UPAEP

# UNIVERSIDAD POPULAR AUTÓNOMA DEL ESTADO DE PUEBLA

## ESCUELA DE SISTEMAS COMPUTACIONALES

# SISTEMA GENERADOR DE CÓDIGOS DE CONTROL NUMÉRICO (SGC)

# TESIS

Que para obtener el título:

LICENCIADO EN SISTEMAS COMPUTACIONALES

Presenta:

María Milagros Ruiz Limón

Asesores:

Ing. Eduardo Gavaldón
Ing. Francisco Enrique Monterrubio Herrera

Puebla. Noviembre, 1995

Dedico el presente trabajo con mucho cariño a mis papás, por haberme hecho parte de esta hermosa familia. Gracias por su ejemplo, entrega, apoyo y amor

A Gisela, Pilar y Blas, mis hermanos, por estar siempre conmigo. Especialmente a Pilar por haberme apoyado y escuchado tantas veces.

A mis tíos Carmina y Gustavo, por que para mí son y han sido como un ángel de la guarda que siempre se hace presente incondicionalmente. Gracias

A Christoph, por que cerca o lejos de ti, estás conmigo. Gracias por haberme ayudado a concluir este trabajo.

A Paco, por ser quien eres para mí y por todos los momentos buenos y malos que hemos compartido. Muchas gracias por haberme asesorado este trabajo.

A mis amigas, quienes son un tesoro invaluable para mí. Especialmente a Myriam y Celia, porque juntas desarrollamos parte de nuestro trabajo de tesis

Agradezco el apoyo recibido al Centro de Capacitación para el Trabajo Industrial número 8, y de manera muy especial al profesor de dicha institucion, Arturo Mendoza Zamora por las asesorías y soporte técnico brindados

A mis maestros, quienes han tenido el interés y la voluntad para compartir sus conocimientos conmigo

A mis asesores y vinodales, Ing. José M Bedolla Cordero, Ing Jaime Castillo Rodríguez e Ing. Eduardo Davaldón por el apoyo recibido

# INDICE

# INTRODUCCIÓN

Desde hace algún tiempo a la fecha los procesos de fabricación han sido fuertemente retados a incrementar su eficiencia y eficacia. El rápido aumento de los productos requeridos por la población, ha empujado a la ciencia y a la técnica a optimar los procesos productivos

Para elevar la productividad han surgido algunas herramientas que satisfacen tales exigencias Estas herramientas son los sistemas automatizados Un sistema automatizado es aquél que está controlado por un sistema computacional, por ejemplo, una fresadora

Las fresadoras son máquinas-herramienta de gran importancia en la industria metal-mecánica. En ellas se efectúa el proceso de fresado Este proceso consiste en labrar piezas en diferentes materiales (acero, aluminio, madera, etc ) por medio de uno o más cortadores giratorios (fresas), que cuentan con múltiples aristas cortantes

Las actividades de diseño y manufactura son tareas que requieren de precisión y mucho tiempo Los sectores productivos, poco a poco van empleando recursos más sofisticados y modernos como lo son los sistemas CAD/CAM (Sistemas de Diseño y Manufactura Asistidos por Computadora) que garantizan alta velocidad y un elevado grado de confiabilidad

Las ventajas que ofrecen las máquinas-fresadoras, si se complementan con sistemas CAD/CAM, son mucho mayores. Se puede incrementar enormemente la cantidad de piezas producidas, la exactitud y a su vez disminuir el desperdicio de materia prima, la mano de obra y los riesgos que se presentan durante el trabajo.

En la carrera profesional en Ingeniería Industrial-Mecánica que se imparte en el Instituto Tecnológico de la Cd. de Puebla (ITP), el proceso de aprendizaje para la fabricación de piezas mecánicas se desarrolla con el uso de máquinas fresadoras convencionales Este proceso de aprendizaje consiste de diseñar en papel la pieza a maquinar y, a partir del modelo, manipular secuencialmente la fresadora para obtener el resultado deseado Las autoridades del ITP han decidido

modernizar este proceso educativo mediante la adquisición y uso de máquinas-fresadoras automatizadas, las cuales obedecen, a través de un sistema computarizado, a una serie de instrucciones dadas por el usuario que en conjunto forman un programa de control numérico (CN).

Los programas para las máquinas de CN pueden ser codificados a través de dos caminos: manual o automatizado En el proceso manual el operador parte del dibujo de la pieza a fabricar que contenga las dimensiones de la misma, enlista las instrucciones en código de CN y las introduce en la fresadora a través del tablero de la misma. La programación automatizada se realiza a través de sistemas CAD-CAM (Diseño y Manufactura Asistidos por Computadora)

El personal de la escuela utiliza la forma manual de programación de la máquina Fresadora de marca EMCO y lenguaje de programación EMCOTRONIC debido a que no cuentan con un sistema automatizado que optimice este procedimiento La desventaja de la programación manual es la *dificultad que se presenta en el control absoluto sobre las entidades geométricas requeridas para el* desarrollo de alguna pieza. El tiempo que utilizan es largo y los resultados muchas veces no son alentadores Este problema puede ser minimizado mediante el uso de un sistema CAM que permita al usuario pasar automáticamente de un archivo creado en un sistema CAD a la generación del código de CN listo para ser introducido en el tablero de la fresadora y ser ejecutado por la misma.

Pensando en esto surgió la idea de desarrollar un sistema que genere códigos de control numérico para la fresadora EMCO, es decir, que facilite la programación de las órdenes que serán ejecutadas en ella, siendo así el objetivo fundamental de este trabajo el de desarrollar un sistema que automatice la programación de la máquina fresadora de marca EMCO con lenguaje de programación EMCOTRONIC que se encuentra en el laboratorio de manufactura avanzada del Instituto Tecnológico de la Ciudad de Puebla

Para la optimización del uso de la fresadora EMCO en estudio, se propone el empleo del Sistema Generador de Códigos (Sistema GC) que es un sistema de CAM que en conjunto con un sistema de CAD (AutoCAD), facilita a los usuarios de la fresadora EMCO-EMCOTRONIC la programación de piezas en dos dimensiones con profundidad constante, incrementando la rapidez y sencillez con respecto al sistema tradicional. En sí, este sistema CAD-CAM-CN se presenta en tres fases principales

1 El uso de algún CAD que ya exista en el mercado y que permita almacenar sus archivos en formato DXF, por ejemplo AutoCAD

2. El uso del Sistema GC (CAM) que corre en cuatro fases principales·

- Carga a la memoria y despliega un archivo que el usuario seleccione a la vez Estos archivos deben estar almacenados en formato DXF y pueden ser creados en el paquete de CAD

- Captura de variables que establecen las condiciones de fresado. selección de herramienta, compensación de herramienta, altura de seguridad, uso de refrigerante, profundidad de fresado, datos del material, cada una de éstas se explica en el capítulo 5

- Seleccionar una trayectoria de fresado o bien permitir al Sistema GC generarla automáticamente.

- Generación de los siguientes resultados representación grafica del fresado y la generación del código de CN.

3 Introducción del listado de CN a la fresadora para ser ejecutado.

## CAPÍTULOS QUE INCLUYE ESTE TRABAJO

El capítulo uno, "LA EDUCACION TÉCNICA", hace referencia al concepto de Educación Técnica y los niveles principales en los cuales ésta se desarrolla Contiene una descripción de la Escuela Técnica y una reseña histórica de la Educación Técnica en México. Finalmente se presenta una breve descripción de la Educación Técnica en el Instituto Tecnológico Regional de Puebla y de cómo se realiza en él la capacitación en los procesos mecánicos

El capítulo dos, "EL PROCESO DE FRESADO", contiene los conceptos básicos y fundamentales del proceso de fresado Contiene también la descripción de la estructura básica de una fresadora, las diferentes clases de cortadores de que pueden hacer uso estas máquinas-herramienta y una clasificación de las mismas.

El capítulo tres, "EL DISEÑO Y LA MANUFACTURA ASISTIDOS POR COMPUTADORA", provee conocimientos fundamentales del diseño y la manufactura asistidos por computadora En este

capítulo se da una introducción de ambas actividades, así como de los sistemas que en ellas se emplean Se habla también del control numérico, el cual es un proceso de control a través de un programa

El capítulo cuatro, "INTERFASE", presenta una introducción al concepto de interfase y las diferentes interfases básicas de que consta el Sistema GC, las cuales son tres: 1) entre el sistema de CAD y el Sistema GC, 2) entre el Sistema GC y el usuario y 3) entre el Sistema GC y la fresadora EMCO

El capítulo cinco, "FRESADORA EMCO-EMCOTRONIC", se refiere a los conceptos básicos y fundamentales de esta fresadora, sus características y los elementos necesarios para ejecutar el fresado

El capítulo seis, "DESARROLLO DEL SISTEMA GC", presenta los conceptos fundamentales del desarrollo del Sistema GC, la justificación de su desarrollo, la definición del proyecto, los alcances y limitaciones, así como los manuales técnico y del usuario

# CAPITULO 1. LA EDUCACIÓN TÉCNICA

Dado el objetivo fundamental de este trabajo, que es el de desarrollar un sistema que automatice la programación de la máquina fresadora de marca EMCO que se encuentra en el laboratorio de manufactura avanzada del Instituto Tecnológico de la Ciudad de Puebla, se presentan en este capítulo algunos conceptos básicos de Educación Técnica debido a que se trata de una máquina herramienta de tipo didáctico.

Este capítulo hace referencia al concepto de Educación Técnica y los niveles principales en los cuales se desarrolla Contiene una descripción de la Escuela Técnica y una reseña histórica de la Educación Técnica en México. Finalmente se presenta una breve descripción de la Educación Técnica en el Instituto Tecnológico Regional de Puebla y de cómo se realiza en él la capacitación en los procesos de manufactura El contenido de este capítulo está basado en el libro La educación técnica en México (1980-1990) publicado por la Secretaria de Educación Pública de México.

## 1.1 LA EDUCACION TÉCNICA.

La Educación Técnica es la rama de la educación que se ocupa de preparar personas para diversas profesiones y de dotar a los individuos de hábitos y capacidades que promuevan su desarrollo Esta educación debe ser eminentemente técnica y profesional, con un contenido humanístico complementario adecuado

La Educación Técnica consiste en llevar a una persona hasta el conocimiento técnico por medio de una serie de ejercicios estimulados por la reflexión y basados en la ciencia

En si, la educación debe tener por fin, desarrollar ciertas cualidades del carácter tales como la atención, la reflexión, el juicio, la iniciativa, la disciplina, el espíritu de solidaridad, la perseverancia, la voluntad, etc El término educación lleva implícito el concepto de vinculación, ya que sería estéril cualquier proceso educativo emprendido sin considerar el fin social o económico al que está dirigido

La Educacion Técnica tiene los siguientes objetivos básicos:

1 Preparar al alumno para que se desempeñe en su vocación

2 Ofrecerle la oportunidad de aprender asuntos de orden cultural

3o Enlazar adecuadamente el estudio para que, si lo desea, pueda luego continuar otras carreras superiores por medio de cursos apropiados

## 1.2 NIVELES PRINCIPALES DE LA EDUCACION TÉCNICA.

La Educación Técnica se desarrolla en tres niveles principales:

- Universitario
- Técnico
- Laboral

El primero se desenvuelve en las universidades y en los institutos tecnológicos superiores. Estos últimos, por su orientación específicamente técnica y especializada forma ingenieros cuyo nivel cultural y de preparación general es diferente Las universidades procuran formar integralmente al hombre a la vez que le otorgan una profesión Los institutos tecnológicos procuran más adiestrar personas para ejercer la ciencia del ingeniero, sin preocuparse tanto por dar una formación completa Los ingenieros técnicos son los que han cursado estudios técnicos superiores, y están habilitados para ejecutar cálculos en los cuales se necesita el análisis matemático Pueden hacer ensayos de laboratorio e interpretar los resultados

El segundo se desenvuelve en las escuelas técnicas Los candidatos a cursar este nivel deben haber finalizado la escuela primaria y/o secundaria Los técnicos son las personas que han cursado estudios completos y tienen conocimientos teóricos y prácticos. Conocen los procesos comunes de cálculo, los metodos de control y medida, y saben preparar planes y ejecutarlos.

El tercero se desarrolla en las empresas que se ocupan de preparar personas para el ejercicio de oficios y de perfeccionar a los obreros que aprendieron por sus propios medios.

## 1.3 LA ESCUELA TÉCNICA.

La escuela técnica y la enseñanza que en ella se imparte es muy costosa. Los ambientes de que consta son los siguientes

1. EL AULA DE CLASE Es el lugar en que se enseña la teoría y las materias de cultura general Es el ámbito fundamental a nivel universidad.

2 EL LABORATORIO Es el local destinado a las demostraciones prácticas sobre leyes de física, de química y de tecnología. En el se hacen mediciones y comprobaciones, pero en general, no se adquieren habitos ni se hace práctica profesional específica

3 EL TALLER Es el ámbito fundamental a nivel laboral y técnico, porque en él se aprenden las habilidades manuales que son, en mayor o menor grado, la base de la profesión Debe estar dotado de todo lo necesario y, sobre todo, debe contar con un funcionamiento adecuado En los talleres existen

   - Máquinas herramientas. (tornos, fresas, etc )
   - Herramientas manuales (pinzas, soldadores, etc )
   - Materiales de consumo. (aceites, combustibles, hojas de sierra, mechas, etc )
   - Materiales para elaboración. (materia prima, hierro, madera, componentes de radio, cables, etc )

4 EL AULA DE DIBUJO.- En el aula los alumnos aprenden el idioma del dibujo, el cual es tal vez el más usado por los técnicos El aula debe contar con una iluminación superior a lo normal, con adecuados lugares para guardar los elementos de trabajo y las piezas o modelos que se usan para práctica, además de los instrumentos de medición y control que se emplean

5. LA DIRECCIÓN Y LA ADMINISTRACIÓN.

6. AUXILIARES

Desde el punto de vista de una metodología, las asignaturas de una escuela a nivel universidad y técnico se pueden clasificar en la siguiente forma:

- Asignaturas científicas.
- Asignaturas técnicas.
- Asignaturas culturales.

Las ASIGNATURAS CIENTÍFICAS son la física, la química y la matemática. Su presencia en las escuelas de Educación Técnica se debe a que la tecnología (y las asignaturas técnicas) tienen todas una base científica que las justifica. Además, las asignaturas científicas son una contribución importantísima a la formación general del individuo.

Las ASIGNATURAS TÉCNICAS son las que resultan indispensables para aprender los fundamentos de un oficio o profesión y que permiten al individuo adquirir habilidades que luego le servirán, entre otras cosas, para obtener una adecuada retribución por sus servicios. Estas asignaturas transmiten conocimientos específicos y son distintas para cada especialidad. Las más importantes son:

- EL TALLER. Es el lugar en que se enseña todo lo relacionado con el trabajo manual; es la más importante a nivel laboral y de valor sobresaliente en las de nivel técnico. Se requieren sólo algunas horas para hacer un "maquinista", pero en cambio se necesitan meses o años para formar un buen operario especializado, capaz de interpretar los planos que se le dan y de ejecutar una determinada pieza sin la ayuda de sus superiores. La mente guía a la mano, y sobre esta afirmación debe trabajarse en el taller.

- LA TECNOLOGÍA. Es la materia (o materias) que acompañan al taller y se encarga de explicar el porqué de las cosas. Se apoya en los conocimientos científicos y la síntesis del trabajo. Se encarga de estudiar los materiales, las herramientas, las máquinas necesarias y los métodos de trabajo. El taller describe, mientras que la tecnología explica y se puede realizar fuera del taller si así conviene.

- EL DIBUJO TÉCNICO. Es el intermediario entre el pensamiento abstracto y la acción real. Es el camino mínimo entre el deseo de crear una cosa y el objeto mismo.

Las ASIGNATURAS CULTURALES son aquellas que no son indispensables al oficio o la profesión, pero sí al ser humano y al ciudadano para su vida de relación.

## 1.4 LA EDUCACION TÉCNICA EN MÉXICO A TRAVÉS DEL TIEMPO.

En México, como en toda nacion, el problema de educación y capacitación de los jóvenes para la participación más adecuada y la futura dirección de las empresas de que formen parte, es importante para garantizar sus intereses y el progreso de la sociedad

En México, la institucionalización de la enseñanza tecnológica y científica tuvo sus orígenes en la época colonial, con la fundación del Real Seminario de Minería.

A partir de 1932 comienza a funcionar la Preparatoria Técnica de cuatro años y como nivel superior se sitúan las escuelas de Altos Estudios Técnicos En el período presidencial del general Lázaro Cárdenas se crea el Instituto Politécnico Nacional (IPN) Al crearse el IPN la preparatoria técnica de cuatro años se subdividió en dos niveles de dos años cada uno: la prevocacional y la vocacional Después la prevocacional se organizó en tres años conformando lo que ahora se conoce como secundaria técnica Al poco tiempo de su creación las escuelas vocacionales se reestructuran dando origen a los Centros de Estudios Científicos y Tecnológicos a nivel bachillerato, llamados CECyT

En 1948 nacen los Institutos Tecnológicos Regionales, al amparo administrativo del Instituto Politecnico Nacional y con la finalidad inicial de capacitar a la gente de acuerdo a las características de cada region, brindando desde este año hasta 1959, atención educativa en prevocacional, vocacional y carreras técnicas medias Los Institutos Tecnológicos Regionales se desenvuelven estrechamente ligados al desarrollo del país

## 1.5 CAPACITACIÓN EN LOS PROCESOS MECÁNICOS EN EL INSTITUTO TECNOLÓGICO REGIONAL DE PUEBLA.

En la carrera profesional en Ingeniería Industrial-Mecánica que se imparte en el Instituto Tecnologico de la Cd de Puebla (ITP), el proceso de aprendizaje para la fabricación de piezas mecanicas se desarrolla con el uso de máquinas fresadoras convencionales Este proceso de aprendizaje consiste de diseñar en papel la pieza a maquinar, y a partir del modelo, manipular secuencialmente la fresadora para obtener el resultado deseado. Las autoridades del ITP han decidido modernizar este proceso educativo mediante la adquisición y uso de máquinas-fresadoras

automatizadas, las cuales obedecen, a través de un sistema computarizado, a una serie de instrucciones dadas por el usuario que en conjunto forman un programa de CN.

Los programas para las máquinas de CNC (Control Numérico Computarizado) pueden ser codificados a través de dos caminos: manual o automatizado. En el proceso manual el operador parte del dibujo de la pieza a fabricar que contenga las dimensiones de la misma, enlista las instrucciones en código de CN y las introduce en la fresadora a través del tablero de la misma. La programación automatizada se realiza a través de sistemas CAD-CAM (Diseño y Manufactura Asistidos por Computadora).

El personal de la escuela utiliza la forma manual de programación de la máquina Fresadora EMCO debido a que no cuentan con un sistema automatizado que optimice este procedimiento. La desventaja de la programación manual es la dificultad que se presenta en el control absoluto sobre las entidades geométricas requeridas para el desarrollo de alguna pieza. El tiempo que utilizan es largo y los resultados muchas veces no son alentadores. Este problema puede ser minimizado mediante el uso de un sistema CAM que permita al usuario pasar automáticamente de un archivo creado en un sistema CAD a la generación del código de CN listo para ser introducido en el tablero de la fresadora y ser ejecutado por la misma.

# CAPITULO 2. EL PROCESO DE FRESADO

Dado el objetivo fundamental de este trabajo, que es el de desarrollar un sistema que automatice la programación de la máquina fresadora de marca EMCO que se encuentra en el laboratorio de manufactura avanzada del Instituto Tecnológico de la Ciudad de Puebla, se presentan en este capítulo los conceptos básicos y fundamentales del proceso de fresado. Contiene también la descripción de la estructura básica de una fresadora, las diferentes clases de cortadores de que pueden hacer uso estas máquinas-herramienta y una clasificación de las mismas. Este capítulo está basado al libro LA FRESADORA de Toledo Matus (1989) y en la tesis GENERADOR DE CODIGOS DE CONTROL NUMERICO del Ingeniero Francisco Monterrubio (Instituto Politécnico Nacional, 1993).

## 2.1 EL PROCESO DEL FRESADO

Las máquinas fresadoras son de gran importancia en la industria metal mecánica. En ellas se efectúa el proceso de fresado.

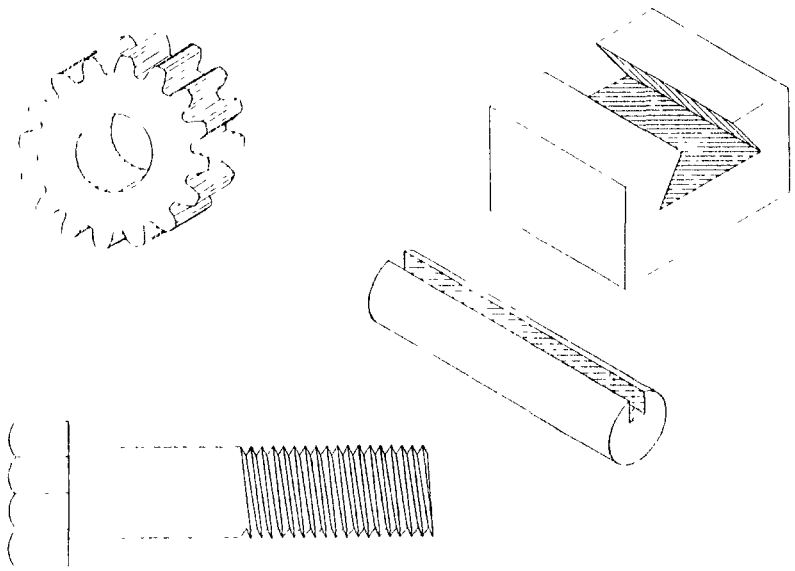
El fresado es un proceso mecanizado que consiste en cortar material de una pieza empleando una serie de cortadores giratorios llamados fresas, los cuales están provistos con múltiples aristas cortantes.

El proceso de fresado se puede realizar en una gran variedad de materiales aunque normalmente se efectúa sobre metales, destacando principalmente el acero, hierro fundido, latones, bronces, aluminio, etc. De entre los no metálicos destacan los plásticos.

La fresadora se emplea para realizar superficies planas o perfiles irregulares, pudiendo también utilizarse para tallar engranes y roscas, ranuras, chaveteros, taladrar agujeros y graduar con precisión medidas regularmente espaciadas (ver fig. 2.1.1), para lo cual se fabrican fresas en varias formas y tamaños, teniendo cada una de ellas una finalidad específica.

En el fresado se combinan dos clases de movimientos: el movimiento rotatorio del cortador y el avance de éste, que es la velocidad con que penetra en el material a trabajar.

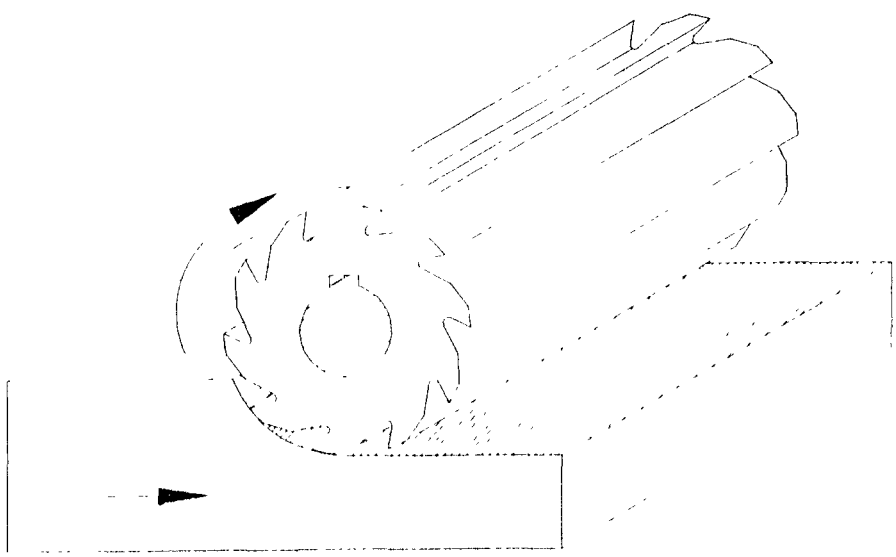
**Figura 2.1.1 Piezas elaboradas por una fresadora**



**Fuente : Toledo Matus, 1989**

En la figura 2.1.2 tenemos la representación gráfica de una fresa en movimiento relativo con un material. En un proceso de fresado puede avanzar la herramienta y el material permanecer estático o viceversa, también pueden moverse ambos simultáneamente

**Fig. 2.1.2 Proceso de fresado.**

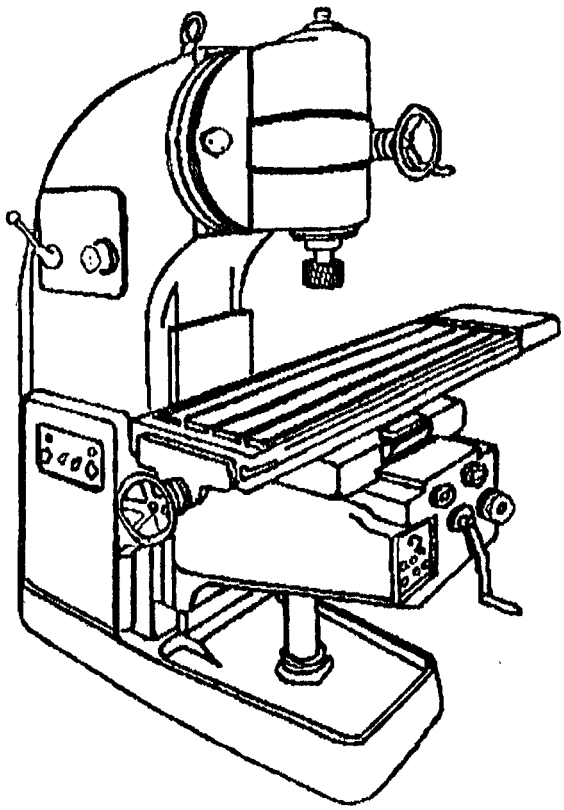


**Fuente : Toledo Matus, 1989.**

## 2.2. ESTRUCTURA DE UNA FRESADORA.

Las partes principales de las fresadoras se muestran en la figura 2.2.1, y son : la columna (1), el caballete (2), la cartela (3), la columna (4), el husillo (5), y el brazo superior (6).

**Figura 2.2.1 Componentes de una fresadora.**



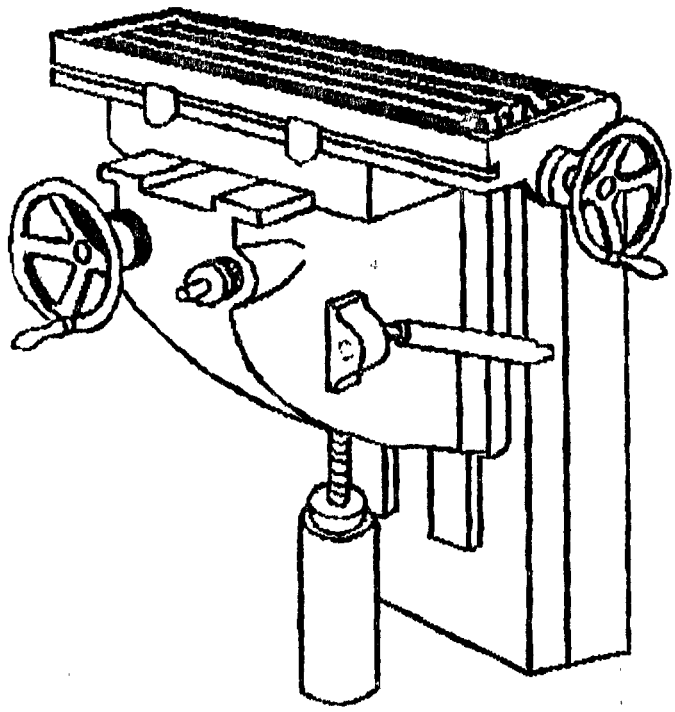
**Fuente : Toledo Matus, 1989.**

La **columna**, incluyéndose la **base**, es la pieza que sostiene todas las demás partes de la máquina.

La **cartela** sostiene el caballete. Dentro de ella se hallan los engranajes de cambio de avance. La cartela o consola puede subirse o bajarse sobre la cara de la columna; su altura puede ajustarse mediante el tornillo de elevación que le sirve de soporte.

El **caballete** sostiene la mesa. Se apoya sobre las superficies de la cartela mecanizadas con precisión, las cuales le sirven de guía. (ver fig. 2.2.2).

**Figura 2.2.2 El caballete.**



**Fuente : Toledo Matus, 1989.**

La mesa sirve para sujetar la pieza. A lo largo de toda la longitud de superficie superior tiene mecanizadas ranuras en "T", las cuales sirven para alinear la pieza o el dispositivo de sujeción de la misma. Los pernos que se usan para sujetar la pieza o el dispositivo de sujeción se ajustan en las ranuras en "T". (v. fig. 2.2.3).

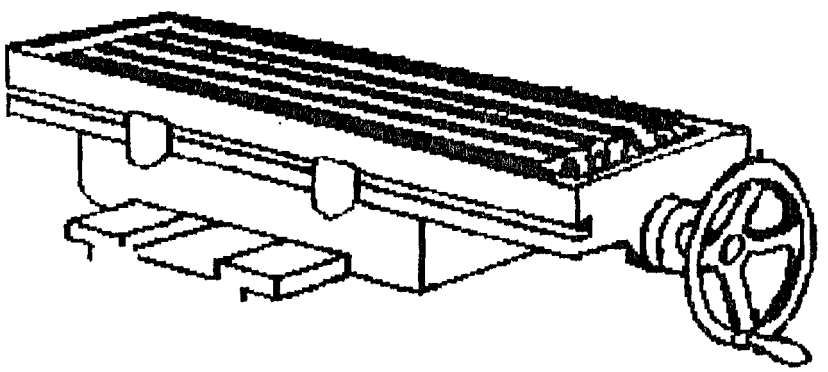
El **husillo** sujeta y acciona diversas herramientas. Es un árbol (árbol porta fresas) que va montado sobre balerps alojados en la columna. El husillo es accionado por un motor eléctrico.

El **brazo superior** va montado encima de la columna y sostiene el árbol portafresas. Es

El **brazo superior** va montado encima de la columna y sostiene el árbol portafresas. Es ajustable y puede fijarse en cualquier posición.

Además, cada una de las máquinas en un taller mecánico cuenta con uno o más accesorios, que se diseñan para incrementar la versatilidad de éstas, proporcionando u a mayor capacidad de trabajo.

**Figura 2.2.3 Mesa de una fresadora.**



**Fuente : Toledo Matus, 1989.**

## 2.3. EL CORTADOR.

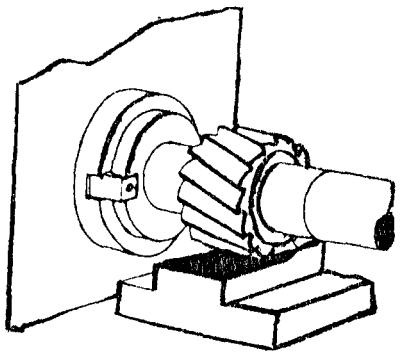
Un cortador para fresadora (fresa) es una herramienta rotatoria de corte que se pone en contacto con la pieza de trabajo y quita el metal en forma de virutas. Normalmente el cortador se sujeta en una posición fija (pero rotatoria) y la pieza de trabajo se mueve. Los cortadores para fresadora pueden ser de muchos tamaños, formas y clases.

### 2.3.1. CLASES DE CORTADORES.

Entre las clases de cortadores más comunes tenemos :

a. **CORTADORES DE APLANAR.** Estos cortadores son de forma cilindrica con cortadores sobre la periferia. Se utilizan para fresar superficies planas (ver fig. 2.3.1 (a)) La máquina fresadora EMCO-EMCOTRONIC no ha sido diseñada para ellos.

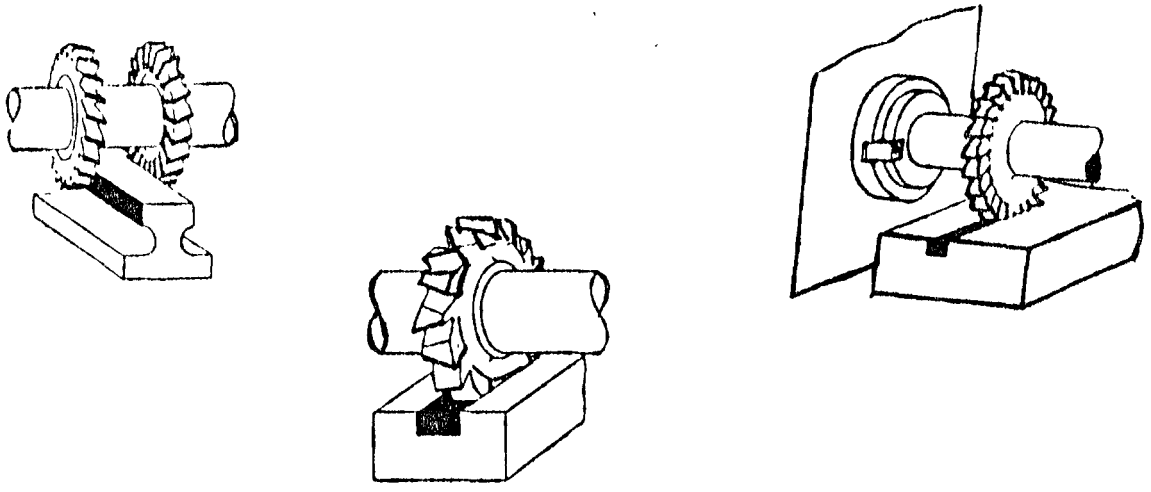
**Fig. 2.3.1 (a) Cortadores de aplanar.**



**Fuente : Toledo Matus, 1989.**

b. **CORTADORES PARA FRESADO LATERAL.** Estas fresas tienen dientes alrededor de su periferia y también sobre uno o ambos lados (ver fig. 2.3.1 (b)). La fresadora EMCO-EMCOTRONIC no ha sido diseñada para hacer uso de este tipo de cortadores.

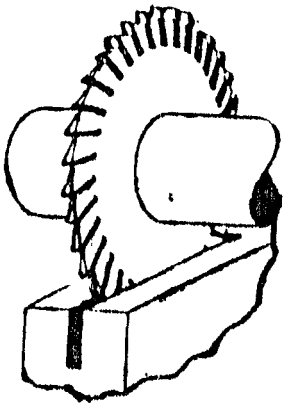
Fig. 2.3.1 (b) Cortadores para fresado lateral.



Fuente : Toledo Matus, 1989.

c. **SIERRAS CORTADORAS DE METALES.** Estas se parecen a pequeñas hojas de sierras circulares. Tienen dientes alrededor de la circunferencia y algunas también las tienen laterales. La fresadora EMCO-EMCOTRONIC no ha sido diseñada para hacer uso de este tipo de cortadores.

Fig. 2.3.1 (c) Sierras cortadoras de metales.



Fuente : Toledo Matus, 1989.

**d. FRESAS PARA CORTE FRONTAL.** Se diseñan para fresar ranuras y cavidades donde no se pueda usar la fresa de tipo de árbol ordinaria (ver fig. 2.3 1 (d)). La fresadora EMCO-EMCOTRONIC ha sido diseñada para hacer uso de este tipo de cortadores

**Fig. 2.3.1 (d) Fresas para corte frontal.**

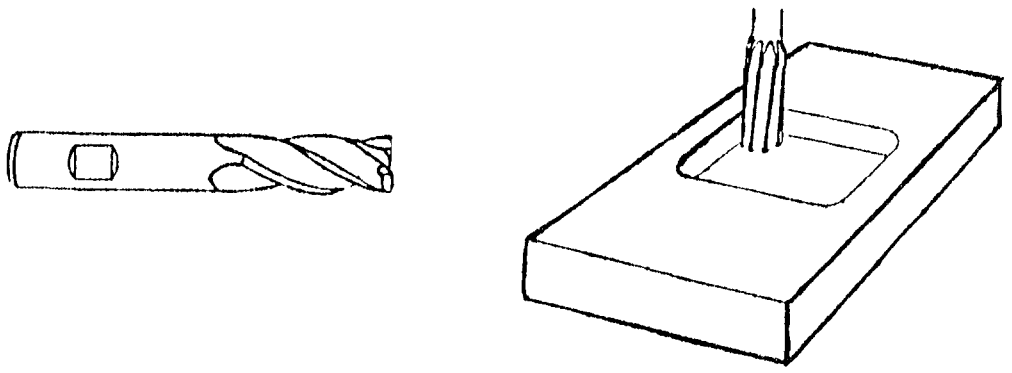
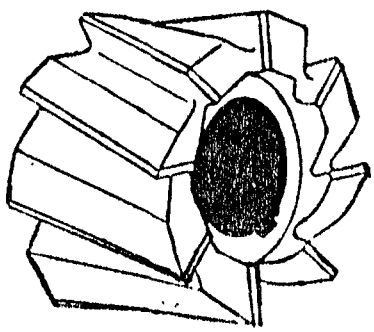


**Figura :** Toledo **Matus, 1989.**

**e. FRESAS PARA CORTE FRONTAL DEL TIPO HUECO.** Estas tienen un agujero central para montarlas en un árbol corto. Tienen dientes cortantes en el extremo y alrededor de la periferia (ver fig. 2 3 1 (e)). La fresadora EMCO-EMCOTRONIC ha sido diseñada para hacer uso de este tipo de cortadores.

**Fig. 2.3.1 (e) Fresas de corte frontal del tipo hueco.**

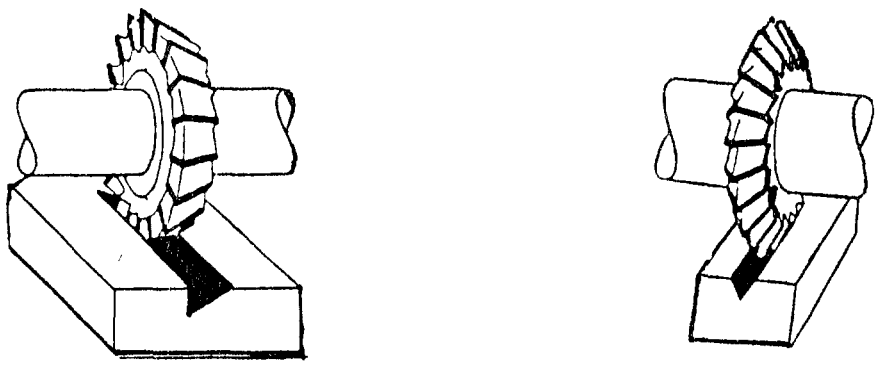


**Fuente : Toledo Matus, 1989.**

f. **FRESAS DE CORTE ANGULAR.** Estas pueden ser de diferentes tipos. Los más importantes son :

**1.Fresas de corte angular.** Sirven para fresar ángulos distintos a los de 90 grados (v.fig. 2.3.1 (1)).

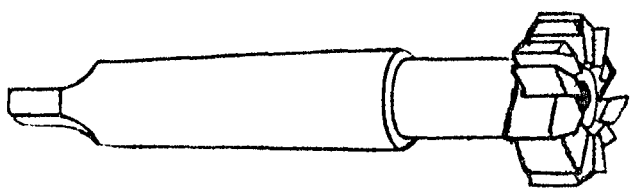
**Fig. 2.3.1 (1) Fresas de corte angular.**



**Fuente : Toledo Matus, 1989.**

2. **FRESAS PARA RANURAS EN T.** Las fresas para cortar ranuras en T se utilizan para cortar este tipo de ranuras en la parte superior de las mesas para las fresadoras. ( ver. fig. 2.3.1 (a)).

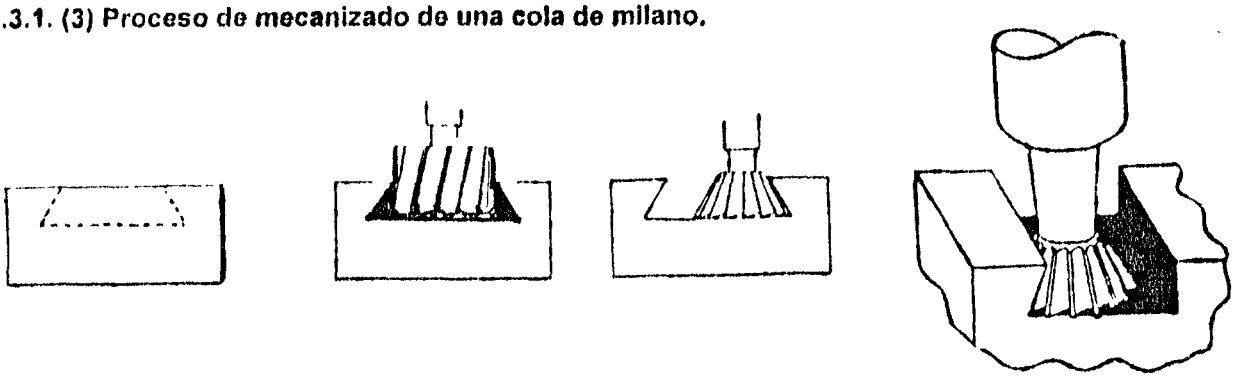
**Fig. 2.3.1 (2) Fresas para ranuras en T.**



**Fuente : Toledo Matus, 1989.**

3. **FRESAS PARA RANURAS DE COLA DE MILANO.** En la fig. 2.3.1 (3) se muestra el proceso de mecanizado de una cola de milano partiendo de un bloque.

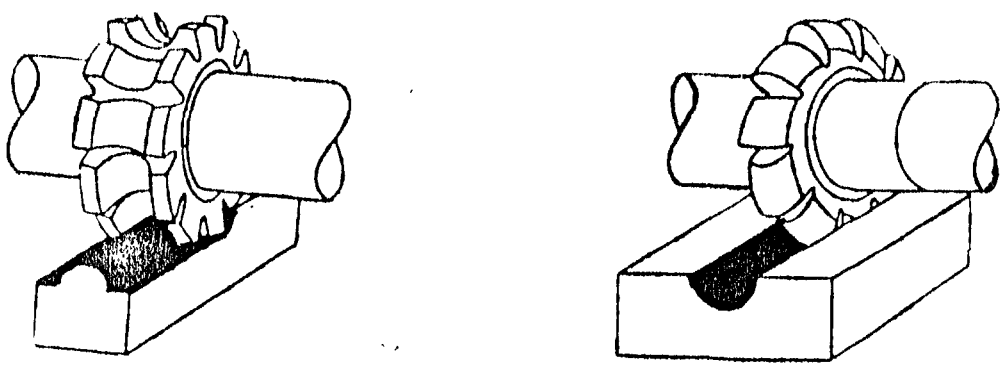
**Fig. 3.3.1. (3) Proceso de mecanizado de una cola de milano.**



**Fuente : Toledo Matus, 1989.**

4. **FRESAS CON FORMA DE RELIEVE.** Estas se fabrican en varias formas para fresar un superficie en forma irregular.

**Fig. 3.3.1 (4) Fresas con forma de relieve.**



**Fuente : Toledo Matus, 1989.**

## 2.3.2. VELOCIDADES DE CORTE Y AVANCE.

La velocidad de corte de una fresa es la que se tiene en su periferia; por lo general se mide en pies o en metros por minuto. No puede establecerse una regla definida para alguna velocidad a la cual una fresa debe funcionar, pues existen demasiados factores que requieren consideración, entre ellos cabe citar:

1. La dureza del material a fresar.
2. La profundidad a la pasada.
3. La magnitud del avance.
4. El material utilizado para fabricar la fresa.
5. La forma, el tamaño y construcción de la pieza a fresar.
6. Las condiciones de la máquina y de la fresa.
7. El empleo de un refrigerante, el tipo de refrigerante y su eficiencia. El refrigerante es una sustancia que se aplica a la herramienta para evitar que se caliente durante el proceso del fresado.

A diferencia de las herramientas de corte utilizadas en los tornos (buriles), los filos de la fresa están sólo un instante de revolución en contacto con la pieza, pudiendo así refrigerarse el tiempo restante de la revolución, pero no se debe permitir que el cortador gire a una velocidad que origine calor excesivo, ya que éste dañaría la herramienta.

La fórmula siguiente se utiliza para determinar las revoluciones por minuto (RPM) a que debe de girar una fresa, para lo que se emplean las velocidades de corte de los materiales.

$$R.P.M. = Vc * C / \pi * D$$

donde :

| | |
|---|---|
| R.P.M. | Revoluciones por minuto. |
| Vc | Velocidad de corte. |
| C | Constante de conversión. |
| $\pi$ | 3.1416. |
| D | Diámetro del cortador. |

Las virutas son arrancadas en el fresado por medio de la rotación de la fresa, cuyos filos están dispuestos en forma circunferencial. Los dientes o filos de una fresa tienen forma de cuña, para conseguir que la fresa pueda arrancar el material por medio del avance de la pieza o en ocasiones por el avance de la herramienta cortante.

## 2.4. CLASIFICACIÓN DE LAS FRESADORAS.

Existe una extensa variedad de fresadoras que difieren en dimensiones y tipo. Cada tipo específico es especialmente adecuado para una clase de trabajo en particular. Se clasifican en tres grupos:

1. Fresadoras de columna y cartela.
2. Fresadoras de gran producción.
3. Fresadoras especiales.

### 2.4.1. FRESADORAS DE COLUMNA Y CARTELA.

Son las más comunes. Son máquinas para trabajos en general con una gama completa de velocidades y avances que pueden regularse lo mismo a mano que automáticamente. Dentro de las fresadoras de columna y cartela existen también tres estilos de fresadoras:

#### 2.4.1.1 FRESADORA HORIZONTAL. (ver fig. 2.4.1.1).

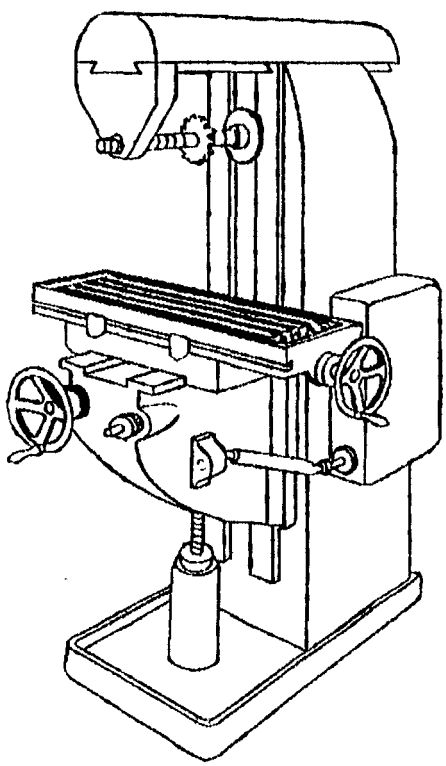
La característica principal de la máquina es que el husillo está colocado horizontalmente. En este tipo de fresadora la mesa realiza dos movimientos que son el transversal (paralela al husillo) y

vertical (hacia arriba y hacia abajo con respecto al husillo), es decir que la pieza puede moverse en direcciones perpendiculares entre sí. La herramienta más usada en este tipo de máquinas es la fresa cilíndrica.

El uso de esta máquina es con piezas que necesitan cortes profundos a velocidad más o menos rápida; esto se puede llevar a cabo gracias a la construcción rígida de la máquina. También es empleada para efectuar fresados genéricos, como alisado de superficies y tallado de ranuras rectas de diferentes perfiles.

**Fig. 2.4.1.1 Fresadora horizontal.**

husillo

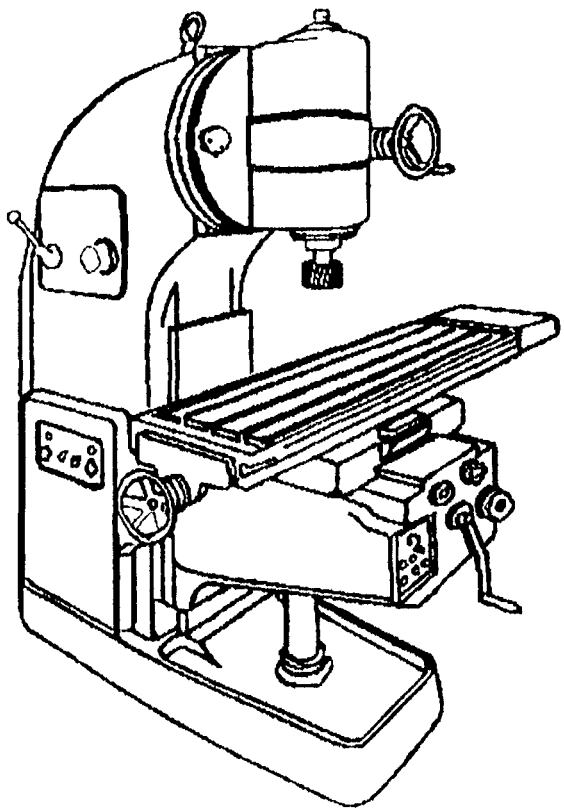


**Fuente: Toledo Matus, 1989**

**2.4.1.2 Fresadora vertical (ver fig. 2.4.1.2).**

Las fresadoras verticales son máquinas muy robustas, especialmente las de gran potencia tienen una forma característica constituida por una pesada columna curvada hacia adelante.

**FIG. 2.4.1.2 Fresadora vertical**



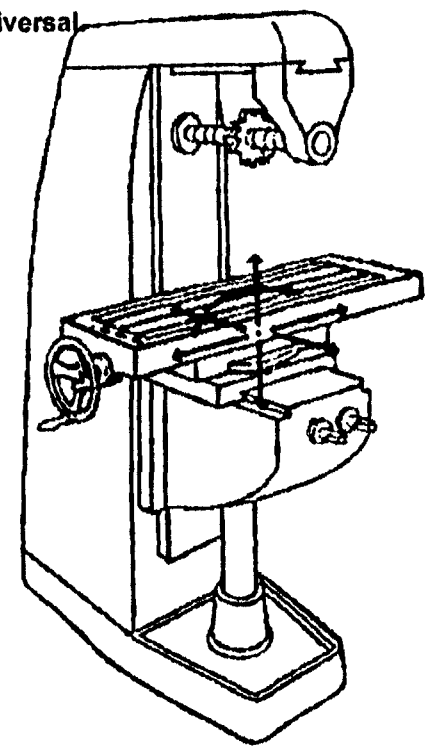
**Fuente : Toledo Matus, 1989.**

En esta máquina el husillo es perpendicular a la superficie de la mesa, es decir, está colocado verticalmente Este husillo tiene un movimiento vertical, y la mesa puede moverse vertical, longitudinal y transversalmente La fresadora de husillo vertical puede usarse para el fresado frontal y de perfiles y fresado de cavidades. En la fresadora vertical se utilizan los cortes o fresas con forma cilindrica

### 2.4.1.3 Fresadora universal (ver fig. 2.4.1.3).

La fresadora universal tiene un aspecto muy parecido al de la fresadora horizontal o corriente. Este tipo de fresadoras tiene la modalidad de que el husillo puede tomar todas las inclinaciones posibles respecto a la mesa de la máquina.

**Figura 2.4.1.3 Fresadora Universal**



**Fuente : Toledo Matus, 1989.**

### 2.4.2. FRESADORAS DE GRAN PRODUCCIÓN.

Las fresadoras de gran producción se usan principalmente para producir piezas mecanizadas en grandes cantidades, pudiéndose aplicar en una extensa variedad de operaciones de fresado. Estas máquinas pueden utilizarse para maquinar superficies planas y mecanizar perfiles especiales por medio de una combinación de fresas montadas en el árbol portafresas. El husillo gira sobre baleros situados en el soporte del mismo, el cual puede moverse verticalmente sobre unas guías mecanizadas.

### 2.4.3. FRESADORAS ESPECIALES.

Las fresadoras especiales pueden ser muy similares a las de gran producción, con la diferencia de que éstas máquinas tienen aspectos especiales en su construcción, como la fresadora copiadora La fresadora copiadora está adaptada con algunos aditamentos que nos permiten maquinar piezas de forma compleja. El uso de esta máquina es principalmente para la producción de modelos para fundición, moldes de plástico, etc.

# CAPITULO 3. EL DISEÑO Y LA MANUFACTURA ASISTIDOS POR COMPUTADORA

Dado el objetivo fundamental de este trabajo, que es el de desarrollar un sistema que automatice la programación de la máquina fresadora EMCO -EMCOTRONIC que se encuentra en el laboratorio de manufactura avanzada del Instituto Tecnológico de la Ciudad de Puebla, se presentan en este capítulo los conceptos básicos y fundamentales del diseño y la manufactura asistidos por computadora ( CAD-CAM) y de los sistemas que en ellas se emplean. Se habla también del control numérico, el cuál es un proceso de control de máquinas-herramienta a través de un programa. Este capítulo está basado en el libro de Computer Aided Manufacturing de Tien Chien Chang (1990) y en la tesis Generador de Códigos de Control Numérico del Ingeniero Francisco Monterrubio Herrera (Instituto Politécnico Nacional, 1993).

## 3.1. EL DISEÑO Y LA MANUFACTURA ASISTIDOS POR COMPUTADORA.

Las actividades de diseño y manufactura son tareas que requieren de precisión y en las que se invierte mucho tiempo, tanto para llevarlas a cabo, como para revisarlas y hacer las modificaciones pertinentes en caso necesario.

El diseño asistido por computadora (en inglés Computer Aided Design - CAD) se puede definir simplemente como el uso de la computadora en el proceso de diseño. En el proceso de diseño, una computadora puede ser usada tanto en el análisis, la síntesis y la representación de los diseños.

La manufactura asistida por computadora (en inglés Computer Aided Manufacturig - CAM) es el uso de la computadora en el proceso de la manufactura. La manufactura es el conjunto de operaciones y actividades correlacionadas, las cuales incluyen el diseño del producto, la selección del material, la

planeación, la producción, la inspección, la dirección y la mercadotecnia para las industrias manufactureras.

Los sectores productivos, para alcanzar sus metas, poco a poco van adquiriendo recursos más sofisticados y modernos, como lo son los sistemas CAD/CAM, que no sólo garantizan velocidad en tales procesos, sino también un elevado grado de confiabilidad en los resultados.

## 3.2. LOS SISTEMAS DE DISEÑO ASISTIDO POR COMPUTADORA.

Un sistema de CAD está formado de tres partes principales :

1. Hardware, que consiste de una computadora y dispositivos de entrada y salida.
2. Paquete de CAD, que es el software aplicado.
3. Sistema Operativo, que es el software que enlaza el paquete de CAD y el hardware.

Las aplicaciones de CAD pueden variar de un software a otro, ya que unos pueden ser sólo editores de dibujo, otros pueden tener entre sus utilerías funciones de análisis de esfuerzos o también de cinemática. Tales aplicaciones pueden agruparse de la siguiente manera:

1. Modelado geométrico.
2. Análisis de ingeniería.
3. Revisión del diseño.
4. Trazados automáticos.

A continuación se explica cada uno de ellos :

1. **El modelado geométrico** es una técnica que consiste en proveer a la computadora de la descripción geométrica de una parte. Los sistemas de CAD convencionales son modelos geométricos que usan modelos de superficie orientada en dos o tres dimensiones. En el modelado geométrico se crean las figuras mediante el uso de comandos de dibujo, edición, visualización, etc.

2. **El análisis de ingeniería** consiste en simular las condiciones a las que se someterá la pieza que se esté diseñando, para observar su comportamiento y poder realizar los cambios que sean

pertinentes, ya sea de dimensiones o de materiales, pudiéndonos apoyar en una base de datos que tenga información almacenada sobre las características de diferentes materiales. Un método eficaz para hacer análisis de ingeniería es el de elemento finito, que consiste en dividir el dibujo de una pieza en un gran número de elementos, para que al simular las condiciones en que ésta trabajará, se identifique qué zonas son las más afectadas y proceder a reforzarlas si así se requiere.

3. **La revisión del diseño** consiste en evaluar su funcionalidad cuando una pieza trabaja junto con otras, es decir, para asegurar que todas las piezas ensamblen perfectamente y que no haya obstrucción cuando se presente movimiento entre ellas, lo que se puede hacer si se cuenta con un sistema CAD que tenga la capacidad de hacer una animación representando los movimientos que harán las piezas diseñadas.

4. **Los trazados automáticos** son esquemas que no son elaborados con los comandos de dibujo primarios (líneas, arcos, círculos, etc.) sino con otros que pueden indicar sus dimensiones. Otro tipo de trazados automáticos son los patrones de sombreado y las diferentes vistas que se pueden obtener de una pieza dibujada en tres dimensiones, ya sea la vista superior, la vista frontal, lateral o en isométrico.

## 3.3. EL CONTROL NUMÉRICO.

El control numérico (CN) se define como el control de procesos por medio de números, letras y símbolos que se agrupan en un programa. Tiene su mayor aplicación en los procesos de maquinado como el taladrado, el fresado, el torneado, el afilado de herramientas, etc. Algunas ventajas que ofrecen las máquinas con CN son la exactitud, la repetibilidad y el ahorro de tiempo con respecto a las máquinas convencionales.

Un sistema de control numérico tiene los siguientes componentes:

1. Programa de instrucciones.
2. Unidad de control.
3. Máquina herramienta o proceso a controlar.

Cada uno de éstos se explica a continuación :

1. **El programa de instrucciones.** Es una secuencia detallada de pasos que deberá realizar la máquina controlada. En él se especifican claramente todos los movimientos lineales o circulares, con sus direcciones y velocidades, así como los códigos de operaciones auxiliares (cambio de herramienta, encendido o paro del husillo, activación o desactivación del refrigerante, etc.).

2. **La unidad de control** consiste en un hardware que interpreta el programa de instrucciones y lo traduce en movimientos que la máquina realiza. Las unidades de control varían entre ellas, pero por lo general constan de un lector de cinta magnética o una unidad de discos flexibles, canales de emisión de señales y de retroalimentación con la máquina herramienta y un panel de control. También puede incluir una pantalla en la que se visualizan los movimientos de la máquina y el programa cargado en la memoria. Además de estas características, las unidades de control más modernas también cuentan con:

    - Calculadora, que incluye resolución de triángulos rectángulos y otros cálculos geométricos con la ayuda de editores de dibujo.

    - Simuladores de los procesos que se realizarán en las máquinas-herramienta.

3. **La máquina-herramienta o proceso a controlar** es el dispositivo que va a ejecutar cada instrucción programada para realizar el manejo y maquinado de materiales.

## 3.4. LOS SISTEMAS DE MANUFACTURA ASISTIDA POR COMPUTADORA.

Los sistemas de CAM aplicados al CN tienen por objeto optimar las tareas de maquinado. Con la inclusión de estos sistemas, los programadores de las máquinas-herramienta ya no tienen la necesidad de invertir un gran número de horas de trabajo en elaborar y revisar los programas, porque todas estas actividades se hacen de una manera automática y sin riesgo a través de la computadora. Además, la computadora permite visualizar si existe algún error o si es necesario cambiar el orden de los procesos de manufactura a desarrollar.

Las aplicaciones de los programas de CAM son muy variadas, ya que los diseños de estos sistemas se hacen teniendo en mente diferentes necesidades que se desean satisfacer, por lo que

algunos son más completos que otros. En general, las principales funciones que desarrollan son las siguientes:

- Generar los códigos de control numérico que servirán para controlar las máquinas herramienta.
- Simular los procesos de maquinado.

La generación de los códigos de CN se hace utilizando la base de datos que se forma al elaborar un dibujo en un sistema de CAD. En esta base de datos se almacena toda la información de cada una de las entidades de dibujo desarrolladas. Por ejemplo, algunos de los datos que tiene una línea son: el tipo de entidad de que se trata (línea en este caso) y las coordenadas de sus puntos inicial y final. En este caso, el programa de CAM puede interpretar a esta entidad de dibujo como un maquinado lineal que va del punto de inicio al punto final marcados en la base de datos, con lo que tenemos un maquinado lineal partiendo de un dibujo elaborado en un sistema de CAD. En este punto salta a la vista la necesidad de tener integrados el CAD y el CAM, pues la primera es herramienta de la segunda, ya que al momento de que algún usuario esté trabajando en la generación de códigos de CN, es posible que se identifiquen errores en el diseño y se requiera de un editor de dibujo para corregirlos.

La simulación de los procesos de maquinado se emplea para detectar errores y corregirlos. Ayuda a evitar el peligro de dañar el material, la máquina-herramienta o peor aún, de ocasionar un accidente.

# CAPITULO 4. INTERFASE

Dado el objetivo fundamental de este trabajo, que es el de desarrollar un sistema que automatice la programación de la máquina fresadora EMCO -EMCOTRONIC que se encuentra en el laboratorio de manufactura avanzada del Instituto Tecnológico de la Ciudad de Puebla, se presentan en este capítulo los conceptos básicos y fundamentales de interfase, que es un enlace entre dos elementos iguales o distintos, también presenta las diferentes interfases básicas de que consta el Sistema GC, las cuales son tres: 1) entre el sistema de CAD y el Sistema GC, 2) entre el Sistema GC y el usuario, 3) entre el Sistema GC y la fresadora EMCO.

## 4.1. CONCEPTO DE INTERFASE COMPUTACIONAL.

Una interfase computacional es una herramienta que puede pertenecer al hardware o al software de un equipo de cómputo y tiene como finalidad relacionar dos o más elementos distintos. Por ejemplo, una interfase de tipo usuario-computadora se establece a través de las pantallas. La finalidad de esas pantallas es establecer una comunicación entre el usuario y el paquete, guiándolo para poder trabajar. Por lo general, los paquetes comerciales presentan interfases con formatos amigables al usuario.

Otro tipo de interfase es la que se tiene para comunicar o relacionar dos elementos de hardware, por ejemplo una computadora con una impresora por medio de un conjunto de cables, puertos y software que los controle.

Por último, se menciona la interfase entre dos herramientas computacionales distintas, es decir, entre dos lenguajes o paquetes computacionales. Este tipo de interfase (software) es muy útil, ya que la información resultante de alguna aplicación puede seguir siendo procesada por otra herramienta distinta, simplificando así el trabajo al usuario. Para aclarar este concepto se puede decir que la función de este tipo de interfase es hacer compatible la información para dos o más ambientes de trabajo

distintos, por ejemplo, los archivos en código ASCII pueden ser la interfase entre diferentes procesadores de palabras como: Word Perfect, Works, Chi-Write, Word, etc., debido a que un mismo archivo puede ser generado y procesado en cualquiera de ellos.

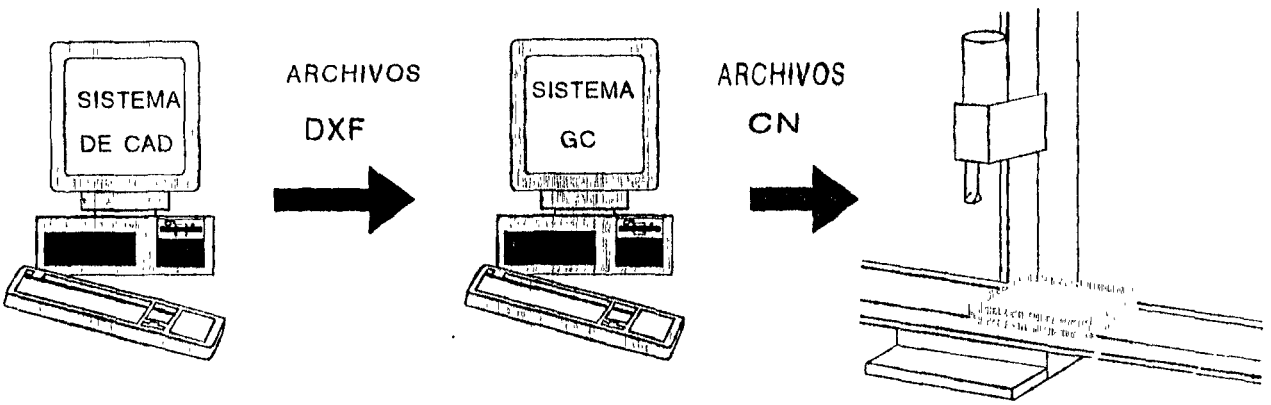
Como se puede ver, cualquier tipo de interfase es siempre útil y ofrece muchas ventajas al usuario. En el Sistema GC se contemplan básicamente tres interfases (ver fig. 4.1) :

1. Entre el Sistema de CAD (AutoCAD) y el Sistema GC la interfase son los archivos DXF, que son generados por el primero y procesados por el segundo.

2. Entre el Sistema GC y el usuario a través de una serie de pantallas que guían a la persona en el uso de este software.

3 Entre el Sistema GC y la fresadora la interfase son los archivos de control numérico, que son generados por el primero y ejecutados por el segundo.

**Figura 4.1 Interfases del Sistema GC**



## 4.2. INTERFASE ENTRE EL SISTEMA DE CAD Y EL SISTEMA GC.

La interfase entre el sistema de CAD y el Sistema GC son archivos con formatos accesibles para ambos.

Los sistemas de CAD abiertos tienen la característica de almacenar sus bases de datos en archivos con diferentes formatos estandarizados. Entre estos formatos tenemos DXB, IGES, DXF, etc. El formato seleccionado como interfase para el Sistema GC fue el DXF debido a que este formato almacena sus archivos en código ASCII y esto los hace de fácil comprensión. Estos archivos son generados en el sistema de CAD, almacenados en disco y procesados en el Sistema GC.

### 4.2.1. ARCHIVOS DXF

DXF (Drawing Exchange File) es un archivo ASCII (texto normal) que contiene toda la información requerida para reconstruir un dibujo. Las **ventajas** de los archivos DXF son:

- Existe mucha información acerca de ellos.
- El formato DXF es universal porque está en código ASCII.
- Es fácil de entender su contenido.
- Pueden estar en diferentes idiomas (inglés, francés, alemán, español o italiano).
- Fácilmente se elimina de ellos la información innecesaria (basura).

La **desventaja** de éstos archivos es que son grandes, comparados con los otros formatos mencionados.

### 4.2.2. ORGANIZACIÓN DE LOS ARCHIVOS DXF.

Un archivo DXF está compuesto de una multiplicidad de grupos, cada uno de los cuales ocupa dos líneas en el archivo DXF. La primera línea de un grupo es el "código del grupo" el cual es un entero positivo o cero. La segunda línea de un grupo es el valor del grupo, presentado en un formato (entero, punto flotante, letrero) que depende del tipo especificado por el código del grupo.

La organización de un archivo DXF presenta cuatro divisiones principales, aunque no son indispensables las cuatro para reconstruir el dibujo.

1. Sección de ENCABEZADO ( HEADER ).
2. Sección de TABLAS ( TABLES ).
3. Sección de BLOQUES ( BLOCKS ).
4. Sección de ENTIDADES ( ENTITIES ).

De estas cuatro secciones, la de entidades es la única necesaria y suficiente para el Sistema GC. De las otras tres se hace una breve referencia y descripción a continuación.

La sección de **ENCABEZADO** es la primera sección que se encuentra en un archivo DXF. Presenta información general acerca del dibujo. Por ejemplo: las coordenadas de los límites del dibujo.

La sección de **TABLAS** contiene los valores de los operandos más complejos del dibujo. Cada parte determina una parte específica de información. Las tablas que incluye son:

- Tabla del tipo de línea (LTYPE).
- Tabla del nivel.
- Tabla del estilo de texto (STYLE).
- Tabla de la vista.
- Tabla del de coordenadas en uso (UCS).

**etc.**

La sección de **BLOQUES** contiene todas las definiciones de los bloques, es decir, las entidades que forman parte de cada uno. Un bloque es un conjunto de entidades que se consideran como un todo y no como entidades separadas.

### 4.2.3. SECCIÓN DE ENTIDADES

La sección de **ENTIDADES** es suficiente para el uso del Sistema GC debido a que contiene todas las entidades del dibujo. Es muy parecida a la sección de BLOQUES.

En esta sección algunos grupos que definen una entidad siempre aparecen; otros son opcionales y aparecen solamente si difieren de los valores de default. Las entidades se encuentran almacenadas en ésta sección una a una, y generalmente en diferente orden al que siguió el usuario cuando creó el dibujo en el de CAD.

Un archivo DXF indica el inicio de una entidad (línea, arco, círculo) con el código cero. Después de encontrar el cero, sabemos que a continuación se presenta el nombre de la entidad, escrito con letras (LINE, ARC, CIRCLE, etc.). Después del nombre deben seguir las coordenadas de las entidades, y para saber de que coordenada se trata antes de cada una de ellas aparece un código, por ejemplo, para la línea tenemos:

| | |
|---|---|
| 0: | nos indica que empieza una entidad. |
| LINE: | la entidad de que se trata. |
| 10: | indica que continúa su coordenada inicial en x. |
| 6.8: | coordenada inicial en x. |
| 20: | indica que continúa su coordenada inicial en y. |
| 20.3: | coordenada inicial en y. |
| 30: | indica la coordenada inicial en z. Si z es diferente de 0, el Sistema GC marca error de código por tratarse de un sistema para dos dimensiones. |
| 16.4: | coordenada inicial en z. |
| 11: | indica que continúa su coordenada final en x. |

8.8: coordenada final en x.

21: indica que continúa su coordenada final en y.

25.8: coordenada final en y.

30: indica su coordenada final en z. Si z es diiferente de 0, el Sistema GC marca error de código por tratarse de un sistema para dos dimensiones.

20 4: coordenada final en z.

A continuación se muestran las entidades que han sido consideradas en el Sistema GC y los códigos de sus coordenadas :

LINE Representa una linea en cualquier sentido. Sus códigos de grupo para dos dimensiones son:

10 - Punto inicial en x.

20 - Punto inicial en y.

11 - Punto final en x.

21 - Punto final en y.

CIRCLE Representa un círculo. Sus códigos de grupo para dos dimensiones son :

10 - Coordenada del centro en x.

20 - Coordenada del centro en y.

30 - Radio.

ARC Representa un arco. Sus códigos de grupo para dos dimensiones son:

10 - Coordenada del centro en x.

20 - Coordenada del centro en y.

40 - Radio.

50 - Ángulo inicial expresado en grados.

51 - Ángulo final expresado en grados.

En cada entidad puede existir información que puede ser ignorada por el Sistema GC por ser innecesaria para el mismo. Por ejemplo, el color de una entidad, que puede ser desplegado en la pantalla de la computadora por un sistema de CAD pero no tiene efecto sobre el maquinado.

### 4.2.4. CUERPO PRINCIPAL DE UN ARCHIVO DXF

```
 0          ;indica que continúa una entidad, entrada o separador.
SECTION     ;indica el inicio de una sección.
 2          ;indica que continúa un nombre o atributo.
HEADER      ;indica el inicio de la sección encabezado.
          <<< Aquí van las variables del encabezado >>>

 0          ; indica que continúa una entidad, entrada o separador.
ENDSEC      ; indica fin de sección ( de HEADER ).
 0          ; indica que continúa una entidad, entrada o separador.
SECTION     ; indica el inicio de una sección.
 2          ; indica que continúa un nombre o atributo.
TABLES      ; indica el inicio de la sección tablas
          <<< Aquí van las variables de las tablas >>>

 0          ; indica que continúa una entidad, entrada o separador.
ENDSEC      ; indica fin de sección ( de TABLES ).
 0          ; indica que continúa una entidad, entrada o separador.
SECTION     ; indica el inicio de una sección.
 2          ; indica que continúa un nombre o atributo.
BLOCKS      ; indica el inicio de la sección bloques
          <<< Aquí van las variables de los bloques >>>
```

```
 0          ; indica que continúa una entidad, entrada o separador.
ENDSEC      ; indica fin de sección ( de BLOCKS ).
 0          ; indica que continúa una entidad, entrada o separador.
SECTION     , indica el inicio de una sección.
 2          ; indica que continúa un nombre o atributo.
ENTITIES    ; indica el inicio de la sección entidades.
        <<< Aquí van las entidades >>>

 0          ; indica que continúa una entidad, entrada o separador.
ENDSEC      , indica fin de sección ( de ENTITIES ).
 0          ; indica que continúa una entidad, entrada o separador.
EOF         , indica fin del archivo.
```

## 4.3. INTERFASE SISTEMA GC-USUARIO.

Esta interfase consiste en una serie de pantallas en modo gráfico donde se despliegan las funciones del Sistema GC, con el objeto de guiar al usuario en el uso del mismo.

## 4.4. INTERFASE GENERADORA DE CÓDIGO DE CONTROL NUMÉRICO-FRESADORA.

El punto que une al Generador de Códigos de Control Numérico con la fresadora son los listados de código de control numérico. Estos listados son producidos en el Sistema GC y utilizados por la fresadora.

Para transportar los listados de CN de la computadora a la fresadora se puede recurrir a alguna de las siguientes opciones:

1. Imprimir el listado y teclearlo en el tablero de la fresadora

2. Desplegar el listado en pantalla desde un editor de ASCII y teclearlo en el tablero de la fresadora.

3. Almacenar los códigos de control numérico en un diskette de 3.5" de baja densidad en un archivo con un formato especial, que alcena varios códigos de CN. Este formato especial es dado por programas comerciales, como el simulador de EMCOTRONIC. Una vez que el diskette contiene los códigos, éste se introduce en el drive de la máquina fresadora.

4. Conectar la computadora y la fresadora con un cable a través del puerto serial y mandar los listados directamente a la máquina-herramienta. En este caso, es necesario que la fresadora cuente con el software y el hardware necesarios, y que a su vez la computadora disponga del software requerido. Esta opción no ha sido contemplada en el desarrollo del Sistema GC.

### 4.4.1. LENGUAJE EMCOTRONIC.

Un programa en el lenguaje EMCOTRONIC se encuentra compuesto de varios bloques. Cada bloque a su vez cuenta con la información necesaria para la ejecución de un comando.

Las palabras dentro de los bloques son combinaciones de letras con dígitos, que especifican qué tarea será realizada con la fresadora. A las letras de las palabras se les conoce también con el nombre de direcciones. Los códigos para las direcciones "G" y "M" son los siguientes :

**CODIGOS "G" .** Algunos de los códigos G o funciones preoperatorias son las siguientes:

| FUNCION PREPARATORIA | DEFINICION |
|---|---|
| G00 | Posicionamiento punto a punto. |
| G01 | Interpolación lineal. |
| G02 | Interpolación circular en sentido de las manecillas del reloj (dos dimensiones). |
| G03 | Interpolación circular en sentido contrario al de las manecillas del reloj (dos dimensio |
| G04 | Pausa temporizada. |

| | |
|---|---|
| G20 | Introducción de datos en pulgadas. |
| G21 | Introducción de datos en milímetros. |
| G28 | Referencia de casa o punto cero de referencia. |
| G40 | Cancela compensación de herramienta. |
| G41 | Compensación de herramienta a la izquierda. |
| G42 | Compensación de herramienta a la derecha. |
| G43 | Compensación de longitud de herramienta (más larga que la original). |
| G81 | Ciclo fijo de taladrado sin desahogo de virutas. |
| G82 | Ciclo fijo de taladrado con pausa en el fondo. |
| G90 | Programación absoluta. |

**CODIGOS "M"**

También existen funciones miscelàneas o auxiliares como son paro de programa, inicio, cambios de herramienta, etc. Mencionaremos solo algunas de ellas :

| FUNCION MISCELANEA | DEFINICION |
|---|---|
| M00 | Paro de programa. |
| M02 | Fin de programa. |
| M03 | Giro del husillo en sentido de las manecillas del reloj. |
| M04 | Giro del husillo en sentido contrario al de las manecillas del reloj. |
| M05 | Detiene el husillo. |

| | |
|---|---|
| M06 | Cambio de herramienta. |
| M08 | Enciende refrigerante. |
| M09 | Cancela refrigerante. |

A continuación se da un ejemplo de un programa en lenguaje EMCOTRONIC y una explicación de las partes que lo componen.

| N | G | X | Y | Z | F |
|---|---|---|---|---|---|
| 00 | 01 | 0 | 0 | -1000 | 120 |
| 01 | 01 | 5000 | 0 | 0 | 120 |
| 02 | 01 | 5000 | 5000 | 0 | 120 |
| 03 | 01 | 0 | 5000 | 0 | 120 |
| 04 | 01 | 0 | 0 | 0 | 120 |
| 05 | 01 | 0 | 0 | 1000 | |

La programación de la fresadora en estudio se hace por medio de códigos que indican al control la acción por desarrollar; tales códigos están clasificados de acuerdo a su propósito en dos grupos de órdenes: las preparatorias y las auxiliares o misceláneas.

- La letra N se utiliza para numerar cada uno de los bloques programados.
- En la columna G se escribe la información codificada de la tarea que debe realizar la fresadora. Existe una gran variedad de funciones G, las cuales se presentan en el apéndice correspondiente.
- Las direcciones X, Y, Z especifican el curso que tomará la herramienta de corte de la fresadora, es decir, son las coordenadas del punto al cual debe llegar la herramienta. Estas direcciones se deben especificar en centésimas de milímetro o bien en milésimas de pulgada.

- La dirección F se utiliza para indicar el avance más adecuado de la herramienta de corte para cada movimiento de arranque de virutas. En la EMCO-EMCOTRONIC el avance se programa en milímetros/minuto o bien en décimas de pulgada/minuto.

- La dirección M se codifica en la misma columna donde se codifican las funciones G, y se les llama funciones adicionales.

- La dirección S se utiliza para establecer la velocidad o número de revoluciones a las que debe girar la herramienta de corte.

- La dirección T se utiliza para decir el número de herramienta que se ocupará.

- Las direcciones J, K son instrucciones que se ocupan para definir un maquinado circular.

# CAPITULO 5. FRESADORA EMCO-EMCOTRONIC.

Dado el objetivo fundamental de este trabajo, que es el de desarrollar un sistema que automatice la programación de la máquina fresadora EMCO que se encuentra en el laboratorio de manufactura avanzada del Instituto Tecnológico de la Ciudad de Puebla, se presentan en este capítulo los conceptos básicos y fundamentales de esta fresadora, sus características y los elementos necesarios para ejecutar el fresado.

## 5.1. CARACTERÍSTICAS DE LA FRESADORA EMCO-EMCOTRONIC.

La fresadora EMCO empleada para desarrollar el Sistema GC es de tipo didáctico, pues su uso se restringe a materiales blandos y a profundidades de fresado pequeñas, ya que su capacidad no se compara con la de máquinas industriales. Es una fresadora vertical, pues el husillo o eje portaherramientas es perpendicular a la superficie de la mesa.

El tablero de control de esta fresadora permite la introducción manual de los datos del programa de CN a través de teclas y perillas. Cuenta con teclas para mover los carros sentido longitudinal, transversal y vertical (control manual), teclado alfanumérico para la captura de los programas, tecla de paro de emergencia, unidad de lectura y escritura de diskettes de 3.5" de doble densidad, cuyo objetivo es el de almacenar los programas que han sido introducidos a la máquina-herramienta a través del teclado.

## 5.2. ELEMENTOS NECESARIAS PARA EJECUTAR EL FRESADO.

Los elementos necesarios para poder ejecutar el fresado son:

- la máquina-herramienta

- el material en el que se va a fresar
- el listado formado por las instrucciones de control numérico, que pudo ser generado en el Sistema GC.

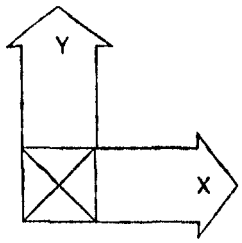
Para generar este último, es necesario que el usuario especifique una serie de datos. Estas especificaciones están basadas en las características de la pieza a fresar y son: origen, altura de seguridad, refrigerante, velocidad de corte, compensación de herramienta, profundidad total del fresado, profundidad por pasada y herramienta.

A su vez, con la ayuda de esta información es posible calcular las revoluciones por minuto, avance y número de pasadas. A continuación se describe cada una de ellas.

### 5.2.1. ORIGEN

Es el punto de referencia desde el cual se registran todas las acotaciones de las trayectorias de fresado, es decir, todos los cálculos de desplazamiento del cortador. En el Sistema GC el origen se representa con el símbolo siguiente :

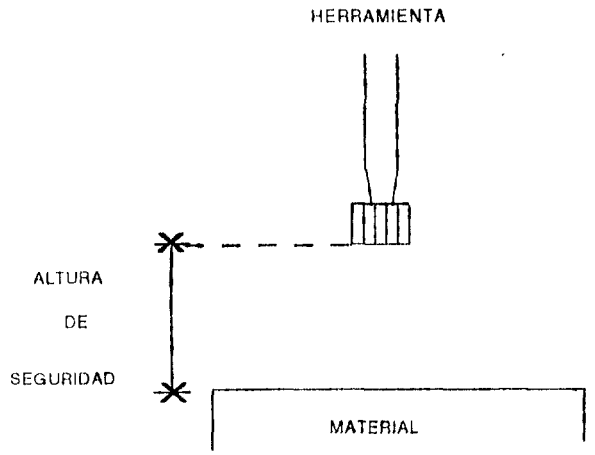
**Figura 5.2.1 ORIGEN**




### 5.2.2. ALTURA DE SEGURIDAD.

Es la distancia que debe subir el cortador sobre el material a trabajar para no causar daños cuando se requieran movimientos sin arranque de viruta, pues solo se hacen para posicionar la herramienta en un lugar determinado.

**Figura 5.2.2 Altura de seguridad**




### 5.2.3. REFRIGERANTE.

Cuando una herramienta de corte está en contacto con el metal hay una elevada fricción, lo que genera calor y la herramienta tiende a ablandarse por lo que se daña su filo. El refrigerante, que por lo general es aceite se aplica a la herramienta para enfriarla y evitar este problema.

### 5.2.4. MATERIAL DE LA PIEZA A MECANIZAR.

Como ya se mencionó, se recomienda que los materiales que se trabajen en esta máquina no sean muy duros, con el fin de no forzar los motores de la fresadora. Los materiales que se recomiendan son : aluminio, magnesio, cobre, plomo y plásticos. Las velocidades de corte, que son indispensables para los cálculos de avances y de revoluciones por minuto, deben mantenerse de acuerdo al manual de la fresadora, dentro de las siguientes especificaciones :

44 m/min Para metales blandos, como aluminio.

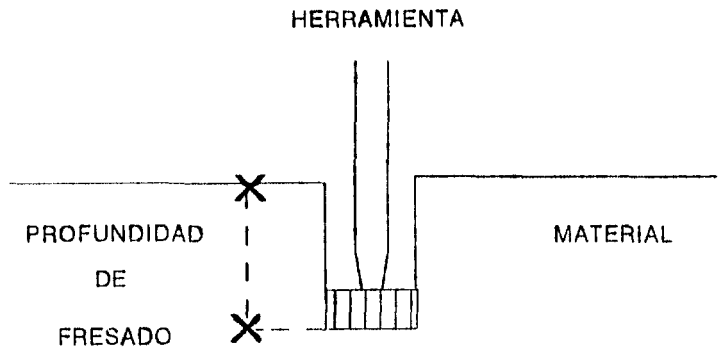
35 m/min Para plástico blando.

25 m/min Para plástico duro.

### 5.2.5. PROFUNDIDAD

Hay dos tipos de profundidades, la profundidad total y la profundidad por pasada. La primera se define como la distancia total que penetra la fresa en el material. La segunda es la distancia que baja el cortador cada vez pasa cortando el material, la suma de varias pasadas debe alcanzar la profundidad total (ver figura 5.2.5).

**FIGURA 5.2.5. PROFUNDIDAD**




### 5.2.6. HERRAMIENTAS

En la máquina fresadora en estudio se permite el uso de brocas y diferentes tipos de cortadores y de todo tipo de cortadores o fresas (verticales, de disco, de forma, etc.). El equipo cuenta con cambio automático de herramientas, por lo que no se necesita programar un movimiento a una posición de cambio de herramienta (casa).

# CAPITULO 6. DESARROLLO DEL SISTEMA GC

Dado el objetivo fundamental de este trabajo, que es el de desarrollar un sistema que automatice la programación de la máquina fresadora EMCO que se encuentra en el laboratorio de manufactura avanzada del Instituto Tecnológico de la Ciudad de Puebla, se presentan en este capítulo los conceptos fundamentales del desarrollo del Sistema GC, la justificación de su desarrollo, la definición del proyecto, los alcances y limitaciones, así como los manuales técnico y del usuario.

## 6.1. JUSTIFICACIÓN DEL SISTEMA GC

En la carrera profesional en Ingeniería Industrial-Mecánica que se imparte en el Instituto Tecnológico de la Cd. de Puebla (ITP), el proceso de aprendizaje para la fabricación de piezas mecánicas se desarrolla con el uso de máquinas fresadoras convencionales. Este proceso de aprendizaje consiste de diseñar en papel la pieza a maquinar, y a partir del modelo, manipular secuencialmente la fresadora para obtener el resultado deseado. Las autoridades del ITP han decidido modernizar este proceso educativo mediante la adquisición y uso de máquinas-fresadoras automatizadas, las cuales obedecen, a través de un sistema computarizado, a una serie de instrucciones dadas por el usuario que en conjunto forman un programa de CN.

Los programas para las máquinas de CNC pueden ser codificados a través de dos caminos: manual o automatizado. En el proceso manual el operador parte del dibujo de la pieza a fabricar que contenga las dimensiones de la misma, enlista las instrucciones en código de CN y las introduce en la fresadora a través del tablero de la misma. La programación automatizada se realiza a través de sistemas CAD-CAM (Diseño y Manufactura Asistidos por Computadora).

El personal de la escuela utiliza la forma manual de programación de la máquina fresadora EMCO debido a que no cuentan con un sistema automatizado que optimice este procedimiento La desventaja de la programación manual es la dificultad que se presenta en el control absoluto sobre las

identidades geométricas requeridas para el desarrollo de alguna pieza. El tiempo que utilizan es largo y los resultados muchas veces no son alentadores. Este problema puede ser minimizado mediante el uso de un sistema CAM que permita al usuario pasar automáticamente de un archivo creado en un sistema CAD a la generación del código de CN listo para ser introducido en el tablero de la fresadora y ser ejecutado por la misma.

## 6.2. DEFINICIÓN DEL SISTEMA GC

El Sistema GC es un sistema de CAM que en conjunto con un sistema de CAD (AutoCAD), facilita a los usuarios de la fresadora EMCO-EMCOTRONIC la programación de piezas en dos dimensiones con profundidad constante, incrementando la rapidez, y sencillez con respecto al sistema tradicional. En si, el sistema CAD-CAM-CN se presenta en tres fases principales:

1. El uso de algún CAD que ya exista en el mercado y que permita almacenar sus archivos en formato DXF, por ejemplo AutoCAD.

2. El uso de un CAM que correrá en cuatro fases principales:

- Cargar a la memoria y despliegue de un archivo que el usuario seleccione a la vez. Estos archivos deben estar almacenados en formato DXF y pueden ser creados en el paquete de CAD.

- Captura de variables que establecen las condiciones de fresado: selección de herramienta, compensación de herramienta, altura de seguridad, uso de refrigerante, profundidad de fresado, datos del material.

- Seleccionar una trayectoria de fresado o bien permitir al sistema generarla automáticamente.

- Generación de los siguientes resultados: representación gráfica del fresado y la generación del código de CNC. Salida del código a impresora.

3. Introducción del listado de CN a la fresadora para ser ejecutado.

## 6.3. ALCANCES

- El sistema corre bajo ambiente DOS.

- El sistema es capaz de extraer archivos solo en formato DXF.

- El sistema solo toma en cuenta archivos en dos dimensiones y con entidades independientes, es decir, no considera bloques de entidades como las polilíneas.
- El sistema interpreta el archivo DXF, despliega en pantalla la gráfica sin permitir modificaciones en la misma.
- Todos los datos requeridos por la fresadora tienen que ser proporcionados necesariamente por el usuario.
- La representación gráfica del fresado no toma en cuenta tiempo.
- El código de CN generado solo puede ser almacenado en un archivo de trabajo o darle salida a impresora.
- El sistema es limitado solo para la fresadora EMCO de lenguaje EMCOTRONIC.

# MANUAL TÉCNICO DEL SISTEMA GC

## I. ESTRUCTURA GENERAL DEL SISTEMA.

El Sistema GC forma parte de un sistema mayor CAD-CAM-CN que, en conjunto, permite generar códigos de control numérico apartir de archivos CAD para se ejecutados en la fresadora correspondiente. En el ANEXO 2 se muestra un diagrama de flujo de datos de éste sistema y en el ANEXO 3 un diagrama de entidad relación

El Sistema GC está programado en lenguaje C de manera estructurada y descendente. Fué desarrollado en Turbo C de Borland, en una computadora personal (PC) compatible con IBM 80386 SX, con un sistema operativo DOS versión 5.0. Está formado por un archivo ejecutable SGC.EXE compilado a partir de veintiséis archivos organizados de la siguiente manera:

- Un archivo proyecto ( SGC.PRJ) y su auxiliar (SGC.DSK).

- Veinte archivos de código fuente que ejecutan las diferentes opciones del Sistema GC.

- Cuatro archivos que contienen el código de las librerías propias del Sistema GC.

Los nombres de todos los archivos empiezan con las letras "SGC" que significan Sistema Generador de Códigos. El archivo de proyecto (SGC.PRJ) y su auxiliar (SGC.DSK) contienen la información necesaria para construir el Sistema GC desde el editor de Turbo C. En SGC.PRJ se almacenan los nombres de todos los archivos a compilar y a ligar que forman el archivo ejecutable (SGC.EXE), es decir, los archivos a que a que se hace referencia en los puntos dos y tres de la clasificación. La opción de "proyectos" de Turbo C permite crear los ejecutables de códigos de gran tamaño.

En los archivos a que se refiere el segundo punto de la clasificación anterior, después de la parte alfabética, sigue una parte numérica cuyo orden se muestra en el diagrama de navegación del ANEXO 1. También se presenta un diagrama de transición de estados en el ANEXO 4. A continuación se desglosan estos nombres, así como el número de página correspondiente en este manual :

| Nombre-archivo | # página | Función |
|---|---|---|
| SGC000.C | 50 | Programa principal |
| SGC100.C | 50 | Carga archivos *.DXF |
| SGC101.C | 54 | Carga archivos *.F1 |
| SGC110.C | 54 | Salva archivos *.F1 |
| SGC120.C | 54 | Renombra archivos |
| SGC130.C | 54 | Borra archivos |
| SGC210.C | 54 | Origen |
| SGC220.C | 55 | Altura |
| SGC230.C | 55 | Refrigerante |
| SGC240.C | 55 | Material |
| SGC300.C | 55 | Trayectoria |
| SGC301.C | 55 | Trayectoria manual |
| SGC302.C | 55 | Trayect automática |
| SGC310.C | 56 | Herramienta |
| SGC320.C | 56 | Profundidad |
| SGC400.C | 56 | CódigoCN |
| SGC410.C | 57 | Simular |
| SGC420.C | 57 | Imprimir |

Los nombres de los archivos de las librerías están formados por dos partes alfabéticas. La primera por "SGC", y la segunda por "L" (de librería) y la letra inicial del nombre de las librerías. Así, tenemos cuatro librerías

| | | |
|---|---|---|
| SGCLA.C | Librería de archivos | 58 |
| SGCLC.C | Librería de captura de datos | 58 |
| SGCLG.C | Librería de graficos | 59 |
| SGCLM C | Librería de menús | 59 |

Librerías de Turbo C que llama el Sistema GC, a través de los archivos anteriores:

| | | | | | |
|---|---|---|---|---|---|
| stdio.h | graphics.h | stdlib.h | dos.h | dir.h | string.h |
| conio.h | math h | alloc.h | ctype.h | bios.h | |

## II. ARCHIVOS DEL SISTEMA.

### ARCHIVO SGC000.C

Este archivo almacena el programa principal del Sistema GC. Incluye los diferentes archivos que conforman el sistema (archivos nombrados en la página anterior) y las librerías propias del lenguaje C. Declara constantes, variables globales y estructuras de datos. Su función es inicializar el Sistema GC y dirigir la ejecución del mismo de manera global.

SUBRUTINAS QUE CONTIENE ESTE ARCHIVO (a la derecha de cada uno se encuentra el número de página en el cual se encuentra su descripción) :



### ARCHIVO SGC100.C

Su función es cargar archivos con extensión .DXF Los archivos son almacenados en una lista ligada, en donde en cada nodo se encuentran los parámetros de una entidad. además existen en la lista nodos de control. La lista ligada está organizada de la siguiente manera :

Cada nodo tiene los campos :

| CAMPO | Tipo de datos | CAMPO | Tipo de datos |
|---|---|---|---|
| entidad | - cadena de caracteres | a1 | - tipo real |
| x1 | - tipo real | a2 | - tipo real |
| y1 | - tipo real | bloques | - tipo entero |
| x2 | - tipo real | next | - tipo apuntador |
| y2 | - tipo real | before | - tipo apuntador |
| h | - tipo real | ESH | - tipo apuntador |
| k | - tipo real | EST | - tipo apuntador |
| r | - tipo real | | |

La lista ligada inicia siempre con el nodo INICIO. El nodo INICIO se presenta solo una vez en toda la lista y sirve para almacenar los datos generales para el fresado. Utiliza los siguientes campos del nodo:

| CAMPO | Dato que almacena |
|---|---|
| entidad | - la palabra INICIO |
| x1 | - Origen en X |
| y1 | - Origen en Y |
| r | - altura |
| a1 | - refrigerante |
| a2 | - velocidad de corte |
| next | - apunta al siguiente nodo |
| before | - apunto a una dirección nula |

Los nodos DATOS son aquellos que almacenan los datos necesarios para el fresado de un bloque definido de entidades. Todas las entidades (líneas, arcos, círculos) que se encuentren después de un nodo de DATOS y antes de otro nodo de DATOS, se fresan con los datos que aparecen en el primero. Utiliza los siguientes campos del nodo:

| CAMPO | Dato que almacena |
|---|---|
| entidad | - La palabra DATOS. |
| y1 | - Profundidad por pasada. |
| x2 | - Tipo de herramienta. 0=broca, 1=cortador. |
| y2 | - Refrigerante. |
| h | - Avance de la herrmienta. |
| k | - Número de aristas cortantes. |
| r | - Diámetro. |
| a1 | - Número de la herramienta. |
| bloques | - Número de bloque activo. |
| next | ' Apunta al siguiente nodo de la lista. |
| before | - Apunta al nodo anterior de la lista. |

Los nodos de LINE (línea) almacenan los parámetros de líneas. Utiliza los siguientes campos del nodo:

| CAMPO | Dato que almacena |
|---|---|
| entidad | - La palabra LINE. |
| x1 | - Coordenada inicial sobre el eje x. |
| y1 | - Coordenada inicial sobre el eje y. |
| x2 | - Coordenada final sobre el eje x. |
| y2 | - Coordenada final sobre el eje y. |
| next | - Apunta al siguiente nodo de la lista. |
| before | - Apunta al nodo anterior de la lista. |
| ESH | - Apuntador auxiliar para el ordenamiento automático de las entidades. |
| EST | - Apuntador auxiliar para el ordenamiento automático de las entidades. |

Los nodos de ARC (arco) almacenan las coordenadas necesarias para arcos. Los campos del nodo que utilizan son:

| CAMPO | Dato que almacena |
|---|---|
| entidad | - La palabra ARC. |
| x1 | - Coordenada inicial sobre el eje x. |
| y1 | - Coordenada inicial sobre el eje y. |
| x2 | - Coordenada final sobre el eje x. |
| y2 | - Coordenada final sobre el eje y. |
| h | - Coordenada del centro sobre el eje x. |
| k | - Coordenada del centro sobre el eje y. |
| r | - Radio. |
| a1 | - Ángulo inicial. |
| a2 | - Ángulo final. |
| next | - Apunta al siguiente nodo de la lista. |
| before | - Apunta al nodo anterior de la lista. |

ESH - Apuntador auxiliar para el ordenamiento automático de las entidades.

EST - Apuntador auxiliar para el ordenamiento automático de las entidades.

Los nodos de CIRCLE (círculo) almacenan las coordenadas necesarias para arcos. Los campos del nodo que utilizan son:

**CAMPO** **Dato que almacena**

entidad - La palabra CIRCLE.

h - Coordenada del centro sobre el eje x.

k - Coordenada del centro sobre el eje y.

r - Radio

next - Apunta al siguiente nodo de la lista.

before - Apunta al nodo anterior de la lista.

ESH - Apuntador auxiliar para el ordenamiento automático de las entidades.

EST - Apuntador auxiliar para el ordenamiento automático de las entidades.

SUBRUTINAS QUE CONTIENE ESTE ARCHIVO (a la derecha de cada uno se encuentra el número de página en el cual se encuentra su descripción) :

**ARCHIVO SGC101.C**

Su función es dirigir la lectura de un archivo con extensión .F1. Utiliza las subrutinas del archivo anterior.

SUBRUTINAS QUE CONTIENE ESTE ARCHIVO (a la derecha de cada uno se encuentra el número de página en el cual se encuentra su descripción) :



**ARCHIVO SGC110.C**

Su función es ejecutar la opción de "Salvar" del Sistema GC La subrutina que conforma este archivo es : SalvaDXFF1 () y su descripción se encuentra en la página 66

**ARCHIVO SGC120.C**

Su función es ejecutar la opción de "Renombrar" del Sistema GC. Está formado por la subrutina RenombrarArchivo () y su descripción se encuentra en la página 66 de este manual.

**ARCHIVO SGC130.C**

Su función es ejecutar la opción de "Borrar" del Sistema GC. Está formado por la subrutina BorrarArchivo () y su descripción se encuentra en la página 67 de este manual.

**ARCHIVO SGC210.C**

Su función es dirigir la opción de "Origen", la cual se encarga de seleccionar la posición del origen de coordenadas

SUBRUTINAS QUE CONTIENE ESTE ARCHIVO (a la derecha de cada uno se encuentra el número de página en el cual se encuentra su descripción) :

**ARCHIVO SGC220.C**

Su función es ejecutar la opción de altura del Sistema GC. Está formado por la subrutina Altura () y su descripción se encuentra en la página 69 de este manual.

**ARCHIVO SGC230.C**

Su función es ejecutar la opción de "Refrigerante" del Sistema GC. Está formado por la subrutina Refrigerante () y su descripción se encuentra en la página 70 de este manual.

**ARCHIVO SGC240.C**

Su función es ejecutar la opción de "Material" del Sistema GC. Está formado por la subrutina Material () y su descripción se encuentra en la página 70 de este manual.

**ARCHIVO SGC300.C**

Su función es dirigir la ejecución de la opción "Trayectoria" del Sistema GC. Está formado por la subrutina Trayectoria () y su descripción se encuentra en la página 70 de este manual.

**ARCHIVO SGC301.C**

Su función es ejecutar la opción de Selección manual de trayectoria de fresado".

SUBRUTINAS QUE CONTIENE ESTE ARCHIVO (a la derecha de cada uno se encuentra el número de página en el cual se encuentra su descripción)



**ARCHIVO SGC302.C**

Su función es ejecutar la operación de selección de entidades de manera automática.

SUBRUTINAS QUE CONTIENE ESTE ARCHIVO (a la derecha de cada uno se encuentra el número de página en el cual se encuentra su descripción) :



**ARCHIVO SGC310.C**

Su función es ejecutar la opción de "Herramienta" del Sistema GC.

SUBRUTINAS QUE CONTIENE ESTE ARCHIVO (a la derecha de cada uno se encuentra el número de página en el cual se encuentra su descripción) :



**ARCHIVO SGC320.C**

Su función es ejecutar la opción de "Profundidad" del Sistema GC. Está formado por la subrutina Profundidad () cuya descripción se encuentra en la página 78 y por la subrutina LeeProfundidad que se encuentra en la página 78 de este manual.

**ARCHIVO SGC400**

Este archivo se encarga de dirigir la conversión a código de control numérico. Sus subrutinas son:

SUBRUTINAS QUE CONTIENE ESTE ARCHIVO (a la derecha de cada uno se encuentra el número de página en el cual se encuentra su descripción)



**ARCHIVO SGC410**

Se encarga de realizar la simulación del fresado.

VARIABLES GLOBALES DEL SISTEMA GC QUE INCLUYE ESTE ARCHIVO.

SUBRUTINAS QUE CONTIENE ESTE ARCHIVO (a la derecha de cada uno se encuentra el número de página en el cual se encuentra su descripción)



**ARCHIVO SGC420.C**

Se encarga de mandar los archivos en código ASCII a la impresora. Está formado por la subrutina imprimir () y su descripción se encuentra en la página 85 de este manual.

## III. LIBRERÍAS PROPIAS DEL SISTEMA GC

### ARCHIVO SCGLA.C

En el se encuentran almacenadas las subrutinas que ayudan al manejo de archivos. Contiene la declaración del nodo que, en conjunto con otros nodos del mismo tipo, forman una lista ligada con los nombres de los archivos de un directorio específico. Los campos declarados en el nodo son :

| CAMPO | Tipo de dato | Dato que almacena |
|---|---|---|
| name | cadena de caracteres | almacena el nombre del archivo. |
| next | apuntador | almacena la dirección del siguiente nodo. |
| before | apuntador | almacena la dirección al nodo anterior. |

SUBRUTINAS QUE CONTIENE ESTE ARCHIVO (a la derecha de cada uno se encuentra el número de página en el cual se encuentra su descripción) :



### ARCHIVO SGCLC.C

Este archivo almacena las funciones necesarias para la captura de datos. Contiene las siguientes subrutinas :

## ARCHIVO SGCLG.C

Este archivo contiene las subrutinas propias del Sistema GC que manejan gráficos. A continuación se despliegan sus nombres y números de página en éste manual :



## ARCHIVO SGCLM.C

Este archivo almacena las subrutinas correspondientes al manejo de menús. Son las siguientes

# IV. SUBRUTINAS DEL SISTEMA SGC.

A continuación se describe cada una de las subrutinas.

## SUBRUTINA main ()

FUNCIÓN : Este es el programa principal. Se encarga de dirigir de manera general la ejecución del Sistema GC.
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Initialize (), pantalla_principal(), inicializa_raton(), pon_posicion_raton(), cursor_on(), CoordenadasMouse(), movimiento_raton(), lee_raton(), lee_teclado(), FreeDXF().
PROCESO: Después de inicializar variables el programa principal se carga de inicializar el sistema a través del llamado de las siguientes subrutinas : Initialize(), pantalla_principal(), inicializa_raton(), pon_posicion_raton(), cursor_on(), CoordenadasMouse (). Mediante un ciclo, identifica que tipo de dispositivo de entrada está usando el usuario : teclado o mouse, y llama a la subrutina correspondiente para cada uno de éstos.
Finalmente libera la memoria usada por la lista ligada y restaura el modo de video.

## SUBRUTINA lee_raton ()

FORMATO : void lee_raton(void);
FUNCIÓN . Este es el programa principal Se encarga de dirigir de manera general la ejecución del Sistema GC
DATOS DE ENTRADA : Ninguno
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA . izquierdab_pulsado(), espera_on (), RecorreMenuP (), AbreMenu(), CierraVentana(), EjecutaSubMenu(), derechab_pulsado(), Salida().

## SUBRUTINA lee_teclado ()

FORMATO : lee_teclado(void);
FUNCIÓN : Esta subrutina ejecuta el Sistema GC cuando el usuario utiliza como dispositivo de entrada el teclado . Identifica que tarea quiere ejecutar el usuario de acuerdo al nivel actual y la tecla que pulsó
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA . RecorreMenuP (), AbreMenu (), ayuda (), RecorreMenus(), CierraVentana (), EjecutaSubMenu (), Salida ().

PROCESO : Inicializa variables. Captura el valor pulsado en el teclado. Si la tecla no fue ESC identifica en que nivel se encuentra el usuario.

Si se encuentra en el nivel 0 recorre el menú principal a la derecha o a la izquierda o despliega un submenú si se tecleo la flecha hacia abajo. Si se tecleó F10, despliega la ayuda. Si se tecleó la letra 'A' se despliega el menú de ARCHIVO, si se tecleó la letra 'I' se despliega el menú de INICIO, si se tecleó la letra 'P' se despliega el menú de PROCESO, si se tecleó la letra 'S' se despliega el menú de SALIDAS. Si se tecleó "retorno de carro" se despliega el submenú desplegado en diferente color.

Si el nivel es 1 se recorren los submenús hacia arriba o hacia abajo, o bien se despliega un nuevo submenú si se tecleó IZQUIERDA o DERECHA. O llama a "ayuda()". Ejecuta alguna opción de los submenús si se tecleó alguna letra o "retorno de carro".

Si se tecleó ESC, se da fin al programa si se encuentra el usuario en el nivel 0. O se cierra la ventana abierta si el nivel actual es 1 o 2.

**SUBRUTINA EjecutaSubMenu ()**

FORMATO : void EjecutaSubMenu(void);
FUNCIÓN : Esta subrutina llama a la subrutina que ejecuta la tarea especifica que el usuario quiere.
DATOS DE ENTRADA : Opción del menú principal. Opción del submenú.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : cursor_off(), CargaDXF (), GraficaDXF (), SalvaDXFF1 (), RenombrarArchivo(), BorrarArchivo(), Salida(), MueveOrigen (), Altura(), Refrigerante(), material(), Trayectoria (), Profundidad (), Herramienta (), cn (), simular(), imprimir(), cursor_on().
PROCESO : Inicializa variables. Llama a la subrutina correspondiente de acuerdo a la opción del menú y submenú en que se encuentre el usuario

| Opción del menú | Opción del submenú | |
|---|---|---|
| 0 | | |
| | 0) CargaDXF (), GraficaDXF () | 3) BorrarArchivo() |
| | 1) SalvaDXFF1 () | 4) Salida() |
| | 2) RenombrarArchivo() | |
| 1 | | |
| | 0) MueveOrigen () | 2) Refrigerante() |
| | 1) Altura() | 3) Material() |
| 2 | | |
| | 1) Trayectoria () | 3) Profundidad () |

2) Herramienta ()

3

1) cn ()
2) simular()
3) imprimir()

**SUBRUTINA Salida ()**

FORMATO : void Salida(void);
FUNCIÓN : Esta subrutina pregunta al usuario si realmente desea salir del Sistema GC.
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : salva_ventana(), WindowG (), CierraVentana().
PROCESO : Pregunta al usuario si desea salir del Sistema GC. Si quiere salir, restaura el video y sale con la función "exit()".
Si no quiere salir, continua ejecutando el sistema.

**SUBRUTINA ayuda ()**

FORMATO : void ayuda(int);
FUNCIÓN : Esta subrutina despliega la ayuda.
DATOS DE ENTRADA : Número de la ayuda.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : cursor_off(), salva_ventana (), WindowG (), CierraVentana (), cursor_on().
PROCESO : Traza la ventana de ayuda. Abre el archivo correspondiente de ayuda: cnchelp1 o cnchelp2 y despliega la ayuda.

**SUBRUTINA CargaDXF ().**

FORMATO : void CargaDXF(char[]);
FUNCIÓN : Esta subrutina se encarga de cargar un archivo DXF. Lee de disco el archivo DXF o F1, el cual se encuentra en código ASCII y almacena los datos en una lista doblemente ligada.
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : El nombre del archivo y su ruta en la variable RutaTotal. Una lista doblemente ligada, en la cual, cada nodo contiene los datos de una entidad de dibujo.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : GetFileName (), mensajes (), pausa (), leeDXF(), inicia_headdxf (), entidades(), FreeDXF().
PROCESO: Después de inicializar variables, esta subrutina lee el nombre del archivo a cargar a través de la subrutina "GetFileName". Verifica que el nombre del archivo tenga la extensión DXF o F1. Si no

hay error, abre el archivo correspondiente. Dentro de un ciclo lee cada entidad del archivo y la almacena en la lista ligada llamando a la subrutina "entidades". Cierra el archivo.

**SUBRUTINA VerificaSiEntidadDentroDeLimites ().**

FORMATO : int VerificaSiEntDentroDeLimites (int);
FUNCIÓN : Verifica si la entidad se encuentra de los límites de la mesa de la fresadora.
DATOS DE ENTRADA : El número de la entidad.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Si se trata de una línea recta, verifica si los extremos de la entidad están dentro de los límites del material representados lógicamente en la pantalla. Si se trata de un arco, se hace un ciclo que revisa punto por punto el arco, para cada punto se verifica si se encuentra dentro de los límites. Si se círculo, se verifica si las coordenadas del centro no sobrepasan los límites del material.

**PROCESO:SUBRUTINA FreeDXF().**

FORMATO : void FreeDXF(void);
FUNCIÓN : Esta subrutina da de baja la lista headdxf-taildxf.
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO: Mediante un ciclo se recorre toda la lista "headdxf-taildxf" y se libera la memoria reservada para cada nodo.

**SUBRUTINA entidades ().**

FORMATO . void entidades (int*);
FUNCIÓN : Esta subrutina identifica el tipo de entidad de dibujo del archivo DXF o F1.
DATOS DE ENTRADA : El archivo DXF.
DATOS DE SALIDA : Si hay o no error en el archivo.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : leeDXF(), DatosInicio(), DatosProceso (), linea(), circulo(), arc ().
PROCESO: Se inicializan variables. Se lee un dato del archivo. En un ciclo se identifica el tipo de dato leído y se llama a la subrutina correspondiente. El tipo de dato puede ser la palabra "INICIO", "DATOS", "LINE", "CIRCLE", "ARC", "ENDSEC". Por cada entidad se crea un nodo que se añade a la lista, en el cual se almacenan los parámetros de la linea, arco o círculo.

**SUBRUTINA inicia_headdxf ()**

FORMATO : void inicia_head_dxf(void);

FUNCIÓN : Inicializa la lista DXF-F1.
DATOS DE ENTRADA : Nodo a insertar.
DATOS DE SALIDA : El nodo insertado en la lista DXF-F1.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : FreeDXF ().
PROCESO : Libera la lista DXF-F1, para inicializarla nuevamente. Reserva memoria para un nuevo nodo de la lista DXF-F1. Inicializa los apuntadores del nodo con NULL. Almacena los demás valores del nodo.

**SUBRUTINA linea ()**

FORMATO : void linea (char *, int*)
FUNCIÓN : Almacenar en la lista ligada los valores leídos para la línea.
DATOS DE ENTRADA : Archivo DXF.
DATOS DE SALIDA : Aumenta un nodo en la lista ligada con los datos de la línea.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : InsertaNodoDXF (), leeDXF ().
PROCESO : Declara de variables. Inserta un nuevo nodo en la lista ligada. Dentro de un ciclo se recorre parte del archivo hasta encontrar el número 10 (primer dato de línea). A continuación se almacenan en la lista los demás valores de la línea, identificándolos con los códigos 20, 11, 21. También se identifican errores en el archivo.

**SUBRUTINA circulo ()**

FORMATO . void circulo ( char *, int *)
FUNCIÓN : Almacenar los parámetros del circulo en la lista ligada.
DATOS DE ENTRADA : Archivo DXF.
DATOS DE SALIDA : Aumenta un nodo en la lista ligada con los datos del circulo.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : InsertaNodoDXF (), leeDXF ().
PROCESO : Declara de variables. Inserta un nuevo nodo en la lista ligada. Dentro de un ciclo se recorre parte del archivo hasta encontrar el número 10 (primer dato del circulo). A continuación se almacenan en la lista los demás valores de la círculo, identificándolos con los códigos 20, 40. También se identifican errores en el archivo.

**SUBRUTINA arco ()**

FORMATO : void arc ( char*, int *)
FUNCIÓN : Almacenar los parámetros del arco en la lista ligada.
DATOS DE ENTRADA : Archivo DXF.
DATOS DE SALIDA : Aumenta un nodo en la lista ligada con los datos del arco.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : InsertaNodoDXF (), leeDXF ().
PROCESO : Declara de variables. Inserta un nuevo nodo en la lista ligada. Dentro de un ciclo se recorre parte del archivo hasta encontrar el número 10 (primer dato del arco). A continuación se

almacenan en la lista los demás valores del arco, identificándolos con los códigos 20, 40, 50, 51. También se identifican errores en el archivo.

**SUBRUTINA InsertaNodoDXF ()**

FORMATO : void InsertaNodoDXF (struct nododxf *)
FUNCIÓN : Inserta un nodo en la lista ligada DXF-F1.
DATOS DE ENTRADA : Nodo a insertar.
DATOS DE SALIDA : El nodo insertado en la lista DXF-F1.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Si la lista esta vacía inicializa la lista con el nuevo nodo. Si la lista no esta vacía inserta el nuevo al final de la lista.

**SUBRUTINA leeDXF**

FORMATO . void leeDXF(int *g,float *g1, TYPE *t,int *error)
FUNCIÓN · Lee del archivo el código y su valor
DATOS DE ENTRADA : El archivo DXF.
DATOS DE SALIDA : g=codigo de la funcion, g1 = valor flotante, s = cadena de caracteres, no. del mensaje de error
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA :
PROCESO · Lee un dato del archivo. si leyó un número menor a 10 o 999, lee el siguiente dato el cuál debe ser una cadene · ARC, CIRCLE, DATOS, INICIO, ENTITIES, ENDSEC.

**SUBRUTINA DatosInicio ()**

FORMATO · void DatosInicio (void)
FUNCIÓN : Inicializa la lista DXF-F1 con los datos de inicio : origen, altura de seguridad, refrigerante y material.
DATOS DE ENTRADA : El archivo DXF - F1.
DATOS DE SALIDA : El primer nodo de la lista DXF-F1.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO . Lee los datos de inicio del archivo DXF - F1 y los almacena en un nodo, el cual inserta en la lista ligada.

**SUBRUTINA DatosProceso ()**

FORMATO . void DatosProceso (void)
FUNCIÓN : Almacena en la lista DXF-F1 los datos de proceso : compensación, profundidad y herramienta.
DATOS DE ENTRADA : El archivo DXF - F1.

DATOS DE SALIDA . El segundo nodo de la lista DXF-F1.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Lee la compensación, profundidad y herramienta y los almacena en un nodo, el cual inserta en la lista ligada.

**SUBRUTINA SalvaDXFF1 ()**

FORMATO : void SalvaDXFF1 (void)
FUNCIÓN : Almacena la lista ligada en el disco.
DATOS DE ENTRADA : ninguno.
DATOS DE SALIDA :El archivo F1.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : salva_ventana (), WindowG(), AjustaFileName (), CierraVentana(), cursor_on(), mensajes (), pausa().
PROCESO : Inicializa variables y despliega una ventana. Pregunta al usuario el nombre del archivo a salvar, lo lee y lo valida. Si no hay error abre el archivo. Recorre la lista ligada y almacena en el disco dato por dato junto con sus códigos y letreros.

**SUBRUTINA RenombrarArchivo ()**

FORMATO : void RenombrarArchivo (void)
FUNCIÓN : Asigna el nuevo nombre que el usuario quiere dar a algún archivo.
DATOS DE ENTRADA : Nombre del archivo a renombrar.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : salva_ventana (), WindowG (), CierraVentana (), GetFileName (), separa (), CapturaNombre (), mensajes ().
PROCESO : Después de declarar variables, esta subrutina captura el nombre del archivo que el usuario quiere renombrar. Si no hay error, captura el nuevo nombre valida que sean consistentes las rutas y renombra.

**SUBRUTINA BorrarArchivo ()**

FORMATO : void BorrarArchivo (void)
FUNCIÓN : Borra del disco el archivo seleccionado por el usuario.
DATOS DE ENTRADA : Nombre del archivo a borrar.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : salva_ventana (), WindowG (), CierraVentana (), GetFileName (), separa (), CapturaNombre (), mensajes ()
PROCESO : Declaración de variables. Captura el nombre del archivo que se desea borrar. Pregunta al usuario si realmente lo quiere eliminar y si accede entonces lo borra.

**SUBRUTINA MueveOrigen ()**

FORMATO : void MueveOrigen (void)
FUNCIÓN :
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : CierraVentana(), PuntoFinalEntidad(), salva_ventana, GraficaOrigen(), pon_posicion_raton (), MuestraCoordenadas(), desplazamiento(), cursor_on(), GraficaDXF (), mensajes ()
PROCESO : Declaración de variables. Verifica si hay algún archivo DXF-F1 activo en la lista ligada. Si sí, despliega tres opciones:

dar el punto Final de la entidad ( P ), dar las coordenadas exactas ( C ) o desplazarse sobre la pantalla ( D ) Dependiendo de la opción que se escogió, se llama a la subrutina correspondiente.

**SUBRUTINA desplazamiento ()**

FORMATO : void desplazamiento (void)
FUNCIÓN : Captura el origen que el usuario da, desplazándolo con el MOUSE o las flechas.
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : movimiento_raton (), OrigenMouse (), MuestraCoordenadas (), tecla_pulsada (), OrigenFlechas ().
PROCESO : Declaración de variables. Se inicia un ciclo dentro del cual se lee la nueva posición que el usuario quiere para el origen con el MOUSE o con las flechas.

**SUBRUTINA OrigenFlechas ()**

FORMATO : void OrigenFlechas (void)
FUNCIÓN . Captura el origen que el usuario da, desplazándolo con las flechas
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : flechas (), GraficaOrigen (), CierraVentana (), salva_ventana (), pon_posicion_raton ().
PROCESO : Declaración de variables. Se lee la tecla que se oprimió, si fué RETURN se dibuja el origen en pantalla en su nueva posición y se finaliza la operación. Si fué ESC, se dibuja el origen en su posición original. Sí se tecleó una flecha, se avanza el origen en la dirección deseada siempre y cuando la nueva posición se encuentre dentro de los límites de la pantalla. Si se oprimió F10, se llama a la ayuda.

**SUBRUTINA OrigenMouse ()**

FORMATO : void OrigenMouse (void)
FUNCIÓN : Captura el origen que el usuario da, desplazándolo con el mouse.
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : posicion_raton (), CierraVentana(), GraficaOrigen (), salva_ventana (), derechab_pulsado (), espera_on (), izquierdab_pulsado ().
PROCESO : Se verifica si el usuario está desplazando el mouse u oprimiendo sus teclas. Si está desplazando el mouse, se desplaza simultáneamente el origen en pantalla, respetando los límites del área de trabajo. Si se oprimió el botón izquierdo, se coloca nuevamente el origen en su posición original. Si se oprimió el botón derecho se traza el origen en su nueva posición.

**SUBRUTINA OrigenEnEntidad**

FORMATO : void OrigenEnEntidad (struct nododxf *pnodo, int x, int y)
FUNCIÓN : Busca en que extremo de la entidad el usuario quirere el origen.
DATOS DE ENTRADA : El nodo de la entidad a analizar, x,y las coordenadas del punto seleccionado por el usuario en pantalla.
DATOS DE SALIDA : Las coordenadas del origen en el nodo de INICIO.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : GráficaOrigen().
PROCESO: Calculo de las distancias del punto seleccionado a ambos extremos de la entidad. Compara las dos distancias y almacena las coordenadas correspondientes a la menor distancia en el nodo de INICIO.

**SUBRUTINA PuntoFinalEntidad**

FORMATO : void PuntoFinalEntidad()
FUNCIÓN : Dirige el proceso de seleccion de entidad y en que extremo de esa entidad se encuentra el origen.
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : SeleccionarEntidad(), Origen EnEntidad().
PROCESO : Llama la subrutina de seleccionar entidad y a la subrutina que identifica en que extremoa de la entidad se encontrará el origen.

**SUBRUTINA LeeCoordenadaExacta ()**

FORMATO : float LeeCoordenadaExacta (char, int, int, int, int)
FUNCIÓN : Captura de teclado un número real.
DATOS DE ENTRADA : Las coordenadas de la ventana donde se va a capturar el número.

DATOS DE SALIDA : El número leído.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : WindowG(), ayuda (), mensajes ().
PROCESO : Se inicializa la variable Num con la coordenada actual del origen, puede ser X o Y, según corresponda, se convierte a cadena de caracteres y se despliega el valor en pantalla. Se inicia un ciclo, dentro del cuál se captura de teclado el nuevo número real. Durante la captura el usuario puede teclear ESC, para salir de la ventana de captura, BS que es la tecla que generalmente, en los paquetes comerciales borra el caracter anterior. F10, que la llave que llama a la ayuda y cualquier número o el punto decimal.

**SUBRUTINA Altura ()**

FORMATO : void Altura ()
FUNCIÓN : Captura la altura de seguridad de fresado.
DATOS DE ENTRADA : La altura en el nodo de inicio : headdxf->r
DATOS DE SALIDA : La altura almacenada en el nodo de inicio : headdxf->r.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : salva_ventana(), WindowG (), Real_Cadena (), LeeNumeroReal(), CierraVentana(), cursor_on(), mensajes ().
PROCESO : Después de leer variables, se despliega un letrero que pide al usuario la altura de seguridad en mm. y se lee su respuesta.

**SUBRUTINA Refrigerante ()**

FORMATO : void Refrigerante (void);
FUNCIÓN : Pregunta al usuario si hay o no refrigerante.
DATOS DE ENTRADA . La bandera de refrigerante : headdxf->a1 (0=No, 1=Si).
DATOS DE SALIDA . La bandera de refrigerante : headdxf->a1 (0=No, 1=Si).
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : salva_ventana(), WindowG (), CierraVentana(), mensajes ().
PROCESO Se despliega el valor actual del refrigerante, ("s" o "n"), el cual se encuentra almacenado en forma numérica ("0" o "1") en headdxf->a1. Se inicia un ciclo de lectura, el cual finaliza con la lectura de 's', 'S', 'n', 'N', ESC o RETURN.

**SUBRUTINA Material()**

FORMATO : void Material (void);
FUNCIÓN : Captura la velocidad de corte del material.
DATOS DE ENTRADA : La velocidad de corte del material almacenada en headdxf->a2.
DATOS DE SALIDA : La nueva velocidad de corte del material almacenada en headdxf->a2.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : cursor_off(), salva_ventana(), WindowG(), Real_Cadena() LeeNumeroReal(), CierraVentana(), cursor_on (), mensajes ().

PROCESO : Declara variables, despliega un letrero que pide la nueva velocidad de corte del material. Captura el nuevo dato

**SUBRUTINA Trayectoria ()**

FORMATO : void Trayectoria (void);
FUNCIÓN : Captura el tipo de trayectoria (Manual/Automático (M/A) y queda almacenado en la variable "trayectoria".
DATOS DE ENTRADA : El tipo de trayectoria.
DATOS DE SALIDA : El nuevo tipo de trayectoria.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA:cursor_off(), salva_ventana(), WindowG(), CierraVentana(), ayuda (),TrayecManual(), TrayecAutomatica(), mensajes ().
PROCESO Después de declarar variables, se despliega el mensaje "Trayectoria Manual o Automática (M/A )", se captura el nuevo tipo de trayectoria y se ejecuta TrayecManual() o TrayecAutomatica(), según corresponda.

**SUBRUTINA TrayecManual ()**

FORMATO : int TrayecManual (int );
FUNCIÓN : Permite la selección manual de entidades para su fresado posterior en el orden que desee el usuario
DATOS DE ENTRADA .El número de bloque a seleccionar.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : du_nodo(), CreaDatos(), SeleccionarEntidad(), cursor_on(), TrazaSentido().
PROCESO:Verifica si la selecc6n ya ha sido inicializada anteriormente. Si no es así, inicialel vector 'bloques' con el primer número de bloque que ha sido dado por el usuario e inicializa el primer nodo ("INICIO") de la lista ligada.
Si la selección ya habia sido inicializada se coloca el apuntador "aux" en la posición correcta de la lista ligada para hacer las modificaciones requeridas. Dentro de un ciclo se seleccionan una a una las entidades a agregar, se añaden en la lista y se imprimen en pantalla.

**SUBRUTINA CreaDatos ()**

FORMATO . struct nododxf *CreaDatos (struct nododxf *)
FUNCIÓN : Crea un nod de DATOS en la lista ligada.
DATOS DE ENTRADA . el apuntador a la direccón en la lista ligada en que se va a añadir el nuevo nodo.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA :Ninguna.

PROCESO· Crea un nodo del tip nododxf. Inicializa sus campo con valores de default para un nodo "DATOS"
Liga el nodo en la lista ligada.

**SUBRUTINA BorraBloque ()**

FORMATO : void BorraBloque (int );
FUNCIÓN : Borra un bloque de la lista ligada de selección de entidades por trayectoria manual.
DATOS DE ENTRADA : El número de bloque a borrar.
DATOS DE SALIDA . Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Coloca un apuntador en el nodo que inicia la parte de la lista a borrar. Borra nodo por nodo y restaura la lista de datos. Actualiza el vector "bloques"

**SUBRUTINA Sentido ()**

FORMATO : struct nododxf *Sentido (struct nododxf *pnodo, int x, int y)
FUNCIÓN : Determina el sentido de fresado de una entidad de acuerdo a la parte de la entidad en que se colocó el icono seleccionador.
DATOS DE ENTRADA : La entidad a la cual se le va a determinar su sentido, las coordenadas del en la pantalla punto que el usuario seleccionó
DATOS DE SALIDA · Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO Se calculan las dos distancias entre el punto seleccionado y los dos puntos extremos de la entidad
Se comparan las dos distancias y se hace un intercambio de coordenadas si es necesario. Es necesario cuando las coordenadas correspondientes a la distancia menor están como coordenadas finales de la entidad.

Como Turbo C restringe el trazo de arcos a sentido antihorario, en caso de un fresado en sentido horario, se almacenan los datos correctamente (horario) pero se invierten en el momento de trazarlos en pantalla. La bandera indicadora de este caso es el encontrar los ángulos del arco a trazar en pantalla o simular con signo negativo.

**SUBRUTINA dup_nodo ()**

FORMATO : struct nododxf *dup_nodo (struct nododxf *)
FUNCIÓN · Crea un nodo destino y copia en él los datos del nodo fuente.
DATOS DE ENTRADA Nodo fuente.
DATOS DE SALIDA : Nodo destino.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA · Ninguna.

PROCESO . Esta subrutina crea un nuevo nodo y copia campo por campo en él los datos del nodo fuente.

**SUBRUTINA TrayecAutomatica ()**

FORMATO : void TrayecAutomatica (void);
FUNCIÓN : Busca una trayectoria de fresado.
DATOS DE ENTRADA : Lista fuente : lista ligada con las entidades a fresar.
DATOS DE SALIDA : Lista destino : Lista ligada con las entidades a fresar ya ordenadas.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Free (), SeparaBarrenos (), pausa (),recursion (), AumentaBarrenos (), GraficaDXF ().
PROCESO : Después de declarar variables, se inicializan con NULL las listas auxiliares : headList2, headList3, headList4. Inserta en una lista nueva los barrenos (CIRCLES), mediante el llamado de la función SeparaBarrenos (). Recorre la lista fuente para ordenar las entidades LINEs y ARCs en una segunda nueva lista. Finalmente une la lista de ARCs y LINEs al final de la lista de barrenos.

**SUBRUTINA SeparaBarrenos ()**

FORMATO : void SeparaBarrenos (void);
FUNCIÓN : Esta Subrutina separa los barrenos (CIRCLEs) de la lista ligada headdxf y los almacena en la lista HeadBroca - TailBroca.
DATOS DE ENTRADA : Lista fuente : lista ligada con las entidades a fresar.
DATOS DE SALIDA : Lista destino : Lista ligada HeadBroca -TailBroca con los barrenos a fresar.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : dup_nodo().
PROCESO : Después de declarar variables, recorre la lista fuente y genera dos listas : 1a) HeadBroca - TailBroca que es la lista donde se separan los datos de inicio (INICIO), los datos de fresado para los barrenos (DATOS) y los datos de fresado para el resto de las entidades. 2a) Theaddxf-Ttaildxf que es la lista donde se agrupan solo los LINEs y ARCs para poder ser ordenados posteriormente.

**SUBRUTINA recursion ()**

FORMATO : void recursion (struct nododxf *, char, int)
FUNCIÓN : Esta subrutina busca ordenar las entidades de dibujo. Trata de encontrar una ruta conveniente de fresado. Trabaja solamente para arcos y líneas.
DATOS DE ENTRADA : Lista ligada con las entidades a ordenar.
DATOS DE SALIDA : La lista ligada con las entidades ordenadas.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Free (), InsertaRecursion (), FaltaE (), EntidadesSeguidasCola(), dup_nodo (), recursion (), EntidadesSeguidasCabeza(), relocaliza_headaux(), IntercambiaListaMenor().

PROCESO : Esta subrutina es recursiva. Primero inserta un nodo en una lista destino, busca si existe al menos un nodo en la lista fuente que no haya sido ordenado. Si lo encuentra, busca colocarlo a la cola o a la cabeza de la lista destino. Una vez terminada esta tarea, busca ordenar la siguiente entidad.

**SUBRUTINA IntercambiaListaMenor ()**

FORMATO : void IntercambiaListaMenor ()
FUNCIÓN : Intercambia la nueva lista headList1 por headList2, headList3 o headList4, siempre y cuando tenga menor número de bloques.
DATOS DE ENTRADA : Listas ligadas headList1, headList2, headList3, headList4.
DATOS DE SALIDA : Listas ligadas headList1, headList2, headList3, headList4.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : duplicaL1 ()
PROCESO :
- Si la lista headList1 tiene menor número de bloques que headList2 o bien si esta última no existe se liberan headList2, headList, headList3 y headList4. Se duplica headList1 en headList2. Si no:
- Si la lista headList1 tiene el mismo número de bloques que headList2 y headList3 está en NULL, se duplica headList1 en headList3. Si no:
- Si headList4 está en NULL, se duplica la lista headList1 y se almacena en headList4.

**SUBRUTINA duplicaL1 ()**

FORMATO : void duplicaL1 (void)
FUNCIÓN : Duplica la lista headList1.
DATOS DE ENTRADA : Lista headList1.
DATOS DE SALIDA : Lista dupHeadL1.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : dup_nodo ()
PROCESO : Al recorrer la lista headList1, se duplica cada uno de sus nodos y se almacenan en la lista dupHeadL1.

**SUBRUTINA EntidadesSeguidasCola ()**

FORMATO void EntidadesSeguidasCola (void);
FUNCIÓN : Forma una lista (ESeguidas) con todas aquellas entidades cuyas coordenadas coinciden con las de la última entidad ordenada.
DATOS DE ENTRADA : Lista fuente.
DATOS DE SALIDA : Lista ligada ESeguidas.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : FaltaE (), DistanciaPuntos(), dup_nodo ()
PROCESO : Se forma la lista ESeguidas con la entidades cuyas coordenadas coinciden exactamente con las coordenadas de la entidad ordenada al final.

**SUBRUTINA EntidadesSeguidasCabeza ()**

FORMATO : void EntidadesSeguidasCabeza (void);
FUNCIÓN : Forma una lista (ESeguidas) con todas aquellas entidades cuyas coordenadas coinciden con las de la primera entidad ordenada.
DATOS DE ENTRADA : Lista fuente.
DATOS DE SALIDA : Lista ligada ESeguidas.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : FaltaE (), DistanciaPuntos(), dup_nodo ().
PROCESO : Se forma la lista ESeguidas con la entidades cuyas coordenadas coinciden exactamente con las coordenadas de la entidad ordenada al principio.

**SUBRUTINA DistanciaPuntos ()**

FORMATO : float DistanciaPuntos (float, float, float, float)
FUNCIÓN : Esta subrutina encuentra la distancia entre los puntos (x1,y1) y (x2,y2).
DATOS DE ENTRADA : Las coordenadas de los puntos (x1,y1) y (x2,y2).
DATOS DE SALIDA : La distancia entre ambos puntos.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA . Ninguna.
PROCESO : Calculo de la distancia entre puntos.

**SUBRUTINA InsertaRecursion ()**

FORMATO : void InsertaRecursion (struct nododxf *, char, int);
FUNCIÓN Inserta un nuevo nodo en la lista headListL1.
DATOS DE ENTRADA : El nodo a insertar y la lista headListL1.
DATOS DE SALIDA : La lista headListL1 con el nodo insertado.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Se crea un nuevo nodo, se inserta en la lista headListL1 en la posición que le corresponda.

**SUBRUTINA FaltaE()**

FORMATO : struct nododxf *FaltaE (struct nododxf *auxdxf)
FUNCIÓN : Identifica cual es la siguiente entidad a ordenar de la lista headdxf.
DATOS DE ENTRADA : el apuntador "auxdxf" que apunta a la lista fuente de entidades (headdxf) y el apuntador "auxL1" que apunta a la lista : headList1.
DATOS DE SALIDA : La entidad a ordenar.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Después de declarar variables, se inicia un ciclo que recorre la lista fuente de entidades "headdxf" iniciando el recorrido desde el apuntador auxiliar auxdxf. A su vez, dentro de este ciclo se recorre la lista headList1. En cada iteración, se inevestiga si las entidades coinciden. Si no coinciden se

da fin a la subrutina FaltE() dando como valor de retorno el nodo que de la lista headdxf que no coincidió Si al menos una entidad no coincide, se da fin a la subrutina retornando NULL.

**SUBRUTINA AumentaBarrenos ()**

FORMATO : void AumentaBarrenos (void);
FUNCIÓN : Busca si la última entidad ordenada en la lista headList1 es el inicio de un nuevo bloque. Si sí, asigna un nuevo barreno (CIRCLE) para la fácil penetración del cortador que fresará esa nueva serie de entidades
DATOS DE ENTRADA . La lista ligada headList1.
DATOS DE SALIDA :
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA . Ninguna.
PROCESO :

**SUBRUTINA Free ()**

FORMATO : void Free (struct nododxf *head)
FUNCIÓN : Elimina una lista ligada.
DATOS DE ENTRADA : La lista ligada a eliminar.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Recorre la lista ligada eliminando nodo por nodo.

**PROCESO: SUBRUTINA Herramienta()**

FORMATO void Herramienta (void);
FUNCIÓN : Captura el tipo de herramienta : 0=Broca, 1=Cortador en PDatos->y2. Y llama a las subrutinas correspondientes para la lectura del resto de los datos.
DATOS DE ENTRADA : La lista ligada de entidades.
DATOS DE SALIDA . NInguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA: LeeHerramTrayAut(), LeeHerramTrayMan(), mensajes(), LeeNumeroBloque().
PROCESO . Se identifica si se trató de una trayectoria manual o automática. Si fué automática se llama a la trayectoria de LeeHerramTrayAut(), si no, se inicia un ciclo que lee números de bloque y para cada uno de éstos se llama a la subrutina de LeeHerramTrayMan().

**LeeHerramientaTrayMan**

FORMATO . int LeeHerramientaTrayMan (),
FUNCIÓN : Lee la herramienta y demás datos para nodos de trayectoria manual.
DATOS DE ENTRADA : El nodo de la lista ligada de entidades a modificar.

DATOS DE SALIDA El nodo modificado.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : salva_ventana(), WindowG (), Real_Cadena (), LeeNumeroReal (), CierraVentana (), ayuda (), broca(), cortador(), CierraVentana().
PROCESO Despliega una ventana que pide el número de la herramienta. Lee el nuevo número de la herramienta.
Despliega otra ventana que preguna si se trata de "Broca o Cortador (B/C): ", y lee. Lee los datos restantes de las herramientas mediante el llamado de las subrutinas broca() o cortador(), según corresponda.

**LeeHerramientaTrayAut**

FORMATO
FUNCIÓN : int LeeHerramientaTrayAut (struct nododxf *).
DATOS DE ENTRADA : El nodo de la herramienta a actualizar
DATOS DE SALIDA : El nodo actualizado
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA :
FORMATO : void Herramienta (void);
FUNCIÓN : Captura el tipo de herramienta : 0=Broca, 1=Cortador en PDatos->y2. Y llama a las subrutinas correspondientes para la lectura del resto de los datos.
DATOS DE ENTRADA : La tipo de herramienta anterior.
DATOS DE SALIDA : El nuevo tipo de herramienta.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA: cursor_off(), salva_ventana(), WindowG(), ayuda () CierraVentana(), broca(), cortador(), cursor_on(), mensajes ().
PROCESO : Después de declarar variables, se despliega el tipo de herramienta : Broca o Cortador (B/C), y se captura el nuevo tipo. Si la respuesta fué "B", se ejecuta la subrutina broca(), si la respuesta fué "C", se ejecuta la subrutina cortador().

**SUBRUTINA broca()**

FORMATO : void broca (void);
FUNCIÓN : Captura datos para la broca : El diámetro en PDatos->r. El avance en PDatos->h.
DATOS DE ENTRADA : El diámetro y el avance de la broca
DATOS DE SALIDA : El nuevo diámetro y avance de la broca.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : WindowG (), Real_Cadena (), LeeNumeroReal().
PROCESO . Después de declarar variables, se despliega el diámetro de la broca y se lee el nuevo valor A continuación se despliega el avance de la broca (mm./seg) y se captura el nuevo dato

**SUBRUTINA cortador ()**

FORMATO : void cortador (void);
FUNCIÓN : Captura los datos del cortador y los almacena en la lista ligada : el diámetro en PDatos->r, el avance por diente del cortador en PDatos->h y el número de aristas cortantes en PDatos->k.
DATOS DE ENTRADA : Los datos del cortador.
DATOS DE SALIDA : Los nuevos datos del cortador.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : WindowG (), Real_Cadena (), LeeNumeroReal().
PROCESO : Después de declarar variables, se despliega el diámetro del cortador y se lee el nuevo valor. A continuación se despliega el avance del cortador en (mm/diente del cortador) y se captura el nuevo dato. Por último se despliega y lee el número de aristas cortantes.

**SUBRUTINA Profundida()**

FORMATO . void Profundidad (void);
FUNCIÓN : Dirige la selección del nodo DATOS para el cual se va a leer la profundidad y la captura de ésta.
DATOS DE ENTRADA : La lista ligada de entidades.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : LeeProfundidad
PROCESO: Declaración de variables. Si la selección de la trayectoria fué automática, se llama dos veces la subrutina LeeProfundidad: una para la broca y otra para el cortador. Si la trayectoria fué manual, se inicia un ciclo de lectura de número de bloque, para saber que valor se va a actualizar. Se coloca el apuntador en el nodo DATOS correcto y se la profundidad.

**SUBRUTINA LeeProfundidad ()**

FORMATO : int LeeProfundidad();
FUNCIÓN : Captura la profundidad total y por pasada en PDatos.
DATOS DE ENTRADA : La compensación en headdxf->x1.
DATOS DE SALIDA : La compensación en headdxf->x1.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA: cursor_off(), salva_ventana(), WindowG(), CierraVentana(), ayuda ().
PROCESO: Despliega una ventana que captura la profundidad total y la almacena en el nodo. Posteriormente despliega una ventana para la captura la profundidad por pasada.

**SUBRUTINA cn ()**

FORMATO void cn (void);
FUNCIÓN · Dirige la generación del código de control numérico.
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : inicio (), pausa (), datosCN (), circuloCN (), lineaCN(), arcoCN ().
PROCESO : Se declaran variables. Se llama a la subrutina de inicio. Se almacenan en el archivo ".cn" las primeras líneas del programa, que son las que inicializan la fresadora. Dentro de un ciclo se recorre la lista ligada que interpreta nodo por nodo de acuerdo a las palabras: "DATOS", "CIRCLE", "LINE", "ARC".

**SUBRUTINA inicio ()**

FORMATO : int inicio (void),
FUNCIÓN : Inicializa el archivo .cn, en el cual se va a codifiar en código de control nuérico la lista ligada.
DATOS DE ENTRADA . Ninguno.
DATOS DE SALIDA : Bandera de error.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA :mensajes(), salva_ventana(), WindowG(), cursor_on(),
CapturaNombre (), AjustaFileName (), CierraVentana().
PROCESO · Se verifica si hay datos en la lisa ligada y si la trayectoria y el origen ya fueron seleccionados. Si no es así, se manda un mensaje de error, si sí, se lee el nombre del archivo de trabajo y se abre.

**SUBRUTINA datosCN ()**

FORMATO : void datosCN (int *, float *, float *, float *),
FUNCIÓN : Extraer de la lista ligada los datos necesarios para el fresado del siguiente bloque de entidades
DATOS DE ENTRADA . Número de renglón del listado .cn.
DATOS DE SALIDA : Nuevo número de renglón del listado .cn. El avance del cortador, el número de pasadas y la profundidad por pasada para el siguiente bloque de entidades.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA · Ninguna.
PROCESO · Se extrae de la lista ligada el número de pasadas. Se calcula la profundidad por pasada. Se calcula el avance y se escriben en el archivo las líneas de código correspondientes.

**SUBRUTINA lineaCN ()**

FORMATO int lineaCN ( int, float, float, struct nododxf);
FUNCIÓN : Escribe en disco una línea, en código cn, para fresar una línea.
DATOS DE ENTRADA : Número de línea del listado de .cn. El avance del cortador. La profundidad del fresado y el nodo que almacena las coordenadas.
DATOS DE SALIDA : El nuevo número de línea del listado .cn.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Incrementa el contador de líneas. Se baja la herramienta. Se escribe en el archivo la línea de programa que ejecuta el comando. Se sube la herramienta

**SUBRUTINA arcoCN ()**

FORMATO : int arcoCN (int, float, float, struct nododxf *);
FUNCIÓN Escribe en disco una línea, en código cn, para fresar un arco.
DATOS DE ENTRADA : Número de línea del listado de .cn. El avance del cortador. La profundidad del fresado y el nodo que almacena las coordenadas.
DATOS DE SALIDA : El nuevo número de línea del listado .cn.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO . Incrementa el contador de líneas. Se baja la herramienta Se escribe en el archivo la línea de programa que ejecuta el comando. Se sube la herramienta.

**SUBRUTINA circuloCN ()**

FORMATO . int circuloCN ( int, float)
FUNCIÓN : Escribe en disco una línea, en código cn, para ejecutar un barreno.
DATOS DE ENTRADA : Número de línea del listado de .cn. El avance del cortador.
DATOS DE SALIDA . El nuevo número de línea del listado .cn.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Incrementa el contador de líneas. Imprime en el archivo una línea en código de control numérico que al ejecutarse forma un barreno.

**SUBRUTINA simular ()**

FORMATO : void simular (void),
FUNCIÓN : Dirige la simulación del fresado de un archivo .cn
DATOS DE ENTRADA : Ninguno
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : WindowG (), mensajes (), tecla_pulsada(), pausa (), N(), G(), M(), T()
PROCESO : Se declaran variables. Se inicializan las coordenadas (x,y) del husillo visto desde arriba. Se crea un ambiente gráfico en pantalla para simular el fresado. El área de trabajo en pantalla se

reduce a una escala de 3 a 1 (3 pixels del monitor por cada mm de la fresadora) , considerando que los datos reales en la fresadora son largo=185 mm y ancho=100 mm. Se lee el nombre del archivo a fresar y se abre el archivo. Se abre un ciclo para leer el archivo Dentro del ciclo se lee e interpreta cada renglon, y deacuerdo al código N, G, M o T se ejecuta una diferente tarea mediante el llamado de una subrutina.

**SUBRUTINA G ()**

FORMATO : char G (char *p)
FUNCIÓN : Interpreta el código "G" del archivo .cn. El código G representa desplazamiento.
DATOS DE ENTRADA : La cadena de caracteres a interpretar.
DATOS DE SALIDA : Bandera de error.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : SeparaDato(), SubeBajaHus (), AvanceHusillo (), MueveHusillo ().
PROCESO : Después de declarar e inicializar variables, se separa el siguiente caracter de la cadena de entrada y se convierte a número entero. Se inicia un ciclo que termina cuando se llega al final de la cadena de entrada, dentro de éste se separa el siguiente dato y se convierte a número entero. Este número entero representa el número de comando G, y puede ser :

0 - Desplaza rápidamente el cortador en cualquer dirección lineal.
1 - Desplaza lentamente el cortador en cualquier dirección lineal.
2 - Desplaza el cortador en forma circular en sentido horario.
3 - Desplaza el cortador en forma circular en sentido antihorario.
40 - Sin compensación.
41 - Compensación izquierda.
42 - Compensación derecha
53 -
54 -
82 - Barreno.

Fuera ya del ciclo, si el código "N" fué 0 o 1 se sube o baja el husillo, según corresponda y se desplaza el husillo en estado encendido o apagado. Si "N" fué 82, se realiza un barreno con el llamado de las subrutinas : SubeBajaHus (), AvanceHusillo () y MueveHusillo ().

**SUBRUTINA M ()**

FORMATO : char M (char *p);
FUNCIÓN : Interpreta el código M del archivo de control numérico.
DATOS DE ENTRADA : Una cadena de caracteres del archivo .cn.
DATOS DE SALIDA : Bandera de error.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : SeparaDato (), OnOffHusillo (), Refrig ().
PROCESO : Se separa el siguiente código de la cadena de caracteres. Si el código es 3 gira husillo en sentido de las manecillas del reloj Si es 4 gira husillo en sentido contrario a las manecillas del reloj Si

es 5 apaga husillo Si es 8 enciende el refrigerante. Si es 9 apaga el refrigerante. El código 30 indica fin del programa de cn

**SUBRUTINA N ()**

FORMATO : void N (char *p, char *error)
FUNCIÓN : Interpreta el código N del archivo de control numérico.
DATOS DE ENTRADA : La cadena de caracteres a interpretar.
DATOS DE SALIDA : Bandera de error.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : SeparaDato ().
PROCESO : Declaración de variables Separa un dato de la cadena de caracteres de entrada Este dato debe ser un número entero y debe ser mayor al número leído en el comando "N" anterior. Esta subrutina verifica que el número de comando del listado de código de control numérico, sea ascendente

**SUBRUTINA CambioHerr ()**

FORMATO : void CambioHerr (char FlagInOutHerr)
FUNCIÓN : Esta subrutina representa el cambio de herramienta en pantalla.
DATOS DE ENTRADA Una bandera que indica si la herramienta sale o entra.
DATOS DE SALIDA :
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : TrazaH ().
PROCESO : Se declaran e inicializan variables. Según la bandera de entrada-salida de la herramienta inicializamos las variables de posición inicial, final e incremento-decremento. Dentro de un ciclo, se traza y se borra pausadamente la herramienta, dando la impresión de movimiento.

**SUBRUTINA TrazaH ()**

FORMATO : void TrazaH (int, int, int);
FUNCIÓN : Traza la herramienta en pantalla vista lateralmente.
DATOS DE ENTRADA : Las coordenadas de la herramienta. El color.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Se traza linea por línea un cortador.

**SUBRUTINA OnOffHusillo ()**

FORMATO : char OnOffHusillo (int, char)
FUNCIÓN : Enciende o apaga el husillo.
DATOS DE ENTRADA : El estado del husillo y la cadena de caracteres del archivo .cn.
DATOS DE SALIDA : El estado del husillo. Bandera de error.

SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : MueveHusillo (), SeparaDato ().
PROCESO . El estado del husillo 3 o 4 indica que el husillo está apagado. Si es así, se enciende. Si el husillo está encendido se apaga.

**SUBRUTINA AvanceHusillo ()**

FORMATO : void AvanceHusillo (int, int, int, int, int)
FUNCIÓN : Desplaza el husillo visto desde arriba desde la posición (x,y) hasta (x2,y2).
DATOS DE ENTRADA : Las coordenadas de inicio (x,y). Las coordenadas del punto final (x2, y2). Una bandera que indica si se trata de la función G0 o G1.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : CaminaHusillo ().
PROCESO . Si la función es G0, el incremento de avance del husillo es mayor que si la función es G1, así que, a la variable "incremento" se le asigna un valor de acuerdo a la función de que se trate. Se calcula el valor de la pendiente de avance entre el eje Y y X.

Si la pendiente m es m<=1 y m>=0, se inicia un ciclo que recorre el husillo desde la posición x inicial hasta la posición x final, con un cierto incremento de avance. Si la pendiente m es m>1, dentro de un ciclo se avanza el husillo desde la posición y inicial hasta la final.

**SUBRUTINA CaminaHusillo ()**

FORMATO : void CaminaHusillo (int, float, char)
FUNCIÓN : Desplaza el husillo, visto desde arriba, sobre el área de trabajo representado gráficamente en pantalla.
DATOS DE ENTRADA : Bandera del eje x o y. Valores para calcular la posición en X y Y.
DATOS DE SALIDA : ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : salva_ventana (), TrazaHusPos (), MueveHusillo (), CierraVentana().
PROCESO : Según la bandera "op" (para el eje x o y), se inicializan las variables x2 y y2. Si el husillo está apagado, se representa en pantalla sin movimiento. Si está encendido, se representa con movimiento. Si la coordenada del husillo en el eje Z es menor que 0, quierer decir que el cortador está penetrando en el material y se representa en color negro.

**SUBRUTINA divide ()**

FORMATO : float divide (int, int)
FUNCIÓN . Divide dos números.
DATOS DE ENTRADA : el numerador y el denominador.
DATOS DE SALIDA : el cociente.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.

PROCESO . Se divide el número**rador** entre el denominador. Se busca primero la parte entera del cociente, se añade el punto y se busca la parte decimal. El cociente se almacena inicialmente en una cadena de caracteres y posteriormente se convierte a flotante.

**SUBRUTINA TrazaHusPos ()**

FORMATO void TrazaHusPos (int, int, int, int);
FUNCIÓN : Traza el husillo en cualquiera tres diferentes posiciones. Lo traza visto desde arriba.
DATOS DE ENTRADA : Las coordenadas del centro del husillo. La posicion de movimiento del husillo (1=0 grados, 2=30 grados ó 3=60 grados).
DATOS DE SALIDA : ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA Ninguna.
PROCESO : Traza un circulo que representa al husillo Traza una cruz dentro del circulo deacuerdo a la posición indicada por el dato de entrada.

**SUBRUTINA MueveHusillo ()**

FORMATO : void MueveHusillo (int, int, int);
FUNCIÓN : traza y borra el husillo consecutivamente, dando la impresión de movimiento giratorio.
DATOS DE ENTRADA : las coordenadas del centro del husillo. El número de giros que da el husillo.
DATOS DE SALIDA : ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : TrazaHusPos ().
PROCESO : Se ejecuta un ciclo tantas veces como giros da el husillo. Dentro de éste ciclo se traza y se borra el husillo en sus tres posiciones diferentes a través del llamado de la funciôm TrazaHusPos ().

**SUBRUTINA SeparaDato ()**

FORMATO . void SeparaDato (char, char)
FUNCIÓN . separa un dato (número entero o flotante o cadena de caracteres).
DATOS DE ENTRADA : ninguno.
DATOS DE SALIDA : el tipo de dato y el dato.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Después de declarar e inicializar variables se inicia un ciclo de lectura del archivo que finaliza con ' ', '\n' o '\0'

**SUBRUTINA Refrig ()**

FORMATO void Refrig ( int);

FUNCIÓN : Despliega un letrero que indica si el refrigerante esta encendido o apagado.
DATOS DE ENTRADA El estado del refrigerante : encendido o apagado.
DATOS DE SALIDA : ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Un ciclo imprime varias veces en pantalla un mensaje de refrigerante encendido o apagado, según corresponda. Da la impresión de estar parpadeando.

**SUBRUTINA T ()**

FORMATO : char T (char *p)
FUNCIÓN . Interpreta el código T del listado de control numérico que se refiere al cambio de herramienta.
DATOS DE ENTRADA : La cadena de caracteres a interpretar.
DATOS DE SALIDA : ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA . SeparaDato (), CambioHerr ().
PROCESO : Declara variables. Lee un dato en el archivo .cn. Este dato corresponde al número de herramienta Si la herramienta anterior es diferente a la nueva, se interpreta que se hizo un cambio de herramienta y se representa gráficamente en pantalla la salida de la primera y la entrada de la segunda.

**SUBRUTINA SubeBajaHus ()**

FORMATO : void SubeBajaHus (char, int, int, int)
FUNCIÓN : Dirige el movimiento hacia arriba y hacia abajo del husillo, con movimiento giratorio.
DATOS DE ENTRADA : Bandera que inidica si sube o baja la herramienta. Nueva posición hacia arriba o hacia abajo del husillo. Coordenadas del centro del husillo visto desde arriba.
DATOS DE SALIDA : ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : SubeBajaH (), MueveHusillo ().
PROCESO Si el husillo está encendido y se mueve hacia arriba, se ejecuta el ciclo que sube pausadamente el husillo a su vez que hace girar el cortador visto desde arriba. Si se mueve hacia abajo, se baja pausadamente y se hace girar el cortador. Si el husillo está apagado, simplemente se sube o baja el husillo visto lateralmente, sin hacer girar el cortador visto desde arriba.

**SUBRUTINA SubeBajaH ()**

FORMATO : void SubeBajaH (char, int)
FUNCIÓN . Dibuja la herramienta en pantalla en desplazamiento hacia arriba o hacia abajo.
DATOS DE ENTRADA : Bandera que indica si la herramienta "sube" o "baja. El desplazamiento.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : TrazaH () .
PROCESO Borra la herramienta trazada en pantalla. Inicializa el "fin" y el "inicio" del recorrido. Si la herramienta sube, inicializa incremento como -1. Si la herramienta baja la inicializa como 1 Ejecuta un

ciclo dentro del cual traza y borra el husillo pausadamente, dando el efecto de movimiento hacia arriba o hacia abajo

**SUBRUTINA imprimir ()**

FORMATO : void imprimir (void);
FUNCIÓN : Manda a impresora un archivo.
DATOS DE ENTRADA : ninguno.
DATOS DE SALIDA : ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : salva_ventana (), WindowG(), CapturaNombre (), AjustaFileName (), CierraVentana(), cursor_on(), GetFileName (), mensajes (), pausa ()
PROCESO : Después de declarar variables, se captura el nombre del archivo a imprimir. Si el archivo existe se inicia un ciclo de recorrido del archivo el cual se manda caracter por caracter a impresora.

**SUBRUTINA GetFileName ()**

FORMATO : int GetFileName (char *, int, int, int, int)
FUNCIÓN : Controla la captura del nombre del archivo.
DATOS DE ENTRADA : Las coordenadas de la subventana del directorio.
DATOS DE SALIDA : La Ruta seleccionada por el usuario y una bandera de ESC.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : FreeDIR (), salva_ventana(), WindowG (), DefaultDir (), CargaDir (), DespliegaDir (), CierraVentana (), separa (), cursor_on(), DesplazaDir (), ayuda (), lee_cadena ()
PROCESO: Declaración e inicialización de variables. Se llama a la subrutina DefaulDir (), que da encuentra la ruta del directorio en que se está trabajando. Se carga el directorio de la ruta encontrada (CargaDir()) y se despliega (DespliegaDir()). Se lee de teclado el comando a ejecutar que puede ser :

TAB : Si el cursor está en la ventana de lectura de la ruta, se pasa a la ventana de despliegue de archivos y éstos se despliegan.
Si el cursor está en la ventana de despliegue de archivos, se sube a la ventana de lectura de la ruta.
RETURN : Si el cursor está en la ventana de lectura de la ruta, se despliegan los archivos de la ruta indicada y se pasa el cursor a esa ventana. Si el cursor está en la ventana de despliegue de archivos, se considera como el nombre archivo deseado aquel sobre el cuál se encuentra el cursor cuando se oprime RETURN.
ESC : Cierra la ventana de directorio.
FLECHAS : Desplazan el cursor hacia arriba o hacia abajo del directorio.
F10 : Despliega la ayuda.
default : Si el cursor se encuentra en la ventana de lectura de la ruta se llama al subrutina lee_cadena (), para leer la ruta.

**SUBRUTINA DefaultDir ()**

FORMATO : void DefaultDir ( int, char *, int, int, int)
FUNCIÓN : Forma la ruta y/o el archivo por default.
DATOS DE ENTRADA : La ruta y el nombre del archivo actuales.
DATOS DE SALIDA : La RutaTotal, la Ruta y el nombre del archivo actualizados.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : AjustaFileName(), WindowG ().
PROCESO: Declaración de variables. Inicializa variables. Encuentra consecutivamente los datos que forman el nombre de la ruta actual.

Concatena en el siguiente orden :

1. El "drive", que puede ser A, B, C o D.
2. El nombre del directorio activo.
3. El nombre de cualquier archivo "*".
4. La extensión " DXF"

**SUBRUTINA DesplazaDir ()**

FORMATO : void DesplazaDir ( int);
FUNCIÓN : Desplaza el cursor a lo largo del directorio desplegado con el uso de las flechas.
DATOS DE ENTRADA : Lista ligada del directorio y sus apuntadores de inicio y fin. Apuntadores de inicio y fin de la parte visible en la pantalla. Coordenadas de la ventana.
DATOS DE SALIDA : Tarea que quiere ejecutar el usuario. Opción del dir.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO: Desplaza el directorio, según la tecla pulsada por el usuario, hacia ARRIBA o hacia ABAJO.

**SUBRUTINA DespliegaDir ()**

FORMATO : void DespliegaDir (int, int, int, int, int);
FUNCIÓN : Despliega el directorio en la pantalla y remarca la opción actual.
DATOS DE ENTRADA : Lista ligada del directorio y sus apuntadores de inicio y fin. Apuntadores de inicio y fin de la parte visible dentro de la ventana.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : WindowG ().
PROCESO: Declara e inicializa variables. Se despliega la ventana del directorio. Dentro de un ciclo se recorre la parte visible de la ventana de la lista ligada que almacena el directorio y se imprime cada nombre en pantalla.

**SUBRUTINA AjustaFileName ()**

FORMATO : void AjustaFileName (char Ruta[40]);
FUNCIÓN : Ajusta la ruta y el nombre del archivo a 8 caracteres máximo.
DATOS DE ENTRADA : Nombre del archivo sin ajustar.
DATOS DE SALIDA : Nombre del archivo ajustado.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO: Declaración e inicialización de variables. Se inicia un ciclo que recorre la Ruta y almacena uno a uno los caracteres hasta fin de "Ruta" o hasta encontrar '.' o llegar al máximo que es de 8 caracteres Finalmente se inicia un ciclo que ajusta la extension a máximo 3 caracteres.

**SUBRUTINA separa ()**

FORMATO void separa (char*, char*, char[]);
FUNCIÓN : Separa la ruta y el nombre del archivo del nombre completo.
DATOS DE ENTRADA : La cadena completa con la Ruta y el nombre.
DATOS DE SALIDA : La ruta (Ruta) y el nombre del archivo.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA . Ninguna.
PROCESO: Declara e inicializa variables. Dentro de un ciclo se recorre el nombre completo del archivo con un apuntador. Dentro del ciclo se almacen caracter por caracter en la variable "Ruta1" la parte correspondiente a la ruta y en "FileName" la parte correspondiente al nombre del archivo.

**SUBRUTINA CargaDir ()**

FORMATO : void CargaDir (char *, char *)
FUNCIÓN : Carga los nombres de los archivos en una lista ligada.
DATOS DE ENTRADA : La ruta a seguir para cargar el directorio del disco.
DATOS DE SALIDA : Una lista doblemente ligada en donde cada nodo es un archivo del directorio.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : FreeDIR ().
PROCESO Se declaran e inicializan variables. Se abre un ciclo que recorre el directorio señalado por la ruta total. Dentro de este ciclo se crean nodos para almacenar en una lista ligada los nombres de los archivos.

**SUBRUTINA FreeDIR ()**

FORMATO : void FreeDIR (void);
FUNCIÓN : Libera la lista dinámica que almacena el directorio.
DATOS DE ENTRADA : Lista del directorio.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO: Dentro de un ciclo se recorre la lista ligada que almacena el directorio (headdir - taildir) y se libera nodo por nodo.

**SUBRUTINA tecla_pulsada ()**

FORMATO : tecla_pulsada (void)
FUNCIÓN . Captura la tecla pulsada en el teclado.
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA El código de la tecla pulsada.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA · Ninguna.
PROCESO: Captua la tecla pulsada.

**SUBRUTINA cursor_off ()**

FORMATO · cursor_off()
FUNCIÓN Desaparece el cursor de pantalla.
DATOS DE ENTRADA : ninguno.
DATOS DE SALIDA ·ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA . ninguna.
PROCESO· Borra el cursor de pantalla y la restablece a través de una interrupción.

**SUBRUTINA cursor_on ()**

FORMATO . cursor_on()
FUNCIÓN · Dibuja el cursor en pantalla.
DATOS DE ENTRADA · Ninguno
DATOS DE SALIDA . Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO· Dibuja en pantalla el cursor gráfico a través de una interrupción.

**SUBRUTINA derechab_pulsado ()**

FORMATO : derechab_puslado()
FUNCIÓN . Identifica si fué pulsado el botón derecho del ratón.
DATOS DE ENTRADA : ninguno.
DATOS DE SALIDA · la respuesta de si fué pulsado o no.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : NInguna.
PROCESO: Mediante una interrupción. la computadora identifica si fué pulsado el botón derecho del ratón

**SUBRUTINA izquierdab_pulsado ()**

FORMATO . izquierdab_pulsado()
FUNCIÓN . identifica si fué pulsado el botón izquierdo del ratón.
DATOS DE ENTRADA .ninguno.

DATOS DE SALIDA . ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA · Ninguna.
PROCESO· Mediante una interrupción. la computadora identifica si fué pulsado el botón izquierdo del ratón.

**SUBRUTINA pon_posicion_raton ()**

FORMATO : void pon_posicion_raton (int, int );
FUNCIÓN · Da una nueva posición al mouse en las coordenandas (x,y).
DATOS DE ENTRADA : Las coordenadas (x,y) del ratón.
DATOS DE SALIDA · NInguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : cmouses().
PROCESO. Llama a la subrutina cmouses() con los argumentos correspondientes para asignar una nueva posición al mouse.

**SUBRUTINA posicion_raton ()**

FORMATO : void posicion_raton (int *x, int *y);
FUNCIÓN : Retorna las coordenadas de la posicion del raton
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Las coordenadas (x,y) del mouse.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : cmouses().
PROCESO: Llama a la subrutina cmouses() con los argumentos correspondientes para leer la posición actual del mouse y actualiza las variables (x,y).

**SUBRUTINA cmouses ()**

FORMATO : void cmouses (int *fnum, int *arg2, int *arg3, int *arg4)
FUNCIÓN : Ejecuta las interrupciones del raton según la función de entrada.
DATOS DE ENTRADA · El número de interrupción a ejecutar
DATOS DE SALIDA : Valores de los argumentos según la interrupción generada.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguno.
PROCESO: Declaración de variables Si la interrupción es 0, se reciben los valores: estado del mouse en el registro ax y número de botones en el registro bx. Si la interrupción es tres, se reciben los valores estaso del botón en el registro bx, posicion horizontal del cursor en cx y posición vertical del cursor en dx. Si la interrupción es cuatro, se recibe la nueva posicion horizontal del cursor en dx. Si la interrupción es 11, se recibe un contador horizontal en cx y un contador vertical en dx.

**SUBRUTINA flechas ()**

FORMATO · int flechas (void)
FUNCIÓN · Controla el cursor por teclado.
DATOS DE ENTRADA Ninguno.
DATOS DE SALIDA · Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA · Ninguna.
PROCESO: captura tecla de teclado. Dependiendo de la tecla capturada se identifica si el usuario oprimió la flecha que se desplaza hacia arriba, hacia abajo, hacia la derecha, hacia la izquierda, hacia arriba a la izquirda, arriba a la derecha, abajo a la izquierda, abajo a la derecha, si se llamó al a ayuda (F10) o si se quiere cambiar la velocidad de desplazamiento sobre el área de trabajo de entidades. También se hacen los ajustes necesarios para las tareas anteriores.

**SUBRUTINA obt_tecla ()**

FORMATO · obt_tecla (void);
FUNCIÓN : Una captura de teclado.
DATOS DE ENTRADA : Buffer de teclado.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISITEMA GC QUE LLAMA : Ninguna.
PROCESO: Ejecuta la interrupción 86 para lectura de teclado.

**SUBRUTINA vete_xy ()**

FORMATO void vete_xy (int x,int y)
FUNCIÓN · Posiciona el cursor en la posición (x,y) de la pantalla.
DATOS DE ENTRADA : Datos de entrada : coordenadas (x,y) del cursor.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO· Inicializa los registros del procesador con los siguientes valores: ah = funcion de direccionamiento del cursor, dl = columna, dh = fila, bh = página de video Ejecuta la interrupción 86.

**SUBRUTINA lee_cadena ()**

FORMATO : void lee_cadena (char *, int, int, int, int, int)
FUNCIÓN . Lee una cadena de caracteres en una variable.
DATOS DE ENTRADA : La ruta anterior. Las coordenadas de la ventana de captura.
DATOS DE SALIDA . La nueva ruta.

SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : WindowG ().
PROCESO : Declaración e inicialización de variables. Con un ciclo se recorre la cadena de caracteres y se coloca el apuntador al final de Ruta. Almacena el caracter leído al final de la Ruta.

**SUBRUTINA espera_on ()**

FORMATO : void espera_on (int)
FUNCIÓN : Espera mientras esta oprimido un boton del mouse.
DATOS DE ENTRADA : El botón pulsado.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : izquierdab_pulsado(), derechab_pulsado().
PROCESO. Realiza un ciclo que termina hasta que el botón izquierdo o derecho es soltado.

**SUBRUTINA mensajes ()**

FORMATO : void mensajes ( int)
FUNCIÓN : Despliega en pantalla un mensaje.
DATOS DE ENTRADA El número de mensaje
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : cursor_off(), salva_ventana(), WindowG (), CierraVentana(), cursor_on().
PROCESO: Prepara la ventana de mensajes. Despliega el mensaje correspondiente de acuerdo al número de entrada. Restaura la ventana cuando el usuario oprime una tecla.

**SUBRUTINA pausa ()**

FORMATO : void pausa (int, int);
FUNCIÓN : Despliega en pantalla un mensaje.
DATOS DE ENTRADA : El número de la pausa.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : salva_ventana (), WindowG (), CierraVentana ().
PROCESO: Prepara la ventana de mensajes. Despliega el mensaje correspondiente de acuerdo al número de entrada. Aparece el mensaje antes de iniciar una tarea propia del sistema GC y la desaparece cuando termina de ejecutarla.

**SUBRUTINA SeleccionarEntidad ()**

FORMATO : struct nododxf *SeleccionarEntidad();
FUNCIÓN Dirige la selección de entidades desplegadas gráficamente en pantalla.
DATOS DE ENTRADA : Ninguno.

DATOS DE SALIDA La lista ligada de entidades seleccionadas.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : WindowG (), pon_posicion_raton(), cursor_on(), movimiento_raton(), posicion_raton(), izquierdab_pulsado(), espera_en(), derechab_pulsado(), tecla_pulsada (),
flechas (), pon_posicion_raton (), ayuda(), identificar_entidad_en_lista().
PROCESO: Despliega una ventata que indica que se seleccione la entidad. Se inicia un ciclo que detecta si el usuario está usando el mouse o el teclado para desplazarse en pantalla. De acuerdo a un punto de la pantalla seleccionado por el usuario se identifica si éste pertenece o no a alguna de las entidades de la lista ligada.

**SUBRUTINA identificar_entidad_en_lista ()**

FORMATO : struct nododxf *identificar_entidad_en_lista (int , int );
FUNCIÓN : Identifica si el punto (x,y) pertenece a alguna entidade de la lista ligada.
DATOS DE ENTRADA : Las coordenadas del punto (x,y).
DATOS DE SALIDA : Un apuntador al nodo al cual pertenece el punto (x,y).
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA :
PROCESO: Declaración de variables. Recorre la lista ligada de entidades. Para cada entidad (línea, arco o círculo), se identifica si el punto (x,y) pertenece o no

**SUBRUTINA is_closeLine()**

FORMATO : int is_closeLine (int , int , int , int , int , int );
FUNCIÓN : Revisa si el punto (x,Y) pertenece o no a esa línea.
DATOS DE ENTRADA . Las coordenadas de los extremos de la línea y las coordenadas del punto en estudio.
DATOS DE SALIDA : Uno si el punto pertenece a la línea y cero si no pertenece.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : NInguna.

**SUBRUTINA is_closeArc ()**

FORMATO : int is_Arc (int ,int , float ,f loat, int ,int int);
FUNCIÓN : Revisa si el punto (x,y) pertenece o no al arco.
DATOS DE ENTRADA : Las coordenadas del centro del arco. Los ángulos inicial y final del arco y el radio Las coordenadas del punto en estudio (x,y)
DATOS DE SALIDA : Uno si el punto pertenece al arco y cero si no pertenece.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.

**SUBRUTINA is_closeCirc ()**

FORMATO . int is_Circ (int , int , int , int , int )

FUNCIÓN : Revisa si el punto en estudio (x,y) pertenece o no al círculo.
DATOS DE ENTRADA . Las coordenadas del centro del círculo, su radio. Las coordenadas del punto en estudio.
DATOS DE SALIDA Uno si el punto pertenece a la circunferencia. Cero si no pertenence.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.

**SUBRUTINA MensajesDosOpciones ()**

FORMATO : int MensajesDosOpciones (int )
FUNCIÓN : Despliega una ventana que da al usuario dos opciones para que escoja alguna.
DATOS DE ENTRADA : El número del mensaje.
DATOS DE SALIDA : La opción seleccionada por el usuario.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : salva_ventana() WindowG(),
PROCESO:Según la opción correspondiente a la opción de la variable de entrada, se despliega en pantalla un mensaje con dos diferentes opciones. Se lee la opción. Se cierra la ventana de lectura.

**SUBRUTINA LeeNumeroBloque ()**

FORMATO int LeeNumeroBloque ()
FUNCIÓN : Lee el número del bloque que agrupa entidades con caracteísticas similares de fresado.
DATOS DE ENTRADA . Ninguno.
DATOS DE SALIDA : El número del bloque.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : WindosG(), salva_ventana(), LeeNum(), CierraVentana(), cursor_on(), cursor_off().
PROCESO:Despliega una ventana con los números de los bloques que ya están activos. Llama a la subrutina que lee el nuevo número de bloque.

**SUBRUTINA LeeNum ()**

FORMATO : int LeeNum (int x, int y)
FUNCIÓN : Lee un número entero.
DATOS DE ENTRADA : Las coordenadas en pantalla para la lectura del número.
DATOS DE SALIDA : El número leído
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO: Lee caracter por caracter el número. Captura teclas de control también, como el retroceso, ayuda, etc.

**SUBRUTINA identificar_entidad_en_lista ()**

FORMATO : struct nododxf *identificar_entidad_en_lista (int, int)
FUNCIÓN Encuentra la entidad a la cual corresponde el punto (x,y).

DATOS DE ENTRADA : Las coordenadas del punto (x,y).
DATOS DE SALIDA : Un nodo que almacena los datos de la entidad a la cual corresponde el punto con coordenadas: (x,y)
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Después de declarar variables, se inicia un ciclo para recorrer la lista ligada. Dentro del ciclo se identifica si el punto con coordenadas (x,y) forma parte de las entidades LINE, CIRCLE o ARC.

**SUBRUTINA CapturaNombre ()**

FORMATO : void CapturaNombre (char*,int, int, int, int,int)
FUNCIÓN : Lee la ruta y el nombre del archivo.
DATOS DE ENTRADA : ninguno.
DATOS DE SALIDA : La ruta el nombre del archivo.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : salva_ventana (), WindowG(), CierraVentana().
PROCESO : Después de declarar variables, dentro de un ciclo se lee cada caracter que el usuario necesita para formar el nombre del archivo, hasta salir (RETURN o ESC). También se puede llamar a la ayuda durante la lectura.

**SUBRUTINA Real_Cadena**

FORMATO : void Real_Cadena (float, char *)
FUNCIÓN : Transforma un número real a cadena de caracteres.
DATOS DE ENTRADA : El número real.
DATOS DE SALIDA : Una cadena de caracteres.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.

**SUBRUTINA LeeNumeroReal ()**

FORMATO : void LeeNumeroReal ( char*, int, int, int, int, int);
FUNCIÓN : Lee un número real como cadena de caracteres.
DATOS DE ENTRADA : Las coordenadas de la ventana de lectura del número.
DATOS DE SALIDA : El número real como cadena de caracteres.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : WindowG ()
PROCESO : Se declaran variables. Se recorre el número real para identificar si existe en él el punto decimal. Se despliega el número en pantalla. Se inicia un ciclo de captura de datos hasta encontrar retorno de carro o ESC. Si el usuario presiona la flecha de retroceso, se borra el último caracter de la lista. Con F10, se llama a la ayuda. Si presiona algún número se almacena en la lista de caracteres y con ESC, se puede salir de la lectura.

**SUBRUTINA Initialize ()**

FORMATO : void Initialize (void);
FUNCIÓN : Inicializa el modo gráfico.
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Dimensiones de la pantalla.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO: Detacta el tipo de monitor en que se está trabajando. Inicializa la pantalla en modo gráfico. Encuentra las dimensiones de la pantalla.

**SUBRUTINA pantalla_principal ()**

FORMATO : pantalla_principal ()
FUNCIÓN : Despliega la pantalla principal.
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO: Limpia la pantalla, despliega fondo, establece el color de la sombra, despliega sombra, despliega menu principla, establece tipo de letra, remarca primera opcion del menu principal.

**SUBRUTINA salva_ventana ()**

FORMATO : void salva_ventana( int, int, int, int)
FUNCIÓN : Salva en memoria principal una parte de la pantalla.
DATOS DE ENTRADA : Coordenadas de la ventana a salvar.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO: Salva en memoria secundaria una parte de la pantalla.

**SUBRUTINA WindowG ()**

FORMATO void WindowG (int, int, int, int, int, int, int);
FUNCIÓN : Despliega una ventana de acuerdo a los parámetros especificados.
DATOS DE ENTRADA : esquinas superior izquierda e inferior derechaj de la ventana. Colores de fondo y marco. Tamaño de la sombra.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO. Despliega una ventana en pantalla, establece su sombra, tipo de linea, color del fondo . Despliega fondo y marco.

**SUBRUTINA CoordenadasMouse ()**

FORMATO . void CoordenadasMouse (void);
FUNCIÓN : Calcula y almacena las coordenadas en pantalla del área de trabajo del mouse.
DATOS DE ENTRADA : Ninguno.
DATOS DE SALIDA : NInguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO: Declaración de variables. Realiza un ciclo que almacena el área de trabajo del menú principal. Realiza ciclos, cada uno de los cuales almacena el área de trabajo de cada submenú.

**SUBRUTINA PintaTrazos ()**

FORMATO : void PintaTrazos (struct nododxf *)
FUNCIÓN : Traza en pantalla una entidad de dibujo como línea, arco y círculo.
DATOS DE ENTRADA : El apuntador al nodo que almacena la entidad.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguno.
PROCESO: Identifica de que tipo de entidad se trata (línea, arco o círculo) y la traza en pantalla.

**SUBRUTINA CierraVentana ()**

FORMATO : void CierraVentana (void);
FUNCIÓN : Restaura el área de la pantalla donde se desplegó la ventana BuffWin.
DATOS DE ENTRADA : La variable que almacena la ventana (BuffWin).
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : cursor_off (), cursor_on ().
PROCESO . Cierra la ventana abierta y restaura la pantalla

**SUBRUTINA MuestraCoordenadas ()**

FORMATO : void MuestraCoordenadas ( int, int);
FUNCIÓN : Despliega las coordenadas del cursor en pantalla.
DATOS DE ENTRADA : Las coordenadas actuales del cursor.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO· Prepara la pantalla. Despliega los valores (x,y) actuales del cursor.

**SUBRUTINA GraficaDXF ();**

FORMATO · void GraficaDXF (struct nododxf *, int, int),

FUNCIÓN : Despliega gráficamente en pantalla las entidades.
DATOS DE ENTRADA : La lista de entidades a graficar.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : GraficaOrigen (), TrazaSentido ().
PROCESO : Se declaran e inicializan variables. Se prepara la pantalla para desplegar la gráfica. Dentro de un ciclo se recorre la lista de entidades y se traza cada una de éstas ( LINE, CIRCLE, ARC). Si se requiere, se traza el sentido de ellas mediante el llamado de la subrutina TrazaSentido ().

**SUBRUTINA TrazaSentido ()**

FORMATO : void TrazaSentido (struct nododxf *)
FUNCIÓN . Traza, en el centro de la entidad almacenada en el nodo de entrada, una flecha que indica el sentido del fresado.
DATOS DE ENTRADA : El nodo que almacena la entidad a trabajar.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO : Se declaran e inicilizan variables Se inicia un ciclo dentro del cual se recorre la lista de entidades a fresar. De acuerdo a la entidad de que se trate (LINE o ARC) se accesa a una serie de comandos que calcula los valores para trazar las flechas en ellas Se trazan la flechas.

**SUBRUTINA ValorKK1 ()**

FORMATO : void ValorKK1 ( float, float, float, float, int, int);
FUNCIÓN : Encuentra el valor de K, que es una variable para trazar el sentido de los arcos, de acuerdo al cuadrante actual.
DATOS DE ENTRADA : Los valores de las esquinas superior izquierda e inferior derecha.
DATOS DE SALIDA : k y k1.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO: De acuerdo al cuadrante actual, se calculan los valores para "k" y "k1".

**SUBRUTINA GraficaOrigen ()**

FORMATO : void GraficaOrigen ( int, int, int);
FUNCIÓN : Traza icono del origen en pantalla
DATOS DE ENTRADA : Coordenadas del origen y el color.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO. Traza en pantalla el origen (línea por línea).

**SUBRUTINA RecorreMenuP ()**

FORMATO void RecorreMenuP (int OpM);
FUNCIÓN : Recorre en pantalla una posición el menú principal.
DATOS DE ENTRADA : La nueva opción del menú en variable auxiliar.
DATOS DE SALIDA : La nueva opción del menú en variable principal.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA . Ninguna.
PROCESO: Desmarca la opción anterior del menú y remarca la nueva.

**SUBRUTINA CoordenadasMenu ()**

FORMATO : void CoordenadasMenu (int, int, int, int, int, int, int); FUNCIÓN : Calcula las coordenadas de las ventanas de los submenús.
DATOS DE ENTRADA : El número de menu (Nmenu), y la posicion del menu en vector el vector (pos) y el tamaño de la sombra.
DATOS DE SALIDA · Las coordenadas de la esquina superior izquierda e inferior derecha de la ventana del menú.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO. Calcula las coordenadas de la esquina superior izquierda e inferior derecha de la ventana del menú

**SUBRUTINA RecorreMenus ()**

FORMATO · void RecorreMenus ( int, char [][], char)
FUNCIÓN : Remarca una nueva opción del submenú.
DATOS DE ENTRADA · Número de opciones del menú y el menú en sí.
DATOS DE SALIDA : Opción del menu o ESC.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA : Ninguna.
PROCESO: Se despliegan en pantalla los letreros. De acuerdo a la flecha teclada se incrementan o decrementan las variables que almacenan el número de submenú actual y se imprime éste en otro color en pantalla.

**SUBRUTINA AbreMenu ()**

FORMATO . void AbreMenu (void);
FUNCIÓN Despliega los submenus en la pantalla.
DATOS DE ENTRADA : El número de menú actual.
DATOS DE SALIDA : Ninguno.
SUBRUTINAS PROPIAS DEL SISTEMA GC QUE LLAMA CoordenadasMenu(), salva_ventana(), WindowG(), cursor_on(), cursor_off ().
PROCESO. Se inicializa la variable Nmenu de acuerdo al número de menú actual. Se calculan las coordenadas correspondientes a esa ventana, se despliega junto con sus opciones.

# CONCLUSIONES

Dado el objetivo fundamental de este trabajo que fué el de desarrollar un sistema que automatice la programación de la máquina fresadora de marca EMCO con lenguaje de programación EMCOTRONIC que se encuentra en el laboratorio de manufactura avanzada del Instituto Tecnológico de la Ciudad de Puebla podemos concluir lo siguiente :

- Los procesos automatizados reducen tiempo y son más precisos

- Los procesos de simulación permiten la optimización de actividades y la detección de errores.

- Un código libre de errores implica ahorro de tiempo y recursos.

- Algunos sistemas CAD/CAM son herramientas que facilitan la programación de los equipos con control numérico computarizado.

- Los sistemas computacionales con interfases amigables para el usuario son de fácil acceso

-Un sistema generador de códigos requiere de que el usuario cuente con los conocimientos básicos generales necesarios para poder hacer uso de él.

- Los códigos de control numérico no pueden ser generalizados en uno solo puesto que el código de cada máquina - herramienta puede diferir del de las demás en menor o mayor grado.

- La tercera dimensión en una pieza diseñada en dos dimensiones se la da la profundidad de maquinado.

- El objetivo de este trabajo se cumplió al haberse concluído el desarrollo del Sistema GC, el cual genera automáticamente códigos que pueden ser interpretados por la fresadora EMCO, partiendo de archivos de dibujo en formato DXF, auque no pudo ser implementado en el Instituto Tecnológico Regional de Puebla.

- La eficiencia del Sistema GC puede ser evaluada hasta que se ponga en práctica el mismo, lo cual no se ha hecho debido a cuestiones de tipo administrativo.

El desarrollo de la ciencia y del a técnica es en uno de los principales motores que impulsan el desarrollo económico de los países.

Si desde el seno familiar un pueblo trabaja unido compartiendo valores como honestidad, unidad, sensatez, apoyo, humanidad, constancia, entereza, lealtad y deseos de superación entre otros, al ser puestos en práctica se enriquecen mutuamente sus habitantes y difícilmente un factor externo podría romper con la armonía alcanzada por todos, no siendo así cuando un factor interno es el que falla, y principalmente si el resto del pueblo le ha tenido confianza.

Los cambios acelerados que se han presentado en el desarrollo humano desde hace dos siglos han hecho que las economías fuertes sean intrínsecas a su desarrollo tecnológico. De tal manera que los países que gozan de bases técnicas y científicas sólidas gozan de una situación económicamente más fuerte que aquellas que no han encontrado la forma de crecer en estas áreas.

Por todo esto, se puede considerar como una responsabilidad de la sociedad mexicana y principalmente de las personas que tenemos o hemos tenido la invaluable oportunidad de recibir una educación superior, de buscar nuevos senderos que aunque recorran zonas áridas y difíciles de atravesar, tengan como menta el enriquecer la ciencia y la técnica mexicanas para lograr un país más sólido que dependa en menor grado de tecnologías extranjeras de alto precio.

**María Milagros Ruiz Limón.**

# GLOSARIO

**ALTURA DE SEGURIDAD** Es la distancia que debe subir el cortador sobre el material a trabajar para no causar daños cuando se requiera un movimiento sin arranque de viruta.

**AutoCAD** Paquete comercial de CAD.

**ÁRBOL PORTAFRESAS.** Es el eje de una fresadora que sostiene a los cortadores. También recibe el nombre de husillo.

**ARISTA CORTANTE.** Es cada una de las salientes del cortador que se encarga de realizar el corte.

**ASCII** (American Standard Code for Information Interchange) Código Estándar Americano de intercambio de información. Este código estandarizado es muy usual en la transmisión de Datos.

**CAD** Diseño Asistido por computadora.

**CAM** Manufactura asistida por computadora

**CN** (Control numérico). Es el control de procesos por medio de números, letras y símbolos que se agrupan en un programa.

**CNC** (Control numérico computarizado) Es el control numérico en el cual se almacenan datos o programas elaborados por computadoras y utilizado para desempeñar funciones básicas con el uso de una computadora.

**COMPENSACIÓN** Es la distancia a que se coloca el cortador con respecto a la línea de fresado.

**CORTADOR** Herramienta de corte utilizada por maquinas fresadoras. También recibe el nombre de fresa.

**DXB** Binary Exchange File. Archivo almacenado en binario que contiene toda la información requerida para reconstruir un dibujo en un formato específico.

**DXF** (Drawing Exchange File) Es un archivo ASCII que contiene toda la información requerida para reconstruir un dibujo en un formato específico.

**EMCOTRONIC** Uno de los controladores para las máquinas-herramienta EMCO.

**FRESA** Herramienta de corte utilizada por máquinas fresadoras. También recibe el nombre de cortador

**FRESADO** Proceso que consiste en labrar piezas de diferentes materiales (acero, aluminio, madera, etc.) por medio de uno o más cortadores giratorios (fresas) que cuentan con múltiples aristas cortantes.

**FRESADORA** Máquina-herramienta de la industria metal mecánica en la cual se efectúa el proceso de fresado.

**G** Se refiere al código G. Es una función preparatoria programada para una máquina de control numérico para simplificar el proceso de programación. Se utiliza con la letra "G" precedida por dos dígitos que pueden indicar interpolaciones, programación absoluta o incremental, métrico o inglés, etc.

**HARDWARE** Este termino se refiere a los aspectos físicos de una máquina de control o una computadora.

**HERRAMIENTA** En una máquina-herramienta se refiere los elementos que realizan el corte del material, como brocas y cortadores o fresas.

**HUSILLO** Es el eje que sostiene a los cortadores. También recibe el nombre de árbol portafresas.

**IGES** Initial Graphics Exchange Standard. Formato para almacenar archivos gráficos de intercambio.

**INTERFASE** : Es el medio por el cual pueden estar unidos dos elementos separados. Hay una interfase entre una máquina herramienta de control numérico programable y una computadora la cual nos permite el enlace de diversos programas.

**M** Código de control numérico que se refiere a las funciones misceláneas. Son los códigos utilizados en máquinas de control numérico que tienen la capacidad de accionar los dispositivos de dichas máquinas. Entre otras tareas estos códigos pueden controlar la conexión o desconexión del husillo o el refrigerante, etc.

**MANUFACTURA** Es un conjunto de operaciones y actividades correlacionadas y encaminadas a la producción, las cuales incluyen el diseño del producto, la selección del material, la planeación, la producción, la inspección, la dirección y la mercadotecnia para la industria manufacturera.

**MAQUINA-HERRAMIENA.** Elemento mecánico que facilita la ejecución de tareas específicas en el área industrial, educativo o científico. Por ejemplo: fresado, torneado, troquelado, prensado, etc.

**PROFUNDIDAD DE MAQUINADO** Se refiere a la distancia que penetra el cortador en el material.

"**S**" se utiliza para establecer la velocidad o número de revoluciones a las que debe girar la herramienta de corte.

**REFRIGERANTE** Aceite que se aplica a la herramienta para evitar el sobre calentamiento en el momento del corte.

**SISTEMA** "...Conjunto de dos o más elementos interrelacionados de cualquier especie..." según Russell L. Ackoff.

**SIMULADOR** Sistema que representa o imita el funcionamiento de algo.

**SOFTWARE** : Es la colección de programas, rutinas y documentos escritos en algún lenguaje de computadora para realizar alguna operación dentro de un computador.

**T** Es un código de identificación de un comando seleccionador de herramienta y el comando de compensación de herramienta.

**TÉCNICA** Conjunto de procedimientos y recursos de que se sirve una ciencia o un arte.

# BIBLIOGRAFÍA


- AMSTEAD B. H. OSTWALD P. F. BEGEMAN M. L.
  PROCESOS DE MANUFACTURA VERSION SI. MEXICO, 1989.
  EDITORIAL CONTINENTAL S. A.

- ARRIAGA, L.
  LAS MAQUINAS HERRAMIENTAS CON CONTROL NUMERICO. MEXICO, D.F. 1993.
  INSTITUTO POLITÉCNICO NACIONAL.

- CARL HANSER PUBLISHERS MUNICH, VIENNA, NEW YORK.
  PRACTICAL CNC-TRAINING PART 2: EXAMPLES AND EXERCISES, 1989.

- KERNIGHAN
  LENGUAJE DE PROGRAMACION C, 1986
  PRENTICE HALL.

- MONTERRUBIO HERRERA FRANCISCO ENRIQUE.
  TESIS PROFESIONAL. MEXICO, D.F., 1993.
  INSTITUTO POLITÉCNICO NACIONAL. ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA ELÉCTRICA.

- NADREAU R.
  EL TORNO Y LA FRESADORA

EDICIONES GUSTAVO GILI. MEXICO 1984.

- SCHILDT HERBERT.
C GUIA PARA USUARIOS EXPERTOS, 1991
MC GRAW HILL.

- SECRETARÍA DE EDUCACIÓN PÚBLICA DE MÉXICO.
LA EDUCACIÓN TÉCNICA EN MÉXICO (1980-1990).
S.E.P.

- TIEN-CHIEN CHANG, RICHARD A. WYSK Y HSU - PIN WANG.
COMPUTER AIDED MANUFACTURING. INTERNATIONAL AND SYSTEMS ENGINEERING. 1988
EDIT. PRENTICE HALL.

- TOLEDO MATUS.
FRESADORA. APUNTES PARA EL ALUMNO. MEXICO, 1989.
INSTITUTO POLITECNICO NACIONAL.

**MANUALES :**

- BORLAND.
TURBO C++ GUIA DEL PROGRAMADOR, 1990
MC GRAW HILL.

- MANUAL DE OPERACIÓN EMCOTRONIC TM 02, FRESADO. EMCO, 1991.

- SCHILDT HERBERT.
C MANUAL DE REFERENCIA, 1991.
MC GRAW HILL.

# ANEXOS

# MANUAL DEL USUARIO

El Sistema GC sirve para generar automáticamente listados de código CN para la fresadora EMCO-EMCOTRINIC.

En si, el Sistema GC funciona dentro de un sistema mayor CAD-CAM-CN, el cual obedece a tres fases principales .

1. El uso de algún CAD que ya exista en el mercado y que permita almacenar sus archivos en formato DXF, por ejemplo AutoCAD. Al crear estos archivos se deben usar únicamente las siguientes entidades de dibujo: líneas (LINE), arcos (ARC) y círculos (CIRCLE) debido a que el Sistema GC está diseñado para identificar solo a éstas. Cabe aclarar que los círculos son interpretados como barrenos. Estos archivos son almacenados en unidades, cada una de estas unidades es considerada como un milímetro por el Sistema GC. Para el caso de AutoCAD, se recomienda salvar exclusivamente la sección de entidades con el comando DXFOUT.

2. El Sistema GC corre en cuatro fases principales.

   - Extracción y despliegue de un archivo que el usuario seleccione a la vez. Estos archivos deberán estar almacenados en formato DXF y pueden ser creados en el paquete de CAD.

   - Captura de variables que establecen las condiciones de fresado: selección del origen, altura de seguridad, uso de refrigerante, datos velocidad de corte del material.

   - Selección de la trayectoria de fresado de manera automática o manual. Especificación de la profundidad de fresado, las herramientas.

   - Generación de los siguientes resultados: representación gráfica del fresado, la generación del código de CN y la salida a impresora.

3. El listado de CN generado por el Sistema GC se introduce a la fresadora y se ejecuta el maquinado

## USO DEL SISTEMA GC

El Sistema GC, corre bajo el ambiente DOS de las máquinas PC. Para instalarlo solo es necesario copiar sus archivos al disco en el cual se va a trabajar. Se ejecuta con el nombre SGC.

Una vez iniciado el sistema, se despliega en pantalla un menú que consiste de cuatro opciones: Archivo, Inicio, Proceso y Salidas. Cada una de estas opciones pueden ser seleccionadas mediante el desplazamiento hacia la izquierda o a la derecha con el uso de las flechas, con el uso del mouse o bien al teclear la letra mayúscula del menú deseado. Una ventana de ayuda es desplegada al teclear F10.

A continuación se describe cada opción

### ARCHIVO

La opción de ARCHIVO consiste a su vez de cinco opciones :

Cargar, salvaR, Renombrar, Borrar, Salir. Para seleccionar cada uno de ellos se puede desplazar el cursor hacia arriba y hacia abajo con el uso de las flechas, con el uso del mouse o bien al teclear la letra mayúscula de la opción deseada. A continuación se describe cada una de ellas:

#### CARGAR

La opción de cargar despliega una ventana con los nombres del directorio de la ruta actual. Para cambiar de ventana se puede hacer uso de la tecla del tabulador (TAB) y para seleccionar un archivo se puede teclear el nombre completo en la parte superior de la ventana o bien con retorno de carro en la ventana inferior. Una vez seleccionado el archivo a cargar se extraen sus entidades y se

despliegan en pantalla si es posible. Si no es posible, es decir que el archivo no sigue el formato necesario se muestra un mensaje de error.

"Cargar" solo trabaja con archivos .DXF o .F1. Los archivos .DXF son los que se encuentran almacenados directamente en formado DXF . Los archivos F1 son los que ya han sido trabajados con el Sistema GC y en ellos se encuentran valores propios del fresado. Cada unidad del Sistema GC representa un mm. La ventana que se despliega en gris obscuro representa la mesa de la fresadora, sus dimensiones reales son de 185 mm. por 100 mm.

**SALVAR**

La opción de salvar se encarga de almacenar en disco el dibujo que está activo en la memoria de la computadora. Además de almacenar las entidades de dibujo, guarda los valores de las diferentes variables de fresado como son: Origen, Altura, Refrigerante, Material, Trayectoria, Herramientas y Profundidades. La extensión de default es .F1.

**RENOMBRAR**

Parte del despliegue de la ventana de directorios, que funciona de la misma manera que la de "Cargar" Una vez seleccionado el archivo a renombrar, se despliega otra ventana de lectura del nuevo nombre. Una restricción para esta opción es que no se puede cambiar la ruta del archivo, solamente su nombre

**BORRAR**

Parte del despliegue de la ventana de directorios, que funciona de la misma manera que la de "Cargar" Una vez seleccionado el archivo a borrar, se despliega otra ventana de confirmación de borrado del archivo. La respuesta puede se "s" o "n" , o bien ESC para cancelar la tarea.

**SALIR**

Sirve para terminar la ejecución del Sistema GC y regresar al DOS.

# INICIO

Esta opción consiste a su vez de cinco opciones Origen, Altura, Refrigerante, Material y Herramientas. Para seleccionar cada uno de ellos se puede desplazar el cursor hacia arriba y hacia abajo con el uso de las flechas, con el uso del mouse o bien al teclear la letra mayúscula de la opción deseada. Estas cuatro opciones son las variables, necesarias para el fresado, que permanecen constantes y solo se leen una vez para cada archivo de trabajo. A continuación se describe cada uno :

## ORIGEN

Sirve para establecer el origen de coordenadas del fresado. Puede seleccionarse de tres formas diferentes :

1. Tecleando numéricamente las coordenadas exactas (x,y)

2. Seleccionando el punto en la pantalla donde se quiere establecer el origen. El desplazamiento en pantalla puede hacerse con el uso de las flechas o del mouse. Los segmentos de desplazamiento de las flechas se pueden agrandar o acortar con las teclas + y -.

3. Seleccionando alguna entidad con el uso del mouse o de las flechas y colocando el origen en el extremo de la entidad más cercano al punto elegido.

## ALTURA

Se refiere a la altura de seguridad. La altura de seguridad es la distancia que se mide desde la superficie del material a fresar hasta el cortador. Sirve para desplazar el cortador de un punto a otro sin cortar o dañar el material. También sirve para proteger la herramienta.

## REFRIGERANTE

Para la lectura del refrigerante es suficiente con teclear "s" o "n", según se desee trabajar o no con él El refrigerante es un aceite que evita el sobre calentamiento de la herramienta.

## MATERIAL

Dentro de esta opción hay a su vez dos opciones: Dimensiones del material y Velocidad de corte del material. La primera sirve para redibujar el material, en forma rectangular, de acuerdo a las dimensiones que se le indiquen al Sistema GC. El objetivo de esta opcion es ayudar al usuario a ver, con mayor claridad, el tamaño conveniente de su material para el fresado. Los datos necesarios que se deben proporcionar son las coordenadas en x,y de las esquinas inferior izquierda y superior derecha del material. La segunda se refiere a la velocidad (en revoluciones por minuto) de fresado del material. Es importante usar la velocidad apropiada para evitar el sobre esfuerzo de la fresa y su rompimiento.

## HERRAMIENTAS

"Herramientas" sirve para almacenar las posibles herramientas a usar durante del fresado. Presenta una ventana con números del 1 al 10 que corresponden, uno a uno, a todas las herramientas existentes fisicamente en la fresadora. La tarea del usuario en esta opción es hacer corresponder las características reales de las herramientas que desee usar (diámetro, numero de aristas cortantes, avance) con las correspondientes en el Sistema GC. De tal forma que si el usuario selecciona un número de herramienta, la computadora despliega una ventana que pregunta que tipo de herramienta es, broca o fresa. Si el usuario marcó broca, se despliegan las preguntas : diámetro de la broca números de aristas cortantes. Si seleccionó fresa, además de las dos preguntas anteriores se despliega la pregunta . avance por mm del material. El objetivo de conocer estos datos, es el de poder calcular la velocidad y el avance de corte, necesarios para generar el código CN.

## PROCESO

La opción de PROCESO consiste de cuatro opciones: Trayectoria, Profundidad y Herramienta Para seleccionar cada uno de ellos se puede desplazar el cursor hacia arriba y hacia abajo con el uso de las flechas, con el uso del mouse o bien al teclear la letra mayúscula de la opción deseada.

## TRAYECTORIA.

La TRAYECTORIA es la opción que permite al usuario indicar el orden y la dirección en que quiere que se fresen las entidades. Si no se especifica la trayectoria, el código no puede ser generado por que no se sabe que entidad se va a fresar primero y por cual seguir. En le Sistema GC, la trayectoria puede especificarse de dos maneras diferentes: automáticamente y manualmente.

En la trayectoria manual el usuario puede seleccionar, con el uso de las flechas o del mouse, las entidades que quiere fresar, agrupándolas dentro de diferentes bloques. Cada bloque se identifica con un número entero. Al seleccionar cada entidad el usuario indica automáticamente la dirección en que quiere que se frese. Esta selección puede darse con el uso de las flechas del teclado o con el mouse. La velocidad y precisión de desplazamiento con las flechas puede aumentarse o disminuirse con el uso de las teclas + o -. Esta dirección va del punto inicial al final, donde el punto inicial es aquel que esté más cerca del punto marcado por el usuario. Dentro de la selección manual, para indicar que ya se quiere finalizar con la selección de entidades de un bloque, se teclea ESC. La selección manual no acepta modificaciones en entidades específicas pero si admite eliminar bloques completos, de tal forma que si el usuario comete algún error, por ejemplo que seleccione la dirección de fresado de una entidad de manera equivocada, para corregirlo tiene que eliminar el bloque completo y volver a seleccionar.

En la trayectoria automática, la computadora se encarga de escoger una trayectoria de fresado Inicialmente se pregunta al usuario si desea aumentar barrenos a su dibujo, con el objeto de añadirlos donde sea necesario para no causar daños a las fresas en el caso de que el materia sea muy duro y/o la profundidad no sea muy pequeña. Posteriomente encuentra aquella ruta que frese la mayor cantidad de entidades de manera continua. Sus limitaciones son: que el usuario no puede modificarla por si mismo, que solo permite un cambio de herramienta (de broca a cortador), que admite una misma profundidad para todas las entidades.

## HERRAMIENTA

La opción de herramienta solamente puede ser activada después de la selección de la trayectoria y de preferencia antes que la de la profundidad ya que ésta última trabaja en base al tipo de herramienta seleccionado y después de la opción "Herramientas" del menú "INICIO". Si la trayectoria ué automática, se captura un número existente en el menú de herramientas para la broca y otro para la

fresa. Si la trayectoria fué manual, se pregunta al usuario el número de bloque al cual se le va a asignar una herramienta y, por supuesto, el número de la herramienta.

## PROFUNDIDAD

La opción de la profundidad solamente puede ser activada después de la selección de la trayectoria y de preferencia, después de la selección de la herramienta. Si la trayectoria fue automática, se captura la profundidad total y por pasada para dos bloques de entidades: la broca y el cortador. Si la trayectoria fué manual se pregunta al usuario el número del bloque y se captura la profundidad total y por pasada de éste La profundidad total es la máxima distancia (en mm.) que va a penetrar el cortador en el material. La profundidad por pasada es la máxima distancia (en mm.) que va a bajar el cortador n el material cada vez que pasa por el material. Al indicar la profunidad se debe cuidar que no sufra fracturas la herramienta, que depende de la dureza del material y de las características del cortador; y que la profundidad total no sobrepase el ancho del material para no ocasionar daños en la mesa de la fresadora

# SALIDAS

La opción de SALIDAS consiste de tres opciones : Simular, Código CN e Imprimir. Para seleccionar cada uno de ellos se puede desplazar el cursor hacia arriba y hacia abajo con el uso de las flechas, con el uso del mouse o bien al teclear la letra mayúscula de la opción deseada. A continuación se describe cada una de ellas

## SIMULAR

Realiza la representación del fresado en pantalla a partir de los datos indicados en las opciones de ARCHIVO, INIICO y SALIDA. Proponen dos modos diferentes de simulación: el automático y el manual. En el automático se ejecuta la simulación desde el principio hasta el final sin parar, en el manual se hace una pausa, que se puede romper presionando cualquier tecla, cada vez que se interpreta un comando de control numérico.

## CÓDIGO CN

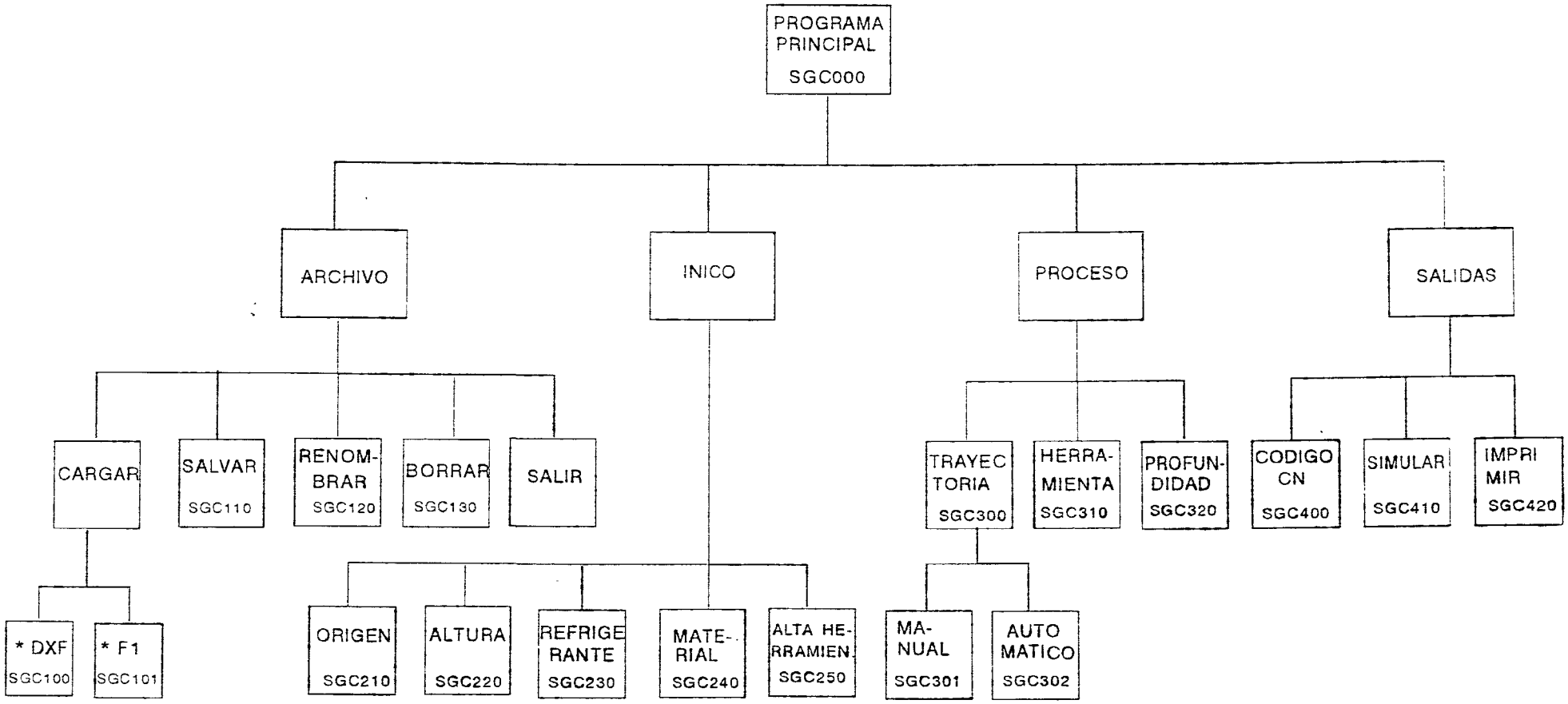
CÓDIGO CN es la opción del Sistema GC que genera los códigos de control numérico. El usuario deber recorrer las opciones necesarias del menú de ARCHIVO, INICIO y PROCESO antes de tratar de generar los códigos. Después de que el usuario activa esta opción debe introducir el nombre del archivo con el que desea generar el código en disco. El archivo se genera en código ASCII automáticamente a partir del archivo activo en memoria y de los datos introducidos.

## IMPRIMIR

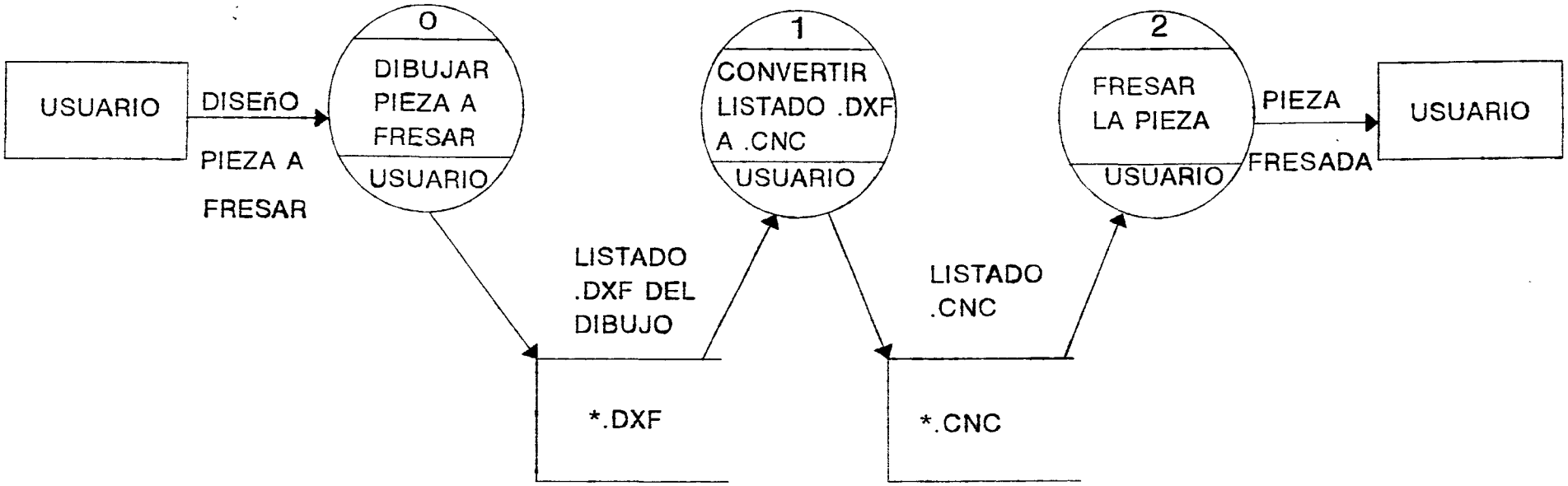
Parte del despliegue de la ventana de directorios, que funciona de la misma manera que la de "Cargar". Una vez seleccionado el archivo ASCII a imprimir se produce la salida.

Una vez generado el código, se puede realizar la tarea final: introducir el código CN a la fresadora y realizar el maquinado. Para introducirlo el usuario puede teclearlo directamente sobre el teclado de la fresadora, o bien puede pasarlo directamente de la computadora a la fresadora en un diskette de 3 5" de baja densidad. Este diskette debe ser formateado inicialmente con el sistema operativo DOS, como cualquier diskette y a continuación se le debe dar un formato especial que requiere la fresadora EMCO-EMCOTRONIC, y que puede ser generado con el paquete comercial EDISK, diseñado especilamente para esta máquina-herramienta por la compañia EMCO de Austria.

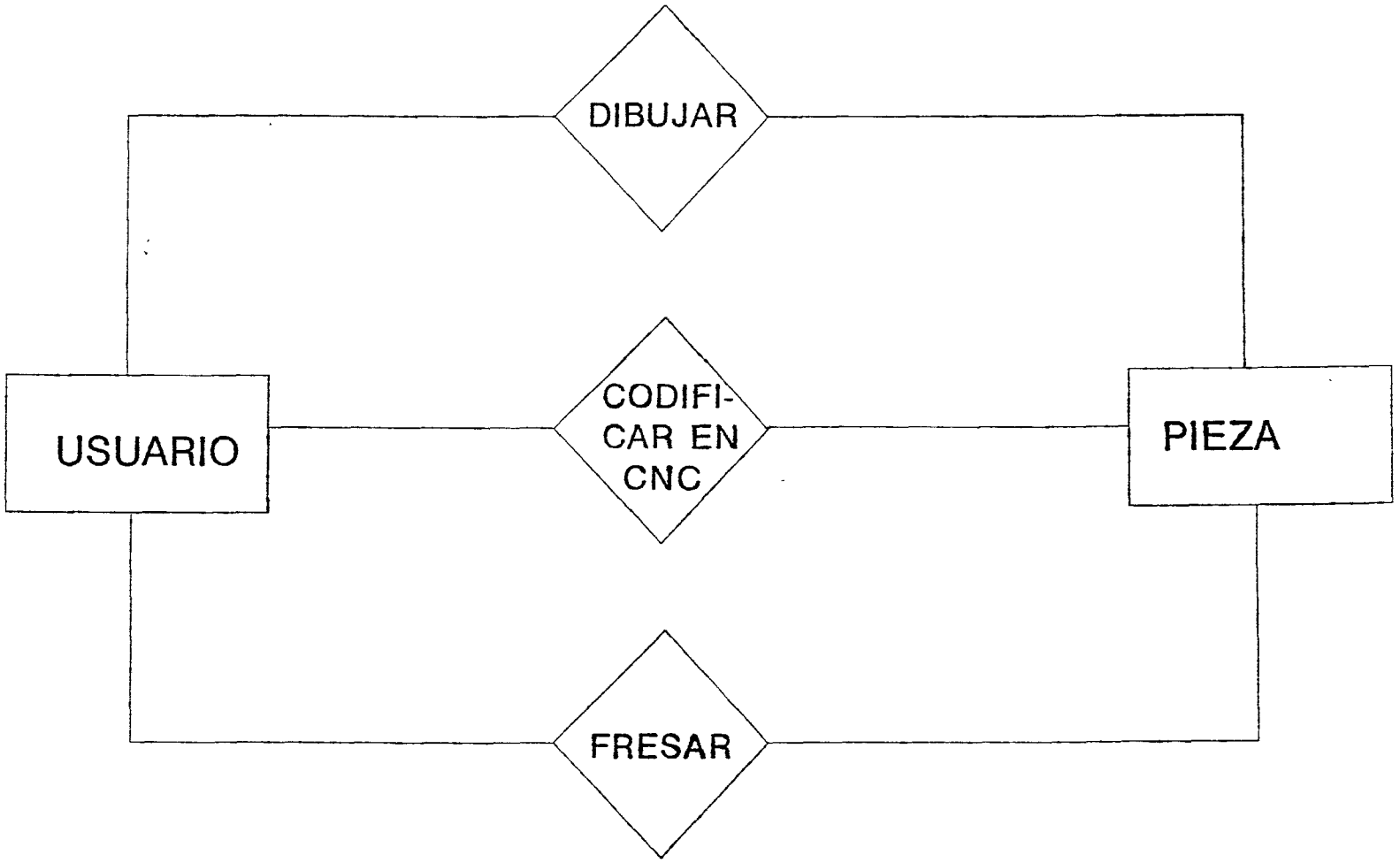
# DIAGRAMA DE NAVEGACION DEL SISTEMA GC



# CAD - SISTEMA GC - FRESADORA
# DIAGRAMA DE FLUJO DE DATOS

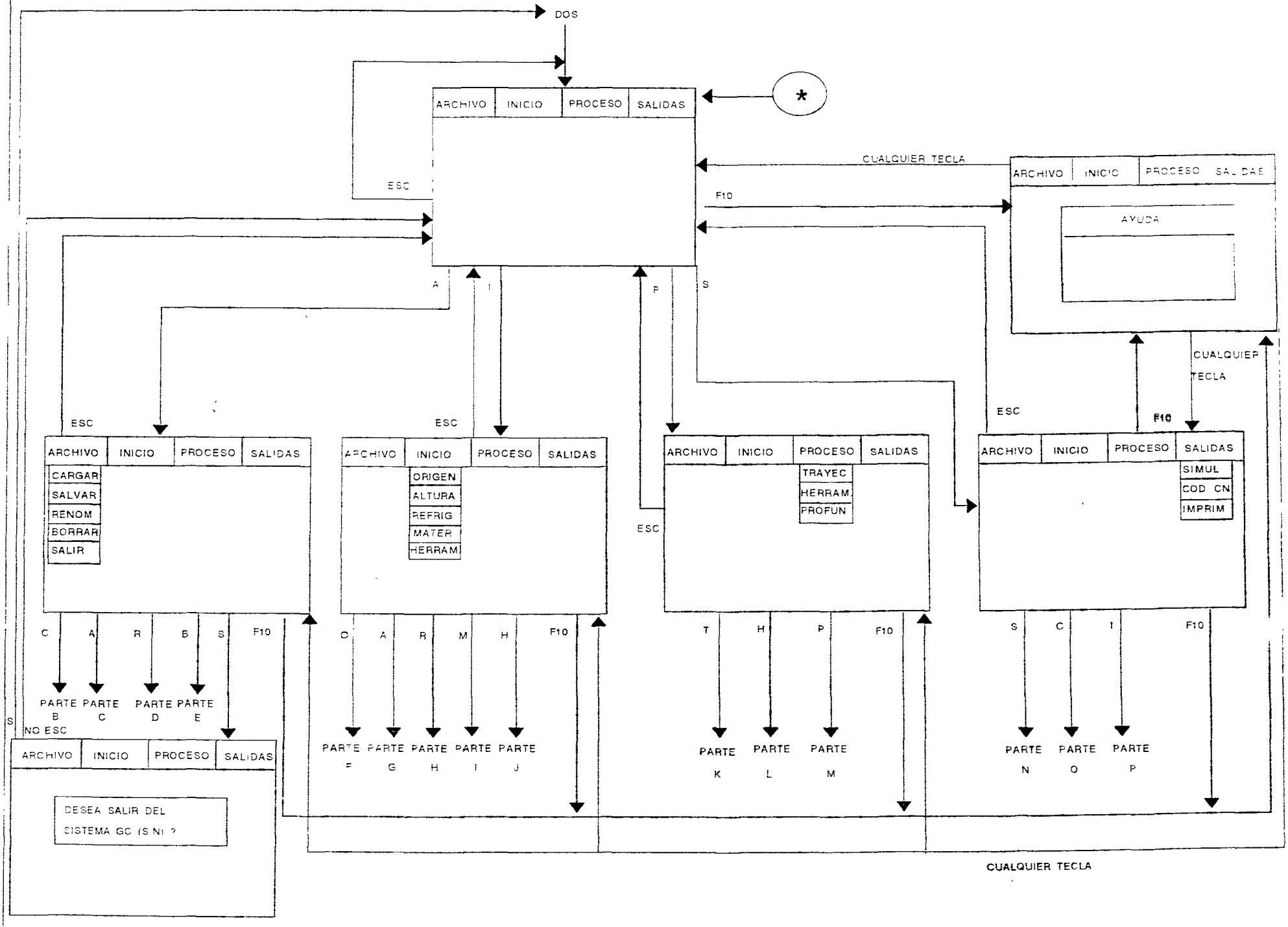


# CAD - SISTEMA GC - FRESADORA
# DIAGRAMA DE ENTIDAD RELACION



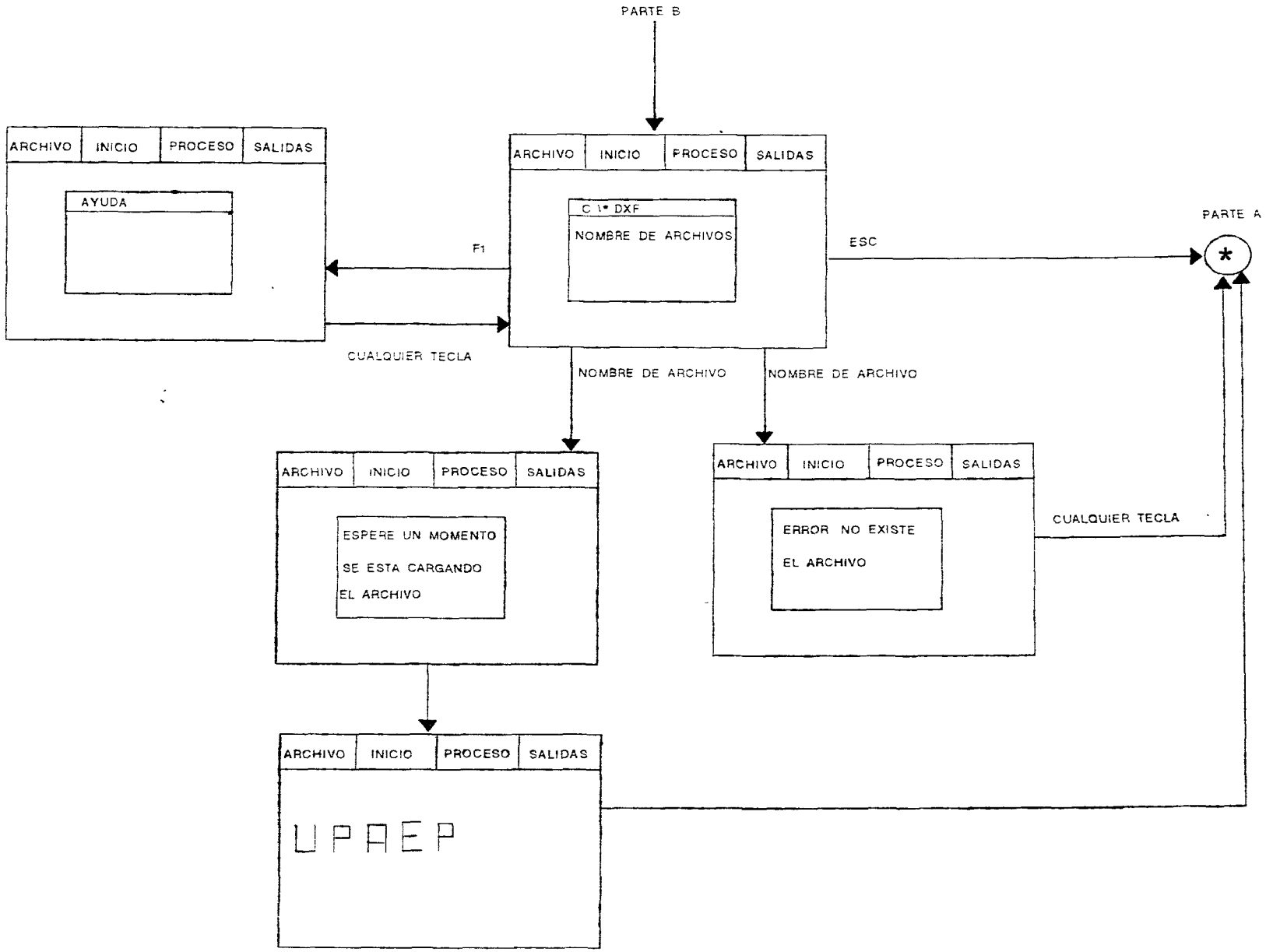
# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE A



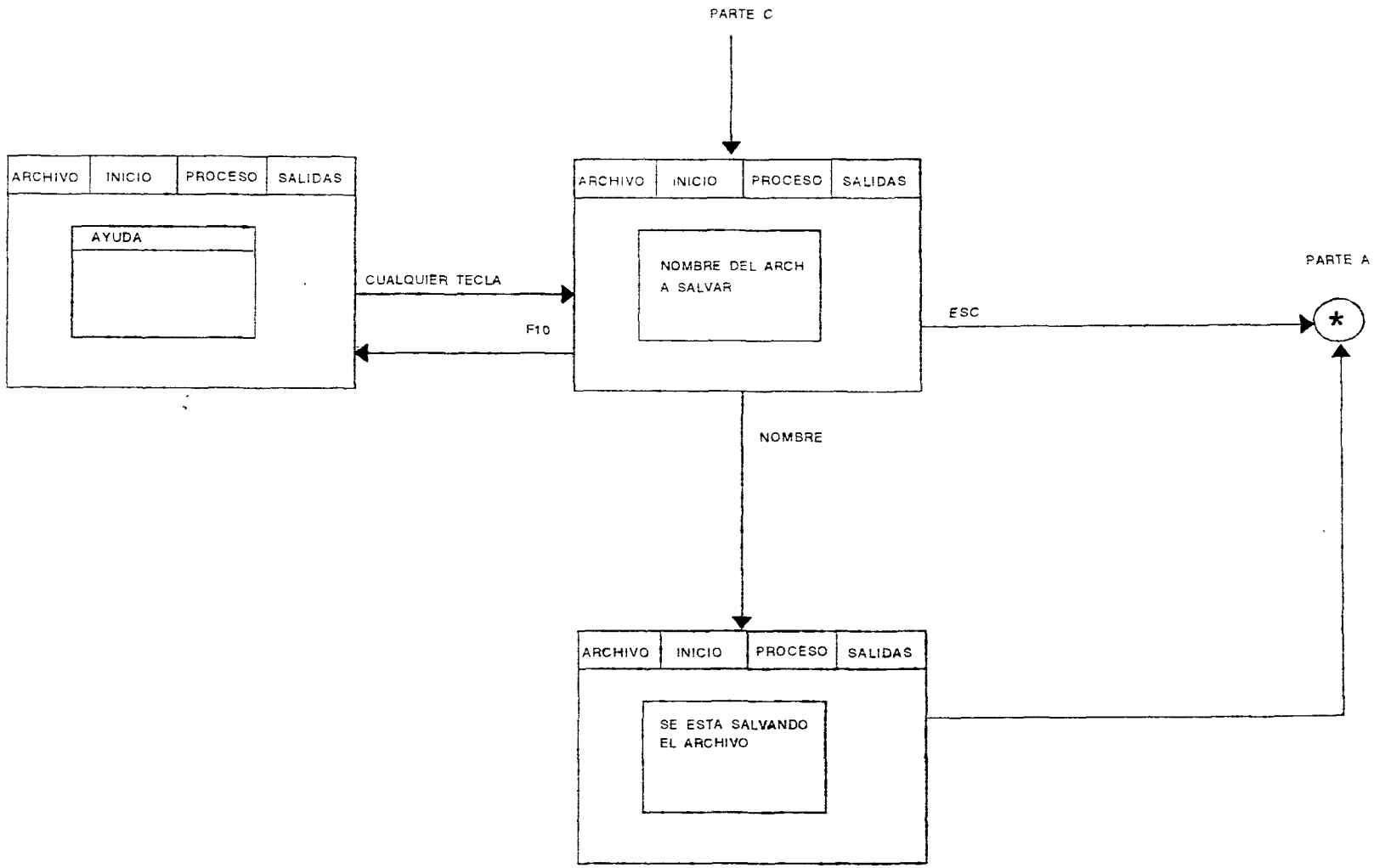
# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE B



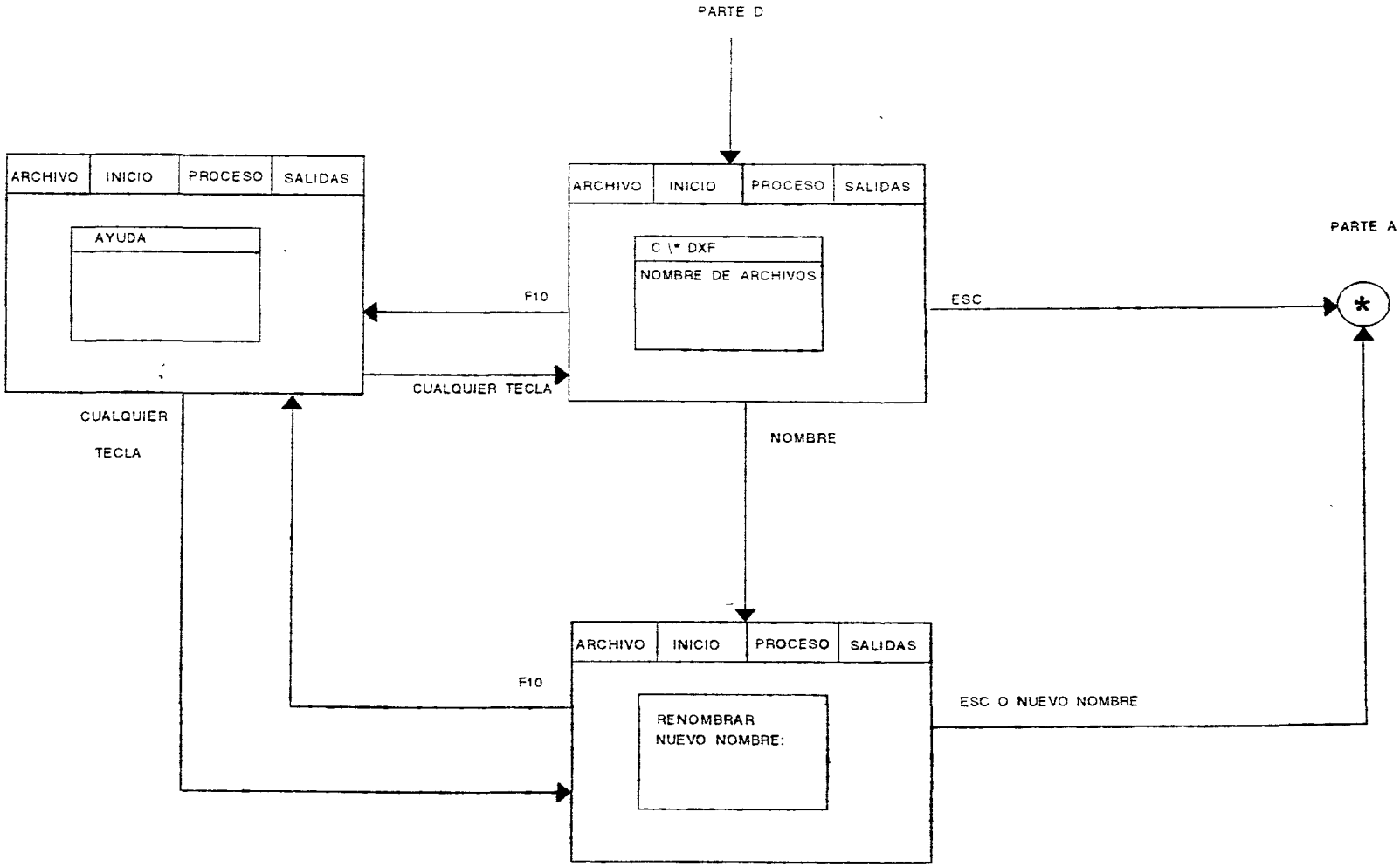
# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE C



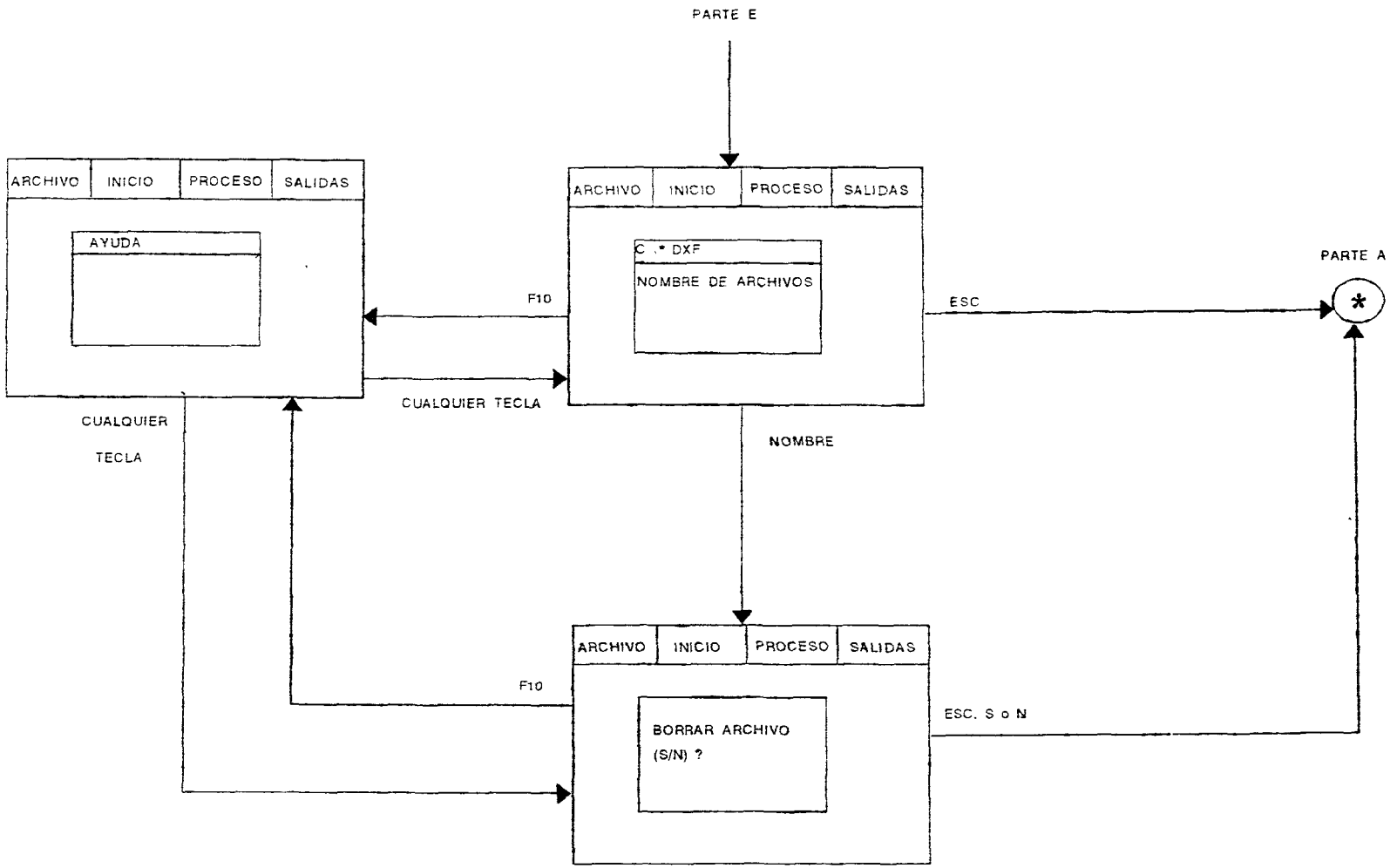
# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE D



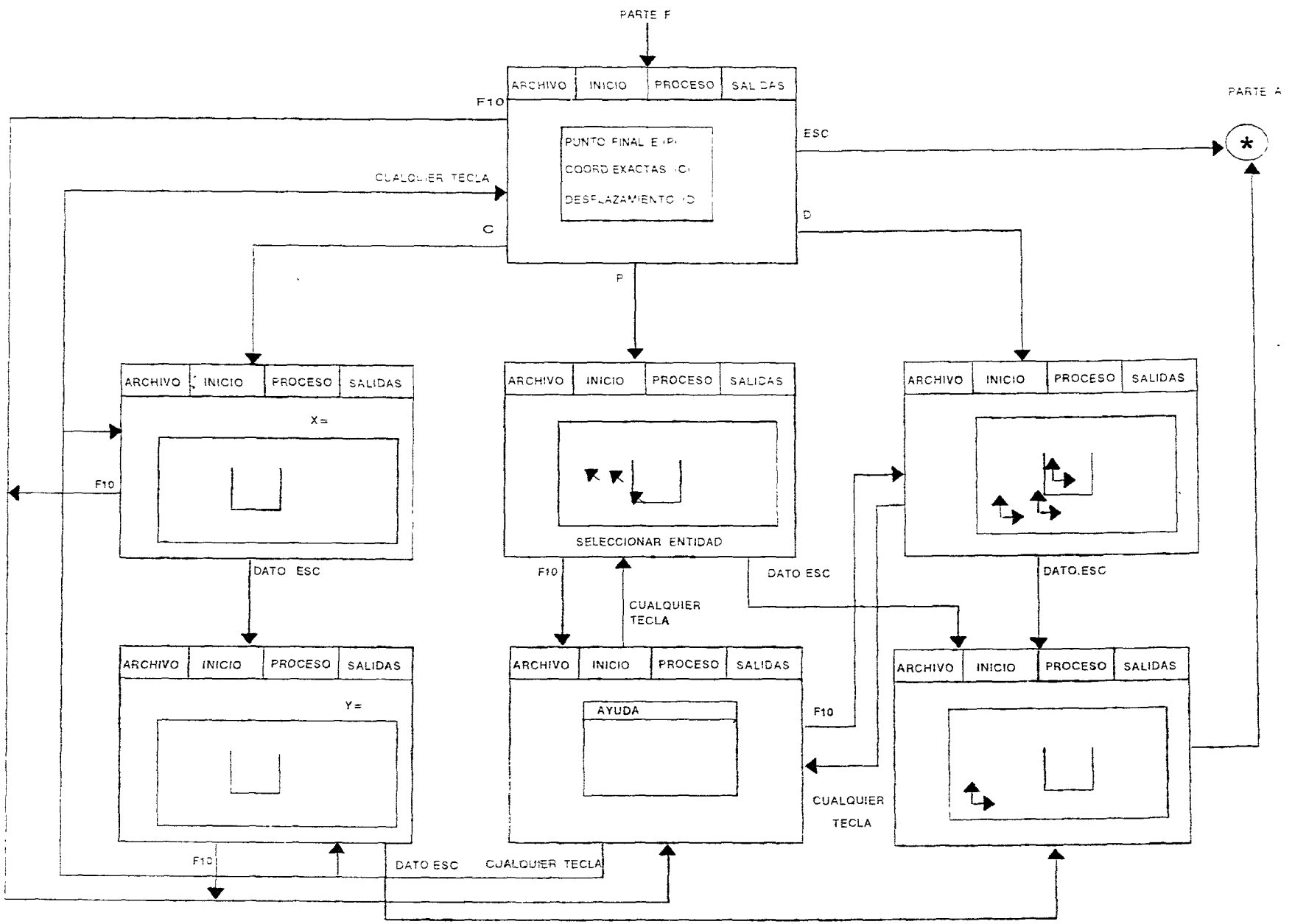
# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE E



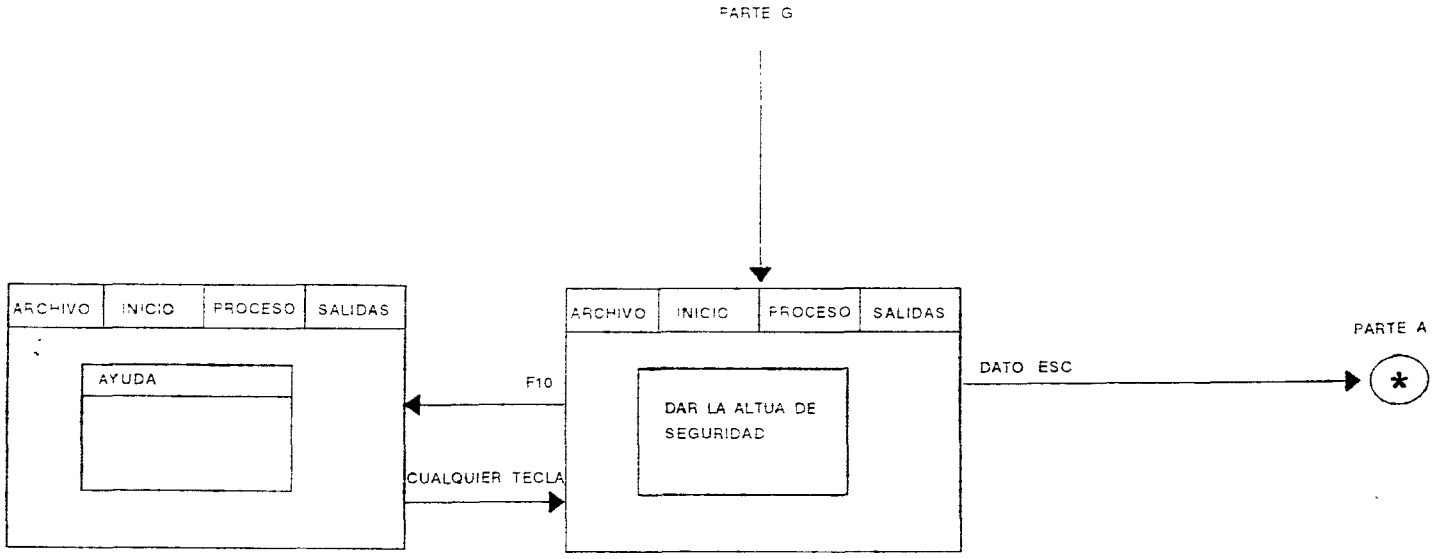
PARTE F

# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC



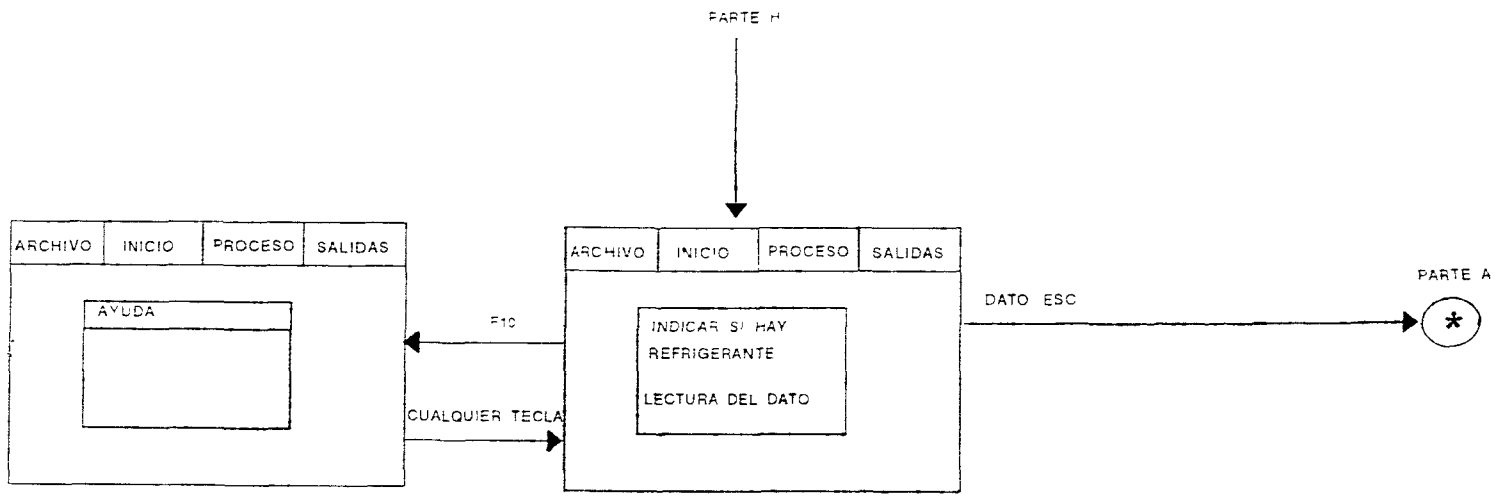
# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE G

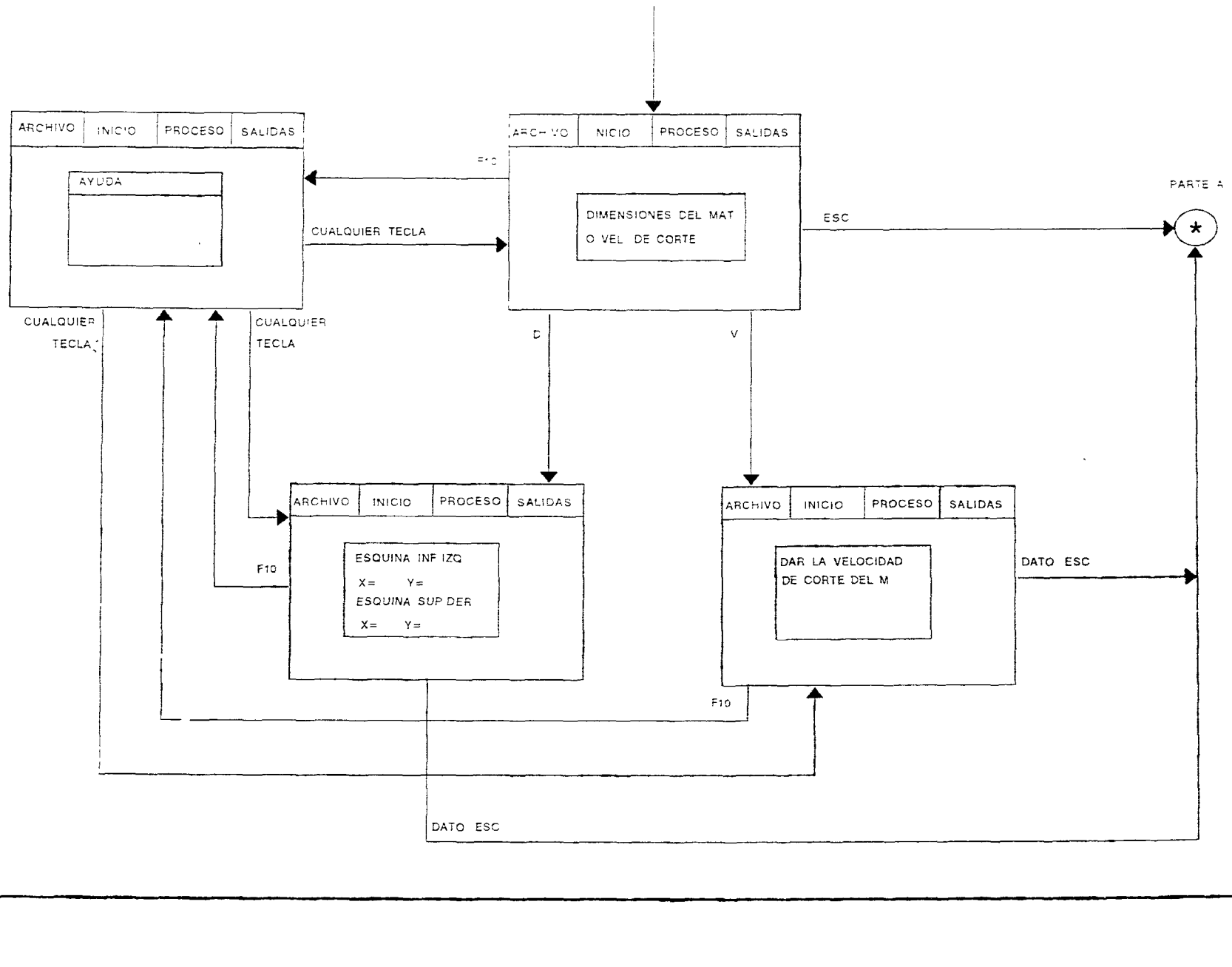


# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE H

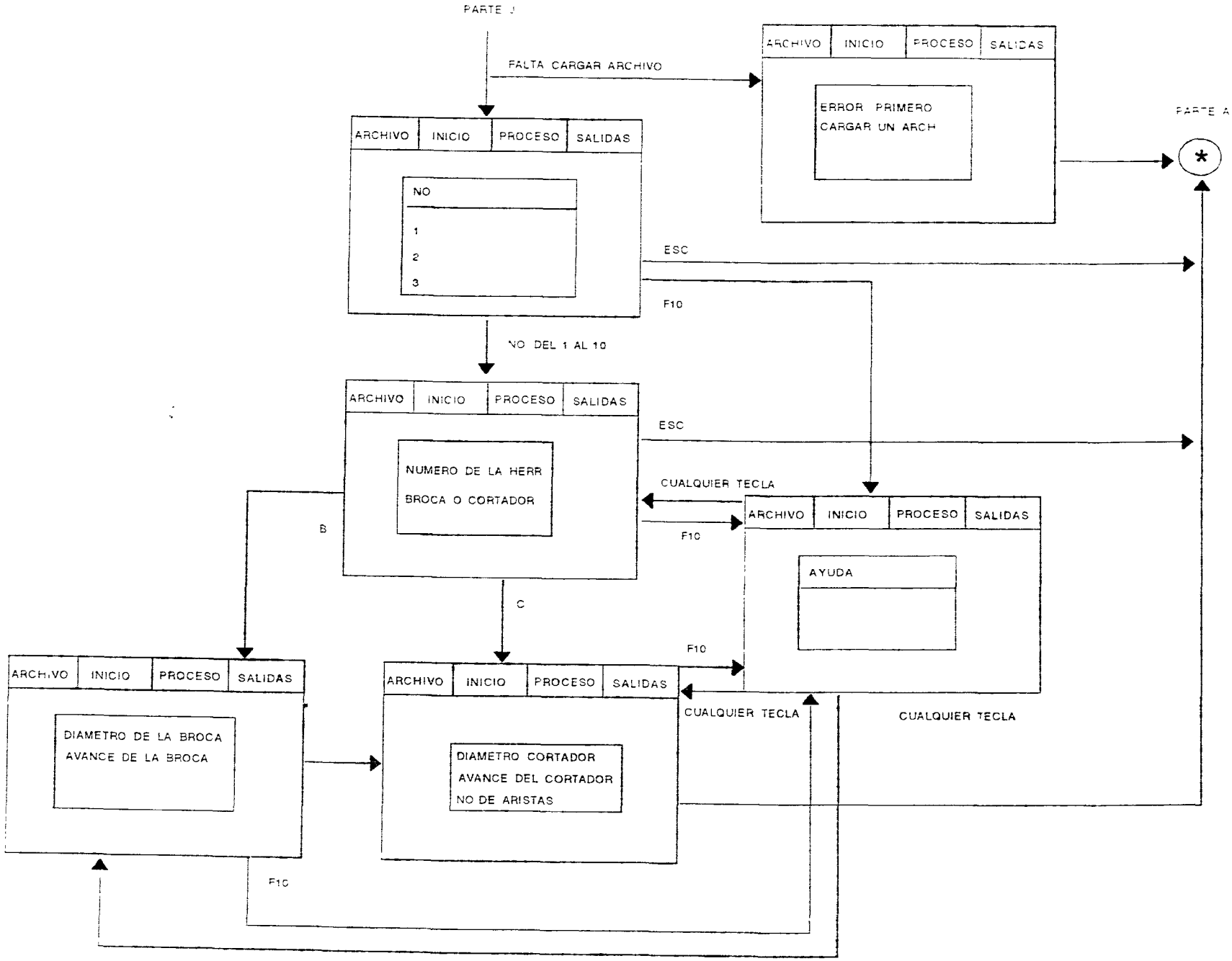


# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC



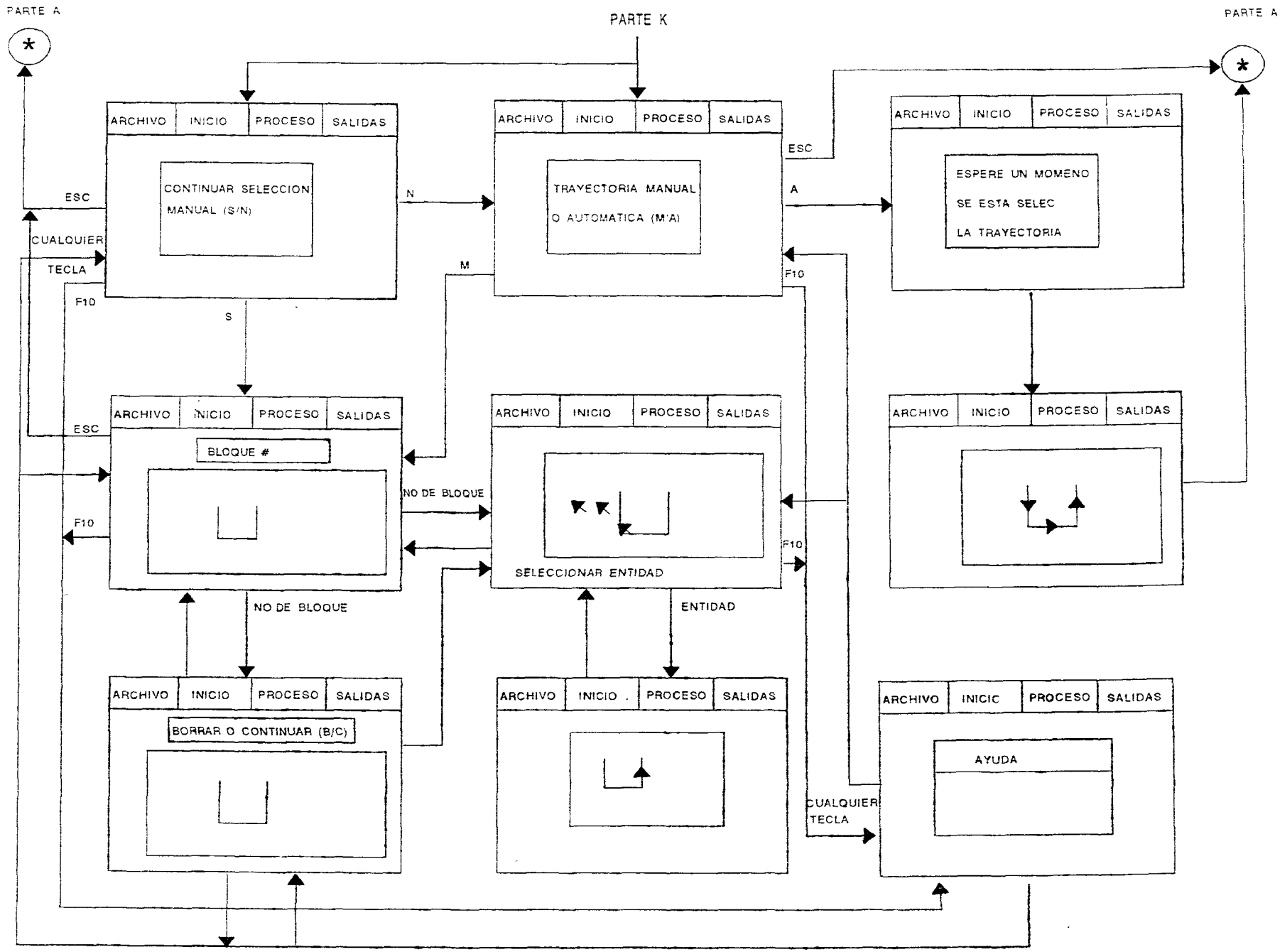
# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE J



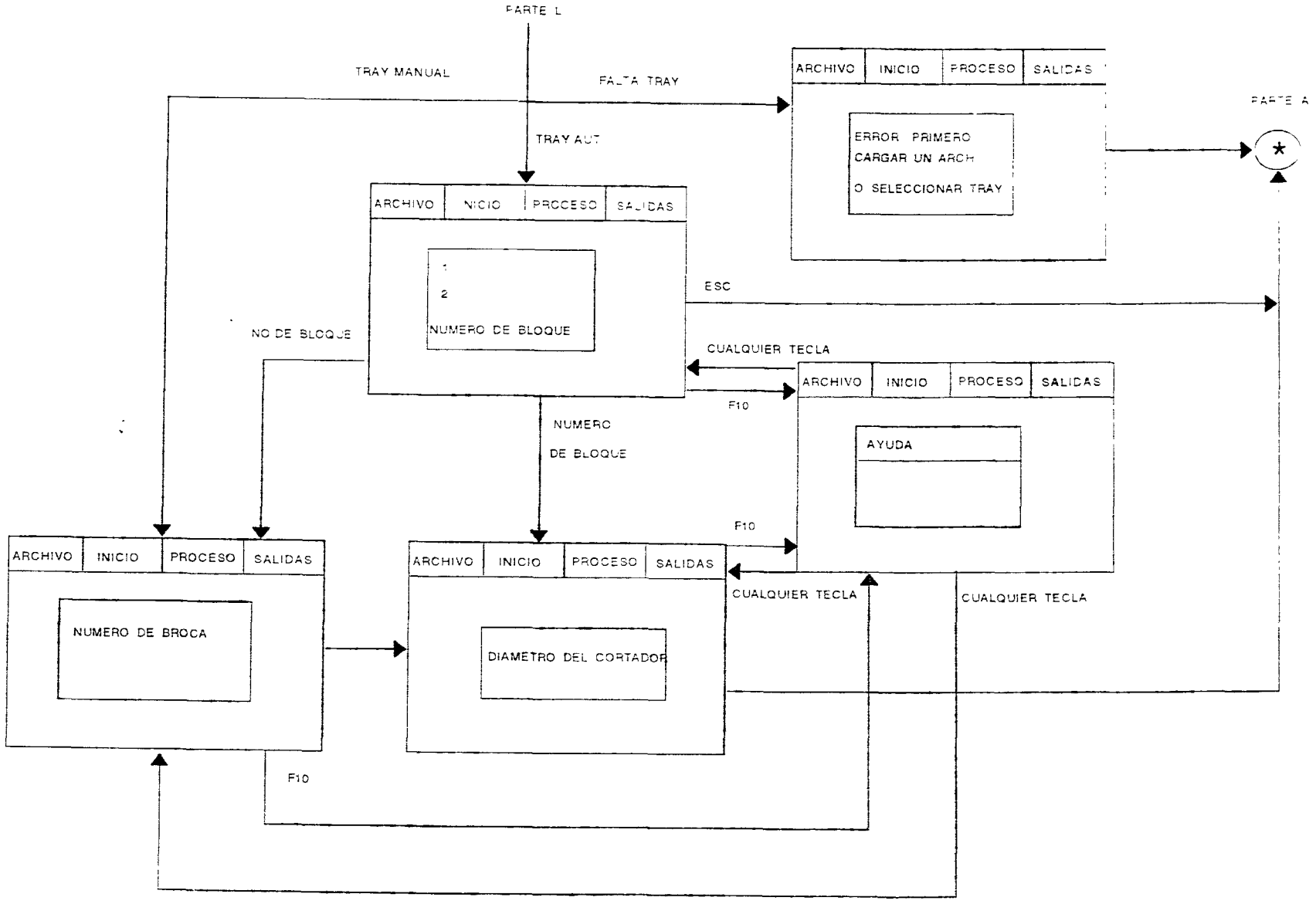
# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE K



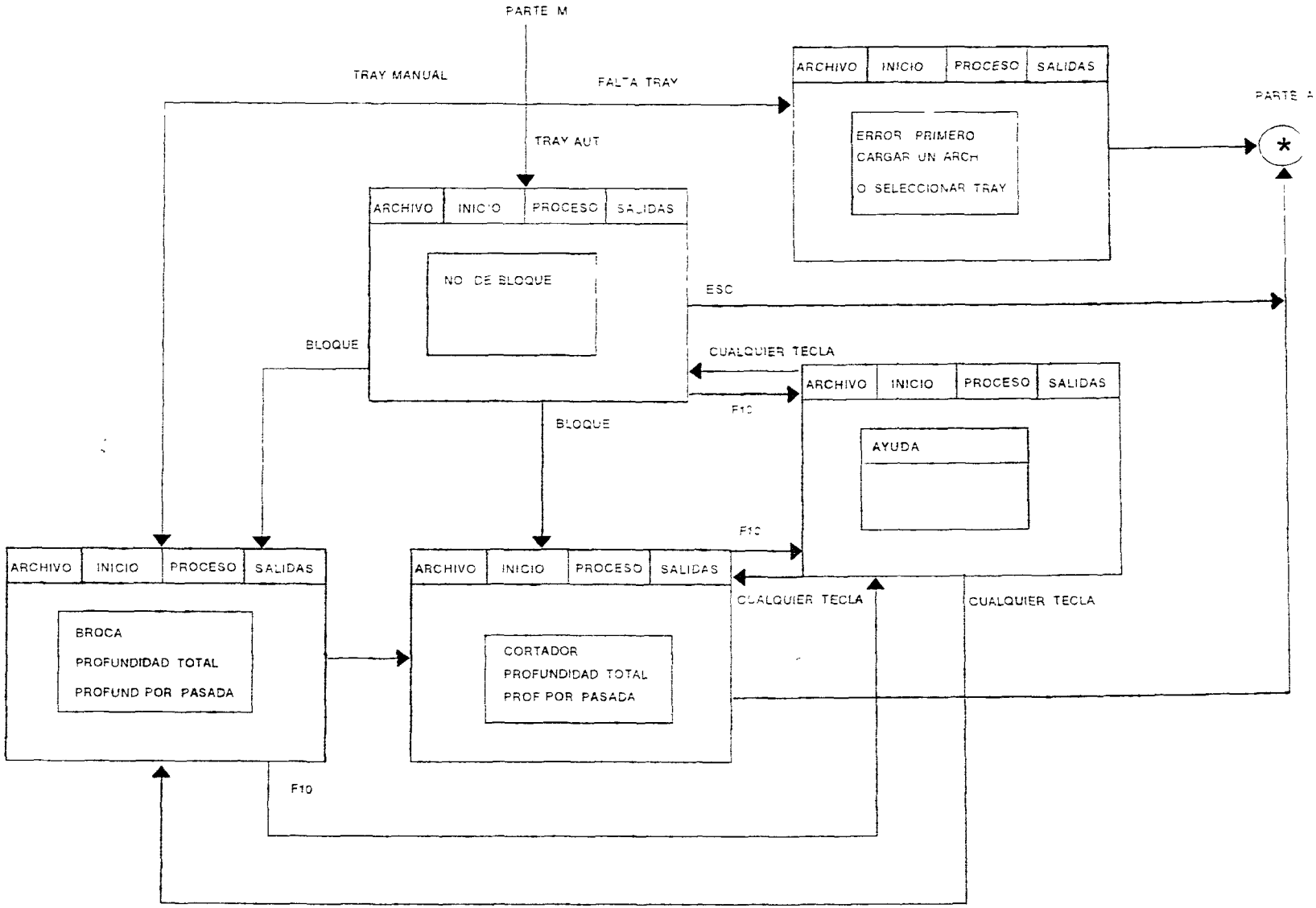
# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE L



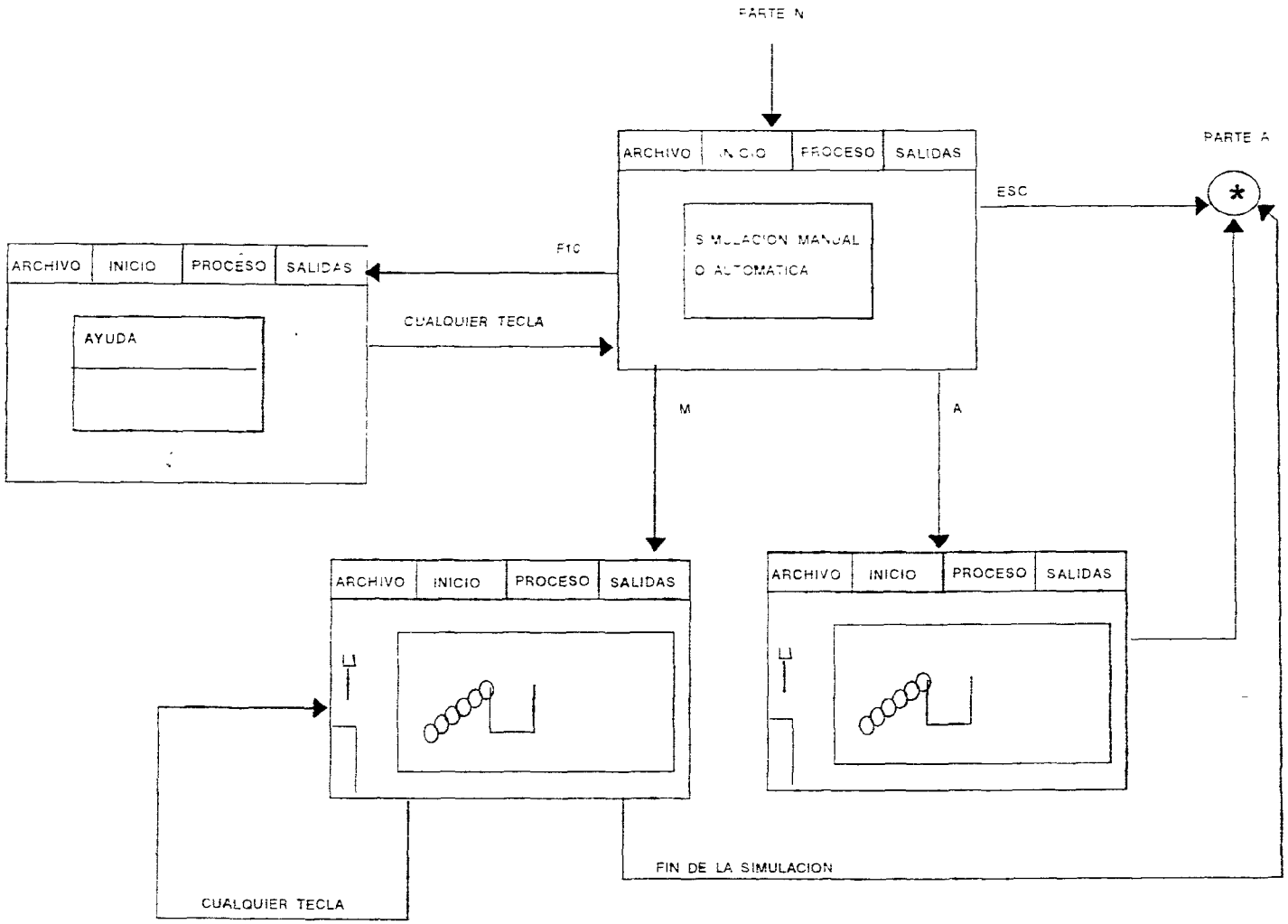
# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC





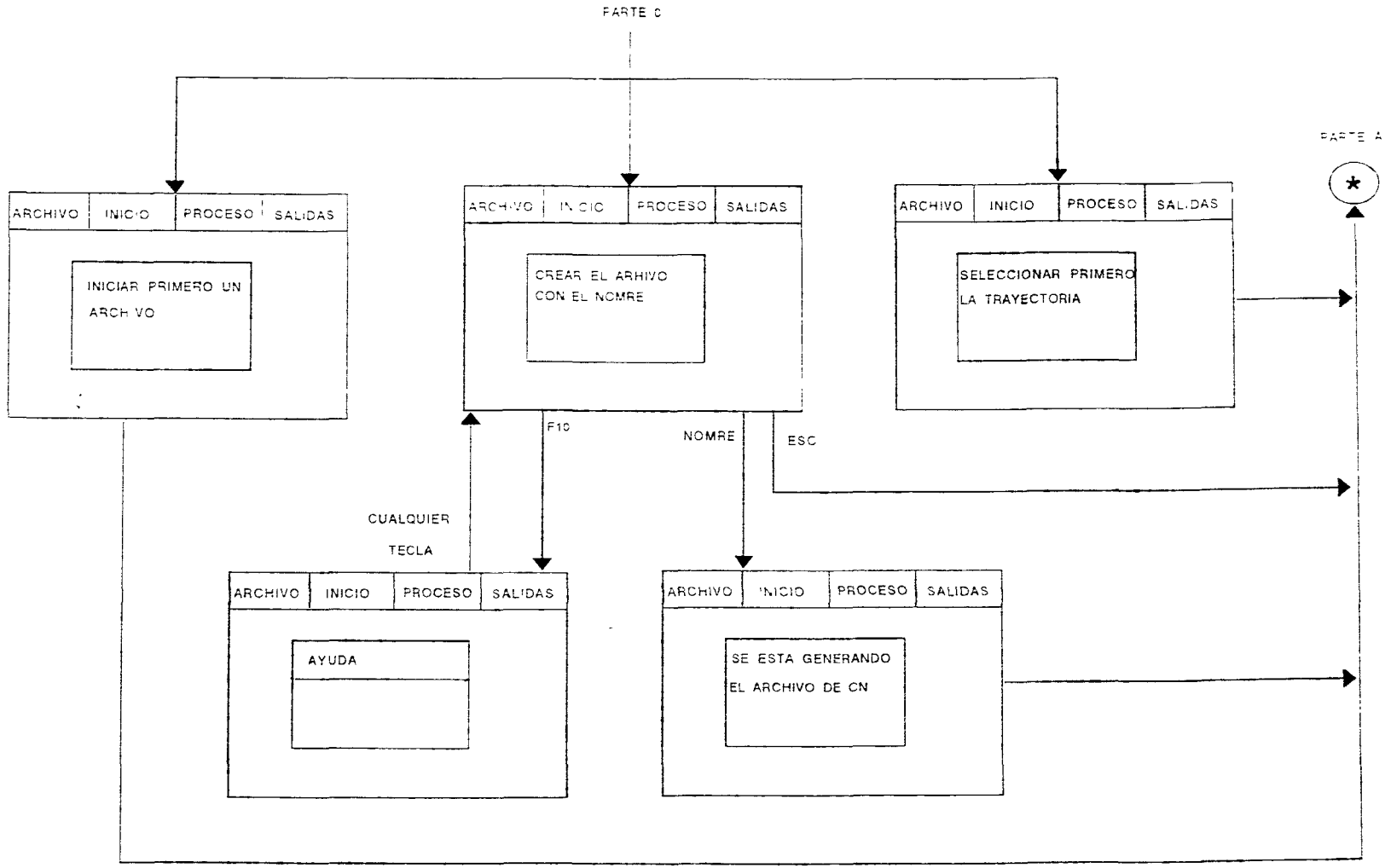
# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE N



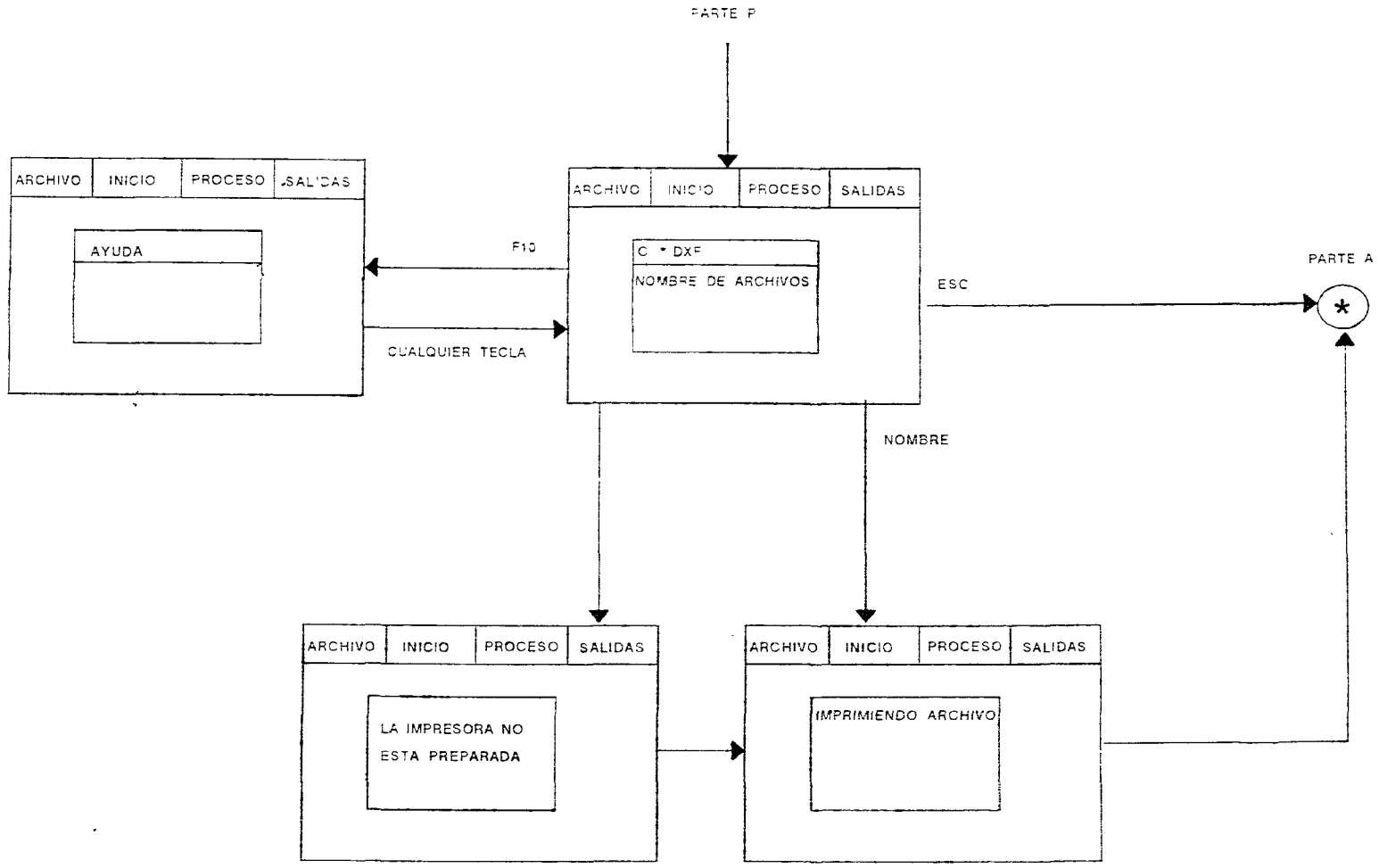
# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE O



# DIAGRAMA DE TRANSICION DE ESTADOS DEL SISTEMA GC

PARTE P



# LISTADO DEL SISTEMA GC

```
/*****************************************************/
/*              C O N S T A N T E S                  */
/*****************************************************/

#include <stdio h>
#include <dir.h>
#include "sgclc.h"

#define ESC             0x1b            /* Define the
escape key  */
#define BS          8
#define F10         68
#define ARRIBA      72
#define ABAJO       80
#define DER         77
#define IZQ         75
#define HOME        71
#define END         79
#define UP          73
#define DOWN        81
#define RETURN      13
#define TAB         9
#define SIZE_DIR    12
#define SIZE_CHAR_DIR  15
#define MOUSE_NO_MOVIDO 0
#define MOUSE_DER    1
#define MOUSE_IZQ    2
#define MOUSE_ARR    3
#define MOUSE_AB     4
#define IZQB         1
#define DERB         2
#define MAXMENU      13
#define DIS          0 0001
#define CapX1        180
#define CapX2        470
#define CapY1        50  // 150
#define CapY2        115  // 215
#define CapSizeX     290
#define CapSizeY     65
#define ESCALA 3   //DXF
#define ESCALA1 35
#define MesaX 80
#define MesaY 135  // 100
#define AnchoMesa 300
#define LargoMesa 555

#define TRUE  1
#define FALSE 0

#define MAX_BLOQUES 50

/*****************************************************/
/*              P R O T O T I P O S                  */
/*****************************************************/

void lee_raton (int, struct actMouseArray_t *, int *),
int lee_teclado (struct actMouseArray_t *, int *),
void EjecutaSubMenu (void);
void Salida(void);
void ayuda (int op),
void getpath(void);

/*****************************************************/
/*   V A R I A B L E S   G L O B A L E S             */
/*****************************************************/

extern void *BuffWin,
extern int nivel;
extern int MouseInstalado,
extern int x,y,z,RadioH;
extern char OnOff;
extern char compen,
extern int   GraphDriver,                /* The
Graphics device driver                   */
extern int   GraphMode;                  /* The
Graphics mode value                      */
extern double AspectRatio;               /* Aspect ratio
of a pixel on the screen*/
extern int   MaxX, MaxY;                 /* The
maximum resolution of the screen */
extern int   MaxColors;                  /* The
maximum # of colors available            */
extern int   ErrorCode;                  /* Reports any
graphics errors                 */
extern struct palettetype palette;                  /*
Used to read palette info       */
extern char menu1[4][MAXMENU],
menu2[5][MAXMENU],

            menu3[5][MAXMENU],
menu4[4][MAXMENU],

            menu5[4][MAXMENU];
extern int Nmenu1, Nmenu2, Nmenu3, Nmenu4,
Nmenu5,
extern int AreaMouse [30][4],
extern FILE *fp;        //DXF
extern int w;
extern char mypath[128],
```

```
typedef enum {
 ENTITIES,
 ENDSEC,
 INICIO,
 DATOS,
 LINE,
 ARC,
 CIRCLE
} TYPE,

typedef enum {
 HERR_BROCA,
 HERR_COTADOR
} HERR_T;

typedef enum {
 REFRIGERANTE_NO,
 REFRIGERANTE_SI
} REFR_T,

#define ON  1
#define OFF 0

 /**DXF**DXF** ESTRUCTURA QUE ALMACENA LAS
ENTIDADES *******DXF DXF DXF**/
 struct nododxf {
          TYPE entidad;
          union {
           struct {
            float x1, y1;
            float x2, y2,
           } line,
           struct {
            float x1, y1,
            float x2, y2,
            float h, k;
            float r,
            float a1, a2,
           } arc,
           struct {
            float h, k;
            float r,
           } circle;
           struct {
            float compensacion,
            float prof_total,
            float prof_pasada,
            HERR_T tipo_herr,
            float avance,
            int   num_aristas,
            float diametro,
            int   num_herr,
           } datos,
           struct {
            float origen_X, origen_Y;
            float dim_mat_X1, dim_mat_Y1;
            float dim_mat_X2, dim_mat_Y2;
            float altura;
            REFR_T refrigerante,
            float velocidad_mat,
           } inicio,
          } u,
          int bloque,
          int num_ent,
          struct nododxf *next, *before, *ESH, *EST,
 },

extern struct nododxf
*headdxf,*taildxf,*Theaddxf,*Ttaildxf;

struct HTA {
   int num;
   char tipo,
   float diam;
   float avance;
   float dientes,
};
extern struct HTA hta[10];

/*  rrrrrraaaaatttttooon*/

extern int bloques[50];
extern int x,y,
extern int bandera_on, raton_on;
extern char delta_x, delta_y,
extern int inc,
extern int comienza_x, comienza_y, fin_x, fin_y,
primer_punto;
extern int mousex, mousey;
extern int OpMenu;
extern int OpSubMenu;
extern char
RutaTotal[MAXPATH],ArchActual[MAXPATH];
extern char trayectoria[2];
/************************************************/
/*                  LIBRERIAS                   */
/************************************************/
#include <stdio.h>
#include <graphics.h>
#include <stdlib.h>
#include <dos.h>
#include <dir.h>
#include <string.h>
#include <conio.h>
#include <math.h>
#include <malloc.h>
#include <ctype.h>
#include <bios.h>

#include "sgc000.h"
#include "sgc100.h"
#include "sgc110.h"
#include "sgc120.h"
#include "sgc130.h"
#include "sgc200.h"
#include "sgc210.h"
#include "sgc220.h"
#include "sgc230.h"
#include "sgc240.h"
#include "sgc250.h"
#include "sgc300.h"
#include "sgc310.h"
#include "sgc320.h"
#include "sgc400.h"
#include "sgc410.h"
#include "sgc420.h"
#include "sgcla.h"
#include "sgclc.h"
#include "sgclg.h"
#include "sgclm.h"
```

```
/*************************************************************/
/*      V A R I A B L E S    G L O B A L E S                 */
/*************************************************************/

int   GraphDriver,              /* The Graphics device
driver          */
int   GraphMode,                /* The Graphics mode
value           */
double AspectRatio,                     /* Aspect ratio
of a pixel on the screen*/
int   MaxX, MaxY,               /* The maximum
resolution of the screen */
int   MaxColors;                /* The maximum # of
colors available    */
int   ErrorCode,                /* Reports any graphics
errors          */
struct palettetype palette,             /* Used to read
palette info        */
char
menu1[4][MAXMENU]={"Archivo","Inicio","Proceso","S
alidas"},

menu2[5][MAXMENU]={"Cargar","sAlvar","Renombrar",
"Borrar","Salir"},

menu3[5][MAXMENU]={"Origen","Altura","Refigerante",
"Material","Definir h "},

menu4[4][MAXMENU]={"Trayectoria","Herramienta","Pr
ofundidad"},

menu5[4][MAXMENU]={"Simulador","Codigo
cn","Imprimir"},
int Nmenu1=4, Nmenu2=5, Nmenu3=5, Nmenu4=3,
Nmenu5=3,
int AreaMouse [30][4],
FILE *fp,       //DXF
int w=0,
//unsigned short int Buffer[1000],
void *BuffWin,
int nivel=0;
int MouseInstalado,
int z=5,RadioH=3,
char OnOff=OFF,
char compensacion='c',
struct nododxf
*headdxf=NULL,*taildxf=NULL,*Theaddxf=NULL,*Ttaild
xf=NULL,

/*  rrrrrraaaaattttooon*/

int x=60, y=60,
int bandera_on=1, raton_bandera_on,
char delta_x, delta_y,
int inc=11,
int comienza_x=0, comienza_y=0, fin_x=0, fin_y=0,
primer_punto=1,
int mousex, mousey;
int OpMenu=0,
int OpSubMenu=0;
char RutaTotal[MAXPATH],ArchActual[MAXPATH],
char trayectoria[2],
int bloques[MAX_BLOQUES],


char heapStr[20],
long heapSize(void),
char mypath[128],    // holds path of program


void getpath() {
 if (_argc>0) {     // _argc holds number of
arguments+1
   char
drive[MAXDRIVE],dir[MAXDIR],file[MAXFILE],ext[MAX
EXT];
   fnsplit(_argv[0], drive, dir, file, ext);
   strcpy(mypath, drive);
   strcat(mypath, dir);
 }
}


/*************************************************************/
/*      P R O G R A M A     P R I N C I P A L
*/
/*************************************************************/

int main()
{ char done=0,
 int i, action,
 unsigned long newMem,oldMem,
 struct actMouseArray_t aMA[10],
 int count=1,
 RutaTotal[0]=0,
 getpath(),
 Initialize(),                  /* Set system into
Graphics mode      */
 pantalla_principal(),
 ini_raton(),
 pon_raton(y*2, x),
 setviewport(0, 0, MaxX, MaxY, 1),
 CoordenadasMouse (aMA, &count);
 oldMem=-2;
 do {
newMem=heapSize(),
   if (newMem!=oldMem ) {
    oldMem=newMem;
    setfillstyle(SOLID_FILL, LIGHTGRAY);
    bar(521, 470, 639, 479);
    setcolor(BLACK);
    outtextxy(521, 470, heapStr);
   }
   action=pulsado(aMA, count, TRUE);
   switch (action) {
   case KEY_PRESSED
    lee_teclado(aMA, &count);
    break,
   case MOUSE_HELP:
    ayuda (1),
    break,
   case MOUSE_ESC
    if (nivel==0) Salida(); else {
          RestoreWindowG(); nivel=0;
    }
    break;
   default:
    lee_raton(action, aMA, &count);
    break;
   }
 } while (!done),
 return(0),
```

```
}

/*****************************************************************/
        control con el raton
/*****************************************************************/

void lee_raton (int action, struct actMouseArray_t
*aMA, int *count) {
  if (nivel>0) RestoreWindowG(),
  if (action<Nmenu1) {
    RecorreMenuP (action),
    AbreMenu(aMA, count),
    nivel=1,
  } else {
    OpSubMenu=action-Nmenu1,
    EjecutaSubMenu(),
    nivel=0,
  }
}

/*****************************************************************/
      control  con el teclado
/*****************************************************************/

int lee_teclado (struct actMouseArray_t *aMA, int
*count)
{int i,j, OpM, Nmenu,
 static int c,
 c=getch(),
 if (c!=ESC)
   switch (nivel) {
     case 0 if (c==0)
              { c=getch(),
                switch (c) {
                      case IZQ  if (OpMenu==0)
OpM=Nmenu1-1,
                                else OpM=OpMenu-1;
                                RecorreMenuP (OpM),
                      break,
                      case DER  if
(OpMenu==Nmenu1-1) OpM=0,
                                else
OpM=OpMenu+1,
                                RecorreMenuP(OpM);
                      break,
                      case ABAJO  AbreMenu (aMA,
count),
                                  nivel=1,
                      break,
                      case F10  ayuda (1),
                      break,
                } // switch (c)
              } // if
              else
              { c=toupper (c),
                switch (c) {
                      case
'A' RecorreMenuP(0),c=RETURN;break;
                      case
'I' RecorreMenuP(1),c=RETURN,break,
                      case
'P' RecorreMenuP(2);c=RETURN,break,
                      case
'S' RecorreMenuP(3);c=RETURN,break;
                }
                if (c==RETURN)
                {       AbreMenu(aMA, count);
                        OpSubMenu=0,
                        nivel=1,
                }
              }
   break, // case 0
   case 1 if (c==0)
              { c=getch(),
                if (c==ARRIBA || c==ABAJO)
                switch (OpMenu) {
                      case 0  RecorreMenus
(Nmenu2,menu2,c); break;
                      case 1: RecorreMenus
(Nmenu3,menu3,c); break;
                      case 2: RecorreMenus
(Nmenu4,menu4,c); break;
                      case 3  RecorreMenus
(Nmenu5,menu5,c); break;
                }
                else
                { if (c==IZQ || c==DER || c==F10)
                        RestoreWindowG();
                      switch (c) {
                       case IZQ  if (OpMenu==0)
OpM=Nmenu1-1,
                                  else OpM=OpMenu-1,
                                  OpSubMenu=0;
                                  RecorreMenuP(OpM);
                      break;
                      case DER  if
(OpMenu==Nmenu1-1) OpM=0,
                                  else
OpM=OpMenu+1,
                                  OpSubMenu=0,
                                  RecorreMenuP(OpM);
                      break,
                      case F10 :
                       switch (OpMenu) {
                              case 0   ayuda (2),
                              break,
                              case 1   ayuda (3),
                              break;
                              case 2.  ayuda (4);
                              break;
                              case 3.  ayuda (5),
                              break;
                       }
                      }
                      if (c==IZQ || c==DER || c==F10)
                       AbreMenu(aMA, count),
                }
              }
              else
              { c=toupper (c),
                switch (OpMenu) {
                      case 0: switch (c) {
                                  case
'C' OpSubMenu=0,c=RETURN,break;
                                  case
'A' OpSubMenu=1;c=RETURN,break;
                                  case
'R' OpSubMenu=2;c=RETURN;break,
                                  case
'B' OpSubMenu=3;c=RETURN;break;
                                  case
'S' OpSubMenu=4;c=RETURN;break,
                                  }
                      break,
```

```
                        case 1.switch (c) {
                                        case
'O'.OpSubMenu=0,c=RETURN,break,
                                        case
'A' OpSubMenu=1,c=RETURN;break;
                                        case
'R' OpSubMenu=2,c=RETURN,break;
                                        case
'M' OpSubMenu=3,c=RETURN,break;
                                        case
'D' OpSubMenu=4,c=RETURN;break;
                                        }
                        break;
                        case 2.switch (c) {
                                        case
'T' OpSubMenu=0,c=RETURN;break;
                                        case
'H'.OpSubMenu=1,c=RETURN;break;
                                        case
'P'.OpSubMenu=2,c=RETURN,break;
                                        }
                        break,
                        case 3 switch (c) {
                                        case
'S' OpSubMenu=0;c=RETURN;break;
                                        case
'C'.OpSubMenu=1;c=RETURN,break,
                                        case
'I' OpSubMenu=2,c=RETURN,break,
                                        }
                        break,
                }
                if (c==RETURN)
                { RestoreWindowG(),
                        EjecutaSubMenu (),
                        nivel=0,
                        OpSubMenu=0,
                }
               }
   break, // case1
} // switch
else  // ESC
switch (nivel) {
        case 0  Salida (),   //fin del programa
        break,
        case 1
        case 2  RestoreWindowG(),
                nivel=0,
        break,
}//switch
return 0,
}


/*************************************************************/
            ejecuta cada menu
/*************************************************************/

void EjecutaSubMenu ()
{ char op
    switch (OpMenu) {
        case 0.switch (OpSubMenu) {
                        case 0: CargaDXF (RutaTotal);
                                GraficaDXF
(headdxf,WHITE);
                        break,
                        case 1:SalvaDXFF1 (RutaTotal),
                        break;
                        case 2:RenombrarArchivo(),
                        break,
                        case 3:BorrarArchivo();
                        break;
                        case 4:Salida();
                        break;
                }
        break; //case 0
        case 1: switch (OpSubMenu) {
                        case 0:Origen ();
                        break;
                        case 1:Altura();
                        break;
                        case 2:Refrigerante();
                        break;
                        case 3:Material();
                        break;
                        case 4:DefinirHer();
                        break;
                }
        break; // case 1
        case 2: switch (OpSubMenu) {
                        case 0:Trayectoria ();
                        break,
                        case 1:Herramienta (),
                        break;
                        case 2 Profundidad (),
                        break;
                        case 3://Compensacion (),
                        break;
                }
        break, // case 2
        case 3: switch (OpSubMenu) {
                        case 0:simular();
                        break;
                        case 1:cn ('n');   // no es para el
simulador
                        break;
                        case 2.imprimir(),
                        break,
                }
        break, // case 5
  } // switch
}


/*************************************************************/
/*              SALIDA DEL SISTEMA                           */
/*************************************************************/

void Salida()
{ char c;
  c=DosOp("Salir del Sistema GC? (S/N): N", NULL,
25);
  if (c=='S')
  { FreeL (headdxf);
    headdxf=taildxf=NULL,
    FreeL (Theaddxf);
    Theaddxf=Ttaildxf=NULL,
    closegraph(),
    exit(0),
  }
}
```

```
void ayuda (int op)
{ char reng[80],
  int i=1,j=0,longi,
  char s[MAXPATH],
  int top=100, left=175,
  WindowG(left, top, 290, 160 , CYAN, WHITE, 10,
   "AYUDA", SAVE_HELP),
  strcpy(s, mypath),
  if (op<200) {
    strcat(s, "sgchelp1 "),
  } else {
    if (op<300)
     strcat(s, "sgchelp2."),
     else strcat(s, "sgchelp3 "),
  }
  fp=fopen(s, "r"),
  if (fp!=NULL) {
    while (!feof(fp) && j<op*14-14) {
      fgets (reng, 35, fp),
      j++,
    }
    setcolor (BLACK),
    fgets(reng, 35, fp),
    setcolor (LIGHTRED),
    for (i=0; i<13 && !feof(fp); i++) {
      fgets(reng,35,fp);
      longi=strlen (reng);
      if (reng[longi-1]=='\n')
            reng[longi-1]='\0',
      if (reng[0]!='\0') {
            outtextxy(left+10, top+15+i*10, reng),
      }
      if (i==1) setcolor (BLACK),
    }
    fclose (fp);
  }
  if (pulsado(NULL, 0, FALSE)==KEY_PRESSED)
getch();
  RestoreWindowG(),
}



/*****************************************************************/
/*     CARGAR ARCHIVO  DXF  -- CNC111 C          */
/*****************************************************************/

#include <stdio h>
#include <dir h>
#include <graphics h>
#include <string h>
#include <math h>
#include <stdlib h>

#include "sgc000 h"
#include "sgc100 h"
#include "sgc101.h"
#include "sgc301 h"
#include "sgcla h"
#include "sgclc h"

struct HTA hta[10],
 /*****************************************************************/
/*       SUBRUTINA QUE DIRIGE LA EXTRACCION
DE ENTIDADES DE "DXF"     */
/*****************************************************************/
```

```
void CargaDXF(char RutaTotal[40]) // DXF(char *arch)
{char aux[10],exten[4],op;
 float g1;
 int g,error=-1,cont=-1,i,
 TYPE s;
 int largo= SIZE_DIR/2*SIZE_CHAR_DIR;
 error=GetFileName
(RutaTotal,170,80,460,largo+150);
  if (error==-1)  // No error
  { for (i=0; i<strlen(RutaTotal) && cont==-1; i++)
     if (RutaTotal[i]==' ') cont=i,
    if (cont==-1) cont++;
    strcpy (exten,RutaTotal+cont+1);
    strupr (exten),
    if (strcmp(exten,"DXF")!=0 &&
strcmp(exten,"F1")!=0)
       mensajes (7),
    else
    { if ((fp=fopen(RutaTotal,"r"))==NULL)  // ABRE EL
ARCHIVO
             mensajes (4),
      else
      { trayectoria[0]='A',
            for (i=0,i<50;i++) bloques[i]=-1;
            strcpy (ArchActual,RutaTotal),
            pausa (1,1),
            do                          // LLEGA A LA
SECCION DE ENTIDADES
            { leeDXF(&g,&g1,&s);
            } while(s!=ENTITIES && g!=EOF),
            if (s==ENTITIES)
            { ini_headdxf ();
              entidades(&error); // SEPARA ENTIDADES
            }
            else
              error=5,
            pausa (1,0);
            if (error!=-1)
            { mensajes (error);
              FreeL(headdxf);
              headdxf=taildxf=NULL;
              FreeL(Theaddxf);
              Theaddxf=Ttaildxf=NULL;
            }
            fclose (fp),
      } //else
   } //else
  }
}



int VerLim (TYPE entidad)
{ int i=0, ang=0,
  float x,y,radi;
  switch (entidad) {
    case LINE   // LINE
              if (taildxf->u line.x1>=0 && taildxf-
>u line x1<=185
              && taildxf->u line x2>=0 && taildxf-
>u.line x2<=185
              && taildxf->u.line y1>=0 && taildxf-
>u.line.y1<=100
              && taildxf->u.line y2>=0 && taildxf-
>u line.y2<=100)
                return (-1),
              else return (11);
```

```
  case CIRCLE: // CIRCLE
            if (taildxf->u circle.h>=0 && taildxf-
>u circle h<=185
            && taildxf->u circle k>=0 && taildxf-
>u.circle k<=100)
              return (-1),
            else return (11);
  case ARC: // ARC
            if (taildxf->u.arc a1>taildxf->u.arc.a2)
              ang=360;
            for (i=taildxf->u.arc.a1; i<=taildxf-
>u.arc.a2+ang; i++)
            { radi=M_PI*i/180;
              x=(cos(radi)*taildxf->u.arc.r+taildxf-
>u.arc.h);
              y=(sin(radi)*taildxf->u.arc.r+taildxf-
>u arc k);
              if (x<0 || x>185
              || y<0 || y>100)
                return (11);
            }
  }
  return (-1);
}


/*****************************************************************/
/*  SUBRUTINA QUE IDENTIFICA ENTIDADES      /
/*                                                               */
/* Datos de entrada : El archivo DXF
/* Datos de salida . ninguno                                     */
/* Por cada entidad se crea un nodo que se a¤ade a la
lista, en el cual  */
/* se almacenan los parametros de la linea, arco o
circulo.          */
/*****************************************************************/

void entidades(int *error)
{ float g1;
  int g,bloq=-1,
  struct nododxf *aux,
  char was_datos;
  char sigue,                  // sigue indica si hay error
  TYPE s;                      // s almacena una cadena
del arch. DXF
  leeDXF (&g,&g1,&s),          // lectura al archivo
  do                         // Recorre seccion de
entidades
  { was_datos=FALSE,
    if (g==0)                  // 0 precede una cadena de
caracteres
    { if (s==INICIO)
    {       DatosInicio ();
            leeDXF(&g,&g1,&s),        // innecesarias
    }
    else
    {if (s==DATOS)
     { if (Theaddxf==NULL)
            { Theaddxf=dup_nodo(headdxf),
              if (Theaddxf==NULL) {
                return;
              }
              Ttaildxf=Theaddxf,
            }
            DatosProceso ();
            bloq++,
            bloques[bloq]=Ttaildxf->bloque;
            leeDXF(&g,&g1,&s);          // innecesarias
            was_datos=TRUE,
    }
    else
    {if (s==LINE)
          { linea(&s,error);     // Llama a linea si la
entidad es "LINE"
            if (*error==-1)
              *error=VerLim (0);
          }
          else
          { if (s==CIRCLE)
            { circulo(&s,error);  // Llama a circulo si la
ent es "CIRCLE"
              *error=VerLim (1);
            }
            else
            { if (s==ARC)
              { arco(&s,error),   // Lama a arco si es la
entidad es "ARC"
                *error=VerLim (2),
              }
              else
                *error=5;
            }
          }                      // else
    }
   }
   if (Theaddxf!=NULL && taildxf->entidad!=INICIO &&
was_datos==FALSE)
   { aux=dup_nodo(taildxf);
     Ttaildxf->next=aux;
     aux->before=Ttaildxf;
     Ttaildxf=aux;
}
  }
 } while (s!=ENDSEC && g!=EOF && *error==-
1);//Finaliza lectura del archivo
}                            // fin de entidades


/*****************************************************************/
/*          ALMACENA PARAMETROS DE LA LINEA
EN LA LISTA          */
/*                                                               */
/* Datos de entrada . Archivo DXF                                */
/* Datos de salida : Lecturas del archivo que finalizan
"LINE" (g y s),  */
/*        indicador de entidades erroneas                        */
/*****************************************************************/

  void linea (TYPE *s,int *error)
  { float g1,
    int g,
    struct nododxf *ptrdxf;      // apuntador auxiliar de
tipo nodo DXF
    ptrdxf = newNodoDXF(LINE, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0); //reserva
    // almacena "LINE" en "entidad" del nodo
    if (ptrdxf==NULL) {
      *error=OUT_OF_MEM;
      return;
    }

    InsDXF (ptrdxf);      // inserta nodo en la lista de
memoria
    do
```

```
    leeDXF(&g,&g1,s),            // lectura del archivo
DXF
   while (g!=10 && g!=0  && g!=EOF),
   if (g==10)
   { ptrdxf->u line x1=g1,       // almacena
coordenada x inicial
     leeDXF(&g,&g1,s),           // lectura del archivo
DXF
     if (g==20)
     { ptrdxf->u line y1=g1,     // almacena
coordenada y inicial
          leeDXF(&g,&g1,s),      // lectura del
archivo DXF
          if (g==30 && g1!=0)
            *error=6,
          else
          { if (g!=11)
              leeDXF(&g,&g1,s),  // lectura del
archivo DXF
            if (g==11)
            { ptrdxf->u line x2=g1;  // almacena
coordenada x inicial
              leeDXF(&g,&g1,s),  // lectura del
archivo DXF
              if (g==21)
              { ptrdxf->u line y2=g1;  // almacena
coordenada y inicial
                leeDXF(&g,&g1,s),  // lectura del
archivo DXF
                if (g==31)
                { leeDXF(&g,&g1,s);  // lectura del
archivo DXF
                    if (g1!=0)
                      *error=6,
                }
              }
              else
                *error=5,
            }
            else
              *error=5,
          }
     }
     else
          *error=5,
   }
   else
     *error=5,
 }                              // fin de linea()



/*************************************************************/
 /*          ALMACENA PARAMETROS DEL CIRCULO
EN LA LISTA          */
 /*                                                          */
 /* Datos de entrada : Archivo DXF                          */
 /* Datos de salida . Lecturas del archivo que finalizan
"CIRCLE" (g y s),*/
 /*          indicador de entidades erroneas
*/
/*************************************************************/

 void circulo (TYPE *s,int *error)
 { float g1,
   int g,
   struct nododxf *ptrdxf,      // apuntador auxiliar de
tipo nodo DXF
   ptrdxf = newNodoDXF(CIRCLE, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0), //reserva
   //almacena "CIRCLE" en "entidad"
   if (ptrdxf==NULL) {
    *error=OUT_OF_MEM;
    return,
   }
   InsDXF (ptrdxf);             // inserta nodo en la lista
de memoria

   do
    leeDXF (&g,&g1,s);
   while (g!=10 && g!=0  && g!=EOF);
   if (g==10)
   { ptrdxf->u.circle.h=g1;      // almacena
coordenada x inicial
     leeDXF(&g,&g1,s);           // lectura del archivo
DXF
     if (g==20)
     { ptrdxf->u.circle k=g1;    // almacena coordenada
y inicial
          leeDXF(&g,&g1,s);      // lectura del
archivo DXF
          if (g==30 && g1!=0)
            *error=6;
          else
          { if (g!=40)
              leeDXF(&g,&g1,s);  // lectura del
archivo DXF
            if (g==40)
            { ptrdxf->u.circle.r=g1;  // almacena
coordenada x inicial
              leeDXF(&g,&g1,s);  // lectura del
archivo DXF
            }
            else
              *error=5,
          }
     }
     else
          *error=5;
   }
   else
    *error=5;
 }


/*************************************************************/
 /*          ALMACENA PARAMETROS DEL ARCO
EN LA LISTA          */
/*************************************************************/

 void arco (TYPE *s,int *error)
 { int g;
   float g1,radi,a1,a2;
   struct nododxf *ptrdxf;      // apuntador auxiliar de
tipo nodo DXF
   ptrdxf = newNodoDXF(ARC, 0, 0, 0, 0, 0, 0, 0, 0, 0,
-1), //reserva
   //almacena "ARC" en "entidad"
   if (ptrdxf==NULL) {
    *error=OUT_OF_MEM,
    return,
   }
```

```
   InsDXF (ptrdxf),            // inserta nodo en la lista
de memoria

   do
   { leeDXF(&g,&g1,s),              // lectura del archivo
DXF
     if (g==111)                  // bandera del sentido
           ptrdxf->bloque=g1;
   } while (g!=10 && g!=0  && g!=EOF);
   if (ptrdxf->bloque==-1) ptrdxf->bloque=0,
   if (g==10)
   { ptrdxf->u arc h=g1;           // almacena
coordenada x inicial
     leeDXF(&g,&g1,s),             // lectura del archivo
DXF
     if (g==20)
     { ptrdxf->u arc.k=g1,         // almacena
coordenada y inicial
           leeDXF(&g,&g1,s),          // lectura del
archivo DXF
           if (g==30 && g1!=0)
             *error=6,
           else
           { if (g!=40)
               leeDXF(&g,&g1,s),      // lectura del
archivo DXF
             if (g==40)
             { ptrdxf->u arc r=g1,
               leeDXF(&g,&g1,s),      // lectura del
archivo DXF
               if (g==50)
               { ptrdxf->u.arc a1=g1,
                 leeDXF(&g,&g1,s),    // lectura del
archivo DXF
                 if (g==51)
                 { ptrdxf->u arc a2=g1,
                       ptrdxf->bloque=0,    // sentido
antihorario del arco
                 }
                 leeDXF(&g,&g1,s),    // lectura del
archivo DXF

               }
               else
                 *error=5,
             }
             else
               *error=5,
           }
     }
     else
           *error=5,
   }
   else
     *error=5;
  if (*error==-1)
  { /*---------- Calculo de los punto inicial y final del arco
----------*/
    float r=ptrdxf->u arc r,
    a1=ptrdxf->u arc.a1,
    a2=ptrdxf->u arc a2,
    if (r<0) {
      r=-r;
      radi = a1 * M_PI / 180,   // convierte a rad los
grados
      ptrdxf->u arc x1 = ptrdxf->u arc.h+cos(radi)*r,  //
calcula x1
      ptrdxf->u.arc y1 = ptrdxf->u.arc.k+sin (radi)*r;
//calcula y1
      radi =  a2 * M_PI / 180; // convierte a rad los
grados
      ptrdxf->u.arc x2 = ptrdxf->u.arc.h+cos(radi)*r,  //
calcula x2
      ptrdxf->u.arc y2 = ptrdxf->u.arc k+sin (radi)*r;
//calcula y2
    } else {
      radi = a1 * M_PI / 180;   // convierte a rad los
grados
      ptrdxf->u.arc x2 = ptrdxf->u.arc.h+cos(radi)*r;  //
calcula x2
      ptrdxf->u arc.y2 = ptrdxf->u.arc.k+sin (radi)*r;
//calcula y2
      radi =  a2 * M_PI / 180, // convierte a rad los
grados
      ptrdxf->u arc.x1 = ptrdxf->u arc.h+cos(radi)*r,  //
calcula x1
      ptrdxf->u arc.y1 = ptrdxf->u arc.k+sin (radi)*r;
//calcula y1
    }
  }
}

 /*************************************************************/
 /*  LEE DEL ARCHIVO EL CODIGO Y SU VALOR
*/ /*                                                          */
 /* Datos de entrada : El archivo DXF                        */
 /* Datos de salida : g  = codigo de la funcion              */
 /*                   g1 = valor flotante                    */
 /*                s  = cadena de caracteres                 */
 /*************************************************************/

 void leeDXF(int *g,float *g1, TYPE *t)
 { char aux[22],                 // almacena una linea del
archivo DXF
   char s[10],
   if (fscanf(fp,"%s",aux)==EOF)
     *g=-1,
   else
   { *g=atoi(aux),
     if (fscanf(fp,"%s",aux)==EOF)
           *g=-1,
     else
     { if (*g<10 || *g==999)          // si aux es una cadena
->
       { if (strcmp(aux,"EOF")==0)
             *g=EOF;
           if (strcmp(aux, "ARC")==0) *t=ARC;
           else if (strcmp(aux,"LINE")==0) *t=LINE;
           else if (strcmp(aux,"CIRCLE")==0)
*t=CIRCLE,
           else if (strcmp(aux,"DATOS")==0)
*t=DATOS,
           else if (strcmp(aux,"INICIO")==0) *t=INICIO,
           else if (strcmp(aux,"ENTITIES")==0)
*t=ENTITIES,
           else if (strcmp(aux,"ENDSEC")==0)
*t=ENDSEC,
           else *t=-1,
       }
       else                        // si aux no es un cadena ->
           *g1=atof(aux),
     }
   }
 }                                 // fin de leeDXF ()
```

```
/*****************************************************************/
/*  INSERTA UN NODO EN LA LISTA DE "DXF"     */
/*                                                               */
/* Datos de entrada   Nodo a insertar                            */
/* Datos de salida   Lista con nodo insertado                    */
 *****************************************************************/

void InsDXF (struct nododxf *ptrdxf)
{
 if (headdxf==NULL)        // si la lista esta vacia ->
 { headdxf=ptrdxf;         // inicializa apuntador cabeza
   ptrdxf->next=NULL,          // inicializa ap. al
siguiente
   ptrdxf->before=NULL,        // inicializa ap  al
anterior
 }
 else                      // si la lista no esta vacia ->
 { ptrdxf->next=NULL,          // inicializa ap. al
siguiente en NULL
   taildxf->next=ptrdxf,       // liga el nodo a la lista
por la cola
   ptrdxf->before=taildxf;     // liga el nodo a la lista
hacia atras
 }
 taildxf=ptrdxf,
 ptrdxf->ESH=NULL,           // ap.de entidades
seguidas por la cabeza
 ptrdxf->EST=NULL,           // ap de entidades
seguidas por la cola
}                          // fin de InsDXF



/*****************************************************************/
/*   SUBRUTINA QUE DA DE BAJA LA LISTA
headdxf-taildxf    */
/*****************************************************************/

void FreeL (struct nododxf *head)
{ if (head!=NULL)
 { while (head->next!=NULL)          // recorre la lista
   { head=head->next,          // incrementa head
     freeNodoDXF (head->before),       // libera aux
              }
   freeNodoDXF (head),
 }
}                          // fin de FreeL



/*****************************************************************/
/*          CARGA ARCHIVOS  F1  --  CNC112 C     */
/*****************************************************************/

#include "sgc000 h"
#include "sgc101 h"
#include "sgcla h"
#include <graphics h>
#include <stdlib h>
#include <string h>

void DatosInicio ()  //int *g, char *s,int *eerronea)
 { float g1,
   char s[20],
   int i,tray,
   fscanf (fp,"%s",s); fscanf(fp,"%s",s), // lee un codigo
del archivo DXF
   headdxf->u inicio.origen_X=atof(s);        //
almacena la coordenada X del origen
   fscanf (fp,"%s",s); fscanf(fp,"%s",s); // lee un codigo
del archivo DXF
   headdxf->u.inicio.origen_Y=atof(s);        //
almacena la coordenada Y del origen
   fscanf (fp,"%s",s), fscanf(fp,"%s",s), // lee un codigo
del archivo DXF
   headdxf->u inicio.dim_mat_X1=atof(s),        //
coord. X1 de las dimensiones del mat.
   fscanf (fp,"%s",s); fscanf(fp,"%s",s), // lee un codigo
del archivo DXF
   headdxf->u.inicio.dim_mat_Y1=atof(s);        //
coord  Y1 de las dimensiones del mat.
   fscanf (fp,"%s",s); fscanf(fp,"%s",s); // lee un codigo
del archivo DXF
   headdxf->u.inicio dim_mat_X2=atof(s);        //
coord. X2 de las dimensiones del mat.
   fscanf (fp,"%s",s); fscanf(fp,"%s",s); // lee un codigo
del archivo DXF
   headdxf->u.inicio.dim_mat_Y2=atof(s),        //
coord  Y2 de las dimensiones del mat.
   fscanf (fp,"%s",s); fscanf(fp,"%s",s); // lee un codigo
del archivo DXF
   headdxf->u inicio.altura=atof(s);         // almacena
la altura de seguridad
   fscanf (fp,"%s",s); fscanf(fp,"%s",s); // lee un codigo
del archivo DXF
   headdxf->u.inicio refrigerante=(REFR_T) atof(s),//
almacena 1 si hay refrigerante, 0 si no
   fscanf (fp,"%s",s); fscanf(fp,"%s",s); // lee un codigo
del archivo DXF
   headdxf->u inicio velocidad_mat=atof(s),        //
almacena la velocidad de corte del mat.
   fscanf (fp,"%s",s);         // lee letrero 'Herramientas"
   memset(hta, 0, sizeof(struct HTA)*10);
   do
   { fscanf (fp,"%s",s),
     if (strcmp(s,"Trayectoria")!=0)
     { i=atoi(s),
            if (i>0) hta[i-1].num=i;
            fscanf (fp,"%s",s);
            if (i>0) if (strcmp(s,"B")==0)
              hta[i-1] tipo='B';
            else
              hta[i-1].tipo='F';
            fscanf (fp,"%s",s);
            if (i>0) hta[i-1].diam=atof(s);
            fscanf (fp,"%s",s),
            if (i>0) hta[i-1].avance=atof(s);
            fscanf (fp,"%s",s);
            if (i>0) hta[i-1] dientes=atof(s);
     }
   } while (strcmp (s,"Trayectoria")!=0);
   fscanf(fp,"%s",s); // lee un codigo del archivo DXF
   tray=atoi(s);
   if (tray==0) trayectoria[0]='M';  // almacena el tipo
de trayectoria
   else trayectoria[0]='A';
   taildxf=headdxf;
 }                              // fin de linea()
```

```
void DatosProceso ()
{   float g1,
    char s[20],
    float
prof_total,prof_pasada,tipo,avance,num_almanca,diam
etro,num_herr,
    int bloque,
    struct nododxf *PDatos,       // apuntador auxiliar de
tipo nodo DXF
    fscanf(fp,"%s",s);  // lee un 0
    fscanf(fp,"%s",s),  // lee un codigo del archivo DXF
    prof_total=atof(s), // almacena la profundidad total
    fscanf(fp,"%s",s),  // lee un codigo del archivo DXF
    prof_pasada=atof(s), // almacena la profundidad por
pasada
    fscanf(fp,"%s",s);  // lee un codigo del archivo DXF
    tipo=atof(s);       // tip de herramienta 1=broca,
2=cortador
    fscanf(fp,"%s",s),  // lee un codigo del archivo DXF
    avance=atof(s),     // almacena el avance
    fscanf(fp,"%s",s),  // lee un codigo del archivo DXF
    num_almanca=atof(s); // almacena el numero de
dientes cortantes
    fscanf(fp,"%s",s),  // lee un codigo del archivo DXF
    diametro=atof(s),   // almacena el diametro de la
herramienta
    fscanf(fp,"%s",s),  // lee un codigo del archivo DXF
    num_herr=atof(s),   // almacena el numero de la
herramienta
    fscanf(fp,"%s",s),  // lee un codigo del archivo DXF
    bloque=atoi(s),     // almacena el numero de bloque
    PDatos =
newNodoDXF(DATOS,0,prof_total,prof_pasada,tipo,

avance,num_almanca,diametro,num_herr,0,bloque),
    if (PDatos==NULL) return,
    Ttaildxf->next=PDatos,
    PDatos->before=TtaildxF,
    TtaildxF=PDatos,
}



/*****************************************************************/
/*          SALVA ARCHIVO PREPROCESADO A
DISCO -- CNC120 C          */
/*****************************************************************/

#include <dir h>
#include <graphics h>
#include <string h>
#include <conio.h>

#include "sgc000 h"
#include "sgc110 h"
#include "sgcla h"
#include "sgclc h"
#include "sgclg h"

/*****************************************************************/
/*      SUBRUTINA QUE DIRIGE LA EXTRACCION
DE ENTIDADES DE "DXF"     */
/*****************************************************************/

 void SalvaDXFF1() // DXF(char *arch)
 {char aux[10],NameAux[MAXPATH];
  struct nododxf *auxdxf;
  int salir,i=0;
  char s[15],
  strcpy (NameAux,ArchActual);
  if (headdxf!=NULL)
  { while (NameAux[i]!='.')
      i++,
    strcpy (NameAux+i+1,"F1");
    WindowG(-1, -1, CapSizeX, CapSizeY, BLUE,
WHITE, 10,
     NO_HEADLINE, SAVE_NORM);
    OutTextWindow (10,8,"Salvar el archivo con el
nombre "),
    WindowG(-1, CapY1+25, CapSizeX-20, CapSizeY-
45, CYAN, WHITE, 0,
     NO_HEADLINE, NO_SAVE);
    setcolor (BLUE);
    OutTextWindow (12,30,NameAux);
    salir=LeeName (NameAux, 12, 30, 120, 30),
    AjustaFileName (NameAux),
    RestoreWindowG();
    if (salir!=ESC)
    { if ((fp=fopen(NameAux,"w"))==NULL)  // ABRE EL
ARCHIVO
            mensajes (4);
      else
      {strcpy (ArchActual,NameAux);
       pausa(2,1);
       if (Theaddxf==NULL)
            auxdxf=headdxf,
       else
            auxdxf=Theaddxf;
           do
       { if (auxdxf->entidad==INICIO)
            { fprintf(fp,"0\nENTITIES\n0\nINICIO\n"),
              fprintf(fp,"OrigenX\n%f\nOrigenY\n%f\n",
               auxdxf->u.inicio.origen_X, auxdxf-
>u.inicio.origen_Y);

fprintf(fp,"DimMatX1\n%f\nDimMatY1\n%f\n",
               auxdxf->u.inicio.dim_mat_X1, auxdxf-
>u.inicio.dim_mat_Y1),

fprintf(fp,"DimMatX2\n%f\nDimMatY2\n%f\n",
               auxdxf->u.inicio.dim_mat_X2, auxdxf-
>u.inicio.dim_mat_Y2),
               fprintf(fp,"Altura\n%f\nRefrigerante\n%d\n",
               auxdxf->u.inicio.altura, auxdxf-
>u.inicio.refrigerante);
               fprintf(fp,"Velocidad_Material\n%f\n",
               auxdxf->u.inicio.velocidad_mat);
               fprintf(fp,"Herramientas\n");
               for (i=0;i<10;i++)
               { if (hta[i].num!=-1)
                 { fprintf (fp,"%d\n",hta[i].num)
                   fprintf (fp,"%c\n",hta[i].tipo);
                   fprintf (fp,"%f\n",hta[i].diam);
                   fprintf (fp,"%f\n",hta[i].avance);
                   fprintf (fp,"%f\n",hta[i].dientes);
                 }
               }
               if (trayectoria[0]=='M')
                 fprintf(fp,"Trayectoria\n0\n"),  // 0 =
trayectoria manual
               else
                 fprintf(fp,"Trayectoria\n1\n"),  // 1 =
trayectoria automatica
```

```
            } else if (auxdxf->entidad==DATOS)
            { fprintf (fp,"0\nDATOS\n0\n"),
              fprintf (fp,"%f\n%f\n%d\n",
               auxdxf->u datos prof_total, auxdxf-
>u datos prof_pasada,
               (int) auxdxf->u datos tipo_herr),
//Prof Total, ProfPasada, Tipo de H (0 o 1)
               fprintf (fp,"%f\n%d\n",
               auxdxf->u datos avance, auxdxf-
>u datos num_aristas), //avace,No de aristas
               fprintf (fp,"%f\n%d\n",
               auxdxf->u datos diametro, auxdxf-
>u datos num_herr); //diam , No.de la H
               fprintf (fp,"%d\n", auxdxf->bloque), //
numero de bloque
            } else if (auxdxf->entidad==LINE)
              fprintf
(fp,"0\nLINE\n10\n%f\n20\n%f\n11\n%f\n21\n%f\n",
               auxdxf->u line x1, auxdxf->u line y1,
               auxdxf->u line x2, auxdxf->u.line y2),
            else if (auxdxf->entidad==ARC)
              fprintf
(fp,"0\nARC\n10\n%f\n20\n%f\n40\n%f\n50\n%f\n51\n
%f\n",
               auxdxf->u arc h, auxdxf->u arc k, auxdxf-
>u.arc r,
               auxdxf->u arc a1, auxdxf->u arc a2),
            else if (auxdxf->entidad==CIRCLE)
              fprintf
(fp,"0\nCIRCLE\n10\n%f\n20\n%f\n40\n%f\n",
               auxdxf->u circle h, auxdxf->u circle k,
auxdxf->u.circle r),
            auxdxf=auxdxf->next,
     } while (auxdxf!=NULL),
     fprintf (fp,"0\nENDSEC\n0\nEOF\n"),
     fclose (fp),
     pausa (2,0),
   } // else
  } // if salir
 }
}

/*************************************************************/
/*  REMOMBRAR UN ARCHIVO - CNC130 C          */
/*************************************************************/

#include <dir h>
#include <graphics h>
#include <string h>

#include "sgc000 h"
#include "sgcla h"
#include "sgclc h"
#include "sgclg h"

/*************************************************************/
/*  ASIGNA EL NUEVO NOMBRE DADO POR EL
USUARIO A UN ARCH  YA EXISTENTE              */
/*                                                           */
/*  Datos de entrada : Nombre del archivo a renombrar*/
/*  Datos de salida   Ninguno                    */
/*************************************************************/

extern int NormLeft, NormTop;
void RenombrarArchivo ()
{ int longitud,
  char aux[13], Ruta[30], Ruta1[30], nombre[15];
  char Nombre[MAXPATH],     // nuevo Nombre-arch.
dado por el usuario
  int c,largo=SIZE_DIR/2*SIZE_CHAR_DIR,
  c=GetFileName (RutaTotal,170,80,460,largo+150),
  if (c==-1)  // No error
  { WindowG(-1, -1, CapSizeX, CapSizeY+30, BLUE,
WHITE, 10,
    "Renombrar el archivo:", SAVE_NORM),
    settextstyle (TRIPLEX_FONT, HORIZ_DIR, 1),
    separa (RutaTotal,Ruta,Nombre);
    setcolor (YELLOW);
    longitud=strlen(Ruta);
    OutTextWindow2(5+longitud*8,30,Nombre),
    setcolor (WHITE);
    OutTextWindow2(5,50,"Nombre:");  // etiqueta que
espera el nuevo nombre
    OutTextWindow2(5,70,"NOTA:La RUTA no debe
modificarse"); // aclaracion
    OutTextWindow2(5,30,Ruta);          // despliega la
ruta
    WindowG (NormLeft+65, NormTop+45, 13*8+10, 15,
CYAN, WHITE,
     0, NO_HEADLINE, NO_SAVE);
    setcolor (BLUE);
    OutTextWindow (70, 49, Nombre);
    c=LeeName(Nombre, 70, 49, 130, 12);
    RestoreWindowG();
    if (c!=ESC)                  // si el usuario no desea
salir ->
    { separa (Nombre,Ruta1,nombre); // separa nueva
ruta de nuevo nombre
      if (strcmp(Ruta,Ruta1)==0 || strcmp(Ruta1,"")==0)
      {       strcat (Ruta, nombre);    // se conserva la
ruta anterior
              if (rename (RutaTotal,Ruta)!=0) // renombra
                mensajes (3),
              else
                strcpy (RutaTotal,Nombre),
      }
      else
              mensajes (3);
    }
  }
}

/*************************************************************/
/*              BORRAR UN ARCHIVO - CNC140 C      */
/*************************************************************/

#include <dir.h>
#include <graphics.h>
#include <string.h>
#include <conio.h>
#include "sgc000 h"
#include "sgc130.h"
#include "sgcla h"
#include "sgclc.h"
#include "sgclg h"

/*************************************************************/
/*       BORRA UN ARCHIVO DEL DISCO SEGUN LA
Ruta DADA POR EL USUARIO   */
/*                                                           */
/*  Datos de entrada : Nombre del archivo a borrar     */
/*  Datos de salida . Ninguno                          */
/*************************************************************/
```

```
void BorrarArchivo ()
{ char c,*p;
  int largo=SIZE_DIR/2*SIZE_CHAR_DIR;
  c=GetFileName (RutaTotal,170,80,460,largo+150);
  p = strchr (RutaTotal,'*'),
  if (c==-1 && p==NULL) //No error
  {
    c=DosOp(RutaTotal, " Borrar? (S/N): N", 9),
    if (c=='S')
      if (remove (RutaTotal)!=0)        // borra del disco el
archivo
            mensajes (3);
  }
}


/*****************************************************************/
/*                  ORIGEN -- CNC210 C                           */
/*                                                               */
/*****************************************************************/

#include <dir h>
#include <conio h>
#include <ctype.h>
#include <string.h>
#include <graphics h>
#include <stdlib.h>
#include <math.h>

#include "sgc000 h"
#include "sgc210 h"
#include "sgcla.h"
#include "sgclc.h"
#include "sgclg h"

/*****************************************************************/
/*                  DESPLAZA EL ORIGEN                           */
/*                                                               */
/*****************************************************************/

void Origen ()
{int c=0;
 if (headdxf!=NULL)
 {WindowG (-1, -1, CapSizeX, CapSizeY+15, BLUE,
WHITE, 10,
  NO_HEADLINE, SAVE_NORM),
  settextstyle (TRIPLEX_FONT, HORIZ_DIR, 1),
  setcolor (WHITE),
  OutTextWindow (5,20,"Punto Final de entidad ( P )"),
  OutTextWindow (5,35,"Coordenadas exactas   ( C )");
  OutTextWindow (5,50,"o Desplazamiento      ( D )");
  setcolor (YELLOW),
  OutTextWindow (150,5,"ORIGEN");
  OutTextWindow (250,50, "D");
  c=getch();
  c=toupper (c);
  while (c!='P' && c!='C' && c!='D' && c!=ESC &&
c!=RETURN)
  { c=getch();
    c=toupper(c),
  }
  RestoreWindowG(),
  if (c!=ESC)
  { if (c=='P')
      PuntoFinalEntidad(),
    else
    { if (headdxf->u inicio origen_X==-1 && headdxf-
>u inicio.origen_Y==-1)
      { headdxf->u.inicio.origen_X=MesaX;
            headdxf-
>u inicio origen_Y=MesaY+AnchoMesa;
      }
      else
      {     GraficaOrigen (headdxf->u.inicio.origen_X,
                           headdxf->u.inicio.origen_Y,
LIGHTGRAY);
      }
      SalvaVentana (headdxf->u.inicio.origen_X-20,
                    headdxf->u.inicio.origen_Y-85,
                    85, 103);
      GraficaOrigen (headdxf->u.inicio.origen_X,
                    headdxf->u.inicio.origen_Y,
WHITE),
      mousex=headdxf->u inicio.origen_X;
mousey=headdxf->u.inicio.origen_Y;
      pon_raton (mousex,mousey);
      MuestraCoordenadas (mousex,mousey);
      if (c==RETURN || c=='D')
            desplazamiento();
      else
      if (c=='C')
      { headdxf-
>u inicio origen_X=LeeCoordenadaExacta('X',500,52,6
50,73);
            headdxf-
>u inicio origen_Y=LeeCoordenadaExacta('Y',580,52,6
35,73),
      }
    }
    if (Theaddxf!=NULL)
    { Theaddxf->u.inicio.origen_X=headdxf-
>u.inicio.origen_X;
      Theaddxf->u inicio.origen_Y=headdxf-
>u inicio origen_Y;
    }
    CierraVentana();
    GraficaDXF (headdxf,WHITE);
  }
 }
 else
   mensajes (1);
 nivel=0,
}

/*****************************************************************/
/*    Selecciona cualquier punto para el origen . con el
mouse, con     */
/*    las flechas o con numeros                                  */
/*****************************************************************/

 void desplazamiento()
 {char c=0,
  do
  { if (MouseInstalado==1)
    { move_raton (&delta_x, &delta_y);   //indica si el
raton ha sido movido
      if (delta_x || delta_y || izq_pulsado() ||
der_pulsado())
      { c=OrigenMouse ();
            MuestraCoordenadas (mousex,mousey);
      }
    }
    if (tecla_pulsada ())
```

```
  { c=OrigenFlechas (),
    MuestraCoordenadas (mousex,mousey),
  }
 } while (c==0),
}

/*****************************************************************/
/*  DESPLAZA EL ORIGEN CON LAS FLECHAS      */
/*****************************************************************/

int OrigenFlechas ()
{ int c;
  char leeX[3],leeY[3],

  c=flechas();
  if (mousey<MesaY || mousey>MesaY+AnchoMesa)
   mousey=headdxf->u.inicio.origen_Y,
  if (mousex<MesaX || mousex>MesaX+LargoMesa)
   mousex=headdxf->u inicio.origen_X;

  if (c==RETURN)
  { GraficaOrigen (headdxf->u.inicio.origen_X,
                   headdxf-
>u.inicio.origen_Y,WHITE);
    return (1),
  }

  if (c==ESC)
  { headdxf->u inicio origen_X=MesaX,
    headdxf->u inicio origen_Y=MesaY+AnchoMesa,
    GraficaOrigen (headdxf->u inicio.origen_X,
                   headdxf-
>u inicio origen_Y,WHITE),
    return (1),
  }

  if (c==0)
  { CierraVentana(),
    headdxf->u inicio origen_X=mousex,
    headdxf->u inicio origen_Y=mousey;
    if (headdxf->u inicio origen_X+65<=639)
      SalvaVentana (headdxf->u inicio origen_X-20,
                    headdxf->u inicio origen_Y-85,
                    85, 103),
    else
      SalvaVentana (headdxf->u inicio origen_X-20,
                    headdxf->u inicio.origen_Y-85,
                    639-headdxf-
>u inicio origen_X+20,
                    103);
    GraficaOrigen (headdxf->u inicio.origen_X,
                   headdxf-
>u.inicio origen_Y,WHITE);
    pon_raton (mousex,mousey);
  }
  if (c==F10)
  { ayuda(12);
  }
 return (0),
}

/*****************************************************************/
/*DESPLAZA EL ORIGEN CON AYUDA DEL
MOUSE*/
/*****************************************************************/

int OrigenMouse ()
{ pos_raton (&mousex,&mousey);
  if (delta_x || delta_y)
  { if (mousey<MesaY || mousey>MesaY+AnchoMesa)
      mousey=headdxf->u inicio.origen_Y,
    if (mousex<MesaX || mousex>MesaX+LargoMesa)
     mousex=headdxf->u.inicio.origen_X;
    RestoreWindowG(),
    headdxf->u.inicio.origen_X=mousex;
    headdxf->u inicio.origen_Y=mousey;
    if (headdxf->u.inicio.origen_X+65<=639)
      SalvaVentana (headdxf->u.inicio.origen_X-20,
                    headdxf->u.inicio.origen_Y-85,
                    85, 103);
    else
      SalvaVentana (headdxf->u.inicio.origen_X-20,
                    headdxf->u.inicio.origen_Y-85,
                    639-headdxf-
>u inicio origen_X+20,
                    103);
    GraficaOrigen (headdxf->u.inicio.origen_X,
                   headdxf-
>u inicio.origen_Y,WHITE);
    return (0),
  }
  if (izq_pulsado ())
  { espera_on (IZQB);
    GraficaOrigen (headdxf->u.inicio.origen_X,
                   headdxf-
>u inicio origen_Y,WHITE);
    return (1),
  }
  if (der_pulsado())
  { espera_on (DERB);
    headdxf->u.inicio.origen_X=MesaX,
    headdxf->u.inicio.origen_Y=MesaY+AnchoMesa,
    GraficaOrigen (headdxf->u.inicio.origen_X,
                   headdxf-
>u inicio origen_Y,WHITE);
    return (1);
  }
  return (0);
}


/*****************************************************************/
/*                  LEE UN NUMERO REAL                 */
/* altura, velocidad del material, profundidad,
Herramienta  */
/*****************************************************************/

extern int NormLeft, NormTop,

float LeeCoordenadaExacta (char xy, int x1, int y1, int
x2, int y2)
{ int i=0,j,BanderaPunto=1,
  float Num,
  int s_left=NormLeft, s_top=NormTop,
  char c,*p,AuxNum[3],leeNum[6],AuxN[6],
  NormLeft=x1; NormTop=y1;
  if (xy=='X')
  { j=(mousex-MesaX)/3;
    itoa (j, leeNum, 10);
    strcat (leeNum,".");
    j=(mousex-MesaX)%3;
    j=j*100/3,
    itoa (j, AuxNum, 10);
    strcat (leeNum,AuxNum);
```

```
}
else
{ j=(MesaY+AnchoMesa-mousey)/3;
  itoa (j, leeNum, 10),
  strcat (leeNum," "),
  j=(MesaY+AnchoMesa-mousey)%3,
  j=j*100/3,
  itoa (j, AuxNum, 10),
  strcat (leeNum,AuxNum),
}
strcpy (AuxN,leeNum),
i=strlen(leeNum),
i--,
p=leeNum,
for (j=0,j<=i,j++)
   p++,
WindowG(x1, y1, x2-x1, y2-y1, BLUE, WHITE, 0,
 NO_HEADLINE, NO_SAVE),
setcolor (YELLOW),
OutTextWindow (3,7,leeNum),
/*---------- Lectura de teclado ----------*/
do {
  c=getch(),
  while (c!=RETURN && c!=ESC)
  { switch (c) {
        case BS if (i>=0)       // borra el ultimo
caracter leido
                    { p--,
                      if (*p=='.') BanderaPunto=0,
                      *p='\x0',
                      WindowG(x1, y1, x2-x1, y2-y1,
BLUE, WHITE, 0,
                       NO_HEADLINE, NO_SAVE),
                      setcolor (YELLOW),
                      OutTextWindow (3,7,leeNum), //
despliega Ruta en pantalla
                      i--,       // decrementa total de
caracteres
                    } break,
    case 0  c=getch(),
            if (c==F10)
            { ayuda(11),
            }
    break,
    default  { if ((isdigit(c) || c==' ' &&
BanderaPunto==0) && i<6)
                    { if (c==' ') BanderaPunto=1,
                      *p=c,// almacena caracter al final
de Ruta
                      p++;
                      *p='\x0',
                      WindowG(x1, y1, x2-x1, y2-y1,
BLUE, WHITE, 0,
                       NO_HEADLINE, NO_SAVE),
                      setcolor (YELLOW);
                      OutTextWindow (3,7,leeNum), //
despliega Ruta en pantalla
                      i++,       // incrementa total de
caracteres
                    }
            }
  }                 // switch (c)
  c=getch();
  }
  if (c==ESC)
  { strcpy (leeNum,AuxN),
    WindowG(x1, y1, x2-x1, y2-y1, BLUE, WHITE, 0,
     NO_HEADLINE, NO_SAVE),
    setcolor (YELLOW);
    OutTextWindow (3,7,leeNum);
   }
   Num=atof(leeNum);
   if (xy=='X' && (Num<0 || Num>185) || xy=='Y' &&
(Num<0 || Num>100))
     mensajes (9);
  } while (xy=='X' && (Num<0 || Num>185) || xy=='Y' &&
(Num<0 || Num>100)),
  if (xy=='X') Num=Num*3+MesaX;
  else
    Num=MesaY+AnchoMesa-3*Num;
  NormLeft=s_left, NormTop=s_top;
  return (Num);
}                       // quiera de estas tres
condiciones



void PuntoFinalEntidad()
{ struct nododxf *pnodo;
  pnodo=SelecEnt (0);
  if (pnodo!=NULL)
    OrigenEnEntidad (pnodo,mousex,mousey);
}



void OrigenEnEntidad (struct nododxf *pnodo, int x, int
y)
{ int distancia1, distancia2;
   distancia1=sqrt(pow(MesaX+pnodo-
>u.line.x1*ESCALA-x,2)+
                    pow(MesaY+AnchoMesa-pnodo-
>u.line.y1*ESCALA-y,2));
   distancia2=sqrt(pow(MesaX+pnodo-
>u.line.x2*ESCALA-x,2)+
                    pow(MesaY+AnchoMesa-pnodo-
>u.line.y2*ESCALA-y,2));
  if (distancia1<=distancia2)
  { headdxf->u.inicio_origen_X=MesaX+pnodo-
>u.line.x1*ESCALA;
    headdxf->u.inicio_origen_Y=
     MesaY+AnchoMesa-pnodo->u.line.y1*ESCALA;
  }
  else
  { headdxf->u.inicio_origen_X=MesaX+pnodo-
>u.line.x2*ESCALA;
    headdxf->u.inicio_origen_Y=
     MesaY+AnchoMesa-pnodo->u.line.y2*ESCALA;
  }
  GraficaOrigen (headdxf->u.inicio_origen_X,
          headdxf->u.inicio_origen_Y,WHITE),
}



/*****************************************************/
/*                ALTURA -- CNC230 C
*/
/*****************************************************/

#include <dir.h>
#include <graphics.h>
#include <stdlib.h>
#include <string.h>
```

```
#include <conio h>
#include <ctype h>

#include "sgc000 h"
#include "sgc220 h"
#include "sgclc h"
#include "sgclg h"

/*****************************************************************/
/*          CAPTURA ALTURA Y QUEDA
ALMACENADA EN headdxf->r        */
/*****************************************************************/

void Altura()
{ char altura[15];
  if (headdxf!=NULL)
  { WindowG (-1, -1, CapSizeX, CapSizeY, BLUE,
WHITE, 10,
     NO_HEADLINE, SAVE_NORM);
    settextstyle (TRIPLEX_FONT, HORIZ_DIR, 1),
    setcolor (WHITE),
    OutTextWindow (10, 10, "Altura de seguridad (en
mm)  "),
    Real_Cadena (headdxf->u inicio altura, altura),
    LeeReal(altura, 20, 25, 220),
    headdxf->u.inicio altura=atof(altura),
    RestoreWindowG(),
  }
  else
    mensajes (1),
}



/*****************************************************************/
/*                REFRIGERANTE - CNC230 C
*/
/*****************************************************************/

#include <dir h>
#include <graphics h>
#include <string h>
#include <conio h>

#include "sgc000 h"
#include "sgc230 h"
#include "sgclc h"
#include "sgclg h"

/*****************************************************************/
/*        CAPTURA REFRIGERANTE Y QUEDA
ALMACENADO EN              */
/*          headdxf->a1 como 0 o 1  (0=No, 1=Si)     */
/*****************************************************************/

void Refrigerante()
{ char c, s[40],
  if (headdxf!=NULL)
  { strcpy(s, "Refrigerante (S/N)  "),
    if (headdxf-
>u inicio refrigerante==REFRIGERANTE_NO) strcat(s,
"N"),
     else strcat(s, "S"),
    c=DosOp(s, NULL, 14),
    if (c=='S')
    { headdxf-
>u inicio refrigerante=REFRIGERANTE_SI;
      if (Theaddxf!=NULL)
            Theaddxf-
>u inicio refrigerante=REFRIGERANTE_SI;
    }
    else if (c!=ESC)
    { headdxf-
>u inicio.refrigerante=REFRIGERANTE_NO;
      if (Theaddxf!=NULL)
            Theaddxf-
>u inicio refrigerante=REFRIGERANTE_NO;
    }
  }
  else
    mensajes (1);
}


/*****************************************************************/
/*                MATERIAL - CNC240.C
*/
/*****************************************************************/

#include <dir.h>
#include <graphics h>
#include <stdlib h>
#include <string h>

#include "sgc000 h"
#include "sgc240 h"
#include "sgc100 h"
#include "sgclc h"
#include "sgclg.h"

/*****************************************************************/
/*          CAPTURA MATERIAL Y QUEDA
ALMACENADA EN headdxf->a2       */
/*****************************************************************/

void Material()
{ char material[15],num[10],
  float n,
  int op,
  if (headdxf==NULL)
     ini_headdxf (),
  op=DosOp("Dimensiones del material o", "Velocidad
de corte (D/V): D", 25),
  if (op!=ESC)
  { WindowG (-1, -1, CapSizeX, CapSizeY, BLUE,
WHITE, 10,
     NO_HEADLINE, SAVE_NORM);
    settextstyle (TRIPLEX_FONT, HORIZ_DIR, 1),
    setcolor (WHITE),
    if (op=='D' || op==RETURN)
    { OutTextWindow (10,5,"Esquina inferior izquierda .
"),
      OutTextWindow (10,20,"X="),
      do
      { Real_Cadena (headdxf-
>u inicio dim_mat_X1,num),
            op=LeeReal (num, 35, 20, 0),
            n=atof(num),
      }while (n>185 && op!=ESC),
      if (op!=ESC)
      { headdxf->u.inicio dim_mat_X1=n,
            setcolor (WHITE),
            OutTextWindow (130,20,"Y=");
            do
```

```
          { Real_Cadena (headdxf-
>u inicio dim_mat_Y1,num),
            op=LeeReal(num, 155, 20, 0),
            n=atof(num),
           } while (n>100 && op!=ESC);
           if (op!=ESC)
           { headdxf->u inicio dim_mat_Y1=n,
            setcolor (WHITE);
            OutTextWindow (10,35,"Esquina superior
derecha  "),
            OutTextWindow (10,50,"X="),
            do
            { Real_Cadena (headdxf-
>u.inicio.dim_mat_X2,num),
             op=LeeReal(num, 35, 50, 0);
             n=atof(num),
            } while (n>185 && op!=ESC);
            if (op!=ESC)
            { headdxf->u inicio dim_mat_X2=n,
             setcolor (WHITE),
             OutTextWindow (130,50,"Y="),
             do

             { Real_Cadena (headdxf-
>u inicio dim_mat_Y2,num),
              op=LeeReal(num, 155, 50, 0),
              n=atof(num),
             } while (n>100 && op!=ESC),
             if (op!=ESC)
              headdxf->u inicio dim_mat_Y2=n,
            }
           }
    }
    if (Theaddxf!=NULL)
    { Theaddxf->u inicio dim_mat_X1=headdxf-
>u inicio dim_mat_X1,
          Theaddxf->u inicio dim_mat_Y1=headdxf-
>u inicio dim_mat_Y1,
          Theaddxf->u inicio dim_mat_X2=headdxf-
>u.inicio dim_mat_X2,
          Theaddxf->u inicio dim_mat_Y2=headdxf-
>u inicio dim_mat_Y2,
    }
    setcolor (YELLOW),              // traza el material
    GraficaDXF (headdxf,WHITE),
   }
   else
   { OutTextWindow (10,10,"Velocidad de corte del
material  "),
    OutTextWindow (10,35,"( en mm/seg )  "),
    Real_Cadena (headdxf-
>u inicio velocidad_mat,material),
    LeeReal(material, 135, 35, 240),
    headdxf->u inicio velocidad_mat=atof(material),
    if (Theaddxf!=NULL)
          Theaddxf->u.inicio velocidad_mat=headdxf-
>u inicio velocidad_mat,
   }
   RestoreWindowG(),
  }
}


/*****************************************************/
/*          DEFINICION DE HERRAMIENTAS      */
/*****************************************************/
```

```
#include <dir h>
#include <graphics h>
#include <string.h>
#include <stdlib.h>
#include <conio h>
#include <ctype.h>

#include "sgc000 h"
#include "sgc250.h"
#include "sgclc.h"
#include 'sgclg.h"

/*****************************************************/
/* CAPTURA HERRAMIENTA Y QUEDA
ALMACENADA EN  */
/*                                                   */
/* El numero de la herramienta      PDatos->a1     */
/* El tipo 0=Broca, 1=fresa en      PDatos->y2     */
/* El diametro en                   PDatos->r      */
/* El avance en                     PDatos->h      */
/* El numero de aristas cortante en  PDatos->k (solo
para fresa)  */
/*****************************************************/



void DefinirHer ()
{ int NumHer,
  char op,
    do
    { NumHer=MenuHer(NULL),
      if (NumHer!=ESC)
      {       if (hta[NumHer-1].num!=0)
              { op=DosOp("Modificar o Eliminar herram ",
"(M/E)  M", 25),
               if (op==RETURN) op='M';
               if (op=='E')
               { hta[NumHer-1].num=0;
                 hta[NumHer-1].diam=0;
                 hta[NumHer-1] avance=0;
                 hta[NumHer-1] dientes=0;
               }
              }
              else op='M',
              if (op=='M') LeeHerramienta (NumHer),
      }
    } while (NumHer!=ESC),
}


int LeeHerramienta (int NumHer)
{ char c,herram[5];
  char s[40],
  int salir;
  strcpy(s, "Broca o Fresa (B/F)· ");
  if (hta[NumHer-1] tipo=='B') strcat(s, "B"),
   else if (hta[NumHer-1] tipo=='F') strcat(s, "F"),
   else strcat(s, " ");
  c=DosOp(s, NULL, 19),
  if (c != ESC) {
    hta[NumHer-1].tipo=c;
    if (c=='B') broca(NumHer),
     else fresa(NumHer);
  }
  return (c),
}
```

```
/*****************************************************************/
/*    CAPTURA ELEMENTOS PARA LA BROCA                            */
/*                                                               */
/*  El tipo 0=Broca,              PDatos->y2                     */
/*  El diametro en                PDatos->r                      */
/*  El avance en                  PDatos->h                      */
/*****************************************************************/

int broca (int NumHer)
{ char broca[15],
  int salir;
  struct nododxf *PDatos;
  WindowG(-1, -1, CapSizeX, CapSizeY, BLUE,
WHITE, 10,
   NO_HEADLINE, SAVE_NORM);
  settextstyle (TRIPLEX_FONT, HORIZ_DIR, 1),
  setcolor (WHITE),
  OutTextWindow (10,8,"Diametro de la broca (en mm)
  "),
  Real_Cadena (hta[NumHer-1] diam,broca),
  salir=LeeReal(broca, 20, 25, 330),
  if (salir!=ESC)
  { hta[NumHer-1] diam=atof(broca);
    settextstyle (TRIPLEX_FONT, HORIZ_DIR, 1),
    setcolor (WHITE),
    OutTextWindow (10,8,"Avance de la broca (mm/seg)
  "),
    Real_Cadena (hta[NumHer-1] avance,broca),
    salir=LeeReal(broca, 20, 25, 331),
    if (salir!=ESC)
    { hta[NumHer-1] avance=atof(broca),
      hta[NumHer-1] num=NumHer,
    }
  }
  RestoreWindowG(),
  return (salir),
}


/*****************************************************************/
*  CAPTURA ELEMENTOS PARA EL CORTADOR                           */
/*                                                               */
/*  El tipo 1=fresa en      PDatos->y2                           */
/*  El diametro en                PDatos->r                      */
/*  El avance por diente del fresa   PDatos->h
/*  El numero de aristas cortante en   PDatos->k  (solo
para fresa)   */
/*****************************************************************
*

int fresa (int NumHer)
{ char fresa[15],
  struct nododxf *PDatos,
  int salir,
  WindowG (-1, -1, CapSizeX, CapSizeY, BLUE,
WHITE, 10,
   NO_HEADLINE, SAVE_NORM);
  settextstyle (TRIPLEX_FONT, HORIZ_DIR, 1),
  setcolor (WHITE),
  OutTextWindow (10,8,"Diametro del fresa (en mm)  ");
  Real_Cadena (hta[NumHer-1] diam,fresa);
  salir=LeeReal(fresa, 20, 25, 332),
  if (salir!=ESC)
  { hta[NumHer-1] diam=atof(fresa),
    setcolor (WHITE),
    OutTextWindow (10,8,"Avance en mm/diente del
fresa  "),
    Real_Cadena (hta[NumHer-1].avance,fresa),
    salir=LeeReal(fresa, 20, 25, 333);
    if (salir!=ESC)
    { hta[NumHer-1].avance=atof(fresa);
      setcolor (WHITE);
      OutTextWindow (10,8,"Numero de aristas cortantes
."),
      Real_Cadena (hta[NumHer-1].dientes,fresa),
      salir=LeeReal(fresa, 20, 25, 334);
      if (salir!=ESC)
       { hta[NumHer-1] dientes=atof(fresa);
            hta[NumHer-1] num=NumHer;
       }
    }
  }
  RestoreWindowG(),
  return (salir),
}

extern NormLeft, NormTop,
#define OFS_Y_TABLE 16

int MenuHer (char *s)
{ int i,
  int largo=SIZE_DIR/2*SIZE_CHAR_DIR,
  char cadena[15];
  struct actMouseArray_t aMA[10],
WindowG (-1, -1, 290, largo+70, CYAN, WHITE, 5,
   "No.TIPO DIAMETRO AVANCE DIENTES",
SAVE_NORM),
  if (s!=NULL) {
    setcolor (YELLOW);
    OutTextWindow (15, largo+70-15, s),
  }
  settextstyle (TRIPLEX_FONT, HORIZ_DIR, 1),
  setcolor (BLUE);
  for (i=0; i<10; i++)
  { itoa (i+1, cadena, 10);
    if (i==9) OutTextWindow
(2,i*11+OFS_Y_TABLE,cadena);
    else OutTextWindow
(9,i*11+OFS_Y_TABLE,cadena);
    aMA[i] x1=NormLeft+2;
    aMA[i] y1=NormTop+i*11+OFS_Y_TABLE;
    aMA[i] x2=NormLeft+290-4;
    aMA[i].y2=NormTop+i*11+11+OFS_Y_TABLE,
    if (hta[i] num!=0)
    { if (hta[i].tipo=='B')
      { OutTextWindow
(25,i*11+OFS_Y_TABLE,"BROCA");
            Real_Cadena (hta[i].diam,cadena),
            OutTextWindow
(80,i*11+OFS_Y_TABLE,cadena),
            Real_Cadena (hta[i].avance,cadena);
            OutTextWindow
(160,i*11+OFS_Y_TABLE,cadena),
      }
      else
      { OutTextWindow
(25,i*11+OFS_Y_TABLE,"FRESA");
            Real_Cadena (hta[i] diam,cadena);
            OutTextWindow
(80,i*11+OFS_Y_TABLE,cadena),
            Real_Cadena (hta[i].avance,cadena);
```

```
            OutTextWindow
(160,i*11+OFS_Y_TABLE,cadena);
            Real_Cadena (hta[i] dientes,cadena),
            OutTextWindow
(250,i*11+OFS_Y_TABLE,cadena),
      }
    }
  }
  setcolor (BLACK),
  OutTextWindow (10,130,"Dar un n£mero del 1 al 10
:");
  do
  { i=LeeNum (230, 130, aMA, 10, 234);
  } while ((i<1 || i>10) && i!=ESC);
  RestoreWindowG(),
  return (i);
}


/*****************************************************************/
/*                TRAYECTORIA  - CNC340 C                        */
/*****************************************************************/

#include <dir h>
#include <graphics h>

#include "sgc000 h"
#include "sgc100.h"
#include "sgc300 h"
#include "sgc301.h"
#include "sgc302 h"
#include "sgclg h"
#include "sgclc h"

void Trayectoria()
{ char op='s',
  int bloque,i=0,
  struct nododxf *nx,
  if (headdxf!=NULL)
  { if (trayectoria[0]=='M')            // si esta activa
trayectoria manual
      op=DosOp("Continuar con la selecci¢n manual",
                        "ya inicializada (S/N). S", 25),
    if (trayectoria[0]=='A' && op!=ESC || op=='N') //
trayectora automatica
    { trayectoria[0]='A',
      op=DosOp("Trayectoria Manual o Automatica",
"(M/A)  M", 25),
      if (op!=ESC)
      { for (i=0, i<MAX_BLOQUES, i++) bloques[i]=-1,
            nx=headdxf,
            while (nx!=NULL) {
              nx->bloque=-1,
              nx=nx->next,
            }
            FreeL (Theaddxf),
            Theaddxf=Taildxf=NULL,
            GraficaDXF (headdxf,WHITE),
      }
    }
    if (op!=ESC)
    { if (op=='A')              // se selecciono trayectoria
automatica
      { trayectoria[0]='A',
            TrayecAutomatica (),
            return,
      }
      do
      {     bloque=LeeBloq (),
            if (bloque!=ESC)
            { trayectoria[0]='M',
              op='N',                   // N= Nuevo bloque
              for (i=0; i<MAX_BLOQUES && bloques[i]!=-
1 && bloques[i]!=bloque; i++);
              if (bloques[i]==bloque) op='O'; // O= (old)
Bloque ya activado
              if (op=='O')
              { op=DosOp("Agregar entidades al bloque
o",
                              "eliminarlo (A/E): A", 25);
              }
              if (op=='E')
              { BorraBloque (bloque), // Se elimina el
bloque
                GraficaDXF (headdxf,WHITE);
              }
              if (op=='N' || op=='A') op=TrayecManual
(bloque), // Selecciona ent.
            }
        } while (op!=ESC && bloque!=ESC);
      }
    }
    else
      mensajes (1);
//  if (trayectoria[0]=='M')
//    GraficaDXF (headdxf,WHITE);
/*  setfillstyle (SOLID_FILL,LIGHTGRAY); // define
estilo de lineas
    bar ( 0, 450, MaxX, MaxY),    // dibuja una barra
ancha como fondo
    setcolor (BLACK),
    line (1,455,640,455),
    line (1,450,640,450);
    OutTextWindow (1,460,"Archivo en uso : "),
    OutTextWindow (150,460,ArchActual);
    OutTextWindow (510,460,"F10 -> Ayuda"), */
}



#include <graphics.h>
#include <string.h>
#include <conio.h>
#include <stdlib h>
#include <ctype.h>
#include <math h>

#include "sgc000.h"
#include "sgc301.h"
#include "sgc302.h"
#include "sgcla.h"
#include "sgclc.h"
#include "sgclg h"

int TrayecManual (int bloque)
{ int i=-1,j=-1, auxil;
  int found,
  struct nododxf *aux,*pnodo,*xn;

if (Theaddxf==NULL)
 { Theaddxf=dup_nodo(headdxf);
   if (Theaddxf==NULL) {
     return(-1);
   }
```

```
  bloques[0]=bloque,
  aux=CreaDatos (Theaddxf),
  if (aux==NULL) {
   freeNodoDXF(Theaddxf),
   Theaddxf=NULL;
   return(-1);
  }
  aux->bloque=bloque;
 }
 else
 { aux=Theaddxf,
   while (bloques[i+1]!=-1 && bloques[i+1]<=bloque)
i++,

   if (i!=-1)
   { while (j<=i && aux->next!=NULL)
     { if (j<i) aux=aux->next;
            else if (aux->entidad!=DATOS)
              aux=aux->next,
            if (aux->entidad==DATOS) j++,
     }
     while (aux->next!=NULL && aux->next-
>entidad!=DATOS)
            aux=aux->next,
   }
   if (i==-1 || bloques[i]!=bloque)
   { i++,
     aux=CreaDatos (aux),
     if (aux==NULL) {
            return(-1),
     }
     j=0,
     while (bloques[j]!=-1) j++,
     if (bloques[j-1]>bloque)
            while (j>i)
            { bloques [j]=bloques[j-1],
              j--,
            }
     bloques[j]=bloque,
     aux->bloque=bloque,
   }
  }
  found=0,
  xn=headdxf,
  while (xn!=NULL) {
   if (xn->bloque!=bloque) xn->num_ent=-1, else {
    xn->num_ent=0,found++,
   }
   xn=xn->next,
  }
  do
  { char good=FALSE,
    pnodo=SelecEnt (found),
    if (pnodo!=NULL && pnodo->num_ent==-1) {
     float dis,
     if (aux->entidad==DATOS) good=TRUE;
          else if (aux->entidad==CIRCLE && pnodo-
>entidad==CIRCLE) good=TRUE;
          else if (pnodo->entidad!=CIRCLE) {
           dis=DistanciaPuntos(pnodo->u line x1, aux-
>u line x2,
                                pnodo->u line.y1,
aux->u line.y2);
           if (dis<DIS) good=TRUE; else {
            dis=DistanciaPuntos(pnodo->u line x2,
aux->u.line x2,
                                pnodo->u line y2,
aux->u line y2),
            if (dis<DIS) {
             float h,
             good=TRUE;
             h=pnodo->u.line x1;
             pnodo->u.line.x1=pnodo->u.line.x2;
             pnodo->u.line.x2=h;
             h=pnodo->u.line y1;
             pnodo->u.line.y1=pnodo->u.line.y2;
             pnodo->u line.y2=h;
             if (pnodo->entidad==ARC)
              pnodo->u.arc.r=-pnodo->u.arc r;
            }

           }
     }
    }
    if (good==TRUE)
    { pnodo->num_ent=0,
      pnodo->bloque=bloque;
      found++,
      pnodo=dup_nodo(pnodo),
      if (aux->next!=NULL)
      { pnodo->next=aux->next;
            pnodo->next->before=pnodo,
      }
      pnodo->before=aux,
      aux->next=pnodo,
      if (aux->entidad==DATOS)
            if (pnodo->entidad==CIRCLE) aux-
>u datos tipo_herr=HERR_BROCA;
            else aux-
>u datos tipo_herr=HERR_COTADOR,
      aux=pnodo;
      if (bloque==8) setcolor (LIGHTGREEN);
      else setcolor (bloque),
      if (pnodo->entidad==LINE)
            line (MesaX+pnodo->u.line x1*ESCALA,
                  MesaY+AnchoMesa-pnodo-
>u line y1*ESCALA,
                  MesaX+pnodo->u.line.x2*ESCALA,
                  MesaY+AnchoMesa-pnodo-
>u line y2*ESCALA),
           else if (pnodo->entidad==CIRCLE)
            circle (MesaX+pnodo->u.circle.h*ESCALA,
                  MesaY+AnchoMesa-pnodo-
>u circle k*ESCALA,
                  pnodo->u.circle.r*ESCALA),
           else if (pnodo->entidad==ARC) {
            float r=pnodo->u.arc.r;
            if (r<0) r=-r,
            arc (MesaX+pnodo->u.arc.h*ESCALA,
                  MesaY+AnchoMesa-pnodo-
>u arc k*ESCALA,
                  pnodo->u arc.a1, pnodo->u.arc.a2,
r*ESCALA),
      }
      if (found==1) aux=Sentido (aux,mousex,mousey),
      if (bloque%16==8) TrazaSentido
(aux,LIGHTGREEN);
      else TrazaSentido (aux,bloque%16),
     }
   } while (pnodo!=NULL);
  write_status_line(),
  return(0);
}
```

```
struct nododxf *CreaDatos (struct nododxf *aux)
 { struct nododxf *datos = newNodoDXF(DATOS,
    1,   // compensacion
   -5,   // profundidad total
   -5,   // profundidad por pasada
    0,   // tipo de herramienta 0=broca, 1=cortador
   0 004, // el avance de la herramienta
    4,   // el numero de dientes del cortador
    3,   // diametro de la herramienta
    1,   // numero de la herramienta
    0, 0),
  if (datos==NULL) return(datos),
  if (aux->next!=NULL)
   { datos->next=aux->next;
     datos->next->before=datos;
   }
   else
          datos->next=NULL,
   datos->before=aux,
   aux->next=datos,
   aux=datos,
   return (aux),
 }


void BorraBloque (int bloque)
 { int i=0,
   struct nododxf *aux,*aux1,

   aux=Theaddxf->next,
   while (bloques[i]!=bloque)
   { if (aux->next->entidad==DATOS) i++,
     aux=aux->next,
   }
   aux1=aux->before,
   while (aux->next!=NULL && aux->next-
>entidad!=DATOS)
   { aux1->next=aux->next,
     freeNodoDXF (aux),
     aux=aux1->next,
   }
   if (aux->next!=NULL)
   { aux1->next=aux->next,
     aux1->next->before=aux1,
   }
   else aux1->next=NULL,
   freeNodoDXF (aux),

   aux=headdxf->next,
   while (aux!=NULL)
   { if (aux->bloque==bloque)
          aux->bloque=-1,
     aux=aux->next,
   }
   while (bloques[i+1]!=-1)
   { bloques[i]=bloques[i+1],
     i++,
   }
   bloques[i]=-1,
}


struct nododxf *Sentido (struct nododxf *pnodo, int x,
int y)
{ int distancia1, distancia2,
  float auxiliar,
   distancia1=sqrt(pow(MesaX+pnodo-
>u line x1*ESCALA-x,2)+
                   pow(MesaY+AnchoMesa-pnodo-
>u line y1*ESCALA-y,2)),
   distancia2=sqrt(pow(MesaX+pnodo-
>u line x2*ESCALA-x,2)+
                   pow(MesaY+AnchoMesa-pnodo-
>u line y2*ESCALA-y,2));
  if (distancia1>distancia2)
  { auxiliar=pnodo->u.line.x1;
    pnodo->u.line.x1=pnodo->u.line.x2;
    pnodo->u line.x2=auxiliar;
    auxiliar=pnodo->u.line.y1;
    pnodo->u line y1=pnodo->u.line.y2;
    pnodo->u line.y2=auxiliar;
    if (pnodo->entidad==ARC)
     pnodo->u arc r=-pnodo->u arc r,
//     auxiliar=pnodo->u arc a1,      // el -1 solo es una
bandera para indicar que
//     pnodo->u.arc.r=-pnodo->u arc.r; // el sentido del
arco va encontra del reloj
//     pnodo->u.arc.a2=auxiliar,  // los datos se
encuentran almacenados correc-
                                    // tamente pero
                                    // 1) hay que
invertir los angulos, y los
                                    //    puntos
inicial y final en el momento de
                                    //    dibujarlos
en pantalla
                                    // 2) volverlos
positivos en la conversion a CN
  }
  return (pnodo),
 }



#include <dir.h>
#include <graphics.h>
#include <string.h>
#include <math.h>

#include "sgc000.h"
#include "sgc100.h"
#include "sgc301.h"
#include "sgc302.h"
#include "sgcla.h"
#include "sgclc.h"
#include "sgclg.h"

struct nododxf
*headList1=NULL,*tailList1=NULL,*headList2=NULL,

          *tailList2=NULL,*headList3=NULL,*tailList3=
NULL,
                    *headList4=NULL,*tailList4=NULL,

struct nododxf *dupHeadL1=NULL, *dupTailL1=NULL,
*headaux=NULL;
int IntercambioCont=0;
struct nododxf *HeadBroca, *TailBroca,
int heart_break,
```

```
void TrayecAutomatica ()
{ int op='N',
  struct nododxf *xn,
  int i,

Theaddxf=TtaildxF=dupHeadL1=dupTailL1=headaux=H
eadBroca=TailBroca=NULL,
  i=0,
  xn=headdxf,
  while (xn!=NULL) {
    xn->num_ent=i++,
    xn=xn->next;
  }
  op=DosOp("Aumentar barrenos en caso", "necesario
? (S/N)· N", 25);
  if (op==ESC) return;
  SeparaBarrenos (),
  if (Theaddxf!=NULL)
  {
    pausa (5,1),
    recursion (Theaddxt,'t',1),
    if (tailList2->bloque>=tailList3->bloque &&
    tailList2->bloque>=tailList4->bloque ||
headList3==NULL)
    { TailBroca->next=headList2;
      headList2->before=TailBroca,
      headList2=tailList2=NULL,
    }
    else
      if (tailList3->bloque>=tailList4->bloque ||
headList4==NULL)
      {       TailBroca->next=headList3,
              headList3->before=TailBroca,
              headList3=tailList3=NULL;
      }
      else
      {       TailBroca->next=headList4,
              headList4->before=TailBroca,
              headList4=tailList4=NULL,
      }
  }
  FreeL (Theaddxf),
  Theaddxf=HeadBroca,
  HeadBroca=TailBroca=NULL,
  TtaildxF=TailBroca,
  if (op=='S')
     AumentaBarrenos (),
  pausa (5,0),
  GraficaDXF (Theaddxf,BLACK),
  FreeL (headList1),
  FreeL (headList2),
  FreeL (headList3),
  FreeL (headList4),

headList1=headList2=headList3=tailList1=tailList2=tailL
ist4=NULL,
}


/*****************************************************************/
/*      Esta Subrutina separa los barrenos (CIRCLE) de
las lineas y    */
/*      los arcos. Genera dos listas .
*/
/*a) HeadBroca, TailBroca que es la lista donde se
separa:      */
/* INICIO, DATOS (de barrenos), CIRCLE (barrenos) y
DATOS de   */
/* ARCs y LINEs                                    */
/* 2a) Theaddxf, TtaildxF que es la lista donde se
agrupan solo   */
/* los LINEs y ARCs para poder ser ordenados
posteriormente.  */
/*****************************************************************/

void SeparaBarrenos ()
{ struct nododxf *aux,*new;
  HeadBroca=dup_nodo(headdxf);
  TailBroca=HeadBroca;
  aux=headdxf->next;
  while (aux!=NULL)
  { if (aux->entidad==CIRCLE)
    { new=dup_nodo(aux),
      TailBroca->next=new,
      new->before=TailBroca,
      TailBroca=new;
    }
    else
    { if (Theaddxf==NULL)
      {       Theaddxf=dup_nodo(aux),
              TtaildxF=Theaddxf;
      }
      else
      { new=dup_nodo(aux);
              TtaildxF->next=new;
              new->before=TtaildxF,
              TtaildxF=new;
      }
    }
    aux=aux->next;
  }
  if (TailBroca->entidad==CIRCLE)
  { aux = newNodoDXF(DATOS,
     1,  // compensacion
    -5,  // profundidad total
    -5,  // profundidad por pasada
     0,  // tipo de herramienta 0=broca, 1=cortador
     004, // el avance de la herramienta
     4,  // el numero de dientes del cortador
     3,  // diametro de la herramienta
     1,  // numero de la herramienta
     0,
     1), // numero de bloque
    if (aux==NULL) return,
    aux->next=HeadBroca->next,
    aux->next->before=aux,
    HeadBroca->next=aux,
    aux->before=HeadBroca,
  }
  if (Theaddxf!=NULL)
  { aux = newNodoDXF(DATOS,
     1,  // compensacion
    -5,  // profundidad total
    -5,  // profundidad por pasada
     1,  // tipo de herramienta 0=broca, 1=cortador
     004, // el avance de la herramienta
     4,  // el numero de dientes del cortador
     3,  // diametro de la herramienta
     2,  // numero de la herramienta
     0,
     2),  // numero de bloque
    if (aux==NULL) return;
    TailBroca->next=aux;
```

```
   aux->before=TailBroca,
   TailBroca=aux,
  }
}

extern int NormLeft, NormTop,

void recursion (struct nododxf *nodo, char headtail, int
bloque)
{ struct nododxf *ESeguidas=NULL,
  struct nododxf *FaltaEnt,*aux,
  char b[3],
  InsertaRecursion (nodo, headtail, bloque);
  setfillstyle(SOLID_FILL, LIGHTGRAY);
  bar(NormLeft+258, NormTop+11, NormLeft+268,
NormTop+19),
  setcolor(BLACK);
  switch (++heart_break%4) {
    case 0: OutTextWindow2(260, 12, "-"), break,
    case 1  OutTextWindow2(260, 12, "\\"), break,
    case 2. OutTextWindow2(260, 12, "I"), break,
    case 3  OutTextWindow2(260, 12, "/"); break,
  }
  FaltaEnt=FaltaE (Theaddxf),
  if (FaltaEnt != NULL)
  { EntidadesSeguidasCola(),
    if (tailList1->EST !=NULL)
    { ESeguidas=tailList1->EST,
      while (ESeguidas!=NULL)
      { nodo = dup_nodo (ESeguidas),
            nodo->bloque=bloque,
            recursion (nodo,'t', bloque),
//          if (IntercambioCont==5)
//             return 0,
            tailList1=tailList1->before,
            freeNodoDXF (tailList1->next),
            tailList1->next=NULL,
            ESeguidas=ESeguidas->EST,
      }
      ESeguidas=tailList1->EST,
      FreeL (ESeguidas),
      ESeguidas=NULL,  // ojo error
      tailList1->EST=NULL,
    }
    else
    { EntidadesSeguidasCabeza (),
      if (headaux->ESH !=NULL)
      { ESeguidas=headaux->ESH,
          while (ESeguidas!=NULL)
          { nodo = dup_nodo (ESeguidas),
            nodo->bloque=bloque;
            recursion (nodo,'c', bloque),
//          if (IntercambioCont==5)
//             return 0,
            if (headaux==headList1)
            { headaux=headaux->next;
              headList1=headaux;
              freeNodoDXF (headaux->before),
              headaux->before=NULL;
            }
            else
            { if (headaux->before!=NULL)
              { headaux=headaux->next,
                headaux->before=headaux->before-
>before,
                freeNodoDXF (headaux->before->next);
                headaux->before->next=headaux,
              }
              else
              { freeNodoDXF (headaux);
                headaux=headList1;
                headaux->before=NULL;
              }
            }
            ESeguidas=ESeguidas->ESH;
          }
          ESeguidas=headaux->ESH;
          FreeL (ESeguidas);
          ESeguidas=NULL;  // ojo ******* error
          headaux->ESH=NULL;
      }
      else
      { nodo=FaltaE (Theaddxf),
          if (nodo !=NULL)
          { bloque++,
            headaux=nodo;
            recursion(nodo,'t', bloque),
//          if (IntercambioCont==5)
//             return 0,
            if ( headaux!=headList1  &&
headaux==tailList1 )
              relocaliza_headaux ();
            tailList1=tailList1->before,
            freeNodoDXF (tailList1->next);
            tailList1->next=NULL,
          }
      }
    }
  }
  else
   if (IntercambioCont<3)
   { IntercambiaListaMenor();
     IntercambioCont++;
   }
//  return 0,
}


void relocaliza_headaux (void)
{ int bloq,
  headaux=headaux->before,
  bloq = headaux->bloque,
  while (bloq==headaux->before->bloque)
        headaux=headaux->before;
}


void IntercambiaListaMenor (void)
{
 if (tailList1->bloque < tailList2->bloque ||
tailList2==NULL)
 { FreeL (headList2);
   FreeL (headList3);
   FreeL (headList4);
   headList2=headList3=headList4=NULL;
   duplicaL1 (),
   headList2=dupHeadL1,
   tailList2=dupTailL1;
   dupHeadL1=dupTailL1=NULL,
 }
 else
 { if (tailList2!=NULL && tailList1->bloque==tailList2-
>bloque)
   { if (headList3==NULL)
```

```
    { duplicaL1 (),
          headList3=dupHeadL1;
          tailList3=dupTailL1,
          dupHeadL1=dupTailL1=NULL,
    }
    else
    { if (headList4==NULL)
          { duplicaL1 (),
            headList4=dupHeadL1,
            tailList4=dupTailL1,
            dupHeadL1=dupTailL1=NULL;
          }
    }
   }
  }
}

void EntidadesSeguidasCola (void)
{ struct nododxf *auxdxf,*ESeguidas;
  float aux,dis;
  auxdxf = Theaddxf,

  ESeguidas=tailList1,
  while (auxdxf != NULL)
  { auxdxf=FaltaE (auxdxf),
    if (auxdxf != NULL)
    { dis=DistanciaPuntos(tailList1->u line x2,auxdxf-
>u line.x1,
                                tailList1-
>u line.y2,auxdxf->u line.y1),
     if (dis<=DIS)
     { ESeguidas->EST=dup_nodo (auxdxf),
          ESeguidas=ESeguidas->EST;
     }
     dis=DistanciaPuntos(tailList1->u line x2,auxdxf-
>u.line x2,
                                tailList1-
>u line.y2,auxdxf->u line y2),
     if (dis<=DIS)
     { ESeguidas->EST=dup_nodo(auxdxf),
          ESeguidas=ESeguidas->EST;
          aux=ESeguidas->u line x1;
          ESeguidas->u line x1=ESeguidas->u.line x2,
          ESeguidas->u line x2=aux;
          aux=ESeguidas->u line y1,
          ESeguidas->u line y1=ESeguidas->u.line y2;
          ESeguidas->u line.y2=aux,
          ESeguidas->u.arc r=-ESeguidas->u arc r,
     }
     auxdxf=auxdxf->next;
    }
  }
}

void EntidadesSeguidasCabeza (void)
{ struct nododxf *auxdxf,*ESeguidas;
  float aux,dis,
  auxdxf = Theaddxf;

  ESeguidas=headaux,
  while (auxdxf != NULL)
  { auxdxf=FaltaE (auxdxf);
    if (auxdxf != NULL)
    { dis=DistanciaPuntos(headaux->u line x1,auxdxf-
>u line.x2,
                                headaux-
>u.line y1,auxdxf->u line y2),
     if (dis<=DIS)
     { ESeguidas->ESH=dup_nodo (auxdxf);
          ESeguidas=ESeguidas->ESH;
     }
     dis=DistanciaPuntos(headaux->u.line.x1,auxdxf-
>u line x1,
                                headaux-
>u.line y1,auxdxf->u.line.y1),
     if (dis<=DIS)
     { ESeguidas->ESH=dup_nodo(auxdxf);
          ESeguidas=ESeguidas->ESH;
          aux=ESeguidas->u.arc.x1;
          ESeguidas->u.arc.x1=ESeguidas->u arc.x2;
          ESeguidas->u.arc.x2=aux;
          aux=ESeguidas->u.arc.y1;
          ESeguidas->u.arc.y1=ESeguidas->u.arc.y2;
          ESeguidas->u.arc.y2=aux;
          ESeguidas->u.arc.r=-ESeguidas->u.arc.r,
     }
     auxdxf=auxdxf->next;
    }
  }
}



void InsertaRecursion (struct nododxf *nodo_x, char
headtail, int bloque)
{ struct nododxf *aux;
  aux = dup_nodo (nodo_x);
  aux->bloque=bloque,

  if (headaux==NULL)
  { headList1=aux;
    headaux=aux;
    tailList1=aux,
  }
  else
  { if (headtail=='t')
    { tailList1->next=aux;
      aux->before=tailList1;
      if (aux->bloque!=tailList1->bloque)
            headaux=aux;
      tailList1=aux;
    }
    else
    { if (headList1==headaux)
      { headList1->before=aux,
            aux->next=headList1;
            headList1=aux,
      }
      else
      { headaux->before->next=aux,
            aux->next=headaux;
            aux->before=headaux->before,
            headaux->before=aux;
      }
      headaux=aux;
    }
  }
}

struct nododxf *FaltaE (struct nododxf *auxdxf)
{ struct nododxf *auxL1;
  char found,
```

```
  while (auxdxf!=NULL) {
   auxL1=headList1,
   found=FALSE,
   while (auxL1!=NULL && found==FALSE) {
    if (auxL1->num_ent==auxdxf->num_ent)
found=TRUE,
     auxL1=auxL1->next;
   }
   if (found==FALSE) return (auxdxf),
   auxdxf=auxdxf->next,
  }
  return (NULL);
}

struct nododxf *oldFaltaE (struct nododxf *auxdxf)
{ struct nododxf *auxL1,
  char iguales,

  while (auxdxf!=NULL)
  { auxL1=headList1,
   iguales=FALSE;
   while (auxL1!=NULL && iguales==FALSE)
   {
    char equ=FALSE;
    if (auxdxf->entidad==ARC) {
          if (auxdxf->u arc r>0) {
           if (auxL1->u.arc.r>0) {
            if (auxdxf->u.arc r==auxL1->u arc r)
equ=TRUE,
           } else {
            if (auxdxf->u.arc r==-auxL1->u arc r)
equ=TRUE,
           }
          } else {
           if (auxL1->u arc r>0) {
            if (-auxdxf->u arc r==auxL1->u arc r)
equ=TRUE,
           } else {
            if (auxdxf->u arc r==auxL1->u arc.r)
equ=TRUE,
           }
          }
    }
    if ( auxdxf->entidad==LINE &&
          (auxdxf->u.line x1==auxL1->u line x1 &&
           auxdxf->u line y1==auxL1->u line y1 &&
           auxdxf->u line x2==auxL1->u line x2 &&
           auxdxf->u line y2==auxL1->u line y2 ||
           auxdxf->u line x1==auxL1->u line x2 &&
           auxdxf->u line y1==auxL1->u.line y2 &&
           auxdxf->u line x2==auxL1->u line x1 &&
           auxdxf->u line y2==auxL1->u line y1
          )
                                ||
          auxdxf->entidad==CIRCLE &&
          auxdxf->u.circle h==auxL1->u.circle h &&
          auxdxf->u circle.k==auxL1->u circle.k &&
          auxdxf->u circle r==auxL1->u circle r
                                ||
          auxdxf->entidad==ARC &&
          auxdxf->u arc h==auxL1->u arc h &&
          auxdxf->u arc k==auxL1->u.arc.k &&
          equ==TRUE &&
          auxdxf->u.arc.a1==auxL1->u arc a1 &&
          auxdxf->u.arc.a2==auxL1->u.arc a2
         )
        iguales=TRUE,
     auxL1=auxL1->next;
    }
    if (iguales==FALSE)
        return (auxdxf);
   auxdxf=auxdxf->next;
  }
  return (NULL),
}


void AumentaBarrenos ()
{ struct nododxf *barren=NULL, *aux, *aux1;
  float dis;
  if (TtaildxfF->entidad!=CIRCLE)
  { aux=TheaddxfF->next->next;
   if (aux->entidad==CIRCLE)
   { barren=TheaddxfF->next;
    while (aux->entidad!=DATOS)
        aux=aux->next,
   }
   else
    aux=TheaddxfF->next;
   while (aux!=NULL)
   { aux=aux->next,
    if (aux!=NULL)
    { if (barren==NULL)
        { barren=newNodoDXF(DATOS,
          1,  // compensacion
          -5,  // profundidad total
          -5,  // profundidad por pasada
          0,  // tipo de herramienta 0=broca,
1=cortador
          .004, // el avance de la herramienta
          4,  // el numero de dientes del cortador
          3,  // diametro de la herramienta
          1,  // numero de la herramienta
          0,
          1);
         if (barren==NULL) return,
         barren->next=TheaddxfF->next;
         barren->next->before=barren,
         TheaddxfF->next=barren;
         barren->before=TheaddxfF;
        }
        aux1=barren->next,
        dis=10000,
        while (aux1->entidad!=DATOS &&
dis==10000)
        { if (aux1->entidad==CIRCLE)
          dis=DistanciaPuntos(aux1->u.circle.h, aux-
>u line x1,
                                aux1-
>u.circle k, aux->u line.y1);
          else if (aux1->entidad==ARC)
           dis=DistanciaPuntos(aux1->u arc h, aux-
>u line x1,
                                aux1->u arc k,
aux->u line y1),
          else dis=100000,
         if (dis>=DIS)
          dis=10000,
         aux1=aux1->next,
        }
        if (dis==10000)
        { aux1=newNodoDXF(CIRCLE,0,0,0,0, aux-
>u.line x1,
                        aux->u line y1,2,0,0,-1),
```

```
            if (aux1==NULL) return;
            aux1->next=barren->next,
            barren->next->before=aux1;
            barren->next=aux1,
            aux1->before=barren;
           }
           while (aux->bloque==aux->next->bloque &&
aux->next!=NULL)
            aux=aux->next,
    }
   }
  }
}

/*****************************************************************/
/*              HERRAMIENTA  - CNC330.C                          */
/*****************************************************************/

#include <dir h>
#include <graphics.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>

#include "sgc000 h"
#include "sgc250.h"
#include "sgc310.h"
#include "sgclc h"
#include "sgclg h"

/*****************************************************************/
/*        CAPTURA HERRAMIENTA Y QUEDA
ALMACENADA EN                 */
/*                                                               */
/* El numero de la herramienta        PDatos->a1      */
/* El tipo 0=Broca, 1=Cortador en     PDatos->y2      */
/* El diametro en                     PDatos->r       */
/* El avance en                       PDatos->h       */
/* El numero de aristas cortante en   PDatos->k  (solo
para cortador)  */
/*****************************************************************/



void Herramienta ()
{ struct nododxf *aux,
  int i,j,bloque,salir=ESC,
  if (Theaddxf!=NULL)
  { aux=Theaddxf->next,
    if (trayectoria [0]=='A')
    { if (aux!=NULL && aux->next!=NULL && aux->next-
>entidad==CIRCLE)
     {      salir=MenuHer("ESCOGER UN NUMERO
DE BROCA"),
            if (salir!=ESC)
            { aux->u datos num_herr=salir;
              aux->u datos tipo_herr=hta[salir-1] tipo,
              aux->u datos avance=hta[salir-1].avance;
              aux->u datos num_aristas=hta[salir-
1] dientes;
              aux->u datos diametro=hta[salir-1] diam,
              do
              { aux=aux->next,
              } while (aux!=NULL && aux-
>entidad!=DATOS),
            }
     }
    if (aux!=NULL)
    { salir=MenuHer("ESCOGER UN NUMERO DE
FRESA"),
            if (salir!=ESC)
            { aux->u.datos.num_herr=salir;
              aux->u datos.tipo_herr=hta[salir-1].tipo;
              aux->u.datos avance=hta[salir-1].avance,
              aux->u.datos num_aristas=hta[salir-
1] dientes,
              aux->u.datos.diametro=hta[salir-1].diam,
            }
    }
   }
   else
     do
     { do
            { bloque=LeeBloq (),
              if (bloque!=ESC)
               for (i=0; i<MAX_BLOQUES &&
bloques[i]!=bloque && bloques[i]!=-1, i++);
            } while (bloque!=ESC &&
(i>MAX_BLOQUES || bloques[i]==-1)),
            if (bloque!=ESC)
            { aux=Theaddxf->next;
              j=-1;
              while (j<i && aux!=NULL)
              { if (aux->entidad==DATOS) j++;
                if (j<i) aux=aux->next,
              }
              if (aux!=NULL)
              {  if (aux->next->entidad==CIRCLE)
                      salir=MenuHer("ESCOGER UN
NUMERO DE BROCA"),
                 else
                      salir=MenuHer("ESCOGER UN
NUMERO DE FRESA"),
                 if (salir!=ESC)
                 {      aux->u datos num_herr=salir;
                        aux->u.datos.tipo_herr=hta[salir-
1] tipo,
                        aux->u.datos avance=hta[salir-
1] avance,
                        aux-
>u datos num_aristas=hta[salir-1].dientes;
                        aux->u.datos diametro=hta[salir-
1] diam,
                 }
               }
              }
     } while (bloque!=ESC);
 }
 else
   if (headdxf==NULL)
     mensajes(1);
   else
     mensajes (2); //aux==NULL
}

/*****************************************************************
/*              PROFUNDIDAD - CNC320.C                           */
/*****************************************************************

#include <dir h>
#include <graphics h>
#include <stdlib.h>
```

```
#include "sgc000 h"
#include "sgc320 h"
#include "sgclc h"
#include "sgclg h"

/*************************************************************/
CAPTURA PROFUNDIDAD TOTAL Y PROFUNDIDAD
POR PASADA          */
/*    Y QUEDAN ALMACENADAS EN  P.Total=PDatos-
>y1, P.Pasada=PDatos->x2    */
/*************************************************************/


void Profundidad ()
{ struct nododxf *aux,
  int i,j,bloque,salir,
  if (Theaddxf!=NULL)
  { aux=Theaddxf->next,
    if (trayectoria [0]=='A')
    { if (aux!=NULL)
            salir=LeeProfundidad (aux),
      if (salir!=ESC)
      { do
            { aux=aux->next,
            } while (aux!=NULL && aux-
>entidad!=DATOS),
            if (aux!=NULL)
              LeeProfundidad (aux),
      }
    }
    else
      do
      { do
            { bloque=LeeBloq (),
              if (bloque!=ESC)
               for (i=0; i<MAX_BLOQUES &&
bloques[i]!=bloque && bloques[i]!=-1, i++),
            } while (bloque!=ESC &&
(i>MAX_BLOQUES || bloques[i]==-1)),
            if (bloque!=ESC)
            { aux=Theaddxf->next,
              j=-1,
              while (j<i && aux!=NULL)
              { if (aux->entidad==DATOS) j++,
                if (j<i) aux=aux->next,
              }
              if (aux!=NULL)
                salir=LeeProfundidad (aux),
            }
      } while (bloque!=ESC)
   }
   else
    if (headdxf==NULL)
      mensajes(1),
    else
      mensajes (2), //aux==NULL
}



int LeeProfundidad(struct nododxf *PDatos)
{ char profun[15],
  float ProfunP,ProfunT,
  int salir=0,
  long h,
  if (headdxf!=NULL)
  { do
    { WindowG(-1, -1, CapSizeX, CapSizeY, BLUE,
WHITE, 10,
       NO_HEADLINE, SAVE_NORM),
      settextstyle (TRIPLEX_FONT, HORIZ_DIR, 1),
      setcolor (WHITE),
      if (PDatos->u datos tipo_herr==HERR_BROCA)
       OutTextWindow (10,10,"Profundidad TOTAL
BROCA (mm) "),
       else OutTextWindow (10,10,"Profundidad TOTAL
CORTADOR (mm)."),
      Real_Cadena (PDatos->u datos prof_total,profun),
      salir=LeeReal(profun, 15, 27, 320),
      ProfunT=atof(profun),
      h=(long) (ProfunT*1000),
      if (h>0) {
            if (ProfunT*1000-h>=0 5) h++,
      } else if (h-ProfunT*1000>=0.5) h--,
      ProfunT=(float) h/1000,
      RestoreWindowG(),
      if (salir!=ESC)
      { if (PDatos->u.datos.tipo_herr==HERR_BROCA)
            { PDatos->u.datos prof_total=ProfunT,
              ProfunT=-10,     // estas asignaciones no
influyen para el usuari,
              ProfunP=-2;      // solo se usan para salir
del ciclo while
            }
            else
            { WindowG(-1, -1, CapSizeX, CapSizeY,
BLUE, WHITE, 10,
               NO_HEADLINE, SAVE_NORM),
              setcolor (WHITE),
              OutTextWindow (10, 10, "Prof. por pasada
CORTADOR (mm).");
              if (PDatos->u datos.prof_pasada<ProfunT)
               PDatos->u datos prof_pasada=ProfunT,
              Real_Cadena (PDatos-
>u datos prof_pasada,profun);
              salir=LeeReal(profun, 15, 27, 321),
              RestoreWindowG(),
              if (salir!=ESC)
              { ProfunP=atof(profun),
                h=(long) (ProfunP*1000);
                if (h>0) {
                  if (ProfunP*1000-h>=0.5) h++,
                } else if (h-ProfunP*1000>=0.5) h--,
                ProfunP=(float) h/1000,
                if (ProfunT <0 && ProfunP<0 &&
ProfunT<ProfunP ||
                      ProfunP==ProfunT || ProfunT>0)
                { PDatos->u datos prof_total=ProfunT,
                  PDatos->u datos prof_pasada=ProfunP,
                }
                else
                      mensajes (13);       // error en
datos de profundidad
              }
            }
      }
    } while (salir!=ESC && ProfunT<0 && ProfunP<0 &&
ProfunT > ProfunP
    && ProfunP!=ProfunT);
  }
  return (salir),
}
```