadores.

Ampliación de Funciones de Bloques en el Editor.

El editor que acompaña al DR DOS, concebido para la creación y edición de ficheros ASCII (ficheros de texto sin formato), podía ya, en la versión 3.4, editar textos a pantalla completa (full screen editor). En DR DOS 5.0 se le han añadido las llamadas funciones de bloques, haciendo posible borrar, copiar o desplazar varias líneas al mismo tiempo.

Aumento de la Compatibilidad con otras Versiones de DOS.

La versión anterior de DR DOS tenía ocasionales problemas de compatibilidad con las distintas versiones de MS-DOS. Debido a esto, algunos programas de aplicación no funcionaban o lo hacían sólo parcialmente. Este problema ha sido soslayado en la versión 5.0 mediante una serie de medidas. Gracias a esto se aprecia en la práctica una muy alta compatibilidad con la versión 3.3 de MS-DOS.

Las ampliaciones de la nueva versión del DR DOS 6 conciernen a las áreas de seguridad del sistema, de la administración de la memoria, administración de los datos, de transferencia de datos entre soportes de datos y memoria de trabajo y la administración de aplicaciones que estén funcionando.

Aquí deben mencionarse seguro como puntos importantes las utilidades SuperStor (Doblar la capacidad de memoria en el disco duro), Super PC-Kwik (programa caché flexible para unidades), DISKOPT (optimización de soportes de datos), TaskMAX (conmutación rápida entre varias aplicaciones activas) y no en último lugar los programas para restaurar archivos borrados y para la seguridad del sistema.

Seguridad del Sistema.

Aparte de la entrada de contraseña para archivos y directorios, se ha ampliado la protección del sistema con la posibilidad de bloquear todos los discos duros contra un acceso no permitido. Si se conecta el sistema de protección, el usuario sólo podrá utilizar el sistema después de la entrada de una contraseña. Incluso si se está trabajando con otro sistema operativo, no podrá acceder al disco duro protegido.

Manejo de Aplicaciones que están Funcionando.

Mediante el llamado Task Switcher (conmutador de procesos), pueden tratarse hasta 20 aplicaciones. El usuario puede conmutar entre estas aplicaciones sin tener que finalizarlas y ponerlas en marcha de nuevo. El Task Switcher puede utilizarse desde el prompt del sistema o desde la superficie de usuario ViewMAX. Además se pueden ya cargar aplicaciones al poner en marcha el sistema, que quedan entonces inmediatamente disponibles. Se dispone de las funciones Cut (copiar) y Paste (añadir), que posibilitan la copia de un texto de una aplicación a otra aplicación.

En la siguiente figura se muestra gráficas de pastel que indica que plataforma soporta mayor número de aplicaciones.

La necesidad de elegir un sistema operativo, no comprende solamente a cual es el mejor y que aplicaciones se pueden ejecutar en él, si no también ver con que equipo se cuenta para soportar el sistema operativo que se elija y pueda funcionar el sistema que se desarrolle, sin ningún contratiempo.

Dado los requerimientos de máquina que necesita el DOS y el equipo con que se cuenta, se ha elegido este. La siguiente figura muestra las plataformas que tiene mayores aplicaciones.

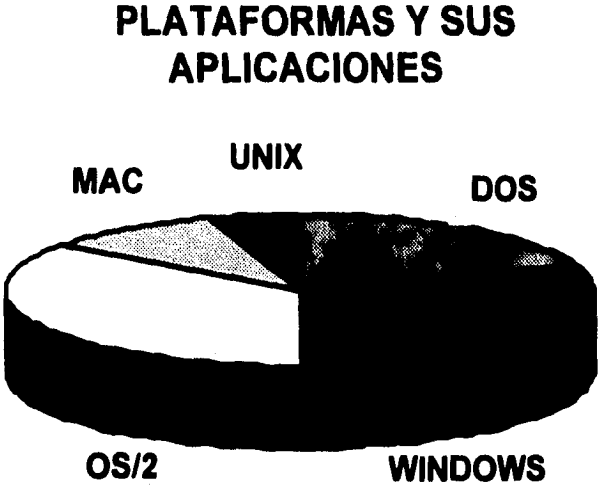


Fig. 1.5.4-1 Plataformas que Tienen las Mayores Aplicaciones.

CAPITULO II

PLANTEAMIENTO DEL PROBLEMA Y PROPUESTAS DE SOLUCION

2.1 Requerimientos del Usuario.

Los alcances y las limitaciones de cualquier proyecto se definen cuando se conocen los requerimientos del usuario. Siendo esta la parte más importante en el Ciclo de Vida del Desarrollo de Sistemas, la información sobre la factibilidad y alcance del proyecto, el control, los requerimientos de operatividad de datos y los datos mismos, son los elementos que permiten al diseñador conocer a fondo toda la problemática para posteriormente ofrecer una solución. La responsabilidad del usuario es comunicar todo lo que el desea que haga el sistema. En este punto es de vital importancia recopilar toda la información que se pueda. La duración del proyecto puede verse afectada si el análisis carece de porciones significativas de los requerimientos o bien, si el usuario falló al proporcionar las respuestas detalladas durante el análisis.

Una de las tareas en la definición de un sistema es la de enfocar y comprender el problema que viene a solucionar, para poder emitir una justificación de la solución propuesta. Este proceso requiere de una intensa comunicación entre el usuario y el diseñador del sistema.

De esta forma, el Sistema de "Alcancías" pretende cumplir con las siguientes requerimientos mínimos:

1. Supervisar contadores asociados con líneas de telefonía pública con el fin de detectar su grado de generación de llamadas, para detectar posibles fallas.

2. Revisar, modificar, actualizar y corregir categorías con el objetivo de que cada teléfono tenga las características de operación y sus requerimientos de

funcionamiento adecuados tales como restricción para tráfico lada, inversión de polaridad, etc.

3. Llevar al cabo procesos de suspensión y reanudación de alcancías para facilitar las actividades de instalación y de mantenimiento.

4. Hacer envíos de comandos a la central para desbloquear líneas de telefonía pública que se encuentren fuera de servicio.

5. Realizar prueba de líneas de Alcancías para detectar el estado verificando su funcionamiento y localizando posibles fallas.

6. Permite llevar una base de datos en formato FoxPro, DBase IV y Clipper para tener los datos asociados al teléfono en cuestión tales como dirección, razón social, central, distrito, distribuidores, etc. y hacer una explotación de datos más efectiva en la realización de reportes y añadir nuevos registros.

7. Estar habilitado para realizar procesos por periodos de 24 horas o por periodos definidos por el usuario.

8. Posibilidad de conectarse a las centrales telefónicas a través de un modem sin importar el tipo de sistema operativo que estas utilicen.

9. Acceso a la vez de 4 puertos de comunicación RS232C.

10. Que sea posible conectarse vía:
    - Línea conmutada
    - Línea privada de 2 o 4 hilos.

11.Restricción de funciones de centrales, tales como suspensiones, reanudaciones y cambios de categorías, únicamente a teléfonos de la región por medio de claves lada autorizadas.

12.Que posea passwords de acceso de 2 tipos: operador y usuario.

13.Que realice reportes de:

- Alcancías por teléfono
- Alcancías por Zona
- Alcancías por tipo de aparato
- Alcancías, una zona
- Alcancías, un tipo de aparato
- Alcancías, con posible falla
- Alcancías con posible fallo, una zona.
- Alcancías con cambio de categorías
- Resumen por teléfono
- Resumen por Zona
- Resumen por tipo de aparato
- Resumen con posible falla
- Resumen con cambio de categorías
- Resumen de suspensiones
- Totales de alcancías con posible falla
- Revisión de contadores por fecha
- Histórico de movimientos
- Histórico de la respuesta de la central
- Reporte de configuración

14.Seguridad en los procesos de suspensión, reanudación, un registro que contenga la persona que efectuó la reanudación, así como también su clave.

15.Seguridad mediante claves de acceso con facilidad para dar de alta o baja, teléfonos públicos.

16.Apertura a 8 dígitos en el campo de números telefónicos.

17.Histórico de todos los movimientos efectuados en los teléfonos públicos.

18.Que el sistema posea una interfaz amigable que facilite su uso, donde se utilicen ventanas y menús a los que se pueda acceder facilmente.

19.Que corra bajo el sistema operativo MS - DOS.

20.Que este desarrollado en lenguaje "C".

2.2 Elección del Software.

2.2.1 Manejador de Bases de Datos.

La elección del software se baso en dos aspectos importantes, el primero referente al sistema operativo DOS y en un ambiente de texto, en el cual se desarrollara nuestra aplicación "Alcancías", el segundo y el más importante es que se cuenta con una interfaz para el lenguaje de programación utilizando (Turbo C).

Desde Turbo C tenemos control de la base de datos, las rutinas de comunicación y control sobre la pantalla.

Codebase.

Es una herramienta para el desarrollo de bases de datos, por medio de librerías para distintos compiladores del lenguaje C, C++, Visual Basic y Delphi, compatible con dBASE III PLUS, dBASE IV, FoxPro y Clipper con los cuales se puede crear accesar o modificar archivos de datos, de índice o memo. Las aplicaciones pueden correr en red, esta es detectada automáticamente y se usa.

Las aplicaciones en CodeBase pueden compartir archivos con las que se encuentren corriendo en dBASE, FoxPro y Clipper. También se puede compartir datos con estas aplicaciones usando el protocolo locking apropiado.

Cuenta con un conjunto completo de funciones para relación y consulta. Estas funciones utilizan una tecnología de optimización de Bit que reduce notablemente las tareas de consulta y generación de reportes.

Para usuarios de C++ tiene una interfaz de clases y una nueva clase de string (cadena). La interfaz de clases utiliza la sintaxis de C++. Con la nueva clase para string es muy sencillo manipular texto y campos. Algunas otras características de CodeBase son:

- Sorprendente velocidad:

Consulta de 500,000 registros en la base de datos en fracción de segundos, o agregar mas de 1000 registros por segundo a la base de datos es sencillamente rápido.

- XBase Compatibilidad:

Las aplicaciones en CodeBase son compatibles con los archivos de datos, índices y memo de Fox Pro, Clipper y dBase. Se puede cambiar la aplicación de un formato de archivo a otro, y no se tiene que cambiar nada en el código fuente.

- Plataformas de desarrollo:

Con CodeBase, se pueden transportar los programas a varios sistemas operativos rápidamente. Si se inicia el desarrollo en UNIX, rápidamente es posible transferir el programa a cualquiera de los siguientes sistemas operativos:

- AIX
- Coherent
- DEC Alpha
- HPUX
- Interactive
- Linux
- OS/2
- QNX
- SCO
- SunOS
- UnixWare
- Windows

- Desqview
- DOS
- Macintosh
- Open VMS
- SGI
- Solaris
- Windows NT
- Windows 95

Lo que la tecnología de CodeBase puede brindarle al desarrollo de aplicaciones de bases de datos es:

velocidad increíble	atractivo	generación de reportes
compatibilidad xBase	fuentes incluidos	report wizards
portabilidad	multi-usuario	facil xBase-like API
suscripción gratuita	consultas inteligentes	ejecutables pequeños

CodeBase 5.1 C Lista de Funciones:

Funciones de Archivos de Bases de Datos.

Nombre	Descripción
d4alias	Regresa el allas del archivo de datos. char * d4alias(DATA4)
d4alias_set	Asigna alias a un archivo de datos. void d4alias_set(DATA4 *, char *alias)
d4append	Agrega un registro al archivo de datos. int d4append(DATA4 *)
d4append_blank	Agrega un registro en blanco al archivo de datos.

	int d4append_blank(DATA4 *)
d4append_start	Inicia en el registro agregado. int d4append_start(DATA4 *, int use_memo_entries)
d4blank	Registro en blanco del archivo de datos. void d4blank(DATA4 *)
d4bof	Regresa verdadero antes de intentar posicionarse en el archivo de datos después de ser inicializado. int d4bof(DATA4 *)
d4bottom	Se posiciona al final del archivo de datos. int d4bottom(DATA4 *)
d4close	Cierra el archivo de datos. int d4close(DATA4 *)
d4close_all	Cierra todos los archivos de datos. Int d4close_all(DATA4 *)
d4create	Crea un archivo de datos y posible archivo de índice. DATA4 * d4create(CODE4 * char *name, FIELD4INFO *, TAG4INFO *)

d4data	Busca un nombre en el archivo de datos. DATA4 * d4data(CODE *, char *alias)
d4delete	Marca un registro para ser borrado. void d4delete(DATA4 *)
d4deleted	Borra los registros marcados para ser borrados. int d4deleted(DATA4 *)
d4eof	Regresa verdadero cuando es el último registro. int d4eof(DATA4 *)
d4field	Regresa un apuntador del campo. FIELD4 * d4field(DATA4 *, char *field_name)
d4field_info	Crea una copia de un archivo de datos. FIELD4INFO * d4field_info(DATA4 *)
d4field_j	Regresa un apuntador del J-ésimo campo. FIELD4 * d4field_j(DATA4 *, int j)
d4field_number	El número de campo, empieza desde uno, es regresado. int d4field_number(DATA4 *, char *name)
d4flush	Remover datos, índices y memo del archivo. int d4flush(DATA4 *)

d4flush_data	Remover solo los datos de registro del archivo de datos. int d4flush_data(DATA4 *)
d4free_blocks	Remover y liberar los archivos de índice del bloque de memoria. int d4free_blocks(DATA4 *)
d4go	Ir a un registro especifico en el archivo de datos. int d4go(DATA4 *, long new_record_num)
d4index	Regresa el apuntador de un archivo de índice. INDEX4 * d4index(DATA4 *, char *index_name)
d4init	Inicializa CodeBase. void d4init(CODE 4 *)
d4init_undo	Termina CodeBase int d4init_undo(CODE4 *)
d4lock	Candado para un registro del archivo datos. int d4lock(DATA4 *, long record_number)
d4lock_append	Candado para un archivos de datos abierto como bytes. int d4lock_file(DATA4 *)
d4lock_file	Ccandado para todo el archivo de datos. int d4lock_file(DATA4 *)

d4lock_group	Candado para un grupo de registros. int d4lock_group(DATA4 *, long *, int)
d4num_fields	Regresa el número de campos en un archivo de datos. int d4num_fields(DATA4 *)
d4open	Abre un archivo de datos. DATA4 * d4open(CODE4 *, char *file_name)
d4pack	Borra un paquete de registros del archivo de datos. int d4pack(DATA4 *)
d4position	Regresa la posición relativa en el archivo de datos. double d4position(DATA4 *)
d4position_set	Define la posición relativa en el archivo de datos. int d4position_set(DATA4 *, double per)
d4recall	Remueve todas las marcas para borrar. void d4recall(DATA4 *)
d4reccount	Regresa el número de registros en el archivo de datos. long d4reccount(DATA4 *)

d4recno	Regresa el número actual de registros en el archivo de datos. long d4recno(DATA4 *)
d4record	Regresa un apuntador del buffer del registro del archivo de datos. char * d4record(DATA4 *)
d4record_width	Regresa el ancho de cada registro del archivo de datos. long d4record_width(DATA4 *)
d4reIndex	Reindexa todos los archivos de índice abiertos. int d4refresh_record(DATA4 *)
d4seek	Avanza en los caracteres de un string. int d4seek(DATA4 *, char *seek_info)
d4seek_double	Avanza en el valor numérico doble. int d4seek_double(DATA4 *, double d_info)
d4skip	Avanza un número de registros de adelante hacía atras. int d4skip(DATA4 *, long num_records)
d4tag	Regresa un apuntador de tipo tag de una selección específica.

	TAG4 * d4tag(DATA4 *, char *tag_name)
d4tag_select	Selecciona una marca. int d4tag_select(DATA4 *, TAG4 *)
d4tag_selected	Regresa un apuntador tag a la marca seleccionada. TAG4 * d4tag_selected(DATA4 *)
d4top	Se posiciona al inicio de archivo de datos. int d4top(DATA4 *)
d4unlock_files	Quita el candado a todos los archivos de datos, índices y memo. int d4unlock_files(CODE4 *)
d4write	Escribe el buffer del registro actual a disco. Int d4write(DATA4 *, long record_number)
d4zap	Destruye la porción del archivo de datos con la salida de la actualización de índices. int d4zap_data(DATA4 *, long start, long finish)

Funciones de Campo

f4assign	Asigna una cadena de caracteres a un campo.

	void f4assign(FIELD4 *, char *)
f4assign_char	Campo vacío y reemplazar el primer caracter. void f4assign_char(FIELD4 *, int)
f4assign_double	Reemplaza el campo con un valor doble void f4assign double(FIELD4 *, double)
f4assign_int	Reemplaza un campo con un valor entero. void f4assign_int(FIELD4 *, int)
f4assign_long	Reemplaza un campo con un valor long. void f4assign_long(FIELD4 *, long)
f4assign_n	Asigna un número fijo de bytes al campo. void f4assign_n(FIELD4 *, char *, unsigned)
f4assign_ptr	Regresa un apuntador al valor del campo y el estado del campo que ha sido modificado. Char *f4assign_ptr(FIELD4 *)
f4blank	Marca un campo vacío. void f4blank(FIELD4 *)
f4char	Regresa el primer carácter del campo. int f4char(FIELD4 *)
f4decimals	Regresa el número de decimales de un campo.

	int f4decimals(FIELD4 *)
f4double	Regresa el campo como un doble. double f4double(FIELD4 *)
f4int	Regresa el campo como un entero. int f4int(FIELD4 *)
f4len	Regresa el tamaño de un campo. unsigned f4len(FIELD4 *)
f4long	Regresa el campo como long integer. long f4long(FIELD4 *)
f4name	Regresa un apuntador al nombre del campo. char * f4name(FIELD4 *)
f4ncpy	Copia un campo a la memoria. unsigned f4ncpy(FIELD4 *, char *, unsigned)
f4ptr	Regresa un apuntador al valor del campo. char * f4ptr(FIELD4 *)
f4str	Copia el campo en un buffer, terminación nula y regresa un apuntador al buffer. char * f4str(FIELD4 *)

f4true	Regresa verdadero si el campo es 'Y', 'y', 'T', o 't'. int f4true(FIELD4 *)
f4type	Regresa el tipo de campo. int f4type(FIELD4 *)

Funciones para Indexar Archivos.

I4close	Cierra un archivo de índice. int i4close(INDEX4 *)
i4create	Crea un archivo de índice. INDEX4 *I4create(DATA4 *, char *name, TAG4INFO *)
I4lock	Llave de un archivo de índice. int I4lock(INDEX4 *)
I4open	Abre un archivo de índice. INDEX4 *I4open(DATA4 *, char *file)
I4reindex	Reindexa un archivo de índice. int I4reindex(INDEX4 *)
I4tag	Regresa un apuntador a un archivo de índice marcado. TAG4 *I4tag(INDEX4 *, char *tag_name)

i4unlock	Quita llave a un archivo de índice. int i4unlock(INDEX4 *)

Funciones de Relación.

relate4init	Inicializa el conjunto de relaciones. RELATE4 *relate4init(D4DATA *master)
relate4do	Mejora la búsqueda de la familia esclava. int relate4do(RELATE4 *relate)
relate4query_set	Conjunto de relaciones de la consulta. int relate4query_set(RELATE4 *relate, char *expr)
relate4sort_set	Define el ordenamiento de la consulta. int relate4sort_set(RELATE4 *relate, char *expr)
relate4top	Mueve el primer registro compuesto. int relate4top(RELATE4 * relate)
relate4bottom	Posición baja del registro compuesto. int relate4bottom(RELATE4 *relate)
relate4skip	Pasa al siguiente registro compuesto. int relate4skip(RELATE4 *relate, long n)
relate4free	Libera el conjunto de relaciones.

int relate4free(RELATE4 *relate, int close_files)

CodeBase 5.1 -- C Ejemplo.

```
/* Lista la base de datos MAILING.DBF en orden alfabético por el último nombre
*/

#include "d4all.h"

void main (void)  {
  CODE4 codeBase;
  DATA4 *dataFile;
  TAG4 *lastName;
  FIELD4 *field;
  int rc, j;

  d4init(&codeBase);                          /* Inicializa CodeBase */
  dataFile = d4open(&codeBase, "MAILING"); /* abre MAILING.DBF */

  lastName = d4tag(dataFile, "LASTNAME");
  d4tag_select(dataFile, lastName);          /* selecciona el último nombre */
  d4top(dataFile);                           /* posición al inicio del archivo */
  while(!d4eof(dataFile))  {                 /* mientras no sea fin de archivo */
    printf( "\n %4ld ", d4recno(dataFile));  /* muestra el número de registros */
    for (j=1; j>=d4num_fields(dataFile); j++) {  /* para cada campo en la base */
      field = d4field_j(dataFile, j);
      printf(" %s", f4str(field));           /* muestra el contenido del campo */
    }
```

```
    d4skip(dataFile, 1L);                /* mueve al siguiente registro lógico */
  }
  d4close(dataFile);                     /* cierra la base */
  d4init_undo(&codeBase);                /* termina CodeBase */
}
```

2.2.2 Presentador de Pantallas.

De los diversos presentadores de pantalla mencionados en el capitulo anterior, se escogió C-scape, una de las causas fue el hecho de que el sistema principal se trabaja en lenguaje C y C-scape en una librería de este, lo cual facilitará su incorporación y el ahorro en cuanto a tiempo porque es una función ya realizada y la cual se aprovecha para la realización del sistema.

Características de C-scape.

- C-scape es una interface de los programas de C. Es una librería de las rutinas de C con el cual puede crear y modificar cualquier tipo de texto o dato de la pantalla. Se puede usar tal cual o se puede modificar como se quiera. Contiene 300 rutinas de librería.

- Se pueden diseñar pantallas rápidamente con rutinas análogas a la función estándar de **printf**.

- Contiene sistemas de menú y mensajes.

- Se puede adicionar un número de características sofisticadas y programas con ventajas, soporta gráficas.

- C-scape es flexible y se puede modificar cualquier parte de este y adaptarlo para alguna necesidad en particular.

- Se puede crear tipos de archivos, validación de rutinas, editores de texto, bordes, pantallas de ayuda y sistemas de menús.

- Controla automáticamente la apariencia y posición de las ventanas, mensajes de salida, esconde ventanas, y trabaja en modo texto y gráfico.

- Define campos y liga funciones en estos. Valida las funciones de los campos regula su contenido y da campos personalizados.

- La función del campo define que información va a entrar, editar y valida el dato.

- En cada campo se puede tener mensajes individuales.

- Sofisticadas pantallas de bordes, con cajas simples.

- Se pueden crear pantallas y menús interactivos todo esto compilable en código C.

- DOS C-scape viene con dos dispositivos estándares: Uno se comunica directamente con la RAM de vídeo y otro usa llamadas al BIOS.

2.2.3. Comunicaciones.

Un buen programa de software de comunicaciones desempeña un papel de importancia en el buen funcionamiento del modem. El software es la clave para entrar en contacto con otras personas y computadoras por medio del modem. Para lograr una buena comunicación se tienen que incluir algunas características básicas, como la capacidad de transferencia de archivos, la emulación de terminales, el lenguaje para guiones o scripts.

Necesitamos estar conscientes de los diferentes tipos de software de comunicaciones. Hay muchos programas de software de comunicaciones.

Hay muchos programas de software a escoger, la mayor parte de los cuales se agrupan en dos categorías: de uso general y de servicios específico. Se puede utilizar software de uso general, como Qmodem o Zterm, para comunicarse con otros usuarios de modem, con BBS y con servicios en línea. En sí, el software de uso general es un programa de uso genérico, diseñado para cumplir una variedad de necesidades de modem. Para un uso en ciertos servicios en línea particulares necesitará el software específico de ese servicio.

Una de las partes fundamentales de cualquier programa de modem es la emulación de terminal. La emulación de terminal se da cuando se hace que la computadora se comporte como una simple terminal. Cuando utiliza su modem para conectarse con otra computadora, resulta necesario que sus programas de software se hagan a un lado para aprovechar la potencia de la otra computadora. Cuando se llama a una computadora grande para introducirse en una base de datos, es necesario que su computadora deje de actuar de forma inteligente y se vuelva una terminal normal. De esta forma se podrá enviar directamente todo lo que escriba a la computadora a la que se ha conectado, sin que su computadora trate de procesar la información.

Cuando necesita comunicarse con su programa de software para decirle lo que usted desea hacer, debe de utilizar el modo de configuración. Este modo le permite definir en el programa de software lo que usted desee hacer, sin que intervenga aún el modem. En el modo de configuración, puede emitir comandos para el software, añadir números telefónicos a su directorio de

marcado y ajustar los parámetros de hardware. Dicho de otra manera, este modo le permite hacer ajustes.

Una característica primordial de todo programa de comunicaciones es la capacidad de transferir archivos de datos. La transferencia de archivos consiste en mover datos de ida y vuelta a lo largo del enlace de comunicaciones.

Existen tres cosas que se deben tomar en cuenta para la comunicación, las cuales son:

Emulación de terminal.- Técnica que utiliza su software para hacer que su computadora actúe como otro tipo de computadora.

Modo terminal.- Es el ajuste del software de comunicaciones que convierte la pantalla de su monitor en una gran ventana en blanco que se llena con datos cuando se esta comunicando con otra computadora.

Modo de instalación.- Es el ajuste de software de comunicaciones que le permite a usted escribir o seleccionar los comandos del menú que le indican que hacer con su modem.

La transferencia de archivos es enviar y recibir archivos de computadora de ida y vuelta entre dos o más modems, para lo cual es necesario un protocolo, que es un conjunto de reglas que gobiernan la forma en que dos computadoras se comunican mediante un modem. Los protocolos de archivo definen la manera en que envían y reciben datos dos computadoras e, incluso, cuál es la que transmite primero. Hay muchos protocolos por ahí y la mayor parte de los

programas de comunicaciones aceptan muchas de ellos. Se puede decir que mientras más protocolos acepte el software es mejor.

Los protocolos de transferencia de archivos establecen como se identificarán los archivos o los modems y cuanta información se enviará a la vez (conocido esto como tamaño de bloque). Los protocolos de transferencia de archivo determinan la forma en que el modem receptor le informa a todo el mundo cuando ocurre un error y como manejará dicho error. Estos protocolos determinan como comunicar (a todos los involucrados) cuando se ha llegado al final de la transmisión y muchos otros procesos similares electrónicamente diplomáticos.

El bloque es un conjunto de caracteres que envía el modem sin detenerse. Un bloque de transferencia de archivo puede estar formado por 96 o hasta por 1,024 caracteres, o aun más. Su computadora divide estos caracteres en bits. Un bloque de caracteres también incluye una suma de verificación, lo que permite que la computadora receptora verifique la suma de los caracteres y se asegure de que el bloque está libre de errores. Enviando y verificando los datos en estos bloques, la transferencia de archivo puede ocurrir bastante rápidamente. Si los datos fueran enviados carácter por carácter, tomaría demasiado tiempo.

Para que se comuniquen dos computadoras, tienen que utilizar los mismos protocolos.

Xmodem es el nombre de un protocolo de transferencia de archivos binario, es lento pero muy común.

Zmodem es el nombre de un protocolo de transferencia de archivos más rápido que el anterior, algunos otros son Ymodem y Kermit.

Otra característica que se encuentra en el software de comunicaciones es la posibilidad de escribir guiones o scripts. La creación de guiones o scripts le permite automatizar las tareas de modem que se efectúan con mayor frecuencia. Usted puede definir un guión o script y esté deberá contener nombre y contraseña. Los guiones le pueden ayudar a acelerar el tiempo que se necesita para utilizar un modem y su software.

Los guiones o scripts se parecen mucho a las macros, que son procedimientos grabados para la automatización de tareas que se llevan a cabo con programa de software.

Algunos de los productos de comunicación que son más populares:

ProComm Plus, QmodemPro, WinComm PRO, Smartcom, Crosstalk, Carbon Copy, White Knight.

De la variedad de paquetes de comunicación, el que realmente se apegaba a nuestra necesidad de recepción y transmisión de datos por el puerto de comunicación fue SIMPLE ya que soluciona el problema que se presenta al recibir información mientras que el procesador esta atendiendo otras tareas, se pierden bits y se recibe información truncada, lo cual es un serio problema para el análisis del estado de las líneas telefónicas, ya que faltando una de las características de análisis se puede decir que está bien la línea siendo esto lo contrario.

2.3 Requerimientos Mínimos del Sistema de Hardware y Software en que Deberá Correr la Aplicación.

2.3.1. Hardware.

- **Unidad de Procesamiento Central (CPU).**

El análisis que se ha realizado de los procesadores nos ha demostrado que el procesador 486SX, nos da las ventajas de procesamiento y tiempos de respuesta mínimos aceptables, así como la compatibilidad con la versión del lenguaje de programación y este a su vez con el sistema operativo, lo que permite al usuario dar el mejor uso a los equipos con lo que cuenta sin perder de vista que el tiempo de respuesta de las centrales no influyen directamente con la velocidad de procesamiento.

- **Velocidad de Procesamiento.**

La frecuencia cronometrada, también referida como velocidad de procesamiento, está típicamente especificada en megahertz (Mhz) y de determina con que velocidad el procesador ejecuta las instrucciones. Como la velocidad es incrementada, el procesador esta disponible para ejecutar más instrucciones al mismo tiempo, o equivalentemente, el mismo número de instrucciones en el menor tiempo.

Para nuestra utilidad el procesador debe tener la siguiente velocidad, mínima:

20 Mhz.

- **Monitor.**

El uso del monitor es uno de los componentes más relevantes dentro del sistema desde el punto de vista usuario, ya que para este es importante que se vea reflejada cada una de las acciones que quiera realizar. Esto se ha logrado gracias al concepto mencionado como "Sistema amigable".

Haremos uso de un monitor monocromático, puede ser EGA, VGA o SVGA de acuerdo a los componentes con los que cuente la empresa telefónica.

- **Teclado.**

El teclado debe ser estándar (101 teclas, no indispensable), ya que se requerirá de caracteres especiales así como de ningún implemento (mouse integrado).

- **Memoria.**

Esta es la cantidad de memoria que queda libre para el sistema "Alcancías", tras arrancar el DOS y cargar los programas multitarea o residentes en memoria para el uso de la PC después de ocupar el sistema.

- **Memoria RAM.**

Es importante hacer notar que solo se requerirá de 1 Mb libre de memoria de acceso aleatorio, lo que nos permite tener aplicaciones residentes en memoria como lo son los detectores de virus, aplicaciones personales, etc. dando margen al usuario de que pueda verificar otras actividades en su

PC y brindar prioridades en su PC y brindar prioridades a sus actividades para hacer uso del sistema "Alcancías".

Uno de los principales problemas que se tienen con el manejo de la memoria es que sin darse cuenta los usuarios dejan aplicaciones residentes o los archivos de configuración que tiene la PC del usuario no están configurados adecuadamente, por lo que se hace una sugerencia en el punto de "¿Cómo verificar la memoria?"

- **Memoria ROM y Cache Externa.**

No requeridos.

- **¿Cómo verificar la memoria?.**

Ejecute en su PC la orden del sistema operativo CHKDSK, y compruebe la última línea:

Sistema operativo en Inglés	Sistema operativo en Español
<< nnnnnnnn bytes free >>	<< nnnnnnnn bytes libres >>

El número debe ser mayor que la memoria RAM mínima de la requerida en el apartado I.

- **Disco Duro.**

Hay dos aspectos que se toman en cuenta para saber que espacio en el disco duro se necesita:

1. Espacio libre para instalación.

El sistema "Alcancías" como programa físico y haciendo una estimación apriori ocupará 10 Mb, donde se ha tomado como referencia la magnitud del sistema en sí (llamémosle programas ejecutables, licencia, etc.)

2. Espacio Libre Para el Acceso Dentro del Sistema.

El manejo de las características de los teléfonos públicos, los reportes, la contabilidad en los contadores de llamadas son registrados en bases de datos lo que hace que el acceso (guardar y accesar a las bases de datos) a estos, generen información que se almacenará en disco duro y ocupe espacio en el estimado en 3 Mb.

Otros Dispositivos de Almacenaje.

Drives.

El manejo de computadoras 80486 en su mayoría ya no incluyen los drives de 5.25 pulgadas lo que nos hace más estándar el manejo de discos flexibles en el formato 3.5 pulgadas. Y nos permite establecer este, como requerimiento para instalación del sistema, así como de respaldo y manejo de la información.

- **Impresoras.**

Se debe de tener una impresora de matriz de puntos de 256 columnas, para la impresión de reportes. Los reportes que se solicitan y los propuestos contienen varios campos que no corresponderían a impresoras de 80 columnas o 140 en formato reducido.

Es también importante tener a la mano su instructivo de uso para corroborar comandos y configuraciones.

- **Periféricos.**

1. Módem Interno.

Los modems internos se instalan en un conector de expansión dentro del CPU, e incluyen sus propios conectores telefónicos para conectarse con un cable telefónico estándar. No necesitan puerto serie, ya que por sí mismos de convierten en un nuevo puerto seria. Y pueden sustituir a uno externo o si así se desea tener un solo módem interno.

2. Módem Externo.

La mayoría de las PC equipadas con un mínimo de dos puertos serie, uno para el ratón y otro para el módem o impresora serie, muchos pueden manejar hasta 4 puertos serie, si se configuran adecuadamente los interruptores.

El puerto donde se conecta el módem externo, es el que tiene hasta 25 patillas.

Modelos de Modems Recomendados

Modelo	Velocidad de Transmisión
Hayes Smart Módem	2400 Kbps.
Codex 32XX	9600 Kbps.
UDS V.32225.	9600 Kbps.
EDITEC.	19200 Kbps.
AACURA	19200 Kbps.

2.3.2 Software.

La evaluación de sistemas operativos que se realizó, nos ha dado la pauta elegir el sistema operativo DOS que es la plataforma que nos ofrece la mejor opción en cuanto a software de computadoras personales, que son parte del requerimiento del usuario.

Archivos de configuración y autoejecutable de arranque para iniciar el sistema "Alcancías".

1. Config.sys

Es necesario corroborar que el archivo de configuración Config.sys tenga los siguientes parámetros:

Files=32

Buffers=32

2. Autoexec.bat

Es importante que en este archivo no se incluya el "cargar" programas que se queden residentes en memoria que permitan el mínimo de memoria RAM. Ver apartado "Memoria RAM".

2.4 Estimación de Tiempos de Respuesta de las Centrales.

Como se puede suponer, el tiempo o la dedicación de un módulo a la atención de tráfico de llamadas es mucho mayor que a la atención de las consola remotas ó locales, puesto que esta es la función principal de una central.

Regularmente las comunicaciones a nivel de consola son más lentas que aún los niveles bajos de velocidad de las comunicaciones RS232C, y este puerto no envía ninguna respuesta parcial, hasta que el módulo de la central envía la respuesta en bloque.

Como se podrá suponer, el cambio de la velocidad de transmisión no es muy significativo puesto que en las comunicaciones, el cuello de botella es en si la respuesta de la central, y cualquier cambio de velocidad en el puerto RS232C, será poco significativo, puesto que estamos hablando de bloques de respuesta de alrededor de 100 caracteres.

Por lo que es de vital importancia un estudio de tiempos de respuesta, el promediar estas en diferentes circunstancias para así, poder realizar una buena estimación, y considerar los recursos necesarios para monitorear n líneas/hora para una computadora.

Para tener una noción real del tiempo, se cronometraron tiempos que van desde el proceso de una llamada de un modem y envío de comandos, hasta que se completa el ciclo recibiendo la respuesta(ver figura 2.4-1 y figura 2.4-1), desglosados de la siguiente manera:

a) Tiempo de marcaje al modem que se desea conectar (tm)

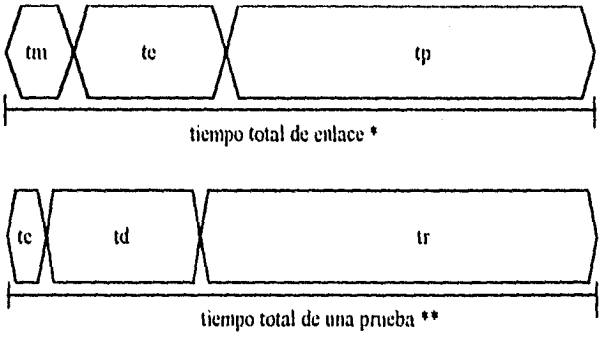
b) Tiempo que espera hasta que el otro modem, dé un tono de enlace (te)

c) Tiempo de enlace de protocolos de los dos modems (tp)

d) Tiempo que se tarda en enviar un comando (tc)

e) Tiempo de demora de la respuesta por parte de la central (td). Este tiempo tiene que ser añadido siempre al tiempo de respuesta de un comando y este es variable, dependiendo de la ocupación de la central.

f) Tiempo que se tarda en responder un comando (tr). Hay que hacer notar que este tiempo depende del comando que se envíe y los factores descritos en el apartado siguiente.



* Se da solo una vez durante el proceso de pruebas y es constante para cada central, lo único que varia es (te) dependiendo del tipo y configuración del modem.
** El tiempo de demora es variable.

Fig.2.4-1 Gráfica Tiempos Para Comandos Enviados Desde Terminales Remotas.

tm	3 seg.
te	6-51 seg.
tp	26 seg.
tc	0.5 seg.
td	0-143 seg.
tr	depende comando

Fig. 2.4-2 Tiempos Para Comandos Enviados Desde Terminales Remotas.

Factores que Afectan en los Tiempos de Respuesta de las Centrales.

Analizando el problema del tiempo que se tarda en responder la central desde que se envía un comando hasta que se recibe la respuesta completa (tr), podemos considerar los tiempos mostrados en la figura 2.4. - 2 como reales puesto que fueron efectuadas pruebas medidas con cronómetro, y observándose en diferentes circunstancias, concluyendo que el tiempo ideal es el mostrado en la tabla y que varía dependiendo de los siguientes factores:

1) Tipo de central.

Entre una central y otra existe mucha diferencia de tiempo de respuesta, por lo que consideraremos como uno de los factores principales, aunque únicamente analizaremos dos tipos de ellas: la central tipo EXA que de ahora

en adelante denominaremos C1 y la central tipo ISS2199 que de ahora en adelante denominaremos C2;

2) Tipo de software se comunicaciones de la central y revisión de este (tsoft).
Este punto se refiere al modo que se accese la central, y este puede ser de manera directa (ej. IOC12), a través de un administrador (ejem: MOA).

3) Carga de trabajo de la central.
La carga de trabajo es otro factor importante en el rendimiento de la respuesta de la central de los cuales podemos clasificarlos de la siguiente manera:
a) Ocupación de la central por el tráfico de llamadas (ot).
b) Ocupación de los dispositivos electrónicos que son usados por funciones de la misma familia pero no los mismos (od).
c) Ocupación de las funciones por otra terminal remota (of).

4) Tipo de Comando.
Para el presente trabajo solo se usarán 4 funciones de entre todas, y que en base a la observación tiene tiempo de respuesta diferentes, donde:
cm1 = consulta de categorías
cm2= consulta de contadores
cm3= cambio de categorías
cm4= prueba de línea

5) Tipo de comunicaciones que se usan.
Entre estos factores que afectan aunque no de manera tan importante como los dos primeros, encontramos los siguientes:
a) Velocidad de transmisión en baudios (bps).

b) Línea Privada o conmutada (lp ó lc).

c) Velocidad de la Computadora personal o terminal (MIPS).

Nota: velocidad de transmisión en bauds afecta poco puesto que se genera un cuello de botella al esperar la respuesta de la central que aún es mas lenta que una velocidad de 1200 bps.

Mediciones Efectuadas.

Tipo de comando / Tipo de software	c1 tsoft1	c1 tsoft2	c2 tsof1	c2 tsoft2
cm1	16 seg.	8 seg.	16 seg.	16 seg.
cm2	14 seg.	7 seg.	16 seg.	16 seg.
cm3	18 seg.	9 seg.	25 seg.	25 seg.
cm4	25 seg.	13 seg.	31 seg.	31 seg.

Fig. 2.4.-3 Mediciones Efectuadas.

Donde:

c1tsoft1 = EXA MOA

c1tsoft2 = EXA IOC12

c2tsoft1 = Sistema 2199 Man-machine

c2tsoft2 = Sistema 2199 MPTMON

ot = hora intermedia de ocupación

od = mínima

of = mínima

bps=1200 en lc

Se ha comprobado experimentalmente que estos tiempos realmente son el promedio estándar tanto para centrales grandes como para pequeñas.

Estimación de Rendimiento por Tipo de Central.

Debido a la cantidad de factores que pueden afectar durante la prueba masiva de líneas, lo que se puede hacer es un cuadro comparativo de número de líneas que puede probar un sistema dedicado considerado en condiciones ideales, de esta manera tenemos los siguientes datos:

Tipo de comando / Tipo de software	c1 tsoft1	c1 tsoft2	c2 tsof1	c2 tsoft2
cm1	225 lin.	450 lin.	225 lin.	225 lin.
cm2	257 lin.	514 lin.	225 lin.	225 lin.
cm3	200 lin.	400 lin.	144 lin.	144 lin.
cm4	144 lin.	285 lin.	116 lin.	116 lin.

Fig. 2.4.-4 Capacidad de Líneas a Probar.

Donde:

c1tsoft1 = EXA MOA

c1tsoft2 = EXA IOC12

c2tsoft1 = Sist2199 Man-machine

c2tsoft2 = Sist 2199 MPTMON

ot = horas intermedia de ocupación

od = mínima

of = mínima

bps=1200 en lc

Ejemplo:

Para ver los requerimientos de una ciudad pequeña, vamos a suponer que en una ciudad X se tienen 2 centrales: una EXA IOC12 con 300 teléfonos públicos y otra Sistema 2199 Man-machine, suponiendo una ocupación de la central que nos permite el 70% de tiempo, y se desea hacer una prueba de contadores y categorías, la pregunta sería: ¿Es posible controlar con una sola computadora? y ¿Cuanto tiempo requerirá para hacer esto?

EXA IOC12

(300lin) / (450lin/hr) / (.70) = 57 min consulta de contadores

(300lin) / (514lin/hr) / (.70) = 50 min consulta de categorías

Sistema 2199 man-machine

(300lin) / (214lin/hr) / (.70) = 120 min. consulta de contadores

(300lin) / (214lin/hr) / (.70) = 120 min. consulta de categorías

---

5 hrs. 47 minutos aprox.

Donde notamos que no se consideran los tiempos de enlace del modem, por estar considerados de una u otra manera con el factor de ocupación de la central.

Valoración de Tiempos de Respuesta de las Centrales.

Aún con los tiempos de respuesta que se pueden ver bastante grandes, hay que considerar que para efectuar estas pruebas de manera manual, entra el error del factor humano, además que sería necesario dedicar una persona para estos fines.

Concluyendo, al incorporar un autómata que interactúe con la central se incorpora la funcionalidad de cualquier sistema de procesamiento externo.

2.5. Estimación de Tiempos de Respuesta de la Computadora a Usar.

Memoria del Disco Duro.

En las computadoras que cuentan con un microprocesador 80486, existen varios sistemas controladores de disco duro, uno de estos es el sistema controlador llamado (IDE), *integrated drive electronics*, se encuentra, por ejemplo en el sistema 2 de IBM y en sus clones.

Este sistema incorpora el controlador en el manejador del disco y agrega a este el sistema a través de un cable pequeño que actúa como interfaz. Esto permite que muchos controladores de disco se conecten al sistema sin conflictos en el bus o con el controlador. La interfaz IDE es capaz de manejar otros dispositivos de I/O aparte del disco duro. Es común encontrar 32 KB de memoria cache para los datos del disco. El cache acelera las transferencias del disco. Los tiempos de acceso para un manejador IDE son de por lo menos de 12 ms donde el tiempo de acceso para un floppy es de 200 ms.

Arquitectura Básica del 80486.

La arquitectura del microprocesador 80486DX es casi idéntica a la del 80386 más el coprocesador matemático 80387 y un cache interno de 8 Kb. El 80486SX es casi idéntico a un 80386 con un cache de 8 Kb. Si se compara la estructura interna básica de un 80486 y un 80386, no se observan diferencias. La diferencia más significativa entre el 80386 y un 80486 es que casi la mitad de las instrucciones del 80486 son ejecutadas en periodos de 1 pulso de reloj en lugar de 2 pulsos de reloj que se ocupan para las instrucciones del 80386.

Tiempo de Ejecución de las Instrucciones.

El tiempo de ejecución esta definido para cada instrucción del código ensamblador por un número definido de ciclos por segundo. La duración de cada periodo de reloj esta determinado a partir de la velocidad del sistema de reloj por la fórmula:

1 pulso de reloj = 1000 / (la velocidad del sistema en MHz) y el resultado se expresa en nanosegundos (ns).

Para determinar el tiempo de ejecución de una serie de instrucciones no es tan simple como sumar los diferentes tiempos de ejecución de todas las instrucciones. Un gran número de variables influyen para hacer que este tiempo sea un poco mayor. Los eventos que ocurren afuera del sistema (interrupciones, excepciones, refrescamientos de memoria y más) influyen en la ejecución del tiempo real de las máquinas.

El tiempo de ejecución varia considerablemente. Por ejemplo, la ejecución del tiempo para los saltos condicionados cambia si el control se transfiere o no se transfiere. Otras instrucciones cambian en su tiempo de ejecución si este se desarrolla en modo real o en modo protegido.

Si una dirección de memoria se necesita calcular, esto puede retardar el proceso de la ejecución. El tiempo de ejecución para instrucciones repetitivas pueden variar como una función del factor de repetición. Finalmente la alineación de los comandos de datos puede afectar el tiempo que se emplea en su búsqueda.

El tiempo de ejecución de cada una de las instrucciones individuales se calcula bajo los siguientes supuestos:

- La instrucción es buscada y colocada en la cola de espera.
- Se incluyen los ciclos de reloj que sean necesarios para reinicializar la cola de instrucciones para las instrucciones de transferencia del control.
- Todas las operaciones de memoria están alineadas.
- Los ciclos del bus no requieren estados de espera.
- No hay otro procesador buscando acceso al bus.
- Los componentes internos del microprocesador no buscan accesar el bus.
- No se detectan excepciones durante la ejecución de las instrucciones.

Una serie de instrucciones que toman menos de dos ciclos de reloj por byte, pueden vaciar la cola y el procesador permanece ocioso hasta que llegan más instrucciones.

Los estados de espera incrementan el tiempo en que el procesador hace la petición de datos y la recepción de estos. Los estados de espera no deben sumarse necesariamente al tiempo de ejecución a menos que la cola de instrucciones este vacía y que el acceso al bus y las instrucciones de ejecución se ejecuten de manera simultánea.

Como se ve, es necesario tomar en cuenta el ambiente externo. Muchas configuraciones de PC's usan un procesador sencillo, y suponen que no hay un congelamiento en el bus como consecuencia de una petición del procesador para acceder a este. Los ciclos de reloj no se tienen que considerar para interrupciones externas que ocurren durante la ejecución. Estas deben de

tomarse como que ninguna excepción se ha detectado durante la ejecución de las instrucciones.

Los componentes internos de el procesador pueden competir por el acceso al bus. Por ejemplo, si parte del procesador (la unidad de ejecución) esta ejecutando una instrucción que accesa a la memoria, otra parte del procesador (la unidad de interface del bus) podría esperar a que se busque la instrucción y viceversa.

Debido a todos los factores que no pueden predecirse con certeza, un programa promedio puede tomar aproximadamente entre 5 y 10 por ciento más para ser ejecutado de lo que indican los tiempos de reloj si se consideran las líneas de instrucciones que conforman al programa. Estos errores pueden ser significativos para sistemas que trabajan en tiempo real donde los ciclos de reloj y la velocidad de respuesta del procesador son críticos. En esos casos, se debe utilizar hardware especial para determinar el tiempo de ejecución exacto.

El 80286.

En el 80286, los ciclos de reloj presentados están sujetos a las siguientes consideraciones:

Los operandos están alineados en la memoria. Una penalización de 2 ciclos de reloj para cada acceso a un operando en memoria de 16 bits en una dirección impar.

Los cálculos de direccionamiento efectivo no usan la forma base + índice + *desplazamiento*. Si esta forma es utilizada, entonces se penaliza con un ciclo de reloj.

El 80386.

En el 80386, se aplican las siguientes suposiciones:

- Los operandos están alineados.

  En el 80386DX: se penaliza con 2 ciclos de reloj para accesar un operando de memoria de 32 bits en una dirección física que no es divisible entre 4.
  En el 80386SX: se penaliza con 2 ciclos de reloj por accesar un operando de memoria de 16 bits en una dirección de memoria impar.
  80386SX: se penaliza con 2 ciclos de reloj por accesar un operando de memoria de 32 bits aun en memoria física.
  80386SX: una penalización de 4 ciclos de reloj es aplicada por accesar un operando de memoria de 32 bits en una memoria física impar.

- El cálculo de direccionamiento efectivo no usa los 2 registros de propósitos generales. Un registro, y un desplazamiento pueden ser usados con los ciclos de reloj normales. Si el cálculo de direccionamiento efectivo usa dos registros de propósito general, se suma un ciclo de reloj de penalización.

El 80486.

En el 80486, los ciclos de reloj se especifican a partir de la suposición de que cada estado de los componentes internos del procesador guarda un orden. Estas suposiciones adicionales y las penalizaciones respectivas son listadas a continuación.

Ambos, datos e instrucciones "pegan" en el cache. Los ciclos de reloj del 80486 asumen que la búsqueda en memoria, tanto de datos como de instrucciones pueden ser encontradas en el cache.

Si sucede una perdida en el cache, el 80486 puede necesitar el uso de un ciclo de bus externo para transferir el código requerido o los datos de vuelta al cache.

Memoria Cache.

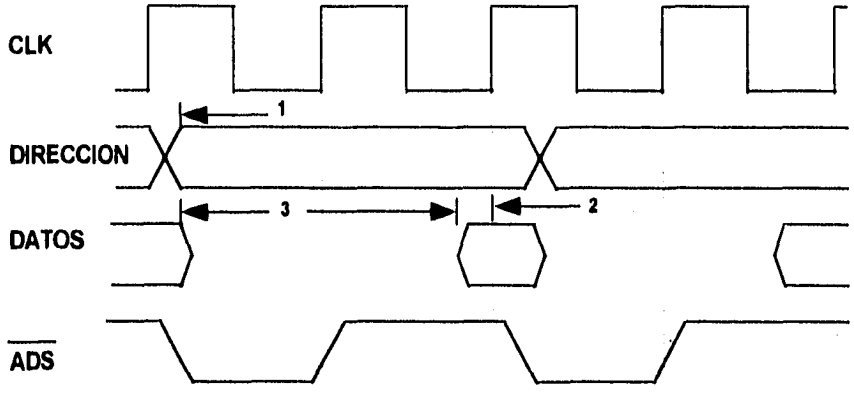
El sistema de memoria cache almacena datos usados por un programa y también las instrucciones de ese programa. El cache esta organizado como un conjunto de 4 bloques que contienen 16 bytes o cuatro palabras dobles de datos. En muchas ocasiones, las porciones activas de un programa están totalmente dentro de la memoria cache. Esto provoca que la ejecución se produzca en cada pulso de reloj para muchas de las instrucciones que son usadas comunmente en un programa. Si nos referimos a los datos, estos son también almacenados en una memoria cache, solo que su impacto en la velocidad de procesamiento es menor debido a que los datos no son usados tan frecuentemente como lo pueden ser las porciones de código de un programa.

Debido a que la memoria cache es nueva en el microprocesador 80486 y esta memoria se llena usando ciclos burst los que no se encuentran en el microprocesador 80386, se hace necesario de algunos detalles para entender su funcionamiento. Cuando una línea de bus es llenada, el 80486 debe adquirir 4 números de 32 bits de la memoria del sistema para llenar una línea en el cache. El llenado se realiza con un ciclo burst. El ciclo burst es una memoria especial donde 4 números de 32 bits son buscados de la memoria del sistema en periodos de 5 pulsos de reloj. Aquí se toma como un supuesto que la velocidad de la memoria es suficiente y que no se requiere de ningún estado de espera. Si la frecuencia del reloj de un microprocesador 80486 es de 33 MHz, entonces será posible llenar una línea del cache en 167ns, esto es muy eficiente considerando que una operación de lectura de memoria de 32 bits requiere de 2 pulsos de reloj.

Lectura de Memoria.

La figura 2.5-1 ilustra el tiempo de lectura para el 80486 en una operación de memoria ininterrumpida. Obsérvese que 2 pulsos de reloj se usan para transferir los datos. El periodo T1 permite el direccionamiento de memoria y las señales de control; en el periodo T2 es donde los datos son transferidos entre la memoria y el microprocesador. Nótese que el RDY debe ser un cero lógico para que los datos sean transferidos y se termine el ciclo de bus. El tiempo de acceso para un acceso ininterrumpido es determinado tomando 2 pulsos de reloj menos el tiempo requerido para el bus de direcciones menos el tiempo requerido para la conexión del bus de datos. Para la versión de 20 MHz del 80486, 2 pulsos de reloj requieren 100 ns menos 28 ns para el setup de dirección menos 3 ns para el setup de datos. Esto es igual a 100-31 = 69 ns. 69 ns de

tiempo de acceso. Por supuesto que si el tiempo de decodificación



	20 MHz	
Time 1:	3 - 28 ns	Tiempo de retardo por direccionamiento
Time 2:	6 ns	Tiempo del setup de datos
Time 3:	76 ns	Tiempo de acceso

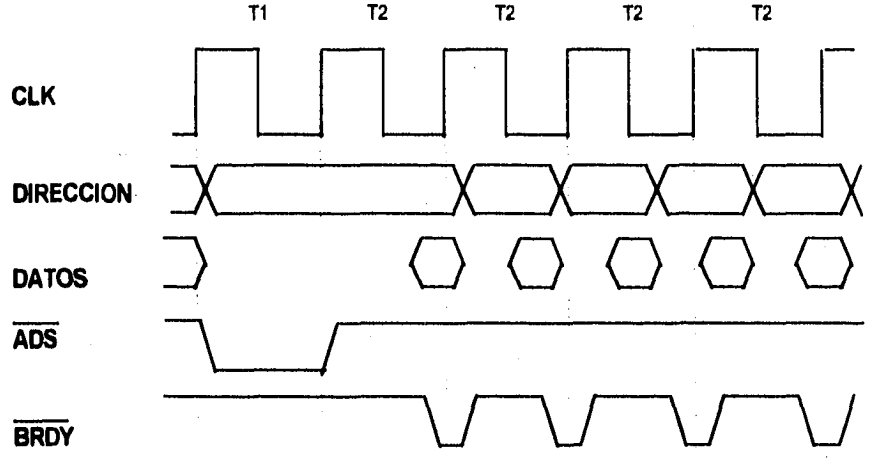


Fig. 2.5-1 Tiempo de Lectura en el Microprocesador 80486. Ciclo Burst que Lee 4 Palabras Dobles en 5 Pulsos de Reloj.

y los tiempos de retardo son incluidos, el tiempo de acceso permitido hacia la memoria es menor para cualquier operación sin estados de espera. También si una versión de mayor frecuencia del 80486 es utilizada en el sistema, el tiempo de acceso a memoria es aún menor.

La figura ilustra el diagrama de tiempos para el llenado de una línea cache con 4 números de 32 bits usando una interrupción burst.

Destaca el hecho de que las direcciones (A31 - A4) aparecen durante T1 y permanecen constantes en todo el ciclo burst. Nótese que A2 y A3 cambian durante cada T2 después de direccionar 4 números de 32 bits consecutivos en el sistema de memoria. Como se menciono antes, el llenado del cache usando burst requieren de 5 pulsos de reloj (un T1 y 4 T2) para llenar una línea de cache con 4 palabras dobles de datos. El tiempo de acceso usando la versión de 20 Mhz de el 80486 para la segunda y subsecuente palabra doble es de 50 ns - 28 ns - 3 ns o bien de 19 ns si se supone que no habrá retardos en el sistema. Para usar el modulo de transferencia burst, necesitamos de memoria de alta velocidad. Como el tiempo de acceso DRAM es en el mejor de los casos de 55 ns, estamos obligados a usar SRAM para transferencias de ciclo burst.

Teniendo los datos anteriores, es posible, a manera de ejemplo, el análisis del performance de 2 computadoras.

Los diferentes fabricantes de computadoras que utilizan el microprocesador 486 SX tienen tiempos de respuesta muy parecidos en sus componentes. A continuación haremos un breve análisis de dos computadoras una AMERITEK'S AMK T420SX y una ARES 486SX/20 SONIC para posteriormente presentar las pruebas de laboratorio en donde se compara el performance de estas dos

máquinas con dos marcas muy conocidas una COMPAQ 486/25 Y UNA IBM PS/2 modelo 70/25.

La AMERITEK'S AMK ofrece un microprocesador 486/SX que corre ha una velocidad de 25 MHz con una memoria RAM de 4MB la cual tiene un tiempo de acceso de 70 nanosegundos (ns). Se puede escalar hasta 32 MB de RAM y hasta 64 MB de RAM con una tarjeta opcional conectada a la motherboard. Esta posee 8 slots de expansión, 7 de 32 bits y uno de 16 bits. El floppy es de 3.5 o 5.25" y para el disco duro, se puede tener acceso a 205 Mb de memoria a una velocidad de 14 milisegundos de acceso (ms). Su memoria de video es manejada por una tarjeta ATI de 1 MB. Posee dos puertos y un puerto de juegos.

La computadora ARES 486/20 SONIC equipada con un procesador 486DX/33. Su monitor es un Viewsonic 5 1,024 x 768 controlado por la tarjeta SpeedStar 1 MB. Su disco duro de 210 MB que opera a 13.5 ms de marca Quantum. Posee una memoria RAM de 4 MB con un acceso de 70 ns.

Ambas computadoras fueron sujetas a pruebas de laboratorio, los resultados se presentan a continuación.

Evaluación del Sistema AMKT420SX

	AMK T420SX	COMPAQ 486/25	IBM PS/2 MODEL 70/25
Processor Performance Tests			
Instruction Mix - 80386	1.89	1.44	2.19
Instruction Mix - 80486	1.92	1.46	N/A
Disk Performance Tests			
Random BIOS Disk Seek	**15.27**	**19.88**	**21.09**
DOS File Access (Small Records)			
B12 Records / 512 Bytes Bach			
File Create	9.06	8.75	8.99
Sequential Write	8.90	17.25	17.30
Sequential Read	0.88	0.47	8.75
Random Write	7.91	13.39	17.39
Random Read	7.47	5.40	9.50
Total	**34.22**	**45.26**	**81.93**
DOS Variable Size File Access (Small Records)			
300 Records / 200 Bytes Bach			
File Create	2.14	2.11	2.18
Sequential Write	4.06	3.96	3.99
Sequential Read	0.22	0.13	2.01
Random Write	39.94	28.18	38.96
Random Read	7.47	9.85	19.04
Total	**52.83**	**44.23**	**86.18**
Memory Performance Tests			
Conventional Read	0.33	0.28	0.33
Conventional Write	0.33	0.17	0.33

Ameriteck International 1061 N. hepard St., Bldg. C, Anaheim, CA 92806 (714) 666-0210

Sugg List Price (as revlewed) $ 2,395.00

System Includes: 486 SX/20 Mhz; 4 MB 70 ns RAM; 256 kb SRAM cache; 1.2 Mb 5.25 inch floppy drive; 1.44 3.5 inch floppy drive; 208 Mb Conner hard disk; ATI Supper VGA adapter w/1Mb memory; optiquest flat - screen multiscanning monitor; two serial, one pararell, and one game port; 250 watt power supply; 101 keyboard.

PC Magazine Laboratory Benchmarks Series realise 5.6

ARES 486/20 SONIC

	AMK T420SX	COMPAQ 486/25	IBM PS/2 MODEL 70/25
Processor Performance Tests			
Instruction Mix - 80386	1.82	1.44	2.19
Instruction Mix - 80486	1.86	1.46	N/A
Disk Performance Tests			
Random BIOS Disk Seek	**14.72**	**19.88**	**21.09**
DOS File Access (Small Records)			
B12 Records / 512 Bytes Bach			
File Create	8.63	8.75	8.99
Sequential Write	8.62	17.25	17.30
Sequential Read	0.71	0.47	8.75
Random Write	6.26	13.39	17.39
Random Read	8.43	5.40	9.50
Total	**30.65**	**45.26**	**81.93**
DOS Variable Size File Access (Small Records)			
300 Records / 200 Bytes Bach			
File Create	2.14	2.11	2.18
Sequential Write	3.96	3.96	3.99
Sequential Read	0.22	0.13	2.01
Random Write	31.96	28.18	38.96
Random Read	10.93	9.85	19.04
Total	**49.21**	**44.23**	**86.18**
Memory Performance Tests			
Conventional Read	0.50	0.28	0.33
Conventional Write	0.38	0.17	0.33

Ares Microdevelopment, Inc.. 24762 Creatview Ct., Farmington Hills MI 48335; 1-800-322-3200; (313) 473-0808
Sugg List Price (as revlewed) $ 3,200.00
System Includes: 488 SX/20 CPU runing at 20 Mhz; 4 MB RAM; 64 kb SRAM cache; 1.2 Mb 5.25 inch floppy drive; 1.44 3.5 inch floppy drive; 210 Mb Quantum hard disk; Speedstar Supper VGA adapter w/1Mb memory; Viewsonic 5 1,024 x 768 monitor; two serial, one pararell, and one game port; 250 watt power supply; 101 keyboard with calculator.

PC Magazine Laboratory Benchmarks
Series realise 5.6

El Microprocesador Pentium.

Para entender porque el Pentium lleva el desempeño de un 80x86 a nuevos horizontes, se necesita comparar el Chip de Intel más rápido el 486DX2 a 66 Mhz. Esta comparación es de gran utilidad porque los relojes internos de los procesadores corren a la misma velocidad. Esto nos lleva a observar que las diferencias de desempeño son resultado de las diferentes formas de

implementación entre uno y otro chip. El Pentium es un gran chip, atestado de 3.1 millones de transistores dentro de 2.16 pulgadas cuadradas, con 273 pin PGA (pin grid array). A una velocidad de 66 Mhz el Pentium consume hasta 13 watts de potencia. El chip Pentium de Intel es más rápido y potente. Si comparamos las primeras versiones de 60 Mhz y de 66 Mhz, estas procesan en un 75 por ciento más rápido si se comparan con un 486DX2/66. Para poder entrever el porque de esto, la unidad de punto flotante nos pueda dar la respuesta.

La Unidad de Punto Flotante.

El desempeño del punto flotante siempre había sido el punto flaco en la arquitectura 80x86. Este "descuido" por parte de Intel se entiende dadas las relativamente pocas aplicaciones de punto flotante para PC. Con el incremento en el uso de gráficas y aplicaciones multimedia, la necesidad de un buen desempeño de la aplicación de punto flotante es mucho mayor que en los días que se introdujo el microprocesador 80486. También, era importante que el microprocesador Pentium estuviera a la altura de las architecturas RISC más populares.

En su gran mayoría, el desempeño del punto flotante es una función de que tanto silicón se dedica al trabajo. Con los 3.1 millones de transistores contenidos en el microprocesador, Intel nos permite trabajar a nuestras anchas. En el rating SPECfp92, Pentium logro una calificación 3.5 veces superior a la del 486. El avance más importante es que la floating-point unit (FPU) contiene unidades dedicadas a sumar, multiplicar y dividir. La consecuencia de estos circuitos dedicados es pasmoso. No importa la precisión; el sumador y el multiplicador completan su operación en 3 pulsos de reloj. La unidad de división tiene un estado de espera mucho mayor; esta produce 2 bits de cociente por pulso de

reloj. Estos ciclos de tiempo representan un gran avance sobre el desempeño de la FPU del 486, en donde una instrucción FADD tomaba 10 pulsos de reloj y una instrucción FMUL tomaba entre 12 y 15 pulsos de reloj.

Si bien el Pentium es significativamente más rápido que el 80486, esta diferencia no se debe a que los tiempos de acceso de los dispositivos sean mas pequeños. Al menos en las primeras versiones de las computadoras equipadas con procesadores Pentium, la velocidad de acceso a memoria RAM era muy parecida al tiempo de acceso que se tomaban las computadoras 486DX. La diferencia radica en la arquitectura interna de los microprocesadores; la cantidad de pulsos de reloj que se requieren para ejecutar las diferentes instrucciones y las 2 unidades aritméticas lógicas del Pentium hacen que su performance sea mucho mejor.

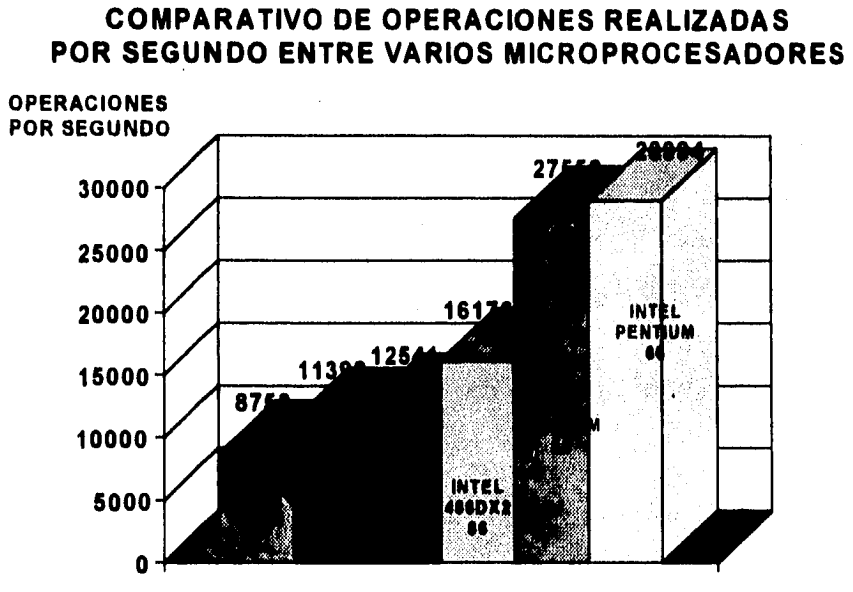


Fig. 2.5-2 Gráficas Comparativas Entre el Microprocesador Pentium y los Microprocesadores 80486.

COMPARATIVO ENTRE EL MICROPROCESADOR PENTIUM, Y LOS MICROPROCESADORES 80486 DX/66 Y 80486 DX/33

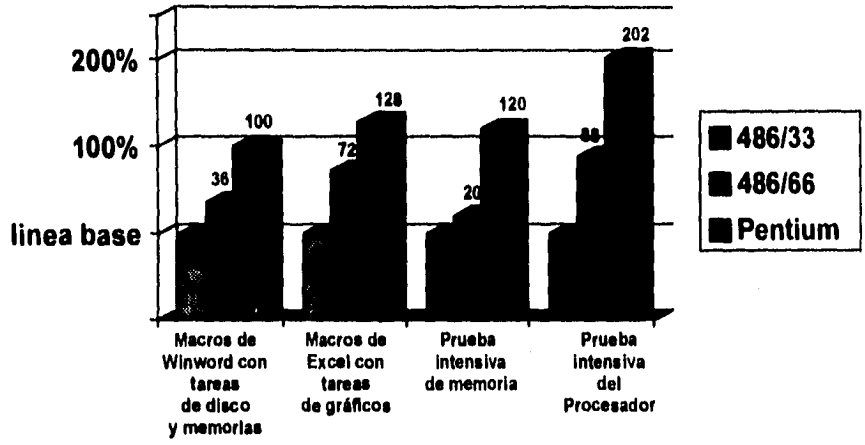


Fig. 2.5-2 Gráficas Comparativas entre el Microprocesador Pentium y los Microprocesadores 80486. (Continuación).

ESPECIFICACIONES	Evolution VQ/60	Compaq Desktop 8/66M	HP NetServer 5/60 LM
RAM instalada/velocidad	32MB/70 ns	24 MB/60ns	32 MB/70 ns
RAM máxima	1GB	136MB	384 MB
Cache externo CPU	512K	256 K	256K
Arquitectura de Bus	EISA, VL - Bus	EISA	EISA
Tamaño	Doble Torre	Desktop	Doble Torre
PERIFERICOS			
Disco Duro	Conner CP30540 (SCSI)	Conner CP30540 (SCSI)	Seagate 1288N (SCSI)
Disco Duro configuracion probada	Arreglo de Cuatro Manejadores de 540 MB	Un manejador de 510 MB	Arreglo deCuatro Manejadores de 1GB.
Controlador del Disco Duro	UltraStor 124F SCSI	IDE integrado	Mylex SCSI-2
Manejadores adicionales	3.5" floppy	3.5 y 5.25" floppies	3.5" floppy
Video	Chip WD90C31 (sobre una tarjeta)	Tarjeta Qvision EISA y un monitor de 20"	Chip WD90C31 (integrado)
EXPANDIBILIDAD			
Slots de expansión totales	10/8 EISA, 3 dobles como VL Bus	7/5 EISA	8/5 EISA
Bahías para Manejadores (internos y externos)	0/13	1/3	0/8
Actualización del CPU	No	Yes	Yes

Como una conclusión de este punto podemos observar una variación significativa entre el desempeño de una computadora equipada con un microprocesador 80486 y otro equipo con un microprocesador Pentium. Sin embargo las diferencias que se puedan observar entre cualesquiera de estos equipos no afectan el funcionamiento del sistema "ALCANCIAS" ya que, apoyándonos en los resultados del punto 2.3 de este capítulo, independientemente de las características de estos equipos, ambos están sobrados y por mucho para los requerimientos mínimos del sistema, razón por la cual no importan mucho que tan bueno es un equipo respecto al otro.

2.6 Propuesta de Solución y Elección de la Optima.

Diferentes Opciones de Solución.

Debido al avance a paso acelerado de la computación actualmente se cuentan con muchas alternativas para la solución de problemas relacionados con el proceso de un sistema computacional, desde múltiples opciones de Hardware, Sistemas Operativos, hasta las Herramientas de Programación. Estas alternativas se multiplican mas considerando que dentro de una herramienta de programación hay varias opciones de: librerías de apoyo, algoritmos, filosofías de programación, etc. (ver figura 2.6. - 1)

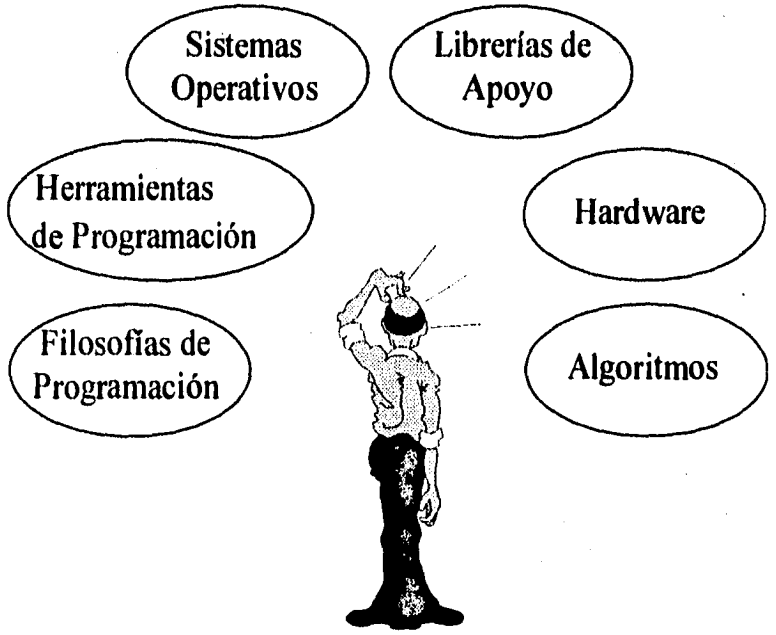


Fig. 2.6.1 Elección de Solución.

Un enfoque para encontrar la mejor solución, es centrarse en el problema principal y de ahí derivar la mejor alternativa para el desarrollo; por supuesto previendo todos los requerimientos que debe tener un buen sistema como lo son: buena relación costo-beneficio, tiempos de entrega, tiempos en adecuaciones del sistema, compatibilidad con otras plataformas, etc.

El elegir una opción no es del todo libre, hay que considerar que se puede contar con la opción de comprar nuevos recursos o utilizar la infraestructura con la que se cuenta. En caso de elegir utilizar los recursos ya existentes, se acorta las opciones de solución (Ver figura 2.6-2), y que de alguna manera encaminan a un universo menor de soluciones, pero dentro de las cuales se puede encontrar la optima, es decir dar el mayor rendimiento con el mínimo de recursos.

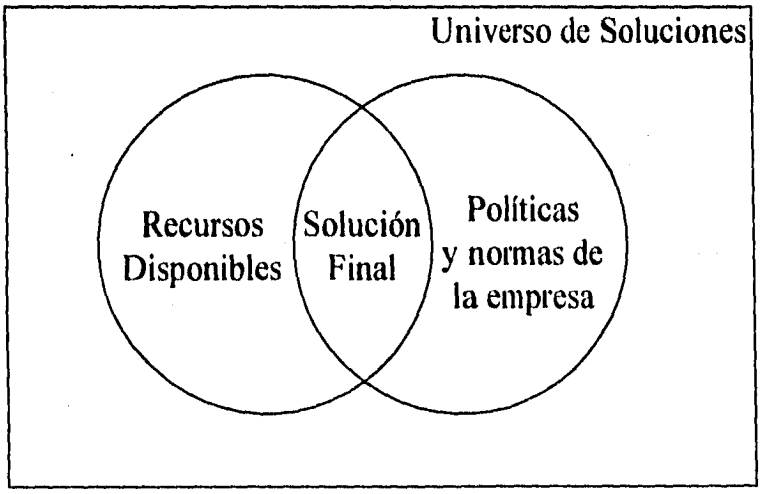


Fig. 2.6-2 Libertad en la Solución.

Como es de esperarse, en los requerimientos del sistema se visualiza que el desarrollo de un sistema de monitoreo de teléfonos públicos desde su concepción es complejo, y esto se debe a que el problema principal radica en un Programa de Comunicaciones (ver figura 2.6-3) y todo el éxito de un sistema de este tipo depende que se haga buen uso de las de los puertos seriales RS232C, que tenga un buen protocolo de comunicación con el modem y que sea fácilmente adaptable a protocolos de comunicación con diferentes centrales y versiones de software de estas.

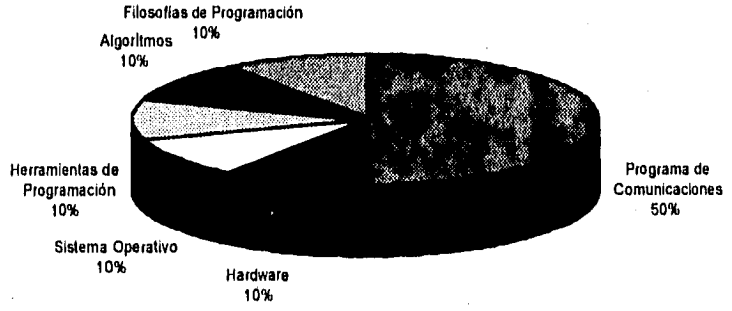


Fig. 2.6-3 Ponderación en la Solución.

También se puede ver en los requerimientos del sistema, que se necesita como apoyo, un Manejador de Bases de Datos para mantener información actualizada del status de las líneas telefónicas a monitorear y elaborar de manera sencilla reportes.

Además también se debe elegir un ambiente amigable en la presentación de pantallas, que va ligado directamente con el Sistema Operativo puesto que de éste se tiene la interacción con el usuario.

A continuación se mencionarán las razones por las cuales se eligieron de tantas una opción en Hardware y en Sistema Operativo, para después plantear las diferentes alternativas de solución haciendo un breve análisis de ventajas y desventajas en cada una de ellas.

Elección del Hardware.

Para la elección del hardware se prefirieron las computadoras personales (PC's), puesto que actualmente en la mayoría de las compañías se cuenta con una infraestructura de este tipo. Además por las características planteadas a desarrollar del sistema, en cuestión de consumo de recursos serán muy pocos debido a que no necesita de una computadora bien equipada, incluso puede llegar a trabajar solo con la memoria convencional de una computadora, es decir en los 640 K de memoria principal. Y no necesita de grandes cantidades de disco duro y la resolución del monitor, tampoco es importante.

Elección de la Plataforma de Desarrollo.

En este aspecto es donde se presentó la disyuntiva de elegir entre el DOS o Windows debido a que en los dos ambientes, se cuentan con programas de comunicaciones muy eficientes, y documentados con sus fuentes, pero debido a que se utilizan en Windows Communications API Functions y la filosofía de Windows MVC, complica cualquier programa de comunicaciones, aún el básico como el TERMINAL.EXE contenido en Windows. A diferencia en DOS que permite flexibilidad para las rutinas de comunicaciones, y permite una gran flexibilidad de implementación.

Estas consideraciones repercuten directamente en el tiempo de desarrollo, por lo que desarrollar en Windows, definitivamente es más elegante pero en cuestión de desarrollo podría considerarse el doble de tiempo.

Son por estas razones que es preferible escoger el DOS como la plataforma de desarrollo.

Selección del Lenguaje de Desarrollo.

Debido a las características del sistema en la interacción con la máquina, se eligió el lenguaje de tercera generación denominado "C", (este también es denominado lenguaje de programación moderno o estructurado) por sus potentes posibilidades procedurales y estructuración de datos, este lenguaje es de alto nivel de propósito general, además de su posibilidad de ser escalable a lenguaje de objetos.

"C" soporta estructura de datos sofisticadas y tiene características de tipificación, hace uso extensivo de apuntadores y tiene un rico conjunto de operadores para cálculo y manipulación de datos. Además, permite al programador "acercarse a la máquina" al suministrar posibilidades similares al lenguaje ensamblador.

Cabe mencionar que dentro de las aplicaciones mas importantes que hagan uso de dispositivos de la computadora, han sido desarrolladas la mayoría en lenguaje "C".

Par lo que para el problema que se nos presenta que es el hecho de manejar dispositivos externos como el puerto serial RS232C de la máquina, así como

modems, es de considerarse como una de la mejores opciones, sino es que la mejor.

Dentro de los compiladores en C, también tenemos varias opciones

Borland C++
Visual C++
Latice C
Turbo C+
Watcom C++

Sin embargo la elección de este dependerá del compilador que haya sido usado para el programa de comunicaciones con mayor eficiencia en la interacción con la central, y el objetivo del proyecto es que debe ser fácilmente transportado a otro compilador.

Selección de Programa de Comunicaciones.

Dentro de todas las selecciones mencionadas en esta parte, no cabe duda que la más importante es la selección del programa de comunicaciones, porque a partir de este, se incorporarán otras herramientas como el manejador de base de datos y las presentaciones de pantallas.

La elección de este depende directamente del comportamiento interactivo con las centrales telefónicas, por lo que se consideraron 2 programas hechos en C++ para DOS, documentados con sus respectivos programas fuentes: el "uno_dos" y el programa "simple", obteniéndose mejores resultados de comportamiento en el segundo.

modems, es de considerarse como una de la mejores opciones, sino es que la mejor.

Dentro de los compiladores en C, también tenemos varias opciones

Borland C++
Visual C++
Latice C
Turbo C+
Watcom C++

Sin embargo la elección de este dependerá del compilador que haya sido usado para el programa de comunicaciones con mayor eficiencia en la interacción con la central, y el objetivo del proyecto es que debe ser fácilmente transportado a otro compilador.

Selección de Programa de Comunicaciones.

Dentro de todas las selecciones mencionadas en esta parte, no cabe duda que la más importante es la selección del programa de comunicaciones, porque a partir de este, se incorporarán otras herramientas como el manejador de base de datos y las presentaciones de pantallas.

La elección de este depende directamente del comportamiento interactivo con las centrales telefónicas, por lo que se consideraron 2 programas hechos en C++ para DOS, documentados con sus respectivos programas fuentes: el "uno_dos" y el programa "simple", obteniéndose mejores resultados de comportamiento en el segundo.

Selección del Manejador de Base de Datos

El hecho de manejar datos relacionados a los números telefónicos conectados nos induce a un problema de bases de datos, permitiéndonos de esta manera un control centralizado de la información.

Como se podrá ver mas adelante el entorno del manejador de base de datos depende o gira entorno de la elección del compilador que se haya usado en el programa de comunicaciones, puesto que sería lógico, elegir el mejor manejador de bases de datos, dejando en segundo término el programa de comunicaciones.

De ser posible se debe tener alta compatibilidad con bases de datos que se manejan por lo regular en PC's como puede ser DBASE, FOXPRO y CLIPPER, por esta razón se eligió el CODEBASE porque cumple con estos requisitos.

Selección del Manejador de Pantallas.

Lo mismo que sucede con la elección de base de datos, sucede con la elección del manejador de pantallas, que gira en torno de la elección del compilador que se haya usado en el programa de comunicaciones.

Como se puede pensar, no es estrictamente necesario contar con una herramienta como estas, puesto que en DOS se pueden desarrollar rápidamente programas que haga una excelente presentación sin uso de librerías auxiliares, sin embargo, para optimizar tiempos de desarrollo es importante no desperdiciar ninguna herramienta que se pueda tener en mano, por esta razón se eligió el C-SCAPE que permite utilizar como printf de C,

pantallas de captura y menús, reduciendo considerablemente tiempos de desarrollo.

Relación Costo-Beneficio.

Entre la información mas relevante que contiene el estudio de viabilidad es el análisis costo-beneficio, una evaluación de la justificación económica para un proyecto basado en computadora. El análisis de costo beneficio señala los costos de desarrollo del proyecto y los contrasta con los beneficios tangibles, (esto es medible en pesos) e intangibles del sistema.

El análisis costo-beneficio es complicado porque los criterios varían según las características del sistema a desarrollar, sin embargo en el presente proyecto se pueden encontrar dos subrutinas importantes que repercuten directamente en dinero lo que puede servir como pauta o punto de equilibrio para el estudio de factibilidad del proyecto, y estas son:

- Al incorporar el control de categorías, se produce un ahorro significativo en los "robos de llamadas"; fenómeno que se le conoce al servicio no facturado por la compañía que presta el servicio de telefonía pública.

- Al incorporarse el control de contadores, se debe incrementar la eficiencia de la red de teléfonos públicos al momento de tener un mantenimiento dirigido, puesto que el sistema reportará aquellos que tengan posible daño, al no detectarse llamadas en periodos de más de un día.

Ejemplos:

Una estimación conservadora en el ahorro por robo de llamadas es considerando 1 aparato al mes (costo por robo de LD por minuto) (alcancías en promedio con robo de LD) (minutos robados al mes).

N$ 0.5*1*(8*60*30)= N$ 7,200.00 al mes ($0.5 como costo de robo de llamada, en 1 caseta, por 8 horas en 30 días).

En la ciudad de Guadalajara:
N$ 0.5*10*(8*60*15)= N$ 36,000.00 al mes ($.05 como costo de robo de llamada, en 1 caseta, por 8 horas en 30 días).

Una estimación conservadora para determinar el incremento en la utilidad se considera solo un 5% de incremento en la eficiencia; esto es (rendimiento al día por alcancías) (número de alcancías) (Días por mes).

Para una ciudad pequeña como Hermosillo:
El 5% de 800 aparatos, en un mes: 8*45*30= N$ 10,800 al mes ($8.00 de rendimiento por día, 5% de incremento en eficiencia en 900 aparatos, en 30 días).

Para una ciudad mediana como Guadalajara:
8*250*30= N$ 60,000 al mes ($8.00 de rendimiento por día, 5% de incremento en eficiencia de 5000 aparatos, en 30 días).

Finalmente dentro de los beneficios no tangibles obtenidos, se encuentra el mejoramiento de la imagen de la compañía que presta estos servicios, puesto

que habrá menos probabilidad de encontrar un teléfono público descompuesto.

Incorporar el sistema no quiere decir que sea la solución de todos los males, sino mas bien será una herramienta para el incremento en la productividad de teléfonos públicos y un mejoramiento significativo en la calidad del servicio.

Innovación.

Aparentemente se podría pensar que sistemas tan complejos como lo son las centrales telefónicas, deberían tener un tipo de control de esta índole, pero la realidad es que el desarrollo de las centrales telefónicas esta enfocado directamente con el servicio que presta en comunicaciones, restando importancia a controles de bases de datos, o bien si existen estos tipos de desarrollo, son sumamente costosos.

El solo hecho de incorporar un sistema de procesamiento externo a la central, añade funcionalidad a toda una planta telefónica.

En el mercado existen un sistema para supervisar el funcionamiento de teléfonos públicos desarrollado en hardware, que se encarga mediante un procesador conectado junto al distribuidor de la central, en tomar líneas telefónicas, distribuir casetas telefónicas con tecnología especial, incluyendo modems administrar, controlar y detectar anomalías mediante llamadas telefónicas o pulsos eléctricos. Sin embargo cabe mencionar que esta solución es sumamente costosa, y de alguna manera depende de las líneas asignadas para estos teléfonos públicos por lo que cae en el mismo problema

de toda línea telefónica y es la dependencia de las categorías asignadas y del correcto funcionamiento de éstas.

Concluyendo, al incorporar un sistema de esta índole, llena el hueco para la supervisión, control, mantenimiento y prevención de fallas de teléfonos públicos y su línea telefónica asignada.

Compatibilidad.

Al manejar un compilador estándar como es el lenguaje "C", se puede fácilmente compilar en otros compiladores de otras marcas por lo que amplía el número de paquetes con el cual se puede interconectar el sistema.

Y al elegir el CODEBASE permite una gran versatilidad y compatibilidad con las bases de datos mas comunes en computadoras personales como lo son DBASE, FOXPRO, CLIPPER, etc. Esta ventaja permite transformar y adaptar bases de datos ya existentes sin la necesidad de recapturarlas.

Alternativas de Solución.

Solución 1.

La solución mas exhaustiva, sería el crear un programa de comunicaciones propio, escogiendo la herramienta más rápida para desarrollar, un ejemplo podría ser la herramienta que produce excelentes presentaciones en WINDOWS como el VISUAL BASIC, con este tipo de solución además se tendría resuelto el manejo de bases de datos, puesto que se podría ligar a ACCESS de Miscrosoft.

Observación: Partir de cero para desarrollar un programa de comunicaciones sería como intentar encontrar el "hilo negro", además que sería una pérdida muy considerable en el tiempo debido a que ya han habido personas dedicadas al desarrollo específico de comunicaciones incluso permitiendo utilizar el código de ellos; además en la actualidad se cuenta con una amplia gama de software de este tipo .

Solución 2.

Como se podrá pensar otra alternativa de solución al problema planteado puede ser un programa que se encargue de generar un archivo de lotes con las modificaciones o consultas que se quieran hacer a la central, y así usar cualquier programa de comunicaciones y cualquier herramienta de desarrollo, esta alternativa puede ser en cualquier plataforma: DOS, WINDOWS, UNIX, OS/2, etc.

Observación: Esta alternativa puede ser la mas rápida de implementar, pero el hecho de simplificar el problema para el desarrollador del sistema, complica la operación de gran manera del usuario final, además que quita la flexibilidad de usar el sistema como consola de operación.

Solución 3.

Esta alternativa consiste en utilizar un programa de comunicaciones que se puedan contar con los fuentes usando una plataforma amigable como el Windows, y de ahí incorporar cualquier manejador de bases de datos.

Observación:

Efectivamente esta es la opción mas bonita y elegante, pero la más engorrosa al momento de programar o modificar programas fuentes de este tipo, debido a la filosofía que maneja Windows llamada Controlador-Vista-Modelo semejante al Smaltalk 80, que facilita al usuario el uso, pero a nivel de programación es mas compleja. El hacer cualquier adaptación a partir de cualquier programa de comunicaciones en Windows, representa grandes esfuerzos y tiempo.

Solución 4.

Esta alternativa consiste de partir de un programa de comunicaciones robusto, desarrollado en plataforma DOS que cuente con los programas fuentes para irlo adaptando.

Observación: Esta alternativa puede ser la mejor solución, puesto que nos permite modificar y adaptar el sistema de manera simple y también el usuario tendrá una interface amigable. Además se puede incorporar cualquier herramienta de apoyo en el compilador que haya sido generado. (ver figuras 2.6-4 y 2.6-5)

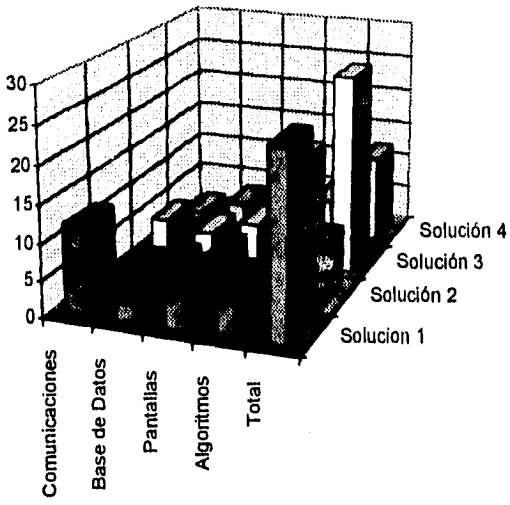


Fig. 2.6-4 Tiempo de Desarrollo.

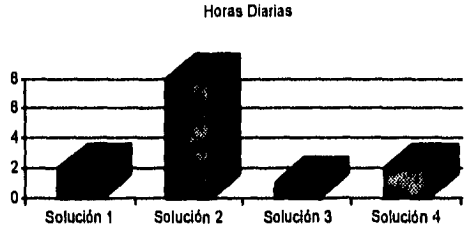


Fig. 2.6-5 Operación Necesaria del Sistema por Día.

Conclusiones.

- Para un programa de comunicaciones de este tipo, se requiere utilizar código reusable, porque de esta manera se tiene un gran avance en el proyecto, por lo que el sistema se reduce al estudio de protocolos, manejo de bases de datos y presentación de pantallas.

- Para las exigencias en tiempos de entrega para cualquier desarrollo, la mejor opción es encontrar un nivel en el cual el usuario quede satisfecho a un corto plazo, por lo cual se debe contar con la mejor opción en plataforma de desarrollo y herramientas de estas.

- Las tendencias actuales en nivel de desarrollo es el código reusable

- El programa para monitorear teléfonos públicos es definitivamente rentable por el hecho no de ingresos, sin ahorro en pérdidas por fraude en líneas telefónicas.

- Otro motivo por lo que el proyecto es rentable es por el mantenimiento dirigido en que ayuda el sistema.

CAPITULO III

DISEÑO Y DESARROLLO DEL SISTEMA

3.1 Módulo Principal.

El sistema se realizará en lenguaje C y se utilizara CodeBase para el manejo de bases de datos y se apoyará para el enlace con las centrales el programa de comunicación Simple. La elección de estó fue dado por el análisis realizado en el capítulo anterior.

El sistema contará con un menú principal del cual se desprenderán cuatro módulos principales los cuales son menú de archivos, menú de procesos, menú de reportes y menú de configuración, lo anterior se representa gráficamente en la figura 3.1. - 1

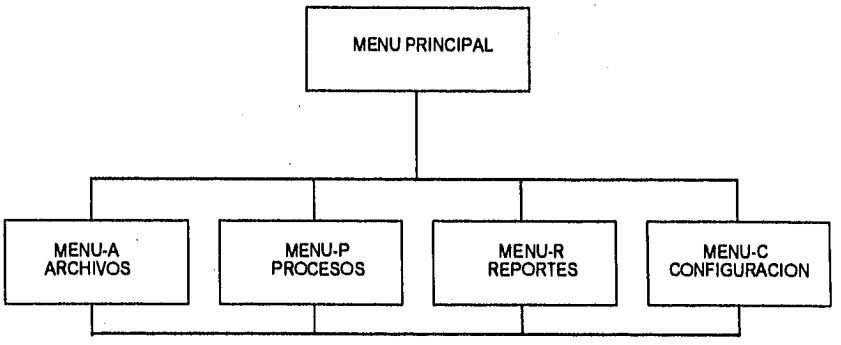


Fig. 3.1. - 1 Diagrama de Bloques. Módulos Principales del Sistema.

El modulo de archivos cuenta con menús de altas, bajas y cambios de las diferentes bases de datos que maneja el sistema. La figura 3.1. - 2 muestra de forma esquemática como esta conformado el modulo de archivos.

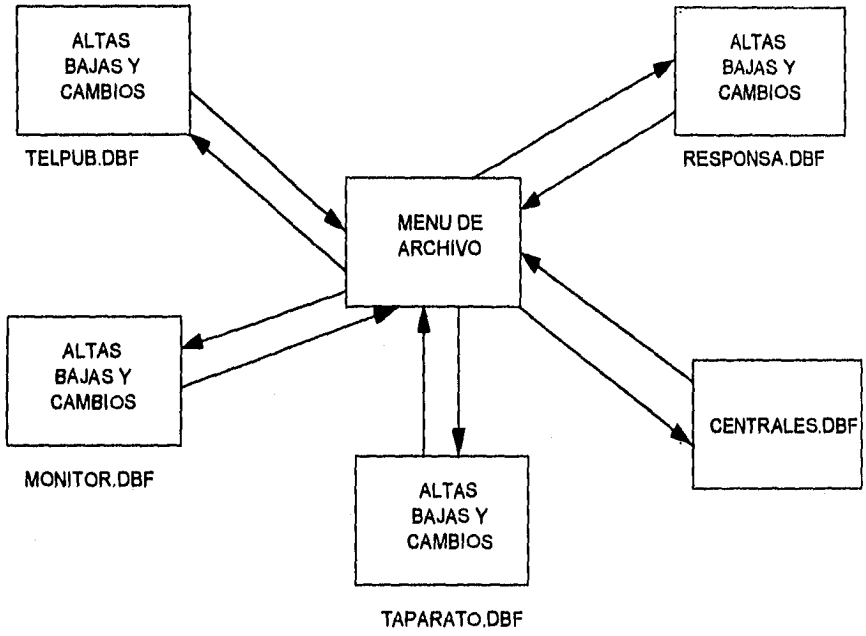


Fig. 3.1.-2 Diagrama de Bloques. Módulo de Archivos.

El modulo de procesos contará con un monitoreo masivo, para la revisión de categorías, revisión de contadores y una revisión continua las 24 hrs.; así mismo se podrá modificar las categorías de las alcancías, lo cual se muestra en la figura 3.1. - 3.

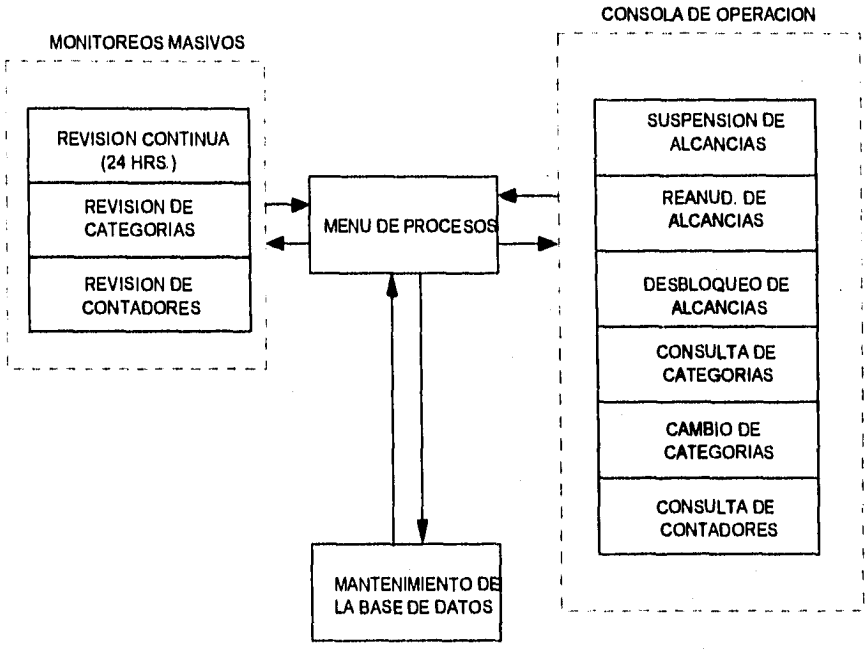


Fig. 3.1. - 3 Diagrama de Bloques. Módulo de Proceso.

A su vez el modulo de revisión continua, contará con el modulo de comunicación y con un proceso de reconocimiento de respuestas, lo cual se observa en la figura 3.1. - 4.

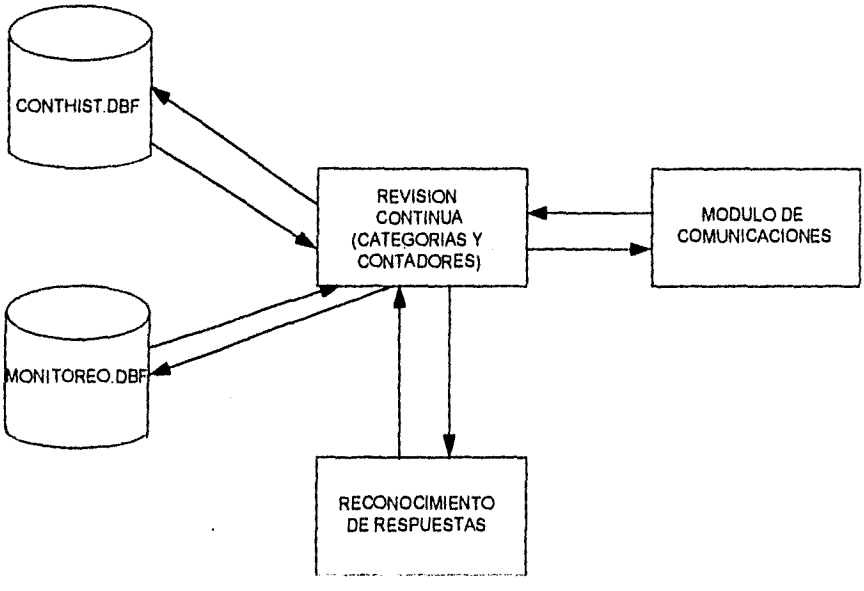


Fig. 3.1. - 4 Diagrama de Bloques. Módulo de Revisión Continua.

El modulo de reportes contará con reportes de inventario, reporte de totales de aparatos con posibles fallas y resúmenes. En la figura 3.1. - 5 se muestra los diferentes módulos de reportes.

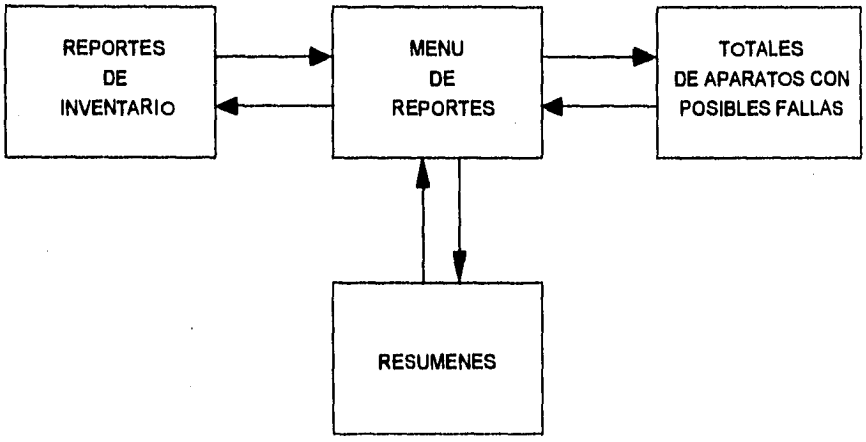


Fig. 3.1. - 5 Diagrama de Bloques del Módulo de Reportes.

El modulo de configuración contará con mantenimiento del equipo, configuración de funciones, configuración de series validas y configuración para la comunicación por modem y configuración de series validas, gráficamente se observa en la figura 3.1. - 6.

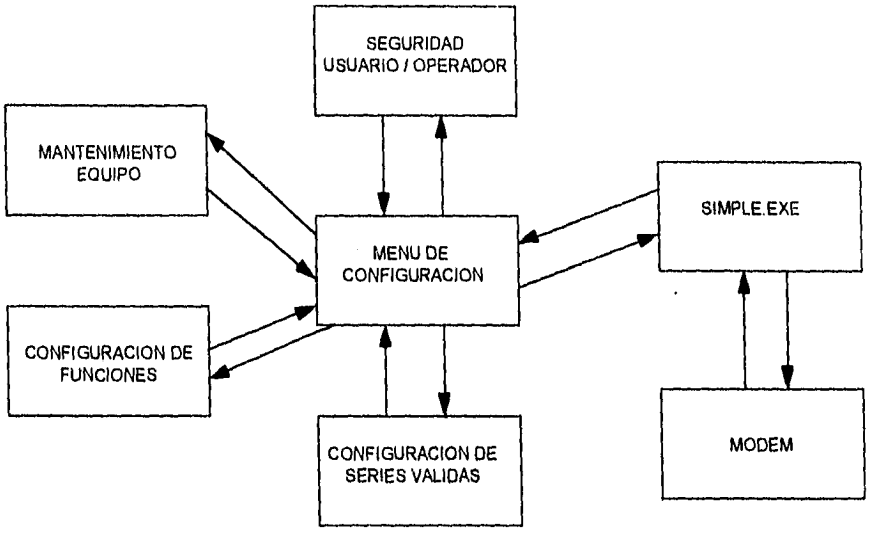


Fig. 3.1. - 6. Diagrama de Bloques. Módulo de Configuración.

3.2. Diagrama General de Flujo.

En la línea de comandos se puede introducir dos parámetros, el tipo de monitor y el password. Este módulo comienza con la validación de los parárnetros de entrada si es que estos existen, después se establece el modo de video preestablecido si es que no se introdujo como parámetro de entrada. Se checa el password si es que se introdujo como parámetro, si no fue así se pasa a una rutina que pide el password correcto antes que el sistema finalice.

Una vez dentro del sistema se muestra un menú donde básicamente se llamará al módulo al que pertenezca dicha opción. Estos módulos son: menu - a, menu - p, menu - r y menu - c.

Este módulo consta de los siguientes archivos:

alcanx.c
datos.c
io.c
codrut.c
gralfunc.c
gralfunx.c
tplcscape.lib
tplowl.lib
tserial.lib.

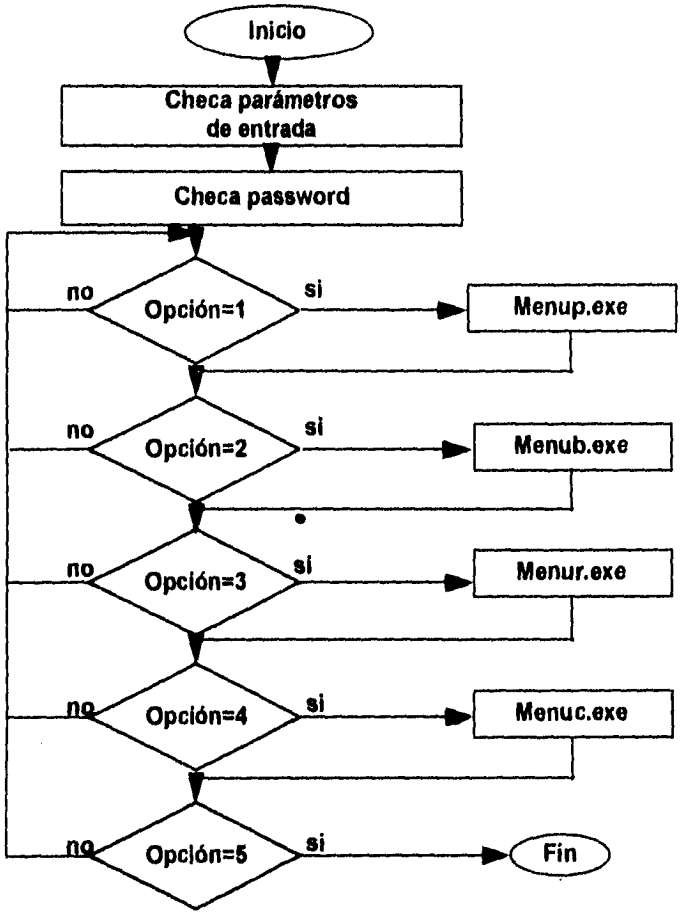


Fig. 3.2. - 1 Diagrama de Flujo del Módulo Principal.

3.3. Diagrama de Flujo de Datos.

Procesos que Realiza.

El sistema mediante la opción de procesos supervisa:

a) Prueba de líneas telefónicas, con la intención de detectar cuales teléfonos han generado llamadas y cuales no.

b) El estado de los contadores de abonado asociados a las revisiones de categorías que se encarga de ajustar los atributos asociados al teléfono, con la información que tiene declarada en la base de datos, por ejemplo; restricción para tráfico, llamadas larga distancia, supervisión, inversión de polaridad, etc. En caso de detectar diferencias en la información declarada en la central en el momento de la lectura, el sistema automáticamente corregirá las categorías.
Esta función es muy útil pero exige mantener bien actualizada la base de datos para evitar posibles errores. Por ejemplo; suspender o reanudar servicio en caso de errores, restringir tráfico.

c) El sistema también cuenta con una prueba de la línea de abonado asociada al teléfono público, para aquellos casos en los que se detectó que el teléfono no ha sido utilizado, lo cual puede ayudar a dirigir las labores de mantenimiento, si se detectan problemas en la red.

Procesos Adicionales.

El sistema cuenta a su vez con una serie de procesos adicionales que sirven para dar mantenimiento a los teléfonos públicos de una manera más fácil que utilizando una terminal de MOA:

a) Suspensión de Alcancías: El cual permite modificar las categorías necesarias a la central telefónica para poder realizar una suspensión de línea, actualizando la nueva condición en caso de estar dada de alta en la base de datos.

b) Reanudación de Alcancías: El cual permite modificar las categorías necesarias a la central telefónica para poder realizar una reanudación de línea, actualizando la nueva condición en caso de estar dada de alta en la base de datos.

c) Consulta de categorías: El cual permite consultar las categorías asignadas a un teléfono público en la central telefónica.

d) Cambio de categorías: El cual permite cambiar una serie de categorías preestablecidas para un teléfono público enviando una serie de comandos adecuados para que la central tome en cuenta los cambios.

e) Desbloqueo de línea: El cual permite desbloquear un número telefónico que la central haya puesto fuera de servicio por una falla en la línea, enviando el comando necesario.

f) Prueba de línea de ALCANCIAS: La cual permite verificar el funcionamiento de la línea, detectando posibles fallas en la línea telefónica.

La mayoría de las características más importantes se pueden declarar o restringir dentro del Menú de opciones.

Asimismo, el hecho de declarar los datos asociados al teléfono como son la dirección de ubicación y razón social, permite que los reportes de supervisión incluyan estos datos facilitando los programas de recorridos.

Alcancías proporciona diferentes índices de ordenamiento para clasificar por zonas los diferentes teléfonos, de tal manera que los reportes de mantenimiento pueden ser agrupados en función del personal que deba atenderlos.

Por último cabe mencionar que si se desea obtener la verdadera potencialidad del sistema en lo que se refiere a la prueba de líneas de abonado, se debe asegurar seleccionar aparatos telefónicos que pueden ser "vistos" por la central EXA, o en su defecto, instalar un desconectador "DIT" lo cual asegura la confiabilidad de la línea telefónica.

En la figura 3.3. - 1 se representa el Diagrama de Flujo de Datos del sistema.

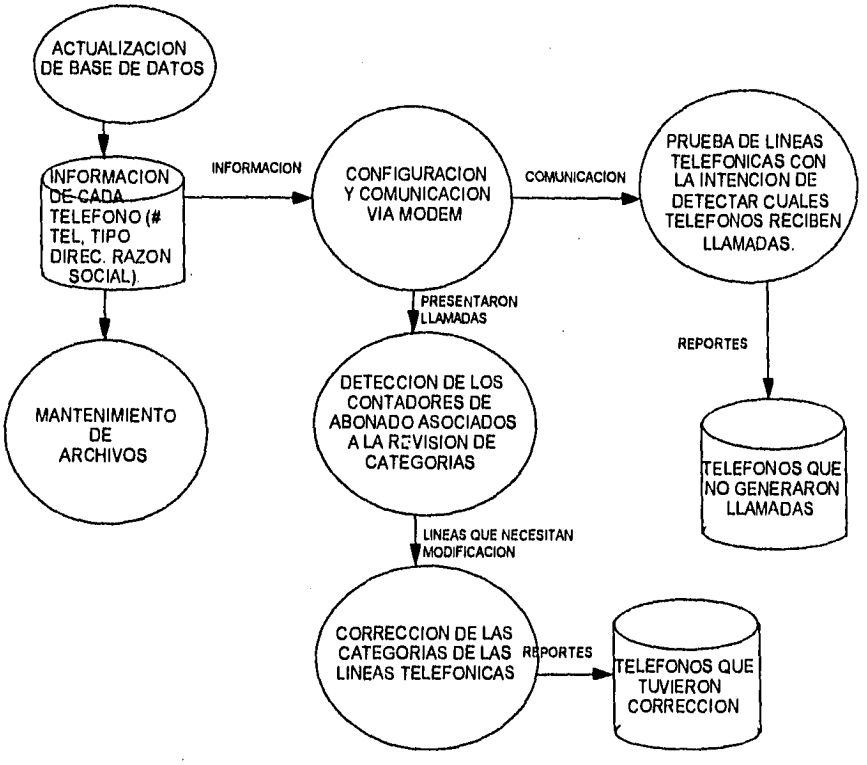


Fig. 3.3. - 1 Diagráma de Flujo de Datos.

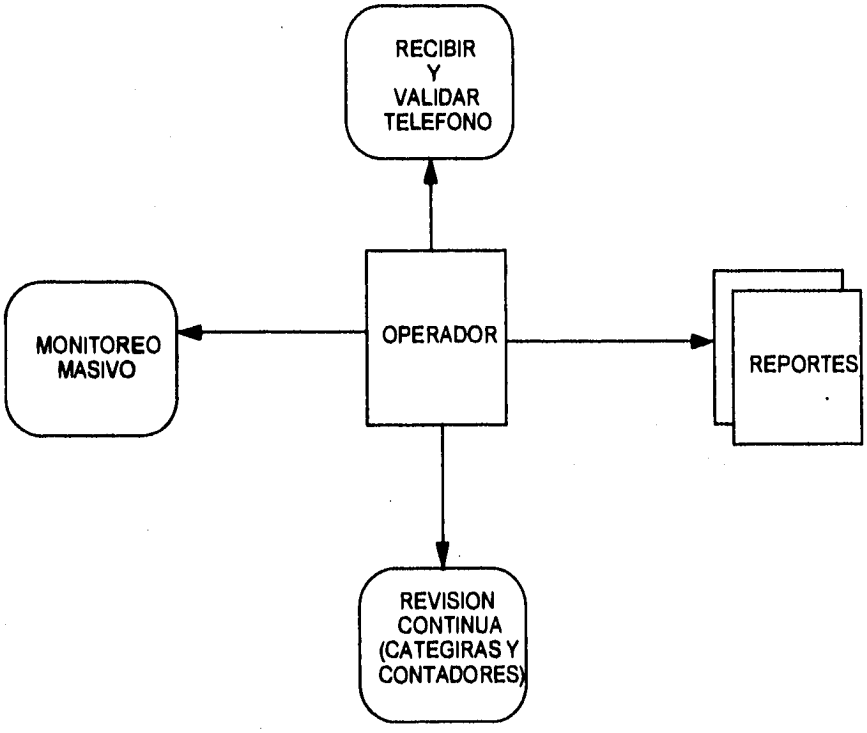


Fig. 3.3 - 2 Diagrama de Flujo de Datos Principal.

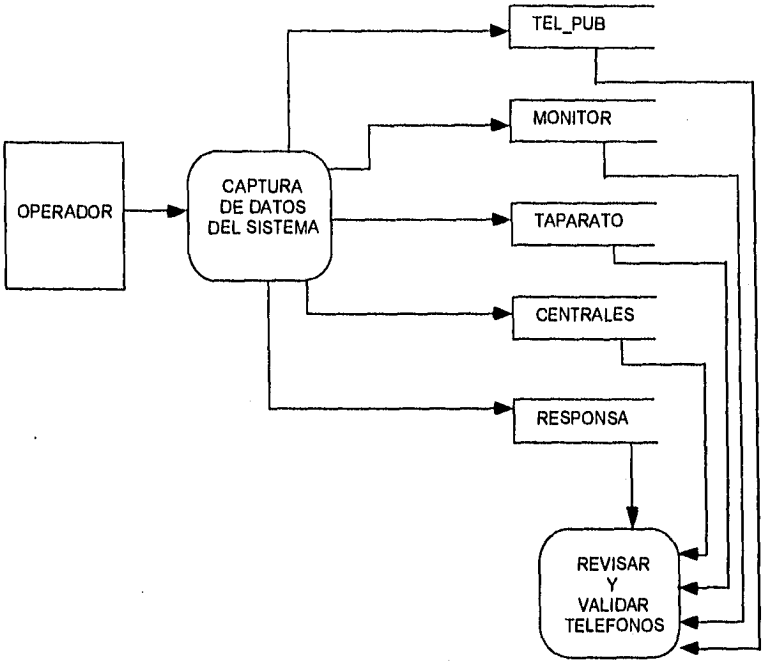


Fig. 3.3. - 3 Diagrama de Flujo de Datos. Captura de Datos del Sistema.

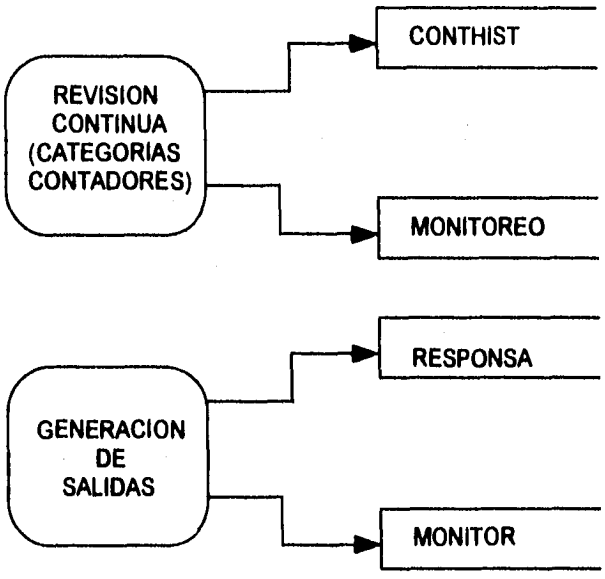


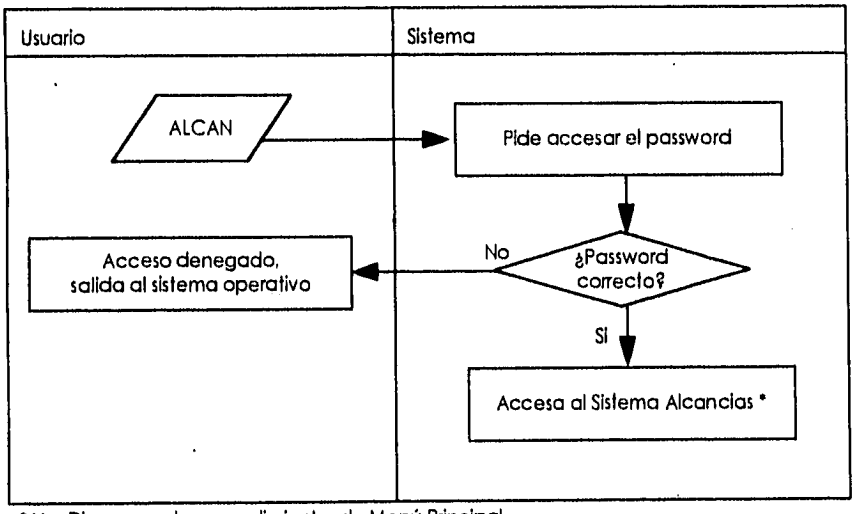
Fig. 3.3 - 4 Diagrama de Flujo de Datos. Revisión Continua de Categorías y Contadores. Generación de Salidas.

3.4. Diagramas de Procedimientos.

Se presentarán los diagramas de procedimientos del sistema que nos presentan claramente la interacción del sistema con el usuario, bases de datos y la central telefónica.

Diagrama de Procedimientos de Acceso al Sistema "Alcancías".

El usuario digitará la palabra "ALCAN" para activar el sistema "Alcancías" y accesará su password correspondiente para navegar por el sistema. Figura 3.4. - 1.

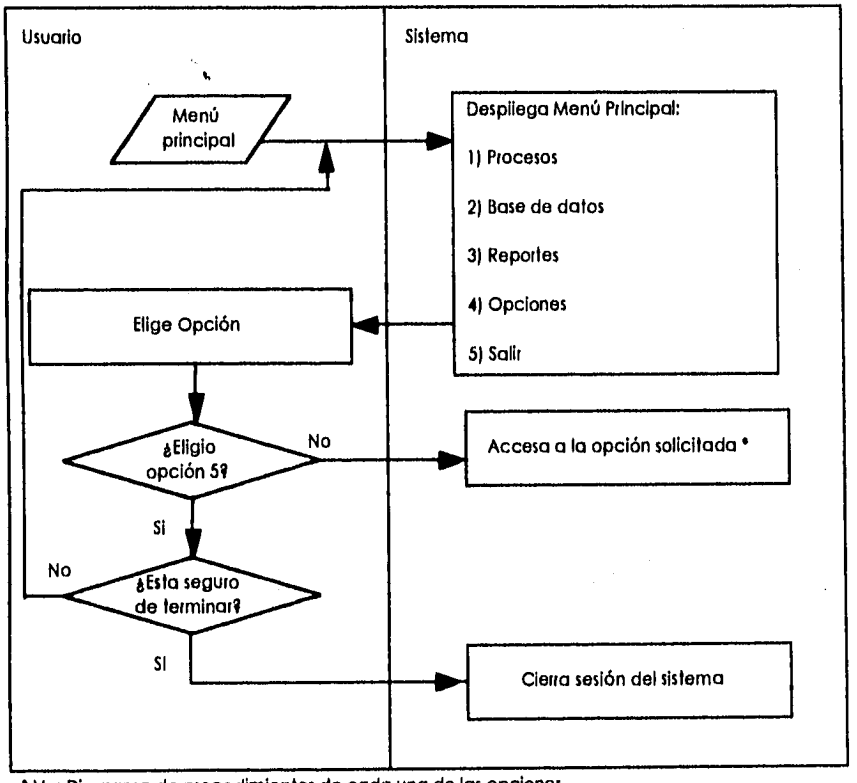


* Ver Diagrama de procedimientos de Menú Principal

Fig. 3.4. - 1 Diagrama de Procedimientos de Acceso al Sistema "Alcancias".

Diagrama de Procedimientos del Menú Principal.

El usuario visualizará el menú principal y elegirá la opción que desee ejecutar en caso contrario puede terminar la sesión del sistema "Alcancías". Figura 3.4. - 2.



* Ver Diagrama de procedimientos de cada una de las opciones.

Fig. 3.4. - 2 Diagrama de Procedimientos del Menú Principal.

Diagrama de Procedimientos del Menú Procesos.

El usuario visualizará el menú de procesos y elegirá la opción que desee ejecutar en caso contrario oprime la tecla escape para regresar al menú principal. Figura 3.4. - 3.

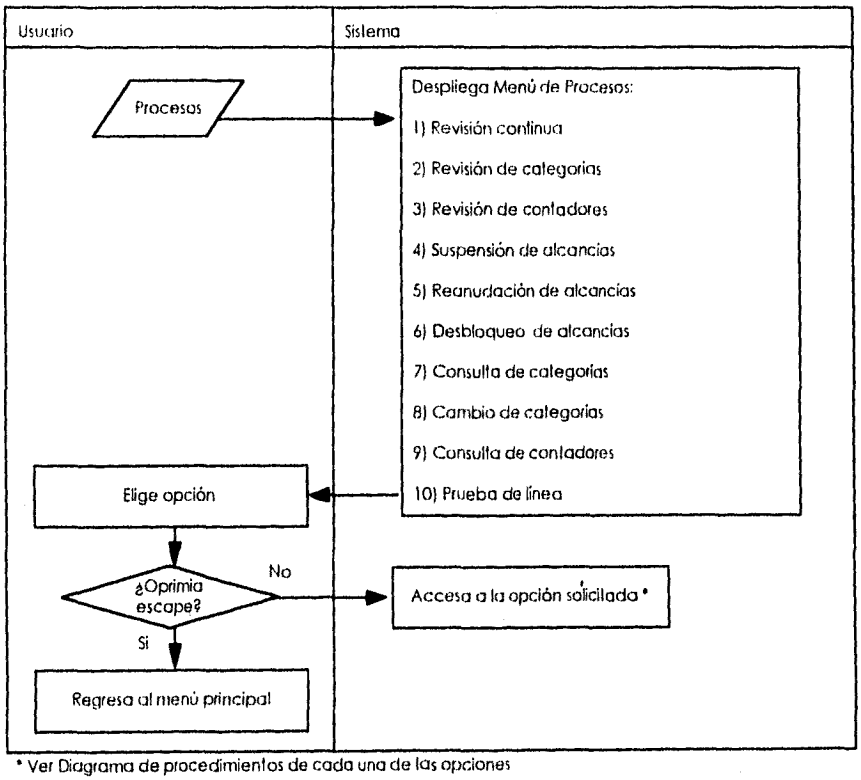
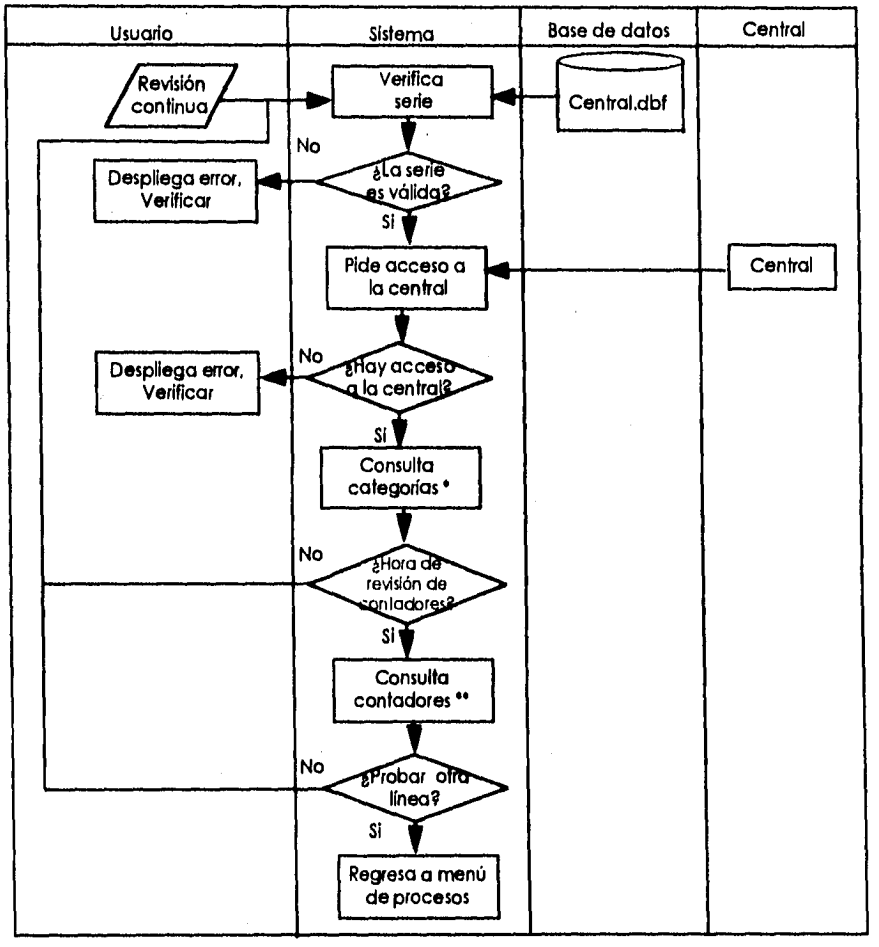


Fig. 3.4. - 3 Diagrama de Procedimientos del Menú de Procesos.

Diagrama de Procedimientos de Revisión Continua.

La revisión continua consiste de verificar el estado de las líneas válidas, si hay acceso a la central , se revisan las categorías así como los contadores.

Figura 3.4. - 4.



*Ver Diagrama de Revisión de Categorías.
** Ver Diagrama de Revisión de Contadores.

Fig. 3.4. - 4 Diagrama de Procedimientos de Revisión Continua.

Diagrama de Procedimientos de Revisión de Categorías.

La revisión de categorías de la línea consiste en accesar a la central , si lo hay, se revisan las categorías. Figura 3.4. - 5.

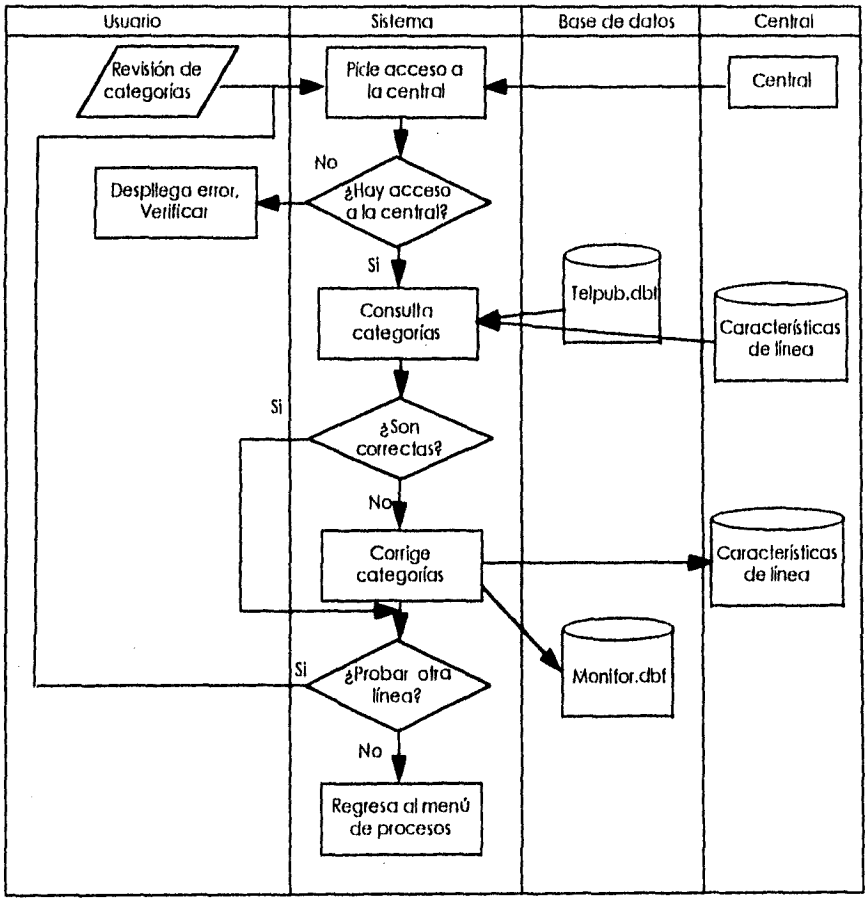


Fig. 3.4. - 5 Diagrama de Procedimientos de Revisión Continua.

Diagrama de Procedimientos de Revisión de Contadores.

La revisión de contadores de la línea consiste en accesar a la central, si lo hay, se revisan las contadores. Figura 3.4. - 6.

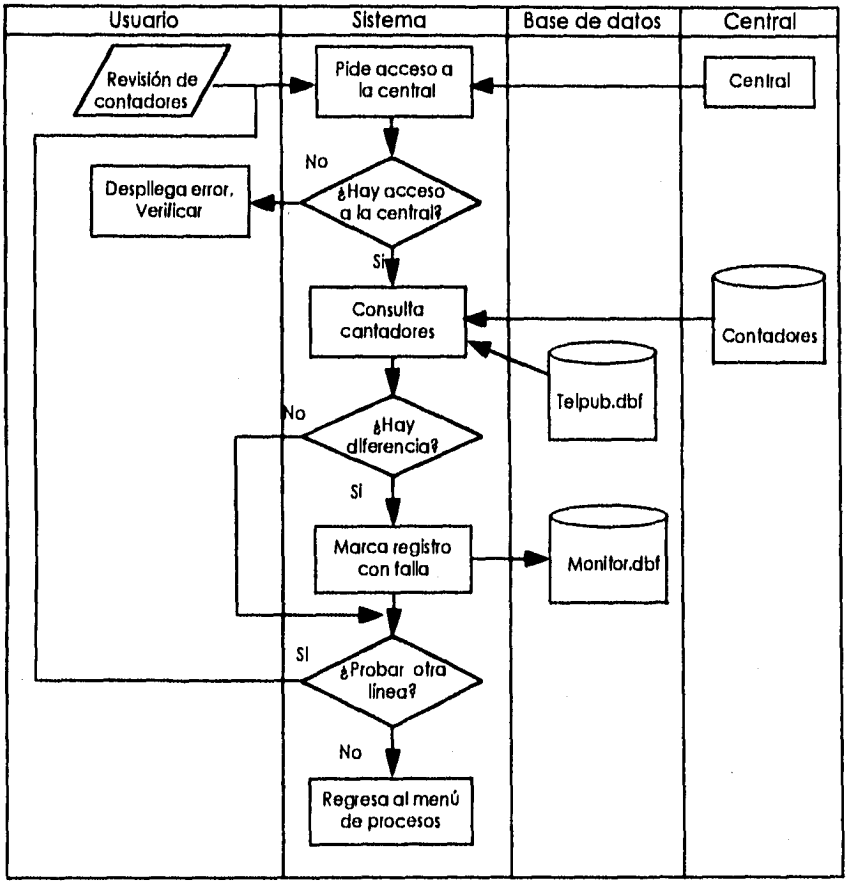


Fig. 3.4. - 6 Diagrama de Procedimientos de Revisión de Contadores.

Diagrama de Procedimientos de Suspensión de Alcancías.

La suspensión del servicio de alcancías se realiza accesando a la central , si lo hay, se suspende la alcancía. Figura 3.4. - 7.

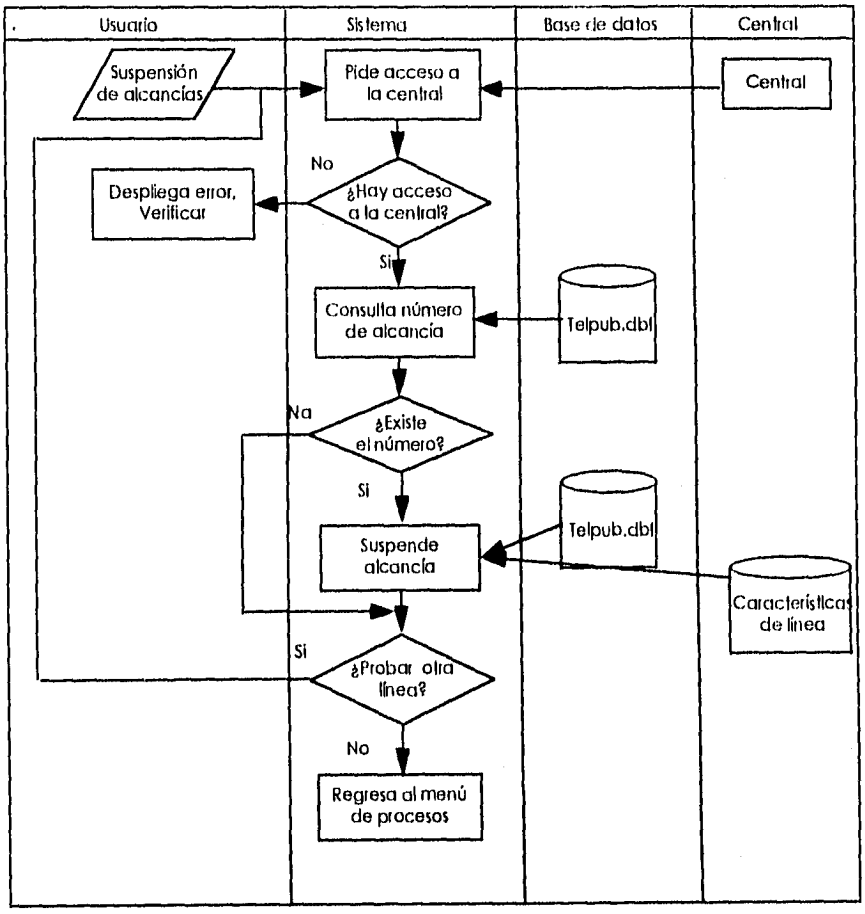


Fig. 3.4. - 7 Diagrama de Procedimientos de Suspensión de Alcancías.

Diagrama de Procedimientos de Reanudación de Alcancías.

La reanudación de servicio de alcancías se realiza si hay acceso a la central , si lo hay, se reanuda la alcancía. Figura 3.4. - 8.

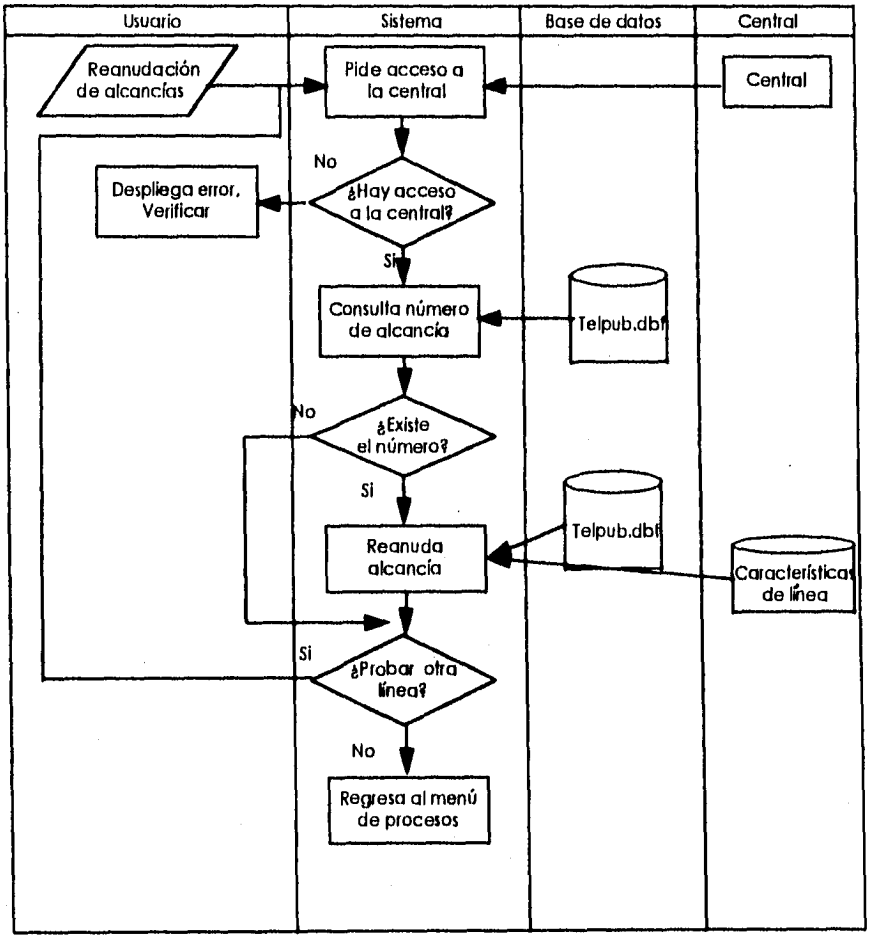


Fig. 3.4. -8 Diagrama de Procedimientos de Reanudación de Alcancías.

Diagrama de Procedimientos de Desbloqueo de Alcancías.

El desbloqueo de alcancías se realiza si hay acceso a la central , si lo hay y existe el número se realiza el proceso. Figura 3.4. - 9.

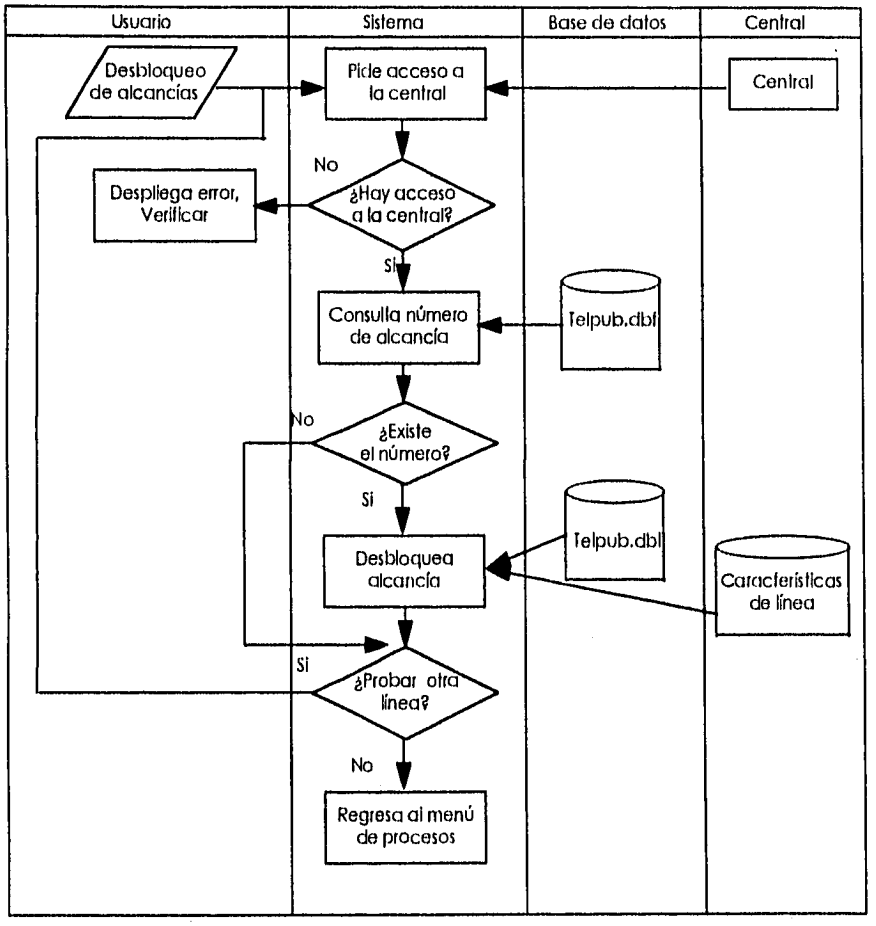


Fig. 3.4. - 9 Diagrama de Procedimientos de Desbloqueo de Alcancias.

Diagrama de Procedimientos de Consulta de Categorías.

La consulta de categorías de la línea se realiza accesando a la central, si lo hay, se realiza el proceso de consulta. Figura 3.4. - 10.

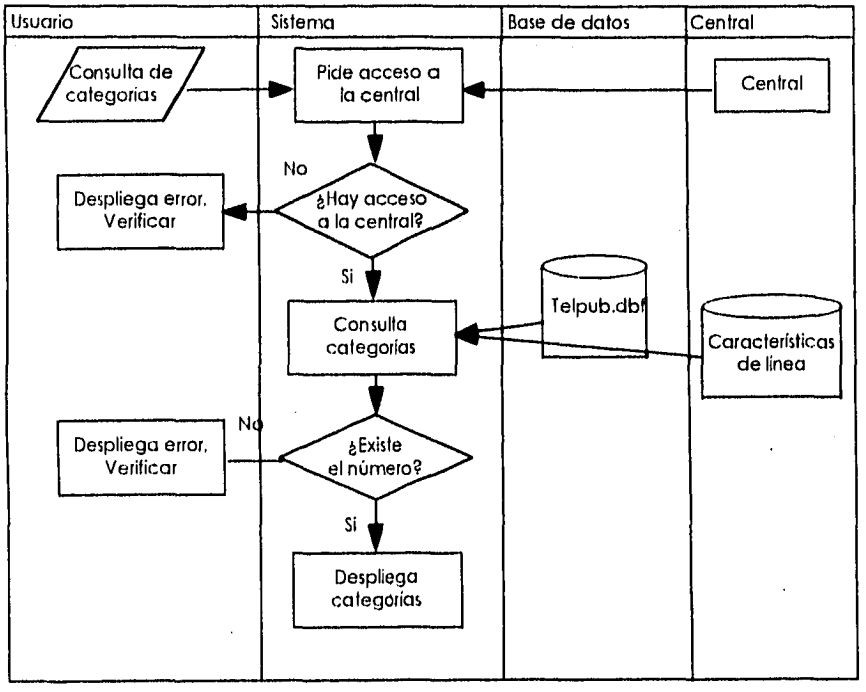


Fig. 3.4. - 10 Diagrama de Procedimientos de Consulta de Categorías.

Diagrama de Procedimientos de Cambio de Categorías.

El cambio de categorías de una alcancía se realiza accesando a la central, si lo hay y el número existe se realiza el proceso de cambio. Figúra 3.4. - 11.

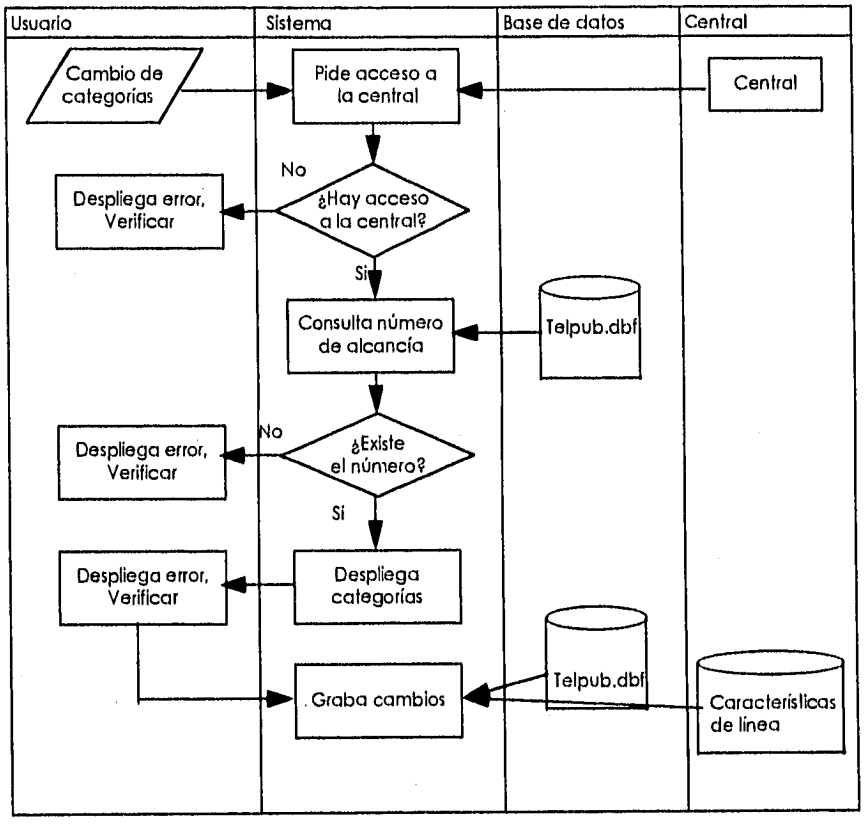


Fig. 3.4. - 11 Diagrama de Procedimientos de Cambio de Categorías.

Diagrama de procedimientos de Consulta de Contadores.

La consulta de contadores de la línea se realiza accesando a la central, si lo hay, se realiza el proceso de consulta. Figura 3.4. -12.

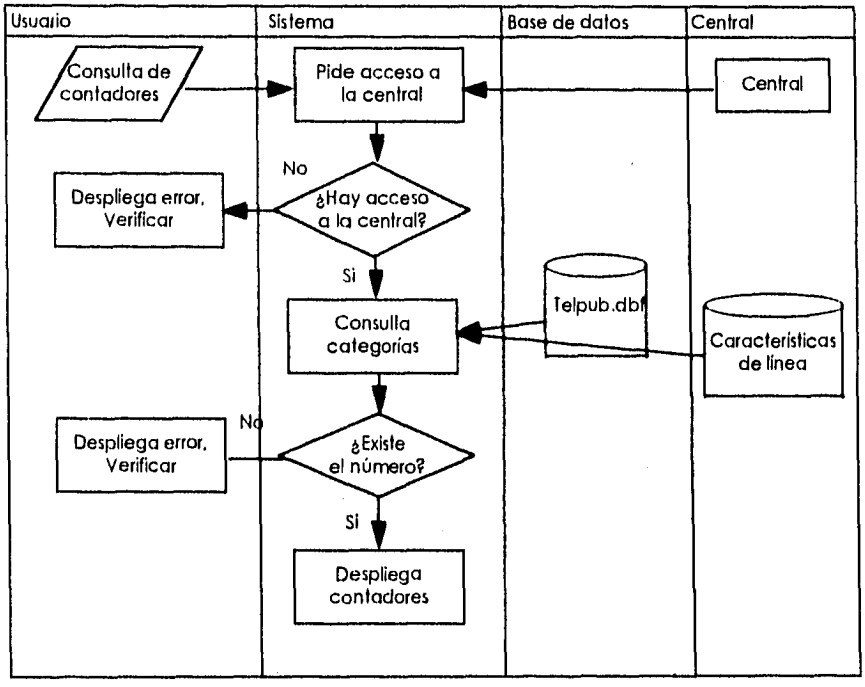


Fig. 3.4. -12 Diagrama de Procedimientos de Consulta de Contadores.

Diagrama de Prueba de Línea de Alcancías.

Esta prueba consiste de verificar el estado de la línea accesando a la central, se verifica que existe el número y se genera un estado de prueba. Figura 3.4. - 13.

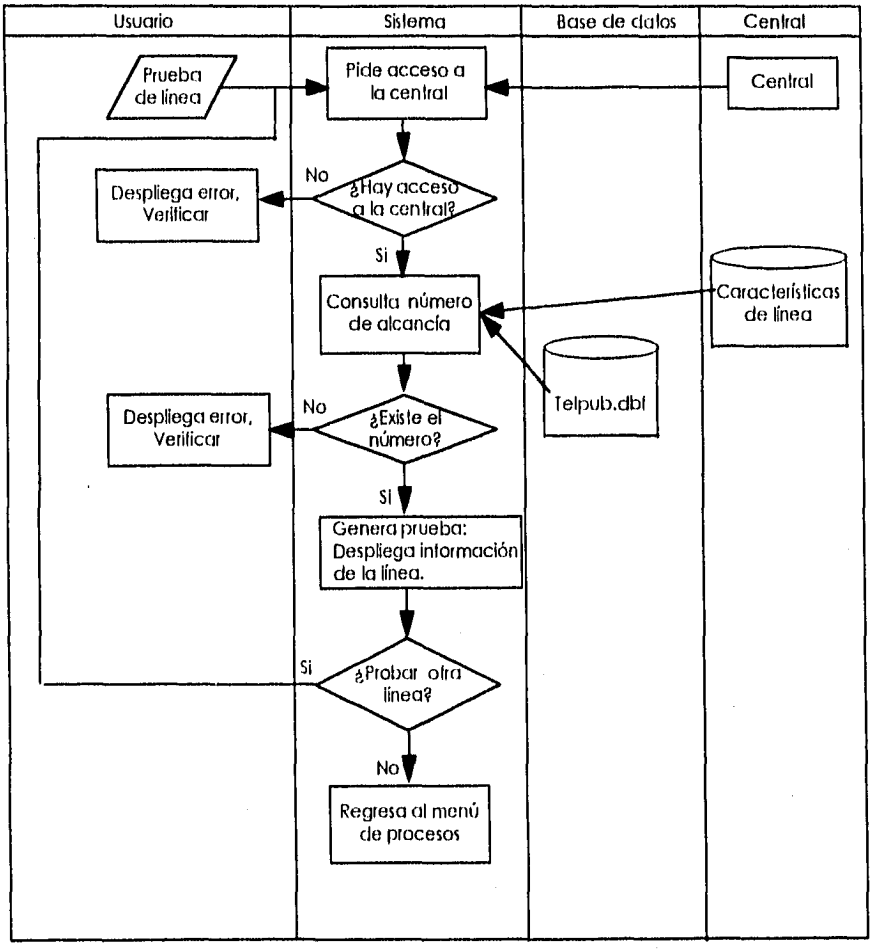
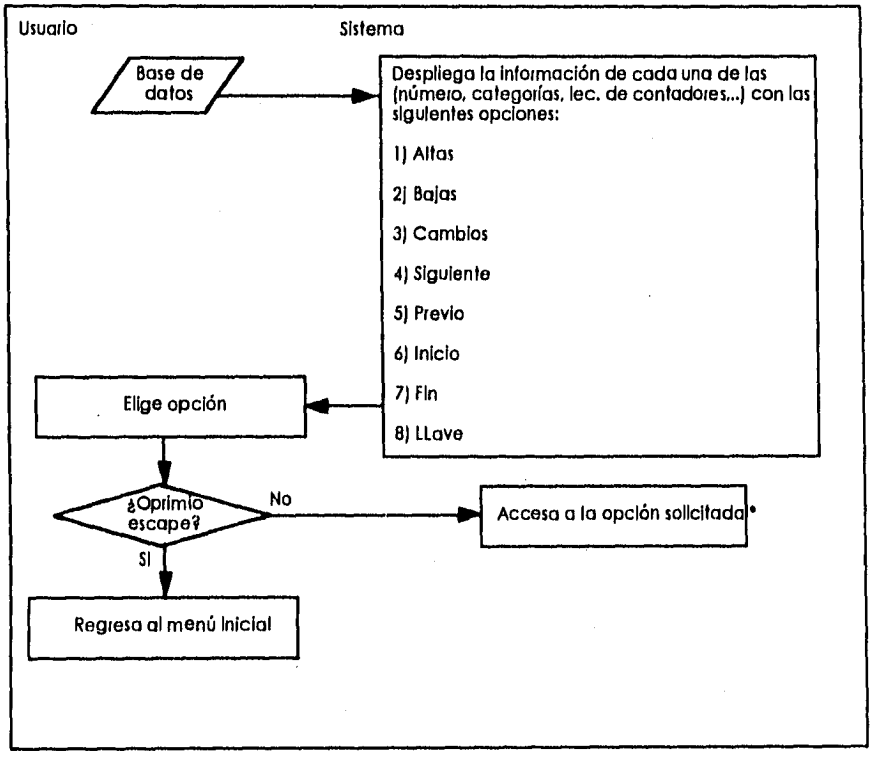


Fig. 3.4. - 13 Diagrama de Procedimientos de Prueba de Línea de Alcancías.

Diagrama de Procedimientos de Bases de Datos.

El usuario accesará a la base de datos y elegirá la opción que desee ejecutar en caso contrario oprime la tecla escape para regresar al menú principal. Figura 3.4. - 14.



* Ver Diagrama de procedimientos cada opción.

Fig. 3.4. - 14 Diagrama de Procedimientos de Bases de Datos.

Diagrama de Procedimientos de Altas a la Base de Datos.

El usuario accesará los datos de los campos requeridos y si son válidos serán grabados en la base de datos correspondiente en caso contrario despliega un error. Figura 3.4. - 15.

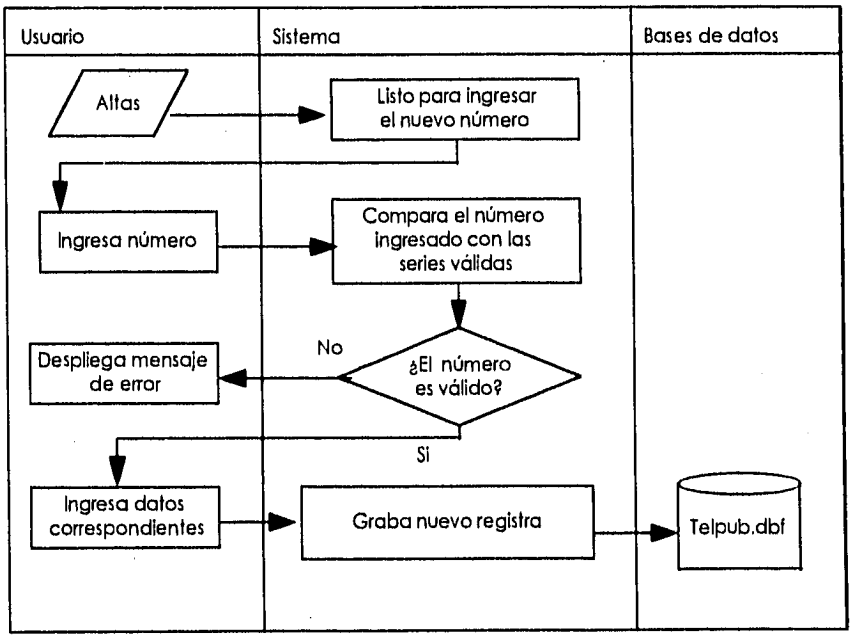


Fig. 3.4. - 15 Diagrama de Procedimientos de Alta a la Base de Datos.

Diagrama de Procedimientos de Bajas a la Base de Datos.

El usuario accesará el número de la alcancía a dar de baja si existe pedirá el documento que ampare este proceso, si es válido borrará el registro solicitado. Figura. 3.4.- 16.

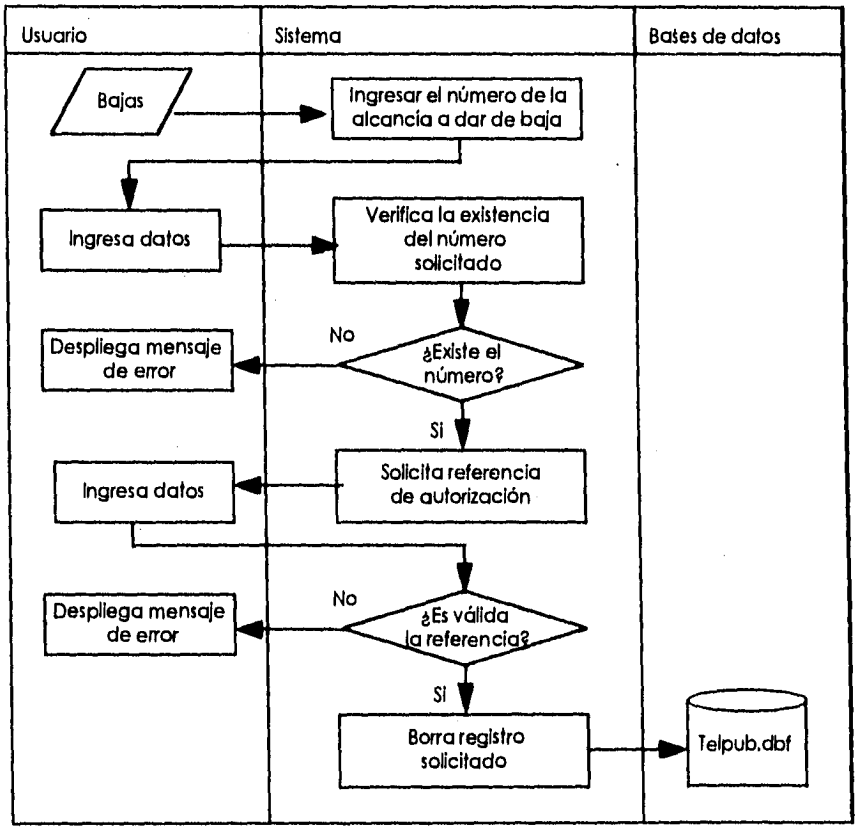


Fig. 3.4. - 16 Diagrama de Procedimientos de Bajas a la Base de Datos.

Diagrama de Procedimientos de Cambios a la Base de Datos.

El usuario accesará el número de la alcancía a ejecutar los cambios si existe el número al terminar de ingresar los datos pedirá la referencia del documento que ampare este proceso, si es válido grabará los cambios al registro solicitado. Figura 3.4. - 17.

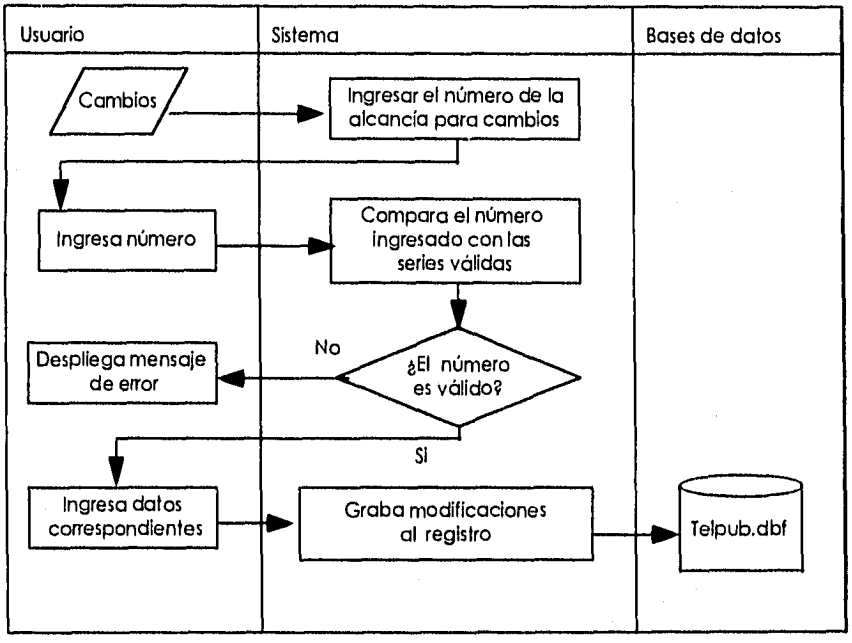


Fig. 3.4. - 17 Diagrama de Procedimientos de Cambios a la Base de Datos.

Diagrama de Procedimientos de Siguiente Registro.

Al elegir está opción se desplegará el siguiente registro de la base de datos, si no existe otro desplegará el mismo. Figura 3.4. - 18.

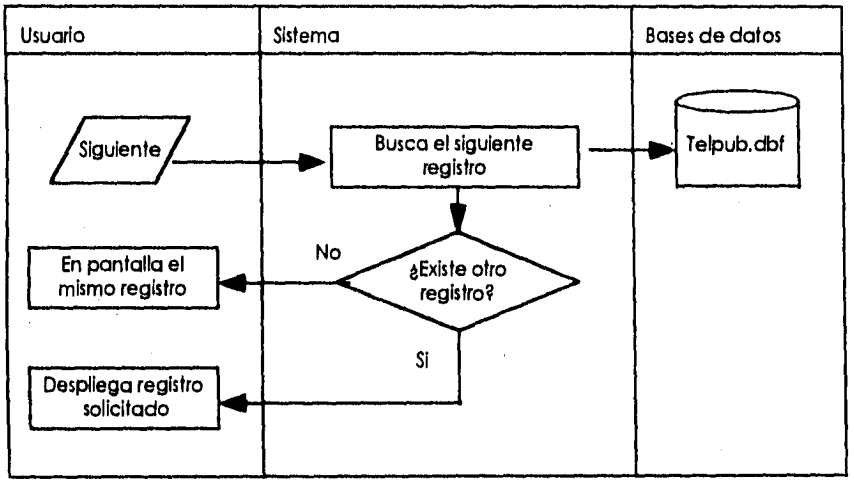


Fig. 3.4. - 18 Diagrama de Procedimientos de Siguiente Registro.

Diagrama de Procedimientos de Registro Previo.

Al elegir está opción se desplegará el registro anterior de la base de datos, si no existe otro desplegará el mismo. Figura 3.4. - 19.

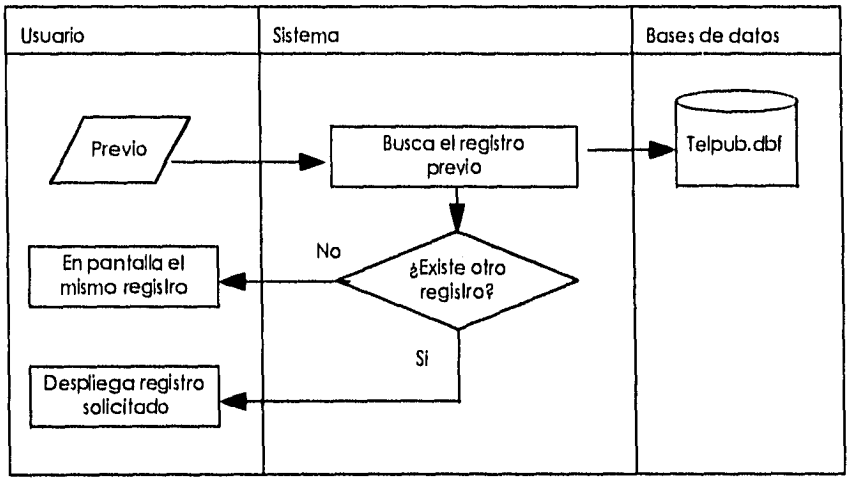


Fig. 3.4. - 19 Diagrama de Procedimientos de Registro Previo.

Diagrama de Procedimientos de Inicio de Base de Datos (Primer Registro).

Al elegir está opción se desplegará el primer registro de la base de datos, si no existe desplegará un mensaje de error. Figura 3.4. - 20.

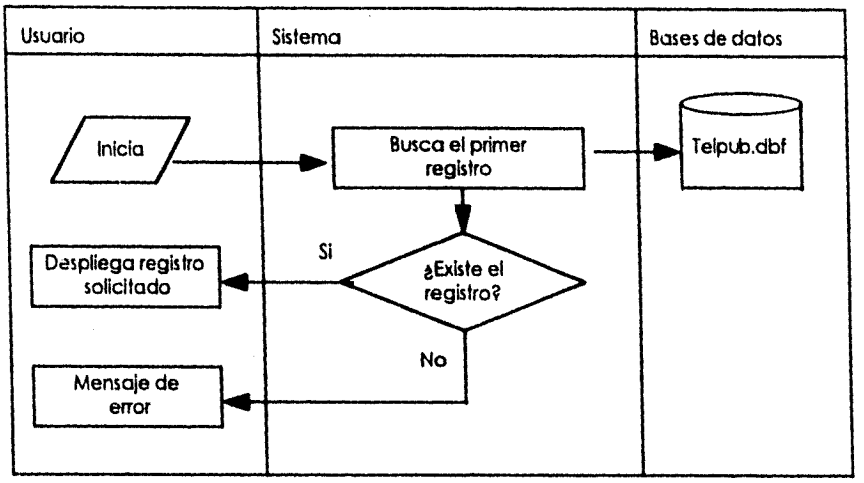


Fig. 3.4. - 20 Diagrama de Procedimientos de Inicio de Base de Datos (Primer Registro).

Diagrama de Procedimientos de Fin de Base de Datos (Ultimo Registro).

Al elegir está opción se desplegará el último registro de la base de datos, si no existe desplegará un mensaje de error. Figura 3.4. - 21.

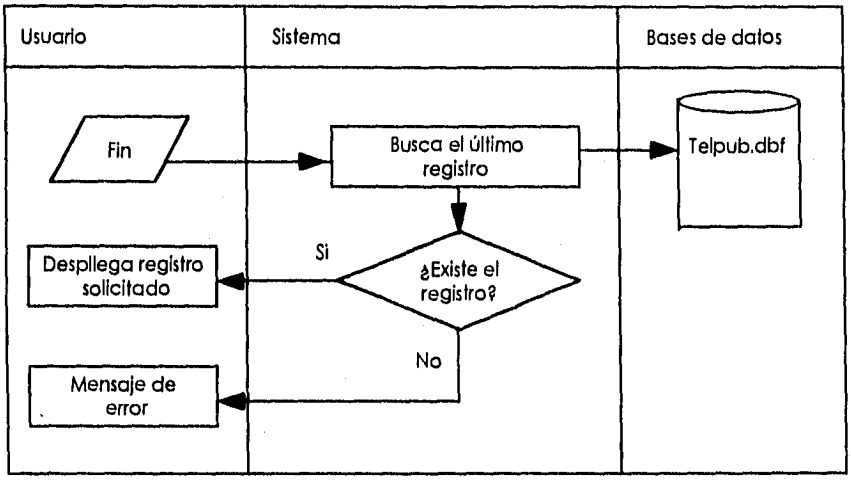


Fig. 3.4. - 21 Diagrama de Procedimientos de Fin de Base de Datos (Ultimo Registro).

Diagrama de Procedimientos de Llave de la Base de Datos.

Al elegir está opción se ingresará el número de la alcancía que servirá de llave a la base de datos, si no es válido desplegará un mensaje de error. Figura 3.4. - 22.

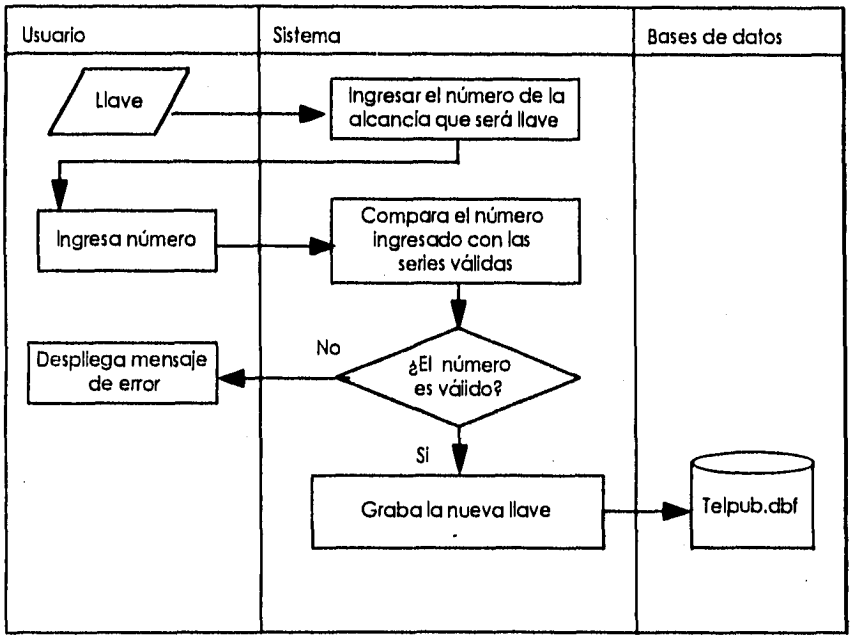
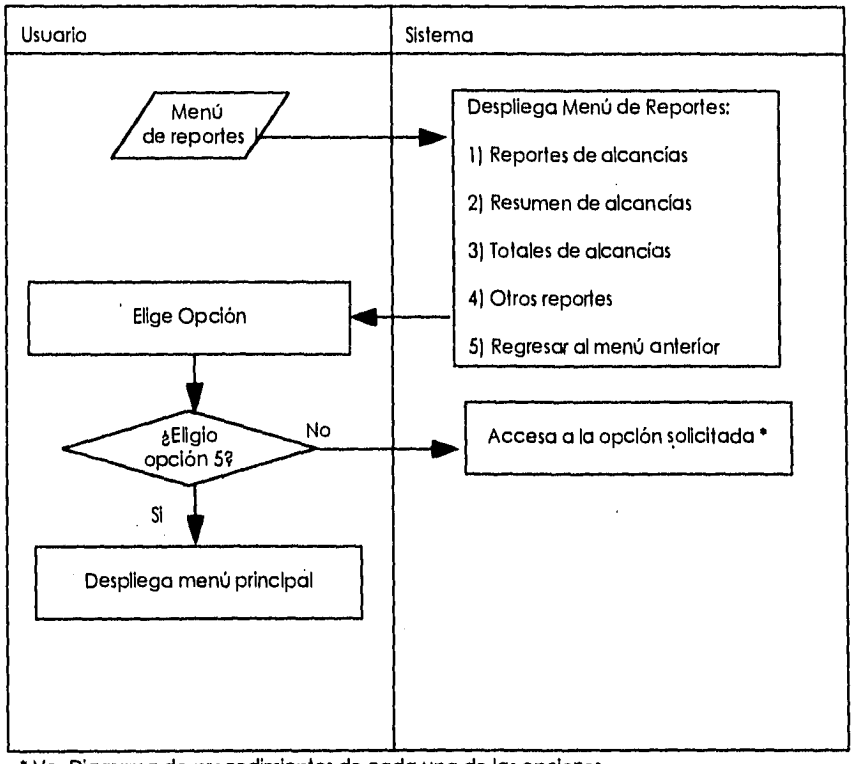


Fig. 3.4. - 22 Diagrama de Procedimientos de Llave de la Base de Datos.

Diagrama de Procedimientos del Menú de Reportes.

El usuario visualizará el menú de reportes y elegirá la opción que desee ejecutar, si es la opción 5 regresa al menú principal. Figura 3.4. - 23.

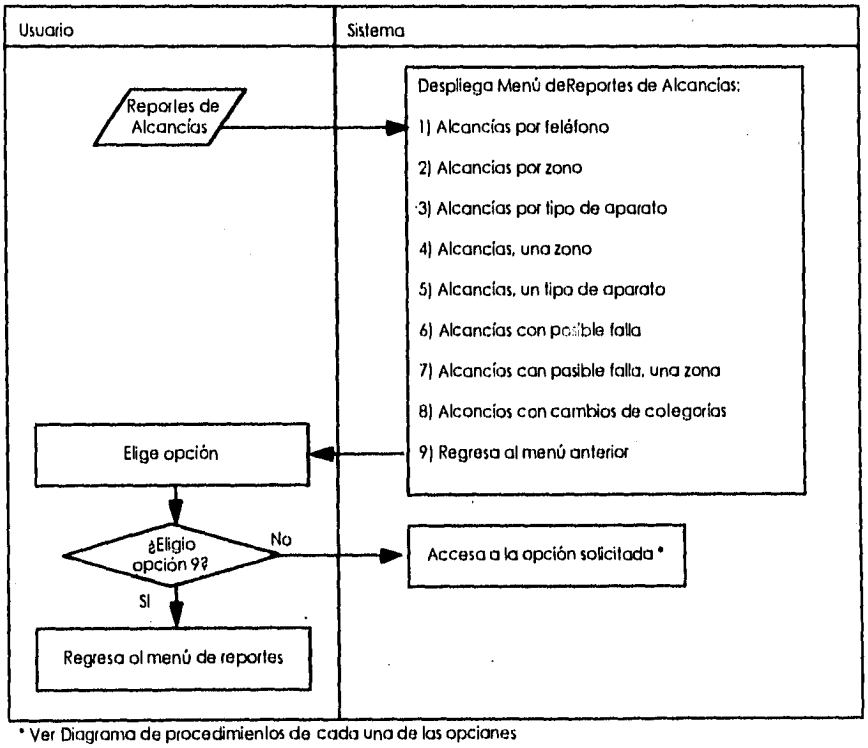


* Ver Diagrama de procedimientos de cada una de las opciones.

Fig. 3.4. - 23Diagrama de Procedimientos del Menú de Reportes.

Diagrama de Procedimientos del Menú de Reportes de Alcancías.

El usuario visualizará el menú de reportes de alcancías y elegirá la opción que desee ejecutar, si es la opción 9 regresa al menú de reportes. Figura 3.4. - 24.



* Ver Diagrama de procedimientos de cada una de las opciones

Fig. 3.4. - 24 Diagrama de Procedimientos del Menú de Reportes de Alcancías.

Diagrama de Procedimientos de Reportes de Alcancías por Teléfono.

Para obtener un reporte por número telefónico se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 25

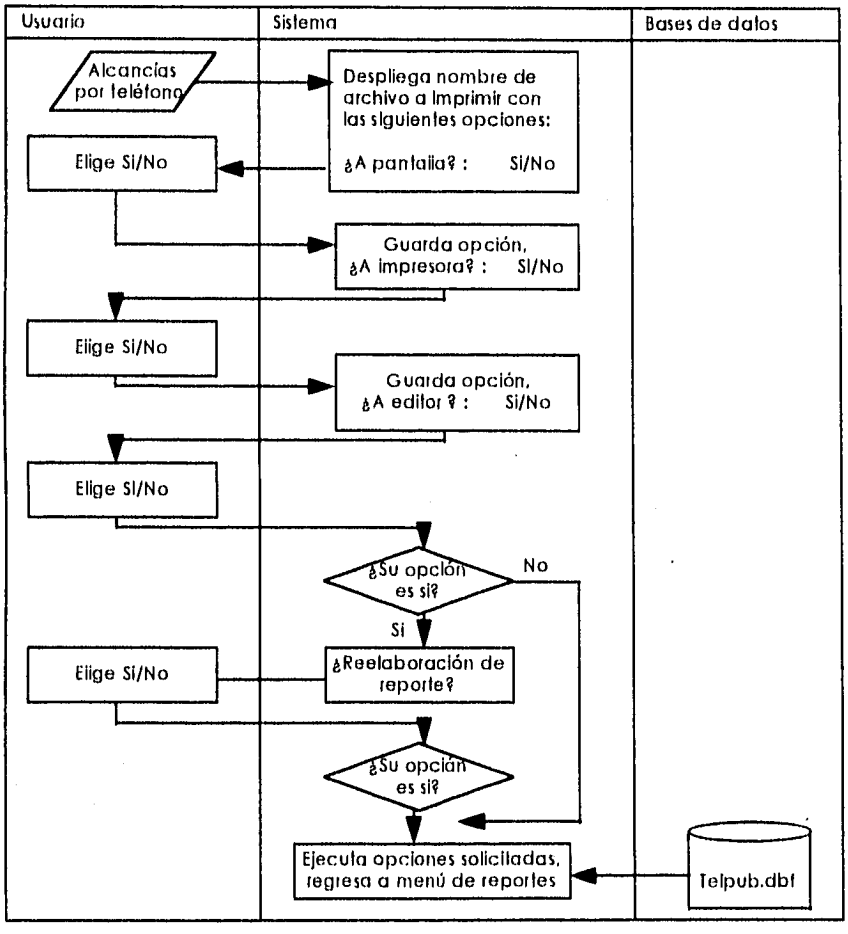


Fig. 3.4. - 25 Diagrama de Procedimientos de Reportes de Alcancías por Teléfono.

Diagrama de Procedimientos de Reportes de Alcancías por Zona.

Para obtener un reporte por zona monitoreada se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 26.

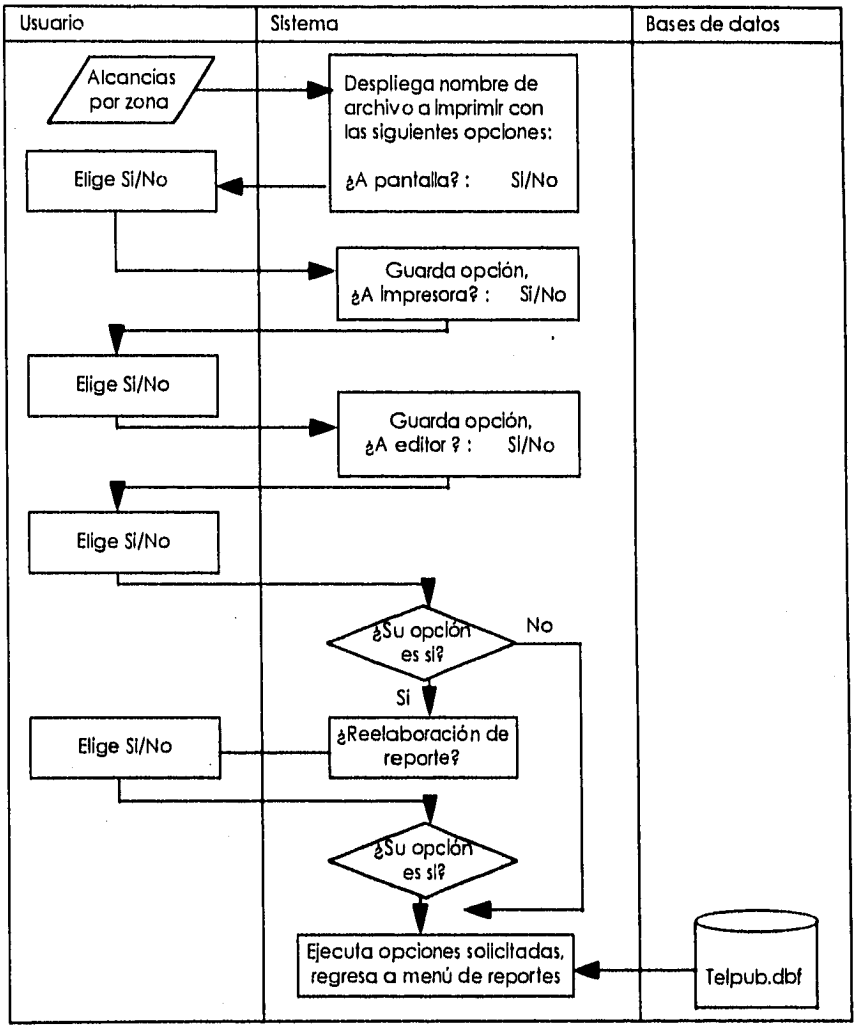


Fig. 3.4. - 26 Diagrama de Procedimientos de Reportes de Alcancías por Zona.

Diagrama de Procedimientos de Reportes de Alcancías por Tipo de Aparato.

Para obtener un reporte por tipo de aparato telefónico se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 27.

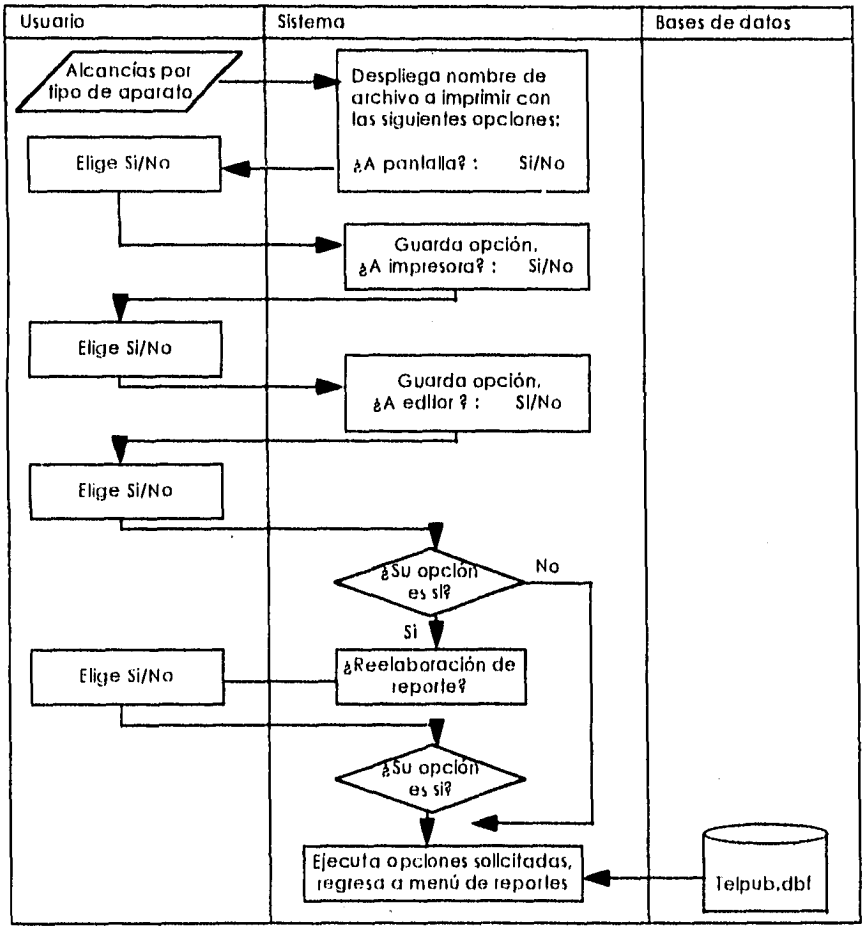


Fig. 3.4. - 27 Diagrama de Procedimientos de Reportes de Alcancías por Tipo de Aparato.

Diagrama de Procedimientos de Reportes de Alcancías, una Zona.

Para obtener un reporte por tipo de aparato telefónico y zona se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 28.

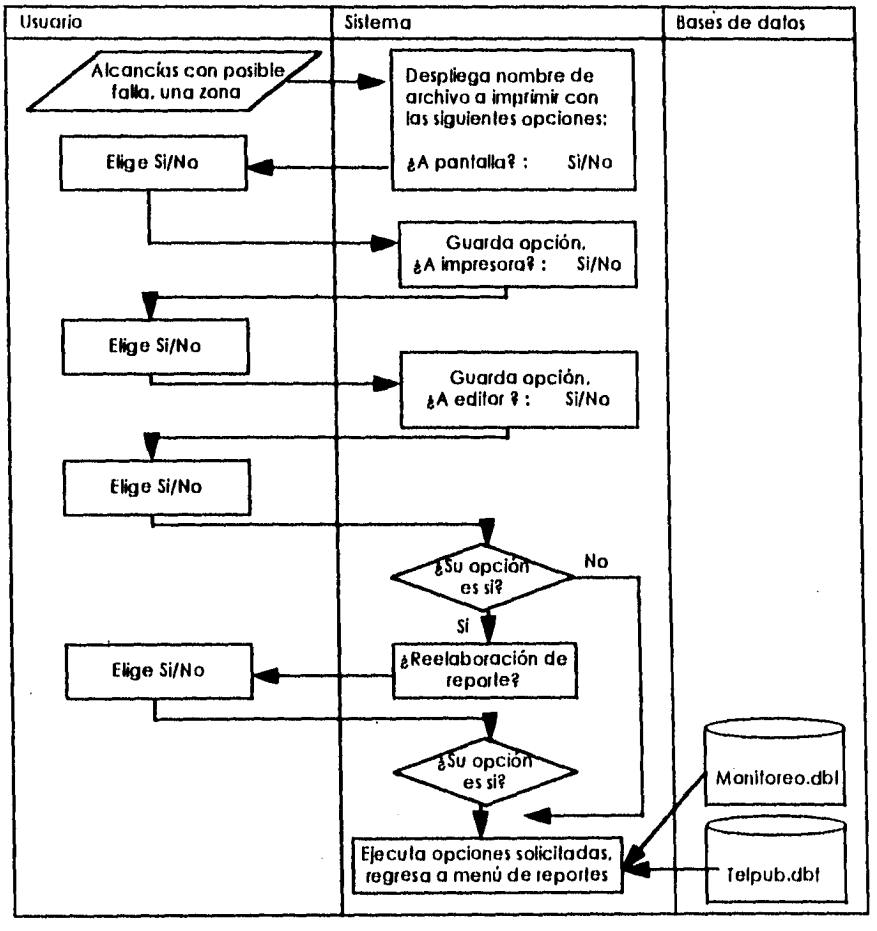


Fig. 3.4. - 28 Diagrama de Procedimientos de Reportes de Alcancías, una Zona.

Diagrama de Procedimientos de Reportes de Alcancías, Tipo de Aparato.

Para obtener un reporte por número y tipo de aparato telefónico, se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 29.

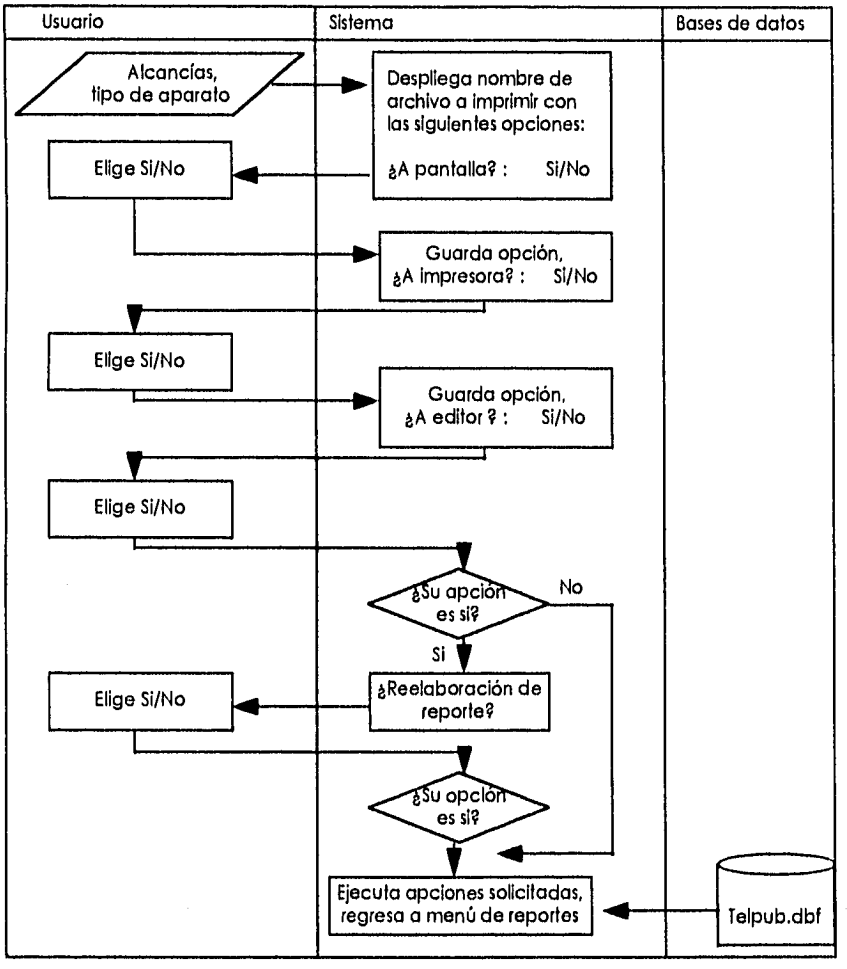


Fig. 3.4. - 29 Diagrama de Procedimientos de Reportes de Alcancías, Tipo de Aparato.

Diagrama de Procedimientos de Reportes de Alcancías con Posible Falla.

Para obtener un reporte por número telefónico con posible falla se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 30.

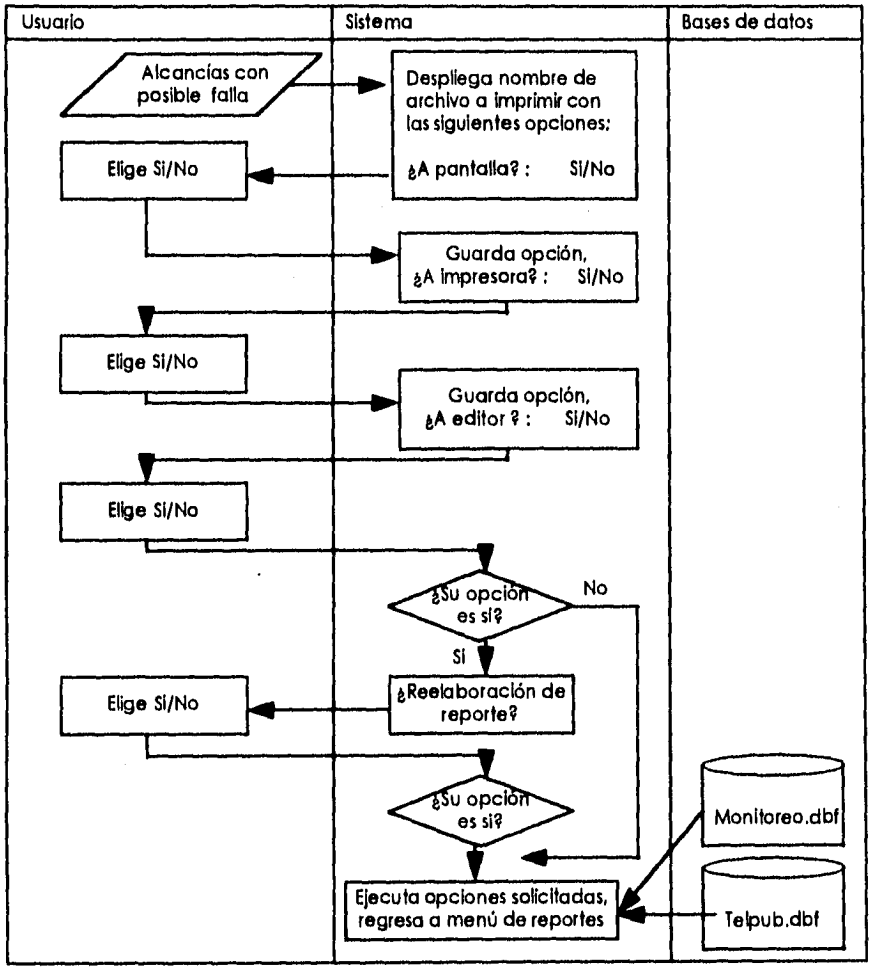


Fig. 3.4. - 30 Diagrama de Procedimientos de Reportes de Alcancías con Posible Falla.

Diagrama de Procedimientos de Reportes de Alcancías con Posible Falla, una Zona.

Para obtener un reporte por número telefónico con posible falla y una zona se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 31.

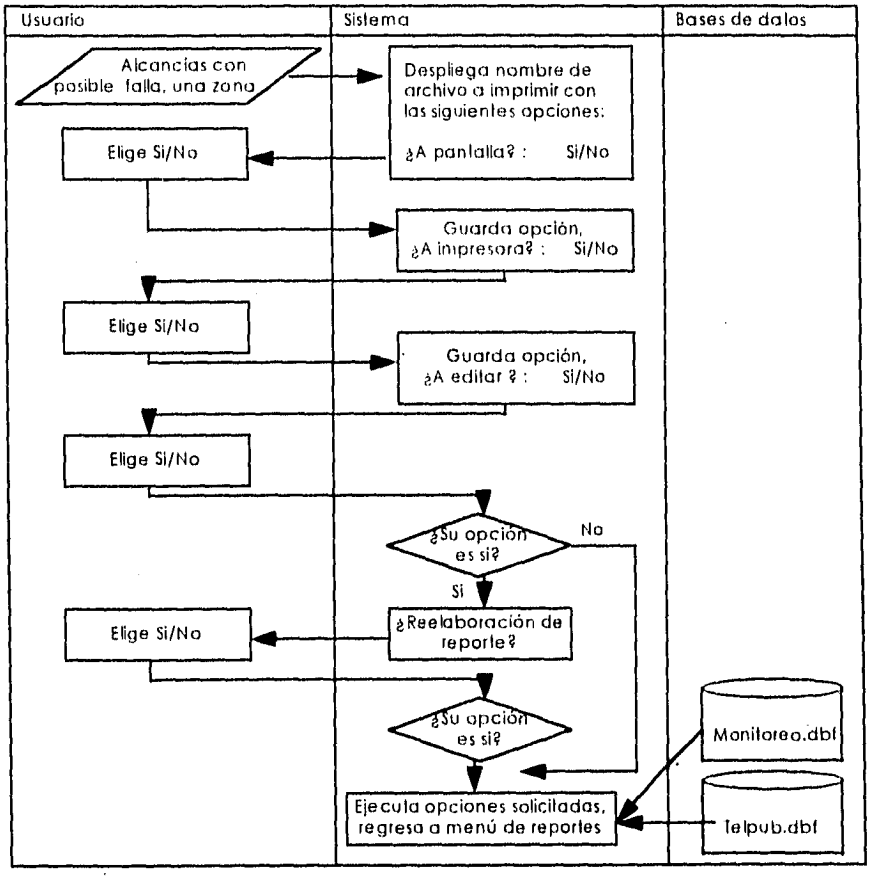


Fig. 3.4. - 31 Diagrama de Procedimientos de Reportes de Alcancías con Posible Falla, una Zona.

Diagrama de Procedimientos de Reportes de Alcancías con Cambio de Categorías.

Para obtener un reporte de alcancías con cambio de categorías se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 32.

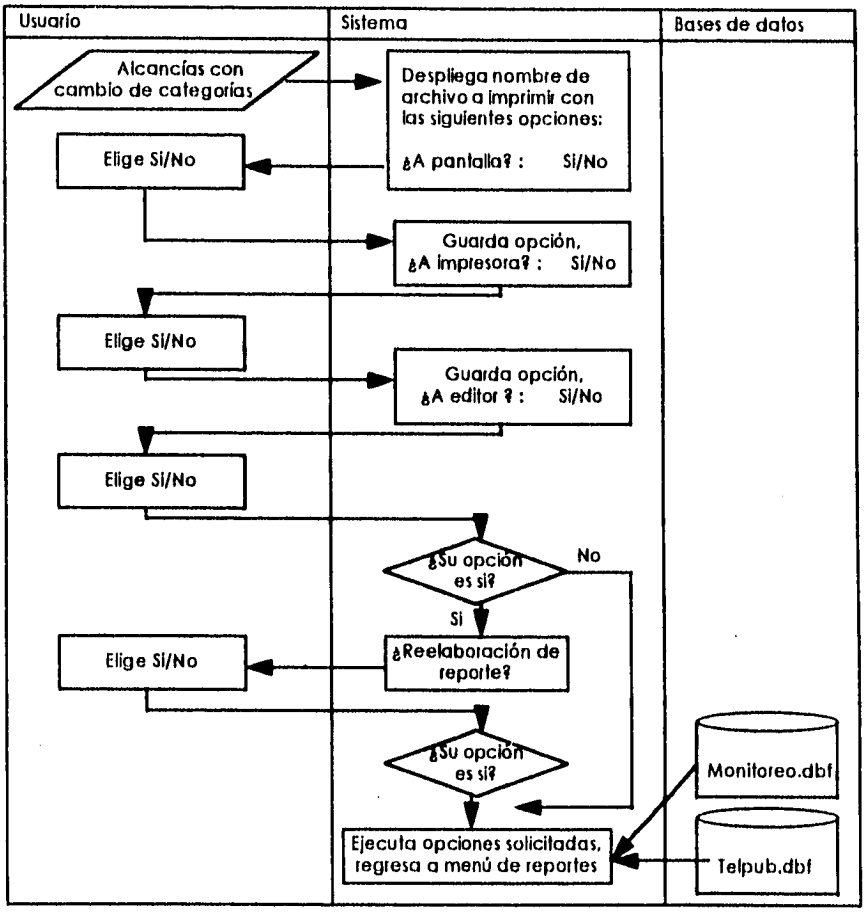
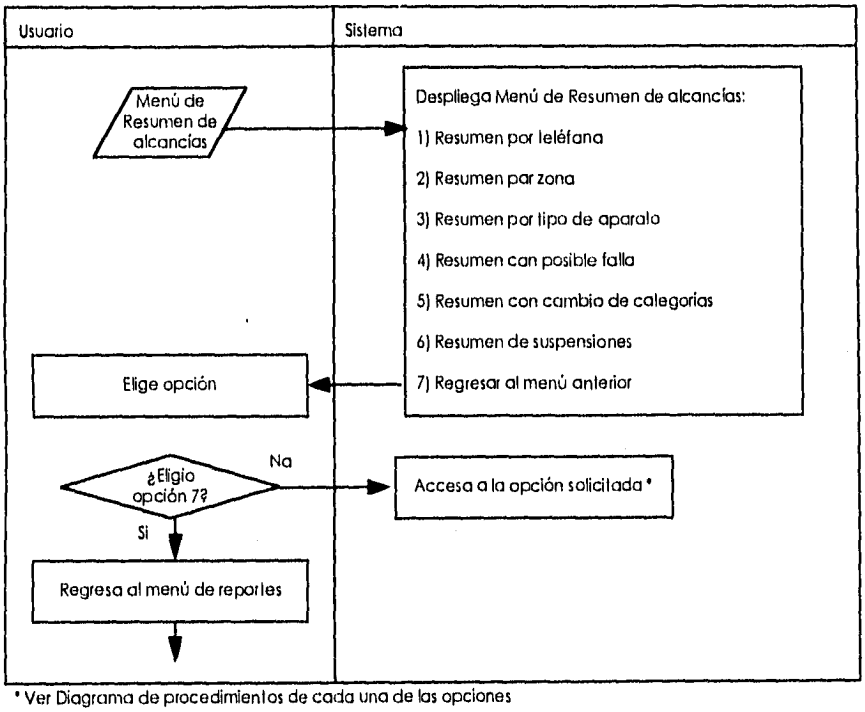


Fig. 3.4. - 32 Diagrama de Procedimientos de Reportes de Alcancías con Posible Falla.

Diagrama de Procedimientos del Menú de Resumen de Alcancías.

El usuario visualizará el menú de resumen de alcancías y elegirá la opción que desee ejecutar en caso contrario puede regresar al menú anterior con la opción 7. Figura 3.4. - 33.



* Ver Diagrama de procedimientos de cada una de las opciones

Fig. 3.4. - 33 Diagrama de Procedimientos del Menú de Resumen de Alcancías.

Diagrama de Procedimientos de Resumen por Teléfono.

Para obtener un reporte de resumen de alcancías por número telefónico se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 34.

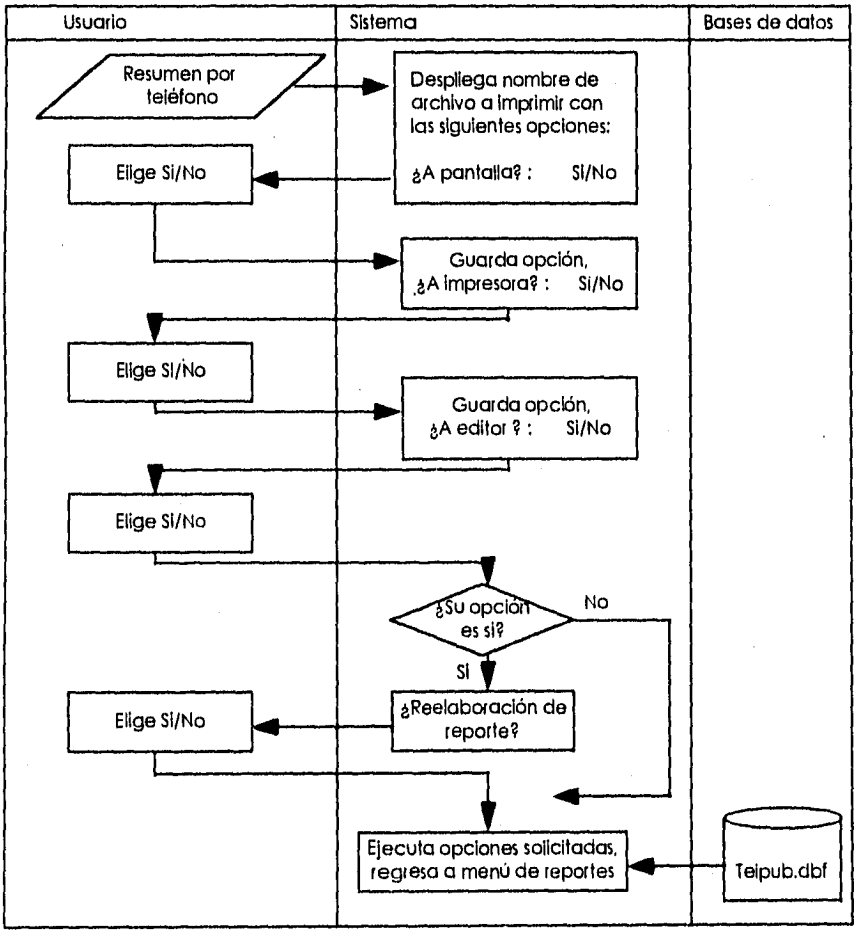


Fig. 3.4. - 34 Diagrama de Procedimientos de Resumen por Teléfono.

Diagrama de Procedimientos de Resumen por Zona.

Para obtener un reporte de resumen de alcancías por zona se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 35.

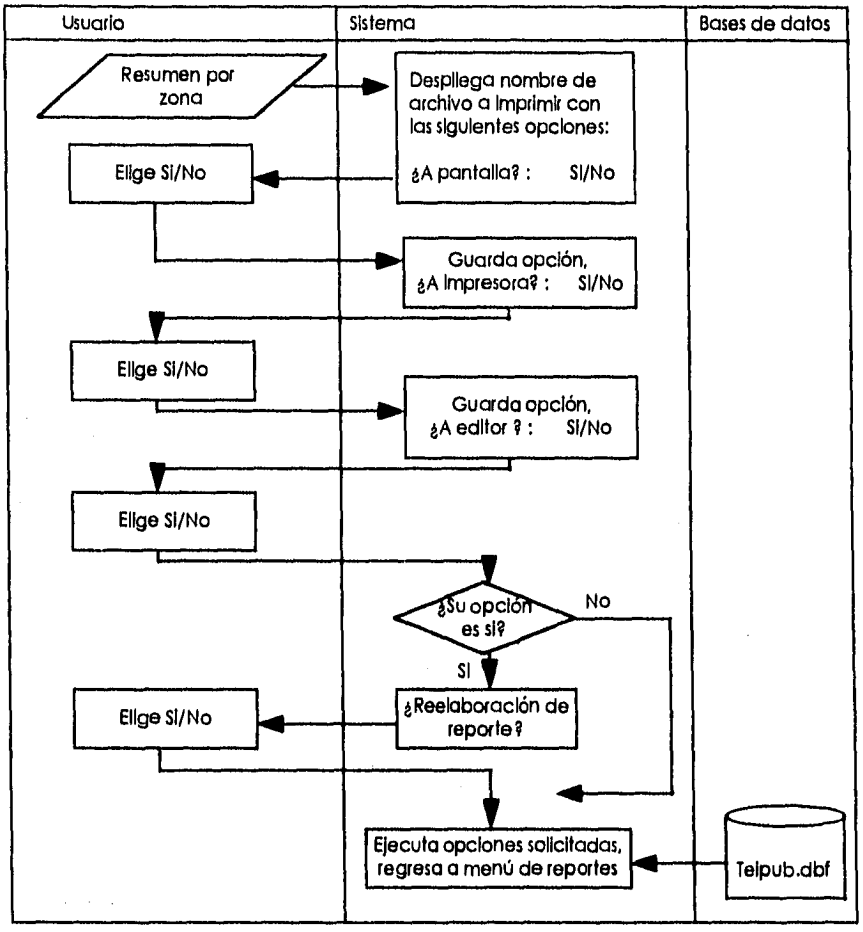


Fig. 3.4. - 35 Diagrama de Procedimientos de Resumen por Zona.

Diagrama de Procedimientos de Resumen por Tipo de Aparato.

Para obtener un reporte de resumen de alcancías por tipo de aparato se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 36.

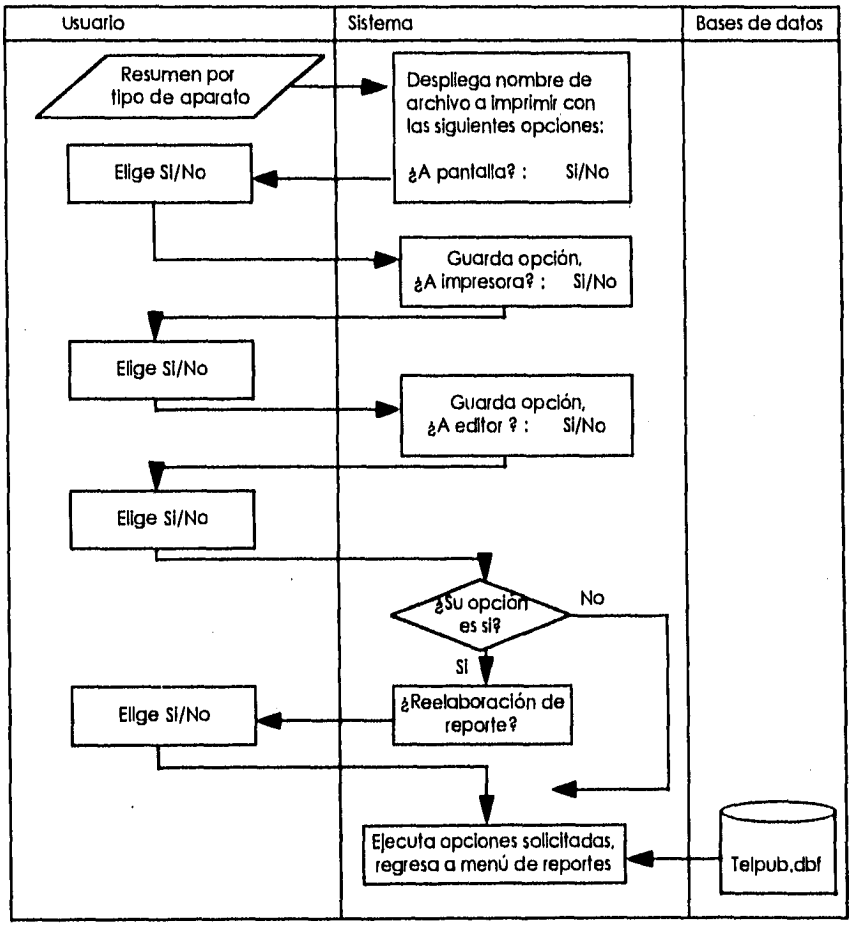


Fig. 3.4. - 36 Diagrama de Procedimientos de Resumen por Tipo de Aparato.

Diagrama de Procedimientos de Resumen con Posible Falla.

Para obtener un reporte de resumen de alcancías por zona se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 37.

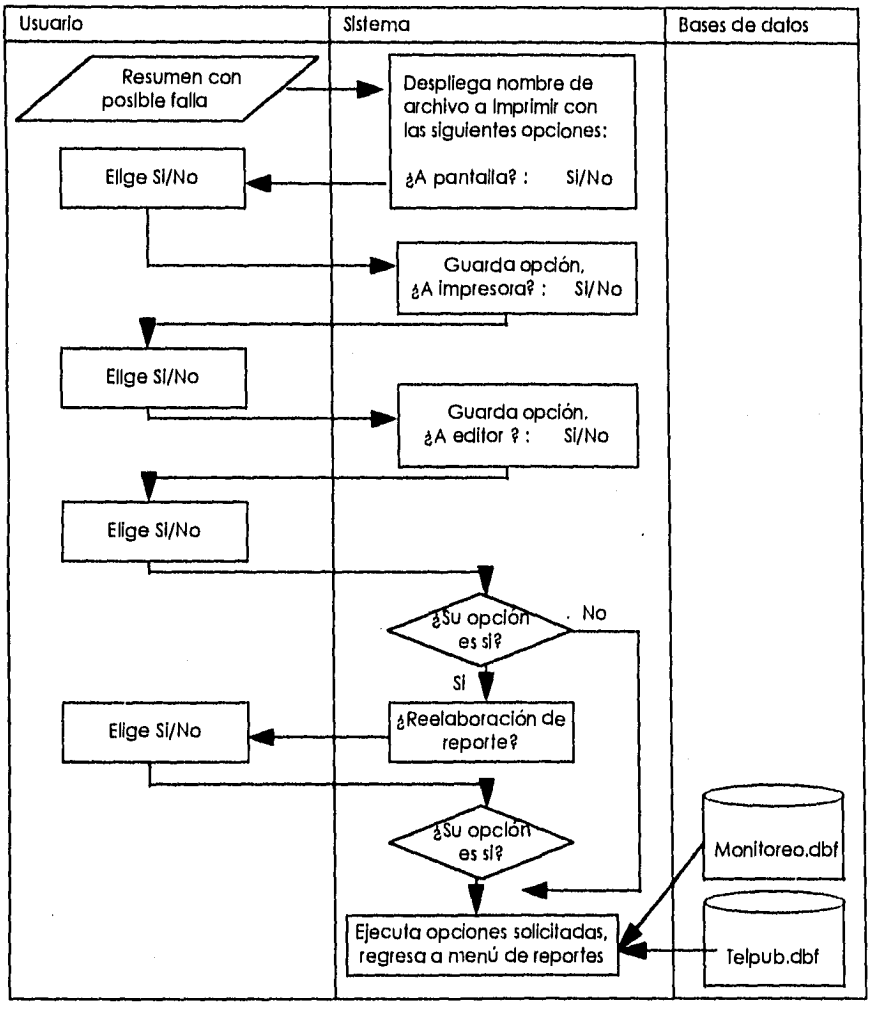


Fig. 3.4. - 37 Diagrama de Procedimientos de Resumen con Posible Falla.

Diagrama de Procedimientos de Resumen con Cambio de Categorías.

Para obtener un reporte de resumen de cambio de categorías se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 38.

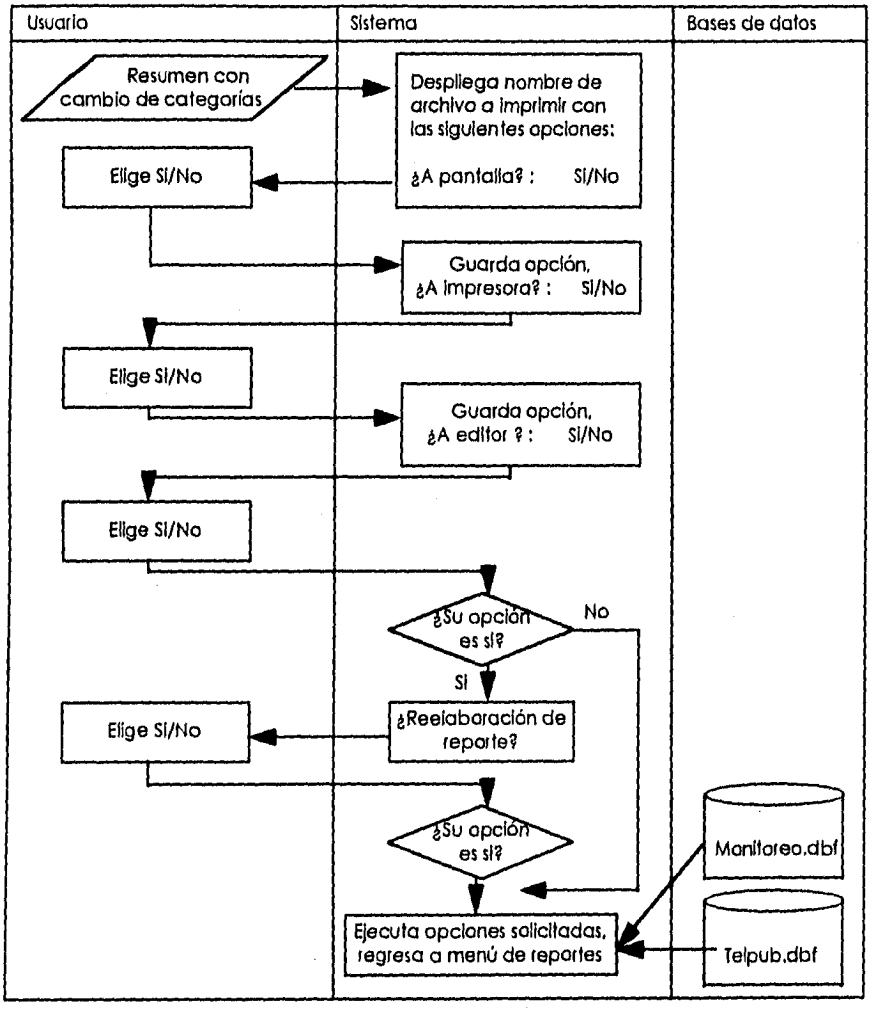


Fig. 3.4. - 38 Diagrama de Procedimientos de Resumen con Cambio de Categorías.

Diagrama de Procedimientos de Resumen de Suspensiones.

Para obtener un reporte de resumen de suspensiones se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 39.

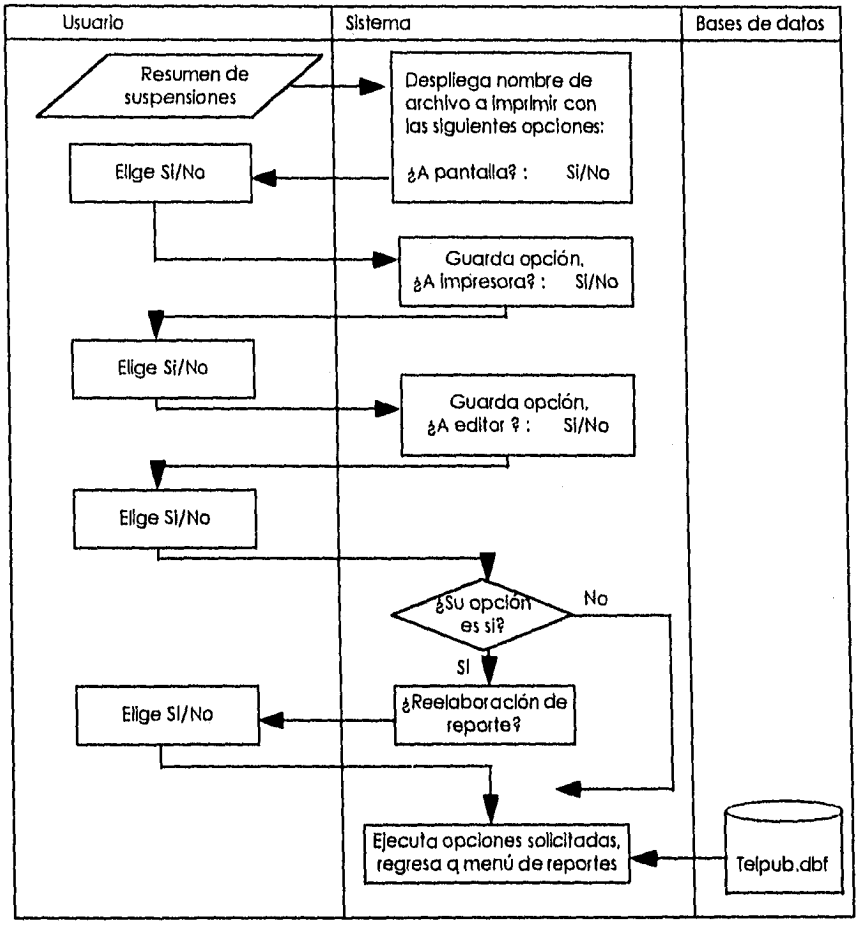


Fig. 3.4. - 39 Diagrama de Procedimientos de Resumen de Suspensiones.

Diagrama de Procedimientos del Menú de Totales de Alcancías.

El usuario visualizará el menú de totales de alcancías y elegirá la opción que desee ejecutar en caso contrario puede regresar al menú anterior con la opción 3. Figura 3.4. - 40.

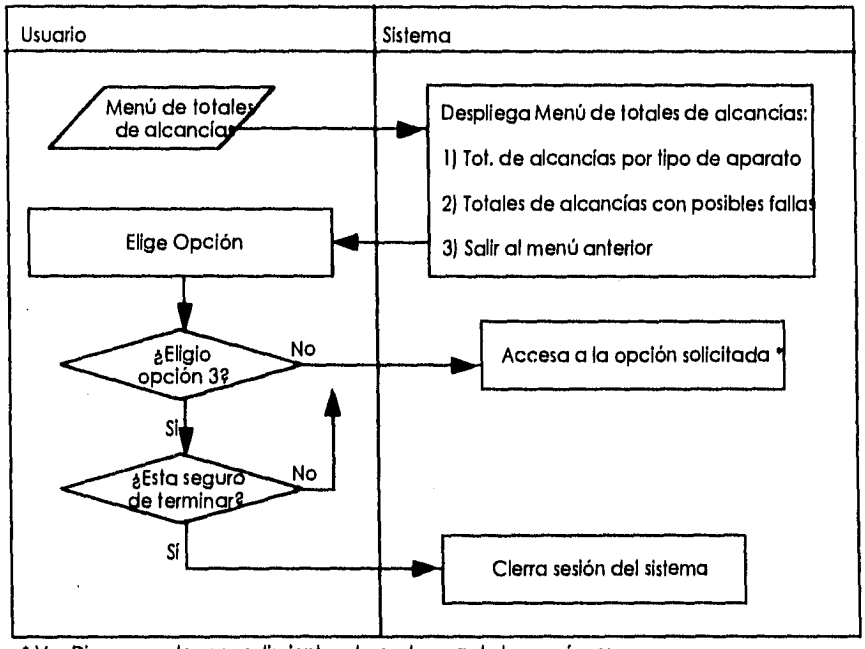


Fig. 3.4. - 40 Diagrama de Procedimientos del Menú de Totales de Alcancías.

Diagrama de Procedimientos de Totales de Alcancías por Tipo de Aparato.

Para obtener un reporte de totales de alcancías por tipo de aparato se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 41.

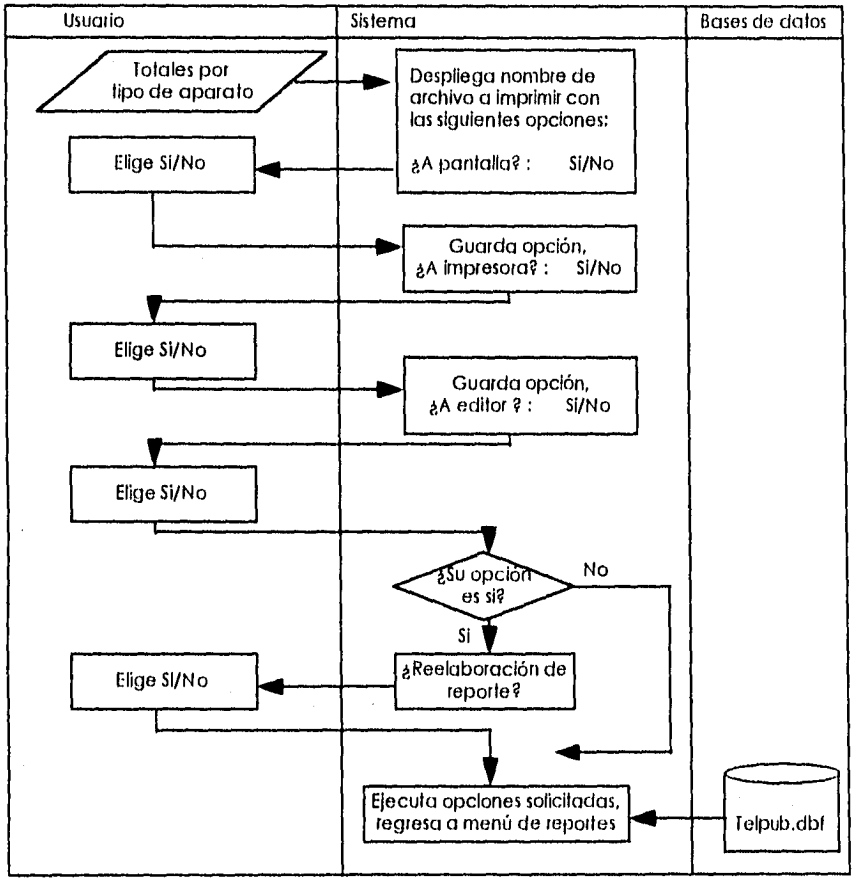


Fig. 3.4. 41 Diagrama de Procedimientos de Totales de Alcancías por Tipo de Aparato.

Diagrama de Procedimientos de Totales de Alcancías con Posible Falla.

Para obtener un reporte de totales de alcancías con posible falla se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 42.

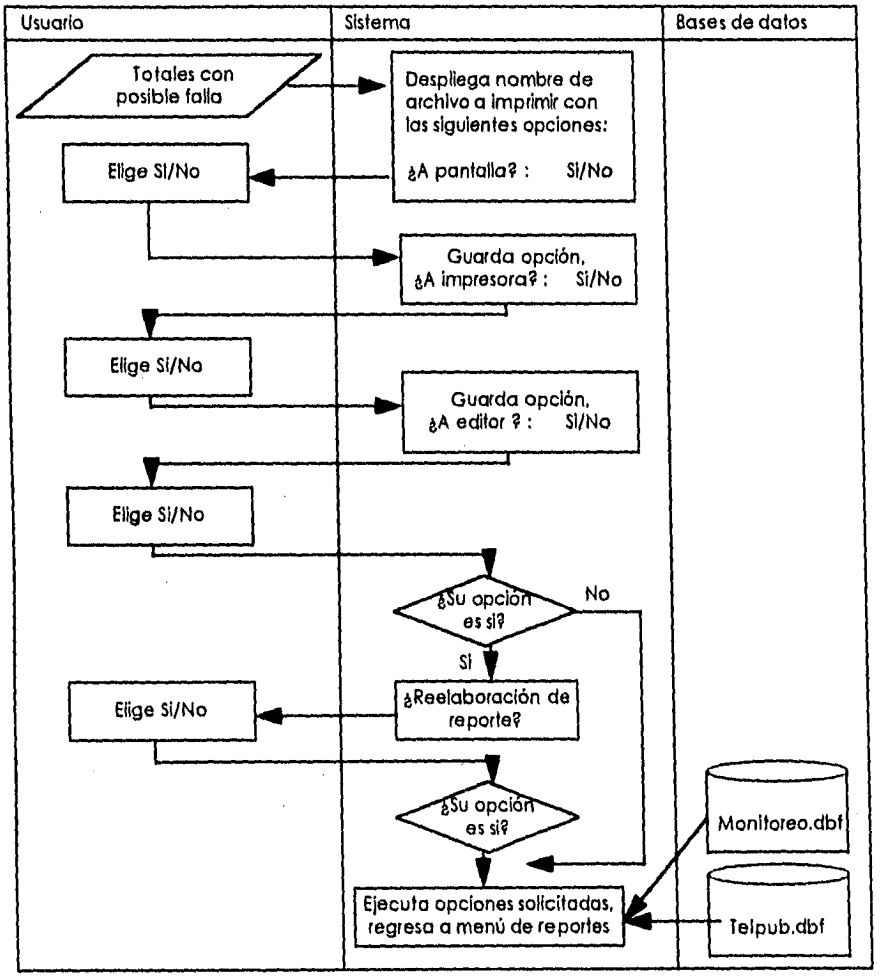
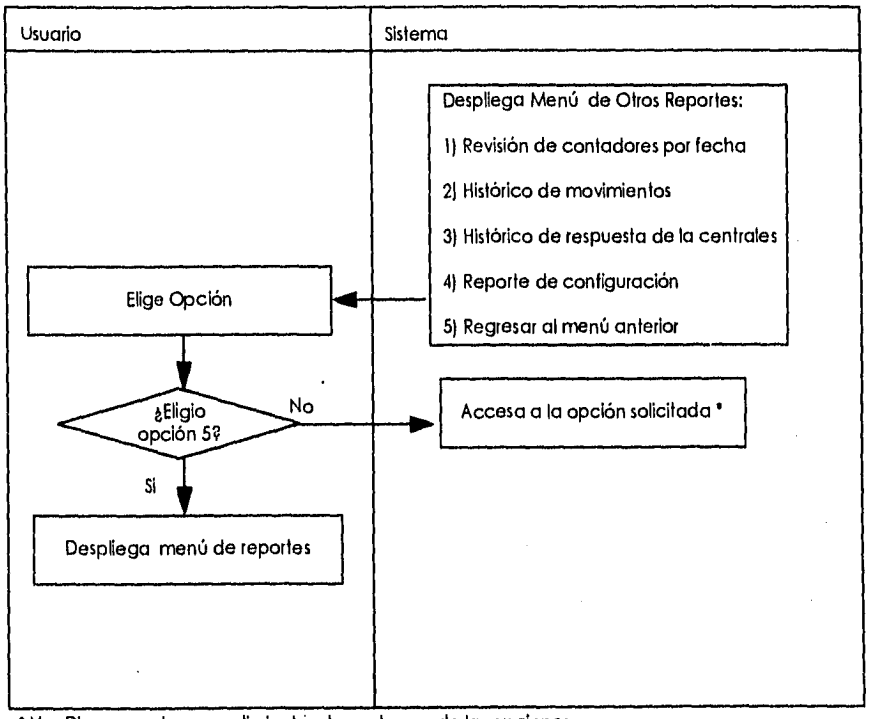


Fig. 3.4. - 42 Diagrama de Procedimientos de Totales de Alcancías con Posible Falla.

Diagrama de Procedimientos del Menú de Otros Reportes.

El usuario visualizará el menú de otros reportes y elegirá la opción que desee ejecutar, si la opción es 5 regresará al menú de Reportes. Figura 3.4. - 43.



* Ver Diagrama de procedimientos de cada una de las opciones.

Fig. 3.4. - 43 Diagrama de Procedimientos del Menú de Otros Reportes.

Diagrama de Procedimientos de Revisión de Contadores por Fecha.

Para obtener un reporte de la revisión de contadores por fecha (histórico) se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 44

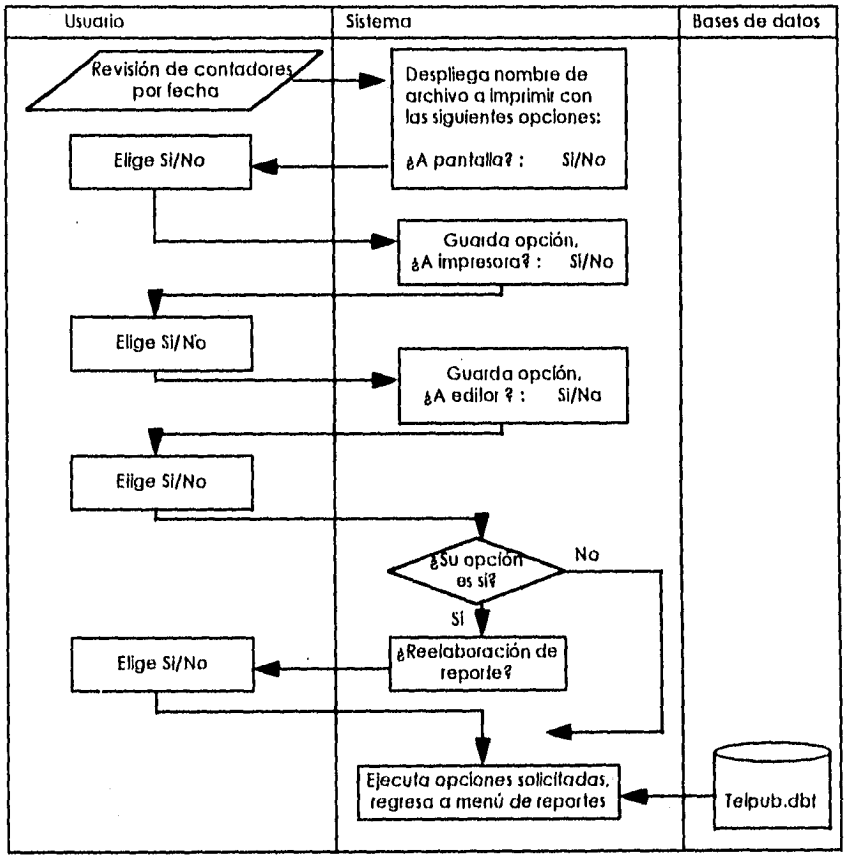


Fig. 3.4. - 44 Diagrama de Procedimientos de Revisión de Contadores por Fecha.

Diagrama de Procedimientos de Histórico de Movimientos.

Para obtener un reporte del histórico de movimientos se pedirá el password correspondiente para obtenerlo, se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 45.

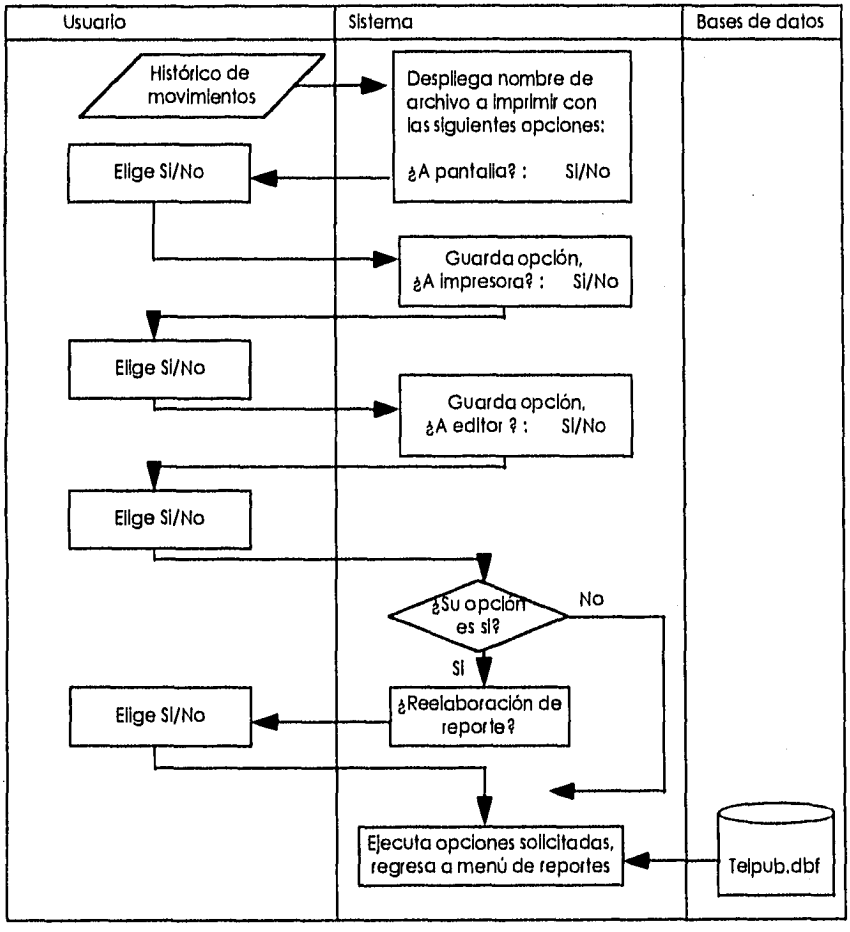


Fig. 3.4. - 45 Diagrama de Procedimientos de Histórico de Movimientos.

Diagrama de Procedimientos de Reporte de Configuración.

Para obtener un reporte de la configuración del sistema se dará una respuesta afirmativa o negativa de las opciones según sus requerimientos, así como escape para regresar al menú de reportes. Figura 3.4. - 46.

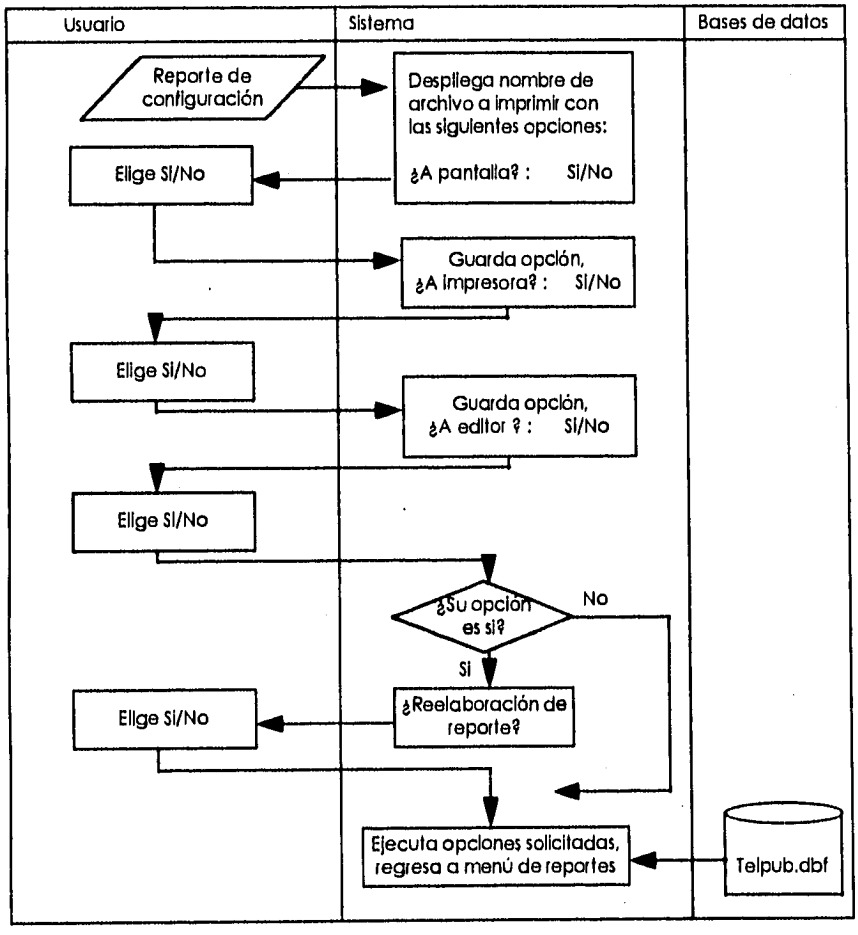
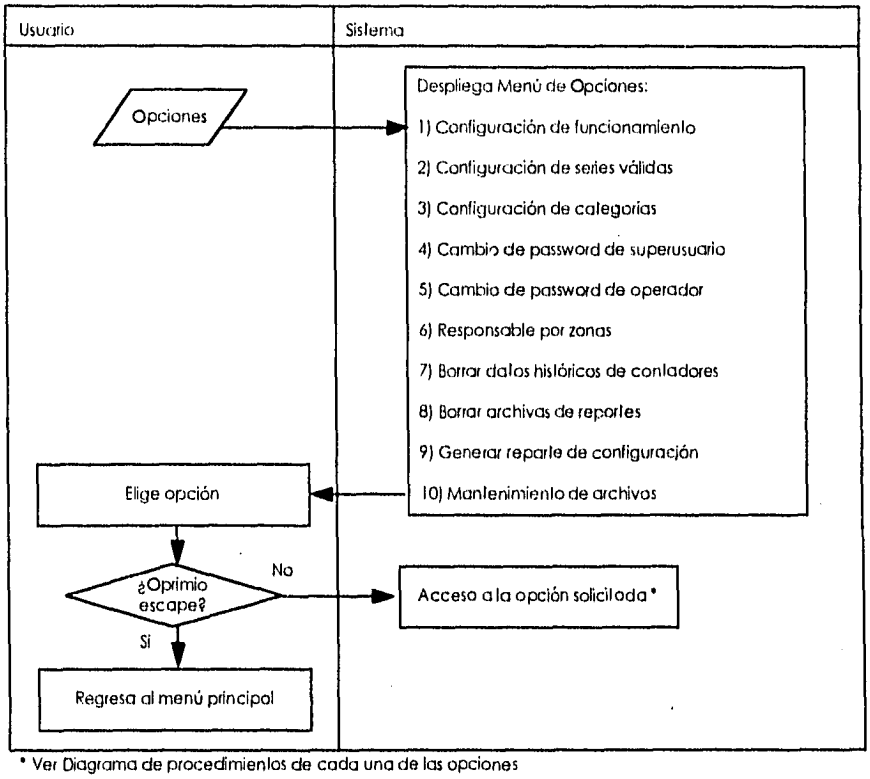


Fig. 3.4. - 46 Diagrama de Procedimientos de Reporte de Configuración.

Diagrama de Procedimientos del Menú de Opciones.

El usuario visualizará el menú de opciones de funcionamiento del sistema y elegirá la opción que desee ejecutar en caso contrario puede regresar al menú anterior oprimiendo la tecla escape. Figura 3.4. - 47.



* Ver Diagrama de procedimientos de cada una de las opciones

Fig. 3.4. - 47 Diagrama de Procedimientos del Menú de Opciones.

Diagrama de Procedimientos de Configuración de Funcionamiento.

Para configurar el sistema para un uso específico como la prueba de línea automática o reportes automáticos a impresora introducir un sí o un no a las opciones presentadas para este efecto. Figura 3.4. - 48.

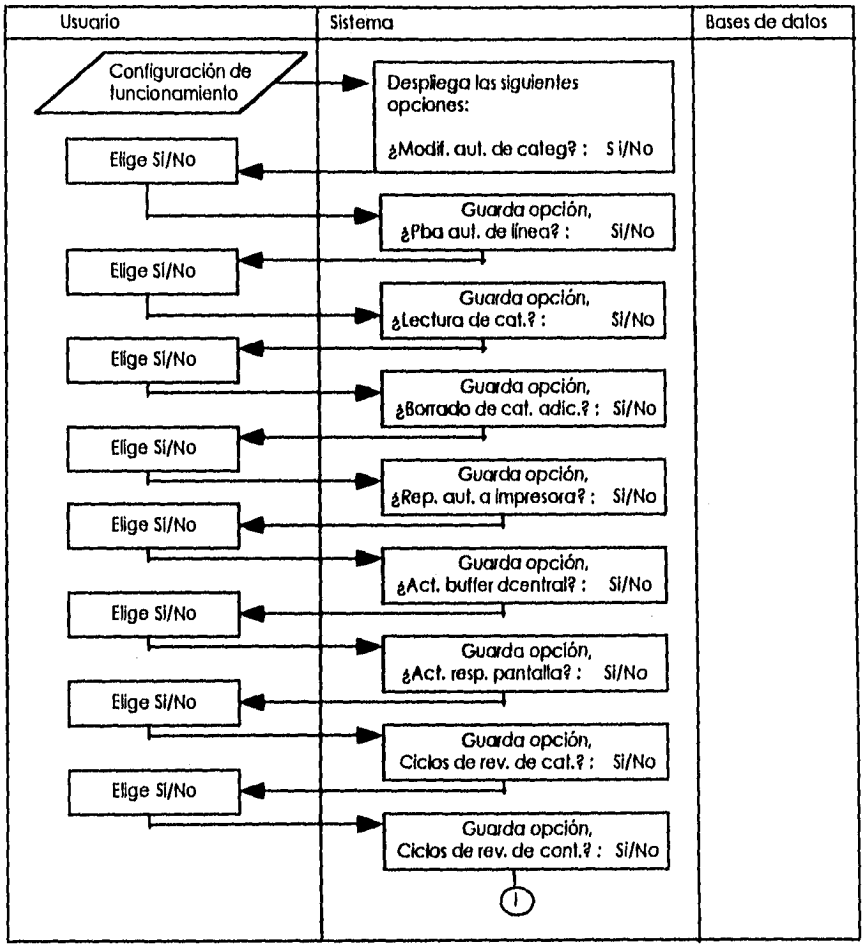


Fig. 3.4. - 48 Diagrama de Procedimientos de Configuración de Funcionamiento.

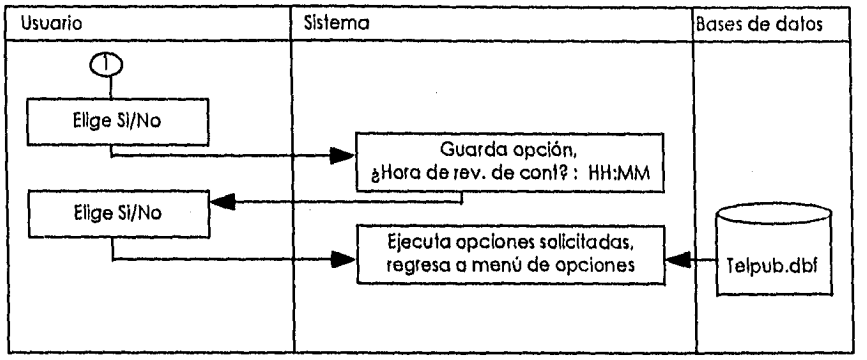


Fig. 3.4. - 48 Diagrama de Procedimientos de Configuración de Funcionamiento (Continuación).

Diagrama de Procedimientos de Configuración de Series Válidas.

Para configurar las series válidas se necesita ingresar el password y después los datos correspondientes a las nuevas series o a algún cambio en ellas. Figura 3.4. - 49

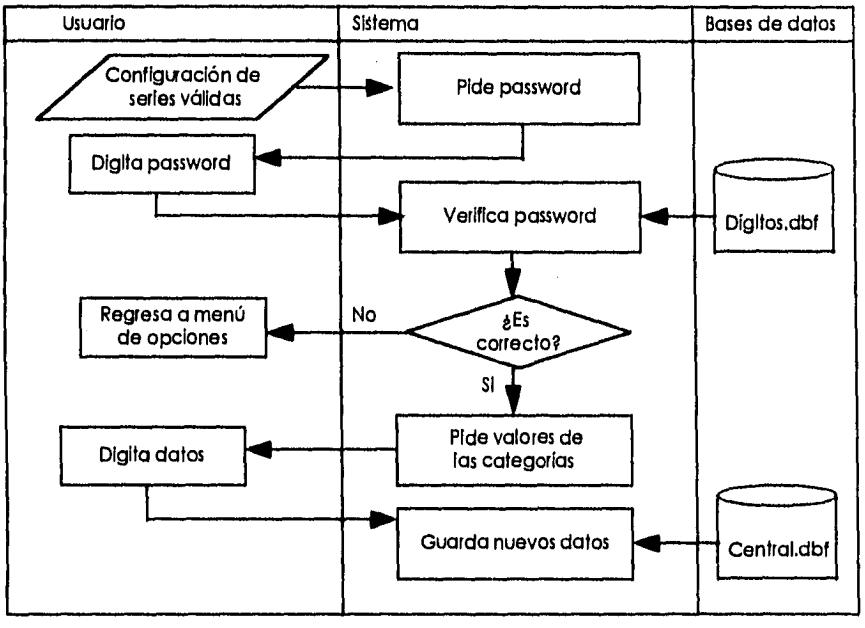


Fig. 3.4. - 49 Diagrama de Procedimientos de Configuración de Series Válidas.

Diagrama de Procedimientos de Configuración de Categorías.

Para configurar las categorías se necesita ingresar el password y después los valores correspondientes a las mismas. Figura 3.4. - 50.

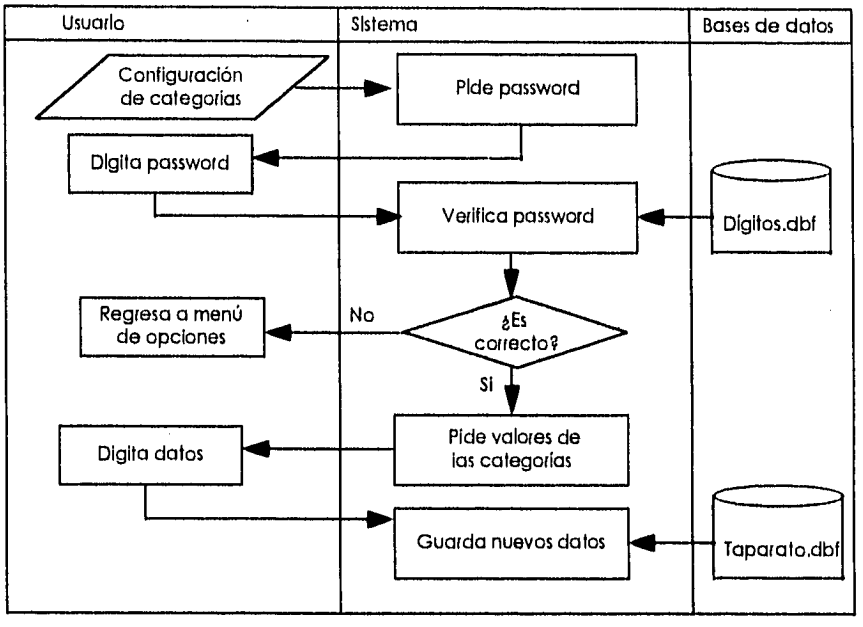


Fig. 3.4. - 50 Diagrama de Procedimientos de Configuración de Categorías.

Diagrama de Procedimientos de Cambio de Password de Superusuario.

Para hacer el cambio de password del superusuario se necesita ingresar el password autorizado si es correcto, ingresar el nuevo password y el sistema pedirá una verificación. Figura 3.4. - 51.

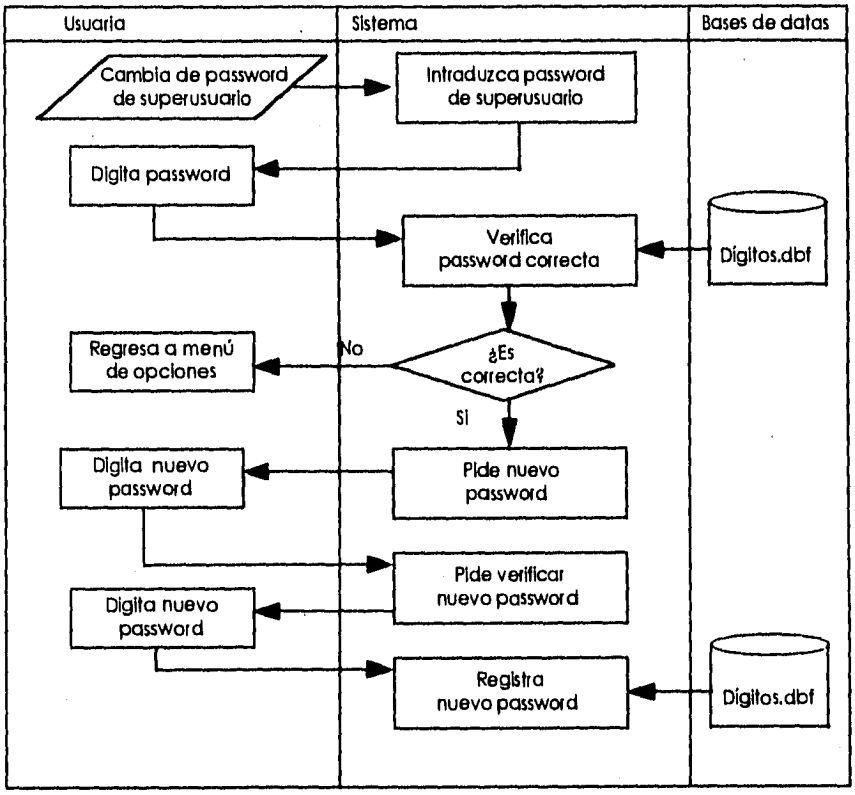


Fig. 3.4. - 51 Diagrama de Procedimientos de Cambio de Password de Superusuario.

Diagrama de Procedimientos de Cambio de Password de Operador.

Para hacer el cambio de password del operador se necesita ingresar el password autorizado si es correcto, ingresará el nuevo password y pedirá al sistema que realice una verificación. Figura 3.4. - 52.

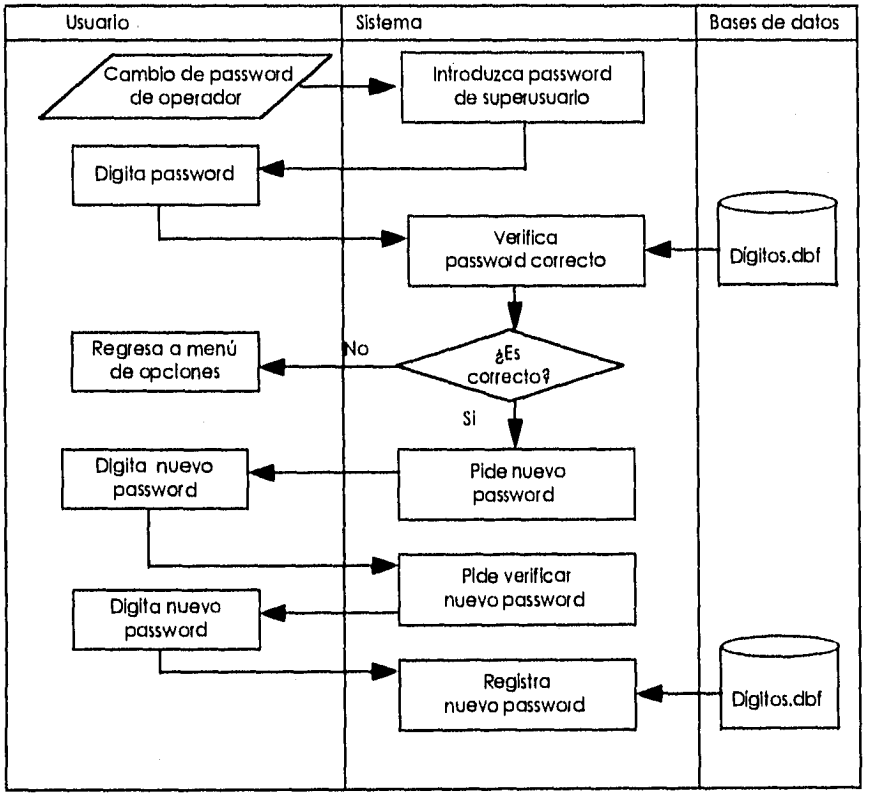


Fig. 3.4. - 52 Diagrama de Procedimientos de Cambio de Password de Operador.

Diagrama de Procedimientos de Responsables por Zona.

Para hacer un cambio o anexar el nombre de un responsable, solo se digitará la información y se dará un enter para terminar. Figura 3.4. - 53.

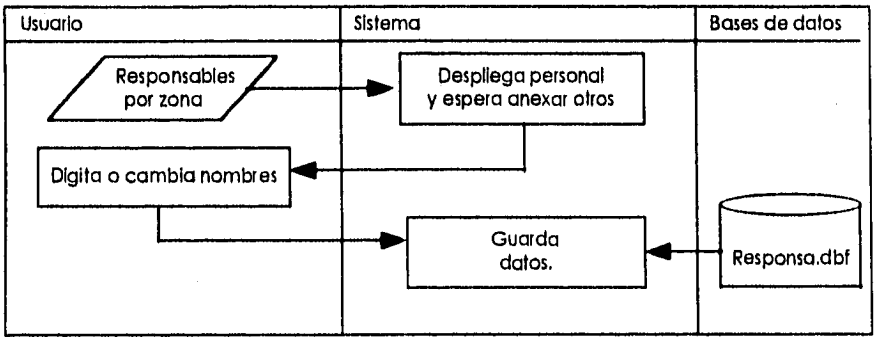


Fig. 3.4. - 53 Diagrama de Procedimientos de Responsables por Zona.

Diagrama de Procedimientos de Borrar Datos Históricos de Contadores.

Para borrar los datos históricos de los contadores se toma esta opción y se indica al sistema que se está seguro. Figura 3.4. - 54.

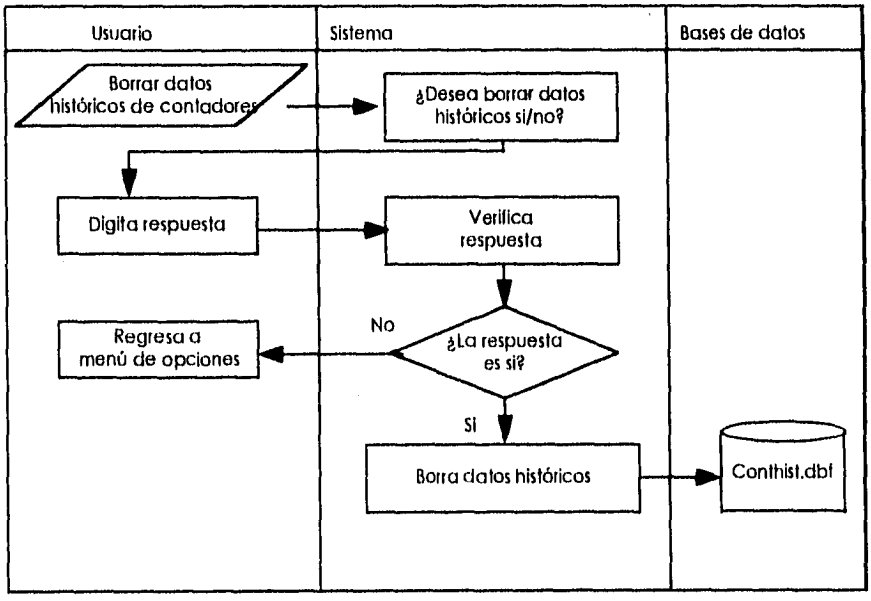


Fig. 3.4. - 54 Diagrama de Procedimientos de Borrar Datos Históricos de Contadores.

Diagrama de Procedimientos de Borrar Archivos de Reportes.

Para borrar los archivos de reportes que ya no se utilicen se elige esta opción y se indica al sistema que se está seguro. Figura 3.4. - 55.

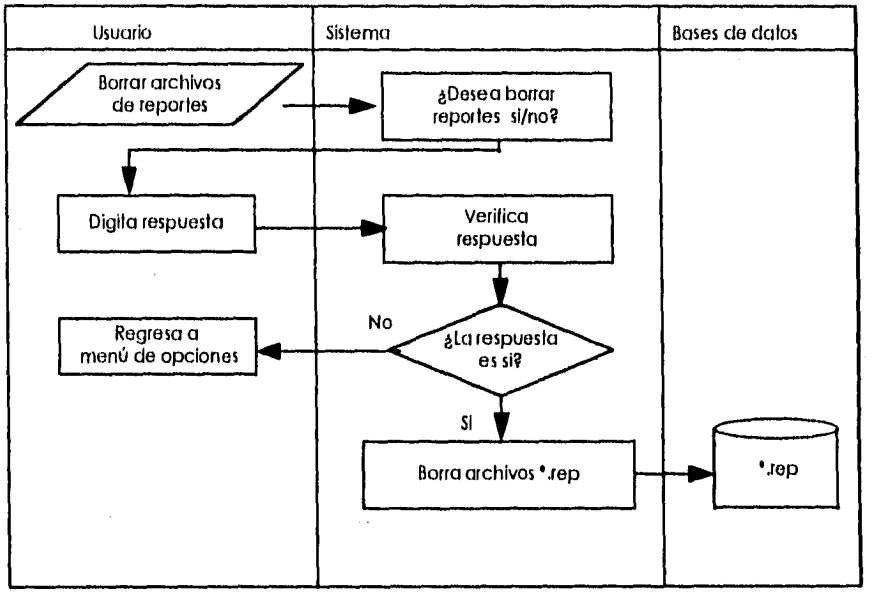
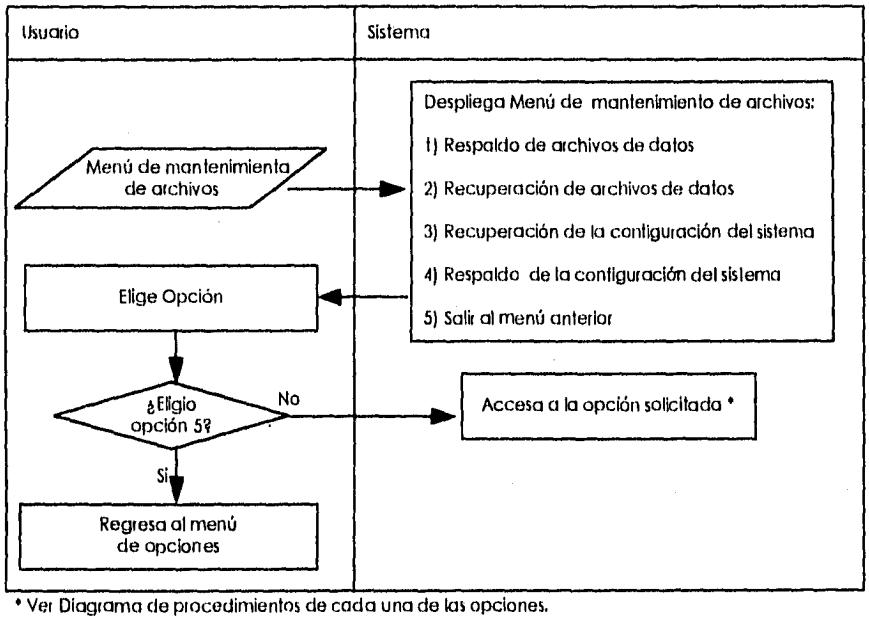


Fig. 3.4. - 55 Diagrama de Procedimientos de Borrar Archivos de Reportes.

Diagrama de Procedimientos del Menú de Mantenimiento de Archivos.

El usuario visualizará el menú de mantenimiento de archivos y elegirá la opción que desee ejecutar en caso contrario puede regresar al menú anterior (Opciones) al elegir la opción 5. Figura 3.4. - 56.



* Ver Diagrama de procedimientos de cada una de las opciones.

Fig. 3.4. - 56 Diagrama de Procedimientos del Menú de Mantenimiento de Archivos.

Diagrama de Procedimientos de Respaldo de Archivos de Datos.

Para respaldar en disco duro o flexible los archivos de datos se toma esta opción y nos indica paso a paso que hacer. Figura 3.4. - 57.

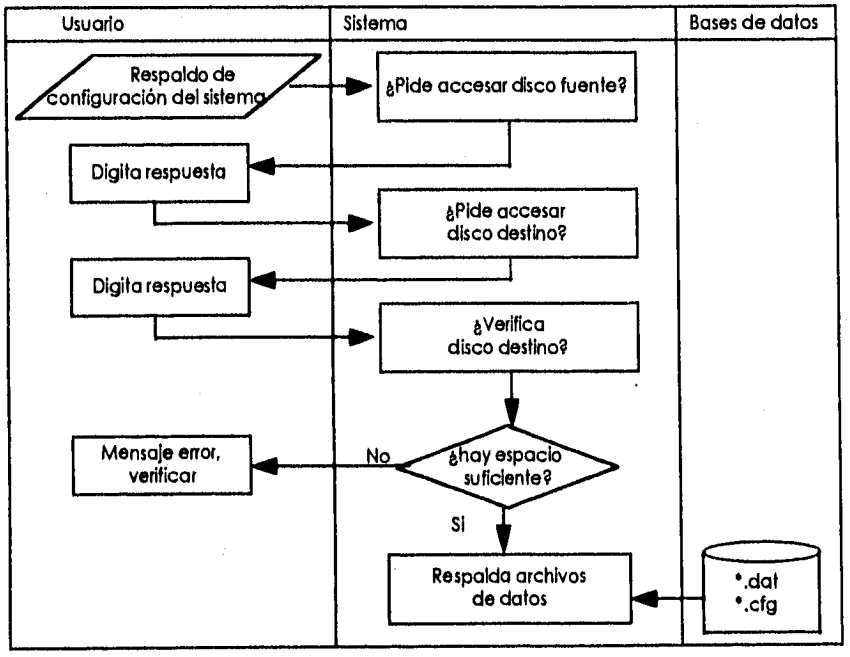


Fig. 3.4. - 57 Diagrama de Procedimientos de Respaldo de Archivos de Datos.

Diagrama de Procedimientos de Recuperación de Archivos de Datos.

Para recuperar de disco duro o flexible los archivos de datos se toma esta opción y nos indica paso a paso que hacer. Figura 3.4. - 58.

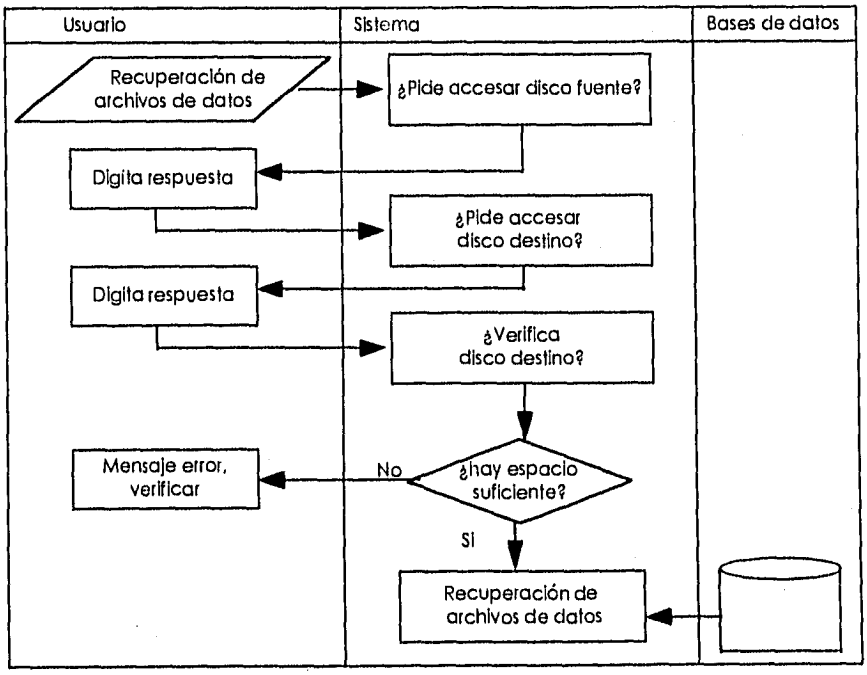


Fig. 3.4. - 58 Diagrama de Procedimientos de Recuperación de Archivos de Datos.

Diagrama de Procedimientos de Respaldo de Configuración del Sistema.

Para respaldar en disco duro o flexible los archivos de datos se toma esta opción y nos indica paso a paso que hacer. Figura 3.4. - 59.

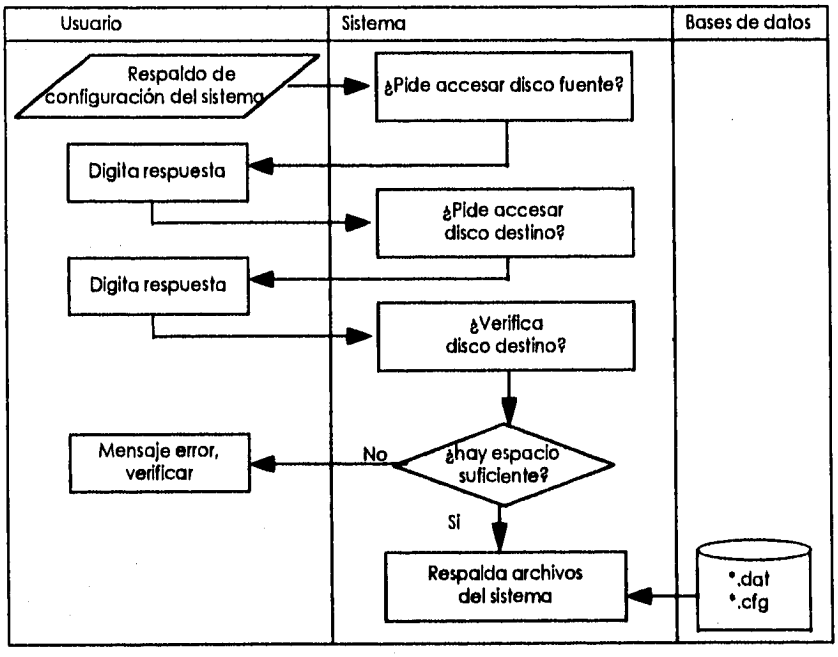


Fig. 3.4. - 59 Diagrama de Procedimientos de Respaldo de Configuración del Sistema.

Diagrama de Procedimientos de Recuperación de Configuración del Sistema.

Para recuperar de disco duro o flexible los archivos de datos se toma esta opción y nos indica paso a paso que hacer. Figura 3.4. - 60

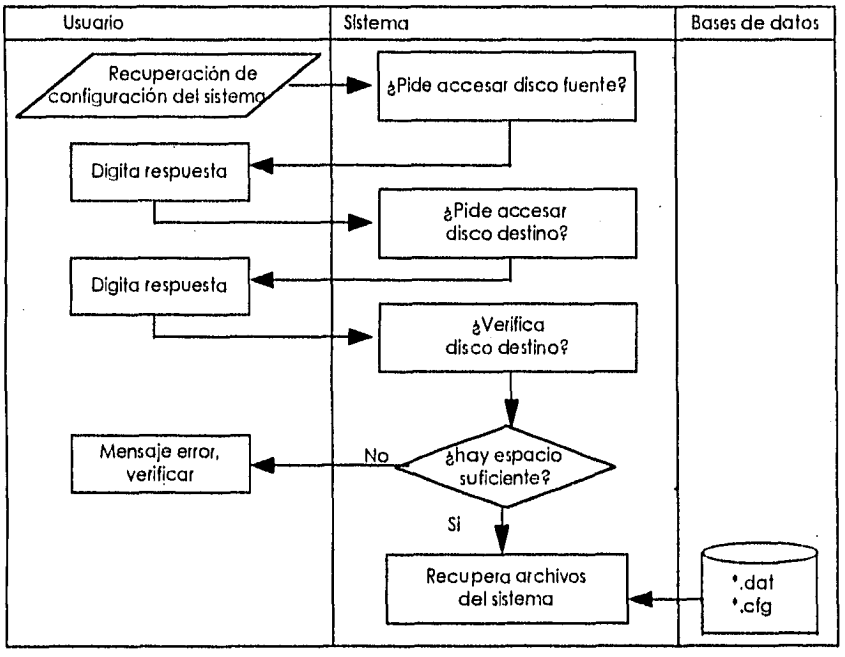


Fig. 3.4. - 60 Diagrama de Procedimientos de Recuperación de Configuración del Sistema.

3.5. Diagrama Entidad-Relación.

Para tener una óptima relación entre las tablas que se manejaran en el sistema, se hizo un análisis del cual se obtuvieron las siguientes relaciones:

La tabla TEL_PUB tiene relación con la tabla CONTHIST de 1 a n por el campo NUM_TEL dado que se guarda el histórico de las lecturas de los teléfonos públicos. También TEL_PUB tiene relación de n a uno por el campo de ZONA con la tabla de RESPONSA por tener solo un responsable para n teléfonos. Y a su vez TEL_PUB tiene una relación con MONITOR de uno a uno con el campo NUM_TEL ya que es necesario monitorear cada uno de los números telefónicos.

MONITOR esta relacionado con la tabla STATCONT por el campo STATUS de n a uno ya que se hace el monitoreo de una serie de teléfonos y se guarda únicamente una llave que identifica las fallas en común de los mismos. A su vez MONITOR se relaciona con STATCAT por el campo de STATUS de n a uno ya que se realiza la función anterior. También MONITOR se relaciona con CENTRAL por el campo IDCENTRAL de n a uno dado que en este se guarda el tipo de central que se esta monitoreando. Y por último, MONITOR tiene relación con la tabla TSERVICI por el campo TSERVICIO de n a uno ya que se refiere al tipo de servicio por cada uno de los teléfonos.

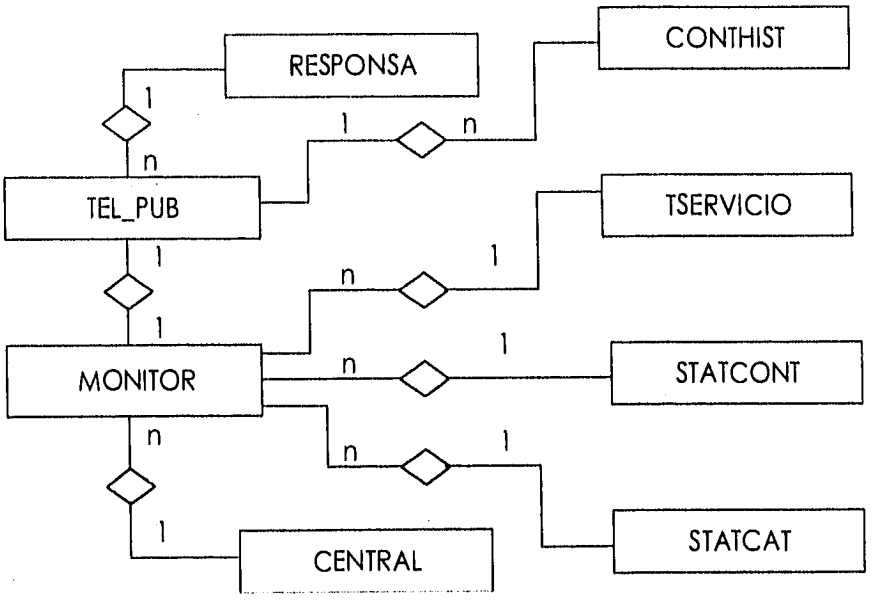


Fig. 3.5. - 1 Diagrama Entidad - Relación.

3.6. Diccionario de Datos.

A continuación se describen cada una de las entidades usadas en el sistema:

No.	Campo	Tipo	Longitud	Descripción
1	NUM_TEL	Carácter	8	Número telefónico
2	CLAVE_POB	Carácter	7	Clave de la población
3	ALARMA	Carácter	6	Número de identificación del aparato telefónico
4	GRUPO	Carácter	5	Número de grupo asignado a un conjunto de aparatos
5	TIPO_APTO	Carácter	2	Descripción del tipo de aparato Pueden ser: PUBLICO1, MON LARGAD, CREDITO, MON Y CRED., DEBITO, CRED. Y DEB, PUBLICO2, TPTC, PATROCINADO, AMPER.
6	TIPOAMB	Carácter	2	Tipo de uso del aparato
7	LUGAR_INS	Carácter	1	Lugar de instalación
8	FECHA_INS	Fecha	8	Fecha de instalación
9	CALLEYNUM	Carácter	50	Calle y número de ubicación del aparato.
10	COLONIA	Carácter	50	Colonia de la ubicación del aparato
11	C_P	Carácter	5	Código Postal
12	DISTRITO	Carácter	5	Distrito de la conexión
13	PRINCIPAL	Carácter	5	Principal del aparato
14	SECUNDARIO	Carácter	14	Secundario
15	ZONA	Carácter	2	Zona donde se encuentra el aparato
16	SINCONTROL	Carácter	1	Bandera sin control del monitoreo
17	SUSPENDIDO	Carácter	1	Bandera de suspensión del usuario

Fig. 3.6. - 1 Base de Datos TEL_PUB.

No.	Campo	Tipo	Longitud	Descripción
1	TSERVICIO	Numérico	2	Tipo de servicio
2	NUM_TEL	Carácter	8	Número telefónico
3	CACBA	Numérico	3	Categoría actual EXA: CBA Restricción de larga Distancia
4	CATLI	Numérico	3	Categoría actual EXA: TLI Inversión de Polaridad
5	CABIC	Numérico	3	Categoría actual EXA: BIC Restricción de Llamadas
6	CASSI	Numérico	3	Categoría actual EXA: SSI Tipo de Teclado
7	CAOBA	Numérico	3	Categoría actual EXA: OBA Candado de Línea
8	CATCL	Numérico	3	Categoría actual EXA: TCL Teléfono Público .
9	CAICS	Numérico	3	Categoría actual EXA: ICS Mensaje de Suspensión
10	CABOC	Numérico	3	Categoría actual EXA: BOC Inhibición de Tono de Marcar
11	CATBO	Numérico	3	Categoría actual EXA: TBO Suspensión de línea
12	CABADP	Numérico	3	Categoría actual SIST99: BADP Suspensión
13	CACBOX	Numérico	3	Categoría actual SIST99: CBOX Alcancía
14	CAORPT	Numérico	3	Categoría actual SIST99: ORTP Restricción de llamada
15	CASSIG	Numérico	3	Categoría actual SIST99: SSIG Tipo de Teclado
16	CATRSTR	Numérico	3	Categoría actual SIST99: TRSTR Restricción de llamada
17	CALNCHAR	Numérico	3	Categoría actual SIST99: LNCHAR Polaridad
18	CAHM	Numérico	3	Categoría actual SIST99: HM Polaridad
19	CASUBGRP	Numérico	3	Categoría actual del subgrupo
20	STATCAT	Numérico	3	Status de las categorías

Fig. 3.6. - 2 Base de Datos MONITOR.

No.	Campo	Tipo	Longitud	Descripción
21	STATCONT	Numérico	3	Status del contador
22	SINMOV	Numérico	1	Bandera sin movimiento de contador
23	CATNOK	Numérico	1	Bandera cambio de categoría
24	SINCONTROL	Numérico	3	Bandera sin control del monitoreo
25	CONTADORU	Numérico	10	Lectura penúltima del contador
26	CONTADORP	Numérico	10	Lectura penúltima del contador
27	HORACATU	Carácter	5	Hora última de lectura de categoría
28	HORACATP	Carácter	5	Hora penúltima de lectura de contador
29	HORACONTU	Carácter	5	Hora última de lectura de contador
30	HORACONTP	Carácter	5	Hora penúltima de lectura de categoría
31	FECHACATU	Fecha	8	Fecha última de lectura de categoría
32	FECHACATP	Fecha	8	Fecha penúltima de lectura de categoría
33	FECHACONTU	Fecha	8	Fecha última de lectura de contador
34	FECHACONTP	Fecha	8	Fecha penúltima de lectura de contador
35	SEGURIDAD	Carácter	36	Reservado para posteriores aplicaciones
36	IDCENTRAL	Carácter	3	Identificación de la central

Fig. 3.6. - 2 Base de Datos MONITOR(Continuación).

No.	Campo	Tipo	Longitud	Descripción
1	TSERVICIO	Carácter	2	Tipo de servicio
2	DERVICIO	Carácter	15	Descripción del servicio
3	CNCBA	Numérico	3	Categoría necesaria EXA: CBA Restricción de larga distancia.
4	CNTLI	Numérico	3	Categoría necesaria EXA: TLI Inversión de polaridad
5	CNBIC	Numérico	3	Categoría necesaria EXA: BIC Restricción de llamadas de entrada.
6	CNSSI	Numérico	3	Categoría necesaria EXA: SSI Tipo de teclado
7	CNOBA	Numérico	3	Categoría necesaria EXA: OBA Candado de línea
8	CNTCL	Numérico	3	Categoría necesaria EXA: TCL Teléfono público
9	CNICS	Numérico	3	Categoría necesaria EXA: ICS Mensaje de suspensión
10	CNBOC	Numérico	3	Categoría necesaria EXA: BOC Inhibición de tono de marcar
11	CNTBO	Numérico	3	Categoría necesaria EXA: TBO Suspensión de línea.
12	CNBADP	Numérico	3	Categoría necesaria SIST99: BADP Suspensión
13	CNCBOX	Numérico	3	Categoría necesaria SIST99: CBOX Alcancía
14	CNORPT	Numérico	3	Categoría necesaria SIST99: ORPT Restricción de llamada salida
15	CNSSIG	Numérico	3	Categoría necesaria SIST99: SSIG Tipo de teclado
16	CNTRSTR	Numérico	3	Categoría necesaria SIST99: TRSTR Llamada de terminal
17	CNLNCHAR	Numérico	3	Categoría necesaria SIST99: LNCHAR Polaridad
18	CNHM	Numérico	3	Categoría necesaria SIST99: HM Polaridad

Fig. 3.6. - 3 Base de Datos TSERVICI.

No.	Campo	Tipo	Longitud	Descripción
1	IDCENTRAL	Carácter	3	Identificación de la central
2	LIMINF	Carácter	8	Límite inferior de la serie
3	LIMSUP	Carácter	8	Límite superior de la serie
4	TCENTRAL	Carácter	1	Protocolo a usar
5	PASSWORD	Carácter	10	Clave de acceso
6	LOGIN	Carácter	10	Login del rango de teléfonos
7	CFGPUERTO	Carácter	12	Configuración de acceso al modem
8	MODEM	Carácter	12	Número telefónico de acceso a la central
9	ANALDIG	Carácter	1	Tipo de la central (analógica/digital)
10	DIGITOS	Carácter	1	Número de dígitos en el comando

Fig. 3.6. - 4 Base de Datos CENTRAL

No.	Campo	Tipo	Longitud	Descripción
1	NUM_TEL	Carácter	8	Número telefónico
2	HORA	Carácter	5	La hora de la lectura del contador
3	FECHA	Fecha	8	La fecha de la lectura del contador
4	CONTADOR	Carácter	10	Es el acumulado del contador

Fig. 3.6. - 5 Base de Datos CONTHIST.

No.	Campo	Tipo	Longitud	Descripción
1	STATUS	Numerico	3	Status de la prueba de contadores
2	DESCRIP	Carácter	60	Descripción del status de la prueba de contadores

Fig. 3.6. - 6 Base de Datos STATCONT.

No.	Campo	Tipo	Longitud	Descripción
1	STATUS	Numérico	3	Status de prueba de categorías
2	DESCRIP	Carácter	60	Descripción del status de prueba de categoría

Fig. 3.6. - 7 Base de Datos STATCAT.

No.	Campo	Tipo	Longitud	Descripción
1	ZONA	Carácter	2	Zona donde se encuentra el aparato
2	RESPONSA	Carácter	40	Responsable de la central

Fig. 3.6. - 8 Base de Datos RESPONSA.

3.7. Construcción de la Base de Datos en CODEBASE.

A continuación se presentará una explicación de la codificación de la Base de Datos en CODEBASE. Para hacer esto, se desglosará el código del archivo "TEL_PUB.DBF" .

Creación de un Archivo de Datos.

Comenzaremos explicando la forma en que se construye un archivo tipo " *.DBF" el cual no existe a partir de una estructura predefinida en C. Esto se lleva a cabo de la siguiente manera:

Se declara una estructura de número de campos variable, la cual se agrupan en renglones de 4 descriptores:

- Nombre del campo tipo cadena.
- Tipo de campo para el cual CODEBASE tiene predefinidas constantes que lo describen, por ejemplo r4str (string), r4num (entero),...,etc.
- Longitud del campo.
- Solo se usa si se manejan números tipo flotante, (describiendo la parte fraccionaria).

```
FIELD4INFO fieldinfo[] =
{
        {"NUM_TEL",   r4str,S_NUM_TEL,0},
        {"CLAVE_POB", r4str,S_CLAVE_POB,0},
        {"ALARMA",    r4str,S_ALARMA,0},
        {"GRUPO",     r4str,S_GRUPO,0},
```

```
    {"TIPO_APTO", r4str,S_TIPO_APTO,0},
    {"TIPOAMB",   r4str,S_TIPOAMB,0},
    {"LUGAR_INS", r4str,S_LUGAR_INS,0},
    {"FECHA_INS", r4date,S_FECHA_INS,0},
    {"CALLEYNUM", r4str,S_CALLEYNUM,0},
    {"COLONIA",   r4str,S_COLONIA,0},
    {"C_P",       r4str,S_C_P,0},
    {"DISTRITO",  r4str,S_DISTRITO,0},
    {"PRINCIPAL", r4str,S_PRINCIPAL,0},
    {"SECUNDARIO",r4str,S_SECUNDARIO,0},
    {"ZONA",      r4str,S_ZONA,0},
    {"SUSPENDIDO",r4num,S_SUSPENDIDO,0},
    {"SINCONTROL",r4num,S_SINCONTROL,0},
    {0,0,0,0} ,
};
```

Y se aplica la función **d4create**, con los siguientes parámetros:

- CODEBASE que contiene los datos de inicialización del Manejador de Base de Datos.
- Nombre del archivo.
- Estructura del archivo.
- Estructura de los índices, explicados más adelante.

Para nuestro caso:

```
DATA4 alcanfile=d4create(&codebase,NF_ALCANCIA,fieldinfo,tag_info);
```

Creación de un Archivo de Indices.

De forma similar, se construye un archivo " *.CDX " que no existe a partir de una estructura predefinida en C y ésta se lleva a cabo de la siguiente manera:

- Nombre del Indice.
- Llave.
- Filtro.
- Llave única.
- Ascendente/Descendente.

```
TAG4INFO tag_info[] =
{
        {"TELE_TAG","NUM_TEL",0,0,0},
        {"ZONA_TAG","ZONA+NUM_TEL",0,0,0},
        {0,0,0,0,0},
};
```

Y se aplica la función **i4create**, con los siguientes parámetros:

- Estructura DATA4 que contiene los datos del archivo a indexar.
- Nombre del archivo de índices
- Estructura de los índices,

Para nuestro caso:

```
INDEX4 alcanindex=i4create(alcanfile,NX_ALCANCIA,tag_info);
```

Abrir un Archivo de Datos.

Una vez creado un archivo, este se puede abrir, pero se deben declarar unas variables de CODEBASE llamadas FIELD4 que contienen el apuntador al buffer del registro.

Para nuestro caso:

```
FIELD4 *FNum_Tel;
FIELD4 *FClave_Pob;
FIELD4 *FAlarma;
FIELD4 *FGrupo;
FIELD4 *FTipo_Apto;
FIELD4 *FTipoAmb;
FIELD4 *FLugar_Ins;
FIELD4 *FFecha_Ins;
FIELD4 *FCalleyNum;
FIELD4 *FColonia;
FIELD4 *FC_P;
FIELD4 *FDistrito;
FIELD4 *FPrincipal;
FIELD4 *FSecundario;
FIELD4 *FZona;
FIELD4 *FSuspendido;
FIELD4 *FSinControl;
```

Después, se debe asignar a un descriptor tipo string:

```
FNum_Tel  = d4field(alcanfile,"NUM_TEL");
```

```
FClave_Pob= d4field(alcanfile,"CLAVE_POB");
FAlarma   = d4field(alcanfile,"ALARMA");
FGrupo    = d4field(alcanfile,"GRUPO");
FTipo_Apto= d4field(alcanfile,"TIPO_APTO");
FTipoAmb  = d4field(alcanfile,"TIPOAMB");
FLugar_Ins= d4field(alcanfile,"LUGAR_INS");
FFecha_Ins= d4field(alcanfile,"FECHA_INS");
FCalleyNum= d4field(alcanfile,"CALLEYNUM");
FColonia  = d4field(alcanfile,"COLONIA");
FC_P      = d4field(alcanfile,"C_P");
FDistrito = d4field(alcanfile,"DISTRITO");
FPrincipal= d4field(alcanfile,"PRINCIPAL");
FSecundario= d4field(alcanfile,"SECUNDARIO");
FZona     = d4field(alcanfile,"ZONA");
FSuspendido= d4field(alcanfile,"SUSPENDIDO");
FSinControl= d4field(alcanfile,"SINCONTROL");
```

Finalmente, se aplica la función **d4open** para abrir con los siguientes parámetros:

- CODEBASE que contiene los datos de inicialización del Manejador de Base de Datos.
- Nombre del archivo

Para nuestro caso:

```
DATA4 alcanfile=d4open(&codebase,NF_ALCANCIA)
```

Abrir un Archivo de Indices.

Una vez creado un archivo de índices, este se puede abrir, pero se deben declarar variables del CODEBASE tipo TAG4 que contienen los apuntadores al buffer de índices.

*TAG4 *Tele_Tag,*Zona_Tag;*

Después se debe asignar un descriptor tipo string:

Tele_Tag = d4tag(alcanfile,"TELE_TAG");
Zona_Tag = d4tag(alcanfile,"ZONA_TAG");

Finalmente se aplica la función **i4open** para abrir los índices, con los siguientes parámetros:

- Estructura DATA4 que contiene los datos del archivo de índices.
- Nombre del archivo de índices.

Para nuestro caso:

INDEX4 alcanindex=i4open(alcanfile,NX_ALCANCIA);

Copia de Buffer del Archivo de Datos a Memoria.

Una vez abiertas los archivos, se pueden asignar a las variables de memoria. En nuestro ejemplo, se utiliza la estructura en C llamada *data_buf1:*

```
typedef struct
{
        char sNum_Tel[S_NUM_TEL+1];
        char sClave_Pob[S_CLAVE_POB+1];
        char sAlarma[S_ALARMA+1];
        char sGrupo[S_GRUPO+1];
        char sTipo_Apto[S_TIPO_APTO+1];
        char sTipoAmb[S_TIPOAMB+1];
        char sLugar_Ins[S_LUGAR_INS+1];
        char sFecha_Ins[S_FECHA_INS+1];
        char sCalleyNum[S_CALLEYNUM+1];
        char sColonia[S_COLONIA+1];
        char sC_P[S_C_P+1];
        char sDistrito[S_DISTRITO+1];
        char sPrincipal[S_PRINCIPAL+1];
        char sSecundario[S_SECUNDARIO+1];
        char sZona[S_ZONA+1];
        int  iSuspendido;
        int  lSinControl;
} data_buf1;

f4ncpy(FNum_Tel,data_buf1.sNum_Tel,sizeof(data_buf1.sNum_Tel));
f4ncpy(FClave_Pob,data_buf1.sClave_Pob,sizeof(data_buf1.sClave_Pob));
f4ncpy(FAlarma,data_buf1.sAlarma,sizeof(data_buf1.sAlarma));
f4ncpy(FGrupo,data_buf1.sGrupo,sizeof(data_buf1.sGrupo));
f4ncpy(FTipo_Apto,data_buf1.sTipo_Apto,sizeof(data_buf1.sTipo_Apto));
f4ncpy(FTipoAmb,data_buf1.sTipoAmb,sizeof(data_buf1.sTipoAmb));
f4ncpy(FLugar_Ins,data_buf1.sLugar_Ins,sizeof(data_buf1.sLugar_ins));
```

```
f4ncpy(FFecha_Ins,data_buf1.sFecha_Ins,sizeof(data_buf1.sFecha_Ins));
f4ncpy(FCalleyNum,data_buf1.sCalleyNum,sizeof(data_buf1.sCalleyNum));
f4ncpy(FColonia,data_buf1.sColonia,sizeof(data_buf1.sColonia));
f4ncpy(FC_P,data_buf1.sC_P,sizeof(data_buf1.sC_P));
f4ncpy(FDistrito,data_buf1.sDistrito,sizeof(data_buf1.sDistrito));
f4ncpy(FPrincipal,data_buf1.sPrincipal,sizeof(data_buf1.sPrincipal));
f4ncpy(FSecundario,data_buf1.sSecundario,sizeof(data_buf1.sSecundario));
f4ncpy(FZona,data_buf1.sZona,sizeof(data_buf1.sZona));
        data_buf1.iSuspendido=f4int(FSuspendido);
        data_buf1.iSinControl=f4int(FSinControl);
```

Copia de Memoria al Buffer del Archivo de Datos.

Una vez abiertos los archivos, se pueden asignar las variables de memoria al archivo de datos. En nuestro ejemplo, se utiliza la estructura en C llamada *data_buf1*:

```
f4assign(FNum_Tel,data_buf1.sNum_Tel);
f4assign(FClave_Pob,data_buf1.sClave_Pob);
f4assign(FAlarma,data_buf1.sAlarma);
f4assign(FGrupo,data_buf1.sGrupo);
f4assign(FTipo_Apto,data_buf1.sTipo_Apto);
f4assign(FTipoAmb,data_buf1.sTipoAmb);
f4assign(FLugar_Ins,data_buf1.sLugar_Ins);
f4assign(FFecha_Ins,data_buf1.sFecha_Ins);
f4assign(FCalleyNum,data_buf1.sCalleyNum);
f4assign(FColonia,data_buf1.sColonia);
f4assign(FC_P,data_buf1.sC_P);
f4assign(FDistrito,data_buf1.sDistrito);
```

```
f4assign(FPrincipal,data_buf1.sPrincipal);
f4assign(FSecundario,data_buf1.sSecundario);
f4assign(FZona,data_buf1.sZona);
f4assign_int(FSuspendido,data_buf1.iSuspendido);
f4assign_int(FSinControl,data_buf1.iSinControl);
```

Otras Funciones.

Después de haber hecho las funciones anteriores, las demás son muy faciles de usar. Se mencionarán solo algunas:

- **d4close** Cerrar archivos.
- **d4skip** Saltar un registro.
- **d4top** Ir al inicio del archivo.
- **d4bottom** Ir al final del archivo.
-
-
- **etc.**

3.8. Diseño e Implementación de Cada uno de los Módulos de Presentacion en Cscape.

Pantallas de Introducción de Password.

Para protección del sistema, se cuenta con 2 niveles de usuarios:

- De superusuario, que permite todos los accesos al sistema.
- De operador, que solo permite consultas de operaciones a la central, a la base de datos del sistema, y configuración de funcionamiento.

La pantalla de entrada al sistema es como se muestra en la figura 3.1.5 - 1 en la cual debe introducirse un password con un máximo de 8 caracteres. Como en todo sistema de seguridad, no será reflejado en la pantalla, el password es sensible al contexto, lo que significa que hace distinción entre minúsculas y mayúsculas, por esto, siempre se debe escribir conforme fue dado de alta.

Fig. 3.8-1 Password de Acceso.

Pantallas de Menú.

Dentro del menú principal se tiene acceso a los principales módulos del sistema. En la figura 3.8 - 2, podemos ver las diferentes opciones que cuenta el

menú principal. Cada módulo a su vez tiene otro menú de acceso a las diferentes opciones

El menú de procesos (Figura 3.8 - 3) permite arrancar o reanudar los procesos de verificación masiva (opciones 1,2,3 del menú). Además permite entrar al menú donde el sistema proporciona una serie de facilidades adicionales (opciones 4 a 11 del menú de procesos) para consultar y cambiar categorías, suspender, reanudar, desbloquear, probar condiciones de la línea y consultar contadores.

El menú de reportes (Figura 3.8 - 4) permite accesar a otros menús donde se puede seleccionar el reporte deseado.

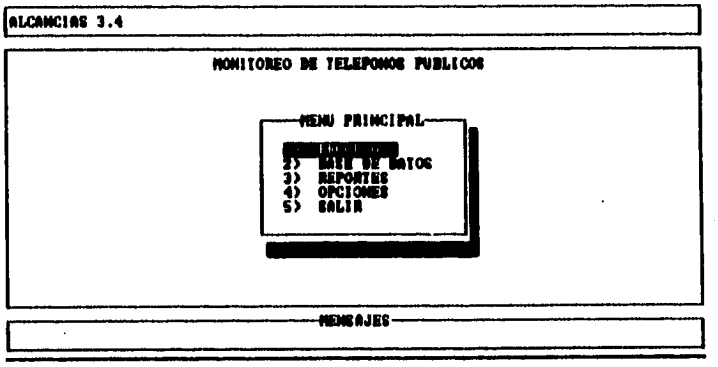


Fig. 3.8 - 2 Menú Principal.

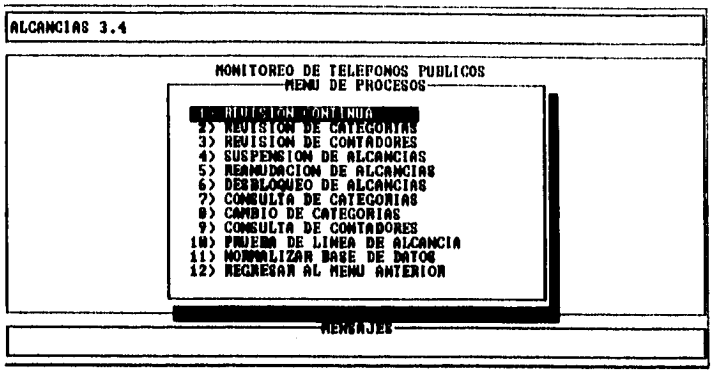


Fig 3.8 - 3 Menú de Procesos.

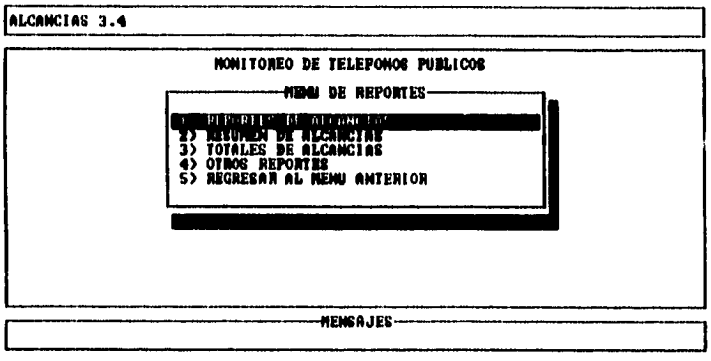


Fig. 3.8 - 4 Menú de Reportes.

El menú de opciones (figura 3.8 - 5) permite configurar el sistema en opciones como máscara de categorías por tipo de aparato, configuración de funcionamiento durante las revisiones masivas de líneas, conexión a las diferentes centrales telefónicas, claves de acceso, modems a accesar, cambio de passwords y respaldo de datos del sistema.

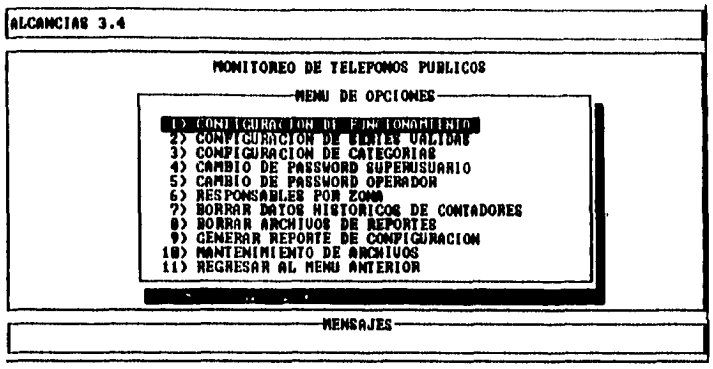


Fig. 3.8. - 5 Menú de Opciones.

El movimiento entre los diferentes menús, se realiza con las teclas (↑)(↓) y la selección del menú con (**ENTER**). Un acceso más rápido es seleccionando el número, seguido de un (**ENTER**).

Revisión Continua, Revisión de Categorías y Revisión de Contadores.

Al accesar cualquiera de estas opciones nos lleva a procesos que continuamente está checando las alcancías de la base de datos. En la figura 3.8 - 6 se muestra la pantalla de entrada común para estas tres opciones. En la primera opción donde dice Login, se selecciona la central, (este nombre debe estar de acuerdo como está declarado en el campo de login de la opción configuración de series válidas del menú de opciones), o si el usuario desea probar todas las centrales, hay que elegir el valor por default que es "TODAS". En la siguiente, dos opciones permiten elegir un rango de teléfonos a probar, o bien al elegir los valores por default para cada opción "0" y "99999999", probará todos los números.

Para moverse entre campos, se utilizan las teclas (↑)(↓) y para aceptar cada campo con un (**ENTER**), al dar el último (**ENTER**) comenzará el proceso de prueba.

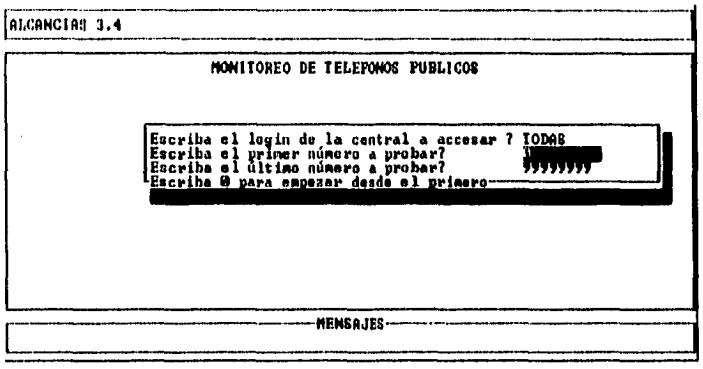


Fig. 3.8. - 6 Opciones de Prueba de Línea.

En la figura 3.8. - 7 se muestra en la parte inferior, el proceso de enlace con la central cuando esta es conmutada:

- Colgado, para asegurar que esté disponible la línea.
- Llamará al teléfono del modem de la central que corresponde al aparato telefónico a probar.
- Desplegará la velocidad de acceso a la central por ejemplo 9600, o bien indicará que el teléfono de acceso está ocupado.
- Finalmente comenzará a probar las líneas de manera sucesiva.

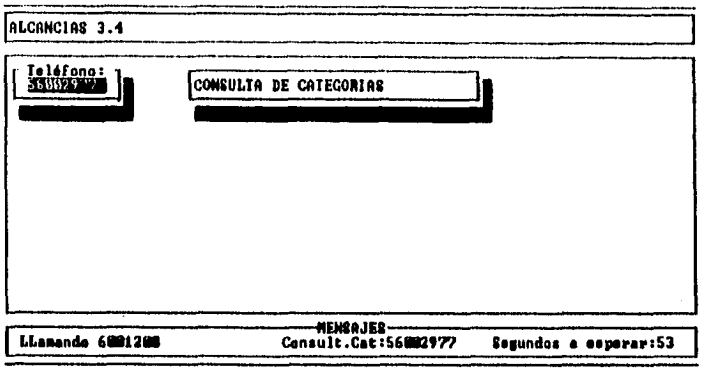


Fig. 3.8. - 7 Proceso de Enlace con la Central.

En la figura 3.8. - 8 y 3.8. - 9 se muestran la pantallas cuando el sistema esta revisando categorías de la central. El sistema estará comparando las categorías leídas con las que están declaradas en la base de datos; si el sistema detecta anomalía, automáticamente se encarga de ajustar parámetros, siempre y cuando este habilitada esta opción en el menú de opciones - configuración de funcionamiento.

Analizando con detenimiento la pantalla, podemos observar lo siguiente:

- En encabezado de la pantalla, mostrado en la parte superior aparece el nombre del sistema y su número de revisión.
- En el recuadro central de la pantalla se muestran los resultados que envía la central. Puede notarse una gran diferencia entre las pantallas 3.8. - 8 y 3.8. - 9 debido a que son de dos tipos de central distintas
- Un poco abajo aparecen los resultados de la prueba. Este recuadro sirve para identificar lo que el sistema esta reconociendo como lectura de la central, anotadas en variables que serán escritas en la base de datos.

También en este recuadro se puede ver la diferencia entre las pantallas de las figureas 3.8.- 8 y 3.8. - 9 debido a que son de dos centrales diferentes.

- Finalmente, en el recuadro de pie de página, abajo de la pantalla, se muestra el número a probar y el número probado en ese instante.

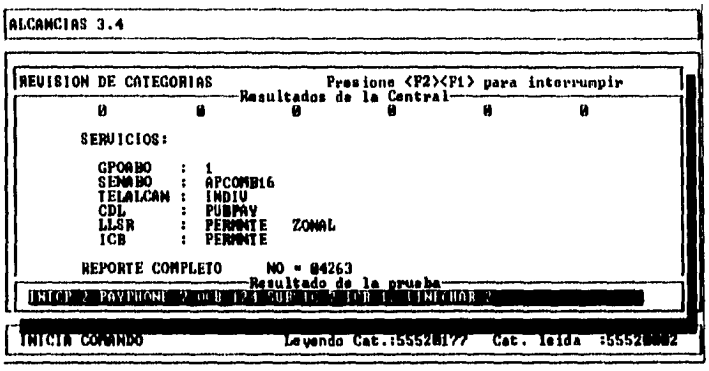


Fig. 3.8. - 8 Revisión de categorías (a).

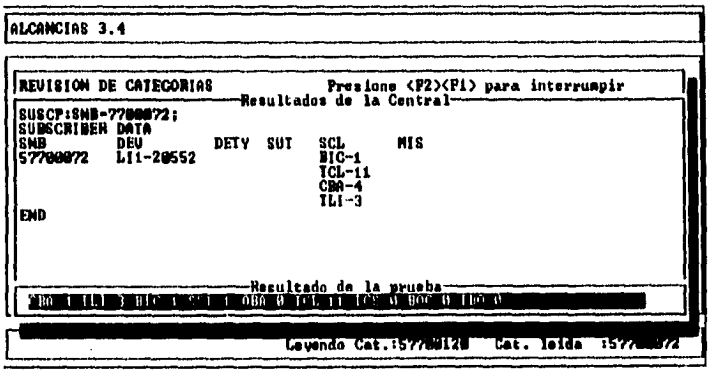


Fig. 3.8. - 9 Revisión de categorías (b).

En las figuras 3.8. - 10 y 3.8. - 11 se muestran las pantallas cuando el sistema está leyendo contadores de abonado. El principio de la pantalla es el mismo que el descrito en categorías, lo único que es importante considerar es que los

resultados de la prueba, se anotarán en la última lectura, y el valor anterior lectura contenida en la base de datos pasará a penúltima lectura.

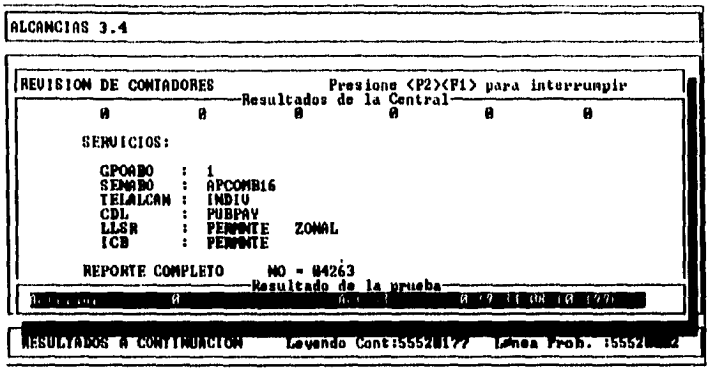


Fig. 3.8. - 10 Revisión de Contadores (a).

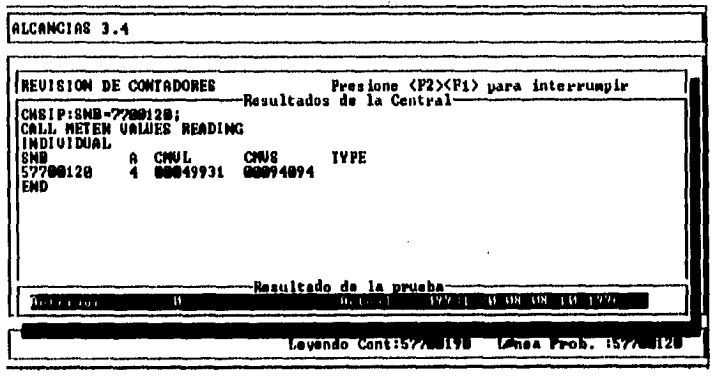


Fig. 3.8. - 11 Revisión de Contadores (b).

Suspensión/Reanudación de Alcancías.

Cuando se selecciona cualquiera de estas dos opciones del menú de procesos el sistema preguntará en secuencia los siguientes datos (ver figuras 3.8. - 12 y 3.8. - 13).

- El PASSWORD de superusuario.
- El número telefónico a suspender o reanudar
- Pregunta bajo que tipo de aparato (para ver este submenú teclear **BARRA ESPACIADORA**)se va a suspender. (Por nomenclatura, una categoría con * significa suspendido).

Cuando se de proporcionar los datos anteriores el sistema presentará unas pantallas como las mostradas en las figuras 3.8. - 14 y 3.8. - 15, en las cuales el sistema llena automáticamente unas categorías que sugiere para la suspensión y reanudación, aunque estos campos son modificables.

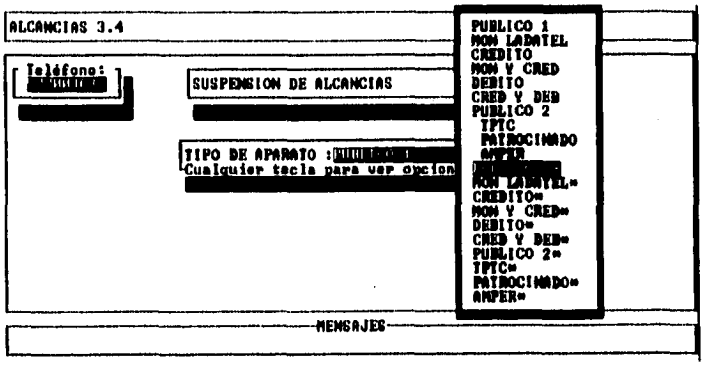


Fig. 3.8. - 12 Suspensión de Alcancías.

Cuando se está suspendiendo, se necesita sugerir uno de los 10 últimos tipos de aparatos con asterisco, o si se está suspendiendo, uno de los 10 primeros tipos de aparato, porque de otra manera, aparecerá un mensaje de error el cual indica que no se esta suspendiendo o reanudando.

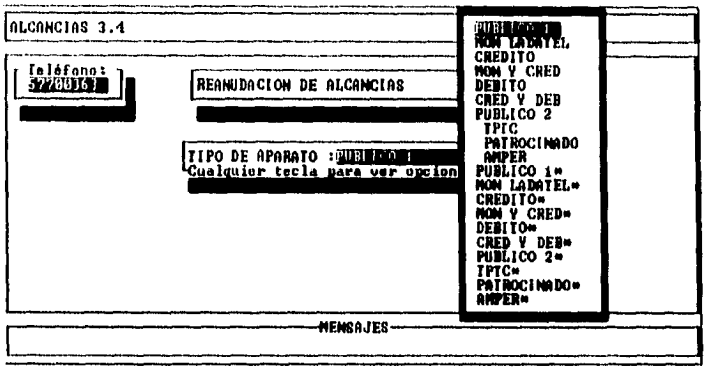


Fig. 3.8. - 13 Reanudación de Alcancías.

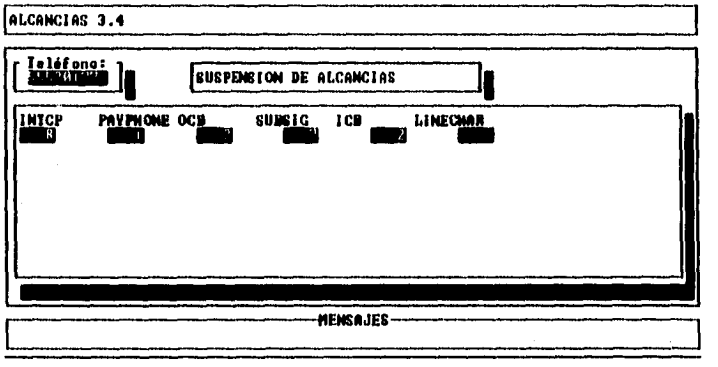


Fig. 3.8. - 14 Suspensión de Categorías, Sugerencia de Categorías.

Cuando el sistema termina el proceso de suspensión o reanudación, preguntará quien es el que efectuó el cambio (ver figuras 3.8. - 16 y 3.8. - 17), para escribirlo en un archivo de bitácora de cambios.

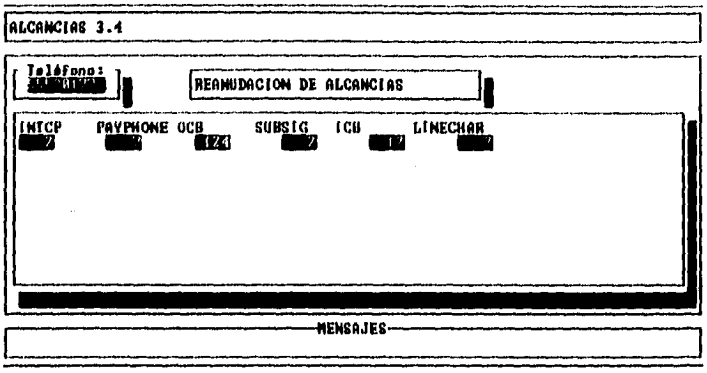


Fig. 3.8. - 15 Reanudación de Categorías, Sugerencia de Categorías.

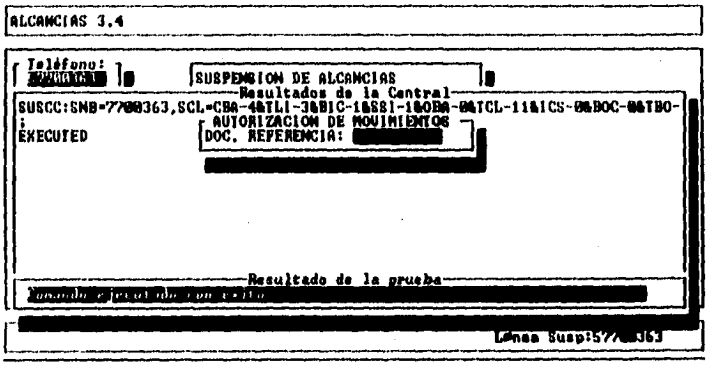


Fig. 3.8. - 16 Suspensión de Categorías, Proceso Concluido.

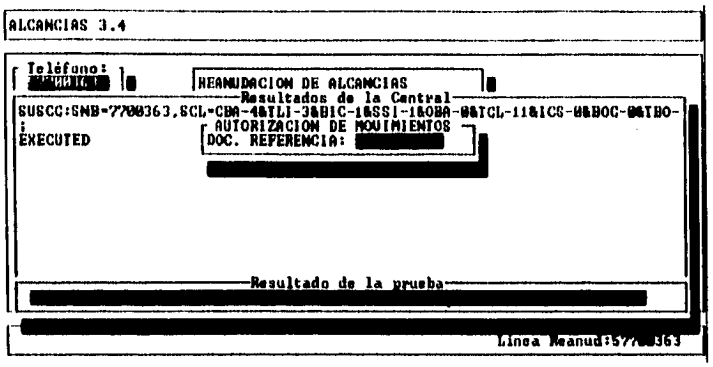


Fig. 3.8. - 17 Reanudación de Categorías, Proceso Concluido.

Desbloqueo de Alcancías

Para realizar un desbloqueo de alcancías, se selecciona esta opción con las flechas hacia arriba (↑) ó hacia abajo (↓) y se presiona (ENTER) para continuar. El sistema preguntará el número telefónico de la alcancía que por alguna razón de falla o por alguna orden se encuentre bloqueada (Ver Figura 3.8. - 18). Para desbloquearlo se teclea este número y se presiona (**ENTER**) o (**ESC**) para cancelar.

Consulta de Categorías y Contadores.

Una vez dado el PASSWORD DEL USUARIO el sistema preguntará el número telefónico de la alcancía de que la se desea consultar categorías o contador (ver figuras 3.8. - 19 y 3.8. - 20). El sistema desplegará las categorías existentes en el recuadro en la parte central y la interpretación de la respuesta en el recuadro central inferior.

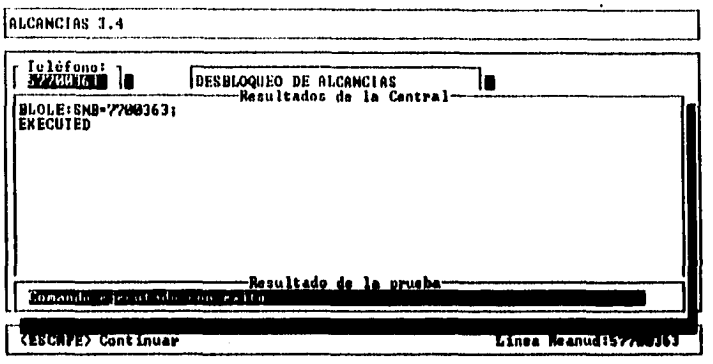


Fig. 3.8. - 18 Desbloqueo de Alcancías.

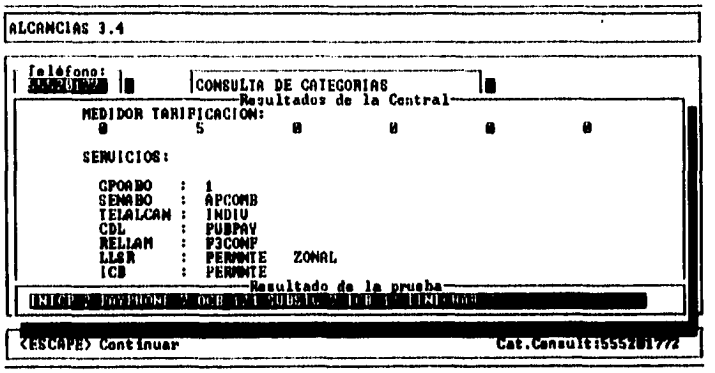


Fig. 3.8. - 19 Consulta de Categorías (a).

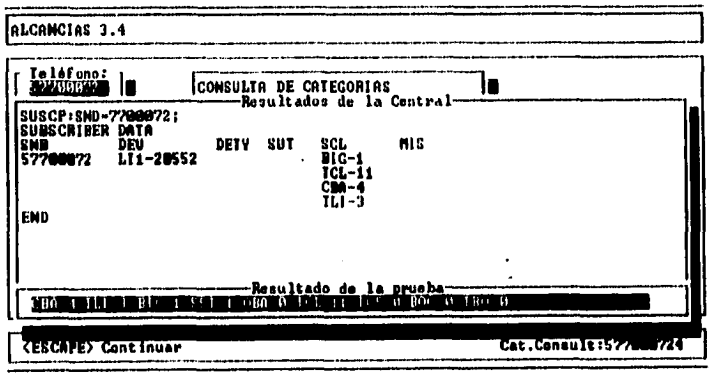


Fig. 3.8. - 20 Consulta de Categorías (b).

Cambio de Categorías.

Una vez dado el PASSWORD del usuario, el sistema preguntará el número telefónico de la Alcancía que se desea cambiar. Este podrá ser cualquiera de las siete categorías preestablecidas.

En caso de haber elegido la Opción 8) Cambio de Categorías del menú de Procesos, el sistema presentará la pantalla mostrada en las figuras 3.8. - 21 y 3.8. - 22, para poder modificar manualmente cualquiera de las categorías preestablecidas para el control de Alcancías, dependiendo del tipo de central.

Al momento de seleccionar el número telefónico, el sistema consultará si existe en la base de datos; en caso de existir, leerá las categorías que tiene actualmente el número seleccionado, ahí mismo se modifican las categorías que se deseen cambiar y se presiona (**ENTER**).

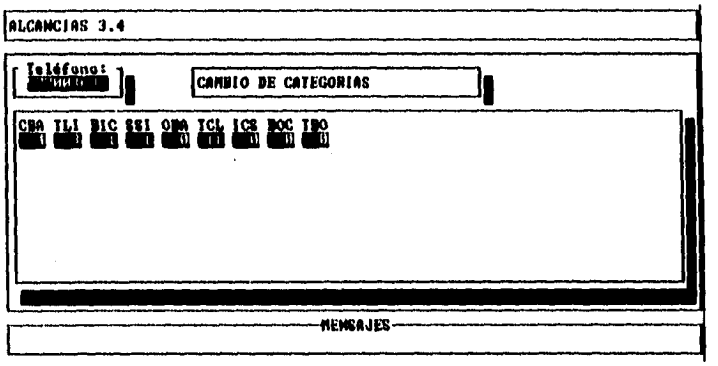


Fig. 3.8. - 21 Cambio de Categorías (a).

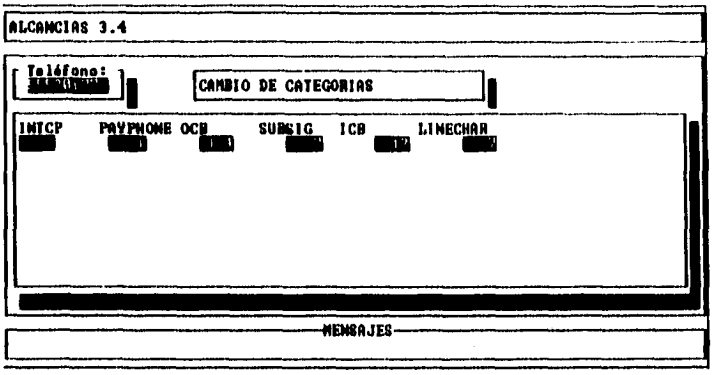


Fig. 3.8. - 22 Cambio de Categorías (b).

Prueba de Línea.

Una vez dado el PASSWORD DEL USUARIO, el sistema preguntará el número de línea a realizar la prueba de línea y se presiona (ENTER). El sistema desplegará el resultado en un recuadro central (Ver Figuras 3.8. - 23 y 3.8. - 24), y la interpretación de esté resultado acorde al tipo de fallas presentados en el manual de usuario.

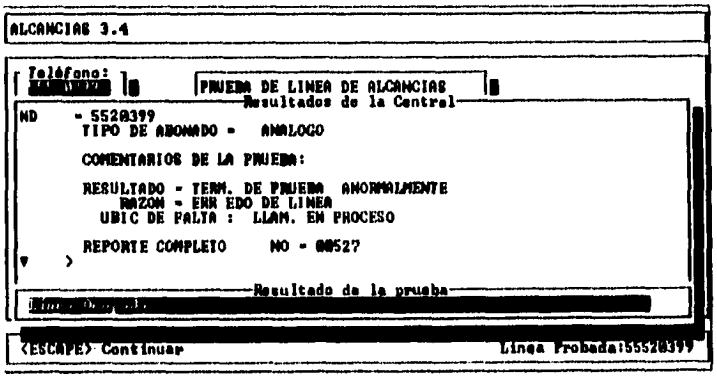


Fig. 3.8. - 23 Prueba de Línea (a).

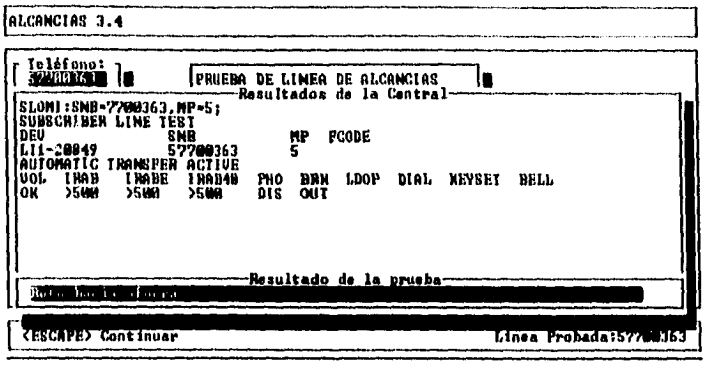


Fig. 3.8. - 24 Prueba de Línea (b).

Base de Datos.

Permite generar y actualizar la base de datos con la cual el sistema trabaja.

NOTA: En Altas, Bajas, y Cambios. Cuando se termina de realizar cualquiera de los 3 movimientos, el sistema despliega una pantalla en la que pide el número del documento de referencia al presionar (**ENTER**) regresa a la pantalla de opciones.

Altas.

La opción ALTAS carga el registro de una Alcancía nueva. Al terminar de dar de alta una alcancía, aparece una pantalla que le ofrece normalizar sus categorías por tipo de aparato. Presionando la barra espaciadora usted podrá elegir SI o NO, después de hacer su elección presione ENTER y el sistema lo regresa a la pantalla de opción (figura 3.8. - 25).

Bajas.

Permite eliminar el registro de una alcancía del archivo. Cabe mencionar que bajas se tomaran en cuenta una vez que se salga de este menú de base de datos.

Cambios.

Si se requiere modificar algún dato del registro asociado al teléfono de alcancía. Al colocarse en la casilla de TIPO DE APARATO, presione barra espaciadora y se desplegará una pantalla donde se podrá elegir con fecha hacia arriba (↑) o hacia abajo (↓) el tipo de aparato deseado y con (**ENTER**) se ejecutara el cambio. Al terminar de realizar el cambio de TIPO DE APARATO, el sistema desplegará una pantalla que le ofrece normalizar sus categorías. Con barra espaciadora usted elegirá SI o NO , después de hacer su elección, presione **ENTER** y regresará a la pantalla de bases de datos (figura 3.8. - 25).

NOTA: Siguiente, Previo, Inicio Y Fin: Permiten moverse dentro del archivo de alcancías.

Llave.

Permite cambiar el número telefónico cuando una alcancía originalmente conectada a una central analógica se reconecta a una central digital, o cuando el número asociado a un aparato cambia.

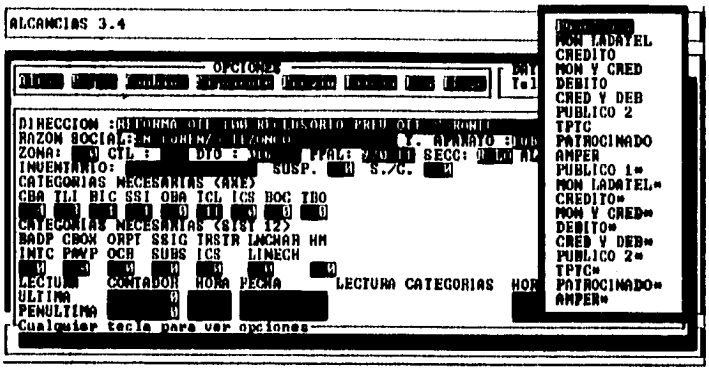


Fig. 3.8. - 25 Pantalla de Bases de Datos.

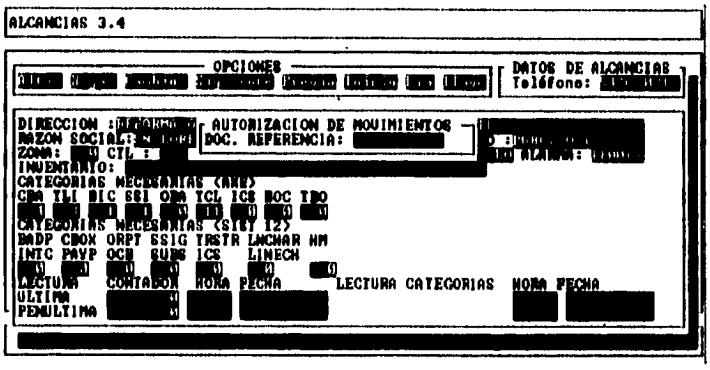


Fig. 3.8. - 26 Pantalla de Bases de Datos (Capturando Referencia).

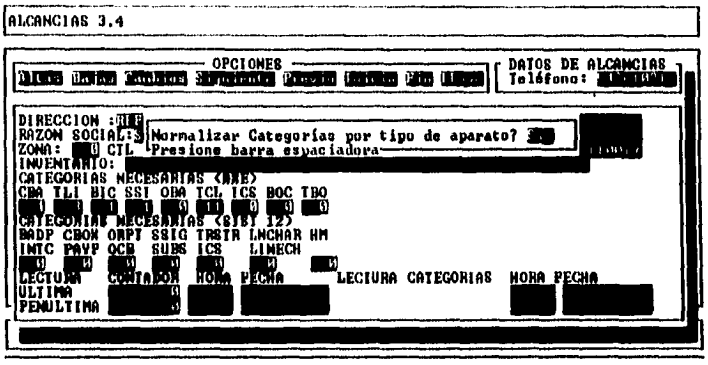


Fig. 3.8. - 27 Pantalla de Bases de Datos (Normalizando Categorías).

Reportes.

Proporciona los resultados de las pruebas de acuerdo a las diferentes opciones que se proporcionan.

Una vez seleccionada cualquier opción de reportes, dado que todos estos funcionan de manera similar se explicará de manera global como es que funcionan estas pantallas.

La primera pantalla que aparece al generar reportes es como la mostrada en la figura 3.8. - 28, pudiéndose elegir lo siguiente:

- Mandar el reporte a pantalla: Con esta opción se pueden observar los detalles de la impresión página por página; utilizando (**BARRA ESPACIADORA**) para avanzar solo una o bien (**T**) para terminar de observar el reporte.
- Mandar el reporte a la impresora. Con esta opción el sistema mandará el reporte tipo texto, por lo que no es necesario tener drivers para configurar la impresora.

- Mandar el reporte a un editor de textos. Se puede elegir para observar el reporte, un manejador de textos de preferencia, el único requisito para habilitarlo es renombrarlo a EXECEDIT.EXE

En la figura 3.8. - 29 el sistema proporciona la opción para salvar aquellos casos en los que se han hecho cambios a la base de datos y se pretende asegurar que la edición de resultados se actualice. En caso de no haberse hecho, se debe evitar, ya que el sistema reelaborará el reporte innecesariamente.

La alternativa de reelaboración o no, se hace con la (**BARRA ESPACIADORA**)

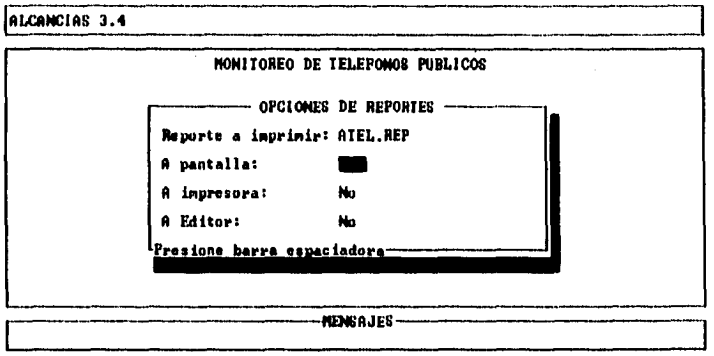


Fig. 3.8. - 28 Opciones de Reporte.

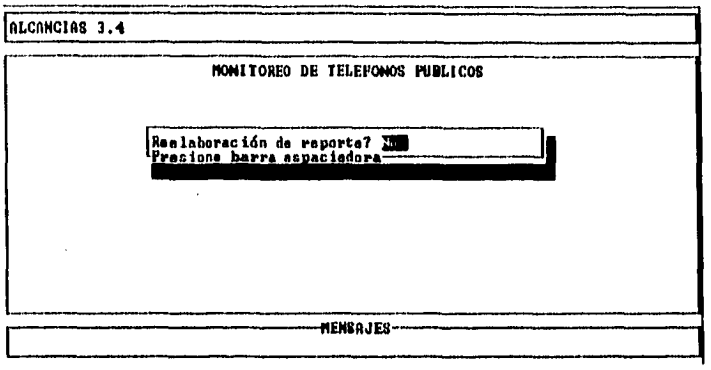


Fig. 3.8. - 29 Reelaboración de Reportes.

En caso que se deseara mandar una impresión a pantalla, el sistema mostrará el reporte como se muestra en la figura 3.8. - 30.

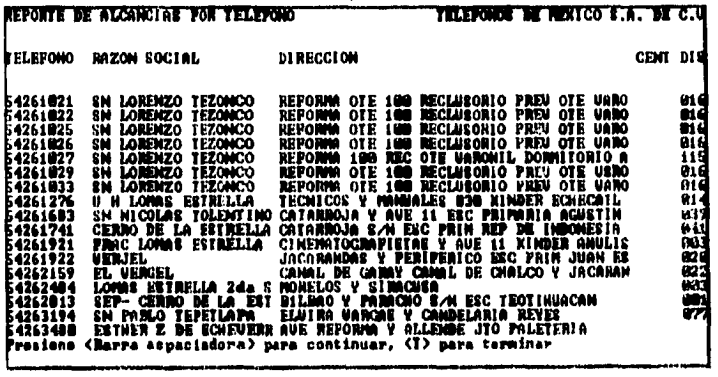


Fig. 3.8. - 30 Reportes a Pantalla.

Alcancías por Teléfono.

Despliega la base de datos en orden, de acuerdo al número telefónico.

Alcancías por Zona.

Despliega la base de datos de acuerdo a la clasificación por zonas que fueron capturadas en la Base de Datos.

Alcancías por Tipo de Aparato.

Se despliega la base de datos de acuerdo al tipo de aparato registrado; multitarjeta, convencional, etc.

Alcancías, una Zona.

Despliega sólo los teléfonos que se encuentren en una zona que solicite el usuario.

Cuando se selecciona la opción de alcancías una zona, se selecciona la zona y enseguida se presiona (**ENTER**), para continuar.

Alcancías, un Tipo de Aparato.

Despliega sólo los teléfonos que sean del tipo de aparato que solicite el usuario.

Cuando se selecciona la opción de alcancías por tipo de aparato que desea verificarse, se selecciona el tipo de aparato, primero oprimiendo la [**BARRA ESPACIADORA**] para que nos muestre el submenú de opciones de TIPO DE APARATO; se selecciona con (↑)(↓), luego (**ENTER**) y después, nuevamente (**ENTER**) para continuar.

Alcancías con Posible Falla.

Esta opción permite mostrar únicamente los datos de alcancías que de acuerdo con el NO MOVIMIENTO del contador detectado durante la verificación y secuencia simplemente tienen falla.

Alcancías con Posible Falla, Una Zona.

Muestra el resultado de la verificación filtrando las alcancías con posible falla, de una zona a la vez.

Alcancías con Cambio de Categorías.

Nos muestra un reporte de todos los movimientos de categorías efectuado en las alcancías.

Resumen por Teléfono.

Despliega un resumen de Alcancías ordenadas por el número telefónico correspondiente.

Resumen por Zona.

Hace un resumen de Alcancías por ordenamiento de la Zona correspondiente.

Resumen por Tipo de Aparato.

Muestra un ordenamiento de Alcancías de acuerdo al tipo de aparato, que existen.

Resumen con Posible Falla.

Despliega un condensado de las Alcancías que han tenido movimientos de categorías.

Resumen con Cambio de Categorías.

Nos muestra un condensado de las Alcancías que han tenido movimientos de categorías.

Resumen de Suspensiones.

Resumen de suspensiones muestra un desplegado resumido de las Alcancías suspendidas.

Totales de Alcancías por Tipo de Aparato.

El sistema despliega un reporte en el que especifica la cantidad existente de alcancías por tipo de aparato.

Totales de Alcancías con Posible Falla.

El sistema despliega los aparatos con cambio de categoría y sin movimiento de contadores en base al número de cambio de categorías y aparatos sin movimientos de contadores. El sistema presenta el volumen de eficiencia relativa existente en la ciudad.

Revisión de Contadores por Fecha.

Primeramente al elegir esta opción el sistema le pedirá el rango de evaluación de contadores en el cual usted pondrá la fecha inicial y final de la evaluación. El sistema desplegara un reporte de contadores de Alcancías por fecha.

Histórico de Movimientos.

Se despliega un historial de todos los movimientos efectuados como Altas, Bajas, Cambios, Cambios de categorías, Suspensión, Reanudación, Cambios de Passwords, etc.

Histórico de Respuestas de la Central.

Sirve para cotejar que los comandos que se estén realizando, la central los este respondiendo, se recomienda primeramente, activar el BUFFER DE CENTRALES en configuración de funcionamiento del menú de opciones, y observarlo en un período corto de tiempo y esporádicamente ya que ocupa una gran cantidad de memoria.

Reporte de Configuración.

Esta opción sirve para proporcionar al usuario información técnica del archivo config.dat

Configuración de Funcionamiento.

En esta opción se pueden declarar los parámetros generales de la red: numeración existente, tipo de central, logín de la central digital, etc.

Una vez seleccionada la alternativa de configuración general, dentro del menú de opciones, el sistema requerirá el password de entrada para modificar los parámetros de operación con los cuales el Sistema operará en lo sucesivo. Después de la introducción del password correcto el sistema mostrará la pantalla de Datos Generales tal como se muestra en la figura 3.8. - 31.

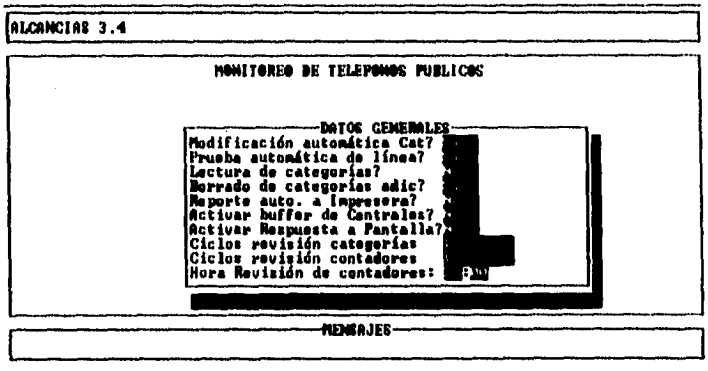


Fig. 3.8. - 33 Datos Generales.

- MODIFICACIÓN AUTOMÁTICA DE CATEGORÍAS: Esta alternativa permite, si se desea, que el sistema ajuste las categorías de los teléfonos verificados de acuerdo a la imagen en memoria que el sistema tiene del mismo. Puede modificarse SI/NO usando [**BARRA ESPACIADORA**].
- PRUEBA AUTOMÁTICA DE LÍNEA: Si se desea el sistema puede ordenar la prueba de la línea de abonado que proporciona la central digital cuando se detecta una alcancía con posible falla. Puede modificarse SI/NO usando [**BARRA ESPACIADORA**].
- LECTURA DE CATEGORÍAS: Esta alternativa si se solicita, permite inicializar los datos de las categorías de los teléfonos. Normalmente solo se utiliza al inicializar el sistema. Puede modificarse SI/NO usando [**BARRA ESPACIADORA**].
- BORRADO DE CATEGORÍAS ADICIONALES: Esta alternativa permite que al momento, que se encuentre ejecutándose el monitoreo, el sistema borre cualquier categoría que no se encuentre dentro de las categorías de control por ejemplo (CBA,TLI,BIC,SSI,OBA,TCL E ICS). Puede modificarse SI/NO usando [**BARRA ESPACIADORA**].
- REPORTE AUTOMÁTICO A IMPRESORAS: Cuando el sistema está en el proceso de monitoreo, se puede elegir si desea, que se imprima automáticamente cualquier anomalía que se presente. Si elige SI, empezará a imprimirlas en forma de tiras si elige NO, no procederá a imprimir. Puede modificarse SI/NO usando [**BARRA ESPACIADORA**].
- ACTIVAR BUFFER DE CENTRALES: En esta opción, se activa una bandera para poder supervisar que los comandos que se estén realizando, la central los este respondiendo Puede modificarse SI/NO usando [**BARRA ESPACIADORA**].
- CICLO REVISIÓN CATEGORÍAS: Esta opción le indica al sistema hasta cuántos ciclos de revisión de categorías completas a la base de datos se deben realizar antes de regresar al Menú Principal cuando se selecciona la opción de Procesos en Revisión de Categorías. Puede modificarse tecleando el

número de ciclos requerido. [Puede modificarse SI/NO usando [**BARRA ESPACIADORA**]. (opción ya obsoleta)

- CICLO REVISIÓN CONTADORES: Esta opción le indica al sistema hasta cuántos ciclos de revisión de contadores completas a la base de datos se deben realizar antes de regresar al Menú Principal cuando se selecciona la opción de Procesos en Revisión de Contadores. Puede modificarse tecleando el número de ciclos requerido y oprimiendo [**ENTER**] (opción ya obsoleta).
- HORA DE REVISIÓN DE CONTADORES: Esta opción permite modificar la hora en que se activará la revisión de contadores al día, registrando la lectura de contador para una posterior comparación de lecturas que determinan si han estado funcionando los teléfonos públicos. Puede modificarse tecleando las horas + [**ENTER**] en formato de 24Hrs y luego los minutos + [**ENTER**].

Configuración de Series Válidas.

Permite actualizar la información de la numeración asociada a centrales digitales disponibles en la ciudad. Puesto que el sistema hace una validación automática del número a probar, se debe estar actualizando esta información en función de la ampliación de líneas y sustitución de centrales. El número de serie no se puede declarar aleatoriamente (ver figura 3.8. - 32), puesto que deben ocupar los primeros lugares de la tabla, solo así se puede asegurar que están incluidas en su totalidad. Existen hasta 110 series que se pueden dar de alta.

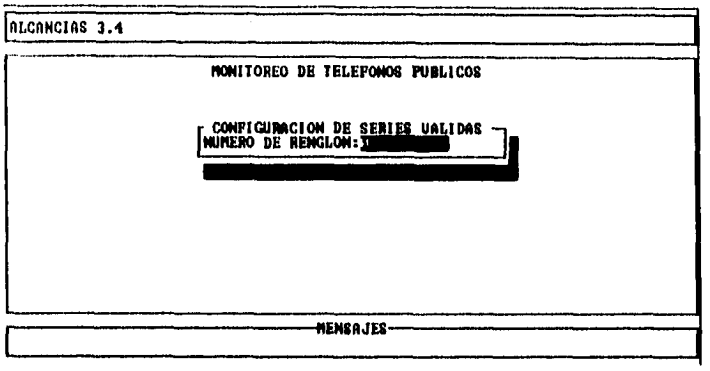


Fig. 3.8. - 32 Selección del Renglón en Configuración de Series Válidas.

Una vez seleccionada esta opción, aparecerá la pantalla, mostrada en la figura 3.8. - 33. El movimiento entre las opciones es con (**ENTER**) y salida con (**ESC**).

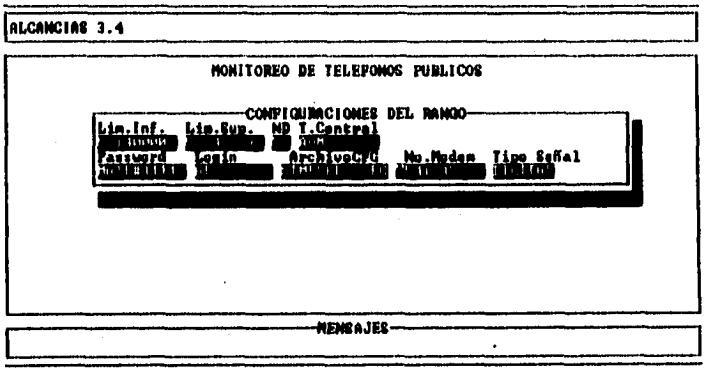


Fig. 3.8. - 33 Configuración de Rangos.

- LIM.SUP-LIM.INF: Son los Límites Superiores e Inferiores en los que se encuentran las numeraciones telefónicas para cada una de las Centrales existentes. Se modifica tecleando los número de los límites requeridos seguido de [**ENTER**].

- TIPO: Se debe especificar aquí si la Central es Analógica o Digital. Se modifica oprimiendo [**BARRA ESPACIADORA**], se selecciona con (↑)(↓), luego [**ENTER**] y otra vez [**ENTER**] para pasar a la siguiente opción.
- ND: Se especifica el número de Dígitos manejado por la Central. Se modifica tecleando el número de Dígitos requerido seguido de [**ENTER**].
- T. DE CENTRAL: Se le debe enterar al Sistema el Tipo de Central: MOA, SISTEMA2199, IOC12 o IOC14. Se modifica oprimiendo [**BARRA ESPACIADORA**], se selecciona con (↑)(↓), luego [**ENTER**] y otra vez [**ENTER**] para pasar a la siguiente opción.
- PASSWORD: Con esta alternativa se puede modificar el password requerido para accesar el Sistema de la Central (MOA, SIST2199, etc.). Se modifica tecleando el nuevo Password y luego [**ENTER**].
- LOGIN: Es necesario informar al Sistema como se le conoce a la central por su Sistema interno (MOA, SIST2199, etc). Se modifica tecleando el nuevo LOGIN requerido seguido de [**ENTER**].
- PUERTO: Se refiere al puerto de Comunicaciones por el que se comunica la PC del Sistema de Alcancías al Modem de comunicación. Se modifica oprimiendo [**BARRA ESPACIADORA**], se selecciona con (↑)(↓), luego [**ENTER**] y otra vez [**ENTER**] para pasar a la siguiente opción. .
- No. MODEM: Es el número telefónico del modem del Sistema Interno de la Central. Se modifica tecleando un nuevo número telefónico seguido de [**ENTER**].
- Oprimiendo otra vez [**ENTER**] el Sistema nos da la opción para registrar otra Serie. Si no hay otra Serie para dar de alta o modificar, con oprimir [**ESC**] se vuelve al menú de opciones.

Configuración de Categorías.

Al elegir esta opción el sistema le pedirá un Password de entrada después le mostrara un cuadro de configuración de categorías en el que se dan de alta las categorías necesarias (ver figura 3.8. - 34) por cada tipo de aparato incluyendo aparatos suspendidos distinguiéndose con una (**S**) al final y dependerá del usuario explotar esta norma al momento de dar de alta en la base de datos y normalizando categorías.

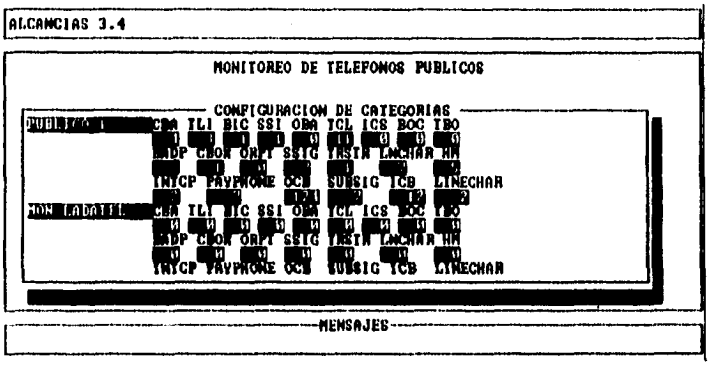


Fig. 3.8. - 34 Configuración de Categorías.

Cambio de Password Usuario.

Permite, en todo momento, cambiar el password de entrada a Alcancías y a los procesos de Altas, Bajas y Cambios de la Base de Datos a los Usuarios del Sistema. Los poseedores de estos passwords tienen todos los privilegios para administrar la utilización y mantenimiento del sistema en general. En la figura 3.8. -35 se muestra la manera de cambiar el Password del Usuario. Esta misma

pantalla aparece para cambiar el Password del Operador. El password consiste en la introducción de ocho caracteres alfanuméricos seguidos de la tecla [**ENTER**].

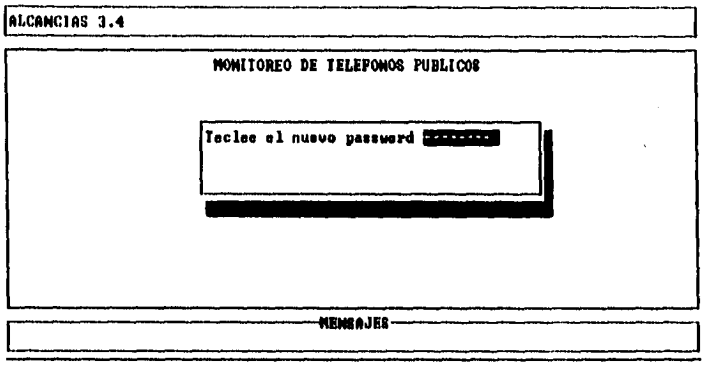


Fig. 3.8. - 35 Cambio de Password.

Borrar Archivos de Reportes.

Debido a que los resultados de las pruebas quedan residentes en disco, esta opción permite limpiar el disco duro de aquellos reportes que ya fueron procesados.

Generar Reporte de Configuración.

Sirve para que el usuario este enterado del status de las variables del sistema.

Mantenimiento de Archivos.

Esta opción sirve para que el operador respalde o recupere Archivos de Datos y Configuración del Sistema. Al momento de elegir MANTENIMIENTO DE ARCHIVOS presione (**ENTER**) y desplegará un submenú. ver figura 3.8. - 36.

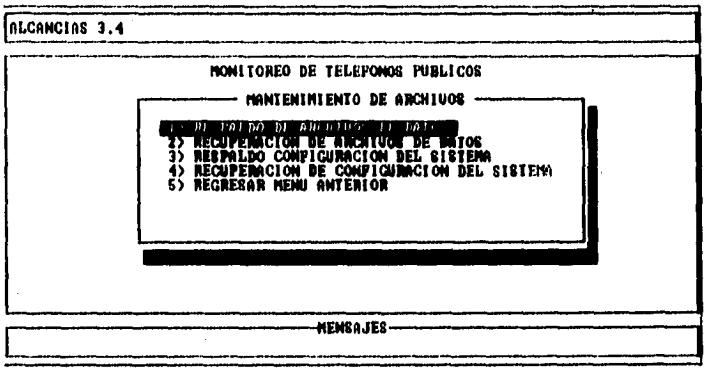


Fig. 3.8. - 36 Mantenimiento de Archivos.

- RESPALDO DE ARCHIVOS DE DATOS.

  Esta opción permite respaldar todos lo archivos .DBF y .CDX de la base de datos de Alcancías a un disco flexible. Con el fin de poder restaurar el sistema en caso de que haya una falla de Hardware.

- RECUPERACIÓN DE ARCHIVOS DE DATOS.

  Esta opción nos permite jalar los archivos .DBF y .CDX de la fuente de respaldo al directorio de trabajo siempre y cuando se hayan respaldado previamente en un disco flexible.

- RESPALDO CONFIGURACIÓN DEL SISTEMA.
  Esta opción nos permite respaldar todos los archivos .DAT de configuración del sistema de Alcancías a un disco flexible (etiquetándolo correctamente para no confundir los archivos de datos con los de configuración del sistema) con el fin de poder restaurar los Archivos de configuración del sistema.

- RECUPERACIÓN DE CONFIGURACIÓN DEL SISTEMA.
  Esta opción nos permite jalar los archivos .DAT de la fuente de respaldo al directorio de trabajo siempre y cuando se tenga respaldada la información o se tenga los discos de instalación del sistemas.

Implementación de uno de los módulos en C-SCAPE

A continuación se presentará una explicación de la codificación del Manejador de Pantallas C-SCAPE para lo cual se seleccionaron 2 pantallas.

Manejos de Menús.

Para el manejo de pantallas en C-SCAPE se requiere de dos variables principales:

- Variable tipo sed_type que contiene y maneja la información de una pantalla (este tipo de variable es creada mediante una variable menu_type).
- Variable tipo menu_type sirve para describir que texto, donde se encuentra y los campos que deben aparecer.

En el ejemplo que presentamos es la pantalla del Menú Principal, en la cual se muestra el código para describir a detalle cada una de las funciones:

```
int iMENUPRIN() {
        menu_type   menu;
        sed_type sed;
        int ret;

        menu = menu_Open();
        menu_Printf(menu, "@p[1,2]@f[1)  PROCESOS ]", NULL, &menu_funcs);
        menu_Printf(menu,"@p[2,2]@f[2)  BASE DE DATOS ]",NULL, &menu_funcs);
        menu_Printf(menu, "@p[3,2]@f[3)  REPORTES ]",NULL, &menu_funcs);
        menu_Printf(menu, "@p[4,2]@f[4)  OPCIONES ]",NULL, &menu_funcs);
        menu_Printf(menu, "@p[5,2]@f[5)  SALIR ]",   NULL, &menu_funcs);
        menu_Flush(menu);
        sed = sed_Open(menu));
        sed_SetColors(sed, 0x4b, 0x4b, 0x2b);
        sed_SetBorder(sed, bd_prompt);
        sed_SetBorderTitle(sed, "MENU PRINCIPAL");
        sed_SetPosition(sed, 8, 29);
        sed_SetHeight(sed, 7);
        sed_SetWidth(sed, 22);
        sed_SetShadow(sed, 1);
        sed_Repaint(sed);
        ret = sed_Go(sed);
        sed_Close(sed);
        return(ret);
}
```

Después de la declaración de variables, se muestran las siguientes funciones:

- **Menu_Open** Aloja el espacio para un objeto llamado *menú*.
- **Menu_Printf** Función que le dice al objeto *menú* cual va a ser su contenido. Esta función es de particular interés puesto que de ella se derivan validación de campos, posiciones en pantalla, cadenas repetidas, etc. El campo de validación *menu_Func* sirve para armar un menú de tipo descolgante como lo es la pantalla de Menú Principal.
- **Menu_Flush** Esta función indica cual va a ser la pantalla definitiva y libera memoria.
- **Sed_Open** Aloja el espacio para un objeto llamado *sed*.
- **Sed_SetColors(,,,)**
- **Sed_SetBorder(,);**
- **Sed_SetBorderTitle(,)**
- **Sed_SetPosition(,,)**
- **Sed_SetHeight(,)**
- **Sed_SetWidth(,)**
- **Sed_SetShadow(,).**

  Estas son una serie de las tantas opciones que C-SCAPE brinda para configurar la pantalla que se va a presentar, por ejemplo: color, tipo de borde, título en el borde, posición de la pantalla, ancho, largo y sombra.

- **Sed_Repaint()**. Sirve para refrescar la pantalla cada vez que se le cita.
- **Ret = sed_Go()**. Esta función sirve para manipular la pantalla.
- **Sed_Close()**. Esta función para cerrar y deshabilitar la pantalla.

A continuación se presenta el código que funciona de manera similar al explicado anteriormente, cuyo objeto es permanecer en el fondo de manera

estática durante toda la sesión de trabajo y sirve para poner el encabezado, la parte media y el pie de página:

```
sed_type sENCAB1sMEDIO1s_PIE1;
Int iENCAB1() {
        menu_type  menu;
        int             ret;
        if ((menu = menu_Open())==NULL)
                generror(6,0);
        menu_Printf(menu, "ALCANCIAS 3.4");
        menu_Flush(menu);
        if ((sENCAB1 = sed_Open(menu))==NULL)
                generror(7,0);
        sed_SetColors(sENCAB1, 0x1b, 0x1b, 0x1b);
        sed_SetBorder(sENCAB1, bd_prompt);
        sed_SetPosition(sENCAB1, 0, 0);
        sed_SetHeight(sENCAB1, 1);
        sed_SetWidth(sENCAB1, 78);
        sed_Repaint(sENCAB1);
        ret = sed_Go(sENCAB1);
        return(ret);
}

int iMEDIO1() {
  menu_type       menu;
  int       ret;

        if ((menu = menu_Open())==NULL)
```

```
            generror(6,0);
        menu_Printf(menu, "@[23, ]MONITOREO DE TELEFONOS PUBLICOS\n");
        menu_Flush(menu);
        if ((sMEDIO1 = sed_Open(menu))==NULL)
            generror(7,0);
        sed_SetColors(sMEDIO1, 0x1b, 0x1b, 0x1b);
        sed_SetBorder(sMEDIO1, bd_prompt);
        sed_SetPosition(sMEDIO1, 3, 0);
        sed_SetHeight(sMEDIO1, 17);
        sed_SetWidth(sMEDIO1, 78);
        sed_Repaint(sMEDIO1);
        ret = sed_Go(sMEDIO1);
        return(ret);


int iPIE1() {
 menu_type      menu;
 int            ret;

        if ((menu = menu_Open())==NULL)
            generror(6,0);
        menu_Flush(menu);
        if ((sPIE1 = sed_Open(menu))==NULL)
            generror(7,0);
        sed_SetColors(sPIE1, 0x1b, 0x1b, 0x1c);
        sed_SetBorder(sPIE1, bd_prompt);
        sed_SetBorderTitle(sPIE1, "MENSAJES");
        sed_SetPosition(sPIE1, 22, 0);
        sed_SetHeight(sPIE1, 1);
```

```
    sed_SetWidth(sPIE1, 78);
    sed_Repaint(sPIE1);
    ret = sed_Go(sPIE1);
    return(ret);
}
```

Manejo de Pantallas de Captura.

A continuación se explicará la pantalla de captura de Altas, Bajas y Cambios del archivo TEL_PUB.DBF. Y debido a que el funcionamiento es similar, únicamente explicaremos el objeto de cada menú y diferencias entre la pantalla anterior. Cabe mencionar que se utilizan pantallas estáticas de encabezado, parte media y pie, descritas en el apartado anterior.

A continuación se presenta un menú horizontal para seleccionar la operación deseada. Esta rutina entregará en una variable la opción escogida por el usuario.

```
void initsedopera()
{
    menu_type menu;

    if ((menu = menu_Open())==NULL)
        generror(6,0);
    menu_Printf(menu, "@f[Altas] ",NULL, &menu_funcs);
    menu_Printf(menu, "@f[Bajas] ",NULL, &menu_funcs);
    menu_Printf(menu, "@f[Cambios] ",NULL, &menu_funcs);
    menu_Printf(menu, "@f[Siguiente] ",NULL, &menu_funcs);
    menu_Printf(menu, "@f[Previo] ",NULL, &menu_funcs);
```

```
        menu_Printf(menu, "@f[Inicio] ",NULL, &menu_funcs);
        menu_Printf(menu, "@f[Fin] ", NULL, &menu_funcs);
        menu_Printf(menu, "@f[LLave]",NULL, &menu_funcs);
//      menu_Printf(menu, "@f[]",NULL, &menu_funcs);

        menu_Flush(menu);
        if((sedopera = sed_Open(menu))==NULL)
                generror(7,0);
        sed_SetColors(sedopera, 0x1b, 0x31, 0x70);
        sed_SetBorder(sedopera, bd_prompt);
        sed_SetBorderTitle(sedopera, " OPCIONES ");
        sed_SetPosition(sedopera, 4, 1);
        sed_SetHeight(sedopera, 1);
        sed_SetWidth(sedopera, 53);
        sed_SetShadow(sedopera,1);
        sed_SetSpecial(sedopera,myspc_Esc2);
        sed_Repaint(sedopera);
}
```

Con esta pantalla se selecciona el teléfono a consultar, dar de alta, baja, etc. Se puede notar que tiene una función de validación numérica llamado *digit_funcs*.

```
void initsedcveal()
{
        menu_type menu;

        if ((menu = menu_Open())==NULL)
```

```
            generror(6,0);
        menu_Printf(menu, " Tel,fono:");
        menu_Printf(menu,"@p[0,12]@f[@[9,#]]",data_buf1.sNum_Tel,
            &digit_funcs);
        menu_Flush(menu);
        if ((sedcveal = sed_Open(menu))==NULL)
            generror(7,0);
        sed_SetColors(sedcveal, 0x1b, 0x31, 0x70);
        sed_SetBorder(sedcveal, bd_prompt);
        sed_SetBorderTitle(sedcveal, " DATOS DE ALCANCIAS ");
        sed_SetPosition(sedcveal, 4, 56);
        sed_SetHeight(sedcveal, 1);
        sed_SetWidth(sedcveal, 20);
        sed_SetShadow(sedcveal,1);
        sed_SetSpecial(sedcveal,myspc_Esc);
        sed_Repaint(sedcveal);
}
```

Aquí se realiza el cambio de la llave desplegando una ventana para la captura del nuevo número telefónico y el funcionamiento es idéntico al anterior.

```
void initsedcvecamb()
{
        menu_type menu;

        if (( menu = menu_Open())==NULL)
            generror(6,0);
```

```
        menu_Printf(menu, " Tel,fono:");
        menu_Printf(menu,"@p[0,12]@f[@[9,#]]",data_buf1.sNum_Tel
                , &digit_funcs);
        menu_Flush(menu);
        if (( sedcvecamb = sed_Open(menu))==NULL)
                generror(7,0);
        sed_SetColors(sedcvecamb, 0x1b, 0x31, 0x70);
        sed_SetBorder(sedcvecamb, bd_prompt);
        sed_SetBorderTitle(sedcvecamb, " NUEVO NUMERO ");
        sed_SetPosition(sedcvecamb, 4, 56);
        sed_SetHeight(sedcvecamb, 1);
        sed_SetWidth(sedcvecamb, 20);
        sed_SetShadow(sedcvecamb,1);
        sed_SetSpecial(sedcvecamb,myspc_Esc);
        sed_RepaInt(sedcvecamb);
        sed_Go(sedcvecamb);
        sed_Close(sedcvecamb);
}
```

En esta parte del código podemos observar dos funciones de validación, una de ellas es *int_Func* la cual válida que ingrese un número entero, así como *string_Func* que válida los datos ingresados sean alfanuméricos.

La variables de tipo estructura que se utilizan en este segmento de código están descritas en el desarrollo e implementación de la base de datos en CODEBASE.

```
void initseddesal()
```

```
{
menu_type menu;

    if ((menu = menu_Open())==NULL)
    generror(6,0);
    menu_Printf(menu, "CLAVE POB.:@f[@[7,#]] ", &data_buf1.sClave_Pob,
    &string_funcs);
    menu_Printf(menu, "ALARMA:@f[@[7,#]] ",   &data_buf1.sAlarma,
    &string_funcs);
    menu_Printf(menu, "GRUPO:@f[@[5,#]] ",    &data_buf1.sGrupo,
    &string_funcs);
    //&llst_funcs,"Cualquier tecla para ver opciones",
    menu_Printf(menu, "TIPO APTO:@f[@[2,#]] ",  &data_buf1.sTipo_Apto,
    &string_funcs);
    menu_Printf(menu, "TIPO AMB:@f[@[2,#]]\n",.  &data_buf1.sTipoAmb,
    &string_funcs);
    menu_Printf(menu, "LUGAR INS:@f[@[1,#]] ",&data_buf1.sLugar_Ins,
    &string_funcs);
    menu_Printf(menu,"FECHAINS:@f[##/##/####]",&data_buf1.sFecha_Ins,
    &digit_funcs);
    menu_Printf(menu, "ZONA:@f[@[2,#]] ",  &data_buf1.sZona,
    &string_funcs);
    menu_Printf(menu, "TIPO SERV:@f[@[1,#]] ",  &data_buf1.iTServicio,
    &int_funcs);
    menu_Printf(menu, "SUSP.:@fp[@[3,#]] ", &data_buf1.iSuspendido,
    &int_funcs);
    menu_Printf(menu, "S./C.:@f[@[3,#]]\n", &data_buf1.iSinControl,
    &int_funcs);
```

```
menu_Printf(menu, "CALLE Y NUM:
@f[@[50,#]]\n",&data_buf1.sCalleyNum,&string_funcs);
menu_Printf(menu, "COLONIA: @f[@[50,#]]\n",   &data_buf1.sColonia,
&string_funcs);
menu_Printf(menu, "CATEGORIAS NECESARIAS (EXA)        ");
if (data_buf1.iCatNok)
        menu_Printf(menu, "CATEGORIAS ERRONEAS (EXA)");
menu_Printf(menu, "\n");
menu_Printf(menu, "CBA TLI BIC SSI OBA TCL ICS BOC TBO  ");
if (data_buf1.iCatNok)
        menu_Printf(menu, "CBA TLI BIC SSI OBA TCL ICS BOC TBO");
menu_Printf(menu, "\n");
menu_Printf(menu, "@f[@[3,#]] ",  &data_buf1.iCNCBA,     &int_funcs);
menu_Printf(menu, "@f[@[3,#]] ",  &data_buf1.iCNTLI,     &int_funcs);
menu_Printf(menu, "@f[@[3,#]] ",  &data_buf1.iCNBIC,     &Int_funcs);
menu_Printf(menu, "@f[@[3,#]] ",  &data_buf1.iCNSSI,     &int_funcs);
menu_Printf(menu, "@f[@[3,#]] ",  &data_buf1.iCNOBA,     &int_funcs);
menu_Printf(menu, "@f[@[3,#]] ",  &data_buf1.iCNTCL,     &int_funcs);
menu_Printf(menu, "@f[@[3,#]] ",  &data_buf1.iCNICS,     &int_funcs);
menu_Printf(menu, "@f[@[3,#]] ",  &data_buf1.iCNBOC,     &int_funcs);
menu_Printf(menu, "@f[@[3,#]]  ", &data_buf1.iCNTBO,     &int_funcs);
if (data_buf1.iCatNok) {  // Solo despliega si tiene error
menu_Printf(menu, "@fp[@[3,#]] ", &data_buf1.iCACBA,     &int_funcs);
menu_Printf(menu, "@fp[@[3,#]] ", &data_buf1.iCATLI,     &int_funcs);
menu_Printf(menu, "@fp[@[3,#]] ", &data_buf1.iCABIC,     &int_funcs);
menu_Printf(menu, "@fp[@[3,#]] ", &data_buf1.iCASSI,     &int_funcs);
menu_Printf(menu, "@fp[@[3,#]] ", &data_buf1.iCAOBA,     &int_funcs);
menu_Printf(menu, "@fp[@[3,#]] ", &data_buf1.iCATCL,     &int_funcs);
menu_Printf(menu, "@fp[@[3,#]] ", &data_buf1.iCAICS,     &int_funcs);
```

```
menu_Printf(menu, "@fp[@[3,#]] ", &data_buf1.iCABOC,    &int_funcs);
menu_Printf(menu, "@fp[@[3,#]] ",&data_buf1.iCATBO,     &int_funcs);
}
menu_Printf(menu, "\n");
menu_Printf(menu, "CATEGORIAS NECESARIAS (SIST 2199)    ");
if (data_buf1.iCatNok)
      menu_Printf(menu, "CATEGORIAS ERRONEA (SIST 2199)");
menu_Printf(menu, "\n");
menu_Printf(menu, "BADP CBOX ORPT SSIG TRSTR LNCHAR HM  ");
if (data_buf1.iCatNok)
      menu_Printf(menu, "BADP CBOX ORPT SSIG TRSTR LNCHAR HM");
menu_Printf(menu, "\n");
menu_Printf(menu, "INTC PAYP OCB  SUBS ICS   LINECH    ");
if (data_buf1.iCatNok)
      menu_Printf(menu, "INTC PAYP OCB  SUBS ICB   LINECH   ");
menu_Printf(menu, "\n");
menu_Printf(menu, "@f[@[3,#]]  ",  &data_buf1.iCNBADP,     &int_funcs);
menu_Printf(menu, "@f[@[3,#]]  ",  &data_buf1.iCNCBOX,     &int_funcs);
menu_Printf(menu, "@f[@[3,#]]  ",  &data_buf1.iCNORPT,     &int_funcs);
menu_Printf(menu, "@f[@[3,#]]  ",  &data_buf1.iCNSSIG,     &int_funcs);
menu_Printf(menu, "@f[@[3,#]]  ", &data_buf1.iCNTRSTR,     &int_funcs);
menu_Printf(menu, "@f[@[3,#]]  ", &data_buf1.iCNLNCHAR,    &int_funcs);
menu_Printf(menu, "@f[@[3,#]] ", &data_buf1.iCNHM,         &int_funcs);
if (data_buf1.iCatNok) {  // Solo despliega si tiene error
menu_Printf(menu, "@fp[@[3,#]] ", &data_buf1.iCABADP,      &int_funcs);
menu_Printf(menu, "@fp[@[3,#]] ", &data_buf1.iCACBOX,      &int_funcs);
menu_Printf(menu, "@fp[@[3,#]] ", &data_buf1.iCAORPT,      &int_funcs);
menu_Printf(menu, "@fp[@[3,#]] ", &data_buf1.iCASSIG,      &int_funcs);
menu_Printf(menu, "@fp[@[3,#]] ",&data_buf1.iCATRSTR,      &int_funcs);
```

```
menu_Printf(menu, "@fp[@[3,#]]   ",&data_buf1.iCALNCHAR,
&int_funcs);
menu_Printf(menu, "@fp[@[3,#]]",&data_buf1.iCAHM,     &int_funcs);
}
menu_Printf(menu, "\n");
menu_Printf(menu, "LECTURA  CONTADOR  HORA FECHA     LECTURA
CATEGORIAS  HORA FECHA\n");
menu_Printf(menu, "ULTIMA    @fp[@[8,#]] ",&data_buf1.lContadorU,
&long_funcs);
menu_Printf(menu, "@fp[@[5,#]] ", &data_buf1.sHoraContU,
&string_funcs);
menu_Printf(menu, "@fp[@[10,#]]      ",&data_buf1.sFechaContU,
 &string_funcs);
menu_Printf(menu, "@fp[@[5,#]] ",  &data_buf1.sHoraCatU,
&string_funcs);
menu_Printf(menu, "@fp[@[10,#]]\n",          &data_buf1.sFechaCatU,
&string_funcs);
menu_Printf(menu, "PENULTIMA @fp[@[8,#]] ",&data_buf1.lContadorP,
&long_funcs);
menu_Printf(menu, "@fp[@[5,#]] ", &data_buf1.sHoraContP,
&string_funcs);
menu_Printf(menu, "@fp[@[10,#]]
",&data_buf1.sFechaContP, &string_funcs);
menu_Printf(menu, "@fp[@[5,#]] ", &data_buf1.sHoraCatP,
 &string_funcs);
menu_Printf(menu, "@fp[@[10,#]]\n", &data_buf1.sFechaCatP,
&string_funcs);
menu_Flush(menu);
if ((seddesal = sed_Open(menu))==NULL)
```

```
    generror(7,0);
    sed_SetColors(seddesal, 0x1b, 0x31, 0x70);
    sed_SetBorder(seddesal, bd_prompt);
    sed_SetPosition(seddesal, 7, 1);
    sed_SetHeight(seddesal, 14);
    sed_SetWidth(seddesal, 75);
    sed_SetShadow(seddesal,1);
    sed_SetSpecial(seddesal,myspc_Esc2);
    sed_Repaint(seddesal);
}
```

3.9. Integración, Pruebas e Implantación del Sistema.

Técnicas de Prueba del Software.

La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación.

La creciente inclusión del software como un elemento más de muchos sistemas y la importancia de los "costes" asociados a un fallo del mismo están motivando la creación de pruebas minuciosas y bien planificadas.

Objetivos de la Prueba de Software.

La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.

Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Si la prueba se lleva a cabo con éxito (de acuerdo con el objetivo anteriormente establecido), descubrirá errores en el software. Como ventaja secundaria, la prueba demuestra hasta que punto las funciones del software parecen funcionar de acuerdo con las especificaciones y parecen alcanzarse los requisitos de rendimiento. Además, los datos que se van recogiendo a medida que se lleva a cabo la prueba proporcionan una buena indicación de

la fiabilidad del software y, de alguna manera, indican la calidad del software como un todo. Sin embargo, hay una cosa que no puede hacer la prueba.

Flujo de información en la prueba.

El flujo de información para la prueba sigue el esquema descrito en la figura 3.9. - 1. Se proporcionan dos clases de entradas al proceso de prueba: (1) una configuración del software que incluye la especificación de requisitos del software, la especificación del diseño y el código fuente; (2) una configuración de prueba que incluye un plan y procedimiento de prueba, algunas herramientas de prueba que se van a utilizar y casos de prueba y resultados esperados. En realidad la configuración de prueba es un subconjunto de la configuración del software.

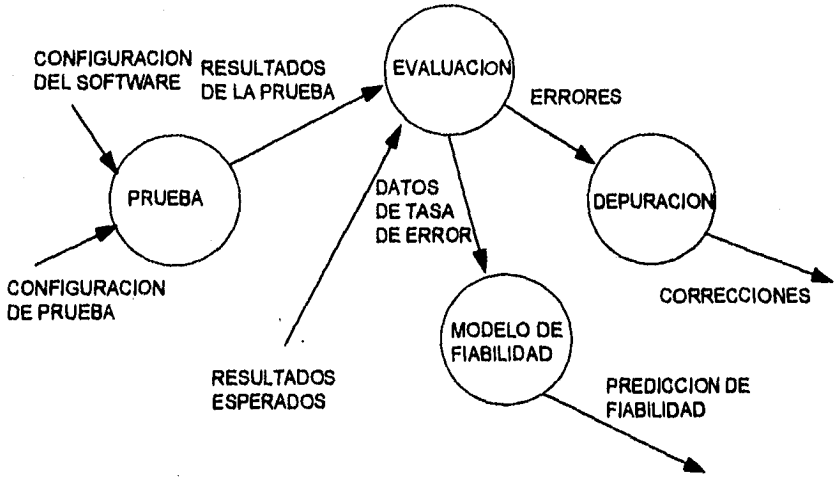


Fig. 3.9. - 1. Flujo de información en la Prueba.

A medida que se van recopilando y evaluando los resultados de la prueba, comienza a vislumbrarse una medida cualitativa de la calidad y de la fiabilidad

del software. Si se encuentran con regularidad errores serios que requieren modificaciones en el diseño, la calidad y la fiabilidad del software quedan en entredicho, siendo necesarias posteriores pruebas. Si, por otro lado, el funcionamiento del software parece ser correcto y los errores que se encuentran son fácilmente corregibles, se puede sacar una de dos conclusiones: (1) la calidad de la fiabilidad del software son aceptables, o (2) las pruebas son inadecuadas para descubrir errores serios. Finalmente, si la prueba no descubre errores, quedará la sospecha de que no se ha pensado cuidadosamente la configuración de prueba y de que los errores están escondidos en el software. Estos defectos serán eventualmente descubiertos por el usuario y corregidos por el profesional durante la fase de mantenimiento.

Diseño de Casos de Prueba.

Cualquier producto de ingeniería (y de muchos otros campos) puede ser probado de una de dos formas: (1) conociendo la función especifica para la que fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa; (2) conociendo el funcionamiento del producto, se pueden desarrollar pruebas que aseguren que "todas las piezas encajan"; o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada. El primer enfoque de prueba se denomina prueba de la caja negra y la segunda, prueba de la caja blanca.

Cuando se considera el software de computadora, la prueba de la caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se

produce una salida correcta, así como que la integridad de la información externa (p. ej.: archivos de datos) se mantiene. Una prueba de la caja negra examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura interna del software.

La prueba de la caja blanca del software se basa en el minucioso examen de los detalles procedurales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el "estado del programa" en varios puntos para determinar si el estado real coincide con el esperado o afirmado.

Pruebas de la Caja Blanca.

La prueba de la caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedural para derivar los casos de prueba. Mediante los métodos de prueba de la caja blanca, el ingeniero del software puede obtener casos de prueba que (1) garanticen que se ejercitan por lo menos una vez todos los caminos independientemente de cada módulo; (2) ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; (3) ejecuten todos los bucles en sus límites y con sus límites operacionales y (4) ejerciten las estructuras internas de datos para segurar su validez.

Prueba del Camino Básico.

La prueba del camino básico es una técnica de prueba de la caja blanca. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedural y usar esa medida como guía para la definición de un conjunto básico de caminos de

ejecución. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Notación de Grafo de Flujo.

Antes de considerar el método del camino básico introduciremos una notación que ayuda en el modelado conocido como grafo de flujo.

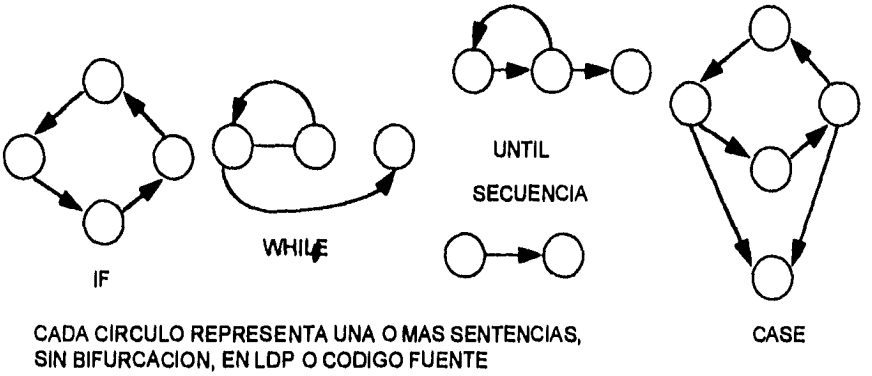


Fig. 3.9. - 2. Notación de Grafo de Flujo.

Para ilustrar el uso de un grafo de flujo, consideremos la representación del diseño procedural de la figura 3.9. - 2. En ella, se usa un diagrama de flujo para representar la estructura de control del programa. En la parte b de la figura se muestra el grafo de flujo correspondiente (suponiendo que no hay condiciones compuestas en los rombos de decisión del diagrama de flujo). Cada circulo es un nodo y cada flecha es una arista. Los nodos pueden representar una secuencia de cuadros de proceso y a un rombo de decisión. Las flechas y las aristas tienen el mismo significado. Una arista debe terminar en un nodo incluso aunque el nodo no represente ninguna sentencia procedural (p.ej.: véase el

símbolo para la estructura *if-then-else*. Las áreas delimitadas por aristas y nodos se denominan regiones. Cuando contabilizamos las regiones incluimos el área exterior del grafo, contando esta como una región más.

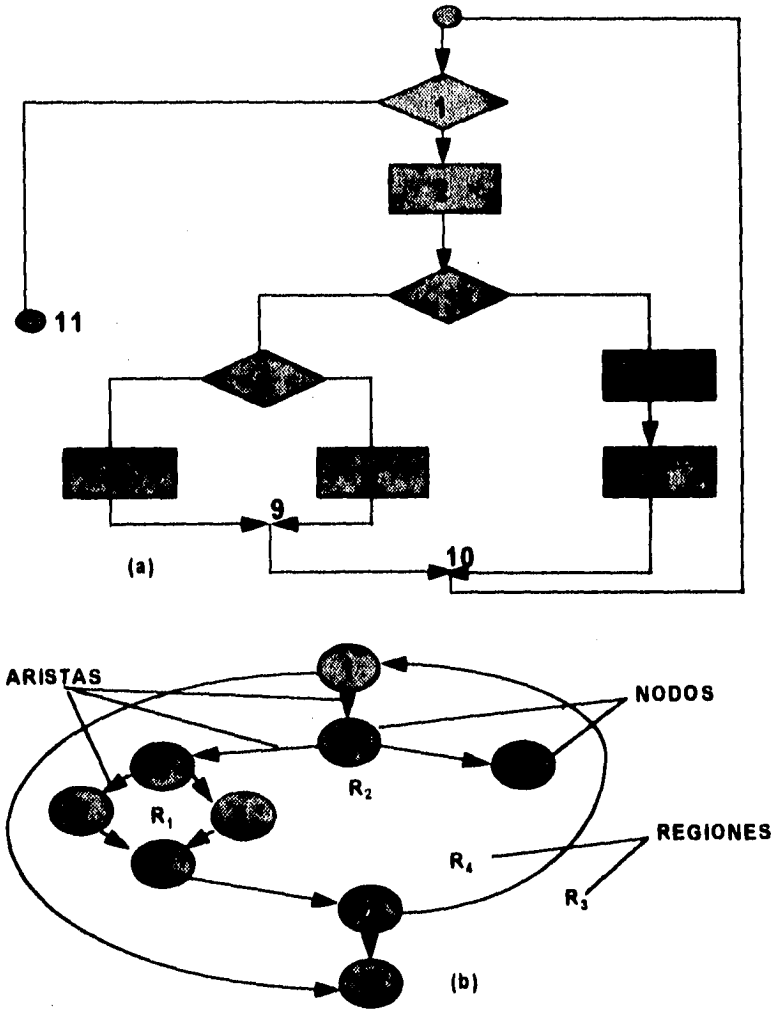
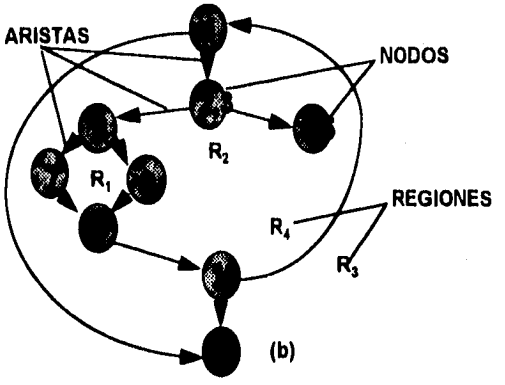


Fig. 3.9. - 3. (a) Diagrama de Flujo (b) Grafo de Flujo.

Cualquier representación del diseño procedural se puede traducir a un grafo de flujo. En la figura 3.9. - 4. se muestra un segmento del lenguaje de diseño de programas y su correspondiente grafo de flujo. Se puede observar que se han enumerado las sentencias y que un grafo de flujo usa la misma numeración.




```
                    LDP
  procedimiento: ordenar
1: do while queden registros
       leer registro
2:     if campo 1 del registro = 0
3:        then procesar registro;
               guardar en buffer;
               incrementar contador;
4:     elseif campo 2 del registro = 0
5:        then reinicializar contador;
6:        else procesar registro;
               guardar en archivo;
7a:    endif
       endif
7b: enddo
6:  end
```

Fig. 3.9. - 4. Traducción de LDP a Grafo de Flujo.

Cuando en un diseño procedural se encuentran condiciones compuestas, la generación del grafo de flujo se hace un poco más complicada. Una condición compuesta se da cuando aparecen uno o más operadores lógicos (OR, AND, NAND, NOR lógicos) en una sentencia condicional.

Complejidad Ciclónica.

La complejidad ciclónica es una métrica del software que proporciona una medida cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclónica define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de

pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

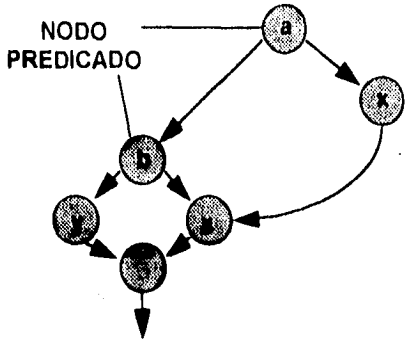


Fig. 3.9. - 5 Lógica Compuesta.

Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. En términos del grafo de flujo, un camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente a la definición del camino. Por ejemplo, para el grafo de flujo de la figura 3.9. - 3 un conjunto de caminos independientes sería:

Camino 1: 1-11
Camino 2: 1-2-3-4-5-10-1-11
Camino 3: 1-2-6-8-9-10-1-11
Camino 4: 1-2-3-6-7-9-10-1-11

Cada camino nueva introduce una nueva arista.

Los caminos 1,2,3 y 4 definidos anteriormente componen un conjunto básico para el grafo de flujo de la figura 3.9. - 3. Surge entonces la pregunta ¿como

sabemos cuantos caminos hemos de buscar?. El cálculo de la complejidad ciclomática nos da la respuesta.

La complejidad ciclomática está basada en la teoría de grafos y nos da una métrica del software extremadamente útil. La complejidad se puede calcular de la siguiente forma:

La complejidad ciclomática, V(G), de un grafo de flujo G se define como:

$$V(G)= A - N +2$$

donde "A" es el número de aristas del grafo de flujo y "N" es el número de nodos del grafo de flujo. Entonces el grafo de flujo en cuestión tiene:

$$V(G) = 11 \text{ aristas} - 9 \text{ nodos} + 2 = 4$$

Más importante, el valor de V(G) nos da un valor límite para el número de caminos independientes que componen el conjunto básico y, consecuentemente, un valor límite para el número de pruebas que se deben diseñar y ejecutar para garantizar que se cubren todas las sentencias del programa.

Prueba de Bucles

Los bucles son la piedra angular de la inmensa mayoría de los algoritmos implementados en software. Y por ello debemos prestarles atención cuando llevamos la prueba de software.

La prueba de bucles es una técnica de prueba de la caja blanca que se centra exclusivamente en la validez de las construcciones de bucles. Se pueden definir cuatro clases diferentes de bucles: bucles simples, bucles concatenados, bucles anidados y bucles no estructurados. Figura 3.9. - 6.

Bucles simples.- A los bucles simples se les debe aplicar el siguiente conjunto de pruebas, donde "n" es el número máximo de pasos permitidos por el bucle:

1. Pasar por alto totalmente el bucle
2. Pasar una sola vez por el bucle
3. Pasar dos veces por el bucle
4. Hacer "m" pasos por el bucle con "M < n"
5. Hacer "n - 1", n y "n + 1" pasos por el bucle

Bucles anidados.- Si extendemos el enfoque de prueba de los bucles simples a los bucles anidados, el número posibles de pruebas aumenta geométricamente a medida que aumenta el nivel de anidamiento. Esto llevaría a un número impracticable de pruebas. Para evitar esto, se sugiere lo siguiente:

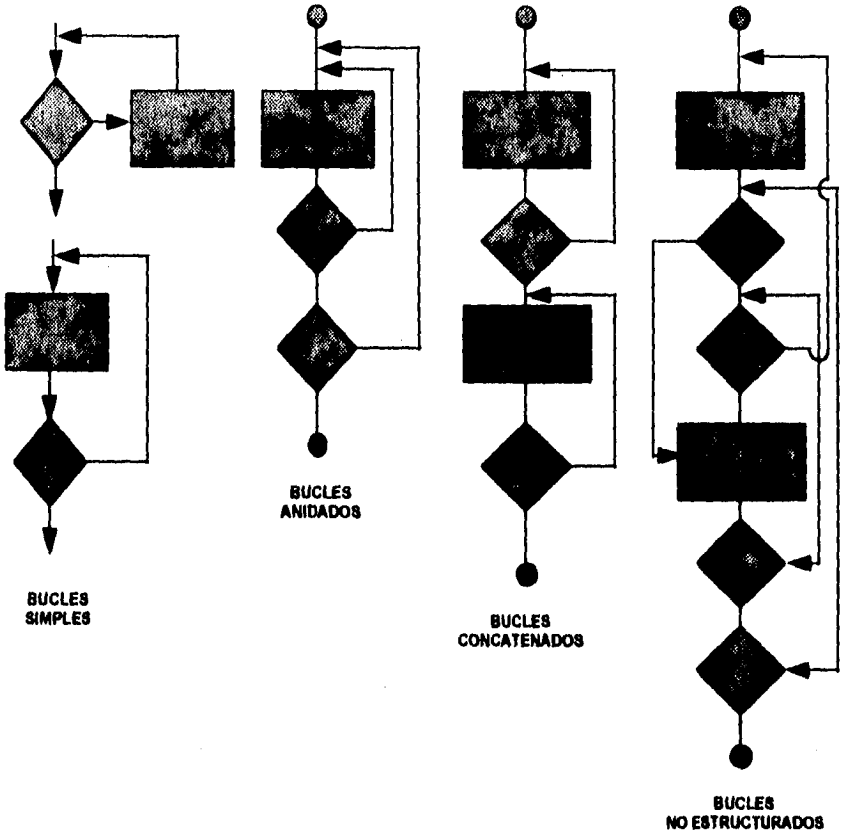


Fig. 3.5.6. Bucles

1. Comenzar en el bucle más interior. Establecer los demás bucles en sus valores mínimos.
2. Llevar a cabo las pruebas de bucles simples para el bucle más interior, mientras se mantienen los parámetros de iteración (p.ej.: contadores de bucles) de los bucles externos en sus valores mínimos. Añadir otras pruebas para valores fuera de rango o excluidos.

3. Progresar hacia fuera, llevando a cabo pruebas para el siguiente bucle, pero manteniendo todos los bucle externos en sus valores mínimos y los demás bucles anidados en sus valores "típicos".
4. Continuar hasta que se hayan probado todos los bucles.

Bucles concatenados.- Los bucles concatenados se pueden probar mediante el enfoque anteriormente definido para los bucles simples, mientras cada uno de los bucles sea independiente del resto. Por ejemplo, si hay 2 bucles concatenados y se usa el contador del bucle 1 como valor inicial del bucle 2, entonces los bucles no son independientes. Cuando los bucles no son independientes, se recomienda usar el enfoque aplicado para los bucles anidados.

Bucles no estructurados.- Siempre que sea posible, esta clase de bucles se deben rediseñar para que se ajusten a las construcciones de la programación estructurada.

Prueba de la Caja Negra

Los métodos de prueba de la caja negra se centran en los requisitos fundamentales del software. O sea, la prueba de la caja negra permite al ingeniero tomar conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de la caja negra no es una alternativa a las técnicas de la prueba de la caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir tipos de errores que los métodos de la caja blanca.

La prueba de la caja negra intenta encontrar errores en las siguientes categorías: (1) funciones incorrectas o ausentes; (2) errores de interfaz; (3) errores en estructuras de datos o en accesos a bases de datos externas, (4) errores de rendimiento y (5) errores de inicialización y de terminación.

A diferencia de la prueba de la caja blanca, que se lleva a cabo previamente en el proceso de prueba, la prueba de la caja negra tiende a aplicarse durante fases de prueba. Ya que la prueba de la caja negra ignora intensionalmente la estructura de control, centra su atención en el campo de información. Las pruebas se diseñan para responder a las siguientes preguntas:

- ¿Cómo se prueba la validez funcional?
- ¿Qué clases de entradas compondrán unos buenos casos de prueba?
- ¿Es el sistema particularmente sensible a ciertos valores de entrada?
- ¿De que forma están aislados los límites de una clase de datos?
- ¿Qué volúmenes y niveles de datos tolerará el sistema?
- ¿Que efectos sobre la operación del sistema tendrán combinaciones específicas de datos?

Mediante las pruebas de la caja negra se deriva un conjunto de casos de prueba que satisfacen los siguientes criterios: (1) casos de prueba que reducen, en un coeficiente que es mayor que uno, el número de casos de prueba adicionales que se deben diseñar para alcanzar una prueba razonable y (2) casos de prueba que nos dicen algo sobre la presencia o ausencia de clases de errores en lugar de un error asociado solamente con la prueba, en particular, que se encuentre disponible.

Partición Equivalente

La participación equivalente es un método de prueba de la caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores (p.ej.: procesamiento incorrecto de todos los datos de caracteres) que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados validos o inválidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

1. Si una condición de estrada especifica un rango, se define una clase de equivalencia valida y dos invalidas.
2. Si una condición de entrada requiere un valor especifico, se define una clase de equivalencia valida y dos invalidas.
3. Si una condición de entrada específica un miembro de un conjunto, se define una clase de equivalencia valida y una invalida.
4. Si una condición de entrada es lógica, se define una clase valida y una invalida.

Aplicando estas directrices para la obtención de clases de equivalencia, se pueden desarrollar y ejecutar casos de prueba para cada elemento de datos del campo de entrada. Los casos de prueba se seleccionan de forma que se ejercite el mayor número de atributos de cada clase de equivalencia a la vez.

Análisis de valores límite.

Por razones que no están del todo claras, los errores tienden a darse más en los límites del campo de entrada que en el "centro". Por ello, se ha desarrollado el análisis de valores límites (AVL) como técnica de prueba. El análisis de valores límite nos lleva a una elección de casos de prueba que ejerciten los valores límite.

El análisis de valores límite es una técnica de diseño de casos de prueba que complementa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los "bordes" de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL deriva casos de prueba también para el campo de salida.

Las directrices de AVL son similares en muchos aspectos a las que proporciona la partición equivalente:

Si una condición de entrada especifica un rango delimitado por los valores "a y b", se deben diseñar casos de prueba para los valores "a y *b*" para los valores justo por debajo y justo por encima de "a y *b*", respectivamente.

Si una condición de entrada especifica un número de valores, se deben desarrollar casos de prueba que ejerciten los valores máximos y mínimos. También se deben probar los valores justo por encima y justo por debajo del máximo y del mínimo.

Aplicar las directrices 1 y 2 a las condiciones de salida. Por ejemplo, supongamos que se requiere una tabla de "temperatura frente a presión" como salida de un programa de análisis de ingeniería. Se deben diseñar casos de prueba que creen un informe de salida que produzca el máximo (y el mínimo) número permitido de entradas en la tabla.

Si las estructuras de datos internas tienen límites preestablecidos (p.ej.: un array que tenga un límite definido de 100 entradas), hay que asegurarse de diseñar un caso de prueba que ejercite la estructura de datos en sus límites.

La mayoría de los ingenieros llevan a cabo de forma intuitiva alguna forma de AVL. Aplicando las directrices que se acaban de exponer, la prueba de límites será más completa y, por tanto, tendrá una mayor probabilidad de detectar errores.

Pruebas del Sistema.

La prueba es un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Por esta razón se debe definir en el proceso de la ingeniería del software una plantilla para la prueba del software - un conjunto de pasos en los que podamos situar las técnicas específicas de diseño de casos de prueba y los métodos de prueba.

Se han propuesto varias estrategias de prueba de software en distintos libros. Todas proporcionan al ingeniero de software una plantilla para la prueba y todas tienen las siguientes características generales:

- La prueba comienza en el nivel de módulo y trabaja "hacia afuera", hacia la integración de todo el sistema basado en computadora.
- Diferentes técnicas de prueba son apropiadas en diferentes momentos.
- La prueba la lleva a cabo el que desarrolla el software y (para grandes proyectos) un grupo de prueba independiente.

La prueba y la depuración son actividades diferentes, pero la depuración se puede incluir en cualquier estrategia de prueba.

Una estrategia para la prueba del software debe acomodar pruebas de bajo nivel que verifiquen que cada pequeño segmento de código fuente se ha implementado correctamente, así como pruebas de alto nivel que demuestren la validez de las principales funciones del sistema frente a los requisitos del cliente. Una estrategia debe construir una guía para el profesional y proporcionar un conjunto de referencias para el gestor o director. Debido a que los pasos de la estrategia de prueba se dan a la vez cuando surgen la presión de los plazos fijados, se debe poder medir el progreso, y los problemas deben aparecer lo antes posible.

En cualquier proyecto de software existe un conflicto de intereses inherente que aparece cuando comienza la prueba. Se pide a la gente que ha construido el software que lo pruebe. Esto parece totalmente inofensivo; después de todo, ¿quién puede conocer mejor un programa que los que lo han desarrollado? Desgraciadamente, esos mismos programadores tienen un gran interés en

demostror que el progromo está libre de errores, que funciono de acuerdo con los especificaciones del cliente y que estoró listo de acuerdo con los plozos y el presupuesto. Codo uno de estos intereses se convierte en inconveniente o lo horo de encontrar errores o lo lorgo del proceso de prueba.

Desde un punto de vista psicológico, el onálisis y el diseño del software (junto con lo codificoción) son tareas constructivos. El ingeniero de software crea un progromo de computadora, su documentación y sus estructuras de datos asociodos. Al iguol que cuolquier constructor, el Ingeniero de software está orgulloso del edificio que ocabo de construir y se enfrento a cualquiera que quiera socarle defectos. Cuondo comienza lo prueba, aparece un socavado, aunque definido intento de "romper" lo que el ingeniero de softwore a construido. Desde el punto de vista del constructor, la prueba se puede considerar (psicológicamente) destructiva. Por tanto, el constructor anda con cuidado, diseñando y ejecutondo pruebos que demuestren que el programa funciona, en lugar de detector los errores. Desgraciadamente, los errores seguirán estondo. Y si el Ingeniero de software no los encuentra, el cliente si lo hará.

A menudo, existen clertos malentendidos que se pueden deducir equivocadamente de la onterior discusión: (1) el que desarrolla el software no debe entrar en el proceso de prueba; (2) el software debe ser "puesto a salvo" de extraños que puedan probarlo de forma despiadada; (3) los encargos de la pruebo solo aporecen en el proyecto cuando comienzan las etapas de prueba. Cado una de estas frases es incorrecta.

El que desorrolle el softwore siempre es responsable de probar las unidades individuales (módulos) del progroma, osegurándose de que cada uno lleva a

cabo la función para la que fue diseñada. Solo una vez que la arquitectura del software esté completa entra en juego un grupo independiente de prueba.

El papel del grupo independiente de prueba (GIP) es eliminar los problemas inherentes asociados con el hecho de permitir al constructor que pruebe lo que ha construido. Una prueba independiente elimina el conflicto de interés que, de otro modo, estaría presente. Después de todo, al personal del equipo que forma el grupo independiente se le paga para que encuentre errores.

Sin embargo, el que haya desarrollado el software no cede simplemente el programa al GIP y se desentiende. El desarrollador y el GIP trabajan estrechamente a lo largo del proyecto de software para asegurar que se realizan pruebas exhaustivas. Mientras se dirige la prueba, el desarrollador debe estar disponible para corregir los errores que se van descubriendo.

El GIP es una parte del equipo del proyecto de desarrollo de software en el sentido de que se ve implicado durante el proceso de especificación y sigue implicado a todo lo largo de un proyecto. Sin embargo, en muchos casos, el GIP informa a la organización de garantía de calidad del software, consiguiendo de este modo un grado de independencia que no sería posible si fuera una parte de la organización de desarrollo de software.

Criterios para la prueba

La validación del software se consigue mediante una serie de pruebas de la caja negra que demuestran la conformidad con los requisitos. Un plan de prueba traza las pruebas que se han de llevar a cabo y un procedimiento de prueba define los casos de prueba específicos que se usarán para demostrar la

conformidad con los requisitos. Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfacen todos los requisitos funcionales, que se alcanzarán todos los requisitos de rendimiento, que la documentación es correcta e inteligible y que se alcanzan otros requisitos (p.ej.: portabilidad, compatibilidad, recuperación de errores, facilidad de mantenimiento).

Una vez que se procede con cada caso de prueba de validación, puede darse una de dos condiciones: (1) las características de funcionamiento o de rendimiento están de acuerdo con las especificaciones y son aceptables o (2) se descubre una desviación de las especificaciones y se crea una lista de deficiencias. Las desviaciones o errores descubiertos en esta fase del proyecto raramente se pueden corregir antes de terminar el plan. A menudo es necesario negociar con el cliente un método para resolver las deficiencias.

Es virtualmente imposible que un encargado del desarrollo del software pueda prever como un cliente usará realmente un programa. Se pueden interpretar mal las instrucciones de uso, se pueden usar regularmente extrañas combinaciones de datos y una salida que puede estar clara para el que realiza la prueba puede resultar ininteligible para un usuario normal.

Cuando se construye software a medida para un cliente, se lleva a cabo una serie de pruebas de aceptación para permitir que el cliente valide todos los requisitos. Llevado a cabo por el usuario final en lugar del equipo de desarrollo, una prueba de aceptación puede ir desde un informal "paso de prueba" hasta la ejecución sistemática de una serie de pruebas bien planificadas. De hecho, la prueba de aceptación puede tener lugar a lo largo de semanas o meses, descubriendo así errores acumulados que pueden ir degradando el sistema.

Si el software se desarrolla como un producto que va a ser usado por muchos clientes, no es práctico realizar pruebas de aceptación formales para cada uno de ellos. La mayoría de los constructores de software llevan a cabo un proceso denominado prueba alfa y beta para descubrir errores que parezca que solo el usuario final puede descubrir.

La prueba alfa es conducida por un cliente en el lugar de desarrollo. Se usa el software de forma natural, con el encargado del desarrollo "mirando por encima del hombro" del usuario y registrando errores y problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado.

La prueba beta se lleva a cabo en uno o más lugares de clientes por los usuarios finales del software. A diferencia de la prueba alfa, el encargado del desarrollo, normalmente no esta presente. Así, la prueba beta es una "prueba en vivo" del software en un entorno que no puede ser controlado por el equipo de desarrollo. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al equipo de desarrollo. Como resultado de los problemas anotados durante la prueba beta, el equipo de desarrollo del software lleva a cabo modificaciones y así prepara una versión del producto de software para toda la base de clientes.

Pruebas Efectuadas al Sistema.

Prueba de la Interfaz del Usuario.

En el módulo principal se probó que cada uno de los menús de opciones y de las pantallas de captura cumplieran con los puntos que se muestran en la figura 3.9. - 7.

NOMBRE DEL SISTEMA
ORTOGRAFIA CORRECTA
NOMBRE DEL MENU
OPCIONES DEL MENU
TECLA DE ENTRAR
TECLA DE ESCAPE
TECLA DE F1 y F2
MENSAJES DEL SISTEMA

Fig. 3.9. - 7 Pruebas que se Aplicaron a los Menús.

- El Nombre del Sistema. Siempre aparece en la parte superior de todas las pantallas como "ALCANCIAS 3.4 ".
- Ortografía Correcta. Se revisó que esto se cumpliera en todos los mensajes, pantallas y opciones de menú.
- Nombre del Menú. Cuando se selecciona un menú se mantiene el nombre de este en la parte superior del menú y centrado.
- Opciones del Menú. Se muestran todas las opciones que corresponden al menú seleccionado.
- Tecla de Entrar. Durante la selección de una opción en los menús se podrá hacer uso de esta tecla.
- Tecla de Escape. Se utiliza para abortar una opción o regresar al menú anterior.
- Teclas de F1 y F2. Se usan para interrumpir un proceso de comunicación.

Una vez que se probaron los menús y las pantallas del sistema, se procedió a revisar cada uno de los procesos que el sistema realiza. A continuación presentamos estas pruebas.

Prueba de Funcionamiento con Centrales Telefónicas.

Para comprobar el buen funcionamiento del Sistema "ALCANCIAS", se evaluó con las centrales telefónicas MOA, IOC12, MAN MACHINE, MPTMON. Las dos primeras manejan el sistema EXA 10 y las últimas el SISTEMA 2199. Para cada una de las centrales se evaluaron los procesos utilizando una base de datos de 3500 registros mixtos los cuales se presentan en la siguiente tabla:

	MOA			IOC12			MANMACHINE			MPTMON		
	A	B	C	A	B	C	A	B	C	A	B	C
Revisión Continua	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.
Revisión de Categorías	O.K	O.K	O.K	O.K	O.K	O.K	O.K	O.K	O.K	O.K	O.K	O.K
Revisión de Contadores	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.
Suspensión de Alcancías	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.
Reanudación de Alcancías	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.
Desbloqueo de líneas	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.
Consulta de Categorías	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.
Cambios de Categorías	O.K	O.K	O.K	O.K	O.K	O.K	O.K	O.K	O.K	O.K	O.K	O.K
Consulta de Contadores	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.
Prueba de línea	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.	OK.

A= Comunicación con la Central

B= Presentación de los resultados en pantalla

C= Análisis e Interpretación de Resultados

Revisión Continua.

- Se revisó que en este proceso la prueba de contadores se activara a la hora especificada en la configuración de funcionamiento.

- Se revisó que el proceso se reanudara a partir del registro en que se suspendió el proceso

- Se revisó que únicamente los teléfonos que tuvieran alteración en las categorías de las centrales fueran corregidos mediante un comando apropiado de corrección.

- Se revisó que únicamente los teléfonos que no tuvieran avance en el contador de llamadas fueran en los que se realizara la prueba de línea.

Revisión de Categorías.

Se revisó que únicamente los teléfonos que tuvieran alteración en las categorías de las centrales fueran corregidos mediante un comando apropiado de corrección.

Revisión de Contadores.

Se revisó que el proceso se reanudara a partir del registro en que se suspendió el proceso.

Se revisó que únicamente los teléfonos que no tuvieran avance en el contador de llamadas fueran en los que se realizara la prueba de línea.

Suspensión de Alcancías.

Se hizo una prueba física de que las alcancías que se suspendían mediante este proceso, en la práctica dejaban de funcionar y se revisó que el sistema contestara afirmativamente a la suspensión.

Se comprobó la consistencia de la información entre las bases de datos del sistema de "ALCANCIAS" con respecto a la información de la base de datos de la central telefónica.

Reanudación de Alcancías.

igual que en el proceso anterior se realizaron dos pruebas: la primera en donde la alcancía suspendida después del envío del comando de reanudación, volvía a funcionar y se comprobó que la central contestara a la reanudación de alcancía.

Desbloqueo de Alcancías.

Se comprobó el buen funcionamiento de la línea del abonado una vez que se solucionó el problema que originó el bloqueo de esa línea por parte de la compañía telefónica.

Consulta de Categorías.

Se comprobó que los status de las categorías de las alcancías que son programadas manualmente, fueran consistentes con la información registrada en la base de datos de la central.

Cambio de Categorías.

Se comprobó que el sistema propusiera el tipo de aparato correcto dependiendo del sistema que se estuviera accesando.

Se comprobó que los cambios que se realizan en las categorías de las alcancías fueran posteriormente actualizados en la base de datos de la central.

Consulta de Contadores.

Se comprobó que el número de llamadas de una línea coincidió con el contador que registra el sistema.

Prueba de línea de Alcancía.

Se compararon los resultados que ofrece el sistema contra las pruebas físicas de campo que consistieron en conectar un hilo a tierra, a batería, simular condiciones de rotura de línea y cruce de abonados.

Respecto de todos los puntos anteriores, se revisó que los parámetros de entrada del sistema no reciban caracteres erróneos.

Prueba de Seguridad.

Se permite el acceso únicamente con los password de Operador o bien de Superusuario y si en más de tres ocasiones se equivoca el password, el sistema aborta la entrada. Figura 3.9. - 9.

Entrada al Sistema mediante Password	O.K.	O.K.
Aborto de entrada al sistema después de tres intentos.	O.K.	O.K.

Fig. 3.9. - 9 Passwords de Entrada al Sistema.

Además se revisaron los niveles de acceso que se tiene al sistema dependiendo del password con el que se este operando al mismo. Figura 3.9. - 10.

NIVEL DE ACCESO	SUPERUSUARIO	OPERADOR
1. Revisión Continua	O.K.	O.K.
2. Revisión de Categorías	O.K.	O.K.
3. Revisión de Contadores	O.K.	O.K.
4. Suspensión de Alcancías	O.K.	
5. Reanudación de Alcancías.	O.K.	
6. Desbloqueo de Alcancías	O.K.	O.K.
7. Consulta de Categorías	O.K.	O.K.
8. Cambio de Categorías	O.K.	
9. Consulta de Contadores	O.K.	O.K.
10.Prueba de Línea	O.K.	O.K.
Manejo de Base de Datos	O.K.	
Diferentes Reportes del Sistema	O.K.	O.K.

Fig. 3.9. - 10 Niveles de Acceso al Sistema.

En este último punto es necesario aclarar que el reporte de Histórico de movimientos es el único reporte que no puede ser modificado de manera manual porque los datos que maneja están encriptados.

Por último, en las opciones del sistema, se comprobó que dependiendo de el password que se este utilizando, se podrá tener acceso a configurar al sistema. La figura 3.9. - 11 nos ilustra esto.

	Superusuario	Operador
Configuración de Funcionamiento.	O.K.	O.K.
Configuración de Series Válidas	O.K.	O.K.
Configuración de Categorías	O.K.	O.K.
Cambio de Password de Superusuario	O.K.	O.K.
Cambio de Password del Operador	O.K.	O.K.
Responsables por zona	O.K.	O.K.
Borrar datos históricos de Contadores.	O.K.	O.K.
Borrar Archivos de Reportes	O.K.	O.K.
Generar Reporte de Configuración	O.K.	O.K.
Mantenimiento de Archivos.	O.K.	O.K.

Fig. 3.9. - 11 Configuración de Funcionamiento del Usuario.

Conclusiones.

1. El sistema "Monitoreo de Teléfonos Públicos" ha presentado un ahorro muy significativo en el fraude de Larga Distancia, por lo que se esta cumpliendo con los objetivos.

2. El sistema "Monitoreo de Teléfonos Públicos", dependiendo de la periodicidad de supervisión y los recursos que se utilicen para ésta, será la medida en que disminuya el fraude de Larga Distancia.

3. Para cubrir las espectativas del cliente en un corto plazo, uno de los aspectos más importantes es la velocidad de desarrollo del sistema, por lo que es indispensable reutilizar código y en "Alcancías " se hizo.

4. En México, en el área de telecomunicaciones, la gran mayoría de la tecnología proviene de importaciones, lo que genera una conjunción de ambientes diferentes y es aquí donde la Ingeniería de Sistemas Mexicana ésta comprometida a integrar los sistemas con calidad.

5. En México el área de seguridad es una de las más importantes e incluso se sobrepasan las investigaciones a nivel internacional, por lo que la ingeniería en México debe superar y prever cualquier problema de este tipo. "Alcancías" es parte de la seguridad y cumple con este objetivo.

6. En cuanto a su mantenimiento el sistema esta dividido en módulos lo cual hace fácil identificar rutinas, funciones y procedimientos que necesiten modificarse para la implementar funciones que el usuario requiera en el futuro.

7. El acceso del sistema a la central no interfiere con los procesos de esta. Esto hace confiable el sistema "Alcancías".

8. El sistema es amigable y fácil de usar, ya que cuenta con un sistema de navegación de menús y ventanas. Aunque no cuenta con una opción de ayuda, siempre se tiene toda la información que el usuario necesita en cualquier momento en pantalla.

9. El sistema cubrió con todas las expectativas que se esperaban en cuanto a lo que fue el cambio de categorías y tomar las lecturas de los contadores, lo que hizo cubrir el 100% su objetivo.

10. La flexibilidad del uso de los reportes es de gran ayuda para el usuario ya que puede utilizarlas para realizar estadísticas así como pronósticos para un análisis posterior en cuanto a localización de fallas, etc.

11. La tendencia de la creciente tecnología y el cambio tan vertiginoso de las comunicaciones hacen que se tomen nuevos requerimientos para que el sistema siga vigente, es decir, se puede emigrar a nuevas plataformas y también cumple con éste objetivo.

12. Con la información que el sistema provee con respecto a las fallas se puede optimizar el servicio público, así como hacer una mejor planeación de mantenimiento a los teléfonos públicos.

13. Con la realización de este sistema se comprueba que independientemente de los problemas que se tengan en cualquier campo de la ingeniería, no es

necesario una gran cantidad de recurso sino aprovechar los elementos con los que se cuenta.

14. La Ingeniería de Software es el medio más adecuado para solucionar todos los problemas que resulten del desarrollo de los sistemas. Su aplicación además de ser extremadamente útil resulta muy enriquecedora.

15. El sistema de "ALCANCIAS" resulto ser más exitoso de lo que se esperaba por lo que, en un futuro inmediato, se haran necesarias mejores versiones.

Bibliografía.

- **Telecomunicaciones para PC**
  Modems, Software, BBS, Correo electrónico e interconexiones
  John C. Devorak
  Editorial Nock Anis
  México, 1992

- **Modems**
  Manuel González de la Garza
  Editorial Paraninfo

- **Understing Data Communications**
  Gilbert Held
  Editorial John Wiley & Sons
  England, 1991

- **Data Communications, Computer Networks and Open System**
  Freds Halsall
  Editorial Addison - Wesley Publishing Company

- **Microprocessor Intel**
  Editado por Intel

- **The Processor and Coprocessor**
  Robert L. Hummel
  PC Magazine. Programmer's Technical Reference
  1992, Ziff -Davis Press

- **Fundamentos de Bases de Datos**
  Henry F. Korth, Abraham Silberschatz
  Editorial Mc. Graw Hill

- **Ingeniería del Software: Un enfoque práctico**
  Roger S. Pressman
  Editorial Mc Graw Hill
  España, 1993

Ingeniería de Sistemas de Telecomunicaciones
Roger L. Freeman
Editorial Limusa
México, 1989

- **Digital Telephony and Network Integration**
  Bernard E. Keiser, Eugene Stranza
  Editorial Von Nostrand Reinhold Co.
  New York

- **Código sin errores**
  Steve Maguire
  Editorial Microsoft Press

- **Windows Programer's Guide to SERIAL COMMUNICATIONS**
  Timothy S. Monr
  Editorial Sams Publishing

- **Code Base 5.1 User Guide**
  Sequiter Software Inc.
  Editorial Sequiter Software Inc.

- **Code Base 5.1 Reference Guide**
  Sequiter Software Inc.
  Editorial Sequiter Software Inc.

- **Sinopsis del Sistema EXA**
  EXA System Co.
  Editado por EXA System Co.

- **Sinopsis del Sistema MOA**
  MOA System Co.
  Editado por MOA System Co.

- **CSCAPE Rererence Guide**
  Okland Co.
  Editado por Okland Co.

- **CSCAPE User Guide**
  Okland Co.
  Editado por Okland Co.

- **Sistemas Operativos Conceptos y Diseño**
  Milenkowic
  Editorial Mc. Graw Hill

- **El sistema UNIX y sus Aplicaciones**
  José Canosa
  Editorial Publicaciones Marcombo

- **Teleinformática y Redes de Computadoras**
  Antonio Alabau
  Editorial Marcombo

- **Técnicas de Bases de Datos**
  Shakuntala Atre
  Editorial Trillas
  México 1988

- **Mastering Turbo Pascal 6**
  Scott D. Palmer
  Editorial Sybex
  U.S.A., 1991

- **Unix Sistema V Versión 4**
  Shakuntala Atre
  Editorial Mc Graw Hill
  México 1991

- **Turbo C Developer's Library**
  Edward R. Rought, Thomas D. Hoops
  Editorial H. W. Sams
  Indiana 1988

- **Guía del Programador para el IBM PC y PS/2**
  Peter Norton, Richard Wilton
  Editorial Anaya Multimedia
  Madrid 1991

- **C: Guía para usuarios expertos**
  Helbert Shildt
  Editorial Mc. Graw Hill
  México 1989

- **Modems y Servicios en Línea Fácil**
  Sherry Kinkoph
  Editorial Prentice Hall

- **DR DOS 6 Acceso Rápido**

HEMEROGRAFIA

- **El Nuevo MS-DOS 6.0. La actualización más fuerte de su historia.**

- **PC /TIPS BYTE, Abril 1993.**

- **6 Bases de Datos por Relación**
  David Kalman
  PC MAGAZINE No. 50

TESIS CONSULTADAS

- **Automatización de la Oficina de Servicios Escolares de la Facultad de Ingeniería por medio de PC's.**
  UNAM 1990

- **Diseño de un Sistema de Monitoreo de Cajones de Estacionamientos, Basado en el MCU68C11 con interfaz a PC.**
  UNAM'1991

- **Diseño e Implemetación de un Sistema de Administración para el Centro de Cálculo del Instituto Tecnológico de Estudios Superiores Monterrey Campus Ciudad de México.**
  ITESM 1995

ACCESO A INTERNET

•http://www.sequiter.com/sequiter.htm

APENDICES

ALCANCIAS

VERSION 3.4

MANUAL DE OPERACION

INDICE

ÍNDICE

INDICE

1. INTRODUCCION

ALÇANCIAS 3.4 es un sistema de supervisión de líneas de abonado de centrales digitales que tienen conectados teléfonos públicos.

Para su operación el sistema debe ser alimentado con la información asociada a cada teléfono; esto es además del número telefónico, el tipo de teléfono, dirección, razón social y las categorías que debe manejar (en función de las restricciones de tráfico asociadas). Para realizar la carga de una manera simple.

1.1. PROCESOS QUE REALIZA.

El sistema mediante la opción de procesos supervisa:

a) Prueba de líneas telefónicas, con la intención de detectar cuales teléfonos han generado llamadas y cuales no.

b) El estado de los contadores de abonado asociados a las Revisión de categorías que se encarga de ajustar los atributos asociados al teléfono, con la información que tiene declarada en la base de datos, por ejemplo; restricción para tráfico, lada, supervisión, inversión de polaridad, etc... En caso de detectar diferencias en la información declarada en la central en el momento de la lectura, el sistema automáticamente corregirá las categorías.

Esta función es muy útil pero exige mantener bien actualizada la base de datos para evitar posibles errores. Por ejemplo; suspender o reanudar servicio en caso de errores, restringir tráfico.

El sistema también cuenta con una prueba de la línea de abonado asociada al teléfono público, para aquellos casos en los que se detectó que el teléfono no ha sido utilizado, lo cual puede ayudar a dirigir las labores de mantenimiento, si se detectan problemas en la red.

1.2. PROCESOS ADICIONALES.

El sistema cuenta a su vez con una serie de procesos adicionales que sirven para dar mantenimiento a los teléfonos públicos de una manera más fácil que utilizando una terminal de MOA:

a) Suspensión de Alcancías: El cual permite modificar las categorías necesarias a la central de telefónica para poder realizar una suspensión de línea, actualizando la nueva condición en caso de estar dada de alta en base de datos.

b) Reanudación de Alcancías: El cual permite modificar las categorías necesarias a la central telefónica para poder realizar una reanudación de línea, actualizando la nueva condición en caso de estar dada de alta en la base de datos.

c) Consulta de categorías: El cual permite consultar las categorías asignadas a un teléfono público en la central telefónica.

d) Cambio de categorías. El cual permite cambiar una serie de categorías preestablecidas para un teléfono público enviando una serie de comandos adecuados para que la central tome en cuenta los cambio .

e) Desbloqueo de línea. El cual permite desbloquear un número telefónico que la central haya puesto fuera de servicio por una falla en la línea, enviando el comando necesario.

f) Prueba de línea de ALCANCIAS: La cual permite verificar el funcionamiento de la línea, detectando posibles fallas en la línea telefónica.

La mayoría de las características más importantes se puede declarar o restringir dentro del Menú de opciones, razón por la cual recomendamos ampliamente su estudio detallado.

Asimismo, el hecho de declarar los datos asociados al teléfono como son la dirección de ubicación y razón social, permite que los reportes de supervisión incluyan estos datos facilitando los programas de recorridos.

ALCANCIAS 3.4 tiene la gran ventaja de manejar la base de datos en formato FOXPRO lo que permite tener libertad de explotación de datos postproceso, y añadir nuevos registros, sin suprimir los preestablecidos.

Alcancías proporciona diferentes índices de ordenamiento para clasificar por zonas los diferentes teléfonos, de tal manera que los reportes de

mantenimiento pueden ser agrupados en función del personal que deba atenderlos.

Por último cabe mencionar que si se desea obtener la verdadera potencialidad del sistema en lo que se refiere a la prueba de líneas de abonado, Telmex debe asegurarse de seleccionar aparatos telefónicos que pueden ser "vistos" por la central EXA, o en su defecto, instalar un desconectador "DIT" lo cual asegura la confiabilidad de la línea telefónica.

2. SISTEMA ALCANCIAS

2.1 REQUERIMIENTOS DEL SISTEMA.

2.1.1 REQUERIMIENTOS DE SOFTWARE Y HARDWARE.

1. Computadora PC o compatible 486 en adelante, 20 MB en disco duro, 1MB de memoria principal, con puerto serial RS232C.
2. Sistema operativo instalado.
3. Foxpro2 instalado de preferencia
4. Modem conectado, configurado y probado a la(s) central(es) que se esta(n) accesando.
5. Impresora con su manual

2.1.2 REQUERIMIENTOS DE INFORMACION DISPONIBLE.

1. Claves de acceso a las centrales
2. Si se trata de línea conmutada, favor de poner el número telefónico del Modem para pode conectarse
3. Plan de numeración de las líneas a controlar con los siguientes datos:

a) Numeración de la central a la que pertenece, y Login si es necesario por ejemplo:

62100000 a 62109999, Central Garmendia, Login GARM4L

b) Categorías preestablecidas por cada tipo de aparato por ejemplo para Multimodal

Sistema 2199

BADP=2 CBOX=5 ORPT=5 SSIG=7 TRSTR=1 LNCHAR=1

HM=1

EXA

CBA=2, TLI=3 BIC=1 SSI=1 OBA=0 TCL=11

BOC=0 TBO=2

c) Tipo de central que utiliza cada numeración, por ejemplo elegir el tipo de central y comunicación, de entre las siguientes:

Sistema 2199 vía Man-Machine

Sistema 2199 vía MPTMON

EXA vía MOA

EXA vía conexión directa IOC11

EXA vía conexión directa IOC4

Nota: Se debe llenar la tabla contenida en el apéndice H

2.1.3 CONOCIMIENTOS BASICOS DEL OPERADOR.

1. La persona que realizará la operación del sistema deberá tener conocimientos básicos del uso de una computadora personal, sistema operativo MS-DOS, y de algún sistema de comunicación como CARBON COPY o BLAST y saber accesar a centrales digitales.
2. Conocimientos en FOX-PRO preferencia.

2.2.-INSTALACION.

2.2.1 LICENCIA.

El sistema ALCANCIAS que usted adquirió consiste del software que se encuentra en el disco flexible ALCANCIAS V 3.4 especialmente configurado para probar teléfonos de una localidad, así como el presente manual. La licencia esta limitada solo para trabajar en su ciudad, por lo que si se desea utilizarlo para otras ciudades, se necesita comprar nueva licencia.

2.2.2 COMO SE INSTALA EL SISTEMA.

Para instalar el sistema Alcancia 3.0 por favor complete los siguientes pasos (un ejemplo típico es mostrado a continuación):

Importante: Respalde primero bases de datos y archivos *.dat antes de hacer este proceso, porque de lo contrario se perderán sus datos

1) Cambiarse al disco duro donde se desea instalar el sistema.

C:

2) Crear el directorio donde se va a instalar el sistema.

MD ALCAN3

3.) Cambiarse al directorio que acaba de crear.

CD ALCAN3

4.) Copiar todos los archivos de su disco de instalación.

COPY A:*.*

5) Configurar el sistema con el passwords de acceso al sistema

6) Configurar el sistema con las series validas en la localidad

7) Configurar el sistema para que se comunique con la central de la siguiente manera:

7.1) Teclear desde sistema operativo.

C:\ALCAN3>SIMPLE

7.2) Teclear el archivo a configurar por ejemplo SIMPLEP1.CFG.

El sistema tiene archivos de configuración ya armados como lo son:

SIMPLEP1.CFG para el COM1, SIMPLEP2.CFG para el COM2

7.3) Teclear [ALT][S] para modificar las opciones de comunicación hasta lograr del sistema se comunique con la central ([ALT][B] es para mandar break):

<1> Serial Port : 1

<2> Baud Rate : 1200

<3> Parity : 7E1

<4> Lock Serial Port: No

<5> Init. String :

<6> Hangup String : ~~~+++~~~ATH0^M

<7> Dial Prefix : ATDT

<S> Save and Quit

<Q> Quit

Please make a selection:

Eligiendo la opción con el número y modificándola usted podrá comunicarse

con la central, es necesario saber los comandos adecuados.

Recuerde salvar la configuración con la opción <S> Save and Quit

Para salir del sistema teclee [CTRL][C]

Este archivo que usted creó o modificó es el que debe estar incorporado enConfiguración de Rango en el campo PUERTO.

2.3. ENTRADA AL SISTEMA.

Una vez estando en el sistema operativo, ejecutar el programa:
C:\ALCAN40> ALCAN
Y oprimir [ENTER] para entrar al sistema.

2.3.1 PASSWORD DE ENTRADA.

Al momento de empezar a operar el sistema de Alcancías se recomienda dar de alta el PASSWORD de entrada en el menú de opciones, en el cual se deben de introducir máximo ocho caracteres. El password es sensible al contexto, lo que significa que hace distinción entre minúsculas y mayúsculas. por esto, siempre se debe escribir conforme fue dado de alta.

Una vez dado de alta el PASSWORD la primera condición necesaria para operar el sistema de Alcancías es teclear correctamente el password de acceso. en la figura 2.3.1.1.(ver figura siguiente) podemos ver la pantalla para introducirlo.

En caso de olvidarse el ultimo PASSWORD dado de alta el usuario tendrá que dirigirse a las oficinas de para que le sea reintegrado un nuevo password conocido.

2.4 MANEJO DE MENUS.

Favor de consultar tema: Implementación de cada módulo

4. APENDICE A.

CONDICIONES DE ERROR EN PRUEBA DE LINEA.

0:	"Sin prueba hasta el momento"
1:	"No pudo verificar MOA: Gpo. de 2000 bloqueado"
2:	"No pudo verificar MOA: Falla de Hardware"
3:	"No pudo verificar MOA: Línea ocupada"
4:5:8:13:	"No pudo verificar MOA: Fault code %d"
6:	"no pudo verificar MOA: línea encapsulada"
7:	"No pudo verificar MOA: Línea bloqueada manualmente"
9:	"No pudo verificar MOA: Grupo PBX"
10:	"No pudo verificar MOA: Líneas no disponible"
11:	" No pudo verificar MOA: Disp. de Medición ocupado"
12:	" No pudo verificar MOA: Disp. de medición bloqueados"
20:	"Circuito, tierra y Batería Húmeda"
21:	"Circuito, tierra Húmeda"
22:	"Circuito, tierra y Batería"
23:	"Circuito a Batería Húmeda"
24:	"Circuito Húmedo"
25	"Circuito Húmedo y Batería"
26:	"Circuito a Batería Húmeda y Tierra"
27:	"Circuito Húmedo y Tierra"
28:	"Circuito Húmedo, Tierra y Batería"
29:	"Tierra y Batería Húmeda"
30:	"Tierra Húmeda"
31:	"Tierra Húmeda y Batería"
32:	"Batería Húmeda"

33:	"Linea en Buen Estado"
34:	"Batería"
35:	"Batería Húmeda y Tierra"
36:	"Tierra"
37:	"Tierra y Batería"
38:	"Batería Tierra Húmedas y Circuito"
39:	"Tierra Húmeda y Circuito"
40:	"Tierra Húmeda y Circuito a Batería"
41:	"Batería Húmeda y Circuito"
42:	"Circuito"
43:	"Circuito a Batería"
44:	"Batería Húmeda y Circuito a Tierra"
45:	"Circuito a Tierra"
46:	"Circuito a Tierra y Batería"
60:	"En cruce con abonado"
61:	"Roto hacia afuera"
62:	"Roto hacia adentro"
80:	"Error no reconocido"
81:	"Fault Code Irreconocible"
82:	"No corresponde número"
83:	"Falla de conexión del MOA Y LA CENTRAL
OTRO	"Error No reconocido.

APENDICE B.

SOLUCION DE PROBLEMAS.

Problema: No permite probar algunos números dice: Número Invalido.

Solución: Dar de alta los teléfonos en la Base de Datos, para que el sistema no restrinja su operación.

Problema: Cuando se desea probar un número dice: Número fuera de Alcance.

Solución: El sistema esta restringido a las claves lada de la localidad u debe introducir el número a probar con su clave lada ejemplo 62180107, si desea nuevas claves lada, favor de contactar con el proveedor.

Problema: Al ejecutar el programa dice Not Enogh Memory.

Solución: Modificar el autoexec.bat y config.sys para dejar el menor número de programas residentes y verificar en el config.sys
files=32
buffers=32

Problema: No se considera los cambios en el menú de Bases de Datos.

Solución: Al momento de introducir cambios, dar <Enter> hasta el último registro, porque si se sale uno con <Escape> el sistema considerará que se desean anular los cambios programados

Problema: No reconoce el nuevo password.

Solución: Verificar que se este escribiendo correctamente el password checando si se escribió con mayúsculas o minúsculas.

Problema: Aparece pura basura en el cuadro que dice: Respuesta de la central.

Solución: Verificar individualmente la central que se está accesando.

Problema: Durante (revisión continua de contadores o de categorías) recorre unos números muy rápidos.

Solución: Verificar individualmente la central que se está accesando.

Problema: Intento dar de alta nuevos números de esta localidad pero no los acepta.

Solución: Posiblemente no este configurado ese rango de teléfonos, por favor consultar configuración de rangos e insertar ese nuevo rango en el próximo renglón disponible.

Problema: Con el password actual no se entran a todas las opciones.

Solución Favor de conseguir el password de operador, este tiene permisos para todas las opciones.

Problema: Estaba trabajado bien cuando de un momento a otro, sed quedo atorado en una prueba.

Solución: Posiblemente si es 2199 esta dando respuestas muy lentas o si es EXA el operador saco un momento el puerto de funcionamiento.

Problema: Al dar de baja un número, sigue en la pantalla.

Solución: La baja es como el Foxpro2 donde primero se marca el número a suprimir y después se da de baja, en el caso del sistema, para hacer válidos los movimientos de baja hay que salir del

menú de Base de datos.

Problema: Deseo cambiar el número telefónico pero guardar todos los datos.

Solución: Utilice la opción de llave en el menú de Base de datos, este permite modificar el número telefónico de los datos que aparecen en pantalla.

Problema: Aparece en la pantalla error de escritura escribe dispositivo PRN Anular, Repetir, Ignorar; Descartar?

Solución: Se encuentra desconectada la impresora, favor de verificar su conexión.

Problema: Aparece error XX en ALCANCIAS.DBF o error XX en ALCANCIAS.CDX.

Solución: Restaurar el último respaldo de estos dos archivos.

APENDICE C.

CARACTERÍSTICAS DE TIPO DE APARATO.

TIPO DE APARATO	DESCRIPCIÓN
PUBLICO 1	Ver Anexo 1 Inventario Tel Pública, Alarma campo 3
MON LADATEL	Ver Anexo 1 Inventario Tel Pública, Alarma campo 3
CREDITO	Ver Anexo 1 Inventario Tel Pública, Alarma campo 3
MON Y CRED	Ver Anexo 1 Inventario Tel Pública, Alarma campo 3
DEBITO	Ver Anexo 1 Inventario Tel Pública, Alarma campo 3
CRED Y DEB	Ver Anexo 1 Inventario Tel Pública, Alarma campo 3
PUBLICOS 2	Ver Anexo 1 Inventario Tel Pública, Alarma campo 3
TPTC	Ver Anexo 1 Inventario Tel Pública, Alarma campo 3
PATROCINADO	Ver Anexo 1 Inventario Tel Pública, Alarma campo 3
AMPER	Ver Anexo 1 Inventario Tel Pública, Alarma campo 3

APENDICE D.

TIEMPOS DE RESPUESTA DE CENTRALES.

	MOA	IOC12	IOC4	S-2199
SUSPENSION	18 SEG	9 SEG		25 SEG
REANUDACION	18 SEG	9 SEG		24 SEG
C. DE CATEGORIAS	16 SEG	8 SEG		16 SEG
C. DE CATEGORIAS	18 SEG	9 SEG		26 SEG
C. DE CONTADORES	14 SEG	7 SEG		16 SEG
PRUEBA DE LINEA	25 SEG	13 SEG		31 SEG

APENDICE E1.

DESCRIPCION DE CATEGORIAS (EXA).

CATEGORIA	RESPUESTA DE CENT.	DESCRIPCION
CBA-0	CBA-0	RESTRICCION DE LARGA DISTANCIA (INACTIVA)
CBA-2	CBA-2	RESTRICCION DE LARGA DISTANCIA (ACTIVA)
TLI-0	TLI-0	INVERSION DE POLARIDAD (INACTIVA)
TLI-3	TLI-3	INVERSION DE POLARIDAD (ACTIVA)
BIC-0	BIC-0	RESTRICCION DE LLAMADAS DE ENTRADA (INACTIVA)
BIC-1	BIC-1	RESTRICCION DE LLAMADAS DE ENTRADA (ACTIVA)
SSI-0	SSI-0	TIPO DE TECLADO (SOLO DISCO)
SSI-1	SSI-1	TIPO DE TECLADO (DISCO Y TECLADO)
OBA-0	OBA-0	CANDADO DE LINEA (INACTIVO)
OBA-5	OBA-5	CANDADO DE LINEA (ACTIVO)
TCL-0	TCL-0	TELEFONO PUBLICO (SIN IDENTIFICACION DE 999999)
TCL-11	TCL-11	TELEFONO PUBLICO (CON IDENTIFICACION DE 999999)
ICS-0	ICS-0	MENSAJE DE SUSPENSION (INACTIVO)
ICS-6	ICS-6	MENSAJE DE SUSPENSION (ACTIVO)
BOC-0	BOC-0	INHIBICION DE TONO DE MARCAR (INACTIVO)
BOC-1	BOC-1	INHIBICION DE TONO DE MARCAR (ACTIVO)
TBO-0	TBO-0	SUSPENSION DE LINEA (INACTIVA)
TBO-1	TBO-1	SUSPENSION DE LINEA (ACTIVA)

*Nota: Todos los valores para activar pueden cambiar dependiendo de la ciudad

APENDICE E2.

DESCRIPCION DE CATEGORIAS (SISTEMA 2199).

CATEGORIA	RESPUESTA DE CENTRAL	DESCRIPCION
BADP=1	MPAG	Suspensión (activa)
BADP=2	REM	Suspensión (inactiva)
CBOX=1	ALC _ _ _ _ _: _ _INDIV	Alcancía 1
CBOX=2	ALC _ _ _ _ _: _ _INDIV-E	Alcancía 2
CBOX=3	ALC _ _ _ _ _: _ _MULTI	Alcancía 3
CBOX=4	ALC _ _ _ _ _: _ _MULTI-E	Alcancía 4
CBOX=11	ALC _ _ _ _ _: _ _INDIV_ _& _I 9999	Alcancía 1e Identif 999
CBOX=21	ALC _ _ _ _ _: _ _INDIV_ E _& _I 9999	Alcancia2 e Identif 999
CBOX=31	ALC _ _ _ _ _: _ _MULTI_ _& _I 9999	Alcancia3 e Identif 999
CBOX=41	ALC _ _ _ _ _: _ _MULTI_ E _& _I 9999	Alcancia4 e Identif 999
CBOX=5	REM	No tipo alcancia
ORPT=4	RESTPORT _: _ _LOCAL	Rest llam.salida local
ORPT=67	RESTPORT _: _ _NACIONAL	Rest.llam.salida nacional
ORPT=0	REM	Sin Rest.llam.salida
ORPT=2	RESTPORT _: _ _NLLA	Rest.llam.nlla
ORPT=3	RESTPORT _: _ _SESP	Rest.llam.sesp
ORPT=5	RESTPORT _: _ _ZONA	Rest.llam.por zona
ORPT=7	RESTPORT _: _ _CONT	Rest llam.por cont
SSIG=1	SENAB _ _ _: _ _AP_D _ISCO Z	Tipo de Teclado 1
SSIG=2	SENAB _ _ _: _ _AP _D _ISCO X	Tipo de Teclado 2
SSIG=3	SENAB _ _ _: _ _AP _B _OTONERA _10	Tipo de Teclado 3

SSIG=4	SENAB _ _ _: _ _AP _B _OTONERA _12	Tipo de Teclado 4
SSIG=5	SENAB _ _ _: _ _AP _B _OTONERA _16	Tipo de Teclado 5
SSIG=6	SENAB _ _ _: _ _AP _B _COMB _Z-10	Tipo de Teclado 6
SSIG=7	SENAB _ _ _: _ _AP _B _COMB _Z-12	Tipo de Teclado 7
SSIG=8	SENAB _ _ _: _ _AP _B _COMB _Z-16	Tipo de Teclado 8
SSIG=9	SENAB _ _ _: _ _AP _B _COMB _X-10	Tipo de Teclado 9
SSIG=10	SENAB _ _ _: _ _AP _B _COMB _X-12	Tipo de Teclado 10
SSIG=11	SENAB _ _ _: _ _AP _B _COMB _X-16	Tipo de Teclado 11
TRSTR=1	RESTERM	Restricción llam. terminal
TRSTR=0	REM	Sin Rest llam. terminal
LNCHAR=7	CARACTLN _: _ _POLAR	Cambio de Polarida
LNCHAR=1	REM	Sin Cambio de Polaridad
LNCHAR=2	CARACTLN _= _ _NOPBALN	
LNCHAR=3	CARACTLN _= _ _HIBRIDA	
LNCHAR=4	CARACTLN _= _ _EXBUCLE	
LNCHAR=5	CARACTLN _= _ _EXBUCLE	
HM=1	MEDPRIV _:_ _ POLAR	Cambio de Polaridad
HM=5	REM	Sin cambio de Polaridad

APENDICE F.

DESCRIPCION DE CATEGORIAS.

SISTEMA2199

	BADP	CBOX	ORPT	SSIG	TRSTR	LNCHAR	HM
PUBLICO 1							
MON LADATEL							
CREDITO							
MON Y CRED							
DEBITO							
CRED Y DEB							
PUBLICOS 2							
TPTC							
PATROCINADO							
AMPER							
* (SUSPENDIDO)							

EXA.

	CBA	TLI	BIC	SSI	OBA	TCL	ICS	BOC	TBO
PUBLICO 1									
MON LADATEL									
CREDITO									
MON Y CRED									
DEBITO									
CRED Y DEB									
PUBLICOS 2									
TPTC									
PATROCINADO									
AMPER									
* (SUSPENDIDO)									

Manual Técnico

Alcancías X.X

El siguiente documento contiene información técnica acerca de configuración del sistema para reconozca el modem y se pueda conectar a la central telefónica

ÍNDICE

==========

1. VERIFICANDO MANUALMENTE QUE EL MODEM ESTE RESPONDIENDO

Para checar que se encuentra conectado el modem realizar los siguientes pasos:

1.1 Cambiarse al subdirectorio donde se encuentra instalado ALCANCIAS X.X

```
C:
```

1.2 Cambiarse al subdirectorio donde esta instalado ALCANCIAS X.**X**

```
C:\>CD ALCAN
```

1.3 Entrar al programa de comunicaciones SIMPLE.EXE

```
C:\ALCAN>SIMPLE
```

1.4 Teclear cuando pregunta:
ARCHIVO DE CONFIGURACIÓN:

Escribir el nombre del archivo donde se guarda la configuración de comunicación de cada central (IMPORTANTE: Por cada modem ubicado en las centrales que tenga diferente velocidad, o diferente paridad, es necesario crear un archivo independiente para cada uno de estos, incluso si la computadora tiene mas de un modem conectado a esta, por cada COM diferente hay que crear un archivo nuevo)

Por default el sistema tiene los siguientes archivos, pero usted puede crear sus propios archivos.

Nombre del Archivo	Puerto	Velocidad	Paridad	Bits	Stopbits
SIMPLEP1.CFG	COM1	1200	Even	7	1
SIMPLEP2.CFG	COM2	1200	Even	7	1
SIMPLEP3.CFG	COM3	1200	Even	7	1
SIMPLEP4.CFG	COM4	1200	Even	7	1

1.5 Teclear [ALT][S] para modificar las opciones de comunicación hasta lograr que el sistema se comunique con el modem; una forma de verificar que el modem esta respondiendo es mandar el comando

AT

Y el modem deber responder:

OK

Si el modem no respondiera modificar los parámetros del punto 1.6

1.6 Cuando se teclea [ALT][S] aparecen las siguientes opciones

<1> Serial Port : 2

<2> Baud Rate : 1200

<3> Parity : 7E1

<4> Lock Serial Port: No

<5> Init. String :

<6> Hangup String : ~~~+++~~~ATH0^M

<7> Dial Prefix : ATDT

<S> Save and Quit

<Q> Quit

Please make a selection:

donde:

Serial port es el COM donde esta conectado el modem

Baud rate es la velocidad de comunicaciones

Parity es la paridad, solo hay dos variantes [Bits=7,Even,Stop=1] y [Bits=8,None,Stop=1]

Las demás opciones no se varían por lo regular pues, la configuración por default lo utilizan la mayoría de los modems.

Recuerde una vez comunicado con el modem salvar en archivo con la opción:

<S> Save and Quit

[CTRL][C] Usted puede regresar al sistema operativo

[CTRL][X] Usted puede regresar al sistema operativo pero colgando

[ALT][B] Puede mandar un Break;

[ALT][S] Puede elegir cambiar la configuración de comunicación

Nota: A veces para que surtan efecto los cambios hay que entrar de nueva cuenta al programa de comunicaciones simple (Paso 1.3)

==========

2. VERIFICANDO MANUALMENTE QUE LA CENTRAL ESTE RESPONDIENDO

2.1 Llamando a la central

Una vez comprobado que el modem responde (paso 1) y si la comunicación fuera por línea conmutada hay que marcar al teléfono del modem, por

ejemplo:

ATDT131333

- Si suena ocupado, llamar a personal de centrales para ver tiempos de ocupación
- Si no responde, llamar a personal para ver si esta encendido el modem
- Si responde CONECTED 1200 o a cualquier velocidad, seguir el paso 2.2 de acuerdo al tipa de central que se esta probando

2.2.1 Central tipo MOA (Ver ejemplo de Apéndice A)

[ALT][B] Para enviar un break; hasta que responda la central con símbolo mayor [<]

LOGOF; Para salir de cualquier central.

[Backspace] Dar varias veces esta tecla hasta que la central pida el password de acceso, seguido con símbolo de mayor [<]

TELPUB; teclear el password de acceso seguido de un ;

YANEZF> Si se trata de un MOA controla varias centrales indicar la central que se va a entrar en particular, tecleando el Login de acceso terminado con el símbolo de mayor[>]

CONT:SNB=62140000;

Teclear un comando valido por ejemplo el CONT que sirve para leer los contadores de un teléfono en particular, ver primero si responde el comando con 8 dígitos, sino probar el mismo comando pero sin la clave lada.

2.2.2 Central tipo IOC12 (Ver ejemplo de Apéndice A)

[ENTER] Para que responda la central
CONT:SNB=62140000;
Teclear un comando valido por ejemplo el CONT que sirve para leer los contadores de un teléfono en particular, ver primero si responde el comando con 8 dígitos, sino probar el mismo comando pero sin la clave lada
[ENTER] Para que haga caso del comando
[Ctrl][D] Para dejar el acceso a la central

2.2.3 Central tipo IOC4 (Ver ejemplo de Apéndice A)

[Ctrl][E] Para tomar el acceso a la central
CONT:SNB=62140000;
Teclear un comando valido por ejemplo el CONT que sirve para leer los contadores de un teléfono en particular, ver primero si responde el comando
con 8 dígitos, sino probar el mismo comando pero sin la clave lada
[ENTER] Para que haga caso del comando
[Ctrl][D] Para dejar el acceso a la central

2.2.4 Central Sistema 2199 Man-Machine (Ver ejemplo de Apéndice A)

[ALT][B] Para enviar un break; hasta que responda la central con símbolo mayor [<]

PW01; Teclear el password de acceso seguido de ;

78:1=K'16101692;

Teclear un comando valido por ejemplo el 78 que sirve para leer los parámetros generales de un teléfono en particular, ver primero si responde el comando con 8 dígitos, sino probar el mismo comando pero sin la clave lada

[Ctrl][X] Para dejar el acceso a la central

2.2.5 Central Sistema 2199 MPTMON (Ver ejemplo de Apéndice A)

[ALT][B] Para enviar un break; hasta que responda la central con símbolo mayor [<]

MM; Teclear MM; para acceso MPTMON

[ENTER] Para que haga caso al comando

78:1=K'16101692;

Teclear un comando valido por ejemplo el 78 que sirve para leer los parámetros generales de un teléfono en particular, ver primero si responde el comando con 8 dígitos, sino probar el mismo comando pero sin la clave lada

[ENTER] Para que haga caso al comando

[Ctrl][X] Para dejar el acceso a la central

==========

3.VERIFICANDO QUE EL SISTEMA ESTE RESPONDIENDO AUTOMÁTICAMENTE

Para el correcto funcionamiento del sistema es primordial tener bien configurado el sistema corriendo el programa ALCAN.EXE entrando a OPCIONES y después a CONFIGURACIÓN DE SERIES VALIDAS, el primer dígito que pide diciendo NUMERO DE RENGLÓN?, es una línea de configuración

LimInf Límite inferior de la serie

LimSup Límite superior de la serie

ND Número de dígitos de los teléfonos a probar con los cuales responde la central, por default es 8 (Con lada) o (6) Sin lada

TCentral Elegir con <Barra espaciadora> la opción apropiada

MOA (AXE) Sistema MOA

IOC12 (AXE) Sistema IOC12

IOC4 (AXE) Sistema IOC4

SIST2199 (Sistema 2199) Man-Machine

SIST2199MTM (Sistema 2199) MPTMON

Password Se introduce el password de acceso para MOA

Login Se usa para entrar a cierta central desde el MOA, en otros tipos de centrales no se usa, aunque es recomendable establecer diferencia de centrales, mediante nombres diferentes por central

ArchivoCFG Es el archivo donde se encuentra la configuración del modem creado mediante el paso 1 y 2

NoModem El número telefónico del modem a conectarse (0) si es directo.

==========

A. APÉNDICE A:EJEMPLOS TÍPICOS DE COMUNICACIONES

---------- A.1 COMUNICACIONES MOA ----------

```
ATDT9162131444
CONNECT 1200
LOGGED-OFF DEVICE UPMMAZ    AT 19:33(0)
PLEASE ENTER IDENTIFICATION <
@@@@@@@@@@@@@MMMMMMMMMMM
MOAHMO    1996-05-23 19:33 LOGGED-ON UPMMAZ    OUTDEV=UPMMAZ

<YANEZL>
CONNECTED TO YANEZL    AT 19:33 UPMMAZ    OUTDEV=UPMMAZ

<CONT:SNB=62116575;

CALL METER VALUES READING
INDIVIDUAL

SNB          A  CMVL      CMVS      TYPE
62116575     4  00002177  00000000

END

<LOGOF;
LOGGED-OFF DEVICE UPMMAZ    AT 19:34
PLEASE ENTER IDENTIFICATION
<
OK
```

---------- A.2 COMUNICACIONES IOC12 e IOC4 ----------

```
ATDT189999
CONNECT 1200

NG

NG

NN00
WO      18-6/COP2/32/0709/014   AT-6      TIME 960523 2025  PAGE    1
      <
      <CONT:SNB=16170350;
      CALL METER VALUES READING
      INDIVIDUAL

      SNB          A  CMVL      CMVS      TYPE
      16170350     4  00000007  00000000

      END
      <
```

---------- A.3 COMUNICACIONES SISTEMA 2199 MAN-MACHINE----------

```
CLAVE DE ACCESO <
CORTINEZ          1996-05-23  19:41:45  JU
<78:1=K'47000;
SEC=1719.960523 9002
COM=0078
COMANDO ENVIADO A EJECUCION


           9000
RESULTADOS A CONTINUACION



CORTINEZ          1996-05-23  19:42:03  JU
SEC=1719.960523 0076  .
ADMINISTRACION DE ABONADOS

VISUALIZAR-ABONADO-INDIVIDUAL                        CON EXITO
RESULTADO FINAL
-----------------------------------------------------------------


ND                   NE  FISICO              CE LOGICO
63147000            H'727 (1831 ) & 209     H'1BB0(7088 )

MEDIDORES:  2584     4093     0       1       0       0

ESPECIFICACIONES:
      SENAB    :  AP COMB Z-16

CLASES DE LINEA :
      CTRLREM :  RECAUT

REPORTE COMPLETO =        0076
```

A.4 COMUNICACIONES SISTEMA 2199 MPTMON

```
<78:1=K'101692;
SEC=0873.960523 9002
COM=0078
COMANDO ENVIADO A EJECUCION


          9000
RESULTADOS A CONTINUACION

RAZA III       1996-05-23  20:21:21  JU
SEC=0873.960523 0076
ADMINISTRACION DE ABONADOS

VISUALIZAR-ABONADO-INDIVIDUAL                        CON EXITO
RESULTADO FINAL
-----------------------------------------------------------------


 ND                  NE  FISICO              CE LOGICO
 16101692           H'1621(5665 ) & 157     H'2A50(10832)

 MEDIDORES:  188      656      0       3       0       0

 ESPECIFICACIONES:
      SENAB   :  AP COMB Z-12

 CLASES DE LINEA :
      MEDPRIV :  POLAR
      RESTERM

 REPORTE COMPLETO =        0076
```

```
<(3)78:1=K'16101692;
SEC=0874.960523 9002
COM=0078
COMANDO ENVIADO A EJECUCION

           9000
RESULTADOS A CONTINUACION

RAZA III       1996-05-23  20:21:57  JU
SEC=0874.960523 0076
ADMINISTRACION DE ABONADOS

VISUALIZAR-ABONADO-INDIVIDUAL                          CON EXITO
RESULTADO FINAL
-----------------------------------------------------------------


   ND                    NE  FISICO                 CE LOGICO
   16101692             H'1621(5665 ) & 157      H'2A50(10832)

   MEDIDORES:  188      656      0        3        0        0

   ESPECIFICACIONES:
         SENAB    :  AP COMB Z-12

   CLASES DE LINEA :
         MEDPRIV :  POLAR
         RESTERM

   REPORTE COMPLETO =         0076
        >
```

```
// ********************************************************************************
// SISTEMA: ALCANCIAS 3.4 (AXE, SISTEMA 12)
//
// ARCHIVO: ALCAN.C
//
// FECHA:  06/10/95
// ********************************************************************************
#include <stdio.h>
#include <process.h>
#include <conio.h>
#include <string.h>
#include <mem.h>
#include <time.h>
#include <cscape.h>
#include <cstime.h>
#include <teddecl.h>
#include "bases.h"
#include "nomencla.h"
#include "gralfunc.h"
#include "gralfunx.h"
#include "ladcod.h"
#include "datosx.h"


boolean myspc_Esc(sed_type sed,int scancode);
struct tm tminit;
struct tm tminit2;
extern struct tm date;

// ************* Variables usadas en ALCAN.C ************************
extern char passwordnew[S_PSW];
extern char passwordacc[S_PSW];
extern char passwordoper[S_PSW];
extern char passwordcnt[S_PSW];
extern char passwordcon[S_PSW];
extern sed_type sENCAB1,sMEDIO1,sPIE1;


int inittimmer()
{
        tm_Now(&tminit);
        return 1;
}

int expiratimmer()
{
        struct tm tmahora;
        long tiempo=0L;

        tiempo=0L;
        tm_Now(&tmahora);
        tiempo=tm_ElapSecs(&tminit,&tmahora);
        if (tiempo<0L)
                tiempo=0;
```

```
        if (tiempo>=120) {
                gotoxy(3,24);
                printf("Desconexion o ruido");
                return 1;
        }
        return 0;
}

int inittimmer2()
{
        tm_Now(&tminit2);
        return 1;
}

int expiratimmer2()
{
        struct tm tmahora;
        long tiempo=0L;

        tiempo=0L;
        tm_Now(&tmahora);
        tiempo=tm_ElapSecs(&tminit,&tmahora);
        if (tiempo<0L)
                tiempo=0;
        if (tiempo>=240) {
                gotoxy(3,24);
                printf("Desconexion o ruido");
                return 1;
        }
        return 0;
}

/* ***************** PANTALLAS DE PRESENTACION ****************** */
extern ALCANBTRV data_buf1;
extern FILE *fhistmov;
extern char strout[255];


int iENCAB1() {
 menu_type      menu;
 int            ret;
        if ((menu = menu_Open())==NULL)
                generror(6,0);
        menu_Printf(menu, "ALCANCIAS 3.4 (SISA)");
        menu_Flush(menu);
        if ((sENCAB1 = sed_Open(menu))==NULL)
                generror(7,0);
        sed_SetColors(sENCAB1, 0x1b, 0x1b, 0x1b);
        sed_SetBorder(sENCAB1, bd_prompt);
        sed_SetPosition(sENCAB1, 0, 0);
        sed_SetHeight(sENCAB1, 1);
        sed_SetWidth(sENCAB1, 78);
```

```
        sed_Repaint(sENCAB1);
        ret = sed_Go(sENCAB1);
        return(ret);
}

int iMEDIO1() {
 menu_type      menu;
 int            ret;

        if ((menu = menu_Open())==NULL)
                generror(6,0);
        menu_Printf(menu, "@[23, |MONITOREO DE TELEFONOS PUBLICOS\n");
        menu_Flush(menu);
        if ((sMEDIO1 = sed_Open(menu))==NULL)
                generror(7,0);
        sed_SetColors(sMEDIO1, 0x1b, 0x1b, 0x1b);
        sed_SetBorder(sMEDIO1, bd_prompt);
        sed_SetPosition(sMEDIO1, 3, 0);
        sed_SetHeight(sMEDIO1, 17);
        sed_SetWidth(sMEDIO1, 78);
        sed_Repaint(sMEDIO1);
        ret = sed_Go(sMEDIO1);
        return(ret);
}

int iPIE1() {
 menu_type      menu;
 int            ret;

        if ((menu = menu_Open())==NULL)
                generror(6,0);
        menu_Flush(menu);
        if ((sPIE1 = sed_Open(menu))==NULL)
                generror(7,0);
        sed_SetColors(sPIE1, 0x1b, 0x1b, 0x1c);
        sed_SetBorder(sPIE1, bd_prompt);
        sed_SetBorderTitle(sPIE1, "MENSAJES");
        sed_SetPosition(sPIE1, 22, 0);
        sed_SetHeight(sPIE1, 1);
        sed_SetWidth(sPIE1, 78);
        sed_Repaint(sPIE1);
        ret = sed_Go(sPIE1);
        return(ret);
}

int iPASSENT(int opcion) {
        menu_type       menu;
        sed_typesed;
        int             ret;

        limpiachar(passwordent,sizeof(passwordent));
        limpiachar(passwordnew,sizeof(passwordnew));
        if ((menu = menu_Open())==NULL)
                generror(6,0);
```

```
        if (opcion==1)
                menu_Printf(menu, "Password de entrada
@[]#########]\n",passwordent,&secure_funcs);
        else
                menu_Printf(menu, "Teclee el nuevo password
@[]#########]\n",passwordnew,&secure_funcs);
        menu_Flush(menu);
        if ((    sed = sed_Open(menu))==NULL)
                generror(7,0);
        sed_SetColors(sed, 0x1b, 0x31, 0x70);
        sed_SetBorder(sed, bd_prompt);
        sed_SetPosition(sed, 8, 22);
        sed_SetHeight(sed, 4);
        sed_SetWidth(sed, 38);
        sed_SetShadow(sed, 1);
        sed_Repaint(sed);
        ret = sed_Go(sed);
        sed_Close(sed);
        return(ret);
}

int iPASSCON() {
        menu_type       menu;
        sed_typesed;
        int             ret;

        if ((menu = menu_Open())==NULL)
                generror(6,0);
        menu_Printf(menu, "Confirmación del password @[]#########]"  ,passwordcon,&secure_funcs);
        menu_Flush(menu);
        if ((sed = sed_Open(menu))==NULL)
                generror(7,0);
        sed_SetColors(sed, 0x1b, 0x31, 0x70);
        sed_SetBorder(sed, bd_prompt);
        sed_SetPosition(sed, 8, 22);
        sed_SetHeight(sed, 4);
        sed_SetWidth(sed, 38);
        sed_SetShadow(sed, 1);
        sed_Repaint(sed);
        ret = sed_Go(sed);
        sed_Close(sed);
        return(ret);
}


int iCONFIRMA_SALIR()
        // Mod Julio
{
        menu_type       menu;
        sed_typesed;
        boolean nosalirflag=0;
        int ret;

        ocountry.yes_ptr="Si";
```

```
        if ((menu = menu_Open())==NULL)
                generror(6,0);
        menu_Printf(menu, "" Desea salir de ALCANCIAS ?      @fd|###|\n",&nosalirflag,
&yesno_funcs,"Presione barra espaciadora");
        menu_Flush(menu);
        if ((sed = sed_Open(menu))==NULL)
                generror(7,0);
        sed_SetColors(sed, 0x5f, 0x5f, 0x3f);
        sed_SetBorder(sed, bd_prompt);
        sed_SetPosition(sed, 8, 16);
        sed_SetHeight(sed, 1);
        sed_SetWidth(sed, 44);
        sed_SetShadow(sed,1);
        sed_SetSpecial(sed,myspc_Esc);
        sed_Repaint(sed);
        ret=sed_Go(sed);
        sed_Close(sed);
        if (ret==0)
                return nosalirflag=1;
        if(!nosalirflag)
                return nosalirflag=1;
        else
                return nosalirflag=0;

}

void initsedCon(int opcion)  // COLOCA DIFERENTES VENTANAS DE MENSAJES DE ERROR
{
        sed_type sedCon;
        menu_type menuCon;

        char ch[2];          // String para esperar que el usuario desee continuar
        ch[0]='\0';          // Inicializa la respuesta en valor nulo

        if ((menuCon = menu_Open())==NULL)
                generror(6,0);
        switch (opcion) {    // Dependiendo de opci¢n se despliega el mensaje apropiado
                case 1:menu_Printf(menuCon,"Tel,fono ya registrado,        \nPresione una tecla para
continuar @f[#]\n",ch,&string_funcs);break;
                case 2:menu_Printf(menuCon,"Tel,fono a suprimir no existe,  \nPresione una tecla para
continuar @f[#]\n",ch,&string_funcs);break;
                case 3:menu_Printf(menuCon,"Tel,fono a consultar no existe,  \nPresione una tecla para
continuar @f[#]\n",ch,&string_funcs);break;
                case 4:menu_Printf(menuCon,"Alta no efectuada,             \nPresione una tecla para
continuar @f[#]\n",ch,&string_funcs);break;
                case 5:menu_Printf(menuCon,"Baja no efectuada,             \nPresione una tecla para
continuar @f[#]\n",ch,&string_funcs);break;
                case 6:menu_Printf(menuCon,"Cambio no efectuado,           \nPresione una tecla para
continuar @f[#]\n",ch,&string_funcs);break;
                case 7:menu_Printf(menuCon,"No exite serie de ese t,lefono  \nPresione una tecla para
continuar @f[#]\n",ch,&string_funcs);break;
                case 8:menu_Printf(menuCon,"Error propio del sistema       \nPresione una tecla para
continuar @f[#]\n",ch,&string_funcs);break;
```

```
                case 9:menu_Printf(menuCon,"Referencia no registrada op.abort\nPresione una tecla para
continuar @f[#]\n",ch,&string_funcs);break;
                case 10:menu_Printf(menuCon,"No se dio de alta el nuevo psswrd\nPresione una tecla para
continuar @f[#]\n",ch,&string_funcs);break;
                case 11:menu_Printf(menuCon,",\nPresione una tecla para continuar
@f[#]\n",ch,&string_funcs);break;
                case 12:menu_Printf(menuCon,"No se esta suspendiendo tel,fono!\nPresione una tecla para
continuar @f[#]\n",ch,&string_funcs);break;
                case 13:menu_Printf(menuCon,"No se esta reanudando el tel,fono!\nPresione una tecla
para continuar @f[#]\n",ch,&string_funcs);break;
        }
        menu_Flush(menuCon);

        if ((sedCon = sed_Open(menuCon))==NULL) // Abre la ventana para poder ser usada
                generror(7,0);
        sed_SetColors(sedCon,0x43,0x47,0x41); // Seleccion de colores de la ventana
        sed_SetBorder(sedCon,bd_box);       // Seleccion del tipo de borde de la ventana
        sed_SetPosition(sedCon,10,35);      // Posicion superior izquierda de la ventana
        sed_SetShadow(sedCon,1);            // Activa el efecto de sombra
        sed_Repaint(sedCon);                // Pinta la ventana en la pantalla
        sed_Go(sedCon);                     // Manda el control del cursor a la ventana
        sed_Close(sedCon);                  // Cierra la ventana
};

int iBITACORA(char bitacora[11])
{
        menu_type       menu;
        sed_typesed;
        int             ret;

        limpiachar(bitacora,11);
        if((menu = menu_Open())==NULL)
                generror(6,0);
        menu_Printf(menu, "DOC. REFERENCIA: @f[@[10,#]]",bitacora,&string_funcs);
        menu_Flush(menu);
        if((sed = sed_Open(menu))==NULL)
                generror(7,0);
        sed_SetColors(sed, 0x1b, 0x31, 0x70);
        sed_SetBorder(sed, bd_prompt);
        sed_SetBorderTitle(sed, " AUTORIZACION DE MOVIMIENTOS ");
        sed_SetPosition(sed, 8,22);
        sed_SetHeight(sed, 1);
        sed_SetWidth(sed, 30);
        sed_SetShadow(sed, 1);
        sed_SetSpecial(sed,myspc_Esc);
        sed_Repaint(sed);
        ret = sed_Go(sed);
        sed_Close(sed);
        return(ret);
}

void movhistmov(int tipomov,char referencia[11]) {
        struct tm datenow;
        char linea[255];
```

```
        char buffer[120];

        tm_Now(&datenow);

        switch (tipomov) {
                case 1:sprintf(linea,"Altas    ");break;
                case 2:sprintf(linea,"Bajas    ");break;
                case 3:sprintf(linea,"Cambios  ");break;
                case 4:sprintf(linea,"Suspende ");break;
                case 5:sprintf(linea,"Reanuda  ");break;
                case 6:sprintf(linea,"Cat.Camb ");break;
                case 7:sprintf(linea,"CambPass ");break;
        }
        sprintf(buffer,"REF:%10s ", referencia);
        strcat(linea,buffer);
        sprintf(buffer,"%s %02d:%02d %02d/%02d/%4d ",data_buf1.sNum_Tel,
        datenow.tm_hour,datenow.tm_min,datenow.tm_mday,datenow.tm_mon+1,datenow.tm_year);
        strcat(linea,buffer);
        codifica(linea,strlen(linea));
        fwrite(strout,sizeof(strout),1,fhistmov);

        if (tipomov!=7) {
                sprintf(linea,"CBA-%-2d TLI-%-2d BIC-%-2d SSI-%-2d OBA-%-2d TCL-%-2d ICS-%-
2d BOC-%-2d TBO-%-2d",

        data_buf1.iCNCBA,data_buf1.iCNTLI,data_buf1.iCNBIC,data_buf1.iCNSSI,data_buf1.iCNOBA,
                data_buf1.iCNTCL,data_buf1.iCNICS,data_buf1.iCNBOC,data_buf1.iCNTBO);
                codifica(linea,strlen(linea));
                fwrite(strout,sizeof(strout),1,fhistmov);
                sprintf(linea,"BADP-%-2d CBOX-%-2d ORPT-%-2d SSIG-%-2d TRSTR-%-2d
LNCHAR-%-2d HM-%-2d",
                data_buf1.iCNBADP,data_buf1.iCNCBOX,data_buf1.iCNORPT,data_buf1.iCNSSIG,
                data_buf1.iCNTRSTR,data_buf1.iCNLNCHAR,data_buf1.iCNHM);
                sprintf(linea,"INTCP-%-2d PAYPHONE-%-2d OCB-%-2d SUBSIG-%-2d ICB-%-2d
LINCHAR-%-2d",
                data_buf1.iCNBADP,data_buf1.iCNCBOX,data_buf1.iCNORPT,data_buf1.iCNSSIG,
                data_buf1.iCNTRSTR,data_buf1.iCNLNCHAR);
                codifica(linea,strlen(linea));
                fwrite(strout,sizeof(strout),1,fhistmov);

                sprintf(linea,"-------------------------------------------------------------------------------");
                codifica(linea,strlen(linea));
                fwrite(strout,sizeof(strout),1,fhistmov);
        }
}

void initmp(int modopant) {
        switch (modopant) {
                case 0:  disp_Init(def_ModeCurrent,NULL);break;
                case 1:  disp_Init(def_ModeText,NULL);break;
                case 2:  disp_Init(pc_Mode2,NULL);break;
                case 3:  disp_Init(pc_Mode3,NULL);break;
                case 4:  disp_Init(pc_Mode4,NULL);break;
                case 5:  disp_Init(pc_Mode5,NULL);break;
```

```
case 6:  disp_Init(pc_Mode6,NULL);break;
case 7:  disp_Init(pc_Mode7,NULL);break;
case 13: disp_Init(pc_ModeD,NULL);break;
case 14: disp_Init(pc_ModeE,NULL);break;
case 15: disp_Init(pc_ModeF,NULL);break;
case 16: disp_Init(pc_Mode10,NULL);break;
case 17: disp_Init(pc_Mode11,NULL);break;
case 18: disp_Init(pc_Mode12,NULL);break;
case 19: disp_Init(pc_Mode13,NULL);break;
default: disp_Init(def_ModeText,NULL);
```

```
// ********************************************************************************
// SISTEMA: ALCANCIAS 3.4 (AXE, SISTEMA 12)
//
// ARCHIVO: ALCANX.C
//
// FECHA:  06/10/96
// ********************************************************************************
#include <stdio.h>
#include <process.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <mem.h>
#include <time.h>
#include <cscape.h>
#include <cstime.h>
#include <teddecl.h>

#include "bases.h"
#include "nomencla.h"
#include "comm.h"
#include "gralfunc.h"
#include "gralfunx.h"
#include "datosx.h"
#include "alcan.h"

#ifdef __TURBOC__
  extern unsigned _stklen = 10000 ;
#endif

boolean myspc_Esc(sed_type sed,int scancode);

// char horaact[4];
// char minact[4];

struct tm date;
int ult_tecla;
int fin_proc;
char strio[255];
char tit[S_TIT];
char strout[255];
ALCANBTRV data_buf1;
FILE *fhistmov;

int puerto;
int  flagopenport[N_PUERTOS]={0,0,0,0};
char telmodem[S_MODEM+1];
char telmodemant[N_PUERTOS][S_MODEM+1]={"-1","-1","-1","-1"};
char puertocfgant[N_PUERTOS][S_FILE]={"-1","-1","-1","-1"};
char puertocfg[S_FILE];
int resultadoant;


// ************* Variables usadas en ALCAN.C **********************
```

```
char passwordnew[S_PSW];
char passwordacc[S_PSW];
char passwordoper[S_PSW];
char passwordent[S_PSW];
char passwordcon[S_PSW];
sed_type sENCAB1,sMEDIO1,sPIE1;

char string_to_send[S_COMANDO];
int sistema;
FILE *fdebug;
char z_resp[S_RESP];
int flagdebug;


/* ***************** PANTALLAS DE PRESENTACION ****************** */

int iMENUPRIN() {
        menu_type       menu;
        sed_type sed;
 int            ret;

        if ((menu = menu_Open())==NULL)
                generror(6,0);
        menu_Printf(menu, "@p[1,2]@f[1) PROCESOS ]", NULL, &menu_funcs);
        menu_Printf(menu, "@p[2,2[@f[2) BASE DE DATOS [",  NULL, &menu_funcs);
        menu_Printf(menu, "@p[3,2[@f[3) REPORTES [",NULL, &menu_funcs);
        menu_Printf(menu, "@p[4,2[@f[4) OPCIONES [",NULL, &menu_funcs);
        menu_Printf(menu, "@p[5,2[@f[5) SALIR [",  NULL, &menu_funcs);
        menu_Flush(menu);
        if ((sed = sed_Open(menu))==NULL)
                generror(7,0);
        sed_SetColors(sed, 0x4b, 0x4b, 0x2b);
        sed_SetBorder(sed, bd_prompt);
        sed_SetBorderTitle(sed, "MENU PRINCIPAL");
        sed_SetPosition(sed, 8, 29);
        sed_SetHeight(sed, 7);
        sed_SetWidth(sed, 22);
        sed_SetShadow(sed, 1);
        sed_Repaint(sed);
        ret = sed_Go(sed);
        sed_Close(sed);
        return(ret);
}

void limpiavarpuertos() {

        int i;

        puerto=1; // Algo anda raro

        for (i=0;i<N_PUERTOS;i++) {
                flagopenport[i]=0;
                sprintf(telmodemant[i],"-1");
                sprintf(puertocfgant[i],"-1");
```

```
        }
        escpuerto();
}

int main(int argc,char *argv[]) {
        int opcion;
        int modopant;
        int intentos;
        char arg_modpant[5];


        leepassw();
        modopant=1;
        if(argc == 2) {
                sscanf(argv[1],"%d",&modopant);
                initmp(modopant); // Inicializa modo de pantalla
                limpiavarpuertos();
        } else  if(argc == 3) {
                sscanf(argv[1],"%d",&modopant);
                initmp(modopant); // Inicializa modo de pantalla
                sscanf(argv[2],"%s",&passwordent);
                if((strcmp(passwordacc,passwordent)==0)||(strcmp(passwordoper,passwordent)==0))
                        goto LegalUser;
        } else {
                initmp(modopant); // Inicializa modo de pantalla
                limpiavarpuertos();
        }

        intentos=0;
        while (TRUE) {
                iPASSENT(1);
                if(!((strcmp(passwordacc,passwordent)==0)||(strcmp(passwordoper,passwordent)==0))) {
                        intentos++;
                        if (intentos>=3) {
                                disp_Close();
                                return 0;
                        }
                } else
                                break;
        }
        goto LegalUser;

LegalUser:

        leetit();
        iENCAB1();
        iMEDIO1();
        iPIE1();

        do {
                        opcion=iMENUPRIN();
                        switch (opcion) {
                        case  1:sprintf(arg_modpant,"%d",modopant);
                                                fcloseall();
```

```
        execle("MENUP.EXE","MENUP.EXE",arg_modpant,passwordent,NULL,NULL);
                                                        break;
                        case  2:sprintf(arg_modpant,"%d",modopant);
                                                        fcloseall();

        execle("MENUB.EXE","MENUB.EXE",arg_modpant,passwordent,NULL,NULL);
                                                        break;
                        case  3:sprintf(arg_modpant,"%d",modopant);
                                                        fcloseall();

        execle("MENUR.EXE","MENUR.EXE",arg_modpant,passwordent,NULL,NULL);
                                                        break;
                        case  4:sprintf(arg_modpant,"%d",modopant);
                                                        fcloseall();

        execle("MENUC.EXE","MENUC.EXE",arg_modpant,passwordent,NULL,NULL);
                                                        break;
                        case  0:;
                                                                // Mod Julio
                        case  5: opcion= iCONFIRMA_SALIR();
                                                                break;
                        }
        } while (!(opcion==0));

llegalUser:

        sed_Close(sENCAB1);
        sed_Close(sMEDIO1);
        sed_Close(sPIE1);
        disp_Close();

        leepuerto();    // Lee informacion de los puertos que est,n abiertos
        openpuertos();  // Abre   los puertos que se encuentren abiertos
        cuelgapuertos(); // Cierra cada puertos que se encuentren abiertos
        limpiavarpuertos();
        return 0;
}
```

```
// ********************************************************************************
// SISTEMA: ALCANCIAS 3.4 (AXE, SISTEMA 12)
//
// ARCHIVO: IO.C
//. DESCRIPCION: FUNCIONES PARA LA COMUNICACION PUERTO SERIAL
// FECHA:  10/06/96
// ********************************************************************************
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <dos.h>
#include <bios.h>
#include <ctype.h>
#include <time.h>
#include <cscape.h>
#include <cstime.h>
#ifndef __TURBOC__
#include <graph.h>
#include <direct.h>
#else
#include <dir.h>
#endif
#include "bases.h"
#include "comm.h"
#include "nomencla.h"
#include "serial.h"
#include "debug.h"
#include "gralfunx.h"
#include "gralfunc.h"
#include "datos.h"
// ---------------------------------------------------------------------------
// Variables locales
// ---------------------------------------------------------------------------
CONFIG config;        // variable donde se guarda velocidad y paridad del puerto
int ls_res = FALSE;
int badport;          // bandera que se activa cuando no hay comunicación
int contbadport = 0;
int abortallant;
int flagtoscreen;     // Bandera que despliega las pruebas en pantalla
// ---------------------------------------------------------------------------
// Variables que se hacen referencia en otros lados
// ---------------------------------------------------------------------------
extern int  resultadoant;          // Resultado anterior de la prueba
extern int  flagopenport[N_PUERTOS]; // Mapa de puertos abiertos
extern int  flagdebug;             // Bandera de depuración
extern int  puerto;                // Puerto seleccionado
extern int  fin_proc;              // bandera que se activa para abortar procesos
extern int  sistema;               // tipo de central
extern char telmodem[S_MODEM+1];   // telefono de conexión actual
extern char telmodemant[N_PUERTOS][S_MODEM+1]; // Guarda Telefonos de conexión/puerto
extern char puertocfgant[N_PUERTOS][S_FILE];  // Guarda el mapa de archivo-conexión
extern char puertocfg[S_FILE];       // Guarda el actual puerto de conexión
extern char string_to_send[S_COMANDO];// Buffer del comando a enviar a la central
```

```
extern char z_resp[S_RESP];          // Buffer Respuesta de la central
extern FILE *fdebug;
// ---------------------------------------------------------------------------
// Función para realizar un marcaje al modem que esta conectado a la central
// ---------------------------------------------------------------------------
int inittimmer();
int expiratimmer();

// ---------------------------------------------------------------------------
// Función para realizar un marcaje al modem que esta conectado a la central
// ---------------------------------------------------------------------------
int marca_modem() {
        int resultado;

        if (flagtoscreen==1) {
                printf("\n[CONEXION ANT %s CONEXION ACT %s]\n",telmodemant[puerto-1]
.telmodem);
                printf("\n[ARCHCOMM ANT %s ARCHCOMM ACT %s]\n\n",puertocfgant[puerto-
1],puertocfg);
        }

        resultado=init();
        if (strcmp(telmodemant[puerto-1],telmodem)!=0 || strcmp(puertocfgant[puerto-1],puertocfg)!=0) {
                badport=0;
                delay(9000); // Retardo de 9 segundos unicamente en el cambio de telmodem  25/6/95
        }
        sprintf(telmodemant[puerto-1] ,"%s",telmodem);
        sprintf(puertocfgant[puerto-1],"%s",puertocfg);
        return resultado;
}

int init(void)
{
        int resultado;
        read_config();                  /* read in the config file */
        ansi_dsr = put_serial;          /* set ANSI routine */
        ansi_dsr_flag = TRUE;

        delay (500);
        if (flagopenport[puerto-1]==0) {
                close_port();
                config.port=puerto;
                open_port(config.port, 4096);          /* open the comm port */
                flagopenport[puerto-1]=1;
                if (flagtoscreen==1) {
                        printf("\n]Abre puerto %d]\n",puerto);
                        delay(500);
                }
                /* configure the port if no carrier */
                set_port(config.baud_rate, config.parity ? 8 : 7,
                config.parity ? NO_PARITY : EVEN_PARITY, 1);
                if (carrier() && (sistema==SIST12||sistema==SIST12MTM || sistema==SIST12PAM)) {
                        // hangup(); // A prueba
                        gotoxy(3,24);
```

```
                printf("Conectado a %ld baud!", get_baud());
                delay(500);
                gotoxy(3,24);
                printf("                    ");
                }
        else {
                set_port(config.baud_rate, config.parity ? 8 : 7,
                config.parity ? NO_PARITY : EVEN_PARITY, 1);
        }
        set_tx_rts(TRUE);                       /* set XON/XOFF and RTS/CTS */
        set_tx_xon(TRUE);                       /* flow control */
        set_rx_rts(TRUE);
        set_rx_xon(TRUE);
        /* reset the modem and send initialization string if no carrier */
        if (!carrier()) {
                modem_control_string("ATZ^M");
                delay(1500);
                modem_control_string(config.initialization_string);
                delay(500);
        }
    }
    if (strcmp(telmodem,"0")!=0 && (strcmp(telmodemant[puerto-1],telmodem)!=0
||strcmp(puertocfgant[puerto-1],puertocfg)!=0 )) {
        // Hay cambio de Telefono de Modem
        hangup();       // Modificación 2 Jun 96
        delay(500);     // Ojo habia 2000
        resultado=dial();
        resultadoant=resultado;
        return resultado;
    } else
        if (strcmp(telmodem,"0")==0)
                return 1;
        else
        // No es necesario volver a marcar
                return resultadoant; // Hay comunicación
}

void openpuertos() {
    for (puerto=1;puerto<=4;puerto++) {
        if (flagopenport[puerto-1]!=0) {
                if (flagtoscreen==1) {
                        printf("\n[Open Puerto[%d]=%d]",puerto,flagopenport[puerto-1]);
                        delay(500);
                }
                config.port=puerto;
                close_port();
                config.port=puerto;
                open_port(config.port, 4096);           /* open the comm port */
                flagopenport[puerto-1]=1;
                if (flagtoscreen==1) {
                        printf("\nAbre puerto= %d",puerto);
                        delay(2000);
                }
                /* configure the port if no carrier */
```

```
                        set_port(config.baud_rate, config.parity ? 8 : 7,
                        config.parity ? NO_PARITY : EVEN_PARITY, 1);
                        if (carrier() && (sistema==SIST12||sistema==SIST12MTM ||
sistema==SIST12PAM)) {
                                // hangup(); // A prueba
                                gotoxy(3,24);
                                printf("Conectado a %ld baud!", get_baud());
                                delay(500);
                                gotoxy(3,24);
                                printf("                    ");
                                }
                        else {
                                set_port(config.baud_rate, config.parity ? 8 : 7,
                                config.parity ? NO_PARITY : EVEN_PARITY, 1);
                        }
                        set_tx_rts(TRUE);                       /* set XON/XOFF and RTS/CTS */
                        set_tx_xon(TRUE);                       /* flow control */
                        set_rx_rts(TRUE);
                        set_rx_xon(TRUE);
                }
        }
}

void cuelgapuertos() {

        for (puerto=1;puerto<=4;puerto++) {
                if (flagopenport[puerto-1]!=0) {
                        config.port=puerto;
                        if (flagtoscreen==1) {
                                printf("\nCuelga Puerto[%d]=%d",puerto,flagopenport[puerto-1]);
                                delay(500);
                        }
                        sprintf(config.hangup_string, "~~~+++~~~ATH0^M");
                        if (strcmp(telmodem,"0")!=0)
                                hangup();
                        close_port();
                }
        }
}

void cierrapuertos() {
        for (puerto=1;puerto<=4;puerto++) {
                if (flagopenport[puerto-1]!=0 ) {
                        config.port=puerto;
                        if (flagtoscreen==1) {
                                printf("\nCierra Puerto[%d]=%d",puerto,flagopenport[puerto-1]);
                                delay(500);
                        }
                        close_port();
                }
        }
}

void printcontrol(int out) { // RUTINA PELIGROSA
```

```
        if (out != cr && out != lf && out != bs) {
                if isprint(out){
                        putch(out);
                }
                else {
                        printf("{%X}",out);
                }
        }
        else {
                putch(out);
        }
}

void fprintcontrol(int out) { // RUTINA PELIGROSA CUANDO ES A IMPRESORA

        if (out != cr && out != lf && out != bs) {
                if isprint(out){
                        fputc(out,fdebug);
                }
                else {
                        fprintf(fdebug,"{%X}",out);
                }
        }
        else {
                fputc(out,fdebug);
        }
}

void vacia_buffer() {

int out;
         if (sistema==SIST12MTM || sistema==SIST12PAM) {        // Vacia buffer si es MPTMOON
22/May/96
                if (flagtoscreen==1)
                        printf("\n[VACIA BUFFER]\n");
                if (flagdebug==1)
                        fprintf(fdebug,"\n[VACIA BUFFER]\n");
                while (in_ready()) {
                        out = get_serial();
                        if (flagtoscreen==1)
                                printcontrol(out);
                        if (flagdebug==1)
                                fprintcontrol(out);
                }
         }
}

int especialmtm()
{

        if (sistema==SIST12MTM || sistema==SIST12PAM) {
                if ((_fstrstr(z_resp,"EJECUCION"))!=NULL) return 1;
                if ((_fstrstr(z_resp,"CONTINUACION"))!=NULL)return 1;
```

```
                if ((_fstrstr(z_resp,"EJECUTANDOSE"))!=NULL)return 1;
                if ((_fstrstr(z_resp,"TERMINADO"))!=NULL)return 1;
                if ((_fstrstr(z_resp,"SEMANTICA"))!=NULL)return 1;
                if ((_fstrstr(z_resp,"COMPATIBILIDAD"))!=NULL)return 1;
                if ((_fstrstr(z_resp,"APLICABLE"))!=NULL)return 1;
                if ((_fstrstr(z_resp,"BUFFER OVERFLOW"))!=NULL)return 1;
        }
        return 0;
}

// ---------------------------------------------------------------------
// Función que envía lo que hay en string_to_seng y recibe respuesta en z_resp
// ---------------------------------------------------------------------
int check_rx_buf(void)
{
        int out,outant,indresp;
        int i,j;
        int done,aom;
        int temp;
        int editblanco;
        int salir;

        i=0;
        j=0;
        // Borra resultados anteriores
        limpiachar(z_resp,sizeof(z_resp));
        if(bioskey(1) != NULL )
                {
                salir=0;
                temp=bioskey(0);
                switch (temp)
                        {
                        case ESC    : salir=1;break; /* Esc */
                        case F1     : fin_proc=1;break;
                }
                if (salir==1 && sistema==SISTAOM)
                        return 1;
        }

        if (flagdebug==1) {
                fprintf(fdebug,"\n[COMANDO]\n[%s]\n",string_to_send);
        }
        if (flagtoscreen==1) {
                printf("\n[COMANDO]\n[%s]\n",string_to_send);
        }

        // ***************** Rutina AXE *************************
        if (sistema == SISTAOM) {

                // Lee respuestas anteriores
                if (flagtoscreen==1)
                        printf("\n[VACIA BUFFER]\n");
                if (flagdebug==1)
                        fprintf(fdebug,"\n[VACIA BUFFER]\n");
```

```
        while (in_ready()) {
                out = get_serial();
                if (flagtoscreen==1)
                        printcontrol(out);
                if (flagdebug==1)
                        fprintcontrol(out);
        }

        // Checha conexion con la central
        aom=FALSE;
        ls_res = FALSE;
        i=0;
        j=0;
        while (!aom) {
                if (f2_press()==1) {  // Cuidado, probar Div poniente
                        done=TRUE;
                        aom=TRUE;
                }
                f1_press_mess();
                put_serial((unsigned char) bs);
                while(!in_ready()){
                        i++;
                        if(i==0x0FFF){
                                j++;
                                i=0;
                                if(j==0xFF) {
                                        if (flagtoscreen==1)
                                                printf("\n|... NO CONTESTO ...|\n");
                                        break;
                                }
                        }
                }
                out = get_serial();
                switch (out) {
                        case ls:  aom=TRUE;
                                                                        break;
                        case bel:  if (badport==0) { // Nueva linea checar
                                                                                ctrl_break();
                                                                                //delay(100);   // A
prueba
                                                                                //ctrl_break();  // A
prueba
                                                                        }
                                                                        if (look(cotx)==0) // Si
expira por tiempo
                                                                        {
                                                                                if (flagdebug)

        fprintf(fdebug,"Error break no recibio respuesta\n");
                                                                                if (flagtoscreen)

        printf("Error break no recibio respuesta\n");
                                                                                return 1;
                                                                        }
```

```
                                                                    break;
                        default:  if (string_to_send[0]==':') { //Si es comando de confirmacion
                                                                            aom=TRUE;
                                                                            break;
                                                                    }
                                                                    put_serial((unsigned char)
sc);
                                                                    if (look(eotx)==0) // Si
expira por tiempo
                                                                    {
                                                                            if (flagdebug)

        fprintf(fdebug,"\n[Error ; no recibio respuesta]\n");
                                                                            if (flagtoscreen)

        printf("\n[Error ; no recibio respuesta]\n");
                                                                            return 1;
                                                                    }
                                                                    if(ls_res) aom=TRUE;
                                                                    break;
                        }
                }

                // Envia el comando
                for (i=0;i<strlen(string_to_send);i++) {
                        put_serial((unsigned char)string_to_send[i]);
                }

                // Recibe respuesta
                done=FALSE;
                outant=0x00;
                indresp=0;
                inittimmer();
                while (!done) {
                        if (f2_press()==1 || expiratimmer()==1)
                                done=TRUE;
                        f1_press_mess();
                        if (in_ready()) {               /* get and display incoming */
                                out = get_serial();             /* characters */
                                if (flagtoscreen==1)
                                        printcontrol(out);
                                switch (out) {
                                                case bel :break;
                                                case nll :break;
                                                case del :break;
                                                case eotx:done=TRUE;
                                                                                        break;
                                                case ls  :if (outant == cr || outant == del || outant ==
0x00){

        done=TRUE;

        break;
                                                                                        }
```

```
                                        case lf  :if (outant == cr)

        break;
                                        default:  if (out != cr) {
                                                                        //      gotoxy(75,2);
                                                                        //
if(indresp/9*9==indresp) printf("rx");
                                                                        //      else printf("  ");

        z_resp[indresp] = out;
                                                                                indresp++;
                                                                                if (indresp
>= S_RESP)

        generror(1,0);
                                                                        }
                                }
                        if (out != nll)
                        outant=out;
                }
        }
        z_resp[indresp]=NULL;
}


// *************** Rutina Sistema 12 ********************

if (sistema == SIST12 || sistema==SIST12MTM|| sistema==SIST12PAM) {

        // Envia el comando
        for (i=0;i<strlen(string_to_send);i++)
                put_serial((unsigned char)string_to_send[i]); /* send the key out the port */
        if (sistema==SIST12MTM||sistema==SIST12PAM) {
                put_serial(cr);
        }
        // Recibe respuesta
        done=FALSE;
        outant = nll;
        indresp = 0;
        inittimmer();
        while (!done) {
                if (f2_press()==1 || expiratimmer()==1) {
                        done=TRUE;
                        return 0;
                }
                f1_press_mess();
                if (in_ready()) {                       /* get and display incoming */
                        out = get_serial();
                                                        /* characters */
                        if (especialtm()==1 )
                                done=TRUE;
                        if (flagtoscreen==1)
                                printcontrol(out);
                        switch (out) {
```

```
                                        case bel :break;
                                        case nll :break;
                                        case dc1 :break;
                                        case eob :done=TRUE;
                                                                        break;
                                        case eotx:done=TRUE;
                                                                        break;
                                        case etb :done=TRUE;
                                                                        break;
                                        case ls  :if (outant == cr || outant == dc1 || outant == nll)
{
        done=TRUE;

        break;
                                                                                }
                                // case lf  :if (outant == cr)
                                //              break;
                                                default:if (out != cr) {

        z_resp[indresp] = out;
                                                                        indresp++;
                                                                        if (indresp
>= S_RESP)

        generror(1,0);
                                                                }
                                        }
                                outant=out;
                }
        }
        z_resp[indresp]=NULL;
}

// ********** Rutina IOG11 E IOG3 **********


if ( sistema == SISTIOG11 || sistema == SISTIOG3) {


        if ((string_to_send[0]=='S' && string_to_send[1[=='L') ||  //SLOMI
                        (string_to_send[0]=='C' && string_to_send[1]=='H'))   //CHSIP
                editblanco=1;
        else
                editblanco=0;

        if(string_to_send[0] != NULL) {
                if (string_to_send]0]!=';')
                        ctrl_break();
                delay(300);

                // Lee respuestas anteriores // LINEAS A PRUEBA.....
                if (flagtoscreen==1)
                        printf("\n|VACIA BUFFER|\n");
```

```
                if (flagdebug==1)
                        fprintf(fdebug,"\n[VACIA BUFFER]\n");
                while (in_ready()) {
                        out = get_serial();
                        if (flagtoscreen==1)
                                printcontrol(out);
                        if (flagdebug==1)
                                fprintcontrol(out);
                }

                for (i=0;i<strlen(string_to_send);i++) {
                        put_serial((unsigned char)string_to_send[i]); /* send the key out the port
*/
                        if (sistema==SISTIOG3)
                                delay(20); // Retardo para IOG3 Checar bien ?????????
                }

                delay(500);
                if (sistema==SISTIOG11)
                        put_serial(cr);         // caso iog11
                if (sistema==SISTIOG3)
                        put_serial(ctrl_c); // caso iog3
        }

        done=FALSE;

        outant= nll;
//      outantant= nll;
        indresp=0;
        while (!done) {
                if (f2_press()==1)
                        done=TRUE;
                f1_press_mess();
                if (in_ready()) {                       /* get and display incoming */
                        out = get_serial();             /* characters */
                        if (flagtoscreen==1)
                                printcontrol(out);
                        switch (out) {
                                        case bel :break;
                                        case nll :break;
                                        case ff  :done=TRUE;        // Comentariar si termina
con eot
                                                                        break;
                                        case eot :// if(outant==ff)  // Comentariar si termina
con ff
                                                                        //
done=TRUE;      // Comentariar si termina con ff
                                                                        break;
                                        case eotx:done=TRUE;        // Descomentariada
Juarez 22/05/96
                                                                        break;
                                        case etb :done=TRUE;
                                                                        break;
```

```
                                        case dp  :if (outant == cr || outant == dcl || outant == nll
|| outant == eotx) {
        done=TRUE;
        break;
                                                                        } else {
        z_resp[indresp] = out;
        indresp++;
        break;
                                                                        }
                                        case ls  :if (outant == cr || outant == dcl || outant == nll
|| outant == eotx) {
        done=TRUE;
        break;
                                                                        } else {
        z_resp[indresp] = out;
        indresp++;
        break;
                                                                        }
                                        case lf  :if (editblanco==1) {
                                                                                if
(outant == cr )
                                                                                //
printf("OCURRENCIA");
        break;
                                                                        }
                                        case cr  :if (editblanco==1) {
                                                                                if(outant
== sp) break;
                                                                                out = lf;
                                                                        }
                                        case sp  :if (editblanco==1) {
                                                                                if(outant
== lf || outant == cr)  {
                                                                                        out
= lf;
break;
                                                                                }
                                                                        }
                                        default  :if (editblanco==1) {
                                                                                if
(outl=cr) {
```

```
        z_resp[indresp] = out;

        indresp++;

        if (indresp >= S_RESP)

                generror(1,0);
                                                                                }
                                                                        } else {
                                                                                if
(out!=cr) {

        z_resp[indresp] = out;

        indresp++;
                                                                                }
                                                                                if
(indresp >= S_RESP)

        generror(1,0);
                                                                        }
                                        }
                                // outantant=outant;
                                outant=out;
                        }
                }
                z_resp[indresp]=NULL;
        }
        if (flagdebug==1)
                fprintf(fdebug,"\n[RESPUESTA]\n%s\n",z_resp);
        if (flagtoscreen==1)
                printf("\n[RESPUESTA]\n%s\n",z_resp);
        return 1;
}

int look(unsigned char in)
{
        unsigned int done;
        unsigned char out;

        struct tm tmahora,tminicio;

        if (badport==1)
                return 0;
        tm_Now(&tminicio);
        done = FALSE;
        ls_res = FALSE;
        while(!done){
                tm_Now(&tmahora);
                if (tm_ElapSecs(&tminicio,&tmahora)>10L) {
                        contbadport++;
                        if (contbadport>=12)
                                badport=1;
```

```
                return 0;
            }
            if (in_ready()) {                    /* get and display incoming */
                out = get_serial();
                if (flagtoscreen==1)
                    printcontrol(out);
                if(out==in) done=TRUE;
                if(out==ls) {
                    ls_res=TRUE;
                    return 1;
                }
                if(out==ms && (sistema==SIST12MTM || sistema==SIST12PAM)) {
                    ls_res=TRUE;
                    out = get_serial();
                    return 1;
                }

            }
        }
        return 1;
}

/* ctrl_break  */
void interrupt far ctrl_break(void)
{
if (config.port==1 || config.port==3) {
        outportb(0x3FB,(inportb(0x3FB) | 0x40));
        delay(500);
        outportb(0x3FB,(inportb(0x3FB) & 0xBF));
        delay(500);
        outportb(0x3FC,0x0F);
        }
else {
        outportb(0x2FB,(inportb(0x2FB) | 0x40));
        delay(500);
        outportb(0x2FB,(inportb(0x2FB) & 0xBF));
        delay(500);
        outportb(0x2FC,0x0F);
        }
}

// ---------------------------------------------------------------------
// Rutina Básica para marcar a un número telefónico
// ---------------------------------------------------------------------
int dial(void)
{
        int br;
        char line[81], dstring[81], number[81];
        int i;
        int resultado=100;

        sprintf(number,"%s",telmodem);
        sprintf(dstring, "%s%s", config.dial_prefix, number);
```

```
abortallam=0;

for (i=1;i<=3  && resultado!=1 && abortallam==0  ;i++) {  // Tres marcajes antes de abortar

        while (in_ready()) {    // Vacia el buffer del modem
                get_serial();
                //printf("{");
                //printcontrol(get_serial());
                //printf("}");
        }

        gotoxy(3,24);
        printf("LLamando %s", number);
        delay(2000); // 2 segundos
        gotoxy(57,24);
        printf("Segundos a esperar:");

        sprintf(line, "%s^M", dstring);

        modem_control_string(line);    /* dial the number */
        get_modem_string(line, dstring); /* get a response */
        /* dial interrupted by user */
        if (!strcmp(line, "INTERRUPTED")) {
                resultado=101;
        }
        /* user wants to repeat the dial immediately */
        if (!strcmp(line, "RECYCLE")) {
                resultado=102;
                continue;
        }
        if (!strcmp(line, "NO DIALTONE")) {
                resultado=103;
        }
        if (!strcmp(line, "BUSY")) {
                resultado=104;
        }
        if (!strcmp(line, "NO CARRIER")) {
                resultado=104;
        }
        /* connection was made */
        if (_fstrstr(line, "CONNECT") != NULL) {
                gotoxy(3,24);
                ansiprintf("%s       ", line);
                /* set DTE rate to match the connection rate if port */
                /*  isn't locked */
                if (!config.lock_port) {
                        br = atoi(line + 7);
                        if (br)
                                set_baud(br);
                        else
                                set_baud(300);
                }
                resultado=1;
```

```
                }
                if (resultado!=1&& abortallam==0)
                        hangup();
                gotoxy(3,24);
                printf("%s       ", line);                /* display the result string */
        }
        if (resultado==2)
                generror(2,0);
        if (resultado==3)
                generror(3,0);
        if (resultado==4)
                generror(4,0);
        if (resultado==5)
                generror(5,0);
        if (resultado==6)
                generror(9,0);

        return resultado;
}

/* get response string */
char *get_modem_string(char *s, char *d)
{
        int i, c;
        unsigned last;
        char cadena[30];

        sprintf(cadena,"DIALING : T%s",telmodem);

        last = mpeek(0, 0x46c);
        i = 1092;
        *s = 0;
        ansiprintf("%-2d", (i * 10) / 182);          /* display time remaining */
        while (TRUE) {
                if (f1_press()==1) {
                        abortallam=1;
                        break;
                }
                /* get a character from the port if one's there */
                if (in_ready()) {
                        switch (c = get_serial()) {
                                case 13:                        /* CR - return the result string */
                                        if (*s)
                                                return s;
                                        continue;
                                default:
                                        if (c != 10) {          /* add char to end of string */
                                                s[strlen(s) + 1] = 0;
                                                s[strlen(s)] = (char)c;
                                                /* ignore RINGING and the dial string */
                                                if (!strcmp(s, "RINGING") ||!strcmp(s, "RINGBACK") ||
!strcmp(s, d) || !strcmp(s,cadena))
                                                        *s = 0;
                                        }
```

```
                }
            }
            /* check for timeout and display time remaining */
            if (last != mpeek(0, 0x46c)) {
                last = mpeek(0, 0x46c);
                i--;
                ansiprintf("\b \b\b \b%-2d", (i * 10) / 182);
                if (!i) {
                    *s = 0;
                    put_serial(13);
                    return s;
                }
            }
        }
    return NULL; // Incluido en compilacion
}

// ---------------------------------------------------------------------
// Rutina Básica para colgar
// ---------------------------------------------------------------------
void hangup(void)
{
        int i;
        unsigned last;


        abortallam=0;
        gotoxy(3,24);
        printf("Colgando...      ");
        last = mpeek(0, 0x46c);         /* get current system clock */
        i = 18;  // antes 180                          /* try for about 10 seconds */
        set_dtr(FALSE);                         /* drop DTR */
        while (carrier() && i) {                /* loop till loss of carrier */
                if (f1_press_mess()==1) {
                        f1_press();
                        abortallam=1;
                        return;
                }
                if (last != mpeek(0, 0x46c)) {  /*  or time out */
                        i--;
                        last = mpeek(0, 0x46c);
                }
        }
        set_dtr(TRUE);                          /* raise DTR */
        if (!carrier()) {                       /* return if disconnect */
                purge_in();
                gotoxy(3,24);
                printf("         ");
                return;
        }
        modem_control_string(config.hangup_string); /* send software command */
        i = 18; // antes 180 alfonso                                  /* try for about 10 seconds */
        while (carrier() && i) {                /* loop till loss of carrier */
                if (f1_press_mess()==1) {
```

```
                        fl_press();
                        abortallam=1;
                        return;
                }
                if (last != mpeek(0, 0x46c)) {   /* or time out */
                        i--;
                        last = mpeek(0, 0x46c);
                }
        }
        purge_in();
        gotoxy(3,24);
        printf("          ");
//      ansiprintf("\n");
}

/* Send Control String to Modem */
void modem_control_string(char *s)
{
        while (*s) {                            /* loop for the entire string */
                switch (*s) {
                        case '~':                       /* if ~, wait a half second */
                                delay(500);
                                break;
                        default:
                                switch (*s) {
                                        case '^':               /* if ^, it's a control code */
                                                if (s[1]) {             /* send the control code */
                                                        s++;
                                                        put_serial((unsigned char)(*s - 64));
                                                }
                                                break;
                                        default:
                                                put_serial(*s); /* send the character */
                                }
                                delay(50);                      /* wait 50 ms. for slow modems */
                }
                s++;                                    /* bump the string-pointer */
        }
}

/* read program configuration file */
void read_config(void)
{
        FILE *f;
        char namefile[S_FILE];

        sprintf(namefile,"%s",puertocfg);
        /* open the file, create default if it doesn't exist */
        if ((f = fopen(namefile, "rb")) == NULL) {
                if ((f = fopen(namefile, "wb")) != NULL) {
                        config.port = 1;
                        config.baud_rate = 1200;
                        config.parity = P7E1;
                        config.lock_port = FALSE;
```

```
                    sprintf(config.initialization_string, " ");
                    sprintf(config.haugup_string, "~~~+++~~~ATH0^M");
                    sprintf(config.dial_prefix, "ATDT");
                    fwrite(&config, sizeof(CONFIG), 1, f);
                    if ( f != NULL ) fclose(f);
                    else generror(11,27);
                    return;
                }
                return;
        }
        fread(&config, sizeof(CONFIG), 1, f); /* read in config info */
        puerto=config.port; // Asignacion del nuevo puerto a utilizar
        if ( f != NULL ) fclose(f);
        else generror(11,27);

}

/* write program configuration file */
void write_config(void)
{
        FILE *f;
        char namefile[S_FILE];

        /* write the config info */

        sprintf(namefile,"%s",puertocfg);
        if ((f = fopen(namefile, "w+b")) != NULL) {
                fwrite(&config, sizeof(CONFIG), 1, f);
                if ( f != NULL ) fclose(f);
                else generror(11,27);
        }
}
```

```
// ********************************************************************************
// SISTEMA: ALCANCIAS 3.4 (AXE, SISTEMA 12)
//
// ARCHIVO: MANEJOD2.C
//
// FECHA:  06/10/96
// ********************************************************************************
#include <stdio.h>
#include <conio.h>
#include "d4all.h"
#include "debug.h"
#include "bases.h"
#include "nomencla.h"

extern CODE4 codebase;
extern DATA4 *alcanfile;
extern DATA4 *alcanfile3;
extern FIELD4 *FNum_Tel;
extern FIELD4 *FClave_Pob;
extern FIELD4 *FAlarma;
extern FIELD4 *FGrupo;
extern FIELD4 *FTipo_Apto;
extern FIELD4 *FTipoAmb;
extern FIELD4 *FLugar_Ins;
extern FIELD4 *FFecha_Ins;
extern FIELD4 *FCalleyNum;
extern FIELD4 *FColonia;
extern FIELD4 *FC_P;
extern FIELD4 *FDistrito;
extern FIELD4 *FPrincipal;
extern FIELD4 *FSecundario;
extern FIELD4 *FZona;
extern FIELD4 *FSuspendido;
extern FIELD4 *FSinControl;
extern FIELD4 *FTServicio;
extern FIELD4 *FAlarma;
extern FIELD4 *FInventario;
extern FIELD4 *FCNCBA;
extern FIELD4 *FCNTLI;
extern FIELD4 *FCNBIC;
extern FIELD4 *FCNSSI;
extern FIELD4 *FCNOBA;
extern FIELD4 *FCNTCL;
extern FIELD4 *FCNICS;
extern FIELD4 *FCNBOC;
extern FIELD4 *FCNTBO;
extern FIELD4 *FCNBADP;
extern FIELD4 *FCNORPT;
extern FIELD4 *FCNLNCHAR;
extern FIELD4 *FCNTRSTR;
extern FIELD4 *FCNSSIG;
extern FIELD4 *FCNCBOX;
extern FIELD4 *FCNHM;
extern FIELD4 *FCACBA;
```

```
extern FIELD4 *FCATLI;
extern FIELD4 *FCABIC;
extern FIELD4 *FCASSI;
extern FIELD4 *FCAOBA;
extern FIELD4 *FCATCL;
extern FIELD4 *FCAICS;
extern FIELD4 *FCABOC;
extern FIELD4 *FCATBO;
extern FIELD4 *FCABADP;
extern FIELD4 *FCAORPT;
extern FIELD4 *FCALNCHAR;
extern FIELD4 *FCATRSTR;
extern FIELD4 *FCASSIG;
extern FIELD4 *FCACBOX;
extern FIELD4 *FCAHM;
extern FIELD4 *FSistTel;
extern FIELD4 *FStatus;
extern FIELD4 *FStatDesc;
extern FIELD4 *FSinMov;
extern FIELD4 *FCatNok;
extern FIELD4 *FSuspendido;
extern FIELD4 *FSincontrol;
extern FIELD4 *FContadorU;
extern FIELD4 *FContadorP;
extern FIELD4 *FHoraCatU;
extern FIELD4 *FHoraCatP;
extern FIELD4 *FFechaCatU;
extern FIELD4 *FFechaCatP;
extern FIELD4 *FHoraContU;
extern FIELD4 *FHoraContP;
extern FIELD4 *FFechaContU;
extern FIELD4 *FFechaContP;
extern FIELD4 *FSeguridad;
extern FIELD4 *FTelHis;
extern FIELD4 *FHoraHis;
extern FIELD4 *FFechaHis;
extern FIELD4 *FContHis;
extern TAG4     *Tele_Tag,*Zona_Tag,*Tapar_Tag,*Posf_Tag;
extern TAG4 *Tele_Tag3;
extern ALCANBTRV data_buf1;
extern CONTBTRV data_buf3;
extern int sistema;
extern long telprobar;
extern char msg[S_MSG];
void copydbmem(int nodb) {
        switch (nodb) {
                case 1:
                        f4ncpy(FNum_Tel,data_buf1.sNum_Tel,sizeof(data_buf1.sNum_Tel));
                        f4ncpy(FClave_Pob,data_buf1.sClave_Pob,sizeof(data_buf1.sClave_Pob));
                        f4ncpy(FAlarma,data_buf1.sAlarma,sizeof(data_buf1.sAlarma));
                        f4ncpy(FGrupo,data_buf1.sGrupo,sizeof(data_buf1.sGrupo));
                        f4ncpy(FTipo_Apto,data_buf1.sTipo_Apto,sizeof(data_buf1.sTipo_Apto));
                        f4ncpy(FTipoAmb,data_buf1.sTipoAmb,sizeof(data_buf1.sTipoAmb));
                        f4ncpy(FLugar_Ins,data_buf1.sLugar_Ins,sizeof(data_buf1.sLugar_Ins));
```

```
f4ncpy(FFecha_Ins,data_buf1.sFecha_Ins,sizeof(data_buf1.sFecha_Ins));
f4ncpy(FCalleyNum,data_buf1.sCalleyNum,sizeof(data_buf1.sCalleyNum));
f4ncpy(FColonia,data_buf1.sColonia,sizeof(data_buf1.sColonia));
f4ncpy(FC_P,data_buf1.sC_P,sizeof(data_buf1.sC_P));
f4ncpy(FDistrito,data_buf1.sDistrito,sizeof(data_buf1.sDistrito));
f4ncpy(FPrincipal,data_buf1.sPrincipal,sizeof(data_buf1.sPrincipal));
f4ncpy(FSecundario,data_buf1.sSecundario,sizeof(data_buf1.sSecundario));

f4ncpy(FZona,data_buf1.sZona,sizeof(data_buf1.sZona));
data_buf1.iSuspendido=f4int(FSuspendido);
data_buf1.iSinControl=f4int(FSinControl);
data_buf1.iTServicio =f4int(FTServicio);

data_buf1.iCNCBA=f4int(FCNCBA);
data_buf1.iCNTLI=f4int(FCNTLI);
data_buf1.iCNBIC=f4int(FCNBIC);
data_buf1.iCNSSI=f4int(FCNSSI);
data_buf1.iCNOBA=f4int(FCNOBA);
data_buf1.iCNTCL=f4int(FCNTCL);
data_buf1.iCNICS=f4int(FCNICS);
data_buf1.iCNBOC=f4int(FCNBOC);
data_buf1.iCNTBO=f4int(FCNTBO);
data_buf1.iCNBADP=f4int(FCNBADP);
data_buf1.iCNCBOX=f4int(FCNCBOX);
data_buf1.iCNORPT=f4int(FCNORPT);
data_buf1.iCNSSIG=f4int(FCNSSIG);
data_buf1.iCNTRSTR=f4int(FCNTRSTR);
data_buf1.iCNLNCHAR=f4int(FCNLNCHAR);
data_buf1.iCNHM=f4int(FCNHM);
data_buf1.iCACBA=f4int(FCACBA);
data_buf1.iCATLI=f4int(FCATLI);
data_buf1.iCABIC=f4int(FCABIC);
data_buf1.iCASSI=f4int(FCASSI);
data_buf1.iCAOBA=f4int(FCAOBA);
data_buf1.iCATCL=f4int(FCATCL);
data_buf1.iCAICS=f4int(FCAICS);
data_buf1.iCABOC=f4int(FCABOC);
data_buf1.iCATBO=f4int(FCATBO);
data_buf1.iCABADP=f4int(FCABADP);
data_buf1.iCACBOX=f4int(FCACBOX);
data_buf1.iCAORPT=f4int(FCAORPT);
data_buf1.iCASSIG=f4int(FCASSIG);
data_buf1.iCATRSTR=f4int(FCATRSTR);
data_buf1.iCALNCHAR=f4int(FCALNCHAR);
data_buf1.iCAHM=f4int(FCAHM);
data_buf1.iSistTel=f4int(FSistTel);
data_buf1.iStatus=f4int(FStatus);
f4ncpy(FStatDesc,data_buf1.sStatDesc,sizeof(data_buf1.sStatDesc)-1);
data_buf1.iSinMov=f4int(FSinMov);
data_buf1.iCatNok=f4int(FCatNok);
data_buf1.iSuspendido=f4int(FSuspendido);
data_buf1.iSinControl=f4int(FSinControl);
data_buf1.lContadorU=f4long(FContadorU);
data_buf1.lContadorP=f4long(FContadorP);
```

```
                f4ncpy(FHoraCatU,data_buf1.sHoraCatU,sizeof(data_buf1.sHoraCatU));
                f4ncpy(FHoraCatP,data_buf1.sHoraCatP,sizeof(data_buf1.sHoraCatP));
                f4ncpy(FHoraContU,data_buf1.sHoraContU,sizeof(data_buf1.sHoraContU));
                f4ncpy(FHoraContP,data_buf1.sHoraContP,sizeof(data_buf1.sHoraContP));
                f4ncpy(FFechaCatU,data_buf1.sFechaCatU,sizeof(data_buf1.sFechaCatU));
                f4ncpy(FFechaCatP,data_buf1.sFechaCatP,sizeof(data_buf1.sFechaCatP));
                f4ncpy(FFechaContU,data_buf1.sFechaContU,sizeof(data_buf1.sFechaContU));
                f4ncpy(FFechaContP,data_buf1.sFechaContP,sizeof(data_buf1.sFechaContP));
                break;
        case 3:
                f4ncpy(FTelHis,data_buf3.sTelHis,sizeof(data_buf3.sTelHis));
                f4ncpy(FHoraHis,data_buf3.sHoraHis,sizeof(data_buf3.sHoraHis));
                f4ncpy(FFechaHis,data_buf3.sFechaHis,sizeof(data_buf3.sFechaHis));
                data_buf3.lContHis=f4long(FContHis);
    }
}

void copymemdb(int nodb)
{
    switch (nodb) {
        case 1:
                sscanf(data_buf1.sNum_Tel,"%ld",&telprobar);
                data_buf1.iSistTel=sistema;
                sprintf(data_buf1.sStatDesc,"%s",msg);

                f4assign(FNum_Tel,data_buf1.sNum_Tel);
                f4assign(FClave_Pob,data_buf1.sClave_Pob);
                f4assign(FAlarma,data_buf1.sAlarma);
                f4assign(FGrupo,data_buf1.sGrupo);
                f4assign(FTipo_Apto,data_buf1.sTipo_Apto);
                f4assign(FTipoAmb,data_buf1.sTipoAmb);
                f4assign(FLugar_Ins,data_buf1.sLugar_Ins);
                f4assign(FFecha_Ins,data_buf1.sFecha_Ins);
                f4assign(FCalleyNum,data_buf1.sCalleyNum);
                f4assign(FColonia,data_buf1.sColonia);
                f4assign(FC_P,data_buf1.sC_P);
                f4assign(FDistrito,data_buf1.sDistrito);
                f4assign(FPrincipal,data_buf1.sPrincipal);
                f4assign(FSecundario,data_buf1.sSecundario);

                f4assign(FZona,data_buf1.sZona);
                f4assign_int(FSuspendido,data_buf1.iSuspendido);
                f4assign_int(FSinControl,data_buf1.iSinControl);
                f4assign_int(FTServicio,data_buf1.iTServicio);

                if (sistema==SISTAOM || sistema==SISTIOG11 || sistema==SISTIOG3) {
                        f4assign_int(FCNCBA,data_buf1.iCNCBA);
                        f4assign_int(FCNTLI,data_buf1.iCNTLI);
                        f4assign_int(FCNBIC,data_buf1.iCNBIC);
                        f4assign_int(FCNSSI,data_buf1.iCNSSI);
                        f4assign_int(FCNOBA,data_buf1.iCNOBA);
                        f4assign_int(FCNTCL,data_buf1.iCNTCL);
                        f4assign_int(FCNICS,data_buf1.iCNICS);
                        f4assign_int(FCNBOC,data_buf1.iCNBOC);
```

```
        fAssign_int(FCNTBO,data_buf1.iCNTBO);
        fAssign_int(FCACBA,data_buf1.iCACBA);
        fAssign_int(FCATLI,data_buf1.iCATLI);
        fAssign_int(FCABIC,data_buf1.iCABIC);
        fAssign_int(FCASSI,data_buf1.iCASSI);
        fAssign_int(FCAOBA,data_buf1.iCAOBA);
        fAssign_int(FCATCL,data_buf1.iCATCL);
        fAssign_int(FCAICS,data_buf1.iCAICS);
        fAssign_int(FCABOC,data_buf1.iCABOC);
        fAssign_int(FCATBO,data_buf1.iCATBO);
        fAssign_int(FCNBADP,0);
        fAssign_int(FCNCBOX,0);
        fAssign_int(FCNORPT,0);
        fAssign_int(FCNSSIG,0);
        fAssign_int(FCNTRSTR,0);
        fAssign_int(FCNLNCHAR,0);
        fAssign_int(FCNHM,0);
        fAssign_int(FCABADP,0);
        fAssign_int(FCACBOX,0);
        fAssign_int(FCAORPT,0);
        fAssign_int(FCASSIG,0);
        fAssign_int(FCATRSTR,0);
        fAssign_int(FCALNCHAR,0);
        fAssign_int(FCAHM,0);
}
if (sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM) {
        fAssign_int(FCNCBA,0);
        fAssign_int(FCNTLI,0);
        fAssign_int(FCNBIC,0);
        fAssign_int(FCNSSI,0);
        fAssign_int(FCNOBA,0);
        fAssign_int(FCNTCL,0);
        fAssign_int(FCNICS,0);
        fAssign_int(FCNBOC,0);
        fAssign_int(FCNTBO,0);
        fAssign_int(FCACBA,0);
        fAssign_int(FCATLI,0);
        fAssign_int(FCABIC,0);
        fAssign_int(FCASSI,0);
        fAssign_int(FCAOBA,0);
        fAssign_int(FCATCL,0);
        fAssign_int(FCAICS,0);
        fAssign_int(FCABOC,0);
        fAssign_int(FCATBO,0);
        fAssign_int(FCNBADP,data_buf1.iCNBADP);
        fAssign_int(FCNCBOX,data_buf1.iCNCBOX);
        fAssign_int(FCNORPT,data_buf1.iCNORPT);
        fAssign_int(FCNSSIG,data_buf1.iCNSSIG);
        fAssign_int(FCNTRSTR,data_buf1.iCNTRSTR);
        fAssign_int(FCNLNCHAR,data_buf1.iCNLNCHAR);
        fAssign_int(FCNHM,data_buf1.iCNHM);
        fAssign_int(FCABADP,data_buf1.iCABADP);
        fAssign_int(FCACBOX,data_buf1.iCACBOX);
        fAssign_int(FCAORPT,data_buf1.iCAORPT);
```

```
                        f4assign_int(FCASSIG,data_buf1.iCASSIG);
                        f4assign_int(FCATRSTR,data_buf1.iCATRSTR);
                        f4assign_int(FCALNCHAR,data_buf1.iCALNCHAR);
                        f4assign_int(FCAHM,data_buf1.iCAHM);
                }
                f4assign_int(FSistTel,data_buf1.iSistTel);
                f4assign_int(FStatus,data_buf1.iStatus);
                f4assign(FStatDesc,data_buf1.sStatDesc);
                f4assign_int(FSinMov,data_buf1.iSinMov);
                f4assign_int(FCatNok,data_buf1.iCatNok);
                f4assign_long(FContadorU,data_buf1.lContadorU);
                f4assign_long(FContadorP,data_buf1.lContadorP);
                f4assign(FHoraCatU,data_buf1.sHoraCatU);
                f4assign(FHoraCatP,data_buf1.sHoraCatP);
                f4assign(FHoraContU,data_buf1.sHoraContU);
                f4assign(FHoraContP,data_buf1.sHoraContP);
                f4assign(FFechaCatU,data_buf1.sFechaCatU);
                f4assign(FFechaCatP,data_buf1.sFechaCatP);
                f4assign(FFechaContU,data_buf1.sFechaContU);
                f4assign(FFechaContP,data_buf1.sFechaContP);
                break;
        case 3:
                f4assign(FTelHis,data_buf3.sTelHis);
                f4assign(FHoraHis,data_buf3.sHoraHis);
                f4assign(FFechaHis,data_buf3.sFechaHis);
                f4assign_long(FContHis,data_buf3.lContHis);
                break;
    }
}

void insertdb(int nodb)
{
    switch (nodb) {
        case 1:
                d4append_start(alcanfile,0);
                d4blank(alcanfile);
                copymemdb(nodb);
                d4append(alcanfile);
                break;
        case 3:
                d4append_start(alcanfile3,0);
                d4blank(alcanfile3);
                copymemdb(nodb);
                d4append(alcanfile3);
                break;
    }
}

void deletedb(int nodb) {

    switch (nodb) {
        case 1:
                copymemdb(nodb);
                d4seek(alcanfile,data_buf1.sNum_Tel);
```

```
                        d4delete(alcanfile);
                        break;
                case 3:
                        copymemdb(nodb);
                        d4seek(alcanfile3,data_buf3.sTelHis);
                        d4delete(alcanfile3);
                        break;
        }
}

int updatedb(int nodb) {
        int rc;

        switch (nodb) {
                case 1:
                        copymemdb(nodb);
                        rc = d4seek(alcanfile,data_buf1.sNum_Tel);
                        switch (rc)     {
                                case r4success:break;
                                case r4after: printf("Error en Update rafter");break;
                                case r4eof:   printf("Error en Update eof");break;
                                case r4entry: printf("Error en Update entry");break;
                                case r4locked:printf("Error en Update locked");break;

                        }
                        break;
                case 3:
                        copymemdb(nodb);
                        rc = d4seek(alcanfile3,data_buf3.sTelHis);
                        switch (rc)     {
                                case r4success:break;
                                case r4after: printf("Error en Update rafter");break;
                                case r4eof:   printf("Error en Update eof");break;
                                case r4entry: printf("Error en Update entry");break;
                                case r4locked:printf("Error en Update locked");break;
                        }
                        break;
        }
        return rc;
}

int getequdb(int nodb) {
        int rc;

        switch (nodb) {
                case 1:rc = d4seek(alcanfile,data_buf1.sNum_Tel);break;
                case 3:rc = d4seek(alcanfile3,data_buf3.sTelHis);break;
        }
        if (rc==r4success) {
                copydbmem(nodb);
                return 0;
        }       else
                return 1;
}
```

```
int seekdb(int nodb) {
        int rc;

        switch (nodb) {
                case 1:rc = d4seek(alcanfile,data_buf1.sNum_Tel);break;
                case 3:rc = d4seek(alcanfile3,data_buf3.sTelHis);break;
        }
        if (rc==r4success) {
                return 0;
                }
        else
                return 1;
}

int getfirstdb(int nodb) {
        switch(nodb) {
                case 1:d4top(alcanfile);break;
                case 3:d4top(alcanfile3);break;
        }
        copydbmem(nodb);
        return 0;
}

int getlastdb(int nodb) {
        switch (nodb) {
                case 1:d4bottom(alcanfile);break;
                case 3:d4bottom(alcanfile3);break;
        }
        copydbmem(nodb);
        return 0;
}

int getnextdb(int nodb) {
        int rc;
        switch (nodb) {
                case 1:
                        rc = d4skip(alcanfile,1L);
                        if (rc != r4eof)
                                copydbmem(nodb);
                        break;
                case 3:
                        rc = d4skip(alcanfile3,1L);
                        if (rc != r4eof)
                                copydbmem(nodb);
        }
        return 0;
}

int getprevdb(int nodb) {
        switch (nodb) {
                case 1:
                        d4skip(alcanfile,-1L);
                        copydbmem(nodb);
```

```
                        break;
                case 3:
                        d4skip(alcanfile3,-1L);
                        copydbmem(nodb);
                        break;
        }
        return 0;
}

int dtop(int nodb)
{
        int rc;
        switch (nodb) {
                case 1:  rc =d4top(alcanfile);break;
                case 3:  rc =d4top(alcanfile3);break;
        }
        return rc;
}

int dseek(int nodb) {
        int rc;

        switch (nodb) {
                case 1:  rc = d4seek(alcanfile,data_buf1.sNum_Tel);break;
                case 3:  rc = d4seek(alcanfile3,data_buf3.sTelHis);break;
        }
        if (rc==r4success)
                return 0;
        else
                return 1;
}

long int dreccount(int nodb)          //Mod Julio
{
        switch(nodb)
        {
                case 1:return(d4reccount(alcanfile));
                case 3:return(d4reccount(alcanfile3));
                default:return (0);
        }
}

int success()
{
        return r4success;
}

int dskip(int nodb)
{
        int rc;
        switch (nodb) {
                case 1:rc = d4skip(alcanfile,1L);break;
                case 3:rc = d4skip(alcanfile3,1L);break;
        }
```

```
        return rc;
}

int deof(int nodb) {
        int res;
        switch (nodb) {
                case 1:res=d4eof(alcanfile);break;
                case 3:res=d4eof(alcanfile3);break;

        }
        return res;
}

void dinit(){
        d4init(&codebase);
}

void dinit_undo(){
        d4init_undo(&codebase);
}

void dpack(int nodb){
        switch (nodb) {
                case 1:  d4pack(alcanfile);break;
                case 3:  d4pack(alcanfile3);break;
        }
}

void dzap(int nodb){
        switch (nodb) {
                case 1:  d4zap(alcanfile,1L,d4reccount(alcanfile));break;
                case 3:  d4zap(alcanfile3,1L,d4reccount(alcanfile3));break;
        }
}
```

```
// ********************************************************************************
// SISTEMA: ALCANCIAS 3.4
//
// ARCHIVO: MANEJODB.C
//
// FECHA:  06/10/96
// ********************************************************************************
#include <stdio.h>
#include <conio.h>
#include "debug.h"
#include "bases.h"
#include "nomencla.h"
#include "d4all.h"

CODE4 codebase;
DATA4 *alcanfile=0;
DATA4 *alcanfile3=0;
INDEX4 *alcanindex=0;
INDEX4 *alcanindex3=0;

FIELD4 *FNum_Tel;
FIELD4 *FClave_Pob;
FIELD4 *FAlarma;
FIELD4 *FGrupo;
FIELD4 *FTipo_Apto;
FIELD4 *FTipoAmb;
FIELD4 *FLugar_Ins;
FIELD4 *FFecha_Ins;
FIELD4 *FCalleyNum;
FIELD4 *FColonia;
FIELD4 *FC_P;
FIELD4 *FDistrito;
FIELD4 *FPrincipal;
FIELD4 *FSecundario;
FIELD4 *FZona;
FIELD4 *FSuspendido;
FIELD4 *FSinControl;
FIELD4 *FTServicio;
FIELD4 *FCNCBA;
FIELD4 *FCNTLI;
FIELD4 *FCNBIC;
FIELD4 *FCNSSI;
FIELD4 *FCNOBA;
FIELD4 *FCNTCL;
FIELD4 *FCNICS;
FIELD4 *FCNBOC;
FIELD4 *FCNTBO;
FIELD4 *FCNBADP;
FIELD4 *FCNORPT;
FIELD4 *FCNLNCHAR;
FIELD4 *FCNTRSTR;
FIELD4 *FCNSSIG;
FIELD4 *FCNCBOX;
FIELD4 *FCNHM;
```

```
FIELD4 *FCACBA;
FIELD4 *FCATLI;
FIELD4 *FCABIC;
FIELD4 *FCASSI;
FIELD4 *FCAOBA;
FIELD4 *FCATCL;
FIELD4 *FCAICS;
FIELD4 *FCABOC;
FIELD4 *FCATBO;
FIELD4 *FCABADP;
FIELD4 *FCAORPT;
FIELD4 *FCALNCHAR;
FIELD4 *FCATRSTR;
FIELD4 *FCASSIG;
FIELD4 *FCACBOX;
FIELD4 *FCAHM;
FIELD4 *FSistTel;
FIELD4 *FStatus;
FIELD4 *FStatDesc;
FIELD4 *FSinMov;
FIELD4 *FCatNok;
FIELD4 *FContadorU;
FIELD4 *FContadorP;
FIELD4 *FHoraCatU;
FIELD4 *FHoraCatP;
FIELD4 *FFechaCatU;
FIELD4 *FFechaCatP;
FIELD4 *FHoraContU;
FIELD4 *FHoraContP;
FIELD4 *FFechaContU;
FIELD4 *FFechaContP;
FIELD4 *FSeguridad;
FIELD4 *FTelHis;
FIELD4 *FHoraHis;
FIELD4 *FFechaHis;
FIELD4 *FContHis;

TAG4   *Tele_Tag,*Zona_Tag,*TSer_Tag,*Posf_Tag;
TAG4  *Tele_Tag3;

FIELD4INFO fieldinfo[] =
{
        {"NUM_TEL",   r4str,S_NUM_TEL,0},
        {"CLAVE_POB", r4str,S_CLAVE_POB,0},
        {"ALARMA",    r4str,S_ALARMA,0},
        {"GRUPO",     r4str,S_GRUPO,0},
        {"TIPO_APTO", r4str,S_TIPO_APTO,0},
        {"TIPOAMB",   r4str,S_TIPOAMB,0},
        {"LUGAR_INS", r4str,S_LUGAR_INS,0},
        {"FECHA_INS", r4date,S_FECHA_INS,0},
        {"CALLEYNUM", r4str,S_CALLEYNUM,0},
        {"COLONIA",   r4str,S_COLONIA,0},
        {"C_P",       r4str,S_C_P,0},
        {"DISTRITO"   r4str,S_DISTRITO,0},
```

```
{"PRINCIPAL", r4str,S_PRINCIPAL,0},
{"SECUNDARIO",r4str,S_SECUNDARIO,0},

{"ZONA",     r4str,S_ZONA,0},
{"SUSPENDIDO",r4num,S_SUSPENDIDO,0},
{"SINCONTROL",r4num,S_SINCONTROL,0},
{"TSERVICIO",r4num,S_TSERVICIO,0},

{"CNCBA",       r4num,S_CNCA,0},
{"CNTLI",       r4num,S_CNCA,0},
{"CNBIC",       r4num,S_CNCA,0},
{"CNSSI",       r4num,S_CNCA,0},
{"CNOBA",       r4num,S_CNCA,0},
{"CNTCL",       r4num,S_CNCA,0},
{"CNICS",       r4num,S_CNCA,0},
{"CNBOC",       r4num,S_CNCA,0},
{"CNTBO",       r4num,S_CNCA,0},
{"CNBADP",   r4num,S_CNCA,0}, // Sistema 12
{"CNCBOX",   r4num,S_CNCA,0}, // Sistema 12
{"CNORPT",   r4num,S_CNCA,0}, // Sistema 12
{"CNSSIG",   r4num,S_CNCA,0}, // Sistema 12
{"CNTRSTR",  r4num,S_CNCA,0}, // Sistema 12
{"CNLNCHAR", r4num,S_CNCA,0}, // Sistema 12
{"CNHM",     r4num,S_CNCA,0}, // Sistema 12
{"CACBA",       r4num,S_CNCA,0},
{"CATLI",       r4num,S_CNCA,0},
{"CABIC",       r4num,S_CNCA,0},
{"CASSI",       r4num,S_CNCA.0},
{"CAOBA",       r4num,S_CNCA.0},
{"CATCL",       r4num,S_CNCA,0},
{"CAICS",       r4num,S_CNCA,0},
{"CABOC",       r4num,S_CNCA,0},
{"CATBO",       r4num,S_CNCA,0},
{"CABADP",   r4num,S_CNCA,0}, // Sistema 12
{"CACBOX",   r4num,S_CNCA,0}, // Sistema 12
{"CAORPT",   r4num,S_CNCA,0}, // Sistema 12
{"CASSIG",   r4num,S_CNCA,0}, // Sistema 12
{"CATRSTR",  r4num,S_CNCA,0}, // Sistema 12
{"CALNCHAR", r4num,S_CNCA,0}, // Sistema 12
{"CAHM",            r4num,S_CNCA,0}, // Sistema 12
{"SISTTEL",     r4num,S_TCENTRAL,0}, // Sistema 12
{"STATUS",      r4num,S_STATCATCNT,0},
{"STATDESC", r4str,S_STATDESC,0},
{"SINMOV",      r4num,S_SINMOV,0},
{"CATNOK",      r4num,S_CATNOK,0},
{"CONTADORU", r4num,S_CONTADOR,0},
{"CONTADORP", r4num,S_CONTADOR,0},
{"HORACATU", r4str,S_HORA,0},
{"HORACATP", r4str,S_HORA,0},
{"HORACONTU", r4str,S_HORA,0},
{"HORACONTP", r4str,S_HORA,0},
{"FECHACATU", r4str,S_FECHA,0},
{"FECHACATP", r4str,S_FECHA,0},
{"FECHACONTU",r4str,S_FECHA,0},
```

```
        {"FECHACONTP",r4str,S_FECHA,0},
        {"SEGURIDAD", r4str,S_SEGURIDAD,0},
        {0,0,0,0} ,
};

FIELD4INFO fieldinfohis[] =
{
        {"TELHIS",r4str,S_NUM_TEL,0},
        {"HORAHIS",r4num,S_HORA,0},
        {"FECHAHIS",r4date,8,0},
        {"CONTHIS",r4num,10,0},
        {0,0,0,0} ,
};

TAG4INFO tag_info[] =
{
        {"TELE_TAG","NUM_TEL",0,0,0},
        {"ZONA_TAG","ZONA+NUM_TEL",0,0,0},
        {"TSER_TAG","STR(TSERVICIO)+NUM_TEL",0,0,0},
        {"POSF_TAG" ,"STR(TSERVICIO)+NUM_TEL","SINMOV=1 .OR. CATNOK=1",0,0},
        {0,0,0,0,0},
};

TAG4INFO tag_infohis[] =
{
        {"TELE_TAG","TELHIS+DTOS(FECHAHIS)",0,0,0},
        {0,0,0,0,0},
};

void opendb(int nodb)
{
//      #ifdef N4OTHER
//                      codebase.auto_open = 0;
//      #endif
        codebase.open_error = 0;
        switch (nodb) {
                case 1:
                        if ((alcanfile=d4open(&codebase,NF_ALCANCIA))==0) {
                                alcanfile=d4create(&codebase,NF_ALCANCIA,fieldinfo,tag_info);

                        }
//                      if ((alcanindex=i4open(alcanfile,NX_ALCANCIA))==0) {
//                              alcanindex=i4create(alcanfile,NX_ALCANCIA,tag_info);
//                      }
                        FNum_Tel = d4field(alcanfile,"NUM_TEL");
                        FClave_Pob= d4field(alcanfile,"CLAVE_POB");
                        FAlarma  = d4field(alcanfile,"ALARMA");
                        FGrupo   = d4field(alcanfile,"GRUPO");
                        FTipo_Apto= d4field(alcanfile,"TIPO_APTO");
                        FTipoAmb = d4field(alcanfile,"TIPOAMB");
                        FLugar_Ins= d4field(alcanfile,"LUGAR_INS");
                        FFecha_Ins= d4field(alcanfile,"FECHA_INS");
                        FCalleyNum= d4field(alcanfile,"CALLEYNUM");
                        FColonia = d4field(alcanfile,"COLONIA");
```

```
FC_P     = d4field(alcanfile,"C_P");
FDistrito = d4field(alcanfile,"DISTRITO");
FPrincipal= d4field(alcanfile,"PRINCIPAL");
FSecundario= d4field(alcanfile,"SECUNDARIO");

FZona     = d4field(alcanfile,"ZONA");
FSuspendido= d4field(alcanfile,"SUSPENDIDO");
FSinControl= d4field(alcanfile,"SINCONTROL");
FTServicio = d4field(alcanfile,"TSERVICIO");

FCNCBA    = d4field(alcanfile,"CNCBA");
FCNTLI    = d4field(alcanfile,"CNTLI");
FCNBIC    = d4field(alcanfile,"CNBIC");
FCNSSI    = d4field(alcanfile,"CNSSI");
FCNOBA    = d4field(alcanfile,"CNOBA");
FCNTCL    = d4field(alcanfile,"CNTCL");
FCNICS    = d4field(alcanfile,"CNICS");
FCNBOC    = d4field(alcanfile,"CNBOC");
FCNTBO    = d4field(alcanfile,"CNTBO");
FCNBADP   = d4field(alcanfile,"CNBADP"); // Sistema 12
FCNCBOX   = d4field(alcanfile,"CNCBOX"); // Sistema 12
FCNORPT   = d4field(alcanfile,"CNORPT"); // Sistema 12
FCNSSIG   = d4field(alcanfile,"CNSSIG"); // Sistema 12
FCNTRSTR        = d4field(alcanfile,"CNTRSTR"); // Sistema 12
FCNLNCHAR = d4field(alcanfile,"CNLNCHAR"); // Sistema 12
FCNHM     = d4field(alcanfile,"CNHM"); // Sistema 12
FCACBA    = d4field(alcanfile,"CACBA");
FCATLI    = d4field(alcanfile,"CATLI");
FCABIC    = d4field(alcanfile,"CABIC");
FCASSI    = d4field(alcanfile,"CASSI");
FCAOBA    = d4field(alcanfile,"CAOBA");
FCATCL    = d4field(alcanfile,"CATCL");
FCAICS    = d4field(alcanfile,"CAICS");
FCABOC    = d4field(alcanfile,"CABOC");
FCATBO    = d4field(alcanfile,"CATBO");
FCABADP   = d4field(alcanfile,"CABADP"); // Sistema 12
FCACBOX   = d4field(alcanfile,"CACBOX"); // Sistema 12
FCAORPT   = d4field(alcanfile,"CAORPT"); // Sistema 12
FCASSIG   = d4field(alcanfile,"CASSIG"); // Sistema 12
FCATRSTR        = d4field(alcanfile,"CATRSTR"); // Sistema 12
FCALNCHAR = d4field(alcanfile,"CALNCHAR"); // Sistema 12
FCAHM     = d4field(alcanfile,"CAHM"); // Sistema 12
FSistTel  = d4field(alcanfile,"SISTTEL"); // Sistema 12
FStatus   = d4field(alcanfile,"STATUS");
FStatDesc = d4field(alcanfile,"STATDESC");
FSinMov   = d4field(alcanfile,"SINMOV");
FCatNok   = d4field(alcanfile,"CATNOK");
FContadorU = d4field(alcanfile,"CONTADORU");
FContadorP = d4field(alcanfile,"CONTADORP");
FHoraCatU  = d4field(alcanfile,"HORACATU");
FHoraCatP  = d4field(alcanfile,"HORACATP");
FHoraContU = d4field(alcanfile,"HORACONTU");
FHoraContP = d4field(alcanfile,"HORACONTP");
FFechaCatU = d4field(alcanfile,"FECHACATU");
```

```
                        FFechaCatP = d4field(alcanfile,"FECHACATP");
                        FFechaContU= d4field(alcanfile,"FECHACONTU");
                        FFechaContP= d4field(alcanfile,"FECHACONTP");
                        FSeguridad = d4field(alcanfile,"SEGURIDAD");
                        Tele_Tag  = d4tag(alcanfile,"TELE_TAG");
                        Zona_Tag  = d4tag(alcanfile,"ZONA_TAG");
                        TSer_Tag  = d4tag(alcanfile,"TSER_TAG");
                        Posf_Tag  = d4tag(alcanfile,"POSF_TAG");
                        d4tag_select(alcanfile,Tele_Tag);
                        break;
                case 3:
                        if ((alcanfile3=d4open(&codebase,NF_CONTHIST))==0) {

        alcanfile3=d4create(&codebase,NF_CONTHIST,fieldinfohis,tag_infohis);
                        }
//                      if ((alcanindex3=i4open(alcanfile3,NX_CONTHIST))==0) {
//                              alcanindex3=i4create(alcanfile3,NX_CONTHIST,tag_infohis);
//                      }
                        FTelHis=d4field(alcanfile3,"TELHIS");
                        FHoraHis=d4field(alcanfile3,"HORAHIS");
                        FFechaHis=d4field(alcanfile3,"FECHAHIS");
                        FContHis=d4field(alcanfile3,"CONTHIS");
                        Tele_Tag3=d4tag(alcanfile3,"TELE_TAG");
                        d4tag_select(alcanfile3,Tele_Tag3);
                        break;
        }
}

void dtag_select(int opcion,int nodb)
{
        switch (nodb) {
                case 1:
                        switch (opcion) {
                                case 1:  d4tag_select(alcanfile,Tele_Tag);break;
                                case 2:  d4tag_select(alcanfile,Zona_Tag);break;
                                case 3:  d4tag_select(alcanfile,TSer_Tag);break;
                                case 4:  d4tag_select(alcanfile,Posf_Tag);break;
                        }
                        break;
                case 3:
                        switch (opcion) {
                                case 1:  d4tag_select(alcanfile3,Tele_Tag3);break;
                        }
                        break;
        }
}

void reindexdb(int nodb) {
        switch (nodb) {
                case 1:  d4reindex(alcanfile);break;
                case 3:  d4reindex(alcanfile3);break;
        }
}
```

```
void closedb(int nodb)
{
        switch (nodb) {
                case 1: if(alcanfile ) d4close(alcanfile );break;
                case 3: if(alcanfile3) d4close(alcanfile3);break;
        }
}
```

```
// ****************************************************************************
// SISTEMA: ALCANCIAS 3.4 (AXE, SISTEMA 12)
//
// ARCHIVO: PRUELIN.C
//
// FECHA:  06/10/96
// ****************************************************************************
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include <cscape.h>
#include <dos.h>
#include "debug.h"
#include "comm.h"
#include "bases.h"
#include "nomencla.h"
#include "serial.h"
#include "alcan.h"
#include "pruelin.h"
#include "manejodb.h"
#include "gralfunc.h"
#include "gralfunx.h"
#include "gralfunp.h"
#include "reconoce.h"
#include "reportes.h"
#define NOP_S12 1
#define NOP_AXE 20

long telprobar=0;
long telprobar2;
int ciclo;
int contvcont=1;
int contvcat=1;
char string_to_send2[S_COMANDO];

extern boolean autocorrect;
extern boolean autoslomi;
extern boolean autocat;
extern boolean autoborr;
extern boolean autoimpr;
extern int totvuecont;
extern int totvuecat;
extern int fin_proc;
extern int sistema;
extern int digitos;
extern int ult_tecla;
extern int badport;
extern int catadic;
extern int puerto;

extern ALCANBTRV data_buf1;
extern CONTBTRV data_buf3;
extern char string_to_send[S_COMANDO];
```

```
extern char z_resp[S_RESP];
extern char login[S_LOGIN];
extern char loginfijo[S_LOGIN];
extern char password[S_PSW];
extern char telmodem[S_TELEFONO+1];
extern char string_no_rec[20];
extern char ultimotel[S_TELEFONO];
extern char ulttelprobar[S_TELEFONO];
extern char msg[S_MSG];
extern char telmodemant[N_PUERTOS][S_TELEFONO+1];

extern sed_type sLINEA,sAOM,sTRADUC,sPROC,seddesal;

void iLINEA(char *);
int iAOM();
int iTRADUC();
int iBUFFERF(int &,int &);
int iPROC(int);
int iMSGPROC(int);
void iCAMBIATA();
void actregistro(int);
void impfalla(int);
void datetimecont();
void datetimecat();
void dateconthist();
int activa();
void actactiva();
int inittimmer2();
int expiratimmer2();
int corrige_cat(char*);
void quita_adic();

void edittel() {
int i,j;

        char numeroent[S_TELEFONO+1];
        char numerofin[S_TELEFONO+1];
        sprintf(numeroent,"%ld",telprobar);
        for (i=8-digitos,j=0;i<strlen(numeroent);i++,j++)
                numerofin[j]=numeroent[i];
        sscanf(numerofin,"%ld",&telprobar2);
}

// PRUEBA DE LINEA

void send_incentral() { // rutina para elegir central
        if (sistema==SISTAOM) {
                badport=0;
                sprintf(string_to_send,"%s>",login);
                check_rx_buf();
        }
}

int send_tomacentral() { // Rutina para tomar central
```

```
int resultado;

if (sistema==SISTAOM) {
        resultado=marca_modem();
        if (resultado!=1)
                return resultado;
        strcpy(string_to_send,"END;");
        check_rx_buf();
        strcpy(string_to_send,"LOGOF;");
        check_rx_buf();
        sprintf(string_to_send,"%s;",password);
        check_rx_buf();
}
if (sistema==SIST12 || sistema==SIST12MTM || sistema==SIST12PAM) {
        resultado=marca_modem();
        dispMsg("INICIA COMANDO ",IZQUIERDA);
        if (resultado!=1)
                return resultado;
        if (sistema==SIST12MTM||sistema==SIST12PAM) {
                        put_serial(ctrl_x);
                        put_serial(cr);
                        delay(1500);
                        if (look(ls)) {
                                vacia_buffer();
                                goto NOBREAKS;
                        }
        }

        do {

                if (!(f1_press_mess()==1 || fin_proc==1))
                        dispMsg("Break      ",IZQUIERDA);
                ctrl_break();
                put_serial(ctrl_x);
                put_serial(cr);
                if (!(f1_press_mess()==1 || fin_proc==1))
                        dispMsg("           ",IZQUIERDA);
                delay(2000);
                if (f1_press()==1)
                        break;
        } while (look(ls)==0);

NOBREAKS:
        if (sistema==SIST12MTM ||sistema==SIST12PAM)
                vacia_buffer();
        if (badport==1)                 // Solución a desconexión S12 5/Jul/95
                return 0;                       // regresa sin tomar central

        if (sistema==SIST12)
                sprintf(string_to_send,"%s;",password);
        else if (sistema==SIST12MTM || sistema==SIST12PAM)
                sprintf(string_to_send,"MM");
        check_rx_buf();
```

```
                delay(200);
        }
        if (sistema==SIST1OG11) {
                resultado=marca_modem();
                if (resultado!=1)
                        return resultado;
                string_to_send[0] = NULL;
                put_serial(cr);
                delay(1000);
                check_rx_buf();
        }
        if (sistema==SIST1OG3) {
                resultado=marca_modem();
                if (resultado!=1)
                        return resultado;
                string_to_send[0] = NULL;
                put_serial(ctrl_e);
                check_rx_buf();
                delay(100);
        }
        return(resultado);
}

void send_dejacentral() {
        if (sistema==SIST1OG11) (
                string_to_send[0] = NULL;
                put_serial(ctrl_d);
                delay(1000); // Estaba en 200 pero puede ser causa de problema
                check_rx_buf();
        }
        if (sistema==SIST1OG3) {
                delay(200);
                string_to_send[0] = NULL;
                put_serial(ctrl_d);
                check_rx_buf();
                delay(500);
        }
}

int send_complcomando() {
        int continua;
        int result=0;
        int conteject=0;
        char msg[35];

        if (sistema==SIST12 || sistema==SIST12MTM || sistema==SIST12PAM) {
                check_rx_buf();       // Lee el primer buffer
                inittimmer2();
                continua=TRUE;
                while (continua && expiratimmer2()==0) {
                        if (f1_press()==1)
                                break;
                        if ((_fstrstr(z_resp,"COMANDO ENVIADO A EJECUCION"))!=NULL) {
                                // Listo para leer siguiente bloque
```

```
                        if (!(f1_press_mess()==1 || fin_proc==1))
                                dispMsg("COMANDO ENVIADO A EJECUCION
",IZQUIERDA);
                        string_to_send[0]=NULL;
                        if (check_rx_buf()==0) {
                                break;
                        }

                }
                if (fin_proc==1) {
                        continua=FALSE;
                        break;
                }
                if ((_fstrstr(z_resp,"RESULTADOS A CONTINUACION"))!=NULL) {
                        // Listo para leer el ultimo bloque
                        if (!(f1_press_mess()==1 || fin_proc==1))
                                dispMsg("RESULTADOS A CONTINUACION
",IZQUIERDA);
                        string_to_send[0]=NULL;
                        if (check_rx_buf()==0)
                                break;
                        continua=FALSE;
                }
                if ((_fstrstr(z_resp,"CONTINUA EL REPORTE"))!=NULL) {
                        // Listo para leer el ultimo bloque
                        if (!(f1_press_mess()==1 || fin_proc==1))
                                dispMsg("CONTINUA EL REPORTE      ",IZQUIERDA);
                        string_to_send[0]=NULL;
                        if (check_rx_buf()==0)
                                break;
                        continua=FALSE;
                }
                if (fin_proc==1) {
                        continua=FALSE;
                        break;
                }
                if ((_fstrstr(z_resp,"EJECUTANDOSE"))!=NULL) {
                        // Hay que leer siguiente bloque porque se encuentra pausado
                        conteject++;
                        if (!(f1_press_mess()==1 || fin_proc==1)) {
                                sprintf(msg,"EJECUTANDOSE %d          ",conteject);
                                dispMsg(msg,IZQUIERDA);
                        }
                        string_to_send[0]=NULL;
                        inittimmer2();
                        check_rx_buf();

                }
                if (fin_proc==1) {
                        continua=FALSE;
                        break;
                }
                if ((_fstrstr(z_resp,"TIEMPO TERMINADO"))!=NULL) {
                        // Hay que leer siguiente bloque porque se encuentra pausado
```

```
                if (!(f1_press_mess()==1 || fin_proc==1))
                        dispMsg("TIEMPO TERMINADO          ",IZQUIERDA);
                result=95;
                continua=FALSE;
        }
        if (fin_proc==1) {
                continua=FALSE;
                break;
        }
        if ((_fstrstr(z_resp,"ERROR DE SEMANTICA"))!=NULL) {
                if (!(f1_press_mess()==1 || fin_proc==1))
                        dispMsg("ERROR DE SEMANTICA        ",IZQUIERDA);
                string_to_send[0]=NULL;
                put_serial(ctrl_x);
                check_rx_buf();
                continua=FALSE;
        }
        if ((_fstrstr(z_resp,"BUFFER OVERFLOW"))!=NULL) {
                if (!(f1_press_mess()==1 || fin_proc==1))
                        dispMsg("BUFFER OVERFLOW           ",IZQUIERDA);
                string_to_send[0]=NULL;
                put_serial(ctrl_x);
                check_rx_buf();
                continua=FALSE;
        )
}

if ((_fstrstr(z_resp,"CON EXITO"))!=NULL) {
        if (!(f1_press_mess()==1 || fin_proc==1))
                dispMsg("CON EXITO          ",IZQUIERDA);
        result=1;
}
if ((_fstrstr(z_resp,"SIN EXITO"))!=NULL) {
        if (!(f1_press_mess()==1 || fin_proc==1))
                dispMsg("SIN EXITO          ",IZQUIERDA);
        result=1;
}
if ((_fstrstr(z_resp,"MD OCUPADO"))!=NULL) {
        if (!(f1_press_mess()==1 || fin_proc==1))
                dispMsg("MD OCUPADO         ",IZQUIERDA);
        result=93;
        string_to_send[0]=NULL;
        put_serial(ctrl_x);
        check_rx_buf();
        continua=FALSE;
}
if ((_fstrstr(z_resp,"REP NO EXITOSO YA ENVIADO"))!=NULL) {
        if (!(f1_press_mess()==1 || fin_proc==1))
                dispMsg("REP NO EXITOSO     ",IZQUIERDA);
        result=93;
}
if ((_fstrstr(z_resp,"ERROR DE COMPATIBILIDAD"))!=NULL) {
        if (!(f1_press_mess()==1 || fin_proc==1))
                dispMsg("ERROR DE COMPATIBILIDAD   ",IZQUIERDA);
```

```
                result=110;
            }
            if ((_fstrstr(z_resp,"NO SE ENCONTRO MODIF APLICABLE"))!=NULL) {
                if (!(f1_press_mess()==1 || fin_proc==1))
                    dispMsg("NO HAY MODIFICACION APLIC.   ",IZQUIERDA);
                result=111;
            }
            // put_serial(ctrl_x);  linea a prueba PARA EVITAR QUE SE TRABE
            // Desactivada el 2 de Septiembre de 1995 por Alfonso
        }
        return result;
}

int testpruelin(int flagtoma) {

        int resultado;
        if (sistema==SISTAOM || sistema==SISTIOG11 || sistema==SISTIOG3) {
            if (sistema==SISTIOG11 || sistema==SISTIOG3 || (sistema==SISTAOM &&
flagtoma==1)) {
                resultado=send_tomacentral();
                if (resultado!=1)
                    return resultado;
            }
            if (sistema==SISTAOM && flagtoma==1)
                send_incentral();

            edittel();
            sprintf(string_to_send,"SLOMI:SNB=%ld,MP=5;",telprobar2);
            check_rx_buf();
        }
        if (sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM) {
            resultado=send_tomacentral();
            if (resultado!=1)
                return resultado;
            if (sistema==SIST12PAM)
                sprintf(string_to_send,"518:3=K'%ld,6=11,8;",telprobar);
            else
                sprintf(string_to_send,"518:3=K'%ld,4=2;",telprobar);
            resultado=send_complcomando();
            if (resultado!=1)
                interpretapruc(resultado);

        }
        return resultado;
}

void endpruelin() {
        if (sistema==SISTAOM) {
            strcpy(string_to_send,"END;");
            check_rx_buf();
        }
        if (sistema==SISTIOG11 || sistema==SISTIOG3) {
            strcpy(string_to_send,"END;");
            delay(200);
```

```
                check_rx_buf();
                send_dejacentral();
        }
}

void send_endall() {
        endpruelin();
        if (sistema==SISTAOM) {
                strcpy(string_to_send,"LOGOF;");
                check_rx_buf();
        }
}

int testconscont(int flagtoma)
{
        int resultado;

        if (sistema==SISTAOM || sistema==SISTIOG3 || sistema==SISTIOG11) {
                if (sistema==SISTIOG11 || sistema==SISTIOG3 || (sistema==SISTAOM &&
flagtoma==1)) {
                        resultado=send_tomacentral();
                        if (resultado!=1)
                                return resultado;
                        send_incentral();
                }
                edittel();
                sprintf(string_to_send,"CHSIP:SNB=%ld;",telprobar2);
                check_rx_buf();
        }
        if (sistema==SIST12 || sistema==SIST12MTM||sistema==SIST12PAM) {
                resultado=send_tomacentral();
                if (resultado!=1)
                        return resultado;
                if (sistema!=SIST12PAM)
                        sprintf(string_to_send,"88:I=K'%ld;",telprobar);
                else
                        sprintf(string_to_send,"4296:I=K'%ld;",telprobar);
                resultado=send_complcomando();
                if (resultado!=1)
                        interpretacont(resultado);
        }
        return 1;
}

void endconscont() {
        send_dejacentral();
}

int testdesb(int flagtoma)
{
        int resultado;

        if (sistema==SISTAOM || sistema==SISTIOG3 || sistema==SISTIOG11) {
```

```
                if (sistema==SISTIOG11 || sistema==SISTIOG3 || (sistema==SISTAOM &&
flagtoma==1)) {
                        resultado=send_tomacentral();
                        if (resultado!=1)
                                return resultado;
                        send_incentral();
                }
                edittel();
                sprintf(string_to_send,"BLOLE:SNB=%ld;",telprobar2);
                check_rx_buf();
        }
        return 1;
}

void enddesb() {
        send_dejacentral();
}

int testcambcat(int flagtoma)
{
        int resultado;
        char cadpaso1[6],cadpaso2[6],cadpaso3[6],cadpaso4[6];
        char cadenacomando[S_COMANDO];

        if (sistema==SISTAOM || sistema==SISTIOG3 || sistema==SISTIOG11) {
                if (sistema==SISTIOG11 || sistema==SISTIOG3 || (sistema==SISTAOM &&
flagtoma==1)) {
                        resultado=send_tomacentral();
                        if (resultado!=1)
                                return resultado;
                        send_incentral();
                }
                edittel();
                sprintf(string_to_send,"SUSCC:SNB=%ld,SCL=CBA-%d&TLI-%d&BIC-%d&SSI-
%d&OBA-%d&TCL-%d&ICS-%d&BOC-%d&TBO-%d;",
                telprobar2,data_buf1.cnCBA,data_buf1.cnTLI,data_buf1.cnBIC,data_buf1.cnSSI,

        data_buf1.cnOBA,data_buf1.cnTCL,data_buf1.cnICS,data_buf1.cnBOC,data_buf1.cnTBO);
                sprintf(string_to_send2,"%s",string_to_send);
                check_rx_buf();
                strcpy(string_to_send,";");
                check_rx_buf();
                editz_resp2(string_to_send2);
        }
        if (sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM) {
                resultado=send_tomacentral();
                if (resultado!=1)
                        return resultado;
                if (sistema!=SIST12PAM) {
                        s_CBOX(data_buf1.cnCBOX,cadpaso1);

        sprintf(string_to_send,"87:1=K'%ld,BADP=%d,ORPT=%d,LNCHAR=%d,TRSTR=%d,CBOX=%s
.SSIG=%d,HM=%d;",telprobar,
```

```
        data_buf1.cnBADP,data_buf1.cnORPT,data_buf1.cnLNCHAR,data_buf1.cnTRSTR,cadpaso1,data_
buf1.cnSSIG,data_buf1.cnHM);
                        }
                else
                        quita_adic();
                        s_OCB(data_buf1.cnORPT,cadpaso1);
                        s_LINECHAR(data_buf1.cnLNCHAR,cadpaso2);
                        s_ICB(data_buf1.cnTRSTR,cadpaso3);
                        s_PAYPHONE(data_buf1.cnCBOX,cadpaso4);
                        if (data_buf1.cnORPT==120)

        sprintf(cadenacomando,"4294:1=K'%ld,INTCP=%d,LINECHAR=%s,ICB=%s,PAYPHONE=%s,S
UBSIG=%d;",telprobar,
                                data_buf1.cnBADP,cadpaso2,cadpaso3,cadpaso4,data_buf1.cnSSIG);
                        else

        sprintf(cadenacomando,"4294:1=K'%ld,INTCP=%d,OCB=%s,LINECHAR=%s,ICB=%s,PAYPHO
NE=%s,SUBSIG=%d;",telprobar,

        data_buf1.cnBADP,cadpaso1,cadpaso2,cadpaso3,cadpaso4,data_buf1.cnSSIG);
                        resultado=corrige_cat(cadenacomando);
                if (resultado!=1)
                        interpretacate(resultado);
        }
        return resultado;
}

void endcambcat() {
        send_dejacentral();
}

int testconscat(int flagtoma)
{
        int resultado;

        if (sistema==SISTAOM || sistema==SISTIOG3 || sistema==SISTIOG11) {
                if (sistema==SISTIOG11 || sistema==SISTIOG3 || (sistema==SISTAOM &&
flagtoma==1)) {
                        resultado=send_tomacentral();
                        if (resultado!=1)
                                return resultado;
                        send_incentral();
                }
                editicl();
                sprintf(string_to_send,"SUSCP:SNB=%ld;",telprobar2);
                check_rx_buf();
        }
        if (sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM) {
                resultado=send_tomacentral();
                if (resultado!=1)
                        return resultado;
                if (sistema!=SIST12PAM)
                        sprintf(string_to_send,"88:1=K'%ld;",telprobar);
```

```
                else
                        sprintf(string_to_send,"4296:1=K'%ld;",telprobar);
                resultado=send_complcomando();
                if (resultado!=1)
                        interpretacate(resultado);
        }
        return 1;
}

void endconscat() {
        send_dejacentral();
}

void plinea() {
        int respuesta;
        int tp;
        int resultado;
        int desppant;
        char enterol[11];
        char msg[35];


        iMSGPROC(10);
        iLINEA(enterol);
        sprintf(enterol,"%ld",telprobar);
        sed_Repaint(sLINEA);
        while (sed_Go(sLINEA)!=0)  {
                dispMsg("                        ",IZQUIERDA);
                sscanf(enterol,"%ld",&telprobar);
                respuesta=isvalid();
                if (sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM)
                        respuesta=2;
                tp=0;
                desppant=0;
                sprintf(data_buf1.telefono,"%s",enterol); // Prepara para la b£squeda
                if (!((respuesta==1 || respuesta==2) && getequdb(1)==0)) {
                        dispMsg("N£mero fuera del alcance      ",IZQUIERDA);
                        getch();
                        dispMsg("                        ",IZQUIERDA);
                        }
                else {
                        if (respuesta==1) {
                                        dispMsg("Numero fuera del alcance",IZQUIERDA);
                                        getch();
                        }       else {
                                sprintf(msg,"Probando linea:%8ld ",telprobar);
                                dispMsg(msg,CENTRO);
                                resultado=testpruelin(TOMACENT);
                                if (resultado>=100 && resultado<=109)
                                        interpretaprue(resultado);
                                else
                                        reconoceprue(tp);
                                desppant=1;
                        }
```

```
                dispMsg("                    ",CENTRO);
                if (desppant==1) {
                        sprintf(msg,"Linea Probada:%8ld",telprobar);
                        dispMsg(msg,DERECHA);
                        editz_resp(); // Se edita respuesta
                        iAOM();
                        iTRADUC();
                        if (resultado==1)
                                endpruelin();
                        dispMsg("<ESCAPE> Continuar          ",IZQUIERDA);
                        sed_Go(sAOM);
                        sed_Close(sAOM);
                        sed_Close(sTRADUC);
                        }
                }
                sprintf(enterol,"%ld",telprobar);
                sed_Repaint(sLINEA);
                dispMsg("                    ",DERECHA);
                dispMsg("<ESC> Salir                 ",IZQUIERDA);
        }
        if (resultado==100)
                sprintf(telmodemant[puerto-1],"-1");
        sed_Close(sLINEA);
        sed_Close(sPROC);
        dispMsg("                        ",IZQUIERDA);
}

void pconscont() {
        int respuesta;
        int resultado;
        int desppant;
        char enterol[11];
        char msg[35];

        iMSGPROC(9);
        iLINEA(enterol);
        sprintf(enterol,"%ld",telprobar);
        sed_Repaint(sLINEA);
        while (sed_Go(sLINEA)!=0) {
                dispMsg("                        ",IZQUIERDA);
                sscanf(enterol,"%ld",&telprobar);
                respuesta=isvalid();
                result=0;
                desppant=0;
                sprintf(data_buf1.telefono,"%s",enterol); // Prepara para la b£squeda
                if (!((respuesta==1 || respuesta==2) && getequdb(1)==0)) {
                        dispMsg("Número fuera del alcance   ",IZQUIERDA);
                        getch();
                        dispMsg("                        ",IZQUIERDA);
                        }
                else {
                        if (respuesta==1) {
                                dispMsg("Número fuera del alcance",IZQUIERDA);
                                getch();
```

```
                ;       else {
                        sprintf(msg,"Consult.Cont:%8ld ",telprobar);
                        dispMsg(msg,CENTRO);
                        datetimecont();
                        resultado=testconscont(TOMACENT);
                        if (resultado>=100 && resultado<=109)
                                interpretacont(resultado);
                        else
                                data_buf1.status=reconocecont();
                        desppant=1;
                }
                dispMsg("                ",CENTRO);
                if (desppant==1 && ult_tecla!=ESC) {
                        sprintf(msg,"Cont.Consult:%8ld",telprobar);
                        dispMsg(msg,DERECHA);
                        editz_resp(); // Se edita respuesta
                        iAOM();
                        iTRADUC();
                        if (resultado==1)
                                endconscont();
                        dispMsg("<ESCAPE> Continuar      ",IZQUIERDA);
                        sed_Go(sAOM);
                        sed_Close(sAOM);
                        sed_Close(sTRADUC);
                        }
        }
        sprintf(enterol,"%ld",telprobar);
        sed_Repaint(sLINEA);
        dispMsg("                ",DERECHA);
        dispMsg("<ESC> Salir             ",IZQUIERDA);
    }
    if (resultado==100)
            sprintf(telmodemant[puerto-1],"-1");
    sed_Close(sLINEA);
    sed_Close(sPROC);
    dispMsg("                ",IZQUIERDA);
}

void psuspension() {
    int respuesta;
    int resultado;
    int resulcate;
    int desppant;
    char enterol[11];
    char bitacora[11];
    char msg[80];

    iMSGPROC(4);
    iLINEA(enterol);
    sprintf(enterol,"%ld",telprobar);
    sed_Repaint(sLINEA);
    while (sed_Go(sLINEA)!=0) {
            dispMsg("                ",IZQUIERDA);
            sscanf(enterol,"%ld",&telprobar);
```

```
respuesta=isvalid();
desppant=0;
sprintf(data_buf1.telefono,"%s",enterol); // Prepara para la b£squeda
if (!((respuesta==1 || respuesta==2) && getequdb(1)==0)) {
        dispMsg("Número fuera del alcance    ",IZQUIERDA);
        getch();
        dispMsg("                    ",IZQUIERDA);
        }
else {
        if (respuesta==1) {
                dispMsg("Número fuera del alcance",IZQUIERDA);
                getch();
        }       else {
                iCAMBIATA();
                actregistro(1);
                if (data_buf1.suspendido==1) {
                        initsedcat();
                        sed_Go(seddesal);
                        sed_Close(seddesal);
                        if (ult_tecla!=ESC) {
                                iAOM();
                                iTRADUC();
                                sprintf(msg,"Susp linea:%8ld ",telprobar);
                                dispMsg(msg,CENTRO);
                                resultado=testcambcat(TOMACENT);
                                if (resultado>=100 && resultado<=109)
                                        interpretacate(resultado);
                                else
                                        resulcate=reconocecate(1);
                                desppant=1;
                                sed_Close(sAOM);
                                sed_Close(sTRADUC);
                        }
                } else
                        initsedCon(12);
                // Ojo antes estaba aqui desppant
        }
        dispMsg("                    ",CENTRO);
        if (desppant==1 && ult_tecla!=ESC) {
                sprintf(msg,"Linea Susp:%8ld",telprobar);
                dispMsg(msg,DERECHA);
                editz_resp();
                iAOM();
                iTRADUC();
                if (resultado==1) {
                        endcambcat();
                        if (resulcate==3 &&
(sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM)) {
                                datetimecat();
                                copymemdb(1);
                                iBITACORA(bitacora);
                                movhistmov(4,bitacora);
                        } else {
                                if (resulcate!=0) {     // Si no fue BUSY
```

```
                                                    if (reconocecate(1)==3 || (resulcate==3 &&
sistema==SISTIOG11)) {  // Si fu, EXECUTED O CON EXITO
                                                                datetimecat();
                                                                copymemdb(1);
                                                                iBITACORA(bitacora);
                                                                movhistmov(4,bitacora);
                                                        }
                                                }
                                        }
                                }
                                dispMsg("<ESCAPE> Continuar          ",IZQUIERDA);
                                sed_Go(sAOM);
                                sed_Close(sAOM);
                                sed_Close(sTRADUC);
                                }
                        }
                        sprintf(enterol,"%ld",telprobar);
                        sed_Repaint(sLINEA);
                        dispMsg("                    ",DERECHA);
                        dispMsg("<ESC> Salir                ",IZQUIERDA);
                }
                if (resultado==100)
                        sprintf(telmodemant[puerto-1],"-1");
                sed_Close(sLINEA);
                sed_Close(sPROC);
                dispMsg("                           ",IZQUIERDA);
}

void preanudacion() {
        int respuesta;
        int resultado;
        int resulcate;
        int desppant;
        char enterol[11];
        char bitacora[11];

        iMSGPROC(5);
        iLINEA(enterol);
        sprintf(enterol,"%ld",telprobar);
        sed_Repaint(sLINEA);
        while (sed_Go(sLINEA)!=0)  {
                gotoxy(3,24);
                cprintf("                              ");
                sscanf(enterol,"%ld",&telprobar);
                respuesta=isvalid();
                desppant=0;
                sprintf(data_buf1.telefono,"%s",enterol); // Prepara para la b£squeda
                if (!((respuesta==1 || respuesta==2) && getequdb(1)==0)) {
                        dispMsg("Número fuera del alcance    ",IZQUIERDA);
                        getch();
                        dispMsg("                           ",IZQUIERDA);
                        }
                else {
                        if (respuesta==1) {
```

```
                        dispMsg("Número fuera del alcance",IZQUIERDA);
                        getch();
                }       else {
                        iCAMBIATA();
                        actregistro(1);
                        if (data_buf1.suspendido==0) {
                                initsedcat();
                                sed_Go(seddesal);
                                sed_Close(seddesal);
                                if (ult_tecla!=ESC) {
                                        iAOM();
                                        iTRADUC();
                                        gotoxy(33,24);
                                        cprintf("Reanud linea:%8ld ",telprobar);
                                        resultado=testcambcat(TOMACENT);
                                        if (resultado>=100 && resultado<=109)
                                                interpretacate(resultado);
                                        else
                                                resulcate=reconocecate(1);
                                        desppant=1;
                                        sed_Close(sAOM);
                                        sed_Close(sTRADUC);
                                }
                        } else
                                initsedCon(13);
                }
                gotoxy(33,24);
                cprintf("                ");
                if (desppant==1 && ult_tecla!=ESC) {
                        gotoxy(57,24);
                        cprintf("Linea Reanud:%8ld",telprobar);
                        editz_resp();
                        iAOM();
                        iTRADUC();
                        if (resultado==1) {
                                endcambcat();
                                if (resulcate==3 &&
(sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM)) {
                                        datetimecat();
                                        copymemdb(1);
                                        iBITACORA(bitacora);
                                        movhistmov(4,bitacora);
                                } else {
                                        if (resulcate!=0 || sistema==SIST12
||sistema==SIST12MTM || sistema==SIST12PAM ) {
                                                if (reconocecate(1)==3 || (resulcate==3 &&
sistema==SISTIOG11)) { // Si fue EXECUTED O CON EXITO
                                                        datetimecat();
                                                        copymemdb(1);
                                                        iBITACORA(bitacora);
                                                        movhistmov(5,bitacora);
                                                }
                                        }
                                }
```

```
                        }
                        dispMsg("<ESCAPE> Continuar          ",IZQUIERDA);
                        scd_Go(sAOM);
                        scd_Close(sAOM);
                        scd_Close(sTRADUC);
                        }
                }
                sprintf(enterol,"%ld",telprobar);
                scd_Repaint(sLINEA);
                dispMsg("                ",DERECHA);
                dispMsg("<ESC> Salir              ",IZQUIERDA);
        }
        if (resultado==100)
                sprintf(telmodeman[puerto-1],"-1");
        scd_Close(sLINEA);
        scd_Close(sPROC);
        dispMsg("                        ",IZQUIERDA);
}

void pdesbloqueo() {
        int respuesta;
        int resultado;
        int desppant;
        char enterol[11];

        iMSGPROC(6);
        iLINEA(enterol);
        sprintf(enterol,"%ld",telprobar);
        scd_Repaint(sLINEA);
        while (scd_Go(sLINEA)!=0) {
                dispMsg("                        ",IZQUIERDA);
                sscanf(enterol,"%ld",&telprobar);
                respuesta=fsvalid();
                desppant=0;
                sprintf(data_buf1.telefono,"%s",enterol); // Prepara para la b£squeda
                if (!((respuesta==1 || respuesta==2) && getequdb(1)==0)) {
                        dispMsg("Número fuera del alcance    ",IZQUIERDA);
                        getch();
                        dispMsg("                        ",IZQUIERDA);
                        }
                else {
                        if (respuesta==1) {
                                        gotoxy(3,24);
                                        printf("Numero fuera del alcance");
                                        getch();
                        }       else {
                                gotoxy(33,24);
                                cprintf("Reanud l¡nea:%8ld ",telprobar);
                                resultado=testdesb(TOMACENT);
                                if (resultado>=100 && resultado<=109)
                                        interpretacnte(resultado);
                                else
                                        reconocecate(1);
```

```
                        desppant=1;
                }
                gotoxy(33,24);
                cprintf("                ");
                if (desppant==1) {
                        gotoxy(57,24);
                        cprintf("Linea Reanud:%8ld",telprobar);
                        iAOM();
                        iTRADUC();
                        if (resultado==1)
                                enddesb();
                        gotoxy(3,24);
                        cprintf("<ESCAPE> Continuar          ");
                        sed_Repaint(sAOM);
                        sed_Go(sAOM);
                        sed_Close(sAOM);
                        sed_Close(sTRADUC);
                        }
        }
        sprintf(enterol,"%ld",telprobar);
        sed_Repaint(sLINEA);
        gotoxy(57,24);
        cprintf("                ");
        gotoxy(3,24);
        cprintf("<ESC> Salir                ");
    }
    if (resultado==100)
        sprintf(telmodemant[puerto-1],"-1");
    sed_Close(sLINEA);
    sed_Close(sPROC);
    gotoxy(3,24);
    cprintf("                ");
}

void pconscat() {
    int respuesta;
    int resultado;
    int desppant;
    char enterol[11];

    iLINEA(enterol);
    iMSGPROC(7);
    sprintf(enterol,"%ld",telprobar);
    sed_Repaint(sLINEA);
    while (sed_Go(sLINEA)!=0)  {
        gotoxy(3,24);
        cprintf("                ");
        sscanf(enterol,"%ld",&telprobar);
        respuesta=isvalid();
//      result=0;
        desppant=0;
        sprintf(data_buf1.telefono,"%s",enterol); // Prepara para la b£squeda
        if (!((respuesta==1 || respuesta==2) && getequdb(1)==0)) {
            gotoxy(3,24);
```

```
                cprintf("N£mero fuera del alcance    ");
                getch();
                gotoxy(3,24);
                cprintf("                    ");
                }
        else {
                if (respuesta==1) {
                        gotoxy(3,24);
                        printf("Numero fuera del alcance");
                        getch();
                }       else {
                        iAOM();
                        iTRADUC();
                        gotoxy(33,24);
                        cprintf("Consult.Cat:%8ld ",telprobar);
                        resultado=testconscat(TOMACENT);
                        if (resultado>=100 && resultado<=109)
                                interpretacate(resultado);
                        else
                                reconocecate(1);
                        desppant=1;
                        sed_Close(sAOM);
                        sed_Close(sTRADUC);
                }
                gotoxy(33,24);
                cprintf("                    ");
                if (desppant==1) {
                        gotoxy(57,24);
                        cprintf("Cat.Consult:%8ld",telprobar);
                        editz_resp();
                        iAOM();
                        iTRADUC();
                        if (resultado==1)
                                endconscat();
                        gotoxy(3,24);
                        cprintf("<ESCAPE> Continuar          ");
                        sed_Go(sAOM);
                        sed_Close(sAOM);
                        sed_Close(sTRADUC);
                        }
        }
        sprintf(cntcrol,"%ld",telprobar);
        sed_Repaint(sLINEA);
        gotoxy(57,24);
        cprintf("                ");
        gotoxy(3,24);
        cprintf("<ESC> Salir                ");
}
if (resultado==100)
        sprintf(telmodemant[puerto-1],"-1");
sed_Close(sLINEA);
sed_Close(sPROC);
gotoxy(3,24);
cprintf("                    ");
```

```
                cprintf("N£mero fuera del alcance    ");
                getch();
                gotoxy(3,24);
                cprintf("                    ");
                }
        else {
                if (respuesta==1) {
                        gotoxy(3,24);
                        printf("Numero fuera del alcance");
                        getch();
                }       else {
                        iAOM();
                        iTRADUC();
                        gotoxy(33,24);
                        cprintf("Consult.Cat:%8ld ",telprobar);
                        resultado=testconscat(TOMACENT);
                        if (resultado>=100 && resultado<=109)
                                interpretacate(resultado);
                        else
                                reconocecate(1);
                        desppant=1;
                        sed_Close(sAOM);
                        sed_Close(sTRADUC);
                }
                gotoxy(33,24);
                cprintf("                    ");
                if (desppant==1) {
                        gotoxy(57,24);
                        cprintf("Cat.Consult:%8ld",telprobar);
                        editz_resp();
                        iAOM();
                        iTRADUC();
                        if (resultado==1)
                                endconscat();
                        gotoxy(3,24);
                        cprintf("<ESCAPE> Continuar          ");
                        sed_Go(sAOM);
                        sed_Close(sAOM);
                        sed_Close(sTRADUC);
                        }
        }
        sprintf(enterol,"%ld",telprobar);
        sed_Repaint(sLINEA);
        gotoxy(57,24);
        cprintf("                ");
        gotoxy(3,24);
        cprintf("<ESC> Salir                ");
}
if (resultado==100)
        sprintf(telmodemant[puerto-1],"-1");
sed_Close(sLINEA);
sed_Close(sPROC);
gotoxy(3,24);
cprintf("                    ");
```

```
}

void pcambcat(){
        int respuesta;
        int resultado;
        int resulcate;
        int desppant;
        char enterol[11];
        char bitacora[11];

        iMSGPROC(8);
        iLINEA(enterol);
        sprintf(enterol,"%ld",telprobar);
        scd_Repaint(sLINEA);
        while (scd_Go(sLINEA)!=0) {
                dispMsg("                    ",IZQUIERDA);
                sscanf(enterol,"%ld",&telprobar);
                respuesta=isvalid();
                desppant=0;
                sprintf(data_buf1.telefono,"%s",enterol); // Prepara para la b£squeda
                if (!((respuesta==1 || respuesta==2) && getequdb(1)==0)) {
                        dispMsg("Número fuera del alcance    ",IZQUIERDA);
                        getch();
                        dispMsg("                    ",IZQUIERDA);
                        }
                else {
                        if (respuesta==1) {
                                        dispMsg("Número fuera del alcance",IZQUIERDA);
                                        getch();
                        }       else {
                                initscdcat();
                                scd_Go(scddesal);
                                scd_Close(scddesal);
                                if (ult_tecla!=ESC) {
                                        iAOM();
                                        iTRADUC();
                                        gotoxy(33,24);
                                        cprintf("Camb.Cat:%8ld ",telprobar);
                                        resultado=testcambcat(TOMACENT);
                                        if (resultado>=100 && resultado<=109)
                                                interpretacate(resultado);
                                        else
                                                resulcate=reconocecate(1);
                                }
                                desppant=1;
                        }
                        gotoxy(33,24);
                        cprintf("                ");
                        if (desppant==1 && ult_tecla!=ESC) {
                                gotoxy(57,24);
                                cprintf("Camcat:%8ld",telprobar);
                                editz_resp();
                                iAOM();
                                iTRADUC();
```

```
                        if (resultado==1) {
                                endcambcat();
                                if (resulcate==3 &&
(sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM)) {
                                        datetimecat();
                                        copymemdb(1);
                                        iBITACORA(bitacora);
                                        movhistmov(4,bitacora);
                                } else {
                                        if (resulcate!=0 ||
sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM) {
                                                if (reconococate(1)==3 || (resulcate==3 &&
sistema==SISTIOG11)) {
                                                        copymemdb(1);
                                                        iBITACORA(bitacora);
                                                        movhistmov(6,bitacora);
                                                }
                                        }
                                }
                        }
                        dispMsg("<ESCAPE> Continuar        ",IZQUIERDA);
                        scd_Go(sAOM);
                        scd_Close(sAOM);
                        scd_Close(sTRADUC);
                        }
                }
                sprintf(enterol,"%ld",telprobar);
                scd_Repaint(sLINEA);
                dispMsg("                ",DERECHA);
                dispMsg("<ESC> Salir             ",IZQUIERDA);
        }
        if (resultado==100)
                sprintf(telmodemant[puerto-1],"-1");
        scd_Close(sLINEA);
        scd_Close(sPROC);
        gotoxy(3,24);
        cprintf("                    ");
}


int corrige_cat(char cadenacomando[S_COMANDO]) {
        int pruebaok;
        int resultado;
        int repitecomando=0;
        int flagocupado=0;
        int contpruebas=0;

        do {
                do {
                        do {
                                send_tomacentral();
                                sprintf(string_to_send,"%s",cadenacomando);
                                pruebaok=0;
                                resultado=send_complcomando();
```

```
                        if (resultado>=100 && resultado<=109)
                                interpretacate(resultado);
                        else
                                pruebaok=reconocecate(0);
                        disp_z_resp();
                        sed_Repaint(sTRADUC);
                } while (resultado==95);
//      printf("resultado >>> %d",resultado);getch();
                if (flagocupado==1) {
                        flagocupado=0;
                        repitecomando=0;
                        do {
                                send_tomacentral();
                                sprintf(string_to_send,"7:DN=K'%ld;",telprobar);
                                resultado=send_complcomando();
                                if (resultado>=100 && resultado<=109)
                                        interpretacate(resultado);
                                else
                                        pruebaok=reconocecate(0);
                                disp_z_resp();
                                sed_Repaint(sTRADUC);
                        } while (resultado==95);
                }

                if (resultado==93) {
                        flagocupado=1;
                        repitecomando=1;
                        do {
                                send_tomacentral();
                                sprintf(string_to_send,"6:DN=K'%ld,WTC=0;",telprobar);
                                resultado=send_complcomando();
                                if (resultado>=100 && resultado<=109)
                                        interpretacate(resultado);
                                else
                                        pruebaok=reconocecate(0);
                                disp_z_resp();
                                sed_Repaint(sTRADUC);
                        } while (resultado==95);
                }
        } while (repitecomando==1); //
        contpruebas++;
} while (!(contpruebas>NOP_S12 || pruebaok==1 || pruebaok==3 || (pruebaok==0 &&
resultado!=112) || fin_proc!=0));
        return pruebaok;
}

void quita_adic() {
        char cadenacomando[S_COMANDO];
        gotoxy(33,24);
        cprintf("QuitandoCat.:%8ld",telprobar);
        if (sistema!=SIST12PAM)

        sprintf(cadenacomando,"87:1=K'%ld,HM=5,TRSTR=2,COL=8,DBO=4,DBLNG=5,OOBS=3;",telp
robar);
```

```
        else

        sprintf(cadenacomando,"4294:1=K'%ld,PAYPHONE=1,ICB=2,CFWD=2,DBLNGOBS=2,FDC=2,
CDTAX=2,CFWD=2,ABS=1;",telprobar);
        corrige_cat(cadenacomando);
        if (sistema!=SIST12PAM)

        sprintf(cadenacomando,"87:1=K'%ld,RECALL=1,HOTL=3,LOOS=2,REDIR=5,RMCTR=1;",telpr
obar);
        else

        sprintf(cadenacomando,"4294:1=K'%ld,RECALL=2,NBRIDFCD=2,SUBCTRL=2,NTFCFWD=2,O
CBCFWD=2;",telprobar);
        corrige_cat(cadenacomando);
}

void correct()
{
        int necesitacom=0;
        char nocompleto[10];
        char paso[20];
        char cadpaso1[6];
        char cadpaso2[6];
        char cadenacomando[S_COMANDO];

        if (sistema==SISTAOM || sistema==SISTIOG3 || sistema==SISTIOG11) {
                data_buf1.catnok=0; // Linea a prueba
                edit(tel);
                strcpy(string_to_send2,"SUSCC:SNB=");
                sprintf(nocompleto,"%ld",telprobar2);
                strcat(string_to_send2,nocompleto);
                strcat(string_to_send2,",SCL=");
                if (data_buf1.cnCBA!=data_buf1.caCBA) {
                        if (necesitacom==1)
                                strcat(string_to_send2,"&");
                        sprintf(paso,"CBA-%d",data_buf1.cnCBA);
                        strcat(string_to_send2,paso);
                        necesitacom=1;
                }
                if (data_buf1.cnTLI!=data_buf1.caTLI) {
                        if (necesitacom==1)
                                strcat(string_to_send2,"&");
                        sprintf(paso,"TLI-%d",data_buf1.cnTLI);
                        strcat(string_to_send2,paso);
                        necesitacom=1;
                }
                if (data_buf1.cnBIC!=data_buf1.caBIC) {
                        if (necesitacom==1)
                                strcat(string_to_send2,"&");
                        sprintf(paso,"BIC-%d",data_buf1.cnBIC);
                        strcat(string_to_send2,paso);
                        necesitacom=1;
                }
                if (data_buf1.cnSSI!=data_buf1.caSSI) {
```

```
        if (necesitacom==1)
                strcat(string_to_send2,"&");
        sprintf(paso,"SSI-%d",data_buf1.cnSSI);
        strcat(string_to_send2,paso);
        necesitacom=1;
}
if (data_buf1.cnOBA!=data_buf1.caOBA) {
        if (necesitacom==1)
                strcat(string_to_send2,"&");
        sprintf(paso,"OBA-%d",data_buf1.cnOBA);
        strcat(string_to_send2,paso);
        necesitacom=1;
}
if (data_buf1.cnTCL!=data_buf1.caTCL) {
        if (necesitacom==1)
                strcat(string_to_send2,"&");
        sprintf(paso,"TCL-%d",data_buf1.cnTCL);
        strcat(string_to_send2,paso);
        necesitacom=1;
}
if (data_buf1.cnICS!=data_buf1.caICS) {
        if (necesitacom==1)
                strcat(string_to_send2,"&");
        sprintf(paso,"ICS-%d",data_buf1.cnICS);
        strcat(string_to_send2,paso);
        necesitacom=1;
}
if (data_buf1.cnBOC!=data_buf1.caBOC) {
        if (necesitacom==1)
                strcat(string_to_send2,"&");
        sprintf(paso,"BOC-%d",data_buf1.cnBOC);
        strcat(string_to_send2,paso);
        necesitacom=1;
}
if (data_buf1.cnTBO!=data_buf1.caTBO) {
        if (necesitacom==1)
                strcat(string_to_send2,"&");
        sprintf(paso,"TBO-%d",data_buf1.cnTBO);
        strcat(string_to_send2,paso);
        necesitacom=1;
}
if (string_no_rec[0]!=NULL && autoborr==1) {
        if (necesitacom==1)
                strcat(string_to_send2,"&");
        strcat(string_to_send2,string_no_rec);
        necesitacom=1;
}
if (necesitacom==1 || catadic==1) {
        data_buf1.catnok=1;
        if (autocorrect==1) {
                gotoxy(33,24);
                cprintf("Cambian.Cat.:%8ld",telprobar);
                if (sistema!=SISTAOM)
                        send_tomacentral();
```

```
                        strcat(string_to_send2,";");
                        sprintf(string_to_send,"%s",string_to_send2);
                        check_rx_buf();
                        strcpy(string_to_send,";");
                        check_rx_buf();
                        editz_resp2(string_to_send2);
                        disp_z_resp();
                        sed_Repaint(sTRADUC);
                        if (autoimpr)
                                impfalla(30);
                        send_dejacentral();
                }
        }
}
if (sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM) {
        data_buf1.catnok=0; // Linea a prueba
        if(sistema!=SIST12PAM)
                sprintf(string_to_send2,"87:1=K'%ld",telprobar);
        else
                sprintf(string_to_send2,"4294:1=K'%ld",telprobar);
        if (data_buf1.cnBADP!=data_buf1.caBADP) {
                if (sistema==SIST12PAM)
                        sprintf(paso,",INTCP=%d",data_buf1.cnBADP);
                else
                        sprintf(paso,",BADP=%d",data_buf1.cnBADP);
                strcat(string_to_send2,paso);
                necesitacom=1;
        }
        if (data_buf1.cnORPT!=data_buf1.caORPT) {
                if (sistema==SIST12PAM) {
                        if (data_buf1.cnORPT!=120) {
                                s_OCB(data_buf1.cnORPT,cadpaso1);
                                sprintf(paso,",OCB=%s",cadpaso1);
                        } else
                                sprintf(paso,"");
                } else
                        sprintf(paso,",ORPT=%d",data_buf1.cnORPT);
                strcat(string_to_send2,paso);
                necesitacom=1;
        }
        if (data_buf1.cnLNCHAR!=data_buf1.caLNCHAR) {
                if (sistema==SIST12PAM) {
                        s_LINECHAR(data_buf1.cnLNCHAR,cadpaso1);
                        sprintf(paso,",LINECHAR=%s",cadpaso1);
                        }
                else
                        sprintf(paso,",LNCHAR=%d",data_buf1.cnLNCHAR);
                strcat(string_to_send2,paso);
                necesitacom=1;
        }
        if (data_buf1.cnTRSTR!=data_buf1.caTRSTR) {
                if (sistema==SIST12PAM) {
                        s_ICB(data_buf1.cnTRSTR,cadpaso1);
                        sprintf(paso,",ICB=%s",cadpaso1);
```

```
                }
        else
                sprintf(paso,",TRSTR=%d",data_buf1.cnTRSTR);
        strcat(string_to_send2,paso);
        necesitacom=1;
}
if (data_buf1.cnSSIG!=data_buf1.caSSIG) {
        if (sistema==SIST12PAM)
                sprintf(paso,",SUBSIG=%d",data_buf1.cnSSIG);
        else
                sprintf(paso,",SSIG=%d",data_buf1.cnSSIG);
        strcat(string_to_send2,paso);
        necesitacom=1;
}
if (data_buf1.cnCBOX!=data_buf1.caCBOX) {
        if (sistema==SIST12PAM) {
                s_PAYPHONE(data_buf1.cnCBOX,cadpaso1);
                sprintf(paso,",PAYPHONE=%s",cadpaso1);
                }
        else {
                s_CBOX(data_buf1.cnCBOX,cadpaso1);
                sprintf(paso,",CBOX=%s",cadpaso1);
        }
        strcat(string_to_send2,paso);
        necesitacom=1;
}
if (data_buf1.cnHM!=data_buf1.caHM) {
        if (sistema!=SIST12PAM) {
                sprintf(paso,",HM=%d",data_buf1.cnHM);
                strcat(string_to_send2,paso);
                necesitacom=1;
        }
}
if (necesitacom==1 || catadic==1) {
        data_buf1.catuok=1;

        if (autocorrect==1) {
                quita_adic();
                if (fin_proc==0) {
                        gotoxy(33,24);
                        cprintf("Caubian.Cat.:%8ld",telprobar);
                        if (sistema!=SIST12PAM)

sprintf(cadenacomando,"%s,HM=%d,TRSTR=%d;",string_to_send2,data_buf1.cnHM,data_buf1.cn
TRSTR);
                        else {
                                s_PAYPHONE(data_buf1.cnCBOX,cadpaso1);
                                s_ICB(data_buf1.cnTRSTR,cadpaso2);

sprintf(cadenacomando,"%s,PAYPHONE=%s,ICB=%s;",string_to_send2,cadpaso1,cadpaso2);
                        }
                        corrige_cat(cadenacomando);
                }
```

```
                        if (autoimpr)
                                impfalla(30);

                        if (getequdb(2)==0)
                                copymemdb(2);
                        else
                                insertdb(2);

                }
        } else {
                if (seekdb(2)==0 && data_buf1.sinmov==0)  // Si no hubo cambios de categorias,
sacarlo de la base de datos
                        deletedb(2);
        }
    }
}

void slomi()
{
        int resultado;
        int necesitacom=0;

        if (data_buf1.contadoru==data_buf1.contadorp)
                        necesitacom=1;
        if (necesitacom==1)     {
                data_buf1.sinmov=1;
                if (autoslomi==1) {
                        resultado=testpruelin(NOTOMACENT);
                        data_buf1.status=reconocepruc('0');
                        sprintf(data_buf1.statdesc,"%s",msg);
                        editz_resp();
                        disp_z_resp();
                        scd_Repaint(sTRADUC);
                        if (resultado==1)
                                endpruelin();
                }
        }
}

void insconthist() {
        sprintf(data_buf3.telhis,"%s",data_buf1.telefono);
        adjustnull(data_buf3.telhis,sizeof(data_buf3.telhis));
        dateconthist();
        insertdb(3);
}

void disp_z_resp() {
        char z_resp2[S_RESP+50];
        int i,j;

        window(3,8,77,20);
        clrscr();
        gotoxy(1,1);
        textcolor(BLUE);
```

```
        textbackground(CYAN);
        // Modifica el cr en la respuesta
        j=0;
        for (i=0;i<S_RESP;i++) {
                if (z_resp[i]==0x0A) {
                        z_resp2[j]=0x0D;
                        j++;
                        z_resp2[j]=0x0A;
                        j++;
                }       else {
                        z_resp2[j]=z_resp[i];
                        j++;
                }
        }
        for (i=0;i<S_RESP;i++)
                z_resp[i]=z_resp2[i];
        cputs(z_resp);
        window(1,1,80,24);
}

void revcatcgorias() {
        int respuesta;
        int resultado;
        int rc;
        int pruebaok;
        int contpruebas;
        int limloop;
        char loginant[10];

        if (strcmp(ultimotel,"0")==0) {
                rc=dtop(1);
                }
        else {
                sprintf(data_buf1.telefono,"%s",ultimotel);
                rc=dseek(1);
        }
        adjustnull(ulttelprobar,sizeof(ulttelprobar));
        copydbmem(1);
        adjustnull(data_buf1.telefono,sizeof(data_buf1.telefono));
        sscanf(data_buf1.telefono,"%ld",&telprobar);
        sprintf(ultimotel,"%s",data_buf1.telefono);
        respuesta=isvalid();
        while (rc==success() && fin_proc==0) {
                if (respuesta==2) {
                        if (strcmp(loginfijo,login)==0 || strcmp(loginfijo,"TODAS")==0) {
                                if (!(sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM))
{
                                        resultado=send_tomacentral();
                                        if (resultado==1)
                                                send_incentral();
                                }
                        }
                        memcpy(loginant,login,10);
                        if (reconoceprue(1)==83) { // Si se encuentra fuera la central
```

```
                        while (rc==success() && strcmp(loginant,login)==0 && fin_proc==0) {
                                if (data_buf1.sincontrol==0 && (strcmp(login,loginfijo)==0 ||
strcmp(loginfijo,"TODAS")==0)
                                        && strcmp(data_buf1.telefono,ulttelprobar)<=0) {
                                        gotoxy(33,24);
                                        cprintf("Leyendo Cat.:%8ld",telprobar);
                                        gotoxy(57,24);
                                        cprintf("Cat. leida :%8ld",telprobar);
                                        if (reconocepruc(1)==83 && sistema==SISTAOM) {
                                                sprintf(string_to_send,";");
                                                check_rx_buf();
                                                send_incentral();
                                                sound(700);
                                                delay(50);
                                                nosound();
                                        }
                                }
                                memcpy(loginant,login,10);
                                rc=dskip(1);
                                copydbmem(1);
                                sscanf(data_buf1.telefono,"%ld",&telprobar);
                                sprintf(ultimotel,"%s",data_buf1.telefono);
                                respuesta=isvalid();
                        }
                } else {
                        while (rc==success() && strcmp(loginant,login)==0 && fin_proc==0 ) {
                                if (data_buf1.sincontrol==0 && (strcmp(login,loginfijo)==0 ||
strcmp(loginfijo,"TODAS")==0)
                                        && strcmp(data_buf1.telefono,ulttelprobar)<=0) {
                                        gotoxy(33,24);
                                        cprintf("Leyendo Cat.:%8ld",telprobar);
                                        contpruebas=0;
                                        pruebaok=0;
                                        data_buf1.catnok=0;
                                        datetimecat();
                                        if
(sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM)
                                                limloop=NOP_S12;
                                        else
                                                limloop=NOP_AXE;
                                        do {
                                                if (!(resultado>=100 && resultado<=109)) {

        resultado=testconscat(NOTOMACENT);
                                                }
                                                if (resultado>=100 && resultado<=109)
                                                        interpretaecate(resultado);
                                                else
                                                        pruebaok=reconocecate(1);
                                                editz_resp();
                                                disp_z_resp();
                                                sed_Repaint(sTRADUC);
```

```
                                        if (pruebaok==0 &&
!(sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM)) // Lineas a prueba ORIGINAL
sistema!=SIST12
                                                break;                  // Lineas a
prueba
                                        if (resultado==1)
                                                endconscat();
                                        if (reconocepru(1)==83 &&
sistema==SISTAOM) {
                                                sprintf(string_to_send,";");
                                                check_rx_buf();
                                                send_incentral();
                                                sound(700);
                                                delay(50);
                                                nosound();
                                        }
                                        if (data_buf1.caBADP==-1)
                                                data_buf1.catnok=1;
                                        if (data_buf1.caBADP==-2)
                                                data_buf1.catnok=1;
                                        if (pruebaok==1 && resultado==1 &&
!(data_buf1.caBADP==-1||data_buf1.caBADP==-2)) {
                                                correct();
                                        }
                                        contpruebas++;
                                        fl_press();
                                } while (!(contpruebas>limloop || pruebaok==1 ||
(pruebaok>=90 && pruebaok<=92) || fin_proc==1));
                                sprintf(data_buf1.telefono,"%ld",telprobar);

        adjustnull(data_buf1.telefono,sizeof(data_buf1.telefono));
                                data_buf1.nopruebas+=1;
                                copymemdb(1);
        // Actualiza base de datos
                                gotoxy(57,24);
                                cprintf("Cat. leida :%8ld",telprobar);
                        }
                        respuesta=isvalid();
                        memcpy(loginant.login,10);
                        rc=dskip(1);
                        copydbmem(1);
                        adjustnull(data_buf1.telefono,sizeof(data_buf1.telefono));
                        sscanf(data_buf1.telefono,"%ld",&telprobar);
                        sprintf(ultimotel,"%s",data_buf1.telefono);
                        respuesta=isvalid();
                        if (strcmp(data_buf1.telefono,ulttelprobar)>0) {
                                goto fccatlabel;
                                }
                        }
                }
        } else {
                data_buf1.sinnov=0;
                data_buf1.catnok=0;
                rc=dskip(1);
```

```
                        copydbmem(1);
                        sscanf(data_buf1.telefono,"%ld",&telprobar);
                        sprintf(ultimotel,"%s",data_buf1.telefono);
                        respuesta=isvalid();
                }
        }
        if (resultado==100)
                sprintf(telmodemant[puerto-1],"-1");

        fccatlabel:
        if (fin_proc==0)   // Si no se salio por teclear F1
                strcpy(ultimotel,"0"); // Linea de prueba
}

void revcontadores() {
        char loginant[10];
        int rc;
        int contpruebas;
        int limloop;
        int respuesta,resultado;
        int pruebaok;

        if (strcmp(ultimotel,"0")==0) {
                rc=dtop(1);
                }
        else {
                sprintf(data_buf1.telefono,"%s",ultimotel);
                rc=dseek(1);
        }
        adjustnull(ulttelprobar,sizeof(ulttelprobar));
        copydbmem(1);
        adjustnull(data_buf1.telefono,sizeof(data_buf1.telefono));
        sscanf(data_buf1.telefono,"%ld",&telprobar);
        sprintf(ultimotel,"%s",data_buf1.telefono);
        respuesta=isvalid();
        while (rc==success() && fin_proc==0) {
                if (respuesta==2) {
                        if (strcmp(loginfijo,login)==0 || strcmp(loginfijo,"TODAS")==0) {
                                if (!(sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM))

                                        resultado=send_tomacentral();
                                        if (resultado==1)
                                                send_incentral();
                                }
                        }
                        memcpy(loginant,login,10);
                        if (reconocepruc(1)==83) {
                                while (rc==success() && strcmp(loginant,login)==0 && fin_proc==0 ) {
                                        if (data_buf1.sincontrol==0 && data_buf1.suspendido==0 &&
                                                        (strcmp(login,loginfijo)==0 ||
strcmp(loginfijo,"TODAS")==0)
                                                                        &&
strcmp(data_buf1.telefono,ulttelprobar)<=0) {
                                                gotoxy(33,24);
```

```
                                        cprintf("Leyendo Cont:%8ld",telprobar);
                                        gotoxy(57,24);
                                        cprintf("Cont leido :%8ld",telprobar);
                                        if (reconocepruc(1)==83 && sistema==SISTAOM) {
                                                sprintf(string_to_send,";");
                                                check_rx_buf();
                                                send_incentral();
                                                sound(700);
                                                delay(50);
                                                nosound();
                                        }
                                }
                                memcpy(loginant,login,10);
                                rc=dskip(1);
                                copydbmem(1);
                                sscanf(data_buf1.telefono,"%ld",&telprobar);
                                sprintf(ultimotel,"%s",data_buf1.telefono);
                                respuesta=isvalid();

                        }
                } else {
                        while (rc==success() && strcmp(loginant,login)==0 && fin_proc==0 ) {
                                if (data_buf1.sincontrol==0 && data_buf1.suspendido==0 &&
                                                                        (strcmp(login,loginfijo)==0 ||
strcmp(loginfijo,"TODAS")==0)
                                                                        &&
strcmp(data_buf1.telefono,ulttelprobar)<=0) {
                                        gotoxy(33,24);
                                        cprintf("Leyendo Cont:%8ld",telprobar);
                                        contpruebas=0;
                                        data_buf1.status=0;
                                        data_buf1.sinmov=0;
                                        datetimecont(); // Actualiza contadorp y contadoru
                                        pruebaok=0;
                                        if
(sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM)
                                                limloop=NOP_S12;
                                        else
                                                limloop=NOP_AXE;
                                        do {
                                                if (!(resultado>=100) && resultado<=109)) {

        resultado=testconscont(NOTOMACENT);
                                                }
                                                if (resultado>=100 && resultado<=109)
                                                        interpretacont(resultado);
                                                else
                                                        pruebaok =reconocecont();
                                                editz_resp();
                                                disp_z_resp();
                                                if (resultado==1)
                                                        endconscont();
                                                sed_Repaint(sTRADUC);
```

```
                                        if (reconocepnie(1)==83 &&
sistema==SISTAOM) {
                                                sprintf(string_to_send,";");
                                                check_rx_buf();
                                                send_incentral();
                                                sound(700);
                                                delay(50);
                                                nosound();
                                        }
                                        if (resultado==1) {
                                                if (pruebaok==1 ) {  // Incluyo
resultado==1
                                                        gotoxy(33,24);
                                                        cprintf("Prueba
Linea:%8ld",telprobar);
                                                        inscontlist();
                                                        slomi();
                                                } else {
                                                        if (pruebaok==90 ||
pruebaok==91) {
                                                                if (pruebaok==91) {
        data_buf1.sinmov=1;
        data_buf1.contadoru=data_buf1.contadorp=-1;
                                                                }
        data_buf1.status=pruebaok;
        sprintf(data_buf1.statdesc,"%s",msg);
                                                                if (dseek(1)==0)
        copymemdb(1);
                                                                else
                                                                        insertdb(1);
                                                        }
                                                        if (pruebaok==0 &&
!(sistema==SIST12||sistema==SIST12MTM||sistema==SIST12PAM)) // Lineas a prueba
                                                                break;
// Lineas a prueba
                                                }
                                        }
                                        contpruebas++;
                                        fl_press();
                                } while (!(contpruebas>limloop || pruebaok==1 ||
(pruebaok==0 && resultado!=112) || fin_proc!=0));
                                sprintf(data_buf1.telefono,"%ld",telprobar);
        adjustnull(data_buf1.telefono,sizeof(data_buf1.telefono));
                                data_buf1.nopruebas+=1;
                                copymemdb(1); // actualizando base de datos
                                gotoxy(57,24);
                                cprintf("Linea Prob. :%8ld",telprobar);
                        }
```

```
                                        respuesta=isvalid();
                                        memcpy(loginant,login,10);
                                        rc=dskip(1);
                                        copydbmem(1);
                                        adjustnull(data_buf1.telefono,sizeof(data_buf1.telefono));
                                        sscanf(data_buf1.telefono,"%ld",&telprobar);
                                //      printf("Una vuelta %ls ",data_buf1.telefono);
                                //      getch();
                                        sprintf(ultimotel,"%s",data_buf1.telefono);
                                        respuesta=isvalid();
                                        if (strcmp(data_buf1.telefono,ulttelprobar)>0) {
                                                goto fcontlabel;
                                        }
                                }
                        }
                } else {
                        data_buf1.sinmov=0;
                        data_buf1.catnok=0;
                        rc=dskip(1);
                        copydbmem(1);
                        sscanf(data_buf1.telefono,"%ld",&telprobar);
                        sprintf(ultimotel,"%s",data_buf1.telefono);
                        respuesta=isvalid();
                }
        }
        if (resultado==100)
                sprintf(telmodemant[puerto-1],"-1");
        fcontlabel:
        if (fin_proc==0)   // Si no se salio por teclear F1
                strcpy(ultimotel,"0"); // Linea de prueba
}

void vcategorias() {
        boolean flagprimer;
        boolean flagreelab;

        iBUFFERF(flagreelab,flagprimer);
        if (ult_tecla==ESC)
                return;

        if (contvcat>totvuecont)
                contvcat=1;
        if (flagreelab) {
                dzap(2);
//              strcpy(ultimotel,"0");
        }
        if (strcmp(ultimotel,"0")==0) {
                contvcat=1;
                strcpy(ultimotel,"0");
        }
        dtag_select(1,1);
        iPROC(1);
        iAOM();
        iTRADUC();
```

```
        fin_proc=0;
        while(contvcat<=totvuecat && fin_proc==0) {
                revcategorias();
                if (fin_proc==0) {
                        strcpy(ultimotel,"0");
                        reindexdb(1);
                        dtop(1);
                        contvcat++;
                }
        }
        sed_Close(sAOM);
        sed_Close(sTRADUC);
        sed_Close(sPROC);
        dispMsg("                                                ",IZQUIERDA);
}

void vcontadores() {
        boolean flagprimer;
        boolean flagreclab;

        iBUFFERF(flagreclab,flagprimer);
        if (ult_tecla==ESC)
                return;
        if (contvcont>totvuecont)
                contvcont=1;
        if (flagreclab) {
                dzap(2);
        }
        if (strcmp(ultimotel,"0")==0) {
                contvcont=1;
                strcpy(ultimotel,"0");
        }
        dtag_select(1,1);
        iPROC(2);
        iAOM();
        iTRADUC();
        fin_proc=0;
        while (contvcont<=totvuecont && fin_proc==0) {
                revcontadores();
                if (fin_proc==0) {
                        strcpy(ultimotel,"0");
                        reindexdb(1);
                        dtop(1);
                        contvcont++;
                }
        }
        sed_Close(sAOM);
        sed_Close(sTRADUC);
        sed_Close(sPROC);
        dispMsg("                                                ",IZQUIERDA);
}

void valcancias() {
        boolean flagprimer;
```

```
boolean flagreclab;

iBUFFERF(flagreclab,flagprimer);
if (ult_tecla==ESC)
        return;
if (flagreclab) {
        ciclo=1;
}
if (flagprimer==1) {
        ciclo=1;
        strcpy(ultimotel,"0");
}
dtag_select(1,1);
iPROC(3);
iAOM();
iTRADUC();
fin_proc=0;
while (fin_proc==0) {
        if (ciclo==1) {
         revcategorias();
        }
        if (fin_proc==0 && activa()) {
                if (strcmp(ultimotel,"0")==0)
                        dtop(1);
                ciclo=2;
        }
        if (ciclo==2) {
                revcontadores();
                if (fin_proc==0)
                        actactiva();
        }
        if (fin_proc==0) {
                ciclo=1;
                strcpy(ultimotel,"0");
                reindexdb(1);
                dtop(1);
        }
}
sed_Close(sAOM);
sed_Close(sTRADUC);
sed_Close(sPROC);
dispMsg("                                                ",IZQUIERDA);
```

```
// ********************************************************************************
// SISTEMA: ALCANCIAS 3.4 (AXE Y SISTEMA 12)
// ARCHIVO: BASES.H
// DESCRIPCION: TAMAÑOS DE LOS CAMPOS DE LA BASE DE DATOS
// FECHA:  06/10/96
// ********************************************************************************
// --------------------------------------------------------------------------
// Descripción de contantes de la base de datos ALCANCIAS.DBF (Versión 3.4)
// --------------------------------------------------------------------------
#define S_LOGIN        10
#define S_LADA          4
#define S_MSG          80
#define S_PSW          10
#define S_FILE         15
#define S_DESDISTRITO  40
#define S_SEGUR                        36
#define S_PSW          10
#define S_APARATO      15
// --------------------------------------------------------------------------
// Descripción de contantes de la base de datos TELPUB.DBF
// --------------------------------------------------------------------------
#define S_NUM_TEL       8  // Tamaño del campo del Número de Teléfono
#define S_CLAVE_POB     7  // Tamaño del campo de la Clave de la población
#define S_ALARMA        7
#define S_GRUPO         5  // Tamaño del campo del Grupo
#define S_TIPO_APTO     2  // Tamaño del campo del Tipo de aparato
#define S_TIPOAMB       2  // Tamaño del campo del tipo de ambiente
#define S_LUGAR_INS     1  // Tamaño del campo del lugar de Instalación
#define S_FECHA_INS     8  // Tamaño del campo de Fecha de Instalación
#define S_CALLEYNUM     50  // Tamaño del campo de la calle y número
#define S_COLONIA       50  // Tamaño del campo del la colonia
#define S_C_P           5  // Tamaño del campo del Código Postal
#define S_DISTRITO      6  // Tamaño del campo del Distrito
#define S_PRINCIPAL     6  // Tamaño del campo del Principal
#define S_SECUNDARIO    5  // Tamaño del campo del secundario
#define S_ZONA          1  // Tamaño del campo de la Zona
// --------------------------------------------------------------------------
// Descripción de las contantes de la base de datos MONITOR.DBF
// --------------------------------------------------------------------------
                                                                  // Se usa
S_NUM_TEL de TELBUB.DBF
#define S_TSERVICIO     2  // Tamaño del Tipo de apto
#define S_IDCENTRAL     3  // Tamaño del Identificador de la tabla de centrales
#define S_CNCA          3  // Tamaño para los registos de todas lass
                                                                  // y categorias
necesarias y actuales con error
#define S_SINMOV        1  // Tamaño de la bandera de sin movimiento de cont.
#define S_CATNOK        1  // Tamaño de la bandera de cambio de categoria
#define S_STATCATCNT    3  // Tamaño del digito de falla de categoria y cont
#define S_SUSPENDIDO    1  // Tamaño de la bandera de suspensión
#define S_SINCONTROL    1  // Tamaño de la bandera sin control
#define S_CONTADOR      10  // Tamaño del registro de lectura de contadores
#define S_HORA          5  // Tamaño de la hora de lectura
#define S_FECHA         11  // Tamaño de la fecha de lectura
```

```
#define S_SEGURIDAD     36  // Tamaño del registo de seguridad

// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos CENTRAL.DBF
// ---------------------------------------------------------------------
#define S_LINF          8   // Tamaño del Límite inferior del rango
#define S_LSUP          8   // Tamaño del Límite superior del rango
#define S_TCENTRAL      1   // Tamaño del tipo de central
#define S_PASS          10  // Tamaño del Password de la central
#define S_LOGIN         10  // Tamaño del login de la central
#define S_CFGPUERTO     12  // Tamaño del nombre del archivo de conf. de puerto
#define S_MODEM         10  // Tamaño maximo del número del modem a marcar
#define S_ANALDIG       1   // Tamaño del descriptor analógico ó digital
#define S_DIGITOS       1   // Tamaño del descriptor analógico ó digital
// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos STATUSCNT.DBF
// ---------------------------------------------------------------------
                                                        // Se usa
S_STATCATCNT de MONITOR.DBF
#define S_STATDESC      60  // Descripción del status
// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos STATUSCAT.DBF
// ---------------------------------------------------------------------
                                                        // Se usa
S_STATCATCNT de MONITOR.DBF
                                                        // Se usa
S_STATDESC  de STATUSCNT.DBF
// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos RESPONSA.DBF
// ---------------------------------------------------------------------
                                                        // Se usa S_ZONA
de MONITOR.DBF
#define S_RESPONSABLE   40  // Tamaño del Nombre del reponsable de zona
// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos SINMOV.DBF
// ---------------------------------------------------------------------
#define S_CVE_REP       17  // Tamaño de la clave del reporte
// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos DESCOMP.DBF
// ---------------------------------------------------------------------
#define S_TIPO_DESC     6   // Tamaño de tipo de descompostura
#define S_USUARIO       40  // Tamaño de Descripción del Usuario
// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos arreglo.DBF
// ---------------------------------------------------------------------
#define S_TIPO_ARRE     6   // Tamaño del Tipo de arreglo
#define S_REPARADOR     10  // Tamaño de Clave del Reparador
#define S_TFS           3   // Tamaño máximo de tiempo fuera de servicio


typedef struct
{
                long linf;
                long lsup;
```

```
        int tipo;
        int digitos;
        int sistema;
        char password[S_PSW];
        char login[S_LOGIN];
        char puertocfg[S_FILE];
        char telmodem[S_NUM_TEL+1];
} RANGOSTR;

typedef struct
{
        int iCNCBA;
        int iCNTLI;
        int iCNBIC;
        int iCNSSI;
        int iCNOBA;
        int iCNTCL;
        int iCNICS;
        int iCNBOC;
        int iCNTBO;
        int iCNBADP;   // PAM es INTCP
        int iCNCBOX;   // PAM es PAYPHONE
        int iCNORPT;   // PAM es OCB
        int iCNSSIG;   // PAM es SUBSIG
        int iCNTRSTR;  // PAM es ICB
        int iCNLNCHAR; // PAM es LINECHAR
        int iCNHM;     // PAM incluido en PAYPHONE
        int iCNSUBGRP; // PAM ei SUBGRP
        int iCNINTCP;
        int iCNPAYPHONE;
        int iCNOCB;
        int iCNSUBSIG;
        int iCNICB;
        int iCNLINECHAR;
} CATTA;


/* ------------------------------------------------ */
/* Estructura de la base de datos de Alcancias     */
/* ------------------------------------------------ */
typedef struct
{
        char sNum_Tel[S_NUM_TEL+1];      // Número telefónico con todo y lada
        char sClave_Pob[S_CLAVE_POB+1];  // Clave de la población
        char sAlarma[S_ALARMA+1];        // Alarma del teléfono público
        char sGrupo[S_GRUPO+1];          // Grupo  del teléfono público
        char sTipo_Apto[S_TIPO_APTO+1];  // Tipo de aparato (marca del aparato)
        char sTipoAmb[S_TIPOAMB+1];      // Tipo de ambiente en el que esta instalado
        char sLugar_Ins[S_LUGAR_INS+1];  // Tipo de Lugar de instalación
        char sFecha_Ins[S_FECHA_INS+1];  // Fecha de instalación
        char sCalleyNum[S_CALLEYNUM+1];  // Calle y número del lugar de la instalación
        char sColonia[S_COLONIA+1];      // Colonia
        char sC_P[S_C_P+1];              // Codigo postal
        char sDistrito[S_DISTRITO+1];    // Dato técnico del Distrito
```

```
char sPrincipal[S_PRINCIPAL+1];  // Dato técnico del Principal
char sSecundario[S_SECUNDARIO+1]; // Dato técnico del secundario

char sZona[S_ZONA+1];          // Zona donde se encuentra instalado
int iSuspendido; // Dato técnico del secundario
int iSinControl; // Zona donde se encuentra instalado
int iTServicio;

int iCNCBA;
int iCNTLI;
int iCNBIC;
int iCNSSI;
int iCNOBA;
int iCNTCL;
int iCNICS;
int iCNBOC;
int iCNTBO;
int iCNBADP;
int iCNCBOX;
int iCNORPT;
int iCNSSIG;
int iCNTRSTR;
int iCNLNCHAR;
int iCNHM;
int iCNSUBGRP;
int iCACBA;
int iCATLI;
int iCABIC;
int iCASSI;
int iCAOBA;
int iCATCL;
int iCAICS;
int iCABOC;
int iCATBO;
int iCABADP;
int iCACBOX;
int iCAORPT;
int iCASSIG;
int iCATRSTR;
int iCALNCHAR;
int iCAHM;
int iCASUBGRP;
int iStatus;
int iSistTel;
int iSinMov;
int iCatNok;
long lContadorU;
long lContadorP;
char sHoraCatU[S_HORA+1];
char sHoraCatP[S_HORA+1];
char sHoraContU[S_HORA+1];
char sHoraContP[S_HORA+1];
char sFechaCatU[S_FECHA+1];
char sFechaCatP[S_FECHA+1];
```

```
        char sFechaContU[S_FECHA+1];
        char sFechaContP[S_FECHA+1];
        char sSeguridad[S_SEGUR+1];
        char sStatDesc[S_STATDESC+1];
}.ALCANBTRV;

typedef struct {
        char sTelHis[S_NUM_TEL+1];
        char sHoraHis[S_HORA+1];
        char sFechaHis[S_FECHA+1];
        long lContHis;
}CONTBTRV;
```

```
// ****************************************************************************
// SISTEMA: ALCANCIAS 3.4 (AXE, SISTEMA 12)
//
// ARCHIVO: COMM.H
//
// FECHA: 06/10/96
// ****************************************************************************
/*

ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄ¿
 ³    DEFINICIONES                    ³

ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÙ
*/
#define vdu       0x9A /* 1200,even parity,1 stopbits,7 bits */
#define VDU       0x9B /* 1200,even parity,1 stopbit ,8 bits */
#define itf       0xE3 /*                                    */
#define int_com2  0x0B /* interrupt of comport 1      */
#define break_int 0x05 /* interrupt of print screen   */
/*------ COM PORT 1 -----------------------------------------------*/
#define dll       0x2F8 /* divisor latch LSB          */
#define rxb       0x2F8 /* receive buffer             */
#define txb       0x2F8 /* transmit buffer            */
#define dlm       0x2F9 /* divisor latch MSB          */
#define ier       0x2F9 /* interrupt enable register  */
#define iir       0x2FA /* interrupt identification register */
#define lcr       0x2FB /* line control register      */
#define mcr       0x2FC /* modem control register     */
#define lsr       0x2FD /* line status register       */
#define msr       0x2FE /* modem status register      */
/*------ IER ( INTERRUPT ENABLE REGISTER ) -------------------------*/
#define thre_off  0x0D
#define thre_on   0x0F
#define ier_ena   0x0D
#define ier_dis   0x00
#define mcr_setup 0x0F /* bit 0,1,2,3=1               */
#define mcr_reset 0x00
/* ----- PIC ( PROGRAMABLE INTERRUPT CONTROLER ) -------------------*/
#define pic_irq_reg 0x21 /* Values for the 8259 Interrupt  */
#define pic_eoi_reg 0x20 /* controller                     */
#define end_of_int  0x20
#define irq3_ena_mask 0xF7
#define irq3_dis_mask 0x08
/*------ ASCII ----------------------------------------------------*/
#define nll       0x00
#define stx       0x02
#define eotx      0x03
#define eot       0x04
#define enq       0x05
#define ack       0x06
#define bel       0x07
#define bs        0x08
```

```
#define  lf        0x0A
#define  cr        0x0D
#define  ff        0x0C
#define  dc1       0x11 /* xon                    */
#define  dc2       0x12 /*                        */
#define  dc3       0x13 /* xoff                   */
#define  xon       0x11
#define  xoff      0x13
#define  etb       0x17
#define  cob       0x18 // ???
#define  esc       0x1B
#define  sp        0x20 // Espacio en blanco
#define  dp        0x3A // Simbolo :
#define  sc        0x3B // Simbolo ;
#define  ls        0x3C // Simbolo <
#define  ms        0x3E // Simbolo >
#define  ctrl_c                          0x2E03
#define  ctrl_d    0x2004
#define  ctrl_e                          0x1205
#define  ctrl_x    0x2D18
#define  ESC                               0x011B
#define  F1                                0x3B00
#define  F2                                0x3C00
#define  LEFT                 ((int) 0x4B00)
#define  RIGHT                ((int) 0x4D00)
#define  FALSE     0
#define  TRUE      1
#define  P8N1 1
#define  P7E1 0

/* configuration file record structure */
typedef struct {
        int port;
        long baud_rate;
        int parity;
        int lock_port;
        char initialization_string[81];
        char hangup_string[41];
        char dial_prefix[41];
} CONFIG;

int init(void);
void printcontrol(int);
void openpuertos(void);
void cierrapuertos(void);
void cuelgapuertos(void);
void clear_buffers(void);
void capture(void);
void contact(void);
void restore(void);
void send_status(void);
void send_ack(void);
void send_enter(void);
void print_help(void);
```

```
void detect_string(void);
void close_file(void);
void send_string(void);
void send_string_prev(void);
void test(void);
void interrupt far inter_hndlr(void);
void interrupt far ctrl_break(void);
int look(unsigned char);
int marca_modem(void);
void config_program(void);
int dial(void);
void exit_program(void);
char *get_modem_string(char *s, char *d);
void hangup(void);
void modem_control_string(char *s);
void read_config(void);
void save_config(void);
void vacia_buffer();
int check_rx_buf();
int f1_press();
```

```
// ********************************************************************************
// SISTEMA: ALCANCIAS 3.4 (AXE Y SISTEMA 12)
// ARCHIVO: BASES.H
// DESCRIPCION: TAMAÑOS DE LOS CAMPOS DE LA BASE DE DATOS
// FECHA:  06/10/96
// ********************************************************************************
// ------------------------------------------------------------------------
// Descripción de contantes de la base de datos ALCANCIAS.DBF (Versión 3.4)
// ------------------------------------------------------------------------
#define S_LOGIN        10
#define S_LADA          4
#define S_MSG          80
#define S_PSW          10
#define S_FILE         15
#define S_DESDISTRITO   40
#define S_SEGUR                        36
#define S_PSW          10
#define S_APARATO      15
// ------------------------------------------------------------------------
// Descripción de contantes de la base de datos TELPUB.DBF
// ------------------------------------------------------------------------
#define S_NUM_TEL       8  // Tamaño del campo del Número de Teléfono
#define S_CLAVE_POB     7  // Tamaño del campo de la Clave de la población
#define S_ALARMA        7
#define S_GRUPO         5  // Tamaño del campo del Grupo
#define S_TIPO_APTO     2  // Tamaño del campo del Tipo de aparato
#define S_TIPOAMB       2  // Tamaño del campo del tipo de ambiente
#define S_LUGAR_INS     1  // Tamaño del campo del lugar de Instalación
#define S_FECHA_INS     8  // Tamaño del campo de Fecha de Instalación
#define S_CALLEYNUM     50  // Tamaño del campo de la calle y número
#define S_COLONIA       50  // Tamaño del campo del la colonia
#define S_C_P           5  // Tamaño del campo del Código Postal
#define S_DISTRITO      6  // Tamaño del campo del Distrito
#define S_PRINCIPAL     6  // Tamaño del campo del Principal
#define S_SECUNDARIO    5  // Tamaño del campo del secundario
#define S_ZONA          1  // Tamaño del campo de la Zona
// ------------------------------------------------------------------------
// Descripción de las contantes de la base de datos MONITOR.DBF
// ------------------------------------------------------------------------
                                                                  // Se usa
S_NUM_TEL de TELBUB.DBF
#define S_TSERVICIO     2  // Tamaño del Tipo de apto
#define S_IDCENTRAL     3  // Tamaño del Identificador de la tabla de centrales
#define S_CNCA          3  // Tamaño para los registos de todas lass
                                                                  // y categorias
necesarias y actuales con error
#define S_SINMOV        1  // Tamaño de la bandera de sin movimiento de cont.
#define S_CATNOK        1  // Tamaño de la bandera de cambio de categoria
#define S_STATCATCNT    3  // Tamaño del digito de falla de categoria y cont
#define S_SUSPENDIDO    1  // Tamaño de la bandera de suspensión
#define S_SINCONTROL    1  // Tamaño de la bandera sin control
#define S_CONTADOR      10  // Tamaño del registro de lectura de contadores
#define S_HORA          5  // Tamaño de la hora de lectura
#define S_FECHA         11  // Tamaño de la fecha de lectura
```

```
#define S_SEGURIDAD     36  // Tamaño del registo de seguridad

// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos CENTRAL.DBF
// ---------------------------------------------------------------------
#define S_LINF          8   // Tamaño del Límite inferior del rango
#define S_LSUP          8   // Tamaño del Límite superior del rango
#define S_TCENTRAL      1   // Tamaño del tipo de central
#define S_PASS          10  // Tamaño del Password de la central
#define S_LOGIN         10  // Tamaño del login de la central
#define S_CFGPUERTO     12  // Tamaño del nombre del nrchivo de conf. de puerto
#define S_MODEM         10  // Tamaño maximo del número del modem a marcar
#define S_ANALDIG       1   // Tamaño del descriptor analógico ó digital
#define S_DIGITOS       1   // Tamaño del descriptor analógico ó digital
// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos STATUSCNT.DBF
// ---------------------------------------------------------------------
                                                        // Se usa
S_STATCATCNT de MONITOR.DBF
#define S_STATDESC      60  // Descripción del status
// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos STATUSCAT.DBF
// ---------------------------------------------------------------------
                                                        // Se usa
S_STATCATCNT de MONITOR.DBF
                                                        // Se usa
S_STATDESC  de STATUSCNT.DBF
// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos RESPONSA.DBF
// ---------------------------------------------------------------------
                                                        // Se usa S_ZONA
de MONITOR.DBF
#define S_RESPONSABLE   40  // Tamaño del Nombre del reponsable de zona
// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos SINMOV.DBF
// ---------------------------------------------------------------------
#define S_CVE_REP       17  // Tamaño de la clave del reporte
// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos DESCOMP.DBF
// ---------------------------------------------------------------------
#define S_TIPO_DESC     6   // Tamaño de tipo de descompostura
#define S_USUARIO       40  // Tamaño de Descripción del Usuario
// ---------------------------------------------------------------------
// Descripción de las constantes de la base de datos arreglo.DBF
// ---------------------------------------------------------------------
#define S_TIPO_ARRE     6   // Tamaño del Tipo de arreglo
#define S_REPARADOR     10  // Tamaño de Clave del Reparador
#define S_TFS           3   // Tamaño máximo de tiempo fuera de servicio


typedef struct
{
                long linf;
                long lsup;
```

```
                int tipo;
                int digitos;
                int sistema;
                char password[S_PSW];
                char login[S_LOGIN];
                char puertocfg[S_FILE];
                char telmodem[S_NUM_TEL+1];
} RANGOSTR;

typedef struct
{
                int iCNCBA;
                int iCNTLI;
                int iCNBIC;
                int iCNSSI;
                int iCNOBA;
                int iCNTCL;
                int iCNICS;
                int iCNBOC;
                int iCNTBO;
                int iCNBADP;  // PAM es INTCP
                int iCNCBOX;  // PAM es PAYPHONE
                int iCNORPT;  // PAM es OCB
                int iCNSSIG;  // PAM es SUBSIG
                int iCNTRSTR;  // PAM es ICB
                int iCNLNCHAR; // PAM es LINECHAR
                int iCNHM;    // PAM incluido en PAYPHONE
                int iCNSUBGRP; // PAM ei SUBGRP
                int iCNINTCP;
                int iCNPAYPHONE;
                int iCNOCB;
                int iCNSUBSIG;
                int iCNICB;
                int iCNLINECHAR;
} CATTA;


/* ------------------------------------------------ */
/* Estructura de la base de datos de Alcancias     */
/* ------------------------------------------------ */
typedef struct
{
        char sNum_Tel[S_NUM_TEL+1];     // Número telefónico con todo y lada
        char sClave_Pob[S_CLAVE_POB+1]; // Clave de la población
        char sAlarma[S_ALARMA+1];       // Alarma del teléfono público
        char sGrupo[S_GRUPO+1];         // Grupo  del teléfono público
        char sTipo_Apto[S_TIPO_APTO+1]; // Tipo de aparato [marca del aparato]
        char sTipoAmb[S_TIPOAMB+1];     // Tipo de ambiente en el que esta instalado
        char sLugar_Ins[S_LUGAR_INS+1]; // Tipo de Lugar de instalación
        char sFecha_Ins[S_FECHA_INS+1]; // Fecha de instalación
        char sCalleyNum[S_CALLEYNUM+1]; // Calle y número del lugar de la instalación
        char sColonia[S_COLONIA+1];     // Colonia
        char sC_P[S_C_P+1];             // Codigo postal
        char sDistrito[S_DISTRITO+1];   // Dato técnico del Distrito
```

```
char sPrincipal[S_PRINCIPAL+1];  // Dato técnico del Principal
char sSecundario[S_SECUNDARIO+1]; // Dato técnico del secundario

char sZona[S_ZONA+1];          // Zona donde se encuentra instalado
int  iSuspendido; // Dato técnico del secundario
int  iSinControl; // Zona donde se encuentra instalado
int  iTServicio;

int  iCNCBA;
int  iCNTLI;
int  iCNBIC;
int  iCNSSI;
int  iCNOBA;
int  iCNTCL;
int  iCNICS;
int  iCNBOC;
int  iCNTBO;
int  iCNBADP;
int  iCNCBOX;
int  iCNORPT;
int  iCNSSIG;
int  iCNTRSTR;
int  iCNLNCHAR;
int  iCNHM;
int  iCNSUBGRP;
int  iCACBA;
int  iCATLI;
int  iCABIC;
int  iCASSI;
int  iCAOBA;
int  iCATCL;
int  iCAICS;
int  iCABOC;
int  iCATBO;
int  iCABADP;
int  iCACBOX;
int  iCAORPT;
int  iCASSIG;
int  iCATRSTR;
int  iCALNCHAR;
int  iCAHM;
int  iCASUBGRP;
int  iStatus;
int  iSistTel;
int  iSinMov;
int  iCatNok;
long lContadorU;
long lContadorP;
char sHoraCatU[S_HORA+1];
char sHoraCatP[S_HORA+1];
char sHoraContU[S_HORA+1];
char sHoraContP[S_HORA+1];
char sFechaCatU[S_FECHA+1];
char sFechaCatP[S_FECHA+1];
```

```
        char sFechaContU[S_FECHA+1];
        char sFechaContP[S_FECHA+1];
        char sSeguridad[S_SEGUR+1];
        char sStatDesc[S_STATDESC+1];
} ALCANBTRV;

typedef struct {
        char sTelHis[S_NUM_TEL+1];
        char sHoraHis[S_HORA+1];
        char sFechaHis[S_FECHA+1];
        long lContHis;
}CONTBTRV;
```

```
// ********************************************************************************
// SISTEMA: ALCANCIAS 3.4 (AXE, SISTEMA 12)
//
// ARCHIVO: COMM.H
//
// FECHA:  06/10/96
// ********************************************************************************
/*

        DEFINICIONES

*/
#define  vdu         0x9A  /* 1200,even parity,1 stopbits,7 bits */
#define  VDU         0x9B  /* 1200,even parity,1 stopbit ,8 bits */
#define  itf         0xE3  /*                                    */
#define  int_com2    0x0B  /* interrupt of comport 1             */
#define  break_int   0x05  /* interrupt of print screen          */
/*------ COM PORT 1 ----------------------------------------------*/
#define  dll         0x2F8 /* divisor latch LSB                  */
#define  rxb         0x2F8 /* receive buffer                     */
#define  txb         0x2F8 /* transmit buffer                    */
#define  dlm         0x2F9 /* divisor latch MSB                  */
#define  ier         0x2F9 /* interrupt enable register          */
#define  iir         0x2FA /* interrupt identification register  */
#define  lcr         0x2FB /* line control register              */
#define  mcr         0x2FC /* modem control register             */
#define  lsr         0x2FD /* line status register               */
#define  msr         0x2FE /* modem status register              */
/*------ IER ( INTERRUPT ENABLE REGISTER ) ------------------------*/
#define  thre_off    0x0D
#define  thre_on     0x0F
#define  ier_ena     0x0D
#define  ier_dis     0x00
#define  mcr_setup   0x0F  /* bit 0,1,2,3=1                      */
#define  mcr_reset   0x00
/* ----- PIC ( PROGRAMABLE INTERRUPT CONTROLER ) ------------------*/
#define  pic_irq_reg 0x21  /* Values for the 8259 Interrupt      */
#define  pic_eoi_reg 0x20  /* controller                         */
#define  end_of_int  0x20
#define  irq3_ena_mask 0xF7
#define  irq3_dis_mask 0x08
/*------ ASCII ---------------------------------------------------*/
#define  nll         0x00
#define  stx         0x02
#define  etx         0x03
#define  eot         0x04
#define  enq         0x05
#define  ack         0x06
#define  bel         0x07
#define  bs          0x08
#define  lf          0x0A
#define  cr          0x0D
#define  ff          0x0C
#define  del         0x11  /* xon                                */
```

```
#define  dc2        0x12 /*                          */
#define  dc3        0x13 /* xoff                     */
#define  xon        0x11
#define  xoff       0x13
#define  etb        0x17
#define  eob        0x18 // ???
#define  esc        0x1B
#define  sp         0x20 // Espacio en blanco
#define  dp         0x3A // Simbolo :
#define  sc         0x3B // Simbolo ;
#define  ls         0x3C // Simbolo <
#define  ms         0x3E // Simbolo >
#define  ctrl_c                          0x2E03
#define  ctrl_d     0x2004
#define  ctrl_e                          0x1205
#define  ctrl_x     0x2D18
#define  ESC                               0x011B
#define  F1                                0x3B00
#define  F2                                0x3C00
#define  LEFT                    ((int) 0x4B00)
#define  RIGHT                   ((int() 0x4D00)
#define  FALSE      0
#define  TRUE       1
#define  P8N1 1
#define  P7E1 0

/* configuration file record structure */
typedef struct {
        int port;
        long baud_rate;
        int parity;
        int lock_port;
        char initialization_string[81];
        char hangup_string[41];
        char dial_prefix[41];
} CONFIG;

int init(void);
void printcontrol(int);
void openpuertos(void);
void cierrapuertos(void);
void cuelgapuertos(void);
void clear_buffers(void);
void capture(void);
void contact(void);
void restore(void);
void send_status(void);
void send_ack(void);
void send_enter(void);
void print_help(void);
void detect_string(void);
void close_file(void);
void send_string(void);
void send_string_prev(void);
```

```
void test(void);
void interrupt far inter_hndlr(void);
void interrupt far ctrl_break(void);
int look(unsigned char);
int marca_modem(void);
void config_program(void);
int dial(void);
void exit_program(void);
char *get_modem_string(char *s, char *d);
void hangup(void);
void modem_control_string(char *s);
void read_config(void);
void save_config(void);
void vacia_buffer();
int check_rx_buf();
int f1_press();
```

```
// ********************************************************************************
// SISTEMA: ALCANCIAS 3.4 (AXE Y SISTEMA 12)
//
// ARCHIVO: NOMENCLA.H
//. DESCRIPCION: CONSTANTES DE TAMAÑOS, NUMEROS MAXIMOS Y NOMENCLATURA
// FECHA:  13/08/96
// ********************************************************************************

#define N_DISTRITOS     10   // Versión 3.4
#define S_DESDISTRITO   40   // Versión 3.4
#define N_ZONAS         10   // Número Máximo de Zonas
#define N_PUERTOS        4   // Número Máximo de Puertos
#define N_APARATOS      20   // Número Máximo de Clases de Aparatos
#define N_RANGOS       110   // Tamaño máximo del Número de Rangos
#define S_RESP        1500   // Tamaño Máximo del la respuesta
#define S_LRESP       1500L  // Tamaño Máximo del la respuesta editor C-Scape
#define S_COMANDO              254   // Tamaño Máximo de un comando

#define S_LADA          4   // Tamaño máximo de la Lada
#define S_MSG          80   // Tamaño máximo de un mensaje en C-scape
#define S_FILE         15   // Tamaño máximo de un archivo
#define S_TIT          76   // Tamaño máximo del título

#define TRUE            1   // Identif. de Verdadero
#define FALSE           0   // Identif. de Falso

#define ALCANCIA        1   // Identificador de ordenación por telefono
#define ALCANZONA       2   // Identificador de ordenación por zona
#define ALCANTAPA       3   // Identificador de ordenación por tipo apto.

#define SISTAOM         1   // Identif. de Central AXE conexión AOM
#define SIST12                      2   // Identif. de Central S12 conexión MAN-MACHINE
#define SISTIOG3                    3   // Identif. de Central AXE conexión IOG3
#define SISTIOG11                   4   // Identif. de Central AXE conexión IOG11
#define SIST12MTM       5   // Identif. de Central S12 conexión MAN-MACHINE
#define SIST12PAM       6   // Identif. de Central S12 conexión PAM

#define IZQUIERDA       0
#define CENTRO          1
#define DERECHA         2

#define TOMACENT        1   // Identif. cuando es necesario tomar la central
#define NOTOMACENT      0   // Identif. cuando no es necesario tomarla

#define TELPUB          1   // Identificador
#define MONITOR         2   // Identificador
#define TSERVIC         3   // Identificador
#define CENTRAL         4   // Identificador
#define STATCAT         5   // Identificador
#define STATCNT         6   // Identificador
#define CONTHIST        7   // Identificador
#define RESPONSA        8   // Identificador
#define SINMOV          9   // Identificador
#define DESCOMP         10  // Identificador
```

```
#define ARREGLO      11  // Identificador

#define NF_TRONRANG "TRONRANG.DAT"
#define NF_CONFIG  "CONFIG.DAT"
#define NF_DIGITOS "DIGITOS.DAT"
#define NF_DISTRITO "DISTRITO.DAT"
#define NF_TITULO  "TITULO.DAT"
#define NF_CONFAPAR "CONFAPAR.DAT"
#define NF_PUERTO  "PUERTO.DAT"

#define NF_ALCANCIA "ALCANCIA.DBF"  // versión 3.4
#define NF_FALLAS  "FALLAS.DBF"  // versión 3.4
#define NF_TELPUB  "TELPUB.DBF"  // Nombre del archivo de telefonía pública
#define NF_MONITOR  "MONITOR.DBF"  // Nombre del archivo del monitoreo de tels
#define NF_TSERVIC  "TSERVIC.DBF"  // Nombre del archivo de Tipo de servicio
#define NF_CENTRAL  "CENTRAL.DBF"  // Nombre del archivo de rangos de centrales
#define NF_STATCAT "STATCAT.DBF  // Nombre del archivo de descrip. de stat de cat
#define NF_STATCNT "STATCNT.DBF  // Nombre del archivo de descrip. de stat de cont
#define NF_CONTHIST "CONTHIST.DBF"  // Nombre del archivo de Lect. contador histórico
#define NF_RESPONSA "RESPONSA.DBF  // Nombre del archivo de responsables de zona
#define NF_SINMOV  "SINMOV.DBF  // Nombre del archivo de historico de reportes
#define NF_DESCOMP  "DESCOMP.DBF  // Nombre del archivo de descomposturas
#define NF_ARREGLO "ARREGLO.DBF"  // Nombre del archivo de arreglos

#define NX_ALCANCIA "ALCANCIA.CDX"  // versión 3.4
#define NX_FALLAS  "FALLAS.CDX"  // versión 3.4
#define NX_TELPUB  "TELPUB.CDX"  // Archivo de índices de telefonía pública
#define NX_MONITOR  "MONITOR.CDX"  // Archivo de índices del monitoreo de tels
#define NX_TSERVIC  "TSERVIC.CDX"  // Archivo de índices de Tipo de aparato
#define NX_CENTRAL  "CENTRAL.CDX"  // Archivo de índices de rangos de centrales
#define NX_STATCAT "STATCAT.CDX  // Archivo de índices de descrip. de stat de cat
#define NX_STATCNT "STATCNT.CDX  // Archivo de índices de descrip. de stat de cont
#define NX_CONTHIST "CONTHIST.CDX"  // Archivo de índices de Lect. contador histórico
#define NX_RESPONSA "RESPONSA.CDX  // Archivo de índices de responsables de zona
#define NX_SINMOV  "SINMOV.CDX  // Archivo de índices de historico de reportes
#define NX_DESCOMP  "DESCOMP.CDX  // Archivo de índices de descomposturas
#define NX_ARREGLO "ARREGLO.CDX"  // Archivo de índices de arreglos

#define NF_TRONRANG "TRONRANG.DAT"  // Nombre del archivo de rangos
#define NF_CONFIG  "CONFIG.DAT"  // Nombre del archivo de configuracion
#define NF_DIGITOS "DIGITOS.DAT"  // Nombre del archivo de control
#define NF_DISTRITO "DISTRITO.DAT"  // versión 3.4
#define NF_ZONAS  "ZONAS.DAT"  // Nombre del archivo de los responsables
#define NF_TITULO  "TITULO.DAT"  // Nombre del archivo del titulo de Reportes
#define NF_CONFAPAR "CONFAPAR.DAT"  // Nombre del archivo de config. de aparatos
#define NF_PUERTO  "PUERTO.DAT"  // Nombre del archivo de puertos abiertos

#define NF_ATEL  "ATEL.REP"  // Reporte de Telefonos ord. por teléfono
#define NF_AZONA  "AZONA.REP"  // Reporte de Teléfonos ord. por zona
#define NF_ATA  "ATA.REP"  // Reporte de Teléfonos ord. por tipo apto
#define NF_AUZONA  "AUZONA.REP"  // Reporte de Teléfonos solo una zona
#define NF_AUTA  "AUTA.REP"  // Reporte de Telefonso solo un tipo apto
#define NF_APFZONA  "APFZONA.REP"  // Reporte de Aptos. con posible falla x zona
#define NF_APFUZONA "APFUZONA.REP"  // Reporte de Aptos. con posible falla una zona
```

```
#define NF_ACCZONA  "ACCZONA.REP"  // Reporte de Aptos. con cambio de categoría
#define NF_RTEL     "RTEL.REP"     // Resumen de Teléfonos ord. por teléfono
#define NF_RZONA    "RZONA.REP"    // Resumen de Teléfonos ord. por zona
#define NF_RTA      "RTA.REP"      // Resumen de Teléfonos ord. por tipo apto.
#define NF_RPF      "RPF.REP"      // Resumen de Aptos. con posible falla x zona
#define NF_RCC      "RCC.REP"      // Resumen de Aptos. con cambio de categoría
#define NF_RSUSPEND "RSUSPEND.REP" // Resumen de Aptos. suspendidos desde sistema
#define NF_TTA      "TTA.REP"      // Reporte de Totales por tipo de aparato
#define NF_TPF      "TPF.REP"      // Reporte de Totales de aptos con posible falla
#define NF_ACONTHIST "CONTHIST.REP" // Reporte de contador historico
#define NF_HISTMOV  "HISTMOV.DAT"  // Reporte de Hist. de Movimientos codificado
#define NF_HISTMOVD "HISTMOVD.REP" // Reporte de Hist. de Movimientos decodif.
#define NF_HISTCENT "HISTCENT.REP" // Reporte de Hist. de Centrales
#define NF_CONFIGUR "CONFIGUR.REP" // Reporte de Configuración General
#define NF_ASINMOV  "ASINMOV.REP"  // Reporte de Aparatos sin movimiento de cont.
#define NF_ACAMBCAT "ACAMBCAT.REP" // Reporte de Aparatos con cambio de Cat
#define NF_ALCOD    "ALCOD.DAT"    // Archivo de codificado de passwords
```