8
2 Ej.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES

" C U A U T I T L A N "

" DISEÑO DE PISTAS PARA CIRCUITOS " IMPRESOS POR COMPUTADORA

T E S I S

QUE PARA OBTENER EL TITULO DE :

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A :

EDUARDO CASTILLO CASTAÑEDA

Cuautitlán Izcalli 1 9 8 7

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

I INTRODUCCION

La evolución en el campo de la computación, ha dado lugar a la creación de una nueva técnica conocida como AUTOMATIZACION -- que en terminos sencillos es la utilización de computadoras digitales para el auxilio en la generación, verificación y almacenamiento de datos y documentos que constituyen el diseño de sistemas digitales, ésto es, sistemas digitales para crear sistemas - digitales.

El diseño del circuito impreso y el montaje de los elemen-- tos físicos en la tableta, son los procesos finales de esta automatización.

En la actualidad se recurre, aún en muchas empresas, al diseño manual del circuito impreso, lo que implica varios días de- trabajo de dibujantes y técnicos; el trabajo final, en estos casos, no es siempre el óptimo, sobre todo en circuitos impresos - complejos o de pequeñas dimensiones y los errores humanos producen perdidas considerables en tiempo y recursos. Cuando se ha-- bla de producción en serie, un diseño defectuoso, desde el punto de vista de la optimización acarrea altos costos de fabricación- ya que, por ejemplo, no es igual utilizar 10 gr. de cobre para - los conductores que utilizar 15 o 20 gr.

El diseño del circuito impreso puede subdividirse en proble

mas mas especificos:

1.- PARTICIONAMIENTO Y ASIGNACION.- Esto es, el análisis para la optimización del circuito eléctronico, desde el punto de - vista de componentes, nodos y mallas.

2.- ALOJAMIENTO.- Es la distribución de los elementos en la tableta (incluye elección de su tamaño).

3.- RECORRIDO (Routing).- Es el proceso que define formal-- mente las trayectorias precisas de los conductores para la interconexión de los elementos en la tableta.

Dada la magnitud que representa el problema del Diseño del- Circuito Impreso, el tema de esta Tesis se enfoca al último punto:

EL RECORRIDO

Para la solución se parte de que la distribución, de los -- elementos en la tableta, ya ha sido realizada y se trata, entonces de configurar las trayectorias de los conductores en la ta-- bleta, tomando en cuenta los siguientes puntos:

- Espaciamiento entre conductores.
- Ancho de conductor.
- Cruces no permitidos.
- Optimización de área de conductor.

Los Capítulos siguientes presentan una solución al problema del recorrido, utilizando un Sistema Computacional en lenguaje - de alto nivel (PASCAL); las ventajas al utilizar una computadora son: velocidad de diseño, optimización en el área de conductor utilizada y exactitud en la escala de diseño.

El segundo capítulo de este trabajo presenta conceptos simples, aunque no básicos, de programación, con el objetivo de que se familiarice al lector con las estructuras y el lenguaje manejados en el desarrollo y utilización del Sistema.

El tercer capítulo tiene un objetivo similar al anterior, - brindar al lector elementos y conceptos fundamentales sobre el - Diseño de Circuitos Impresos.

El cuarto capítulo es propiamente el diseño del Sistema Computacional, en él se indican algoritmos y su fundamentación teórica. Es en este capítulo donde se describe la parte "inteligente".

El quinto capítulo es el manual del Usuario del Sistema, en este, se da una explicación detallada de cómo utilizarlo en la - práctica.

Se ha omitido, hasta donde ha sido posible, la utilización- de palabras demasiado técnicas, y se ha tratado de expresar todo en un lenguaje sencillo y común sin llegar a deteriorar el fondo

del tema; la programación es compleja, subjetiva, es sencillo -- "perderse" en lo abstracto de un algoritmo, el lenguaje no debe- ser un obstáculo más para su entendimiento.

II CONCEPTOS DE PROGRAMACION

Un lenguaje de alto nivel es un medio de comunicación entre el ser humano y una computadora, es por medio de este que el programador puede "ordenar" a la computadora que efectúe una secuencia determinada de operaciones lógicas o aritméticas para el proceso de la información.

El lenguaje de Alto Nivel utilizado para la programación de este Sistema es PASCAL (ref 7), que fue diseñado por el Suizo N. Wirth en 1971. Se eligió este lenguaje porque permite, en forma natural, la programación estructurada para lograr sencillez, claridad y continuidad en el programa; además PASCAL cuenta con la posibilidad de manejar estructuras dinámicas para la optimización de memoria.

Ya programado el Sistema, se utiliza un lenguaje definido en el mismo, para establecer la comunicación entre Sistema y Usuario final, de forma tal que, no es necesario que se tengan grandes conocimientos de computación para poder elaborar un Diseño de Circuito Impreso.

a) Estructura y diseño de Lenguajes.

Todo lenguaje esta fundamentado en un VOCABULARIO. Sus elementos se llaman normalmente PALABRAS; en el campo de los lenguajes formales, estas palabras se llaman SIMBOLOS. Es una caracte

rística de los lenguajes que algunas secuencias de símbolos sean consideradas FRASES correctas del lenguaje, y otras sean incorrectas o mal formadas.

La Gramática, sintaxis o estructura del lenguaje, determina que una secuencia de palabras sea una frase correcta o no. De hecho se define la sintaxis como el conjunto de reglas o formulas que determinan el conjunto de frases, formalmente correctas. Sin embargo, tal conjunto de reglas no solo permite decidir si una cierta secuencia de palabras es una frase, sino que, también algo que es más importante, dotan a la frase de una estructura que ayuda a encontrar su significado. Resulta evidente por ello que la sintaxis y la semántica (significado) estén intimamente ligadas. Por lo tanto, las definiciones estructurales deben considerarse siempre como ayuda para un fín superior. Esto no debe impedir estudiar inicialmente los aspectos estructurales sin considerar los aspectos de significado e interpretación.

Sea por ejemplo la frase: "María sueña". "María" es el sujeto y "sueña" es el predicado. Esta frase pertenece al lenguaje que puede ser definido, por ejemplo, por la sintaxis siguiente:

<frase> ::= <sujeto> <predicado>

<sujeto> ::= María / Pedro

<predicado> ::= Sueña / Come

El significado de las líneas anteriores es:

a) Una frase está formada por un sujeto seguido de un predicado.

b) Un sujeto es, bien la palabra "María", o bien "Pedro"

c) Un predicado es, bien la palabra "Sueño", o bien la palabra "come"

La idea es que una frase puede deducirse a partir del símbolo inicial <frase>, aplicando repetidamente las reglas de sustitución. La notación con que se describen estas reglas se llama BNF (Backus Naur Form). Los objetos <frase>, <Sujeto> y <predicado> se llaman símbolos NO TERMINALES; "María", "Pedro", "sueña" y "come" se llaman símbolos TERMINALES, y las reglas sintácticas PRODUCCIONES. Los símbolos ::= y / son METASIMBOLOS de la notación BNF.

El objetivo primordial de los analizadores de lenguaje no es la generación de frases sino el reconocimiento de ellas y de su estructura. Esto lleva consigo reconstruir, al leer una frase, los pasos que conducen a la generación de la misma, y seguirlos de nuevo. La complejidad de esta tarea depende fundamentalmente del tipo de reglas sintácticas utilizadas para definir el lenguaje. La teoría del análisis sintáctico se encarga de desarrollar algoritmos de reconocimiento para lenguajes con reglas estructurales bastante complicadas. Aquí sin embargo, solo se pretende mostrar los puntos principales de un método que sirve para construir " reconocedores" suficientemente sencillos y eficaces.

Es obvio que no hay que molestarse en buscar un algoritmo - que reconozca cualquier lenguaje dado, sino que dado un lenguaje, definir un algoritmo eficaz para analizar ese lenguaje específico.

Una primera secuencia de la eficacia requerida es que, en - cada paso de análisis de una frase, la elección de que hacer a - continuación debe basarse solo en el estado de la computación y- en un único símbolo (el siguiente a analizar). Otra consecuencia la de mayor importancia, es que no se anule ningún paso realizado previamente. Normalmente, estas dos restricciones se conocen con el término técnico de un SIMBOLO ADELANTE SIN VUELTA ATRAS.

El método básico a considerar es el llamado ANALISIS DESCENDENTE, porque intenta reconstruir los pasos de generación a partir del símbolo inicial, hasta llegar a la frase final, avanzando en la estructura desde arriba hacia abajo.

Se abordará ahora el problema de cómo convertir este algo-- ritmo en algo concreto, fácil de programar. Se pueden aplicar - dos técnicas que son escencialmente diferentes. Una es diseñar- un programa analizador descendente que valga para cualquier gramatica posible. En este caso, se codifica una gramática específica como una cierta estructura de datos que el programa lee, y este opera en base a ella. La otra técnica, es diseñar un programa analizador que sea específico para el lenguaje dado, y cons--

truirlo sistemáticamente según un conjunto de reglas que trans-- forman una sintaxis dada en una secuencia de instrucciones, es - decir en un programa.

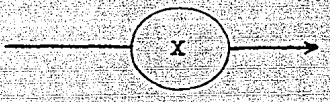
En cualquiera de las dos técnicas que se utilice, es necesario representar la sintaxis dada por UN GRAFO DE RECONOCIMIENTO, que llamaremos DIAGRAMA DE SINTAXIS. Este diagrama refleja el - flujo de control durante el análisis de una frase.

Una característica del método descendente es que el objetivo del proceso de análisis se conoce al comienzo del mismo. El- objetivo es reconocer una frase, es decir, una secuencia de símbolos generables a partir del símbilo inicial. La aplicación de una producción, es decir, la sustitución de un símbolo único por una secuencia de símbolos, corresponde a la división de un objetivo único en un número de objetivos parciales, a conseguir en - orden específico. Por ello, el método descendente se conoce con el nombre de ANALISIS POR OBJETIVOS.

Al construir un analizador, es fácil aprovechar esta correspondencia obvia entre símbolos no terminales y objetivos; se - - construye un analizador parcial para cada símbolo no terminal. Cada analizador parcial tiene el objetivo de reconocer una sub-- frase generable a partir de su correspondiente símbolo no terminal. Como se desea construir un grafo que represente el analizador completo, se hará corresponder un subgrafo con cada símbolo-

no terminal. Esto conduce a las siguientes convenciones para -- construir un diagrama de sintaxis:

1.- Cada aparición de un símbolo no terminal corresponde a- una instrucción que reconozca este símbolo, seguida del avance - de la posición de lectura del símbolo siguiente de la secuencia- de entrada. Esto representado por el símbolo terminal encerrado en un círculo:



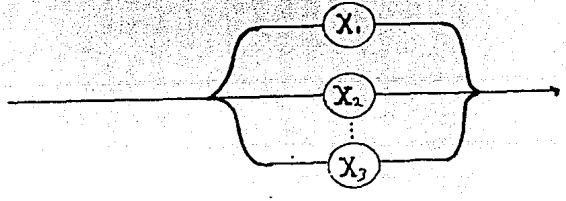
la dirección de la flecha indica el flujo del análisis.

2.- Cada aparición de un símbolo no terminal corresponde a- una activación de un analizador parcial. Esto se representa en- el diagrama con el símbolo no terminal encerrado por un cuadro:

3.- Para una producción de la forma:

B ::= x1 / x2 / ... / xn

se construye el grafo:



4.- Para una producción de la forma:

$$\langle A \rangle ::= \langle B \rangle \langle C \rangle \ldots \langle Z \rangle$$

se construye el grafo:

5.- Para una producción de la forma:

$$\langle A \rangle ::= \{ x \}$$

se construye el grafo:

Ejemplo, sea el lenguaje dado por:

A ::= x/ (B)

B ::= A C

C ::= { + A }

Los símbolos terminales son "+", "x", "(" y ")"; mientras que "{" y "}" pertenecen al BNF ampliado y son, por tanto, metasímbolos. El lenguaje que puede generarse a partir de A de expresiones como:

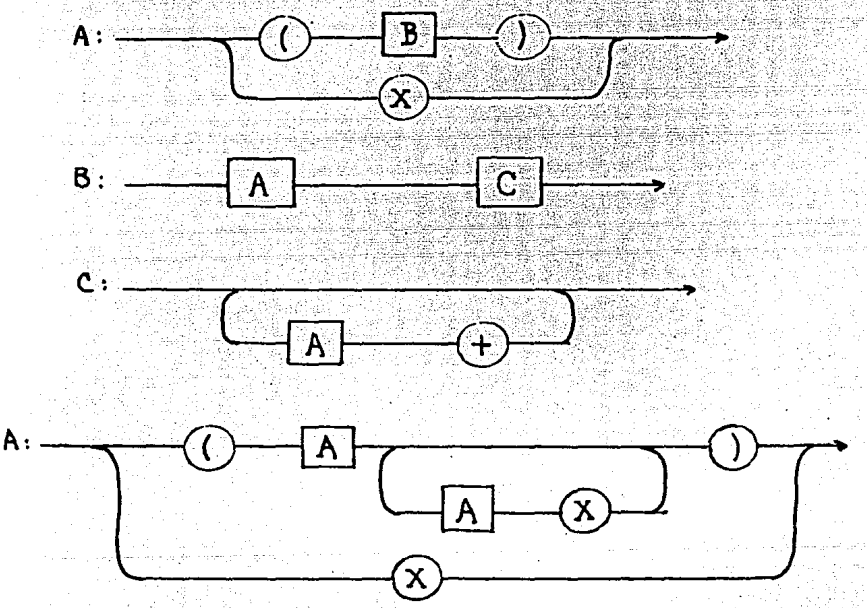
x

(x)

(x + x)

((x))

Aplicando las convenciones anteriores podemos obtener los siguientes diagramas, al combinarlos podemos obtener uno solo mediante sustituciones apropiadas:



El diagrama de Sintaxis es una representación equivalente de la grámatica del lenguaje; puede ser utilizado en lugar del conjunto de producciones BNF. Es una representación de la sintaxis muy apropiada y, en muchos casos, preferible a la BNF. Desde luego, el grafo presenta una imágen de la estructura del lenguaje más clara y concisa y, además, ayuda a entender mejor el -

proceso del análisis sintáctico.

Otro aspecto importante e interesante acerca del análisis de lenguajes es el tratamiento de errores sintácticos. Un analizador no debe unicamente determinar si una secuencia de símbolos de entrada pertenece o no al lenguaje. Como subproducto de este proceso el analizador debe descubrir también la estructura inherente en la frase y tan pronto como se detecte una frase mal - - construida, el analizador debe dar un mensaje de error apropiado y poder continuar el proceso de análisis, probablemente para encontrar otros errores. Solo se puede continuar haciendo algún supuesto sobre la naturaleza del error o saltando una parte de la secuencia de entrada, que venga a continuación, o haciendo ambas cosas.

b) Estructuras Dinámicas.

Algunos lenguajes de alto nivel, tienen la capacidad de manejar estructuras dinámicas (ref 6) aparte de las comunes estructuras estáticas, la manipulación de estas estructuras dinámicas proveen al lenguaje de la flexibilidad de optimizar los recursos, en cuanto a memoria se refiere, de la computadora en que están implementadas.

Las estructuras estáticas permiten el almacenamiento de datos que serán utilizados durante la ejecución de las instrucciones del lenguaje de alto nivel; la cantidad de memoria para este

almacenamiento, está definida en las mismas instrucciones del -- programa. De esta forma una vez definido el conjunto de instrucciones, que forman el programa, queda implícitamente definida el área de memoria de la computadora que va a ser utilizada cuando- las instrucciones sean ejecutadas.

Debe notarse que el número de datos, o la cantidad de información a procesar, no siempre está totalmente establecida, y en- algunos casos, la cantidad de datos puede tener grandes variaciones.

Ahora bien, si un conjunto de instrucciones, que manejan solo estructuras estáticas, limita implícitamente la memoria a -- utilizar y por tanto el número de datos a procesar, se podría -- pensar entonces que: para cada conjunto de estas instrucciones - se tiene un número máximo de datos que pueden ser procesados. ¿Qué sucedería con un conjunto de instrucciones, si el número de datos a procesar fuera mayor que el número que puede ser procesado?

La respuesta es evidente, dicho conjunto de instrucciones - deberá ser modificado, en este caso debe permitir un mayor número de datos a procesar.

Sin embargo, ¿Qué pasaría, si por el contrario, el número - de datos a procesar es considerablemente menor que el máximo nú-

mero que el conjunto de instrucciones previene?. En este caso se incurre en un desperdicio de memoria.

Por ejemplo, se tiene un conjunto de instrucciones, que utilizan estructuras estáticas, que permiten manipular un número de 10 000 datos de cierto tipo y usan para ello un determinado espacio en la memoria de la computadora; llegado el momento de procesar la información se observa que el número de datos a manejar - deben ser 20 000, sucede entonces que el conjunto de instrucciones debe ser modificado y el programador lo altera para manejar no 20 000 sino que, previniendo otro posible aumento, lo deja en 30 000. En una segunda ocasión, de proceso de información, se - observa que el número de datos son solo 5 000; aquí el programador ya no altera su conjunto de instrucciones, porque este satisface los requerimientos de memoria, pero esta inutilizando más - del 75% de la memoria que reservó, ésto es, claramente, una mala administración de recursos.

Es claro concluir que este es un problema de optimización. Es aquí, por tanto, que se vuelve evidente la necesidad de que - el conjunto de instrucciones no limite el número de datos a procesar, y sea este mismo número de datos, el que determine la cantidad de memoria a utilizar.

Para resolver esta necesidad se crean las ESTRUCTURAS DINAMICAS, sin embargo no todos los lenguajes de alto nivel poseen -

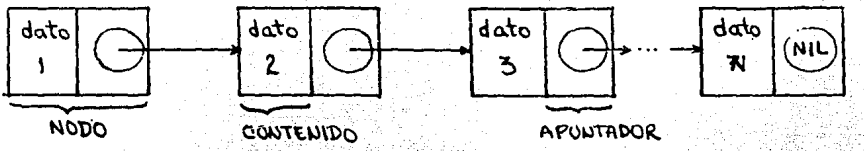
la capacidad de manejarlas. De esta forma un conjunto de ins- - trucciones o programa que maneja estructuras dinámicas puede ser totalmente invariante a pesar de que haya variaciones considerables en el número de datos.

En este caso, si el número de datos se incrementa, el con-- junto de instrucciones permite aumentar la disponibilidad de memoria utilizable, y por el contrario, si el número de datos es - mínimo, también será mínima la memoria utilizada para el procesamiento.

Las estructuras dinámicas básicas son las LISTAS y los ARBOLES. No se intentará realizar todo un tratado acerca de estas estructuras, que son merecedoras de un análisis no trivial; se mencionará solamente la conceptualización necesaria para el entendimiento de los capítulos destinados a la implantación del Siste-- ma, tema central de esta tesis (ref 6).

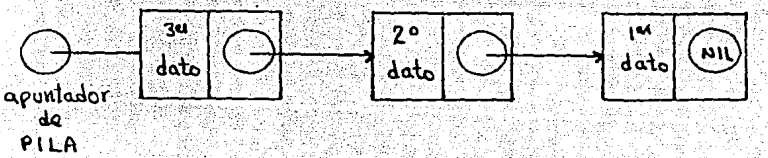
LISTAS.- Una lista es un conjunto de localidades de memoria, los elementos que forman esta lista se llaman NODOS, un nodo está formado por un CONTENIDO Y UN APUNTADOR, el contenido es el - dato a procesar y el apuntador (número de alguna localidad de - memoria), establece la relación entre el nodo actual y el próximo.

Esquemáticamente tenemos:

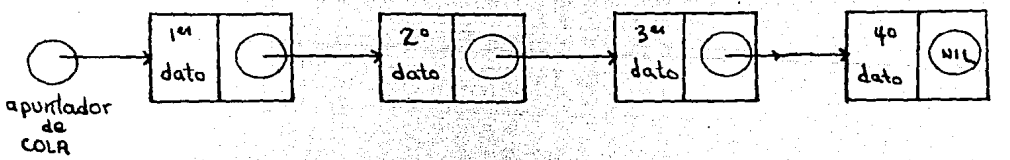


NIL, por definición, será el apuntador con el que designaremos la no existencia de nodo siguiente.

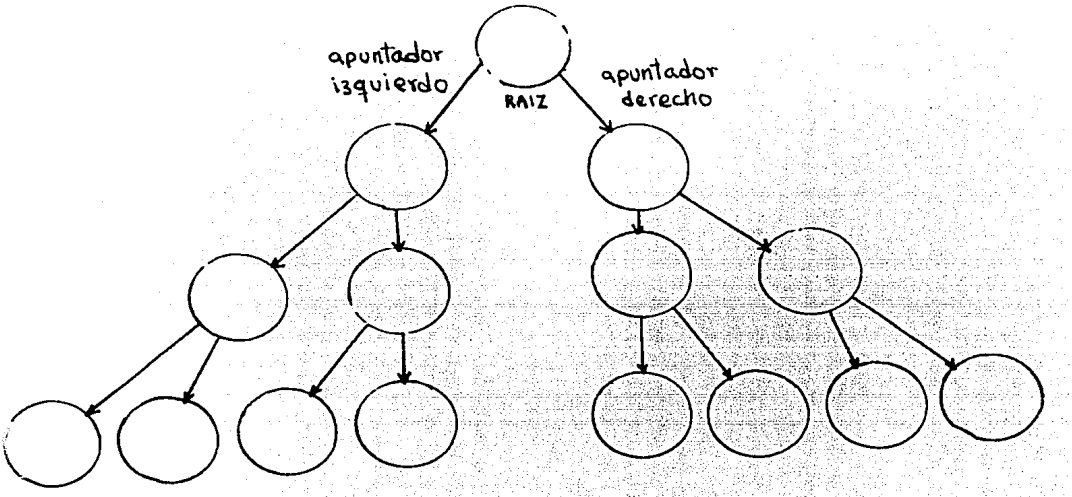
Las listas a su vez se subdividen en PILAS Y COLAS. Las pilas son listas que poseen un apuntador, APUNTADOR DE PILA:



Las COLAS son listas que poseen un apuntador, APUNTADOR DE-COLA que apunta al primer elemento insertado:



ARBOLES.- Son también un conjunto de localidades de memoria sus elementos son llamados nodos, un nodo está formado por una - RAIZ y por dos o más APUNTADORES que lo relacionan con igual número de nodos. En este caso la RAIZ contiene el dato a almace-- nar para su proceso posterior. Los árboles son representados:


apuntador
izquierdo
RAIZ
apuntador
derecho

III CRITERIOS PARA EL DISEÑO DE CIRCUITOS IMPRESOS.

El gran auge de las placas de circuitos impresos, se debe - principalmente, a la existencia de algunas ventajas frente a los antiguos métodos de conexiones mediante cables, las más importantes de estas ventajas son:

- Las conexiones entre los puntos quedan establecidas durante la elaboración del patrón de la placa, por tal motivo- son imposibles las conexiones erróneas entre elementos en el momento del montaje; lo anterior podía ocurrir antes - por falta de atención en los cableados.

- Ya diseñada la tableta y la ubicación de los componentes- no se precisa, para la colocación de los mismos, ningún - conocimiento sobre la técnica de circuitos o sobre su interpretación; con esto puede emplearse mano de obra no especializada.

- Posibilidades de automatización a mayor escala, al evitar el manejo de cables.

- Disminución en peso y volúmen, las pistas ocupan menos espacio que los cables y el peso es menor.

- Una consecuencia del punto anterior es el bajo costo, ya -- que el consumo de cable disminuye.

Para la fabricación de las tabletas para circuitos impresos, en general, se utilizan procesos de corrosión en una placa de material aislante recubierta de una capa de metal uniforme, imprimiendo previamente el diseño de las pistas conductoras.

a) Sistema de Malla.

La utilización de un sistema de malla es escencial para facilitar la ubicación de los componentes, la localización de las áreas terminales y el trazado de los conductores en la tableta.

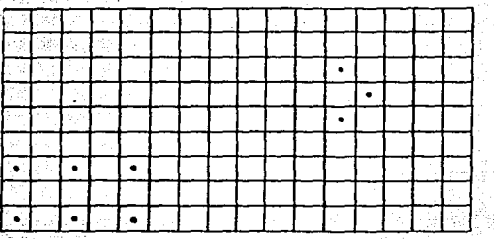
El sistema de malla es obligatorio si se realizara automatización en el diseño, fabricación y ensamblado del circuito impreso, porque las máquinas que realicen estas funciones estarán -- coordinadas al sistema de localización en la malla.

Una MALLA es una zona de trabajo bidimensional y rectangular que consiste de un grupo de líneas paralelas equidistantes, sobrepuestas a otro grupo de lineas paralelas equidistantes también, pero perpendiculares al otro grupo; las intersecciones delás lineas determinan la base para el sistema de localización incremental.

Los incrementos estandard comunmente utilizados son:

0.100 in(2.54 mm), 0.50 in (1.27 mm) y 0.025 in(0.635 mm)

Sistema de Malla:



En la actualidad el sistema de malla más utilizado es el de 0.100 in(2.54 mm), inclusive las tabletas de prueba comerciales- manejan un sistema similar. A cada una de estas celdas le llamaremos UNIDAD DE MALLA.

b) Escala.

Todos los diseños de circuitos impresos deben ser realizados en una escala aumentada; errores de dibujo, tales como áreas terminales desalineadas, respecto al sistema de malla, variaciones en el espaciamiento entre los conductores e imperfecciones en el ancho de los conductores, por los utensilios de dibujo, son disminuidos proporcionalmente con la reducción del diseño al tamaño final de la tableta.

Las escalas más utilizadas son generalmente:

2 : 1 , 4 : 1 y 1 : 1 en orden de preferencia.

La razón para la utilización de estas escalas, es la dispo-

nibilidad de elementos de ayuda para el ' Artwork ' y el diseño en estas escalas.

La escala 1 : 1 debe ser evitada cuando sea posible, ya que no tiene reducción en caso de error en el dibujo; esta escala solo debe ser utilizada cuando:

1.- No existe facilidad de reducción disponible.

2.- Para trabajo prototipo, donde la velocidad y costos son importantes.

3.- Cuando el patrón maestro es realizado con técnicas automatizadas que aseguran un error mínimo en el dibujo.

La elección de la escala 4 : 1 reduce grandemente los errores de dibujo y debe ser utilizada solamente cuando las dimensiones del circuito lo requieran y se tengan los medios necesarios para implementarlo.

Por las razones anteriores, la escala más común es la 2 : 1 ya que no tiene muchas dificultades para su implementación y si produce los requerimientos dimensionales suficientes.

c) Tamaño de la Tableta y Número de Placas.

La determinación del tamaño y el número de placas de un circuito impreso es la parte inicial más importante para el diseño.

Las tabletas para circuitos impresos pueden ser diseñados y fabricados en tres configuraciones básicas:

Placa sencilla contiene todas las pistas sobre una cara con los componentes en el lado opuesto.

Las tabletas multiplaca emparedada son un gran número de tabletas delgadas unidas, con los componentes en una de ellas o algunas veces en las dos placas exteriores. Una sola de estas tabletas puede tener hasta 20 placas.

Las tabletas multiplaca tienen las pistas en ambas caras.

Para la determinación del tamaño y el número de placas, la consideración básica es el costo de la fabricación y del diseño, aunque también es importante la magnitud del circuito electrónico a montar en la tableta y de su complejidad.

Por lo general una tableta multiplaca emparedada es menos cara que muchas tabletas de placa sencilla.

Mecanicamente las tabletas grandes tienen una gran tendencia a torcerse, sin embargo las pruebas se dificultan en una tableta pequeña.

Las tabletas multiplaca requieren, generalmente, un menor -

tiempo de diseño que las multiplaca emperedadas y las de placa - sencilla.

El tamaño de la tableta, idealmente, debe ser lo suficientemente grande para contener los componentes y las interconexiones y además ser económicamente producible.

Un método para estimar el tamaño final de la tableta es el- calcular el área requerida para cada tipo de componente, multi-- plicarla por el número de componentes similares; para circuitos- integrados se debe considerar un área adicional por la gran cantidad de circuitería asociada requerida (esto es generalmente - estimado de diseños previos), una vez calculadas las áreas para cada tipo sumarlas y dividirlas por el área de diseño en un tamaño propuesto de tableta. No se debe incluir el área a ser utilizada por los conectores, bordes de la tableta, hardware montado, etc.

Si el resultado de este cálculo, en porciento, es del 80% o menor indica, por lo general, que el diseño es posible; si el índice está entre 80% y 90% pueden ser necesarias otras consideraciones. Pero si el índice es mayor de 90% debe ser utilizado -- otro tamaño de tableta más grande, o el circuito debe ser dividido en pequeñas tabletas.

e) Ancho de conductor y Espaciamiento.

Se deben tomar consideraciones cuidadosas en el ancho de -- los conductores y el espaciamiento cuando se diseña un circuito- impreso, si el conductor es muy angosto puede ocasionar sobreca- lentamientos en la tableta final y dañar físicamente a los ele-- mentos.

Por otro lado, espaciamientos muy estrechos pueden ocasio-- nar corto circuito. Estos dos parámetros podrían llegar a ser - excesivamente grandes y producir altos costos de fabricación y - espacio inutilizable en la tableta.

La anchura de los conductores y su grosor debe ser determi- nado en base a la corriente que circulará por dicho conductor.

Deben evitarse conductores más anchos de 0.500 in (12.7 *mm*) para prevenir ampollamiento o alabeo durante el proceso de sol-- dar los componentes.

Un conductor nominal con ancho de 0.050 in (1.27 mm) es re- comendado para aplicaciones de bajo voltaje.

El espaciamiento entre conductores debe ser determinado con siderando el voltaje pico entre conductores, la altitud a la que el circuito impreso trabajará y el tipo de revestimiento que se-

rá aplicado a la tableta. Un espaciamiento nominal de 0.031 - - in (.79 mm o de 0.050 in (1.27 mm) es recomendado para aplicaciones de bajo voltaje.

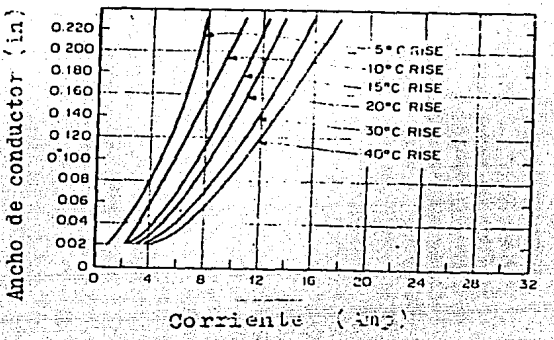
El espaciamiento nominal entre conductores borde de la tableta o de un componente montado debe ser de 0.100 in (2.54 mm) a 0.250 in (6.35 mm).

La figura y tabla siguientes utilizadas para la determinación del ancho de conductor y espaciamiento:

MIL-STD-275 B Separación entre conductores según las condiciones de servicio.

Para conductores en placas que trabajaran del nivel del mar a 10 000 pies.

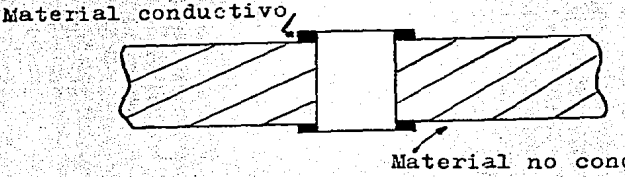
Tensión entre conductores (Volts cc o de pico)	Separación mínima (in) Calidad A	Calidad B
0-50	0.015	0.080
51-150	0.026	0.080
151-300	0.062	0.125
301-500	0.125	0.300
Más de 500	0.0003por volt	0.0006 pv



e) Agujeros para terminales

Existen dos tipos de agujeros para terminales comunmente -- utilizados en los circuitos impresos, agujeros no laminados y -- agujeros laminados.

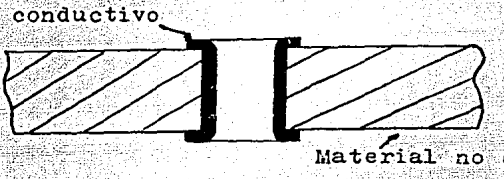
Los agujeros no laminados tienen material no conductivo y - pueden ser implementados solamente taladrando la tableta, una -- vista lateral de la tableta en este caso es:



Los agujeros laminados tienen material conductivo en su interior y su implementación empieza como la de los no laminados,- pero después se deposita material conductivo en su interior, en- las paredes del agujero, formando así conexión eléctrica en am--

bas caras de la tableta (como se puede observar, éstos agujeros son ideales cuando se utilicen tabletas multiplaca).

Una vista lateral de la tableta en este caso es:



Para determinar el diámetro mínimo de un agujero no laminado se puede tomar por norma que este no debe ser más grande que 0.02 in (0.51 mm) que la terminal a insertar; para los laminados este diámetro no debe ser más grande que 0.028 in (0.71 mm) que la terminal a insertar en el agujero.

El número de tamaños diferentes de agujeros en un circuito impreso debe ser reducido al mínimo. En la medida que este número se incrementa, también se incrementan los costos y las dificultades para la producción de las tabletas. Generalmente un número entre 3 y 5 (inclusive) de tamaños de agujeros son suficientes para acomodar todos los componentes.

El espaciamiento entre agujeros debe ser tal que las áreas terminales que los rodean cubran los requerimientos mínimos para espaciamientos entre conductores.

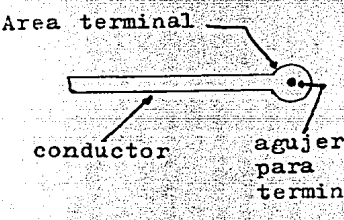
El espacio entre agujeros debe ser igual o mayor al grosor-

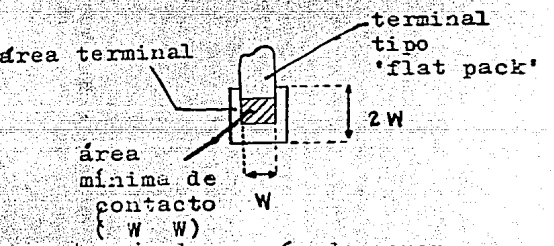
de la tableta.

F) Areas Terminales.

El área terminal es una porción del circuito impreso utilizado para realizar la conexión eléctrica entre la terminal de un componente y la pista conductora del circuito eléctrico.

Debe haber una área terminal separada para cada terminal de cada componente. Generalmente estas áreas deben rodear completamente a los agujeros donde serán montados los componentes, una excepción a esta regla es cuando se utilizan componentes de tipo "flat pack" cuyas terminales están montadas en la superficie de la tableta.





El contorno o forma de las áreas terminales varía de acuerdo a la preferencia del diseñador y de acuerdo a las ventajas o desventajas que presenta cada forma.

Formas cuadriláteras permiten una adhesión mayor a la tableta, estas formas son muy utilizadas cuando se tiene un agujero para una terminal. Sin embargo cuando este tipo de áreas son co

locadas en la proximidad con conductores o con otras áreas terminales, pueden contribuir a problemas de puenteo durante la implementación de la soldadura. Formas redondas o elípticas reducen el problema anterior en las mismas circunstancias.

Algunos diseñadores prefieren el uso de formas en "lágrima" este tipo de forma provee de una conexión mecánica más fuerte. La forma de "lágrima" resulta un tanto difícil de implementar en el dibujo del diseño, por el alineamiento y además porque generalmente se tienen que utilizar rebordes complementarios.

Otro factor, es el aumento de la densidad de conductor utilizado en la tableta, al usar formas en "lágrima". Por estas razones la mayoría de los diseñadores consideran innecesario el uso de esta forma como área terminal.

A continuación se muestran las formas más comunes de Areas-Terminales:

Redonda

Cuadrilátera

Elíptica

'lágrima'

El tamaño del Area Terminal esta determinada según los siguientes criterios:

A = Tamaño máximo del agujero.

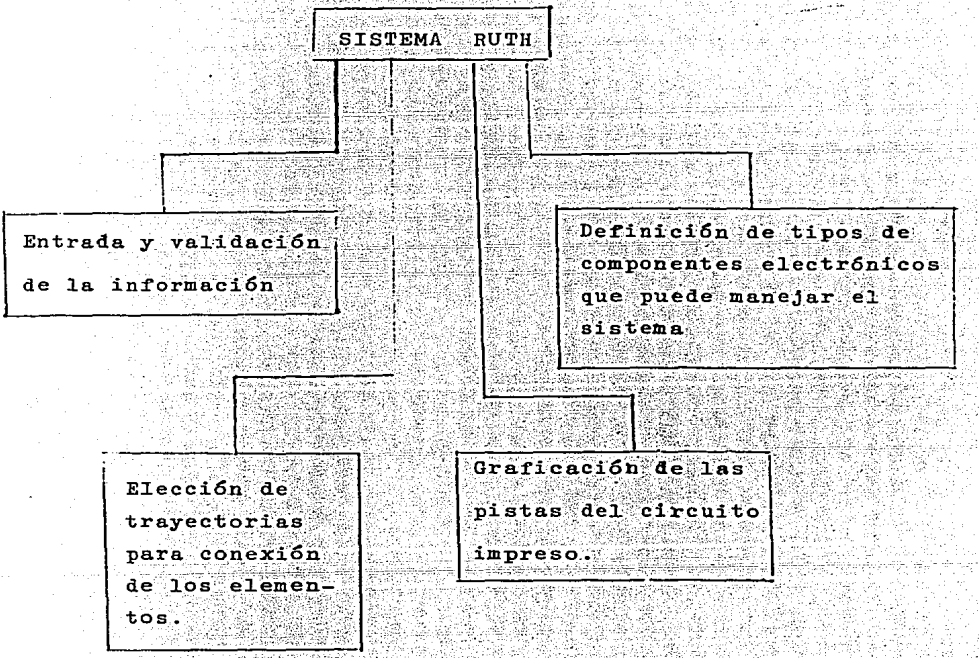
AC = Dimensión del anillo concéntrico (parte que rodea al agujero)

Y se calcula según :

$$\text{Area Terminal} = A + 2AC$$

La norma MIL-STD-275D especifica 0.015 in (0.38 mm) para el mínimo anillo concéntrico en agujeros no laminados y para agujeros laminados de 0.005 in (0.13 mm) a 0.002 in (0.05 mm).

IV DISEÑO DEL SISTEMA



La figura anterior, demuestra la configuración lógica del Sistema, se puede observar que, el Sistema RUTH está formado en base a cuatro procesos.

A continuación se desarrollarán en pseudocódigo cada uno de los procesos anteriores. (El pseudocódigo es una representación del algoritmo a seguir: es un sustituto, utilizado en los lenguajes estructurados, de los conocidos "diagramas de flujo").

```
INICIO PROCESO 1
ENTRADA DEL ARCHIVO DE DATOS
ANALISIS SINTACTICO Y SEMANTICO
SI EL ANALISIS FUE CORRECTO ENTONCES:
                    EMPEZAR BLOQUE_A
                    GENERAR UNA TABLA DE DATOS
                    DESPLEGAR RESULTADOS DE LA VALIDACION
                    DISPONIBLE A PROCESO 2
                    FIN BLOQUE_A
          SINO:
          EMPEZAR BLOQUE_B
          CREAR UN ARCHIVO DE ERRORES LISTABLE
          DESPLEGAR RESULTADO DE LA VALIDACION
          NO DISPONIBLE A PROCESO_2
          FIN BLOQUE_B
FIN PROCESO_1
```

La entrada de datos no es más que una lectura secuencial de los registros del Archivo; el análisis Sintáctico y Semántico será desarrollado a continuación; la Generación de la tabla de da-

tos y del Archivo de errores son solamente escrituras, también - secuenciales, en archivos; la disponibilidad a proceso_2 es la - asignación de FALSO o VERDADERO a una 'bandera' para saber si se - ejecutará o no el proceso_2.

INICIO ANALISIS_SINTACTICO_SEMANTICO
ANALISIS DE DIMENSIONES DE LA TABLETA
ANALISIS DE LISTA DE ZONAS NO PERMITIDAS PARA TRAYECTORIAS
ANALISIS DE ESPACIAMIENTO ENTRE PISTAS
ANALISIS DE NUMERO DE CARAS A UTILIZAR EN EL DISEÑO
ANALISIS DE LISTA DE DESCRIPTORES
ANALISIS DE LISTA DE CONEXIONES
FIN DE ANALISIS_SINTACTICO_SEMANTICO.

El análisis consiste en verificar que el archivo de datos - que se lee sea congruente, sintacticamente, con los diagramas -- del pequeño lenguaje definido para establecer la comunicación entre USUARIO Y SISTEMA.

Observese que la técnica utilizada para el tratamiento de - errores, es la de continuar con el análisis sin cuestionar si el anterior paso generó errores de sintaxis.

El archivo de datos contiene información sobre la definición de la tableta, definición de elementos a interconectar (descriptores) y definición de conexiones entre estos elementos, ca-

la una de estas definiciones es analizada sintáctica y semántica mente.

Este análisis se realiza extrayendo carácter por carácter - de cada uno de los registros, hasta formar símbolos, símbolos -- que son comparados con los definidos en los diagramas de sinta-- xis.

En este caso los análisis de: Dimensiones de la tableta, de la lista de zonas no permitidas para trayectorias, de espacia- - miento entre pistas y del número de caras a utilizar en el dise- ño, que pertenecen a la definición de la tableta, son relativa-- mente sencillos según la técnica anterior (análisis por símbo-- los).

Para el caso del ANALISIS DE LISTA DE DESCRIPTORES, se de-- ben realizar ciertas consideraciones importantes:

En esta parte, el usuario define los componentes electróni- cos que va a utilizar en el circuito impreso, así como su posi-- ción en la tableta; la definición de estos componentes debe ser- fundamentada en los tipos de elementos que el sistema tiene ya - definidos o bien en los que el usuario define previamente en el- proceso_4. Estos tipos de elementos son almacenados en el archi vo, de tal manera que cuando el usuario hace referencia a ellos- en la lista de descriptores se realiza una búsqueda en el archi-

vo mencionado para establecer las características físicas del -- componente electrónico que va a ser utilizado.

De esta forma se obtiene comodidad para el diseñador, ya -- que solo tiene que indicar el nombre del elemento y su posición- en la tableta, de otro modo tendría que especificar su tamaño, - número de pins, número de niveles, separación entre pins para ca da nivel, etc. El pseudocódigo para este análisis es:

```
INICIO ANALISI_LISTA_DE_DESCRIPTORES
HACER HASTA TERMINAR CON LA LISTA DE DESCRIPTORES:
  EMPEZAR BLOQUE_A
  LEER A UN DESCRIPTOR
  ANALISIS SINTACTICO
  SI EL ANALISIS FUE CORRECTO ENTONCES:
                    EMPEZAR BLOQUE_AA
                    BUSCAR ELEMENTO EN ARCHIVO DE TIPOS
                    SI ELEMENTO NO EXISTE ENTONCES:
                                  PONER BANDERA DE ERROR
                     SINO:
                     EMPEZAR BLOQUE_AAA
                     VALIDAR POSICIONES DE LOS PINS
                     CREAR ESTRUCTURA QUE CONTENGA___
                     __INFORMACION DEL DESCRIPTOR
                     FIN BLOQUE_AAA
```

```
                FIN BLOQUE_AA
            SINO:
            PONER BANDERA DE ERROR
        FIN BLOQUE_A
    FIN ANALISIS_LISTA_DE_DESCRIPTORES
```

Una vez que el elemento fue encontrado en el archivo de tipos, se extrae la información de su registro, que permite determinar el número de pins y la localización precisa en la tableta de cada uno de ellos, almacenándose ésto en una estructura interna.

Ya realizado el análisis de la lista de descriptores, empieza el análisis en la ista de conexiones, ésta indica los pins que deben ser interconectados y los elementos que les corresponden, así como el tipo de conductor que debe ser utilizado para establecer la conexión, este análisis es como sigue:

```
INICIO ANALISIS_LISTA_DE_CONEXIONES
HACER HASTA TERMINAR CON LA LISTA DE CONEXIONES:
      EMPEZAR BLOQUE_A
      LEER UNA CONEXION
      HACER HASTA LEER TODOS LOS CONECTORES:
            EMPEZAR BLOQUE_AA
            ANALISIS SINTACTICO
            SI EL ANALISIS FUE CORRECTO ENTONCES:
                        EMPEZAR BLOQUE_AAA
                        BUSCAR CONECTOR EN ESTRUCTURA/__
                        _INTERNA
                        SI CONECTOR NO EXISTE ENTONCES:
                                    PONER BANDERA DE ERROR
                          SINO:
                          EMPEZAR BLOQUE_B
                          ESCRIBIR EN TABLA DE DATOS LAS__
                          __COORDENADAS DEL PIN A INTER___
                          __CONECTAR
                          FIN BLOQUE_B
                        FIN BLOQUE_AAA
                  SINO:
                  PONER BANDERA DE ERROR
            FIN BLOQUE_AA
      TOMAR NUEVO REGISTRO EN TABLA DE DATOS
      FIN BLOQUE_A
      FIN ANALISIS_LISTA_DE_CONEXIONES
```

La lista de conexiones establece el nombre del elemento a - interconectar y el número del pin del mismo, a ésto se le llama- CONECTOR de tal manera, una conexión existe si se tienen al me-- nos dos conectores (orígen y destino), pero puede darse el caso- de que se tengan más de dos conectores, se hablaría entonces de- una INTERCONEXION entre pins.

El análisis anterior va transformando esos números de pins- en coordenadas en unidades de malla, por tanto al final del análisis se tendrán unicamente puntos en la tableta en coordenadas- rectangulares, que deberán ser unidos.

Finalmente el proceso_1 dirige el control al proceso_2, unicamente si no fueron detectados errores en la sintaxis o semántica del archivo de datos.

b) Elección de trayectorias

Ahora se desarrollará el proceso_2, que es el más complicado por los algoritmos utilizados; este proceso "elige" las trayectorias que han de seguir los conductores para establecer las conexiones entre los elementos.

```
INICIO PROCESO_2
ENTRADA DE LA TABLA DE DATOS GENERADA EN PROCESO_1
DETERMINACION DE LA LISTA DE CONEXIONES
SI EL NUMERO DE CARAS DE DISEÑO ES DOS ENTONCES:
                    ELEGIR  CONEXIONES PARA CADA CARA
HACER HASTA EL NUMERO DE CARAS:
      EMPEZAR BLOQUE A
      ORDENAR LISTA DE CONEXIONES
      CREACION DE MALLA SEGUN DIMENSIONES
      UBICAR PUNTOS EN LA MALLA
      CANCELACION DE ZONAS NO PERMITIDAS
      UNION DE PUNTOS
      SI FUERON POSIBLES TODAS LAS CONEXIONES ENTONCES:
                              GENERAR ARCHIVO RESULTADO
                SINO:
                INDICAR QUE CONEXIONES NO PUDIERON REALIZARSE
      FIN BLOQUE_A
FIN PROCESO_2
```

En este proceso se utiliza el archivo de datos generado en-

el proceso_1 que contiene toda la información ya validada.

La determinación de la lista de conexiones consiste en definir solamente un par de coordenadas para ser conectadas, ya que- como se recordará una conexión puede tener dos o más conectores.

Para la determinación de esta lista partiremos del hecho de que para interconectar N pins se necesitan (N-1) conexiones. Se trata entonces de establecer (N - 1) parejas de pins, de -- tal forma que involucren un recorrido mínimo.

En algoritmo en pseudocódigo será el siguiente:

```
INICIO DETERMINACION_LISTA_DE_CONEXIONES
HACER HASTA TERMINAR CON LA LISTA DE CONEXIONES:
      EMPEZAR BLOQUE_A
      ESTABLECER LAS COORDENADAS DE LOS N PINS
      HACER PARA I= 1 HASTA
            EMPEZAR BLOQUE_B
            SEA PIN DE REFERENCIA = PIN I
            CALCULAR LAS PAREJAS A CONECTAR SEGUN LA__
            __DISTANCIA MAS CORTA
              EVALUAR UNA DISTANCIA I TOTAL
              FIN BLOQUE_B
      LA DISTANCIA TOTAL MAS CORTA DETERMINARA LA LISTA_
      __DE PAREJAS A CONECTAR
```

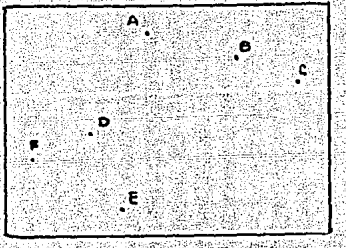
ESCRIBIR LAS PAREJAS DE PUNTOS EN UN NUEVO ARCHIVO

FIN BLOQUE_A

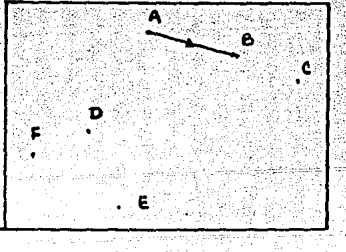
FIN DETERMINACION_LISTA_DE_CONEXIONES

Para que la solución del problema sea más clara, se ejemplificará el método utilizado.

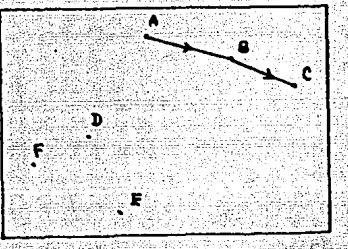
Sean los siguientes puntos a interconectar, con sus respectivas coordenadas rectangulares:



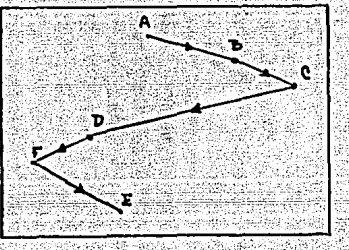
Primero tómese como punto de orígen al punto A, calcúlese ahora la distancia entre A y los 5 puntos restantes, sea el punto B el de menor distancia a A, entonces se establece conexión entre ellos.



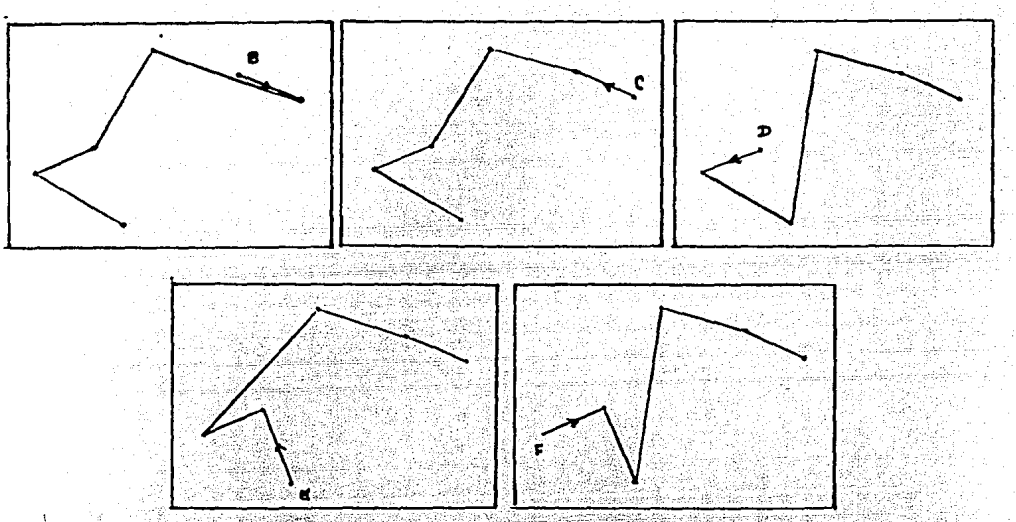
Tómese ahora el punto B como orígen y calcúlese la distancia entre B y los 4 puntos restantes, sea el punto C el de menor distancia a B, entonces se establece conexión entre ellos:



Tómese ahora el punto C como orígen y ... finalmente se tendría una interconexión como la siguiente:



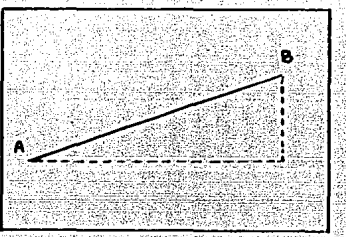
Sin embargo obsérvese que se puede tener 5 configuraciones diferentes dependiendo del pin que se tome como referencia inicial, en este caso fue el punto A. Las configuraciones restantes son:



Para cada configuración se calculará la distancia total recorrida. La configuración elegida será la que involucre una distancia menor.

En el caso de que se utilicen dos caras en el diseño se aplicará un algoritmo para establecer que conexiones deben realizarse en una cara y cuales en la otra, este algoritmo se fundamentará en base a las conexiones "verticales" y "horizontales", las conexiones "verticales" serán las que contengan un mayor número de unidades de malla en el eje 'y' y las "horizontales" en el eje 'x'

Por ejemplo, para unir los puntos A (1,1) y B (5,3)



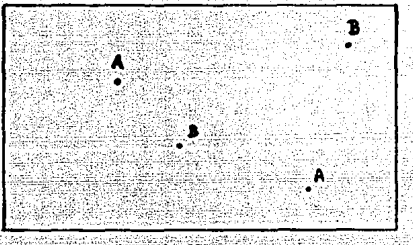
Se necesitarán según el triángulo rectángulo 4 unidades en- el eje 'x' y 2 en el eje 'y', por lo tanto esta conexión será -- considerada como "horizontal".

Para evitar el cruce de conductores en el diseño final, se- establece que las conexiones "verticales" deberán ser colocadas- en una cara de la tableta y las "horizontales" en la otra.

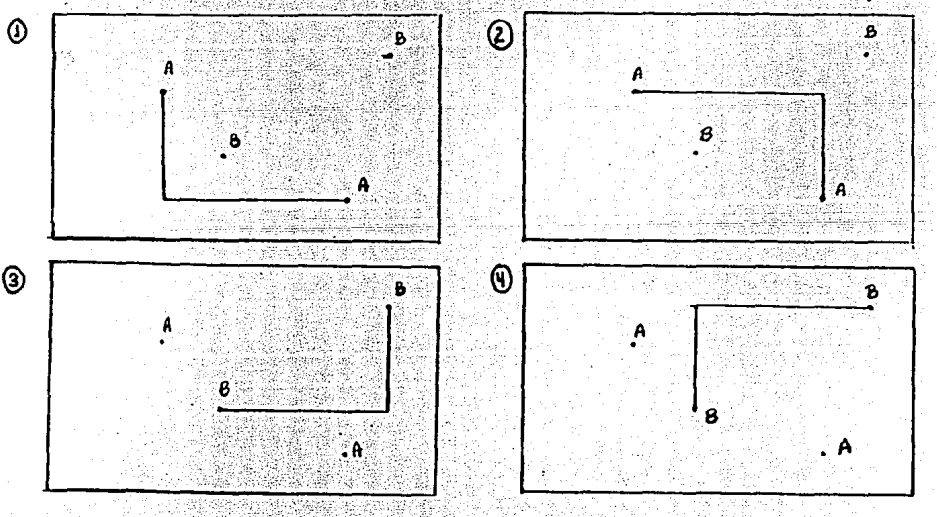
Una vez determinada la lista de conexiones (o las listas - de conexiones en caso de utilizar dos caras de la tableta) se -- realizará un ordenamiento de esta lista con la finalidad de de-- terminar exactamente "cuándo" debe ser analizada la unión, ésto- es, en que orden deben ser procesadas cada una de las conexio- - nes.

La siguiente figura ilustrará la importancia de esta pregun ta:

Supóngase que deben realizarse dos conexiones, una entre el par de pins "A" y la otra entre el par de pins "B":



Como se puede observar, existen dos posibles opciones para interconectar a cada pareja de puntos:



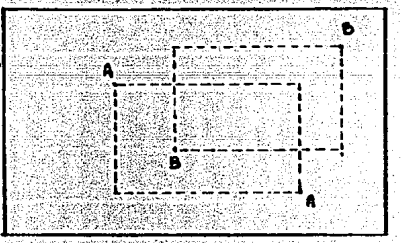
Ahora surge una pregunta: ¿Qué opciones ocasionarán dificultad al unir el otro par de puntos?

Una breve inspección de los cuatro casos, muestra que en la figura 2, al unir primero los puntos "A", la trayectoria para -- unir los puntos "B" ya no será óptima, ésto es, ninguna de las - opciones 3 y 4 para unir los puntos "B", puede ser realizada.

Por lo tanto en esta situación, debe decidirse por unir primero los puntos "B", asegurando con ésto, al menos, una trayectoria óptima posible para unir los puntos "A".

Analizando este caso, se podría establecer la siguiente regla:

Si un punto de "B" aparece en el rectángulo "A" (esto es,- el rectángulo que tiene a los puntos "A" como esquinas opues- - tas), entonces la conexión de los puntos "B" debe hacerse primero.



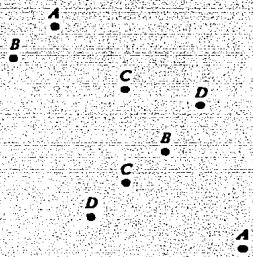
Generalizando la regla para aplicarla a el caso en estudio:

Sean N conexiones a realizarse, cada una entre un par de -- puntos, el número de prioridad de la conexión I será igual al nú

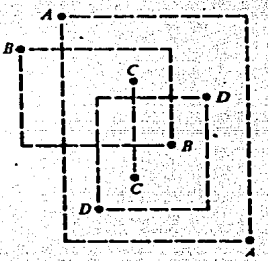
mero de puntos que se encuentren dentro del rectángulo I.

Para ejemplificar el uso de esta regla supóngase el si- - - guiente caso:

Considérense los siguientes 4 puntos:



Dibujando ahora los rectángulos para cada punto:



Obsérvese que, según la regla establecida:

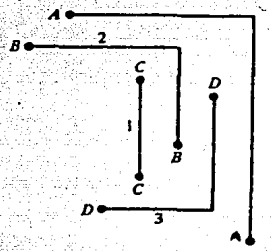
prioridad para A = 5 puntos inscritos

" " B = 1 " "

" " C = 0 " "

" " D = 2 " "

Por lo tanto, el orden en que deben ser conectados los pa-- res de puntos es: C, B, D, A; que produce como resultado:



Si llegase a obtenerse una misma prioridad para dos o más - conexiones, se considerará, para fines del ordenamiento la distancia geométrica que existe entre cada par de puntos; uniendo- primero los pares de puntos que involucren la distancia más corta.

Resumiendo, el ordenamiento de la lista de conexiones se -- explica mediante el siguiente algoritmo en pseudocódigo:

```
INICIO ORDENAMIENTO_LISTA_DE_CONEXIONES
HACER PARA TODA LA LISTA DE CONEXIONES
      EMPEZAR BLOQUE_A
      LEER COORDENADAS DE LA PAREJA DE PUNTOS
      ESTABLECER EL RECTANGULO DEL PAR DE PUNTOS
      DETERMINAR ANALITICAMENTE SI EL RESTO DE LAS COORDENA__
      __DAS ESTA INSCRITA EN ESE RECTANGULO
      CALCULAR LA DISTANCIA ENTRE EL PAR DE PUNTOS
      FIN BLOQUE_A
```

EFECTUAR UN ORDENAMIENTO SEGUN EL NUMERO DE PUNTOS INSCRI__
__TOS EN CADA RECTANGULO Y SEGUN LA DISTANCIA
REESCRITURA DE LA LISTA DE CONEXIONES EN UN NUEVO ARCHIVO
FIN ORDENAMIENTO_LISTA_DE_CONEXIONES

Para establecer el rectángulo en base al par de puntos, solo hay que considerar los valores máximos y mínimos en cada uno de los ejes, ésto es:

Sea el par de puntos A1 (1,4) y A2 (5,2), donde:

xmín=1 xmáx=5 ymín=2 ymáx=4

Para determinar si un punto B de coordenadas (x,y) esta inscrito en el rectángulo solo se tiene que realizar la siguiente consideración:

B (x,y) estará inscrito si y solo si:

$xmín \leq x \leq xmáx$ y además: $ymin \leq y \leq y\ max$

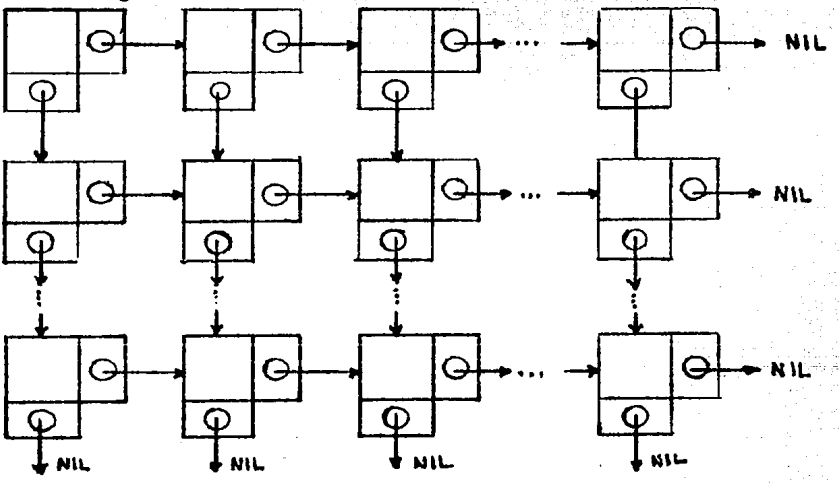
(nótese que se considera inscrito aún cuando el punto se encuentre en el perímetro del rectángulo).

Para el ordenamiento de la lista, ya establecida la prioridad, se utilizará el método de la burbuja (ref. 7).

Una vez ordenada la lista de conexiones, se procederá a la-

"creación de la malla"; ésto consiste en la creación de una estructura dinámica que simulará una matriz; en esta matriz serán ubicados todos los pins del circuito impreso y además las zonas no permitidas para trayectorias en la tableta.

La estructura dinámica que simulará una matriz tiene la siguiente configuración:



Como se observa, la estructura dinámica esta formada por -- NODOS que tienen un CONTENIDO Y 2 APUNTADORES.

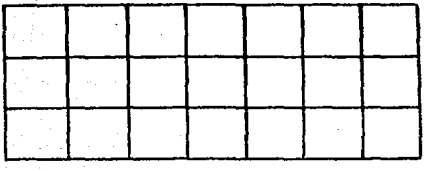
Cada nodo simbolizará una unidad de malla y si se toma en - cuenta que, tanto las coordenadas cartesianas de los pins y de- las zonas no permitidas para trayectorias, así como las dimen-- siones de la tableta, están dadas en unidades de malla, podemos concluir que la estructura dinámica creada puede representar fá

cilmente la tableta donde se diseñará el circuito impreso. (El lector debe considerar que, por ejemplo: para manipular una tableta de 10 x 20 cm^2. tomando en cuenta que una unidad de malla tiene 2.54 x 2.54 mm^2. de superficie, se necesitará una matriz de aproximadamente 394 renglones por 788 columnas).

Fundamentalmente, el CONTENIDO de cada NODO tendrá 2 estados: vacio y ocupado, ocupado significará que por esa unidad de malla pasa un conductor para unir dos pins o que está alojado un pin en esa unidad de malla (u.m.) o se encuentra en una zona no permitida para conexiones; vacio significará que la unidad de malla está disponible para que una posible trayectoria, que interconecta dos pins, la ocupe.

Los dos apuntadores del nodo apuntan a los nodos mas próximos: derecha y abajo.

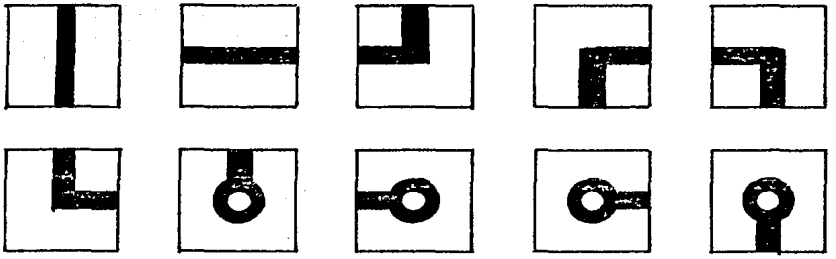
Dado que el esquema de la estructura dinámica, a utilizar como malla, tiene un dibujo un tanto complicado, utilizaremos la siguiente representación para manipularla fácilmente:



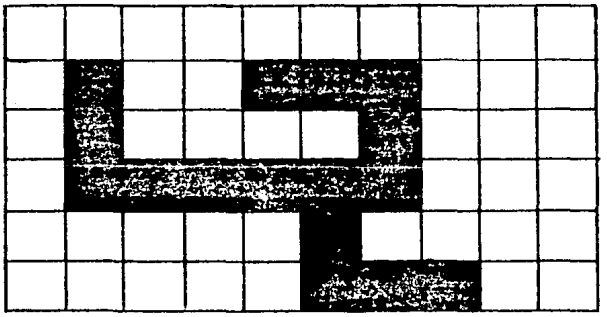
Malla para simbolizar una tableta de 7x3 unidades.

La ubicación de puntos o pins en la malla y de las zonas no permitidas por trayectorias, se hace simplemente asignándole a- las celdas correspondientes (según las coordenadas cartesia- - nas) el estado de ocupado. Representaremos el estado de OCUPADO rellenando con oscuro la celda.

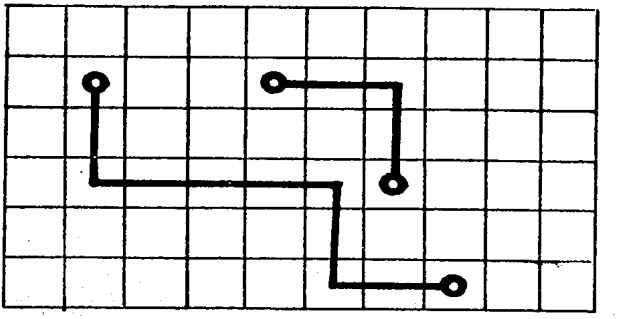
Toda celda ocupada será llamada unidad de interconexión, esta, físicamente, puede tener las siguientes características:



De tal forma, la siguiente malla ejemplo:



tiene la siguiente configuración en unidades de interconexión:

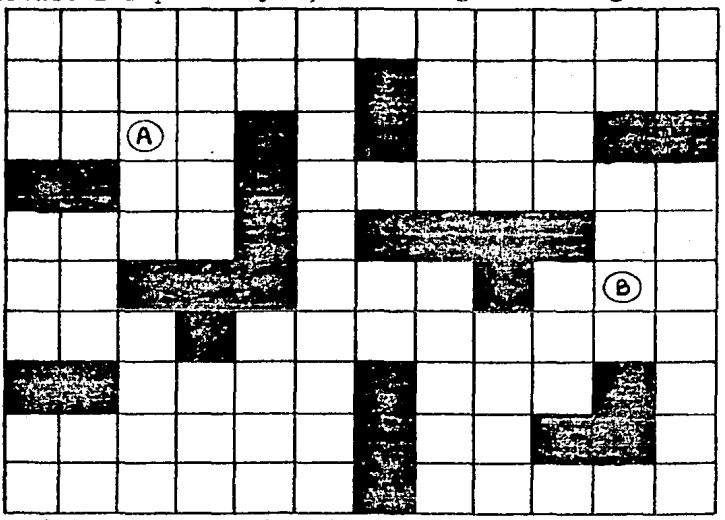


Sigue ahora la parte más importante e interesante de este tema, la elección de las trayectorias que deberán seguir los conductores para interconectar dos pins; estas trayectorias, como debe suponerse, deben ser óptimas, ésto es, en general para unir par de puntos pueden llegar a existir un buen número de posibles caminos, sin embargo, el número se reduce cuando se considere el camino más corto. Aún con ésto, pueden llegar a existir dos o más caminos igual de cortos, en este caso se considerará el camino que implique menos problemas posteriores al intentar las conexiones de los siguientes pares de puntos.

Para lo anterior, se utilizó el llamado "Algoritmo de Lee", pero modificado para adaptarlo a la aplicación que se pretende dar en este tema.

A continuación se describirá dicho algoritmo:

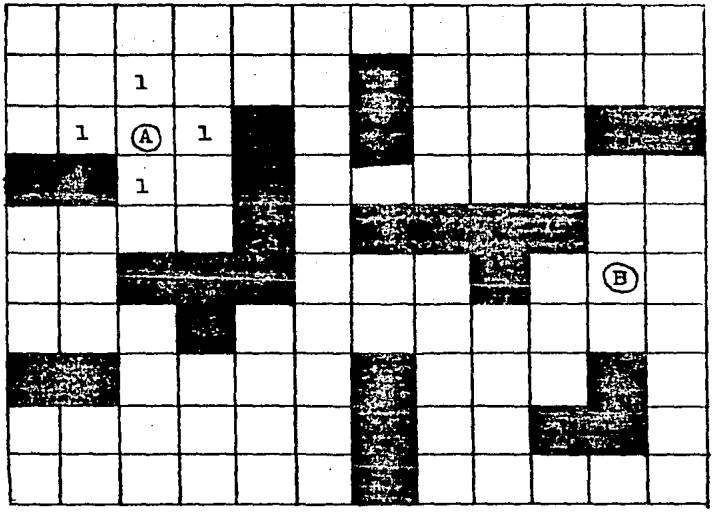
Considérense los pins A y B, en la siguiente figura:



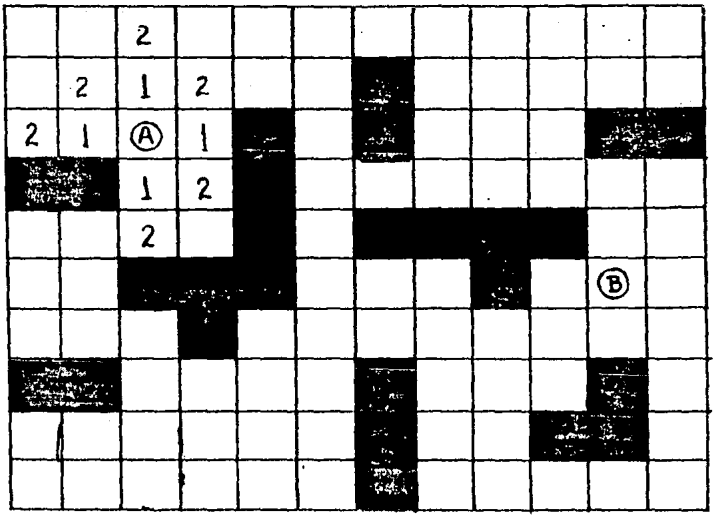
¿Cuál sería la trayectoria más corta para unirlos?, es única?

Como el lector observa, el ejemplo es un caso hipotético de una posible tableta con celdas que ya se encuentran ocupadas.

El proceso empieza con la elección de uno de los pins como inicio, tomemos en esté caso el pin A (esta elección no es relevante); inicialmente se coloca un "1" en cada celda VACIA inmediatamente adyacente a la celda que contiene al pin A, ésto es:

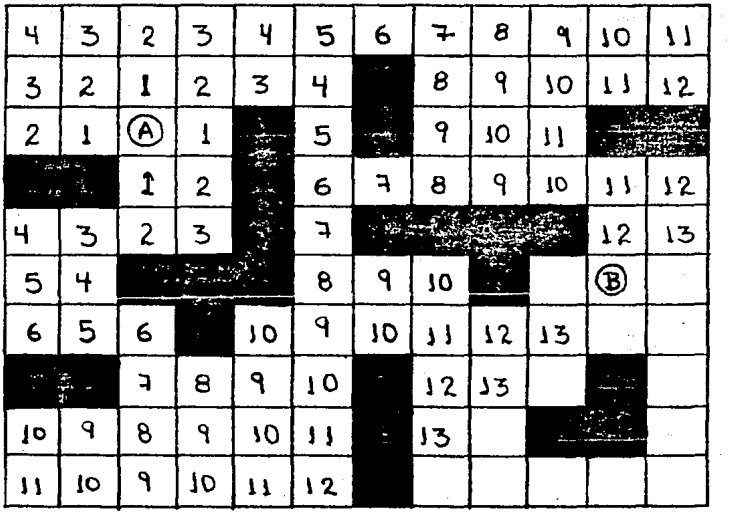


En seguida se coloca un "2" en todas las celdas vacias adyacentes a esas que contienen el "1":



Después se coloca un "3" en las adyacentes a los "2"s, y -- así sucesivamente. El proceso continua en esta manera hasta -- que uno de dos resultados ocurre:

Que el camino "se bloquee", ésto es, que en K-ésimo paso no existan celdas adyacentes vacias a esas que contengan el número (k - 1), 6, como en este caso, que se llegue a la celda que - contiene al punto B. Esto sucede en el paso número "13", como- se muestra en la figura:



Inmediatamente de ésto, dos factores son ya conocidos:

Existe un camino desde A hasta B, el camino más corto entre estos dos puntos tienen una longitud de 13 celdas. Todo lo que

resta, es entonces, encontrar este camino y "etiquetar" las celdas con el estado de ocupado.

Como se puede observar el algoritmo tiene la siguiente lógica: Se toma como punto de partida uno de los puntos. El "1" colocado en la celda indica que existen inicialmente N caminos posibles donde N es el número de celdas donde se puso el "1".

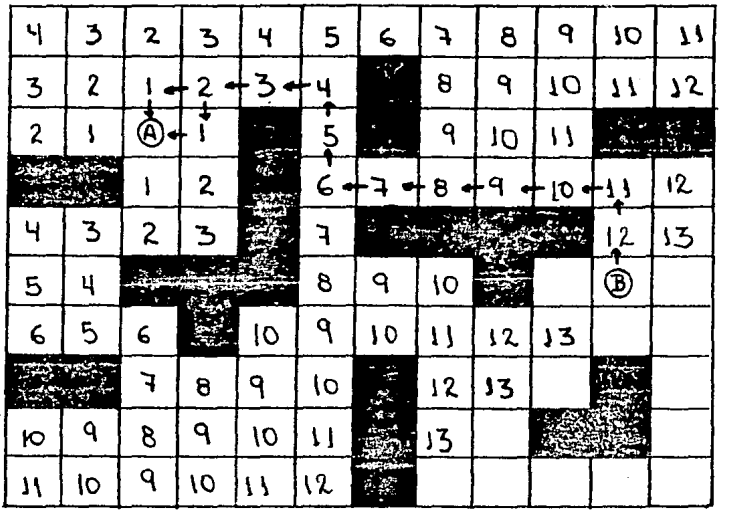
El "2" colocado indica que es posible un camino en esa dirección, y de igual forma los números siguientes. Siempre que un número siguiente es colocado denota un posible camino.

De tal manera que, si no se ha llegado al punto destino y ya no es posible colocar el número siguiente se concluye que no existe trayectoria capaz de unir los puntos. En caso contrario se asegura la existencia de un camino.

Para encontrar este camino hay que aplicar el siguiente razonamiento:

Ya "alcanzado" el punto B, en el punto "13", se entiende que debe haber un "12" adyacente a él, este "12" debe ser adyacente a un "11", etc.

Procediendo de esta manera es sencillo encontrar un camino hasta llegar a A, como se muestra en la figura:



Ocurre una complicación cuando existe una elección de celdas, ésto es, cuando hay dos o más celdas con un " (k - 1)" inmediatamente adyacente a un "k", ésto sucede en nuestro ejemplo cuando se llega a la celda marcada con un "2" en la anterior figura, como se observa existen dos "1" adyacentes a esa celda. En teoría, cualquiera de estas celdas puede ser elegida y se obtiene un camino, pero si se quiere conocer el camino óptimo debe realizarse esa elección tomando en cuenta otros factores.

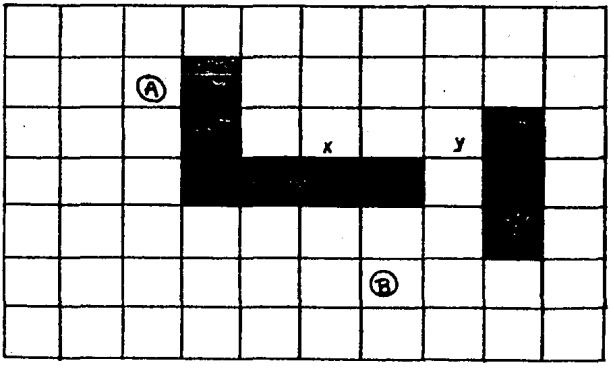
El algoritmo de Lee, permite el diseño de las trayectorias que implican la longitud mínima total de conductor utilizada pa

ra las conexiones, sin embargo, una vez establecido él o los caminos posibles a seguir para la unión de dos pins, el problema significativo no es solamente encontrar el más corto sino que además debe ser el camino que producirá menor conflicto para las interconexiones siguientes.

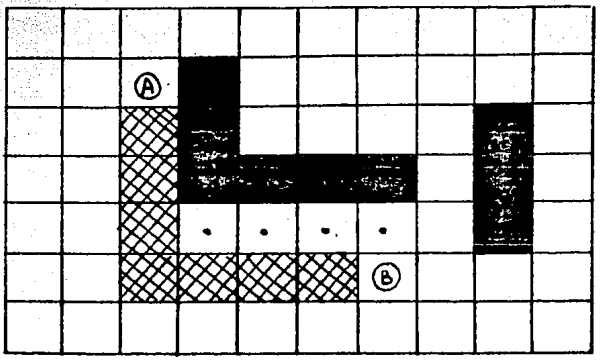
Podría pensarse que lo anterior se asegura con muchos caminos cortos, sin embargo ésto es un error, ya que un "buen camino" corto podría dificultar, y por tanto hacer más largos a los siguientes caminos.

Se analizará entonces otro caso donde el problema es evidente y que ayudará para su solución:

Sean los conductores X y Y, que ya han sido utilizados para la interconexión de dos pins, y se pretende conectar A con B, como se muestra en la figura:

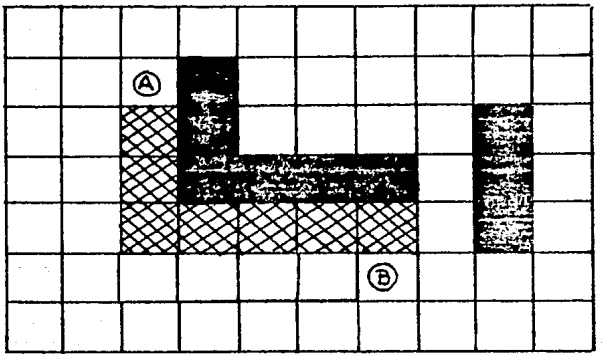


Según el algoritmo anterior, una de las trayectorias más -- cortas para la unión de estos puntos puede ser:



Sin embargo, como se puede observar, existen 4 celdas de la maya (marcadas por un ".") que han quedado ya canceladas para futuras trayectorias, es decir, ningún conductor podrá ya pasar por ellas; ésto provoca menor área disponible para las siguientes conexiones y por tanto una mayor dificultad para realizar-- las.

Otro camino, para establecer la conexión anterior, ofrece - mejores resultados al evitar el desperdicio de celdas, este - - otro camino puede ser también generado con el algoritmo ya ex-- plicado. Este camino tiene la siguiente trayectoria:



Como se puede observar, esta trayectoria tiene la misma longitud que la anterior, pero previene futuras conexiones al no - desperdiciar área de la tableta, la característica notoria de - este camino es que tiende a unirse a las trayectorias ya realizadas para evitar huecos.

Es precisamente aquí donde se realiza la modificación al Algoritmo de Lee, para establecer las conexiones. Ahora, el procedimiento modificado será el siguiente:

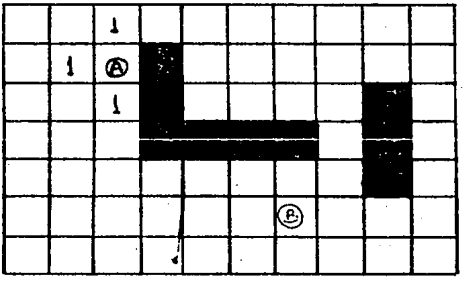
En principio, la asignación de números a las celdas será -- exactamente igual al método ya explicado, pero a cada celda, -- además de este número le será asignado un "peso", este peso llevará la "historia" del camino, ésto es, representará el número de celdas adyacentes, ya ocupadas por trayectorias que se llevan hasta esa celda.

Por ejemplo, un peso de "3" en una celda significará que, - hasta ese momento, el camino seguido, ha pasado adyacentemente- por 3 celdas ya ocupadas.

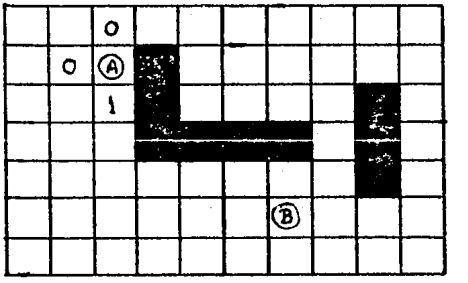
Se utilizará el ejemplo anterior para explicar esta modificación:

Tendremos ahora, para hacer mas sencilla la explicación, -- dos mallas una llamada MALLA DE AVANCE, que representará los números consecutivos, cuya generación ya fue obtenida; y otra, -- que contendrá los pesos de cada celda y que llamaremos MALLA DE PESO. Para un primer avance se tendrá

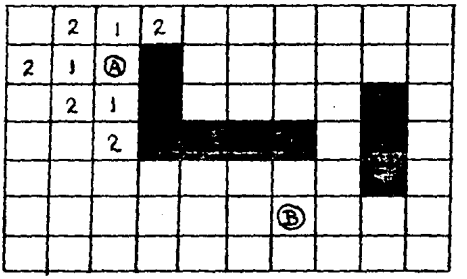
MALLA DE AVANCE

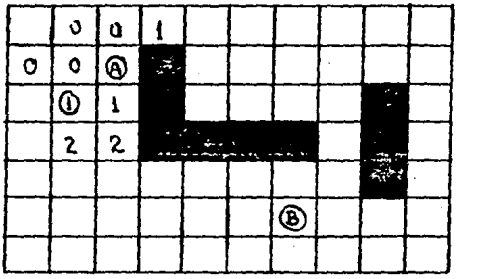


MALLA DE PESO



El "1" indicado en la malla de peso, se asigna porque a la- derecha de esa celda existe ya una celda ocupada. Para el se-- gundo avance se tiene:



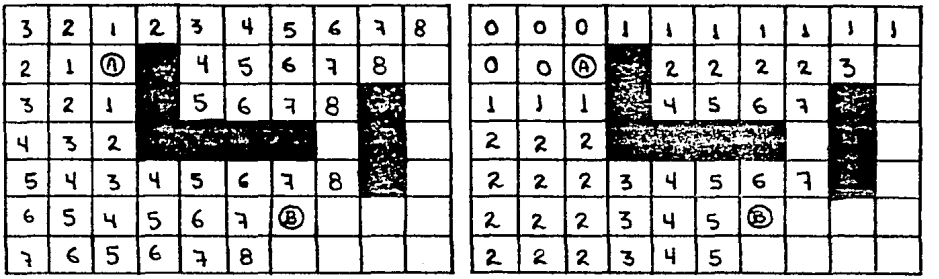


Obsérvese que cada vez que se asigna el número de avance para las celdas, se hace la asignación del peso para la misma, el peso de esta celda es calculado con la suma entre el peso máximo de las celdas adyacentes y el número de celdas adyacentes -- ocupadas, es decir:

Peso celda = peso máximo celda adyacente + número celdas adyacentes ocupadas.

Esta es la razón por la que el peso de la celda encerrada - en un circulo es "1" y no "0".

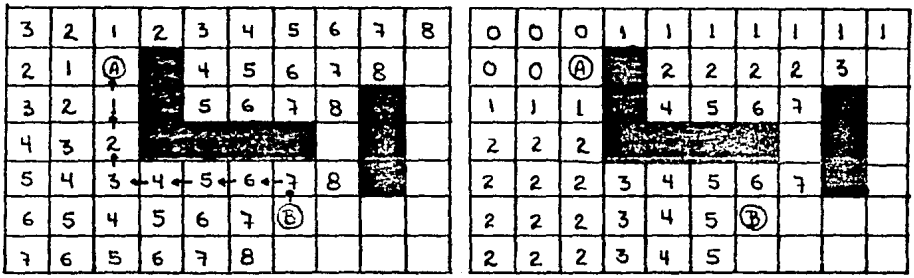
Siguiendo de la misma forma, se obtienen las siguientes mallas:



Ahora solo resta " retroceder inteligentemente " para encontrar el camino óptimo:

Tomando como inicio el punto "B" en la malla de avance, se observa que se tienen dos opciones: La celda superior y la celda izquierda, para la elección se recurre a la malla de peso, esta indica que la celda superior tiene un peso de "6", ésto es existen 6 celdas adyacentes que están ocupadas a través de ese camino; en cambio en la celda izquierda solo existen 5 celdas ocupadas a través de ese camino, por lo tanto, la elección es para la celda superior.

Continuando de la misma forma, se obtiene la siguiente trayectoria:



No se debe olvidar que la malla de peso solo se utiliza - - cuando existen opciones en la malla de avance.

Para cada pareja de puntos a unir se procederá de la misma- forma, y para cada trayectoria generada se almacenarán, en un - archivo las coordenadas de cada una de las celdas que componen- dicha trayectoria.

El pseudocódigo que describe el algoritmo anterior es:

```
INICIO UNION_DE_PUNTOS
HACER HASTA TERMINAR CON LA LISTA DE CONEXIONES
        EMPEZAR BLOQUE_A
        LEER PAR DE PUNTOS A CONECTAR
        ALOJARLOS EN LA MALLA
        DETERMINAR PUNTO ORIGEN Y PUNTO DESTINO
```

```
        GENERAR VALORES PARA MALLA DE AVANCE Y MALLA DE PESO
        SI LA CONEXION FUE POSIBLE ENTONCES:
                EMPEZAR BLOQUE_B
                APLICAR RETROCESO PARA DETERMINAR EL CAMINO
                ALMACENAR LAS COORDENADAS DEL CAMINO
                FIN BLOQUE_B
          SINO:
                INDICAR QUE LA CONEXION NO PUDO EFECTUARSE
        FIN BLOQUE_A
FIN UNION_DE_PUNTOS.
```

El pseudocódigo para la generación de los valores de las mallas, que equivale al algoritmo de Lee modificado, es:

```
INICIO GENERACION_DE_VALORES_PARA_MALLAS.
LA LISTA DE CELDAS ACTUALES SERA LA CELDA ORIGEN
HACER HASTA LLEGAR A CELDA DESTINO O HASTA CAMINO BLOQUEADO:
        EMPEZAR BLOQUE_A
        INCREMENTAR CONTADOR "N"
        HACER MIENTRAS LA LISTA DE CELDAS ACTUALES NO ESTA VACIA:
                EMPEZAR BLOQUE_B
                ASIGNAR "N" A LAS CELDAS VACIAS ADYACENTES A LA_
                __ACTUAL EN LA MALLA DE AVANCE
                ALMACENAR ESTAS CELDAS EN UNA LISTA TEMPORAL
                CALCULAR EL PESO DE LA CELDA ACTUAL Y DEJARLO__
```

```
        __EN LA MALLA DE PESO
        FIN BLOQUE_B
    SI LA LISTA TEMPORAL ESTA VACIA Y NO SE HA LLEGADO EN
                ENTONCES CAMINO BLOQUEADO
        SINO:
        EMPEZAR BLOQUE_C
        HACER LISTA DE CELDAS ACTUALES IGUAL A LISTA_
        __TEMPORAL
        BORRAR LA LISTA TEMPORAL
        FIN BLOQUE_C
    FIN BLOQUE_A
FIN GENERACION_DE_VALORES_PARA_MALLAS
```

Una vez determinada la existencia de un camino de N celdas, se debe aplicar el método de retroceso, cuyo pseudocódigo se -- muestra a continuación:

```
INICIO METODO_DE_RETROCESO
HACER MIENTRAS NO SE LLEGUE AL PUNTO ORIGEN:
    EMPEZAR BLOQUE_A
    DECREMENTAR CONTADOR "N"
    DETERMINAR LA CELDA QUE CONTENGA, EN LA MALLA DE AVAN_
    __CE EL VALOR DE N
    SI EXISTE MAS DE UNA CELDA CON "N" ELEGIR SEGUN LA MA
    __LLA DE PESO
```

```
        ALMACENAR EN UNA LISTA FINAL LAS COORDENADAS DE LA __
        __CELDA ELEGIDA Y DEJARLA EN ESTADO OCUPADO
        FIN BLOQUE_A
FIN DE METODO_DE_RETROCESO
```

C) Graficación de las Pistas.

Como ya se observó, el resultado final del proceso_2, es un archivo que contendrá las coordenadas de las celdas que forman las trayectorias que permiten la conexión de todos los pins.

El pseudocódigo para este proceso es el siguiente:

```
INICIO PROCESO_3
VERIFICAR ESTADO CORRECTO DEL DISPOSITIVO DE SALIDA
DETERMINAR LA ESCALA A GRAFICAR
LEER ARCHIVO GENERADO EN EL PROCESO_2
HACER HASTA EL NUMERO DE CARAS:
        EMPEZAR BLOQUE_A
        HACER HASTA EL NUMERO DE TRAYECTORIAS GENERADAS:
                EMPEZAR BLOQUE_B
                ASIGNAR A UNA COORDENADA "X" LA PRIMER COORDE__
                __NADA DE LA TRAYECTORIA
                HACER HASTA EL NUMERO DE COORDENADAS DE LA TRA_
                YECTORIA:
                EMPEZAR BLOQUE_C
```

```
                ASIGNAR A UNA COORDENADA "Y" LA SIGUIEN__
                __TE COORDENADA
                AJUSTAR LAS UNIDADES DE MALLA A LA ESCALA
                REQUERIDA
                UNIR POR UNA LINEA RECTA LAS COORDENADAS-
                "X" y "Y"
                ASIGNAR A COORDENADA "X" EL VALOR DE "Y"
                FIN BLOQUE_C

            FIN BLOQUE_B
        FIN BLOQUE_A
    FIN PROCESO_3
```

Como ya se mencionó, en este proceso se realiza la graficación de las pistas del circuito impreso en alguno de los dispositivos de salida (graficador ó terminal), por lo que primero se lleva a cabo una verificación lógica del estado de dicho dispositivo.

El algoritmo para este proceso de graficación, es muy simple y consiste solamente en tomar pares de coordenadas y unirlas mediante lineas rectas para generar trayectorias continuas que representan las pistas del diseño.

Para la realización de estas gráficas se utilizó un paquete

de graficación propiedad de Hewlett-Packard, llamado GRAPHICS - II, que puede ser usado en forma programática desde PASCAL; este paquete solo se empleo por comodidad y la adaptación del sistema para que funcione sin él, no representa grandes problemas.

d) Definición de Tipos.

El último proceso, indicado como proceso_4, no significa que necesariamente es utilizado después del proceso_3 y si podría decirse que funciona un tanto independiente de los otros procesos.

En este se efectúa una definición física de los elementos electrónicos que el sistema es capaz de manipular; esta definición consiste en la especificación de las dimensiones del encapsulado que el sistema habrá de "memorizar" para su futura utilización.

Basicamente las dimensiones a considerar para todo encapsulado deben ser:

Número de niveles del encapsulado.- Indica cuantas agrupaciones lineales de pins se tienen.

Separación en unidades de malla entre cada uno de los pins del primero y del segundo nivel.

Número de pins en el primero y segundo nivel.

Especificación sobre la separación entre los niveles.

Para mayor información sobre estos parámetros consultar el capítulo de Manual de Usuario.

Este proceso también es utilizado como consulta para conocer los tipos de encapsulado que el sistema puede manejar.

Este proceso evita al diseñador de circuitos impresos tener que especificar las coordenadas de todos los pins de cada encapsulado, en el momento de indicar la colocación de los elementos en la tableta de conexiones, ya que con la descripción física realizada solo tiene que indicar el tipo de encapsulado a utilizar, la posición del primero de sus pins y la orientación del elemento en la tableta.

El pseudocódigo que describe este proceso es el siguiente:

```
INICIO PROCESO_4
SI SOLO SE REQUIERE CONSULTA ENTONCES:
                DESPLEGAR LOS TIPOS DE ENCAPSULADOS EXISTEN
                TES
SI SE REQUIERE DEFINICION DE TIPOS DE ENCAPSULADO ENTONCES:
          EMPEZAR BLOQUE_A
          ENTRADA DEL ARCHIVO DE DATOS
          ENTRADA DEL ARCHIVO DE TIPOS
          ANALISIS SINTACTICO Y SEMANTICO
          SI EL ANALISIS FUE CORRECTO ENTONCES:
                    EMPEZAR BLOQUE_B
                    GENERAR TABLA DE DATOS
                    AÑADIR ESTA TABLA AL ARCHIVO DE TIPOS
```

```
                FIN BLOQUE_B
            SINO:
                DESPLEGAR LOS ERRORES EN EL VIDEO
        DESPLEGAR RESULTADO FINAL DEL PROCESO
        FIN BLOQUE_A
FIN PROCESO_4
```

Existe parecido entre el proceso_1 y el que se acaba de describir, en cuanto a la validación de información que realizan,- en este caso el análisis sintáctico y semántico es realizado sobre las dimensiones que se especifican para el encapsulado.

V MANUAL DE USUARIO

Este capítulo está dirigido al usuario final del sistema, - es conveniente que quien lo utilice haya leido previamente los- capítulos anteriores, aunque no es indispensable, ésto es, para consultar la utilización del Sistema podría solo hacerse refe-- rencia a este capítulo.

Se presupone que el diseñador debe tener conocimientos básicos de la operación de la computadora HP/1000 (crear, modifi-- car, listar, imprimir y borrar archivos, así como ejecutar programas).

El sistema ha sido diseñado de tal forma que, cada vez que- el usuario incurre en errores, aparecen mensajes en el video de la terminal, que le indican una o más formas probables para la- solución del problema. El usuario debe leerlos con cuidado para elegir el que considere conveniente.

a) Constitución del Sistema.

El sistema esta formado básicamente por 4 procesos:

1.- Entrada y validación de datos.

2.- Elección de trayectorias para las conexiones.

3.- Graficación de las pistas del Circuito Impreso.

4.- Generación de tipos de encapsulados que se utilizarán - en el circuito.

Para la realización de estos cuatro procesos, el Sistema -- utiliza:

tres programas ejecutables: DIPCI.RUN, GRAFCI.RUN, GENTIP.- RUN, un archivo de datos: TIPS.RUN

El programa DIPC.RUN realiza los dos primeros procesos, esto es, recibe información analizar, la valida (verifica su congruencia) y en caso de no encontrar errores, genera las trayectorias a seguir por los conductores.

El programa GRAFCI.RUN efectúa el tercer proceso, ésto es,- la graficación de las pistas del circuito impreso a diseñar, en los dispositivos previstos para este fin.

El programa GENTIP.RUN genera nuevos tipos de encapsulados- para que puedan ser dados como dato en el primer proceso, ade--

más, muestra en el video que tipos son ya permitidos por el sistema; como se observa GENTIP.RUN ejecuta el cuarto proceso.

El archivo TIPS.DAT contiene los tipos de encapsulado que el sistema permite utilizar.

El usuario puede preguntarse la razón de separar los procesos de elección de trayectorias y el de graficación de pistas.- ésto se hace con la finalidad de permitir al usuario obtener las gráficas o diseños que considere necesarios sin tener que efectuar procesos ya realizados.

b) Conceptos Básicos.

Para la comprensión clara de este manual, es necesario explicar al usuario la simbología y el significado de los diagramas de sintaxis que se utilizan para la descripción del sistema.

Podría pensarse que se ha plasmado en este manual un pequeño lenguaje, mediante el cual el usuario comunicará al sistema los datos necesarios para la elaboración de un diseño.

Sin olvidar que un lenguaje es un medio de comunicación, podemos decir que el lenguaje es un conjunto de reglas sintácticas, que al mezclarse dan orígen a frases que describen situaciones o eventos.

Piénsese en una frase del idioma español como:

"El mono come un platano"

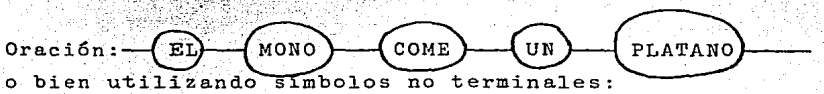
Establezcamos ahora reglas para la producción de esta oración:

ORACION esta definida por: SUJETO Y PREDICADO.
SUJETO esta " " : FRASE SUSTANTIVA
FRASE SUSTANTIVA " " : ARTICULO Y NOMBRE PROPIO
PREDICADO " " " : VERBO Y COMPLEMENTO
COMPLEMENTO" " " : FRASE SUSTANTIVA
ARTICULO " " " : EL o UN o ...
NOMBRE PROPIO" " " : MONO o PLATANO o ...
VERBO " " " : COME o ...

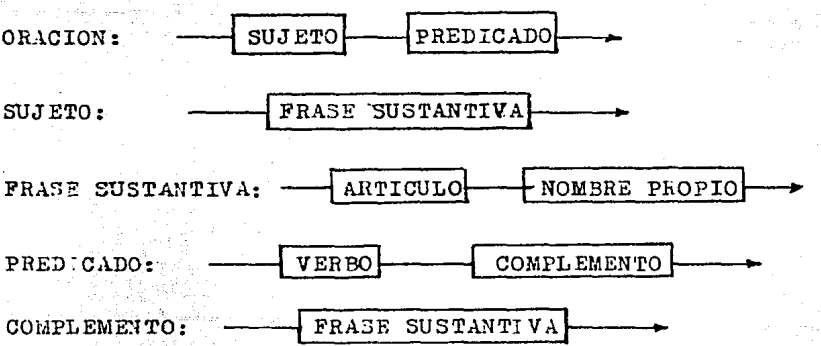
Se puede notar que existen dos tipos de SIMBOLOS, símbolos-no terminales, ésto es, aquellos que aún están definidos en terminos de "algo" (oración, sujeto, predicado, frase sustantiva,-complemento, artículo, nombre propio y verbo), y símbolos terminales, aquellos que ya no están definidos en terminos de otra cosa (EL, UN, MONO, PLATANO, COME, ...).

Para hacer una diferencia, los símbolos terminales serán encerrados en un círculo y los no terminales en cuadrado, y para-establecer un orden y secuencia, los símbolos no terminales y -terminales se unirán con una línea vertical continua.

Expresando la oración ejemplo, según estas convenciones:

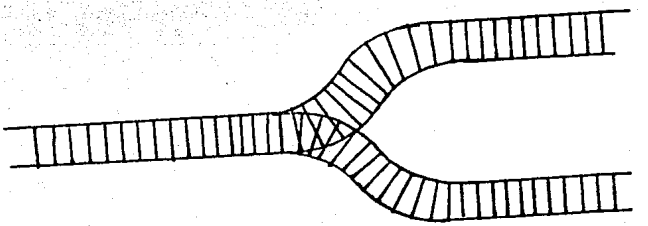


o bien utilizando símbolos no terminales:

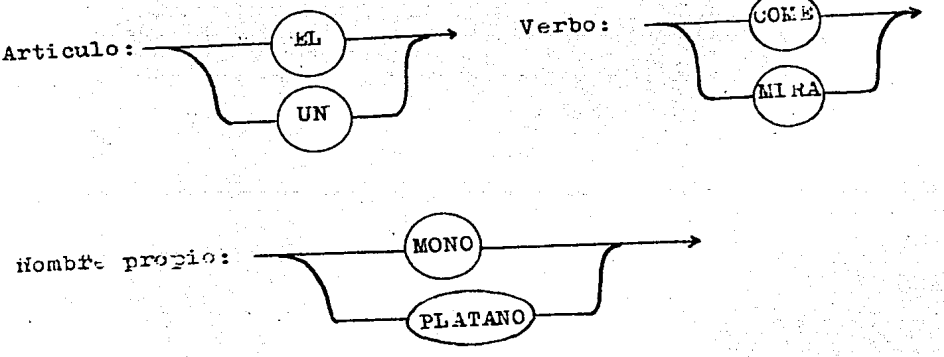


Al intentar describir algún otro símbolo no terminal, se ob

serva que, tienen dos o más opciones, imagínese el usuario la - trayectoria de un ferrocarril en una vía que tiene dos o más caminos:



de igual forma:



Lo anterior son los diagramas de sintaxis para ORACION.

(aunque se debe aclarar que en Español, el sentido de "oración" es mucho más complejo que el considerado en este ejemplo).

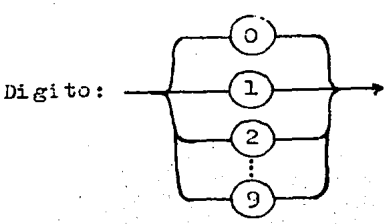
Una vez establecidos los diagramas de sintaxis, se puede calificar a un conjunto de símbolos como parte o no de un lenguaje; sin embargo, del ejemplo anterior, siguiendo los diagramas de sintaxis, podría producirse la siguiente frase:

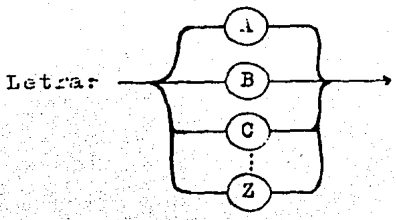
" El plátano come un mono "

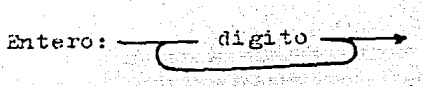
Sintácticamente la frase es correcta, pero no tiene sentido, su significado no es correcto, según nuestros conocimientos los plátanos no comen monos. Se puede decir entonces, que para que una frase de algún lenguaje sea parte de él, debe cumplir dos condiciones: la condición de sintaxis y la de semántica - - (significado).

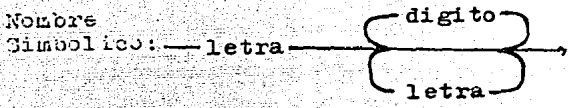
En este manual se encuentran los diagramas de sintaxis que describen el lenguaje mediante el cual el usuario comunicará información al sistema, la semántica a considerar dependerá de si los datos " comunican " algo o si contradicen la información declarada.

Los diagramas básicos para la descripción del lenguaje son:



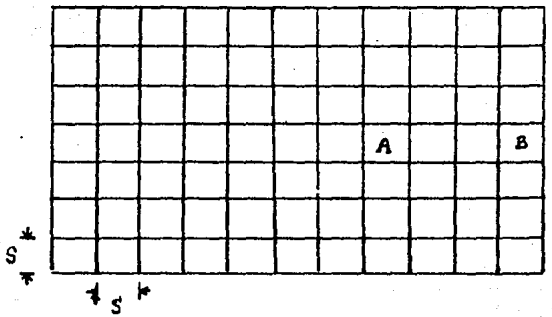






En la descripción de los datos, el usuario encontrará que - frecuentemente se utiliza el concepto de UNIDADES DE MALLA, - - que se explicará a continuación:

Para la descripción de los elementos a interconectar, y para determinar su ubicación en la tableta, supóngase a la tableta dividida por lineas horizontales y lineas verticales que - - guardan un espaciamiento constante entre ellas:



Ahora la tableta tomará el nombre de MALLA y cada una de --

sus divisiones será una UNIDAD DE MALLA.

En la malla anterior se han colocado 2 puntos, estos puntos estan definidos por sus coordenadas rectangulares: tomando como origen la parte inferior izquierda de la malla, las coordenadas de estos puntos son: A (8,4) y B (11,4).

Con el fin de normalizar esta Unidad de Malla se propone un espaciamiento entre lineas de: S = 0.10 in = 2.54 mm

El usuario debe tener siempre en cuenta, que los datos estarán referidos a esta unidad de malla, por lo que es conveniente que la distribución de los elementos sea realizada en una malla como la definida, o bien en una que guarde las proporciones.

c) Entrada de Datos.

Dado que, la cantidad de datos puede llegar a ser muy grande, y para evitar errores en la introducción de los mismos al proceso, los datos serán implementados en un archivo independiente y no en el momento de la ejecución del Sistema, ésto es, el programa no se ejecutará en forma interactiva con el usuario, y todo requerimiento de información, será tomado de este archivo de datos; lo anterior tiene la gran ventaja de que el usuario puede hacer modificaciones en algunas partes de su archivo de datos sin tener que dar todos los datos nuevamente.

De modo normal de Operación de la Computadora donde fué implementado el Sistema, provee de un EDITOR de archivos por pantalla, donde puede ser creado o modificado el archivo ya mencionado.

La estructura de este archivo es la siguiente:

INICIO;

BLOQUE DE DEFINICION DE TABLETA

BLOQUE DE DEFINICION DE ELEMENTOS

BLOQUE DE DEFINICION DE CONEXIONES

FIN;

El bloque de Definición de la Tableta tiene la siguiente estructura:

INICIO DT;

DIMENSIONES DE TABLETA

LISTA DE ZONAS NO PERMITIDAS-

-PARA TRAYECTORIAS.

CARAS A UTILIZAR EN EL DISEÑO

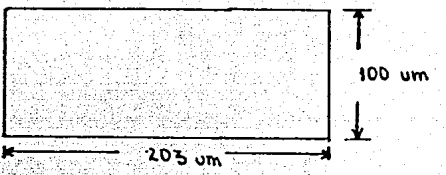
FIN DT;

Cada bloque de esta estructura equivale a una línea dentro- del archivo de datos, exceptuando el bloque de "lista de zo- - nas...", en este, cada zona especificada será una línea.

+ DIMENSIONES DE TABLETA.- En esta parte se especifica el - tamaño de la tableta, en unidades de malla; para ésto es necesario indicar dos parámetros: ancho y largo, según el diagrama:

ancho, largo: entero

Ejemplo, sea la tableta:



Se debe teclear:

D_100,203;

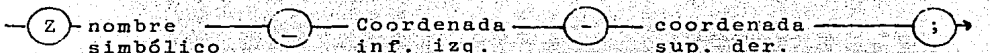
+ LISTA DE ZONAS NO PERMITIDAS PARA TRAYECTORIAS.- Especifica áreas de la tableta donde el usuario considera, para fines -- propios, que no deben existir conductores. El sistema respetará estas zonas y no serán invadidas por ninguna pista.

El bloque tiene la siguiente estructura:

```
INICIO LZ;
ZONA  1
ZONA  2
  ...
  ...
  ...
ZONA  K
FIN LZ;
```

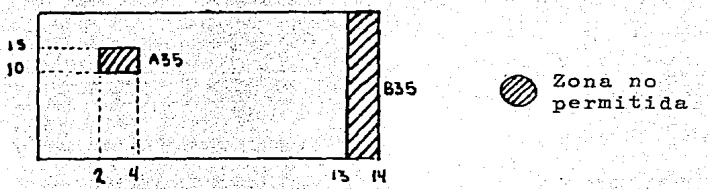
Donde el bloque "ZONA K " especifica un área cuadrilátera.

La sintaxis debe ser:



abcisa
ordenada: entero

Ejemplo, sea la tableta y las zonas:

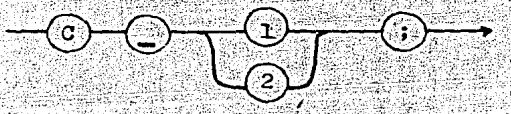


Nota.+ El orígen para indicar coordenadas, siempre será tomado en la parte inferior izquierda de la tableta, las coordenadas deberán ser dadas en unidades de malla.

Para el ejemplo, se tecleará, suponiendo nombres simbólicos de A35 y B34 para las zonas:

```
INICIO_LZ;
ZA35_2,10-4,15;
ZB34_13,0-14,18;
FIN_LZ;
```

+ CARAS A UTILIZAR EN EL DISEÑO.- Este parámetro especifica si el diseño es requerido en ambas caras de la tableta o solamente en una cara. La sintaxis que se debe cumplir es:



Ejemplo:

C_2; Indica que se usarán ambos lados de la tableta.

El bloque de Definición de Elementos tiene la siguiente estructura:

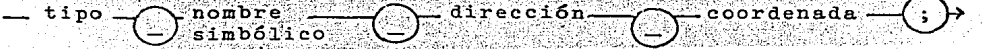
INICIO DE;
DESCRIPTOR 1
DESCRIPTOR 2
...
...
....
DESCRIPTOR N
FIN DE;

Cada bloque de esta estructura equivale a una línea dentro del archivo de datos.

+ DESCRIPTOR N.- Mediante este parámetro el usuario indica al sistema qué tipo de encapsulados utilizará y cuál será su -- ubicación en la tableta, en general, aquí se especifica qué elementos serán alojados y su posición en coordenadas rectangulares en u.m.

El sistema, desde su creación, contiene una tabla de tipos- de encapsulados (transistores, circuitos integrados, etc.) -- donde se indican dimensiones para separación entre pins. El -- usuario puede consultar e incluso aumentar esta tabla de tipos- (ver inciso f) en este capítulo). El usuario solo puede hacer referencia a elementos ya definidos en la tabla de tipos.

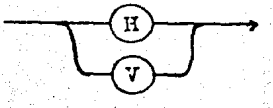
La sintaxis válida para este parámetro es:



donde:

"tipo" debe ser alguno de los existentes en la tabla de tipos. " nombre simbólico " es un identificador para el usuario,- con el cual hará referencia al elemento en el momento de indicar conexiones.

"dirección", indica cómo esta ubicado el elemento en la tableta nótese que solo son permitidas dos direcciones:

Dirección: 

"coordenada", denota la posición en coordenadas rectangulares y en unidades de malla del pin número 1 del encapsulado o elemento a considerar. Para normalizar cuál será considerado como pin 1, ver inciso f) en este capítulo.

El usuario debe tener cuidado al definir esta coordenada, ya que es la base para la enumeración que se hará de los pins del encapsulado y de su disposición en la tableta.

Para mayor claridad daremos un ejemplo de una tabla de tipos, como la que el usuario puede consultar, y la forma en que debe ser utilizada en la definición de elementos:

Sea la tabla de tipos:

tipo	pins Nivel 1	pins Nivel 2	separación Nivel 1	separación Nivel 2	Coordenada del 1er. pin en nivel 2, referido a pin 1 del nivel 1	
T0220	3	0	4	0	0	0
T6D	8	8	4	4	0	15
T039	2	1	8	0	4	4
⋮	⋮	⋮	⋮	⋮	⋮	⋮

(en la parte correspondiente al inciso f) de este capítulo se - detalla el significado de toda la tabla).

Ejemplos:

T0220_POT_H_23,30; — Indica un encapsulado tipo "T0220" que el usuario llama "POT", por distinguirlo solamente, que tiene una posición horizontal, ésto es, los 3 pins (indicados en la tabla) están horizontalmente en la tableta y las coordenadas del pin 1 son - - (50,10) u.m.

(tipo, nombre simbólico, coordenadas pin 1, posición)

T6D_ACOPLO_V_20,30; — Indica un encapsulado tipo- "T6D" que el usuario llama- "ACOPLO", en una posición - vertical, ésto es los 8 - - pins (indicados en la ta- - bla) están verticalmente en la tableta, las coordenadas del pin 1 son (50,10) u.m.

El bloque de Definición de Conexiones tiene la estructura:

INICIO DC;

CONEXION 1

CONEXION 2

...

...

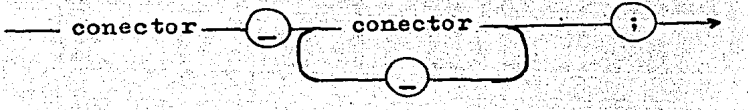
...

CONEXION N

FIN DC;

Cada bloque de esta estructura equivale a una línea en el archivo de datos.

+ CONEXION N.- Especifica los pins que deben ser interconectados, para ésto es necesario indicar el elemento (de los definidos en el Bloque de Definición de Elementos) y el número de pin de ese elemento que se conectará. El diagrama de sintaxis para este parámetro será:



Conector:

pin: entero

Se presupone que el "nombre simbólico" a ser utilizado en esta especificación, fue ya declarado en el bloque de Definición de Elementos.

Ejemplo:

POT/1-POT/3-ACOPLO/2; Indica que el pin 1 del elemento "POT" (previamente definido) se conectará con el pin 3 del mismo elemento y además se conectará con el pin 2 del elemento llamado "ACOPLO".

En resúmen, un ejemplo válido como archivo de datos puede ser el siguiente:

```
INICIO;
      INICIO_DT;
            D_200,300;
            INICIO_LZ;
                  ZA35_2,10-4,15;
                  ZB34_13,0-14,18;
            FIN_LZ;
            C_1;
      FIN_DT;
      INICIO_DE;
```

```
            T0220_POT_H_23,30;
            T6D_ACOPLO_V_50,10;
    FIN_DE;
    INICIO_DC;
            POT/1-POT/3-ACOPLO/2;
            ACOPLO/2-T6D/7;
    FIN_DC;
FIN;
```

El usuario tendrá en cuenta las siguientes restricciones:

- Todas las letras utilizadas serán mayusculas.
- La sangría es opcional y en caso de haberla puede ser del número de espacios que se requiera.
- Toda linea terminará con ";"
- No son permitidas continuaciones de linea.
- Los espacios en blanco no son permitidos, solo en la sangría.
- No se permiten lineas en blanco dentro del archivo.
- El nombre del archivo de datos puede ser el que más convenga al usuario.
- No se deben repetir nombres de componentes en la definición.
- El entero más grande debe ser 32767
- El número de caracteres para nombre simbólico será de 16- como máximo.

d) Verificación de Datos.

Una vez que el archivo de datos ya ha sido creado, el sistema está listo para iniciar su proceso; el primer paso en éste, es la validación de la información contenida en el archivo de datos.

Esta validación es para compensar la no interactividad del sistema con el usuario e intentar que la información a procesar sea lógicamente aceptable, o al menos congruentemente sintáctica con los requerimientos del análisis.

En esta parte se utiliza el programa DIPCI.RUN, cuyo objetivo es el de validar la información y en caso de no encontrar errores, la procesa y genera las trayectorias. En el caso que se encuentren errores en la sintaxis o en la semántica, el programa aborta la ejecución del sistema al terminar la validación. Al abortar la ejecución se despliega en el vidéo de la terminal, el número de errores detectados.

Los errores pueden ser producidos por caracteres ilegales en los datos o por una secuencia de ellos que no sigue las reglas establecidas en los diagramas de sintaxis, o bien porque existen datos que no tienen poder informativo o no son compatibles con la información ya validada.

Ya desplegado el número de errores, el sistema devuelve el control al modo normal de operación de la computadora, solo si existe al menos un error.

Un ejemplo de esta validación en el vidéo es:

Número de errores : 5

Ejecución Abortada por el Sistema.

Errores indicados en Archivo @.LST.

Para que el usuario conozca los errores en que ha incurrido, el Sistema genera un archivo listable, cuyo contenido es similar al archivo de datos, solamente que este tiene señalizaciones en las lineas donde detectó algún tipo de error, estas señalizaciones son códigos numéricos, cada código tiene asociado un tipo de error. Al final del archivo se totalizan los errores y se exponen los códigos numéricos con una explicación de su significado. Este archivo listable de errores puede ser desplegado en el vidéo de la terminal o bien impreso mediante la ejecución de comandos propios del modo normal de operación de la computadora.

e) Ejecución del Sistema.

Como ya se mencionó, el primer paso para la elaboración del diseño es la entrada y validación de información. Esto se logra ejecutando el programa DIPCI.RUN, pasando como parámetro el nombre del archivo donde residen los datos.

Este programa debe ser ejecutado en el modo normal de operación de la computadora llamado CI.

El usuario debe teclear: DIPCI ARCH.DAT)
) = tecla RETURN

donde ARCH.DAT es el nombre del archivo donde están los datos.

Ya realizado este paso, el sistema empieza la validación de datos, el tiempo qeu tarda en realizarla es proporcional al número de datos.

Si el sistema encuentra algún error en este archivo de datos, aborta la ejecución y retorna a CI, en caso contrario despliega el siguiente mensaje:

Validación terminada
Errores : 0
Empieza proceso para generación de --
trayectorias
...

Entonces comienzan a aparecer en el vidéo pequeños mensajes que indican al usuario en que parte del proceso se encuentra el sistema, ésto es con el fin de que se tenga una noción sobre el tiempo de análisis de información.

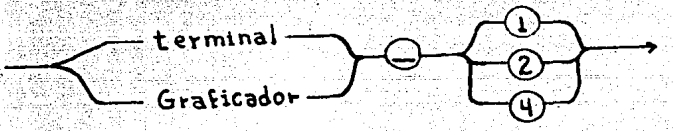
Si la generación de trayectorias termina sin contratiempos, el sistema crea un archivo llamado PISTAS.DAT que contiene el resultado del proceso anterior, ésto es, posee toda la información para poder "dibujar", en los dispositivos destinados para ello, las pistas del diseño del circuito impreso. En el caso de que existieran cruces inevitables, las coordenadas de los puntos en conflicto serán grabadas en un archivo llamado PISTAS.LST que podrá ser consultado por el usuario, además serán desplegados mensajes de error en el vidéo. Cabe añadir que, en caso de cruce inevitable entre algunos puntos, la ejecución del proceso no se detiene sino que continua intentando las otras conexiones sin tomar en cuenta el problema anterior.

Se crea además otro archivo : PISTAS.TXT, que el usuario puede listar o imprimir y donde encontrará información referente a su diseño, como:

- Area de conductor utilizada (en in^2 y en %)
- Area disponible en la tableta "
- Número de trayectorias generadas.
- Densidad de agujeros por in^2
- Tipos de encapsulados utilizados.

Ya con el archivo PISTAS.DAT creado, se puede ejecutar el - tercer proceso del sistema, la graficación de las pistas del cir cuito impreso. Para esto se ejecuta el programa GRAFCI.RUN.

Este programa necesita un parámetro, que indicará la escala a utilizar y el dispositivo donde será graficado el resultado, - este parámetro sigue la siguiente sintaxis:



Los números 1,2,4 indican la escala a ser utilizada, ésto - es: 1:1, 2:1, 4:1, respectivamente; estas escalas solo tienen -- sentido cuando se use el graficador, ya que en el caso de la ter minal, el diseño solo será proporcional al real y su tamaño de-- pende de las dimensiones de la pantalla.

La manera de ejecutar este programa es, en el modo CI:

GRAFCI GRAFICADOR 2) (por ejemplo)

El sistema puede detectar un error, ya sea en la sintaxis - del parámetro o bien porque no exista el archivo PISTAS.DAT o -- porque este archivo no haya sido generado por DIPCI.RUN o tam- - bién porque encuentre algún error en los dispositivos de salida-

(en este caso se deberán a condiciones físicas inadecuadas). En cualquiera de estos casos el error es explicado detalladamente y se propone al usuario una solución para remediarlo.

En el caso que el diseño sea requerido en las dos caras de la tableta, las pistas de una cara aparecerán en color rojo y las otras, en la cara posterior, aparecerán en azul.

f) Generación y Consulta de Tipos.

El sistema tiene implementada una tabla de tipos que contiene los tipos de encapsulados que pueden ser utilizados en el archivo de datos (específicamente en el bloque de Definición de -- Elementos).

Esta tabla puede ser consultada o bien aumentada por el usuario según sus requerimientos. Para ser consultada el usuario debe teclear, en el modo CI:

GENTIP ↵ : Esto produce un listado en pantalla
GENTIP 6 ↵ : Produce un listado en impresora.

En el caso que el usuario quiera aumentar esta tabla de tipos debe crear un archivo de datos que contendrá parámetros que utiliza el sistema para este fin. Se debe considerar que el objetivo de esta generación es definir DIMENSIONES y forma de los encapsulados.

El diagrama de sintaxis para la especificación de estos parámetros es la siguiente, (el usuario tecleará una linea por cada tipo de elemento a generar):

donde: "pins 1" es el número de pins en el nivel 1.

"pins 2" es el número de pins en el nivel 2.

"s1" es la separación de pins en el nivel 1 en u.m.

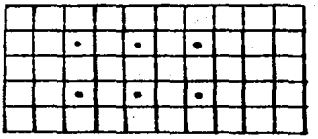
"s2" es la separación de pins en el nivel 2 en u.m.

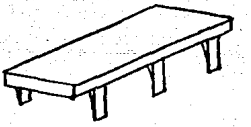
"coordenada" es la coordenada del primer pin en el nivel 2, referido al primer pin del nivel 1.

"nombre simbólico" será el nombre del tipo del encapsulado.

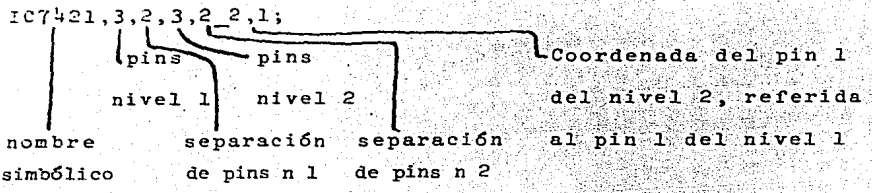
Se explicará ahora a que se refiere el termino "nivel":

Ejemplo 1, sea el elemento cuyo encapsulado tenga la siguiente configuración de pins:

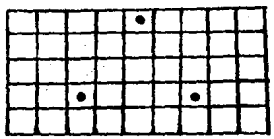




Como se puede observar este encapsulado tiene 2 niveles, el nivel inferior siempre será considerado como nivel 1. Para las figuras anteriores la descripción del encapsulado debe ser:



Ejemplo 2, sea el elemento:

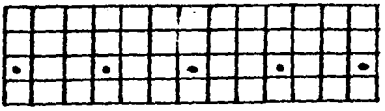


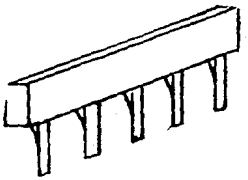
Como se observa, este elemento también tiene dos niveles, - el primer nivel tiene 2 pins y una separación de 4 u.m. entre -- pins, el segundo nivel tiene 1 pin y la separación entre pins es de 0 (no hay otro pin para hacer la referencia) y el pin del nivel 2 tiene por coordenada (2,3) referido al primero pin del nivel 1.

En este caso la descripción del encapsulado es:

T039,2,4,1,0_2,3;

Ejemplo 3, sea el encapsulado:





El elemento tiene solo un nivel y la separación entre pinses de 3 u.m., los demás parámetros deben ser ceros. La descripción en este caso es:

LIN5,5,3,0,0_0,0;

El único nivel que puede tener cero pins es el nivel 2.

Resumiendo, un ejemplo del contenido del archivo para la generación de tipos de encapsulados puede ser:

```
INICIO;
IC7421,3,2,3,2_0,2;
T039,2,4,1,0_2,3;
LIN5,5,3,0,0_0,0;
FIN;
```

Otra parte importante para el usuario, es la forma en que el sistema enumera los pins, ya que esta enumeración deberá concordar con la que el usuario indica en el momento de establecer las conexiones entre pins. Para los ejemplos mencionados.

3

4 5 6

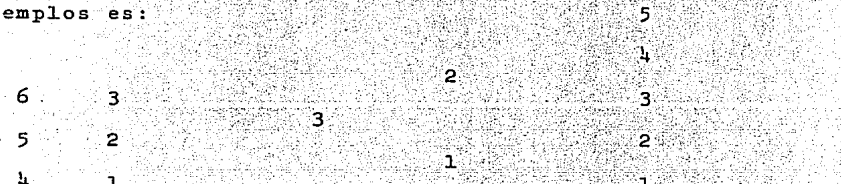
1 2 3 4 5

1 2 3 1 2

El sistema enumera de izquierda a derecha en el nivel 1, y de derecha a izquierda en el nivel 2.

Se debe también considerar que la forma en que el usuario genera sus elementos o encapsulados será, convencionalmente tomada, como la dirección horizontal. La dirección vertical se obtendrá rotando el encapsulado 90' en el sentido de las manecillas del reloj, pero contrario, siendo el eje de giro el pin 1.

De esta forma, la posición considerada como vertical para los ejemplos es:



Ya creado el archivo de datos con la información necesaria para la generación de tipos, el usuario debe ejecutar el programa GENTIP.RUN de la siguiente forma:

gentip archl.dat ↵

donde ARCHL.DAT es el nombre del archivo de datos.

En caso de encontrar errores en este archivo de datos, no es creado ningún elemento nuevo y son desplegados, en el vidéo los errores en que el usuario incurrió.

Cuando el usuario genere el archivo de datos debe tener encuenta las restricciones mencionadas al final del inciso c)

Antes de utilizar el sistema, se recomienda al usuaric consultar los tipos ya definidos de encapsulados desde su creación.

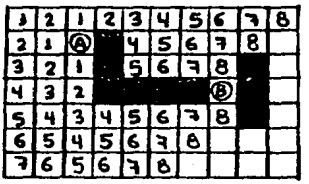
C O N C L U S I O N E S

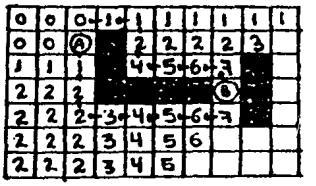
Un análisis posterior a la finalización del Sistema, hizo - resaltar algunos detalles que se deben considerar al efectuar -- cualquier diseño de pistas; estos detalles se pueden resumir en- tres puntos:

1) El sistema nunca consideró, ni el espaciamiento entre -- conductores ni el ancho de las pistas, como debió haber sido, ya que se limitó a considerar estos parámetros como constantes para todo circuito a diseñar, sin tomar en cuenta que en algunas ocasiones son necesarios varios tipos de ancho de conductor y espaciamiento. Esto se debió a la rigidez de la malla considerada,- así como de las unidades de malla convenidas, es decir, el camino a seguir tenía que involucrar a TODA la celda. Una posible - solución a este problema sería plantear celdas de tamaño "variable", ésto es, celdas cuyas dimensiones puedan variar dependiendo del ancho del conductor necesario para establecer la conexión entre dos pins.

2) El Algoritmo de Lee modificado, podría ser todavía mejorado si se considerara que, aún cuando se recurre a una malla de peso para la elección de celdas en el caso en que existan trayectorias igual de cortas, podrían existir, en esta misma malla de- peso, dos o más opciones a seguir, esto ocurre en el ejemplo con

siderado en la página 51, si el punto B estuviera ubicado en la-celda que muestra la siguiente figura:





Una posible solución a este conflicto, es crear al mismo -- tiempo que las otras mallas, una malla de CAMBIOS DE DIRECCION-- que contabilice el número de "esquinas" que la trayectoria ha -- realizado hasta esa celda. Tomando como punto de decisión la -- elección de la celda que implique un camino con menos cambios de dirección. Sin embargo esta solución puede no ser muy conveniente, si no se cuenta con buenos recursos de memoria, ya que la -- creación de otra malla involucra el uso de una buena cantidad de memoria en la computadora.

3) En el caso de los cruces inevitables, y contarse con dos caras para efectuar el diseño, el lugar de limitarse a reportar-la conexión no posible, puede intentarse la utilización de la cara contraria, ésto es, en vez de tener cuatro posibles celdas a-seguir, se podrían tener cinco: 4 celdas de la cara actual y 1 - celda que permite el movimiento a la cara contraria.

Como se puede observar, el Sistema presentado puede ser su-

jeto a mejoras notables, con el fin de hacerlo más eficaz, lo -- aquí se presentó fue solo el trabajo inicial de lo que se espera sea una futura automatización en el Diseño de Circuitos Impre- - sos.

B I B L I O G R A F I A

(1) DESIGN AUTOMATION OF DIGITAL SISTEMS
Breuer Melvin E.
Prentice Hall, 1972.

(2) DESIGN CIRCUITS IN SPACE TECHNOLOGY
Linden, Albert E.
Prentice Hall, 1962

(3) HANDBOOK OF ELECTRONIC PACKAGING
Charles A. Harper.
Mc Graw Hill, 1969.

(4) PRINTED CIRCUITS HANDBOOK.
Clyde F. Coombs Jr.
Mc Graw Hill, 1979.

(5) ELECTRONIC DRAFTING AND DESIGN.
Nicholas M. Raskhodoff
Prentice Hall, 1972.

(6) THE ART OF COMPUTER PROGRAMMING VOL. 1
Donald E Knuth
Addison Wesley, 1973.

(7) ALGORITHMS + DATA STRUCTURES = PROGRAMS
Wirth N.
Prentice Hall, 1976.

(8) PRINTED CIRCUIT DRAFTING TECHNICAL MANUAL & CATALOG.
Bishop Graphics, Inc. 1981.

(9) CATALOGO DE SEMICONDUCTORES
Fairchild 1981.