UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO E IMPLEMENTACION DE UN SISTEMA DE CONTROL PARA TORNOS AUTOMATICOS BASADO EN LA PC-AT

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

MA. MERCEDES BOLAÑOS CASTELLANOS

DIRECTOR DE TESIS: ING. MIGUEL ANGEL CRUZ LEON

CIUDAD UNIVERSITARIA. 1998.

AGRADECIMIENTOS

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL PARA TORNOS AUTOMÁTICOS BASADO EN LA PC - AT.

ÍNDICE

ÍNDICE

INTRODUCCIÓN

INTRODUCCIÓN

Dentro del área de máquinas y herramientas, la automatización juega un papel *muy importante*, sobre todo para la producción en serie, pero generalmente las máquinas con algún elemento de control automático incrementan sus costos enormemente.

Los países que se dedican a la fabricación de maquinaria la ofrecen a costos que muchas empresas no pueden pagar y aunque a veces puedan *adquirir esos equipos el costo del* mantenimiento y actualizaciones (u otros aditamentos) implican que pronto dejen de funcionar o se vuelvan obsoletos.

Otras empresas compran maquinaria usada, que les falta alguna parte o totalmente descompuesta (como desecho y a precios de "chatarra") para repararla, adaptarle algún control y trabajar con ella.

El presente trabajo se desarrolló con este último fin, el torno que se desea controlar llegó a la empresa incompleto, sin funcionar, y después de hacerle algunos ajustes (mecánicos y neumáticos para lograr sus movimientos manualmente) se deseaba lograr la automatización de él.

Los medios disponibles para elaborar este control de automatización son muy variados, se pueden usar microcontroladores, *PLC's (Programmable Logic Controllers) o incluso* electrónica digital discreta (como flip-flops, compuertas, etc.) pero el método que aquí se plantea intenta disminuir el costo y aumentar la eficiencia de los sistemas mencionados anteriormente, en los cuales la capacidad de memoria es limitada y el costo y complejidad de su elaboración puede ser elevado.

La computadora, como elemento para el control, *resulta una herramienta muy práctica* y útil, así pués la integraremos junto con las interfaces adecuadas para lograr controlar la operación del torno.

Para lograr el desarrollo planteado, primeramente (en el capítulo 1) se describen los elementos necesarios para entender el funcionamiento de todas las partes que integrarán el sistema de control elegido: *el torno, la computadora, las interfaces y el programa de* control.

El capítulo 2 describe, en base al torno analizado, el desarrollo de las interfaces necesarias; en el capítulo 3 se plantea el desarrollo del programa de control, los algoritmos de control y la interfaz con el usuario.

En el capítulo 4 se mencionan las pruebas parciales e integrales que se fueron desarrollando hasta lograr nuestro objetivo.

El capítulo 5 contiene resultados y conclusiones del trabajo elaborado y finalmente se anexan, en los apéndices, el manual de usuario y código fuente del programa de control además de los circuitos (diagramas completos) de las interfaces desarrolladas.

El trabajo realizado, además de incrementar la productividad y reducir costos, desea ser sencillo, fácil de usar y amigable a los operadores de este tipo de maquinaria.

CAPÍTULO 1

ANTECEDENTES

CAPÍTULO 1 - ANTECEDENTES

1.1.- OPERACIÓN DE TORNOS AUTOMÁTICOS

DEFINICIÓN

Un torno es una máquina - herramienta que puede realizar algunos sólidos de revolución.

Los hay de operación manual y automática. En éstos últimos se prepara un conjunto de herramientas y un programa para cada pieza que se desee crear, éste se puede ejecutar automáticamente por un sistema de control. Si el programa se puede ejecutar ciclicamente podemos obtener varias piezas sin interrumpir (detener) la operación de la máquina.

Aparte de sistemas mecánicos la operación de este tipo de máquinas-herramientas está dada por sistemas eléctricos, hidraúlicos, neumáticos o la combinación de varios.

1.1.1- MOVIMIENTOS PARA EL DESPLAZAMIENTO

Este tipo de máquinas puede tener más de un eje para realizar desplazamiento en los planos longitudinal, transversal y/o vertical, estos planos los designamos como los ejes coordenados X, Y y Z. Sobre cada uno de estos ejes se montan distintas herramientas que al desplazarse generan el desbaste y/o corte en el material de trabajo. Generalmente el eje Z lo relacionamos con la herramienta principal que tendrá desplazamientos sobre el eje principal de la máquina.

Sobre los ejes anteriores pueden tener lugar movimientos rotatorios (para alguna herramienta), generando ejes rotatorios A, B y C (ver fig. 1.1).

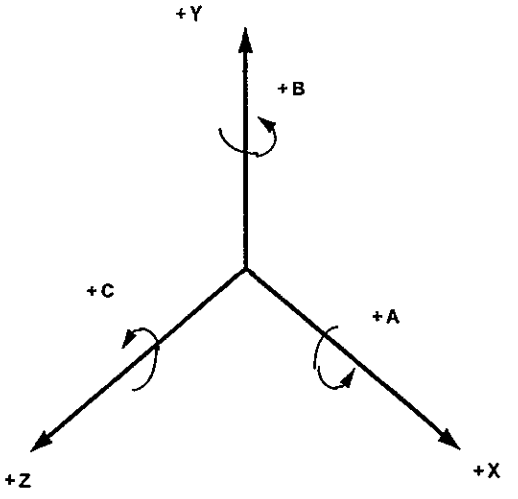


FIG.1.1 - MOVIMIENTOS ROTATORIOS

El rango o la velocidad a la cual ocurren los desplazamientos se expresa en pies o metros por minuto, o en pulgadas o milímetros por revolución del eje de la máquina. Esta velocidad depende del tipo de material a cortar y la herramienta utilizada.

Todos los movimientos están limitados por la operación de microinterruptores o sensores de proximidad, para evitar dañar partes de la máquina durante los desplazamientos que se realizan.

1.1.2- TIPOS DE AVANCE

Los desplazamientos en los distintos ejes pueden ser proporcionados por diferentes tipos de avance como son:

- El avance hidraúlico se puede generar por medio de la actuación de cilindros hidraúlicos.
- El avance neumático se puede dar por medio de la actuación de cilindros neumáticos (ver fig. 1.2).

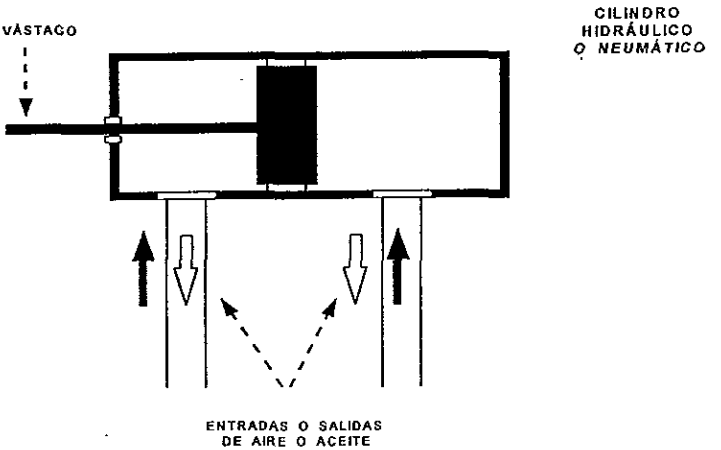


FIG.1.2 - CILINDRO HIDRÁULICO O NEUMÁTICO

- El avance mecánico puede ser más exacto que los otros tipos de avance ya que cada paso que dá es de una longitud determinada, y generalmente es proporcionada por la operación de una flecha de avance sobre un tornillo sin fin.

1.1.3.- EJE CENTRAL DE LA MÁQUINA

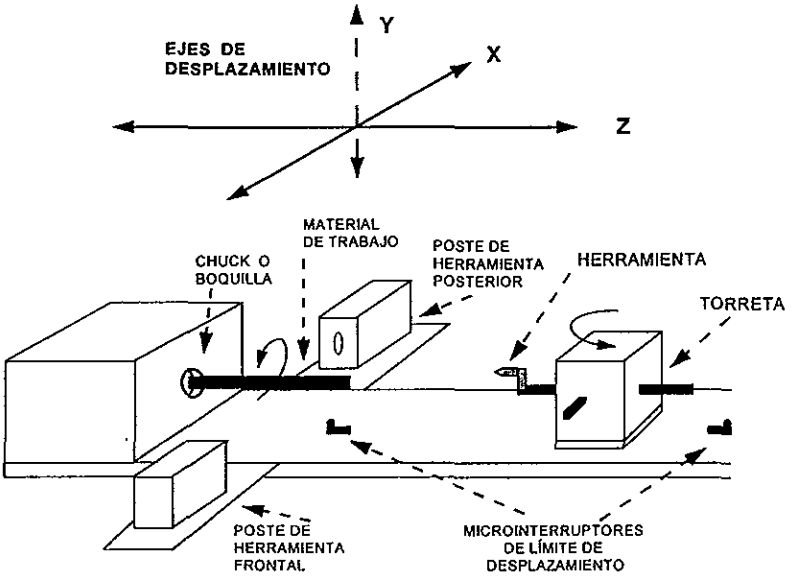
El eje central de la máquina es manejado, directa o indirectamente por motores eléctricos. Un control sobre éste generalmente incluye paro, comienzo, dirección y velocidad de la rotación. Esta velocidad depende del tipo de material a cortar y la herramienta utilizada, los fabricantes de herramientas de corte publican velocidades recomendadas dependiendo de los materiales usados.

COLOCACIÓN Y AVANCE DEL MATERIAL DE TRABAJO

El material de trabajo (p.e. cilindro metálico, o de madera, dependiendo del tipo de torno) se introduce en este eje central que lo sostendrá y hará girar. El sentido y velocidad del giro pueden variar para adecuarse a la herramienta que se esté utilizando en ese momento.

El material se introduce y un mandril (chuck) lo presiona para sujetarlo antes de que comience a girar. Al comenzar a girar se procede a ir generando ya sea manual o automáticamente los demás movimientos para ir dando forma a la pieza.

El avance del material de trabajo puede ser mecánicamente, con la intervención manual de un operador, o por algún sistema hidraúlico o neumático.



ESTRUCTURA DEL TORNO

FIG.1.3 - ESTRUCTURA DEL TORNO

1.1.4.-HERRAMIENTAS DE LA TORRETA Y ADITAMENTOS

Sobre el riel del movimiento en el eje Z se desliza la torreta (parte del torno en la que se montan las herramientas, ver fig. 1.3) la cual puede soportar varias herramientas a la vez y girar en un sentido (indexado) para permitir el cambio de herramienta. Al acercarse a la parte frontal, la torreta, se aproxima al material de trabajo realizando el maquinado y al retraerse a la parte posterior se aleja realizando el cambio de herramienta.

Sobre el riel del eje X se pueden desplazar dos herramientas, la frontal y la posterior, cada una de éstas puede acercarse o retirarse del centro (donde está ubicado el material).

En el eje Y también puede ir alguna herramienta o dispositivo para corte u otra función.

Existe una gran variedad de herramientas y se van diseñando nuevas conforme a las necesidades de la pieza a elaborar (dependiendo de la forma de ellas es el tipo de corte que generan). Son de distintos materiales, dependiendo el tipo de material a cortar, pero las más resistentes contienen una parte o son totalmente de carburo (ver fig. 1.4).

Los aditamentos son dispositivos que también se acoplan al torno (por ejemplo en la torreta) y facilitan ya una función en específico, como hacer un machueleado o un roscado en menos pasos.

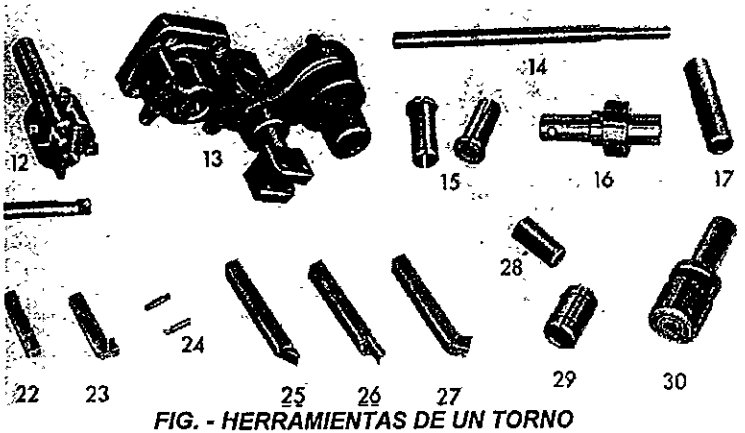


FIG. - HERRAMIENTAS DE UN TORNO

1.1.5.- FUNCIONES DE SOPORTE

Existen muchas funciones extras que se realizan en un torno para obtener un correcto funcionamiento y mejores resultados, como son aplicar enfriador al material, retirar la rebaba mientras la máquina realiza otro corte, monitorear el correcto herramentaje y ajuste de distancias.

La intervención de operadores experimentados ayuda mucho al elaborar las piezas, pues se deben tener conocimientos de los tipos de metales para realizar los cortes, el tipo de enfriador a usar, el volumen de metal a ser removido, la dureza de la herramienta a usar para con ellos hacer una correcta secuencia de operaciones junto con las adecuadas velocidades de corte y avances.

1.1.6.- TORNO HERBERT

Este torno es de operación automática, aunque puede realizar algunas operaciones manualmente ya que originalmente cuenta con un gabinete y un tablero de control donde pueden realizarse operaciones (movimientos) simples o preparar la programación de una secuencia de pasos (ver fig. 1.5).

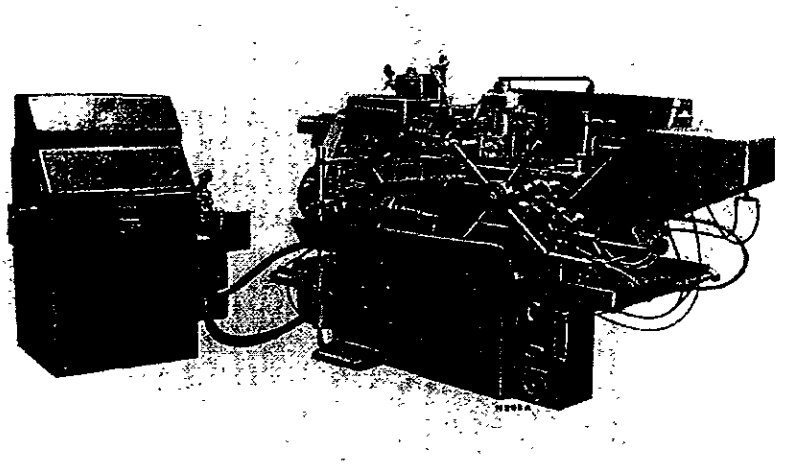


FIG. 1.5 - TORNO HERBERT Y GABINETE DE CONTROL

El torno opera con una alimentación de aire y electricidad. Cuenta con una serie de electroválvulas que activan distintas partes de la máquina (al recibir una señal eléctrica permiten el paso de aire para activar una zona de la máquina, ver fig. 1.6).

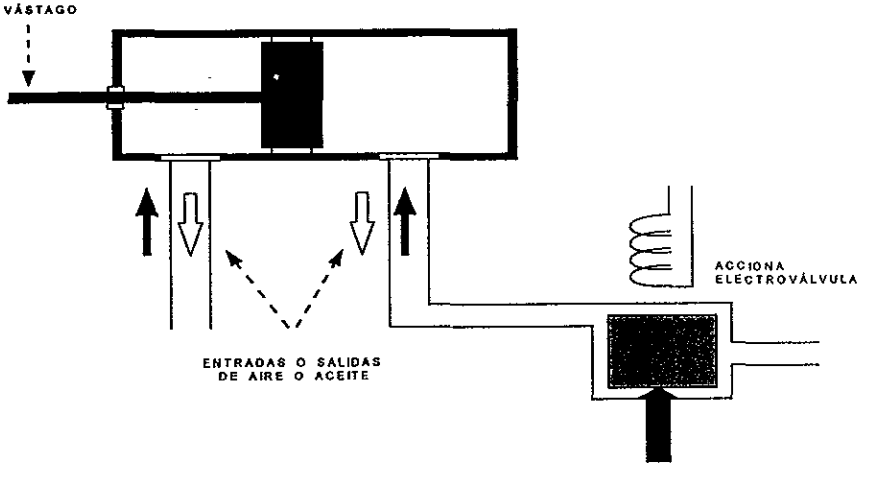


FIG. 1.6 - ELECTROVÁLVULA

En el gabinete de control están los interruptores para indicar si el torno va a trabajar manual o automáticamente. Ahí también están los interruptores asociados al *desplazamiento, tipos de avance y rangos de avance.*

1.1.6.1.- DESPLAZAMIENTOS DEL CABEZAL o TORRETA

Este torno tiene desplazamientos sobre 2 ejes, X y Z. El cabezal (donde se ubica la torreta) es automáticamente sujetado al desplazamiento sobre el eje Z durante la operación de la máquina por un cilindro neumático. Este cilindro incorpora un distribuidor el cual presuriza los cilindros usados por las herramientas, el taladro de cabeza y otros aditamentos.

La torreta indexa (gira) al retraerse sobre el eje Z para permitir el cambio de herramienta y la distancia de avance deseada (ver fig. 1.7).

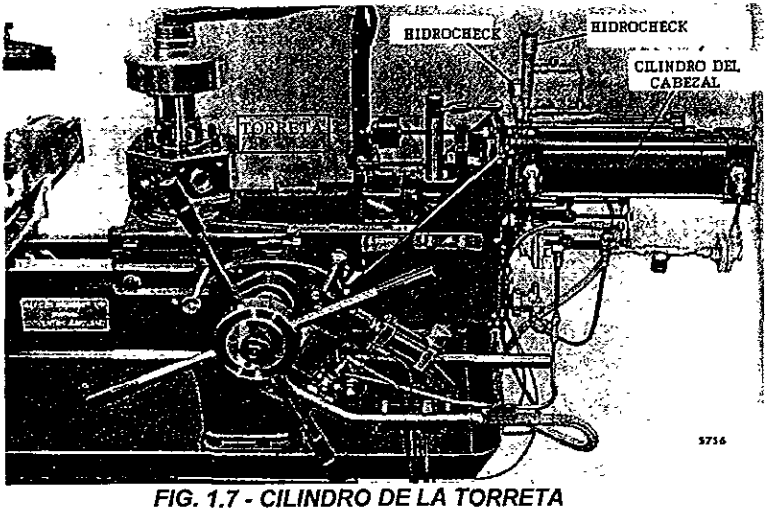


FIG. 1.7 - CILINDRO DE LA TORRETA

TIPOS DE AVANCE PARA EL DESPLAZAMIENTO DE LA TORRETA

Los distintos tipos de avance de la torreta, en este torno, pueden ser de tipo hidroneumático o mecánico y son 3 :

- ***QUICK POWER TRANSVERSAL (o AVANCE RÁPIDO)***

  Éste habilita el acercamiento rápido (a una posición inicial) y la retracción rápida y es impartido por un cilindro largo neumático.

- ***AVANCE MECÁNICO***

  El avance mecánico es transmitido por una flecha de avance operando através de un tornillo sin fin y un volante en el cabezal. Los avances son aprox. de 0.304, 0.152 y

0.076 mm. por revolución o de 0.012, 0.006 y 0.003 pulgadas por revolución. Estos avances son respectivamente "Grueso", "Medio" y "Fino".

- ***AVANCES HIDRONEUMÁTICOS***

Los avances hidroneumáticos son infinitamente variables desde 12.7 mm. (0.5 pulgadas) por minuto y son normalmente usados cuando son requeridos avances más rápidos que el avance mecánico grueso, como para roscados. Hay dos hidrochecks (hidroválvulas para el sistema de amortiguamiento) montados a los lados del cilindro de desplazamiento del cabezal y su uso restringe (frena) el avance del desplazamiento del cabezal para proporcionar el avance a velocidad constante (ver fig. 1.8).

El rango de avance de cada unidad se regula girando la perilla del regulador en la parte superior del hidrockeck para alterar el rango de flujo del aceite.

Generalmente antes de usar alguno de los dos últimos tipos de avance se utiliza el avance rápido (Quick Power) para llegar a una posición (la cual es detectada por un microinterruptor) a partir de la cual se usará otro tipo de avance que será el que nos llevará hasta el límite de avance permitido.

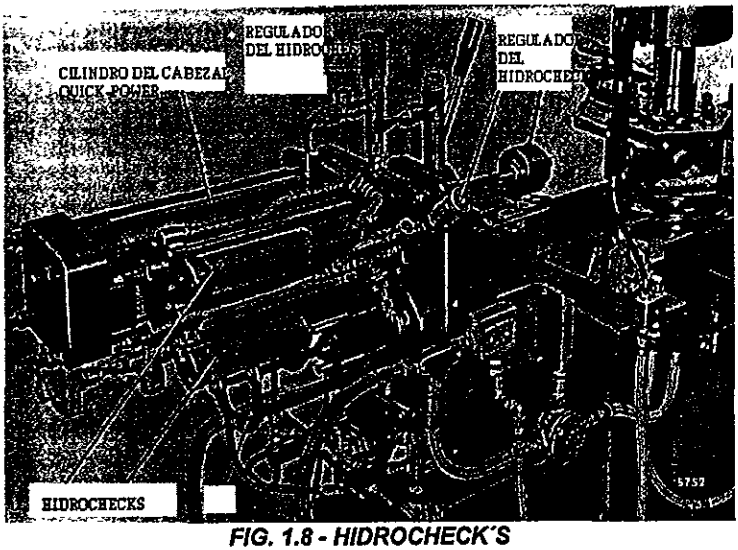


FIG. 1.8 - HIDROCHECK´S

FUNCIONAMIENTO DE LOS HIDROCHECKS

Los hidrochecks funcionan como sigue: cuando el cabezal se está desplazando, el aceite circula en la malla inferior y presenta una oposición despreciable al desplazamiento del cabezal. Si durante el deslizamiento se activa un microinterruptor se energiza una electroválvula que permite el paso de aire moviendo la válvula de libramiento de manera que la malla inferior queda bloqueada y el aceite fluye en la malla superior con un flujo determinado por la posición del regulador fijando con esto *la velocidad de avance.* Cuando el cabezal se retrae el aceite fluye através de perforaciones en el pistón (válvula de plato) sin importar que la válvula de libramiento esté o no activada, de esta manera la retracción no es restringida. Cuando se completa la retracción la válvula de plato se reestablece impidiendo el flujo del aceite en un sentido. Cuando el pistón se desplaza el volumen de su eje desplaza aceite dentro del cilindro, este aceite es retirado hacia el cilindro de compensación o cilindro de balance, lo contrario ocurre cuando el eje del pistón sale. Una indicación de la cantidad de aceite desplazada del cilindro la da el vástago del cilindro (ver fig. 1.9).

Los hidrocheck son extremadamente sensibles a variaciones en la presión. La presión de operación óptima para este torno es de 80 p.s.i. (5.6 Kg/cm^2).

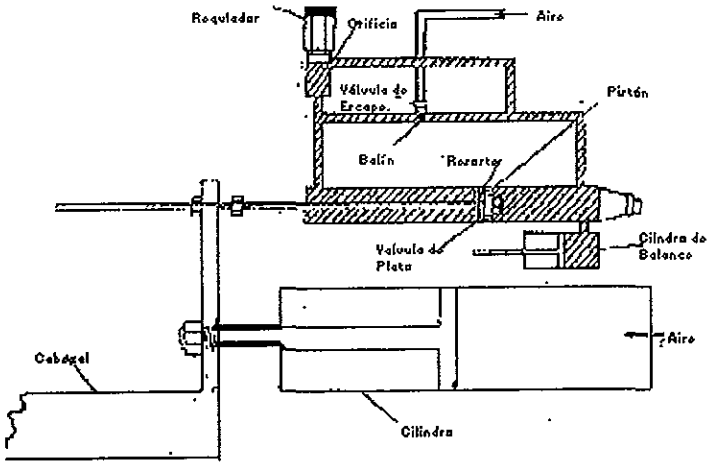


FIG. 1.9 - ESQUEMA DE UN HIDROCHECK

1.1.6.2- DEZPLAZAMIENTO CRUZADO

El desplazamiento cruzado está montado en un soporte sobre la cama del torno para dar movimiento sobre el eje X.

Un cilindro neumático de tres posiciones, colocado en la parte trasera del soporte , mueve el desplazamiento cruzado através del soporte desde su posición central para avanzar o retraer el poste de herramientas delantero o el trasero (ver fig. 1.10).

AVANCES PARA EL DEZPLAZAMIENTO CRUZADO

Dos cilindros hidráulicos, montados al frente del soporte, proporcionan el avance para cada poste de herramienta. El rango de avance de cada poste de herramientas es ajustable girando una perilla graduada.

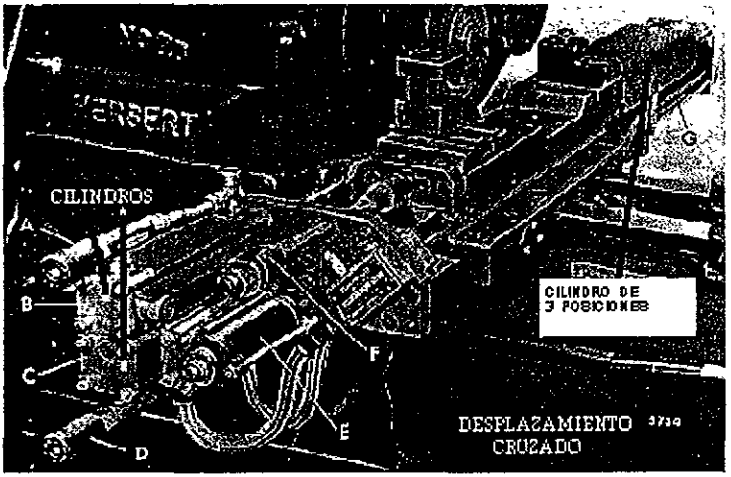


FIG. 1.10 - DESPLAZAMIENTO CRUZADO

DETECCIÓN DE MOVIMIENTOS

Para los desplazamientos en ambos ejes X y Z el torno cuenta con *microinterruptores* colocados al final de las distancias de avance permitidas, que nos permiten saber cuando se puede o no avanzar más.

1.1.6.3.- EL EJE CENTRAL DE LA MÁQUINA

Un motor eléctrico principal maneja 4 velocidades y está conectado a una caja con un clutch de altas/bajas velocidades y un sistema de engranes para dar un total de 16 diferentes velocidades (en 2 rangos de 8 velocidades). Este torno soporta velocidades del giro de su eje que van desde 28 hasta 2550 r.p.m. además de poder girar en ambos sentidos (ver fig. 1.11).

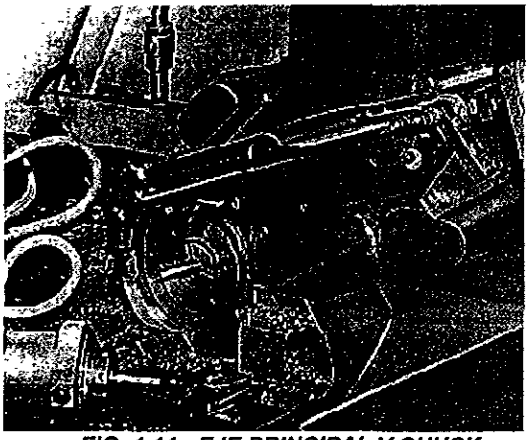


FIG. 1.11 - EJE PRINCIPAL Y CHUCK

1.1.6.4.- OPERACIÓN AUTOMÁTICA Y PROGRAMACIÓN

Originalmente este torno contaba con una unidad de control (un gabinete aparte y un tablero) para programar su operación automática, pudiendo ejecutar pasos simples y ciclos sencillos o repetitivos.

En el gabinete estaban interruptores asociados a los tipos y rangos de avances, dirección de los movimientos, velocidades del eje del motor, tiempos usados y tipo de ciclo a ejecutar. También estaban botones para comenzar a ejecutar el ciclo seleccionado, el cual podía ser un programa elaborado en el tablero asociado.

El tablero rectangular era una matriz de distintas funciones (1 a la 36) contra pasos (de la A a la Z). Estas funciones proporcionaban el mismo desempeño que el gabinete con sus interruptores. En cada paso se iban seleccionando (insertando un pequeño cilindro metálico en el orificio asociado) las funciones que se deseaban realizar, así al final se obtenía una secuencia de programación a ejecutar (serie de pasos con distintas funciones) sin estar cambiando manualmente los interruptores del gabinete (ver fig. 1.12).

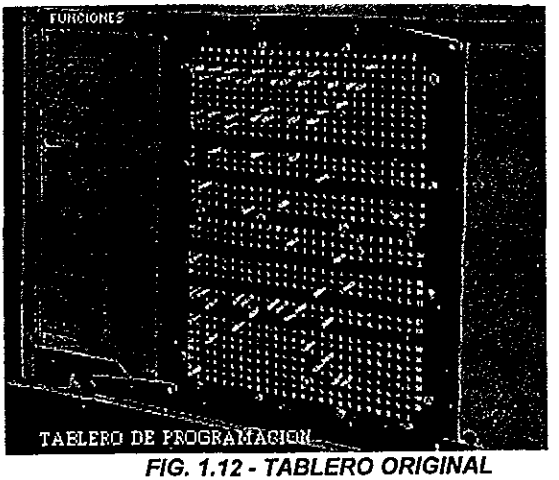


FIG. 1.12 - TABLERO ORIGINAL

La secuencia de programación obtenida se podía ejecutar desde el gabinete de control *automáticamente*, cada paso indicaba qué movimientos y/o selecciones debían hacerse en el torno, dependiendo lo indicado en cada paso se avanzaba al paso siguiente en el programa por la detección de la presión de algún microinterruptor (por un movimiento) o por la terminación de un tiempo (programado para la duración de ese paso).

No todas las funciones eran usadas en el tablero original y además el número de pasos posibles en un programa estaba limitado por el número de columnas disponibles en el tablero (de la A a la Z), así pues, sí para poder elaborar una pieza en el torno requería de más de este número de pasos debían hacerse 2 programaciones del tablero y ejecutar una después de la otra.

1 - 2 ARQUITECTURA Y ORGANIZACIÓN DE UNA PC - AT .

DEFINICIÓN

Una Computadora Personal ó *PC* puede ser cualquiera de los siguientes sistemas:

- un sistema de 8 / 16 bits (PC Clase XT)
- un sistema de 8 / 16 / 32 / 64 bits (PC Clase AT)

Los sistemas de clase *XT* (Tecnología Extendida) están basados en un microprocesador de 8 ó 16 bits (8088 ó 8086) e integran un bús tipo ISA (Industry Standard Arquitecture) para su sistema de expansión (slots).

Los sistemas de clase *AT* (Tecnología Avanzada) usan microprocesadores más avanzados de 16, 32 ó 64 bits (como el 80286, 80386, 80486, Péntium ó P6) y sistemas de expansión con buses de 16 ó más bits como pueden ser:

- Bús ISA de 16 bits
- Bús EISA (Enhanced ISA) de 16 / 32 bits
- Bús MCA (Micro Channel Arquitecture) de 16 / 32 bits
- Bús PC-Card (PCMCIA) de 16 bits
- Bús Local VESA de 32 / 64 bits
- Bús PCI (Periferical Component Interconnect) de 32 / 64 bits.

1.2.1.- ARQUITECTURA DE MICROPROCESADORES

La parte más importante de una PC es el procesador o CPU (Central Processing Unit), éste realiza todo el procesamiento y cálculos de el sistema.

El primer microprocesador desarrollado fue el 4004 de 4 bits (desarrollado por Intel), siguiendo con la familia de Intel surgieron otros de 8 bits como el 8008, el 8080 y el 8085, más adelante los de 16 bits como el 8088, 8086 y el 80286, posteriormente los de de 32 bits 80386 y 80486 y por último los de 64 bits Péntium y P6 (ver tabla 1.2).

PROCESADORES 8088 y 8086

El 8088 original corría a 4.77 Mhz. y fue rediseñado para correr a 8 Mhz. al igual que el 8086.

PROCESADORES 80188 y 80186

La estructura de registros de estos microsprocesadores es virtualmente idéntica a los de los 8088 / 8086, solo que contienen más vectores de interrupciones reservados y otras características integradas de Entrada / Salida (I/O). Este chip no se integró en la construcción de computadoras si no como controlador.

PROCESADORES 80286

Este micro es una versión avanzada del 8086 que es diseñado para ambientes multiusuario y multitarea. También incorpora una unidad manejadora de memoria (fig. 1.13). Las velocidades de esos chips son de 6, 8, 10, 12, 16 o hasta 20 Mhz. Este chip puede tener dos modos de operación: modo real y modo protegido (pudiendo direccionar 16 Mb. físicamente y hasta 1Gb. virtualmente).

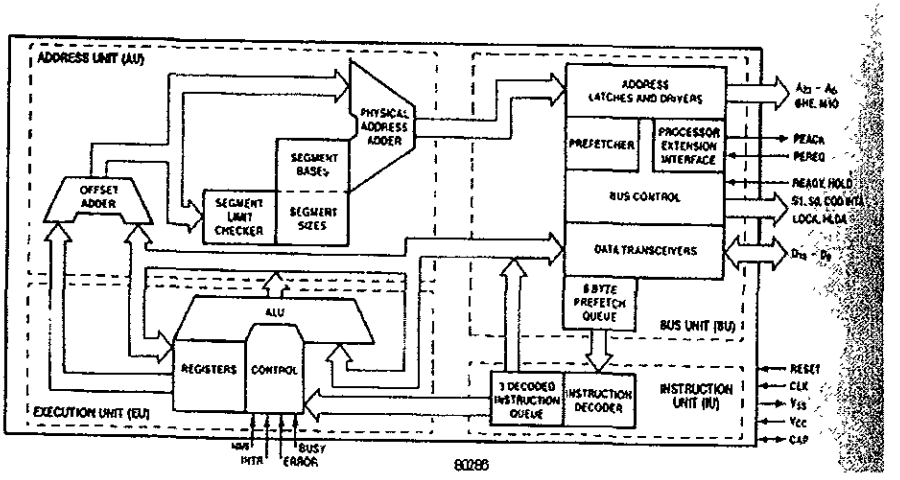


FIG. 1.13 - DIAGRAMA A BLOQUES DEL 80286

PROCESADORES 80386

Este microprocesador está optimizado para operación en alta velocidad y ambientes multitarea. Amplía su unidad manejadora de memoria. Las velocidades manejadas por estos chips son de 16, 20, 25, 33 o hasta 40 Mhz. El número de ciclos de reloj promedio en la ejecución de instrucciones es de 4. Dispone de tres modos de operación: modo real, modo protegido y modo real virtual (simular la operación de varios 8086´s en modo real, como si fueran varias máquinas virtuales).

PROCESADORES 80486

El número de ciclos de reloj promedio en la ejecución de instrucciones es de 2. Tienen un caché (de tipo 1 incluido, esto es 8 KB para instrucciones y datos) y pueden tener un coprocesador matemático (MCP ó FPU Floating Point Unit) integrado en el chip del micro. Utilizan ciclos de memoria de modo Burst (Los datos son transferidos de la memoria como localidades de 32 bits). Las velocidades manejadas por estos chips van desde 16 hasta 120 Mhz. También dispone de tres modos de operación: real, protegido y virtual real (Tabla 1.1).

Veloc. del reloj de la tarjeta	16 Mhz.	20 Mhz.	25 Mhz.	33 Mhz.	40 Mhz.	50 Mhz.
DX2 veloc proc.	32 Mhz.	40 Mhz.	50 Mhz.	66 Mhz.	80 Mhz.	No Aplica
DX4 (modo 2X) de Velocidad	32 Mhz.	40 Mhz.	50 Mhz.	66 Mhz.	80 Mhz.	100 Mhz.
DX4(modo 2.5X) de Velocidad	40 Mhz.	50 Mhz.	63 Mhz.	83 Mhz.	100Mhz	No Aplica
DX4 (modo 3X) de Velocidad	48 Mhz.	60 Mhz.	75 Mhz.	100 Mhz	120Mhz	No Aplica

Tabla 1.1 - Velocidades de operación del 80486 - DX2 y DX4 contra velocidades de reloj de la tarjeta madre.

PROCESADORES PÉNTIUM (Quinta Generación)

Este procesador puede ejecutar instrucciones a una velocidad de dos instrucciones por ciclo. Cuenta con dos líneas de ejecución de instrucciones, donde al proceso de operar dos instrucciones simultáneamente en diferentes líneas se le llama enparejamiento. El péntium contiene dos cachés internos (tipo 1), también contiene un coprocesador matemático interno. Algunos de estos chips requieren rangos de voltajes menores a 5V para operar.

PROCESADORES Péntium-Pro (Sexta Generación)

Este procesador cuenta con tres líneas de ejecución de instrucciones, con las cuales puede ejecutar múltiples instrucciones en un ciclo. Puede ejecutar instrucciones fuera de orden y tiene predicción dinámica de saltos y capacidades de ejecución especulativa. El núcleo del microprocesador es RISC (Reduced Instruccion Set Computer) mientras la interface de instrucciones externas es una CISC (Complex Instruccion Set Computer) clásica de Intel. Las velocidades de este micro pueden superar los 300 Mhz. Contiene internamente cachés tipo 1 y tipo 2 colocado, éste último, más cerca del procesador.

PROCESADORES INTEL Péntium MMX (Quinta y Sexta Generación)

Permiten a los procesadores optimizar la ejecución de tareas repetitivas. Operan con 64 bits de datos a la vez. Son más eficientes.

CLONES DE PROCESADORES

Existen otras compañías como AMD y Cyrix, que han desarrollado procesadores compatibles con los de la familia de Intel para uso en las computadoras personales. La mayoría de estos procesadores son compatibles en los pines con los originales pero otros requieren tarjetas-madres diseñadas especialmente para poder operar e integrarse en una computadora.

Procesador	Tamaño de Registros internos	Ancho del Bús de Datos	Ancho del Bús de Direcciones	Máximo de Memoria a Direccionar
8088	16 bits	8 bits	20 bits	1 Mb.
8086	16 bits	16 bits	20 bits	1 Mb.
80286	16 bits	16 bits	24 bits	16 Mb.
80386SX	32 bits	16 bits	24 bits	16 Mb.
80386SL	32 bits	16 bits	24 bits	16 Mb.
80386DX	32 bits	32 bits	32 bits	4 Gb.
80486SX	32 bits	32 bits	32 bits	4 Gb.
80486SL	32 bits	32 bits	32 bits	4 Gb.
80486DX	32 bits	32 bits	32 bits	4 Gb.
PéntiumOD	32 bits	32 bits	32 bits	4 Gb.
Péntium 75+	32 bits	64 bits	32 bits	4 Gb.
P6	32 bits	64 bits	36 bits	64 Gb.

TABLA 1.2 -ESPECIFICACIONES DE MICROPROCESADORES INTEL

Hubo otras compañías, como Motorola, que desarrollaron sus propias familias de microprocesadores pero en el área de las PC`s compatibles la familia elegida para integrarse en los sistemas de cómputo fue la de Intel.

A diferencia de un microcontrolador el microprocesador no incluye los dispositivos periféricos en su arquitectura interna si no que se le añaden para su manejo en forma externa.

Una de las maneras más comunes de describir un procesador es por el tamaño de su bús de datos y su bús direcciones. Un bús es simplemente una serie de conexiones que transportan señales de características similares.

1.2.1.1- TIPOS DE BUSES

BÚS DE DATOS

El bús de datos es usado para enviar y recibir datos de y hacia dispositivos periféricos y memorias externas.

Algunos procesadores tiene un bús de datos interno distinto a su bús de datos externo. Por ejemplo en el 8088 y el 80386SX su bús de datos interno es el doble de su bús de datos externo, por lo tanto antes de operar con los registros del microprocesador se deben de llenar con el dato adecuado en dos ciclos. Pero también existe el caso contrario en que el bús externo es el doble del bús interno, como en el Péntium.

BÚS DE DIRECCIONES

El bús de Direcciones se usa para describir una localidad de memoria a la cual el dato va a ser enviado o de la que va a ser obtenido. También sirve para direccionar dispositivos de entrada y salida (I/O).

El ancho de este bús determina el máximo de memoria que puede ser direccionado por cada microprocesador, éste se calcula usando como base el número 2 y elevándolo al número de bits del ancho del bús de direcciones.

BÚS DE CONTROL

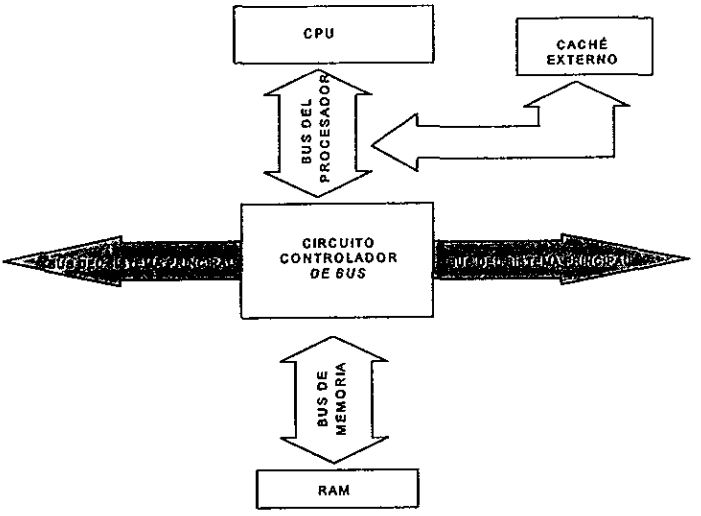
Este bús proveé señales de control para indicar que tipo de operación debe realizarse con la memoria u otro dispositivo de I/O.

BÚS DEL PROCESADOR

Este bús es usado para transferir datos entre el CPU y el bús del sistema principal y opera al mismo rango de reloj que el CPU.

BÚS DE MEMORIA

Es usado para transferir información entre el CPU y la memoria principal (RAM) del sistema. En casi todos los sistemas con velocidades de 16 o más Mhz. existe un circuito controlador de interfaz entre el bús del procesador y el de la memoria principal (ya que es más lento el acceso a la memoria RAM que al procesador), este circuito generalmente también es responsable del manejo del bús de I/O (fig.1.14).



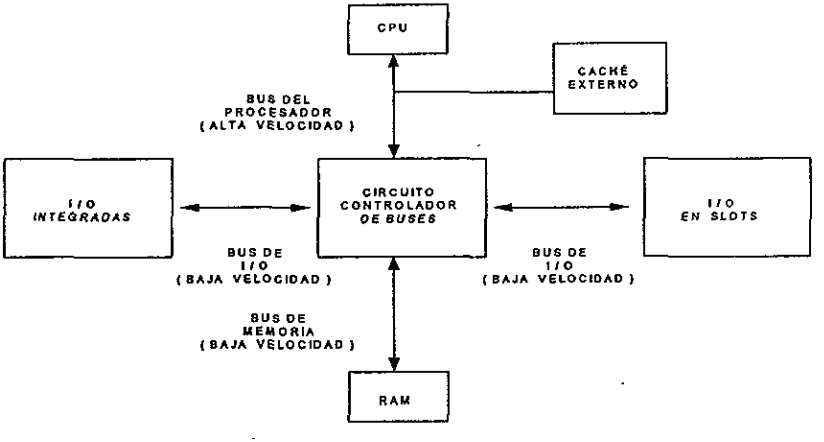
BUS DE MEMORIA

FIG. 1.14 - BÚS DE MEMORIA

BÚS DE I / O

Este bús también es llamado bús de expansión y permite al CPU comunicarse con dispositivos periféricos. Todo lo que va o viene desde un dispositivo (como el sistema de video, drives e impresora) viaja através de este bús (fig. 1.15).

Al añadir dispositivos a la computadora expandimos sus capacidades. Los slots de expansión nos ayudan para esto, el número de slots existentes en una computadora varía.



ESQUEMA DE BUSES TRADICIONAL EN UNA PC

FIG. BUSES DE UNA PC.

Los principales tipos de buses de I/O y sus características son:

- **Bús ISA** (Industry Satndard Arquitecture) fue introducido por IBM en 1981, existen dos versiones ,

la primera versión maneja 8 líneas de datos (8 bits de transferencia) y 20 de direcciones, permitiendo al slots manejar 1Mb. de memoria, esta versión usada en XT´s operaba a 4.777Mhz.

Las dimensiones de las tarjetas adaptadoras ISA de 8 bits son:

4.2 pulgadas (106.68mm) de alto
13.13 pulgadas (333.5mm) de largo
0.5 (12.7mm) pulgadas de ancho

la segunda versión maneja 16 líneas de datos y 24 de direcciones, esta versión usada en AT´s opera en el rango de 6 - 8.33 Mhz.

Las dimensiones de las tarjetas adaptadoras ISA de 16 bits son:

4.8 pulgadas (121.92mm) de alto
13.13 pulgadas (333.5mm) de largo
0.5 (12.7mm) pulgadas de ancho

- **Bús MCA** (*Micro Channel Arquitecture*) *desarrollado también por IBM, debido al* surgimiento de sistemas de 32 bits, es completamente diferente del ISA. Los cuatro tipos de slots involucrados en este diseño son:

de 16 bits, de 16 bits con extensiones de video
de 16 bits con extensiones de memoria relacionadas y de 32 bits.

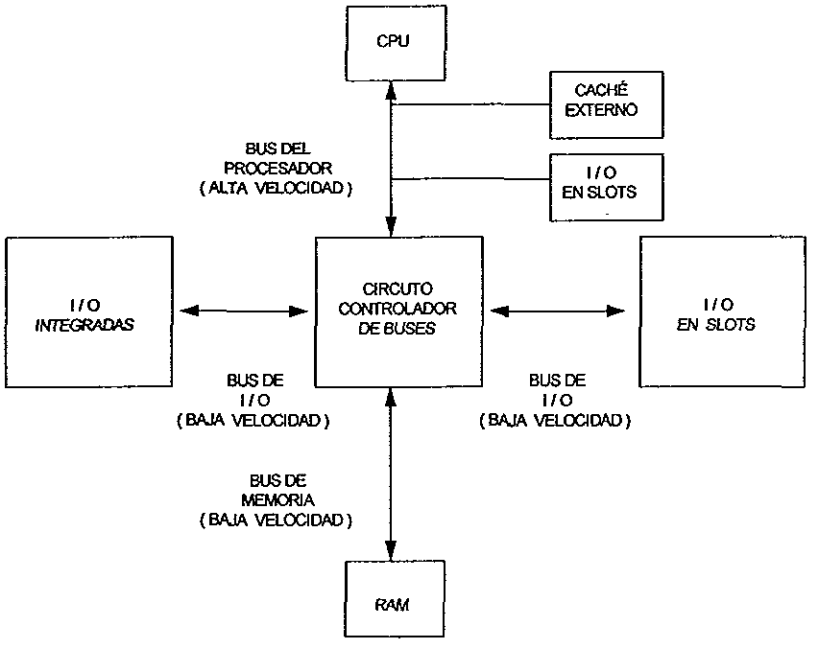
- **BÚS EISA** (Extendend Industry Standard Arquitecture) fue desarrollado por un comite encabezado por Compaq, conserva la compatibilidad con ISA. Este tipo de slot puede manejar hasta 32 bits de información con operación de 8.33 Mhz.

Las dimensiones de las tarjetas adaptadoras EISA son:

5 pulgadas (127mm) de alto
13.13 pulgadas (333.5mm) de largo
0.5 (12.7mm) pulgadas de ancho

- **VESA - Local Bús o VL-Bús** (Video Electronics Standard Arquitecture) fue desarrollado por un comité encabezado por NEC, al ser un bús local aprovecha la velocidad misma del procesador. Este tipo de slots tiene limitaciones de velocidad (a velocidades mayores de 33 Mhz. causa problemas) y de dependencia (fue desarrollado para chips 80486 y requiere un chip de puente para otros sistemas, ver fig. 1.16).

Si se tiene un sistema EISA o MCA los slots de VL-Bús son extensiones de esos slots.



ESQUEMA DE OPERACIÓN DEL BUS LOCAL EN UNA PC

FIG. 1.16 - BÚS LOCAL

- **BÚS PCI** (Periferical Component Interconnect) fue creado por Intel, ofreciendo las ventajas de un bús local, independencia del microprocesador pero con aprovechamiento de las múltiples capacidades de éste.

Las especificaciones del bús PCI rediseñan el bús de la PC tradicional por la inserción de otro bús entre el CPU y el bús de I/O nativo mediante el uso de chips puentes (fig. 1.17).

Las tarjetas PCI usan un conector de tipo MCA. El tamaño de estas tarjetas puede ser el mismo que de cualquier otra usada en otro tipo de slot de I/O.

Existen tres distintas configuraciones de los pines de estas tarjetas dependiendo si el sistema en que se usan trabaja con 5 V, con 3.3 V o con ambos.

Otra característica de PCI es su diseño para la especificación "Plug and Play" de Intel (La tarjeta no tiene switches ni jumpers y son configuradas por el software).

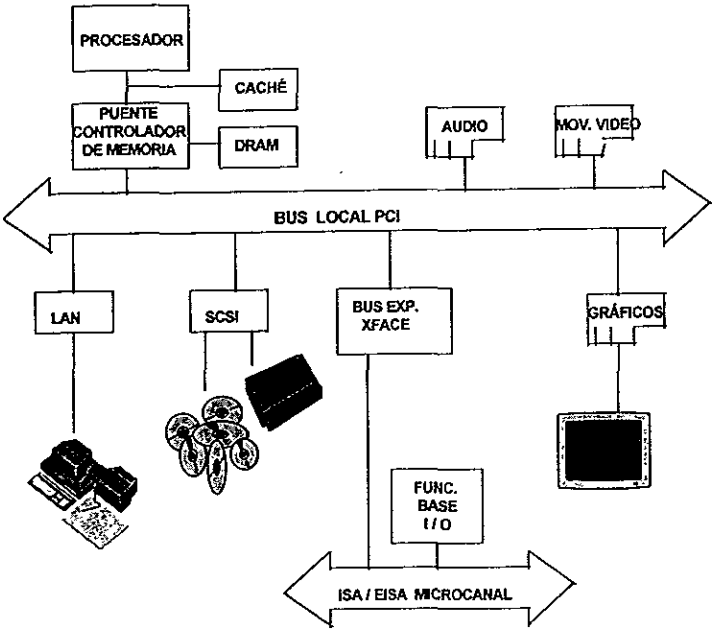


DIAGRAMA CONCEPTUAL DEL BUS PCI

FIG. 1.17 - BÚS PCI

- **BÚS PC-CARD PCMCIA** fue desarrollada por un consorcio de más de 300 compañías para su adaptación en sistemas portátiles (como laptop y notebook) permitiendo grandes capacidades de expansión de estos (memoria, discos, fax-modems). Existen cuatro tipos:

  PCMCIA -Tipo I. Pueden manejar tarjetas de 3.3 mm de espesor. Se diseñó para trabajar con tarjetas de expansión de memoria.

  PCMCIA -Tipo II. Pueden manejar tarjetas de 5 mm de espesor y tarjetas de tipo I.

  PCMCIA -Tipo III. Pueden manejar tarjetas de 10.5 mm de espesor para manejar discos duros removibles. Puede manejar tarjetas de tipo I y II.

  PCMCIA -Tipo IV. Pueden manejar tarjetas de tipo I, II y III . Manejan discos duros de menos de 10.5 mm de espesor.

1.2.1.2.- REGISTROS INTERNOS

El tamaño de los registros internos del microprocesador es una indicación de cuanta información puede éste operar a la vez (fig. 1.18).

Los registros son entidades de almacenamiento ubicadas dentro del microprocesador y se categorizan en tres grupos:

- Registros de propósito general

  AX (Accumulator) - almacena el resultado temporal después de una operación aritmética o lógica.
  BX (Base) - almacena la dirección base (offset) de datos localizados en memoria.
  CX (Count) - contiene la cuenta para algunas instrucciones como una cuenta de corrimiento.
  DX (Data) - registro de propósito general que también almacena la parte más significativa del producto después de una multiplicación de 16 o 32 bits, la parte más significativa del dividendo antes de una división y el número de puerto para una instrucción de I/O variable.

- Registros apuntadores e índices

  SP (Stack Pointer) - usado para direccionar datos en un stack (estructura de tipo LIFO).
  BP (Base Pointer) - usado para direccionar un arreglo de datos en el stack de memoria.
  SI (Source Index) - usado para direccionar datos fuentes indirectamente para uso con las instrucciones de cadenas.
  DI (Destination Index) - generalmente usado para direccionar el destino del dato indirectamente para el uso con instrucciones de cadenas.
  IP (Instruccion Pointer) - usado para direccionar la próxima instrucción a ejecutar por el microprocesador.

- *Registros de segmento*

Estos registros generan direcciones de memoria mayores al sumarse con el contenido de registros que contienen el offset o desplazamiento en una dirección y en la dirección resultante se almacena información de acuerdo a cada segmento.

CS (Code) - El segmento de código es una sección de memoria que almacena programas y procedimientos usados por los programas.
DS (Data) - en el segmento de datos está la mayoría de los datos usados por un programa.
ES (Extra) - es un segmento adicional que es usado por algunas instrucciones de cadenas.
SS (Stack) - aqui se define él área de memoria usada para el stack.
FS y GS - son registros de segmentos suplementarios que están disponibles en el 80386 y 80486.

Existe un registro adicional llamado registro de Banderas que indica el estado de el microprocesador así como el control de su operación, los bits importantes en ese registro son:

C (Carry) - indica acarreos o condiciones de error.
P (Parity) - Indica la paridad.
A (Auxiliary Carry) - indica un acarreo entre los bits de las posiciones 3 y 4 de un resultado.
Z (Zero) - indica que el resultado de una operación fue cero.
S (Sign) - indica el signo.
T (Trap) - usada en el depurado.
I (Interrup) - controla la operación de las peticiones de interrupciones.
D (Direction) - controla la selección del incremento o decremento para los registros DI y SI.
O (Overflow) - indica un sobreflujos o desbordamientos.
IOPL (input/output privilege level) - usado en la operación del modo protegido para seleccionar el nivel de privilegios para dispositivos de I/O.
NT (Nested Task) - indica que la tarea actual está anidada con otra tarea en modo de operación protegido.
VM (Virtual Mode) - Selecciona el modo virtual de operación en un sistema en modo protegido.
AC (Aligment Check) - verifica si se direccionó una palabra o palabras dobles en un espacio incorrecto.

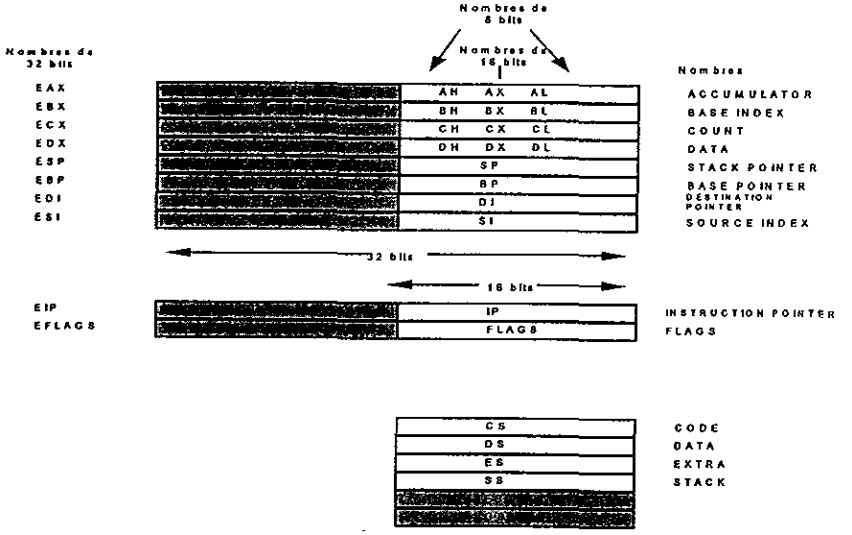


FIG. 1.18 - REGISTROS INTERNOS

1.2.2.- MEMORIA Y EL MICROPROCESADOR

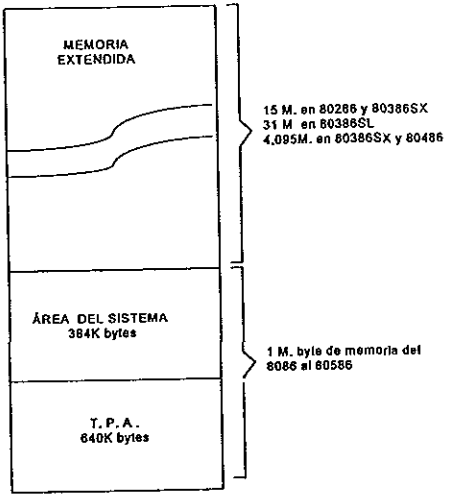
El espacio de direcciones en un sistema basado en microprocesadores es referido como memoria lógica o memoria física. La estructura de la memoria lógica es diferente de la estructura de la memoria física. La memoria lógica es la memoria del sistema como la ve el programador, mientras la memoria física es la estructura de hardware actual del sistema de memoria.

El espacio de memoria lógica es básicamente el mismo para todos los procesadores de Intel y sus clones, está numerada por bytes.

Los bancos de memoria física difieren en longitud. En el 8088 tiene 8 bits de ancho, en el 8086, 80286 y 80386SX es de 16 bits y en el 80386DX, 80486 y Pentium es de 32 bits.

1.2.2.1.- MEMORIA EN LA COMPUTADORA PERSONAL

La computadora personal fue basada originalmente en el procesador 8088, así que su memoria principal era solo de 1Mb. A partir del 80286 es posible direccionar arriba de 1Mb., esta memoria adicional es llamada Sistema de Memoria Extendida (XMS).



MAPA DE LA MEMORIA EN UNA PC .

FIG. 1.19 - MAPA DE MEMORIA

Los primeros 640 K bytes de la memoria del sistema en las PC´s son llamados área de programas transitorios (TPA, transient program area) . La TPA contiene memoria RAM para almacenar aplicaciones de software, el sistema operativo y varios programas que controlan dispositivos de I/O. Almacenado después de la TPA está el área del sistema que contiene memoria BIOS (basic I/O system) para controlar el sistema, la RAM de video para almacenar imágenes de video y áreas abiertas que pueden ser usadas por páginas EMS y opciones instalables para el sistema de la computadora. Arriba del primer megabyte de memoria tenemos el sistema de memoria extendida que contiene programas de caché de disco y otros segmentos de datos definidos por el sistema operativo (ver fig. 1.19).

1.2.2.2.-MODOS DE OPERACIÓN Y GENERACIÓN DE DIRECCIONES EN EL 80286

El 80286 tiene dos modos de operación, el modo real y el modo protegido. Cuando se opera en el modo real, el micro imita a los micros 8088 o 8086 y está basado en solo 1Mb. de espacio en memoria. En el modo protegido, el espacio de direcciones reales se expande a 16Mb., el cual se puede mapear en un gigabyte de memoria virtual por tarea. (Un sistema con memoria virtual almacena instrucciones en memoria secundaria y trae bloques de ellas a memoria principal cuando las va necesitando). Cualquier programa escrito para el 8088/8086 correrá sin ninguna modificación en este último modo.

- ***DIRECCIONAMIENTO DE MEMORIA EN EL MODO REAL***

Una dirección de segmento y una dirección de desplazamiento (offset) generan una dirección de memoria en este modo. La dirección del segmento, localizada en uno de los registros de segmento, define la dirección de inicio de un segmento de memoria de 64Kbytes. La dirección de desplazamiento selecciona una localidad en el segmento de memoria de 64Kbytes, a partir de su dirección de inicio.

En el modo real a cada segmento de registro internamente se le añade un *0H* al final de su parte derecha para formar direcciones de memoria de 20 bits.

- ***DIRECCIONAMIENTO DE MEMORIA EN EL MODO PROTEGIDO***

La dirección del segmento, mencionada en el direccionamiento en modo real, no está presente en el modo protegido. En lugar de esta dirección, el registro de segmento contiene un selector que selecciona uno de 8,192 descriptores de una tabla de descriptores. Los descriptores describen la localidad, longitud y derechos de acceso de un segmento de memoria. Indirectamente el registro de segmento todavía selecciona un segmento de memoria, pero no directamente como en el modo real (ver fig. 1.20).

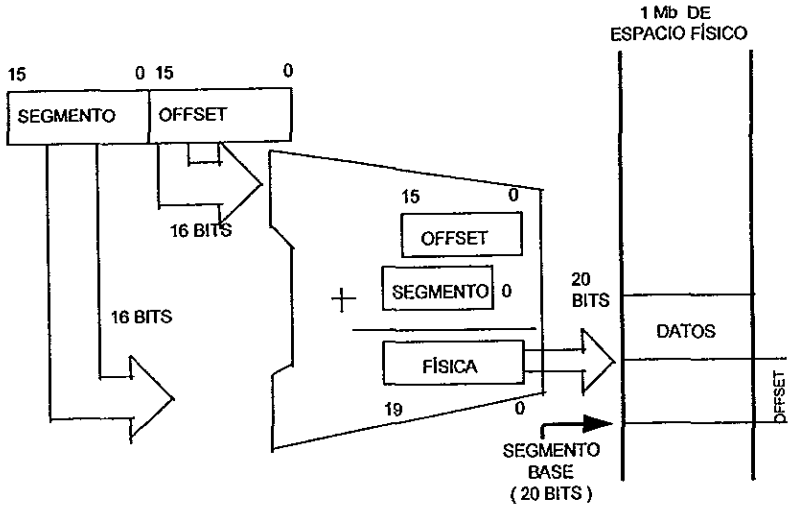
Hay dos tablas de descriptores usadas con los registros de segmento: una contiene descriptores globales (contienen segmentos que aplican a todos los programas) y otra contiene descriptores locales (son únicos a una aplicación). Cada tabla de descriptores contiene 8,192 descriptores para un total de 16,384 descriptores disponibles a la vez. Como los descriptores definen un segmento de memoria esto permite tener hasta 16,384 segmentos de memoria para ser descritos por cada aplicación.

7	00000000	00000000	6
5	DERECHOS DE ACCESO	BASE (B23 - B16)	4
3	BASE (B15 - B0)	BASE (B15 - B0)	2
1	LIMITE (L15 - L0)	LIMITE (L15-L0)	0

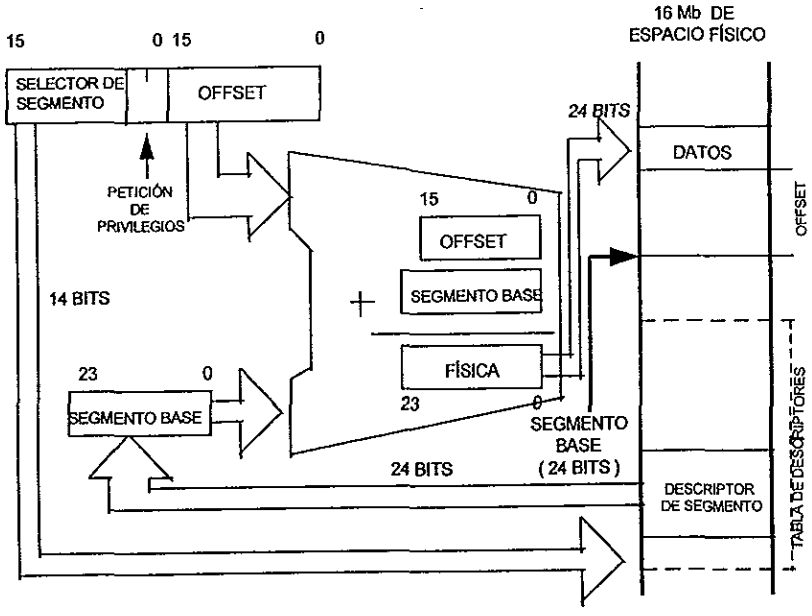
FIG. 1.20 - DESCRIPTOR DEL 80286

La porción de dirección base del descriptor es usada para indicar la localidad de inicio del segmento de memoria. En el 80286 la dirección base es de 24 bits, así que los segmentos pueden empezar en cualquier lugar de los 16 MB´s. de memoria, el segmento límite contiene la última dirección de desplazamiento encontrada en un segmento.

Los dos modos de direccionamiento mencionados se ilustran a continuación (fig. 1.21):



GENERACIÓN DE DIRECCIONES EN EL 8088 / 8086 -*MODO REAL*



GENERACIÓN DE DIRECCIONES EN EL 80286 -*MODO PROTEGIDO*

FIG. 1.21 - GENERACIÓN DE DIRECCIONES EN AMBOS MODOS

1.2.3.-INTERRUPCIONES

Las interrupciones son muy útiles para manejar dispositivos de I/O que proporcionan o necesitan rangos de transferencia de datos relativamente lentos.

Los microprocesadores de Intel incluyen dos pines para petición de interrupciones el INTR (interrup request) y NMI (nonmaskable interrup) y un pin (INTA) que reconoce las peticiones de interrupción através de INTR. También tienen interrupciones de software INT, INTO, INT 3 y BOUND. Dos banderas de interrupción IF (interruption flag) y TF (trap flag), son también usadas y una instrucción de retorno especial (IRET).

MANEJO DE INTERRUPCIONES

Cuando ocurre una petición de interrupción, el dispositivo que la genera dá un número específico al CPU, en base a ese número la dirección de la rutina que la atenderá se extrae de una tabla. La tabla de los vectores de interrupción está localizada en los primeros 1,024 bytes de memoria en las direcciones 000000H - 0003FFH. Ésta contiene 256 diferentes vectores (de 4 bytes de longitud). Cada vector contiene la dirección de una rutina de servicio de interrupción (ISR). Los últimos 224 vectores están disponibles para aplicaciones de usuario (ver fig. 1.22).

De las interrupciones de software:

INT n - llama a la ISR que comienza en la dirección representada por en vector número n.

INT 3 - es usada como una interrupción de rompimiento porque es fácil insertar una instrución de 1 byte en un programa.

BOUND - ésta tiene dos operandos (compara un registro con dos palabras de memoria) y genera una interrupción de tipo 5.

INTO - ésta checa la bandera de desbordamiento (overflow) y genera una interrupción de tipo 4.

Las banderas de interrupción son limpiadas después de que el contenido del registro de banderas es almacenado en el stack.

Después de la ejecución de la ISR, el estado del micro se restablece sacando los datos de los registros almacenados en el stack.

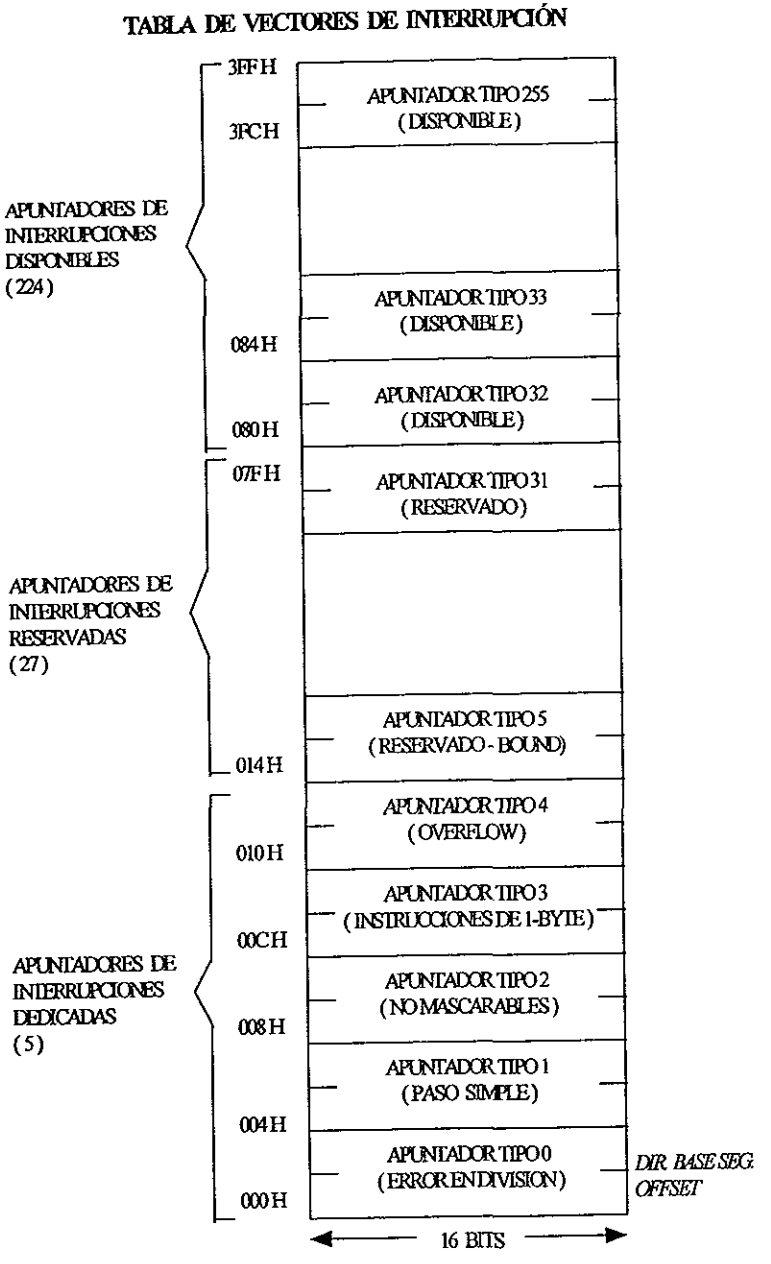


FIG. 1.22 - VECTORES DE INTERRUPCIÓN

1.2.4.- RECURSOS DEL SISTEMA PARA DESARROLLO DE INTERFACES DE I/O.

Los recursos del sistema son los canales de comunicación, direcciones y otras señales usadas por dispositivos de hardware para comunicarse en el bús. Esos recursos del sistema típicamente incluyen:

- Canales IRQ (Interrupt Request)
- Canales DMA (Direct Memory Access)
- Direcciones de Puertos I/O y Direcciones de memoria

Al desarrollar una tarjeta u otro dispositivo de I/O no siempre se tienen los mismos requerimientos.

1.2.4.1.- INTERRUPCIONES (IRQ's)

Estos canales o interrupciones de hardware, son usadas por varios dispositivos para indicar a la tarjeta madre que una petición debe ser atendida, son representados por cables en la tarjeta madre (líneas en el cto. impreso) y en los conectores de los slots.

Las interrupciones de hardware son generalmente priorizadas por sus números, con algunas excepciones las interrupciones de más alta prioridad tienen los números más bajos, éstas pueden interrumpir a las de más baja prioridad, esto implica que pueden ocurrir interrupciones anidadas.

1.2.4.2.-CANALES DMA (Direct Memory Access)

Estos canales son usados por dispositivos de comunicaciones de alta velocidad. Los canales DMA algunas veces pueden ser compartidos si los dispositivos no son del tipo que pudieran necesitarlos al mismo tiempo.

Los accesos directos a memoria generalmente ocurren entre un dispositivo de I/O y memoria sin el uso del microprocesador. Un pin en el micro (HOLD) es usado para solicitar una acción de DMA y otro pin (HLDA) saca una señal que reconoce la petición. Cuando se detecta la señal de HOLD (un 1 lógico) el microprocesador, detiene la ejecución del programa, coloca sus buses de direcciones, datos y control en un estado de alta impedancia e indica, sacando un 1 lógico por el pin HLDA que el HOLD ha sido atendido.

1.2.4.3.- PUERTOS DE I/O

El set de instrucciones del microprocesador cuenta con instrucciones para leer (IN) y enviar (OUT) datos (almacenados en el registro AL de 8 bits o AX de 16 bits) a dispositivos de I/O.

La dirección del dispositivo de I/O externo es llamada "fija" (fixed) si es de 8 bits (almacenada en la instrucción), y "variable" si son 16 bits (almacenados en el registro DX). A la dirección del dispositivo también se le llama número de puerto (puerto físico).

Cada vez que se transfiere un dato usando una instrucción de I/O, la dirección de I/O aparece en el bús de direcciones y el dispositivo periférico la decodifica en la misma forma que se decodifica una dirección de memoria. Si la dirección es fija aparece la dirección en el bús en los bits de A7 - A0 con A8 - A15 en cero, y si es variable aparece de A15 - A0. Esto significa que las primeras 256 direcciones de puertos (00H - FFH) son accesadas por direcciones fijas o variables pero cualquier dirección de 0100H - FFFFH es solo accesada por una dirección variable.

Hay dos métodos completamente diferentes de crear las interfaces de I/O al microprocesador (fig. 1.23):

- *I/O Aislado* : Es la técnica más común usada en la computadora personal, aquí las localidades de I/O son aisladas del sistema de memoria en un espacio diferente. Las direcciones para dispositivos de I/O aislados son llamadas puertos (físicos). Usando este método no utilizamos ningún espacio en memoria para almacenar información relativa al dispositivo. Señales de control separadas que indican una operación de lectura de I/O (IORC) o una escritura de I/O (IOWC) se generan al usar este tipo de acceso (usando las instrucciones IN, INS, OUT y OUTS del set del microprocesador).

- ***I/O Mapeado a memoria*** : aquí no se usan las mismas instrucciones de transferencia si no alguna otra para transferencia de datos entre el procesador y memoria. Un dispositivo mapeado a memoria es tratado como una localidad en el mapa de memoria, una porción de memoria es usada como mapeo para el I/O, esto puede reducir la cantidad de memoria disponible para aplicaciones. Aquí no se generan señales separadas de control lo que también puede reducir el tamaño del circuito requerido para decodificar.

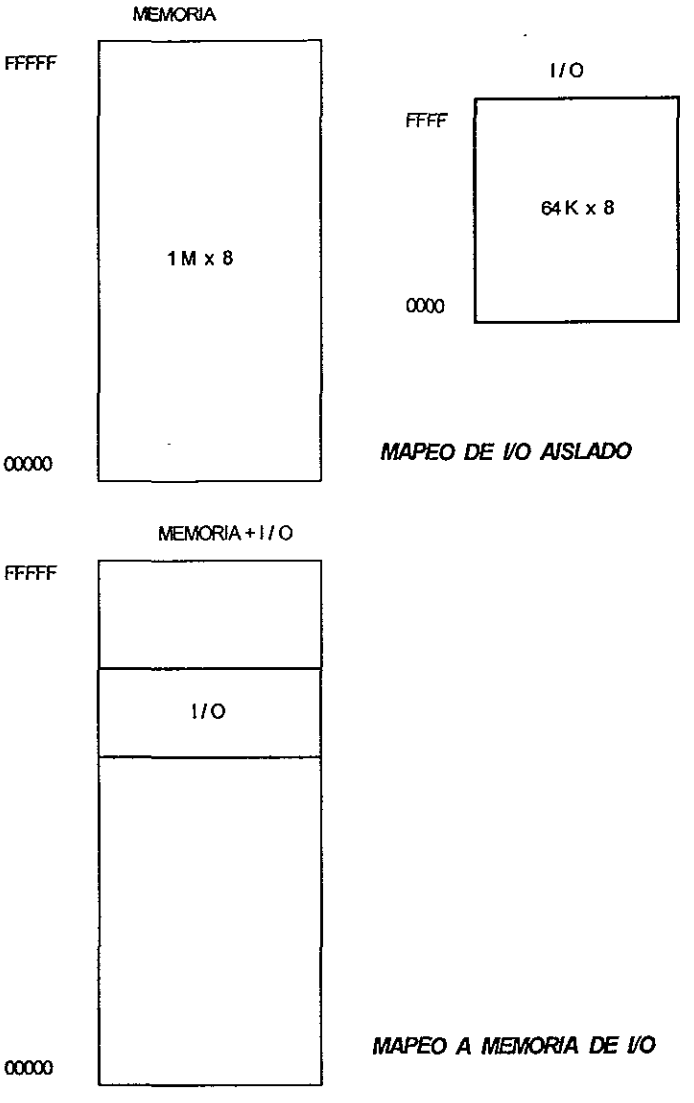
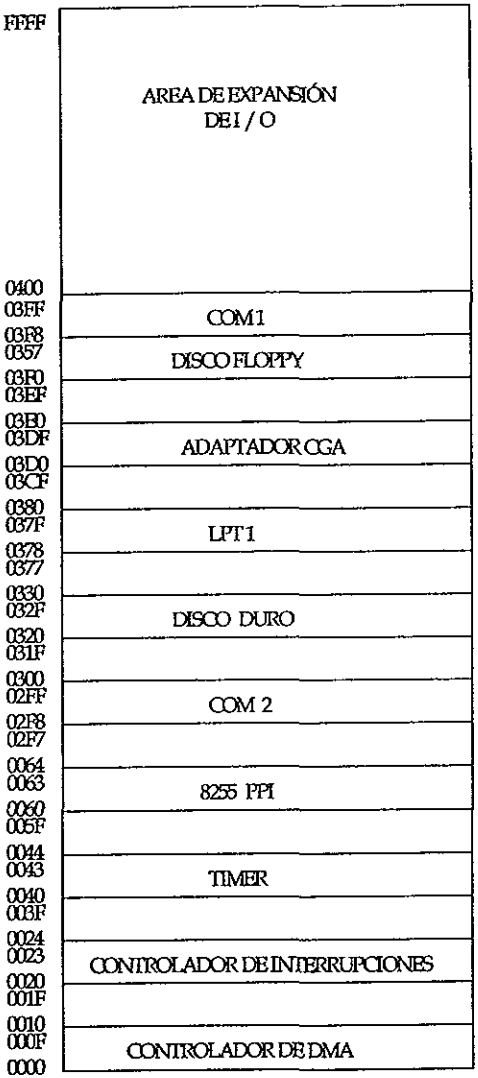


FIG. 1.23 - MAPEOS DE I/O.

En la computadora personal los puertos de I/O aislados son usados para controlar dispositivos periféricos, en ella se usa parte del mapa de I/O para funciones dedicadas, el espacio de puertos entre 0000H y 03FFH está normalmente reservado para el sistema y de 0400H a FFFH está generalmente disponible para aplicaciones de usuario (ver fig. 1.24).



MAPA BÁSICO DE I/O PARA UNA COMPUTADORA PERSONAL

FIG. 1.24 - MAPA DE I/O

1.3 - MANEJADORES DE CORRIENTE DIRECTA Y CORRIENTE ALTERNA

Considerando que la PC entrega señales digitales (0 o 5 Vcd) nos vemos en la necesidad de acoplar éstas, a las corrientes y voltajes que manejan este tipo de máquinas (tornos, fresadoras, mandriladora, etc.), que por lo regular cuentan para su control con electroválvulas, solenoides, motores de corriente alterna y corriente directa.

El acoplamiento se realiza dependiendo de que tipo de alimentación se tenga para los dispositivos antes mencionados; así encontramos electroválvulas activadas con c.d. y c.a. que permiten o no el paso de un fluido (gas o líquido) que genera un movimiento de alta *potencia*. Por lo tanto se describirán a continuación los dispositivos electrónicos necesarios para poder manejar los diferentes tipos de carga.

1.3.1.- MANEJADORES DE CORRIENTE DIRECTA

Para realizar el control de cargas de corriente directa los dispositivos que más comúnmente se utilizan son: los relevadores y los transistores. En la actualidad la mayor parte de cargas alimentadas con corriente directa son activadas por medio de transistores, ya que ocupan un espacio pequeño, son de bajo costo y requieren una corriente pequeña en la base para excitar adecuadamente a la carga, que puede manejar cientos de veces mayor corriente que la de la base.

El *transistor*, dispositivo de 3 terminales, se puede ver como dos uniones ***pn*** (o dos diodos) colocadas espalda con espalda. Recordando las secciones que lo conforman, el *emisor*, capa de tamaño medio diseñada para emitir o inyectar electrones está bastante contaminado, la *base*, con una contaminación media es una capa delgada diseñada para permitir el paso de electrones y el *colector*, capa grande diseñada para captar electrones está poco contaminado (fig. 1.25).

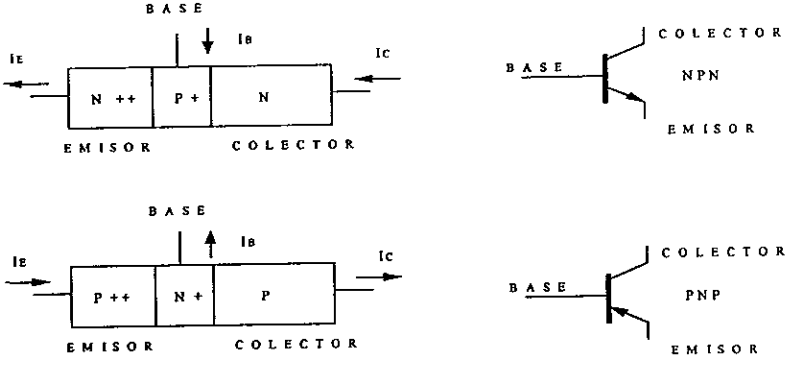


FIG. 1.25 - TRANSISTORES

El transistor cuenta con 4 regiones de operación básicas , ya que cada una de las dos uniones puede tener polarización directa o inversa (fig. 1.26).

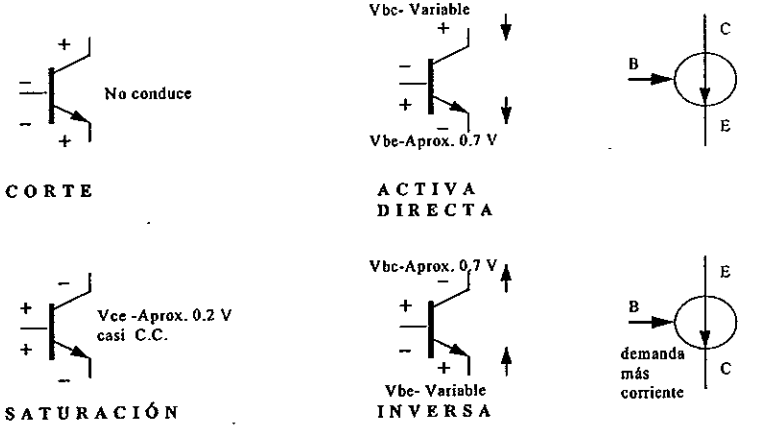
Región de Corte. Cuando la polarización de ambas uniones es inversa, solo fluye una pequeña corriente de fuga através del dispositivo así que estaría apagado.

Región activa directa. Aquí la unión Base-Emisor tiene polarización directa y la unión Base-Colector tiene polarización inversa, así nuestro dispositivo puede funcionar como una fuente de corriente accionada por corriente.

Región inversa. Aquí la unión Base-Colector tiene polarización directa y la unión Base-Emisor tiene polarización inversa, nuevamente nuestro dispositivo puede funcionar como una fuente de corriente accionada por corriente, pero demandará una corriente mayor para operar pués hay que recordar que el emisor es la capa más contaminada.

En estas dos últimas regiones de operación, la corriente de base es quien controla el flujo de corriente através del transistor, dependiendo de la cantidad de corriente inyectada por la base, provocará que fluya una determinada corriente del colector ($I_C = \beta I_B$) resultando una corriente de emisor ($I_E = I_B + I_C$) .

Región de Saturación. Aquí ambas uniones tienen polarización directa, así entre Colector y Emisor hay una diferencia de voltaje muy pequeña (aprox. 0.2 V) fluyendo en el dispositivo la corriente casi sin resistencia (como en un corto circuito).



REGIONES DE OPERACIÓN DEL TRANSISTOR
(de acuerdo a su polarización)

FIG. 1.26 - TRANSISTORES

Al contar con las regiones de corte (no conducción) y la de saturación (conducción), el transistor es un dispositivo que nos es muy útil para pasar de un estado de encendido a uno de apagado (conmutación).

1.3.2.- MANEJADORES DE CORRIENTE ALTERNA

TIRISTORES

El nombre de *Tiristor* se refiere a una familia genérica de dispositivos de cuatro capas *pnpn* con características de conmutación muy útiles.

Para el manejo de corriente alterna se cuenta con SCR's, TRIAC's, UJT's, SIDAC's, de los cuales los más usados son los dos primeros, así es posible el disparo de contactores, solenoides y en algunos casos electroválvulas que requieren alimentación de corriente alterna (c.a.).

El SCR (Silicon Controlled Rectifier) se utiliza en controles de relevadores, muestreadores, cargadores de batería, circuitos de protección, inversores y circuitos de control (entre otras cosas).

Los SCR's se pueden construir para control de potencia en la región de megawatts y soportan corrientes hasta de 1500 A a 2000V.

La siguiente figura (1.27) ilustra el símbolo de dispositivo y la construcción del SCR, junto con el esquema equivalente al SCR como un circuito de dos transistores TBJ, en esencia el SCR es eso, un par de transistores npn y pnp en los que la base de uno está conectada con el colector del otro.

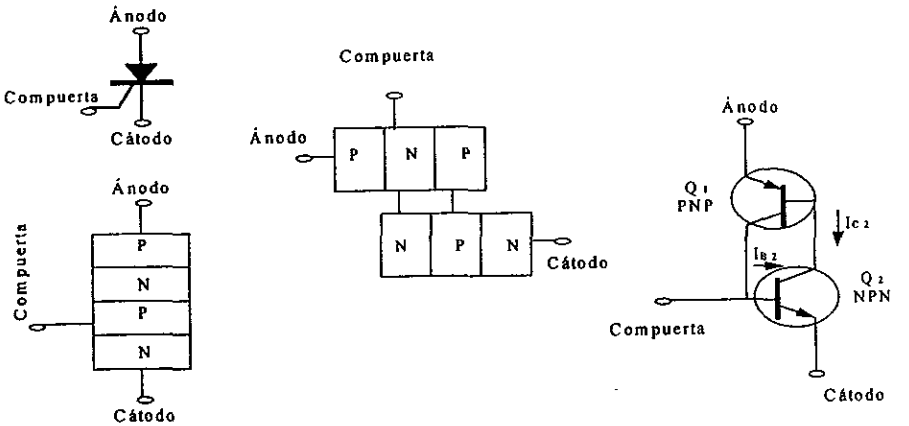


FIG. 1.27 - SCR's

Podemos ver que cuando el transistor npn está apagado (no hay corriente en la compuerta del dispositivo) el otro transistor, pnp, también lo está, ya que la corriente de la base del transistor pnp es abastecida por el colector del transistor npn, este modo en el dispositivo se denomina de *bloqueo directo*.

Pero cuando aplicando un voltaje positivo al ánodo se aplica una corriente en la compuerta, el transistor npn comenzará a conducir corriente, accionando la base del transistor pnp y debido a la interconexión el efecto es regenerativo y cada transistor lleva al otro a saturación, así el SCR permanece encendido aunque la corriente de compuerta se lleve a cero.

Para apagar el SCR, debe producirse algún tipo de interrupción en la fuente de tensión del ánodo o del cátodo, que provoque que las corrientes dentro de él no sean suficientes *para mantenerlo encendido.*

A veces es posible encender un SCR incluso sin que haya corriente de compuerta, por ejemplo si la temperatura del dispositivo aumenta y la corriente de fuga através del SCR alcanza la corriente de encendido, otro ejemplo es cuando el voltaje del ánodo se incrementa más allá del valor máximo de bloqueo y la unión del bloqueo entra entonces en avalancha y la corriente disrruptiva es suficiente para encender el dispositivo.

Un SCR es capaz de controlar la corriente solo en una dirección (cuando el voltaje del ánodo es positivo), un TRIAC nos proporciona control bidireccional de la corriente con una compuerta que puede ser accionada por señales de cualquier polaridad. El TRIAC se constituye, esencialmente como dos SCR's en antiparalelo con una sola compuerta y su símbolo se muestra a continuación (fig.1.28):

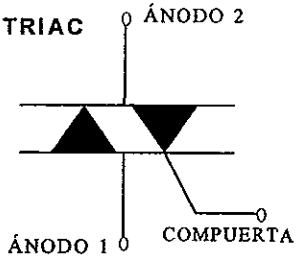


FIG. 1.28 - TRIAC

1.3.3.- ACOPLAMIENTO ÓPTICO, OPTOACOPLADORES U OPTOAISLADORES

Los usamos cuando queremos acoplar dos circuitos eléctricos sin hacer conexiones eléctricas, ésto es con el objeto de que si falla un componente (se daña, ocurre un corto circuito, etc) no se dañen más partes del circuito aislado.

Este tipo de dispositivos son muy útiles en el acoplamiento de señales y son los fototransistores, fotodiodos y fototriacs, debido a que permiten el aislamiento eléctrico de las señales de alta potencia que pueden provocar daño al sistema de control (PC). Este tipo de dispositivos los encontramos encapsulados dentro de un circuito integrado (CI) y nos permiten aislamientos de hasta 7KV. De esta manera el circuito de control queda protegido aún cuando puedan existir fallas externas, por ejemplo: cables en corto circuito, alimentación incorrecta de la máquina y fallas humanas.

En estos dispositivos, el trayecto de la luz, de emisor a detector, está totalmente encerrado en el componente y no se pueden modificar en forma externa. El grado de aislamiento eléctrico entre los dos dispositivos es controlado por los materiales en el trayecto de la luz y por la distancia física entre el emisor y el detector. A mayor distancia mejor aislamiento.

El diodo emisor de luz o LED (Light Emitting Díode), es capaz de cambiar una fuente de energía eléctrica en una fuente de energía lumínica, ésto es posible ya que al hacerse la recombinación de electrones en el material en vez de liberarse energía en forma de calor se producen fotones.

El fotodiodo realiza la función inversa el LED, transformando la fuente de energía lumínica en corriente eléctrica.

Un fototransistor es un transistor en el que se hace incidir luz sobre la unión base-colector, esta luz actúa sobre la base inyectando portadores a la unión (generando la corriente de base) así la corriente de colector se amplifica en base a la corriente fotoeléctrica que recibe.

Un fototriac, recibe de igual forma una fotocorriente para su funcionamiento.

En nuestro caso debido a que no es recomendable unir tierra de corriente directa con el común de corrienta alterna, y como mencionamos las señales de control son señales de corriente directa y para protección requieren un aislamiento eléctrico, éste se logró usando fotoTRIAC´s. y LED´s infrarrojos.

1.4.- SENSORES Y SEÑALES DE ALARMA

Al estar trabajando con máquinas automatizadas es muy importante tomar en cuenta algunas señales de realimentación, para detectar si la máquina se está desplazando dentro de un área válida sin dañar ninguna de sus partes ni otro dispositivo o persona, con la constante lectura de esas señales podemos determinar si existe o no una condición de error y detener el proceso que se esté realizando.

A continuación describiremos algunos de los elementos y dispositivos que se utilizan para sensar la presencia de esas señales.

1.4.1.- INTERRUPTORES DE LÍMITE ELECTROMECÁNICO

Existe una gran variedad de interruptores de límite electromecánico. Los materiales que se usan en su construcción pueden ser zinc, aluminio o acero resistentes a la corrosión y al agua (fig.1.29).

Al ser activado el interruptor cierra o abre un circuito que se utiliza para indicar que se detectó un límite o una posición determinada.

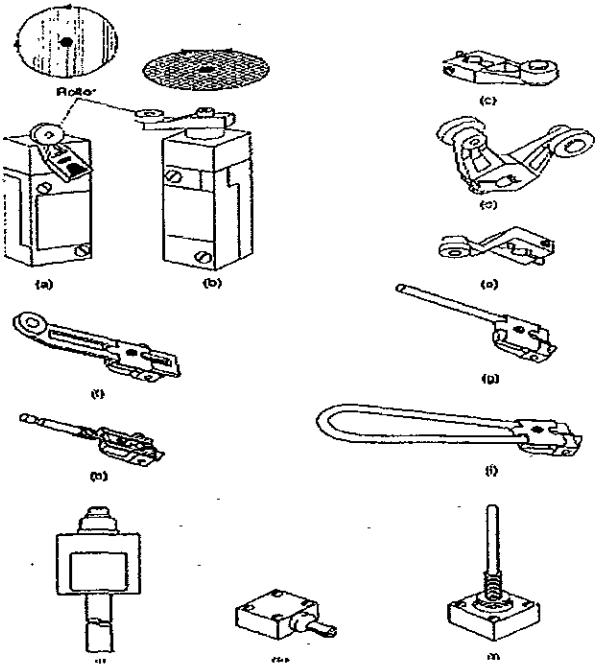


FIG. 1.29 - MICROINTERRUPTORES

1.4.2.- SENSORES DE PROXIMIDAD

Estos sensores no necesitan tener contacto físico con el objeto para activarse.

1.4.2.1.- INDUCTIVOS

Los primeros sensores que utilizaban esta tecnología detectaban solo el hierro, hoy en día se pueden detectar otros materiales como *aluminio, latón, cobre, aceros* y cualquier material ferromagnéticos. Estos sensores generan un campo sensible y pueden detectar la presencia cercana de algún metal (figuras 1.30 y 1.31).

En el sensor se encuentra un circuito tanque **LC** (la parte de inductancia **L** del circuito tiene una bobina con núcleo de aire o de ferrita) el cuál es excitado por un amplificador de realimentación positiva que lo mantiene oscilando, generando una onda senoidal que puede variar en amplitud *dependiendo si se le acerca un material metálico*, con la presencia de ese material enfrente de la bobina del oscilador se generan corrientes parásitas en la superficie metálica, ésto causa que la amplitud de las ondas se decremente o desaparezca, este cambio de amplitud, que es enviado a un sensor analógico o a un detector de nivel, da información de presencia o ausencia del material.

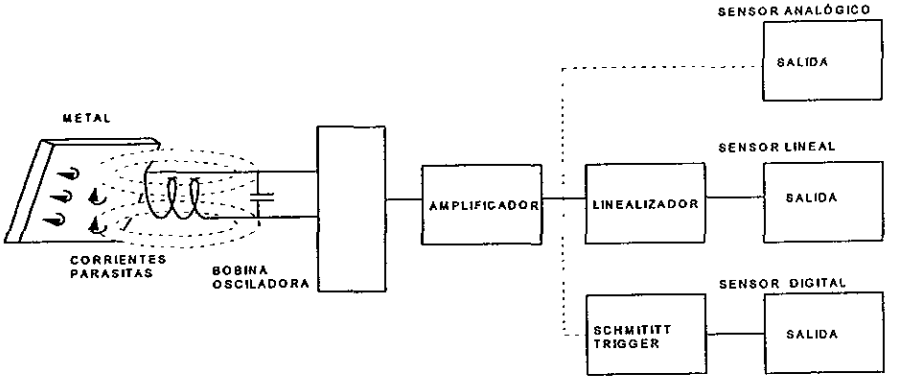


FIG. 1.30 - SENSOR INDUCTIVO

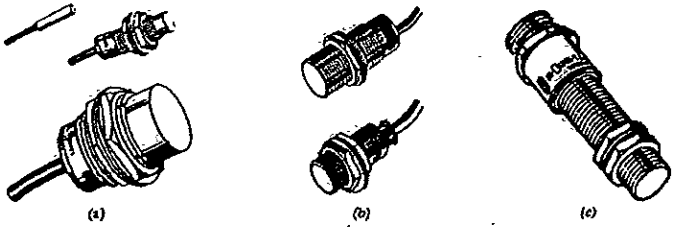


FIG. 1.31 - SENSORES INDUCTIVOS COMERCIALES

1.4.2..- CAPACITIVOS

Estos sensores, producen un campo eléctrico oscilante que es sensitivo y detecta un cambio por masa del dieléctrico entre los electrodos. Pueden detectar muchos materiales como líquidos, metal, plástico, porcelana, cerámica, madera, aceite, etc. (figuras 1.32, 1.33 y 1.34)

La capacitancia está dada por:

$$C = D \times A / d$$

donde A = área del electrodo
D = constante dieléctrica del material entre electrodos
d = distancia entre electrodos

cuando no existe otro material entre los electrodos D del aire = 1.

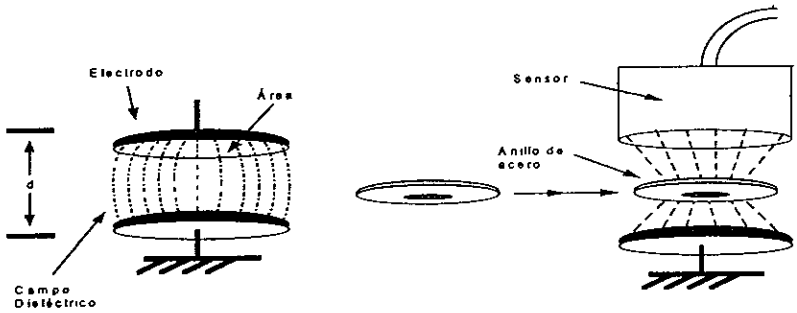


FIG. 1.32 - SENSORES CAPACITIVOS

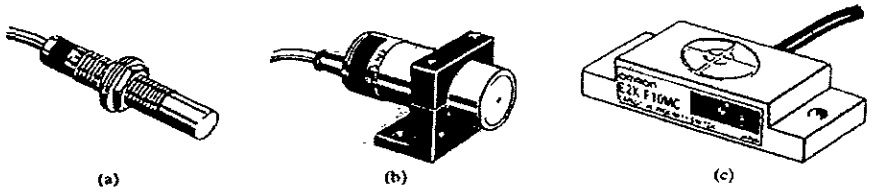


FIG. 1.33 - EJEMPLO DE SENSORES CAPACITIVOS COMERCIALES

FIG. 1.34 - SENSORES CAPACITIVOS E INDUCTIVOS

1.4.2.4.- ULTRASÓNICOS

Los sensores ultrasónicos caen en dos categorías:

- ***los de Resonancia***, los cuales producen un señal de onda continua de frecuencia modulada,
- ***los de eco de pulsos***, los cuales operan como un sistema sonar, donde se evalúa el tiempo requerido para que la señal viaje entre el sensor y el objeto destino para calcular la distancia.

Muchos sensores usan un transductor el cuál sirve como un emisor y receptor. Por definición, *cualquier sonido* excediendo los 20,000 Hz. es considerado en el rango ultrasónico, un transductor ultrasónico industrial operará aproximadamente en 215,000 Hz.

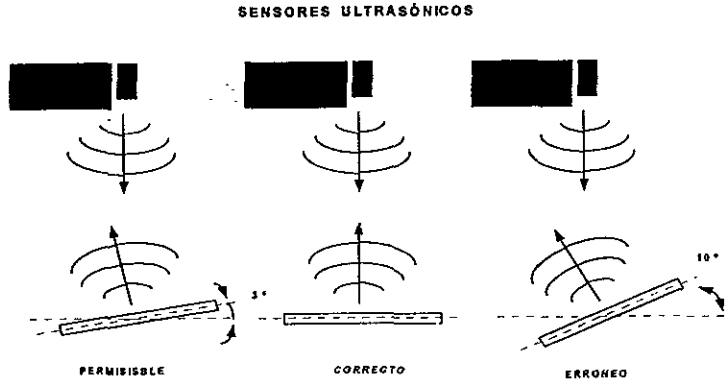


FIG. 1.37 - ÁNGULOS PERMISIBLES

Para que este tipo de sensores funcione bien, no debe haber ángulos de inclinación mayores a 3 grados, pues parte de la señal de regreso se pierde y la medición real sería en este caso decrementada y por lo tanto errónea (fig.1.37).

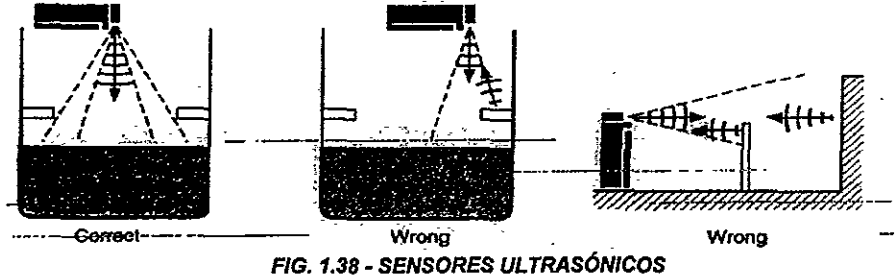
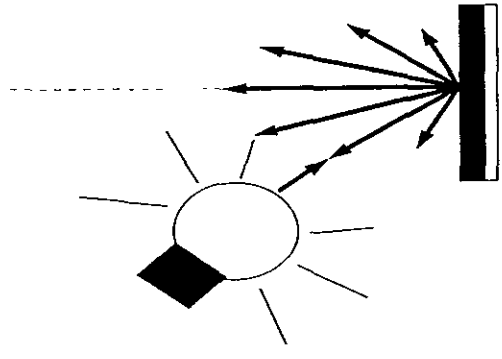


FIG. 1.38 - SENSORES ULTRASÓNICOS

También se debe evitar la presencia de otras superficies que reflejen antes de tiempo el sonido o que absorban parte de él (fig.1.38).

1.4.2.5.- FOTOELÉCTRICOS

Los rayos emitidos por una fuente de luz viajan en líneas rectas. Cuando esos rayos chocan con un objeto, son reflejados por él. Dependiendo de las características y estructura de la superficie del objeto, los rayos se esparcen en varias direcciones, no solamente regresan directamente a la superficie, en base a ésto existen sensores fotoeléctricos (fig.1.39).



EFECTO FOTOELÉCTRICO

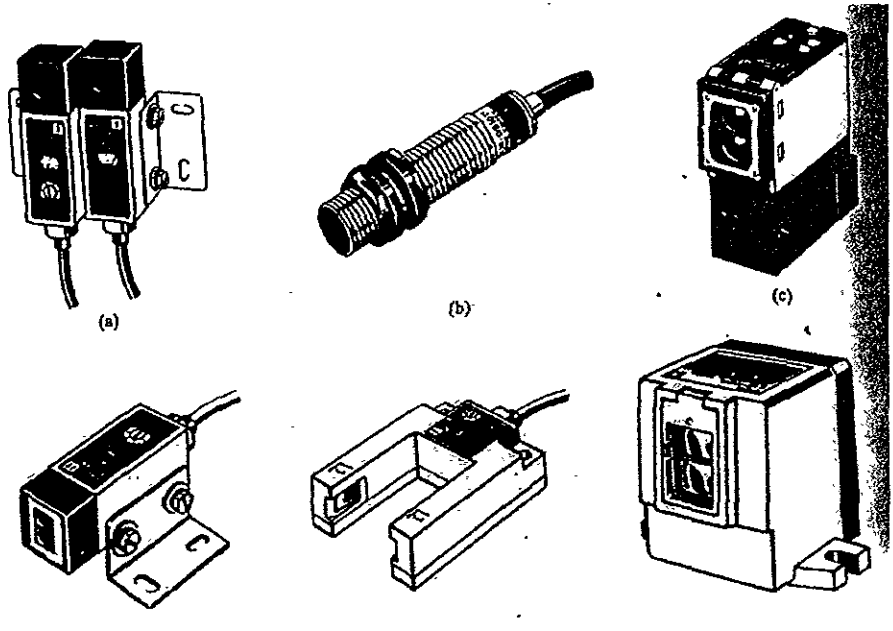


FIG. 1.39 - SENSORES FOTOELÉCTRICOS

CAPÍTULO 2

DISEÑO

DE LAS INTERFACES

CAPÍTULO 2 - DISEÑO DE LAS INTERFACES

El torno actualmente opera, además de su alimentación de aire, con la activación (encendido o apagado) de electroválvulas y contactores. Por lo anterior en nuestro control necesitamos poder manejar señales digitales de alta potencia.

La computadora es capaz de entregarnos señales digitales (0 - 5 Vcd) que es necesario acoplar a las corrientes y voltajes manejados por el torno mediante una etapa de potencia, pero además al usar la computadora podemos aprovecharla para que ella misma (su microprocesador) ejerza directamente el control de la máquina y no exista otro dispositivo como un PLC o un microcontrolador que lo haga ya que podría resultar menos económico al tener que diseñar, aparte, una interfaz para que el operador (usuario) pueda programar la secuencia deseada en la máquina.

Para lograr nuestro objetivo de controlar el torno con la PC desarrollamos el diseño de dos tarjetas (interfaces), una para la interfaz de potencia y otra para el acoplamiento con la PC, aparte del programa de control (fig. 2.1).

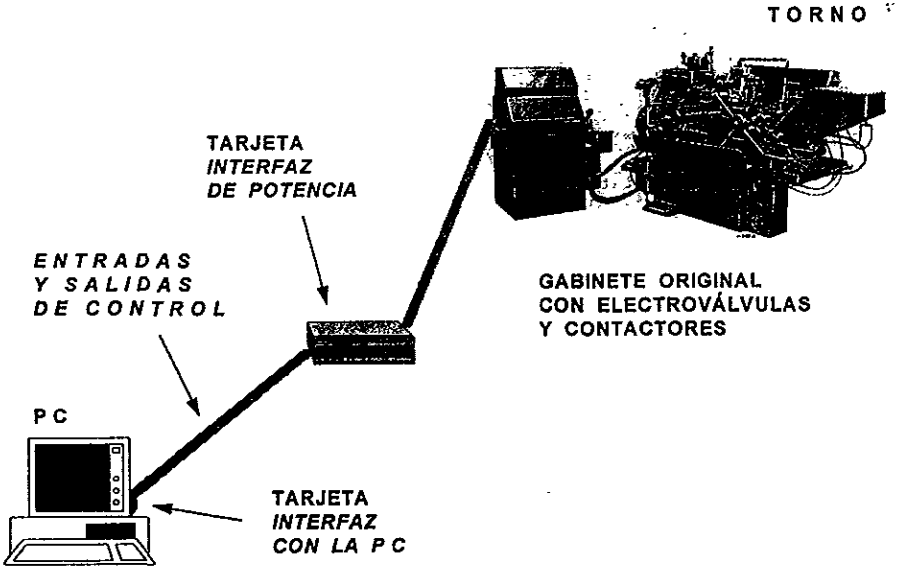


FIG. 2.1 - ESQUEMA DEL DISEÑO PARA EL CONTROL.

2.1.- DISEÑO DE LA INTERFAZ DE POTENCIA

Las salidas de control de la PC son las entradas para la etapa de potencia (conectada al gabinete del torno), y las entradas de control a la PC son el resultado leido de la tarjeta con la etapa de potencia dados por el sensado de microinterruptores que energizan un circuito de corriente directa.

Tomando en cuenta lo anterior la tarjeta con la etapa de potencia (fig. 2.4) recibirá y entregará a la PC señales de 0 - 5 V_{CD}, por un lado y por el otro manejará 127 V_{CA} para energizar a las electroválvulas y contactores, además de los 12 V_{CD} para los microinterruptores, por lo tanto dentro de ella y por seguridad usaremos dispositivos que nos permitan hacer un acoplamiento óptico (aislamiento) .

Las salidas de la PC (0 - 5 V_{CD}), con sus corrientes pequeñas, nos ayudan a controlar corriente alterna mediante un acoplamiento usando un LED y un fototriac, de esta forma cuando en la PC hay como salida un "1" lógico el LED activá un fototriac que a su vez disparará otro triac de potencia y éste energiza una electroválvula (a su solenoide correspondiente) o el contactor correspondiente generando la activación de alguna parte del torno, y cuando existe un "0" lógico el LED estará apagado, y al no haber un voltaje y una corriente que mantenga encendido el primer fototriac éste se apaga y apaga al triac que a su vez deja de activar la electroválvula o el contactor (fig. 2.2).

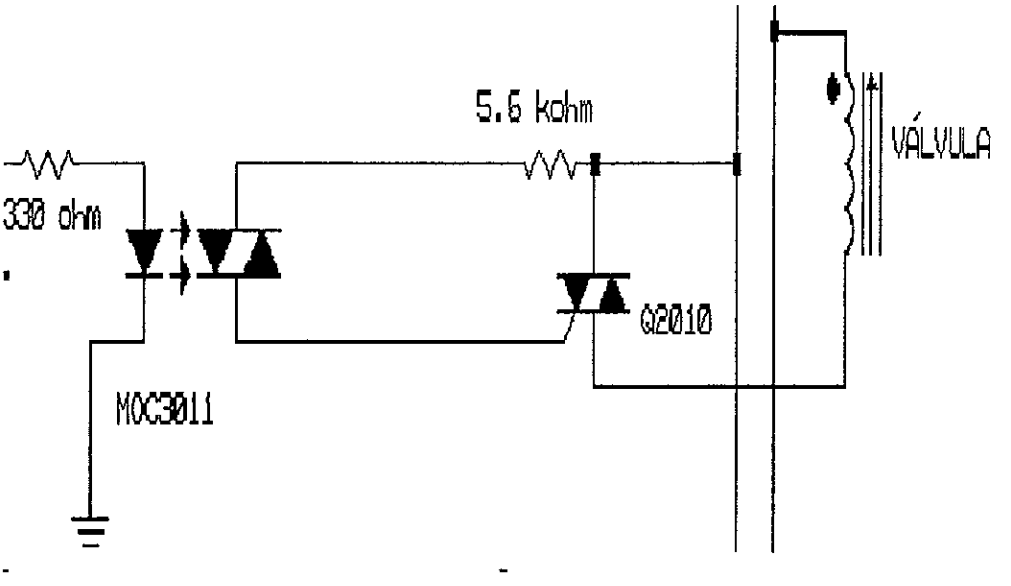


FIG. 2.2 - ACOPLAMIENTO ÓPTICO DE LA ETAPA DE POTENCIA

Los dispositivos usados aquí son:

- ***MOC3011*** - Consiste de un diodo emisor infrarrojo opticamente acoplado a un fototriac del cuál las características más importantes son:

VAISLAMIENTO (VISO)	7500VCA
VRUPTURA (VDRM)	250V
Máxima Frecuencia de operación (dv/dt)	10V/µseg
Corriente de salida (ITSM)	10mA

- ***TRIAC Q2010*** - cuyas características son:

VDRM	600V
Corriente de salida (IT)	10A
Corriente de disparo (IGT)	50mA
Máxima Frecuencia de operación (dv/dt)	50V/µseg

Las entradas a la PC, también señales de 0 - 5 VCD y pequeñas corrientes, las obtenemos a partir de un circuito de 12 VCD que se energiza al cerrarse algún microinterruptor en el torno ésto produce que un LED se encienda y active un fototransistor que nos ayuda a obtener un "0" lógico si el circuito está energizado o un "1" lógico si el microinterruptor está abierto (fig. 2.3).

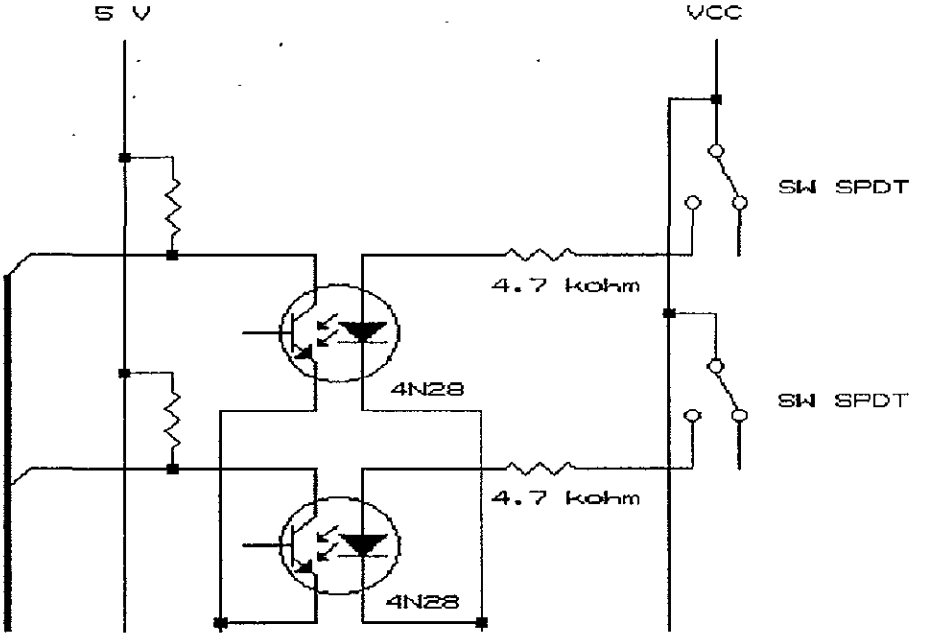
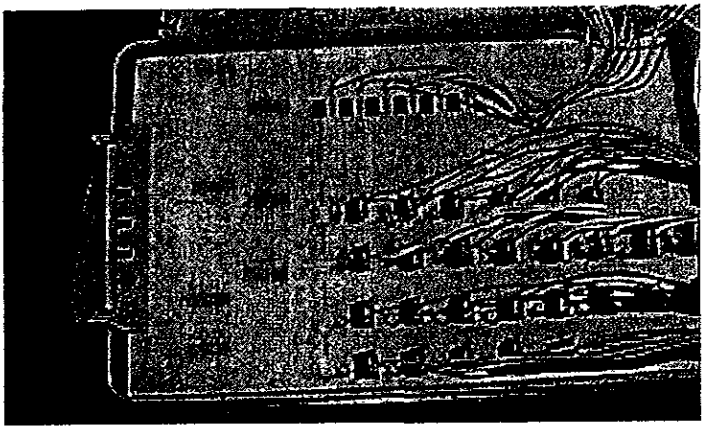


FIG. 2.3 - ACOPLAMIENTO ÓPTICO EN LA ETAPA DE POTENCIA

Los dispositivos usados aquí son:

- ***4N28*** - Consiste de un diodo emisor infrarrojo opticamente acoplado a un fototransistor del cuál las características más importantes son:

IFORWARD (IF)	60mA
Ic	150mA
VCEO	30V
VAISLAMIENTO (VISO)	7500VCA
Voltaje de encendido (VF)	1.15V
Tiempo de encendido (Ton)	2.8µseg
Tiempo de apagado (Toff)	4.5µseg



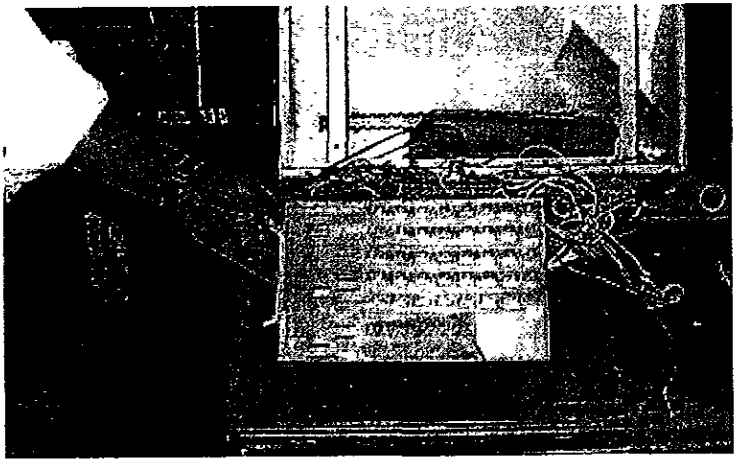


FIG. 2.4 - INTERFAZ DE POTENCIA

2.2.- DISEÑO DE LA INTERFAZ CON LA PC.

La computadora, donde reside el programa para controlar el torno, necesita poder enviar (salidas de control) y recibir (entradas de control) datos mediante algún dispositivo de entrada y/o salida (I/O).

Estos datos o señales digitales (0 - 5 Vcd) se pueden obtener y/o generar desde el programa usando algún puerto ya definido en la computadora, como el serial (donde necesitamos manejar este tipo de transmisión) o el paralelo (limitado en número de bits a usar como entradas y como salidas).

Debido al número de entradas y salidas de control necesarias para manejar las señales del torno, en este caso 16 y 40 respectivamente, se consideró más adecuado la elaboración de una tarjeta para manejarlas, utilizando las direcciones de algunos puertos físicos no utilizados.

2.2.1.- DETERMINACION DE LOS RECURSOS Y SEÑALES DEL SISTEMA A USAR.

Al decidir elaborar nuestra interfaz de entrada/salida (I/O) a la PC optamos por diseñar una tarjeta basada en el bús tipo ISA de 8 bits, así como manejar un mapeo de I/O aislado (usando puertos físicos de I/O) eligiendo un rango de direcciones para nuestra tarjeta.

Lo primero que analizamos durante el diseño fue que señales nos eran útiles de las que se presentan en el bús ISA al escribir y al leer información de una dirección elegida por nosotros como puerto físico.

Al realizar una escritura al puerto físico elegido, con el osciloscopio pudimos observar que además de la dirección (A0 - A15) y los datos enviados (D0 - D7) se presenta la señal IOW (I/O Write, fig. 2.5).

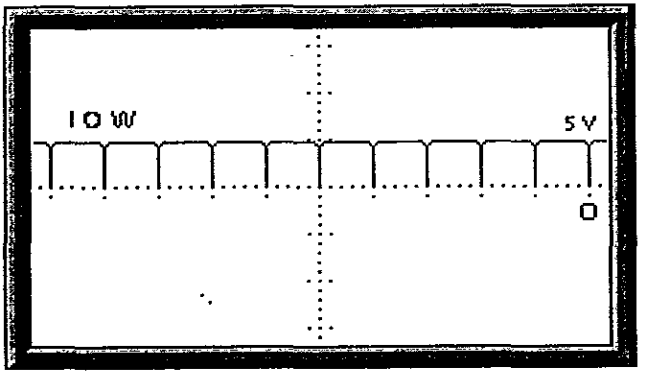


FIG. 2.5 - GRAFICA DE OSCILOSCOPIO - IOW

De la misma forma al realizar una lectura de algún puerto físico, podemos ver que además de estar presente la dirección, la computadora captura la información del bús de datos al presentarse la señal IOR (I/O Read, fig. 2.6).

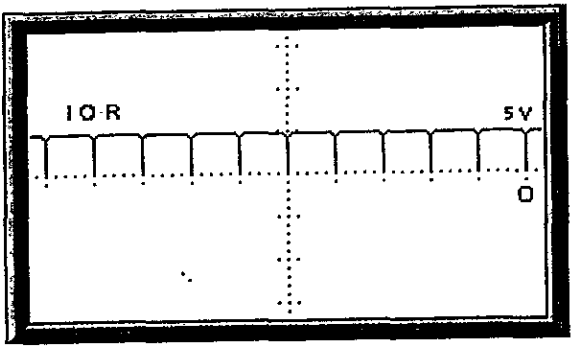


FIG. 2.6 - GRAFICA DE OSCILOSCOPIO - IOR

Así pues estas dos señales nos sirven para capturar y liberar información del y al bús de datos en el momento adecuado para no generar un corto circuito u otro problema a la computadora y llegar a dañarla.

Con estas dos señales (IOW e IOR) , las direcciones (A0 - A15) , los datos (D0 - D7) , el voltaje (5 V_{CD}) y la tierra (0V) diseñamos la tarjeta.

2.2.2- ASIGNACIÓN DE DIRECCIONES DE LOS PUERTOS

La tarjeta maneja tanto entradas como salidas, éstas están agrupadas de 8 en 8, para cada grupo de 8 bits (1 byte) se asoció una dirección del mapa de direcciones de I/O aislado de la PC que no es utilizada por ningún otro dispositivo.

Se elegió manejar el rango de direcciones $F000 a $FFFF con el circuito decodificador de direcciones y combinando ésto con las señales de lectura (IOR) y escritura (IOW) al bús, designamos 5 direcciones para los dispositivos de salida y 2 direcciones para los dispositivos de entrada (Tabla 2.1).

LATCHES	DIR. SALIDAS	DIR. ENTRADAS	BUFFERS
L1	$F000	$F005	B1
L2	$F001	$F006	B2
L3	$F002		
L4	$F003		
L5	$F004		

TABLA 2.1

2.2.2.1.- DECODIFICACIÓN DE DIRECCIONES.

Tomando las direcciones (A0 - A15) y las señales IOR e IOW, usamos dos decodificadores 74HC138 y algunas compuertas NAND (74HC08) para generar las señales que activan los dispositivos de entrada y los de salida, quedando nuestro circuito de decodificación *de la siguiente forma* (fig. 2.7):

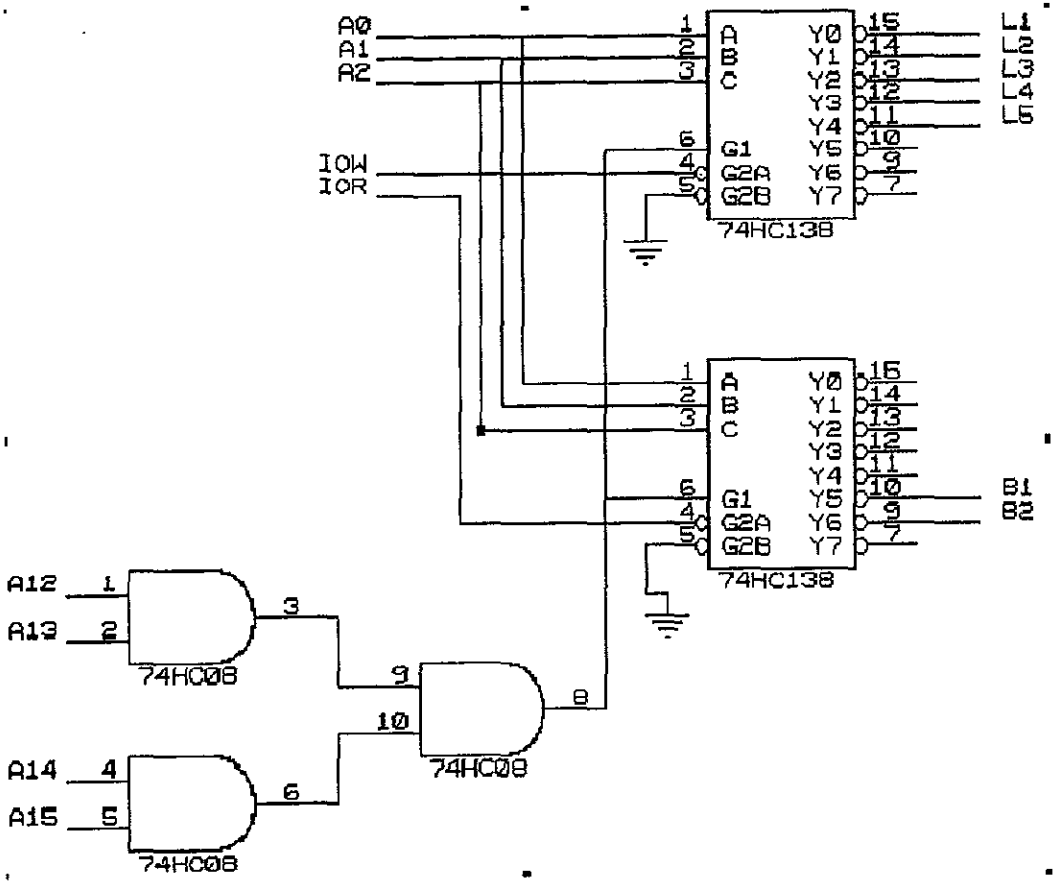


FIG. 2.7 - CIRCUITO DECODIFICADOR DE DIRECCIONES

Las 5 primeras salidas del primer decodificador activan los latches (de L1 a L5) y 2 salidas del segundo decodificador activan los bufers (B1 y B2).

2.2.3.- DISPOSITIVOS DE ENTRADA Y SALIDA.

Los dispositivos de entrada son buffers (3 edos) que permiten que el dato esté conectado al bús de datos solamente durante el tiempo indicado por el decodificador (cuando la PC hace la lectura) y los dispositivos de salida son latches que capturan el dato de salida en el momento en que éste aparece en el bús (cuando la PC realiza la escritura). Las señales que activan los buffers y latches provienen de los decodificadores 74HC138.

Con estos dispositivos añadimos una arquitectura a la tarjeta de la siguiente forma (fig. 2.8):

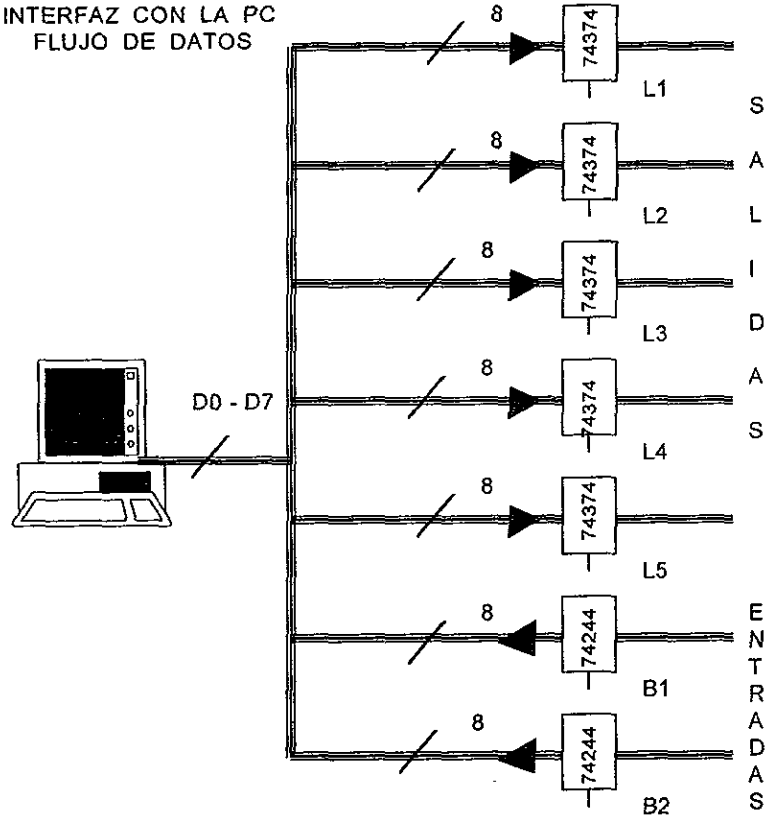
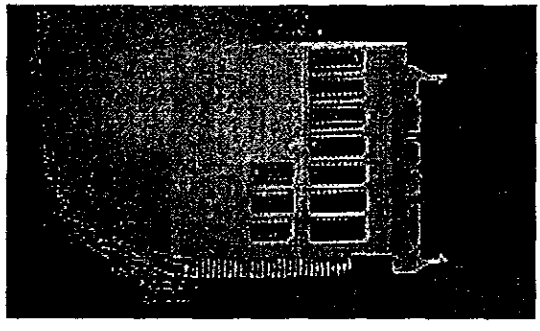


FIG. 2.8 - FLUJO DE DATOS

Finalmente la tarjeta (fig. 2.9), mediante un cable plano, envía en *42 líneas, 40 bits de salidas* y 2 bits con los voltajes (Vcc y tierra), y recibe en otras 16 líneas 16 bits de entradas para el control.

Para mayor *detalle de las arquitecturas* de ambas interfaces ver el apéndice C "Diagramas de los circuitos de las interfaces".



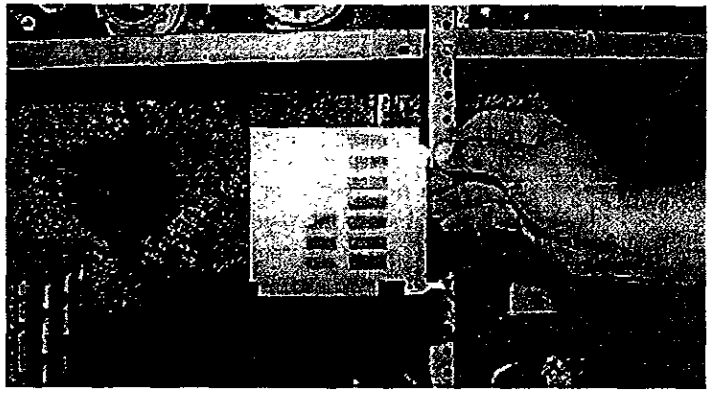


FIG. 2.9 - TARJETA DE INTERFAZ CON LA PC

CAPÍTULO 3

DISEÑO DEL SISTEMA DE PROGRAMACIÓN

CAPÍTULO 3 - DISEÑO DEL SISTEMA DE PROGRAMACIÓN.

Al utilizar una computadora para el diseño de nuestro control, contamos con grandes beneficios como el poder elegir un lenguaje de programación adecuado para manejar la comunicación con los dispositivos de I/O, poder manipular la información fácilmente, poder diseñar nuestra interfaz con el usuario (operador) y tener conectividad con otros sistemas, entre otras cosas.

En el torno que usamos, anteriormente, se creaba una secuencia de programación (programa para la pieza a elaborar) que constaba de una serie de pasos dentro de los cuales en cada uno se indicaban que movimientos o funciones se iban a realizar, el tablero donde se marcaban estos pasos (fig. 3.1), al irlos ejecutando mandaba señales al gabinete de control para activar o no las electroválvulas y contactores.

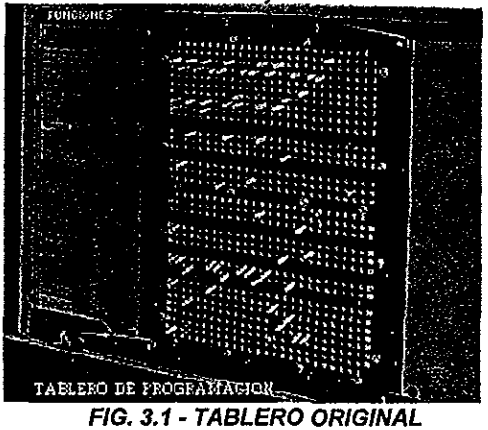


FIG. 3.1 - TABLERO ORIGINAL

Una vez que estaba listo y probado el programa en el torno debía escribirse sobre una plantilla (hoja) de papel, con los mismos renglones (funciones) y columnas (pasos) que el tablero original para poder volver a marcarlo en el tablero si se iba a usar posteriormente, este erá un mapa de programación (fig. 3.2).

FIG. 3.2 - MAPA DE PROGRAMACIÓN DEL TORNO

En otros tornos observados se procede de manera similar, por ejemplo en un torno MTA el tablero de control es electrónico (fig. 3.3) y se van grabando las selecciones hechas con los interruptores y perillas de control para que éstas formen un paso, se cambian las selecciones y se graba otro paso hasta tener el número de pasos o la secuencia de programación deseada para después ejecutarla automáticamente, ya sea una vez o en un ciclo repetitivo. Aquí el programa es almancenado en una memoria RAM y se pierde si se apaga la máquina y la batería llega a descargarse.

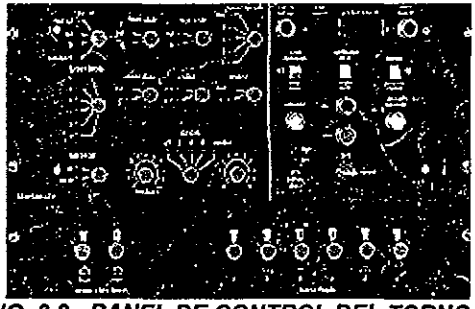


FIG. 3.3 - PANEL DE CONTROL DEL TORNO MTA.

En algunos tornos el programa se grababa en una cinta, para después recuperarlo y ejecutarlo.

En todas estas máquinas la operación del control es similar: definir en uno o varios pasos las funciones a ir ejecutándose para ir elaborando una pieza en el torno.

Los tornos tienen movimientos comunes para su operación, algunos tienen funciones de más u otras de menos pero su funcionamiento o secuencias de operación son similares, ésto nos ayuda en el diseño de nuestro programa de control.

ELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN.

Dentro de la amplia gama de lenguajes que actualmente existen se consideró entre los más adecuados para el desarrollo del programa a Pascal y 'C' debido a que son muy flexibles y amplios a la vez y se puede programar con ellos tanto la parte de comunicación a la tarjeta de interfaz como la interfaz con el usuario.

Se hace el desarrollo en lenguaje 'C' para aprovechar la experiencia en el manejo de éste que ya se tiene, así como sus facilidades para el desarrollo de programas de interface y manejo de información.

3.1.- ACCESO AL BÚS DE DATOS Y MANEJO DE INFORMACIÓN.

La tarjeta de interfaz con la PC es capaz de colocar y/o tomar información del bús de datos cuando se presentan las señales IOW e IOR en el bús de control. Recordando del *capítulo 1*, estas señales las generan las instrucciones IN, INS, OUT y OUTS del conjunto de instrucciones del microprocesador y también las generan las funciones inport(), inportb(), outport() y outportb() disponibles en compiladores del lenguaje 'C' (Tabla 3.1).

FUNCIONES PARA LEER Y ESCRIBIR A	PUERTOS FÍSICOS
int **inport** (int *portid*)	Lee una palabra del puerto indicado
unsigned char **inport** (int *portid*)	*Lee un byte del puerto indicado*
void **outport** (int *portid*, int *value*)	Escribe una palabra al puerto indicado
void **outportb** (int *portid*, unsigned char *value*)	*Escribe un byte al puerto indicado*

TABLA 3.1

Los *datos (información de control)* los agrupamos en bytes de control (8 bits) ya que la tarjeta maneja 8 bits a la vez, estos 8 bits podemos representarlos en distintos tipos de datos manejados por el lenguaje elegido (Tabla 3.2).

TIPO DE DATO EN LENG. 'C'	MEMORIA USADA	RANGO REP. DECIMAL	REP. HEXADECIMAL
char	1 byte	-128 a +127	00 - FF
unsigned char	1 byte	0 a +255	
short int	1 byte	-128 a +127	00 - FF
unsigned short int	1 byte	0 a +255	
int	2 bytes	-32768 a +32767	0000 - FFFF
unsigned int	2 bytes	0 a +65535	

TABLA 3.2

3.1.1.- ESCRITURA A UN PUERTO FÍSICO

Para escribir información a un puerto físico (ésto implica colocar la información deseada en el bús de datos y generar la señal IOW) debemos poner la información deseada en un tipo de dato y mandarla a la dirección de ese puerto, por ejemplo, si tenemos un dato de longitud de un byte y deseamos escribir unos (1's) al puerto con la dirección F000 en hexadecimal hay que realizar la siguiente secuencia:

```
.
.
unsigned char dato_sal          /* declaración de la variable para poner
.                                  el dato */
.
dato_sal=0xFF;                  /* pone 1´s en los 8 bits del dato */
.
.
outportb($F000,dato_sal);       /* escribe 1´s al puerto F000 */
```

3.1.2.- LECTURA DE UN PUERTO FÍSICO

Para leer información a un puerto físico (ésto implica leer información del bús de datos en el momento en que se genera la señal IOR) indicamos la dirección del puerto deseado y obtenemos en una variable la información (dato) que contiene ese puerto, por ejemplo, para leer un byte del puerto con la dirección FF05 en hexadecimal hay que realizar la siguiente secuencia:

```
.
.
unsigned char dato_ent          /* declaración de la variable para guardar
.                                  el dato */
.
dato_ent=inportb($FF05);        /* lee el byte del puerto FF05 y lo deja en la
                                      variable dato_ent */
```

3.2.- GENERACIÓN DE LOS ALGORITMOS DE CONTROL

Al tener los elementos necesarios para comunicarnos con el torno (interfaces y programa), hace falta definir los algoritmos de control, es decir, ¿qué queremos controlar?, ¿qué comportamiento queremos obtener del torno y bajo que condiciones?, para ésto primero analizaremos bajo que principios funciona el torno y de que elementos disponemos para su operación.

3.2.1.- DESCRIPCIÓN LÓGICA DE LA OPERACIÓN DEL TORNO

El tablero de control original enviaba señales eléctricas que activaban las electroválvulas y/o contactores para mover alguna parte del torno.

Para este desarrollo contamos con un diagrama de los "Circuitos de Aire" (fig. 3.4) del torno, indicando el flujo que el aire puede seguir y las electroválvulas que existen en él. Al ver que electroválvula se activa se vé que función o movimiento activará el torno.

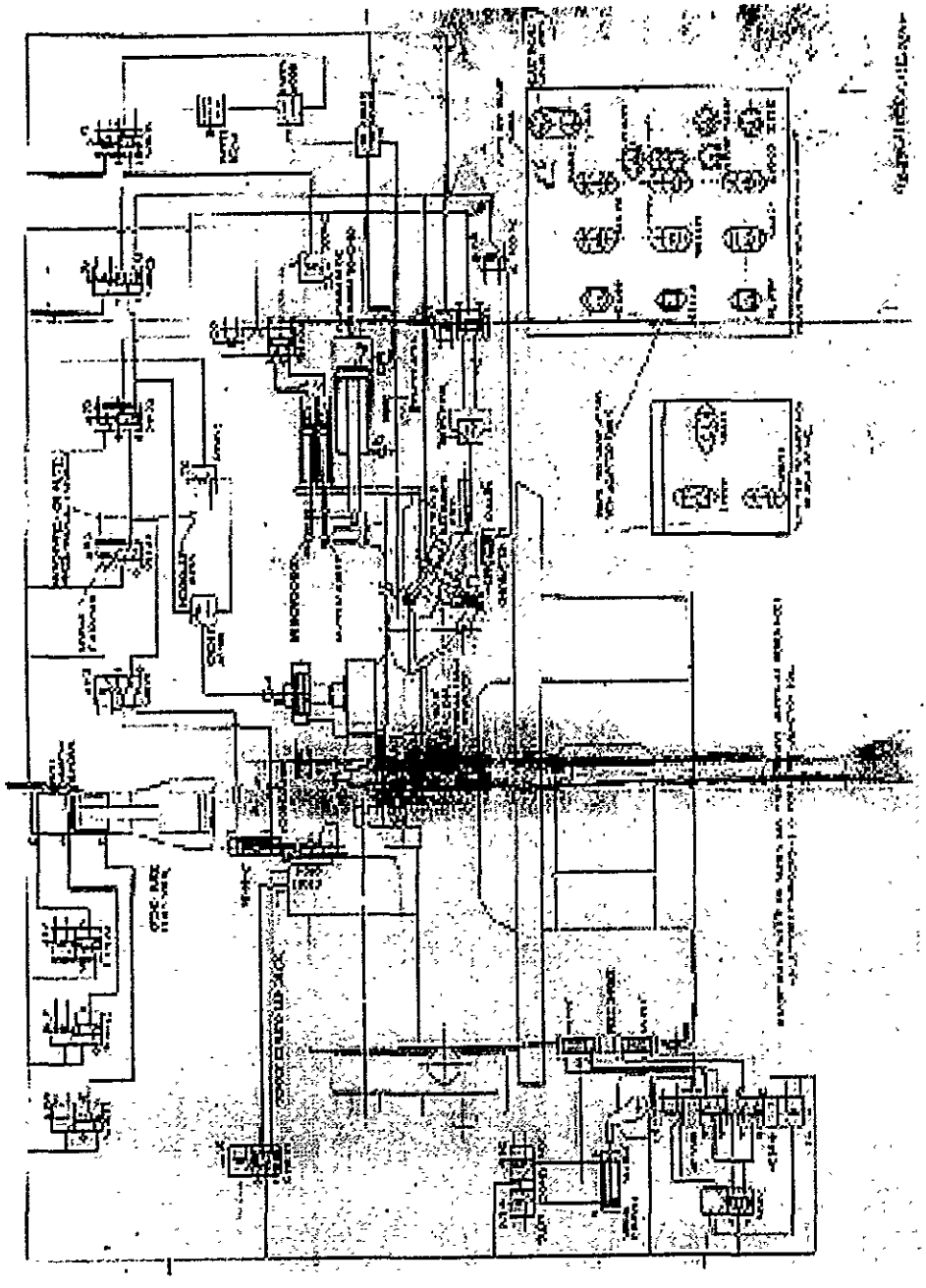


FIG. 1.52 - CIRCUITOS DE AIRE Y ELECTROVÁLVULAS DEL TORNO

En el gabinete de conexiones eléctricas del torno hay una tablilla numerada donde al mandarle señales eléctricas se activan las electroválvulas o contactores mencionados y provocan que se realicen las distintas funciones asociadas.

La siguiente es una tabla (3.3) de las electroválvulas y contactores existentes, la identificación que tienen en la tablilla de conexiones y la función que realizan al activarlas.

Electro-Válvula	No. de Conex.	Función que realiza	No. de bit asociado	Dirección de puerto
AS	153	Alimentación neumática gral. del avance rápido	12	$F000
BS	357	*Avance rápido del RAM*	20	$F001
CS	356	Hidros 1 y 2	25	$F001
DS	285	Avance mecánico	22	$F001
ES	290	Selección de avance mecánico (lento)	16	$F000
FS	294	Selección de avance mecánico (rápido)	21	$F001
GS	221	Carro transversal (delantero)	11	$F000
HS	201	Carro transversal (trasero)	10	$F000
JS	200	Carro transversal (trasero)	13	$F000
KS	224	Chiprrupter	24	$F001
MS	B	Engrane, velocidades lentas	14	$F000
NS	503	Engrane, velocidades rápidas	17	$F000
OS	298	Hidro 2	27	$F001
TS	160	Herramienta auxiliar de la torreta	26	$F001
VS	225	Abrir chuck	23	$F001
WS	188	Workcatcher	15	$F000
XS	187	Barrenado intermitente	30	$F002
YS	298	Hidro 2	27	$F001
CONT1	61	Contactor principal para motor	31	$F002
CONT2	125	Motor alimentador atrás	32	$F002
CONT3	154	Motor alimentador adelante	33	$F002
CONT4	360	Giro motor atrás	34	$F002
CONT5	502	Giro motor adelante	35	$F002
CONT6	F	Velocidad 'A'	36	$F002
CONT7	E	Velocidad 'B'	37	$F002
CONT8	D	Velocidad 'C'	40	$F003
CONT9	C	Velocidad 'D'	41	$F003

TABLA 3.3

En la penúltima columna de la tabla anterior se indica un número de bit con el que se identifica (se relaciona), desde el programa de control en la computadora, qué electroválvula o contactor se quiere activar, el primer dígito de esta columna indica un número de byte y el segundo un número de bit. Los bytes usados corresponden a los datos que saldrán por los latches (L1 a L5) de la tarjeta de interfaz .

La última columna de la tabla indica la dirección del puerto físico a la que se debe escribir para colocar la información en los latches.

Combinando la activación de electroválvulas y contactores se obtienen las funciones necesarias que debe realizar el torno para que el usuario al programarlas pueda operar automáticamente la máquina.

La siguiente tabla (3.4) indica, en la primera columna, los números de la tablilla de conexiones que se deben activar al mismo tiempo y en la segunda columna la función (respuesta) que se obtendrá del torno.

Conexiones a activar	Función que se realiza
153,357	Avance de torreta (RAM)
153	Regreso de torreta
153,357,356	Avance con Hidro No. 1
153,357,356,298	Avance con Hidro No. 2
221	Avance del frontal cruzado
200,201	Avance del posterior cruzado
153,188	Avance de Workcatcher
153	Regreso de Workcatcher
225	Abrir Chuck
290	Selección de avance mecánico (lento)
294	Selección de avance mecánico (rápido)
B	Saca embrague mecánico, velocidades lentas
503	Mete embrague mecánico, velocidades rápidas
153,357,285	Selección del avance mecánico
224	Chiprrupter
61	Arranque de contactor principal para motor
125	Arranque de motor alimentador atrás
154	Motor alimentador adelante
360	Giro motor atrás
502	Giro motor adelante
F	Velocidad 'A'
E	Velocidad 'B'
D	Velocidad 'C'
C	Velocidad 'D'

TABLA 3.4

Las funciones anteriores las contiene el programa para que el operador pueda programarlas al igual que otras como los tiempos de duración de dichas funciones (Para más detalle ver el apéndice 'A' "Manual de usuario del programa").

3.2.2.-DESCRIPCIÓN DEL CONTROL SEGUIDO EN LAS FUNCIONES DESDE LA PC

Recordando del capítulo 1, el torno para realizar sus movimientos cuenta con un área en la que puede desplazarse, por ésto si la función que va a realizar es un avance solo avanzará si ni ha llegado al límite de ese desplazamiento, los límites, de este torno, nos lo indican los microinterruptores descritos en la siguiente tabla (3.5):

Microinterruptor	Detecta	No. de bit asociado	Dirección de puerto
LS2	Límite para el avance delantero de la torreta.	15	$F005
LS3	Límite para el avance trasero de la torreta.	13	$F005
LS4	Se debe cambiar el avance usado o activar un tiempo.	14	$F005
LS5	Límite para el frontal del movimiento cruzado.	17	$F005
LS6	Límite para el posterior del movimiento cruzado.	16	$F005

TABLA 3.5

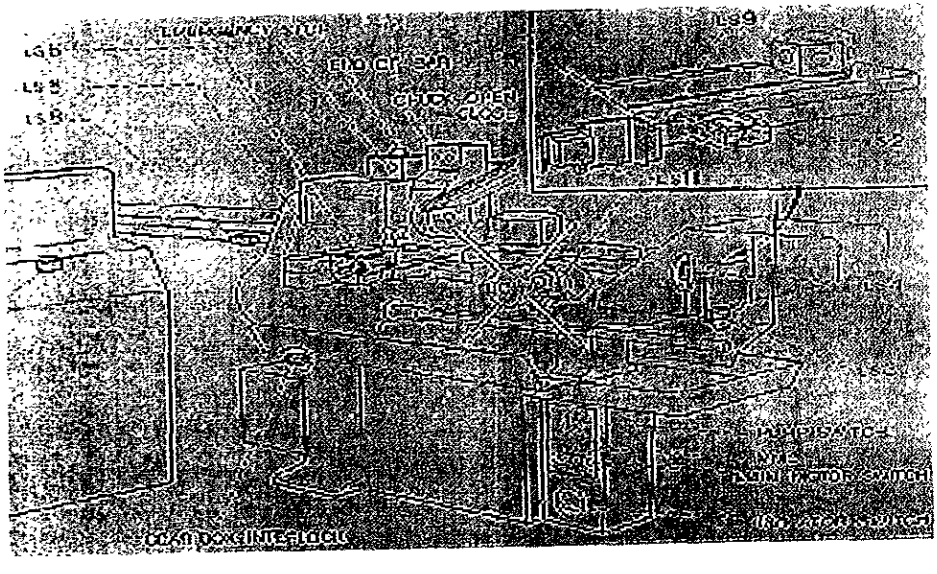


FIG. 3.5 - UBICACIÓN DE LOS MICROINTERRUPTORES

En esta tabla también indicamos él bit asociado y la dirección del puerto donde debemos leer (sensar) el estado de los microinterruptores (fig. 3.5).

Algunas funciones rigen su duración por la detección de estos microinterruptores pero otras, como el encendido de motores, dependen del tiempo de duración que se les programe.

El programa de control enviará señales (escribirá a los puertos) para que la tarjeta de interfaz de potencia con el torno active en la tablilla de conexiones las indicadas para cada función (de acuerdo a la tabla 3.4 vista anteriormente), al mismo tiempo que estará sensando (leyendo de los puertos) los microinterruptores para ver los límites de avance.

A continuación hay una pequeña descripción de lo que el programa hace con cada función disponible para operar el torno:

AVANCE DE TORRETA - envía las señales de activación necesarias hasta que detecta al microinterruptor LS2.

REGRESO DE TORRETA - envía las señales de activación necesarias hasta que detecta al microinterruptor LS4

AVANCE CON HIDRO 1 - envía las señales del AVANCE DE TORRETA y al detectar al microinterruptor LS3 envía las señales de activación necesarias para que se active el HIDRO No. 1 hasta que detecta al microinterruptor LS2.

AVANCE CON HIDRO 2 - envía las señales del AVANCE DE TORRETA y al detectar al microinterruptor LS3 envía las señales de activación necesarias para que se active el HIDRO No. 2 hasta que detecta al microinterruptor LS2.

SELECCIÓN DEL AVANCE MECÁNICO DE LA TORRETA - envía las señales del AVANCE DE TORRETA y al detectar al microinterruptor LS3 envía las señales de activación necesarias para que se active el AVANCE MECÁNICO hasta que detecta al microinterruptor LS2.

SELECCIÓN DE RANGO DE AVANCE MECÁNICO (+ LENTO) - envía las señales de activación necesarias.

SELECCIÓN DE RANGO DE AVANCE MECÁNICO (+ RÁPIDO) - envía las señales de activación necesarias.

AVANCE DEL FRONTAL CRUZADO - envía las señales de activación necesarias hasta que detecta al microinterruptor LS5.

AVANCE DEL POSTERIOR CRUZADO - envía las señales de activación necesarias hasta que detecta al microinterruptor LS6.

AVANCE DEL WORKCATCHER - envía las señales de activación necesarias.

REGRESO DEL WORKCATCHER - envía las señales de activación necesarias.

ABRIR CHUCK - envía las señales de activación necesarias.

CHIPRRUPTER - envía las señales de activación necesarias.

SACA EMBRAGUE MECÁNICO DE LAS VELOCIDADES LENTAS - envía las señales de activación necesarias.

METE EMBRAGUE MECÁNICO DE LAS VELOCIDADES RAPIDAS - envía las señales de activación necesarias.

ARRANQUE DE CONTACTOR PRINCIPAL PARA MOTOR - envía las señales de activación necesarias.

GIRO DE MOTOR ATRÁS **<** -. envía las señales de activación necesarias.

GIRO DE MOTOR ADELANTE **-** **>** envía las señales de activación necesarias.

VELOCIDAD 'A' - envía las señales de activación necesarias.

VELOCIDAD 'B' - envía las señales de activación necesarias.

VELOCIDAD 'C' - envía las señales de activación necesarias.

VELOCIDAD 'D' - envía las señales de activación necesarias.

ARRANQUE DE MOTOR ALIMENTADOR ATRÁS - envía las señales de activación necesarias.

ARRANQUE DE MOTOR ALIMENTADOR ADELANTE - envía las señales de activación necesarias.

TIEMPO_1 - envía una señal para indicar que transcurrió el tiempo programado.

TIEMPO_2 - envía una señal para indicar que transcurrió el tiempo programado.

TIEMPO_3 - envía una señal para indicar que transcurrió el tiempo programado.

TIEMPO_4 - envía una señal para indicar que transcurrió el tiempo programado.

TIEMPO_5 - envía una señal para indicar que transcurrió el tiempo programado.

TIEMPO_6 - envía una señal para indicar que transcurrió el tiempo programado.

Las funciones que solo envían señales dependerán del tiempo de duración que les indiquemos y este tiempo puede ser programado o determinado por la detección del microinterruptor asociado a otra función con la que se combinen, ya que se pueden programar varias funciones para ejecutarse a la vez.

3.2.3.- DIAGRAMA DE CONTROL DESDE LA COMPUTADORA

Durante la ejecución de una secuencia de operación existen funciones que envían solo un conjunto de señales (una sola salida) y otras que envían un primer conjunto de señales (tienen una *SALIDA PARCIAL* implicando un *AVANCE PARCIAL*) y al detectar un *microinterruptor envían un complemento* o cambio de señales para terminar la ejecución de esa función (*AVANCE TOTAL*).

La computadora estará leyendo y escribiendo información a los puertos, dependiendo de las funciones programadas, con un esquema de la siguiente forma (fig. 3.6):

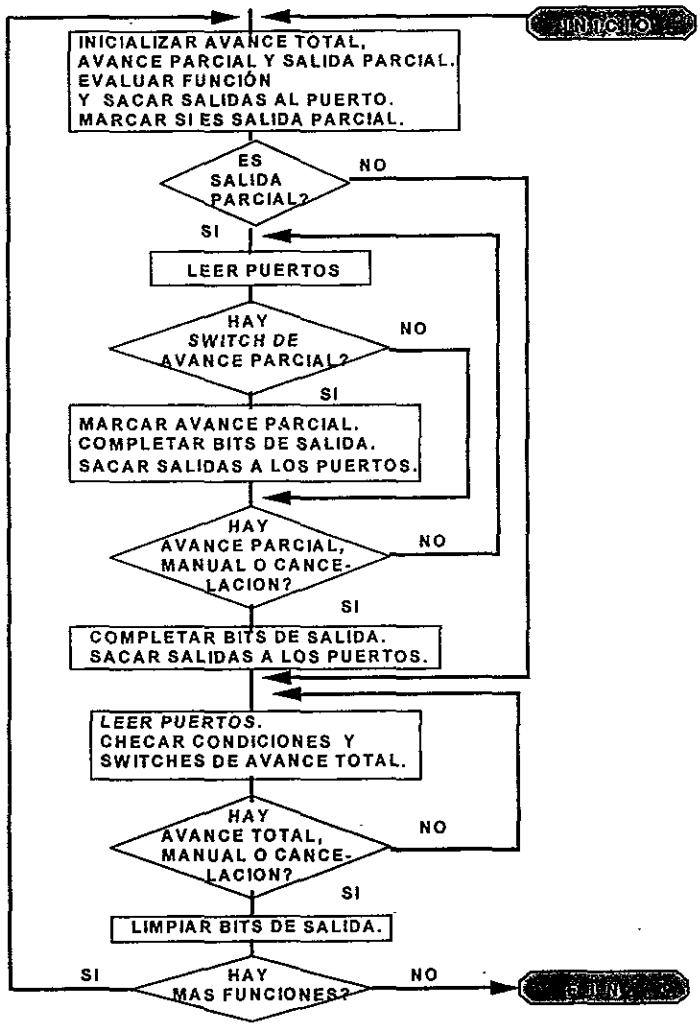


FIG. 3.6 - ESQUEMA DE CONTROL

3.3.- DESARROLLO DE LA INTERFAZ CON EL USUARIO

En la mayoría de los tornos automáticos la manera en que el operador selecciona una función a realizar en el torno es marcando mediante interruptores y perillas lo que desea ejecutar, esta selección, si el torno se puede programar, se almacena junto con otras (con distintas funciones) para crear una serie de pasos o secuencia de trabajo.

Pensando en el tablero de programación original de este torno, era práctico poder ir seleccionando las funciones disponibles, tan solo con marcarlas y después ejecutar todo el programa. La interfaz que aquí se desarrolla está basada en esa idea de crear, visualmente, una matriz donde las columnas sean las funciones y los renglones los pasos que irá realizando el torno.

Al realizarla se buscó, que fuera una interfaz simple, fácil de manejar (muchos operadores de tornos nunca han tenido ningún contacto con una computadora) y que no tuvieran que aprender muchos comandos o todo un lenguaje de programación .

La interfaz consta de 2 pantallas:

- **Pantalla No. 1** - ***Tablero de programación,*** donde el usuario puede marcar las funciones deseadas para crear su secuencia de operación, ejecutarla, modificarla y una vez probada tiene la posibilidad de almacenarla en el disco de la computadora para recuperarla posteriormente.

- **Pantalla No. 2** - ***Pantalla de parámetros,*** donde el usuario introduce parámetros como tiempos y número de veces que realizará una ejecución (Núm. de ciclos).

El programa de control maneja tanto la interfaz con el usuario como la comunicación con la tarjeta de interfaz de la PC, de esta forma el operador através de él puede controlar el torno.

El código del programa se anexa en el apéndice 'B' al igual que el manual de uso del mismo en el apéndice 'A', en este último apéndice podemos ver como se realiza la programación del torno através de la descripción de las pantallas del sistema.

CAPÍTULO 4

PRUEBAS

DE OPERACIÓN

CAPÍTULO 4 - PRUEBAS DE OPERACIÓN

Durante el desarrollo de las interfaces (con la PC y la etapa de potencia con el torno) y el programa de control se realizaron varias pruebas descritas a continuación:

- ***Prueba No. 1*** - Se realizaron pequeños programas que leían y escribían información al bús de datos de la computadora y al mismo tiempo en el slot (ISA) de la computadora se checaron en el osciloscopio que señales estaban presentes durante la lectura y cuales durante la escritura.

- ***Resultados*** - Se identificó que al escribir está presente la señal IOW y al leer está la señal IOR, ambas del bús tipo ISA.

- ***Prueba No. 2*** - En base a los resultados de la prueba No. 1, se diseñó la tarjeta de interfaz con la PC y se conectó en un slot de la máquina donde se volvieron a probar los programas de lectura y escritura de información a los puertos físicos para los que se diseñó la tarjeta (fig. 4.1).

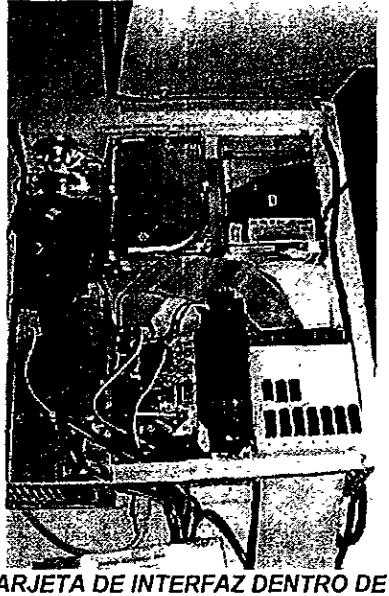


FIG. 4.1 - TARJETA DE INTERFAZ DENTRO DE LA PC.

- ***Resultados*** - Al correr el programa de escritura los latches de la tarjeta capturaban la información de salida correctamente (0 ó 1´s lógicos) y al correr el de escritura los buffers capturaban los voltajes (0 ó 1´s lógicos) que se conectaron en sus entradas adecuadamente.

- ***Prueba No. 3***- En el torno, se probó, enviando señales eléctricas, la activación de cada una (y de combinaciones) de las electroválvulas y contactores para ver la función que realizaban (fig. 4.2).

- ***Resultados*** - Se obtuvieron las funciones del torno que se realizan al enviar una señal o varias a la vez dentro de sus conexiones eléctricas.

FIG. 4.2 - GABINETE DE CONTROL DEL TORNO

- ***Prueba No. 4***- La interfaz de potencia con el torno se conectó al gabinete de éste para que activara las electroválvulas y contactores además de detectar la activación de los microinterruptores de límite de posición (fig. 4.3).

- ***Resultados*** - Al poner en la entrada de la etapa de potencia voltajes de encendido (0 ó 1´s lógicos), los elementos (TRIAC´s, transistores, etc.) soportaron y mantuvieron la activación de las electroválvulas y contactores, y al presionar los microinterruptores del torno la etapa de potencia entregó los voltajes (0 ó 1´s lógicos) correspondientes.

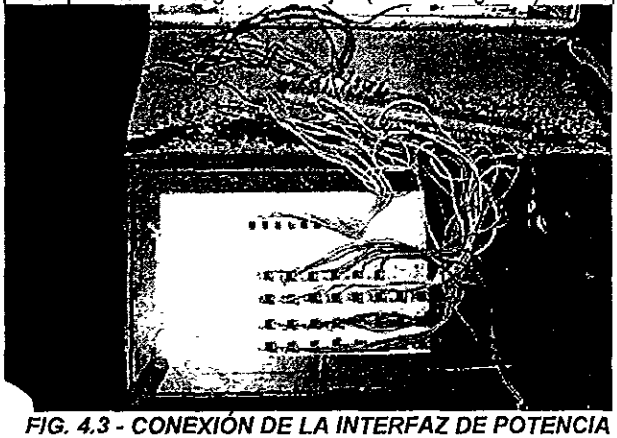


FIG. 4.3 - CONEXIÓN DE LA INTERFAZ DE POTENCIA AL GABINETE DEL TORNO

- ***Prueba No. 5***- Se conectó el torno a la etapa de potencia y ésta a su vez a la tarjeta de interfaz con la PC y con un pequeño programa se enviaban señales sencillas para ir prendiendo cada electroválvula y/o contactor (fig. 4.4).

- ***Resultados*** - Las señales enviadas pudieron combinarse (enviándolas al mismo tiempo) desde el programa para lograr realizar movimientos y funciones en el torno.

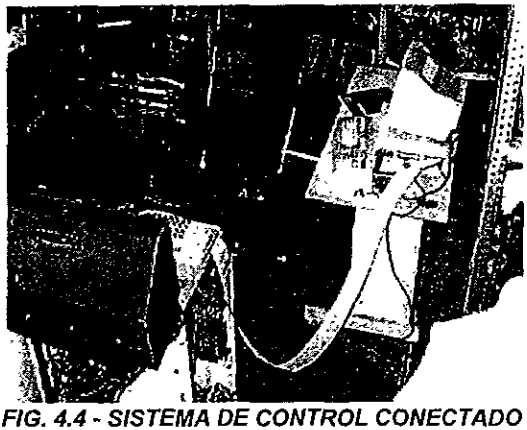


FIG. 4.4 - SISTEMA DE CONTROL CONECTADO

- ***Prueba No. 6***- Se corrió en la computadora el programa de control final, el cual ya agrupaba las combinaciones de bits (señales) a enviar por cada función marcada en pantalla.

- ***Resultados*** - Se pudo programar la operación automática del torno (fig. 4.5).

FIG. 4.5 - PRUEBAS DE PROGRAMACIÓN

Después de esta prueba se les explicó, a los operadores del torno, como manejar el programa de control, ellos se familiarizaron rápidamente con él, dieron sugerencias y comenzaron a elaborar programas para producir piezas.

Es importante mencionar en este punto que el operador, que posiblemente nunca había utilizado una computadora, cuenta con una experiencia (al conocer de materiales y herramientas) que le ayuda mucho en la realización de una secuencia de trabajo para elaborar una pieza (fig. 4.6).

FIG. 4.6 - OPERADOR TRABAJANDO

Actualmente el torno está produciendo varias piezas y lo manejan distintos operadores y ya hay otros tornos a los que se les implantará el mismo control (fig. 4.7).

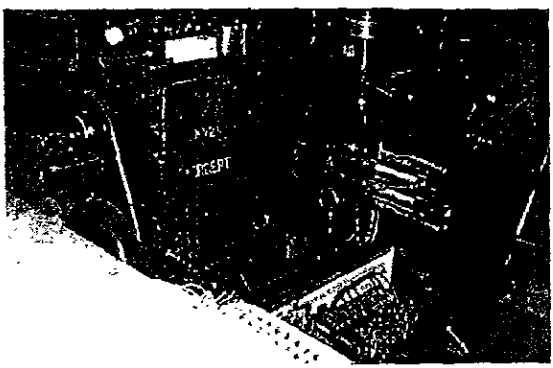


FIG. 4.7 - TORNOS HERBERT

CAPÍTULO 5

RESULTADOS

Y CONCLUSIONES

CAPÍTULO 5 .- RESULTADOS Y CONCLUSIONES

Durante el desarrollo del sistema de control, fue bueno contar con información acerca del funcionamiento del torno pero, sobre todo al comienzo, por no estar familiarizados con este tipo de máquinas, fue muy importante la asesoría de técnicos y operadores para ver si las fallas que se presentaban eran mecánicas o neumáticas, checar conexiones eléctricas y entender adecuadamente, en base a la operación del torno, lo que ellos necesitaban.

Con la experiencia, en esta área de trabajo, de los operadores y al observar el funcionamiento de distintos tornos se concluyó que todos tienen principios de operación comunes, en todos ellos se realiza una secuencia de pasos para obtener el resultado deseado.

Basado en esto último y al ver que otras máquinas de esta área, como troqueladoras y fresadoras, tienen principios de operación similares, se obtuvo el desarrollo de un control digital que puede ser adaptable a este tipo de máquinas que realizan una secuencia de trabajo.

La gente que trabaja con el torno se familiarizó rápidamente con el sistema obtenido, "debido a su facilidad de uso", comentaron.

Al usar la computadora en este diseño se facilitó el trabajo para crear la interface con el usuario ya que se aprovechan los recursos de ella misma (teclado, monitor, etc.), también se aprovecharon sus facilidades de expansión (puertos físicos) para añadirle la tarjeta que captura y envía información, esta tarjeta tiene la ventaja de que puede ser utilizada por cualquier computadora que tenga un slot ISA de 8 bits y las direcciones que utiliza libres de otro dispositivo. El monitor de la computadora puede ser monocromático y/o TTL incluso ya que se maneja solo texto en las pantallas.

Todo el desarrollo está basado en principios simples de la operación de cada parte de las que integran este sistema, desde como generar y recibir las señales de control desde y a la computadora hasta el funcionamiento de las etapas de potencia y aislamiento con el torno.

El desarrollo obtenido resulta a un precio accesible para cualquier empresa, tomando en cuenta el precio que tiene las tarjetas de captura y liberación de información para computadoras en el mercado o los módulos de entradas y salidas para PLC´s además de otros elementos necesarios en este tipo de sistemas.

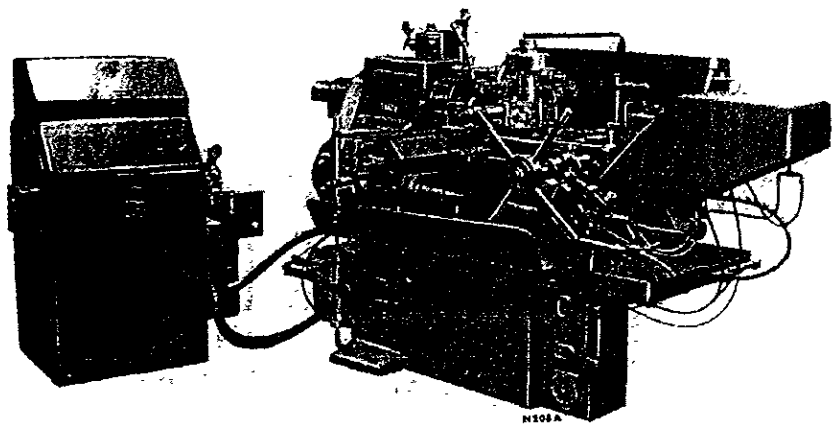
Se observó que el sistema ya integrado al ambiente de trabajo presenta un buen desempeño y funcionamiento.

APÉNDICE "A"

MANUAL DE USUARIO

DEL PROGRAMA

MANUAL DEL PROGRAMA DE CONTROL PARA EL TORNO HERBERT



FEBRERO DE 1998

ÍNDICE GENERAL

DESCRIPCIÓN GENERAL.-

El programa está diseñado para correr en cualquier computadora personal de tipo AT (80286 o superior) que cuente con un slot tipo ISA disponible para insertarle la tarjeta de interfaz diseñada para captura y salida de información de este programa hacia la interfaz de potencia del torno.

El objetivo de este programa es que el operador pueda programar una secuencia de operaciones, agrupándolas en pasos, para que al ejecutar esa serie de pasos automáticamente el torno vaya siguiendo las indicaciones programadas y dando lugar a una nueva pieza.

Una ventaja aquí es que podemos estar modificando y probando la secuencia para que quede lista antes de grabarla y también podemos recuperarla en otra ocasión por si la vamos a volver a necesitar. Además el programa puede estar ejecutándose en un ciclo repetitivo "n" veces (de 0 a 999).

El programa consta de 2 pantallas, en una se lleva a cabo la programación de la secuencia a ejecutar y en otra se introducen parámetros requeridos por esa programación (como son tiempos y el número de ciclos repetitivos si se desea).

ENTRADA AL SISTEMA.-

Para entrar al sistema desde el prompt del sistema operativo debemos ir al directorio donde se encuentra el programa:

c:\>\cd torno

y ejecutar el programa:

*c:\torno>**plugbo***

lo anterior nos llevará a la pantalla de programación del sistema.

DESCRIPCIÓN DE PANTALLAS.-

PANTALLA # 1 : TABLERO DE PROGRAMACIÓN

Esta pantalla es donde llegamos al entrar al sistema, observamos en ella dos renglones o barras en la parte superior, la primera nos indicará siempre con un número la columna dentro del área cuadriculada donde nos encontramos y nos dirá también que función se activará en el torno si marcamos ese cuadrito y la segunda nos indica que teclas podemos utilizar para trabajar :

F2: nos permite grabar en el disco duro de la *computadora*, la secuencia de *programación*,

F3: inicializa el tablero de programación y la pantalla de *parámetros* para crear una nueva secuencia,

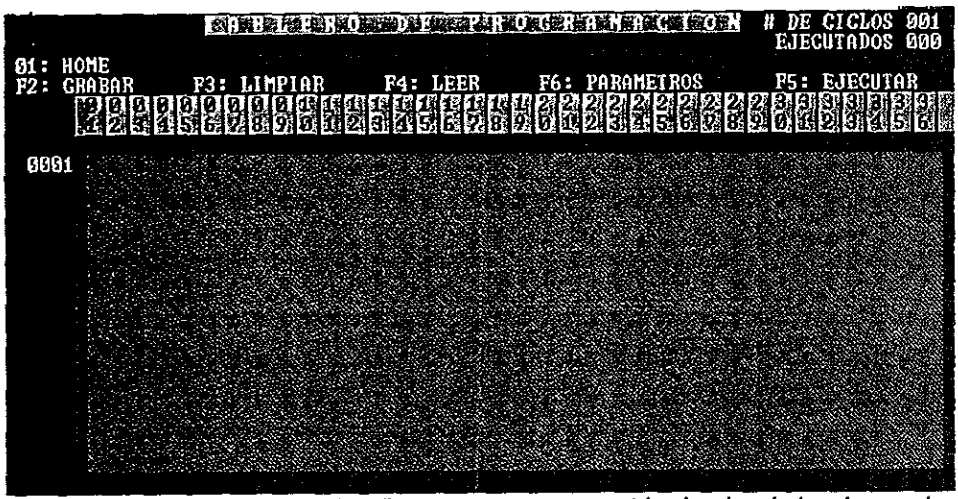
F4: recupera del disco duro una secuencia de programación antes grabada,

F5: nos permite ejecutar la secuencia de programación que tengamos en ese momento en pantalla,

F6: *nos lleva a la pantalla de parámetros,*

F10: ésta solo es visible durante la ejecución del *programa* y nos sirve para *cancelar la misma,*

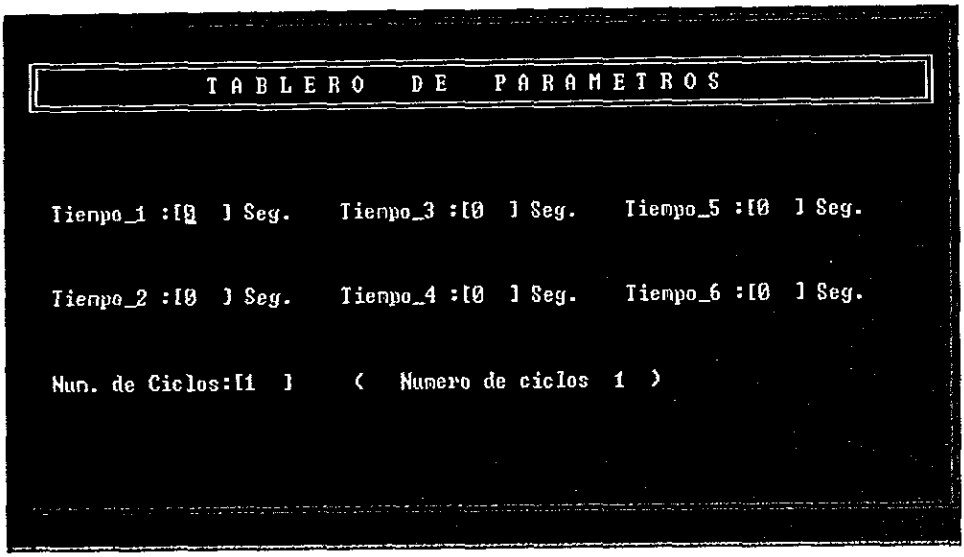
ESC: desde esta pantalla al presionar la tecla *ESC terminamos la ejecución* del programa y *regresamos* al sistema operativo.



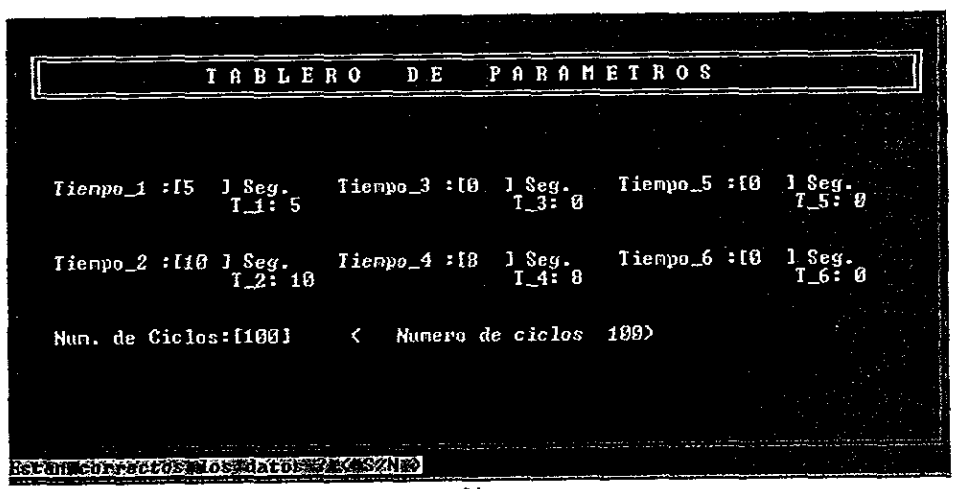
El área cuadriculada (azul) representa una matriz donde al desplazar el cursor a la izquierda o a la derecha podemos ir marcando o desmarcando usando la tecla *enter* las funciones *deseadas*, cada renglón en ella representa un paso (conjunto de funciones) y si nos desplazamos hacia abajo creamos un nuevo paso (cosa que identificamos al ver aparecer a la izquierda de la pantalla fuera de la cuadrícula un nuevo número consecutivo), *para* desplazarnos en las 4 direcciones *posibles* debemos usar las *flechitas* del teclado.

También aparecen aquí, en la esquina *superior derecha*, el número de veces *que vamos* a ejecutar la secuencia programada y un contador de esa ejecución.

PANTALLA # 2 : ***TABLERO DE PARAMETROS***



A esta pantalla llegamos de la pantalla # 1 al usar la tecla F6, aquí debemos introducir los tiempos deseados, si es que los necesitamos para la programación, y el número de veces que ejecutaremos el programa (Núm. de ciclos). Solo debemos teclear el número deseado y presionar *enter* para avanzar al siguiente tiempo, si no deseamos modificar solo presionamos enter y nos toma el número que tengamos ahí, después del número de ciclos aparece en la parte inferior una barra que nos pregunta si los datos están correctos o no, si decimos que Si (tecleando una S) el programa nos regresa a la pantalla # 1, sino, se nos vuelve a pedir que introduzcamos los tiempos y número de ciclos deseados.



FUNCIONES DISPONIBLES PARA LA PROGRAMACIÓN.-

Dentro de la matriz que representa el tablero de programación cada renglón representa un paso y cada columna representa una función que se puede programar para que ocurra en ese paso. Aunque hay lugar para 36 funciones actualmente no se utilizan todas.

Las funciones actualmente disponibles para operar el torno son:

01: ***HOME*** - señal necesaria para indicar que es el último paso a ejecutar en una secuencia.

02: ***AVANCE DE TORRETA*** - debe ser marcada para permitir el desplazamiento de la torreta. Se debe combinar con los avances de los hidros 1 ó 2 ó con el avance mecánico.

03: ***REGRESO DE TORRETA*** - indicar el retorno hacia atrás de la torreta.

04: ***AVANCE CON HIDRO 1*** - se debe combinar con el avance de torreta para avanzar usando el hidro # 1.

05: ***AVANCE CON HIDRO 2*** - se debe combinar con el avance de torreta para avanzar usando el hidro # 2.

06: ***SELECCIÓN DEL AVANCE MECÁNICO DE LA TORRETA*** - se debe combinar con el avance de torreta para avanzar usando el avance mecánico.

07: ***SELECCIÓN DE RANGO DE AVANCE MECÁNICO (+ LENTO)*** - se debe usar con el avance mecánico.

08: ***SELECCIÓN DE RANGO DE AVANCE MECÁNICO (+ RÁPIDO)*** - se debe usar con el avance mecánico.

09: ***AVANCE DEL FRONTAL CRUZADO*** - activa la entrada del poste frontal cruzado.

10: ***AVANCE DEL POSTERIOR CRUZADO*** - activa la entrada del poste posterior cruzado.

11: ***AVANCE DEL WORKCATCHER*** - activa la bajada del workcatcher.

12: ***REGRESO DEL WORKCATCHER*** - activa la subida del workcatcher.

13: ***ABRIR CHUCK*** - abre el chuck del torno.

14: ***CHIPRRUPTER*** - activa el chiprrupter.

15: ***SACA EMBRAGUE MECÁNICO DE LAS VELOCIDADES LENTAS*** - para activar las velocidades lentas del motor del eje principal.

16: ***METE EMBRAGUE MECÁNICO DE LAS VELOCIDADES RÁPIDAS*** - para activar las velocidades rápidas del motor del eje principal.

17: ***ARRANQUE DE CONTACTOR PRINCIPAL PARA MOTOR*** - señal necesaria para encender al motor del eje principal del torno. Se combina con el marcado de un sentido de giro y una velocidad

18: ***GIRO DE MOTOR ATRÁS*** **<-** indica el sentido del giro del eje principal.

19: ***GIRO DE MOTOR ADELANTE*** **->** indica el sentido del giro del eje principal.

20: ***VELOCIDAD 'A'*** - activa esta velocidad en el motor.

21: ***VELOCIDAD 'B'*** - activa esta velocidad en el motor.

22: ***VELOCIDAD 'C'*** - activa esta velocidad en el motor.

23: ***VELOCIDAD 'D'*** - activa esta velocidad en el motor.

24: ***ARRANQUE DE MOTOR ALIMENTADOR ATRÁS*** - activa el alimentador del material en un sentido.

25: ***ARRANQUE DE MOTOR ALIMENTADOR ADELANTE*** - activa el alimentador del material en el otro sentido.

26: ***TIEMPO_1*** - indicar que se empleará un tiempo para cambiar al siguiente paso.

27: ***TIEMPO_2*** - indicar que se empleará un tiempo para cambiar al siguiente paso.

28: ***TIEMPO_3*** - indicar que se empleará un tiempo para cambiar al siguiente paso.

29: ***TIEMPO_4*** - indicar que se empleará un tiempo para cambiar al siguiente paso.

30: ***TIEMPO_5*** - indicar que se empleará un tiempo para cambiar al siguiente paso.

31: ***TIEMPO_6*** - indicar que se empleará un tiempo para cambiar al siguiente paso.

32, 33, 34, 35, 36: ***SIN USO***

Si se llegan a marcar en el mismo paso dos o más señales que causen conflicto al torno, como pueden ser avance y regreso de torreta, el programa indica un error al tratar de ejecutar esa secuencia.

PROGRAMACIÓN DE UNA SECUENCIA.-

Antes de crear una nueva secuencia podemos presionar la tecla *F3* para asegurarnos de inicializar el tablero.

Para ir creando una secuencia se deben ir indicando las funciones deseadas en cada paso, para programar una función tan solo se debe marcar posicionando el cursor en la columna deseada y presionando la tecla *enter* con ésto se nota que aparece un punto brillante en esa posición indicando que esa función se activó. Los movimientos deseados en el torno los lograremos combinando la activación de las distintas funciones en cada paso.

La creación de un nuevo paso tan solo es presionando la tecla de flechita para abajo, así podemos crear una secuencia de 'n' pasos.

Al ir creando la secuencia puede que necesitemos programar tiempos para el avance de un paso a otro, en la pantalla de programación solo debemos marcar que tiempo(s) (del 1 al 6) vamos a usar y en la pantalla de parámetros debemos introducir el valor de ese tiempo(s).

A continuación se muestra una pequeña secuencia que realiza lo siguiente:

PASO 1 : avanza la torreta rápidamente,

PASO 2 : se retrae la torreta hacia atrás,

PASO 3 : avanza la torreta con el hidro # 1,

PASO 4 : se retrae la torreta hacia atrás,

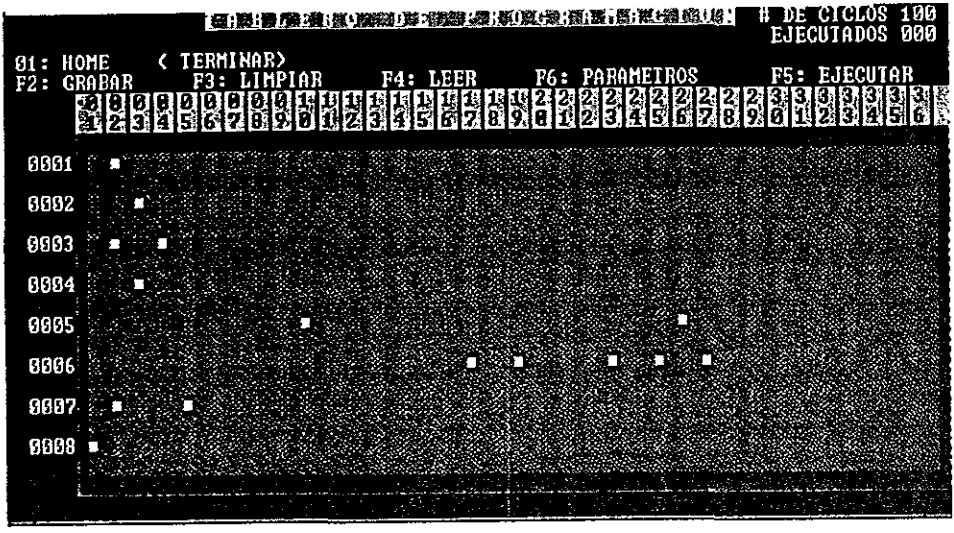
PASO 5 : avanza el posterior cruzado durante el tiempo indicado en el tiempo_1,

PASO 6 : arranca el motor del eje principal con la velocidad 'D' y girando hacia adelante, además de arrancar el motor del alimentador hacia adelante durante el tiempo indicado en el tiempo_2,

PASO 7 : avanza la torreta con el hidro # 2,

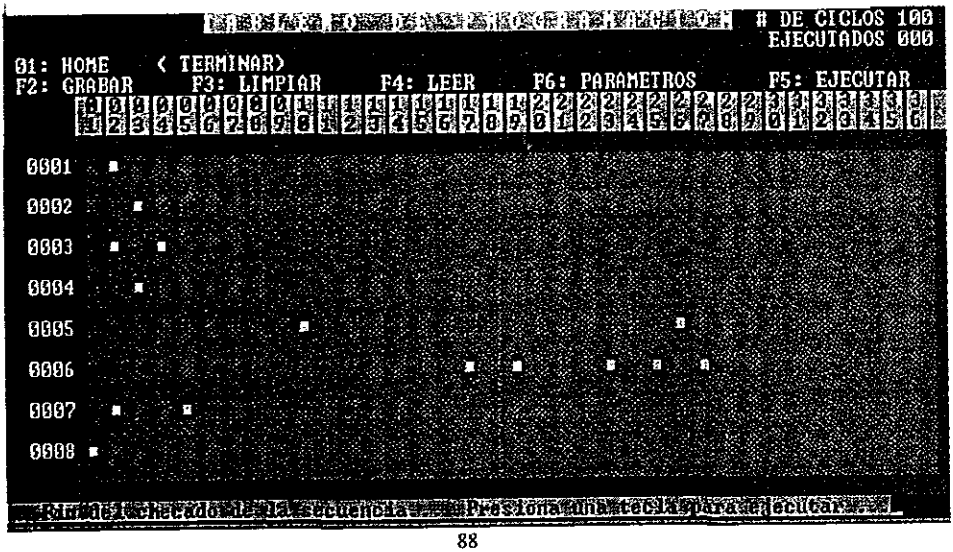
PASO 8 : indica la finalización de esa secuencia.

en el tablero de programación se vería de la siguiente forma:

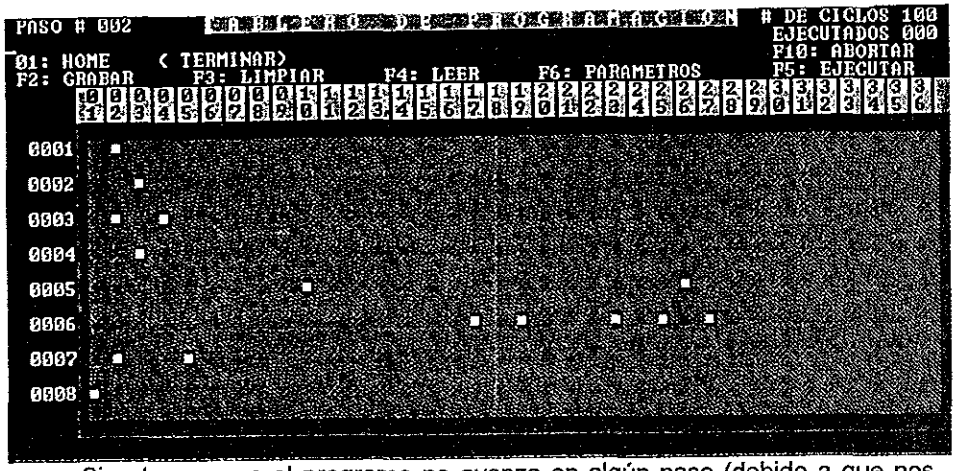


EJECUCIÓN DE UNA SECUENCIA DE PROGRAMACIÓN.-

Cuando tenemos en la pantalla lista nuestra secuencia (y todo ya conectado y encendido) podemos iniciar la ejecución automática al presionar la tecla *F5*, antes de ejecutarse los pasos son checados por el programa para ver que no se hayan marcado funciones contradictorias que puedan dañar o invalidar la operación del torno, si todo está correcto nos aparece en la parte inferior de la pantalla una barra con un mensaje pidiéndonos presionar otra tecla para comenzar con la ejecución:

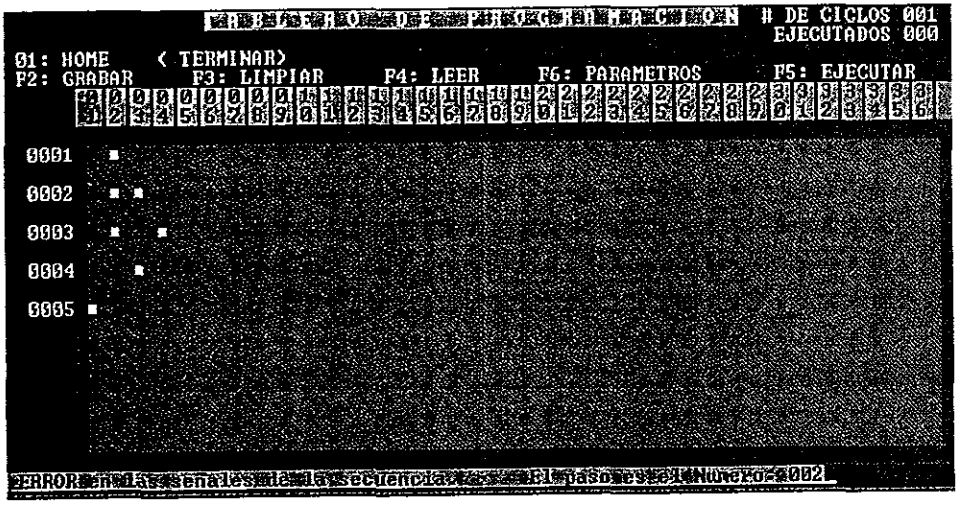


al presionar cualquier otra tecla comienza la ejecución y podemos notar que en la esquina superior izquierda se nos va indicando (parpadeando) el número de paso que se va ejecutando, además de que en la esquina superior derecha se va incrementando el contador del *número de veces que repite* el programa (si es que le pusimos un número de ciclos mayor a 1 en la pantalla de parámetros).



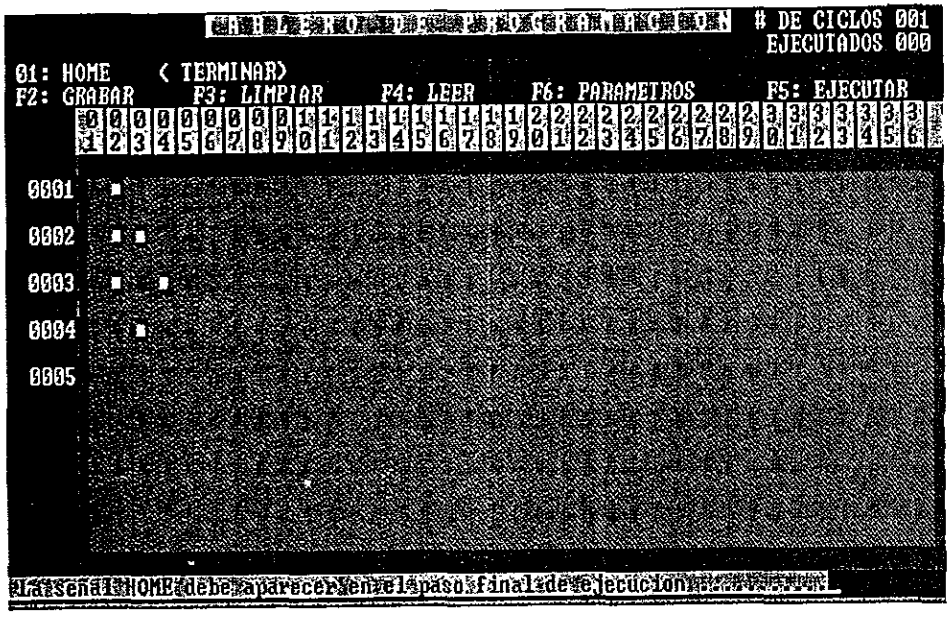
Si notamos que el programa no avanza en algún paso (debido a que nos faltó algo en la programación) podemos forzarlo a avanzar al siguiente paso presionando la tecla *F5* nuevamente.

Si existen errores en la programación (contradicciones o marcados de funciones indebidas) se indica un mensaje de error diciéndonos el número de paso en que se encuentra ese error para que podamos corregirlo y volver a presionar *F5* hasta que ya no aparezcan errores.



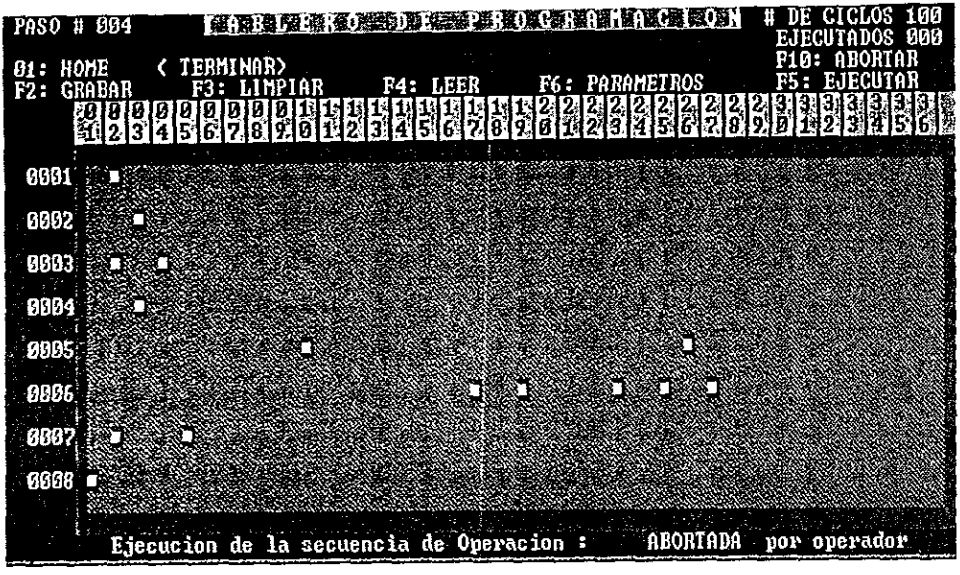
Al final de la ejecución la pantalla quedará como antes de comenzar la ejecución.

Otro mensaje de error que puede aparecernos al final del checado de la secuencia y antes de la ejecución, es cuando olvidamos marcar en que paso termina la ejecución de la misma. Siempre debe estar indicado el final de la ejecución de la secuencia en el paso deseado.



CANCELACIÓN DURANTE LA *EJECUCIÓN* DE UNA SECUENCIA.-

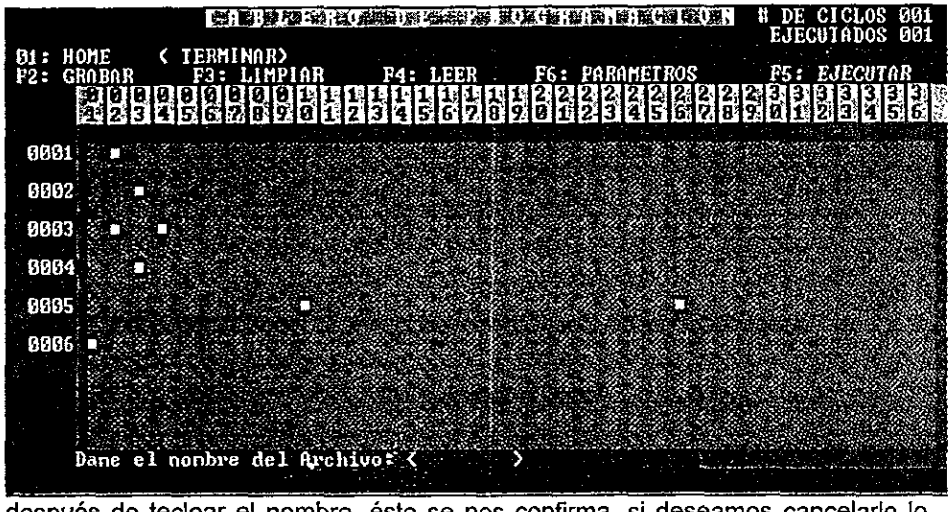
Si estamos ejecutando una secuencia y por algún motivo deseamos cancelar la ejecución de la misma solo debemos presionar la tecla *F10* y se nos indica que se abortó esa ejecución.



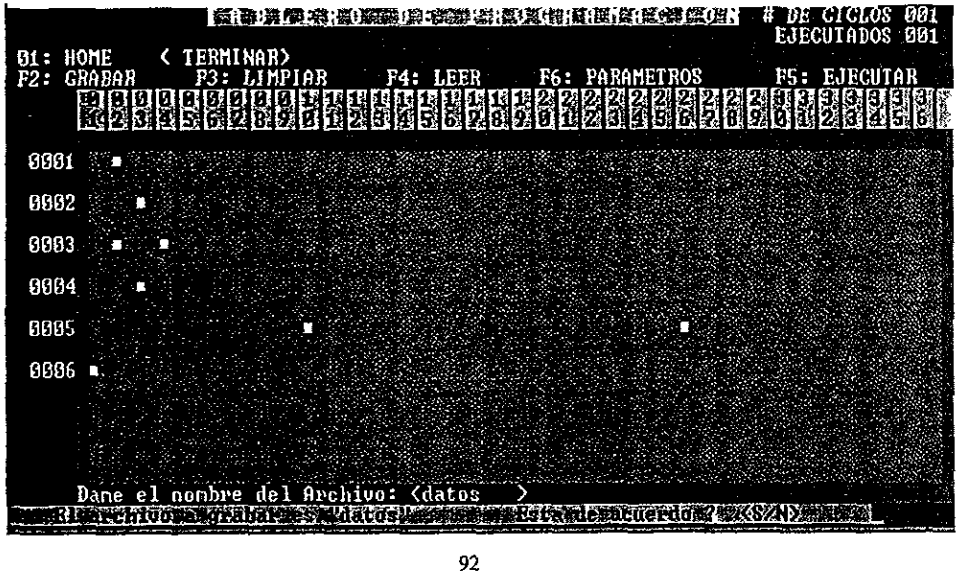
SALVAR UNA SECUENCIA O PROGRAMA.-

Cuando tenemos lista y probada la secuencia en la pantalla es el momento de grabarla en el disco duro para no perderla al apagar la máquina o al realizar otra nueva.

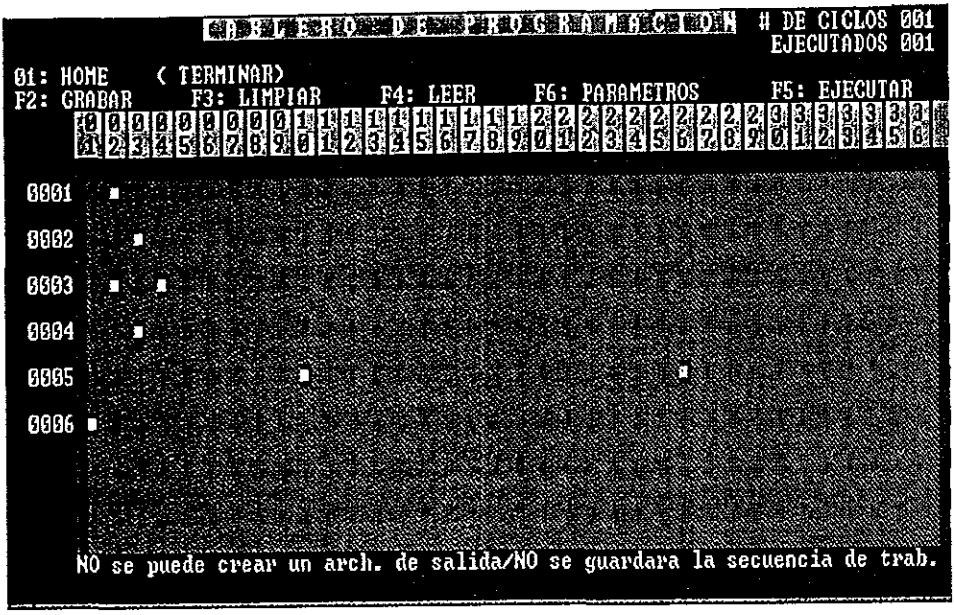
Para grabarla presionamos la tecla F2 y se nos pide un nombre para el archivo, debemos teclear no más de 8 caracteres:



después de teclear el nombre, éste se nos confirma, si deseamos cancelarlo lo hacemos sino aceptamos con una 'S' y la secuencia quedará almacenada.

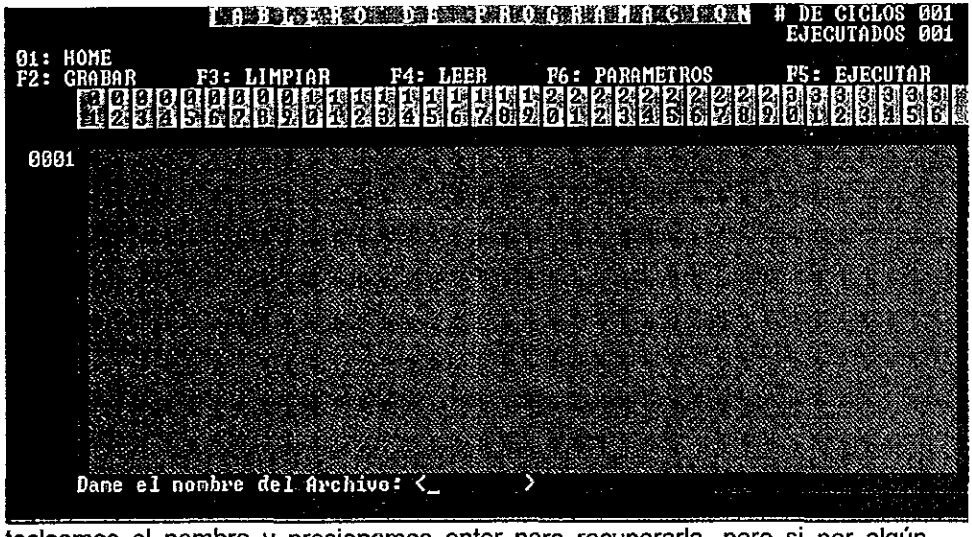


si cuando estamos grabando existe algún problema al crear los archivos se nos indicará en la pantalla y la secuencia ***NO quedará almacenada*** en la computadora.

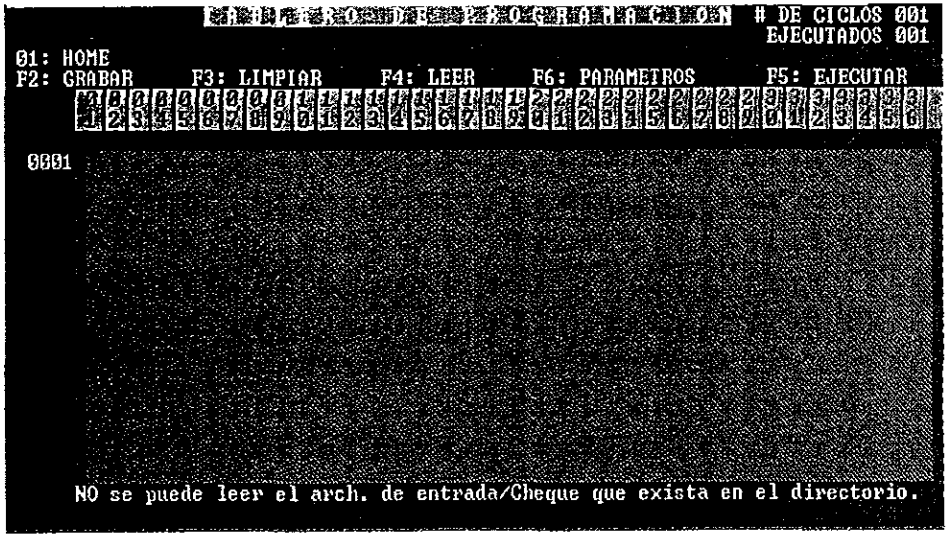


RECUPERAR UNA SECUENCIA O PROGRAMA.-

Para recuperar una secuencia previamente guardada en el disco duro de la computadora, tan solo presionamos *F4* y se nos pide el nombre con que fue grabada:



tecleamos el nombre y presionamos enter para recuperarla, pero si por algún motivo no encuentra esa secuencia (porque esté dañada o haya sido borrada) el programa nos lo indicará.



APÉNDICE "B"

CÓDIGO

DEL PROGRAMA

```
*************************************************************************************
***** * * *                          PLUGBO.C          .              * * * ****
≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈
≈ AUTORA: Ma  Mercedes Bolaños Castellanos  <FACULTAD DE INGENIERIA C.U.>
≈
≈ DESCRIPCION DEL PROGRAMA:
≈         Fué elaborado para permitir la programación para automatizar el torno
≈ mediante la escritura y lectura de los puertos físicos donde se conec-
≈ ta la tarjeta de interfaz, la interfaz con el usuario simula un table-
≈ ro de programación donde se marca la secuencia de operación deseada,
≈ la cuál se puede almacenar al disco duro para recuperarla posterior-
≈ mente.
≈
≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈
*************************************************************************************
*************************************************************************************/


#include <conio.h>
#include <ctype.h>
#include <dos h>
#include <bios.h>
#include <time.h>
#include <mem h>
#include <stdio h>
#include <string h>
#include <stdlib.h>

#define PRIN_Y 8                          /* Definiciones */
#define PRIN_X 8
#define NUM_REN_VIS 14
#define AVANCE 2
#define ESC 283
#define ENTER 7181
#define ARRIBA 18432
#define ABAJO 20480
#define DERECHA 19712
#define IZQUIERDA 19200
#define BARRA 14624
#define F1 15104
#define F2 15360
#define F3 15616
#define F4 15872
#define F5 16128
#define F6 16384
#define C_IND 0x5F
#define C_TAB 0x3B
#define C_ERR 0x4F
#define C_NOR 0x0F
#define C_DAT 0x9E
#define TIEMPO 400
#define OK 0
#define NO_OK  -1
#define L_ESC 27
#define L_TAB 9
#define L_RET 8
#define L_BARRA 32
#define L_ENTER 13
#define VIDEO 0x10

                                          /* Estructuras */
struct ele_sec{
  unsigned char dato_1;
  unsigned char dato_2;
  unsigned char dato_3;
  unsigned char dato_4;
  unsigned char dato_5;
  struct ele_sec *sig,
} *ini_sec, *fin_sec, *pos_sec, *aux_sec, *aux_2, *aux_pos_sec;
```

```
struct direccion{                          /* Para las direcciones */
  char far *p;
} temp;

                                           /* Puntero a la dirección de la interrupción 9 */
  struct direccion far *addr = (struct direccion far *) 36;


                                           /* Puntero a la dirección de la interrupción 60 */
  struct direccion far *int9 = (struct direccion far *) 24;



                                           /* Para los datos */
struct menu
{
  char nombre[50];
  char valor[30],
  int y;
  int x;
  int len;
  int tipo;
},

struct menu *pmenu, var_menu[]=
{
 {"ARC_ENT","",36,23,8,0},
 {"ARC_SAL","",36,23,8,0},
 {"TIME_1","",9,16,3,1},
 {"TIME_2","",13,16,3,1},
 {"TIME_3","",9,40,3,1},
 {"TIME_4","",13,40,3,1},
 {"TIME_5","",9,64,3,1},
 {"TIME_6","",13,64,3,1},
 {"NUMERO_CICLOS","1",17,21,3,1}
},


                                           /* Funciones */
void ini_ptos(void),
void des_inf(void);
int pausa(char * msg);
void despliega(int);
void pinta_tablero(void);
short crea_nvo_ele_sec(void),
short existe_ele_sec(short ele);
short pos_rel_x(short pos_act_x);
short dato_de_ubicacion(short pos_rel);
short bit_de_ubicacion(short pos_rel);
void cambia_status_pan(short dato_u, short bit_u),
void pinta_status(int y);
void borra_sec(void);
void esc(short x,short y, char * msg);
void marco(short x1,short y1,short x2,short y2, unsigned int color);
void graba_sec(char *archivo);
void lee_arc_sec(char *archivo);
short crea_primer_ele(void);
void copia(char * fte,char * des,short posi,short longit),
short ejecuta_sec(void);
void interrupt aplicacion(void);
short prendido(unsigned char  dato_eval ,short x);
void cambia(unsigned char * ,short x);
int input_x_y(struct menu *ppmenu);
int checa_sec(void);
void menu_param(void);
int input_x_y_otro(struct menu *ppmenu);
void ini_tiempos(void);
short tiempo_transcurrido(short);
```

```
                                        /* Variables globales */
int o_pos_x, o_pos_y,o_o_pos_x, o_o_pos_y, sel_x, sel_y, valor_x;
unsigned int num_act_sec=0;
unsigned int cont_ele_sec=0;
char buffer2[1200];
char msg_e[90];
char * error_tec;
long bios_time;
int tiempo_act, tiempo_ini_1, tiempo_ini_2, tiempo_ele_1, tiempo_ele_2;
int tiempo_ini_3, tiempo_ini_4, tiempo_ele_3, tiempo_ele_4;
int tiempo_ini_5, tiempo_ini_6, tiempo_ele_5, tiempo_ele_6,
int ciclos_ele=1;
int ciclos_eje=0;


int tecla_f5 = 0;                       /*  Para las teclas */
int tecla_f10 = 0;


                                        /* Direcciones de los puertos */


unsigned int d_s_byte_1 = 0xFF00;       /* Salidas              */
unsigned int d_s_byte_2 = 0xFF01;
unsigned int d_s_byte_3 = 0xFF02;
unsigned int d_s_byte_4 = 0xFF03;
unsigned int d_s_byte_5 = 0xFF04,

unsigned int d_e_byte_1 = 0xFF05;       /* Entradas             */
unsigned int d_e_byte_2 = 0xFF06;

                                        /* Salidas a los puertos físicos */
unsigned char c_s_byte_1 = 0x00 ;
unsigned char c_s_byte_2 = 0x00 ;
unsigned char c_s_byte_3 = 0x00 ;
unsigned char c_s_byte_4 = 0x00 ;
unsigned char c_s_byte_5 = 0x00 ;

                                        /* Entradas a los puertos físicos */
unsigned char c_e_byte_1 = 0x00 ;
unsigned char c_e_byte_2 = 0x00 ;


char ruta_sal[50];                      /* Para archivo de salida */
char ruta_sal_1[50];
char ruta_ent[50];                      /* Para archivo de entrada */
char ruta_ent_1[50];
char buffer[4096];



int main(void)
{
  int tecla,i, res, res_l, continua;
  short cancelado=0;
  char *archivo_sal = "       ";
  char *archivo_ent = "       ";

  ini_sec=fin_sec=pos_sec=aux_sec=aux_2=aux_pos_sec=NULL;

  ini_ptos();
  ini_tiempos();
  textattr(C_NOR);
  clrscr();
  pinta_tablero();                      /* Presentación del tablero inicial en blanco */

  if (!crea_primer_ele())
  {
   num_act_sec=0;
```

```
gotoxy(1,25);
textattr(C_ERR);
cprintf(" No hay memoria suficiente para almacenar la secuencia...          ");
delay(TIEMPO*4);
textattr(C_NOR);
exit(0);
};

while (( tecla=bioskey(0)) != ESC )     /* Movimiento sobre el tablero */
{

        if ( tecla == ABAJO )
        {

                sel_y=sel_y+AVANCE;
                num_act_sec++;                  /* Numero de elemento en secuencia */
                aux_2=pos_sec;    /* Ubico nodo con datos en pos_sec*/
                if((!existe_ele_sec(num_act_sec) && (crea_nvo_ele_sec())) || (existe_ele_sec(num_act_sec)) )
                {
                 if (sel_y > PRIN_Y+NUM_REN_VIS)      /* Si paso el final del tablero inicial 8+14=22*/
                 {                              /* No hay fin en la secuencia                  */
                  if(num_act_sec > (NUM_REN_VIS/AVANCE +1))             /* Si debo repintar los de arriba */
                  {
                        sel_y=sel_y-AVANCE;
                        aux_pos_sec=pos_sec;
                        for (i=NUM_REN_VIS; i > PRIN_Y - (NUM_REN_VIS/AVANCE) ;i=i-AVANCE)  /* 8-7=1 */
                        {
                          gotoxy(PRIN_X-5,sel_y-i);
                          cprintf("%04d",num_act_sec-(i/AVANCE));
                          existe_ele_sec(num_act_sec-(i/AVANCE));
                          pinta_status(sel_y-i);
                        }
                        pos_sec=aux_pos_sec;
                  }
                 }
                 gotoxy(PRIN_X-5,sel_y);
                 cprintf("%04d",num_act_sec);
                 gotoxy(sel_x,sel_y);           /* Pinto posición actual */
                 pinta_status(sel_y);
                 valor_x=pos_rel_x(sel_x);
                 des_inf();
                }
                else

                {
                  pos_sec=aux_2;
                  sel_y=sel_y-AVANCE;
                  num_act_sec--;
                  cont_ele_sec--;

                  o_pos_x= wherex();
                  o_pos_y= wherey();
                  gotoxy(1,25);
                  textattr(C_ERR);
                  cprintf(" Ya no se puede incrementar el numero de pasos de la Secuencia...No Mem.");
                  delay(TIEMPO*4);
                  textattr(C_TAB);
                  gotoxy(1,25);
                  cprintf("                                                    ");
                  textattr(C_NOR);
                  gotoxy(o_pos_x,o_pos_y);
                }

        }     /* Fin de Flecha abajo */
        else
                if ( tecla == ARRIBA)
                {
                  sel_y=sel_y-AVANCE;
                  num_act_sec--;
                  if (sel_y < PRIN_Y)      /* Si paso el inicio del tablero */
```

```
{
  if(num_act_sec==0)    /* Si es el primer elemento */
  {
        num_act_sec=1;
        sel_y= PRIN_Y;
  }
  else
  {
                              /* Si habia recorrido hacia abajo el despliegue */
        sel_y=sel_y+AVANCE;    /* Repinto los de abajo                  */
        aux_pos_sec=pos_sec,
        for (i=AVANCE; i < PRIN_Y+(NUM_REN_VIS/AVANCE) ;i=i+AVANCE)  /* 8+7=15 */
        {
          gotoxy(PRIN_X-5,sel_y+i),
          cprintf("%04d",num_act_sec+(i/AVANCE));
          existe_ele_sec(num_act_sec+(i/AVANCE));
          pinta_status(sel_y+i);
        }
        pos_sec=aux_pos_sec;
  }
}
gotoxy(PRIN_X-5,sel_y),
cprintf("%04d",num_act_sec);
existe_ele_sec(num_act_sec ); /* Ubico nodo con datos */
gotoxy(sel_x,sel_y);              /* Pinto posición actual  */
pinta_status(sel_y),
valor_x=pos_rel_x(sel_x);
des_inf();
}
else
  if ( tecla == IZQUIERDA)
  {

        sel_x=sel_x-2;
        if (sel_x < PRIN_X)
              sel_x= sel_x+2;
        gotoxy(sel_x,sel_y);
        despliega(sel_x);
        valor_x=pos_rel_x(sel_x);
        des_inf();
  }
  else
        if ( tecla == DERECHA)
        {
          sel_x=sel_x+2;
          if (sel_x > PRIN_X+70)
                sel_x= sel_x-2,
          gotoxy(sel_x,sel_y);
          despliega(sel_x);
          valor_x=pos_rel_x(sel_x);
          des_inf();
        }
        else
          if ( tecla == ENTER )
          {
                valor_x=pos_rel_x(sel_x),

                cambia_status_pan(dato_de_ubicacion(valor_x),bit_de_ubicacion(valor_x));
                textattr(C_TAB);
                gotoxy(sel_x,sel_y);
                des_inf();
                textattr(C_NOR);
          }
          else
            if ( tecla == F2 )
            {
              o_pos_x= wherex(),
              o_pos_y= wherey(),

                textattr(C_IND),
```

```
                esc(7,24,"Dame el nombre del Archivo: <        >              ");
                gotoxy(36,24);

                pmenu= &var_menu[1];
                pmenu->valor[0]='\0';
                res=8;
                do
                {
                do
                {
                   res=input_x_y(pmenu);
                   gotoxy(36,24);
                  if(res==L_RET)
                  {
                    pmenu->valor[strlen(pmenu->valor)-1]='\0';
                    gotoxy(36+strlen(pmenu->valor),24);
                    cprintf(" ");
                    gotoxy(36+strlen(pmenu->valor),24);
                  }
                }while(res!=OK && res!=NO_OK);
                if (res==OK)
                {
                  res_l=0;
                  continua=0;
                  sprintf(msg_e,"El archivo a grabar es \"%s\" ,Esta de acuerdo ?  (S/N) ",pmenu->valor);
                  res_l=pausa(msg_e);
                  if(toupper(res_l)=='S')
                    continua=1;
                  else
                    pmenu->valor[0]='\0';
                }
                else
                {
                  pmenu->valor[0]='\0';
                  break;                /* rompe el while */
                }
                }while(!continua);
                strcpy(archivo_sal,pmenu->valor);

                if(res==OK)
                  graba_sec(archivo_sal);    /* Grabación de la secuencia */

                textattr(C_TAB);
                gotoxy(PRIN_X-1,PRIN_Y+16);
                cprintf("                                                    ");
                textattr(C_TAB);
                gotoxy(7,24);
                cprintf("                                                    ");
                textattr(C_NOR);
                gotoxy(o_pos_x,o_pos_y);
}
else
        if(tecla == F3)
        {
          borra_sec();
          pinta_tablero();
          ini_tiempos();
          if(!crea_primer_ele())
          {
            num_act_sec=0;
            gotoxy(1,25);
            textattr(C_ERR);
            cprintf(" No hay memoria suficiente para almacenar la secuencia .          ");
            delay(TIEMPO*4);
            textattr(C_NOR);
            gotoxy(1,25);
            cprintf("                                                    ");
            break;        /*      exit(0);  */
          };
        }
```

```
else
  if(tecla == F4)
  {
    o_pos_x= wherex();
    o_pos_y= wherey();
    borra_sec();
    pinta_tablero();
    ini_tiempos();
    textattr(C_IND);
    esc(7,24,"Dame el nombre del Archivo: <       >              ");
    gotoxy(36,24);
    pmenu= &var_menu[0];
    pmenu->valor[0]='\0';
    res=8;
    do
    {
          res=input_x_y(pmenu);
          gotoxy(36,24);
          if(res==L_RET)
          {
            pmenu->valor[strlen(pmenu->valor)-1]='\0';
            gotoxy(36+strlen(pmenu->valor),24);
            cprintf(" ");
            gotoxy(36+strlen(pmenu->valor),24);
          }

    }while(res!=OK && res!=NO_OK);

    strcpy(archivo_ent,pmenu->valor);

    gotoxy(o_pos_x,o_pos_y);

    if(res==OK)
          lee_arc_sec(archivo_ent);
    else        /* crear el primer nodo */
          if (!crea_primer_ele())
          {
            num_act_sec=0;
            gotoxy(1,25);
            textattr(C_ERR);
            cprintf("No hay memoria suficiente para almacenar la secuencia...      ");
            delay(TIEMPO*4);
            textattr(C_NOR);
            break;
          };
    textattr(C_TAB);
    gotoxy(PRIN_X-1,PRIN_Y+16);
    cprintf("                                                                    ");
    textattr(C_NOR);
    gotoxy(PRIN_X,PRIN_Y);
    }
    else
          if(tecla == F5)
          {
            ciclos_eje=0;
            o_o_pos_x=wherex();
            o_o_pos_y=wherey();
            gotoxy(65,2);
            textattr(0x1F);
            cprintf(" EJECUTADOS %03d",ciclos_eje);
            textattr(C_NOR);

            if(checa_sec())
            {
                  cancelado=0;
                  do
                  {
                    if(ejecuta_sec())
                    {
                      textattr(0x1F);
```

```
                                        ciclos_eje++,
                                        gotoxy(65,2);
                                        cprintf(" EJECUTADOS %03d",ciclos_eje),
                                        textattr(C_NOR),
                                       }
                                       else
                                       {
                                        cancelado=1;
                                       }
                                      }while(ciclos_eje!=ciclos_ele && !cancelado );

                                    }
                                    gotoxy(o_o_pos_x,o_o_pos_y);
                                  }
                                   else
                                    if(tecla == F1)
                                    {
                                      o_o_pos_x=wherex();
                                      o_o_pos_y=wherey();
                                      gettext(1, 1, 80, 25, buffer);
                                      clrscr();
                                      printf("Saliendo al Sistema Operativo... Teclee EXIT para regresar . "),
                                      system("C:\\command.com");
                                      clrscr(),
                                      puttext(1, 1, 80, 25, buffer);
                                      gotoxy(o_o_pos_x,o_o_pos_y);
                                    }
                                    else
                                     if(tecla == F6)
                                     {
                                            o_o_pos_x=wherex();
                                            o_o_pos_y=wherey();
                                            gettext(1, 1, 80, 25, buffer),
                                            clrscr(),
                                            menu_param(),
                                            puttext(1, 1, 80, 25, buffer);
                                            textattr(C_IND);
                                            gotoxy(65,1);
                                            cprintf("# DE CICLOS %03d",ciclos_ele);
                                            textattr(C_NOR),
                                            gotoxy(o_o_pos_x,o_o_pos_y),
                                     }

        }                                   /* Fin de la Secuencia, Del While */

 textattr(C_NOR);
 borra_sec();
  return 0;
}


void despliega(int f_x)
{
 int aux=0;
 aux=f_x - PRIN_X;
 textattr(0x5F);
 o_pos_x= wherex();
 o_pos_y= wherey();

 switch(aux)
 {
  case 0 :
          gotoxy(1,3);
          cprintf(" 01: HOME   ( TERMINAR)                                    ");
          break;
  case 2 :
          gotoxy(1,3);
          cprintf(" 02: AVANCE DE TORRETA                                     ");
          break;
  case 4 :
```

```
        gotoxy(1,3);
        cprintf(" 03: REGRESO DE TORRETA                                    ");
        break;
case 6 :
        gotoxy(1,3),
        cprintf(" 04: AVANCE CON HIDRO No.1                                 ");
        break;
case 8 :
        gotoxy(1,3);
        cprintf(" 05: AVANCE CON HIDRO No.2                                 ");
        break;
case 10 :
        gotoxy(1,3);
        cprintf(" 06: SELECCIÓN DE AVANCE MECÁNICO DE LA TORRETA            ");
        break;
case 12 :
        gotoxy(1,3);
        cprintf(" 07: SELECCIÓN DE RANGO DE AVANCE MECÁNICO ( + LENTO )     ");
        break;
case 14 :
        gotoxy(1,3);
        cprintf(" 08: SELECCIÓN DE RANGO DE AVANCE MECÁNICO ( + RÁPIDO )    ");
        break;
case 16 :
        gotoxy(1,3);
        cprintf(" 09: AVANCE DEL FRONTAL CRUZADO                            ");
        break;
case 18 .
        gotoxy(1,3);
        cprintf(" 10: AVANCE DEL POSTERIOR CRUZADO                          ");
        break;
case 20 :
        gotoxy(1,3);
        cprintf(" 11: AVANCE DE WORKCATCHER                                 ");
        break;
case 22 :
        gotoxy(1,3),
        cprintf(" 12: REGRESO DE WORKCATCHER                                ");
        break;
case 24 .
        gotoxy(1,3),
        cprintf(" 13: ABRIR CHUCK                                           ");
        break;
case 26 :
        gotoxy(1,3),
        cprintf(" 14: CHIPRRUPTER                                           ");
        break;
case 28 :
        gotoxy(1,3),
        cprintf(" 15: SACA EMBRAGUE MECÁNICO DE LAS VELOCIDADES LENTAS      ");
        break;
case 30 :
        gotoxy(1,3);
        cprintf(" 16: METE EMBRAGUE MECÁNICO DE LAS VELOCIDADES RÁPIDAS     ");
        break;
case 32 :
        gotoxy(1,3);
        cprintf(" 17: ARRANQUE DE CONTACTOR PRINCIPAL PARA MOTOR            ");
        break;
case 34 :
        gotoxy(1,3);
        cprintf(" 18: GIRO DE MOTOR ATRÁS      <-                           ");
        break;
case 36 :
        gotoxy(1,3);
        cprintf(" 19. GIRO DE MOTOR ADELANTE    ->                          ");
        break;
case 38 :
        gotoxy(1,3);
        cprintf(" 20: VELOCIDAD 'A'                                         ");
```

```
        break;
case 40 :
        gotoxy(1,3);
        cprintf(" 21: VELOCIDAD `B`                                    ");
        break;
case 42 :
        gotoxy(1,3);
        cprintf(" 22. VELOCIDAD `C`                                    ");
        break;
case 44 :
        gotoxy(1,3);
        cprintf(" 23: VELOCIDAD `D`                                    ");
        break;
case 46 :
        gotoxy(1,3);
        cprintf(" 24: ARRANQUE DE MOTOR ALIMENTADOR ATRÁS                    ");
        break;
case 48 :
        gotoxy(1,3);
        cprintf(" 25: ARRANQUE DE MOTOR ALIMENTADOR ADELANTE                 ");
        break;
case 50 :
        gotoxy(1,3);
        cprintf(" 26· TIEMPO_1 , ( Marcarlo dando F6 en la pantalla de parámetros ) ");
        break;
case 52 :
        gotoxy(1,3);
        cprintf(" 27: TIEMPO_2 , ( Marcarlo dando F6 en la pantalla de parámetros ) ");
        break;
case 54 :
        gotoxy(1,3);
        cprintf(" 28  TIEMPO_3 , ( Marcarlo dando F6 en la pantalla de parámetros ) ");
        break;
case 56 :
        gotoxy(1,3);
        cprintf(" 29. TIEMPO_4 , ( Marcarlo dando F6 en la pantalla de parámetros ) ");
        break;
case 58 :
        gotoxy(1,3);
        cprintf(" 30: TIEMPO_5 , ( Marcarlo dando F6 en la pantalla de parámetros ) ");
        break;
case 60 :
        gotoxy(1,3);
        cprintf(" 31: TIEMPO_6 , ( Marcarlo dando F6 en la pantalla de parámetros ) ");
        break;
case 62 :
        gotoxy(1,3);
        cprintf(" 32· SIN USO                                          ");
        break;
case 64 :
        gotoxy(1,3);
        cprintf(" 33: SIN USO                                          ");
        break;
case 66 :
        gotoxy(1,3);
        cprintf(" 34: SIN USO                                          ");
        break;
case 68 :
        gotoxy(1,3);
        cprintf(" 35: SIN USO                                          ");
        break;
case 70 :
        gotoxy(1,3);
        cprintf(" 36: SIN USO                                          ");
        break;
}
textattr(C_NOR);
gotoxy(o_pos_x,o_pos_y);
}
```

```
int pausa(char * msg)
{
          short o_pos_x, o_pos_y;
          int respuesta,
          char buffer_pausa[192],
          o_pos_x= wherex(),
          o_pos_y= wherey();
          gotoxy(1,25);
          gettext(1, 25, 80, 25, buffer_pausa),
          textattr(0x7C);
          cprintf("%s",msg);
          respuesta=getch();
          puttext(1, 25, 80, 25, buffer_pausa);
          textattr(C_NOR),
          gotoxy(o_pos_x,o_pos_y);
          return(respuesta);
}

void des_inf()
{
          char buffer_pausa[192],
          o_pos_x= wherex(),
          o_pos_y= wherey();
          gotoxy(1,25);
          gettext(1, 25, 80, 25, buffer_pausa);
          textattr(C_IND);
          puttext(1, 25, 80, 25, buffer_pausa);
          textattr(C_NOR);
          gotoxy(o_pos_x,o_pos_y);
}

void pinta_tablero(void)
{
  textattr(0x7F);
  gotoxy(18,1);
  cprintf("T A B L E R O   D E   P R O G R A M A C I Ó N");
  textattr(C_IND);
  gotoxy(65,1);
  cprintf("# DE CICLOS %03d",ciclos_ele);
  textattr(0x1F);
  gotoxy(65,2);
  cprintf(" EJECUTADOS %03d",ciclos_eje),
  textattr(0x7C);
  gotoxy(PRIN_X-1,PRIN_Y-3);
  cprintf(" 0³0³0³0³0³0³0³0³0³1³1³1³1³1³1³1³1³1³1³2³2³2³2³2³2³2³2³2³2³3³3³3³3³3³3³3³ "),
  gotoxy(PRIN_X-1,PRIN_Y-2);
  cprintf(" 1³2³3³4³5³6³7³8³9³0³1³2³3³4³5³6³7³8³9³0³1³2³3³4³5³6³7³8³9³0³1³2³3³4³5³6³ ");
  textattr(C_TAB);
  gotoxy(PRIN_X-1,PRIN_Y-1);          /* Pintado del tablero */
  cprintf("                                                                          ");
  gotoxy(PRIN_X-1,PRIN_Y),
  cprintf(" ±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±° ");
  gotoxy(PRIN_X-1,PRIN_Y+1);
  cprintf(" •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••• ");
  gotoxy(PRIN_X-1,PRIN_Y+2),
  cprintf(" ±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±° ");
  gotoxy(PRIN_X-1,PRIN_Y+3),
  cprintf(" •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••• ");
  gotoxy(PRIN_X-1,PRIN_Y+4);
  cprintf(" ±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±° "),
  gotoxy(PRIN_X-1,PRIN_Y+5);
  cprintf(" •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••• ");
  gotoxy(PRIN_X-1,PRIN_Y+6),
  cprintf(" ±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±° ");
  gotoxy(PRIN_X-1,PRIN_Y+7);
  cprintf(" •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••• ");
  gotoxy(PRIN_X-1,PRIN_Y+8);
  cprintf(" ±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±° ");
  gotoxy(PRIN_X-1,PRIN_Y+9);
  cprintf(" •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••• ");
```

```
  gotoxy(PRIN_X-1,PRIN_Y+10);
  cprintf(" ±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±° ");
  gotoxy(PRIN_X-1,PRIN_Y+11);
  cprintf(" •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••• ");
  gotoxy(PRIN_X-1,PRIN_Y+12);
  cprintf(" ±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±° ").
  gotoxy(PRIN_X-1,PRIN_Y+13);
  cprintf(" •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••• ");
  gotoxy(PRIN_X-1,PRIN_Y+14);
  cprintf(" ±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±°±° ").
  gotoxy(PRIN_X-1,PRIN_Y+15);
  cprintf(" •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••• ");
  gotoxy(PRIN_X-1,PRIN_Y+16);
  cprintf("                                                      ");
                                       /* Condiciones de no existir sec. */
  sel_x=PRIN_X.                        /* Posicion inicial en el tablero */
  sel_y=PRIN_Y;
  textattr(C_IND);                      /* Mensaje de posicion uno */
  gotoxy(1,3);
  cprintf(" 01. HOME                                        ");
  textattr(0x1F);
  gotoxy(1,4);
  cprintf(" F2: GRABAR    F3: LIMPIAR    F4: LEER    F6: PARAMETROS    F5: EJECUTAR  ").
  num_act_sec=1;                       /* núm. de Posición en la secuencia */
  textattr(C_NOR);
  gotoxy(sel_x-5,sel_y).               /* pintado del 001 tab. */
  cprintf("%04d",num_act_sec);
  gotoxy(sel_x,sel_y);                 /* Posición inicial */
}


short crea_nvo_ele_sec(void)           /* Para elementos de la secuencia */
{
   struct ele_sec *nvo_ele_sec;

   nvo_ele_sec = (struct ele_sec *)malloc(sizeof(struct ele_sec));
   if(!nvo_ele_sec)
   {
     sprintf (msg_e,"No se puede crear nuevo elemento en Memoria").
     return(0);
   }
   if(ini_sec==NULL)
   {
     ini_sec=fin_sec=nvo_ele_sec;
   }
   else
   {
     fin_sec=fin_sec->sig=nvo_ele_sec;
   }
   fin_sec->dato_1=0x00;
   fin_sec->dato_2=0x00.
   fin_sec->dato_3=0x00;
   fin_sec->dato_4=0x00.
   fin_sec->dato_5=0x00;
   fin_sec->sig=NULL;
   pos_sec=fin_sec;
   cont_ele_sec++;
   return(1);
}

short existe_ele_sec(short nodo_bus)
{
   short busca=0;
   aux_sec=ini_sec.
   busca++;
   do
   {
     if(nodo_bus!=busca)
     {
            aux_sec=aux_sec->sig;
```

```
            busca++;
    }
    else
    {
            pos_sec=aux_sec;
            return busca;
    }
  }while(aux_sec!=NULL);
  return(0);
}

short pos_rel_x(short pos_act_x)
{
  int res=0;
  res= (((pos_act_x)-PRIN_X)/2)+1;
  return res;
}

short dato_de_ubicacion(short pos_rel)
{
  int res=0;
  int cos=0;
  cos = pos_rel/8 ;
  res = pos_rel%8 ;
  if (!res)
    return(cos);
  else
    return(cos+1);
}


short bit_de_ubicacion(short pos_rel)
{
  int res=0;
  int cos=0;
  if (pos_rel < 9)
  {
    return(pos_rel);
  }
  else
  {
    cos = pos_rel/8 ;
    res = pos_rel%8 ;
    if(!res)
            return(8);
    else
            return(res);
  }
}

void cambia_status_pan(short dato_u, short bit_u)
{
  short esta_prendido=0;
  unsigned char dato_u_c='\0';

  switch(dato_u){
                  case 1:
                          dato_u_c=pos_sec->dato_1; break;
                  case 2:
                          dato_u_c=pos_sec->dato_2; break;
                  case 3:
                          dato_u_c=pos_sec->dato_3; break;
                  case 4:
                          dato_u_c=pos_sec->dato_4; break;
                  case 5:
                          dato_u_c=pos_sec->dato_5; break;
  }
  esta_prendido=prendido(dato_u_c,bit_u);
  cambia(&dato_u_c,bit_u);
```

```
 switch(dato_u){
                        case 1:
                                pos_sec->dato_1=dato_u_c; break;
                        case 2:
                                pos_sec->dato_2=dato_u_c; break;
                        case 3:
                                pos_sec->dato_3=dato_u_c; break;
                        case 4:
                                pos_sec->dato_4=dato_u_c; break;
                        case 5:
                                pos_sec->dato_5=dato_u_c; break;
 }
 if(esta_prendido)
 {
   textattr(C_TAB);
   cprintf("±");
 }
 else
 {
   textattr(0x3E);
   cprintf("þ");
 }
}


void pinta_status(int y)
{
   int pos_rel, pos_act_x;
   int bit_u, dato_u,
   unsigned char dato_u_c='\0';
   unsigned char dato_aux='\0';
   short esta_prendido=0,
   o_pos_x= wherex(),
   o_pos_y= wherey();
   for (pos_rel=1; pos_rel < 37 ; pos_rel++)
   {
          pos_act_x = ((pos_rel-1)*2) + PRIN_X ,
          dato_u=dato_de_ubicacion(pos_rel);
          bit_u=bit_de_ubicacion(pos_rel);

          switch(dato_u){
                        case 1:
                                dato_u_c=pos_sec->dato_1; break,
                        case 2:
                                dato_u_c=pos_sec->dato_2, break;
                        case 3:
                                dato_u_c=pos_sec->dato_3; break;
                        case 4:
                                dato_u_c=pos_sec->dato_4; break;
                        case 5:
                                dato_u_c=pos_sec->dato_5; break;
          }
          gotoxy(pos_act_x,y);
           esta_prendido=prendido(dato_u_c,bit_u);
          if(esta_prendido)
          {                             /* prendido */
            textattr(0x3E);
            cprintf("þ");
          }
          else
          {
            textattr(C_TAB),
            cprintf("±");
          }
   }        /* Fin del For */
   textattr(C_NOR);
   gotoxy(o_pos_x,o_pos_y);
}
```

```
void marco(short x1,short y1,short x2,short y2,unsigned int color)
{
        int i;
        textattr(color);
        for (i=x1;i<x2;++i)
 {
                gotoxy(i,y1);
                cprintf("Í");
                gotoxy(i,y2);
                cprintf("Í");
        }
        for (i=y1;i<y2,++i)
        {
                gotoxy(x1,i);
                cprintf("º");
                gotoxy(x2,i);
                cprintf("º");
        }
        gotoxy(x1,y1);  cprintf("É");
        gotoxy(x2,y1);  cprintf("»");
        gotoxy(x1,y2);  cprintf("È");
        gotoxy(x2,y2);  cprintf("¼");
        textattr(C_NOR);
}


void esc(short x,short y, char * msg)
{
 gotoxy(x,y);
 cprintf("%s",msg);
}

void graba_sec(char *archivo)
{
  FILE *fp, *fpa ;
  aux_pos_sec=ini_sec;
  memset(ruta_sal,'\0',50);
  memset(ruta_sal_1,'\0',50);
  strcat(ruta_sal,"C:\\TORNO\\");
  strcat(ruta_sal,archivo);
  strcat(ruta_sal_1,ruta_sal);
  strcat(ruta_sal,".PLU");
  strcat(ruta_sal_1,".TPO");
  if( (fp=fopen(ruta_sal,"w"))==NULL )
  {
    textattr(C_ERR);
    esc(7,24,"NO se puede crear un arch. de salida/NO se guardara la secuencia de trab. ");
    delay(TIEMPO*4);
    gotoxy(PRIN_X-1,PRIN_Y+16);
    textattr(C_TAB);
    cprintf("                                                    ");
    textattr(C_NOR);
    gotoxy(o_pos_x,o_pos_y);
    return;
  }
  if( (fpa=fopen(ruta_sal_1,"w"))==NULL )
  {
    textattr(C_ERR);
    esc(7,24,"NO se puede crear un arch. de salida/NO se guardara la secuencia de trab. ");
    delay(TIEMPO*4);
    gotoxy(PRIN_X-1,PRIN_Y+16);
    textattr(C_TAB);
    cprintf("                                                    ");
    textattr(C_NOR);
    gotoxy(o_pos_x,o_pos_y);
    return;
  }
  while(aux_pos_sec != NULL)
  {
```

```
    fprintf(fp,"%03d%03d%03d%03d%03d%03d\n",aux_pos_sec->dato_1,aux_pos_sec->dato_2\
                    ,aux_pos_sec->dato_3,aux_pos_sec->dato_4,aux_pos_sec->dato_5),
    aux_pos_sec=aux_pos_sec->sig;
  }

fprintf(fpa,"%03d%03d%03d%03d%03d%03d",tiempo_ele_1,tiempo_ele_2,tiempo_ele_3,tiempo_ele_4,tiempo_ele_5,tiempo_ele_6
),
  fclose(fp);
  fclose(fpa),
}

void lee_arc_sec(char *archivo)
{
  char *lectura, lec_3[3], *bas;
  FILE *fp1, *fp1a ;
  int aux=0,
  int aux_2=0,
  int c_l_i=0,
  int c_l_c=0,
  memset(ruta_ent,'\0',50);
  memset(ruta_ent_1,'\0',50);
  strcat(ruta_ent,"C:\\TORNO\\"),
  strcat(ruta_ent,archivo),
  strcat(ruta_ent_1,ruta_ent);
  strcat(ruta_ent," PLU");
  strcat(ruta_ent_1,".TPO"),
  if( ( (fp1=fopen(ruta_ent,"r"))==NULL ) ||( (fp1a=fopen(ruta_ent_1,"r"))==NULL ) )
  {
    textattr(C_ERR),
    esc(7,24,"NO se puede leer el arch. de entrada/Cheque que exista en el directorio  "),
    delay(TIEMPO*4),
    gotoxy(PRIN_X-1,PRIN_Y+16),
    cprintf("                                                    "),
    textattr(C_NOR);
    gotoxy(o_pos_x,o_pos_y);
    borra_sec(),
    pinta_tablero();
    if (!crea_primer_ele())
    {
          num_act_sec=0,
          gotoxy(1,25);
          textattr(C_ERR);
          cprintf(" No hay memoria suficiente para almacenar la secuencia.          "),
          delay(TIEMPO*4);
          textattr(C_NOR);
          fclose(fp1);
          return;
    },
    return;
  }
  while(!feof(fp1))
  {
    fread(lectura,16,1,fp1);
    c_l_l++;
    if(strlen(lectura)==0)
          c_l_l --;
  }
  fseek(fp1,0L,SEEK_SET),
  while((!feof(fp1)) && c_l_l!=c_l_c+1 )
  {
   if(crea_nvo_ele_sec())
   {
          fread(lectura,15,1,fp1);
          fread(bas,1,1,fp1);
          copia(lectura,lec_3,1,3);
          pos_sec->dato_1=atoi(lec_3);
          copia(lectura,lec_3,4,3);
          pos_sec->dato_2=atoi(lec_3);
          copia(lectura,lec_3,7,3);
          pos_sec->dato_3=atoi(lec_3);
```

```
        copia(lectura,lec_3,10,3);
        pos_sec->dato_4=atoi(lec_3);
        copia(lectura,lec_3,13,3);
        pos_sec->dato_5=atoi(lec_3);
        c_f_c++;
  }
  if (sel_y == PRIN_Y  && cont_ele_sec==1)
  {
    pinta_status(sel_y);
  }
  if (sel_y == PRIN_Y+2  && cont_ele_sec==2)
  {
    pinta_status(sel_y);
    gotoxy(PRIN_X-5,PRIN_Y+2);
    cprintf("%04d",cont_ele_sec);
  }
  if (sel_y == PRIN_Y+4  && cont_ele_sec==3)
  {
    pinta_status(sel_y);
    gotoxy(PRIN_X-5,PRIN_Y+4);
    cprintf("%04d",cont_ele_sec);
  }
  if (sel_y == PRIN_Y+6  && cont_ele_sec==4)
  {
    pinta_status(sel_y);
    gotoxy(PRIN_X-5,PRIN_Y+6);
    cprintf("%04d",cont_ele_sec);
  }
  if (sel_y == PRIN_Y+8  && cont_ele_sec==5)
  {
    pinta_status(sel_y);
    gotoxy(PRIN_X-5,PRIN_Y+8);
    cprintf("%04d",cont_ele_sec);
  }
  if (sel_y == PRIN_Y+10  && cont_ele_sec==6)
  {
    pinta_status(sel_y);
    gotoxy(PRIN_X-5,PRIN_Y+10);
    cprintf("%04d",cont_ele_sec);
  }
  if (sel_y == PRIN_Y+12  && cont_ele_sec==7)
  {
    pinta_status(sel_y);
    gotoxy(PRIN_X-5,PRIN_Y+12);
    cprintf("%04d",cont_ele_sec);
  }
  if (sel_y == PRIN_Y+14  && cont_ele_sec==8)
  {
    pinta_status(sel_y);
    gotoxy(PRIN_X-5,PRIN_Y+14);
    cprintf("%04d",cont_ele_sec);
  }
    sel_y=sel_y+AVANCE;
}/* Fin del While */

ini_tiempos();
fread(lectura,18,1,fp1a);
pmenu=&var_menu[2];
copia(lectura,lec_3,1,3);
strcpy(pmenu->valor,lec_3);
tiempo_ele_1=atoi(lec_3);
pmenu=&var_menu[3];
copia(lectura,lec_3,4,3);
strcpy(pmenu->valor,lec_3);
tiempo_ele_2=atoi(lec_3);
pmenu=&var_menu[4];
copia(lectura,lec_3,7,3);
strcpy(pmenu->valor,lec_3);
tiempo_ele_3=atoi(lec_3);
pmenu=&var_menu[5];
```

```
  copia(lectura,lec_3,10,3);
  strcpy(pmenu->valor,lec_3);
  tiempo_ele_4=atoi(lec_3);
  pmenu=&var_menu[6];
  copia(lectura,lec_3,13,3);
  strcpy(pmenu->valor,lec_3);
  tiempo_ele_5=atoi(lec_3);
  pmenu=&var_menu[7];
  copia(lectura,lec_3,16,3);
  strcpy(pmenu->valor,lec_3);
  tiempo_ele_6=atoi(lec_3);
  fclose(fp1);
  fclose(fp1a);
  pos_sec=ini_sec;
  sel_y=PRIN_Y;
}

void borra_sec(void)
{
  int i;
  while(ini_sec!=NULL)
  {
    aux_sec=ini_sec;
    ini_sec=ini_sec->sig;
    free(aux_sec);
  }
  textattr(C_NOR);
  sel_y=PRIN_X;
  for (i=0; i < PRIN_Y+(NUM_REN_VIS/AVANCE)  ;i=i+AVANCE)
  {
    gotoxy(PRIN_X-5,sel_y+i);
    cprintf("   ",num_act_sec+(i/AVANCE));
  }
  num_act_sec=0;
  cont_ele_sec=0;
}

short crea_primer_ele(void)
{
  if(crea_nvo_ele_sec())       /* La primera vez que entra */
  {
    pos_sec=ini_sec;           /* Apuntador a primer elemento de la sec. */
    cont_ele_sec=1;
    return(1);
  }
  return(0);
}

 void copia(char * fte,char * des,short posi,short longit)
{
  char *fuente, *destino;
  destino=des;
  fuente=fte+posi-1;
  for(fuente;fuente<(fte+posi+longit-1);fuente++)
  {
    *destino=*fuente;
    destino++;
  }
  *destino='\0';
}

short ejecuta_sec(void)
{
  short avance_total, error=0;
  short avance_parcial;
  short salida_parcial;
  unsigned int num_act_sec_aux;
  int9->p = addr->p;                    /* Cambio del vector de interrupción 9 */
  addr->p = (char far *) aplicacion;
  aux_2=pos_sec;                        /* Apunta al primer elemento */
```

```
pos_sec=ini_sec;
num_act_sec_aux=num_act_sec;
num_act_sec=1;
gotoxy(66,3);
textattr(C_ERR);
cprintf("F10: ABORTAR  ");
textattr(0x9F);
gotoxy(66,4);
cprintf("F5: EJECUTAR  ");
do
{
        avance_total=0;
        avance_parcial=0;
        salida_parcial=0;

/*****************************************************************/
/* Mapeado Pantalla contra funciones */
/*****************************************************************/

        if (prendido(pos_sec->dato_1,1))    /* Home, retorno a inicio */
        {
          ini_ptos();
          break;
        }
        if (prendido(pos_sec->dato_1,2))    /* Avance de Torreta */
        {
          cambia(&c_s_byte_1,3);
          cambia(&c_s_byte_2,1);
        }
        if (prendido(pos_sec->dato_1,3))    /* Regreso de Torreta */
        {
          cambia(&c_s_byte_1,3);
        }
        if (prendido(pos_sec->dato_2,1))    /* Avance del mov. frontal cruzado */
        {
          cambia(&c_s_byte_1,2);
        }
        if (prendido(pos_sec->dato_2,2))    /* Avance del mov. posterior cruzado */
        {
          cambia(&c_s_byte_1,1);
          cambia(&c_s_byte_1,4);
        }
        if (prendido(pos_sec->dato_2,3))    /* Avance del workcatcher */
        {
          cambia(&c_s_byte_1,3);
          cambia(&c_s_byte_1,6);
        }
        if (prendido(pos_sec->dato_2,4))    /* Regreso del workcatcher */
        {
          cambia(&c_s_byte_1,3);
        }
        if (prendido(pos_sec->dato_2,5))    /* Abrir Chuck */
        {
          cambia(&c_s_byte_2,4);
        }
        if (prendido(pos_sec->dato_2,6))    /* Chiprruter */
        {
          cambia(&c_s_byte_2,5);
        }
        if (prendido(pos_sec->dato_2,7))    /* Saca embrague mec. Veloc. Lentas */
        {
          cambia(&c_s_byte_1,5);
        }
        if (prendido(pos_sec->dato_2,8))    /* Mete embrague mec  Veloc. Rápidas */
        {
          cambia(&c_s_byte_1,8);
        }
        if (prendido(pos_sec->dato_3,1))    /* Arranque de contactor para motor principal */
        {
```

```
  cambia(&c_s_byte_3,2);
}
if (prendido(pos_sec->dato_3,2))    /* Giro de motor hacia Atrás */
{
  cambia(&c_s_byte_3,5);
}
if (prendido(pos_sec->dato_3,3))    /* Giro de motor hacia Adelante */
{
  cambia(&c_s_byte_3,6);
}
if (prendido(pos_sec->dato_3,4))    /* Velocidad `A` */
{
  cambia(&c_s_byte_3,7);
}
if (prendido(pos_sec->dato_3,5))    /* Velocidad `B` */
{
  cambia(&c_s_byte_3,8);
}
if (prendido(pos_sec->dato_3,6))    /* Velocidad `C` */
{
  cambia(&c_s_byte_4,1);
}
if (prendido(pos_sec->dato_3,7))    /* Velocidad `D` */
{
  cambia(&c_s_byte_4,2);
}
if (prendido(pos_sec->dato_3,8))    /* Arranque de motor Alimentador Atrás */
{
  cambia(&c_s_byte_3,3);
}

if (prendido(pos_sec->dato_4,1))    /* Arranque de motor Alimentador Adelante */
{
  cambia(&c_s_byte_3,4);
}
if (prendido(pos_sec->dato_4,2))    /* Tomo tiempo inicial_1 */
{
  bios_time = biostime(0, 0L);          /* Tomo tiempo inicial */
  tiempo_ini_1 = (int) bios_time / CLK_TCK;
}
if (prendido(pos_sec->dato_4,3))    /* Tomo tiempo inicial_2 */
{
  bios_time = biostime(0, 0L);          /* Tomo tiempo inicial */
  tiempo_ini_2 = (int) bios_time / CLK_TCK;
}
if (prendido(pos_sec->dato_4,4))    /* Tomo tiempo inicial_3 */
{
  bios_time = biostime(0, 0L);          /* Tomo tiempo inicial */
  tiempo_ini_3 = (int) bios_time / CLK_TCK;
}
if (prendido(pos_sec->dato_4,5))    /* Tomo tiempo inicial_4 */
{
  bios_time = biostime(0, 0L);          /* Tomo tiempo inicial */
  tiempo_ini_4 = (int) bios_time / CLK_TCK;
}
if (prendido(pos_sec->dato_4,6))    /* Tomo tiempo inicial_5 */
{
  bios_time = biostime(0, 0L);          /* Tomo tiempo inicial */
  tiempo_ini_5 = (int) bios_time / CLK_TCK;
}
if (prendido(pos_sec->dato_4,7))    /* Tomo tiempo inicial_6 */
{
  bios_time = biostime(0, 0L);          /* Tomo tiempo inicial */
  tiempo_ini_6 = (int) bios_time / CLK_TCK;
}
textattr(C_DAT);
gotoxy(1,1);
cprintf(" PASO # %03d",num_act_sec);
gotoxy(1,2);
textattr(C_NOR);
```

```
        delay(200);
/*************************************************************************/
                                        /* Salida de datos a la tarjeta */

        outportb(d_s_byte_1,c_s_byte_1);
        outportb(d_s_byte_2,c_s_byte_2);
        outportb(d_s_byte_3,c_s_byte_3);
        outportb(d_s_byte_4,c_s_byte_4);
        outportb(d_s_byte_5,c_s_byte_5);
                                        /* lectura de los puertos   */

        c_e_byte_1=inportb(d_e_byte_1);
        c_e_byte_2=inportb(d_e_byte_2);


/*************************************************************************/
                                        /* Checar si es salida parcial */

    if ((prendido(pos_sec->dato_1,2)))  /* Avance de torreta solo */
            salida_parcial=1;

    if ((prendido(pos_sec->dato_1,2)) &&  (prendido(pos_sec->dato_1,4))) /* H.1 */
            salida_parcial=1;

    if ((prendido(pos_sec->dato_1,2)) &&  (prendido(pos_sec->dato_1,5))) /* H 2 */
            salida_parcial=1;

    if ((prendido(pos_sec->dato_1,2)) &&  (prendido(pos_sec->dato_1,6))) /* A.M */
            salida_parcial=1;

if(salida_parcial)
   {
            do
            {
                                        /* lectura de los puertos   */
              c_e_byte_1=inportb(d_e_byte_1);
              c_e_byte_2=inportb(d_e_byte_2);
                                        /* Checa Microinterruptores de avance parc  */
              if ((prendido(pos_sec->dato_1,2)) &&  (!prendido(c_e_byte_1,4))) /* Avance de torreta y LS3 */
                avance_parcial=1;

              if ((prendido(pos_sec->dato_1,2)) && (prendido(pos_sec->dato_4,2))  &&  (!prendido(c_e_byte_1,4))//
                  && (!prendido(pos_sec->dato_1,6)) ) /* Avance de torreta, tiempo_1 y LS3 y No Avanc. Mec. */


              {
                                                                /* Quito aire de torreta */

/*******/
                cambia(&c_s_byte_1,3);
                cambia(&c_s_byte_2,1);
/********/
                outportb(d_s_byte_1,c_s_byte_1);
                outportb(d_s_byte_2,c_s_byte_2);
                outportb(d_s_byte_3,c_s_byte_3);
                outportb(d_s_byte_4,c_s_byte_4);
                outportb(d_s_byte_5,c_s_byte_5);

                bios_time = biostime(0, 0L);            /* Tomo tiempo inicial_1 */
                tiempo_ini_1 = (int) bios_time / CLK_TCK;
                avance_parcial=1;
              }
/*************************************************/
              if ((prendido(pos_sec->dato_1,2)) && (prendido(pos_sec->dato_4,3))  &&  (!prendido(c_e_byte_1,4)) //
                  && (!prendido(pos_sec->dato_1,6))  ) /* Avance de torreta, tiempo_2 y LS3 y No Avan. Mec  */
              {
                                                                /* Quito aire de torreta */

/*******/
```

```
            cambia(&c_s_byte_1,3);
            cambia(&c_s_byte_2,1);
/********/
            outportb(d_s_byte_1,c_s_byte_1);
            outportb(d_s_byte_2,c_s_byte_2);
            outportb(d_s_byte_3,c_s_byte_3);
            outportb(d_s_byte_4,c_s_byte_4);
            outportb(d_s_byte_5,c_s_byte_5);

            bios_time = biostime(0, 0L);          /* Tomo tiempo inicial_2 */
            tiempo_ini_2 = (int) bios_time / CLK_TCK;
            avance_parcial=1;
          }
/*************************************************/
          if ((prendido(pos_sec->dato_1,2)) && (prendido(pos_sec->dato_4,4))  &&  (!prendido(c_e_byte_1,4)) //
              &&  (!prendido(pos_sec->dato_1,6)) ) /* Avance de torreta, tiempo_3 y LS3 y No Avanc  Mec. */
          {
                                                          /* Quito aire de torreta */

/********/
            cambia(&c_s_byte_1,3);
            cambia(&c_s_byte_2,1);
/********/
            outportb(d_s_byte_1,c_s_byte_1);
            outportb(d_s_byte_2,c_s_byte_2);
            outportb(d_s_byte_3,c_s_byte_3);
            outportb(d_s_byte_4,c_s_byte_4);
            outportb(d_s_byte_5,c_s_byte_5);

            bios_time = biostime(0, 0L);          /* Tomo tiempo inicial_3 */
            tiempo_ini_3 = (int) bios_time / CLK_TCK;
            avance_parcial=1;
          }
/*************************************************/
          if ((prendido(pos_sec->dato_1,2)) && (prendido(pos_sec->dato_4,5))  &&  (!prendido(c_e_byte_1,4)) //
              &&  (!prendido(pos_sec->dato_1,6)) ) /* Avance de torreta, tiempo_4 y LS3 y No Avanc. Mec  */
          {
                                                          /* Quito aire de torreta */

/********/
            cambia(&c_s_byte_1,3);
            cambia(&c_s_byte_2,1);
/********/
            outportb(d_s_byte_1,c_s_byte_1);
            outportb(d_s_byte_2,c_s_byte_2);
            outportb(d_s_byte_3,c_s_byte_3);
            outportb(d_s_byte_4,c_s_byte_4);
            outportb(d_s_byte_5,c_s_byte_5);

            bios_time = biostime(0, 0L);          /* Tomo tiempo inicial_4 */
            tiempo_ini_4 = (int) bios_time / CLK_TCK;
            avance_parcial=1;
          }
/*************************************************/
          if ((prendido(pos_sec->dato_1,2)) && (prendido(pos_sec->dato_4,6))  &&  (!prendido(c_e_byte_1,4))//
              &&  (!prendido(pos_sec->dato_1,6)) ) /* Avance de torreta, tiempo_5 y LS3 y No Avanc. Mec. */
          {
                                                          /* Quito aire de torreta */


/********/
            cambia(&c_s_byte_1,3);
            cambia(&c_s_byte_2,1);
/********/
            outportb(d_s_byte_1,c_s_byte_1);
            outportb(d_s_byte_2,c_s_byte_2);
            outportb(d_s_byte_3,c_s_byte_3);
            outportb(d_s_byte_4,c_s_byte_4);
            outportb(d_s_byte_5,c_s_byte_5);
```

```
            bios_time = biostime(0, 0L);            /* Tomo tiempo inicial_5 */
            tiempo_ini_5 = (int) bios_time / CLK_TCK;
            avance_parcial=1;
          }
/*************************************************/
          if ((prendido(pos_sec->dato_1,2)) && (prendido(pos_sec->dato_4,7)) &&  (!prendido(c_e_byte_1,4)) //
             &&  (!prendido(pos_sec->dato_1,6)) ) /* Avance de torreta, tiempo_6 y LS3 y No Avanc. Mec. */
          {
                                                  /* Quito aire de torreta */

/*******/
            cambia(&c_s_byte_1,3);
            cambia(&c_s_byte_2,1);
/********/
            outportb(d_s_byte_1,c_s_byte_1);
            outportb(d_s_byte_2,c_s_byte_2);
            outportb(d_s_byte_3,c_s_byte_3);
            outportb(d_s_byte_4,c_s_byte_4);
            outportb(d_s_byte_5,c_s_byte_5);

            bios_time = biostime(0, 0L);            /* Tomo tiempo inicial_6 */
            tiempo_ini_6 = (int) bios_time / CLK_TCK;
            avance_parcial=1;
          }
/*************************************************/

        }while((!avance_parcial)  && (tecla_f10==0)  && (tecla_f5==0));
                                  /* Complementar salidas  al puertos */

          if (prendido(pos_sec->dato_1,4))    /* Avance con Hidro No. 1 */
          {
            cambia(&c_s_byte_2,6);
          }
          if (prendido(pos_sec->dato_1,5))    /* Avance con Hidro No  2 */
          {
            cambia(&c_s_byte_2,6);
            cambia(&c_s_byte_2,8);
          }
          if (prendido(pos_sec->dato_1,6))    /* Entrada del avance mecánico */
          {
            cambia(&c_s_byte_2,3);
          }
          if (prendido(pos_sec->dato_1,7))    /* Selección de rango de avance mecánico + lento */
          {
            cambia(&c_s_byte_1,7);
          }
          if (prendido(pos_sec->dato_1,8))    /* Selección de rango de avance mecánico + rápido */
          {
           cambia(&c_s_byte_2,2);
          }
                                  Salidas complementadas */
          outportb(d_s_byte_1,c_s_byte_1);
          outportb(d_s_byte_2,c_s_byte_2);
          outportb(d_s_byte_3,c_s_byte_3);
          outportb(d_s_byte_4,c_s_byte_4);
          outportb(d_s_byte_5,c_s_byte_5);
  } /* Fin del if parcial */

  do                              /* Ejecuta mientra va avanzando y hay elementos */
  {
                                  /* Ciclo Repetitivo */

      des_inf();                  /* Conjunto de Instrucciones a ejecutar */
                                  /* por cada renglón del tablero  */

       if ((prendido(pos_sec->dato_1,2)) && (prendido(pos_sec->dato_4,2))) /* Avance torreta y tiempo_1 */
       {
        if(tiempo_transcurrido(1))
            avance_total=1;
```

```
        }
        if ((prendido(pos_sec->dato_4,2)))    /* tiempo_1 */
        {
         if(tiempo_transcurrido(1))
               avance_total=1;
        }
/*****/
        if ((prendido(pos_sec->dato_1,2)) && (prendido(pos_sec->dato_4,3))) /* Avance torreta y tiempo_2 */
        {
         if(tiempo_transcurrido(2))
               avance_total=1;
        }
        if ((prendido(pos_sec->dato_4,3)))    /* tiempo_2 */
        {
         if(tiempo_transcurrido(2))
               avance_total=1;
        }
/*****/
        if ((prendido(pos_sec->dato_1,2)) && (prendido(pos_sec->dato_4,4))) /* Avance torreta y tiempo_3 */
        {
         if(tiempo_transcurrido(3))
               avance_total=1;
        }
        if ((prendido(pos_sec->dato_4,4)))    /* tiempo_3 */
        {
         if(tiempo_transcurrido(3))
               avance_total=1;
        }
/*****/
        if ((prendido(pos_sec->dato_1,2)) && (prendido(pos_sec->dato_4,5))) /* Avance torreta y tiempo_4 */
        {
         if(tiempo_transcurrido(4))
               avance_total=1;
        }
        if ((prendido(pos_sec->dato_4,5)))    /* tiempo_4 */
        {
         if(tiempo_transcurrido(4))
               avance_total=1;
        }
/*****/
        if ((prendido(pos_sec->dato_1,2)) && (prendido(pos_sec->dato_4,6))) /* Avance torreta y tiempo_5 */
        {
         if(tiempo_transcurrido(5))
               avance_total=1;
        }
        if ((prendido(pos_sec->dato_4,6)))    /* tiempo_5 */
        {
         if(tiempo_transcurrido(5))
               avance_total=1;
        }
/*****/
        if ((prendido(pos_sec->dato_1,2)) && (prendido(pos_sec->dato_4,7))) /* Avance torreta y tiempo_6 */
        {
         if(tiempo_transcurrido(6))
               avance_total=1;
        }
        if ((prendido(pos_sec->dato_4,7)))    /* tiempo_6 */
        {
         if(tiempo_transcurrido(6))
               avance_total=1;
        }
                                    /* lectura de los puertos   */
       c_e_byte_1=inportb(d_e_byte_1);
       c_e_byte_2=inportb(d_e_byte_2);
                                    /* Checa Microinterruptores de avance total */

       if(prendido(pos_sec->dato_1,2)&& (!prendido(c_e_byte_1,6) ) ) /* Torreta Adelante y LS2 */
       {
         avance_total=1;
       }
```

```
        if(prendido(pos_sec->dato_1,3)&& (!prendido(c_e_byte_1,5) ) ) /* Torreta Atras y LS4 */
        {
          avance_total=1;
        }
        if(prendido(pos_sec->dato_2,1)&& (!prendido(c_e_byte_1,8) ) ) /* Frontal Cruzado y LS5 */
        {
          avance_total=1;
        }
        if(prendido(pos_sec->dato_2,2)&& (!prendido(c_e_byte_1,7) ) ) /* Posterior Cruzado y LS6 */
        {
          avance_total=1;
        }
}while(((tecla_f5==0) && (tecla_f10==0))&&(pos_sec->sig !=NULL) && (!avance_total) );

if(tecla_f5)
        tecla_f5=0;

if(tecla_f10)
{
        error=1;
        gotoxy(4,25);
        textattr(C_ERR);
        cprintf("     Ejecución de la secuencia de Operación :    ABORTADA  por operador ");
        delay(TIEMPO*4);
        textattr(C_NOR),
        gotoxy(4,25);
        cprintf("                                                  ");
        o_pos_x=wherex();
        o_pos_y=wherey();
        gotoxy(66,3);
        cprintf("F10: ABORTAR  ");
        textattr(0x1F);
        gotoxy(66,4);
        cprintf("F5: EJECUTAR  ");
        gotoxy(o_pos_x,o_pos_y);
          c_s_byte_1='\0';                    /* Limpia salidas */
          c_s_byte_2='\0',
          c_s_byte_3='\0';
          c_s_byte_4='\0',
          c_s_byte_5='\0'
        break;
 }
 c_s_byte_1='\0';                    /* Limpia salidas */
 c_s_byte_2='\0',
 c_s_byte_3='\0';
 c_s_byte_4='\0',
 c_s_byte_5='\0';
 pos_sec=pos_sec->sig;
 num_act_sec++;                              /* lleva la cuenta de los recorridos + 1 */

}while(pos_sec!=NULL);               /*  Mientras existan nodos en Memoria   */
ini_ptos();                          /* mientras hay elementos, conj. de instrucciones */
addr->p = int9->p;                   /*  Reestable el vector de la interrupción 9 */
tecla_f5 = 0;                        /*  Para las teclas */
tecla_f10 = 0;
o_pos_x=wherex();
o_pos_y=wherey();
gotoxy(66,3);
textattr(C_IND);
cprintf("          "),
textattr(0x1F);
gotoxy(66,4);
cprintf("F5: EJECUTAR  ");
textattr(C_NOR);
gotoxy(1,1);
cprintf("          ");
gotoxy(1,2);
cprintf("                ");
gotoxy(o_pos_x,o_pos_y);
pos_sec=aux_2;
```

```
  num_act_sec=num_act_sec_aux;
  existe_ele_sec(num_act_sec_aux);
  if (error)
    return(0);
  return(1);
}

void interrupt aplicacion(void)
{
  char far *t = (char far *) 1050;          /* Direccion de puntero de cabeza */
  geninterrupt(60);
  if( *t != *(t+2) )                        /* Si no vacia */
  {
    t += *t-30+5;                           /* Avanza a la posición caracter */
    if(*t == 63)                                    /* Si la tecla es F5 */
    {
          bioskey(0);                       /* Limpia la tecla F5 */
          tecla_f5=1;
    }
    else
          if(*t == 68)                      /* Si la tecla es f10 */
          {
            bioskey(0);                     /* Limpia la tecla f10 */
            tecla_f10=1;
          }
          else
          {
            bioskey(0);                     /* Limpia el  buffer */
          }
  }
}

                                            /* Checa si esta prendido un bit de un byte */
short prendido(unsigned char  dato_eval ,short x)
{
 unsigned char dato_aux='\0';

 switch(x)
 {
   case 1.
          dato_aux=0x01; break;   /* 0b00000001 */
   case 2:
          dato_aux=0x02; break;
   case 3:
          dato_aux=0x04; break;
   case 4.
          dato_aux=0x08, break;
   case 5:
          dato_aux=0x10; break;
   case 6:
           dato_aux=0x20; break;
   case 7:
          dato_aux=0x40; break;
   case 8:
          dato_aux=0x80; break;  /* 0b10000000 */
 }
 if(dato_eval & dato_aux)
   return(1);
 return(0);
}

void cambia(unsigned char * dato_mod,short x)
{
 unsigned char dato_aux='\0';

 switch(x)
 {
   case 1:
          dato_aux=0x01; break;   /* 0b00000001 */
   case 2:
```

```
            dato_aux=0x02; break;
    case 3:
            dato_aux=0x04; break;
    case 4:
            dato_aux=0x08; break;
    case 5:
            dato_aux=0x10; break;
    case 6:
            dato_aux=0x20; break;
    case 7:
            dato_aux=0x40; break;
    case 8:
            dato_aux=0x80; break;  /* 0b10000000 */
  }
  *dato_mod = *dato_mod ^ dato_aux;
}

int input_x_y(struct menu *ppmenu)
{
  unsigned int key , i;
  textattr(C_NOR);
  gotoxy(ppmenu->x+strlen(ppmenu->valor),ppmenu->y+1);
  for (i=0+strlen(ppmenu->valor) ;  (i<ppmenu->len)  ; )
  {
    key = getch();

    switch(key)
    {
            case L_ESC:
                    return(NO_OK);
            case L_TAB:
                    textattr(C_NOR);
                    return(key);
            case L_RET:                             /*  Regreso <- */
              if(i > 0)
              {
               return(L_RET);
               i--;
               ppmenu->valor[i]='\0';
               textattr(C_NOR);
               cprintf(" ");
               break;
              }
              else
              {
               break;
              }
            case L_ENTER:
              if(i > 0)                     /*  Para asegurar que tecleen mínimo una letra */
              {
               ppmenu->valor[i]='\0';
               textattr(C_NOR);
               for (; i < ppmenu->len ;i++)
               {
                       gotoxy(ppmenu->x+i,ppmenu->y+1);
                       cprintf(" ");
               }
               textattr(C_NOR);
               return(OK);
              }
              else
              {
               break;
              }
              default:                      /* 48 -> 57 para números */
               if(ppmenu->tipo==1)
               {
                if( (key > 47 && key <= 57) || (key == 32) )  /* Acepta números y el espacio */
                {
                       gotoxy(ppmenu->x+i,ppmenu->y+1);
```

```
                cprintf("%c",key);
                ppmenu->valor[i]=key;
                ppmenu->valor[i+1]='\0';
                i++;
            }
          }
                                    /* 32 -> 125 para cadenas */
          if(ppmenu->tipo==0)
          {
            if( (key >= 32 && key <= 125) )
            {
                gotoxy(ppmenu->x+i,ppmenu->y+1);
                cprintf("%c",key);
                ppmenu->valor[i]=key;
                ppmenu->valor[i+1]='\0';
                i++;
            }
          }
    } /* Fin del switch */
  }/* Fin del For */
  textattr(C_NOR);
  return(OK);
}

void ini_ptos(void)
{                                   /* Inicialización de salidas en puertos */
   outportb(d_s_byte_1,0x00);
   outportb(d_s_byte_2,0x00);
   outportb(d_s_byte_3,0x00);
   outportb(d_s_byte_4,0x00);
   outportb(d_s_byte_5,0x00);
}

int checa_sec(void)
{
   unsigned int num_act_sec_aux;
   char texto[90];
   int checado=1;
   short fin_de_secuencia=0;
   memset(texto,'\0',90);
   aux_2=pos_sec;                   /* Apunta al primer elemento */
   pos_sec=ini_sec;
   num_act_sec_aux=num_act_sec;
   num_act_sec=1;
   do                               /* Chequeo de errores */
   {
          if (prendido(pos_sec->dato_1,1))
             fin_de_secuencia=1;
          if (prendido(pos_sec->dato_1,2) && prendido(pos_sec->dato_1,3) ) /* Avance y Regreso Torreta */
          {
           checado=0;
           sprintf(texto," ERROR en las señales de la secuencia      El paso es el Número: %03d", num_act_sec);
           pausa(texto);
           fin_de_secuencia=1;
           break;
          }
          if (prendido(pos_sec->dato_1,4) && prendido(pos_sec->dato_1,5) ) /* Hidro1 e Hodro2 */
          {
           checado=0;
           sprintf(texto," ERROR en las señales de la secuencia      El paso es el Número: %03d", num_act_sec);
           pausa(texto);
           fin_de_secuencia=1;
           break;
          }
          if (prendido(pos_sec->dato_1,4) && prendido(pos_sec->dato_1,6) ) /* Hidro1 e Avan. Mec.*/
          {
           checado=0;
           sprintf(texto," ERROR en las señales de la secuencia.     El paso es el Número: %03d", num_act_sec);
           pausa(texto);
           fin_de_secuencia=1;
```

```
            break;
           }
           if (prendido(pos_sec->dato_1,5) && prendido(pos_sec->dato_1,6) ) /* Hidro2 e Avan  Mec.*/
           {
            checado=0;
            sprintf(texto," ERROR en las señales de la secuencia.     El paso es el Número  %03d", num_act_sec),
            pausa(texto),
            fin_de_secuencia=1;
            break;
           }
           if (prendido(pos_sec->dato_1,8) && prendido(pos_sec->dato_1,7) ) /* Rango + Rápido y + Lento de Avance Mec. */
           {
            checado=0;
            sprintf(texto," ERROR en las señales de la secuencia.     El paso es el Número: %03d", num_act_sec);
            pausa(texto);
            fin_de_secuencia=1;
            break;
           }
     des_inf();
     pos_sec=pos_sec->sig;
     num_act_sec++;                    /* lleva la cuenta de los recorridos + 1 */

   }while(pos_sec!=NULL);              /* mientras hay elementos */
   pos_sec=aux_2;
   num_act_sec=num_act_sec_aux,
   existe_ele_sec(num_act_sec_aux);
   if(!fin_de_secuencia)
   {
    sprintf(texto," La señal HOME debe aparecer en el paso final de ejecución     .....");
    pausa(texto);
    checado=0,
   }
   pausa("  Fin del checado de la secuencia...  Presiona una tecla para ejecutar ...");
   return(checado);
}

void menu_param(void)
{
   char tiempo_elegido[3];
   char ciclos_elegido[3];
   int lectura=0;
   int salida=0,
   char res;
   marco(3,3,78,5,0x0B);
   marco(1,1,80,24,0x0C);
   gotoxy(18,4 );
   cprintf("T A B L E R O   D E   PARÁMETROS");
   gotoxy(5, 10);                    /* Pintado */
   cprintf("Tiempo_1 : ");
   gotoxy(5+10, 10);
   cprintf("[   ] Seg.");
   gotoxy(5, 14);
   cprintf("Tiempo_2 : "),
   gotoxy(5+10, 14),
   cprintf("[   ] Seg.");
   gotoxy(29, 10);
   cprintf("Tiempo_3 : ");
   gotoxy(29+10, 10);
   cprintf("[   ] Seg.");
   gotoxy(29, 14);
   cprintf("Tiempo_4 : ");
   gotoxy(29+10, 14);
   cprintf("[   ] Seg.");
   gotoxy(53, 10);
   cprintf("Tiempo_5 : "),
   gotoxy(53+10, 10);
   cprintf("[   ] Seg.");
   gotoxy(53, 14);
   cprintf("Tiempo_6 : ");
   gotoxy(53+10, 14);
```

```
cprintf("[   ] Seg.");
gotoxy(5, 18);
cprintf("Num. de Ciclos:");
gotoxy(5+15, 18);
cprintf("[   ]    (  Numero de ciclos  %d  ) ",ciclos_ele);
pmenu= &var_menu[8];                          /* Leer números_ciclos */
if(strlen(pmenu->valor)!=0)
{
 gotoxy(5+15+1, 18);
 cprintf("%s",pmenu->valor);
}

pmenu= &var_menu[7];                          /* Leer tiempo_6 */
if(strlen(pmenu->valor)!=0)
{
  gotoxy(53+10+1, 14);
  cprintf("%s",pmenu->valor);
}
pmenu= &var_menu[6];                          /* Leer tiempo_5 */
if(strlen(pmenu->valor)!=0)
{
  gotoxy(53+10+1, 10);
  cprintf("%s",pmenu->valor);
}
pmenu= &var_menu[5];                          /* Leer tiempo_4 */
if(strlen(pmenu->valor)!=0)
{
  gotoxy(29+10+1, 14);
  cprintf("%s",pmenu->valor);
}
pmenu= &var_menu[4];                          /* Leer tiempo_3 */
if(strlen(pmenu->valor)!=0)
{
  gotoxy(29+10+1, 10);
  cprintf("%s",pmenu->valor),
}
pmenu= &var_menu[3];                          /* Leer tiempo_2 */
if(strlen(pmenu->valor)!=0)
{
  gotoxy(5+10+1, 14),
  cprintf("%s",pmenu->valor);
}
do
{
  pmenu= &var_menu[2];                        /* Leer tiempo_1 */

  if(strlen(pmenu->valor)!=0)
  {
        gotoxy(5+10+1, 10);
        cprintf("%s",pmenu->valor);
  }
  lectura=input_x_y_otro(pmenu),
  strcpy(tiempo_elegido,pmenu->valor);
  tiempo_ele_1=atoi(tiempo_elegido);
  gotoxy(25,11);
  cprintf("    ");
  gotoxy(15,11);
  cprintf("    T_1: %d",tiempo_ele_1);
 if ((lectura==L_TAB)|| (lectura==OK))
 {
  pmenu= &var_menu[3];                        /* Leer tiempo_2 */
  if(strlen(pmenu->valor)!=0)
  {
        gotoxy(5+10+1, 14);
        cprintf("%s",pmenu->valor);
  }
   input_x_y_otro(pmenu);
   strcpy(tiempo_elegido,pmenu->valor);
   tiempo_ele_2=atoi(tiempo_elegido);
   gotoxy(25,15);
```

```
 cprintf("    "),
 gotoxy(15,15);
 cprintf("    T_2: %d",tiempo_ele_2);
}
                              /*********************/
if ((lectura==L_TAB)|| (lectura==OK))
{
 pmenu= &var_menu[4];                    /* Leer tiempo_3 */
 if(strlen(pmenu->valor)!=0)
 {
        gotoxy(29+10+1, 10);
        cprintf("%s",pmenu->valor);
 }
 input_x_y_otro(pmenu),
 strcpy(tiempo_elegido,pmenu->valor),
 tiempo_ele_3=atoi(tiempo_elegido),
 gotoxy(49,11);
 cprintf("    ");
 gotoxy(39,11);
 cprintf("    T_3: %d",tiempo_ele_3),
}
                              /*********************/


if ((lectura==L_TAB)|| (lectura==OK))
{
 pmenu= &var_menu[5],                    /* Leer tiempo_4 */
 if(strlen(pmenu->valor)!=0)
 {
        gotoxy(29+10+1, 14);
        cprintf("%s",pmenu->valor);
 }
 input_x_y_otro(pmenu);
 strcpy(tiempo_elegido,pmenu->valor);
 tiempo_ele_4=atoi(tiempo_elegido);
 gotoxy(49,15);
 cprintf("    "),
 gotoxy(39,15);
 cprintf("    T_4: %d",tiempo_ele_4);
}
                              /*********************/
if ((lectura==L_TAB)|| (lectura==OK))

 pmenu= &var_menu[6],                    /* Leer tiempo_5 */
 if(strlen(pmenu->valor)!=0)
 {
        gotoxy(53+10+1, 10);
        cprintf("%s",pmenu->valor);
 }
 input_x_y_otro(pmenu);
 strcpy(tiempo_elegido,pmenu->valor),
 tiempo_ele_5=atoi(tiempo_elegido);
 gotoxy(73,11);
 cprintf("    ");
 gotoxy(63,11);
 cprintf("    T_5: %d",tiempo_ele_5);
}
                              /*********************/
if ((lectura==L_TAB)|| (lectura==OK))
{
 pmenu= &var_menu[7];                      /* Leer tiempo_6 */
 if(strlen(pmenu->valor)!=0)
 {
        gotoxy(53+10+1, 14);
        cprintf("%s",pmenu->valor);
 }
 input_x_y_otro(pmenu);
 strcpy(tiempo_elegido,pmenu->valor);
 tiempo_ele_6=atoi(tiempo_elegido);
 gotoxy(73,15),
```

```
    cprintf("     ");
    gotoxy(63,15);
    cprintf("    T_6: %d",tiempo_ele_6);
   }
   if ((lectura==L_TAB)|| (lectura==OK))
   {
    pmenu= &var_menu[8];                    /* Leer número_de_ciclos */
   if(strlen(pmenu->valor)!=0)
   {
          gotoxy(5+15+1, 18);
          cprintf("%s",pmenu->valor);
   }
    input_x_y_otro(pmenu);
    strcpy(ciclos_elegido,pmenu->valor);
    ciclos_ele=atoi(ciclos_elegido);
    gotoxy(52,18);
    cprintf("   ");
    gotoxy(52,18);
    cprintf("%s",pmenu->valor);
   }
   fflush(stdin);
   res=pausa("Estan correctos los datos ? ( S/N )");
   if(toupper(res)=='S')
    salida=1;
  }while(!salida);
}

int input_x_y_otro(struct menu *ppmenu)
{
 unsigned int key , i;
 textattr(C_NOR);
 gotoxy(ppmenu->x,ppmenu->y+1);
 for (i=0 ; i<ppmenu->len ; )
 {
   key = getch();

   switch(key)
   {
          case L_TAB:
                textattr(C_NOR);
            gotoxy(ppmenu->x,ppmenu->y+1);
            cprintf("%s",pmenu->valor);
                return(key);
          case L_RET:                       /*  Regreso <- */
            if(i > 0)
            {
             i--;
             ppmenu->valor[i]='\0';
             textattr(C_NOR);
             gotoxy(ppmenu->x+i,ppmenu->y+1);
             cprintf(" ");
             gotoxy(ppmenu->x+i,ppmenu->y+1);
             break;
            }
            else
            {
             break;
            }
          case L_ENTER:

            if(i > 0 )
            {
             ppmenu->valor[i]='\0';
             textattr(C_NOR);
             for (; i < ppmenu->len ,i++)
             {
                gotoxy(ppmenu->x+i,ppmenu->y+1);
                cprintf(" ");
             }
             textattr(C_NOR);
```

```
            gotoxy(ppmenu->x,ppmenu->y+1);
            cprintf("%s",pmenu->valor);
            return(OK);
          }
          else
          {
            if(i == 0  &&  (strlen(ppmenu->valor))!=0)
            {
                  return(OK);
                  break;
            }
          }
          default:
          if(ppmenu->tipo==1)
          {
           if( (key > 47 && key <= 57) || (key == 32) )  /* Acepta números y el espacio */
           {
                  gotoxy(ppmenu->x+i,ppmenu->y+1);
                  cprintf("%c",key);
                  ppmenu->valor[i]=key;
                  ppmenu->valor[i+1]='\0';
                  i++;
           }
          }
          if(ppmenu->tipo==0)
          {
           if( (key >= 32 && key <= 125) )
           {
                  gotoxy(ppmenu->x+i,ppmenu->y+1);
                  cprintf("%c",key);
                  ppmenu->valor[i]=key;
                  ppmenu->valor[i+1]='\0';
                  i++;
           }
          }
    } /* Fin del switch */
  }/* Fin del For */
  textattr(C_NOR);
  return(OK);
}

void ini_tiempos(void)
{
    pmenu= &var_menu[2];                    /* Leer tiempo_1 */
    pmenu->valor[0]='0';
    pmenu->valor[1]='\0';
    tiempo_ele_1=0;
    pmenu= &var_menu[3];                    /* Leer tiempo_2 */
    pmenu->valor[0]='0';
    pmenu->valor[1]='\0';
    tiempo_ele_2=0;
    pmenu= &var_menu[4];                    /* Leer tiempo_3 */
    pmenu->valor[0]='0';
    pmenu->valor[1]='\0';
    tiempo_ele_3=0;
    pmenu= &var_menu[5];                    /* Leer tiempo_4 */
    pmenu->valor[0]='0';
    pmenu->valor[1]='\0';
    tiempo_ele_4=0;
    pmenu= &var_menu[6];                    /* Leer tiempo_5 */
    pmenu->valor[0]='0';
    pmenu->valor[1]='\0';
    tiempo_ele_5=0;
    pmenu= &var_menu[7];                    /* Leer tiempo_6 */
    pmenu->valor[0]='0';
    pmenu->valor[1]='\0';
    tiempo_ele_6=0;
}
```

```
short tiempo_transcurrido(short tiempo_ele)
{
  int tiempo_trans=0;  bios_time = biostime(0, 0L);
  tiempo_act = (int) bios_time / CLK_TCK;     /* Actualización del tiempo */
  if ( tiempo_ele == 1)
  {
    tiempo_trans=tiempo_act - tiempo_ini_1;
    if (tiempo_trans >= tiempo_ele_1)
          return(1);
  }
  if ( tiempo_ele == 2)
  {
    tiempo_trans=tiempo_act - tiempo_ini_2;
    if (tiempo_trans >= tiempo_ele_2)
          return(1);
  }
  if ( tiempo_ele == 3)
  {
    tiempo_trans=tiempo_act - tiempo_ini_3;
    if (tiempo_trans >= tiempo_ele_3)
          return(1);
  }
  if ( tiempo_ele == 4)
  {
    tiempo_trans=tiempo_act - tiempo_ini_4;
    if (tiempo_trans >= tiempo_ele_4)
          return(1);
  }
  if ( tiempo_ele == 5)
  {
    tiempo_trans=tiempo_act - tiempo_ini_5;
    if (tiempo_trans >= tiempo_ele_5)
          return(1);
  }
  if ( tiempo_ele == 6)
  {
    tiempo_trans=tiempo_act - tiempo_ini_6;
    if (tiempo_trans >= tiempo_ele_6)
          return(1);
  }
  return(0);
}
```

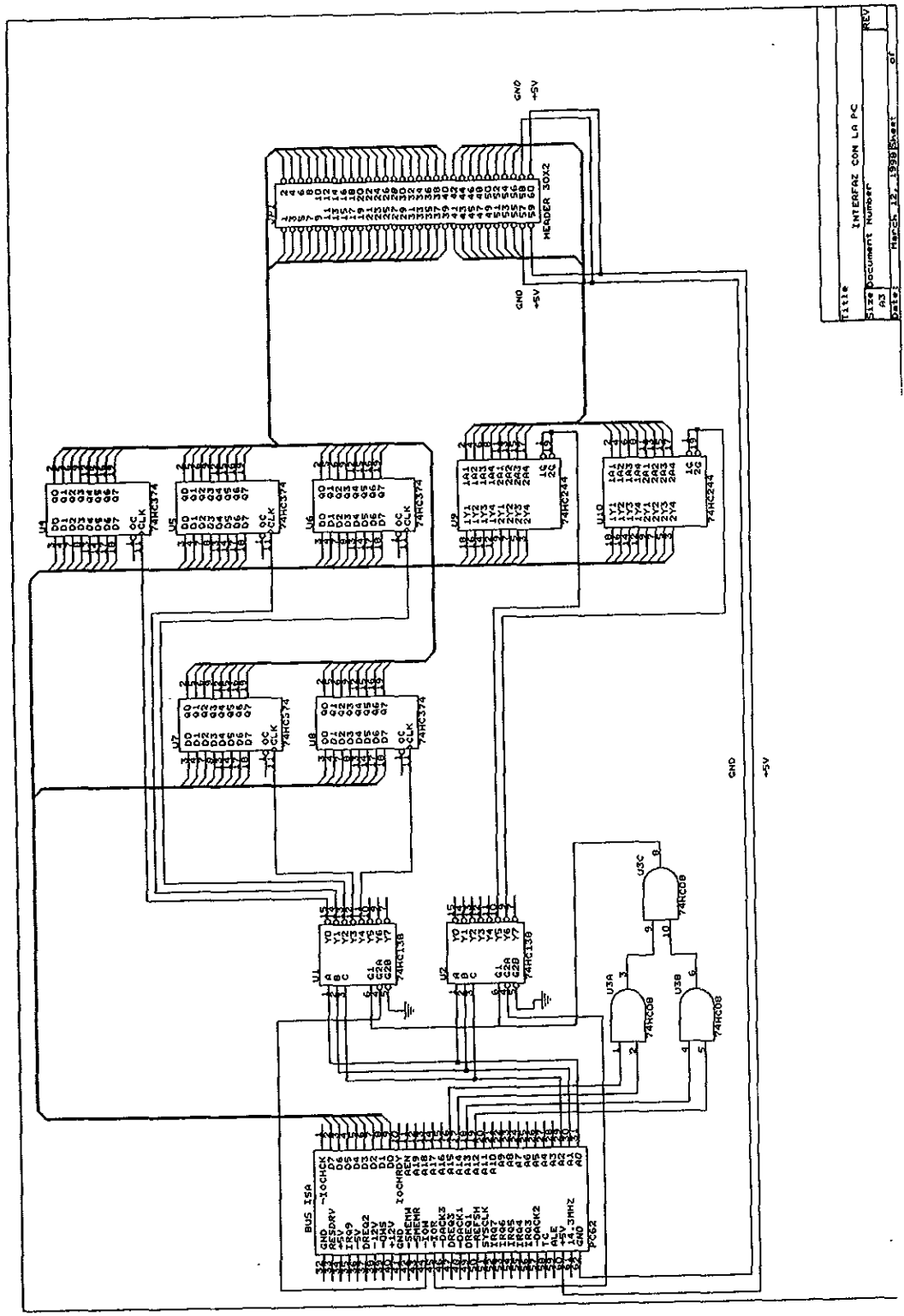
APÉNDICE "C"

DIAGRAMAS DE

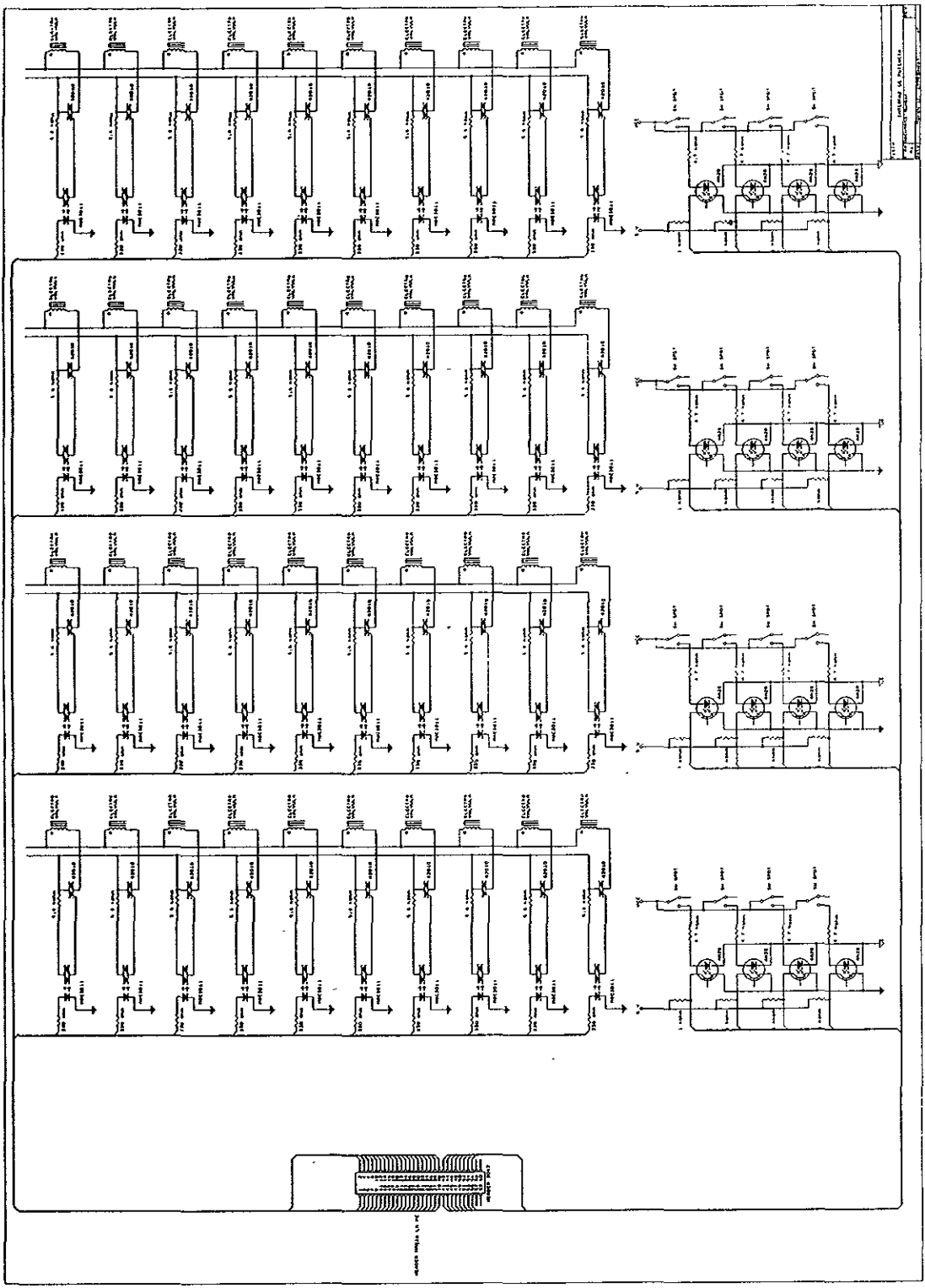
LOS CIRCUITOS

DE LAS INTERFACES

INTERFAZ CON LA PC


INTERFAZ CON LA PC
Title
Size
A3
Document Number
REV
Date: March 12, 1998
Sheet
of
HEADER 30X2
74HC374
74HC244
74HC138
74HC08
BUS ISA
GND
+5V

INTERFAZ DE POTENCIA



BIBLIOGRAFÍA

BIBLIOGRAFÍA

- The Intel Microprocessors 8086/8088, 80186, 80286, 80386 and 80486. Arquitecture, Programming and Interfacing.
  Barry B.Brey.
  Edit. Prentice Hall.

- Programming and Interfacing the 8086/8088 Microprocessor.
  Roy Godoy.
  Maxwell Macmillan Publishing Company.

- Upgradind and Reparing PC´s 5th. Edition.
  Scott Mueller.
  Edit. QUE.

- Diseño Elétronico.
  Savani, Roden, Carpenter.
  Edit. Addison-Wesley Iberoamericana.

- Enciclopedia de la Electrónica Ingeniería y Técnica.
  Depto. de Ingeniería Eléctrica y en Computación de la Universidad de Florida.
  Edit. Oceano/Centrum.

- Process/Industrial and Controls Handbook.
  Geometric and Motion Sensors.

- Herbert Lathe, Operating Setting Handbook.

- An Introduction to CNC Machinning and Programming.
  David Givs and Thomas M. Crandell.

- C++ Library Reference. V 3.1.
  Borland.

- C: Guía para usuarios expertos.
  Mc. Graw Hill.

- Control Engineering.
  Junio de 1997, Vol. 44 Núm.10.
  "The Personal Computer Takes Control ".

- PC-Magazine.
  Septiembre de 1997.
  "The Best Processor".