43
2ej.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO Y CONSTRUCCION DE UN SISTEMA ELECTRONICO PARA LA OPERACION Y CONTROL DE MAQUINAS HERRAMIENTAS DE CONTROL NUMERICO

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A N :

JOSE GUILLERMO CURIEL TOLIVIA

MIGUEL ANGEL MARTINEZ MORALES

ASESOR DE TESIS: ING YUKIHIRO MINAMI KOYAMA (FI UNAM)

CIUDAD UNIVERSITARIA, 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

C O N T E N I D O

TEMA I
INTRODUCCION Y OBJETIVOS

A lo largo de la historia el hombre siempre ha buscado la forma de realizar cualquier trabajo aplicando el menor esfuerzo. Con esta idea surgieron las máquinas simples que han ayudado al hombre en todo tipo de labores, desde las más sencillas hasta las más complicadas. Sin embargo, no conforme con facilitarse el trabajo, el hombre también ha buscado la forma de hacerlo tan rápido como le sea posible. Por esto, durante la revolución industrial (siglo XVIII), aparecieron las máquinas tejedoras controladas mediante tarjetas perforadas, con las cuales se aumentaba la producción y se abarataba el costo del producto. Con este tipo de máquinas surgió el control numérico y dio inicio un proceso de automatización de las industrias, que perdura hasta nuestros días.

Poco a poco las máquinas de control numérico se fueron aplicando en otras áreas de la industria diferentes a la textil, pero el proceso era el mismo: se hacía el diseño de la pieza, después se

escribía el programa de control numérico y posteriormente se perforaban las tarjetas y se les alimentaban a la máquina.

Posteriormente, siguiendo la misma idea de facilitarle al hombre la realización de sus labores, se inventaron las computadoras electrónicas. Inicialmente se usaron para realizar operaciones matemáticas muy complejas, evitando muchos errores y haciéndolas en mucho menor tiempo. La forma en que se programaban estas computadoras era mediante tarjetas perforadas.

Gracias al desarrollo de la electrónica, las primeras computadoras, que ocupaban todo un edificio, se fueron reduciendo en tamaño y creciendo en aplicaciones. También se fue cambiando la forma de almacenar la información y se empezaron a usar cintas de papel perforado que posteriormente se cambiaron por cintas de material magnético hasta llegar a los discos flexibles o diquetes.

Cuando el hombre se dio cuenta de que una computadora podía elaborar cálculos repetitivos y que una máquina de control numérico realiza movimientos repetitivos, fue cuando decidió aplicar una computadora a un proceso de control numérico. Inicialmente la computadora sólo se usó para la perforación de las tarjetas o de la cinta de papel, ya que después de hacer la perforación, las tarjetas o la cinta se alimentaba a la máquina de control numérico. Sin embargo, cuando se empezó a usar el material magnético para la programación de las computadoras, éstas se conectaron directamente con las máquinas de control numérico, de tal forma que bastaba programar la computadora para hacer funcionar la máquina, con la ventaja de que las instrucciones quedaban almacenadas en el material magnético y podía repetirse el proceso correpondiente con mayor facilidad, y no como en el caso de la cinta de papel perforado, en el cual cada vez que se quería ejecutar el proceso, había que introducir la cinta a la máquina.

El control numérico por computadora ha seguido evolucionando a pasos agigantados en los tres aspéctos principales que lo componen: máquinas de control numérico, computadoras electrónicas (hardware) y programas de control (software). En cuanto al primer aspecto, se han creado máquinas más precisas capaces de producir piezas mecánicas extremadamente pequeñas o piezas con formas muy complejas. Podríamos decir que la máquina más sofisticada es un robot, que debe poseer un mínimo de seis grados de libertad, lo que le permite moverse prácticamente como un brazo humano.

En cuanto a las computadoras, son cada día más pequeñas en tamaño y más grandes en capacidad de memoria. También se ha mejorado considerablemente la velocidad de procesamiento de la información.

Por último gracias a la gran capacidad de memoria de las computadoras, se han llegado a desarrollar programas de diseño y de control de procesos de manufactura. El hecho de diseñar un dibujo con la computadora se conoce como CAD, del inglés Computer Aided Design (diseño asistido por computadora), y al hecho de controlar un proceso mecánico mediante la computadora se le conoce como CAM, del inglés Computer Aided Manufacturing (manufactura asistida por computadora). Los programas más sofisticados en la actualidad son aquellos que controlan toda la manufactura de una pieza mecánica a partir de un dibujo hecho en la computadora.

Actualmente en México se requieren sistemas automáticos de este tipo, para modernizar y hacer más eficientes los procesos industriales, todo esto sin elevar el costo de dichos procesos, o mejor aún, tratando de reducir el costo. Es por ello que este trabajao tiene como finalidad desarrollar un sistema elctrónico capaz de cargar datos desde una computadora, y con ellos poder operar una máquina herramienta de control numérico. La comunicación entre la compuadora y el dispositivo desarrollado

será en forma serial, es decir, los dígitos binarios de un caracter trasmitido llegarán uno tras otro y no todos juntos como en el caso de la comunicación en paralelo.

Dependiendo de la complejidad de la máquina herramienta, ésta puede tener movimiento en cinco o más ejes. El objetivo mínimo de este trabajo es el control de una máquina de 2 1/2 D (dos dimensiones y media), lo cual significa que se controlarán tres ejes, pero sólo dos de ellos simultáneamente. En otras palabras, controlaremos tres motores, dos de ellos simultáneamente y el tercero de manera independiente. La interfaz diseñada será capaz de controlar cualquier máquina de 2 1/2 D y sólo bastará, en algunos casos, hacer ciertas modificaciones al programa de comunicación de la computadora y a la máquina.

Otro de los objetivos es que, al finalizar este proyecto, la interfaz desarrollada sirva como base para futuras investigaciones, ya que pensamos que el alcance de un proyecto de control, es enorme. Entre otras mejoras, podríamos mencionar el control de más motores (ejes) y la elaboración de un programa de diseño asistido por computadora que fuera capaz de controlar a la interfaz diseñada. Es decir, pretendemos que este trabajo sirva como punto de partida para el desarrollo de un sistema de diseño y manufactura asistidos por computadora (CAD/CAM).

TEMA II
ANALISIS DEL PROBLEMA

Antes de iniciar el desarrollo del sistema electrónico, analizamos algunos temas que serían necesarios para llevar a buen término el trabajo. El análisis de estos temas nos dio una idea de los pasos que teníamos que seguir y nos ayudó también a generar varias opciones para solucionar el problema. Cuando se tenían ya las opciones bien definidas, se eligió la que cumplía más satisfactoriamente los criterios de optimación que habíamos establecido.

II.1 RECOPILACION DE INFORMACION

En esta sección mostraremos brevemente la información referente al diseño y manufactura asistidos por computadora, los aspectos principales de la comunicación serie y una breve descripción del funcionamiento de los motores de pasos.

II.1.1 MANUFACTURA ASISTIDA POR COMPUTADORA

Cualquier proceso de fabricación automática que esté controlado por computadora es conocido en ingeniería como un proceso CAM (de las siglas Computer-Aided Manufacturing, manufactura asistida por computadora). Este proceso tuvo su origen en la década de los 40 cuando John T. Parsons propuso una máquina cortadora automática que era controlada mediante tarjetas perforadas que se le alimentaban, para moverse en pequeños pasos incrementales. En 1949 la Fuerza Aérea de los Estados Unidos de América, comisionó al Laboratorio de Servomecanismos del Instituto Tecnológico de Massachusetts para desarrollar una máquina basada en el concepto de Parsons. Fue así como nació el control numérico, abreviado CN en las obras de ingeniería.

En 1957 fueron usadas las primeras instalaciones CN en la producción. Lo que empezó a causar dificultades fue el hecho de generar programas para las máquinas. Para solucionar este problema, el Instituto Tecnológico de Massachusetts, que ya había desarrollado la cinta de papel perforada, desarrolló un lenguaje de programación llamado APT (Automatically Programmed Tools), es decir, herramientas programadas automáticamente. El objetivo era tener un lenguaje simbólico capaz de programar la pieza de una manera directa, es decir, mediante instrucciones simples como círculo, línea, cortar, etc. En 1962 el primer sistema de programación APT fue usado industrialmente.

En la década de los 70 hubo un cambio en la filosofía de la producción, y el control numérico fue visto como parte de un concepto más amplio: Manufactura Asistida por Computadora (CAM).

CAM no sólo se relaciona con el control numérico, sino con el control y monitoreo de la producción, administración de la materia prima, etc. El énfasis en el uso de las computadoras en procesos de manufactura ha desarrollado nuevas formas de control

numérico: CNC (control numérico por computadora) y DNC (control numérico directo).

El control numérico no debe ser entendido como un tipo de máquina herramienta sino como una técnica para controlar una gran variedad de máquinas. En este sentido debe hablarse más bien de un programa de control numérico, que se define como un conjunto de bloques de instrucciones que manejan a la máquina para realizar una tarea específica. La instrucción más importante que una máquina CN recibe, es aquélla que la coloca en una posición. Como no todas las máquinas necesitan movimientos en todos los sentidos, su clasificación es de la siguiente manera.

a) Control axial en 2D (2 dimensiones): Proporciona movimientos de herramienta a lo largo de dos ejes simultáneamente. Una aplicación típica es el torneado.

b) Control axial en 2 1/2 D (2 dimensiones y media): Los movimientos de la herramienta pueden ser a lo largo de tres ejes, pero sólo permite movimientos simultáneos a lo largo de un máximo de dos ejes por operación. Se utiliza normalmente para perfiles de fresado en 2D básicos que requieren profundidad programable, tales como ranuras y cavidades.

c) Control axial en 3D (3 dimensiones): Los movimientos de la herramienta son a lo largo de tres ejes simultáneamente. Se pueden utilizar para cortes complejos, pero su capacidad está limitada por la variación del ángulo relativo a la pieza de trabajo durante la operación.

d) Control de 5 ejes: Proporciona movimientos de herramienta a lo largo de tres ejes simultáneamente y permite también el movimiento angular de los ejes del husillo de la herramienta. Esto posibilita a la herramienta a permanecer perpendicularmente a la superficie de trabajo en todo momento.

La *Figura II.1*, en la siguiente página, muestra los movimientos anteriores.

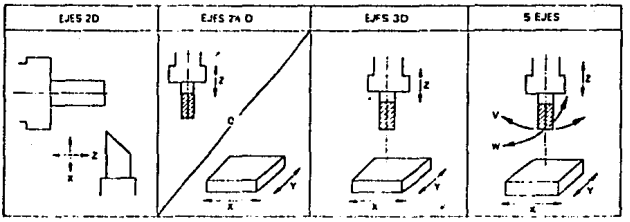


Figura II.1 Ejes de movimiento en máquinas CN

Para que la herramienta de corte inicie su trabajo, debe primero colocarse en su lugar. Para esto existen dos métodos que dependen del sistema de referencia que se tome. El primero de ellos es el posicionamiento absoluto, en el cual se fija el sistema de referencia y se alimentan a la máquina las coordenadas x, y, z actuales de cada punto, es decir, cada punto en el espacio tiene sus coordenadas específicas.

El segundo método de posicionamiento es el incremental, en el cual se toma como sistema de referencia la última posición, es decir, la posición en la que esté la máquina antes de efectuar otro movimiento.

Una vez que se han fijado los puntos, existen tres métodos para mover la herramienta de una coordenada a otra. El primero de ellos es conocido como "sistema punto a punto". En este método cada eje es movido independientemente, por lo cual el movimiento de una coordenada a otra es por pasos. Este tipo de movimiento puede ser empleado sólo en aplicaciones de control numérico de operación discreta, como por ejemplo perforaciones. El segundo método se conoce como "corte recto", y consiste en producir una diagonal, manteniendo una relación uno a uno entre el movimiento de los dos ejes, hasta llegar a un punto en el que el movimiento de uno de los dos ejes pueda ser nulo, para llegar a la coordenada deseada.

El tercer método, conocido como "sistema contorno", es el más versátil y sofisticado de todos. Mediante este método se genera una línea recta al interpolar las coordenadas extremas, con lo cual la máquina herramienta se transporta directamente de una coordenada a otra. Algunos sistemas tienen capacidades adicionales de interpolación y prácticamente se puede trazar cualquier figura con este método.

En la *Figura II.2* se compara el movimiento típico de cada uno de los sistemas.

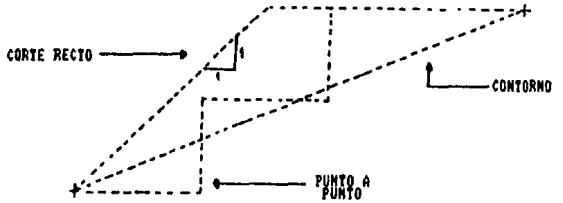


Figura II.2 Sistemas de control de trayectorias

El programa CN más común es aquél que comanda el maquinado completo de una pieza. Este tipo de programa se denomina "programa de pieza" y es uno de los principales componentes de un proceso CAM.

Una máquina CN sólo funcionará si las instrucciones adecuadas se desarrollan y son trasmitidas al control de la máquina. Este proceso es conocido como ciclo de comunicación y comienza cuando se desarrolla el programa de pieza.

El siguiente paso en el ciclo de comunicación es la transferencia física del programa de pieza a la unidad de control de la máquina. Un programa de pieza puede contener cientos o miles de caracteres alfanuméricos y símbolos esreciales. Por esta razón

los datos son codificados en formatos compactos que pueden ser fácilmente descifrados por la unidad de control de la máquina.

Existen cuatro tipos básicos para alimentar el programa de pieza a la máquina CN. Estos son de tarjeta perforada, de cinta perforada, mediante un medio magnético y mediante una comunicación directa de una computadora con la máquina CN. Cada tipo de alimentación equivale también al tipo de almacenamiento de la información y representa un incremento en el nivel de sofisticación de la máquina.

Tarjetas perforadas

La tarjeta perforada tiene una aplicación limitada actualmente. El código de tarjeta perforada fue diseñado por Herman Hollerith en 1887 y el método mostró gran eficiencia, ya que era relativamente sencillo intercambiar las tarjetas con nuevas instrucciones con lo cual la secuencia de la máquina cambiaría. Aunque en una tarjeta perforada se puede ocupar cualquier tipo de código, el más común es el BCD (Decimal Codificado en Binario).

Algunos tipos de tarjetas estándares son la IBM de 80 columnas y las Remington Rand de 45 y 90 columnas; la única diferencia entre ellas es la separación de las perforaciones. Para aplicaciones de control numérico la más usada es la tarjeta IBM, que tiene las siguientes características: 3.25 pulgadas (82.55 mm) de ancho, 7.375 pulgadas (187.33 mm) de largo y 0.007 pulgadas (0.178 mm) de espesor. Las perforaciones, que tienen un tamaño y separación específicos, están distribuidas en 12 renglones y 80 columnas. El código para cada caracter se representa definiendo dos secciones. Los tres renglones superiores son llamados renglones de zona y los nueve renglones inferiores representan los dígitos.

Cinta perforada

Este ha sido el método más empleado en las máquinas CN. La medida del carrete está estandarizada a 1 pulgada (25.4 mm) de ancho. Las cintas son de diversos materiales como el papel, papel y plástico, aluminio y plástico; esto depende principalmente de las características del dispositivo lector de la cinta.

Antes de alimentar a la máquina CN con la cinta perforada, deben seguirse algunos pasos. Estos se muestran en la *Figura II.3*.

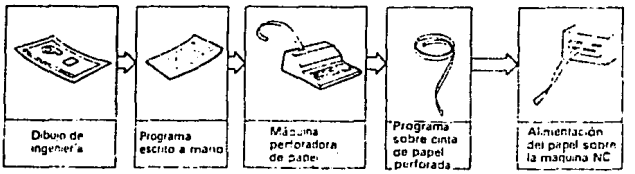


Figura II.3 Alimentación de una máquina CN con cinta de papel perforado

La perforación de la cinta es hecha mediante un aparato que tiene un teclado parecido al de una máquina de escribir, con algunas teclas de funciones especiales. La cinta perforada también puede generarse desde una computadora que esté conectada a una máquina perforadora automática. De esta forma se puede almacenar la información en un medio magnético, aunque la máquina CN trabaje con cinta perforada. La velocidad de entrada de los datos a la máquina CN, depende de la velocidad de lectura y varía desde 10 hasta 1000 caracteres por segundo.

Al no tener memoria, la máquina simplemente lee cada bloque de instrucciones y realiza la operación que se le indica. De esta forma, cada vez que se requiera el maquinado de esa pieza bastará

con alimentar a la máquina esa cinta perforada, que puede almacenarse para usos futuros.

Tanto en una cinta de papel perforada, como en una tarjeta perforada, el "1" binario se representa mediante una perforación y el "0" binario, mediante la ausencia de perforación. Una cinta perforada tiene la estructura que se muestra en la *Figura II.4*.

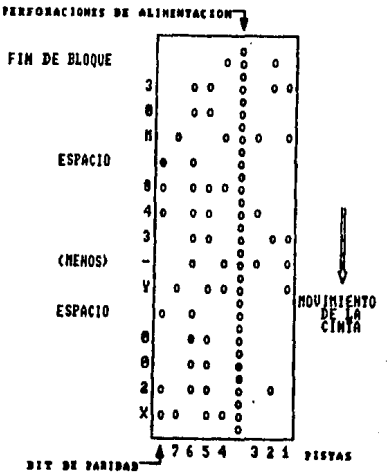


Figura II.4 Estructura de una cinta de papel perforado

El diagrama muestra 8 pistas. La perforación más a la izquierda se utiliza como una verificación de paridad, para detectar posibles errores durante la trasmisión de datos. Entre las pistas 3 y 4 existe una fila de perforaciones destinadas a ayudar al transporte de la cinta. Todos los signos numéricos tienen un "1" binario en las pistas 5 y 6. Todas las letras tienen un "1" binario en la pista 7. Debe destacarse que el sistema de

numeración no es estrictamente binario, sino que al igual que en las tarjetas perforadas se utiliza el sistema BCD.

Material magnético

El uso de material magnético comenzó en la década de los 40 cuando se crearon cintas con una película de óxido de hierro para grabar pulsos magnéticos.

Los códigos usados en este método son los mismos que aquéllos empleados en las tarjetas y cintas perforadas. Sin embargo, las cintas magnéticas son generadas usando computadoras y no pueden ser editadas manualmente. En sus principios, la cinta magnética tuvo una aplicación limitada en control numérico, debido a lo frágil del medio, ya que la humedad, el aceite y el polvo podían causar errores de lectura, y en el medio ambiente de la industria los metales ferrosos y las herramientas magnetizadas alterarían los datos de la cinta.

Posteriormente surgieron los "cassettes" de cinta magnética, similares a aquéllos utilizados en grabaciones de sonido. Estos tuvieron una aplicación mayor aunque no fueron del todo favorables, ya que las vibraciones de la máquina podían mover las cabezas de lectura, provocando errores en la señal de información.

Actualmente existe un medio magnético más confiable, que es el disco flexible. Un disco flexible puede contener la información equivalente a 2000 pies (609.6 m) de cinta perforada, y su aplicación está directamente relacionada con el método que se explica a continuación.

Comunicación directa con una computadora

El continuo desarrollo de los métodos de almacenamiento ha llevado a la creación de otras formas del control numérico, como son el CNC (control numérico computarizado) y el DNC (control numérico directo). En un sistema CNC, la unidad de control de la máquina se remplaza por una microcomputadora programable, con lo cual los programas CN pueden ser almacenados en la memoria de la computadora, reduciéndose así el uso de medios físicos externos de almacenamiento.

Un sistema DNC es el más sofisticado de los sistemas CN, ya que pueden ser controladas varias máquinas simultáneamente. Todos los programas CN son almacenados en dispositivos periféricos de gran capacidad de almacenamiento. El programa CN se ejecuta en la unidad central de proceso de una computadora digital y puede mandar información a velocidades de 160,000 caracteres por segundo.

Para el almacenamiento de datos, sin importar el medio en el que se almacenen, existen cuatro códigos que son el decimal para cálculos manuales, el octal y el hexadecimal para aplicaciones de computadora y el binario para control electrónico. De todos estos códigos, para aplicaciones CN se usa el binario, ya que la unidad de control de la máquina contiene circuitería electrónica, que sólo responde a las condiciones de apagado y prendido. En un sistema binario estas condiciones son fácilmente representadas mediante "0" y "1", y se conocen como dígitos binarios o bits (del inglés binary digits).

Un sistema binario puro es raramente usado en aplicaciones CN. En su lugar, como ya se dijo antes, se ocupa uno conocido como BCD o código ASCII (American Standard Code for Information Interchange, código americano estándar para intercambio de información).

Este código emplea números binarios de 7 bits para representar todos los caracteres alfanuméricos del programa de pieza.

El conjunto de estos caracteres se muestra en la *Tabla II.1.*

CARACTER	CODIGO BINARIO $2^6 2^5 2^4 2^3 2^2 2^1 2^0$	DECIMAL EQUIVALENTE
0	0 1 1 0 0 0 0	48
1	0 1 1 0 0 0 1	49
2	0 1 1 0 0 1 0	50
3	0 1 1 0 0 1 1	51
4	0 1 1 0 1 0 0	52
5	0 1 1 0 1 0 1	53
6	0 1 1 0 1 1 0	54
7	0 1 1 0 1 1 1	55
8	0 1 1 1 0 0 0	56
9	0 1 1 1 0 0 1	57
A	1 0 0 0 0 0 1	65
B	1 0 0 0 0 1 0	66
C	1 0 0 0 0 1 1	67
D	1 0 0 0 1 0 0	68
E	1 0 0 0 1 0 1	69
F	1 0 0 0 1 1 0	70
G	1 0 0 0 1 1 1	71
H	1 0 0 1 0 0 0	72
I	1 0 0 1 0 0 1	73
J	1 0 0 1 0 1 0	74
K	1 0 0 1 0 1 1	75
L	1 0 0 1 1 0 0	76
M	1 0 0 1 1 0 1	77
N	1 0 0 1 1 1 0	78
O	1 0 0 1 1 1 1	79
P	1 0 1 0 0 0 0	80
Q	1 0 1 0 0 0 1	81
R	1 0 1 0 0 1 0	82
S	1 0 1 0 0 1 1	83
T	1 0 1 0 1 0 0	84
U	1 0 1 0 1 0 1	85
V	1 0 1 0 1 1 0	86
W	1 0 1 0 1 1 1	87
X	1 0 1 1 0 0 0	88
Y	1 0 1 1 0 0 1	89
Z	1 0 1 1 0 1 0	90
Espacio	0 1 0 0 0 0 0	32
&	0 1 0 0 1 1 0	38
'	0 1 0 1 1 1 1	63
!	0 1 0 0 0 0 1	33

CARACTER	CODIGO BINARIO $2^6 2^5 2^4 2^3 2^2 2^1 2^0$	DECIMAL EQUIVALENTE
(	0 1 0 1 0 0 0	40
)	0 1 0 1 0 0 1	41
.	0 1 0 0 1 1 1	39
:	0 1 0 1 1 1 0	46
;	0 1 1 1 0 1 0	58
-	0 1 1 1 0 1 1	59
a	1 1 0 0 0 0 1	97
b	1 1 0 0 0 1 0	98
c	1 1 0 0 0 1 1	99
d	1 1 0 0 1 0 0	100
e	1 1 0 0 1 0 1	101
f	1 1 0 0 1 1 0	102
g	1 1 0 0 1 1 1	103
h	1 1 0 1 0 0 0	104
i	1 1 0 1 0 0 1	105
j	1 1 0 1 0 1 0	106
k	1 1 0 1 0 1 1	107
l	1 1 0 1 1 0 0	108
m	1 1 0 1 1 0 1	109
n	1 1 0 1 1 1 0	110
o	1 1 0 1 1 1 1	111
p	1 1 1 0 0 0 0	112
q	1 1 1 0 0 0 1	113
r	1 1 1 0 0 1 0	114
s	1 1 1 0 0 1 1	115
t	1 1 1 0 1 0 0	116
u	1 1 1 0 1 0 1	117
v	1 1 1 0 1 1 0	118
w	1 1 1 0 1 1 1	119
x	1 1 1 1 0 0 0	120
y	1 1 1 1 0 0 1	121
z	1 1 1 1 0 1 0	122
+	0 1 0 1 0 1 1	43
-	0 1 0 1 1 0 1	45
*	0 1 0 1 0 1 0	42
/	0 1 0 1 1 1 1	47
<	0 1 1 1 1 0 0	60
=	0 1 1 1 1 0 1	61
>	0 1 1 1 1 1 0	62

Tabla II.1 Código ASCII para aplicaciones CN

La información es trasmitida con un cierto formato a la unidad de control de la máquina. De acuerdo al formato cada bloque debe ser capaz de especificar los datos dimensionales y adimensionales. Los primeros incluyen los comandos de movimiento lineal y angular

de la máquina. Los datos adimensionales son las funciones preparatorias usadas para describir tipos específicos de movimiento, las funciones misceláneas que controlan la operación de la máquina así como las especificaciones de alimentación, velocidad y las características de la herramienta.

Por convención un bloque CN debe llevar el siguiente orden:

n g xyzab f s t m eob

donde:

n = número de secuencia
g = función preparatoria
xyzab = datos dimensionales
f = función de alimentación
s = función de velocidad
t = función de herramienta
m = función miscelánea
eob = fin de bloque.

Para la entrada de datos CN son usados cuatro formatos básicos, los cuales son:

Formato de arreglo secuencial. Requiere que cada bloque sea de la misma longitud y contenga el mismo número de caracteres. Esta restricción hace que el bloque sea dividido en sub-bloques que corresponden a cada uno de los datos mencionados anteriormente. Como la longitud del bloque es invariable, todos los valores deben aparecer aunque haya datos repetidos. Todos los valores tienen un número predefinido de dígitos.

Formato de bloque direccionado. Elimina la necesidad de dar información redundante en bloques sucesivos, mediante la especificación de un cambio de código. El cambio de código sigue al número de secuencia del bloque e indica qué valores deben ser cambiados con respecto a los bloques precedentes. Todos los datos deben contener un número predefinido de dígitos.

Formato de tabulador secuencial. Usa un símbolo especial llamado tabulador (tab) para separar los valores de los datos de un bloque. Dos o más tabuladores seguidos indican que los datos que normalmente irían en esa posición son redundantes, y por lo tanto son omitidos.

Formato de palabra direccionada. Es el único esquema de entrada que usa datos alfanuméricos (letras y números). Cada valor numérico es precedido por una letra que indica el tipo de dato. De esta manera, la información redundante se omite al eliminar la letra apropiada. En algunos tipos de control que utilizan este formato, los ceros iniciales y finales pueden ser omitidos, con lo cual la longitud del bloque se reduce.

Los valores adimensionales del formato pueden obtenerse de dos formas. En la primera, los valores son seleccionados de estándares predefinidos. En la segunda forma, los valores son calculados de acuerdo a ciertas operaciones. Del primer tipo encontramos el número de secuencia y las funciones preparatorias, misceláneas y de herramienta.

El número de secuencia (código n) se usa para identificar cada bloque del programa CN, con lo cual cualquier comando puede ser localizado rápidamente. En general, las unidades de control requieren que estos números sean introducidos en forma ascendente.

La función preparatoria (código g), se usa como un elemento de comunicación para preparar a la unidad de control de la máquina a realizar una función tal como una interpolación lineal. Una lista de los códigos g se muestran en la *Tabla II.2*.

Las funciones misceláneas (código m), son usadas para designar un modo particular de operación, y en general están relacionadas con condiciones opuestas de operación, por ejemplo, refrigerante

activado, refrigerante desactivado. La *Tabla II.3* muestra los códigos para las funciones misceláneas.

La función de herramienta (código t) se usa junto con la función miscelánea para cambios de herramientas (código m06) y como un medio de direccionar la nueva herramienta. Los centros de maquinado de control numérico, tienen un aparato automático para cambios de herramienta. Los códigos t pueden tener números de hasta cinco dígitos de longitud. El número depende del orden en que se seleccionen las herramientas en la torreta.

CODIGO	FUNCION
000	POSICIONAMIENTO PUNTO A PUNTO
001	INTERPOLACION LINEAL
002	INTERPOLACION CIRCULAR CW
003	INTERPOLACION CIRCULAR CCW
004	SOLTAR
005	AGARRAR
006-007	DESASIGNADO
008	ACELERACION
009	DESACELERACION
010	INTERPOLACION LINEAL PARA DIMENSIONES LARGAS
011	INTERPOLACION LINEAL PARA DIMENSIONES CORTAS
012	DESASIGNADO
013-016	SELECCION DEL EJE
017	SELECCION DEL PLANO XY
018	SELECCION DEL PLANO ZX
019	SELECCION DEL PLANO YZ
020	INTERPOLACION CIRCULAR CW PARA DIMENSIONES LARGAS
021	INTERPOLACION CIRCULAR CW PARA DIMENSIONES CORTAS
022-027	DESASIGNADO
028-029	DESASIGNADOS PERMANENTEMENTE
030	INTERPOLACION CIRCULAR CCW PARA DIMENSIONES LARGAS
031	INTERPOLACION CIRCULAR CCW PARA DIMENSIONES CORTAS
032	DESASIGNADO
033	CORTE CONSTANTE
034	CORTE INCREMENTAL
035	CORTE DECREMENTAL
036-039	RESERVADOS PARA CONTROL
040	COMPENSACION DE CORTE CANCELADA
041	COMPENSACION DE CORTE A LA IZQUIERDA
042	COMPENSACION DE CORTE A LA DERECHA
043-049	SI SE USA ES PARA COMPENSACION DE CORTE, SI NO ES DESASIGNADA
050-059	DESASIGNADO
060-079	RESERVADO PARA POSICIONAMIENTO
080	CICLO DE FIJADO CANCELADO
081	CICLO DE FIJADO 1
082	CICLO DE FIJADO 2
083	CICLO DE FIJADO 3
084	CICLO DE FIJADO 4
085	CICLO DE FIJADO 5
086	CICLO DE FIJADO 6
087	CICLO DE FIJADO 7
088	CICLO DE FIJADO 8
089	CICLO DE FIJADO 9
090-099	DESASIGNADO

Tabla II.2 Funciones preparatorias

CODIGO	FUNCION
M00	DETENER EL PROGRAMA
M01	PARADA OPCIONAL
M02	FIN DEL PROGRAMA
M03	BROCA CW
M04	BROCA CCW
M05	APAGAR BROCA
M06	CAMBIO DE HERRAMIENTA
M07	ENFRIADOR 2 ENCENDIDO
M08	ENFRIADOR 1 ENCENDIDO
M09	ENFRIADOR APAGADO
M10	ATORNILLAR
M11	DESATORNILLAR
M12	DESASIGNADO
M13	BROCA CW Y ENFRIADOR ENCENDIDO
M14	BROCA CCW Y ENFRIADOR ENCENDIDO
M15	MOVIMIENTO +
M16	MOVIMIENTO -
M17-M24	DESASIGNADO
M25-M29	DESASIGNADO PERMANENTEMENTE
M30	FIN DE LA CINTA
M31	DESVIO DE ENCLAVAMIENTOS
M32-M35	VELOCIDAD CONSTANTE DE CORTE
M36-M39	DESASIGNADO
M40-M45	SI SE USA ES PARA CAMBIOS DE ENGRANES, SI NO ESTA DESASIGNADO
M46-M49	RESERVADOS PARA CONTROL
M50-M99	DESASIGNADOS

Tabla II.3 Funciones misceláneas

Los valores adimensionales que son calculados mediante operaciones son las funciones de alimentación y velocidad. Para realizar estos cálculos existen dos métodos: el "Código 3-mágico" y el "Código del tiempo inverso". En ambos se tiene por objetivo que cualquier valor pueda ser representado usando sólo tres y cuatro dígitos respectivamente.

Código 3-mágico

Para codificar datos con este método, el punto decimal debe moverse de tal forma que las cifras enteras sean nulas y el primer dígito diferente de cero ocupe el lugar de los décimos. Después se expresa la cantidad en notación científica, redondeándola a dos cifras significativas. El primer dígito del código será el exponente obtenido previamente más 3. De aquí el nombre del método. Por ejemplo, considérese una velocidad de alimentación de 2.44 rpm. Siguiendo el procedimiento de codificación tenemos:

2.44 = .24E1
Entonces 1 + 3 = 4
Por lo tanto 2.44 = 424

Debido a la definición del código, no todos los valores pueden ser expresados con él. El primer dígito no debe ser negativo, debe ser siempre cero o positivo. El segundo no puede ser cero, por lo tanto el número más pequeño del código será 010. Si efectuamos el procedimiento de codificación al revés; el exponente será 0-3=-3. Es decir, el código 010 equivale al número 0.10E-3 = 0.00010. El primer dígito más grande es 9, por lo cual el valor máximo del código será 999 que equivale a 0.99E6 = 990000.

Código del tiempo inverso

Con éste se obtiene una palabra codificada que es equivalente al recíproco del tiempo de interpolación en minutos. El tiempo inverso de alimentación Fc puede expresarse como:

$$Fc = 1/t = V/S \qquad \text{(Interpolación lineal)}$$

o

$$Fc = \theta/t = V/R \qquad \text{(Interpolación circular)}$$

donde:

- V = velocidad en el eje de interpolación lineal o velocidad de alimentación para interpolación circular
- S = distancia entre puntos
- R = radio del arco
- θ = longitud del arco en radianes
- t = tiempo.

Para especificar el código f se requieren cuatro dígitos, lo cual hace que este código tenga una desventaja importante cuando las velocidades son muy bajas, ya que al redondear habrá un error porcentual grande, es decir, si por ejemplo el resultado de la operación es 0.00135, al redondear a 4 cifras obtendremos 0.0014, lo que equivale a una diferencia del 3.7% del valor original.

Este problema se soluciona incrementando la resolución del código, aunque se empleen más dígitos para la palabra de control.

II.4.1 Enlace CAD/CAM

El control numérico por computadora puede incrementar su funcionalidad si se conectan varias máquinas a una computadora central que tendrá la función de crear los programas de pieza correspondientes y alimentar los datos directamente a cada máquina. Este es el método conocido como control numérico directo, mencionado anteriormente, y su principal ventaja es que la máquina de control numérico puede interconectarse, por medio de la computadora central, a una base de datos de un sistema CAD (Diseño Asistido por Computadora).

El proceso de interconectar una máquina de control numérico a un sistema CAD, es lo que se conoce como enlace CAD/CAM. Actualmente este proceso cobra cada día más importancia, debido a la eficacia de sus resultados, ya que permite el diseño más rápido de las piezas necesarias para algún determinado equipo y después, directamente, se puede maquinar la pieza sin necesidad de hacer un programa de control numérico, es decir, la computadora de interconexión desarrolla automáticamente los códigos de control numérico y los alimenta a la máquina.

Sin embargo, aunque los sistemas CAD presentan muchas comodidades, entre las que destacan la rapidez del dibujo, la posibilidad de tener proyecciones en dos y tres dimensiones y la facilidad de realizar análisis complejos, no todos los sistemas CAD pueden interconectarse a una máquina de control numérico. Los sistemas interconectables emplean una técnica conocida como "control numérico gráfico" que se realiza en pantallas VDU (unidad de representación visual). Esta técnica puede proporcionar también simulaciones de corte de herramienta y la realización de cálculos trigonométricos complejos.

Los sistemas CAD interconectables, se dividen en tres bloques principales, que se describen a continuación:

a) Definición de la geometría. Supone la descomposición de la forma del objeto o pieza en sus elementos geométricos primitivos, entre los que se incluyen puntos, líneas, círculos, conos y esferas. Mientras en la pantalla se puede ver un círculo o una línea, también se guardan estas instrucciones en un archivo a parte, que después se utilizará para la fabricación.

b) Procesador de fabricación. Utiliza las definiciones geométricas para generar los datos necesarios para producir el componente. Los datos típicos que se introducen en esta etapa son el tamaño de la herramienta de corte, la velocidad del husillo y la velocidad de alimentación. La información relativa a la dirección del movimiento de la herramienta y el camino de corte necesario, se introducen también en esta etapa.

c) Posprocesador. En esta parte se convierten las instrucciones del programa de dibujo en instrucciones codificadas que pueden ser entendidas por la máquina de control numérico.

El enlace entre la computadora central y las máquinas de control numérico, está dado en forma serial mediante un simple cable plano que cumpla con el estándar RS-232C. En la siguiente sección se explicará brevemente la comunicación serial.

II.1.2 COMUNICACION SERIAL

La comunicación serial, o serie como se mencionará a partir de este momento, surgió debido a la necesidad que se tenía de reducir el número de cables para trasmitir señales digitales a grandes distancias. La mejor forma que se encontró para hacer esto fue a través de las líneas telefónicas y el uso de los módems. Un módem (modulador-demodulador) es un dispositivo que, del lado trasmisor, modula las señales digitales para enviarlas por la línea telefónica en forma de cambios de frecuencia y, del

del lado receptor, demodula estos cambios de frecuencia para convertirlos en pulsos digitales.

La forma en que se trasmiten los pulsos serie puede ser síncrona o asíncrona. En la comunicación síncrona debe esperarse una señal de tiempo para empezar a trasmitir y recibir, mientras que en la asíncrona, el envío y recepción de datos no depende de ninguna señal de tiempo.

Sin embargo, como puede haber trasmisión de datos en cualquier momento, es necesario que la palabra digital que se vaya a trasmitir, tenga cierto formato que reconozcan todos los dispositivos serie. Este formato se muestra en la *Figura II.5.*

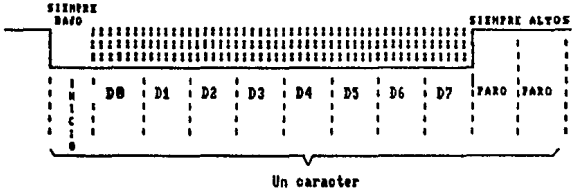


Figura II.5 Formato de una palabra serie asíncrona

El bit de inicio debe ser siempre un nivel bajo. Los bits de datos, marcados con Dn, pueden ser 4, 5, 6, 7 u 8. El bit de paridad puede estar o no presente, y en caso de que esté, puede ser "1" o "0". Los bits de parada deben ser siempre altos y pueden ser 1, 1 1/2 ó 2 bits. De esta forma, una comunicación serie debe empezar siempre con un nivel bajo y acabar con un nivel alto.

El bit de paridad sirve para detectar errores en la comunicación. Los tipos de paridad que existen son: par, impar, de marca, de espacio y no paridad.

La paridad par indica que la suma de todos los bits de datos, incluyendo el bit de paridad, deben dar un número par de 1's. Por ejemplo, si la palabra fuera

1001010

el bit de paridad debería ser entonces, un "1". Si la palabra es

1100000

entonces, el bit de paridad sería un "0".

Para la paridad impar, la suma de los bits de datos, incluyendo el bit de paridad, debe ser un número impar de 1's. La paridad de marca significa que el bit de paridad siempre es "1". La de espacio, es cuando el bit siempre es "0". No paridad es cuando no se trasmite bit de paridad y no se comprueba paridad en la recepción.

Los datos serie pueden ser enviados a diferentes velocidades, pero es muy importante que se trasmitan y se reciban a la misma velocidad. La velocidad con que se trasmiten y/o reciben los datos, se conoce como baudaje (su unidad es el baud y se representa por Bd) y se define como el inverso del tiempo que tarda en enviarse un bit. Si, por ejemplo, un bit tarda en enviarse 1.67 ms, entonces el baudaje será: 1/1.67E-3 = 600Bd. Los valores de baudaje comunes son: 75, 110,150, 300, 600, 1200, 2400, 4800, 9600 y 19200 Bd.

II.1.2.1 Estándar RS-232C

Para que una comunicación serie funcione correctamente se necesita, a parte de las características ya mencionadas, que los niveles de voltaje, que representan el "1" y el "0" sean iguales para cada caso; se necesita también de unas señales de protocolo para preparar a los dispositivos a trasmitir o recibir. Todo esto se incluye en el estándar RS-232C que se explica a continuación.

Publicado por la Asociación de Industrias Electrónicas (EIA por sus iniciales en inglés), el estándar RS-232C define las señales de protocolo, los niveles de voltaje de las señales y un conector de 25 terminales, necesarios para la comunicación serie.

Actualmente este estándar es el más usado para la comunicación serie y es el que viene incluido en la mayoría de las computadoras personales. Está especificado para una distancia máxima de 15 m (50 pies) a una velocidad máxima de 20,000 Bd. Para velocidades de trasmisión menores, se pueden usar cables de 610 a 915 m (2,000 a 3,000 pies). Existen otras versiones de este estándar, como el RS-422 y el RS-423, en los que se ve mejorada la velocidad de trasmisión principalmente.

El conector RS-232C tiene 25 terminales, aunque en la mayoría de los sistemas serie, sólo se ocupan las terminales 1 a 8 y la 20; exisitendo inclusive computadoras que traen un conector serie RS-232 de 9 terminales. La razón de esta reducción de terminales es que las patas restantes son líneas secundarias, es decir, líneas que tienen la misma función pero que nos son usadas a menos que sean sistemas muy complejos de trasmisión. Por este motivo se describirán sólo las terminales más comunes.

La *Figura II.6* muestra las terminales principales y su descripción, así como si son entradas o salidas. La *Figura II.7* muestra cómo es la distribución de funciones para un conector de 9 terminales, así como la forma física de ambos conectores.

Las terminales descritas en ambas figuras, con excepción de TxD, RxD y GND, son las que conforman las señales de protocolo. Un protocolo es la forma en que la computadora y la interfaz se piden permiso para trasmitir datos, para avisar que ya están preparados para recibir datos, para interrumpir la trasmisión, etc.

TERMINAL #	NOMBRE	DESCRIPCION	ENTRADA/SALIDA
1		TIERRA DE PROTECCION	
2	TxD	TRASMISION DE DATOS	SALIDA
3	RxD	RECEPCION DE DATOS	ENTRADA
4	$\overline{RTS}$	PETICION DE TRASMITIR	SALIDA
5	$\overline{CTS}$	BORRAR PARA TRASMITIR	ENTRADA
6	$\overline{DSR}$	CONJUNTO DE DATOS LISTO	ENTRADA
7	$\overline{GND}$	TIERRA	
8	$\overline{CD}$	DETECTORA DE PORTADORA DE DATOS	ENTRADA
9 a 19	SECUNDARIAS	NO CONECTAR	
20	$\overline{DTR}$	PREPARADO PARA TRASMITIR	SALIDA
21	SECUNDARIA	NO CONECTAR	
22		INDICADOR DE LLAMADA	
23 a 25	SECUNDARIAS	NO CONECTAR	

Figura II.6 Descripción de las terminales RS-232C

TERMINAL #	NOMBRE
1	$\overline{CD}$
2	RxD
3	TxD
4	$\overline{DTR}$
5	$\overline{GND}$
6	$\overline{DSR}$
7	$\overline{RTS}$
8	$\overline{CTS}$
9	INDICADOR DE LLAMADA

(a)

(b)

(c)

Figura II.7 a)Descripción del conector de 9 terminales. b)Conector de 25 terminales c)Conector de 9 terminales.

El protocolo, por lo tanto, es la forma en que dialogan computadora e interfaz y se lleva a cabo de la siguiente manera: después de encender la computadora, ésta corre una rutina de autoverificación y envía la señal DTR a la interfaz. La interfaz responde con la señal DSR para indicar que está funcionando. La computadora envía la señal RTS a la interfaz y después de un intervalo apropiado de tiempo, la interfaz responde con CTS. La computadora entonces envía los datos serie por la terminal TxD.

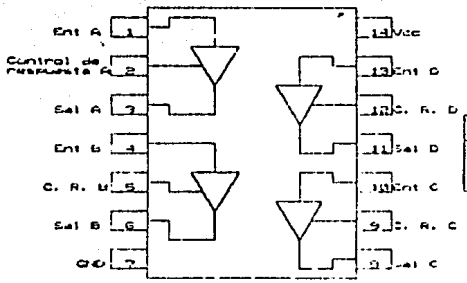
Cuando la computadora debe recibir datos, el protocolo es el mismo, sólo que la computadora envía las señales que enviaba la interfaz y viceversa, enviando la interfaz, además, la señal CD.

Un protocolo posterior puede llevarse a cabo, ya dentro del programa, para indicarle al dispositivo externo el inicio de la trasmisión y el fin de datos, o por parte del dispositivo hacia la computadora para pedirle más datos, etc. Este protocolo se explicará cuando se desarrolle el programa de comunicación definitivo.

En cuanto a los niveles de voltaje, las señales RS-232C están estandarizadas a los siguientes valores: un "1" lógico o marca, es un voltaje entre -3 y -15 volts. Un "0" lógico o espacio, es un voltaje entre +3 y +15 volts.

Debido a que la interfaz trabajará con circuitos TTL y CMOS, deben convertirse estos voltajes a niveles de +5 V para el "1" y 0 V para el "0". Esto se logra utilizando los circuitos integrados de Motorola MC1488 y MC1489, mismos que se explican a continuación.

Para convertir las señales RS-232C a señales TTL, utilizamos el circuito MC1489. El circuito y su tabla de verdad se presentan en la *Figura II.8*. Se polariza con +5 V y tierra.



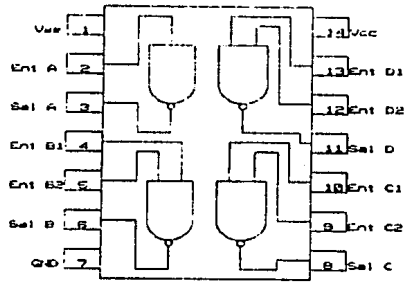
(a)

ENTRADA RS-232C	SALIDA TTL	NIVEL LOGICO
+12 V	0 V	0
-12 V	+5 V	1

(b)

Figura II.8 a) Interior y b) Tabla de verdad del MC1489

El MC1488 es el circuito utilizado para convertir señales TTL a señales RS-232C. Se polariza con +12 V, -12 V y tierra. Tanto el circuito como su tabla de verdad se muestran en la *Figura II.9*.



(a)

ENTRADA TTL	SALIDA RS-232C	NIVEL LOGICO
0 V	+12 V	0
+5 V	-12 V	1

(b)

Figura II.9 a) Interior y b) Tabla de verdad del MC1488

II.1.2.2 Conexiones serie especiales

Lo que se conecta al puerto serie de una computadora, generalmente es un módem. El módem, como ya se dijo antes,

trasmite y recibe señales entre computadoras distantes a través de la línea telefónica. Sin embargo, cuando no se emplea la línea telefónica para la comunicación, se puede ocupar una de las configuraciones especiales que son conocidas como "Módem nulo" y "Minimódem".

La configuración módem nulo sirve para conectar dos computadoras que no estén muy alejadas. La *Figura II.10*, en la siguiente página, muestra la forma de conectarlas. Nótese que las terminales van cruzadas, de tal forma que las señales que sirven de pregunta a una computadora, a la otra le sirven de respuesta.

En la configuración minimódem sólo se usa una computadora que se conecta a cualquier dispositivo externo. Esta conexión se presenta en la *Figura II.11* de la siguiente página.

Debe notarse que en este último caso, el dispositivo externo no necesita generar las señales de protocolo, ya que la misma computadora se pregunta y se responde sola. La única característica especial que debe poseer el dispositivo externo, es la de ser capaz de recibir y trasmitir datos a la velocidad que la computadora ordene. Cabe señalar que esta configuración fue la que empleamos para hacer las pruebas iniciales de trasmisión y recepción de la computadora.

II.1.3 MOTORES DE PULSOS

Cuando un proceso mecánico requiere de mucha precisión, se usan los motores de pulsos. Estos motores, conocidos también como motores de pasos, se hacen girar mediante pulsos digitales en uno u otro sentido, dependiendo del orden de los pulsos.

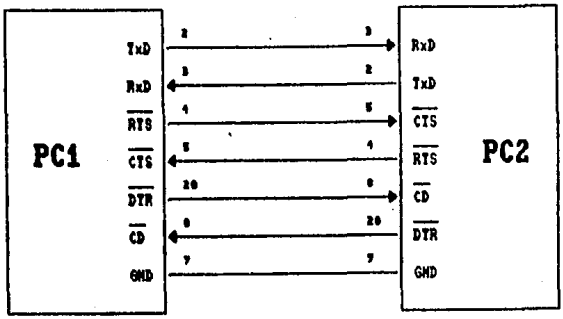


Figura 11.10 Configuración módem nulo

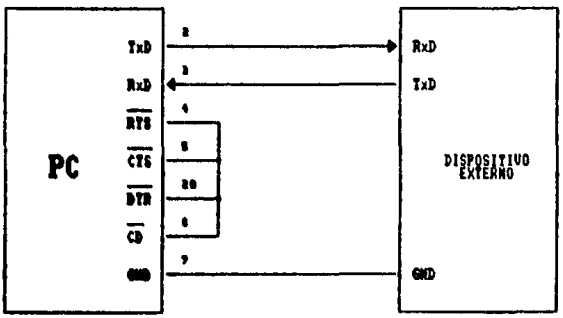


Figura 11.11 Configuración minimódem

En general los pulsos con que funcionan los motores de pasos son:

A	B	$\overline{A}$	$\overline{B}$
0	0	1	1
0	1	1	0
1	1	0	0
1	0	0	1

La forma en que estos pulsos generan el movimiento, se explicará tomando como base la *Figura II.12*.

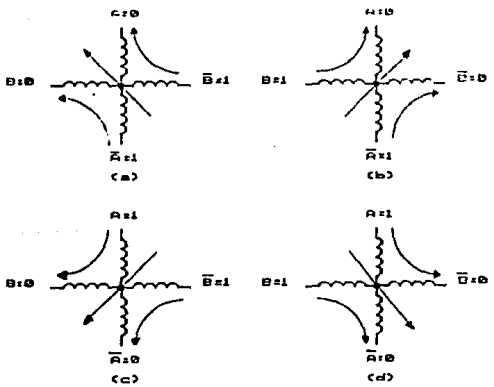


Figura II.12 Movimiento del motor de pulsos

En las figuras a), b), c) y d), se muestra cómo, al polarizar dos devanados y los otros dos mandarlos a tierra, se genera un par que hace que el motor gire. De la *Figura II.12* se puede apreciar también que los devanados que se polaricen deben ser adyacentes, ya que si fueran opuestos no se generaría el par que lo haría girar.

Entre las principales ventajas que presentan los motores de pasos, está la gran facilidad con la que se pueden hacer girar en uno u otro sentido y, sobre todo, la facilidad con la que se pueden frenar, ya que basta con dejar de mandarles los pulsos y el motor se detendrá automáticamente. Debido a esta facilidad de frenado es por lo que se ocupan estos motores para procesos mecánicos de alta precisión.

El grado de precisión de un motor se mide en el número de pasos que puede dar en una vuelta completa. Así, mientras más pasos tenga en cada revolución, será más preciso, ya que cada paso será de menos grados. Por ejemplo, si un motor es de 100 pasos por revolución, cada paso será de 3.6 grados. Por lo tanto, si otro motor es de 200 pasos por revolución (paso = 1.8 grados), será más preciso que el primero. El número de pasos que un motor puede dar en cada revolución, depende únicamente de sus características de diseño, y no puede ser alterado.

Uno de los aspectos en que se debe tener más cuidado para hacer funcionar un motor de pasos, es el ancho de los pulsos, ya que si son muy angostos, el motor no los reconocerá y no se moverá. El ancho mínimo de los pulsos es otra característica de diseño, y va relacionada con la velocidad máxima del motor.

Lo anterior debe tomarse en cuenta al desarrollar la etapa de aceleración de los motores, ya que si se pretenden mover a máxima velocidad desde el principio, y tienen mucha carga acoplada a su flecha, será difícil que puedan arrancar. Por otro lado, la etapa de aceleración es importante porque si se quisieran mover a mínima velocidad siempre, sí se arrancaría con carga acoplada, pero el proceso sería muy lento. Es obvio que se necesita también una etapa de desaceleración, para poder frenar el motor en el lugar deseado a pesar de que tenga mucha carga acoplada.

II.2 Generacion de opciones de solucion

El circuito electrónico que vamos a desarrollar deberá tener tres etapas principales. Estas se muestran a manera de diagrama de bloques en la *Figura II.13*.

Figura II.13 Diagrama de bloques de la interfaz

Cada uno de los bloques tiene implícitas otras necesidades. A continuación describiremos con más detalle cada uno de los bloques.

El bloque de recepción serial, que debe cumplir con el estándar RS-232C descrito anteriormente, debe incluir un generador de baudaje para producir los pulsos de trasmisión y/o recepción a la misma velocidad que la computadora. Debido a que la interfaz esperará un dato de la computadora para trasmitirle otro de regreso, este bloque deberá soportar, por lo menos, una comunicación serie semidúplex, es decir, que se pueda recibir y trasmitir pero no simultáneamente.

Por otro lado, la memoria RAM (del inglés Random Access Memory: memoria de acceso aleatorio) debe poder almacenar datos de 8 bits ya que la computadora trasmitirá con ese formato. Mientras más capacidad tenga la memoria será mejor, aunque en ningún momento es restricción, ya que en caso de ser muy pequeña lo único que sucedería es que la computadora tendría que trasmitir los datos en bloques de n datos y esperar a recibir una señal de la interfaz para enviar el siguiente bloque.

Todo el bloque debe ser controlado por un programa que estará almacenado en una memoria ROM (del inglés Read Only Memory: memoria de sólo lectura). La magnitud de esta memoria debe ser suficiente para almacenar los datos necesarios para la comunicación entre la interfaz y la computadora, así como para el programa de control de tres motores simultáneamente. Haciendo un cálculo previo necesitaríamos una memoria ROM de por lo menos 4 kbytes.

El procesamiento de datos debe realizarse, necesariamente, por medio de un microprocesador que tenga por lo menos tres puertos de salida. Por estos puertos saldrán los pulsos que controlarán a los motores, por lo que los puertos deben ser paralelos, es decir, que los pulsos salgan simultáneamente. Los puertos de salida deben ser de 4 bits, ya que los motores de pulsos que usaremos necesitan de 4 pulsos para moverse.

De acuerdo a todo lo anterior decidimos usar un microcontrolador de la familia MCS-51 de Intel, que cuenta en una sola pastilla con un puerto serial full duplex, reloj interno con el que se puede generar el baudaje, memoria RAM de 128 bytes, memoria ROM de 4 kbytes, 4 puertos de entrada/salida de 8 bits. Puede accesar hasta 64 kbytes de memoria ROM externa y 64 kbytes de memoria RAM externa. Contiene un micropocesador o CPU de 8 bits.

En la *Figura II.14*, de la siguiente página, se muestra el diagrama de bloques de esta familia de microcontroladores.

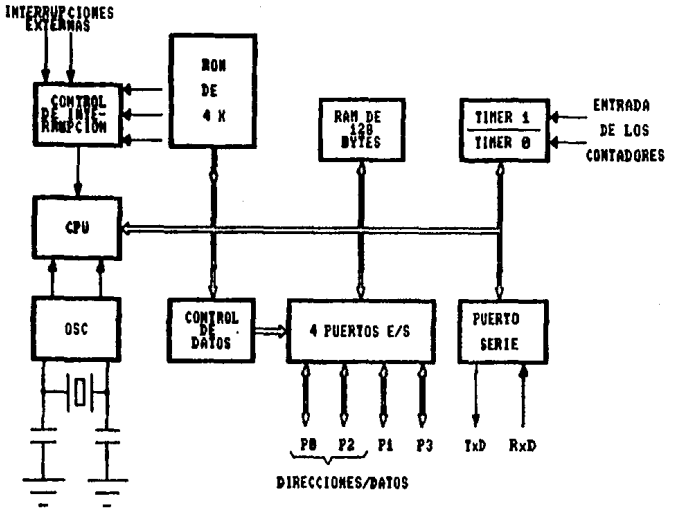


Figura II.14 Diagrama de bloques de la familia MCS-51

Dentro de esta familia existen tres versiones principales que se describen a continuación:

a) 8031: esta es la versión más sencilla de la familia. Tiene todas las características con excepción de la memoria ROM.

b) 8051: esta versión incluye todas las características pero su memoria ROM no puede ser borrada, es decir, se programa una vez y así se queda.

c) 8751: es el más completo de los tres. Además de todas las características incluye una memoria EPROM, es decir, una memoria que puede ser programada y borrada varias veces.

Dentro de la misma familia existen otros microcontroladores más complejos, con mejoras a las versiones anteriores. Como ejemplo tenemos a los integrados 8032, 8052 y 8752 que tienen las mismas características; sólo cambian en la capacidad de sus memorias, que para estos casos son de 256 bytes de memoria RAM y 8 kbytes de memoria ROM.

Tomando como base los integrados 8031, 8051 y 8751, generamos tres opciones que a continuación describiremos. En la sección II.3 las analizaremos para elegir la mejor.

II.2.1 USO DEL MICROCONTROLADOR 8031, CON EXPANSION DE PUERTOS MEDIANTE COMPUERTAS LOGICAS

Como se mencionó anteriormente, el 8031 no tiene ROM interna, por lo cual es necesario guardar el programa de control en una memoria ROM externa. Para poder accesar el programa externo deben usarse los puertos P0 y P2 como líneas de direcciones y datos respectivamente. Esto hace que tengamos sólo dos puertos disponibles. En la *Figura II.15*, de la siguiente página, podemos ver que el puerto P3 contiene funciones alternas tales como la entrada y salida del puerto serie (RxD y TxD) y las señales de escritura y lectura de la memoria externa (WR y RD). Esto implica que tendríamos que destinar el puerto P3 a trabajar con sus funciones alternas, por lo cual sólo nos queda disponible un puerto de 8 bits como salida.

Por otro lado, como se mencionó anteriormente, los motores de pulsos, o motores de pasos como son mejor conocidos, necesitan de una secuencia de pulsos para girar en uno u otro sentido. En nuestro caso, los pulsos binarios que necesitamos son 0011_B, 0110_B, 1100_B y 1001_B, que equivalen a los valores hexadecimales 03_H, 06_H, $0C_H$ y 09_H. El orden en que salgan estos pulsos no debe alterarse, es decir, después de un 03_H deberá seguir un 06_H o un 09_H para que gire en un sentido o en el otro.

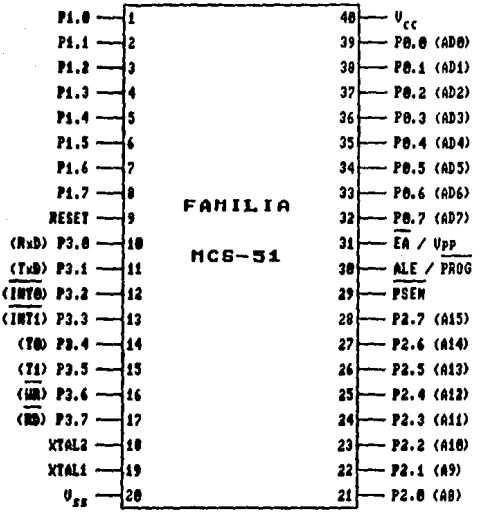


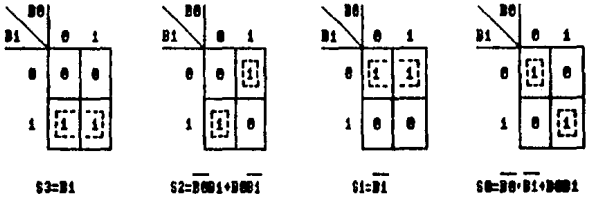
Figura II.15 Descripción de los pines de la familia MCS-51

Para obtener los tres grupos de pulsos que necesitamos a partir de un solo puerto de salida de 8 bits, se pensó en tomar dos bits para cada motor y con una decodificación de compuertas lógicas, obtener cuatro bits. La *Figura II.16* muestra los pulsos que tendríamos a la salida del puerto P1 y los pulsos que debemos obtener a la salida de la etapa decodificadora.

P0.1	P0.0	S3	S2	S1	S0
0	0	0	0	1	1
0	1	0	1	1	0
1	0	1	1	0	0
1	1	1	0	0	1

Figura II.16 Pulsos de entrada y salida de la etapa decodificadora

Para diseñar la etapa decodificadora se hicieron los mapas de Karnaugh de cada salida. Entonces, de la *Figura II.16*, cambiando PO.0 por BO y PO.1 por B1, tenemos:



La etapa decodificadora se muestra en la *Figura II.17*.

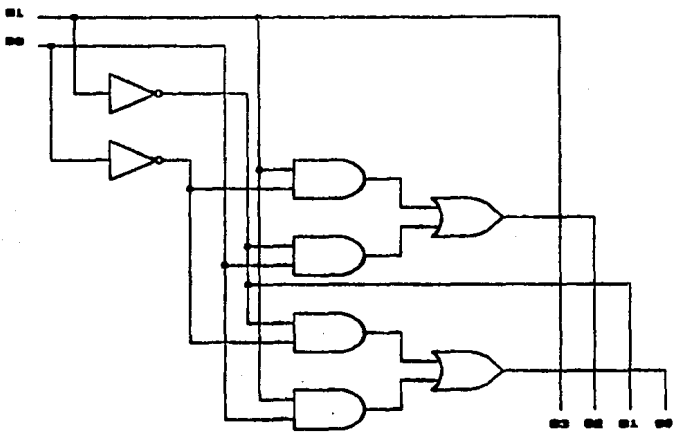


Figura II.17 Etapa decodificadora

Si conectamos esta etapa en cada par de salidas del puerto PO, en la configuración mostrada en la siguiente página (*Figura II.18*), podríamos controlar hasta cuatro motores.

Figura II.18 Configuración básica con la memoria ROM externa

II.2.2 USO DEL MICROCONTROLADOR 8031, CON EXPANSION DE PUERTOS CON EL 8155

Esta opción es muy similar a la anterior, la parte que cambia es la expansión de puertos. En este caso se usa el circuito integrado 8155 que se muestra en la *Figura II.19*. Este circuito es en realidad una memoria RAM con puertos de entrada/salida. Es un circuito periférico propio de la familia MCS-51 de microcontroladores cuyos puertos pueden ser programados como entradas y/o como salidas. Tiene 256 bytes de memoria RAM.

La conexión del circuito se muestra en la *Figura II.20*.

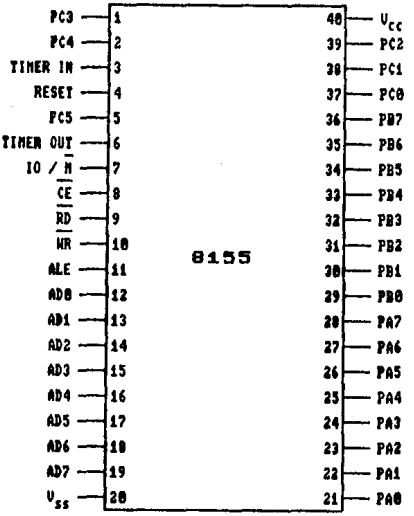


Figura II.19 Diagrama de los pines del 8155

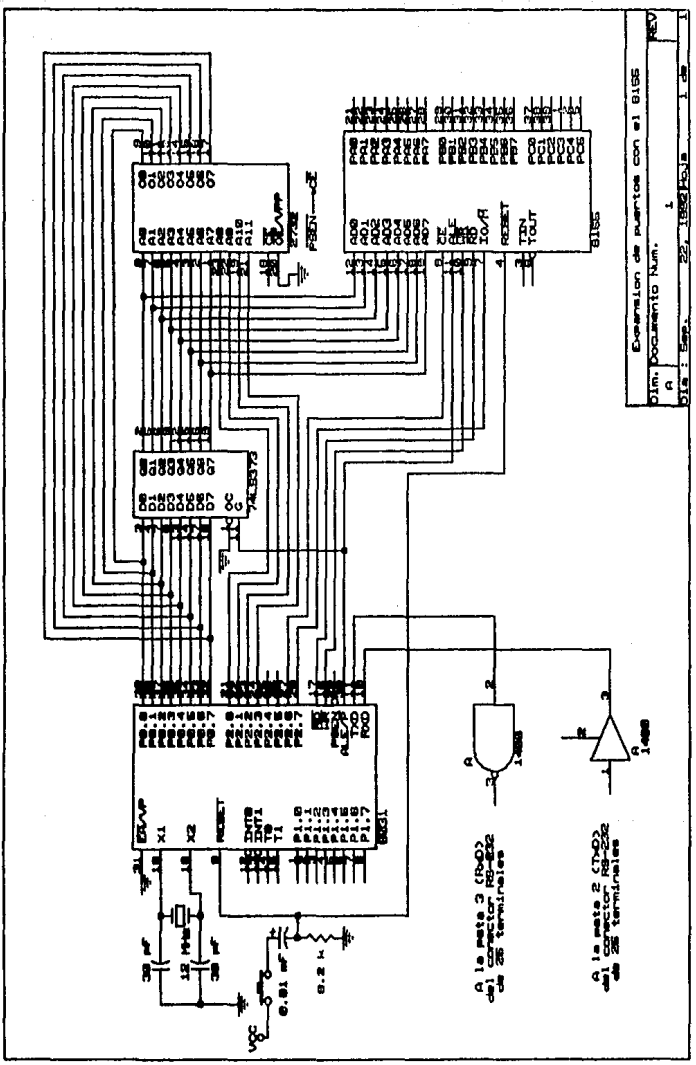


Figura II.20 Expansión de puertos con el 8155

II.2.9 USO DEL MICROCONTROLADOR 8751

Este microcontrolador ya posee memoria EPROM interna, por lo cual no es necesario una memoria externa. Esto hace que nos queden tres puertos de 8 bits disponibles como salidas.

El 8751 es exactamente igual en el orden de sus patas al 8031. La única diferencia entre ellos es la memoria EPROM del 8751.

La interconexión de los componentes para esta opción se muestra en la *Figura II.21*, en la siguiente página.

II.3 SELECCION DE LA MEJOR OPCION

Para poder tomar una determinación consideramos tres criterios de selección. El primero de ellos fue con respecto al costo de cada opción, el segundo, según al número de componentes empleados y el tercero fue de acuerdo al volumen de producción.

De acuerdo al primer criterio, eliminaríamos definitivamente la tercera opción ya que el 8751 es muy caro; sin embargo, en cuanto al espacio es el que ocupa menos, ya que sólo tendría tres circuitos integrados. La primera opción es la más barata y no ocupa mucho espacio, sin embargo, al tratar de desarrollar el programa de control nos dimos cuenta que iba a ser muy difícil conseguir el control de tres ejes. La segunda opción es de un costo intermedio y ocupa aproximadamente el mismo espacio que la primera.

Las ventajas y desventajas que tienen entre sí, son las siguientes: la segunda opción tiene la ventaja sobre las otras dos de tener 256 bytes de RAM, ya que el 8155 tiene esa capacidad de memoria, mientras que las otras sólo tienen 128 bytes de los cuales no son todos disponibles para almacenar datos, ya que

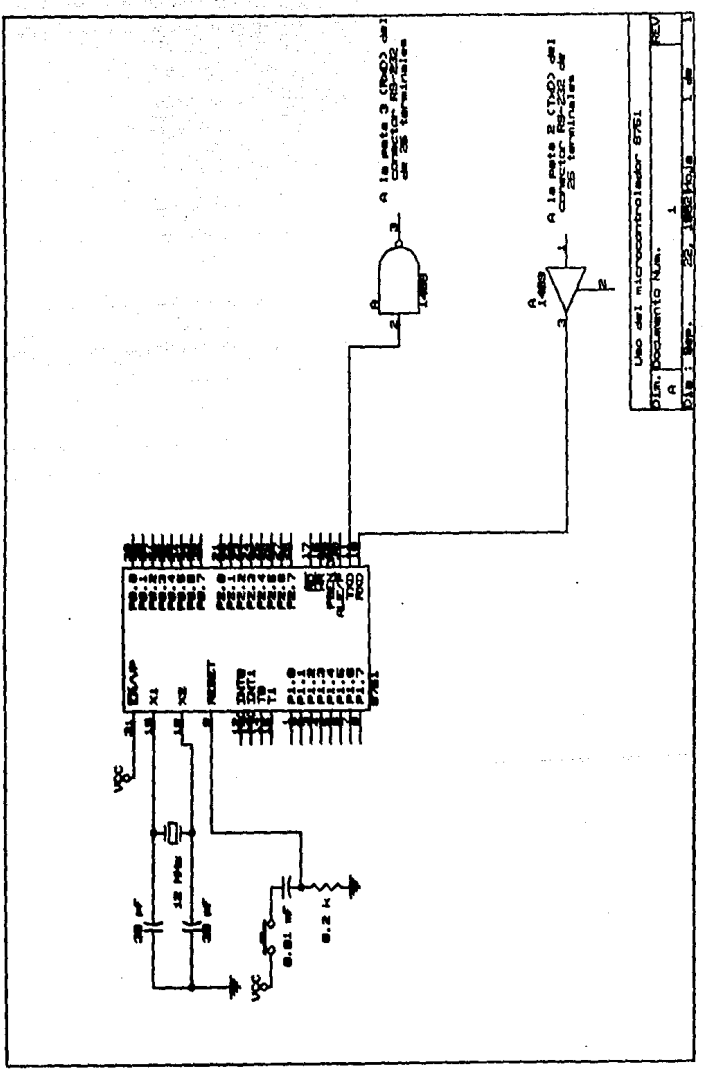


Figura 11.21 Uso del microcontrolador 8751

existen registros de funciones especiales que tienen localidades específicas dentro de la memoria RAM.

En cuanto a programa, la segunda opción tiene la desventaja de que, además de las instrucciones propias para controlar a los tres motores, deben incluirse instrucciones que programen los puertos del 8155 como salidas y, en caso de que se requiera alguna retroalimentación de los motores hacia la interfaz, habría que programarlos como puertos de entrada. Por el contrario, en la primera y tercera opción, para que el puerto sea de entrada o salida basta con leer o escribir en el puerto respectivamente.

También en cuanto a programa, como ya se dijo anteriormente, la primera opción va a ser muy complicada ya que, para poder controlar los motores, sería necesario cargar tres tablas de pulsos, lo que por una parte ocuparía más localidades de memoria ROM y por otro lado, alargaría el programa de control debido a que habría que estar haciendo sumas y restas con las tablas de los pulsos, dependiendo del sentido de giro de los motores.

Por último, la tercera opción, aunque aparentemente sea la más cara, tiene la ventaja sobre las otras dos de que su programa de control va a ser muy sencillo. La sencillez se debe a que al tener disponibles tres puertos de 8 bits, podemos ocupar 1 puerto para controlar dos motores, ya que los motores son de 4 bits. El tercer motor se controla con un puerto independiente y todavía queda un puerto de 8 bits, que puede ser empleado como entrada en caso de que se requiriera alguna retroalimentación de los motores hacia la interfaz.

Por otro lado, decimos que aparentemente es la opción más cara ya que, como prototipo, efectivamente tendrá un costo muy superior a las otras dos opciones; sin embargo, una vez que se tenga el programa final, se puede cambiar al microcontrolador 8051 y grabarlo con las instrucciones definitivas, con lo cual se tiene

una nueva opción más barata, y que ocupa menos espacio que las dos anteriores.

De esta forma, la opción que elegimos fue la tercera, ya que si se quiere producir en grandes cantidades este dispositivo, dicha opción es la que cumple con los requisitos óptimos de costo y espacio. En la sección III.1 se mostrará esta opción con su etapa de potencia incluida y los valores de los elementos usados.

TEMA 3
DISEÑO Y CONSTRUCCION

En este capítulo se diseñará la opción elegida; esto incluye los programas de control y de comunicación además de la etapa de potencia.

Entre los aspectos principales de los programas, destaca el que se refiere al cálculo del baudaje, ya que es muy importante que la computadora y la interfaz tengan exactamente la misma velocidad, tanto para trasmitir como para recibir la información.

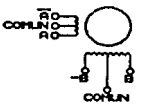
Como se emplearán motores de pasos, la etapa de potencia se desarrlló de acuerdo a las características de los mismos. Cuando se requiera mover motores más grandes, se deberá diseñar otra etapa diferente.

III.1 Circuito Electronico

Antes de mostrar el diagrama electrónico completo, con la opción elegida, diseñaremos la etapa de potencia.

Debido a que sólo necesitamos pulsos amplificados, empleremos un transistor en configuración de interruptor, es decir, el transistor trabajará únicamente en las regiones de corte (interruptor abierto) y de saturación (interruptor cerrado).

El motor de pasos tiene el siguiente símbolo:



Suponiendo que A = 12 V y A = 0 V, tenemos que Q1 está saturado (cerrado) y Q2 está en corte (abierto) por lo cual el circuito equivalente es el que se muestra en la *Figura III.1*.

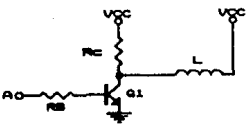
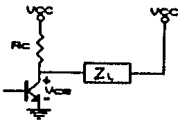


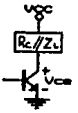
Figura III.1 Etapa de potencia

De la figura se observa que Rc queda en paralelo con la impedancia de L. Para obtener un mejor par en el motor, nos interesa que Rc >> Z_L de modo que circule la mayor parte de la corriente a través del embobinado del motor.

Escogemos Rc = 10 kΩ y de los datos de placa del motor tenemos que Z_L = 75 Ω/fase. Entonces, cuando el transistor esté saturado tendremos el siguiente circuito:



y como Rc está en paralelo con Z_L:



$R_c//Z_L = 10000x75/(10000+75)$

$R_c//Z_L = 74.44\ \Omega$

Aplicando la ley de Kirchhoff de voltáje a través de esa malla, tenemos:

$$V_{cc} - I_c(R_c//Z_L) - V_{CE} = 0$$

En la región de saturación $V_{CE} = 0$, por lo cual:

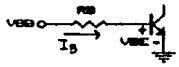
$$I_c = V_{cc}/(R_c//Z_L)$$

$$I_c = 12/74.44 = 161.2\ mA$$

Para asegurar la saturación del transistor se ocupa la regla 10:1, es decir, la corriente de base debe ser la décima parte de la corriente de colector. De acuerdo a esto:

$$I_B = I_c/10 = 16.12\ mA$$

Para analizar la malla de la base del transistor ocupamos el siguiente circuito:



donde V_{BB} es el valor de voltaje cuando el pulso A tiene un nivel alto (mínimo de 3.5 V).

Aplicando la ley de Kirchhoff de voltaje en esta malla, tenemos:

$$V_{BB} - I_B R_B - V_{BE} = 0$$

Despejando la resistencia de base tenemos:

$$R_B = (V_{BB} - V_{BE})/I_B = (3.5 - 0.7)/16.12mA$$

$$R_B = 173.69\ \Omega$$

Tomamos el valor comercial, $R_B = 150\ \Omega$.

Para que un transistor pueda usarse en esta aplicación, debe ser capaz de manejar más de 161.2 mA a través del colector. Si se escoge un transistor que maneje 200 mA, será necesario usar disipadores de calor, ya que los transistores estarán manejando siempre su máxima capacidad de corriente. El uso de disipadores influye en el espacio empleado para el circuito.

De acuerdo a lo anterior, decidimos usar el transistor TIP121, que además de su bajo costo, presenta internamente una configuración "par Darlington" como se muestra en la *Figura III.2*

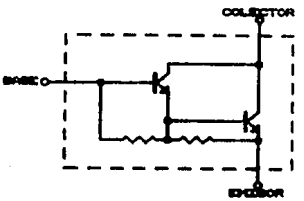


Figura III.2 Par Darlington

La ventaja de emplear un par Darlington radica en que esta configuración presenta una impedancia muy grande a la entrada y una impedancia de salida muy baja. Esta característica hace que la señal de la fuente, en este caso los pulsos generados por el microcontrolador, no se disminuyan cuando se conecta el motor. Otra ventaja de esta configuración es la ganancia de corriente (β) tan alta que se puede obtener.

Las características del TIP121 se muestran a continuación.

I_C = 5 A
V_{CE} = 80 V (saturación)
β = 1000 - 20000

El hecho de que la corriente de colector esté tan sobrada, nos da la ventaja de que no necesitamos usar disipadores de calor, ya que en esta aplicación nunca llegaremos a ese nivel de corriente. Por otro lado, nos permite conectar motores más grandes sin necesidad de cambiar la etapa de potencia.

El diagrama electrónico definitivo, incluyendo la etapa de potencia, se muestra en la *Figura III.3* de la siguiente página.

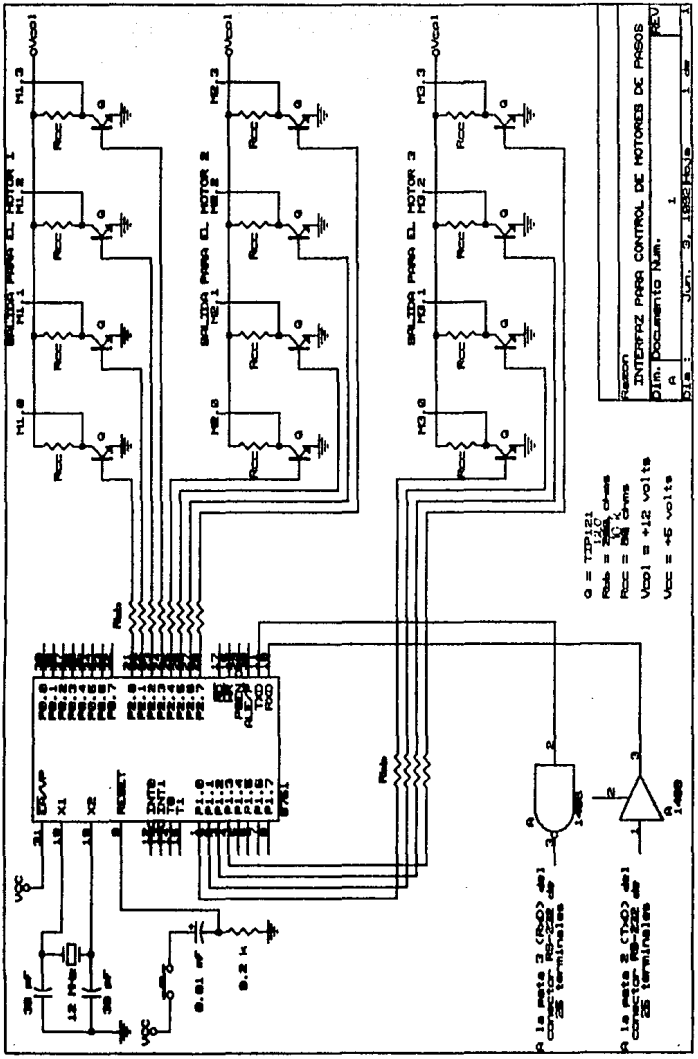


Figura III.3 Diagrama electrónico definitivo

III.2 Programa de Operacion y Control

El programa de operación y control es el que se encargará del funcionamiento de la interfaz. Este programa debe hacerse cargo de comunicar a la interfaz con la computadora, de procesar los datos recibidos y de generar los pulsos necesarios para controlar a los motores.

Consideramos que la comunicación serie es la parte más compleja del programa, por lo cual analizaremos primero todo lo referente a dicha comunicación.

III.2.1 Modos de operacion de la comunicacion serie para el 8751

El puerto serie del 8751 es "dúplex completo", lo cual significa que puede trasmitir y recibir simultáneamente. Los registros de trasmisión y recepción del puerto serie son accesados mediante el registro especial SBUF (Serial BUFfer). Al escribir en SBUF, se carga el registro de trasmisión, y al leer de SBUF se accesa físicamente a un registro separado de recepción.

El puerto serie puede trabajar en cuatro modos distintos.

Modo 0. En este modo de operación funciona como un registro de corrimiento. Los datos entran y salen a través de la pata RxD. Se trasmiten y reciben 8 bits empezando por el menos significativo. El baudaje se fija a 1/12 de la frecuencia del oscilador.

Modo 1. 10 bits son trasmitidos (a través de TxD) o recibidos (a través de RxD). Un bit de inicio (siempre bajo), 8 bits de datos empezando por el menos significativo y un bit de parada (siempre alto). El baudaje puede ser programado.

Modo 2. 11 bits se trasmiten (a través de TxD) o se reciben (a través de RxD). Un bit de inicio (siempre bajo), 8 bits de datos empezando por el menos significativo,

un 9º bit programable y un bit de parada (siempre alto). El 9º bit puede emplearse como bit de paridad. El baudaje se fija a 1/32 ó 1/64 de la frecuencia del oscilador.

Modo 3. Es igual en todos los aspectos al modo 2 excepto en el baudaje, ya que en este caso se puede programar.

En cualquiera de estos modos la trasmisión se inicia por cualquier instrucción que use al registro SBUF como un registro de destino. La recepción es iniciada, en el modo 0, mediante la condición RI = 0 y REN = 1 (Tanto RI como REN son bits direccionables del registro SCON, que se explicarán en la sección III.2.2). En los otros modos, la recepción se inicia mediante la entrada del bit de inicio, si REN = 1.

La selección del modo de operación se hace mediante programa en el registro SCON. Este, y los otros registros se explicarán a continuación.

III.2.2 REGISTROS ESPECIALES PARA LA COMUNICACION

El 8751 tiene 20 registros especiales que son empleados para funciones específicas. Estos registros son los siguientes.

PSW: condición de la palabra del programa (Program Status Word).
PCON: control de encendido (Power Control Register).
IE: habilitador de interrupciones (Interrupt Enable).
IP: prioridad de interrupciones (Interrupt Priority).
TCON: control del contador (Timer/Counter Control).
TMOD: control de modo del contador (Timer/Counter Mode).
SCON: control del puerto serie (Serial Port Control).
SBUF: comunicación serie (Serial Data Buffer).
ACC: acumulador.
B: registro especial para multiplicaciones y divisiones.
SP: apuntador (Stack Pointer).

DPTR: apuntador de datos de 2 bytes (Data Pointer 2 bytes).
DPL: byte menos significativo.
DPH: byte más significativo.
P0: puerto de entrada/salida 0.
P1: puerto de entrada/salida 1.
P2: puerto de entrada/salida 2.
P3: puerto de entrada/salida 3.
TH0: byte más significativo del contador 0.
TL0: byte menos significativo del contador 0.
TH1: byte más significativo del contador 1.
TL1: byte menos significativo del contador 1.

Once de estos registros tienen la facilidad de tener sus bits direccionables. Es decir, como todos estos registros son de 8 bits, es posible, al programar el microcontrolador, referirse al 3o bit del registro SCON, por ejemplo. Los registros que tienen esta característica son los siguientes: ACC, B, PSW, P0, P1, P2, P3, IP, IE, TCON y SCON.

También se puede notar que, aunque el 8751 es un microcontrolador de 8 bits, es posible trabajar con datos de 16 bits gracias al registro DPTR. Los contadores también pueden trabajar con datos de 16 bits.

De todos los registros anteriores, los que se emplean para la comunicación serie se explicarán a continuación con más detalle. El orden en que se describirán no es de acuerdo a su importancia, sino al orden en que fueron mencionados anteriormente.

El primer registro es el PCON. El orden de sus bits se presenta a continuación.

SMOD	---	---	---	GF1	GF0	PD	IDL

SMOD	Bit de doble velocidad de baudaje. Si el contador 1 se usa para generar el baudaje y SMOD = 1, el baudaje será duplicado cuando el puerto serie sea usado en los modos 1, 2 ó 3.
—	Reservado para usos futuros.
—	Reservado para usos futuros.
—	Reservado para usos futuros.
GF1	Bandera de uso general.
GF0	Bandera de uso general.
PD	Poner en 1 este bit activa el apagado (Power Down) automático. Sólo está disponible en la versión CHMOS.
IDL	Si IDL = 1 se activa el modo de inactividad (Idle). Sólo está disponible en la versión CHMOS.

De este registro el bit que nos interesa es el SMOD, ya que dependiendo del valor de baudaje que podamos obtener, sabremos si lo ponemos en 0 o en 1. Los bits reservados para usos futuros deben llevar un 0.

El registro IE tiene el siguiente orden en sus bits.

EA	—	ET2	ES	ET1	EX1	ET0	EX0

EA	IE.7	Si es 0, no se reconocerá ninguna interrupción. Si es 1, se habilitarán las interrupciones cuyo bit sea 1.
—	IE.6	Reservado para usos futuros.
ET2	IE.5	Interrupción del sobreflujo del contador 2.
ES	IE.4	Interrupción del puerto serie.
ET1	IE.3	Interrupción del sobreflujo del contador 1.
EX1	IE.2	Interrupción externa 1.
ET0	IE.1	Interrupción del sobreflujo del contador 0.
EX0	IE.0	Interrupción externa 0.

El bit IE.5 sólo es para el 8052. Para el 8051, los bits IE.5 e IE.6 deben llevar 0. Cuando una interrupción sea detectada el programa saltará a una localidad específica, llamada "vector de interrupción", en donde estará la rutina que deba ejecutarse de acuerdo a la interrupción que se haya activado. Estos vectores ocupan cada uno, 8 bytes de memoria ROM por lo que si alguna rutina va a emplear más localidades, deberá saltar a localidades de memoria desocupadas, a menos que no se vayan a usar los vectores de interrupción siguientes. Estos vectores se muestran a continuación.

Fuente de interrupción	Vector	Descripción
IE0	0003H	Interrupción externa 0
TF1	000BH	Sobreflujo del contador 0
IE1	0013H	Interrupción externa 1
TF1	001BH	Sobreflujo del contador 1
RI & TI	0023H	Interrupción serie
TF2 & EXF2	002BH	Sobreflujo del contador 2

La última interrupción es sólo para el 8052. Para la comunicación serie, la interrupción se da cuando se prende ya sea la bandera de recepción (RI) o la de trasmisión (TI), por lo cual la interrupción se genera mediante la operación lógica "OR" de RI y TI. La rutina de interrupción debe identificar cual de las dos banderas generó la interrupción y entonces desactivar dicha bandera.

Las interrupciones anteriores pueden presentarse en cualquier momento y no forzosamente una después de otra. Sin embargo, aunque se presenten todas a la vez, sólo será posible atenderlas una por una. Para esto existe un nivel de prioridad, que se presenta a continuación en orden de mayor a menor prioridad.

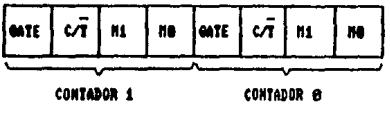
IE0
TF0
IE1
TF1
RI+TI
TF2+EXF2

Esta prioridad puede ser alterada mediante el registro IP que tiene el siguiente orden.

---	IP.7	Reservado para usos futuros
---	IP.6	Reservado para usos futuros
PT2	IP.5	Prioridad del contador 2 (Sólo para el 8052)
PS	IP.4	Prioridad del puerto serie
PT1	IP.3	Prioridad del contador 1
PX1	IP.2	Prioridad de la interrupción externa 1
PT0	IP.1	Prioridad del contador 0
PX0	IP.0	Prioridad de la interrupción externa 0

Para el 8051, los bits IP.7, IP.6 e IP.5 deben llevar un 0. Al activar cualquier bit, tendrá mayor prioridad la interrupción correspondiente y con un 0, tendrá menor prioridad. Cuando se generen dos interrupciones simultáneas, se atenderá la de mayor prioridad. Si se está ejecutando una rutina de interrupción, sólo podrá detenerse si llega una interrupción de mayor prioridad. Si la interrupción que llegue es de igual o menor prioridad, no se interrumpirá la rutina que se esté atendiendo.

El siguiente registro que se ocupa para la comunicación es el TMOD, cuyos bits se muestran a continuación.



GATE Si es 1, sólo funcionará mientras el pin INTx (x puede ser 0 ó 1 dependiendo del contador) está activado mediante un control electrónico. Si es 0, sólo funcionará mientras TRx sea 1 (TRx es del registro TCON y se controla mediante programa)

C/$\overline{T}$ Selecciona con 1, el modo contador (control con reloj

externo), y con 0 el modo base de tiempo (control con el reloj interno). Este último se conoce como Timer y así se nombrará a partir de este momento

M1, M0 Selecciona el modo de funcionamiento de acuerdo a la siguiente tabla.

M1	M0	Modo de operación	Descripción
0	0	0	Timer de 13 bits (Compatible con la familia MCS-48)
0	1	1	Timer/Contador de 16 bits
1	0	2	Autorecarga de 8 bits
1	1	3	Para el Timer 0, TL0 es un contador de 8 bits controlado por el Timer 0. TH0 es un contador de 8 bits controlado por el Timer 1. El Timer 1 se detiene en este modo.

Es necesario seleccionar el Timer 1 en modo 2, ya que para generar el baudaje se requiere esta condición.

El registro SCON, que propiamente controlará la comunicación serie, se muestra a continuación.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0 SCON.7 Selecciona el modo de operación

SM1 SCON.6 Selecciona el modo de operación

SM2 SCON.5 Habilita la intercomunicación entre procesadores disponible en los modos 2 y 3. Si es 1, RI no se activará si el 9o bit recibido (RB8) es 0. En modo 1, si SM2 = 1, RI no se activará si no se recibe un bit de parada válido. En modo 0, SM2 debe ser 0.

REN SCON.4 Se activa o desactiva por programa para habilitar o deshabilitar la recepción.

TB8 SCON.3 Es el 9o bit que se trasmitirá en los modos 2 y 3. Se activa o desactiva mediante programa.

RB8 SCON.2 Es el 9o bit que se recibirá en los modos 2 y 3. En el modo 1, si SM2 = 0, RB8 es el bit de parada que

fue recibido. En el modo 0 no se usa.

TI SCON.1 Bandera de trasmisión. Se activa mediante circuitería electrónica al final del octavo bit en el modo 0; o al principio del bit de parada en los otros modos. Se desactiva por programa.

RI SCON.0 Bandera de recepción. Se activa mediante circuitería electrónica al final del octavo bit en modo 0, o a la mitad del bit de parada en los otros modos. Debe ser desactivado por programa.

La forma en que se selecciona el modo de operación es el siguiente.

SM0	SM1	Modo	Descripción	Baudaje
0	0	0	Registro de corrimiento	Fosc/12**
0	1	1	UART de 8 bits*	Variable
1	0	2	UART de 9 bits*	Fosc/64 ó Fosc/32
1	1	3	UART de 9 bits*	Variable

*UART significa receptor/trasmisor asíncrono universal (Universal Asynchronous Receiver Transmitter).
** Fosc = frecuencia de oscilación.

El registro TH1 se emplea para generar el baudaje y se explicará en la siguiente sección.

III.2.3 CALCULO DEL BAUDAJE

El cálculo del baudaje dependerá del modo de comunicación que se seleccione. En el modo 0, que es el más sencillo, el baudaje está fijado a 1/12 de la frecuencia del oscilador. Para generar este baudaje basta con seleccionar el modo 0 en el registro SCON.

En el modo 2 el baudaje tiene 2 valores fijos y la selección depende del valor que tenga el bit SMOD del registro PCON. Si SMOD = 0, que es el valor que tiene inmediatamente después de reinicializar el sistema (RESET), el baudaje será 1/64 de la frecuencia del oscilador. Si SMOD = 1, el baudaje es 1/32 de la frecuencia del oscilador.

Para los modos 1 y 3, el baudaje está determinado por el valor de rrecarga del Timer 1. Para este caso, la interrupción del Timer 1 debe ser deshabilitada y debe configurarse para funcionar en modo de autorecarga, es decir, la parte alta del registro TMOD debe ser 0010B. De esta forma, el baudaje está dado por la siguiente fórmula:

$$\text{Baudaje} = \frac{2^{SMOD}}{32} \times \frac{\text{Frecuencia del oscilador}}{12[256-(TH1)]} \qquad \text{------- (a)}$$

Debe recordarse que SMOD puede tomar los valores de 0 ó 1, por lo cual si hacemos k = 2^SMOD, tendríamos que k puede tomar los valores de 1 ó 2 respectivamente.

La mayoría de las veces lo que se conoce es el valor del baudaje deseado, y se quiere conocer el valor de recarga para TH1. Por lo tanto, de la ecuación (a), tenemos:

$$TH1 = 256 - \frac{k(\text{Frecuencia del oscilador})}{384(\text{Baudaje})} \qquad \text{------- (b)}$$

TH1 debe ser un valor entero. Redondear TH1 a un valor entero puede no producir el baudaje deseado por lo que sería necesario escoger un cristal con otra frecuencia.

Si se quieren obtener valores de baudaje pequeños, debe dejarse la interrupción del Timer 1 habilitada, configurar al Timer 1 para funcionar como un contador de 16 bits (parte alta de TMOD = 0001B) y usar la interrupción del Timer 1 para hacer una recarga de 16 bits.

En la siguiente página se muestra una tabla (*Tabla III.1*) con valores de baudaje comunes obtenidos a partir del Timer 1.

BAUDAJE (Bd)	f_{osc} (MHz)	SMOD	TIMER 1		
			C/T	MODO	VALOR DE RECARGA
62.5 K	12	1	0	2	FFH
19.2 K	11.059	1	0	2	FDH
9.6 K	11.059	0	0	2	FDH
4.8 K	11.059	0	0	2	FAH
2.4 K	11.059	0	0	2	F4H
1.2 K	11.059	0	0	2	E8H
137.5	11.986	0	0	2	1DH
110	6	0	0	2	72H
110	12	0	0	1	FEEBH

Tabla III.1 Generación de baudajes comunes

En el desarrollo del programa de control, elegimos realizar la comunicación en el modo 1. El circuito electrónico tiene un cristal de 12 MHz y decidimos generar un baudaje de 2400 Bd. Entonces, de la fórmula (b) y haciendo k = 1 debido a que SMOD = 0, tenemos:

$$TH1 = 256 - \frac{(1)(12x10^6)}{(384)(2400)}$$

$$TH1 = 242.979$$

Como se dijo anteriormente, el valor de TH1 debe ser entero. Sin embargo, antes de cambiar de cristal redondeamos a 243, que en hexadecimal equivale a un valor de recarga de F3H. Calculamos el baudaje que se conseguía con este valor y, de la fórmula (a) obtuvimos lo siguiente:

$$Baudaje = \frac{(1)(12x10^6)}{32x12x(256-243)} = 2403.85$$

Como es una variación de menos del 1% del valor deseado (2400), dejamos F3H como valor de recarga del Timer 1. Al hacer la prueba física de la comunicación, no se tuvo ningún problema con el baudaje.

III.2.4 PRUEBAS PRELIMINARES

Las pruebas básicas que se efectuaron con la interfaz sin conectarlo a la computadora, fueron principalmente para familiarizarnos con la programación del microcontrolador.

Esto se debió a que el objetivo es la conexión con la computadora. Por esta razón, mencionaremos sólo dos pruebas, las cuales posteriormente nos sirvieron para controlar a los motores mediante los datos que trasmitiría la computadora.

La primera de estas pruebas fue el control del número de pasos que debía girar el motor. Para este caso empleamos el circuito electrónico mostrado en la *Figura II.18*. Como no se usó la comunicación serie, el número de pasos se dió desde el puerto P3. El diagrama de flujo se muestra en la *Figura III.4*, en la siguiente página.

Cuando se hicieron las pruebas con comunicación, este mismo programa fue empleado añadiéndole simplemente la etapa de comunicación.

La prueba anterior movía al motor a una velocidad mínima, y como se dijo anteriormente, para que el proceso sea más rápido se requiere acelerar al motor. El motivo de esta aceleración es que cuando el motor tenga una carga grande acoplada a su flecha no podrá moverla a máxima velocidad desde un principio, sino que deberá acelerarse hasta la velocidad máxima y después desacelerarse para poder detenerse en el punto deseado.

La siguiente prueba consistirá en generar etapas de aceleración y desaceleración tomando en cuenta las características del motor de pasos que vamos a emplear.

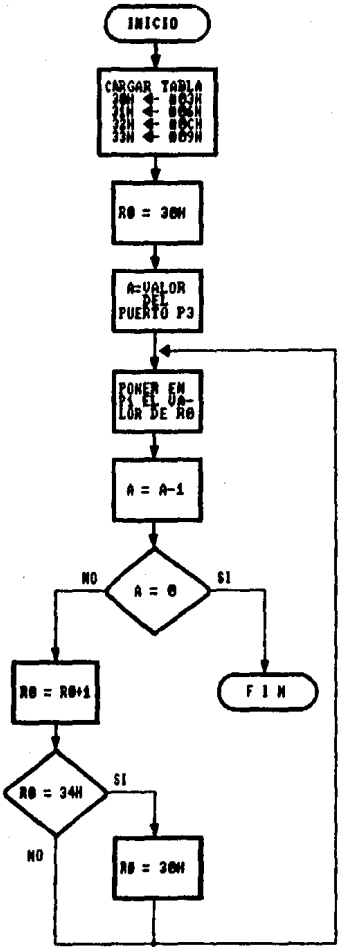


Figura III.4 Control del número de pasos desde el puerto P3

La rutina de aceleración consiste básicamente en una serie de instrucciones que al ser ejecutadas generan un tiempo de retardo que sea variable, es decir, que cada vez que se ejecute dicha rutina tarde menos tiempo en realizarse. Por el contrario, en la desaceleración la rutina cada vez durará más. En el caso de motores de pasos debe fijarse un valor mínimo de pasos para acelerar. Si el motor va a dar menos pasos que los fijados, debe hacerlo a velocidad mínima. El diagrama de flujo de la rutina de retardo, con la cual se mueve el motor a velocidad mínima, se muestra en la *Figura III.5*.

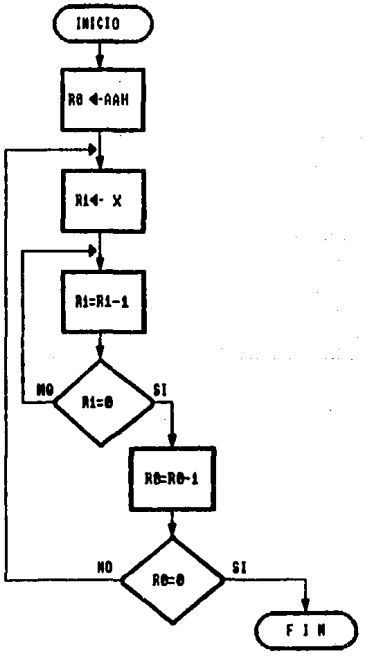


Figura III.5 Rutina de retardo

Como puede verse, nosotros fijamos el valor de R0 y debemos obtener el valor de X. Para poder calcularlo necesitamos saber cuánto tiempo tarda en ejecutarse las instrucciones que realizan esta rutina. A continuación se muestra el tiempo de ejecución de cada una de las instrucciones:

```
    MOV R0,#0AAH        12T
β:  NOP                 12T
    MOV R1,X            12T
α:  NOP                 12T
    DJNZ R1,α           24T
    DJNZ R0,β           24T
    RET                 24T
```

donde T = 1/12 MHz

El tiempo total de ejecución de esta rutina, debe ser igual a la duración del pulso que el motor requiere para moverse a mínima velocidad. En nuestro caso, fijamos la frecuencia mínima a 8 Hz, es decir, el pulso más ancho es de

$$1/8Hz = 0.125\ s$$

El valor de X debe obtenerse de la siguiente ecuación:

$$12T + 12T + 12T + 12T + (24T + 12T)X + (24T + 12T + 12T + 12T + (24T + 12T)X)AA_H + 24T = 0.125\ s$$

Convirtiendo el valor hexadecimal de AA_H a su valor decimal y reduciendo la expresión anterior, tenemos:

$$(6156T)(X) + 10272T = 0.125\ s$$

de donde

$$X = \frac{0.125 - 10272T}{6156T}$$

$$X = 241.9_D$$

Convirtiéndolo a hexadecimal tenemos

$$X = F2_H$$

De esta forma, si se carga el número F2H en la rutina de retardo el motor girará a velocidad mínima. Como se dijo anteriormente, para que se tenga aceleración la rutina debe variar hasta llegar a la velocidad de crucero del motor, que en este caso corresponde a una frecuencia de 70 Hz, es decir, el ancho del pulso es de

$$1/70 \text{ Hz} = 14.28 \text{ ms}$$

por lo tanto, la diferencia entre los anchos de pulso de la velocidad máxima y mínima es

$$125 \text{ ms} - 14.28 \text{ ms} = 110.72 \text{ ms}$$

El número de pasos mínimo que debe dar el motor para entrar a la rutina de aceleración lo fijamos en 90 pasos; que corresponden a 45 de aceleración y 45 de desaceleración. Por lo tanto, cada pulso de aceleración o desaceleración debe variar con respecto al anterior en

$$110.72 \text{ ms}/45 = 2.46 \text{ ms}$$

El siguiente valor de recarga corresponde, entonces, a un tiempo de

$$125 \text{ ms} - 2.46 \text{ ms} = 122.54 \text{ ms}$$

Por lo tanto, el valor de recarga es

$$X = \frac{0.12254 - 10272T}{6156T}$$

$$X = 237.2_D = ED_H$$

A partir de estos dos valores obtenemos el valor constante que debemos restar o sumar, dependiendo si es aceleración o desaceleración respectivamente, para obtener la diferencia de 2.46 ms entre cada pulso.

Este valor constante es

$$F2_H - ED_H = 5_H$$

De todo lo anterior podemos elaborar el diagrama de flujo definitivo de la aceleración, que se muestra en la *Figura III.6.*

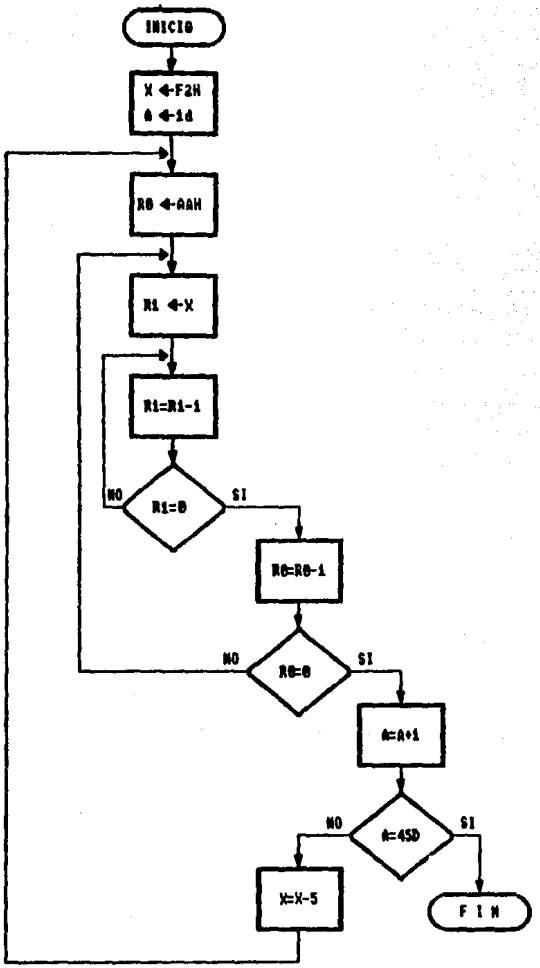


Figura III.6 Rutina de aceleración

La rutina de desaceleración será exactamente igual excepto que X inicia con un valor de 11H y en vez de restarle, se le suman 5H cada vez hasta llegar a F2H.

La forma en que probamos esta etapa fue fijando un número de pasos y, combinando los diagramas de flujo anteriores, hacer que el motor se acelerara, girara a velocidad máxima y desacelerara.

El programa de control definitivo se mostrará en la sección III.3.2.3, en donde se mencionan las pruebas de interconexión de la computadora con la interfaz.

III.3 Programa de Comunicacion de la Computadora con el Sistema Electronico

Debido a los objetivos propios de este trabajo, el programa de comunicación debe ser capaz de recibir y trasmitir datos en forma serie, desde y hacia la interfaz, a través del estándar de comunicación RS-232C.

El programa de comunicación se elaboró con el lenguaje BASIC (GW-BASIC), y requiere que el sistema operativo sea el MS-DOS versión 3.2 o posterior.

Usando este lenguaje se puede establecer una comunicación serie asíncrona, es decir, la trasmisión y la recepción de datos no depende de ninguna señal de tiempo, por lo que pueden existir en cualquier momento. Otra característica importante es que la comunicación puede establecerse como "dúplex completo" (full dúplex), lo que significa que tanto la trasmisión como la recepción de datos, de parte de la computadora y de la interfaz, pueden llevarse a cabo al mismo tiempo, como sucede en el caso de la voz en una conversación telefónica.

Antes de comenzar la comunicación, sin importar en qué lenguaje se hizo el programa, debe inicializarse el puerto serie de la computadora. Esto se hace con la orden del MS-DOS llamada MODE, que tiene el siguiente formato:

MODE COMn(:)*baud*(,*paridad*(,*datos*(,*parada*(,*p*))))

En donde:

n es el número del adaptador de comunicaciones, 1 ó 2

baud es la velocidad de trasmisión; 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600 ó 19200 Bd. Sólo se ponen los dos primeros dígitos

paridad puede ser N (ninguna), O (impar), E (par). El valor por omisión es E

datos es el número de bits de datos, 7 u 8. El valor por omisión es 7

parada es 1 ó 2. Si *baud* es 110, el valor por omisión es 2. En los demás casos es 1

p evita errores por límite de tiempo (device timeout). Se usa principalmente cuando se conecta una impresora en el puerto serie.

Una vez hecho esto, la computadora estará lista para efectuar una comunicación serie.

III.3.1 BASIC PARA COMUNICACIONES

La trasmisión y recepción de datos mediante el BASIC, se lleva a cabo de la misma manera en que se accesa a un archivo de disco. De hecho debe inicializarse un archivo de comunicaciones que activará una memoria intermedia de trasmisión y otra de recepción. En estas memorias se almacenan los datos que se trasmitirán, y los datos que se han ido recibiendo para leerlos en determinado momento.

El espacio reservado para la memoria intermedia de recepción puede alterarse al cargar el BASIC con el siguiente formato:

BASICA /C:n

En donde n es el número de datos (bytes) que se podrán almacenar en la memoria intermedia de recepción. Puede tomar un valor máximo de 32767, el valor por omisión es 256 y si se hace n = 0, se desactiva el puerto serie. Para líneas de alta velocidad (mayores a 1200 Bd), se recomienda un valor mínimo de 1024.

Para la memoria intermedia de trasmisión se tiene siempre un valor de 128 bytes.

A continuación se describirán las instrucciones del BASIC, necesarias para la comunicación serie. La primera de ellas es la que abre el archivo de comunicaciones y tiene el siguiente formato:

OPEN "COM(n):(*baud*)(,*paridad*)(,*datos*)(,*parada*)(,RS)(,CS(n))(,DS(n))(,CD(n))(,LF)(,PE)" AS (#) *archivo*

En donde:

n es el número del adaptador de comunicaciones, 1 ó 2

baud es la velocidad de trasmisión/recepción, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600 ó 19200 Bd. El valor por omisión es 300

paridad puede ser S (espacio), M (marca), O (impar), E (par) o N (ninguna). El valor por omisión es E

datos es el número de bits de datos. Los valores válidos son 4, 5, 6, 7 y 8. El valor por omisión es 7. Para datos numéricos debe ser 8

parada es 1 ó 2. Si *baud* es 75 ó 110, el valor por omisión es 2. En los demás casos es 1

archivo es el número de archivo al cual se hará referencia para trasmitir o recibir datos.

Las opciones especiales funcionan de la siguiente forma:

RS suprime la señal RTS

CS(n) controla la señal CTS

DS(n) controla la señal DSR

CD(n) controla la señal CD
LF envía un avance de línea en cada retorno de carro
PE habilita la comprobación de paridad.

En estas opciones, n es el número de milisegundos que se espera para recibir la señal antes de que ocurra un error por límite de tiempo (device timeout). Puede tomar valores de 0 a 65535, el valor por omisión es 1000 y si n = 0, no se verifica el estado de la señal.

El parámetro LF sirve para aplicaciones en las que se use una impresora serie. La opción PE activa la comprobación de paridad, ya que por defecto no está activada. Los errores de paridad producen errores del dispositivo de entrada/salida (device I/O error).

Para habilitar la recepción de datos se ocupa la instrucción COM(n), donde n es el número del adaptador de comunicaciones (1 ó 2). Esta instrucción tiene tres opciones que son: ON, OFF y STOP. Cuando es ON, se permite la intercepción de los datos recibidos al encontrar la instrucción ON COM(n) GOSUB, que se explicará más adelante.

Si está en OFF, la intercepción no se ejecuta y tampoco se tienen en cuenta las actividades de comunicaciones, aunque sí se lleven a cabo. Si se escoge la opción STOP no se realiza la intercepción, aunque sí se tiene en cuenta cualquier actividad de comunicaciones y se ejecuta una intercepción inmediata cuando el programa encuentra un COM(n) ON.

Después de habilitar la recepción, el programa salta a una subrutina en la que se leerán los datos existentes en la memoria intermedia de recepción. Este salto se hace mediante la siguiente instrucción:

ON COM(*n*) GOSUB *línea*

en donde n es el número del adaptador de comunicaciones (1 ó 2) y

línea es el número de línea en donde empieza la subrutina de lectura.

Como se dijo antes, es necesario que se ejecute un COM(n) ON antes de esta instrucción, con lo que el BASIC comprobará, cada vez que se ejecute una instrucción, si ha entrado un caracter al adaptador de comunicaciones especificado. Si es así, se salta a la subrutina de lectura.

Al final de la subrutina debe aparecer la instrucción RETURN, la cual hace automáticamente un COM(n) ON, a menos que se haya especificado otra cosa dentro de la subrutina.

Al llegar a la subrutina a veces es necesario saber cuántos caracteres hay en la memoria intermedia de recepción, ya que en algunas aplicaciones es conveniente esperar cierto número de datos para después procesarlos. Para conocer este número se ocupa la instrucción LOC(*archivo*), en donde *archivo* debe ser el mismo número con el que se abrió el archivo de comunicaciones.

Esta instrucción se usa con IF . . . THEN, en caso de querer condicionar el número de caracteres para hacer una u otra cosa, y junto con la instrucción INPUT$ como se explicará a continuación.

Mediante la instrucción INPUT$ se lee el contenido de la memoria intermedia de recepción. Esta instrucción tiene el siguiente formato:

INPUT$(x(,(#)*archivo*))

En donde x es el número de caracteres a leer y *archivo* es el número con que el que se abrió el archivo de comunicaciones.

Si se quiere leer todo el contenido de la memoria intermedia debe hacerse x = LOC(*archivo*), quedando la instrucción de la siguiente forma:

INPUT$(LOC(*archivo*)(,(#)*archivo*))

Para leer el contenido de la memoria también se pueden ocupar las instrucciones INPUT# y LINE INPUT#; sin embargo, para los archivos de comunicaciones se prefiere la instrucción INPUT$, ya que con ella se pueden leer todos los caracteres ASCII, mientras que con las otras dos se filtran los datos de entrada porque reconocen como delimitadores a algunos caracteres, es decir, ciertos caracteres no son leídos.

La trasmisión de datos serie es más sencilla que la recepción, puesto que no requiere de tantas instrucciones. Para trasmitir un dato mediante el BASIC, es necesario simplemente abrir el archivo de comunicaciones y enviar el dato mediante la siguiente instrucción:

PRINT #*archivo*,*dato*

En donde *archivo* es el mismo número con el que se abrió el archivo de comunicaciones y dato es el valor a trasmitir.

III.3.2 PRUEBAS PRELIMINARES

Después de haber visto las instrucciones necesarias para una comunicación serie, se hicieron algunas pruebas de trasmisión y recepción que a continuación se ennumeran.

III.3.2.1 Trasmisión

Como se dijo anteriormente, la trasmisión es más sencilla que la recepción, por lo que se intentó primero sacar algún dato por el puerto serie de la computadora.

Para estas pruebas se conectó la computadora en minimódem (configuración explicada en la sección II.1.2), y el nivel de voltaje de RS-232C a TTL se convirtió con el circuito de la *Figura III.7*, mostrado en la siguiente página.

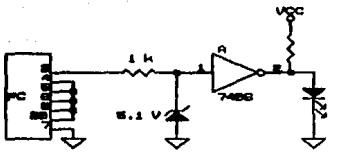
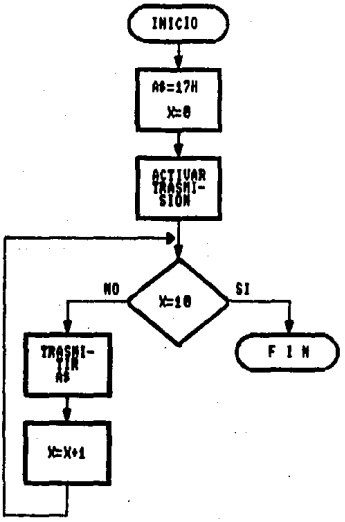


Figura III.7 Conversión de voltajes RS-232C a TTL

El diagrama de flujo y el programa BASIC para esta aplicación, se muestran en la *Figura III.8*.




```
5 A$=HEX$(23)
10 OPEN "COM1:110,N,8,1" AS #3
20 FOR X=1 TO 10
30 PRINT #3,A$
40 NEXT X
50 END
```

Figura III.8 Programa para la prueba de trasmistón

El mayor problema que se tuvo en esta aplicación fue que el circuito integrado 7406, que es una compuerta inversora con salida colector abierto, debe llevar una resistencia de levantamiento (pull up) a la salida y no se había puesto, por lo que el LED no parpadeaba, dando la impresión de que la computadora no trasmitía. Al ponerle esta resistencia el circuito funcionó inmediatamente.

III.3.2.2 Recepción

Para verificar que la computadora recibiera datos, se hizo la conexión que aparece en la *Figura III.9*, en la cual se ve que la computadora leería el mismo dato que ella trasmitiera. Nótese que la pata 8 no está conectada debido a que en las pruebas de trasmisión se notó que no era necesaria.

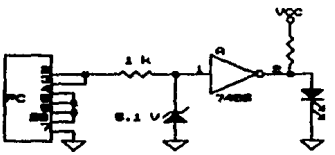
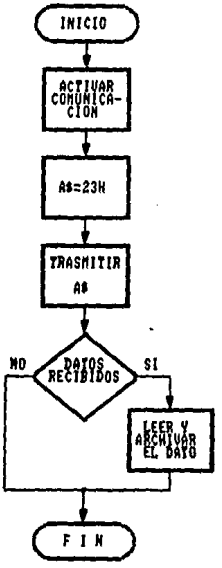


Figura III.9 Conexión para las pruebas de recepción

El primer programa que se intentó correr, junto con su diagrama de flujo, se muestran en la *Figura III.10* en la siguiente página. Este primer programa no funcionó. El problema que tenía era que nunca saltaba a la subrutina de recepción debido a que no se habilitaba la recepción de datos por la falta de la instrucción COM(n) ON. Se agregó esta instrucción en la línea 45, con n = 1, pero el programa siguió sin trabajar.




```
10 OPEN "O",#2,"ENTRADA"
20 OPEN "COM1:110,N,8,1" AS #3
30 A$=HEX$(23)
40 PRINT #3,A$
50 ON COM(1) GOSUB 70
60 END
70 S$=INPUT$(LOC(3),#3)
80 PRINT #2,S$
90 RETURN
```

Figura III.10 Programa para las pruebas de recepción

Al analizar las señales del protocolo se observó que la señal CD (carrier detect, detectora de portadora), que se había desconectado por no hacer falta en la trasmisión, era la encargada de habilitar la recepción. Se conectó esta pata (número 8 del conector de 25 terminales) en la configuración minimódem y se logró que el programa saltara a la subrutina de recepción. Sin embargo, el archivo ENTRADA, donde se almacenarían los datos recibidos, no tenía nada.

El problema que se presentaba ahora era que el programa que se tenía no daba tiempo a que se almacenara el dato en la memoria intermedia. Se hizo entonces un nuevo programa en el cual siempre se preguntara si había datos, y al entrar a la subrutina no saliera hasta que se terminaran de leer los datos de la memoria intermedia de recepción.

El diagrama de flujo y el programa de esta nueva versión para la recepción, se muestran en la siguiente página en la *Figura III.11*.

También en la siguiente página, en la *Figura III.12*, se muestra el programa con el que se leyó el contenido del archivo ENTRADA, que fue en donde se almacenaron los datos recibidos. Al leer este archivo, se pudo asegurar que la computadora ya trasmitía y recibía correctamente.

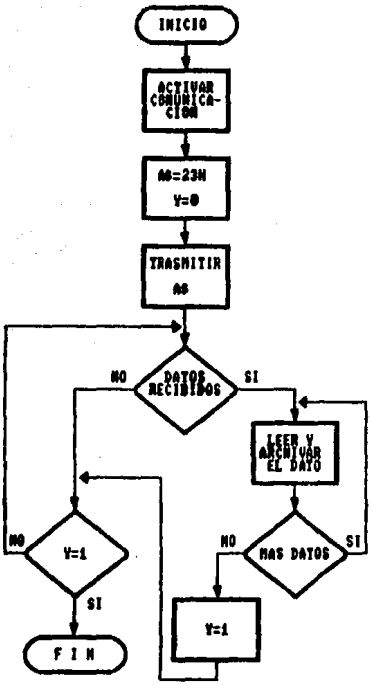
III.3.2.3 Conexión con el 8751

Ahora que ya conocemos la forma de trasmitir y recibir un número con la computadora, podemos iniciar las pruebas de comunicación entre la computadora y la interfaz.

Estas pruebas fueron en orden ascendente de dificultad hasta conseguir que ambas partes trasmitieran y recibieran. Es importante aclarar que de acuerdo a las características del proyecto, la comuniciación debe ser semi-dúplex, ya que la interfaz esperará siempre un valor de la computadora para ejecutar algo, y la computadora esperará a recibir una señal que le indique que la interfaz ya terminó de procesar los datos anteriores, para enviarle más datos.

Para todas estas pruebas el baudaje se fijó a 2400 Bd.

La primera prueba que se realizó fue de trasmisión de la interfaz hacia la computadora. En este caso, la interfaz debía enviar 10




```
5 CLS
10 OPEN "O",#2,"ENTRADA"
20 OPEN "COM1:110,N,8,1" AS #3
30 A$=HEX$(23): Y=0
40 PRINT #3,A$
45 COM(1) ON
50 ON COM(1) GOSUB 70
60 IF Y=1 THEN 65 ELSE 50
65 CLOSE: END
70 S$=INPUT$(LOC(3),#3)
80 PRINT #2,S$
85 IF EOF(3) THEN 90 ELSE 70
90 Y=1
100 RETURN
```

Figura II.11 Programa y diagrama de flujo de recepción

```
10 OPEN "I",#2,"ENTRADA"
20 INPUT #2,P$
30 PRINT P$,
40 IF EOF(2) THEN END
50 GOTO 20
```

Figura III.12 Programa de lectura de datos recibidos

números conocidos a la computadora y ésta los desplegaría en la pantalla. El circuito empleado para esta prueba fue el que se mostró en la *Figura II.18*.

Los diagramas de flujo del programa BASIC y del programa de control del microcontrolador, se muestran en la *Figura III.13* en la siguiente página.

Los problemas que se tuvieron en esta prueba se debieron a la circuitería electrónica y fueron los siguientes. En primer lugar, el circuito integrado MC1489 estaba mal conectado. Un error más considerable aún fue detectado al ver en el osciloscopio las señales RxD y TxD (patas 10 y 11 respectivamente) del 8051. Resultó que la señal TxD salía con mucho ruido, por lo que se dedujo que el microcontrolador estaba mal. Se cambió por otro y la comunicación funcionó correctamente.

Sin embargo, aunque ya había comunicación, no siempre se leían todos los datos. Esto ocurría porque al encender la interfaz se trasmitían los 10 valores inmediatamente y entonces, si el programa BASIC no había empezado a correr, algunos valores no eran detectados. Por otro lado, si se corría el programa BASIC sin encender la interfaz, se generaba el error por límite de tiempo (device timeout) ya que no se generaban las señales de protocolo. La única forma en que se veía que sí funcionaban ambos programas era corriéndolos al mismo tiempo, es decir, justo en el momento en que se corría el programa BASIC se encendía la interfaz, y entonces sí se recibían todos los valores.

Lo que aparecía en la pantalla de la computadora no eran los valores hexadecimales que se trasmitían, sino los caracteres del código ASCII correspondientes a dichos valores.

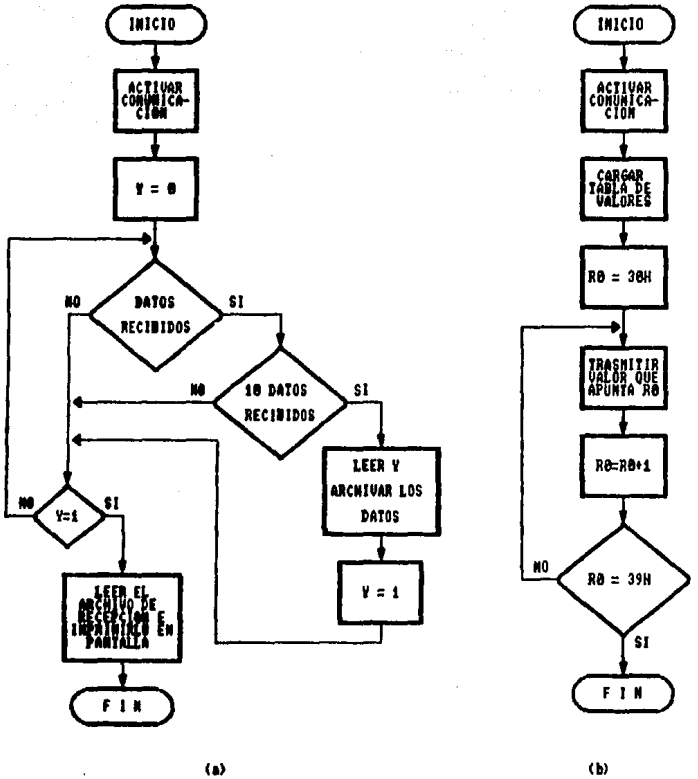


Figura III.13 Diagramas de flujo de los programas de
a) BASIC para trasmisión
b) control para recepción

De esta prueba inicial nos dimos cuenta de que era necesario que la computadora iniciara la comunicación para evitar la pérdida de los datos. Por esta razón, la siguiente prueba consistiría en trasmitir un número desde la computadora y visualizarlo, por el puerto P1 del microcontrolador, mediante LEDS.

El circuito electrónico que se empleó fue el que se mostró en la *Figura II.18*.

El diagrama de flujo del programa BASIC de trasmisión se muestra en la *Figura III.14*, y el del programa de control, se muestra en la *Figura III.15* en la siguiente página.

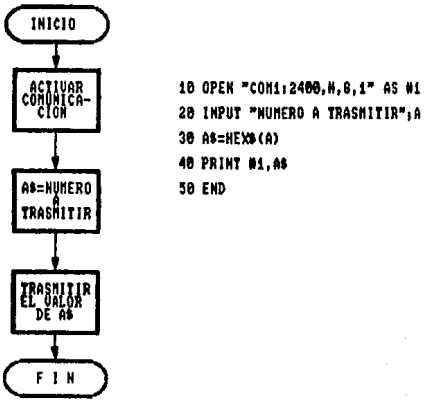


Figura III.14 Diagrama de flujo de trasmisión

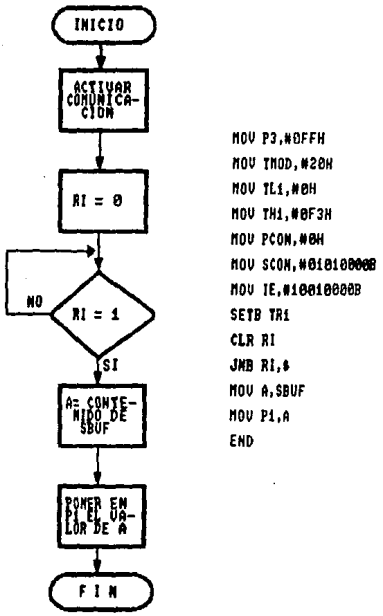


Figura III.15 Diagrama de flujo de recepción

El principal problema que se tuvo en esta aplicación fue que el valor que aparecía en los LEDS no correspondía al valor hexadecimal del número que supuestamente se había enviado. Como al parecer sí había comunicación, el problema debía ser algún detalle en uno o ambos programas. Revisamos primero el programa de control y encontramos que al habilitar la interrupción de la comunicación serie con la instrucción MOV IE,#10010000B, estaba obligándose al microcontrolador a saltar a la localidad 0023H en el momento que recibiera un valor. Como no se tenía grabada ninguna instrucción ahí, el programa se perdía. Cambiamos esta línea por MOV IE,#0H con lo cual deshabilitábamos toda

interrupción, ya que de todas formas nuestro programa debía recibir debido a que esperábamos a que se activara la bandera de recepción con la instrucción JNB RI,$.

Probamos nuevamente la comunicación pero siguió sin funcionar adecuadamente. Revisamos entonces el programa BASIC y después de consultar algunos libros, vimos que la computadora trasmite en código ASCII, por lo cual era necesario usar la instrucción

PRINT CHR$(x)

donde x es el número decimal que se quiere trasmitir. Con esta instrucción se envía directamente el valor hexadecimal de x.

El siguiente paso era lograr una comunicación semi-dúplex, es decir, la computadora envía un número y la interfaz le regresa otro. Para este caso, los programas funcionarían de tal forma, que el valor que recibiera la interfaz se visualizaría en el puerto P1 mediante LEDS, y después se trasmitiría un valor conocido hacia la computadora que se vería en la pantalla.

Inicialmente, la parte de recepción del programa BASIC fue la misma que se usó en la prueba de recepción de la computadora. Sin embargo, no funcionó adecuadamente debido a que la instrucción ON COM(1) GOSUB no es recomendable cuando se requiere recibir un solo dato. Por esta razón se empleó otro método que consiste en preguntar si hay datos en el archivo de comunicaciones, y seguir preguntando hasta que entre algún caracter. El diagrama de flujo de este programa se muestra en la *Figura III.16 a)*.

Por otra parte, el programa de control se obtuvo uniendo los dos programas anteriores. Su diagrama de flujo se muestra en la *Figura III.16 b)*.

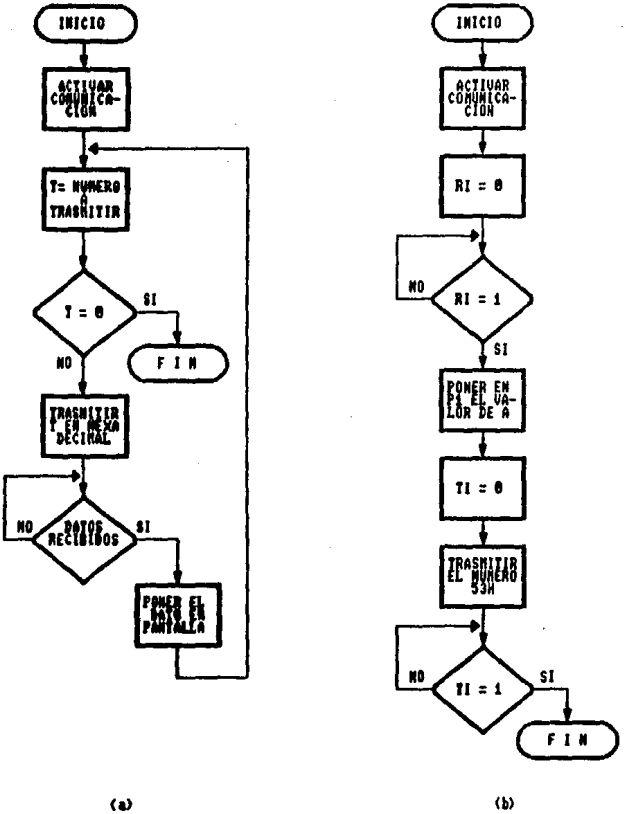


Figura III.16 Diagramas de flujo de los programas
a) BASIC de trasmisión/recepción
b) de control de recepción/trasmisión

Ambos programas funcionaron bien, por lo tanto la siguiente prueba sería el procesamiento de los datos. La primera prueba de este tipo fue el control del número de pasos de un motor, siempre en un mismo sentido. Para esta prueba, el programa de control fue el mismo que se empleó cuando no se tenía comunicación serie, simplemente se le agregó la etapa de comunicación.

El programa BASIC empleado fue el mismo que se usó en la trasmisión de datos.

Esta prueba ya no tuvo contratiempos por lo que el siguiente paso sería la realización de los programas definitivos.

III.3.2 PROGRAMAS DEFINITIVOS

Como se mencionó anteriormente deben elaborarse dos programas, uno para la computadora y otro para la interfaz. Ambos programas contarán con una etapa de comunicación. Para poder desarrollar esta etapa de comunicación debemos especificar ciertas señales de protocolo para que la trasmisión de datos sea eficiente.

Estas señales de protocolo se muestran en la *Tabla III.2*, de la siguiente página.

VALOR DECIMAL	VALOR HEXADECIMAL	TRANSMITIDO/RECIBIDO POR LA INTERFAZ	SIGNIFICADO
19	13	Recibido	Van a trasmitirse datos desde la computadora
76	4C	Trasmitido	Interfaz lista para recibir datos
81	81	Recibido	Datos del motor 1
82	82	Recibido	Datos del motor 2
83	83	Recibido	Datos del motor 3
83	53	Trasmitido	Tabla de memorización seleccionada. Lo envía después de uno de los tres datos anteriores
83	83	Recibido	Sentido de giro "horario"
84	84	Recibido	Sentido de giro "antihorario"
85	85	Recibido	El motor no gira
PH	PH	Recibido	Byte más significativo del número de pasos
PL	PL	Recibido	Byte menos significativo del número de pasos
77	4D	Trasmitido	Dato memorizado. Lo envía cada vez que recibe uno de los cinco datos anteriores
69	45	Recibido	Indica final de paquete. Empieza la ejecución
70	46	Trasmitido	Terminó de ejecutar los datos del paquete. Espera nuevo paquete o señal de fin de datos
84	54	Recibido	Fin de datos
65	41	Trasmitido	F I N D E E J E C U C I Ó N

Tabla III.2 Señales de protocolo

PROGRAMA DE LA COMPUTADORA

Elaboraremos primeramente el programa de la computadora (desarrollado en lenguaje BASIC).

Este programa debe considerar principalmente la memoria RAM tan reducida de la interfaz y la gran cantidad de datos que puede tener un proceso de control numérico. Debido a esto, enviaremos

los datos desde la computadora en paquetes. Cada paquete se compondrá de 15 bloques que tendrán el siguiente formato:

M G PH PL

donde:

M es el número de motor (1, 2 ó 3)
G es el sentido de giro (3 horario, 4 antihorario, 5 paro)
PH es la parte alta del número de pasos
PL es la parte baja del número de pasos

Debe notarse que, a diferencia de las pruebas preliminares, ahora el número de pasos ocupa dos localidades. Esto implica que los motores podrán girar más de 255 pasos ya que se manejan datos de 16 bits (2 bytes).

Cada paquete está compuesto por 15 bloques como máximo, que corresponden a 60 datos máximo. No es obligatorio que se trasmitan 60 datos, ya que si algún motor no gira (G = 05H), ese bloque tendrá sólo 2 datos.

Antes y después de enviar el bloque correspondiente al motor 3, debe trasmitirse un par de datos que controlarán un tiempo de espera entre el movimiento de los dos primeros motores y el motor 3. El primer dato que debe mandarse es un "0", con el cual la interfaz sabrá que se le enviará un tiempo. El otro dato es un valor entre 0 y 255, que corresponderá al número de segundos que habrá antes o después de que se mueva el motor 3, según sea el caso. De acuerdo a lo anterior, podemos decir que los datos se enviarán en el siguiente orden:

M1 - M2 - T1 - M3 - T2

donde Mn corresponde al paquete de datos del motor n
T1 es el tiempo de espera antes del movimiento de M3
T2 es el tiempo de espera después del movimiento de M3

El diagrama de flujo del programa BASIC definitivo se muestra en las siguientes páginas, en la *Figura III.17*.

Diagrama de flujo definitivo del programa BASIC de trasmision de datos

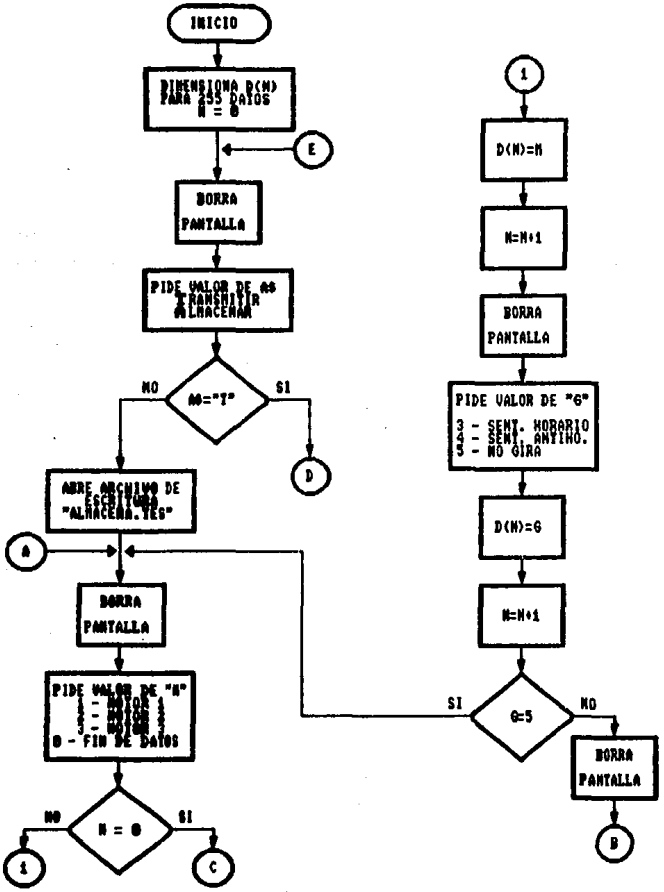


Figura III.17 Diagrama de flujo del programa BASIC

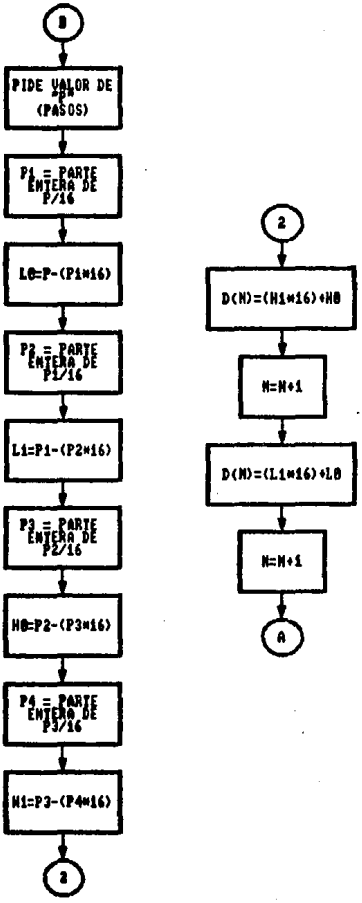


Figura III.17 Diagrama de flujo del programa BASIC (Cont.)

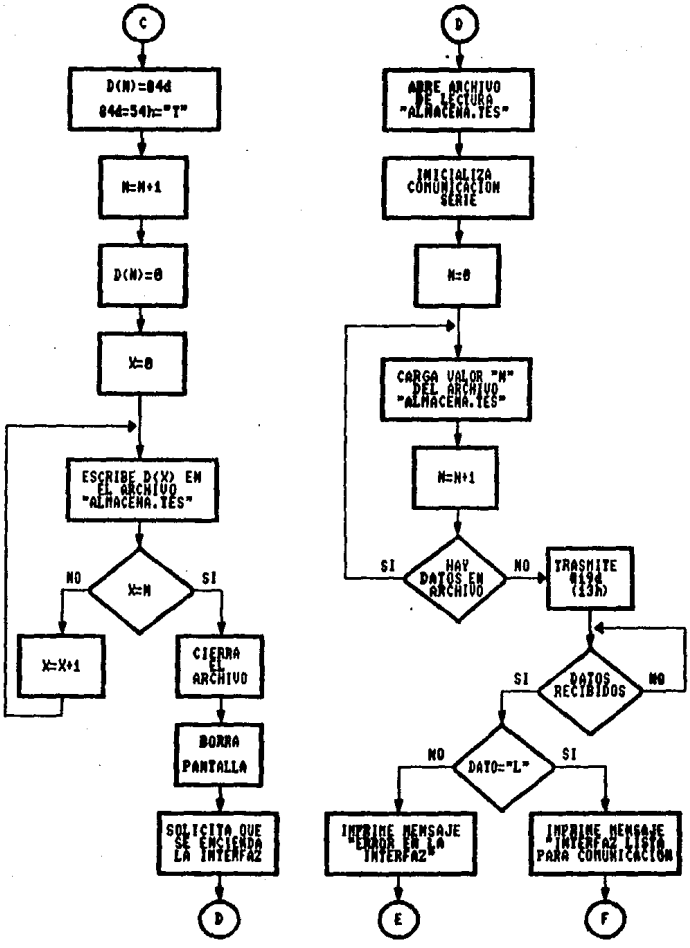


Figura III.17 Diagrama de flujo del programa BASIC (Cont.)

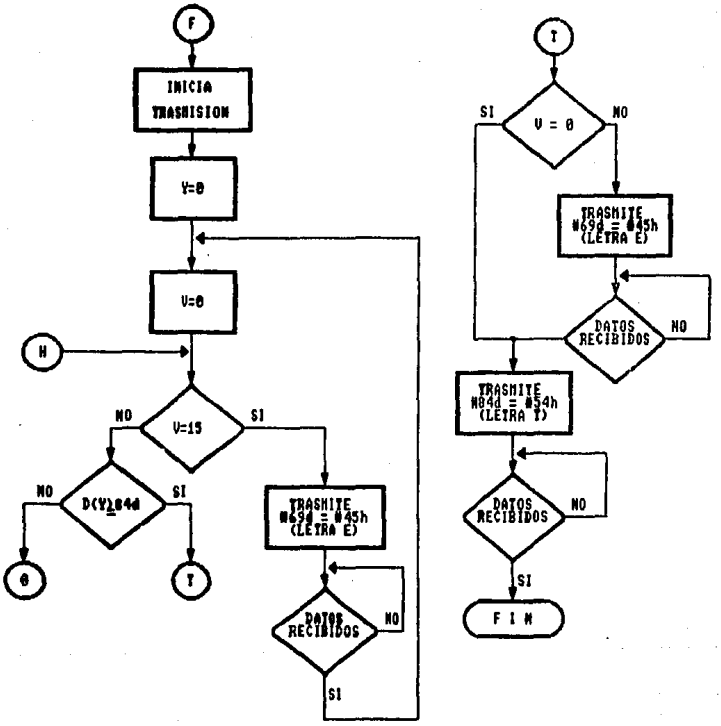


Figura III.17 Diagrama de flujo del programa BASIC (Cont.)

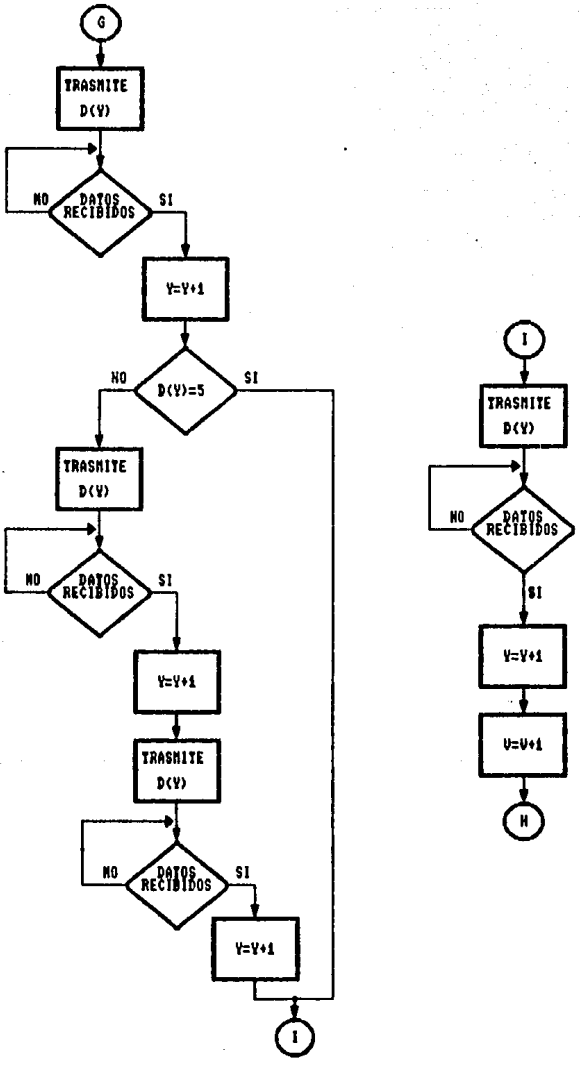


Figura III.17 Diagrama de flujo del programa BASIC (Cont.)

PROGRAMA DE LA INTERFAZ

El programa del sistema operativo permite controlar el funcionamiento de tres motores de pasos, de los cuales dos motores (denominados Motor 1 y Motor 2), son los encargados de realizar el movimiento de la pieza mecánica en el plano XY. Cabe mencionar que el movimiento de estos dos motores puede realizarse de manera simultánea o independientemente según se desee.

El tercer motor (denominado Motor 3), es el encargado de accionar la pieza mecánica en el plano Z. La característica de este motor es que funcionará cada vez que alguno o los dos motores anteriores, según sea que giren independiente o dependientemente respectivamente, hayan terminado de ejecutar sus pasos correspondientes; o si se desea que no gire, sólo se le indica con el valor correspondiente de espera (05H). El número de pasos máximo que podrán ejecutar los motores es de 65535 (FFFFH).

El programa del sistema operativo se divide en tres etapas, las cuales se mencionan a continuación:

1.- Etapa de adquisición de datos.
2.- Etapa de inicialización de los valores de registros y localidades de memoria para el llamado a rutinas.
3.- Etapa de ejecución de rutinas.

Antes de describir en detalle las etapas anteriores, estableceremos el mapa de memoria interno del microcontrolador, el cual se presenta en la *Tabla III.3*, en la siguiente página.

En la *Tabla III.4*, de la siguiente página, se muestran las localidades de memoria empleadas para el almacenamiento de los datos que harán funcionar los motores.

LOCALIDADES	FUNCION
75H - 7FH	APUNTADOR DE PILA (STACK POINTER)
20H - 74H	MEMORIA DE USO GENERAL
10H - 17H	BANCO DE REGISTRO 2
08H - 0FH	BANCO DE REGISTRO 1
00H - 07H	BANCO DE REGISTRO 0

Tabla III.3 Mapa de memoria interno del microcontrolador

LOCALIDAD	USO
20H	ALMACENA EL BYTE MAS SIGNIFICATIVO DE LA DIFERENCIA DEL NUMERO DE PASOS DEL MOTOR 1 Y EL MOTOR 2, CUANDO GIRAN JUNTOS
21H	ALMACENA EL BYTE MENOS SIGNIFICATIVO DE LA DIFERENCIA DEL NUMERO DE PASOS DEL MOTOR 1 Y EL MOTOR 2, CUANDO GIRAN JUNTOS
22H	ALMACENA EL BYTE MAS SIGNIFICATIVO DEL MOTOR QUE GIRARA SOLO O QUE TIENE EL NUMERO DE PASOS MENOR CUANDO GIRAN JUNTOS
23H	ALMACENA EL BYTE MENOS SIGNIFICATIVO DEL MOTOR QUE GIRARA SOLO O QUE TIENE EL NUMERO DE PASOS MENOR CUANDO GIRAN JUNTOS
24H	ALMACENA MARCA DEL MOTOR QUE GIRARA SOLO O QUE TIENE QUE DAR MAYOR NUMERO DE PASOS EN EL CASO DE QUE GIREN LOS DOS MOTORES JUNTOS SI 24H = #FEH, ES MOTOR 1 SI 24H = #FFH, ES MOTOR 2
26H	ALMACENA EL VALOR DEL PULSO INICIAL PARA EL MOTOR QUE GIRARA SOLO O DEL MOTOR 1, CUANDO GIRAN JUNTOS
27H	ALMACENA EL VALOR DEL PULSO INICIAL PARA EL MOTOR 2 CUANDO GIRAN JUNTOS
28H	ALMACENA EL VALOR DE TIEMPO ENTRE CADA PASO DE LOS MOTORES
29H	ALMACENA EL VALOR QUE DETERMINA SI EL PULSO DE SALIDA SERA POR EL PUERTO P1 O POR EL PUERTO P2 SI 29H = #00H, SALIDA POR PUERTO P2 SI 29H = #03H, SALIDA POR PUERTO P1
2AH	INDICA SI LOS DOS BYTES DE LA DIFERENCIA DEL NUMERO DE PASOS DE LOS MOTORES 1 Y 2, ES CERO SI 2AH = #00H, LA DIFERENCIA ES CERO SI 2AH = #01H, AL MENOS UN BYTE ES DIFERENTE DE CERO
2BH	GUARDA EL VALOR DEL PULSO EN QUE SE QUEDO EL MOTOR 1 CUANDO EL MOTOR 2 SIGUE GIRANDO
2EH	ALMACENA EL BYTE MAS SIGNIFICATIVO CUANDO SE GIRA MAS DE 255 PASOS
30H - 33H	ALMACENA LOS VALORES DE LOS PULSOS DE GIRO DEL MOTOR 1
34H - 37H	ALMACENA LOS VALORES DE LOS PULSOS DE GIRO DEL MOTOR 2
38H - 44H	ALMACENA EL VALOR DEL TIEMPO PROGRAMABLE DE ESPERA
45H - 54H	ALMACENA LOS DATOS DE SENTIDO DE GIRO Y NUMERO DE PASOS DEL MOTOR 1
56H - 65H	ALMACENA LOS DATOS DE SENTIDO DE GIRO Y NUMERO DE PASOS DEL MOTOR 2
66H - 74H	ALMACENA LOS DATOS DE SENTIDO DE GIRO Y NUMERO DE PASOS DEL MOTOR 3

Tabla III.4 Localidades específicas

La explicación del sistema operativo y de sus subrutinas se hará auxiliándose de los diagramas de flujo.

En la *Figura III.18* se presenta el diagrama de flujo general de operación del sistema operativo.

Como se observa en la *Tabla III.4*, de la localidad 45H a la 74H se ha dividido la memoria RAM interna en tres sectores, en donde se almacenarán los datos para cada motor como se indica en la misma tabla. Hacemos notar que se necesitan tres localidades de memoria RAM para establecer el sentido de giro (una localidad) y el número de pasos (dos localidades) para cada motor, excepto cuando se desee que un motor determinado no gire y en este caso sólo se requiere una localidad de memoria, la cual tendrá el indicador de que el motor correspondiente no girará en ese momento (05H).

Después de recibir datos de los motores 1 y 2, se deberá recibir un "0" para seleccionar una nueva tabla en donde se almacenará el valor correspondiente a un tiempo de espera en segundos. Esta tabla empezará en la localidad 3BH y terminará en la 44H. Cada vez que se termine el movimiento de los dos primeros motores se ejecutará una rutina de tiempo de 1 segundo tantas veces como lo indique el valor del tiempo de espera. Después se moverá el motor 3 y al terminar su movimiento, se volverá a ejecutar dicha rutina. El tiempo máximo de espera es de 255 segundos.

Para que el microcontrolador reconozca el fin de un paquete de datos, se graba en la localidad siguiente al último dato de los motores 1 y 2 el valor 45H. De esta manera se ejecutarán las instrucciones que se establezcan en el paquete de datos hasta que se encuentre el valor 45H en los datos del motor 1 y 2. Una vez que se haya detectado este valor aparecerá en la pantalla de la computadora la letra "F" y en este momento la computadora verifica si hay más datos para ser trasmitidos al

microcontrolador. En caso afirmativo, se envía el siguiente paquete de datos y nuevamente el último dato que se debe trasmitir será el valor 45H.

En el caso de que la computadora verifique que ya no hay más datos para trasmitir, entonces envía el valor 54H (letra T). El microcontrolador al reconocer este valor envía el valor 41H (letra A) y acaba su operación.

Como se puede apreciar, la computadora es quien tiene el control del sistema y al microcontrolador sólo le corresponde ejecutar las instrucciones que se le indiquen.

Descripción general de las etapas del sistema operativo

1. Etapa de adquisición de datos

En esta etapa se establece el protocolo mencionado anteriormente entre la computadora y el microcontrolador. Para iniciar la adquisición de datos, se debe trasmitir desde la computadora una clave de acceso para que el microcontrolador inicie el proceso de almacenamiento de los datos correspondientes al sentido de giro y número de pasos para cada motor. La clave de acceso que se debe trasmitir corresponde al valor 13H. Si no se trasmite este valor, aparecerá en la pantalla de la computadora el mensaje "OTRO" que indica que el acceso fue negado y se debe repetir el procedimiento.

Una vez que se tiene acceso aparecerá en la pantalla el mensaje "LISTO" y en ese momento se estará en condiciones para trasmitir los datos de sentido de giro y número de pasos que se almacenarán.

Los valores correspondientes al número de motor no se grabarán en la memoria RAM del microcontrolador, sólo nos indicarán el rango

de localidades en donde se deben grabar los datos de sentido de giro y número de pasos para cada motor.

Cuando se ha permitido el acceso para almacenar datos, el primer valor que se tiene que trasmitir deberá corresponder a un dato de motor. Para indicar que se ha recibido un dato correspondiente a un motor aparecerá en la pantalla la letra "S" para indicar que se seleccionó la tabla de memorización correspondiente. El siguiente dato que se debe trasmitir corresponde al sentido de giro, entonces, al ser recibido este dato aparecerá en la pantalla la letra "M" para indicar que el dato ha sido memorizado. Posteriormente se enviará el dato correspondiente al número de pasos que podrá ser hasta 65535. Como este valor corresponde a 2 bytes, primero se debe trasmitir el valor del byte más significativo el cual será almacenado en una localidad después que la localidad del sentido de giro y aparecerá en la pantalla la letra "M". Lo mismo ocurrirá cuando se envíe el valor del byte menos significativo.

En el caso de que se desee que un motor no gire, después de enviar el valor del motor se debe trasmitir el valor 05H que indica que el motor correspondiente se va a esperar.

De esta manera se irán almacenando los datos de tal forma que el microcontrolador reconoce si el dato que está analizando corresponde al sentido de giro o al número de pasos sin que exista posibilidad de error.

2. Etapa de inicialización de los valores de registros y localidades de memoria para el llamado a rutinas

En esta etapa se van analizando las localidades correspondientes al motor 1 y al motor 2 inicialmente, en donde se pueden presentar los siguientes casos:

2a) El motor 1 terminó sus datos (valor 45H en su última localidad) y el motor 2 continúa

En este caso se verifica si el dato del motor 2 corresponde a un valor 03H (sentido hacia adelante) o 04H (sentido hacia atrás). El motor 2 girará el número de pasos correspondientes al sentido de giro que le indique el dato anterior. La rutina encargada de ejecutar estos pasos se denomina RUT1 y anteriormente se deben establecer las condiciones para que se ejecuten correctamente los pasos. Las condiciones son las siguientes:

22H <---- contiene el valor del byte más significativo del número de pasos.

23H <--- contiene el valor del byte menos significativo del número de pasos.

24H <--- contiene la marca del motor 2 (FFH).

29H <--- indicador para tener la salida de los pulsos por el puerto P2 (29H = #00H).

R2 <--- indica el sentido de giro del motor.
Si R2 = 00H gira hacia adelante.
Si R2 = 01H gira hacia atrás.

El valor de la localidad del pulso de inicio de giro del motor 2 está establecido por dos registros (R4 y R6). Si el valor de R4 es 00H indica que es la primera vez que se acciona el motor 2 El valor de la localidad del pulso inicial se establece en la *Tabla III.5*, y se almacena en la localidad 26H.

R4 = #00H

SENTIDO DE GIRO	LOCALIDAD INICIAL DEL PULSO
ADELANTE	34H
ATRAS	37H

Tabla III.5 Determinación del valor del pulso inicial para el motor 2

Si el valor del registro R4 no es 00H, entonces se verifica el valor del registro R6, el cual contiene la localidad del último pulso que dio el motor 2.

Para el sentido hacia adelante tenemos: si R6 = #37H, se carga la localidad 26H con el valor #34H, en caso contrario se carga con el valor del registro R6.

Para el sentido hacia atrás tenemos: si R6 = #34H, se carga la localidad 26H con el valor #37H, en caso contrario se carga con el valor del registro R6.

Una vez que el motor 2 terminó de ejecutar sus pasos se pregunta por la localidad correspondiente al sentido de giro del motor 3. Los valores de giro hacia adelante y hacia atrás del motor 3, son iguales a los del motor 2, pero si el valor corresponde a 05H entonces el motor no girará.

Las rutina denominada VER3, cuyo diagrama de flujo se muestra en la *Figura III.37* (pág. 153), establece las condiciones para ejecutar los pasos del motor 3 (RUT1). Sólo se mencionarán las condiciones que difieren de las mencionadas para el motor 2, que son las siguientes:

24H <--- Marca del motor 3 (FEH).

29H <--- Indicador para tener la salida de los pulsos por el puerto P1. (29H = #03H)

Este proceso seguirá hasta que la localidad analizada del motor 2 contenga también el valor 45H. En este caso han terminado los datos del paquete correspondiente.

2b) El motor 1 continúa y el motor 2 terminó datos

La explicación para este caso esencialmente es la misma que para el caso anterior, sólo cambia la localidad 24H que tendrá el

valor FEH por referirse ahora al motor 1. Las condiciones para el motor 3 son las mismas que para el caso anterior.

El valor de la localidad del pulso de inicio de giro del motor 1 está determinado por los registros R3 y R5. Si el valor de R3 es #00H indica que es la primera vez que se acciona el motor 1. El valor inicial se establece como lo indica la *Tabla III.6*, y se almacena en la localidad 27H.

R3 = #00H

SENTIDO DE GIRO	LOCALIDAD INICIAL DEL PULSO
ADELANTE	30H
ATRAS	33H

Tabla III.6 Determinación del valor del pulso inicial para el motor 1

Si el valor del registro R3 no es #00H, entonces se verifica el valor del registro R5 el cual contiene la localidad del último pulso que dio el motor 1.

Para el sentido hacia adelante tenemos: si R5 = #33H se carga en la localidad 26H el valor #30H, en caso contrario se carga con el valor del registro R5.

Para el sentido hacia atrás tenemos: si R5 = #30H se carga en la localidad 26H el valor #33H, en caso contrario se carga con el valor del registro R5.

El valor de la localidad del pulso inicial para el motor 3 se determina de la misma manera que para el motor 1, sólo que los registros empleados corresponden al banco 1.

2c) Ambos motores tienen datos

En este caso se analizan los sentidos de giro de los dos motores y las combinaciones posibles de movimiento. La *Tabla III.7* muestra el valor del indicador (registro R2) para cada situación.

MOTOR 1	MOTOR 2	SENTIDO DE GIRO	INDICADOR (R2)
#03H	#03H	ADELANTE-ADELANTE	#00H
#04H	#04H	ATRAS-ATRAS	#01H
#03H	#04H	ADELANTE-ATRAS	#02H
#04H	#03H	ATRAS-ADELANTE	#03H

Tabla III.7 Combinaciones posibles de movimiento

La rutina encargada del movimiento de los dos motores se denomina RUT2, y su diagrama de flujo se presenta en la *Figura III.20*. Como en este caso los dos motores girarán juntos, es necesario establecer la diferencia del número de pasos entre ellos. Esto se realiza mediante la rutina denominada RESTA cuyo diagrama de flujo se presenta en la *Figura III.25* (pág. 131).

La rutina RESTA es la encargada de establecer qué motor tiene el mayor número de pasos o si son iguales.

Si son iguales, carga en la localidad 2AH el valor #00H, en caso contrario se carga el valor #01H.

Los motores girarán juntos el número de pasos menor y posteriormente continuará girando el motor que tiene el mayor número de pasos, tantos pasos como lo indique el resultado de la resta, en el caso de que no hayan sido iguales.

Las condiciones que se establecen en la rutina RESTA son las siguientes:

20H	<---	Byte más significativo de la resta
21H	<---	Byte menos significativo de la resta
22H	<---	Byte más significativo del motor que tiene el menor número de pasos
23H	<---	Byte menos significativo del motor que tiene el menor número de pasos
2AH	<---	Indica si el resultado de la resta fue cero (2AH = #00H)
24H	<---	Marca del motor que seguirá girando 24H = FEH gira el motor 1 24H = FFH gira el motor 2.

El valor de la localidad del pulso inicial para los motores se establece de igual forma que en los casos anteriores, sólo que ahora la localidad que guarda el valor del pulso del motor 1 es la 26H y la del motor 2 es la 27H. Las condiciones para el motor 3 son idénticas.

Cada vez que el motor 3 termina de ejecutar sus instrucciones se verifica la siguiente localidad del motor 1 y del motor 2. En caso de que haya más datos de sentido de giro se repite el procedimiento anterior; si las localidades contienen el valor 45H significa que se terminó de ejecutar el paquete de datos y se está en condiciones de recibir un nuevo paquete. Por último, si se identifica el valor 54H, indica que la computadora ya no tiene más datos que trasmitir y se termina la operación.

3. Etapa de ejecución de rutina

La rutina que se encarga de controlar el movimiento de un solo motor (sea motor 1, 2 o 3), se denomina RUT1 y su diagrama de flujo se muestra en la *Figura III.19* (pág.119).

En esta rutina se encuentran anidadas otras rutinas las cuales se mencionan a continuación, así como la función que realizan.

MENOS: se encarga de ejecutar los pasos que indica el byte menos significativo del número de pasos. Su diagrama de flujo se encuentra en la *Figura III.20* (pág. 121).

MAS: se encarga de ejecutar los pasos que indica el byte más significativo. Su diagrama de flujo se muestra en la *Figura III.22* (pág. 125).

RES: se encarga de restablecer los valores de las localidades válidas de los pulsos entre la ejecución de dos rutinas diferentes. Su diagrama de flujo se muestra en la *Figura III.24* (pág. 129).

EST3: se encarga de establecer los valores de las localidades válidas de los pulsos dentro de la ejecución de una misma rutina. Su diagrama de flujo se encuentra en la *Figura III.23* (pág. 127).

La rutina denominada RUT2, cuyo diagrama de flujo aparece en la *Figura III.26* (pág. 132), se encarga de realizar la misma función de RUT1 con la variante de que en este caso son dos motores los que se controlan a la vez. Se compone de las siguientes rutinas, que tienen las mismas funciones que las anteriores, respectivamente, y cuyos diagramas de flujo aparecen en las figuras que se mencionan a continuación de cada una de ellas.

MENOS1 (*Figura III.27*, pág. 134)
MAS1 (*Figura III.29*, pág. 138)
EST1 (*Figura III.30*, pág. 140)
RES1 (*Figura III.31*, pág. 142).

Dentro de esta misma rutina se encuentra otra denominada GIRA1 cuyo diagrama de flujo se muestra en la *Figura III.32* (pág.144).

Esta rutina se ejecuta cuando al menos un byte del resultado de la resta es diferente de cero y también se conforma de otras dos rutinas denominadas MENOSM1 y MASM1 cuya función es similar a las de RUT1, sólo que se toma en cuenta qué motor fue el que siguió girando. Sus diagramas de flujo están en las *Figuras III.33* y *III.35* respectivamente (págs. 145 y 149).

También, dentro de GIRA1, se encuentra la rutina EST2 (*Figura III.36*, pág. 152), que tiene la función de establecer las localidades de los pulsos para el movimiento del motor que siga girando.

Existen otras rutinas auxiliares que verifican el sentido de giro del motor que se activó. Estas rutinas son:

SENT1 (*Figura III.21*, pág. 124)
SENT2 (*Figura III.28*, pág. 137)
SENT3 (*Figura III.34*, pág. 148).

En las páginas siguientes se muestran los diagramas de flujo correspondientes al programa de la interfaz.

Diagrama de flujo general de operacion del sistema

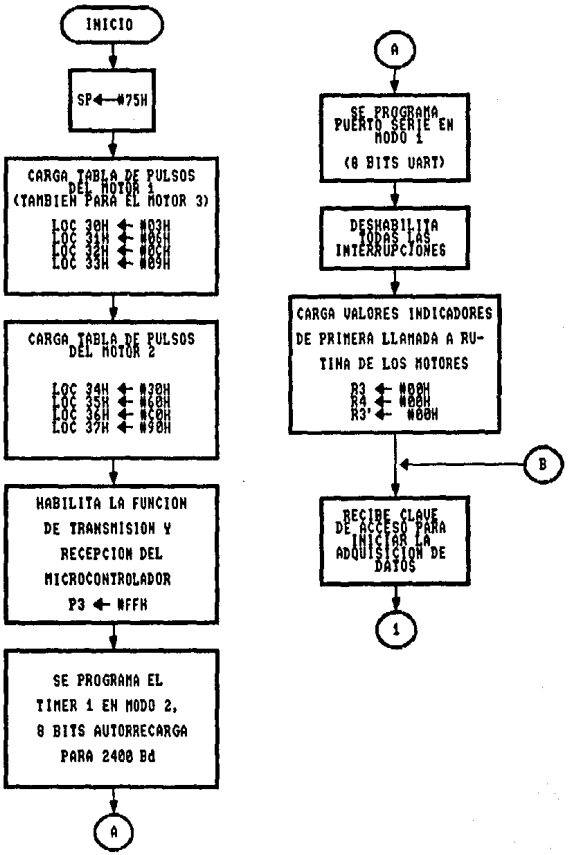


Figura III.18 Diagrama de flujo del programa de control

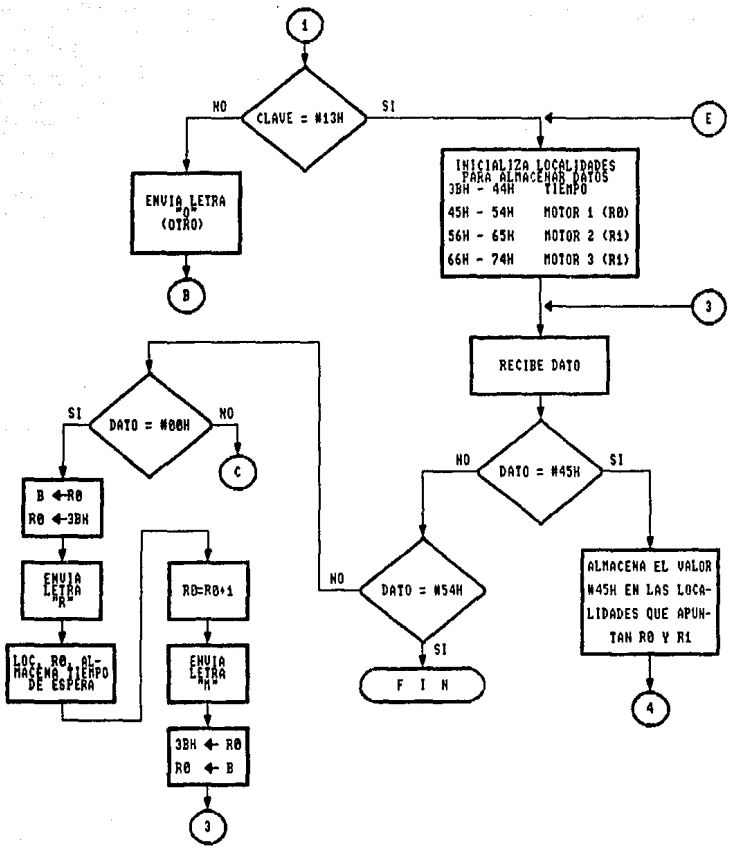


Figura III.18 Diagrama de flujo del programa de control (Cont.)

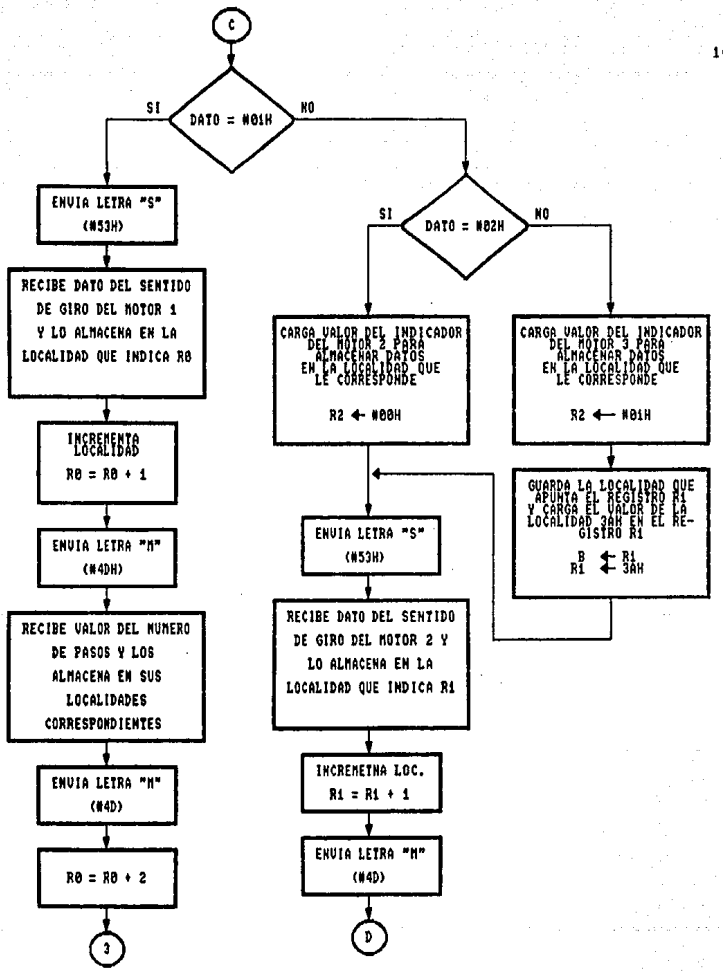


Figura III.18 Diagrama de flujo del programa de control (Cont.)

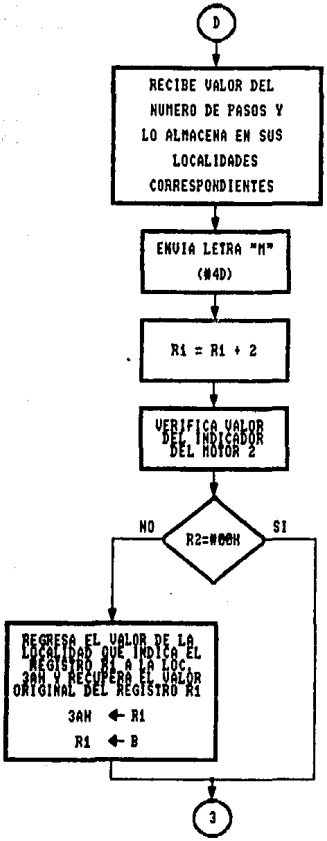


Figura III.18 Diagrama de flujo del programa de control (Cont.)

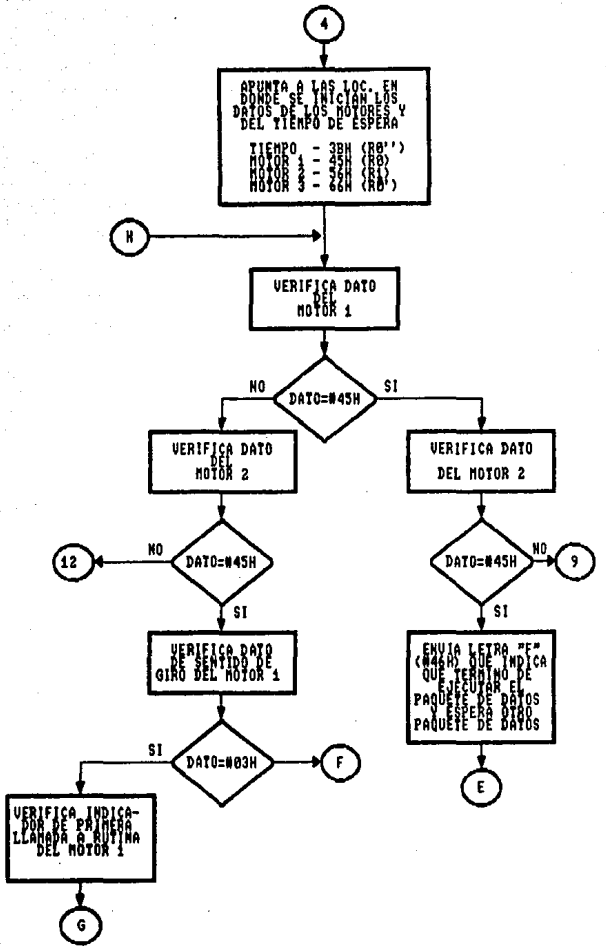


Figura III.18 Diagrama de flujo del programa de control (Cont.)

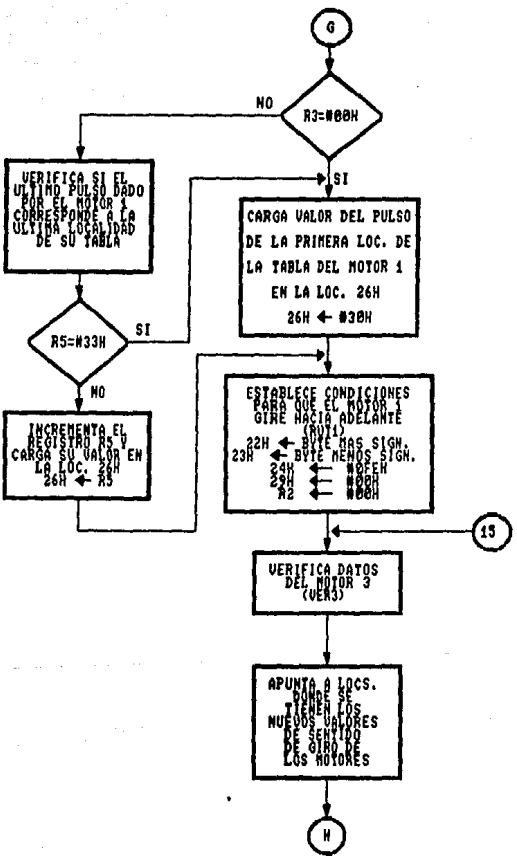


Figura III.18 Diagrama de flujo del programa de control (Cont.)

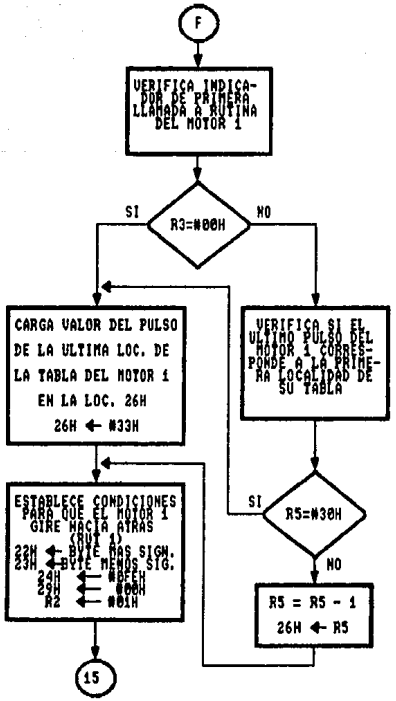


Figura III.18 Diagrama de flujo del programa de control (Cont.)

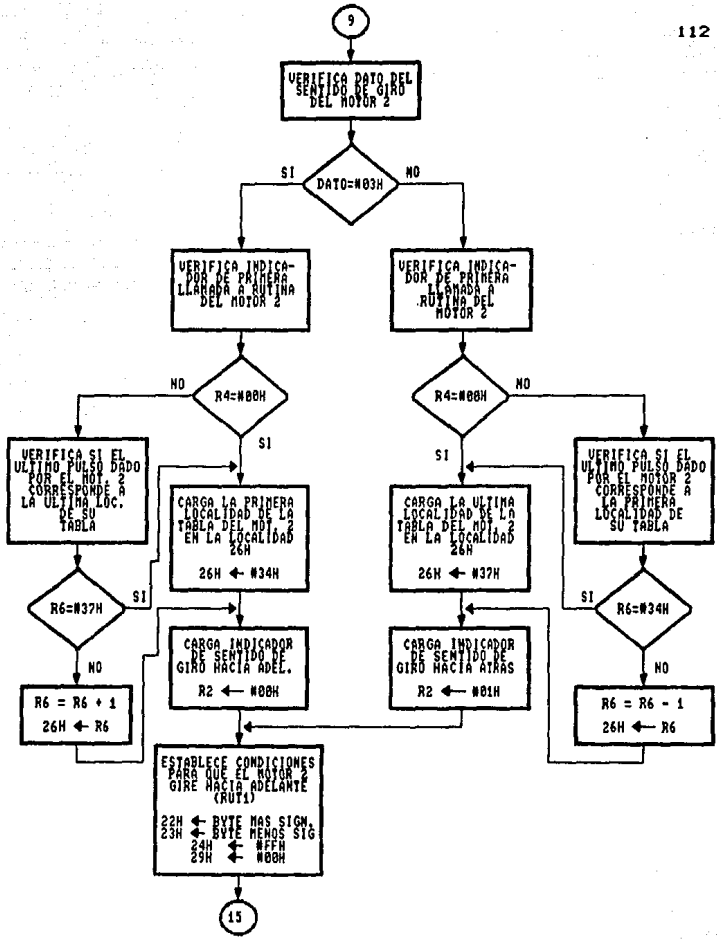


Figura III.18 Diagrama de flujo del programa de control (Cont.)

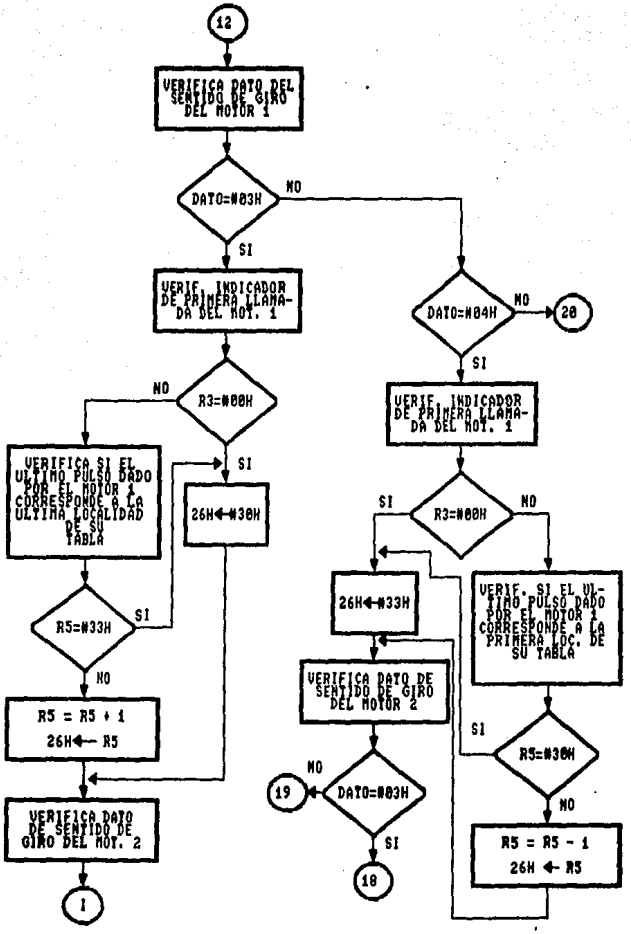


Figura III.18 Diagrama de flujo del programa de control (Cont.)

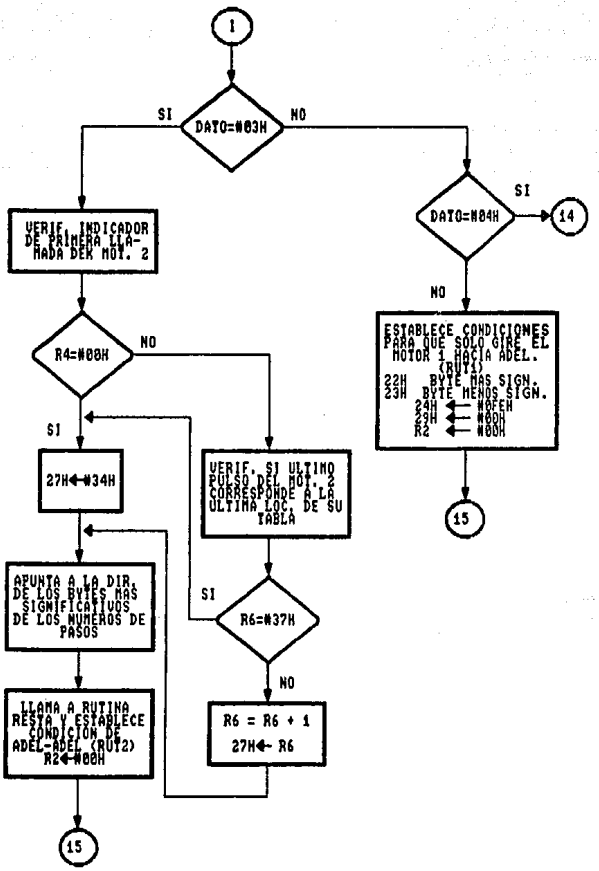


Figura III.18 Diagrama de flujo del programa de control (Cont.)

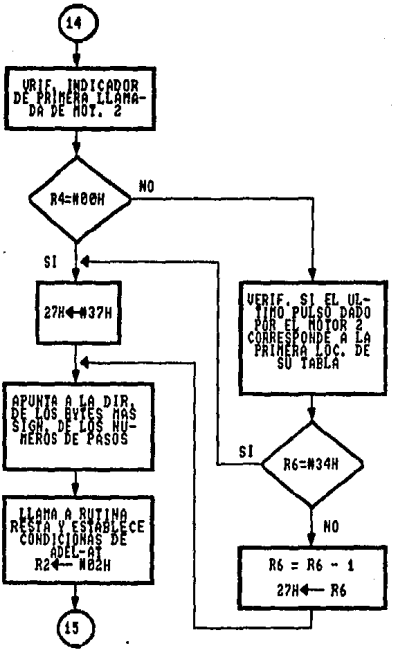


Figura III.18 Diagrama de flujo del programa de control (Cont..)

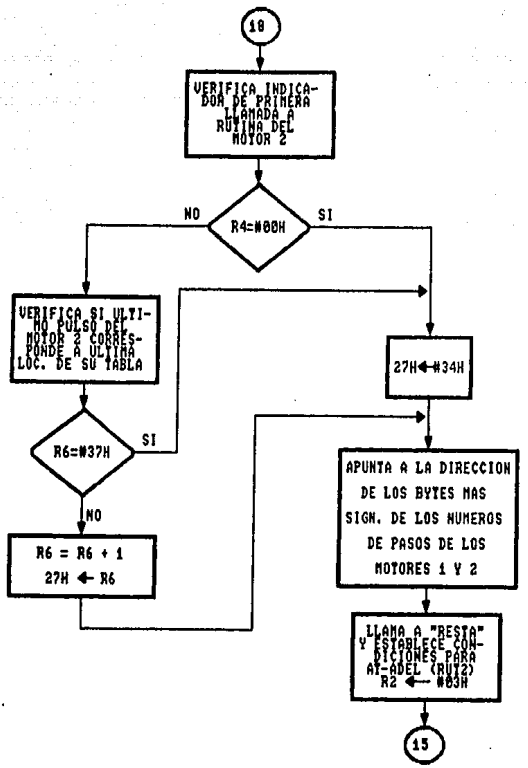


Figura III.18 Diagrama de flujo del programa de control (Cont.)

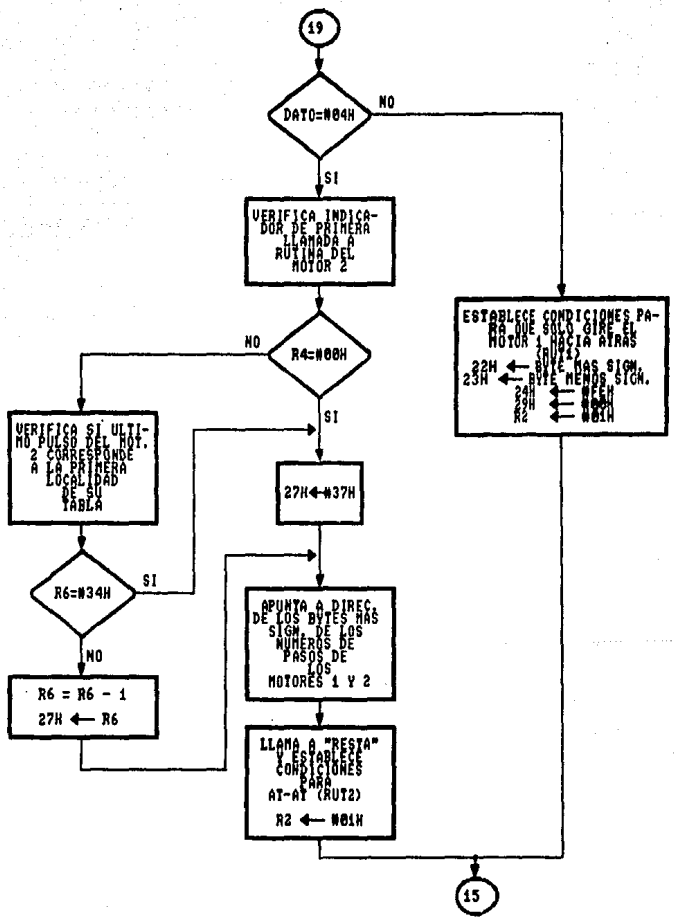


Figura III.20 Rutina RUT2

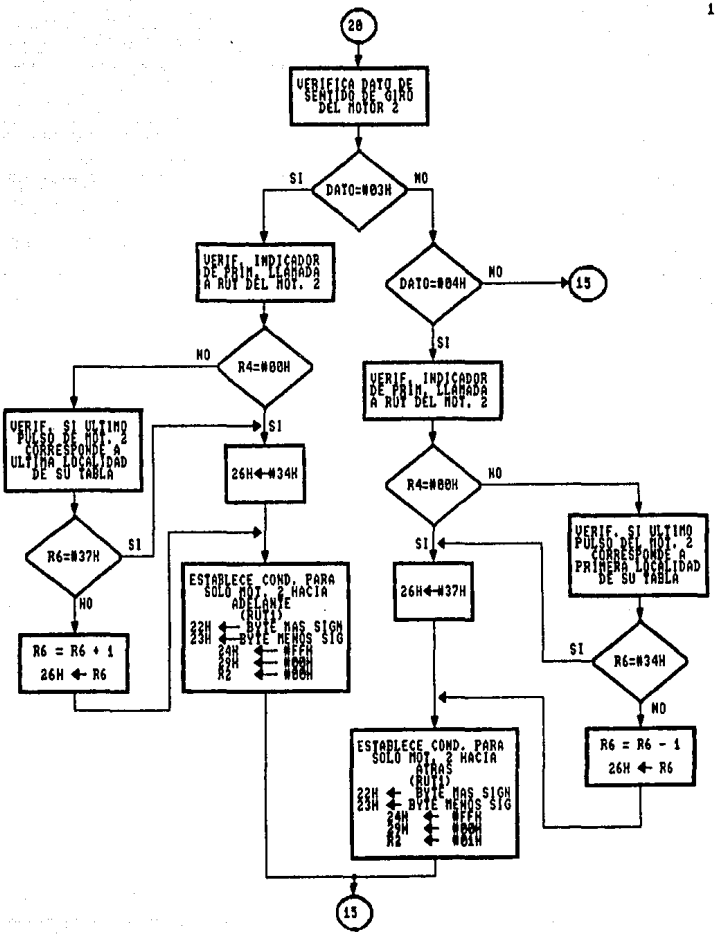


Figura III.20 Rutina RUT2 (Cont.)

DIAGRAMA DE FLUJO DE LA RUTINA 1 PARA EL CASO EN QUE SOLO GIRA UN MOTOR (RUT1)

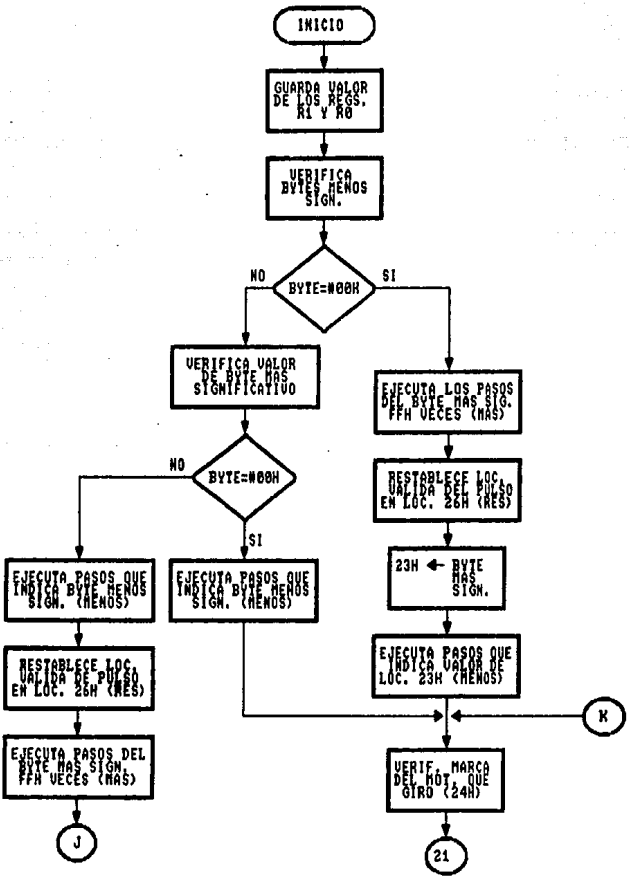


Figura III.19 Rutina RUT1

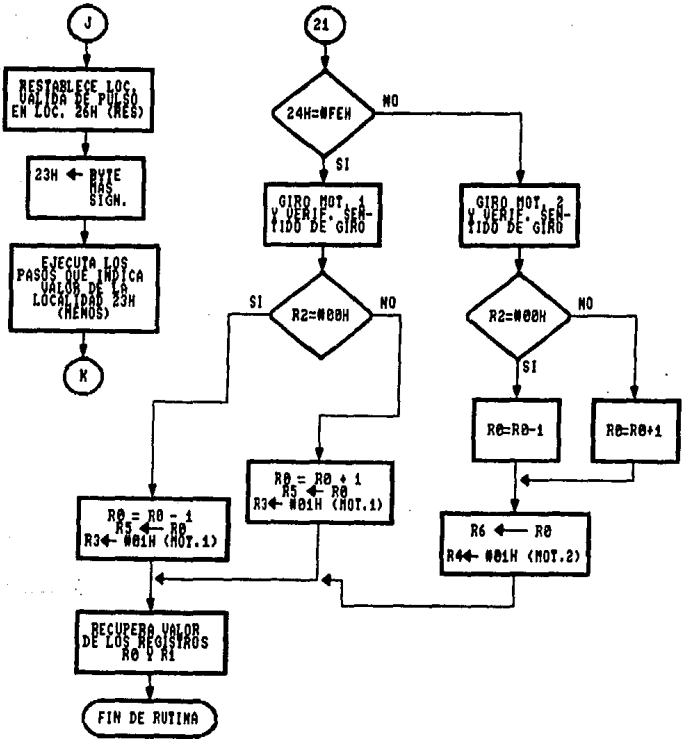


Figura III.19 Rutina RUT1 (Cont.)

DIAGRAMA DE FLUJO DE LA RUTINA QUE EJECUTA EL NUMERO DE PASOS QUE INDICA LA LOCALIDAD DEL BYTE MENOS SIGNIFICATIVO (23H) (MENOS)

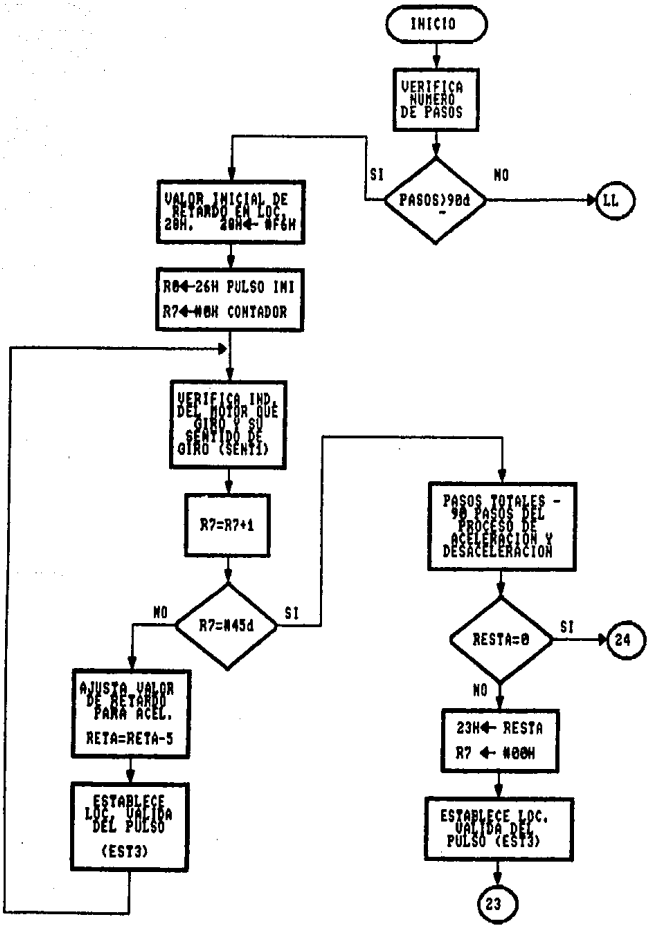


Figura III.20 Rutina MENOS

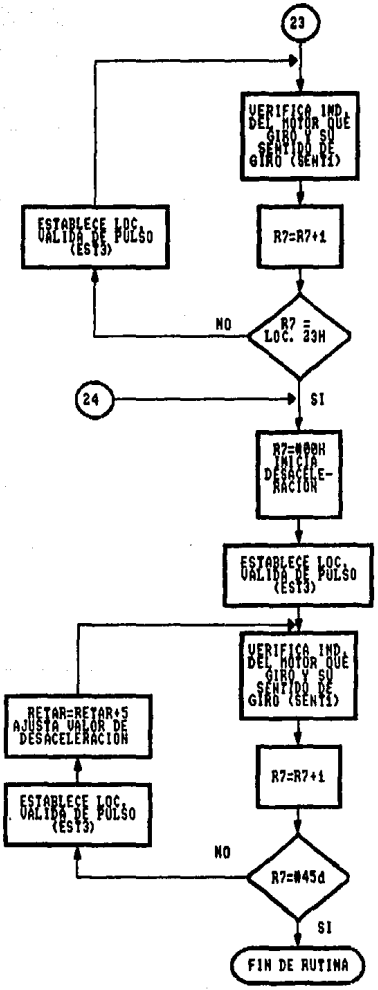


Figura III.20 Rutina MENOS (Cont.)

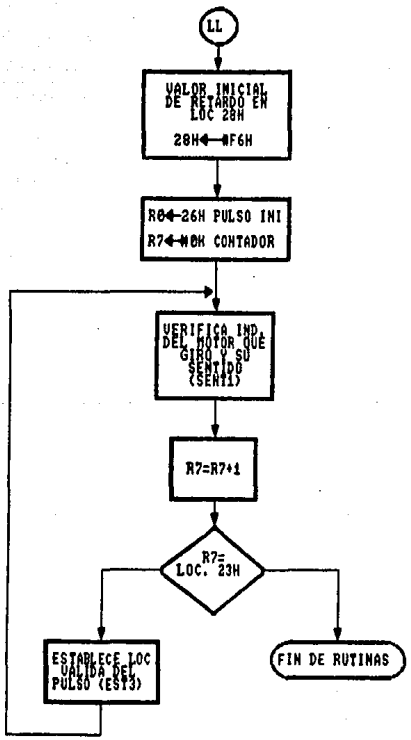


Figura III.20 Rutina MENOS (Cont.)

DIAGRAMA DE FLUJO DE LA RUTINA QUE VERIFICA EL MOTOR QUE GIRO Y SU SENTIDO DE GIRO (SENTI)

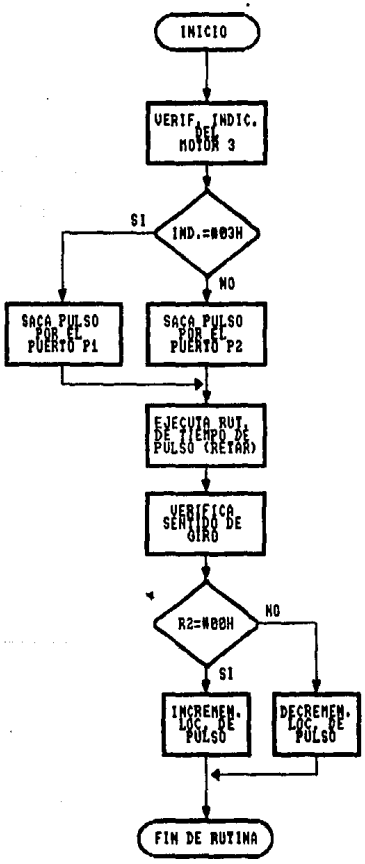


Figura III.21 Rutina SENTI

DIAGRAMA DE FLUJO DE LA RUTINA QUE EJECUTA LOS PASOS QUE INDICA LA LOCALIDAD DEL BYTE MAS SIGNIFICATIVO (22H) (MAS)

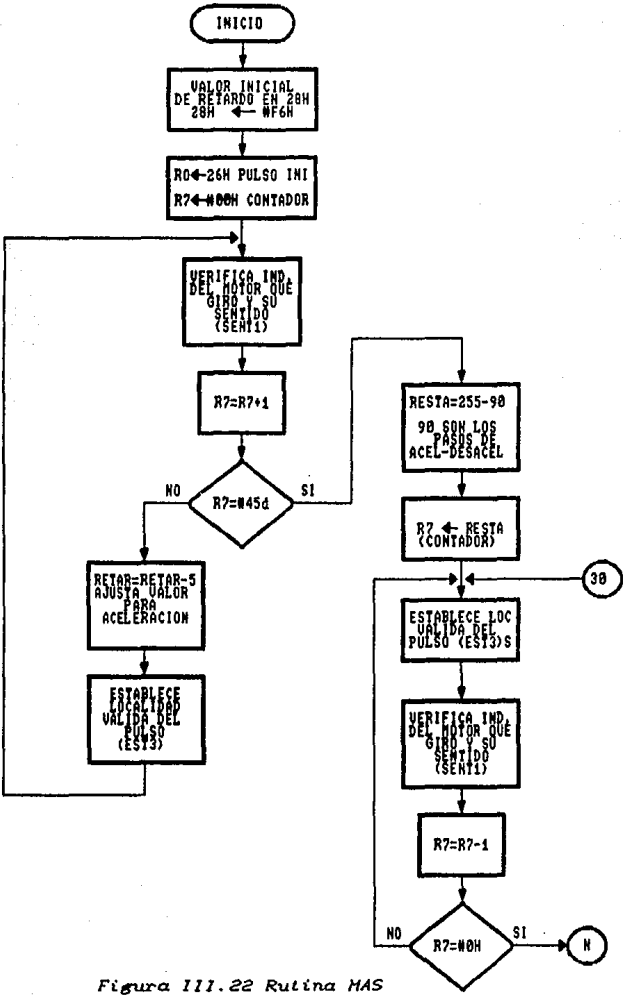


Figura III.22 Rutina MAS

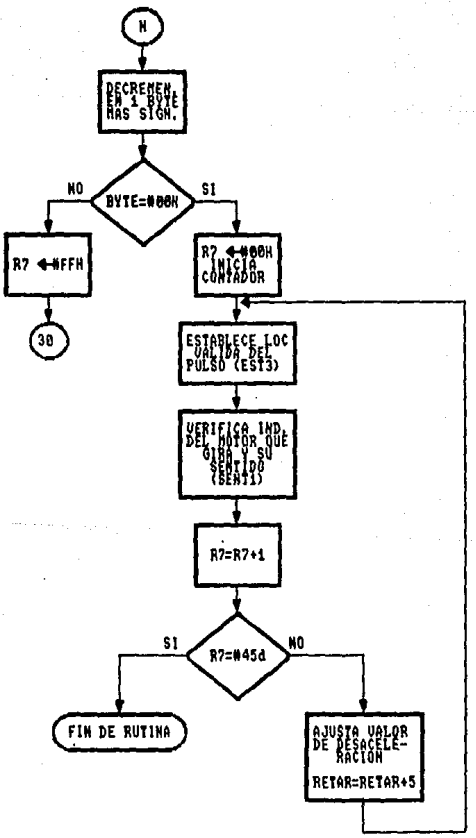


Figura III.23 Rutina MENOS (Cont.)

DIAGRAMA DE FLUJO QUE ESTABLECE LOS VALORES DE LAS LOCALIDADES VALIDAS PARA LOS PULSOS DENTRO DE UNA MISMA RUTINA PARA UN SOLO MOTOR (EST3)

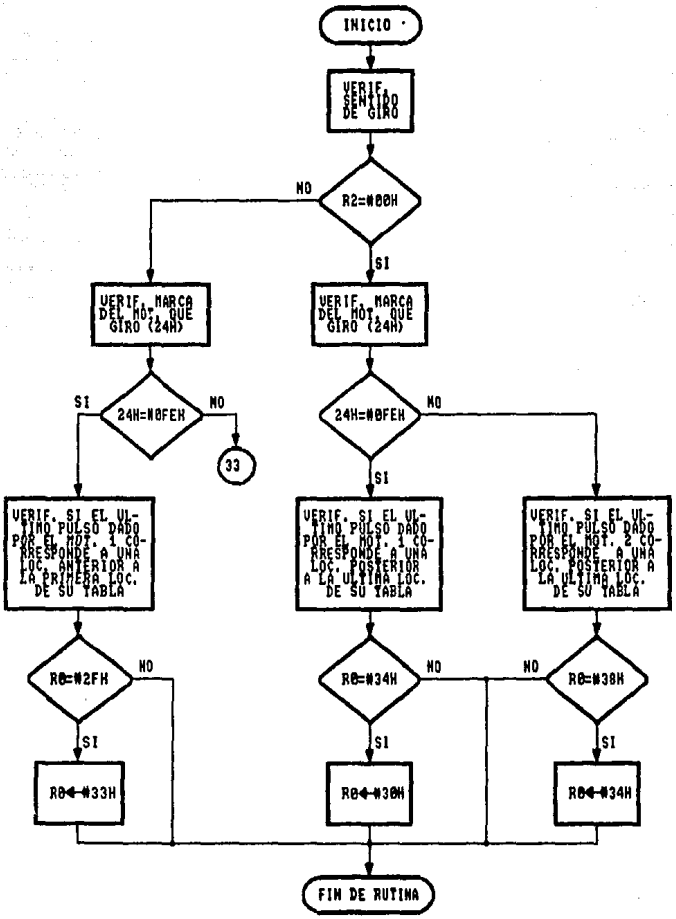


Figura III.23 Rutina EST3

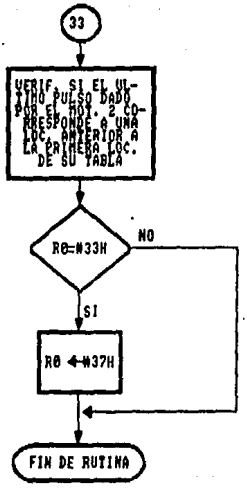


Figura III.23 Rutina EST3 (Cont.)

DIAGRAMA DE FLUJO DE LA RUTINA QUE RESTABLECE EL VALOR DEL PULSO DE INICIO CUANDO SE PASA DE UNA RUTINA DE EJECUCION DE PASOS A OTRA (RES)

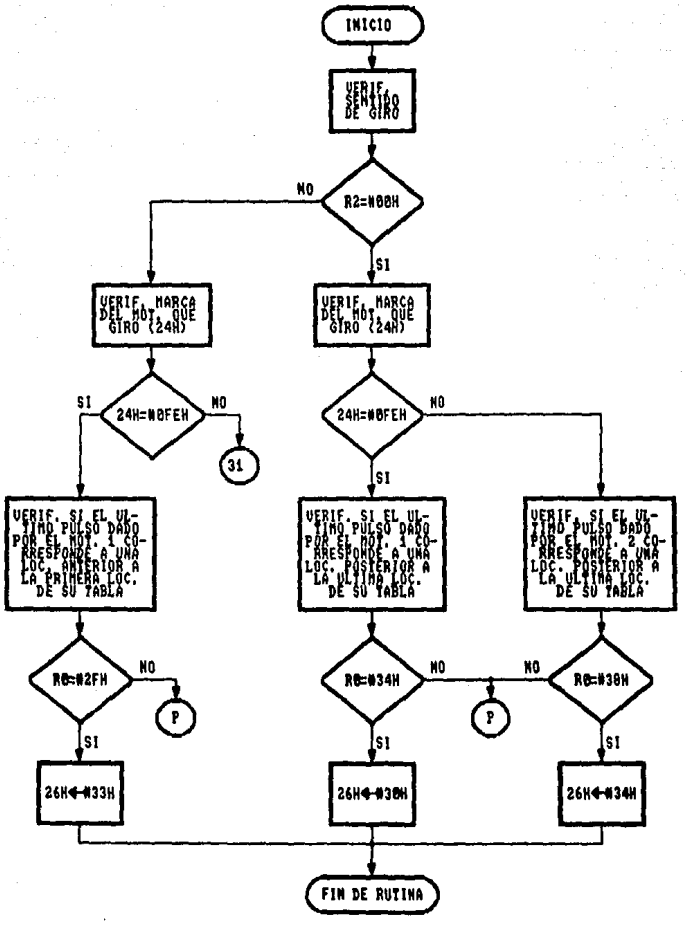


Figura III.24 Rutina RES

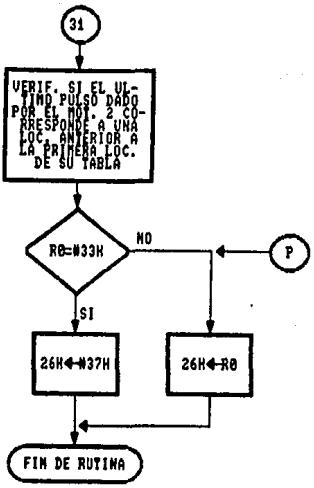


Figura III.24 Rutina RES (Cont.)

RUTINA QUE OBTIENE LA DIFERENCIA DEL NUMERO DE PASOS DE LOS MOTORES 1 Y 2 (RESTA)

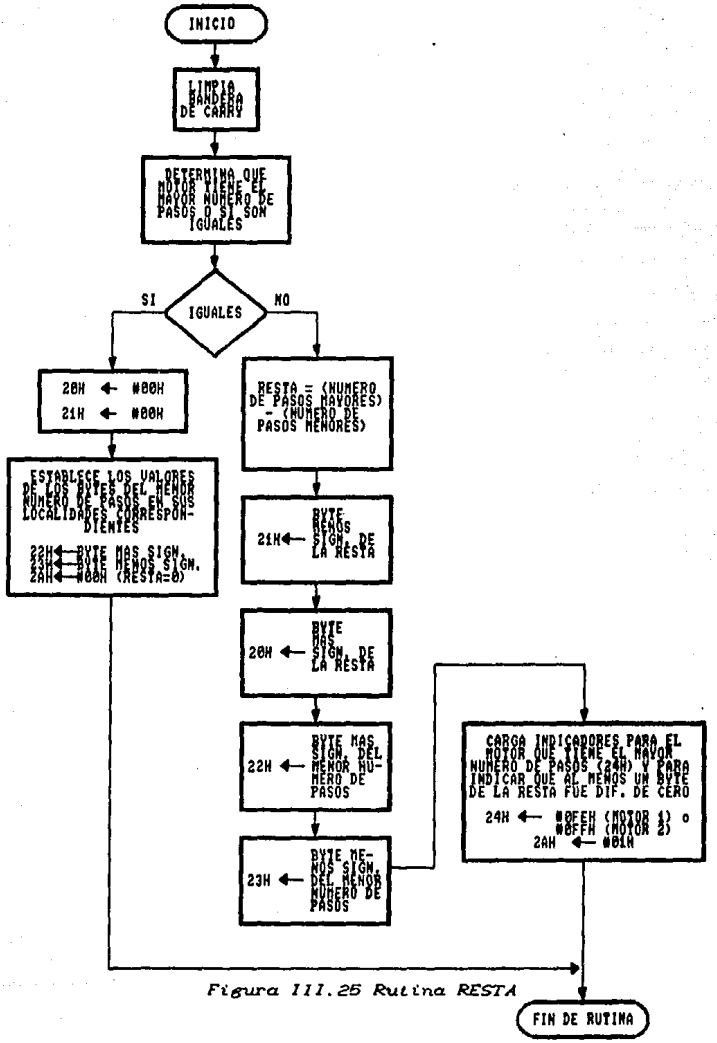


Figura III.25 Rutina RESTA

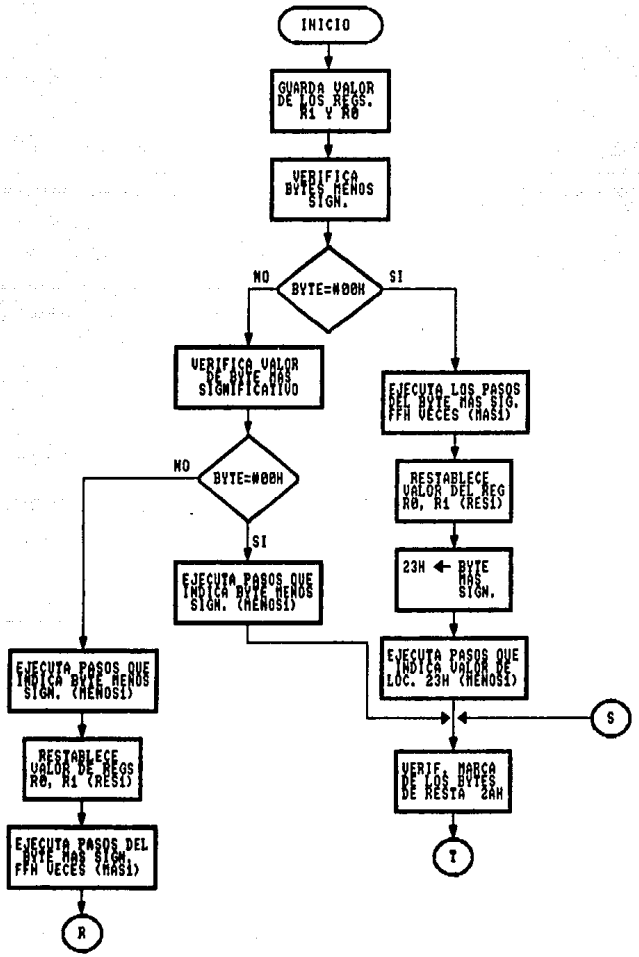


Figura III.26 Rutina RUT2

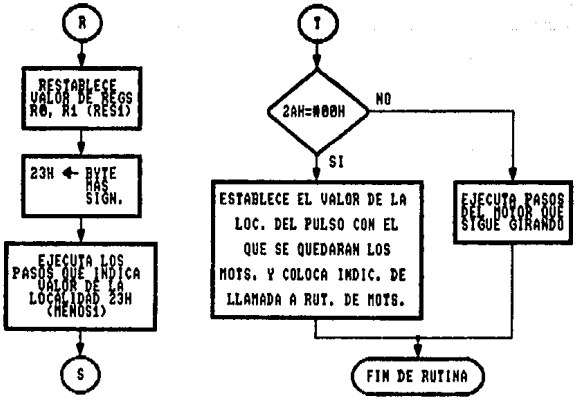


Figura III.26 Rutina RUT2 (Cont.)

DIAGRAMA DE FLUJO DE LA RUTINA QUE EJECUTA EL NUMERO DE PASOS QUE INDICA LA LOCALIDAD DEL BYTE MENOS SIGNIFICATIVO CUANDO GIRAN DOS MOTORES (MENOS 1)

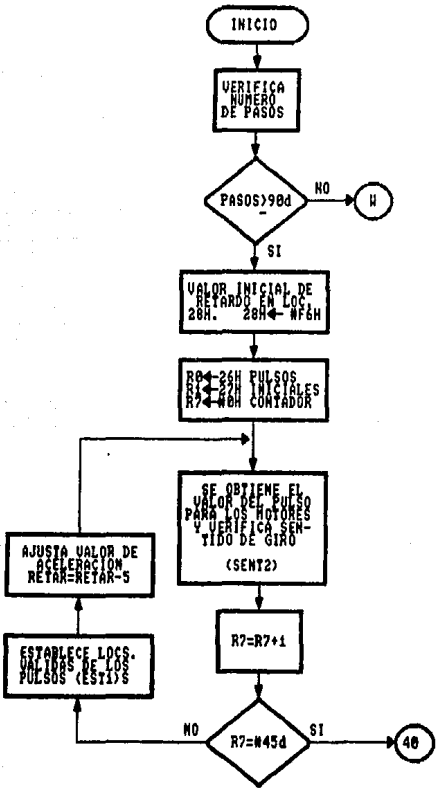


Figura III.27 Rutina MENOS1

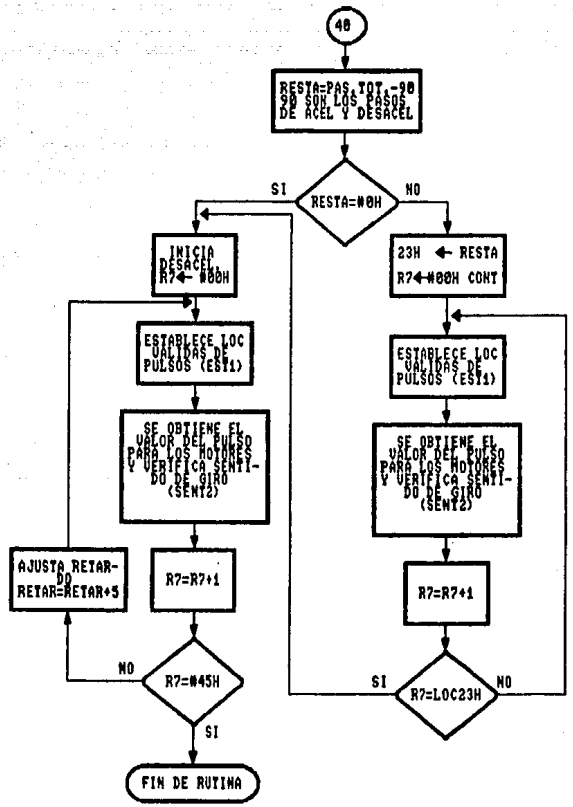


Figura III.27 Rutina MENOS1 (Cont.)

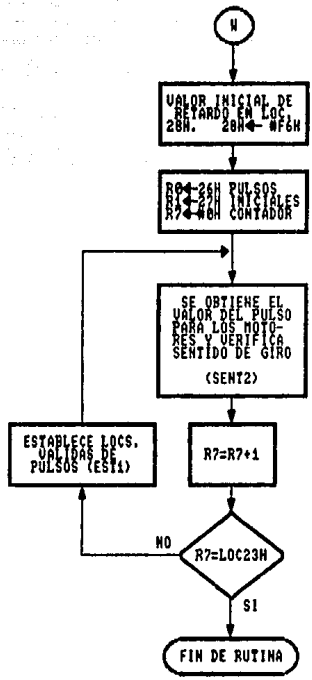


Figura III.27 Rutina MENOS1 (Cont.)

DIAGRAMA DE FLUJO DE LA RUTINA QUE ESTABLECE EL VALOR DEL PULSO PARA LOS MOTORES X Y Z VERIFICANDO SU SENTIDO DE GIRO (SENT2)

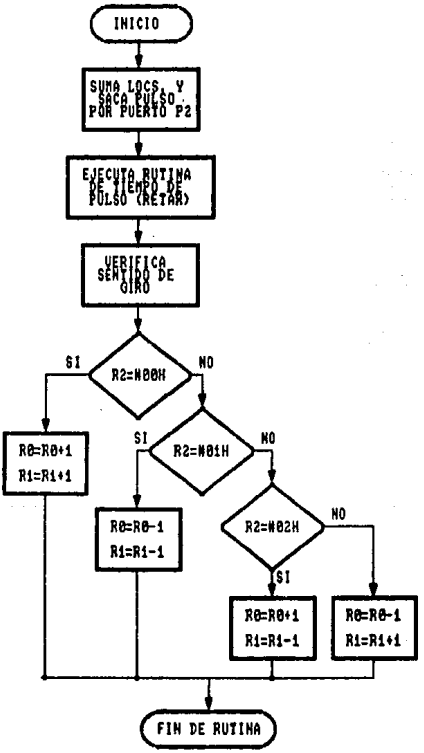


Figura III.28 Rutina SENT2

DIAGRAMA DE FLUJO DE LA RUTINA QUE EJECUTA EL NUMERO DE PASOS QUE INDICA LA LOCALIDAD DEL BYTE MAS SIGNIFICATIVO CUANDO GIRAN DOS MOTORES (MAS1)

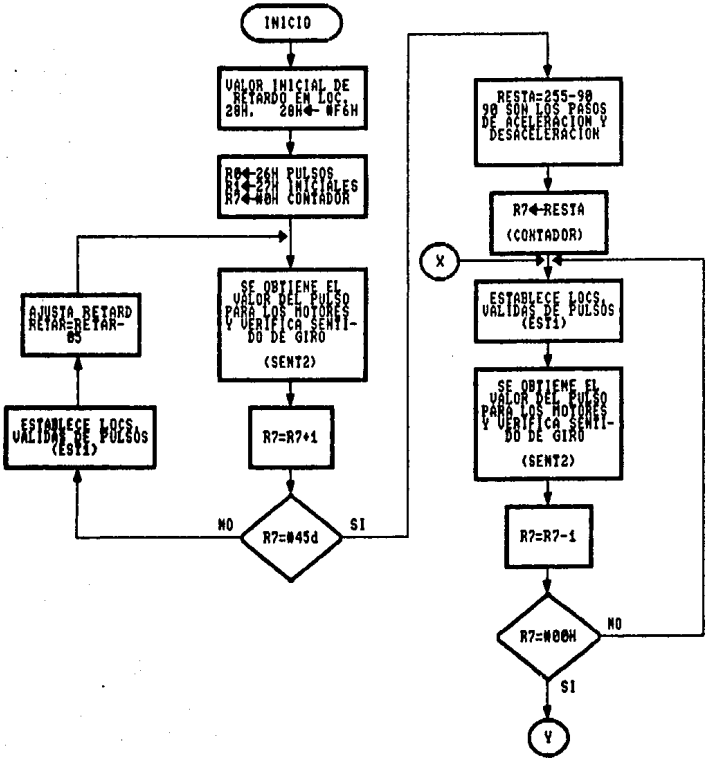


Figura III.29 Rutina MAS1

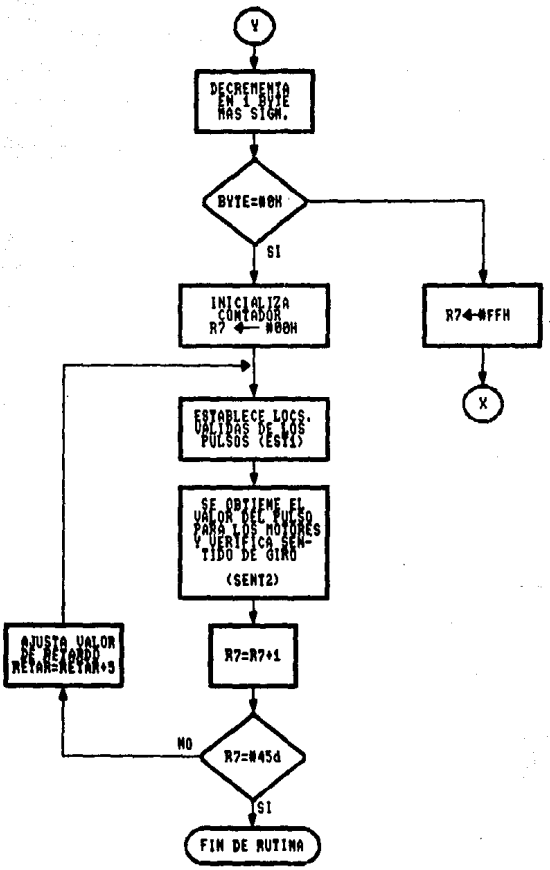


Figura III.29 Rutina MAS1 (Cont.)

DIAGRAMA DE FLUJO DE LA RUTINA QUE ESTABLECE LOS VALORES DE LAS LOCALIDADES VALIDAS PARA LOS PULSOS DENTRO DE UNA MISMA RUTINA, CUANDO GIRAN DOS MOTORES (ST1)

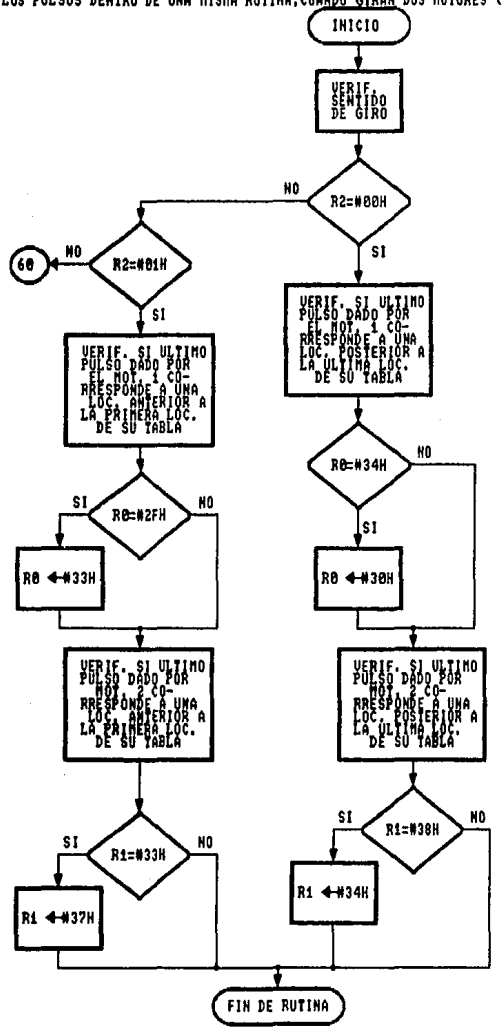


Figura III.28 Rutina MAS1 (Cont.)

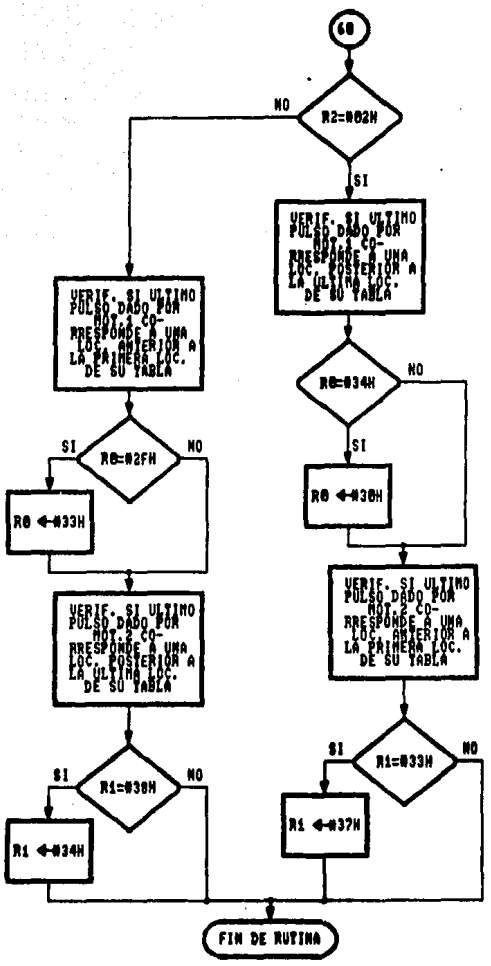


Figura III.30 Rutina EST1 (Cont.)

DIAGRAMA DE FLUJO DE LA RUTINA QUE RESTABLECE LOS VALORES DE LOS PULSOS DE INICIO CUANDO SE PASA DE UNA RUTINA DE EJECUCION DE PASOS PARA DOS MOTORES, A OTRA (RES1)

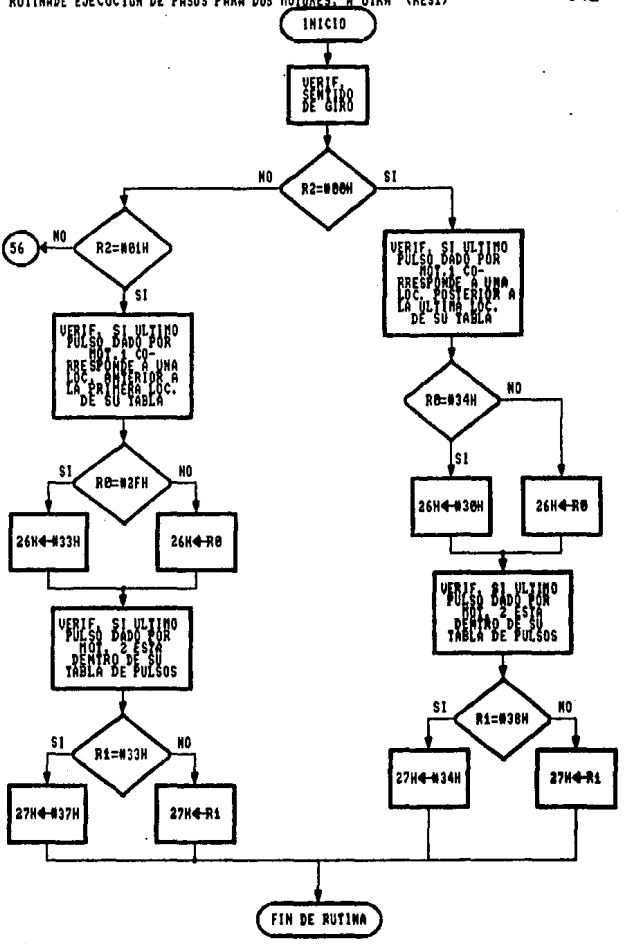


Figura III.31 Rutina RES1

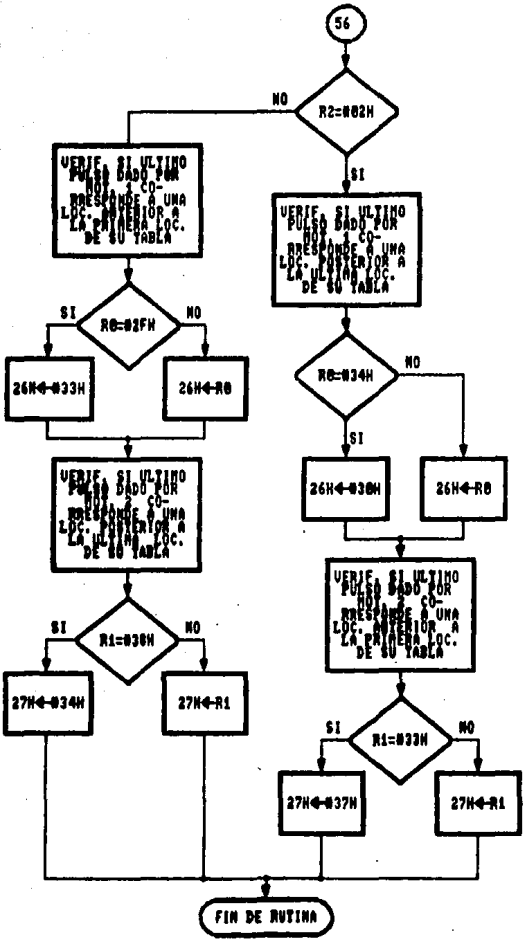


Figura III.31 Rutina RESI (Cont.)

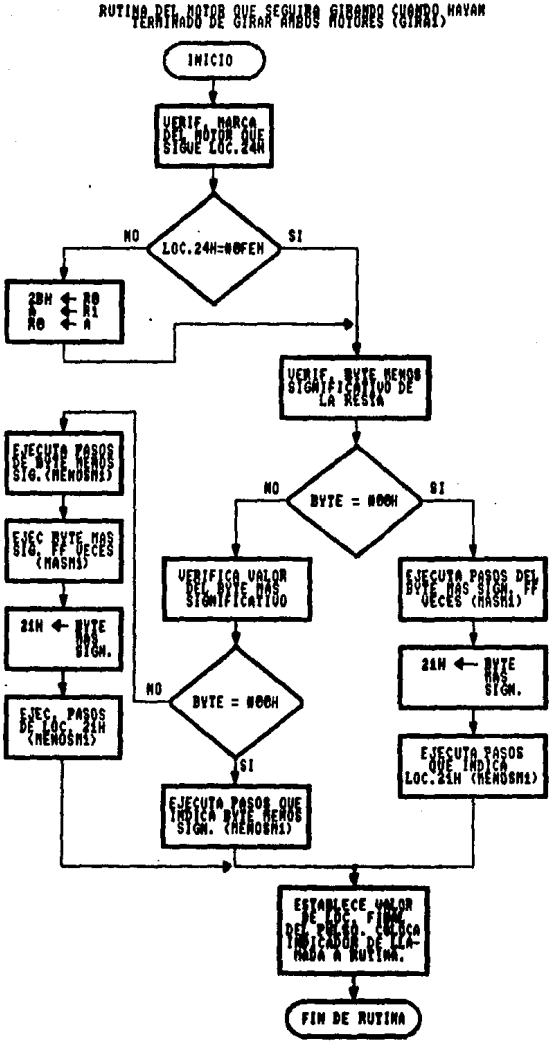


Figura III.30 Rutina ESTI (Cont.)

RUTINA QUE EJECUTA LOS PASOS QUE INDICA LA LOCALIDAD DEL BYTE MENOS SIGNIFICATIVO DEL MOTOR QUE SEGUIRA GIRANDO (MENOSM1)

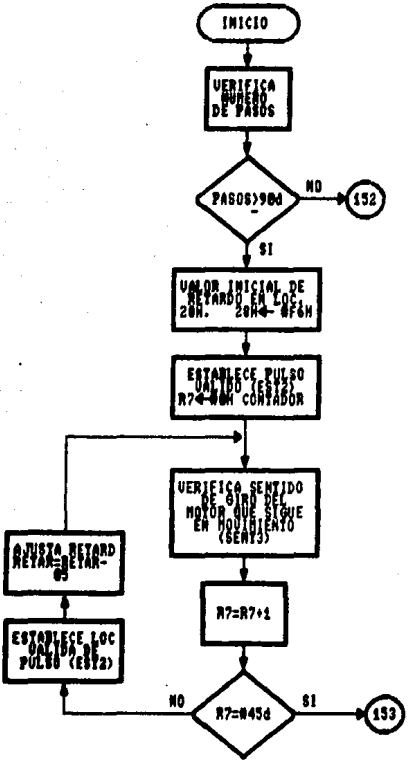


Figura III.33 Rutina MENOSM1

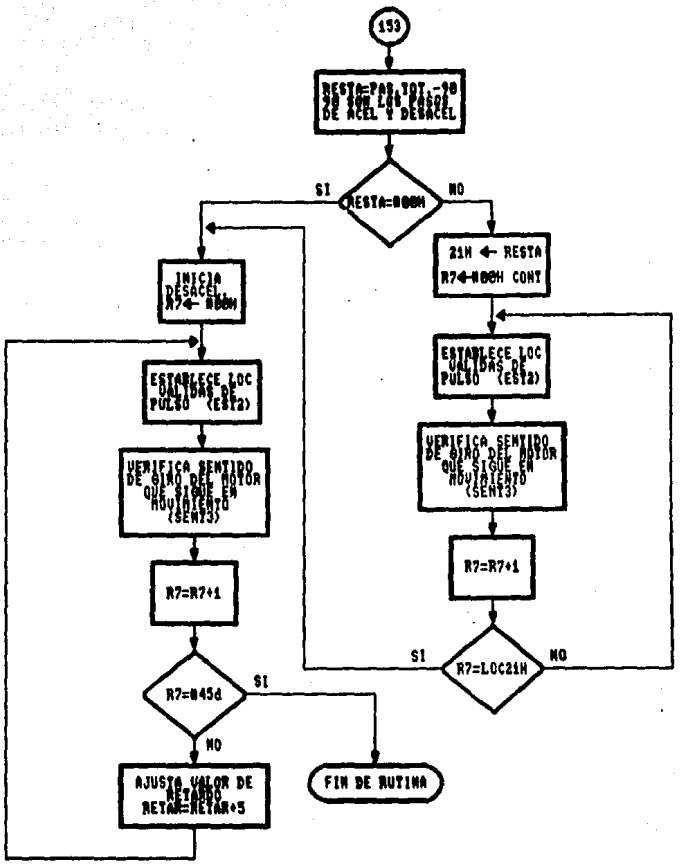


Figura III.33 Rutina MENOSM1 (Cont.)

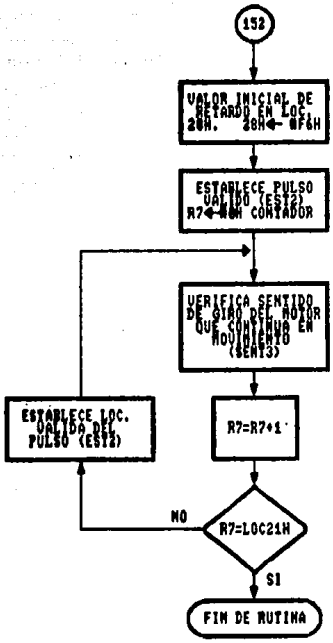


Figura III.33 Rutina MENOSM1 (Cont.)

DIAGRAMA DE FLUJO DE LA RUTINA QUE VERIFICA
EL SENTIDO DE GIRO DEL MOTOR
QUE CONTINUA EN MOVIMIENTO
(SENT3)

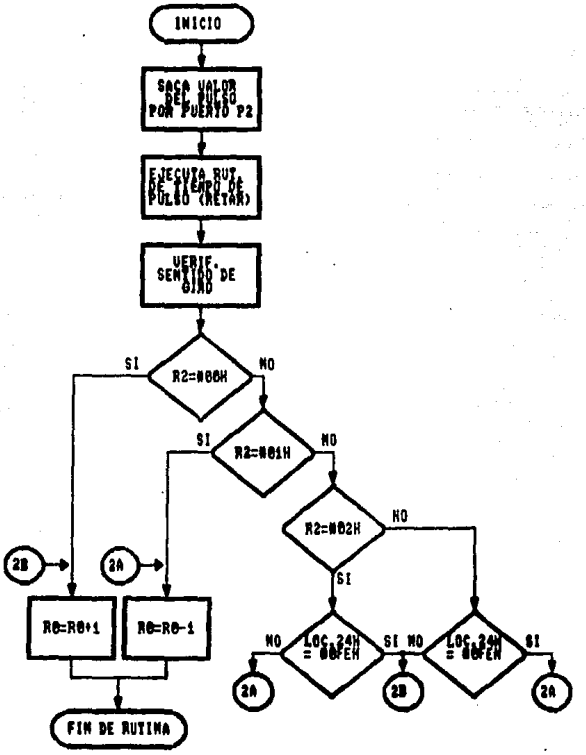


Figura III.34 Rutina SENT3

RUTINA QUE EJECUTA LOS PASOS QUE INDICA LA LOCALIDAD DEL BYTE MAS SIGNIFICATIVO DEL MOTOR QUE SEGUIRA GIRANDO (MASM1)

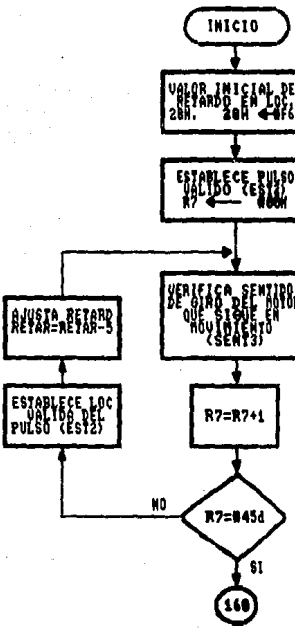


Figura III.35 Rutina MASM1

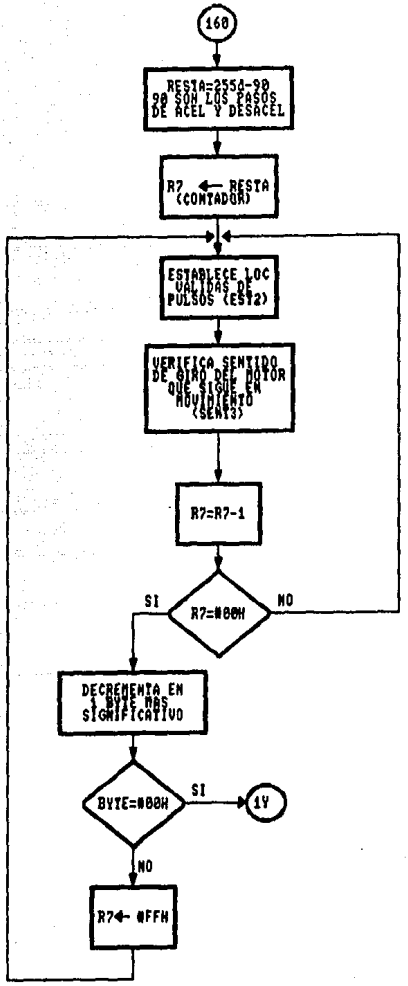


Figura III.35 Rutina MASM1 (Cont.)

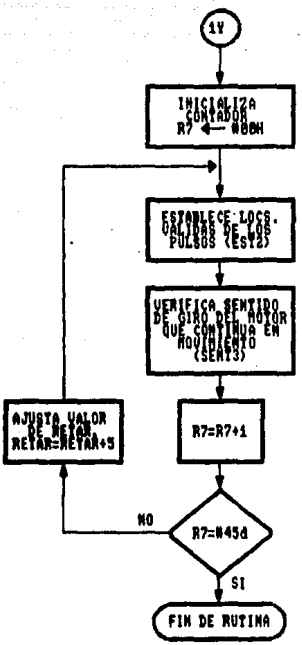


Figura III.35 Rutina MASMI (Cont.)

RUTINA QUE ESTABLECE LOS VALORES DE LAS LOCALIDADES VALIDAS DE LOS PULSOS PARA EL MOTOR QUE SIGUE GIRANDO (EST2)

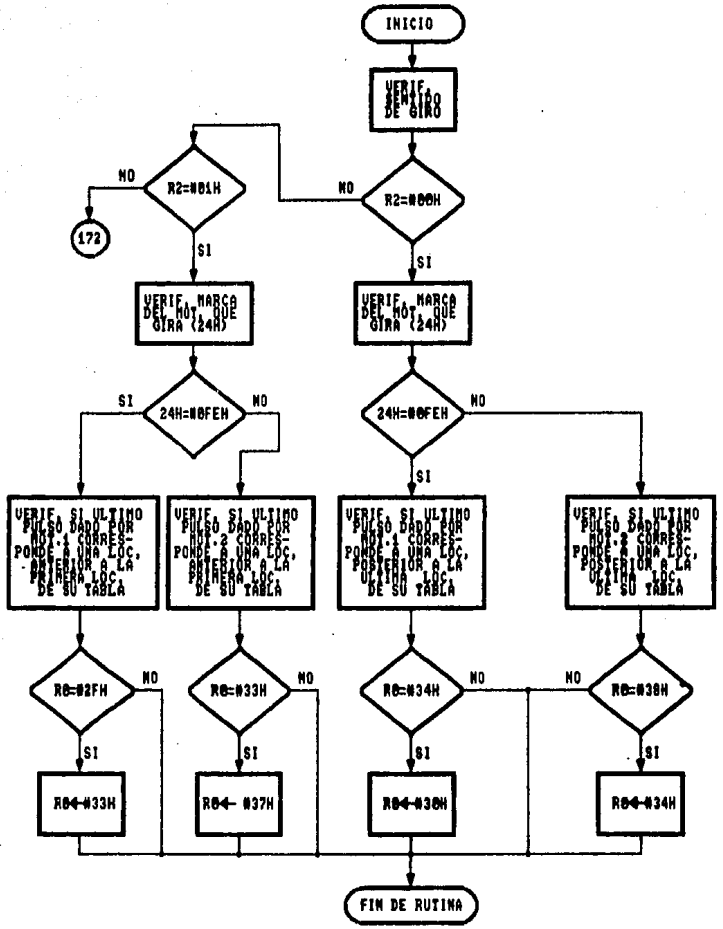


Figura III.36 Rutina EST2

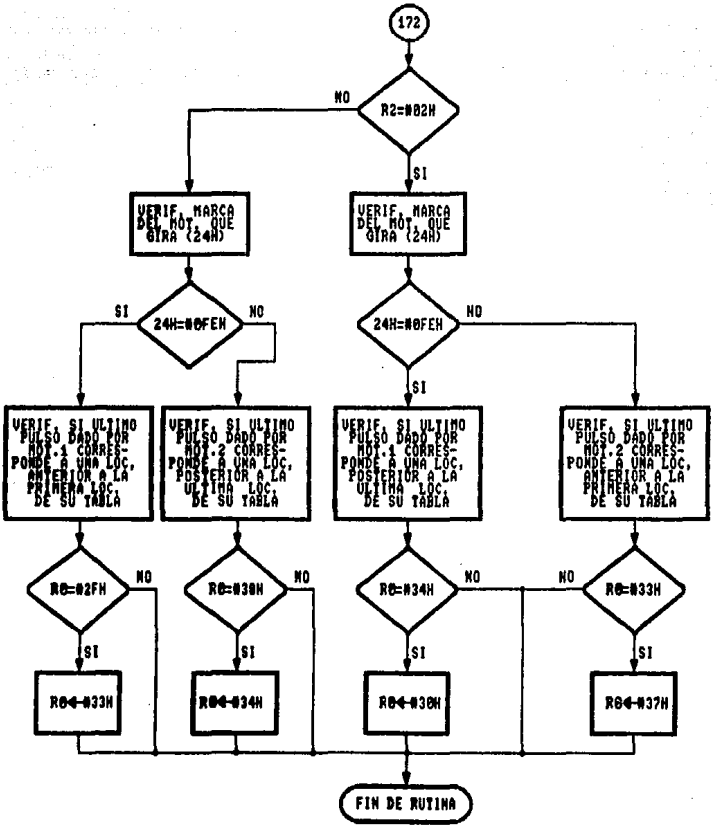


Figura III.36 Rutina EST2 (Cont.)

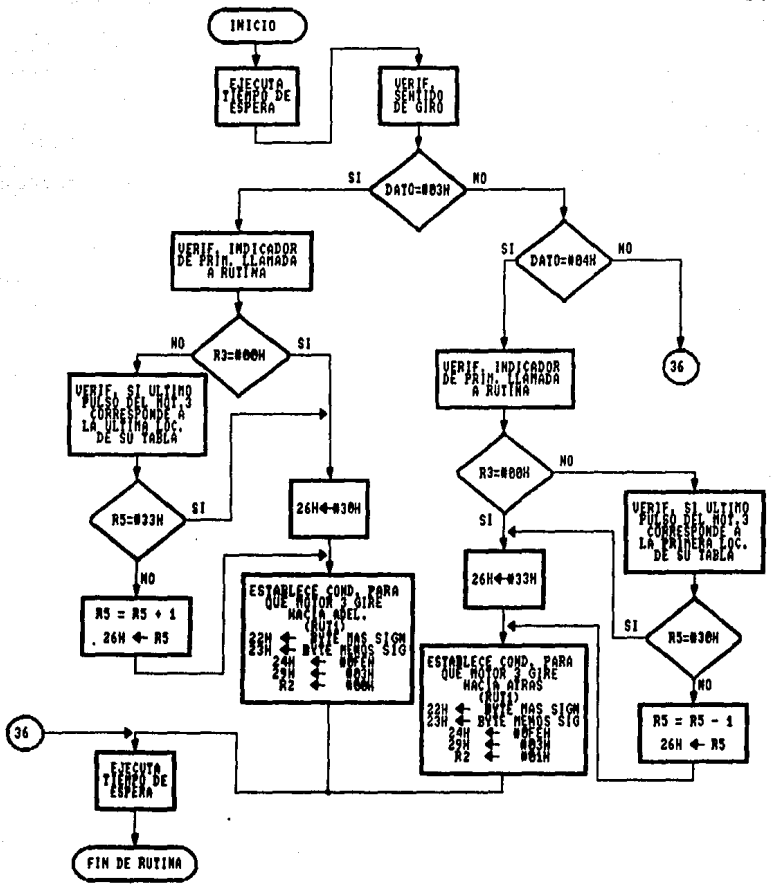


Figura III.37 Rutina VER3

TEMA IV
MODIFICACIONES

Al momento de ejecutar los programas definitivos, para probar el funcionamiento de la interfaz, nos dimos cuenta de que eran necesárias algunas modificaciones tanto en el programa BASIC de trasmisión como en el programa de control de la interfaz.

En cuanto al programa BASIC, la modificación que se le hizo consiste en la forma de pedir los datos al usuario. Con este nuevo programa el usario no deberá teclear los datos correspondientes al número de motor y sentido de giro. Tampoco dará el número de pasos. El programa actual, cuyo diagrama de flujo se muestra en la *Figura IV.1*, de la siguiente página, requiere que el usuario sólo indique la distancia que debe recorrer el mecanismo en cada uno de sus ejes, anteponiendo un signo + o − para indicar que el movimiento será hacia adelante o hacia atrás respectivamente.

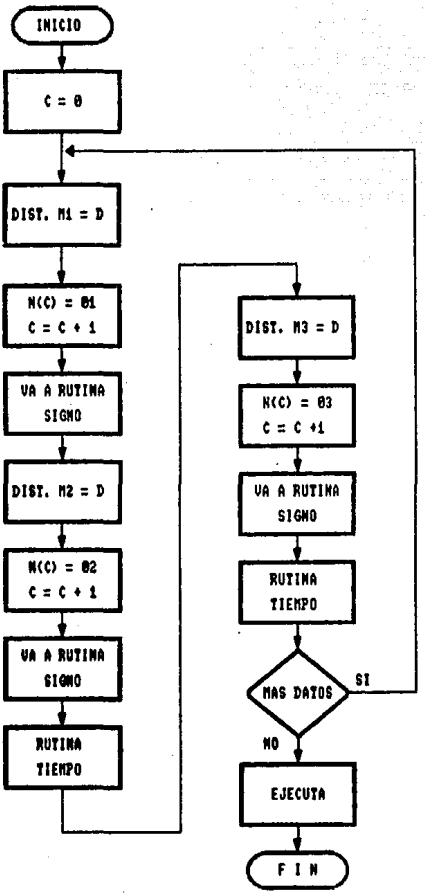


Figura IV.1 Programa BASIC modificado

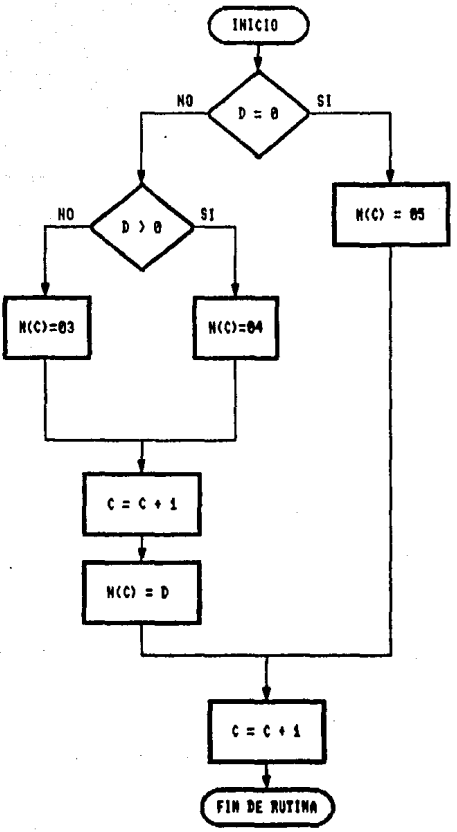


Figura IV.1 Programa BASIC modificado (Cont.)

Para el programa de control de la interfaz, la modificación que debe hacerse consiste en que siempre que inicie un proceso, lo haga a partir del mismo lugar. Esto es muy importante para evitar que el mecanismo se salga de sus límites de trabajo. Para conseguir esto tiene que tomarse en cuenta la circuitería electrónica y el programa de control. La parte electrónica debe sensar los extremos del mecanismo y realimentar una señal al microcontrolador para indicarle que está en uno u otro extremo. Una forma de hacer esto sería mediante unas perforaciones hechas en cada extremo del mecanismo tal como se muestra en la *Figura IV.2.*

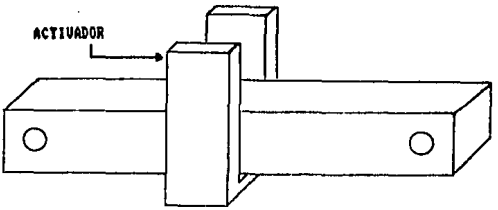


Figura IV.1 Sistema para la determinación de la posición de origen

Lo que se marca en la figura anterior como activador consistiría en un fototransistor activado por un led infrarrojo como se muestra en la *Figura IV.3.*

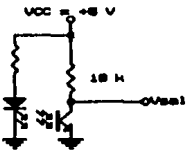


Figura IV.3 Activador optoelectrónico

La forma en que la señal enviada por el activador hará que el mecanismo se coloque en la posición de origen será de la

siguiente forma. El programa, antes de hacer cualquier cosa, debe sacar pulsos para que los motores (uno a la vez) se vayan moviendo hacia un lado especificado. Cada vez que el microcontrolador saque un pulso deberá leer el dato que haya en el puerto P0. Si el primer valor que lea es 00H sacará un pulso más. Si es 01H, que indica que se activó el pin P0.0, dejará de enviar pulsos para ese motor, pues estará en la posición de origen. En caso contrario seguirá sacando pulsos hasta leer el valor 02H, que indica que se activó el pin P0.1. En ese momento sacará los pulsos en sentido contrario hasta que se active el pin P0.0. Después hará lo mismo con el motor 2 y los pines P0.2 y P0.3 y cuando esté ese eje en el origen, realizará lo mismo con los pines P0.4 y P0.5, para el motor 3.

En la *Figura IV.4* de la siguiente página, se muestra el diagrama de flujo que corresponde a la colocación del mecanismo en la posición de origen. Esta parte del programa iría al inicio del programa general.

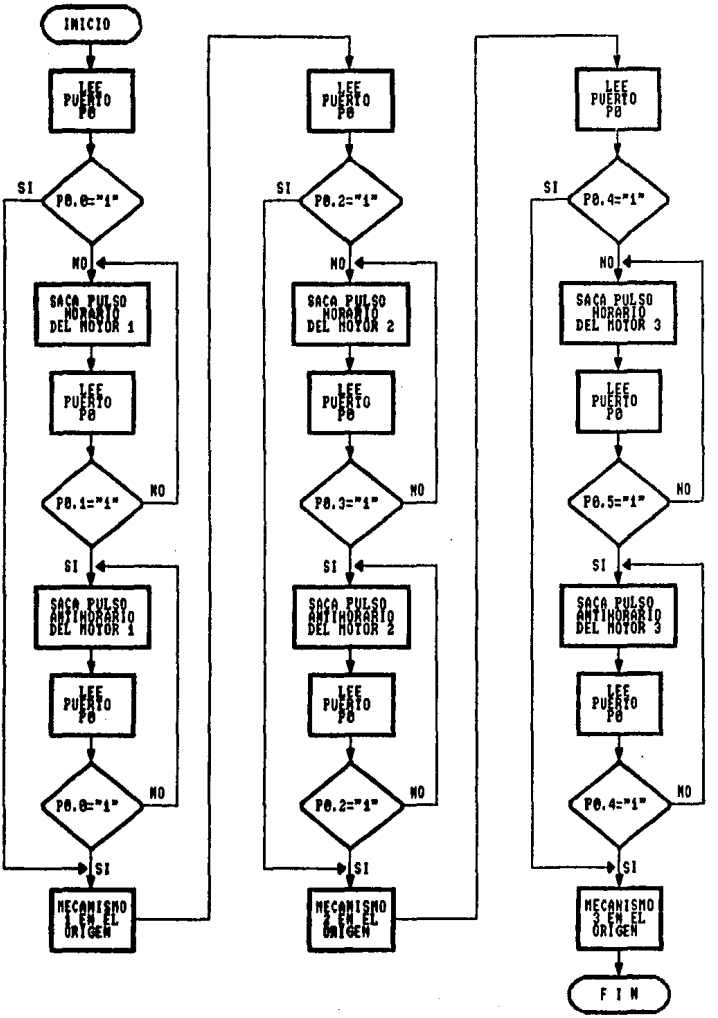


Figura IV.4 Colocación en la posición de origen

TEMA V
CONCLUSIONES

Al término de cualquier trabajo, se deben analizar los resultados obtenidos para determinar si se cumplieron los objetivos establecidos.

El objetivo de nuestro trabajo fue el diseño y construcción de un sitema elctrónico que permitiera operar una máquina herramienta de control numérico (de 2 1/2 D como mínimo) a partir de la información enviada desde una computadora.

El sistema electrónico se basó en el microcontrolador 8751 como unidad central de procesamiento. Para evitar el uso de una memoria RAM externa al microcontrolador, debido a la pequeña capacidad que poseé intermamente, se estructuró el programa de control de la computadora de tal manera que se enviaran paquetes de datos y así poder ocupar el rango de la memoria RAM disponible del microcontrolador en varias ocasiones; por lo tanto, el alcance de almacenamiento de instrucciones ya no depende de su

capacidad de memoria interna, sino de la capacidad del archivo que se crea al iniciar la operación del sistema.

Lo anterior nos permitió que fuera mínima la circuitería del diseño y, por lo tanto, más económico, que es un factor a considerar en el diseño de todo trabajo.

Por lo que respecta al programa de control del microcontrolador, se puede decir que cumple con la característica de modularidad, que es un factor importante para posibles modificaciones que se puedan presentar.

Por otra parte, para que nuestro trabajo resultara más eficiente se hicieron algunas modificaciones para que las instrucciones a ejecutar se dieran directamente en valores de distancia (en mm) para el desplazamiento y mediante los signos (+) y (-) para el sentido del desplazamiento, lo que representa una mayor facilidad de operación del sistema.

Por último, también se implementó un tiempo de espera programable entre el movimiento de los motores con la intención de aumentar las aplicaciones de este sistema.

Para verificar la operación completa del sistema, se construyó una pieza mecánica de tres ejes que representan el control mínimo de movimiento que se pretendió. Al realizar las pruebas de operación, podemos decir que los objetivos planteados al inicio de nuestro trabajo se cumplieron en su totalidad.

BIBLIOGRAFIA

1. Intel, "8-Bit Embedded Controllers", Intel Corporation, Santa Clara, EUA, 1990.

2. B. Hawkes, "Cadcam", Paraninfo, Madrid, España, 1989.

3. Roger S. Pressman & John E. Williams, "Numerical Control & Computer Aided Manufacturing", John Wiley & Sons Inc., EUA, 1977.

4. Douglas V. Hall, "Microprocessors and Digital Systems", Second Edition, International Student Edition, Mc Graw-Hill.

5. Graham & Field, "IBM/PC. Guía del IBM PC (DOS 2.0) y XT", Segunda Edición, Osborne/Mc Graw-Hill, México, 1990.

6. Microsoft, "MS-DOS. User's Guide ans User's Reference", Microsoft Corporation, EUA, 1987.

7. Microsoft, "GW-BASIC. User's Guide and User's Reference", Microsoft Corporation, EUA, 1987.

8. RCA, "SK Series Solid State Replacement Guide", RCA Corporation, Nueva Jersey, EUA, 1986.

9. National Semiconductor, "LS/S/TTL Logic Databook", National Semiconductor Corporation, Santa Clara, EUA, 1987.

10. Malvino, "Principios de Electrónica", Segunda Edición, Mc Graw-Hill, México, 1988.

11. Deschamps y otros, "Apuntes de Computadoras y Programación", FI UNAM, México, 1985.