ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

DISEÑO DE UN CONTROLADOR BASADO EN UN COMPUTADOR PERSONAL PARA LA PERFORACION AUTOMATICA DE CIRCUITOS IMPRESOS MONTADO EN LOS LABORATORIOS DEL COLEGIO SALESIANO DOMINGO SAVIO

TESIS DE GRADO

Previo a la obtención del Título de

INGENIERO EN ELECTRICIDAD ESPECIALIZACION ELECTRONICA

Presentada por :

LUIS SILVIO CORDOVA RIVADENEIRA

Guayaquil - Ecuador

1999

AGRADECIMIENTO

AGRADEZCO A DIOS POR GUIAR MIS PASOS POR EL CAMINO DEL BIEN.

A MI MADRE POR HABERME APOYADO A LO LARGO DE TODA MI VIDA DE ESTUDIANTE. AL ING. HUGO VILLAVICENCIO DIRECTOR DE MONOGRAFIA, POR SU PACIENCIA Y CONFIANZA EN MI. A MIS BUENOS AMIGOS ANTONIO GARCIA, CRISTIAN SANTI Y JOSE MANUEL PANIAGUA POR SU AYUDA INCONDICIONAL PARA LA ELABORACION DEL PRESENTE TRABAJO. Y A MI ESPOSA CRISTINA POR DARME SIEMPRE FUERZA PARA SEGUIR ADELANTE.

DEDICATORIA

DEDICO EL PRESENTE TRABAJO MONOGRAFICO A TODA MI FAMILIA, ESPECIALMENTE A MI ESPOSA CRISTINA Y A MI HIJO LUIS STEVEN POR SU PACIENCIA Y COMPRENSION, A MIS MAESTROS Y A TODOS MIS COMPAÑEROS, ESPERANDO QUE EL MISMO SIRVA COMO CONSULTA EN LOS FUTUROS TRABAJOS MONOGRAFICOS.

TRIBUNAL

ING. ARMANDO ALTAMIRANO

SUB-DECANO DE LA FACULTAD

DE INGENIERIA ELECTRICA

ING. HUGO VILLAVICENCIO

DIRECTOR DE TESIS

ING. ALBERTO LARCO

MIEMBRO PRINCIPAL

DEL TRIBUNAL

ING. NELSON LAVEDRA

MIEMBRO PRINCIPAL

DEL TRIBUNAL

RESUMEN

El presente trabajo monográfico trata sobre el diseño y construcción de un Controlador basado en un computador personal, el cual permite perforar tarjetas de circuito impreso de una manera más eficiente, rápida y automática .

El controlador consta de una circuitería electrónica conectada a un computador, y de un programa elaborado en Lenguaje "C", para complementar el control.

La circuitería electrónica se encarga del control de los movimientos de los motores de paso para los ejes X, Y y Z , del motor DC para la perforación de la tarjeta y además de la protección de los límites de carrera de los ejes antes mencionados.

La circuitería electrónica tiene también protecciones de sobrecorriente en cualquiera de los tres ejes por si se diera el caso.

Además existe una tarjeta de interfase conectada en un slot del computador, la cual posee 2 integrados 8255 o Interfase Programable Paralela (PPI), y 2 integrados 8253 que son contadores descendentes utilizados para el conteo de los pulsos de los motores de paso.

El programa realizado en Lenguaje "C" es capaz de leer las coordenadas de los puntos de perforación de un archivo .PCB o sea de TANGO, programa para realizar el trazado de las pistas de un circuito electrónico, y transformarlo a un formato propio de la máquina, luego la ordena y por último manda esta información a la circuitería electrónica para lograr la perforación final del circuito.

El programa además puede mover los ejes manualmente mediante las teclas direccionales y grabar las coordenadas de otra tarjeta ya perforada que se monte sobre la

DECLARACION EXPRESA

"LA RESPONSABILIDAD POR LOS HECHOS, IDEAS Y DOCTRINAS EXPUESTAS EN ESTA TESIS, ME CORRESPONDEN EXCLUSIVAMENTE; Y, EL PATRIMONIO INTELECTUAL DE LA MISMA, A LA ESCUELA SUPERIOR POLITECNICA DEL LITORAL"

(REGLAMENTO DE EXAMENES Y TITULOS PROFESIONALES DE LA ESPOL)

LUIS SILVIO CORDOVA RIVADENEIRA

mesa de trabajo, una vez obtenida esta información se monta una nueva tarjeta y se procede a taladrarla.

Todo el sistema puede ser alimentado por la red eléctrica ya sea en 110v. o con 220v.

INDICE GENERAL

INDICE DE TABLAS Y GRAFICOS

TABLAS

CAPITULO 1
TARJETA DE INTERFASE

CAPITULO 2
MOTOR DE PASO

CAPITULO 4
CARACTERISTICAS DE LA FUENTE DE PODER

GRAFICOS

CAPITULO 2
MOTOR DE PASO

CAPITULO 3
DISPOSITIVOS SENSORES

CAPITULO 4
CARACTERISTICAS DE LA FUENTE DE PODER

CAPITULO 5
ESPECIFICACIONES DEL PROGRAMA

CAPITULO 6
MANUAL DEL USUARIO

CAPITULO 1

TARJETA DE INTERFASE

En una máquina controlada por computadora, es de gran importancia mantener una muy buena comunicación en ambos sentidos, lo que nos permite conseguir un alto grado de precisión en la tarea que se pretende efectuar.

Esta comunicación es lograda mediante un circuito de interfase, el cual puede enviar o recibir datos ya sea en forma serial o paralela, dependiendo de los requerimientos básicos de la máquina.

En el siguiente capitulo, detallaremos el circuito de interfase que se empleo para la comunicación, computadora-máquina.

PUERTO DE IMPRESORA CENTRONIC

Al analizar de que forma sería la comunicación entre la máquina y el computador, me detuve a pensar en las dos posibilidades que me brinda el computador, las cuales son, serial y paralelo.

En el caso de elegir la comunicación serial surgieron muchas dificultades, como el de lograr la sincronización de los datos enviados-recibidos, y por ende la elaboración de un circuito externo para la detección y envío de las señales, desde y hacia el computador.

Por esta razón se optó por emplear el puerto de la impresora Centronic, el cual es un puerto paralelo que poseen todas las computadoras, no difieren de una marca a otra, y es muy sencillo de manejar.

ESPECIFICACIONES DEL PUERTO CENTRONIC O PARALELO

El puerto Centronic o paralelo es uno de los puertos encontrados en una computadora, con una gran cantidad de aplicaciones, pero son en realidad las impresoras, sean estas de

cualquier tipo o marca, las que sacan mayor provecho de él, es por esta razón que a continuación detallaremos cada uno de los pines encontrados en el conector DB25, vistos con aplicación a las impresoras.

Las señales usadas en el puerto centronic están descritas en la tabla 1.1

Señal	Función	Pin # DB-25
Strobe*	1.5 us. pulso para el reloj del dato desde el procesador a la impresora.	1
Dato 0	Bit 0 de salida del byte de dato	2
Dato 1	Bit 1 de salida del byte de dato	3
Dato 2	Bit 2 de salida del byte de dato	4
Dato 3	Bit 3 de salida del byte de dato	5
Dato 4	Bit 4 de salida del byte de dato	6
Dato 5	Bit 5 de salida del byte de dato	7
Dato 6	Bit 6 de salida del byte de dato	8
Dato 7	Bit 7 de salida del byte de dato	9
ACK	Entrada al computador desde la impresora Bajo indica que el byte de datos fue recibido	10
Busy	Entrada al computador desde la impresora. Alto indica que la impresora no puede recibir datos	11
Papel vacío	Entrada al computador desde la impresora	12
Select	Entrada al computador desde la impresora.	13
Auto FD*	Salida a la impresora	14
Error	Entrada al computador desde la impresora.	15
Init	Salida a la impresora para limpiar el buffer y resetear la impresora.	16
SLCT IN*	Salida a la impresora, computador listo	17
Ground	Señal común de tierra	18-25

* Estas señales son activadas con bajo.

Tabla 1.1 Señales del Puerto Centronic

Como se puede observar en la tabla 1.1, solo podemos utilizar un total de 17 señales entre salidas y entradas, distribuidas en 12 señales de salida y 5 señales de entrada.

Es debido al escaso número de señales disponibles y al elevado número de señales requeridas, que opté por no utilizar el puerto centronic comúnmente llamado paralelo. En vista de esta situación decidí implementar la comunicación en base a una interfase programable paralela PPI, la cual me permite una cantidad mayor de señales de entrada y salida requeridas para la comunicación máquina-computadora.

INTERFASE PROGRAMABLE 8255

La interfase programable 8255 es una tarjeta de comunicación , con la que fácilmente se puede controlar un proceso determinado, con la ayuda de un computador.

Esta interfase es capaz de manejar un total de 48 señales, las que pueden ser programadas como entradas o salidas dependiendo las necesidades requeridas.

Estas 48 señales son obtenidas mediante 2 integrados que son las interfases paralelas programables (PPI) 8255, los cuales controlan 24 señales de entrada o salida cada una.

Además, esta tarjeta posee 3 contadores independientes de 16 bit de conteo cada uno, con pulsos de reloj arriba de los 2Mhz. encapsulados en un mismo integrado llamado 8253. Posee también 16 leds que sirven como indicadores de nivel de entrada o salida de datos.

Esta tarjeta va ubicada en uno de los slot perteneciente al computador, es decir se engancha a la barra de datos, dirección y control de los microprocesadores de la familia INTEL.

La tarjeta posee un DECODER, que consiste en una lógica capaz de habilitar a cualquiera de los integrados presentes, o sea, al 8253 o algunos de los dos 8255, todo esto gracias a una correcta programación de la tarjeta.

El 8253 es un temporizador/contador programable, especificamente diseñado para usarse con el sistema microcomputador de INTEL, y va a ser el encargado de contar los pulsos recibidos por los motores de paso.

DESCRIPCIÓN FUNCIONAL

Para describir el funcionamiento de la tarjeta de interfase, es necesario analizar por partes su modo de operación. Es decir, se describirá al 8255, que es propiamente la interfase paralela programable y luego al contador programable 8253 .

INTERFASE PARALELO PROGRAMABLE (8255)

El 8255 consta de 3 puertos de entrada o salida paralelo, que se los denomina PA, PB, PC. Las 8 líneas del puerto PC se dividen en dos grupos de 4 líneas cada uno, que pueden trabajar conjuntamente con el puerto PA y con el puerto PB.

Cada puerto tiene características específicas, lo que proporciona una gran flexibilidad a este circuito integrado.

Internamente el 8255 dispone de 4 registros internos: 3 para almacenar información que entra o sale de los puertos, y uno dedicado más al control general del integrado.

Mediante la programación del registro de control, se configuran los puertos, y el funcionamiento del PPI, es decir, que puertos serán los de salida y cuales los de entrada, además de indicar el modo de funcionamiento con el que se va a trabajar.

Inicialmente después de un reset, todas las líneas de entrada y salida se configuran como entradas.

FUNCION DE LOS PINES DEL 8255

Aparte de las 24 señales destinadas para la entrada y salida de datos en los 3 puertos, el PPI utiliza los restantes pines para los fines siguientes.

WR *: Señal de escritura en el PPI, que recibe desde el CPU*

RD *: Señal de lectura en el PPI.*

CS *: Pin destinado para la selección del chip.*

IO/M*: Indica si se accede al mapa de entrada o al mapa de memoria*

RESET*:Pone a cero el registro de control y configura como entrada a todas las líneas de los puertos.*

A0-A1: *Selecciona cual de los registros internos del PPI se procede a leer o escribir. Dichos elementos son: PA, PB, PC y registros de control. Estos pines son conectados a los bit menos significativos de la barra de dirección.*

La operación básica del 8255 se observa en la tabla 1.2

A1	*A0*	*RD*	*WR*	*CS*	*operación de entrada (lectura)*
0	*0*	*0*	*1*	*0*	*Puerto A - barra de datos*
0	*1*	*0*	*1*	*0*	*Puerto B - barra de datos*
1	*0*	*0*	*1*	*0*	*Puerto C - barra de datos*
					Operación de salida (escritura)
0	*0*	*1*	*0*	*0*	*Barra de datos - Puerto A*
0	*1*	*1*	*0*	*0*	*Barra de datos - Puerto B*
1	*0*	*1*	*0*	*0*	*Barra de datos - Puerto C*
1	*1*	*1*	*0*	*0*	*Barra de datos - control*
					Función de habilitación
X	*X*	*X*	*X*	*1*	*Barra de datos - 3 estados*
1	*1*	*0*	*1*	*0*	*Condición ilegal*
X	*x*	*1*	*1*	*0*	*Barra de datos - 3 estados*

Tabla 1.2 *Operación básica del 8255*

PROGRAMA DE LA PALABRA DE CONTROL

La palabra de control debe ser programada según los requerimientos del trabajo a realizar, es por esta razón que explicaré, lo que pretendo obtener con la ayuda del PPI.

Como primer paso detallaré el número de entradas y salidas requeridas por el proyecto, para de esta manera poder definir que puertos utilizaré como entrada, y cual como salida, de datos

ENTRADAS

Como señales de entrada, tenemos las señales provenientes de los sensores de fines de carrera, montados a ambos extremos de cada uno de los tres ejes utilizados, lo que suman 6

entradas. Estas 6 entradas las tomo de la puerta PA del primer PPI, a la que posteriormente llamaré P1A.

Existen también tres señales provenientes de los contadores, las que nos indican el fin de conteo de los valores asignados a los contadores, y por ende el paro de los motores de paso para cada uno de los tres ejes. Estas 3 entradas las tomo de la puerta PB del segundo PPI, a la que posteriormente llamaré P2B.

En total necesito 9 señales de entrada.

SALIDAS

Como señales de salida se contabilizan 12 las que se describen a continuación:

Cuatro señales que controlan la velocidad de los ejes X y Y. Estas señales salidan del puerto P1B. Tres señales que indican el arranque de los motores de paso correspondientes a los ejes X, Y y Z. Tres señales más que proporcionan el sentido del movimiento de los ejes, es decir: izquierda o derecha para los ejes X y Y, y arriba o abajo para el eje Z. Dos señales más que indican al programa si los motores de los ejes X y Y se encuentran en marcha o paro, para regular la cantidad de corriente, todas estas 8 señales se las obtienen del puerto P2A.

Una vez especificado esto se procede a programar la palabra de control de los 2 PPI.

D7	D6	D5	D4	D3	D2	D1	D0

PPI #1

D7 = Mode set flag activo = 1

D6 D5 = Modo A = 00

D4 = Puerto A 0=>salida, 1=>entrada = 1

D3 = parte alta de C 0=>salida, 1=>entrada = d'ont care

D2 = Modo B = 0

D1 = Puerto B 0=>salida, 1=>entrada = 0

D0 = parte baja de C 0=>salida, 1=>entrada = d'ont care

1	0	0	1	0	0	0	0

En hexadecimal la palabra de control para el PPI # 1 será 90

PPI #2

D7 = Mode set flag activo = 1

D6 D5 = Modo A = 00

D4 = Puerto A 0=>salida, 1=>entrada = 0

D3 = parte alta de C 0=>salida, 1=>entrada = d'ont care

D2 = Modo B = 0

D1 = Puerto B 0=>salida, 1=>entrada = 1

D0 = parte baja de C 0=>salida, 1=>entrada = d'ont care

1	0	0	0	0	0	1	0

En hexadecimal la palabra de control para el PPI # 2 será 82

CONTADOR PROGRAMABLE 8253

El 8253 es un dispositivo de temporización y conteo, fácilmente programable mediante un sistema de software. El 8253 resuelve uno de los problemas más comunes en sistemas de microcomputadores, la generación de tiempos de retardos en software de control.

Contiene 3 contadores de 16 bits, cada uno con capacidad para manejar entrada de reloj de hasta 10 Mhz.

Otras funciones del temporizador/contador, además de los retardos, que pueden ser implementados son:

- *Ciclos de reloj*
- *Contador de eventos*
- *Disparador digital*
- *Generador programable*
- *Generador de ondas Cuadradas*
- *Multiplicador binario*
- *Generador de formas de ondas complejas*
- *Controlador de motores complejos*

BUFFER DE LA BARRA DE DATOS

La barra bidireccional de 3 estados con buffer de 8 bit, sirve como interfase del 8253

El buffer de la barra de datos tiene 3 funciones básicas.

1) Programar los modos del 8253

2) Cargar los registro de conteo

3) Leer los valores de conteo

ESCRITURA/LECTURA LÓGICA

La escritura/lectura lógica acepta entradas desde la barra del sistema y esta genera señales de control para dispositivos. La escritura/lectura puede ser habilitada o deshabilitada por el CS (chip selector), si es que el dispositivo ha sido seleccionada por el sistema lógico.

LECTURA (RD)

Un bajo en esta entrada informa al 8253 que el CPU le esta enviando datos, los cuales pueden ser especificando la información del modo de trabajo, o , de un valor para ser cargado en el contador.

A0 , A1

Estas entradas son normalmente conectadas a la barra de dirección.

Su función es para seleccionar a uno de los tres contadores para ser operados y para seleccionar la dirección del registro de la palabra de control, lo cual selecciona el modo de trabajo.

CHIP SELECT (CS)

Un bajo en esta entrada habilita al 8253, no lee ni escribe a menos que el dispositivo sea seleccionado con un nivel bajo.

REGISTRO DE LA PALABRA DE CONTROL

El registro de la palabra de control es seleccionado por A0 y A1 cuando ambos son 1. Este registro acepta y guarda la información proveniente desde el buffer de la barra de datos. La información guardada en este registro controla el modo de operación de cada contador, selecciona el conteo binario o BCD y la forma de cargar un valor en cada registro de conteo.

En la palabra de control se puede solamente escribir, mas no leer información de ella.

CONTADOR #0, CONTADOR #1, CONTADOR #2

Estos tres contadores tienen idénticas operaciones por tal motivo, será descrito solo uno. Cada contador consiste de un contador decreciente de 16 bit pre-seteable. El contador puede operar en forma binaria o BCD, y la salida es configurada por la selección de los MODOS

guardada en el registro de la palabra control. Los contadores son totalmente independientes y cada uno puede trabajar en diferentes MODOS.

SISTEMA DE INTERFASE DEL 8253

El 8253 es un componente de los microcomputadoras INTEL y se interfasa de la misma manera como los otros periféricos de la familia.

Las entradas de selección A0 y A1 se conectan a las señales de la barra de dirección del CPU. La señal CS puede ser obtenida directamente de la barra de direcciones usando el método de la selección lineal. O puede ser conectada a la salida de un decodificador.

La operación de cada contador es determinada cuando es programado. Cada contador debe ser programado antes de ser usado. Los contadores no usado no necesitan ser programados.

FORMATO DE LA PALABRA DE CONTROL

A continuación se muestra el formato de la palabra de control

D7	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*
SC1	*SC0*	*RL1*	*RL0*	*M2*	*M1*	*M0*	*BCD*

Selección del contador a usar

SC1	*SC0*	
0	*0*	*Seleccionar contador 0*
0	*1*	*Seleccionar contador 1*
1	*0*	*Seleccionar contador 2*
1	*1*	*Ilegal*

Tabla 1.3 *Contadores a usar*

Tabla 1.4 *RL- Read/Load*

RL1	RL0	
0	0	Operación de conteo
1	0	R/L solo el MSB
0	1	R/L solo el LSB
1	1	R/L LSB primero, luego el MSB

Tabla 1.5 *Modo de operación de los contadores*

M2	M1	M0	
0	0	0	Modo 0
0	0	1	Modo 1
X	1	0	Modo 2
X	1	1	Modo 3
1	0	0	Modo 4
1	0	1	Modo 5

Tabla 1.6 *Forma de conteo*

BCD	
0	Conteo binario de 16 bits
1	Conteo BCD (4 décadas)

PROGRAMA DE LA PALABRA DE CONTROL DE LOS CONTADORES

Antes de encontrar la palabra de control de los contadores es necesario explicar la función básica de estos.

Cada circuito integrado, es decir cada 8253 poseen tres contadores programables independientes los que sirven para cargar un número determinado, que representan el número de pulsos que recibirá el motor de paso. El primer contador guardará el número total de pulsos que recibirá el motor de paso, mientras que un segundo contador guardará el mismo número disminuido en un porcentaje. Los contadores comenzarán a decrementarse hasta llegar a cero, el segundo contador llegará primero, el cual enviará una señal a un circuito electrónico, que es el encargado de dar los pulsos de reloj que llegan a los mismos contadores, para disminuir la

frecuencia de los pulsos y de esta manera ir disminuyendo la velocidad del motor de paso hasta que llegue a cero el primer contador.

Para el otro eje sucederá lo mismo. es decir que necesitaremos los 2 circuitos integrados 8253 para controlar la distancia recorrida y el frenado de los ejes X y Y.

Para el eje Z tan solo se necesita un contador que almacene el número total de pulsos. Aquí no se utilizará el frenado ya que como la distancia del eje Z es pequeña no se requiere una velocidad muy alta y por lo tanto al terminar el conteo no habrá movimiento del eje por la inercia.

Una vez especificado esto se procede a programar a los 8253

SC1	*SC0*	*RL1*	*RL0*	*M2*	*M1*	*M0*	*BCD*
D7	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*

CONTADOR #1 , #2

CONT A

0	*0*	*1*	*1*	*0*	*0*	*0*	*0*

En hexadecimal la palabra de control para el contador A será 30

CONT B

0	*1*	*1*	*1*	*0*	*0*	*0*	*0*

En hexadecimal la palabra de control para el contador B será 70

CONT C

1	*0*	*1*	*1*	*0*	*0*	*0*	*0*

En hexadecimal la palabra de control para el contador C será B0

CAPITULO 2

CARACTERÍSTICAS GENERALES DE UN MOTOR DE PASO

Un motor de paso es un transductor electromecánico que convierte señales electrónicas de tipo digital, es decir ondas cuadradas, en una rotación del árbol del motor. Cada impulso produce una rotación angular del árbol del motor el cual es llamado comúnmente PASO o STEEP.

El PASO es variable de motor a motor y depende de las características electromecánicas de su fabricación; actualmente los motores de paso son realizados con valores típicos de paso de 1.8 , 7.5 y 15 grados. Ya que la rotación angular es proporcional al número de pulsos aplicados, un motor con 1.8 grados completa un giro de 360 grados con 200 pulsos de mando, mientras que un motor de paso de 7.5 grados cumple un giro completo con 48 pulsos de mando.

PRINCIPIO DE FUNCIONAMIENTO

El principio de funcionamiento de un motor de paso se basa sobre la interacción entre el campo magnético fijo del rotor y la fuerza electromotriz variable generada por la corriente circulante en los devanados

Tal corriente tiene un valor fijo, la fuerza electromotriz generada por los polos del estator es constante existiendo entonces un punto de equilibrio estable. En esta situación el rotor se mueve hacia el punto más cercano de equilibrio y luego se detiene en posición fija como se muestra en la Gráfico.2.1

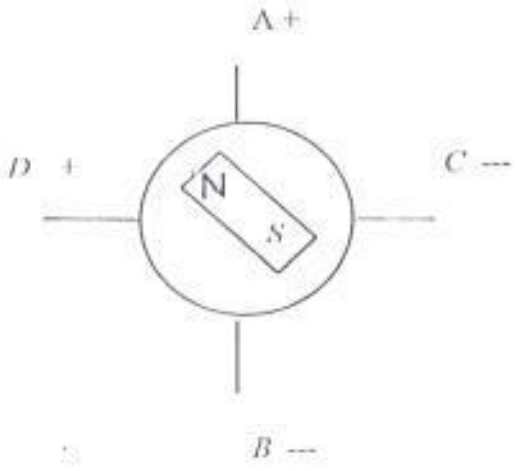


Gráfico 2.1 Posición fija del Rotor

Cuando los devanados son activados siguiendo una oportuna secuencia, la fuerza magnetomotriz generada por los polos del estator variará su polaridad y entonces el punto de equilibrio cambiará su posición siguiendo la misma secuencia. El rotor entonces es obligado a moverse, siguiendo el cambio de la secuencia del punto de equilibrio.

Se aplica una tensión de oportuna polaridad solo a los devanados que deben ser activadas para generar la necesaria fuerza magnetomotriz.

En la Gráfico mostrada a continuación están esquemáticamente representadas algunas secuencias de activación simulando que a cada distinta activación el rotor copia por simplicidad la rotación de 90 grados.

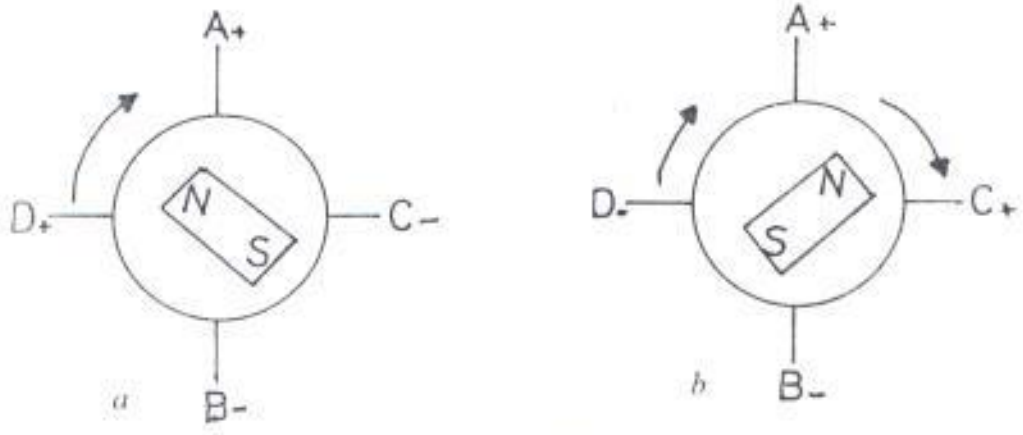
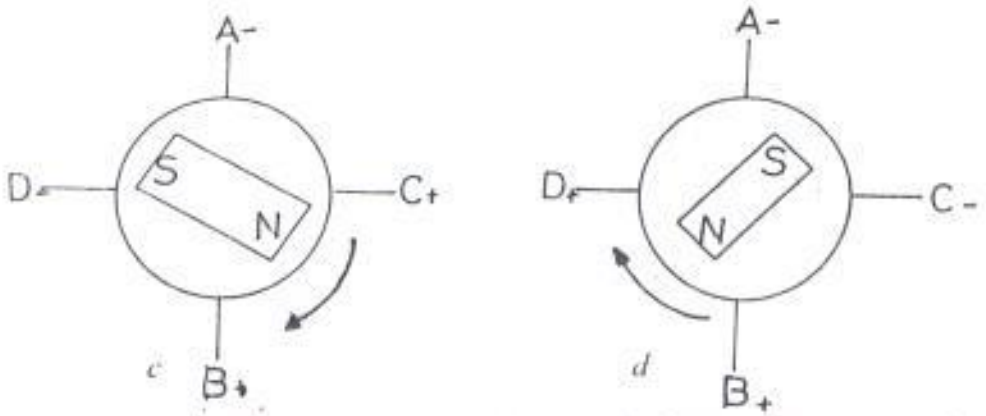


Gráfico2.2 Rotación de 90 grados del rotor en sentido horario

A) Aplicando tensión a los devanados A y D circulan por ellos la corriente, en los correspondientes estatores se genera por consiguiente fuerza magnetomotriz de polaridad n que mantienen en equilibrio constante la expansión polar sur del magneto permanente del rotor

B) Aplicando ahora tensión a los devanado A y C circula en ellas corriente, por lo tanto las fem de polaridad n se generan en los correspondientes estatores y la expansión polar sur del rotor es obligada a seguir la nueva posición de equilibrio realizando una rotación de 90 grados en sentido horario.



C) Si se aplica ahora tensión a los devanados B y C la fuerza magnetomotriz de polaridad N se generan en dos estatores B y C y el rotor esta obligado a efectuar una nueva rotación de 90 grados siempre en sentido horario.

D) En modo análogo se generan en su sección siempre puntos de equilibrio que producen la rotación continua del rotor.

Para invertir el sentido de rotación es suficiente invertir la secuencia de excitación de las devanados

TIPOS DE MOTORES DE PASO

Del punto de vista tecnológico existen dos tipos fundamentales de motores de paso:

- *Motor de paso a imán permanente*

- *Motor de paso a riluctancia variable*

Para ambos tipos de motores, el estator es realizado con un cierto número de espiras.

La diferencia entre ambos motores radica en las características del rotor, que es de cualquier modo siempre realizado sin devanados.

En los motores de paso a imán permanente el rotor es construido de un material permanentemente magnetizado, mientras que el motor a riluctancia variable el rotor es construido de un material ferromagnético que se magnetiza solo con el campo magnético generado por el estator.

Para obtener motores de paso con ángulos de paso bastante pequeños (7.5, 3.75, 1.8 grados) se debe adoptar técnicas constructivas mecánicas decisivamente complejas.

A continuación en la tabla 2.1 se muestra comparativamente las características principales de ambos motores de paso.

CARACTERÍSTICAS	*MOTOR DE PASO A IMÁN PERMANENTE*	*MOTOR DE PASO A RILUCTANCIA VARIABLE*
EFICIENCIA	*ALTA*	*BAJA*
VELOCIDAD ANGULAR DE ROTACIÓN	*BAJA*	*ALTA*
PRECESIÓN DE POSICIÓN	*MUY ELEVADA*	*BAJA*
INERCIA DE ROTOR	*ALTA*	*BAJA*
VALOR TÍPICO DE ÁNGULO DE PASO	*1.8, 15, 30 grados*	*7.5, 15, 30 grados*

tabla 2.1 Características de los tipos de motores de paso.

Motores de paso a imán permanente son mas eficientes, y se encuentran disponibles en versiones de alta potencia con una elevada precisión de posicionamiento.

Motor de paso a reluctancia variable es por el contrario más simple su construcción, tienen una inercia rotórica muy baja, una elevada velocidad de trabajo pero una limitada precisión de posicionamiento.

CONFIGURACIÓN DE LOS MOTORES DE PASO

De lo dicho anteriormente queda demostrado que la rotación del rotor se obtiene mandando correctamente y alternativamente la polaridad del campo magnético de la expansión polar del estator.

Para aplicar entonces la secuencia de mando con la correcta sucesión de pasos y la correcta polaridad es importante conocer la construcción interna de los devanados de un motor de pasos.

En la práctica existen dos diversas configuraciones de motores de paso:

- Motor de paso bipolar
- Motor de paso unipolar

Para cada una de las dos configuraciones de motores, se debe realizar diferentes técnicas de mando.

MOTOR DE PASO BIPOLAR

Estos motores son realizados con dos solos devanados distintos, como se representa en la Gráfico 2.4

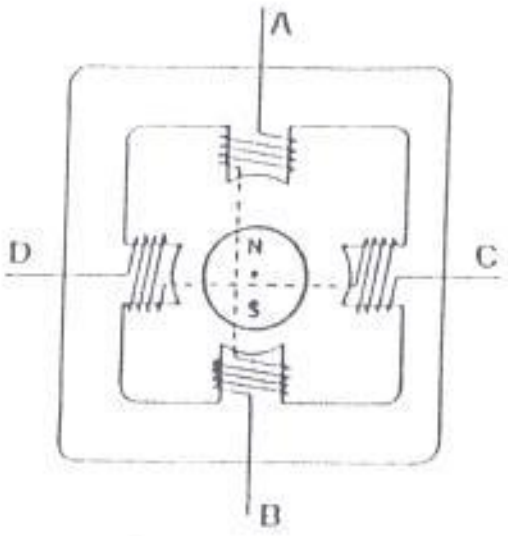


Gráfico 2.4 Motor de paso bipolar

Con este tipo de motores, para obtener la inversión de la polaridad del campo magnético es necesario invertir el sentido de la circulación de la corriente de los dos devanados del estator. Esto en la practica, se realiza invirtiendo la polaridad de la tensión aplicada al devanado del estator.

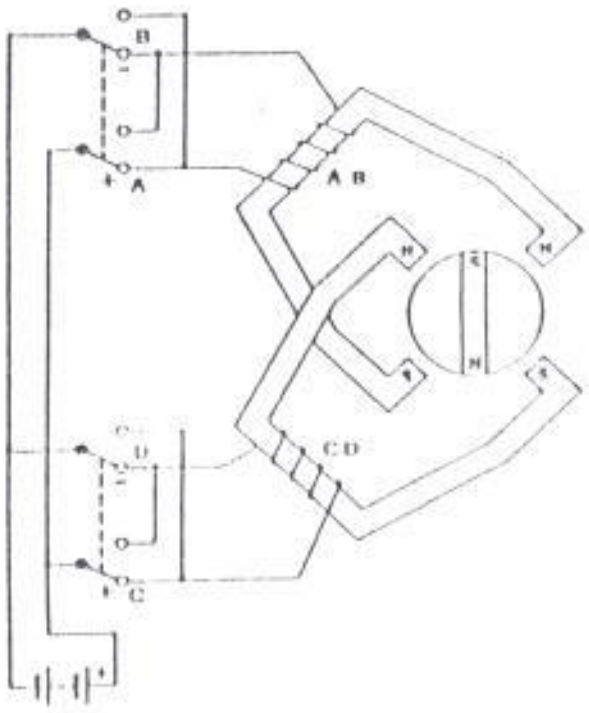


Grafico 2.4 Inversion de la polaridad del campo.

A continuación se muestra en la Grafico 2.5 un esquema simplificado del principio del motor de paso bipolar.

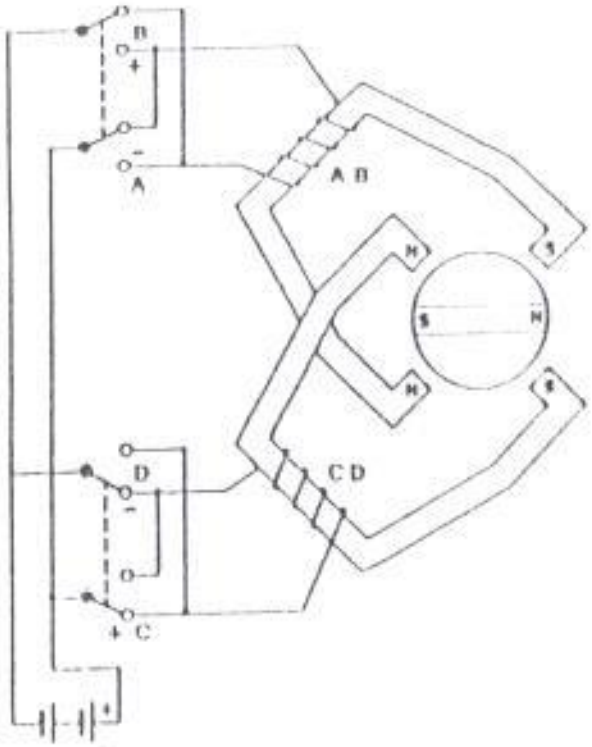


Grafico 2.5 Principio del motor de paso bipolar.

Este tipo de mando requiere entonces una realización muy compleja del circuito electrónico, el cual es comúnmente llamado MANDO BIPOLAR a causa de la doble polaridad de la tensión que se debe aplicar a los devanados.

A continuación se representa esquemáticamente una secuencia completa de mando para un motor de paso bipolar con ambos devanados activados. Gráfico 2.6

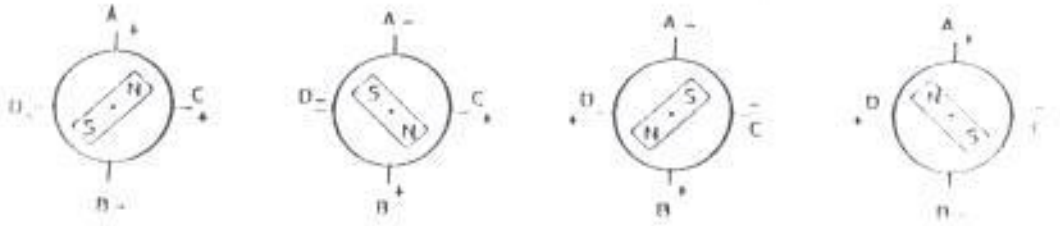


Gráfico 2.6 Secuencia de mando completa de un motor bipolar.

En la tabla 2.2 se puede representar eficazmente la procedencia completa de la secuencia de mando, que permite la rotación uniforme y continua del rotor.

	A	B	C	D
ESTADO INICIAL	+	+	-	-
1er IMPULSO	-	+	+	-
2do IMPULSO	-	-	+	+
3er IMPULSO	+	-	-	+
4to IMPULSO	+	+	-	-

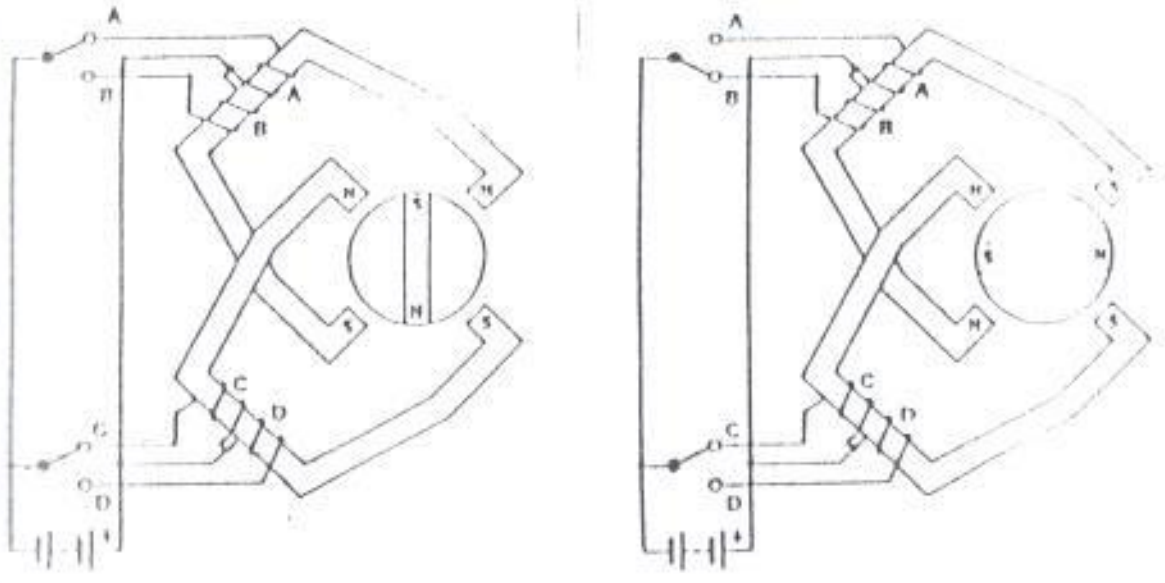
Tabla 2.2 Secuencia de mando de un motor de pasos bipolar.

MOTOR DE PASO UNIPOLAR

Estos motores de paso son realizados con devanados de estator independientes o con devanados de estator de una pieza central, de modo de tener dos medios devanados opuestos uno del otro.

Con la segunda ejecución se genera un campo magnético opuesto sin haber invertido la polaridad de la tensión de alimentación.

En la Gráfica 2.7 se muestra un esquema simplificado del principio del motor paso a paso unipolar, del circuito de polarización de los devanados.



Gráfica 2.7 Principio del motor de paso unipolar.

Estos tipos de motor requieren la realización de circuitos electrónicos extremadamente simples y comúnmente llamados MANDO UNIPOLAR, ya que la tensión aplicada a los devanados mantiene constante su polaridad.

El siguiente esquema de la Gráfico 2.8 representa una secuencia completa de un mando para un motor de paso unipolar.

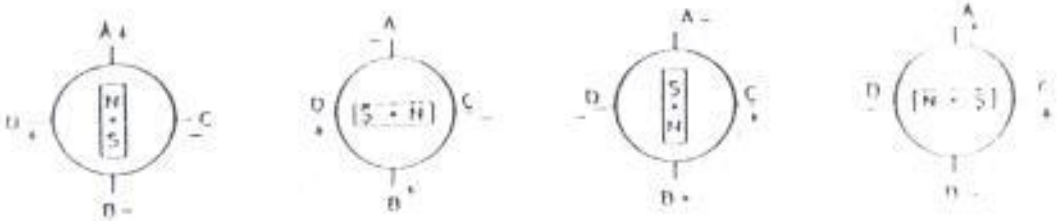


Fig. 2.8 Secuencia de mando completa de un motor unipolar.

En la tabla 2.3 se puede eficazmente representar la secuencia completa de mando.

	A	B	C	D
ESTADO INICIAL	1	--	--	1
1er. IMPULSO	--	1	--	1
2do. IMPULSO	--	1	1	--
3er. IMPULSO	1	--	1	--
4to. IMPULSO	1	--	--	1

TABLA 2.3 Secuencia de mando de un motor de paso unipolar.

OPERACIÓN PASO COMPLETO

Este es un modo típico de operación. Simultáneamente la alimentación de pasos es para ambos bobinados permitiendo así el máximo momento de torsión.

Las señales de entrada de Fase determinan la dirección del flujo de corriente en los devanados, mientras que las señales de entrada I0 e I1 indican de la corriente. Este es el más simple método de control para ser implementado.

OPERACIÓN PASO MEDIO

Este modo permite la resolución doble del motor y además elimina las vibraciones producidas. Este fuerza es aplicada alternativamente para un bobinado y después para ambos. En posición medio paso, solo un bobinado es energizado, teniendo en este momento la más baja torsión sobre el motor perno eje. Las mismas señales de control que fueron usadas para la operación paso completo son aplicados para a las señales de entrada de "Fase".

VENTAJAS Y DESVENTAJAS DEL MEDIO PASO

Una ventaja esencial de un motor de paso operando en medio paso, es la posición de resolución incrementada por el factor 2.

Desde un motor de 3.6 grados se consigue 1.8 grados, lo cual significa 200 pasos por revolución.

Esto no es siempre, pero a menudo usted esta forzado a operar a medio paso para evitar que las operaciones sean disturbadas por las vibraciones del motor.

Esto puede provocar que el motor tenga menos momento de torsión en ciertos pasos y pierda completamente esta posición.

El hecho de que la operación a paso medio no genera una solución, se debe a que presenta ciertas desventajas:

El paso medio del sistema necesita 2 pulsos de reloj como mínimo para igualar al sistema de paso completo ; la frecuencia del reloj es 2 veces mas alta que para el paso entero.

En la posición del medio paso el motor tiene solamente cerca de la mitad de la torsión que para el paso completo.

Por esta razón muchos sistemas usan la operación del paso medio solamente si la frecuencia del reloj del motor no produce resonancia de vibración.

PUENTE MOSFET

Este es el circuito de potencia para dar movimiento al eje X o al eje Y, ya que ambos tienen igual funcionamiento. Como se aprecia en el circuito de la Gráfico 2.9, existen 4 MOSFET piloteados por medio de transistores, que reciben señales correspondientes a la secuencia a seguir, por medio de sus bases. Estas señales provienen del circuito de control del eje X y las denominaremos A B C D.

Cuando se cierran los mosfet 1 y4 se energiza una de las 2 bobinas del motor de paso, produciendo una secuencia A+.

Cuando se cierran los mosfet 2 y 3 se energiza la misma bobina, pero la corriente esta vez es en sentido contrario, a ésta secuencia se la llamará A-

Un segundo puente de Mosfet de idénticas características energizará la segunda bobina del motor de paso con denominación B+ o B- según sea el caso.

De esta forma ambos puentes se encargan de energizar ambas bobinas del motor de paso.

Por último la resistencia al final del puente, sirve para sensar la corriente que pasa a través del bobinado del motor de paso.

Esta corriente que circula a través de la resistencia produce un voltaje que es enviado a un circuito de control que necesita ésta información.

Los voltajes apropiados para cerrar los MOSFET son los indicados en la Gráfico 2.9, es decir:

Vi *equivalente a 12 v, cierra los Mosfet inferiores*

Vp. *equivalente a 35v, es el voltaje de funcionamiento del puente*

Vs *equivalente a 50 v, cierra los Mosfet superiores*

CIRCUITO DE CONTROL DEL MOVIMIENTO DEL EJE X o Y

Para describir mejor su funcionamiento lo analizaremos en dos partes por separado

***La primera parte** tiene que ver con el control del puente de Mosfet y*

***La segunda parte** del circuito tiene que ver con el control de la secuencia por medio del programa, para la ejecución del movimiento del eje.*

Como se aprecia en la Gráfico 2.10, existe un circuito integrado digital denominado L297 que es el que genera la secuencia deseada de pulsos, ya sea para delante o para atrás dependiendo de la señal AD/ATRÁS proveniente del computador. De igual manera este integrado es el encargado de trabajar a medio paso o paso entero.

Además de las salidas ABCD posee 2 salidas más denominadas INH1 e INH2, las cuales pasan a través de una lógica de puertas AND para obtener las señales ABCD y A'B'C'D' que son las que sirven para pilotear a los Mosfet de potencia de ambos puentes descritos anteriormente.

Las señales provenientes de las resistencias sensoras de ambos puentes llamadas SENS1 y SENS2 entran también al L297 los cuales sirven para sensar el voltaje de las fases activas A y B y también de C y D.

A su vez estas señales son comparadas con un voltaje de referencia por medio de los 2 operacionales 741.

Si alguno de estos voltajes es mayor que el de referencia se obtendrá un nivel bajo en la entrada de la puerta AND con lo que la salida de la misma puerta también será bajo y mantendrá el transistor en corte lo que producirá que se interrumpa el paso de corriente por el Relé que es el encargado de energizar todo este circuito.

Es decir êste circuito sirve de protección contra corrientes demasiada altas.

Para volver a energizar el circuito es necesario nuevamente pulsar la botonera.

En el Pin 15 del L297 ingresamos el voltaje de referencia del circuito del chopper, este valor es diferente dependiendo de la secuencia en que se está, esto se lo realiza por medio de una lógica que tiene como señales de ingreso las señales INH1 e INH2.

Existe también un 555 en conGráficoción astable que sirve para producir la frecuencia de chopper.

Esta frecuencia la hacemos variar dependiendo si estamos en marcha o paro de los motores, por medio de interruptores analógicos CD 4016 piloteados por la señal, de marcha o paro ,M/P proveniente del computador.

La señal de reloj del astable ingresa a un monoastable 74LS221 para luego ingresar definitivamente al controlador L297 con lo que quedaría descrito el funcionamiento de ésta parte del circuito de control.

La segunda parte de este circuito toma y recibe señales de la tarjeta de interface montada en uno de los SLOT de computador y dibujada en la Gráfico 2.11.

Analizaremos el funcionamiento del circuito tomando en cuenta los pasos que sigue el programa..

1.-) Sale por el puerto P1B el valor de la velocidad del movimiento con valores binario de nomás 1111.

Estos datos pasan a través de un buffer 74244 para activar interruptores analógicos CD4016, obteniéndose de esta forma un convertidor de digital a analógico, con ayuda de un circuito con operacionales, resistencias y diodos.

2.-) Se pone en el puerto de salida el correspondiente dato que especifica la dirección del movimiento del eje, mediante la señal adelante/atrás (1 = adelante, 0 = atrás).

Este dato que sale de la tarjeta de interface además de llegar al controlador L297 pin No17 para declarar la dirección del movimiento del motor, es procesado mediante una lógica de puertas AND conjuntamente con los datos enviados de los sensores de fines de carrera del eje, de tal forma que si llegase el taladro al extremo del eje , esta lógica enviará una señal para bloquear los pulsos de reloj que producen el movimiento del eje en ese sentido, este bloqueo permanecerá hasta sacar el taladro de dicha posición o cambiar la dirección del movimiento del eje.

De tal manera podemos decir que el taladro posee protecciones de fines de carrera no solamente por medio del software, sino también de hardware.

3.-) Se carga los contadores con el número de pulsos necesarios para recorrer cierta distancia; cuando esto sucede, las señales de la tarjeta OUT 1 y OUT 2 se ponen en bajo.

Ambas señales indican el estado de los contadores ; con bajo significa que todavía no llega el conteo al valor predeterminado en el contador, mientras que en alto significa la finalización del conteo..

El contador 2 carga el valor total de los pulsos requeridos, mientras que el primer contador carga solo el 90 % de los pulsos requeridos, de tal forma que éste finalizará primero el conteo, y enviando esta señal a un circuito apropiado, comenzará a disminuir la velocidad de los pulsos para ir frenando el movimiento del eje hasta que finalice el conteo el segundo contador.

4-) Se sube el bit de Marcha /Paro el cual conmuta interruptores para cambiar la frecuencia de chopper descrita anteriormente.

5-) Se sube el bit de start, que conjuntamente con la señal OUT 1 ingresan a una puerta AND la cual conmuta un interruptor analógico CD4016 para comenzar la carga de un condensador el que irá aumentando poco a poco el voltaje con una rapidez que dependa del dato 'Velocidad' enviado por el puerto P1B y de un circuito con operacionales, transistores y zener que proporcionan una corriente constante, lo que produce una carga lineal del condensador.

Este valor de voltaje del condensador ingresa a un seguidor de voltaje realizado con un operacional, la salida del operacional es el valor requerido por un convertidor de voltaje a frecuencia (CI4046) para producir los pulsos de reloj que ingresan al controlador LM229 y a la tarjeta de interface, específicamente a los contadores, que serán los encargados de ir contando el número de pulsos de reloj, hasta que dicho número sea igual al predeterminado anteriormente.

Cuando el 90% de los pulsos requeridos halla ingresado, la señal OUT 1 conmutará el interruptor que produce la carga del condensador y éste empezará ahora a descargarse, lo que influirá disminuyendo la velocidad del 10 % de los pulsos restantes, teniendo como resultado la disminución de la velocidad del eje hasta su total paro.

DESCRIPCIÓN DEL EJE Z

Este circuito mostrado en la Gráfico 2.12, sirve para controlar el movimiento del eje Z hacia arriba o hacia abajo, para lo cual dispone de un contador UP/DOWN 74191 que cuenta ascendente o descendentemente dependiendo de la señal proveniente del computador

La salida de este contador ingresa a las entradas de control de un decodificador 74138, el que a su salida tiene una lógica compuesta de 3 puertas AND de 3 entradas cada una, las cuales sirven para llevar a corte o saturación a 3 transistores en conGráficoción Darlington que pilotean cada una de las bobinas del motor de Paso, produciendo la siguiente secuencia.

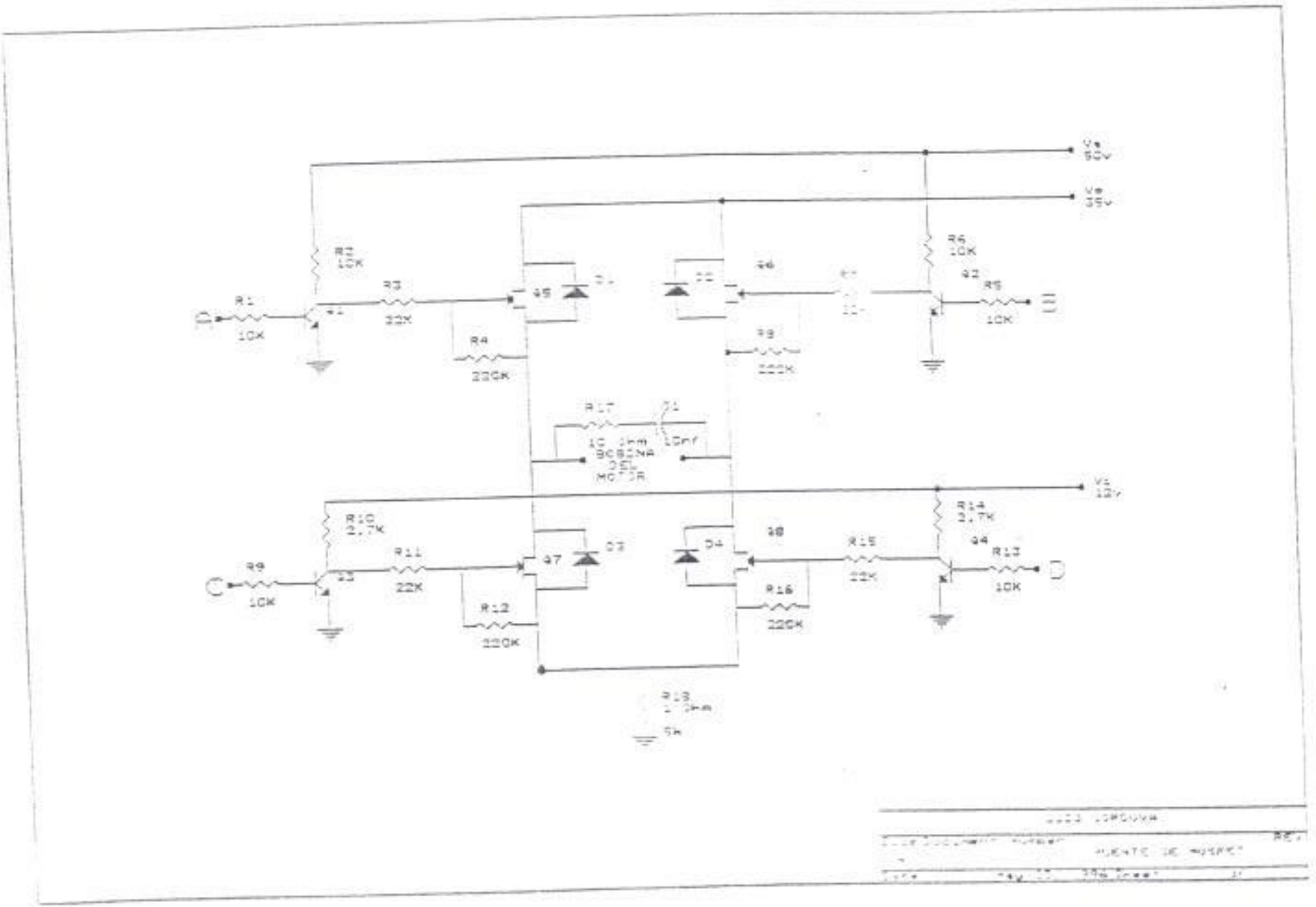
A - AB - B - BC - C - CA Ascendente

CA - C - BC -B - AB - A Descendente

La rapidez de la secuencia es dada por un oscilador 555 que sirve como señal de reloj para el contador. Este oscilador posee un potenciómetro para regular la velocidad de subida y otro para regular la velocidad de bajada, esto es posible gracias a un par de interruptores que automáticamente se conmutan dependiendo de la señal ARRIBA / ABAJO proveniente del computador.

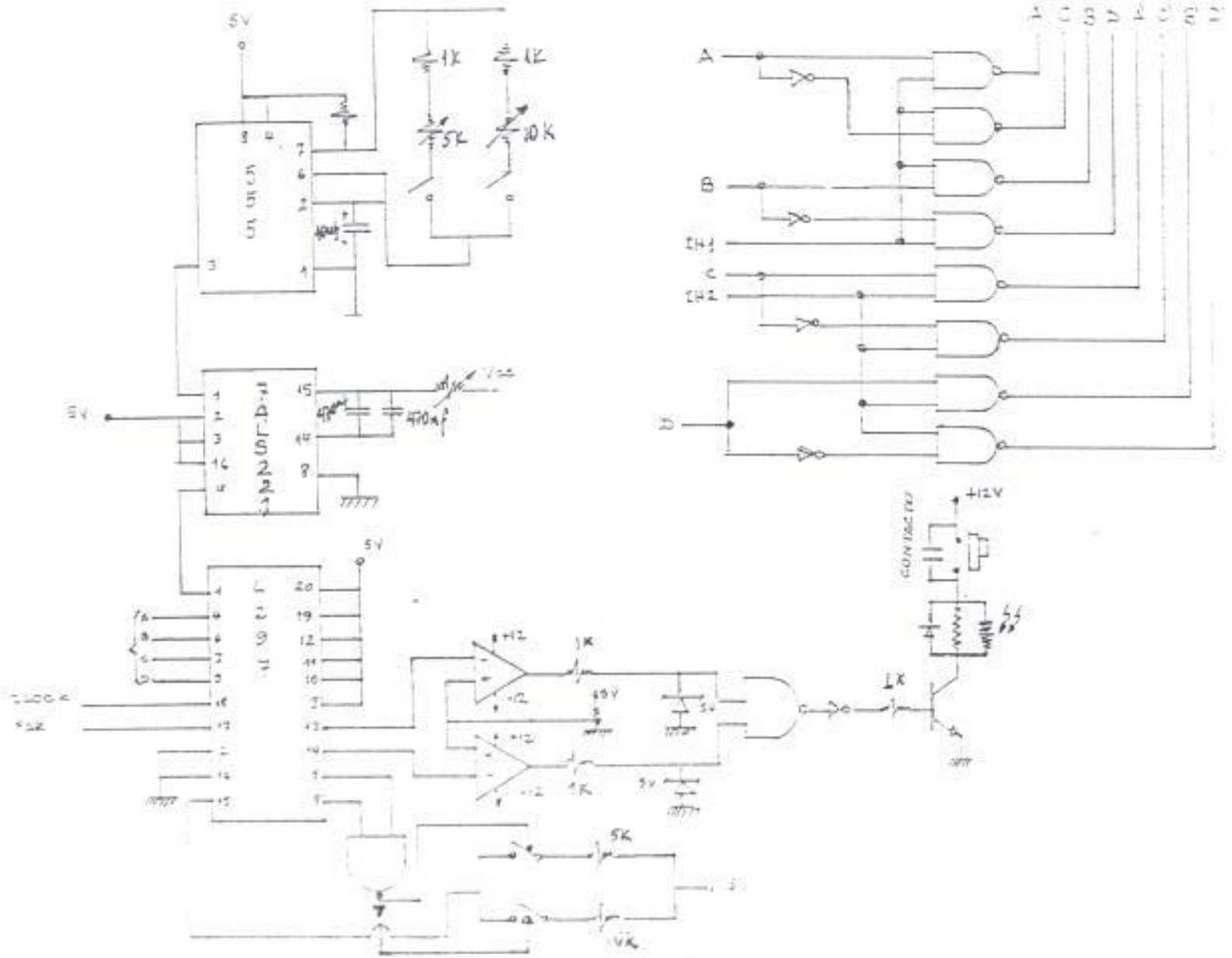
Existe otra lógica que recoge las señales de los sensores de fines de carrera inferior y superior y los procesa de tal manera que si llegara el taladro hacia un extremo de la carrera esta lógica mediante una señal baja deshabilitaría al 555 y por ende se detendría inmediatamente el motor y el movimiento de dicho eje Z, hasta que ingrese una señal que de movimiento contrario al eje, esta señal es ARRIBA / ABAJO.

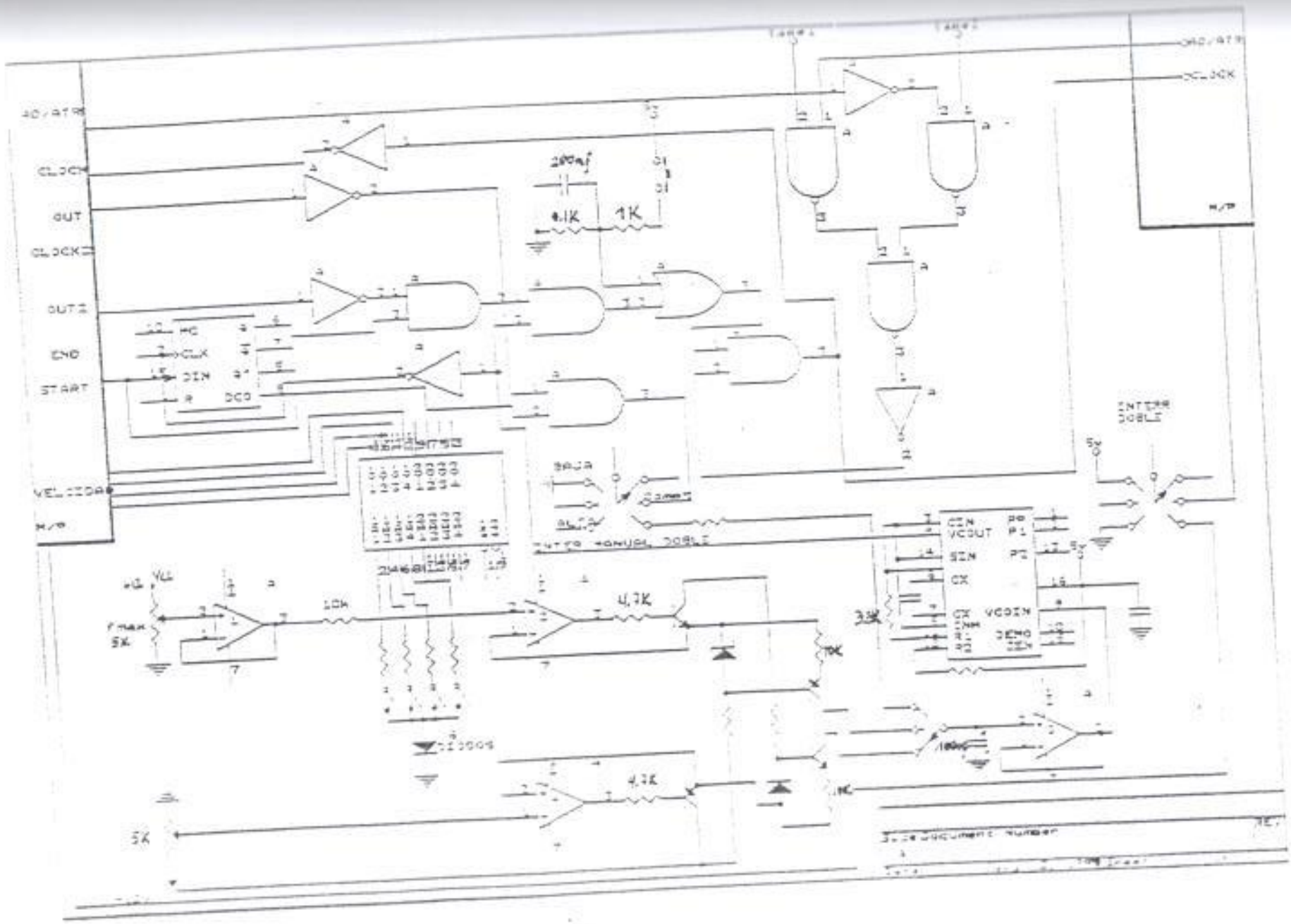
De igual forma esta lógica sensa las señales START y OUT que vienen del computador, con los que dan inicio y parada al movimiento de este eje Z.


R1
10K
R2
10K
R3
22K
R4
220K
R5
10K
R6
10K
R7
R8
220K
R9
10K
R10
2.7K
R11
22K
R12
220K
R13
R14
2.7K
R15
22K
R16
220K
R17
D1
D2
D3
D4
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
MOTOR

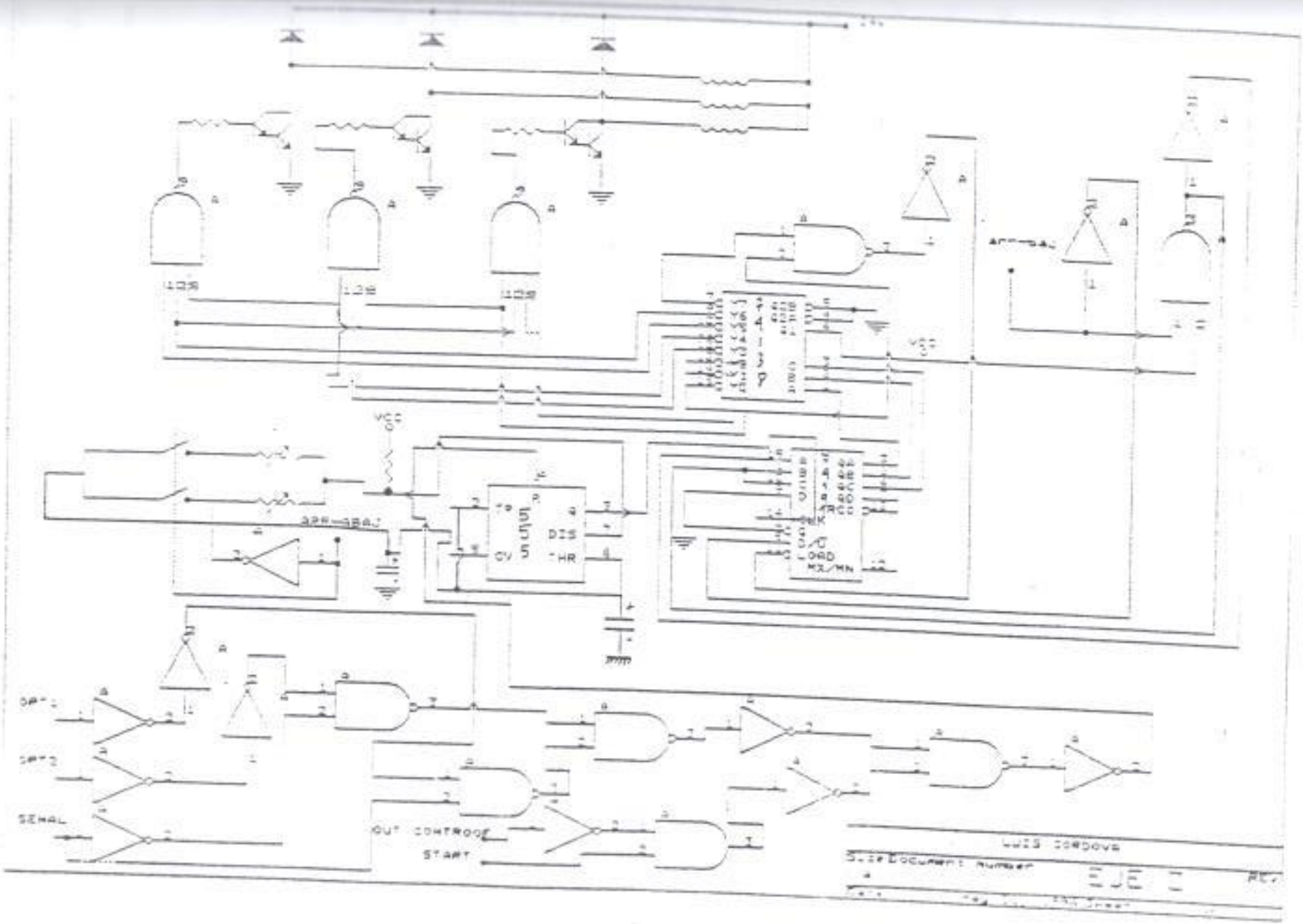
G. 2. 9.

CONTROL X




CLOCK
OUT
CLOCK
OUT2
END
START
VELOCITY
4.1K
1K
200pf
10K
4.7K
5K
VCOUT
SIN
CX
VCOIN
DEMO

G. 2. 11.



G. 2. 12

CAPÍTULO 3

DISPOSITIVOS SENSORES

INTRODUCCIÓN

En la actualidad, es muy frecuente encontrar todo tipo de sensores en las máquinas diseñadas para controlar algún proceso industrial. Estos dispositivos son los encargados de sensar los estados en que se encuentra la máquina y comunicarlo al controlador de la misma, para que este a su vez, continúe con el proceso, lo modifique o lo detenga según sea el caso.

SENSORES FINES DE CARRERA

El taladro electrónico se encuentra montado sobre una mesa de trabajo de 130 cm de longitud y 80 cm. de ancho, siendo ésta la dimensión total de la misma, mas no la dimensión real de trabajo la cual es mas pequeña, y es de 90 cm de largo por 60cm. de ancho.

Una vez especificado esto se entiende que la tarjeta más grande que pueda perforar el taladro sería igual a la dimensión real de trabajo de la mesa, por lo que si la tarjeta rebasa estos límites puede ocasionar problemas en los motores de paso ya que estos seguirían mandando pulsos para avanzar y la carrera del eje se habrá terminado, lo que le proporcionaría al motor un calentamiento de sus partes internas, como lo son sus devanados, y acto seguido causaría la pérdida del motor. . Es por esta razón que se emplea un par de topes de fines de carrera para cada uno de los ejes que se encuentran en movimiento es decir ejes, X, Y, Z.

Estas señales de respuesta de los sensores de fines de carrera están continuamente monitorizadas por un adecuado software de control, el cual se encuentra leyendo ésta información proveniente de los sensores cada vez que el manda a moverse alguno de los motores de paso de los diferentes ejes.

Si se llegara al fin de uno de los ejes el programa se encargara de suprimir la orden START , la cual, es la que habilita al circuito destinado a proporcionar los pulsos, de ésta manera se detendrá el proceso de movimiento de los motores de paso, y proporcionará un mensaje en la pantalla advirtiendo lo sucedido y especificando lo que se debe hacer.

De igual forma, la señal proveniente de los sensores es tratada por un circuito electrónico, que detecta esta anomalía y también se encarga de bloquear el movimiento de los motores. Es decir, los sensores proporcionan una doble protección en este caso, mediante el Hardware y el Software.

TIPO DE SENSOR ESCOGIDO

Existen una gran variedad de sensores de fines de carrera ya sean eléctricos, electrónicos o mecánicos, todos ellos capaces de funcionar adecuadamente, pero el del tipo electrónico es más rápida su respuesta, por tal motivo opté por elegir un sensor tipo óptico, llamado módulo interruptor opto acoplador de denominación ECG 3100 el cual posee un circuito emisor realizado en base a un diodo, y un circuito receptor realizado mediante un transistor el que trabajará en corte o saturación dependiendo del estado y de la posición de la máquina. Ver fig. 3.1.

FIG 3.1

Como el taladro posee tres ejes, X, Y y Z, hubo la necesidad de implementar 6 sensores opto acopladores, es decir 2 por cada eje.

Para tener un a mejor respuesta de los sensores, se elaboró un circuito capaz de recibir la información de los opto acopladores y enviarsela al circuito electrónico de control de los ejes X, Y y Z, y a la Interfase Paralela Programable PPI, la cual a su vez está conectada al computador, tal como se muestra en el diagrama de bloques de la fig.3.2

DIAGRAMA DE BLOQUES DEL CIRCUITO SENSOR

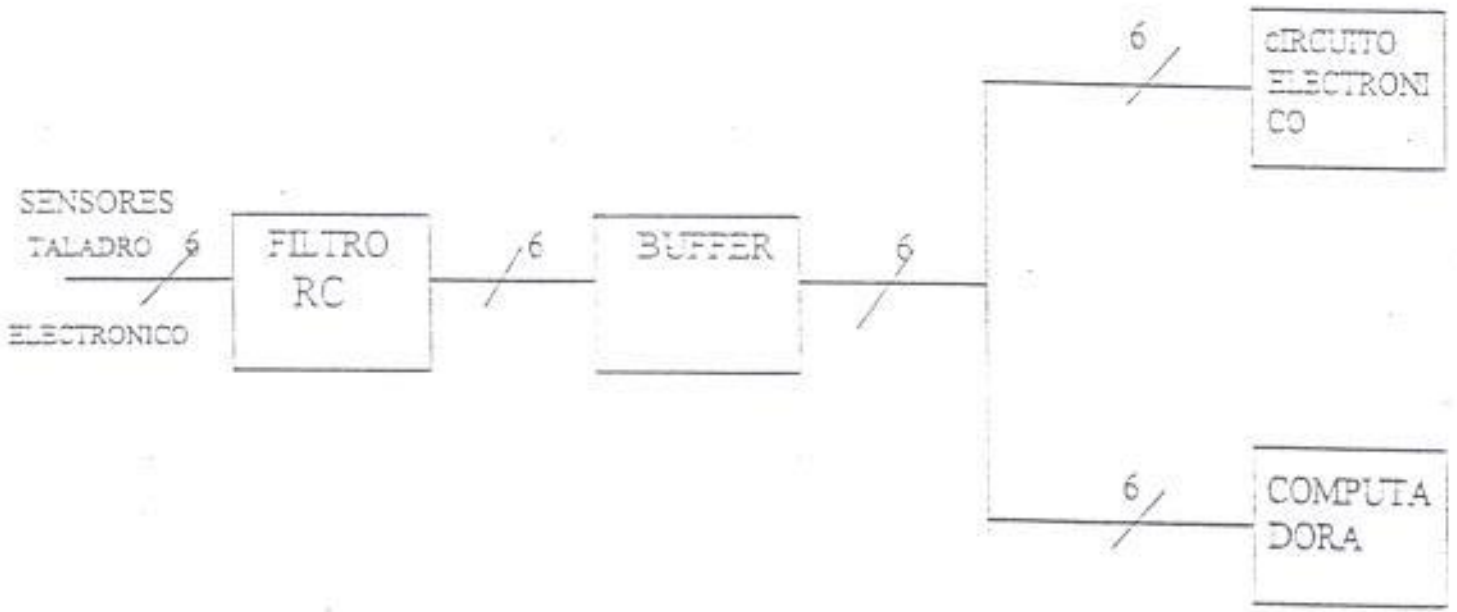


DIAGRAMA ESQUEMATICO DEL CIRCUITO SENSOR.

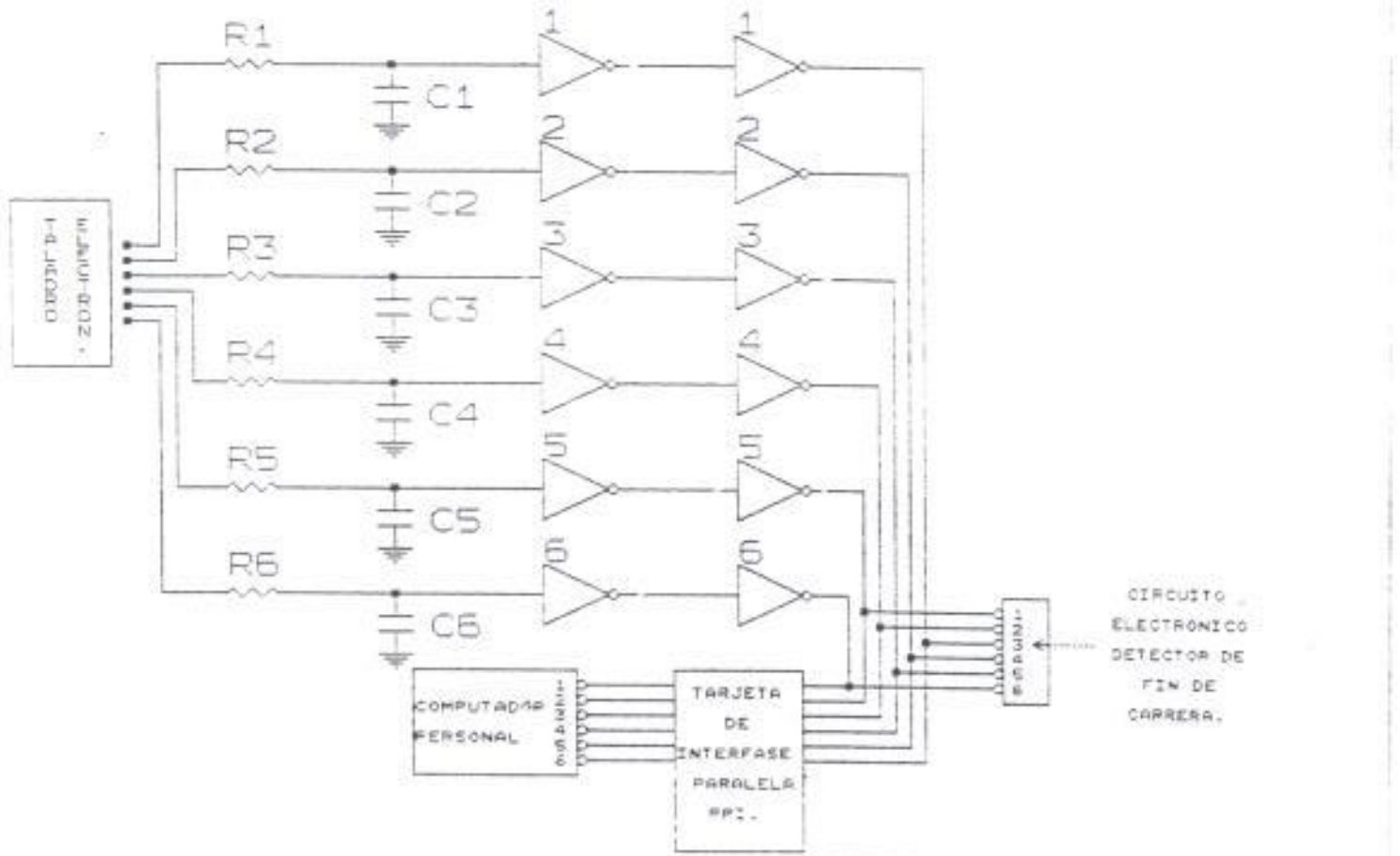


FIG 3.3

CAPÍTULO 4

CARACTERÍSTICAS DE LA FUENTE DE PODER

INTRODUCCIÓN

Como es conocido, toda máquina requiere diferentes tipos de voltajes ya sean estos alternos o continuos, y esta máquina no es la excepción, por tal motivo creí importante dedicar un capítulo entero a la construcción de la fuente de poder con sus respectivos voltajes.

En este capítulo estudiaremos las características mas importantes de la fuente de poder requerida para alimentar al taladro electrónico, esto es posible luego de un análisis de todos los circuitos perteneciente a la máquina ya sean estos digitales o analógicos para establecer los voltajes adecuados de funcionamiento.

En realidad la fuente de alimentación del taladro no es otra cosa que varias fuentes de diferentes voltajes que forman la gran fuente.

ESTUDIO DE LOS VOLTAJES REQUERIDOS

Para analizar los voltajes requeridos por el taladro es importante dividir los circuitos electrónicos en :

- Circuitos digitales.
- Circuitos analógicos.

CIRCUITOS DIGITALES

Todos los circuitos digitales encontrados en la máquina funcionan con tecnología TTL. los que requieren 5v. para ser alimentados. Los circuitos digitales se encuentran en los ejes X, Y y Z.

Tanto en el eje X como en el eje Y existen 14 integrados, mientras que en el eje Z existen 9 integrados, lo que nos da un total de 37 integrados alimentados por una fuente de 5 voltios.

Si asumimos que cada integrado consume un promedio de 20 mA, el consumo total de los circuitos digitales seria entonces de 740 mA.

Este consumo de corriente que no excede de 1 A. puede ser proporcionado normalmente por el circuito integrado LM7805 que maneja hasta 1 Amp. con un voltaje de 5v. positivos. A continuación se ilustra lo antes mencionado en la tabla 4.1.

Tipo de Eje	# DE C.I.	POT. POR C.I. (mA)	POTENCIA (mA)
EJE X	14	20	280
EJE Y	14	20	280
EJE Z	9	20	180
		TOTAL	740

Tabla 4.1 Consumo de Potencia de los C.I. digitales

La fuente completa para alimentar los circuitos digitales seria entonces la siguiente:

- Transformador reductor de voltaje a 8v. rms en el secundario, con una potencia de 12 vatios, lo que significa un consumo de hasta 1.5 Amp. en el secundario.

- Puente rectificador de diodos ECG166 con caracteristicas de 2 Amp. de corriente directa de rectificación y con un voltaje inverso de pico de 100 voltios.

- Filtro de voltaje a la entrada del circuito regulador, con un valor de 100 uf.

- Circuito regulador de tensión LM7805 de 5v. positivos y 1 Amp. máximo de corriente.

- Filtro de 47uf. para suavizar aún más el rizado y proporcionar un voltaje constante.

De esta manera queda completamente descrita la fuente digital. Ver Gráfico 4.1

CIRCUITOS ANALÓGICOS

A diferencia de los circuitos digitales, los analógicos pueden funcionar con diferentes valores de voltaje, inclusive con valores negativos, en lo que se refiere a su polarización.

De igual manera pueden manejar en sus entradas valores no discretos y con cualquier tipo de polaridad.

Una vez establecido esto podemos entrar de lleno al análisis de los circuitos analógicos en lo que se refiere a los voltajes a utilizar, para lo cual dividiremos el estudio en ejes X, Y, Z. tanto de los circuitos de control como para los de potencia de los motores.

Como existe similitud entre los ejes X y Y será suficiente analizar uno de ellos para comprender al otro. En este caso analizaremos el eje X.

Comenzaremos el estudio del circuito de control del eje X, el cual contiene circuitos seguidores de voltaje, amplificadores y comparadores de voltaje todos ellos construidos con circuitos integrados LM741 , polarizados con voltajes bipolares de **+12v.** y **-12v.** además de switch analógicos polarizados también con 12v. positivos, y relés con bobinas que se excitan a los 12v.

El circuito de control del eje Z no requiere de más dispositivos que los digitales para su funcionamiento.

En conclusión existen 12 integrados LM741, 3 relés de 12v. y 4 switch 4016 de la familia CMOS, en todos los circuitos de control de la máquina, los cuales no excederán el amperio de corriente y podrán ser suministrados sin mayor tramite por los circuitos integrados LM7812 de tensión positiva de 12v. y el LM7912 de tensión negativa de 12v.

La tabla 4.2 muestra el número de circuitos integrados analógicos usados con su respectiva potencia.

ELEMENTO	# DE COMPONENTES	POT.POR COMPONENTES (mA)	POTENCIA TOTAL (mA)
LM741	12	40	480
4016	4	40	160
RELE	3	40	120
		TOTAL	760

Tabla 4.2 Consumo de los Circuitos Analógicos de control o baja Potencia

La fuente completa para alimentar los circuitos analógicos de control seria entonces la siguiente:

- Transformador reductor de voltaje a 26v. rms en el secundario, con tab central y una potencia de 39 vatios, lo que significa un consumo de hasta 1.5 Amp. en el secundario.

- Puente rectificador de diodos ECG166 con características de 2 Amp. de corriente directa de rectificación y con un voltaje inverso de pico de 100 voltios.

- Doble filtro de voltaje a la entrada de los circuitos reguladores positivo y negativo, con un valor de 470uf.

- Circuito regulador de tensión LM7812 de 12v. positivos y 1 Amp. máximo de corriente.

- Circuito regulador de tensión LM7912 de 12v. negativos y 1 Amp. máximo de corriente.

- Filtro de 100uf. para suavizar aun mas el rizado y proporcionar un voltaje constante, uno para cada salida de los reguladores, tanto positivo como negativo.

De esta manera queda especificada la fuente analógica para los circuitos de control de la máquina. Ver Gráfico 4.1

CIRCUITO ANALÓGICO DE POTENCIA

De igual manera dividiremos el análisis de los circuitos en ejes X, Y y Z recordando que para los circuitos de potencia del eje X, también se cumple que poseen iguales características que para el eje Y, por lo que tan solo analizaremos uno de ellos.

El eje X posee un par de tarjetas para alimentar a los motores de paso, por estas tarjetas y a través de los transistores MOSFET que se encuentran en ella, circula la corriente necesaria para energizar los 2 devanados de los motores de paso y por ende lograr el movimiento del eje.

*Esta tarjeta posee un puente de MOSFET polarizado a **40v**, valor de voltaje que será suministrado a los motores de paso, por lo que se necesitara por lo menos un voltaje de 52v. en la compuerta de los MOSFET superiores, y 12v. en las compuertas de los MOSFET inferiores, para que puedan cerrarse y conducir. Pero para no aumentar luego las características del transformador al pretender obtener los 52v. referenciados a tierra, se puede recurrir a otra fuente de 12v. referenciada a los 40v. del puente de MOSFET para así completar los 52v.*

Los 12v. necesarios para cerrar los MOSFET inferiores, pueden ser tomados de la fuente analógica de control que ya hemos descrito anteriormente.

*Entre los datos de placa del motor de paso se especifica la corriente necesaria de trabajo, con un valor de 2Amp. por lo que, en la construcción del transformador la corriente del secundario esta calculada para **6 Amp**. Mientras que la corriente necesaria para la fuente de 12v. referenciada a los 40v. es de apenas **1 Amp**.*

Por otra parte el eje Z se desplaza gracias a un motor de paso de 3 bobinas independientes con características de placa de **24v.** y **3 Amp.** de corriente, por lo que se necesitara un secundario con las características antes mencionadas.

Recordemos que el taladro electrónico posee también un motor de corriente continua que es el encargado de perforar la tarjeta que se esta trabajando. Este motor DC esta compuesto de un devanado de estator alimentado a **8v**, con una corriente no mayor a **1 Amp**. y de un devanado de rotor alimentado a **50v**. Como la fuerza que realizara el motor DC no es demasiado grande, bastara con **2 Amp**. para alimentarse.

A continuación la tabla 4.3 muestra los valores de voltaje requeridos por los circuitos analógicos de potencia.

	Vp	Vs	Vi	Vz	ROTOR	ESTATOR
VOLTAJE DC (v)	40	52	12	24	50	8
CORRIENTE	6	1	1	3	2	1
VOLTAJE EN TRANSF. AC	35	13	13	22	40	10

Tabla 4.3 Consumo de los Circuitos Analógicos de alta Potencia

La fuente completa para alimentar los circuitos analógicos de potencia la podremos ver en el Gráfico 4.1 y sería entonces la siguiente:

- Transformador reductor de voltaje a 35v. rms en el secundario, y una potencia de 210 vatios, lo que significa un consumo de hasta 6 Amp. en el secundario.

- Puente rectificador de diodos ECG166 con características de 2 Amp. de corriente directa de rectificación y con un voltaje inverso de pico de 100 voltios.

- Filtro de 470uf. a la salida del puente rectificador para reducir el rizado en la salida y obtener un voltaje constante de 40v.

- Transformador reductor de voltaje a 13v. rms en el secundario, y una potencia de 13 vatios, lo que significa un consumo de 1 Amp. en el secundario.

- Puente rectificador de diodos ECG166 con caracteristicas de 2 Amp. de corriente directa de rectificación y con un voltaje inverso de pico de 100 voltios.

- Filtro de 100uf. a la salida del puente rectificador para reducir el rizado en el voltaje de salida.

- Circuito regulador de tensión LM7812 de 12v. positivos y 1 Amp. máximo de corriente el cual será referenciado a la fuente descrita anteriormente

- Filtro de 100 uf. para suavizar aún más el rizado y proporcionar un voltaje constante

- Transformador reductor de voltaje a 22v. rms en el secundario, y una potencia de 66 vatios, lo que significa un consumo de hasta 3 Amp. en el secundario.

- Puente rectificador de diodos ECG166 con caracteristicas de 2 Amp. de corriente directa de rectificación y con voltaje inverso de pico de 100 voltios.

- Filtro de 470 uf. a la salida del puente rectificador para reducir el rizado en la salida y obtener un voltaje constante de 24v.

- Transformador reductor de voltaje a 10v. rms en el secundario, y una potencia de 10 vatios, lo que significa un consumo de 1 Amp. en el secundario.

- Puente rectificador de diodos ECG166 con caracteristicas de 2 Amp. de corriente directa de rectificación y con un voltaje inverso de pico de 100 voltios.

- *Filtro de 100 uf. a la salida del puente rectificador para reducir el rizado de voltaje.*

- *Circuito regulador de tensión LM7808 de 8v. positivos y 1 Amp. máximo de corriente*

- *Filtro de 10 uf. para suavizar aún más el rizado y proporcionar un voltaje constante*

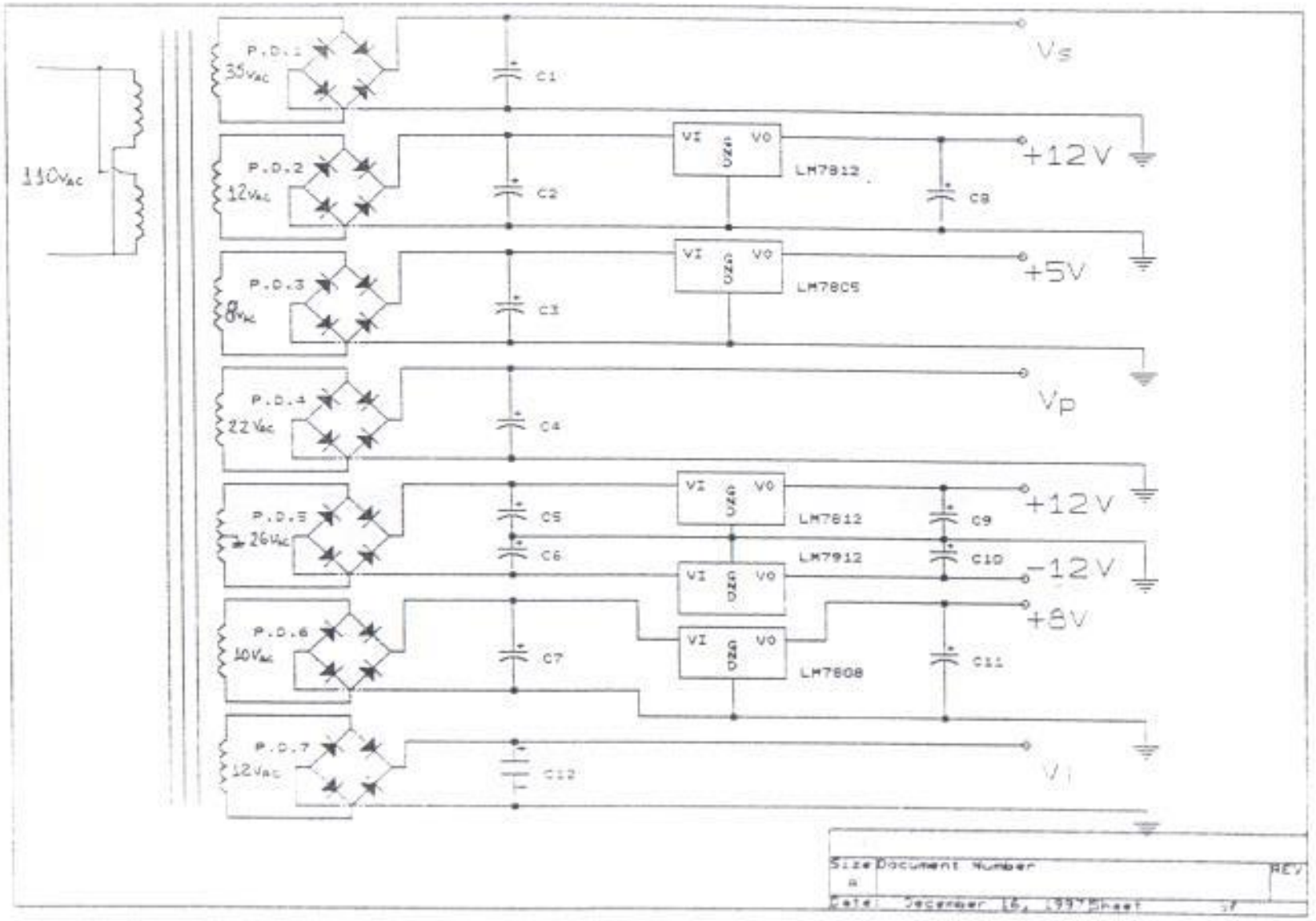
- *Transformador reductor de voltaje a 40v. rms en el secundario, y una potencia de 80 watios, lo que significa un consumo de hasta 2 Amp. en el secundario.*

- *Puente rectificador de diodos ECG166 con características de 2 Amp. de corriente directa de rectificación y con un voltaje inverso de pico de 100 voltios.*

- *Filtro de 470 uf. a la salida del puente rectificador para reducir el rizado en la salida y obtener un voltaje constante de 50v.*

De esta manera queda especificada la fuente analógica para los circuitos de control de la máquina, y por ende la fuente total del taladro.

A continuación, en el Gráfico 4.1 se muestra el diagrama esquemático del circuito de la fuente


110VAC
P.D.1
35VAC
C1
Vs
P.D.2
12VAC
C2
VI
VO
LM7812
C8
+12V
P.D.3
8VAC
C3
LM7805
+5V
P.D.4
22VAC
C4
Vp
P.D.5
26VAC
C5
C6
LM7812
C9
LM7912
C10
+12V
-12V
P.D.6
10VAC
C7
LM7808
C11
+8V
P.D.7
12VAC
C12
V1
Size
A
Document Number
REV
Date: December 16, 1997 Sheet of

CAPITULO 5

ESPECIFICACIONES DEL PROGRAMA

INTRODUCCION

En este capítulo se tratará exclusivamente de la estructura del programa, realizado en "Lenguaje C", que controla la máquina taladradora, para lo cual analizaremos paso a paso cada una de las funciones que se implementaron, con sus respectivos diagramas de flujo, lo que ayudará a comprender mejor la realización y funcionamiento del programa.

FUNCION PRINCIPAL

La función Principal del programa es la que controlará al taladro, y es la encargada de organizar todo el proceso a seguir.

Esta función comienza limpiando la pantalla del ordenador para acto seguido llamar a las funciones que inicializan el modo gráfico.

Luego se encerará la variable llamada opción la cual será usada más adelante. Posteriormente se llama a la función Hola, la que mostrará una pantalla de presentación del programa, que permanecerá por unos pocos segundos y luego chequeará el valor de la variable opción, la cual es una variable global, que podrá tomar cualquier valor del 1 al 4 y que será asignada dentro de la función 'Menú'. Si este valor es igual a 4 se presentará nuevamente la función 'Hola' que despedirá el programa llamando luego a la función 'Close graf' que cerrará el modo gráfico y terminará el programa. Por el contrario si el valor de 'opción' es diferente se quedará en un lazo hasta que sea igual a 4.

Si el valor de opción es igual a 3 se llamará a la función 'Perforación'. Si el valor de opción es igual a 2 se llamará a la función Abrir_Archivo. Si el valor de la variable opción es 1 se llamará a la función Crear_Archivo. Una vez que se llamó a cualquiera de las funciones mencionadas regresa a preguntar nuevamente el valor de la variable opción para volver a repetir el ciclo hasta que este sea igual a 4 que es cuando finaliza el programa. Ver Gráfico 5.1

FUNCION ABRE ARCHIVO

Esta función tiene como objetivo principal el de coger un archivo del dibujo del programa Tango, abrirlo y obtener de él la información necesaria sobre las coordenadas de cada uno de los Pad existentes en el dibujo. Dicho de otra manera, esta función sirve como filtros de información desechando otras entidades como lo son pistas y componentes, cogiendo tan solo los Pad con sus coordenadas, los que son guardados en un archivo intermedio con extensión TXT. Este archivo intermedio consta de coordenadas de X y Y con sus respectivos sentidos de movimiento, es decir hacia un lado o hacia el otro.

Además esta función se encarga de reordenar las coordenadas de los ejes para optimizar el trabajo y realizar las perforaciones de una manera ordenada.

En realidad la información de las coordenadas de cada Pad es la distancia relativa que existe entre un Pad y el siguiente.

A continuación se describe el flujo de la función Abre_Archivo. Declaramos las variables a utilizar, y le asignamos un valor de cero a todas ellas, luego se procede a abrir un archivo que será el archivo intermedio donde se guardará la información de los Pad.

Se pide el nombre y la ruta del archivo donde se encuentra el dibujo que se desea perforar, esto es un archivo del programa tango con extensión .PCB.

Si el nombre del archivo pedido no se encuentra en la ruta especificada saldrá un mensaje que el archivo no existe, de lo contrario abrirá este archivo para sacar información necesaria. El archivo intermedio tendrá el mismo nombre que el archivo con extensión .PCB del programa tango, y en el se guardará solo la distancia y la dirección que ambos ejes tomarán.

Una vez que se han tomado todos los datos necesarios procedentes del archivo con extensión .PCB se procede a hacer un arreglo de dos dimensiones para ordenar en forma ascendente las coordenadas absolutas del eje X.

El siguiente paso sería el de transformar las coordenadas absolutas de los Pad a coordenadas relativas, esto es que cada dato sea la distancia que hay que recorrer desde un punto al siguiente.

Luego de esto se procede a escribir en el archivo intermedio los datos en el siguiente orden: distancia en X (X), dirección en X (SX) distancia en Y (Y), dirección en Y (SY), finalizando con un punto y coma.

Como los datos de distancia en X han sido ordenados, la dirección (SX) siempre será la misma, la que llamaremos SX = 1

Mientras que si la resta de la coordenada en Y del siguiente punto con la coordenada en Y del punto anterior es positiva, la dirección SY será igual a 1, caso contrario SY será igual a 0.

Una vez que se hayan escrito todas las coordenadas en el archivo intermedio se procede a cerrar tanto el archivo intermedio como el archivo con extensión .PCB del programa tango con lo que finalizamos la función Abre_Archivo. Ver gráfico 5.2

FORMATO DEL ARCHIVO INTERMEDIO

Este fichero contendrá la información de los puntos donde debemos hacer los agujeros respetando el siguiente formato:

INICIO

X(distancia)SX(dirección 0 o 1)Y(distancia)SY(dirección 0 o 1);

FIN

El fichero comienza con la palabra inicio y termina con la palabra fin.

La distancia es referenciada al punto anterior. La dirección puede ser 0 para la izquierda en cualquiera de los dos ejes, o 1 para la derecha.

Al finalizar cada línea de dato se pondrá un punto y coma. A continuación presentamos un ejemplo de cuatro Pads que forman un cuadrado de 40 x 40.

INICIO

X40SX1Y0SY0;

X0SX0Y40SY1;

X40SX0Y0SY0;

X0SX0Y40SY0;

FIN

FUNCION CREAR_ ARCHIVO

Esta función tiene como objetivo crear un archivo en el que se guarden coordenadas de un determinado número de Pads o agujeros a taladrar.

Para este fin nos valemos de la ayuda de las teclas sube y baja, delante y atrás de la computadora, como se detalla a continuación.

En primer lugar creamos un archivo en el que se guardarán las coordenadas de los Pads, luego se espera a que se pulse alguna tecla, si ésta es diferente de alguna de las teclas funcionales se cerrará el archivo creado y se terminará la función Crear_Archivo.

De lo contrario, si la tecla pulsada corresponde a una funcional, se limpiará la pantalla y saldrá un mensaje que indica utilizar las flechas del teclado.

Si esta tecla es la flecha derecha llamara a la función MOVIMIENTO_XY, asignándole un valor de avance en el sentido positivo X, así mismo un contador de coordenadas en X se incrementará en un valor igual al que fue enviado a la función MOVIMIENTO_XY

Si la tecla funcional fuera la izquierda llamará a la función MOVIMIENTO_XY, asignándole un valor de retroceso en sentido negativo X. Así mismo un contador de coordenadas en X se decrementará en un valor igual al que fue enviado a la función MOVIMIENTO_XY.

Si la tecla funcional fuera hacia arriba llamará a la función MOVIMIENTO_XY, asignándole un valor de adelanto en el sentido positivo Y. Así mismo otro contador de coordenadas Y se incrementará en un valor igual al que fue enviado a la función MOVIMIENTO_XY.

Si la tecla funcional fuera hacia abajo llamará a la función MOVIMIENTO_XY, asignándole un valor de retroceso en sentido negativo Y. De igual forma el contador de coordenadas Y se decrementará en un valor igual al que fue enviado a la función MOVIMIENTO_XY.

Si la tecla funcional fuera F1, haría un cambio de el valor de la distancia que se mueve tanto el eje X como el eje Y, es decir cambiaría de un avance largo a un avance corto o viceversa, especificándose en la pantalla que tipo de avance es el que está presente.

Mientras que si la tecla funcional pulsada fuera F2 llamaría a la función Sube_Baja, que es la encargada del movimiento hacia arriba o hacia abajo del eje Z.

Si se pulsara F3 las variables X y Y tomarán los valores correspondientes de los contadores de coordenadas en X y Y para ser escritos en el archivo que se ha creado.

Si el valor de la variable X es positivo, la dirección SX=0 de lo contrario SX=1.De igual manera se procede para la dirección SY.

De esta forma se va escribiendo la información de cada Pad, es decir, distancia y dirección, con el mismo formato del archivo intermedio. Luego de esto se enceran los contadores tanto de las coordenadas en X como en Y y se espera nuevamente a que se pulse alguna tecla, y se vuelva a repetir el ciclo hasta que la tecla pulsada sea diferente a una funcional, con lo que finaliza la función Crear_Archivo. Ver el Gráfico 5.3

FUNCION HOLA

La función hola es una pantalla de presentación y despedida la que aparecerá al inicio y al final de la sesión de trabajo.

Esta pantalla de presentación y despedida nos indica el nombre de proyecto, es decir Taladro Efectrónico controlado por computadora. Además el nombre de la institución que auspició y colaboró para que este proyecto se haga realidad como es el colegio Salesiano "Domingo Savio". También muestra el nombre del autor del trabajo o sea Luis Córdova R. y la

Universidad encargada de impartir en mi, los conocimientos adquiridos, es decir la Escuela Superior Politécnica del Litoral.

Esta pantalla permanecerá con estas indicaciones por un lapso de 3seg. luego de lo cual se limpiará la pantalla dando fin a la función hola. Ver Gráfico 5.4

FUNCION INIPUERTO

Esta función tiene como principal objetivo el de escribir la palabra de control de cada uno de los dos puertos paralelos programables de interface PPI (8255) y de los contadores programables (8253). En otras palabras, aquí se programa a los PPI y a los contadores para asignarles el modo en que deberán trabajar. Ver Gráfico 5.5

FUNCION MARCHA - PARO

Esta función fue creada con la finalidad de reducir los niveles de corriente que manejará el motor de paso cuando éste se encuentre detenido, mientras que al estar en movimiento aumentara la cantidad de corriente.

Esto es fácil realizar ya que contamos con un Hardware apropiado, el cual solo necesita un bit para saber si se encuentra en marcha o en paro.

El flujo para esta función viene dado de la siguiente manera.

Asignación de valores para algunas variables, luego la salida del bit correspondiente por el puerto PORT2A que indicará la marcha con un 1, y el paro con un 0. Ver Gráfico 5.6

FUNCION MENSAJE

Esta función sirve para indicarnos algunos mensajes que el usuario o el operario del programa debe conocer para un mejor manejo del mismo. Estos mensajes son mostrados en cualquiera parte de la pantalla, ya sea en la parte superior, en el centro o en la parte inferior de la misma. Estas frases pueden ser de diferente color y tamaño, tal como se muestran en los Gráficos 5.7 , 5.8, 5.9, 5.10

Algunos de estos mensajes son los siguientes:

< F1 > Rápido- lento

< F2 > Subir- bajar

< F3 > Graba Pad

Movimiento manual de los ejes X y Y

Pulse cualquier tecla para ingresar al menú principal.

Utilice las flechas.

Movimiento manual del eje Z

FUNCION MENU

La función menú es la encargada de mostrar en pantalla al usuario 4 diferentes alternativas a las que puede acceder como son:

1.-) Graba perforaciones de otras tarjetas.

2.-) Captura archivo de programa tango.

3.-) Perforación de la tarjeta.

4.-) Salir del programa.

Una vez que el usuario escoge una de las alternativas presentada debe ingresar el número que corresponde a ella, el cual es guardado en la variable opción.

Luego se limpia la pantalla y retorna el valor de la variable opción a la función Principal acto seguido finaliza la función menú. Ver Gráfico 5.11

FUNCION MOVIMIENTO_XY

Esta función tiene como objetivo el de mover los ejes X y Y de acuerdo a la distancia que le sea enviada a la misma. Además se encarga de bajar el nivel de corriente a los motores cuando los ejes no estén desplazándose, para proteger al motor de sobrecalentamientos innecesarios cuando no estén en movimiento.

Esta función además regula la velocidad a la que van a ser movidos los ejes, en un rango de velocidad de 1 a 15.

A esta función le es mandada la distancia entre cada Pad, encontrada en el archivo intermedio, para que esta a su vez cargue con este valor un contador, el que se ira decrementando con cada pulso dado a los motores de paso, proporcionado por un circuito electrónico, hasta llegar a cero, interrumpiendo el movimiento de dichos motores.

Un segundo contador también es cargado con el valor de la distancia del Pad menos el 10 por ciento de ella. Este contador de igual manera con cada pulso ira decrementándose hasta llegar a cero. Como este valor es menor que el del otro contador finalizara primero dando una orden, a otro circuito electrónico, para que disminuya la velocidad de los pulsos y por ende del movimiento de los ejes, de una manera gradual. Esto sucederá para ambos ejes X y Y.

Una vez explicada la utilidad de la función MOVIMIENTO_XY, procedemos a mostrar el grama de flujo correspondiente a dicha función:

Como primer paso tenemos la definición de las variables a utilizarse en esta rutina, luego untamos sobre la velocidad requerida para el movimiento de los ejes.

Acto seguido cargamos los valores de desplazamiento de los ejes en los contadores tanto X como para Y, esto es cuatro contadores, dos para cada eje.

A continuación se indican la dirección que tomarán los ejes X y Y la cual puede ser hacia lante o hacia atrás para un eje, y hacia la izquierda o derecha para el otro.

En este punto corresponde llamar a la función Marcha_Paro para que incremente el nivel corriente en los motores, es decir los deje listos para moverse.

Ahora toca subir el bit de inicio para dar arranque al movimiento de los ejes X y Y y erar por el bit de parada que será cuando el contador mande una señal indicando que llego a . Mientras se espera dicha señal se chequeara los extremos de los ejes, con el propósito de antizar que la parte móvil no llegue a dichos fines. Si esto sucediera inmediatamente se endria el movimiento de los ejes y se mostraría en pantalla un correspondiente mensaje de lo rrido.

Una vez que el bit de parada llego, se procederá a bajar el bit de inicio, y conjuntamente se mara a la función Marcha_Paro para que baje el nivel de corriente de los motores hasta que evamente sea llamada la función MOVIMIENTO_XY.

Esta sería la secuencia de trabajo de ésta función. Ver Gráfico 5.12

FUNCION MOVIMIENTO_Z

Esta función esta dedicada a controlar el movimiento del eje Z del taladro, o sea el desplazamiento hacia arriba y hacia abajo de la pieza que sujeta la broca.

Para lograr esto la función MOVIMIENTO_Z carga un contador con la distancia que va a desplazarse el eje Z.

Esta distancia puede ser ingresada por teclado con un valor máximo de 120 pulsos; sin embargo tiene un valor por defecto de 100 pulsos. Los pulsos generados por el Hardware para el movimiento del motor de paso del eje Z son también tomados por el contador mencionado anteriormente, este comenzará a decrementarse hasta llegar a un valor de cero, lo que detendrá el movimiento del eje.

Además esta función se encargará del control de dirección del eje. A continuación se detalla el diagrama de flujo de dicha función.

Como primer paso tenemos una definición de las variables a utilizarse, para luego poner en bajo las salidas del puerto P2A que es el encargado de las señales de control

Luego de esto se carga el contador con el valor de pulsos que controla el desplazamiento del eje Z.

A continuación se indica la dirección que tomará el eje, la que será 1 si el movimiento es hacia arriba y 0 si es hacia abajo.

Ahora toca poner en alto el bit de inicio de movimiento y comienza a censar el bit de parada que será cuando el contador mande una señal indicando que llego a cero. Mientras espera esto se chequeará los fines e inicio de carrera localizados en el eje, de tal manera se garantizara que la parte móvil no chocará con algunos de los extremos del eje, si esto ocurriera se detendria el movimiento y se mostraría un correspondiente mensaje.

Cuando el bit de parada haya llegado, se bajará el bit de inicio y se terminarán las instrucciones de la función MOVIMIENTO_Z. Ver Gráfico 5.13

FUNCION PERFORACION

La función perforación es la encargada de controlar los ejes X, Y, Z, y a la vez activar el movimiento de rotación de la broca para perforar la tarjeta ,es decir esta función ubicará las coordenadas del agujero.

En el diagrama de flujo para esta función se inicia abriendo un archivo en el cual se guardará la información, proveniente del archivo intermedio, sobre cada uno de los agujeros a taladrar, esta información es la distancia que deben recorrer los ejes y el sentido en el que se dirigirán, esto es para ambos ejes X y Y.

Luego, llama a la función 'MOVIMIENTO_Z' a la que se le enviará un valor correspondiente a la distancia y el sentido en que se moverá el eje Z , que para este caso será hacia arriba.

Una vez aquí se ingresará el nombre del archivo que se va a perforar con un máximo de 8 letras. Este archivo es verificado si verdaderamente existe mediante una instrucción, si este no existiera saldría del lazo, llamaría nuevamente a la función MOVIMIENTO_Z, esta vez con dirección hacia abajo, cerraría el archivo y finalizaría la función perforación. En caso contrario de que el archivo especificado exista preguntaría si el dato interrogado es el dato de fin de archivo si esto es verdad saldría nuevamente del lazo llamaría a la función MOVIMIENTO_Z con dirección hacia abajo cerraría el archivo y finalizaría la función perforación. En cambio si esto

fuera falso'tomaria los datos del archivo que esta abierto y se los asignaría a las variables X, SX, Y, SY, que corresponde a las distancias y las direcciones que se deben mover los ejes X y Y.

Luego llamamos a la función MOVIMIENTO_XY, enviándole los respectivos valores de X, SX, Y, SY, con los que se moverán ambos ejes. De igual manera llamamos a una función encargada de pintar en la pantalla los agujeros que se vayan realizando.

Una vez localizado el punto exacto de perforación se llama a la función MOVIMIENTO_Z con dirección hacia abajo para que taladre la tarjeta, una vez taladrado nuevamente llama a la función MOVIMIENTO_Z, con la misma distancia hacia arriba.

Cuando ya esté perforado el agujero regresa a preguntar nuevamente si el siguiente dato es el último del archivo. Si no lo es realiza los mismos pasos mencionados, de lo contrario finaliza la función perforación. Ver Gráfico 5.14

FUNCION SUBE Y BAJA

Esta función tiene como objetivo principal el de mover el eje Z ya sea hacia arriba o hacia abajo, mediante las teclas de flechas hacia arriba y hacia abajo que posee la computadora.

El flujo de esta función es realizado de la siguiente manera.

En primer momento se muestra en pantalla un mensaje que indica las opciones nombradas anteriormente, es decir avance largo, avance corto y utilice las flechas; luego se chequea que el usuario pulse algunas de las teclas extendidas como F1, flecha arriba y flecha abajo. Si el usuario pulsara F1 y el avance del eje Z fuera largo, automáticamente cambiaría a un avance corto. Por el contrario si el avance fuera corto, al presionar F1, se cambiaría de corto a largo.

Si pulsamos las flechas hacia arriba se llamará a la función MOVIMIENTO_Z, enviándole un valor de avance con dirección hacia arriba.

Si pulsamos las flechas hacia abajo se llamará a la función MOVIMIENTO_Z, enviándole un valor de avance con dirección hacia abajo.

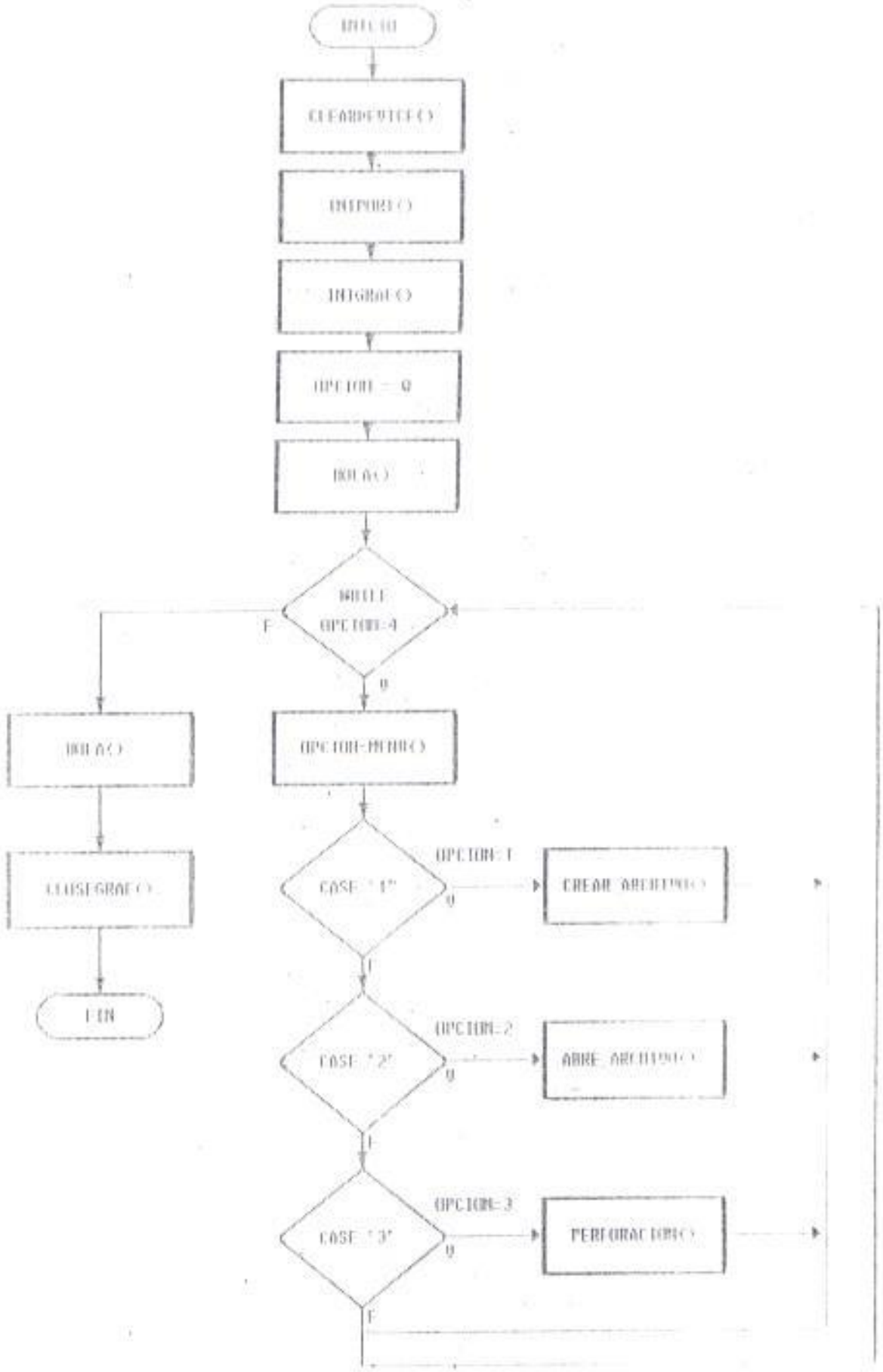
Si pulsamos otra tecla diferente de las extendidas saldremos de ésta función. Ver Gráfico 5.15

FUNCION CHEQUEA_MICROS

Esta función es la encargada de chequear a través del puerto, si se ha llegado con la parte móvil del taladro a uno de los fines de carrera, ya sean estos del eje X, Y o Z.

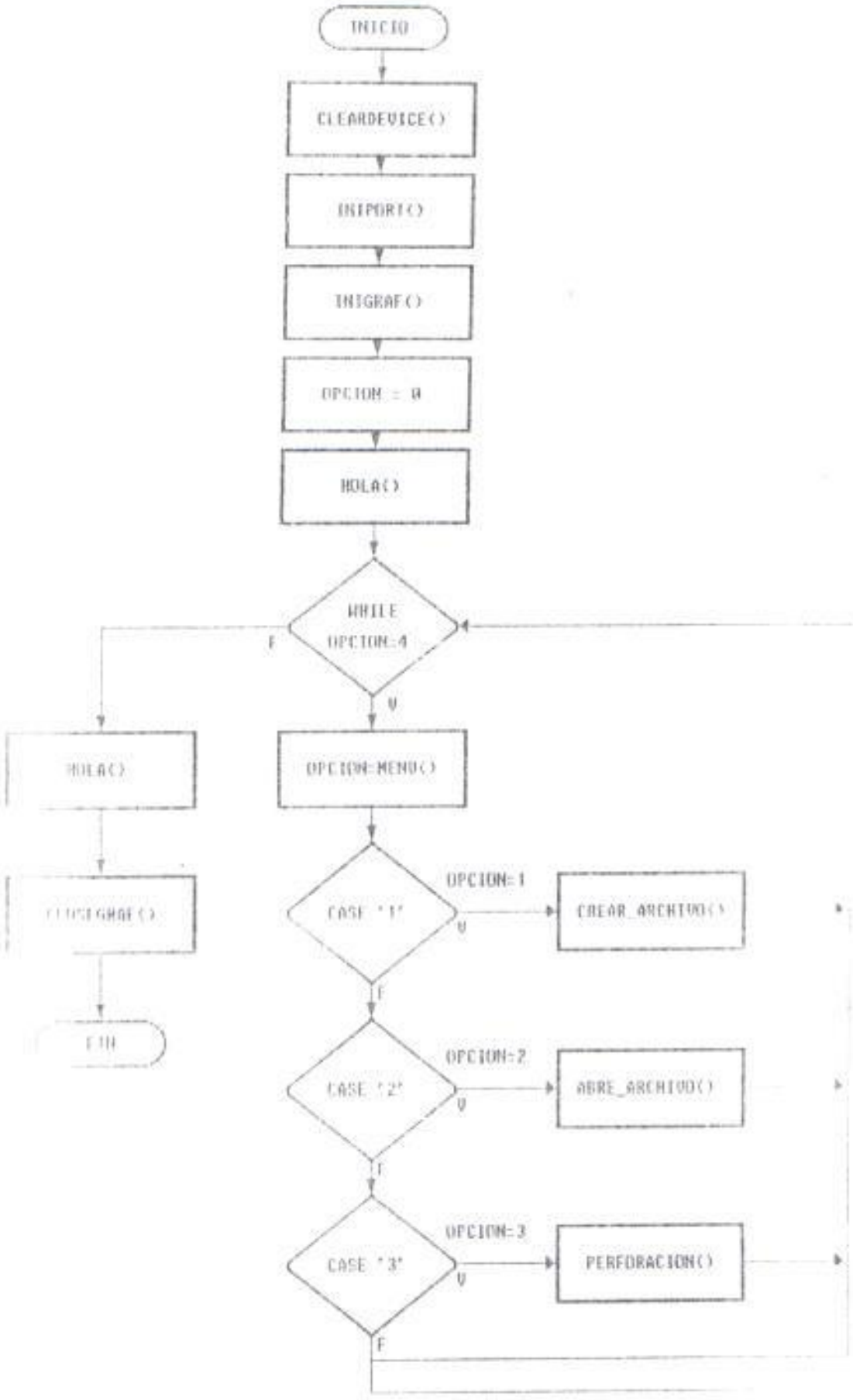
Esta función comienza reconociendo la variable Byte, luego de esto lee la información proveniente del puerto llamado PORT1A el cual es el encargado de recoger los niveles de voltaje de los optoaisladores, siendo este dato guardado en la variable Byte para después preguntarle si este dato es diferente a 0, lo que significaría que se llego a uno de los fines de carrera, si la pregunta es verdadera, o sea que es diferente a 0, entonces tomará la variable global llamada fin, un valor de 1, y mostrará en pantalla un mensaje que nos indicará que se llegó a uno de los topes de alguno de los ejes. Ver Gráfico 5.16

FUNCION MAIN( )

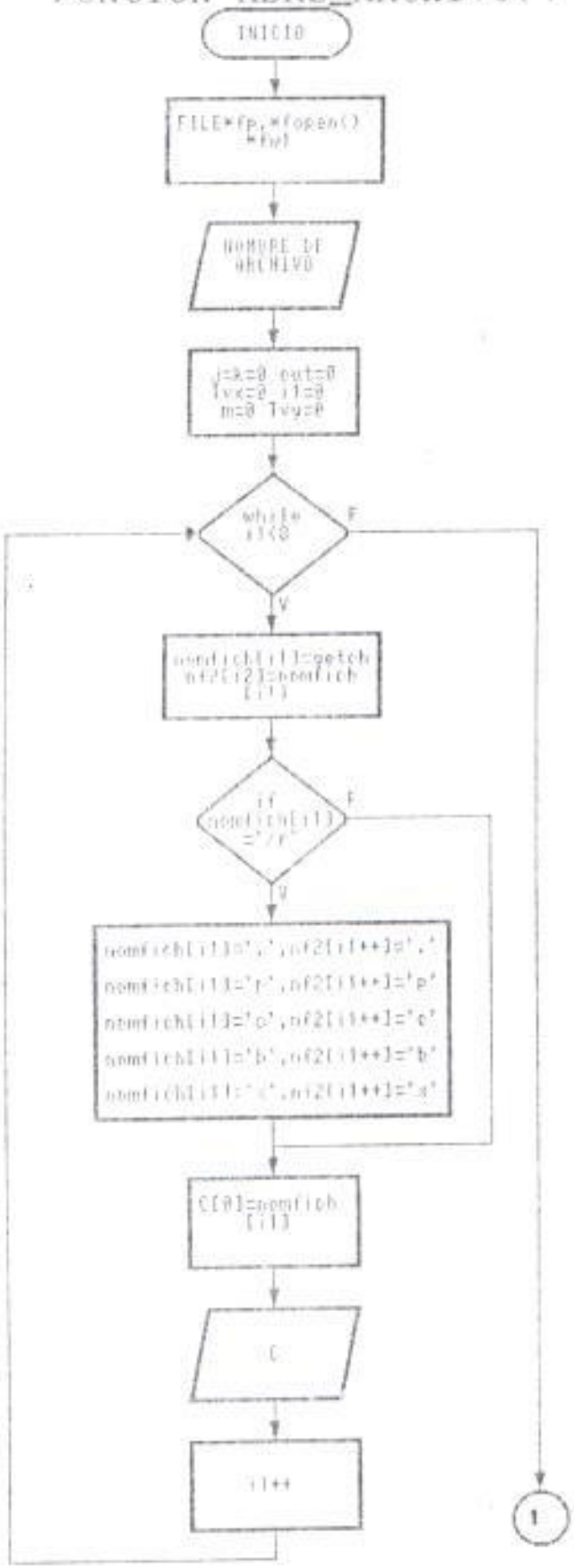


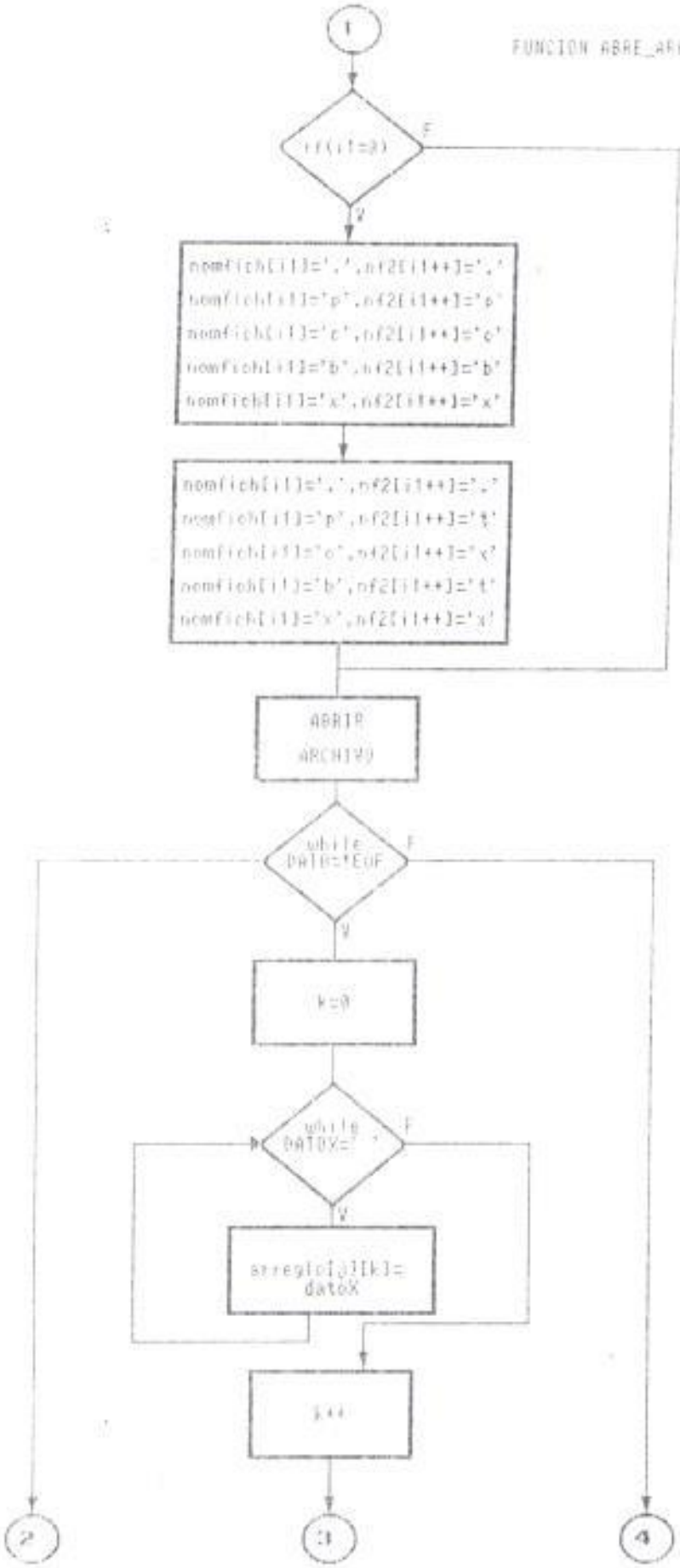
G.5.4.

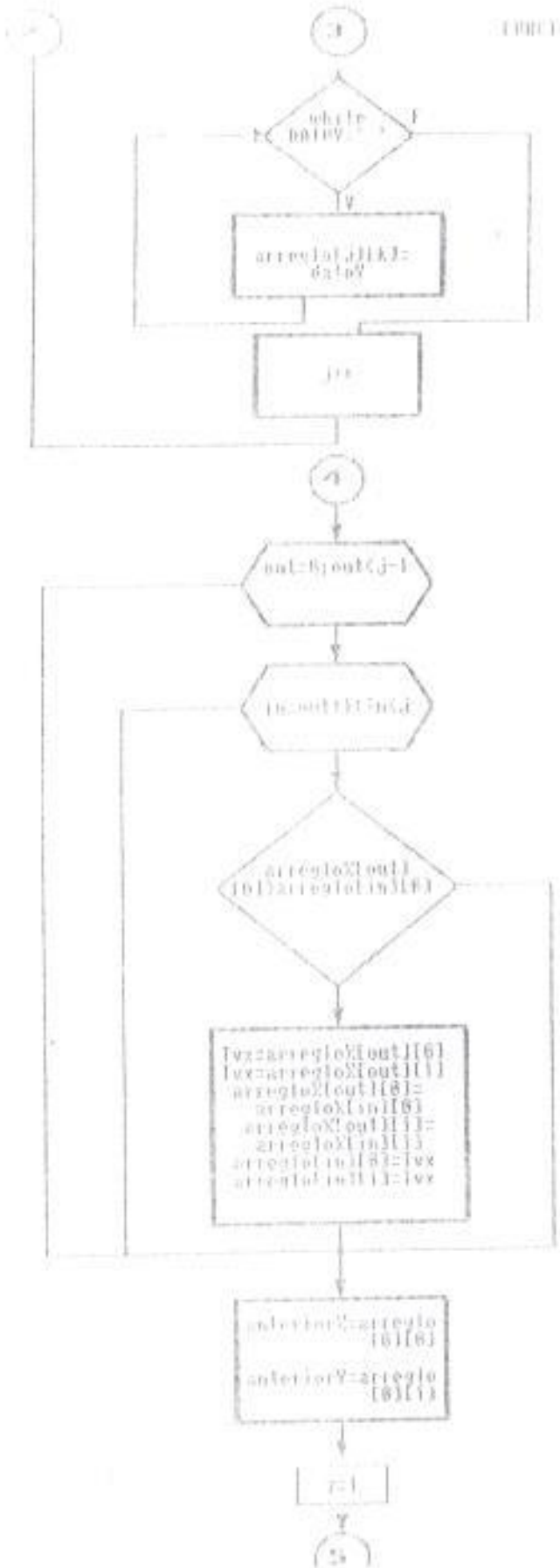
FUNCION PRINCIPAL

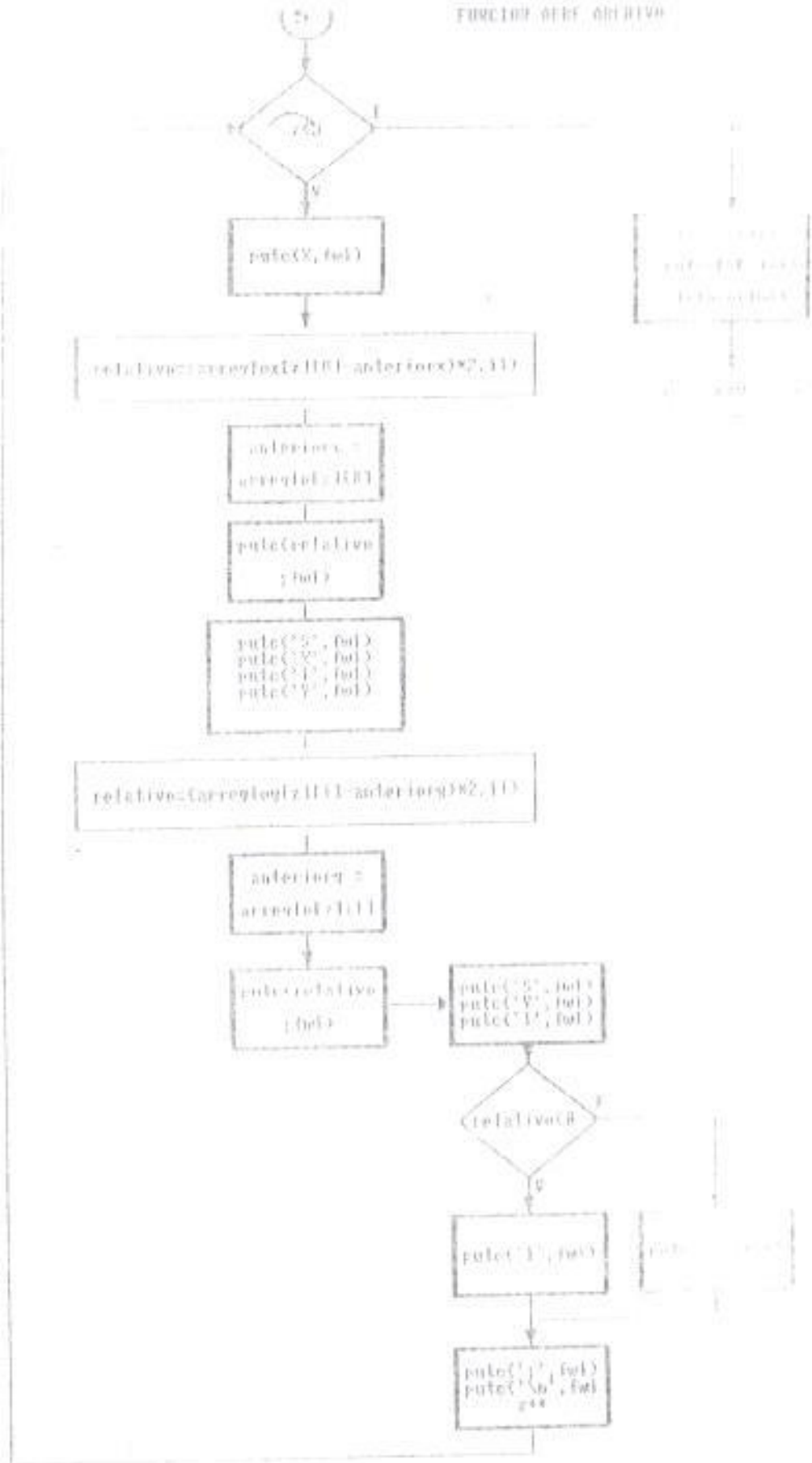


FUNCION ABRE_ARCHIVO( )

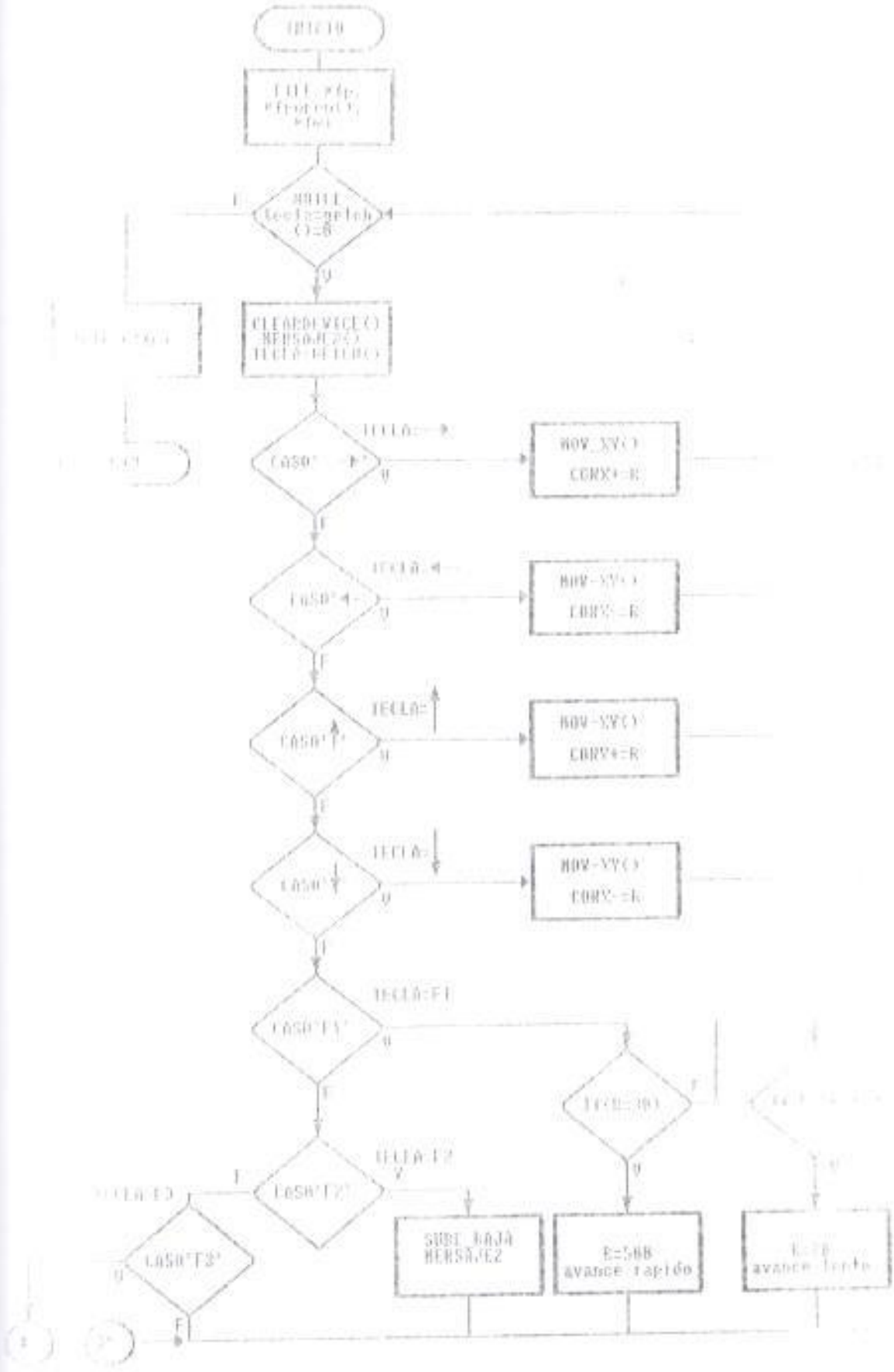


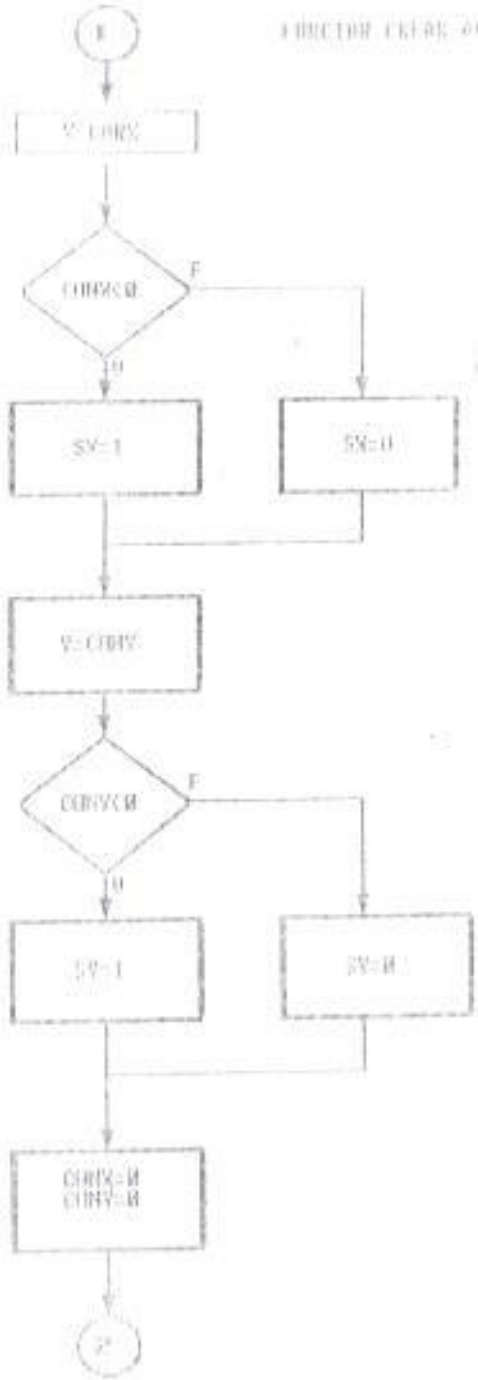

1
FUNCION ABRE_ARCHIVO
if(i!=0)
F
V
nomfich[i]=',',nf2[i++]=','
nomfich[i]='p',nf2[i++]='p'
nomfich[i]='c',nf2[i++]='c'
nomfich[i]='b',nf2[i++]='b'
nomfich[i]='x',nf2[i++]='x'
nomfich[i]=',',nf2[i++]=','
nomfich[i]='p',nf2[i++]='p'
nomfich[i]='c',nf2[i++]='c'
nomfich[i]='b',nf2[i++]='b'
nomfich[i]='x',nf2[i++]='x'
ABRIR
ARCHIVO
while
DATO!=EOF
F
V
k=0
while
DATOX=' '
F
V
arreglo[j][k]=
datoX
k++
2
3
4




putc(X,fwl)
putc('S',fwl)
putc('V',fwl)
putc('I',fwl)
putc('Y',fwl)
anterior =
putc(relative
,fwl)
anterior =
putc('S',fwl)
putc('V',fwl)
putc('I',fwl)
putc('\b',fwl

FUNCION CREAR_ARCHIVO( )




F
F

FUNCION HOLA ( )

FUNCION INIPORT ( )

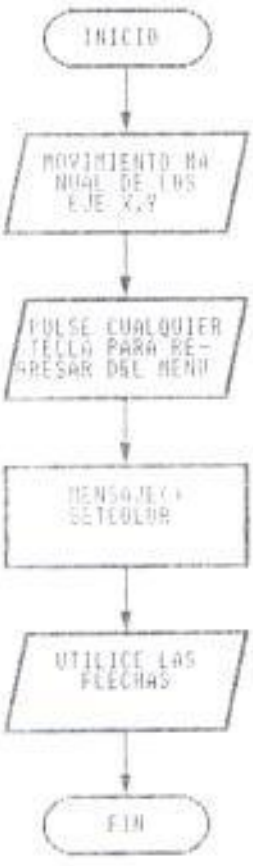
FUNCION MARCHA_PARO ( )

G. S. o

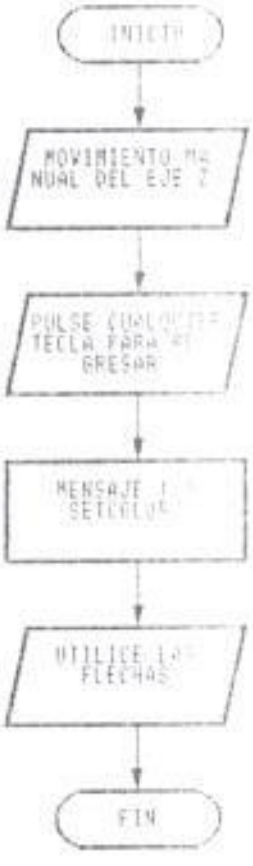
MENSAJE( )

MENSAJE1( )

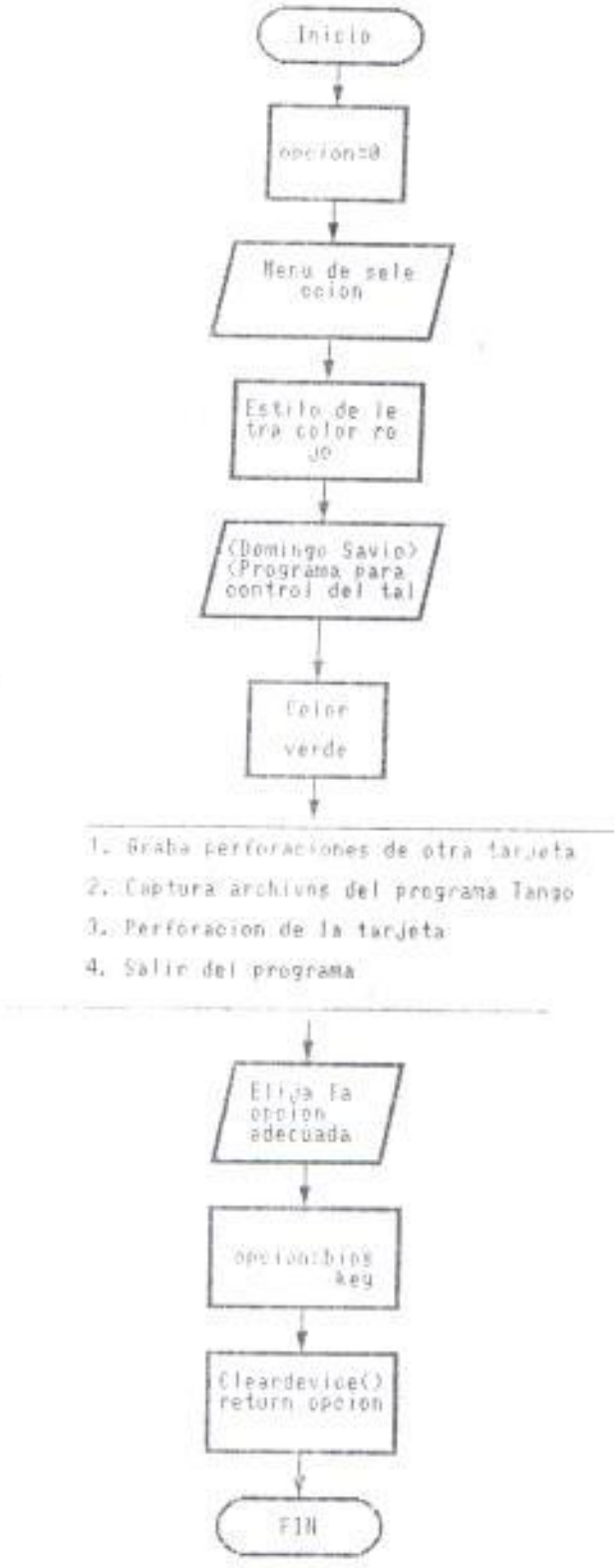
MENSAJE2( )



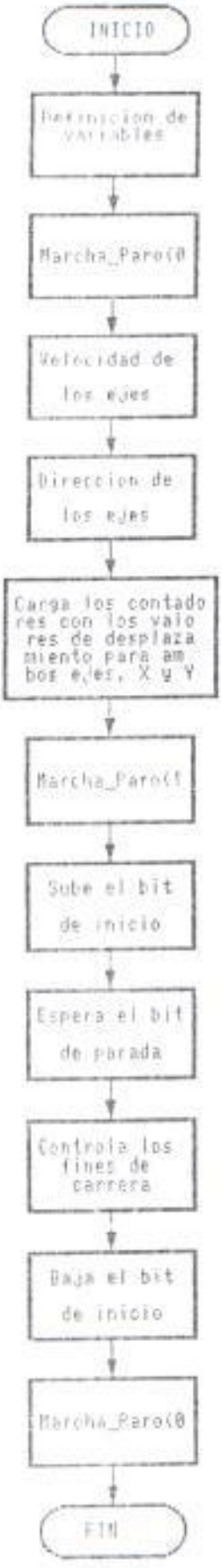
MENSAJE3( )



FUNCION MENU ( )

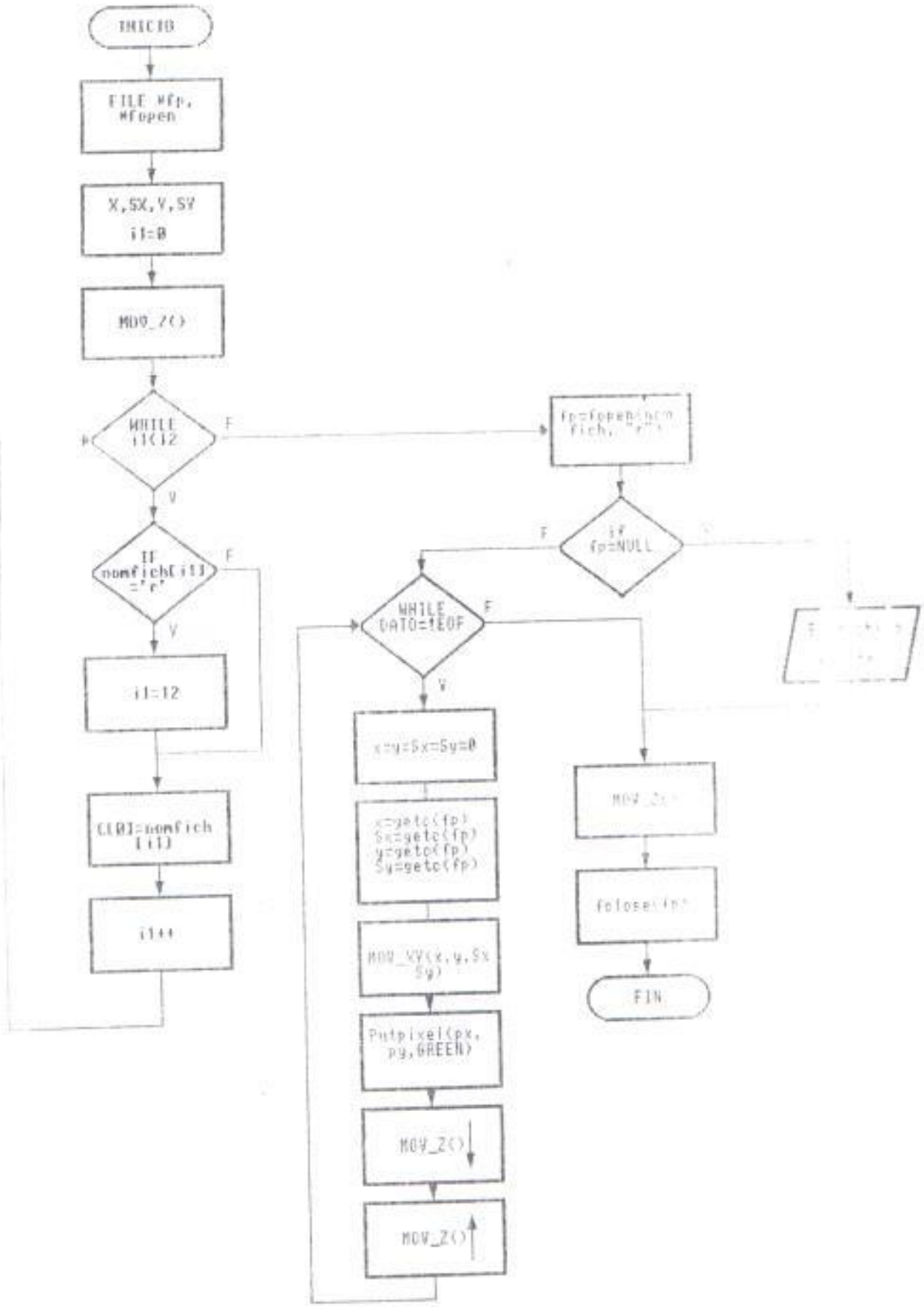


FUNCION MOVIMIENTO_XY( )

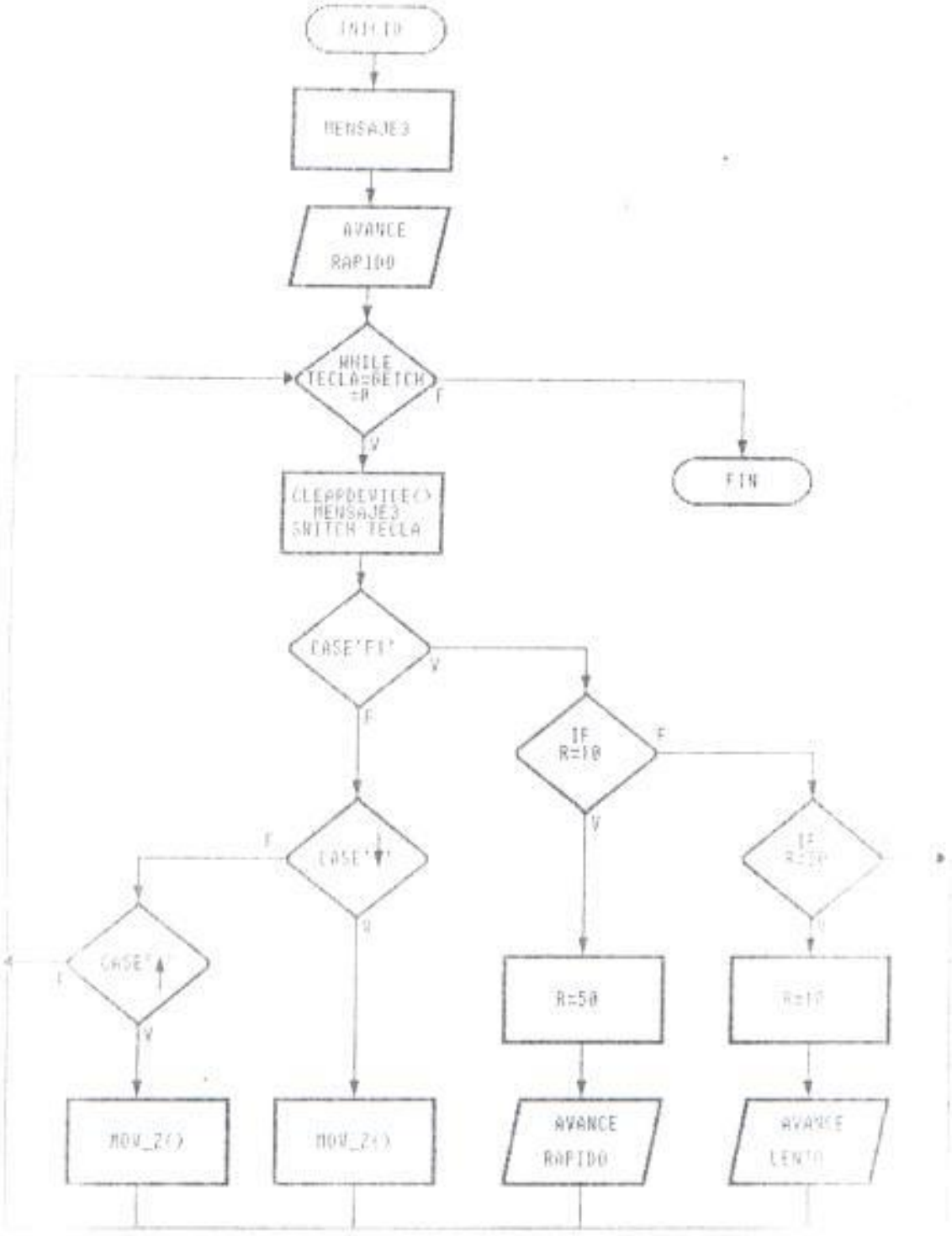


FUNCION MOVIMIENTO_Z ( )

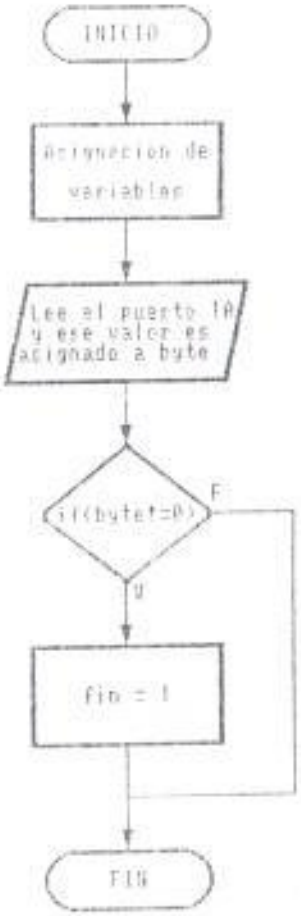
FUNCION PERFORACION ( )



FUNCION SUBE_BAJA( )



FUNCION FIN_TOPE( )



G.5.16

CAPITULO 6

MANUAL DEL USUARIO

INTRODUCCION

Este capítulo tiene como finalidad el de presentar un manual al usuario, que le sirva de ayuda en el manejo del programa realizado para la perforación automática de circuitos impresos.

Aquí se verá el camino que debe tomar el usuario para el manejo de cada una de las opciones que proporcione el programa. Además se indicarán las soluciones para posibles fallas que puedan ocurrir en el transcurso del trabajo con la maquina.

INSTRUCTIVO DEL PROGRAMA

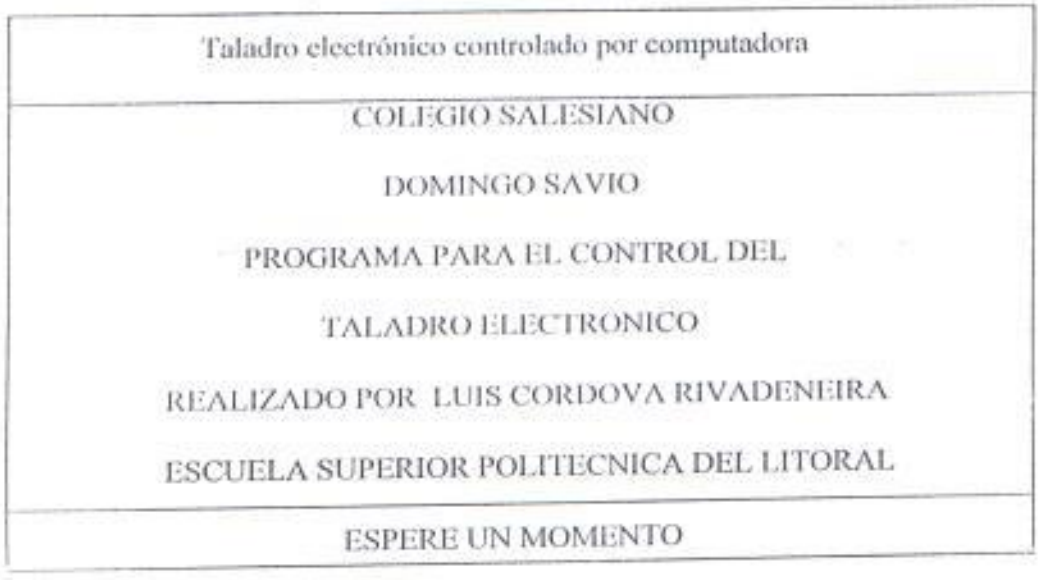
Una vez que el usuario este dispuesto a trabajar con el taladro deberá activar el interruptor de encendido general que da paso a la corriente.

Luego de esto deberá pulsar dos botoneras correspondiente a los ejes X y Y, las que activaran un relé que servirá para energizar los circuitos de control de cada uno de los ejes.

Cuando ya se haya realizado estos pasos se procederá a encender la computadora, y llamaremos al programa que trabaja conjuntamente con la máquina escribiendo taladro y pulsando la tecla enter.

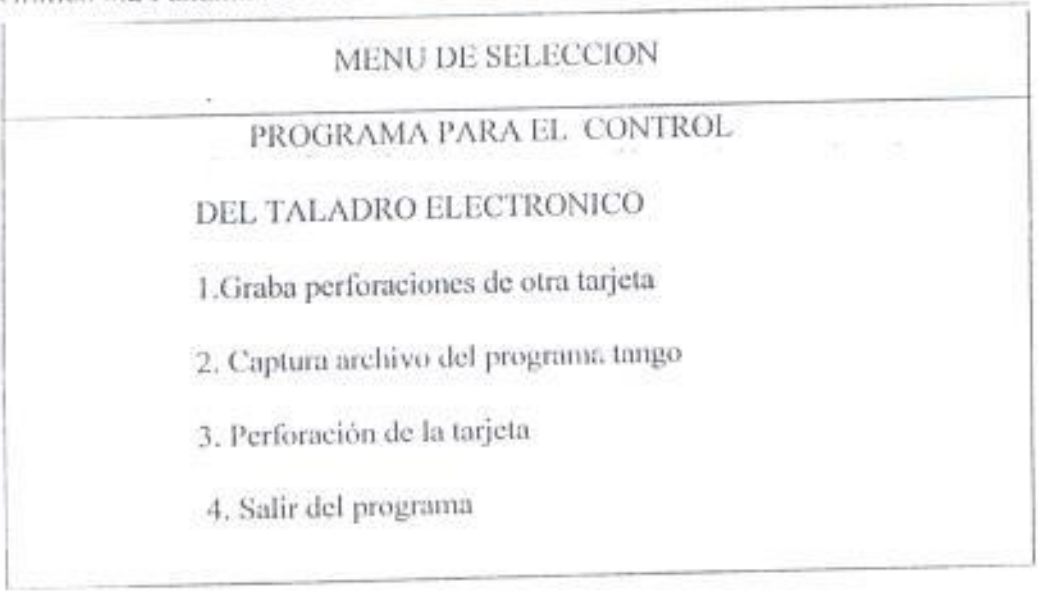
Aparecerá entonces la pantalla de presentación del programa por unos cuantos segundos, como se muestra a continuación en la Gráfico 6 1.

Gráfico 6.1 Pantalla de presentación.



Luego de esto entramos a otra pantalla que corresponde al Menú principal, donde encontraremos cuatro opciones a escoger. Ver gráfico # 6.2.

Gráfico 6.2 Pantalla del Menú de selección



OPCION 1

Si nuestra idea es la de grabar en un archivo las coordenadas de los Pads de una tarjeta ya perforada para luego agujerear una nueva en base a esas coordenadas ya grabadas, debemos seleccionar la opción número uno.

Una vez que se selecciona esta opción se ingresa a una nueva pantalla, la que nos indica que podemos mover cualquiera de las cuatros teclas asignadas con flechas, para de esta manera ubicar la broca sobre el agujero que se desee grabar. Ver gráfico # 6.3.

Gráfico 6.3 Movimiento manual de los ejes X, y Y avance largo

Movimiento manual de los ejes X y Y
UTILICE LAS FLECHAS AVANCE LARGO <F1> Rápido-lento <F2> Subir-bajar <F3> Graba Pad
Pulse cualquier tecla para regresar al menú principal

Al pulsar F1 conmutamos la distancia que se desplazaran los ejes, de un avance largo a un avance corto, indicándose en la pantalla el tipo de avance en que nos encontramos. Ver Gráfico # 6.4

Gráfico 6.4 Movimiento manual de los ejes X, Y avance corto

Movimiento manual de los ejes X y Y
UTILICE LAS FLECHAS AVANCE CORTO <F1> Rápido-lento <F2> Subir-bajar <F3> Graba Pad
Pulse cualquier tecla para regresar al menú principal

Para verificar que nos encontramos justo encima del agujero que deseamos grabar podemos pulsar F2, con lo que ingresaremos a una nueva pantalla que controlará tan solo el movimiento del eje Z con las teclas flecha arriba y flecha abajo. De igual manera aquí podemos pulsar F1 para dar un recorrido largo o corto del eje Z. En esta pantalla no hay manera de mover los ejes X y Y.

Ver gráfico # 6.5

Gráfico 6.5 Movimiento manual del eje Z avance rápido

Movimiento manual del eje Z
UTILICE LAS FLECHAS AVANCE RAPIDO <F1> Rápido-lento <F3> Graba Pad
Pulse cualquier tecla para regresar.

Si ya estamos completamente seguros de la ubicación de la broca con respecto al agujero, debemos pulsar F3 para grabar dicha coordenadas del agujero seleccionado.

Al pulsar cualquier tecla regresaremos a la pantalla anterior que controla tan solo los movimientos en direeción X y Y Gráfico # . Aquí podremos ubicarnos nuevamente encima de otro agujero. Cabe destacar que desde ésta pantalla también podemos grabar las coordenadas del Pad.

Pulsando cualquier tecla regresaremos al menú de selección

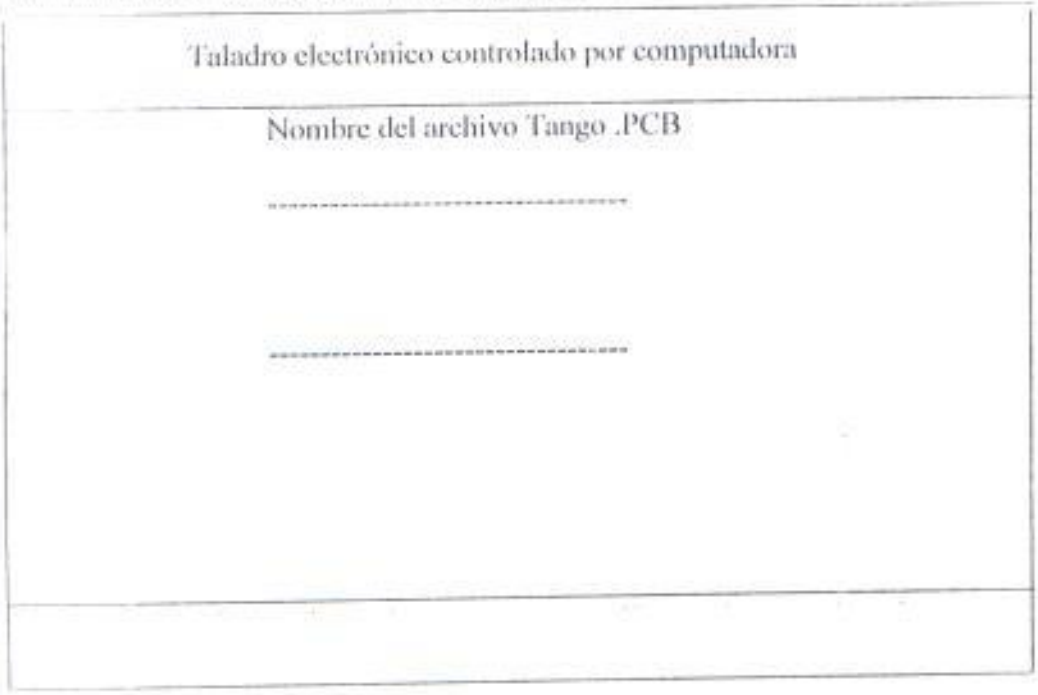
El archivo creado con esta opción y que contiene las coordenadas de los Pad de la tarjeta ya perforada, se llamará siempre **TAL2.TXT**. Cada vez que se entre a esta opción se borrará dicho archivo y quedará listo para recibir los nuevos datos de otra tarjeta ya perforada.

OPCION 2

La opción 2 del menú del programa sirve para llamar a un archivo que contenga algún dibujo realizado con el programa tango, para luego crear otro archivo intermedio con el mismo nombre del archivo donde se encuentra el dibujo.

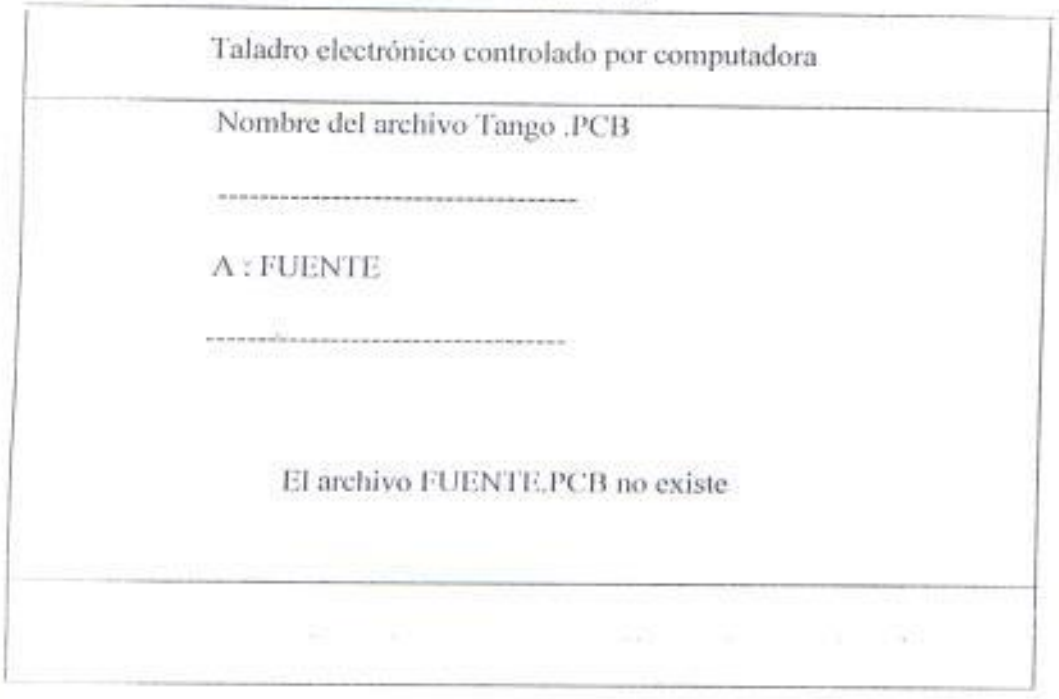
Al pulsar el numero 2 el programa muestra en la pantalla, que debemos ingresar la ruta y el nombre del dibujo en tango, tal como se ve en el gráfico # 6.6

Gráfico 6.6 Pantalla de Abrir un Archivo de extensión PCB



Si este archivo no existiera automáticamente sale en pantalla un mensaje que dice que el archivo con ese nombre o con esa ruta no existe, como se indica a continuación en el siguiente gráfico # 6.7

Gráfico 6.7 Pantalla de Archivo PCB no existente



Por el contrario, si existiera, comenzará a crearse el archivo intermedio con las coordenadas de los Pad y con los formatos revisados.

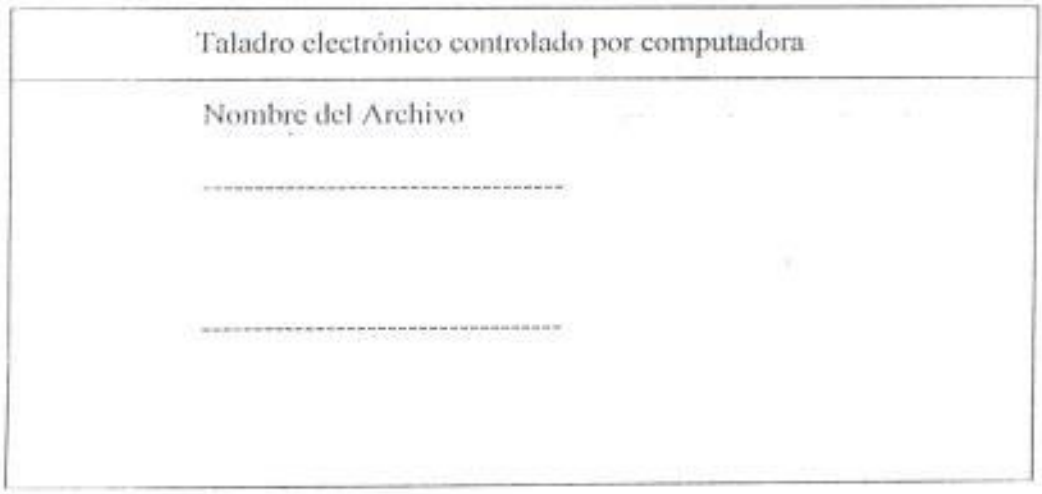
OPCION 3

Al pulsar la opción 3 aparecerá una pantalla que nos preguntará el nombre de archivo intermedio que contiene las coordenadas de los agujeros que se van a taladrar.

El nombre y la ruta del archivo intermedio serán los mismos que los nombres y la ruta del archivo del dibujo de programa tango con la extensión .PCB.

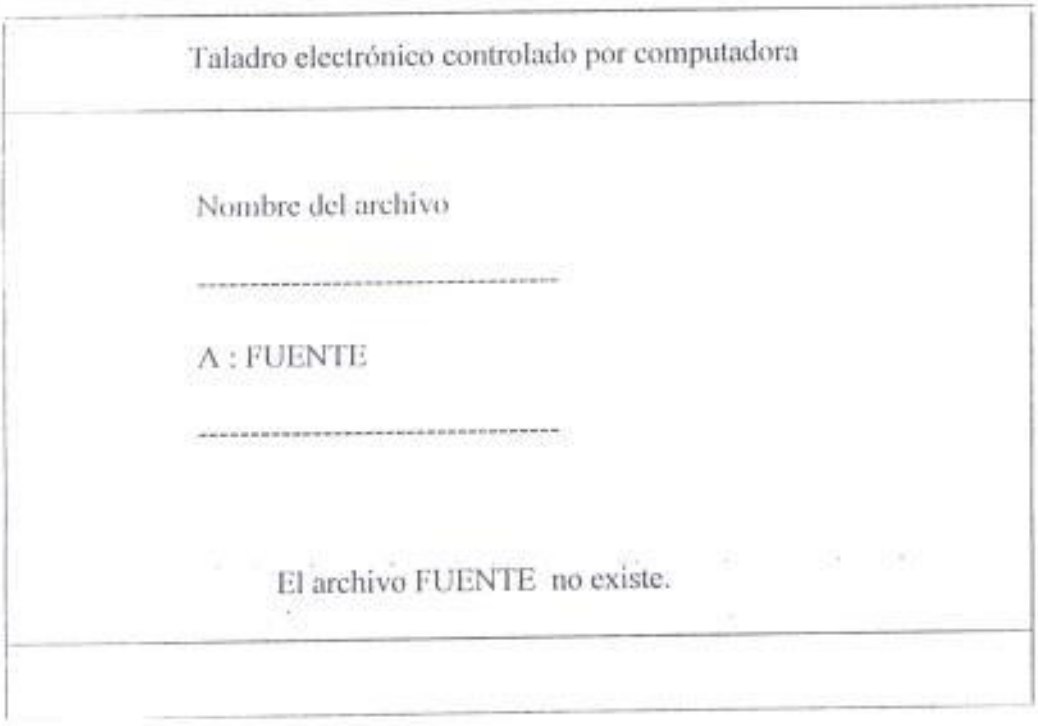
En cambio si queremos perforar la tarjeta con los datos del archivo creado en base a la opción 1 del menú, debemos poner **TAL2.TXT**. Ver gráfico # 6.8

Gráfico 6.8 Pantalla de Abrir un Archivo Intermedio de extensión TXT



Si el nombre del archivo que contiene las coordenadas de los Pad no se encuentra en la ruta especificada se mostrará un mensaje en pantalla indicando que este archivo no existe. Con lo cual debemos pulsar cualquier tecla para regresar al menú principal. Ver gráfico # 6.9

Gráfico 6.9 Pantalla de Archivo TXT no existente



En cambio, si el nombre y la ruta del archivo son válidos se proceder a taladrar la nueva tarjeta, mostrándonos además en pantalla mediante puntos, todos los agujeros que se vayan perforando.

Una vez finalizado la perforación debemos de pulsar cualquier tecla para nuevamente regresar al menú principal

OPCION 4

Por ultimo, si queremos finalizar la sesión de trabajo pulsamos la opción 4, con lo que aparecerá una pantalla de despedida similar a la de presentación y por un lapso igual de tiempo. Gráfico # 6.1

Luego de esto solo queda poner en OFF el interruptor general de la máquina para que ésta quede apagada.

Acto seguido procedemos a apagar la computadora.

APENDICE A

LISTADO DEL PROGRAMA

LISTADO DEL PROGRAMA

```
/* DECLARACION DE INCLUDES */

#ifdef __TINY__ .

#error BGIDEMO.

#endif

#include <dos.h>

#include <bios.h>

#include <math.h>

#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

#include <stdarg.h>

#include <graphics.h>

#define ESC   0x1b        /* Define tecla de escape */

char *Fonts[] = {  "DefaultFont",  "TriplexFont",  "SmallFont",  "SansSerifFont",

                   "GothicFont" };

char *LineStyles[] = {"SolidLn", "DottedLn", "CenterLn",  "DashedLn",  "UserBitLn" };

char *TextDirect[] = { "HorizDir",  "VertDir" };

char *HorizJust[] = {  "LeftText",  "CenterText",  "RightText"  };

char *VertJust[] = { "BottomText",  "CenterText",  "TopText"

struct PTS { int x, y; };

int   GraphDriver;     /* Dispositivo controlador de gráficos  */

int   GraphMode;       /* El valor del modo gráfico   */

double AspectRatio;     /* relación de aspecto pixels pantalla */
```

```
int   MaxX, MaxY;      /* máxima resolución de la pantalla */
int   MaxColors;       /* máximo # de colores */
int   ErrorCode;       /* error de gráficos */
struct palettetype palette; /* Usado para leer la información de la paleta */
/* DEFINICION DEL PUERTO PARALELO P1 Y VARIABLES DE SALIDA */
int port1a=0x1B0;  /* UTILIZADO EN LA ENTRADA - TOPES -  */
int port1b=0x1B1;  /* UTILIZADO EN LA SALIDA - VELOCIDAD - */
int port1c=0x1B2;  /* NO UTILIZADO */
int regp1=0x1B3;
/* DEFINICION DEL PUERTO PARALELO P2 Y VARIABLES DE SALIDA */
int port2a=0x1B4;  /* UTILIZADO EN LA SALIDA - EJES X,Y,Z - */
int port2b=0x1B5;  /* UTILIZADO EN LA ENTRADA - EJES X,Y,Z - */
int port2c=0x1B6;  /* NO UTILIZADO */;
int regp2=0x1B7;
/* DEFINICION DEL PUERTO DEL CONTADOR 1 */
int count1a=0x1B8;  /* CONTADOR T-A EJE X */
int count1b=0x1B9;  /* CONTADOR T EJE X */
int count1c=0x1BA;  /* CONTADOR DEL EJE Z */
int regc1=0x1BB;
/* DEFINICION DEL PUERTO DEL CONTADOR 2 */
int count2a=0x1BC;  /* CONTADOR T-A EJE Y */
int count2b=0x1BD;  /* CONTADOR T EJE Y */
int count2c=0x1BE;  /* NO UTILIZADO */
int regc2=0x1BF;
```

```
/* DECLARACION DE FUNCIÓNES */
void iniport(void);          /* Inicializa los puertos */
void movimiento_xy(int,int,int,int); /* Rutina para mover los ejes X y Y */
void movimiento_z(int,int);          /* Rutina para mover el eje Z  */
void marcha_paro(int,int);    /* Rutina para enviar mas corriente a los motores según sea el
caso */
void Chequea_micros(void);          /* Detecta si algún eje llego a un extremo */
void Abre_archivo(void);      /* Abre un archivo para guardar coordenadas */
void Crear_archivo(void);     /* Crea un archivo de coordenadas */
void Perforacion(void);       /* Rutina para perforar la tarjeta */
void Initialize(void);        /* inicializa el modo gráfico */
void Hola(void);              /* presentación y despedida */
void MainWindow(char *header); /* Proporciona una ventana principal */
void StatusLine(char *msg);   /* Proporciona una línea de mensaje  */
void gprintxy(int posx,int posy,char *formato,...); /* permite escribir en pantalla  */
void DrawBorder(void);        /* dibuja contorno a la pantalla */
void changetextstyle(int font, int direction, int charsize);  /* Cambia el estilo de letra */
void Sube_baja(void);         /* Rutina para subir o bajar al taladro  */
void mensaje(void);           /* Presenta mensajes en pantalla  */
void mensaje1(void);          /* Presenta mensajes en pantalla  */
void mensaje2(void);          /* Presenta mensajes en pantalla  */
void mensaje3(void);          /* Presenta mensajes en pantalla  */
int  gprintf(int *xloc, int *yloc, char *fmt, ... ); /* permite escribir en pantalla  */
int  Menu(void);              /* pantalla del menú de selección */
```

```
/*  Variables globales  */
int error;       /* Determina un error  */
int velocidad;   /* Determina la velocidad del movimiento del eje  */
int outp2a;      /* Dato que sale por el puerto port2A  */
int ftx, itx, fty, ity, ftz, itz;  /* f=final, i=inicio, t=tope */
/*******************************/
/*   PROGRAMA PRINCIPAL      */
/*******************************/
void main()
{
  int opcion=0;        /* Control de la opción */
  clrscr();            /* Limpia la pantalla  */
  iniport();           /* Inicializa los puertos */
  Initialize();        /* Inicializa el modo gráfico  */
  setbkcolor(BLACK);
  Hola();              /* Pantalla de presentación  */
  while (opcion!=52)
  {
   opcion=Menu();      /* muestra pantalla menú */
   switch(opción)
   {
     case 49 : Crear_archivo(); break;
     case 50 : Abre_archivo();break;
     case 51 : Perforación();break;
```

```
    default :  break;
   };
 };
 Hola();            /* Pantalla de despedida  */
 closegraph();      /* activa el modo texto */  };
/*******************************************************/
/* RUTINA PARA LA PERFORACION DE UNA TARJETA */
/*******************************************************/
void Perforación()
{
  struct viewporttype viewinfo;   /* Estructura para leer los viewport */
  int i1=0;
  int x, sx, y, sy, px=100, py=250;
  FILE *fp, *fopen();
  char dato, c[1]={0}, nomfich[13]={0};
/*  movimiento_z(80,0);*/
  MainWindow( "Taladro electrónico controlado por computadora " );
  setcolor(GREEN);
  getviewsettings( &viewinfo );
  changetextstyle( TRIPLEX_FONT, HORIZ_DIR, 4 );
  settextjustify( CENTER_TEXT, CENTER_TEXT );
  outtextxy( 200, 50, " Nombre del archivo");
  setcolor(RED);
  gprintxy(200,100, "  ----------");  /* Pide el nombre del Archivo  */
```

```
gprintxy(200,140, "  ----------"); /*  a perforar  */
setcolor(YELLOW);
while (i1<12)
 {
 if ((nomfich[i1]=getche()) == '\r')
  {
   nomfich[i1]='\x0';
   i1=12;
  };
  c[0]=nomfich[i1];
  gprintxy(150+(i1*15), 120, c);
  i1++;
};
fp=fopen(nomfich,"r");          /* Si fp es NULL  entonces  */
if (fp==NULL)                   /* el Archivo no existe  */
{
 gprintxy( 300, 180,"El archivo %s no existe.", nomfich );
}else
{
 while ((dato=getc(fp)) != EOF)   /* Lee los datos de Archivo */
  {
   x=0;
   if (dato=='X')
    {
```

```
while ((dato=getc(fp)) != 'S')
{
 x=(x*10)+(dato-48);     /* Guarda la coordenada en X */
};
dato=getc(fp);
sx=(getc(fp)-48);    /* Guarda la dirección de movimiento en X */
dato=getc(fp);
y=0;
if (dato=='Y')
{
  while((dato=getc(fp)) != 'S')
  {
   y=(y*10)+(dato-48);    /* Guarda la coordenada en Y */
  };
  dato=getc(fp);
  sy=(getc(fp)-48);  /* Guarda la dirección de movimiento en Y */
  dato=getc(fp);
  dato=getc(fp);
 }else
 {
  error=2;
 };
}else
{
```

```
        error=1;
    }
    if (error==0)
    {
        movimiento_xy(x,sx,y,sy);  /* Manda las coordenadas a la */
                                   función que mueve los ejes X y Y */
        if (sx==0) px-=(x/100);
        if (sx==1) px+=(x/100);
        if (sy==0) py+=(y/100);
        if (sy==1) py-=(y/100);
        putpixel( px, py, GREEN ); /* Pinta un punto perforado  */
                                   /* en la pantalla  */
        movimiento_z(100,1);   /* Mueve para abajo el eje Z  */
        movimiento_z(100,0);   /* Mueve para arriba el eje Z  */
    }else
    {
        gotoxy(10,22);
        printf("Error en el formato del archivo %12s.", nomfich);
    /*  getch();*/
        break;
    };
  };
};
/*  movimiento_z(80,1);*/
```

```
  fclose(fp);
  getch();
 }
/*****************************************************/
/* CREA UN ARCHIVO EN BASE A OTRA TARJETA */
/*****************************************************/
void Crear_archivo(void)
{
  struct viewporttype viewinfo;    /* Estructura para leer el viewport */
  int tecla, R=500, CONX=0, CONY=0, i;
  char numero[25];
  FILE *fp, *fopen(), *fw1;
  char dato, nomfich[13]={"tal2.txt"};
  getviewsettings( &viewinfo ); /* Lee la configuración del viewport */
  mensaje2();
  outtextxy( 200, 150, " Avance rápido");
  fw1=fopen(nomfich,"w");
  while ( (tecla=getch() ) == 0 )
    {
     cleardevice();
     mensaje2();
     tecla=getch();
     switch (tecla)  /* Si se pulso las flechas mueve los ejes en ese sentido */
         {
```

```
case 77 : movimiento_xy(R, 1, 2, 0 ); CONX+=R; break;
case 75 : movimiento_xy(R, 0, 2, 0 ); CONX-=R; break;
case 72 : movimiento_xy(2, 0, R, 1 ); CONY+=R; break;
case 80 : movimiento_xy(2, 0, R, 0 ); CONY-=R; break;
case 59 : if (R==30) /*Chequea si se pulso alguna tecla funciónal */
          R = 500;
          else if (R==500)
          R = 30;
          switch(R)
           {
            case 500 : outtextxy( 200, 150, " Avance rápido"); break;
            case 30 : outtextxy( 200, 150, " Avance lento  "); break;
           }; break;
case 60 : Sube_baja();
          mensaje2();
          break;
case 61 : putc('X',fw1);  /* F3 graba la coordenada con el */
          if (CONX<0)     /* formato apropiado en el Archivo */
        {       itoa(CONX*(-1),numero,10);  }
                else      {
        if (CONX>2)  {
                itoa(CONX,numero,10);
         }else   {
        número[0]='3';
```

```
numero[1]='\x0';
};
};
i=0;
while (numero[i]!='\x0')
putc(numero[i++],fw1);
putc('S',fw1);
putc('X',fw1);
if (CONX<0)
{
putc('1',fw1);
}else
{
putc('0',fw1);
};
putc('Y',fw1);
if (CONY<0)
{
itoa(CONY*(-1),numero,10);
}else
{
if (CONY>3)
{
itoa(CONY,numero,10);
```

```
        }else
        {
        numero[0]='3';
        numero[1]='\x0';
        };
        };
        i=0;
        while (numero[i]!='\x0')
        putc(numero[i++],fw1);
        putc('S',fw1);
        putc('Y',fw1);
        if (CONY<0)
        {
        putc('1',fw1);
        }else
        {
        putc('0',fw1);
        };
        putc(';',fw1);
        putc('\n',fw1);
        CONX=0;
        CONY=0;
        break;
default : break;
```

```
            };
    };
    putc(EOF,fw1);  /* Indica el fin de Archivo */
    fclose(fw1);    /* Cierra el Archivo  */
};
/*******************************************/
/* UTILIZA FLECHAS PARA SUBIR O BAJAR  */
/*******************************************/
void Sube_baja(void)
{
  int tecla, R=50;
  struct viewporttype viewinfo;   /* Estructura para leer el viewport */
  getviewsettings( &viewinfo );   /* Lee configuración de viewport */
  mensaje3();
  outtextxy( 200, 150, " Avance rápido");
  while ( (tecla=getch() ) == 0 )
   {
    cleardevice();
    mensaje3();
    tecla=getch();
    switch (tecla)   /* Chequea si se pulsa un tecla funciónal */
       {
        case 59 : if (R==10)  /* F1 varia el avance de lento a rápido */
                    R = 50;
```

```
                else if (R==50)
                R = 10;
                 switch(R)
                 {
                     case 50 : outtextxy( 200, 150, " Avance rapido"); break;
                     case 10 : outtextxy( 200, 150, " Avance lento"); break;
                 }; break;
          case 72 : movimiento_z(R,0); break; /* Mueve el eje Z hacia abajo */
          case 80 : movimiento_z(R,1); break; /* Mueve el eje Z hacia arriba */
          default : break;
         };
    };
};
/*****************************************************************/
/* CONVERSION DE UN ARCHIVO PROCEDENTE DE TANGO
Abre un Archivo de Tango, toma solamente las coordenadas de los PADs
a taladrar y crea un Archivo con su propio formato */
/*****************************************************************/
void Abre_archivo(void)
{
  struct viewporttype viewinfo;    /* Estructura para leer viewport */
  int i1=0, anteriorx=0, anteriory=0, relativo=0;
  int x, sx, y, sy, in, i, z, out, j=0, k=0, Tvx, Tvy, hx, hy;
  FILE *fp, *fopen(), *fw1;
```

```
char dato, numero[25], c[1]={0}, nomfich[13]={0}, nf2[13]={0};
int arreglox[1000][2]={0};
MainWindow( "Taladro electrónico controlado por computadora " );
setcolor(GREEN);
getviewsettings( &viewinfo );    /* Read viewport settings     */
changetextstyle( TRIPLEX_FONT, HORIZ_DIR, 2 );
settextjustify( CENTER_TEXT, CENTER_TEXT );
outtextxy( 200, 50, "Nombre archivo TANGO .PCB");
setcolor(RED);
gprintxy(200,100, "   ----------");
gprintxy(200,140, "   ----------");
setcolor(YELLOW);
while (i1<8)       /* Pide el nombre de un Archivo */
{                  /* procedente de Tango */
 nomfich[i1]=getch();
 nf2[i1]=nomfich[i1];
 if (nomfich[i1] == '\r')
 {
  nomfich[i1]='.'; nf2[i1++]='.';
  nomfich[i1]='p'; nf2[i1++]='t';
  nomfich[i1]='c'; nf2[i1++]='x';
  nomfich[i1]='b'; nf2[i1++]='t';
  nomfich[i1]='\x0'; nf2[i1++]='\x0';
 };
```

```
c[0]= nomfich[i1];
gprintxy(150+(i1*15), 120, c);
i1++;
};
if (i1==8)
{
 nomfich[i1]='.'; nf2[i1++]='.';
 nomfich[i1]='p'; nf2[i1++]='t';
 nomfich[i1]='c'; nf2[i1++]='x';
 nomfich[i1]='b'; nf2[i1++]='t';
 nomfich[i1]='\x0'; nf2[i1++]='\x0';
};
 fp=fopen(nomfich,"r");
if (fp==NULL)
{
 gprintxy( 300, 180,"El archivo %s no existe.", nomfich );
 getch();
}else
{
 fw1=fopen(nf2,"w");
 while ((dato=getc(fp)) != EOF)
 {              /* Realiza un filtro de los datos del Archivo de */
  if (dato!='P')   /* Tango, de tal forma que solo toma los datos */
  {              /* correspondiente a los PADs y no de otras entidades */
```

```
        while (dato!='\n') dato=getc(fp);
        continúe;
    };
  dato  getc(fp);
  while (dato !='')
      {
       dato=getc(fp);
      };
 dato=getc(fp);
 while (dato !='')
      {
       dato=getc(fp);
      };
dato=getc(fp);
x=0;
k=0;
if (dato==' ')
{
    while ((dato=getc(fp)) != ' ')
    {
     x=(x*10)+(dato-48);
    };
    arreglox[j][k]=x;   /* Guarda la coordenada X en un Arreglo  */
    k++;
```

```
    y=0;
    if (dato==' ')
    {
     while((dato=getc(fp)) != ' ')
     {
      y=(y*10)+(dato-48);
     };
     arreglox[j][k]=y;   /* Guarda la coordenada Y en un Arreglo  */
     j++;
     dato=getc(fp);
     while (dato!='\n')  {dato=getc(fp);};
     }else
    {
     error=2;
    };
} else
{
    error=1;
}
if (error==0)
{
}else
{
    gotoxy(10,22);
```

```
        printf("Error en el formato del archivo %12s.", nomfich);
      getch();
     };
    };
  };              /* Ordena las coordenadas de los PADs para */
/* realizar las perforaciones en orden */
  for (out=0; out<j-1; out++)
   for (in=out+1; in<j; in++)
        if (arreglox[out][0] > arreglox[in][0] )
           {
            Tvx = arreglox[out][0];
            Tvy = arreglox[out][1];
            arreglox[out][0]=arreglox[in][0];
            arreglox[out][1]=arreglox[in][1];
            arreglox[in][0]=Tvx;
            arreglox[in][1]=Tvy;
           };
   anteriorx=arreglox[0][0];
   anteriory=arreglox[0][1];
  z=1;
  while ( z<j )
     {
        i=0;
        putc('X',fw1);
```

```
relativo=(arreglox[z][0]-anteriorx)*2.11;
anteriorx=arreglox[z][0];
if ( relativo != 0)
 {
  itoa(relativo,numero,10);
 }else
  {
   numero[0]='3';
   numero[1]='\x0';
  };
while (numero[i]!='\x0')
putc(numero[i++],fw1);   /* Escribe las coordenadas de los PADs  */
putc('S',fw1);          /* en el nuevo Archivo intermedio con  */
putc('X',fw1);          /* formato X##SX##Y##SY## */
putc('1',fw1);
i=0;
putc('Y',fw1);
relativo=(arreglox[z][1]-anteriory)*2.11;
anteriory=arreglox[z][1];
if (relativo < 0)
 {
  itoa(relativo*(-1),numero,10);
 }else
  {
```

```
        if (relativo > 2)
        {
         itoa(relativo,numero,10);
        }else
         {
          numero[0]='3';
          numero[1]='\x0';
         };
    };
while (numero[i]!='\x0')
putc(numero[i++],fw1);
putc('S',fw1);
putc('Y',fw1);
if (relativo < 0)
 {
putc('1',fw1);
}else
   {
    putc('0',fw1);
   };
putc(';',fw1);
putc('\n',fw1);
z++;
}
```

```
   fclose(fp);     /* Cierra el Archivo de Tango */
   putc(EOF,fw1);  /* Escribe fin de Archivo intermedio */
   fclose(fw1);    /* Cierra el Archivo intermedio */
}
/*************************************/
/* DA UN MENSAJE EN LA PANTALLA */
/*************************************/
void mensaje(void)                    /* Función que permite mostrar un */
{                                     /* mensaje en pantalla  */
  setfillstyle(SOLID_FILL,BLUE);
  bar3d(115,312,340,212,4,1);
  setcolor(GREEN);
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
  outtextxy(220,240," <F1> - RAPIDO-LENTO");
  outtextxy(220,260," <F2> - SUBIR-BAJAR");
  outtextxy(220,280," <F3> - GRABA PAD");
};
/*************************************/
/* DA UN MENSAJE EN LA PANTALLA */
/*************************************/
void mensaje1(void)
{                     /* Función que permite mostrar un */
                      /* mensaje en pantalla  */
  setfillstyle(SOLID_FILL,BLUE);
```

```
bar3d(115,312,340,212,4,1);
setcolor(GREEN);
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
outtextxy(220,240," <F1> - RAPIDO-LENTO");
outtextxy(220,260," <F3> - GRABA PAD");
};
/*************************************/
/* DA UN MENSAJE EN LA PANTALLA */
/*************************************/
void mensaje2(void)
{
                /* Función que permite mostrar un */
                /* mensaje en pantalla  */
  MainWindow( "Movimiento manual de los ejes X y Y  " );
  StatusLine( "Pulse cualquier tecla para regresar al menú principal." );
  mensaje();
  changetextstyle( TRIPLEX_FONT, HORIZ_DIR, 3 );
  settextjustify( CENTER_TEXT, CENTER_TEXT );
  setcolor(GREEN);
  outtextxy( 200, 130, " Utilice las flechas");
  setcolor(RED);
  }
```

```
/*****************************************/
/* DA UN MENSAJE EN LA PANTALLA */
/*****************************************/
void mensaje3(void)        /* Función que permite mostrar un */
                           /* mensaje en pantalla  */
 {
  MainWindow( "Movimiento manual del eje Z " );
  StatusLine( "Pulse cualquier tecla para regresar." );
  mensaje1();
  changetextstyle( TRIPLEX_FONT, HORIZ_DIR, 3 );
  settextjustify( CENTER_TEXT, CENTER_TEXT );
  setcolor(GREEN);
  outtextxy( 200, 130, " Utilice las flechas");
  setcolor(RED);
 }
/************************************************/
/* RUTINA DE INICIALIZACION DE LOS PUERTOS */
/************************************************/
void iniport(void)
{
/* PROGRAMACION DEL PUERTO P1A EN ENTRADA Y EL P1B EN SALIDA */
  outportb(regp1,0x99);
/* PROGRAMACION DEL PUERTO P2A EN SALIDA Y EL P2B EN ENTRADA */
  outportb(regp2,0x8B);
```

```
/* PR7OGRAMACION DEL CONTADOR C1A EN MODO 0 */
  outportb(regc1,0x30);
/* PROGRAMACION DEL CONTADOR C1B EN MODO 0 */
  outportb(regc1,0x70);
/* PROGRAMACION DEL CONTADOR C1C EN MODO 0 */
  outportb(regc1,0xB0);
/* PROGRAMACION DEL CONTADOR C2A EN MODO 0 */
  outportb(regc2,0x30);
/* PROGRAMACION DEL CONTADOR C2B EN MODO 0 */
   outportb(regc2,0x70);
};

/*********************************/
/*  INICIALIZA MODO GRÁFICO  */
/*********************************/
void Initialize(void)
{
 int xasp, yasp;
 GraphDriver = DETECT;
 initgraph( &GraphDriver, &GraphMode, "C:\\TC\\BGI" );
 ErrorCode = graphresult();  /* Lee resultados de inicialización/
 if( ErrorCode != grOk )
    {        /* Error ocurrido durante la inicialización */
  printf(" Error Gráfico de Sistema: %s\n", grapherrormsg( ErrorCode
```

```
outtextxy( 300, 50, "Colegio Salesiano");
outtextxy( 300, 85,"Domingo Savio" );
setcolor(GREEN);
outtextxy( 320, 145," Programa para el control del  " );
outtextxy( 300, 185," Taladro Electrónico " );
settextjustify( LEFT_TEXT, LEFT_TEXT );
outtextxy( 45, 250," ");
setcolor(GREEN);
outtextxy( 35, 375, "   ");
StatusLine( "Espere un momento." );
delay(1000);   /* Mantiene la pantalla de presentación por 1s. */
cleardevice(); /* Limpia la pantalla gráfica  */
}

/*********************************************/
/*  MAINWINDOW  */
/* Realiza una ventana principal alrededor de la pantalla */
/*********************************************/
void MainWindow( char *header )
{
 int height;
 cleardevice();          /* Limpia la pantalla gráfica */
 setcolor( MaxColors - 1 ); /* Configura como blanco el color actual */
 setviewport(0,0,MaxX,MaxY,1); /* Abre el puerto para toda la pantalla */
```

```
height = textheight( "H" );   /* Da la altura básica para el texto */
changetextstyle( DEFAULT_FONT, HORIZ_DIR, 1 );
settextjustify( CENTER_TEXT, TOP_TEXT );
outtextxy( MaxX/2, 2, header );
setviewport( 0, height+4, MaxX, MaxY-(height+4), 1 );
DrawBorder();
setviewport( 1, height+5, MaxX-1, MaxY-(height+5), 1 );
}

/************************************/
/*  STATUSLINE: */
/* Realiza una línea al final de la pantalla */
/************************************/
void StatusLine( char *msg )
{
 int height;
 setviewport(0,0,MaxX,MaxY,1); /* Abre el puerto para toda la pantalla */
 setcolor( MaxColors - 1 );   /* Configura como blanco el color actual */
 changetextstyle( DEFAULT_FONT, HORIZ_DIR, 1 );
 settextjustify( CENTER_TEXT, TOP_TEXT );
 setlinestyle( SOLID_LINE, 0, NORM_WIDTH );
 setfillstyle( EMPTY_FILL, 0 );
 height = textheight( "H" );  /* Determina la altura actual del texto */
 bar( 0, MaxY-(height+4), MaxX, MaxY );
```

```
  rectangle( 0, MaxY-(height+4), MaxX, MaxY );
  outtextxy( MaxX/2, MaxY-(height+2), msg );
  setviewport( 1, height+5, MaxX-1, MaxY-(height+5), 1 );
};

/*****************************************/
/*  DRAWBORDER: */
Dibuja una línea alrededor de la pantalla actual */
/*****************************************/
void DrawBorder(void)
{
 struct viewporttype vp;
 setcolor( MaxColors - 1 );   /* Configura como blanco el color actual */
 setlinestyle( SOLID_LINE, 0, NORM_WIDTH );
 getviewsettings( &vp );
 rectangle( 0, 0, vp.right-vp.left, vp.bottom-vp.top );
};

/*********************************************************/
/* CHANGETEXTSTYLE: igual a settextstyle, pero chequea errores */
/* ocurridos cargando el Archivo que contiene las fuentes de letras
/*********************************************************/
void changetextstyle(int font, int direction, int charsize)
{
```

```
) );
   exit( 1 );
  }
  getpalette( &palette );          /* Lee la paleta desde el board     */
  MaxColors = getmaxcolor() + 1;   /* Lee el máximo numero de colores*/
  MaxX = getmaxx();
  MaxY = getmaxy();                /* Lee la talla de la pantalla */
  getaspectratio( &xasp, &yasp );  /* Lee el aspecto del hardware */
  AspectRatio = (double)xasp / (double)yasp; /* Da el factor de corrección */
}
/************************************/
/*   PANTALLA DE PRESENTACIÓN  */
/************************************/
void Hola(void)
{
 struct viewporttype viewinfo;    /* Estructura para leer viewport */
 int h,w;
 MainWindow( "Taladro electrónico controlado por computadora." );
 setcolor(RED);
 getviewsettings( &viewinfo );    /* Lee configuración del viewport */
 changetextstyle( TRIPLEX_FONT, HORIZ_DIR, 4 );
 settextjustify( CENTER_TEXT, CENTER_TEXT );
                                  /* Comienza a escribir lo que aparecerá */
                                  /* en la pantalla de presentación */
```

```
int ErrorCode;
graphresult();              /* clear error code     */
settextstyle(font, direction, charsize);
ErrorCode = graphresult();      /* check result        */
if( ErrorCode != grOk ){        /* if error occured    */
  closegraph();
  printf(" Graphics System Error: %s\n", grapherrormsg( ErrorCode) );
  exit( 1 );
 }
}

/*************************/
/*  MENU DE SELECCION  */
/*************************/
int Menu(void)
{
 struct viewporttype viewinfo;  /* Estructura para leer el viewport */
 int h, w;
 int opcion=0;
 MainWindow( "Menu de selección." );
 getviewsettings( &viewinfo );  /* Lee la configuración del viewport */
 changetextstyle( TRIPLEX_FONT, HORIZ_DIR, 3 );
 settextjustify( LEFT_TEXT, LEFT_TEXT );
 h = viewinfo.bottom - viewinfo.top;
```

```
 w = viewinfo.right  - viewinfo.left;
 setcolor(RED);          /* Cambia el color */
 outtextxy( w/2-90, h/2-180, "<< D. SAVIO >>" ); /* comienza escribir en */
 outtextxy( w/2-200, h/2-130, "Programa  ");    /* la pantalla del Menú */
 outtextxy( w/2-200, h/2-100, "  Taladro .");
 setcolor(GREEN);
 outtextxy( w/2-250, h/2-40, "1. Graba perforaciones de otra tarjeta." );
 outtextxy( w/2-250, h/2, "2. Captura archivo del programa Tango." );
 outtextxy( w/2-250, h/2+40, "3. Perforación de la tarjeta." );
 outtextxy( w/2-250, h/2+80, "4. Salir del programa." );
 StatusLine( "Elija la opcion adecuada." );
 opcion=bioskey(0) & 0x00ff;
 cleardevice();       /* Limpia la pantalla gráfica */
 return(opcion);
}
/*****************************/
/* ESCRIBE EN MODO TEXTO */
/* Permite escribir en modo texto */
/*****************************/
void gprintxy(int posx,int posy,char *formato,...)
{
  va_list argptr;
 char cadena[140];
 struct textsettingstype infotexto;
```

```
  va_start(argptr,formato);
  vsprintf(cadena,formato,argptr);
  gettextsettings(&infotexto);
  outtextxy(posx,posy,cadena);
  va_end( argptr );
}
/*****************************************************/
/* RUTINA PARA EL MOVIMIENTO DE LOS EJES X-Y */
/*****************************************************/
void movimiento_xy(int outc1a, int adx, int outc2a, int ady)
 {          /* Definición de variables locales */
  int endx, endy, outc2b, outc1b, outp2c;
  unsigned char lsb, msb;
  outp2a=0;
  outc2b=0; outc1b=0;
  outp2c=0;
  marcha_paro(0,0);  /* Baja la corriente en ambos ejes X y Y */
  outp2a=(outp2a & 247);
  outportb(port2a,outp2a);
  velocidad=10 ;       /* VELOCIDAD DEL EJE Y  - VEL - ENTRE 0 Y 15 */
  outportb(port1b,velocidad);
  outp2a = ( outp2a & 223) | ( ady * 32);  /* SALIDA ADELANTE ATRAS */
  outp2a = ( outp2a & 251) | (adx * 4);
  outportb(port2a,outp2a);
```

```
lsb = outc2a & 255;   /* Carga contador C2A = T-A - OUTC2A -ENTRE 0 Y 65536 */
msb = ((outc2a | 255)-255)/256;
outportb (count2a,lsb);    /* CARGA EL VALOR DE CONTEO EN C2A ; LSB */
outportb (count2a,msb);    /* CARGA EL VALOR DE CONTEO EN C2A ; MSB */
lsb = outc1a & 255;  /* Carga contador C1A = T-A - OUTC1A - ENTRE 0 Y 65536 */
msb = ((outc1a | 255)-255)/256;
outportb (count1a,lsb);    /* CARGA EL VALOR DE CONTEO EN C1A ; LSB */
outportb (count1a,msb);    /* CARGA EL VALOR DE CONTEO EN C1A ; MSB */
                /* CARGA CONTADOR C2B = T -OUTC2B - ENTRE 0 Y 65536 */
outc2b = 0.8 * outc2a;
lsb = outc2b & 255;
msb = ((outc2b | 255)-255)/256;
outportb(count2b,lsb);
outportb(count2b,msb);
                    /* CARGA CONTADOR C1B = T -OUTC1B - ENTRE 0 Y 65536 */
outc1b = 0.8 * outc1a;
lsb = outc1b & 255;
msb = ((outc1b | 255)-255)/256;
outportb(count1b,lsb);
outportb(count1b,msb);
marcha_paro(1,1);          /* Sube la Corriente en ambos ejes X y Y */
outp2a=(outp2a & 246) | 9;  /* SUBE BIT DE START */
outportb(port2a,outp2a);
endy=inportb(port2b) & 8;   /* ESPERA EL BIT DE STOP */
```

```
endx=inportb(port2b) & 1;
while (endy == 0 || endx == 0)
{
  endy=inportb(port2b) & 8;
  endx=inportb(port2b) & 1;
  Chequea_micros();            /* CONTROLA LOS TOPES */
};
outp2a=(outp2a & 246);      /* BAJA EL BIT DE START */
outportb(port2a,outp2a);
marcha_paro(0,0);           /* Baja la Corriente en ambos ejes X y Y */
};
/*************************************************/
/*  RUTINA PARA EL MOVIMIENTO DEL EJE Z */
/*************************************************/
void movimiento_z(int pul, int ar)
{
  int p2a,endz;
  unsigned char lsb,msb;
  p2a=0;
  outportb(port2a,p2a);    /* PONE BAJO LAS SALIDAS DEL PUERTO P2A */
  lsb=pul & 255;           /* VALOR DE CONTEO PUL DEL EJE Z ENTRE 0 Y 65535 */
  msb= ((pul | 255)-255)/256;
  p2a=((p2a & 127) | (ar*128)); /* INDICA SI SUBE O BAJA EL TALADRO */
  outportb(port2a,p2a);
```

```
outportb(count1c,lsb);      /* CARGA EL VALOR DE CONTEO EN 1C (LSB) */
outportb(count1c,msb);      /* CARGA EL VALOR DE CONTEO EN 1C (MSB) */
p2a=p2a | 64;               /* PONE EN ALTO START */
outportb(port2a,p2a);
endz=inportb(port2b) & 4;   /* ESPERA EL BIT DE STOP */
while (endz == 4)
 {
   endz=inportb(port2b) & 4;
   Chequea_micros();        /* CONTROLA LOS TOPES */
 }
 p2a=(p2a & 63);            /* BAJA EL BIT DE START */
 outportb(port2a,outp2a);
}
/*********************************************/
/* MARCHA Y PARO  */
/* Controla la Corriente alta o baja para los ejes X y Y */
/*********************************************/
void marcha_paro(int max, int may)
{
  outp2a=(outp2a & 239) | (may * 16);
  outp2a=(outp2a & 253) | (max * 2);
  outportb(port2a,outp2a);
}
```

```
/***************/
/* FIN DE EJE  */
/***************/
void Chequea_micros(void)
{
 unsigned char byte;
 byte=inportb(port1a) & 63;
 ftx=0; itx=0; fty=0; ity=0; ftz=0; itz=0;
 if ((byte & 1) == 1)  /* Chequea si alguno de los 6 micros correspondiente */
 {                     /* a los 3 ejes se encuentra activado  */
  ftx=1;
 };
 if ((byte & 2) == 2)
 {
  itx=1;
 } ;
 if ((byte & 4) == 4)
 {
  fty=1;
 };
 if ((byte & 8) == 8)
 {
  ity=1;
 };
```

```
  if ((byte & 16) == 16)
 {
  ftz=1;
 };
 if ((byte & 32) == 32)
 {
  itz=1;
 };
};
```

APENDICE B

PRESUPUESTO DEL PROYECTO

Listado de componentes con sus respectivos precios.

Componentes digitales

Cantidad	descripción	Precio unitario	Precio total
10	74LS00	0.65	6.50
4	74LS04	0.65	2.60
2	74LS08	0.65	1.30
1	74LS11	0.68	0.68
3	74LS14	0.65	1.95
2	74LS32	0.69	1.38
1	74LS138	0.75	0.75
2	74LS175	0.75	1.50
1	74LS191	0.84	0.84
2	74LS221	0.80	1.60
3	74LS244	0.82	2.46
2	L297	2.00	4.00
7	4016	1.2	8.4
2	4046	1.2	2.4

Componentes analógicos

Cantidad	descripción	Precio unitario	Precio total
3	555	0.35	1.05
12	LM741	0.40	4.80
6	ECG 3100	3.20	19.2

Reguladores de voltaje

Cantidad	descripción	Precio unitario	Precio total
1	LM7805	0.65	0.65
1	LM7808	0.65	0.65
2	LM7812	0.70	1.40
1	LM7912	0.70	0.70

Componentes de potencia.

Cantidad	descripción	Precio unitario	Precio total
16	Mosfet IRF521	8.00	128.00

Transistores

Cantidad	descripción	Precio unitario	Precio total
20	BC237	0.20	4.00
3	DD135	0.20	0.60
4	2N3906	0.10	0.40
12	2N3904	0.10	1.20

Rele

Cantidad	descripción	Precio unitario	Precio total
3	Relé de 24v.	3.00	9.00

Condensadores

Cantidad	descripción	Precio unitario	Precio total
10	22nf 50v.	0.15	1.50
6	0.47uf 100v.	0.15	0.90
10	1uf 100v.	0.15	1.50
2	4.7uf 25v.	0.15	0.30
8	10 uf 50v.	0.15	1.20
2	47uf 50v.	0.40	0.80
2	200uf 50v.	0.30	0.60

Diodos

Cantidad	descripción	Precio unitario	Precio total
20	1N4007	0.15	3.00
20	5406	0.15	3.00
14	LED	0.15	2.10
5	Zener 5v.	0.20	1.00

Rectificadores

Cantidad	descripción	Precio unitario	Precio total
6	200v. 2A.	0.40	2.40

Resistencias

Cantidad	descripción	Precio unitario	Precio total
150	1/2 W.	0.05	7.50

Potenciometros

Cantidad	descripción	Precio unitario	Precio total
2	50K.	0.70	1.40
8	10K	0.70	5.60
4	1K.	0.70	2.80

Componentes varios

Cantidad	descripción	Precio unitario	Precio total
1	Transformador	45.00	45.00
4	Pulsadores NA	0.50	2.00
9	Interruptores	0.70	6.30
4	Borneras	0.70	2.80
25	Disipadores	0.90	22.5
12	Fusibles de 2A.	0.20	2.40
2	Fusibles de 3A.	0.15	0.30
14	Porta fusibles	0.60	8.40
7	Conectores de 4 ter.	1.00	7.00
5	Conectores DB9	0.90	4.50
2	Ventilador de 12v.	5.00	10.00
1	Enchufe para 110v.	0.65	0.65
20	Mts. de cable # 14	0.10	2.00
16	Mts. de cable # 20	0.10	1.60
10m.	Alambres multipar	0.40	4.00
10	Mts.espagueti metál.	1.00	10.00
1	Mueble de metal	50.00	50.00
1	Mesa de perforación	80.00	80.00

1	Computador 386	400	400
1	Tarjeta de I.8255	60.00	60.00
10	Tarjetas electrónicas		50.00
2	Motores de Paso de 4.6A.	70.00	140.00
1	Motor de paso de 1.5A.	45.00	45.00
1	Motor DC de 1/8 HP.	25.00	25.00
4	Ejes de acero	25.00	100.00
4	Tornillos 1.2m paso 4	30.00	120.00
2	Cajas para reducir el juego mecánico	10.00	20.00

TOTAL 1463.00

CONCLUSIONES

El Software que utiliza el controlador, es un programa elaborado completamente en Lenguaje C desarrollado exclusivamente por el diseñador de la máquina.

Las tarjetas electrónicas que posee el controlador fueron realizadas completamente en los Laboratorios del Colegio Domingo Savio, a excepción de la tarjeta de interfase que se encuentra montada en uno de los slot de la computadora.

El Software que posee la máquina de perforación fue diseñado para que cualquier usuario con pocas nociones de computación la pueda utilizar.

Esta máquina también puede ser instalada en cualquier empresa que se dedique a la fabricación de Circuitos Impresos para acelerar los procesos de producción y fabricación.

El controlador tan sólo con unos pequeños cambios en el Software es capaz de capturar datos de otros programas como el AUTOCAD para ejecutar dibujos y aplicarlos en la utilización de esta máquina.

El diseño del controlador puede servir a futuros alumnos que estén interesados en controlar motores de paso.

RECOMENDACIONES

Se recomienda seguir los siguientes pasos para poner en marcha a la máquina:

- Encender la computadora
- Entrar en el programa que controla el proceso
- Encender el interruptor principal de la máquina
- Encender cada uno de los dos ejes X y Y
- Proceder a trabajar como se especificó en el Manual del Usuario (Capítulo 6)

En caso de llegar a uno de los límites se pueden hacer dos cosas:

- Apagar la máquina y manualmente sacarla de esa posición
- Salir del programa de perforación y mediante el teclado direccional sacarlo de esa posición.

En la medida de lo posible se recomienda tener a la máquina en un lugar fresco y bien ventilado.

Verificar que se encuentren encendidos los ventiladores encargados de enfriar los Mosfet de potencia.

Si por alguna causa no deseada se apaga uno de los ejes (existe protección para sobre corriente) es recomendable que se comience nuevamente desde el principio el proceso.

Es necesario que los ejes X, Y y Z , y todas las partes móviles se encuentren siempre bien lubricados, para garantizar una vida útil larga de las partes.

Bibliografía:

1. Folleto de Guía de Laboratorio
   De lorenzo, Control de Motores de paso, pp 1-25 29-32

2. Manual de Integrados de la SGS Thompson pp 70 - 102

3. Ronald J. Tocci Sistemas Digitales (5ta edición México 1993) pp 48 - 62

4. Robert Lafore y Grupo Waite , Programación en Microsoft "C" pp 27 – 300