34
2es.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ingeniería

Diseño del Control Numérico (C.N.C) de una Máquina de Corte por Descargas Eléctricas (E.D.M)

T E S I S

Que para obtener el título de:

INGENIERO MECANICO ELECTRICISTA

Area Eléctrica Electrónica

p r e s e n t a n

EDUARDO MARIO CARREON MATA
MAURICIO FABIAN SANCHEZ RESENDIZ
RICARDO VILLANUEVA HERNANDEZ

Director: Ing. Miguel Angel Cruz León

Ciudad Universitaria 1998

AGRADECIMIENTOS

A mi esposa Lourdes e hijos Abraham e Isaac, por la paciencia que tuvieron y las carencias que pasaron en el periodo de desarrollo de este trabajo.

Al Ingeniero Miguel Angel Cruz León y a su esposa María Mercedes Bolaños por su tiempo y dedicación, así como por la paciencia que nos brindaron, que más que asesores de tesis los considero mis amigos.

A mi familia, padres y hermanos, con múltiples diferencias somos eso.

A la UNAM por brindarme la oportunidad de estudiar.

Eduardo Mario Carreón Mata

A mis padres por su cariño y amor, mi más profundo agradecimiento.

A todas las personas que me ayudaron para concluir esta tesis.

Mauricio Sanchez Reséndiz

A mis padres Mardonia y Alfredo, por la confianza y el apoyo que me dieron durante mis estudios, y en especial por su cariño.

A mis amigos y compañeros, y a Edmundo por su amistad.

A la U.N.A.M. y a la Facultad de Ingeniería por la formación académica que me proporcionaron.

Al Ing. Miguel Angel Cruz León y a su esposa, cuyo asesoramiento nos ha permitido el desarrollo del presente trabajo.

A Ninfa Gloria por su apoyo y cariño, y por todo lo que representa en mi vida.

Ricardo Villanueva Hernández

DISEÑO DEL CONTROL NÚMERICO (C.N.C.) DE UNA MÁQUINA DE CORTE POR DESCARGAS ELÉCTRICAS (E.D.M.)

Índice General

INTRODUCCIÓN.....1

1 PRINCIPIOS DE OPERACIÓN DE LAS MÁQUINAS DE CORTE.....2

1.1 ANTECEDENTES.....2
1.2 COMPARACIÓN DE MAQUINADO POR DESCARGAS ELÉCTRICAS Y LOS EFECTOS DE UNA TORMENTA ELÉCTRICA.....2
1.3 MAQUINADO POR DESCARGAS ELÉCTRICAS (E.D.M.).....3
1.3.1 DEFINICIÓN DEL PROCESO.....3
1.4 PRINCIPIO FÍSICO DE E.D.M......5
1.5 TASA DE DESGASTE DEL ELECTRODO.....7
1.6 DESCRIPCIÓN DEL SISTEMA DE CONTROL (CNC) DE UNA MÁQUINA DE CORTE POR MEDIO DE DESCARGAS ELÉCTRICAS (E.D.M).....8

2 DESCRIPCIÓN DE CONTROLADORES PARA MOTORES DE CD Y MOTORES DE PASOS.....11

2.1 INTRODUCCIÓN.....11
2.2 EL MOTOR DE PASOS.....12
2.3 PRINCIPIOS DEL MOTOR DE DC.....16
2.4 VARIACIÓN DE VELOCIDAD DE UN MOTOR SHUNT DC.....18
2.5 CONTROL DE VOLTAJE Y CORRIENTE DE ARMADURA USANDO TIRISTORES.....19
2.6 SISTEMA DE CONTROL DE VELOCIDAD MONOFÁSICO Y DE MEDIA ONDA PARA UN MOTOR SHUNT DC.....20
2.7 CONTROL REVERSIBLE DE VELOCIDAD.....21
2.8 MODULACIÓN POR ANCHO DE PULSO (PWM).....23

3 OPERACIÓN DE CODIFICADORES DE DESPLAZAMIENTO (ENCODERS).....25

3.1 APLICACIÓN DE LOS ENCODER.....26
3.2 CONTINUACIÓN APLICACIONES TÍPICAS.....27
3.3 PRINCIPIOS DE SENSADO.....29

3.4 SENSORES MAGNÉTICOS.....29
3.5 SELECCIÓN ESTANDAR DE SALIDA.....37

4 MANEJO DE SEÑALES DE ERROR Y LÍMITE DE POSICIÓN.....39

4.1 PRINCIPIOS DE OPERACIÓN DE LOS INTERRUPTORES DE LÍMITE DE POSICIÓN.....39
4.2 INTERRUPTORES DE LÍMITE DE POSICIÓN INDUCTIVOS.....39
4.3 INTERRUPTORES DE LÍMITE DE POSICIÓN CAPACITIVOS.....40
4.4 SELECCIÓN DE INTERRUPTORES DE LÍMITE DE POSICION.....40
4.5 CONSIDERACIONES ADICIONALES.....41
4.6 SENSORES DE ULTRA PRECISIÓN.....43

5 SELECCIÓN DE MICROCONTROLADORES.....44

5.1 MICROCONTROLADOR MC68HC11.....45
5.2 CARACTERISTICAS DEL MICROCONTROLADOR MC68HC11.....46

6 CONVERTIDOR D/A.....52

7 DISEÑO Y CONSTRUCCIÓN DE LA INTERFACE CON LA PC.....58

7.1 SEÑALES A USAR.....58
7.2 ASIGNACIÓN DE DIRECCIONES DE PUERTOS.....59
7.3 DECODIFICACIÓN DE DIRECCIONES.....60
7.4 DISPOSITIVOS DE ENTRADA Y SALIDA.....60

8 DESARROLLO DEL PROGRAMA DEL CONTROL 63

8.1 DESCRIPCIÓN DEL CONTROL.....63
8.2 DESCRIPCIÓN DEL PROGRAMA.....65
8.3 ENTRADA AL PROGRAMA.....66
8.4 DESCRIPCIÓN DE PANTALLAS.....66
8.5 DISEÑO DEL SISTEMA DE PROGRAMACIÓN.....70
8.6 CONTROL NUMÉRICO COMPUTARIZADO (CNC).....70
8.7 ELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN.....73
8.8 ACCESO AL BUS DE DATOS Y MANEJO DE INFORMACIÓN.....74
8.9 ESCRITURA A UN PUERTO FÍSICO.....74
8.10 LECTURA DE UN PUERTO FÍSICO.....75
8.11 INTERFACES DE CONTROL.....75

8.12 DESARROLLO DE PROGRAMAS DEL PROTOCOLO DE COMUNICACIÓN.....75
8.12.1 PROGRAMAS PARA COMUNICAR LA PC CON EL MICROCONTROLADOR.....75
8.12.2 SUBRUTINAS DE TIEMPO Y LECTURA DE PUERTO PARA EL PROGRAMA DE COMUNICACIÓN DE LA COMPUTADORA PERSONAL (PC) AL MICROCONTROLADOR 80
8.12.3 PROGRAMA PARA ESTABLECER LA COMUNICACIÓN DEL MICROCONTROLADOR A LA COMPUTADORA PERSONAL (PC).....81
8.12.4 SUBRUTINA DE TIEMPO 2 PARA EL PROGRAMA DE COMUNICACIÓN DEL MICROCONTROLADOR A LA COMPUTADORA PERSONAL (PC).....86
8.13 DESARROLLO DE PROGRAMAS DE DETECCIÓN DE POSICIÓN.....87
8.13.1 PROCEDIMIENTO PARA MEDIR LA LONGITUD DE MATERIAL CORTADO.....87
8.13.2 PROCEDIMIENTO PARA COLOCAR EL CABEZAL DE LA MÁQUINA DE CORTE EN EL ORIGEN.....92
8.13.3 PROGRAMA PARA DETECTAR LA ACTIVACIÓN DE INTERRUPTORES (ALARMAS)CUANDO SE ENCUENTRA OPERANDO LA MÁQUINA.....96
8.14 DESARROLLO DE PROGRAMA DE DETECCIÓN DE VELOCIDAD.....100

9 RESULTADOS Y CONCLUSIONES.....105

APÉNDICE A. DIAGRAMAS DE FLUJO DEL PROGRAMA EN C.....110

APÉNDICE B. DIAGRAMA DE SEÑALES DEL PROTOCOLO DE COMUNICACIÓN.....126

APÉNDICE C. PROGRAMAS DEL MICROCONTROLADOR.....128

APÉNDICE D. PROGRAMA EN LENGUAJE C.....143

APÉNDICE E. DIAGRAMA DE LA INTERFAZ PC-MCU.....178

BIBLIOGRAFÍA.....180

INTRODUCCIÓN

Dentro de la industria metal - mecánica se han desarrollado diversas tecnologías para el troquelado y corte de materiales. Incluso la aparición de nuevas herramientas o el mejoramiento de las herramientas convencionales ha permitido mejorar los procesos de corte.

Uno de los procesos que se ha perfeccionado con el paso del tiempo es el de corte de materiales (metálicos), a través del método de corte por descargas eléctricas (E.D.M. por sus siglas en inglés), el cual permite realizar cortes de piezas de muy pequeñas dimensiones o el troquelado fino de moldes que con el uso de herramientas convencionales sería prácticamente imposible conseguir, por lo que se justifica el desarrollo de un control numérico para este tipo de máquinas, lo cual es el tema central del presente trabajo.

En la actualidad las tecnologías existentes en este campo han sido desarrolladas y comercializadas por empresas japonesas, taiwanesas, norteamericanas y suizas, sin embargo los costos de inversión para sistemas nuevos de este tipo alcanzan varios miles de dólares.

Existen en México una gran cantidad de equipos de corte E.D.M. que no están siendo utilizados debido a los grandes costos involucrados en la puesta en marcha de los mismos. La adquisición de un sistema de control numérico que permita automatizar los procesos de corte forza al interesado a recurrir al extranjero para su disponibilidad, con los consiguientes gastos y la falta de soporte local en caso de requerirse.

El presente trabajo tiene por finalidad el diseño del control numérico (CNC) que permita la operación de una máquina de corte por descargas eléctricas (EDM) y que sirva como alternativa para los poseedores de equipos actualmente desaprovechados.

Para tener una apreciación más detallada de los elementos que están involucrados en el presente trabajo de investigación, se pretende mencionar los antecedentes de esta tecnología, una descripción de controladores para motores de Corriente Directa (CD) y motores de pasos, operación de los codificadores de desplazamiento (Encoder), manejo de señales de error y límite de posición, descripción del microcontrolador utilizado (68HC11) y convertidores D/A.

Posteriormente se presentará el diseño de la interface que será utilizada para la comunicación entre el microcontrolador y la computadora (PC), así como el desarrollo de los programas que controlarán este proceso.

Capítulo 1

PRINCIPIOS DE OPERACIÓN DE LAS MÁQUINAS DE CORTE

1.1 ANTECEDENTES

Las primeras investigaciones referentes a los efectos de la erosión causada por descargas eléctricas fueron realizadas alrededor de 1770 por el científico de origen inglés Priestley. En épocas más recientes, en 1943, los científicos soviéticos B.R. y N.I. Lazarenko decidieron aprovechar el efecto de las descargas eléctricas para desarrollar un método controlado para el maquinado de metales.

El principio de descarga utilizado entonces, y conocido como circuito Lazarenko, ha sido tomado como modelo y mejorado, se utiliza hasta la fecha en fuentes de poder para máquinas E.D.M. (Maquinado por Descargas Eléctricas).

1.2 COMPARACIÓN DE MAQUINADO POR DESCARGAS ELÉCTRICAS Y LOS EFECTOS DE UNA TORMENTA ELÉCTRICA

En una tormenta, el arco de corriente eléctrica se establece entre los objetos mas altos o más cercanos a las nubes, como se ve en la figura 1. La descarga provoca una onda de presión de aire y la luminosidad es la energía eléctrica la cual realiza el trabajo, como por ejemplo, resquebrajar un árbol. En descargas E.D.M., ésta forma un arco eléctrico desde el electrodo hasta el punto más cercano sobre la pieza de trabajo; la chispa crea altas temperaturas (8000°C - 12000°C) en el punto de impacto ocasionando que el material se vaporice. Parte del dieléctrico se vaporiza también creando una onda de choque con alta presión, la cual forza al material vaporizado a esparcirse.

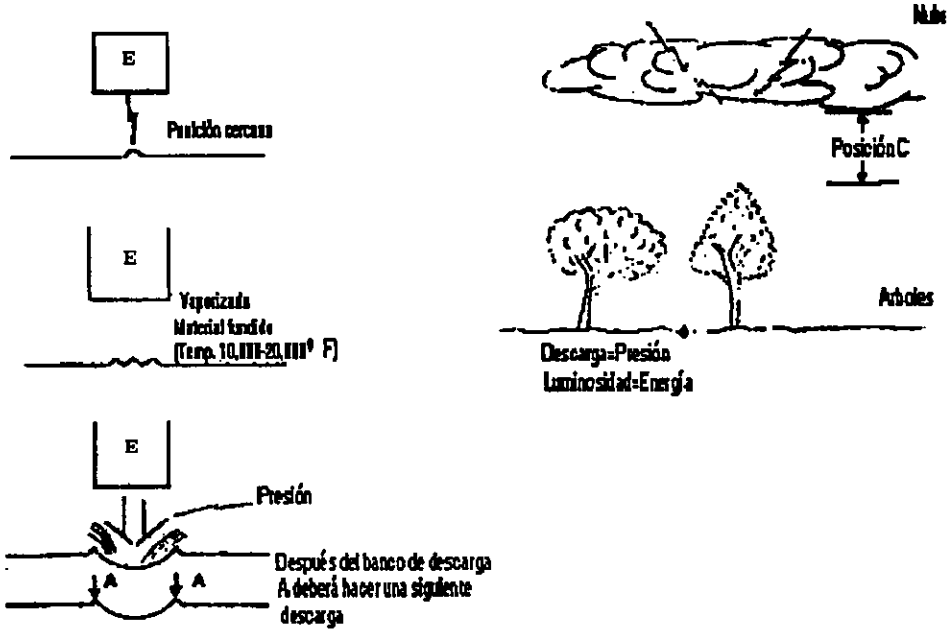


Fig. 1 Comparación de maquinado por descargas eléctricas y tormenta eléctrica

1.3 MAQUINADO POR DESCARGAS ELÉCTRICAS (E.D.M.)

1.3.1 DEFINICIÓN DEL PROCESO

E.D.M. se caracteriza por la eliminación o remoción de material producido por una serie de descargas eléctricas no estacionarias espaciadas en el tiempo realizadas entre un electrodo-herramienta y un electrodo-pieza en proceso, es decir, solamente ocurre una descarga al mismo tiempo y estas deben realizarse a través de un fluido no conductor.

El proceso de E.D.M. tiene dos aplicaciones fundamentales:

- Para maquinar materiales duros, los cuales no son fácilmente maquinables por métodos convencionales
- Para reproducir automáticamente cualquier forma o secuencia de corte.

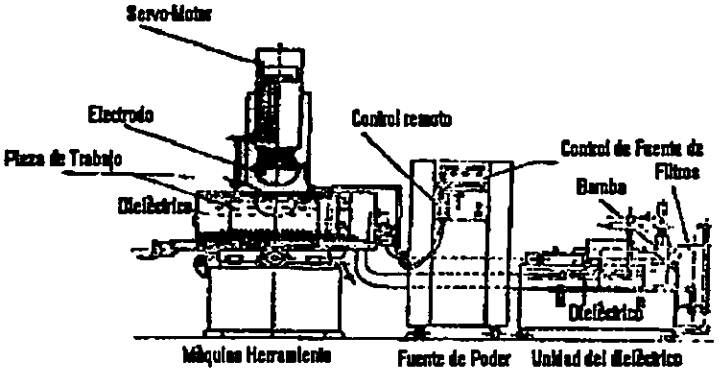


Fig. 2 Descripción de máquina E.D.M.

El esquema básico de una máquina de E.D.M. se ilustra en la figura 2. Una de las ventajas de utilizar este método es que los metales pueden ser removidos sin importar la dureza. Cuando la dureza del material se incrementa, el tiempo que toma el corte del material aumenta debido a la dificultad ocasionada por la dureza hasta llegar a un punto en donde la dureza es tal que es prácticamente imposible el corte con herramientas convencionales.

Generalmente el método E.D.M. se utiliza para el maquinado de materiales exóticos como carburo de tungsteno, acero, etc., materiales con tratamiento térmico, maquinado en ambas partes del molde con cavidades de tres dimensiones y dados de forja, y para la remoción de formas irregulares y barrenos. Adicionalmente materiales como el acero inoxidable y metales brillosos son difíciles de maquinar con herramientas convencionales. Un ejemplo de este tipo de cortes es mostrado en la figura 3.2.

Para maquinados de moldes complicados o de tres cavidades, el uso de herramientas convencionales no aseguran obtener la mayor precisión. Cuando se requiere de acabados finos, la E.D.M. debe ser empleada, debido a que la precisión del maquinado E.D.M. se encuentra en el rango de +/- 0.005 mm a +/-0.01 mm aunque depende de las condiciones de maquinado y de la destreza del operador. En la figura 3.1 se muestra una máquina de E.D.M. moderna.

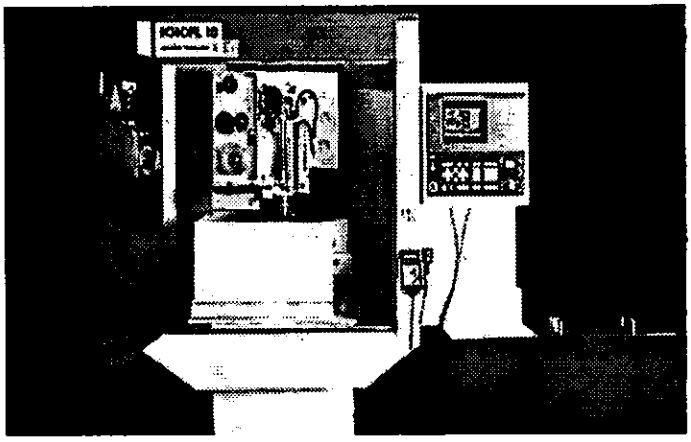


Fig. 3.1 Máquina de corte E.D.M.

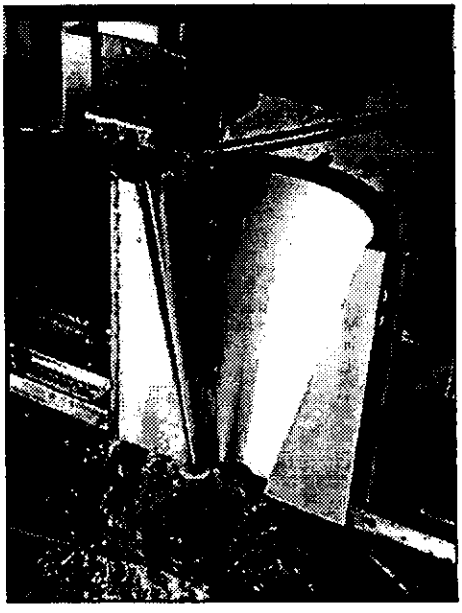


Fig. 3.2. Ejemplo de corte de material metálico con maquinado E.D.M.

1.4 PRINCIPIO FÍSICO DE E.D.M.

En E.D.M. se requiere de electrodo y una pieza de trabajo sumergida en un fluido no conductor (dieléctrico). Para generar la circulación de corriente entre las dos partes, es necesario de un voltaje más alto que el voltaje de ruptura a la distancia que se aplique. Este voltaje de ruptura depende de la distancia entre los electrodos en su punto más cercano, las características aislantes del fluido dieléctrico y al nivel de contaminación en el área de descargas.

En el punto entre el electrodo y la pieza donde el campo eléctrico es más fuerte ocurre una descarga de la siguiente manera:

- Bajo el efecto del campo eléctrico los iones positivos libres y los electrones son acelerados formando ellos un canal ionizado eléctricamente conductor.
- Durante este período la corriente puede pasar a través del canal, se inicia una chispa entre los electrodos y se forma un plasma.
- El plasma alcanza una temperatura entre 8000° C y 12000° C expandiéndose bajo el efecto de numerosos impactos de partículas cargadas causando instantáneamente la fundición local del material en ambas superficies de los conductores.
- Simultáneamente y debido a la vaporización de los electrodos y del fluido eléctrico, una burbuja de gas se expande e incrementa su presión. Al momento de que la corriente es cerrada, la burbuja de gas se desprende por la caída repentina de la temperatura generando fuerzas dinámicas que resultan en remoción de material que se extrae del cráter.
- El material fundido es resolidificado en el fluido dieléctrico como pequeñas esferas y evacuadas por el mismo fluido.

La erosión de los electrodos es asimétrica, y esto depende de la polaridad, conductividad térmica del material, punto de fusión, duración e intensidad de las descargas. La erosión es conocida como desgaste cuando ocurre en la herramienta electrodo y como material removido en la pieza de trabajo. En las figuras 4a y 4b se muestran los diferentes tipos de maquinado.

Seleccionando adecuadamente el material del electrodo y controlando las descargas, variando su duración, intensidad y polaridad se obtiene una asimetría significativa, por ejemplo 99.5% de erosión en la pieza en proceso y 0.5% en la herramienta electrodo.

Fig. 4.a Principios de corte por E.D.M.

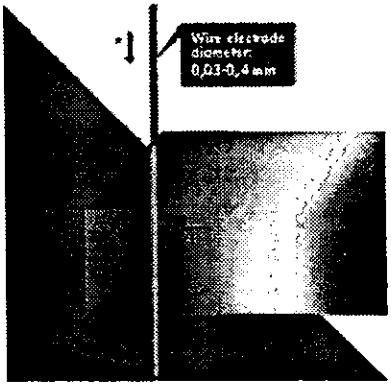


Fig. 4.b Diferentes tipos de maquinado por E.D.M.

1.5 TASA DE DESGASTE DEL ELECTRODO

La tasa de desgaste del electrodo se define como la cantidad de desgaste del electrodo en relación con el material removido expresada en porciento:

$$\text{Tasa de Desgaste de Electrodo (\%)} = \frac{\text{Cantidad de Desgaste del Electrodo}}{\text{Cantidad de Material Removido}} \times 100$$

Esta expresión puede ser definida de tres formas diferentes dependiendo de las mediciones que se hagan en peso, volumen o largo:

- La Tasa de Desgaste de Electrodo en peso se obtiene pesando las piezas antes y después del trabajo para poder medir la pérdida de peso; tiene la ventaja de ser muy fácil pero con la desventaja de que los materiales que se usan tanto de electrodos como de pieza a trabajar tienen densidades tan distintas que no hacen posible la comparación entre dos combinaciones Electrodo - Material - Fuente diferentes.
- El cálculo del volumen desgastado tanto de electrodo como de material es muy difícil de calcular directamente, por lo que conviene definirlo en función del desgaste en peso y de las densidades respectivas de la siguiente forma:

$$\text{Tasa de Desgaste en Volumen} = \text{Tasa de Desgaste en peso} \times \frac{\text{Peso específico de P.T}}{\text{Peso esp. del electrodo}}$$

- La tasa de desgaste en longitud está dada por la razón de la profundidad de la cavidad maquinada a la longitud del desgaste del electrodo y se define como:

$$A(\%) = \frac{B - C}{C} \times 100$$

Donde:

B = lectura en la máquina del largo total que el electrodo penetró

C = profundidad medida del maquinado

1.6. DESCRIPCIÓN DEL SISTEMA DE CONTROL NÚMERICO DE UNA MÁQUINA DE CORTE POR MEDIO DE DESCARGAS ELÉCTRICAS (E.D.M).

La presente investigación parte de la necesidad de diseñar un sistema de control numérico eficaz, el cual sea capaz de controlar el funcionamiento de una máquina EDM en dos planos, y que este mismo sistema pueda ser adaptable a otro tipo de máquinas herramientas, que desarrollen su trabajo en dos ejes.

El sistema completo que se tomó en consideración para el presente trabajo de investigación se compone de:

1.- Máquina de corte EDM, la cual a su vez contiene los siguientes elementos

a) Dos motores de desplazamiento para los ejes X e Y.
b) Un motor para la recolección del hilo utilizado en el corte.
c) Dos Encoder acoplados a los ejes de los motores para la medición y control de la velocidad y el desplazamiento.
d) Dos servos que controlan el arranque, la velocidad y el paro de los motores para cada uno de los ejes.
e) Mesa de trabajo en la que se colocará el material a cortar.
f) Sensores de límite de posición.
g) Sensores de alarma (Hilo roto, falta de agua)

2.- Microcomputadora PC o compatible

3.- Tarjeta de Interfaz para comunicar la máquina de corte EDM con la Microcomputadora. Esta tarjeta a su vez contiene el diseño electrónico de los circuitos de los microcontroladores, de los componentes de entrada y salida de datos (buffers y latches) y del convertidor digital-analógico.

4.- Programación de la interfaz gráfica, desarrollado en lenguaje de programación "C"

5.- Programa de Control Numérico (CNC) para los microcontroladores.

A continuación se describe a modo general la operación completa del sistema CNC, tomando en cuenta que para esta explicación se considera que las conexiones físicas entre los diversos componentes, así como la energización o encendido de los mismos ya se ha llevado a cabo previamente.

Al momento del encendido el programa de control CNC realizará las siguientes funciones de verificación o inicialización: estado de los sensores de límite de posición. Se utilizarán sensores de límite de posición mecánicos los cuales estarán situados en cada uno de los cuadrantes imaginarios del área máxima de trabajo, así como en la parte central de la misma. Estos sensores serán habilitados cuando el cabezal alcance alguno de los límites fijados, enviando una señal al microprocesador para que se alerte sobre la posición del corte o para indicarle que se encuentra fuera de rango y detener el proceso. También, se utilizarán sensores mecánicos para vigilar la

existencia del hilo de corte y para la ausencia o presencia del fluido no conductor, que para el caso que se presenta, es agua destilada.

Estos sensores solamente se habilitarán ante un error, esto es, mientras el equipo se encuentre operando normalmente estarán cerrados, y solamente ante la presencia de un error, abrirán el circuito indicando al microprocesador que detenga el proceso de corte y envíe la señal de alarma a la computadora.

Una vez verificadas las condiciones de las alarmas, los microcontroladores determinarán si es necesario realizar un breve desplazamiento de los motores de cada uno de los ejes (en caso de que se encuentren en el origen), o moverlos hasta alcanzar dicho punto, considerado en el primer cuadrante. Posteriormente, a través de palabras de control previamente definidas, los microcontroladores indicarán a la Microcomputadora que se encuentran listos para recibir los datos de corte generados por el operador. La comunicación entre los microcontroladores y la PC se realizará por medio de 4 buses, 2 para señales de control y 2 para datos

Para el desarrollo del presente diseño se tomó en consideración que el operador de la máquina realizará el control de la misma a través de una computadora personal (PC), en donde, por medio de una interfaz gráfica se introducirán los parámetros necesarios para iniciar una secuencia de corte. La interfaz gráfica que fue diseñada para este trabajo de investigación se realizó completamente en lenguaje "C" buscando que su diseño fuera lo más amigable y sencillo para el operador. Es importante mencionar que este programa se desarrolló únicamente con la finalidad de complementar el diseño CNC y que puede ser mejorado, abriendo la posibilidad de futuros trabajos de programación.

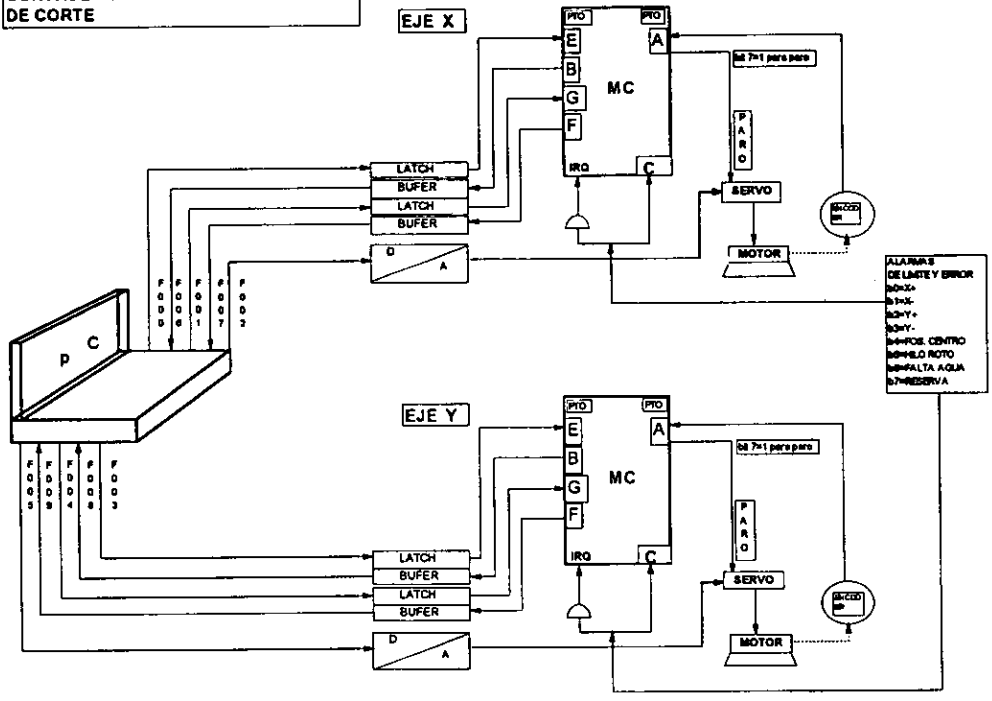
Una vez que el operador ha introducido en la PC los datos necesarios (posición inicial, velocidad de corte, ángulo de inclinación y longitud del corte) estos serán enviados por ella al momento que el operador indica al programa en la PC que inicie el corte, a los microcontroladores a través de los dispositivos de salida (Latches). Los valores serán almacenados dentro del MCU y éste le notificará permanentemente a la PC de cualquier evento que ocurra en la máquina de corte a través de los dispositivos de entrada (Buffers).

Los datos almacenados en localidades de memoria del MCU serán comparados con los datos generados por los Encoder acoplados a los ejes de los motores los cuales servirán para obtener la velocidad a la que trabajan los motores así como el desplazamiento que van consiguiendo.

Los datos de velocidad tecleados por el operador son dirigidos al convertidor Digital–Analógico (DAC) quién los prócesa para obtener la información necesaria, la cual será a su vez recibida por los Servos para el arranque y movimiento de los motores.

Una vez que se inicia el proceso de corte, el control completo de la máquina de corte será realizado por los Microcontroladores, los cuales a su vez enviarán información a la PC referente al avance del corte expresado en porcentaje. Los microcontroladores podrán, en cualquier momento del proceso, detenerlo en caso de error o al finalizar el corte, siendo que la PC funcionará únicamente como pantalla del estado del corte. Un diagrama global del sistema de control se presenta a continuación:

ESQUEMA A BLOQUES DEL SISTEMA DE CONTROL NUMÉRICO DE UNA MÁQUINA DE CORTE



Capítulo 2

DESCRIPCIÓN DE CONTROLADORES PARA MOTORES DE CD Y MOTORES DE PASOS

2.1 INTRODUCCIÓN

En muchas de las situaciones industriales, los motores son operados directamente de las líneas de alimentación *AC* o *DC*. Es decir, los terminales de los devanados del motor están conectados directamente a las líneas que entregan la corriente eléctrica. En estas situaciones, el funcionamiento del motor está determinado por la naturaleza de la carga mecánica conectada a su eje. En pocas palabras si la carga es fácil de manejar el motor tenderá a entregar un torque relativamente pequeño, y girará a alta velocidad. Si la carga es difícil de manejar, el motor tenderá a entregar un gran torque, y girará a baja velocidad. Lo importante es que el funcionamiento del motor está determinado por su *carga* (para un voltaje de alimentación fijo), y el operador no tiene control sobre su operación.

En las situaciones industriales modernas, hay muchas aplicaciones que requieren que el operador sea capaz de intervenir para controlar la velocidad del motor, o bien que el control sea automático.

Los motores de DC tienen varias aplicaciones y frecuentemente se utilizan controladores de velocidad para los mismos. Los motores de DC pueden proveer un torque de arranque grande y es posible obtener el control de velocidad de estos motores. Los métodos para el control de motores de DC son normalmente simples, dicho control generalmente es realizado con tiristores, y son menos costosos que los controladores para motores de AC.

También existen aplicaciones en las que además de tener un control sobre la velocidad de un motor, se necesita un control mucho más preciso, es decir; a veces necesitamos una rotación parcial, un giro de unos cuantos grados. Los motores de pasos pueden ofrecer un control de posición con un alto grado de exactitud.

En el presente capítulo se dará una descripción del funcionamiento de los controladores para motores de pasos y motores de corriente directa.

2.2 EL MOTOR DE PASOS

Los motores de pasos son usados en una variedad de aplicaciones de automatización en las cuales es necesario un torque relativamente pequeño. Los motores de pasos pueden ser usados por su grado de precisión, sin la necesidad de un complicado sistema de retroalimentación que indique la posición.

Cuando el motor de pasos es alimentado con un pulso, el motor gira un ángulo determinado por ejemplo, 1.8°. Cada pulso de entrada resulta en una rotación, mediante la cual se obtiene un cierto ángulo ya clasificado. Por ejemplo, con 41 pulsos a 1.8° por pulso, el motor giraría precisamente 73.8°. La tolerancia radial en la que se ubican estos motores, es típicamente del 3%-5% por paso. Para el ángulo de 1.8° del ejemplo anterior, su tolerancia sería del 4%, (0.04)x(1.8) por lo que se tendrá una tolerancia de al rededor de los 0.07°.

Los motores de pasos normalmente no necesitan indicadores de posición de retroalimentación, por lo que ellos funcionan, para un punto determinado, confiable y cabalmente cuando son programados adecuadamente. En aplicaciones criticas, sin embargo, deben de ser usados los indicadores de posición de retroalimentación para un control preciso. En la figura 5 se muestra la apariencia típica de los motores de pasos.

Fig. 5 Motores de pasos

La figura 6 muestra una tabla que enlista algunos de los estándares existentes para el ángulo por paso, conjuntamente con el paso máximo típico a clasificar. La tabla también incluye el número de pasos por revolución. La relación entre los pasos por revolución y el ángulo por paso esta dada por la siguiente formula:

Ángulo por paso = 360°/(núm. de pasos por revolución)

Pasos (completos) por revolución	Angulo de paso (Grados)	Rango máximo de giro (pasos por segundo)
400	0.72	1000
200	1.8	2000
96	3.75	1000
48	7.5	1000
24	15	600
20	18	500

Fig. 6 Rangos de motores de paso estándar

El torque que pueden manejar los distintos modelos de motores de pasos, es de 0.5 a 5000 onzas por pulgada. El rotor del motor de pasos son de imán permanente o de reluctancia variable. En esta investigación se analiza el tipo de rotor de imán permanente. El motor de pasos típico tiene 5 o 6 alambres para su conexión. El esquema de conexión típica es mostrado en la figura 7. En algunos motores el cable negro y el blanco son combinados en una conexión común. El esquema de conexión es el mismo para todos los modelos con el mismo número de pasos por revolución.

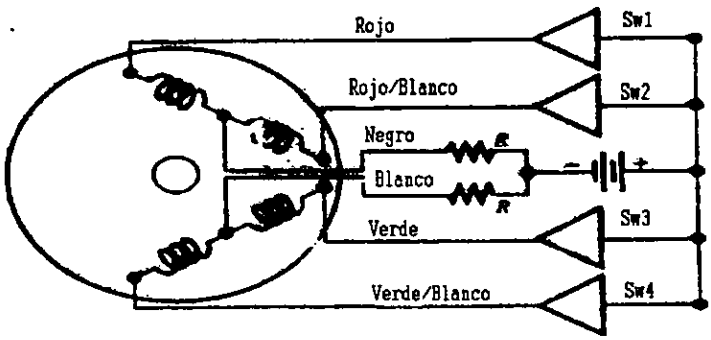


Fig. 7 Conexiones típicas de un motor de pasos

El motor de pasos puede operar en modo de *FULL-STEP* o en modo de *HALF-STEP*. Para ilustrar estos dos modos, se muestra la figura 8. En la figura, el motor tiene un estator de cuatro polos y un rotor de seis polos. Hay cuatro interruptores de control. Cada interruptor enciende la bobina de un determinado polo, como se muestra. Cuando se enciende, un polo esta orientado a un polo norte del rotor. Cuando esta apagado, el polo es magnéticamente neutral. El rotor tiene seis polos magnéticos con la polaridad mostrada. Cuando el interruptor esta en ON, la alineación del rotor y el estator comienza. Pasando los interruptores de ON a OFF en secuencia permite que el motor de un paso. Para *full steps*, el rotor gira 30°, y para *half steps*, 15°.

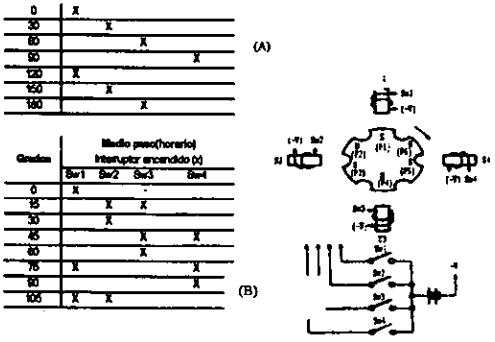


Fig. 8 Principio operacional de un motor de pasos

El interruptor de ON-OFF para paso completo y medio paso están ilustrados en las Tablas A y B de la figura 8. Para paso completo, como se muestra en la Tabla A, P1 se alinea con S1 inicialmente. Cuando S2 esta en ON, el rotor gira 30°, hasta que P3 se alinea con S2. Las secuencias posteriores de cada ángulo por paso se ilustraron en la Tabla A. Para medio paso, la secuencia de interruptores se muestra en la Tabla B. La posición inicial es la misma que la de la Tabla A. Para un giro de 15°, S3 y S4 se colocan en ON ambos interruptores, como se muestra en la Tabla B. El polo P3 del rotor se alinea entre S2 y S3, ya que ambos tienen una fuerza magnética equivalente. La secuencia para 15° por paso continúa como se muestra en la Tabla B de la figura 8.

Si se requiere que el motor conmute cierto número de veces por segundo, este no puede usar interruptores mecánicos para su control, por lo que debe usarse un IC que produzca los pulsos apropiados para su operación. La secuencia de pulsos que se requiere se muestra en la figura 9 para una velocidad de 10 pasos por segundo. Las secuencias para ambas, CW y CCW son mostradas.

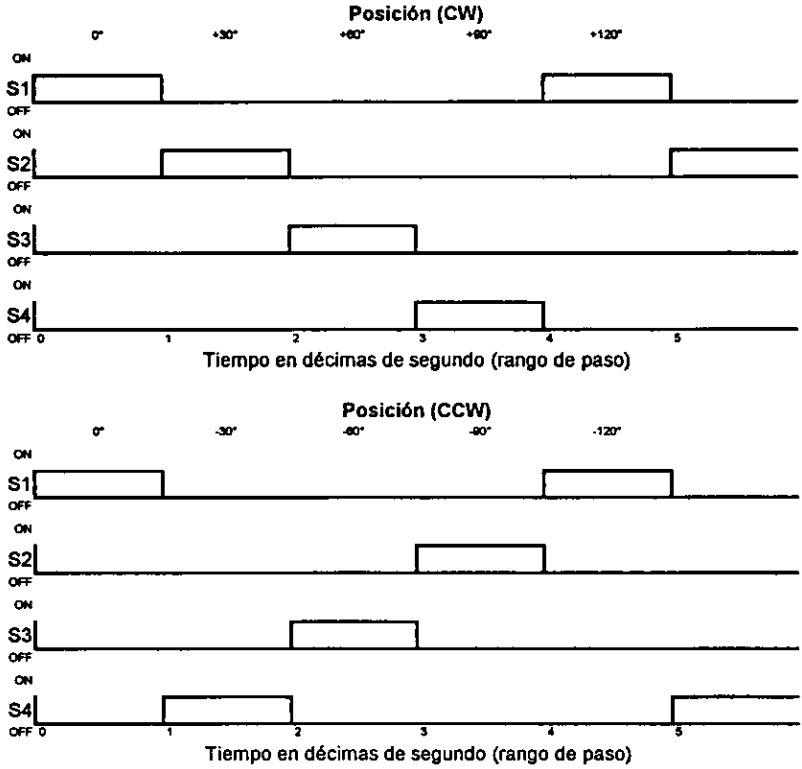


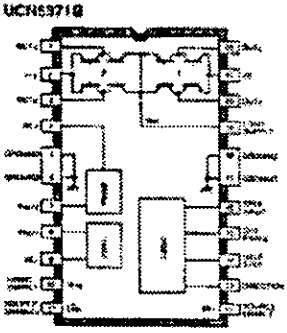
Fig. 9 Control de pulsos de un motor de pasos

Un IC típico para controlar motores de pasos es el que se ve en la figura 10, cada terminal esta identificada con base a las especificaciones existentes. Este IC en particular es para un motor de dos fases. Otros integrados están disponibles para cuatro o más numero de fases.

La función de cada terminal del IC es la siguiente:

1. OUTB *Voltaje de salida a la linea B del motor (fase B-D)*
2. VE2 *Corriente para protección de sobrecarga y control*
3. OUTD *Voltaje de salida a la linea D del motor (fase B-D)*
4. RC2 *Control de apagado para la fase B-D*
5. Ground *Tierra fisica*
6. Ground *Igual que 5*
7. Vref1 *Voltaje de error para sobrecarga fase B-D*
8. Vref2 *Voltaje de error para sobrecarga fase A-C*
9. RC1 *Control de apagado para la fase A-C*
10. LS *Voltaje de alimentación (+5 V)*
11. SE2 *Habilita la fase B-D con un HIGH; deshabilita la fase B-C con un LOW*
12. SE1 *Igual a 11 para A-C*

13. Dir	*Dirección de giro; HIGH - da un paso en sentido CW; LOW - da un paso en sentido CCW*
14. Half step	*HIGH- avanza medio paso (15° half step); LOW- avanza un paso completo (30° full step)*
15. One phase	*HIGH solo se activa B-D o A-C; LOW una fase o ambas se activan*
16. Step input	*Detecta un paso cuando el voltaje de este va de LOW a HIGH.*
17. Ground	*Igual que 5*
18. Ground	*Igual que 5*
19. Ld Sup	*Voltaje de alimentación del motor - 45V DC voltaje típico*
20. OUTA	*Voltaje de salida a la línea A del motor (fase A-C)*
21. VE1	*Igual a 2 para fase A-C*
22. OUTc	*Voltaje de salida a la línea C del motor (fase A-C)*



Rangos máximos absolutos a Ta=25°C

Voltaje de alimentación al motor Vbb.....**45V**
Corriente de salida Iout (continua)..... **1.0A**
(Pico)..... **1.5A**
Voltaje de alimentación lógico Vdo..... **7.0V**
Voltaje lógico de entrada
Rango Vin .....**-0.3 V a Vdo+0.3V**
Voltaje de sensado Ve..... **1.0V**
Disipación de potencia del
encapsulado Pd.....**ver gráfica**
Rango de temperatura de
operación Ta.....**-20°C a +85°C**
Rango de temperatura de
almacenaje Ts..... **-55°C a +150°C**

Fig. 10 Circuito Integrado Controlador de Motores de Pasos bifásico y sus características principales

Cuando este IC es usado en conjunto con otros circuitos digitales de pulsos, puede cont.rolarse con exactitud los movimientos del motor de pasos.

2.3 PRINCIPIOS DEL MOTOR DE DC

Un motor eléctrico transforma la energía eléctrica en un torque. En la figura 11 podemos ver el principio de operación básico de un motor de DC. Cualquier motor de DC consta de dos imanes. El primero de estos es un imán fijo llamado CAMPO. El segundo imán esta unido a un eje y puede girar libremente. Este es llamado ARMADURA. En algunos textos *el campo es llamado estator y la armadura se le conoce como rotor.*

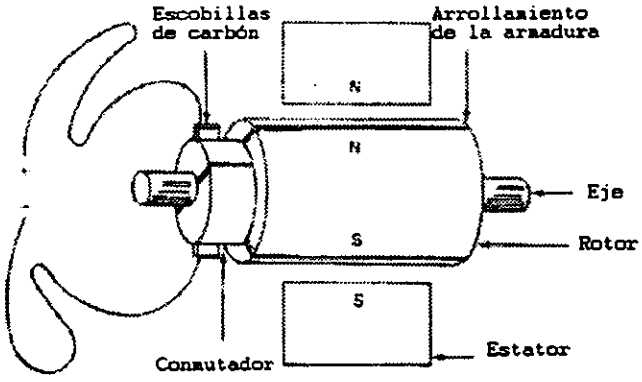


Fig. 11 Estructura simple del motor de DC

El campo en la figura 11 es un imán permanente. La armadura consta de varias vueltas de alambre, formando una bobina, cuando la corriente directa fluye a través de esté, forma un electroimán.

Puesto que la armadura debe estar libre para girar, la energía es conectada al devanado de la armadura a través del conmutador que también se encuentra unido al eje. Las dos partes del conmutador se encuentran aisladas una de la otra y también del eje metálico del motor. La energía llega al conmutador a través de los carbones, los cuales resbalan sobre el mismo.

Cuando la energía es aplicada al devanado de la armadura, la corriente fluirá de la terminal negativa de la fuente, a través del devanado, y regresará a la fuente de DC. El devanado se transforma en un electroimán. En la figura 11 la polaridad del electroimán es tal, que el polo norte magnético de la armadura esta cercano al polo norte magnético del estator. Puesto que dos fuerzas magnéticas de la misma polaridad se repelen una a la otra, la armadura será forzada a alejarse del campo magnético del estator, entonces el eje del motor girará.

Como la armadura se mueve, su polo norte comienza a aproximarse al polo sur del estator, y la atracción entre dos diferentes polos mantiene el movimiento hasta que el polo norte de la armadura esté enfrente del polo sur del estator.

El movimiento podría detenerse en este punto, excepto que el conmutador girará con la armadura. El resultado de esto es que la polaridad de la fuente de energía que le llega a los devanados de la armadura estará invertida. Esto, naturalmente, invierte la polaridad magnética de la armadura, provocando que el polo norte magnético de la armadura ahora pase a ser el polo sur, que es repelido por el polo sur magnético del estator, manteniendo el movimiento. Este es el principio que hay detrás de todos los motores de DC.

Cuando nosotros pasamos un alambre a través de un campo magnético, un voltaje es inducido en el alambre. El voltaje inducido en las bobinas de la armadura es de polaridad opuesta

al voltaje aplicado. Este voltaje inducido se le llama Fuerza Contraelectromotriz (FCEM). En un motor sin carga, la FCEM será casi equivalente al voltaje de armadura aplicado, reduciendo así la corriente de armadura.

El motor shunt es el más común de los motores de DC. Tiene una buena regulación de velocidad, y aún cuando la velocidad del motor se reduce, con un incremento de la carga, es considerado un motor de velocidad constante.

2.4 VARIACIÓN DE VELOCIDAD DE UN MOTOR SHUNT DC

Básicamente, hay dos maneras de variar la velocidad de rotación de un motor shunt DC:

1.- Ajustando el voltaje (y corriente) aplicado al devanado de campo.
2.- Ajustando el voltaje (y corriente) aplicado a la armadura.

Control de campo. A medida que el voltaje de campo se incrementa, por ejemplo, reduciendo Rv en la figura 12(b), la corriente de campo se incrementa. Esto produce un campo magnético más fuerte, el cual induce una mayor FCEM en el devanado de armadura. Esta FCEM más grande tiende oponerse al voltaje DC aplicado y entonces reduce la corriente de armadura, (IA). Por consiguiente, un aumento en la corriente de campo hace que el motor disminuya su velocidad hasta el punto que la FCEM inducida regrese a su valor normal (aproximadamente).

Sin embargo, hay una gran desventaja al controlar la velocidad desde el devanado de campo: Para aumentar la velocidad, se debe reducir If y debilitar el campo magnético, con esto disminuye la habilidad de producción de torque del motor.

Control de armadura. A medida que el voltaje y la corriente de armadura se incrementan (por la reducción Rv en la figura 12(c)), el motor comienza a girar más rápido, lo cual normalmente requiere de más torque.. La razón del aumento de la velocidad es que al aumento del voltaje de armadura demanda un incremento en la FCEM para limitar el incremento de la corriente de armadura a una cantidad razonable. La única manera como puede incrementarse la FCEM es haciendo que el devanado de armadura gire más rápido, dado que la fuerza del campo magnético es fija. En esta instancia, todos los ingredientes están presentes para aumentar la producción de torque dado que la fuerza del campo magnético se mantiene constante e IA es incrementada.

El problema con el método de control de armadura de la figura 12 (c), es que Rv, el reóstato, debe manejar la corriente de armadura, la cual es relativamente grande. Por lo tanto el reóstato debe ser físicamente grande y costoso, y disipará una cantidad de energía considerable.

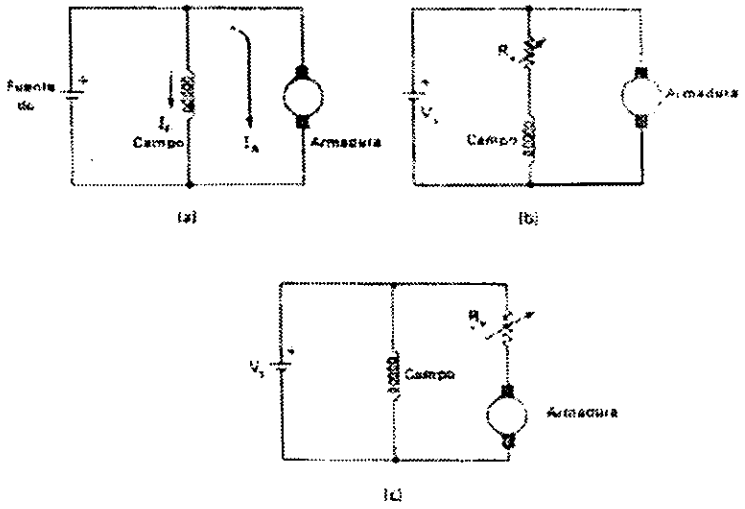


Fig. 12 (a) Representación esquemática de un motor shunt dc. (b) un reóstato en serie con el devanado de campo para controlar la velocidad del motor. (c) Un reóstato en serie con la armadura para controlar la velocidad del motor.

2.5 CONTROL DE VOLTAJE Y CORRIENTE DE ARMADURA USANDO TIRISTORES.

Un SCR puede ejecutar el trabajo del reóstato en el control de la corriente promedio a una carga. Además, un SCR o cualquier tiristor de potencia no tiene los inconvenientes de los reóstatos de alta potencia. La distribución general de un sistema de control de velocidad con SCR se ilustra en la figura 13.

La fuente de AC es rectificada para proporcionar potencia para el devanado de campo. El SCR proporciona control y rectificación de media onda al devanado de armadura. Utilizando ángulos de disparo pequeños en el SCR, el promedio del voltaje y la corriente de armadura se incrementan y el motor puede girar más rápido. Cebando tardíamente el SCR (aumentando el ángulo de disparo), se reduce el promedio de voltaje y la corriente de armadura, y el motor gira más lento

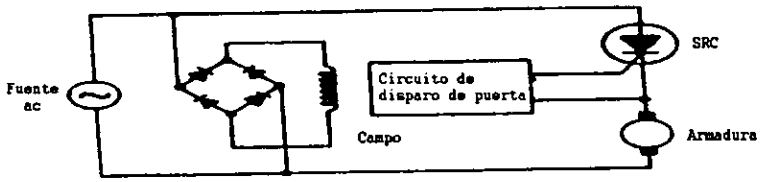


Fig.13 Un SCR en serie con la armadura para controlar la velocidad del motor.

2.6 SISTEMA DE CONTROL DE VELOCIDAD MONOFÁSICO Y DE MEDIA ONDA PARA UN MOTOR SHUNT DC

La figura 14 muestra un circuito simple de media onda para controlar la velocidad de un motor DC. La velocidad se ajusta por medio del potenciómetro de ajuste de velocidad de 25 K. A medida que el potenciómetro es movido hacia arriba (su contacto móvil se mueve alejándose de tierra) crece la velocidad del motor. Esto sucede debido a que el voltaje de compuerta respecto a tierra se hace una porción más grande del voltaje de línea AC permitiendo que el voltaje de compuerta a cátodo alcance el voltaje de disparo del SCR más pronto en el ciclo.

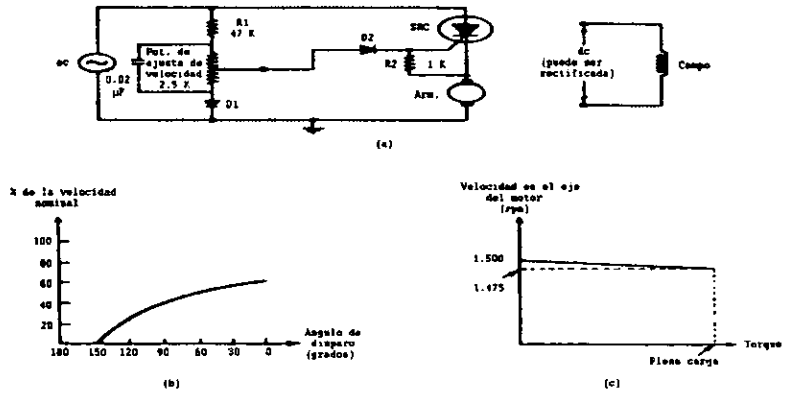


Figura 14 (a) diagrama esquemático de un circuito operador de media onda con SCR. (b) gráfica de velocidad en el eje versus ángulo de disparo para el circuito de la parte (a).(c) gráfica de velocidad versus torque para un valor fijo determinado del potenciómetro de ajuste de velocidad en la parte (a) .

A medida que el potenciómetro de ajuste de velocidad se mueve hacia abajo, el voltaje de puerta a tierra se vuelve una pequeña porción del voltaje de línea, de modo que toma más tiempo para que V(GK) alcance el valor necesario para cebar el SCR.

La relación entre la velocidad y el ángulo de disparo para este sistema está graficada en la figura 14 (b). Como puede verse, es imposible que el motor alcance el 100% de su velocidad nominal porque el sistema solamente puede entregar potencia de media onda a la armadura.

La habilidad del sistema de control de velocidad para mantener la velocidad del motor constante frente a variaciones en la carga se denomina regulación de carga. Una Expresión matemática para la regulación de carga está dada como:

Regulación de carga =RPMV - RPMPL / RPMPL

donde RPMV significa la velocidad de rotación en vacío (sin carga). La frase en vacío significa que el torque resistente de la carga que tiende a disminuir la velocidad del motor es igual a cero. RPMPL significa la velocidad de rotación a plena carga, lo cual significa que el torque resistente de la carga que tiende a disminuir la velocidad del motor es máxima. Puede verse de la ecuación anterior que entre más pequeño sea el cambio en velocidad desde la condición de vacío a la condición de plena carga, más pequeña es la regulación de carga. Por tanto, entre más pequeño sea el valor de la regulación de carga, mejor será el sistema de control.

Como ejemplo específico del calculo de regulación de carga, refirámonos a la figura 14 (c). Podemos ver que la velocidad en vacío es de 1500 r.p.m. y que la velocidad de plena carga es de 1475 r.p.m. Por tanto, la regulación de carga está dada por:

reg. de carga = (1500 r.p.m. - 1475 r.p.m.)/1475 r.p.m. = 0.017 o 1.7%

Para muchas aplicaciones industriales, una regulación de carga del 1.7% es bastante adecuada

2.7. CONTROL REVERSIBLE DE VELOCIDAD

Algunas aplicaciones de control de velocidad en la industria requieren que la rotación de un motor sea reversible. La inversión de la dirección de rotación puede efectuarse de dos maneras:

1. Invirtiendo la dirección de la corriente de campo, manteniendo la misma dirección de la corriente de armadura.
2. Invirtiendo la dirección de la corriente de armadura, manteniendo la misma dirección de la corriente de campo.

La figura 15 muestra como pude invertirse la corriente de armadura en un sistema de control de velocidad de onda completa. El método más directo para invertir la corriente de armadura o la corriente de campo es utilizando separadamente dos arrancadores de motor. El contactor directo hace que la corriente fluya a través de la armadura en una dirección, mientras que otro contactor reversa, hace que la corriente fluya en la dirección opuesta.

En la figura 15 (a) el contactor DIR se energiza al presionar el botón pulsador ARRANQUE DIRECTO. Mientras el contactor REV se encuentre fuera en este momento, el contactor DIR se energizará y enclavará gracias a su contacto N.A. en paralelo con el botón pulsador. El operador puede entonces liberar el botón ARRANQUE DIRECTO y el contactor energizado hasta cuando se presione el botón pulsador PARADA.

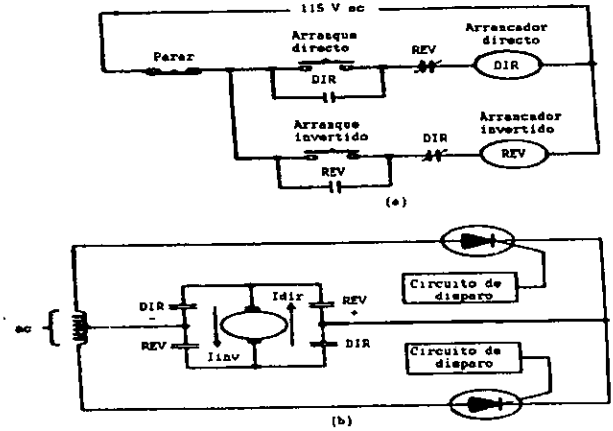


Figura 15. Sistema operador reversible de onda completa con SCR . (a) Circuito de control de arranque del motor. (b) Circuito de armadura. Los SCR se ceban en semiciclos alternados haciendo que el voltaje de armadura tenga la polaridad indicada. La dirección de la corriente de la armadura depende de cual de los contactos DIR o REF están cerrados.

En la figura 15 (b) puede verse que cuando están cerrados los contactos DIR, la corriente fluye a través de la armadura de abajo hacia arriba, con esto produce rotación en una cierta dirección (asumamos en el sentido de las manecillas del reloj). Cuando están cerrados los contactos REV, la corriente de armadura fluye de arriba hacia abajo, de este modo produce rotación en el sentido contrario de las manecillas del reloj. Como ya sabemos, la velocidad de rotación se controla por el ángulo de disparo de los SCR.

El control de onda completa reversible sin utilizar dispositivos con contactos (contactores, botones pulsadores, etc.) es el circuito de la figura 16. En la figura 16, la dirección de rotación está determinada por el circuito de disparo que esté habilitado. Si esta habilitado el circuito de disparo directo, los dos SCR de la parte de arriba se cebarán en semiciclos alternados de la línea de AC, y enviarán corriente a través de la armadura de derecha a izquierda. Si está habilitado el circuito de disparo invertido, los dos SCR de abajo se cebarán en semiciclos alternados de la línea de AC, y enviarán corriente a través de la armadura de izquierda a derecha, como se indica. El método para habilitar un circuito de disparo mientras se inhabilita el otro no se muestra en la figura 16.

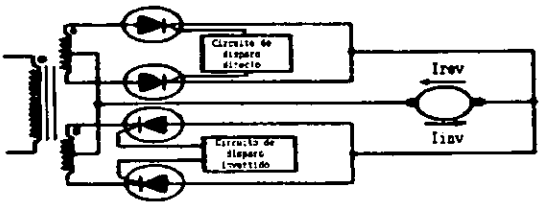


Figura 16. Sistema manejador reversible de onda completa y totalmente de estado sólido.

2.8 MODULACIÓN POR ANCHO DE PULSO (PWM)

La energía suministrada a una carga usando un sistema basado en tiristores tiene una desventaja: Al alimentar a tensión reducida, se obtiene un par bajo.

Una atractiva alternativa es la Modulación por Ancho de Pulso (PWM). El concepto es bastante simple. En lugar de operar un transistor en su región lineal, este es operado en modo conmutado. El transistor es operado en la región de saturación o en la región de corte. Bajo estas condiciones, la carga esta sujeta a pulsos de voltaje de una amplitud casi igual al voltaje de alimentación. Al controlar la duración (ancho) de los pulsos, y al tener una apropiada filtración de estos pulsos, la cantidad promedio de voltaje suministrado a la carga puede ser controlado.

La principal ventaja del control por PWM es su alta eficiencia. Cuando el transistor es saturado, el voltaje que cae en él, es de unos cuantos volts máximo. Aún cuando la corriente sea elevada a través del transistor, la disipación de energía será baja. Cuando el transistor está en la región de corte, la corriente a través de el será insignificante.

Un sistema de control de PWM consiste principalmente en cuatro componentes: una forma de onda triangular (o de rampa), un comparador, un interruptor para la alimentación, y un filtro de salida. En la figura 17 un amplificador operacional LM324 es utilizado como comparador; Q1 y Q2 forman el interruptor para la alimentación; y D1, L y C forman el filtro de salida. Es recomendable comprender que en muchos casos la frecuencia de oscilación es generalmente cercana a los 20khz, y que el voltaje de control varía muy lentamente con respecto a esta frecuencia. Como la oscilación del voltaje de salida varía sobre y abajo del nivel del voltaje de control, los pulsos son generados a la salida de U1A. Estos pulsos hacen que los transistores Q1 y Q2 pasen alternadamente de la región de saturación a la región de corte a una razón proporcional a la frecuencia de oscilación, funcionando como un interruptor. La duración de cada pulso es determinado por el nivel del voltaje de control. La salida filtrada de los pulsos produce un voltaje que es igual al producto del ciclo de trabajo y el voltaje de alimentación.

La relación entre el voltaje de control, la oscilación de salida, la salida de U1A, y el voltaje de la carga es mostrado en la figura 18. Note como el promedio del voltaje hacia la carga se incrementa al igual que el periodo de los pulsos.

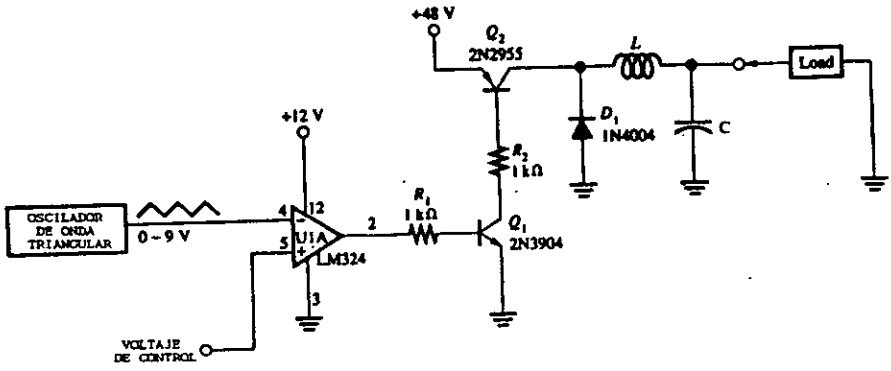


Fig. 17. Sistema modulación por ancho de pulso

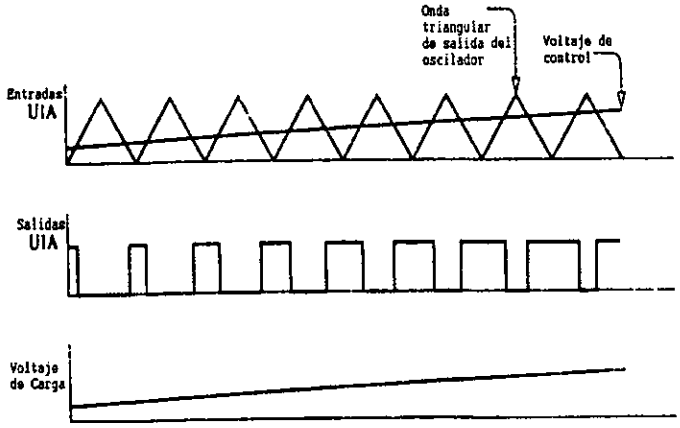


Fig. 18. Relación voltajes en la modulación por ancho de banda.

Capítulo 3

OPERACIÓN DE CODIFICADORES DE DESPLAZAMIENTO (ENCODER)

Estos dispositivos (CODIFICADORES) tienen amplia aplicación tanto en la industria, medicina, productos de oficina, periféricos de computadoras, así como en la industria aeroespacial. Su diseño proporciona confiabilidad y una operación libre de fallas en ambientes difíciles de la industria. A continuación listamos algunas de las características de los Encoder para trabajo pesado :

- Construcción robusta, herméticos y de molde troquelado.
- Diseño para trabajo industrial pesado.
- Larga flecha de acero inoxidable.
- Fuente de luz de estado sólido para el LED en el caso de tecnología óptica
- Rango amplio de temperatura de trabajo

Para el cada vez mayor ruido electromagnético que se presenta en el medio ambiente de las fábricas, los Encoder pueden incluir manejadores de línea diferencial, lo cual ayuda a minimizar los efectos de conducción y radiación del ruido, lo que garantiza la transmisión de la señal en forma integra.

Para facilitar su montaje en la industria los Encoder incluyen características mecánicas estándar, así como interfaces eléctricas estándar, lo cual hace posible que los Encoder puedan ser seleccionados con confianza.

Las formas de onda que entregan a su salida (onda cuadrada) son compatibles con la mayoría de los instrumentos PCL'S, contadores electrónicos y manejadores de motores, la mayoría de los Encoder incluyen el manejador de línea diferencial, esto permite su uso en ambientes de alto ruido electromagnético y permiten longitudes de transmisión más largas ó aumento de la capacidad de salida. Adicionalmente manejan un rango alto de voltajes.

En el mercado existen ENCODER con dos tipos de tecnología de sensado :

- Tecnología magnética.
  Esta tecnología proporciona buena resolución, muy alta velocidad de operación y excelente resistencia a la humedad y al polvo, así como a los golpes mecánicos y variaciones térmicas.

- Tecnología óptica.
  Presenta las mismas características que la anterior y adicionalmente un empaquetado robusto y confiable así como una larga vida de operación en la mayoría de los ambientes industriales.

3.1. APLICACIÓN DE LOS ENCODER

Los Encoder son compatibles con la mayoría de los sistemas de monitoreo y control, son particularmente adaptados a la industria así como a los ambientes que predominan en las fábricas.

Vía los sistemas de computo industrial, instrumentos y contadores, se puede llevar a cabo la programación de velocidad, longitud, posición y movimiento a partir de la toma de datos de las señales que entregan los Encoder y con esto tener varias posibilidades de aplicación de los mismos en diferentes procesos, siendo algunos de ellos los sig.:

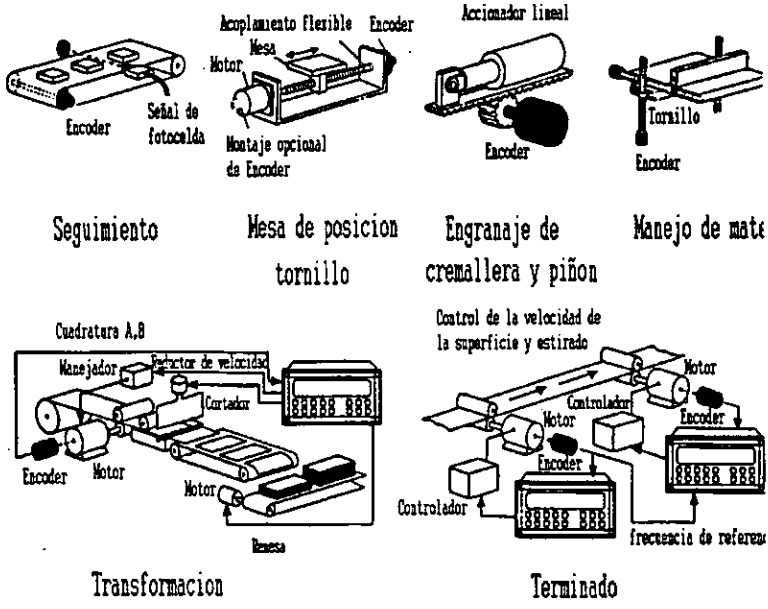
- Maquinado de herramientas
- Cortes a longitud
- Procesos de seguimiento
- Mesa de posición de tornillo de bola
- En sistemas de engranaje
- Manejo de materiales
- Procesos de transformación
- Procesos de terminado
- Empaquetado, llenado y etiquetado
- Sistemas de control numérico de máquinas de corte por descargas eléctricas

A continuación se muestran en forma esquemática los diferentes procesos y la posible ubicación de los Encoder.

Máquinado Cortes a longitud

Fig. 19 Aplicaciones típicas de los Encoder en procesos de fabricación

3.2 CONTINUACIÓN APLICACIONES TÍPICAS



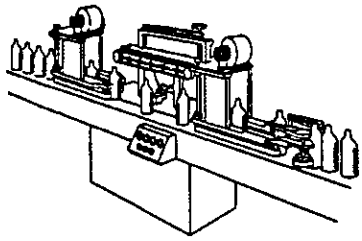


Fig. 20 Algunas aplicaciones de los Encoder.

Con lo anterior se tiene como resultado que en la actualidad existan en la industria SISTEMAS DE MANUFACTURA FLEXIBLES, los que podríamos representar en forma esquemática como sigue :

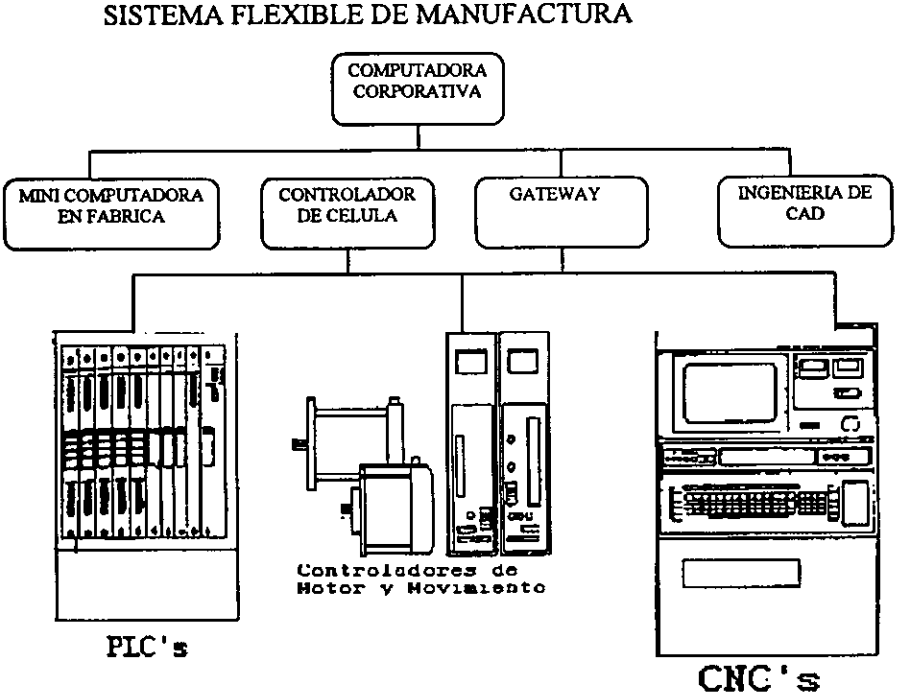


Fig. 21a. Diagrama a bloques de un sistema de manufactura flexible

A continuación se muestran algunos modelos de Encoder.

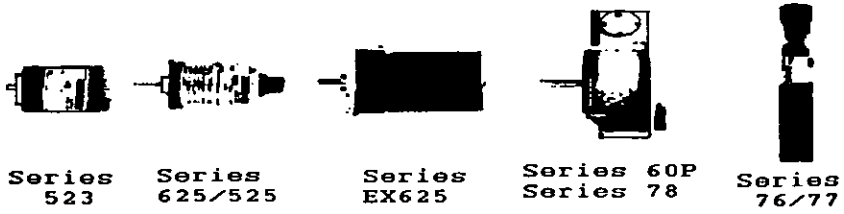


Fig. 21b. Figuras de diferentes modelos de Encoder comerciales

3.3. PRINCIPIOS DE SENSADO

Como mencionamos anteriormente trataremos dos formas de sensado, óptico y magnético, las que describiremos a continuación :

- Sensado óptico

Para describir este proceso nos basaremos en la figura 22.

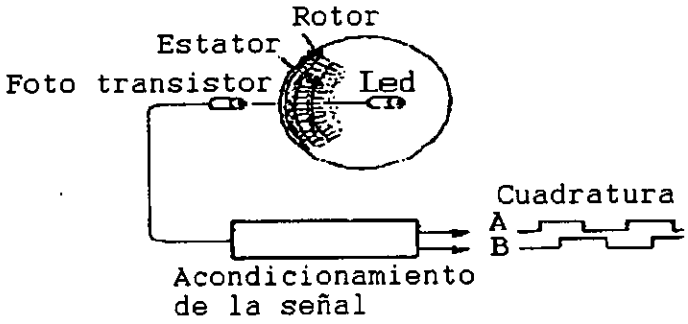


Fig. 22 Sensado óptico

En este proceso los Encoder ópticos utilizan un rotor con un patrón de cuenta que interrumpe la fuente de luz del LED y cambia la salida de los sensores del fototransistor, la señal es amplificada y formada internamente. Los Encoder ópticos proporcionan operación desde velocidad cero y tienen alta capacidad de resolución.

La señal de salida que entregan los Encoder es cuadrada y permite el monitoreo de la dirección del proceso la cual puede invertirse (hacia adelante o hacia atrás) ó puede guardar la posición neta cuando no se mueva u oscila mecánicamente. Es recomendable utilizar Encoder con señal cuadrada bidireccional, ya que son aplicables a la mayoría de necesidades de posición, velocidad y longitud.

3.4. SENSORES MAGNÉTICOS

Existen varios tipos de sensores magnéticos, dentro de estos tenemos los siguientes.

- Reluctancia variable

Los dispositivos de reluctancia variable (fig. 23) utilizan dientes de engranes ferromagnéticos para perturbar el flujo, causando Un cambio en la reluctancia. Un voltaje pulsante proporciona el

movimiento mecánico, es generado en el embobinado. Son dispositivos pasivos de máxima confiabilidad. La operación a baja velocidad es limitada alrededor de 50 RPM.

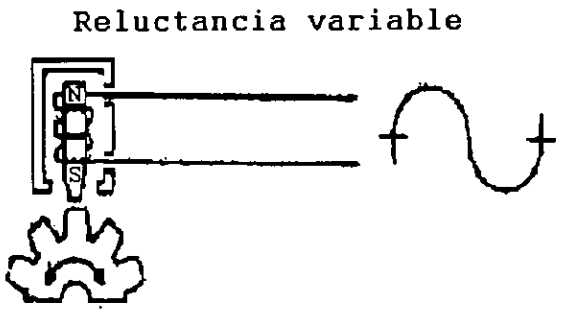


Fig. 23 Sensor de reluctancia variable

- Magneto resistivos

Nos apoyaremos para su descripción en las figuras 24 y 25 siguientes.

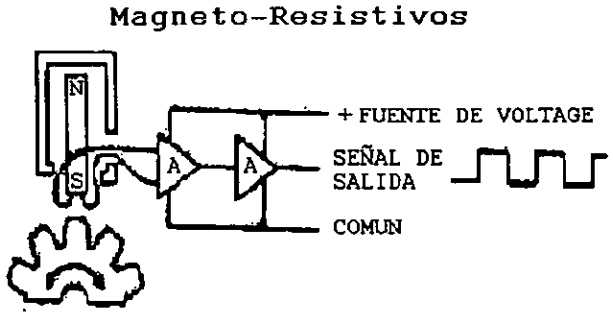


Fig. 24 Principio de operación de los sensores magneto-resistivos

Los dispositivos magneto resistivos contienen un circuito de alta sensitividad que reacciona al movimiento de los dientes de los engranes ferromagnéticos variando su resistencia. El circuito mostrado en la figura 24 se comporta como un circuito puente resistivo que reacciona al movimiento de los engranes haciendo variar la resistencia de una de las ramas (MR), logrando con esto el desequilibrio del circuito produciéndose una circulación de corriente y en consecuencia una variación del voltaje, estas variaciones son amplificadas para crear la señal de salida. Operación a velocidad cero, alta confiabilidad y un amplio rango de temperatura son las características principales de estos sensores.

- Magneto resistivos de alta resolución de tambor rotatorio.

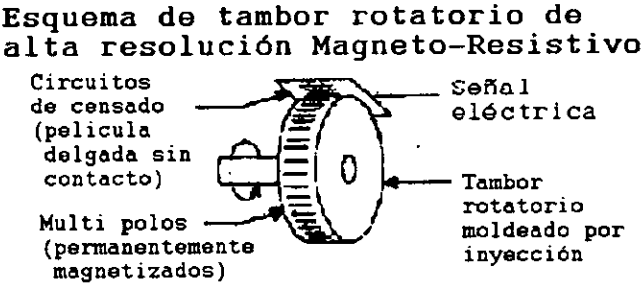


Fig. 25 Sensor magneto-resistivo de tambor rotatorio

El circuito de sensado consiste de dos elementos magneto-resistivos (MR1 y MR2) que responden alternadamente a la codificación de los polos magnéticos en el tambor giratorio. La salida eléctrica es entonces acondicionada y amplificada para proporcionar un flujo de ondas eléctricas cuadradas que corresponden al movimiento del tambor. La figura 26 representa en forma esquemática el circuito básico de sensado.

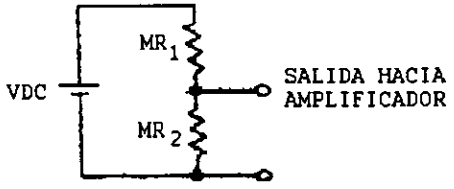


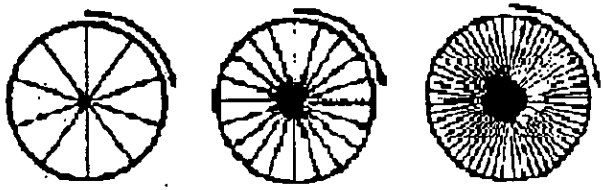
Fig. 26 Circuito básico de los sensores magneto-resistivos

El sensado ocurre en el perímetro, o más bien en la cara del tambor, haciendo la unidad más durable y menos susceptible a golpes y vibraciones. Esta tecnología de sensado magnético y no óptico es más resistente a la mugre, grasa, humedad y otros contaminantes comúnmente encontrados en ambientes industriales.

- Resolución y medición incremental

La resolución (fig. 27) es el número de segmentos o unidades de medición en una revolución del árbol del Encoder. Es la unidad más pequeña de movimiento detectado por el Encoder. Si una revolución del árbol transductor es dividida en mil segmentos, la resolución será 0,001, igualmente, si ésta es dividida en 10 segmentos la resolución será de 0.1. Mediante los Encoder se puede realizar la medición de movimiento o posición mediante incrementos a partir de un punto origen definido (medición incremental), este puede ser de 1 a 5000 pulsos por revolución y con una selección adecuada del Encoder y un instrumento contador, el rango de selección puede

ser extendido hasta 20000 pulsos por revolución, esto se puede lograr con la detección de flancos de los pulsos. El Encoder seleccionado debe tener resolución igual o mayor que la requerida por la aplicación.

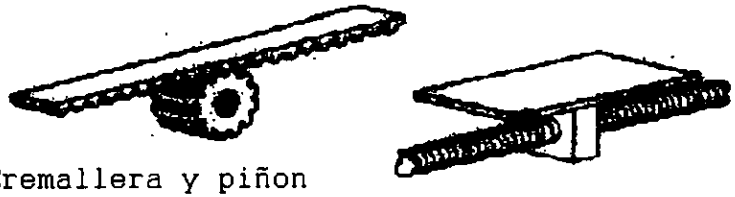


Ejemplos de diferentes resoluciones

Fig. 27

- Técnica de medición lineal / rectilínea.

Los Encoder no están limitados para medición de movimiento rotatorio, a través de medios mecánicos, usualmente mediante el uso de cremalleras y piñón o avance de tornillos fig. (28), el Encoder puede medir movimiento lineal. Estos sistemas mecánicos con engranes y acoplamiento son comúnmente usados en máquinas para conversión del movimiento rotatorio de un motor eléctrico al movimiento rectilíneo deseado. Cuando los Encoders son aplicados a estos sistemas mecánicos, ellos pueden retroalimentar datos de posición y movimiento a los controladores del sistema.



Cremallera y piñon

Mesa conducida por tornillo

Elementos mecánicos típicos rotativos/lineales

Fig. 28 Ejemplo de elementos mecánicos para medición de desplazamiento por Encoder

- Mediciones digitales de posición y longitud.

Digitalmente la medición del longitud y posición es lograda por acumulación y cuenta de pulsos digitales, que están relacionados con la unidad de longitud y movimiento. La longitud o posición total es acumulada desde un punto de referencia establecido antes de principiar la medición. La calibración del número de pulsos por unidad medida es logrado vía la selección del transductor apropiado (Encoder con resolución adecuada). Los dispositivos típicos industriales para efectuar una medición acumulada digital con Encoders, incluyen contadores electrónicos,

instrumentos, controladores lógicos programables (PLC's), controladores numéricos computarizados (CNS's), etc.

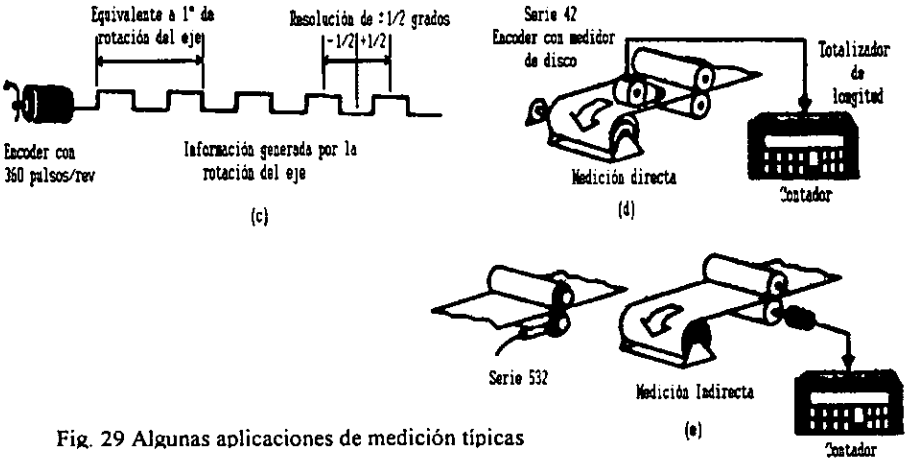


Fig. 29 Algunas aplicaciones de medición típicas

- Medición de velocidad digital.

Un tacómetro digital o medidor de velocidad es simplemente un contador electrónico con una referencia de tiempo. Una técnica de medición del tiempo base es una comprobación de la cuenta recibida desde un Encoder sobre un período dado de tiempo (ver fig. 30). Los Encoders con 60 P.P.R o picos magnéticos con un engrane de 60 dientes, son más frecuentemente seleccionados para medir en R.P.M.; por ejemplo, para mostrar 1 pie por minuto (F.P.M.) usted necesita al menos un B.P.R. por pié sin tener en cuenta la resolución (1,0.1,0.01).

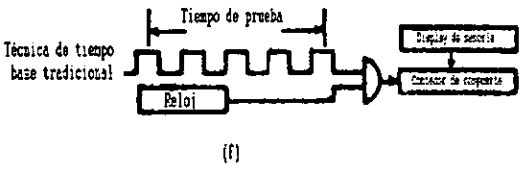


Fig. 30 Comprobación de velocidad en base a tiempo

La medición del intervalo de tiempo entre pulsos por medio de un Encoder es otra técnica, la cual es ilustrada en la fig. 31. Esta técnica es para los incrementos o codificadores con un PPR (Pulsos por Revolución).

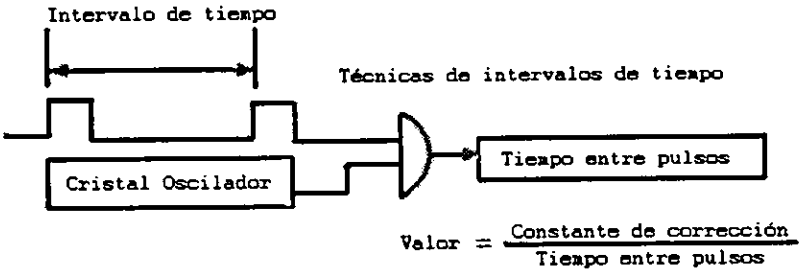


Fig. 31 Técnicas de intervalo de tiempo.

- Medición bidireccional

La mayoría de los sistemas incrementales usan dos canales de salida para el sensado de la posición bidireccional.

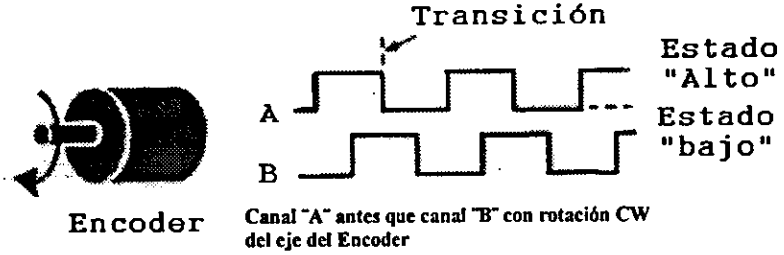


Fig. 32 Medición bidireccional

Esto permite que un contador electrónico incremente la cuenta en cada transición y monitoree el estado del canal opuesto durante esta transición. Usando esta información, podremos determinar si A es primero que B, y de este modo obtener la dirección (cuenta hacia arriba / hacia abajo).

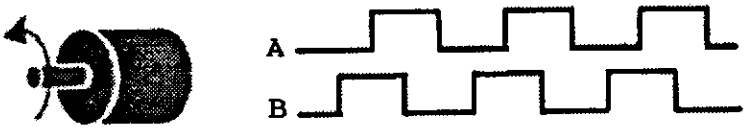


Fig. 33 Encoder Bidireccional

En algunas aplicaciones de arranque y paro unidireccional, es importante tener la información bidireccional (canal A y canal B) incluso si la rotación inversa de la flecha no es requerida.

Un error en la cuenta podrá ocurrir con un Encoder de un solo canal debido a la vibración inherente de la máquina. Por ejemplo un error en la cuenta puede ocurrir con un Encoder de un solo canal en la aplicación de arranque y paro si está mecánicamente rotando, cuando la forma de onda de salida está en transición. Tal que una vibración de la flecha mecánica subsecuentemente forzará el regreso de la salida y adelanta el cruce de este flanco, el contador incrementará la cuenta en cada transición hacia arriba,, incluso aun cuando el sistema está virtualmente parado.

Mediante la utilización de un Encoder bidireccional, el contador monitorea la transición en esta relación del estado del canal opuesto y puede generar información confiable de la posición.

- Multiplicación de pulsos

La mayoría de los instrumentos contadores electrónicos y PLC's incorporan detección bidireccional de alta velocidad en sus circuitos electrónicos. Varios de estos circuitos de detección tienen una característica adicional para manejar 1X, 2X, ó 4X la resolución básica de Encoder. Por ejemplo, estos monitores controladores pueden ser colocados para contar los flancos ascendentes o descendentes del tren de pulsos para la entrada del canal A.

Con esto se duplica (X2) el número de pulsos contado para una rotación del Encoder, lo cual puede mejorar la resolución de la cuenta además de permitir el contar tanto los flancos ascendentes y descendentes del tren de pulsos de la señal A y B. De este modo se logra contar 4 veces (X4) para una misma rotación.

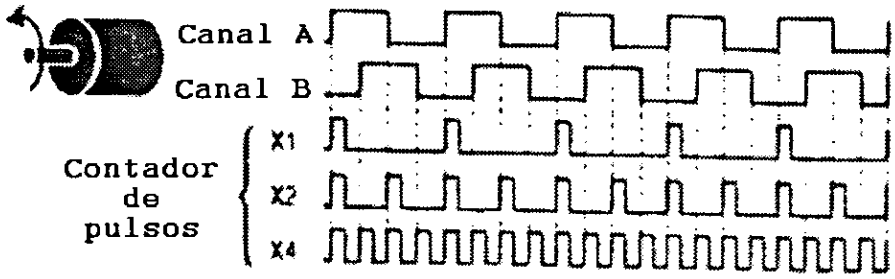


Fig. 34 Encoder multiplicador de pulsos

10,000pulsos/giro pueden ser generados desde un Encoder de 2 canales de 2,500 ciclos.

- Interpretación de la velocidad de operación.

Todos los traductores tienen limitaciones electrónicas y mecánicas inherentes, con respecto a la velocidad.

La combinación de los factores de diseño incluyendo cojinetes, respuesta en frecuencia de los circuitos electrónicos y los PPR (Pulsos por Revolución) de la aplicación, etc., se combinan para determinar "la máxima velocidad de operación " en cualquier operación dada. Al exceder la velocidad máxima puede resultar un dato incorrecto o falla prematura. Para determinar la máxima velocidad del Encoder para una aplicación dada tenemos lo sig. :

1.- Determinar máxima velocidad electrónica de operación en RPM.

$$RPM = \frac{\text{Respuesta en frecuencia del Encoder (KHZ) X 60}}{\text{PPR del Encoder}}$$

2.- Si la RPM calculada en el paso 1 es menor o igual a la máxima especificación mecánica de RPM del Encoder, entonces la RPM calculada del paso 1 es la máxima especificación de velocidad de operación para esta particular aplicación del Encoder.

Los requerimientos electrónicos adicionales para utilizar señales diferenciales están económicamente justificados más frecuentemente cuando:

a. La distancia de transmisión excede aproximadamente 50 pies. Las señales diferenciales usadas con manejadores de línea pueden ser transmitida con buen éxito en cientos de pies con cable apropiado y buenas tierras.

b. Alta inmunidad de ruido eléctrico para las señales de línea sea necesaria.

3.- Si la RPM calculada del paso1 es mayor que la máxima especificación mecánica en RPM del Encoder, entonces la máxima especificación mecánica en RPM es la máxima velocidad de operación para esta aplicación del Encoder.

4.- Compare la máxima velocidad de operación determinada en el paso 2 con la requerida en la aplicación.

- Terminología de las señales de Salida y Clasificación para ENCODERS

Los Encoder trasmiten señales de medición digital utilizando circuitos de corriente directa para alcanzar las velocidades de transmisión más altas, la transmisión es alcanzada por medio del flujo de corriente desde el Encoder con el flujo de corriente para o desde el Encoder (corriente de sumidero) salida de corriente de sumidero o fuente de corriente.

Varios de los circuitos de salida de los Encoder son diseñados para ser compatibles con la mayoría de instrumentos, contadores, controladores y manejadores, de motores, tal que el usuario puede seleccionar y aplicar una unidad con confianza.

3.5. SELECCIÓN ESTANDAR DE SALIDA

5-26 VDC	CORRIENTE DE SUMIDERO
5-15 VDC	MANEJADORES DE LINEA DIFERENCIAL

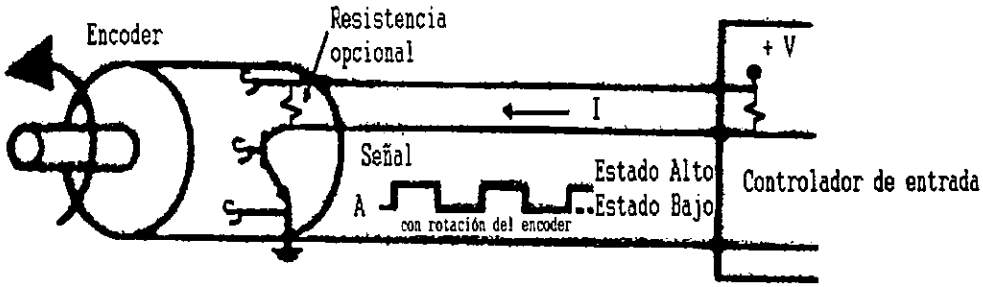


Fig. 35 Flujo típico de corriente en el circuito del Encoder.

- Terminología de la señal de salida.

Como mencionamos previamente los Encoder pueden tener salidas de un solo canal (señal A), dos canales (señal A y B) para mediciones bidireccionales y marcador de canal de pulsos (señal Z) para proporcionar varios requerimientos funcionales para aplicaciones de retroalimentación, velocidad, longitud, posición, etc., en suma las señales son típicamente transmitidas como un solo final o con salidas diferenciales, complementarias.

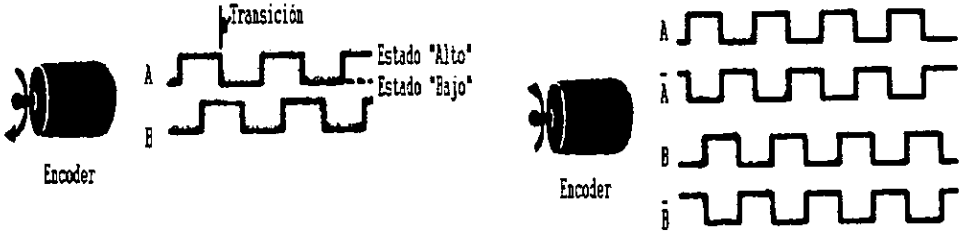


Fig. 36 Encoder con dos y cuatro salidas

Capítulo 4

MANEJO DE SEÑALES DE ERROR Y LÍMITE DE POSICIÓN

Un elemento importante a considerar en las máquinas de corte por descargas eléctricas es el rango máximo en el cual el material o la mesa de trabajo puede desplazarse. Debido a que, este movimiento no es perpetuo o infinito, es necesario considerar las distancias máximas de desplazamiento de los motores antes de generar un estado de error en el equipo. Para ello, se debe contemplar el diseño de un circuito electrónico que permita acoplar las señales analógicas generadas por los sensores de límite de posición y convertirlas a señales digitales que pueda procesar el microcontrolador.

4.1 PRINCIPIOS DE OPERACIÓN DE LOS INTERRUPTORES DE LIMITE DE POSICIÓN

Un interruptor de límite de posición es un sensor de estado sólido generalmente, el cual detectará la presencia de cualquier material adecuado (referido como objetivo) y produce una señal eléctrica de salida mientras el material objetivo se encuentre dentro del rango de sensado. Existen dos tipos de tecnologías utilizadas para el diseño de interruptores de límite de posición, cada uno tiene sus propias e individuales ventajas y aplicaciones, en la figura 37 podemos ver algunos ejemplos de este tipo de sensores.

4.2 INTERRUPTORES DE LÍMITE DE POSICIÓN INDUCTIVOS

Los interruptores de límite de posición inductivos generan un campo de sensado en el cual pueden detectar la presencia de cualquier objetivo de metal. En muchas aplicaciones, los interruptores de límite de posición inductivos son usados para reemplazar la función de un interruptor mecánico, sin la necesidad de contacto físico con bordes, palancas o actuadores utilizados por diversos dispositivos mecánicos.

Otra aplicación primaria de estos sensores es el manejo de materiales. Varias partes o componentes metálicos pueden ser detectados mientras se encuentran en movimiento durante su proceso de manufactura o ensamble. Los interruptores de límite de posición inductivos pueden ser utilizados para confirmar la presencia o localización del material y entregar señales utilizadas para contar, separar, diversificar, inspeccionar o cualquier otro proceso automatizado.

4.3 INTERRUPTORES DE LÍMITE DE POSICIÓN CAPACITIVOS

Al igual que los interruptores de límite de posición inductivos, los interruptores de límite de posición capacitivos son sensores simples, libres de error utilizados para la detección de objetos a distancias cortas. Los interruptores de límite de posición capacitivos sensan un cambio en el dieléctrico del objeto en lugar de la presencia de un objeto de metal. Este tipo de sensores tienen la capacidad de detectar varios tipos de materiales, como por ejemplo líquidos, metal, vidrio, plástico, porcelana, cerámica, madera, piel, goma, comida, agua, aceite, etc.

La habilidad para discriminar los cambios en la masa del objeto permite su uso en aplicaciones únicas, que de otra manera sería muy difícil o imposible de resolver. Por ejemplo pueden "ver" a través de un contenedor o recipiente y determinar la presencia del contenido en el interior. Son muy útiles para verificar el contenido de llenado de materiales terrosos en cajas o de líquidos en botellas.

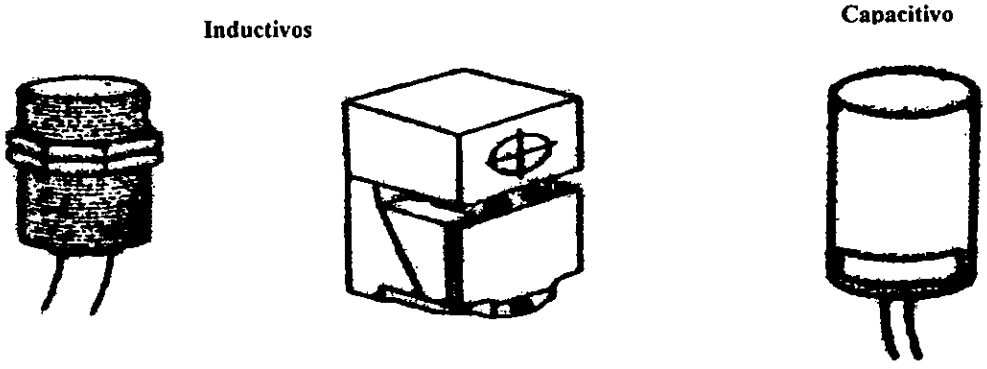


Fig. 37 Tipos de interruptores de límite de posición

4.4 SELECCIÓN DE INTERRUPTORES DE LÍMITE DE POSICIÓN

Para encontrar el sensor adecuado para una aplicación específica, primeramente se debe de considerar por su tecnología (inductivo o capacitivo), tamaño del empaque, requerimientos de montaje, y características de las señales eléctricas que producen de salida.

Los sensores inductivos son la opción adecuada cuando se reemplazan interruptores mecánicos o en el diseño de maquinaria que requiera de controles de secuencia o sincronización. Pueden sensar sin contacto rodamientos, palancas u otros actuadores metálicos que podrían haber operado interruptores mecánicos. Este tipo de sensores también son usados frecuentemente para sensar el flujo de producción o avance de partes metálicas en la línea.

Los sensores capacitivos pueden detectar la presencia o localización de objetos hechos de casi cualquier material. De cualquier forma, son especialmente útiles cuando los materiales son no metálicos o para determinar información adicional del objeto, como por ejemplo: si una caja se ha llenado adecuadamente con el material de relleno o si el material se encuentra húmedo o seco.

Existen varios tipos, tamaños y formas de encapsulado para estos sensores. Los de tipo tubular son más robustos, sencillos de instalar y de uso relativamente más pesado, se encuentran en una gran variedad de diámetros. Generalmente son fabricados en acero inoxidable o níquel-bronce para asegurar su correcto funcionamiento aún en medios ambientes adversos.

Los requerimientos de voltaje de alimentación y señales de salida para este tipo de sensores pueden ser seleccionados para trabajar tanto con una fuente de alimentación de AC o de DC y la carga externa que controlarán.

Los sensores del tipo DC pueden operar en un rango amplio de voltajes, son compatibles con entradas de DC para microcontroladores programables, contadores u otro tipo de equipo; existen con diseños de salida del tipo NPN o PNP; las salidas pueden conmutar directamente relevadores de DC, solenoides, etc.

Los sensores de AC pueden operar en rangos de 90 a 220 VAC; existen algunos diseños con alta impedancia para conexión directa a microcontroladores programables con entradas que van desde 20 a 250 VAC; son de diseño simple de dos alambres que se conectan en serie con la carga externa y pueden conmutar directamente relevadores de AC, solenoides, etc.

4.5 CONSIDERACIONES ADICIONALES

Existen dos tipos de sensores: con blindaje y sin blindaje, los cuales se observan en la figura 38. Los sensores de proximidad inductivos sin blindaje pueden ser reconocidos por una parte plástica en la punta del sensor que se extiende de la punta y alrededor del tubo metálico. Este tipo de sensores tiene la ventaja de aumentar el campo de sensado hasta en un 50% más del tamaño originalmente dado con respecto a los sensores con blindaje, de cualquier manera, no deberán ser montados de manera que las partes metálicas a sensar se encuentren demasiado cerca de la punta de extensión.

Los sensores con blindaje tienen la cara de detección de flujo directo con el frente del tubo de su material. Pueden ser montados sin preocupación de las superficies del metal que los rodea más que con la consideración de enfocarse directamente al campo a detectar.

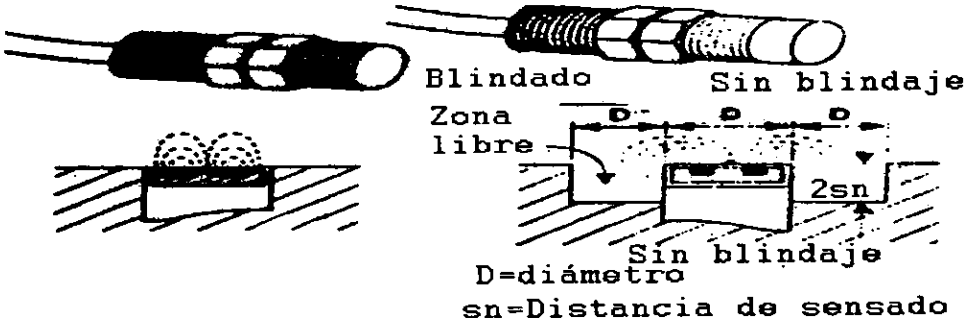


Fig. 38 Especificaciones de sensado

La distancia a detectar representa el espacio máximo de trabajo entre el sensor y el material objetivo y es directamente proporcional al tamaño físico de la unidad, sensores de proximidad más grandes tienen mayor distancia de detección que los más pequeños. La especificación está basada en el material objetivo siendo acero colado, y de igual o mayor tamaño que la cara de detección de los sensores de proximidad. Metales no ferrosos o materiales objetivos de menores dimensiones reducirán la distancia práctica de detección.

Las especificaciones de velocidad de los sensores de proximidad está relacionada con su capacidad para detectar ciclos repetitivos en el material objetivo o el reconocimiento de que el material objetivo está dentro de su campo de detección por períodos de corta duración. Generalmente los sensores alimentados con voltaje de DC son considerablemente más rápidos que las versiones de AC.

Cada día existen nuevas y creativas aplicaciones para los sensores inductivos y capacitivos, entre los cuales se pueden mencionar algunos de ellos:

INDUCTIVOS

Selección o conteo de partes metálicas
Control de partes faltantes
Robótica
Detección de Herramientas rotas
Confirmación de posición de válvulas
Detección de atascamientos

CAPACITIVOS

Partes no metálicas
Control de llenado de líquidos o polvos
Confirmación de contenido en empaques sellados
Detección de redes
Control de derrames o llenado en menor cantidad al embotellar líquidos
Alarma de final de rollo

4.6 SENSORES DE ULTRA PRECISIÓN

Existen también algunos dispositivos que tienen la función de fijar los límites de movimiento de ciertos materiales con contacto o por proximidad. Estos dispositivos son conocidos en el mercado como UPS por sus siglas en inglés (Ultra Precisión Switch) y son similares en apariencia a los sensores de proximidad de forma tubular.

Este tipo de dispositivos pueden ser usados para tareas de medición y control como bombeo, posicionamiento y medición en maquinaria industrial o aplicaciones de Robótica. Tienen una repeticionalidad de 0.00004 pulgadas (0.001 mm), requieren de muy pequeña fuerza para accionarlos y una vida útil de más de 10,000,000 de operaciones.

En el presente trabajo de investigación, los sensores de proximidad son utilizados para definir o establecer las áreas máximas en las que podrá desplazarse el cabezal de la máquina de corte E.D.M. Serán utililzados cinco sensores, cuatro de ellos estarán localizados en cada uno de los cuadrantes imaginarios por los que se desplazará el cabezal y el quinto estará localizado en la posición central. La función de estos sensores será la de informar al microcontrolador de la cercanía del límite máximo de desplazamiento y, en su caso, detener el proceso.

Adicionalmente a su función de detectar la proximidad del límite, estos sensores nos ayudarán a localizar la posición exacta del cabezal al iniciar el corte, en caso de una falla en el sistema, o al iniciar un nuevo corte. La señal proporcionada por uno o varios de los sensores activados informará al microcontrolador de la posición del cabezal en el cuadrante que se encuentre activado y por lo tanto nos permitirán conocer la distancia que debe recorrer el cabezal antes de posicionarse en la localidad deseada.

Capítulo 5

SELECCIÓN DE MICROCONTROLADORES

Con el transcurso del tiempo y con ello el desarrollo de la tecnología, la electrónica ha ido evolucionando rápidamente.

Primeramente, aparecieron unos dispositivos electrónicos llamados "microprocesadores". Estos al principio eran utilizados en calculadoras, sintetizadores musicales, etc., los cuales no requerían de gran capacidad de procesamiento. Pero con el desarrollo de estos dispositivos, se fueron utilizando en equipos más sofisticados y que requerían mayor capacidad de procesamiento como las microcomputadoras.

En general, los microprocesadores son sistemas de cómputo utilizados para procesamiento de información, los cuales están constituidos por una parte física (hardware) y una parte administrativa (software). Cuentan con :

- Unidad Central de Proceso
- Unidad de Memoria
- Unidad de Entrada
- Unidad de Salida
- Dispositivos periféricos: memorias, etc., en general, hardware adicional.

Pero debido a la necesidad de crear equipos cada vez más sofisticados, se tuvieron que construir estos dispositivos cada vez más pequeños, juntando en un sólo "circuito integrado" al microprocesador y a los dispositivos periféricos, llamando a este nuevo circuito "microcontrolador".

Hoy en día, existen una gran cantidad de marcas de microprocesadores y de microcontroladores como: Motorola, Intel, Mitsubishi, Nec, Siemens, Texas Instruments, Zilog, Toshiba, etc..

La selección del microcontrolador es un punto muy importante para el análisis y el desarrollo del proyecto, ya que este dispositivo es el que va a estar controlando la interfaz con la máquina de corte.

Para la elección del microcontrolador se analizaron diversos factores como:

- La capacidad de procesamiento de información.
- La disponibilidad de obtenerlo en el mercado facilmente y a un precio accesible.
- La facilidad de programación.

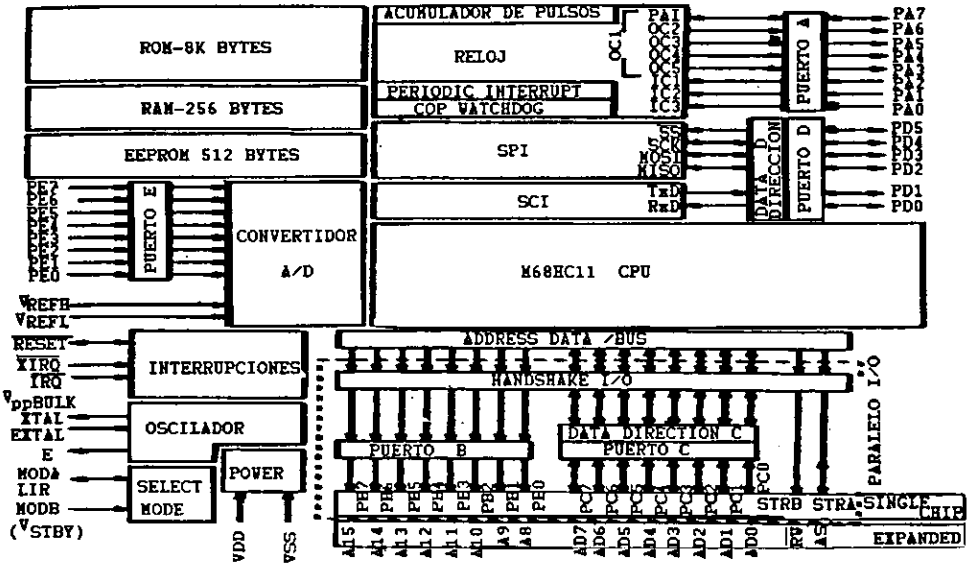
Por estas razones, se decidió utilizar el microcontrolador de Motorola MC68HC11, un microcontrolador de más fácil acceso en el mercado.

Este dispositivo, además del microprocesador contiene memorias y dispositivos de entrada - salida (E/S), además de que pueden tener algunos otros elementos como temporizadores, convertidor A/D, etc.

5.1 MICROCONTROLADOR MC68HC11

El microcontrolador MC68HC11 es uno de los MCU más poderosos en 8 bits. Este integrado fabricado por Motorola con tecnología de alta densidad Metal Óxido Semiconductor Complementaria, tiene como CPU un microprocesador 6800 mejorado. Los códigos de operación tanto para el microprocesador 6800 como para el microcontrolador MC68HC11 son los mismos, sin embargo éste último tiene más registros e instrucciones, puede operar a mayores frecuencias (8 Mhz.) y su registro de código de condición (de 8 bits) permite la operación en modo de espera para así consumir menos energía.

Las figuras 39 y 40 muestran la estructura lógica y fisica del microcontrolador MC68HC11.



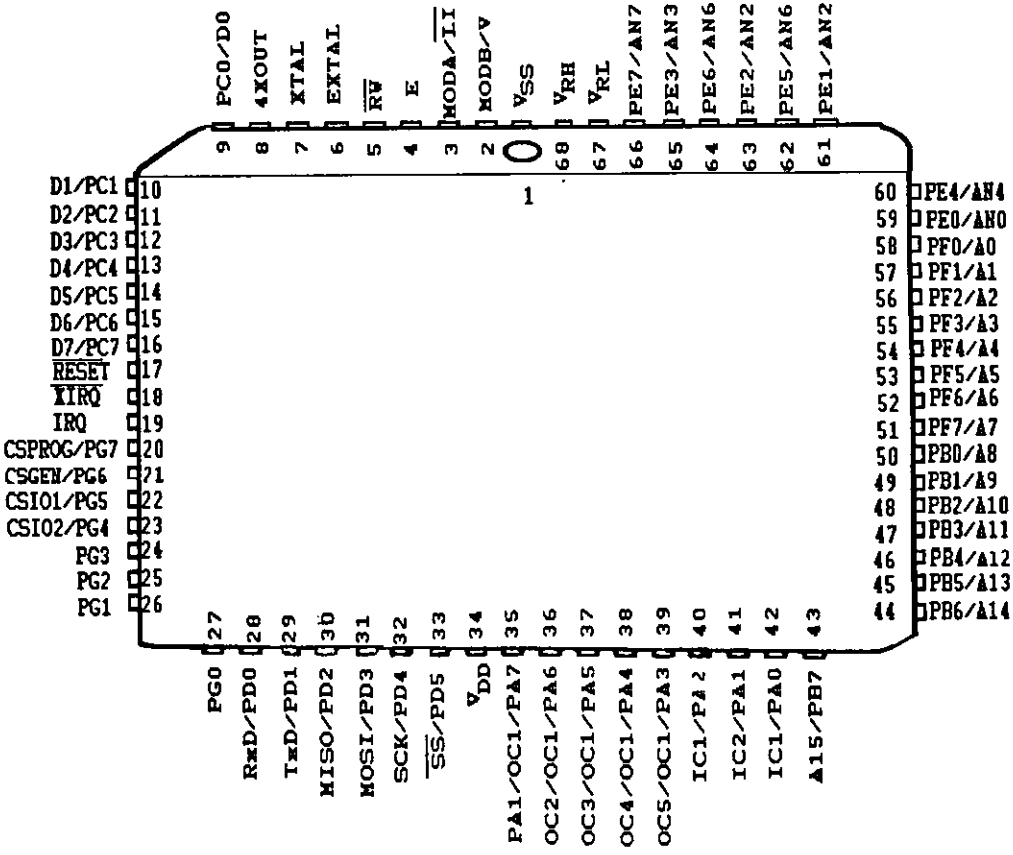


Fig. 39 y 40 Estructura del MC68HC11

5.2 CARACTERISTICAS DEL MICROCONTROLADOR MC68HC11

El microcontrolador 68HC11 tiene las siguientes características:

* Unidad Central de Proceso (CPU)
* Memorias
* Puertos paralelos
* Puertos seriales

* Interrupciones
* Temporizador
* Convertidor analógico - digital

* UNIDAD CENTRAL DE PROCESO (CPU)

El CPU del microcontrolador MC68HC11 ha sido optimizado para bajo consumo de energía y un alto desempeño de operación a frecuencias mayores de 4 Mhz y contiene lo siguiente:

- Acumuladores A y B: son registros de 8 bits cada uno en los cuales se hacen las operaciones aritméticas y se obtienen los resultados. Cuando se unen los dos acumuladores se le llama acumulador D y el registro es de 16 bits.

- Registros de índice X (IX) e Y (IY): son registros de 16 bits cada uno y son utilizados generalmente como referencia (offset) al formar una dirección o como registros temporales de almacenamiento.

- Apuntador de pila (SP): es un registro de 16 bits, el cual contiene la dirección de la pila donde se guardará el siguiente dato al mandarlo a la pila. Al llevar un byte y guardarlo en la pila, el SP se decrementa una unidad de memoria. Al sacar un dato, el SP toma el dato y luego incrementa una localidad.

- Contador de programa (PC): es un registro de 16 bits que contiene la dirección de la siguiente instrucción a ejecutarse.

- Registro de código de condición (CCR): es un registro de 8 bits (banderas) los cuales se muestran a continuación.

La figura 41 muestra la estructura del CPU.

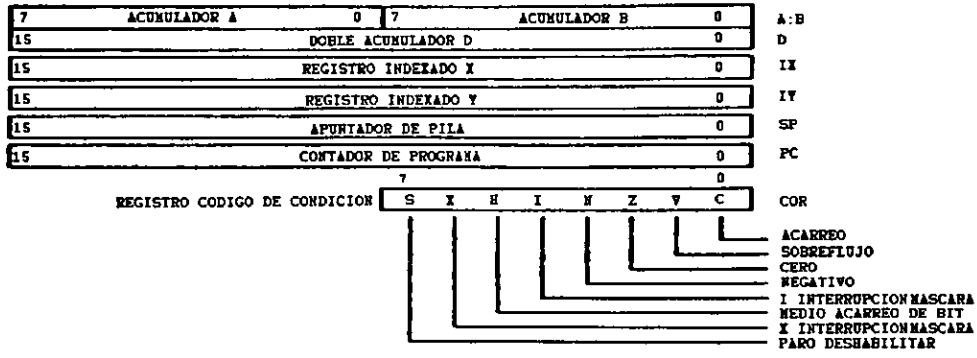


Fig. 41 Estructura del C.P.U.

*** MEMORIAS**

El microcontrolador 68HC11 tiene las siguientes memorias:
- RAM
- ROM
- EEPROM
- Memoria principal (RAM,EPROM o EEPROM)

Dependiendo de la familia del microcontrolador, la memoria varia considerablemente. Las familias disponibles son las series: A, D, E, F, G, K, L, M y P.

*** PUERTOS PARALELOS**

El Microcontrolador HC11 versión F tiene 7 puertos paralelos y son:

- Puerto A: Las líneas PA0-PA7 son entrada – salida (E/S). Todas estas terminales también trabajan para el subsistema temporizador.

- Puerto B: Las líneas PB0-PB7 son salidas (S). En el modo expandido llevan la parte alta del bus de direcciones.

- Puerto C: Las líneas PC0-PC7 son entrada - salida (E/S). En el modo expandido llevan multiplexados el bus de datos con la parte baja del bus de direcciones

- Puerto D: Las líneas PD0-PD5 son entrada - salida (E/S). Llevan la información de datos de control de los puertos serie SPI (Interfaz Periférica Serie) y SCI (Interfaz de Comunicación Serie), el síncrono y el asíncrono.

- Puerto E: Las líneas PE0-PE7 son entradas de propósito general (E). También son las entradas al convertidor analógico - digital.

- Puerto F: Las líneas PF0-PF7 son salidas (S).

- Puerto G: Las líneas PG0-PG7 son entrada - salida (E/S).

Además se tienen las líneas STRA y STRB, terminales de control de los puertos paralelos B y C llamadas líneas strobo, pero en el modo expandido son la línea R/W (lectura/escritura) y AS strobo de dirección que permiten mandar al exterior la orden de lectura - escritura y demultiplexar los buses de datos y de direcciones.

*** PUERTOS SERIALES**

La interface serial esta compuesta por dos partes:

- Interfaz de comunicación serie asíncrona SCI: Este dispositivo está hecho para operar como una interface RS-232, pero con niveles de voltaje TTL y necesita que algún circuito haga el acoplamiento a voltajes para esta norma, tanto en la transmisión (Tx) como en la recepción (Rx).

- Interfaz periférica serie SPI: Este dispositivo es una interfaz síncrona que permite el intercambio de información entre dos sistemas, transmite y recibe al mismo tiempo, uno de ellos hace las veces de amo e inicia todas las operaciones de transferencia de datos.

* INTERRUPCIONES

Una de las características más importantes de este microcontrolador es el gran número de interrupciones con que puede operar, son 18 para todos los bloques, contando con una interrupción de tiempo real. 3 de las interrupciones llegan por terminales externas e interrumpen directamente al microcontrolador, son llamadas igual que las terminales RESET, XIRQ e IRQ.

Existen 4 tipos de RESET:
- Externo: requiere que se ponga la terminal en bajo.
- Encendido: actúa cuando se detecta un voltaje de alimentación menor al normal.
- Falla del COP(computadora operando propiamente): actúa cuando en - determinado tiempo- no se detecta un código de operación legal.
- Falla del monitor del reloj: actúa al fallar el reloj del sistema.

El RESET no guarda registros en la pila y por lo tanto no permite regresar al programa interrumpido.

Al salir de la condición de RESET se lee el estado de 2 terminales externas llamadas MODA y MODB, que determinan los siguientes modos de operación:
- Modo de un sólo chip
- Modo multiplexado expandido
- Modo especial bootstrap
- Modo especial de prueba

El XIRQ es una interrupción no enmascarable. Las interrupciones no enmascarables (NMI) se diseñaron para atender situaciones de emergencia externas, ya que no preguntan por ninguna bandera para poder entrar, sin embargo un interruptor rebotando puede ocasionar que la interrupción se interrumpa a si misma retrasando el programa de servicio de la interrupción.

En cambio, la interrupción IRQ es enmascarable y externa que entra por la terminal IRQ.

* TEMPORIZADOR

Cuenta con un temporizador muy poderoso que puede hacer operaciones de captura de entrada y de comparación de salida, y acumulación de pulsos o tiempos.

Todo el subsistema temporizador opera con el puerto A, que tiene 8 líneas bidireccionales (PA0-PA7).

La base del subsistema es un contador de carrera libre de 16 bits o registro contador temporizador (TCNT) que es manejado por el reloj E del microcontrolador.

En las operaciones de captura de entrada se detecta una transición por una terminal de entrada y se captura en un registro la cuenta que en ese momento tiene el contador, se puede programar para que las transiciones activas sean las positivas, las negativas o ambas, el tiempo queda almacenado en registros llamados de Captura de Entrada del Temporizador (TIC).

Las operaciones de salida se llaman comparaciones porque se realizan cuando se igualan el contador temporizador (TCNT) con alguno de los registros de 16 bits llamados comparación de salida del temporizador (TOC).

* CONVERTIDOR ANALÓGICO - DIGITAL

El convertidor analógico - digital (A/D) del microcontrolador HC11 emplea la técnica de Aproximaciones Sucesivas para la conversión, pero con la peculiaridad que usa una redistribución de carga totalmente capacitiva, lo cual hace innecesario tener un circuito de muestreo y retención a la entrada del convertidor, como es normal en los convertidores convencionales.

El convertidor es de 8 canales de 8 bits con resolución de +/- ½ bits. Las terminales que utiliza son las líneas del puerto E. Se deben de utilizar voltajes de referencia V(RH) y V(RL), de tal forma que la diferencia entre estos esté en el rango de 2.5 - 5 V. El convertidor asigna a V(RH) el valor FF y a V(RL) el valor 00. Si la diferencia es igual a 5 V la resolución será 5/255 V, es decir 19 mV, si la diferencia disminuye se tendrá mejor resolución.

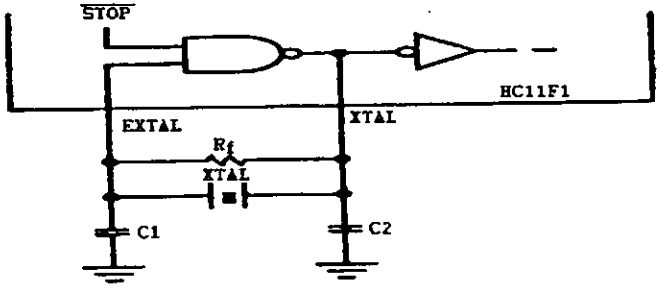
* CARACTERISTICAS ELÉCTRICAS DEL MCU

- Voltaje de alimentación: 5 volts +/- 5%

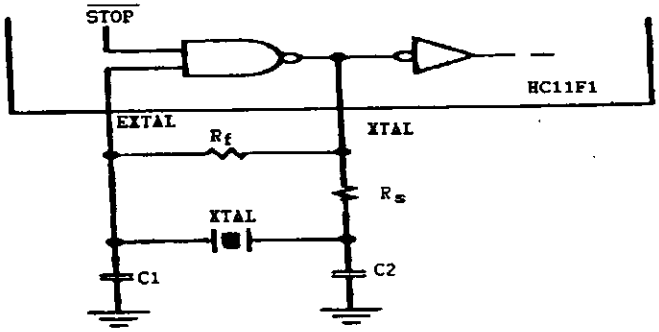
- Temperatura de operación: 0 - 70 °C

- Frecuencia de operación: Hasta 8 Mhz controlado por un cristal de cuarzo

Las líneas XTAL y EXTAL sirven para conectar el cristal que determina la frecuencia de operación del sistema, que es la frecuencia del cristal entre 4. Se observa que XTAL va en la dirección de salida y EXTAL en la de entrada, esto es porque también puede aplicarse una señal externa. LIR carga el registro de instrucción que indica cuando se ejecuta una instrucción y que está pensada para correr programas por pasos y V(stby) que es para dar alimentación a la RAM y que ésta no pierda su información cuando no este presente V(dd). La figura 42 muestra la conexión para obtener diferentes frecuencias de operación.



CONEXIONES CRISTAL DE ALTA FRECUENCIA



CONEXIONES CRISTAL BAJA FRECUENCIA

Fig. 42 Frecuencia de operación

Capitulo 6

CONVERTIDOR D/A

Las señales analógicas son aquellas que están situadas entre los extremos alto y bajo de una fuente de alimentación, estas varían continuamente mientras que las señales digitales asumen solo uno de estos dos niveles bajo o alto.

La figura 44 muestra el diagrama de bloques de un convertidor digital analógico, en forma de diagrama de bloques. Este circuito tiene 4 líneas de entrada, en las cuales pueden ser colocados niveles de voltaje alto o bajo (0s y 1s). El circuito transforma estas combinaciones de alto y bajo en un voltaje analógico equivalente. El valor digital 1111 por ejemplo representara una salida analógica de 15V, mientras que un valor digital 0000 producirá una salida de 0V. Una entrada digital de 0101 nos dará una salida de 5V. A continuación se muestra la forma de onda de salida con sus 16 niveles de resolución donde cada nivel representa 1V.

D	C	B	A	VOLTAJE (V)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

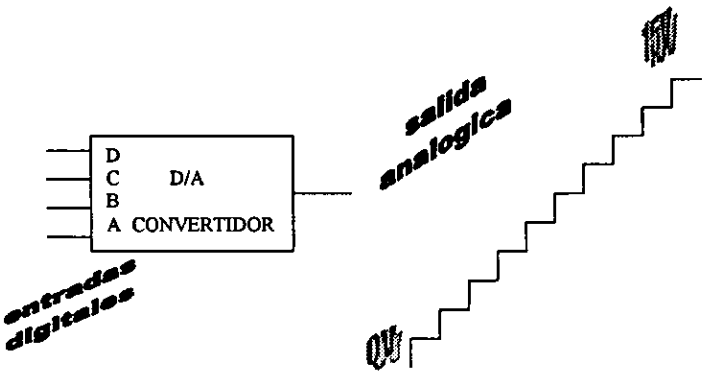


Fig. 44 Señales de entrada y salida en un DAC.

Como elemento básico de un convertidor digital a analógico (DAC) se puede emplear el divisor resistivo o bien la escalera. En la red resistiva tiene lugar realmente la traducción de una señal digital en una tensión analógica. Sin embargo, para completar el diseño del convertidor D/A son necesarios circuitos adicionales.

Como parte integrante del convertidor D/A debe haber un registro que se pueda utilizar para almacenar la información digital. Este registro puede ser de cualquiera de los muchos tipos que se conocen. El registro más sencillo está formado por flip-flops RS, con un flip-flop por bit. También deben haber buffers entre el registro y la red resistiva para que todas las señales digitales presentadas a la red tengan un mismo nivel constante. Finalmente, debe haber alguna forma de control de activación en la entrada de registro para que todos los flip-flops puedan ser ajustados con la información correcta del sistema digital. En la figura 45 está representado un convertidor completo D/A en forma de diagrama de bloques.

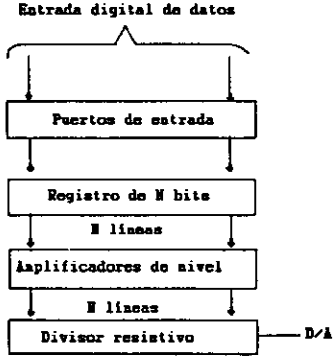


Fig. 45 Diagrama de bloques convertidor D/A

Se puede extender o ampliar el diagrama de bloques representado en la figura 46 se tiene el esquema completo para el convertidor D/A de 4 bits. Se observará que la red de resistores utilizada es del tipo de escalera.

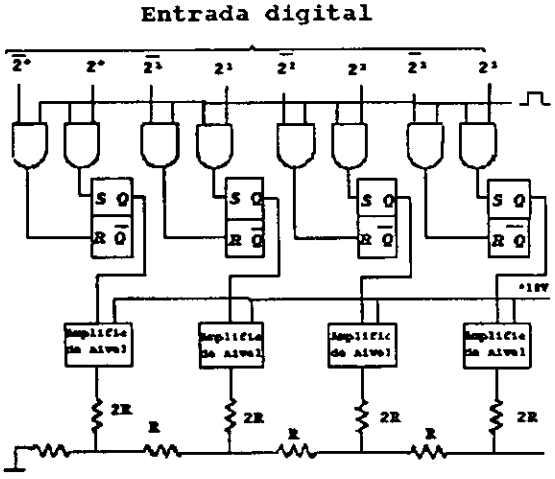


Fig. 46 Convertidor D/A de 4 bits

Cada uno de los amplificadores de nivel tiene dos entradas: una entrada es la de + 10 V desde la fuente de tensión de precisión, y la otra es de un flip-flop. Los amplificadores trabajan de manera tal que cuando la entrada desde un flip-flop es alta, la salida del amplificador es de + 10 V. Cuando la entrada desde el flip-flop es baja, la salida es 0 V.

Los cuatro flip-flops forman el registro necesario para almacenar la información digital. El flip-flop de la derecha representa el MSB (Bit mas significativo), y el flip-flop de la izquierda representa el LSB (Bit menos significativo). Cada flip-flop es un simple cerrojo RS y requiere un nivel positivo en las entradas R o S para su puesta a cero o a uno (reset o set). El esquema para entrar información en el registro es sencillo y puede ser fácilmente comprendido. Con este esquema particular, los flip-flops no necesitan ser puesto a cero (o a uno) cada vez que se entra nueva información. Cuando la línea de ENTRADA DE LECTURA (read in) pasa al nivel alto, sólo una de las dos salidas de puerta conectadas a cada flip-flop es alta, y el flip-flop es puesto a uno o a cero. Así se entran los datos en el registro cada vez que ocurre el impulso (muestreo) de ENTRADA DE LECTURA.

Muy a menudo es necesario descodificar más de una señal; por ejemplo, las coordenadas X e Y para una placa de trazado. En este caso, hay dos maneras para descodificar las señales.

El primer y más obvio método consiste simplemente en utilizar un convertidor D/A para cada señal. Este método, se encuentra representado a continuación, tiene la ventaja de que cada señal que debe ser descodificada está retenida en su registro y entonces se mantiene fija la tensión analógica de salida. Las líneas de entrada digitales son conectadas en paralelo en cada

convertidor. Después es seleccionado el convertidor respectivo para la descodificación por las líneas seleccionadas.

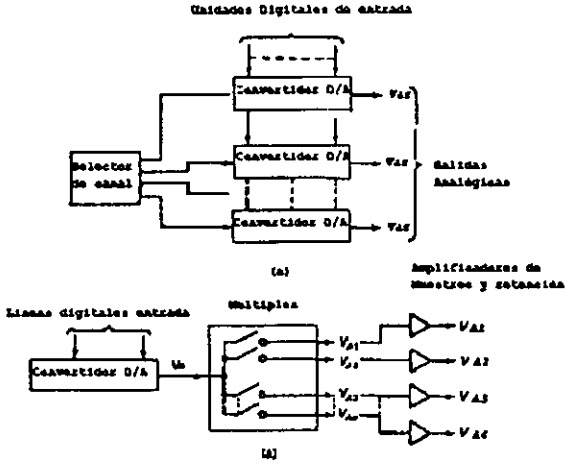


Fig. 47 Descodificación de un numero de señales (a) Método de selección de canal
(b) Método múltiplex

El segundo método implica el uso de sólo un convertidor D/A y la conmutación de su salida. Esto es lo que se llama multiplexado, y el sistema está representado en la figura 48. Aquí el inconveniente es que la señal analógica de salida debe ser retenida entre los períodos de muestreo, y las salidas deben estar, por lo tanto, equipadas con amplificadores de muestra y retención.

Un amplificador operacional (OA) conectado como se muestra en la figura 48 constituye un amplificador de tensión no inversor de ganancia unitaria, es decir, Vsal = V ent. A continuación se muestra como son utilizados tales amplificadores operacionales con un condensador para formar un amplificador de muestreo y retención. Cuando está cerrado el interruptor, el condensador carga al convertidor D/A hasta la tensión de salida. Cuando el interruptor está abierto, el condensador mantiene el nivel de tensión hasta el siguiente tiempo de muestreo. El amplificador operacional provee una gran impedancia de entrada para que no se descargue el condensador apreciablemente y al mismo tiempo proporciona la ganancia necesaria para excitar los circuitos externos.

Cuando se emplea el convertidor D/A conjuntamente con un multiplexor, el máximo ritmo a que puede operar el convertidor debe ser tenido en cuenta. Cada vez que son desplazados los datos en el registro aparecen transitorios en la salida del convertidor. Esto es debido principalmente al hecho de que cada flip-flop tiene tiempos diferentes de subida y caída. Se debe dejar que transcurra un tiempo de ajuste entre el tiempo en que se desplazan los datos en el registro y el tiempo en que se lee la salida de tensión analógica. Este tiempo de ajuste es el factor

principal que determina la salida de máxima relación de multiplexado. El peor caso ocurre cuando cambian todos los bits (por ejemplo, desde 1000 hasta 0111).

Desde luego, los condensadores de los amplificadores de muestreo y retención no son capaces de retener indefinidamente una tensión; por tanto, el ritmo o frecuencia de muestreo debe ser suficiente para asegurar que estas tensiones no decaigan apreciablemente entre los muestreos. La frecuencia de muestreo es una función de la capacidad de los condensadores así como de la frecuencia de la señal analógica que es previsible en la salida del convertidor.

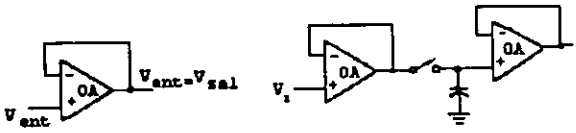


Fig. 48 (a) Amplificador de ganancia unitaria (b) Circuito de muestreo y retención

Dos pruebas sencillas pero importantes que se pueden realizar para comprobar si la operación del convertidor D/A es correcta son la prueba de exactitud en estado estacionario y la prueba de monotonicidad.

La prueba de exactitud en estado estacionario implica el establecimiento en el registro de entrada de un número digital conocido, midiendo la salida analógica con un medidor de precisión y comparándola con el valor teórico.

La prueba de monotonicidad significa verificar que la tensión de salida aumenta regularmente cuando aumenta la señal digital de entrada. Esto se puede efectuar utilizando un contador para la señal digital de entrada y observando la salida analógica en un osciloscopio. Para que la monotonicidad sea correcta, la forma de onda de salida debe ser una escalera perfecta, como muestra en la figura 49.

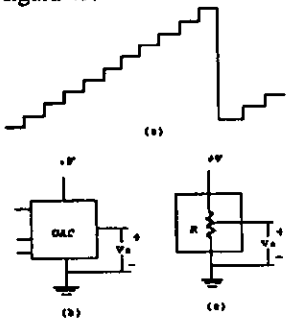


Fig. 49 Forma de onda de tensión de salida correcta para prueba de monotonicidad

Los escalones deben estar igualmente espaciado y ser de la misma amplitud. La falta de escalones, la diferente amplitud de ellos o la discontinuidad hacia abajo de los escalones indican mal funcionamiento.

La prueba de monotonicidad no sirve para verificar la precisión del sistema, pero si éste pasa la prueba, es relativamente seguro que el error del convertidor es menor que 1 LSB.

Se puede considerar a un DAC como un bloque lógico que tiene numerosas entradas digitales y una sola salida analógica, como se ven en la figura 49 b. Es interesante comparar este bloque lógico con el potenciómetro representado en el inciso c de la misma figura. La tensión analógica de salida del DAC se controla por señales digitales de entrada, mientras que la tensión analógica de salida del potenciómetro se controla por la rotación mecánica del eje del potenciómetro. Considerado de esta manera, es fácil ver que se puede utilizar un DAC para generar una forma de onda de tensión (diente de sierra, triangular, senoidal, etc.). Es, en efecto, un generador de tensión digitalmente controlado.

Dentro de nuestro trabajo, el convertidor digital analógico tiene la función de recibir de la computadora los datos de velocidad dados por el operador, procesarlos y convertirlos en un valor analógico el cual sirve como control de los servos en el arranque y movimiento de los motores.

Capítulo 7

DISEÑO Y CONSTRUCCIÓN DE LA INTERFACE CON LA PC

Muchas computadoras proveen de por lo menos un puerto de comunicaciones de propósito general. Este puerto de comunicaciones puede ser usado por algunos dispositivos periféricos externos o como una liga hacia otra computadora.

La computadora, donde reside el programa escrito en "C" para controlar la maquina E.D.M, necesita poder enviar y recibir señales (salidas y entradas de control) mediante algún dispositivo de entrada y/o salida (I/O).

Estos datos o señales digitales (0 – 5 Vcd) se pueden manejar desde el programa usando algún puerto ya definido en la computadora:

Serial .- Los datos son enviados bit por bit, primero se forman en un grupo conocido como frame. Cada frame de datos es precedido por un bit de inicio y después de que el ultimo bit es enviado se manda un bit de paro.

Paralelo.- Se encuentra limitado en número de bits a usar como entradas y como salidas: contiene 12 salidas y 5 entradas, los primeros 8 bits datos se decodifican hacia la dirección 378 hex, el segundo grupo de 4 bits es usado para señales de control y se decodifica hacia la dirección 37 A hex.

Debido al número de entradas y salidas de control necesarias para manejar las señales de la maquina E.D.M, se consideró más adecuado la elaboración de una tarjeta para manejarlas, utilizando las direcciones de algunos puertos fisicos no utilizados.

7.1. SEÑALES A USAR

Para comenzar a construir la interface de comunicación entrada/salida (Input/Output) a la PC se eligió el diseño de una tarjeta basada en la arquitectura del bus tipo ISA de 8 bits, así como utilizar un mapeo de I/O aislado (usando puertos fisicos de I/O) seleccionando un rango de direcciones para nuestra tarjeta.

Al iniciar el diseño, se comenzó por hacer una descripción de que señales nos eran útiles de las que se presentan en el bus ISA al escribir y al leer información de una dirección elegida por nosotros como puerto físico.

Cuando se ejecuta una escritura al puerto físico seleccionado, con el osciloscopio se observó que además de la dirección (AO – A15) y los bits de datos (DO – D7) se presentan la señal IOW.

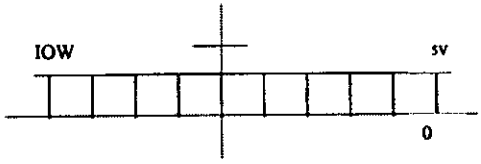


Fig. 50 Señal IOW vista en osciloscopio

De la misma forma al realizar una lectura de algún puerto físico, podemos ver que además de estar presenta la dirección, la computadora captura la información del bus de datos al presentarse la señal IOR.

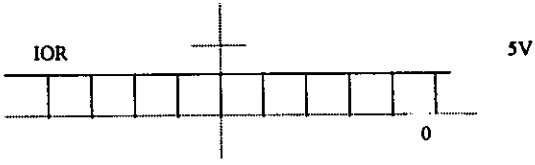


Fig. 51 Señal IOR vista en osciloscopio

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

Con estas dos señales (IOW e IOR), las direcciones (AO –A15), los datos (DO –D7), el voltaje (5 VCD) y la tierra (OV) diseñamos la tarjeta.

7.2 ASIGNACIÓN DE DIRECCIONES DE LOS PUERTOS

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

Se eligió manejar el rango de direcciones $FOOO a $FFFF con el circuito decodificador de direcciones y combinando esto con las señales de lectura (IOR) y escritura (IOW) al bus, designamos 3 direcciones para los dispositivos de salida y 2 direcciones para los dispositivos de entrada .

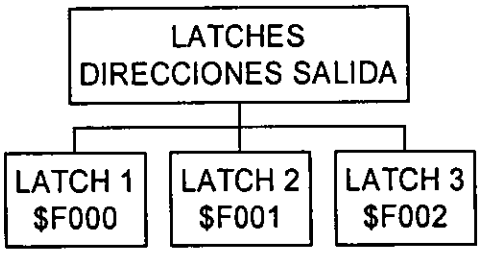


Fig. 52 Direcciones de Salidas

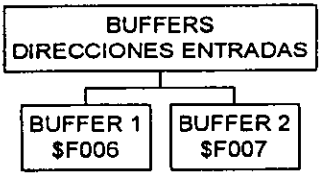


Fig. 53 Direcciones de Entradas

7.3. DECODIFICACIÓN DE DIRECCIONES

Tomando las dirección (AO – A15) y las señales IOR e IOW, usamos dos decodificadores 74HC138 y algunas compuertas NAND (74HC08) para generar las señales que activan los dispositivos de entrada y los de salida, quedando nuestro circuito de decodificación como se muestra en la figura 54.

Las 3 primeras salidas del primer decodificador activan los latches (de L1 a L3) y 2 salidas del segundo decodificador activan los buffers (B1 y B2).

7.4. DISPOSITIVOS DE ENTRADA Y SALIDA

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

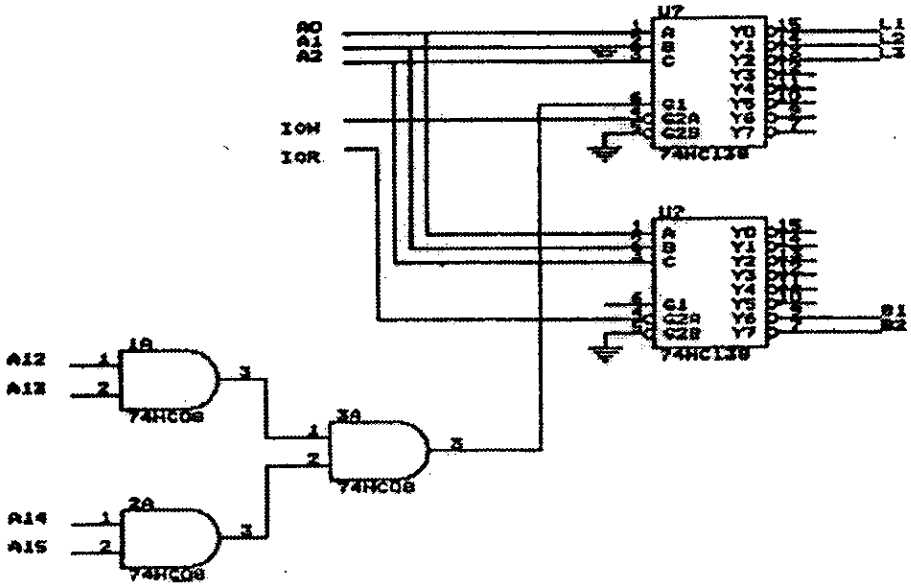


Fig. 54. Circuito decodificador de Direcciones.

Con estos dispositivos se diseño la tarjeta con la siguiente arquitectura muestra en la figura 55

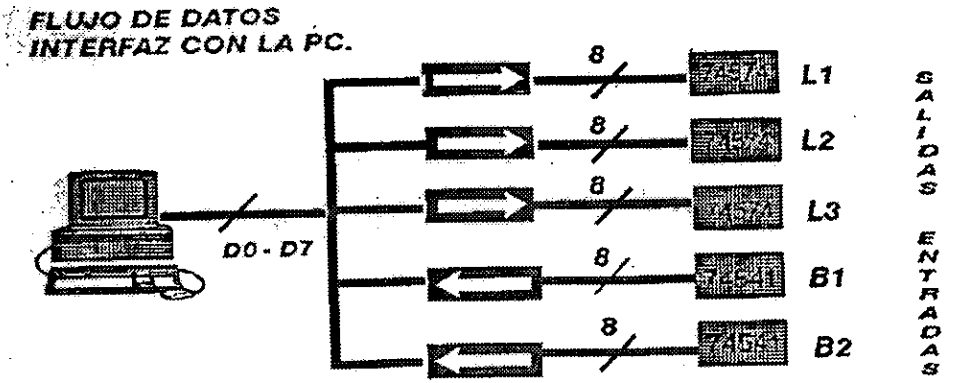


Fig. 55. Flujo de Datos

Finalmente, la tarjetas mediante un cable plano envia en 26 lineas, 24 bits de salidas y 2 con los voltajes (Vcc y tierra), y recibe en otras 16 líneas 16 bits de entradas.

Para mayor detalle de las arquitecturas de ambas interfaces ver el apéndice.

Capítulo 8

DESARROLLO DEL PROGRAMA DE CONTROL

8.1 DESCRIPCIÓN DEL CONTROL

El sistema de control enviará y recibirá señales (escribir y leer de puertos) para que el microcontrolador vía los parámetros capturados de los Encoder, haga un seguimiento de la secuencia de corte, además al mismo tiempo los programas estarán sensando (leyendo los buffers) posibles señales de alarma como: velocidad de corte, hilo roto, falta de agua o que se encienda alguno de los sensores de límite de posición.

A continuación se menciona brevemente lo que el programa hace con cada palabra de control para manejar la máquina de corte:

Para la definición de las palabras de control se utilizó la siguiente nomenclatura:

PC : se estará refiriendo a la computadora personal

MCU : se estará refiriendo al microcontrolador.

P_CONT_MC - Envía señales de control de la PC al MCU en la dirección de puerto 0xF000.

P_DAT_MC – Envía señales de datos de la PC al MCU en la dirección de puerto 0xF001.

P_CONT_PC - Envía señales de control del MCU a la PC en la dirección de puerto 0xF005.

P_DAT_PC – Envía señales de datos del MCU a la PC en la dirección de puerto 0xF006.

CONT_OK_E1 – Palabra de control que manda la PC al MCU, su valor es 0x00F0.

CONT_OK_E2 – Palabra de control que manda la PC al MCU, su valor es 0x0080

CONT_OK_E3 – Palabra de control que manda la PC al MCU, su valor es 0x00FF.

CONT_OK_L1 – Palabra de control que manda el MCU a la PC, su valor es 0x0080.

CONT_OK_L2 – Palabra de control que manda el MCU a la PC, su valor es 0x00F0.

DAT_OK_L – Palabra de control de datos que manda el MCU a la PC, su valor es 0x00F1.

DAT_OK_E - Palabra de control de datos que manda la PC al MCU, su valor es 0x0080.

V_CONT__P_B1 – Palabra de control del primer byte de velocidad + tolerancia, su valor es 0x0001.

V_CONT__P_B2 – Palabra de control del segundo byte de velocidad + tolerancia, su valor es 0x0002.

V_CONT__P_B3 – Palabra de control del tercer byte de velocidad + tolerancia, su valor es 0x0003.

V_CONT__R_B1 – Palabra de control del primer byte de velocidad real, su valor es 0x0004.

V_CONT__R_B2 – Palabra de control del segundo byte de velocidad real, su valor es 0x0005.

V_CONT__R_B3 – Palabra de control del tercer byte de velocidad real, su valor es 0x0006.

V_CONT__N_B1 – Palabra de control del primer byte de velocidad – tolerancia, su valor es 0x0007.

V_CONT__N_B2 – Palabra de control del segundo byte de velocidad – tolerancia, su valor es 0x0008.

V_CONT__N_B3 – Palabra de control del tercer byte de velocidad – tolerancia, su valor es 0x0009.

D_CONT_B1 – Palabra de control del primer byte de desplazamiento (línea a cortar), su valor es 0x000A.

D_CONT_B2 - Palabra de control del segundo byte de desplazamiento (línea a cortar), su valor es 0x000B.

D_CONT_B3 - Palabra de control del tercer byte de desplazamiento (línea a cortar), su valor es 0x000C.

X0_CONT_B1 – Palabra de control del primer byte del punto de origen, su valor es 0x000D.

X0_CONT_B2 - Palabra de control del segundo byte del punto de origen, su valor es 0x000E.

X0_CONT_B3 - Palabra de control del tercer byte del punto de origen, su valor es 0x000F.

P_CONT_B1 – Palabra de control del primer byte de desplazamiento (porcentaje de corte), su valor es 0x0001.

P_CONT_B2 - Palabra de control del segundo byte de desplazamiento (porcentaje de corte), su valor es 0x0002.

P_CONT_B3 - Palabra de control del tercer byte de desplazamiento (porcentaje de corte), su valor es 0x0003.

VEL_OK - Palabra de control que indica que el estado de la velocidad es correcto, su valor es 0x00F5.

VEL_ERR - Palabra de control que indica que el estado de la velocidad es incorrecto, su valor es 0x0004.

ALARMA_OK – Palabra de control que indica que el estado de las alarmas esta correcto, su valor es 0x00FA.

ALARMA_ERR - Palabra de control que indica que el estado de las alarmas esta incorrecto, su valor es 0x0005.

8.2. DESCRIPCIÓN DEL PROGRAMA

El programa fue diseñado para ser operado en cualquier tipo de computadora personal del tipo AT (286, 386, 486, Pentium, etc..) que cuente con un slot tipo ISA disponible para insertar la tarjeta de interfaz diseñada para la captura y salida de información del programa.

La finalidad del programa es que el operador por medio de un mínimo de operaciones y a través de una interfaz gráfica amigable pueda programar una secuencia de corte, que al ejecutarla en la máquina de corte E.D.M vaya siguiendo la ruta programada.

Una de las ventajas de este programa es la de que se puede simular la secuencia antes de ejecutarla, además de que al estarse realizando el corte, el operador visualiza en la pantalla el grado de avance en el corte del material expresado en %.

8.3. ENTRADA AL PROGRAMA

Para entrar al programa nos ubicamos en el símbolo prompt del sistema operativo, después nos cambiamos al directorio donde se encuentra el programa.

C:\> cd edm

Y se corre el programa

C:\edm>corte

Una vez escrito lo anterior se entrara a la pantalla principal del sistema.

8.4. DESCRIPCIÓN DE PANTALLAS

PANTALLA PRINCIPAL

El programa fue diseñado totalmente para un ambiente gráfico por lo que es indispensable que la computadora donde se encuentre cargado el sistema cuente con un Mouse.

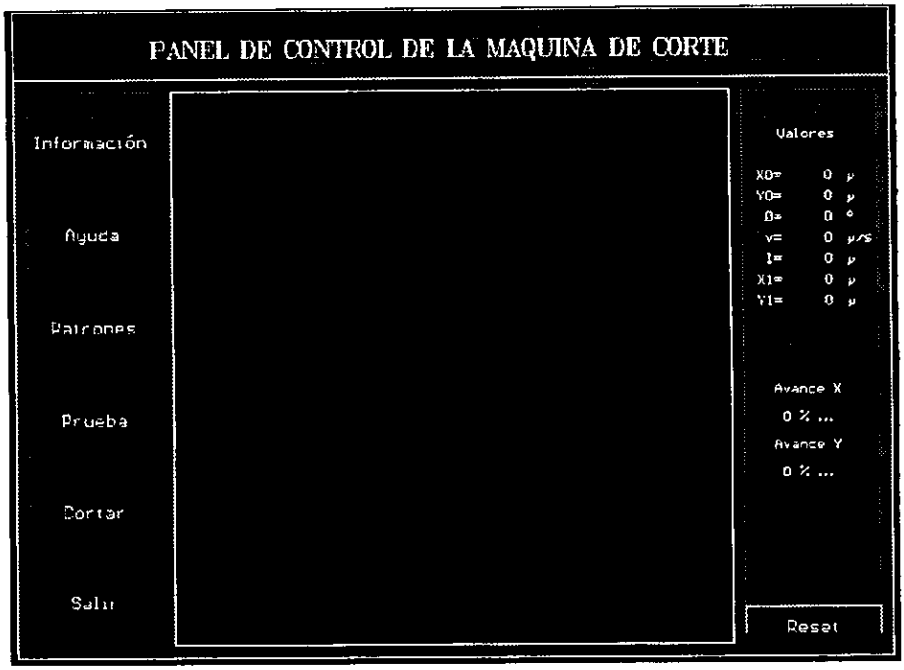
En la parte lateral izquierda de esta pantalla aparecen 6 botones, donde cada uno de los cuales realizan las siguientes funciones.

Información.- Al darle un clic en este botón se encuentran los datos referentes a la ficha técnica de la máquina de control numérico (marca, dimensiones, modelo, número de ejes) así como la de los demás elementos del sistema; servos, características del electrodo, capacidad de la cuba del dieléctrico, etc.

Ayuda.- En esta opción se encuentra la ayuda en línea sobre todas las funciones del sistema de control.

Patrones.- En esta opción se introducen al sistema los valores de: longitud, velocidad, origen y ángulo de la recta para realizar el corte del material.

Prueba.- Dentro de esta opción se verifica las condiciones iniciales de la máquina de corte.



Cortar.- En esta opción se empieza a realizar el corte del material, mostrando en la pantalla la información de el porcentaje de avance del corte.

Salir.- Al elegir esta opción se interrumpe la ejecución del programa, se tiene la opción de abandonarlo y regresar al prompt del sistema operativo.

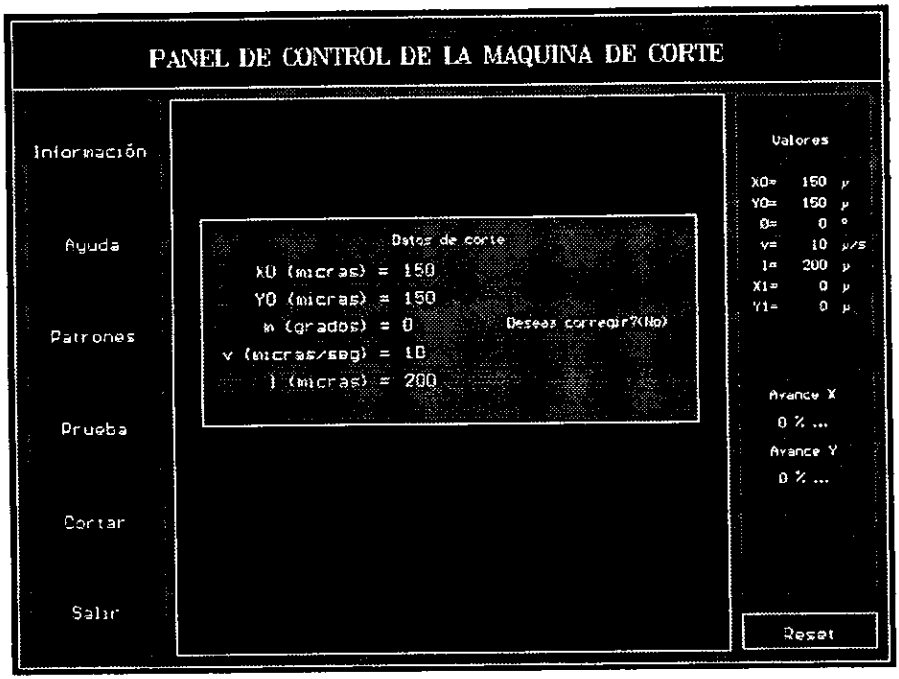
PANTALLA DE PARÁMETROS

A esta pantalla se llega al seleccionar la opción de patrones, la mesa de corte es el área donde se monta el material a cortar, esta mesa es simulada en la pantalla, y tiene una área de trabajo establecida de 4cm. x 4cm. Nuestra referencia de origen (x=0,y=0) se encuentra en la parte inferior izquierda del área de trabajo.

Aquí debemos de introducir los siguientes valores:

Xo.- Este valor representa el valor (expresado en micras) de la coordenada del plano horizontal, a partir de la cual empezará el corte.

Yo.- Este valor representa el valor (expresado en micras) de la coordenada del plano vertical, a partir de la cual empezará el corte.

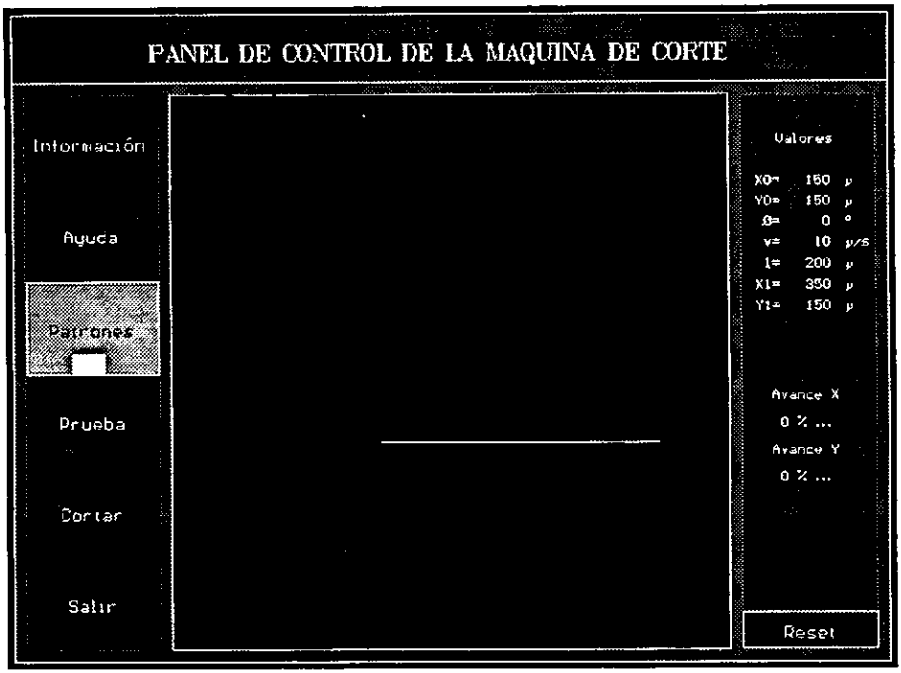


m.- Es el valor del ángulo de inclinación.

v. – Representa la velocidad del corte dada por el operador, la cual permanece constante a través de toda la secuencia de corte, este valor de velocidad es convertido por el programa a un número binario de 3 bytes el cual es almacenado en la memoria interna del microcontrolador.

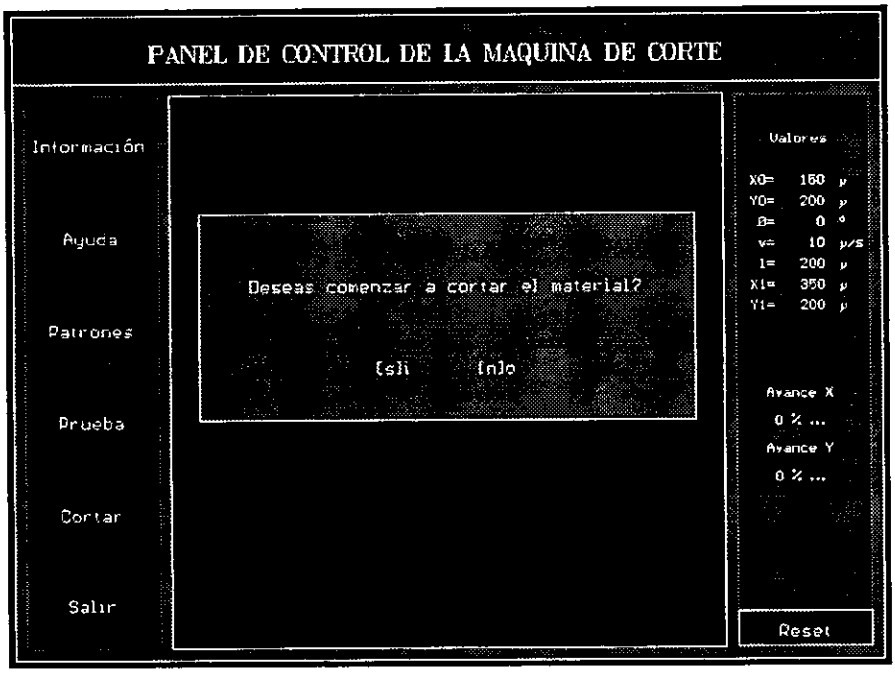
l. – Este valor es dado por el operador y representa la longitud del corte (expresado en micras), el programa convierte este valor a un número binario formado por 3 bytes el cual es almacenado en localidades de memoria definidas del microcontrolador. Para que el operador visualice en pantalla la recta a cortar se tomó como base que 1 milímetro sea igual a 1 micra.

Una vez que el operador ha terminado de ingresar los datos anteriores, se nos pregunta si los datos están correctos o no, si decimos que si (tecleando una S) el programa nos dibujará en pantalla la recta a cortar dados nuestros parámetros, si no se nos vuelve a pedir que introduzcamos nuevamente los datos.



Ya estando en pantalla la recta a cortar, se observa que nuestros parámetros de corte aparecen anotados en la parte derecha de la pantalla, incluyendo además las coordenadas finales de la secuencia.

Por ultimo con la opción **cortar** se ejecuta la acción.



Una vez que la secuencia de corte empieza se podrá ver desplegado en la pantalla el nivel de avance, esta se realiza a través de la siguiente lógica: el operador ingresa una cantidad numérica de micras a cortar la cual es convertida y almacenada en un numero de 3 bytes, este valor va a ser enviado al microcontrolador donde permanentemente va a ser comparado con él numero de 3 bytes que se forma del conteo de pulsos del Encoder.

8.5. DISEÑO DEL SISTEMA DE PROGRAMACIÓN

8.6. CONTROL NUMÉRICO COMPUTARIZADO (CNC)

El control numérico (NC) es el término utilizado para describir el control de movimientos y otras funciones de una máquina por medio de instrucciones expresadas como series de números e inicializadas por un sistema de control electrónico.

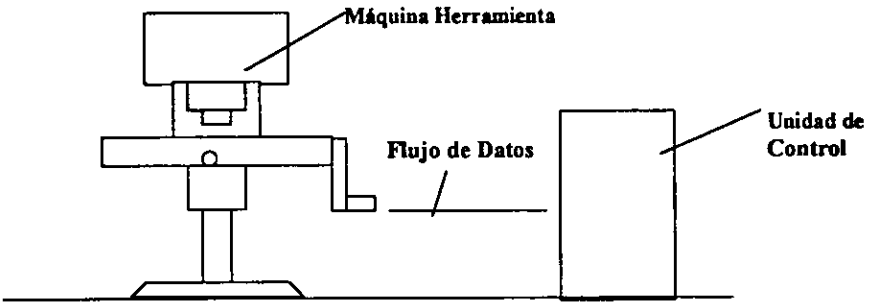


Fig. 56. Sistema de Control Básico: Control numérico

El control numérico computarizado es el término que se utiliza cuando el sistema de control lo realiza una computadora.

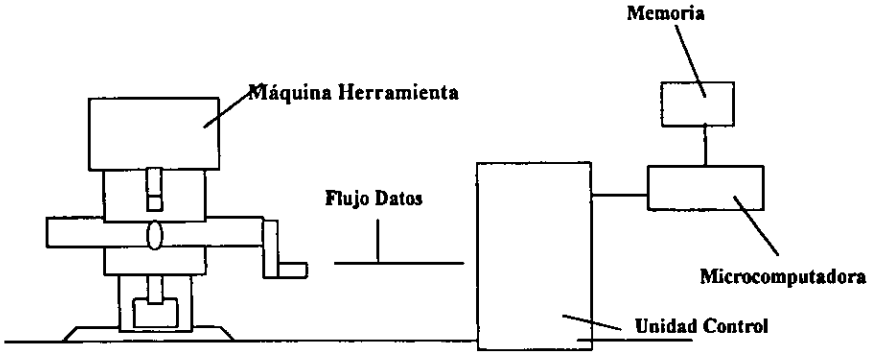


Fig. 57. Sistema de Control Básico: Control Numérico Computarizado.

Al usar una computadora en nuestro control se obtienen los siguiente beneficios: almacenamiento de programas adicionales, edición de programas, ejecución de programas desde memoria, realizar diagnósticos sobre el estado de control de la máquina, algunas rutinas especiales. La unidad de control puede estar fuera o dentro de la estructura principal de la máquina.

El control numérico por medio de una computadora es aplicado en un amplio rango de procesos de manufactura como: cortadoras de metales, fresadoras, tornos, maquinado por descargas eléctricas. Además el control numérico es muy económico para producciones en masa,

lote y en muchos casos en producción simple, muchos factores contribuyen a esta viabilidad económica entre los más importantes tenemos:

- Altos rangos de productividad.
- Uniformidad en el producto.
- Reducción en el rechazo de componentes.
- Menor dependencia del operador.

El control de máquinas herramientas mediante esta vía nos ofrece la capacidad de trabajar casi todas las horas del día virtualmente sin supervisión. Cada función tradicionalmente realizada por el operador puede ser archivada en un programa de maquinado CNC.

El CNC utiliza el método de programación por direccionamiento de palabra el cual está ampliamente basado en códigos dados por el standard de la International Standard Organization (ISO) y la Electronic Industries Association (EIA), donde los programas deberán ser compilados usando códigos identificados por letras, en particular "G" y "M". Cada código direccionado es aplicado anteponiéndose al dato e indica la realización de cierta función dentro del sistema de control.

Las normas ISO Y EIA contienen 99 códigos "G" y 99 códigos "M", en donde primero va la letra y después los dos dígitos. Los códigos "G" son funciones predeterminadas y son usados para configurar el modo de operación requerido por la unidad de control de máquinas herramientas para conducir el maquinado, si el movimiento debe ser recto/lineal, o radial/circular por ejemplo. En general estos códigos se relacionan con el control de movimientos del eje. A continuación unos ejemplos más comunes de códigos "G".

G00	Posicionamiento lineal rápido
G01	Posicionamiento lineal
G02	Interpolación circular, sentido horario
G03	Interpolación circular, sentido antihorario
G33	Enroscar cortador, guía constante
G41	Compensación a la izquierda
G42	Compensación a la derecha
G70	Programación en pulgadas
G71	Programación métrica

Los códigos "M" o funciones misceláneas, se usan para establecer otros requerimientos que están relacionados con el movimiento. Por ejemplo si se usa un eje de movimiento activo o suministran líquido enfriador. Ejemplos más comunes de códigos "M".

M00	Paro de programa
M01	Paro opcional
M02	Fin de programa
M03	Eje en sentido horario
M04	Eje en sentido antihorario
M05	Eje apagado
M06	Cambiar herramienta

M08	Enfriador encendido
M09	Enfriador apagado
M30	Fin de cinta

Al recurrir a una computadora personal como el elemento principal de nuestro sistema control, contamos con grandes ventajas:

Libertad de elección de un lenguaje de programación adecuado para manejar la comunicación con los dispositivos de I/O, manipulación sencilla de los datos entre los componentes del sistema, realizar una interfaz amigable con el usuario (operador) y tener compatibilidad con otros sistemas, entre otras cosas.

La máquina E.D.M sobre la que se diseñó el sistema, usaba los llamados códigos "G" para una secuencia de programación (programada para la pieza a elaborar) que constaba de una serie de comandos dentro de los cuales en cada uno se indicaban que movimientos o funciones se iban a realizar.

En la mayoría de estas máquinas E.D.M de esta generación, las secuencias de corte se almacenaban en cinta, para dado el momento de reproducción de determinado corte, recuperar estos programas y ejecutarlos.

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

Las E.D.M tienen movimientos comunes para su operación, algunos tienen funciones de más u otras de menos pero su funcionamiento o secuencias de operación son similares, éstos nos ayuda en el diseño de nuestro programa de control.

8.7. ELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN

En la actualidad existen en el mercado varios lenguajes de programación cada uno de ellos con sus propias características, se consideró entre los más adecuados para el desarrollo del programa a Pascal y 'C' debido a que son muy flexibles y amplios a la vez y se puede programar con ellos tanto la parte de comunicación a la tarjeta de interfaz como la interfaz con el usuario.

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

8.8. ACCESO AL BUS DE DATOS Y MANEJO DE INFORMACIÓN.

La tarjeta de interface en la PC realiza la acción de enviar y recibir información del y hacia el bus de datos cuando están presentes las señales IOW e IOR en el bus de control, tales tareas las realizan las funciones inport (), inportb(), outport() y outportb() disponibles en compiladores del lenguaje `C'

FUNCIONES PARA LEER Y ESCRIBIR A PUERTOS FÍSICOS

Int inport (int portid)	Lee una palabra del puerto indicado
Unsigned char inport (int portid)	Lee un byte del puerto indicado
Void outport (int portid, int value)	Escribe una palabra al puerto indicado
Void outportb (int portid, unsigned char value)	Escribe un byte al puerto indicado

Los datos (información de control) los agrupamos en bytes de control (8 bits) ya que las tarjetas manejan 8 bits a la vez, estos 8 bits podemos representarlos en distintos tipos de datos manejados por el lenguaje elegido.

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

8.9. ESCRITURA A UN PUERTO FÍSICO

Para escribir información a un puerto fisico (esto significa establecer ciertos valores de información en un instante dado en el bus de datos y generar la señal IOW) debemos poner la información deseada en un tipo de dato y mandarla a la dirección de ese puerto, por ejemplo, si tenemos un dato de longitud de un byte y deseamos escribir unos (1´s) al puerto con la dirección F000 en hexadecimal hay que realizar la siguiente secuencia:

```
Unsigned char dato_sal          /* declaración de la variable para poner el dato */

Dato sal= OxFF;                 /* pone 1´s en los 8 bits del dato */

Outportb ($F000,dato_sal);          /*escribe 1´s al puerto F000 +/
```

8.10. LECTURA DE UN PUERTO FÍSICO

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

```
Unsigned char dato_ent          /* declaración de la variable para guardar el dato */

Dato_ent=inportb ($FF05);       /* lee el byte del puerto FF05 y lo deja en la variable
                                   dato_ent */
```

8.11. INTERFACES DE CONTROL

Para lograr nuestro objetivo de realizar el control sobre una máquina E.D.M fue necesario el diseño de una interface (tarjeta) que acople las señales de entrada y salida de control con la PC, así como la configuración y programación de la tarjeta EVBU (tarjeta de evaluación universal), la cual esta conformada por el microcontrolador M68HC11F1 y circuiteria adicional, todo esto manejado por el programa de control escrito en "C".

8.12. DESARROLLO DE PROGRAMAS DE PROTOCOLOS DE COMUNICACIÓN

8.12.1 PROGRAMA PARA COMUNICAR LA COMPUTADORA PERSONAL (PC) CON EL MICROCONTROLADOR

En esta parte se definirá el procedimiento a seguir para la obtención y almacenamiento de los datos que el microcontrolador recibe de la PC.

Primeramente se considera que deben de existir 8 buses, 2 de datos y 2 de control para el eje horizontal , y 2 de datos y 2 de control para el eje vertical, lo cual se propone de esta forma para que el microcontrolador pueda leer la palabra de control e inmediatamente el dato que corresponde a esta palabra, evitando así utilizar una comunicación sincrona. Para este efecto

utilizaremos de la PC las direcciones F000 y F006, conectadas a los puertos E y B del microcontrolador, como salida y entrada de palabras de control ,respectivamente. Así mismo a las direcciones F001 y F007 conectadas a los puertos G y F ,como salida y entrada de datos, respectivamente. Todo lo anteriormente expuesto es para el eje horizontal, para el eje vertical se tomarán de la PC las direcciones F002 y F008 para palabras de control, y F003 y F009 para datos; las conexiones a los puertos del segundo microcontrolador que censará el eje vertical son iguales puertos E y B para control y los puertos G y F para datos.

En este programa se tomará en cuenta que la PC y el microcontrolador deben de efectuar una prueba de comunicación entre ellos en donde la PC envía una palabra de control, que será $F0, por la dirección F000 al puerto E del microcontrolador, y donde el microcontrolador le contestará con un $80 por su puerto B a la dirección F006 de la PC, cuando la palabra que recibió sea la correcta. En caso contrario la PC y el microcontrolador deberán esperar un determinado tiempo para establecer la comunicación, y en caso de no hacerlo en el tiempo previsto se despliega un mensaje de error en la PC y el microcontrolador para su programa.

Una vez establecida la comunicación el microcontrolador deberá de leer del puerto E la palabra de control que le indique el dato a recibir, (primero recibirá los datos de velocidad (9 bytes), después longitud a cortar (3 bytes) y por último origen para el eje horizontal (3 bytes), el segundo microcontrolador recibirá los datos que corresponden al eje vertical) si la palabra de control es correcta procede a contestarle a la PC con un $F0, por el puerto B, que le indica que puede enviar el dato, procediendo a leer el dato que se encuentra en el puerto G y a almacenarlo en una localidad de memoria, enviándole a la PC, por el puerto B, un $F1 que le indicará que fue leído el dato, continuando así hasta leer los 15 bytes que contienen todos los datos.

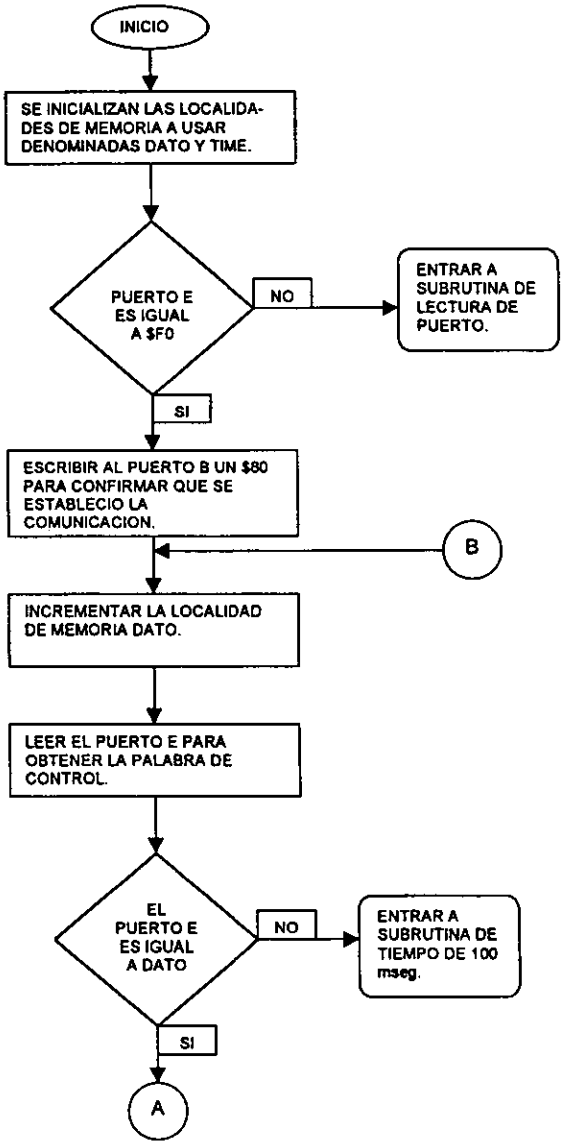
Si la palabra de control fuera incorrecta el microcontrolador deberá de entrar a una rutina de tiempo, de donde saldrá cuando la palabra de control sea correcta (continuando con la secuencia antes descrita) o cuando transcurra un periodo de 100 ms., en este caso el MCU detendrá el funcionamiento de la máquina de corte y enviará un mensaje de error de comunicación, un $CC, a la PC, el cual es equivalente a un mensaje de error de comunicación el cual será observado en el monitor.

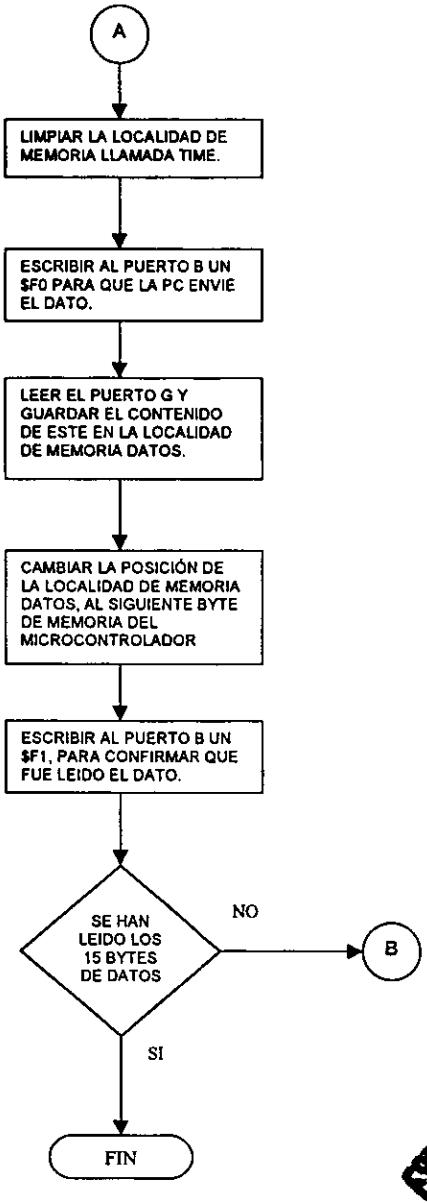
A continuación se listan los códigos que utilizarán la PC y el microcontrolador y que se manejarán en este programa.

- $F0; mensaje que envía la PC al microcontrolador para establecer la comunicación.
- $01-$0F; palabras de control que envía la PC al microcontrolador con los datos de velocidad, longitud de corte y origen.
- $01;primer byte de velocidad máxima.
- $02;segundo byte de velocidad máxima.
- $03;tercer byte de velocidad máxima.
- $04;primer byte de velocidad exacta.
- $05;segundo byte de velocidad exacta.
- $06;tercer byte de velocidad exacta.
- $07;primer byte de velocidad mínima.

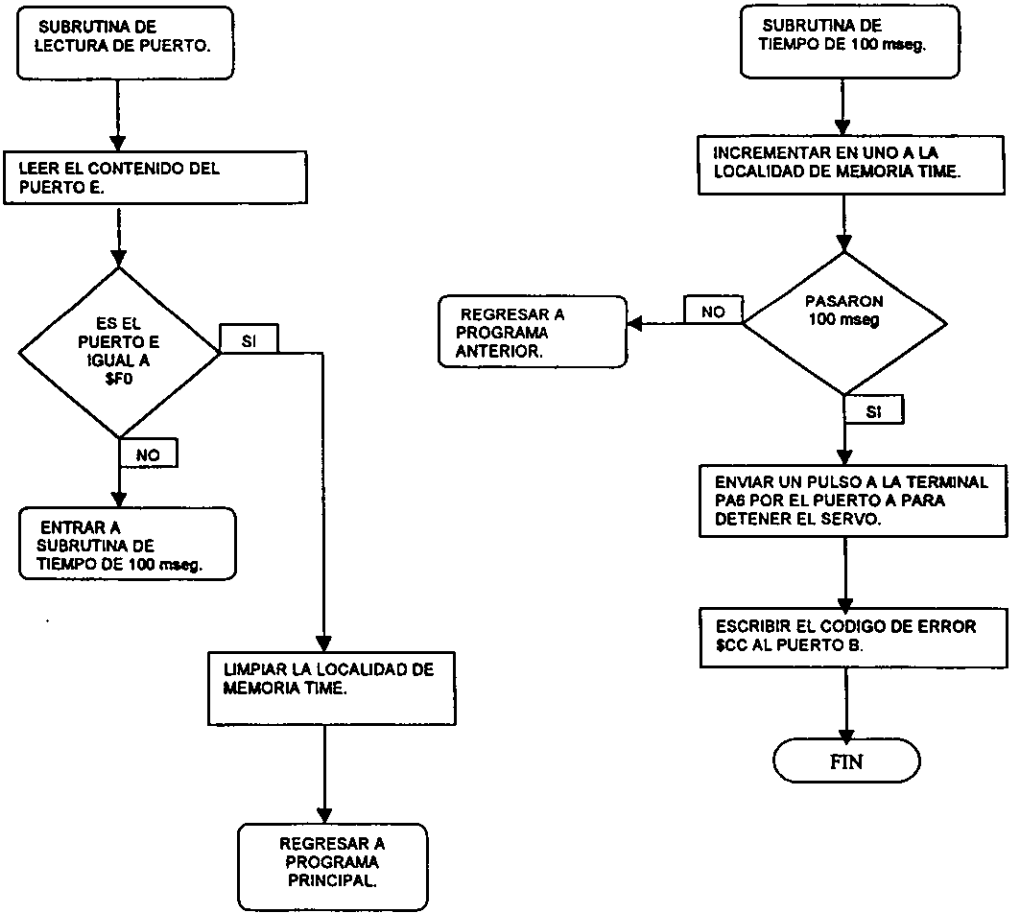
- $08;segundo byte de velocidad mínima.
- $09;tercer byte de velocidad mínima.
- $0A; primer byte de longitud de corte.
- $0B; segundo byte de longitud de corte.
- $0C; tercer byte de longitud de corte.
- $0D;primer byte de origen.
- $0E;segundo byte de origen.
- $0F;tercer byte de origen.
- $80; mensaje del microcontrolador a la PC que indica que existe comunicación.
- $F0; mensaje del microcontrolador a la PC que indica que fue leída la palabra de control.
- $F1; mensaje del microcontrolador a la PC que indica que fue leído el dato.
- $CC; mensaje del microcontrolador a la PC que indica error en la comunicación.

A continuación se presenta el diagrama de flujo para esta parte. El programa se encuentra en el Apéndice 2.


INICIO
SE INICIALIZAN LAS LOCALIDADES DE MEMORIA A USAR DENOMINADAS DATO Y TIME.
PUERTO E ES IGUAL A $F0
NO
ENTRAR A SUBRUTINA DE LECTURA DE PUERTO.
SI
ESCRIBIR AL PUERTO B UN $80 PARA CONFIRMAR QUE SE ESTABLECIO LA COMUNICACION.
B
INCREMENTAR LA LOCALIDAD DE MEMORIA DATO.
LEER EL PUERTO E PARA OBTENER LA PALABRA DE CONTROL.
EL PUERTO E ES IGUAL A DATO
NO
ENTRAR A SUBRUTINA DE TIEMPO DE 100 mseg.
SI
A



8.12.2 SUBRUTINAS DE TIEMPO Y LECTURA DE PUERTO PARA EL PROGRAMA DE COMUNICACIÓN DE LA COMPUTADORA PERSONAL (PC) A EL MICROCONTROLADOR



8.12.3 PROGRAMA PARA ESTABLECER LA COMUNICACIÓN DE EL MICROCONTROLADOR A LA COMPUTADORA PERSONAL (PC)

En esta parte se definirá el procedimiento a seguir para la obtención y almacenamiento de los datos que la PC recibe del microcontrolador.

Como se explicó en el programa anterior, se procederá a establecer la prueba de comunicación entre el microcontrolador y la PC, solo que en esta ocasión el microcontrolador será quien inicie el proceso de comunicación enviando un $F0, como palabra de control, por su puerto B y esperando recibir un $80, por su puerto E ,como contestación por parte de la PC, para proceder a enviar los datos. En el caso de que no recibir la palabra correcta, el microcontrolador entrará en una rutina de tiempo saliendo de esta hasta que le sea enviada la palabra de control correcta, o pasen 100 mseg. En este caso se detendrá el funcionamiento de la máquina de corte y se enviará un mensaje de error de comunicación, $CC como se había mencionado en secciones anteriores.

Una vez establecida nuevamente la comunicación, el microcontrolador deberá enviar la posición actual de la máquina de corte contenida en 3 bytes, enviando primero la palabra de control por el puerto B, que corresponde al primer byte de posición, esperando que la PC le conteste por el puerto E con un $FF. Una vez recibido el código $FF el microcontrolador enviará el dato por el puerto F, la PC envía al microcontrolador un $80, como palabra de control, por el puerto E cuando a recibido el dato. Este procedimiento se sigue para los otros dos bytes de posición, este programa realizará lo anterior con ayuda de una localidad de memoria denominada NUMDATO, que le indica al microcontrolador que tipo de palabra de control debe enviar por su puerto B.

Cuando se ha terminado de enviar la posición, el microcontrolador envía el estado de la velocidad; si esta no se altera enviará un $F5 a la PC, como palabra de control por el puerto B, esperando el $FF que la PC le envíe para continuar con el programa. En caso de existir error en la velocidad el microcontrolador enviará un $04 como palabra de control, para así indicarle a la PC que le será enviado un dato de error de velocidad por el puerto B, a lo cual la PC contestará con un $80.

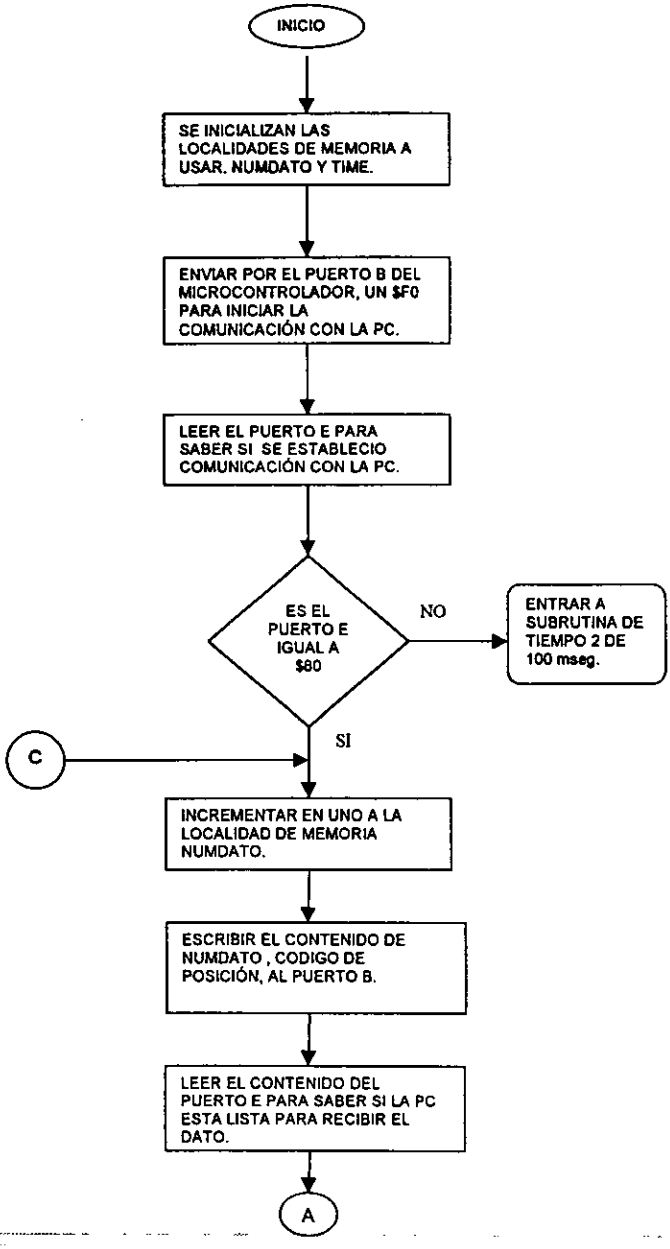
El siguiente paso, una vez que se ha confirmado que la velocidad es correcta se debe de ver la condición de alarmas (interruptores de limite de posición, de hilo roto, de nivel del agua de la bomba), el microcontrolador envía a la PC un $FA cuando no existe problema, y uno de los códigos que se exponen en el programa de alarmas. Cuando alguna de estas alarmas se activa, se envía un dato por el puerto F donde indica que tipo de alarma que fue activada. Debe de mencionarse que el microcontrolador espera como contestación de la PC, un $FF cuando el microcontrolador le envío una palabra de control y un $80 cuando lo que le envío fue un dato.

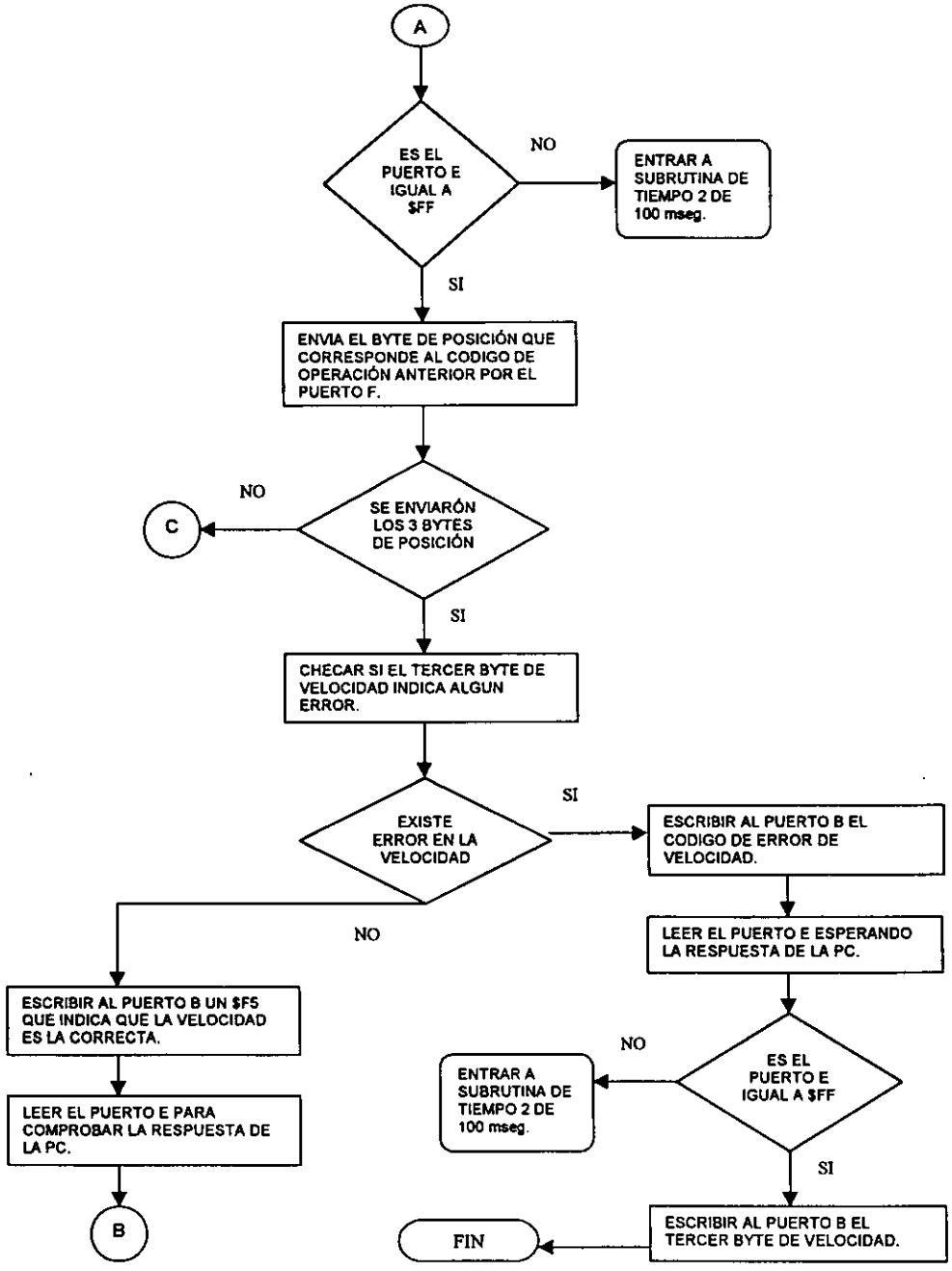
Este programa lo continuará realizando el microcontrolador hasta terminar el corte. A continuación se presentan los códigos a utilizar en este programa.

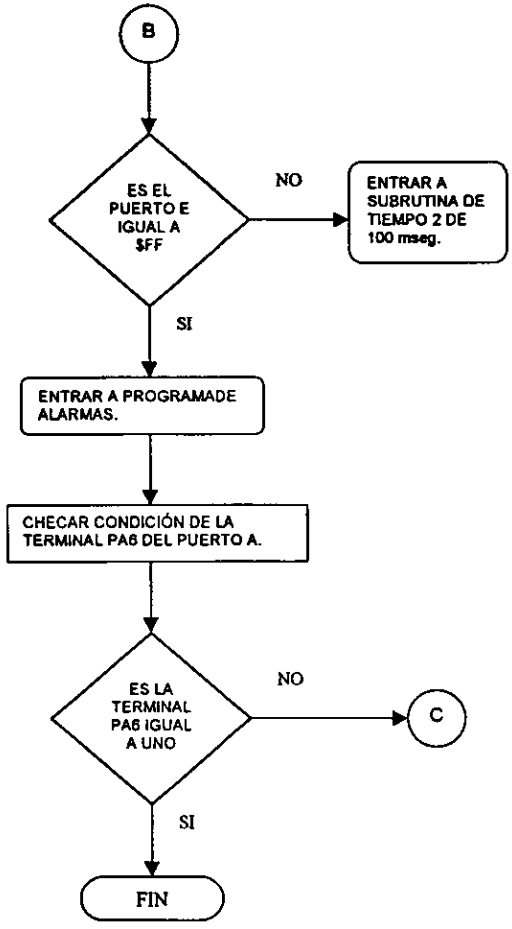
- $F0; mensaje que envía el microcontrolador a la PC para establecer la comunicación.
- $01-$03; palabras de control que envía el microcontrolador a la PC indicando posición.

- $01;primer byte de posición.
- $02;segundo byte de posición.
- $03;tercer byte de posición.
- $80; mensaje de la PC a el microcontrolador que indica que existe comunicación o que la PC leyó un dato .
- $FF; mensaje de la PC al microcontrolador que fue leída la palabra de control.
- $F5; mensaje del microcontrolador a la PC que indica que la velocidad es correcta.
- $FA; mensaje del microcontrolador a la PC que indica que las alarmas no se han activado.
- $04; mensaje del microcontrolador a la PC que indica error en la velocidad.
- $05; mensaje del microcontrolador a la PC que indica que alguna alarma esta activada.
- $CC; mensaje del microcontrolador a la PC que indica error en la comunicación.

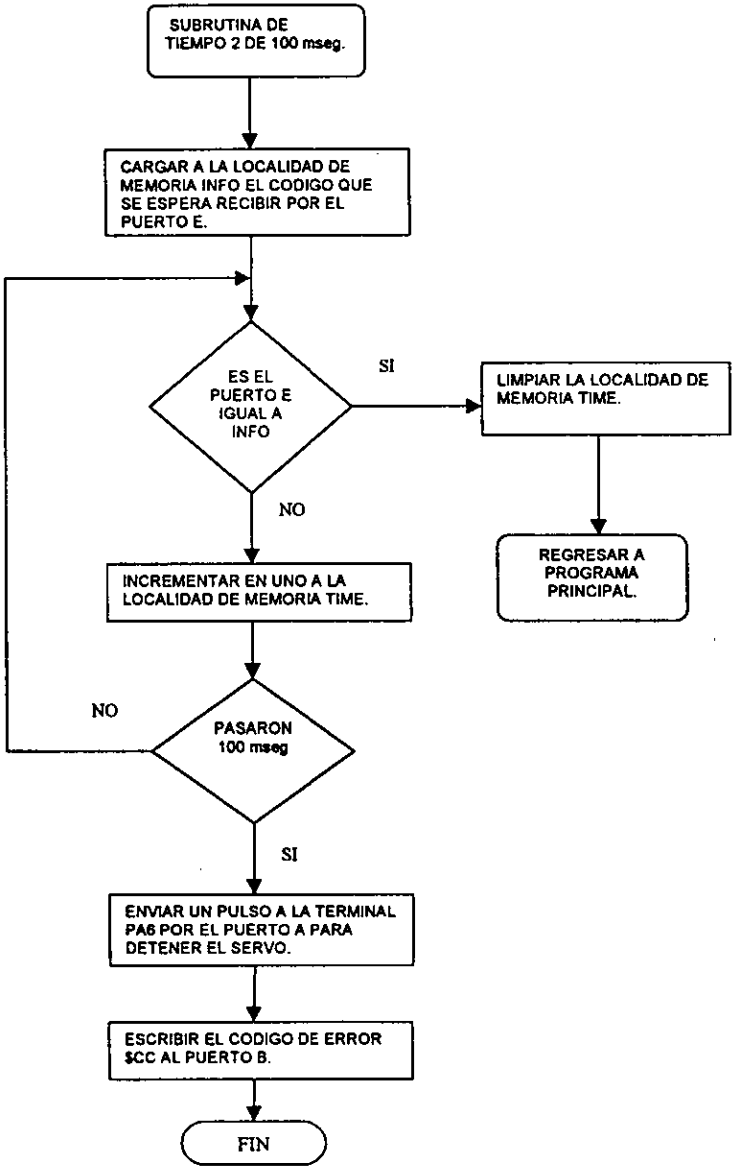
A continuación se presenta el diagrama de flujo para esta parte. El programa se encuentra en el Apéndice 2.




A
ES EL PUERTO E IGUAL A $FF
NO
ENTRAR A SUBRUTINA DE TIEMPO 2 DE 100 mseg.
SI
ENVIA EL BYTE DE POSICIÓN QUE CORRESPONDE AL CODIGO DE OPERACIÓN ANTERIOR POR EL PUERTO F.
NO
C
SE ENVIARÓN LOS 3 BYTES DE POSICIÓN
SI
CHECAR SI EL TERCER BYTE DE VELOCIDAD INDICA ALGUN ERROR.
SI
EXISTE ERROR EN LA VELOCIDAD
ESCRIBIR AL PUERTO B EL CODIGO DE ERROR DE VELOCIDAD.
NO
LEER EL PUERTO E ESPERANDO LA RESPUESTA DE LA PC.
ESCRIBIR AL PUERTO B UN $F5 QUE INDICA QUE LA VELOCIDAD ES LA CORRECTA.
NO
ES EL PUERTO E IGUAL A $FF
ENTRAR A SUBRUTINA DE TIEMPO 2 DE 100 mseg.
LEER EL PUERTO E PARA COMPROBAR LA RESPUESTA DE LA PC.
SI
B
ESCRIBIR AL PUERTO B EL TERCER BYTE DE VELOCIDAD.
FIN


B
ES EL PUERTO E IGUAL A $FF
NO
ENTRAR A SUBRUTINA DE TIEMPO 2 DE 100 mseg.
SI
ENTRAR A PROGRAMADE ALARMAS.
CHECAR CONDICIÓN DE LA TERMINAL PA6 DEL PUERTO A.
ES LA TERMINAL PA6 IGUAL A UNO
NO
C
SI
FIN

8.12.4 SUBRUTINA DE TIEMPO 2 PARA EL PROGRAMA DE COMUNICACIÓN DE EL MICROCONTROLADOR A LA COMPUTADORA PERSONAL (PC).



8.13 DESARROLLO DE PROGRAMAS DE DETECCIÓN DE POSICIÓN

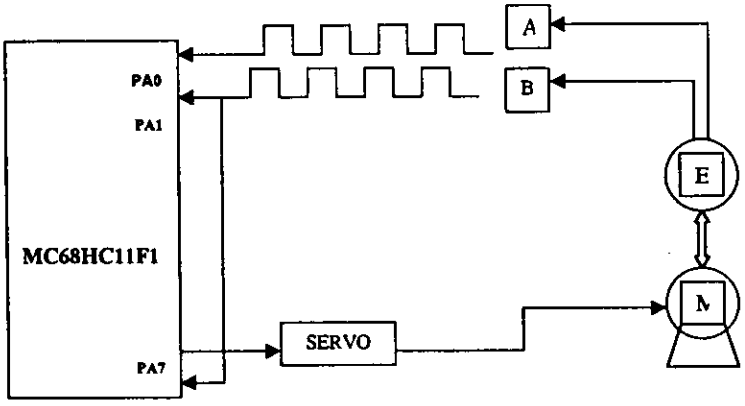
8.13.1 PROCEDIMIENTO PARA MEDIR LA LONGITUD DE MATERIAL CORTADO

Se debe de tomar en consideración los siguientes puntos; los pulsos que provienen del encoder tienen una equivalencia de un micrómetro por pulso (1pulso=1μm), además de que el programa debe detectar el cambio de sentido del motor, y que debe de parar el movimiento del motor una vez que se llegue a la distancia señalada, siendo que la máxima longitud a cortar es de 4 cm.

Para realizar lo anteriormente expuesto utilizaremos tres entradas y una salida del puerto A y el acumulador de pulsos del microcontrolador.

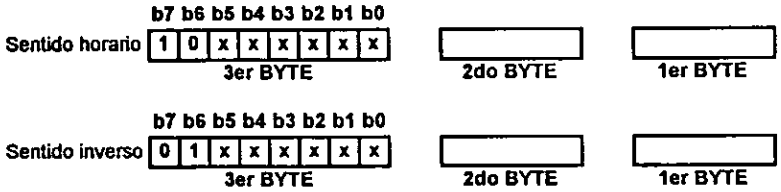
La primera consideración a tomar es el que tienen que usar 3 bytes para la medición de la longitud a cortar, a los cuales se les dará el nombre de AVANCE, AVANCE1 y AVANCE2, que son; primer byte , segundo byte y tercer byte de longitud de material cortado, respectivamente; en donde del tercer byte, en los bits b7 y b6, se colocará el sentido hacia donde gira el motor.

Primeramente el programa debe detectar el sentido hacia donde se mueve el eje del tornillo sin fin para establecer cuando se inicia su movimiento, sea este horario o antihorario, para lo cual utilizaremos dos entradas del puerto A donde llegarán las dos señales que provienen del encoder, como lo muestra el siguiente diagrama.



Aprovechando que las señales están desfasadas entre sí, el microcontrolador debe de detectar cual de estas dos señales ocurre primero, valiéndonos que el microcontrolador puede detectar tanto; flancos de subida como flancos de bajada, se puede programar el registro TCTL2, para que el microcontrolador detecte cualquier flanco que ocurra en las terminales PA0 y PA1, señalándolo en el registro TFLG1 mediante la activación de las banderas IC2F o IC3F; tomando la consideración que sí el primer flanco que detecte sea el de PA0 este se tomará

como positivo, almacenándose en el 3er byte con un 1 en b7 y un 0 en b6; y si se detecta primero el flanco en PA1 el sentido será inverso, b7 tendrá un 0 y b6 un 1.



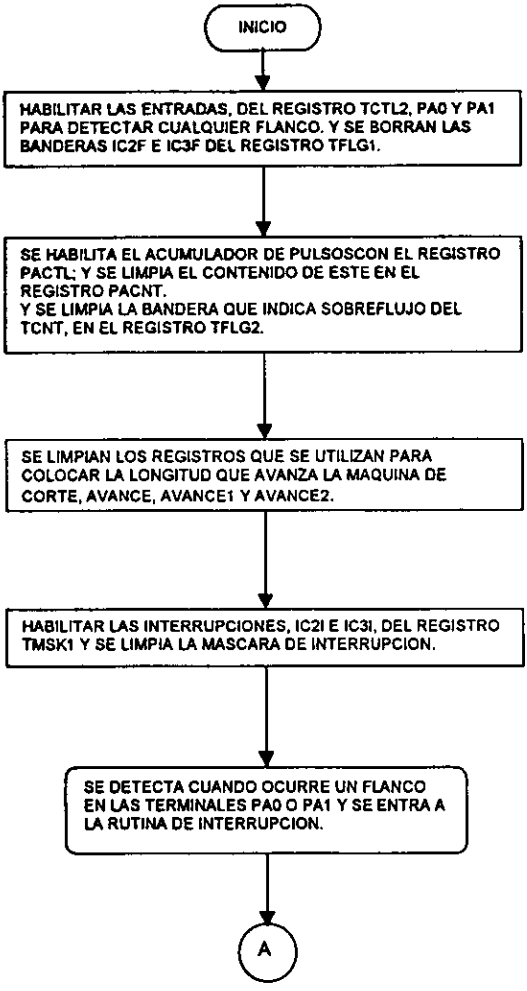
Debe de tomarse en cuenta que cuando sé este detectando el sentido, hacia donde gira el motor, el programa debe de interrumpirse mientras se indique el sentido, continuando el programa una vez que finalice esta tarea. Para lo cual activaremos las interrupciones IC2I e IC3I, del registro TMSK1, que realizan una interrupción en el programa principal en cuanto se active alguna bandera del registro TFLG1, entrando a un programa secundario donde se detectara el sentido del motor; una vez realizado este programa secundario se saldrá de él, regresando al programa principal, a la siguiente instrucción de donde ocurrió la interrupción.

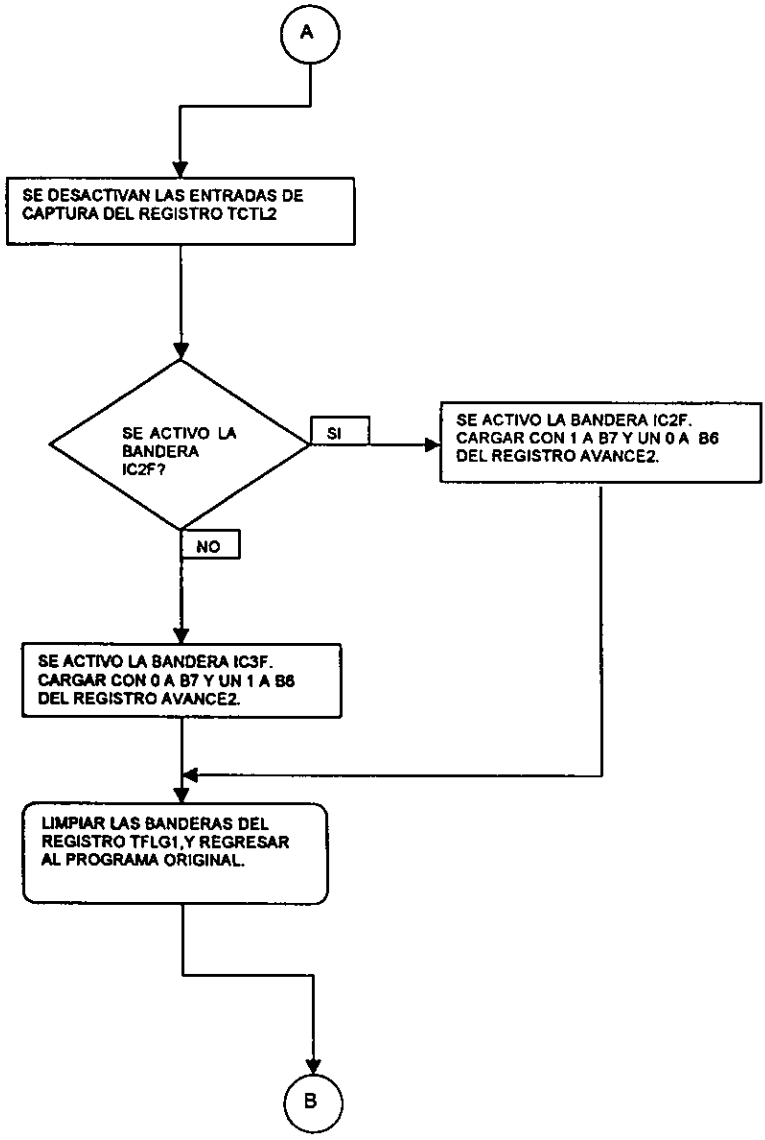
El programa principal debe de comparar un número de 3 bytes, que contiene la distancia que debe de cortar la máquina, que es una información que la PC le envía al microcontrolador, y que está almacenado en ciertas localidades de memoria, llamadas LONG, LONG1 y LONG2, del mismo microcontrolador; contra otro número de 3 bytes que se obtiene del conteo de los pulsos del encoder que le llegan al acumulador de pulsos, que como se mencionó anteriormente llamaremos AVANCE, AVANCE1 y AVANCE2. Una vez que estos números sean iguales, se mandará un pulso por el puerto A en PA6 que detenga al servo concluyendo el programa.

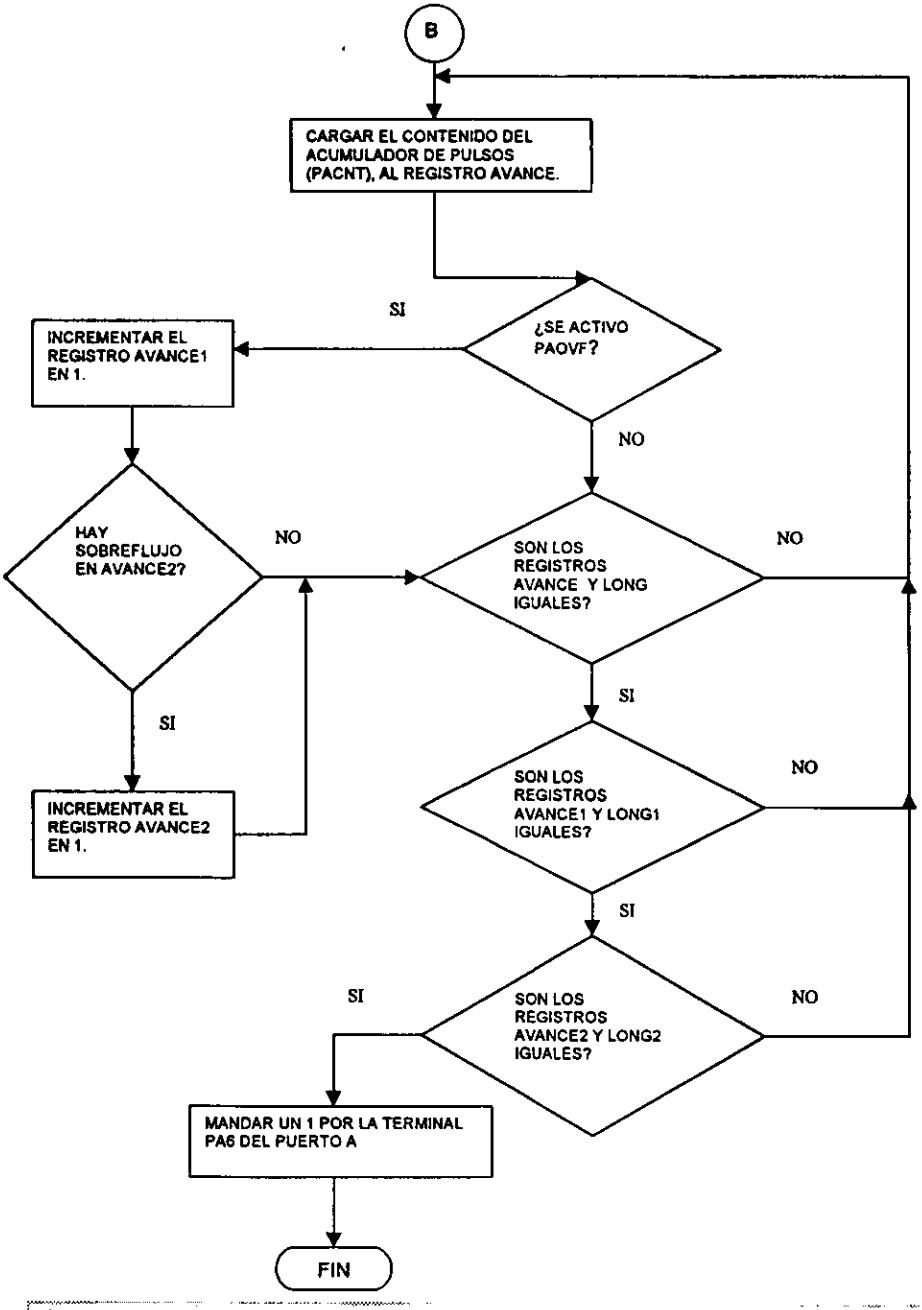
La forma en que se mide la longitud de material cortado y que se almacena en AVANCE, AVANCE1 y AVANCE2 es la siguiente; los pulsos del encoder entran a la terminal PA7 del microcontrolador, que es la entrada del acumulador de pulsos, estos se contabilizan en el registro PACNT, pasando enseguida al registro AVANCE, que es el byte menos significativo, comparándose continuamente con el registro LONG, que a su vez es el byte menos significativo, una vez que estos sean iguales, se procederá a comparar los bytes medios que son AVANCE1 y LONG1, y si estos son iguales, se comparan los bytes más significativos que son AVANCE2 y LONG2; solo cuando los tres bytes sean igual entre si se ordenará parar el proceso.

En el momento que el registro PACNT tenga un sobreflujo, se encenderá la bandera PAOVF del registro TFLG2, cada vez que esto ocurra se deberá incrementar en uno al registro AVANCE1 y borrar la bandera de sobreflujo, y cuando el registro AVANCE1 tenga un sobre flujo, que se detectará cuando ocurra un cambio de $FF a $00, se deberá incrementar en uno al registro AVANCE2.

Enseguida se presenta el diagrama de flujo para este programa. El listado del programa se encuentra en el Apéndice 2.

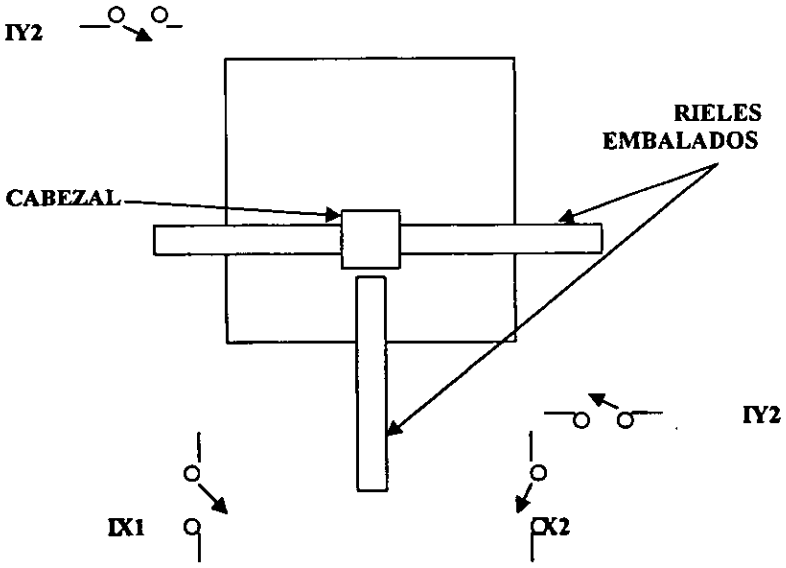

INICIO
HABILITAR LAS ENTRADAS, DEL REGISTRO TCTL2, PA0 Y PA1 PARA DETECTAR CUALQUIER FLANCO. Y SE BORRAN LAS BANDERAS IC2F E IC3F DEL REGISTRO TFLG1.
SE HABILITA EL ACUMULADOR DE PULSOSCON EL REGISTRO PACTL; Y SE LIMPIA EL CONTENIDO DE ESTE EN EL REGISTRO PACNT.
Y SE LIMPIA LA BANDERA QUE INDICA SOBREFLUJO DEL TCNT, EN EL REGISTRO TFLG2.
SE LIMPIAN LOS REGISTROS QUE SE UTILIZAN PARA COLOCAR LA LONGITUD QUE AVANZA LA MAQUINA DE CORTE, AVANCE, AVANCE1 Y AVANCE2.
HABILITAR LAS INTERRUPCIONES, IC2I E IC3I, DEL REGISTRO TMSK1 Y SE LIMPIA LA MASCARA DE INTERRUPCION.
SE DETECTA CUANDO OCURRE UN FLANCO EN LAS TERMINALES PA0 O PA1 Y SE ENTRA A LA RUTINA DE INTERRUPCION.
A


A
SE DESACTIVAN LAS ENTRADAS DE CAPTURA DEL REGISTRO TCTL2
SE ACTIVO LA BANDERA IC2F?
SI
SE ACTIVO LA BANDERA IC2F. CARGAR CON 1 A B7 Y UN 0 A B6 DEL REGISTRO AVANCE2.
NO
SE ACTIVO LA BANDERA IC3F. CARGAR CON 0 A B7 Y UN 1 A B6 DEL REGISTRO AVANCE2.
LIMPIAR LAS BANDERAS DEL REGISTRO TFLG1,Y REGRESAR AL PROGRAMA ORIGINAL.
B


B
CARGAR EL CONTENIDO DEL ACUMULADOR DE PULSOS (PACNT), AL REGISTRO AVANCE.
¿SE ACTIVO PAOVF?
SI
INCREMENTAR EL REGISTRO AVANCE1 EN 1.
NO
HAY SOBREFLUJO EN AVANCE2?
NO
SI
INCREMENTAR EL REGISTRO AVANCE2 EN 1.
SON LOS REGISTROS AVANCE Y LONG IGUALES?
NO
SI
SON LOS REGISTROS AVANCE1 Y LONG1 IGUALES?
NO
SI
SON LOS REGISTROS AVANCE2 Y LONG2 IGUALES?
SI
NO
MANDAR UN 1 POR LA TERMINAL PA6 DEL PUERTO A
FIN

8.13.2 PROCEDIMIENTO PARA COLOCAR EL CABEZAL DE LA MAQUINA DE CORTE EN EL ORIGEN

Para la realización de este programa se tomarán en cuenta los siguientes elementos; 4 interruptores de limite de posición que se encuentran ubicados en los bordes del área de corte como se muestra a continuación en el siguiente esquema.



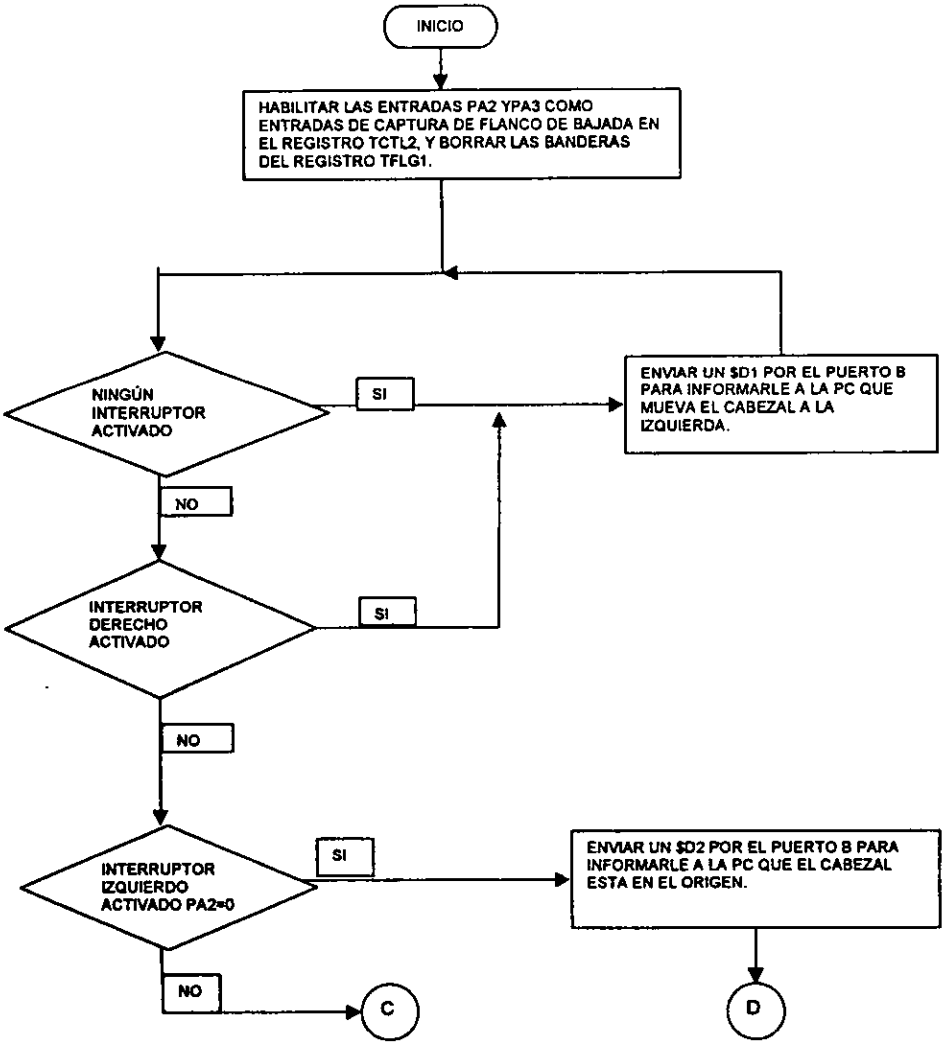
Se utilizará el puerto A para detectar cuando los interruptores son accionados, las terminales PA2 y PA3 estarán sensando el estado de los interruptores en el eje horizontal; para el eje vertical se utilizará el segundo microcontrolador, usando un programa igual al que se desarrollará en esta sección.

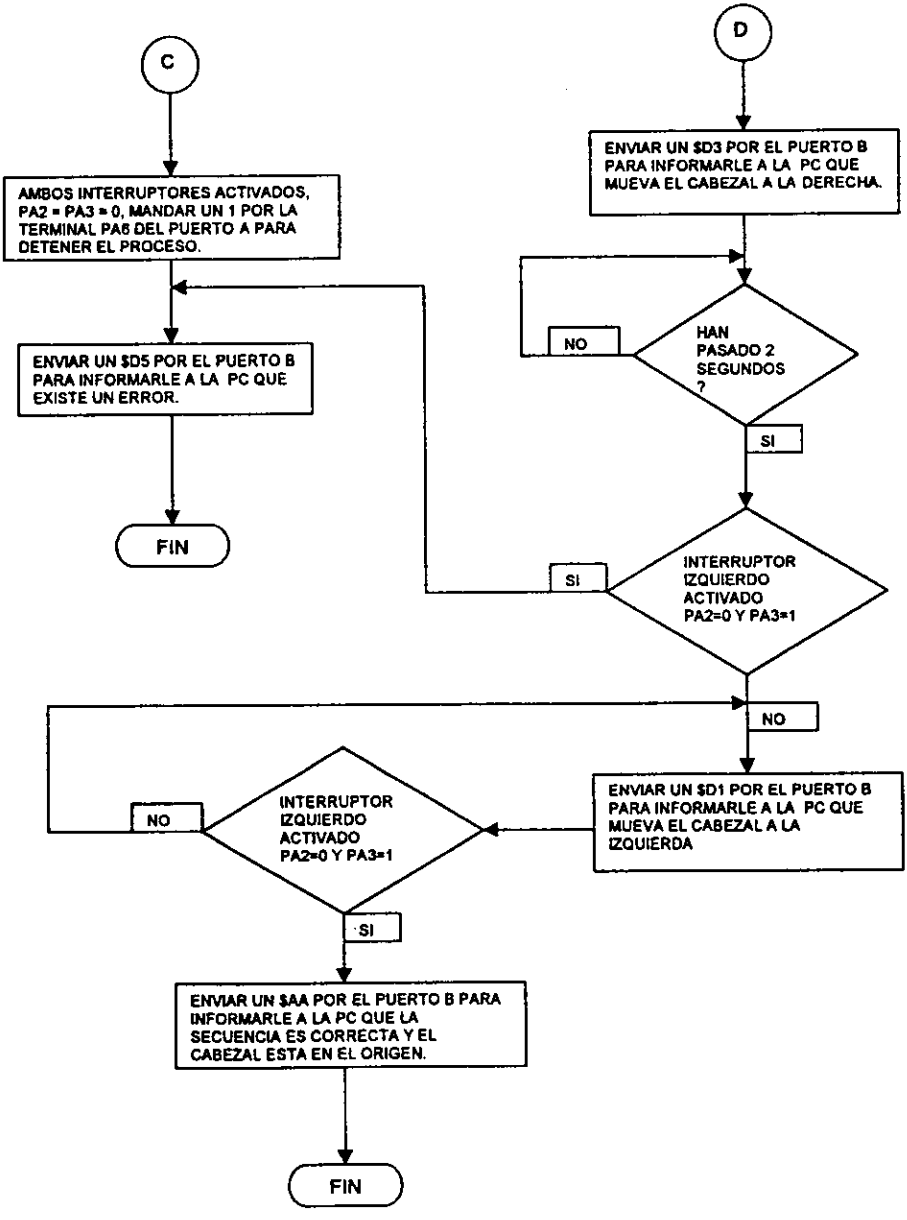
A los interruptores se les conocerá como IX1, IX2, IY1 e IY2; utilizando la siguiente lógica: cuando esté activado un interruptor se considera como 0 lógico y desactivado como un 1 lógico; por lo que si se tiene que IX1=IX2=1,el cabezal de la máquina no está activando ningún interruptor por lo que no se encuentra en ningún borde del área de corte; si IX1=1 e IX2=0 el cabezal se encuentra a la derecha del eje horizontal; cuando IX1=0 e IX2=1 el cabezal esta a la izquierda del eje horizontal por lo que se encuentra en el origen; y cuando se tenga que IX1=IX2=0 es una indicación de error que alguno de los interruptores a sufrido algún desperfecto. Para el eje vertical se utilizará la misma lógica antes descrita.

Cuando se tenga que IX1=IX2=1 ó IX1=1 e IX2=0 se le deberá de comunicar a la PC que debe de mover el cabezal de la máquina hacia la izquierda y cuando se detecte que está en el origen, parar el motor y avisar a la PC que se encuentra en posición. En el caso que IX1=0 e IX2=1 se le comunicará a la PC que debe de mover el cabezal a la derecha durante cierto tiempo (para nuestro caso 2 segundos) y después moverlo a la izquierda parando el microcontrolador el motor cuando detecte que el cabezal llegue al origen horizontal. Esto con la finalidad de observar que el interruptor IX1 funcione correctamente. Para el caso de IX1=IX2=0 se enviará un mensaje de error a la PC para que no se accione ningún motor y el operador verifique la máquina. Los códigos que el microcontrolador enviará a la PC son:

- $D1; orden para mover el cabezal a la izquierda.
- $D2; el cabezal esta en el origen y se procederá a observar que funcione correctamente el interruptor.
- $D3; orden para mover el cabezal a la derecha.
- $D4; el cabezal esta en el origen y los interruptores funcionan adecuadamente.
- $D5; error, existe más de un interruptor activado.
- $AA; cabezal en el origen, interruptores funcionan correctamente.

A continuación se presenta el diagrama de flujo para esta parte. El listado del programa se encuentra en el Apéndice 2.

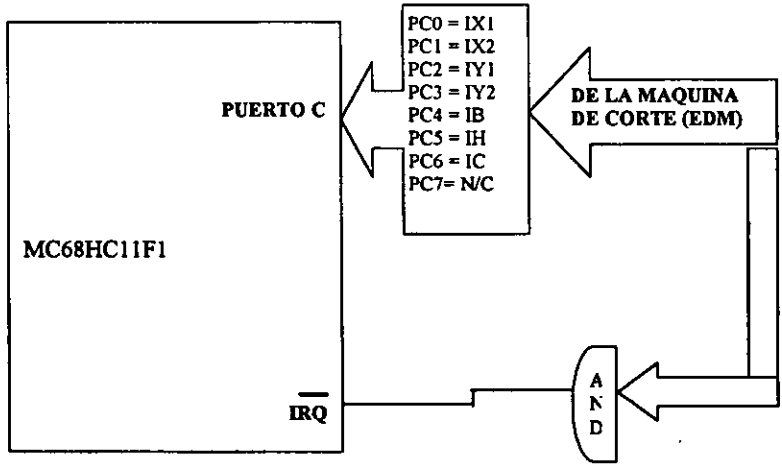

INICIO
HABILITAR LAS ENTRADAS PA2 YPA3 COMO ENTRADAS DE CAPTURA DE FLANCO DE BAJADA EN EL REGISTRO TCTL2, Y BORRAR LAS BANDERAS DEL REGISTRO TFLG1.
NINGÚN INTERRUPTOR ACTIVADO
SI
ENVIAR UN $D1 POR EL PUERTO B PARA INFORMARLE A LA PC QUE MUEVA EL CABEZAL A LA IZQUIERDA.
NO
INTERRUPTOR DERECHO ACTIVADO
SI
NO
INTERRUPTOR IZQUIERDO ACTIVADO PA2=0
SI
ENVIAR UN $D2 POR EL PUERTO B PARA INFORMARLE A LA PC QUE EL CABEZAL ESTA EN EL ORIGEN.
NO
C
D


C
AMBOS INTERRUPTORES ACTIVADOS, PA2 = PA3 = 0, MANDAR UN 1 POR LA TERMINAL PA6 DEL PUERTO A PARA DETENER EL PROCESO.
ENVIAR UN $D5 POR EL PUERTO B PARA INFORMARLE A LA PC QUE EXISTE UN ERROR.
FIN
D
ENVIAR UN $D3 POR EL PUERTO B PARA INFORMARLE A LA PC QUE MUEVA EL CABEZAL A LA DERECHA.
NO
HAN PASADO 2 SEGUNDOS ?
SI
SI
INTERRUPTOR IZQUIERDO ACTIVADO PA2=0 Y PA3=1
NO
ENVIAR UN $D1 POR EL PUERTO B PARA INFORMARLE A LA PC QUE MUEVA EL CABEZAL A LA IZQUIERDA
NO
INTERRUPTOR IZQUIERDO ACTIVADO PA2=0 Y PA3=1
SI
ENVIAR UN $AA POR EL PUERTO B PARA INFORMARLE A LA PC QUE LA SECUENCIA ES CORRECTA Y EL CABEZAL ESTA EN EL ORIGEN.
FIN

8.13.3 PROGRAMA PARA DETECTAR LA ACTIVACIÓN DE INTERRUPTORES (ALARMAS) CUANDO SE ENCUENTRA OPERANDO LA MÁQUINA.

Para cuando la máquina de corte este operando se debe de observar que trabaje dentro del área de corte, que el hilo de corte no se rompa, que la bomba tenga agua y que ninguna alarma se quede encendida, en todos estos casos la operación de la máquina debe parar para que el operador repare el desperfecto.

Para este efecto usaremos el puerto C configurándolo como entrada. Utilizaremos los interruptores de límite de posición IX1, IX2, IY1 e IY2, también interruptores que censan la falta de agua (IB) y cuando el hilo se rompe (IH). También censaremos un interruptor que nos indica cuando el cabezal de la máquina de corte se encuentra en el centro del área de corte (IC), sin embargo este sólo servirá como indicador, ya que la máquina no interrumpirá su proceso al llegar a este punto, el microcontrolador deberá parar el proceso de la máquina cuando detecte que se han activado alguno de los demás interruptores.

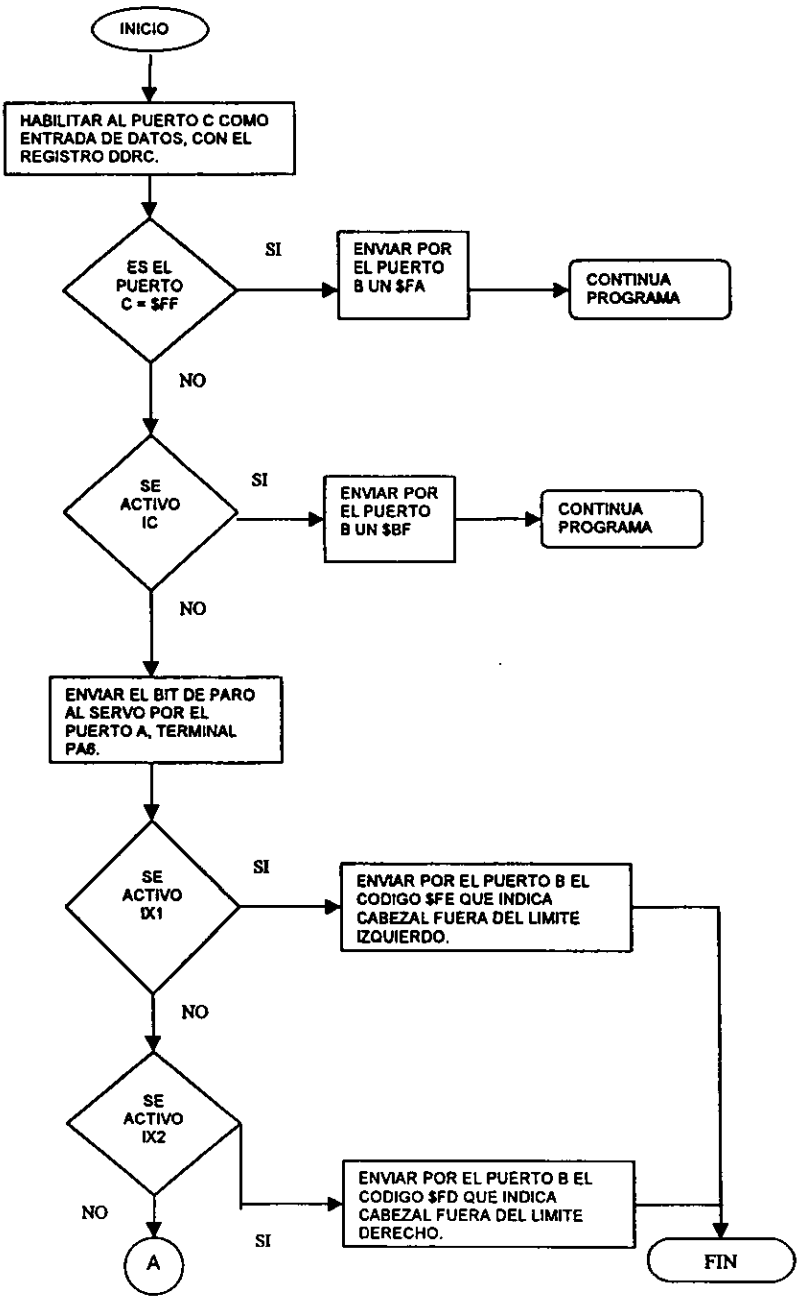


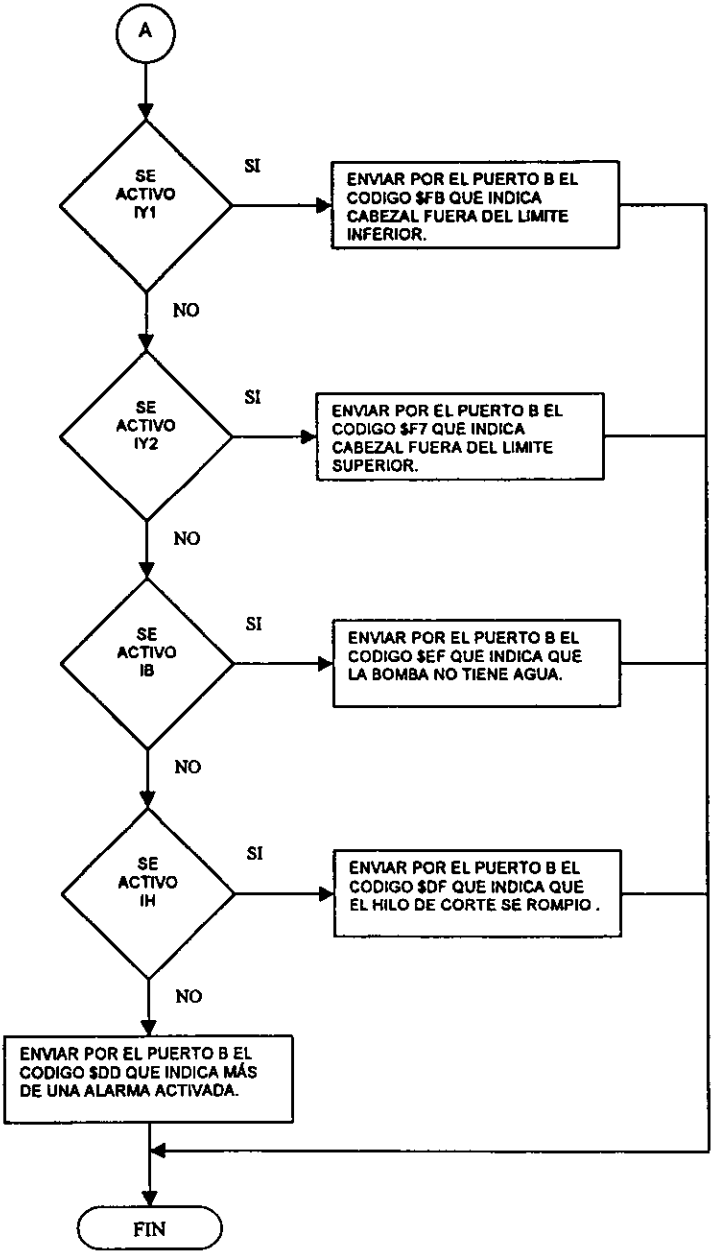
Todos los interruptores además de llegar al puerto C, serán conectados a una compuerta AND y la salida de esta a la entrada IRQ negada del microcontrolador, para que ocurra una interrupción en el programa, además de enviar un código a la PC por el puerto B para que el operador observe en donde estuvo el error.

A continuación se presenta los códigos de error que enviará el microcontrolador a la PC y el diagrama de flujo para esta parte. El listado del programa se encuentra en el Apéndice 2.

- $FE; indica que el interruptor IX1 se activó y por lo tanto se encuentra fuera de límite izquierdo.

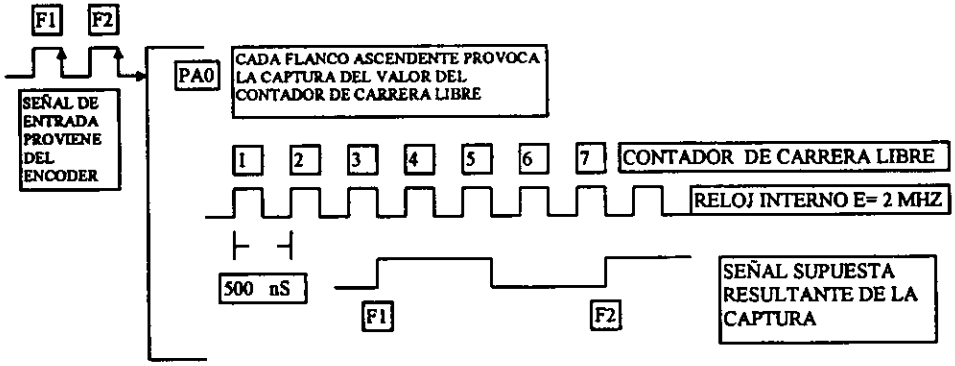
- **$FD**; indica que el interruptor IX2 se activó y por lo tanto se encuentra fuera de límite derecho.
- **$FB**; indica que el interruptor IY1 se activó y por lo tanto se encuentra fuera de límite inferior.
- **$F7**; indica que el interruptor iY2 se activó y por lo tanto se encuentra fuera de límite superior.
- **$EF**; indica que el interruptor IB se activó y por lo tanto la bomba no tiene agua
- **$DF**; indica que el interruptor IH se activó y por lo tanto el hilo de corte se ha roto.
- **$BF**; indica que el interruptor IC se activó y por lo tanto el cabezal esta al centro.
- **$DD**; error grave, más de una alarma activada.


INICIO
HABILITAR AL PUERTO C COMO ENTRADA DE DATOS, CON EL REGISTRO DDRC.
ES EL PUERTO C = $FF
SI
ENVIAR POR EL PUERTO B UN $FA
CONTINUA PROGRAMA
NO
SE ACTIVO IC
SI
ENVIAR POR EL PUERTO B UN $BF
CONTINUA PROGRAMA
NO
ENVIAR EL BIT DE PARO AL SERVO POR EL PUERTO A, TERMINAL PA6.
SE ACTIVO IX1
SI
ENVIAR POR EL PUERTO B EL CODIGO $FE QUE INDICA CABEZAL FUERA DEL LIMITE IZQUIERDO.
NO
SE ACTIVO IX2
NO
A
SI
ENVIAR POR EL PUERTO B EL CODIGO $FD QUE INDICA CABEZAL FUERA DEL LIMITE DERECHO.
FIN

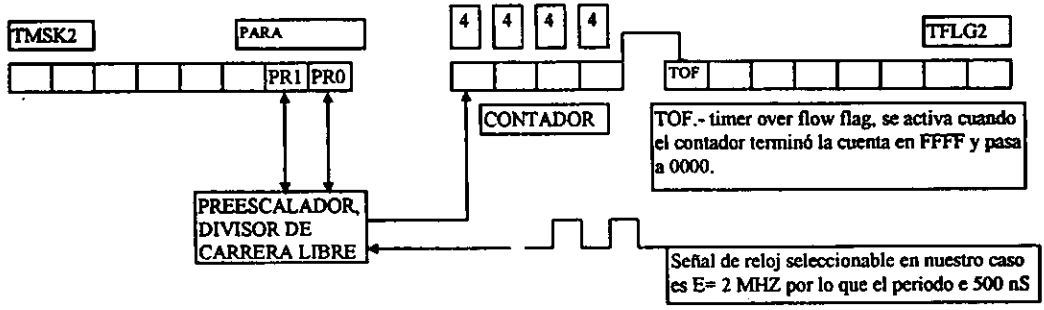

A
SE ACTIVO IY1
SI
ENVIAR POR EL PUERTO B EL CODIGO $FB QUE INDICA CABEZAL FUERA DEL LIMITE INFERIOR.
NO
SE ACTIVO IY2
SI
ENVIAR POR EL PUERTO B EL CODIGO $F7 QUE INDICA CABEZAL FUERA DEL LIMITE SUPERIOR.
NO
SE ACTIVO IB
SI
ENVIAR POR EL PUERTO B EL CODIGO $EF QUE INDICA QUE LA BOMBA NO TIENE AGUA.
NO
SE ACTIVO IH
SI
ENVIAR POR EL PUERTO B EL CODIGO $DF QUE INDICA QUE EL HILO DE CORTE SE ROMPIO .
NO
ENVIAR POR EL PUERTO B EL CODIGO $DD QUE INDICA MÁS DE UNA ALARMA ACTIVADA.
FIN

8.14. DESARROLLO DE PROGRAMA DE DETECCIÓN DE VELOCIDAD.

Para llevar a cabo este procedimiento, se requiere programar el puerto "A" como entrada de captura, ejemplificaremos esto en la fig. Sig.



Asimismo se utiliza el contador de carrera libre, representado en forma esquemática en la fig.

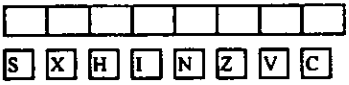


Como consecuencia del periodo el contador de carrera libre se incrementará una unidad cada 500 nS.

Los bits PR0 y PR1, solo pueden ser modificados en los 64 primeros ciclos de reloj iniciales después del reset.

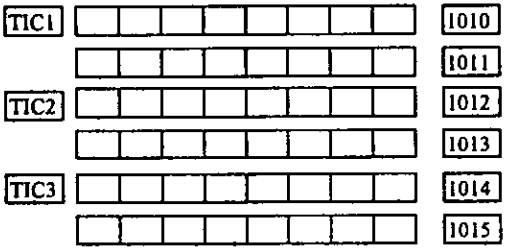
Los registros principales que se utilizan son los que a continuación se mencionan :

CCR.- registro de código de condición.



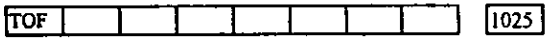
S.- Paro de reloj interno
X.- Mascara de interconexión (interrupción mascarable)
H.- Acarreo medio
I.- Bit de interrupción
N.- Negativo
Z.- Cero
V.- Sobre flujo
C.- Acarreo

Registro de captura.



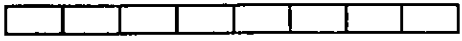
En estos se almacena el valor de la captura, utilizando cada registro dos localidades de memoria.

TFLG2



Este es el registro de bandera de interrupciones del Timer el bit que nos interesa es el bit 7 (TOF), ya que nos indica cuando el contador de carrera libre terminó su cuenta y reinicia nuevamente.

TMSK1



En el registro TMSK1 Máscara de interrupciones del Timer se habilitan las interrupciones del timer.

TCTL2

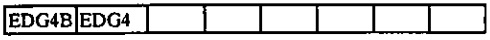


DIAGRAMA DE FLUJO DE PROGRAMA DE DETECCION DE LA VELOCIDAD DE CORTE

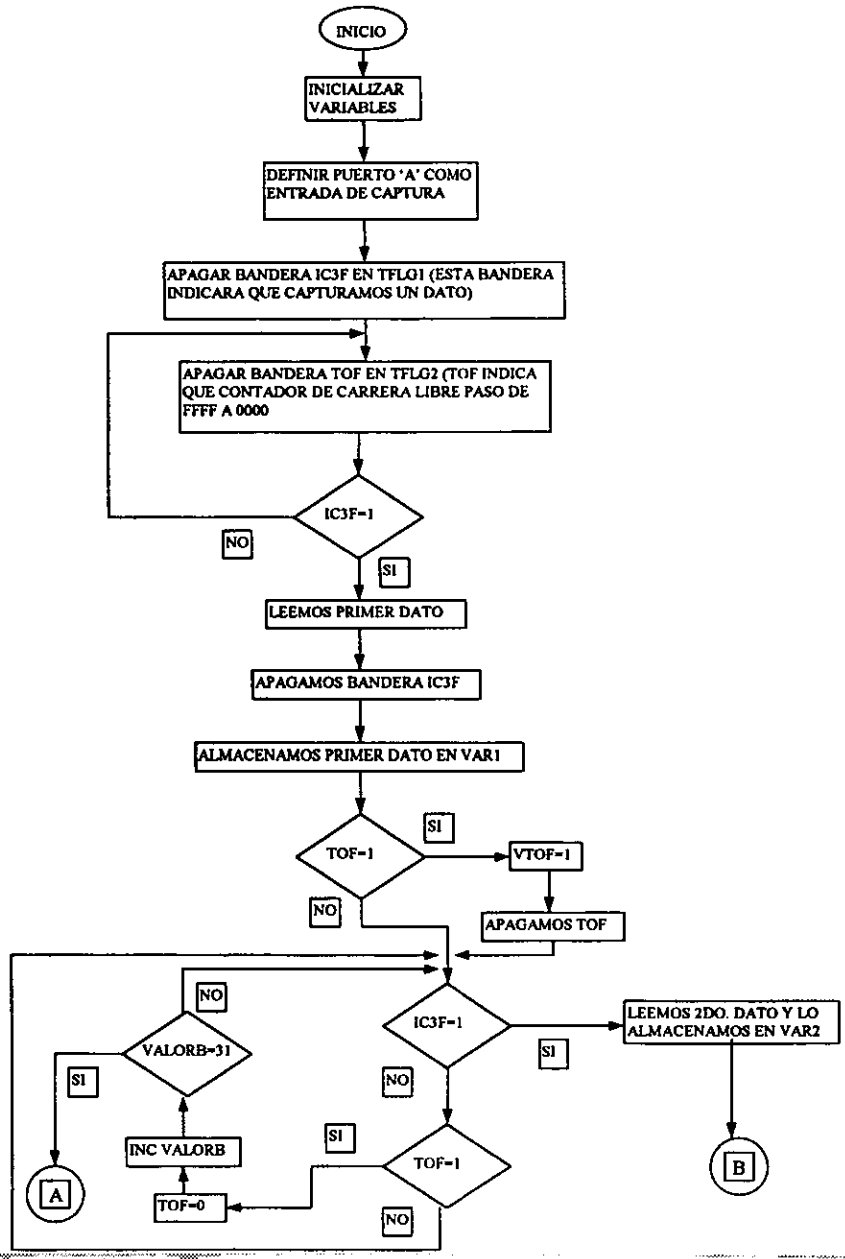


DIAGRAMA DE FLUJO DE PROGRAMA DE DETECCION DE LA VELOCIDAD DE CORTE CONTINUACION

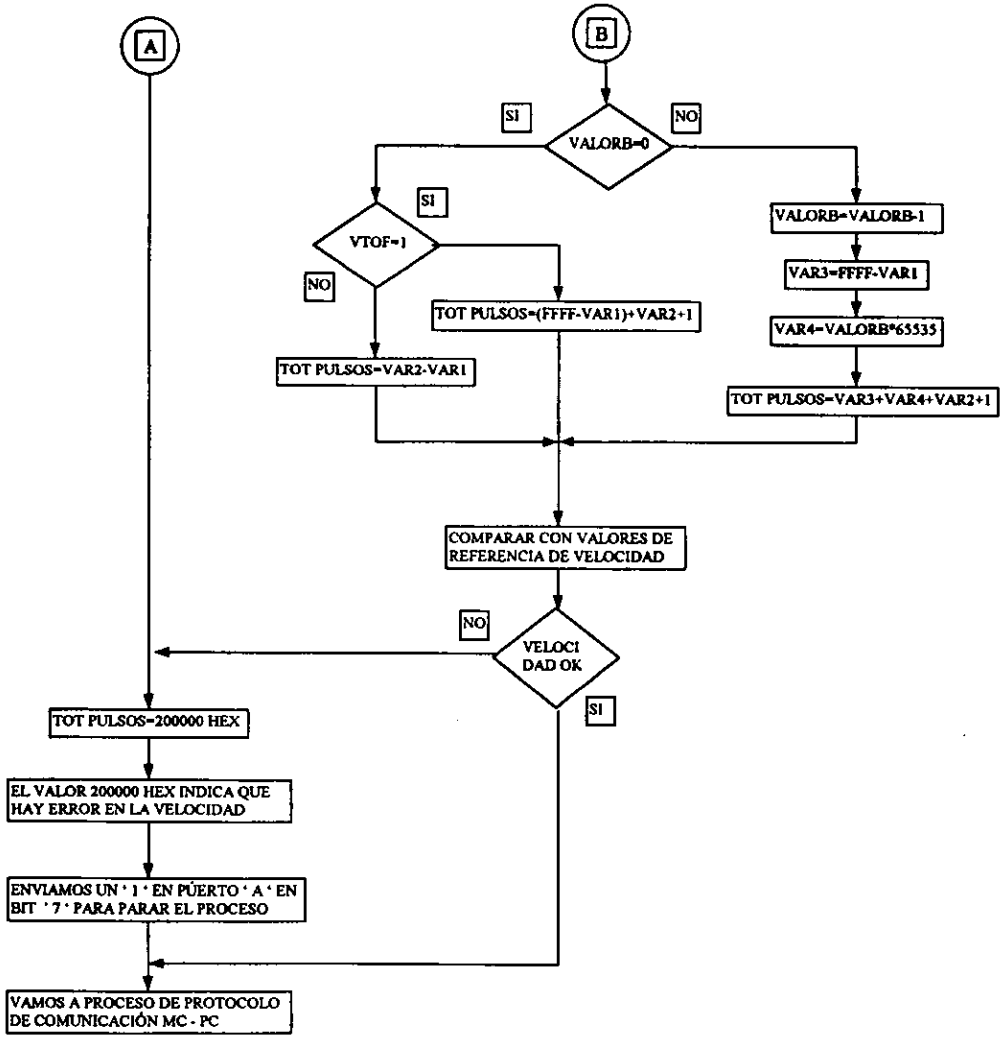
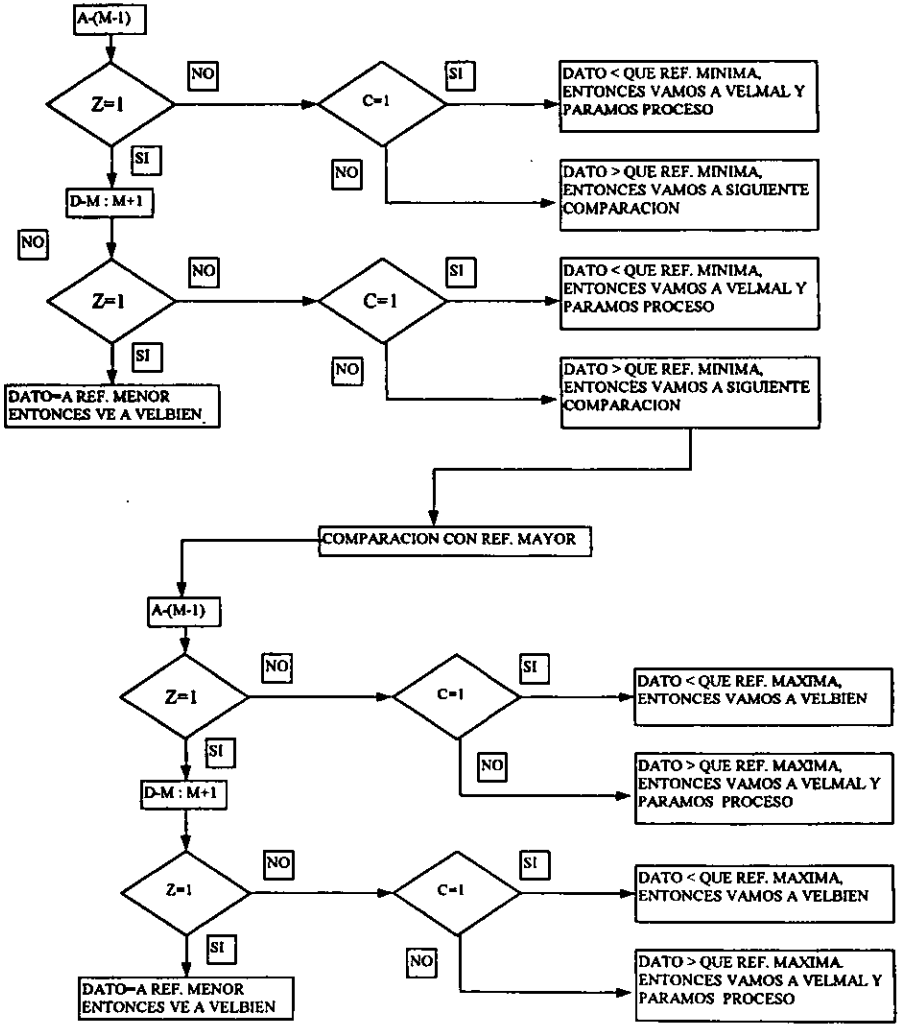


DIAGRAMA DE FLUJO DE PROGRAMA DE DETECCION DE LA VELOCIDAD DE CORTE CONTINUACION

PARTE COMPARACION DE VELOCIDAD OBTENIDA, CONTRA LIMITE DE VELOCIDAD INFERIOR Y SUPERIOR

A= VELOCIDAD OBTENIDA PARTE ALTA
D= VELOCIDAD OBTENIDA PARTE MEDIA Y BAJA
(M-1)= PARTE ALTA DEL DATO DE COMPARACION SEGÚN CORRESPONDA
M Y M-1= PARTE MEDIA Y BAJA DEL DATO DE COMPARACIÓN SEGÚN CORRESPONDA



Capítulo 9

RESULTADOS Y CONCLUSIONES

9.1. PRUEBAS DE OPERACIÓN

Una vez terminado el diseño y el desarrollo de la interface de la pc y el programa de control se realizaron varias pruebas encaminadas a corroborar su correcta operación tomando como base el protocolo de comunicación PC – MCU y viceversa, del cuál su diagrama de flujo se muestra en la sección de apéndices.

PRUEBA No. 1 – Se realizaron pequeños programas que leían y escribían información al bus de datos de la computadora y al mismo tiempo en el slot de la computadora, se checaron en el osciloscopio que señales estaban presentes durante la lectura y cuales dentro de la escritura.

RESULTADOS – Se verificó que al escribir la PC presenta la señal IOW y al leer está presente la señal IOR, ambas observadas del bus de control de la computadora.

PRUEBA No. 2 – En base a los resultados de la prueba No. 1, se diseño la tarjeta de interfaz con la PC y se conectó en un slot de la computadora de prueba donde se volvieron a checar los programas de lectura y escritura de información a los puertos físicos para los que se diseño la tarjeta.

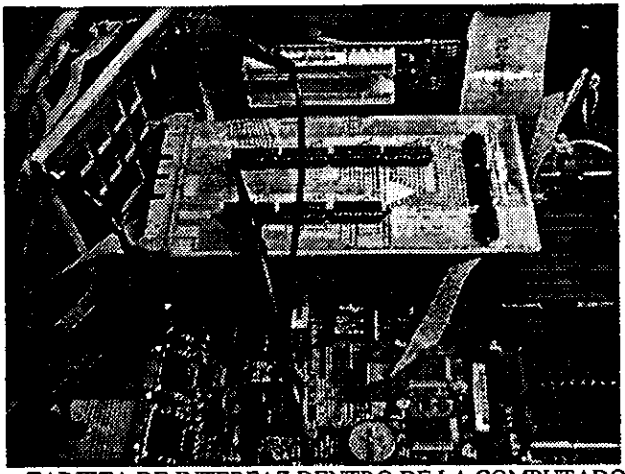


Fig. – TARJETA DE INTERFAZ DENTRO DE LA COMPUTADORA.

RESULTADOS – Al correr el programa de escritura los latches de la tarjeta captura la información de salida correctamente (0 o 1's lógicos) y al correr el de escritura los buffers capturaban los voltajes (0 o 1's lógicos) que se conectaron en sus entradas adecuadamente.

PRUEBA No. 3 - Con la tarjeta de evaluación universal del MCU HC11 y otra computadora personal, se realizaron pruebas de comunicación mediante la ejecución de programas de lectura y escritura de información.

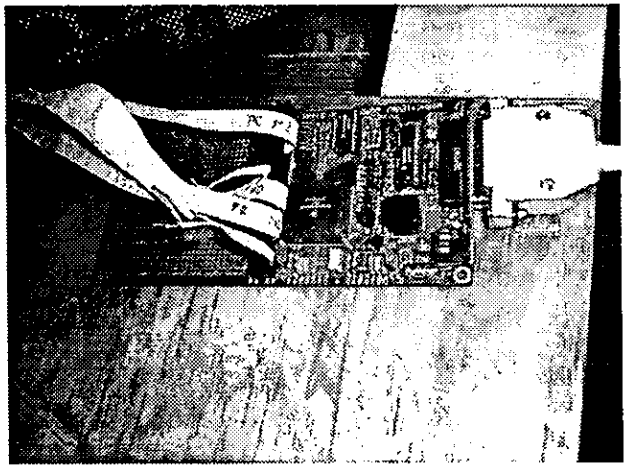
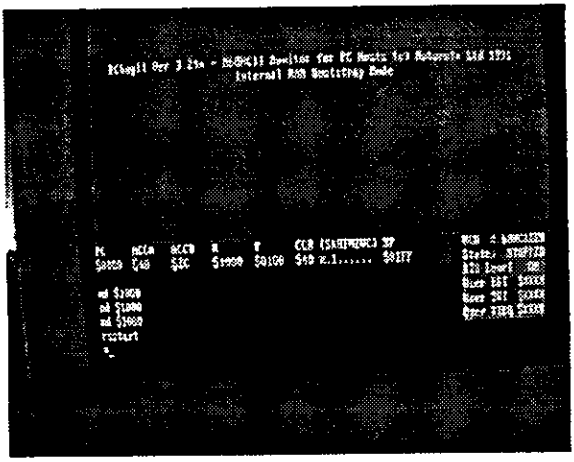
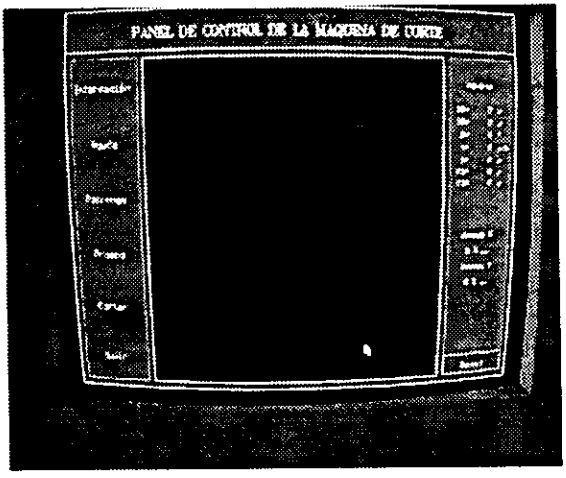


Fig. – Tarjeta de evaluación del MCU HC11

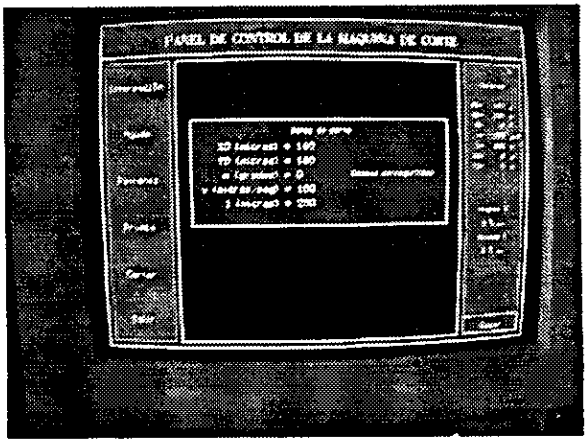
RESULTADOS- Al correr los programas de escritura y lectura de datos en el HC11 por medio del software de programación del HC11, corroboramos que esta funcionando correctamente la comunicación PC – MCU.



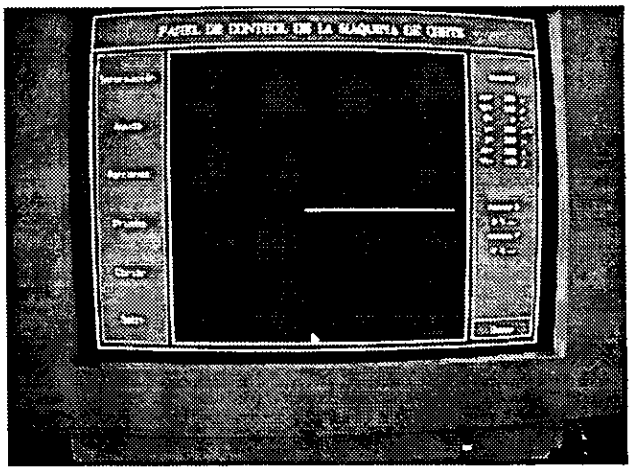
PRUEBA No. 4 – Se corrió desde el sistema operativo de la computadora el programa de control final.



Se ingresaron los parámetros necesarios para la ejecución de una secuencia de corte.



RESULTADOS – Se logro programar la operación automática de una secuencia de corte hacia la maquina E.D.M.



CONCLUSIONES

A partir de la experiencia obtenida y de los resultados que arrojaron los diferentes programas realizados en el momento de su simulación, podríamos concluir que el proyecto es factible de ser implementado para el control CNC del proceso de una máquina de corte EDM. Muy importante es tener en cuenta que el presente desarrollo solo contempla el diseño de cortes de líneas rectas por el hecho de que se manejan solamente dos ejes X e Y quedando abierta la expansión del sistema a cortes circulares, curvas etc.. con la inclusión de algoritmos de programación que se basen en la interpolación, dentro del avance en la investigación de este proyecto se encontró que existen en el mercado máquinas con controles de corte de hasta cuatro variables X, Y, Z y Rotación para cortes cónicos, por lo que con el presente trabajo queda la puerta abierta para el análisis y las adecuaciones necesarias al diseño en aplicaciones que pudieran competir con las existentes en el mercado.

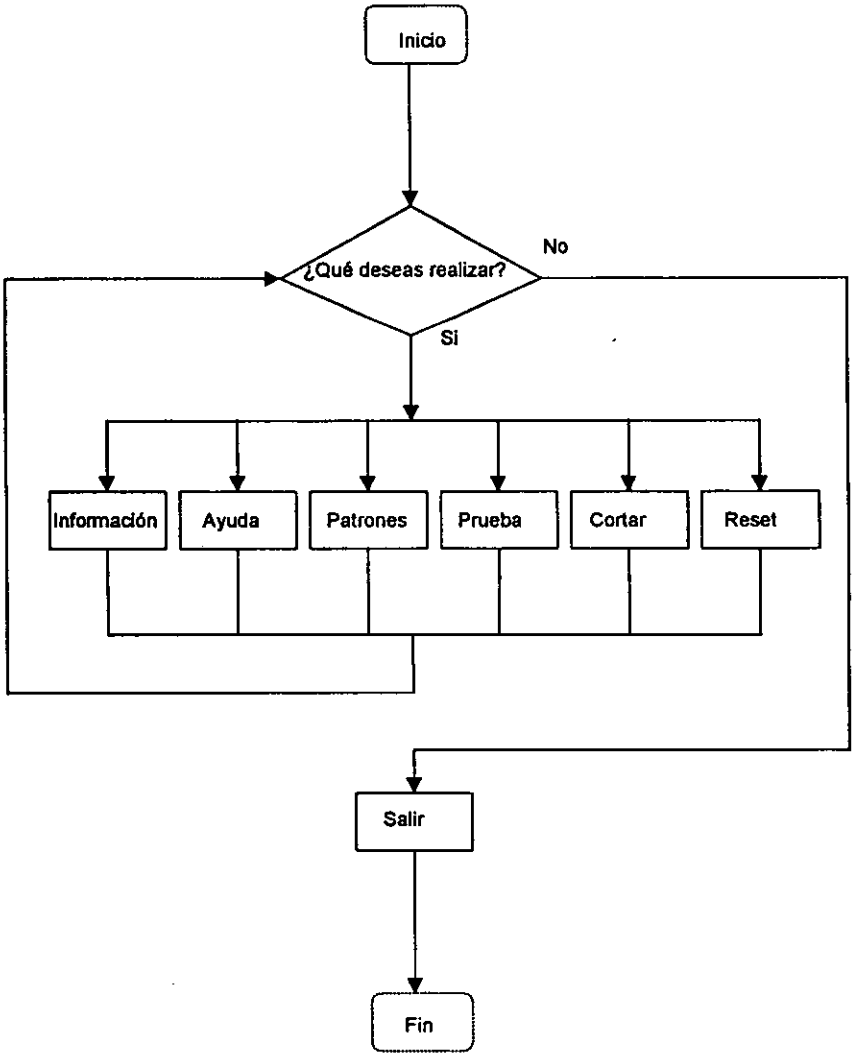
Otro punto importante detectado, es el de que cualquier proyecto que se realice para este tipo de máquinas debe prever lo necesario para la interrelación Hombre - Máquina que permita al operador tener una visión del proceso de corte o del estado de las alarmas (Posición, Hilo, Límite, Velocidad, etc.) , con lo que se realice la toma de decisión de continuar o detener el corte que se está realizando. Para el presente trabajo de investigación se desarrolló basándose en un ambiente gráfico en lenguaje "C" y a la definición de que el microcontrolador fuera el encargado de parar el proceso en el caso de presentarse cualquier alarma y a su vez, éste informará a la PC del paro del proceso y de la condición de alarma, para que el operador tome las acciones que correspondan.

Dentro del aspecto económico de este proyecto se encontró que es muy barato pues se basa en gran parte en una computadora personal de las mas sencillas como una AT, además de que el precio de estos equipos ha bajado considerablemente dentro de los últimos tiempos.

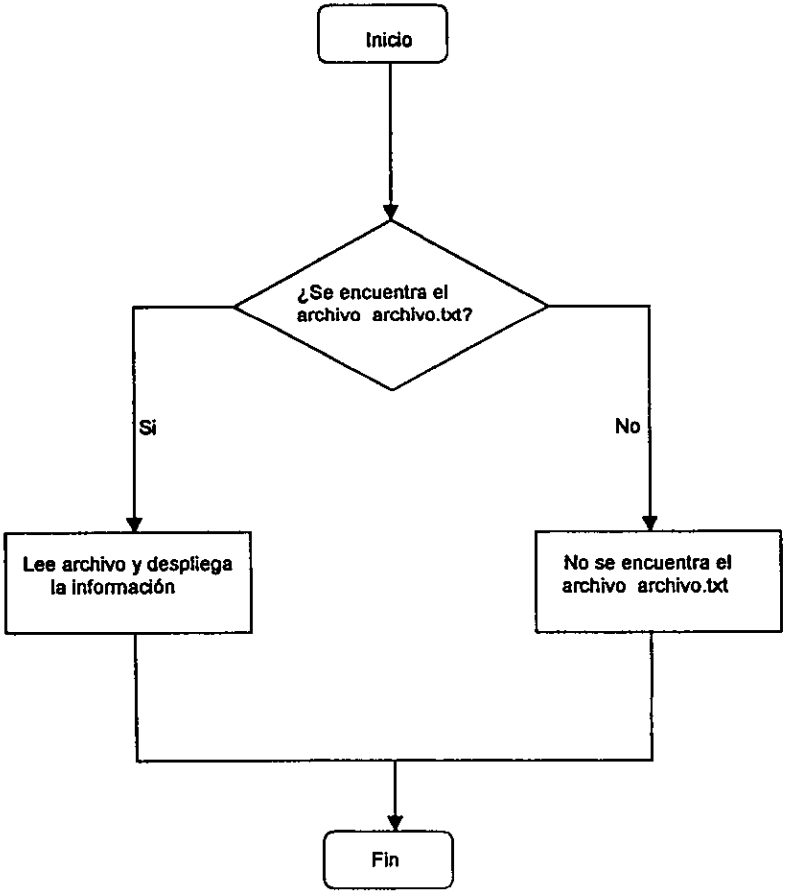
Apéndice A

Diagramas de flujo del programa en C

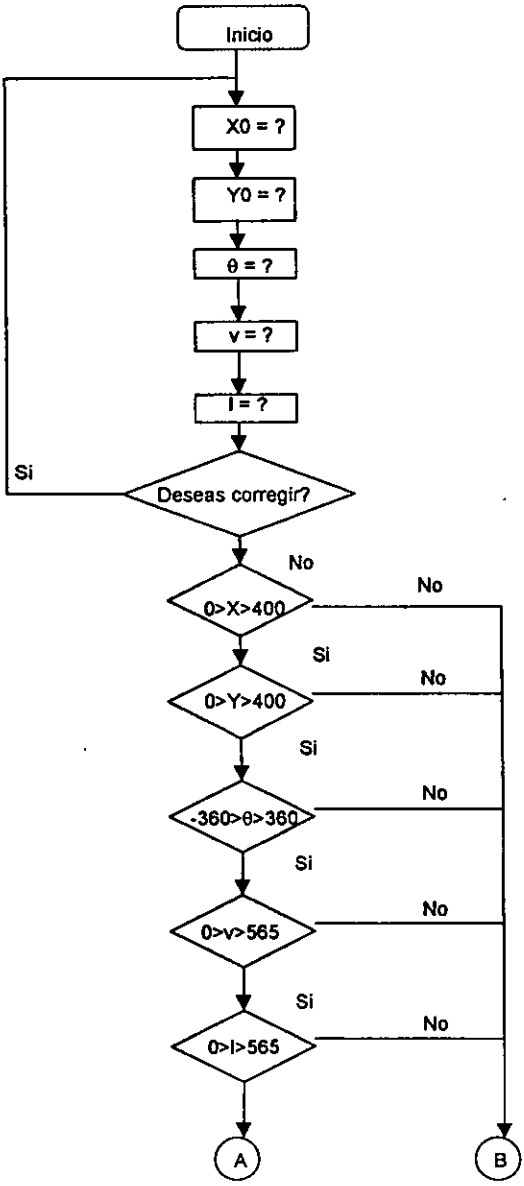
Función Menú Principal

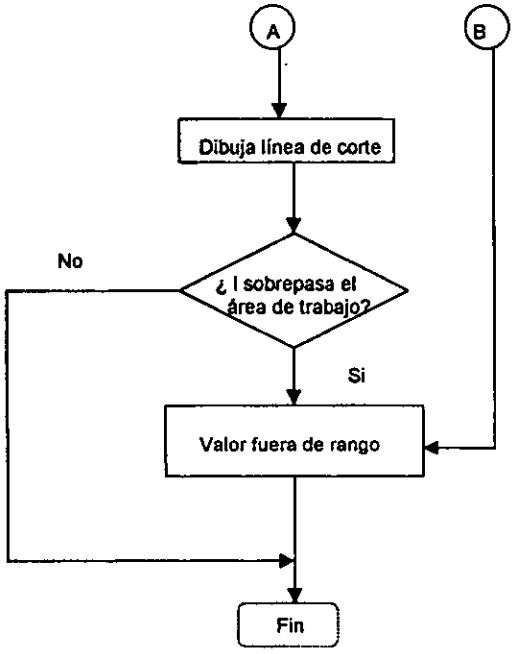


Función Menú Información y Ayuda

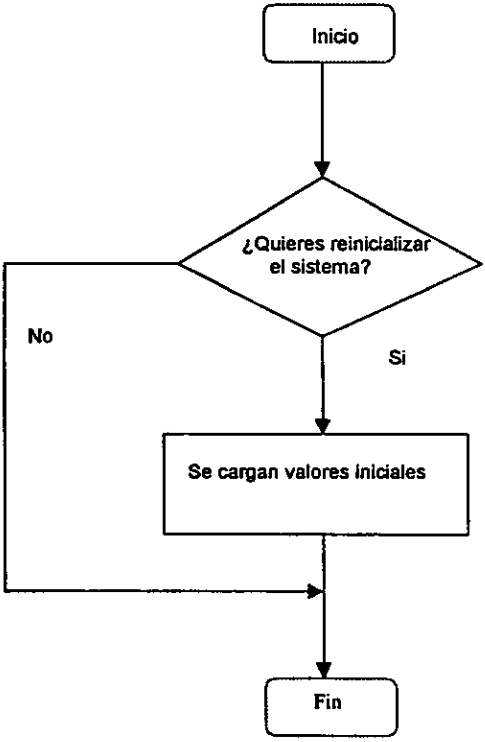


Función Menú Patrones

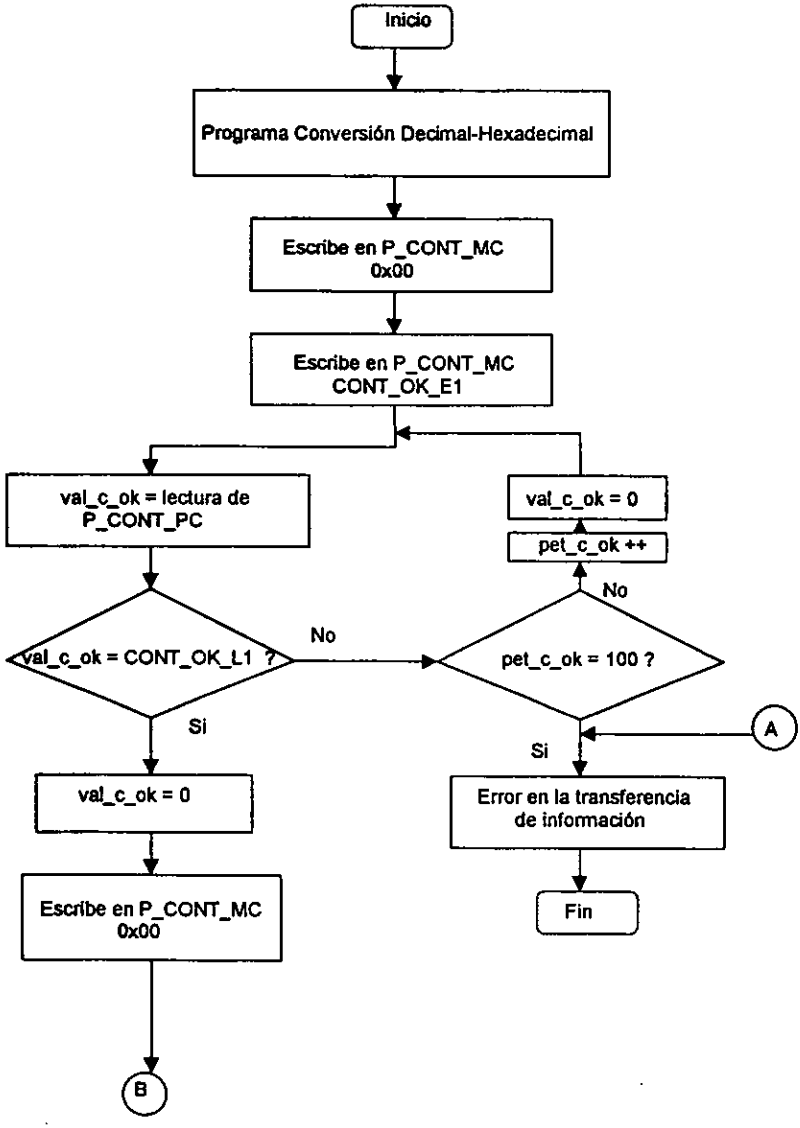


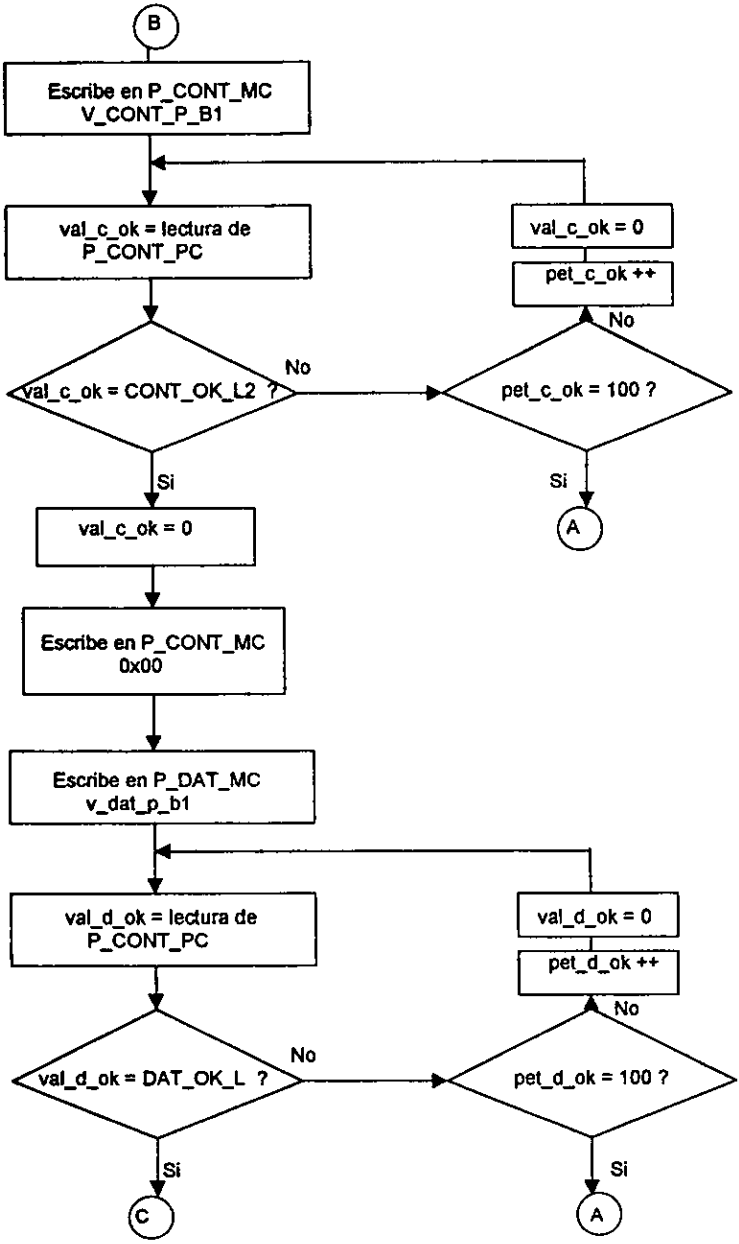

A
B
Dibuja línea de corte
No
¿ I sobrepasa el área de trabajo?
Si
Valor fuera de rango
Fin

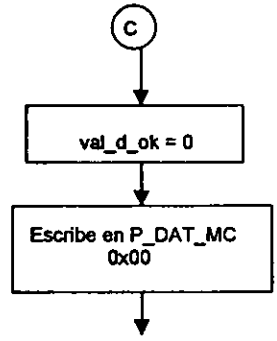
Función Menú Reset



Función Menú Prueba y Cortar




B
Escribe en P_CONT_MC
V_CONT_P_B1
val_c_ok = lectura de
P_CONT_PC
val_c_ok = 0
pet_c_ok ++
No
val_c_ok = CONT_OK_L2 ?
No
pet_c_ok = 100 ?
Si
Si
A
val_c_ok = 0
Escribe en P_CONT_MC
0x00
Escribe en P_DAT_MC
v_dat_p_b1
val_d_ok = lectura de
P_CONT_PC
val_d_ok = 0
pet_d_ok ++
No
val_d_ok = DAT_OK_L ?
No
pet_d_ok = 100 ?
Si
Si
C
A



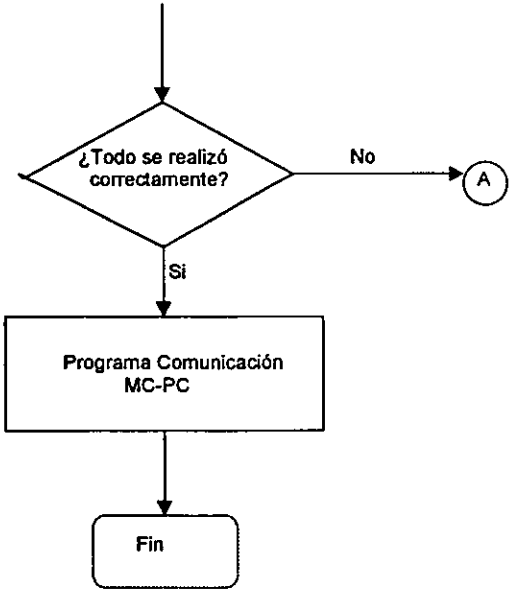
Transferencia velocidad máxima byte 2 y 3

Transferencia velocidad real byte 1,2 y 3

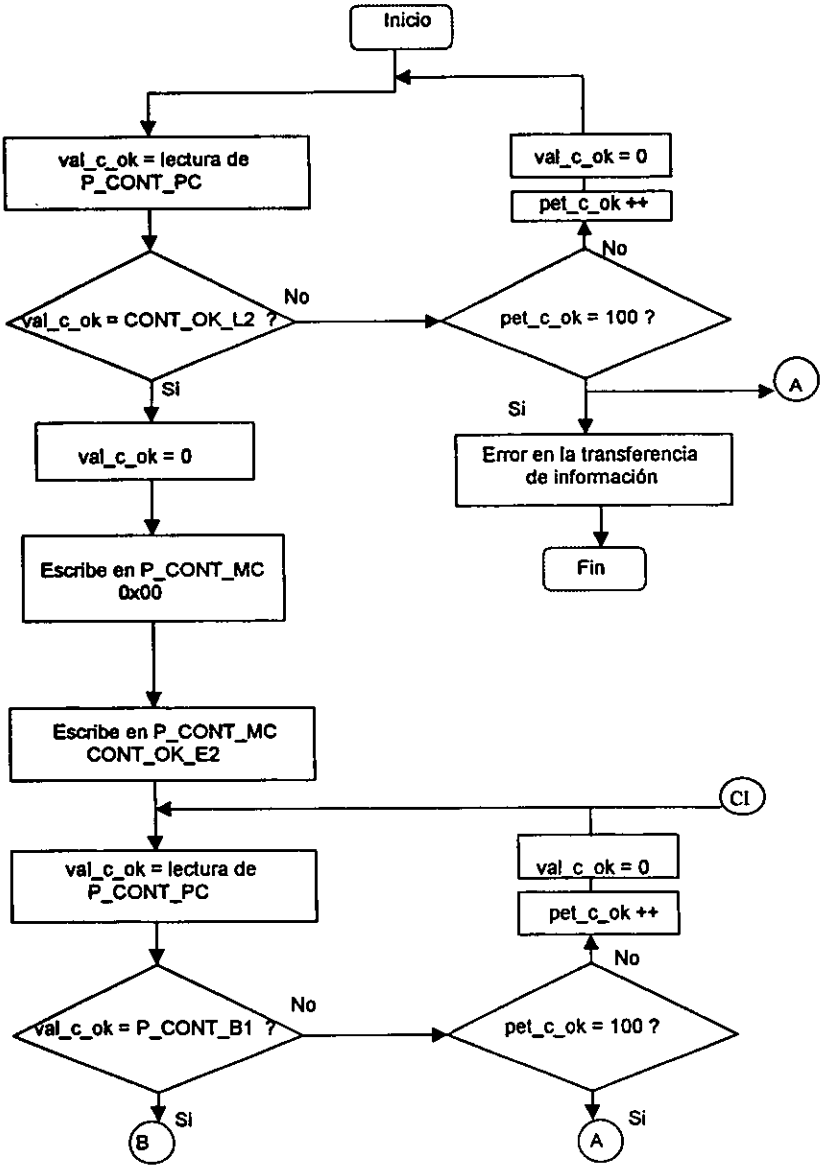
Transferencia velocidad mínima byte 1, 2 y 3

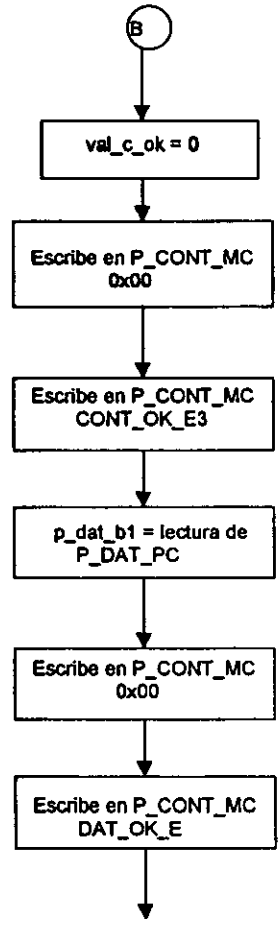
Transferencia posición origen byte 1, 2 y 3

Transferencia longitud byte 1, 2 y 3

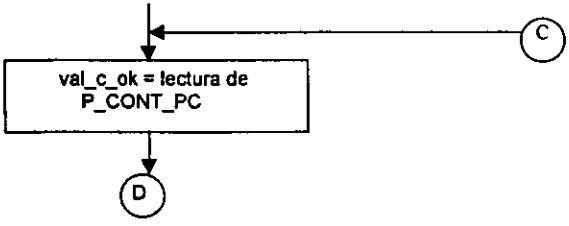


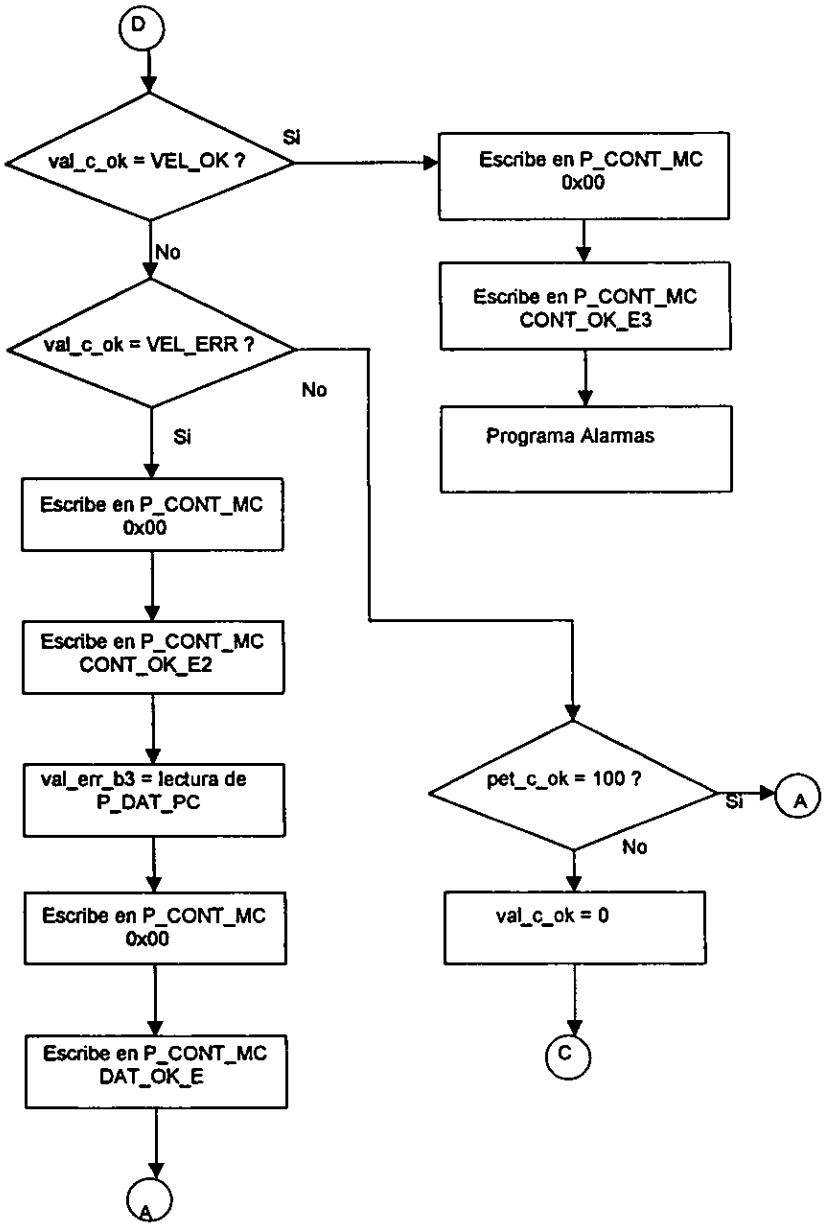
Programa MC - PC



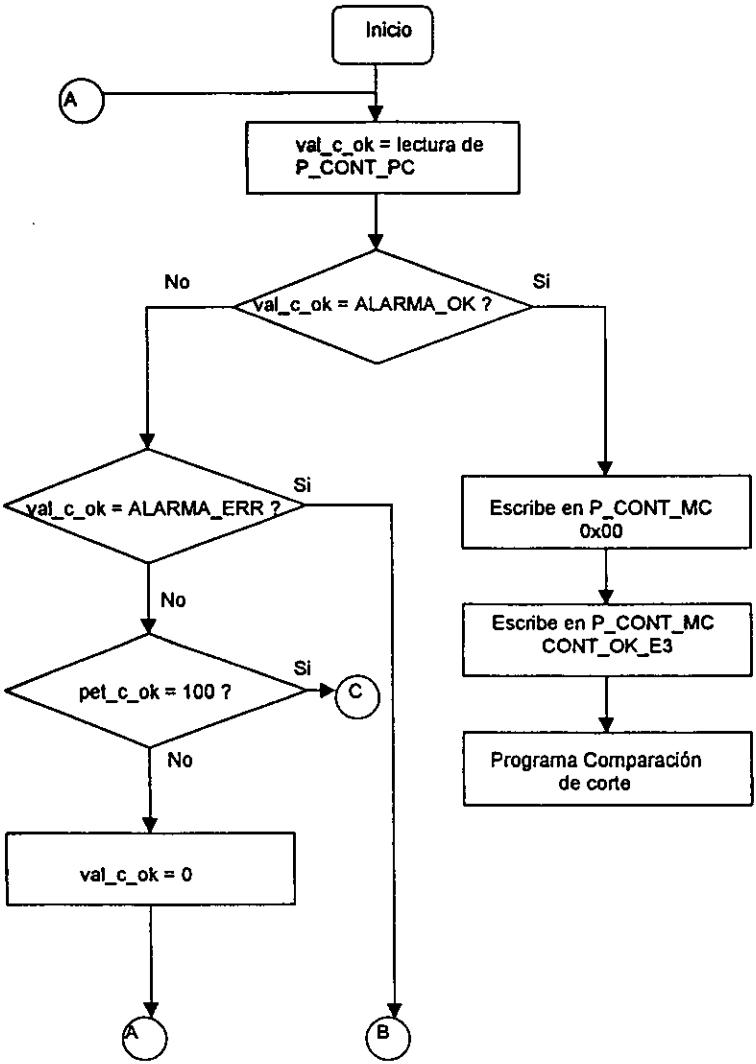


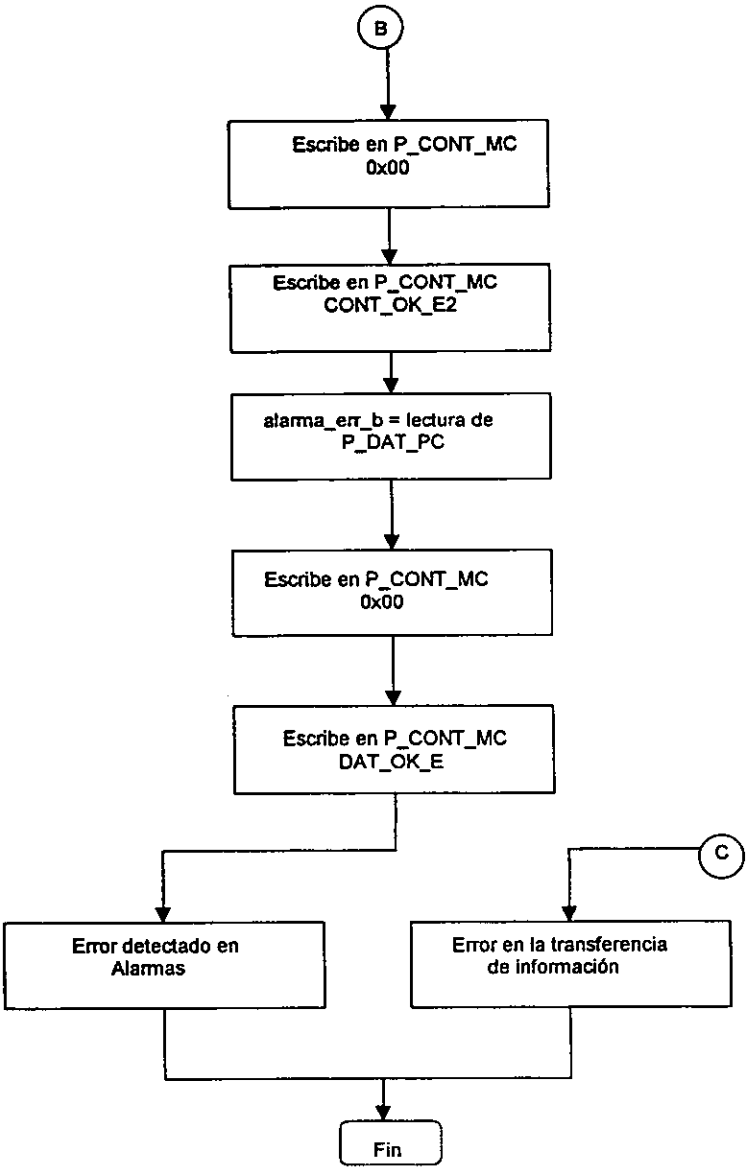
Transferencia del 2o. y 3er. byte de posición



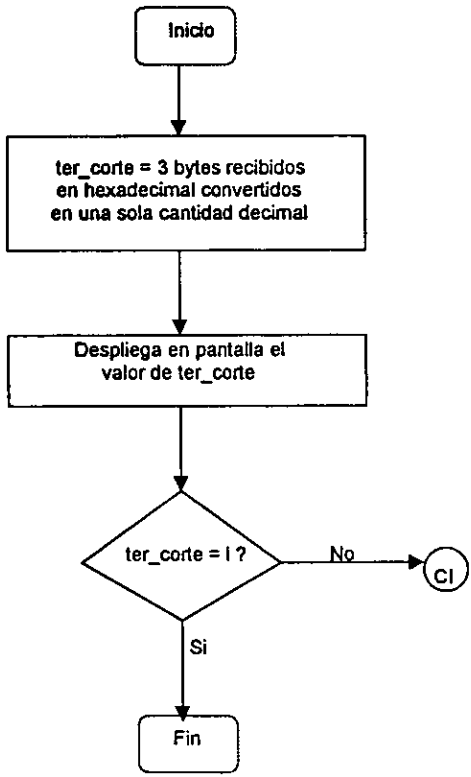

D
val_c_ok = VEL_OK ?
Si
No
Escribe en P_CONT_MC
0x00
Escribe en P_CONT_MC
CONT_OK_E3
Programa Alarmas
val_c_ok = VEL_ERR ?
No
Si
Escribe en P_CONT_MC
0x00
Escribe en P_CONT_MC
CONT_OK_E2
val_err_b3 = lectura de
P_DAT_PC
Escribe en P_CONT_MC
0x00
Escribe en P_CONT_MC
DAT_OK_E
A
pet_c_ok = 100 ?
Si
A
No
val_c_ok = 0
C

Programa Alarmas

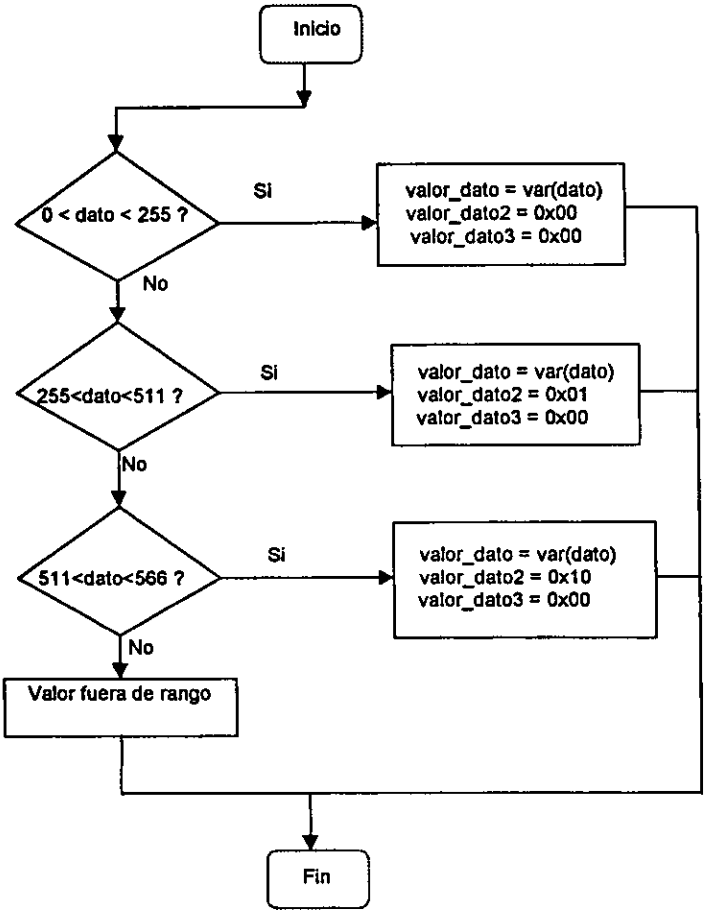



B
Escribe en P_CONT_MC
0x00
Escribe en P_CONT_MC
CONT_OK_E2
alarma_err_b = lectura de
P_DAT_PC
Escribe en P_CONT_MC
0x00
Escribe en P_CONT_MC
DAT_OK_E
C
Error detectado en
Alarmas
Error en la transferencia
de información
Fin

Programa Comparación Corte



Programa Conversión Decimal - Hexadecimal



Apéndice B

Diagrama de Señales del Protocolo de Comunicación

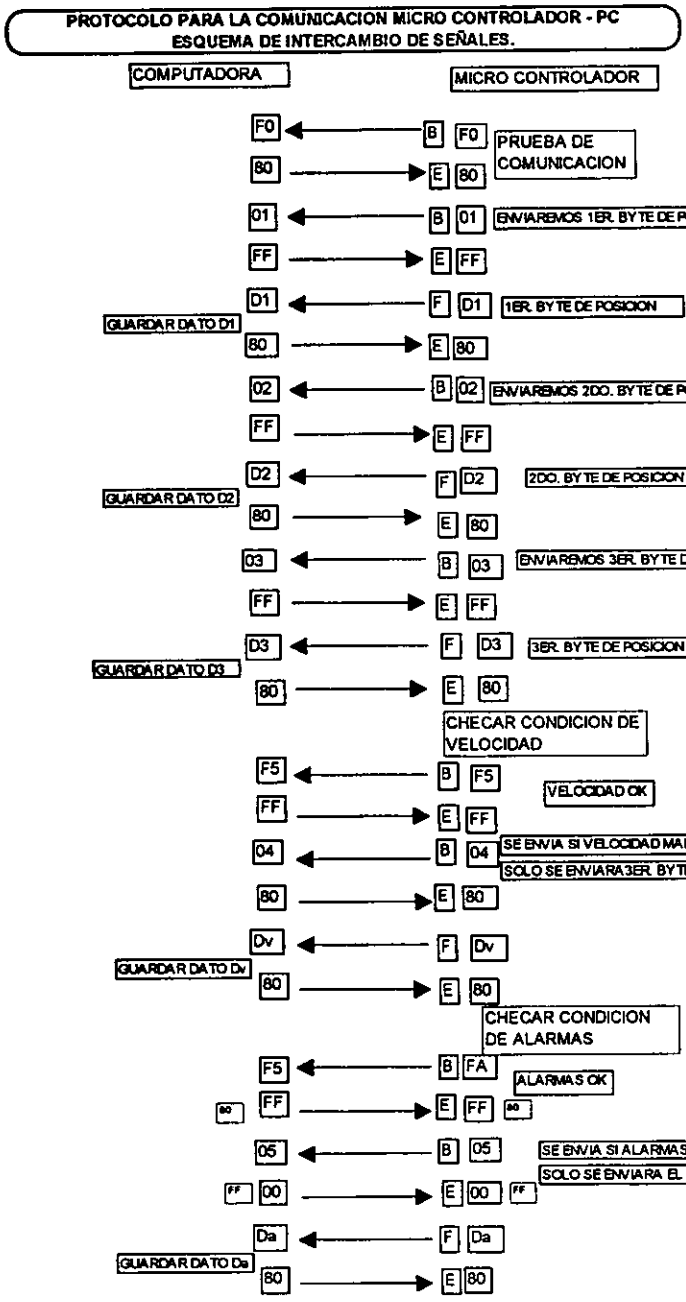


Diagrama de Flujo del protocolo de comunicación del Microcontrolador a la PC

Apéndice C

Programas del Microcontrolador

```
*************************************************************
*PROGRAMA QUE OBTIENE PULSOS PARA VELOCIDAD VIA FRECUENCIA*
*************************************************************
*ELABORO EMCM,RGC,DML,MSR,RVH/971121*980319 PROG.OK PULFV11*
*************************************************************
DEFSEG FREC,START=$B600
SEG FREC

PORTA     EQU $00
TIC3      EQU $14
TCTL2     EQU $21
TFLG1     EQU $23
TFLG2     EQU $25
VAR1      EQU $C0       ;AQUI CARGAMOS 1ER. DATO DE CAPTURA
VAR2      EQU $C2       ;AQUI CARGAMOS 2DO. DATO DE CAPTURA
RESULBM1  EQU $C4
RESULBM2  EQU $C5       ;AQUI RESPALDAMOS EL DATO DE VALORB-2
RESUL0    EQU $C6
RESUL2    EQU $C7       ;C7 C8
BMENOS1   EQU $C9       ;UNICAMENTE PARA GUARDAR VALOR 01 PARA RESTA SOBRE B
VALORBM   EQU $CB       ;PARA GUARDAR VALOR MAXIMO DE TOF'S B=31
VALORB    EQU $CC       ;SE GUARDA VALOR INC B PARA RESPALDO
TOTPUL0   EQU $CD       ;ALT   BAJA
TOTPUL    EQU $CE       ;CD,CE,CF DIRECCIONES PARA EL DATO RESULTANTE DE PULSOS
VAR3      EQU $D1
VAR4      EQU $D3
VTOF      EQU $D5       ;SOLO PARA RESPALDAR EL TOF EN LA PRIMER LECTURA
VTOF2     EQU $D6       ;TOT TOF
VALVEL1   EQU $E1
VALVEL2   EQU $E2
VALVEL3   EQU $E3
VALVEL4   EQU $E4
VALVEL5   EQU $E5
VALVEL6   EQU $E6
VALVEL7   EQU $E7
VALVEL8   EQU $E8

INICIO:LDY #$0100
       LDX #$1000
       LDAA #$00
       STAA RESULBM1,Y
       STAA VTOF,Y
       LDAA #$01
       STAA BMENOS1,Y
       LDAA #$1F          ;PARA ALMACENAR VALOR DE B=31
       STAA VALORBM,Y
       LDAA #$01
       STAA TCTL2,X       ;DEFINE PA0 COMO ENTRADA DE CAPTURA CON FLANCO +
       BSET TFLG1,X,$01   ;APAGA LA BANDERA IC3F
REVISAR: BSET TFLG2,X,$80 ;APAGA LA BANDERA DE TOF
       BITA TFLG1,X,$01   ;PREGUNTA SI SE ACTIVA LA BANDERA IC3F
       BEQ  REVISAR
       LDD  TIC3,X        ;LEEMOS PRIMER DATO DE CAPTURA
       BSET TFLG1,X,$01   ;APAGA LA BANDERA IC3F
       STD  VAR1,Y        ;ALMACENAMOS PRIMER DATO CAPTURADO
       LDAA #$80          ;PARA RESPALDAR TOF SI SE ACTIVA SU BANDERA
```

```
        BITA TFLG2,X,$80
        BEQ  RTOF0          ;SI TOF=0 NO LO RESPALDA
        LDAA VTOF,Y
        INCA
        STAA VTOF,Y
        BSET TFLG2,X,$80    ;APAGA LA BANDERA DE TOF
RTOF0:  LDAB #$00
        STAB VALORB,Y       ;VALOR INICIAL DE B=0
REVISAR2: LDAA #$01
        BITA TFLG1,X,$01    ;PREGUNTAR SI SE ACTIVA LA BANDERA IC3F
        BEQ  REVISAR1
        BRA  REVISAR3
REVISAR1: LDAA #$80
        BITA TFLG2,X,$80    ;PREGUNTA SI SE ACTIVA LA BANDERA DE TOF
        BEQ  REVISAR2       ;REGRESA SI TOF=0
        BSET TFLG2,X,$80    ;BORRA TOF
        INCB
        STAB VALORB,Y       ;RESPALDAMOS VALOR DE B= NUMEROS DE TOF
        SUBB VALORBM,Y
        BEQ  ERROR          ;CHECAMOS SI B=TOF=31
        LDAB VALORB,Y
        BRA  REVISAR2       ;NO SE HA LLEGADO A 31 ES DECIR >2000000
REVISAR3: LDD TIC3,X        ;LEEMOS SEGUNDO DATO DE CAPTURA
        STD  VAR2,Y         ;ALMACENAMOS SEGUNDO DATO CAPTURADO
        LDAB VALORB,Y
        BEQ  B0TOF1         ;VA A B0TOF1  SI VALORB=0
        BRA  BMENUNO        ;VA A BMENUNO SI VALORB>0 INDICANDO HUBO + DE UN TOF
B0TOF1:LDAA VTOF,Y          ;AQUI VERIFICAMOS EL TOF DE LAS DOS 1RAS. LECTURAS
        BEQ  RUTINAA        ;SI B=0yTOF=0 SE VA A RUTINA VAR2-VAR1
        BRA  REVISA4        ;SI B=0 Y SOLO HUBO UN TOF=1 VA A REVISA4
BMENUNO:LDAB VALORB,Y       ;EN BMENUNO REALIZAMOS((FFFF-VAR1)+VALORB*FFFF+VAR2+1)
        ADDB VTOF,Y         ;***
        STAB VTOF2,Y        ;TOT TOF
        SUBB BMENOS1,Y      ;AQUI TOTPULSOS=VAR3+VAR4+VAR2+1
        SUBB BMENOS1,Y      ;***
        STAB RESULBM2,Y     ;ALMACENAMOS DATO DE (VALORB+VTOF)-2
        LDD  #$FFFF
        SUBD VAR1,Y
        STD  VAR3,Y
        LDD  #$FFFF         ;A PARTIR DE AQUI PROCESO DE B*FFFF=VAR4
        SUBD RESULBM1,Y
        STD  RESUL2,Y       ;C7,C8
        LDAB RESULBM2,Y
        STAB RESUL0,Y       ;RESULT PRELIMINAR DE PULSOS EN C6,C7,C8
        LDD  RESUL2,Y
        ADDD VAR3,Y
        BCS  SUMA1          ;SI CARRI=1 VAMOS A SUMA1
        BRA  FIN3
SUMA1:  STD  TOTPUL,Y
        LDAA RESUL0,Y
        INCA                ;AQUI YA NO HABRA CARRI 3ER. BAYTE
        STAA RESUL0,Y
        STAA TOTPUL0,Y      ;EL DATO QUEDA EN CD,CE,CF
        LDD  TOTPUL,Y       ;********
        ADDD VAR2,Y
        BCS  SUMA11
        STD  TOTPUL,Y
```

```
        BRA  SUMAFIN
SUMA11: STD  TOTPUL,Y
        LDAA TOTPUL0,Y
        INCA
        STAA TOTPUL0,Y
SUMAFIN: BRA  COMP1
ERROR:  BRA  ERROR1          ;SOLO POR EL OFSET
RUTINAA: BRA  RUTINA         ;SOLO POR EL OFSET
FIN3:   STD  TOTPUL,Y        ;PARTE MEDIA Y BAJA
        LDAA RESUL0,Y
        STAA TOTPUL0,Y       ;EL DATO QUEDA EN CD,CE,CF
        LDD  TOTPUL,Y
        ADDD VAR2,Y
        BCS  FIN33
        STD  TOTPUL,Y
        BRA  FINFIN3
REVISA4: BRA  REVISA41       ;SOLO POR EL OFSET
FIN33:  STD  TOTPUL,Y
        LDAA TOTPUL0,Y
        INCA
        STAA TOTPUL0,Y
FINFIN3: BRA  COMP1
RUTINA: LDD  VAR2,Y          ;AQUI SE REALIZA  VAR2-VAR1  VALORB=0 VTOF=0
        SUBD VAR1,Y          ;VAR2-VAR1 CONTADOR NO REBASO FFFF TOF=0
FIN:    STD  TOTPUL,Y        ;ALMACENAMOS TOT DE PULSOS PARTE MEDIA Y BAJA 0EY0F
        LDAA #$00
        STAA TOTPUL0,Y      ;CARGAMOS CON 00 TOT PULSOS PARTE MAS SIGNIFICATIVA 0D
        BRA  COMP1
COMP1:  LDAA TOTPUL0,Y       ;VALOR DE VELOCIDAD RUTINA DE COMPARACION INICIO
        SUBA VALVEL1,Y
        BEQ  COMP2
        BCC  SIGCOMP         ;VAMOS A COMPARAR CONTRA REF VEL MAXIMA
        BRA  VELMAL
COMP2:  LDD  TOTPUL,Y        ;000E Y 000F
        SUBD VALVEL2,Y
        BEQ  VELBIEN
        BCC  SIGCOMP         ;VAMOS A COMPARAR CONTRA REF VEL MAXIMA
        BRA  VELMAL
SIGCOMP: LDAA TOTPUL0,Y
        SUBA VALVEL7,Y
        BEQ  COMP3
        BCC  VELMAL          ;EL DATO DE PULSOS ES MAYOR QUE REF VEL MAX
        BRA  VELBIEN         ;EL DATO DE PULSOS ES MENOR QUE REF VEL MAX
COMP3:  LDD  TOTPUL,Y
        SUBD VALVEL8,Y
        BEQ  VELBIEN
        BCC  VELMAL
        BRA  VELBIEN
ERROR1: LDD  #$0000          ;8482 AQUI SE FABRICA EL 200000 HEX = A ERROR
        STD  TOTPUL,Y
        LDAA #$20            ;1E
        STAA TOTPUL0,Y       ;PARTE MAS SIGNIFICATIVA
        BRA  VELMAL
REVISA41: LDD  #$FFFF        ;RUTINA CUANDO TOF=1 Y B=0 (FFFF-VAR1)+VAR2+1
        SUBD VAR1,Y
        ADDD VAR2,Y
        STD  VAR4,Y
```

```
        BCS  EXICARRY       ;CHECAMOS SI (FFFF-VAR1)+VAR2 PRODUCE CARRY
        LDAA #$00
        STAA TOTPUL0,Y
        BRA  CONTINUA
EXICARRY:LDAA #$01
        STAA TOTPUL0,Y
CONTINUA:LDD  VAR4,Y
        ADDD #$0001
        STD  VAR4,Y
        BCS  EXICARRY2
        LDD  VAR4,Y
        STD  TOTPUL,Y
        BRA  COMP1
EXICARRY2:LDAA #$01
        STAA TOTPUL0,Y
        LDD  VAR4,Y
        STD  TOTPUL,Y
        BRA  COMP1          ;AQUI TERMINA (FFFF-VAR1)+VAR2+1 CONTADOR REVASO FFFF
VELBIEN: BRA  ROBERTO       ;VELOCIDAD CORRECTA CONTINUA PROCESO
VELMAL:  LDAA #$80          ;VELOCIDAD FUERA DE RANGO
        STAA $1000      ;POR PUERTO 'A BIT 7 =1' MANDAMOS UN '1'PARA PARAR PROCESO
        LDAA #$20
        STAA TOTPUL0,Y
        LDD  #$0000
        STD  TOTPUL,Y
        BRA  FINTOT2
ROBERTO: LDAA #$00
        BRA ROBERTO
FINTOT2: LDAB #$00
        BRA FINTOT2
        END
```

```
*****************************************************
*PROGRAMA PARA COMUNICAR EL MICROPROCESADOR Y LA PC*
*              PC-MICRO VERSION 6                   *
*****************************************************

        DEFSEG COMUN, START=$B600
        SEG COMUN

PORTF     EQU  $03
PORTE     EQU  $0A
PORTB     EQU  $04
PORTG     EQU  $03
PORTA     EQU  $00
TIME      EQU  $20
DATO      EQU  $22
DATOS     EQU  $00
PACTL     EQU  $26
DAVE      EQU  $10
DAVE1     EQU  $11

          LDX  #$1000
          LDY  #$0100
          LDS  #$FF
          LDAA #$80
          STAA PACTL,X

          LDAA #$00
          STAA TIME,Y        ;LOC. MEM. QUE CONTROLA EL TIEMPO DE ESPERA
          STAA DATO,Y        ;LOC. MEM. QUE INDICA EL CODIGO DEL DATO A RECIBIR
MAS:      LDAA PORTE,X       ;LEE LO QUE LE ENVIA LA PC
          STAA DAVE,Y
          CMPA #$F0
          BEQ  WAIT          ;SI LA PALABRA ES CORRECTA BRINCA, SI NO SIGUE
          BRA  MAS
WAIT:     BCLR TIME,Y,$FF
          LDAA #$80
          STAA PORTB,X       ;CONFIRMACION DE COMUNICACION OK Y ESPERA DATO
          STAA DAVE1,Y
          BRA  MAS
SIGUE:    INC  DATO,Y
OTRO:     LDAA PORTE,X
          CMPA DATO,Y        ;VERIFICA QUE LA PALABRA DE CONTROL SEA CORRECTA
          BEQ  ROMPE         ;SI LA PALABRA ES CORRECTA BRINCA
          BSR  RUTIM         ;ENTRA A RUTINA DE TIEMPO SI LA PALABRA ES INCORRECTA
          BRA  OTRO
ROMPE:    BCLR TIME,Y,$FF
          LDAA #$F0
          STAA PORTB,X       ;CONTESTA QUE PUEDE EMPEZAR A ENVIAR DATOS
          BSR  DTOS          ;SE GUARDAN LOS DATOS
          LDAA #$F1
          STAA PORTB,X       ;CONTESTA QUE FUE LEIDO EL DATO
          LDAB DATO,Y
          CMPB #$0F
          BCS  SIGUE
          BRA  FIN
```

```
ESPERA:   BRCLR PORTE,X,$80,ESPERAR     ;COMPRUEBA SI EXISTE UN 1 EN b7
          BRCLR PORTE,X,$7F,OKK         ;OBSERVA QUE HAY CEROS EN LOS OTROS BITS
ESPERAR:  BSR  RUTIM             ;SI NO RECIBE EL CODIGO SE VA A RUTINA
          BRA  ESPERA
OKK:      BCLR TIME,Y,$FF        ;LIMPIA LOC. MEM. SI EL CODIGO ES OK
          RTS

RUTIM:    INC  TIME,Y
          LDAB TIME,Y
          CMPB #$FF
          BLO  OK2
          BSET PORTA,X,$80       ;ENVIA UN PULSO AL SERVO DETENIENDOO EL MOTOR
          LDAA #$CC              ;ERROR DE COMUNICACION
          STAA PORTB,X           ;ENVIA CONDICION DE ERROR
          BRA  FIN
OK2:      RTS

DTOS:     LDAA PORTG,X           ;LEE EL DATO DEL PUERTO
          STAA DATOS,Y           ;ALMACENA EL DATO
          INY                 ;PASA A LA SIGUIENTE LOC. MEM. DONDE RECIBE OTRO DATO
          RTS
FIN:      END
```

```
*****************************************************
*PROGRAMA PARA COMUNICAR EL MICROPROCESADOR Y LA PC*
*               MICRO-PC VERSION 1                  *
*****************************************************

DEFSEG COMUN, START=$0100
SEG COMUN

PORTF     EQU  $03
PORTE     EQU  $0A
PORTB     EQU  $04
PORTG     EQU  $03
PORTA     EQU  $00
TIME      EQU  $20
NUMDATO   EQU  $22
INFO      EQU  $24
DATOS     EQU  $00
PACTL     EQU  $26
LOC       EQU  $28


          LDX  #$1000
          LDY  #$0000
          LDS  #$FF
          LDAA #$80
          STAA PACTL,X

          LDAA #$00
          STAA TIME          ;LOC. MEM. QUE CONTROLA EL TIEMPO DE ESPERA
          STAA NUMDATO       ;LOC. MEM. QUE INDICA EL CODIGO DEL DATO A RECIBIR
          LDAA #$F0
          STAA PORTB,X       ;PRUEBA DE COMUNICACION, ESPERA RESPUESTA

SIGUE:    LDAA #$80
          BSR  RESP1
          INC  NUMDATO
          LDAA NUMDATO
          STAA PORTB,X
          LDAA #$FF
          BSR  RESP1
DTOS:     LDAA DATOS,Y       ;LEE EL DATO DE LA LOC. MEM.
          STAA PORTF,X       ;ENVIA EL DATO
          LDAA #$03
          CMPA DATOS,Y
          BEQ  GO
          INY                ;PASA A LA SIGUIENTE LOC. MEM. DONDE RECIBE OTRO DATO
          BRA  SIGUE


RESP1:    STAA INFO
          LDAA PORTE,X       ;LEE LO QUE LE ENVIA LA PC
          CMPA INFO
          BEQ  RETURN1       ;SI LA PALABRA ES CORRECTA BRINCA, SI NO SIGUE
          BSR  RUTIM         ;ENTRA A RUTINA DE TIEMPO
          BRA  RESP1
```

```
RETURN1:  BCLR TIME,$FF
          RTS


RUTIM:    INC  TIME
          LDAB TIME
          CMPB #$5B
          BLO  OK2
          BSET PORTA,X,$80      ;ENVIA UN PULSO AL SERVO DETENIENDOO EL MOTOR
          LDAA #$CC             ;ERROR DE COMUNICACION
          STAA PORTB,X          ;ENVIA CONDICION DE ERROR
          BRA  FIN
OK2:      RTS

GO:       LDAA LOC,X            ;LOC.MEM. QUE CONTIENE EL 3er BYTE DE VELOCIDAD
          CMPA #$22
          BEQ  ERROR            ;SI EXISTE ERROR EN LA VELOCIDAD VA A ERROR
          LDAA #$F5
          STAA PORTB,X          ;ENVIA MENSAJE DE QUE LA VELOCIDAD ESTA CORRECTA
          LDAA #$FF
          BSR  RESP1            ;ESPERA LA RESPUESTA DE LA PC
```

```
*********************************************
*PROGRAMA PARA DETECTAR LA ACTIVACION DE   *
*              LOS SWITCHES                 *
*********************************************


          BRA  FIN

ERROR:    INC  INFO
          LDAA INFO
          STAA PORTB,X
          LDAA #$80
          BSR  RESP1
          LDAA LOC,X
          STAA PORTB,X
          LDAA #$80
          BSR  RESP1

FIN       END
```

```
*****************************************
*PROGRAMA PARA HACER UN CONTADOR DE PULSOS*
*                PRUEBA-9               *
*****************************************
DEFSEG PRUEBA, START=$B600
SEG PRUEBA

TCTL2   EQU   $21
TMSK1   EQU   $22
TFLG1   EQU   $23
TFLG2   EQU   $25
PACTL   EQU   $26
PACNT   EQU   $27
PORTE   EQU   $0A

LONG    EQU   $4A       ;LOCALIDAD DE MEMORIA,PARTE BAJA,DE DONDE SE TOMA
                        ;LA DISTANCIA A RECORRER
LONG1   EQU   $4B       ;LOCALIDAD DE MEMORIA,PARTE MEDIA,DE DONDE SE TOMA
                        ;LA DISTANCIA A RECORRER
LONG2   EQU   $4C       ;LOCALIDAD DE MEMORIA,PARTE ALTA,DE DONDE SE TOMA
                        ;LA DISTANCIA A RECORRER
AVANCE  EQU   $21       ;LOCALIDAD DE MEMORIA,PARTE BAJA,DE DONDE SE DEJA
                        ;LA DISTANCIA RECORRIDA
AVANCE1 EQU   $22       ;LOCALIDAD DE MEMORIA,PARTE MEDIA,DE DONDE SE DEJA
                        ;LA DISTANCIA RECORRIDA
AVANCE2 EQU   $23       ;LOCALIDAD DE MEMORIA,PARTE ALTA,DE DONDE SE DEJA
                        ;LA DISTANCIA RECORRIDA

        LDY   #$0100
        LDX   #$1000
INICIO  LDAA  #$0F        ;HABILITA LA CAPTURA DE CUALQUIER FLANCO
        STAA  TCTL2,X     ;EN PA0 Y PA1
        BCLR  TFLG1,X,$FC ;BORRA BANDERAS IC2F E IC3F
        BSET  TMSK1,X,$03 ;HABILITA LAS INTERRUPCIONES IC2I E IC3I
        BSET  PACTL,X,$50 ;HABILITA ACUMULADOR DE PULSOS
        CLR   PACNT,X     ;LIMPIA EL ACUMULADOR DE PULSOS
        BSET  TFLG2,X,$20 ;APAGA BANDERA DE SOBRE FLUJO
        CLI               ;LIMPIA MASCARA DE INTERRUPCIÓN

        LDD   #$0000
        STAA  AVANCE,Y
        STAA  AVANCE1,Y
        BCLR  AVANCE2,Y,$3F
VALALTO LDAA  PACNT,X
        STAA  AVANCE,Y          ;CARGA EL ACU.PULSOS A UNA LOC.MEM.
        BRCLR TFLG2,X,$20,SIGUE ;SE ACTIVO LA BANDERA DE SOBRE FLUJO (PAOVF=1)
        BRA   OTRA              ;SI PAOVF=1 BRINCA
INCRE   BSET  TFLG2,X,$20
        INC   AVANCE2,Y         ;INCREMENTA AVANCE2 = AVANCE2 + 1
COMPAR  LDAB  AVANCE2,Y
        ANDB  #$3F
        CMPB  LONG2,Y           ;COMPARA AVANCE2 CON LONG2
        BCS   VALALTO           ;SI AVANCE2 = LONG2 CONTINUA, SI NO, BRINCA
        LDAB  AVANCE1,Y
        CMPB  LONG1,Y
        BCS   VALALTO           ;SI AVANCE1 = LONG1 CONTINUA, SI NO, BRINCA
```

```
        CMPA  LONG,Y
        BCS   VALALTO            ;SI AVANCE = LONG CONTINUA, SI NO, BRINCA
MAS     BRCLR PORTE,X,$01,NO     ;EL PORTE PREGUNA SI VIENE OTRO DATO PE0=1 ES SI
        BRA   INICIO
NO      BRCLR PORTE,X,$02,MAS    ;CON PE1=1 TERMINA EL PROGRAMA
        BRA   FIN
OTRA    INC   AVANCE1,Y          ;INCREMENTA AVANCE1 = AVANCE1 + 1
        BRCLR AVANCE1,Y,$FF,INCRE  ;SE TIENEN QUE CUMPLIR LAS DOS PARA INCREMENTAR
                                   ;A AVANCE2
        BSET  TFLG2,X,$20
SIGUE   CMPA  LONG,Y              ;COMPARA AVANCE CON LONG
        BCS   VALALTO             ;SI AVANCE < LONG BRINCA
        LDAB  AVANCE1,Y
        CMPB  LONG1,Y             ;COMPARA AVANCE1 CON LONG1
        BCS   VALALTO             ;SI AVANCE1 = LONG1 CONTINUA, SI NO, BRINCA
        BRA   COMPAR


INTER   SEI
        BCLR  TCTL2,X,$0F         ;SE DESACTIVAN LAS ENTRADAS DE CAPTURA
        LDX   #$1000
        BRCLR TFLG1,X,$01,OK ;SI IC2F=1 SENTIDO INVERSO, SI IC3F=1 SENTIDO HORARIO
        BCLR  AVANCE2,Y,$40       ;SE BORRA EL BIT DE SENTIDO ANTERIOR
        BSET  AVANCE2,Y,$80       ;BIT7=1, SENTIDO HORARIO
        BCLR  TCTL2,X,$0F         ;SE DESACTIVAN LAS ENTRADAS DE CAPTURA
        BRA   SET
OK      BCLR  AVANCE2,Y,$80       ;SE BORRA EL BIT DE SENTIDO ANTERIOR
        BSET  AVANCE2,Y,$40       ;BIT6=1, SENTIDO INVERSO
SET     BSET  TFLG1,X,$03         ;BORRA BANDERAS IC2F E IC3F
        RTI
FIN     END
```

```
*******************************************
*PROGRAMA PARA HACER UN CONTADOR DE PULSOS*
*                 SWITCH                  *
*******************************************
DEFSEG SWITCH, START=$B600
SEG SWITCH

TCTL2   EQU   $21
TMSK1   EQU   $22
TFLG1   EQU   $23
TFLG2   EQU   $25
PACTL   EQU   $26
PACNT   EQU   $27
PORTB   EQU   $04


        LDX   #$1000
        LDAA  #$0A              ;CAPTURA DE FLANCO "-"
        STAA  TCTL2,X           ;EN PA0 Y PA1 SWITCHES X1 Y X2
        BCLR  TFLG1,X,$FC       ;BORRA BANDERAS IC2F E IC3F
OTRA    BRCLR TFLG1,X,$03,JUMP1 ;PRUEBA SI PA0=PA1=0
        BRCLR TFLG1,X,$02,JUMP1 ;PRUEBA SI PA0=0 PA1=1
        BRCLR TFLG1,X,$01,JUMP2 ;PRUEBA SI PA0=1 PA1=0
ERROR   LDAA  #$F1              ;ERROR FATAL (PA0=PA1=1)
        STAA  PORTB,X           ;ENVIA INFORMACION.
        BCLR  TFLG1,X,$FC       ;BORRA BANDERAS IC2F E IC3F
        BRA   FIN


JUMP1   LDAA  #$D1              ;MOVER CABEZAL A LA IZQUIERDA
        STAA  PORTB,X           ;SW1=SW2=0 à SW1=0 Y SW2=1
        BCLR  TFLG1,X,$FC       ;BORRA BANDERAS IC2F E IC3F
        BRA   OTRA

JUMP2   LDAA  #$D3              ;OK, CABEZAL EN EL ORIGEN.
        STAA  PORTB,X           ;SW1=1 Y SW2=0
        BCLR  TFLG1,X,$FC       ;BORRA BANDERAS IC2F E IC3F
        LDAA  #$D2              ;MOVER CABEZAL A LA DERECHA
        STAA  PORTB,X
        LDAB  #$00
REP     INCB
        CMPB  #$FF              ;MOTOR ENCENDIDO DURANTE 1 SEGUNDO
        BLO   REP
REP1    INCB
        CMPB  #$FF              ;MOTOR ENCENDIDO DURANTE 1 SEGUNDO
        BLO   REP1
        LDAA  #$D1              ;MOVER CABEZAL A LA IZQUIERDA
        STAA  PORTB,X
MAS     BRCLR TFLG1,X,$02,MAS   ;PRUEBA SI PA0=1 PA1=0
FIN     END
```

```
*******************************************
*PROGRAMA PARA DETECTAR LA ACTIVACION DE  *
*            LOS SWITCHES                 *
*******************************************
DEFSEG SWITCHES, START=$B600
SEG SWITCHES

PORTC   EQU  $05
PORTF   EQU  $03
PORTB   EQU  $04
DDRC    EQU  $07


        LDX   #$1000
        LDAB  #$00
        STAB  DDRC,X              ;ACTIVA COMO ENTRADAS AL PUERTO C

        BRCLR PORTC,X,$FF         ;CHECA
        LDAA  #$FA
        STAA  PORTB,X             ;ENVIA INFORMACION.
        BRA   FIN
CHECA   BSET  PORTA,X,$40
        LDAA  #$05
        STAA  PORTB,X

        BRCLR PORTC,X,$01,ERROR1 ;SE ENVIA $FE
        BRCLR PORTC,X,$02,ERROR2 ;SE ENVIA $FD
        BRCLR PORTC,X,$04,ERROR3 ;SE ENVIA $FB
        BRCLR PORTC,X,$08,ERROR4 ;SE ENVIA $F7
        BRCLR PORTC,X,$10,ERROR5 ;SE ENVIA $EF
        BRCLR PORTC,X,$20,ERROR6 ;SE ENVIA $DF
        BRCLR PORTC,X,$40,ERROR7 ;SE ENVIA $BF

ERROR1  BRCL  PORTC,X,$FE,ERROR8 ;DOS ALARMAS ACTIVADAS
        BSET  PORTF,X,$FE
        BRA   FIN

ERROR2  BRCLR PORTC,X,$FD,ERROR8 ;DOS ALARMAS ACTIVADAS
        BSET  PORTF,X,$FD
        BRA   FIN

ERROR3  BRCLR PORTC,X,$FB,ERROR8 ;DOS ALARMAS ACTIVADAS
        BSET  PORTF,X,$FB
        BRA   FIN

ERROR4  BRCLR PORTC,X,$F7,ERROR8 ;DOS ALARMAS ACTIVADAS
        BSET  PORTF,X,$F7
        BRA   FIN

ERROR5  BRCLR PORTC,X,$EF,ERROR8 ;DOS ALARMAS ACTIVADAS
        BSET  PORTF,X,$EF
        BRA   FIN

ERROR6  BRCLR PORTC,X,$DF,ERROR8 ;DOS ALARMAS ACTIVADAS
        BSET  PORTF,X,$DF
        BRA   FIN
```

```
ERROR7   BRCLR PORTC,X,$BF,ERROR8 ;DOS ALARMAS ACTIVADAS
         BSET  PORTF,X,$BF
         BRA   FIN

ERROR8   BSET  PORTF,X,$DD        ;ERROR GRAVE, ACTIVACION DE MAS DE UNA ALARMA
         BRA   FIN
FIN      END
```

Apéndice D

Programa en Lenguaje C

```
/*********************************************************************/
/*****************       PROGRAMA PARA LA TESIS        ***************/
/*****************  DISENO DEL CONTROL NUMERICO (CNC)  ***************/
/*****************  DE UNA MAQUINA DE CORTE (EDM) POR  ***************/
/*****************       DESCARGAS ELECTRICAS          ***************/
/*********************************************************************/
/*** Integrantes:                                                  ***/
/***                E. Mario Carreon Mata                          ***/
/***                Roberto Gonzalez Campos                        ***/
/***                David Mendez Lopez                             ***/
/***                Mauricio Sanchez Resendiz                      ***/
/***                Ricardo Villanueva Hernandez                   ***/
/*** Director:     Miguel Angel Cruz Leon                          ***/
/*********************************************************************/


/****************************** LIBRERIAS ****************************/

#include <alloc.h>
#include <conio.h>
#include <dos.h>
#include <graphics.h>
#include <math.h>
#include <mem.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


/********************** DEFINICION DE VALORES ************************/

#define call_mouse int86(0x33,&inreg,&outreg)
#define FALSE      0
#define TRUE       1
#define OFF        0
#define ON         1
#define INC        15
#define ESC 0x1b


/********** DEFINICION VALORES DE PROTOCOLO DE COMUNICACION *********/

#define P_CONT_MC       0xF000   //Escritura  PC-MC
#define P_DAT_MC        0xF001   //Escritura  PC-MC
#define P_CONT_PC       0xF005   //Lectura    MC-PC
#define P_DAT_PC        0xF006   //Lectura    MC-PC
#define CONT_OK_E1      0x00F0   /*11110000*/
#define CONT_OK_E2      0x0080   /*10000000*/
#define CONT_OK_E3      0x00FF   /*11111111*/
#define CONT_OK_L1      0x0080   /*10000000*/
#define CONT_OK_L2      0x00F0   /*11110000*/
#define DAT_OK_L        0x00F1   /*11110001*/
#define DAT_OK_E        0x0080   /*10000000*/
#define V_CONT_P_B1     0x0001   /*00000001*/
#define V_CONT_P_B2     0x0002   /*00000010*/
```

```
#define V_CONT_P_B3     0x0003      /*00000011*/
#define V_CONT_R_B1     0x0004      /*00000100*/
#define V_CONT_R_B2     0x0005      /*00000101*/
#define V_CONT_R_B3     0x0006      /*00000110*/
#define V_CONT_N_B1     0x0007      /*00000111*/
#define V_CONT_N_B2     0x0008      /*00001000*/
#define V_CONT_N_B3     0x0009      /*00001001*/
#define D_CONT_B1       0x000A      /*00001010*/
#define D_CONT_B2       0x000B      /*00001011*/
#define D_CONT_B3       0x000C      /*00001100*/
#define X0_CONT_B1      0x000D      /*00001101*/
#define X0_CONT_B2      0x000E      /*00001110*/
#define X0_CONT_B3      0x000F      /*00001111*/
#define P_CONT_B1       0x0001      /*00000001*/
#define P_CONT_B2       0x0002      /*00000010*/
#define P_CONT_B3       0x0003      /*00000011*/
#define VEL_OK          0x00F5      /*11110101*/
#define VEL_ERR         0x0004      /*00000100*/
#define ALARMA_OK       0x00FA      /*11111010*/
#define ALARMA_ERR      0x0005      /*00000101*/


/********************** DEFINICION DE TIPOS *************************/

typedef struct
{
  int button_status;
  int button_count;
  int xaxis;
  int yaxis;
} Mstatus;


/********************** ENCABEZADO DE FUNCIONES *********************/

/********** Funciones para obtener interface para el usuario ********/

void  menu_principal(void);
int   menu(int nopciones,int c_barra1,int c_letrero1,int c_barra2,int
c_letrero2,int marco);
int   entra_ventana(int x1,int y1,int x2, int y2);
int   sal_ventana(int x1,int y1,int x2,int y2);
int   far *salva_vent(int x1,int y1,int x2,int y2);
void  restaura_vent(int x,int y,int far *iptr);
void  suelta_boton(void);
void  Mshow(int showstat);
void  Npos(Mstatus posicion);
void  Mini(void);
Mstatus Mpos();


/******** Funciones para la operacion de la maquina de corte ********/

void  pantalla(void);
void  estadisticas(void);
void  cuadro_in(int c_marco,int c_fondo);
void  cuadro_out(void);
```

```
void  lect_info(void);
void  lect_ayuda(void);
void  obtener_datos(void);
int   lee_valor(char *texto);
void  X1_Y1(int x_ini,int y_ini,int pd,int lon);
void  dibujo(int x_ini,int y_ini,int pd,int vl,long lon);
void  valores(void);
void  prueba(void);
void  conv_d_h(void);
void  dec_hex(int dato);
int   hex_dec(int dato1,int dato2,int dato3);
void  cortar(void);
int   pc_mc();
void  mc_pc(void);
void  mens_error(int error);
void  lect_porc(int porc);
void  salir(void);
void  reset(void);


/************************ VARIABLES GLOBALES ************************/

int gdriver=DETECT;     // variables para inicializar el modo grafico
int gmode;
int posx=0, posy=0;
int pen=0, vel=0, vel_p=0, vel_n=0, dis=0;
int error=0;

struct opos{
           int x1;
           int y1;
           int x2;
           int y2;
           char msg[20];
         } posvent[20];

union REGS inreg, outreg;
int Mview;
Mstatus status;
int far *iptr1, *iptr2, *iptr3, *iptr4, *iptr5, *iptr6;
int far *iptr7, *iptr8, *iptr9, *iptr10, *iptr11, *iptr12;
unsigned long tamano;


/******** INICIALIZACION DE VALORES PROTOCOLO COMUNICACION **********/

int valor_dato=0x00, valor_dato2=0x00, valor_dato3=0x00;
int v_dat_p_b1=0x00, v_dat_p_b2=0x00,  v_dat_p_b3=0x00;
int v_dat_r_b1=0x00, v_dat_r_b2=0x00,  v_dat_r_b3=0x00;
int v_dat_n_b1=0x00, v_dat_n_b2=0x00,  v_dat_n_b3=0x00;
int d_dat_b1=0x00,   d_dat_b2=0x00,    d_dat_b3=0x00;
int X0_dat_b1=0x00,  X0_dat_b2=0x00,   X0_dat_b3=0x00;

int p_dat_b1=0x00,   p_dat_b2=0x00,    p_dat_b3=0x00;
int vel_err_b3=0x00, alarma_err_b=0x00;

int val_c_ok=0, val_d_ok=0;
```

```
int pet_c_ok=0, pet_d_ok=0;

/************************ PROGRAMA PRINCIPAL ************************/

void main(void)
{
  initgraph(&gdriver,&gmode,"a:\\borlandc\\bgi");
  Mini();
  menu_principal();
  closegraph();
}

/*********************************************************************/
/***************************** FUNCIONES *****************************/
/*********************************************************************/

/********** FUNCIONES PARA OBTENER INTERFACE PARA EL USUARIO ********/

/****************** FUNCION QUE HACE EL MENU PRINCIPAL **************/

void menu_principal(void)
{
  int i;
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,0x00);
  Mshow(FALSE);
  suelta_boton();
  pantalla();
  do
    {
      posvent[0].x1 = 15;
      posvent[0].y1 = 65;
      posvent[0].x2 = 110;
      posvent[0].y2 = 132;
      strcpy(posvent[0].msg,"Informaci¢n");

      posvent[1].x1 = 15;
      posvent[1].y1 = 132;
      posvent[1].x2 = 110;
      posvent[1].y2 = 199;
      strcpy(posvent[1].msg,"Ayuda");

      posvent[2].x1 = 15;
      posvent[2].y1 = 199;
      posvent[2].x2 = 110;
      posvent[2].y2 = 266;
      strcpy(posvent[2].msg,"Patrones");

      posvent[3].x1 = 15;
      posvent[3].y1 = 266;
      posvent[3].x2 = 110;
      posvent[3].y2 = 333;
      strcpy(posvent[3].msg,"Prueba");

      posvent[4].x1 = 15;
```

```
posvent[4].y1 = 333;
posvent[4].x2 = 110;
posvent[4].y2 = 400;
strcpy(posvent[4].msg,"Cortar");

posvent[5].x1 = 15;
posvent[5].y1 = 400;
posvent[5].x2 = 110;
posvent[5].y2 = 465;
strcpy(posvent[5].msg,"Salir");

posvent[6].x1 = 531;
posvent[6].y1 = 440;
posvent[6].x2 = 625;
posvent[6].y2 = 464;
strcpy(posvent[6].msg,"Reset");

i=menu(8,EGA_BLUE,EGA_YELLOW,EGA_LIGHTGRAY,EGA_BLUE,EGA_WHITE);
switch(i){
        case 1 : // Menu "Informacion"
                 iptr1=salva_vent(140,150,500,300);
                 cuadro_in(EGA_WHITE,EGA_RED);
                 lect_info();
                 restaura_vent(140,150,iptr1);
                 cuadro_out();
                 break;

        case 2 : // Menu "Ayuda"
                 iptr2=salva_vent(140,150,500,300);
                 cuadro_in(EGA_WHITE,EGA_RED);
                 lect_ayuda();
                 restaura_vent(140,150,iptr2);
                 cuadro_out();
                 break;

        case 3 : // Menu "Patrones"
                 iptr3=salva_vent(140,150,500,300);
                 cuadro_in(EGA_WHITE,EGA_RED);
                 obtener_datos();
                 X1_Y1(posx,posy,pen,dis);
                 restaura_vent(140,150,iptr3);
                 delay(700);
                 dibujo(posx,posy,pen,vel,dis);
                 cuadro_out();
                 break;

        case 4 : // Menu "Prueba"
                 prueba();
                 break;

        case 5 : // Menu "Cortar"
                 cortar();
                 break;

        case 6 : // Menu "Salir"
                 iptr6=salva_vent(140,150,500,300);
                 cuadro_in(EGA_WHITE,EGA_RED);
```

```
                    salir();
                    restaura_vent(140,150,iptr6);
                    cuadro_out();
                    break;

             case 7 : // Reinicializar
                    iptr7=salva_vent(140,150,500,300);
                    cuadro_in(EGA_WHITE,EGA_RED);
                    reset();
                    restaura_vent(140,150,iptr7);
                    cuadro_out();
                    break;

             case 8 : //Default
                    break;
             }
    }
  while(i<8);
}


/******** FUNCION PARA CONSTRUIR LOS MENUS DE LAS VENTANAS *********/

int menu(int nopciones,int c_barra1,int c_letrero1,int c_barra2,int c_letrero2,int
marco)
{
  int i;
  int boton=0;
  settextjustify(1,1);
  settextstyle(SMALL_FONT,HORIZ_DIR,5);
  setcolor(c_letrero1);
  for(i=0;i<nopciones;i++)
     {
       moveto(posvent[i].x1+((posvent[i].x2-posvent[i].x1) >> 1 ),
            posvent[i].y1+((posvent[i].y2-posvent[i].y1) >> 1 ));
       outtext(posvent[i].msg);
     }
  Mshow(TRUE);
  i=0;
  do
    {
      if(entra_ventana(posvent[i].x1,posvent[i].y1,posvent[i].x2,posvent[i].y2))
      {
        Mshow(FALSE);
        setfillstyle(SOLID_FILL,c_barra2);
        bar(posvent[i].x1,posvent[i].y1,posvent[i].x2,posvent[i].y2);
        setcolor(marco);
        rectangle(posvent[i].x1,posvent[i].y1,posvent[i].x2,posvent[i].y2);
        setcolor(c_letrero2);
        moveto(posvent[i].x1+((posvent[i].x2-posvent[i].x1) >> 1 ),
             posvent[i].y1+((posvent[i].y2-posvent[i].y1) >> 1 ));
        outtext(posvent[i].msg);
        Mshow(TRUE);

boton=sal_ventana(posvent[i].x1,posvent[i].y1,posvent[i].x2,posvent[i].y2);
        if(boton)boton=i+1;
        Mshow(FALSE);
```

```
        setfillstyle(SOLID_FILL,c_barra1);
        bar(posvent[i].x1,posvent[i].y1,posvent[i].x2,posvent[i].y2);
        setcolor(c_letrero1);
        moveto(posvent[i].x1+((posvent[i].x2-posvent[i].x1) >> 1 ),
             posvent[i].y1+((posvent[i].y2-posvent[i].y1) >> 1 ));
        outtext(posvent[i].msg);
        Mshow(TRUE);
      };
      i++;
      if(i==nopciones)i=0;
    } while(!boton);
  return(boton);
}


/********* FUNCION QUE HACE QUE EL MOUSE DETECTE LA VENTANA *********/

int  entra_ventana(int x1,int y1,int x2, int y2)
{
  status=Mpos();
  if(!((status.xaxis>x1)&&(status.xaxis<x2)&&
       (status.yaxis>y1)&&(status.yaxis<y2))) return(0);
    else
    return(1);
}


/** FUNCION QUE HACE QUE EL MOUSE DETECTE QUE SALIO DE LA VENTANA ***/

int sal_ventana(int x1,int y1,int x2,int y2)
{
  do
    {
      status=Mpos();
      if(status.button_status) return(status.button_status);
    }
  while((status.xaxis>x1)&&(status.xaxis<x2)&&
      (status.yaxis>y1)&&(status.yaxis<y2));
       return(0);
}



/***************** FUNCION QUE SALVA UNA VENTANA ********************/

int far *salva_vent(int x1,int y1,int x2,int y2)
{
  int far *iptr;
  tamano=imagesize(x1,y1,x2,y2);
  iptr=farmalloc(tamano);
  if(iptr==NULL)
    {
      setfillstyle(SOLID_FILL,EGA_RED);
      bar(90,370,550,435);
      setcolor(EGA_YELLOW);
      rectangle(90,370,550,435);
      settextjustify(1,1);
```

```
      settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
      outtextxy(330,400,"Error, no hay memoria, reinicializar equipo");
    }
  getimage(x1,y1,x2,y2,iptr);
  return(iptr);
}


/***************** FUNCION QUE RESTAURA UNA VENTANA ******************/

void restaura_vent(int x,int y,int far *iptr)
{
  putimage(x,y,iptr,COPY_PUT);
  farfree(iptr);
}


/********* FUNCION QUE DETECTA QUE EL BOTON SE HA SOLTADO ***********/

void suelta_boton(void)
{
  do
    {
      status=Mpos();
    }
  while(status.button_status!=0);
}


/************************ FUNCION DEL RATON *************************/

void Mshow(int showstat)
{
  if(showstat)
    {
      inreg.x.ax=1;
      if(!Mview)call_mouse;
      Mview=ON;
    }
    else
    {
      inreg.x.ax=2;
      if(Mview)call_mouse;
      Mview=OFF;
    }
}


/************* FUNCION QUE ESTABLECE LA POSICION DEL RATON ***********/

void Npos(Mstatus posicion)
{
  inreg.x.ax=4;
  inreg.x.cx=posicion.xaxis;
  inreg.x.dx=posicion.yaxis;
  call_mouse;
}
```

```
/***************** FUNCION ACTIVA EL ESTADO DEL RATON ****************/

void Mini(void)
{
  inreg.x.ax=0;
  call_mouse;
}


/*************** FUNCION QUE LEE EL ESTADO DEL RATON ***************/

Mstatus Mpos()
{
  static Mstatus m;
  inreg.x.ax=3;
  call_mouse;
  m.button_status=outreg.x.bx;
  m.xaxis=outreg.x.cx;
  m.yaxis=outreg.x.dx;
  return(m);
}


/*************************************************************************/
/******** FUNCIONES PARA LA OPERACION DE LA MAQUINA DE CORTE ********/
/*************************************************************************/

/******** FUNCION QUE DIBUJA LOS COLORES DE FONDO DEL PROGRAMA ******/

void pantalla(void)
{
/****** Construccion del cuadro principal *******/
  setfillstyle(SOLID_FILL,EGA_BLACK);
  bar(0,0,640,480);
  setfillstyle(SOLID_FILL,EGA_RED);
  bar(5,56,635,475);
  setcolor(EGA_WHITE);
  rectangle(5,5,635,475);
  line(5,55,635,55);

/****** Construccion del titulo *****************/
  setfillstyle(SOLID_FILL,EGA_BLUE);
  bar(6,6,634,54);
  setcolor(EGA_YELLOW);
  settextjustify(1,1);
  settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
  outtextxy(320,30,"PANEL DE CONTROL DE LA MAQUINA DE CORTE");

/******* Construccion de la zona de trabajo *****/
/* Area=400x400,(x1,y1)=(120,65),(x2,y2)=(520,465)*/
  setfillstyle(SOLID_FILL,EGA_BLACK);
  bar(119,64,521,466);
  setcolor(EGA_WHITE);
  rectangle(119,64,521,466);
```

```
/****** Construccion del Menu de la derecha *****/
  setfillstyle(SOLID_FILL,EGA_BLUE);
  bar(530,65,626,465);
  setcolor(EGA_GREEN);
  rectangle(530,65,626,465);
  estadisticas();
  setcolor(EGA_YELLOW);
  rectangle(530,439,626,465);

/***** Construccion del Menu de la izquierda ****/
  setfillstyle(SOLID_FILL,EGA_BLUE);
  bar(14,64,111,466);
  setcolor(EGA_GREEN);
  rectangle(14,64,111,466);
}


/******* FUNCION QUE ESCRIBE VALORES INICIALES EN PANTALLA **********/

void estadisticas(void)
{
  setcolor(EGA_WHITE);
  settextjustify(1,0);
  settextstyle(SMALL_FONT,HORIZ_DIR,4);
  outtextxy(575,100,"Valores");
  settextjustify(2,0);
  outtextxy(595,130,"X0=      0");
  outtextxy(595,145,"Y0=      0");
  outtextxy(595,160,"□=      0");
  outtextxy(595,175,"v=      0");
  outtextxy(595,190,"l=      0");
  outtextxy(595,205,"X1=      0");
  outtextxy(595,220,"Y1=      0");
  settextjustify(0,0);
  outtextxy(605,130,"æ");
  outtextxy(605,145,"æ");
  outtextxy(605,160,"ø");
  outtextxy(605,175,"æ/s");
  outtextxy(605,190,"æ");
  outtextxy(605,205,"æ");
  outtextxy(605,220,"æ");
  settextjustify(1,1);
  outtextxy(575,280,"Avance X");
  outtextxy(575,300,"0 % ...");
  outtextxy(575,320,"Avance Y");
  outtextxy(575,340,"0 % ...");
}


/************** FUNCION QUE DESACTIVA FLECHA DEL MOUSE **************/
/***************     HACE MARCO Y FONDO DE SUBMENUS       ***************/

void cuadro_in(int c_marco,int c_fondo)
{
  status=Mpos();
  Mshow(FALSE);
  setfillstyle(SOLID_FILL,c_fondo);
```

```
  bar(140,150,500,300);
  setcolor(c_marco);
  rectangle(140,150,500,300);
}

/************* FUNCION QUE ACTIVA LA FLECHA DEL MOUSE ***************/

void cuadro_out(void)
{
  Npos(status);
  Mshow(TRUE);
}

/*************** FUNCION QUE LEE ARCHIVO DE INFORMACION *************/

void lect_info(void)
{
  FILE *fp;
  char msg[] = "a:\\info.txt";
  char buf[61];
  int i=0;
  int con=0;
  if((fp = fopen(msg, "r"))== NULL)
    {
      settextjustify(1,1);
      settextstyle(SMALL_FONT,HORIZ_DIR,4);
      outtextxy(320,200,"El archivo info.txt no se encuentra");
      outtextxy(320,280,"Presiona cualquier tecla para continuar");
    }
    else
    {
      /* seek to the beginning of the file */
      fseek(fp,0, SEEK_SET);
      while(!feof(fp))
         {
           /* Lee el archivo y lo despliega */
           memset(buf,'\0',61);
           fread(buf, 60, 1, fp);
           con++;
           settextjustify(0,1);
           settextstyle(SMALL_FONT,HORIZ_DIR,4);
           outtextxy(170,150+INC*con,buf);
           outtextxy(220,280,"Presiona <enter> para continuar");
           if(con==6)
             {
             if(ESC==getch()) goto a;
             setfillstyle(SOLID_FILL,EGA_BROWN);
             bar(141,151,499,270);
             for(i=1;i<=6;i++) con=0;
             }
         }
    }
  getch();
a:fclose(fp);
}
```

```
/*************** FUNCION QUE LEE EL ARCHIVO DE AYUDA ****************/

void lect_ayuda(void)
{
  FILE *fp;
  char msg[] = "a:\\ayuda.txt";
  char buf[61];
  int i;
  int con=0;
  if((fp = fopen(msg, "r"))== NULL)
    {
      settextjustify(1,1);
      settextstyle(SMALL_FONT,HORIZ_DIR,4);
      outtextxy(320,200,"El archivo ayuda.txt no se encuentra");
      outtextxy(320,280,"Presiona cualquier tecla para continuar");
    }
    else
    {
      /* seek to the beginning of the file */
      fseek(fp,0, SEEK_SET);
      while(!feof(fp))
         {
           /* Lee el archivo y lo despliega */
           memset(buf,'\0',61);
           fread(buf, 60, 1, fp);
           con++;
           settextjustify(0,1);
           settextstyle(SMALL_FONT,HORIZ_DIR,4);
           outtextxy(155,150+INC*con,buf);
           settextstyle(SMALL_FONT,HORIZ_DIR,4);
           outtextxy(220,280,"Presiona <enter> para continuar");
           if(con==6)
             {
             if(ESC==getch()) goto b;
             setfillstyle(SOLID_FILL,EGA_BROWN);
             bar(141,151,499,270);
             for(i=1;i<=6;i++) con=0;
             }
         }
    }
  getch();
b:fclose(fp);
}


/********** FUNCION PARA OBTENER DATOS DE CORTE EN PANTALLA *********/

void obtener_datos(void)
{
  char ch;
  char valor1[10];
  char valor2[10];
  char valor3[10];
  char valor4[10];
  char valor5[10];
```

```
settextjustify(1,1);
settextstyle(SMALL_FONT,HORIZ_DIR,4);
outtextxy(320,165,"Datos de corte");
do
  {
    setfillstyle(SOLID_FILL,EGA_RED);
    bar(145,180,495,295);
    moveto(180,180); // Obtiene punto de origen X0
    posx=lee_valor("X0 (micras) = ");
    itoa(posx,valor1,10);
    setfillstyle(SOLID_FILL,EGA_BLUE);
    bar(560,120,600,135);
    settextjustify(2,0);
    settextstyle(SMALL_FONT,HORIZ_DIR,4);
    outtextxy(595,130,valor1);

    moveto(180,200); // Obtiene punto de origen Y0
    posy=lee_valor("Y0 (micras) = ");
    itoa(posy,valor2,10);
    setfillstyle(SOLID_FILL,EGA_BLUE);
    bar(560,135,600,150);
    settextjustify(2,0);
    settextstyle(SMALL_FONT,HORIZ_DIR,4);
    outtextxy(595,145,valor2);

    moveto(186,220); // Obtiene angulo de origen □
    pen=lee_valor("m (grados) = ");
    itoa(pen,valor3,10);
    setfillstyle(SOLID_FILL,EGA_BLUE);
    bar(560,150,600,165);
    settextjustify(2,0);
    settextstyle(SMALL_FONT,HORIZ_DIR,4);
    outtextxy(595,160,valor3);

    moveto(156,240); // Obtiene velocidad de corte v
    vel=lee_valor("v (micras/seg) = ");
    itoa(vel,valor4,10);
    setfillstyle(SOLID_FILL,EGA_BLUE);
    bar(560,165,600,180);
    settextjustify(2,0);
    settextstyle(SMALL_FONT,HORIZ_DIR,4);
    outtextxy(595,175,valor4);

    moveto(190,260); // Obtiene longitud de corte l
    dis=lee_valor("l (micras) = ");
    itoa(dis,valor5,10);
    setfillstyle(SOLID_FILL,EGA_BLUE);
    bar(560,180,600,195);
    settextjustify(2,0);
    settextstyle(SMALL_FONT,HORIZ_DIR,4);
    outtextxy(595,190,valor5);

    outtextxy(480,230,"Deseas corregir?(No)");
    ch=getch();
  }
while((ch=='s')||(ch=='S'));
```

```
  vel_p=vel+(0.20)*vel;
  vel_n=vel-(0.20)*vel;
}


/*************** FUNCION QUE LEE LOS VALORES DE CORTE ***************/

int lee_valor(char *texto)
{
  char ch;
  char numero[10];
  int i=0, j=0;
  int xi;
  int bakgnd;
  int xp, yp;
  int xcur, xcura;
  settextstyle(SMALL_FONT,HORIZ_DIR,5);
  settextjustify(LEFT_TEXT,TOP_TEXT);
  outtext(texto);
  xp=getx();
  yp=gety();
  xi=xp;
  bakgnd=getpixel(xp,yp);
  for(i=0;i<=4;i++)
     {
       numero[i]=' ';
     }
  numero[4]='\0';
  i=0;
  xcur=getx();
  xcura=xcur;
  while(!kbhit())
       {
       setfillstyle(SOLID_FILL,EGA_WHITE);
       bar(xcur,gety()+13,xcur+10,gety()+14);
       delay(100);
       setfillstyle(SOLID_FILL,bakgnd);
       bar(xcur,gety()+13,xcur+10,gety()+14);
       delay(100);
       }
  while(((ch=getch())!='\r'))
       {
       if((!(ch>=48 && ch<=57 )) && (ch!=8) && (ch!=45)) /*Solo aceptar numeros o
regreso <-*/
         {
           while(!kbhit())
              {
                setfillstyle(SOLID_FILL,EGA_WHITE);
                bar(xcur,gety()+13,xcur+10,gety()+14);
                delay(100);
                setfillstyle(SOLID_FILL,bakgnd);
                bar(xcur,gety()+13,xcur+10,gety()+14);
                delay(100);
              }
         }
         else
         {
```

```
            numero[i++]=ch;
            outtextxy(xi,gety(),numero);
            xcur+=8;
            while((!kbhit())&&(ch!=8))
               {
                 setfillstyle(SOLID_FILL,EGA_WHITE);
                 bar(xcur,gety()+13,xcur+10,gety()+14);
                 delay(100);
                 setfillstyle(SOLID_FILL,bakgnd);
                 bar(xcur,gety()+13,xcur+10,gety()+14);
                 delay(100);
               }
            if(ch==8) //backspace
              {
              xcur=xcura;
              setcolor(bakgnd);
              i=0;
              outtextxy(xi,gety(),numero);
              setcolor(EGA_WHITE);
              for(j=0;j<=4;j++)
                 {
                   numero[j]=' ';
                 }
              numero[4]='\0';
              }
            while(!kbhit())
               {
                 setfillstyle(SOLID_FILL,EGA_WHITE);
                 bar(xcur,gety()+13,xcur+10,gety()+14);
                 delay(100);
                 setfillstyle(SOLID_FILL,bakgnd);
                 bar(xcur,gety()+13,xcur+10,gety()+14);
                 delay(100);
               }
            if(i>=5) break;
          }
        }
  numero[i]='\n';
  return(atoi(numero));
}

/*** FUNCION QUE CALCULA LOS PUNTOS FINALES DE LA LINEA DE CORTE ****/

void X1_Y1(int x_ini,int y_ini,int pd,int lon)
{
  int x=0 ,y=0;
  int inx=0, iny=0;
  float pend_x=0, pend_y=0;
  char valor1[10];
  char valor2[10];

  pend_x=cos(pd*3.1416/180);
  pend_y=sin(pd*3.1416/180);
  x=lon*pend_x;
  y=lon*pend_y;
  inx=x_ini+x;
```

```
  iny=y_ini+y;

  itoa(inx,valor1,10);
  setfillstyle(SOLID_FILL,EGA_BLUE);
  bar(560,195,600,210);
  settextjustify(2,0);
  settextstyle(SMALL_FONT,HORIZ_DIR,4);
  outtextxy(595,205,valor1);

  itoa(iny,valor2,10);
  setfillstyle(SOLID_FILL,EGA_BLUE);
  bar(560,210,600,225);
  settextjustify(2,0);
  settextstyle(SMALL_FONT,HORIZ_DIR,4);
  outtextxy(595,220,valor2);
}


/*************** FUNCION QUE DIBUJA LA LINEA DE CORTE ***************/

void dibujo(int x_ini,int y_ini,int pd,int vl,long lon)
{
  float pendiente=0;
  int inx=0, iny=0;
  long x=0, y=0;
  long a=0, b=0, c=0;

  if((pd>=-360 && pd<=360)&&(x_ini>=0 && x_ini<=400)&&(y_ini>=0 &&
y_ini<=400)&&(lon>=0 && lon<=565))
    {
      pendiente=tan(-pd*3.1416/180);
      inx=x_ini+120;
      iny=-y_ini+465;
      c=lon*lon;

      /* Funcion uno*/
      if((pd>=-45 && pd<=45)||(pd>=315 && pd<=360)||(pd<=-315 && pd>=-360))
      {
        for(x=0;x<1000;x++)
          {
            y=pendiente*x;
            a=x*x;
            b=y*y;
            if((a+b)>c) break;
            if((iny+y)<65 || (iny+y)>465 || (inx+x)<120 || (inx+x)>520 )
            {
              valores();
              break;
            }
            putpixel(inx+x,iny+y,EGA_YELLOW);
          }
      }

      /*Funcion dos*/
      if((pd>225 && pd<315)||(pd<-45 && pd>-135))
      {
        for(y=0;y<1000;y++)
```

```
        {
          x=y/pendiente;
          a=x*x;
          b=y*y;
          if((a+b)>c) break;
          if((iny+y)<65 || (iny+y)>465 || (inx+x)<120 || (inx+x)>520 )
          {
            valores();
            break;
          }
          putpixel(inx+x,iny+y,EGA_YELLOW);
        }
    }

    /* Funcion tres)*/
    if((pd>=135 && pd<=225)||(pd<=-135 && pd>=-225))
    {
      for(x=0;x>-1000;x--)
          {
            y=pendiente*x;
            a=x*x;
            b=y*y;
            if((a+b)>c) break;
            if((iny+y)<65 || (iny+y)>465 || (inx+x)<120 || (inx+x)>520 )
            {
              valores();
              break;
            }
            putpixel(inx+x,iny+y,EGA_YELLOW);
          }
    }

    /*Funcion cuatro*/
    if((pd>45 && pd<135)||(pd<-225 && pd>-315))
    {
      for(y=0;y>-1000;y--)
          {
            x=y/pendiente;
            a=x*x;
            b=y*y;
            if((a+b)>c) break;
            if((iny+y)<65 || (iny+y)>465 || (inx+x)<120 || (inx+x)>520 )
            {
              valores();
              break;
            }
            putpixel(inx+x,iny+y,EGA_YELLOW);
          }
    }
  }
  else
  {
    iptr10=salva_vent(140,150,500,300);
    cuadro_in(EGA_WHITE,EGA_RED);
    settextjustify(1,1);
    settextstyle(SMALL_FONT,HORIZ_DIR,4);
    outtextxy(320,160,"ERROR");
```

```
      if((x_ini<0 || x_ini>400)) outtextxy(320,180,"EL valor de X0 esta fuera de
rango");
      if((y_ini<0 || y_ini>400)) outtextxy(320,200,"EL valor de Y0 esta fuera de
rango");
      if((pd<-360 || pd>360)) outtextxy(320,220,"EL valor de □ esta fuera de
rango");
      if((vl<0 || vl>565)) outtextxy(320,240,"EL valor de v esta fuera de rango");
      if((lon<0 || lon>565)) outtextxy(320,260,"EL valor de l esta fuera de
rango");
      outtextxy(320,280,"[Introduce nuevamente los valores]");
      getch();
      restaura_vent(140,150,iptr10);
      cuadro_out();
    }
}

/* FUNCION QUE MUESTRA MENSAJE CUANDO L SOBREPASA LOS LIMITES DE DIBUJO */

void valores(void)
{
  iptr11=salva_vent(140,150,500,300);
  cuadro_in(EGA_WHITE,EGA_RED);
  settextjustify(1,1);
  settextstyle(SMALL_FONT,HORIZ_DIR,4);
  outtextxy(320,170,"ERROR");
  outtextxy(320,200,"El valor de l es muy grande y la");
  outtextxy(320,220,"figura sale cortada");
  outtextxy(320,260,"Introduce nuevamente los valores");
  getch();
  restaura_vent(140,150,iptr11);
  setfillstyle(SOLID_FILL,EGA_BLUE);
  bar(532,67,624,438);
  estadisticas();
  setfillstyle(SOLID_FILL,EGA_BLACK);
  bar(119,64,521,466);
  setcolor(EGA_WHITE);
  rectangle(119,64,521,466);
  cuadro_out();
}

/************* FUNCION PARA PROBAR LA MAQUINA DE CORTE **************/

void prueba(void)
{
/*  char ch;
  settextjustify(1,1);
  settextstyle(SMALL_FONT,HORIZ_DIR,5);
  outtextxy(320,200,"Quieres realizar prueba del equipo?");
  outtextxy(320,260,"[s]i      [n]o");
  ch=getch();
  if((ch=='s')||(ch=='S'))
    {
      delay(500);
    }*/
  char ch;
```

```
  iptr5=salva_vent(140,150,500,300);
  cuadro_in(EGA_WHITE,EGA_RED);
  settextjustify(1,1);
  settextstyle(SMALL_FONT,HORIZ_DIR,5);
  outtextxy(320,200,"Deseas realizar prueba del equipo?");
  outtextxy(320,260,"[s]i       [n]o");
  ch=getch();
  restaura_vent(140,150,iptr5);
  cuadro_out();
  if((ch=='s')||(ch=='S'))
    {
      posx=0, posy=0, pen=0, dis=400, vel=100;
      conv_d_h();
      if (pc_mc()) mc_pc();
    }
}


/* FUNCION QUE CONVIERTE DECIMAL-HEXADECIMAL Y MANDA DATOS AL PUERTO */

void dec_hex(int dato)
{
  int dato2;
  int dato3;
  int var[580]={ 0x0,0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8,0x9,0xA,0xB,0xC,0xD,0xE,
            0x0F,0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1A,
            0x1B,0x1C,0x1D,0x1E,0x1F,0x20,0x21,0x22,0x23,0x24,0x25,0x26,
            0x27,0x28,0x29,0x2A,0x2B,0x2C,0x2D,0x2E,0x2F,0x30,0x31,0x32,
            0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3A,0x3B,0x3C,0x3D,0x3E,
            0x3F,0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4A,
            0x4B,0x4C,0x4D,0x4E,0x4F,0x50,0x51,0x52,0x53,0x54,0x55,0x56,
            0x57,0x58,0x59,0x5A,0x5B,0x5C,0x5D,0x5E,0x5F,0x60,0x61,0x62,
            0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x6A,0x6B,0x6C,0x6D,0x6E,
            0x6F,0x70,0x71,0x72,0x73,0x74,0x75,0x76,0x77,0x78,0x79,0x7A,
            0x7B,0x7C,0x7D,0x7E,0x7F,0x80,0x81,0x82,0x83,0x84,0x85,0x86,
            0x87,0x88,0x89,0x8A,0x8B,0x8C,0x8D,0x8E,0x8F,0x90,0x91,0x92,
            0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9A,0x9B,0x9C,0x9D,0x9E,
            0x9F,0xA0,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0xAA,
            0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,
            0xB7,0xB8,0xB9,0xBA,0xBB,0xBC,0xBD,0xBE,0xBF,0xC0,0xC1,0xC2,
            0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0xCA,0xCB,0xCC,0xCD,0xCE,
            0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0xDA,
            0xDB,0xDC,0xDD,0xDE,0xDF,0xE0,0xE1,0xE2,0xE3,0xE4,0xE5,0xE6,
            0xE7,0xE8,0xE9,0xEA,0xEB,0xEC,0xED,0xEE,0xEF,0xF0,0xF1,0xF2,
            0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0xFA,0xFB,0xFC,0xFD,0xFE,
            0x0FF,0x100,0x101,0x102,0x103,0x104,0x105,0x106,0x107,0x108,
            0x109,0x10A,0x10B,0x10C,0x10D,0x10E,0x10F,0x110,0x111,0x112,
            0x113,0x114,0x115,0x116,0x117,0x118,0x119,0x11A,0x11B,0x11C,
            0x11D,0x11E,0x11F,0x120,0x121,0x122,0x123,0x124,0x125,0x126,
            0x127,0x128,0x129,0x12A,0x12B,0x12C,0x12D,0x12E,0x12F,0x130,
            0x131,0x132,0x133,0x134,0x135,0x136,0x137,0x138,0x139,0x13A,
            0x13B,0x13C,0x13D,0x13E,0x13F,0x140,0x141,0x142,0x143,0x144,
            0x145,0x146,0x147,0x148,0x149,0x14A,0x14B,0x14C,0x14D,0x14E,
            0x14F,0x150,0x151,0x152,0x153,0x154,0x155,0x156,0x157,0x158,
            0x159,0x15A,0x15B,0x15C,0x15D,0x15E,0x15F,0x160,0x161,0x162,
            0x163,0x164,0x165,0x166,0x167,0x168,0x169,0x16A,0x16B,0x16C,
            0x16D,0x16E,0x16F,0x170,0x171,0x172,0x173,0x174,0x175,0x176,
```

```
            0x177,0x178,0x179,0x17A,0x17B,0x17C,0x17D,0x17E,0x17F,0x180,
            0x181,0x182,0x183,0x184,0x185,0x186,0x187,0x188,0x189,0x18A,
            0x18B,0x18C,0x18D,0x18E,0x18F,0x190,0x191,0x192,0x193,0x194,
            0x195,0x196,0x197,0x198,0x199,0x19A,0x19B,0x19C,0x19D,0x19E,
            0x19F,0x1A0,0x1A1,0x1A2,0x1A3,0x1A4,0x1A5,0x1A6,0x1A7,0x1A8,
            0x1A9,0x1AA,0x1AB,0x1AC,0x1AD,0x1AE,0x1AF,0x1B0,0x1B1,0x1B2,
            0x1B3,0x1B4,0x1B5,0x1B6,0x1B7,0x1B8,0x1B9,0x1BA,0x1BB,0x1BC,
            0x1BD,0x1BE,0x1BF,0x1C0,0x1C1,0x1C2,0x1C3,0x1C4,0x1C5,0x1C6,
            0x1C7,0x1C8,0x1C9,0x1CA,0x1CB,0x1CC,0x1CD,0x1CE,0x1CF,0x1D0,
            0x1D1,0x1D2,0x1D3,0x1D4,0x1D5,0x1D6,0x1D7,0x1D8,0x1D9,0x1DA,
            0x1DB,0x1DC,0x1DD,0x1DE,0x1DF,0x1E0,0x1E1,0x1E2,0x1E3,0x1E4,
            0x1E5,0x1E6,0x1E7,0x1E8,0x1E9,0x1EA,0x1EB,0x1EC,0x1ED,0x1EE,
            0x1EF,0x1F0,0x1F1,0x1F2,0x1F3,0x1F4,0x1F5,0x1F6,0x1F7,0x1F8,
            0x1F9,0x1FA,0x1FB,0x1FC,0x1FD,0x1FE,0x1FF,0x200,0x201,0x202,
            0x203,0x204,0x205,0x206,0x207,0x208,0x209,0x20A,0x20B,0x20C,
            0x20D,0x20E,0x20F,0x210,0x211,0x212,0x213,0x214,0x215,0x216,
            0x217,0x218,0x219,0x21A,0x21B,0x21C,0x21D,0x21E,0x21F,0x220,
            0x221,0x222,0x223,0x224,0x225,0x226,0x227,0x228,0x229,0x22A,
            0x22B,0x22C,0x22D,0x22E,0x22F,0x230,0x231,0x232,0x233,0x234,
            0x235,0x236,0x237,0x238,0x239,0x23A,0x23B,0x23C,0x23D,0x23E,
            0x23F};

  valor_dato=var[dato];
  if(dato<=255)
    {
      dato2=0;
      dato3=0;
      valor_dato2=var[dato2];
      valor_dato3=var[dato3];
    }
  if(dato>255 && dato<=511)
    {
      dato2=1;
      dato3=0;
      valor_dato2=var[dato2];
      valor_dato3=var[dato3];
    }
  if(dato>511 && dato<=566)
    {
      dato2=2;
      dato3=0;
      valor_dato2=var[dato2];
      valor_dato3=var[dato3];
    }
}

/****************** FUNCION QUE CORTA EL MATERIAL ******************/

void cortar(void)
{
  char ch;
  iptr5=salva_vent(140,150,500,300);
  cuadro_in(EGA_WHITE,EGA_RED);
  settextjustify(1,1);
  settextstyle(SMALL_FONT,HORIZ_DIR,5);
  outtextxy(320,200,"Deseas comenzar a cortar el material?");
```

```
  outtextxy(320,260,"[s]i       [n]o");
  ch=getch();
  restaura_vent(140,150,iptr5);
  cuadro_out();
  if((ch=='s')||(ch=='S'))
    {
      conv_d_h();
      if (pc_mc()) mc_pc();
    }
}


/************ FUNCION QUE MUESTRA MENSAJE ERROR COMUNICACION ********/

void mens_error(int error)
{
      iptr12=salva_vent(140,150,500,300);
      cuadro_in(EGA_WHITE,EGA_RED);
      settextjustify(1,1);
      settextstyle(SMALL_FONT,HORIZ_DIR,4);
      outtextxy(320,190,"ERROR");
      if(error==0) outtextxy(320,220,"Error en la transferencia de control PC-
MC");
      if(error==1) outtextxy(320,220,"Error en la transferencia de datos PC-MC");
      if(error==2) outtextxy(320,220,"Error en la transferencia de control MC-
PC");
      if(error==3) outtextxy(320,220,"Error en la transferencia de datos MC-PC");
      if(error==4) outtextxy(320,220,"Error en la velocidad");
      if(error==5) outtextxy(320,220,"Se prendieron alarmas");
      if(error==6) outtextxy(320,220,"Se termino de realizar el corte");
      outtextxy(320,280,"<Presiona cualquier tecla para continuar>");
      getch();
      restaura_vent(140,150,iptr12);
      cuadro_out();
}


/************** FUNCION QUE LE EL PORCENTAJE DE CORTE ***************/

void lect_porc(int porc)
{
  char val_porc[10];
  itoa(porc,val_porc,10);
  setfillstyle(SOLID_FILL,EGA_BROWN);
  bar(540,235,568,245);
  settextjustify(1,1);
  settextstyle(SMALL_FONT,HORIZ_DIR,4);
  outtextxy(555,240,val_porc);
}


/**************** FUNCION PARA SALIR DEL PROGRAMA ******************/

void salir(void)
{
  char ch;
  settextjustify(1,1);
```

```
  settextstyle(SMALL_FONT,HORIZ_DIR,5);
  outtextxy(320,200,"Deseas salir?");
  outtextxy(320,260,"[s]i      [n]o");
  ch=getch();
  if((ch=='s')||(ch=='S'))
    {
      delay(700);
      closegraph();
      exit(0);
    }
}

/*************** FUNCION PARA REINICIALIZAR EL PROGRAMA *************/

void reset(void)
{
  char ch;
  settextjustify(1,1);
  settextstyle(SMALL_FONT,HORIZ_DIR,5);
  outtextxy(320,200,"Deseas reinicializar el programa?");
  outtextxy(320,260,"[s]i      [n]o");
  ch=getch();
  if((ch=='s')||(ch=='S'))
    {
      delay(500);
      farfree(iptr7);
      menu_principal();
    }
}

/******** FUNCIONES QUE CONVIERTEN DE HEXADECIMAL A DECIMAL *********/

int hex_dec(int dato1,int dato2,int dato3)
{
  int dato_total;
  dato3=0;
  if(dato2 == 0)
    {
      dato_total=dato1;
    }

  if(dato2 == 1)
    {
      dato_total=dato1+256;
    }
  if(dato2 == 10)
    {
      dato_total=dato1+512;
    }
  return(dato_total);
}

/***************** COMUNICACION PC-MICROCONTROLADOR *****************/
```

```
int pc_mc()
{
  outportb(P_CONT_MC,0x00);
  outportb(P_CONT_MC,CONT_OK_E1); //Escribe ok en el bus de control del M.
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L1)  break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L1))
       {
        error=0;
        goto FIN;
       }
       val_c_ok='\0';
     }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);

/****************** Transferencia dato de velocidad + ***************/
/***************************** ler byte ******************************/
  outportb(P_CONT_MC,V_CONT_P_B1); //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L2) break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
       {
        error=0;
        goto FIN;
       }
       val_c_ok='\0';
     }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,v_dat_p_b1);   //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
       val_d_ok=inportb(P_CONT_PC);
       if(val_d_ok == DAT_OK_L) break;
       if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
       {
        error=1;
        goto FIN;
       }
       val_d_ok='\0';
     }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

/***************************** 2o byte ******************************/
  outportb(P_CONT_MC,V_CONT_P_B2);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L2) break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
       {
```

```
        error=0;
        goto FIN;
       }
       val_c_ok='\0';
     }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,v_dat_p_b2);     //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
       val_d_ok=inportb(P_CONT_PC);
       if(val_d_ok == DAT_OK_L) break;
       if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
       {
        error=1;
        goto FIN;
       }
       val_d_ok='\0';
     }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

/***************************** 3er byte ******************************/
  outportb(P_CONT_MC,V_CONT_P_B3);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L2) break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
       {
        error=0;
        goto FIN;
       }
       val_c_ok='\0';
     }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,v_dat_p_b3);     //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
       val_d_ok=inportb(P_CONT_PC);
       if(val_d_ok == DAT_OK_L) break;
       if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
       {
        error=1;
        goto FIN;
       }
       val_d_ok='\0';
     }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);
  getch();

/**************** Transferencia dato de velocidad real **************/
/***************************** ler byte ******************************/
  outportb(P_CONT_MC,V_CONT_R_B1);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
```

```
    {
      val_c_ok=inportb(P_CONT_PC);
      if(val_c_ok == CONT_OK_L2) break;
      if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
      {
       error=0;
       goto FIN;
      }
      val_c_ok='\0';
    }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,v_dat_r_b1);    //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
    {
      val_d_ok=inportb(P_CONT_PC);
      if(val_d_ok == DAT_OK_L) break;
      if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
      {
       error=1;
       goto FIN;
      }
      val_d_ok='\0';
    }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

/***************************** 2o byte ******************************/
  outportb(P_CONT_MC,V_CONT_R_B2);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
    {
      val_c_ok=inportb(P_CONT_PC);
      if(val_c_ok == CONT_OK_L2) break;
      if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
      {
       error=0;
       goto FIN;
      }
      val_c_ok='\0';
    }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,v_dat_r_b2);    //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
    {
      val_d_ok=inportb(P_CONT_PC);
      if(val_d_ok == DAT_OK_L) break;
      if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
      {
       error=1;
       goto FIN;
      }
      val_d_ok='\0';
    }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);
```

```
/*************************** 3er byte ******************************/
  outportb(P_CONT_MC,V_CONT_R_B3);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L2) break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
       {
        error=0;
        goto FIN;
       }
       val_c_ok='\0';
     }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,v_dat_r_b3);    //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
       val_d_ok=inportb(P_CONT_PC);
       if(val_d_ok == DAT_OK_L) break;
       if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
       {
        error=1;
        goto FIN;
       }
       val_d_ok='\0';
     }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

/****************** Transferencia dato de velocidad - ***************/
/*************************** 1er byte ******************************/
  outportb(P_CONT_MC,V_CONT_N_B1);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L2) break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
       {
        error=0;
        goto FIN;
       }
       val_c_ok='\0';
     }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,v_dat_n_b1);    //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
       val_d_ok=inportb(P_CONT_PC);
       if(val_d_ok == DAT_OK_L) break;
       if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
       {
        error=1;
        goto FIN;
       }
       val_d_ok='\0';
```

```
    }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

/***************************** 2o byte ******************************/
  outportb(P_CONT_MC,V_CONT_N_B2);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L2) break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
       {
        error=0;
        goto FIN;
       }
       val_c_ok='\0';
     }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,v_dat_n_b2);    //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
       val_d_ok=inportb(P_CONT_PC);
       if(val_d_ok == DAT_OK_L) break;
       if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
       {
        error=1;
        goto FIN;
       }
       val_d_ok='\0';
     }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

/***************************** 3er byte ******************************/
  outportb(P_CONT_MC,V_CONT_N_B3);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L2) break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
       {
        error=0;
        goto FIN;
       }
       val_c_ok='\0';
     }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,v_dat_n_b3);    //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
       val_d_ok=inportb(P_CONT_PC);
       if(val_d_ok == DAT_OK_L) break;
       if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
       {
        error=1;
```

```
        goto FIN;
       }
       val_d_ok='\0';
     }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

/***************** Transferencia dato de desplazamiento *************/
/***************************** ler byte ****************************/
  outportb(P_CONT_MC,D_CONT_B1);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L2) break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
       {
        error=0;
        goto FIN;
       }
       val_c_ok='\0';
     }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,d_dat_b1);    //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
       val_d_ok=inportb(P_CONT_PC);
       if(val_d_ok == DAT_OK_L) break;
       if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
       {
        error=1;
        goto FIN;
       }
       val_d_ok='\0';
     }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

/***************************** 2o byte *****************************/
  outportb(P_CONT_MC,D_CONT_B2);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L2) break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
       {
        error=0;
        goto FIN;
       }
       val_c_ok='\0';
     }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,d_dat_b2);    //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
       val_d_ok=inportb(P_CONT_PC);
```

```
        if(val_d_ok == DAT_OK_L) break;
        if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
        {
         error=1;
         goto FIN;
        }
        val_d_ok='\0';
      }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

/***************************** 3er byte ******************************/
  outportb(P_CONT_MC,D_CONT_B3);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
        val_c_ok=inportb(P_CONT_PC);
        if(val_c_ok == CONT_OK_L2) break;
        if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
        {
         error=0;
         goto FIN;
        }
        val_c_ok='\0';
      }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,d_dat_b3);     //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
        val_d_ok=inportb(P_CONT_PC);
        if(val_d_ok == DAT_OK_L) break;
        if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
        {
         error=1;
         goto FIN;
        }
        val_d_ok='\0';
      }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

/******************** Transferencia dato de origen *****************/
/***************************** 1er byte ******************************/
  outportb(P_CONT_MC,X0_CONT_B1);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
        val_c_ok=inportb(P_CONT_PC);
        if(val_c_ok == CONT_OK_L2) break;
        if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
        {
         error=0;
         goto FIN;
        }
        val_c_ok='\0';
      }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
```

```
  outportb(P_DAT_MC,X0_dat_b1);    //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
       val_d_ok=inportb(P_CONT_PC);
       if(val_d_ok == DAT_OK_L) break;
       if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
       {
        error=1;
        goto FIN;
       }
       val_d_ok='\0';
     }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

/***************************** 2o byte ******************************/
  outportb(P_CONT_MC,X0_CONT_B2);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L2) break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
       {
        error=0;
        goto FIN;
       }
       val_c_ok='\0';
     }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,X0_dat_b2);    //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
       val_d_ok=inportb(P_CONT_PC);
       if(val_d_ok == DAT_OK_L) break;
       if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
       {
        error=1;
        goto FIN;
       }
       val_d_ok='\0';
     }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

/***************************** 3er byte ******************************/
  outportb(P_CONT_MC,X0_CONT_B3);  //Manda al MC el byte de control
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L2) break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
       {
        error=0;
        goto FIN;
       }
       val_c_ok='\0';
```

```
    }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_DAT_MC,X0_dat_b3);     //Manda al MC el byte de informacion
  for(pet_d_ok=0;pet_d_ok<=100;pet_d_ok++)
     {
       val_d_ok=inportb(P_CONT_PC);
       if(val_d_ok == DAT_OK_L) break;
       if((pet_d_ok == 100)&&(val_d_ok != DAT_OK_L))
       {
        error=1;
        goto FIN;
       }
       val_d_ok='\0';
     }
  val_d_ok='\0';
  outportb(P_DAT_MC,0x00);

  error=7;
  if(error!=7)
    {
FIN: mens_error(error);
     return(0);
    }
    else
    {
      return(1);
    }
}


/***************** COMUNICACION MICROCONTROLADOR-PC *****************/

void mc_pc(void)
{
  int ciclo;
  int ter_corte;

  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
     {
       val_c_ok=inportb(P_CONT_PC);
       if(val_c_ok == CONT_OK_L2)  break;
       if((pet_c_ok == 100)&&(val_c_ok != CONT_OK_L2))
       {
        error=2;
        goto FIN;
       }
       val_c_ok='\0';
     }     //Si es ok,continua la operacion
  val_c_ok='\0';
  outportb(P_CONT_MC,CONT_OK_E2);

  for(ciclo=0;ciclo<1E1000000;ciclo++)
  {
/****************** Transferencia dato de posicion ******************/
/***************************** ler byte *****************************/
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
```

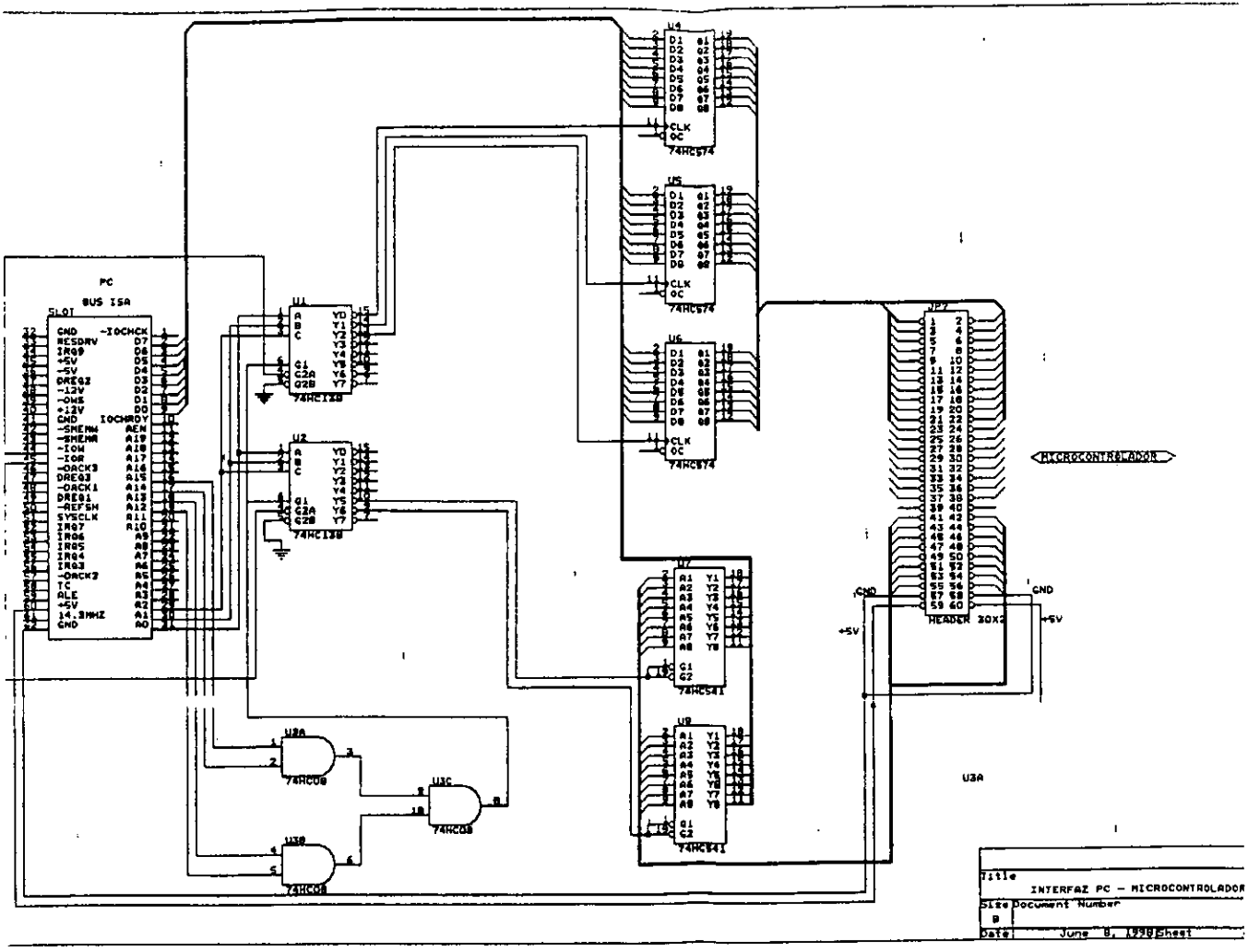
```
    {
      val_c_ok=inportb(P_CONT_PC);
      if(val_c_ok == P_CONT_B1)  break;
      if((pet_c_ok == 100)&&(val_c_ok != P_CONT_B1))
      {
       error=2;
       goto FIN;
      }
      val_c_ok='\0';
    }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_CONT_MC,CONT_OK_E3);
  p_dat_b1=inportb(P_DAT_PC);
  outportb(P_CONT_MC,0x00);
  outportb(P_CONT_MC,DAT_OK_E);

/***************************** 2o byte *******************************/
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
    {
      val_c_ok=inportb(P_CONT_PC);
      if(val_c_ok == P_CONT_B2) break;
      if((pet_c_ok == 100)&&(val_c_ok != P_CONT_B2))
      {
       error=2;
       goto FIN;
      }
      val_c_ok='\0';
    }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_CONT_MC,CONT_OK_E3);
  p_dat_b2=inportb(P_DAT_PC);
  outportb(P_CONT_MC,0x00);
  outportb(P_CONT_MC,DAT_OK_E);

/***************************** 3er byte *******************************/
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
    {
      val_c_ok=inportb(P_CONT_PC);
      if(val_c_ok == P_CONT_B3) break;
      if((pet_c_ok == 100)&&(val_c_ok != P_CONT_B3))
      {
       error=2;
       goto FIN;
      }
      val_c_ok='\0';
    }
  val_c_ok='\0';
  outportb(P_CONT_MC,0x00);
  outportb(P_CONT_MC,CONT_OK_E3);
  p_dat_b3=inportb(P_DAT_PC);
  outportb(P_CONT_MC,0x00);
  outportb(P_CONT_MC,DAT_OK_E);

/****************** Revisa condicion de velocidad ******************/
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
```

```
    {
      val_c_ok=inportb(P_CONT_PC);
      if(val_c_ok == VEL_OK)
      {
        outportb(P_CONT_MC,0x00);
        outportb(P_CONT_MC,CONT_OK_E3);
        break;
      }
      if(val_c_ok == VEL_ERR)
      {
        outportb(P_CONT_MC,0x00);
        outportb(P_CONT_MC,CONT_OK_E2);
        vel_err_b3=inportb(P_DAT_PC);
        outportb(P_CONT_MC,0x00);
        outportb(P_CONT_MC,DAT_OK_E);
        error=4;
        goto FIN;
      }
      if((pet_c_ok == 100)&&((val_c_ok != VEL_OK)||(val_c_ok != VEL_ERR)))
      {
       error=4;
       goto FIN;
      }
      val_c_ok='\0';
    }
  val_c_ok='\0';

/****************** Revisa condicion de alarmas ******************/
  for(pet_c_ok=0;pet_c_ok<=100;pet_c_ok++)
    {
      val_c_ok=inportb(P_CONT_PC);
      if(val_c_ok == ALARMA_OK)
      {
        outportb(P_CONT_MC,0x00);
        outportb(P_CONT_MC,CONT_OK_E3);
        break;
      }
      if(val_c_ok == ALARMA_ERR)
      {
        outportb(P_CONT_MC,0x00);
        outportb(P_CONT_MC,CONT_OK_E2);
        alarma_err_b=inportb(P_DAT_PC);
        outportb(P_CONT_MC,0x00);
        outportb(P_CONT_MC,DAT_OK_E);
        error=5;
        goto FIN;
      }
      if((pet_c_ok == 100)&&((val_c_ok != ALARMA_OK)||(val_c_ok != ALARMA_ERR)))
      {
       error=5;
       goto FIN;
      }
      val_c_ok='\0';
    }
  val_c_ok='\0';
```

```
/************** Comparacion para ver si acabo el corte **************/

  ter_corte=hex_dec(p_dat_b1,p_dat_b2,p_dat_b3);
  lect_porc(ter_corte);
  if(ter_corte == dis)
    {
      error=6;
      goto FIN;
    }
  setfillstyle(SOLID_FILL,EGA_BROWN);
  bar(540,235,568,245);
  }
FIN: mens_error(error);
}


/************** FUNCION PARA CAMBIAR UN DATO A 3 BYTES **************/

void conv_d_h(void)
{
  dec_hex(posx);
  X0_dat_b1=valor_dato;
  X0_dat_b2=valor_dato2;
  X0_dat_b3=valor_dato3;
  dec_hex(vel);
  v_dat_r_b1=valor_dato;
  v_dat_r_b2=valor_dato2;
  v_dat_r_b3=valor_dato3;
  dec_hex(vel_p);
  v_dat_p_b1=valor_dato;
  v_dat_p_b2=valor_dato2;
  v_dat_p_b3=valor_dato3;
  dec_hex(vel_n);
  v_dat_n_b1=valor_dato;
  v_dat_n_b2=valor_dato2;
  v_dat_n_b3=valor_dato3;
  dec_hex(dis);
  d_dat_b1=valor_dato;
  d_dat_b2=valor_dato2;
  d_dat_b3=valor_dato3;
}

/*************************** FIN DEL PROGRAMA ************************/
```

Apéndice E

Diagrama de la interfaz PC-MCU


PC
BUS ISA
SLOT
74HC574
74HC138
74HC541
74HC08
MICROCONTROLADOR
HEADER 30X2
+5V
GND
U3A
Title
INTERFAZ PC - MICROCONTROLADOR
Size
Document Number
B
Date: June 8, 1998
Sheet

Bibliografía

BIBLIOGRAFÍA

- Kernighan, Brian. El lenguaje de programación C. Prentice-Hall.
- C++ Library Reference.
- Webb, John. Industrial Control Electronics.
- Rashid, Muhanman H. Power electronics: Circuits, devices and aplication.
- Malone, Timothy J. Electronica Industrial: Dispositivos y sistemas.
- Harrington, John. Automate Process Control Electronics.
- Malvino, Albert. Principios y aplicaciones digitales.
- Givs, David. An Introduction to CNC Machinning and Programming.
- HC11 Reference Manual.
- Referencias en Internet:
  - Http://www.cs.curtin.edu.au/~cdillon/rail-pwm.html
  - http://manufacture.com.tw/~corisma
  - http://www.nulogic.com