ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERIA ELECTRICA

"Diseno de Circuitos Impresos Asistido por Ordenador"

TESIS DE GRADO

Previa a la Obtencion del Titulo de

INGENIERO EN ELECTRICIDAD

ESPECIALIZACION: ELECTRONICA

Presentada por:

IVAN ERNESTO AMAT DIAZ

GUAYAQUIL, ECUADOR
1989

A G R A D E C I M I E N T O

Al Centro de Servicios de Computación de la ESPOL por las facilidades prestadas.

A las empresas HARDTEC y SINFO-Q por el apoyo de sus instalaciones.

Al Ing. HUGO VILLAVICENCIO, Director de Tesis, por su gran ayuda en la realización del presente trabajo.

D E D I C A T O R I A

- A mis padres, por su gran y constante apoyo en el trayecto de mi vida estudiantil, les dedico la culminación de mi carrera y el futuro de su hijo como profesional.

- A mis hermanos, para que persigan el esfuerzo y la constancia, les ofrezco compartir la alegría y la satisfacción de cumplir con una anhelada meta.

- A Germania, de quien he recibido la ayuda permanente, le dedico con cariño esta obra.

MIEMBROS DEL TRIBUNAL

ING. HERNAN GUTIERREZ V.

PRESIDENTE DEL TRIBUNAL

ING. HUGO VILLAVICENCIO V.

DIRECTOR DE TESIS

ING. JUAN CARLOS AVILES C.

MIEMBRO PRINCIPAL

ING. EDGAR IZQUIERDO O.

MIEMBRO PRINCIPAL

DECLARACION EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestas en esta tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Exámenes y Títulos Profesionales de la ESPOL).

IVAN ERNESTO AMAT DIAZ

RESUMEN

El desarrollo de programas de computación, responde a una necesidad actual de optimizar y aliviar pesados trabajos manuales. Así, antaño para crear un diseño de circuitos impresos, había que poseer talentos de estratega y de dibujante, por lo que el ingeniero electrónico debía recurrir a especialistas del ramo para obtener sus tarjetas.

Pero gracias a los ordenadores es posible hoy para cualquier usuario realizar estos diseños con la calidad de un profesional.

Para cubrir esta necesidad, se produjo un programa especialmente orientado a crear los diseños de las pistas, tan sencillo de operar que basta solo con ingresar los datos apropiados y obtener así los diagramas deseados.

El plan de elaboración de esta tesis está descrito en 4 capítulos bien definidos:

El capítulo I, le da un enfoque teórico a los "métodos clásicos" de fabricación, y contiene una recopilación de experiencias de optimización en la distribución de los componentes sobre la tarjeta impresa.

El capitulo II, entra de lleno a la "teoria de los algoritmos" con una descripción detallada de técnicas de administración de circuitos impresos, pasando por los distintos métodos de automatización en colocación, conección y edición de circuitos.

En el capitulo III, encontraremos ya el listado del programa obtenido con algunas de las técnicas del capitulo anterior aplicadas en varias rutinas. Ademas de la descripción de las rutinas más importantes, se acompañan los flujogramas , comentarios de funcionamiento y ejecución del mismo.

Por último en el capitulo IV, brindamos el manual de operación, donde incluimos un ejemplo práctico que acompaña la explicación del manejo del programa por parte del usuario.

CAPITULO III

EL PROGRAMA DEL SISTEMA

CAPITULO IV

EL MANUAL DEL USUARIO

CAPITULO IV

EL MANUAL DEL USUARIO

INTRODUCCION

El diseño de circuitos impresos es tal vez uno de los pasos más engorrosos con que cuenta la implementación de un diseño electrónico, puesto que su elaboración implica dotes de paciencia, habilidad y muchos borradores antes de obtener un diseño acorde a sus necesidades.

Ofrecer al diseñador de circuitos impresos una herramienta que ahorre tanto trabajo, es muy importante y necesario para optimizar su labor; y es ahí que nació la idea de utilizar un ordenador que nos dé una alternativa para acortar el tiempo que se invierte en dichos diseños.

Esta tesis pretende alcanzar dicho objetivo, usando un programa especialmente orientado para ayudar al diseñador a agilitar su trabajo. Esto se logrará convirtiendo al computador en una fuente de información para el diseñador principiante y en un hábil pizarrón de dibujo para el diseñador avanzado.

Finalmente, como producto de la realización de este sistema quedan las rutinas, trucos y métodos de programación que serán de gran ayuda a los seguidores del compilador C, quienes valorarán este programa por los recursos que utiliza del entorno gráfico y de sus periféricos.

C A P I T U L O I

TEORIA DEL DISEÑO DE CIRCUITOS IMPRESOS

1.1 OBJETIVO

Desde la aparición de las primeras tarjetas electrónicas que reemplazaron al enmarañado cableado, fue necesario aplicar métodos de confección de circuitos impresos; es así que con el paso del tiempo fueron apareciendo ciertas "reglas" que se fijaron como resultado de una mejor optimización de la tarjeta.

Este capítulo procura revisar algunas de aquellas características para una mejor confección, que aunque sean manuales son precisamente la base del diseño automatizado que pretendemos realizar.

1.2 INTRODUCCION

En esta tesis, la realización automática del diseño de circuitos impresos requiere de la intervención directa del diseñador.

Asi, durante el desarrollo del programa; además de ingresar los datos de los elementos y sus conecciones el usuario debe establecer también la posición de cada uno de ellos.

Como verá, una tarjeta impresa irá acorde a la disposición de los componentes y un diseño final mejorado dependerá de una mejor distribución de los elementos sobre la tarjeta.

Por lo tanto, este capitulo cubrirá aquellos aspectos teóricos necesarios para optimizar el montaje de componentes sobre el tablero.

1.3 DISTRIBUCION DE COMPONENTES DISCRETOS

Los siguientes pasos serán tomados en cuenta como prioritarios, para simplificar el proceso del diseño de circuitos impresos:

1. Todos los simbolos o designaciones de referencias usados por el ingeniero deberán ser entendidos como estándar.

2. En la fase de especificacion se identificará el tamaño del cuerpo del elemento (ancho y longitud máximo).

3. El tamaño del cuerpo se usará para la determinación del espaciamiento entre los ejes axiales de los elementos que se montarán sobre la placa.

4. Fijese en la polaridad indicada en todos los componentes, tales como: capacitores, diodos (indicando el cátodo) y de algunos transistores (indicando el emisor, base y colector).

5. Determine el diámetro del eje axial para seleccionar el tamaño del agujero en su esquemático.

6. Determine la orientación del componente visto del lado superior del mismo tal como el emisor, base

y colector del transistor, del potenciómetro o resistencia variable, indicando los pines 1,2,3; asi como capacitores variables, relés, transformadores, bobinas, etc.

7. Antes de orientar su esquemático al plano eléctrico o de interconecciones, vea los requerimientos de espacio. Ello conservará un espacio relativo para los diferentes tamaños de elementos sobre el tablero tipico. La fig. 1.3.a muestra un arreglo importante versus tamaño del cuerpo.

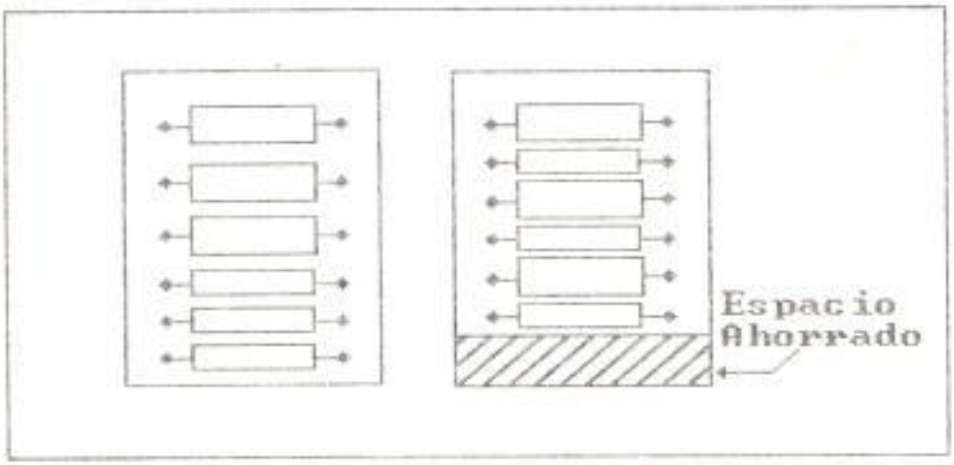


Fig. 1.3.a Aprovechamiento del espacio fisico

8. El esquemático de la fig. 1.3.b será usado para ilustrar el agrupamiento de componentes discretos.

   A. Asumimos que los pasos 1 al 7 fueron considerados para el tamaño de los componentes, revise que todos los componentes tengan designado un referencia propia.

10. El arreglo de la fig. 1.3.c es el mismo de la fig. 1.3.d pero con otra disposición de los componentes.

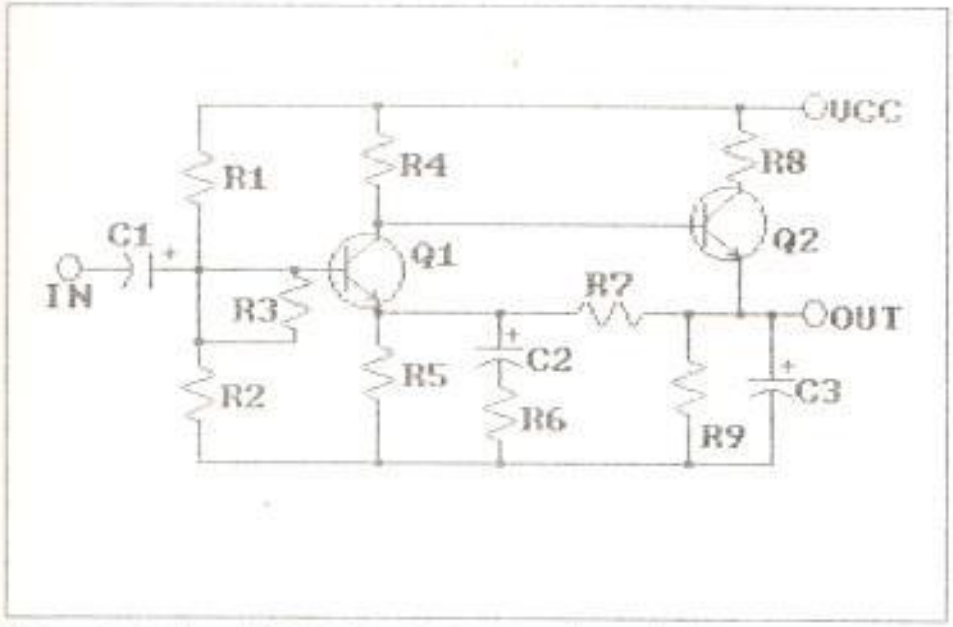


Fig. 1.3.f Segundo ejemplo de práctica

11. El esquemático de la fig. 1.3.f será usado como un segundo grupo de ejercicios de componentes discretos.

A. Asumiendo los pasos 1 al 7, considere los tamaños de los componentes y chequee que dichos componentes tengan una designación de referencia apropiada.

B. Nuevamente como en el paso 8B el layout inicial partirá del arreglo de los componentes que determinen el mejor lugar práctico y efectivo, eléctrica y mecánicamente hablando.

12. La fig. 1.3.g es el borrador del método correcto del esquematico de la fig. 1.3.f

13. El layout de la fig. 1.3.h muestra el camino equivocado de repartir los componentes, debido a que no se agrupó los elementos en una área cuadrada ó rectangular.

1.4 CARACTERISTICAS DEL MONTAJE DE LOS COMPONENTES SOBRE EL TABLERO

El montaje de componentes es un paso importante dentro de la fase de diseño de circuitos impresos. Esta importancia radica en especificar el número de componentes a ser montados sobre la tarjeta con sus respectivas dimensiones especificadas. Esto es complicado para algunos espaciamientos normalizados rigurosos que hacen del trabajo de montaje más dificultoso. Sin embargo, siguiendo algunas reglas básicas, dicho trabajo puede ser más sencillo.

Uno de los pasos de mayor cuidado físico es determinar cómo un componente estará montado sobre la tarjeta finalizada. Trate de imaginar un montajista tratando de introducir resistencias de 0.025 pulgadas de diámetro a través de agujeros de 0.020 pulgadas, las consecuencias serían desastrosas. Por lo que siguiendo los pasos que se indican mas adelante, Ud. no tendrá estas dificultades. La figura 1.4.a da el detalle del montaje de los componentes.

Los tres pasos en el montaje de componentes son:

- determinar el espacio entre terminales
- establecer el diámetro del agujero ;
- y fijar el diámetro del terminal impreso.

Fig. 1.4.a Descripción de un montaje sobre tarjeta

1.4.1 CALCULO DEL ESPACIO ENTRE LAS CONECCIONES

Tal como se ve en la fig. 1.4.a el espacio entre terminales de un componente es la distancia desde el centro del eje de un terminal hacia el centro del otro eje; esta distancia incluye la longitud del cuerpo del elemento más las porciones de sus respectivos ejes.

Hay tres maneras para determinar dicho espacio para un componente cualquiera:

1- El fabricante proporciona la información del montaje correcto en los catálogos respectivos, y en este caso no hace falta cálculo alguno.

2- Si el espacio entre terminales no es provisto por el fabricante, entonces chequee si la distancia del cuerpo del elemento es dado. En la mayor parte de los catálogos de los fabricantes se encuentran listados las dimensiònes máximas del caso en cuestión.

Utilizando dichas medidas máximas, un espaciamiento de los terminales podrá entonces ser determinado. Por ejemplo observando en los manuales las equivalencias de los capacitores cerámicos por su voltaje y capacitancia para hallar dicho espaciamiento.

3- El tercer método no es tan preciso como el primer o el segundo método. Si las dimensiones de los componentes montados o el espacio entre terminales no pueden ser hallados en catálogos del fabricante, dicho componente deberá ser dimensionado por el usuario. Intente recoger un modelo del fabricante o distribuidor. Una parte entonces ya es obtenida, y asi utilice la misma fórmula del método 2, y determine el espaciamiento de los terminales tomando en cuenta la máxima longitud requerida del cuerpo del elemento.

- Fórmula para el espaciamiento de terminales:

Para usar esta fórmula debe conocer el tamaño del casco del componente (el tipo de eje axial). Ver fig. 1.4.b

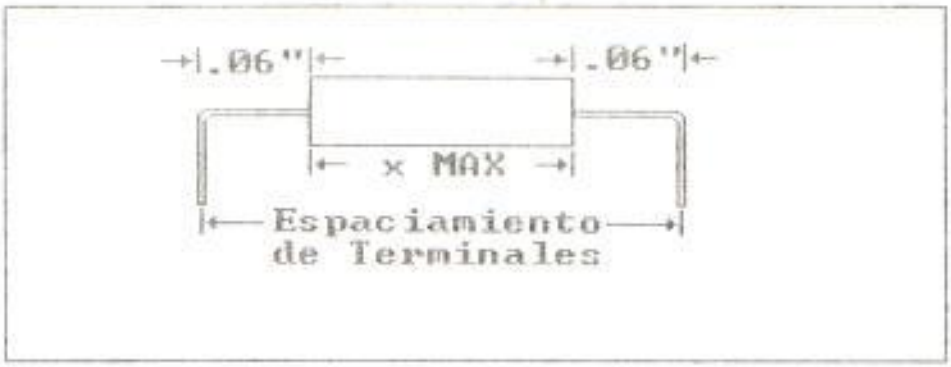


Fig. 1.4.b Dimensionado del espacio entre terminales.

Tome el cuerpo máximo "X"+0.06 pulgadas (1.5 mm) minimas de cada lado del cuerpo, (esto nos dará el minimo espaciamiento entre terminales).

Un punto importante que hay que tener en mente es que esta fórmula da el minimo espaciamiento. Por que la mayoria de los componentes deben ser extendidos cada 0.1 pulgadas (2,54 mm) en la escala 1/1, a esta acción se le denomina desplegar sobre rejilla.

Por ejemplo si los cálculos para el espaciamiento de un componente desarrollado

dan 0.46 pulgadas (1.16 mm) para crear una escala 2/1; el redondeo final será de 0.5 pulgadas (12.7 mm) en escala 1/1 ó 1.0 pulgadas (25.4 mm) en la escala 2/1.

1.4.2 DETERMINACION DEL DIAMETRO DEL AGUJERO

El diámetro del agujero y el tamaño del terminal dependen uno del otro. Usando la fórmula que se describe mas adelante el tamaño del agujero deberá ser determinado antes que la dimensión del terminal final. Por supuesto, antes de calcular el diámetro del agujero se debe conocer primero el diámetro del eje del componente a ser insertado.

Encontrar el diámetro del eje de un componente puede ser hecho por dos formas: Uno, observando el catálogo del fabricante y encontrando la lista de máximos diámetros y segundo a falta de información, obtener un "modelo actual" del elemento y dimensionar el diámetro del eje.

A continuación veremos entonces los cálculos que se requieren:

- Fórmula para calcular el diámetro del agujero:

Añada 0.006 pulgadas (0.15 mm) mínimo a 0.02 pulgadas (0.51 mm) máximo sobre el máximo diámetro del eje; esto será igual al tamaño mínimo del agujero tal como en la figura 1.4.c

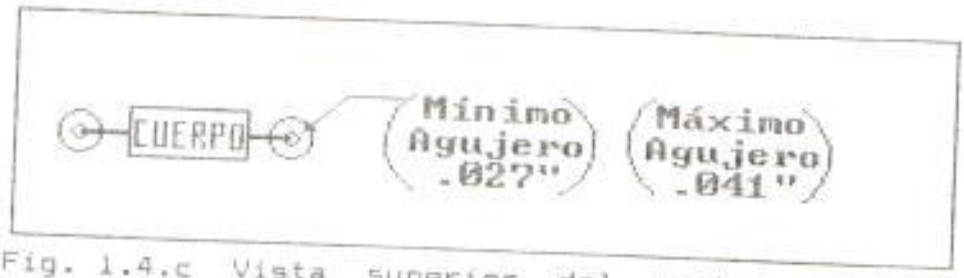


Fig. 1.4.c Vista superior del montaje de un componente

-Fórmula para calcular el tamaño del terminal:

Añada 0.020 pulgadas (0.51 mm) de anillo mínimo sobre el máximo diámetro del agujero. (0.20 pulgadas mínimo del anillo anular es solo una referencia). Así si nuestro cálculo nos da un diámetro del agujero de 0.37 pulgadas, el resultado será 0.37+0.40=0.77 pulgadas (1.95 mm) que es el mínimo tamaño del terminal a ser usado. Ver figura 1.4.d

Fig. 1.4.d Descripción del terminal.

1.5 DISEÑO Y CARACTERISTICAS DE LAS PISTAS

Aunque todo esquemático es no dimensionado, es una exepción para el montaje de componentes en circuitos impresos, donde la tarjeta debe poseer caracteristicas especiales debido a limitaciones de tipo fisico, electrónico, magnético, arquitectónico, etc.

Pero será necesario elegir un patrón o modelo de pista que sea lo suficientemente tolerante y que a la vez facilite el diseño de las tarjetas impresas. Asi, el esquemático de la tarjeta y el arte deberán ser hechos a la misma escala por razones prácticas. Sin embargo, deberán tenerse en cuenta la escala a la que haya sido diseñada (generalmente 2/1 ó 4/1), que luego podrá ser fotográficamente reducida a la escala 1/1.

De tal manera, un diseño de las pistas deberá ser ajustado a una rejilla que es en si un patrón normalizado de 0.050 pulgadas (1.27 mm) ó 0.10 pulgadas (2.54 mm) de ancho; que generalmente es utilizado en todo diseño, tal como se observa en la figura 1.5.a.

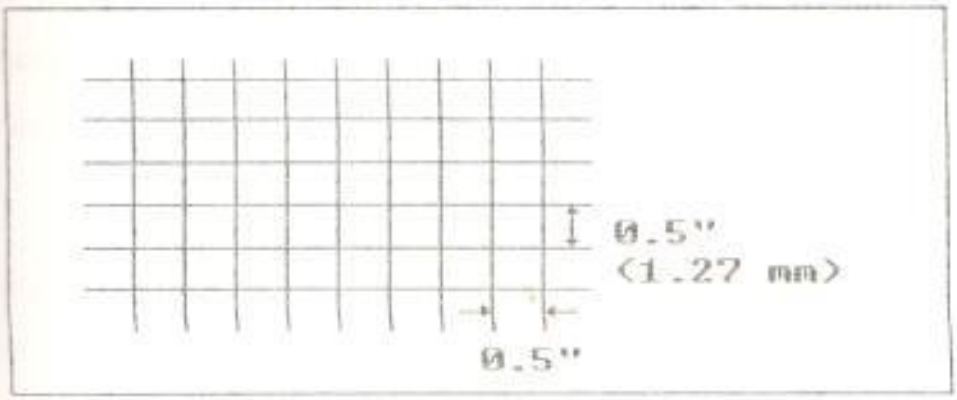


Fig. 1.5.a Rejilla normalizada.

Además de las limitaciones anteriores, será necesario tener en cuenta las rutas que seguirán las pistas, ya que ellas deberán estar conformes a algunas reglas básicas que vamos a describir.

Por ejemplo, el ángulo de los conductores serán de preferencia de 30, 45 ó 90 grados; y aunque las pistas pueden ser dirigidas casi a cualquier ángulo, es importante que las trazas corran paralelas a un mismo ángulo por razones de uniformidad.

El ángulo mínimo a la que cualquier pista puede ser colocado es de 90 grados. Angulos menores a 90 grados proporcionan situaciones inesperadas durante el proceso de fabricación, y pueden provocar en el proceso de grabado que soluciones queden dentro del ángulo con material de exceso. Un ejemplo de esto se observa en la figura 1.5.b

A fin de continuar con el normalizado de nuestras pistas, será necesario introducir el concepto de trazos de señal.

Un trazo de señal es generalmente considerado como otro mas, tal como los de fuerza o de tierra. Estas trazas son generalmente voltajes y corrientes bajas, además que ellos no requieren pistas o brechas de aire gruesas para una función apropiada.

Las trazas sobre placas son realizadas a menudo por cintas adhesivas especiales y aplicadas en el esquemático durante el proceso de diseño.

La mayoría de las cintas aceptadas por la industria hoy en dia es de 0.03-0.04 pulgadas (0.78 - 0.81 mm) en la escala 2/1 con 0.03-0.04 de brecha entre trazas o terminales adyacentes. En casos de tarjetas extremadamente densas no es muy común ver cintas con trazas de 0.031 pulgadas (0.78 mm) y algunas veces 0.026 pulgadas (0.66 mm) pero deberá tener en cuenta que las trazas y sus brechas deberian disminuir y el costo y la no fiabilidad de la tarjeta aumentar.

La combinación de 0.04 pulgadas (0.81 mm) de traza y brecha mencionadas antes proporcionan un conductor de 0.02 pulgadas (0.41 mm) sobre la tarjeta actual. Esta combinación proporciona un alto material para conducción de la electricidad y también la

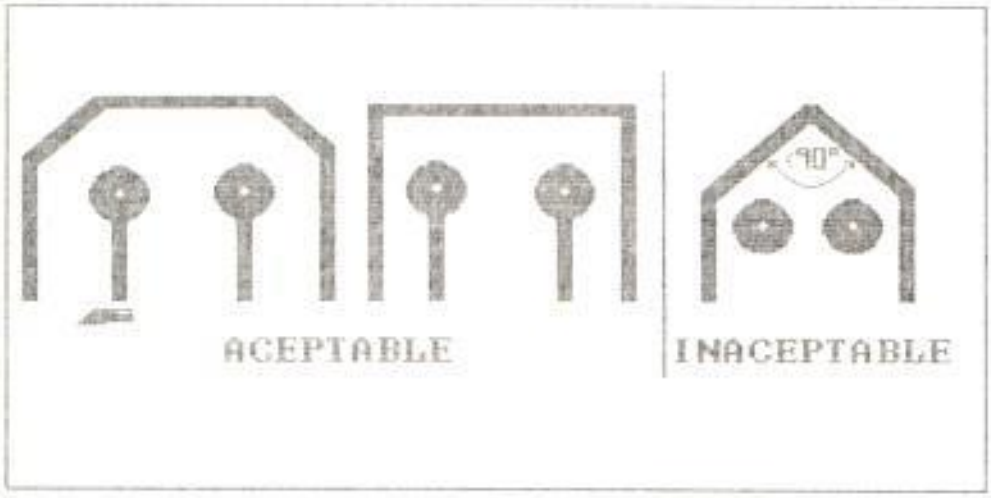


Fig. 1.5.b Angulos de las pistas

El mismo problema discutido en los ángulos de las pistas ocurrirá en las rutas de las pistas, si la unión de las pistas que se dirigen a un terminal vienen de un conjunto de direcciones distintas, pues durante el grabado se acumulará porciones de material no deseable en las uniones. Ver en la figura. 1.5.c

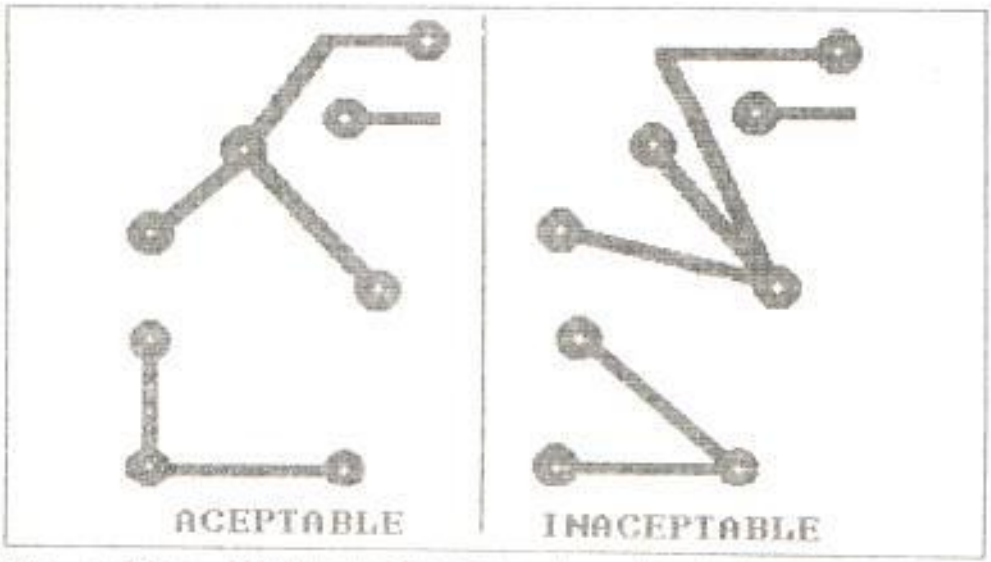


Fig. 1.5.c Comparación de rutas distintas

suficiente tolerancia para la fabricación de la tarjeta desde el diseño inicial a través de la fotografía, grabado, blindaje y el proceso de soldada. Cada una de estas etapas permite una cantidad aceptable de tolerancia de errores que no afectan la fiabilidad de la tarjeta final.

En casos donde los anchos de trazas sean inferiores a 0.26 ó 0.31 pulgadas (0.66 ó 0.78 mm) será muy critico intentar mantener al menos 0.040 pulgadas (0.81 mm) de brecha de aire, no así una brecha del mismo ancho como en la figura 1.5.d, pues algún puente se crearia en dicha brecha durante el proceso de soldada.

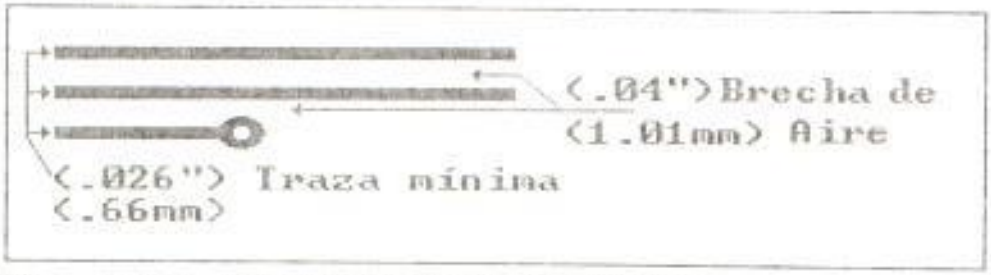


Fig. 1.5.d Dimensiones de las Pistas.

C A P I T U L O I I

EL SISTEMA DISEÑADOR DE CIRCUITOS IMPRESOS

2.1 OBJETIVO

El propósito de este capitulo, es el de dar un enfoque teórico a los algoritmos que forman parte de cualquier programa orientado al diseño de circuitos impresos, y como un caso especial, el nuestro.

Aunque nuestro programa por razones de sencillez no posee las virtudes de otros semejantes, este capitulo describe varias técnicas muy difundidas entre los sistemas CAD avanzados, por este motivo se entenderá que el contenido de los distintos subapartados tiene como objetivo principal el de proporcionar al lector de un material teórico relacionado al fundamento del diseño de circuitos impresos en general; y que parte de el será la base de nuestro sistema diseñador.

De esta manera se complementará a la "teoria clásica" descrita en el capitulo I y asi tener una visión global la finalidad del programa principal y sus rutinas más importantes.

2.2 INTRODUCCION

El trabajo final del diseñador electrónico consiste en la realización de un borrador del dibujo del esquema electrónico, o bien en confeccionar una lista de conexiones o incluso, en la de ambos. El borrador incluirá una lista que identificará todos y cada uno de los componentes electrónicos a utilizar; este o estos son los documentos finales del trabajo del especialista electrónico.

El diseñador de circuitos impresos parte de aquel o aquellos documentos, y su primera tarea suele ser la elaboración de un esquema de conexionado apropiado para la realización de los trabajos que seguirán. Las herramientas que utilizará el diseñador en esta tarea indican ya la diferencia entre la utilización de un sistema automatizado o no; mediante el uso de un sistema CAD el diseñador sustituirá lápices, plumas, borradores, papeles, etc. por dispositivos interactivos tales como tabletas gráficas, lápices luminosos, pantallas de uno o varios colores y, sobre todo, las facilidades de uso de todos estos elementos proporcionados por el sistema de programación apropiado.

Un sistema CAD es necesario para la automatización de la elaboración de esquemas de conexionado y

necesario para la delineación automatizada. Esta aplicación particular estará contituido fundamentalmente por un menú, de la que habrán de elegirse las diferentes opciones; independientemente del equipo utilizado y de la manera en que haya de hacerse, las opciones fundamentales deben afectar a tres clases de elementos:

1) Simbolos de componentes electrónicos.
2) Unión entre los terminales de los simbolos.
3) Tipos de textos o identificadores de elementos.

La manera normal de operar sobre un sistema de esta clase consiste en elegir algunos de los simbolos del menú, mediante el elemento interactivo utilizado; tableta, lápiz luminoso, ruedas, o en nuestro caso el ratón, y a continuación colocarlos en los lugares deseados, comprobándose toda la operación en la pantalla; después de colocar algunos simbolos, se realiza la conexión entre sus terminales, tras la elección del tipo de raya -continua, de trazos etc.- que haya sido seleccionada para hacer las uniones. El esquema de conexionado queda acabado cuando se identifican mediante nombres las conexiones deseadas y se introduce la información alfanumérica que se estima necesaria.

Para la realización de estos esquemas el diseñador debe utilizar un sistema provisto de las técnicas básicas de los sistemas de dibujo: la ya comentada selección en un menú, la utilización de una rejilla para forzar el acceso a los terminales de los simbolos y para facilitar el dibujo de lineas verticales y horizontales, y la utilización de una biblioteca de simbolos predefinidos. En esta última técnica (elección de los simbolos a utilizar de entre aquellos ya definidos), conduce a una normalización muy interesante, ya que se evita la utilización de diferentes simbolos por distintos diseñadores para referirse a la misma función, lo que a veces da lugar a esquemas incomprensibles.

Una vez obtenido el esquema del conexionado se ven aparecer las ventajas del diseño automatizado ya que toda la información contenida en el esquema a él asociada queda almacenada en el computador; de manera que cualquier información posterior que se realice en el esquema puede ser fácilmente introducida mediante las sencillas técnicas de borrado interactivo y pueden almacenarse tanto la versión original como las sucesivas modificaciones, con las facilidades que ello proporciona a las tareas de mantenimiento y puesta a punto de diversas versiones. Quizás la ventaja más interesante que aparece es el hecho de que todo lo que se haga a

partir de ahora será coherente; es decir que las técnicas informáticas se utilizará para asegurar que no hay discrepancias entre los documentos susecivamente generados, y que el diseñador no tendrá necesidad de realizar las verificaciones que deberia hacer si el diseño hubiese sido manual, verificaciones que suponen la utilización de una gran cantidad de tiempo.

El problema fundamental que queda por resolver es el de encontrar los caminos que unen los terminales de cada conexión (y esto para todas las conexiones), una vez que se hayan colocado todos los componentes sobre la tarjeta y se haya asignado cada función lógica en el esquema electrónico el componente o parte de componentes que la realiza fisicamente. Cualquiera de estas dos ultimas posibilidades (asignación de funciones y colocación de componentes) puede ser utilizada para facilitar la solución del problema inicial.

Evidentemente, puede plantearse el problema de trazado de manera global y utilizarse la asignación y la colocación como herramientas para resolverlo, pero esto no es posible hacerlo por las limitaciones de tipo práctico (tiempo de cálculo y capacidad de memoria que se plantean. En lugar de ello, se realiza una resolución secuencial de los problemas por este orden: asignación, colocación y trazado.

2.3 ESPACIO DE TRABAJO

No se ha dicho nada hasta ahora de las características físicas de la tarjeta que está definiéndose en cada caso concreto, pero evidentemente, en algún momento ha de hacerse.

La primera característica de la que hablamos, es la de la forma de la tarjeta impresa a diseñar; la forma más sencilla y al mismo tiempo la más usual es la rectangular, pero a veces se utilizan formas muy complicadas, que generalmente se obtienen de transformaciones de una rectangular. Por ejemplo, en una tarjeta que debe ir en un teléfono tipo "góndola" deberá definirse el espacio correspondiente al agujero por el que pasa el disco de marcar, mientras que algunas que deben ir instaladas en un cilindro, serán de forma circular.

Con un sistema interactivo, la forma es fácilmente definible mediante la utilización de un sencillo sistema de dibujo; además de la forma exterior deberán definirse los taladros que se utilizan para el paso de tornillos u otros elementos, pero el más interesante de la utilización de un sistema de diseño consistente en que, una vez que la forma ha sido definida, el programa de mecanizado para el control numérico de la fabricación del circuito podrá ser generado automáticamente.

El siguiente paso consiste en la definición de la tecnología de fabricación que se va utilizar y que influirá grandemente en el diseño. Son varias las concreciones que se han de hacer. La primera es el número de capas que poseerá el circuito impreso (huelgos de decir que cuanto mayor sea el número de capas, mayor suele ser la facilidad del diseño, aunque resulte más cara la fabricación), siendo los diseños más usuales aquellos que se realizan en tarjetas de dos caras. Otra cosa que se ha de concretar es la "clase" del circuito que se va a diseñar, indicándose con esto el número de pistas que podrán pasarse por décima de pulgada (unidad de medida que suele usarse, ya que es la separación entre patillas de circuitos integrados). También hay que definir las formas que adoptarán las perforaciones que han de realizarse sobre la tarjeta, pues existe una gran variedad motivada bien por su naturaleza - al corresponder a patillas de circuitos integrados ó a gruesos componentes discretos - e incluso por la moda, dando lugar estas formas a que haya que realizar cierta reserva de espacio cada vez que se introduce una perforación en el diseño.

2.4 COLOCACION DE COMPONENTES

El problema de la colocación de componentes sobre la tarjeta puede ser definido como "encontrar una disposición de elementos de manera que respetando las restricciones de tipo tecnológico, se posibilita el conexionado". La definición anterior parece un poco vaga pero es la correcta, ya que lo que se busca es hacer posible el trazado de las conexiones, aunque, cuando se utiliza algún método automático, éste sustituye la noción de posibilidad por otra semejante que sea cuantificable, como se verá más adelante.

Los métodos utilizados para obtener una colocación pueden ser clasificados en dos grupos totalmente diferentes: manuales, es el diseñador el encargado de decidir donde han de colocarse cada uno de los componentes; mientras que, cuando se utilizan métodos automáticos, es el computador el que decide donde ha de hacerse.

2.4.1 METODOS MANUALES

Los métodos manuales pueden ser clasificados a su vez en dos grupos: directos y realimentados.

En los métodos directos (que pueden ser interactivos), el operador coloca cada elemento en el lugar que quiere, teniendo en cuenta su experiencia y las restricciones de diseño (elementos que deben ocupar posiciones determinadas, otros que que deben estar muy cerca entre si, etc.).

En los métodos realimentados, el operador actúa de la misma manera que los directos, pero recibe información que utiliza para actuar nuevamente. El ejemplo más conocido es el sistema gráfico que presenta en la pantalla los elementos ya colocados y el que está colocándose destacado del resto (normalmente asociado al cursor del periférico interactivo) con unas líneas que parten de sus pastillas, o de su centro, lo unen a los elementos ya colocados con los que está conectado; estas líneas tienen el aspecto de ser elásticas, ya que al mover el elemento en cuestión por la pantalla mantienen la unión estirándose o encogiéndose. Estas líneas presentan una idelización de las conexiones, y el operador recibe una imagen del aspecto que puede que tenga estas conexiones. Evidentemente estos sistemas son muy espectaculares y diversos de utilizar, una

característica necesaria de los diseños, aunque su efectividad depende de lo que se acerque al conexionado ideal.

2.4.2 METODOS DE COLOCACION AUTOMATICOS

Como se ha dicho antes, con la utilización de estos métodos se obtiene una disposición de los componentes sobre la tarjeta sin participación del usuario; es decir, la experiencia de los diseñadores ha sido incluida en el software de un programa y, de alguna manera, se comporta como ellos ya que para su funcionamiento, hacen una estimación de cómo va a ser realizada cada cada conexión y buscan una colocación que la facilite. Veamos a continuación cómo se plantea y cómo se resuelve este problema.

2.4.3 BONDAD DE LA SOLUCION

Se dijo anteriormente que la colocación busca facilitar la realización del conexionado; por lo tanto, un análisis del conexionado que se obtenga proporciona una medida de la bondad de la colocación a posterior.

Los métodos automáticos hacen a priori una idealización del trazado y suponen que éste

hará de una manera determinada; sin preocuparse por los obstáculos que lo complicarán.

Son varias las suposiciones que suelen hacerse sobre la manera en que va a realizarse el conexionado de los terminales que forman cada una de las conexiones; veamos las más usuales:

1) mediante un árbol de longitud mínima cuyos vértices son los terminales que deben conectarse (árbol de expanción mínima);
2) mediante un árbol de longitud mínima cuyos vértices incluyen los terminales, pero que también pueden incluir vértices auxiliares (árbol de Steiner);
3) mediante un camino, cadena, que comience en uno de los terminales y acabe en otro, recorriendo el resto.

Una vez elegido el tipo de conexionado ideal que se va a utilizar, cada conexión puede quedar reducida a la medida de su longitud, y este número es utilizado para calcular la bondad de la colocación que esté probándose; bien por la simple suma de longitudes o por la suma de estas longitudes multiplicada por un factor especial para cada conexión, si se supone que unas son más críticas que otras.

La elección de un árbol u otro debe depender del método que luego se utilice para realizar el conexionado, de manera que la solución encontrada lo haya sido con un criterio semejante al que vaya a utilizarse en el trazado de las conexiones; de los tres expuestos, el que más se asemeja al proceso intelectual seguido por un diseñador o por un sistema basado en técnica de la inteligencia artificial es el llamado árbol de Steiner.

A efectos prácticos en cambio, el árbol de Steiner presenta varios problemas, siendo el más importante, y definitivo, el que no se conozca un método para calcularlo, cuando el número de puntos a conectar sea un poco alto (mayor de cinco); por lo anterior, suele utilizarse el árbol de expansión mínima, aunque también es necesario utilizar bastante tiempo de computador para calcularlo.

Existe un algoritmo sencillo para calcular el árbol de expansión mínima, que se puede describir de la siguiente manera:

1) Empezar a formar el árbol con cualquier terminal de la conexión;

2) Elegir un terminal ya incluido en el árbol y otro no incluido en él todavía, de manera que la distancia entre ambos sea mínima;

3) Incluir en el árbol el terminal elegido y que no estaba incluido;

4) Si el árbol incluye a todos los terminales acabar, si no, ir al paso número 2.

2.5 METODOS ENUMERATIVOS, CONSTRUCTIVOS E ITERATIVOS

El problema de la colocación puede ser enunciado como el establecer una aplicación inyectiva (no se puede hacer corresponder varios elementos a una misma posición) entre un conjunto de E elementos y P posiciones, de manera que se optimice el valor de una función que mida la bondad de la correspondencia.

Los métodos de colocación más sencillos conceptualmente son los de tipo enumerativo, que consiste sencillamente en un generador de todas las soluciones, al que se le añade un calculador de la bondad de cada una de ellas, de manera que al final se elija la mejor; el problema que presentan es que es imposible utilizarlos, dado el número tan elevado de soluciones que deben ser probadas. Por ejemplo si el numero de posiciones es igual al de elementos, el número de soluciones es factorial de P (p!).

Los métodos constructivos forman la configuración mediante la adición de un nuevo elemento a un conjunto de ellos, o nucleo, ya colocados; operan sobre un conjunto de elementos sin colocar, de entre los cuales extraen uno mediante un criterio de selección y, a continuación, lo añaden al nucleo en un lugar elegido mediante un criterio de colocación. Con estos métodos no se conoce la bondad de la colocación hasta que no se ha terminado la ejecución del algoritmo, pero tiene la ventaja de que son muy rápidos de ejecución y que la aproximación suelen ser suficiente en muchos casos.

Los métodos iteractivos parten de una configuración inicial, seleccionan un conjunto de elementos, e intercambian sus posiciones (haciendo intervenir en el juego a las posiciones vacias, si las hubiere), calculan la bondad de la nueva configuración y la comparan con la del anterior; si la nueva configuración es mejor que la antigua, la sustituyen y se la considera como inicial repitiéndose el proceso. En el caso de empate hay que definir un criterio de selección para saber qué hacer en cada caso, si elegir la nueva o parar.

2.5.1 METODOS CONSTRUCTIVOS

Como ya se ha dicho estos métodos van formando una solución tal como se hace una construcción, añadiendo a lo construido un nuevo "ladrillo" o elemento a la vez. Se diferencian unos de otros en los criterios seguidos para realizar las funciones de selección del siguiente elemento a colocar y de elección de la posición donde será colocado; los más conocidos son los citados en los siguientes subapartados.

2.5.2 COLOCACION POR PAREJAS AFINES

El criterio de selección utiliza una función que mide la conectividad entre cada dos elementos; esta función de pares (P) se define para cada pareja de elementos (X e Y) como el número de conexiones que existen entre ellos , es decir:

$$P(X,Y) = P(X,Y) = \text{número de conexiones entre X e Y.}$$

Para utilizar esta función, en cada momento el conjunto de elementos a colocar (E) se divide en dos subconjuntos; los colocados (C) y los no colocados (N), de manera que la función

P(X,Y) sea máxima al hacer variar (Y) entre todos los elementos ya colocados (C).

$$X,Y/P(X,Y) = \max. P(X_i,Y_i),\ X_i \in N,\ Y_i \in C$$

De esta manera quedan destacados del resto dos elementos uno de ellos (X) perteneciente al conjunto de no colocados, y otro (Y) al de los ya colocados.

Una vez elegido el elemento (X) hay que buscar un sitio donde colocarlo; para ello se elije el criterio de colocarlo en la posición libre más cercana al elemento (Y) que hizo máxima la función P(X,Y).

A continuación ha de extraerse (X) del conjunto (N) y ha de añadirse al (C), repitiéndose el proceso hasta que el conjunto de no colocados (N) quede vacio.

Todo este proceso comienza con la construcción del conjunto (C) inicial; para ello se utiliza uno de los siguientes criterios:

1) se elige la pareja (X,Y) de máximo valor P(X,Y) para formar el conjunto (C) y se la coloca en el centro de la tarjeta;

2) Si existiesen algunos elementos que debiesen ocupar una posición determinada, ellos constituirán el núcleo inicial de (C). Esto nos hace destacar el hecho de la existencia de algunos elementos que deben ocupar una posición fija - por ejemplo, conectores - y que no participan en el juego de la colocación, aunque su influencia sobre el resto queda de manifiesto al formar inicialmente el conjunto (C).

El tiempo necesario para la ejecución de un programa basado en este método es proporcional al cuadrado del número de elementos.

2.5.3 COLOCACION POR CRECIMIENTO EN RACIMO

Este método utiliza la misma función P(X,Y) que el método de las parejas afines y por tanto necesita la realización, de la participación del conjunto (E) en los colocados (C) y el de los no colocados (N); pero en lugar de buscar la pareja de elementos afines, busca el elemento no colocado más afín al conjunto de todos los ya colocados, mediante el uso de la expresión f(x).

$$X \in N/f(X) = \max \sum_{\forall} P(X,Yi),\ Yi \in C$$

El lugar elegido para colocar (X) debe ser aquella posición libre que sea la más cercana al centro de gravedad de las posiciones ocupadas por los elementos (Yi), asignando a estas posiciones un peso que indique la conectividad con el nuevo elemento; la fórmula que calcule el centro de gravedad ponderado así definido es:

$$ZG = \sum_{\forall} P(X,Yi).Zi \ / \ \sum_{\forall} P(X,Yi)$$

siendo (Zi) la posición del elemento (Yi) y extendiendo la suma a todo el conjunto de elementos colocados (C).

Este método comienza con la formación del núcleo inicial de la misma forma que el de las parejas afines y acaba como aquél al agotar el conjunto de elementos no colocados; así mismo, el tiempo de cálculo utilizado por éste método crece también con el cuadrado del número de elementos no colocados inicialmente.

Un detalle interesante de resaltar es que la función utilizada por ámbos métodos P(X,Y) ha de ser calculada una sola vez al comienzo y,

también, que esta función puede ser modificada, si se quiere tener en cuenta la diferente importancia asignada a cada conexión.

2.5.4 METODOS ITERATIVOS

Los métodos iterativos utilizan más tiempo de cálculo que los constructivos, pero tienen la ventaja de que pueden utilizar la función de bondad de la colocación que están analizando para controlar su propio proceso; si esa función está bien elegida y proporciona una buena medida de la calidad del conexionado o trazado de las conexiones que después se realizará, los resultados que proporcionarán serán mejores que los suministrados por aquellos.

Estos métodos deben partir de una configuración inicial que puede ser establecida por un método constructivo aleatoriamente o por un diseñador experimentado, luego transformarla en otra, comparar sus bondades estimadas y proceder en consecuencia; considerar la nueva como inicial, y repetir el proceso, o darlo por terminado.

En contra de estos métodos puede arguirse que sus resultados son muy dependientes de la dispocisión inicial y que conducen a resultados óptimos sólo localmente.

2.5.5 INTERCAMBIO DE PAREJAS

El funcionamiento de éste método consiste en la elección de dos elementos y en el intercambio de sus posiciones. La elección de los elementos que realizan el intercambio en cada momento da lugar a la aparición de variantes del método fundamental.

Una variante simple es la realización de un intercambio predeterminado entre posiciones o entre elementos, llamándose en este caso iteración a la serie completa de intercambios y tomándose la decisión de acabar o continuar el proceso al final de cada iteración.

Una variante consiste en definir las parejas y el órden de su intercambio de la siguiente manera: la primera está formada por el módulo o elemento con el mayor número de conexiones y el módulo con el menor número de ellas, la segunda con el siguiente con el mayor número y el siguiente con el menor, etc: hasta

completar las n!/2! parejas (siendo n el numero de elementos que pueden intercambiar sus posiciones).

Otro método sencillo conceptualmente está basado en una elección aleatoria de los elementos que permutarán sus posiciones.

Un método más interesante que los anteriores está basado en el establecimiento de una fórmula de bondad en la que aparezca destacada la aportación de cada elemento y utilizándose esta aportación para realizar el intercambio, al elegir aquel elemento que contribuye con en término menor de bondad estimada de la colocación actual. Existen dos maneras de utilizar este elemento: una de ellas consiste en buscar una pareja que aumente la bondad de colocación al hacer el intercambio, probándose una tras otra, y llamando a esto una iteración; la otra consiste en buscar la pareja para intercambiar de la misma manera, pero no interrumpiendo el proceso cuando se encuentre una que aumente la bondad, sino probando todas las posibles parejas y eligiendo aquella que más aumente (este método del gradiente utilizado en investigación operativa).

completar las n(n-1)/2 parejas (siendo n el número de elementos que pueden intercambiar sus posiciones).

Otro método sencillo conceptualmente está basado en una elección aleatoria de los elementos que permutarán sus posiciones.

Un método más interesante que los anteriores está basado en el establecimiento de una fórmula de bondad en la que aparezca destacada la aportación de cada elemento y utilizándose esta aportación para realizar el intercambio, al elegir aquel elemento que contribuye con en término menor de bondad estimada de la colocación actual. Existen dos maneras de utilizar este elemento: una de ellas consiste en buscar una pareja que aumente la bondad de colocación al hacer el intercambio, probándose una tras otra, y llamando a esto una iteración; la otra consiste en buscar la pareja para intercambiar de la misma manera, pero no interrumpiendo el proceso cuando se encuentre una que aumente la bondad, sino probando todas las posibles parejas y eligiendo aquella que más aumente (este método del gradiente utilizado en investigación operativa).

2.5.6 COLOCACION Y ASIGNACION CUADRATICA

Dado que el tiempo de cálculo depende del árbol utilizado para simular el conexionado, existe una aproximación de este conexionado por una función sencilla que transforma el problema de la colocación en otro conocido en otras áreas de la matemática y es llamado "problema de la asignación cuadrática".

Este método consiste en transformar cada conexión entre (n) elementos en la unión de todas las conexiones entre cada posible pareja de estos elementos, y suponer que esa pareja esté unida por un camino de distancia mínima, que normalmente es la distancia rectilínea.

De esta manera, la función que habrá que optimizar será:

$$\sum_{x} \sum_{y} P(X,Y).D(X,Y)$$

donde P(X,Y) coincide con la función definida anteriormente en el apartado dedicado a las parejas afines y D(X,Y) es la distancia que existe entre las posiciones asignadas a los elementos (X) e (Y). Normalmente la distancia utilizada es la rectilínea. Hemos de destacar que esta función no define una "bondad" de

colocación, sino lo contrario -podríamos llamarlo "maldad"- y que, por tanto, es una función que ha de minimizarse.

Esta aproximación al problema de la colocación por medio de la asignación cuadrática, es utilizado en combinación con el intercambio por parejas citado anteriormente y con otros como los de relajación, aunque hemos de resaltar que una solución óptima del problema de asignación cuadrática no tiene por qué ser óptima para el problema de la colocación.

2.5.7 METODOS DE RELAJACION

Hay unos métodos basados en la transformación a problemas de asignación que utilizan una analogía con los sistemas mecánicos de cuerpos sometidos a fuerzas de atracción; estos métodos se llaman de relajación, y suponen que los elementos se comportan como partículas unidas por resortes elásticos que ejercen una atracción entre ellos, descrita por la ley de Hooke, siendo el valor de la constante de atracción igual al de la función (P) de la que ya se ha hablado; por tanto, sobre el elemento (X) se ejerce una fuerza resultante igual a la

suma vectorial de todas las individuales debidas a la atracción de los demás elementos que son:

$$\overrightarrow{F(Y)} = \sum_{y} P(XY)\ \overrightarrow{D(XY)}$$

Si a un sistema de estas caracteristicas se le dejase oscilar libremente, alcanzaria una posición de equilibrio que seria la de minima tensión y que seria la solución al problema de asignación cuadrática.

Existen varios métodos que utilizan diversas estrategias para buscar esta posición, estando basados todos ellos en la búsqueda de la posición del punto de espacio en el que la resultante de las fuerzas aplicadas es nula, y colocando en esta posición si está libre; si no lo estuviese, ha de establecerse un criterio para probar otras posiciones, y estudiar después de un ciclo si se ha producido alguna mejora en la bondad de la colocación.

2.5.8 CONCLUCIONES DEL CRITERIO DE COLOCACION

Se han expuesto aqui una serie de métodos para obtener automáticamente la colocación de elementos que permite obtener fácilmente el conexionado, pero la pregunta que puede hacérsenos es: ¿Cuál es el mejor método para utilizar en la práctica?.

La respuesta a esta pregunta es: "no sabemos"....Coincidiendo esta contestación con la que puede extraerse de los sistemas comerciales que preguntan al usuario: ¿Cuál es el método que quiera que se utilice, el primero, el segundo, el tercero, etc?.

Para acabar este apartado, diremos que la colocación realiza una transformación de esta lista de conexiones entre terminales de componentes en otra lista de conexiones entre puntos definidos por sus coordenadas, y que esta lista es la que se utiliza para abordar el problema del trazado de conexiones.

2.6 EL CONECCIONADO

Una vez decidida la colocación de todos los componentes sobre la tarjeta del circuito impreso, nos enfrentamos al problema más difícil del diseño del mismo: encontrar el camino que une cada elemento de terminales que constituye una conección sin producir ningún contacto con el resto de los terminales y conecciones, respetando ciertas condiciones, como puede ser la anchura de alguna de estas pistas, e intentando minimizar la longitud total del trazado y el número de agujeros necesarios para realizar cambios de cara.

Como en el caso de colocación, existen dos maneras de aprovechar la colaboración del computador en el proceso de encontrar el conexionado: una de ellas es empleándolo como herramienta interactiva, y la otra como sistema automático de diseño. En cualquiera de los dos casos aparece una gran ventaja sobre los sistemas manuales, que es la seguridad que proporciona al usuario de que, si se ha establecido correctamente la lista de coneccionado, no se producirá la conección equivocada de terminales ni quedará alguno de éstos no conectado, sin que el sistema lo detecte y advierta.

2.6.1 TRAZADO INTERACTIVO

La herramienta fundamental que utilizan los sistemas interactivos sigue siendo un editor gráfico, como en el caso del establecimiento del esquema del conexionado lógico y en el de la colocación, de manera que su funcionamiento será semejante a aquellos, aunque con peculiaridades asociadas al problema que se va a tratar.

La base está constituida por una minuta en la que se pueden elegir una serie de opciones: unas relativas a las conexiones ya establecidas, y otras relacionadas con las nuevas.

La operación fundamental a realizarse sobre una conexión ya establecida es su "selección", que en un sistema gráfico se traduce por la distinción que se hace entre ella y las demás, normalmente haciéndola parpadear o cambiando de color, accediendo después a las mismas opciones que pueden realizarse en una nueva conexión. Estas opciones son la adición de nuevos componentes a la conexión, como tramos y perforaciones; el borrado de alguno de estos a su reposición si acaban de ser borrados; y su modificación.

Evidentemente, de la facilidad con que pueden realizarse estas operaciones elementales y sus variaciones dependerá la mayor utilidad de un sistema particular.

Hemos de señalar que el tamaño normal de cualquiera de las pantallas utilizadas (alrededor de 21 pulgadas de diagonal, hace que sea totalmente necesaria la utilización del efecto lupa para distinguir los detalles de la tarjeta. Además, en estos sistemas es muy fácil la utilización de una reticula que fuerce la separación mínima entre pistas.

2.6.2 ALGORITMO PARA LA DETERMINACION AUTOMATICA DEL CONECCIONADO

Es la parte más importante de todo el proceso de diseño de circuitos impresos, ya que a ella es a la que es necesario dedicar un mayor tiempo; aquí presentamos dos algoritmos: uno, que podemos llamar clásico, y otro basado en técnicas de inteligencia artificial.

2.6.3 ALGORITMO DE LEE

Casi todos los sistemas que poseen un algoritmo de conexionado automático están basados en el desarrollado por Lee en 1961, cuyos fundamentos presentamos a continuación.

El problema abstracto que trata de resolver, es el de encontar un camino que enlace dos celdas de un espacio cuadriculado, de manera que, partiendo de una de ellas (A), se alcance la otra (B) sin pasar por ninguna celda perteneciente a un conjunto de puntos prohibidos y obteniendo un camino de longitud minima; además, el camino sólo permite movimientos horizontales y verticales.

El procedimiento se basa en considerar la celda inicial (A) como un foco emisor de ondas que se propagan a las celdas más cercanas, es decir, a aquellas situadas a distancia unidad marcándose estas celdas con un 1; siguiendo en la analogia ondulatoria, las celdas marcadas con un 1 son consideradas como nuevos centros emisores que se propagan a los puntos cercanos que son marcados con un 2; el proceso de considerar cada punto alcanzado como un nuevo centro emisor se continua hasta que se alcanza el punto buscado (B) o hasta que ya no puede propagarse el "movimiento ondulatorio", lo que indica que los puntos no pueden conectarse. Aunque no se ha dicho explicitamente, las posiciones prohibidas no pueden ser alcanzadas y tampoco las ya marcadas; pudiendo verse el proceso en la figura.

Una vez alcanzado el punto buscado (B) (la marca a él asignada coincidirá con la distancia del camino necesario para llegar a él), ha de realizarse otro proceso, que es el de encontrar el camino propiamente dicho. El procedimiento utilizado es el siguiente: de la celda (B) marcada con la marca (m) ha de pasarse a otra vecina de marca (m-1), de ésta a otra de marca inferior en una unidad y asi sucesivamente hasta alcazar el punto inicial (A). Evidentemente, este procedimiento puede proporcionar más de un camino, ya que en muchos casos puede accederse a varias celdas vecinas que poseen el valor de marca inferior a una unidad exactamente; un criterio que suele utilizarse es el de no cambiar la direccion en este proceso, si es posible. El camino encontrado está especificado por el recorrido en este segundo procedimiento. (fig. 2.6.a).

2.6.4 UN SISTEMA BASADO EN TECNICAS DE INTELIGENCIA ARTIFICIAL PARA OBTENER EL CONECCIONADO

Dado que el problema de encontar el conexionado requiere de la utilización de su inteligencia por parte del diseñador que pretende resolverlo de una manera manual o

interactiva, se dice que el computador que también está intentando resolverlo se comporta de una manera más o menos inteligente: lo anterior ha llevado a utilizar las técnicas de inteligencia artifical en la realización de un sistema de diseño realizado en el Instituto de Automática Industrial del Consejo Superior de Investigaciones Científicas (CSIC), conocido con el nombre de sistema DOCIL.

El problema que se trata de resolver es el de encontrar una sucesión de "segmentos" que, partiendo de un terminal que se quiere conectar, alcance a otro de la conexión; a continuación, desde otro terminal - si lo hay - ha de encontrarse la sucesión que lo une a la solución encontrada anteriormente, llamada "conección parcial", y se repite el proceso hasta que se agotan todos los terminales que constituyen una conección. De esta manera el problema original, consistente en unir entre sí todos los terminales de una conección, ha sido reducido al unir cada uno de ellos a la conección parcial de los terminales ya unidos que, si bien no es coincidente con el original es más semejante que el de unir dos puntos.

El método consiste en "generar" los cuatro segmentos que parten del terminal a conectar, prolongándolos todo lo posible hasta alcanzar la cota más lejana del rectángulo envolvente de la conección parcial; si el segmento se aleja de este "rectángulo solución", se construye sólo de longitud unidad, siendo este un criterio general que siempre se empleará.

Si algunos de estos segmentos generados alcanzan algún punto de la solución, se acaba el proceso y la cadena de segmentos que une el terminal con la conección parcial se añade a ésta para volver a comenzar el proceso con otro terminal de la conección, si existe, o acabar con ella, si ya están todos conectados. (fig. 2.6.b)

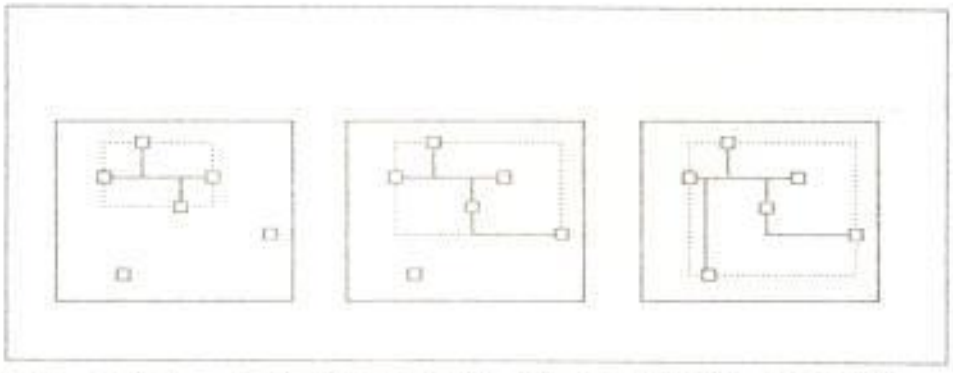


Fig. 2.6.b Crecimiento de la conección parcial

Si no se alcanza la solución han de generarse nuevos segmentos que puedan alcanzarla; estos segmentos nacen de los anteriores, por lo que los llamamos "sucesores", y lo hacen siempre

perpendicularmente a ellos; siguiendo el mismo criterio que con los iniciales, es decir, prolongándolos hasta alcanzar la máxima cota del rectángulo solución o dejándolo crecer sólo una unidad si se alejan de ella.

El proceso de generación de sucesores es que está controlado por una heurística que lo hace parecer inteligente, ya que, si fuese aleatorio o generase los sucesores de todos los segmentos creados, parecería dotado de una vana esperanza de transformarla un método enumerativo. La heurística consta de dos partes: la primera o de selección de elemento progenitor y la segunda o de creación de sus sucesores.

La elección del segmento progenitor se hace mediante una función de evaluación que asocia a cada segmento un valor que indica el coste estimado de alcanzar la solución a través de él: ésta función tiene en cuenta una serie de parámetros como son la distancia al terminal, la distancia a la solución, el número de taladros y, lo más importante, la capacidad de los sucesores para llegar a la solución. Esta facultad de estimar la capacidad de los sucesores para llegar a la solución la

llamamos "estado de tramo" y puede alcanzar cinco valores, con el siguiente significado (fig. 2.6.c, 2.6.d):

1) es posible que algun sucesor alcance la solución directamente;
2) no es posible que algun sucesor alcance la solución directamente, pero si en la vez siguiente y además por un camino de distancia estimada igual a la del estado 1;
3) para generar su sucesor el segmento debe modificarse a si mismo, ampliándose más allá de la cota de la solución, aumentado por tanto, la longitud estimada del camino que lleva a la solución;
4) los sucesores sólo pueden ser generados alejándose de la solución;
5) el segmento está "muerto" y, por tanto, es incapaz de generar sucesores.

La creación de sucesores también está controlada por el estado del segmento, de manera que no se generan todos los sucesores en un momento determinado, sino que en cada estado sólo se generan aquellos que han llevado a la suposición de que son los que tienen mayores posibilidades de alcanzar la solución.

Durante su existencia, los segmentos pasan sucesivamente por los diversos estados que han sido descritos, siguiendo el siguiente esquema:

a) un segmento se crea siempre en estado 1;

b) cuando el generador de sucesores realiza su trabajo con un segmento en estado 1, 2 ó 4, el segmento pasa después al estado sucesivo, independientemente del número de sucesores generados;

c) un segmento en estado 3 sólo da lugar a un sucesor, o ninguno, en cada ocasión en que es seleccionado, y permanece en este estado hasta encontrar un obstáculo que impida su prolongación, pasando al estado 4.

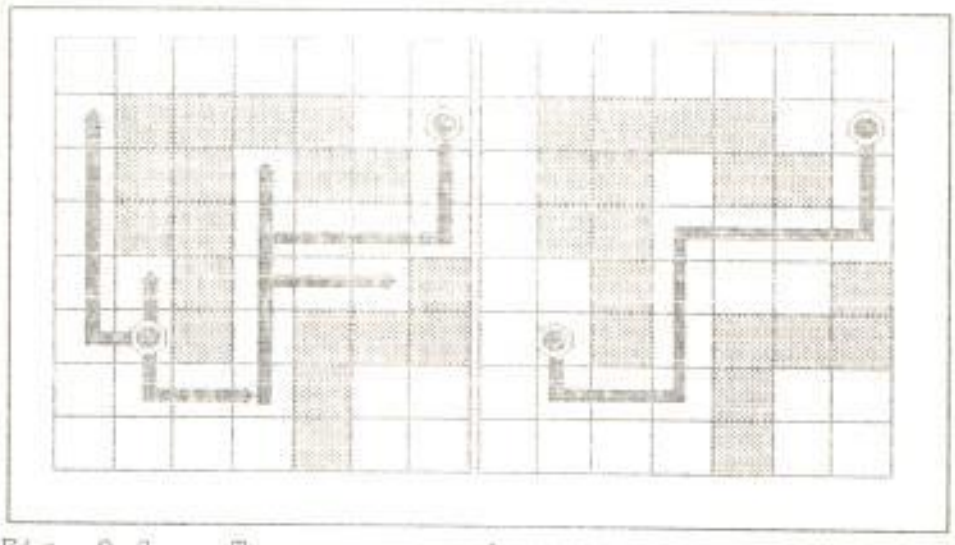


Fig. 2.6.e Tramos generados entre A y B

2.6.5 REALIZACION PRACTICA

Se ha construido un sistema basado en un algoritmo descrito en el apartado anterior, que acepta como entrada la lista de conexiones a realizar y que produce como salida un fichero con todas las soluciones y una lista de los terminales que no fue capaz de conectar y la referencia a la conexión en la que estaban incluidos.

La lista de conexiones puede establecerse mediante un sistema interactivo o mediante un lenguaje especializado, que además lleva aparejado un sistema experto en la realización de conexiones de memorias y buses.

El sistema general no es una herramienta que sustituye a un diseñador, sino una herramienta que necesita la cooperación de esta persona, ya que los resultados dependen de su experiencia que se concreta en la colocación de los componentes y en la ordenación de las conexiones, ya que éstas se realizan de la manera secuencial en que se le indica.

El fichero de resultados se utiliza para pasar al siguiente proceso que es el de la

fabricación, bien directamente o transformando y completando las soluciones obtenidas. En la actualidad se ha realizado un sistema que permite la modificación interactiva de un fichero solución por medio de una pantalla, de una manera semejante a la que se describió en el apartado correspondiente y otros diversos que realizan las transformaciones de datos necesarias para abordar las tareas de fabricación, como son, por ejemplo, la obtención del dibujo de las dos caras de una tarjeta superpuestas para inspeccionarlas, o separarlas para generar los clichés correspondientes o la creación de un fichero de taladros, etc.

EL SISTEMA OPERATIVO GEM

El paquete completo de GEM (Administrador del Entorno Gráfico), en su mayor parte escrito en C, representa una unidad de función orientada gráficamente. GEM ofrece todas las rutinas para la entrada y salida de datos, empezando por la edición de una cadena de texto sobre la pantalla, hasta el tratamiento de máscaras de entradas y salida, denominadas preferentemente como "formularios" bajo GEM.

Los responsables de estos trabajos son por una parte la "INTERFACE DE MAQUINA VIRTUAL" que llamaremos (VDI) y por la otra, la "RUTINA DE APLICACION AL ENTORNO" (AES). Detrás de estas denominaciones se esconden las partes responsables del manejo del hardware del ordenador, pudiendo considerarlo el programador como una caja negra. El VDI permite incluso la existencia de dos sistemas distintos de coordenadas, de los cuales uno direcciona un dispositivo normalizado matemáticamente exácto, de modo que a los programas que trabajan con estas coordenadas les es indiferente, si la salida se efectúa a través de una pantalla, o mediante una impresora. Con ello se explica ya el primer paso incondicional de cada una de las aplicaciones GEM, es decir la apertura de la estación de trabajo. Finalmente, el VDI deberá saber a donde y en que forma se emitirán los datos. Para el programador, la mayor ventaja es no tener ningún tipo de problema al copiar un programa a otro dispositivo de salida o bien a otro ordenador GEM, así pues, podrá decirse que las rutinas de entradas y salida no se modifica absolutamente nada.

Además, el VDI prevé llamadas del sistema para las distintas operaciones de gráficos, entre las que figuran así rutinas de caracteres (Polyline, Circle, Fill) y funciones de entrada y salida de textos.

finalmente, el AES procesa los datos que van llegando de todos los dispositivos posibles de entrada. A su campo de trabajo pertenecen la lectura del teclado, la valoración de los movimientos del ratón y el control del botón del ratón, así como el control de los distintos elementos de salida, como pueden ser las ventanas, los menu y los iconos. Además, el AES es el responsable del proceso ordenado de un sistema operativo multitarea.

2.7.1 LA INTERFASE DE MAQUINA VIRTUAL (VDI)

Bien visto, el VDI consta de dos partes, el Sistema Operativo de Dispositivos Gráficos (GDOS), que en definitiva contiene las informaciones sobre los juegos de caracteres a ser utilizados, o sea las fuentes. Asi pues deberá comunicarse p.ej. al VDI que la estación de trabajo del monitor SW dispone de dos rutuladores de color y que el campo de las coordenadas a direccionar se encuentra entre 0,0 y 639,399. Si por el contrario se indica como estación de trabajo un monitor color, sabiendo también que el poder de resolución de esta estación de trabajo no es demasiado elevado.

Así pues, es bastante evidente que cuando el programador determina una línea horizontal de una longitud de 600 puntos, ésta aparecerá sobre el monitor de colores con 320 puntos, que son los que se representa por línea, o sea, aproximadamente la mitad de la misma. Para poder ofrecer la mencionada independencia de los dispositivos, el VDI conoce dos sistemas de coodenadas distintos entre sí,

- NDC, Coordenadas de Dispositivo Normalizado, y
- RC, Coordenadas de barrido.

Mientras las coordenadas de barrido (raster) se corresponden con los puntos de imagen físicamente disponibles, y que en el caso del Atari se encuentran en el campo de 320x200 y 640x400 puntos, o que en el caso de graficadores se miden en pasos de x e y, las coordenadas "Normalizada" se refieren a una superficie de imagen imaginaria. Su orientación corresponde a nuestro acostumbrado sistema cartesiano de coordenadas, es decir, que el punto 0,0 se encuentra en el ángulo inferior izquierdo y el punto con los valores más elevados para x e y, se encuentran en el ángulo superior derecho de la superficie de caracteres.

De esta manera se extiende el campo de las coordenadas normalizadas de 0,0 a 32767,32767, lo que corresponde a una superficie de imagen de porte geométricamente correcto con una capacidad de resolución muy elevada (o a un tamaño adecuado).

El programador de GEM podrá elegir el sistema de coordenadas que desea utilizar. Si se decide por el NDC, los valores entrados se convertirán durante la ejecución del programa GDOS, parte correspondiente del GEM, a las coordenadas reales exitentes. Si se entra un cuadrado de longitud 100, aparecerá asi mismo como un cuadrado en el monitor. Sin embargo, si se decide por las coordenadas de barrido, el VDI no realizará ningún tipo de conversión. El promagrador o el programa en cuestión deberán considerar por si mismo todas las modificaciones a escala.

El sistema NDC presenta ventajas en conexión con el VDI, ante todo al intercambiar gráficos entre dispositivos periféricos distintos. De esta manera tendremos siempre dificultades con las coordenadas de barrido usuales y con el mando de impresoras con las que estabamos

acostumbrados, especialmente en lo concerniente a la relación entre longitud y anchura del dibujo. En un sistema de coordenadas de 600x200 la relación entre las coordenadas de la pantalla será aproximadamente 1:1.8; los rectángulos que aquí aparecen como cuadrados no aparecerá nunca más en una impresora como tales, siempre y cuando no se trabaje con el mismo sistema.

Sin embargo, en el caso de un direccionamiento al VDI, el programa controlador del dispositivo podrá efectuar las conversiones necesarias; las representaciones del tipo que fueren, aparecerán de inmediato en todos los dispositivos periféricos. Como ventaja puede ponerse de manifiesto el elevado tiempo que requieren tales programas; además, deberá convertirse la situación de cada uno de los puntos gráficos al sistema de coordenadas actual. Por este motivo se aconseja trabajar siempre con la coordenadas de barrido, a no ser que tenga que escribir necesariamente programas muy portables.

2.7.2 LA RUTINA DE APLICACION AL ENTORNO

El AES se compone de varias partes:

- las bibliotecas de subprogramas
- El despachador
- El Interpretador de Ordenes
- El almacenamiento de accesorios
- El Registro de menús/Alarma

El Registro de menús/alarma contribuye especialmente a la rápida función del GEM. De esta manera se intercalará dentro del registro de datos del menú, la parte de la pantalla que se sobrepondrá durante el desarrollo del menú.

Después de utilizar el menú correspondiente, un subprograma del AES se encargará de restaurar rápidamente el espacio de trabajo. Asi pues no será especialmente necesario que la aplicación, o sea en este caso Ud. como promagrador, se ocupe de esta acción. Aqui deberá únicamente memorizar que tan sólo se reservará el espacio en el que, como máximo, pueda intercalarse un cuarto del contenido de la pantalla. Por este motivo no deberia preocuparse de extender una tabla de avisos que ocupe la totalidad de la pantalla.

De modo similar se aplicará el Almacenador de accesorios que mas adelante nos interesará en

gran manera. Aquí no se almacenará unicamente datos, sino que en esta área de memoria reservada se cargarán programas auxiliares.

La misión del despachador consiste en que el procesador pueda acceder a los trabajos que deban ejecutarse a un mismo tiempo. La expresión "a un mismo tiempo" es un concepto relativo, ya que lo que para nosotros aparentemente se ejecuta al unisono, en una cadencia de 8 Mhz puede existir una distancia de millones de pasos.

Con el fin de no desperdiciar un valioso tiempo de cálculo, el despachador ejecuta dos listas: la "lista disponible", en la que se especifican todos los progaramas que se están procesando y que esperan la señal de la CPU, y la "lista no disponible", en la que se detallan todos los procesos que deben esperar a una determinada condición antes de ser ejecutados. Una de esas condiciones sería:

- una pulsación de teclas.
- el accionamiento de un mando de ratón.
- el movimiento del ratón.
- un mensaje.
- o la finalización de un determinado intervalo de tiempo.

Finalmente, la biblioteca contiene todos los subprogramas para el tratamiento de las ventanas, para la consulta y tratamiento del ratón, así como para la indicación de mensajes del sistema, como pueden ser los de las "cajas de diálogo" y de los menus. Incluso podrán encontrarse aquí las rutinas de consulta de las cajas de diálogo (OK, Cancel, Retry) y las funciones de mando para las ventanas.

Evidentemente, el programador que trabaje con GEM podrá acceder a estas rutinas. La biblioteca AES ofrece, entre otros, rutinas que se ocupan del programa a ser procesado, es decir, rutinas que se ocupan de que el programa de aplicación tenga preferencia entre otras tareas. El programador deberá constatar aquí, si el programa debe comportarse como una aplicación o bien como un accesorio.

El Gestor de Pantalla se hará cargo del control del ratón, tan pronto se encuentre fuera de la superficie de trabajo de la ventana momentáneamente activa. Como superficie de trabajo se define el contenido de la ventana; así pues, las líneas del título y de información no forman parte de esta superficie.

El Gestor de Pantalla se activará cuando el usuario abandone el campo comprendido en la parte superior de la ventana, o sea, cuando por ejemplo utilice la lista de menús. Supervisa todos los pasos del usuario y los protocolos, es decir, que mande el mensaje a la aplicación que es está procesando, para que dibuje nuevamente la ventana.

El interprete de órdenes que funciona en el sistema Unix: es en realidad una parte de las bibliotecas AES. Se encuentra situado después de la llamada del "sobremesa" y al principio de la "lista disponible". Es responsable de la llamada reglamentaria de una aplicación. De esta manera, al iniciarse el programa el "sobremesa" entrega al interpretador de órdenes la información sobre si se trata de una aplicación TOS o de una GEM; además, se le facilita el "Nombre del Archivo" para el subdirectorio (que en GEM aparece como archivo). Seguidamente finaliza el "sobremesa" y el interpretador de órdenes queda como responsable de la carga e inicio de la aplicación.

2.8 EL LENGUAJE C

La necesidad de acceder al Sistema Operativo GEM como parte de nuestro programa, nos llevó a seleccionar un lenguaje que sea capaz de integrarlo al mismo tiempo con todas sus características y ventajas que este ofrece. Por tal motivo, se eligió el lenguaje C, por ser el que reune las condiciones requeridas para nuestras exigencias.

El lenguaje C se lo define como un lenguaje de nivel medio, esto quiere decir que posee tanto propiedades de un lenguaje de alto nivel (Ej. Pascal); como propiedades de un lenguaje de bajo nivel como son los ensambladores de manera que pueden manejar bits y bytes además de acceder a funciones propias del computador.

El C se utilizó inicialmente para programar Sistemas Operativos (como el GEM por Ej.) y actualmente su uso se incrementa dia a dia por lo atractivo de sus cualidades de rapidez y portabilidad entre otras.

El lenguje C se lo define también como un lenguaje estructurado tal como Pascal; y este se basa en la utilización de bloques, y cada bloque representa un conjunto de sentencias que se relacionan lógicamente entre si.

Este lenguaje se basa en el concepto de construcción con módulos, que llamaremos funciones. Así, un programa C es una colección de una o más funciones. Por lo tanto para escribir un programa, primero hay que crear las funciones para finalmente unirlas.

A la vez una función es una subrutina que contiene una o más sentencias C, y que lleva a cabo una o más tareas. En un codigo C bien escrito, cada función lleva a cabo sólo una tarea. Cada función tiene un nombre y una lista de argumentos que recibirá la función. En general se le puede dar a cada una función el nombre que quiera, excepto el de main, que estará reservado para la función que inicia la ejecución del programa.

A pesar del papel importante que representa este lenguaje en esta Tesis, no es parte del mismo detallarlo con profundidad; pero si es de interes del lector relacionarse con el C, recomendamos observar la bibliografia para informarse de los libros que se dedican exclusivamente al tema.

C A P I T U L O I I I

EL PROGRAMA DEL SISTEMA

3.1 PARAMETROS UTILIZADOS EN EL PROGRAMA Y SIMBOLOGIA

El programa "Diseñador de circuitos impresos", fue realizado exclusivamente para el computador Atari ST utilizando como lenguaje el compilador C de la firma Megamax Inc. cuya librería de funciones nos permitió aprovechar todas las cualidades del entorno operativo GEM, que es la base del manejo del programa.

Una anotación importante del manejo del compilador fue precisamente compilar y encadenar, puesto que el tamaño de los archivos eran de aproximadamente 64 Kb y siendo que el compilador C administra hasta 32Kb se tuvo que dividir en varios archivos:

- DCI.C que contiene las principales rutinas.
- VARIOS.C conteniendo rutinas complementarias.
- FIG.C con los datos de las formas de los elementos
- TERM.C con los datos de los respectivos terminales
- EPSON.C con las rutinas de impresión.

los cuales fueron compilados por separado. Asi mismo, para encadenar (linker) los archivos (y con la limitación de 32K) se incluyó dos "overlay" dentro del programa para obtener finalmente el DCI.PRG

Los archivos complementarios DCI.RSC, DCI.H y DCI.DEF fueron creados por el MMRSC.PRG del compilador C y contienen las estructuras de menú, cajas de diálogo y etiquetas de presentación. Para una mayor información sobre la función de las estructuras, consulte el manual del compilador C.

Antes de pasar a describir las rutinas, veremos en este apartado el significado de las variables más importantes que forman parte del listado:

agu[300] Define los gráficos en impresora.

decmp[500] Contiene índice del fichero de los terminales de las conecciones destino.

depin[500] Contiene el número del pin respectivo.

dist[1000] Contiene la longitud de cada conección entre terminales.

estado[2500] Almacena el tipo de gráfico a imprimir.

filec[12] Nombre del archivo a grabar.

orcmp[500] Contiene índice del fichero de los terminales origen.

orpin[500] contiene el número del pin respectivo.

reloc[25] Datos temporales de ubicación de pines.

tro[1000] Coordenadas normalizadas de los terminales origen.

trd[1000] Coordenadas normalizadas de los terminales destino.

ratonx,ratony,tx,ty,posx,posy,mx,my Coordenadas del ratón.

gl_wchar,gl_hchar,gl_wbox,gl_hbox	son datos del
xdesk,ydesk,wdesk,hdesk	tamaño y
xcaja,ycaja,wcaja,hcaja	posicion
xsel,ysel,wsel,hsel	de las ventanas.
v_esquema,v_texto	Identifica las ventanas.
no_nec,no_neg	retorna valor innecesario
radio	Tamaño del terminal.
fin,final	Condiciona los lazos.
fila,col	Coordenadas para texto.
temp,temp1,i,j,m	Valores temporales.

```
struct tipo{
char cmp[5];                Tipo de elemento en libr.
char rem[4];                Nombre asignado.
char valor[7];              Contiene un comentario.
int modo,cmpx,cmpy;         identificador y coord.
}; struct tipo ficha[80];   Datos de los componentes.

struct tipo1{
char clase[5];              nombre en libreria.
char num[3];                identifica libreria.
int ancho,alto              dimensiones del elemento.
}; struct tipo1 lib[40];    Datos de la libreria
                            existente.
```

Para continuar: recomendaremos al lector revisar la simbologia y su descripción antes de estudiar los diagramas de flujo (ver fig. 3.1.a).

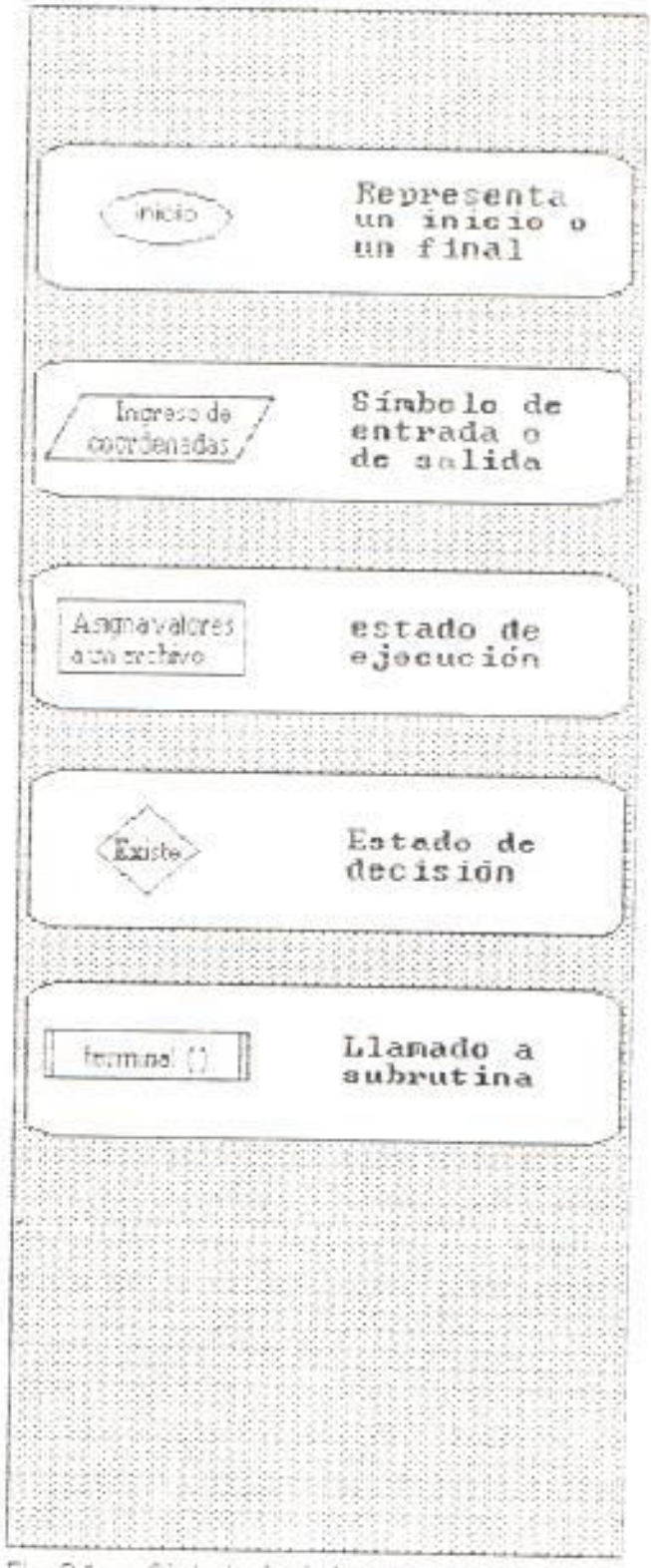


Fig. 3.1.a Simbología de los diagramas de flujo

3.2 PROGRAMA PRINCIPAL

En esta sección veremos las rutinas que forman parte del programa principal.

Cabe anotar que normalmente todo compilador C busca la función **main()** para comenzar a ejecutar el programa, por lo que nosotros decidimos no alterar dicho nombre.

Para una mejor presentación de las rutinas, cada función viene acompañada de:

- Una explicación del objetivo y funcionamiento de la rutina.
- Cuales son las funciones C utilizadas.
- El diagrama de flujo correspondiente.
- y el listado del mismo con los comentarios más importantes.

main()

Funciones Requeridas.- appl_init
graf_handle
wind_get
graf_mouse
menu_bar
graf_growbox
wind_open

Detalle.- Al inicio de la ejecución, el compilador C busca este nombre main() como fuente principal y de ahí parte a las diferentes rutinas que contenga, hasta su última instrucción antes de finalizar el programa.

Por eso, una manera sencilla de revisar el listado del programa, es comenzar por esta función y de ahí dirigirse a todas sus ramificaciones.

Diagrama de Flujo

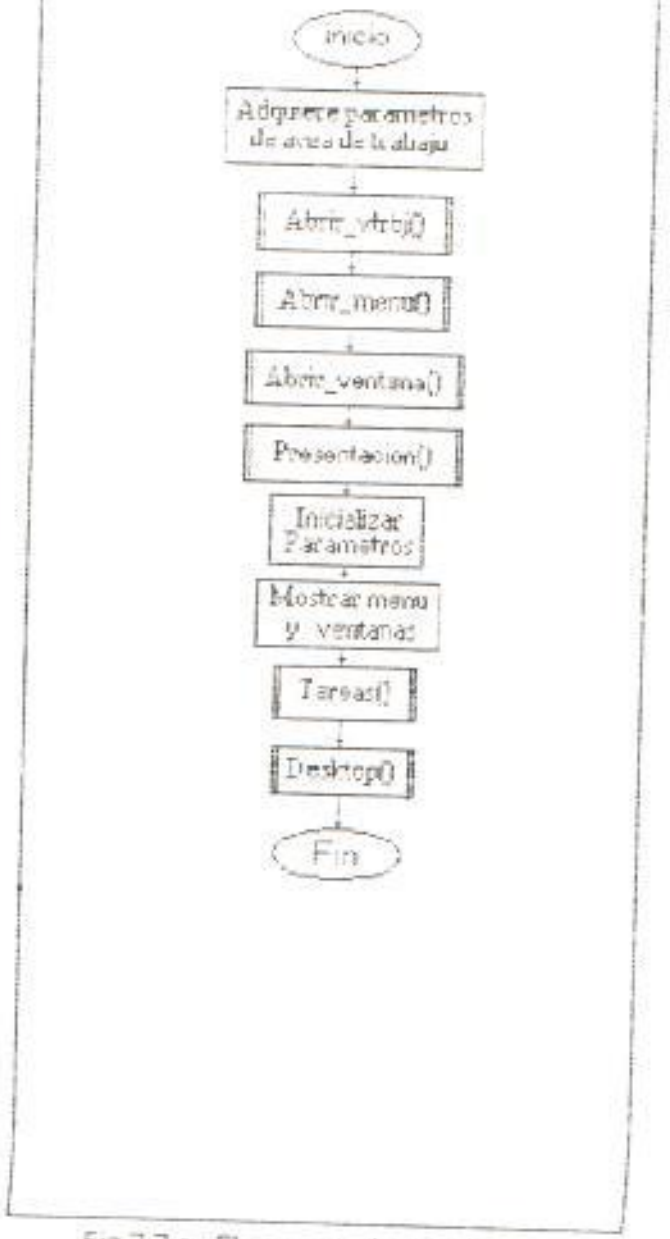


Fig 3.2.a Flujograma de main()

```
=appl_init();                    /* inicializado según estructurar array */
_handle=graf_handle(&gl_wchar,&gl_hchar,&gl_wbox ,&gl_hbox);
                                 /* manipulación del área de trabajo */
(0,WF_WORKXYWH,&xdesk,&ydesk,&wdesk,&hdesk);
();
();
();
();
);                               /* limpia área de trabajo */
W,&no_rec);                      /* define puntero de ratón */
_menu,VERDAD);                   /* presenta el menú en pantalla */
[xdesk+wdesk/2,ydesk+hdesk/2,gl_wbox,gl_ hbox,xdesk,ydesk,wdesk,hdesk);
                                 /* abre ventana menor */
_texto,480,ydesk,wdesk-480,hdesk);
                                 /* abre ventana mayor */
_esquema,0,ydesk,wdesk-160,hdesk);
                                 /* almacena dimensiones de la ventana mayor */
_esquema,WF_WORKXYWH,&xwork,&ywork,&wwork,&hwork);
```

abrir_vtrbj()

Función Requerida.- v_convwk

Detalle.- Esta función se encarga de adquirir los parámetros necesarios para abrir un puesto de trabajo; así, un arreglo llamado work_in[] asume los valores siguientes:

work_in[0] = # de dispositivo
work_in[1] = tipo de linea
work_in[2] = color de lineas
work_in[3] = tipo de marcas
work_in[4] = color de marcas
work_in[5] = tipo de texto
work_in[6] = color de texto
work_in[7] = patrón de relleno interior
work_in[8] = tipo de relleno
work_in[9] = color de relleno
work_in[10] = coordenadas RC

En un lazo asignamos el valor 1 a los 9 primeros arreglos,' que corresponden a las características normalizadas. Una vez establecidas dichas condiciones, se le da un nombre que identifique el área de trabajo; en este caso existen dos áreas llamadas handle y handle_tex.

Diagrama de Flujo

Fig. 3.2.b Flujograma de abrir_virtual

```
...h1()

...;                                              /* declara variable entera */
...=0; j<10; work_in[j++]=1);                     /* inicializa caracteristicas gráficas */
..._in[10]=2;                                     /* utiliza coordenadas RC */
...=phys_handle;
...(work_in,&handle,work_out);                    /* Crea una primer área de trabajo */
..._tex=phys_handle;
...(text_in,&handle_tex,text_out);                /* crea una segunda área de trabajo */
```

abrir_menu()

Funciones Requeridas.- rsrc_load
rsrc_gaddr
form_alert

Detalle.- Esta función se encarga de buscar si existe el fichero DCI.RSC definido bajo NOMBRE, si no lo encuentra envia un mensaje de error y aborta caso contrario lo direcciona en memoria bajo un puntero.

Dentro del fichero cargado se encuentran 3 estructuras:

ARBOL 0, que contiene el menu de comandos y que se direcciona bajo el nombre "arbol_selector".

ARBOL 1, contiene la tarjeta de presentación del programa y se direcciona bajo el nombre de arbol_presenta.

ARBOL 2, contiene un tarjeta de diálogo, y se direcciona bajo arbol_selector.

Diagrama de Flujo

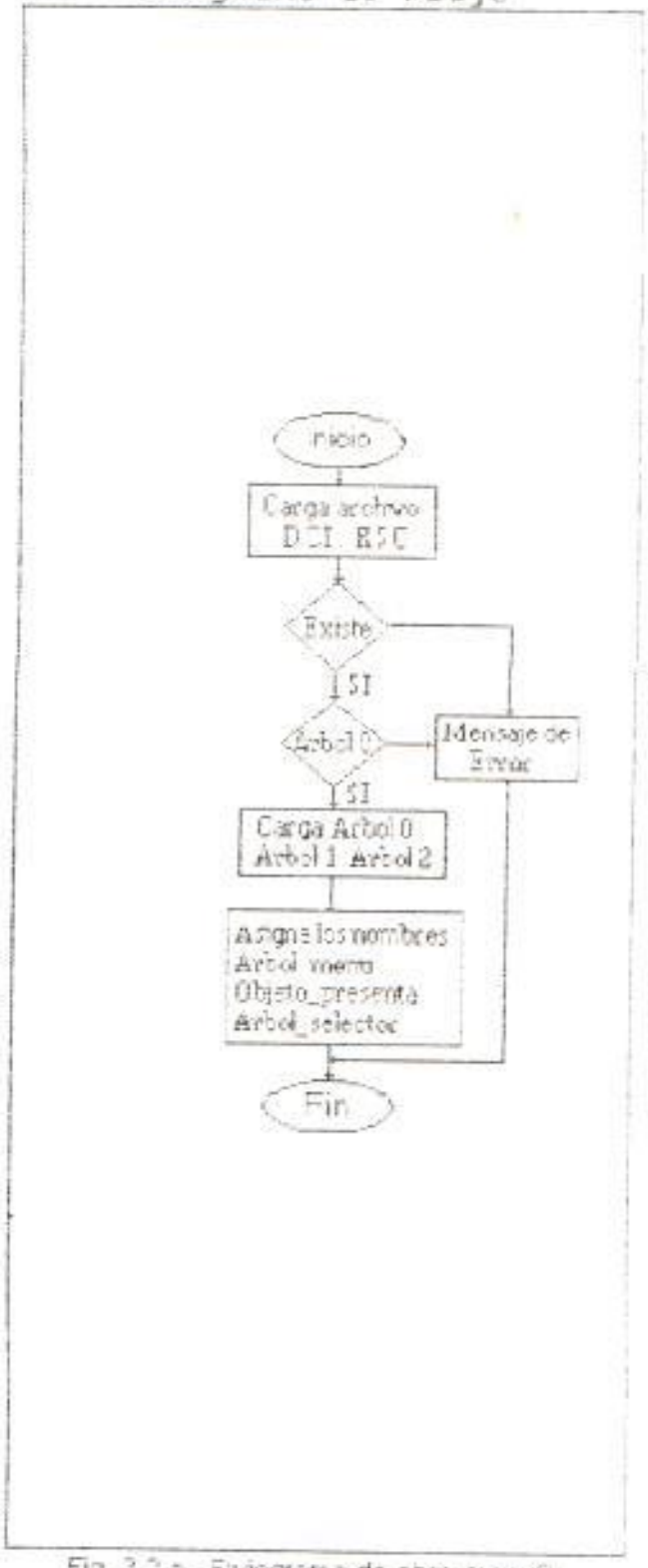


Fig. 3.2.5 Flujograma de abre_menu()

```
}

menu()

_load(NOMBRE))                          /* carga estructura desde el disco */

cador1();                               /* indica que no existe el archivo correspondiente */

_gaddr(B_TREE,ARBOL0,&arbol_menu)==0)   /* detecta si existe o no */

_alert(1,"[3][ERROR GRAVE!| Archivo indisponible][Aborte]");

_gaddr(B_TREE,ARBOL0,&arbol_menu);      /* carga en memoria estructura menú */

_gaddr(B_TREE,ARBOL1,&objeto_presenta); /* carga en memoria estructura presentación */

_gaddr(B_TREE,ARBOL2,&arbol_selector);  /* carga en memoria estructura selección */
```

abrir_ventana()

Funciones Requeridas.- wind_create

wind_set

Detalle.- Tal como se indica, esta rutina se encarga de crear las ventanas (no es lo mismo que displayar) y le proporciona las caracteristicas asignadas.

Asi, una ventana mayor llamada v_esquema estará definida por la constante TIPO_VENT, es decir que se según sus atributos podrá tener un lugar para los comentarios (INFO), un cuadro para cerrar la ventana (CLOSER) y un titulo de ventana (NAME).

mientras que la ventana menor v_texto estará caracterizada además por tener un desplazamiento vertical (VSLIDE | DNARROW | UPARROW).

Por ultimo se le asignan nombres y comentarios a las ventanas con la función wind_set.

Diagrama de Flujo

Fig 5.2.c Flujograma de abre_ventana()

```
wind_create(TIPO_VENT,xdesk,ydesk,wdesk,hdesk);     /* Crea y nombra una ventana mayor */
_esquema,WF_NAME,"Disenador de Circuitos Impresos",0,0);  /* comentario a la ventana */
_esquema,WF_FULLSIZE,500,0,0,0);
id_create(VENT_TEXT,xdesk,ydesk,wdesk,hdesk);       /* crea y nombra una ventana menor */
_texto,WF_NAME," ESPOL ",0,0);                      /* primer comentario de ventana menor */
_texto,WF_INFO,"Version 1.0",0,0);                  /* segundo comentario de ventana menor */
_esquema,WF_INFO," ",0,0);                          /* segundo comentario de ventana mayor */
```

tareas()

Funciones Requeridas.- evnt_multi
menu_tnormal
menu_ienable

Detalle.- Esta es la función más importante de todas en cuanto a la administración de los comandos, pues es en esta rutina donde se consulta el estado permanente del menu y cual ha sido el comando seleccionado.

Lo primero que hace es esperar algún evento en especial que puede ser de ratón, teclado o de mensaje. Esto se realiza atravez de la función evnt_multi; esta función retorna un valor en la variable msgbuff[0], esta indica la clase de evento, en nuestro caso el de selección de item de menú. Para saber cual de todos se eligió, solo se revisa y compara con el valor msgbuff[4].

Diagrama de Flujo

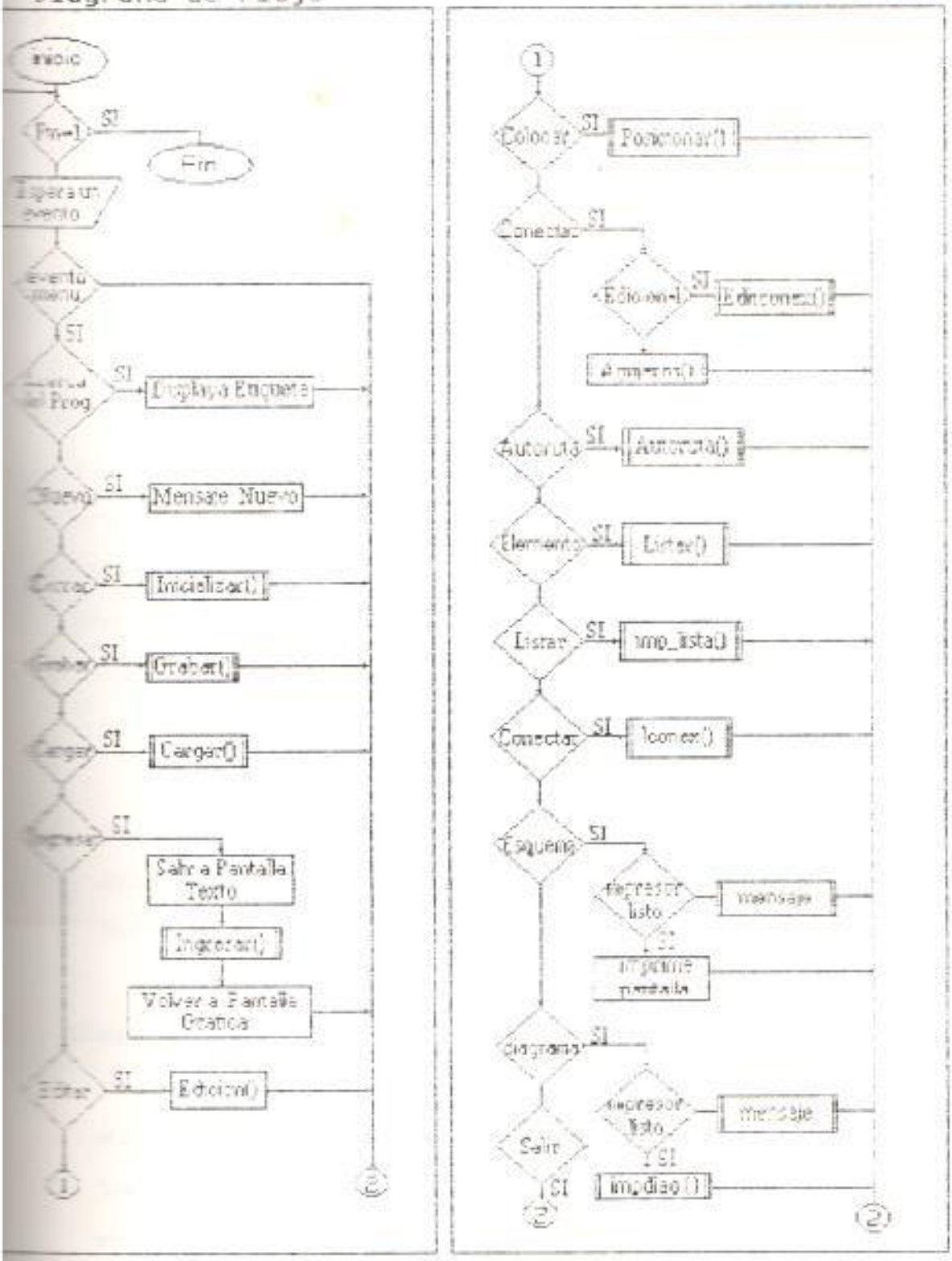


Fig. 3.2.e Flujograma de Iniciar()

```
while(VERDAD){                                  /* mientras la condición se cumpla ejecuta el lazo */
mesg=evnt_multi(MU_MESAG|MU_BUTTON|MU_M1|MU_KEYBD,1,1,1,bandera,0,0,wdesk-20,hdesk-20,0,0,0,0,0,
            msgbuff,0,0,&ratonx,&ratony,&no_nec,&no_nec,&no_nec,&no_nec);
if(mesg & MU_MESAG){                            /* realiza el lazo si hay un evento ó hay mensajes */
switch (msgbuff[0]){                            /* elige clase de evento seleccionado */
  case WM_REDRAW:                               /* evento si es que afectó a las ventanas */
    redibujar(msgbuff[4],msgbuff[5],msgbuff[6],msgbuff[7]);
    break;
  case MN_SELECTED:                             /* evento si es que afectó el menú */
    if(msgbuff[4]==MENSAJE1){                   /* ejecuta si seleccionó comando "Acerca del Prog." */
      raton_nuevo();
      form_alert(0,"[1][DISEÑO DE CIRCUITOS IMPRESOS| ASISTIDO POR ORDENADOR|Creado por:|
                Ivan Amat Diaz| (E.S.P.O.L. - Mayo 1989)][SEGUIR]");
      menu_normal(arbol_menu,msgbuff[3],VERDAD);
      graf_mouse(ARROW,&no_nec);
      break;
    }
    if(msgbuff[4]==ABRIRVEN){                   /* Si es que selecciona comando "Nuevo" */
      menu_ienable(arbol_menu,ABRIRVEN,FALSO);  /* deshabilita comandos */
      menu_ienable(arbol_menu,CARGARVE,FALSO);
      form_alert(1,"[1][Ud. va a crear un diseño nuevo| |Sige el menu <Edicion>][Seguir]");
      menu_ienable(arbol_menu,CERRARVE,VERDAD); /* habilita comandos */
      menu_ienable(arbol_menu,INGRESEN,VERDAD);
```

```
    menu_tnormal(arbol_menu,msgbuff[3],VERDAD);     /* restaura condicion del menú */
    ncomp=0;
    break;
  }
  if(msgbuff[4]==CERRARVE){                         /* comando "Cerrar" */
    mod=form_alert(0,"[3][Reiniciar sin Grabar. | si esta de acuerdo | oprima <Seguir>]
                [Seguir | Cancelar]");
    if(mod == 1){                                   /* realiza lazo si seleccionó caja "Seguir" */
      wind_set(v_texto,WF_NAME," ESPOL ",0,0);
      wind_set(v_texto,WF_INFO,"Version 1.0",0,0);
      wind_set(v_esquema,WF_NAME,"Diseñador de Circuitos Impresos",0,0);
      inicializar();
    }
    menu_tnormal(arbol_menu,msgbuff[3],VERDAD);     /* restaura condición del menú */
    break;
  }
  if(msgbuff[4]==GRABARVE){                         /* comando "Grabar" */
    grabar();
    interesq();
    administra();
    menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
    break;
  }
  if(msgbuff[4]==CARGARVE||msgbuff[4]==BORRARVE){   /* comando "Cargar" ó comando "Borrar" */
    fsel_input("A:\*.CAD",&filec,&no_recc);         /* Displaya directorio de archivos */
    if(msgbuff[4]==BORRARVE){                       /* en comando "Borrar" */
      if(no_recc != 0)    Fdelete(filec);           /* si presiona "Ok" borra el archivo elegido */
      interesq();
```

```
    if(no_recc == 0)    administra();
    menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
    break;
  }
  interseq();                              /* en comando "Cargar" */
  intertext();
  if(no_recc!=0){                          /* si presiona "Ok" */
    edicion=1;                             /* asigna modo edicion */
    inicializar();
    cargar();
    resultado();
    menu_lenable(arbol_menu,ABRIRVEN,FALSO);     /* deshabilita comandos */
    menu_lenable(arbol_menu,CERRARVE,VERDAD);    /* habilita comandos */
    menu_lenable(arbol_menu,ELEMIMP,VERDAD);
    menu_lenable(arbol_menu,CONECIMP,VERDAD);
    menu_lenable(arbol_menu,ESQUEIMP,VERDAD);
    menu_lenable(arbol_menu,IMPDOC,VERDAD);
    menu_lenable(arbol_menu,DIAGIMP,VERDAD);
    menu_lenable(arbol_menu,EDITARVE,VERDAD);
    menu_lenable(arbol_menu,AUTOSUTA,VERDAD);
  }
  if(no_recc!=0)    administra();          /* caso contrario vuelve a la pantalla anterior */
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}
if(msgbuff[4]==INGRVEN){                   /* comando "Ingresar" */
  if(edicion!=1)    menu_lenable(arbol_menu,INGRVEN,FALSO);
```

```
menu_bar(arbol_menu,FALSO);                    /* cierra el menu */
wind_close(v_texto);                           /* cierra ventana menor */
wind_close(v_esquema);                         /* cierra ventana mayor */
no_raton();                                    /* esconde el ratón */
v_enter_cur(handle);                           /* ingresa a la pantalla no gráfica */
ingresar();
v_exit_cur(handle);                            /* sale de la pantalla no gráfica */
menu_bar(arbol_menu,VERDAD);                   /* abre menú */
wind_open(v_texto,480,ydesk,wdesk-480,hdesk);  /* abre ventana menor */
wind_open(v_esquema,0,ydesk,wdesk-160,hdesk);  /* abre ventana mayor */
si_raton();                                    /* muestra ratón */
administra();
menu_ienable(arbol_menu,ELEMIMP,VERDAD);       /* habilita comandos */
menu_ienable(arbol_menu,POSICVEN,VERDAD);
menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
break;
}
if(msgbuff[4]==EDITARVE){                      /* comando "Editar" */
menu_ienable(arbol_menu,EDITARVE,FALSO);
menu_ienable(arbol_menu,IMGPEVEN,VERDAD);
edicion=1;                                     /* entra en modo edición */
form_alert(1,"[1][  Elija en el menu ' ' el tipo de Edicion deseado][Seguir]");
menu_ienable(arbol_menu,ELEMIMP,VERDAD);       /* habilita comandos */
menu_ienable(arbol_menu,CONECIMP,VERDAD);
menu_ienable(arbol_menu,POSICVEN,VERDAD);
menu_ienable(arbol_menu,CONEIVEN,VERDAD);
menu_ienable(arbol_menu,AUTOBUTA,VERDAD);
```

```
    menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
    break;
  }
  if(msgbuff[4]==POSICVEN){                    /* comando "Colocar" */
                                               /* deshabilita si no está en modo edición */
    if(edicion!=1)    menu_ienable(arbol_menu,POSICVEN,FALSO);
    adm=2;                                     /* identificador de rutina */
    posicionar();
    menu_ienable(arbol_menu,ESQUEIMP,VERDAD);
    menu_ienable(arbol_menu,CONEXVEN,VERDAD);
    menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
    break;
  }
  if(msgbuff[4]==CONEXVEN){                    /* comando "Conectar" */
    adm=3;                                     /* identificador de rutina */
                                               /* deshabilita si no esta en modo comando */
    if(edicion!=1)     menu_ienable(arbol_menu,CONEXVEN,FALSO);
    if(edicion==1){                            /* ejecuta si está en modo edición */
      editconex();
      menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
      break;
    }
    radio=2;                                   /* asigna radio de terminales */
    intertext();
    agujeros();
    menu_ienable(arbol_menu,AUTOBOTA,VERDAD);
    menu_ienable(arbol_menu,GRABARVE,VERDAD);
    menu_ienable(arbol_menu,CONSCIMP,VERDAD);
```

```
    menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
    break;
  }
if(msgbuff[4]==AUTORUTA){                          /* comando "Autoruta" */
                                                   /* deshabilita si no está en modo edición */
    if(edicion!=1)    menu_ienable(arbol_menu,AUTORUTA,FALSO);
    interesq();
    intertext();
    autoruta();
    adm=1;                                         /* identificador de rutina */
    menu_ienable(arbol_menu,DIAGIMP,VERDAD);
    menu_ienable(arbol_menu,CONECIMP,VERDAD);
    menu_ienable(arbol_menu,IMPDOC,VERDAD);
    if(edicion!=1)    menu_ienable(arbol_menu,EDITARVE,VERDAD);
    menu_ienable(arbol_menu,GRABARVA,VERDAD);
    menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
    break;
  }
  msgbuff[4]==ELEMIMP){                            /* comando "Elementos" */
    lister(0);
    menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
    break;
  }
  msgbuff[4]==IMPDOC){                             /* comando "Listar" */
    imp_lista();
    menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
    break;
  }
```

```
if(msgbuff[4]==CONECIMP){                          /* comando "Conección" */
  Iconex(0);
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}
if(msgbuff[4]==ESQUEIMP){                          /* comando "Esquema" */
  mod=form_alert(1,"[1][Para Imprimir: |  Pulse <Seguir> ] Seguir|Cancelar]");
  if(mod==1){                                      /* si selecciona "Seguir" */
    while(!Bcostat(0)){                            /* y la impresora no está lista */
      form_alert(1,"[3][La Impresora no esta lista ! | Revise sus conexiones][Seguir]");
      break;
    }                          /* fin de while */
    if(Bcostat(0)==-1){                            /* y la impresora está lista */
      resultado();
      no_raton();
      Setprt(4);
      Scrdmp();
      si_raton();
  } }                          /* fin de if */
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}
if(msgbuff[4]==DIAGIMP){                           /* comando "Diagrama" */
  mod=form_alert(1,"[1][Para Imprimir: |  Pulse <Seguir> ] Seguir|Cancelar]");
  if(mod==1){                                      /* si selecciona "Seguir" */
    while(!Bcostat(0)){                            /* y la impresora no está lista */
      form_alert(1,"[3][La Impresora no esta lista ! | Revise sus conexiones][Seguir]");
```

```
            break;
          }
          if(Bcostat(0)==-1){                    /* y la impresora esta lista */
            lapdlng();
        } }
        menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
        break;
      }
      if(msgbuff[4]==QUIT){                      /* comando "Salir" */
        wind_close(v_texto);                     /* cierra ventana menor */
        wind_close(v_esquema);                   /* cierra ventana mayor */
        wind_delete(v_esquema);                  /* borra ventana mayor */
        wind_delete(v_texto);                    /* borra ventana menor */
        menu_bar(arbol_menu,FALSO);              /* cierra menu */
        rsrc_free();                             /* libera memoria de las estructuras */
        v_clsvwk();                              /* cierra ventana de trabajo */
        appl_exit();                             /* sale al sistema operativo */
        exit();
        break;
      }
      break;
  } /* final del switch */
} /* final del if */
final del while */
de tareas */
```

desktop()

Funciones Requeridas.- wind_close

menu_bar

v_clsvwk

appl_exit

Detalle.- Esta es la última función en ejecutarse y se encarga de cerrar todas las estructuras árbol que se utilizaron durante la ejecución del programa.

Esta es la forma de limpiar la memoria y de reiniciar otro trabajo. Es importante hacerlo, pues de lo contrario el sistema operativo GEM quedaría alterado y habría errores de operación en el mismo u otros programas.

Diagrama de Flujo

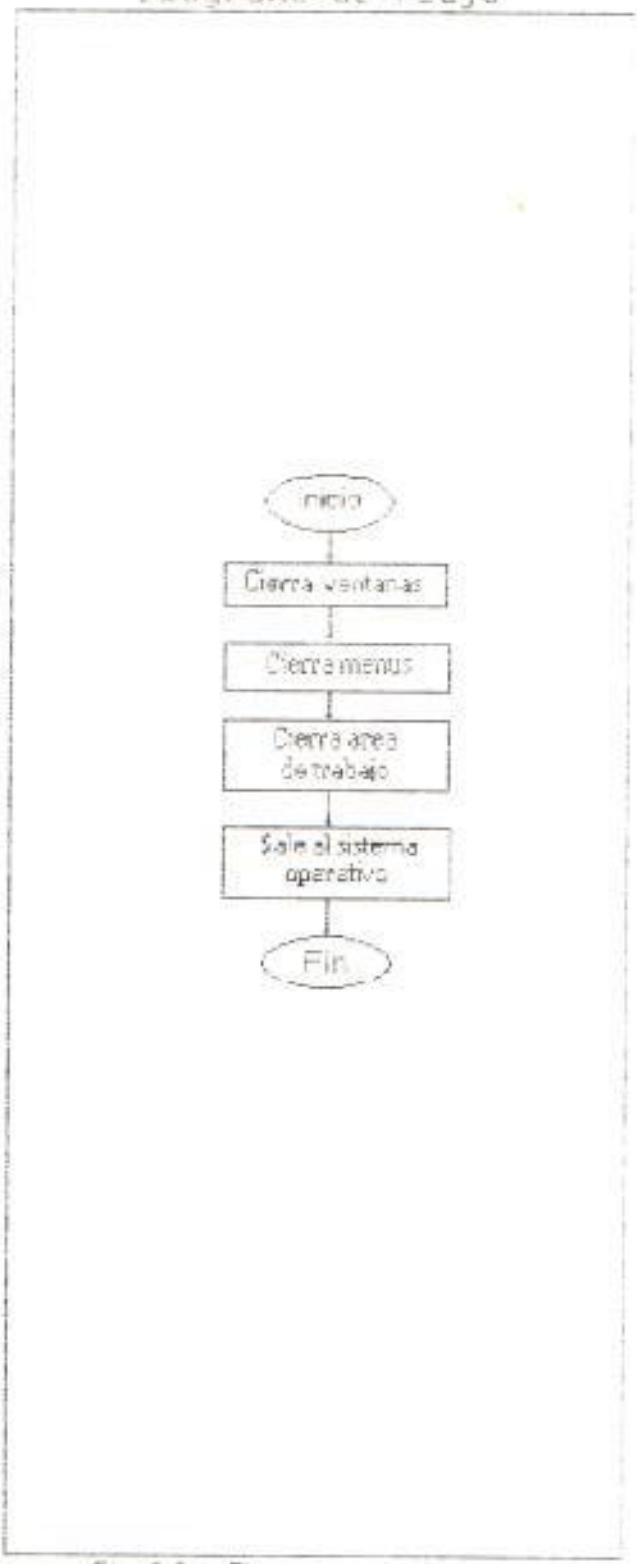


Fig. 3.2.1 Flujograma de desktop1

```
close(v_texto);                    /* cierra ventana menor */
close(v_esquema);                  /* cierra ventana mayor */
load(arbol_menu,FALSO);            /* cierra el menú */
area();                            /* cierra área de trabajo */
exit();                            /* sale al sistema operativo */
```

cargar()

Funciones Requeridas.- fopen

fscanf

fclose

Detalle.- Este es la rutina que se encarga de adquirir los datos necesarios desde el disco de almacenamiento.

Así, lo primero que realiza es declarar un puntero FILE que identifique el archivo como lectura; luego abrirá y comprobará si realmente existe ese fichero llamado, si no existe alertará su ausencia.

Mediante la función fscanf se realizará el llamado al primer dato que representa el número de componentes almacenados y con un lazo integrará todos los datos relacionados a los elementos.

El siguiente paso será llamar al dato que representa el número de conecciones y con otro lazo llamar a todos los datos de las conecciones.

Esta rutina termina cerrando el archivo.

Diagrama de Flujo

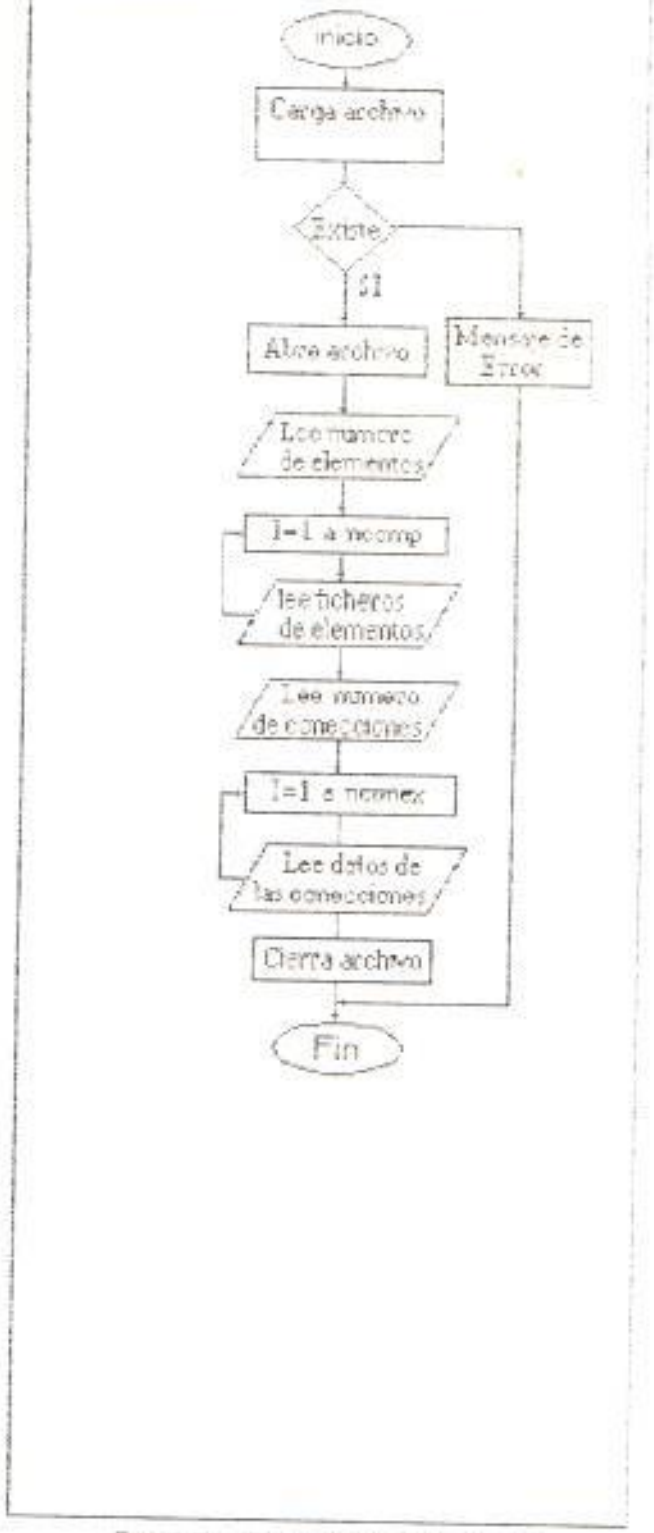


Fig. 3.3.a Flujograma de cargar()

```
mpx,fcmpy,ftro,ftrd,forcmp,forpin,fdecmp,fdcpin;     /* declara variables enteras */
                                                     /* Puntero para nombre de archivo */
=fopen(filec,"r"))==NULL){                           /* Revisa y abre si existe archivo bajo filec */
();                                                  /* si no existe lo alerta */
                                                     /* y sale del programa */

_pequena,WF_NAME,&filec,0,0);                        /* asigna comentario a ventana principal */
();
_c,"%d\n",&ncomp);                                   /* lee desde disco la variable ncomp */
comp;i++){                                           /* lee ncomp veces 6 datos */
_c,"%s %s %s\n",ficha[i].cmp,ficha[i].rem,ficha[i].valor);
_c,"%d %d %d\n",&fmodo,&fcmpx,&fcmpy);
modo=fmodo;                                          /* reasigna los datos en fichas */
mpx=fcmpx;
mpy=fcmpy;

"%d\n",&nconex);                                     /* lee desde disco dato nconex */
nconex;i++){                                         /* lee nconex veces 2 datos */
_c,"%d %d\n",&ftro,&ftrd);
ftro;                                                /* reasigna datos en arreglos */
ftrd;
abs(trd[i]%50-tro[i]%50)+abs(trd[i]/50-tro[i]/50)); /* calcula dist. entre terminales */
```

```
i<nconex;i++){                                  /* lee nconex veces 4 datos */
f(grb_c,"%d %d %d %d\n",&forcap,&forpin,&fdecap,&fdepin);
[i]=forcap;
[i]=forpin;
[i]=fdecap;
[i]=fdepin;


_c);                                            /* Cierra archivo grb_c */
%%%%.%no_sec);
                                                /* Abre en modo edición */
```

dibele()

Funciones Requeridas.- v_pline

Detalle.- Esta rutina posee una doble finalidad, una es la de dibujar en la pantalla un linea en forma de L y la otra la de asignar los valores correspondientes para la futura impresión.

Al inicio esta rutina recibe tres datos que representan a las coordenadas de los terminales de origen, vértice y destino con las que realiza un grafico por pantalla; luego estas coordenadas son separadas en x y para realizar una inspeccion de la disposición de los terminales; y segun su ubicacion otorgará varios valores numericos a un arreglo llamado estado[] que es la que posee la información equivalente a la pantalla para poder imprimir un diagrama acorde a lo que el usuario ve en el monitor.

Diagrama de Flujo

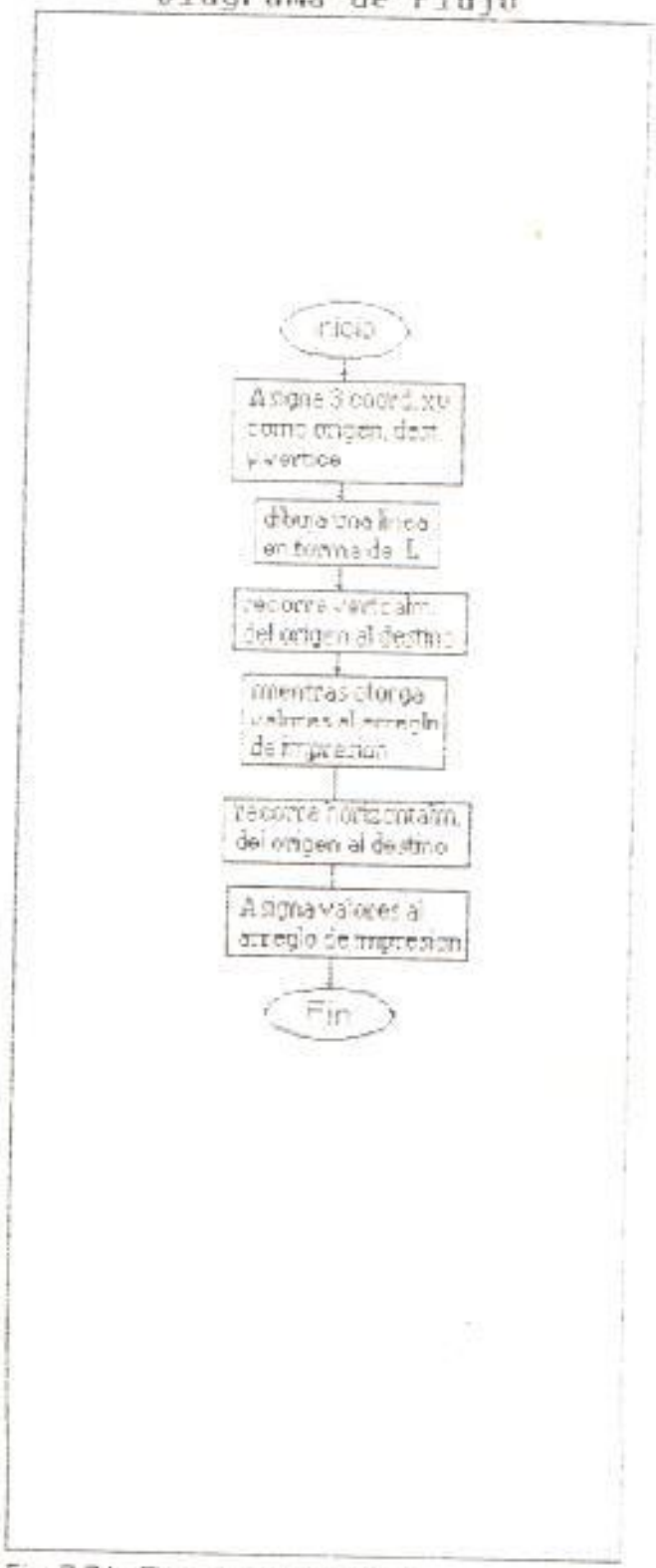


Fig. 3.3.b Flucograma de diseñ

```
.otd,de

.otd,de;                                        /* define variables de ingreso */


                                                /* declara variables arreglo */
.xt,yt,xd,yd;
                                                /* declara variables enteras */
=8+35;    yo=(or/50+1)*7+40;                    /* normaliza coordenadas de origen */
=8+35;    yt=(otd/50+1)*7+40;                   /* vértice */
=8+35;    yd=(de/50+1)*7+40;                    /* y destino */
    lin[1]=yo;                                  /* reasigna coordenadas */
    lin[3]=yt;                                  /* en un arreglo */
    lin[5]=yd;
handle,3,lin);                                  /* dibuje en pantalla lineas en forma de L */
                                                /* define los cuadros de */
<otd-50){                                       /* la rejilla por donde */
 =or+50;                                        /* pasa la linea dibujada */
 tado[or]=82;                                   /* y le indica como */
                                                /* será dibujada por */
=xd){                                           /* la impresora */
 tado[otd]=80;
 le(otd<de-1)
  tado[++otd]=76;


 tado[otd]=74;                                  /* donde 66 equivale a */
 estado[de]==66)   estado[de]=70;               /* un terminal sin */
 estado[de]==68)   estado[de]=72;               /* conectar */
```

```
while(de<otd-1)
   estado[++de]=76;

                                        /* 68 es un terminal con */
                                        /* una raya vertical encima */
                                        /* 70 corresponde a un terminal */
                                        /* con una raya horizontal */
                                        /* a la derecha */
estado[de]==66)  estado[de]=68;
estado[de]==70)  estado[de]=72;         /* 72 equivale a un terminal */
otd+50<de){                             /* con rayas vertical y */
de-50;                                  /* horizontal a la vez */
estado[de]=82;

                                        /* 76 es una recta horizontal */
estado[otd]=78;  estado[or]=70;         /* 82 es una recta vertical */
or<otd-1)
   estado[++or]=76;

otd][
estado[otd]=84;
otd<or-1)
   estado[++otd]=76;
```

dibrect()

Funciones Requeridas.- v_pline

Detalle.- Al igual que la función dibele(), esta rutina tendrá una doble finalidad.

Lo particular de esta rutina es que solo dibujará líneas rectas; para ello recibe como datos de entrada las coordenadas normalizadas de los terminales origen y destino y a la vez actualizará el arreglo estado[] con la información necesaria para informar que existe una recta en el diagrama y que deberá ser impresa después.

Diagrama de Flujo

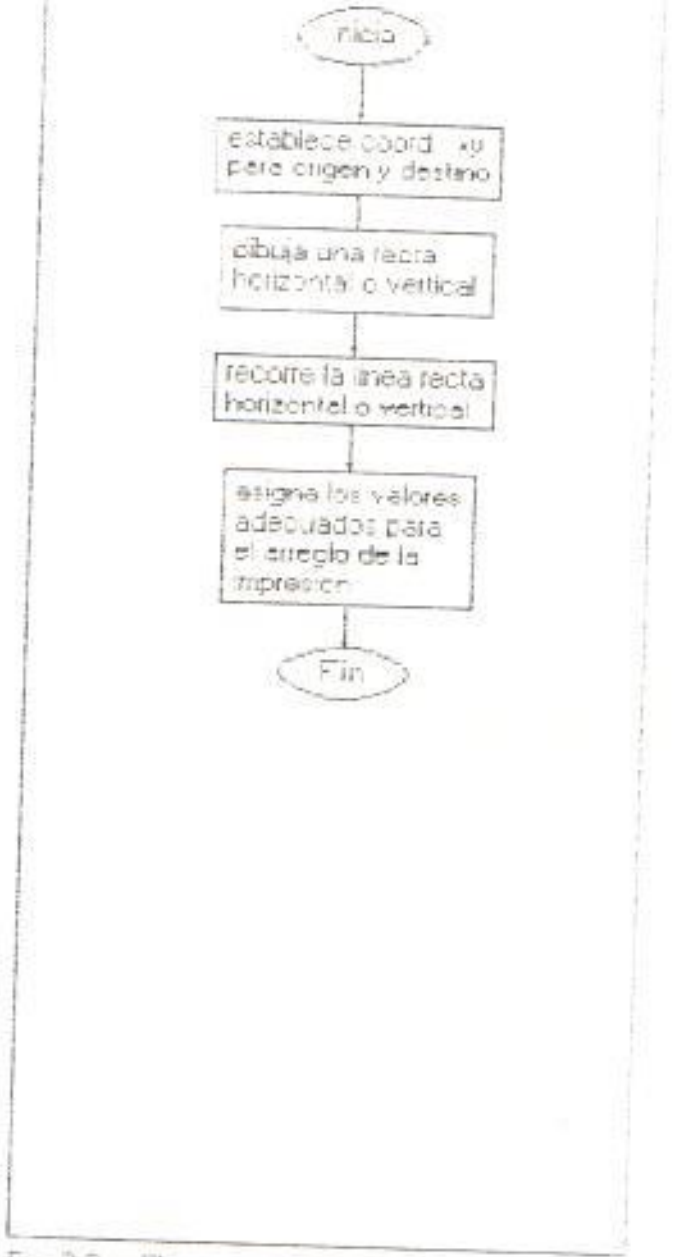


Fig. 3.3.c Flucograma de dibuj

```
or,de)

e;                                          /* variables de ingreso */


];                                          /* arreglo de enteros */
,of,yd;                                     /* declara variables enteras */
=8+35;     yo=(or/50+1)*7+40;               /* normaliza las coordenadas */
=8+35;     yd=(de/50+1)*7+40;

=0;    lin[1]=yo;

=0;    lin[3]=yd;
handle,2,lin);                              /* dibuja una recta */

{
tado[or]==66)  estado[or]=70;               /* asigna tipo gráfico */
tado[or]==68)  estado[or]=72;               /* que será dibujado */
or<de-1)                                    /* por impresora en */
tado[++or]=76;                              /* los cuadros de la */
                                            /* rejilla por donde */
                                            /* pasa la línea dibujada */
{
[de]==66)  estado[de]=68;
[de]==70)  estado[de]=72;
de-50){

=50;

or]=82;

        /* fin de while */
        /* fin de if */
```

le_a()

cciones Requeridas. abs
                     return

talle.- esta rutina nos ofrece como resultado las
rdenadas del vértice de una linea en forma de L.

nero recibe las coordenadas de los terminales origen
estino y con estos datos verifica que en el trayecto
existan ya lugares ocupados.

caso encuentra un puesto ocupado retornará entonces
valor 0 que indica que no podrá dibujar la linea L;
caso contrario devolverá un valor correspondiente a
coordenada del vértice.

Diagrama de Flujo

Fig. 3.2.c Flujograma de ele_al()

```
de)
{                                       /* ingreso de datos de origen y destino */

otd,x1,x2,y1,y2;                        /* declara las variables enteras */
th=de;                                  /* reasigna las variables */
de/50-or/50)*50+tv;                     /* calcula un vértice de L */
    y2=otd;                             /* reasigna las variables */
    x2=th-1;                            /* ajusta parametros horizontal de la L */
th){                                    /* declara que x1 esta en la izquierda */

                                        /* y x2 a la derecha */

re_v(y1,y2)==0 || recorre_h(x1,x2)==0)  /* revisa e informa si el recorrido de L */
n 0;                                    /* esta o no libre */
otd;
```

le_b()

nciones Requeridas.- abs

return

etalle.- Esta es la rutina que se usa como alternativa
n caso de que no se pueda dibujar una L.

a opcion que ofrece esta rutina es la de dibujar una L
nvertida proporcionando la coordenada de su vértice.

l igual que ele_a() es necesario antes revisar el
ecorrido del camino a seguir e informar si la
rayectoria está libre o no.

Diagrama de Flujo

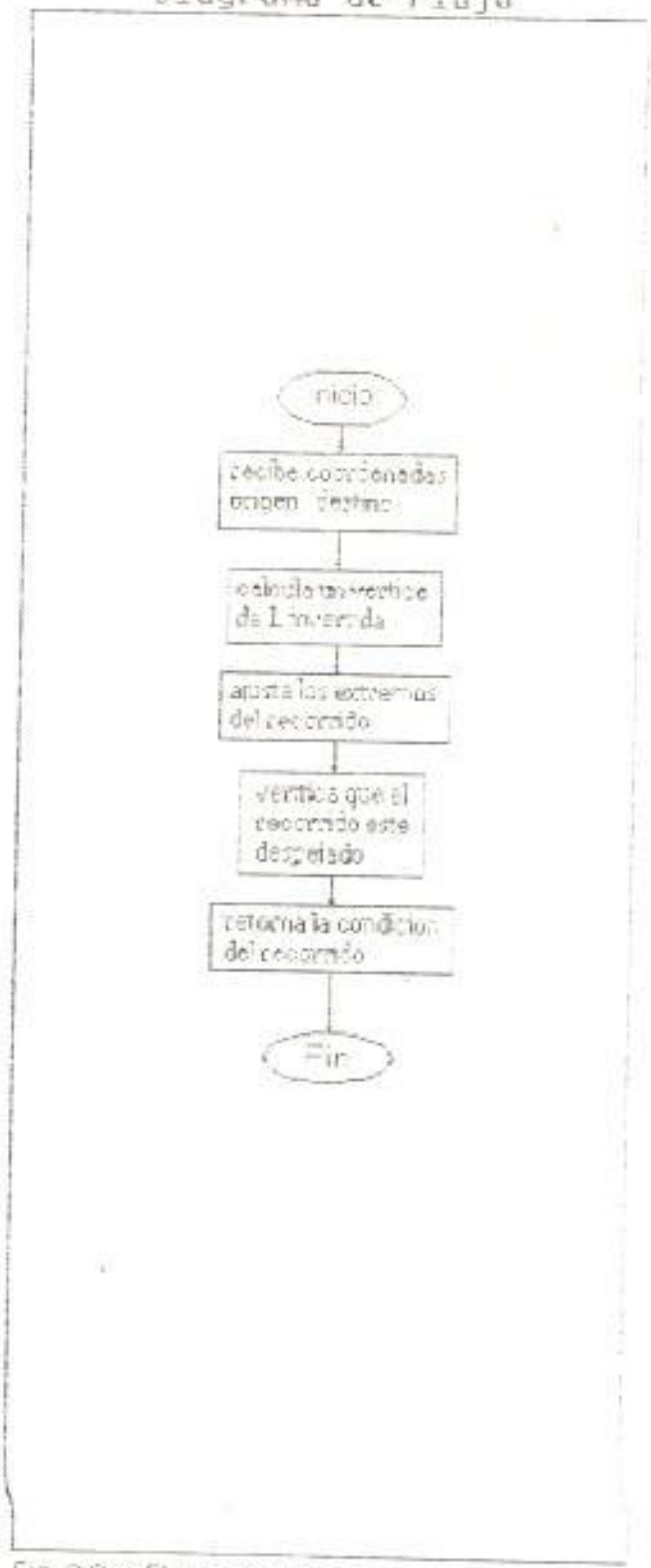


Fig. 3.3.e Flujograma de ele_obj

```
r,de)
de;                                          /* or:origen  de:destino */

th,otd,x1,x2,y1,y2;
: th=or;
v-abs(de/50-or/50)*50;                       /* calcula un segundo vértice de L */
d-50:    y2=tv-50;                           /* ajusta parámetros para que */
d-1:     x2=th-1;                            /* x1 esté a la izquierda, x2 a la derecha */
d<th){                                       /* y1 esté arriba y y2 abajo */
=th;
=otd;

rre_v(y1,y2)==0 || recorre_h(x1,x2)==0)      /* revisa e informa si el */
rn 0;                                        /* recorrido de L */
rn 1;                                        /* está o no libre */
```

enlace()

Funciones Requeridas.- v_gtext

evnt_button

v_circle

graf_dragbox

abs

Detalle.- Para realizar una conección primero habra que ingresar los datos respectivos; este es el objetivo de la rutina enlace().

En primer lugar espera el evento del ratón, de manera que si se presiona el botón la función evnt_button retornará las coordenadas de la pantalla correspondiente al lugar donde ocurrió el evento.

Luego con esos datos se verifica que el apuntador del ratón este dentro de la rejilla de trabajo, pues en caso contrario esperará nuevamente el evento inicial.

Además debe confirmar si se eligió o no el cuadro de salida (fin) para terminar la rutina; también confirmará que lo que el usuario eligió es un terminal. Si pasa todo correctamente se dibujará un circulo sobre el terminal e identificara el pin y el componente correspondiente y almacenará estos datos como origen.

Igual procedimiento será ingresar el dato destino. Y finalmente incrementará el número de conecciones.

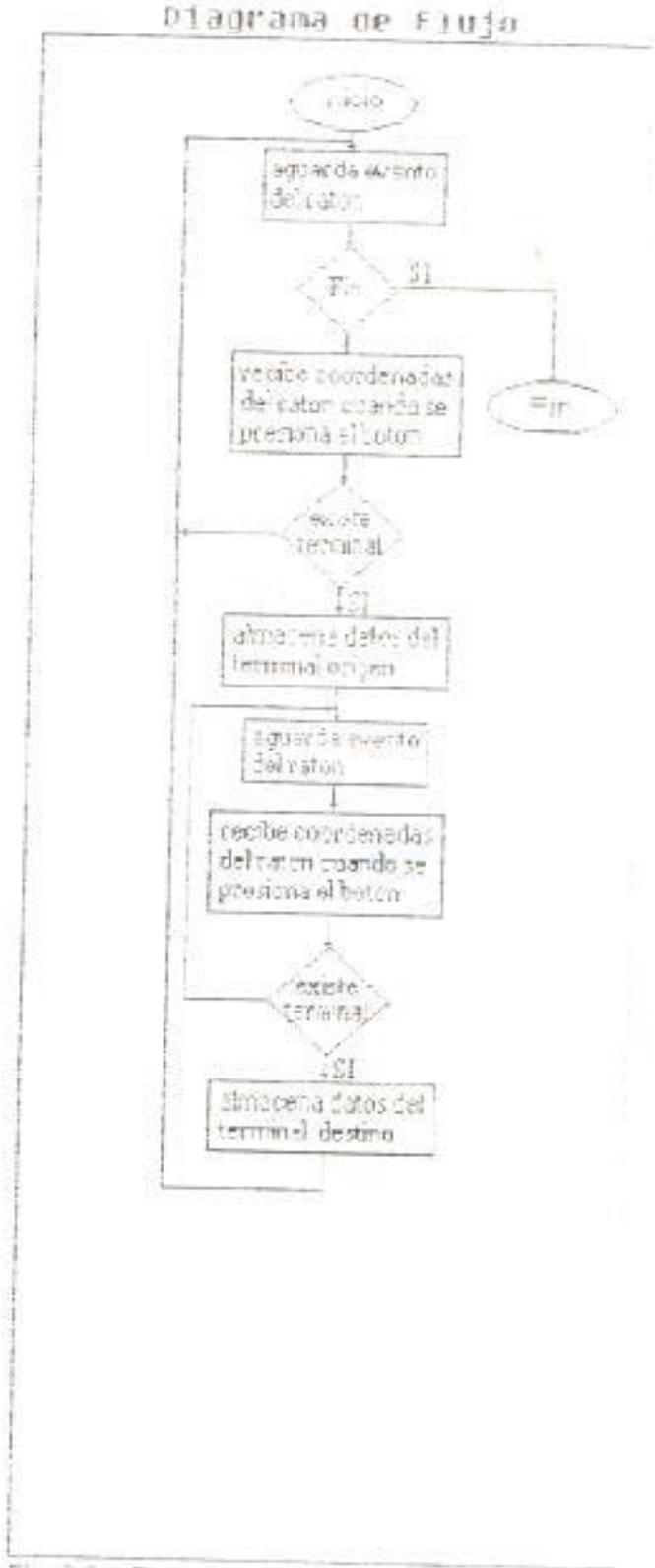


Fig. 3.3.1 Flujograma de eventos

```
                                        /* ja= numero de conexion elejida */

,tempy,refx,refy,tip;

dn,idn;                                 /* declara variables enteras de 8 bits */

_esquema,WF_INFO,"Para terminar de conectar seleccione FIN con el raton",0,0);

ndle,3,44,"fin");                       /* dibuja texto en pantalla */

 ja/20+ja%20==ja/20){                   /* selecciona grupos de 20 conexiones */

=ja%20;

);

);

==VERDAD){

                                        /* realiza contacto con el terminal origen */

tton(1,1,1,&tempx,&tempy,&no_rec,&no_rec);

<30 && tempy<hdesk)                     /* si apunta a "fin" sale de rutina */

);

gBox(8,7,tempx,tempy,39,44,400,350,&ratonx,&ratony);

ratonx-35)/8+1;     tempy=(ratony-40)/7+1;      /* asigna coordenadas */

tempx+35;           refy=7*tempy+40;            /* ejijidas y las */

x=(tempy-1)*50;                                 /* normaliza */

do[tip]<66 || estado[tip]>72) goto primera;     /* revisa si selecciono un terminal */

=tip;                                           /* incluye coordenada en la lista de origenes */

);

(handle,refx,refy,3);
```

```
hole[tip];                                  /*  identifica el elemento del terminal */
a]=ido;
(handle,482,66+8*t,ficha[ido].rem);
hole[tip];                                  /* identifica el numero del pin elejido */
el=idn;
(handle,510,66+8*t,lib[idn].num);
n]);
                                            /* realiza el contacto con el terminal destino */
tion(1,1,1,&tempx,&tempy,&no_nec,&no_nec);
ghox(8,7,tempx,tempy,39,44,400,350,&ratonx,&ratony);
ratonx-35)/8+1;   tempy=(ratony-40)/7+1;    /* asigna las coordenadas */
empx+35;          refy=7*tempy+40;          /* elejidas y las */
ux=(tempy-1)*50;                            /* normaliza */
do[tip]<66 || estado[tip]>72) goto segunda;  /* revisa si elijio un terminal */
=tip;                                       /* incluye dicha coordenada en la lista de destinos */
n]);
e(handle,refx,refy,3);
hole[tip];                                  /* identifica el elemento del terminal */
e]=ido;
(handle,555,66+8*t,ficha[ido].rem);
hole[tip];                                  /* identifica el numero del pin */
e]=idn;
(handle,585,66+8*t,lib[idn].num);
n]);
(abs(trd[ja]%50-tro[ja]%50)+abs(trd[ja]/50-tro[ja]/50);  /*calcula la dist. entre terminales */
ja]=trd[ja]){                               /* realiza un ajuste entre los terminales */
rd[ja]; trd[ja]=tro[ja]; tro[ja]=tip;       /* origen y destino */
ocmp[ja]; decmp[ja]=orcmp[ja]; orcmp[ja]=tip;
```

```
=depin[ja];                          /* realiza un intercambio */
pin[ja]=orpin[ja];                   /* de datos */
pin[ja]=tip;


                                     /* incrementa el número de conecciones */
                                     /* incrementa el número de la linea */

                                     /* asigna cantidad a variable global */
```

figura()

Función Requerida .- v_pline

Detalle.- Los gráficos que observamos en la pantalla y que representan la forma de los componentes, es realizada por esta rutina: la forma como funciona se describe a continuación.

Como se aprecia al pricipio, la rutina requiere de dos datos de entrada (cx , cy): estos son los valores numericos de las coordenadas x e y sobre la pantalla, e indican el lugar donde se dibujará un componente. Un tercer dato de entrada es la variable (mod), que corresponde al modelo de componente registrado en la "biblioteca de elementos": segun el valor de (mod) la rutina escogerá el (case) del mismo valor y ejecutará las instrucciones correspondientes hasta encontrar la sentencia (break), terminando la rutina.

Por ejemplo si (mod) es igual a 7, se asignarán valores al arreglo (lin[]) contenidos en (case 7:), estos valores corresponden a las 9 coordenadas de los vertices necesarios para formar un integrado de ocho pines. Es decir, que cada (case) contiene los datos de cada figura de componentes. (ver comentarios de la rutina).

Diagrama de Flujo

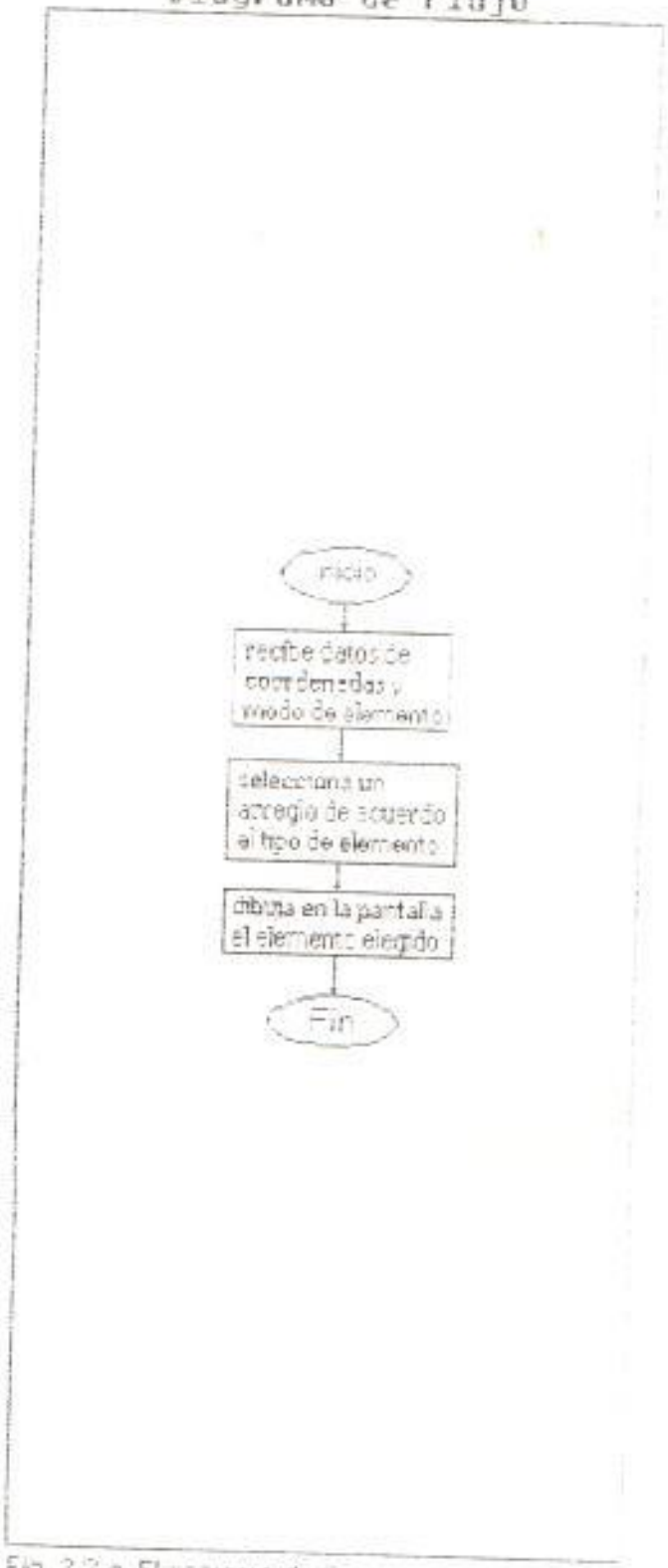


Fig. 3.3.g Flujograma de figura.

```
                                        /* se asignan las variables antes de { */

refx,refy;                              /* variables internas */
refy=7*cy+40;                           /* coordenadas de pantalla normalizada */
                                        /* Elije un case de acuerdo al valor mod */
                                        /* Vértices del esquemático de un terminal */
=refx-ax/2; lin[1]=refy-ay/2; lin[2]=refx+ax/2; lin[3]=refy-ay/2;
=refx+ax/2; lin[5]=refy+ay/2; lin[6]=refx-ax/2; lin[7]=refy+ay/2;
=refx-ax/2; lin[9]=refy-ay/2;
handle,5,lin);

                                        /* Esquema de una resistencia de 1/4W */
refx;          lin[1]=refy;       lin[2]=refx+ax/2;    lin[3]=refy;
refx+ax/2;     lin[5]=refy-ay/2;  lin[6]=refx+4.5*ax;  lin[7]=refy-ay/2;
refx+4.5*ax;   lin[9]=refy;       lin[10]=refx+5*ax;   lin[11]=refy;
refx+4.5*ax;  lin[13]=refy;       lin[14]=refx+4.5*ax; lin[15]=refy+ay/2;
refx+ax/2;    lin[17]=refy+ay/2;  lin[18]=refx+ax/2;   lin[19]=refy;
handle,10,lin);

                                        /* Esquema de una resistencia de 1/2W */
refy;          lin[1]=refy;        lin[2]=refx+ax/2;    lin[3]=refy;
refx+ax/2;     lin[5]=refy-0.8*ay; lin[6]=refx+5.5*ax;  lin[7]=refy-0.8*ay;
refx+5.5*ax;   lin[9]=refy;        lin[10]=refx+6*ax;   lin[11]=refy;
refx+5.5*ax;  lin[13]=refy;        lin[14]=refx+5.5*ax; lin[15]=refy+0.8*ay;
refx+ax/2;    lin[17]=refy+0.8*ay; lin[18]=refx+ax/2;   lin[19]=refy;
```

```
*_pline(handle,10,lin);

break;

= 4:                                        /* Esquema de una resistencia de 1W */

lin[0]=refx-ax/2;   lin[1]=refy-0.2*ay;  lin[2]=refx+6.5*ax;  lin[3]=refy-0.2*ay;

lin[4]=refx+6.5*ax; lin[5]=refy+ay;      lin[6]=refx+7*ax;    lin[7]=refy+ay;

lin[8]=refx+6.5*ax; lin[9]=refy+ay;      lin[10]=refx+6.5*ax; lin[11]=refy+2.2*ay;

lin[12]=refx-ax/2;  lin[13]=refy+2.2*ay; lin[14]=refx-ax/2;   lin[15]=refy+ay;

lin[16]=refx-ax;    lin[17]=refy+ay;     lin[18]=refx-ax/2;   lin[19]=refy+ay;

lin[20]=refx-ax/2;  lin[21]=refy-0.2*ay;

*_pline(handle,11,lin);

break;

                                            /* Esquema de una resistencia de 2W */

lin[0]=refx-ax/2;   lin[1]=refy-ay/2;    lin[2]=refx+7.5*ax;  lin[3]=refy-ay/2;

lin[4]=refx+7.5*ax; lin[5]=refy+ay;      lin[6]=refx+8*ax;    lin[7]=refy+ay;

lin[8]=refx+7.5*ax; lin[9]=refy+ay;      lin[10]=refx+7.5*ax; lin[11]=refy+2.5*ay;

lin[12]=refx-ax/2;  lin[13]=refy+2.5*ay; lin[14]=refx-ax/2;   lin[15]=refy+ay;

lin[16]=refx-ax;    lin[17]=refy+ay;     lin[18]=refx-ax/2;   lin[19]=refy+ay;

lin[20]=refx-ax/2;  lin[21]=refy-ay/2;

_pline(handle,11,lin);

break;

                                            /* Esquema de una resistencia de 5W */

lin[0]=refx-ax/2;   lin[1]=refy;       lin[2]=refx+9.5*ax;  lin[3]=refy;

lin[4]=refx+9.5*ax; lin[5]=refy+2*ay;  lin[6]=refx+10*ax;   lin[7]=refy+2*ay;

lin[8]=refx+9.5*ax; lin[9]=refy+2*ay;  lin[10]=refx+9.5*ax; lin[11]=refy+4*ay;

lin[12]=refx-ax/2;  lin[13]=refy+4*ay; lin[14]=refx-ax/2;   lin[15]=refy+2*ay;

lin[16]=refx-ax;    lin[17]=refy+2*ay; lin[18]=refx-ax/2;   lin[19]=refy+2*ay;

lin[20]=refx-ax/2;  lin[21]=refy;
```

```
_pline(handle,11,lin);
reak;
                                        /* Esquema de un Integrado de 8 pines */
n[0]=refx-ax/2;   lin[1]=refy-ay/2;   lin[2]=refx+3.5*ax; lin[3]=refy-ay/2;
n[4]=refx+3.5*ax; lin[5]=refy+3.5*ay; lin[6]=refx-ax/2;   lin[7]=refy+3.5*ay;
n[8]=refx-ax/2;   lin[9]=refy+2.5*ay; lin[10]=refx;       lin[11]=refy+2.5*ay;
n[12]=refx;       lin[13]=refy+2*ay;  lin[14]=refx-ax/2;  lin[15]=refy+2*ay;
n[16]=refx-ax/2;  lin[17]=refy-ay/2;
pline(handle,9,lin);
                                        /* Esquema de un integrado de 10 pines */
=refx-ax/2;   lin[1]=refy-ay/2;   lin[2]=refx+4.5*ax; lin[3]=refy-ay/2;
=refx+4.5*ax; lin[5]=refy+3.5*ay; lin[6]=refx-ax/2;   lin[7]=refy+3.5*ay;
=refx-ax/2;   lin[9]=refy+2.5*ay; lin[10]=refx;       lin[11]=refy+1.5*ay;
=refx;        lin[13]=refy+2*ay;  lin[14]=refx-ax/2;  lin[15]=refy+2*ay;
=refx-ax/2;   lin[17]=refy-ay/2;
(handle,9,lin);
                                        /* Esquema de un integrado 14 pines */
=refx-ax/2;   lin[1]=refy-ay/2;   lin[2]=refx+6.5*ax; lin[3]=refy-ay/2;
=refx+6.5*ax; lin[5]=refy+3.5*ay; lin[6]=refx-ax/2;   lin[7]=refy+3.5*ay;
=refx-ax/2;   lin[9]=refy+2.5*ay; lin[10]=refx;       lin[11]=refy+2.5*ay;
=refx;        lin[13]=refy+2*ay;  lin[14]=refx-ax/2;  lin[15]=refy+2*ay;
=refx-ax/2;   lin[17]=refy-ay/2;
(handle,9,lin);
```

```
                                        /* Esquema de un integrado de 16 pines */
lin[0]=refx-ax/2;   lin[1]=refy-ay/2;   lin[2]=refx+7.5*ax; lin[3]=refy-ay/2;
lin[4]=refx+7.5*ax; lin[5]=refy+3.5*ay; lin[6]=refx-ax/2;   lin[7]=refy+3.5*ay;
lin[8]=refx-ax/2;   lin[9]=refy+2.5*ay; lin[10]=refx;       lin[11]=refy+2.5*ay;
lin[12]=refx;       lin[13]=refy+2*ay;  lin[14]=refx-ax/2;  lin[15]=refy+2*ay;
lin[16]=refx-ax/2;  lin[17]=refy-ay/2;
lpline(handle,9,lin);

                                        /* Esquema de un integrado de 24 pines */
refx-ax/2;    lin[1]=refy-ay/2;   lin[2]=refx+11.5*ax; lin[3]=refy-ay/2;
refx+11.5*ax; lin[5]=refy+6.5*ay; lin[6]=refx-ax/2;    lin[7]=refy+6.5*ay;
refx-ax/2;    lin[9]=refy+5.5*ay; lin[10]=refx;        lin[11]=refy+5.5*ay;
refx;         lin[13]=refy+5*ay;  lin[14]=refx-ax/2;   lin[15]=refy+5*ay;
refx-ax/2;    lin[17]=refy-ay/2;
handle,9,lin);

                                        /* Esquema de un Diodo de */
+35; lin[1]=7*cy+40; lin[2]=8*cx+39; lin[3]=7*cy+40;
+39; lin[5]=7*cy+43; lin[6]=8*cx+55; lin[7]=7*cy+43;
+55; lin[9]=7*cy+40; lin[10]=8*cx+59; lin[11]=7*cy+40;
+55; lin[13]=7*cy+40; lin[14]=8*cx+55; lin[15]=7*cy+37;
+39; lin[17]=7*cy+37; lin[18]=8*cx+39; lin[19]=7*cy+43;
+43; lin[21]=7*cy+43; lin[22]=8*cx+43; lin[23]=7*cy+37;
handle,12,lin);

                                        /* Esquema de un Diodo */
+35; lin[1]=7*cy+40; lin[2]=8*cx+39; lin[3]=7*cy+40;
+39; lin[5]=7*cy+44; lin[6]=8*cx+63; lin[7]=7*cy+44;
```

```
...8]=8*cx+63;  lin[9]=7*cy+40;  lin[10]=8*cx+67; lin[11]=7*cy+40;
...22]=8*cx+63; lin[13]=7*cy+40; lin[14]=8*cx+63; lin[15]=7*cy+36;
...16]=8*cx+39; lin[17]=7*cy+36; lin[18]=8*cx+39; lin[19]=7*cy+44;
...20]=8*cx+43; lin[21]=7*cy+44; lin[22]=8*cx+43; lin[23]=7*cy+36;
...ine(handle,12,lin);
...end;
...                                        /* Esquema de un Diodo */
...0]=8*cx+35;  lin[1]=7*cy+40;  lin[2]=8*cx+39;  lin[3]=7*cy+40;
...4]=8*cx+39;  lin[5]=7*cy+45;  lin[6]=8*cx+71;  lin[7]=7*cy+45;
...8]=8*cx+71;  lin[9]=7*cy+40;  lin[10]=8*cx+75; lin[11]=7*cy+40;
...12]=8*cx+71; lin[13]=7*cy+40; lin[14]=8*cx+71; lin[15]=7*cy+35;
...16]=8*cx+39; lin[17]=7*cy+35; lin[18]=8*cx+39; lin[19]=7*cy+45;
...20]=8*cx+43; lin[21]=7*cy+45; lin[22]=8*cx+43; lin[23]=7*cy+35;
...handle,12,lin);
...end;
...                                        /* Esquema de un Diodo */
...0]=8*cx+35;  lin[1]=7*cy+40;  lin[2]=8*cx+39;  lin[3]=7*cy+40;
...4]=8*cx+39;  lin[5]=7*cy+47;  lin[6]=8*cx+79;  lin[7]=7*cy+47;
...8]=8*cx+79;  lin[9]=7*cy+40;  lin[10]=8*cx+83; lin[11]=7*cy+40;
...12]=8*cx+79; lin[13]=7*cy+40; lin[14]=8*cx+79; lin[15]=7*cy+33;
...16]=8*cx+39; lin[17]=7*cy+33; lin[18]=8*cx+39; lin[19]=7*cy+47;
...20]=8*cx+43; lin[21]=7*cy+47; lin[22]=8*cx+43; lin[23]=7*cy+33;
...e,12,lin);
...                                        /* Esquema de un Translator */
...      lin[1]=refy-0.75*ay; lin[2]=refx;      lin[3]=refy-ay;
...x+4*ax; lin[5]=refy-ay;      lin[6]=refx+4*ax; lin[7]=refy-ay;
```

```
lin[8]=refx;      lin[9]=refy+my; lin[10]=refx; lin[11]=refy-0.75*my;
lin[12]=refx+4*mx; lin[13]=refy-0.75*my;
_pline(handle,7,lin);
break;

                                        /* Esquema de un transistor */
lin[0]=refx+mx;  lin[1]=refy;       lin[2]=refx+2*mx; lin[3]=refy+my/2;
lin[4]=refx+2*mx; lin[5]=refy+1.5*my; lin[6]=refx;     lin[7]=refy+1.5*my;
lin[8]=refx;     lin[9]=refy+0.5*my; lin[10]=refx+mx;  lin[11]=refy;
_pline(handle,6,lin);
break;

                                        /* Esquema de un transistor */
lin[0]=refx+0.5*mx;  lin[1]=refy-my/2;   lin[2]=refx+1.5*mx;  lin[3]=refy-my/2;
lin[4]=refx+2.5*mx;  lin[5]=refy+0.5*my;  lin[6]=refx+2.5*mx;  lin[7]=refy+1.5*my;
lin[8]=refx+2*mx;    lin[9]=refy+2*my;    lin[10]=refx+2.5*mx; lin[11]=refy+2.5*my;
lin[12]=refx+2*mx;   lin[13]=refy+2*my;   lin[14]=refx+1.5*mx; lin[15]=refy+2.5*my;
lin[16]=refx+0.5*mx; lin[17]=refy+2.5*my; lin[18]=refx-0.5*mx; lin[19]=refy+1.5*my;
lin[20]=refx+0.5*mx; lin[21]=refy+0.5*my; lin[22]=refx+0.5*mx; lin[23]=refy-0.5*my;
_pline(handle,12,lin);
break;

                                        /* Esquema de un Transistor */
lin[0]=refx+6*mx;  lin[1]=refy;        lin[2]=refx+9*mx;  lin[3]=refy;
lin[4]=refx+15*mx; lin[5]=refy+4.5*my;  lin[6]=refx+15*mx; lin[7]=refy+5.5*my;
lin[8]=refx+9*mx;  lin[9]=refy+10*my;   lin[10]=refx+6*mx; lin[11]=refy+10*my;
lin[12]=refx;      lin[13]=refy+5.5*my; lin[14]=refx;      lin[15]=refy+4.5*my;
lin[16]=refx+6*mx; lin[17]=refy;
_pline(handle,9,lin);
break;
```

```
                                        /* Esquema de un Capacitor */
lin[0]=refx+ax;     lin[1]=refy;        lin[2]=refx+7*ax;  lin[3]=refy;
lin[4]=refx+7.5*ax; lin[5]=refy+0.5*ay; lin[6]=refx+8*ax;  lin[7]=refy;
lin[8]=refx+9*ax;   lin[9]=refy;        lin[10]=refx+9*ax; lin[11]=refy+2*ay;
lin[12]=refx+10*ax; lin[13]=refy+2*ay;  lin[14]=refx+9*ax; lin[15]=refy+2*ay;
lin[16]=refx+9*ax;  lin[17]=refy+4*ay;  lin[18]=refx+8*ax; lin[19]=refy+4*ay;
lin[20]=refx+7.5*ax; lin[21]=refy+3.5*ay; lin[22]=refx+7*ax; lin[23]=refy+4*ay;
lin[24]=refx+ax;    lin[25]=refy+4*ay;  lin[26]=refx+ax;   lin[27]=refy+2*ay;
lin[28]=refx;       lin[29]=refy+2*ay;  lin[30]=refx+ax;   lin[31]=refy+2*ay;
lin[32]=refx+ax;    lin[33]=refy+ay;
                                        /* Esquema de un Capacitor */
lin[0]=refx+ax;     lin[1]=refy;        lin[2]=refx+9*ax;  lin[3]=refy;
lin[4]=refx+9.5*ax; lin[5]=refy+0.5*ay; lin[6]=refx+10*ax; lin[7]=refy;
lin[8]=refx+11*ax;  lin[9]=refy;        lin[10]=refx+11*ax; lin[11]=refy+2*ay;
lin[12]=refx+12*ax; lin[13]=refy+2*ay;  lin[14]=refx+11*ax; lin[15]=refy+2*ay;
lin[16]=refx+11*ax; lin[17]=refy+4*ay;  lin[18]=refx+10*ax; lin[19]=refy+4*ay;
lin[20]=refx+9.5*ax; lin[21]=refy+3.5*ay; lin[22]=refx+9*ax; lin[23]=refy+4*ay;
lin[24]=refx+ax;    lin[25]=refy+4*ay;  lin[26]=refx+ax;   lin[27]=refy+2*ay;
lin[28]=refx;       lin[29]=refy+2*ay;  lin[30]=refx+ax;   lin[31]=refy+2*ay;
lin[32]=refx+ax;    lin[33]=refy;
                                        /* Esquema de un Capacitor */
lin[0]=refx+ax;     lin[1]=refy;        lin[2]=refx+14*ax; lin[3]=refy;
lin[4]=refx+14.5*ax; lin[5]=refy+0.5*ay; lin[6]=refx+15*ax; lin[7]=refy;
lin[8]=refx+16*ax;  lin[9]=refy;        lin[10]=refx+16*ax; lin[11]=refy+2*ay;
```

```
lin[12]=refx+17*mx;   lin[13]=refy+2*my;   lin[14]=refx+16*mx; lin[15]=refy+2*my;
lin[16]=refx+16*mx;   lin[17]=refy+4*my;   lin[18]=refx+15*mx; lin[19]=refy+4*my;
lin[20]=refx+14.5*mx; lin[21]=refy+3.5*my; lin[22]=refx+14*mx; lin[23]=refy+4*my;
lin[24]=refx+mx;      lin[25]=refy+4*my;   lin[26]=refx+mx;    lin[27]=refy+2*my;
lin[28]=refx;         lin[29]=refy+2*my;   lin[30]=refx+mx;    lin[31]=refy+2*my;
lin[32]=refx+mx;      lin[33]=refy;
_pline(handle,17,lin);
break;

                                        /* Esquema de un Capacitor */
lin[0]=refx+mx;    lin[1]=refy;       lin[2]=refx+20*mx;  lin[3]=refy;
lin[4]=refx+20*mx; lin[5]=refy+4*my;  lin[6]=refx+21*mx;  lin[7]=refy+4*my;
lin[8]=refx+20*mx; lin[9]=refy+4*my;  lin[10]=refx+20*mx; lin[11]=refy+8*my;
lin[12]=refx+mx;   lin[13]=refy+8*my; lin[14]=refx+mx;    lin[15]=refy+4*my;
lin[16]=refx;      lin[17]=refy+4*my; lin[18]=refx+mx;    lin[19]=refy+4*my;
lin[20]=refx+mx;   lin[21]=refy;
_pline(handle,11,lin);
break;

                                        /* Esquema de un Capacitor */
lin[0]=refx+3.5*mx;  lin[1]=refy+1.5*my;  lin[2]=refx+3.5*mx;  lin[3]=refy+0.5*my;
lin[4]=refx+2.5*mx;  lin[5]=refy-0.5*my;  lin[6]=refx+mx;      lin[7]=refy-0.5*my;
lin[8]=refx;         lin[9]=refy+0.5*my;  lin[10]=refx;        lin[11]=refy+2*my;
lin[12]=refx+mx;     lin[13]=refy+3*my;   lin[14]=refx+2.5*mx; lin[15]=refy+3*my;
lin[16]=refx+3.5*mx; lin[17]=refy+2*my;   lin[18]=refx+3.5*mx; lin[19]=refy+1.5*my;
lin[20]=refx+2.5*mx; lin[21]=refy+1.5*my; lin[22]=refx+3*mx;   lin[23]=refy+1.5*my;
lin[24]=refx+3*mx;   lin[25]=refy+my;     lin[26]=refx+3*mx;   lin[27]=refy+2*my;
_pline(handle,14,lin);
break;
```

```
                                        /* Esquema de un Capacitor */
[0]=refx+6.5*ax;  lin[1]=refy+3*ay;   lin[2]=refx+6.5*ax;  lin[3]=refy+1.5*ay;
[4]=refx+4.5*ax;  lin[5]=refy-0.5*ay; lin[6]=refx+2*ax;   lin[7]=refy-0.5*ay;
[8]=refx;         lin[9]=refy+1.5*ay; lin[10]=refx;       lin[11]=refy+4*ay;
[12]=refx+2*ax;   lin[13]=refy+6*ay;  lin[14]=refx+4.5*ax; lin[15]=refy+6*ay;
[16]=refx+6.5*ax; lin[17]=refy+4*ay;  lin[18]=refx+6.5*ax; lin[19]=refy+2*ay;
[20]=refx+5.5*ax; lin[21]=refy+3*ay;  lin[22]=refx+6*ax;   lin[23]=refy+3*ay;
[24]=refx+6*ax;   lin[25]=refy+2.5*ay; lin[26]=refx+6*ax;  lin[27]=refy+0.5*ay;
ne(handle,14,lin);

                                        /* Esquema de un Capacitor */
=refx+8*ax;   lin[1]=refy+4*ay;   lin[2]=refx+8*ax;   lin[3]=refy+2*ay;
=refx+6*ax;   lin[5]=refy;        lin[6]=refx+2*ax;   lin[7]=refy;
=refx;        lin[9]=refy+2*ay;   lin[10]=refx;       lin[11]=refy+6*ay;
=refx+2*ax;   lin[13]=refy+8*ay;  lin[14]=refx+6*ax;  lin[15]=refy+8*ay;
=refx+8*ax;   lin[17]=refy+6*ay;  lin[18]=refx+8*ax;  lin[19]=refy+4*ay;
=refx+7*ax;   lin[21]=refy+4*ay;  lin[22]=refx+7.5*ax; lin[23]=refy+4*ay;
=refx+7.5*ax; lin[25]=refy+3.5*ay; lin[26]=refx+7.5*ax; lin[27]=refy+4.5*ay;
handle,14,lin);

                                        /* Esquema de un Capacitor */
-0.5*ax; lin[1]=refy-0.5*ay; lin[2]=refx+3.5*ax; lin[3]=refy-0.5*ay;
+3.5*ax; lin[5]=refy+0.5*ay; lin[6]=refx-0.5*ax; lin[7]=refy+0.5*ay;
-0.5*ax; lin[9]=refy-0.5*ay;
handle,5,lin);
```

```
e 28:                                        /* Esquema de un Capacitor */

lin[0]=refx-0.5*ax; lin[1]=refy;     lin[2]=refx+5.5*ax; lin[3]=refy;
lin[4]=refx+5.5*ax; lin[5]=refy+2*ay; lin[6]=refx-0.5*ax; lin[7]=refy+2*ay;
lin[8]=refx-0.5*ax; lin[9]=refy;
_pline(handle,5,lin);
break;
29:
lin[0]=refx+ax;   lin[1]=refy;      lin[2]=refx+6*ax; lin[3]=refy;
lin[4]=refx+7*ax; lin[5]=refy+ay;   lin[6]=refx+7*ax; lin[7]=refy+3*ay;
lin[8]=refx+6*ax; lin[9]=refy+4*ay; lin[10]=refx+ax;  lin[11]=refy+4*ay;
lin[12]=refx;     lin[13]=refy+3*ay; lin[14]=refx;    lin[15]=refy+ay;
lin[16]=refx+ax;  lin[17]=refy;
line(handle,9,lin);
break;
                                     /* Esquema de un Capacitor */
lin[0]=refx+ax;   lin[1]=refy;      lin[2]=refx+9*ax;  lin[3]=refy;
lin[4]=refx+10*ax; lin[5]=refy+ay;  lin[6]=refx+10*ax; lin[7]=refy+5*ay;
lin[8]=refx+9*ax;  lin[9]=refy+6*ay; lin[10]=refx+ax;  lin[11]=refy+6*ay;
lin[12]=refx;     lin[13]=refy+5*ay; lin[14]=refx;    lin[15]=refy+ay;
lin[16]=refx+ax;  lin[17]=refy;
line(handle,9,lin);
```

grabar()

Funciones Requeridas.- rsrc_gaddr

form_center

form_dial

objc_draw

form_do

f_printf

fopen

fclose

Detalle.- Esta rutina se encarga del manejo de una etiqueta de diálogo con el que interactúa el usuario y utiliza para ingresar el nombre con que va grabar su diagrama.

Ingresado el nombre se abrirá el archivo de escritura en el disco de almacenamiento y revisa si es posible hacerlo, pues en caso contrario enviará una alerta y saldrá del programa.

Los datos a ser guardados son: numero de componentes, tipo de componente y su ubicación en coordenadas; número de conecciones y las coordenadas origen-destino con el componente y pin al que pertenecen.

El último paso de la rutina es cerrar el fichero.

Diagrama de Flujo

Fig 3.3 Flujograma de grabar

```
                        /* Asigna variable como un arbol OBJECT */
                        /* puntero de un archivo */
                        /* Define variables enteras */
fcopy,ftro,ftrd,forcmp,forpin,fdecmp,fdepin;
ARBOL3, &tarjeta);                          /* redirecciona ARBOL3 en tarjeta */
tarjeta[NOMBRE].ob_spec->te_ptext = filec;  /* asigna a NOMBRE variable filec */
tarjeta[NOMBRE].ob_spec->te_txtlen = 10;    /* Define máxima longitud de variable */
tarjeta,&cx,&cy,&cw,&ch);                   /* Centra dibujo */
_START,cx,cy,cw,ch,0,0,0,0);
jeta,0,10,cx,cy,cw,ch);                     /* Visualiza Dibujo */
eta,0);                                     /* Establece comunicación con la caja */
NCEL].ob_state == SELECTED) {               /* Hacere estado de la caja CANCEL */
NCEL].ob_state &= ~SELECTED;

].ob_state &= ~SELECTED;                    /* Encera estado de la caja LISTO */
_FINISH,cx,cy,cw,ch,0,0,0,0);               /* Cierra caja de dialogo */
filec,"w")==NULL){                          /* Abre un en archivo */
                                            /* Detecta un error en archivo */


"%d\n",ncomp);                              /* Grata ncomp en disco */
```

```
...i<ncomp;i++){                                    /* escribe ncomp veces 6 datos */
...intf(grb_g,"%e %e %e\n",ficha[i].cap.ficha[i].res.ficha[i].valor);
...modo=ficha[i].modo;                              /* reasigna ficheros en variables internas */
...mpx=ficha[i].cmpx;
...mpy=ficha[i].cmpy;
...intf(grb_g,"%02d %02d %01d\n",fmodo.fcmpx,fcmpy);


...grb_g,"%d\n",nconex);                            /* escribe dato nconex en disco */
...i<nconex;i++){                                   /* escribe nconex veces 2 datos */
...tro[i];
...trd[i];
...intf(grb_g,"%04d %04d\n",ftro,ftrd);


...nconex;i++){                                     /* escribe nconex veces 4 datos */
...rcmp=rcmp[i];                                    /* reasigna ficheros en variables internas */
...rpin=rpin[i];
...decmp[i];
...depin[i];
...grb_g,"%04d %04d %04d %04d\n",forcmp,forpin,fdecmp,fdepin);


...g);                                              /* cierra archivo grb_g */
...*.kno_nec);
```

impdiag()

Funcion Requerida.- gemdos

Detalle.- El control total de la impresora por parte del programa, se basa en el envio de ordenes en codigo ASCII, las cuales pueden ser interpretadas por el periferico como una instrucción para realizar diversas funciones que van desde la impresión un caracter hasta el movimiento controlado de las agujas, el carro o el rodillo de alimentación de papel.

En nuestro programa las funciones de impresora requeridas para este tipo de gráficos son:

- Imprimir modo enfatizado.
- Definir caracter de usuario.
- Seleccionar caracter de usuario.
- Avance horizontal por tabulación.
- Avance del rodillo en n/216 pulgadas.
- Regreso controlado del carro.

El procedimiento secuencial de las ordenes a la impresora se describe en el flujograma, lo esencial en esta rutina es aclarar que la impresion del diagrama se hace en base de "caracteres gráficos" que fueron previamente definidos en el programa (ver rutina formato()); es decir, en lugar de imprimir una letra se imprime un motivo gráfico, que formarán juntos a otros el diseño final en perfecta armonia.

Diagrama de Flujo

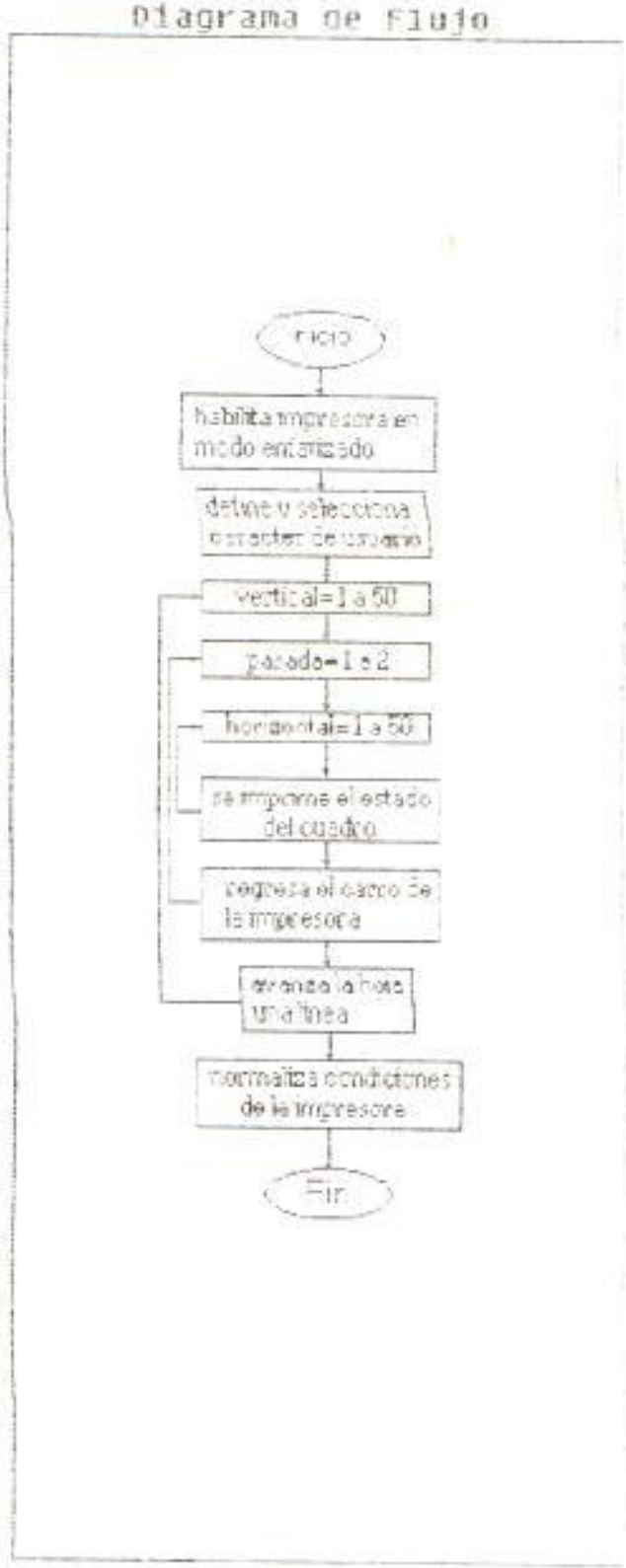


Fig. 3.3.1 Flujograma de impresión

```
                                        /* declara variables enteras */
    gendos(0x5,69);                     /* modo enfatizado */
                                        /* llama a la funcion */
gendos(0x5,58); gendos(0x5,0);          /* selecciona caracter de usuario */
    gendos(0x5,84);                     /* desde @ hasta ~ */
                                        /* contiene al */
                                        /* envia a la impresora formato */
gendos(0x5,37); gendos(0x1);            /* selecciona caracter de usuario */
                                        /* se envia 2 tabulaciones */
                                        /* lazo para 50 lineas */
                                        /* lazo para la segunda pasada */
gendos(0x5,74); gendos(0x5,2);          /* avance de 2/216 pulgadas */
                                        /* lazo para 50 columnas */
                                        /* grafico para la coord. (x,y) */
gendos(0x5,37); gendos(0x5,1);
                                        /* imprime caracter grafico */


gendos(0x5,92); gendos(0x5,168);        /* envia la orden de regresar el
                                           carro sin avance de rodillo */

gendos(0x5,74); gendos(0x5,28);         /* avance de 18/256 pulgadas */
```

informe()

Funciones Requeridas.- rsrc_gaddr

form_center

form_dial

objc_draw

form_do

form_dial

Descripción.- La finalidad de esta rutina es la de ofrecer al usuario una etiqueta de consulta con la información correspondiente a un componente en especial, dirndando los datos de tipo, nombre y valor del mismo.

Ademas, posee la caracteristica de poder editar tanto el nombre como el valor del elemento y para ello la función form_do espera del usuario su requerimiento.

Cualquier cambio que se realice quedará almacenado en las fichas respectivas.

Esta rutina terminará actualizando el diagrama en la pantalla.

Diagrama de Flujo

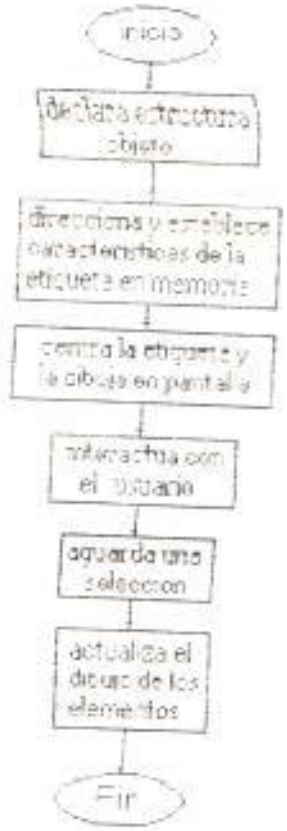


Fig. 3.3: Diagrama de interfaz

```
                                        /* ingreso de numero de ficha */

                                        /* declara a info como estructura OBJECT */
,ch;
(0, ARBOL4, &info);                     /* direcciona en memoria la estructura ARBOL4 */
ob_spec=ficha[inf].cmp;                 /* establece caracteristicas */
*)info[SEM].ob_spec)->te_ptext=ficha[inf].sem;   /* para la edición de la */
*)info[SEM].ob_spec)->te_txtlen=4;       /* caja de diálogo */
*)info[VALOR].ob_spec)->te_ptext=ficha[inf].valor;
*)info[VALOR].ob_spec)->te_txtlen=6;
info,&cx,&cy,&cw,&ch);
_START,cx,cy,cw,ch,0,0,0,0);
nfo,0,10,cx,cy,cw,ch);                  /* plasma en pantalla la etiqueta */
,0);                                    /* interactua con el usuario */
ob_state &= ~SELECTED;                  /* inicializa la caja LIST02 */
_FINISH,cx,cy,cw,ch,0,0,0,0);           /* termina diálogo */
```

ingresar()

Funciones Requeridas.- v_curtext

vs_curaddress

printf

gets

strcmp

strcpy

v_rvc44

Detalle.- Esta es la rutina que acepta el ingreso de los datos de cada elemento. Para ello entra la pantalla en el modo texto donde direccionará el cursor en las columnas respectivas al pedir dichos datos.

Los datos a ingresar son: nombre, tipo y valor y la rutina se encargará de verificar que cumplan con un formato especifico con la función strcmp.

La función gets es la encargada de pedir los datos por el teclado y la función strcpy asignará los datos a los respectivos arreglos.

Una vez terminado de ingresar los datos volveremos a la pantalla en el modo gráfico.

Diagrama de Flujo

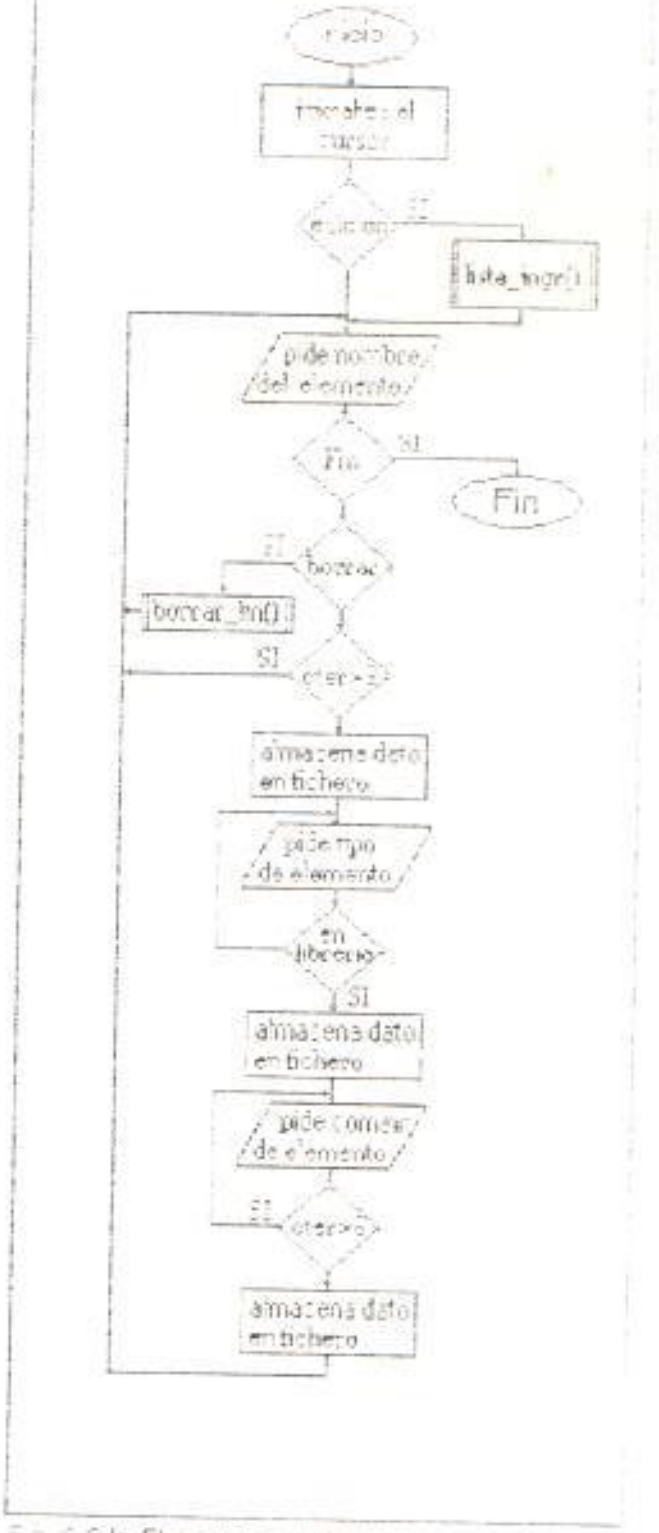


Fig. 3.2.1 Diagrama de ingeniería

```
                                                    /* declara variables enteras */
                                                    /* declara variables texto */
                                                    /* final inicial */

handle,"CMF REF. VALOR   [ Presione <Return> para cada dato ingresado ] ");
                                                    /* direcciona cursor */
handle,"Para Borrar digite <*>    Para Finalizar digite <fin> ");
                                                    /* inicializa datos de */
                                                    /* fila, columna y número */
                                                    /* de fichas */

                                                    /* solo si esta en modo edición */

                                                    /* case para datos de nombre de elemento */
                                                    /* asigna número de ficha */
                                                    /* segun el número de ficha */
                                                    /* escoje un lugar en la */
                                                    /* pantalla y lo escribe */

                                                    /* si hay mas de 39 elementos */
```

```
if(espacio>59){                                  /* si hay mas de 59 elementos */
   movfila=60;
   col=52;
}
if(espacio==-1){                                 /* si no hay lugar para */
   printf("\n lista llena");                     /* mas elementos */
   return;
}
fila=2+espacio-movfila;
_setcursortype(_cursoraddress(handle,fila,col);  /* direcciona el cursor */
gets(dator);                                     /* pide dato por teclado */
if(strcmp(dator,salir)==0 || strcmp(dator,SALIR)==0){  /* si digita fin o FIN sale */
   par=VERDAD;
   return;
}
if(strcmp(dator,edingr)==0){                     /* si digita * salta a modo edicion */
 datcol=4;
 break;
}
prueba=verificar(3,dator);                       /* revisa que dato no tenga mas de 3 caracteres */
if(prueba==1){
 strcpy(ficha[espacio].rea,dator);               /* asigna dato en fichero */
 datcol=2;
 col=col+4;                                      /* nueva columna */
}
}
```

```
case 2:                                  /* case para dato del tipo de elemento */
   vs_curaddress(handle,fila,col);
   gets(datoc);                          /* pide dato por teclado */
   prueba=verificar(4,datoc);            /* identifica si el dato es mayor de 4 caracteres */
   if(prueba==1){
     i=1;
     do {
       if(strcmp(lib[i++].clase,datoc)==0){   /* comprueba que existe dato en la libreria */
         strcpy(ficha[espacio].cmp,datoc);    /* ingresa dato en fichero */
         ficha[espacio].modo=i-1;
         ncomp=ncomp+1;                  /* incrementa el numero de componentes */
         datoi=3;
         col=col+5;                      /* reubica el cursor */
       }
     while(i<31);                        /* 30 cantidad de elementos en libreria */
     if(i>30)
       verificar(0,datoc);
   }
 break;

case 3:                                  /* case para dato de comentario de elemento */
 vs_cureddress(handle,fila,col);
 gets(datov);                            /* pide dato por teclado */
 prueba=verificar(6,datov);              /* verifica que dato no sea mayor de 6 caracteres */
 if(prueba==1){
   strcpy(ficha[espacio].valor,datov);   /* asigna dato en fichero */
   datoi=1;
   col=col-9;                            /* reubica coordenadas */
```

```
}
break;

case 4:                                    /* case para modo edición */
datol=1;                                   /* inicializa case */
borrar_lin();                              /* rutinas para borrar */
gotoxy(handle);                            /* componentes de la */
lista_lugr(ncomp);                         /* lista de ingreso */
gotoxy_curaddress(handle,25,2);            /* direcciona el cursor para colocar comentario */
gotoxy_insert(handle,"Para Borrar digite <*>   Para Finalizar digite <fin>");
break;

                                           /* final del switch */
                                           /* final del while */
                                           /* final de introducir */
```

lcomp()

Funciones Requeridas.- v_gtext

Detalle.- Usamos lcomp para visualizar la ubicación de todos los elementos con su respectivo nombre.

es asi que un lazo que numera la cantidad de componentes, pide los datos del tipo de elemento en la libreria y con las coordenadas traidas desde el fichero ubica en la pantalla cada componente con su respectivo nombre mediante la funcion vgtext.

Finalmente hay que anotar que el ratón debe ser ocultado durante el dibujo para que no se crucen y se pierda parte del grafico.

Diagrama de Flujo

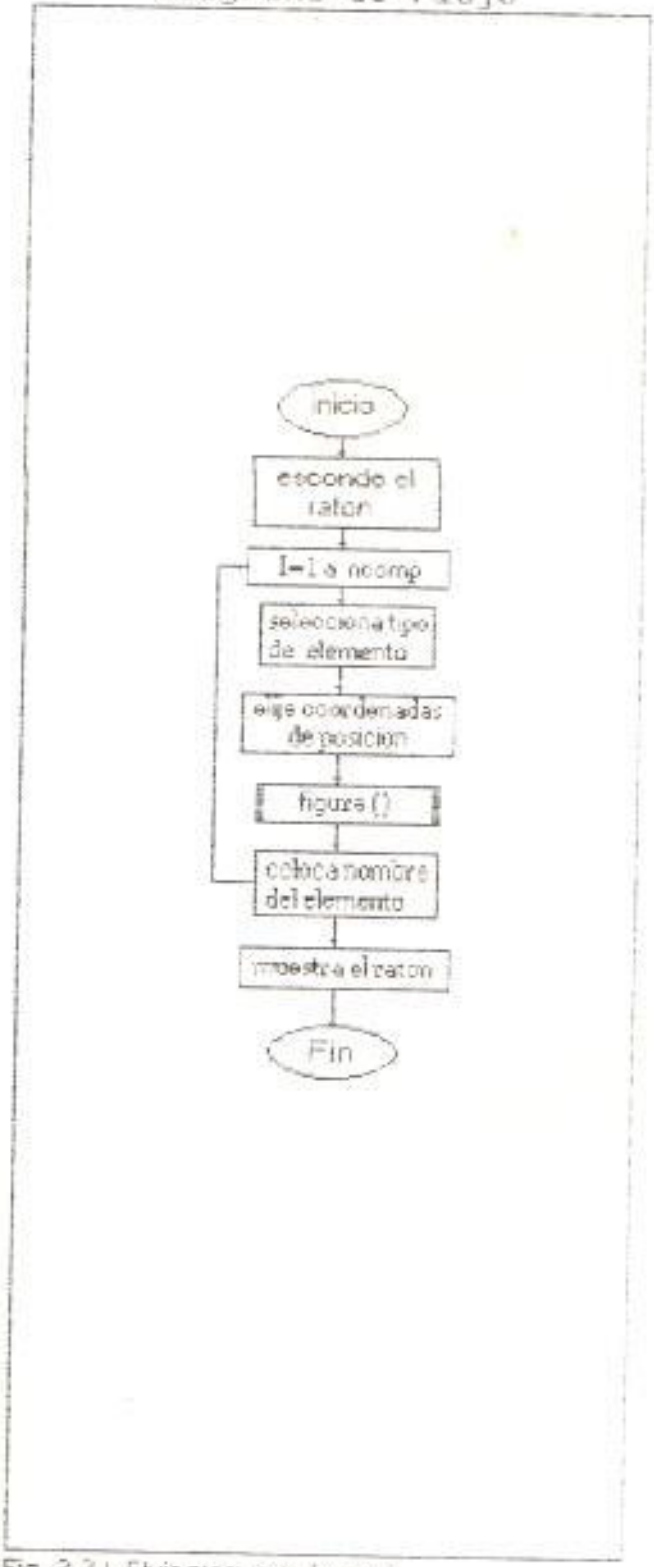


Fig. 3.3: Flujograma de formp1

```
tempx,tempy;                                    /* declara variables enteras */

                                                /* esconde el ratón */

ja<ncomp;ja++){                                 /* realiza lazo ncomp veces */

ficha[ja].modo;                                 /* selecciona forma de componente */

ficha[ja].compx;                                /* asigna coordenadas de colocación */

ficha[ja].compy;

tempx,tempy);                                   /* para displayar componente en pantalla */

handle,8*tempx+35+figw/2,7*tempy+40,ficha[ja].rem);  /* con su nombre encima */


                                                /* enseña el ratón */
```

lconex()

Función Requerida.- v_gtext

Descripción.- Esta es un rutina que nos sirve de consulta durante el manejo del programa.

Su función es la de visualizar en la misma pantalla gráfica un listado de las conecciones con que trabajamos, para ello la función v_gtext escribe según sus coordenadas los textos correspondientes en la ventana menor un listado por filas.

Lconex recibe un dato de entrada que contiene un valor numérico que indica desde que ficha de conección desea el usuario displayar el texto, pues esta rutina visualiza un listado agrupado de 20 conecciones.

Diagrama de Flujo

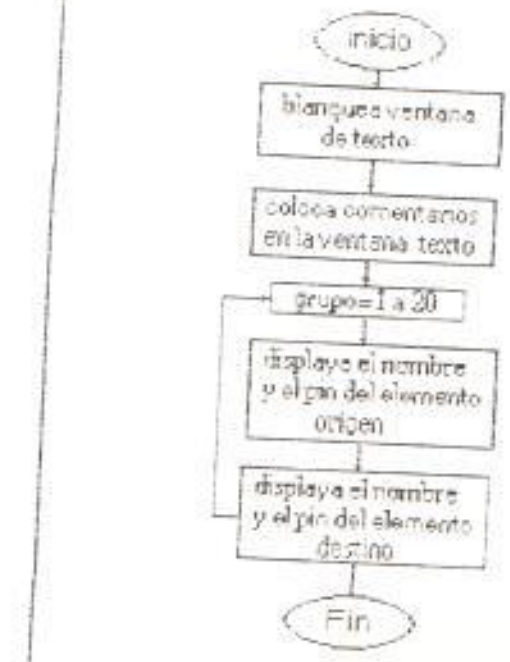


Fig 3.3 m Flujograma de Iconiex(1

```
                                        /* numero de conección elejida */

.idn;

_texto,WF_NAME,"CONECCION",0,0);        /* comentario en ventana menor */
_texto,WF_INFO,"CMP PIN  CMP PIN",0,0); /* segundo comentario en ventana menor */

+19;j++){                               /* ordena por grupos de 20 conecciones */

[j];                                    /* identifica número de pin origen */
[j];                                    /* identifica componente origen */
handle,402,66+8*t,ficha[idc].rem);
handle,510,66+8*t,lib[idn].num);
[j];                                    /* identifica número de pin destino */
[j];                                    /* identifica componente destino */
handle,555,66+8*t,ficha[idc].rem);
handle,585,66+8*t,lib[idn].num);
                                        /* incrementa fila */
                                        /* fin de if */
t:
                                        /* fin de for */
```

posicionar()

Funciones Requeridas.- evnt_button
graf_dragbox
evnt_mouse
v_gtext
wind_set
return

Descripción.- Esta parte del programa es la que se encarga de ingresar la ubicación de los componentes sobre la rejilla de trabajo.

Será entonces necesario contar con la ayuda de las funciones del ratón. Un lazo contará el número de componentes que ya fueron ingresados y evnt_button esperará que el botón del ratón sea presionado y entonces con los datos de ancho y largo del elemento la función graf_dragbox dibujará el perimetro del componente y asi arrastrarlo dentro de la rejilla.

Finalmente dibujará el elemento cuando suelte el botón del ratón.

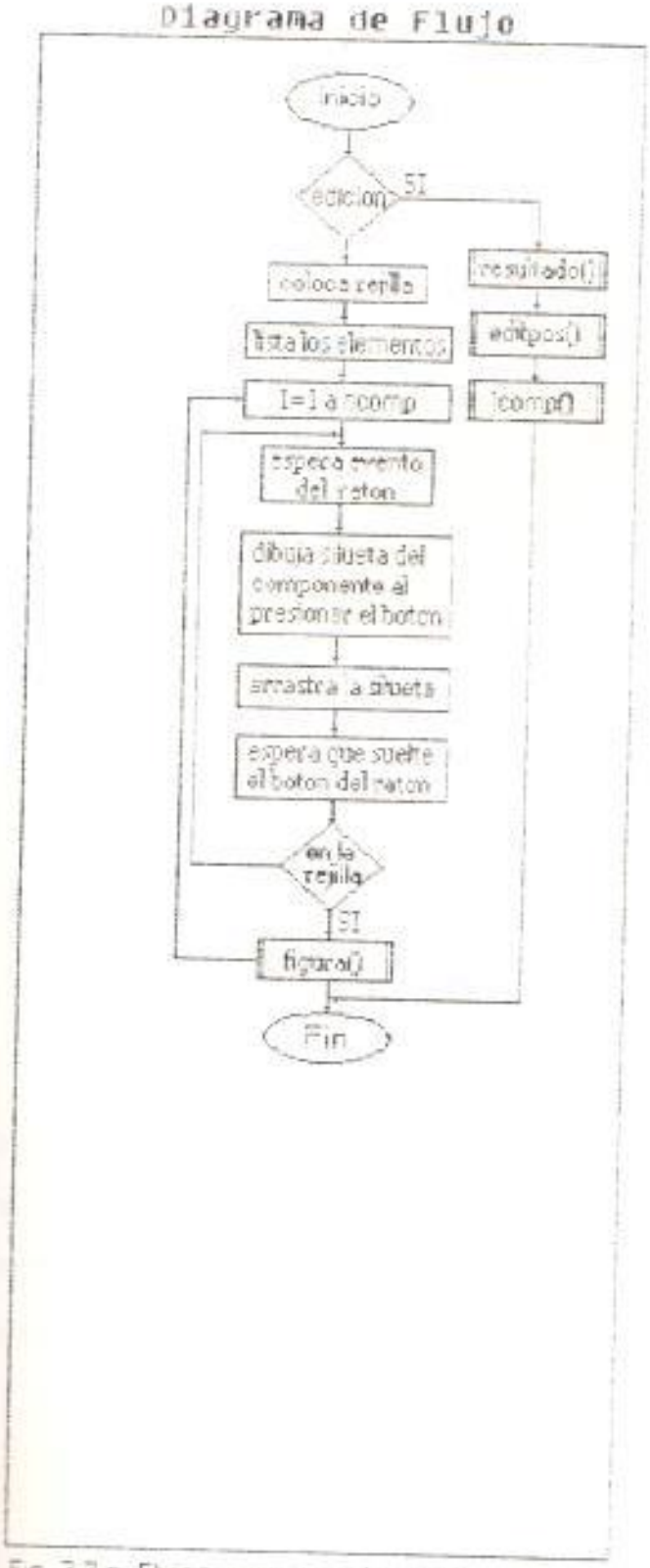


Fig. 3.3.n Flujograma de posicionar()

```
                                                    /* declara variables internas */
                                                    /* si está en modo edición */
                                                    /* ejecuta */
                                                    /* comentarios a la ventana */
                                                    /* menor */
                                                    /* realiza lazo ncomp veces */
                                                    /* coloca textos */
                                                    /* en la ventana */
                                                    /* menor */
                                                    /* si presiona tecla ALT */
                                                    /* tipo de componente */
                                                    /* ancho y alto del elemento */
```

```
mpx=(posx-35)/8+1;    tempy=(posy-40)/7+1;          /* coordenadas del componente */
ut_mouse(0,xdesk,ydesk,wdesk,hdesk,&tx,&ty,&no_pec,&no_pec);
(tx<400 && tx>39 && ty>44 && ty<350){               /* si está dentro de la rejilla */
transito=1;
_raton();
ficha[i].cmpx=tempx; ficha[i].cmpy=tempy;           /* asigna coordenadas en un arreglo */
figura(tempx,tempy);                                /* displaya figura */
_gtext(handle,8*tempx+35+figw/2,7*tempy+40,ficha[i].rem);
_gtext(handle,515,66+8*i,ficha[i].cmp);
_raton();
                                                    /* final de if */
transito!=1)   i--;
=0;
                                                    /* final de for */
```

presentación()

Funciones Requeridas.- graf_mouse
graf_growbox
form_center
form_dial
objc_draw
graf_shrinkbox

Descripción.- al cargar el programa se llama a esta rutina para realizar la presentación del programa.

Una etiqueta de diálogo con la información del programa aparecerá en la pantalla y aguardará a que el usuario seleccione la caja seguir.

La creación de esta etiqueta fue realizada con el programa MMRCS.PRG incluida en el paquete del compilador Megamax C. Una información al respecto la encontrará en el manual de operación del compilador.

Diagrama de Flujo

Fig 3.3.5 Flujograma de presentacion()

```
ion()

ouse(ARROW,&no_rec);                                    /* define puntero tipo flecha */
rowbox(xdesk+wdesk/2,ydesk+hdesk/2,gl_wbox,gl_hbox,xdesk,ydesk,wdesk,hdesk);
nter(objeto_presenta,&xcaja,&ycaja,&wcaja,&hcaja);      /* centra la caja de diálogo */
al(FMD_START,xcaja,ycaja,wcaja,hcaja,0,0,0,0);          /* inicia la caja */
w(objeto_presenta,0,2,xcaja,ycaja,wcaja,hcaja);         /* dibuja la caja de diálogo */
();                                                     /* define nuevo tipo de ratón */
al != VERDAD){
_do(objeto_presenta,0);                                 /* interactua con el usuario */
_ratón();                                               /* esconde el ratón */
_dial(FMD_FINISH,xcaja,ycaja,wcaja,hcaja,0,0,0,0);
=VERDAD;

box(0,0,0,0,xdesk,ydesk,wdesk,hdesk);                   /* cierra perimetro de la caja de diálogo */
```

Raton_Nuevo()

Funciones Requeridas.- vsc_form

graf_mouse

Detalle.-Crea una forma nueva del indicador del raton. Por ejemplo, para llamar la forma de una mano abierta se usa:

graf_mouse(4,&no_rec);

o graf_mouse(FLAT_HAND,&no_rec);

Para nuestro programa, creamos una nueva figura (que no existe en la libreria gráfica) con la leyenda "POLI". Para realizarlo nos valemos de la funcion vsc_form, que contiene dentro de un arreglo de 36 palabras la siguiente información:

Forma[0] = Coordenadas x del indicador, sobre la pantalla.

Forma[1] = Coordenadas y del indicador, sobre la pantalla.

Forma[2] = Debe ser 1.

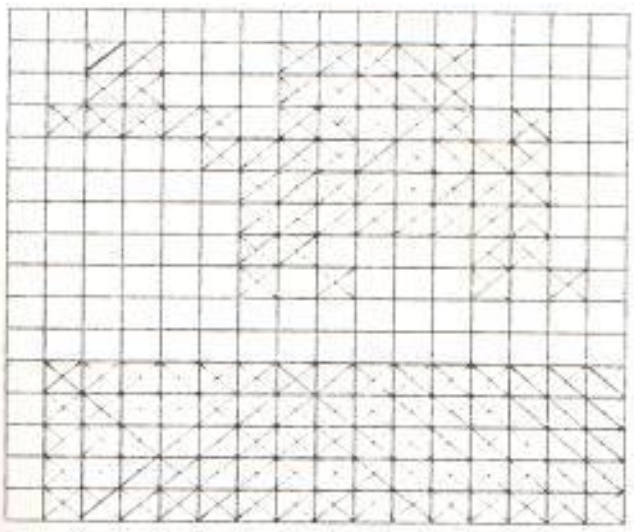
Forma[3] = Indice del color de fondo de la figura.

Forma[4] = Indice del color de superficie de la figura.

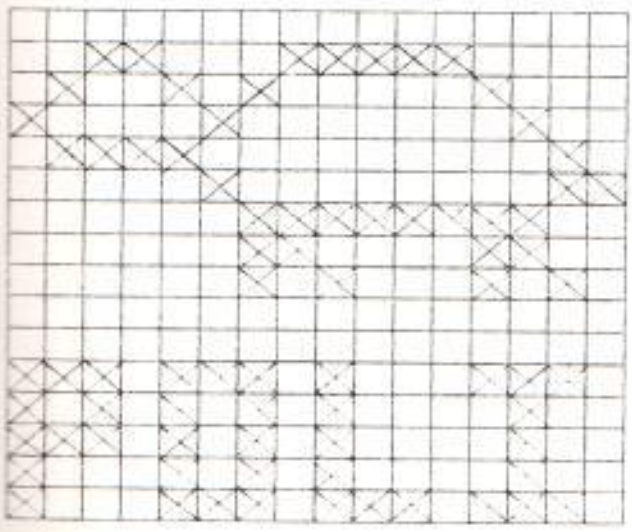
Forma[5] - Forma[20] = Bits que representan la figura del fondo.

Forma[21] - Forma[36]= Bits que representan la figura de superficie.

Una mejor visión de los datos que hay que proporcionar al arreglo se obtendrá a partir del siguiente gràfico que representan la vista superior y el fondo de la figura; los datos obtenidos en el margen derecho es el equivalente decimal del numero binario obtenido.



0
12784
12784
31796
2044
1020
1020
780
650
0
0
32767
32767
32767
32767
32767



0
12784
18952
33796
30722
1027
1020
780
650
0
0
61070
43652
60036
35460
36590

Diagrama de Flujo

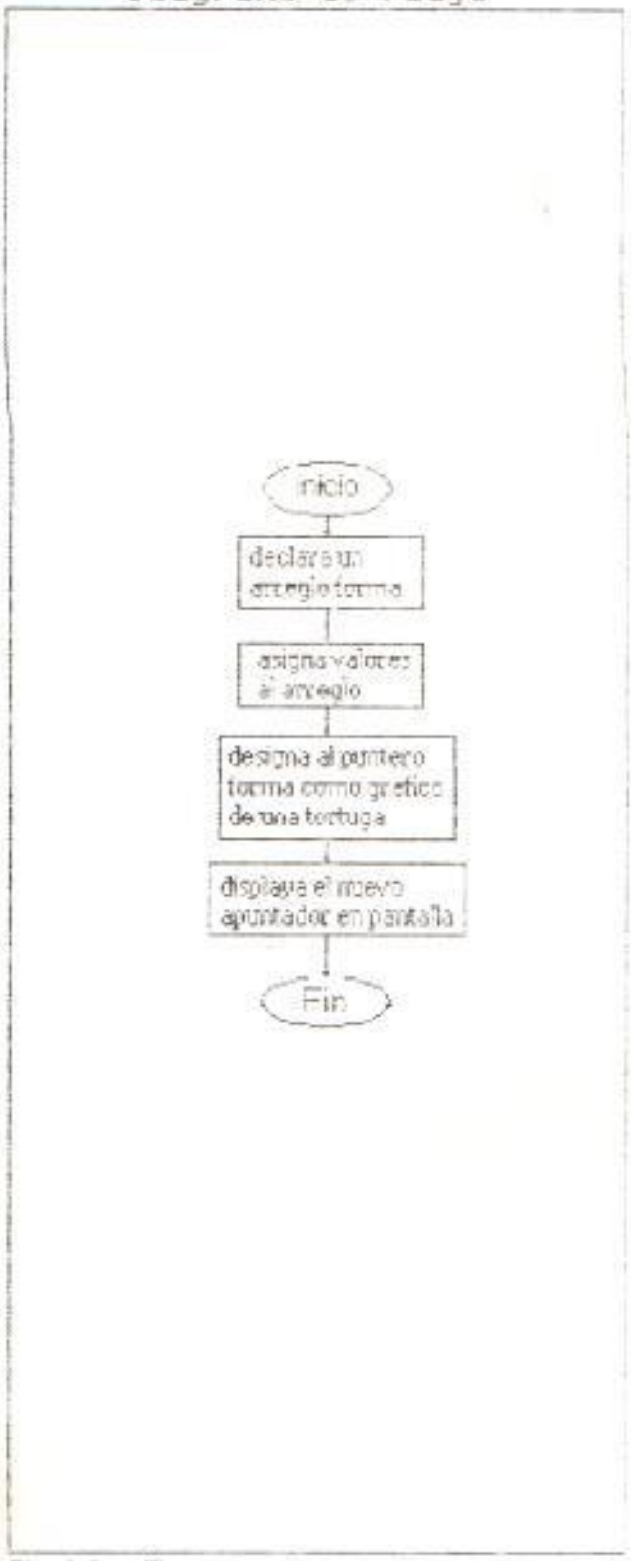


Fig. 3.3.p Flujograma de raton_nuevo()

```
/*****************************************/
/*      Definir indicador de ratón       */
/*****************************************/

                /* Declara un arreglo entero */
    forma[1]=100;    forma[2]=1;
    forma[4]=1;      forma[5]=0;
    forma[7]=12784;  forma[8]=31796;
    forma[10]=1020;  forma[11]=1020;
    forma[13]=650;   forma[14]=0;
    forma[16]=32767; forma[17]=32767;
    forma[19]=32767; forma[20]=32767;
    forma[22]=12784; forma[23]=18952;
    forma[25]=30722; forma[26]=1027;
    forma[28]=780;   forma[29]=650;
    forma[31]=0;     forma[32]=61070;
    forma[34]=60036; forma[35]=35460;
&forma);        /* adquiere los datos */
&forma);      /* despliega en pantalla */
```

recorre_h()

Función Requerida.- return

Detalle.- Esta es la rutina que se encarga de verificar que en el trayecto de una ruta no exista puntos ocupados.

Dos son los datos de entrada que corresponden a las coordenadas de dos cuadros de la rejilla de trabajo, con ellos se procede a rastrear todos los cuadros intermedios y si existe alguno que ya esté ocupado, la rutina retornará un valor 0, de lo contrario retornará un valor 1.

Diagrama de Flujo

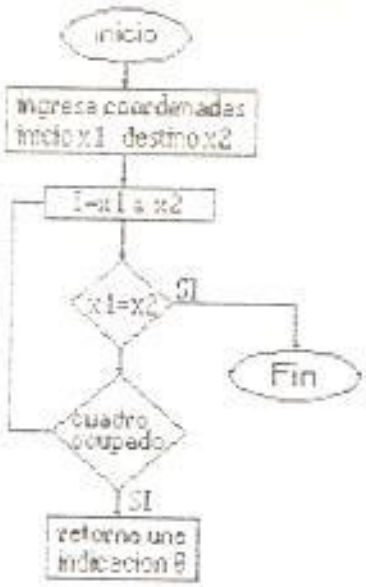


Fig. 3.3.c Diagrama de recorre_h()

```
x1,x2)

                                        /* variables de entrada normalizadas */


x1<=x2){                                /* mientras x1 no alcance a x2 */
x1==x2)   return 1;                     /* si son iguales */
x1++;                                   /* apunta horizontalmente al siguiente cuadro */
tado[x1]>=68)   return 0;               /* retorna un 0 si el cuadro está ocupado */
```

recorre_v()

Función Requerida.- return

Detalle.- Al igual que la rutina recorre_h() esta sensa que la ruta vertical entre dos puntos este despejada.

Para ello necesita recibir primero los datos de las coordenadas de los puntos y rastrear los cuadros intermedios; dos posibles valores pueden ser retornados: 0 si existe un puesto ocupado, y 1 si el camino se encuentra libre.

Diagrama de Flujo

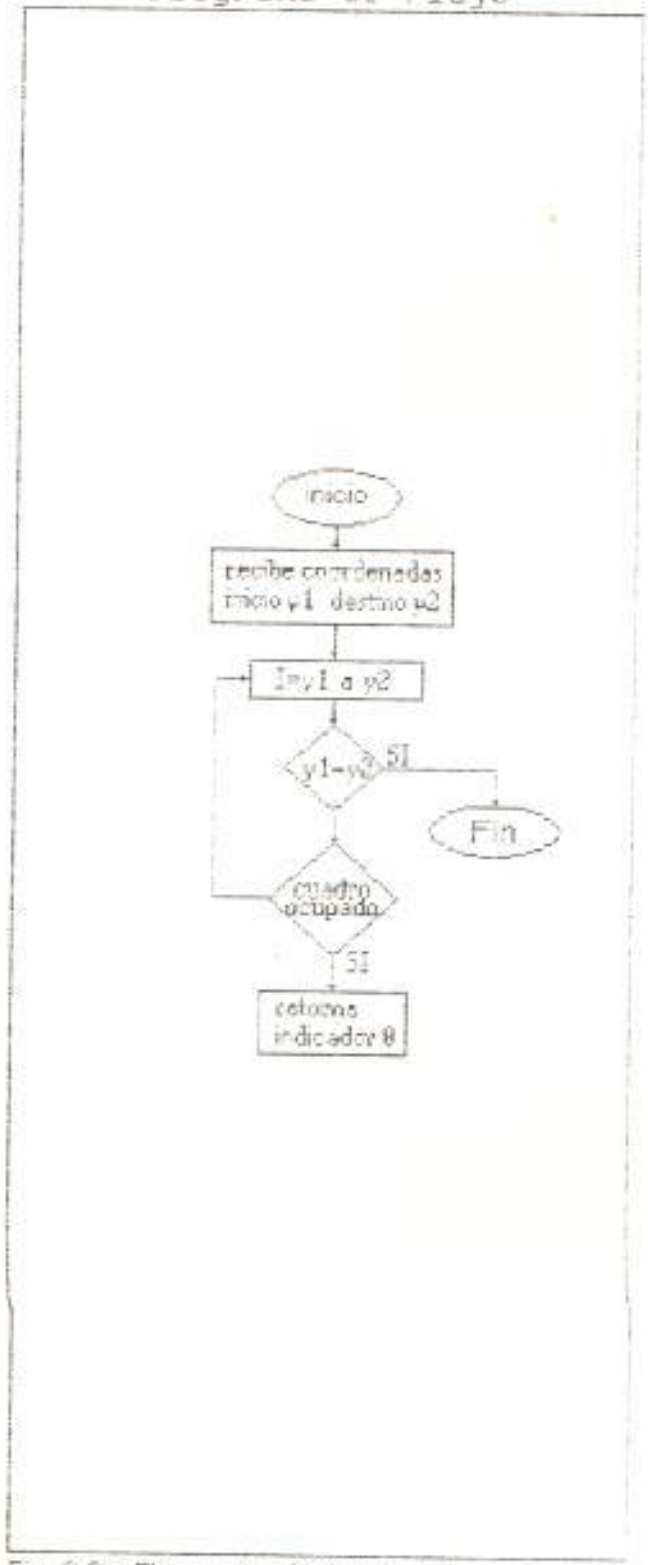


Fig. 3.3.r Flujograma de recorre_v1j

```
y1,y2)

                                        /* variables de entrada normalizadas */


<=y2){                                  /* mientras y1 no alcance y2 */
==y2)   return 1;                       /* si son iguales */
+=80;                                   /* apunta verticalmente al siguiente cuadro */
uado[y1]>=66)   return 0;               /* retorna 0 si el cuadro está ocupado */
```

ruta()

Función Requerida. _gtext

Descripción.- Esta es quizá la rutina más importante, pues contiene el recurso utilizado para enlazar dos puntos en una conección.

Para cumplir este objetivo, primero se recibe un dato numérico correspondiente al numero de la conección a ser tratada.

Una vez obtenido los datos de las coordenadas origen y destino se hallara un posible camino siguiendos los pasos que se describen:

- Están los puntos en linea recta horizontal ?
- Están los puntos en linea recta vertical?
- se prueba un enlace en forma de L
- Se prueba si hay un enlace en forma de L invertida
- intentar escalonadamente conectar dirigiendose a la derecha
- o intentarlo escalonadamente dirigiendose hacia la izquierda
- enlazarlos con un L invertida escalonadamente hacia la derecha
- enlazarlo con L invertida escalonadamente hacia la izquierda
- no se pudo conectar y hay que informarlo al usuario.

Diagrama de Flujo

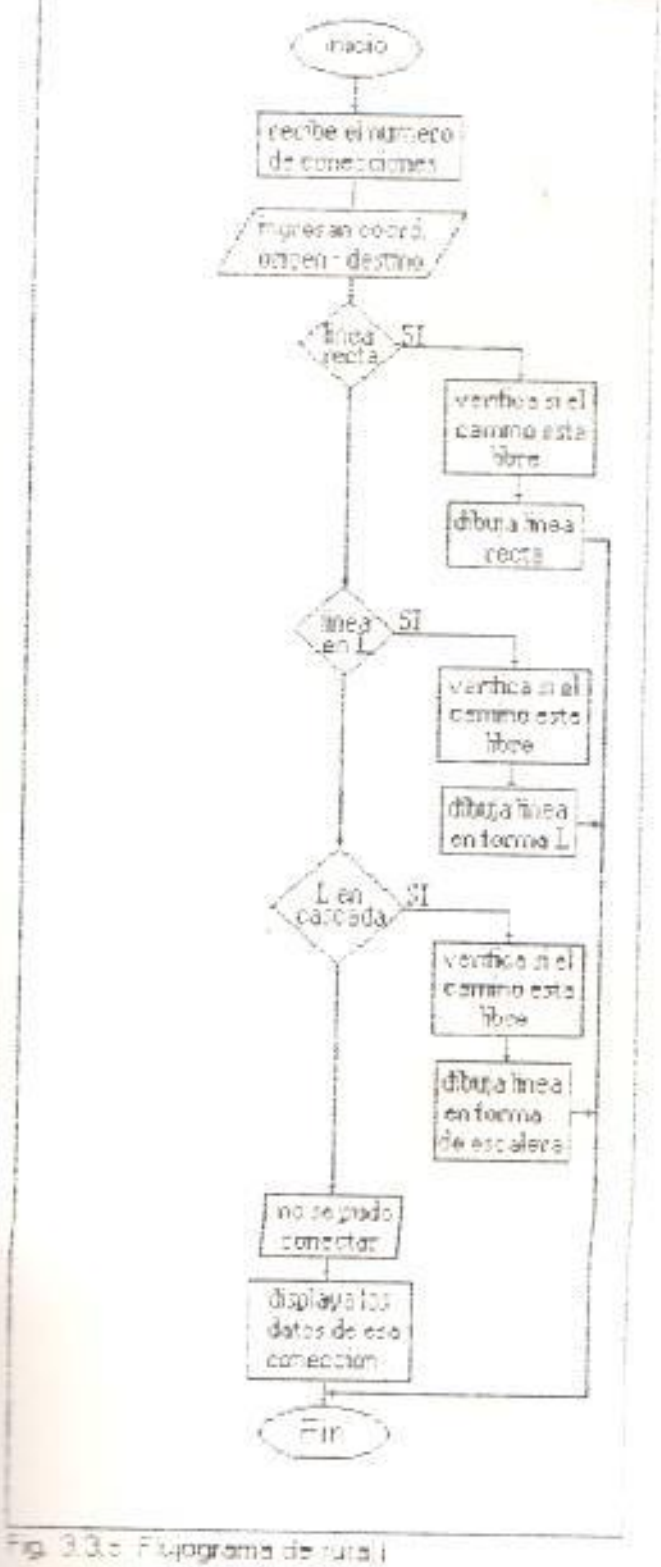


Fig. 3.3.c Flujograma de ruta 1

```
                                        /* número de la conección elejida */

de,tv,th,y1,y2,x1,x2,tt,alt;            /* declara variables enteras */
dc,idn;                                 /* declara enteros de 8 bits */

{
nd){                                    /* elije tipo de conecolón */
                                        /* dibuja recta horizontal */
[jc];       de=trd[jc];
50*8+35;    y1=or/50*7+40;              /* descompone coordenadas en x y */
50*8+35;    y2=de/50*7+40;
 == y2){                                /* si es una disposición horizontal */
-1;                                     /* ajusta los extremos */
orre_h(or,tt) == 0){                    /* revisa si el camino esta libre */
 nd=3;
 ;

(or,de);                                /* dibuja una recta horizontal */
                                        /* inicialize parametros */
}
```

```
2:                                          /* dibuja recta vertical */
tro[jc];        de=trd[jc];
or%50*8+35;     y1=or/50*7+40;              /* descompone coordenadas en x y */
de%50*8+35;     y2=de/50*7+40;
x1 == x2){                                  /* si es una disposición vertical */
or=tro[jc];     de=trd[jc];
if(recorre_v(or,de-50) == 0){               /* revisa si el camino está libre */
  mod=8;
  break;
}
direct(or,de);                              /* dibuja una recta vertical */
mod=1;                                      /* inicializa parametros */
i=0;
break;
}
}
                                            /* dibuja una recta en forma de L */
trd[jc];
recorre_v(or,de);                           /* calcula un vértice de L */
                                            /* si no hay sale de case */
}
}
}
direct(ord,de);                             /* dibuja recta L */
                                            /* inicializa parametros */
```

```
alt=0;
break;

e 4:                                    /* dibuja recta en forma de L invertida */
r=tro[j-1];    de=trd[jc];
td=ele_h(or,de);                        /* calcula vértice de L invertida */
[otd==0){                               /* si no hay salc de case */
 mod++;
 break;

ne{
 dibele(or,otd,de);                     /* dibuja la L invertida */
 mod=1;                                 /* inicializa parámetros */
 alt=0;
 break;

                                        /* dibuja L escalonada hacia la derecha */

50 > trd[jc]%50)   break;               /* determina la disposición de los terminales */
      de=trd[jc];
50-or%50);                              /* calcula un cuadro transitorio */
&& estado[++or]==64){                   /* recorre camino si está libre */
 ele_a(or,de);                          /* calcula vértice de L */
 != 0){                                 /* si puede ... */
   dibele(or,otd,de);                   /* dibuja recta escalonada */
   direct(tro[jc],or);
   estado[or]=78;                       /* ajusta el dato para imprimir */
```

```
    mod=1;                                  /* inicializa los parámetros */
    alt=0;
    or=tt;
  }

break;

                                            /* dibuja L escalonada hacia la izquierda */
++;
[jc]%50 > trd[jc]%50)   break;              /* determina disposición de terminales */
[jc];     de=trd[jc];
(de/50-or/50)*50;                           /* calcula cuadro transitorio */
=50;
or<tt && estado[or]==64){                   /* recorre el camino si está libre */
ele_b(or,de);                               /* clacula vértice de L */
otd != 0){                                  /* si hay, entonces */
  dibele(or,otd,de);                        /* dibuaj una linea */
  direct(tro[jc],or);                       /* escalonada */
  estado[or]=80;                            /* ajuste el dato para imprimir */
  mod=1;                                    /* inicializa los parámetros */
  alt=0;
  or=tt;

                                            /* fin de if */

                                            /* fin de while */
```

```
                                        /* dibuja L invertida y escalonada a la derecha */

    ...[jc]%50]   break;                /* determina disposición de los terminales */
    ...trd[jc];
    ...%50)%50;                          }* calcula el cuadro transitorio *|
    ...

    ...tt && estado[de]==64){            /* verifica que el camino esté libre */
    ...ele_a(or.de);                     /* calcula el vértice de la L */
    ...otd != 0){                        /* si hay un vértice */
      ...bele(or.otd,de);                /* dibuja una línea */
      ...ibrect(de,trd[jc]);             /* escalonada a la derecha */
      ...estado[de]=64;                  /* ajusta los datos para la impresora */
      ...=1;                             /* inicializa los parametros */
      ...=0;
      ...=tt;
    ...

    ...

    ...

                                         /* dibuja linea en L invertida a la izquierda */
    ...

    ...50 > tro[jc]%50)   break;        /* determina la disposición de los terminales */
    ... de=trd[jc];
    ...-de%50);                          }* calcula el cuadro transitorio *|
    ... && estado[++de]==64){            /* verifica que el camino esté libre */
    ...ele_b(or.de);                     /* calcula el vértice de L */
    ... != 0){                           /* si se puede */
```

```
    dibele(or,otd,de);                          /* dibuja una linea */
    dibrect(trd[jc],de);                        /* en forma de escalón */
    estado[de]=74;                              /* ajusta datos para la impresora */
    mod=1;                                      /* inicializa parámetros */
    alt=0;
    de=tt;
}

                                                /* si no puede conectar los terminales **/
                                                /* inicializa parámetros */

    [jc];       de=trd[jc];
=orcmp[jc];    idn=orpin[jc];                   /* identifica componente y pin origen */
ext(handle,482,66+8*ncc,ficha[idc].rem);
handle,510,66+8*ncc,lib[idn].num);
  [jc];    idn=depin[jc];                       /* identifica componente y pin destino */
handle,555,66+8*ncc,ficha[idc].rem);            /* y los presenta en la ventana menor */
handle,585,66+8*ncc,lib[idn].num);
```

terminal()

Función Requerida.- v_circle

Detalle.- Esta rutina tiene por finalidad, dibujar en pantalla la representación de los terminales de cada componente.

Tal como se observa, esta rutina requiere de dos datos de entrada (cx ,cy) que corresponden a las coordenadas x e y de la ventana gráfica y que indican referencialmente donde se ubicarán los terminales. Otro dato es la variable (mod) que por ser variable global no necesita ser declarada en esta rutina: (mod) es la variable que indica el conjunto de terminales agrupados en el (case) correspondiente.

Siguiendo un ejemplo anterior, si (mod) es igual a 7, dos arreglos cirx[] y ciry[] almacenarán las coordenadas de 8 terminales que corresponden a un integrado de ocho pines; y estas serán dibujadas en la pantalla mediante la función v_circle. Al mismo tiempo que realiza el dibujo, también se actualiza el arreglo estado[] con el valor 66: esto indica en el tablero de circuitos que puntos son terminales.

Diagrama de Flujo

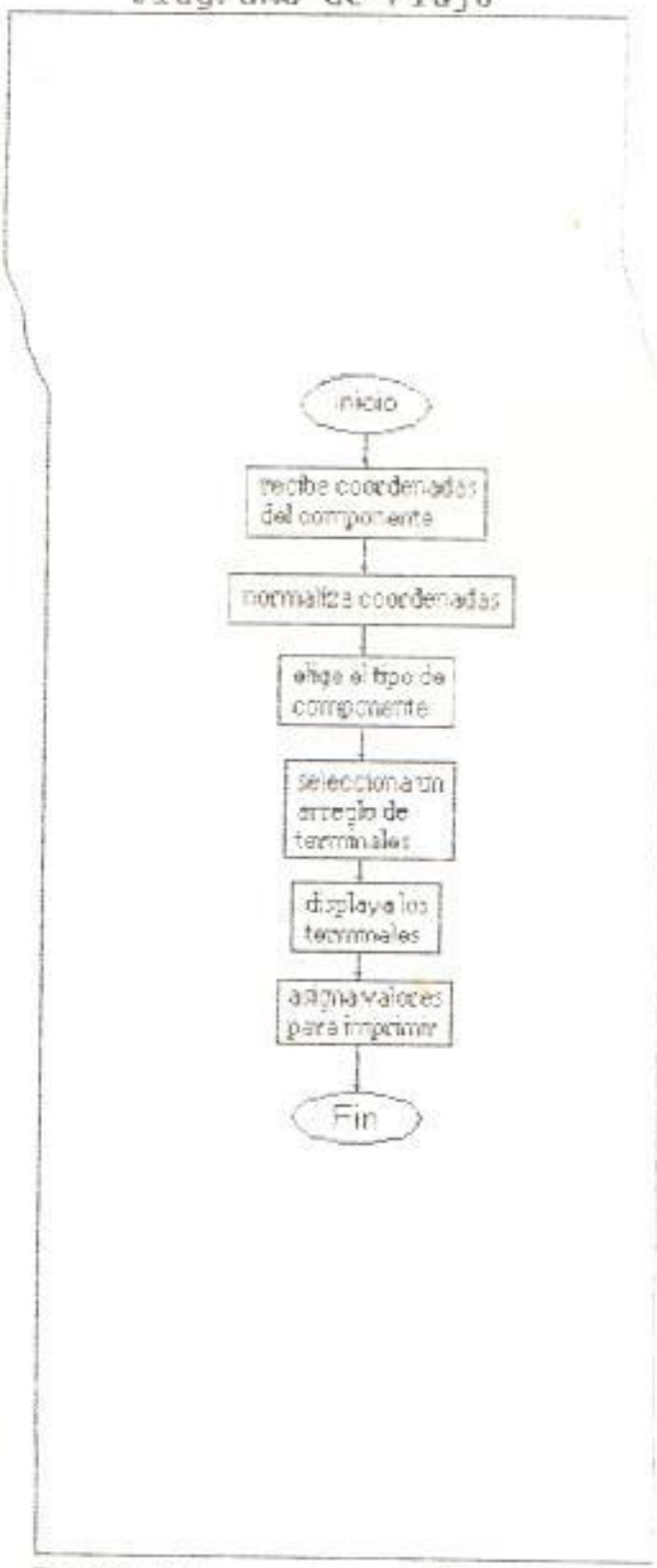


Fig 3.3: Flujograma de terminal()

```
cx,cy)
;                                   /* Asigna dos variables de entrada */

30],ciry[30],refx,refy,tip;         /* Variables internas */
 ncir;

   refy=40+7*cy;                    /* Coordenadas de pantalla normalizada */

                                    /* Elije un case de acuerdo al valor mod */
                                    /* Terminal correspondiente a un punto */

   ciry[1]=refy;

                                    /* Terminales de una resistencia de 1/4W */

   ciry[1]=refy;   cirx[2]=refx+5*ax;   ciry[2]=refy;

                                    /* Terminales de una resistencia de 1/2W */

   ciry[1]=refy;   cirx[2]=refx+6*ax;   ciry[2]=refy;

                                    /* Terminales de una resistencia de 1W */

   ciry[1]=refy+ay;   cirx[2]=refx+7*ax;   ciry[2]=refy+ay;
```

```
5:                                  /* Terminales de una resistencia de 2W */
cir=2;
irx[1]=refx-ax;   ciry[1]=refy+ay;    cirx[2]=refx+3*ax;   ciry[2]=refy+ay;
reak;
                                    /* Terminales de una resistencia de 5W */
=2;
=refx-ax;  ciry[1]=refy+2*ay;   cirx[2]=refx+10*ax;  ciry[2]=refy+2*ay;
                                    /* Terminales de un Integrado de 8 pines */
=8;
x[1]=refx;      ciry[1]=refy+3*ay; cirx[2]=refx+ax;   ciry[2]=refy+3*ay;
x[3]=refx+2*ax; ciry[3]=refy+3*ay; cirx[4]=refx+3*ax; ciry[4]=refy+3*ay;
x[5]=refx+3*ax; ciry[5]=refy;      cirx[6]=refx+2*ax; ciry[6]=refy;
x[7]=refx+ax;   ciry[7]=refy;      cirx[8]=refx;      ciry[8]=refy;
                                    /* Terminales de un integrado de 10 pines */
=refx;      ciry[1]=refy+3*ay; cirx[2]=refx+ax;   ciry[2]=refy+3*ay;
=refx+2*ax; ciry[3]=refy+3*ay; cirx[4]=refx+3*ax; ciry[4]=refy+3*ay;
=refx+4*ax; ciry[5]=refy+3*ay; cirx[6]=refx+4*ax; ciry[6]=refy;
=refx+3*ax; ciry[7]=refy;      cirx[8]=refx+2*ax; ciry[8]=refy;
=refx+ax;   ciry[9]=refy;      cirx[10]=refx;     ciry[10]=refy;
                                    /* Terminales de un integrado de 14 pines */
=refx;      ciry[1]=refy+3*ay; cirx[2]=refx+ax;   ciry[2]=refy+3*ay;
=refx+2*ax; ciry[3]=refy+3*ay; cirx[4]=refx+3*ax; ciry[4]=refy+3*ay;
=refx+6*ax; ciry[5]=refy+3*ay; cirx[6]=refx+5*ax; ciry[6]=refy+3*ay;
```

```
[7]=refx+6*ax;  ciry[7]=refy+3*ay; cirx[8]=refx+6*ax;  ciry[8]=refy;
[9]=refx+5*ax;  ciry[9]=refy;      cirx[10]=refx+4*ax; ciry[10]=refy;
[11]=refx+3*ax; ciry[11]=refy;     cirx[12]=refx+2*ax; ciry[12]=refy;
[13]=refx+ax;   ciry[13]=refy;     cirx[14]=refx;      ciry[14]=refy;
;

                                   /* Terminales de un integrado de 16 pines */
=16;
[1]=refx;       ciry[1]=refy+3*ay; cirx[2]=refx+ax;    ciry[2]=refy+3*ay;
[3]=refx+2*ax;  ciry[3]=refy+3*ay; cirx[4]=refx+3*ax;  ciry[4]=refy+3*ay;
[5]=refx+4*ax;  ciry[5]=refy+3*ay; cirx[6]=refx+5*ax;  ciry[6]=refy+3*ay;
[7]=refx+6*ax;  ciry[7]=refy+3*ay; cirx[8]=refx+7*ax;  ciry[8]=refy+3*ay;
[9]=refx+7*ax;  ciry[9]=refy;      cirx[10]=refx+6*ax; ciry[10]=refy;
[11]=refx+5*ax; ciry[11]=refy;     cirx[12]=refx+4*ax; ciry[12]=refy;
[13]=refx+3*ax; ciry[13]=refy;     cirx[14]=refx+2*ax; ciry[14]=refy;
[15]=refx+ax;   ciry[15]=refy;     cirx[16]=refx;      ciry[16]=refy;
;

                                   /* Terminales de un integrado de 14 pines */
=14;
[1]=refx;       ciry[1]=refy+6*ay;  cirx[2]=refx+ax;    ciry[2]=refy+6*ay;
=refx+2*ax;  ciry[3]=refy+6*ay;  cirx[4]=refx+3*ax;  ciry[4]=refy+6*ay;
=refx+4*ax;  ciry[5]=refy+6*ay;  cirx[6]=refx+5*ax;  ciry[6]=refy+6*ay;
=refx+6*ax;  ciry[7]=refy+6*ay;  cirx[8]=refx+7*ax;  ciry[8]=refy+6*ay;
=refx+6*ax;  ciry[9]=refy+6*ay;  cirx[10]=refx+9*ax; ciry[10]=refy+6*ay;
=refx+5*ax; ciry[11]=refy+6*ay; cirx[12]=refx+11*ax; ciry[12]=refy+6*ay;
=refx+3*ax; ciry[13]=refy;      cirx[14]=refx+10*ax; ciry[14]=refy;
=refx+ax;   ciry[15]=refy;      cirx[16]=refx+8*ax;  ciry[16]=refy;
=refx;      ciry[17]=refy;      cirx[18]=refx+6*ax;  ciry[18]=refy;
```

```
[19]=refx+5*mx;  cry[19]=refy;      cirx[20]=refx+4*mx;  cry[20]=refy;
[21]=refx+3*mx;  cry[21]=refy;      cirx[22]=refx+2*mx;  cry[22]=refy;
[23]=refx+mx;    cry[23]=refy;      cirx[24]=refx;       cry[24]=refy;
;

                                   /* Terminales de un diodo D0-7 */
=2;
[1]=refx;   cry[1]=refy;   cirx[2]=refx+3*mx;   cry[2]=refy;
;

                                   /* Terminales de un diodo D0-27 */
=2;
[1]=refx;    cry[1]=refy;    cirx[2]=refx+4*mx;    cry[2]=refy;
;

                                   /* Terminales de un diodo de 1# */
=2;
[1]=refx;    cry[1]=refy;    cirx[2]=refx+5*mx;    cry[2]=refy;
;

                                   /* Terminales de un diodo de 5# */
=2;
[1]=refx;    cry[1]=refy;    cirx[2]=refx+6*mx;    cry[2]=refy;
;

                                   /* Terminales de un transistor T0-220 */
=3;
=mx;   cry[1]=refy;  cirx[2]=refx+2*mx;    cry[2]=refy;
=3*mx; cry[3]=refy;
;
```

```
                                    /* Terminales de un transistor TO-92 */
=3;
[1]=refx;          ciry[1]=refy+ay;  cirx[2]=refx+ax;  ciry[2]=refy+ay;
[3]=refx+2*ax;     ciry[3]=refy+ay;

                                    /* Terminales de un transistor TO-18 */
=3;
[1]=refx;          ciry[1]=refy+ay;  cirx[2]=refx+ax;  ciry[2]=refy;
[3]=refx+2*ax;     ciry[3]=refy+ay;

                                    /* Terminales de un transistor TO-3 */

=refx+8*ax;  ciry[1]=refy+3*ay;  cirx[2]=refx+8*ax;  ciry[2]=refy+7*ay;

                                    /* Terminales de un Capacitor Electrolitico */

=refx;  ciry[1]=refy+2*ay;  cirx[2]=refx+10*ax;  ciry[2]=refy+2*ay;

                                    /* Terminales de un capacitor electrolitico */

=refx;  ciry[1]=refy+2*ay;  cirx[2]=refx+12*ax;  ciry[2]=refy+2*ay;

                                    /* Terminales de un capacitor electrolitico */

=refx;  ciry[1]=refy+2*ay;  cirx[2]=refx+17*ax;  ciry[2]=refy+2*ay;
```

```
                                /* Terminales de un capacitor electrolitico */
2:
1]=refx;  ciry[1]=refy+4*ay;  cirx[2]=refx+2*ax;  ciry[2]=refy+4*ay;
:
                                /* Terminales de un capacitor electrolitico */
2:
1]=refx+ax;  ciry[1]=refy+ay;  cirx[2]=refx+3*ax;  ciry[2]=refy+ay;
:
                                /* Terminales de un capacitor electrolitico */
2:
1]=refx+2*ax;  ciry[1]=refy+3*ay;  cirx[2]=refx+5*ax;  ciry[2]=refy+3*ay;
:
                                /* Terminales de un capacitor electrolitico */
2:
1]=refx+2*ax;  ciry[1]=refy+4*ay;  cirx[2]=refx+6*ax;  ciry[2]=refy+4*ay;
:
                                /* Terminales de un capacitor cerámico */
2:
1=refx;  ciry[1]=refy;  cirx[2]=refx+3*ax;  ciry[2]=refy;
:
                                /* Terminales de un capacitor cerámico */
2:
1=refx;  ciry[1]=refy+ay;  cirx[2]=refx+5*ax;  ciry[2]=refy+ay;
:
                                /* Terminales de un capacitor cerámico */
2:
1=refx;  ciry[1]=refy+2*ay;  cirx[2]=refx+7*ax;  ciry[2]=refy+2*ay;
:
```

```
};                                  /* Terminales de un capacitor cerámico */
ir=2;
x[1]=refx;  ciry[1]=refy+3*my;  cirx[2]=refx+10*mx;  ciry[2]=refy+3*my;
nt;

j<=noir;j++){                       /* Lazo para contar número de terminales */
rcle(handle,cirx[j],ciry[j],radio);  /* Dibuja un circulo en coordenadas dadas */
cirx[j]-35)/8+((ciry[j]-40)/7-1)*50;  /* Convierte coordenadas x/y en tip */
[tip]=66;                           /* Codifica dicha coordenada como Terminal */
[tip]=temp2;                        /* Almacena nombre del componente */
[tip]=j;                            /* Número referencial del componente */
                                    /* fin de switch */
                                    /* fin de rutina */
```

C A P I T U L O I V

EL MANUAL DEL USUARIO

4.1 OBJETIVO

El objetivo de este capitulo, es el de brindar al usuario toda la información necesaria para que aprenda a manejar el programa "DCI.PRG". Esto incluye:

Conocer los comandos más importantes del programa "Diseñador de Circuitos Impresos", de qué archivos se compone, cómo utilizar los periféricos, con qué información necesita alimentar el sistema, cuáles son los resultados obtenidos, etc.

En definitiva, conocerá cuales son las caracteristicas técnicas más sobresalientes del programa y por que no, cuales son sus limitaciones. Además, como una ayuda didáctica, todos los temas tratados en este capitulo estarán acompañados de gráficos que representan lo que usted verá por el monitor, y para reforzar su aprendizaje, incluimos un ejemplo práctico que resolveremos a lo largo de este tratado.

4.2 REQUISITOS NECESARIOS PARA USAR ESTE PROGRAMA

Para correr el programa DCI.PRG debe como primer paso conocer los equipos que va a utilizar.

- Un computador ATARI 1024ST u otro de la misma serie.
- Un Monitor Monocromático de alta resolucion (640*400 puntos).
- Una Impresora Epson EX-1000 u otro modelo que se ajuste a los comandos que se describen en la rutina Impdiag() (ver Capitulo III).

  Consulte con el manual de operación de su impresora para confirmar la existencia de los comandos, en caso contrario no podrá realizar los gráficos del diseño final.
- Un disco de 3 1/2 pulgadas que contenga el programa "DCI.PRG" y los archivos DCI.RSC, DCI.DEF, DCI.H y LIBRERIA.CAD.
- Un ratón conectado al puerto cero del computador.

Una vez que disponga de estos equipos puede proceder a cargar el programa, cumpliendo los siguientes pasos:

- En primer lugar encienda el monitor.
- Luego haga lo mismo con el computador.

- Introduzca el disco que contenga el programa y espere que aparezca el directorio del mismo sobre la pantalla (ver fig. 4.2.a).

- La impresora podrá prenderla en cualquier momento, pero siempre antes de seleccionar un impreso.

- Para abrir un archivo, fíjese bien que tenga el identificador PRG, lo cual significa que este archivo es ejecutable.

- Para ejecutarlo, dirija el apuntador del ratón hacia el archivo DCI.PRG y una vez que esté sobre él, presione el botón izquierdo del ratón una vez; en ese momento el archivo DCI.PRG aparecerá en modo invertido; finalmente sin mover el ratón presione dos veces seguidas el boton izquierdo del ratón. El programa será entonces ejecutado.

4.3 PASOS PARA CREAR UN DISEÑO NUEVO

Una vez que se ejecuta el DCI.PRG aparecerá en la pantalla la presentación del programa (fig. 4.3.a) y aguardará a que usted presione la tecla <return> o seleccione [seguir] con el ratón.

En este momento dispondrá sobre la pantalla su área de trabajo, compuesto de dos ventanas; la mayor es la ventana de graficos y la segunda la ventana de texto, ademas en la parte superior tendrá el menú de

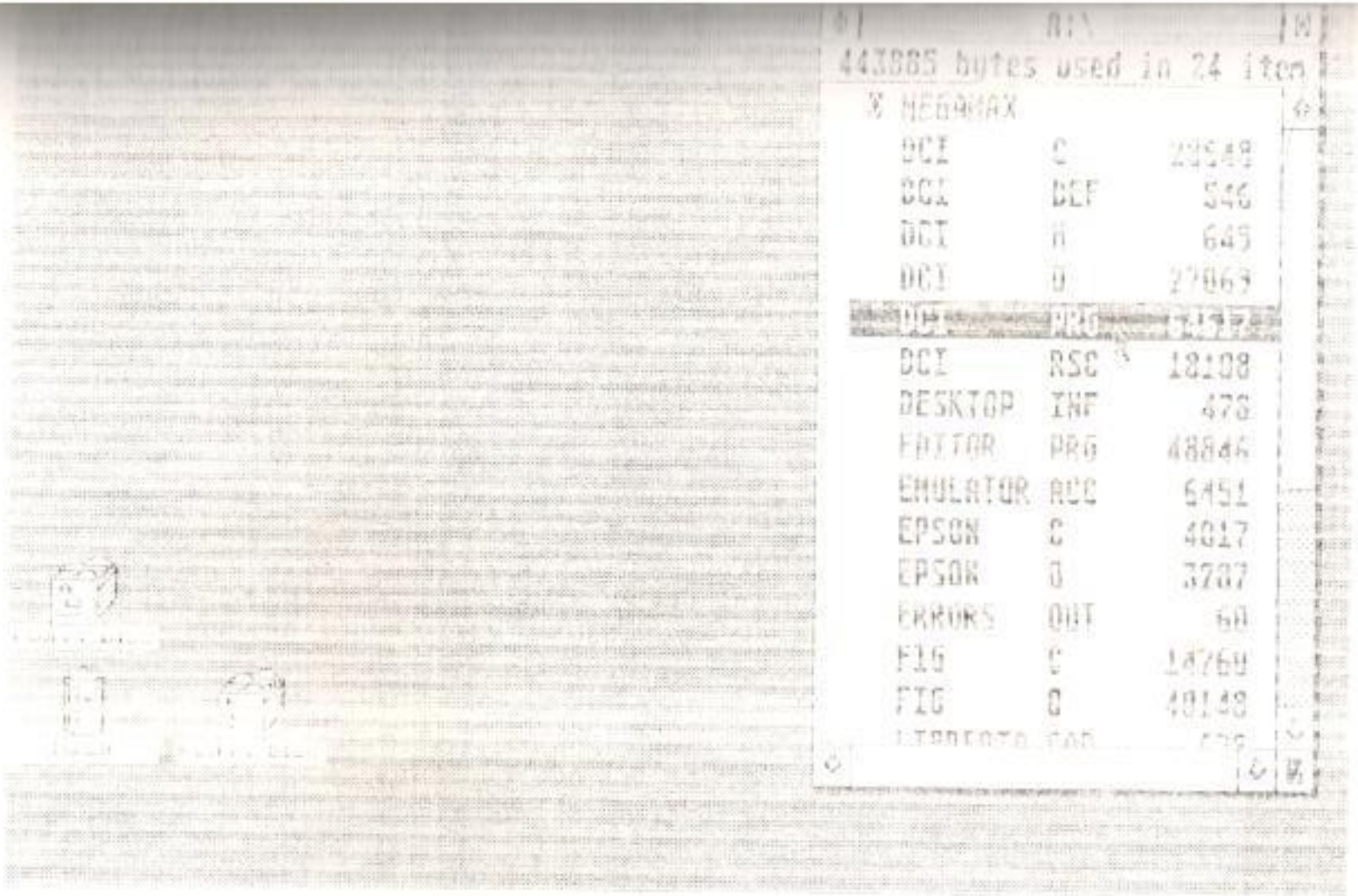


fig. 4.2.a Imagen de la pantalla al encender el CPU

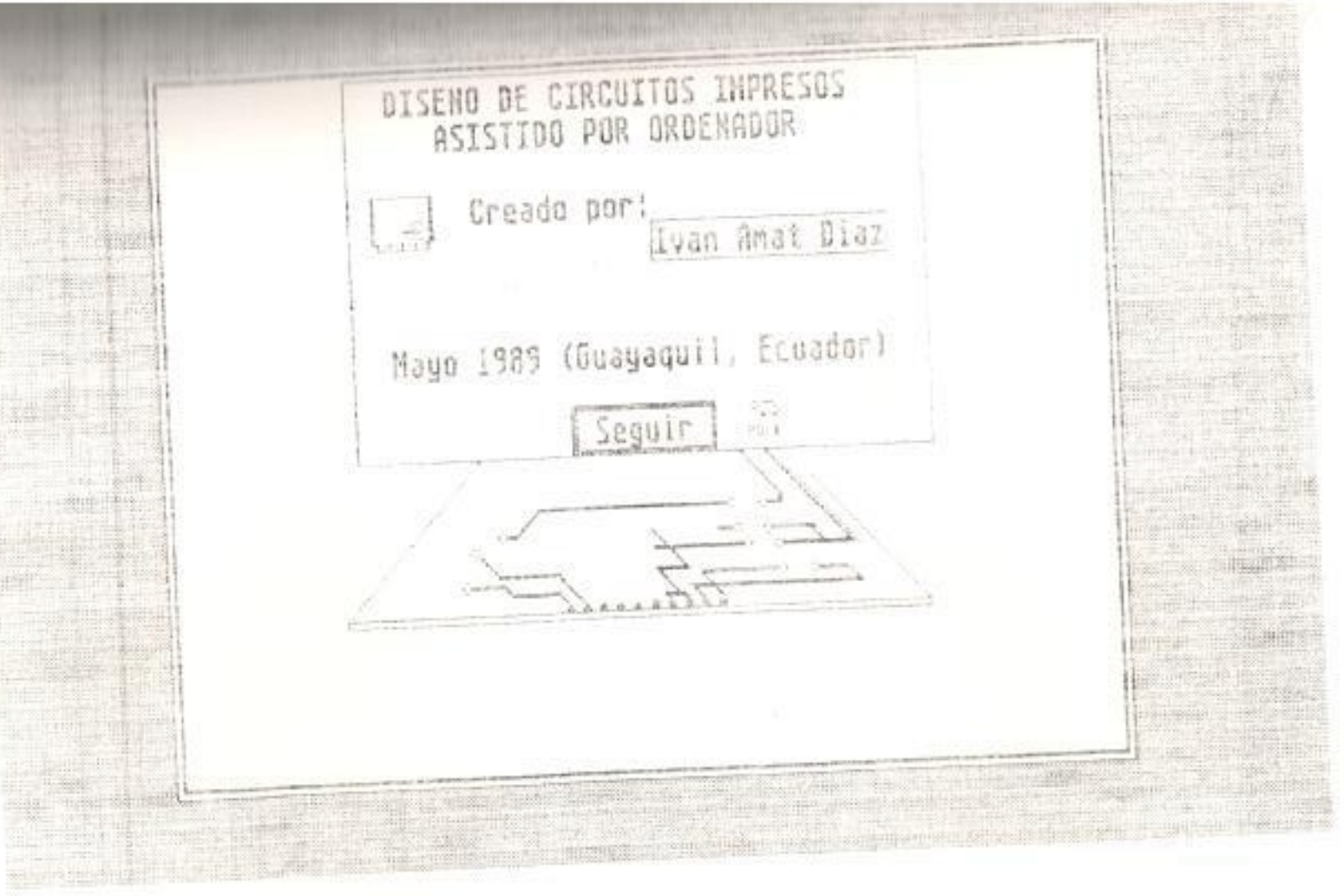


fig. 4.3.a Etiqueta de presentación del programa.

comandos que podrá seleccionar sólo con el ratón.

Para crear un nuevo diseño, guíe el ratón hacia el menú **Archivo** (fig. 4.3.b) y bajando el apuntador seleccione **Nuevo** presionando el botón izquierdo del ratón una vez. Un mensaje aparecerá informándole que en adelante seguirá los pasos del menú **Edición** (fig. 4.3.c), presione <Return>.

4.3.1 INGRESO DE ELEMENTOS

Dirija el ratón hacia el menú **Edición** y elija el comando **Ingresar** (fig. 4.3.1.a), esta es la etapa de ingreso de datos y para ello utilizaremos el teclado.

Ahora con una nueva pantalla, procederemos a proporcionar al programa de todos los elementos que formarán parte del diseño; observe que hay tres columnas (fig. 4.3.1.b) **CMP TIPO VALOR** y el cursor esta bajo la primera. Esto significa que por cada elemento usted define tres características que describen dicho elemento.

En la columna **CMP** deberá digitar el nombre del componente que en adelante se desplegará en el esquemático sobre el elemento; este nombre no deberá exceder los tres caracteres.

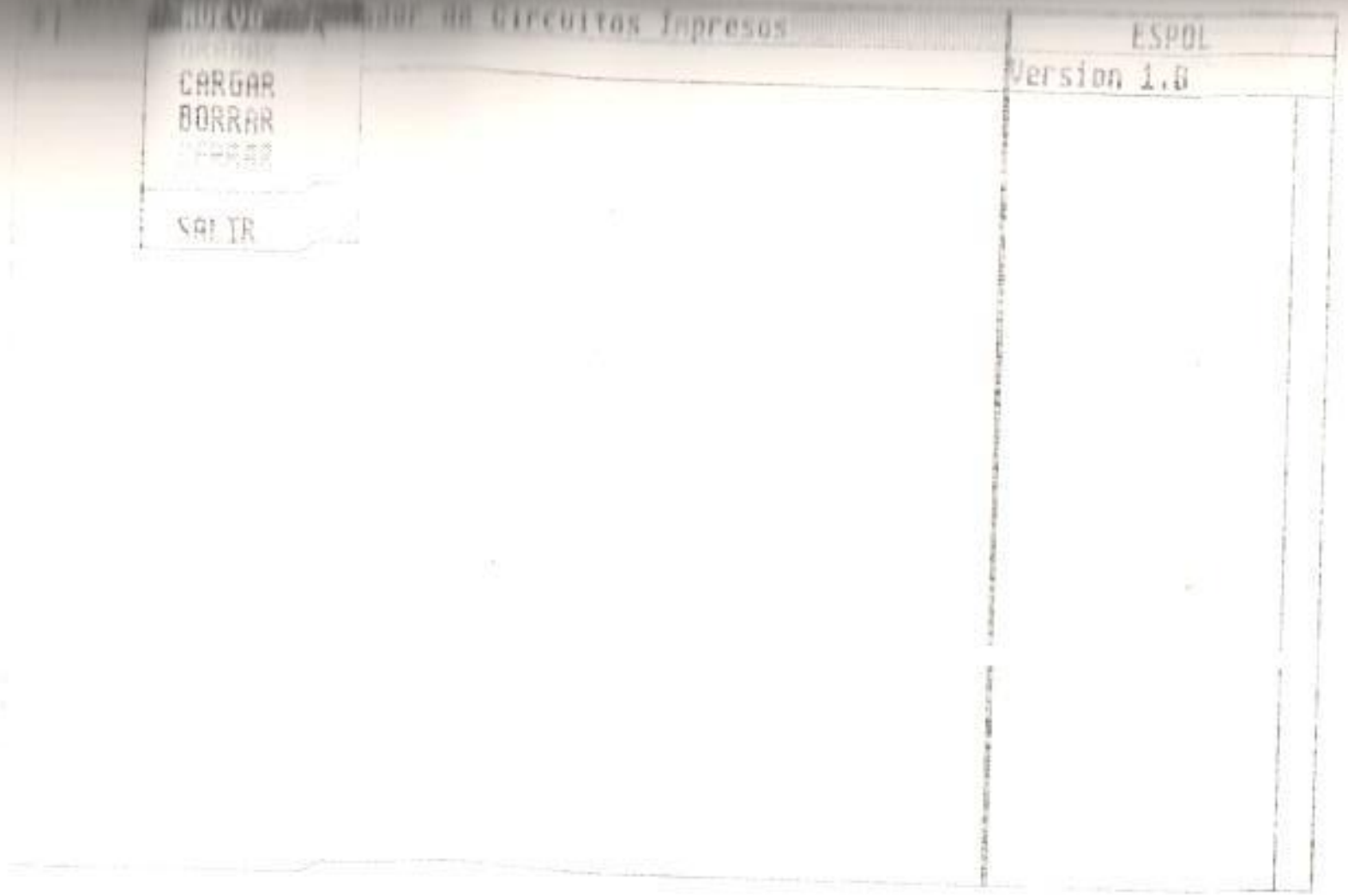


fig. 4.3.b Seleccionando comando Nuevo del menú Archivo.

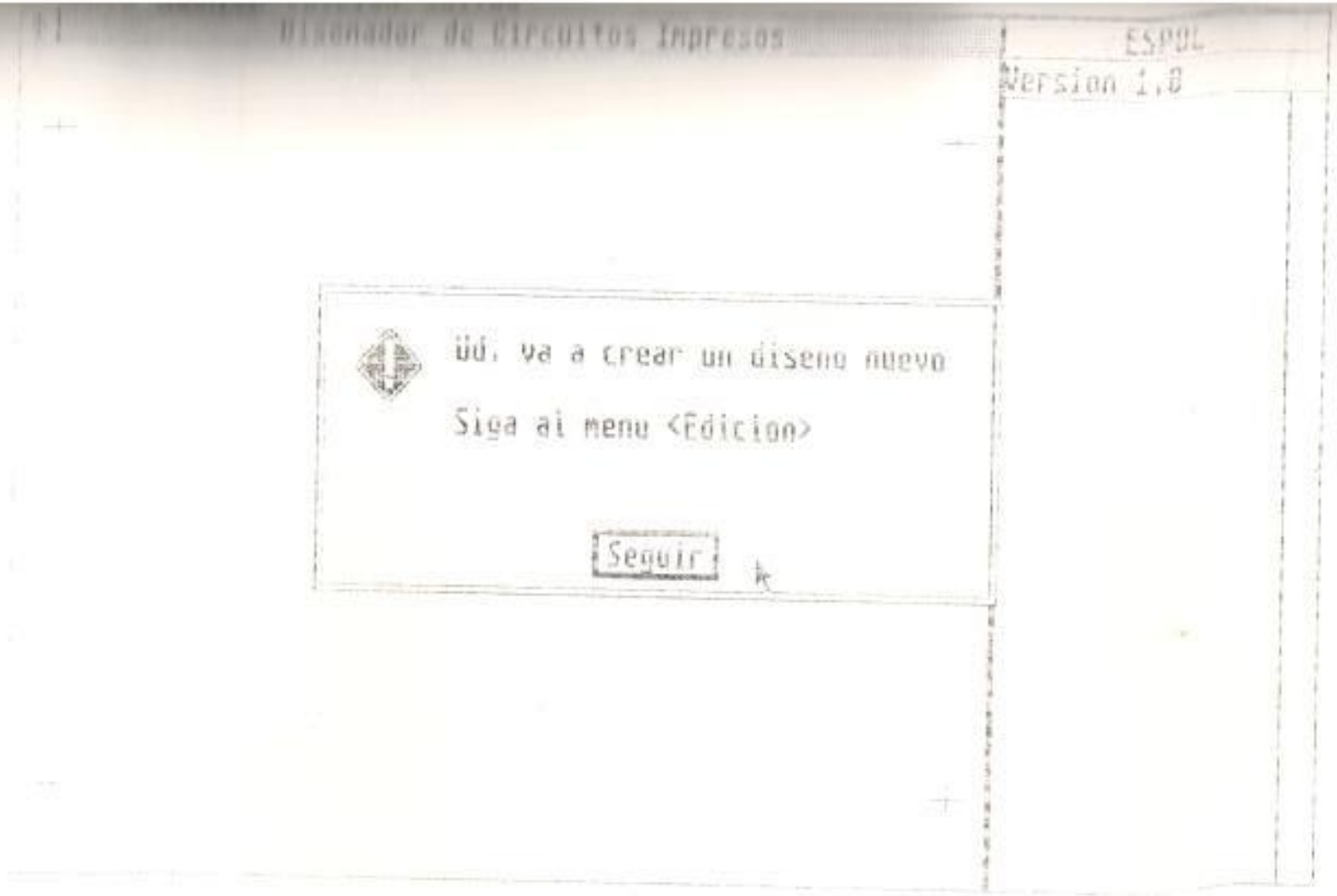


fig. 4.3.c Caja de información.

fig. 4.3.1.a Seleccionando comando Ingresar del menú Archivo.

R7	R/2W	100K	P5	PTO	POT.
R8	R/2W	1Mohm	P6	PTO	de
R9	R/2W	100K	P7	PTO	10K
R10	R/2W	1.2K	P8	PTO	INa
R11	R/2W	4.7K	P9	PTO	INb
R12	R/2W	68K	P10	PTO	S1a
R13	R/2W	30K	P11	PTO	S1b
D3	D5	IN4148	N+	PTO	MEDa
D4	D5	IN4148	N-	PTO	MEDb
D5	D5	IN4148	P14	PTO	S2a
D6	D5	IN4148	P15	PTO	S2b
C3	C8	10nF	fin		
C4	C8	100nF			
C5	C9	100nF			
IA	I8P	TL071			
I2	I8P	TL082			
P1	PTO	LED			
P2	PTO	+			
P3	PTO	LED			

Para Borrar digite <K> Para Finalizar digite <fin>

fig. 4.3.1.b Listado completo luego de ingresar.

En la columna TIPO en cambio digitará la clase de componente que desea ingresar; la siguiente lista proporcionará la libreria de todos los componentes que dispone el programa:

PTO	Punto
R/4W	Resistencia de 1/4W
R/2W	Resistencia de 1/2W
R1W	Resistencia de 1W
R2W	Resistencia de 2W
R5W	Resistencia de 5W
I8P	Integrado de 8 pines
I10P	Integrado de 10 pines
I14P	Integrado de 14 pines
I16P	Integrado de 16 pines
I24P	Integrado de 24 pines
D5	Diodo DO-7
D6	Diodo DO-27
D7	Diodo de 3W
D8	Diodo de 5W
TR1	Transistor TO-220
TR2	Transistor TO-92
TR3	Transistor TO-18
TR4	Transistor TO-3
C1	Capacitor Electrolitico
C2	Capacitor Electrolitico
C3	Capacitor Electrolitico

C4 Capacitor Electrolitico

C5 Capacitor Electrolitico

C6 Capacitor Electrolitico

C7 Capacitor Electrolitico

C8 Capacitor Cerámico

C9 Capacitor Cerámico

C10 Capacitor Cerámico

C11 Capacitor Cerámico

Cualquier otro tipo de componente que digite, será rechazado enseguida (Observe que los nombres están en mayúsculas).

La ultima columna VALOR es para un comentario, pero muy importante es digitarlo, omitirlo solo produce errores.

Cada dato será ingresado cuando presiona la tecla <Return> y el cursor se posiciona automáticamente para pedir el siguiente dato: la forma de editar los datos ingresados será tratado más adelante.

Una vez que terminó de ingresar los componentes bajo la columna CMP digite la palabra fin y así volverá a la pantalla de graficos.

4.3.2 COLOCACION DE COMPONENTES

El siguiente paso es colocar sobre la pantalla los componentes que ya ingresamos; para ello en el menú **Edición** seleccione el comando Colocar (fig. 4.3.2.a), entonces veremos que dentro de la ventana de gráficos se dibuja una rejilla simétrica de 50 por 50 puntos sobre el cual colocaremos todos los elementos y veremos en la ventana de texto la información de los elementos que se vayan colocando.

Ahora podemos proceder de la siguiente manera:

- Presione el botón izquierdo del raton sin soltarlo.
- Ahí verá una silueta que corresponde al primer componente ingresado (la colocacion es secuencial respecto al orden de ingreso).
- Sin soltar el botón arrastre la silueta a travez de la rejilla hasta el lugar que elija.
- Suelte el boton, y allí se dibujará el componente con el nombre del mismo encima, mientras que los datos correspondientes se verán en la ventana de texto.

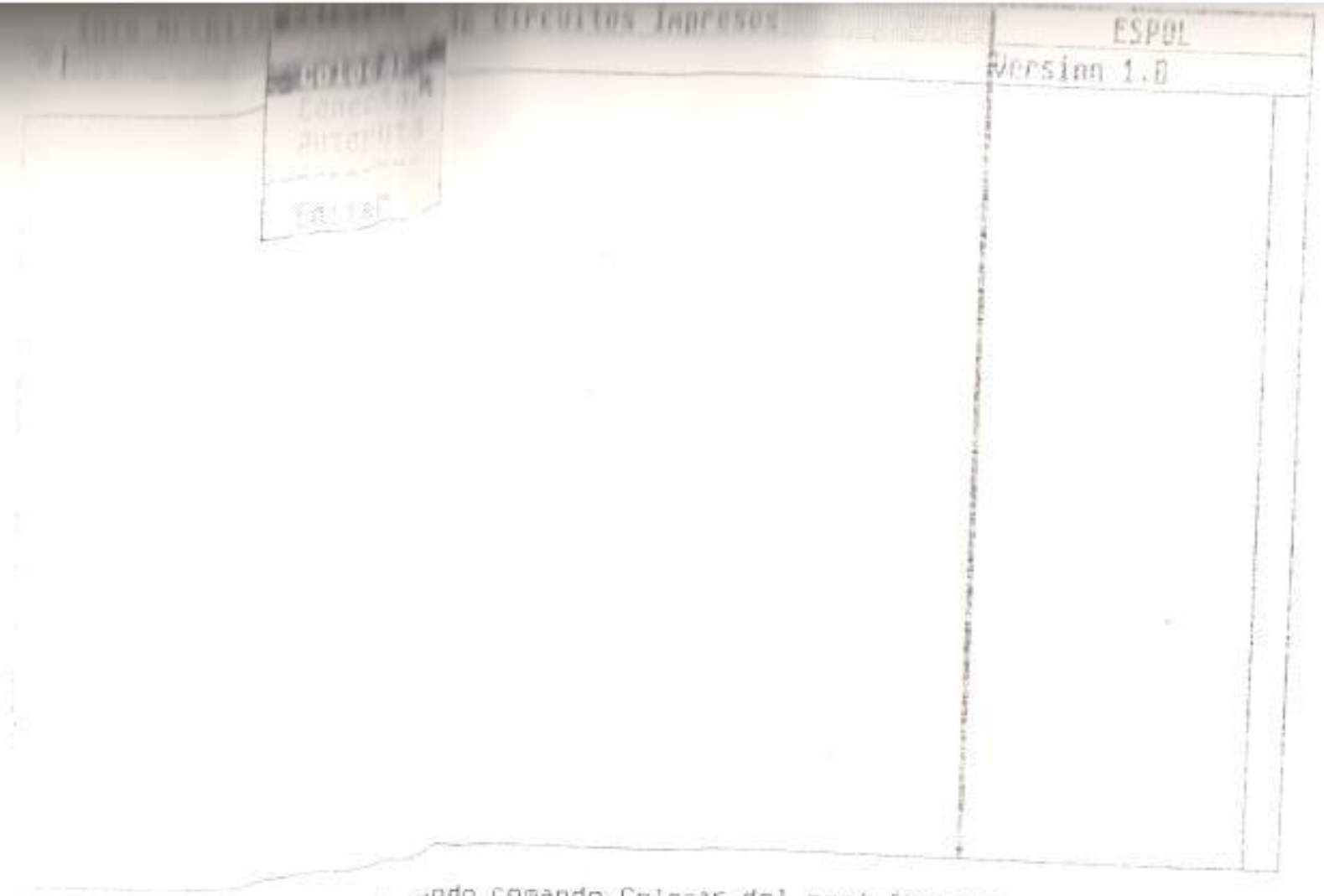


fig. 4.3.2.a Seleccionando comando Colocar del menú Archivo.

- Nuevamente vuelva a presionar el boton y verá la silueta del segundo elemento.

- El procedimiento es igual hasta terminar con todos los componentes ingresados (fig. 4.3.2.b).

- El programa se encarga de que nunca un componente salga de la rejilla, pero es su responsabilidad la ubicación y si es que cae encima de otro componente.

- En todo caso se puede editar la colocación y su explicación se detallara después.

4.3.3 CONECCION ENTRE TERMINALES

Esta es la segunda información que necesita el programa, para establecer cuales son los puntos que serán enlazados formando las pistas de conección.

En el menú **Edición** seleccione **Conectar** (fig. 4.3.3.a) la pantalla dispondrá la presentación de todos los componentes con sus respectivos terminales resaltados mientras que la ventana de texto presentará las conecciones que se realicen.

Para establecer las conecciones siga estos pasos:

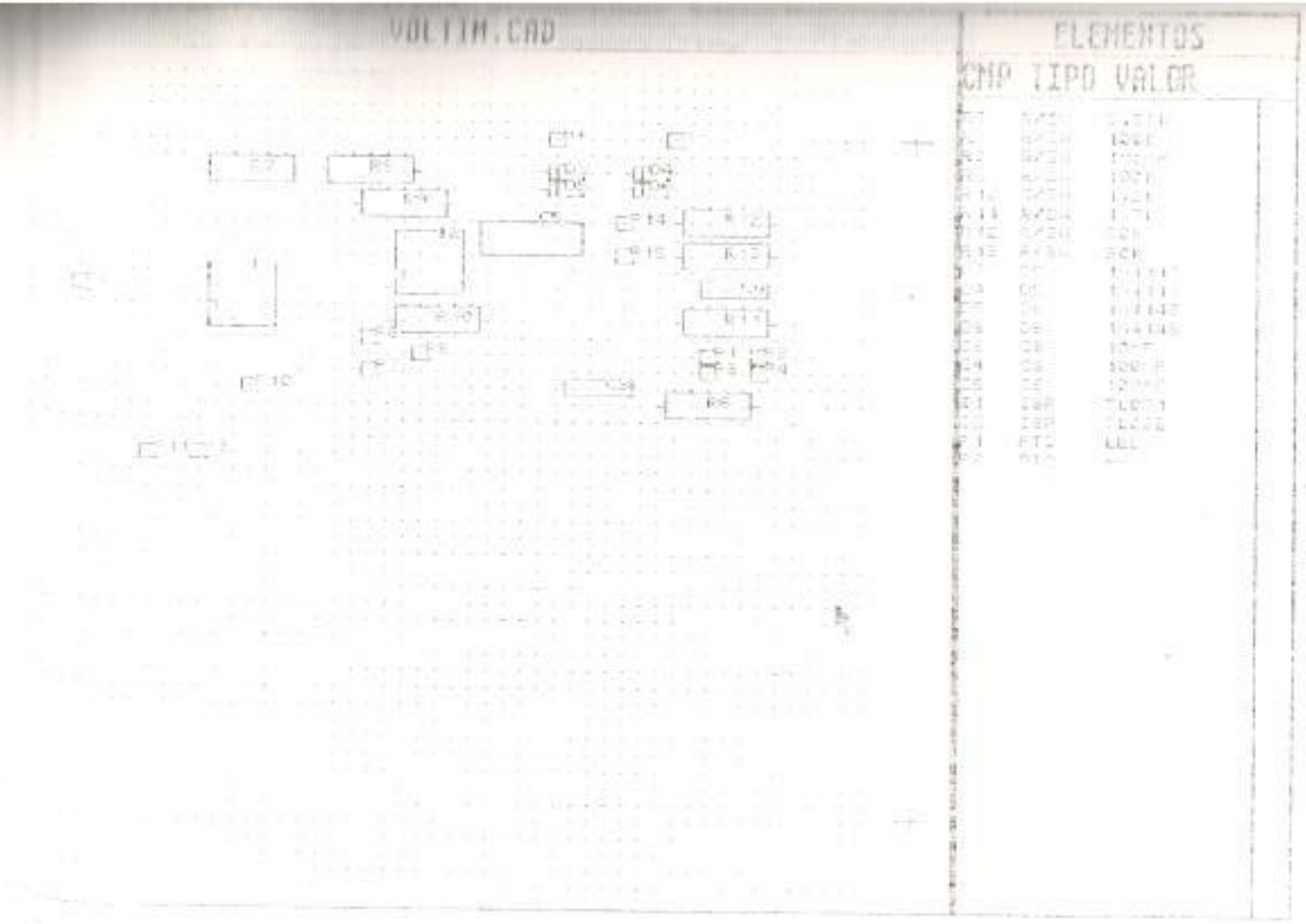


Fig. 4.3.2.b Presentación en pantalla luego de colocar los elementos.

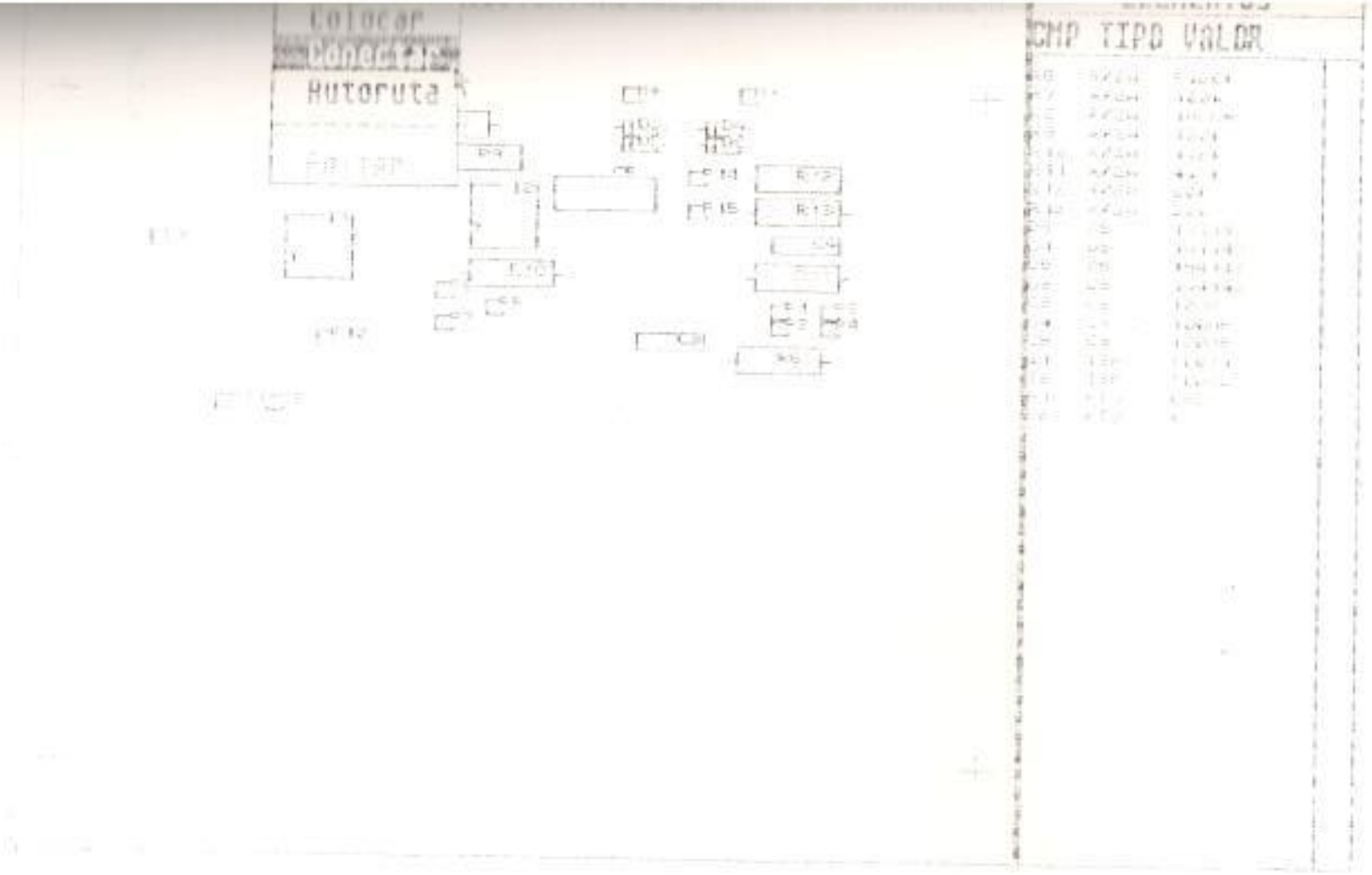


fig. 4.3.3.a Seleccionando comando Conectar del menú Edición.

- Dirija el apuntador del ratón hacia el terminal que formará parte de una conección.
- Presione el botón izquierdo del ratón una vez.
- En la ventana texto aparecerá el dato ingresado.
- Dirijase ahora hacia el segundo terminal.
- Presione el botón izquierdo del ratón.
- Y en la ventana texto se completará la pareja de la primera conección.
- De ahi en adelante establezca todas las conecciones que usted necesite (fig. 4.3.3.b).
- Para terminar lleve el apuntador del ratón hacia la palabra Fin que está en la parte superior izquierda de la ventana gráfica y presione el botón del ratón.
- La edición de conección se realizará después.

4.3.4 AUTORUTA

Con todos los datos ingresados, es decir los componentes, su ubicación y cuales son las

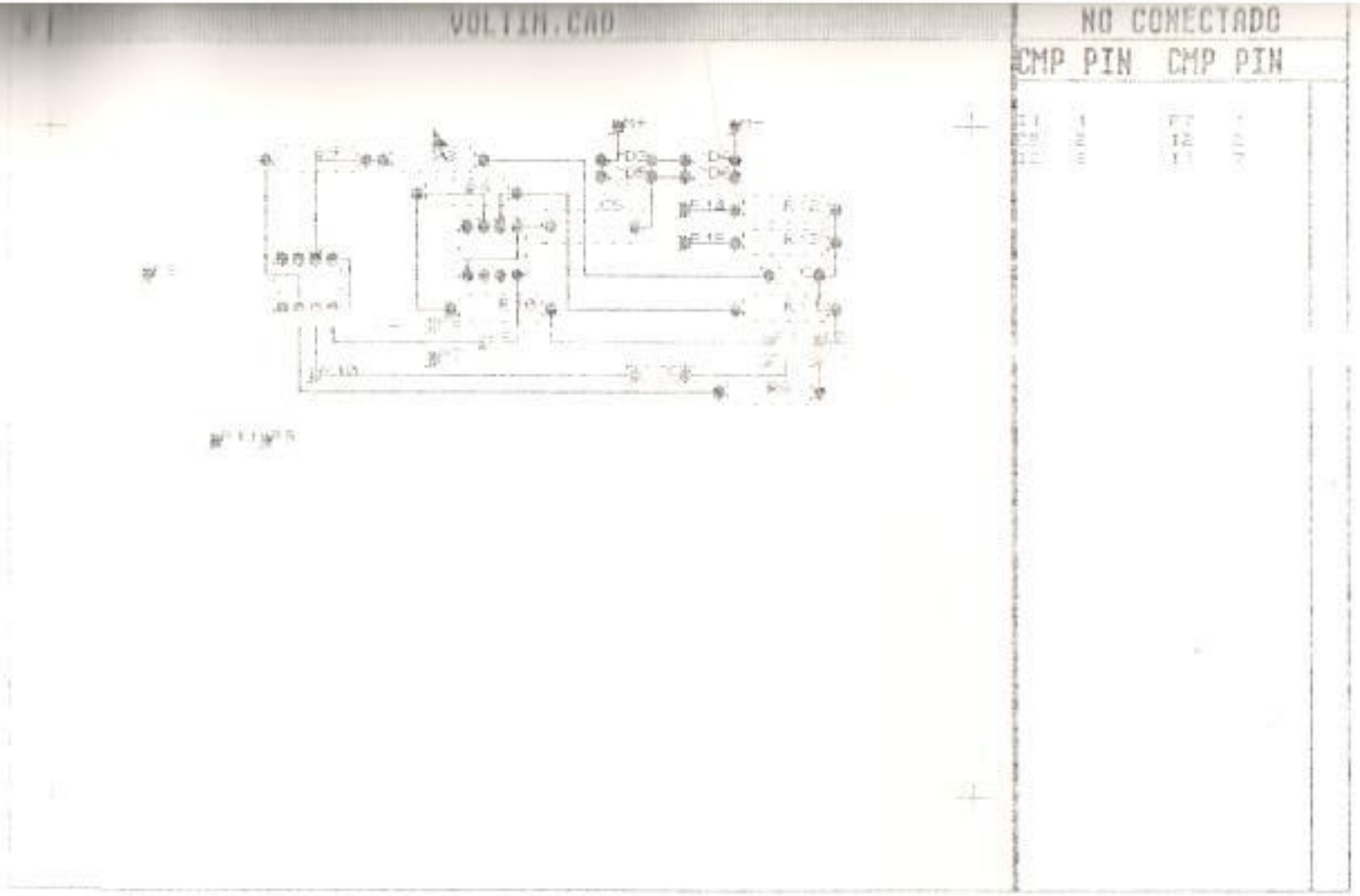


fig. 4.3.3.b Presentación una vez conectado todo.

conecciones a realizar el programa se encargará ahora de establecer las rutas automáticamerte.

Para elegir autoruta seleccione el comando del mismo nombre en el menú **Edición** (fig. 4.3.4.a) lo que verá será el resultado de las pistas enlazadas entre si y de fondo sombreado estará la imagen de los componentes.

Mientras que en la ventana texto se indicarán cuales son las conecciones que no pudieron enlazarse (fig. 4.3.4.b): sin embargo existe la posibilidad de conectar aquellos terminales si realizamos una mejor ubicación de los elementos o si la conección se realiza a otro punto equivalente.

Todo esto corresponde al capitulo de edición.

4.3.5 GRABACION DEL TRABAJO REALIZADO

Por supuesto la ventaja del diagrama obtenido es el de poder conservarlo para su futuro uso.

La forma normal de grabarlo es la siguiente:

- Seleccionar en el menú **Archivo** el comando **Grabar** (fig. 4.3.5.a)

- Aparecerá entonces una tarjeta de diálogo.

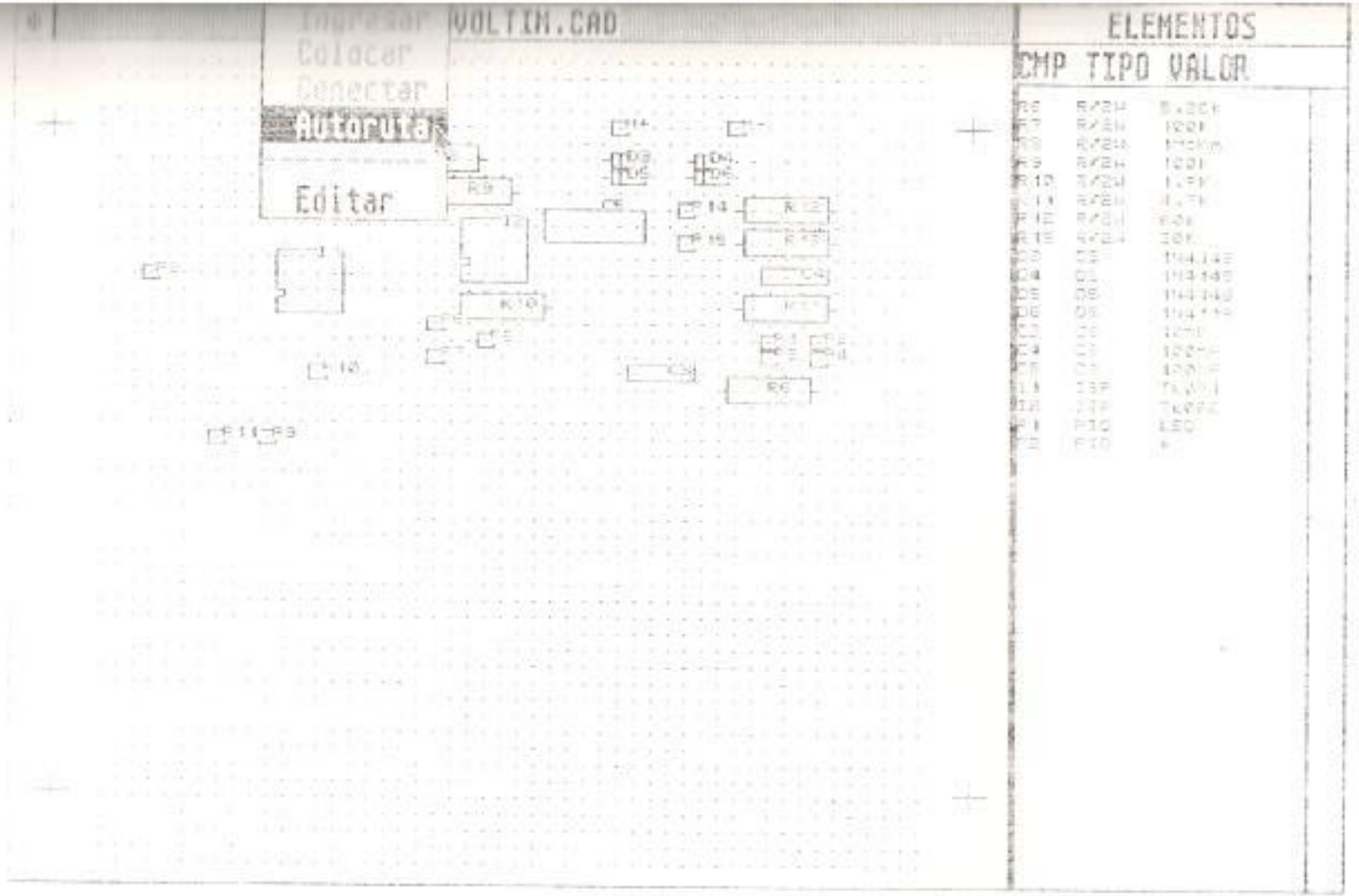


fig. 4.3.4.a Seleccionando comando Autoruta del menú Edición.

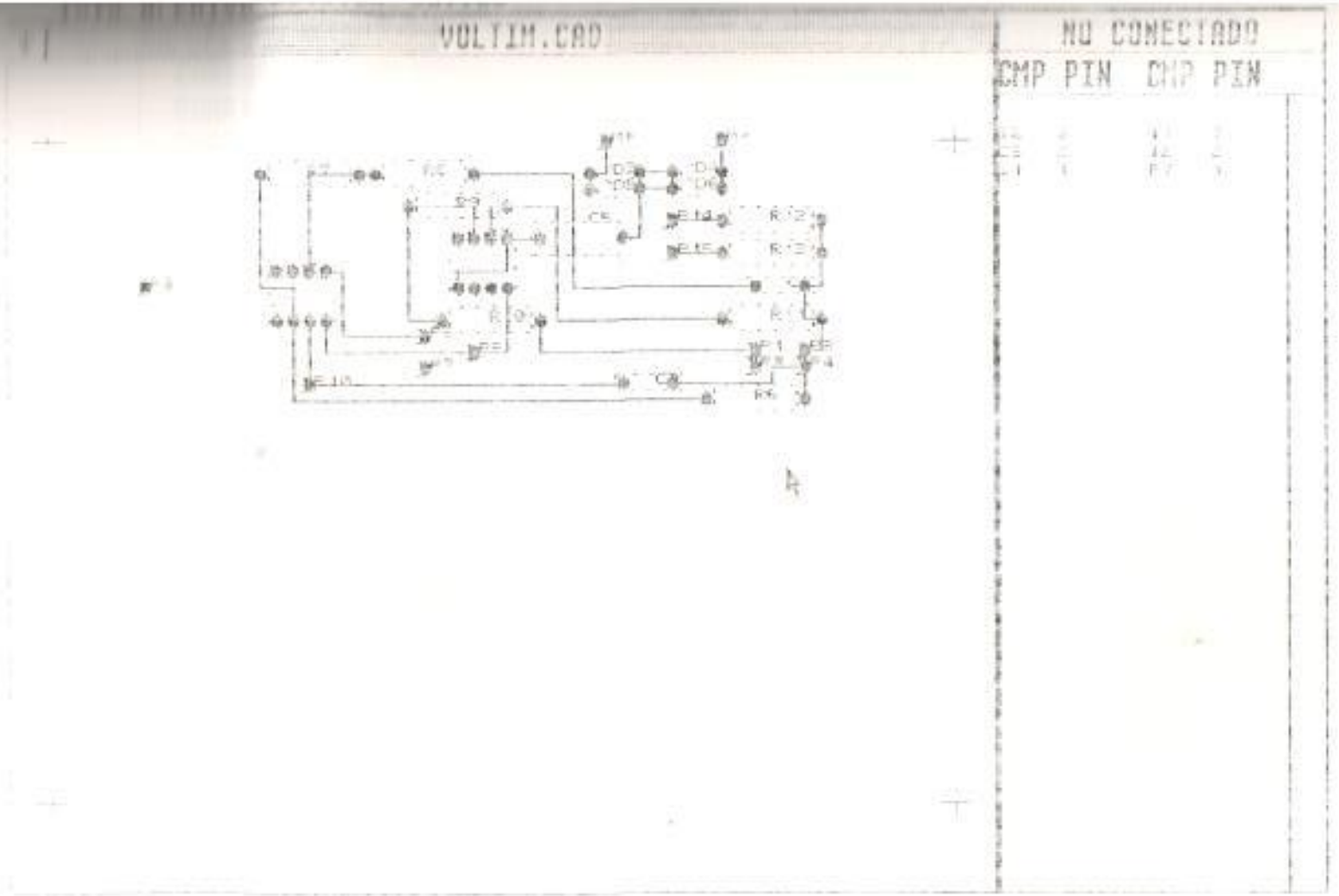


fig. 4.3.4.b Vista por pantalla una vez ejecutada la autoruta.

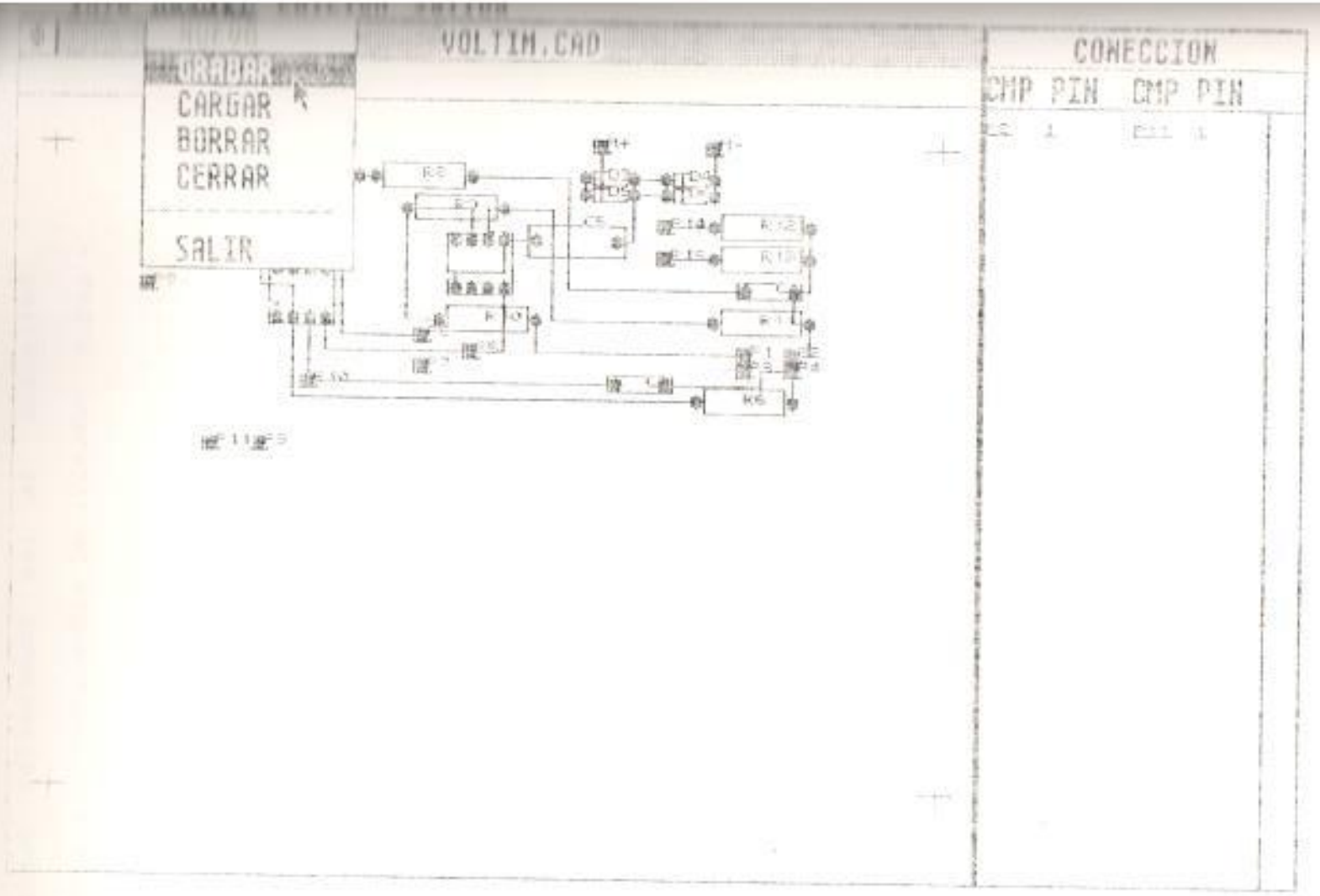


fig. 4.3.5.a Seleccionando comando Grabar del menú Archivo.

230

- Utilice el teclado para digitar el nombre que desee para nombrar a su diagrama, pero sin olvidar añadir la etiqueta .CAD (fig. 4.3.5.b).

- Una vez digitado presione (Return) y espere hasta que el foco de la unidad de disco se apague.

4.3.6 IMPRESION FINAL

La documentación impresa que usted como usuario dispondrá es la siguiente:

- Un listado de todos los componentes y todas las conecciones que forman parte del diseño.

  Para imprimirlo recuerde encender la impresora y tener el papel colocado.

  Con el ratón vaya al menu **Salida** y pulse **Listado** (fig. 4.3.6.a) y aguarde hasta que termine la impresión; si acaso le faltó papel, no apague la impresora, introduzca otra hoja y presione la tecla ON LINE del impresor para que imprima lo que falta. (fig 4.3.6.b).

- Podrá imprimir un esquemático que posee la ubicación de los componentes sobre la rejilla.

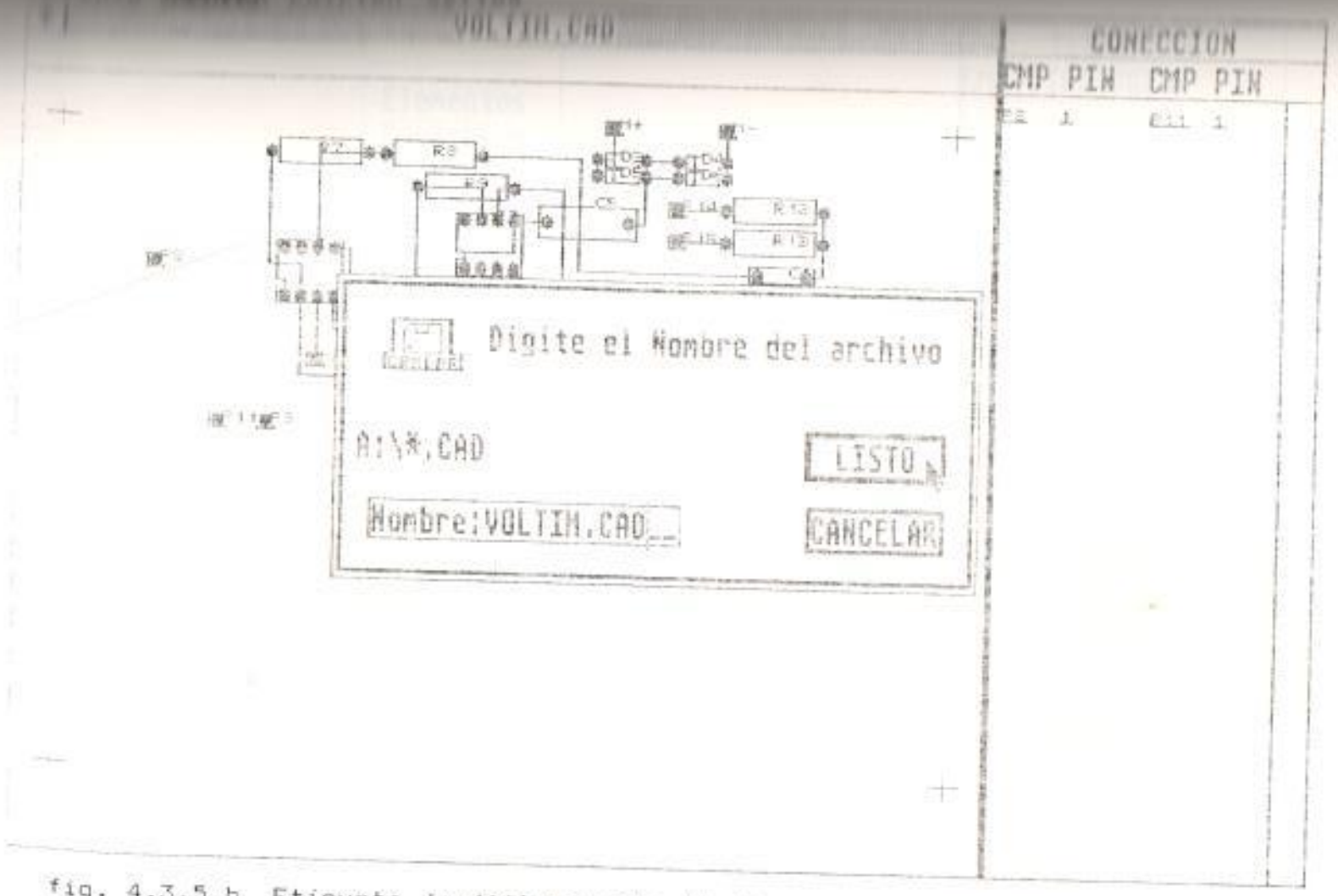


fig. 4.3.5.b Etiqueta de diálogo para digitar nombre a grabar.

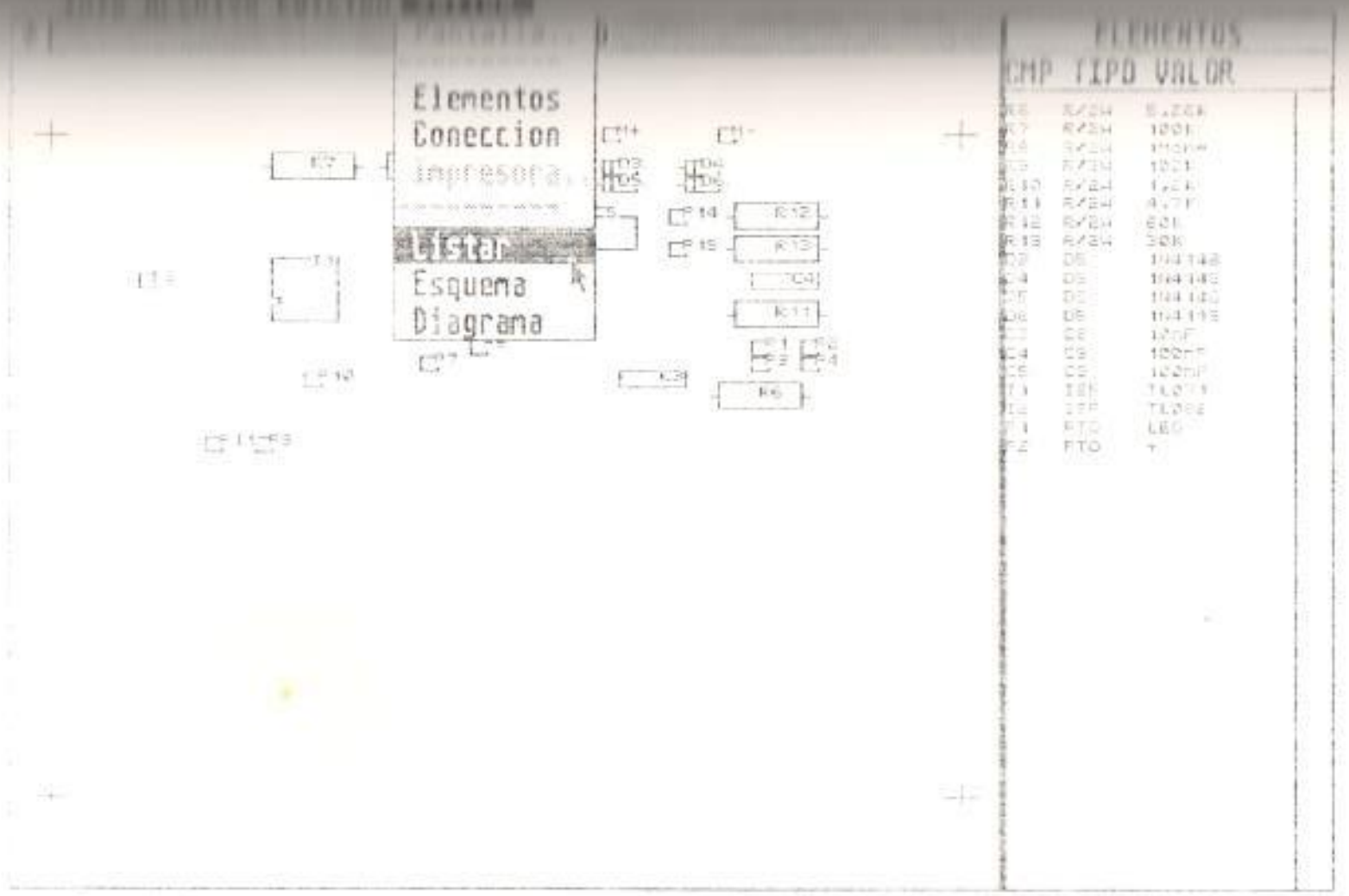


fig. 4.3.6.a Selecciona comando Listar del menú Salida.

Seleccione con el raton el comando **Esquema** del menú **Salida** (fig. 4.3.6.c); una caja de diálogo aparecerá en la pantalla pidiéndole la confirmacion o la cancelacion de la orden. Presione (Return) y el esquemático se imprimirá.

Por ser un documento referencial, unicamente se imprimirá la pantalla. (fig. 4.3.6.d).

- Finalmente el resultado más importante del programa es imprimir el diagrama de las rutas.

En el menú **Salida** seleccione **Diagrama** (fig. 4.3.6.e) y confirme la impresión, en ese momento obtendrá su documento final (fig 4.3.6.f).

Recuerde que por la calidad de impresión, es importante que la cinta de tinta de la impresora sea nueva.

Por supuesto puede realizar las copias que desee y por si acaso su impresora no estuviera lista, el programa lo detecta y se lo advierte.

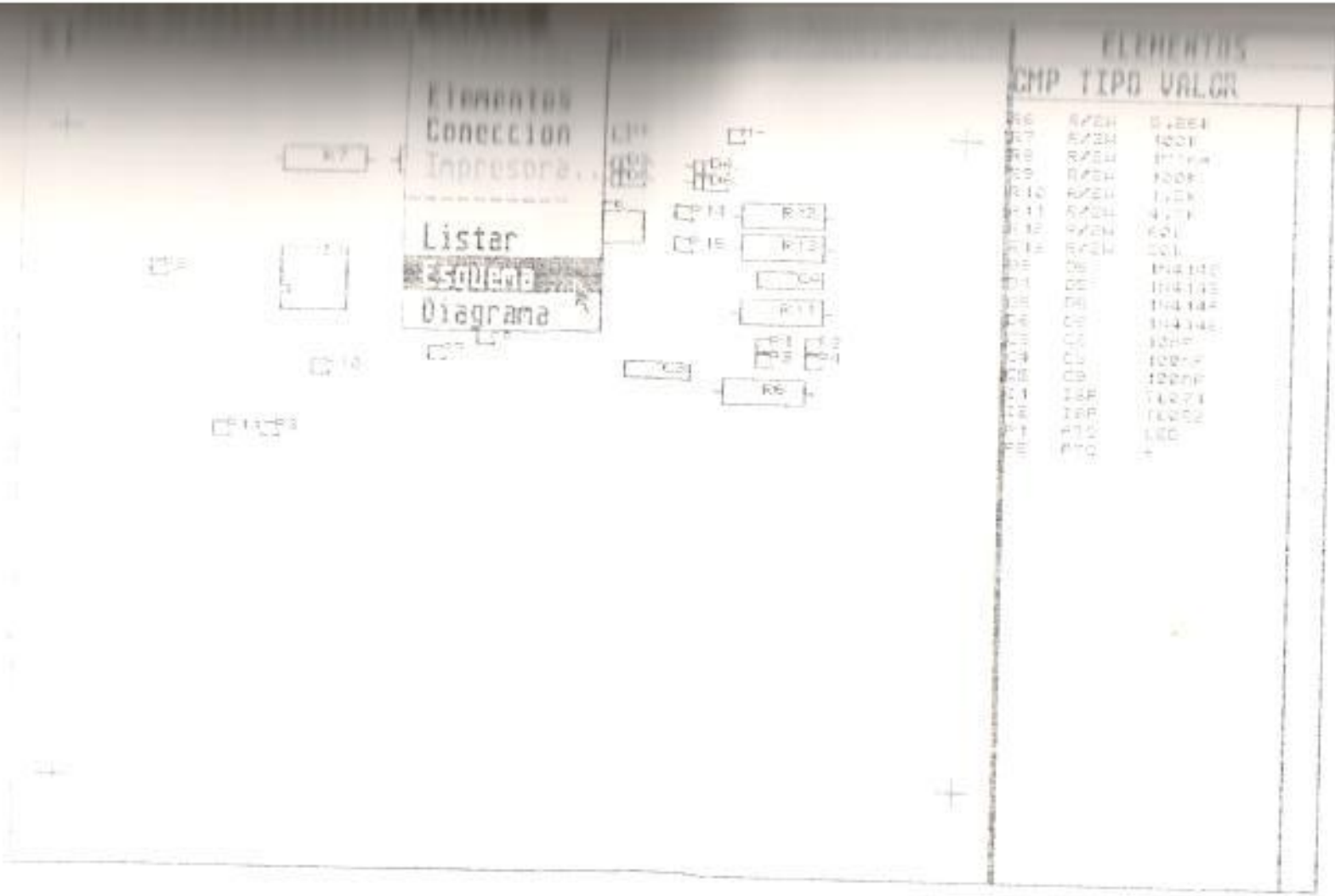


fig. 4.3.6.c Selecciona comando Esquema del menú Salida.

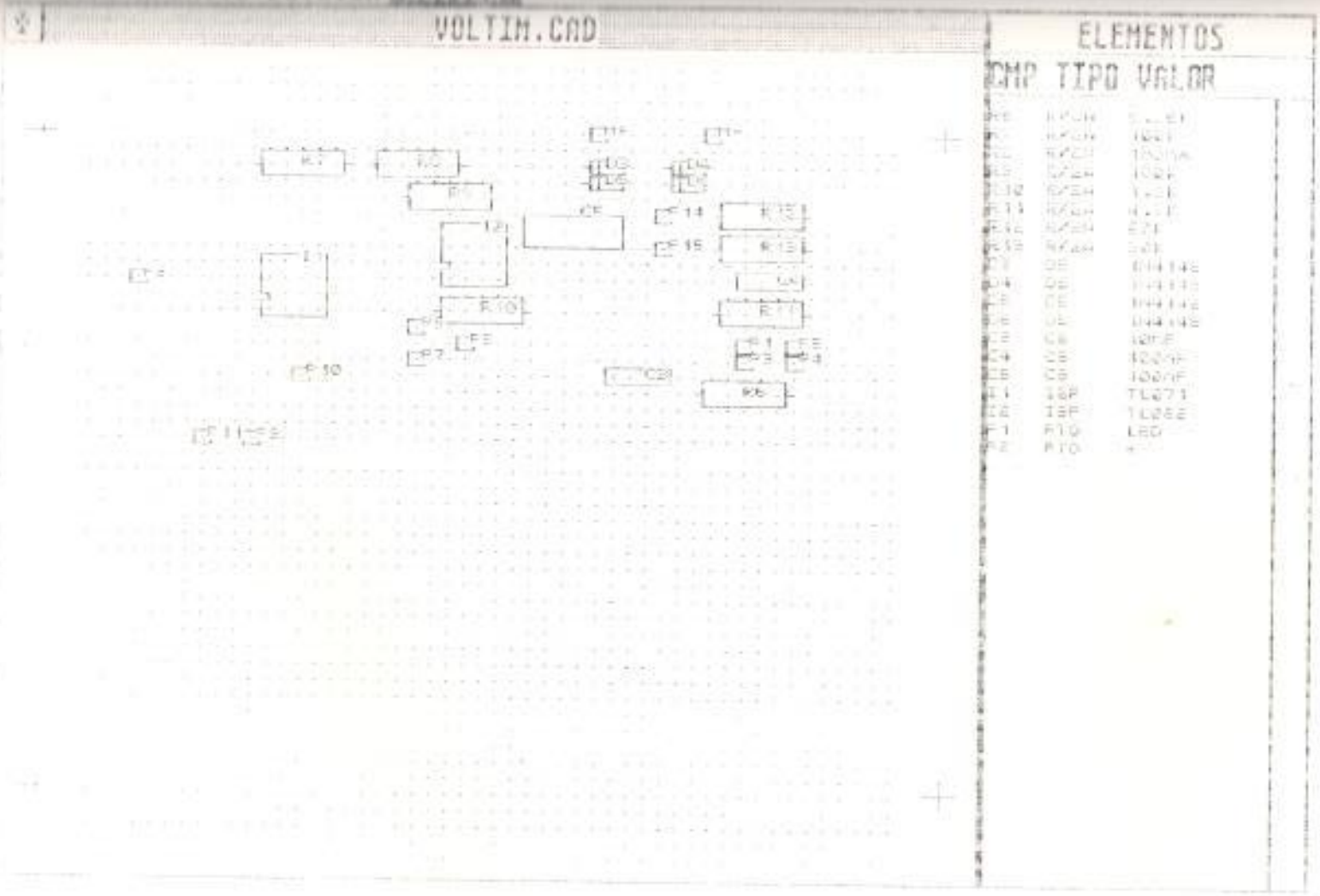


fig. 4.3.6.d Representación por impresora del esquemático.

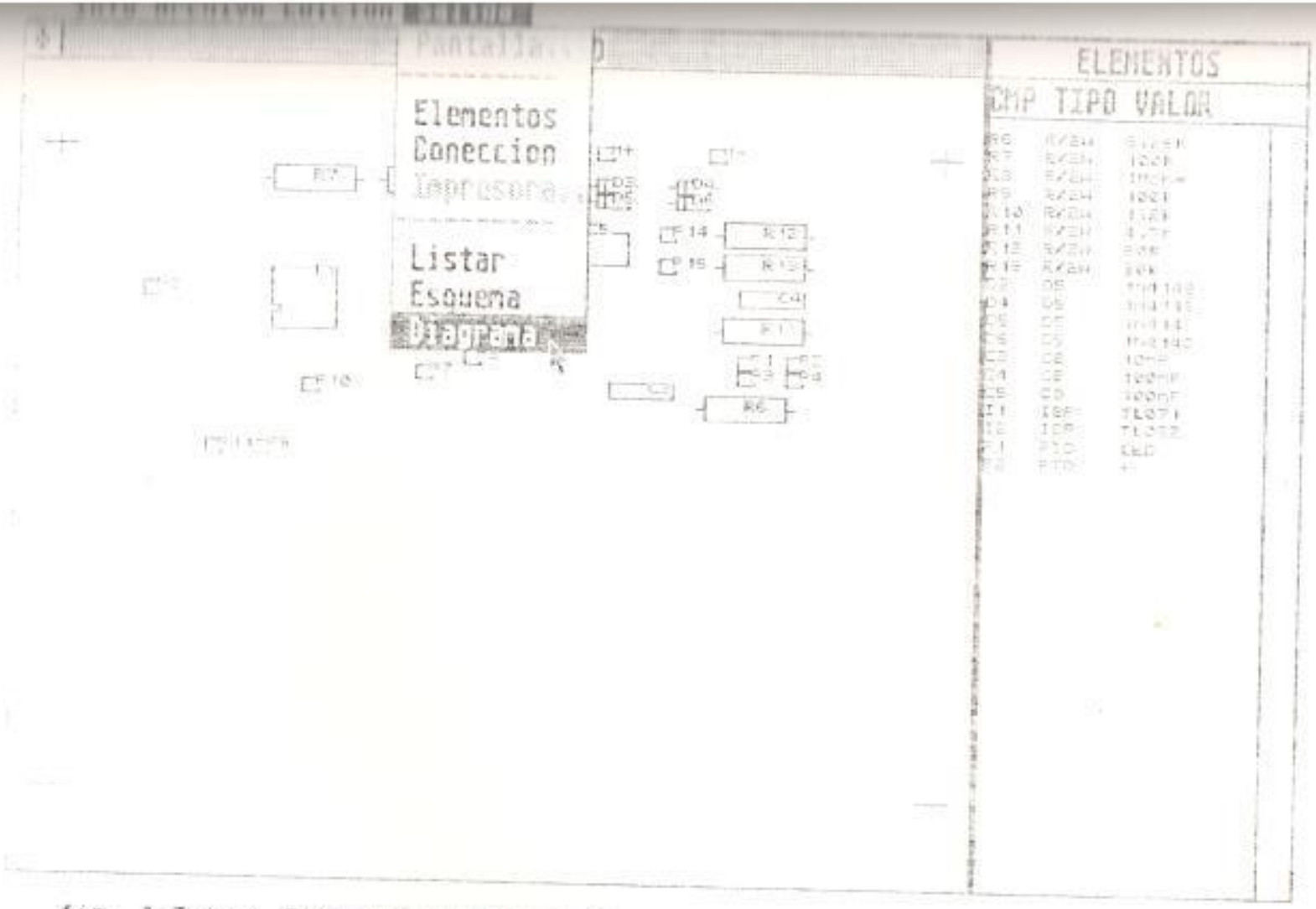


fig. 4.3.6.e Selecciona comando Diagrama del menú Salida.

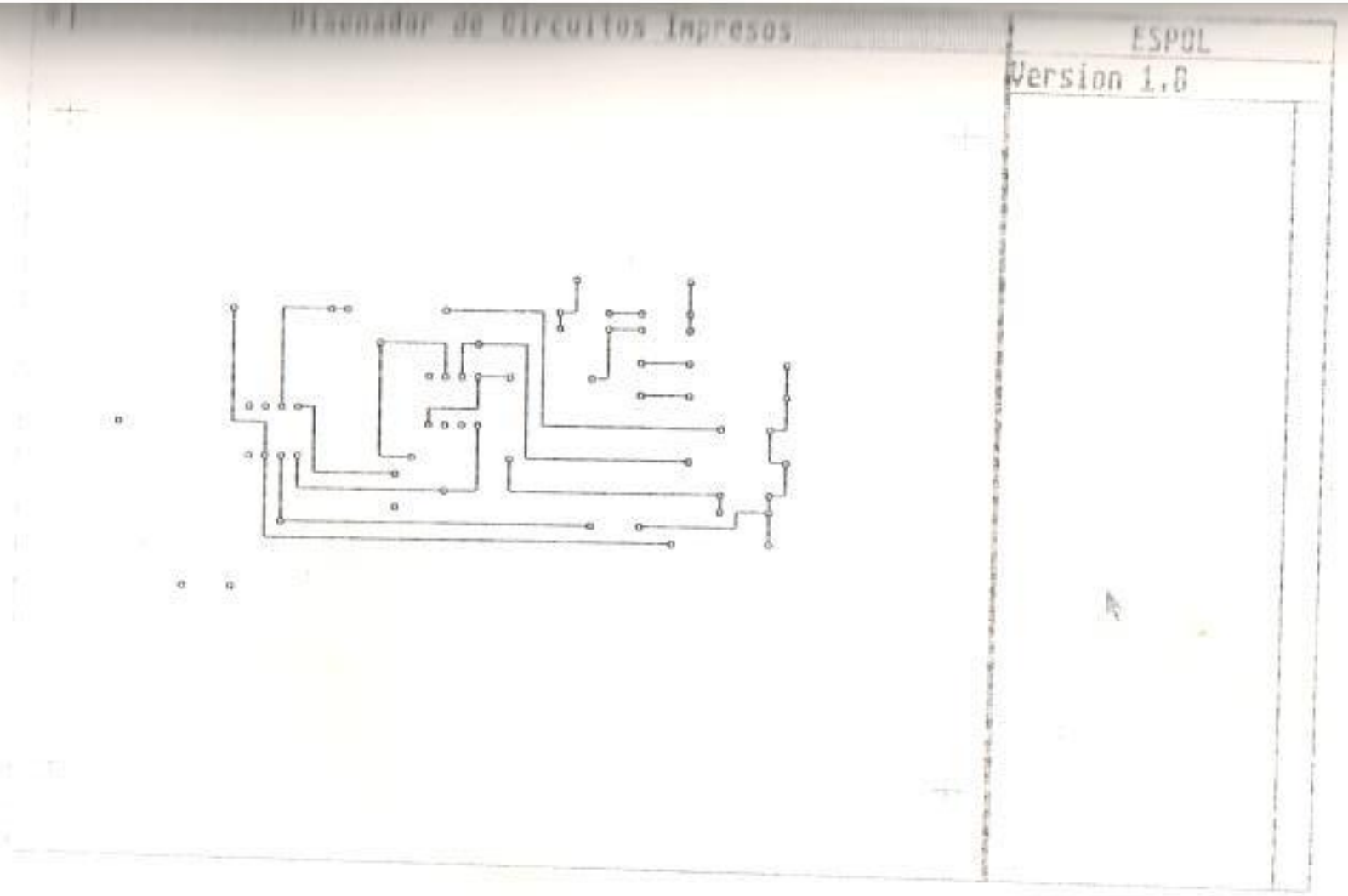


fig. 4.3.6.f Representación final de las pistas por impresora.

4.4 EDICION DE UN DISEÑO YA CREADO

Una de las virtudes de este programa es su edición, es decir que una vez ingresado todos los datos y visto el resultado de la autoruta, es posible realizar los cambios necesarios para optimizar el diagrama. Para entrar en el modo edición solo hay que seleccionar el comando **Editar** del menú **Edición** (fig. 4.4.a) y quedarán habilitados todos los comandos para editar ingreso de datos, colocación de componentes, las conecciones entre otros. Para describir estos comandos utilizaremos el mismo diagrama-ejemplo ya realizado.

4.4.1 EDICION DE LA TABLA DE ELEMENTOS

Para editar el ingreso de componentes solo hay que seleccionar en el menú **Edición** el comando **Ingresar** para pasar a una pantalla que nos *muestra todos los datos de los elementos.*

Aqui podemos ingresar más componentes si es necesario, tal como se explico en la sección 4.3.1; ó podemos borrar aquellos que no deseamos, digitando un asterisco (*) bajo la columna **CMP** y la tecla (Return) (fig. 4.4.1.a); de manera que moviendo el cursor elijamos la linea oscura que será borrada presionando (Return ; si desea borrar otro

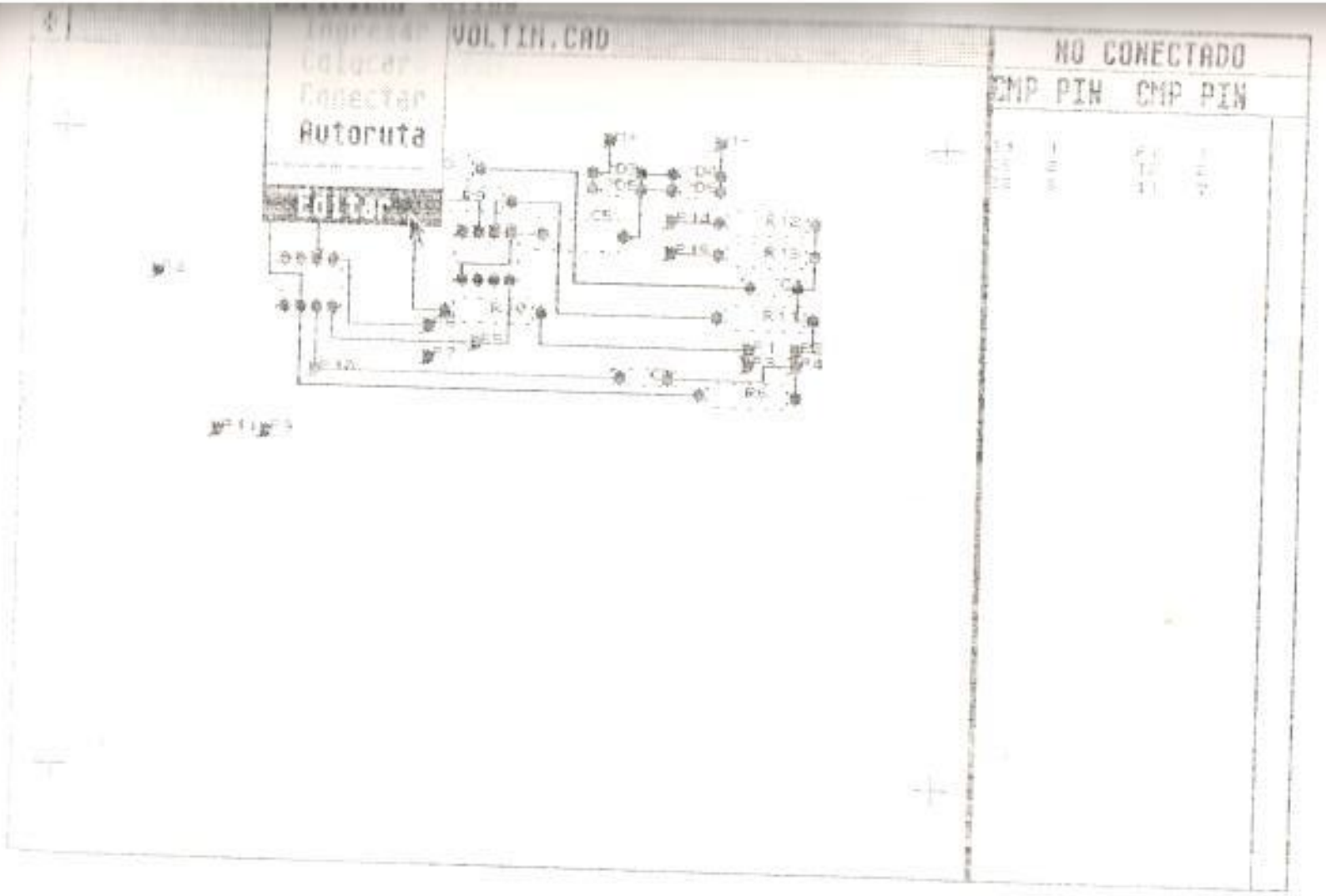


fig. 4.4.a Selecciona comando Editar del menú Edición.

(Presione <return> para cada dato ingresado)

R6	R/2W	5.2GK	P4	PTO	-
R7	R/2W	100K	P5	PTO	POT.
R8	R/2W	4.6GK	P6	PTO	de
R9	R/2W	100K	P7	PTO	10K
R10	R/2W	1.2K	P8	PTO	IHa
R11	R/2W	4.7K	P9	PTO	IHb
R12	R/2W	66K	P10	PTO	S1a
R13	R/2W	36K	P11	PTO	S1b
D3	D5	1N4148	M+	PTO	MEDa
D4	D5	1N4148	M-	PTO	MEDb
D5	D5	1N4148	P14	PTO	S2a
D6	D5	1N4148	P15	PTO	S2b
C3	C8	10nF			
C4	C8	100nF			
C5	C9	100nF			
I1	I8P	TL071			
I2	I8P	TL082			
P1	PTO	LED			
P2	PTO	+			
P3	PTO	LED			

Para borrar elija la linea y presione <Return>

fig. 4.4.1.a Elección de un componente a borrar.

elemento, nuevamente digite (*) y así sucesivamente por cada elemento que quiera eliminar.

De ahí en adelante todo componente borrado no aparecerá en el esquemático y tampoco las conecciones asociadas a sus terminales.

Para terminar la edición de ingreso digite fin y presione <Return>.

4.4.2 EDICION DE LA POSICION DE LOS COMPONENTES

Para editar la posición de los componentes, seleccione simplemente el comando **Colocar** del menú **Edición**. Verá entonces en la ventana de gráficos una rejilla y sobre esta los componentes, mientras que en la ventana texto vemos los datos de los elementos y el primero de ellos subrayado (fig. 4.4.2.a). Con las teclas <↑> y <↓> puede escojer cual es el elemento que va a cambiar de lugar si presiona <Return>, entonces con el ratón tal como se explicó en la sección 4.3.2 puede ubicar el elemento en otra posición; si no estuvo conforme vuelva a presionar <Return> y podrá mover el mismo componente nuevamente.

Como verá, en la ventana texto aparecen hasta 20 elementos, pero si su diagrama posee más

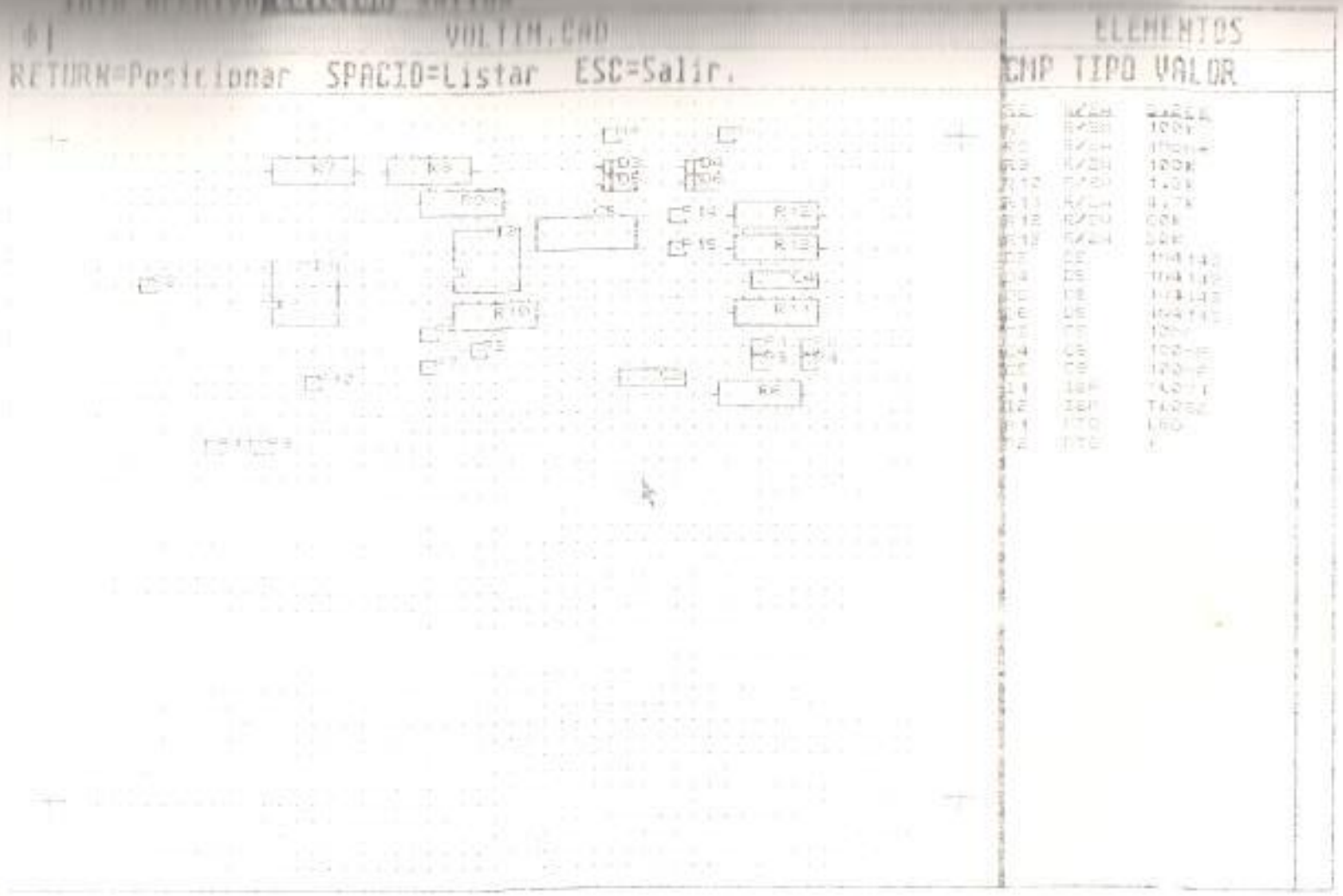


fig. 4.4.2.a Colocación en el modo edición.

que tal cantidad, presione la barra <Espacio> y observará un nuevo grupo de elementos. Puede usar <Espacio> las veces que desee hasta encontrar el componente a editar.

Otra caracteristica de esta edición es que puede cambiar el nombre del elemento y/o su comentario; lo que tiene que hacer es elegir con <↑> <↓> el componente y presionar la tecla <F1>, una caja de dialogos aparecerá (fig. 4.4.2.b) y con el teclado editará los datos; use <Return> para cerrar la caja, e inmediatamente verá los cambios en las ventanas.

Para salir de esta edición de Colocación presione <ESC>.

4.4.3 EDICION DE LAS CONECCIONES

En esta parte del programa usted tiene las facilidades de ingresar o eliminar varias conecciones o simplemente modificarlas.

Para ello seleccione el comando **Conectar** en el menu **Edición** y se displayará en la ventana gráfica los componentes con sus respectivas conecciones, mientras que en la ventana texto vera la descripción escrita de las mismas y la

fig. 4.4.2.b Información y edición de los datos de los elementos.

primera de ellas subrayada (fig. 4.4.3.a). Con las teclas <↑> <↓> elija una de las conecciones y presionando <Return> borrara aquella conección, el resultado aparecera en la pantalla.

Ahora, si lo que desea es ingresar nuevas conecciones, use la tecla <- y pasara a trabajar en la ventana gráfica. Realice la cantidad de conecciones que desee tal como se explico en la seccion 4.3.3 y cuando termine presione el botón del ratón sobre la palabra **fin**.

Para salir de la edición de conección presione la tecla <ESC>.

4.5 APUNTES ACERCA DE LA OPERACION DEL PROGRAMA

Todo lo que aqui vamos a tratar son aquellos comandos con funciones "extras", que procuran dar ayuda al usuario en el desempeño del programa.

Asi tenemos comandos como cargar y borrar que son utilitarios del sistema operativo de gran ayuda ya que sin necesidad de salir del programa podemos acceder a ellos.

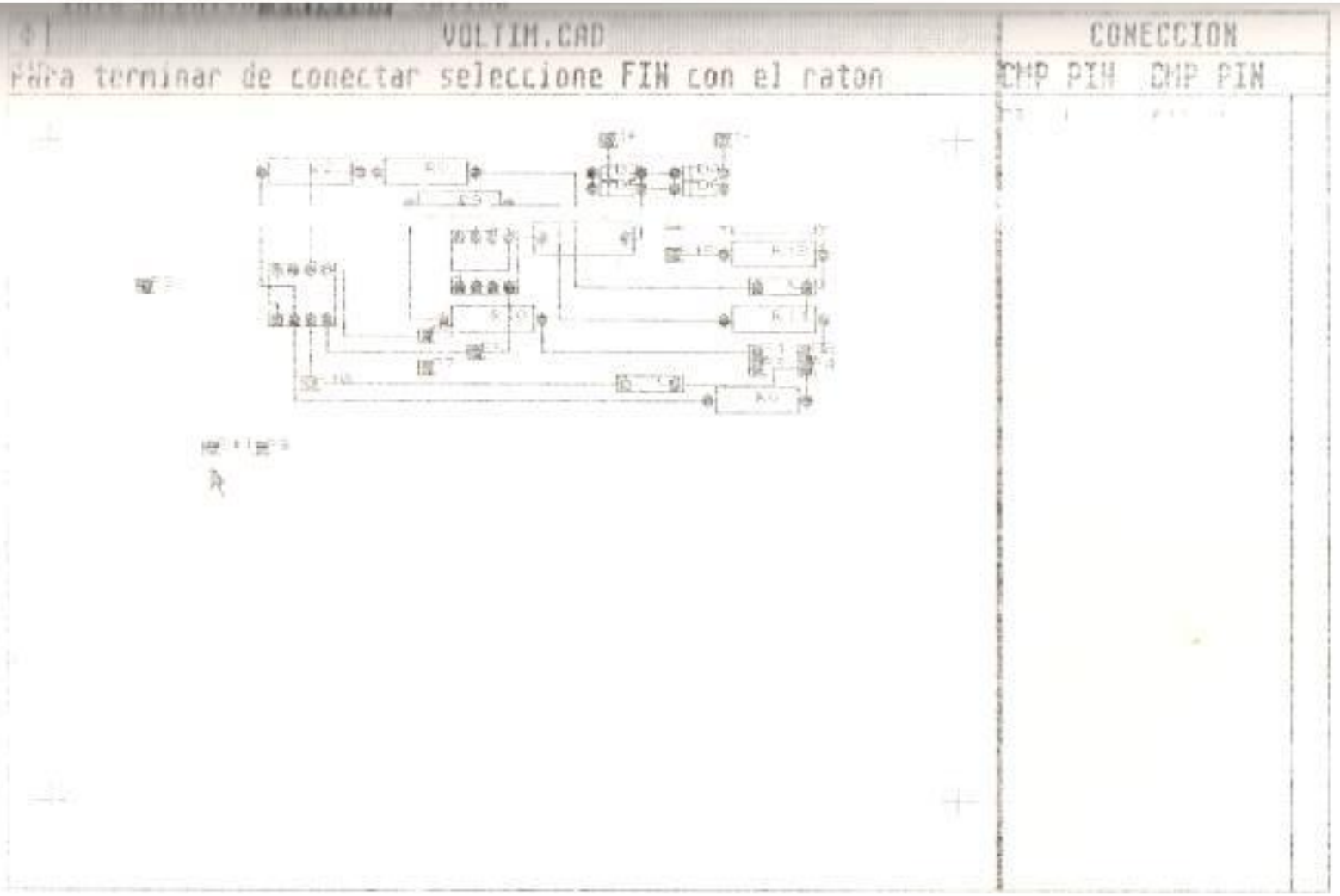


fig. 4.4.3.a Conección en el modo edición.

4.5.1 CARGAR UN ARCHIVO

Mediante el comando del mismo nombre en el menu Archivo (fig. 4.5.1.a) podemos importar un diseño ya creado desde el disco y trabajar con el mismo para editarlo o simplemente para imprimirlo.

Una pequeña ventana aparecerá con el contenido de todos los archivos de etiqueta .cad (fig. 4.5.1.b) entre los cuales podemos elegir una de ellas con el boton del raton y presionando <Return>.

El diagrama y sus datos se visualizan en las correspondientes ventanas y entrara automaticamente al modo edición.

Si acaso no desea cargar ningun archivo, seleccione la caja [cancel] de la ventana de directorio.

4.5.2 BORRAR ARCHIVOS

Con el comando **Borrar** del menú **Archivo**, (fig.4.5.2.a) puede usted elegir un archivo que desee eliminar del disco, para ello seleccione el archivo con el raton posicionándose encima y presionando el boton del raton una vez, y luego la tecla <Return>

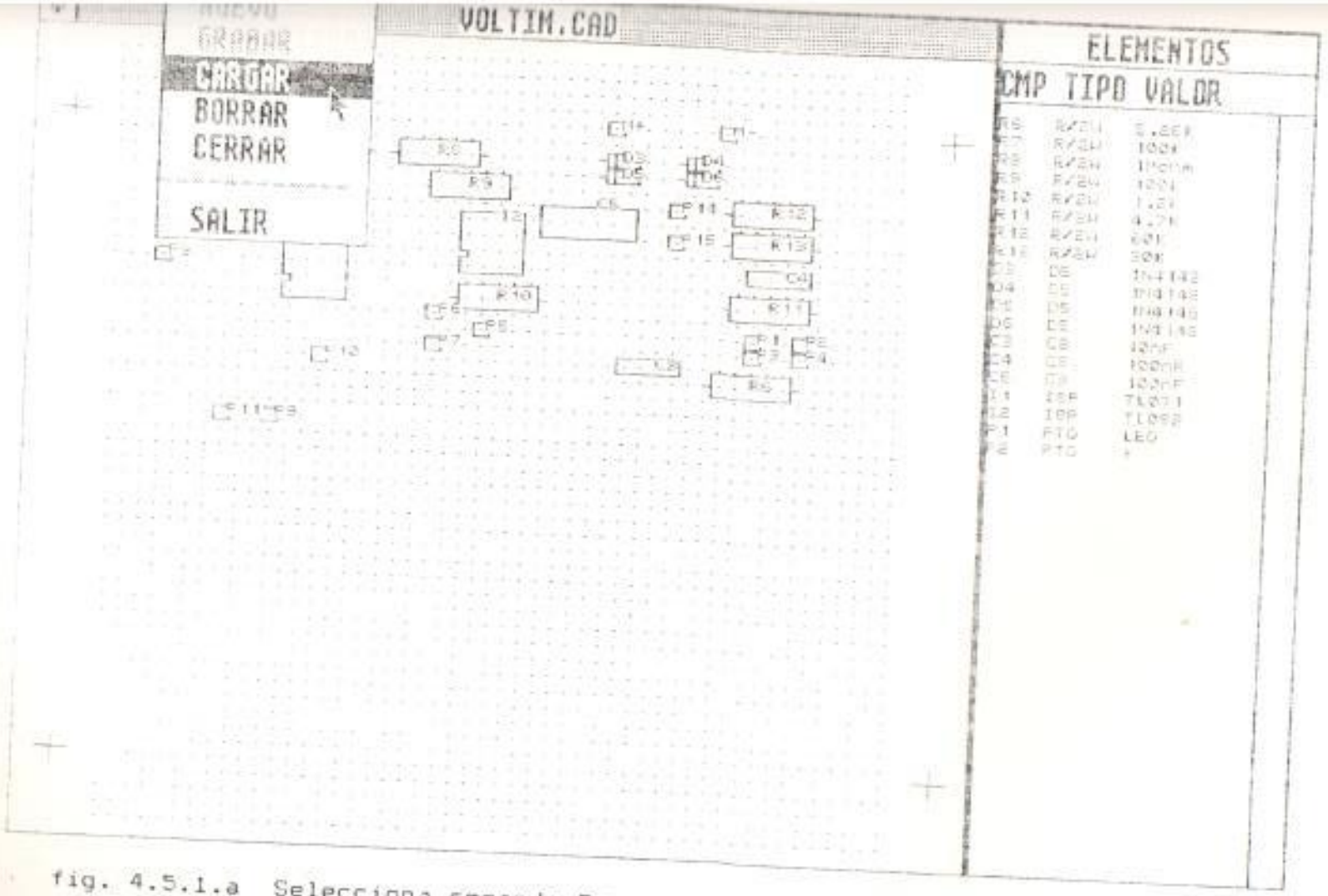


fig. 4.5.1.a Selecciona comando Cargar del menú Archivo.

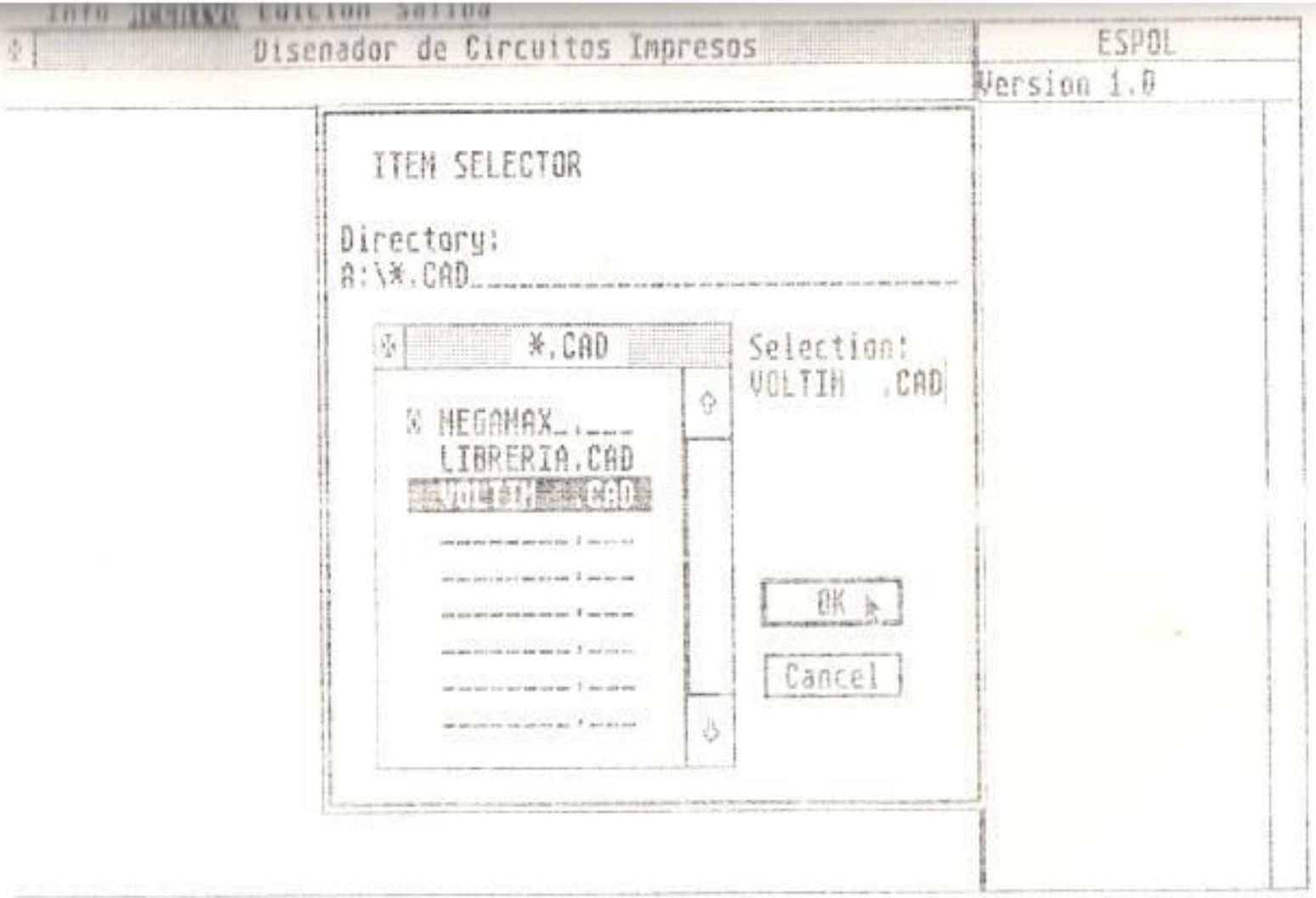


fig. 4.5.1.b Caja de diálogo que contiene los archivos .CAD

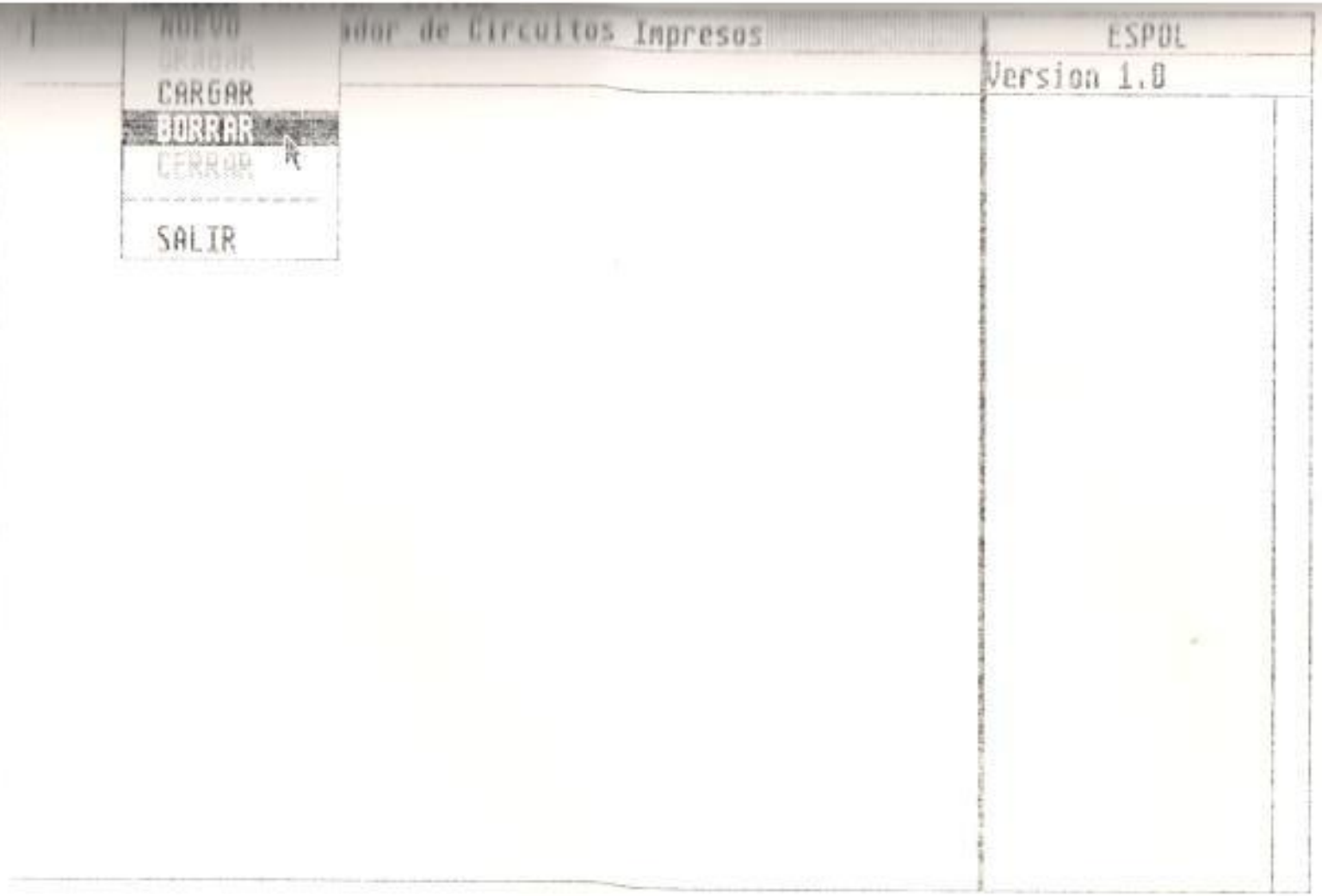


fig. 4.5.2.a Selecciona comando Borrar del menú Archivo.

si acaso no desea borrar nada seleccione la caja [Cancel] con el ratón.

4.5.3 INICIALIZAR EL PROGRAMA

Esta opción, como la anterior es de mucho cuidado pues su función es la de borrar todo el trabajo realizado y preparar el programa para comenzar nuevamente.

Esta opción se realiza con el comando **Cerrar** del menu **Archivo** (fig. 4.5.3.a) entonces una caja de diálogos le pedirá la confirmación o no de dicha orden.

4.5.4 CONSULTAS

Estos comandos consulta son muy sencillos de usarlos y prácticos a la vez, vamos entonces a describirlos:

- El comando **Componente** del archivo **Salida** (fig. 4.5.4.a) nos lista en la ventana texto un grupo de 20 elementos que forman parte del diseño. Si acaso existiesen mas, solo vuelva a llamar al comando y listará otro grupo de elementos.

- El comando **Conección** del archivo **Salida** (fig. 4.5.4.b) realiza en cambio una lista

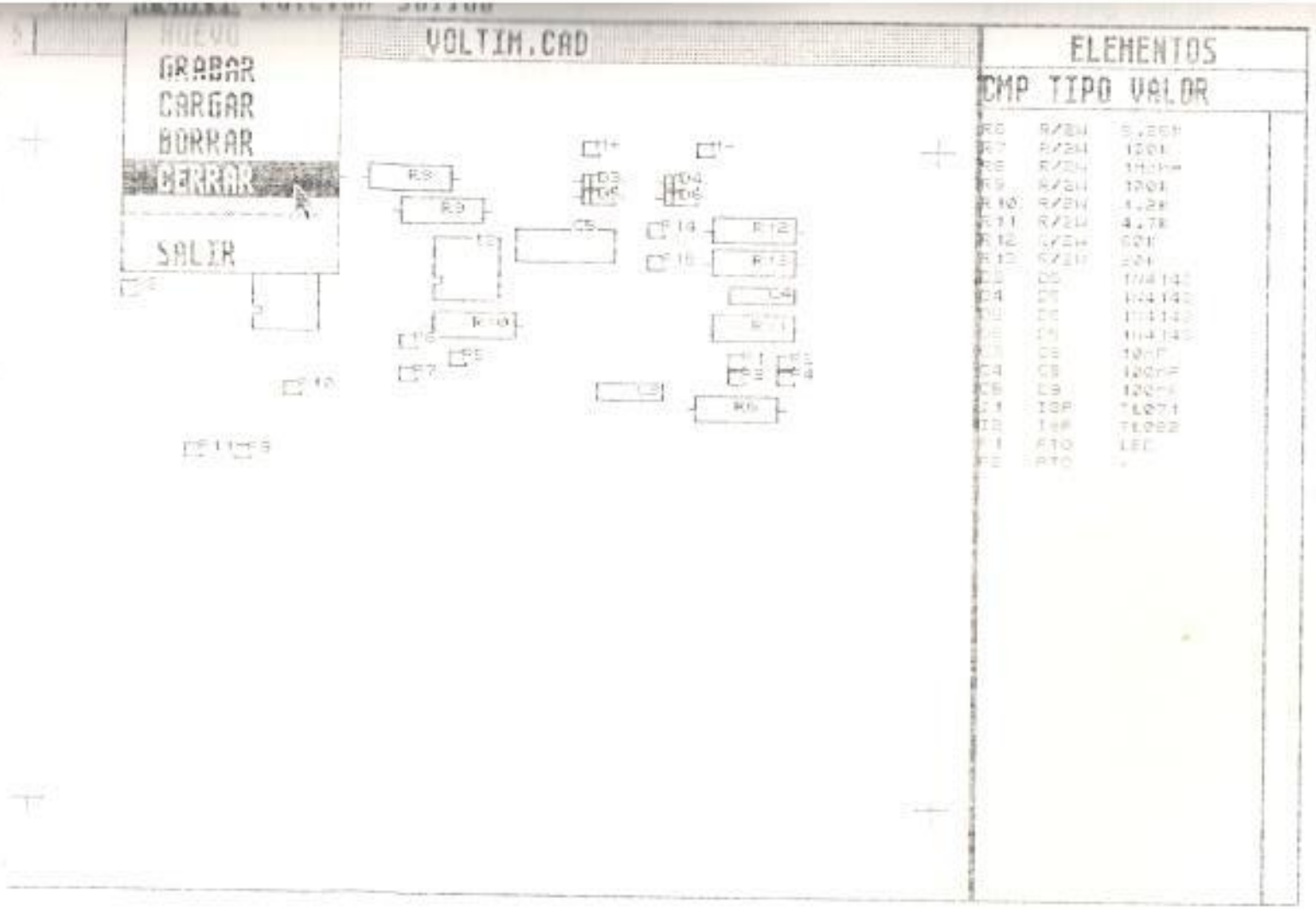


fig. 4.5.3.a Selecciona comando Cerrar del menú Archivo.

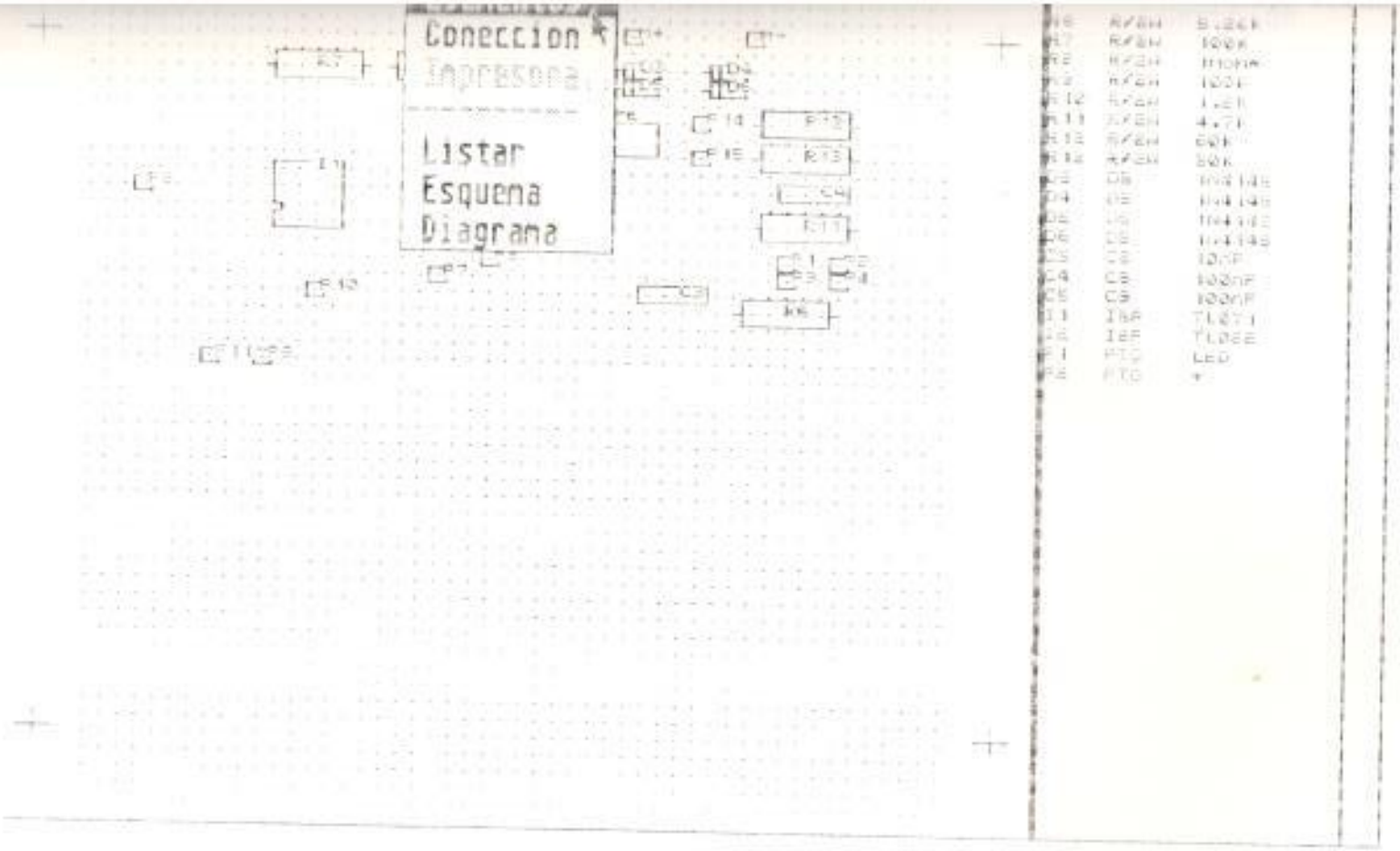


fig. 4.5.4.a Selecciona comando Elementos del menú Salida.

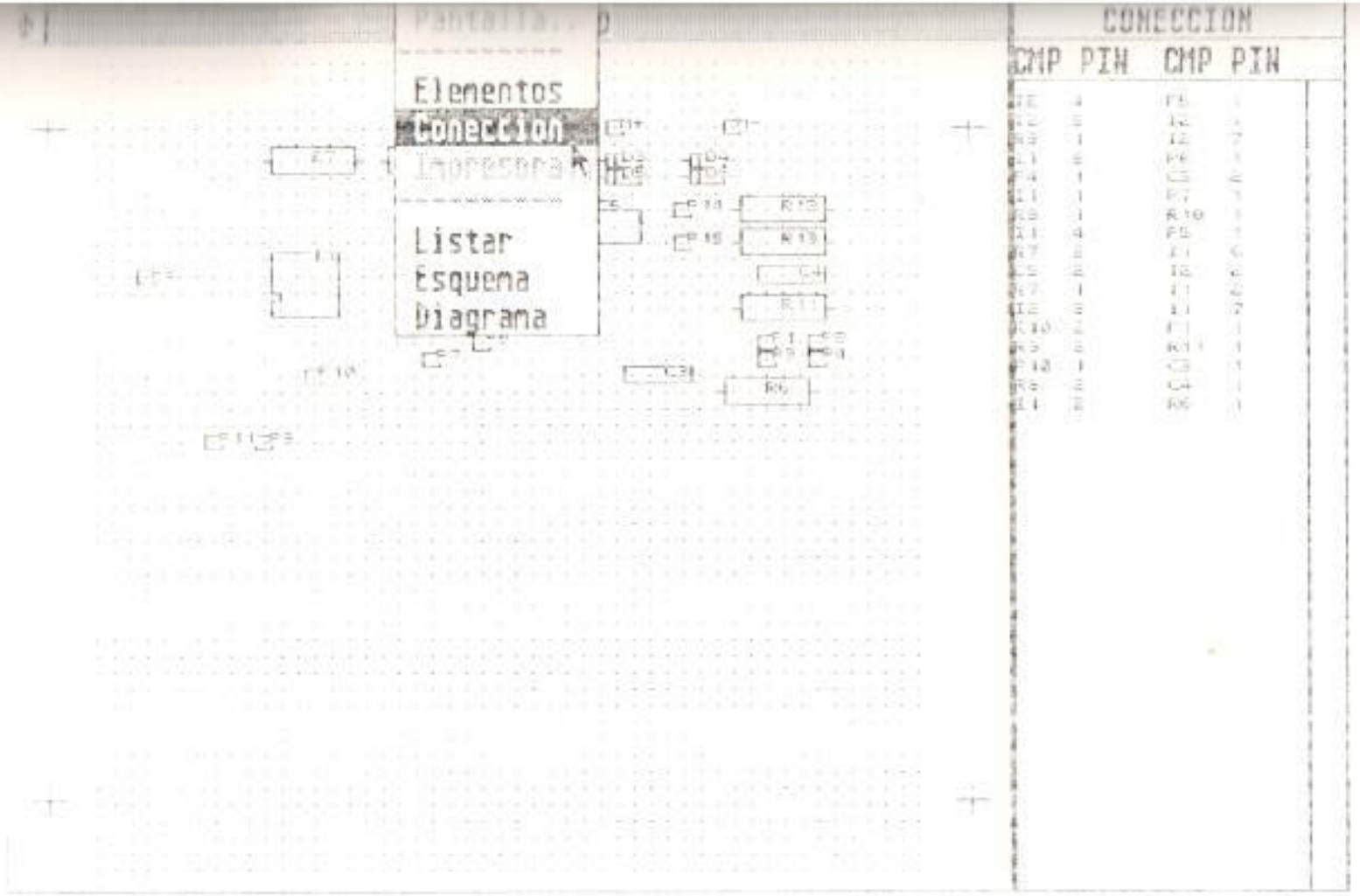


fig. 4.5.4.b Selecciona comando Conección del menú Salida.

de conecciones en la ventana texto y al igual que el anterior, lo realiza por grupos de 20.

- Si está interesado en consultar la librería de componentes, lo que tiene que hacer es cargar el archivo **Libreria.cad** y enseguida visualizará su contenido (fig. 4.5.4.c). Es importante que esta consulta la realice fuera de su diagrama, pues al cargar cualquier archivo, se pierde lo ya creado, si es que no lo ha grabado antes.

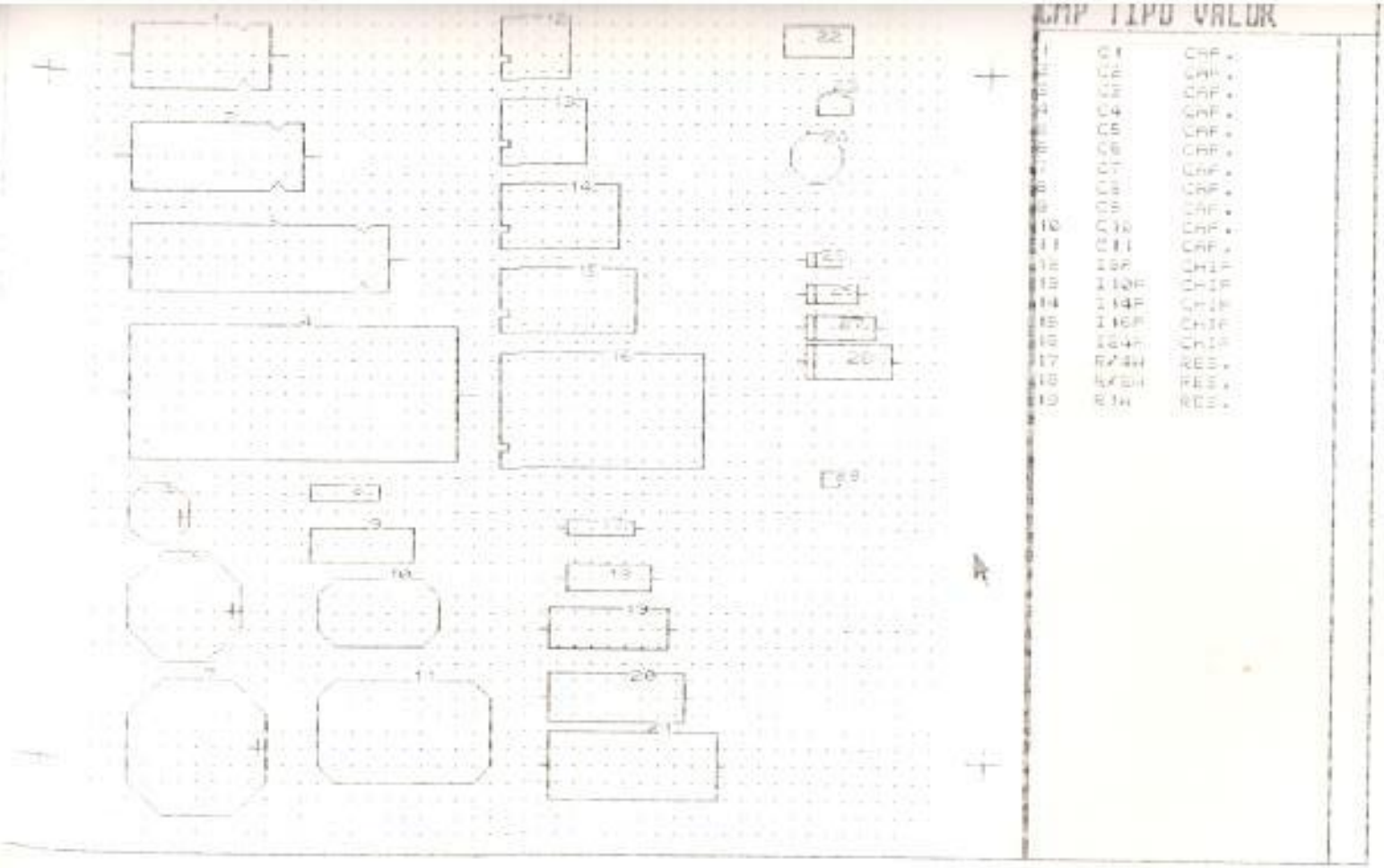


fig. 4.5.4.c Gráfico de todos los componentes de librería.

CONCLUSIONES Y RECOMENDACIONES

La presente tesis ha querido ofrecer al diseñador de circuitos impresos, una herramienta de trabajo que le ofrezca ahorrar el tiempo que dedica en la elaboración de los varios borradores antes de obtener un diseño final. Es asi, que con el computador adecuado y el programa que se ha realizado, hemos convertido al ordenador en el diseñador de los circuitos impresos, recargando en él todo el trabajo sin necesidad de volver a los antiguos borradores.

Vemos entonces, que el producto de utilizar un computador nos dá ventajas que no las teníamos en los métodos manuales de creación de diseños. Por ejemplo, podemos modificar la posición de uno o varios elementos sobre el tablero y entonces las conecciones relacionados a ellos se ajustarán automáticamente; las facilidades de almacenar los diseños para futuros usos con su respectiva documentación y lo más importante, obtener un gráfico impreso del diseño con la calidad de un dibujo original, suficiente para pasar a los procesos de fotografía.

De forma paralela, y tal vez con un mayor alcance por su contenido, la creación de este programa deja una completa documentación de todas las teorías desarrolladas, pues, por su innovación, algoritmos,

recursos y otras facetas relacionadas con el programa, convierten a este sistema en una valiosa fuente de información y consulta para los programadores C, los que desean saber como aprovechar las ventajas del entorno GEM y por que no los que ansian mejorar el programa.

Con la misma finalidad encontramos muchos programas producidos para el diseño de circuitos impresos; lo que nosotros ofrecemos es "una alternativa" que sin llegar al grado de sofisticación de otros, brinda en cambio la adaptabilidad a nuestro medio, pues fue concebida con la mayor sencillez de manejo para nuestros usuarios sin restar la complejidad para desarrollar los diagramas.

A nuestro programa lo hemos llamado DCI versión 1.0, pues deseamos que sea la base de futuras mejoras, ya que posee la estructura necesaria como para crecer y llegar al nivel de operación de otros similares.

```
/*****************************************************/
/*          A R C H I V O    D C I . C               */
/*****************************************************/
/*        DISENADOR DE CIRCUITOS IMPRESOS            */
/*          ASISTIDO POR ORDENADOR                   */
/*                                                   */
/*           Tesis de Electronica                    */
/*   Creado por:                                     */
/*            IVAN  AMAT  DIAZ                       */
/*   MAYO 1989                    E.S.P.O.L.         */
/*           Guayaquil - Ecuador                     */
/*****************************************************/

                        /* libreria de definiciones */
lude <obdefs.h>
lude <gemdefs.h>
lude <osbind.h>
lude <gembind.h>
lude <stdio.h>
lude <math.h>
lude "dci.h"            /* Definiciones RSC */

ne TIPO_VENT  {INFO;CLOSER;NAME}
ne VENT_TEXT {VSLIDE;UNARROW;UPARROW;INFO;CLOSER;NAME}
ne NOMBRE "DCI.RSC"     /* Nombre del fichero RSC */
ne VERDAD 1
ne FALSO 0
ne SALIR "FIN"
ne salir "fin"
ne edingr "*"
                        /* Variables Globales */
ontrl [12];
stin[128];
sin[128];
stout[128];
sout[128];

gbuff[8];
rk_in[11],text_in[11];
rk_out[57],text_out[57];
s[300];
s[1000],trd[1000],dist[1000],reloc[25];
mp[500],orpin[500],decmp[500],depin[500];

ento;
tura,ratony;
_nec,no_neng,no_necb,no_neco,ncomp,ncnaex,ncc;
n,final;
cho,alto,fila,col;
ep,item,bandera,edicion,adm;
ml,boton,m;
,fy,posx,posy,mx,my,nule,radio,
```

```
handle,handle_tex;
phys_handle;
gl_wchar,gl_hchar,gl_wbox,gl_hbox;
xdesk,ydesk,wdesk,hdesk;
xcaja,ycaja,wcaja,hcaja;
xwork,ywork,wwork,hwork;
xsel,ysel,wsel,hsel;
v_esquema,v_texto;
ap_id,cont,l,ag,mod,temp2;
nt int idnbole[2500],idchole[2500],estado[2500];

r filec[12];
r numeral[50];
                        /* Arreglos tipo estructura */

ct tipo {
char cmp[5];
char rem[4];
char valor[7];
int modo,cmpx,cmpy;

rt tipo ficha[80];

ct tipo1{
char clase[5];
char num[ ];
int ancho;
int alto;

ct tipo1 lib[40];

gemdos();
T *arbol_menu;
T *objeto_presenta;
T *arbol_selector;


*****************************************************/
    Abrir Parametros de la Estacion de Trabajo      */
*****************************************************/

_strbj()

t j;
=0; j<10; work_in[j++]=1);
  work_in[10]=2;
ndle=phys_handle;
pnvwk(work_in,&handle,work_out);
ndle_tex=phys_handle;
pnvwk(text_in,&handle_tex,text_out);
```

```
overlay "separa1"                    /* Divide la Compilacion en Partes */

/*************************************************************/
/*      Cerrar parametros de la Estacion de Trabajo          */
/*************************************************************/
exitop()

  wind_close(v_texto);
  wind_close(v_esquema);
  menu_bar(arbol_menu,FALSE);
  v_clsvwk();
  appl_exit();


/*************************************************************/
/*                Programa  Principal                        */
/*************************************************************/
in()

  ap_id=appl_init();
  phys_handle=graf_handle(&gl_wchar,&gl_hchar,&gl_wbox,&gl_hbox);
  wind_get(0,WF_WORKXYWH,&xdesk,&ydesk,&wdesk,&hdesk);
  abrir_virbj();
  abrir_menu();
  abrir_ventana();
  presentacion();
  v_clrwk(handle);
  motivos();
  inestado();
  si_raton();
  graf_mouse(ARROW,&no_mec);
  menu_bar(arbol_menu,VERDAD);
  graf_growbox(xdesk+wdesk/2,ydesk+hdesk/2,gl_wbox,gl_hbox,xdesk,ydesk,wdesk,hdesk);
  wind_open(v_texto,480,ydesk,wdesk-480,hdesk);
  wind_open(v_esquema,0,ydesk,wdesk-160,hdesk);
  wind_get(v_esquema,WF_WORKXYWH,&xwork,&ywork,&wwork,&hwork);
  borras();
  desktop();


/*************************************************************/
/*                Crear  Ventanas                            */
/*************************************************************/

abrir_ventana()

  v_esquema=wind_create(TIPO_VENT,xdesk,ydesk,wdesk,hdesk);
  wind_set(v_esquema,WF_NAME,"Diseñador de Circuitos Impresos",0,0);
  wind_set(v_esquema,WF_VSLSIZE,500,0,0,0);
  v_texto=wind_create(VENT_TEXT,xdesk,ydesk,wdesk-hdesk,
  wind_set(v_texto,WF_NAME," RESPOL ",0,0);
  wind_set(v_texto,WF_INFO,"Version 1.0",0,0);
```

```
/*************************************************************/
                Cargar Ficheros RSC                          */
/*************************************************************/

r_menu()

if(!rsrc_load(NOMBRE))
 Indicador1();
if(rsrc_gaddr(R_TREE,ARBOL0,&arbol_menu)==0)
 form_alert(1,"[3][ERROR GRAVE!! Archivo indisponible][Aborte]");
rsrc_gaddr(R_TREE,ARBOL3,&arbol_menu);
rsrc_gaddr(R_TREE,ARBOL1,&objeto_presental);
rsrc_gaddr(R_TREE,ARBOL2,&arbol_selector);


/*************************************************************/
          Selecciona un Comando del Menu                     */
/*************************************************************/

ma()

ntol;
ile (fin!=VERDAD){

eventoevnt_multi(MU_MESAG|MU_BUTTON|MU_M1|MU_KEYBD,1,1,1,bandera,0,0,wdesk-20,hdesk-20,0,0,0,0,0,
                 msgbuff,0,0,&ratonx,&ratony,&no_sec,&no_sec,&no_sec,&no_sec);
 if(evento & MU_MESAG){
  switch (msgbuff[0]){

     case WM_REDRAW:
       redibujar(msgbuff[4],msgbuff[5],msgbuff[6],msgbuff[7]);
       break;

     case MN_SELECTED:

       if(msgbuff[4]==MENSAJE1){
         raton_nuevo();
         form_alert(1,"[1][DISEÑO DE CIRCUITOS IMPRESOS| ASISTIDO POR ORDENADOR|Creado por:|
                    Ivan Amat Diaz| (E.S.P.O.L. - Mayo 1989)][SEGUIR]");
         menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
         graf_mouse(ARROW,&no_sec);
         break;
       }

       if(msgbuff[4]==ABRIEVEN){
         menu_ienable(arbol_menu,ABRIEVEN,FALSO);
         menu_ienable(arbol_menu,CARGARVE,FALSO);
         form_alert(1,"[1][Ud. va a crear un diseño nuevo! |Siga el menu  fotico.][Seguir]");
         menu_ienable(arbol_menu,CERRARVE,VERDAD);
         menu_ienable(arbol_menu,INGRESPEN,VERDAD);
         menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
         accap=0;
         break;
       }
```

```
if(msgbuff[4]==CERRARVE){
  mod=form_alert(0,"[3][Reiniciar sin Grabar. | si esta de acuerdo | oprima <Seguir>]
                   [Seguir | Cancelar]");
  if(mod == 1){
    wind_set(v_texto,WF_NAME," ESPOL ",0,0);
    wind_set(v_texto,WF_INFO,"Version 1.0",0,0);
    wind_set(v_esquema,WF_NAME,"Disenador de Circuitos Impresos",0,0);
    inicializar();
  }
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}

if(msgbuff[4]==GRABARVE){
  grabar();
  interesq();
  administra();
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}

if(msgbuff[4]==CARGARVE||msgbuff[4]==BORRARVE){
  fsel_input("A:\*.CAD",&filec,&no_necc);
  if(msgbuff[4]==BORRARVE){
     if(no_necc != 0){
        Fdelete(filec);
     }
     interesq();
     if(no_necc == 0)
         administra();
     menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
     break;
  }
  interesq();
  intertext();
  if(no_necc!=0){
    edicion=1;
    inicializar();
    cargar();
    resultado();
    menu_ienable(arbol_menu,ABRIRVEN,FALSO);
    menu_ienable(arbol_menu,CERRARVE,VERDAD);
    menu_ienable(arbol_menu,ELEMIMP,VERDAD);
    menu_ienable(arbol_menu,CONECIMP,VERDAD);
    menu_ienable(arbol_menu,ESQUEIMP,VERDAD);
    menu_ienable(arbol_menu,IMPDOC,VERDAD);
    menu_ienable(arbol_menu,DIASIMP,VERDAD);
    menu_ienable(arbol_menu,EDITARVE,VERDAD);
    menu_ienable(arbol_menu,AUTORUTA,VERDAD);
  }
  if(no_necc!=0) administra();
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}
```

```
if(msgbuff[4]==INGRVEN){
  if(edicion!=1)
     menu_ienable(arbol_menu,INGRVEN,FALSO);
  menu_bar(arbol_menu,FALSO);
  wind_close(v_texto);
  wind_close(v_esquema);
  no_raton();
  v_enter_cur(handle);
  ingresar();
  v_exit_cur(handle);
  menu_bar(arbol_menu,VERDAD);
  wind_open(v_texto,480,ydesk,wdesk-480,hdesk);
  wind_open(v_esquema,0,ydesk,wdesk-160,hdesk);
  si_raton();
  administra();
  menu_ienable(arbol_menu,ELEMIMP,VERDAD);
  menu_ienable(arbol_menu,POSICVEN,VERDAD);
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}

if(msgbuff[4]==EDITARVE){
  menu_ienable(arbol_menu,EDITARVE,FALSO);
  menu_ienable(arbol_menu,INGRVEN,VERDAD);
  edicion=1;
  form_alert(1,"[1][  Elija en el menu [ ] el tipo de Edicion deseado][Seguir]");
  menu_ienable(arbol_menu,ELEMIMP,VERDAD);
  menu_ienable(arbol_menu,CONSCIMP,VERDAD);
  menu_ienable(arbol_menu,POSICVEN,VERDAD);
  menu_ienable(arbol_menu,CONEXVEN,VERDAD);
  menu_ienable(arbol_menu,AUTODUTA,VERDAD);
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}

if(msgbuff[4]==POSICVEN){
  if(edicion!=1)
     menu_ienable(arbol_menu,POSICVEN,FALSO);
  posicionar();
  adm=2;
  menu_ienable(arbol_menu,ESQUEIMP,VERDAD);
  menu_ienable(arbol_menu,CONEXVEN,VERDAD);
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;


  if(msgbuff[4]==SALIR){
  ...

  if(edicion==1){
     editar();
     ...
     break;
```

```
  modo=2;
  intertext();
  mgujeros();
  menu_ienable(arbol_menu,AUTOBUTA,VERDAD);
  menu_ienable(arbol_menu,GRABARVE,VERDAD);
  menu_ienable(arbol_menu,CONECIMP,VERDAD);
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}

if(msgbuff[4]==AUTOBUTA){
  if(edicion!=1)
      menu_ienable(arbol_menu,AUTOBUTA,FALSO);
  interesq();
  intertext();
  autoruta();
  sim=1;
  menu_ienable(arbol_menu,DIAGIMP,VERDAD);
  menu_ienable(arbol_menu,CONECIMP,VERDAD);
  menu_ienable(arbol_menu,IMPDOC,VERDAD);
  if(edicion!=1)
     menu_ienable(arbol_menu,EDITARVE,VERDAD);
  menu_ienable(arbol_menu,GRABARVE,VERDAD);
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}

if(msgbuff[4]==ELEMIMP){
  listar(0);
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}

if(msgbuff[4]==IMPDOC){
  imp_lista();
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}

if(msgbuff[4]==CONECIMP){
  lconnex(0);
  menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
  break;
}

if(msgbuff[4]==ESQUEIMP){
  mod=form_alert(1,"[1][Para Imprimir: |   Pulse Seguir | Seguir|Cancelar]");
  if (mod==1){
    while (!Bcostat(0)){
    form_alert(1,"[3][La impresora no esta lista | Revise sus conexiones][Seguir]");
    break;
  }                          /* fin de while */
```

```
        if(Bcostat(0)==-1){
          resultado();
          no_raton();
          Setprt(4);
          Scrdmp();
          si_raton();
        }}                          /* fin de if */
        menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
        break;
      }

      if(msgbuff[4]==DIAGIMP){
        mod=form_alert(1,"[1][Para Imprimir: ]   Pulse (Seguir) ? Seguir:Cancelar]");
        if(mod==1){
          while (!Bcostat(0)){
            form_alert(1,"[3][La Impresora no esta lista ] [ Revise sus conexiones][Seguir]");
            break;
          }
          if(Bcostat(0)==-1){
            impdiag();
        }}
        menu_tnormal(arbol_menu,msgbuff[3],VERDAD);
        break;
      }

      if(msgbuff[4]==QUIT){
        wind_close(v_texto);
        wind_close(v_esquema);
        wind_delete(v_esquema);
        wind_delete(v_texto);
        menu_bar(arbol_menu,FALSO);
        rsrc_free();
        v_clsvwk();
        appl_exit();
        exit();
        break;
      }
      break;
    } /* final del switch */
  } /* final del if */
 final del while */
de tareas */

**********************************************/
    Ingreso de Datos de los Componentes      */
**********************************************/


clo,prueba,movfila,datol;
].dator[4],datov[5];


Handle,"CMS PRV  VALOR  : Presione (Return) para cada dato ingresado"
ow(handle,27,2);
```

```
v_curtext(handle,"Para Borrar digite (#)    Para Finalizar digite (fin): ");
fila=1; col=1; datoi=1;
if(edicion==1)
   lista_ingr(ncomp);
while(por != VERDAD){
   switch (datoi){
     case 1:
        espacio=buscar_sitio();
        if(espacio>19 && espacio<40){
           movfila=20;  col=18;
        }
        if(espacio>39 && espacio<60){
           movfila=40;  col=35;
        }
        if(espacio>59){
           movfila=60;  col=52;
        }
        if(espacio==-1){
           printf("\n lista llena");
           return;
        }
        fila=2+espacio-movfila;
        vs_curaddress(handle,fila,col);
        gets(dator);
        if(strcmp(dator,salir)==0 || strcmp(dator,SALIR)==0){
          por=VERDAD;
          return;
        }
        if(strcmp(dator,edingr)==0){
          datoi=4; break;
        }
        prueba=verificar(3,dator);
        if(prueba==1){
          strcpy(ficha[espacio].rem,dator);
          datoi=2; col=col+4;
        }
        break;
     case 2:
        vs_curaddress(handle,fila,col);
        gets(datoc);
        prueba=verificar(4,datoc);
        if(prueba==1){
          i=1;
          do {
             if(strcmp(lib[i++].clase,datoc)==0){
               strcpy(ficha[espacio].cmp,datoc);
               ficha[espacio].modo=i-1;
               ncomp=ncomp+1;
               datoi=3; col=col+4;
             }
          } while(i<31);
          if(i>30) verificar(0,datoc);
        }
        break;
```

```
se 3:
    vs_curaddress(handle,fila,col);
    gets(datov);
    prueba=verificar(6,datov);
    if(prueba==1){
      strcpy(ficha[espacio].valor,datov);
      datoi=1;
      col=col-9;
    }
    break;
se 4:
    datoi=1;
    borrar_lin();
    v_rvoff(handle);
    lista_ingr(ncomp);
    vs_curaddress(handle,25,2);
    v_curtext(handle,"Para Borrar digite (4)    Para Finalizar digite (fin)");
    break;
 * final del switch */
final del while */
 de introducir */


*****************************************************/
Comprobar Longitud de Caracteres por Dato Ingresado    */
*****************************************************/

a,palabra)

abra;

(palabra))>n){
raddress(handle,fila,col);
(handle);
n 1;


*****************************************************/
Asigna un Numero disponible a una Ficha Nueva         */
*****************************************************/


t;
ficha[t].cmp[0] && t<40;...
return -1;
```

```
/*****************************************************************/
/*    Visualiza por Pantalla los datos de los componentes        */
/*****************************************************************/

listar(nm)
int nm;

int j,t;
 j=0;
 wind_set(v_texto,WF_NAME," ELEMENTOS ",0,0);
 wind_set(v_texto,WF_INFO,"CMP TIPO VALOR",0,0);
 intertext();
 vst_height(handle,4,&no_sec,&no_sec,&no_sec,&no_sec);
 for(t=nm;t<nm+19;++t){
  if(t<ncomp){
   v_gtext(handle,48,66+8*j,ficha[t].rem);
   v_gtext(handle,550,66+8*j,ficha[t].valor);
   j++;
  }
  else break;

if(edicion==1){
 j=0;
 for(t=nm;t<nm+19;++t){
   if(t<ncomp)
    v_gtext(handle,510,66+8*j++,ficha[t].cmp);
   else break;
}
              /* fin de for */
              /* fin de if  */


/*****************************************************************/
      Ubica los elementos sobre la rejilla                     */
/*****************************************************************/

loear(){

tempx,tempy,figw,figh,transito,alternate;
edicion==1){
resultado();
editpos();
interesq();
comp();
listar(0);
return;

set(v_texto,WF_NAME," COLOCACION ",0,0);
set(v_texto,WF_INFO,"CMP TIPO VALOR",0,0);
resq();
lar();
r(0);
```

```
for(i=0;i<ncomp;i++){
   v_gtext(handle,505,390,"                ");
   v_gtext(handle,510,390,ficha[i].cmp);
   v_gtext(handle,540,390,ficha[i].rem);
   evnt_button(1,1,1,&ratonx,&ratony,&no_nec,&alternate);
   if(alternate==K_ALT)
      i--;
   mod=ficha[i].modo;
   figw=lib[mod].ancho; figh=lib[mod].alto;
   graf_dragbox(figw,figh,ratonx,ratony,39,44,400,350,&posx,&posy);
   tempx=(posx-35)/8+1;     tempy=(posy-40)/7+1;
   evnt_mouse(0,xdesk,ydesk,wdesk,hdesk,&tx,&ty,&no_nec,&no_nec);
   if(tx<400 && tx>39 && ty>44 && ty<350){
     transito=1;
     no_raton();
     ficha[i].cmpx=tempx; ficha[i].cmpy=tempy;
     figura(tempx,tempy);
     v_gtext(handle,8*tempx+35+figw/2,7*tempy+40,ficha[i].rem);
     v_gtext handle,515,66+8*i,ficha[i].cmp);
     si_raton();
   }                          /* final de if */
   if(transito!=1) i--;
   transito=0;
                              /* final de for */


*****************************************************/
     Prepara los elementos a Conectarse             */
*****************************************************/

s()

mpx,tempy,refx,refy,tip;
nt idc,idz;
r[3];
set(v_texto,WF_NAME,"CONECCION",0,0);
set(v_texto,WF_INFO,"CMP PIN  CMP PIN",0,0);

esq();
();
();
*(j);



*****************************************************/
   Realiza la conexion entre dos terminales        */
*****************************************************/

a)


t id;in;
```

```
wind_set(v_esquema,WF_INFO,"Para terminar de conectar seleccione FIN con el raton",0,0);
v_gtext(handle,3,44,"fin");
t=0;
if(ja>19 && ja/20+ja%20==ja/20){ t=ja/20+ja%20; lconex(t); }
final=VERDAD;
while(final==VERDAD){
 primera:
  evnt_button(1,1,1,&tempx,&tempy,&no_nec,&no_nec);
  if(tempx<30 && tempy<hdesk)
    break;
  graf_dragbox(8,7,tempx,tempy,30,44,400,350,&ratonx,&ratony);
  tempx=(ratonx-35)/8+1;    tempy=(ratony-40)/7+1;
  refx=8*tempx+35;          refy=7*tempy+40;
  tip=tempx+(tempy-1)*50;
  if(estado[tip]<66 || estado[tip]>72) goto primera;
  tro[ja]=tip;
  no_raton();
  v_circle(handle,refx,refy,3);
  idc=idchole[tip];
  orcmp[ja]=idc;
  v_gtext(handle,482,66+8*t,ficha[idc].rem);
  idn=idnhole[tip];
  orpin[ja]=idn;
  v_gtext(handle,510,66+8*t,pin[idn].num);
  si_raton();

  graf_dragbox(8,7,tempx,tempy,30,44,400,350,&ratonx,&ratony);
  tempx=(ratonx-35)/8+1;    tempy=(ratony-40)/7+1;
  refx=8*tempx+35;          refy=7*tempy+40;
  tip=tempx+(tempy-1)*50;
  if(estado[tip]<66 || estado[tip]>72) goto segunda;
  trd[ja]=tip;
  no_raton();
  v_circle(handle,refx,refy,3);
  idc=idchole[tip];
  decmp[ja]=idc;
  v_gtext(handle,555,66+8*t,ficha[idc].rem);
  idn=idnhole[tip];
  depin[ja]=idn;
  v_gtext(handle,583,66+8*t,pin[idn].num);
  si_raton();
  dist[ja]=abs(trd[ja]%50-tro[ja]%50)+abs(trd[ja]/50-tro[ja]/50);
  if(tro[ja]>trd[ja]){
    tip=trd[ja]; trd[ja]=tro[ja]; tro[ja]=tip;
    tip=decmp[ja]; decmp[ja]=orcmp[ja]; orcmp[ja]=tip;
    tip=depin[ja]; depin[ja]=orpin[ja]; orpin[ja]=tip;
  }
}

ja=ja;
```

```
**********************************************************/
epara los Elementos a Conectarse Automaticamente      */
**********************************************************/


()

ndo();
=3;
ton();
ype(handle,3);
();
ype(handle,1);
();
flcn();
o();
ton();


**********************************************************/
    Indica todas las conecciones existentes           */
**********************************************************/




et( _texto,WF_NAME,"NO CONECTADO",0,0);
et(=_texto,WF_INFO,"CMP PIN  CMP PIN",0,0);
0;j<nconex;j++) ruta(j);


**********************************************************/
  Elije una alternativa posible de coneccion          */
**********************************************************/



td,de,tv,th,y1,y2,x1,x2,tt,alt;
t idc,idn;
 alt=1;
alt){
ch(mod){
se 1:
or=tro[jc]; de=trd[jc];
x1=or%50*8+35;  y1=or/50*7+40;
x2=de%50*8+35;  y2=de/50*7+40;
if (y1 == y2){
  tt=de-1;
  if(recorre_h(or,tt) == 0) {mod=9; break;}
  dibrect(or,de);
  mod=1; alt=0;
  break;
}
break;
```

```
 case 2:
     or=tro[jc]; de=trd[jc];
     x1=or%50*8+35;  y1=or/50*7+40;
     x2=de%50*8+35;  y2=de/50*7+40;
     if(x1 == x2){
      or=tro[jc]; de=trd[jc];
      if(recorre_v(or,de-50) == 0){
         mod=9;
         break;
      }
      dibrect(or,de);
      mod=1; alt=0;
      break;
    }
    mod++;
    break;
 case 3:
    or=tro[jc];  de=trd[jc];
    otd=e_a(or,de);
    if(otd==0){
       mod++;
       break;
    }
    else{
     dibele(or,otd,de);
     mod=1; alt=0;
     break;
   }
case 4:
   or=tro[jc];  de=trd[jc];
   otd=ele_b(or,de);
   if(otd==0){ mod++; break; }
   else{
    dibele(or,otd,de);
    mod=1;
    alt=0;
    break;
  }
case 5:
  mod++;
  if(tro[jc]%50 > trd[jc]%50) break;
  or=tro[jc]; de=trd[jc];
  tt=or+(de%50-or%50);
  while(or<tt && estado[++or]==64){
    otd=ele_a(or,de);
    if (otd != 0){
     dibele(or,otd,de);
     dibrect(tro[jc],or);
     estado[or]=78;
     mod=1; alt=0;
     or=tt;
    }
  }
  break;
```

```
  case 6:
     mod++;
     if(tro[jc]%50 > trd[jc]%50) break;
     or=tro[jc]; de=trd[jc];
     tt=or+(de/50-or/50)*50;
     or=or+50;
     while(or<tt && estado[or]==64){
       otd=ele_b(or,de);
       if(otd != 0){
         dibele(or,otd,de);
         dibrect(tro[jc],or);
         estado[or]=80;
         mod=1; alt=0;
         or=tt;
       }                          /* fin de if */
       or=or+50;
     }                            /* fin de while */
     break;
  case 7:
     mod++;
     if(trd[jc]%50 > tro[jc]%50) break;
     or=tro[jc]; de=trd[jc];
     tt=de-(de/50-or/50)*50;
     de=de-50;
     while(de>tt && estado[de]==64){
       otd=ele_a(or,de);
       if(otd != 0){
         dibele(or,otd,de);
         dibrect(de,trd[jc]);
         estado[de]=84;
         mod=1;
         alt=0;
         de=tt;
       }
       de=de-50;
     }
     break;
  case 8:
     mod++;
     if(trd[jc]%50 > tro[jc]%50) break;
     or=tro[jc]; de=trd[jc];
     tt=de+(or%50-de%50);
     while(de<tt && estado[++de]==64){
        otd=ele_b(or,de);
        if(otd != 0){
          dibele(or,otd,de);
          dibrect(trd[jc],de);
          estado[de]=74;
          mod=1;
          alt=0;
          de=tt;
        }

     break;
```

```
    case 9:
       mod=1;
       alt=0;
       ncc++;
       or=tro[jc];     de=trd[jc];
       idc=orcmp[jc];  idn=orpin[jc];
       v_gtext(handle,482.66+8*ncc,ficha[idc].rem);
       v_gtext(handle,510.66+8*ncc,lib[idn].num);
       idc=decmp[jc];  idn=depin[jc];
       v_gtext(handle,555.66+8*ncc,ficha[idc].rem);
       v_gtext(handle,585.66+8*ncc,lib[idn].num);
       break;
   }
}



**********************************************************/
            Asigna los vertices de una  L                 */
**********************************************************/

_h(or,de)
or,de;

tv,th,otd,x1,x2,y1,y2;
=or; th=or;
td=abs(de/50-or/50)*50+tv;
=tv;    y2=otd;
=otd-1; x2=th-1;
f(otd>th){x1=th; x2=otd;}
f(recorre_v(y1,y2)==0 || recorre_h(x1,x2)==0)
return 0;
turn otd;


**********************************************************/
      Asigna los vertices de una  L  invertida          */
**********************************************************/

or,de)
,de;

,th,otd,x1,x2,y1,y2;
=, th=or;
tv-abs(de/50-or/50)*50;
d-50; y2=tv-50;
d-1;  x2=th-1;
d>th){ x1=th; x2=otd; }
corre_v(y1,y2)==0 || recorre_h(x1,x2)==0)
rn 0;
 otd;
```

```
/*****************************************************************/
/*      Revisa si hay o  no cruce de rutas horizontales          */
/*****************************************************************/
recorre_v(y1,y2)
int y1,y2;
{
 while(y1<=y2){
    if(y1==y2) return 1;
    y1=y1+50;
    if(estado[y1]>=66) return 0;
 }


/*****************************************************************/
/*      Revisa si existe o no cruce de rutas verticales          */
/*****************************************************************/
ecorre_h(x1,x2)
t x1,x2;

while(x1<=x2);
   if(x1==x2) return 1;
   x1=x1+1;
   if(estado[x1]>=66) return 0;
}


/*****************************************************************/
      Dibuja una línea recta entre dos terminales              */
/*****************************************************************/

rect(or,de)
 or,de;

 lin[5];
 xo,yo,xd,yd;
o=or%50*8+35;    yo=(or/50+1)*7+40;
d=de%50*8+35;    yd=(de/50+1)*7+40;
in[0]=xo;   lin[1]=yo;   lin[2]=xd;   lin[3]=yd;
_pline(handle,2,lin);
f(yo==yd){
 if(estado[or]==66) estado[or]=70;
 if(estado[or]==68) estado[or]=72;
while(or<de-1)
   estado[++or]=74;

o==xd){
f(estado[de]==66) estado[de]=68;
f(estado[de]==70) estado[de]=72;
ile(or<de-50){
 or=or+50;
 estado[or]=82;
                /* fin de while */
                /* fin de if */
```

```
**************************************************************/
  Dibuja entre dos terminales dos lineas en angulo recto   */
**************************************************************/

ble(or,otd,de)
or,otd,de;

lin[6];
xo,yo,xt,yt,xd,yd;
=or%50*8+35;  yo=(or/50+1)*7+40;
=otd%50*8+35; yt=(otd/50+1)*7+40;
=de%50*8+35;  yd=(de/50+1)*7+40;
n[0]=xo;  lin[1]=yo;
n[2]=xt;  lin[3]=yt;
n[4]=xd;  lin[5]=yd;
pline(handle,3,lin);

(yt==yd){
while(or<otd-50){
   or=or+50;
   estado[or]=82;
}
if(xt<=xd){
  estado[otd]=80;
  while(otd<de-1)
     estado[++otd]=76;
}
if(xt>xd){
  estado[otd]=74;
  if(estado[de]==66) estado[de]=70;
  if(estado[de]==68) estado[de]=72;
  while(de<otd-1)
     estado[++de]=76;


yt==yo){
f(estado[de]==66) estado[de]=68;
f(estado[de]==70) estado[de]=72;
hile(otd+50<de){
  de=de-50;
  estado[de]=82;

f(xo<=xt){
 estado[otd]=78;  estado[or]=70;
 while(or<otd-1)
    estado[++or]=76;

(xo>xt){
estado[otd]=84;
while(otd<or-1)
   estado[++otd]=76;
```

```
/*****************************************************************/
/*          Dibuja en Pantalla todos los Componentes             */
/*****************************************************************/
lcomp()
{
int figw,ja,tempx,tempy;
 no_raton();
 for(ja=0;ja<ncomp;ja++){
    mod=ficha[ja].modo;   figw=llb[mod].ancho;
    tempx=ficha[ja].cmpx; tempy=ficha[ja].cmpy;
    figura(tempx,tempy);
    v_gtext(handle,8*tempx+35+figw/2.7*tempy+40,ficha[ja].rem);
 }
 si_raton();
}

/*****************************************************************/
/*     lista por Pantalla los datos sobre las conecciones        */
/*****************************************************************/
lconex(nm)
int nm;
{
int j,t,or,de;
short int idc,idn;
 intertext();
 wind_set(v_texto,WF_NAME,"CONECCION",0,0);
 wind_set(v_texto,WF_INFO,"CMP PIN  CMP PIN",0,0);
 t=0;
 for(j=nm;j<nm+19;j++){
  if(j<nconex){
    idn=orpin[j];   idc=orcmp[j];
    v_gtext(handle,482,66+8*t,ficha[idc].rem);  v_gtext(handle,510,66+8*t,llb[idn].num);
    idn=depin[j];   idc=decmp[j];
    v_gtext(handle,555,66+8*t,ficha[idc].rem);  v_gtext(handle,585,66+8*t,llb[idn].num);
    t++;
  }                       /* fin de if */
  else break;
 }                        /* fin de for */
}

/*****************************************************************/
/*          Dibuja por Pantalla todos los Terminales             */
/*****************************************************************/
lterm()

int tempx,tempy;

  ...
    mod=ficha[j1].modo;     temp2=j1;
    tempx=ficha[j1].cmpx;   tempy=ficha[j1].cmpy;
    terminal(tempx,tempy);
  }

}
```

```
/*****************************************************************/
/*      Dibuja un Esquema Completo del Arte por Pantalla         */
/*****************************************************************/

resultado()
{
   interesq();
   listar(0);
   dibujar();
   lcomp();
}
```

BIBLIOGRAFIA

1. HERBERT SCHILDT. Programación en Lenguaje C. Mc Graw Hill. 1985

2. SZCZEPANOWSKI. GEM Programer's Reference. Abacus Software. 1985

3. MEGAMAX. Megamax C compiler. Manual de Operación, 1986

4. BRUCKMANN. Atari ST Consejos y Trucos. Ferre Moret, 1986

5. BISHOP GRAPHICS. Printer Circuit Drafting Technical Manual & Catalog 10', 1982

6. DARRYL LINDSEY. The Design & Drafting of Printed Circuits. Bishop Graphics. 1985

7. GERITS. Sistemas CAD / CAM / CAE. Serie Mundo Electrónico. 1986