UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA PARA LA EMISION DE REPORTES GRAFICOS
(S E R)

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

Mauricio Acosta Gutiérrez
Gerardo Guillermo León Braunschweiger
Francisco Javier Oropeza Morales

Director: ING. SALVADOR BARRA ARIAS

MEXICO, D. F., AGOSTO DE 1987

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

TABLA DE CONTENIDOS

INTRODUCCION

1.1 ANTECEDENTES HISTORICOS

Es por todos bien sabido que las ayudas gráficas en cualquier tipo de trabajo escrito son verdaderamente útiles. Un esquema o diagrama de cualquier tipo tiene la propiedad de sintetizar ideas, que por medio de palabras, tomaría varias páginas expresarlas.

Las computadoras ofrecen un gran número de posibilidades en el área de dibujo, diseño y gráficas. La herramienta que representa la computadora en esta área tiene las ventajas de poder obtener una gráfica de gran calidad y exactitud y poderla modificar con rapidez y facilidad además del procesamiento de la información a graficar.

El Centro de Cálculo de la Facultad de Ingeniería (CECAFI), lugar donde se desarrolló el trabajo que se presenta como tesis, ha proporcionado siempre a los usuarios de sus equipos, paquetes de computadora para la obtención de gráficas. Se conocen a lo largo de la historia del Centro, paquetes como el "AKD91", el cual, haciendo uso de la impresora de linea, podía obtener gráficas de funciones matemáticas. Si bien la calidad de la gráfica dejaba mucho que desear, por lo menos los usuarios podían tener una aproximación gráfica de sus datos.

Con la adquisición de una impresora-graficadora de matríz de puntos y de un paquete elemental de primitivas gráficas (PLXY), se desarrolló un modesto y limitado sistema para la generación de gráficas del tipo estadístico (barras, histogramas, líneas, diagramas circulares, etc) conocido como "SISTGRAF". Finalmente se puso a disposición de los usuarios un paquete para generar gráficas de tipo matemático (funciones y parejas de puntos). Con estos dos paquetes se pudieron cumplir en parte las necesidades de gráficas de los usuarios del equipo VAX-11/780.

Los usuarios principales de estos paquetes son los alumnos de la Facultad de Ingeniería, los cuales requieren de gráficas para la elaboración de tareas y trabajos escolares y los programadores del Centro de Cálculo, que utilizan los paquetes como una interfaz de salida gráfica para los programas que desarrollan.

La principal desventaja de todos los paquetes gráficos habidos en el Centro de Cálculo es su rigidez, tanto en el modo de uso como en la presentación de las gráficas.

INTRODUCCION

Para poder hacer uso de los paquetes, el usuario debe tener conocimientos básicos del sistema de cómputo VAX-11/780, tanto de editor como de manejo de archivos ya que los paquetes no proporcionan mecanismos de edición, salvado y recuperación de la información, ni tampoco cuentan con ayuda integrada ni con mecanismos de retroalimentación.

Por otra parte, estos paquetes carecen de sistemas eficientes de captura y despliegue de información, por lo que el usuario enfrenta problemas en estos aspectos.

Dado que las interfaces con el usuario de estos paquetes gráficos están netamente enfocadas a procesos batch, la comunicación que existe entre éstos y los usuarios es poco fluida. El usuario debe generar la información a dibujar desde su programa de aplicación, almacenar esta información en archivos y finalmente ejecutar el paquete gráfico direccionando la entrada de datos hacia los archivos del usuario.

Con respecto a la rigidez en la presentación de las gráficas en papel, podemos apreciar que se debe primordialmente a que las rutinas de dibujo se sustentaron en un paquete de primitivas muy elementales (PLXY).

I.2 PROPUESTA DE UN NUEVO SISTEMA

Analizando hasta aquí todos los defectos y virtudes de los paquetes gráficos existentes en el CECAFI, se dió a la tarea de hacer una propuesta de un nuevo sistema gráfico que contemplará los siguientes aspectos:

1) Que el sistema sea flexible tanto para los usuarios interactivos como para aquellos que necesitan una interfaz gráfica para sus programas de aplicación.

2) Que el modo de uso sea novedoso, atractivo y amigable.

3) Que el usuario defina, hasta donde sea posible, el formato de salida de su reporte gráfico.

Con el fín de madurar los tres puntos que se consideran esenciales para el éxito del nuevo sistema, se llevó a cabo un estudio de campo de algunos paquetes comerciales y una encuesta a los usuarios reales y potenciales del nuevo paquete.

1) Paquetes gráficos comerciales

   Los puntos de interés en el estudio de estos paquetes fueron: El tipo de gráficas que se pueden obtener y el modo de uso de los paquetes. La táctica empleada fué dominar el manejo de los paquetes que estuvieran al alcance y conseguir la bibliografía de los restantes.

   De esta manera conocimos a fondo paquetes como Supercalc, Energraphics, Microsoft Windows (Paint) y Autocad, que de una u otra forma agrupan todas las tendencias en gráficas en microcomputadoras.

Con respecto a paquetes hechos para minicomputadoras, en especial para VAX, consultamos bibliografía de algunos paquetes comerciales como lo son: DECgraph y D-PICT, éste último desarrollado por una compañía dedicada al desarrollo de software de gráficas (PANSOPHIC System Inc.)

2) Encuestas a usuarios

Se tienen dos grandes grupos de usuarios potenciales a los cuales se quiere servir con igual esmero. Por una parte los alumnos y profesores de la Facultad de Ingeniería que requieren frecuentemente anexar a sus tareas, trabajos escritos y prácticas de laboratorio, gráficas de diferentes tipos; y por otra los programadores analistas que para sintetizar y dar una mejor presentación a sus reportes de salida prefieren un reporte gráfico a una tabla de numeros y nombres.

Teniendo en cuenta nuestra propia experiencia como alumnos de la Facultad, se pudieron plantear las necesidades gráficas de un alumno además de enriquecer esta experiencia con las entrevistas que llevamos a cabo a algunos profesores cuyas materias hacen gran uso de herramientas gráficas.

Las necesidades gráficas de analistas y programadores del Centro fueron dadas a conocer tanto por nosotros mismos, que somos parte del CECAFI, como por un grupo selecto de programadores del mismo Centro.

Habiendo dado, hasta este momento, un panorama de los paquetes gráficos actuales y de la propuesta de uno nuevo, se pasará ahora a describir el medio ambiente, los recursos de cómputo disponibles y finalmente el producto que se ofrece.

1.3 CENTRO DE DESARROLLO

El presente trabajo se desarrolló como un proyecto del Centro de Cálculo de la Facultad de Ingeniería. Los recursos de cómputo empleados fueron:

1) Minicomputadora VAX-11/780

2) Impresora-graficadora Printronix-600

3) Terminal de video DEC VT100

4) Paquete de rutinas gráficas PGPLOT

5) Compilador FORTRAN-77

6) Rutinas de apoyo "Run time Library (RTL)"

I.4 JUSTIFICACION

Este trabajo surgió como una necesidad de modernizar y ampliar los paquetes gráficos existentes, como un proyecto que pudiera explotar los potenciales del software y hardware gráficos existentes en el sistema VAX-11/780 y como una herramienta de apoyo a los programadores usuarios del equipo VAX del Centro.

Así pues, el Sistema para la Emisión de Reportes Gráficos (SER) es una paquetería de rutinas gráficas avanzadas que permiten generar reportes gráficos de tipo matemático, estadístico y de negocios. Las rutinas pueden ser llamadas desde un lenguaje de alto nivel o usarse a través de una interfaz interactiva que acompaña al sistema.

La interfaz fué diseñada tomando en cuenta las nuevas tendencias de diseño de sistemas interactivos, es decir; sistemas sustentados en un ambiente de ventanas, incorporación de ayudas al usuario, métodos de retroalimentación sistema-usuario, utilerías de edición, salvado y recuperación de información y diseño ergonómico del despliegue de la información.

I.5 CONTENIDO DEL TRABAJO

A continuación se presenta un esbozo del contenido de cada uno de los capítulos que componen este trabajo.

El capítulo uno clasifica, desde un punto de vista particular, los diferentes tipos de paquetes gráficos que existen. Esto es con el fín de dar a conocer la gran gama de software que existe en el área de las graficas y de situar en alguna de estas clasificaciones el sistema que se presenta en esta tesis (SER).

La exposición de los criterios que deben tomarse en cuenta para el diseño de un sistema gráfico se realiza en el capítulo dos. En esa parte se enumeran los principios ergonómicos que debe cumplir un buen sistema interactivo y aquellos con los que debe diseñarse un sistema gráfico.

Haciendo mención de lo que es el sistema SER, el capítulo tres sitúa al SER dentro del contexto de las gráficas, presenta los criterios de diseño y define finalmente el modo de uso del sistema.

En el capítulo cuatro se habla del medio ambiente bajo el cual se desarrolló el sistema (hardware y software) y se presenta un panorama general de las dos partes que integran el sistema (interfaz interactiva y rutinas gráficas).

El capítulo cinco esta dedicado a tratar en exclusiva el diseño de la interfaz interactiva. Por otra parte el capítulo seis se enfoca al diseño de las rutinas gráficas.

El capítulo siete está enfocado al análisis se la aplicación de los factores humanos en el sistema SER.

Finalmente dentro de la parte octava se expresan las conclusiones del trabajo, sus aportaciones y mejoras.

CAPITULO 1
PAQUETES GRAFICOS

Con el fin de tener una mejor apreciación del tipo de sistema gráfico dentro del cual queda englobado este trabajo, se presentará a continuación una clasificación muy particular, del tipo de paquetes gráficos existentes.

En primer lugar se enunciarán los tipos de sistemas gráficos de acuerdo a su forma de uso y posteriormente conforme su aplicación. Finalmente se proporcionarán algunos datos sobre paquetes gráficos existentes en microcomputadoras y equipos mayores.

1.1 POR SU FORMA DE USO

De acuerdo a su forma de uso, los sistemas gráficos se pueden dividir en dos grandes grupos: Los sistemas interactivos y los sistemas formados por un conjunto de rutinas gráficas disponibles al usuario desde un lenguaje de programación.

1.1.1 Sistemas Interactivos

Los sistemas gráficos de este tipo utilizan un despliegue como medio de comunicación con el usuario. Los resultados se muestran directamente en el despliegue con la finalidad de que si es necesario modificarlos se pueda hacer rápida e independientemente del lenguaje de comunicación que emplee el sistema para interactuar con el usuario (como pueden ser los menús, comandos o diálogos).

Las ayudas para el usuario van desde sofisticados dispositivos de entrada/salida, con los cuales se puede introducir o recibir información, hasta simples mecanismos de ayuda textual sobre el manejo del sistema. No debe perderse de vista que siendo el despliegue la herramienta más importante en este tipo de sistemas, su mejor aprovechamiento depende de los factores que se tomen en cuenta para el acomodo y manipulación de la información sobre la pantalla. El capítulo III habla de los factores para el diseño de una interfaz interactiva.

En resumen, se puede decir que las características principales de los sistemas gráficos interactivos son:

PAQUETES GRAFICOS

1) Fácil introducción de los datos que se van a graficar.

El mecanismo empleado puede ir desde un simple editor donde se introduzcan datos numéricos o alfanuméricos, hasta el empleo de un dispositivo periférico como un digitizador, pluma luminosa, etc.

2) Rapidez en el procesamiento de la información y obtención de la gráfica.

Este punto esta muy relacionado con el equipo de cómputo empleado. Dicho equipo puede variar en tamaño y potencialidad desde una microcomputadora hasta un sofisticado sistema CAD/CAM con estaciones de trabajo inteligentes.

3) Facilidad para modificar la gráfica.

Este es un punto muy importante, ya que el usuario por lo regular "juega" con los datos que generan la gráfica hasta que ésta es de su agrado y necesidades. La forma en que se lleva a cabo la modificación de la gráfica se relaciona directamente con el lenguaje de comunicación con el usuario que emplea el sistema. Este puede ser del tipo menú, diálogos, comandos o teclas funcionales.

4) Ejecución en tiempo real.

El usuario puede apreciar inmediatamente las gráfica construidas a partir de sus datoas. De la misma manera cada ver que modifique los datos o agregue algún elemento adicional a su gráfica, el usuario podrá apreciar los resultados al momento.

5) Métodos de almacenamiento y recuperación de la información.

El sistema debe proporcionar mecanismos que permitan salvar las gráficas generadas por el usuario y los datos empleados. Esto es con el fin de poder continuar en otras sesiones los trabajos que no se hayan concluido y de poder archivar aquellos que el usuario juzgue convenientes.

6) Mecanismos para el establecimiento del medio ambiente.

Esta característica es únicamente digna de tomarse en cuenta en los sistemas gráficos para microcomputadoras. Puesto que el mercado de microcomputadoras está poblado de equipos de diversas características y marcas, si se quiere un sistema versátil, éste debe poder hacer uso de los periféricos más usados en el mercado, principalmente pantallas de despliegue, impresoras y graficadores.

En la mayoría de los casos, los sistemas gráficos interactivos comerciales están enfocados a resolver un problema específico. Si la necesidad del usuario es tal que no haya un paquete comercial que resuelva sus problemas o si por otra parte, no se tiene el presupuesto para adquirir uno, existe la posibilidad de crear software "hecho en casa". Si se elige esta opción se cuenta con otro tipo de ayudas como son las bibliotecas de rutinas gráficas.

1.1.2 Bibliotecas De Rutinas Gráficas

Estos tipos de sistemas pueden ir desde una decena de rutinas elementales para trazo de líneas y texto hasta enormes conjuntos de poderosas rutinas para dibujar círculos, rectángulos, rellenar áreas, rotar y escalar figuras y trazar texto de muy diversos estilos.

El uso de paquetes de rutinas gráficas tiene ventajas y desventajas. La principal ventaja es el poder diseñar un sistema gráfico de acuerdo a las necesidades y gustos del usuario, sin embargo, las gran desventaja es que para crear ese sistema gráfico se requiere de un gran esfuerzo de programación y de conocimientos en el área de gráficas.

Con las rutinas gráficas se tiene la opción de crear software gráfico específico para la aplicación del usuario. De esta manera se evita tener que trabajar con paquetes que resuelven a medias sus necesidades. Por otra parte, si las necesidades y aplicaciones del usuario son muy diversas, lo más conveniente es generar un sistema de gráficas de propósito general. La clave del éxito de los sistemas de propósito general radica en proporcionar una buena interfaz de comunicación entre las aplicaciones del usuario y el propio sistema gráfico.

En el ambiente de cómputo a nivel mundial, el período comprendido entre 1963 y 1974 se caracterizó por el desarrollo de muchos proyectos independientes en lo referente a software gráfico. La necesidad de un estándar se hizo evidente y muchos comités se han avocado a esta tarea desde la primera reunión de trabajo de la ACM Siggraph sobre gráficas independientes del dispositivo, apoyada por el National Bureau of Standards en abril de 1974 [1].

La idea primordial en el desarrollo de los estándares es hacer que los cuerpos principales de los programas sean independientes del dispositivo. Estos programas se pueden comunicar con cualquier dispositivo de entrada a través de un manejador del dispositivo (device handler), al cual debe, por supuesto, ser dependiente del dispositivo. Similarmente, se debe tener una interfaz a cualquier tipo de dispositivo de salida por medio de un manejador de dispositivo (device driver). El software independiente del dispositivo podrá entonces servir a varias generaciones de hardware y sería portable de un sistema gráfico a otro de diferente tipo.

Estos esfuerzos condujeron al estándar gráfico CORE, desarrollado por Siggraph en 1979 y de ahí en adelante se produjeron al menos seis estándares que funcionan a niveles nacionales e internacionales.

1) IGES, Initial Graphics Exchange Specification, aprobado en septiembre de 1981 como ANSI Standard Y14.26M. Proporciona intercambio de bases de datos gráficas entre sistemas CAD/CAM.

2) NAPLPS, North American Presentation Level Protocol Syntax, formalmente aceptado como un estándar nacional en CANADA y ANSI a finales de 1983. Este estándar utiliza secuencias de bytes en código ASCII y extensiones de código para describir gráficas y texto en marcos separados.

3) **GKS**, Graphical Kernel System, ahora casi una propuesta oficial. Algunas veces llamado "Un pequeño CORE", GKS define los límites entre las aplicaciones y el paquete de soporte gráfico; no especifica el contenido del soporte gráfico. GKS es en la actualidad un estándar en 2-D, aunque la adición de las extensiones para 3-D está en estudio.

El estándar Graphical Kernel System (GKS) especifica un conjunto de funciones y primitivas gráficas. Las funciones permiten desarrollar aplicaciones para producir dibujos en 2-D en dispositivos gráficos de salida del tipo vector y raster. El estándar define 12 niveles ascendentes compatibles de funcionalidad para cumplir los requerimientos varios de diferentes sistemas gráficos [12].

Como un estándar internacional, GKS proporciona una definición común para una interfaz gráfica que implementa funciones comunes a todas las aplicaciones. Por lo tanto, GKS permite que el programador de aplicaciones se concentre únicamente en los detalles a nivel de programación del sistema. Esto incrementa la productividad y disminuye el tiempo de desarrollo para una aplicación gráfica. GKS hace que una aplicación sea portable entre máquinas que tienen implementado el estándar.

GKS permite que un simple programa de aplicación trabaje con diversos dispositivos de entrada y salida; el programa de aplicación llama a rutinas de biblioteca comunes que manejan dispositivos gráficos virtuales de Entrada/Salida, conocidos como estaciones de trabajo . Para cada tipo de dispositivo real de Entrada/Salida, un manejador gráfico GKS traduce las llamadas a éste a instrucciones específicas del dispositivo. Por lo tanto, para explotar dispositivos de Entrada/Salida las aplicaciones en GKS necesitan solamente un nuevo manejador gráfico y no rescribir el programa de aplicación.

4) **PHIGS**, Programmer's Hierarchical Interface to Graphics Standard. Es una versión actualizada, ampliada y dinámica en 3-D de CORE que funcionará al mismo nivel que GKS.

5) **VDM**, Virtual Device Metafile. Es un formato independiente del dispositivo para dibujos gráficos por computadora concebido para satisfacer tanto los conceptos de CORE, excepto en 3-D, y GKS en cuestión de metafiles. Funcionando a un nivel justamente por encima de los manejadores de dispositivos, está involucrado en la transferencia de información de dibujos entre diferentes dispositivos gráficos.

6) **VDI**, Virtual Device Interface, un tanto detrás de VDM en el proceso de aprobación. Operando al mismo nivel que VDM, VDI es un protocolo de comunicación entre software independiente del dispositivo y código dependiente del dispositivo.

1.2 POR SU APLICACIÓN

En esta sección, más que una clasificación de los paquetes gráficos comerciales de acuerdo a su aplicación, se enumeran las áreas más comunes de uso.

1) Diseño CAD/CAM

Esta es una de las áreas más socorrida con paquetes gráficos. El tipo de sistemas que se ofrecen en este campo es netamente interactivo. Los más poderosos y complejos basan su poderío en enormes equipos de cómputo como lo podría ser una VAX-11/780 o un equipo IBM y en una estación de trabajo dotada de una pantalla de alta resolución con tabletas digitalizadoras y graficadores de precisión. Los mas modestos, pero no por ello menos importantes son los paquetes de diseño para microcomputadoras. En tal caso se requiere únicamente de un video de alta resolución, un teclado estándar y un medio de almacenamiento como un disco duro.

El área de diseño por computadora (CAD Computer Aided Design) incluye a todas esas áreas en las cuales se requiere de planos y dibujos de muy alta precisión como lo son: planos arquitectónicos, dibujos de piezas mecánicas, logotipos comerciales, estructuras arquitectónicas, circuitos electrónicos, etc.

2) Gráficas Matemáticas

En el campo de las matemáticas la ayuda gráfica es una herramienta muy difundida y utilizada. Su principal finalidad es el poder observar el comportamiento de fenómenos de diversa índole representados por un modelo matemático.

Los sistemas gráficos del área matemática permiten al usuario introducir su información en forma de una ecuación matemática o como una serie de puntos localizados en el plano cartesiano o en un espacio tridimensional. Los resultados gráficos se muestran en dos o en tres dimensiones.

Si el sistema soporta ecuaciones en tres dimensiones, se acompaña de operaciones de escalamiento, rotación y traslación de la gráfica ademas de dibujar la función con o sin líneas ocultas.

3) Gráficas para los Negocios y Estadística.

Las gráficas utilizadas en estas dos áreas son tal vez las más populares y comerciales. Muchos de los paquetes que se ofrecen en el mercado, principalmente los llamados paquetes integrados (constan de base de datos, hoja de cálculo, procesador de textos) permiten al usuario obtener en forma gráfica algunos de sus resultados.

El tipo de gráficas a las cuales se refiere este apartado incluye las siguientes: diagramas de barras, histogramas, diagramas circulares, gráficas de línea (puntos XY), polígonos de frecuencia. y las llamadas gráficas de error. Por lo regular este tipo de graficas

acompañan textos de investigaciones, estudios de campo, reportes de trabajo, etc. Proporcionan una gran ayuda en la toma de decisiones de directivos porque además de sintetizar en gran medida los resultados, su estudio y comprensión es mucho más fácil y eficiente.

Muchos de los paquetes de esta área ofrecen ayudas estadísticas al usuario para el tratamiento de sus datos como pueden ser: regresión matemática, suavizamiento de curvas y cálculo de parámetros muestrales.

4) Diseño Artístico

Dentro de esta área del quehacer humano se agrupan paquetes gráficos que facilitan enormemente el diseño de dibujos artísticos. Por lo regular estos paquetes se auxilian de un ratón ("mouse") que hace las funciones de una pluma de dibujo. La pantalla de video simula una hoja de papel sobre la que se hace el dibujo. En la mayoría de los casos el usuario dispone de diferentes anchos de líneas, patrones de relleno de figuras, borrado, alejamiento y acercamiento ("zoom"), movimiento de la hoja de dibujo etc.

Los paquetes de este tipo tratan de simular al máximo las operaciones que usualmente ejecuta un dibujante para crear una obra. De esta manera se proporcionan al usuario herramientas tales como: hojas de papel, lápiz, pincel, brocha, borrador, colores, tijeras y "spray" que operan a través del ratón.

5) Paquetes para el diseño de láminas de presentación.

Los paquetes que se agrupan en esta área son muy útiles para el diseño de diagramas de cualquier tipo, láminas para uso en exposiciones o presentaciones de algún producto. Las herramientas que se proporcionan son principalmente: Numerosos tipos de letras (diferentes estilos y tamaños), mezcla de textos y gráficas, inclusión de dibujos almacenados por el propio sistema, figuras geométricas, flechas, colores, etc. Con el uso de un paquete de este tipo se pueden generar por ejemplo, un diagrama de flujo, un diagrama de bloques de algún proceso , presentar un producto o preparar una exposición audiovisual.

1.3 PAQUETES GRAFICOS COMERCIALES

En el mercado del software es muy fácil encontrar varios paquetes gráficos orientados a alguna de las áreas antes mencionadas. La mayoría de estos paquetes tienen como finalidad primordial la generación de gráficas, mas sin embargo, otros utilizan las gráficas únicamente como una opción para presentar los resultados.

En el área de diseño se ha extendido mucho el uso del paquete AUTOCAD, el cual ofrece grandes facilidades a los usuarios de microcomputadoras. AUTOCAD puede encajar muy bien en cualquier aplicación de dibujo , pero su mayor potencialidad se encuentra en el diseño de planos arquitectónicos. Dentro de este ramo el paquete ofrece las siguientes operaciones: dibujos con acotamientos a escala, diseño en dos y tres dimensiones, ampliación y reducción

de dibujos, puede integrar un dibujo por medio de elementos previamente creados por el mismo paquete, permite obtener los planos a papel en graficadores muy diversos.

Muy por encima de los sistemas de diseño por computadora para equipos pequeños se encuentran los sistemas CAD/CAM sustentados en minicomputadoras. Las compañías que han destacado enormemente en el area de diseño por computadora a gran escala son Hewlett Packard, Gould, IBM e INTERGRAPH. Los paquetes gráficos anteriores son de tal envergadura que requieren tener todo un equipo de cómputo de tamaño mediano dedicado y el uso de estaciones de trabajo inteligentes de muy alta resolución.

La compañía INTERGRAPH , reconocida a nivel mundial en el área de CAD, produce software gráfico de una calidad impresionante. Sus productos requieren de un equipo VAX al cual se le han incluido procesadores extras dedicados al manejo de gráficas. Con tales adaptaciones al hardware se pueden producir sólidos de revolución en unos cuantos segundos, sombrear cuerpos, simular animación, efectuar ampliaciones y reducciones de cuerpos de cualquier complejidad. INTERGRAPH ofrece paquetes de diseño especializados en muy diversas áreas; como son: Topografía, Ingeniería Civil, diseño de circuitos electrónicos, Geodesia, diseño de plantas industriales entre otras.

En el área de los sistemas gráficos matemáticos es difícil encontrar paquetes exclusivos para este tipo de aplicaciones. Dentro de los paquetes comerciales se encuentra ENERGRAPHICS, el cual permite dibujar funciones matemáticas a partir de una ecuación matemática o de una tabla de puntos. El sistema tiene la capacidad de presentar funciones en dos y tres dimensiones, de aplicar operaciones en tres dimensiones como "zoom", rotación, traslación y eliminación de lineas ocultas. Ofrece también medios para obtener la ecuación que representa mejor una muestra de puntos por medio de regresión de diversos ordenes y de suavizar una curva.

Los sistemas gráficos de enfoque estadístico y de negocios son los más prolíferos. Numerosos paquetes para microcomputadoras pueden obtener gráficas de este tipo. Dentro de los paquetes comerciales mas conocidos en esta área se pueden mencionar: LOTUS 1-2-3, FRAMEWORK, SYMPHONY y SUPERCALC. Este último es una hoja electrónica que además de su enorme poderio en cuestiones de cálculos permite gran flexibilidad y variedad en salidas gráficas. Lo más sorprendente es la sencillez con la que se opera el sistema para generar una gráfica. Se pueden obtener siete tipos de graficas, con diferentes estilos de letras para los rótulos, el tamaño de la hoja de impresión la define el usuario, la posición del dibujo dentro de la hoja en variables, se soportan dispositivos de salida como impresoras y graficadores de plumillas, la gráfica se muestra en el video o en papel, se pueden modificar fácilmente tanto las características de la gráfica como los datos que la generan, el número de gráficas y de datos es casi ilimitado.

CAPITULO 2
DISEÑO DE SISTEMAS GRAFICOS

En este capítulo se tratarán los criterios involucrados en el diseño de sistemas gráficos. Por su complejidad, fue necesario dividirlo en dos grandes grupos. El primero corresponde a la interfaz con el usuario y el segundo a las rutinas gráficas. Sin embargo, se podrá apreciar fácilmente, que los lineamientos para el diseño de sistemas gráficos se pueden aplicar con gran éxito, al diseño de sistemas en general. Además, en este capítulo se delinean los criterios que motivaron muchas decisiones tomadas en la realización del proyecto. Por esto, se recomienda destinar especial atención a él.

Todos los sistemas de orientación gráfica, o simplemente gráficos, constan de dos partes. La primera de ellas trata sobre la sección encargada de producir las gráficas o imágenes. A esta sección se le denomina bloque de rutinas gráficas. La otra sección, considerada por muchos como la menos importante, sin serlo así, recibe el nombre de interfaz con el usuario y será el punto a tratar a continuación.

2.1 INTERFAZ CON EL USUARIO

Como Newman y Sproull [2] mencionan en su libro, una de las partes más impredecibles de un sistema gráfico, o programa gráfico, es la interfaz con el usuario. La interfaz es la sección del programa que se encarga de llevar el control de la comunicación de la computadora hacia el usuario y viceversa. Sin embargo, es la interfaz, en última instancia, quien determina si el sistema por completo es del agrado o no del usuario y, eventualmente, si lo usará o no.

Actualmente se destina tán poca atención al diseño de la interfaz, que no es raro encontrar sistemas en donde el usuario debe recordar una gran cantidad de comandos poco claros y donde tiene que oprimir secuencias de teclas sin saber su significado, como Control Q o CAT. Sucede frecuentemente también que, una vez dado el comando, el sistema no responde inmediatamente y cuando así lo hace, el usuario se sorprende o confunde por el resultado. Es común ver que cuando el usuario comete algún descuido insignificante, le provoca al programa reacciones desastrosas, teniendo que comenzar desde el principio otra vez.

Las frustraciones y la disminución de la productividad del usuario son culpa principalmente de un mal diseño de la interfaz. Los diseñadores del sistema pocas veces se preocupan en considerar las características del usuario promedio que utilizará su programa. Y es por esto que elaboran sistemas útiles sólo para

ellos mismos.

Los criterios para el diseño de interfaces gráficas es un tema muy amplio y, por este motivo, sólo se tratará el aspecto que se apega a las características de sistemas semejantes al que motiva la tesis. Sin embargo, en el capítulo 28 de la referencia [2] puede encontrarse más información útil para el diseño de interfaces de sistemas gráficos, altamente interactivos. De los sistemas de este tipo son clásicos los CAD y sistemas para dibujos artísticos. A lo largo de este tema, se podrá apreciar que los lineamientos que se expresan para el diseño de sistemas gráficos no interactivos, son fácilmente aplicables a sistemas de cualquier tipo.

Frecuentemente sucede que todos aquellos que diseñan un sistema, cuando se enfrentan a la necesidad de construir su interfaz, toman muchas decisiones basadas en la experiencia y en las reglas que dicta el sentido común. No obstante, no son la experiencia ni el sentido común quienes deben guiar al diseñador. Gran cantidad de expertos han estudiado y determinado qué factores provocan que un sistema sea del agrado o no del usuario y los han denominado Factores Humanos.

Los Factores Humanos son aquellos que deben incorporarse en la elaboración de cualquier producto, desde una herramienta hasta un sistema de computadora, para que sea utilizable por un ser humano. Este término nació en Inglaterra durante la Primera Guerra Mundial, pero no es sino hasta ahora que ha tomado fuerza bajo el nombre de Ergonomics en Estados Unidos, y de Factores Humanos en el resto del mundo.

A continuación se presenta una información condensada de las reglas de los Factores Humanos aplicables a un sistema de cómputo. Información mas detallada se puede encontrar en las referencias [9] y [10], y en el capítulo 28 de la referencia [2].

2.1.1 Análisis Del Usuario

Lo primero que debe de hacer el diseñador de sistemas es realizar un análisis del usuario promedio que empleará su sistema. De este análisis el diseñador obtendrá información muy valiosa y, eventualmente, logrará conocer cómo el programa debe comunicarse con el usuario. Existen dos puntos en que el diseñador debe concentrar su atención; las siguientes líneas detallan ambas rúbricas.

2.1.1.1 Con Relación A Las Habilidades Del Usuario.

Dos factores, el conocimiento semántico y el conocimiento sintáctico, son los que determinan las habilidades de un usuario. Esencialmente giran alrededor del conocimiento y dominio que un usuario tiene sobre un programa en particular.

Mientras el conocimiento semántico radica en comprender qué hace el sistema, es decir, el grado de comprensión del funcionamiento general de éste, el conocimiento sintáctico trata sobre las acciones que debe realizar el usuario para ejecutar una tarea en particular del sistema.

De acuerdo a las dos categorías cognoscitivas, un usuario puede clasificarse en cuatro niveles:

1) **Usuario Improvisado.**- Se conoce como usuario improvisado a todo aquel que carece de conocimiento semántico y sintáctico. Desconoce con exactitud qué puede realizar el sistema y, más aún, qué debe hacer para realizar una acción en particular.

2) **Usuario Principiante o Novato.**- En esta categoría quedan comprendidos todos los usuarios que tienen un buen conocimiento semantico, pero su conocimiento sintactico es débil y tienen problemas para controlar el sistema.

3) **Usuario Competente.**- Corresponde a aquellos usuarios en los que ambos niveles de conocimiento son satisfactorios.

4) **Usuario Experto.**- Este tipo de usuarios conocen tan bien el sistema que ni siquiera piensan en los comandos que están ejecutando o en lo que hacen, y a diferencia de los usuarios competentes, simplemente operan el sistema de una forma automática. Se pueden comparar al extranjero, que después de vivir muchos años en México, puede hablar el Español sin tener que pensar si una oración está bien formada. Es obvio entonces que sus conocimientos semánticos y sintácticos son también satisfactorios.

2.1.1.2 Forma En Que El Usuario Interactúa Con El Sistema.

Además de considerar las habilidades de un usuario, el diseñador de sistemas debe también concentrar su atención en determinar la forma en que un usuario interactúa con el sistema. Fundamentalmente, trata los aspectos de frecuencia y duración de uso del sistema; así como la libertad que tiene el usuario para emplearlo o no. En base a estos criterios el usuario recibe un nombre distinto. A continuación se describe cada uno de ellos.

1) **Con respecto a la frecuencia de uso.**

De acuerdo a la frecuencia con que un usuario emplea un sistema, se puede clasificar en usuario Casual o usuario Dedicado. El usuario Casual es aquel que emplea el sistema sólo de vez en cuando, y por este motivo es característico que este tipo de usuarios sean Improvisados y sólo en algunos casos podrán llegar a ser Competentes. Por otro lado, están los usuarios Dedicados. A esta categoría corresponden los usuarios que utilizan el sistema como parte de su trabajo cotidiano; y es común que ese tipo de usuarios sean Competentes o Expertos.

2) **Con respecto a la duración de uso.**

A esta rúbrica corresponden dos tipos de usuarios. El usuario Continuo es aquel que puede fijar toda su atención en el momento que usa el sistema, pues no se le interrumpe. A diferencia de éste, el usuario Intermitente tiene que suspender el uso del sistema pues, con frecuencia, recibe una gran cantidad de interrupciones. Tal es el caso de las secretarias cuando usan procesadores de palabras. Para este tipo de usuarios, el sistema debe poder "congelarse", para luego poder retornar al punto exacto donde el sistema fue suspendido.

3) **Con respecto a la libertad de uso.**

Algunos usuarios deben utilizar un sistema debido a que no tienen otra alternativa, pues éste es indispensable para realizar su trabajo. Este tipo de usuarios se catalogan como usuarios Cautivos. También existen los usuarios Libres, los cuales tienen la libertad de usar o no el sistema, ya que cuentan con sistemas manuales o semejantes. Si el sistema no incorpora buenos factores humanos, el usuario Libre eventualmente lo desechará por completo, mientras que el usuario Cautivo sufrirá de tedio, fatiga y en última instancia se resistirá a realizar su trabajo, por lo que requerirá de una supervisión más estrecha.

2.1.2 Análisis De Las Funciones

Además del análisis del usuario, el diseñador debe realizar un análisis de las funciones del sistema. Básicamente, consiste en definir el objetivo general del sistema, las funciones que debe tener para cumplir con el objetivo y los pasos, o acciones, que el usuario debe realizar para llevar a cabo dichas funciones. En otras palabras, es un desgloce de los pasos desde que se inicia el sistema hasta que termina la ejecución del mismo.

De este análisis se desprenden características fundamentales que debe tener el sistema si se desea que incorpore buenos Factores Humanos, como son capacidad de recobrarse de un error, de revisar los datos que se han introducido, etc. En forma paralela, también se determina cómo son los datos que manejará el sistema.

Una vez que se ha caracterizado a los usuarios de acuerdo a las clasificaciones mencionadas y se han determinado las funciones que debe realizar el sistema, es posible entonces, aplicar cinco lineamientos, que el sistema deberá cumplir para incorporar buenos factores humanos. Los cinco lineamientos, o principios de los Factores Humanos, se establecen en el siguiente punto.

2.1.3 Principios De Los Factores Humanos.

Son cinco los principios en donde se establecen los criterios generales que los Factores Humanos dictan para todo sistema; el diseñador deberá intentar aplicarlos en la medida que le sea posible. Versan sobre los siguientes puntos:

1) **El sistema debe incluir una serie de funciones que cubran las necesidades del usuario y así, pueda realizar su trabajo.**- Trata fundamentalmente de las funciones no primordionales, pero convenientes de un sistema, como son despliegue y edición de datos, validación de la información, funciones de ayuda, etc.

2) **Adecuar las características del sistema a las características del usuario.**- Se refiere a la selección de los formatos de comandos (menús, diálogos, teclas funcionales o comandos directos), a la adecuación de los mensajes del sistema al nivel de competencia del usuario, al número de opciones entre las cuales se debe elegir un comando, a la demanda de la memoria de un usuario y al significado de consistencia y estandarización de un sistema.

3) **El sistema debe concordar con el modelo interno del usuario.**- Consiste en diseñar el sistema de tal forma que el usuario pueda aprender a emplear el sistema por pequeños bloques funcionales, y también a complementar el aprendizaje por medio de manuales y cursos.

4) **El flujo operacional del sistema debe concordar con el proceso de pensamiento del usuario.**- Trata sobre el manejo de los retardos en la respuesta de un sistema y de la forma de llamar la atención del ususario.

5) **El sistema debe evitar incomodidades físicas al usuario.**- Se refiere a la adecuación del mobiliario y del equipo de cómputo a las características antropométricas del usuario.

2.1.3.1 Primer Principio De Los Factores Humanos.

"EL SISTEMA DEBE INCLUIR UNA SERIE DE FUNCIONES QUE CUBRAN LAS NECESIDADES DEL USUARIO Y ASI, PUEDA REALIZAR SU TRABAJO"

Además de las funciones esenciales requeridas para que el sistema realice su objetivo fundamental, debe incluir una serie de funciones adicionales que cubran las necesidades particulares del usuario. Por ejemplo, es común que un usuario desee corregir un error en los datos que acaba de introducir. Si para lograrlo debe terminar la ejecución del sistema e introducir todos los datos nuevamente, el usuario terminará por frustrarse. Funciones como la de edición de datos, que

no son parte del objetivo básico del sistema, engrosan la lista de funciones adicionales, útiles en la ejecución del sistema. La lista completa es la siguiente:

1) **Desplegado del estado del sistema**

Es común que al operar sistemas mas o menos complejos, el usuario pierda la visión de dónde se encuentra el comando que está ejecutando, dentro del contexto de la estructura global del sistema. Por este motivo, es indispensable indicarle al usuario qué parte del programa, o comando, se encuentra ejecutando. Y así por ejemplo, cuando el usuario seleccione el comando de edición o impresión del documento en un procesador de palabras, el programa deberá mostrar el estado, edición o impresión, del comando que está ejecutando.

El estado del sistema debe desplegarse en un lugar de la pantalla, procurando que sea siempre en la misma zona. Por ejemplo, en la parte superior. Lo esencial consiste en que el usuario entienda, con el mensaje, qué parte de la estructura global del sistema se está ejecutando.

Frecuentemonte para lograr que el usuario se habitué a localizar rápidamente la información en la pantalla, se establecen zonas que la dividen en secciones. A cada zona se le asigna un propósito. Por ejemplo, una zona puede estar orientada al desplegado del menú de opciones; otra al desplegado de datos y resultados; y otra al envío de mensajes de error y preguntas para que el usuario las responda. Es vital que el uso de las zonas sea consistente, esto es, que los mensajes de error, por ejemplo, siempre se envíen a la misma zona. Algunos sistemas permiten zonas ajustables por el usuario. Esto es útil cuando coexisten varios niveles de usuarios, como Principiantes y Expertos. Así los usuarios Principiantes destinarán gran espacio para la zona de mensajes de error y de preguntas, mientras que los usuarios Expertos, preferirán una zona de desplegado de datos mayor, sacrificando la zona de desplegado de errores. Cuando no es posible tener zonas ajustables, el diseñador del programa deberá decidir las dimensiones de cada zona, atendiéndose a las implicaciones que esto tiene cuando coexisten varios niveles de usuarios. Si es posible, deberá intentar separar cada zona de la pantalla con espacios en blanco, líneas, diversas intensidades en las letras o inclusive distintos colores.

Un aspecto importante en el diseño de la organización de la pantalla, consiste en presentar la información por páginas y no por movimiento vertical ascendente de líneas (scrolling en Inglés). Esto es, la pantalla se debe borrar antes de desplegar nueva información.

2) **Desplegado de Datos.**

El usuario debe poder indicarle al sistema que desea desplegar datos que previamente haya introducido, para poder revisarlos. Pues es muy común que se realice un cambio de pantalla, es decir, cambio de

información desplegada en la pantalla, cuando se teclean nuevos datos. Es conveniente que la información se despliegue con un orden, para que el usuario pueda localizarla fácilmente. El orden puede ser alfabético, numérico o cronológico.

3) **Edición de Datos.**

El usuario frecuentemente comete errores cuando está proporcionando o introduciendo datos al sistema. El programa debe proveer un comando que le permita al usuario editar, es decir, modificar datos previamente introducidos. En la mayoría de los casos la edición se puede realizar en tres etapas. La primera sucede en el preciso momento que el usuario está introduciendo el dato, con las teclas de "espacio para atrás" o de edición. La segunda corresponde a la edición después de que se ha introducido un bloque de datos. El sistema preguntará si se desea corregir algún dato, posteriormente pedirá el número de datos a corregir y finalmente el usuario deberá reescribir el dato erroneo. La tercera, y última etapa, ocurre cuando la edición se realiza después que los datos han ingresado ya a la base de datos, o conjunto de archivos.

4) **Validación en la introducción de Datos.**

Algunas veces se requiere que el dato que escribe el usuario cumpla con características especiales de validación. Tal y como sería, por ejemplo, con el nombre de un derechohabiente de un servicio médico. Se requiere que el nombre que proporciona el usuario concuerde, letra a letra, con el nombre como se dió de alta al derechohabiente. Ante esto se pueden tomar cuatro alternativas:

a) Exigir que el nombre que da el usuario, concuerde perfectamente al almacenado en el archivo. El sistema avisará al usuario si no existe concordancia y pedirá que intente proporcionarlo en forma correcta nuevamente. Sin embargo, esto puede ser sumamente tedioso si frecuentemente el usuario debe proporcionar nombres complicados o largos.

b) Si no hay concordancia con el dato almacenado, el sistema puede intentar predecir cuál debería haber sido el dato de entrada. Pero este método sólo debe emplearse si las consecuencias de la predicción no son graves. Por ejemplo, en el caso de solicitar un desplegado de información detallada de un derechohabiente, lo peor que puede ocurrir es que muestre la información de otro. Pero si la predicción no es correcta cuando se solicita una baja, las consecuencias pueden ser extremadamente graves.

c) Una técnica modificada de la predicción del dato, consiste en mostrar al usuario dicha predicción y pedirle que confirme si está correcta. Sin embargo, este método termina por fatigar al usuario, pues toda acción por simple que sea, requiere de confirmación. Y

como no siempre la predicción es correcta, es indispensable rescribir frecuentemente el dato de entrada.

d) Otra técnica es verificar primero la concordancia. Si no existe, el sistema presenta una lista de opciones, en donde se muestran los datos almacenados más parecidos al de entrada, dejando que el usuario seleccione uno y así él se haga responsable de la acción que se tome.

5) **Recuperación del Error.**

Siempre existen errores al operar un sistema y es indispensable, si no se desea que el usuario deba poner extremado cuidado en su manejo, que tenga buenos mecanismos para recuperarse del error. Existen dos tipos de error:

a) **Errores de Control** .- Son aquellos que llevan al sistema a un estado no deseado, tal y como sucede cuando se elige erróneamente una opción en un menú; el sistema debe permitir salirse de ese estado erróneo. Cuando se emplea un esquema de menú de opciones, simplemente consiste en colocar una opción de "salida".

b) **Error de Datos** .- Sucede cuando los datos se escriben mal o se seleccionan erróneamente datos. El sistema debe, mediante la petición del usuario, regresar al punto exacto de entrada para ese dato específico y así pueda rescribirse.

6) **Función de Ayuda o Asistencia.**

Las funciones de ayuda, o asistencia, son líneas de texto integradas al sistema, que se le muestran al usuario para ayudarlo en su operación. Existen varios aspectos importantes involucrados en el diseño de ayudas. Estos se detallan en los siguientes párrafos.

La función de ayuda se debe diseñar acorde al nivel de experiencia del usuario. Así si el usuario es Improvisado, requerirá que la función de ayuda provea de gran cantidad de ayudas semánticas y sintácticas, donde se expliquen las funciones del sistema y las acciones para operarlo. En caso de que el usuario sea Principiante, la función de asistencia deberá tener ayudas semánticas como simples recordatorios breves, en donde se muestren los lineamientos funcionales básicos del sistema. Pero también deberá tener ayudas sintácticas en forma de listas de comandos con argumentos, por ejemplo.

Si el tipo de usuario es Competente, el nivel de asistencia deberá diseñarse de modo que sólo se muestre una ayuda sintáctica en la forma de una lista sintetizada de comandos, sin argumentos. En caso de que

el usuario sea Experto, lo más recomendable es no proporcionarle una función de ayuda, pues esto lo distraería.

La función de ayuda debe también diseñarse dependiendo de su frecuencia de uso. Así pues, las funciones con menos uso, deben disponer de un mayor número de ayudas. Cuando existen usuarios con distintos niveles de experiencia, la función de ayuda debe de estar orientada al de menor experiencia. Para no distraer a los usuarios con mayor nivel, la función de ayuda puede estructurarse de modo que muestre primero una lista abreviada de comandos para usuarios Competentes. Si el usuario lo demanda, mostrará un segundo nivel, en donde se desplegará una lista más detallada de los comandos, acompañados de argumentos y organizada para proporcionar una idea vaga del funcionamiento del sistema. Un tercer nivel también estará a disposición del usuario, para explicar ampliamente las funciones del sistema. El usuario entonces, irá avanzando por cada nivel hasta encontrar el que satisfaga sus necesidades.

El diseñador debe tener en mente que la función de ayuda se diseña de acuerdo al estado en el que se encuentre el sistema. Así, la ayuda desplegada está relacionada, única y exclusivamente, con el comando del sistema que se está ejecutando en ese momento. Una vez que la ayuda ha terminado, el sistema debe retomar el control a partir del mismo punto de operación, antes de que la ayuda se desplegara.

2.1.3.2 Segundo Principio De Los Factores Humanos.

"ADECUAR LAS CARACTERISTICAS DEL SISTEMA A LAS CARACTERISTICAS DEL USUARIO"

Son seis los aspectos que cubren este principio. Basicamente tratan sobre las características que debe tener el sistema, en función de las limitaciones cognoscitivas y del nivel de experiencia del usuario promedio, esto es, Improvisado, Principiante, Competente o Experto. A continuación se detallan estos aspectos.

1) **Fácil de Entender o Fácil de Usar.**

Lo primero que debe realizar el diseñador de sistemas es determinar si el sistema debe ser "fácil de usar" o "fácil de entender". Un sistema "fácil de entender" es aquel en donde el usuario puede comprender qué hace el sistema y qué tiene que hacer para operarlo. Tiene ayudas para su comprensión donde se enfatiza la asistencia semántica y sintáctica. A diferencia de este, un sistema "fácil de usar", centra su atención en buscar la menor actividad del usuario, ya sea cognoscitiva o física, para utilizar el sistema. En general, un sistema "fácil de usar" se traduce en última instancia en teclear lo menos posible.

Despúes de haber determinado si el sistema debe ser "facil de usar" o "fácil de entender", la seleccion del formato de comandos, para que el usuario ejecute acciones, es automática. Así se tiene que, los comandos de tipo menú de opciones y pregunta-respuesta son "fáciles de entender", mientras que los comandos del tipo órdenes directas son "fáciles de usar".

En los siguientes parrafos, se describen las caracteristicas generales de cada uno de los tipos de comandos; menú, teclas funcionales, pregunta-respuesta y comandos directos.

a) **Menú.**- Una de las formas más populares para comandar una acción consiste en un menú de opciones, o simplemente menú. El menú es una lista en la pantalla, de todas las acciones que el usuario puede ejecutar en un momento dado. El usuario simplemente selecciona de esa lista, cuál acción desea ejecutar. El sistema debe notificar al usuario, cuál accion seleccionó, de modo que pueda comprobar que el sistema recibio perfectamente su instrucción. A esta forma de notificación de "orden recibida" se le conoce como retroalimentación del comando. Existen varias formas de retroalimentar, una de ellas es informar a qué estado ha avanzado o navegado el sistema. Otra consiste en resaltar la opción que el usuario ha seleccionado.

   Son dos los criterios que se emplean para conformar un menú. El primero indica, que debe intentarse agrupar las acciones que tengan funciones semejantes. Y el segundo, marca que deben integrarse en un mismo menú, acciones que frecuentemente se usan. Si esto crea conflicto con el primer criterio, el criterio de funciones semejantes prevalece. Debe también intentarse colocar las opciones que más se usan al principio de la lista y al final, las de menor uso.

   El formato de menú presenta ciertas desventajas. Entre ellas están, que el listado de las opciones puede ocupar gran espacio en la pantalla, el tiempo de despliegue del menú puede ser considerable y que la selección de una acción puede requerir la navegación excesiva entre catálogos. Esto es, tener que seleccionar una gran cantidad de opciones de varios menus, hasta que el sistema muestre la acción específica que se desea. Tal y como sucede en un organigrama empresarial, en donde es necesario pasar por empleados, subjefes, jefes, subgerentes, etc., para poder llegar al director de la empresa.

b) **Teclas Funcionales.**- Otra técnica usada con moderación, consiste en asignarle a un determinado conjunto de teclas, ciertas acciones. De modo que cuando el usuario oprime una tecla funcional, el sistema realiza esa acción, o función.

   Esta técnica si bien es mas rápida para realizar acciones y no ocupa espacio en la pantalla, implica que el emplear gran cantidad de teclas funcionales, puede sobrepasar las capacidades del usuario para recordar las funciones asignadas a cada tecla. Además, como algunas veces no se cuenta con suficientes teclas, a cada una se

les asigna dos o mas funciones. Así, la acción de cada tecla depende del estado del programa. Por ejemplo, cierto sistema puede usar la tecla B para borrar archivos y como letra B en palabras de textos. Este complica altamente el uso del sistema y debe tratar de evitarse.

c) **Pregunta-respuesta**.- En este formato de comandos el sistema envía mensajes al usuario, pidiendo que responda con un "no" o "si", con un valor número, o con una cadena de caracteres; que podría ser, por ejemplo, el nombre de un archivo. Con este valor, el sistema realiza las acciones adecuadas y transita a otro estado. Una vez ahí, el sistema envía nuevamente otra pregunta al usuario. Así sucesivamente hasta que finaliza la operación del sistema.

A pesar de que este formato de comandos, por su simplicidad, tiene simpatizantes especialmente entre los usuarios improvisados y los Principiantes, el usuario debe de responder a un gran número de preguntas para realizar una acción completa, por simple que ésta sea. Sin embargo, en algunas circunstancias es conveniente emplear este formato para acciones de gran riesgo, por ejemplo, cuando el usuario desea borrar un archivo. Para estas ocasiones el sistema debe pedir que el usuario confirme si desea borrar o no un archivo en particular.

d) **Comandos Directos**.- Es básicamente un lenguaje simplificado, en donde una palabra escrita por el usuario realiza una acción en particular. Como es obvio, todas las palabras, o comandos, deben respetar una sintaxis. Frecuentemente, pero no siempre, cada comando tiene asociados una serie de parámetros, modificadores y delimitadores. El siguiente comando, que es ficticio, borra las líneas de un programa en BASIC de la 1 hasta la 100 inclusive y las líneas 300 y 400 también.

Borra 1 hasta 100, 300, 400

En el ejemplo, se puede apreciar que el comando consta de la palabra genérica, del propio comando, en donde se especifica la acción ("Borra"). Consta además, de operandos ("1","100", "300" y "400"). Contiene modificadores ("hasta" y ","), que indican un modo especial de interpretar los operandos. Finalmente existen delimitadores (espacios en blanco) entre los elementos del comando.

Este tipo de formato es muy poderoso, pues con un solo comando puede indicarse, lo que con un formato de menú, significaría transitar entre un buen número de pantallas, o responder a incontables preguntas con el formato de pregunta-respuesta. Sin embargo, el usuario debe tener un nivel Competente o Experto para emplearlo. Y aún así, no se libra de tener que aprender la sintaxis de los comandos, los comandos en sí y la semántica, es decir la función de cada comando. En última instancia, esto se traduce en mucho tiempo invertido.

Existen algunas ideas, no mencionadas en este documento, en relación a la técnica de la elaboración de comandos en base a objetos y acciones. En caso de asi requerirlo, es recomendable consultar el capitulo 28 de la referencia (2), para obtener más información.

A continuación se explicará otro aspecto relacionado con la adecuación de las caracteristicas del sistema a las caracteristicas del usuario.

2) De Acuerdo al Nivel Cognoscitivo del Usuario.

Además del criterio "fácil de usar" o "fácil de entender" para determinar qué tipo de formato de comandos elegir, también se puede seleccionar de acuerdo al nivel de experiencia del usuario.

a) Si el usuario es improvisado, el formato más adecuado es el de menú, ya que proporciona la ayuda semántica requerida. Cada opción del menú debe explicar, al menos someramente, los aspectos funcionales de la opción. El menú es adecuado para la selección de un número limitado de datos discretos, sin embargo, para valores continuos el formato de pregunta-respuesta es ideal, la pregunta debe ser lo más clara posible. Tal es, por ejemplo, el caso de la elección del tipo de dispositivo de almacenamiento. Como existe un número limitado de dispositivos, es posible elaborar una lista completa para que el usuario elija el dispositivo que desea. Si el usuario debe indicar los pesos de una serie de individuos, formar una lista con todos los pesos posibles es inadecuado. Lo apropiado es pedir al usuario que introduzca a la computadora directamente el peso de cada individuo.

b) Para los usuarios Principiantes, también el formato adecuado es el de menú. Como no requiere de ayudas semánticas intensas, las opciones deben abreviarse. Para este tipo de usuarios también es adecuado el esquema de pregunta-respuesta, pero a diferencia del usuario improvisado, las preguntas pueden ser breves, porque un Principiante entiende la función de la acción que va a ejecutar.

c) Los usuarios Competentes prefieren el formato de comandos directos. Los comandos deben ser nemotécnicos para que el usuario recuerde fácilmente su función. El formato de pregunta-respuesta debe emplearse sólo cuando se requiere una secuencia estricta de comandos, y asi, evitar que el usuario tenga que aprender toda la secuencia completa.

d) En el caso de los usuarios Expertos, el formato adecuado es comandos directos. Como el usuario conoce a la perfección el repertorio de comandos, puede controlarlos, ejecutándolos con el órden y ritmo que desea. No es necesario que los comandos sean nemotécnicos ya que el usuario no piensa en el comando, sino que reacciona en forma automática.

e) Algunas veces coexisten varios niveles de usuarios, para este caso se puede elegir la alternativa del formato de menú. Con la diferencia, que se le agrega una opción que cambia al modo de comandos directos. Este esquema es adecuado cuando varía la frecuencia de uso del sistema, o bien, cuando el usuario empieza en Improvisado o Principiantes para eventualmente progresar a Competente o Experto.

3) **Mensajes del Sistema.**

Además de que el formato de comandos debe de adecuarse al nivel del usuario, los mensajes que el sistema envía también deben estar de acuerdo a las características del usuario. Así, los mensajes a los usuarios Improvisados o Principiantes deben ser extensos, de tal suerte que en el propio texto se indique que hacer. Los mensajes a los usuarios Competentes o Expertos deben ser cortos, pero nemotécnicos, pues sólo sirven como simples recordatorios.

Nunca deben emplearse mensajes codificados en forma numérica, sin importar el nivel del usuario. Pues su significado es difícil de recordar aún para los usuarios Expertos.

4) **Amplitud de las Decisiones.**

Una vez que se ha elegido el formato de comandos, es vital centrar la atención en las limitaciones congnoscitivas del usuario. Así por ejemplo, es responsabilidad del diseñador del sistema, decidir qué tantas opciones deben de asignarse a cada menú. O si es el caso, cuántos comandos deben de integrar el repertorio de comandos directos.

La amplitud de las decisiones es el número de elementos entre los que hay que elegir la función a realizar. Así, en el formato de menú, es el número de opciones que lo conforman. Para el formato de comandos directos, es el número de comandos individuales que integran el repertorio. Una práctica sana indica limitar la amplitud de las decisiones a diez. Si se requiere tener una amplitud mayor, se deberá tener asistencias integradas, tal y como una función de ayuda.

5) **Memoria de Corto Plazo.**

La Memoria de Corto Plazo es otra limitación de tipo cognoscitivo y se describe como el proceso de leer o escuchar un dato y retenerlo en la memoria el tiempo suficiente para tecleario. La memoria de corto plazo del usuario está limitada, generalmente a cinco o siete elementos. Por ejemplo, un número de 12 dígitos, es decir 12 elementos, normalmente excede la capacidad de la memoria de corto plazo de la mayoría de las personas. Esto provoca que algunos dígitos sean mal recordados o completamente olvidados.

Una forma de recordar secuencias largas de elementos, consiste en agruparlos en conjuntos de tamaño más manejable. Es asi como un número de larga distancia, que tiene diez digitos, es demasiado largo si no se agrupa en el código de lada, la clave de la ciudad y el número. Este método resulta altamente recomendable cuando el sistema requiere leer secuencias largas de caracteres o números, debe permitir que el usuario escriba pequeños conjuntos de elementos, pues asi le sera mas facil, recordar la cadena completa.

6) **Consistencia y Estandarización.**

Sin importar la longitud de los mensajes del sistema y el formato de comandos, ya sea menú, pregunta-respuesta o comandos directos, es esencial que observen una consistencia en su sintaxis. Es decir, los comandos que hacen lo mismo en distintas partes del sistema, deben ser los mismos. Así, por ejemplo, los comandos u opciones del menú, "FIN", "SALIDA" y "TERMINAR", generalmente significan lo mismo. Sin embargo, sólo uno de ellos puede usarse en todo el sistema. Por extensión, funciones que son completamente diferentes, no deben tener comandos iguales o aún similares, en distintas partes del sistema.

Debe procurarse que el sistema siempre envíe los mensajes de una forma similar, o en el mismo lugar. Tal y como se mencionó en la organización de la pantalla, del primer principio de los factores humanos.

Algunos sistemas son inconsistentes cuando piden al usuario que introduzca un dato. Esto sucede generalmente, cuando en casi todo el programa se espera que el usuario oprima una tecla, como "Return", para indicarle que el dato ha finalizado. Sin embargo, en otras secciones, el sistema no lo hace asi. Cuando el usuario termina de escribir el último caracter, o digito, por ejemplo del registro federal de causantes, automaticamente el sistema transita al siguiente dato, sin esperar que el usuario oprima "Return". Esto puede desconcertar al usuario y debe emplearse solo en contados casos.

2.1.3.3 Tercer Principio De Los Factores Humanos.

"EL SISTEMA DEBE CONCORDAR CON EL MODELO INTERNO DEL USUARIO"

No solo las caracteristicas del sistema deben hacer juego con las caracteristicas del usuario, sino también, el sistema debe concordar con el modelo interno del usuario. Cada persona elabora un modelo interno del sistema. Este modelo, ayuda al usuario a decidir qué comandos debe ejecutar para realizar una acción especifica. De ahí la importancia de la exactitud del modelo interno para operar el sistema correctamente. Asi pues, el modelo interno del sistema esta intrinsecamente relacionado con el conocimiento semántico del mismo.

1) **El Modelo Interno del Usuario.**

El modelo interno del usuario se desarrolla en base a la experiencia, por el uso del sistema en particular u otros sistemas semejantes, ayudado obviamente por los cursos y manuales. Sin embargo, frecuentemente el modelo interno del usuario radica en el entendimiento de un proceso natural. El cual se apoya en el empleo de conceptos familiares. Tal es el caso de una secretaria, a quien le es familiar abrir y cerrar archivos de empleados.

Será entonces, labor del diseñador del sistema, deducir cómo es el modelo interno esperado del usuario y diseñar el sistema lo más apegado al modelo. A pesar de todo, no es tan sencillo; para los usuarios improvisados y Principiantes, el diseñador del sistema puede suponer que el modelo interno del usuario no estará bien desarrollado. Por este motivo, el sistema debe diseñarse para que se presente en pequeños paquetes fáciles de entender. Y así, el usuario emplee sólo una pequeña parte del sistema para realizar una actividad completa. Con esta tónica, conforme su modelo interno se desarrolle, el usuario podrá emplear nuevas secciones del sistema.

Para los usuarios Competentes y Expertos, el diseñador puede esperar que tengan desarrollado un buen modelo interno. Por este motivo, el sistema podrá mostrarse como un paquete integrado, que el usuario podrá emplear con todas sus funciones. Esto implica que a diferencia del caso anterior, el sistema se diseñará para "navegar" fácilmente por todas sus funciones y no para entenderlas fácilmente.

2) **Manuales.**

Con respecto a la documentación, o manuales, se puede mencionar que son esenciales para ayudar al desarrollo interno del modelo del usuario (conocimiento semántico) y para proveer de los procedimientos para operar el sistema (conocimiento sintáctico). La documentación para usuarios improvisados y Principiantes debe contener información semantica intensa, mientras que la documentación para usuarios Competentes y Expertos debe tener menos información semántica y enfocar el material a simples recordatorios.

3) **Cursos.**

Otro elemento fundamental en el desarrollo del modelo interno del usuario son los cursos. Para los usuarios improvisados y Principiantes, los cursos deben concentrarse en desarrollar el modelo interno (conocimiento semantico). Esto se logra enseñando a realizar, poco a poco, funciones completas pequeñas, para después intentar funciones adicionales. Los cursos para los usuarios Competentes y Expertos no requieren de tanto esfuerzo, pues necesitan exclusivamente ajustar el modelo interno del usuario. Esto se logra estableciendo las diferencias entre el nuevo sistema y sistemas anteriores. El tiempo disponible se puede concentrar a impartir el conocimiento sintáctico para operar el sistema.

Como se puede apreciar hasta este punto, la selección del nivel del usuario de acuerdo a sus características congnoscitivas, dicta la pauta en el diseño de sistemas. El desarrollo del modelo interno del usuario, está muy ligado a si el sistema debe ser "fácil de usar" o facil de entender", y esto eventualmente, dicta el tipo de formato de comandos.

2.1.3.4 Cuarto Principio De Los Factores Humanos.

"EL FLUJO OPERACIONAL DEL SISTEMA DEBE CONCORDAR CON EL PROCESO DE PENSAMIENTO DEL USUARIO"

Hasta el momento se han expuesto criterios relacionados con las capacidades cognoscitivas del usuario y de las características que el sistema debe cumplir, si se desea que esté acorde con el usuario. Pero de la respuesta del sistema, otro elemento funcional de gran importancia, nada se ha mencionado. El cuarto principio dicta precisamente, lineamientos básicos para este elemento.

Cuando el usuario desea resolver un problema, desarrolla un modelo mental correspondiente a los pasos que lo solucionan. Si la operación del sistema no está de acuerdo al modelo de solución, interrumpirá frecuentemente al proceso pensante del usuario. Así es que, repetidas veces tendrá que detenerse a pensar, primero en el problema que desea solucionar y luego en cómo funciona el sistema. Esto crea distracción, y si es muy frecuente, termina por decepcionar al usuario. Es por esto, que la secuencia de operación del sistema debe concordar lo más posible con la secuencia natural de solución del problema del usuario. Tal es el caso del mecanismo de introducción de datos para almacenar una forma. Las formas son generalmente documentos que tienen una serie de espacios, que el usuario debe llenar a mano. De estas formas se extrae información, la cual se almacena en la computadora y posteriormente se procesa. Ejemplos clásicos son las formas para el censo poblacional o para la declaración de impuestos. El mecanismo de captura, o introducción de datos de la forma, debe vigilar que la secuencia de datos pedida por el sistema, tenga la misma secuencia que los datos originales de la forma. Lo cual evita que el usuario tenga que buscar constantemente los datos por toda la forma y así, se disminuye la posibilidad de error.

Por otro lado, la respuesta del sistema debe adecuarse al ritmo mental del usuario. Un usuario es capaz de aplicar toda su atención al problema hasta llegar a un punto de terminación. La terminación se entiende como el estado en el cual se cubre un objetivo. Al llegar a ese punto de terminación, el usuario puede liberar la atención del sistema. Teniendo esto en mente, un usuario puede tolerar retardos, inherentes al tiempo de respuesta del sistema, hasta de un segundo sin perder la atención. Es mas, a pesar de que los retardos de cinco segundos se registran como interrupciones definitivas, el usuario normalmente logra conservar la atención. Sin embargo, esto no es cierto para retardos mayores a diez segundos, en donde el flujo operacional del sistema rompe totalmente el flujo natural del usuario. Cuando esto sucede, se dice que ha ocurrido una interrupción antes de un punto de terminación normal, lo cual requiere de un esfuerzo mental adicional, por parte del usuario, para retomar el problema.

Es responsabilidad del diseñador, cuando realiza el análisis de las funciones del sistema, fijar el alcance del poder de atención del usuario, identificar los puntos de terminación y estructurar el sistema para evitar distraer la atención del usuario.

Algunas veces no es posible evitar los retardos inherentes del sistema. Bajo estas circunstancias, el sistema deberá informar al usuario, si la respuesta tardará más de diez segundos y deberá proporcionar, si es posible, una estimación de la duración del retardo. Esto permite al usuario fijar la atención en cualquier otra cosa, sin vigilar constantemente el sistema. Es imperativo entonces, que el sistema informe al usuario también, cuando ha finalizado el retardo.

En otras ocasiones, el sistema debe interrumpir forzosamente el flujo operacional, pues se requiere que el usuario tome una acción en particular. Esto es, cuando un dato no es válido y el usuario debe corregirlo, o bien, cuando existe un error del sistema, por ejemplo, en el momento en que el usuario desea cargar a memoria un archivo no existente.

No importa si el motivo es la terminación de un retardo, o si ha ocurrido un error del sistema, la forma como el sistema debe llamar la atención del usuario, depende de la manera en cómo el usuario interactúa con la pantalla. Si el usuario tiene toda su atención a la pantalla, entonces, un simple mensaje de error o aviso es suficiente. Pero si el usuario mira ocasionalmente el video, las letras del mensaje, más brillantes, intermitentes o de distinto color, son apropiadas para llamar su atención. Cuando el usuario opera el sistema en forma automática (usuario Experto), una alarma sonora es el único medio capaz de interrumpir la atención del usuario. También hay que tomar en cuenta, que la manera en como se llama la atención del usuario, depende del estado normal de la pantalla. La señal audible, que podría ser una campana, no es adecuada si el programa emite bajo condiciones normales, tonos audibles. Algo similar sucede con mensajes intermitentes, en distinto color ó brillantez. Resumiendo, las señales de alarma debe elegirse de acuerdo a estados excepcionales del programa, para que sean factores que llamen efectivamente la atención del usuario.

2.1.3.5 Quinto Principio De Los Factores Humanos.

"EL SISTEMA DEBE EVITAR INCOMODIDADES FISICAS AL USUARIO"

Finalmente, este factor trata los aspectos relacionados con el medio ambiente que rodea al usuario y especialmente enfoca su estudio a la terminal de trabajo. Existen dos aspectos principales, las malas posturas del cuerpo y la fatiga visual, que determinan la comodidad en una terminal.

1) **Las Malas Posturas Corporales.**

   Las malas posturas provocan tensión muscular y eventualmente causan fatiga e incomodidad. También, emplear mobiliario inadecuado propicia una deficiente circulación de la sangre, que en última instancia se traduce en cansancio. La comodidad física se logra con equipo y

mobiliario ajustables a las necesidades del usuario. La silla de trabajo debe permitir ajustar la altura del asiento, del respaldo y la inclinación de la espalda. Pero jamás de tal forma que los pies estén despegados del suelo. Algunas sillas solucionan esto, proporcionando un soporte especial para descansar los pies. La altura de la mesa debe colocar al teclado en una posición natural para las manos del usuario. Los teclados no empotrados al video, permiten que el usuario se siente a una distancia conveniente del video y que adecúe la lejanía del teclado.

2) **La Fatiga Visual**

La fatiga visual se debe esencialmente al reflejo de fuentes de luz incidentes sobre el video. Existen tres métodos simples para reducir el reflejo de luz:

a) Emplear un video ajustable que permita fijar su inclinación horizontal. Esto evita al máximo la luz incidental reflejada.

b) Usar un recubrimiento antireflejante sobre la pantalla. Esto permite reducir notablemente la fatiga visual. Sin embargo, debe cuidarse de mantener el contraste adecuado entre la intensidad de los caracteres y el resto de la pantalla.

c) Orientar las luces tomando en cuenta la posición del video, tratando que no se refleje luz sobre la pantalla. La línea de vista de un usuario, que mira directamente el centro del monitor, puede tener una inclinación entre 15 y 20 grados abajo de la horizontal. La pantalla debe colocarse a una distancia de 40 a 60 cm del ojo, medida desde el ojo del usuario a lo largo de la línea de vista. Las luces incidentes, deben desviarse más allá de 20 grados de la línea de vista. Mas información, relativa a medidas de mobiliario y colocación del equipo, se pueden encontrar en la referencia [10].

Cuando se tienen largos períodos de trabajo, es fundamental considerar lapsos de reposo que permitan relajar el cuerpo. Se recomienda descansar cinco minutos por cada hora, durante largas jornadas de trabajo.

2.1.4 Resumen

El diseñador de sistemas que desea incorporar buenos Factores Humanos debe, primero, considerar el número de personas que emplearán el programa y el tipo de usuario desde el punto de vista congnoscitivo, experiencia, etc. Posteriormente, requiere determinar la secuencia de pasos para operar el programa y finalmente, en base a las características del usuario y del sistema

arrojadas en los dos primeros pasos, intentar incluir en el diseño cada uno de los principios de los Factores Humanos.

Es importante tomar en cuenta, que el proceso de incorporación de Factores Humanos no es riguroso, ni exhaustivo. Es decir, que no siempre es posible aplicar todos los principios de los Factores Humanos. Queda pues, al criterio el diseñador, decidir cuándo es técnicamente factible aplicar un factor.

En esta sección se han delineado los criterios más importantes, que dictan las pautas para el diseño de una interfaz con el usuario. El siguiente capítulo describirá la aplicación práctica de estos principios en el diseño del sistema que motiva a la tesis. A continuación se tratará el tema relacionado con el diseño de la parte gráfica de un sistema.

2.2 RUTINAS GRAFICAS.

En esta sección se describirán los aspectos principales que guían el diseño de la parte gráfica, o rutinas gráficas, de un sistema. También se mencionarán la clase de rutinas que generalmente lo integran. En los siguientes capítulos se podrá observar que de esta rúbrica se desprenden criterios importantes para el diseño de la sección gráfica del Sistema Emisor de Reportes Gráficos, proyecto de esta tesis.

La sección gráfica está integrada por una serie de rutinas o funciones que un programa de aplicación emplea para generar gráficas, o figuras, en un despliegue. Este despliegue puede ser una pantalla de un monitor, un graficador, etc. No siempre un sistema gráfico consta de una interfaz y de las rutinas gráficas; algunas veces consiste exclusivamente de las rutinas en forma de un paquete gráfico; es decir, carece de una interfaz formal con el usuario, pues su objetivo es sencillamente, facilitar la realización de aplicaciones con salidas gráficas.

No importa que el sistema gráfico tenga o no interfaz con el usuario, las rutinas gráficas, deben siempre observar una serie de reglas.

2.2.1 Reglas Para El Diseño De Las Rutinas Gráficas.

Son seis las reglas que un paquete debe cumplir, para considerarse con la solidez suficiente para diseñar aplicaciones basadas en él. Estas reglas se listan a continuación.

1) **Simplicidad.**- Es importante tratar de evitar modalidades complicadas del paquete que el diseñador del programa de aplicación, es decir el usuario, no pueda entender fácilmente. Sin embargo, para el diseñador del paquete gráfico no es sencillo detectar las modalidades complicadas de su paquete, pues al fin y al cabo, es el creador y no puede evitar apreciarlo con cierto paternalismo. Una práctica sana indica escribir el manual de uso antes de diseñar el paquete gráfico y si algo es difícil de explicar cómo funciona, seguramente será difícil de usar.

2) **Consistencia.**- Un sistema consistente es aquel que se comporta de una forma predecible. Esto es, los nombres de las rutinas, la secuencia en que hay que ejecutarlas, la forma en que detectan errores, así como los sistemas coordenados que emplean, siguen patrones constantes a lo largo de todo el paquete.

3) **Completo.**- El diseñador del paquete debe tratar de no omitir rutinas. Cualquier rutina faltante, deberá suplirse y programarse por el usuario, lo cual no siempre es sencillo, ya que frecuentemente el usuario no tiene acceso a las estructuras internas del paquete gráfico. Esto no significa que el diseñador debe dedicarse a realizar indiscriminadamente cualquier rutina gráfica que le cruce por la mente, sino que debe elaborar el menor conjunto posible de rutinas útiles, para el campo de aplicación que se ha trazado.

4) **Robusticidad.**- Es muy frecuente que el usuario emplee incorrectamente el paquete gráfico, y mas aún, es probable que lo use sin leer a consciencia previamente el manual de uso. Cuando el paquete está debilmente programado, puede llegar a terminar su ejecución por error si se usa incorrectamente. Es recomendable entonces, prever errores triviales y comunes de uso. Algunas veces el propio paquete puede pasar por alto estos errores, estimando cuál debe ser el dato correcto. En el caso de que el usuario realice algo verdaderamente grave, entonces, el paquete debe enviar un mensaje de error lo más claro y útil posible. Obviamente no se podrá evitar que bajo condiciones extremas de error, el paquete grafico suspenda su ejecución.

5) **Respuesta.**- La velocidad del comportamiento del paquete debe ser razonable. Sin embargo, casi siempre, la velocidad está sujeta a la respuesta del sistema operativo y a las características de las unidades de despiegado. Un buen paquete gráfico debe evitar que los usuarios tengan que emplear trucos de programación, burlando al paquete, para agilizar su aplicación.

6) **Economía.**- El paquete gráfico, debe tratar de economizar al máximo los recursos de la computadora. Como son el espacio en disco, tiempo de procesador, etc. de tal suerte, que sea rentable generar aplicaciones a partir del paquete gráfico.

Algunas veces, técnicamente no es posible apegarse a todas las reglas, pero tenerlas en mente cuando se realiza el diseño del paquete, proporciona buenos resultados. La sección más difícil de cumplir es que el paquete sea completo, pues es común que el diseñador realice rutinas poco útiles. Y aún así, termina por no quedar satisfecho y dudoso de haber cubierto todas las posibles necesidades del usuario. Afortunadamente, existe una técnica para realizar un paquete gráfico lo más completo posible, la cual consiste en construirlo a partir de una serie de conjuntos de rutinas. Cada conjunto se orienta a un tipo particular de función. Dependiendo de la aplicación, se determina la clase de las funciones, así como el tipo y el número de rutinas de cada conjunto.

Los tres conjuntos básicos de todo paquete gráfico son:

1) **Primitivas Gráficas.**- Se emplean para desplegar líneas rectas, textos de caracteres, arcos circulares y otros elementos simples de este estilo.

2) **Funciones de Ventaneo.**- Permiten seleccionar un sistema coordenado, de acuerdo al tamaño de la figura que se dibuja. También indican si la gráfica se realiza en sólo una fracción de la pantalla.

3) **Funciones Adicionales.**- Contemplan las rutinas encargadas del manejo de los dispositivos gráficos y del reporte de estados.

2.2.2 Primitivas Gráficas.

Algunas veces los dispositivos gráficos, como terminales, graficadores, etc.; tienen interconstruidos electrónicamente una serie de primitivas gráficas, por ejemplo, trazo de puntos, líneas, arcos circulares, trazo de figuras geométricas simples y otras de este tipo. Esto ahorra esfuerzo de programación en la elaboración del paquete. Sin embargo, no importa si los dispositivos gráficos tienen o no interconstruidas las primitivas gráficas, el paquete gráfico debe proporcionarlas.

Existen tres primitivas gráficas a partir de las cuales todas las demás se pueden derivar. Estas primitivas se describen a continuación.

1) MueveA (x,y) Coloca el cursor en la posición (x,y) para desde ahí, dibujar una línea o texto.

2) DibujaA (x,y) Dibuja una línea recta, de la posición donde se encuentra el cursor, al punto (x,y); deja en ese punto el cursor.

3) Dibuja Texto (t) Dibuja el texto "t", a partir de donde se encuentra el cursor. Deja el cursor en la esquina inferior derecha de la última letra del texto.

Dependiendo del paquete gráfico y del lenguaje de programación, seguramente cambian el nombre de las tres primitivas y el orden de los parámetros, pero es muy probable que el paquete proporcione funciones similares.

Es común encontrar que los paquetes gráficos proporcionen primitivas gráficas más elaboradas, como funciones para trazo de líneas de distinto ancho y tipo, para rellenar polígonos, para trazo de figuras geométricas simples, etc. Sin embargo, como se mencionó previamente, todas se pueden elaborar con la combinación de las tres primitivas gráficas antes mencionadas.

2.2.3 Funciones De Ventaneo.

Las funciones de ventaneo son fundamentalmente dos, aunque algunos autores las integran en una sola función. Para respetar la claridad y las convenciones de la referencia [2], se manejan ambas. La lista es la siguiente.

1) DefVentana (vxmin, vymin, vxmax, vymin) Define una ventana, esto es, los valores mínimos y máximos que acotarán la gráfica.

2) DefZona (zxmin, zymin, zxmax, zymax) Define las dimensiones de la sección de la pantalla, que se empleará para dibujar.

2.2.4 Funciones Adicionales

Generalmente todo paquete gráfico cuenta con estas funciones, sobre todo si el paquete es independiente del dispositivo. Esto es, sólo algunas rutinas se diseñan para estar intrínsecamente ligadas al dispositivo gráfico y las demás para dibujar, hacen simples llamadas a estas rutinas. Por este hecho, es relativamente sencillo adaptar el paquete a otros dispositivos gráficos.

1) IniciaGraf Se llama al principio del programa de aplicación. Borra la pantalla o prepara una nueva hoja en el graficador. También inicializa parámetros internos del paquete.

2) Determina (v) Regresa, en un vector por ejemplo, las características de la terminal o dispositivo gráfico como dimensión máxima en x de la pantalla o papel, dimensión máxima en y, ancho de los caracteres de texto, alto de los caracteres de texto, etc.

Este tema no pretende ser una descripción de la parte gráfica del proyecto, ya que el paquete gráfico del sistema es mucho más extenso que el aquí se mostró. Simplemente, el objetivo de este punto, fué detallar los criterios para el diseño de un paquete gráfico. Estos se usaron como fundamento para estructurar el paquete gráfico del proyecto. En los siguientes capítulos se mencionarán los aspectos prácticos de estos lineamientos, involucrados en el diseño de la sección gráfica del sistema.

CAPITULO 3
CARACTERISTICAS DEL SISTEMA

Dentro de las distintas etapas por las que pasó el sistema, una de las más extensas y complejas, fué la dedicada a la definición de sus características, alcances y limitaciones. Su complejidad estribó principalmente en que a diferencia de la mayoría de los sistemas, no existía una necesidad específica y un objetivo a cumplir, así que fué necesario realizar un análisis de los problemas de los usuarios con los paquetes gráficos ya existentes y evaluar sus necesidades actuales tanto para actividades académicas como para actividades de apoyo administrativo.

Las principales actividades realizadas en este estudio y que formaron los criterios que delimitaron la definición del sistema son: entrevistas a profesores, alumnos y personal del Centro de Cálculo, evaluación de paquetes comerciales y recursos de cómputo disponibles.

Una vez reunida toda ésta información, se definieron todas las características del sistema en cuanto a las gráficas disponibles y sus distintas opciones de presentación, así como las formas y filosofías de uso.

Por su importancia, en este capítulo se detallan las principales actividades realizadas, encaminadas a la definición del sistema.

3.1 CRITERIOS EN LA DEFINICION DEL SISTEMA

Los principales criterios que se tomaron en cuenta para la definición del sistema se detallan en esta sección. Las entrevistas a usuarios y las necesidades actuales delimitaron el tipo de sistema requerido, la filosofía de comunicación con el usuario (interfaz) y modalidades de uso se definieron a partir de las entrevistas y evaluación de paquetes comerciales y las características técnicas se definieron a partir de los recursos de cómputo disponibles.

A continuación se resumen las actividades y las conclusiones obtenidas en este estudio.

3.1.1 Entrevistas A Usuarios Potenciales

Las entrevistas se realizaron a los tres tipos de usuarios potenciales del sistema : Personal académico, alumnos y personal de desarrollo de sistemas del Centro de Cálculo.

Debido a que los profesores son las personas que tienen un mayor conocimiento de los problemas y necesidades académicas actuales, fueron los primeros en ser entrevistados. Los profesores con quienes se habló pertenecen a la División de Ciencias Básicas, División Mecánica y Eléctrica y del Centro de Cálculo que están relacionados con materias estadísticas, matemáticas o que requieren de generación de reportes gráficos.

La entrevista contempló algunos aspectos académicos, así como de la presentación de las gráficas de los sistemas ya existentes en el CECAFI y de la forma de uso del sistema a definir. A continuación se resumen los comentarios recibidos que fueron muy satisfactorios :

1) Las gráficas propuestas deben contemplar las opciones de gráficas estadísticas más usuales.

2) Que exista la posibilidad de obtener los reportes gráficos en forma interactiva y a través de llamadas a rutinas.

3) Graficas en 3 dimensiones para aquellas materias de matemáticas básicas.

4) El tratamiento matemática propuesto es el adecuado.

En cuanto a los alumnos y personal del centro, se entrevistó principalmente a las personas que han utilizado los sistemas gráficos ya existentes a disposición de los usuarios. Los comentarios recibidos fueron los siguientes :

1) Mayor flexibilidad para la captura de datos.

2) Un sistema integral que contemple las graficas estadisticas y matemáticas para tener la opción de dibujar un mismo grupo de datos en distintos reportes.

3) Opción de poder llamar al sistema a través de un lenguaje de alto nivel.

3.1.2 Evaluación De Paquetes Gráficos

Con la finalidad de tener un criterio mas amplio con respecto a la definición del sistema se estudiaron algunos paquetes comerciales para distintos equipos de cómputo. Se evaluaron los siguientes aspectos:

1) Interfaz con el usuario. Forma de mostrar las distintas opciones, la manera de introducir y de modificar los datos, las ayudas, la realimentacion al usuario, etcétera.

2) Facilidad de uso.

3) Las gráficas que permiten obtener.

4) Las opciones que modifican la presentación del reporte.

Se estudiaron paquetes con las distintas filosofías que ya se mencionaron en el capítulo 1. Los paquetes estudiados fueron principalmente para microcomputadoras y minicomputadora. Sin embargo, para "mainframes" o minis existen muy pocos sistemas gráficos o sistemas de aplicación con salida gráfica en la UNAM.

El resumen de los sistemas evaluados es :

1) **SUPERCALC.** Hoja de cálculo con salida gráfica.

La interfaz es a través de menús y permite una grán facilidad para la modificación de los datos. Con un pequeño período de entrenamiento, el sistema es muy fácil de usar.

Las gráficas que se pueden obtener son de tipo estadístico y de X vs Y. Tiene una grán cantidad de opciones de presentación, tales como : Distintos tamaños de hoja, hasta cuatro gráficas por hoja, titulos y formatos, entre otras.

Una característica importante es que proporciona una serie de valores iniciales (default) con lo que el usuario puede obtener una gráfica sin necesidad de conocer todas las opciones y sin tener que proporcionar una gran cantidad de datos.

Las ayudas que presenta al usuario son muy claras y completas por lo que sin necesidad de manual se puede operar el sistema.

2) **ENERGRAPHICS.** Programa de aplicación gráfica

Su manejo es a través de menús y pantallas de captura. La forma de presentación es muy sencilla por lo que sin un periodo de entrenamiento es posible operarlo. Sin embargo, no presenta en todos los casos valores "default" por lo que en algunos casos es necesario proporcionar muchos datos.

Es un paquete muy completo ya que proporciona gráficas estadísticas, gráficas matemáticas en dos y tres dimensiones con varias opciones de tratamiento matemático, diseño gráfico en dos y tres dimensiones, presentacion de transparencias y evaluación de funciones matemáticas.

No contempla la opción de ayudas al usuario.

3) **AUTOCAD.** Programa de aplicación de diseño gráfico.

Puesto que es un paquete muy completo, su uso requiere un periodo de entrenamiento más o menos largo, aún cuando presenta ayudas y las opciones son a través de menús o comandos.

Permite diseño gráfico en dos y tres dimensiones. Contempla una gran variedad de opciones todas ellas ligadas al diseño gráfico y la presentación en el graficador.

Algo importante de comentar es el hecho de que son necesarios dispositivos gráficos de alta calidad para lograr un amplio aprovechamiento del paquete. Dichos dispositivos son: Un digitalizador y/o un "Mouse" para la captura de datos, un video de alta resolución para una buena presentación y un graficador de preferencia de colores para obtener reportes de alta calidad. Asimismo es recomendable tener una importante cantidad de memoria principal y de memoria masiva (de preferencia disco duro) para lograr una mayor velocidad de operación.

4) MICROSOFT WINDOWS PAINT. Diseño artístico.

Muy simple de usar ya que presenta las opciones en forma gráfica y grandes ayudas al usuario. Al igual que AUTOCAD necesita de dispositivos gráficos con excepción del digitalizador que no lo soporta y el graficador se sustituye por una impresora de alta calidad.

5) PRODUCTOS INTERGRAPH. Paquetes para el diseño en minicomputadoras.

Son paquetes con aplicaciones muy profesionales que requieren de un equipo muy especializado y sumamente costoso. Son empleados en el diseño mecánico, arquitectónico e ingenieril.

6) PGPLOT y PLXY. Paquetes de rutinas gráficas de aplicación general.

Son paquetes existentes en la VAX 11/780 de la Facultad de Ingeniería que proporcionan una serie de primitivas tales como: Trazo de líneas, de textos y símbolos, de ejes coordenados, de funciones a partir de parejas X-Y, así como definición del archivo de salida, entre otras.

A partir de estas primitivas se pueden desarrollar programas con salidas gráficas de acuerdo a la aplicación deseada.

En cuanto a los dispositivos requeridos solo es necesario una impresora-graficadora Printronix la cual es manejada por los dos paquetes. PGPLOT proporciona además otros dispositivos tanto de video como de impresión.

7) REGIS. Editor gráfico.

Es un editor de primitivas gráficas (trazo de líneas, caracteres, símbolos, círculos, manejo de colores, tipos de líneas, etcétera) para una terminal gráfica de la VAX y a través de vaciados de pantalla a un impresora-graficadora se obtienen reportes impresos.

Contempla ayudas para el usuario.

Una gran desventaja que presenta es que el equipo que requiere es obsoleto y escaso para la comunidad de la facultad.

3.1.3 Necesidades Actuales

Un criterio muy importante para la definición del sistema, fué considerar los requerimientos que existen actualmente entre los usuarios potenciales, los cuales se determinaron a través de un análisis de las necesidades de los alumnos en las materias y que frecuentemente son expresadas en la asesoría que presta el CECAFI así como de las necesidades que existen en él. Una síntesis de ellas se presenta a continuación :

3.1.3.1 CECAFI

-Presentación de reportes gráficos de los resultados obtenidos en sistemas de aplicación. Como ejemplo se pueden mencionar:

1) Análisis estadísticos de aprovechamiento académico de alumnos.

2) Demanda y uso de recursos.

3) Sistema de personal.

4) Estadísticas de inscripciones, encuestas, horarios.

Asimismo frecuentemente se realizan análisis comparativos, distribución de recursos, entre otros, que tendrían una mejor presentación por medio de gráficas que demuestren los resultados obtenidos.

3.1.3.2 Alumnos

Es muy frecuente que los alumnos requieran un reporte gráfico para presentarlo en alguna materia. los casos más frecuentes son:

1) Reportes de práctica de laboratorio.

2) Estudio del comportamiento de ecuaciones matemáticas que representan algún fenómeno(sistemas, simulaciones).

3) Representación de ecuaciones matemáticas de materias de análisis matemático (Ecuaciones Diferenciales, Cálculo, Algebra, etcétera).

4) Reportes estadísticos de análisis de comportamiento de distintos sistemas o métodos (Métodos de ordenamiento, de búsqueda, Control de calidad, Análisis de muestras).

3.1.4 Recursos Disponibles

El Centro de Cálculo cuenta con varios equipos de cómputo a servicio de su personal, de los alumnos y profesores. Sin embargo, el equipo que presenta una mayor cantidad de terminales de trabajo, con una gran posibilidad de atender a la mayoría de los miembros de la facultad, así como contar con un dispositivo de salida gráfica impresa, es el equipo VAX, cuyas características ya se mencionaron.

3.2 CLASIFICACION DEL SER

En base a las principales necesidades que se mencionaron anteriormente el sistema deberá básicamente contemplar las siguientes opciones:

1) Reportes gráficos de tipo matemático y estadístico.

2) Opción de uso interactivo y a través de rutinas.

3) Tratamiento matemático (evaluación de funciones).

Si se quisiera situar al sistema SER en alguna de las clasificaciones antes expuestas se concluiría lo siguiente:

SER es un sistema que se comunica con el usuario de dos formas:

1) **interactivamente**, por medio de una interfaz que hace uso de una terminal de video. Sin embargo las gráficas no aparecen en la pantalla sino en papel. Tal característica no le permite clasificarse como un paquete netamente interactivo.

2) **En base a rutinas gráficas**, el usuario puede crear su propia aplicación desde un lenguaje de programación y hacer llamadas a las rutinas gráficas que integran al sistema SER. En este caso sí se puede considerar a SER como una biblioteca de rutinas gráficas.

Teniendo en cuenta el tipo de gráficas que puede obtener SER, este sistema se puede ubicar dentro de dos clasificaciones. Como un sistema de gráficas para los negocios y estadísticas al igual que como un sistema de gráficas matemáticas de dos dimensiones.

3.3 DEFINICION DEL SER

A continuación se definen las características del sistema a partir del análisis expresado anteriormente. Se indica la forma de uso, los tipos de reportes y sus distintas presentaciones y la comunicación del sistema con el usuario.

3.3.1 Generalidades

El sistema tiene la finalidad de proporcionar un mecanismo para la obtención de gráficas de datos manejados comunmente en las áreas estadísticas, financieras e ingenieriles.

El diseño del sistema gira en torno al usuario. En todo momento se piensa ofrecer un sistema de fácil uso, evitando conversaciones largas con el sistema o llamadas complejas a rutinas. Esto se ve claramente en los lineamientos que siguen la interfaz con el usuario y las rutinas de biblioteca que lo integran y que estarán a disposición del usuario.

El sistema no es un paquete estadístico, financiero o de análisis de ingeniería. Mas bien, es una sofisticada interfaz gráfica fácilmente adaptable dichos paquetes y que permite una rápida representación gráfica de datos provenientes de distintas fuentes.

3.3.1.1 Uso Del Sistema

El sistema contará con dos modalidades de uso: por medio de un lenguaje de alto nivel (a través de rutinas) y como un sistema interactivo.

La finalidad de poder ejecutar el sistema por medio de un lenguaje de alto nivel por medio de llamadas a rutinas, es permitir que las aplicaciones generen directamente su salida gráfica sin la necesidad de llevarse a cabo pasos intermedios. Además de que el usuario llamará solo a las rutinas que considere necesarias para obtener la gráfica de su agrado.

No todas las rutinas que conforman al sistema estarán a disposición del usuario. Solamente aquellas rutinas generales que manejen directamente a los reportes gráficos o fundamentales para la obtención de éstos, podrán ser llamadas desde el programa de aplicación particular.

Existen restricciones evidentes en la secuencia de llamada a las rutinas. Por naturaleza propia de las mismas, no es posible llamarlas en forma aleatoria, implícitamente tiene un orden que debe ser respetado.

Concientes de que una gran cantidad de usuarios potenciales no cuentan con los conocimientos necesarios y tampoco disponen del tiempo para realizar un programa con el cual obtengan las graficas de sus datos, se proporcionara además un sistema interactivo, a través del cual será posible obtener cualquier tipo de gráfica contemplada dentro del catálogo del sistema. Este sistema interactivo contará con una interfaz hacia el usuario, cuya principal característica será evitar conversaciones prolongadas y así obtener rápidamente las gráficas deseadas. Esto a su vez, permitirá que el usuario en poco tiempo adecúe la presentación de sus gráficas de acuerdo a sus necesidades particulares.

3.3.1.2 Usuarios Potenciales

Los usuarios potenciales son todos aquellos que requieran de una herramienta gráfica para el análisis de sus datos. En general existen dos clases de usuarios:

Aquellos que no deseen realizar programas de aplicación o no tienen la disposición para realizar programas completos y usarán el sistema a través de la interfaz interactiva, y aquellos que emplearán el sistema desde un programa de aplicación con llamadas a las rutinas.

Al primer grupo pertenecen parte de los alumnos de ingeniería, la gran mayoría de profesionistas de otras áreas y en general todos aquellos que requieran esporádicamente de una representación gráfica de datos.

El segundo grupo está integrado por aquellos usuarios cuyas aplicaciones son de uso frecuente y que resultaría más tardado usar el sistema a través de la interfaz interactiva que llamando directamente a las rutinas.

3.3.1.3 Aplicaciones

Las áreas de aplicación del sistema son tan variadas como la naturaleza de la procedencia de los datos que generan a las gráficas. Es por esto que el tema de la aplicación no es tan relevante para la producción de las gráficas como el tipo de gráfica elegida para la representación de los datos. De aquí se desprende la importancia de los tipos de reportes gráficos de los cuales se habla mas profundamente en puntos posteriores.

3.3.2 Características Gráficas

En esta sección se indican los tipos de reportes gráficos que proporciona el sistema, así como las distintas formas de presentación que tendrán.

3.3.2.1 Presentación De Una Hoja

1) **Tamaño del papel**

El tamaño de la hoja de impresión quedará definido por el usuario y restringido únicamente por las dimensiones físicas del papel y aquellas que ponga el sistema gráfico utilizado. Sin embargo, para aprovechar al máximo el área de que se dispone en los diferentes formatos del papel se proporcionarán dos formatos preestablecidos: tamaño carta (11" x 8.5") y tamaño estandar CECAFI (8.5" x 15").

Además, se tiene la opción de indicar un desplazamiento del reporte gráfico dentro del tamaño de hoja establecido, ya sea a lo largo o ancho a manera de margenes de la hoja. Con ésto, el reporte podrá tener un área libre para engargolar, perforar o encuadernar.

2) Agrupamiento de gráficas

Existirán tres niveles de agrupamiento de la información a dibujar. El primer nivel (mas bajo) lo constituyen los grupos de datos denominadas tablas. Una tabla de datos es un arreglo de celdas que contiene información para cualquier tipo de reporte, pudiendo ser estos ordenadas, abscisas o rótulos.

El segundo nivel se formará con los grupos de datos que se dibujarán juntos para crear un reporte gráfico de algún tipo en especial. Finalmente los reportes gráficos se podrán agrupar para formar la hoja de dibujo, es decir una hoja de dibujo se puede subdividir con el fín de colocar en ella varios reportes gráficos. El número máximo de reportes por hoja son cuatro.

3) Recuadro

Una hoja de dibujo aparecerá enmarcada por un recuadro, fuera del cual se podrán colocar diferentes rótulos para identificar el reporte. En la figura 3.1 se presenta la forma en que se colocarán los rótulos:

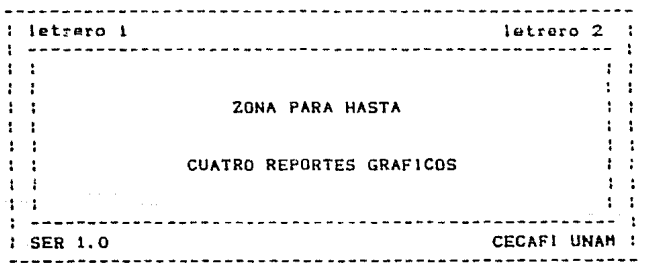


Figura 3-1

3.3.2.2 Tipos De Reporte Gráficos

1) Diagramas de barras

Estas gráficas presentan en un eje coordenado una serie de rectángulos (barras) cuya altura es proporcional al valor que tiene asociado. El eje de las abscisas es discreto y sólo tiene valores en los puntos señalados por el usuario, que es en donde aparecen las barras. Se aplican en los casos en los que se requiere conocer los valores asociados a una serie de elementos para apreciar sus diferencias. Las barras se diferencian entre ellas por medio de

diferentes patrones de dibujo.

2) **Diagramas de barras apiladas**

La presentación de estas gráficas es similar a la de los diagramas de barras con la salvedad de que el valor asociado a cada barra esta constituido por la suma de una serie de elementos , que perteneciendo a diferentes grupos de datos, se asocian a un mismo punto (abscisa). Una barra esta formada por diferentes patrones de dibujo para distinguir cada una de las partes que la conforman.

3) **Linea**

Tienen la misma finalidad de los diagramas de barras, sin embargo la presentación varía. En lugar de aparecer una barra para cada punto, los valores de las ordenadas se unen entre ellos para formar una linea que representa el comportamiento de los datos. Es preferible el uso de estas gráficas en lugar de las de barras cuando los elementos del grupo de datos siguen una secuencia cronológica, es decir, el eje de las X es el tiempo o simplemente los puntos discretos a lo largo de este eje siguen una secuencia rigurosa.

4) **Histogramas**

Son gráficas que se presentan sobre un eje cordenado XY. El eje X está dividido en intervalos definidos por el usuario llamados intervalos de clase. Sobre cada intervalo se coloca una barra cuya altura representa el número de datos que se encontraron en tal intervalo de clase. Su aplicación es netamente estadística y se utilizan para conocer la distribución de probabilidad que poseen los datos.

5) **Polígonos de frecuencias**

Tienen la misma aplicación que los histogramas con la única diferencia que en lugar de que aparezca una barra en cada intervalo, los puntos medios de los intervalos se unen entre ellos, con lineas, a la altura de la frecuencia correspondiente.

6) **X vs Y**

Sobre un eje coordenado XY se trazan puntos (x,y) para formar una curva. A diferencia de todas las demás gráficas, en esta modalidad el eje X es un eje continuo (todos los puntos del eje X están definidos). Su uso principal es el de dibujar puntos provenientes de la evaluación de una función o datos empiricos tomados en un experimento de laboratorio de los cuales nos interesa conocer su comportamiento.

7) **Diagramas circulares**

Estos diagramas también llamados gráficas de "Pie" se presentan como un círculo (Pie) dividido en secciones (rebanadas) las cuales representan un porcentaje del total del círculo que es el 100%. Su utilidad es mostrar en qué porcentaje contribuyen cada uno de los

elementos del grupo de datos para formar un 100% que es la suma de los valores de cada uno de los elementos. Cada una de las rebanadas se rellena con diferentes patrones de dibujo.

8) **Gráficas de Error**

La finalidad de estas gráficas es presentar, para una serie de puntos discretos (considerados como valores nominales) las variaciones (por encima o por debajo del valor nominal), que podrían sufrir tales puntos. Físicamente la gráfica se muestra como una curva continua y para los puntos que se desean analizar se muestran sus valores nominal, mínimo y máximo por medio de un segmento de recta que toca los tres valores antes mencionados.

9) **Gráficas combinadas**

Frecuentemente es necesario comparar grupos de datos representados con diferentes tipos de gráficas dentro de un mismo reporte, por esta razón el sistema proporcionará algunas combinaciones de dos diferentes gráficas de las antes mencionadas.

A continuación se enlistan las combinaciones que se ofrecen :

a) **Barras - Linea**

b) **Barras - X vs Y**

c) **Linea - X vs Y**

d) **Histogramas - Polígonos de frecuencia**

e) **Histogramas - X vs Y**

f) **Polígonos de frecuencia - X vs Y**

3.3.2.3 Presentación De Los Reportes Gráficos

3.3.2.3.1 Información Particular A Cada Reporte

1) **Diagramas de Barras y Linea**

Requieren como mínimo un solo grupo de datos, el cual representará las ordenadas de la gráfica (altura de las barras). Dicho grupo de datos debe contener elementos numéricos únicamente. Si se desean colocar etiquetas a lo largo del eje X para identificar cada una de las barras, se requiere asignar al reporte otro grupo de datos que se tomará como cadenas de carácteres. Las etiquetas sobre el eje X se colocarán perpendicular o paralelamente al eje dependiendo de la longitud de las mismas. En caso de ser demasiado largas, se truncarán teniendo en cuenta que el área para las barras debe ser al menos del

50% del area total de dibujo.

No se tiene restricción en el número de barras que componen la gráfica, sin embargo no se asegura que si se desean dibujar más de 50, éstas se puedan apreciar y distinguir perfectamente unas de las otras. El número de grupos de ordenadas por gráfica queda restringido a 5.

2) **Diagramas de Barras Apiladas**

Requieren como mínimo dos grupos de ordenadas para que el reporte pueda crearse. Adicionalmente se puede asignar al reporte un grupo que contenga etiquetas a ser colocadas sobre el eje X para distinguir cada barra.

Para etiquetar cada uno de los grupos de ordenadas es posible asignar al reporte otro grupo de datos que contenga las cadenas de caracteres que corresponden a cada grupo.

3) **Histogramas y Polígonos de Frecuencia**

Existen dos modos de uso de estos tipos de reportes:

- o Muestra de datos: Antes de proceder a obtener la gráfica, el sistema se encarga de obtener las frecuencias a partir de un grupo de muestras y de los intervalos de clase señalados.

- o Frecuencias: En este caso se requiere un grupo con las frecuencias.

En Polígonos de Frecuencia es posible asignar varios grupos de frecuencias para que se dibujen en el mismo reporte por lo que se podrá asignar un grupo más con las etiquetas que distinguirán a cada grupo de frecuencias.

4) **X vs Y**

Este tipo de gráficas requiere como mínimo 2 grupos de datos: un grupo de ordenadas y un grupo de abscisas. Si se desean colocar varias gráficas en el mismo reporte es necesario especificar para cada una de ellas sus respectivos grupos de ordenadas y de abscisas.

Adicionalmente es posible indicarlos nombres con los cuales se desea identificar cada grupo de datos de la gráfica. Se pueden agrupar como máximo en el mismo reporte 5 gráficas.

Las gráficas X vs Y pueden aparecer en forma continua o discreta teniendo para esta última la posibilidad de suprimir valores (proceso automático) para evitar que la curva aparezca como un aglutinamiento de puntos.

5) **Diagramas Circulares**

Como mínimo es necesario asignar un grupo numérico de datos a este tipo de reporte. Los datos del grupo pueden estar expresados en porcentaje o en forma absoluta. Es importante pero no necesario que se asigne otro grupo con las etiquetas que distingan a cada una de las rebanadas del diagrama circular. Se permite máximo de 20 elementos. No es posible asignar varios grupos de valores para la misma gráfica.

6) **Gráficas de error**

Para obtener un reporte de este tipo se requieren como mínimo dos grupos de datos numéricos (ordenadas) y uno alfanumérico (abscisas). Por lo anterior se puede notar que el eje "X" es discreto y lo comparten todos los grupos.

Cada grupo se distingue por medio de un símbolo especial. El conjunto de puntos que se refieren a la misma abscisa de unen con una línea recta (paralela al eje vertical). Como opción adicional se pueden unir las ordenadas medias de cada grupo de puntos con la misma abscisa, por medio de una curva.

3.3.2.3.2 Marco De Referencia

Este punto trata básicamente de la presentación de los ejes y de un rectángulo con acotaciones cuya finalidad principal es establecer un marco de referencia para las gráficas que dentro de él se alojan.

Del marco de referencia se puede hablar sobre tres aspectos relevantes:

1) **FORMATOS**

Todos los reportes, con excepción de diagramas circulares, estarán enmarcados por un rectángulo. En los bordes izquierdo y derecho se graficarán las acotaciones para Y. En el borde inferior se trazarán las acotaciones de X.

Dentro del rectángulo se pueden trazar otro tipo de ayudas para la presentación del reporte. Existen dos tipos de formatos, los cuales se explican a continuación:

a) **Rejilla** .- Consiste en trazar una serie de marcas dentro del rectángulo de referencia con el propósito de hacer más fácil y clara la localización de puntos alojados dentro del área de dibujo. Son fundamentalmente trazos verticales y horizontales de líneas, puntos y guiones para formar pequeños rectángulos que integran una especie de rejilla.

b) **Líneas de referencia** .- Para este formato será posible trazar hasta cinco líneas horizontales y verticales de referencia. El usuario especificará los valores de las ordenadas o abscisas en los cuales se trazarán las referencias. Si algún valor cae fuera de la ventana de dibujo, el sistema simplemente no dibujará dicha línea de referencia.

Las líneas de referencia verticales sólamente será posible seleccionarlas para gráficas "X vs Y".

Observaciones Especiales:

- Para todos los formatos de rejilla y líneas de referencia se empleará una intensidad de línea distinta al que se usará para el trazo de las gráficas.
- No será posible mezclar algún tipo de rejilla con líneas de referencia.

2) ESCALAS

Sin importar el tipo de formato que se emplee, el marco de referencia será segmentado por medio de marcas "tick marks" las cuales serán acotadas en forma numérica o alfanumérica.

a) Modos de Escalamiento

El número de marcas a lo largo de los ejes dependerá fundamentalmente del tipo de gráfica que se dibuje.

Existen dos opciones para el incremento entre las marcas, **ESCALA AUTOMATICA** y **ESCALA MANUAL**. En escala automática el sistema elígirá el mejor incremento posible. El escalamiento manual es proporcionado por el usuario indicando el valor inicial y final con que se trazará el eje y con estos valores se obtendrá el incremento.

El sistema siempre realizará un recorte de aquellos puntos que rebasen los límites especificados por la cota inicial y la cota final.

Como el sistema es capaz de dibujar más de una gráfica en un solo rectángulo, el usuario deberá especificar todas las funciones que se dibujarán en él y el sistema tomará las acciones adecuadas dependiendo del modo activo de escalamiento.

b) Escalamiento Especial para el Eje Horizontal

Para todas las gráficas excepto "X vs Y", Diagramas Circulares, Histogramas y Polígonos de Frecuencia, no se realiza escalamiento en el eje X sino que se colocarán tantas marcas como número de elementos tengan los grupos de datos proporcionados. En cada marca se colocará un rótulo que corresponde a cada elemento.

c) Escalamiento Especial para el Eje Vertical

El sistema ofrece la posibilidad de tener dos escalas distintas para cada uno de los ejes verticales del rectángulo.

Básicamente consiste en indicar qué grupos de datos se considerarán para determinar la escala del eje izquierdo del rectángulo, y qué grupos se tomarán en cuenta para el eje derecho. O bien, si se tiene la opción de escala manual, indicar los valores inicial y final tanto para el borde derecho como para el izquierdo.

Esta opción implica que el usuario es capaz de dibujar más de una gráfica en el mismo rectángulo pero no con la misma escala vertical. Esto es, el usuario deberá especificar con qué escala vertical de las dos existentes desea dibujar cada grupo de datos.

d) **Escalas Iguales**

Generalmente el incremento escalar entre intervalos no es el mismo para ambos ejes. Esto tiene como consecuencia que figuras como círculos, elipses y en general toda figura geométrica en donde sus proporciones sean vitales para su fisonomía, aparezcan distorsionadas. Este desagradable efecto se puede evitar indicándole al sistema que el incremento entre intervalos en ambos ejes debe ser el mismo.

Esta opción solamente es válida para gráficas del tipo "X vs Y".

e) **Escalas Logarítmicas**

Usualmente se desea que el escalamiento de alguno de los ejes o ambos sea el logaritmo de los valores que se alojan en el eje. El usuario deberá indicar que desea un escalamiento logarítmico para cada uno de los ejes donde así lo quiera.

El escalamiento logarítmico es válido en el eje Y para las gráficas: de barras, línea, X vs Y, histogramas y polígonos de frecuencia. El escalamiento logarítmico es válido para el eje X solamente las graficas "X vs Y".

3) **ROTULOS**

Comunmente se agrega al marco de referencia una serie de rótulos con los cuales se identifican los ejes, a los intervalos no escalares y a ciertos valores puntuales de ciertas gráficas.

a) **Rótulo de los ejes**

El usuario especificará el texto con el cual se identificará a cada uno de los ejes. Es posible que el usuario no especifique alguno, en cuyo caso el sistema asumirá que el rótulo es un texto nulo.

Se pueden especificar tres rótulos: para el segmento horizontal inferior del rectángulo, para el eje Y del segmento izquierdo del rectángulo y para el eje Y del segmento derecho del rectángulo. Todos estos rótulos son centrados a lo largo del segmento. Su inclinación será cero grados con respecto a la horizontal, esto significa que para los rótulos de los ejes verticales se escribirán en forma descendente, una letra abajo de la otra.

b) **Rótulo de valores puntuales**

Como una opción de los diagramas de barras e histogramas se dibujarán los valores absolutos o porcentuales que corresponden a cada barra en la parte superior de ella.

c) Rótulo de marcas

Para todas las gráficas excepto "X vs Y", Histogramas y Polígonos de Frecuencia es posible rotular los intervalos con textos alfanuméricos. El usuario deberá especificar con qué grupo se rotularán dichas marcas. Si el número de elementos del grupo es menor al número de marcas, se asumen como textos nulos los restantes. En caso de que sea mayor se ignoran los sobrantes.

En diagramas circulares se rotulará cada rebanada con un elemento del grupo que especifique. Los rótulos se colocarán en una tabla en donde se indicará la letra y el patrón de dibujo que identifican a la rebanada. Dicha tabla se colocará a la derecha del diagrama.

d) **Rótulo de grupos**

Como se mencionó anteriormente es posible trazar varios grupos de datos en un mismo espacio grafico delimitado por un rectángulo de referencia. El mínimo número de grupos que es posible trazar es uno.

El usuario debe especificar los textos alfanuméricos que rotularán a cada uno de los grupos mencionados. Estos textos se alojarán en la zona de la hoja definida para este propósito.

Para los diagramas circulares solamente es posible tener un solo grupo de datos.

En todos los rótulos es posible emplear símbolos especiales tales como raíz cuadrada, integral, etcétera. Estos símbolos los proporciona el sistema PGPLOT en una tabla conocida como HERSHEY. Solamente se hará uso de algunos de estos símbolos para evitar opciones imprácticas al usuario.

Se contará también con la posibilidad de incluir letras griegas dentro de los rótulos, subíndices y superíndices.

Se emplearán los carácteres definidos en el ASCII estándar incluyendo las letras minúsculas.

3.3.3 Interfaz

Una de las secciones más importantes del sistema de reportes gráficos es sin duda, la correspondiente al manejo del diálogo usuario-sistema. La filosofía de la interfaz radica primordialmente en integrar un sistema "amigable" al usuario y que además no requiera de un gran enfuerzo para comprender qué hace. Existen una serie de factores humanos (**ergonómicos**) que son necesarios considerar para lograr que el sistema cumpla con dichas metas. Por esta razón ya fueron ampliamente expuestos en el capítulo dos.

En lo que resta del capítulo, se explican las características de los usuarios potenciales y las de la interfaz del sistema.

3.3.3.1 Características De Los Usuarios Potenciales

A partir del análisis efectuado con los usuarios, y sabiendo que los alumnos de la Facultad de Ingeniería podrán prescindir del sistema para desempeñar sus actividades académicas y que solamente en determinadas épocas del año se puede esperar un uso intenso del mismo, por un grupo selecto de usuarios que cursan materias relacionadas con las funciones del sistema (estadística, cálculo diferencial, etc). Se puede concluir que el usuario típico del sistema será del tipo **"CASUAL"**. Asimismo, es evidente que dichos usuarios son de tipo **"LIBRE"**, pues nada les impone tener que emplear el sistema.

Dado que la política que sigue el Centro de Cálculo para la reservación de tiempo de terminal impone un límite en el número de horas, podemos esperar que el tipo de usuario sea "INTERMITENTE". Por este motivo el sistema proveerá de un mecanismo para almacenar el estado del sistema y continuar trabajando con él más tarde. A esta modalidad la denominamos "bitácora". El usuario dispondrá de dos comandos: uno para "salvar" la bitácora y otro para "cargarla". Cuando se "salva" una bitácora, el sistema almacenará en disco todos los datos proporcionados hasta ese momento. Si el usuario emplea el comando de "carga" la bitácora, el sistema traerá de disco los datos contenidos en ella, y así permitirle al usuario reanudar su trabajo.

Otro factor humano importante de tomar en consideración consiste en la destreza que tiene el usuario para operar el sistema. Esperando usuarios "casuales", podemos afirmar que la gran mayoría de los usuarios que harán uso del sistema serán del tipo "principiante" y en algunos casos extremos llegarán a ser catalogados dentro del nivel de "novatos", los cuales son deficientes en sus niveles cognoscitivos semánticos y sintácticos, debemos diseñar un sistema "fácil de entender", mas que "fácil de usar". Esto implica que la selección de un lenguaje de comandos del tipo menús de opciones y diálogos pregunta-respuesta es fundamental.

El auxilio semántico será provisto a través del uso de texto explicativo (ayudas) en donde se porporcionará una visión de las funciones del sistema. Este texto estará relacionado con las opciones del menú que se desplieguen en

ese momento, esto es, sólo proporcionará ayudas para las opciones activas en un instante dado.

A continuación se explican las características principales de la interfaz. La interfaz será interactiva con el objeto de que el usuario pueda proporcionar y modificar los datos en línea, así como las características de las gráficas en el menor tiempo posible y sin necesidad de conocer todas las opciones del sistema como ocurriría si no fuera en línea.

Cabe aclarar que el sistema no será totalmente interactivo ya que los reportes no se mostrarán en la estación de trabajo sino en una dispositivo independiente (impresora PRINTRONIX).

3.3.3.2 Características De Uso

1) Las opciones del sistema se mostrarán con valores predefinidos (defaults) los cuales podrán ser modificados por el usuario de acuerdo a sus necesidades. Con esta característica se permitirá que el usuario no tenga que proporcionar muchos datos para obtener una gráfica ya que muchas opciones ya estarán definidas.

   A medida que el usuario tenga más experiencia y conozca el sistema podrá utilizar las diferentes opciones para obtener gráficas más elaboradas al ir modificando los valores predefinidos y al hacer uso de las distintas utilerías que presenta la interfaz.

   Sin embargo, existen algunos datos que forzosamente se deben proporcionar al sistema, como los datos a dibujar, el tipo de gráfica que se desea en el reporte, etcétera.

2) Para indicar las posibles acciones que puede realizar el usuario se utilizararán menús dinámicos. Estos menús mostrarán solo las opciones que en ese momento el usuario puede solicitar. Por ejemplo: el menú principal no mostrará la opción de borrar tablas si no se ha creado al menos una de ellas.

3) Es frecuente que el usuario quiera almacenar el estado de una sesión con el sistema, debido a diferentes razones como: terminación del tiempo reservado para uso de la terminal, obtención posterior de un reporte con características similares a la actual, poder realizar modificaciones a un reporte sin riesgo de que se pierdan las características actuales, entre otras.

   Por estas razones el sistema permitirá guardar en un archivo todas las características que se encuentren definidas en ese momento (bitácora), y posteriormente en otra sesión o en esa misma cargar ese archivo con lo cual se tendrán nuevamente las características definidas en él.

4) Con la finalidad de evitar errores o conflictos en la ejecución del sistema, en la interfaz se validará toda la información que se especifique, como nombres de archivos, restricciones en los rangos de

valores, etcétera.

5) En cada opción que se presente al usuario se mostrará una explicación (ayuda) con el propósito de evitar confusiones o errores en el usuario, que provoquen pérdida de tiempo al tener que solicitar asesoría, consultar el manual u obtener gráficas con características erroneas.

3.3.3.3 Definición De Datos

Cada grupo de datos a dibujar deberá estar almacenado en una tabla. Podrán existir varios grupos, los cuales pueden ser dibujados en una misma gráfica.

Para almacenar los datos en los arreglos se podrá hacer en cualquiera de las siguientes formas:

1) En forma interactiva

Existirá un pequeño editor que permitirá insertar, borrar, cambiar y listar datos que estén contenidos en la tabla.

2) Insertar datos contenidos en un archivo.

Con esta opción se permitirá que los datos contenidos en un archivo ya sea creado por medio del editor o como resultado de un programa, puedan ser dibujados sin que sea necesario capturar los datos nuevamente al correr el Sistema interactivo.

Sólo será necesario definir el archivo y la posición que tienen los datos en dicho archivo, para que el sistema tome de cada registro del archivo los datos y los coloque en una tabla también previamente definida por el usuario.

3) Como una función matemática.

Es frecuente que los datos que un usuario quiere graficar provengan de una ecuación matemática, razón por la cual se proporcionará una biblioteca de funciones en la cual estaran definidas las funciones más comunmente usadas en aplicaciones matemáticas.

Sólo será necesario especificar la ecuación con los nombres usados en matemáticas y no como se usan en lenguajes de programación, así como el rango de valores o puntos a evaluar. El sistema se encargará de evaluar la ecuación y almacenar los resultados en unas tablas de datos.

Las características de la bibliotaca de funciones se detallará más ampliamente en una sección posterior.

3.3.3.4 Asociación De Datos

Para que se pueda proceder a generar un reporte gráfico, es necesario que se asignen a tal reporte los datos que se tomarán para construirlo. Cada reporte requerirá de un mínimo de grupos de datos. Un grupo de datos puede servir para varias gráficas.

3.3.3.5 Biblioteca De Funciones

Debido a que en muchas aplicaciones es necesario obtener los datos a partir de la evaluación de una ecuación matemática explícita o paramétrica, el sistema contendrá una biblioteca de funciones.

La forma de usarla será muy sencilla, sólamente se tendrán que especificar las ecuaciones de acuerdo a la biblioteca de funciones, el rango de evaluación, el incremento y la tabla en donde se almacenarán los resultados.

Por medio de esta utilería se definirán ecuaciones lo más parecido posible al lenguaje matemático, con la finalidad de que el usuario necesite los mínimos conocimientos de cómputo. Se proporcionará una biblioteca de funciones con los nombres que se utilizan en matemáticas y no como se nombran en algún lenguaje de programación, la cantidad de parámetros variará de acuerdo a la función y se utilizarán los operadores matemáticos suma (+), resta (-), división (/), multiplicación (*) y potenciación (**).

Dicha biblioteca estará formada por las funciones ya implantadas en FORTRAN 77 y por otras de uso común en ingeniería que no proporciona dicho lenguaje.

Los nemónicos de las funciones a desarrollar para la biblioteca son:

1) DNormal
2) DSierra
3) DSRect
4) Fac
5) OCuad
6) Senc
7) SRect
8) Trian
9) TRect

Para evitar errores de ejecución se hará un análisis sintáctico-semántico a las ecuaciones que proporcione el usuario. Si existe algún error el sistema tratará de mandar un mensaje para orientar al usuario en la corrección del problema.

Para un mejor mantenimiento, se contará con un archivo de funciones válidas y en el momento que exista algún cambio en las funciones del compilador sólo se cambiarán los nombres en el archivo.

En la evaluación de funciones paramétricas se podrá obtener un grupo de datos a partir de la evaluación de dos ecuaciones matemáticas que cumplan con las características de la biblioteca de funciones y que tienen un parámetro en

común. Esta característica es muy usual para la obtención de círculos, elipses, etc.

3.3.3.6 Despliegue De Información

Para la ejecución en forma interactiva del sistema, se utilizarán terminales VT100 o compatibles. Por esta razón contamos con 24 renglones de 80 columnas para despliegue y captura de información. Para aprovechar lo mejor posible la pantalla se dividirá como se muestra en la figura 3-2.

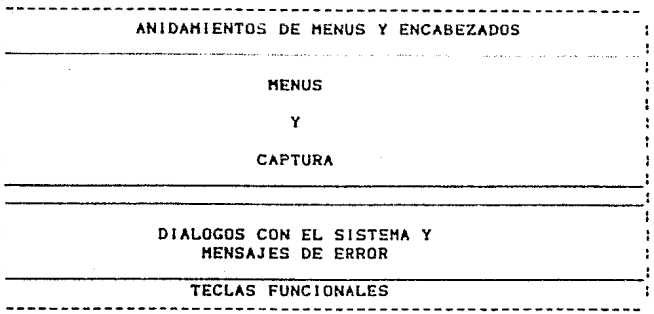


Figura 3-2

3.3.3.6.1 Descripción De Las Zonas

La zona de anidamientos y encabezados desplegará el nivel de menús en que se encuentra el usuario, es decir, las opciones que ha seleccionado para llegar al menú en que se encuentra. También mostrará un pequeño encabezado del sistema.

La zona de menús y captura estará ocupada por el catálogo de opciones que puede utilizar el usuario. Si en alguna opción es necesario capturar mucha información, por ejemplo, captura de tablas y características particulares de una gráfica, la zona se ocupará por una forma de captura. Una vez que se termine la captura, esta zona mostrará el catálogo de opciones que corresponden al nivel en donde se encuentre el usuario.

Si el usuario requiere de ayudas o despliegue de información en la región de menús y captura se abrirá una ventana en donde se mostrará la información solicitada por el usuario.

La zona de diálogo y errores es la zona de trabajo del usuario con excepción del caso de captura masiva. En esta zona el usuario dará las respuestas a las preguntas que le haga el sistema, características de gráficas, etcétera. En esta zona también se mostrarán mensajes del sistema cuando el usuario incurra en algún error en la operación del sistema o en los datos introducidos.

Dependiendo del nivel en donde se encuentre el usuario, la zona de teclas funcionales mostrará qué teclas realizan una función específica. Tiene como objetivo que el usuario no tenga que memorizar la función de dichas teclas y los niveles donde se pueden usar. Algunos ejemplos de teclas funcionales son: ? para ayudas, <ESC> para terminar una captura, <CTRL/T> para desplegar información, entre otras.

Los tamaños de las zonas de menús y captura será de 19 renglones.

Las zonas de antecedentes y teclas funcionales ocuparán un renglón cada una. La zona de diálogos ocupará tres renglones.

Además se utilizará un renglón de separación entre menús y diálogos con el sistema.

3.3.3.6.2 Ejemplos

Con la finalidad de ejemplificar las características mencionadas, se muestra una parte de una posible secuencia de trabajo.

La pantalla que aparecerá inicialmente se muestra en la figura 3-3:

```
PRINCIPAL                                          SER 1.0 CECAFI

                HENU _ PRINCIPAL
                     --------
                     --------
                     --------
                     --------

>>>
>>>
>>>

  <ESC> TERMINAR <?> AYUDA
```

Figura 3-3

Si se solicitara ayuda tecleando ?, la pantalla obtenida se muestra en la figura 3-4.

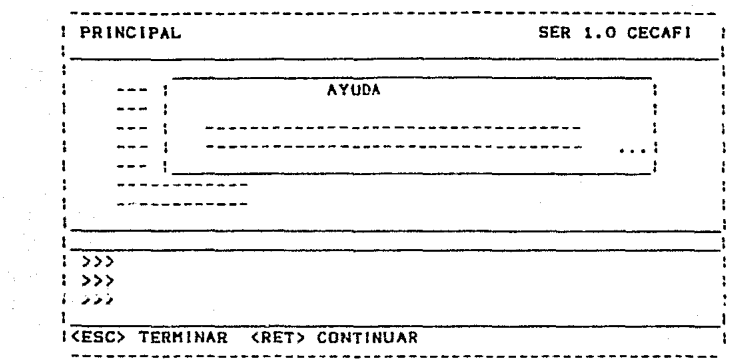


Figura 3-4

Si se teclea la opción de tablas y dentro de ella la opción de captura en forma interactiva, la pantalla mostrada se observa en la figura 3-5.

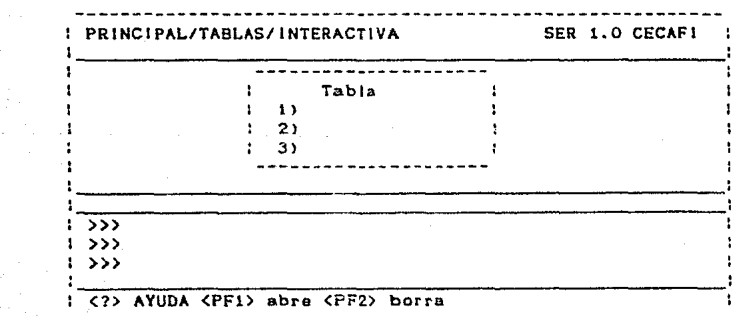


Figura 3-5

3.3.3.7 RUTINAS GRAFICAS

El usuario tendrá la posibilidad de obtener las mismas salidas también por medio de rutinas como a través del sistema interactivo, para lo cual sólo será necesario que desde un programa en FORTRAN las llame de acuerdo a la siguiente secuencia:

1) DEFINICION DEL REPORTE

   Se define las dimensiones de la hoja, desplazamientos de la hoja, número de reportes que tendrá la hoja y nombre del archivo en donde se almacenará.

2) MEDIO AMBIENTE (OPCIONAL)

   Sólo si se desea, se pueden modificar las características del medio ambiente que se tienen por "DEFAULT".

3) TIPOS DE GRAFICAS

   Se indica el tipo de las gráficas que se desea. El número de llamadas de este tipo estará de acuerdo al número de gráficas definidas.

4) TERMINACION DE LA HOJA

CAPITULO 4

DISEÑO DEL SISTEMA

La finalidad de este capítulo es presentar en forma general los criterios de diseño de las dos partes que integran el sistema SER: interfaz interactiva y rutinas gráficas. Los capítulos seis y siete se dedican en exclusiva a los detalles de cada una de estas dos partes.

4.1 MEDIO AMBIENTE

A pesar de que en el capítulo uno ya se habló un poco acerca del medio ambiente en el cual se desarrolló este trabajo, en este apartado se hablará más a fondo del hardware y software empleados para la elaboración del sistema.

4.1.1 Equipo De Cómputo

Teniendo en cuenta que el sistema SER fué un proyecto hecho en y para el CECAFI, los recursos de cómputo empleados fueron exclusivamente aquellos de los que se dispone permanentemente en el Centro.

Los servicios de cómputo que ofrece el CECAFI se basan principalmente en su equipo mayor (VAX-11/780). Esta fué una de las razones principales, además del gran poderío de software del equipo VAX, por las que se seleccionó tal equipo como computadora de desarrollo.

A grandes rasgos, la configuración que presentaba el equipo VAX al día de la elaboración de este escrito era:

1) Un procesador 780.

2) Dos controladores de memoria (cada uno puede manejar hasta 4 MB).

3) 5.5 MB de memoria principal.

4) Cinco unidades de disco RP06 con capacidad de almacenamiento de 176 MB cada una.

5) Una unidad de cinta magnética TU77 con capacidad de grabación de 800 bpi y 1600 bpi.

6) Una impresora de linea LP11 con una velocidad de 600 lpm (impresora de caracteres formados).

7) Una impresora con capacidades gráficas PRINTRONIX 600/LXY02 con una velocidad de impresión de 600 lpm (impresora de matríz de puntos).

Las características más importantes son:

a) Velocidad: 600 lpm. 320 lpm para caracteres de doble altura, 465 lpm cuando se subraya o se imprimen caracteres en minúscula con "descenders".
b) Matríz: El tamaño de la matríz es de 7 x 9 puntos (mayúsculas). En el plano horizontal existen cinco puntos de traslape. En el plano vertical existen siete puntos de traslapo. El espaciamiento horizontal entre puntos es de 0.0167 in. y 0.01389 en espaciamiento vertical.
c) Conjunto de carácteres: Se tiene un conjunto de 96 caracteres ASCII. Se tiene la opcion de poder aumentar el conjunto 64 carácteres más
d) Características especiales: Caracteres de doble altura, modo gráfico, caracteres subrayados, 6 u 8 LPI, Electronic VFU.
e) Modo gráfico: Densidad de 60 puntos por pulgada horizontal, 72 puntos por pulgada vertical. Razón de 33.3 pulgadas por minuto. Bajo supervisión de un programa , se puenden imprimir puntos medios en cualquier columna, con la finalidad de tener una imagen más densa.

8) Una unidad dual de floppy RX02.

9) 48 puertos asíncronos RS232.

Cabe mencionar que previendo las fallas que puedan dejar fuera de servicio al equipo gráfico PRINTRONIX, se desarrollaron dos traductores de archivos formato PRINTRONIX a formato EPSON y DECWRITER IV. De esta manera se pudo haber pensado en otras impresoras, sin embargo, el CECAFI no cuenta con otros modelos de impresoras. Para conectar una impresora al equipo VAX se requiere únicamente que la primera posea una interfaz serie RS232. Si este no es el caso, la impresión de gráficas se puede llevar a cabo a través de una microcomputadora PC. Se traduce el archivo formato PRINTRONIX al formato requerido, se transfiere el nuevo archivo de VAX a PC y se imprime en la impresora conectada a la PC.

4.1.2 Software Empleado

En lo que respecta al software se puede apreciar lo siguiente: El sistema operativo bajo el cual se desarrolló el sistema SER es el VMS V4.2. Dicho sistema operativo proporciona un ambiente multiusuario, opera procesos ON-LINE y BATCH y maneja un esquema de memoria virtual.

4.1.2.1 Lenguaje

De entre los compiladores disponibles (PASCAL, COBOL, BASIC y FORTRAN-77) se seleccionó el de FORTRAN-77 por lo siguiente:

1) El manejo de funciones y operadores matemáticos es más poderoso y sencillo. El tipo de operaciones que involucran las rutinas gráficas son netamente de cálculos matemáticos.

2) FORTRAN es el lenguaje de programación por excelencia en el Centro de Cálculo y por lo tanto el que mejor se conoce.

3) El uso de bloques de datos comunes a varias rutinas es un requisito indispensable en la programación de las rutinas gráficas. Con esto se evita que el usuario tenga que pasar numerosos parámetros ajenos a él.

4) La razón más poderosa es que el paquete de primitivas gráficas utilizado (PGPLOT) está escrito en FORTRAN. El ligado de rutinas FORTRAN con otros lenguajes podría haber causado problemas en el pase de parámetros.

La única desventaja que presenta FORTRAN con respecto a PASCAL, es la ausencia de estructuras dinámicas (apuntadores). Sin embargo estas estructuras se implementaron con las herramientas propias del lenguaje. Es importante mencionar que para homogeneizar la codificación de los programas que integran el sistema se desarrollaron unos estándares de programación FORTRAN. Ver apéndice A.

4.1.2.2 Paquete Gráfico

Además del sistema operativo y el compilador se utilizó un paquete de primitivas gráficas (PGPLOT). Este paquete fué donado a la U.N.A.M. por el Instituto de Astronomía de la Universidad Norteamericana de Caltech. El CECAFI recibió este donativo a través de la Dirección General de Servicios de Cómputo Académico (DGSCA).

PGPLOT puede generar dibujos en las impresoras graficadoras Printronix y Versatec, en el despliegue de video Grinnell, en terminales Tektronics, DEC VT125 y terminales VT100 modificadas. Las rutinas están escritas en VAX-11 FORTRAN y hacen uso del paquete gráfico GRPCKG. En el paquete GRPCKG se encuentran las rutinas gráficas básicas utilizadas por PGPLOT.

De esta serie de rutinas únicamente se utilizaron las primitivas: trazo de puntos, trazo de texto, trazo de líneas, manejo de ventanas, manejo de viewport y trazo de ejes. A partir de estas primitivas se elaboraron rutinas cada vez más complejas hasta llegar a módulos del tipo dibuja diagrama de barras, dibuja diagrama circular, dibuja histograma, etcétera.

Para mayor referencia respecto al tipo de rutinas gráficas que proporciona el paquete de primitivas consúltese el manual de PGPLOT [15].

4.2 VISION GLOBAL

Como ya se ha mencionado en algunos párrafos de este escrito el sistema "SER" se puede dividir en dos grandes partes: Sistema Interactivo y paquete de rutinas gráficas. En ambos casos la filosofía de diseño fué la misma: **DISEÑO POR CAPAS.**

Esta filosofía de diseño se encuentra muy bien ejemplificada en los sistemas operativos. En los sistemas operativos las capas más internas las forman las rutinas que manejan el hardware. Las capas superiores hacen uso de las capas inferiores y van desligando al usuario de tareas primitivas. Un ejemplo de lo antes dicho aclarará la situación. Cuando un usuario que se encuentra sentado frente a una terminal de una computadora teclea un comando, p.e. DIRECTORY, para ver los archivos que residen en su cuenta, se genera el siguiente proceso: el controlador de procesos (JOB CONTROLLER) recibe la petición, la petición se envía al administrador de la información y éste haciendo uso del driver del dispositivo de almacenamiento extrae la información.

De una manera semejante, el sistema "SER" esta sustentado en rutinas que forman familias. Cada familia de rutinas ejecuta funciones muy específicas. Las familias de rutinas de niveles superiores únicamente hacen uso de rutinas de familias inferiores o de su mismo nivel.

Las ventajas de este tipo de diseño se palpan a la vista:

1) **Se evita la repetición de código.** En el caso que un proyecto de software esté siendo desarrollado por varias personas, si no se tiene una comunicación muy estrecha entre los programadores, es muy sencillo que se duplique el código por no haber identificado módulos de utilidad común. Con el uso del diseño por capas, antes de comenzar la programación se identifican las necesidades comunes y se crean familias de rutinas con funciones semejantes.

2) **Facilidad para programar rutinas complejas.** Dado que las funciones de mayor complejidad estan sustentadas en rutinas de menor complejidad, la programación de las primeras resulta muy sencillo. Las rutinas que se encuentran en las capas más externas basan sus algoritmos en el uso de rutinas de niveles internos, de esta manera no se adentran en los detalles y resulta mucho más fácil la programación.

3) **Se facilita el mantenimiento.** En caso de requerir hacer cambios a los algoritmos, ya sea por mejoras o por modificaciones, si estos cambios afectan a muchas rutinas, posiblemente con la modificación de un solo grupo de rutinas se logre el cambio.

4) **Mejora el trabajo en grupo.** Si se está trabajando con un grupo de programadores, es muy recomendable dividir el trabajo de programación por familias de rutinas. De esta manera los programadores tendrán que resolver solo un tipo semejante de problemas (el que corresponde a la familia de rutinas) y no diversificar demasiado el tipo de algoritmos al que se enfrenten. Al conocer mejor los problemas, se obtendrán mejores y más rápidos resultados.

4.2.1 Diseño Del Sistema Interactivo (SI)

El capítulo siete esta dedicado íntegramente al Sistema Interactivo. Sin embargo, para que quede ejemplificada la filosofía del diseño por capas, se presentará una breve explicación.

Debido a que la terminal de video es el instrumento de comunicación con el usuario. Se decidió dividir la pantalla en zonas funcionales. Zona de mensajes y lectura de información, zona de menús y captura de formas, zona de despliegue de teclas funcionales y zona de anidamientos. Por otra parte toda la SI funciona bajo un ambiente de ventanas.

Con esta idea en mente se definieron las características y funcionamiento de cada una de las zonas. Una vez hechas estas definiciones se pudieron distinguir una serie de familias de rutinas.

El nivel más interno lo constituye las familias de rutinas de detección de teclado (KB) y de manejo de video (RTL proporcionadas por el sistema operativo VMS de VAX). Por medio de ellas se puede detectar la presión de alguna tecla o la presión de una secuencia de teclas.

Por otra parte, todo el funcionamiento de las zonas de la pantalla está basado en rutinas de manejo de ventanas y zonas (MVT). El manejador de ventanas es la esencia de toda la SI. Las funciones que proporciona esta familia de rutinas fueron utilizadas para crear varias familias de rutinas de propósito específico (manejadores). Todas las tareas de la SI son desarrolladas por grupos de rutinas de propósito específico llamados manejadores. Así pués, se cuenta con los siguientes manejadores:

1) Ventanas y zonas (MVT, apertura, cierre de ventanas y lectura y escritura sobre una ventana)

2) Menús (MMU, despliegue de opciones y control y validación de la lectura de las opciones)

3) Diálogos (MDI, despligue de mensajes y de captura de datos)

4) Captura (MCA, despligue de formas de captura y lectura y validación de las mismas)

5) Anidamientos (MAN, despliega el nivel en el que se encuentra el usuario)

6) Ayuda (MAY, proporciona documentación en linea del sistema)

7) Estructuras (MES, administra el almacenamiento de la información capturada, proporciona un editor para captura de datos)

8) Teclas Funcionales (MTF, despliega en la zona que se le haya asignado, las teclas válidas para el ámbito de operación)

Un diagrama jerárquico de los módulos de la SI se muestra en la figura 4-1.

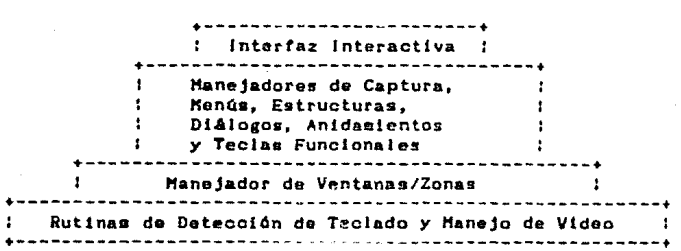


Figura 4-1

Una de las grandes ventajas de los manejadores interactivos es la posible aplicación de éstos en otro tipo de sistemas interactivos. De hecho, la SI del sistema "SER" es un caso práctico del uso de los manejadores. Salvo el manejador de estructuras, ninguno de los otros manejadores esta ligado con el sistema "SER", todos son de propósito general y se les puede dar gran uso como herramientas para el desarrrollo de aplicaciones interactivas.

4.2.2 Diseño De Las Rutinas Gráficas (RG)

El capítulo siete de esta tesis se dedica a la explicación del módulo de rutinas gráficas. A continuación se mostrará un panorama general de este módulo.

En las RG se aprecia mucho mejor la filosofía de diseño por capas. El módulo más elemental de rutinas gráficas (primitivas gráficas) lo constituye el paquete gráfico PGPLOT, del cual ya se habló anteriormente.

En la figura 4-2 se presenta un diagrama que muestra las capas que constituyen las familias de rutinas gráficas.

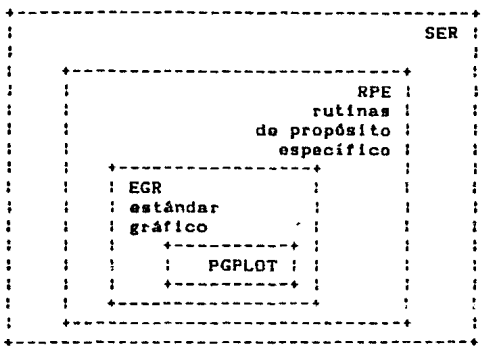


Figura 4-2

Las familias de rutinas que integran este módulo son:

1) **PGPLOT:** Con respecto a este paquete ya se habló un poco en párrafos anteriores.

2) **EGR:** Hace uso de las rutinas de PGPLOT. En este grupo se pueden encontrar rutinas como: trazo de texto, trazo de ejes cartesianos, definición de una ventana, definición de un viewport, etcétera. Es importante el conocimiento de las rutinas de esta familia debido a que si se llegara a trasladar el sistema "SER" a otro ambiente donde no existiera PGPLOT, lo único que habría que reemplazar serían las rutinas EGR por aquellas equivalentes en el sistema huesped.

3) **RPE:** Haciendo uso de las rutinas EGR, las rutinas RPE implementan funciones de utilidad a cada tipo de gráfica que genera el sistema: trazo de un rectángulo, relleno de un rectángulo, trazo de un círculo, unión de una serie de puntos, relleno de un fragmento de círculo, etcétera.

4) **SER:** Las rutinas "SER" son el producto final del sistema. Estas rutinas se ponen a disposición del usuario que desea utilizar el sistema a través de un programa escrito en FORTRAN. Además de las catorce rutinas para cada tipo de gráfica, se proporcionan rutinas de inicio de la gráfica, trazo de títulos, definición del tipo de escalamiento, definición de la hoja de dibujo, terminación de la gráfica, etcétera.

   A diferencia de la generación de gráficas por medio de la interfaz interactiva, el uso de las rutinas gráficas proporciona mayor flexibilidad al usuario. El usuario define las partes del reporte gráfico que deben aparecer y elimina las que crea conveniente.

CAPITULO 5

DISEÑO DE LA INTERFAZ INTERACTIVA

En este capítulo, dedicado al diseño de la interfaz interactiva, se presentan detalladamente los diversos manejadores que la integran.

Para cada manejador se definen sus funciones, las estructuras de datos empleadas y se enlistan las rutinas que lo integran.

5.1 MANEJADOR DE VENTANAS (MVT)

A lo largo de este punto se tratarán las características del Manejador de Ventanas. Para este efecto, el texto se fragmentó en dos grandes grupos: el Manejador de Ventanas y el Manejador de Zonas. El Manejador de Ventanas se describe primero y para ambos manejadores, se detallan las características generales, el algoritmo, la implementación en VAX-11/780, así como una descripción de las rutinas que integran a los manejadores y unos ejemplos. Es importante mencionar que el Manejador de Zonas forma parte del Manejador de Ventanas, sólo que por su extensión se trata en forma especial.

5.1.1 Generalidades

El uso de ventanas se ha extendido ampliamente con el advenimiento de microcomputadoras, cuyos sistemas operativos tienen incorporados un ambiente de ventaneo. Tal es el caso de Macintosh o las microcomputadoras AMIGA y ATARI ST. Sin embargo, las ventanas no son un producto predestinado a las microcomputadoras, por el contrario, durante muchos años el centro de investigación de XEROX en Palo Alto California ha trabajado en el desarrollo de sistemas basados en un ambiente de ventanas, dando por resultado lo que en el medio de minicomputadoras se conocen como estaciones de trabajo (workstations).

Una ventana es una porción de la pantalla que aparece dentro de un vídeo (el monitor de la computadora). Su objetivo es permitir realizar una acción dentro de ella y luego desaparecer en el momento en que no se necesita más. Cuando se abre, o crea, una ventana oculta todo lo que se encuentra bajo ella, incluyendo secciones de otras ventanas. Generalmente se dibuja un borde alrededor de la ventana para delimitarla visualmente de todo lo demás que hay en la pantalla, y también, algunas veces se rotula con un título en la parte superior. Una vez abierta, en la ventana se puede realizar cualquier función que normalsente opera en todo el vídeo, como escribir textos, dibujar figuras, etc. Sin embargo,

cualquier intento de dibujar o escribir fuera de los límites de la ventana se ignora. Esto normalmente se realiza a través de una rutina denominada Frontera (viewport).

Se puede escribir en una ventana siempre y cuando esté activa. Si dos ventanas se traslapan, aquella que se encuentra encima de la otra se dice que está activa. Si las ventanas no se traslapan, solamente una puede estar activa en un momento dado. Luego entonces, sintetizando: la ventana activa siempre será la más recientemente abierta. En la figura 5-1 se muestra un ejemplo de una serie de ventanas abiertas.

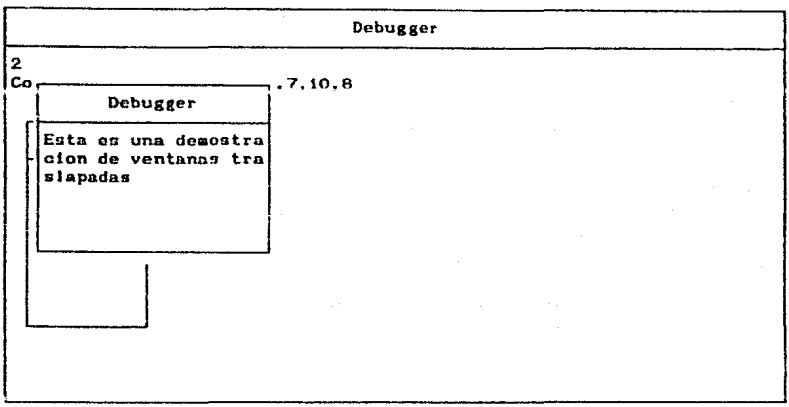


Figura 5-1

Una vez que la tarea de la ventana ha concluido, hay que cerrarla y restaurar lo que había debajo de ella. Para esto, al abrir una ventana, el sistema deberá almacenar la información de la pantalla que va a ocultar la ventana, y así cuando se cierra, pueda restaurar lo que había debajo de ella. El criterio que se usa para indicar que una ventana debe cerrarse es, la ventana más recientemente abierta es la primera en cerrarse.

Para implementar un sistema de ventaneo se requiere de un conjunto muy simple de rutinas. La primera es una rutina para definir una Frontera (viewport), para restringir la escritura o lectura a la ventana activa; una rutina para borrar la porción de la pantalla donde la ventana aparecerá y una rutina de posicionamiento de cursor para el desplegado y la lectura de texto.

5.1.1.1 Abrir Una Ventana

El primer paso para abrir una ventana es especificar su posición, por ejemplo, la esquina superior izquierda y las dimensiones de la ventana (alto y ancho). Además de la ventana, se dibuja un borde y un título, el cual lo debe indicar el usuario. Un requisito indispesable es que el manejo del video de la computadora sea del tipo "Mapeado en Memoria". Esto significa que algún área de RAM de la computadora contiene una imagen exacta de la información que hay en el video. De tal forma que cualquier texto escrito a la memoria aparece en la pantalla, y cualquier información extraida de la memoria debe obligadamente estar en el video. En los sistemas de "Mapeado en Memoria", generalmente cada byte representa un caracter en la pantalla; el valor del byte es el equivalente ASCII del caracter en el video.

Ya que muchos sistemas consideran la esquina superior izquierda como origen de su video (0,0), se tomará en forma convencional que los valores de X crecen hacia la izquierda de la pantalla, y los valores de Y, hacia abajo. Por consiguiente, cuando se especifique la esquina superior izquierda de la ventana, para conservar compatibilidad, se indicarán las coordenadas (X,Y) de este punto, siendo X la columna y Y el renglón de inicio. Para las dimensiones de la ventana, se deben proporcionar al ancho y el alto, el primero en columnas y el segundo en renglones. De este modo, la posición de la esquina inferior derecha de la ventana (X2,Y2), estará dada por las siguientes ecuaciones: X2 = (X + ancho - 1) y Y2 = (Y + Alto - 1). Vale la pena hacer notar que si la ventana incluye un borde y un texto, afectará sus dimensiones reales; por lo que el cálculo de (X2, Y2) debe reflejar este cambio. Por último, también se requiere un "Buffer", es decir un área en memoria, para almacenar la información de la pantalla que se ocultará con la ventana.

Existen tres posibles condiciones de error que pueden ocurrir al abrir una ventana:

1) Las coordenadas (X,Y) se encuentran, al menos una, fuera de los límites; es decir, tienen un valor menor a cero, X excede al borde derecho o Y rebasa el límite inferior de la ventana.

2) El ancho o la altura son demasiado grandes, es decir, X2 o Y2 caen fuera de los límites de la pantalla.

3) El "Buffer" no es suficientemente grande para contener la información que cubrirá la ventana.

Para manejar todas estas posibilidades, la rutina Abre Ventana regresa un código de error informando la razón por la cual la ventana no se pudo abrir.

Uno de los principales puntos es decidir, cuál debe ser el tamaño del "Buffer". La solución más simple es asignarle un espacio suficiente para contener a la ventana más grande; esto es, las dimensiones del video; lo cual implica que se podrá abrir una ventana del tamaño de toda la pantalla. Ante esto surge un problema cuando el tamaño del "Buffer" no es lo suficientemente grande para contener todas las ventanas abiertas. La alternativa que se emplea para solucionar esto, es escribir el contenido del "Buffer" a disco y así, poder utilizarlo nuevamente para almacenar la ventana que se va a abrir. Es decir,

cada vez que se va a abrir una ventana debe escribirse el contenido del "Buffer" en la posición en disco que le corresponde (vease la figura 5-2)

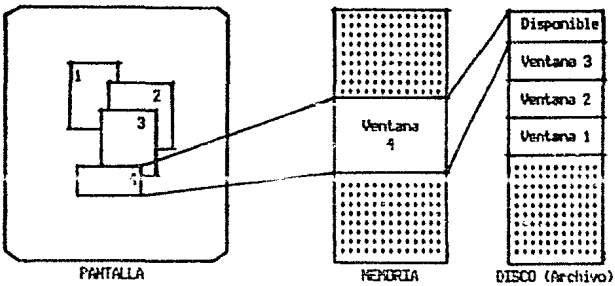


Figura 5-2

Para dibujar los bordes se pueden emplear caracteres gráficos especiales. Sin embargo, es importante considerar el espacio que consume este borde y el título de la ventana, cuando se especifican sus dimensiones. Con respecto a este punto, es posible tomar varios caminos; el primero es considerar que el borde se encuentra dentro de la ventana y la otra alternativa, fuera de ella. Para el manejador en VAX, se decidió considerar que el borde cae fuera de la ventana; de tal manera que las dimensiones de ella especifican el ancho y largo del espacio útil.

Por último, la rutina de la Frontera debe colocar los límites de la ventana de tal forma que ella quede dentro del borde.

5.1.1.2 Cerrar Una Ventana

Para cerrar una ventana primero hay que recuperar del "Buffer" la localización (x, y, alto, ancho) de la ventana más reciente, sus dimensiones y la información de la pantalla que hay que restaurar; es decir, la información que fué cubierta cuando la ventana, que ahora se va a cerrar, se abrió. Es conveniente recordar que la ventana más recientemente abierta es la que se va a cerrar. Por lo tanto si se transfiere la información que está en el "Buffer" hacia la pantalla, la última ventana habrá desaparecido; esto es, la operación inversa a copiar la información de la pantalla al "Buffer". Si ya no hay más ventanas abiertas, simplemente define que la Frontera tenga las dimensiones de la pantalla completa. Si hay más ventanas abiertas, la ventana más reciente debería localizar su información, en el "Buffer"; esta información se encuentra en disco y por lo tanto, debe enviarse de ahí al "Buffer". Por último hay que extraer del "Buffer" las coordenadas de la esquina superior izquierda (X,Y) y

las dimensiones de la ventana, ya que ésta se acaba apenas de recuperar; y son estos valores, los que se emplean para definir a la Frontera.

Para concretar más los conceptos de abrir y cerrar una ventana, a continuación se exponen los pseudocódigos de ambas rutinas.

5.1.2 Pseudocódigo

En este punto se presentan los pseudocódigos de las rutinas Abre Ventana y Cierra Ventana. Es importante hacer notar que la implementación que se realizó en el sistema VAX difiere un poco de estos pseudocodigos, ya que el propósito de éstos es facilitar la comprensión del algoritmo. Las diferencias se explican más adelante en el inciso de implementación en el sistema VAX-11/780.

5.1.2.1 Pseudocódigo De Abre Ventana

El listado 5-1 muestra el pseudocódigo de la rutina Abre Ventana. Existen tres variables globales que se deben compartir con Cierra Ventana y otras rutinas internas: BufEmpleado, Buffer y TamanoBuffer. Buffer es la estructura de datos que almacena la información de la pantalla que queda cubierta por la ventana; TamanoBuffer es el espacio máximo disponible de Buffer para almacenar información y BufEmpleado es una bandera que indica si el Buffer tiene información almacenada o no; debe hacerse igual a FALSO al inicio del programa principal.

De igual forma, existen cuatro constantes globales: Xmin, Ymin, Xmax, Ymax. Xmin tiene la menor coordenada en X permitida, es decir, el valor de X para el borde izquierdo de la pantalla. Su contraparte Xmax tiene el valor máximo permitido para X, o en otras palabras, el valor de X para el borde derecho de la pantalla. En forma análoga, Ymin tiene el menor valor permitido para Y; esto es, el valor de Y para el borde superior de la pantalla. Por último, Ymax tiene el valor máximo permitido para Y, o el valor de Y para el borde inferior de la pantalla.

También existen procedimientos o rutinas que llama Abre Ventana. Estos son AlmacenaBuffer, SCREENADDRRESS, SCREEN, Frontera, BorraPantalla, SETCURSOR y DibujaCaja. AlmacenaBuffer guarda el contenido del Buffer en disco. SCREENADDRRES es una función de la computadora que regresa la dirección en memoria de dónde se encuentra la posición (X,Y) de la pantalla. SCREEN es una función, que dada la dirección en memoria, regresa el equivalente ASCII de cada byte de la pantalla y viceversa. Frontera define los límites de la ventana. SETCURSOR coloca el cursor en una posición de la pantalla. Finalmente, BorraPantalla borra una porción de la pantalla.

En el listado 5-1 se pueden apreciar los parámetros que se pasan a Abre Ventana: las coordenadas de la columna superior izquierda y el ancho y largo de la ventana. Esta rutina regresa un código de error indicando si la ventana no pudo abrirse por alguna razón.

El primer paso en Abre Ventana es inicializar algunas variables, después detectar si existe alguna condición de error. Se verifican cuatro posibles casos de error: la ventana es demasiado grande para el Buffer; las coordenadas

de la esquina superior izquierda están fuera de la pantalla; la ventana es demasiado ancha o demasiado alta para caber en la pantalla; no hay espacio en disco para almacenar el Buffer.

El siguiente paso es guardar en el Buffer la sección de la pantalla que la ventana va a cubrir. Sin embargo, antes debe guardarse en el Buffer la localización de esa porción de la pantalla, es decir, las coordenadas de la esquina superior izquierda y el ancho y el largo del área guardada. En el listado 5-1 se toma la convención que Buffer (i;j) son todas las localidades de Buffer, desde i hasta j. Una vez hecho esto, por medio de un ciclo iterativo se calcula la dirección de inicio de cada renglón del área de la ventana y copia la información de esa región de la pantalla al "Buffer".

Después de haber guardado la porción de la pantalla en el "Buffer", ahora se dibuja la ventana. Para esto, primero se dibuja la caja y finalmente se ajustan las dimensiones de la Frontera para que esté exactamente dentro del borde de la caja. Es importante hacer notar que para el pseudocódigo mostrado en el listado 5-1, el borde de la caja se dibuja dentro de la ventana; esto se hace fundamentalmente para facilitar la compresión del código; sin embargo, en la versión de VAX se empleó otra modalidad, la cual se trataŕa más adelante. A continuación se detalla el pseudocódigo de CierraVentana.

```
AbreVentana (X1, Y1, Ancho, Alto, Error)

! Inicializa variables
Error = 'No hay error'
Tamano = 4 * Ancho * Alto
X2 = X1 + Ancho - 1
Y2 = Y1 + Alto  - 1

! Deteccion de errores
IF Tamano > TamanoBuffer THEN
    Error = 'Demasiado grande'
    EXIT AbreVentana
NOELSE
ENDIF

IF (X1 < XMin  OR  X1 > XMax)  OR  (Y1 < YMin  OR  Y1 > YMax) THEN
    Error = 'Coordenadas Mal'
    EXIT AbreVentana
NOELSE
ENDIF

IF BufUsado THEN
    AlmacenaBuffer (Error) !Guarda el Buffer en disco
    IF Error <> 'No error' THEN
        EXIT AbreVentana
    NOELSE
    ENDIF
NOELSE
ENDIF

! Copia al buffer la informacion que cae debajo de la ventana
```

```
Buffer (1) = X1
Buffer (2) = Y1
Buffer (3) = Ancho
Buffer (4) = Alto

BuffInicio = 5
DO Renglon = Y1 TO Y2 STEP 1
    Direccion = SCREENADDRESS (X1, renglon)
    Buffer (BuffInicio: BuffInicio + Ancho - 1) =
        SCREEN (Direccion : Direccion + Ancho - 1)
    BuffInicio = BuffInicio + Ancho
ENDDO

BufUsado = Verdadero

! Dibuja la ventana en la pantalla y define la frontera
Frontera (X1, Y1, X2, Y2)
BorraPantalla
DibujaCaja (X1, Y1, X2, Y2)
Frontera (X1+1, Y1+3, X2-1, Y2-1)
SETCURSOR (0,0)

END
```

Listado 5-1

5.1.2.2 Pseudocódigo De Cierra Ventana.

Cierra Ventana, como se mencionó en Abre Ventana, comparte las variables globales, las constantes globales y algunos procedimientos como SCREENADDRESS, SCREEN y Frontera. Lo primero que hace el código mostrado en el listado 5-1 es, asegurarse que exista una ventana que cerrar; esto es, que el Buffer contenga información. Después, recupera del Buffer, las coordenadas de la esquina superior izquierda y las dimensiones de la ventana. Una vez hecho esto, ya se pueden copiar los datos del Buffer hacia el monitor, lo cual borra la ventana y restablece lo que había debajo.

Cierra Ventana debe ahora verificar si existe otra ventana abierta. Esto lo hace llamando a BuscaBuffer, la cual copia los datos, almacenados en disco, de la ventana más recientemente abierta. Si existe una ventana abierta, BuscaBuffer, asigna el valor de Verdadero, en caso contrario, lo hace igual a FALSO. CierraVentana verifica el valor de BufUsado y si es verdadero, extrae de Buffer la información requerida para establecer la Frontera, de acuerdo a la ventana que no esté activa y es la más recientemente abierta (sin contar a la que se va a cerrar). Si el valor de BufUsado es falso, la Frontera se establece para que cubra a toda la pantalla. Así entonces, la ventana se ha cerrado.

En el siguiente punto se cubrirán los detalles de la implementación específica del sistema de ventanas en la VAX-11/780.

```
CierraVentana (Error)

! Verifica que existe ventana para cerrarla
Error = 'No Error'
IF BufUsado THEN
    Error = 'No hay ventana que cerrar'
    EXIT Cierraventana
NOELSE
ENDIF

! Extrae X1, Y1 Ancho y Alto
X1 = Buffer (1)
X2 = Buffer (2)
Ancho = Buffer (3)
Alto  = Buffer (4)

! Copia el contenido de Buffer de regreso a la pantalla
BufferInicio = 5
Y2 = Y1 + Alto - 1
DO Renglon = Y1 TO Y2 STEP 1
    Direccion = SCREENADDRESS (X1, renglon)
    SCREEN (Direccion : Direccion + Ancho - 1) =
        Buffer (BuffInicio : BuffInicio + Ancho - 1)
    BuffInico = BuffInicio + Ancho
ENDDO

! Establece las caracteristicas de la ventana que queda,
!si existe alguna.

BuscaBuffer ()

IF BufUsado THEN
    X1 = Buffer (1)
    Y1 = Buffer (2)
    Ancho = Buffer (3)
    Alto  = Buffer (4)
    X2 = X1 + Ancho - 1
    Y2 = Y1 + Alto - 1
    Frontera (X1+1, Y1+1, X2-1, Y2-1)
ELSE
    Frontera (Xmin, Ymin, Xmax, Ymax)
END
```

Listado 5-11

5.1.3 Implementacion En El Sistema VAX-11/780

En esta sección se detallan los aspectos más importantes de la implementación del Manejador de Ventanas en el sistema VAX-11/780, incluyendo las modificaciones esenciales a los algoritmos Abre Ventana y Cierra Ventana. La implementación del Manejador de Ventanas se complicó, esencialmente, por la falta de los elementos básicos: una rutina de Frontera, un esquema de pantalla "Mapeada en Memoria" y rutinas de impresión y lectura que respetarán a la Frontera. A continuación se describen estos aspectos.

Las terminales VT100 y el sistema operativo VMS no contemplan tener en el video un esquema "Mapeado en Memoria" a disposición del usuario; esto es, que no es posible conocer qué información existe en una posición del video en particular. Para resolver este dilema se ideó substituir el esquema de una pantalla mapeada en memoria interna, por una mapeada externamente. Por lo tanto, el propio Manejador de Ventanas, lleva un mapa de la pantalla en memoria, por medio de una matríz de 24 renglones por 80 columnas, que es una copia fiel en RAM, de la información de la pantalla del monitor. De tal suerte, que cada renglón-columna de la matríz, se "mapea" a una posición en la pantalla.

De igual forma, los usuarios de las terminales VT100, conocen perfectamente que si bién estas terminales permiten colocar caracteres gráficos y un atributo específico a cada coordenada (renglón-columna) de la pantalla; no proporcionan un medio para conocer qué atributo tiene un caracter en una posición de la pantalla, o si ese caracter se encuentra en modo gráfico. Para este efecto se ocurrió llevar una matríz alterna a la del mapa de memoria, que almacena para cada posición de la pantalla, qué atributo tiene. De esta forma la matríz resultante también es de 24 renglones por 80 columnas y en cada elemento se almacena en un formato binario, el atributo de cada caracter en la pantalla; la codificación se muestra en la figura 5-3. Se recomienda consultar el manual de las terminales VT100 [7], para tener una visión más exacta de cómo se manejan los atributos y los caracteres gráficos.

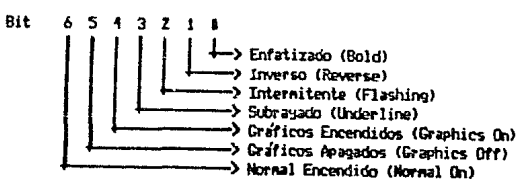


Figura 5-3

Otro punto en que difiere el pseudocódigo de Abre Ventana y la versión en VAX es que el borde de la ventana no se dibuja dentro de ella, sino que el ancho y el alto especifican el espacio útil de la ventana. Esto implica que el cálculo de X2 y Y2, difiere ligeramente para contemplar el borde de la ventana y el título; de tal suerte que Y2 está cuatro espacios más abajo (3 para el borde y 1 para el título) y X2 está dos espacios más a la derecha (2 para el borde).

DISEÑO DE LA INTERFAZ INTERACTIVA

Dado que la instrucción WRITE de FORTRAN-77, no puede actualizar las dos matrices de mapeo de la pantalla cada vez que se realiza una escritura, se decidió elaborar una rutina básica de salida. Esta rutina MVT_EscribePantalla, la cual se detalla en la siguiente sección, se encarga de reconocer qué secuencias de "ESCAPE" vienen en la cadena de caracteres que va a escribir, las codifica en binario y las almacena en la Matríz de Atributos, en la posición del cursor, indicada por los parámetros Renglón y Columna. De igual forma, almacena en la Matríz de Mapeo, la información proveniente de la cadena de caracteres, en el renglón y columna especificados por los paramétros. De esta forma, todas las escrituras a la pantalla de hacen a través de esta rutina básica de salida, y así se mantienen actualizadas las matrices internas, para que concuerden exactamente con la información que existe en el video.

Otra rutina más de la cual carece el sistema VAX, es la que define la Frontera que delimita el video. Es cierto que VAX sí proporciona un manejo de zonas de "SCROLL", que permiten trabajar en una porción de pantalla. Pero la zona de "SCROLL" solo define el renglón inicial y final de la sección de la pantalla de trabajo, y asume que se emplean las 80 columnas. Una rutina de Frontera es mucho más completa, permite delimitar una zona comprendida por un renglón inicial y otro final y las columnas inicial y final. La Frontera obliga que toda escritura y lectura, jamás cruce los bordes de la ventana y que cuando se sobrepase el último renglón de ella, se realice un movimiento vertical ascendente de todos los renglones de la ventana (Scrolling); de una manera análoga como sucede en toda la pantalla. Sin embargo, esta tarea era demasiado complicada para la rutina básica **MVT_EscribePantalla**. Por tal motivo, se realizaron dos rutinas más elaboradas, una de escritura **MVT_ImprimeTexto** y otra de lectura, **MVT_LeeTexto**; estas rutinas respetan la Frontera y hacen "Scrolling" si es necesario. La rutina de escritura detecta si la cadena a escribir no cabe completa en la ventana y la va escribiendo por secciones en cada renglón. La rutina de lectura detecta caracter por caracter y los va escribiendo en la ventana. Cada vez que alcanza el borde derecho, avanza un renglón y continúa escribiendo los caracteres a partir del borde izquierdo de la ventana.

Otro problema que hubo que resolver fué que se detectó que era necesario tener una rutina que controlara el cursor. Es decir, que lo colocara en una posición en particular y pudiera reportar al usuario en qué posición se encontraba en un momento dado. Esto se debe fundamentalmente a que al abrir una nueva ventana es necesario guardar la posición en donde se encuentra el cursor en la ventana activa, para que cuando se cierre la nueva ventana, el cursor quede exactamente en la posición donde se encontraba, antes de abrir la ventana. Como en VAX sí es posible colocar el cursor en una posición, pero no es factible leer las coordenadas en donde se encuentra, se hizo indispensable llevar dos cursores: el externo, o el controlado por la propia teminal y el interno, supeditado al Manejador de Ventanas. El cursor interno lleva un registro exacto de la localización del cursor, de tal forma que sí es posible conocer en un momento dado la posición de éste. Las rutinas de lectura y escritura **MVT_ImprimeTexto** y **MVT_LeeTexto**, actualizan constantemente el cursor interno, mientras que la rutina básica **MVT_EscribePantalla** no tabaja con él, esencialmente para simplificar la tarea de esta rutina.

Por otro lado, se detectó que el atributo activo en una ventana, se perdía al abrir y cerrar una ventana. Lo cual implica, que además de la posición del cursor, debe también almacenarse el tipo de atributo activo que se encuentra en la ventana. El lugar idoneo para almacenar ambos datos es el Buffer; si se

recuerda, las primeras cuatro localidades están destinadas a guardar las coordenadas de la esquina superior izquierda y las dimensiones de la ventana. Ampliando este concepto, se emplean tres localidades extras para almacenar la posición del cursor (x,y) y el código del atributo activo de la ventana. De tal forma, que además de la información de la pantalla cubierta por la ventana, el Buffer se encarga de registrar 7 localidades extras de información, relativas a las características de la ventana.

Para probar el comportamiento del Manejador de Ventanas se realizó un prototipo; con él se detectó que la velocidad del manejador era muy lenta por el acceso a disco que hay que realizar para almacenar el Buffer, cada vez que se abre una ventana. Para mejorar el rendimiento de la versión definitiva en VAX, se decidió ampliar la capacidad del Buffer; de tal suerte, que se comporta como un disco pero en memoria y así, almacena en RAM, cada porción de la pantalla. Cada ventana envía su información al Buffer, y en éste, por medio de apuntadores se va colocando una tras otra, cada porción de la pantalla. De este modo, se evita tener que enviar el Buffer a disco y toma muy pocos segundos abrir y cerrar una ventana. Es importante hacer notar, que como el sistema operativo en VAX tiene un manejo de memoria virtual, efectivamente el Buffer sí se almacena en disco, pero sin que el Manejador de Ventanas tenga que ver con esto; es decir, depende exclusivamente del sistema operativo que el Buffer se guarde en disco. La siguiente rúbrica trata los aspectos relativos a las características de las rutinas, así como una descripción de ellas.

5.1.4 Rutinas Del Manejador De Ventanas

En este punto se describen las características de las rutinas del Manejador de Ventanas, como son el lenguaje de desarrollo, el pase de parámetros, etc; para finalizar con una lista de las rutinas, desglozando sus parámetros y adicionando una breve explicación de sus funciones.

El manejador se desarrolló en FORTRAN-77 empleando los principios de la programación estructurada. Como se apreció en los pseudocódigos de las rutinas Abre Ventana y Cierra Ventana, se requiere de una serie de variables globales, que de no serlo así, el usuario tendría que acarrear como parámetros. En lugar de usar variables globales, se optó por el uso de "COMMONS"; éstos permiten comunicación entre módulos, sin que el usuario tenga que intervenir pasando parámetros que no tienen sentido para él. Por otro lado, se sustituyó la necesidad de inicializar variables globales en el programa principal, por inicializaciones, de variables como BufUsado, en tiempo de compilación.

En la lista de rutinas se podrá apreciar el prefijo de identificación de este manejador: todas las rutinas empiezan con el genérico MVT. A continuación se proporciona la lista completa.

5.1.4.1 Rutinas Básicas

SUBROUTINE MVT_Inicia

Esta subrutina debe llamarse antes que cualquier otra del manejador de ventanas. Inicializa todos los estados internos requeridos para que opere el manejador. Además borra la pantalla, establece la Frontera que la delimita y se define a toda la pantalla como la primera ventana abierta.

INTEGER FUNCTION MVT_Abre (ColIzq, RenInf, Ancho, Alto, Rotulo)

```
Rotulo      CHARACTER * (*), Entrada
            ! Rótulo de la ventana
Alto        INTEGER * 4, Entrada
            ! Alto útil de la ventana
Ancho       INTEGER * 4, Entrada
            ! Ancho útil de la ventana
ColIzq      INTEGER * 4, Entrada
            ! Columna donde empieza la ventana en coordenadas absolutas
RenInf      INTEGER * 4, Entrada
            ! Renglón donde empieza la ventana en coordenadas absolutas
```

Esta función abre una ventana en las coordenadas **ColIzq**, **ColDer** y con el ancho y largo especificados. Además rotula la ventana con el contenido de **Rotulo** y coloca el cursor en la coordenada relativa (0,0), lo cual actualiza el cursor interno.

El ancho se mide en número de caracteres y especifica el espacio útil de la ventana en dirección horizontal. El alto se mide en renglones y especifica el espacio útil de la ventana en dirección vertical. Es por esto que el espacio real que ocupa la ventana son dos columnas más para el ancho (reservadas para el borde de la caja) y cuatro renglones mas a lo alto (reservados para el borde de la caja y el rótulo). Regresa 3 posibles valores:

0 No hubo error
1 Las coordenadas de las ventana están fuera de los límites de la pantalla
2 Las dimensiones de la ventana hacen que una porción quede fuera de la pantalla.

SUBROUTINE MVT_Cierra

Esta rutina cierra la última ventana activa. Restablece la pantalla con lo que estaba debajo de la ventana. Se encarga de dejar el cursor exactamente en la posición donde se encontraba en la ventana previa a la ventana que se cierra y restablece el atributo activo es esa ventana. Esto indica que esta rutina sí actualiza el cursor interno.

5.1.4.2 Rutinas De Entrada-Salida

SUBROUTINE MVT_EscribePantalla(Columna, Renglon, CadenaCar)

```
CadenaCar   CHARACTER * (*), Entrada
            ! Cadena de caracteres que contiene el texto a imprimir
Columna     INTEGER * 4, Entrada
            ! Columna en coordenadas absolutas en la que se
```

```
                !Imprimirá el texto en la pantalla
Renglon         INTEGER * 4, Entrada
                ! Renglón a imprimir el texto (coor. abs.)
```

Esta es la rutina básica de salida por la cual toda impresión a la pantalla debe realizarse. Esta rutina se encarga de actualizar la pantalla, además modifica las matrices de mapeo de la pantalla que lleva el MVT en memoria. Esta rutina acepta secuencias de ESCAPE dentro de la cadena de caracteres, las interpreta codificándolas en binario y finalmente las almacena en las matrices de mapeo. En las secuencias de ESCAPE no se permite el punto y coma (;). Esto es:

```
No es válido          Válido
   ESC[7;1m            ESC[7mESC[1m
```

Las coordenadas (columna, renglón) son en forma absoluta, es decir, el valor (1,1) corresponde a la esquina superior izquierda, y el valor (24,80) corresponde a la esquina inferior derecha. Es responsabilidad del usuario verificar que la longitud del texto no exceda los límites de la ventana activa.

A diferencia de **MVT_ImprimeTexto**, esta rutina no actualiza el cursor interno. Otras rutinas más completas de salida son : **MVT_EscRelPant** y **MVT_ImprimeTexto.**

SUBROUTINE MVT_EscRelPantalla(ColumnaRel, RenglonRel, CadenaCar)

```
CadenaCar       CHARACTER * (*), Entrada
                ! Texto a imprimir.
ColumnaRel      INTEGER * 4, Entrada
                ! Columna, relativa a la frontera, a escribir
                !CadenaCar
RenglonRel      INTEGER * 4, Entrada
                ! Renglón, relativo a la frontera a escribir
                !CadenaCar
```

Esta rutina se basa en **MVT_EscribePantalla**, tiene sus mismas propiedades, sólo que las coordenadas se dan en forma relativa a la frontera activa.

SUBROUTINE MVT_ImprimeTexto(Texto, Salta, Atributo, Permanece)

```
Texto           CHARACTER * (*), Entrada
                ! Texto a imprimir.
Atributo        INTEGER * 4, Entrada
                ! Se codifica de la siguiente manera:
                !BIT  6 5 4 3 2 1 0
                !     ¦ ¦ ¦ ¦ ¦ ¦ ¦
                !     ¦ ¦ ¦ ¦ ¦ ¦ ¦-- Enfatizado
                !     ¦ ¦ ¦ ¦ ¦ ¦-- Inverso
                !     ¦ ¦ ¦ ¦ ¦-- Intermitente
                !     ¦ ¦ ¦ ¦-- Subrayado
                !     ¦ ¦ ¦-- Graficos Encendidos
                !     ¦ ¦--- Graficos Apagados
                !     ¦--- Normal Encendido
Permanece       INTEGER * 4, Entrada
                ! 0 - no se conserva, 1 - el atributo se conserva
Salta           INTEGER * 4, Entrada
```

```
! 0 - salta renglón,  1 - no salta
```

Esta rutina permite imprimir cadenas de caracteres en donde se encuentre el cursor en la pantalla. Es una rutina más elaborada que **MVT_EscribePantalla** y **MVT_EscRelPant**. Permite que el cursor salte o no, un renglón después de efectuar la impresión. Realiza la escritura con el atributo indicado y permite que este atributo se conserve para otro tipo de operaciones como **MVT_EscribePantalla, MVT_EscRelPantalla** y **MVT_LeeTexto**.

Además verifica que el texto no exceda los límites de la frontera y realiza una segmentación de la cadena de caracteres a imprimir; de modo que la cadena ocupe tantos renglones como sea necesario. En caso de que la cadena esté en el límite superior de la ventana (la parte más cercana a la sección de abajo de la pantalla), esta rutina realizará un movimiento de líneas hacia arriba (scrolling), perdiéndose la primera línea y abriendo una línea en la parte superior de la ventana. A diferencia de **MVT_EscribePantalla**, actualiza la posicion del cursor interno.

El algoritmo de esta rutina puede ser tomado por el usuario para realizar una rutina de salida de acuerdo a sus necesidades.

```
SUBROUTINE MVT_LeeTexto ( TextoLeido )

TextoLeido      CHARACTER * (*), Salida
                ! Variable en a depositar los caracteres leídos
```

Esta rutina se encarga de detectar las teclas que oprime el usuario, mostrarlas en la pantalla y asignarlas a **TextoLeido**. Tiene dos teclas especiales, Delete y Enter (Return). Las cuales actúan de acuerdo a su funcionamiento normal en una VT100. Si es necesario realiza un Scrolling (subir un renglón hacia arriba toda la información de la ventana), en caso que el texto leído así lo demande.

Permite mostrar todos los caracteres que se escriban pero la variable **TextoLeido** solo asumirá desde el primero hasta el número de caracteres con que se declaró. Tambien actualiza la posición del cursor interno.

Esta es una rutina básica de entrada, rutinas más elaboradas se pueden crear a partir de ésta, de tal manera que se adecúe a las necesidades del usuario. Para operala se requiere llamar una vez a la función **KB_Init ()** en el programa principal.

5.1.4.3 Rutinas De Reporte Del Estado

```
SUBROUTINE MVT_LeeCursor (CurColRel, CurRenRel)

CurColRel       INTEGER * 4, Salida
                ! Columna del cursor (relativo a la frontera).
CurRenRel       INTEGER * 4, Salida
                ! Renglón del cursor (relativo a la frontera).
```

Esta rutina regresa las coordenadas, relativas a la Frontera, donde se encuentra el cursor interno. Las coordenadas (0,0) corresponden a la esquina de arriba de la Frontera. Las rutinas que sí actualizan el cursor interno son: MVT_Abre, MVT_Cierra, MVT_ColocaCursor, MVT_ImprimeTexto y MVT_LeeTexto.

```
SUBROUTINE MVT_LeeFrontera (ColIzq, RenInf, ColDer, RenSup)

ColIzq    INTEGER * 4, Salida
          ! Columna izquierda de la frontera (coord. absolutas)
ColDer    INTEGER * 4, Salida
          ! Columna derecha de la frontera (coord. absolutas)
RenInf    INTEGER * 4, Salida
          ! Renglón inferior (arriba) de la frontera (coord. abs.)
RenSup    INTEGER * 4, Salida
          ! Renglón superior (abajo) de la frontera (coord. abs.)
```

Esta rutina regresa los límites de la Frontera activa.

```
SUBROUTINE MVT_LeeAtributoAct (Atributo)

Atributo      INTEGER * 4, Salida
              ! Atributo activo codificado en binario
```

Esta rutina obtiene en forma codificada el atributo que se encuentra activo en la pantalla. La codificacion se puede apreciar en MVT_ImprimeTexto.

```
SUBROUTINE MVT_LeeLimPantalla (ColMinPant, RenMinPant, ColMaxPant,
                               RenMaxPant)

ColMaxPant      INTEGER * 4, Salida
                ! Columna del fin de la pantalla (coord. absolutas)
                !generalmente regresa el valor de 80
ColMinPant      INTEGER * 4, Salida
                ! Columna del inicio de la pantalla (coord. abs.)
                !generalemnte regresa el valor de 1
RenMaxPant      INTEGER * 4, Salida
                ! Renglón del fin de la pantalla (coord. abs.)
                !generalmente regresa el valor de 24
RenMinPant      INTEGER * 4, Salida
                ! Renglón del inicio de la pantalla (coord. abs.)
                !generalmente regresa el valor de 1
```

Esta rutina regresa las dimensiones totales de la pantalla.

5.1.4.4 Rutinas De Diversos Propósitos

```
SUBROUTINE MVT_AvanzaRenglon
```

Esta rutina mueve el cursor un renglón más abajo y al borde izquierdo de la Frontera. Si avanza más allá del último renglón de la Frontera, realiza un movimiento de todos los renglones hacia arriba (scrolling). Deja además en

blanco el último renglón de la Frontera.

SUBROUTINE MVT_Refresca

Esta rutina se encarga de refrescar, o redibujar, toda la pantalla. Es útil cuando por alguna razón se reciben mensajes del operador, del sistema, etc., la pantalla de la terminal se llena con información indeseable.

```
SUBROUTINE MVT_InsertaAtributo (IndIni, IndFin, CadCarSinAtrib,
                                CadCarAtribBin, CadCarConEsc, NumCar)

CadCarAtribBin  CHARACTER * (*), Entrada
                ! Cadena de caracteres con los atributos en binario
                !para cada caracter de CadCarSinAtrib
CadCarConEsc    CHARACTER * (*), Salida
                ! Cadena de car. con texto de CadCarSinAtrib (desde
                !IndIni hasta IndFin) y secuencias de Escape
                !de los atributos en binario de CadCarAtribBin
CadCarSinAtrib  CHARACTER * (*), Entrada
                ! Cadena de car. sin atributos, sólo con texto.
IndIni          INTEGER * 4, Entrada
                ! Indice inicial a considerar en CadCarSinAtrib
IndFin          INTEGER * 4, Entrada
                ! Indice final a considerar en CadCarSinAtrib
NumCar          INTEGER * 4, Salida
                ! Número de caracteres con que sale CadCarConEsc
```

Esta rutina inserta en una cadena de caracteres las secuencias de Escape requeridas para que cumpla con los atributos codificados. Se necesitan de las cadenas de caracteres que contienen el texto y los atributos codificados en binario (ver **MVT_ImprimeTexto**). Regresa otra cadena de caracteres con las secuencias de Escape insertadas dentro del texto. La cadena de caracteres se puede imprimir con las rutinas de salida **MVT_EscribePantalla**, **MVT_EscRelPantalla** y **MVT_ImprimeTexto**.

Esta rutina asegura que se inserten el mínimo número de secuencias de Escape para lograr obtener todos los cambios de atributo especificados por el parámetro **CadCarAtribBin**.

```
SUBROUTINE MVT_BorraPantalla (ColIzq, RenInf, ColDer, RenSup)

ColDer      INTEGER * 4, Entrada
            ! Columna derecha de inicio borrado (coord. absolutas)
ColIzq      INTEGER * 4, Entrada
            ! Columna izquierda de fin borrado (coord. absolutas)
RenInf      INTEGER * 4, Entrada
            ! Renglón de arriba de inicio borrado (coord. abs.)
RenSup      INTEGER * 4, Entrada
            ! Renglón de abajo  de fin borrado (coord. abs.)
```

Esta rutina borra la sección de la pantalla especificada por los parámetros.

```
INTEGER FUNCTION MVT_ColocaCursor ( ColRel, RenRel )

ColRel      INTEGER * 4, Entrada
```

```
          ! Columna a colocar el cursor (relativa a la frontera)
RenRel    INTEGER * 4, Entrada
          ! Renglón a colocar el cursor (rel. a la frontera)
```

Esta función coloca el cursor dentro de la ventana. Los parámetros indican la columna y el renglón que se desea, pero éstos se encuentran en forma relativa a la frontera; es decir, el valor (0,0) corresponde a la esquina de arriba (inferior) que delimita a la Frontera (vease IMVT_DefineFrontera). Esta rutina actualiza la posición del cursor interno.

5.1.5 Ejemplo.

El programa en el listado S-III pregunta si se desea que después de escribir el texto "Columna, Rengion, Ancho, Alto ?", el cursor avance un renglon; también lee el atributo con que se va a realizar la escritura en la ventana y finalmente pregunta si ese atributo se va a conservar para la lectura. Con esos datos abre consecutivamente tres ventanas; para cada ventanas el usuario debe proporcionar, separados por comas y sin dejar espacios en blanco, los valores de las coordenadas de la esquina inferior izquierda (columna, renglón) y las dimensiones de la ventana: ancho y alto. Para que el usuario detecte el número de ventanas abiertas, se escribe un número indicando qué ventana se abrió.

Una vez que abre la tercera ventana, el programa pide que se lea un texto cualquiera, cuando el usuario oprime RETURN, la ventana se cierra. Este proceso se repite para las dos ventanas restantes. En la figura S-4 se muestra la pantalla resultante para una corrida del programa.

```
 PROGRAM ejemplo
IMPLICIT NONE

INTEGER
1       Ancho,
1       Alto,
1       Atri,
1       Condicion,
1       Col,
1       KB_Init,
1       MVT__Abre,
1       Permanece,
1       Ren,
1       Salta,
1       Veces

CHARACTER
1       Graficos * 3,
1       Rotulo * 80,
1       Texto * 80,
1       VecesCar * 1

WRITE (6,'(9,A)') ' Valor de salto?'
READ  (5,*) Salta
```

```
WRITE (6,'($,A)') ' Atributo?'
READ  (5,*) Atri

WRITE (6,'($,A)') ' Permanece?'
READ  (5,*) Permanece

Condicion = Kb_Init()
CALL MVT__Inicia

DO WHILE (.TRUE.)

    WRITE (6,'($,A)') ' Col, Ren, Ancho, Alto ?'
    READ  (5,*) Col, Ren, Ancho, Alto
    WRITE (6,'($,A)') ' Rotulo de la Ventana ?'
    READ  (5,'(A)') Rotulo

    Condicion = MVT__Abre (Col, Ren, Ancho, Alto, Rotulo)

    !Abre dos ventanas mas
    DO Veces = 1, 2

        IF (Condicion .eq. 0) THEN
            !Si se pudo abrir la ventana

            !Indica que numero de ventana se abrio
            WRITE (VecesCar,'(BN,I1)') Veces + 1
            CALL MVT__ImprimeTexto (VecesCar, 0, 96, 0)

            !Lee las dimensiones de la siguiente ventana que
            !se a abrir
            CALL MVT__ImprimeTexto ('Col, Ren, Ancho, Alto ?',
                 Salta, Atri, Permanece)
            CALL MVT__LeeTexto(Texto)
            READ (Texto, '(BN, 4I4)' ) Col, Ren, Ancho, Alto

        ENDIF

        !Abre la siguiente ventana
        Condicion = MVT__Abre (Col, Ren, Ancho, Alto, Rotulo)

     ENDDO

     !Cierra las tres ventanas
     DO Veces = 1, 3

        !Lee en la ventan activa un texto
        CALL MVT__LeeTexto(Texto)

        !Cierra la ventana
        CALL MVT__Cierra

    ENDDO

ENDDO
```

```
END
```

Listado 5-III

```
Col, Ren, Ancho, Alto 710,10,10,10
Rotulo de la Ventana ?Ventana
```

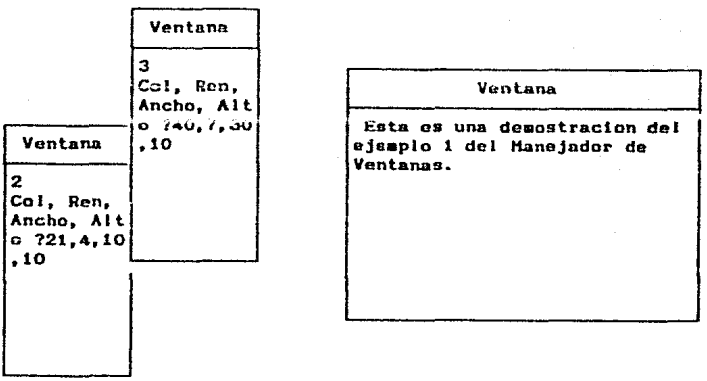


Figura 5-4

5.2 MANEJADOR DE ZONAS

5.2.1 Generalidades

A lo largo de esta seccion se tratará el tema del Manejador de Zonas; incluyendo una descripción de sus funciones y de las razones que motivaron su realización; los pseudocódigos más importantes y finalmente una lista de las rutinas que integran al manejador.

El Manejador de Zonas es en realidad una parte del Manejador de Ventanas. Sin embargo, su elaboración es cronológicamente posterior al Manejador de Ventanas; es por esto que se puede considerar como una adición a este sistema.

Los manejadores de Menús, Diálogos, Teclas Funcionales, Anidamientos y de Captura, motivaron en realidad, la elaboración del Manejador de Zonas. Como se mencionó en el capítulo 4, la pantalla en el Sistema Interactivo, está organizada en cuatro secciones, independientes unas de las otras. Estas secciones, o zonas, funcionan como regiones de SCROLL; es decir, cuando se llega al último renglón de una zona, todos los renglones se mueven hacia arriba, de tal suerte que el primer renglón de esa zona se pierde. También, la posición del cursor debe permanecer inalterable; de tal forma que el Manejador lleva el control del cursor para cada zona; esto implica que pasar de una zona a otra, no

causa ningún efecto en alguna de las otras zonas restantes. Sólo existe una zona activa a un tiempo; es por esto que exclusivamente en una zona de la pantalla a la vez, se puede estar escribiendo o leyendo. Sin embargo, sí es posible dejar "congelada" una zona, para saltar a otra y tabajar en ella.

En un principio se pensó, que esto se podría resolver con varias ventanas abiertas, con la restricción de que no se traslaparan. Desafortunadamente, el Manejador de Ventanas, permite sólo una ventana activa, y esa es la última que se acaba de abrir. Además el concepto de zona demanda de una mayor capacidad, se requiere que diversas porciones de la pantalla estén activas, una a la vez, a petición del usuario.

Una zona es entonces, como una pequeña pantalla, la cual está delimitada, y dentro de esos límites se realiza un "scrolling local". Sólamente que esa porción de la pantalla se puede alojar en cualquier lugar del video. Difiere de la ventana en que una zona no guarda lo que cae debajo de ella y tampoco dibuja un recuadro. De hecho, únicamente delimita la Frontera de la pantalla, sobre la cual va a operar. Bajo este principio los manejadores de Menús, Dialogos, Teclas Funcionales, Anidamientos y Captura, serían capaces de definir la porción del video en la cual van a trabajar, con la seguridad de que la integridad de su información en la pantalla, va a ser respetada. Es de esta manera, que el Manejador de Zonas permite organizar la pantalla, dividiéndola en varias regiones independientes; tal y como los requerimientos del resto de los manejadores lo plantean.

Una de las características del Manejador de Zonas, al igual que el de Ventanas, es que es transparente para el usuario; éste jamás se entera si está trabajando en la pantalla completa o solamente en una porción de ella; siempre y cuando se empleen las rutinas de entrada y salida del Manejador de Ventanas. Además, el usuario no se debe preocupar por la posición del cursor, al reactivar una zona; pués éste es automáticamente localizado. De igual forma, si la escritura se realiza con un atributo en particular (enfatizado, inverso, etc.), este atributo se recupera automáticamente al reactivarse la zona.

5.2.2 Algoritmo

Son tres las rutinas básicas del Manejador de Zonas: Define Zona, Borra Zona y Activa Zona; las tres se fundamentan en el control de la Frontera. Se recordará que en la sección del Manejador de Ventanas, se trató el concepto de Frontera; la cual tiene todas las propiedades de una zona en la pantalla. Por esto, las rutinas del Manejador de Zonas lo que simplemente hacen, es definir una Frontera con las dimensiones de la zona y dejar que el usuario opere esa zona con las rutinas de entrada y salida MVT_ImprimeTexto y MVT_LeeTexto, las cuales sí respetan los límites estipulados para una Frontera. El listado 5-IV muestra los pseudocódigos de Define Zona, Borra Zona y Activa Zona.

Las zonas se manejan a través de números enteros consecutivos; esto es: 1, 2, 3, 4, 5, etc. De tal manera que si se tienen 6 zonas en la pantalla y se desea trabajar con la segunda, se le indica al Manejador de Zonas, que active la zona 2. la rutina **Define Zona**, almacena en una lista, las dimensiones de la Frontera que delimitan a una zona. Además, regresa un número con el cual se hará referencia posteriormente a la zona. Es importante hacer notar que los valores de la lista tienen un valor inicial de -1 y que esta lista (ColIzq, ColDer,

RenInf, RenSup), la conforman variables globales que se comparten por las tres rutinas.

Borra Zona, es la contraparte de **Define Zona**; es decir, sirve para dar de baja una zona. Es sumamente útil cuando ya no se va a trabajar mas con una zona, o se desea liberar espacio de la lista de zonas. Borra zona, lo que sencillamente hace es indicar que la zona, para el número especificado, se encuentra disponible, asignándole un -1.

Finalmente, Activa Zona es la razón por la cual existe Define Zona. Una zona definida se puede activar, o trabajar con ella, llamando a Activa Zona. A ésta se le pasa un número de zona, para que llame a Frontera con los límites almacenados en la lista, de acuerdo al número de zona pasado como parámetro. Una vez hecho esto, toda escritura o lectura realizadas a través de MVT_ImprimeTexto o MVT_LeeTexto, estarán delimitadas a la Frontera de la zona activa.

Es importante hacer notar que el tener una zona activa no implica que no se pueda abrir una ventana; pero hay que recordar que el cerrar una ventana no borra a una zona.

5.2.2.1 Pseudocódigos

```
DefineZona (X1, Y1, X2, Y2, NumeroZona)

! Busca una zona disponible
Zona = 1
DO WHILE ( ColIzq (Zona) <> -1 )
   Zona = Zona + 1
ENDDO

! Almacena las dimensiones de la Frontera
ColIzq (Zona) = X1
ColDer (Zona) = X2
RenInf (Zona) = Y1
RenSup (Zona) = Y2

NumeroZona = Zona

END


BorraZona (NumeroZona)

! Da de baja a la Zona
ColIzq (NumeroZona) = -1

END
```

```
ActivaZona (NumeroZona)

Frontera (Collzq (NumeroZona), Reninf (NumeroZona), ColDer (NumeroZona),
          RenSup (NumeroZona) )

END
```

Listado 5-IV

5.2.3 Rutinas.

En este punto se proporciona una lista de las rutinas que integran al Manejador de Zonas, incluyendo una descripción breve de su función y de los parámetros que requiere cada una.

Las rutinas están elaboradas en FORTRAN 77, empleando los principios de la programación estructurada. Al igual que el manejador de Ventanas, substituye las variables globales por medio de "COMMONS", y como es parte del Manejador de Ventanas, todas las rutinas tienen el prefijo MVT. A continuación se proporciona la lista de las rutinas.

SUBROUTINE MVT_DefineZona (Collzq, Reninf, ColDer, RenSup, NumZonaAsig)

```
Collzq      INTEGER * 4, Entrada
            ! Columna izquierda de la frontera que delimita la zona
            !(coordenadas absolutas)
ColDer      INTEGER * 4, Entrada
            ! Columna derecha de la zona (coord. abs.)
Reninf      INTEGER * 4, Entrada
            ! Renglón de arriba de la frontera que delimita la
            !zona (coordenadas absolutas)
RenSup      INTEGER * 4, Entrada
            ! Renglón de abajo de la zona (coord. abs.)
NumZonaAsig INTEGER * 4, Salida
            ! Número de zona asignada. Regresa -1 si no hay
            !espacio disponible
```

Esta rutina define una zona delimitada por **Collzq, ColDer, Reninf** y **RenSup**, que en última instancia, forman una nueva Frontera. Busca un número de zona (NumZonaAsig) que se le asigna y a partir del cual se hará referencia a esta zona. Ver **MVT_BorraZona, MVT_ActivaZona, MVT_AlmacenaZona** y **MVT_ExtraeZona.**

Esta rutina no actualiza el cursor de zonas, a diferencia de **MVT_ActivaZona, MVT_AlmacenaZona, MVT_BorraZona** y **MVT_ExtraeZona.**

Es responsabilidad del usuario no traslapar zonas. Ya que la zona activa no respetará la frontera de la zona traslapada. Es posible hacer esto para efectos especiales.

Regresa un número positivo, correspondiente al número de zona asignado. Regresa -1 si no hay mas espacio en memoria para asinarle un número a esta zona. Es posible tener 10 zonas simultáneas como máximo. Si regresa -1 se recomienda usar **MVT_BorraZona** para liberar espacio.

SUBROUTINE MVT_BorraZona (NumZona)

NumZona INTEGER * 4
! Número asignado a la zona que se desea dar de baja.

Esta rutina da de baja la zona especificada por **NumZona**. Además guarda la posición del cursor de la última zona activa (si hay alguna).

Sólo las rutinas **MVT_BorraZona**, **MVT_ActivaZona**, **MVT_AlmacenaZona** y **MVT_ExtraeZona** actualizan el cursor del manejador de zonas. Ver la observación de **MVT_ActivaZona**.

SUBROUTINE MVT_ActivaZona (NumZona)

NumZona INTEGER * 4
! Zona que se desea activar.

Esta rutina activa una zona; es decir, establece una Frontera, definida en **MVT_DefineZona**. Guarda las coordenadas del cursor de la zona previamente activa (si la hay) y coloca el cursor donde le corresponde en la zona activa.

Si el programa tiene un comportamiento extraño por usar las zonas en combinación con **MVT_Abre** o **MVT_Cierra**, se recomienda que las llamadas a estas dos rutinas estén, sin importar el orden, acompañadas de **MVT_ActivaZona** o **MVT_BorraZona**. Si esto no funciona, se puede emplear **MVT_AlmacenaZona** y **MVT_ExtraeZona**.

SUBROUTINE MVT_AlmacenaZona ()

Esta rutina guarda el número de zona que está activa y la posición de donde se encuentra el cursor. Se recomienda el uso de esta rutina cuando se desea activar otra zona y se quiere que al terminar de usar la nueva zona, la anterior, de la cual no se conoce su número, se restablezca. Para restaurar esa zona anterior, simplemente se llama a **MVT_ExtraeZona**, siempre y cuando antes de definir la zona nueva se haya llamado a **MVT_AlmacenaZona**.

SUBROUTINE MVT_ExtraeZona ()

Esta rutina extrae la zona almacenada. Se usa en combinación de **MVT_AlmacenaZona**. Esta rutina y **MVT_ActivaZona** sí almacenan el cursor de zonas.

5.2.4 Ejemplo.

Para finalizar este tema se muestra en el listado 5-V un programa, el cual almacena la posición del cursor con **MVT_AlmacenaZona**, abre dos ventanas, extrae las dimensiones de la Frontera y establece, con esos valores, dos zonas que caen exactamente encima de las dos ventanas. Va realizando lecturas de textos en cada una de las zonas; de tal manera que lo que lee en una zona lo imprime en la otra. Cuando el usuario escribe **fin** en la segunda zona, el programa borra la zona 1 y escribe **hola** en la zona 2. Con esto se quiere demostrar que la rutina **MVT_BorraZona** sí almacena el cursor interno de las zonas. Después activa la zona 1, pero como se borró en la operación anterior, al escribir el texto **hola**, éste aparece en la zona 2. Una vez hecho esto, el programa borra la zona 2, y al

intentar activar esta zona y escribir **hola**, el texto aparece en donde se encuentra el cursor, que es en la zona 2. Es importante hacer notar que el borrar una zona, no provoca que una ventana se cierre; es decir, las ventanas siguen existiendo, pero las operaciones de entrada y salida hacen referencia a la ventana activa, que es la segunda. Por consiguiente, el caso inverso, cerrar las ventanas no provoca que las zonas se borren automáticamente.

En la parte final del programa se extrae la posición del cursor que se almacenó en un principio, escribe **hola** y lee un texto para demostrar en dónde se encuentra el cursor. Posteriormente escribe y lee "basura"; es decir, texto que no se escribe con las rutinas MVT de entrada y salida. Esto puede ocurrir con mensajes del sistema o del operador que aparecen en el video en forma indeseable. Cuando se llama a la rutina **MVT_Refresca**, esta información indeseable desaparece. Por último, ambas ventanas se cierran. En la figura 5-5 se muestra el resultado en la pantalla que produce el programa al ejecutarse.

```
PROGRAM Ejemplo
IMPLICIT NONE

INTEGER
1       Condicion,
1       ColDer,
1       Collzq,
1       Kb_Init,
1       MVT_Abre,
1       Reninf,
1       RenSup,
1       ZonaVent1,
1       ZonaVent2

CHARACTER
1     Texto*20

Condicion = Kb_Init ()
CALL MVT_Inicia

!Almacena la posicion del cursor
CALL MVT_AlmacenaZona()

!Abre la ventana 1 y la define como zona
Condicion = MVT_Abre ( 2, 2, 30, 10, 'Zona 1')
CALL MVT_LeeFrontera (Collzq, Reninf, ColDer, RenSup)
CALL MVT_DefineZona  (Collzq, Reninf, ColDer, RenSup, ZonaVent1)

!Abre la ventana 2 y la define como zona
Condicion = MVT_Abre ( 40, 2, 30, 10, 'Zona 2')
CALL MVT_LeeFrontera (Collzq, Reninf, ColDer, RenSup)
CALL MVT_DefineZona  (Collzq, Reninf, ColDer, RenSup, ZonaVent2)

!Activa la zona 2 y pide al usuario que introduzca un texto
CALL MVT_ActivaZona ( ZonaVent2 )
CALL MVT_ImprimeTexto ( 'Escribe un texto ?', 1, 0, 0 )
CALL MVT_LeeTexto ( Texto )
```

```
! Va a leer textos en las ventanas, hasta que en la ventana de la zona
!2, el usuario escriba "fin" con letras minusculas

DO WHILE (Texto .ne. 'fin')

   ! Activa la zona 1; imprime el texto, leido en la zona 2, y lee en
   !esta zona un texto
   CALL MVT_ActivaZona ( ZonaVent1 )
   CALL MVT_ImprimeTexto (Texto, 0, 0, 0)
   CALL MVT_ImprimeTexto ('Escribe un texto ?', 1, 0, 0)
   CALL MVT_LeeTexto (Texto)

   !Activa la zona 2; imprime el texto, leido en la zona 1, y lee en
   !esta zona un texto
   CALL MVT_ActivaZona ( ZonaVent2 )
   CALL MVT_ImprimeTexto (Texto, 0, 0, 0)
   CALL MVT_ImprimeTexto ('Escribe un texto ?', 1, 0, 0)
   CALL MVT_LeeTexto (Texto)

ENDDO

!Esto prueba como se guarda el cursor por la rutina MVT_BorraZona
!Borra la zona 1 y escribe "hola" en la zona 2
CALL MVT_BorraZona  (ZonaVent1)
CALL MVT_ActivaZona (ZonaVent2)
CALL MVT_ImprimeTexto ('hola', 0, 7, 0)

!Activa la zona 1, pero como ya se borro, escribe "hola" en la
!zona 2
CALL MVT_ActivaZona (ZonaVent1)
CALL MVT_ImprimeTexto ('hola', 0, 7, 0)

!Borra la zona 2, pero como ya se borro, escribe "hola" en donde se
!encuentra el cursor
CALL MVT_BorraZona (ZonaVent2)
CALL MVT_ImprimeTexto ('hola', 0, 7, 0)

!Recupera la posicion del cursor, guardada al principio del programa
!y ahi escribe "hola"; una vez hecho esto lee un texto
CALL MVT_ExtraeZona
CALL MVT_ImprimeTexto ('hola', 0, 7, 0)
CALL MVT_ImprimeTexto ('Escribe un texto ?', 1, 6, 1)
CALL MVT_LeeTexto (Texto)

!Escribe basura, y lo borra al refrescar la pantalla
WRITE (6,*) 'Esto es basura, no se escribio con las rutinas de '//
1           'MVT; se va a perder'
WRITE (6,*) 'con el Refresco. El texto que se va a leer ' //
1           'tambien es basura'
WRITE (6,'($,A)') ' Escribe un texto ?'
READ (5, '(A)') Texto
CALL MVT_Refresca

!Cierra ambas ventanas
```

```
CALL MVT_Cierra
CALL MVT_Cierra

END
```

Listado 5-V

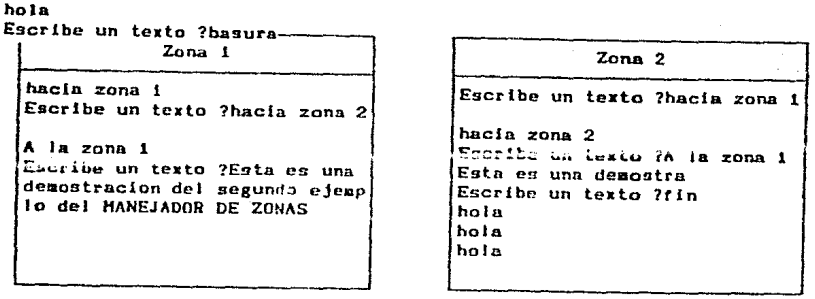


Figura 5-5

Cuando se manejan zonas y ventanas pueden sucitarse resultados un poco extraños. Se recomienda que para evitarlos, siempre que se abra una ventana y se defina una zona que caiga encima de ella, al cerrar dicha ventana, inmediatamente después se borre la zona; ya que si no, las operaciones de entrada y salida, seguirán produciéndose en la zona, apesar de que ya no exista la ventana. En caso de que el cursor no quede colocado donde se esparaba, lo mejor es emplear **MVT_AlamcenaZona** antes de definir la zona y **MVT_ExtraeZona**, después de borrar la zona.

Vale la pena hacer notar que tampoco es indispensable abrir una ventana para definir una zona; de hecho, pueden existir varias zonas sin que exista una sola ventana abierta.

5.3 RUTINAS DE DETECCION DE TECLADO (KB)

5.3.1 Descripción

Las rutinas que a continuación se presentarán no forman parte del trabajo de esta tesis (SER). Se tomaron de un trabajo realizado con anterioridad en el Centro de Cálculo de la Facultad de Ingeniería. Estas rutinas forman parte de la tesis denominada "Diseño Auxiliado por Computadora de Sistemas Eléctricos de Potencia (SELPOT)" desarrollada por los ingenieros Craige Snader Gonzalez y Adrián Alvarez Martínez.

El conjunto de rutinas (KB) se utiliza para detectar desde la simple presión de una tecla hasta la detección de secuencias de teclas utilizando un buffer (type ahead buffer).

Debido a que nuestras necesidades sólo requerían de detección de teclado elemental, únicamente se utilizaron tres de las quince rutinas que conforman la familia de rutinas KB. Por lo tanto solo se hará mención a las rutinas empleadas en nuestro sistema SER. La mayoría de los manejadores del sistema interactivo SER hacen uso de este conjunto de rutinas.

5.3.2 Rutinas

```
INTEGER*4 FUNCTION KB_Init ( )
```

Inicializa las variables comunes al sistema KB y permite sea utilizado el resto de las rutinas KB. Se invoca una sola vez, antes de comenzar a utilizar las rutinas. La función regresa el estado de la ejecución: 0 --> éxito , 1 --> fracaso.

```
INTEGER*4 FUNCTION KB_Read
```

La función anterior hace la lectura de una tecla presionada y regresa como resultado el código de la tecla. Existe un código interno al sistema KB para cada una de las teclas. Si está encendida la opción de "type ahead buffer" se agiliza la detección.

```
INTEGER*4 FUNCTION KB_Set_Tab ( Bandera )

        LOGICAL     Bandera !Indica si debe estar encendida la
                            !opcion de "type ahead buffer".
```

La función anterior tiene la finalidad de indicar si debe ser utilizada la opción de "type ahead buffer". Los valores que puede tomar la bandera son:.TRUE. --> opción encendida, .FALSE. --> opción apagada. La opción de buffer funciona para las detecciones de teclado posteriores.

5.3.3 Operación

El listado 5-VI muestra la forma en que se utilizan las rutinas de detección de teclado.

```
PROGRAM DetectaESC
! Este fragmento de programa inicializa las rutinas de
! detección y entra a un ciclo que termina cunado se
! detecta que se ha presionado la tecla <ESC>.

INTEGER*4
1    KB_Init,
```

```
1     KB_Read,
1     KB_Set_Tab,
1     Status,
1     Tecla

Status = KB_Init ()
Tecla = KB_Read ()
DOWHILE (Tecla .NE. 27)
    !El codigo 27 pertenece a la tecla <ESC>
    TYPE , 'Tecla invalida'
    Tecla = KB_Read ()
ENDDO
END
```

Listado 5-VI

5.4 MANEJADOR DE AYUDA (MAY)

5.4.1 Descripción

Proporcionar rutinas para el despliegue de texto sobre una ventana de la pantalla con la opción de definir tópicos que puedan ser selecionados por el usuario. A la función que realiza este manejador se le conoce como "Documentación en Linea".

Las ayudas que se presentan en la pantalla pueden estar divididas en tópicos o no. En caso de contar con tópicos se presentan dos ventanas de despliegue (la ventana de tópicos y la de texto). En caso de no existir tópicos, únicamente se presenta la ventana de texto.

En el archivo de texto se incluyen comandos propios del manejador. Donde se indicar, entre otras cosas, las coordenadas de las ventanas. La ventana de tópicos es un menú por medio del cual el usuario selecciona la información que desea ver en la ventana de texto. En la ventana de texto se despliega, página a página, el contenido del archivo de texto.

Es importante recordar que este manejador hace uso de las rutinas del manejador de ventanas, manejador de menús y del detector de teclado.

5.4.2 Rutinas

Al usuario se le proporcionan dos rutinas:

```
INTEGER*4 FUNCTION MAY_Inicia ( NombreArchivo )

      CHARACTER *(*) NombreArchivo  !Contiene el nombre
                                    !del archivo de texto.
```

Esta rutina permite indicar al manejador de ayudas cual es el archivo que debe tomar como archivo de texto. Esta rutina debe llamarse cada vez que se desee tomar otro archivo como archivo de texto. Si la rutina regresa un valor de uno, indica que hubo un error, si el resultado es cero, se indica que la rutina se ejecutó con éxito.

```
SUBROUTINE MAY_DespliegaOpciones ( NumeroOpcionesActivas )

      INTEGER*4 NumeroOpcionesActivas !Es el número de opciones
                                      !que se desplegarán en el
                                      !menú de tópicos.
```

Esta rutina abre las dos ventanas que utiliza el manejador: ventana de tópicos y ventana de texto. Al usuario se le presenta el menú de tópicos para que seleccione el que desea, hasta que pida salir del menú. Al salir del menú desaparecen las dos ventanas. Si el número de opciones activas es menor a cero, se asume que se quiere desplegar todo el texto del archivo, es decir, no existen tópicos.

5.4.3 Operación

En esta sección se listarán las convenciones que exige el manejador y se mostrará un ejemplo del uso del mismo.

5.4.3.1 El Archivo De Texto

El manejador requiere del uso de comandos inmersos en el propio archivo de texto. Todos los comandos del manejador se deben comenzar a escribir con un punto (.) en la columna uno (1) del archivo de texto. Los comandos XO, YO, XT, YT, ANCHOT, LARGOT y TITULO deben aparecer antes del primer comando TOPICO o TEXTO.

Los comandos que reconoce el manejador de ayudas se describen a continuación.

```
                .XO;entero-1
                .YO;entero-2
      donde:
                0 <= entero-1 <= 79
                0 <= entero-2 <= 24
```

(XO, YO) definen las coordenadas, referidas a la pantalla de video, de la esquina superior izquierda de la ventana de opciones o tópicos. El largo y ancho de la ventana de tópicos se calculan dinámicamente, dependiendo del número de tópicos y del número de caracteres de los textos de los mismos. La figura 5-6 muestra el uso de los comandos XO y YO.

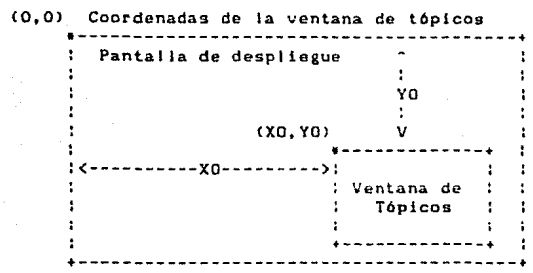


Figura 5-8

```
.XT;entero-1
.YT;entero-2
.ANCHOT;entero-3
.LARGOT;entero-4
```

donde:

```
0 <= entero-1 < 80
0 <= entero-2 < 25
0 <  entero-3 < 80
0 <  entero-4 < 25
```

Estos cuatro comandos definen las dimensiones y posición de la ventana de texto. (XT, YT) definen las coordenadas de la esquina superior izquierda de la ventana y (ANCHOT, LARGOT) el ancho y largo de la misma. El usuario debe cuidar que las ventanas no se vayan a traslapar. La figura 5-7 muestra el uso de los comando (XT, YT, ANCHOT y LARGOT).

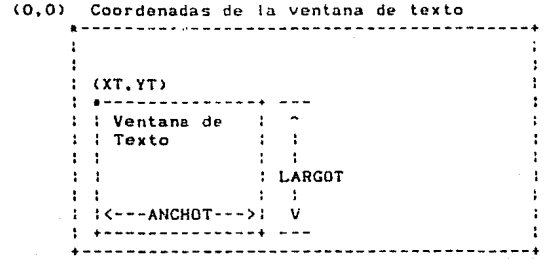


Figura 5-7

```
.TITULO;texto
```

Define el texto con el que se rotula la ventana de tópicos.

.TOPICO (entero);texto

Define una de las opciones que aparecerán en el menú de tópicos. Donde:

texto : Es el rótulo que aparecerá en la ventana de tópicos como una opción del menú.

entero : Número que se asocia a este tópico.

Debe existir un comando .TOPICO para cada opción del menú de tópicos.

.TEXTO (entero)

Indica que el texto que se encuentra a partir de la línea de este comando, hasta el encuentro de otro comando .TEXTO o fin de archivo, pertenecerá al tópico asociado con el valor del parámetro "entero".

.INCLUYE;nombre-archivo

Indica al manejador que el análisis del archivo de texto debe continuar en otro archivo (en el rotulado por el nombre "nombre-archivo") y que cuando se encuentre el fin regrese al archivo original. Esta opción es útil para tener los textos de los tópicos, que se utilicen en varios menús de ayudas, separados en diferentes archivos.

.PAGINA

El comando PAGINA provoca que al estar desplegando los textos de ayuda en la ventana correspondiente, se produzca un fin de página. Es decir, el despliegue de texto se interrumpe y se forza al manejador a desplegar el resto en una nueva página de la ventana.

NOTA: En caso de no incluir en los archivos de texto los comandos XO, YO, XT, YT, ANCHOT o LARGOT se asumirán valores default, que son los siguientes;

```
.XO;45
.YO;1
.XT;1
.YT;1
.ANCHOT;20
.LARGOT;10
.TITULO;AYUDA
```

El listado 5-VII presenta un archivo típico de texto y el efecto que produciría al ser procesado por el manejador de ayudas.

```
;-------- columna numero 1 del archivo de texto
V
.XO;40
.YO;12
.XT;1
.YT;2
.ANCHOT;28
.LARGOT;11
.TITULO;AYUDAS
.TOPICO(1);Operacion de los Menus
.TOPICO(2);Tablas de datos
.TOPICO(3);Bitacoras
.TEXTO(1)
.INCLUYE;ser$ayudas:menu.may
.TEXTO(2)
          TABLAS DE DATOS
     Los datos que se toman para
dibujar una grafica deben estar
contenidos en una tabla. El numero de
tablas de datos que conforman una
grafica dependen del tipo de esta.
Existen dos tipos de tablas de datos:
NUMERICAS y ALFANUMERICAS.

     Dentro de la opcion "INTRODUCIR Y
EDITAR TABLAS DE DATOS" se puede crear
una nueva tabla, modificar el contenido
de una ya existente o eliminar alguna.
.TEXTO(3)
                BITACORAS
  Una BITACORA es un archivo que crea
el propio sistema SER con el fin de
almacenar la informacion de una sesion
de trabajo (Tablas de datos y graficas)
y poder utilizarla en otra ocasion.

    La Opcion "USAR BITACORAS DE
SESIONES DE TRABAJO" permite utilizar
una bitacora previamente creada o crear
una nueva.
```

Listado 5-VII

Si se seleccionara la opcion de "TABLAS DE DATOS", en el momento de desplegar en la pantalla las ventanas de ayudas, se vería lo siguiente. Ver figura 5-8.

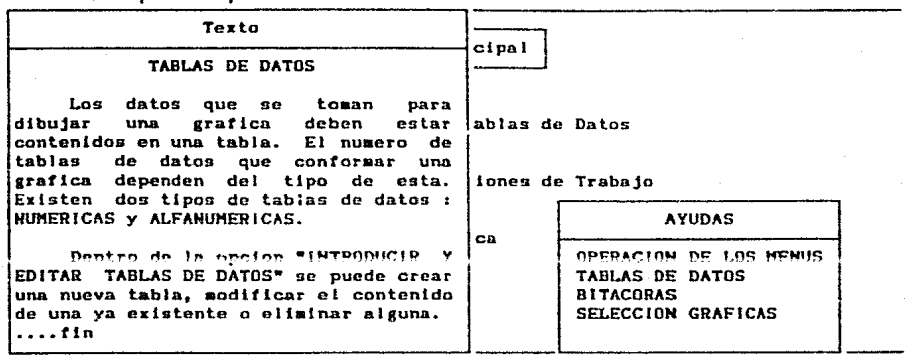


Figura 5-8

5.4.3.2 Llamadas A Las Rutinas

El Listado 5-VIII muestra la forma en que se utilizan las rutinas del manejador de ayudas.

```
PROGRAM Serinteractivo
IMPLICIT NONE
...

INTEGER
i   MAY_inicia,
...

CALL MVT_Inicia
Condicion = KB_Init ( )

!! Organizacion del despliegue
CALL MAN_Inicia (1)
CALL MMU_Inicia (1,2,80,19,1)
CALL MDI_Inicia (20,23)
CALL MTF_Inicia (24)

!! Pantalla de presentacion
Condicion = MAY_Inicia ('SER$AYUDAS:Present.MAY')
      ...
```

```
TermEmp = 2 !Ayuda
DO WHILE (TermEmp .EQ. 2)

    CALL MTF_Despliega (TeclasTex,3)
    CALL MDI_Lee (Terminadores, 2, -1, 0, 0,
   1              TeclaLeida, LongTeclaLeida, TermEmp)
    IF (TermEmp .EQ. 2) THEN
        !Ayuda
        CALL MTF_Parpadea (TeclasTex, 3, 2, 2)

        !! La rutina MAY_DespliegaTopicos abre las
        !! ventanas de ayuda (menu y texto) e introduce
        !! al usuario al manejador de ayudas.

        CALL MAY_DespliegaTopicos (5)

     !NOELSE
     ENDIF
ENDDO
IF (TermEmp .EQ. 0) THEN
    !<RETURN>
    CALL MenuPrincipal
!NOELSE
ENDIF
END
```

Listado 5-VIII

5.5 MANEJADOR DE DIALOGOS. (MDI)

En esta rúbrica se tratarán los detalles más sobresalientes relativos al Manejador de Diálogos. También se mostrará la lista de rutinas que conforman al Manejador y finalmente se proporcionará un programa que ejemplifica su uso.

5.5.1 Generalidades.

El Manejador de Diálogos controla el esquema de **pregunta-respuesta** en un sistema. Básicamente son un conjunto de cuatro rutinas, las cuales versan sobre la escritura y lectura de cadenas de caracteres en una zona de la pantalla. Para este efecto, previamente se definen los renglones inicial y final que conforman a la zona de la pantalla, denominada zona de diálogos.

Una característica esencial del Manejador de Diálogos es que lleva un registro de la posición del cursor y del atributo con que se realizó la última operación de entrada-salida. De tal suerte que permite transferir el control a otra zona y realizar una escritura o una lectura a la zona de diálogos sin afectar al resto de la pantalla. También lleva un control exacto de la posición, dentro de la zona de diálogos, donde le corresponde escribir o leer; por este motivo el usuario, que ha escrito en regiones externas a la zona de diálogos y desea escribir o leer algo dentro de la zona, no debe preocuparse, ya que cuando realice la operación de entrada-salida, el Manejador colocará el cursor

perfectamente en su posición.

5.5.2 Algoritmo.

El algoritmo es muy simple, existe una rutina de inicialización de diálogos, la cual recibe los renglones inicial y final que definen la porción de la pantalla destinada a los diálogos. Esa región funciona como una zona de "SCROLL", es decir, cuando se llega al último renglón de la zona, todos los renglones de la zona suben una posición, perdiéndose el primer renglón de la zona. Para llevar a cabo esto, simplemente se define una zona de 80 columnas y del número de renglones especificados por los parámetros; esto a través de la rutina **MVT_DefineZona** del Manejador de Zonas. Es este manejador el que realmente se encarga de restringir las operaciones de entrada-salida a la zona.

La rutina de lectura en la zona de diálogos es en verdad una ampliación de la rutina **MVT_LeeTexto** del Manejador de Ventanas. Almacena la zona con la rutina **MVT_AlmacenaZona**, para respetar la posición del cursor en la pantalla y activa la zona de diálogos con MVT_AotivaZon. LLeva un cursor interno que comparte con la rutina de escritura, para determinar cuál debe ser la posición del cursor dentro de la zona de diálogos. La lectura se realiza por medio de la rutina **KB_READ** de detección de teclado y permite, a discresión del usuario, mostrar o no en la pantalla, lo que se ha escrito con el teclado. De igual forma recibe una serie de "terminadores", los cuales son teclas que al detectarse que se han oprimido, la lectura finaliza; el RETURN es un terminador predefinido para cualquier lectura. También permite restringir la lectura a un determinado número de caracteres, y a disposición del usuario, puede terminarse la lectura cuando ya se han escrito ese número de caracteres. Por otro lado, es posible comenzar la lectura con un valor inicial; esto fundamentalmente para predecir respuestas y así facilitar el trabajo del usuario. Por ejemplo, una posible respuesta a "Borrar el archivo?" podría ser "no"; a lo cual si el usuario está satisfecho con la pregunta propuesta, simplemente debe oprimir RETURN; en caso contrario debe borrar la respuesta y escribir la que el usuario crea pertinente. Finalmente restablece la Frontera y la posición del cursor, fuera de la zona de diálogos con **MVT_ExtraeZona**.

La rutina de escritura también salva el cursor y la Frontera con **MVT_AlmacenaZona**, actualiza el cursor interno de diálogos y realiza la escritura con la rutina **MVT_ImprimeTexto**. Después restablece la posible zona activa, el cursor y la Frontera con **MVT_ExtraeZona**.

Como las rutinas de diálogos hacen uso del manejador de ventanas y de la rutina KB, es indispensable invocar a **KB_init** y **MVT_Inicia** antes de emplear a este manejador.

5.5.3 Rutinas.

Las rutinas están elaboradas en FORTRAN-77, empleando los principios de la programación estructurada. El uso de variables compartidas entre las rutinas de inicialización, lectura y escritura se implmentaron con el uso de "COMMONS", fundamentalmente para evitar que el usuario tuviera que emplear parámetros sin sentido para él. Todas las rutinas llevan el prefijo MDI que es la abreviatura de Manejador de Diálogos. A continuación se proporciona la lista completa.

5.5.3.1 Rutinas Básicas

SUBROUTINE MDI_Inicia (RenIni, RenFin)

```
RenIni      INTEGER * 4, Entrada
            ! Renglón donde inicia la zona
RenFin      INTEGER * 4, Entrada
            ! Renglón donde termina la zona
```

Esta rutina debe llamarse antes que cualquier otra del manejador. Define la zona delimitada por los renglones pasados por parámetros; llena de blancos esa zona y coloca una línea de separación en **RenIni**; por lo que está a disposición del usuario un renglón menos que los estipulados.

SUBROUTINE MDI_Escribe (Texto, Salta, AtribBit, Permanece)

```
Texto       CHARACTER * (*), Entrada
            ! Texto a imprimir
AtribBit    INTEGER * 4, Entrada
            ! Atributo codificado en forma binaria:
            !  Bit  3  2  1  0
            !       :  :  :  :
            !       :  :  :  -- Enfatizado (Bolding)
            !       :  :  -- Inverso (Reverse)
            !       :  -- Intermitente (Blinking)
            !       -- Subrrayado (Underscore)
Permanece   INTEGER *4, Entrada
            ! 0 - no se conserva el atributo, 1 - si se conserva
Salta       INTEGER *4, Entrada
            ! 0 - salta renglón, 1 - no salta
```

Esta rutina escribe un texto en la zona de diálogos. Lleva un control interno de dónde queda el cursor, para la próxima operación de **MDI_Escribe** o **MDI_Lee**. Permite que el cursor salte o no, después de escribir el contenido de Texto. Se puede hacer que se activen una serie de atributos en el momento de impresión y que éstos permanezcan si a continuación se realiza una lectura con **MDI_Escribe**.

SUBROUTINE MDI_Lee (Terminadores, NumTerm, NumCarLeer, Eco, UsaValIni, Texto, LongTex, TermEmp)

```
Texto       CHARACTER * (*), Entrada-Salida
            ! Texto donde se almacenan los caracteres leídos.
Eco         INTEGER * 4, Entrada
            ! 0 - no se escriben los carac., 1 - si se escriben
LongTex     INTEGER * 4, Entrada-Salida
            ! Número de caracteres que contiene Texto
NumCarLeer  INTEGER * 4, Entrada
            ! Número de caracteres a leer:
            !> 0 - Lee tantos caracteres como NumCarLeer indica
            !< 0 - Lee hasta que ocurre un terminador o RETRUN
NumTerm     INTEGER * 4, Entrada
            ! Número de terminadores del arreglo Terminadores
TermEmp     INTEGER * 4, Salida
```

```
                ! Terminador con que se finalizó la lectura:
                ! > 0 - Indice del terminador empleado
                ! = 0 - Return
                ! < 0 - Ningún terminador empleado
  Terminadores (*)  INTEGER * 4, Entrada
                      ! Contiene los ASCII de las teclas que funcionan
                      !como terminadores. Un equivalente ASCII por índice
  UsaValIni     INTEGER * 4, Entrada
                !  0 - Texto se inicializa en blancos y empieza asignar
                !      a partir del primer caracter.
                !  1 - Texto toma el valor que tiene y empieza asignar
                !      caracteres a partir de LongTex
```

Esta rutina lee un texto escrito por el usuario. Tiene ciertas modalidades en la lectura. Permite emplear terminadores, para teclas funcionales que finalizan la lectura; sin necesidad de especificarlo, el terminador RETURN siempre existe. Puede especificarse que lea un determinado número de caracteres o que la lectura se haga hasta que se oprima un terminador o RETURN.

No deja leer más caracteres que los especificados en la declaración de **Texto**. Así, en caso de que **NumCarLeer** sea mayor al número de bytes con que se declara **Texto**, prevalece el de la declaración; tampoco deja leer más de 128 caracteres.

También permite que conforme se teclean los caracteres, se escriban en la pantalla o no. Puede asignársele un valor inicial a **Texto**, y se encargará de imprimir y tomar en cuenta tantos caracteres como **LongTex** indique. Para facilitar la operación de la lectura se proporcionan dos teclas con cualidades específicas: <DELETE> borra el último caracter escrito y <CTRL/U> borra todo lo que se ha escrito.

5.5.3.2 Rutinas De Diversos Propósitos.

MDI_SuenaCamp (Veces)

```
Veces       INTEGER * 4, Entrada
            ! Número de veces a sonar la campana
```

Esta rutina suena la campana (bell) tantas veces como **Veces** lo indique. El cursor no se afecta por usar esta rutina.

SUBROUTINE MDI_ObtenCur (CurCol, CurRen)

```
CurCol      INTEGER * 4, Salida
            ! Columna donde se encuentra el cursor
            !(relativa a la zona de diálogos)
CurRen      INTEGER * 4, Salida
            ! Renglón del cursor (rel. zona diálogos)
```

Esta rutina regresa la posición del cursor en forma relativa a la zona de diálogos.

SUBROUTINE MDI_PonCur (CurCol, CurRen)

```
CurCol      INTEGER * 4, Entrada
            ! Columna a colocar el cursor (rel. zona de diálogos)
CurRen      INTEGER * 4, Entrada
            ! Renglón a colocar el cursor (rel. zona de diálogos)
```

Esta rutina coloca el cursor en la posición indicada por los parámetros, en la zona de diálogos; se emplea para las rutinas **MDI_Lee** o **MDI_Escribe**. Los valores de columna y renglón son relativos a la zona de diálogos.

5.5.4 Ejemplo.

En el programa del listado 5-IX se muestran las cualidades del Manejador de Diálogos. En un principio define los códigos ASCII de los terminadores y pide los parámetros de la rutina MDI_Escribe y MDI_Lee, después se encarga de escribir un letrero que se lee al principio del programa; este letrero opera como un "PROMPT". También realiza lecturas y escrituras consecutivas de un texto. En la figura 5-9 se muestra la pantalla que resultaría de ejecutar este programa.

```
PROGRAM Ejemplo
IMPLICIT NONE

CHARACTER
1     Letrero*30,
1     Texto * 80

INTEGER
1     AtribBit,
1     Condicion,
1     Eco /1/,
1     KB_Init,
1     LongTex,
1     NumTerm,
1     NumCarLeer,
1     Permanece,
1     RenIni,
1     RenFin,
1     Salta,
1     Terminadores (10),
1     TermEmp

! Valores iniciales terminadores
NumTerm = 5
Terminadores (1) = 9  !tab
Terminadores (2) = ICHAR ('?')
Terminadores (3) = ICHAR ('@')
Terminadores (4) = ICHAR ('!')
```

```
Terminadores (5) = ICHAR ('$')

WRITE (6,'($,A)') ' Letrero que se va a escribir en dialogos?'
READ (5,'(A)')Letrero

WRITE (6,'($,A)') ' Numero de Caracteres a Leer ?'
READ (5,*) NumCarLeer

WRITE (6,'($,A)') ' Renglon Inicial, Renglon Final de la Zona ?'
READ (5,*) Renini, RenFin

WRITE (6,'($,A)') ' Valor de Salta (1 o 0) ?'
READ (5,*) Salta

WRITE (6,'($,A)') ' Atributo codificado en Binario ?'
READ (5,*) AtribBit

WRITE (6,'($,A)') ' Valor de Permanece (1 o 0) ?'
READ (5,*) Permanece

CALL MVT_Inicia
Condicion = KB_Init()

! Lee hasta que se oprima CTRL/Y
DO WHILE (.TRUE.)

   CALL MDI_Inicia (Renini,RenFin)
   CALL MDI_SuenaCamp(5)

   !Escribe un el letrero
   CALL MDI_Escribe (Letrero, Salta, AtribBit, Permanece)

   !Lee y escribe y lo que leyo
   CALL MDI_Lee (Terminadores, NumTerm, NumCarLeer,
1                Eco, 0, Texto, LongTex, TermEmp)
   CALL MDI_Escribe (Texto, Salta, AtribBit, Permanece)

ENDDO

END
```

Listado 5-IX

```
Este es un prompt>              Esta es una demostracion del ejemplo del
 *** Manejador de Dialogos ***
```

Figura 5-9

5.6 MANEJADOR DE TECLAS FUNCIONALES. (MTF)

Esta sección se encarga de describir las puntos más importantes del Manejador de Teclas Funcionales; también al final, se proporcionan una lista de las rutinas que conforman a este manejador y dos ejemplos.

5.6.1 Generalidades.

Este manejador se encarga de desplegar teclas funcionales; entendiéndose por teclas funcionales como aquellas que al oprimirlas realizan una función en particular. Así por ejemplo, en un sistema se podría definir que al oprimir "?" se desplegara un texto de ayuda que asistiera al usuario en la operación del sistema. Es importante mencionar que este manejador no se encarga de la detección de la tecla, sino simplemente de desplegar en un renglón, con el atributo en inverso, las teclas funcionales y sus textos asociados, que explican la función de cada tecla. La detección se debe realizar con el Manejador de Diálogos (con la rutina **MDI_Lee**), con el Manejador de Ventanas (con la rutina **MVT_Lee**) o con la rutina KB_READ de detección de teclado.

El Manejador se encarga de que una vez desplegadas las teclas funcionales, el cursor y el atributo activo que existían antes de invocar al Manejador, permanezcan correctamente, al finalizar el desplegado.

5.6.2 Rutinas.

Las rutinas que conforman a este manejador son tres, la primera se encarga de recibir cuál es el renglón que se usará para desplegar las teclas y con ese valor define una zona, para asegurarse que el resto de la pantalla permanece inalterable cuando se despliegan las teclas funcionales. La rutina de desplegado de texto, es la encargada de salvar el cursor y el atributo con **MVT_AlmacenaZona**, despliega las teclas funcionales en inverso y finalmente restablece el cursor y el atributo almacenados con **MVT_ExtraeZona**. La última rutina está orientada a la realimentación y se encarga de parpadear el texto de una tecla funcional en particular.

La forma en cómo operan estas rutinas es muy simple, a los textos asociados a las teclas funcionales se les extrae el número de caracteres que las conforman y se imprimen uno tras otro, en el renglón de la zona de teclas funcionales con el atributo inverso. Para distinguir la tecla funcional del texto asociado, simplemente se coloca en inverso-enfatizado a la tecla funcional y solamente en inverso al texto asociado.

Las rutinas están escritas en FORTRAN-77 empleando los principios de la programación estructurada y les caracteriza el prefijo MDI. Dado que estas rutinas hacen uso de zonas, es indispensable llamar a la rutina **MVT_Inicia**, para que el Manejador de Teclas Funcionales pueda operar correctamente. A continuación se muestra la lista de las rutinas.

```
SUBROUTINE MTF_Inicia ( Ren )

Ren     INTEGER * 4, Entrada
        ! Renglón a imprimir los textos de las teclas funcionales
```

Esta subrutina debe llamarse antes de cualquier otra del manejador. Establece un renglón en la pantalla en el cual se despliegan los textos de las teclas funcionales.

SUBROUTINE MTF_Despliega (TeclasTex, NumTeclas)

TeclasTex	CHARACTER (*) * (*), Entrada ! Texto de las teclas funcionales. El texto para cada !tecla corresponde a un elemento del arreglo.
NumTeclas	INTEGER * 4, Entrada ! Número de teclas funcionales del arreglo **TeclasTex**

Esta subrutina despliega el arreglo de teclas funcionales en el renglón destinado para este propósito. Los textos asociados aparecen en inverso y en inverso-enfatizado la parte correspondiente a la tecla; la tecla y su texto asociado se distinguen por estar delimitados por un espacio en blanco. Es decir, la primera ocurrencia de un blanco marca dónde termina la tecla funcional y dónde inicia su texto asociado. Por ejemplo, en el primer índice podría aparecer "? Ayuda" y en el segundo índice podría aparecer "PF1 Catalogo"

El número de caracteres de cada tecla no debe exceder 40. Si se desea que un índice de una tecla funcional no se contabilice porque no existe texto asociado, solamente es necesario igualarla a espacios en blanco. Esto es útil para usar una tecla que despliega otro renglón de teclas funcionales y así tener más teclas funcionales de las que se pueden desplegar en un solo renglón.

SUBROUTINE MTF_Parpadea (TeclasTex, NumTeclas, TeclaEmp, NumParpadeos)

TeclasTex	CHARACTER (*) *(*) ! Texto de las teclas funcionales
NumParpadeos	INTEGER * 4, Entrada ! Número de veces que va a parpadear
NumTeclas	INTEGER * 4, Entrada ! Número de elementos del arreglo **TeclasTex**
TeclaEmp	INTEGER * 4, Entrada ! Indice del arreglo **TeclasTex** correspondiente !a la tecla que se desea parpadear

Esta subrutina permite que el texto de una tecla funcional parpadee. Tiene la finalidad de proporcionar una realimentación al usuario.

5.6.3 Ejemplo.

En el listado 5-X se muestran dos ejemplos, el primero despliega tres teclas funcionales y parpadea cada una de ellas dos veces. El segundo ejemplo permite desplegar dos renglones de teclas funcionales. Para este efecto, se agrega una tecla funcional extra ("+" en el ejemplo) que despliega el siguiente bloque de teclas y se declara el arreglo de teclas funcionales del doble de elementos (del tripe si fueran tres bloques). En los primeros elementos se introducen las teclas que se despliegan en el primer bloque, el resto de las teclas se declaran en blanco. El segundo bloque se declara a continuación, dejando en blanco los textos de las teclas del primer bloque. Al final de ambos bloques siempre aparece la tecla extra de desplegado de bloques ("+" en el ejemplo).

Notar que sólo se repiten los textos, los equivalentes ASCII de las teclas funcionales no (**ASCIITeclas** tienen 7 elementos). Con la rutina KB se realiza la detección del teclado; cada vez que el usuario oprime una tecla funcional, la parpadea dos veces y cuando oprime "+", se despliega el siguiente bloque de teclas. Es importante mencionar que en FORTRAN 77 pasar como parámetro **TeclasTex** (**BloqueTecla**), no se pasa un elemento, sino el elemento inicial a partir del cual se considera todo el arreglo. En caso de que esto no existiera en otra implementación de FORTRAN, habría que llevar un arreglo (de 7 elementos en el ejemplo) para cada bloque de teclas funcionales. En las figuras 5-10 y 5-11 se muestran las pantallas correspondientes a cada uno de los dos programas.

```
PROGRAM Ejemplo
IMPLICIT NONE

CHARACTER
1    TeclasTex (3) * 40 /'? Ayuda',
1                       '<Esc> Salida',
1                       '<TAB> Desplegado del Estado'/

INTEGER
1    Vez

CALL HVT_Inicia
CALL MTF_Inicia (12)

CALL MTF_Despliega (TeclasTex, 3)

DO Vez = 1, 3
    CALL MTF_Parpadea (TeclasTex, 3, Vez, 3)
ENDDO

END


PROGRAM Ejemplo2
IMPLICIT NONE

CHARACTER
1    TeclasTex (14) * 40 /
1                        !
1                        !1er bloque
1                        '? Ayuda',
1                        '<TAB> Directorio',
1                        '<LineFeed> Cencelacion',
1                        ' ',  ! @ Refrescar Pantalla
1                        ' ',  ! # Altas
1                        ' ',  ! <ESC> Salir
1                        '+ Siguientes Teclas',
1
1                        ! 2do bloque
1                        ' ',  ! ? Ayuda
1                        ' ',  ! <TAB>
1                        ' ',  ! <LineFeed>
```

```
     1                        '@ Refrescar Pantalla',
     1                        '# Altas',
     1                        '<ESC> Salir',
     1                        '+ Siguientes Teclas'/

     INTEGER
     1   ASCII,
     1   ASCIITeclas (7) /
     1                       63,    !?
     1                       9,     !<TAB>
     1                       10,    !<LineFeed>
     1                       64,    !@
     1                       35,    !#
     1                       27,    !<ESC>
     1                       43/,   !+
     1
     1   BloqueTeclas,
     1   Condicion,
     1   Indice,
     1   IndiceTecla,
     1   KB_Init,
     1   KB_Read,
     1   NumTeclas /7/

     LOGICAL * 1
     1   Encontro

     Condicion = KB_Init ()
     CALL MVT_inicia ()
     CALL MTF_inicia (12)

     BloqueTeclas = 1     ! 1er bloque
     CALL MTF_Despliega (TeclasTex (BloqueTeclas), NumTeclas)

     DO WHILE (.TRUE.)

         ASCII = KB_Read ()

         !Busca la tecla funcional
         Encontro = .FALSE.

         DO Indice = 1, NumTeclas
            IF (ASCII .EQ. ASCIITeclas (Indice)) THEN
                Encontro = .TRUE.
                IndiceTecla = Indice
            !NDELSE
            ENDIF
         ENDDO

         IF (Encontro) THEN

            CALL MTF_Parpadea (TeclasTex (BloqueTeclas),
     1          NumTeclas, IndiceTecla, 2)
```

```
        IF (CHAR(ASCII) .EQ. '+') THEN

            IF (BloqueTeclas .EQ. 1) THEN
                BloqueTeclas = 8
            ELSE  !Segundo bloque BloqueTeclas = 8
                BloqueTeclas = 1
            ENDIF

            CALL MTF_Despliega (TeclasTex (BloqueTeclas), NumTeclas)

        !NOELSE
        ENDIF

    !NOELSE
    ENDIF

ENDDO
END
```

Listado 5-X

? Ayuda <Esc> Salida <TAB> Desplegado del Estado

Figura 5-10

? Ayuda <TAB> Directorio <LineFeed> Cancelacion + Siguientes Teclas

@ Refrescar Pantalla # Altas <ESC> Salir + Siguientes Teclas

Figura 5-11

5.7 MANEJADOR DE ESTRUCTURAS (MES)

5.7.1 Descripción

Las rutinas pertenecientes a este manejador tienen la finalidad de permitir un fácil manejo de las estructuras de datos donde se almacena la información que introduce el usuario al sistema SER a través del Sistema Interactivo.

A continuación se describirán las estructuras de datos utilizadas en la interfaz interactiva y el uso que se les da. El tipo de información que se maneja es: Tablas de datos, directorio de tablas, datos referentes a las gráficas, directori de gráficas y bitácoras de sesiones de trabajo.

5.7.1.1 Tablas De Datos.

Las tablas de datos que contienen la información que se utiliza para dibujar las gráficas se almacenan en una serie de listas ligadas (simuladas por dos arreglos: arreglo de información y arreglo de apuntadores). La información referente a cada tabla se controla por medio de un catálogo de tablas (diversos arreglos).

La figura 5-12 muestra la forma de construir la lista ligada que almacena las tablas de datos.

Estructura de datos para el almacenamiento
de las tablas de datos.

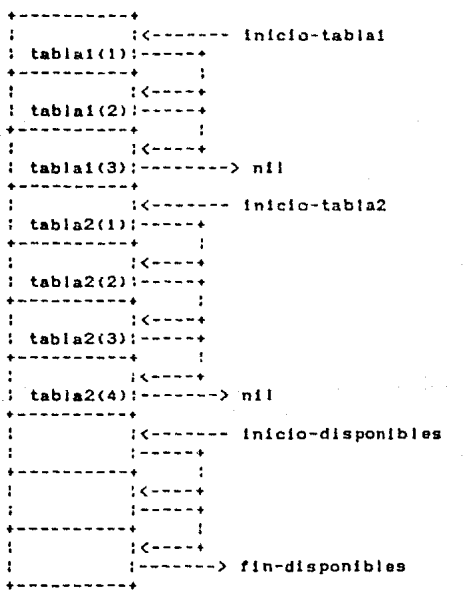


Figura 5-12

La razón por la que se utilizaron listas ligadas fué para simular estructuras dinámicas (no las proporciona el lenguage FORTRAN). El tamaño de las tablas de datos es variable, por lo tanto se requiere de una estructura como la anterior. Debido a que se manejan dos tipos de tablas de datos (numéricas y alfanuméricas), se utilizan también dos vectores comunes (dos listas ligadas); uno numérico y otro alfanumérico.

En cada una de las celdas del vector común se almacena un elemento de alguna tabla, al cual se le asocia un apuntador al siguiente elemento de la tabla. Debido a que el Manejador de Estructuras permite dar de alta nuevas tablas y cancelar alguna de las existentes, el vector común probablemente se fragmente. Sin embargo con el uso de listas ligadas este fenómeno no representa ningún problema.

5.7.1.1.1 Editor De Tablas.

Para crear tablas de datos y modificarlas se creó un editor enfocado al manejo de tablas. A través de dicho editor se pueden introducir nuevas celdas a las tablas (numéricas o alfanuméricas), eliminar celdas o modificar su contenido. El editor hace uso del manejador de ventanas. Al invocarlo se abre una ventana sobre la que se despliega la tabla y se permiten realizar movimientos de avance y retroceso sobre las celdas.

Otras características importantes son: El editor puede suspenderse momentaneamente para realizar otra actividad y restaurarlo en el estado en el que se encontraba. El editor lleva a cabo una validación de datos de captura en linea.

5.7.1.1.2 Estructuras De Datos.

Las estructuras de datos empleadas para almacenar la información referente a las tablas de datos son:

CMES_CatalogoVec

```
Apinic (50)     !INTEGER*2
                !Apuntadores al primer elemento de la tabla
                !almacenada en el vector común.
IdentVec (50)   CHARACTER*10
                !Nombres de las tablas.
TamVec (50)     !INTEGER*2
                !Número de elementos de cada tabla.
TipoVec (50)    CHARACTER*1
                !Tipo de tabla "N" Numérica. "A" Alfanumérica.
NumVecs         !INTEGER*4
                !Número de tablas dadas de alta en el catálogo.
```

Al analizar las estructuras de datos anteriores se puede apreciar que:

1) El número máximo de tablas que se pueden almacenar es de 50.
2) A cada tabla se la reconoce con el índice de la posición que ocupan en los arreglos.
3) Se manejan cuatro arreglos en paralelo (nombre, tamaño, tipo y apuntador a la celda inicial). La información referente a cada tabla se almacena en el mismo índice de cada arreglo.
4) Se maneja un catálogo común tanto para tablas numéricas como para alfanuméricas.

CMES_VectorNumericoComun

```
ApVecNum (5000)     !INTEGER*2
                    !Vector donde se almacenan los apuntadores que
                    !forman la lista ligada que almacena las tablas
```

```
                        !numéricas.
DispNumFinal            INTEGER*2
                        !Apuntador a la última celda libre
                        !del arreglo VecNum.
DispNumInic             INTEGER*2
                        !Apuntador a la primera celda libre
                        !del arreglo VecNum.
VecNum (5000)           REAL*4
                        !Vector común donde se almacenan las
                        !tablas numéricas.
```

Se puede apreciar, a partir del análisis de las estructuras anteriores, que el número de celdas comunes disponibles es de 5000. Por otra parte, el tamaño máximo de una tabla numérica es de 500 celdas. Esto significa que el caso extremo se pueden almacenar diez tablas numéricas de 500 elementos cada una.

CMES_VectorAlfanumericoComun

```
ApVecAlfa (100)         INTEGER*2
                        !Vector donde se almacenan los apuntadores que
                        !forman la lista ligada que almacena las tablas
                        !alfanuméricas.
DispAlfaFinal           INTEGER*2
                        !Apuntador a la última celda libre del arreglo
                        !VecAlfa.
DispAlfaInic            INTEGER*2
                        !Apuntador a la primera celda libre
                        !del arreglo VecAlfa.
VecAlfa (100)           CHARACTER*25
                        !Vector común donde se almacenan
                        !las tablas alfanuméricas.
```

En lo que respecta a las tablas alfanuméricas, se pueden almacenar únicamente 100 celdas de tamaño 25 de este tipo. Este número es pequeño debido a que en las tablas alfanuméricas únicamente se almacenan rótulos y éstos por lo general no son numerosos.

Además de reconocer a cada tabla por medio de su identificador se le reconoce por la posición que ocupa dentro del catálogo de tablas. Como máximo se pueden tener 50 tablas dentro del catálogo.

5.7.1.2 Gráficas

Otro grupo de estructuras de datos de gran importancia es el que se refiere al almacenamiento de la información respecto a las gráficas.

Los datos capturados, a través del manejador de captura, referentes a cada tipo de gráfica, se almacenan en una cadena de caracteres de tamaño máximo 650. El tamaño de esta cadena es variable ya que para cada tipo de gráfica la cantidad de información requerida es diferente. Por este motivo y con el fin de ahorrar espacio en memoria se decidió segmentar la cadena de caracteres de la gráfica en páginas y después almacenar estas páginas en un vector común. Las páginas pertenecientes a una gráfica se encuentran formando una lista ligada de la misma manera que se almacena la información para las tablas de datos. Ver figura 5-13.

Estructura de datos para el almacenamiento de las gráficas.

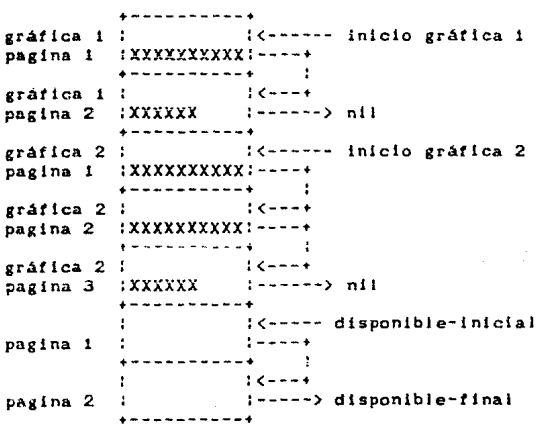


Figura 5-13

5.7.1.2.1 Estructuras De Datos

Las estructuras de datos empleadas para almacenar la información referente a las gráficas son:

CHES_CatalogoGraf

```
IdentGraf(10)       CHARACTER*10
                    !Nombres de las gráficas.
TipoGraf(10)        INTEGER*2
                    !Tipo de la gráfica. Existe un catálogo
                    !donde se asigna un número secuencial a
                    !cada tipo de gráfica (1-14). Aquí se almacena
                    !dicho código.
ApInicGraf(10)      INTEGER*2
                    !Apuntadores a la página inicial, dentro del
                    !vector común, de cada una de las gráficas.
NumGrafs            INTEGER*4
                    !Número de gráficas dadas de alta.
EspacioCatGraf      INTEGER*4
                    !Número de páginas libres en el vector
                    !común.
```

- Comentaremos, al respecto de las estructuras anteriores, que el número máximo de gráficas que pueden almacenarse en el catálogo de gráficas es de diez.

- El tamaño de las páginas del vector común es de diez caracteres.

- El número que se asigna a cada tipo de gráfica es el siguiente:
  1) Barras
  2) Linea
  3) X vs Y
  4) Circular
  5) Histograma
  6) Polígono
  7) Apiladas
  8) Error
  9) Barras-Linea
  10) Barras-X vs Y
  11) Linea-X vs Y
  12) Histograma-Polígono
  13) Histograma-X vs Y
  14) Polígono-X vs Y

CMES_DatosGraf

```
DatosGraf(250)      CHARACTER*10
                    !Vector común donde se almacenan por
                    !páginas cada uno de los datos de cada
                    !gráfica. El tamaño de las páginas es
                    !de 10 bytes.
ApDatosGraf(250)    INTEGER*2
                    !Vector de apuntadores con el cual se construyen
                    !las listas ligadas que forman la información
                    !de cada gráfica.
```

```
DispDatGrafInic     INTEGER*4
                    !Apuntador a la primera página libre
                    !del vector común.
DispDatGrafFin      INTEGER*4
                    !Apuntador a la última página libre
                    !del vector común.
```

CMES_TamGraf

```
Tamgraf(14)         INTEGER*4
                    !Número de páginas que ocupa cada
                    !uno de los 14 tipos de gráficas.
```

En lo que concierne a las estructuras anteriores, se puede decir que el número máximo de páginas disponibles es de 250. Esto representa 2500 bytes y es suficiente para almacenar, en el caso extremo, cinco gráficas del tipo X vs Y (dichas gráficas son las que requieren de mayor espacio).

5.7.1.3 Bitácoras De Sesión.

El sistema SER proporciona a sus usuarios interactivos una manera de respaldar la información introducida durante una sesión de trabajo. Si el usuario así lo desea, puede salvar en disco la información existente hasta el momento en las estructuras de datos de la interfaz y posteriormente utilizarla en otra sesión.

5.7.2 Rutinas

A continuación se presentará una lista de las rutinas pertenecientes al manejador de estructuras y junto con ellas una descripción de sus parámetros y su funcionamiento. Al final de todas las rutinas se proporciona un diccionario de datos que explica el significado de cada parmetro utilizado.

5.7.2.1 Referentes Al Editor De Tablas.

SUBROUTINE MES_InicioEditorVectores

Esta rutina inicializa variables comunes al editor de tablas (vectores) con la finalidad de dejarlo listo para funcionar. Debe ser invocada antes de editar cualquier tabla.

```
SUBROUTINE MES_EditaVector (Tabla, TipoTabla, TeclaFuncSal,
                            TeclasFuncionales, NumTeclasFunc)
```

Esta rutina realiza las siguientes acciones: Despliega en una ventana de la pantalla el contenido de la "tabla", permite editar la tabla mostrada en la ventana (borrar celdas, insertar nuevas celdas, modificar el contenido de una celda y permitir el movimiento por toda la tabla) y en el momento de detectar alguna de las teclas funcionales se abandona el editor.

El uso de las teclas funcionales permite manejar en paralelo con el editor de tablas otras acciones, como lo son: Despliegue de ayudas y despliegue del directorio de tablas. Esto es, se interrumpe momentaneamente la edición de tablas y posteriormente se continua en el estado en el que se encontraba antes de abandonar el editor.

```
INTEGER*4 FUNCTION MES_ManejadorVectores (NomTabla, NumTabla,
                                          TipoTabla, ExisteTabla)
```

Es una rutina que controla al editor de tablas. Abre la ventana donde se despliega la tabla, invoca a las acciones paralelas que se llevan a cabo al salir del editor por la presion de alguna tecla funcional (despliegue de ayudas y despliegue del directorio de tablas), al salir definitivamente del editor la tabla se da de alta en el catálogo de tablas. Esta rutina regresa dos estados: 0 -> se ejecutó con éxito, 1 -> se abortó el proceso.

5.7.2.2 Referentes Al Directorio De Tablas.

```
INTEGER*4 FUNCTION MES_ExisteVector (NomTabla, NumTabla, TipoTabla)
```

Esta función indica si una tabla, cuyo nombre es "NomTabla", se encuentra dada de alta en el directorio de tablas. La función regresa cero (0) cuando la tabla ya existe y uno (1) en caso contrario.

```
SUBROUTINE MES_AltaVecNum (Tabla, NumTabla, ExisteTabla, NomTabla)
```

Introduce la información perteneciente a una tabla de datos numéricos al directorio de tablas numéricas. Si esta tabla ya había sido dada de alta con anterioridad, únicamente se actualiza la información (celdas de la tabla) y en caso contrario se dan de alta todos sus datos.

```
SUBROUTINE MES_AltaVecAlfa (Tabla, NumTabla, ExisteTabla, NomTabla)
```

Esta rutina ejecuta el mismo proceso que MES_AltaVecNum con la diferencia que ésta funciona únicamente para tablas alfanuméricas.

```
LOGICAL*1 FUNCTION MES_HayVectores
```

La función es verdadera si existe por lo menos una tabla en el catálogo de tablas, en caso contrario, la función es falsa.

```
SUBROUTINE MES_BajaVector(NumTabla)
```

Elimina del directorio de tablas a aquella tabla que se encuentra en la posición "NumTabla" del directorio. La tabla ya debe existir.

```
INTEGER*4 FUNCTION MES_LeeArchVec (NomArch, NomTabla, NumTabla,
                                   TipoTabla, ExisteTabla)
```

Esta rutina se encarga de llenar una tabla de datos con la información contenida en un archivo en disco.

```
SUBROUTINE MES_CatVec
```

Abre una ventana en la pantalla y despliega el contenido del directorio de tablas. Para cada tabla existente se muestra el nombre, el tipo y el tamaño de la misma. La figura 5-14 muestra la forma en que se despliega el directorio de tablas.

```
... Alta de una Grafica | X vs Y | Forma 1 | Directorio de Tablas |
  Numero de funciones (1,2,3,4,5) :  [3]
  Numero
```

TABLAS			
NUM	NOMBRE	TAMANO	TIPO
1	DISTNX	500	N
2	DISTNY	500	N
3	DIENTSX	500	N
4	DIENTSY	500	N
5	DIENTSRX	50	N
		...cont	

```
                                                    Tipo de Union
        Nombre de la Funcion                        de los Puntos
[Distribucion Normal                    ]           [1]
[Diente de Sierra                       ]           [3]
[Diente de Sierra Rect.                 ]           [2]

                                        (1) lineas
                                        (2) marcadores
                                        (3) marc. - lineas
                                        (4) marc conec a y = 0

          Titulos Generales de las Funciones
     [Funciones de Biblioteca                   ]
     [del Sistema Interactivo definidas         ]
     [para alumnos.                             ]

-------------------------------------------------------------------------

Msj> Seleccione una opcion
Msj> Comenzar a capturar

 <RET> Continuar <ESC> Terminar
```

Figura 5-14

5.7.2.3 Referentes Al Directorio De Gráficas

```
INTEGER*4 FUNCTION MES_ExisteGrafica (NomGraf, NumGraf, TipoGraf)
```

Esta función indica si una gráfica, cuyo nombre es "NomGraf", se encuentra dada de alta en el directorio de tablas. La función regresa cero (0) cuando la tabla ya existe y uno (1) en caso contrario. La función también regresa la posición de la gráfica en el directorio y su tipo y en caso de no existir regresa la posición que ocuparía si se diera de alta.

```
SUBROUTINE MES_AltaGraf (DatosGraf, NumGraf, ExisteGraf,
                         NomGraf, TipoGraf)
```

Introduce la información perteneciente a una gráfica al directorio de gráficas. Si esta gráfica ya había sido dada de alta con anterioridad, únicamente se actualiza la información (celdas de la gráfica) y en caso contrario se dan de alta todos sus datos.

```
SUBROUTINE MES_BajaGrafica(NumGraf)
```

Elimina del directorio de gráficas a aquella gráfica que se encuentra en la posición "NumGraf" del directorio. La gráfica ya debe existir.

SUBROUTINE MESCataGraf

Abre una ventana en la pantalla y despliega el contenido del directorio de gráficas. Para cada gráfica existente se muestra el nombre y el tipo de la misma. La figura 5-15 muestra la forma en que se despliega el directorio de gráficas.

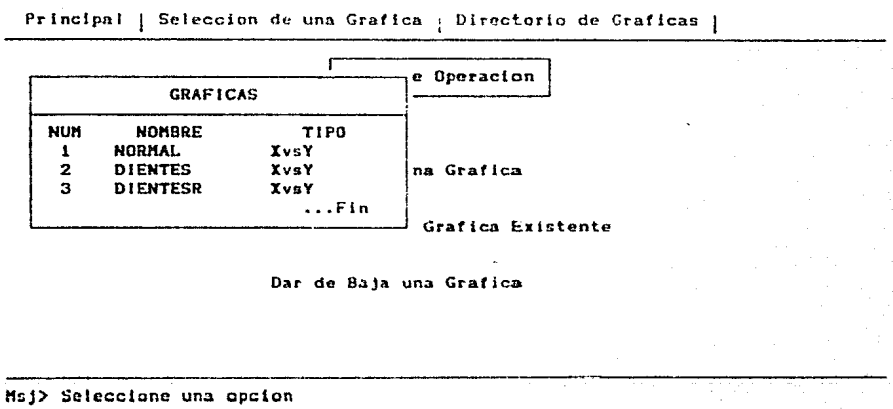


Figura 5-15

LOGICAL*1 FUNCTION MES_HayGraficas

La función es verdadera si existe por lo menos una gráfica en el catálogo de gráficas, en caso contrario, la función es falsa.

5.7.2.4 Referentes A Las Bitácoras De Trabajo.

SUBROUTINE MES_EscribeBitacora (NomBitac)

La rutina anterior se encarga de vaciar a un archivo en disco toda la información contenida en las estructuras de datos del Sistema Interactivo, con la finalidad de poder recuperarla en sesiones posteriores. Esta facilidad permite que el usuario pueda interrumpir su sesión de trabajo en el momento que el lo requiera y poder continuar con la misma en otra ocasión.

SUBROUTINE MES_LeeBitacora (NomBitac)

La rutina anterior se encarga de vaciar a las estructuras de datos del Sistema Interactivo la información contenida en una bitácora de trabajo almacenada en un archivo en disco. Esta facilidad permite que el usuario pueda interrumpir su sesión de trabajo en el momento que el lo requiera y poder continuar con la misma en otra ocasión.

SUBROUTINE MES_CataBit

La rutina anterior muestra en una ventana abierta en el video, los archivos en disco que contienen alguna bitácora de trabajo. Esta rutina hace uso del manejador de ventanas. La figura 5-16 muestra la forma en que se despliega el directorio de bitácoras.

5.7.2.5 Diccionario De Datos

A continuación se presenta una lista de todos los datos usados como parámetros en las rutinas que se explicaron anteriormente y que pertenecen al manejador de estructuras (MES).

```
ExisteTabla     !INTEGER*4
                !Bandera que indica si la tabla ya existe
                !o no en el directorio de tablas:
                !1 -> No existe   0 -> Existe
ExisteGraf      !INTEGER*4
                !Bandera que indica si la gráfica ya
                !existe o no en el directorio de gráficas:
                !1 -> No existe   0 -> Existe
NomArch         CHARACTER *(*)
                !Nombre del archivo que contiene una tabla
```

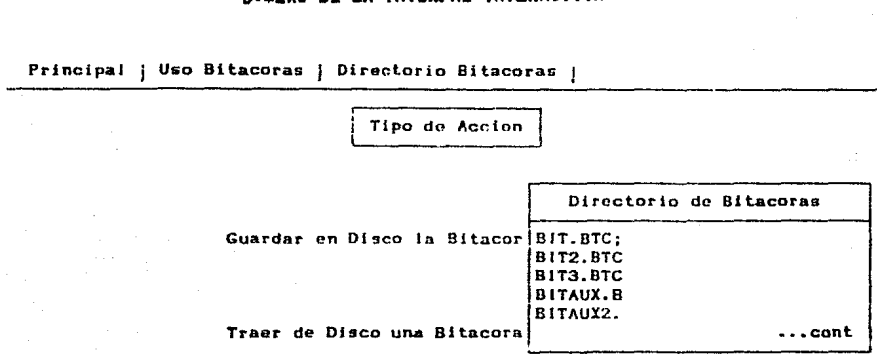


Figura 5-18

```
                        !de datos.
      NomBitac          CHARACTER *(*)
                        !Nombre valido para un archivo en VMS.
                        !Es el nombre de una bitácora de trabajo.
      NomGraf           CHARACTER *(*)
                        !Nombre con el cual se reconoce una gráfica.
      NomTabla          CHARACTER *(*)
                        !Nombre con el cual se reconoce una tabla
                        !de datos.
      NumGraf           INTEGER*4
                        !Posición que ocupa la grafica dentro del
                        !directorio de gráficas. La posición
                        !también identifica a la gráfica.
      NumTabla          INTEGER*4
                        !Posición dentro del directorio de tablas
                        !de la tabla a la cual se hace referencia.
      NumTeclasFunc     INTEGER*4
                        !NÚmero de teclas funcionales que
                        !se encuentran en el arreglo de
                        !teclas funcionales.
      TeclaFuncSal      INTEGER*4
                        !Código de la tecla con la que se
                        !abandonó el editor de tablas.
                        !La salida del editor es temporal.
      TeclasFuncionales(*)  INTEGER*4
                        !Contiene los códigos de las teclas
                        !que permiten la salida del editor.
```

```
TipoTabla       CHARACTER*1
                !Tipo de la tabla :
                !'N' tabla numérica,
                !'A' tabla alfanumérica.
TipoGraf        INTEGER*4
                !Código que indica a cual de los
                !14 tipos de gráficas pertenece la
                !gráfica en cuestión.
Tabla(*)        CHARACTER *(*)
                !Es la tabla de datos que se va a dar
                !de alta, actualizar o editar.
DatosGraf       CHARACTER *(*)
                !Cadena de caracteres que almacena los
                !datos de una gráfica.
```

5.8 MANEJADOR DE ANIDAMIENTOS. (MAN)

Es en este punto donde se describirán los aspectos más importantes relativos al Manejador de Anidamientos. Al final se agrega una lista de rutinas describiendo su función y los parámetros involucrados.

5.8.1 Generalidades.

El Manejador de Anidamientos es el responsable de desplegar el estado del sistema. Usualmente en los programas se reporta al usuario qué parte del sistema se está ejecutando, para que así, esté informado en todo momento de lo que está sucediendo; esto integra lo que se conoce como realimentación.

Los estados se despliegan en un renglón con el atributo en inverso. Cada elemento se separa del otro por medio de una barra. Es posible que todos los estados no quepan en un solo renglón; cuando esto sucede, el sistema despliega todos los estados que quepan procurando que queden los más recientes y coloca tres puntos suspensivos al principio del renglón.

El manejador se encarga de guardar la posición del cursor y el atributo para que una vez que se despliegue el estado, puedan restablecerse automáticamente. Esto asegura que el despiegado del estado es prácticamente inofensivo para cualquier operación de escritura o lectura que se esté realizando. Inclusive guarda las dimensiones de la Frontera de la pantalla, por si acaso se tienen una ventana abierta o una zona activa, y así restablecerlas al terminar el desplegado del estado.

5.8.2 Rutinas.

Son cuatro las rutinas que componen a este manejador. La primera de ellas se encarga de definir la zona destinada al renglón de anidamientos. Las otras dos fundamentalmente llevan una "Pila" para ir almacenando cada uno de los estados. Se eligió esta estructura ya que el último estado introducido debe ser el primero en eliminarse. La cuarta rutina despliega con el atributo inverso los estados, almacenados en la estructura de datos, uno tras otro. Para que el

despliegado no afecte a otras zonas o a ventanas abiertas, esta rutina llama a **MVT_AlmacenaZona** y al terminar el despliegado, invoca a **MVT_ExtraeZona**; por este motivo para emplear al Manejador de Anidamientos es necesario invocar al **MVT_Inicia**. Las rutinas se elaboraron en FORTRAN-77, de acuerdo a los principios de la programación estructurada y les antecede el genérico MAN. A continuación se muestra la lista completa.

```
SUBROUTINE MAN_Inicia ( Renglon )

Renglon     INTEGER *4, Entrada
            ! Renglón destinado a la zona de anidamientos
```

Esta rutina debe llamarse antes de cualquier otra del manejador. Establece un renglón de la pantalla en el cual la zona de anidamientos se desplegará.

```
SUBROUTINE MAN_Inserta ( Texto )

Texto       CHARACTER * (*)
            ! Texto del elemento a incluir en la pila
```

Esta rutina agrega a la lista de elementos (pila), el contenido de **Texto**. Esta rutina se emplea en combinación de **MAN_Despliega** y **MAN_Elimina**. El número maximo de caracteres que puede contener Texto es 40.

```
SUBROUTINE MAN_Elimina ()
```

Esta rutina extrae de la pila el elemento del tope.

```
SUBROUTINE MAN_Despliega ()
```

Esta rutina despliega todos los elementos de la pila. Se despliega a partir del elemento que está en el fondo de la pila, hasta el que está en el tope. El desplegado se hace con el atributo de la pantalla inverso. Si hay mas elementos que los que se pueden desplegar, entonces coloca tres puntos suspensivos al inicio y despliega lo elementos mas recientes.

Cuando el sistema es modular y cada una de las funciones importantes del programa están colocadas en una rutina, el control y despiegado del estado es muy sencillo, basta con colocar una llamada a **MAN_Inserta** y **MAN_Despliega** al inicio de cada rutina; generalmente el mensaje que se anida es representativo del nombre de la rutina. Al terminar la rutina, antes del RETURN, simplemente se coloca un **MAN_Elimina** y **MAN_Despliega**, para indicar que la función ha terminado.

5.8.3 Ejemplo.

En el listado 5-XI aparece un programa que hace uso, en forma sencilla, del Manejador de Anidamientos; define el renglón 12 como la zona de anidamientos y posteriormente pide 8 textos. Cada texto lo va anidando en el estado y desplegando en la pantalla. Por último extrae, uno por uno, los textos del renglón de anidamientos. En la figura 5-17 se muestra una "copia en papel" de la pantalla que resultaría al ejecutar este programa.

```
PROGRAM Ejemplo
IMPLICIT NONE

CHARACTER
1   TEXTO * 30


INTEGER
1   Vez


CALL MVT_Inicia
CALL MAN_Inicia (12)

DO Vez = 1, 5

   WRITE (6,'($,A, I1, A)') ' Escribe un texto', Vez,' ?'
   READ (5, '(A)') Texto
   CALL MAN_Inserta (Texto)
   CALL MAN_Despliega ( )

ENDDO

DO Vez = 1, 8

   CALL MAN_Elimina ( )
   CALL MAN_Despliega ( )

ENDDO

END
```

Listado 5-XI

```
Escribe un texto ?Notar los puntos suspensivos




... tercer | cuarto | quinto | sexto | Notar los puntos suspensivos |
```

Figura 5-17

5.9 MANEJADOR DE CAPTURA (MCA)

La captura de información es una de las partes fundamentales y de mayor importancia en gran cantidad de sistemas interactivos de los cuales el Sistema SER no es la excepción, usando aproximadamente 40 pantallas de captura distintas.

Por esta razón, fué necesario desarrollar un manejador general que realizara el control de la captura de información y evitara que existiera un código particular para cada una de las pantallas. Además, debía ser compatible con los otros manejadores desarrollados y cumplir con las características generales que permitan capturar los datos necesarios para el sistema, y en medida de lo posible los de otras aplicaciones que lo utilicen.

El diseño de este manejador, al igual que los otros del sistema, fué basándose en que el usuario necesitara llamar el mínimo de rutinas, así como definir el menor número de parámetros y características de los campos cuando no tuviera requerimientos muy específicos, ya que el sistema define las condiciones iniciales con que se realiza la captura. Sin embargo, existen una serie de características que se pueden asociar a cada campo en forma particular.

Las características que el manejador contempla fueron definidas de acuerdo a las necesidades que presenta el Sistema SER interactivo, pero debido a que éstas son muy similares a las de la mayoría de las interfacez interactivas, el manejador puede ser aprovechado por una gran cantidad de aplicaciones.

A continuación, se explican las características del manejador, las rutinas y definiciones necesarias para su uso y por último, la descripción técnica básica que permite conocer su estructura interna.

5.9.1 Generalidades

El manejador de captura es un conjunto de rutinas que permiten un fácil desarrollo de interfaces para captura de información por medio de un programa de aplicación.

Las rutinas de control que proporciona el manejador, permiten desplegar una forma de captura en la pantalla a partir de la definición de ella en un archivo de texto. Asimismo controlan el movimiento entre campos, la lectura y validación de ellos de acuerdo al tipo, longitud y características que se le hayan asociado a cada uno de manera particular. A continuación se explican las características generales que tiene el manejador.

5.9.1.1 Presentación En Pantalla

La figura 5-18 muestra la presentación de una forma de captura en la pantalla. Obsérvese que todos los campos se delimitan por parentesis cuadrados con la finalidad de indicar su longitud y que el campo que se esté capturando se muestra en inverso.

```
pruebas de la forma de captura
 Numero de funciones (1,2,3,4,5) :  [5]
 Numero
   de                                                          Tipo de Union
 Funcion     Tabla X       Tabla Y       Nombre de la Funcion    de los Puntos
   1      [1122233344] [4           ] [555666777888999000aaaaabb]      [b]
   2      [b111222333] [4445556667] [77        888999000aaaaabb]      [b]
   3      [b111 222 3] [33 444  55] [566677788899900aaaaabbbb]      [ ]
   4      [           ] [          ] [                        ]      [ ]
   5      [           ] [          ] [                        ]      [ ]
                                                    (1) lineas
                                                    (2) marcadores
                                                    (3) marc. - lineas
                                                    (4) marc conec a y = 0
                    Titulos Generales de las Funciones
                [                                   ]
                [                                   ]
                [                                   ]
```

```
Msj> Leyendo forma de captura
Msj> Comenzar a capturar
```

Figura 5-18

5.9.1.2 Movimiento Entre Campos

Para cambiar de un grupo a otro, se tienen definidas dos teclas: la tecla <RETURN> avanza al siguiente campo y la tecla <-- retrocede uno.

Los movimientos se harán solamente si el valor capturado en ese campo cumple las validaciones que tiene asociado. Si no se cumplen éstas, el manejador indicara el error en que se ha incurrido y escribirá el valor que el campo tenía anteriormente colocando el cursor al principio del campo, con la finalidad de que se teclee el valor en forma correcta.

Dentro de un campo ya capturado y validado, no se permite edición, por lo que para modificarlo será necesario tecleario nuevamente. Sin embargo, cuando un campo se esté capturando existe la posibilidad de usar la tecla <DELETE> para borrar el caracter a la izquierda del cursor.

5.9.1.3 Tipos De Campos

Los campos siempre tendrán asociado un tipo de dato. El manejador validará en linea que la información que se esté capturando cumpla con ese tipo, es decir, si se teclea un caracter no válido el manejadar no lo aceptará.

Los tipos permitidos són:

1) **ENTEROS.**- Sólo se aceptarán los dígitos del 0 al 9 y los signos + y -. Por ejemplo: 234, +73821, -312.

2) **REAL.**- Existen dos modalidades de captura: en forma fraccionaria normal, en donde se permiten los dígitos del 0 al 9, un punto decimal y los signos + y -, por ejemplo: 3.1415, -9373.2; el otro formato es la notación científica en donde son permitidos los dígitos, un punto decimal, los signos + y - y la letra E acompañada de un número entero, por ejemplo: 726.73E-2, -346.5E+7. El valor entero indica la potencia de 10 por la que será multiplicado el número real.

3) **ALFANUMERICO.**- Se aceptarán los dígitos del 0 al 9 y las letras de la A a la Z, ya sean mayúsculas o minúsculas.

4) **CARACTER.**- Se capturará cualquier caracter valido en el codigo ASCII.

5) **SI y NO.**- En este tipo de campos se permitirán solamente las palabras SI y NO en forma mayúscula o minúscula. Si se le define con longitud de 1, aceptará S o N.

6) **X.**- Aceptará sólo los caracteres X y x. Se usa principalmente cuando se desea que se marque o no una posible respuesta.

5.9.1.4 Validaciones

A cada campo se le podrá asociar una o varias validaciones que el manejador debe realizar cada vez que se capture un valor en ese campo. Las validaciones disponibles son:

1) **CAPTURA OBLIGATORIA (REQUERIDOS)**

Por medio de esta validación se indica que un campo deberá forzosamente contener un valor. En el caso que un campo con esta característica no tenga algún valor no se permitirá movimiento al campo siguiente o al anterior. Asimismo, al terminar de capturar, si algún campo requerido no tiene ningún valor, el cursor será colocado en dicho campo sin permitir salir de la forma.

2) **LIMITES INICIAL Y FINAL**

Para que el valor de un campo sólo pueda estar entre un rango de valores, se utiliza esta opción en donde se define un límite inferior y un límite superior que definen dicho rango.

3) **CAMPOS CONDICIONADOS A OTRO (OCULTOS)**

Con la finalidad de que los campos que sólo se requieran capturar bajo ciertas condiciones no se despliegen en la pantalla, existe esta validación la cual permite que sólo cuando un campo tenga un valor específico se desplieguen y se permitan capturar otros campos.

La figura 5-19 muestra una forma de captura en donde existen campos condicionados al campo **Otro eje vertical de referencia? (si/no)**, los cuales serán desplegados y capturados cuando dicho campo valga **SI** como puede observarse en la figura 5-20.

```
pruebas de la forma de captura
Nombre del eje X :   [Frecuencia                              ]

Nombre del eje Y :   [f(x)                                    ]
    Escala eje Y :     (1) Automatica  (2) Manual          [1]

    Acotamiento :      (1) No Logaritmico (2) Logaritmico  [1]

Otro eje vertical de referencia?  (si/no) [NO]
```

```
Msj> Leyendo forma de captura
Msj> Comenzar a capturar
```

Figura 5-19

4) MUTUAMENTE EXCLUYENTES

Con esta opción se indica qué campos son mutuamente excluyentes con lo que cuando uno de ellos tenga un valor asociado, los demás no podrán tener ningún valor; por lo que el manejador cancelará los valores capturados de esos campos. Esta característica es muy útil cuando sólo se puede seleccionar una respuesta de varias posibles.

5) RANGOS VARIABLES

Frecuentemente el rango en que puede estar el valor de un campo no es fijo, sino que depende del valor de otro campo. Por ejemplo cuando se captura el valor inicial y el valor final de algun rango, el valor final deberá ser al menos mayor o igual al valor inicial previamente capturado el cual será variable y sólo se conocerá hasta el momento de la captura.

6) RUTINA DE VALIDACION EXTERNA

```
pruebas de la forma de captura
Nombre del eje X :   [Frecuencia                                    ]

Nombre del eje Y :   [f(x)                                          ]
    Escala eje Y :     (1) Automatica  (2)  Manual          [1]

    Acotamiento :      (1) No Logaritmico (2) Logaritmico   [1]

Otro eje vertical de referencia?   (si/no) [SI]

Nombre del eje Y -derecho-:   [Eje Y derecho                        ]
    Escala eje Y :     (1) Automatica  (2)  Manual          [1]

    Acotamiento :      (1) No Logaritmico (2) Logaritmico   [2]

Asociacion graficas-ejes   (I) -> eje Y izquierdo   (D) -> eje Y derecho
             1 [i]    2 [d]    3 [d]    4 [i]    5 [i]

-------------------------------------------------------------------------

Msj> Comenzar a capturar
Msj> Valor fuera de los limites
```

Figura 5-20

Debido a que existe la posibilidad de que una aplicación requiera verificaciones muy particulares para los campos se permite indicar que un campo sea validado con una rutina externa que sólo será necesario ligarla al programa de aplicación.

5.9.2 Forma De Uso

La información que el manejador necesita como entrada es la definición de las características de la forma que se desplegará en la pantalla. Asimismo, por medio de comandos se dan caracteristicas de validación o control a cada campo en particular. Una vez definidas las formas, el programa de aplicación del usuario llevará por medio de cuatro rutinas el control del despliegue y captura de información. A continuación se describen la manera de realizar lo anterior y por medio de un ejemplo se aplican dichos conceptos.

5.9.2.1 Definicion De Formas

Las formas se definen en un archivo creado con algún editor de textos. El nombre genérico para los archivos de formas es de acuerdo a la sintaxis de los archivos del sistema operativo VMS con la extensión **FRM**.

En cada archivo se podrán definir hasta 20 renglones de 80 columnas cada uno, en donde se indicarán los textos y campos que se desean desplegar en la pantalla de captura. Los **textos** son cadenas de caracteres que sólo muestran la información que se desea capturar o bién titulos en la forma. Los **campos** son zonas de la pantalla en donde se captura la información de acuerdo al tipo y longitud especificada.

La manera de indicar hasta dónde llega la forma de captura (no necesariamente se deben definir 20 renglones), es por medio del fín de archivo o con el primer comando indicado en la forma. Un **comando** permite definir las validaciones que se asociarán a los campos.

En la siguiente sección se explica la forma de indicar los textos, campos y comandos, así como las restricciones que cada uno de ellos tienen.

5.9.2.1.1 Textos Y Campos

Como ya se mencionó, se tienen hasta 20 renglones para definir los textos y campos que aparecerán como forma de captura. En cada renglón pueden existir sólo textos, únicamente campos o la combinación de ambos. Si se deja un renglón en blanco éste será respetado en la forma presentada.

Es muy importante hacer notar que en la posición en que se definan los campos y los textos en el archivo, aparecerán así en la forma de captura. La sintaxis es :

| texto | y \ campo \

En donde:

texto : Puede ser cualquier cadena de caracteres que estén definidos en el código ASCII.

campo : Es la repeticion de un caractor que define el tipo de información que tendrá ese campo. El número de repeticiones indicará la longitud del campo. Los tipos de campos se indican con:

A Alfanumérico.
C Caracter.
E Entero.
R Real.
S Si y NO.
X X.

A continuación se muestra un ejemplo de la definición de una forma de captura:

```
! Num funciones (1 a 5) :: \E\
! Funcion!  :Tabla X:     :Tabla Y: :Tipo de Union:
   :1:     \AAAAAAAAAA\ \AAAAAAAAAA\      \E\
   :2:     \AAAAAAAAAA\ \AAAAAAAAAA\      \E\
                                        :(1) lineas:
                                        :(2) marcadores:
```

Para determinar el número de campo que le corresponde a un texto o a un campo, se considera que un campo tiene asociado un texto, sin embargo, se permite que exista un campo sin texto y viceversa tomándose los siguientes criterios:

1) Si un texto y un campo están continuos en un mismo renglón, se contabilizan como un solo número de campo.

2) Si un campo continua a un texto en distinto renglón se consideran con números de campo distintos.

3) Si dos textos o dos campos están continuos en un mismo renglón, se contabilizan con números de campo distintos.

Para facilitar el conteo de campos se realizó un programa al que sólo es necesario indicarle el archivo en dónde se encuentra la forma y proporciona como salida un reporte indicando los números de campos, su longitud y su posición en forma secuencial. El reporte es útil para definir las restricciones de cada campo y los valores iniciales. Una descripción de este programa se encuentra en el capítulo 7.

Para el ejemplo anterior los campos y textos tendrían la numeración mostrada en el listado 5-XII.

Numero campo	Texto	Campo	Tipo
1	Num funciones (1 a 5) :	E	TC
2	Funcion:		Te
3	Tabla X:		Te
4	Tabla Y:		Te
5	Tipo de Union:		Te
6	1:	AAAAAAAAAA	TC
7		AAAAAAAAAA	Ca
8		E	Ca
9	2:	AAAAAAAAAA	TC
10		AAAAAAAAAA	Ca
11		E	Ca
12	(1) lineas:		Te
13	(2) marcadores:		Te

Nomenclatura: Ca - solo campo de captura
Te - solo texto
TC - campo de texto seguido de captura

Listado 5-XII

5.9.2.1.2 **Comandos**

Los comandos son los que indican las validaciones que se deben realizar a un campo en forma particular. Los campos no necesariamente deben tener comandos asociados a ellos.

Una vez que se defina un comando, no es posible indicar en la forma más textos o campos. La sintaxis de los comandos es la siguiente:

.Comando

El punto siempre deberá estar en la primera columna. Los comandos disponibles se explican a continuación (su funcionamiento se explicó en la sección anterior):

1) **REQUERIDOS**

.nR

En donde n es el número del campo que se desea tenga un valor obligado.

2) **LIMITES INICIAL Y FINAL**

.nLvi,vf

En donde:

n es el número del campo que se quiere limitar.
vi valor mínimo que puede tener el campo.
vf valor máximo que puede tener el campo.

3) OCULTOS

.n(val)O ci[,c2...]

En donde:

n es el número del campo que por su valor hace que se despliegen otros campos.
val es el valor con el que se activan los campos ocultos.
ci,c2... Son los campos o textos que serán desplegados cuando el campo n tenga el valor de val. Se permiten hasta 5 campos ocultos por comando, sin embargo un mismo campo y un mismo valor puede ser definido varias veces hasta ocultar máximo 25 campos.

El comando permite que un campo condionado pueda a su vez tener ocultos otros campos. Un ejemplo de campos ocultos es:

```
.10(3)O 11,12,13,14,15
.10(3)O 16,17
.10(2)O 18,19
.19(SI)O 20,21
```

En donde se puede observar que cuando el campo 10 tiene un valor de 3, se desplegarán los campos 11, 12, 13, 14, 15, 16 y 17. Cuando tenga el valor 2 se desplegarán los campos 18 y 19 y si a su vez el campo 19 tiene un valor de SI se desplegarán los campos 20 y 21.

4) EXCLUYENTES

.c1<>c2[<>c3...]

En donde:

ci,c2... Son los campos mutuamente excluyentes entres ellos Máximo se permite que un campo sea excluyente a otros 5.

5) RANGOS VARIABLES

.c1>c2

En donde:

c1 es el número del campo cuyo valor capturado deberá ser mayor al valor del campo c2.

Existe una variante, con la sintaxis:

.c1>=c2

En donde la condición se cambia a que el campo c1 deberá ser mayor o igual al campo c2

6) VALIDACION EXTERNA

.nVr

En donde:

n es el número del campo que se desea validar.
r es el número de validación que se quiere realizar.

La rutina externa deberá ser una función lógica que regrese un valor falso cuando se cumpla la validación y un valor verdadero cuando no se cumpla. El nombre que debera tener es **RutinaVal** y como parametros recibir una variable entera y una de tipo caracter que indican respectivamente el número de validación y el valor a validar.

El número de validación es con la finalidad que en la rutina se tengan más de una posible validación controlados por una estructura CASE (normalmente implementada en FORTRAN con la instrucción GOTO calculado).

Un ejemplo de una rutina de validación se encuentra en la sección de ejemplo de uso del MCA.

5.9.2.2 Rutinas De Control

Para utilizar las formas de captura definidas es necesario llamar a cuatro rutinas que dan las condiciones iniciales, la forma que se desea utilizar, así como el despliegue y captura de información.

La definición del área de captura sólo será necesario realizarse una vez, mientras que las otras rutinas podran ser repetidas tantas veces como sea necesario, permitiendo con esto que una forma pueda ser redesplegada o se utilicen más de una forma en una solo aplicación.

- **DEFINICION DEL AREA DE CAPTURA**

Antes de comenzar la captura de información, es necesario definir el área de la pantalla que sera ocupada por cada forma. Lo cual se realiza por medio de la rutina:

MCA_Inicia(RengInicial,RengFinal)

En donde:

RengInicial	INTEGER*4,Entrada
RengFinal	INTEGER*4,Entrada

RengInicial indica el renglón en donde se comenzará a desplegar la forma y **RengFinal** en último renglón que ocupará.

Es importante indicar que el primer renglón es el renglón superior de la pantalla y le corresponde el número 1, y el menor renglón será el inferior con el número asociado 24. También debe tomarse en cuenta que el primer renglón del archivo de la forma será colocado en la posición indicada por **RengInicial**, por lo que no es necesario dejar renglones en blanco al principio de la forma.

- **DEFINICION DE LA FORMA A USAR**

Para indicar que forma se desea desplegar, se invoca la siguiente función de tipo INTEGER*4:

```
MCA_IniciaForma(NombreForma)
```

NombreForma es una cadena de caracteres en donde se indica el nombre del archivo (sin tipo),que contiene la definición de la forma. Si el archivo no existe, la función regresará un valor de 1, si se encontró regresa 0.

Debido a que esta función se encarga de leer la forma, almacenando en las estructuras de datos correspondientes los textos, campos y caracteristicas asociadas, deberá ser llamada antes de las rutinas de despliegue y captura de información.

Si existen errores en la forma, estos serán indicados en la zona de diálogos definida.

Cuando se requiera utilizar otra forma de captura deberá llamarse nuevamente a esta rutina con el nombre del archivo que contiene dicha forma.

- **DESPLEGADO DE LA FORMA**

Una vez que se encuentra definida la forma en las estructuras de datos, se puede realizar su despliegue en la pantalla, por medio de la rutina:

```
MCA_DespliegaForma(Valores)
```

En donde **Valores** es una cadena de caracteres de longitud variable, que debe contener los valores iniciales que tendrán los campos de la forma. La manera de crear esta cadena es indicando los valores en forma secuencial respetando el orden y longitud de los campos. Por ejemplo, si

tenemos la siguiente forma:

```
: Numero de funciones (1,2,3,4,5) :: \E\
: Funcion:  :Tabla X:     :Tabla Y: :Tipo de Union:
  :1:     \AAAAAAAAAA\ \AAAAAAAAAA\      \E\
```

Los campos tendrán las siguientes características:

Número Campo	Longitud	Posición Inicial	Posición Final	Valor Inicial
1	1	1	1	1
2	10	2	11	VectorX
3	10	12	21	VectorY
4	1	22	22	3

Valores deberá ser = '1 VectorX VectorY3'
123456789 123456789 12

Es muy importante destacar que aún cuando los campos sean de diversos tipos, deberán ser indicados en forma de caracter, siendo responsabilidad del usuario la conversión dentro de la aplicación al tipo que tiene asociado.

Si esta rutina es invocada nuevamente sin definir otra forma, se desplegará la forma actual con la información indicada en **Valores**.

- **CAPTURA DE INFORMACION**

Para capturar los valores de la forma una vez que se ha desplegado, se invoca a la función de tipo INTEGER*4:

```
MCA_LeeValores(Valores,TextosTeclas,CodTeclas,NumTeclas,PantAnt)
```

En donde:

Valores	CHARACTER * (*),Salida
TextosTeclas	CHARACTER (*) * (*),Entrada
CodTeclas	INTEGER*4 (*),Entrada
NumTeclas	INTEGER*4,Entrada
PantAnt	LOGICAL,Entrada

La función regresa el código de la tecla funcional que se haya utilizado durante la captura de información. El manejador proporciona tres teclas funcionales y las despliega en la zona para ello asignada. Las teclas y su función son:

<ESC> para terminar la captura en forma normal (código 27).

<CTRL/A> para abortar la captura, es decir, ignorar los valores proporcionados (código 1).

<CTRL/F> permite regresar a la pantalla anterior, teniendo el usuario la responsabilidad de invocar a la forma que corresponda a la anterior (código 6) . Esta tecla sólo será activada cuando **PantAnt** tenga un valor verdadero.

Asimismo, existe la posibilidad de que cada aplicación defina a su vez otras teclas funcionales para cada forma que lo requiera, con sólo indicar en **TextosTeclas**, **CodTeclas** y **NumTeclas**, los textos que deberán aparecer para cada tecla funcional, su código ASCII y el número de ellas, respectivamente. Un ejemplo que muestra lo anterior es:

Se desean las siguientes teclas:

Tecla	Texto	Código
?	AYUDA	63
CTRL/B	BITACORAS	2

Los parámetros deberán tener los siguientes valores:

```
TextosTeclas(1) = '<?> Ayuda'
TextosTeclas(2) = '<CTRL/B> Bitacoras'
CodTeclas(1) = 63
CodTeclas(2) = 2
NumTeclas = 2
```

Cabe indicar que si **NumTeclas** es negativo, no se desplegarán las teclas que el manejador define inicialmente, pero seguirán cumpliendo su función.

Una vez que en la captura se detecte una tecla funcional, la aplicación tomará la decisión correspondiente y podrá nuevamente llamar a esta función, regresando el control al campo en donde se detectó la tecla y manteniendo el estado en donde se encontraba. Esto es con la finalidad de que sea posible pasar el control del programa a otra rutina u otro manejador sin que se pierda el estado actual de la forma de captura. Por ejemplo en una llamada a una ayuda, una vez que se haya realizado el despliegue correspondiente, se pueda regresar a la forma de captura con las mismas condiciones que existían en el momento de solicitar la ayuda.

PantAnt sólo es utilizado para indicar si existe una pantalla anterior o no. Si es verdadero, se detectará la tecla funcional <CTRL/F> que terminará la captura con las mismas características que <ESC> pero con distinto código de retorno.

En **Valores** se regresarán los valores que se hayan capturado siguiendo la misma filosofía que en **MCA_DespliegaForma**, es decir, en una cadena de caracteres respetando el orden y longitud de cada campo.

- INICIALIZACION DE OTROS MANEJADORES

Aún cuando no es necesario que la aplicación utilice las rutinas de los otros manejadores es necesario que se inicialicen estos, ya que internamente el MCA tiene relación con ellos. Estas deben llamarse una sola vez, antes que a la rutina **MCA_iniciaForma** y son:

- KB_init (Función Entera)
- MVT_inicia
- MDI_inicia(RengInicial,RengFinal)
- MTF_inicia(Renglon)

Para mayor información sobre estas rutinas consultar la descripción de ellas en este mismo capítulo.

5.9.2.3 Ejemplo

Con el objeto de mostrar la forma de utilizar el MCA se describen todos los aspectos anteriormente descritos a través de un ejemplo que contempla la captura de dos formas, asi como el control de desplegado de ayudas.

Debe observarse que se hace referencia a las rutinas de otros manejadores por necesidad del MCA y del Manejador de ayudas (MAY). Asimismo se muestra una posible forma de llevar el control de formas ligadas (CTRL/F). El manejo de ayudas no es requisito para el funcionamiento del MCA, sin embargo se lleva el control en el programa con la finalidad de mostrar la forma de interactuar con otros manejadores. Para mayor referencia sobre ellos consultar las secciones correspondientes en este mismo capítulo.

A continuación se muestra el programa de control de las formas (listado 5-XIII), después la definicion de ellas (listado 5-XIVy 5-XV) y por ultimo la rutina de validación que se utilizo (listado 5-XVI).

```
PROGRAM MCA_Pruebas

! Este programa muestra la forma de usar el manejador de captura,
! se utilizan asimismo las rutinas del manejador de ayudas
! con la finalidad de mostrar una pequenia aplicacion completa
! Fco. Javier Oropeza Morales
! Mauricio Acosta Gutierrez
! Gerardo Leon B.                    1-jul-1987  Disenio inicial

IMPLICIT NONE

CHARACTER
1     ValoresForma1 * 400,
1     ValoresForma2 * 200,
1     TextosTeclas(5) * 20

INTEGER
1     CodTeclas(5),
```

```
1     Edo1,
1     Edo2,
1     KB_INIT,
1     MCA_IniciaForma,
1     MCA_LeeValores,
1     MVT_open,
1     NumTeclas,
1     NumTopForma(2) /5,8/,
1     Valida(50)

 LOGICAL
1     Completa,
1     PantAnt

 TextosTeclas(1) = '<?> Ayuda'

 CodTeclas(1) = 63

 !!Se activan los manejadores, indicando las zonas de la pantalla
 !!que utilizan

 Edo1 = KB_Init()
 CALL MVT_Inicia
 CALL MCA_Inicia(2,19)
 CALL MDI_Inicia(20,23)
 CALL MTF_Inicia(24)
 CALL MAN_Inicia(1)                  !No es indispensable
 CALL MMU_Inicia(1,2,80,19,1)        !Necesaria para el MAY

 !Se inicializan los valores para las dos formas

 ValoresForma1 = ' '
 ValoresForma1(1:1)     = '1'              !Una sola funcion
 ValoresForma1(47:47)   = '1'              !Tipo Union linea graf 1
 ValoresForma1(93:93)   = '1'              !Tipo Union linea graf 2
 ValoresForma1(139:139) = '1'              !Tipo Union linea graf 3
 ValoresForma1(185:185) = '1'              !Tipo Union linea graf 4
 ValoresForma1(231:231) = '1'              !Tipo Union linea graf 5

 !Valores para la Forma 2
 ValoresForma2 = ' '
 ValoresForma2(41:41)   = '1'              !Escala automatica eje X
 ValoresForma2(62:62)   = '1'              !Escala no logaritmica en X
 ValoresForma2(103:103) = '1'              !Escala automatica eje Y1
 ValoresForma2(124:124) = '1'              !Escala no logaritmica Y1
 ValoresForma2(125:126) = 'NO'             !Otro Eje de referencia
 ValoresForma2(167:167) = '1'              !Escala automatica eje YD
 ValoresForma2(188:188) = '1'              !Escala no logaritmica YD
 ValoresForma2(189:193) = 'DDDDD'          !Todas las graficas a YD
 ValoresForma2(194:194) = '1'              !X y Y no tienen relacion

 NumTeclas = 1
 PantAnt = .FALSE.
 CALL MAN_Inserta('Forma de captura')
```

```
CALL MAN_Despliega
Edo1 = MCA_IniciaForma('XvsY1')
CALL MAY_Inicia('XvsY1.MAY')
CALL MCA_DespliegaForma(ValoresForma1)
DO WHILE ((Edo1 .NE. 1) .AND. (Edo1 .NE. 27))
    Edo1 = MCA_LeeValores(ValoresForma1,TextosTeclas,CodTeclas,
1                                              NumTeclas,PantAnt)
    IF (Edo1 .EQ. -63) THEN    !Tecla ? (Ayuda)
        CALL MAY_DespliegaTopicos(NumTopForma(1))
    ELSEIF (Edo1 .EQ. 27) THEN !Otra pantalla
        PantAnt = .TRUE.
        Edo2 = MCA_IniciaForma('XvsY2')
        CALL MAY_Inicia('XvsY2.MAY')
        CALL MCA_DespliegaForma(ValoresForma2)
        DO WHILE ((Edo2 .NE. 1) .AND. (Edo2 .NE. 27)
1                               .AND. (Edo2 .NE. 6))
            Edo2 = MCA_LeeValores(ValoresForma2,TextosTeclas,
1                                 CodTeclas,NumTeclas,PantAnt)
            IF (Edo2 .EQ. -63) THEN
                CALL MAY_DespliegaTopicos(NumTopForma(2))
            ELSEIF(Edo2 .EQ. 6) THEN !Pantalla anterior
                Edo1 = MCA_IniciaForma('XvsY1')
                CALL MAY_Inicia('XvsY1.MAY')
                CALL MCA_DespliegaForma(ValoresForma1)
                NumTeclas = 1
                PantAnt = .FALSE.
            ENDIF
        ENDDO
!   NOELSE
    ENDIF
ENDDO
CALL MAN_Elimina
END
```

Listado 5-XIII

```
¦ Numero de funciones (1,2,3,4,5) :¦ \E\
¦ Numero ¦
¦   de   ¦                                                          ¦Tipo de Union¦
¦ Funcion¦  ¦Tabla X¦    ¦Tabla Y¦    ¦Nombre de la Funcion¦  ¦de los Puntos¦
   ¦1¦    \AAAAAAAAAA\ \AAAAAAAAAA\ \CCCCCCCCCCCCCCCCCCCCCCCCC\      \E\
   ¦2¦    \AAAAAAAAAA\ \AAAAAAAAAA\ \CCCCCCCCCCCCCCCCCCCCCCCCC\      \E\
   ¦3¦    \AAAAAAAAAA\ \AAAAAAAAAA\ \CCCCCCCCCCCCCCCCCCCCCCCCC\      \E\
   ¦4¦    \AAAAAAAAAA\ \AAAAAAAAAA\ \CCCCCCCCCCCCCCCCCCCCCCCCC\      \E\
   ¦5¦    \AAAAAAAAAA\ \AAAAAAAAAA\ \CCCCCCCCCCCCCCCCCCCCCCCCC\      \E\
                                                        ¦(1) lineas¦
                                                        ¦(2) marcadores¦
                                                        ¦(3) marc. - lineas¦
                                                        ¦(4) marc conec a y = 0¦
                    ¦Titulos Generales de las Funciones¦
                  \CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC\
                  \CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC\
                  \CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC\
.1R
.1L1,5
.10R
.11R
.13L1,4
.17L1,4
.21L1,4
.25L1,4
.29L1,4
.1(1)0 10,11,12,13
.1(2)0 10,11,12,13
.1(2)0 14,15,16,17
.1(3)0 10,11,12,13
.1(3)0 14,15,16,17
.1(3)0 18,19,20,21
.1(4)0 10,11,12,13
.1(4)0 14,15,16,17
.1(4)0 18,19,20,21
.1(4)0 22,23,24,25
.1(5)0 10,11,12,13
.1(5)0 14,15,16,17
.1(5)0 18,19,20,21
.1(5)0 22,23,24,25
.1(5)0 26,27,28,29
```

Listado 6-XIV

```
!Nombre del eje X : ! \CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC\
    !Escala eje X : ! ! (1) Automatica  (2)  Manual         ! \E\
    !X Inicial :! \RRRRRRRRRR\ !X Final   :: \RRRRRRRRRR\
    ! Acotamiento : ! ! (1) No Logaritmico (2) Logaritmico ! \E\
!Nombre del eje Y : ! \CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC\
    !Escala eje Y : ! ! (1) Automatica  (2)  Manual         ! \E\
    !Y Inicial :! \RRRRRRRRRR\ !Y Final   :: \RRRRRRRRRR\
    ! Acotamiento : ! ; (1) No Logaritmico (2) Logaritmico ! \E\
!Otro eje vertical de referencia?! !(si/no)!\SS\
!Nombre del eje Y -derecho-: ! \CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC\
    !Escala eje Y : ! ! (1) Automatica  (2)  Manual         ! \E\
    !Y Inicial :! \RRRRRRRRRR\ !Y Final   :! \RRRRRRRRRR\
    ! Acotamiento : ! ! (1) No Logaritmico (2) Logaritmico ! \E\
!Asociacion graficas-ejes! !(I) -> eje Y Izquierdo! !(D) -> eje Y derecho!
              !1!\C\!   2!\C\!   3!\C\!   4!\C\!   5!\C\
!Escala eje X con respecto a eje Y  : ! !(1) Sin Relacion (2) Iguales! \E\
.3L1,2
.3(2)0 4,5
.5>4
.7L1,2
.10L1,2
.10(2)0 11,12
.12>11
.14L1,2
.16(SI)0 17,18,19,22.23
.16(SI)0 24,25,26
.16(SI)0 27,28,29,30,31
.19L1,2
.19(2)0 20,21
.21>20
.23L1,2
.27V1
.28V1
.29V1
.30V1
.31V1
.33L1.2
```

Listado 5-XV

```
      LOGICAL FUNCTION RutinaVal(NumeroVal,Valor)

      ! Se definen las rutinas de validacion que se haran a los valores de
      ! los campos que se indiquen en las formas de captura

      IMPLICIT NONE

      INTEGER
     1    NumeroVal
```

```
      CHARACTER
     1    Valor * (*)

          CHARACTER
     1        TeclaLeida * 1
          INTEGER
     1        LongTeclaLeida,
     1        TermEmp

        !!Simulacion de la estructura CASE
          GOTO (1) NumeroVal

         !ELSE
              RutinaVal = .FALSE.
              GOTO 99

     1        IF ((Valor .EQ. 'D') .OR. (Valor .EQ. 'd') .OR.
     1            (Valor .EQ. 'I') .OR. (Valor .EQ. 'i')) THEN
                  RutinaVal = .FALSE.
              ELSE
                  CALL MDI_Escribe ('Err> Este campo solo permite '//
     1                                  'valores D o I', 0, 1, 0)
                  CALL MDI_Escribe ('     Oprima '//
     1                    'cualquier tecla para continuar', 1, 1, 0)
                  CALL MDI_SuenaCamp (2)
                  CALL MDI_Lee(27, 0, 1, 0, 0, TeclaLeida, LongTeclaLeida,
     1                         TermEmp)
                  CALL MDI_Escribe (' ', 0, 0, 0)
                  RutinaVal = .TRUE.
              ENDIF
              GOTO 99
 99           CONTINUE
          Return
       end
```

Listado 5-XVI

5.9.3 Descripción Tecnica

Con el objeto de mostrar un panorama básico del diseño e implementación del manejador de captura, en esta sección se describe su estructura general, los seudocódigos de las rutinas de control, así una breve descripción de las rutinas de apoyo. Por último se explican las estructuras de datos que se utilizaron para almacenar las características de las formas y las asociadas a los campos.

5.9.3.1 Estructura General

El desarrollo del manejador se basó en las tres funciones básicas que éste debe cumplir, siendo éstas:

- Lectura de la forma.
- Despliegue en la pantalla de la forma.
- Captura y validación de los campos, así como movimiento dentro de la forma.

Estas funciones se realizan a través de tres módulos que son los que el usuario debe invocar para el control de las formas de captura. Estos módulos a su vez llevan el control de otras rutinas que son las que realizan directamente los pasos necesarios para el funcionamiento del manejador. En las siguientes secciones se detalla la lógica de estos módulos, así como las rutinas que invocan.

Debido a la forma como se diseño el Sistema Interactivo SER, el MCA interactúa con los otros manejadores, principalmente para la comunicación con el usuario. Todo el manejo de la pantalla como posición del cursor y escritura de textos se realiza por medio del Manejador de Ventanas con lo que el MCA es independiente del dispositivo ya que no interactua directamente con él. La lectura se hace por medio de un conjunto de rutinas para la detección de teclado llamada KB desarrollada por personal del CECAFI. Los mensajes se envian por medio del Manejador de Dialogos y se utiliza el Manejador de Teclas Funcionales para desplegar las teclas que usa el manejador asi como las que defina el usuario.

5.9.3.2 Seudocodigos Importantes

Para ilustrar los pasos basicos que se realizan en cada una de los modulos de control, se muestran sus seudocódigos generales en forma resumida sin que coincidan exactamente con el código de dichas rutinas ya que se pretende ilustrar su forma de operación mas que su estructura interna.

RUTINA MCA_IniciaForma

```
Inicio
Abre archivo de forma
IF Existe forma THEN
    MCA_IniciaForma <-- 0
    Inicializa valores de la estructura de datos
    NumCampo <-- 0
    DO WHILE (No fin de archivo y no haya comando y Lineas leidas < 20)
        IF Se define Texto THEN
            NumCampo <-- NumCampo + 1
            Lee texto y almacena en la estructura de textos
        ELSE
            IF Se define Campo THEN
```

```
            IF El anterior fue campo o Es nueva linea THEN
                NumCampo <-- NumCampo + 1
            ENDIF
            Lee campo y almacena en la estructura de campos
        ELSE
            Escribe "Definicion invalida"
        ENDIF
    ENDIF
    IF Fin de linea THEN
        Lee otra linea de la forma
    ENDIF
ENDDO
DO WHILE (Haya comandos y No sea fin de archivo)
    Valida el campo del comando
    IF Campo valido THEN
        Lee, valida y almacena en las estructuras de datos
        de acuerdo al comando
    ELSE
        Escribe "Campo no definido"
    ENDIF
ENDDO
ELSE
    Escribe "No existe la forma"
    MCA_IniciaForma <-- 1
ENDIF
FIN !IniciaForma
```

Listado 5-XVII

RUTINA MCA_DespliegaForma

```
Inicio
Desempaca los valores iniciales y almacena en las estructuras de datos.
Activa la zona de despliegue
Borra la zona de despliegue
NumCampo <-- 1
DO WHILE (NumCampo <= TotalCampos)
    IF Existe texto activo THEN
        Despliega texto con atributos
    ENDIF
    IF Existe campo activo THEN
        Despliega campo con atributos
        Verifica si oculta campos y despliega los que se activen con
        el valor inicial
    ENDIF
    NumCampo <-- NumCampo + 1
ENDDO
```

Listado 5-XVIII

RUTINA MCA_LeeValores

```
Inicio
IF NumTeclasFunc > 0 THEN
    Anexa textos de las teclas del manejador a las del usuario
ENDIF
IF Hay pantalla anterior THEN
    Activa CTRL/F
ENDIF
Despliega teclas funcionales
DO WHILE (TerminaCaptura)
    IF Nuevo campo o campo no cumplio validacion THEN
        Cambia a inverso el campo y despliega valor anterior
        Lee tecla
        IF No es terminador THEN
            Limpia el campo
        ENDIF
    ELSE
        !Es en el caso de regresar nuevamente a capturar
        Lee tecla
        IF Es terminador THEN
            Almacena el valor del campo
        ENDIF
    ENDIF
    IF No es terminador THEN
        CASE TipoCampo THEN
            1 : Lee y valida campo alfabetico
            2 : Lee y valida campo real
            3 : Lee y valida campo entero
            4 : Lee y valida campo si
            5 : Lee y valida campo caracter
            6 : Lee y valida campo x
        ENDCASE
    ELSE
        TerminaCaptura <-- FALSO
        CASE TipoTerminador THEN

            RETURN
            o <--  : IF Valor valido THEN
                        Verifica Ocultos y activa los que cumplan
                        Verifica Excluyentes
                        IF RETURN y no es ultimo campo THEN
                            Rescribe el valor del campo en normal
                            Avanza al siguiente campo
                            Rescribe el valor del campo actual en inverso
                        ELSE
                            IF <-- y no es primer campo THEN
                                Rescribe el valor del campo en normal
                                Regresa al campo anterior
                                Rescribe el valor del campo actual en inverso
                            ENDIF
                        ENDIF
                    ELSE
                        Escribe mensaje de error
                    ENDIF
```

```
            ESC o
            CTRL/F : IF Valor valido THEN
                         IF Hay campo requerido vacio THEN
                             Coloca posicion en campo requerido
                         ELSE
                             Coloca posicion en el primer campo
                             TerminaCaptura <-- Verdadero
                         ENDIF
                     ELSE
                         Escribe mensaje de error
                     ENDIF

            CTRL/A o
            Tec Usu: TerminaCap <-- Verdadero
        ENDCASE
    ENDIF
ENDDO
IF ESC o CTRL/F THEN
    Actualiza los valores capturados
    Empaca en el par_ámetro de retorno
ENDIF
MCA_LeeValores <-- CodigoTerminador
Fin !MCA_LeeValores
```

Listado 5-XIX

5.9.3.3 Rutinas

Las rutinas que en el diagrama de estructura anterior se mostraron, son brevemente explicadas a continuación en su función sin describirse los parámetros ni características particulares debido a que no la finalidad de este texto.

5.9.3.3.1 Lectura De La Forma

- **MCA_LeeForma.** Lee la forma de captura y determina si la información corresponde a un campo o un texto. Termina cuando detecta fin de archivo o un comando.

- **MCA_LeeTexto.** Se encarga de leer, validar y almacenar en las estructuras de datos correspondiente, las características de los textos definidos en una forma (Posición, longitud, contenido).

- **MCA_LeeCampo.** Lee, valida y almacena en las estructuras de datos (Posición, tipo, longitud) las características de cada campo definido en la forma de captura .

5.9.3.3.2 Lectura De Comandos

- **MCA_LeeComandos.** Lee cada una de las lineas del archivo en donde se esperan comandos y determina de que comando se trata para invocar a la rutina correspondiente de validación. También almacena en las estructuras de datos las características definidas. Dichas rutinas son:

  - MCA_ComExcluyentes
  - MCA_ComLimites
  - MCA_ComOcultos
  - MCA_ComRango

  Otras rutinas que auxilian en la validación y lectura de comandos son:

  - **MCA_LeeCampoComando.** Valida si un campo referenciado en un comando está o no definido en la forma.
  - **MCA_LeeTextoComando.** Valida si un texto que se referencía en un comando se encuentra definido en la forma.
  - **MCA_PosNoBlanco.** Regresa la posición a partir de la cual una linea contiene un caracter distinto a un espacio en blanco.

5.9.3.3.3 Empaque Y Desempaque De Valores

- **MCA_Empaca.** Almacena en forma secuencial en una cadena de caracteres toda la información correspondiente a los campos definidos de acuerdo a su longitud.

- **MCA_Desempaca.** Vacia la información de una cadena de caracteres en la estructura de datos correspondiente a los campos.

5.9.3.3.4 Lectura De Campos

En esta sección se agrupan las rutinas que realizan la lectura de la información de cada campo de la forma de acuerdo a su tipo y longitud. Todas esta rutinas realizan la lectura, el eco y el borrado de cada caracter. Lo anterior se lleva a cabo siguiendo un analizador sintáctico propio de cada tipo de campo. Regresan el valor capturado una vez que es detectada una tecla funcional. Las rutinas son:

- MCA_LeeAlf
- MCA_LeeCar
- MCA_LeeEnt

- MCA_LeeReal
- MCA_LeeSi
- MCA_LeeX

Como apoyo a estas rutinas se cuenta con otras tres que realizan directamente la lectura de cada tecla (detectando las teclas funcionales), el borrado y el eco de caracteres, respectivamente. Dichas rutinas son:

- MCA_LeeTecla
- MCA_BorraCar
- MCA_EscCar

5.9.3.3.5 Validación De Campos

- MCA_ValidaCampo. Realiza la validación del valor recién capturado verificando si cumple con las condiciones que el usuario definió en la forma (Rangos, excluyentes, límites). Si cumple las condiciones regresa un valor falso.

- MCA_VerExc. Verifica si un campo es excluyente a otros, si lo es cancela los valores de los otros campos.

- MCA_VerificaOcultos. Determina si el valor de un campo recién capturado activa a otros campos o si el cambio de valor debe ocultar a algunos campos. Si se cumplen estas condiciones será necesario borrar o escribir los campos y textos afectados.

  Debido a que es factible tener hasta 10 niveles de ocultamiento, fué necesario hacer esta verificación en forma recursiva, es decir, para cada campo que se oculte o se despliegue en la pantalla, se verifica si a su vez oculta o activa a otros campos. Para lograr esto, se realizó una simulación de recursividad, utilizando un stack. Las rutinas utilizadas son:

  - MCA_OcultosAnidados
  - MCA_MeteAStack
  - MCA_SacaDeStack

  Asimismo, existe la posibilidad de que un campo pueda ser ocultado o desplegado por otro con distintos valores, por lo que se realizó una rutina que indica en una estructura de datos, si un campo fué activado con el valor actual del campo del que depende. Esta rutina es:

  - MCA_IniciaRecienAct.

5.9.3.3.6 Rutinas De Otro Manejadores

A continuación se enlistan las rutinas utilizadas de otros manejadores, con la finalidad de mostrar la relación del MCA con ellos. Dentro de este capítulo se encuentra una amplia descripción de cada una de ellas.

- Manejador de ventanas (MVT)

  El MCA realiza todas las escrituras a la pantalla a través del manejador de ventanas por requisitos de éste y con la finalidad de mantener cierta independencia con respecto del dispositivo de salida. Las rutinas que se utilizan son:

  - MVT_DefineZona
  - MVT_BorraPantalla
  - MVT_AlmacenaZona
  - MVT_ExtraeZona
  - MVT_ActivaZona
  - MVT_ImprimeTexto
  - MVT_ColocaCursor

- Manejador de dialogos (MDI)

  Los mensajes de error y de información que el manejador hace al usuario los realiza en la zona destinada para ellos por medio del MDI, con las siguientes rutinas:

  - MDI_Escribe
  - MDI_SuenaCamp

- Manejador de teclas funcionales (MTF)

  Las teclas que se desean desplegar en la zona de teclas funcionales son mostradas por medio de la rutina MTF_Despliega.

- Detección de teclado (KB)

  La detección de las teclas se realiza con la rutina KB_Read.

5.9.3.4 Estructuras De Datos

Las estructuras de datos que se utilizan son empleadas para almacenar las características de la forma (campos y textos) y de las validaciones que se realizan a cada campo.

La definición de las estructuras se realizó en base a los principales requerimientos que presenta el manejador como rapidez en el movimiento entre campos y en la validación de los valores capturados. Para lograr esto, se utilizaron esquemas de rápido acceso que no involucran mapeos de valores que aunque son más eficientes requieren mayor tiempo de cálculo. Una de las

desventajas de este tipo de estructuras es que pueden llegar a tener una gran porosidad que sin embargo no afecta en grán medida debido al esquema de manejo de memoria de VMS y de la mayoria de los sistemas operativos.

La grán mayoria de las estructuras utilizadas son arreglos de una o dos dimesiones en donde el primer indice hace referencia directamente al número del campo y si existe un segundo indice éste indica alguna característica especial. Como un ejemplo de las estructuras usadas se muestra en la figura 5-21 la que define las características de los campos.

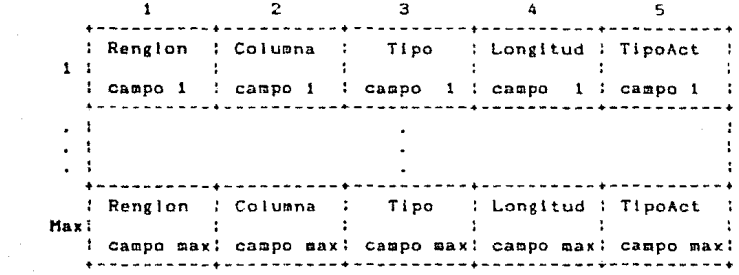


Figura 5-21

La longitud de los estructuras esta en base al número máximo constante de campos que se pueden definir y al número de características que almacena.

A continuación se hace una breve descripción de las estructuras utilizadas, las que se pueden dividir en dos grupos.

- DEFINICION DE LA FORMA

  Se utilizan cuatro arreglos que definen las características (posición, longitud, activo o no) de los textos, de los campos y las cadenas de caracteres que corresponde a los textos y los valores de los campos.

- VALIDACIONES A LOS CAMPOS

  Para cada posible validación se utiliza una estructura distinta en donde se almacenan las valores o los campos contra los que se validará cada campo. Son seis el número de estas estructuras que definen límites, rangos, campos excluyentes, requeridos, ocultos.

Asimismo, existe una estructura formada por una serie de variables que definen el estado de la forma y que es utilizado básicamente para permitir que el usuario pueda salir de capturar una forma y pueda regresar posteriormente con las mismas condiciones con las que salió. Esta estructura indica posición del cursor, número, longitud y atributo del campo y el estado del analizador sintáctico correspondiente al tipo de información que se este capturando .

Con la finalidad de evitarle al usuario el uso de parámetros "dummy", el manejador utiliza el recurso de variables comunes en rutinas que el usuario controla.

5.10 MANEJADOR DE MENUS.

En los siguientes párrafos se tratará el tema del Manejador de Menús, incluyendo una descripción funcional del mismo, así como una lista de las rutinas que integran al Manejador. Por ultimo se incluye un ejemplo que muestra su uso.

5.10.1 Generalidades.

El Manejador de Menús es un conjunto de rutinas encargadas del desplegado de los textos de un menú de opciones. Una vez mostradas las opciones, el Manejador es responsable de esperar a que el usuario seleccione una opción; también se encarga de borrar la pantalla de la zona de desplegado antes de colocar las opciones.

Para seleccionar una opcion, el usuario debe oprimir las flechas (arriba o abajo) de la terminal. En un principio la primera opción aparece con el atributo en inverso; al mismo tiempo que el usuario oprime una de las flechas, el Manejador cambia el atributo del texto que se encuentra en inverso, al atributo normal, y escribe la siguiente opción con fondo en inverso. De esta manera, aparenta que un recuadro con fondo invertido, "camina" conforme el usuario oprime una de las flechas. La flecha "abajo" hace que el recuadro camine hacia la parte inferior de la pantalla, la flecha "arriba", provoca que el recuadro camine en sentido inverso. Cuando se ha llegado a la parte final de la lista de opciones, se pasa a la primera opción del menú, del tal suerte que semeja a una lista circular.

También existe el método rápido de seleccion de opciones, que consiste en oprimir en el teclado, la letra inicial del texto de la opción que se desea seleccionar. Así por ejemplo, si existen cuatro opciones:

Altas
Cambios
Bajas
Consultas

Cuando el usuario oprime la letra "C", el recuadro avanza a la opción "Cambios". Alguna veces existen varias opciones que comienzan con la misma letra; cuando esto sucede, el Manejador de Menús va colocando en una por una, el recuadro en las opciones que comienzan con esa letra. De esta forma en el ejemplo, si el usuario vuelve a oprimir "C", el recuadro avanzará a la opción "Consultas". También en este caso, la lista es circular, de tal suerte que si el usuario oprime "B", el Manejador de Menús recorre todas las opciones, desde la opción donde se encuentra, hasta el final de la lista, buscando una opción que empiece con "B"; si no la encuentra, empieza ahora con el inicio de la lista, hasta encontrar una opción que inicie con la letra "B" o hasta regresar a la opción inicial de la cual partió. En caso de que no encuentre la opción, el manejador suena la campana una vez. Para el ejemplo, el recuadro se

posicionaría en la opcion "Bajas".

Una vez posicionado el recuadro en la opción deseada, el usuario simplemente tiene que oprimir "RETURN", para indicarle al Manejador de Menús que desea esa opción.

El Manejador de Menús permite tener varios bloques de menús de opciones trabajando a un tiempo; de una forma análoga a tener varios archivos abiertos. Por tal motivo el programa de aplicación debe indicar al Manejador, qué bloque de opciones se debe desplegar u operar; esto se realiza pasándole como parámetro un número entero, que indica cuál de todos los bloques de menús presentes se activará.

La opción de salida siempre existe en un menú, aunque el usuario no lo indique; esto es a través de la tecla funcional <ESC>, de tal forma, que cuando el usuario oprime esta tecla, el programa de aplicación debe suponer que el usuario ya no desea trabajar con este menu.

Este Manejador se encarga en forma automática de guardar la posición del cursor, el atributo activo y las dimensiones de la Frontera activa, antes de desplegar o de operar el menú. Esto fundamentalmente para resguardar la información de la pantalla que no pertence al menu; y así, cuando el desplegado o la selección haya concluido, el Manejador automáticamente deje el cursor, el atributo y la Frontera, tal y como se encontraban, antes de que el Manejador de Menús, tomara el control del video. A continuación se presenta una descripción de las rutinas encargadas de realizar todo lo mencionado en los párrafos anteriores.

5.10.2 Rutinas.

En las siguientes lineas se describe la implentación práctica del Manejador de Menús; después se procede con una lista de las rutinas que integran el Manejador. En la lista se incluye una descripción funcional y un desgloce de los parámetros involucrados.

Una de las caracteristicas importantes del Manejador, es que en verdad, su diseño no fué independiente de los demás. Cuando éste se realizó, se previó que en un futuro, en el Sistema Interactivo se tendrían que interrelacionar el Manejador de Menús con el Manejador de Ayudas; de tal suerte que al operar el menu, el Manejador de alguna manera tendría que detectar si el usuario oprimía "?" y en ese caso, desplegar la ayuda correspondiente; con la restricción de que el control regresara al Manejador de Menús. De esta condición se desprendieron dos conclusiones importantes; la primera es que el Manejador debe detectar teclas funcionales y la segunda, es que de algún modo, tiene que desplegar la ayuda correspondiente y luego recuperar el control.

En un principio se pensó que el propio Manejador de Menús invocara directamente al Manejador de Ayudas. Sin embargo, esto disminuía la flexibilidad que se desaba tener con el manejador, pues se hacía dependiente de su entorno, al tener que "conocer" qué archivos de ayuda había que desplegar. Posteriormente con el despliegado de los directorios de tablas y gráficas, se descubrió que era aún más crítico de lo que se pensaba. Lo cual hizo suponer que la solución planteada no era la más correcta, pues no se deseaba elaborar un

Manejador de Menús que pudiera desplegar ayudas y directorios, ya que era probable que muy pocas aplicaciones tuvieran ayudas o directorios que desplegar.

Ante este dilema, se ocurrió que la solución más adecuada era separar la sección del desplegado del menú de la de operación. De esta forma, en una parte del programa se despliega el menú, luego en otra sección, simplemente se opera. Y así, cuando el Manejador detecta que se ha oprimido una tecla funcional, "?" por ejemplo, la operación termina. Conociendo este hecho, el programa de aplicación puede invocar al Manejador de Ayuda o desplegar los directorios, según sea el caso. Una vez terminado el despliegue, el propio programa regresa el control a la sección de operación del Manejador de Menús, invocando a la rutina correspondiente. Esta estructura evita redesplegar el menú de opciones cuando no se requiere; tal caso sucede con la ayuda y los directorios que abren una ventana para cumplir su propósito. Una vez que terminan su ejecución y cierran su ventana, se recupera la parte del menú que ocultaba la ventana y por tal motivo, el menú completo aparece perfectamente en la pantalla. De tal suerte que se puede operar el menú sin necesidad de desplegarlo nuevamemte.

El Manejador está integrado por cuatro rutinas. La primera de ellas se encarga de definir una Zona con la rutina **MVT_DefineZona**, correspondiente a la porción de la pantalla que comprenderá la región de desplegado del menú de opciones. Además recibe el numero de bloque del menú, ya que como se recordará, es posible tener varios bloques de menús presentes a un tiempo, en la pantalla.

Las siguientes dos rutinas son las responsables del desplegado y de la operación del menú. Esencialmente lo que hacen es salvar el cursor, el atributo y las dimensiones de la Frontera activa con la rutina **MVT_AlmacenaZona**. La parte de desplegado borra la pantalla, despliega las opciones y coloca el recuadro en la primera opción. Esta rutina se encarga de encontrar el mejor espaciamiento entre renglones para localizar las opciones y hacer más agradable su presentación; el espaciamiento se guarda en variables "COMMON" que comparten las dos rutinas. También permite, a discresión del usuario, dibujar una línea de separación en la parte superior de la zona y rotular las opciones con un título. Igualmente, es el manejador el que se encarga de desplegar las teclas funcionales asociadas al menú.

La rutina de operación se encarga de detectar las teclas que oprime el usuario con la rutina **MDI_Lee**, verifica si la tecla es una de las teclas funcionales, si no lo es, verifica si es una flecha para mover el recuadro en la direcccion adecuada; si no es una flecha, prueba si la tecla oprimida es la inicial de una de las opciones, para mover el recuadro a la posición correcta. Finalmente, si tampoco existe una opción que comience con la letra correspondiente a la tecla oprimida, suena la campana una vez. La rutina de operación regresa al programa de aplicación qué tecla funcional oprimio el usuario o el número de opción que el usuario seleccionó, según sea el caso.

Tanto la rutina de despliegue como la de operación requieren que se les indique con qué bloque de menú se desea trabajar. Ambas rutinas al final, recuperan el cursor, el atributo y las dimensiones de la Frontera con **MVT_ExtraeZona**.

La cuarta rutina es la contraparte de la primera. Es decir, cancela un bloque de menú; o en otras palabras, borra la zona destinada a un bloque de menús.

Como se podrá suponer, el manejo de varios bloques de menús, simplemente se realizó con el Manejador de Zonas, ya que éste se encarga de delimitar las porciones de la pantalla destinadas a cada uno de los menús de opciones. Para este efecto el Manejador de Menús lleva en "COMMON", un arreglo que almacena el número de zona asignado a cada bloque de menú, los espaciamientos entre renglones y las dimensiones físicas de cada bloque de menú.

Es conveniente mencionar que como el Manejador de Menús invoca a rutinas del Manejador de Ventanas, del Manejador de Diálogos y del Manejador de Teclas Funcionales, para emplear al Manejador de Menús es necesario llamar a las rutinas **MVT_Inicia**, **MDI_Inicia**, **KB_Init ()** y **MTF_Inicia**, al principio del programa de aplicación. Todas las rutinas se realizaron de acuerdo a los principios de la programación estructurada, empleando FORTRAN-77; les caracteriza el prefijo MMU. A continuación se proporciona la lista de rutinas del Manejador de Menús.

SUBROUTINE MMU_Inicia (ColIzq, RenIni, ColDer, RenFin, NumIdentMenu)

```
ColDer      INTEGER * 4, Entrada
            ! Columna Derecha de la Zona del menú (coord. absolutas)
ColIzq      INTEGER * 4, Entrada
            ! Columna Izquierda de la Zona del menú (coord. abs.)
NumIdentMenu    INTEGER * 4, Salida
                ! Número de identificación del menú. Es
                !un número del 1 al 5 indicando el bloque a activar.
RenIni      INTEGER * 4, Entrada
            ! Renglón Inicial de la Zona del menú (coord. abs.)
RenFin      INTEGER * 4, Entrada
            ! Renglón Final de la Zona del menú (coord. abs.)
```

Esta subrutina debe llamarse antes de cualquier otra del manejador. Establece una área de la pantalla en la cual el menú será desplegado. Debe indicarse además de qué bloque de menú se trata; se permite hasta un máximo de cinco bloques. Es decir, hasta cinco zonas de menús simultáneas en la pantalla y cada una, con un menú distinto. Por medio de **NumIdentMenu** es posible poder cambiarse de menú, sin perder el estado del menú anterior.

INTEGER FUNCTION MMU_Despliega (Opciones, NumOpc, NumDes, TipoDespliegue, Titulo, NumIdentMenu)

```
Opciones        CHARACTER (*) *(*), Entrada
                ! Textos de las opciones a desplegar
Titulo          CHARACTER (*) *(*), Entrada
                ! Título que rotula al Menú.
NumDes          INTEGER * 4, Entrada
                ! Número de opciones a desplegar. Puede ser menor
                !a NumOpc.
NumIdentMenu        INTEGER * 4, Entrada
                    ! Número de identificación del bloque de menú.
NumOpc          INTEGER * 4, Entrada
                ! Número de opciones que componen al menú completo.
TipoDespliegue      INTEGER * 4, Entrada
                    ! Indica qué tipo de encabezado se escribe:
                    !  Bit          Valor                   Valor
```

	0	1
0	Linea de Separación	No escribe la linea
1	Escribe el título	No escribe el título

Esta función despliega la opciones en la pantalla. A petición del usuario coloca una linea de separación y opcionalmente puede escribir un título, el cual va enmarcado y se despliega en la parte superior de la zona. Regresa los siguientes valores:

Bit encendido	Condición
Ninguno	No hay error
1	Texto de las opciones más grandes que la pantalla
2	Número de opciones mayor al número de renglones

SUBROUTINE MMU_Acepta (Opciones, NumDes, TeclasFun, TeclasTex, NumTeclas, NumIdentMenu, Opcion)

```
Opciones        CHARACTER (*) *(*), Entrada
                ! Texto de las opciones.
TeclasTex       CHARACTER (*) *(*), Entrada
                ! Texto de las teclas funcionales. Sigue las
                !restricciones del manejador de Teclas Funcionales.
NumDes          INTEGER * 4, Entrada
                ! Número de opciones a desplegar.
NumIdentMenu        INTEGER * 4, Entrada
                    ! Número de identificación del bloque de menú.
NumTeclas       INTEGER * 4, Entrada
                ! Número de Teclas Funcionales.
Opcion          INTEGER * 4, Salida
                ! Opción seleccionada por el usuario. Regresa los
                !siguientes valores:
                ! = 0      Esc
                ! < 0      Tecla Funcional
                ! > 0      Opción del Menu
TeclasFun (*)       INTEGER * 4, Entrada
                    ! Contiene los ASCII de cada tecla funcional.
```

Esta subrutina se encarga de pedir al usuario la opción del menú y avisa que está lista para recibir la opción, enviando el mensaje **"Msj> Seleccione una opcion"**. El usuario selecciona la opción localizando el recuadro en inverso, en la opción deseada, por medio de las teclas (flecha arriba y flecha abajo) y finalmente oprimiendo la tecla de RETURN. Otra forma de hacerlo es oprimiendo la tecla correspondiente a la inicial del texto de la opción deseada y posteriormente oprimiendo RETURN. Recibe también un arreglo de teclas funcionales y su texto; así pues, detecta si el usuario oprimió alguna de esas teclas; el número máximo de teclas funcionales es 14. El usuario debe indicar qué bloque de menú se trata.

La rutina regresa el siguiente valor en Opcion

Valor de Opcion	Significado
Mayor a 0	El valor indica el número de opción seleccionada
Menor a 0	El valor sin considerar el signo indica el número

de tecla funcional oprimida
0 El usuario oprimió Esc lo cual se debe interpretar como salida del Menú.

SUBROUTINE MMU_Termina (NumIdentMenu)

NumIdentMenu ! Número de identificación del bloque de menú

Esta rutina debe emplearse cuando se desea dejar disponible un bloque de menú para otros usos. Básicamente todo menú que inicia con **MMU_Inicia** debe concluir con **MMU_Termina.**

5.10.3 Ejemplo.

El listado 5-XX proporciona un ejemplo del uso del Manejador de Menús, con dos bloques de opciones. Al principio se declaran estos dos bloques, el primero tiene 10 opciones y el segundo, solamente cuatro. También cada bloque tiene sus propias teclas funcionales y se declaran tanto los textos, como los códigos ASCII. Posteriormente se definen los números de opciones que se van a desplegar en cada bloque; para posteriormente detallar los renglones y columnas que delimitan a las zonas de cada bloque de menú. Después se llaman a todas las rutinas de incialización y se despliegan los dos bloques de menús.

Lo que sencillamente hace el programa, es activar el primer menú y pedir al usuario que seleccione una opción o que oprima una tecla funcional. Cuando el usuario hace esto, se muestra en la zona de diálogos, el valor que se regresa en el parámetro Opcion. Después realiza lo mismo, pero para el segundo bloque de opciones. En la figura 5-22 se muestra la pantalla resultante al ejecutar este programa.

```
PROGRAM Ejemplo
IMPLICIT NONE

CHARACTER
1    Opciones1 (10) *30      /
1                            'Altas',
1                            'Bajas',
1                            'Modificaciones',
1                            'Consultas',
1                            'Opcion 5',
1                            'Opcion 6',
1                            'Opcion 7',
1                            'Nuevos Empleados',
1                            'Opcion 9',
1                            'Directorio de Empleados'/,
1    Opciones2 (4) * 30      /
1                            'Bloque 2 Opcion 1',
1                            'Opcion 2',
1                            'Bajas',
1                            'Consultas'/,
1    OpcionCar * 2,
```

```
1     Titulo * 30,
1     TeclasTex1 (3)*30       /
1                             '? Ayuda',
1                             '<TAB> Directorio',
1                             '<LineFeed> Revisa Cta.'/,
1     TeclasTex2 (2)*30       /
1                             '@ Cancelar Cambios',
1                             '$ Copiar Datos'/,
1     Tecla * 1

INTEGER
1     Condicion,
1     Colizq1,
1     ColDer1,
1     Colizq2,
1     ColDer2,
1     Kb_init,
1     NumDes1,
1     NumDes2,
1     NumOpc1,
1     NumOpc2,
1     NumTeclas1,
1     NumTeclas2,
1     MMU_Despliega,
1     Opcion,
1     Renini1,
1     RenFin1,
1     Renini2,
1     RenFin2,
1     RenDialIni,
1     RenDialFin,
1     RenMTF,
1     TeclasFun1 (3)      /
1                          63,   !?
1                          9,    !<TAB>
1                          10    !<LineFeed>
1                         /
1     TeclasFun2 (2)      /
1                         64,   !@
1                         35/   !$

Titulo = 'Menu de ejemplo'

NumDes1 = 6     !Numero de opciones a desplegar del primer bloque
NumOpc1 = 10    !Numero de opciones totales del primer bloque

NumDes2 = 4     !Numero de opciones a desplegar del segundo bloque
NumOpc2 = 4     !Numero de opciones totales del segundo bloque

!Zona de menus 1er bloque
Renini1 = 2
RenFin1 = 17
Colizq1 = 1
ColDer1 = 40
```

```
!Zona de menus del 2do bloque
RenIni2 = 9
RenFin2 = 15
ColIzq2 = 50
ColDer2 = 78

!Zona de dialogos
RenDialIni = 18
RenDialFin = 23

!Zona de Teclas Funcionales
RenMTF = 24

NumTeclas1 = 3     !Numero de teclas funcionales bloque 1

Numteclas2 = 2     !Numero de teclas funcionales bloque 2


CALL MVT_Inicia
Condicion = KB_Init ( )

!Organizacion de la pantalla
CALL MDI_Inicia  (RenDialIni,RenDialFin)
CALL MTF_Inicia  (RenMTF)

!1er bloque de menu
CALL MMU_Inicia  (ColIzq1, RenIni1, ColDer1, RenFin1, 1)

!2do bloque de menu
CALL MMU_Inicia  (ColIzq2, RenIni2, ColDer2, RenFin2, 2)

!Despliega el menu de opciones del bloque 1
Condicion = MMU_Despliega (Opciones1, NumOpc1, NumDes1, 0,
1                          Titulo, 1)

!Despliega el menu de opciones del bloque 2
Condicion = MMU_Despliega (Opciones2, NumOpc2, NumDes2, 2,
1                          ' ', 2)

DO WHILE (.TRUE.)

    !Lee la opcion del menu bloque 1
    CALL MMU_Acepta (Opciones1, NumDes1, TeclasFun1, TeclasTex1,
1                    NumTeclas1, 1, Opcion)

    !Convierte en caracter la variable opcion y la imprime en la zona
    !de dialogos
    WRITE (OpcionCar, '(I2)') Opcion
    CALL MDI_Escribe ('Opcion =  '//OpcionCar, 0, 1, 0)

    !Lee la opcion del menu bloque 2
    CALL MMU_Acepta (Opciones2, NumDes2, TeclasFun2, TeclasTex2,
1                    NuMteclas2, 2, Opcion)
```

```
    !Convierte en caracter la variable opcion y la imprime en la zona
    !de dialogos
    WRITE (OpcionCar, '(I2)') Opcion
    CALL MDI_Escribe ('Opcion =  '//OpcionCar, 0, 1, 0)

ENDDO

END
```

Listado 5-XX

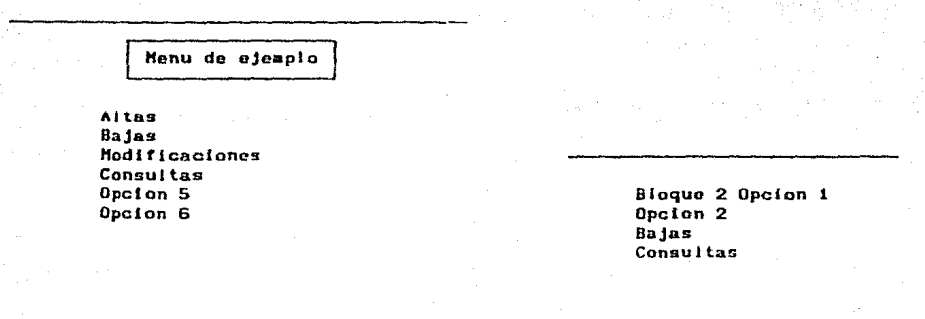


Figura 5-22

Es importante mencionar que no es indispensable emplear siempre más de un bloque de opciones. En el ejemplo, se puede eliminar todo lo concerniente al bloque dos y el programa sigue funcionando.

CAPITULO 6

RUTINAS GRAFICAS

Como ya se mencionó en capítulos anteriores, es muy importante proporcionar no solo la posibilidad de obtener reportes gráficos a través del Sistema Interactivo, sino también por medio de programas de aplicación del usuario. Es por esta razón que todo lo concerniente a las rutinas gráficas se diseñó con la misma filosofía de los manejadores para la interfaz interactiva, es decir, totalmente desligadas del programa de aplicación, en este caso del sistema SER Interactivo.

En este capítulo se explicarán las características generales de las rutinas gráficas, las secuencias de llamado necesarias para la obtención de los reportes y el diseño general de cada una de ellas.

6.1 CARACTERISTICAS

El objetivo básico de las rutinas gráficas a disposición del usuario es obtener en forma sencilla reportes gráficos de tipo estadístico y matemático desde un programa de aplicación.

El desarrollo fue a través del diseño de tres capas o niveles de rutinas, de las cuales el usuario solo necesita llamar al nivel más alto para obtener sus reportes. Sin embargo, también puede utilizar los otros dos niveles si sus aplicaciones así lo requieren.

Es importante indicar que las rutinas desarrolladas no incluyen las primitivas gráficas que directamente generan los reportes, sino que se utilizó un paquete ya instalado en el equipo VAX.

A continuación se explica la estructura general de las rutinas, así como los resultados que se pueden obtener.

6.1.1 Estructura General

En el diseño de las rutinas se observó que los distintos reportes tenían características muy comunes y que además era muy importante tener un estándar de rutinas gráficas básicas que mantuvieran el resto de las rutinas independientes del paquete de primitivas a utilizar. Por estas razones, se diseñaron las rutinas en capas, es decir, cada capa o nivel superior de rutinas se apoya

exclusivamente en las rutinas del mismo nivel o de niveles inferiores.

Se desarrollaron tres niveles de rutinas los cuales se explican a continuación: El nivel superior es el nombrado Sistema Emisor de Reportes (SER) el cual será utilizado por el usuario para obtener sus reportes gráficos. Estas rutinas se apoyan en el nivel de Rutinas de Propósito Específico (RPE) el cual consiste en rutinas que realizan operaciones gráficas comunes a varios tipos de reportes. Por último, está el Estándar Gráfico (EGR) en el cual se agrupan las rutinas que invocan directamente al paquete de primitivas gráficas. En las secciones posteriores se explican ampliamentes cada uno de dichos niveles.

6.1.2 Presentación De Los Reportes

Para la obtención de los reportes se utiliza un Printer-Plotter PRINTRONIX 600. El sistema permite que el usuario defina el tamaño de la hoja de acuerdo a sus necesidades. Una **Hoja de dibujo** será en donde estén contenidos los reportes, teniendo el formato general de presentación de la figura 6-1.

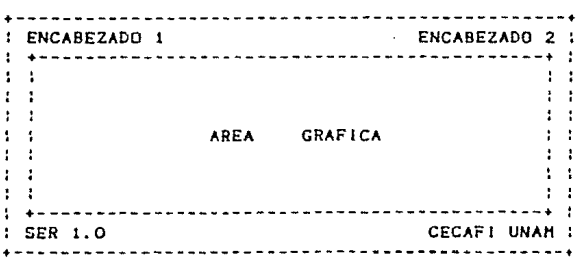


Figura 6-1

Asimismo, es posible definir desplazamientos o márgenes en los cuatro lados de la hoja, con la finalidad de reservar espacios para engargolados o escritos sobre la misma hoja del reporte. Cuando existan estos desplazamientos se trazará un recuadro con líneas punteadas que indiqua el tamaño de la hoja. Dicho recuadro tiene su principal utilidad cuando el tamaño del papel es mayor al tamaño de la hoja definida por el usuario y éste necesite recortar el reporte.

El área gráfica y el recuadro de identificación recibirán el nombre de **Región Gráfica**, es decir, es el espacio disponible después de los desplazamientos, como se muestra en la figura 6-2.

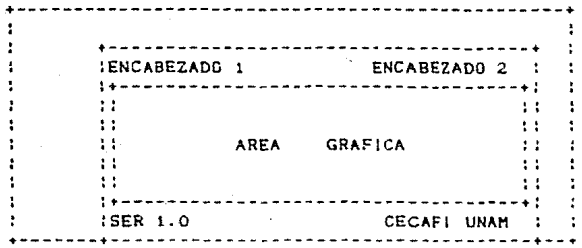


Figura 6-2

En el área gráfica se pueden dibujar 1, 2 ó 4 reportes gráficos distintos. Se llamará Zona al espacio físico que ocupe cada reporte. Las figuras 6-3 y 6-4 muestran los casos en que el área gráfica se divide en 2 y 4 zonas (las lineas punteadas que separan cada zona no se dibujan en la gráfica):

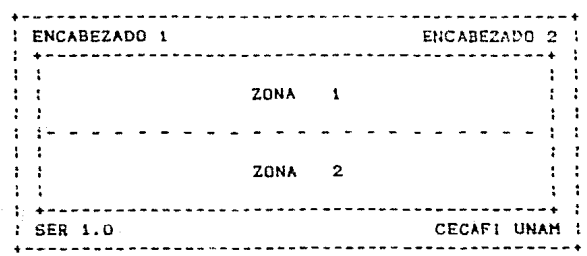


Figura 6-3

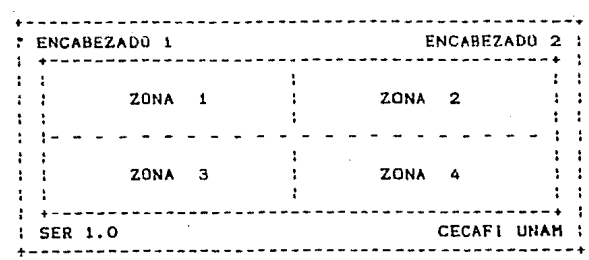


Figura 6-4

A cada reporte se le pueden definir distintas áreas para mejorar su presentación. En la figura 6-5 se muestran las áreas posibles para todos los reportes con excepción de diagramas circulares:

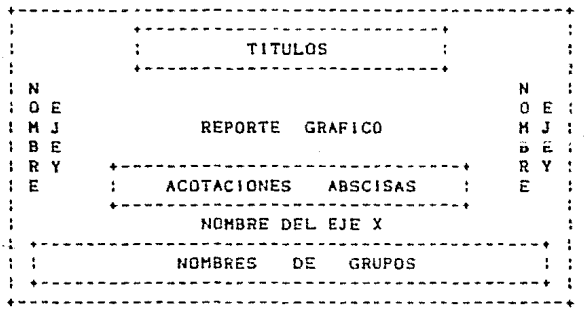


Figura 6-5

Para el caso de diagramas circulares las áreas existentes se observan en la figura 6-6.

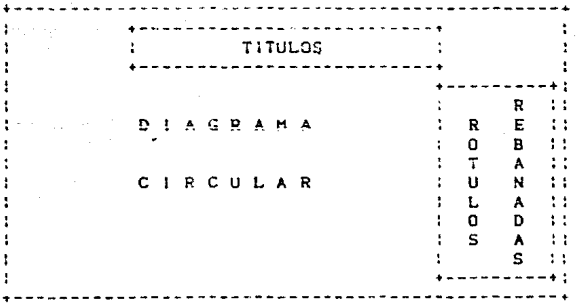


Figura 6-6

Los tipos de reportes gráficos que se pueden obtener son gráficas matemáticas, estadísticas y de negocios y la combinación de la mayoría de ellas. A continuación se enlistan todos los tipos disponibles :

1) Diagrama de Barras.

2) Diagrama de Barras Apiladas.
3) Gráficas de Lineas.
4) Histogramas.
5) Polígonos de frecuencia.
6) Gráficas X vs Y.
7) Diagramas Circulares.
8) Gráficas de Error.
9) Combinación Barras y Lineas.
10) Combinación Barras y X vs Y.
11) Combinación Lineas y X vs Y.
12) Combinación Histogramas y Polígonos de Frecuencia.
13) Combinación Histogramas y X vs Y.
14) Combinación Polígonos de Frecuencia y X vs Y.

Para obtener los reportes gráficos solo es necesario indicar las características de la hoja (Tamaño, desplazamientos y número de reportes) y para cada gráfica los grupos de datos y el número de ellos. Sin embargo, es posible variar su presentación por medio de rutinas que modifican las condiciones que tienen predefinidas cada una de las rutinas gráficas. Las modificaciones que se pueden establecer son : Nombre de los ejes, número y tipo de escalas (logarítmica y manual), uso o no de rejilla o lineas de referencia, tipo de unión de las lineas y tipo de rótulo sobre la barra.

6.2 ESTANDAR GRAFICO (EGR)

Con la finalidad de que las rutinas que generan los reportes gráficos no estuvieran ligadas directamente al paquete de primitivas gráficas que está disponible en el equipo VAX y asimismo contar con rutinas básicas que requiere el sistema y que dicho paquete no contempla, se definio y desarrolló un estándar de rutinas que cubre las necesidades antes expuestas.

El EGR no pretende ser un estándar gráfico completo y de uso general, más bién, contempla las primitivas necesarias para el sistema SER sin pretender cubrir otras aplicaciones. Sin embargo, se estudiaron distintos estándares internacionales como GKS y CORE, pero desarrollar una implementación de alguno de ellos, implica un trabajo tal vez más amplio que el del sistema SER.

La importancia de este nivel radica en que los niveles superiores no interactúan en ningún momento con el paquete de primitivas. Por lo tanto, si dicho paquete fuera cambiado o el sistema se quisiera trasladar, solo sería necesario modificar las llamadas al paquete por parte del EGR o en el peor de los casos rescribir algunas de estas rutinas.

El paquete de primitivas gráficas en el que se sustenta el sistema es PGPLOT, que fué donado a la UNAM. En la descripción técnica del EGR se mencionan las rutinas que se utilizan del paquete y las características de cada una de ellas.

En las siguientes secciones se explican las características de cada una de las rutinas del EGR, agrupadas en forma lógica para una mejor comprensión de ellas. Se indican sus parámetros y las rutinas que utilizan de PGPLOT. Finalmente se detallan las rutinas de PGPLOT utilizadas con el objetivo de que sirva de referencia para conocer las características necesarias para poder

transportar el sistema.

6.2.1 Definición Y Cierre De Un Reporte

Para poder generar una hoja de dibujo es necesario crear un archivo en donde se almacenan todas sus características. Una vez creado este archivo se indica toda la información gráfica y una vez terminada ésta, es necesario cerrar el archivo. Es importante señalar que no es posible indicar más de un archivo simultaneamente, sin embargo cerrado un archivo se puede crear otro. Al finalizar, el archivo obtenido es el que se mandará al dispositivo gráfico. A continuación se enlistan las rutinas que realizan los pasos anteriores.

- **EGR_IniciaHoja(Largo,Ancho,Dispositivo,YDesplazaIzq, XDesplazaDer,YDesplazaIzq,YDesplazaDer)**

Largo	REAL*4,Entrada
Ancho	REAL*4,Entrada
Dispositivo	CHARACTER * (*),Entrada
XDesplazaIzq	REAL*4,Entrada
XDesplazaDer	REAL*4,Entrada
YDesplazaIzq	REAL*4,Entrada
YDesplazaDer	REAL*4,Entrada

Con esta rutina se crea el archivo que contendrá la hoja de dibujo, indicándose las dimensiones del papel (**Largo** y **Ancho**), así como el nombre del archivo y el tipo de dispositivo. Los dos últimos datos se indican en el parámetro **Dispositivo** con el siguiente formato : **NombreArchivo/disp**. En donde **disp** puede tomar los siguientes valores :

pr	indica PRINTRONIX
vt	indica VT125

Los parámetros **XDesplazaIzq**, **XDesplazaDer**, **YDesplazaIzq**, **YDesplazaDer** indican el desplazamiento o margen en pulgadas, que se quiere dejar entre el tamaño de la hoja y el inicio de la región gráfica. Los valores 0.0 indicaran que el inicio de la hoja coincide con el de la region gráfica.

Asimismo, esta rutina inicializa las características de las lineas y textos a trazar. Debe ser llamada antes de cualquier rutina de EGR, y además no puede llamarse nuevamente hasta no invocar a la rutina que finaliza el reporte (EGR_FinHoja).

Rutinas de PGPLOT : PGBEGIN y PGPAPER.

- **EGR_DefineNumZonas(Divisiones)**

Divisiones	INTEGER*4,Entrada

Indica el número de reportes que contendrá la hoja. Dicho número estará dado por **Divisiones**, pudiendo valer 1, 2 ó 4. Si se proporciona un valor de 3 se asumirá 4 y para cualquier valor distinto a estos se asumirá 1.

Esta rutina no es necesario llamarla si sólo se requiere un reporte en la hoja.

- **EGR_AvanzaZona**

Define el área en donde se dibujarán las gráficas de acuerdo al número de divisiones indicadas. Internamente se controla en qué zona se realizarán las gráficas de acuerdo al número de veces que se ha llamado a esta rutina. Aún cuando sólo exista una división es necesario llamar a esta rutina antes de comenzar a dibujar.

- **EGR_FinHoja**

Es necesario llamarla cada que se quiera terminar de invocar rutinas gráficas para ese reporte. Una vez realizada, es posible crear otro reporte o mandar a imprimir el recién obtenido.

Rutinas de PGPLOT : PGEND.

6.2.2 Areas Y Ventanas De Dibujo

En esta sección se explicarán las rutinas que definen las áreas en las que se desea dibujar (VIEWPORT). Asimismo, las rutinas que definen las coordenadas de dichas áreas (WINDOW). Por último, están las rutinas que permiten obtener los valores actuales de las áreas y sus coordenadas.

- **EGR_EstableceArea(XMin,XMax,YMin,YMax)**

XMin	REAL*4,Entrada
XMax	REAL*4,Entrada
YMin	REAL*4,Entrada
YMax	REAL*4,Entrada

Define el área en donde se dibujará. Las coordenadas del área son relativas a la zona que se tiene activa actualmente. Los valores de los parámetros deberán estar en el rango de 0.0 a 1.0.

Rutinas de PGPLOT : PGVPORT.

- **EGR_EstableceAreaReg(XMin,XMax,YMin,YMax)**

XMin	REAL*4,Entrada
XMax	REAL*4,Entrada
YMin	REAL*4,Entrada

YMax REAL*4,Entrada

Define el área en donde se desea dibujar, relativa a la región gráfica que se tiene definida actualmente. La region gráfica es el área que esta disponible una vez que se han realizado los desplazamientos en la hoja.

Rutinas de PGPLOT : PGVPORT.

o **EGR_EstableceAreaHoja(XMin,XMax,YMin,YMax)**

YMin	REAL*4,Entrada
XMax	REAL*4,Entrada
YMin	REAL*4,Entrada
YMax	REAL*4,Entrada

Define el área, relativa al tamaño de la hoja,

en donde se dibujará.

Rutinas de PGPLOT : PGVPORT.

o **EGR_ObtenArea(XMin,XMax,YMin,YMax)**

XMin	REAL*4,Salida
XMax	REAL*4,Salida
YMin	REAL*4,Salida
YMax	REAL*4,Salida

Obtiene el area actual relativa a la zona que se tiene definida.

o **EGR_ObtenAreaReg(XMin,XMax,YMin,YMax)**

XMin	REAL*4,Salida
XMax	REAL*4,Salida
YMin	REAL*4,Salida
YMax	REAL*4,Salida

Obtiene el área actual relativa a la región gráfica disponible.

o **EGR_ObtenAreaHoja(XMin,XMax,YMin,YMax)**

XMin	REAL*4,Salida
XMax	REAL*4,Salida
YMin	REAL*4,Salida
YMax	REAL*4,Salida

Obtiene el área actual relativa a la hoja.

- **EGR_DefineVentana(XMin,XMax,YMin,YMax)**

XMin	REAL*4,Entrada
XMax	REAL*4,Entrada
YMin	REAL*4,Entrada
YMax	REAL*4,Entrada

Establece las coordenadas que tendrá el área que se tiene actualmente definida. Estas coordenadas son las que se tomarán para el trazo o referencia de cualquier gráfico.

Rutinas de PGPLOT : PGWINDOW.

- **EGR_ObtenVentana(XMin,XMax,YMin,YMax)**

XMin	REAL*4,Salida
XMax	REAL*4,Salida
YMin	REAL*4,Salida
YMax	REAL*4,Salida

Obtiene las coordenadas de la ventana que está activa.

6.2.3 Trazo De Líneas

- **EGR_DefineLinea(Ancho,Estilo)**

Ancho	INTEGER*4,Entrada
Estilo	INTEGER*4,Entrada

Define las características de las líneas a dibujar. **Ancho** indica el grosor que tendrá la línea en un rango de 1 (línea normal) a 21. **Estilo** indica el tipo de línea a utilizar, pudiendo ser continua (1), discontinua(2), punteada (4) y la combinación de ellas (3 y 5).

Durante la obtención de una hoja de dibujo se puede llamar tantas veces como se desee. Las características definidas permanecerán hasta que no sea llamada esta rútina nuevamente.

Rutinas de PGPLOT : GRSETLS y GRSETLW.

- **EGR_MueveA(X,Y)**

X	REAL*4,Entrada
Y	REAL*4,Entrada

Mueve la pluma de dibujo a las coordenadas X,Y sin realizar ningún trazo.

Rutinas de PGPLOT : PGMOVE.

o **EGR_TrazaLinea(X,Y)**

X	REAL*4,Entrada
Y	REAL*4,Entrada

Traza una linea recta desde la posición actual de la pluma (el último movimiento de la pluma o el último punto trazado) hasta las coordenadas X,Y. Las características de la línea estarán dadas por la última definición de línea establecida.

Rutinas de PGPLOT : PGDRAW.

6.2.4 Trazo De Textos

o **EGR_DefCarTexto(AnguloTex,OrientaLet,TamanoLet,Estilo)**

AnguloTex	REAL*4,Entrada
OrientaLet	REAL*4,Entrada
TamanoLet	REAL*4,Entrada
Estilo	CHARACTER * (*),Entrada

Por medio de esta rutina se definen las características del texto que se quiere utilizar.

Es muy importante señalar que aunque el paquete PGPLOT tiene la rutina de trazo de caracteres con variación de ángulos, no proporciona orientación, razón por la que fué necesario hacer una nueva rutina que realizara estas dos funciones.

La diferencia entre ángulo y orientación es que el primero es el ángulo con que se dibujará el texto **completo** y la orientación indica la rotación que sufrirá cada letra del texto. Estas dos características son independientes.

El tamaño de la letra se obtiene considerando que el valor de 1, indica 1/40 del tamaño de la zona activa.

Los estilos del texto ("FONTS") son los que PGPLOT proporciona, siendo estos: normal, romano, manuscrito e itálico.

Las características definidas permanecerán para el trazado de todos los textos hasta que no sean definidas otras.

Rutinas de PGPLOT : PGSETC

- **EGR_CalcDimLet(IncrementoXLetra, IncrementoYLetra)**

IncrementoXLetra	REAL*4,Salida
IncrementoYLetra	REAL*4,Salida

Regresa el espacio que ocupará cada letra de acuerdo a las coordenadas actuales tanto en la dirección en X como en Y. Esta rutina es muy útil para calcular el espacio que ocupará un texto y determinar si excede o no el espacio disponible para él. Debe de llamarse antes la rutina **EGR_DefCarTexto**

- **EGR_EscTexto(X,Y,JustifTexto,JustifAncho,JustifAlto,Texto)**

X	REAL*4,Entrada
Y	REAL*4,Entrada
JustifTexto	REAL*4,Entrada
JustifAncho	REAL*4,Entrada
JustifAlto	REAL*4,Entrada
Texto	CHARACTER * (*),Entrada

Permite dibujar un texto de acuerdo a las características definidas con los parámetros y con la rutina EGR_DefCarTexto. El texto se colocará a partir de las coordenadas (X,Y) especificadas y con una posible justificación con respecto a ese punto, de acuerdo al parámetro **JustifTexto**.

Para modificar el punto de inicio de la letra, que normalmente es la esquina inferior izquierda, se usan los parámetros **JustifAlto** y **JustifAncho**.

Se recomienda que las tres justificaciones esten en el rango de 0 a 1, pudiendo ser de - 1 también.

PGPLOT permite introducir en un texto todos los caracteres del código ASCII y además símbolos especiales y características especiales (Subindices, superindices y letras griegas) con solo indicar una antidiagonal (\) dentro del texto y la característica y simbolo deseado. Para mayor información consulte el manual del usuario SER y PGPLOT.

Es muy importante señalar, que el desarrollo de esta rutina tuvo muchas dificultades debida al desconocimiento de la lógica interna de la rutina de texto de PGPLOT y las caracteristicas adicionales que se integraron (orientación y justificaciones). De hecho, sólo se utilizó la rutina de PGPLOT para trazar los caracteres individuales con los tipos de letras que proporciona y no trazar cadenas completas.

Rutinas de PGPLOT : PGTEXT

- **EGR_Simbolo(X,Y,CodigoSimbolo)**

X	REAL*4,Entrada
Y	REAL*4,Entrada
CodigoSimbolo	INTEGER*4,Entrada

Dibuja un símbolo especial en las coordenadas **X,Y**. Existen cinco distintos símbolos a utilizar, indicados por **CodigoSimbolo**

Rutinas de PGPLOT : PGPOINT.

6.2.5 Ejes De Referencia Y Unión De Puntos

- **EGR_DibujaEje(Segmento,Cotas,IncCotas,Marcas,Submarcas,Logaritmica)**

Segmento	CHARACTER * (*),Entrada
Cotas	INTEGER*4,Entrada
IncCotas	REAL*4,Entrada
Marcas	INTEGER*4,Entrada
SubMarcas	INTEGER*4,Entrada
Logaritmica	INTEGER*4,Entrada

Dibuja un segmento de recta acotado que se usa como eje de referéncia, tomando las coordenadas que estén definidas en la ventana actual.

Segmento indica el eje que se desea dibujar (XI, XS, YI y YD) y los demás parámetros indican caracteristicas del eje como: Descripción del valor para cada marca (**Cotas**), el incremento deseado entre dichas cotas (**IncCotas**), la inclusión de marcas y submarcas a lo largo del eje (**Marcas y Submarcas**) y por último el tipo de escala (**Logaritmica**).

Rutinas de PGPLOT : PGBOX

- **EGR_UnePuntos(X,Y,IndiceInicial,IndiceFinal,TipoUnion,TipoLineaSimbolo)**

X	REAL*4,Entrada
Y	REAL*4,Entrada
IndiceInicial	INTEGER*4,Entrada
IndiceFinal	INTEGER*4,Entrada
TipoUnion	INTEGER*4,Entrada
TipoLineaSimbolo	INTEGER*4,Entrada

Dibuja parejas de puntos (x,y) unidas entre ellas de distintas formas. Los datos deben venir agrupados en los arreglos **X,Y**. Es necesario indicar la posición del primer elemento y el del último que se desea dibujar (**IndiceInicial** e **IndiceFinal**) sin que necesariamente coincidan con el inicio y la longitud del arreglo.

Existen distintas formas de unir los puntos debido a las diversas aplicaciones que se pueden tener. Las uniones disponibles son: Unión con lineas rectas(3), sólo símbolo en el punto sin unión (2), unión con lineas rectas y símbolo en el punto(1) y por último símbolos en el punto con una linea al eje X para y=0 (4).

Los tipos de símbolos y lineas son los mismos que se utilizan en la definición de tipo de linea y en símbolo.

- **EGR_ObtenTamanoEjes(TamanoEjeX,TamanoEjeY)**

TamanoEjeX	REAL*4,Salida
TamanoEjeY	REAL*4,Salida

Obtiene el espacio que ocupan los ejes (área definida actualmente) en unidades físicas, es decir, en pulgadas.

6.2.6 Relleno De Arcos Y Rectángulos

Para todas las rutinas que utilizan barras (diagramas de barras e histogramas) era necesario una rutina que permitiera el dibujo de un rectángulo y éste fuera rellenado por algún patrón de dibujo. Asimismo, para los diagramas circulares se necesitó una rutina que hiciera lo mismo pero para segmentos de circulos. Por este motivo, se desarrollaron las rutinas que realizan lo mencionado y a continuación se describen.

Debido a la complejidad y requerimientos que tienen estas rutinas, se desarrollaron algunas rutinas auxiliares que tambien se explican.

- **EGR_RellenaArco(X,Y,AnginIcial,AngFinal,Radio,Patron)**

X	REAL*4,Entrada
Y	REAL*4,Entrada
AnginIcial	REAL*4,Entrada
AngFinal	REAL*4,Entrada
Radio	REAL*4,Entrada
Patron	INTEGER*4,Entrada

Dibuja un arco en las coordenadas **X,Y** de acuerdo a la especificación que indique **Radio** y que comenzará en el **AnguloInicial** hasta el **AnguloFinal**. Los ángulos son en grados considerando que el eje X corresponde a la horizontal del papel y el eje Y en forma paralela a las perforaciones.

El relleno se hará de acuerdo al parámetro **Patron**, permitiendose 20 distintos patrones de relleno.

Es importante resaltar la complejidad que se tuvo en el desarrollo de esta rutina debido entre otras causas, a que el patrón de relleno debe permanecer constante sin importar las coordenadas que se tengan

definidas (ventana). Una rutina que auxilió a resolver este problema fué:

```
IEGR_ObtenFactores(FactorX,FactorY)
```

Los parámetros regresan un factor por el que es necesario multiplicar todas las coordenadas que se vayan a dibujar para garantizar la independencia entre el patrón de relleno y la ventana.

Debido a que los algoritmos se diseñaron utilizando coordenadas polares y los trazos gráficos se realizan en coordenadas cartesianas fué necesario desarrollar dos rutinas que permitieran conversiones entre dichas coordenadas. Las rutinas son:

```
IEGR_PolarRec(Radio,Angulo,X,Y)
IEGR_RecPolar(X,Y,Radio,Angulo)
```

Otra rutina auxiliar es:

```
IEGR_TrazaintArc(Radio,X,Y,AnguloInicial,M1,Dif1,
                 AnguloFinal,M2,Dif2,M0,X0,Y0)
```

Con esta rutina se determinan las intersecciones de acuerdo a las lineas del arco (M1, Dif1,M2, Dif2) y la linea de relleno (M0, X0 y Y0).

o **EGR_RellenaRectangulo(X,Y,AnchoRec,AltoRec,Patron)**

X	REAL*4,Entrada
Y	REAL*4,Entrada
AnchoRec	REAL*4,Entrada
AltoRec	REAL*4,Entrada
Patron	INTEGER*4,Entrada

Dibuja un rectángulo con inicio en las coordenadas X,Y, con la altura y ancho indicados en los parametros **AltoRec** y **AnchoRec**. Con **Patron** se indica el tipo de relleno que se utilizará, existiendo disponibles 20 patrones.

Al igual que en **EGR_RellenaArco** existieron problemas en cuanto a los sistemas de coordenadas y a la obtención de factores para mantener la independencia del patron de relleno y la ventana definida. Por estos motivos también se utilizaron las siguientes rutinas auxiliares:

```
IEGR_ObtenFactores(FactorX,FactorY)
IEGR_PolarRec(Radio,Angulo,X,Y)
IEGR_RecPolar(X,Y,Radio,Angulo)
```

Para el trazado de las lineas de relleno en donde existe intersección con el rectangulo se utiliza la siguiente rutina auxiliar:

IEGR_TrazaIntRec(X1,Y1,X2,Y2,M0,B0)

En la cual se indican las cuatro rectas que definen al rectángulo (X1, Y1, X2, Y2) así como la pendiente y cruce con el origen de la recta de relleno (M0 y B0).

6.2.7 Primitivas Utilizadas

En esta sección se enlista las rutinas del paquete PGPLOT utilizadas por el estandar gráfico. Si en un momento se desea cambiar de paquete de primitivas solo será necesario que el nuevo paquete cuente con rutinas con las caracteristicas que a continuación se mencionan.

Los parámetros que tengan un valor constante, son aquellos que no fueron utilizados por el EGR o porque el paquete PGPLOT los utiliza como "Dummy".

Para una descripción completa de PGPLOT consulte el manual de referencia que esta disponible en el CECAFI [15].

- **PGBEGIN(0,Dispositivo,1,1)**

  Abre el archivo de gráficas, de acuerdo al dispositvo indicado. En el dispositivo se indica el nombre de la gráfica así como el tipo de impresora o terminal a utilizar.

- **PGPAPER(Ancho,Largo/Ancho,0)**

  Permite definir las dimensiones de la hoja, en donde **Ancho** corresponde a la dimensión paralela a las perforaciones del papel y **Largo** se refiere a la horizontal de la hoja.

- **PGCLOSE**

  Cierra el archivo de gráficas generado con un PGBEGIN.

- **PGVPORT(XMin,XMax,YMin,YMax)**

  Define el área en donde se dibujarán todos los trazos que se dibujen (VIEWPORT). Los trazos que esten fuera de esta área no serán dibujados. Los limites de esta área (XMin,XMax,YMin,YMax), son expresados en coordenadas absolutas normalizadas con respecto al tamaño total de la hoja.

- **PGWINDOW(XMin,XMax,YMin,YMax)**

Establece las coordenadas del mundo que tendrá el área que esté definida actualmente (WINDOW).

- **GRSETLS(TipoLinea)**

  Indica el tipo de linea que tendrán las lineas que se dibujen. Permite linea continua, discontinua, punteada y la combinación de ellas.

- **GRSETLW(AnchoLinea)**

  Define el ancho que tendrán las lineas que sean dibujadas, en un rango de 1 a 21.

- **PGMOVE(X,Y)**

  Mueve la pluma de dibujo al punto X,Y de acuerdo a coordenadas del mundo. No dibuja linea.

- **PGDRAW(X,Y)**

  Dibuja una linea de la posición actual de la pluma de dibujo al punto X.Y. Estableciendose este punto como nueva posición de la pluma.

- **PGPOINT(1,X,Y,Simbolo)**

  Dibuja un símbolo en la coordenada X,Y. Simbolo indica el código correspondiente al símbolo deseado.

- **PGSETC(TamanoLetra)**

  Define el tamaño de letra que se utilizará para todos los trazos de textos. El valor de 1.0 corresponde a 1/40 del tamaño de hoja definido.

- **PGTEXT(X,Y,AnguloTexto,0.0,Texto)**

  Traza un texto con inicio en el punto X,Y de acuerdo al ángulo de texto especificado.

- **PGBOX(OpcionesX,IncCotasX,SubintX,OpcionesY,
  IncCotasY,SubintY)**

  Dibuja un recuadro en el área que actualmente esté definida de acuerdo a las caracteristicas que se indican en OpcionesX y OpcionesY. Este recuadro se utiliza como marco de referencia para la gráfica, es decir, se utiliza para indicar ejes o un trazado de rejilla. Las opciones se indican en una cadena de caracteres, en donde cada caracter indica una opción.

  Asimismo, es posible indicar el incremento que debe existir entre cada cota y el número de subintervalos que se quiere tenga el eje.

6.3 RUTINAS DE PROPÓSITO ESPECÍFICO (RPE)

En este nivel, se encuentran las rutinas que realizan una secuencia de pasos comunes a todos o algunos de los reportes gráficos. A continuación se explican cada una de las rutinas siguiendo el mismo criterio que se usó en EGR.

6.3.1 Medio Ambiente

- **RPE_MedioAmbiente(TipoGraf,NombresEjes,Titulos,GruposYd,RotulosGrupos, NumRotulos,Acotaciones,NumElemGrupo,AreaGraf)**

TipoGraf	INTEGER4,Entrada
NombresEjes	CHARACTER (*) *(*),Entrada
Titulos	CHARACTER (*) *(*),Entrada
GruposYD	CHARACTER *(*),Entrada
RotulosGrupos	CHARACTER (*) *(*),Entrada
NumRotulos	INTEGER4,Entrada
Acotaciones	CHARACTER (*) *(*),Entrada
NumElemGrupo	INTEGER4,Entrada
AreaGraf	REAL*4 (4),Salida

Determina las áreas que estarán activas para el reporte de acuerdo a las características que haya definido el usuario.

Para un mayor aprovechamiento de la zona gráfica no se utilizaron áreas fijas sino dinámicas, es decir, áreas que no se vayan a utilizar (por ejemplo que no existen títulos) se ocupan por otras que sí estén activas.

En la sección correspondiente a Presentación de reportes se explican cada una de las zonas que existen.

Para determinar si una zona se activa o no, se verifica que los parámetros que las definen tengan algún valor. Si una zona está activa se define el área que va a ocupar y se llama a la rutina que realiza el trazo de dicha zona, indicándole las coordenadas del área y los parámetros que deben ser trazados.

Como parámetro de salida, regresa el área disponible para dibujar la gráfica correspondiente.

Debido a sus características, este módulo es invocado por todas las rutinas que realizan los reportes gráficos desde el nivel SER. Por lo que existen algunas condiciones particulares para cada tipo de gráfica, por ejemplo, para Diagramas Circulares no existen las mismas zonas que para las otras gráficas.

- **RPE_EscNomEjes(Segmento,XMin,XMax,YMin,YMax,Rotulo)**

Segmento	CHARACTER * (*),Entrada
XMin	REAL*4,Entrada
XMax	REAL*4,Entrada
YMin	REAL*4,Entrada
YMax	REAL*4,Entrada
Rotulo	CHARACTER * (*),Entrada

Dibuja el texto correspondiente al eje que se indique (X, YIzq o YDer) en el área dada por **XMin**, **XMax**, **YMin** y **YMax**

- **RPE_EscTitulos(Titulos)**

Titulos	CHARACTER (*) * (*)

Dibuja los textos correspondientes a los títulos de la gráfica. Pueden ser de 1 a 3 líneas de títulos los que ocupe el área definida por **XMin**, **XMax**, **YMin** y **YMax** dependiendo del número de elementos que tenga el arreglo **Titulos**.

- **RPE_EscAcot(XMin,XMax,YMin,YMax,Acotaciones,NumAcot,EspAcot)**

XMin	REAL*4,Entrada
XMax	REAL*4,Entrada
YMin	REAL*4,Entrada
YMax	REAL*4,Entrada
Acotaciones	CHARACTER (*) * (*)
NumAcot	INTEGER*4,Entrada
EspAcot	REAL*4,Salida

Coloca las acotaciones alfanuméricas que corresponden a cada elemento en las gráficas de Barras, Líneas, Apiladas o Error.

Debido a que el espacio disponible para cada acotación se reduce de acuerdo al número de elementos que tenga la gráfica, la rutina determina si dichas acotaciones caben en forma horizontal o es necesario colocarlas en forma vertical para no truncar caracteres. Sin embargo, las acotaciones que se tracen verticalmente podrán ocupar como máximo el 50 % del área indicada por **XMin**, **XMax**, **YMin** y **YMax**, truncándose las acotaciones que ocupen más de ese porcentaje para que el área disponible a la gráfica sea al menos igual al área ocupada por las acotaciones.

6.3.2 Dibujo De Identificadores De Grupos

- **RPE_DibujaIdentifDC(Rotulos,NumRebanadas,TipoRelleno,**
  XMin,XMax,YMin,YMax)

Rotulos	CHARACTER (*) * (*),Entrada
NumRebanadas	INTEGER*4,Entrada
TipoRelleno	INTEGER*4 (*),Entrada
XMin	REAL*4,Entrada
XMax	REAL*4,Entrada
YMin	REAL*4,Entrada
YMax	REAL*4,Entrada

Dibuja en el área disponible (**XMin, XMax, YMin, YMax**) los rótulos que identifican cada una de las rebanadas de un Diagrama Circular.

A cada rótulo se le incluye un pequeño rectangulo relleno con el mismo patrón de la rebanada que le corresponde.

o **RPE_DibujaIdentif(TipoIdentif,TipoLinSimRelleno,NombreGrupos, Asociación,NumGrupo,XMin,XMax,YMin,YMax)**

TipoIdentif	INTEGER*4 (*),Entrada
TipoLinSimRelleno	INTEGER*4 (*),Entrada
NombreGrupos	CHARACTER (*) * (*),Entrada
Asociacion	CHARACTER (*) * (*),Entrada
NumGrupo	INTEGER*4 (*),Entrada
XMin	REAL*4,Entrada
XMax	REAL*4,Entrada
YMin	REAL*4,Entrada
YMax	REAL*4,Entrada

Dibuja en el área disponible (**XMin, XMax, YMin, YMax**) los rotulos que identifican cada uno de los grupos de datos de cualquier gráfica, con excepción de Diagramas Circulares.

A cada rótulo se le asocia un simbolo, un tipo de linea, la combinación de ambos o un pequeño rectángulo relleno. Por ejemplo, para un Diagrama de Barras se utilizará un rectangulo para identificar los grupos, asi como tal vez se use una linea con un simbolo para un polígono de frecuencia.

En los arreglos **TipoIdentif** y **TipoLinSimRelleno** se indican el tipo de asociación que se hará a cada rótulo (linea, símbolo o rectangulo) y la característica que tendrá (linea discontinua, un rombo como símbolo, etcetera). Asimismo, **Asociación** indica si el grupo está escalado con respecto al eje derecho o izquierdo para que el tipo de letra se modifique de acuerdo a esta relación (Normal eje Y izquierdo, Itálica eje Y derecho). Modifica el tamaño de las letras para que escriba la longitud del texto al mayor tamaño posible.

6.3.3 Marco De Referencia

Para la localización más exacta de los puntos de la gráfica se proporcionan dos herramientas de auxilio, una es colocar una rejilla dentro de los ejes y otra es el trazo de lineas de referencia paralelas al eje X y en algunas gráficas paralelas al eje Y, en valores definidos por el usuario. A continuación se explican las rutinas que realizan los trazos.

- **RPE_RejillaVertical(TipoLinea)**

TipoLinea	INTEGER4,Entrada

  Dibuja lineas en forma paralela al eje Y, en cada acotación que exista en el eje X. La rejilla se puede trazar con alguno de los textos tres tipos de linea disponibles: linea continua (1), discontinua (2) o punteada(3).

- **RPE_RejillaHorizontal(TipoLinea)**

TipoLinea	INTEGER4,Entrada

  Dibuja lineas en forma paralela al eje X, en cada acotación que exista en el eje Y. La rejilla se puede trazar con alguno de los tres tipos de linea disponibles: linea continua (1), discontinua (2) o punteada(3).

- **RPE_TrazaLineaRefHor(LineasRef,NumLineas,Log)**

LineasRef	REAL*4(*),Entrada
NumLineas	INTEGER*4(*),Entrada
Log	INTEGER*4(*),Entrada

  Traza lineas paralelas al eje X, en los valores en Y que indique el usuario (**LineasRef**) dentro del área que esté actualmente definida. Si la escala definida es logaritmica es necesario indicarlo con **Log** igual a 1.

- **RPE_TrazaLineaRefVert(LineasRef,NumLineas,Log)**

LineasRef	REAL*4(*),Entrada
NumLineas	INTEGER*4(*),Entrada
Log	INTEGER*4(*),Entrada

  Traza lineas paralelas al eje Y, en los valores en X que indique el usuario (**LineasRef**) dentro del área que esté actualmente definida. Si la escala definida es logaritmica es necesario indicarlo con **Log** igual a 1.

6.3.4 Rótulos

- **RPE_CalcEspRotBarras(ValorMenMay,AnchoBarra,Porcent, Negativas,VentX,VentY)**

ValorMenMay	REAL*4 (2),Entrada
AnchoBarra	REAL*4,Entrada
Porcent	LOGICAL*1,Entrada
Negativas	INTEGER*4,Entrada
VentX	REAL*4 (2),Entrada
VentY	REAL*4 (2),Entrada/Salida

Determina el espacio que ocuparán los rótulos que se colocarán sobre las barras. Este espacio se usará como parámetro en la rutina RPE_ColocaRotBarra que dibuja dichos rotulos.

Los valores menor y mayor de las barras a dibujar (**ValorMenMay**) permiten calcular el número de dígitos que ocupará el rótulo incluyendose el signo de porcentaje para el caso que así se solicite (**Porcent** igual a verdadero). Si el número de dígitos es mayor al formato establecido, el valor se expresa en forma exponencial. Para realizar esta paso se auxilia de la rutina:

```
IRPE_ObtenCifras(Potencia)
```

Como parámetros de salida regresa las coordenadas que deberá tener la ventana en Y para que los rótulos puedan ser dibujados. Por esta razón es necesario indicar si existen barras solo positivas, solo negativas o la combinación de ambas para determinar si es necesario calcular espacio para uno o dos rótulos. La ventana en X no requiere ser modificada.

Asimismo, determina si el rotulo cabe en forma horizontal sobre la barra o es necesario colocarlos en forma vertical por lo que la ventana también será afectada por esta razon.

- **RPE_ColocaRotBarra(X,Y,AnchoBar,AlturaBar,Valor,Porcent)**

X	REAL*4,Entrada
Y	REAL*4,Entrada
AnchoBar	REAL*4,Entrada
AlturaBar	REAL*4,Entrada
Valor	REAL*4,Entrada
Porcent	LOGICAL*1,Entrada

Dibuja el valor indicado en la barra que está definida por **X, Y** (origen de la barra), **AnchoBar** y **AlturaBar**. Previamente se debe haber llamado a la rutina RPE_CalcEspRotBarras para determinar la ventana necesaria para que los rótulos puedan ser dibujados.

Para la obtención del formato adecuado del rótulo que corresponde al valor proporcionado ya sea en forma exponencial, con porcentaje o no, así como con el signo adecuado, se utiliza la siguiente rutina auxiliar:

```
IRPE_EscribeFormato(Numero,Potencia,Porcent,NumCar)
```

- **RPE_CalcEspRotRebanadas(MayorReb,Porcent,VentX,VentY)**

MayorReb	REAL*4,Entrada
Porcent	LOGICAL*1,Entrada
VentX	REAL*4 (2),Entrada/Salida
VentY	REAL*4 (2),Entrada/Salida

Determina el espacio que ocuparán los rótulos que van asociados a cada rebanada ya sea en forma porcentual o absoluta.

Solo es necesario indicar el valor máximo de todos los elementos del Diagrama Circular. Los parámetros **VentX** y **VentY** indican inicialmente la ventana que está definida y también son utilizados como salida para indicar la ventana que es necesaria definir para que puedan ser dibujados los rótulos.

Si el número de cifras del valor excede al formato establecido, se convierte el valor a notación exponencial. Para determinar este número se utiliza la rutina auxiliar :

```
IRPE_ObtenCifras(Potencia)
```

- **RPE_DibujaRebPorcent(CentroX,CentroY,AnguloInicial,AnguloFinal, Valor,Porcent,Patron)**

CentroX	REAL*4,Entrada
CentroY	REAL*4,Entrada
AnguloInicial	REAL*4,Entrada
AnguloFinal	REAL*4,Entrada
Valor	REAL*4,Entrada
Porcent	LOGICAL*1,Entrada
Patron	INTEGER*4,Entrada

Dibuja el arco especificado con **CentroX**, **CentroY**, **AnguloInicial**, **AnguloFinal**, de acuerdo con el patrón de relleno especificado, dibujando asimismo el valor que se indique.

Previamente, debe ser llamada la rutina **RPE_CalcEspRotRebanadas** para determinar la ventana que deberá ser definida para que los rótulos puedan ser dibujados.

Para la obtención del formato adecuado del rótulo que corresponde al valor proporcionado ya sea en forma exponencial o con porcentaje o no, se utiliza la siguiente rutina auxiliar:

IRPE_EscribeFormato(Numero,Potencia,Porcent,NumCar)

6.4 RUTINAS GRAFICAS DEL SISTEMA EMISOR DE REPORTES (SER)

Las rutinas que llama el usuario para generar los reportes gráficos se agrupan en este nivel. Para generar un reporte sólo es necesario llamar a tres rutinas, sin embargo existen otras que permiten modificar las condiciones de presentación que se tienen asignadas inicialmente. A continuación se presenta la secuencia de llamadas a rutinas del SER que debe realizarse y la descripción de todas ellas:

- Definición de la hoja

  Para definir las caracteristicas de la hoja de dibuja e iniciar su trazo, es necesario invocar a la siguiente rutina:

  SER_DefineHoja(Largo,Ancho,NumDiv,Rotulos,Disp)

  Esta llamada deberá realizarse antes de cualquier otra llamada al SER o después de la rutina **SER_TerminaHoja**.

  Opcionalmente se podran definir márgenes en la hoja de dibujo con la rutina:

  SER_DesplazaGrafica(XInf,XSup,YIzq,YDer)

  Deberá ser llamada **antes** que la rutina **SER_DefineHoja**. Si no se desean márgenes no es necesario invocarla.

- Selección de reportes y su presentacion

  Para cada reporte que contenga la hoja (1, 2 ó 4), opcionalmente se llamarán las rutinas de modificación del reporte y necesariamente a la rutina que define ese reporte. Es decir, cada vez que se termina un reporte las condiciones de presentación para él no serán mantenidas para el siguiente, sino que se tendrán las condiciones que inicialmente presenta SER.

  Las rutinas de modificación son:

  SER_AsignaYD(GruposDer)
  SER_LineasRefX(LineasRef,NumLinRef)
  SER_LineasRefY(LineasRef,NumLinRef)
  SER_Rejilla(TipoRejilla)
  SER_RotulosEje(Eje,Rotulo)

```
SER_Logarit(Eje)
SER_EscalaManual(Eje,ValInicial,ValFinal)
```

Para la selección de la grafica se tienen las siguientes rutinas (no se incluyen la lista de parámetros, para una referencia total consultar la sección de GRAFICAS):

```
SER_Barras
SER_BarrasApiladas
SER_Lineas
SER_Histograma
SER_PoligonoFrec
SER_XvsY
SER_DiagramaCircular
SER_Error
SER_BarrasLinea
SER_BarXvsY
SER_LinXvsY
SER_HistogramaPoligono
SER_HistogramaXvsY
SER_PoligonoXvsY
```

Es muy importante recalcar que para el caso de más de un reporte en la hoja de dibujo será necesario llamar nuevamente a las rutinas de modificación y a la de selección de la gráfica. Al final de este capitulo se muestran algunos ejemplos del llamado de las rutinas SER.

- Terminación de la hoja de dibujo

Para cerrar el archivo en donde se encuentran los reportes es necesario llamar a la rutina :

```
SER_TerminaHoja
```

6.4.1 Definición De Un Reporte

- **SER_DefineHoja(Largo,Ancho,NumDivHoja,Rotulos,Dispos)**

Largo	REAL*4,Entrada
Ancho	REAL*4,Entrada
NumDivHoja	INTEGER*4,Entrada
Rotulos	CHARACTER (*) * (*),Entrada
Dispos	CHARACTER * (*),Entrada

Define e inicia la hoja en donde se harán los reportes gráficos (máximo 4). Aquí se determina el tamaño que debe tener la hoja de dibujo en pulgadas, sin que necesariamente tenga que ser igual al tamaño de la hoja de papel que se encuentre en la impresora, sin

embargo, no deberá ser mayor.

El número de reportes que puede contener una hoja se indica con el parámetro **NumDivHoja**, el cual puede valer 1, 2 ó 4. Los reportes que existan en una hoja pueden variar en tipo y en presentación.

En el arreglo **Rotulos** se colocarán los encabezados que se deseen colocar en el recuadro de identificación general para la hoja. Los rótulos particuales a cada reporte, se indicarán más adelante.

Por último, es necesario indicar el dispositivo en donde se va a imprimir la hoja de dibujo. Actualmente sólo está disponible Printronix. La sintaxis que se debe seguir es : **NombreArch/Pr.** En donde NombreArch sigue las condiciones de VMS incluyendo tipo.

- **SER_DesplazaGrafica(XInf,XSup,YIzq,YDer)**

```
XInf      REAL*4,Entrada
XSup      REAL*4,Entrada
YInf      REAL*4,Entrada
YSup      REAL*4,Entrada
```

Determina los márgenes que existirán entre el tamaño de la hoja y el recuadro de identificación de los reportes. Esta rutina no será necesario llamarla si no se desean márgenes.

Para identificar el tamaño de la hoja se trazará un recuadro punteado con líneas discontinuas para el caso en que la hoja de dibujo no coincida con el tamaño de la hoja de papel.

6.4.2 Modificadores A La Presentación

Este bloque de rutinas se encargan de modificar las características de presentación que inicialmente se establecen para el trazo de los reportes. A continuación se indican las condiciones iniciales existentes:

- Todos los grupos son asociados a la escala Y izquierda.
- No se trazarán líneas de referencia ni rejilla.
- La escala es automática y normal (no logarítmica).
- Los ejes no tendrán rótulos asociados.

Es muy importante destacar que:

- Para cada nuevo reporte se inicializan las condiciones de presentación sin tomarse en cuenta las definidas para el reporte anterior.

- Las rutinas modificadoras deberán ser llamadas antes de invocar a la rutina que genera el reporte.

- Las características que se definan para reportes que no las contemplen simplemente se ignorarán.

- **SER_AsignaYD(GruposDer)**

```
GruposDer      CHARACTER * (*),Entrada
```

Se establecen los grupos que iran asociados a la escala derecha. Se indica con el siguiente formato:

```
NumGrupo1[,NumGrupo2...]
```

En donde NumGrupo1 indica el número de grupo que estará asociado al eje Y derecho. Los Grupos que no se indiquen en esta llamada, serán asociados al eje Y Izquierdo. Por ejemplo:

```
CALL SER_AsignaYD('1,4')
```

Los grupos 1 y 4 estarán asociados al eje Y Derecho y los grupos 2, 3 y 5(si existe) se asociarán al eje Y Izquierdo.

- **SER_LineasRefX(LineasRef,NumLinRef)**

```
LineasRef      REAL*4 (*),Entrada
NumLinRef      INTEGER*4,Entrada
```

Se indican los valores (en **LineasRef**) con respecto al eje X, en los que se trazarán lineas de referencia paralelas al eje Y. Para reportes en donde las acotaciones no sean numéricas, esta opcion no será considerada (Diagramas de Barras, Diagramas de Lineas y Error).

Si algún valor es menor o mayor al rango establecido en la escala, no será trazada linea para ese valor.

Si además se define rejilla, las lineas de referencia no serán trazadas.

- **SER_LineasRefY(LineasRef,NumLinRef)**

```
LineasRef      REAL*4 (*),Entrada
NumLinRef      INTEGER*4,Entrada
```

Se indican los valores (en **LineasRef**) con respecto al eje Y, en los que se trazarán lineas de referencia paralelas al eje X. En el caso de que existan dos escalas de referencia, se considerarán los valores con respecto a la escala Y izquierda.

Si algún valor es menor o mayor al rango establecido en la escala, no será trazada linea para ese valor.

Si además se define rejilla, las líneas de referencia no serán trazadas.

- **SER_Rejilla(TipoRejilla)**

```
TipoRejilla      INTEGER*4,Entrada
```

Indica que debe existir rejilla y además selecciona el tipo de línea que se utilizará. La rejilla es un conjunto de líneas de referencia que inician de las marcas establecidas en el eje. Para reportes en donde las acotaciones del eje X no sean numéricas (Diagramas de Barras, Diagramas de Líneas, Error), se trazará rejilla solo en el eje Y.

Los tipos de línea son: Continua (1), discontinua (2) y punteada (3).

Si además se definen líneas de referencia en X y/o Y, éstas no serán trazadas, solamente se trazará la rejilla.

- **SER_RotulosEje(Eje,Rotulo)**

```
Eje          CHARACTER * (*),Entrada
Rotulo       CHARACTER * (*),Entrada
```

Establece el rótulo que se colocará en el eje que se indique. Para cada eje es necesario llamar a esta rutina. La forma de indicar el eje es proporcionando en una constante o variable la siguiente información: X para eje X, YI para el eje Y izquierdo y YD para el eje Y Derecho. Si el rótulo es mayor al área disponible para él, será truncado dicho rótulo.

- **SER_Logarit(Eje)**

```
Eje      CHARACTER * (*),Entrada
```

Establece que el eje indicado esté escalado en forma logarítmica. La forma de indicar el eje es proporcionando en una constante o variable la siguiente información: X para eje X, YI para el eje Y izquierdo y YD para el eje Y Derecho.

Para Histogramas, Polígonos de frecuencia y Barras apiladas no se permiten escalas logarítmicas.

- **SER_EscalaManual(Eje,ValInicial,ValFinal)**

```
Eje              CHARACTER * (*),Entrada
ValInicial       INTEGER*4,Entrada
ValFinal         INTEGER*4,Entrada
```

Establece el valor inicial y valor final que tendrá el eje indicado. La forma de indicar el eje es proporcionando en una constante o variable la siguiente información: X para eje X, YI para el eje Y izquierdo y YD para el eje Y Derecho.

6.4.3 Gráficas

El grupo de rutinas que a continuación se describen, son las que realizan el dibujo de las gráfica, así como las características de presentación iniciales o las que haya definido el usuario.

Debido a que la mayoría de las rutinas tienen parámetros con características similares, antes de describirlas, se explicarán estos parámetros.

- Grupos de datos

  En varios de los reportes se permite dibujar más de un grupo de datos (Barras, Lineas, XVsY, Barras Apiladas y Polígonos de frecuencia). Es decir, que para un solo valor en X correspondan uno o más valores en Y. Por ejemplo, para una serie de años (Eje X) se quiere comparar las ventas de distintos productos, en donde un grupo tendrá las ventas de un producto, otro grupo corresponderá a las ventas de otro producto, etcétera.

  Debido a que sería muy complicado el paso de un parámetro por cada grupo, se diseñaron las rutinas de tal forma que los valores de todos los grupos se indiquen en un solo parámetro (**Grupos**, **Muestras** o **X** y **Y**). La forma de hacerlo es colocando en el arreglo correspondiente, el primer grupo de datos y a continuación el siguiente grupo y así tantas veces como grupos de datos se tengan. Por ejemplo, si se tienen los siguientes grupos:

  Grupo 1: 10 20 30
  Grupo 2: 8 63 2
  Grupo 3: 99 81 75

  El arreglo que se proporcione como parámetro deberá ser:

  Grupos : 10 20 30 8 63 2 99 81 75

  En todas las rutinas, ya sea que contengan o no varios grupos de datos se deberá indicar el número de elementos que tiene cada grupo. Si existen varios grupos, todos deberán contener el mismo número de elementos con excepción de XvsY, en donde se permiten grupos de tamaño variable.

- Acotaciones Alfanuméricas

  En reportes en donde el eje X no se escala, sino que se colocan textos uniformemente espaciados, es necesario indicar dichos textos en un arreglo de tipo CHARACTER en donde cada elemento puede tener

longitud variable.

El número de acotaciones debe ser igual al número de elementos de los grupos, si es menor se trazarán textos en blanco.

Dependiendo de la longitud de la máxima acotación y del área disponible para cada una de ellas, se colocarán en forma vertical u horizontal con respecto al eje X.

- Nombres de los grupos

Para identificar cada grupo de datos se indica en un arreglo o una variable (si solo existe un grupo de datos) los nombres que deberán ser asociados a cada uno de ellos. La identificación de los grupos se hace mediante distintos tipos de relleno, de símbolos o de líneas, de acuerdo al tipo de reporte.

- Títulos

En todos los reportes se puede colocar una identificación general de una hasta tres líneas. La forma de indicarlo es por medio de un arreglo CHARACTER en donde cada elemento puede tener longitud variable.

- Tipos de Uniones de las líneas

En los reportes que contemplen trazo de líneas (XvsY, Diagramas de Línea o Error) se puede indicar el tipo de unión entre cada punto de la línea.

Los tipos disponibles son: unión con líneas rectas, sin unión (solo símbolo en el punto), unión con líneas rectas y símbolo en el punto y por último símbolos en el punto con una línea al eje X para y=0. En la explicación de las gráficas se indica cuales tienen disponibles cada una de ellas y con que valor se activan.

- Tipo de rótulos sobre la barra

En los reportes en donde existan barras (Diagramas de Barras, Barras Apiladas e Histogramas) es posible indicar que valor se desea colocar sobre la barra, contándose con las siguientes opciones: Valor absoluto y Valor porcental con respecto al grupo. En la explicación de las gráficas se indica cuales tienen disponibles cada una de ellas y con que valor se activan.

Todos los trazos que correspondan a textos serán truncado si exceden el área disponible para ellos.

- **SER_Barras(Grupos,NumGrupos,NumElemsGrupo,Acotaciones, NombresGrupos,Titulos,TipoValorBarra)**

Grupos	REAL*4 (*),Entrada

NumGrupos	INTEGER*4,Entrada
NumElemGrupo	INTEGER*4,Entrada
Acotaciones	CHARACTER (*) * (*),Entrada
NombresGrupos	CHARACTER (*) * (*),Entrada
Titulos	CHARACTER (*) * (*),Entrada
TipoValorBarra	INTEGER*4,Entrada

Dibuja un diagrama de barras de hasta 5 grupos de datos, como el que se muestra en la figura 6-7.

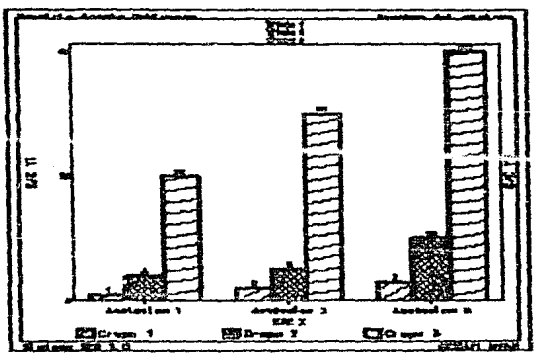


Figura 6-7

Los valores que pueden colocarse sobre la barra son: Valor absoluto (2), Valor porcentual (3) y ninguno (1).

o **SER_BarrasApiladas(Grupos,NumGrupos,NumElemGrupo,Acotaciones, NombresGrupos,Titulos,TipoValorBarra)**

Grupos	REAL*4 (*),Entrada
NumGrupos	INTEGER*4,Entrada
NumElemGrupo	INTEGER*4,Entrada
Acotaciones	CHARACTER (*) * (*),Entrada
NombresGrupos	CHARACTER (*) * (*),Entrada
Titulos	CHARACTER (*) * (*),Entrada
TipoValorBarra	INTEGER*4,Entrada

Dibuja un Diagrama de barras de mínimo 2 y máximo 5 grupos de datos, con la diferencia de que para todos los elementos asociados a una misma acotación serán trazadas sus barras equivalentes una a continuación de la otra, como se muestra en la figura 6-8.

Los valores que pueden colocarse sobre la barra son: Valor absoluto (2) y ninguno (1).

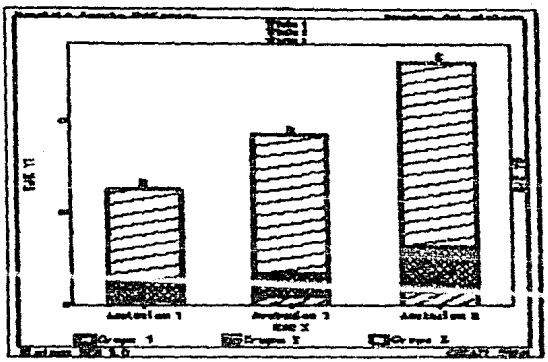


Figura 6-8

c SER_Lineas(Grupos,NumGrupos,NumElemGrupo,Acotaciones,
NombresGrupos,Titulos,TipoUnion)

Grupos	REAL*4 (*),Entrada
NumGrupos	INTEGER*4,Entrada
NumElemGrupo	INTEGER*4,Entrada
Acotaciones	CHARACTER (*) * (*),Entrada
NombresGrupos	CHARACTER (*) * (*),Entrada
Titulos	CHARACTER (*) * (*),Entrada
TipoUnion	INTEGER*4,Entrada

Traza un reporte de lineas de hasta 5 grupos de datos. La diferencia básica entre Diagrama de Lineas y una Grafica de X vs Y, es que en las primeras los valores en X son acotaciones alfanuméricas, mientras en X vs Y se tiene una escala numérica. En la figura 6-9 se muestra un ejemplo de este tipo de reportes.

Los tipos de unión con que se cuenta son: Lineas entre los puntos (3), solo símbolos en cada punto (2), los dos anteriores (1) y símbolo en el punto y una linea que lo une al eje X para Y = 0 (4).

o SER_Histograma(Muestras,NumElemMuestras,NumInt,
LimitesClase,TipoRotBarra,TipoRotYizqDer,
TipoAgrupa,NombresMuestras,Titulos)

Muestras	REAL*4 (*),Entrada
NumElemMuestras	INTEGER*4,Entrada
NumInt	INTEGER*4,Entrada
LimitesClase	REAL*4 (*),Entrada
TipoRotBarra	INTEGER*4,Entrada
TipoRotYizqDer	INTEGER*4 (*),Entrada

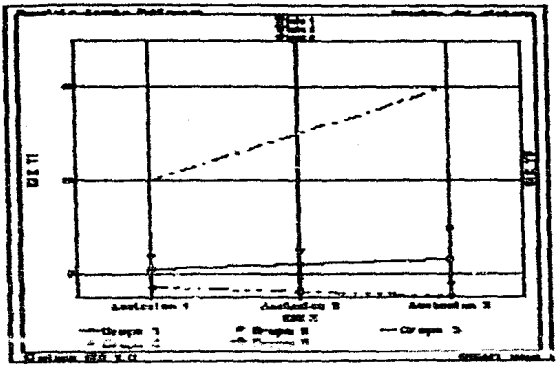


Figura 6-9

TipoAgrupa	INTEGER*4,Entrada
NombresMuestras	CHARACTER (*) * (*),Entrada
Titulos	CHARACTER (*) * (*),Entrada

Dibuja un histograma para una sola muestra de datos, éstas pueden estar dispersas o agrupadas.

En **TipoAgrupa** se indica si los datos están agrupados (2), es decir, los elementos del grupo de datos (**Muestras**) indican la frecuencia de los datos en cada intervalo de clase. Si los datos son directamente las muestras, es decir, están dispersos (**TipoAgrup** = 1) la rutina calcula las frecuencias. El máximo número de intervalos es 25.

Si los datos están agrupados, el parámetro **NumElemMuestras** debe valer cero, ya que **Numint** indica el número de intervalos de clase que contiene **Muestras**.

Si los datos no están agrupados **Numint** indica el número de intervalos en que se agrupan.

En **LimitesClase** se indica el inicio (elemento 1) y el final (elemento 2) de los intervalos de clase. Funciona tanto para datos agrupados como dispersos.

Los tipos de rótulos sobre las barras del histograma que se pueden solicitar son: Frecuencia (1), Frecuencia relativa (2) y ninguno (3). Por último con **TipoRotYIzqDer** se indica el tipo de rótulo para los dos eje verticales: Y Izquierdo (elemento 1) y Y Derecho (elemento 2). Los Textos que se pueden solicitar son: "Frecuencia" (1), "Frec. Relativa" (2) o ningún rótulo (3). La última opción solo es valida para Y Derecho. Un ejemplo de un histograma se observa en la figura 6-10.

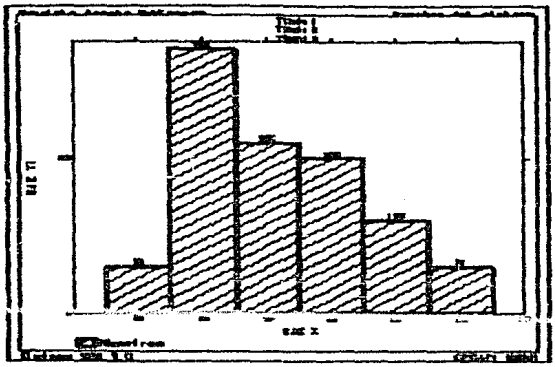


Figura 6-10

- **SER_PoligonoFrec(Muestras,NumPol,NumElemPol,NumInt, LimitesClase,TipoPol,TipoRotYIzqDer, TipoAgrupa,NombresPoligonos,Titulos)**

Muestras	REAL*4 (*),Entrada
NumPol	INTEGER*4,Entrada
NumElemPol	INTEGER*4 (*),Entrada
NumInt	INTEGER*4,Entrada
LimitesClase	REAL*4 (*),Entrada
TipoPol	INTEGER*4,Entrada
TipoRotYIzqDer	INTEGER*4 (*),Entrada
TipoAgrupa	INTEGER*4,Entrada
NombresPoligonos	CHARACTER (*) * (*),Entrada
Titulos	CHARACTER (*) * (*),Entrada

Dibuja un polígono de frecuencias de hasta tres muestras de datos, estas pueden estar dispersas o agrupadas.

En **TipoAgrup** se indica si los datos están agrupados (2), es decir, los elementos del grupo de datos (**Muestras**) indican la frecuencia de los datos en cada intervalo de clase. Si los datos son directamente las muestras, es decir, están dispersos (**TipoAgrup** = 1) la rutina calcula las frecuencias. El máximo número de intervalos es 25.

Si los datos están agrupados, el parámetro **NumElemPol** debe valer cero, ya que **NumInt** indica el número de intervalos de clase que contiene **Muestras**.

Si los datos no estan agrupados **NumInt** indica el número de intervalos en que se requiere se agrupen. Asimismo, **NumElemPol** indicará el número de elementos que cada muestra tiene, pudiendo ser distinto el tamaño entre ellas.

En **LimitesClase** se indica el inicio (elemento 1) y el final (elemento 2) de los intervalos de clase. Funciona tanto para datos agrupados como dispersos.

TipoPol indica el tipo de polígono que se desea, pudiendo valer: Frecuencia (1) o Frecuencia acumulada (2). Por último en **TipoRotYIzqDer** se indica el rótulo para los ejes Y Izquierdo (elemento 1) y Y Derecho (elemento 2). Los Textos que se pueden solicitar son: "Frecuencia" (1), "Frec. Relativa" (2) y ningún rótulo (3). La última opción solo es valida para Y Derecho. Un ejemplo de un polígono de frecuencia se muestra en la figura 6-11.

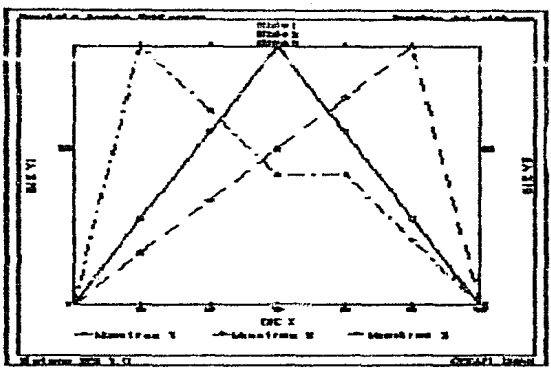


Figura 6-11

- SER_XvsY(X,Y,NumFunciones,IndInicial,IndFinal,
  EscalasIguales,NombresFuno,Titulos,TipoUnion)

X	REAL*4 (*),Entrada
Y	REAL*4 (*),Entrada
NumFunciones	INTEGER*4,Entrada
IndInicial	INTEGER*4 (*),Entrada
IndFinal	INTEGER*4 (*),Entrada
EscalasIguales	INTEGER*4,Entrada
NombresFunc	CHARACTER (*) * (*),Entrada
Titulos	CHARACTER (*) * (*),Entrada
TipoUnion	INTEGER*4 (*),Entrada

Dibuja un reporte de parejas de puntos X,Y, de hasta 5 grupos. Se permite que cada grupo tenga distinto número de elementos. Para indicar el comienzo y final de cada grupo se utilizan los arreglos **IndFinal** e **IndInicial** en donde cada elemento va asociado al número del grupo.

Para cada grupo de Y, debe existirá un grupo X. Si los grupos Y tienen el mismo grupo de X asociado, será necesario que se repitan las X tantas veces como grupos Y existan. Algunos ejemplos son:

```
Grupo Y1: 10 20 30
Grupo X1:  1  3  5
Grupo Y2:  8 63  2 34 23
Grupo X2: 99 81 75 12  3
```

Los arreglos que se proporcionen como parámetros deberán ser:

```
Grupos Y:   10 20 30  8 63  2 34 23
Grupos X:    1  3  5 99 81 75 12  3
IndInicial: 1 4
IndFinal :  3 8
```

Para el caso de un mismo grupo X para varios grupos Y:

```
Grupo Y1: 10 20 30 40
Grupo X1:  1  3  5  7
Grupo Y2:  8 63  2 34
Grupo X2:  1  3  5  7
```

Los arreglos que se proporcionen como parámetros deberán ser:

```
Grupos Y:   10 20 30 40  8 63  2 34
Grupos X:    1  3  5  7  1  3  5  7
IndInicial: 1 5
IndFinal :  4 8
```

El reporte de la figura 6-12 muestra una gráfica de X vs Y.

o **SER_DiagramaCircular(Rebanadas,Porcent,NumRebanadas,RebExplotar, NumRebExplotar,NombresReb,Titulos)**

Rebanadas	REAL*4 (*),Entrada
Porcent	LOGICAL*1,Entrada
NumRebanadas	INTEGER*4,Entrada
RebExplotar	INTEGER*4 (*),Entrada
NumRebExplotar	INTEGER*4,Entrada
NombresReb	CHARACTER * (*),Entrada
Titulos	CHARACTER * (*),Entrada

Dibuja un Diagrama Circular de hasta 20 valores. A cada valor le corresponde en forma proporcional un arco (rebanada) con respecto al total del grupo, en donde dichos valores pueden estar en valor absoluto o en forma porcentual (Porcent igual a verdadero).

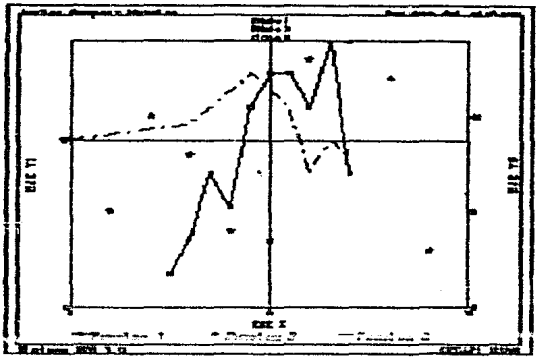


Figura 6-12

Con la finalidad de resaltar algunas rebanadas es posible indicar por medio del arreglo RebExplotar las posiciones de las rebanadas que se desean separar de las otras. Dichas posiciones deberán estar ordenadas ascendentemente dentro del arreglo. La figura 6-13 muestra un Diagrama Circular.

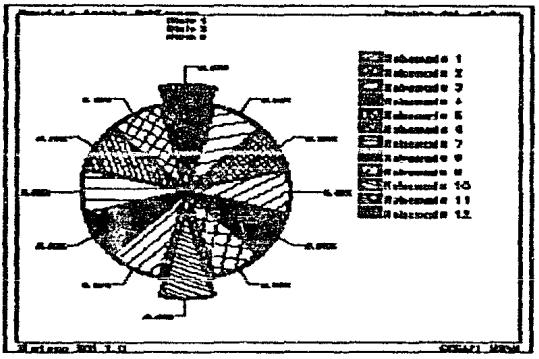


Figura 6-13

- SER_Error(RotulosX,YError,NumGrupos,NumElemGrupo, NombresGrupos,Titulos,TipoUnion)

RotulosX	CHARACTER (*) * (*),Entrada
YError	REAL*4 (*),Entrada
NumGrupos	INTEGER*4,Entrada

NumElemGrupo	INTEGER*4,Entrada
NombresGrupos	CHARACTER (*) * (*),Entrada
Titulos	CHARACTER (*) * (*),Entrada
TipoUnion	INTEGER*4,Entrada

Dibuja para un solo grupo acotaciones en X (**Rotulos**) una serie de valores (**YError**) que son desviaciones sobre un punto. Se pueden tener hasta 5 grupos de desviaciones o errores.

Los tipos de unión son: sin conección de puntos medios (1) y con conección (2). La figura 6-14 muestra un reporte de gráfica de este tipo.

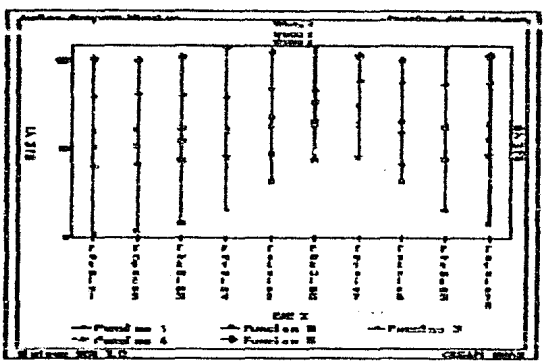


Figura 6-14

- **SER_BarrasLinea(Barras,Linea,NumElemGrupo,Acotaciones,NombresGrupos, Titulos,NumEscalas,TipoValorBarra,TipoUnionLinea)**

Barras	REAL*4 (*),Entrada
Lineas	REAL*4 (*),Entrada
NumElemGrupo	INTEGER*4,Entrada
Acotaciones	CHARACTER (*) * (*),Entrada
NombresGrupos	CHARACTER (*) * (*),Entrada
Titulos	CHARACTER (*) * (*),Entrada
NumEscalas	INTEGER*4,Entrada
TipoValorBarra	INTEGER*4,Entrada
TipoUnionLinea	INTEGER*4,Entrada

Dibuja en el reporte una gráfica de Diagrama de Barras y uno de Lineas, con las mismas características que un reporte sin combinar. Los grupos de datos se indican en **Barras** y **Lineas** (solo un grupo para cada tipo de gráficas).

NumEscalas indica el número de escalas que se desean si vale 1 los dos grupos de datos se escalarán con respecto a YI y si vale 2 las barras serán escaladas con respecto a YI y las líneas a YD.

Los tipos de rótulos sobre la barra y unión de línea son los mismos que para las gráficas sencillas. **TipoValorBarra** si vale 1 será el valor absoluto, 2 el valor porcentual y 3 no habra rótulo. Para **TipoUniónLinea** el valor de 3 indica líneas entre los puntos, 2 solo símbolos en cada punto, los dos anteriores con 1 y símbolo en el punto y una línea que lo une al eje X para Y=0 con 4. La figura 6-15 muestra un reporte generado con esta rutina:

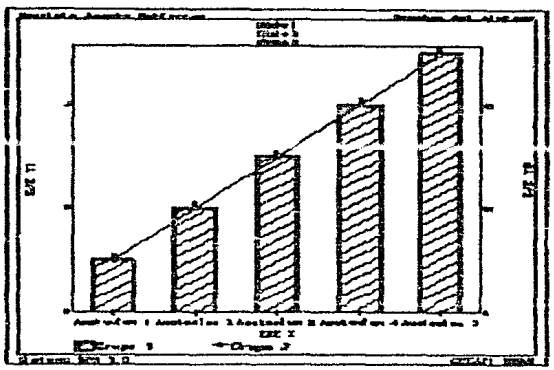


Figura 6-15

- **SER_BarXvsY(XBarras,YBarras,NumElemBarras,NombreBarras,TipoValorBarra, XXvsY,YXvsY,NumElemXvsY,NombreXvsY,TipoUnion,Titulos)**

XBarras	CHARACTER (*) * (*),Entrada
YBarras	REAL*4 (*),Entrada
NumElemBarras	INTEGER*4,Entrada
NombreBarras	CHARACTER * (*),Entrada
TipoValorBarra	INTEGER*4,Entrada
XXvsY	REAL*4 (*),Entrada
YXvsY	REAL*4 (*),Entrada
NumElemXvsY	INTEGER*4,Entrada
NombreXvsY	CHARACTER * (*),Entrada
TipoUnion	INTEGER*4,Entrada
Titulos	CHARACTER (*) * (*),Entrada

Dibuja un Diagrama de Barras (un solo grupo) con una gráfica XvsY (un solo grupo). Las características son las mismas que para gráficas sin combinación. **XBarras** tiene las acotaciones y **YBarras** los valores. Así **XXvsY** y **YXvsY** almacenan los valores de la abscisas y ordenadas respectivamente.

Se tendrán dos escalas: a la Y Izquierda se asociarán las barras y a la Y Derecha los puntos de XvsY. Las marcas sobre el eje X son las acotaciones de las Barras. Las marcas de las abscisas de XvsY no serán dibujadas.

Los tipos de rótulos sobre la barra y tipo de unión de XvsY son los mismos que para las gráficas sencillas. **TipoValorBarra** si vale 1 será el valor absoluto, 2 el valor porcentual y 3 no habra rótulo. Para **TipoUniónLinea** el valor de 3 indica líneas entre los puntos, 2 solo símbolos en cada punto, los dos anteriores con 1 y símbolo en el punto y una línea que lo une al eje X para Y=0 con 4. En la figura 6-16 se muestra un reporte generado con esta rutina:

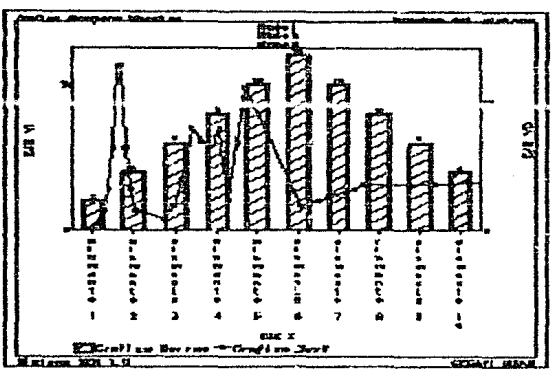


Figura 6-16

- **SER_LinXvsY(XLineas,YLineas,NumElemLineas,NombreLineas, TipoUnionLinea,XXvsY,YXvsY,NumElemXvsY, NombreXvsY,TipoUnionXvsY,Titulos)**

XLineas	CHARACTER (*) * (*),Entrada
YLineas	REAL*4 (*),Entrada
NumElemLineas	INTEGER*4,Entrada
NombreLineas	CHARACTER * (*),Entrada
TipoUnionLinea	INTEGER*4,Entrada
XXvsY	REAL*4 (*),Entrada
YXvsY	REAL*4 (*),Entrada
NumElemXvsY	INTEGER*4,Entrada
NombreXvsY	CHARACTER * (*),Entrada
TipoUnionXvsY	INTEGER*4,Entrada
Titulos	CHARACTER (*) * (*),Entrada

Dibuja un Diagrama de Líneas (un solo grupo) con una gráfica XvsY (un solo grupo). Las características son las mismas que para gráficas sin combinación. **XLineas** tiene las acotaciones y **YLineas** los valores. Así **XXvsY** y **YXvsY** almacenan los valores de la abscisas y ordenadas

respectivamente.

Se tendrán dos escalas: a la Y Izquierda se asociarán las líneas y a la Y Derecha la gráfica XvsY. Las marcas sobre el eje X son las acotaciones de las Líneas. Las marcas para la escala de las abscisas de XvsY no serán dibujadas.

Los tipos de unión de los puntos (X,Y) son los mismos que para las gráficas sencillas. Los valores para **TipoUnionLinea** y **TipoUnionXvsY** son: 3 indica líneas entre los puntos, 2 solo símbolos en cada punto, 1 los dos anteriores y 4 símbolo en el punto y una línea que lo une al eje X para Y=0. La figura 6-17 muestra un reporte generado con esta rutina.

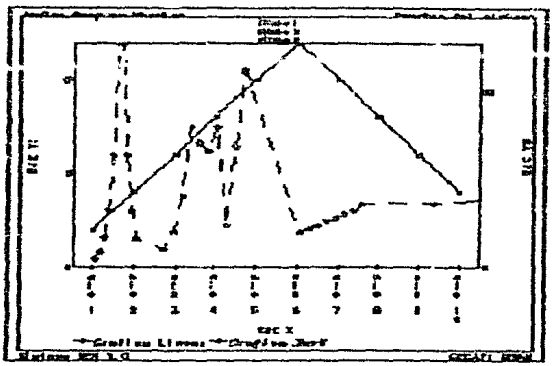


Figura 6-17

- **SER_HistogramaPoligono(Muestras,NumElemHistoPol,NumInt,LimitesClase, TipoRotBarra,TipoRotYIzqDer,TipoAgrupa, NombresHistoPol,Titulos)**

Muestras	REAL*4 (*),Entrada
NumElemHistoPol	INTEGER*4,Entrada
NumInt	INTEGER*4,Entrada
LimitesClase	REAL*4 (*),Entrada
TipoRotBarra	INTEGER*4,Entrada
TipoRotYIzqDer	INTEGER*4 (*),Entrada
TipoAgrupa	INTEGER*4,Entrada
NombresHistoPol	CHARACTER (*) * (*),Entrada
Titulos	CHARACTER (*) * (*),Entrada

Dibuja un histograma de una muestra de datos combinada con un Polígono de Frecuencia de una sola muestra.

Las muestras pueden estar agrupadas (**TipoAgrupa** igual a 2) o dispersos (**TipoAgrupa** igual a 1).

Los tipos de rótulos sobre la barra son los mismos que para las gráficas sencillas. Si **TipoValorBarra** vale 1 será el valor absoluto, 2 el valor porcentual y 3 no habrá rótulo. **TipoRotYIzqDer** indica el tipo de rótulo que se quiere tenga la escala Y. Los Textos que se pueden solicitar son: "Frecuencia" (1), "Frec. Relativa" (2) y ningún rótulo (3).

Un reporte que muestra el resultado de una llamada a esta rutina se observa en la figura 6-18.

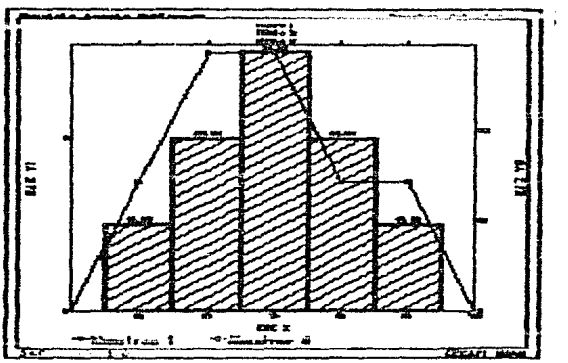


Figura 6-18

- SER_HistoXvsY(Muestra,NumElemMuestra,NumInt,LimitesClase,
  TipoRotBarra,TipoRotYIzq,TipoAgrupa,NombreMuestra,
  XXvsY,YXvsY,NumElemXvsY,NombreXvsY,
  TipoUnionXvsY,Titulos)

Muestra	REAL*4 (*),Entrada
NumElemMuestra	INTEGER*4,Entrada
NumInt	INTEGER*4,Entrada
LimitesClase	REAL*4 (*),Entrada
TipoRotBarra	INTEGER*4,Entrada
TipoRotYIzq	INTEGER*4,Entrada
TipoAgrupa	INTEGER*4,Entrada
NombreMuestra	CHARACTER * (*),Entrada
XXvsY	REAL*4 (*),Entrada
YXvsY	REAL*4 (*),Entrada
NumElemXvsY	INTEGER*4,Entrada
NombreXvsY	CHARACTER * (*),Entrada
TipoUnionXvsY	INTEGER*4,Entrada
Titulos	CHARACTER (*) * (*),Entrada

Dibuja un Histograma de una muestra de datos combinada con una gráfica de XvsY de un solo grupo.

Las muestras pueden estar agrupadas (**TipoAgrupa** igual a 2) o dispersas (**TipoAgrupa** igual a 1).

Se tendrán dos escalas: a la Y Izquierda se asociará el Histograma y a la Y Derecha la gráfica XvsY. Las marcas sobre el eje X son los intervalos de clase. Las marcas para la escala de las abscisas de XvsY no serán dibujadas.

Los tipos de rótulos sobre la barra y unión de XvsY son los mismos que para las gráficas sencillas. **TipoValorBarra** si vale 1 será el valor absoluto, 2 el valor porcentual y 3 no habrá rótulo. Para **TipoUniónLinea** el valor de 3 indica líneas entre los puntos, 2 solo símbolos en cada punto, los dos anteriores con 1 y símbolo en el punto y una línea que lo une al eje X para Y=0 con 4. La figura 6-19 muestra el resultado de una llamada a esta rutina.

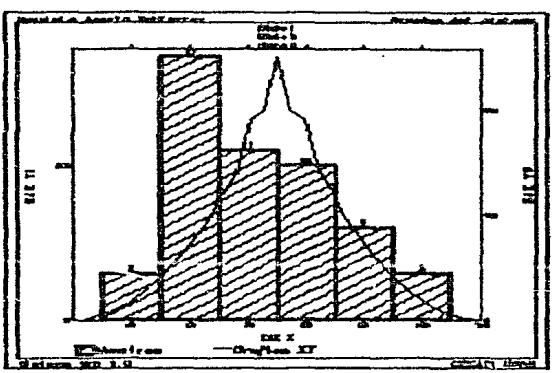


Figura 6-19

- **SER_PoligonoXvsY(Muestra,NumElemMuestra,NumInt,LimitesClase, TipoPol,TipoRotPol,TipoAgrupa,NombreMuestra, XXvsY,YXvsY,NumElemXvsY,NombroXvsY, TipoUnionXvsY,Titulos)**

Muestra	REAL*4 (*),Entiada
NumElemMuestra	INTEGER*4,Entrada
NumInt	INTEGER*4,Entrada
LimitesClase	REAL*4 (*),Entrada
TipoPol	INTEGER*4,Entrada
TipoRotPol	INTEGER*4,Entrada
TipoAgrupa	INTEGER*4,Entrada
NombreMuestra	CHARACTER * (*),Entrada

RUTINAS GRAFICAS

XXvsY	REAL*4 (*),Entrada
YXvsY	REAL*4 (*),Entrada
NumElemXvsY	INTEGER*4,Entrada
NombreXvsY	CHARACTER * (*),Entrada
TipoUnionXvsY	INTEGER*4,Entrada
Titulos	CHARACTER (*) * (*),Entrada

Dibuja un Polígono de Frecuencia de una muestra de datos combinada con una gráfica de XvsY de un solo grupo.

Las muestras pueden estar agrupadas (TipoAgrupa igual a 2) o dispersos (TipoAgrupa igual a 1).

Se tendrán dos escalas: a la Izquierda se asociará el Polígono y a la Y Derecha la gráfica XvsY. Las marcas sobre el eje X son los intervalos de clase. Las marcas para la escala de las abscisas de XvsY no serán dibujadas.

Los tipos de unión son los mismos que para las gráficas sencillas y se indican en TipoUniónLinea. Con un valor de 3 se indican lineas entre los puntos, 2 solo símbolos en cada punto, los dos anteriores con 1 y símbolo en el punto y una linea que lo une al eje X para Y=0 con 4.

El polígono puede ser dibujado con frecuencia normal (1) o con frecuencia relativa (2). El reporte de la figura 6-20 muestra el resultado de una llamada a esta rutina.

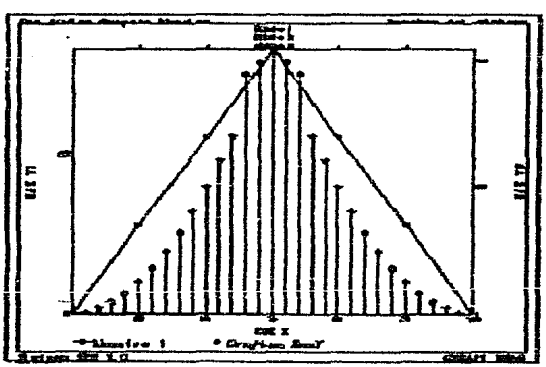


Figura 6-20

6.4.4 Terminación De Un Reporte

- **SER_TERMINAHOJA**

Cierra la hoja de dibuja. Una vez invocada esta rutina, no puede generarse ningún reporte gráfico, sin embargo, sí puede definirse otra hoja de dibujo.

Es necesario llamar a esta rutina, si se quiere asegurar que la hoja de dibujo sea correctamente generada.

6.4.5 Auxiliares

Es esta sección se indican las rutinas que se realizaron como apoyo a las rutinas SER, las cuales no son usadas directamente por la aplicación del usuario.

- **ISER_IniciaCommons**

  Inicializa las condiciones de presentación para los reportes. En la sección MODIFICADORES DE PRESENTACION se indican estas condiciones.

  Es llamada al final de las rutinas que generan los reportes, antes de regresar el control al programa de aplicación.

- **ISER_ObtenMinMax(Vector,IndInicio,IndFin,Minimo,Maximo)**

Vector	REAL*4 (*),Entrada
IndInicio	INTEGER*4,Entrada
IndFin	INTEGER*4,Entrada
Minimo	REAL*4,Salida
Maximo	REAL*4,Salida

  Obtiene el valor mínimo y el máximo de un vector de datos desde el elemento indicado por IndInicio hasta el elemento IndFin.

- **ISER_SumaGrupos(Grupos,NumGrupos,NumElemGrupo,Suma)**

Grupos	REAL*4 (*),Entrada
NumGrupos	INTEGER*4,Entrada
NumElemGrupo	INTEGER*4,Entrada
Suma	REAL*4 (*),Salida

  Obtiene la suma de los valores de cada grupo. Principalmente se utiliza para obtener los porcentajes de los datos con respecto a su grupo.

- **ISER_AgrupaDatos(Grupos,IntClaseInicio,IntClaseFin,NumInt,**
  NumGrupos,NumElemGrupo,TipoAgrupa,Frecuencias)

Grupos	REAL*4 (*),Entrada
IntClaseInicio	REAL*4,Entrada
IntClaseFin	REAL*4,Entrada
NumInt	INTEGER*4,Entrada
NumGrupos	INTEGER*4,Entrada
NumElemGrupo	INTEGER*4 (*4),Entrada/Salida
TipoAgrupa	INTEGER*4,Entrada
Frecuencias	REAL*4 (*),Salida

Es una función que calcula las frecuencias de los datos que se almacenan en el arreglo **Grupos** y las deja en **Frecuencias**. Realizá la asignación de la frecuencia a los intervalos de acuerdo a los valores de **IntClaseInicio** y **IntClaseFin**.

Se pueden calcular las frecuencias para varios grupos de datos y con distinto tamaño.

Los valores que regresa la función son:

0 No hubo error.
1 Elemento menor al intervalo de clase inicial.
2 Elemento mayor al intervalo de clase final.
3 Elemento menor y mayor a los intervalos inicial y final.

o **ISER_ObtenEscalasXY(Eje,Grupos,NumGrupos,IndiceInicial,IndiceFinal, GruposYD,EscManual,Logarit,Ventana)**

Eje	CHARACTER * (*),Entrada
Grupos	REAL*4 (*),Entrada
NumGrupos	INTEGER*4,Entrada
IndiceInicial	INTEGER*4 (*),Entrada
IndiceFinal	INTEGER*4 (*),Entrada
GruposYD	CHARACTER * (*),Entrada
EscManual	REAL*4 (*),Entrada
Logarit	CHARACTER * (*),Entrada
Ventana	REAL*4 (*),Salida

Determina el valor inicial y final (**Ventana**) que debe tener la escala del eje que se haya indicado, solamente para gráficas de XvsY.

La ventana se determina en base a la solicitud de escala manual o automática, en forma logarítmica o no y de acuerdo a los valores de los grupos que estén asociados a ese eje.

Debido a que en las gráficas XvsY se permiten grupos de distintos tamaños, es necesario indicar el elemento inicial y final de cada grupo.

o **ISER_ObtenEscalas(Eje,Grupos,NumGrupos,NumElemGrupo, GruposYD,EscManual,Logarit,Ventana)**

Grupos	REAL*4 (*),Entrada
IntClaseInicio	REAL*4,Entrada
IntClaseFin	REAL*4,Entrada
NumInt	INTEGER*4,Entrada
NumGrupos	INTEGER*4,Entrada
NumElemGrupo	INTEGER*4 (*4),Entrada/Salida
TipoAgrupa	INTEGER*4,Entrada
Frecuencias	REAL*4 (*),Salida

Es una función que calcula las frecuencias de los datos que se almacenan en el arreglo **Grupos** y las deja en **Frecuencias**. Realizá la asignación de la frecuencia a los intervalos de acuerdo a los valores de **IntClaseInicio** y **IntClaseFin**.

Se pueden calcular las frecuencias para varios grupos de datos y con distinto tamaño.

Los valores que regresa la función son:

0 No hubo error.
1 Elemento menor al intervalo de clase inicial.
2 Elemento mayor al intervalo de clase final.
3 Elemento menor y mayor a los intervalos inicial y final.

o **ISER_ObtenEscalasXY(Eje,Grupos,NumGrupos,IndiceInicial,IndiceFinal, GruposYD,EscManual,Logarit,Ventana)**

Eje	CHARACTER * (*),Entrada
Grupos	REAL*4 (*),Entrada
NumGrupos	INTEGER*4,Entrada
IndiceInicial	INTEGER*4 (*),Entrada
IndiceFinal	INTEGER*4 (*),Entrada
GruposYD	CHARACTER * (*),Entrada
EscManual	REAL*4 (*),Entrada
Logarit	CHARACTER * (*),Entrada
Ventana	REAL*4 (*),Salida

Determina el valor inicial y final (**Ventana**) que debe tener la escala del eje que se haya indicado, solamente para gráficas de XvsY.

La ventana se determina en base a la solicitud de escala manual o automática, en forma logarítmica o no y de acuerdo a los valores de los grupos que estén asociados a ese eje.

Debido a que en las gráficas XvsY se permiten grupos de distintos tamaños, es necesario indicar el elemento inicial y final de cada grupo.

o **ISER_ObtenEscalas(Eje,Grupos,NumGrupos,NumElemGrupo, GruposYD,EscManual,Logarit,Ventana)**

Eje	CHARACTER * (*),Entrada
Grupos	REAL*4 (*),Entrada
NumGrupos	INTEGER*4,Entrada
NumElemGrupo	INTEGER*4,Entrada
GruposYd	CHARACTER * (*),Entrada
EscManual	REAL*4 (*),Entrada
Logarit	CHARACTER * (*),Entrada
Ventana	REAL*4 (*),Salida

Determina el valor inicial y final (**Ventana**) que debe tener la escala del eje que se haya indicado para todos los reportes excepto en Diagramas Circulares y XvsY.

La ventana se determina en base a la solicitud de escala manual o automática, en forma logarítmica o no y de acuerdo a los valores de los grupos que estén asociados a ese eje.

o **ISER_EscalasIguales(X,Y,IndInicial,IndFinal,NumFunciones,Referencias, Incremento,VentX,VentY1,VentYD)**

X	REAL*4 (*),Entrada
Y	REAL*4 (*),Entrada
IndInicial	INTEGER*4 (*),Entrada
IndFinal	INTEGER*4 (*),Entrada
NumFunciones	INTEGER*4,Entrada
Referencias	CHARACTER * (*),Entrada
Incremento	INTEGER*4,Salida
VentX	REAL*4 (*),Salida
VentY1	REAL*4 (*),Salida
VentYD	REAL*4 (*),Salida

Obtiene el número de marcas en que deben dividirse los ejes (éstas marcas deben ser tanto física como logicamente iguales) con el fin de poder tener escalas iguales en los tres ejes. Solo se utiliza para XvsY.

Esta característica se utiliza para funciones en que no se desea que por no ser cuadrada la hoja, las gráficas salgan deformadas, por ejemplo: un círculo o una elipse.

Referencias indica a que escala están asociados los grupos: Y Izquierda o Y Derecha.

o **ISER_ParamsError(YError,NumGrupos,NumElemGrupo,MaxMin,Medios,TipoUnion)**

YError	REAL*4 (*),Entrada
NumGrupos	INTEGER*4,Entrada
NumElemGrupo	INTEGER*4,Entrada
MaxMin	REAL*4 (30,2),Entrada
Medios	REAL*4 (*),Entrada

TipoUnion INTEGER*4,Entrada

Del grupo de desviaciones YError obtiene los valores máximos y mínimos para cada acotación (los deposita en el arreglo MaxMin), así como los valores médios. En TipoUnion se indicala forma de conección de los puntos: 1 para que no se conecten puntos medios y 2 para conectarlos.

6.A.6 Ejemplos

En esta sección se muestran tres ejemplo que utilizan las rutinas del SER, para el caso de uno (listado 7-I), dos (listado 7-II) y cuatro reportes (listado 7-III) por hoja.

```
PROGRAM Un_Reporte

! Obtiene un reporte de la combinacion de las graficas de
! Histogramas y XvsY, mostrando las rutinas necesarias para
! su generacion, asi como las rutinas de modificacion de
! presentacion.
IMPLICIT NONE

CHARACTER
1    Dispositivo * 40,
1    NombresFunc   (5) * 40 /'Funcion 1',
1                           'Funcion 2',
1                           'Funcion 3',
1                           'Funcion 4',
1                           'Funcion 5'/,
1    NombreMuestras * 40   /'Muestras'/,
1    NombreXY * 40         /'Grafica XY'/,
1    Rotulos(2) * 30,
1    Titulos (3) * 40 /'Titulo 1',
1                      'Titulo 2',
1                      'Titulo 3'/
 INTEGER
1    NumElemMuestra,
1    NumDiv,
1    NumInt,
1    NumElemXY,
1    TipoRotBarra,
1    TipoRotYIzqDer,
1    TipoAgrup,
1    TipoUnionXY
 REAL
1    Largo,
1    Ancho,
1    LimitesClase (2),
1    Muestras(100) /3*50,
1                  17*60.0,
1                  11*70.0,
```

```
1                    10*80.0,
1                     6*90.0,
1                     3*100.0,
1                    50*0/,
1     X(30) /1,2,3,4,5,6,7,8,9,10,
1            11,12,13,14,15,16,17,18,19,20,
1            21,22,23,24,25,26,27,28,29,30/,
1     Y(30) /1,4,9,16,25,36,49,64,81,100,
1            121,140,190,200,250,200,190,140,121,100,
1            81,64,49,36,25,16,9,4,1,0/

 Ancho = 11.0
 Largo = 8.5
 NumDiv = 1
 Dispositivo='HistoXvsY.PLT/pr'
 Rotulos(1)='Comparacion estadistica'
 Rotulos(2)='Probabilidad'
 CALL SER_DespliegueSimple(1.0,1.0,1.0,1.0)
 CALL SER_DefineHoja(Largo,Ancho,NumDiv,Rotulos,Dispositivo)

 CALL SER_RotulosEje('x ','EJE X')
 CALL SER_RotulosEje('Yi ','EJE Yi')
 CALL SER_RotulosEje('YD ','EJE YD')

 NumInt = 6
 LimitesClase(1) = 45.0
 LimitesClase(2) = 105.0
 NumElemMuestra = 50
 TipoAgrup = 1              !Dispersos
 TipoRotBarra = 1           !Frecuencia absoluta
 TipoRotYizqDer = 2         !Frecuencia relativa
 NumElemXY = 30
 TipoUnionXY = 3

 CALL SER_HistoXVSY (Muestras,NumElemMuestra,NumInt,
1                    LimitesClase,TipoRotBarra,TipoRotYizqDer,
1                    TipoAgrup,NombreMuestras,X,Y,NumElemXY,
1                    NombreXY,TipoUnionXy,Titulos)

 CALL SER_TerminaHoja
 END
```

Listado 7-I

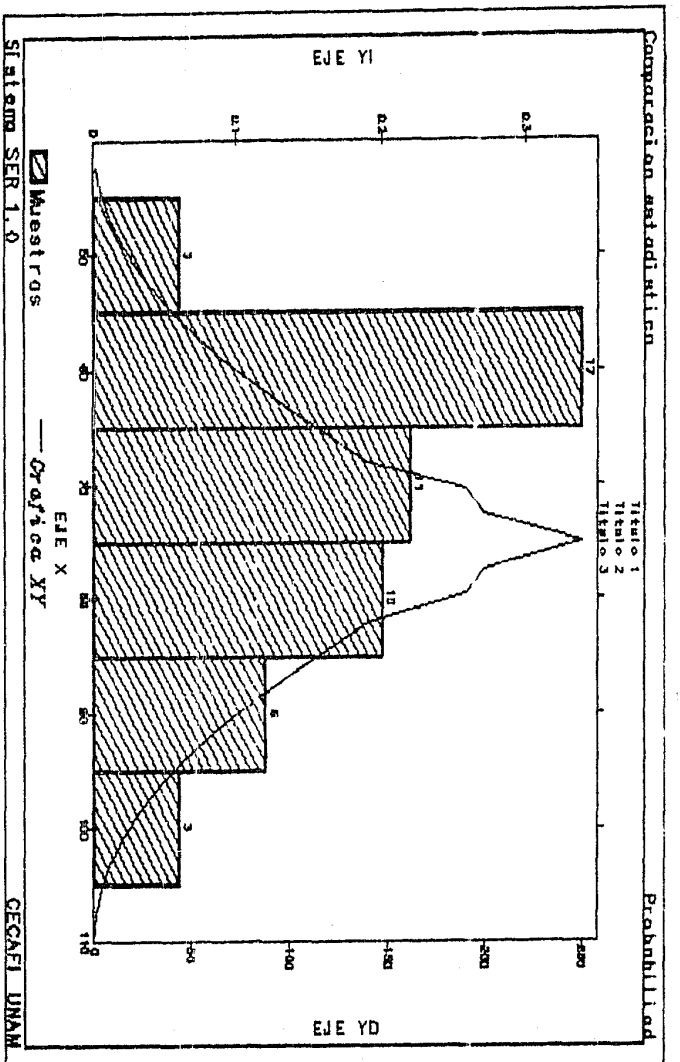

Comparacion estadistica
Probabilidad
EJE YI
EJE YD
EJE X
Muestras
Grafica XY
Titulo 1
Titulo 2
Titulo 3
Sistema SER 1.0
CECAFI UNAM

Figura 6-21

```
PROGRAM Dos_Reportes

! Obtiene dos reportes (Diagrama circular y Histograma) en una sola
! hoja de dibujo.
IMPLICIT NONE

CHARACTER
1    NombreMuestras * 40   /'Muestras'/,
1    NombresReb (6) * 40   /'Rebanada 1',
1                           'Rebanada 2',
1                           'Rebanada 3',
1                           'Rebanada 4',
1                           'Rebanada 5',
1                           'Rebanada 6'/,
1    Titulos (3) * 40,
1    Rotulos (2) * 40
INTEGER
1    NumRebanadas,
1    NumRebExplotar,
1    NumElemMuestra,
1    NumInt,
1    RebExplotar (3) /1,3,5/,
1    TipoRotBarra,
1    TipoRotYIzqDer (2),
1    TipoAgrup
LOGICAL * 1
1    Porcent
REAL
1    AnchoHoja,
1    LargoHoja,
1    LimitesClase (2),
1    Muestras(100)/ 3*50,
1                  10*60,
1                   8*75,
1                  17*90,
1                  12*100,
1                  50*0/,
1    Rebanadas (6)/10.0,20.0,15.0,10.0,5.0,20.0/

LargoHoja = 9.0
AnchoHoja = 6.5
Rotulos(1)='Estadisticas'
Rotulos(2)='Reporte general'

CALL SER_DesplazaGrafica(0.8,0.8,0.5,0.5)
CALL SER_DefineHoja(LargoHoja,AnchoHoja,2,Rotulos,'GRAF.PLT/PR')

!!Se invoca a la rutina que traza un diagrama circular con 6 valores
NumRebanadas = 6
NumRebExplotar = 3
Porcent = .FALSE.
Titulos(1) = 'Diagrama'
Titulos(2) = 'Circular'
```

```
PROGRAM Dos_Reportes

! Obtiene dos reportes (Diagrama circular y Histograma) en una sola
! hoja de dibujo.
IMPLICIT NONE

CHARACTER
1    NombreMuestras * 40   /'Muestras'/,
1    NombresReb (6) * 40   /'Rebanada 1',
1                           'Rebanada 2',
1                           'Rebanada 3',
1                           'Rebanada 4',
1                           'Rebanada 5',
1                           'Rebanada 6'/,
1    Titulos (3) * 40,
1    Rotulos (2) * 40
INTEGER
1    NumRebanadas,
1    NumRebExplotar,
1    NumElemMuestra,
1    NumInt,
1    RebExplotar (3) /1,3,5/,
1    TipoRotBarra,
1    TipoRotYIzqDer (2),
1    TipoAgrup
LOGICAL * 1
1    Porcent
REAL
1    AnchoHoja,
1    LargoHoja,
1    LimitesClase (2),
1    Muestras(100)/ 3*50.
1                  10*60,
1                   8*75,
1                  17*90,
1                  12*100,
1                  50*0/,
1    Rebanadas (6)/10.0,20.0,15.0,10.0,5.0,20.0/

LargoHoja = 9.0
AnchoHoja = 6.5
Rotulos(1)='Estadisticas'
Rotulos(2)='Reporte general'

CALL SER_DesplazaGrafica(0.8,0.8,0.5,0.5)
CALL SER_DefineHoja(LargoHoja,AnchoHoja,2,Rotulos,'GRAF.PLT/PR')

!!Se invoca a la rutina que traza un diagrama circular con 6 valores
NumRebanadas = 6
NumRebExplotar = 3
Porcent = .FALSE.
Titulos(1) = 'Diagrama'
Titulos(2) = 'Circular'
```

```
Titulos(3) = ' '

CALL SER_DiagramaCircular(Rebanadas,Porcent,NumRebanadas,
1                RebExplotar,NumRebExplotar,NombresReb,Titulos)

!!Se definen las caracteristicas del histograma y se llama a la
!!rutina que obtiene su reporte
NumInt = 6
LimitesClase(1) = 45.0
LimitesClase(2) = 105.0
NumElemMuestra = 50.0
TipoAgrup = 1            !Dispersos
TipoRotBarra = 2         !Frecuencia relativa
TipoRotYIzqDer(1) = 2    !Frecuencia relativa
TipoRotYIzqDer(2) = 3    !Sin rotulo

CALL SER_Rejilla(1)

CALL SER_RotulosEje('X ','EJE X')
CALL SER_RotulosEje('YI ','EJE YI')

Titulos(1) = 'Histograma'
Titulos(2) = ' '

CALL SER_Histograma (Muestras,NumElemMuestra,NumInt,
1                LimitesClase,TipoRotBarra,TipoRotYIzqDer,
1                      TipoAgrup,NombreMuestras,Titulos)

CALL SER_TerminaHoja
CLOSE (1)
END
```

Listado 7-II

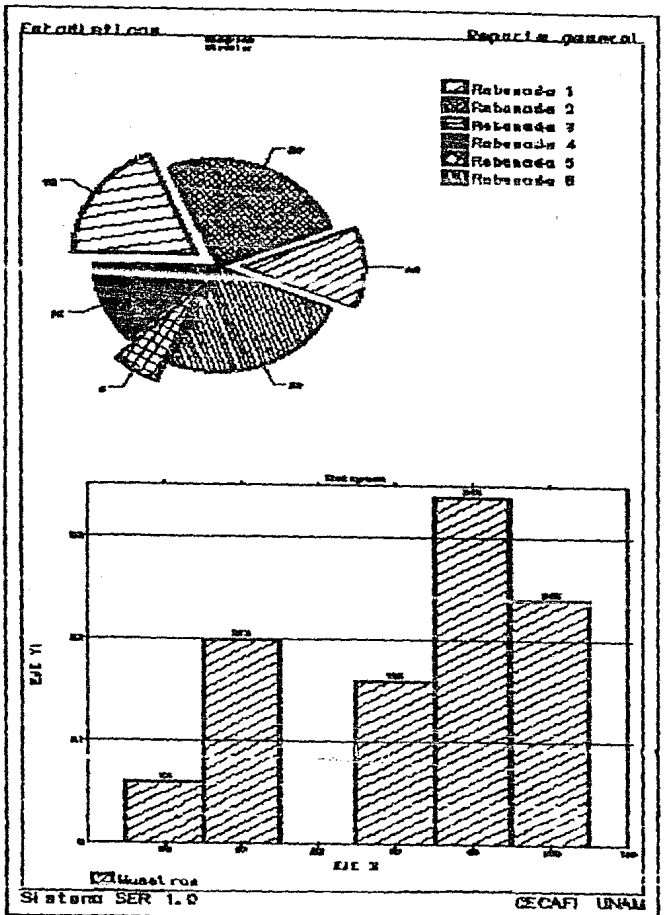

Estadísticas
Reporte general
Rebanada 1
Rebanada 2
Rebanada 3
Rebanada 4
Rebanada 5
Rebanada 6
EJE Y1
EJE X
Muestras
Sistema SER 1.0
CECAFI UNAM

Figura 6-22

```
PROGRAM Cuatro__Reportes

! Permite obtener cuatro reportes en una sola hoja de dibujo,
! mostrando el uso de las rutinas SER
IMPLICIT NONE

CHARACTER
1    Acotaciones(5)*40 /'Acotacion 1',
1                       'Acotacion 2',
1                       'Acotacion 3',
1                       'Acotacion 4',
1                       'Acotacion 5'/,
1    NombresGrupos (5) * 40 /'Grupo 1',
1                            'Grupo 2',
1                            'Grupo 3',
1                            'Grupo 4',
1                            'Grupo 5'/,
1    Titulos (3) * 40 /'Titulo 1',
1                     'Titulo 2',
1                     'Titulo 3'/,
1    Dispositivo * 40,
1    Rotulos(2) * 30,
1    Gruposder * 10
 INTEGER
1    Cont,
1    N,
1    NumDiv,
1    NumElem,
1    NumEsc,
1    NumGrupos,
1    TipoUnion(5) /1,2,3,4,1/,
1    TipoValorBarra
 REAL
1    Ancho,
1    Barras(20),
1    Linea(20),
1    Largo,
1    LineasRef(10),
1    Grupos(25)/1.0,2.0,3.0,4.0,5.0,
1               6.0,7.0,8.0,9.0,10.0,
1               10.0,20.0,30.0,40.0,50.0,
1               -1.0,-2.0,-3.0,-4.0,-5.0,
1               100.0,500.0,1000.0,40.0,1500.0/
 NumDiv = 4
 Ancho = 11.0
 Largo = 8.5
 Dispositivo = 'Graf.plt/pr'
 Rotulos(1) = 'Alumno'
 Rotulos(2) = 'Materia'

 !! Se inicializan las condiciones de la hoja
 CALL SER_DesplazaGrafica(1.0,1.0,1.0,1.0)
 CALL SER_DefineHoja(Largo,Ancho,NumDiv,Rotulos,Dispositivo)
```

```
 DO N=1, NumDiv
     CALL SER_AsignaYD('2,4')
     CALL SER_RotulosEje('X ','Eje X')
     CALL SER_RotulosEje('YI','Eje Y Izq')
     CALL SER_RotulosEje('YD','Eje Y Der')
     IF (N .EQ. 1) THEN
         WRITE(6,'(A)')'Valor sobre la barra'
         READ(5,'(A)')TipoValorBarra
         NumGrupos = 5
         NumElem = 3
         CALL SER_Barras(Grupos,NumGrupos,NumElem,Acotaciones,
1                        NombresGrupos,Titulos,TipoValorBarra)
     ELSEIF (N .EQ. 2) THEN
         NumGrupos = 5
         NumElem = 3
         CALL SER_Lineas(Grupos,NumGrupos,NumElem,Acotaciones,
1                              NombresGrupos,Titulos,TipoUnion)
     ELSEIF (N .EQ. 3) THEN
         WRITE(6,'(A)')'Valor sobre la barra'
         READ(5,'(A)')TipoValorBarra
         NumGrupos = 5
         NumElem = 3
         CALL SER_BarrasApiladas(Grupos,NumGrupos,NumElem,
1                                  Acotaciones,NombresGrupos,
1                                     Titulos,TipoValorBarra)
     ELSEIF (N .EQ. 4) THEN
         WRITE(6,'(A)')'Valor sobre la barra'
         READ(5,'(A)')TipoValorBarra
         WRITE(6,'(A)')'Numero de escalas'
         READ(5,'(A)')NumEsc
         NumElem = 3

         !!Se asignan los valores para barras y lineas
         DO Cont=1, NumElem
             Barras(cont) = Grupos(Cont)
             Linea(Cont) = Grupos(Cont+NumElem)
         ENDDO
         CALL SER_BarrasLinea(Barras,Linea,NumElem,
1                     Acotaciones,NombresGrupos,Titulos,
1                       NumEsc,TipoValorbarra,TipoUnion)
     ENDIF
 ENDDO
 CALL SER_TerminaHoja
 END
```

Listado 7-III

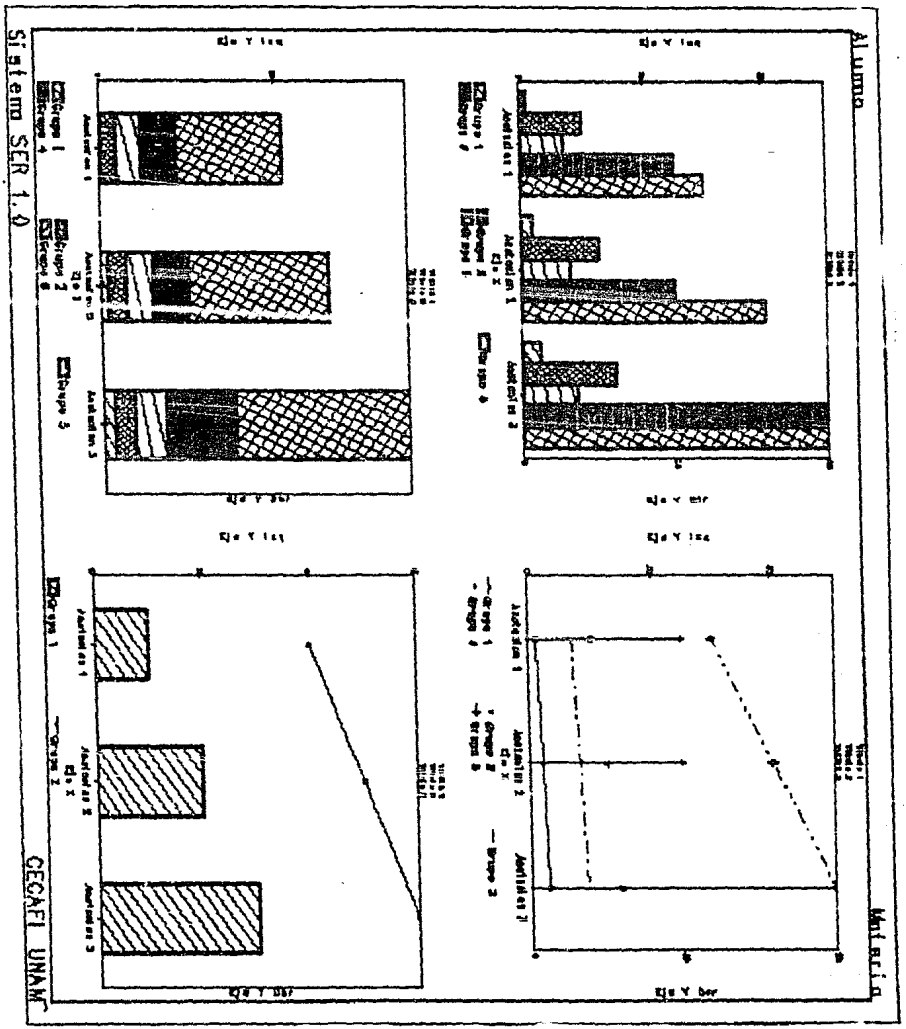


Figura 6-23

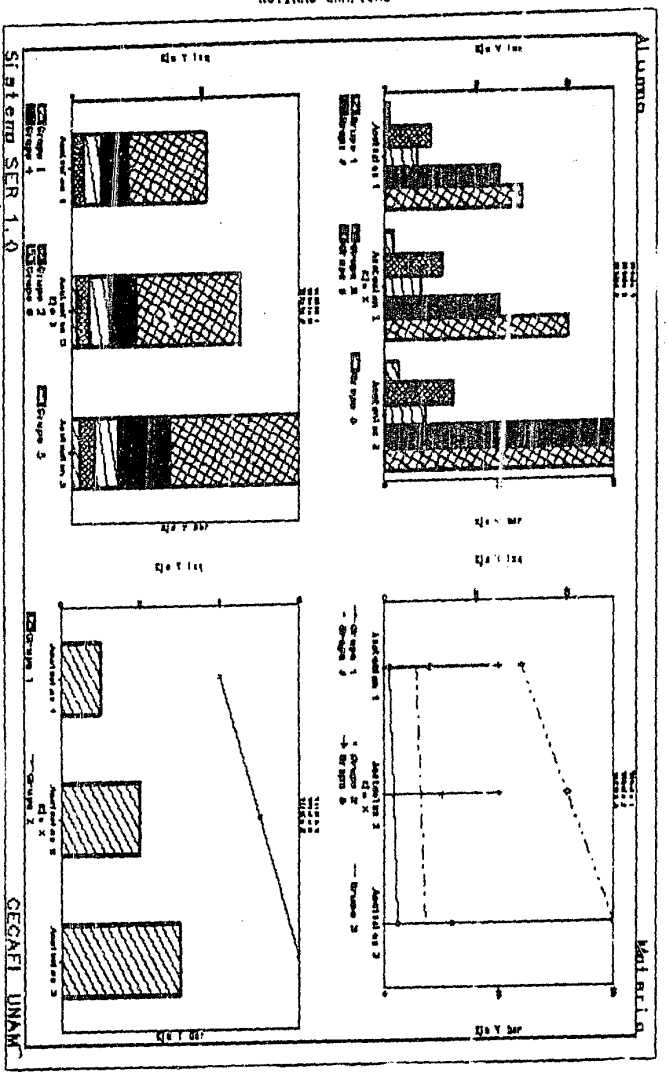


Figura 6-23

CAPITULO 7

SISTEMA INTERACTIVO

En este capítulo se tratará el tema relacionado con el Sistema Interactivo, uno de los productos finales del proyecto de esta tesis, enfocado a la producción en línea de gráficas. Para facilitar su comprensión, el capítulo se dividió en dos secciones. La primera gira en torno de las características externas, es decir, funcionales del sistema, mientras que la segunda se ocupa de los aspectos técnicos de la estructura del mismo.

7.1 CARACTERISTICAS EXTERNAS DEL SISTEMA.

En este tema se mostrará cómo se aplicaron los principios de los Factores Humanos planteados previamente en el capítulo 3. De igual forma, es de suma importancia tener en mente los lineamientos establecidos en el capítulo 4, pues fué ese planteamiento inicial de la definición del sistema interactivo el que se tomó como base para la realización práctica del sistema. Por este motivo se recomienda consultar a consciencia ambos capítulos.

7.1.1 Introducción.

El Sistema Interactivo es un producto final del proyecto de tesis. Tiene el objetivo de proporcionar una herramienta sencilla para la producción en línea de gráficas. Con este sistema los usuarios, por ejemplo los alumnos de la Facultad de Ingeniería de la UNAM, pueden obtener las gráficas que se requieren en el desarrollo de sus trabajos académicos, sin necesidad de elaborar un programa específico en la computadora. De esta forma, se ahorra tiempo y se simplifica la labor de los alumnos, además de mejorar la calidad de los trabajos.

En la figura 7-1 se muestran las pantallas por los cuales transita el usuario, desde el inicio del sistema, pasando por la captura de las características de una gráfica del tipo X vs Y, hasta el punto final de producción del dibujo en la impresora. Recordando lo expuesto en el capítulo 4, se notará que sólo se muestran las pantallas de un tipo de gráfica (X vs Y) y no de los 14 tipos existentes. El resto de las 14 gráficas son mas o menos similares, y por no perder la visión del objetivo central del capítulo, se omitieron. Si se desea consultar a detalle todos los tipos de gráficas, se recomienda revisar el manual de usuario del sistema [17]. De igual forma, esporádicamente se muestran las ayudas y directorios en algunas pantallas. El lector no debe pensar que estas funciones existen exclusivamente para esas

pantallas; por el contrario, las ayudas y directorios generalmente están a disposición del usuario a lo largo de todo el sistema.

A continuación se describen someramente las opciones que integran cada menú y el método de navegación por estas opciones. De igual forma se tratan los puntos terminales como son las pantallas de captura de las gráficas, la captura de tablas de datos y la impresión de gráficas.

7.1.2 Descripción Funcional.

En un principio aparece una pantalla de presentación y posteriormente la primera pantalla denominada menú principal. En ella existen cuatro opciones, siendo las dos primeras las opciones siempre presentes. La tercera, se despliega siempre y cuando existan tablas y la cuarta opción aparece cuando existen gráficas definidas. Las cuatro opciones son las siguientes:

1) **Introducir y Editar Tablas de Datos.**- Con esta opción se pueden crear, modificar o borrar tablas de datos. Estas tablas se emplean para integrar los datos que se dibujan en las gráficas.

2) **Usar Bitacoras de Sesiones de Trabajo.**- Esta modalidad permite guardar en disco todas las tablas y gráficas definidas de una sesión de trabajo, para usarlas en posteriores sesiones. De igual manera, con esta opción se puede traer de disco la bitácora de una sesión previamente almacenada.

3) **Seleccionar Gráficas.**- Es en esta sección donde las tablas creadas con la primera opción del menú principal, se asocian a uno de los 14 tipos de gráficas existentes, dando por resultado lo que el sistema denomina una gráfica definida o dada de alta.

4) **Imprimir Gráficas.**- Con esta opción el usuario puede imprimir las gráficas definidas en la opción anterior. Además, el usuario indica el formato de impresión, así como las dimensiones físicas de la hoja.

A continuación se describen brevemente las opciones que integran la primera opción del menú principal.

1) **Dar de Alta una Tabla de Datos.**- Al seleccionar esta opción deben especificarse el nombre de la tabla a dar de alta y el tipo (alfanumérico o numérico). Después aparece una ventana de captura de tablas con la cual se introducen todos los datos de la tabla.

2) **Evaluar una Función Matemática No Paramétrica.**- Con esta opción se puede evaluar una función del tipo Y = F(X). Al seleccionarla aparece una pantalla de captura donde se introduce una función, la cual debe seguir la sintaxis de una función en FORTRAN 77. Se deben especificar el intervalo de evaluación así como el número de puntos. Los puntos de la función evaluada se depositan en dos tablas indicadas por el usuario.

3) **Evaluar una Función Matemática Paramétrica.-** Esta opción es esencialmente idéntica a la anterior, excepto que la forma de la función es Y = F(T); X = G(T).

4) **Llenar una Tabla con un Archivo.-** Con esta modalidad el usuario puede proporcionar los datos de una tabla con los datos (alfanuméricos o numéricos) que tenga un archivo. El archivo debe haberse creado previamente con un programa de aplicación o con el editor de VAX.

5) **Modificar una Tabla de Datos.-** Una vez indicado el nombre de la tabla a modificar se abre una ventana donde se muestran los datos que contiene la tabla. El usuario puede entonces modificar estos datos.

6) **Dar de baja una Tabla.-** Con esta opción el usuario puede eliminar tablas que ya no desea. Debe especificar el nombre de la tabla que se va a dar de baja.

Las opciones 5 y 6 se muestran solamente si existe al menos una tabla. De esta manera se evita que el usuario seleccione opciones que de antemano se sabe que no se pueden operar. Las opciones de la segunda opción del menú principal serán el punto a tratar a continuación.

1) **Guardar en Disco la Bitácora de Trabajo.-** Con esta opción se permite guardar en un archivo las tablas y las gráficas definidas en la sesión de trabajo. El usuario debe especificar el nombre del archivo donde se guardará la bitácora.

2) **Traer de Disco una Bitácora de Trabajo.-** Esta opción es el complemento de la anterior pues con ella se pueden traer todas las tablas y gráficas definidas de otra sesion de trabajo. El usuario debe especificar el nombre del archivo donde se guardó la bitácora.

Ahora se expone la descripción de la tercera opción del menú principal.

1) **Dar de Alta una Gráfica.-** Con esta opción el usuario debe especificar el nombre de la gráfica que desea dar de alta y el tipo. Para el tipo de la gráfica se muestra un catálogo de 14 tipos de gráficas diferentes divididos en dos pantallas. Una vez proporcionados ambos datos, el usuario procede a la captura de la información de la gráfica que desea (vease la figura 7-1 para la gráfica X vs Y). En las formas de captura se introducen datos como los nombres de las tablas creadas con la primera opción del menú principal, datos particulares de la forma de la gráfica, como el tipo de linea que se usará para conectar a los puntos, etc. Una vez hecho esto, la gráfica ha quedado definida o dada de alta.

2) **Modificar una Gráfica Existente.-** Esta modalidad se emplea para corregir los datos que se proporcionaron en las formas de captura. El usuario debe especificar el nombre de la gráfica e inmediatamente se desplegarán las formas de captura para que corrija los datos erróneos.

3) **Dar de Baja una Gráfica.-** Es una opción con la cual se pueden eliminar gráficas que ya no se desea qué estén definidas. El usuario especifica el nombre de la gráfica y ésta es dada de baja.

Las dos opciones anteriores no aparecen en el catálogo de opciones del menú cuando no se ha dado de alta ninguna gráfica, por este motivo cuando se selecciona la tercera opción del menú principal y todavía no hay gráficas definidas, el sistema ejecuta inmediatamente la opción de alta de una gráfica sin mostrar el menú de opciones.

Por último cuando se selecciona la cuarta opción del menú principal, se muestra una forma de captura donde el usuario especifica el número y los nombres de las gráficas que se van a dibujar, los títulos que rotularán a la hoja, las dimensiones de la hoja incluyendo los márgenes, el número de copias y si desea imprimir el archivo de puntos o sólo generarlo. Posteriormente el sistema elabora el dibujo y lo envía a la impresora PRINTONIX 600, siempre y cuando haya indicado que desea imprimirse el archivo de puntos. En caso de que haya seleccionado que solamente se genere el archivo de puntos, este dibujo se podrá imprimir posteriormente, en la impresora gráfica DECWRITER IV o la impresora graficadora EPSON.

A lo largo de todo el sistema dos elementos asisten al usuario en la operación del mismo. Estos son las teclas funcionales y el desplegado del estado del sistema. El desplegado del estado del sistema consiste en indicarle al usuario, en el primer renglon del video de la terminal, qué parte del programa se está ejecutando. Por otro lado, las teclas funcionales son teclas que al oprimirlas ejecutan una función en particular; así por ejemplo, cuando el usuario oprime la tecla "?", el sistema despliega un texto de ayuda que orienta al usuario en la operación del sistema, (vease la figura 7-1). Las teclas funcionales activas se muestran en el último renglón del video.

Una vez establecidos los elementos funcionales del sistema, lo que procede es detallar la forma en que los principios de los factores humanos se aplicaron a la elaboración del sistema interactivo.

7.1.3 Aplicación De Los Principios De Los Factores Humanos.

A lo largo de este punto se detallará la forma en que fueron aplicados cada uno de los principios de los Factores Humanos en la elaboración del Sistema Interactivo. Este desarrollo puede emplearse como un caso de estudio que complementa al capítulo 3 donde simplemente se expone la teoría.

7.1.3.1 Primer Principio.

DESPLEGADO DEL ESTADO DEL SISTEMA.

El desplegado del estado del sistema se realizó a través del Manejador de Anidamientos, insertando en el primer renglón del video el texto correspondiente a la opción del menú seleccionado. Un aspecto importante que se cuidó a lo largo de todo el sistema fué que estuvieran a disposición del usuario solamente

las funciones que se pudieran ejecutar en un estado específico del sistema. Es decir, sólo se desplegaban las opciones del menú que podían seleccionarse, las teclas funcionales que se pudieran usar o los campos que se pudieran capturar, para la opción que se había seleccionado. Esto último se realizó con la capacidad de campos ocultos y campos ocultos anidados del Manejador de Captura.

Por otro lado, se emplearon dos técnicas extras de retroalimentación además del desplegado del estado del sistema, estas fueron la de orden recibida y de comando ejecutado. La técnica de orden recibida consiste en parpadear el texto de las teclas funcionales de ayuda y directorios, cuando éstas se comandan por el usuario. La técnica de comando ejecutado se empleó a lo largo de todo el sistema para informar al usuario que la opción que ordenó se ha completado, tal es el caso de dar de baja una tabla, dar de baja una gráfica, traer una bitácora, etc., donde el sistema informa cuando la tabla o la gráfica se ha borrado y cuando la bitácora ya se ha cargado en memoria. Esto evita un error que comunmente sucede en algunos sistemas cuando el usuario indica un comando y la computadora empieza a ejecutarlo sin notificar al usuario. Después de unos segundos de iniciada la ejecución, el usuario duda si realmente la computadora recibió el comando o sigue esperando datos, entonces el usuario intenta ejecutarlo nuevamente oprimiendo teclas a diestra y siniestra, para que el sistema responda. Esos datos extras escritos por el usuario, pueden provocar resultados impredecibles cuando la computadora termina de ejecutar la acción que se comandó.

DESPLEGADO Y EDICION DE LOS DATOS.

El desplegado y edición de los datos se hizo a través del Manejador de Estructuras para las tablas de datos y del Manejador de Captura para las formas de captura de gráficas, funciones paramétricas, no paramétricas y para la impresión de gráficas.

VALIDACION.

La validación se realizó para los datos de las Tablas, por el Manejador de Estructuras, para las formas de captura por medio del Manejador de Captura con la modalidad de validación de campos, y finalmente las validaciones adicionales se realizaron por el propio Sistema Interactivo dentro de sus rutinas.

RECUPERACION DEL ERROR.

Un aspecto importante que siempre se mantuvo en mente fué la de proporcionar al usuario un mecanismo para recuperarse de los dos tipos de errores. Los errores de control sucitados por transitar a estados no deseados, se resolvieron con la tecla funcional ESC que permite regresar al menú anterior. Los errores de datos provocados por errores de escritura al proporcionar un dato, se trabajaron en forma particular por los Manejadores de Captura y de Estructuras con las teclas de edición. Por otro lado, en la modalidad de diálogos (pregunta-respuesta) se resolvieron los dos errores también con la tecla ESC. Esta tecla provoca que la pregunta que se acaba de contestar se envíe nuevamente y así el usuario pueda reescribir la respuesta satisfactoriamente o cancelar la pregunta.

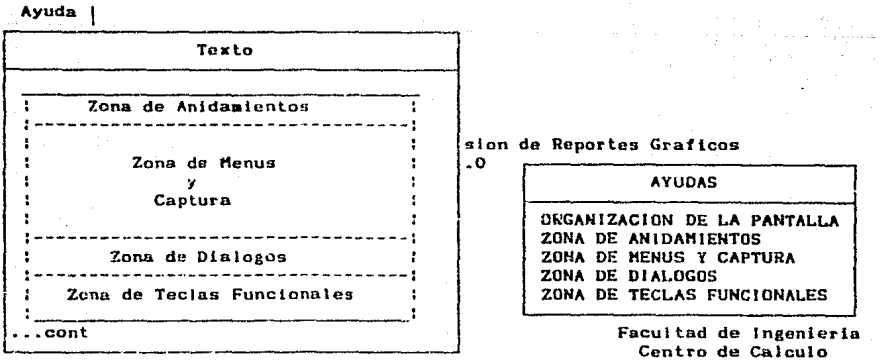
SISTEMA INTERACTIVO

Sistema Interactivo para la Emision de Reportes Graficos
SER Ver 1.0

U N A M
Facultad de Ingenieria
Centro de Calculo

<Esc> Salir ? Ayuda <Return> Comenzar

Figura 7-1a
Pantalla de presentación del sistema.



Msj> Seleccione una opcion
Msj> Seleccione una opcion

<ESC> Salir <RET> Continuar

Figura 7-1b
Despliegue de ayuda en la presentación.

```
Principal |
------------------------------------------------------------
                    Menu Principal

            Introducir y Editar Tablas de Datos

            Usar Bitacoras de Sesiones de Trabajo

            Seleccionar una Grafica

            Imprimir una Grafica
------------------------------------------------------------
Msj> Seleccione una opcion
Msj> Seleccione una opcion

<Esc> Salir ? Ayuda
```

Figura 7-1c
Menú principal.

```
Principal | Ayuda |
+-----------------------------------------+
|                 Texto                   | cipal
|-----------------------------------------|
|        Operacion de los Menus           |
|                                         |
|    La forma de operar los menus es      |
| con las flechas "hacia arriba" y "hacia | ablas de Datos
| abajo".                                 |
|                                         |
|    La seleccion de una opcion se        | iones de Trabajo
| realiza de la siguiente forma:          |
|                                         |        AYUDAS
|    1.- Aparece el mensaje               | ca  OPERACION DE LOS MENUS
|                                         |     TABLAS DE DATOS
|    Msj> Seleccione una Opcion           |     BITACORAS
|                                         |     SELECCION GRAFICA
| ...cont                                 |     IMPRESION GRAFICA
+-----------------------------------------+
Msj> Seleccione una opcion
Msj> Seleccione una opcion

<ESC> Salir <RET> Continuar
```

Figura 7-1d
Ayuda en el menú principal.

SISTEMA INTERACTIVO

Principal | Edicion de Tablas |

Tipo de Operacion

Dar de Alta una Tabla de Datos

Evaluar una Funcion Matem. No Parametrica

Evaluar una Funcion Matem. Parametrica

Llenar una Tabla con un Archivo

Modificar una Tabla de Datos

Dar de Baja una Tabla de Datos

Msj> Seleccione una opcion
Msj> Seleccione una opcion

<Esc> Salir ? Ayuda <CTRL/T> Directorio de Tablas

Figura 7-1e
Menú de opciones de Introducir y Editar Tablas de Datos.

Principal | Edicion de Tablas | Alta de una Tabla |

Tipo de Operacion

Dar de Alta una Tabla de Da

Evaluar una Funcion Matem.

Evaluar una Funcion Matem.

Llenar una Tabla con un Arc

Modificar una Tabla de Dato

Dar de Baja una Tabla de Da

Tabla: CIUDADES

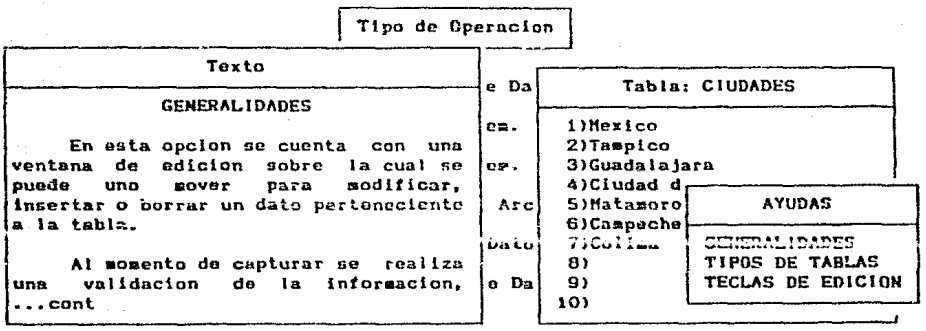
1)Mexico
2)Tampico
3)Guadalajara
4)Ciudad del Carmen
5)Matamoros
6)Campeche
7)Colima
8)
9)
10)

Lee> Nombre de la tabla a dar de alta ?Ciudades
Lee> Tipo de la tabla ([N] Numerica o [A] Alfanumerica) ?a

<ESC> Terminar <CTRL/A> Abortar <CTRL/T> Dir. Tablas <PF1> Sig. Teclas

Figura 7-1f
Editor en Alta de una Tabla.

Principal | Edicion de Tablas | Alta de una Tabla | Ayuda |

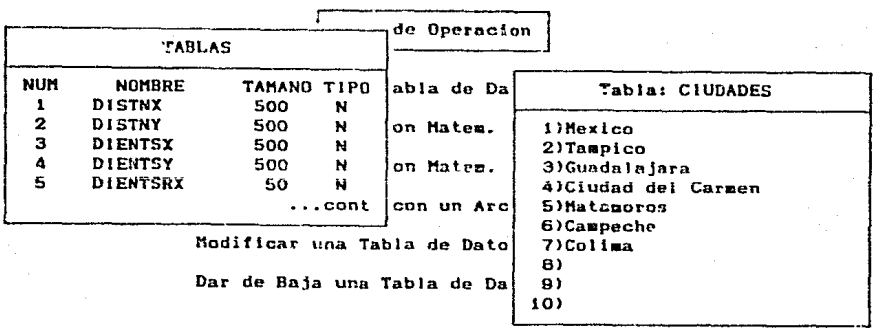


Lze> Tipo de la tabla ([N] Numerica o [A] Alfanumerica) ?a
Msj> Seleccione una opcion

<ESC> Salir <RET> Continuar

Figura 7-1g
Despliegue de ayuda del Editor de Tablas.

... Edicion de Tablas | Alta de una Tabla | Directorio de Tablas |



Modificar una Tabla de Dato
Dar de Baja una Tabla de Da
abla de Da
on Matem.
on Matem.
con un Arc

Msj> Seleccione una opcion
Msj> Seleccione una opcion

<RET> Continuar <ESC> Terminar

Figura 7-1h
Despliegue del Directorio de Tablas.

```
                    SISTEMA INTERACTIVO

 Principal | Edicion de Tablas | Evaluacion Funcion No Parametrica |

          y =  [dnormal(0.0,1.0,x)                                  ]

                  Valor Inicial de X:   [-3.5       ]
                  Valor Final   de X:   [3.5        ]

                    Numero de Puntos:   [500]

          Nombre de la Tabla para depositar
                        los valores de X:  [DISTNX     ]
                        los valores de Y:  [DISTNY     ]

-------------------------------------------------------------------------------
Msj> Comenzar a capturar
Msj> Comenzar a capturar

 <ESC> Evaluar Funcion <CTRL/A> Abortar ? Ayuda <CTRL/T> Dir. Tablas
```

Figura 7-11
Forma de Captura de una Función No Paramétrica.

```
 Principal | Edicion de Tablas | Evaluacion Funcion Parametrica |

          y =  [10 * sind (t) + 5                                   ]

          x =  [10 * cosd (t) + 8                                   ]

                  Valor Inicial del parametro t:   [0         ]
                  Valor Final   del parametro t:   [360       ]

                              Numero de Puntos:    [500]

          Nombre de la Tabla donde se depositaran
                        los valores de X:  [CIRCULOX   ]
                        los valores de Y:  [CIRCULOY   ]

-------------------------------------------------------------------------------
Msj> Campo requerido
Msj> Campo requerido

 <ESC> Evaluar Funcion <CTRL/A> Abortar ? Ayuda <CTRL/T> Dir. Tablas
```

Figura 7-1j
Forma de Captura de una Función Paramétrica.

```
Principal | Uso Bitacoras |
```

```
                    Tipo de Accion

                Guardar en Disco la Bitacora de Trabajo

                Traer de Disco una Bitacora de Trabajo

Msj> Seleccione una opcion
Msj> Seleccione una opcion

 <Esc> Salir ? Ayuda <CTRL/B> Directorio de Bitacoras
```

Figura 7-1k
Menú de opciones de Usar Bitácoras de Sesiones de Trabajo.

```
Principal | Uso Bitacoras | Directorio Bitacoras |
```

```
                    Tipo de Accion

                                  Directorio de Bitacoras
Guardar en Disco la Bitacor       BIT.BTC;
                                  BIT2.BTC
                                  BITAUX.B
                                  BITAUX2.
Traer de Disco una Bitacora                        ... Fin

Msj> Seleccione una opcion
Msj> Seleccione una opcion

 <RET> Continuar <ESC> Terminar
```

Figura 7-1l
Despliegue del Directorio de Bitacoras.

SISTEMA INTERACTIVO

Principal | Seleccion de una Grafica |

Tipo de Operacion

Dar de Alta una Grafica

Modificar una Grafica Existente

Dar de Baja una Grafica

Msj> Seleccione una opcion
Msj> Seleccione una opcion

<Esc> Salir ? Ayuda <CTRL/G> Directorio de Graficas

Figura 7-1m
Menú de opciones de Seleccionar Gráficas.

Principal | Seleccion de una Grafica | Alta de una Grafica |

Tipos de Graficas

Barras
Lineas
X vs Y
Diagrama Circular (Pie)
Histograma
Poligonos de Frecuencia
Barras Apiladas
Graficas de Error
Combinacion de las Anteriores

Lee> Nombre de la grafica a dar de alta ?dnormal
Msj> Seleccione una opcion

<Esc> Salir ? Ayuda

Figura 7-1n
Menú de tipos de gráficas simples en Alta de una Gráfica.

... Alta de una Grafica | Combinacion de Graficas |

Graficas Combinadas

Barras - Lineas

Barras - X vs Y

Lineas - X vs Y

Histograma - Poligono de Frecuencia

Histograma - X vs Y

Poligono de Frecuencia - X vs Y

Msj> Seleccione una opcion
Msj> Seleccione una opcion

<Esc> Salir ? Ayuda

Figura 7-1o
Menu de tipos de gráficas combinadas en Alta de una Gráfica.

... Seleccion de una Grafica | Alta de una Grafica | X vs Y | Forma 1 |
Numero de funciones (1,2,3,4,5) : [3]

Numero de Funcion	Tabla X	Tabla Y	Nombre de la Funcion	Tipo de Union de los Puntos
1	[DISTNX]	[DISTNY]	[Distribucion Normal]	[1]
2	[DIENTSX]	[DIENTESY]	[Diente de Sierra]	[3]
3	[DIENTSRX]	[DIENTSRY]	[Diente de Sierra Rect.]	[2]

(1) lineas
(2) marcadores
(3) marc. - lineas
(4) marc conec a y = 0

Titulos Generales de las Funciones
[Funciones de Biblioteca]
[del Sistema Interactivo definidas]
[para alumnos.]

Msj> Seleccione una opcion
Msj> Comenzar a capturar

<CTRL/T> Directorio de Tablas <PF1> Siguientes Teclas

Figura 7-1p
Primera Forma de Captura para X vs Y.

```
... Seleccion de una Grafica | Alta de una Grafica | X vs Y | Forma 2 |
 Nombre del eje X :   [Frecuencia                                      ]
     Escala eje X :      (1) Automatica  (2) Manual                [1]

      Acotamiento :      (1) No Logaritmico (2) Logaritmico        [1]
 Nombre del eje Y :   [f(x)                                         ]
     Escala eje Y :      (1) Automatica  (2) Manual                [2]
     Y Inicial :  [-10        ]  Y Final   :  [12         ]
      Acotamiento :      (1) No Logaritmico (2) Logaritmico        [2]
 Otro eje vertical de referencia?   (si/no) [SI]
 Nombre del eje Y -derecho-:   [f'(x)                                                ]
     Escala eje Y :      (1) Automatica  (2) Manual                [1]

      Acotamiento :      (1) No Logaritmico (2) Logaritmico        [1]
 Asociacion graficas-ejes   (I) -> eje Y izquierdo   (D) -> eje Y derecho
          1 [I]    2 [d]    3 [d]    4 [D]    5 [D]
 Escala eje X con respecto a eje Y :    (1) Sin Relacion (2) Iguales  [1]
```

```
Msj> Leyendo forma de captura
Msj> Comenzar a capturar

 <CTRL/F> Forma Anterior <CTRL/T> Directorio de Tablas <PF1> Siguientes Teclas
```

Figura 7-1q
Segunda Forma de Captura para X vs Y.

```
... Seleccion de una Grafica | Alta de una Grafica | X vs Y | Forma 3 |


  Al recuadro se le incluye  (1,2 o 3) :   [3]
       (1) Nada
       (2) Rejilla (de guiones,  puntos o lineas)
       (3) Lineas de referencia  en los ejes X y Y

  Lineas de Referencia en el eje X :
     [-10.34     ] [2.56       ] [0          ] [6          ] [4          ]

  Lineas de Referencia en el eje Y :
     [2          ] [4.62       ] [           ] [           ] [           ]
```

```
Msj> Leyendo forma de captura
Msj> Comenzar a capturar

 <ESC> Fin Captura <CTRL/A> Abortar Captura ? Ayuda <PF1> Siguientes Teclas
```

Figura 7-1r
Tercera Forma de Captura para X vs Y.

```
Principal | Impresion de Graficas |
Numero de Graficas (1, 2, 3, 4):     [3]
Nombre de la Grafica 1 : [NORMAL      ]  Nombre de la Grafica 2 : [DIENTES    ]
Nombre de la Grafica 3 : [DIENTESR   ]

      Titulo Izquierdo de la Hoja :   [Prueba TESIS                    ]
      Titulo  Derecho  de la Hoja :   [Proposito Examen Profesional   ]

 Dimensiones de la Hoja (en cm)   (1, 2, 3):   [3]
                    (1) Hoja Carta      (27.94 x 21.59 cm)
                    (2) Hoja Impresora (33.02 x 21.59 cm)
                    (3) Dimensiones dadas por el usuario
        Ancho (horizontal):   [21.59]         Largo (vertical):   [27.94]

 Margenes (en cm) :       Inferior :  [2.0  ]     Superior:  [3.0  ]
                          Izquierdo:  [2.5  ]     Derecho :  [2.0  ]
 El dibujo (1, 2):    [1]
     (1) Se imprime (y se borra)
     (2) No se imprime (y no se borra)        Numero de Copias :     [1]
--------------------------------------------------------------------------------
Msj> Seleccione una opcion
Msj> Comenzar a capturar

 <Esc> Fin Captura <CTRL/A> Abortar ? Ayuda <CTRL/G> Dir. Graficas
```

Figura 7-1s
Forma de Captura para Imprimir Gráficas.

```
 ________________________________________a |
|              Texto                     |
|----------------------------------------|ombre de la Grafica 2 : [DIENTES    ]
|     Margen             Margen          |
|  <-->Izquierdo         Derecho<-->     |
| :o----------------------------o:       |rueba T ________________________
| :o            ^               o:       |roposit|         AYUDAS         |
| :o            : Margen Superior o:     |       |------------------------|
| :o            V               o:       | 3):   | MOVIMIENTO EN LA FORMA |
| :o      _________________     o:       |7.94 x | CAPTURA DE DATOS       |
| :o      :                :    o:       |3.02 x | TECLAS FUNCIONALES     |
| :o      :                :    o:       | por el| TITULOS DE LA HOJA     |
| :o      -----------------     o:       |      L| DIMENSIONES DE LA HOJA |
| :o            ^               o:       |       | MARGENES DE LA HOJA    |
| :o            : Margen Inferior o:     |       | DIVISIONES DE LA HOJA  |
| :o            V               o:       |   ]   | CARACTERES ESPECIALES  |
| :o----------------------------o:       |   ]   |________________________|
|                                        |
| ....fin                                |   Numero de Copias :     [1]
|________________________________________|____________________________________
Msj> Comenzar a capturar
Msj> Seleccione una opcion

 <ESC> Salir <RET> Continuar
```

Figura 7-1t
Despliegue de Ayuda para la Forma de Impresión de Gráficas.

SISTEMA INTERACTIVO

Muchos sistemas rara vez se preocupan por la recuperación antes de ejecutar una operación irreversible. Tal es el caso de la opción de dar de baja una tabla de datos o abortar la captura de una forma, en donde las consecuencias de la ejecución errónea de estas opciones es evidente. Cuando el usuario selecciona opciones de este tipo, el sistema le avisa de las consecuencias de la acción que va a tomar y solicita una confirmación. Esto evita que por error se seleccione una opción que no se puede corregir sencillamente, como lo es borrar indefinidamente una tabla de datos.

FUNCION DE AYUDA.

Los textos que asisten al usuario en la operación del sistema se manipularon a través del Manejador de Ayuda. En todo momento se tuvo presente que las ayudas también estuvieran relacionadas con el estado del sistema, es decir, que existieran ayudas específicas para cada menú de opciones y de hecho, que las ayudas variaran con respecto al número de opciones que se desplegaban en un mismo menú. Esto se hizo con la modalidad de tópicos del Manejador de Ayuda.

7.1.3.2 Segundo Principio.

FACIL DE ENTENDER.

Para que el sistema fuera fácil de entender se eligió un esquema de Menús de opciones. El número de opciones de un menú varía dependiendo si estas opciones se pueden seleccionar o no. Por ejemplo, para el menú principal la opción 4 se despliega sólo si hay al menos una tabla dada de alta y la quinta opción sólo si se dió de alta al menos una gráfica. Las opciones se ordenan en dos bloques, al principio se colocan las opciones que siempre se muestran y las que su despliegue está condicionado después. A su vez, cada bloque se ordena de acuerdo a las opciones más probables de usar primero y las menos probables al final. Todo esto se logra gracias al Manejador de Menús con la modalidad de opciones variables.

También se emplearon teclas funcionales para operaciones donde se requiere pronta acción. Con la finalidad de que el usuario no tenga que memorizar las funciones de cada tecla, a cada una se le acompaña con un texto que indica en forma breve la acción que realiza. Así por ejemplo, "? Ayuda", indica que la tecla de interrogación despliega un texto de ayuda para asistir en la operación del sistema. Un factor importante que se mantiene consistente en todo el sistema es que jamás a una misma tecla funcional se le asigna más de una función. Si se opera el sistema, aparentemente para la tecla <ESC> esto no es cierto, pero lo que en realidad sucede, es que a esta tecla se le cambia el texto asociado en determinados estados del sistema, para aclarar más su función y es por esto que puede parecer que tiene funciones distintas cuando no lo es. El manejo de las teclas, así como el parpadeo de retroalimentación, se realizó por medio del Manejador de Teclas Funcionales. Las teclas se listan a continuación:

1) **<ESC>** Termina la operación que actualmente está realizando y se regresa al punto de donde proviene. Al oprimir esta tecla en captura de formas, el sistema muestra la siguiente forma si la hay, y si es la última, termina la captura de formas, regresando al menú anterior.

2) <CTRL/A> Aborta la captura que esté realizando y regresa al estado inmediatamente anterior. Esta tecla difiera de <ESC>, en que <CTRL/A> se puede emplear exclusivamente en la captura de Tablas y Gráficas e indica que no se quiere que los cambios realizados se actualicen, ya que se cometió un error.

3) ? Abre una ventana y despliega un texto de ayuda para asistir al usuario en la operación del sistema.

4) <CTRL/G> Abre una ventana y despliega el Directorio de Gráficas dadas de alta.

5) <CTRL/B> Abre una ventana y muestra el Directorio de Bitácoras.

6) <CTRL/T> Abre una ventana y despliega el Directorio de Tablas dadas de alta.

7) <PFi> Despliega el siguiente bloque de teclas funcionales. Si es el último bloque, muestra el primero.

8) <CTRL/I> Inserta una línea en la captura de Tablas.

9) <CTRL/E> Elimina una línea en la captura de Tablas.

10) <CTRL/F> Regresa a la Forma anterior, si la hay, en la captura de Formas.

Cuando el sistema solicita que el usuario responda preguntas como el nombre de la bitácora por ejemplo, o la confirmación de una acción de alto riesgo; o bien, cuando el sistema muestra diagnósticos de error y mensajes de retroalimentación, se emplea un esquema de pregunta-respuesta por medio del Manejador de Diálogos. Esto permite que el sistema sea fácil de entender y resuelva todas las situaciones en que los menús de opciones son inapropiados.

Toda salida a través del Manejador de Diálogos se caracteriza de acuerdo al tipo de mensaje y éste a su vez, está en función del siguiente esquema:

1) "Msj> " y la campana de la terminal suena una vez, para cualquier mensaje de aviso. Esto se usa para informar al usuario que el sistema ha completado una acción satisfactoriamente (retroalimentación).

2) "Err> " y la campana de la terminal suena dos veces, para enviar diagnósticos de error. Una vez enviado el diagnóstico, el sistema espera a que el usuario oprima una tecla. Tanto en este esquema como en el anterior, la campana está diseñada para llamar la atención a la zona de diálogos, pues los mensajes y los diagnósticos de error suceden cuando el usuario tiene normalmente su mirada en otra zona de la pantalla. El esquema de esperar a que se oprima una tecla, se diseñó pensando en que las consecuencias de un error ameritan que el usuario lea a consciencia el diagnóstico y así tome las medidas pertinentes. La única forma de asegurar que el flujo de ejecución del sistema no borre el diagnóstico del error, por enviar mas mensajes a la zona de diálogos, es esperar hasta que el usuario oprima una tecla y así

asegurar que ha leído el mensaje.

3) "Lee> ?". Este esquema se usa para leer un dato y se caracteriza por finalizar con un signo de interrogación. Para auxiliar en el tipo de respuesta que espera, generalmente se encierra entre paréntesis una pequeña explicación.

En los tres puntos anteriores se procura que el texto sea lo más claro posible, indicando cuando es viable, qué tipo de acción se puede realizar para satisfacer la petición del sistema, ya sea en mensajes, diagnosticos de error o lectura de un dato.

Finalmente, con respecto a la amplitud del sistema siempre se mantiene a menos de 10 opciones por menú. De hecho para los 14 tipos de gráficas existentes, fué indispensable dividirlas en dos grupos (veanse las figuras 7-1n y 7-1o).

MEMORIA DE CORTO PLAZO.

En muchas ocasiones el usuario trae una bitácora de trabajo, modifica los datos de las tablas o de las gráficas, y finalmente desea guardar la bitácora con el mismo nombre. Al llegar a este punto, es muy probable que el usuario no recuerde con exactitud cuál es el nombre de la bitácora y deba desplegar el catálogo de bitácoras para proporcionar el nombre correcto. Si esta operación es muy frecuente, tal y como sucede en edicion de tablas o gráficas, desplegar constantemente los directorios puede producir ciertas distracciones molestas en la operación del sistema. Para evitar este tipo de interrupciones, el sistema coloca un nombre cada vez que solicita uno, ya sea en bitácoras, tablas o gráficas. De tal manera, que si el usuario está satisfecho con el nombre que el sistema propone, simplemente debe oprimir RETURN, si no lo está, debe borrar el nombre y escribir el que el usuario desea. El sistema predice el nombre de acuerdo al siguiente criterio: para bitácoras, el nombre que propone "en guardar", es el de la última bitácora traída o la ultima bitácora guardada, dependiendo de cual haya sido la última operación. Para "modificación de tablas", la última tabla dada de alta, la última modificada, o la última tabla llenada con un archivo, también dependiendo de cual haya sido la última operación y finalmente, para "modificación de gráficas", la última gráfica dada de alta o modificada, de acuerdo a la última operación llevada a cabo.

CONSISTENCIA Y ESTADARIZACION.

Uno de los puntos más importantes que todo sistema debe cuidar es la consistencia. Por este motivo esta rúbrica se vigiló estrechamente en el diseño del Sistema Interactivo. Esto se puede apreciar en las teclas funcionales, ya que cada tecla tiene asignada una sola función. También en diálogos existe consistencia, ya que los tres formatos se respetan rigurozamente y jamás se mezclan entre ellos. Con respecto a los menús, se procura emplear esquemas convencionales fáciles de seguir por el usuario. Tal es el caso del formato Altas, Bajas, Modificaciones, usado tanto para Tablas como para Gráficas.

Otro aspecto en que muchos sistemas fallan es que en la captura de determinados datos, el sistema no se espera a que el usuario oprima "RETURN". Esto provoca en el usuario confusión y cierto recelo al operar el sistema, pues

tenga que se comporte impredeciblemente al oprimir "RETURN" cuando no es necesario. El Sistema Interactivo es consistente al respecto, jamás transita a otro estado sin esperar a que el usuario oprima "RETURN." Esto a corto plazo, permite que el usuario adquiera más confianza y aprenda fácilmente a emplear el sistema.

7.1.3.3 Tercer Principio

MODELO INTERNO DEL USUARIO

Para resolver este punto, el sistema tiene un diseño con un esquema básico de Altas, Bajas y Modificaciones; emplea conceptos familiares como son Tablas de Datos y finalmente, a lo largo de todo el sistema se provee de preguntas con valores propuestos. Si el usuario está de acuerdo con estas respuestas simplemente oprime RETURN, de tal suerte, que le permite al usuario emplear el sistema conforme su modelo interno se va desarrollando y así, usar opciones más complicadas, de acuerdo a la confianza que el usuario sienta. Este mecanismo se hace patente en el Manejador de Captura con los campos ocultos y valores iniciales, y el Manejador de Diálogos con la prediccion de respuestas en las preguntas.

7.1.3.4 Cuarto Principio.

Para este efecto se empleó un flujo clásico: se proporcionan los datos, se selecciona el tipo de gráfica y se dibuja. Con respecto al manejo de los retardos, el sistema envía mensajes acompañados de una señal sonora para llamar la atención del usuario, provee mensajes en donde se indica que el sistema tardará en responder y además muestra el estado del sistema, por medio del renglón de anidamientos, para mantener informado al usuario en todo momento de la acción que está realizando el sistema.

7.1.3.5 Quinto Principio.

Con respecto al quinto principio de los Factores Humanos, se puede mencionar que sale de la competencia del sistema, pues la compra de mobiliario ajustable o la instalación de iluminación adecuada sobrepasa los objetivos del sistema y más bien debe orientarse a los proyectos particulares de la administración del Centro de Cálculo.

Con este punto se da por terminada la primera parte de este capítulo, relacionada con la estructura externa del Sistema Interactivo, para proceder con la descripción de la estructura interna.

7.2 CARACTERISTICAS INTERNAS DEL SISTEMA.

En esta sección se tratará el tema relacionado con la forma en que las características externas (funcionales) se llevaron a cabo. Debido a que no intenta ser un escrito exhaustivo, deja los detalles para un estudio más profundo del programa fuente. De hecho la idea principal de esta sección, es delinear la filosofía de programación, que caracteriza a cada uno de los módulos que integran al programa fuente. Para hacer más sencilla esta tarea se decidió agruparlos en ocho puntos principales, cada uno correspondiente a las secciones más importantes del programa: las características generales, las formas de captura, las teclas funcionales, la predicción de nombres, la impresión de gráficos, la evaluación de funciones, las ayudas y las rutinas de utilería.

7.2.1 Características Generales.

El programa está escrito en FORTRAN 77 empleando los principios de la programación estructurada, la programación modular y los estándares de programación. A continuación se detallan los aspectos más importantes relativos al manejo del control del programa.

El control del programa con respecto a los menús de opciones se llevó a cabo colocando cada menú en una subrutina, de tal forma que la transición entre un menú y otro, se redujo a simples llamadas a subrutinas (CALL) y el regreso al menú anterior, a retornos normales del tipo RETURN. Sin embargo, el control para regresar al estado anterior en las preguntas, para transitar a la forma anterior en la captura de formas (tercer opción del menú principal) y para manejar errores, se hizo a través de estados. Los estados son muy comunes en la realización de analizadores sintácticos (Scanners). Sin embargo, no es común verlos en programas para tomar acciones específicas, en función de los datos de entrada. Esto se hizo primordialmente por la complejidad del código estructurado que hubiera resultado, debido al manejo de muchas teclas funcionales. Pues tomar tan variadas acciones como son despiegado de ayuda y directorios, regresar al estado anterior y manejo de diversos tipos de errores, hubiera obligado a anidar excesivamente estructuras del tipo DO WHILE y seguramente al uso de banderas. Por otro lado, el uso de estados, además de simplificar la tarea, permite realizar modificaciones rápidamente sin necesidad de anidar nuevas estructuras. Seguramente se puede pensar que esto va en contra de la programación estructurada, sin embargo, se sacrificó el uso de estructuras estándar por claridad, la cual es el objetivo principal de la programamación estructurada. Para mayor información acerca de la construcción de diagramas de estados, se recomienda consultar la referencia [16]. En el listado 7-1 se muestra, a modo de ejemplo del uso de estados, un fragmento del código simplificado, de la subrutina encargada del manejo del menú principal.

```
CALL MAN_Inserta ('Principal')
CALL MAN_Despliega ()

Condicion = MAY_Inicia ('SER$AYUDAS:Principal.MAY')
Condicion = MMU_Despliega (Opciones,4,2,0,Titulo,1)

NumDes = 2
```

```
99      Continue
        !Estado 0

        DespliegaMenu = .TRUE.

        CALL MMU_Acepta (Opciones, NumDes, 63, '? Ayuda', 1, 1, Opcion)

        IF (Opcion .EQ. 1) THEN
            CALL EdicionTablas
        ELSE IF (Opcion .EQ. 2) THEN
            CALL UsoBitacoras(DespliegaMenu)
        ELSE IF (Opcion .EQ. 3) THEN
            CALL SelecGrafica(DespliegaMenu)
        ELSE IF (Opcion .EQ. 4) THEN
            CALL InpGraficas
        ELSE IF (Opcion .EQ. -1) THEN
            ! ?
            CALL MAY_DespliegaTopicos(NumDes + 1)
            GOTO 99
        ELSE IF (Opcion .EQ. 0) THEN
            ! <Esc>
            GOTO 1
        ENDIF

        Condicion = MMU_Despliega (Opciones, 4, NumDes, 0, Titulo, 1)
        GOTO 99

1       CONTINUE
        ! Estado 1  <Esc>

        IF (MES_HayVectores() .OR. MES_HayGraficas () ) THEN
            GOTO 6
        ELSE
            GOTO 8
        ENDIF


6       CONTINUE
        ! Estado 6  <Esc> y Catalogos con informacion

        CALL SalvarBit (Esc)

        IF (Esc) THEN
            GOTO 99
        !NOELSE
        ENDIF

        !! GOTO 8 si continua el estado 8 no se necesita

8       CONTINUE
        ! Estado 8
        !Final
```

Listado 7-1

Otro punto importante que se hace patente al usar el sistema, es que se intenta minimizar el número de cambios de pantalla. En muchas ocasiones se seleccionan opciones que no provocan cambios al contenido del video. En esos casos, el sistema detecta que no es necesario desplegar nuevamente el menú, ya que éste aún se encuentra presente en la pantalla. Este hecho ahorra tiempo, evita distracciones innecesarias al usuario y mejora el comportamiento global del sistema. Todo esto se logró gracias al uso de ventanas y de banderas que se pasan de una subrutina a otra, indicando si la subrutina en cuestión no ha modificado el contenido de la pantalla. A continuación se detallan los aspectos técnicos más importantes relativos al uso y manejo de las formas de captura.

7.2.2 Formas De Captura.

Las formas de captura se emplean para que el usuario introduzca la información específica de las gráficas que va a dibujar, para proporcionar los datos en el módulo de impresión y para introducir funciones matemáticas que desea dibujar. En relación a las formas de las gráficas, debido al número tan grande de formas que se tenían que elaborar, 14 gráficas por 2 formas en promedio por gráfica, fué necesario realizar un programa que facilitara la construcción de las formas, denominado DSCFRM. Este programa, una vez dado el nombre de la forma, obtiene una descripción para cada campo, indicando su longitud y su posición relativa en el registro. Esta información es sumamente útil para la inicialización de valores del Manejador de Captura y la extracción de valores del Manejador de Estructuras. En el listado 7-11 se muestra la descripción para la primera forma de la gráfica X vs Y.

```
: Numero de funciones (1,2,3,4,5) :: \E\
: Numero :
:   de   :                                                          :Tipo de Union:
: Funcion:  :Tabla X:    :Tabla Y:    :Nombre de la Funcion:       :de los Puntos:
   :1:     \AAAAAAAAAA\ \AAAAAAAAAA\ \CCCCCCCCCCCCCCCCCCCCCCCCCC\       \E\
   :2:     \AAAAAAAAAA\ \AAAAAAAAAA\ \CCCCCCCCCCCCCCCCCCCCCCCCCC\       \E\
   :3:     \AAAAAAAAAA\ \AAAAAAAAAA\ \CCCCCCCCCCCCCCCCCCCCCCCCCC\       \E\
   :4:     \AAAAAAAAAA\ \AAAAAAAAAA\ \CCCCCCCCCCCCCCCCCCCCCCCCCC\       \E\
   :5:     \AAAAAAAAAA\ \AAAAAAAAAA\ \CCCCCCCCCCCCCCCCCCCCCCCCCC\       \E\
                                                    :(1) lineas:
                                                    :(2) marcadores:
                                                    :(3) marc. - lineas:
                                                    :(4) marc conec a y = 0:
                  :Titulos Generales de las Funciones:
               \CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC\
               \CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC\
               \CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC\

                              Analisis de los Campos

  Nomenclatura: Ca - campo de captura
                Te - campo solo texto
                TC - campo de texto seguido de captura

  : Numero :                  Campo                    :Camp: Num. Car : Error : Posicion  :
```

1	Numero de funciones (1,2,3,4,5) :	TC	34 - 1		1: 1
2	Numero :	Te	8		
3	de :	Te	8		
4	Tipo de Union:	Te	13		
5	Funcion:	Te	8		
6	Tabla X:	Te	7		
7	Tabla Y:	Te	7		
8	Nombre de la Funcion:	Te	20		
9	de los Puntos:	Te	13		
10	1:	TC	1 - 10		2: 11
11	AAAAAAAAAA:	Ca	10		12: 21
12	CCCCCCCCCCCCCCCCCCCCCCCCC:	Ca	25		22: 46
13	E:	Ca	1		47: 47
14	2:	TC	1 - 10		48: 57
15	AAAAAAAAAA:	Ca	10		58: 67
16	CCCCCCCCCCCCCCCCCCCCCCCCC:	Ca	25		68: 92
17	E:	Ca	1		93: 93
18	3:	TC	1 - 10		94: 103
19	AAAAAAAAAA:	Ca	10		104: 113
20	CCCCCCCCCCCCCCCCCCCCCCCCC:	Ca	25		114: 138
21	E:	Ca	1		139: 139
22	4:	TC	1 - 10		140: 149
23	AAAAAAAAAA:	Ca	10		150: 159
24	CCCCCCCCCCCCCCCCCCCCCCCCC:	Ca	25		160: 184
25	E:	Ca	1		185: 185
26	5:	TC	1 - 10		186: 195
27	AAAAAAAAAA:	Ca	10		196: 205
28	CCCCCCCCCCCCCCCCCCCCCCCCC:	Ca	25		206: 230
29	E:	Ca	1		231: 231
30	(1) lineas:	Te	10		
31	(2) marcadores:	Te	14		
32	(3) marc. - lineas:	Te	18		
33	(4) marc conec a y = 0:	Te	22		
34	Titulos Generales de las Funciones:	Te	34		
35	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC:	Ca	40		232: 271
36	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC:	Ca	40		272: 311
37	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC:	Ca	40		312: 351

Numero de renglones de la forma : 17
Numero total de caracteres de los campos de captura : 351

Listado 7-11

Para organizar la cuenta de desarrollo y facilitar la instalación del sistema, todas las formas se cargan bajo el nombre lógico SER$FORMAS seguido del nombre de la forma. SER$FORMAS está asignado a un subdirectorio de la cuenta de desarrollo. Es precisamente en ese subdirectorio donde se encuentran todas las formas empleadas en el Sistema Interactivo.

Como es obvio, teniendo un número tan grande de tipos de gráficas, no era una solución lógica crear 14 subrutinas, cada una encargada de la captura de sus formas. Además, si bien el Manejador de Captura resuelve todos los problemas

relacionados con la captura de información a nivel de campos, las acciones en particular que se deben tomar en cada tecla funcional, compete a un control externo. Por estos motivos se hizo una subrutina general capaz de llevar el control de la captura de hasta 3 formas. El control consiste en tomar acciones adecuadas con cada tecla funcional, como ayuda, desplegado del directorio de tablas, regreso a la forma anterior, manejo de errores y de la retroalimentación. Mientras el manejo de las teclas funcionales como el desplegado y el parşadeo para retroalimentación se llevó a cabo a través del Manejador de Teclas Funcionales, el control de las acciones de cada tecla se hizo con estados. De igual forma, el desplegado y captura de las formas se efectuó a través del Manejador de Captura y el manejo de la transición entre las tres formas se realizó a través de estados. La validación de la información se realiza a nivel campo, esto es, cuando se captura la información del campo, por medio del Manejador de Captura. Para tal efecto se elaboraron 5 rutinas de validación:

1) **Campos D o I.**- Usada fundamentalmente en las formas donde el usuario debe especificar a que eje se va a escalar una función (eje Y izquierdo o eje Y Derecho).

2) **Tabla Numérica.**- Usada en los campos donde la tabla debe existir y debe ser del tipo numérica, como en las muestras de un histograma.

3) **Tabla Alfanumérica.**- Esta se usa en los campos donde la tabla debe ser alfanumérica y naturalmente, también debe existir. Tal es el caso de los valores en X, como podrían ser los nombres de los meses del año, de los diagramas de barras; o bien, en los nombres de las funciones de una gráfica X vs Y.

4) **Existencia de una Gráfica.**- Su propósito es evitar que en la forma de impresión de gráficas el usuario introduzca una gráfica no existente.

5) **Tabla No Existente y si existe debe ser Numérica.**- Este es un caso muy particular para las formas de evaluación de funciones. Cuando el usuario evalúa por primera vez la función, es probable que las tablas asociadas no existan, pero las subsecuentes veces las tablas numéricas ya fueron creadas. Por lo tanto, previene de que el usuario introduzca una tabla alfanumérica para depositar los valores de los datos numéricos, resultado de la evaluación de la función.

A pesar de que estas rutinas satisfacen perfectamente las necesidades de validación, no detectan los casos que suceden porque el valor de un campo condicione los valores de otros. Por ejemplo, es posible que un usuario seleccione acotamiento logarítmico en un eje y proporcione líneas de referencia negativas. En ambos casos cada campo pasó su validación individual, sin embargo, no es posible tener líneas de referencia negativas y el acotamiento del eje logarítmico. Para resolver esas situaciones, después de capturar cada forma, el sistema invoca a una rutina de validación que verifica todos estos casos. En las rutinas de validación se selecciona, por medio de un CASE, qué clase de verificación, de uno de los tres tipos posibles, se realiza. A continuación se describen los tres tipos de validación por formas.

1) **Tablas de Igual Tamaño.-** Muchas gráficas, por ejemplo X vs Y, requieren que las parejas de datos X y Y, tengan el mismo número de datos. Esta validación verifica que las tablas que integran a la gráfica tengan el mismo número de datos.

2) **Elementos Fuera de los Límites y Número de Elementos Diferente al Número de Intervalos.-** A este caso corresponden las gráficas que tienen que ver con datos estadísticos. Es posible que el usuario, al especificar los límites inicial y final de una muestra de datos, se equivoque y esto provoque que algunos elementos de la muestra caigan fuera de los límites. También puede suceder que el usuario indique que sus datos estan estadísticamente agrupados, entoncen el número de elementos de la muestra debe coincidir forzosamente con el número de intervalos. Ambos casos son verificados por esta rutina.

3) **Lineas de Referencia Positivas.-** Cuando se elige un acotamiento del eje logarítmico, sus lineas de referencia deben ser rigurosamente positivas. En algunas gráficas el tipo de acotamiento se especifica en la segunda forma y la tercera forma contiene los valores de las lineas de referencia. Esta rutina maneja perfectamente estos casos.

Todas estas rutinas dan una idea más clara de cómo se manejó la captura de los 14 tipos de gráficas distintas. No obstante, de todas formas se decidió elaborar 14 rutinas, esencialmente por claridad en la programación; cada una conteniendo una llamada a la rutina que captura las tres formas, denominada CAPTURA. En su interior cada rutina, pasa a CAPTURA, los nombres de las formas, de las ayudas asociadas e inicializa los valores de los campos de cada una de las formas. En el listado 7-III se muestra la rutina para la captura de la gráfica Histograma.

```
SUBROUTINE CapturaHistograma (NomGraf, NumGraf, Alta, CanceloCap)
!
! Despliega las formas de captura correspondientes a Histograma. Si
!la grafica ya existe, extrae los valores existentes y los envia
!como valores iniciales a las formas. Al terminar de capturar
!exitosamente actualiza o da de alta la grafica en el catalogo de
!graficas.
IMPLICIT NONE

!! Parametros
CHARACTER *(*)
1    NomGraf
INTEGER
1   NumGraf
LOGICAL *1
1   Alta,
1   CanceloCap

!! Commons
COMMON /CapturaCar/
1   ArchivosFormas,
1   ArchivosAyudas,
```

```
1   DatGraf

    CHARACTER
1       ArchivosFormas (3) * 30,
1       ArchivosAyudas (3) * 30,
1       DatGraf * 650

COMMON /CapturaEnt/
1   NumeroTopicos

    INTEGER
1       NumeroTopicos (3)

!! Variables
INTEGER
1   Existe,
1   TipoGrafica /5/,
1   TamFormas(2) /179,133/              !Numero de bytes a capturar en
                                        !cada forma
CALL MAN_Inserta ('Histograma')
CALL MAN_Despliega ( )

IF (Alta) THEN
    Existe = 1
    CALL IniciaValoresHistograma(DatGraf, TamFormas)
ELSE
    Existe = 0
    CALL MES_ExtraeGrafDeEstruc(NumGraf, DatGraf)
ENDIF

ArchivosFormas (1) = 'SER$FORMAS:Histo1'
ArchivosFormas (2) = 'SER$FORMAS:Histo2'
ArchivosAyudas (1) = 'SER$AYUDAS:Histo1.MAY'
ArchivosAyudas (2) = 'SER$AYUDAS:Histo2.MAY'
NumeroTopicos  (1) = 8
NumeroTopicos  (2) = 5
CALL Captura (2, Alta, ArchivosFormas, ArchivosAyudas,
1           NumeroTopicos, DatGraf, TamFormas, TipoGrafica,
1           CanceloCap)
IF (.NOT. CanceloCap) THEN
    CALL MES_AltaGraf(DatGraf, NumGraf, Existe, NomGraf, TipoGrafica)
ENDIF

CALL MAN_Elimina ( )
CALL MAN_Despliega ( )

RETURN
END
```

Listado 7-111

En el programa anterior se puede apreciar que el uso de "COMMONS" es fundamentalmente para reducir el consumo de memoria, pues de no tenerlas en "COMMON", esas variables se repitirían para cada una de las 14 gráficas. Simplemente con la variable **DatGraf** en "COMMON" implica usar 650 bytes en vez de 8K. La descripción interna del manejo de las teclas funcionales sigue a continuación.

7.2.3 Teclas Funcionales.

Con respecto a las teclas funcionales no hay mucho que decir, el despliegue y el parpadeo se hacen a través del manejador de teclas funcionales, con las rutinas **MTF_Despliega** y **MTF_Parpadea** respectivamente. El control de las acciones a tomar con cada tecla es externo, esto es, cada programa por medio de estados decide qué rutinas llamar o qué hacer cuando el usuario oprime una tecla.

Lo que es digno de mencionar, es el artificio empleado para desplegar más teclas funcionales que las que caben en un renglón. Debido a que los textos asociados a las teclas funcionales son mas o menos largos para mantener claridad, la tarea de desplegar las teclas funcionales en un solo renglón se complica. Para solucionar el problema se consideraron dos alternativas. La primera, consistía en recortar los textos de las teclas funcionales, lo cual contradecía las recomendaciones de los principios de los Factores Humanos. La segunda, y finalmente por la que se optó, versaba en desplegar las teclas por bloques. Con esta modalidad, el usuario a través de la tecla <PF1>, indica que desea desplegar el siguiente conjunto de teclas funcionales. Cuando el bloque es el último y el usuario presiona <PF1>, el sistema despliega el primer bloque de teclas, de tal suerte que semeja una lista circular. El artificio estriba en tener las teclas funcionales correspondientes a cada bloque, mas la tecla <PF1>. Cada vez que el usuario oprime esta tecla, el programa vuelve a desplegar las teclas funcionales, pero ahora del siguiente bloque. Esto no afecta a la detección de las teclas, pues el desplegado y la detección son totalmente independientes. Lo cual implica que a pesar de que una tecla funcional no se despliegue, sigue estando activa.

7.2.4 Predicción De Nombres.

En esta sección se tocara el tema de la predicción de los nombres, mencionada en la rúbrica de MEMORIA DE CORTO PLAZO de la primera parte del capítulo. Como se había mencionado, el sistema ofrece la modalidad de "recordar" el último nombre empleado en tablas, gráficas y bitácora de acuerdo a diversos criterios, y así ahorrarle al usuario la tarea de memorizar el nombre o consultar el directorio.

La forma en como el sistema "recuerda" un nombre, está ligada a una propiedad del lenguaje FORTRAN 77 conocida como SAVE. Cuando se elabora una subrutina, todas las variables locales no pierden su valor al terminar la ejecución de la misma, sino que lo conservan. Esto permite que las variables sean inicializadas en tiempo de compilación y comiencen la ejecución de la subrutina con el valor final que tuvieron, al salir de ella la vez anterior. Luego entonces, combinando la propiedad SAVE y el modo de valor inicial del Manejador de Diálogos, es posible iniciar una lectura de un nombre, por ejemplo de la tabla a modificar, empleando un nombre guardado con SAVE.

Para las respuestas tipo S/N, no se emplea la modalidad SAVE, sino simplemente la capacidad del Manejador de Diálogos, de iniciar una lectura con un valor inicial. Así pues se le asigna a la respuesta, por ejemplo, el valor "s" y se le indica al Manejador que use ese valor para la lectura. El usuario simplemente debe oprimir RETURN cuando está satisfecho con la predicción o borrarla y escribir su respuesta cuando no lo está. A continuación se detallan los aspectos internos de la sección de impresión de gráficas.

7.2.5 Impresión De Las Gráficas.

En este punto se describen las características de la captura de las formas de impresión, así como de las rutinas de impresión de las gráficas.

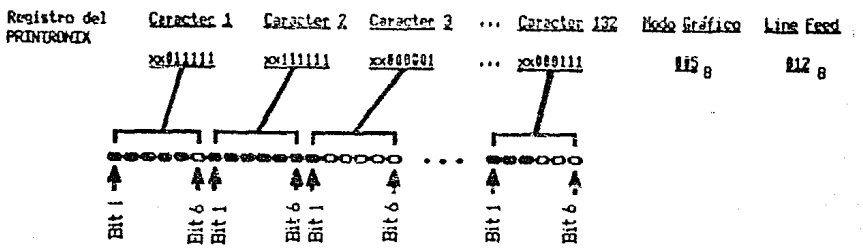
Cuando se selecciona la opción de impresión de gráficas del menú principal, se despliega una forma de captura donde el usuario introduce el número y el nombre de las gráficas a imprimir, las dimensiones de la hoja y de los márgenes, así como los títulos generales, el número de copias que desea y si desea que sólo se genere, y por lo tanto no se imprima, el archivo de puntos. Una vez capturados los datos, se valida que las dimensiones y los márgenes dejen un espacio mínimo de 6 cm., que se consideró el más razonable, a lo largo y a lo ancho de la hoja. Abre el archivo de impresión con el nombre _SER_GRAF.PLT, define al dispositivo gráfico como el PRINTRONIX 600 y establece las dimensiones, márgenes y títulos, capturados en la forma, todo a través de la rutina SER_DefineHoja. Para cada tipo de gráfica se llama a una rutina especializada, encargada de extraer los datos de las gráficas de las estructuras de almacenamiento, e invocar a las rutinas del sistema SER correspondientes que dibujan las gráficas. Si estas rutinas detectan que existe algún error, el módulo de impresión se regresa al estado de captura de la forma, si no, con la función de RTL LIB$SPAWN, se invoca el comando PRINT que envía el archivo _SER_GRAF.PLT, el cual contiene el dibujo, al PRINTRONIX. El número de copias simplemente se realiza con el calificador COPIES del comando PRINT.

Con respecto a las rutinas especializadas de impresión de las gráficas, se puede mencionar que son 14, una para cada tipo de gráfica, las cuales llevan el prefijo Llama. Es decir, LlamaXvsY, LlamaHisto, etc. Cada rutina extrae los campos de una variable tipo caracter, o registro, de las estructuras de almacenamiento de información. Esta variable caracter se obtiene de la rutina MES_EstraeGrafDeEstruc, pasándole como parámetro el nombre que le asignó el usuario a una gráfica. De esta forma cada rutina especializada debe recibir del módulo de impresión, el nombre de gráfica que se va a dibujar; posteriormente, la rutina "LLAMA" convierte cada campo al tipo de variable que le corresponde, numérica o alfanumérica; excepto las tablas, ya que en el registro sólo se proporciona el nombre de la tabla y no su contenido. Para extraer su contenido, existen dos rutinas del Manejador de Estructuras MES_ExtraeVecNum y MES_ExtraeVecAlfa, que permiten extraer el contenido de las tablas, numéricas o alfanuméricas, y depositarlo en un vector real o caracter. Una vez hecho esto, las rutinas "LLAMA" simplemente invocan a las rutinas SER que se encargan de dibujar cada tipo de gráfica a partir de los parámetros extraídos.

Debido a que el PRINTRONIX 600 sufre de descomposturas mas o menos en forma frecuente, y ya que no se cuenta en el Centro con ningún otro tipo de dispositivo manejado por el PGPLOT, el sistema padecía de un punto de gran fragilidad al permitir sólo este dispositivo gráfico de salida. Para mitigar

esta deficiencia se pensó en realizar un traductor. Es decir, un programa que tradujera los puntos del archivo _SER_GRAF.PLT, para el PRINTRONIX 600, a puntos de otra impresora gráfica con la cual sí cuenta el Centro. La primera alternativa fué la impresora gráfica DECWriter IV Graphic Printer. Este periférico formaba parte del equipo gráfico GIGI, pero por faltar todas las terminales gráficas, a esta impresora no se le había encontrado un buen proposito. La única condición para que un dispositivo sea un traductor, es que el número de puntos por pulgada, o resolución, sea mayor o igual al del PRINTRONIX 600. En forma vertical la resolución del DECWRITER IV es de 792 puntos en 11 pulgadas, al igual que el PRINTRONIX. Mientras que horizontalmente, el DEC tiene 1736 puntos en 13 pulgadas y el PRINTRONIX 792.

La información en el PRINTRONIX comprende una línea de 792 puntos en forma horizontal. Cada bloque de 6 puntos se almacena en un byte, de tal suerte que si el primer bit vale 1, indica que el martillo se dispare en la primera posición, el segundo bit controla al segundo martillo y así sucesivamente. El segundo bloque controlará a los martillos del 7 al 12, así hasta completar 792. Los bits 7 y 8 son ignorados en la producción de los puntos (vease la figura 7-2)

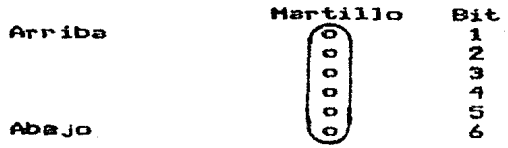


Notar que el primer bit del byte del caracter 1, dispara el martillo de más a la izquierda (martillo 1). La x denota que el valor de ese bit no importa.

Figura 7-2

En la impresora DECWRITE IV, los martillos se manejan no en forma horizontal, sino en forma vertical, tal y como se muestra en la figura 7-3. Así entonces, un byte dispara 6 puntos en forma vertical, para formar un máximo de 1736 columnas de puntos, en un renglon.

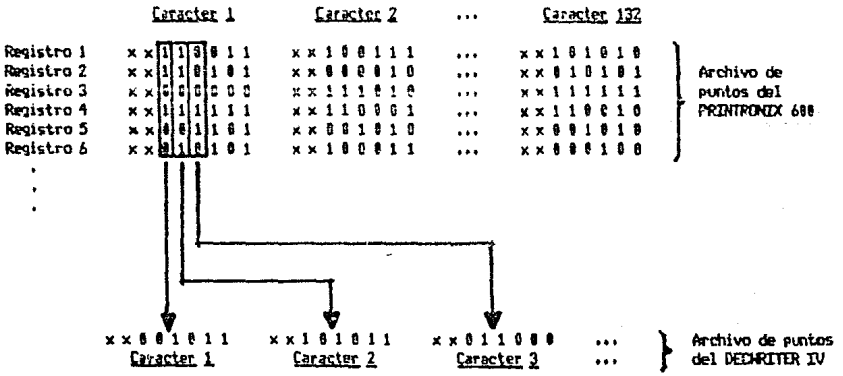
El traductor lo que simplemente hace es leer el archivo _SER_GRAF.PLT, por bloques de 6 registros y agrupar cada 6 bits en forma vertical, en un caracter gráfico del DEC. Así sucesivamente, hasta completar todos los puntos horizontales del PRINTRONIX (vease la figura 7-4). Como la resolución en forma horizontal no es la misma, cada punto del PRINTRONIX lo repite 2.19 (1736/792) veces en forma horizontal y así, genera un archivo con extensión DEC.



Si el byte fuera 00000001 la cabeza de la impresora dispararía el martillo de más arriba.

Figura 7-3

Para poder imprimir este archivo, primero se le indica en la forma de captura que no imprima el dibujo en el PRINTRONIX, sino simplemente genere el archivo de puntos, posteriormente se corre el traductor para que produzca el archivo .DEC. La impresora DECWRITE IV se conecta en un puerto de la VAX destinado a una terminal, se cambia el ancho de esta terminal a 255 bytes con el comando SET TERMINAL/WIDTH=255/PERMANENT y se define este puerto con capacidades de cola de impresión. Posteriormente se envía el archivo _SER_GRAF.DEC a esa cola, con el comando PRINT/QUEUE.



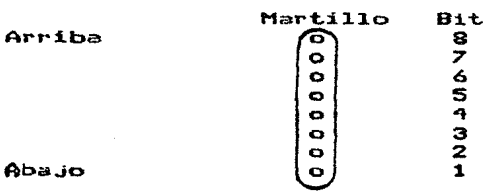
En la figura se aprecia cómo la información se extrae por columna, del archivo de puntos PRINTRONIX, para formar el archivo de puntos del DECWRITER IV. Hótese que el DECWRITER imprime 6 renglones del PRINTRONIX de una sola pasada, pero tiene que recorrer 792 (132x6) columnas. La x significa que el valor de ese bit no interesa.

Figura 7-4

Como la impresora DECWRITE IV no es muy común y pueden existir problemas por mantenimiento, se eligió una impresora un poco más comercial que es la impresora EPSON. La impresora EPSON es familiar en el ambiente de microcomputadoras y por tal motivo su disponibilidad es siempre muy alta. Desafortunadamente, el puerto

de comunicación de esta impresora es paralelo del tipo CENTRONICS y no se puede conectar directamente a la VAX, porque ésta maneja puerto serie. Sin embargo, si se cuenta con una microcomputadora IBM y un emulador de terminal VT100, como es el VTERM de Coefficent Systems Corporation, es posible conectar la microcomputadora como terminal de la VAX y transferir información generada en VAX hacia la microcomputadora y una vez ahí, imprimir cualquier archivo en la impresora EPSON.

Bajo estas circunstancias, la realización del traductor de PRINTRONIX a EPSON fue eminente. La forma en que la impresora PRINTRONIX 600 maneja las gráficas se mencionó previamente; la manera en que la impresora EPSON dibuja los puntos de una gráfica se explica a continuación. Al igual que la impresora DECWRITER IV, la EPSON tienen alojados los puntos de la cabeza de impresión en forma vertical, solo que en lugar de estar conformada por 6 puntos, posee 8 verticales. El bit menos significativo dispara el martillo de más abajo, mientras que el octavo bit dispara el martillo de más arriba, esto es, en forma inversa al DECWRITER IV (vease la figura 7-5).



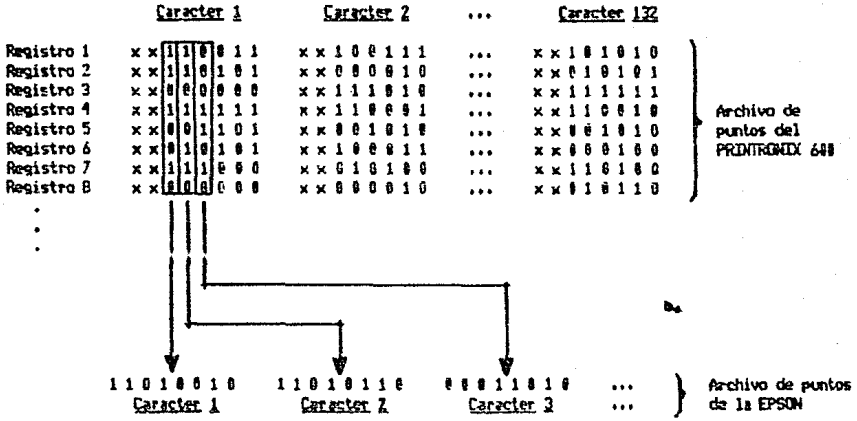
Si el byte fuera 00000001_2, la cabeza de la impresora dispararía el martillo de más abajo.

Figura 7-5

La resolución en forma vertical de la impresora EPSON coincide con la del PRINTRONIX 600, es decir, 792 puntos en 11 pulgadas. En forma horizontal la EPSON cuenta con 120 puntos por pulgada, en modo de doble densidad a velocidad media. Y como el PRINTRONIX posee 60 puntos por pulgada, cada punto de esta impresora se debe imprimir dos veces para conformar el archivo de puntos de la impresora EPSON.

Así pues, el traductor toma el archivo de puntos del PRINTRONIX, lee 8 registros y para cada uno de los caracteres que conforman los 792 puntos horizontales, analiza cada grupo de 6 bits. Como la impresora EPSON funciona como columna y no en forma horizontal, el traductor agrupa en 8 bits cada uno de los 8 registros del PRINTRONIX, para cada bit de los 6 de un caracter PRINTONIX. Así continúa hasta terminar con los 792 puntos del PRINTRONIX (vease la figura 7-6).

Una vez generado el archivo de puntos de la impresora EPSON, el archivo se tranfiere a la microcomputadora IBM y ahí se imprime con un pequeño programa realizado en PASCAL, que se encarga de leer cada byte y enviarlo a la impresora.



En la figura se aprecia cómo la información se extrae por columna, del archivo de puntos PRINTRONIX, para formar el archivo de puntos de la impresora EPSON. Nótese que la EPSON imprime 8 renglones del PRINTRONIX de una sola pasada, pero tiene que recorrer 792 (132x6) columnas. La x significa que el valor de ese bit no interesa.

Figura 7-6

Este archivo no se puede imprimir con el comando PRINT de MS-DOS, ya que como el archivo contiene información gráfica, ciertos caracteres se asumen como fin de archivo y el comando PRINT termina su ejecución antes de encontrar verdaderamente el final del dibujo; el programa se muestra en el listado 7-IV. Como siguiente punto se expone la forma en que se resolvió la opción de Evaluación de Funciones.

```
PROGRAM Imprime
VAR
    Nomb   : STRING [30];
    RegCar : CHAR;
    Arch   : TEXT;
    Salte  : BOOLEAN;

BEGIN

( Archivo a imprimir )
WRITE ('Nombre del archivo?');
READLN (Nomb);
ASSIGN (Arch, Nomb);

WHILE NOT EOF (Arch) DO
```

```
        BEGIN
        Salte := FALSE;

        WHILE NOT (Salte) DO
           BEGIN
           READ (Arch, RegCar);
           WRITE (LST, RegCar);

           IF RegCar = CHR (13) THEN
               BEGIN
               READ (Arch, RegCar);
               WRITE (LST, RegCar);

               IF RegCar = CHR (10) THEN
                  Salte := TRUE;
               {NOELSE}
               {ENDIF}

               END
           {NOELSE}
           {ENDIF}

           END;
        {ENDDO}

     END;
  {ENDDO}

  END.
```

Listado 7-IV

7.2.6 Evaluación De Funciones.

Esta opción se encuentra dentro del submenú de Introducir y Editar Tablas de Datos. A pesar de que no es una opción sumamente importante, merece especial antención por la complejidad que representó su desarrollo. La opción fundamentalmente, permite que un usuario escriba una función matemática para que el sistema la evalúe. Para esto, se proporcionan los límites de evaluación los valores inicial y final, y el número de puntos, por medio de una forma de captura. Con estos parámetros la función se evalúa dejando las parejas de puntos X y Y, en dos tablas previamente especificadas por el usuario. Se contemplan dos tipos de funciones, las paramétricas y las no paramétricas. La solución planteada para proporcionar esta opción fué la siguiente.

Las funciones paramétricas son del tipo:

$X = f(T)$ $\qquad\qquad$ $Y = g(T)$

Donde T es el parámetro. Las funciones no paramétricas se pueden considerar funciones paramétricas del tipo:

X = T Y = g(T)

Esto simplifica la tarea, pues reduce todo a considerar un solo tipo de función y permite que un mismo módulo resuelva las dos opciones: evaluación de funciones paramétricas y no paramétricas.

La técnica radica en generar un archivo que contiene las dos funciones, tanto X como Y. Ambas funciones las debe escribir el usuario respetando las convenciones en FORTRAN para que esto opere adecuadamente. Una sección del código encargada de esto se muestra a continuación:

```
      OPEN (UNIT = 10, FILE = '_SER_Funcion.FOR',
     1      STATUS = 'New', CARRIAGECONTROL = 'None')

      WRITE (10, 1000) 'FUNCTION F (X,T)'
      WRITE (10, 1000) 'IMPLICIT NONE'
      WRITE (10, 1000) 'INCLUDE ' // Apostrofe //
                       'SER$DISCO:DeclFunc.FOR' // Apostrofe
      WRITE (10, 1000) 'REAL X,F,T'
      WRITE (10, 1000) 'X='//FunXSinBlancos
      WRITE (10, 1000) 'F='//FunYSinBlancos
      WRITE (10, 1000) 'END'

      CLOSE (10)
```

Listado 7-V

Para proveer de las funciones predefinidas que FORTRAN no tiene, como son la función SENC, la función triangular, el factorial de un número, etc, se tiene en SER$DISCO un archivo donde se declaran todas estas funciones y así el usuario cuando las llame, el compilador no marque "identificador de declarado". Vale la pena mencionar que cuando la función es No Paramétrica, previamente X se iguala a T. Por medio de la función de RTL de VAX, LIB$PAWN, se lanza un procedimiento de comandos que compila la función y detecta si existen errores de compilación. Si los hay, escribe en un archivo un valor para que el Sistema Interactivo detecte que existieron errores y regrese al usuario a la forma de captura, y así corrija estos errores. Si no tiene errores, la función se liga a una programa llamado _SER_GENERA y a las funciones predifinidas (SER$DISCO:Funciones.OBJ), con el comando LINK, lanzado a través de la rutina de RTL LIB$SPAWN. El programa _SER_GENERA existe en la clave de desarrollo (SER$DISCO) y se encarga de evaluar la función desde el valor inicial del intervalo hasta el valor final, con el número de puntos especificados. Los intervalos de evaluación, así como el número de puntos los obtiene a través de un archivo que el Sistema Interactivo escribe. En el listado 7-VI se muestra el programa _SER_GENERA.

```
PROGRAM _SER_Genera
IMPLICIT NONE

REAL
1   F,
```

```
1   Vo,
1   Vf,
1   X,
1   Y,
1   Inct,
1   t

INTEGER
1   I,
1   NumP

    OPEN (UNIT =1, FILE = '_SER_INTERV.TAB',
1         STATUS = 'OLD')
       READ (1,*) Vo, Vf, Nump
    CLOSE (1)

    OPEN (UNIT = 1, FILE = '_SER_X.TAB', STATUS = 'NEW')
    OPEN (UNIT = 2, FILE = '_SER_Y.TAB', STATUS = 'NEW')

       Inct = (Vf - Vo) / (Nump - 1)
       t = Vo

       DO I=1, NumP

          Y = F(X,t)
          WRITE (1,*) X
          WRITE (2,*) Y
          t = t + Inct
       ENDDO

       CLOSE (2)
       CLOSE (1)

    END
```

Listado 7-VI

Después, se ejecuta el programa _SER_GENERA a través de un procedimiento de comandos, lanzado también con la rutina LIB$SPAWN. Si existe un error de ejecución el procedimiento escribe en un archivo un número que le indica al Sistema Interactivo que hubo un error. El Sistema entonces manda un diagnóstico de error y regresa al usuario a la forma de captura. Si no hay error, permite que _SER_GENERA deposite los valores de X y Y en dos archivos _SER_X.TAB y _SER_Y.TAB, para que posteriormente el Sistema Interactivo los carge en dos tablas, previamente especificadas por el usuario, a través de la rutina del Manejador de Estructuras MES_DeArchATabla.

Como es muy común que un usuario evalúe la misma función varias veces, exclusivamente cambiando los límites de evaluación y el número de puntos hasta encontrar algunos que satisfagan sus necesidades, el sistema evita recompilar la función detectando si la función escrita por el usuario es distinta. Si es igual, implica que la función ya está compilada y sólo se debe ejecutar _SER_GENERA. Para este efecto a la función se le quitan todos los espacios en

blanco y se traduce a mayúsculas, para asegurar que la función no haya sufrido cambios substanciales.

De esta forma se concluye el punto de evaluación de funciones para proceder con la rúbrica de la documentación en linea, conocida como ayudas.

7.2.7 Ayudas.

En esta sección se detallan los puntos más sobresalientes de cómo se llevó a cabo la implementación de las ayudas.

Las ayudas son textos que aparecen en la pantalla cuando el usuario oprime la tecla funcional "?". Los textos de ayuda siempre se escriben dentro de una ventana, de tal suerte, que cuando la ayuda se ha agotado, la ventana se remueve sin afectar la información que había en la pantalla, antes de invocar a la ayuda. Las ayudas que despliega el Sistema Interactivo se efectúan a través del Manejador de Ayudas, de tal forma que la implementación de esta función, se redujo exclusivamente a elaborar los textos. Sin embargo, debido al número de pantallas de menús y al número tan grande de formas de captura, esta tarea no fué tan sencilla.

Como primer punto las ayudas se organizaron en el directorio de desarrollo bajo el nombre lógico de SER$AYUDAS. Para minimizar el número de textos a escribir, se identificaron qué textos eran comunes a varios menús o formas de captura. Una vez determinados los textos que había que escribir, se procedió a elaborarlos a través del procesador de palabras Digital Standard RUNOFF. Esto permitiría modificar fácilmente los textos de las ayudas sin perder el formato de presentación. Los márgenes que se eligieron fueron del 0 al 39, es decir 40 columnas, ya que ese es el ancho definido para las ventanas de ayuda. Cabe mencionar que si las ventanas cambian de ancho, simplemente es necesario modificar los márgenes de los archivos de RUNOFF y reprocesarlos. Para dar una idea más clara de cómo se efectuó esto, en el listado 7-VII se muestra el archivo de ayuda del tipo de grafica X vs Y

```
.Autoparagraph
.Nopaging
.lm 0
.rm 39
.c;GRAFICA X VS. Y
 Una grafica X vs. Y se utiliza para representar en un plano coordenado una
funcion matematica o empirica de una variable dependiente (Y) y de otra
independiente (X). Los ejes coordenados sobre los cuales se traza la grafica
son continuos.
 Este tipo de grafica permite dibujar dentro del mismo plano coordenado como
maximo cinco funciones, esto es con el fin de poder de poder comparar el
comportamiento de diferentes fenomenos.
 Cada una de las funciones que se presenta dibujar debe constar de dos
tablas de datos (Tabla de abscisas y tabla de ordenadas).
 Se proporciona una opcion para definir dos ejes de ordenadas (Eje Y derecho y
eje Y izquierdo) para que el usuario pueda asignar dos escalas diferentes
dependiendo del comportamiento de sus funciones.
```

```
.lit
.PAGINA
        SIN(X) --- EXP(X)
 E    +----------------------+14 E
 j   1+           !          :   j
 e    :   ***     '       ****+12 e
    .5+ *      *   '     *    :
 Y    :*         * /      *    +8  Y
      0*          */       *   :
 I    :           *        *   +4  D
 z -.5+         /  *      *    :   e
 q    :-----'      ****         +0  r
   -1+---+---+---+---+---+---+
.end lit
```

Listado 7-VII

Cuando se procesa el texto con RUNOFF, éste produce un archivo con extensión .MEM, el cual contiene los caracteres de control adecuados para producir en la impresora el texto tal y como se le indicó a RUNOFF. Sin embargo, estos caracteres de control como <CR>, <LF>, <FF>, producen saltos extras de renglón y movimientos indeseables del cursor de la pantalla. Por tal motivo se realizó un programa, denominado CONVIERTE, que produce un archivo .MAY sin los caracteres de control del archivo .MEM, de acuerdo a como lo requiere el Manejador de Ayuda. Es decir, sin caracteres de control, pero sí conteniendo los espacios entre palabras para formar la justificación a la derecha. Vale la pena hacer notar que no se permite que el archivo RUNOFF realice sobreimpresión (OVERSTRIKE) o doble impresión (BOLDING), pues el programa **CONVIERTE** dejaría dos registros en vez de uno para el renglón en donde se sobreimprime o se imprime doble; de igual forma, tampoco se permite que contenga el caracter <TAB>, pues éste produciría más espacios en blanco que los calculados.

Se realizaron alrededor de 102 archivos de ayuda que integran toda la documentación en línea del Sistema Interactivo. Como se mencionó al principio de la sección, las ayudas están divididas en dos grandes secciones, las ayudas de los menús y las de las formas de captura. A continuación se explica brevemente la estructura de las ayudas de los menús.

Los textos de ayuda elaborados para cada menú de opciones tienen un punto en común. Este es un texto que explica cómo se selecciona una opción del menú, y para evitar escribirlo en cada una de las ayudas de los menús de opciones, se escribió un solo archivo que explica el uso de los menús. Este archivo se integra a la ayuda de un menú con el comando del Manejador de Ayuda ".INCLUYE".

Cada texto de ayuda tiene sus rúbricas de explicación, de acuerdo a las opciones que integran el menú. Para efectuar esto, se aprovechó la capacidad de mostrar la ayuda por tópicos, de tal suerte que el usuario, al aparecer una ventana de ayudas, simplemente tiene que seleccionar el tópico del cual desea desplegar su ayuda (vease la figura 7-7). En el listado 7-VIII se muestra una parte del archivo RUNOFF del Menú Principal.

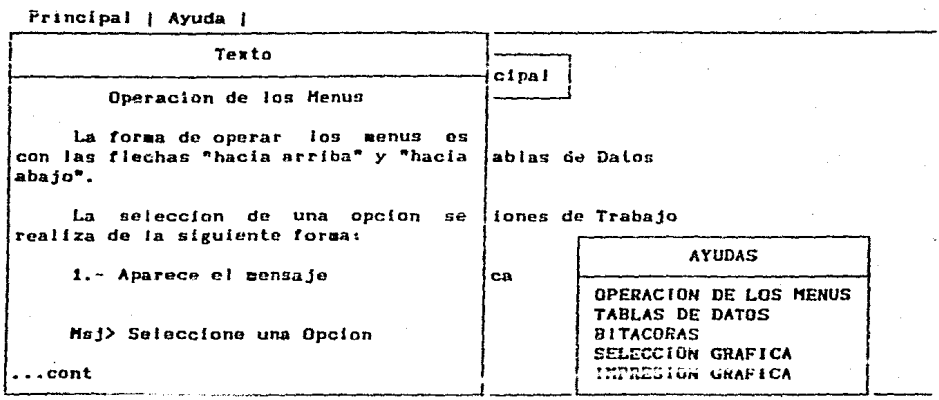


Figura 7-7

Como se puede apreciar, en el archivo de ayuda, simplemente se indican cuáles son los puntos que integran cada tópico y posteriormente se detalla la explicación de cada uno, con el comando del Manejador de Ayuda ".TEXTO".

```
.Autoparagraph
.Nopaging
.lm 0
.rm 39
.literal
.XO;50
.YO;12
.XT;1
.YT;2
.ANCHOT;40
.LARGOT;15
.TITULO;AYUDAS
.TOPICO(1);Operacion de los Menus
.TOPICO(2);Tablas de datos
.TOPICO(3);Bitacoras
.lit
.TEXTO(1)
.INCLUYE;ser$ayudas:menu.may
.TEXTO(2)
.end literal
.c;TABLAS DE DATOS
```

Los datos que se toman para dibujar una grafica deben estar contenidos en una

```
tabla. El numero de tablas de datos que conformar una grafica dependen del
tipo de esta. Existen dos tipos de tablas de
datos : NUMERICAS y ALFANUMERICAS.
 Dentro de la opcion "INTRODUCIR Y EDITAR TABLAS DE DATOS" se puede crear una
nueva tabla, modificar el contenido de una ya existente o eliminar alguna.
.!it
.TEXTO(3)
.end !it
.c;BITACORAS
 Una BITACORA es un archivo que crea el propio sistema SER con el fin de
almacenar la informacion de una sesion de trabajo (Tablas de datos y graficas)
y poder utilizarla en otra ocasion.
 La Opcion "USAR BITACORAS DE SESIONES DE TRABAJO" permite utilizar una
bitacora previamente creada o crear una nueva.
```

Listado 7-VIII

Una de las características de los menús, es que el número de opciones es dinámico. Es decir, solamente se despliegan las opciones que el usuario puede seleccionar. Esta característica también se mantiene en las ayudas, sólo se despliegan tópicos de ayudas de las opciones presentes en el menú. Esto se realizó gracias a la propiedad del Manejador de Ayuda de tópicos variables, la cual consiste en que simplemente el Sistema Interactivo le indica el Manejador de Ayuda, cuántos tópicos debe desplegar. De este modo, siempre se hace coincidir el número de tópicos con las opciones disponibles del menú. A continuación se detalla la estructura de las ayudas, relativas a las formas de captura.

Las ayudas de las formas de captura presentaron un obstáculo importante para llevar a cabo su elaboración: el número tan grande de formas (34). Para cada forma, había que explicar cómo se usaba el Manejador de Captura de formas y detallar el significado de los campos involucrados en las formas. Aquí también el potencial del uso de archivos comunes que se integran a la ayuda a través del comando ".INCLUYE" fué de gran utilidad y redujo toda la tarea a elaborar 18 archivos comunes. Estos archivos comunes contienen una explicación de cómo operar las formas de captura, de los campos más frecuentes y conflictivos de las formas y de los mensajes de error que aparecen al validar la información de una forma. Sin embargo, si bien el uso del comando ".INCLUYE" evita tener que escribir secciones en forma repetitiva, no elude la tarea de agrupar en un archivo de ayuda, a todos los ".INCLUYE" necesarios para una forma de captura. De tal suerte, que sí es necesario elaborar un archivo de ayuda por cada forma de captura, pero con la ventaja de que sólo contiene comandos del tipo ".INCLUYE", tal y como se muestra en la ayuda de la primera forma de captura X vs Y (listado 7-IX).

```
.Autoparagraph
.Nopaging
.lm 0
.rm 39
.literal
.XO;50
.YO;5
.XT;1
.YT;1
.ANCHOT;40
.LARGOT;16
.TOPICO(1);Uso de la Forma
.TOPICO(2);Tipo de Union
.TOPICO(3);Titulos de la Grafica
.TOPICO(4);Caracteres Especiales
.TOPICO(5);Error Num. Elementos Tablas
.TITULO;AYUDAS
.TEXTO(1)
.INCLUYE;ser$ayudas:usocap.may
.TEXTO(2)
.INCLUYE;ser$ayudas:tipoUnion.may
.TEXTO(3)
.INCLUYE;ser$ayudas:titulos.may
.TEXTO(4)
.INCLUYE;ser$ayudas:caracesp.may
.TEXTO(5)
.INCLUYE;ser$ayudas:msjerror1.may
.end lit
```

Listado 7-IX

Al Manejador de Ayuda se le envía el número de tópicos que conforman el archivo de ayuda para cada forma de captura. De tal modo, que al usuario se le presenta un menú de tópicos. Una vez mostrado el menú, el usuario simplemente selecciona el tópico de ayuda que desea desplegar, tal y como sucede en la ayuda de los menús de opciones (vease la figura 7-8).

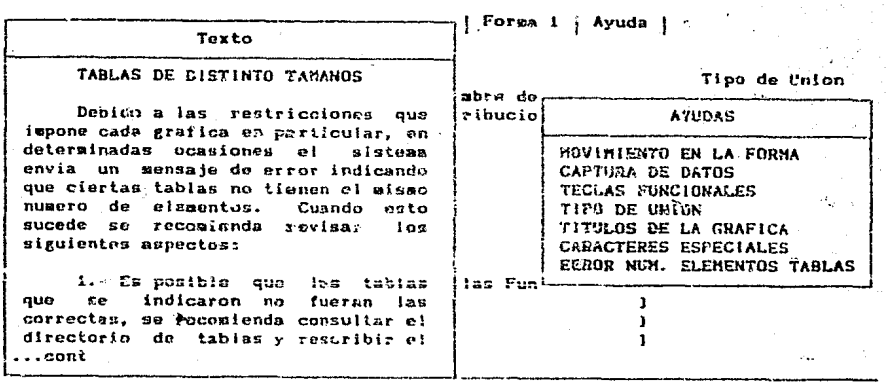


Figura 7-8

Vale la pena mencionar que las ayudas se elaboreron para proporcionar una idea semántica de las funciones de los menús y de los campos. Es decir, las ayudas intentan explicar qué hace una opción y cuál es el objetivo de un campo, y sólo en algunos casos, como en las reglas para escribir una función de evaluación, tratan también el aspecto sintático de las opciones. Por último punto, se describen los detalles relacionados con las utilerías elaboradas para el Sistema Interactivo.

7.2.8 Utilerías.

En este punto se describen las utilerías que se elaboraron especificamente para facilitar la tarea del Sistema Interactivo y que no se proveen por las rutinas RTL de VAX.

A lo largo de todo el sistema se identificaron bloques de uso repotitivo, lo cual amerité que se elaboraran unas rutinas de utilería que satisficieran las demandas del sistema. Las rutinas que se realizaron están involucradas con la eliminación de blancos y traducción a letras mayúsculas de una cadena de caracteres, y con la supresión de blancos iniciales y traducción a letras mayúsculas, también de una cadena de caracteres.

La primera se emplea en el evaluador de funciones, donde se requiere verificar si la función se ha modificado o es igual, y así determinar si se debe recompilar la función o no. Basta recordar que para un compilador los espacios en blanco entre operadores e identificadores son ignorados. La segunda se emplea en las rutinas de validación por campos de captura y en la lectura del

nombres de gráficas y tablas, donde se requiere verificar si tanto la gráfica como la tabla existe o no. Para esto, se debe evitar que un nombre con letras minúsculas se considere diferente al escrito con mayúsculas. También deben eliminarse los espacios en blanco iniciales, para permitir que el usuario por error, oprima la barra espaciadora antes de escribir el nombre de la gráfica o tabla y aún así concuerde con el nombre almacenado en las estructuras. Por estas razones, la segunda utilería es empleada altamente.

Con este punto se dá por terminado el capítulo siete, el cual tiene la finalidad de exponer cómo se aplicaron los principios de los Factores Humanos en la realización del Sistema Interactivo, así como también de detallar la estructura interna del sistema y la forma en que se implementaron todas las funciones expuestas al principio del capítulo. A continuación se prosigue con el último capítulo de la Tesis, que trata de las conclusiones.

CAPITULO 8

CONCLUSIONES

El punto de referencia que se tomó para encontrar las bondades y fallas del sistema SER y las aportaciones que brinda el mismo, fué el medio ambiente en el cual se desarrolló el sistema (CECAFI) y el conocimiento de los sistemas del tipo semejante al antes expuesto.

8.1 APORTACIONES DEL TRABAJO

La principal aportación de este trabajo es el haber satisfecho la necesidad para la cual fué creado. Con este sistema los usuarios del equipo VAX-11/780 del CECAFI cuentan con un paquete para la obtención de gráficas de propósito general. De ahora en adelante ya no se tendrán que desarrollar aplicaciones personales ni aisladas para generar gráficas, a menos que los requerimientos sean en extremo específicos.

1) Modadidades de uso

Si bien es cierto que este sistema no desarrolla una nueva modalidad en el uso de sistemas graficos, si es de los pocos que permiten al usuario utilizarlo en forma interactiva o como una paquetería de rutinas, a través de un lenguaje de alto nivel.

El ofrecer estas dos modalidades de uso hace que el sistema tenga mayor aceptación y difusión; por una parte los usuarios que no requieren aplicaciones fuera de lo común encontrarán que el Sistema interactivo es el más adecuado a sus necesidades, y por otra parte, las personas que desarrollan sistemas y requieren una interfaz gráfica, estarán más a gusto con la paquetería de rutinas gráficas.

A nuestro juicio, esta es una buena aportación porque supera, en este aspecto a los anteriores sistemas gráficos utilizados en el centro de desarrollo de este trabajo.

2) Factores Humanos

El "software" desarrollado en este trabajo fué creado pensando en que las personas que lo utilizarían son de características totalmente heterogeneas. Jamás se pensó que nuestro sistema sería utilizado "en casa" exclusivamente, sino que por el contrario, fué un trabajo que se

hizo pensando en toda la gama de usuarios potenciales.

Para asegurar la aceptación de nuestros usuarios, la interfaz usuario-sistema fué diseñada tratando de aplicar al máximo lo que se ha dado en llamar "Factores Humanos" o Ergonomía. Por nuestra parte, consideramos de gran interés, principalmente para los nuevos sistemas interactivos que sean desarrollados en el CECAFI, el poder presentar un sistema cuya interfaz con el usuario fué diseñada para facilitarle el trabajo a los operarios y no a los diseñadores.

3) "Software" Moderno

La introducción de los factores humanos en el diseño de sistemas involucra un trabajo muchísimo más exhaustivo por parte de los diseñadores. En nuestro caso, el desarrollo de la interfaz, además de involucrar "Factores Humanos", introduce el uso de "software" agradable a la vista y novedoso. Nos referimos al tan difundido uso de "ventanas" de despliegue de información.

Este tipo de "software" fué toda una aportación a la filosofía de diseño de interfaces interactivas en el CECAFI.

El haber tomado el camino del uso de ventanas, trajo consigo un trabajo extra: desarrollar diversos manejadores de despliegue que tuvieran como base el uso de ventanas. Nos referimos a los manejadores de ventanas, diálogos, anidamientos, anidamientos, teclas funcionales, menús, ayudas y captura mencionados en capítulos anteriores.

Si bien el desarrollo de los manejadores antes mencionados fué únicamente el medio para alcanzar nuestros objetivos, también fué una de las principales aportaciones de nuestro trabajo.

Todo el diseño de estos manejadores se llevó a cabo pensando en que pudieran utilizarse en otros sistemas. Ninguno de los manejadores está en lo mas mínimo atado con el sistema SER, por lo que se pueden aplicar para desarrollar cualquier otro tipo de sistema interactivo. El usuario de estos manejadores se evitará muchísimo trabajo (casi todo) en lo que respecta a la interfaz con el usuario.

El programa que gobierna la interfaz con el usuario, en el caso de nuestro trabajo, es un ejemplo real de cómo aplicar estos manejadores en el desarrollo de una interfaz interactiva.

4) Tipos de Gráficas

En lo que respecta a las salidas gráficas del sistema, nuestra aportación principal fué la inclusión de mezclas de gráficas. Con esto nos referimos a la posibilidad de presentar en el mismo marco de referencia (ejes coordenados) dos tipos de gráficas cuyos dominios y contradominios son de diferentes conjuntos.

Como ejemplo podemos mencionar una gráfica X vs Y (función matemática) en combinación con una muestra empírica de datos (histograma), con el fin de poder decidir qué modelo estadístico teórico

se apega más a los datos empíricos.

Este tipo de mezcla de gráficas no lo encontramos en ningún paquete interactivo de los que se investigaron y lo consideramos una aportación nuestra.

8.2 POSIBLES MEJORAS AL SISTEMA

1) Transportabilidad

La transportabilidad de un sistema es un arma de dos filos. Por una parte permite que el "software" sea fácilmente difundido y utilizado por un mayor número de personas, pero por otra parte esto trae consigo que el sistema, al tener que utilizar elementos estándares, requiera de un mayor esfuerzo de programación e ineficiencia en el uso de las bondades del equipo de cómputo donde reside.

En este sentido, nuesto sistema no sigue la tendencia del "software" totalmente atado al equipo ni de aquel totalmente transportable. La principal razón por la que este sistema no es del todo transportable es la falta de un estándar gráfico como pudiera ser GKS. Sin embargo para que las rutinas gráficas puedan funcionar utilizando otro paquete gráfico, lo único que habría que hacer es sustituir lo que llamamos EGR (Estándar Gráfico) dentro de nuestro sistema, por las rutinas con que se cuenten en el equipo destino.

Con respecto al lenguaje de programación se puede asegurar que el 95% del código esta en FORTRAN 77 estándar y que solo un 5% pertenece a rutinas de la biblioteca del sistema VAX, pero que pueden ser sustituidas por rutinas elaboradas por el programador.

En lo que se refiere al diseño de la interfaz con el usuario, ésta es totalmente dependiente de las terminales de video VT100 o compatibles, pues no se cuentan con manejadores (drivers) para otro tipo de terminales. En caso de querer transportar el "software" de la interfaz a otras terminales no compatibles con VT100 se tendría que modificar exclusivamente el manejador de ventanas y requeriría de un esfuerzo mediado, dependiendo de la filosofía de uso de las otras terminales.

2) Interfaz Gráfica

El paquete gráfico utilizado (PGPLOT) ofrece salidas hacia diferentes dispositivos (VT125, Tektronics, Versatec). Por el momento el CECAFI sólo cuenta con equipo gráfico PRINTRONIX, por lo que el sistema únicamente permite salida a este dispositivo.

En caso de adquirir otro tipo de equipo de despliegue gráfico, sería muy provechoso modificar el sistema para que la gráfica se dibujara directamente en la pantalla y el usuario pudiera controlar el flujo de salida. Las modificaciones que se tendrían que hacer al sistema para que pudiera operar sobre dispositivos gráficos es inmediata y no es

digna de tomarse en cuenta.

Existen paquetes para computadoras personales que emulan algunos de los dispositivos de despliegue gráfico que soporta PGPLOT, como son: VT125 y Tektronics con lo que se tienen más posibilidades de incluir el despliegue gráfico en el sistema.

3) Tratamiento Matemático.

En las gráficas de tipo matemático y estadistico es muy común aplicar a los datos experimentales algún tratamiento especial. Esto es con el fin de poder aproximar los datos a algún modelo teórico (regresión) o de poder apreciar un conjunto de datos a través de una curva suave (suavizamiento).

Algunos de los tipos de regresión más conocidos son; regresión lineal, regresión polinomial de orden N, regresión exponencial, regresión de potencial y logarítmica.

APENDICE A

ESTANDARES DE PROGRAMACION FORTRAN 77

1) Relativas al encabezado

   1) Encabezado de cada Rutina

```
!Descripcion breve
 ----------- -----
!renglon en blanco
!Siglas Autor Fecha Contribucion
 ------ ----- ----- ------------
```

   2) Colocación de cada encabezado

```
SUBROUTINE nombre

encabezado
---------
IMPLICIT NONE

subrutina
---------
RETURN
END
```

2) Palabras reservadas en mayusculas.

3) Las variables se escriben con mayuscula y minuscula para dar la separacion entre las palabras. Por ejemplo:

```
NumeroElemArreglo
```

4) Comentarios se escriben mayuscula-minuscula:

```
!Comentario
----------
```

5) Las variables se documentan a su lado derecho:

```
INTEGER*2       columna 40
                ----------
1   variable    !Comentario
    --------     ----------
\   /
 --
 ::
 4 espacios
```

Solo se documentan las variables importantes.

6) Emplear formato TAB pero solo al inicio de la linea. Las lineas de continuacion comienzan con un 1 despues del TAB.

7) Orden de declaración

   1) Las variables se agrupan en el siguiente orden:

```
CHARACTER
INTEGER
LOGICAL
REAL
```

   2) Las variables se ordenan alfabeticamente dentro de cada grupo:

```
CHARACTER
1   Avion,
1   Casa,
1   Zapato
```

   3) Las subrutinas se ordenan alfabeticamente.

   4) Los parametros se declaran antes de las variables y van alineados al encabezado del programa. Siguen el mismo orden de agrupamiento que las variables:

```
CHARACTER
INTEGER
LOGICAL
REAL
```

8) Los arreglos se dimensionan en la misma declaración de las variables.

9) Sangria:

   1) Las palabras reservadas CHARACTER, INTEGER, etc. del bloque de variables se sangran cuatro espacios con respecto a las de los parametros.

2) Los nombres de las variables o parametros se sangran cuatro espacios a la derecha de las palabras reservadas CHARACTER, INTEGER, etc.

10) Cada Common debe llevar la declaracion de sus variables, pero estas deben ir anidadas cuatro espacios:

```
COMMON /CMVTEntSal/
1   var1,
    ----
1   var2
    ----
    INTEGER
1       var1,
        ----
1       var2
        ----
```

Los COMMON van inmediatamente despues de la declaracion de los parametros, alineados con respecto a ellos, y antes de las variables.

11) Inicialización de variables es a la derecha de la misma. El comentario se baja al siguiente renglon en caso de ser necesario:

```
CHARACTER
1   Nombre/Gerardo Leon Braunschweriger/,
1               !Comentarios
                 -----------
                >
                :
                Columna 40.
```

12) Etiquetas, Formatos y Unidades Logicas:
1) Archivos: 1-9
2) Transferencias: 10-99
3) Formatos Salida: 1000-1999
4) Formatos Entrada: 2000-2999

13) No emplear TYPE, PRINT o ACCEPT.

14) Formatos que se repite su uso deben escribirse antes del RETURN. Formatos de uso unico se ponen dentro de la operacion de entrada-salida: '(A)'.

15) Conversiones internas de string a otro tipo y viceversa se hacen con un READ y WRITE interno.

16) Bloques de importancia se separan un renglon arriba y abajo y se antecede de un comentario con doble admiracion:

```
!!bloque importante
```

17) El CASE se implementa con:

   1) GOTO computado para valores enteros.

   2) IF-ELSE IF-ENDIF para el resto de los casos.

18) Convercion de nombres:

   1) De subrutinas: 111Nombre. Donde 111 es el nombre generico rutinas y Nombre, se combinan mayusculas y minusculas para separar palabras.

   2) De variables: Mayuscula y minusculas para separar palabras.

   3) De COMMON: C111Nombre. Donde la C indica que se trata de un common, 111 es el nombre generico del bloque de rutinas a las que pertenece el common y nombre, se combinan mayusculas y minusculas para separar variables.

19) Orden de los parametros en subrutinas es: Parametros de entrada, salida y mixtos:

```
SUBROUTINE A(parametrosentrada,salida,mixtos)
```

20) Impresión de textos se hace con formatos al final del modulo (formatos que se repite su uso).

21) READ y WRITE se deben escribir las palabras UNIT y FMT.

22) Cuando se hace un WRITE y un READ de más de una linea:

```
WRITE (UNIT=6,FMT=100)var1, var2, var3,
                      ----  ----  ----
1                     var4, var5
                      ----  ----
```

23) Separar con un blanco antes y despues del igual.

```
Impuesto = 3.56
```

24) Separacion de un blanco entre operadores logicos.

```
IF (Peso > 65 .AND.  Altura < 1.78) THEN
```

25) Escribir NOELSE.

BIBLIOGRAFIA

[1] Carl Machover, Mare Myers
"Interactive Computer Graphics"
IEEE Computer Graphics, octubre, 1984.

[2] William M. Newman, Robert F. Sproull
"Principles of Interactive Computer Graphics"
McGraw-Hill, segunda edición 1984

[3] "PRINTRONIX 600 Applications Manual"
Printronix, 1979.

[4] "Manual de Usuario Impresora DELTA 10-100 C.P.S"
Coorporacion Electrónica DELTA, S. A.

[5] "CITIZEN 120-D, User's Manual"
Citizen American Corporation, 1985.

[6] "DECwriter IV Graphic Printer, User Guide"
Digital Equipment Corporation, 1981.

[7] "User guide VT100"
Digital Equipment Corporation, 1981.

[8] Bruce Webster
"A Simple Windowing System"
BYTE, marzo y abril de 1986.

[9] Henry Simpson
"A Human-Factors Style Guide For Program Design"
BYTE, abril de 1982.

[10] The Handbook of Computers and Computing
Edited by Arthur H. Siedman and Ivan Flores
Capitulo 46 Ergonomics by Roy L. Chafin
Editorial Van Nostrand Reinhold Company 1984.

BIBLIOGRAFIA

[11] Richard Holcomb, Alan L. Tharp
"The Effect of Windows on Man-machine Interfaces"
(or openinig doors with windows)
Proceeding of the 1985 ACM Computer Science Conference
Agenda for Computing Research: The Challenge for
Creativity, 1985 marzo 12-14

[12] "VAX GKS/Ob. Software Reference Guide"
Digital Equipment Corporation, noviembre de 1984

[13] "Programming in VAX FORTRAN"
Digital Equipment Corporation, septiembre de 1984

[14] "VAX/VMS Run-Time Library Routines Reference Manual"
Digital Equipment Corporation, julio de 1985

[15] "PGPLOT Graphics Subroutines Manual"
Caltech Astronomy Institute.

[16] Alfred V. Aho, Jeffrey D. Ullman
"Principles of Compiler Design"
Addison-Wesley, 1979.

[17] "Manual de Usuario del Sistema Interactivo SER"
Centro de Cálculo de la Facultad de Ingeniería, 1987.