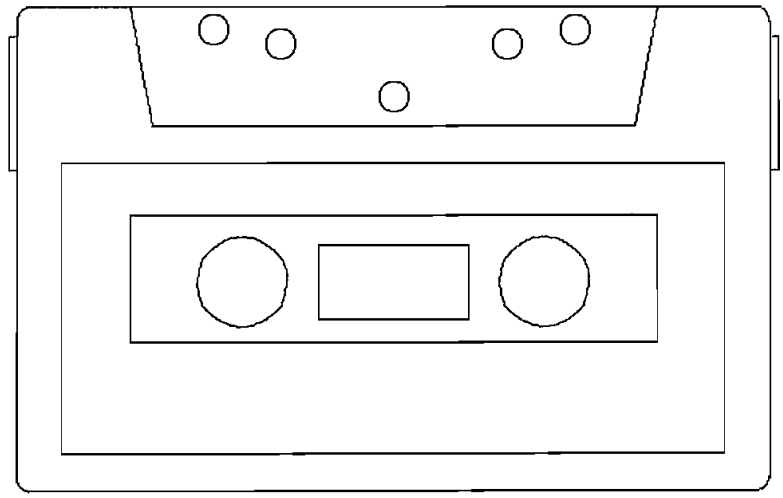
ESCUELA POLITECNICA NACIONAL

"SIMULACION POR HARDWARE DE UN STM"
(SELECTOR DE TEMAS MUSICALES)



POR

RAMIRO ALEJANDRO ROBAYO VASCO

TESIS EN LA ESPECIALIZACION DE

INGENIERIA ELECTRONICA Y TELECOMUNICACIONES

QUITO, AGOSTO DE 1994

CERTIFICO QUE EL PRESENTE TRABAJO HA SIDO ELABORADO EN SU TOTALIDAD POR EL SR. RAMIRO ALEJANDRO ROBAYO VASCO.

ING. ERWIN BARRIGA

DIRECTOR DE TESIS

DEDICATORIA

A MI PADRE, JORGE A. ROBAYO R.

A MI MADRE, MARIANA VASCO S.

TEAORRAMWCYLYIAEFNMUFETXDAUART

AGRADECIMIENTO

A mi Director de Tesis Ing. Erwin Barriga por su inmesurable paciencia y apoyo en los años de realización del presente trabajo. A todos mis maestros y de modo especial al Ing. Bolivar Ledesma, quienes fueron la fuente de mis conocimientos.

A mis hermanos; Martha G, Victor Hugo, Galo Fernando, Jorge Eloy, Juan Rodolfo, Anita, María Helena, Mercedes.

A mi amigo, Christian Foshl por su perseverancia en la contrucción del sistema mecánico del STM. Y a todas las personas que de uno u otro modo han colaborado, personas como Marcelo Patricio, C. Sylvia, Luis Fernando, P. Anibal, S. Justino S, Martha Torres.

INDICE

CAPITULO II

HARDWARE DEL STM

CAPITULO III

SOFTWARE DEL STM

INDICE DE LAS FIGURAS

INTRODUCCION

La idea original del presente trabajo fue la construcción completa de un aparato que sea capaz de escoger un casete elegido por el usuario y permitir escuchar los temas grabados en él, dicho aparato se lo conocerá con el nombre de SELECTOR DE TEMAS MUSICALES (STM). La principal razón para el uso de casetes, es optimizar su aplicación puesto que poseen características idénticas a las de un disco, con la clara diferencia en costo y versatilidad.

La falta de precisión al construir el sistema mecánico de este aparato condujo a reiterados y frustrados intentos por lo cual la idea original tuvo que ser modificada.

El objetivo del presente trabajo se limita a la simulación por hardware del STM, equipo concebido como parte de la idea original. Se lo ha orientado de modo que la simulación tienda a ser lo más real posible.

Dado que varios datos y señales tienen que ser introducidas por el usuario, éste debe estar informado sobre sus características para poder utilizarlo.

La presente tesis se encuentra dividida en 4 capítulos. En el primero se describe la idea general del funcionamiento mecánico del STM. No se detallan dimensiones pues algunas de las piezas mecánicas no fue posible construirlas.

En el segundo capítulo se analizan los circuitos implementados, es decir todo el hardware. Existe un circuito principal y varios auxiliares.

En el tercer capítulo se detalla el proceso de elaboración del software, el programa principal y las subrutinas con sus respectivos diagramas de flujo.

En el último capítulo se da las instrucciones necesarias para el ingreso de datos y la manera en que el usuario debe simular las señales que producirían los movimientos mecánicos no implementados. La simulación se realiza en forma interactiva entre el simulador y el usuario.

El microcontrolador utilizado es el 8751H, cuyas características se describen en el tercer capítulo.

CAPITULO I

SISTEMA MECANICO DEL STM

INTRODUCCION

En este capítulo se describe el funcionamiento mecánico del STM. Ante todo se recalca el hecho que la siguiente explicación no es completamente teórica, pues gran parte del sistema mecánico se lo construyó pese a que los resultados fueron parcialmente satisfactorios.

Aquí no encontramos las especificaciones necesarias como para construir el sistema mecánico del STM únicamente se encuentra la idea general de su funcionamiento mecánico.

La explicación que se da carece de un argot mecánico sin que por ello se la pueda considerar insuficiente como para no comprender el funcionamiento del STM.

1.1 ESPECIFICACIONES DEL STM.

El STM (SELECTOR DE TEMAS MUSICALES) es un aparato que posee gran semejanza con lo que nosotros conocemos como una rockola. La diferencia radica en que una rockola normalmente trabaja con discos, el STM trabaja con casetes. Estos casetes pueden ser de 60 ó 90 minutos.

Cuando en una rockola se realiza una selección se está eligiendo un lado de un cierto disco, en cambio cuando en el STM se realiza una selección se está eligiendo un lado de un casete, es decir todos los temas grabados en ese lado.

El STM puede contener hasta 120 casetes. En cada casete se pueden realizar 2 selecciones, el lado "A" ó el "B", es decir una selección corresponde a un lado de un casete. Por lo tanto podemos encontrar hasta 240 selecciones.

A través de un teclado el usuario puede realizar las selecciones de su preferencia. Cuando el usuario hace una selección está eligiendo música para 30 minutos (si se utiliza casetes de 60 minutos).

El usuario puede realizar hasta 5 selecciones, el STM no acepta ni una sola selección más mientras no se haya terminado de escuchar la primera selección realizada, momento en el cual queda un espacio libre para poder realizar una selección más.

Una vez que se ha escuchado una selección la cinta del casete es devuelta a su posición original (rebobinada al lado "A" del casete) a continuación el casete es devuelto a su posición física original.

El STM posee 2 grabadoras (G0 y G1), esto permite aumentar la velocidad de respuesta puesto que mientras un casete está siendo escuchado, en la otra grabadora ya tiene en stand-by el siguiente casete. De este modo cuando se terminó de escuchar una selección inmediatamente la otra grabadora permitirá escuchar la siguiente.

Para poder entender el sistema mecánico del STM realizaré una analogía con un plotter.

En el plotter tenemos desplazamientos en los 3 sentidos (X,Y,Z), el STM también necesita estos 3 desplazamientos. El movimiento del plotter en el sentido X corresponde al desplazamiento que el STM realiza entre columnas de casetes. El movimiento del plotter en el sentido Y corresponde al desplazamiento que el STM realiza entre filas de casetes. El plotter puede mover el lápiz hacia el papel o retirarlo de él, estos dos movimientos son similares a los que el STM tiene que efectuar para atrapar ó depositar un casete.

En el STM existen 2 mecanismos adicionales que permiten introducir o retirar el casete de una grabadora. Comenzaremos con definir la ubicación de los casetes.

1.2 UBICACION DE LOS CASETES. (Fig. 1.01)

Los 120 casetes que puede contener el STM, deben ser ubicados en 3 filas de 40 columnas. Los casetes tienen que ser colocados en posición vertical, con el lado más ancho del casete hacia arriba. Para cada casete existe un pequeño cajón de la mitad de la altura del casete, a cada uno de ellos en adelante se los denominará "CASILLEROS", se encuentran construidos sobre una base común.

Cada casillero posee una numeración la misma que será determinada por la columna y fila donde se encuentra. Las columnas, están numeradas desde 1 hasta 40 comenzando desde la izquierda. La primera fila tiene la numeración 0, la segunda 2 y la tercera 4, esta numeración un tanto extraña facilitará al momento de realizar el programa.

Cuando se haga alusión a un cierto casillero lo haremos de la siguiente manera, C(X,Y) donde "X" representa la columna e "Y" la fila. Por ejemplo:

C(37,2) Casilla de la fila 2 de casetes en la columna 37.
C(16,0) Casilla de la fila 1 de casetes en la columna 16.
C(40,4) Casilla de la fila 3 de casetes en la columna 40.
C(28,2) Casilla de la fila 2 de casetes en la columna 28.

Todos los casetes SIEMPRE se encuentran con su lado "A" hacia la izquierda y con el "B" hacia la derecha.

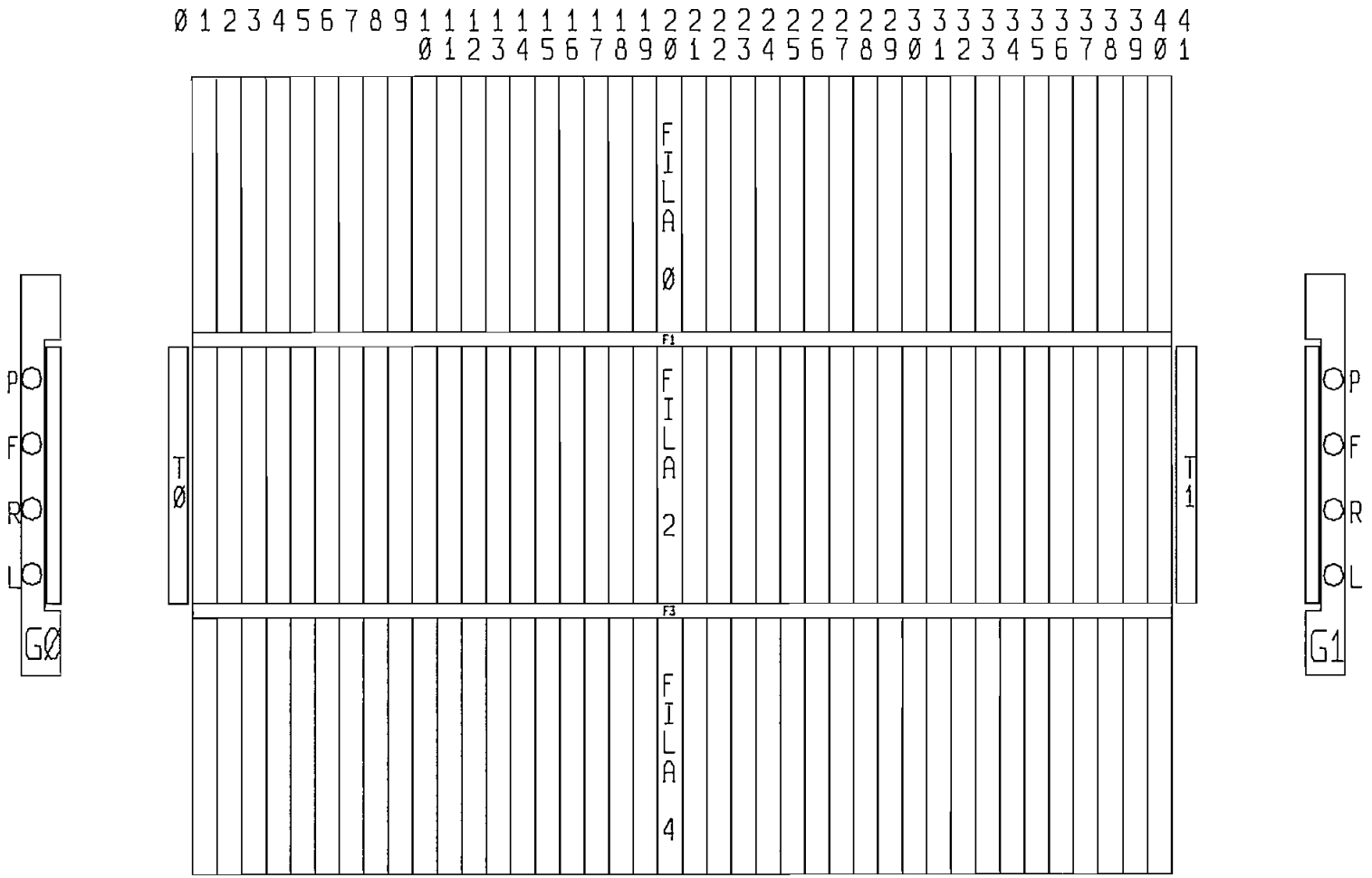

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
FILA 0
F1
FILA 2
F3
FILA 4
T0
T1
P
F
R
L
G0
G1

FIG. 1.01

1.3 UBICACION DE LAS GRABADORAS. (Fig. 1.01)

La misma figura utilizada para explicar la ubicación de los casetes permite representar las 2 grabadoras, G0 corresponde a la grabadora que se encuentra a la izquierda y G1 a la que se encuentra a la derecha.

G0 y G1, son del tipo reversible es decir, es posible ponerlas en PLAY tanto el lado "A" como el "B" del casete sin la necesidad de virar el casete. Cada una de estas grabadoras posee 4 funciones las mismas que son:

PLAY — Permite que los temas grabados en un casete sean escuchados, cuando el casete está siendo escuchado la cinta avanza desde el lado izquierdo hacia el lado derecho en esa grabadora.

FORWARD — Adelanta la cinta del casete que se encuentra dentro de esa grabadora, la cinta avanza desde el lado izquierdo hacia el lado derecho, este avance lo realiza a alta velocidad, estimándose el tiempo en 2 minutos para un casete de 60 minutos.

REWIND — Retrocede la cinta del casete que se encuentra dentro de la grabadora, la cinta va desde el lado derecho hacia el lado izquierdo. Realiza la función inversa a FORWARD. El tiempo estimado de rebobinaje es el mismo que para FORWARD.

LADO Esta función trabaja conjuntamente con PLAY y es la encargada de determinar el lado del casete que va a ser escuchado. Cuando LADO = ØL se escuchará el lado del casete que se encuentra hacia afuera de esa grabadora (lado visible).

Cuando LADO = 1L se escuchará el lado que se encuentra hacia el interior de la grabadora (lado no visible).

Las 2 grabadoras se encuentran frente a frente, por lo tanto para ubicar un casete en el interior de GØ lo hará desde el lado derecho y para ubicar un casete en el interior de G1 lo hará desde el lado izquierdo. Ninguna de las 2 grabadoras tiene la tapa que cubre el casete, esto facilita la ubicación del casete en la grabadora así como el retiro.

*** En esta parte resulta muy importante notar lo siguiente, en la última parte del literal 1.2 se dijo que los casetes <u>siempre</u> están con su lado "A" hacia la izquierda además sabemos que GØ se encuentra a la izquierda y G1 a la derecha. Por lo tanto, cuando deseamos escuchar una selección y ésta se encuentra en el lado "A" de un casete y debe ser colocado en GØ, debemos poner GØ en PLAY-LADO(1). Si ésta misma selección hubiese sido escuchada en G1 se hubiere tenido que poner G1 en PLAY-LADO(Ø). Esta diferencia se debe a que las 2 grabadoras no se encuentran exactamente en la misma posición sino que se están en posiciones simétricas, es decir frente a frente.

1.4 MECANISMO PARA UBICAR Y RETIRAR UN CASETE DE G0.

A más de las 40 columnas de casetes existen 2 posiciones adicionales que son la 0 y la 41.

En (0,2) existe 1 casillero móvil el mismo que es capaz de desplazarse hacia el interior de G0. Este casillero está formado por la tapa del casete de G0 (T0).

Al mecanismo que mueve a T0 desde (0,2) hasta el interior de G0 y viceversa lo denominamos MC0, trabaja exclusivamente para G0.

Cuando el STM tiene que ubicar un casete en el interior de G0 primero lo ubica en T0 (el mecanismo que realiza esto se lo verá más adelante), evidentemente cuando T0 se encuentra en (0,2) luego MC0 se encarga de transportarlo al interior de G0, posteriormente MC0 también es el encargado de retirar el casete de G0 y ubicarlo en (0,2) de donde será transportado a su posición original por otro mecanismo.

En la fig. 1.02 observamos el mecanismo MC0 junto con G0. Necesita de 2 guías y un motor DC (MOT-C0) para que T0 pueda ser movido.

Además son necesarios 2 sensores los cuales permitirán conocer cuando T0 está en el interior de G0 y cuando T0 se encuentra en la posición (0,2).

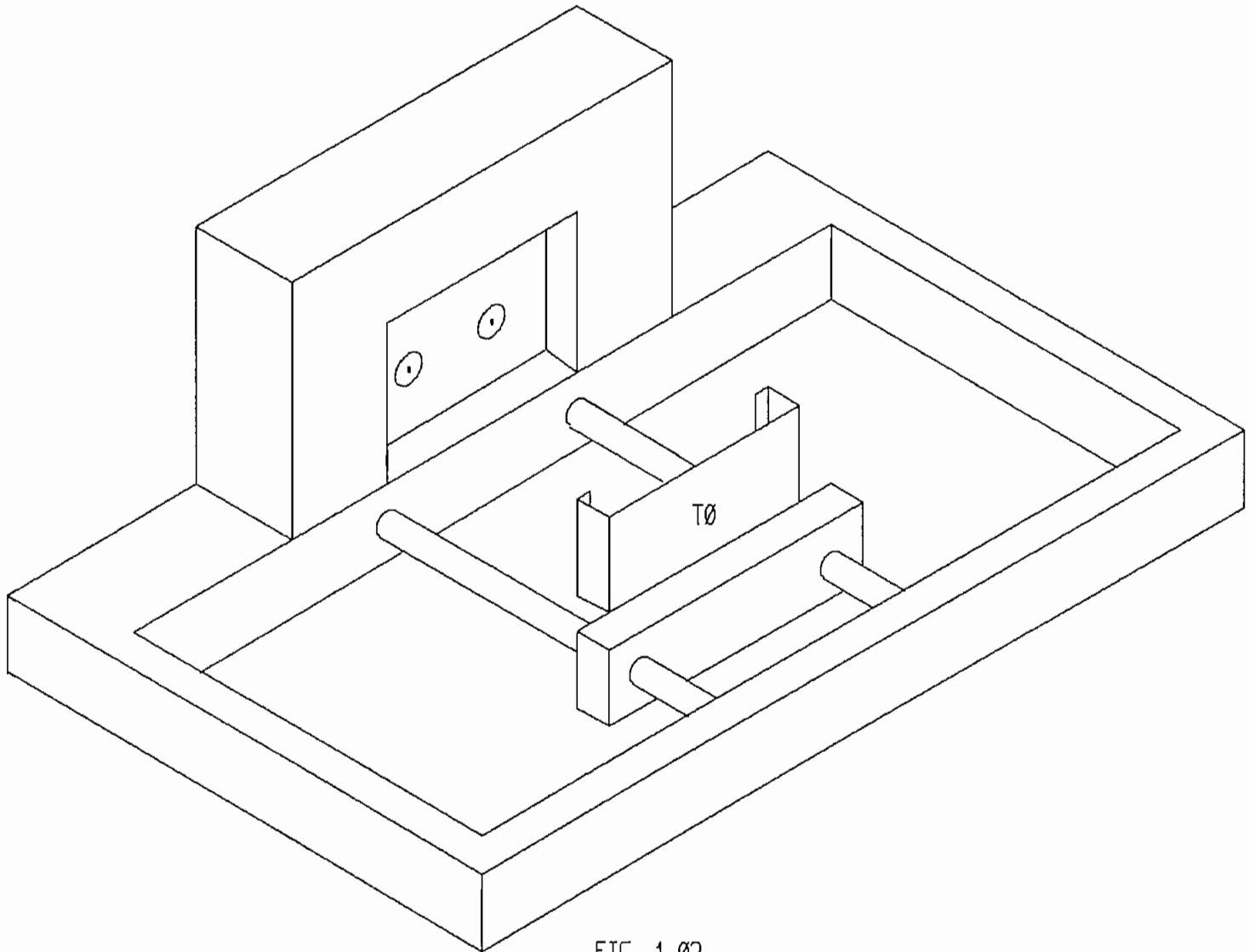


FIG. 1.Ø2

1.5 MECANISMO PARA UBICAR Y RETIRAR UN CASETE DE G1.

El funcionamiento del mecanismo para ubicar y retirar un casete de G1 es similar al mecanismo para ubicar y retirar un casete de G0. Podemos utilizar la misma fig. 1.02 para explicar su funcionamiento.

La posición 41 mencionada en el lit. 1.4 es la homóloga de la posición 0. En (41,2) existe 1 casillero móvil el mismo que es capaz de desplazarse hacia el interior de G1. Está formado por la tapa del casete de G1 (T1).

Al mecanismo que mueve a T1 desde (41,2) hasta el interior de G1 y viceversa lo denominamos MC1, trabaja exclusivamente para G1.

Cuando el STM tiene que ubicar un casete en el interior de G1 primero lo ubica en T1 (el mecanismo que realiza esto se lo verá más adelante), evidentemente cuando T1 se encuentra en (41,2) luego MC1 se encarga de transportarlo al interior de G1, posteriormente MC1 también será el encargado de retirar el casete del interior de G1 y ubicarlo en (41,2) desde donde será transportado a su posición original por otro mecanismo.

La semejanza entre MC0 y MC1 es bastante clara. La única parte en que difieren es que MC0 trabaja exclusivamente para G0 y MC1 para G1.

1.6 MECANISMOS PARA MOVER UN CASETE.

1.6.1 INTRODUCCION. Una vez que sabemos la forma en que se encuentran ubicados los casetes y las grabadoras, necesitamos de ciertos mecanismos capaces de atrapar un casete de una casilla C(Xo,Yo) y depositarlo en cualquier otra C(Xf,Yf), [incluidas C(0,2) y C(41,2)]. Este mecanismo debe moverse sobre cualquiera de estas 122 casillas que existen en el STM.

Para poder atrapar ó depositar un casete se necesita un movimiento tridimensional por lo tanto se requieren de 3 mecanismos cada uno de los cuales debe desplazarse en un determinado sentido (X,Y,Z).

En la fig.1.03, encontramos la infraestructura de estos mecanismos, la misma que se encuentra formada por un rectángulo hecho de ejes calibrados de 9 mm (varilla inoxidable perfectamente redonda), el cuál permitirá fijar sobre él varias piezas.

Dicho rectángulo se encuentra fijado a 10 cm del plano que forman los bordes superiores de los casetes. Es preciso destacar la precisión que debe existir al construirlo pues se precisa de un paralelismo perfecto entre los lados, un error superior a la décima de milímetro es suficiente como para que los mecanismos posteriormente añadidos a él trabajen de un modo defectuoso o simplemente no trabajen.

1.6.2 MECANISMOS X e Y (MX y MY). Sobre cada uno de los ejes EX se encuentra un anillo de bronce, los cuales son similares (AX). Los ejes EY son perfectamente paralelos entre si y se encuentran sólidamente unidos a AX.

Los anillos tienen que ser de bronce puesto que el contenido de carbono que en él existe brinda una lubricación permitiendo que el desplazamiento se torne suave entre EX y AX.

AX junto con EY se desplazan a lo largo de EX, este desplazamiento se produce por la acción de un motor DC (MOT-X) el mismo que los desliza por medio de poleas y bandas. Este movimiento corresponde al desplazamiento entre columnas (sentido X, hacia la izquierda ó hacia la derecha). EY siempre permanece en forma perpendicular a EX.

EX, AX, MOT-X y las poleas forman el mecanismo MX, el cuál desplaza a MZ entre las 42 columnas. En cada columna existe un sensor el mismo que permitirá el control de MX.

Aquí el principal problema fue la elaboración de AX, pues se requiere de una gran exactitud de modo que EY junto con AX puedan deslizarse con facilidad, tampoco deben quedar flojos pues se pierde la perpendicularidad entre EX y EY.

La falta de perpendicularidad se traduce en un incremento en el coeficiente de rozamiento.

Junto a EY existe una pieza en forma de un doble anillo (AY) la cuál se encuentra atravesada por el par de ejes EY. AY se desliza sobre EY. AY del mismo modo que AX debe ser construido de bronce.

Sobre uno de los 2 anillos AX se encuentra un motor DC (MOT-Y) que permite a AY deslizarse sobre EY, de este modo se hace posible el desplazamiento entre las 3 filas de casetes (sentido de las Y).

MY posee 5 sensores, 3 de los cuales determinan cuando EY se encuentra sobre una determinada fila de casetes, el otro sensor permite conocer cuando EY se encuentra justo a la mitad de la fila 1 y la 2. Y el último sensor se encuentra justo en la parte intermedia de la fila 2 y la fila 3 de casetes.

EY, AY, MOT-Y, las poleas y las bandas forman el mecanismo MY, este mecanismo realiza el desplazamiento entre las 3 filas de casetes y sus 2 filas intermedias.

1.6.3 MECANISMO Z (MZ). Sólidamente unido con AY está el mecanismo Z (MZ), el cual permite atrapar (depositar) un casete de (en) una determinada casilla. MZ completa el desplazamiento tridimensional. Se encuentra formado por un pinza semicerrada la misma que es capaz de bajar o subir por la acción de un motor DC (MOT-Z) que se encuentra incorporado en este mecanismo.

Cuando MZ baja la pinza semicerrada, el casete se ajusta por presión de la misma. Si en estas condiciones sube, el casete sube junto con la pinza semicerrada.

MZ tiene únicamente 2 posiciones arriba y abajo. En la parte superior existe un sensor (Za) el cual desconectará a MOT-Z cuando el mecanismo haya llegado al extremo superior. De modo análogo, existe un sensor en la parte inferior (Zb), el mismo que será el encargado de desconectar a MOT-Z.

De todo lo explicado podemos inferir que para atrapar un casete, primero mueve AX hasta la columna donde se encuentra dicho casete y AY hasta la fila correspondiente. Obviamente este par de movimientos implica que MZ se ha desplazado junto con AY y en este momento se encuentre exactamente sobre el casete buscado.

Luego, MZ desciende hasta su extremo inferior y posteriormente ascienda hasta su extremo superior junto con el casete deseado. Es decir que para atrapar el casete se ha requerido de un desplazamiento tridimensional.

Cuando se desea depositar un casete en una casilla el proceso es más complejo, primero MZ junto con dicho casete debe ser desplazado hasta la fila-columna correspondiente a la casilla donde se desea depositarlo. Acto seguido MZ desciende y en este instante el casete ya se encuentra en la casilla pero no está liberado de MZ.

Para que el casete sea liberado, MZ debe moverse hacia una posición intermedia de las filas. De este modo se está obligando a que el casete se quede en esa casilla. Luego de ello MZ asciende sin el casete.

El movimiento en el sentido Z nunca se realiza simultáneamente con otro movimiento. En cambio el movimiento en el sentido de X e Y pueden ser realizados simultáneamente.

1.7 EJEMPLO.

Asumamos que, MZ se encuentra sobre C(20,2) y que la primera selección que se realizó corresponde al casete que se encuentra ubicado en C(37,0). Puesto que es la primera selección el casete tiene que ser ubicado en G0.

AX tiene que desplazarse hasta la columna 37 y simultáneamente AY debe moverse hasta la primera fila de casetes, una vez realizados estos movimientos MZ se encuentra sobre C(37,0).

Luego MZ desciende hasta su extremo inferior, momento en el cual el casete queda atrapado en la pinza semicerrada posteriormente MZ asciende junto con el casete.

AX y AY posteriormente se desplazarán hasta C(0,2), esta posición se encuentra frente a G0 y el casillero que se encuentra allí es móvil según se explicó en 1.4.

Una vez que se encuentran en esta posición MZ desciende hasta cuando el casete esté ubicado en TØ.

Para que el casete se libere de MZ, AY tiene que desplazarse hasta una posición intermedia de filas y una vez que ha llegado hasta esa posición MZ asciende sin el casete. MCØ se encarga de introducir el casete en GØ.

Cuando el casete tiene que ser devuelto a su posición original, primero debe estar en C(Ø,2), luego MZ debe ser desplazado hasta dicha posición.

Acto seguido MZ desciende, el casete es atrapado. Luego MZ asciende y se mueve hasta C(37,Ø). Posteriormente MZ desciende y ubica el casete en la casilla original, el casete se encuentra en la casilla pero no está liberado de MZ.

Para poder liberarse, MZ tiene que moverse hacia la posición intermedia de las filas Ø y 2 es decir la fila 1, de este modo se obliga a que el casete se quede en C(37,Ø). Luego de ello MZ ascenderá ya sin el casete.

CAPITULO II

HARDWARE DEL STM

INTRODUCCION

En este capítulo se explica la necesidad de cada circuito, su funcionamiento y el objetivo de cada una de las partes que conforman el hardware del STM el mismo que se lo ha realizado de modo que su funcionamiento sea lo más simple, claro está que no es la única manera de hacerlo puesto que al diseñar algo siempre habrá futuras modificaciones que llevarán a una optimización tanto en el funcionamiento como a una reducción del material utilizado.

El hardware se encuentra formado por varios circuitos auxiliares y un circuito principal, algunos de estos circuitos auxiliares son seudoautónomos, la razón de llamarlos así se la verá más adelante.

2.1 CIRCUITO DE INGRESO DE DATOS Y DISPLAYS.

De modo similar a lo que sucede en una rockola, también para el STM surge la necesidad de ingresar datos. Para poder ingresar estos datos es necesario un teclado que contenga los 10 dígitos y 2 teclas adicionales, la primera tecla adicional permitirá borrar el último dígito ingresado y la otra almacenar en memoria dicha selección. Por lo tanto es necesario un teclado de 12 teclas, se utilizó uno de 16 teclas dejando a las 4 últimas sin ninguna función, evidentemente esto no representa ningún problema.

Junto al teclado se requiere un decodificador de teclado. El 74C922 es un decodificador de 16 teclas razón por la cual es útil para nuestro caso. Los datos que han sido ingresados tienen que ser almacenados y posteriormente procesados para que en base a esos resultados se realice el control de los distintos mecanismos. Por esta razón se torna indispensable la utilización de un microcontrolador. El microcontrolador elegido es el 8751H debido a su gran versatilidad y capacidad de memoria, a medida que se siga avanzando en el diseño se irá justificando el uso de dicho microcontrolador.

En la fig. 2.01 se puede observar el circuito constituido por el teclado, el decodificador de teclado, los displays, el microcontrolador y adicionalmente se observa el integrado 74154.

Este integrado es un decodificador de direcciones, permite controlar hasta 16 elementos utilizando únicamente 4 pines del microcontrolador. El decodificador de direcciones (I-34) presenta un ØL en una de sus salidas cuando dicha salida es especificada a sus entradas, las demás salidas presentarán un 1L.

Los dígitos que están siendo ingresados desde el teclado aparecerán en un grupo de 3 displays (D6, D5, D4). Se utiliza un segundo grupo de 3 displays (D3, D2, D1) los mismos que exhiben el código de la selección que se está escuchando. Es posible utilizar solamente 3 displays pero esto implica que cuando se hace una nueva selección, el código correspondiente a la selección que se está escuchando tiene que desaparecer unos momentos mientras se realiza una nueva selección. Estos 6 displays se encuentran controlados por las direcciones D6=C, D5=B, D4=A, D3=9, D2=8 y D1=7, estas direcciones son especificadas desde los 4 bits menos significativos del puerto 2 del microcontrolador. En la fig. 2.02 se observa el circuito de los displays.

Cuando no está ninguna tecla presionada las salidas del decodificador se encuentran en alta impedancia en cambio cuando una tecla es presionada en la salida DA (Data available o dato disponible, pin 12 del decodificador) se coloca 1L, este 1L pasa por un inversor (I-37D) obteniéndose un ØL, este ØL tiene dos propósitos; primero inhibir al decodificador en caso de que otra tecla sea presionada antes de haber

sido liberada la primera, y segundo producir la interrupción en el microprocesador (pin 12 del microprocesador). Para nuestro caso interrupción en el microprocesador se produce por transición negativa.

Existen 2 maneras de activar una interrupción en el 8751H, por transición o por estado. En la primera forma cuando existe una transición negativa la subrutina que corresponde a dicha interrupción se ejecuta una sola vez.

Cuando es activada por estado la subrutina correspondiente a esa interrupción se ejecuta una y otra vez mientras dure dicho estado. Se puede elegir el modo de activación por software.

Cuando la interrupción es activada se suspende la ejecución normal del programa y en ese momento el microprocesador lee el dato que está ingresando en los 4 bits más significativos de P2, una vez leído este dato continua con la ejecución del programa desde la parte donde se quedó antes de producirse la interrupción.

Para nuestro caso la interrupción que permite leer la tecla presionada es la única utilizada en todo el circuito. La tecla leída es exhibida (si es un dígito) en uno de los displays D6, D5, D4 siempre y cuando dicha tecla sea parte de un dato válido, las condiciones para que un dato sea válido se las verá más adelante.

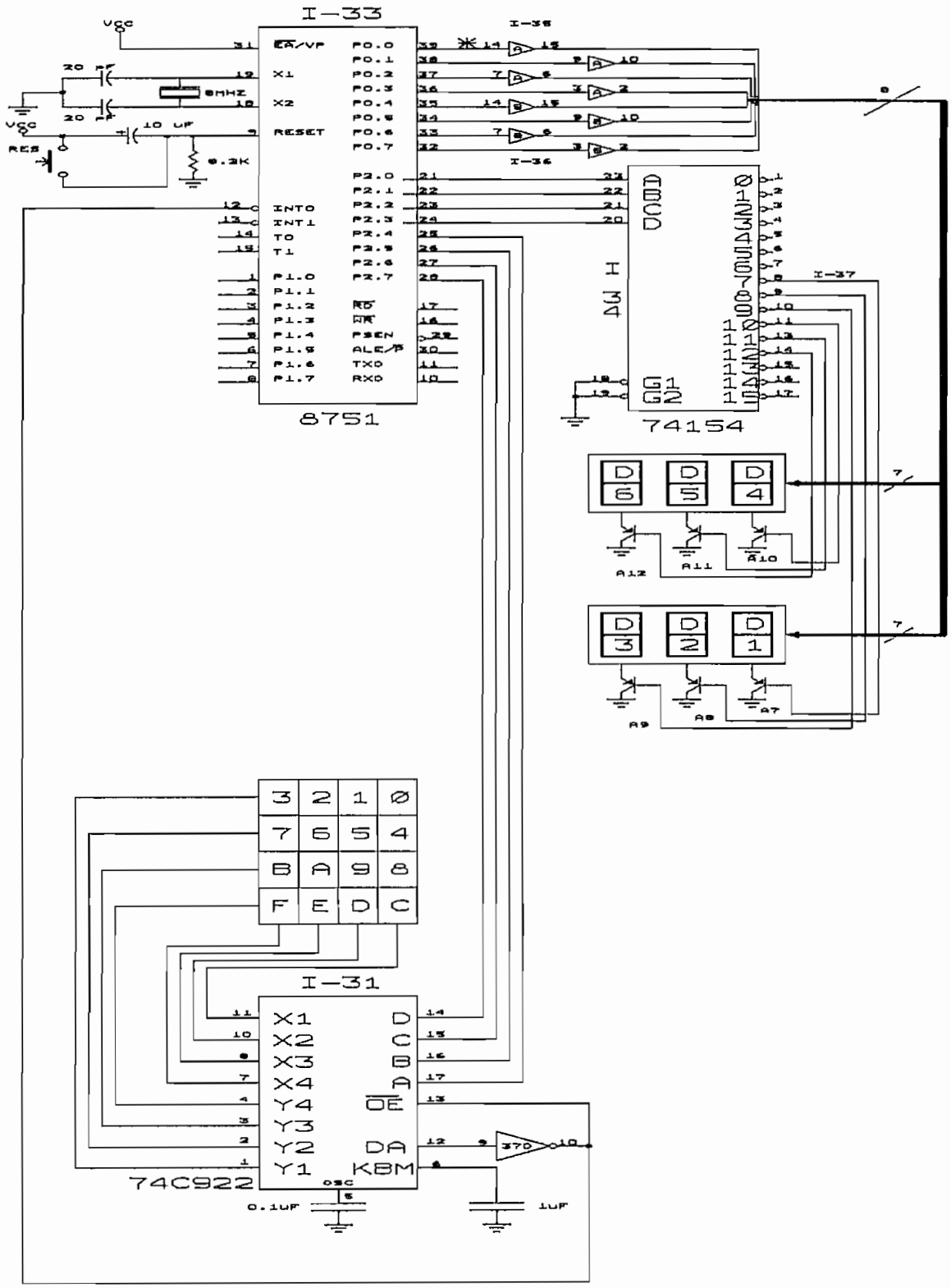
El puerto 0 y los 4 bits menos significativos del puerto 2 del microcontrolador trabajan sincronizadamente. El puerto 0 es lo que en adelante lo conoceremos como BUS DE DATOS y es el encargado de llevar la información a los displays y a los circuitos auxiliares (que serán explicados más adelante). Los 4 bits menos significativos del puerto 2 (junto con el decodificador de direcciones) en cambio es el encargado de señalar que elemento va hacer uso de los datos que en ese momento se encuentran en el BUS DE DATOS (P0).

La frecuencia de muestreo de los displays, es decir la velocidad que un dato permanece en un display es de 1KHz. La corriente que circula en un segmento en el momento de ser muestreado es de 60 mA, puesto que son 6 displays la corriente promedio en cada segmento es de 10 mA.

Dado que el microcontrolador 8751H es el elemento principal de todo el circuito de control del STM, se describen sus principales características.

- 4K bytes de memoria UVEPROM para el programa.
  Se puede expandir hasta 64K bytes con memoria externa.
- 128 bytes de memoria RAM, se usa para variables.
  Expandible a 64K bytes con memoria externa.
- Frecuencia de trabajo entre 3,496503 y 12.004801 Mhz.
- Posee 4 puertos de 8 pines, todos son bidireccionales.
- Voltaje de polarización, 5 VDC.
- El voltaje de programación es de 21 VDC.

CIRCUITO DE INGRESO DE DATOS Y DISPLAYS



- TODOS LOS PINES DE P0 TIENEN RESISTENCIAS DE PULL-UP

FIG. 2.01

CIRCUITO DE LOS DISPLAYS

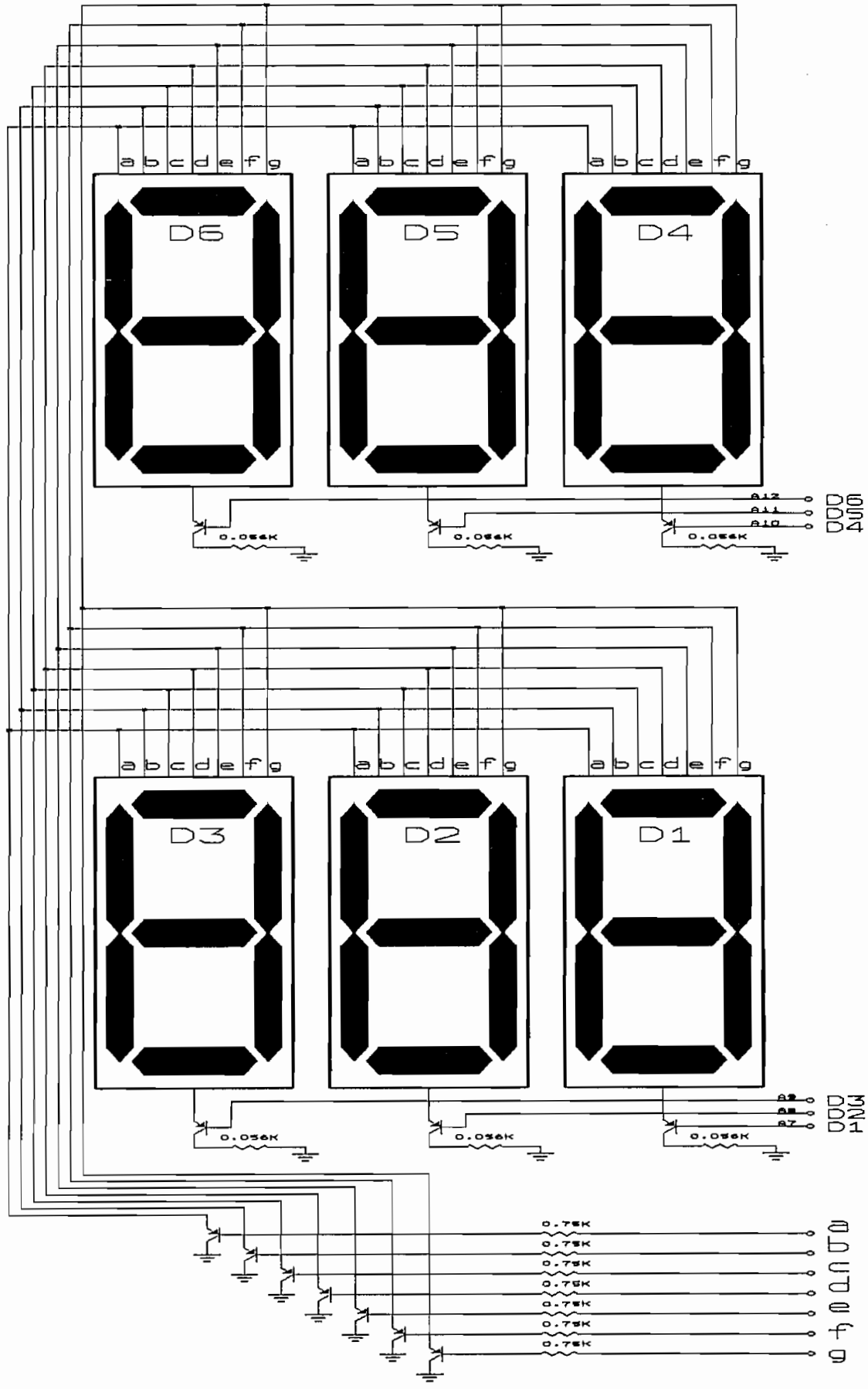


FIG. 2.02

2.2 CIRCUITO MX. (Fig. 2.03 y 2.04).

En el literal 1.6.2 fue descrito el mecanismo MX que es el encargado de permitir el desplazamiento entre las 42 posiciones que existen en el sentido X. Dicho mecanismo posee 1 solo motor DC, este motor es controlado de la siguiente manera:

El microcontrolador se encarga de ponerlo en movimiento ya sea en sentido o en otro (dependiendo de la posición de la que parte y la posición a la cual debe llegar), además especifica la posición hacia la que debe moverse escribiendo en los 6 bits menos significativos la posición hacia la que debe moverse.

Un circuito auxiliar es el encargado de detener al motor cuando haya llegado a la posición solicitada por el microcontrolador, para este fin son utilizados 42 sensores, 1 para cada posición.

Este circuito auxiliar utiliza los siguientes integrados:

I-1 a I-7 74LS151(TTL) Mux 3/8, strobe, salidas Y e Y'.
I-8 74SL374(TTL) Latch tipo D de 8 bits.
I-9 4081(CMOS) 4 compuertas AND de 2 entradas.
I-10 4049(CMOS) 6 inversores.
I-11 4081(CMOS) 4 compuertas AND de 2 entradas.
I-12 4081(CMOS) 4 compuertas OR de 2 entradas.

CIRCUITO DE CONTROL DEL MOTOR X
(PRIMERA PARTE)

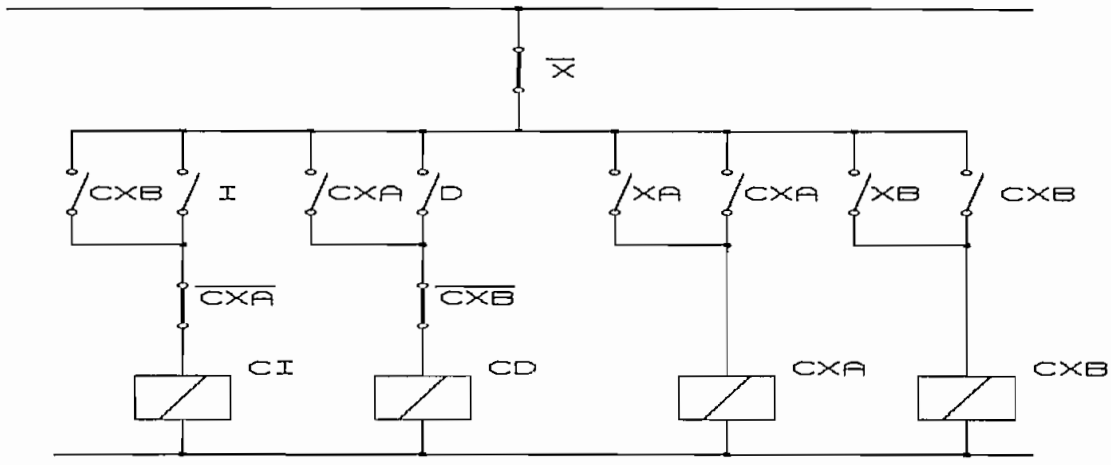


FIG. 2.03a

$$CI = \overline{X}.\overline{CXA}.(CXB+I)$$
$$CD = \overline{X}.\overline{CXB}.(CXA+D)$$
$$CXA = \overline{X}.(XA+CXA)$$
$$CXB = \overline{X}.(XB+CXB)$$

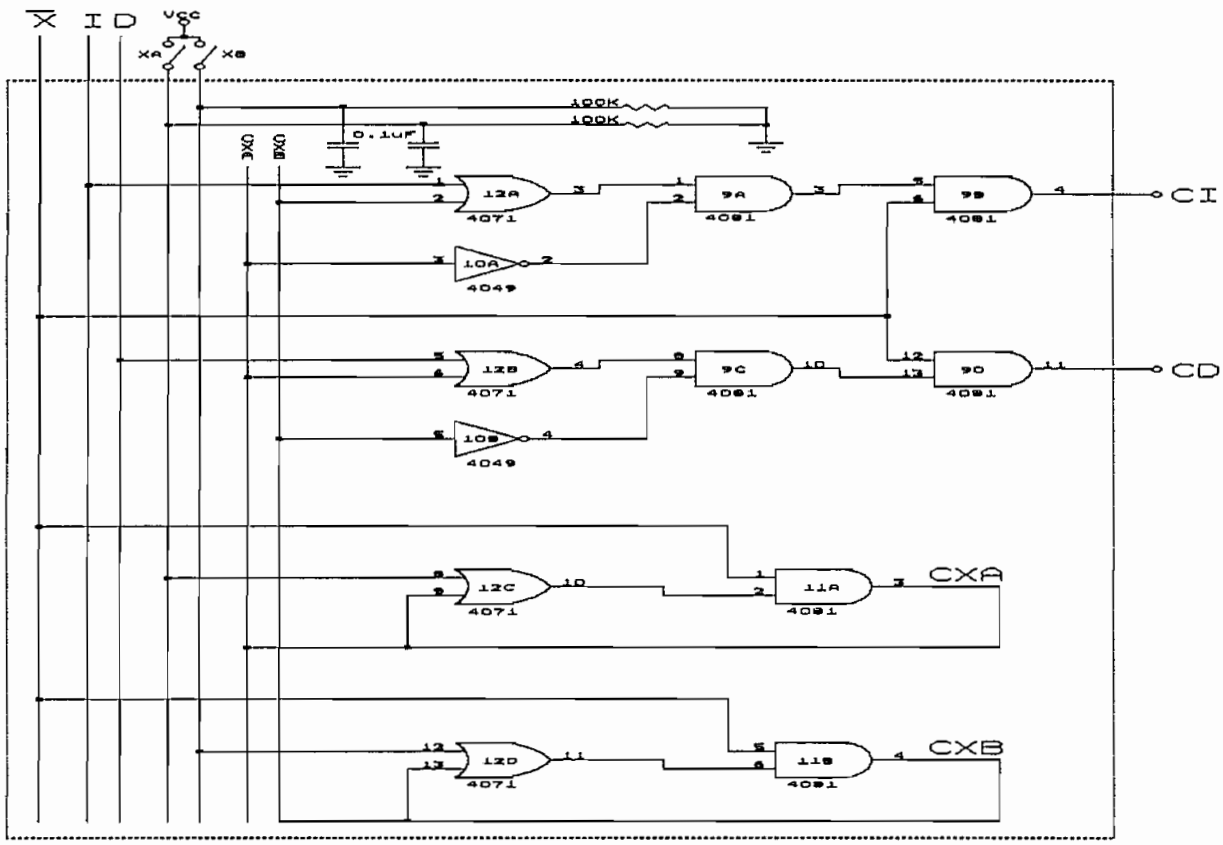
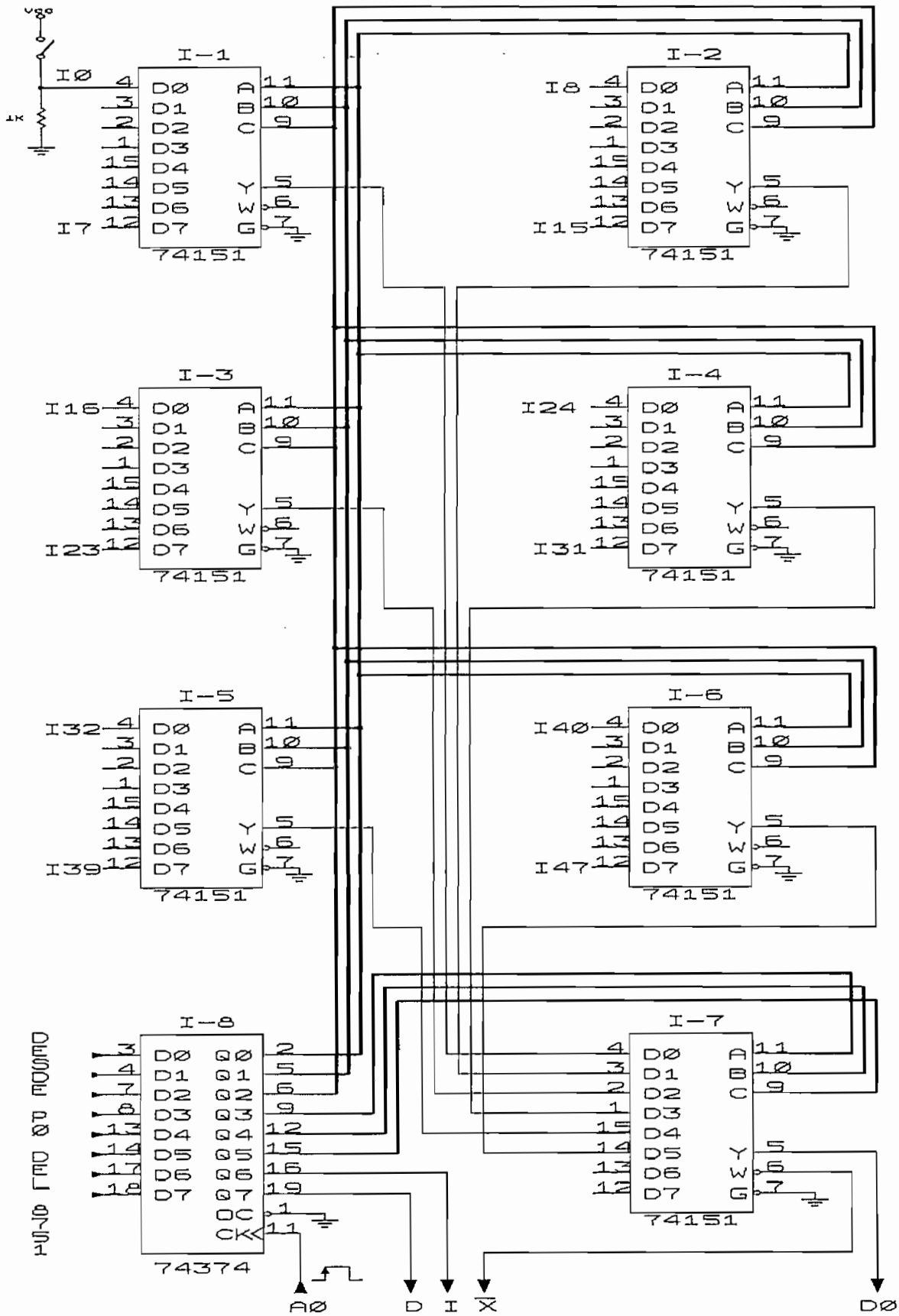


FIG. 2.03b

CIRCUITO DE CONTROL DEL MOTOR X

(SEGUNDA PARTE)



NOTA: I1 hasta I41 similar a I0

FIG. 2.04

En la fig. 2.03 podemos ver que se necesita una señal X', ya que esta señal es importante, primero analicemos de donde procede.

En la fig. 2.04 existen 8 integrados, 7 de los cuales son 74LS151 (multiplexers, desde I-1 hasta I-7) y el otro es un 74LS374 (latch, I-8). Los multiplexers poseen 8 entradas principales, 3 entradas de selección las cuales permiten enrutar una de las 8 entradas principales hacia la salida Y (la salida W es la misma Y pero complementada). Además existe una entrada G (strobe) la misma que para nuestro caso estará permanentemente conectada a ØL.

Las entradas de selección de los 6 primeros multiplexers se encuentran conectadas en paralelo, es decir cuando deseamos que una cierta entrada se enrute hacia la salida de su respectivo multiplexer estamos haciendo exactamente lo mismo en los 6 multiplexers. Las salidas de estos 6 multiplexers son las entradas del séptimo multiplexer (I-7), necesitamos 3 entradas más de selección para poder enrutar una de estas 6 entradas de I-7, en total hemos necesitado 6 entradas de selección.

De este modo hemos obtenido el equivalente de un solo multiplexer de 48 entradas con 6 líneas de selección y una salida Y (y su salida complementada, W). De estas 48 entradas tan solo utilizamos 42, en adelante me referiré como el multiplexer 6/42 (6 líneas de selección, 42 entradas).

A cada columna le corresponde un sensor magnético y una entrada del multiplexer 6/42. Los sensores envían un 1L cuando el motor ha movido al mecanismo a esa posición, este 1L ingresa a una de las entradas del multiplexer 6/42, cuando pasa por una posición que no corresponde a la que es enrutada, la salida en el multiplexer 6/42 es ØL. Por el contrario cuando se encuentra en la posición que es enrutada, la salida en el multiplexer 6/42 es 1L. La salida X' a la que hacemos referencia en la fig. 2.Ø3 es la salida complementada del multiplexer 6/42, la salida que no es complementada de este multiplexer corresponde a DØ. Cuando DØ es 1L implica que MX ha llegado a la posición especificada. En la figura 2.Ø4 tenemos un circuito en la entrada IØ, este circuito permite enviar un ØL cuando no se encuentra en dicha posición, puesto que se trata de circuitos TTL no podemos dejar ninguna entrada abierta pues lo interpretaría como un 1L. Un circuito similar al de IØ es implementado en todas y cada una de las 42 entradas.

El latch (I-8) se encuentra internamente formado por 8 biestables tipo D, por lo tanto posee 8 entradas y 8 salidas, además dispone de la señal OC (output control, control de las salidas) y la señal de reloj que será la encargada de permitir el paso de las entradas hacia las salidas cuando aparezca una transición positiva.

Cuando OC = 1L todas las salidas están en alta impedancia, en nuestro caso tendremos permanentemente OC = ØL.

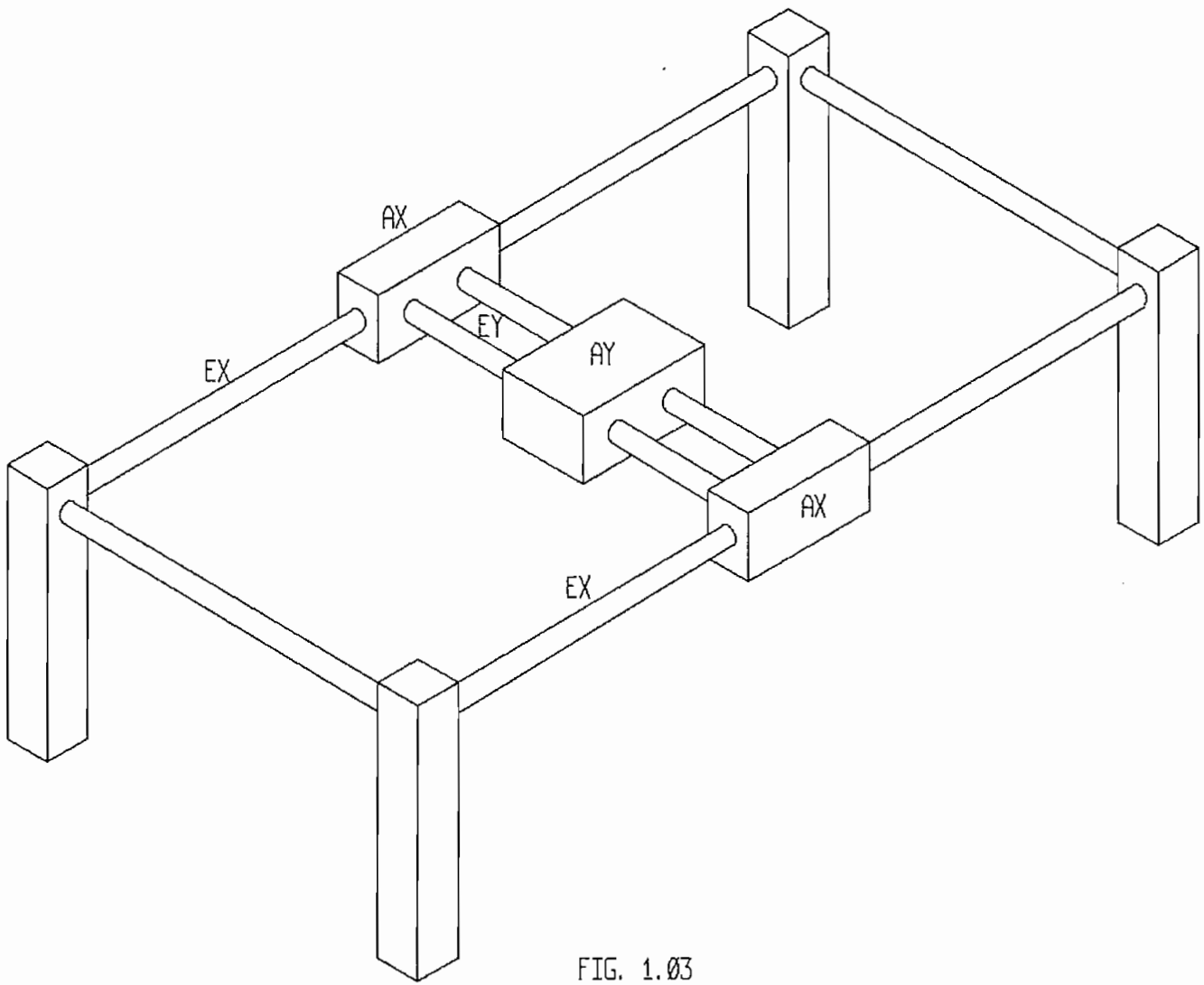


FIG. 1.Ø3

Los 8 bits del bus de datos entran en el latch (I-8), estos datos no estarán presentes en la salida de este latch mientras no aparezca la señal A0 (señal de reloj que aparece cuando en los 4 bits menos significativos de P2 se escribe 0000). De estos 8 bits los 6 primeros son utilizados como entradas de selección del multiplexer 6/42 para así poder enrutar una de las 42 entradas. Los 2 últimos bits son utilizados para dar el sentido con el cual comenzará a moverse el motor que controla el desplazamiento en el sentido X, el último bit indica que debe moverse a la derecha y el penúltimo a la izquierda.

Al tener 42 posiciones en el sentido X por las cuales puede moverse, la posición inicial o de reposo será la posición 20 es decir en el centro, por medio de probabilidades es posible demostrar que esto es necesario para poder reducir el tiempo de respuesta.

P0 = 0001 0100B Posición inicial de MX.

P2 = 1111 0000B Señal de activación de MX.

En la fig. 2.03a podemos observar el circuito de control implementado con relés, CI y CD dan el sentido de giro al motor hacia la izquierda o hacia la derecha (bit 6 y bit 7 del bus de datos), cuando la señal I (P0.6) aparece activa CI obligando a que el mecanismo MX se mueva a la izquierda, solo cuando la señal X' desaparece, MX se detendrá caso contrario no se detendrá a menos que llegue al extremo

izquierdo y active el sensor Xa, este sensor desconecta a CI y conecta a CD obligando a que se mueva hacia la derecha.

En esta nueva condición sucede algo similar, si no se desactiva por medio de la señal X' llegará a ser desactivado por medio de Xb que es un sensor que se encuentra al extremo derecho. En otras palabras, podemos decir que el mecanismo es puesto en marcha y solamente se detiene por medio de la señal X', si esta señal no apareció hasta cuando llegue al extremo MX cambiará de sentido y pasará revisando todas las posiciones y si después de haberlas revisado no ha aparecido la señal X' el motor se desactivará completamente.

En condiciones normales los sensores Xa y Xb no deberían ser activados jamás pues esto implicaría que la posición solicitada no fue encontrada, máximo puede suceder que uno de los 2 sensores sea activado y eso solo en el momento del encendido del STM. La desconexión de un modo normal es por medio de X', por lo tanto Xa y Xb se encuentran exclusivamente por seguridad. La fig. 2.03b es exactamente el mismo circuito que en 2.03a pero implementado con compuertas, el circuito implementado es el de la fig. 2.03b.

Resumiendo lo expuesto sobre MX, la posición inicial en el sentido de X es la posición 20, la posición a la cual debe moverse es escrita en los 6 bits menos significativos de P0, los 2 bits más significativos dan el sentido en que debe girar el motor para ir a buscar esa posición.

Una vez que el dato está escrito en P0 debe ser activado el latch por medio de P2. XA y XB han sido colocados exclusivamente por seguridad. A continuación se dan 2 ejemplos los mismos que permitirán aclarar la explicación precedente.

Ej. 2.2a Si parte de la posición inicial (20) y tiene que moverse a la posición 35 (100011B), implica que tiene que desplazarse hacia la derecha por lo tanto el dato que el microprocesador tiene que escribir en el bus de datos (P0) tiene que ser 10i00011B.

Los 6 bits menos significativos de este dato obligan al multiplexer a enrutar la entrada 35, por lo tanto cuando pase por las demás posiciones el dato de salida del multiplexer será 0L (D0), sólo cuando llegue a la posición 35 el dato será 1L (D0) entonces en ese momento tendremos que X'=0, obligando a que el MX se detenga.

Ej. 2.2b Si MX parte de la posición 41 y debe moverse a la posición 5 (000101B), implica que tiene que desplazarse hacia la izquierda por lo tanto el dato que el microprocesador pondrá sobre el bus de datos (P0) P0 tendrá que ser 01000101B. Los 6 bits menos significativos de este dato obligan al multiplexer a enrutar la entrada 5, por lo tanto cuando pase por las demás posiciones el dato de salida del multiplexer será 0L (D0), sólo cuando llegue a la posición 5 el dato será 1L (D0) en ese momento tendremos X'=0, obligando a que el MX se detenga.

2.3 CIRCUITO MY. (Fig. 2.05 y 2.06).

En la sección 1.6.2 junto con el mecanismo MX fue descrito el mecanismo MY, este mecanismo es el encargado de permitir el desplazamiento entre las filas de casetes. Funciona de un modo similar que MX excepto que su desplazamiento es en el sentido de las Y que el número de posiciones diferentes es 5, 3 de las filas de casetes numeradas con 0, 2 y 4 y 2 posiciones intermedias entre las filas de casetes numeradas con 1 y 3, en cada una de estas posiciones existe un sensor magnético. MY posee un solo motor DC (doble sentido de giro) y su puesta en marcha y especificación a la que debe moverse le corresponde al microcontrolador se detiene cuando haya llegado a la posición solicitada.

Para detenerse requiere de un circuito auxiliar, es decir se mueve por instrucción del microcontrolador y se detiene por la acción del circuito auxiliar, esta es la causa para que se denomine seudoautónomo.

El circuito auxiliar utiliza los siguientes integrados:

I-20 74LS151(TTL) Mux 3/8, strobe, salidas Y e Y'(W).
I-23 74SL374(TTL) Latch tipo D de 8 bits. Trans. pos.
I-18 4081(CMOS) 4 compuertas AND de 2 entradas.
I-22 4049(CMOS) 6 inversores.
I-19 4081(CMOS) 4 compuertas AND de 2 entradas.
I-21 4081(CMOS) 4 compuertas OR de 2 entradas.

En la fig. 2.05 podemos ver que se necesita una señal Y'. Puesto que está señal es bastante importante analicemos primero de donde procede antes de explicar el funcionamiento del circuito representado en la fig. 2.05.

En la fig. 2.06 tenemos 1 multiplexer que posee 8 entradas principales, 3 entradas de selección, 1 entrada G (strobe) y 2 salidas complementadas entre si.

Las 3 entradas de selección permiten enrutar una de las 8 entradas principales hacia la salida Y (la salida W es la misma Y pero complementada). Para nuestro caso la señal G estará conectada permanentemente a 0L. Por ser solo 5 de las 8 entradas que se ocupa me referiré en adelante como el multiplexer 3/5.

Cada una de estas entradas corresponde a una posición en el sentido Y de MZ. En cada una de estas posiciones existe un sensor que cuando es activado envía un 1L, este 1L ingresa a una de las entradas del multiplexer 3/5, cuando el MZ pasa por una posición que no corresponde a la que está siendo enrutada la salida en el multiplexer 3/5 es 0L. Solamente cuando se encuentra sobre la posición que es enrutada la salida en el multiplexer 3/5 es 1L. La salida Y' a la que hacemos referencia en la fig. 2.05 es la salida complementada del multiplexer 3/5, la salida que no es complementada de este multiplexer corresponde a D1. Cuando D1 sea 1L implicará que MY se encuentra sobre la posición pedida.

CIRCUITO DE CONTROL DEL MOTOR Y

(PRIMERA PARTE)

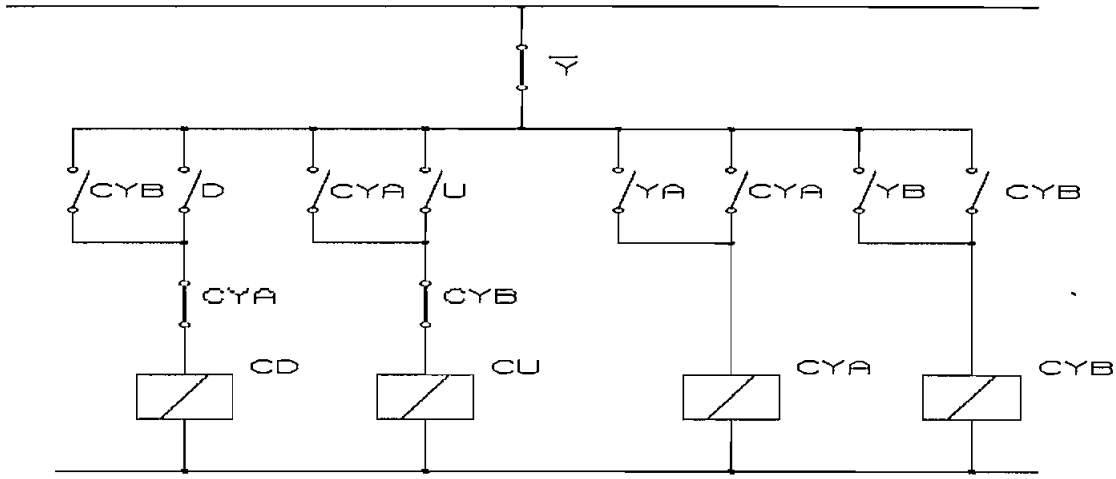


FIG. 2.05a

$$CD = \overline{Y}.\overline{CYA}.(CYB+D)$$
$$CU = \overline{Y}.\overline{CYB}.(CYA+U)$$
$$CYA = \overline{Y}.(YA+CYA)$$
$$CYB = \overline{Y}.(YB+CYB)$$

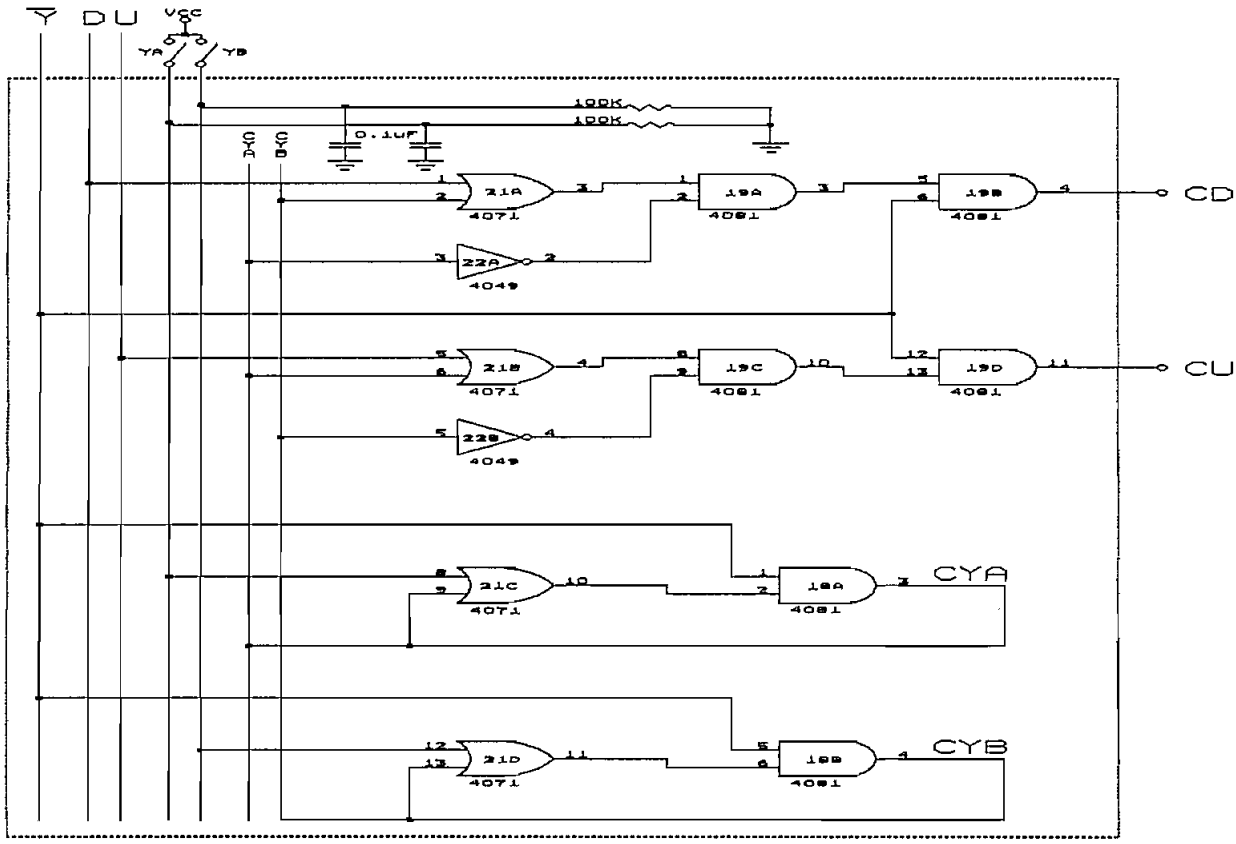


FIG. 2.05b

CIRCUITO DE CONTROL DEL MOTOR Y

(SEGUNDA PARTE)

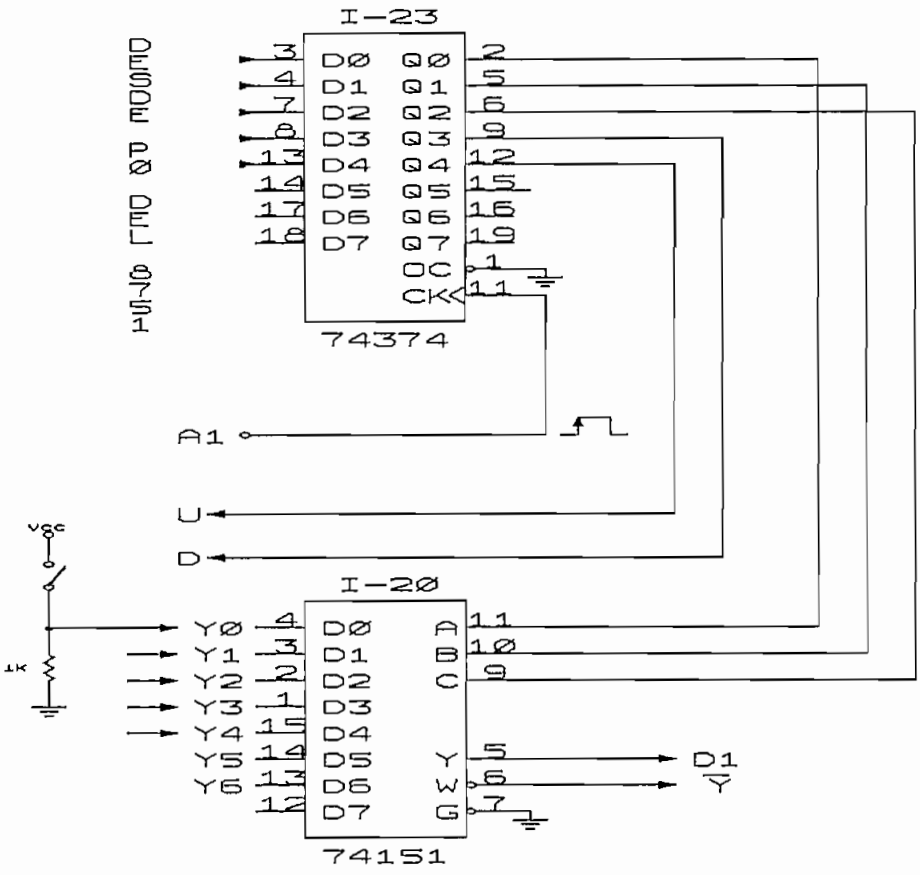


FIG. 2.06

En la figura 2.06 vemos un circuito en la primera entrada del multiplexer, este circuito permite enviar un ØL cuando no se encuentra en dicha posición, la razón de este circuito fue incluída en la explicación de MX. Un circuito similar a este es implementado en todas y cada una de las 5 entradas.

El latch (I-23) es de las mismas características que el utilizado para MX.

Los 8 bits del bus de datos entran en el latch, estos datos no estarán presentes en la salida de este latch mientras no aparezca la señal A1. De estos 8 bits se utilizan 5, los 3 primeros son las entradas de selección del multiplexer 3/5, el bit 3 correspondiente a PØ.3 y el bit 4 a PØ.5 son utilizados para dar el sentido con el cual comenzará a moverse el motor que controla el desplazamiento de MZ en el sentido Y, el bit 3 indica que MY debe moverse hacia UP y el bit 4 hacia DOWN.

La posición inicial o de reposo de MZ en el sentido Y es la posición 2 es decir en el centro, esto reduce el tiempo de respuesta del aparato dado que si encuentra en el centro la distancia que tiene que desplazarse hacia una de las otras 2 filas esta minimizada.

PØ = ØØØØ ØØ1ØB Posición inicial de MY.

P2 = 1111 ØØØ1B Señal de activación de MY.

En la fig. 2.05a se ve el circuito de control del motor implementado con relés, CU y CD dan el sentido de giro al motor hacia UP o hacia DOWN, cuando la señal D (PØ.4) aparece activa CD obligando a que el mecanismo MY se mueva hacia DOWN, solo cuando la señal Y' desaparece, MY se detendrá caso contrario no se detendrá a menos que MY llegue al extremo inferior y active el sensor Yb, este sensor desconecta a CD y conecta a CU.

En esta nueva condición sucede algo similar a lo que estaba sucediendo cuando MY se estaba moviendo hacia abajo, si no se desactiva por medio de la señal Y' llegará a ser desactivado por medio de Xa que es un sensor que se encuentra al extremo superior. En otras palabras, podemos decir que el mecanismo es puesto en marcha por el microcontrolador y solamente se detiene por medio de la señal Y', si esta señal no apareció hasta cuando llegue al extremo, MY cambiará de sentido y pasará revisando todas las posiciones y si después de haberlas revisado no ha aparecido la señal Y' el motor se desactivará completamente.

En condiciones normales de funcionamiento los sensores Ya y Yb no deberían ser activados jamás pues esto implicaría que la posición solicitada no fue encontrada, como máximo puede suceder que uno de los 2 sensores sea activado y eso solo en el momento del encendido del STM. La desconexión de un modo normal es por medio de Y', por lo tanto Ya y Yb se encuentran exclusivamente por seguridad.

En la fig. 2.05b encontramos exactamente el mismo circuito que en 2.05a pero implementado con compuertas evidentemente el circuito implementado es el la fig. 2.05b.

Resumiendo, la posición inicial (en el plano) de MZ en el sentido de Y es la posición 2, la posición a la cual tiene que moverse debe ser escrita en los 3 bits menos significativos de P0, el cuarto y quinto bit de P0 darán el sentido en que debe girar el motor para ir a buscar esa posición.

Una vez que el dato se encuentra escrito en P0 debe ser activado el latch de este circuito por medio de P2. Ya y Yb son sensores que han sido colocados exclusivamente por seguridad.

La similitud que existe entre los 2 circuitos de control MX y MY es evidente, la única diferencia es la cantidad de posiciones que posee cada uno.

Ej. 2.3a Si parte de la posición inicial y tiene que moverse a la posición 0, implica que tiene que desplazarse hacia UP entonces el dato que tiene que colocar en P0 debe ser XXX01000B, los 3 bits menos significativos de este dato obligan al multiplexer enrutar únicamente la entrada 0, cuando pase por las demás posiciones el dato de salida del multiplexer será 0L (D1), sólo cuando llegue a la posición 0 el dato será 1L (D1) entonces en ese momento tendremos que Y'=0, obligando a que el MY se detenga.

2.4 CIRCUITO MZ. (Fig. 2.07 y 2.08).

El mecanismo MZ descrito en la sección 1.6.3, es el encargado de mover un casete en el sentido de las Z, este mecanismo posee únicamente 2 posiciones, razón por la cual su circuito de control, es bastante simple. De modo similar que para los circuitos de control de los mecanismos MX y MY se requiere de un circuito auxiliar capaz de detenerlo cuando haya llegado a la posición solicitada por el microcontrolador. Este circuito utiliza los siguientes integrados:

I-29 74LS151(TTL) Mux 3/8, strobe, salidas Y e Y'(W).
I-30 74SL374(TTL) Latch tipo D de 8 bits. Trans. pos.
I-28 4081(CMOS) 4 compuertas AND de 2 entradas.
I-27 4071(CMOS) 4 compuertas OR de 2 entradas.

Este circuito es mucho más simple que los dos circuitos anteriores. En el circuito de la fig. 2.08 podemos ver que está formado por un multiplexer (I-29) y un latch (I-30). En el latch ingresan los 4 primeros bits del bus de datos, estos bits se harán presentes a la salida del latch por medio de la señal A2. El multiplexer pese a ser 3/8 es utilizado como 2/4. Los 2 primeros bits sirven como señales de selección en el multiplexer, la selección se realiza entre las entradas 1 y 2 del multiplexer, la entrada 1 posee la señal ZA que es una señal proveniente del sensor que se encuentra en la parte superior y la entrada 2 posee ZB que es un sensor ubicado en la parte inferior.

Los otros dos bits sirven para dar el sentido con el cual arrancará el motor que es parte de este mecanismo.

P0 = XXXX 1001B Posición inicial de MZ.

P2 = 1111 0010B Señal de activación de MZ.

Asumimos que parte de la posición inicial, en estas condiciones la única orden posible es que el mecanismo baje, por lo tanto el dato que debe colocar en P0 es XXXX0110B, los 2 primeros bits ingresan al multiplexer y enrutan la entrada 2 (ZB, sensor que se encuentra en la parte inferior), los otros 2 bits indican al motor que arranque de modo que el mecanismo descienda (BAJO) y se detendrá se active el sensor ZB. En cambio cuando tiene que subir P0 debe tener XXXX1001B, con esto MZ se mueve hacia arriba (ARRIBA) y se detendrá cuando haya activado ZA.

Los 2 circuitos de las figuras 2.07a y 2.07b, son equivalentes la única diferencia es que el primero es implementado con relés y el segundo con compuertas, cuando aparece la señal U, CZU se autoalimenta y MZ hace que el mecanismo suba, únicamente se detendrá cuando desaparezca la señal Z' y esto ocurre cuando se activa ZA es decir cuando el mecanismo alcance la parte superior.

El funcionamiento en el otro sentido es similar. Las señales Za y Zb deben ser enviadas por el usuario a través de un interruptor.

CIRCUITO DE CONTROL DEL MOTOR Z

(PRIMERA PARTE)

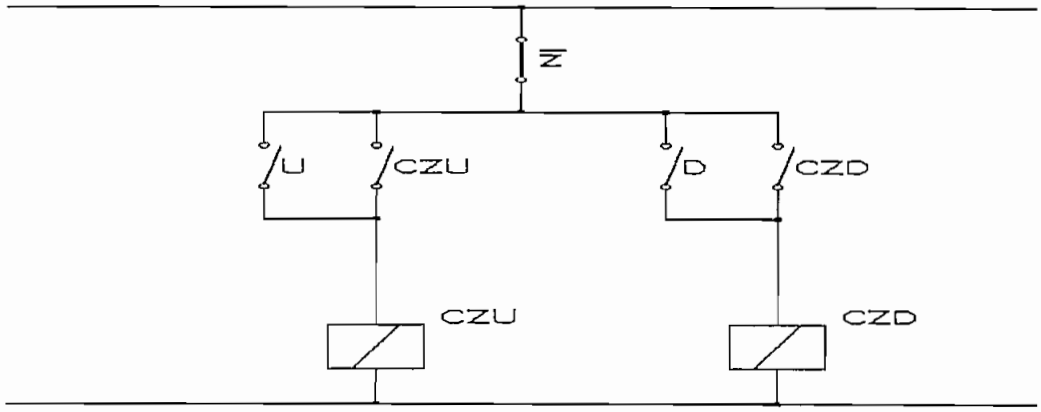


FIG. 2.Ø7a

$$CZU = \overline{Z}.(CZU+U)$$

$$CZD = \overline{Z}.(CZD+D)$$

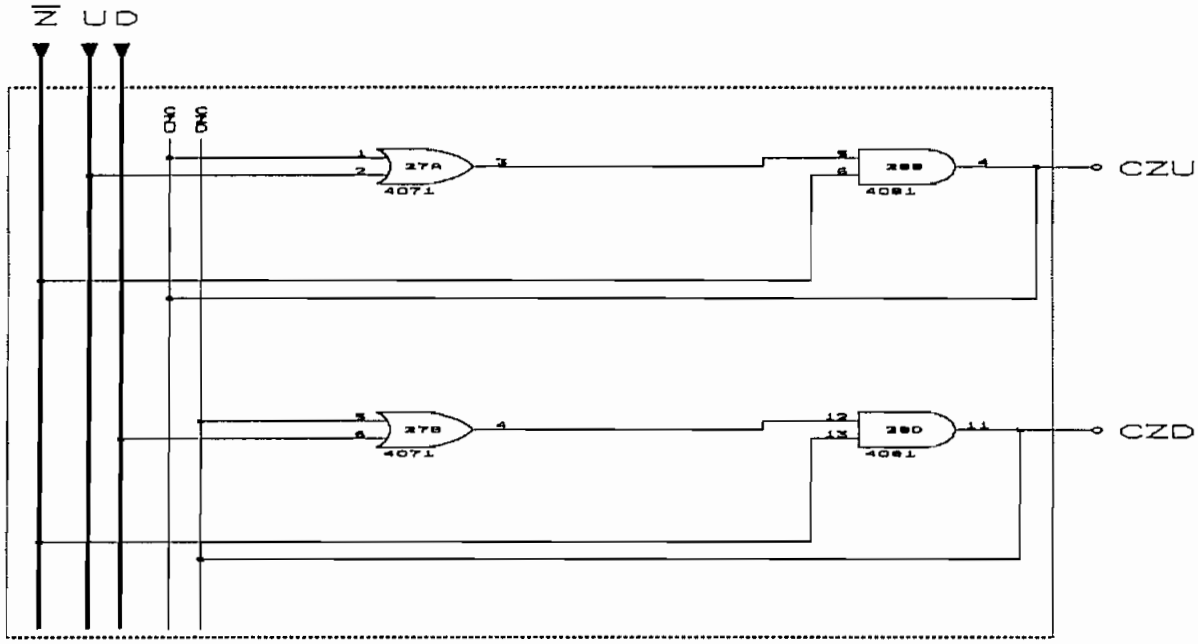


FIG. 2.Ø7b

CIRCUITO DE CONTROL DEL MOTOR Z

(SEGUNDA PARTE)

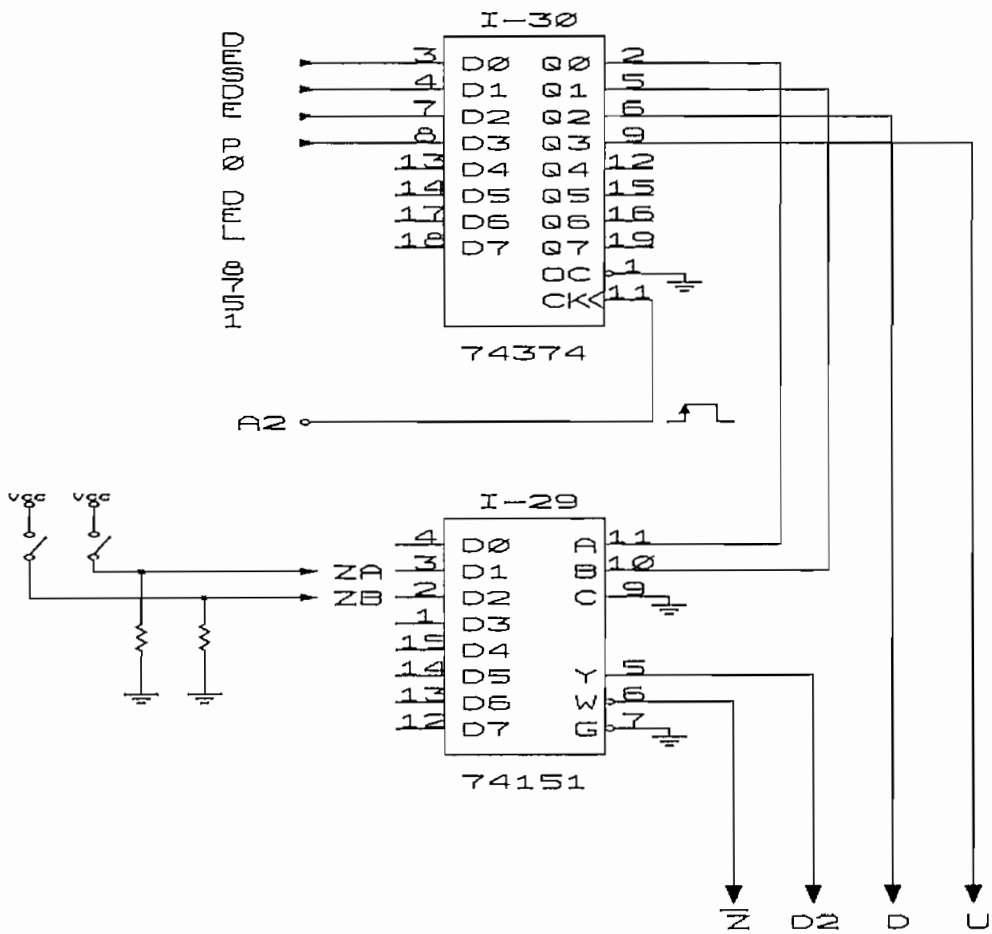


FIG. 2.08

En resumen, este circuito permite el desplazamiento de MZ, posee dos posiciones, una superior y la otra inferior. P0 = XXXX 0110B cuando deseamos que se mueva de la parte superior a la parte inferior. P0 = XXXX 1001B, cuando deseamos que se mueva de la parte inferior a la parte superior. La señal que activa este circuito es A2.

La señal que informa sobre si el mecanismo ha llegado o no a la posición deseada es D2 (D2 = 1L, cuando ya ha llegado).

2.5 CIRCUITO DE MC0 Y MC1. (Fig. 2.09).

En los literales 1.4 y 1.5 fueros descritos estos mecanismos, son los encargados de ingresar o retirar un casete desde G0 y G1 respectivamente, cada uno de estos mecanismos requiere de un motor DC (doble sentido de giro), su control se encuentra realizado completamente por el microcontrolador, en esta parte difiere con los circuitos anteriores, es decir la puesta en marcha y parada del motor es realizada por el 8751H. El usuario debe enviar las señales que indican que el mecanismo ha llegado a la posición solicitada por el microcontrolador.

Este circuito utiliza los siguientes integrados:

I-24 74SL374(TTL) Latch tipo D 8 bits. Trans. pos.

I-26 4011(CMOS) 4 compuertas NAND de 2 entradas.

MC0 y MC1 son mecanismos completamente similares, MC0 mueve un casete desde C(0,2) hasta el interior de G0. En cambio MC1 mueve un casete de la posición (41,2) hasta el interior de G1.

En la fig.2.09 observamos que del bus de datos se ocupan solo los 4 primeros bits, estos 4 bits ingresan al latch (I-24), aquí encontramos una gran diferencia con los circuitos anteriores pues el microcontrolador es el encargado de ponerlos en movimiento y también de detenerlos una vez que

reciba la señal de haber llegado a la posición solicitada (1L mínimo 500 Ms), esta señal tiene que darla el usuario. Aquí vemos la diferencia con los circuitos anteriores los cuales se detenían por si mismos. Cada uno de los 4 bits utilizados la función específica siguiente:

Bit 0. Controla a MC0 Mueve T0, desde C(0,2) hasta G0.

Bit 1. Controla a MC0 Mueve T0, desde G0 hasta C(0,2).

Bit 2. Controla a MC1 Mueve T1, desde C(41,2) hasta G0.

Bit 3. Controla a MC1 Mueve T1, desde G0 hasta C(41,2).

CIRCUITO DE CONTROL DE MCØ Y MC1

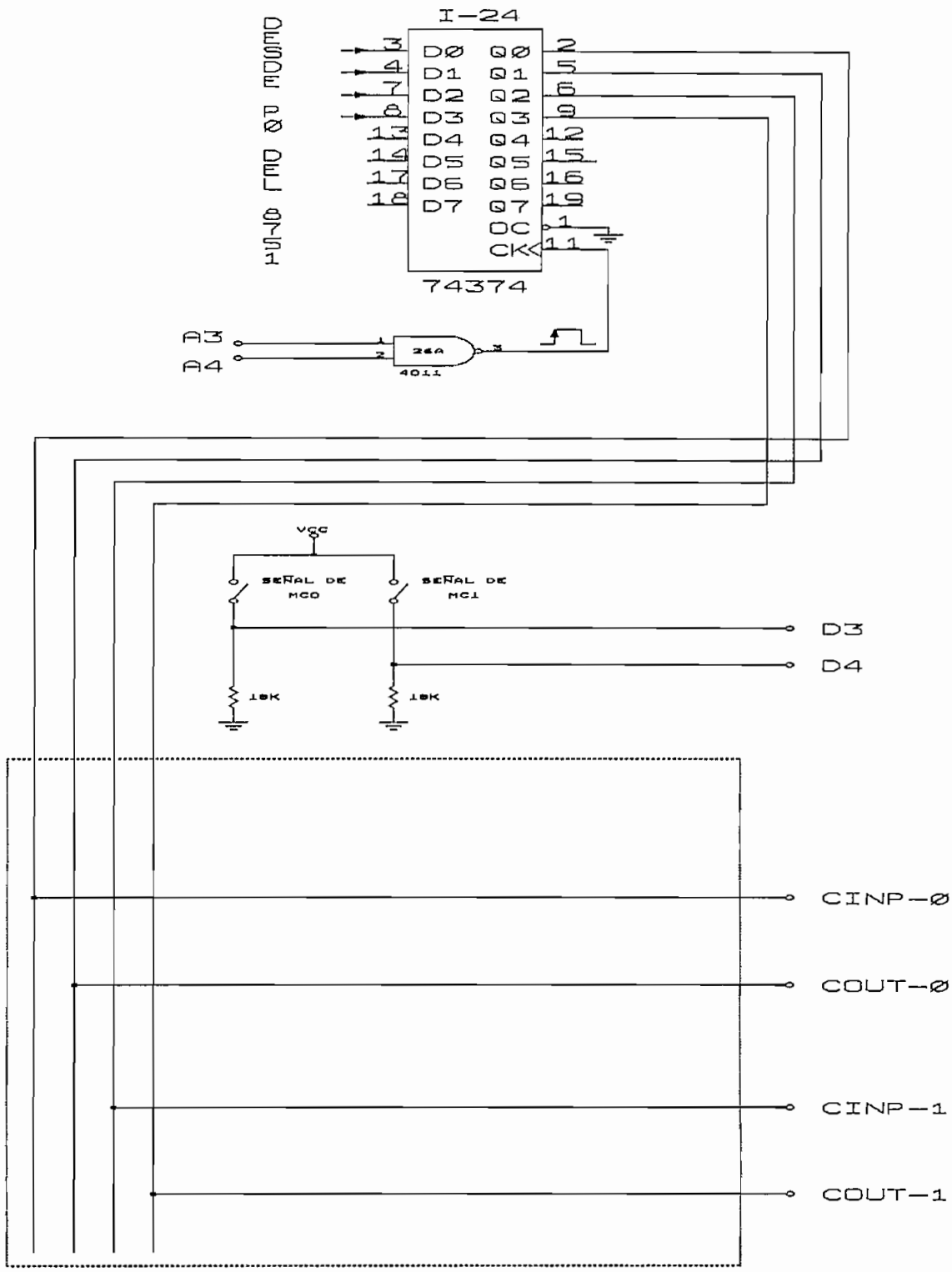


FIG. 2.09

2.6 CIRCUITO DE G0 Y G1. (Fig. 2.10)

Corresponde a los circuitos de control de las dos grabadoras, se necesitan 4 líneas del bus de datos para cada grabadora. Entonces para las 2 grabadoras se ocupa el bus completo de datos, la activación del latch correspondiente a estos datos se realiza con A3 o A4.

Este circuito utiliza los siguientes integrados:

- I-15 74SL374(TTL) Latch tipo D de 8 bits. Trans. pos.
- I-16 4071(CMOS) 4 compuertas OR de 2 entradas.

Cada una de las grabadoras es tipo reversible, esto implica que se puede elegir el lado del casete que se desea escuchar. En cada grabadora tenemos 4 controles el PLAY, FORWARD, REWIND, LADO.

PLAY — Permite escuchar los temas grabados en la cinta magnética del casete.

FORWARD — Determina el avance de la cinta desde el lado izquierdo de la grabadora hasta el lado derecho.

REWIND — Realiza el proceso inverso de FORWARD.

LADO — Permite escoger que lado de la cinta va ha ser leído por PLAY.

Por lo tanto se necesita todos los 8 bits del bus de datos, los 4 primeros bits permiten el control de G0 y los otros 4 el control de G1. La siguiente lista permite observar la función que realiza cada uno de los 8 bits del bus de datos cuando controla a las grabadoras.

Bit 0.	P0.0	PLAY de G0.
Bit 1.	P0.1	FORWARD de G0.
Bit 2.	P0.2	REWIND de G0.
Bit 3.	P0.3	LADO de G0.
Bit 4.	P0.4	PLAY de G1.
Bit 5.	P0.5	FORWARD de G1.
Bit 6.	P0.6	REWIND de G1.
Bit 7.	P0.7	LADO de G1.

Una vez que una determinada orden ha sido dada a una de las grabadoras, la respuesta de que se ha cumplido es simulada por medio de un sensor (1L mínimo por 500 MS). Una vez que el microcontrolador recibe esta respuesta retira la orden dada a dicha grabadora. Como podemos ver en la fig. 2.10, I-15 trabaja con una de las 2 señales de reloj (A5 o A6), sin embargo no es preciso las 2 señales puesto que es suficiente una. La razón para poder activar con una de las 2 señales es una cierta facilidad al efectuar el programa.

La señal S es una señal la da el usuario de un sensor, esta señal es la única capaz de detener el motor de la grabadora ya sea que este en PLAY, FORWARD O REWIND.

CIRCUITO DE CONTROL DE GØ Y G1

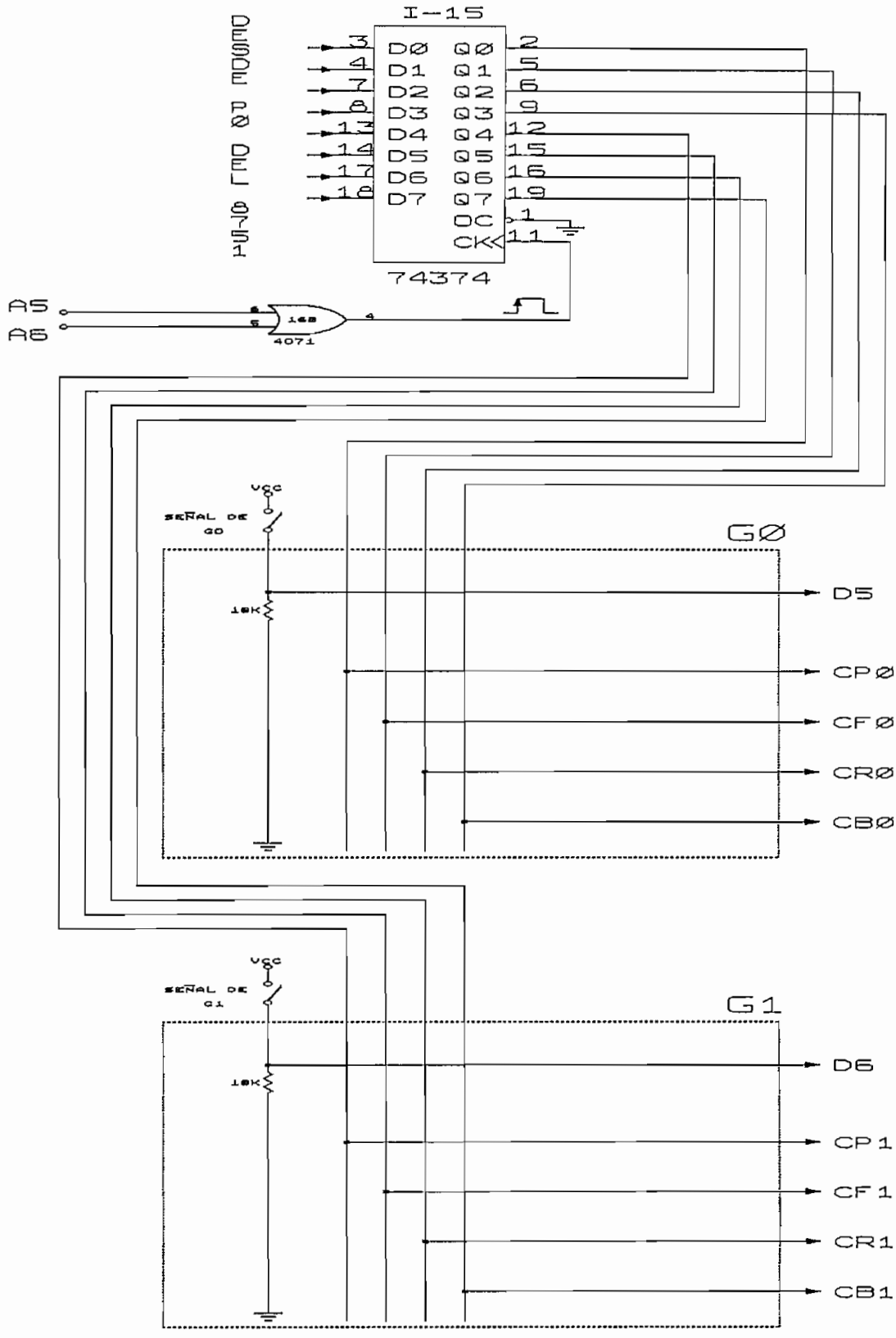


FIG. 2.10

2.7 LATCH DE RESPUESTAS.

Este latch (I-32) es el encargado de recibir las respuestas de los circuitos auxiliares, cuando un cierto mecanismo ya ha llegado a la posición solicitada por el microcontrolador. Estas señales de respuestas son bastante importantes pues en base a ellas el microcontrolador determina que parte del programa debe ser ejecutado.

La ubicación de este latch la podemos encontrar en la fig. 2.11, se trata de un integrado 74LS373. Este latch es activado por estado y cuando el microcontrolador tiene que leer las respuestas de los elementos coloca en los 4 bits menos significativos 1111, de este modo activa al latch y seguidamente lee el puerto 1.

Las respuestas leídas en el latch son:

D0 MX

D1 MY

D2 MZ

D3 MC0

D4 MC1

D5 G0

D6 G1

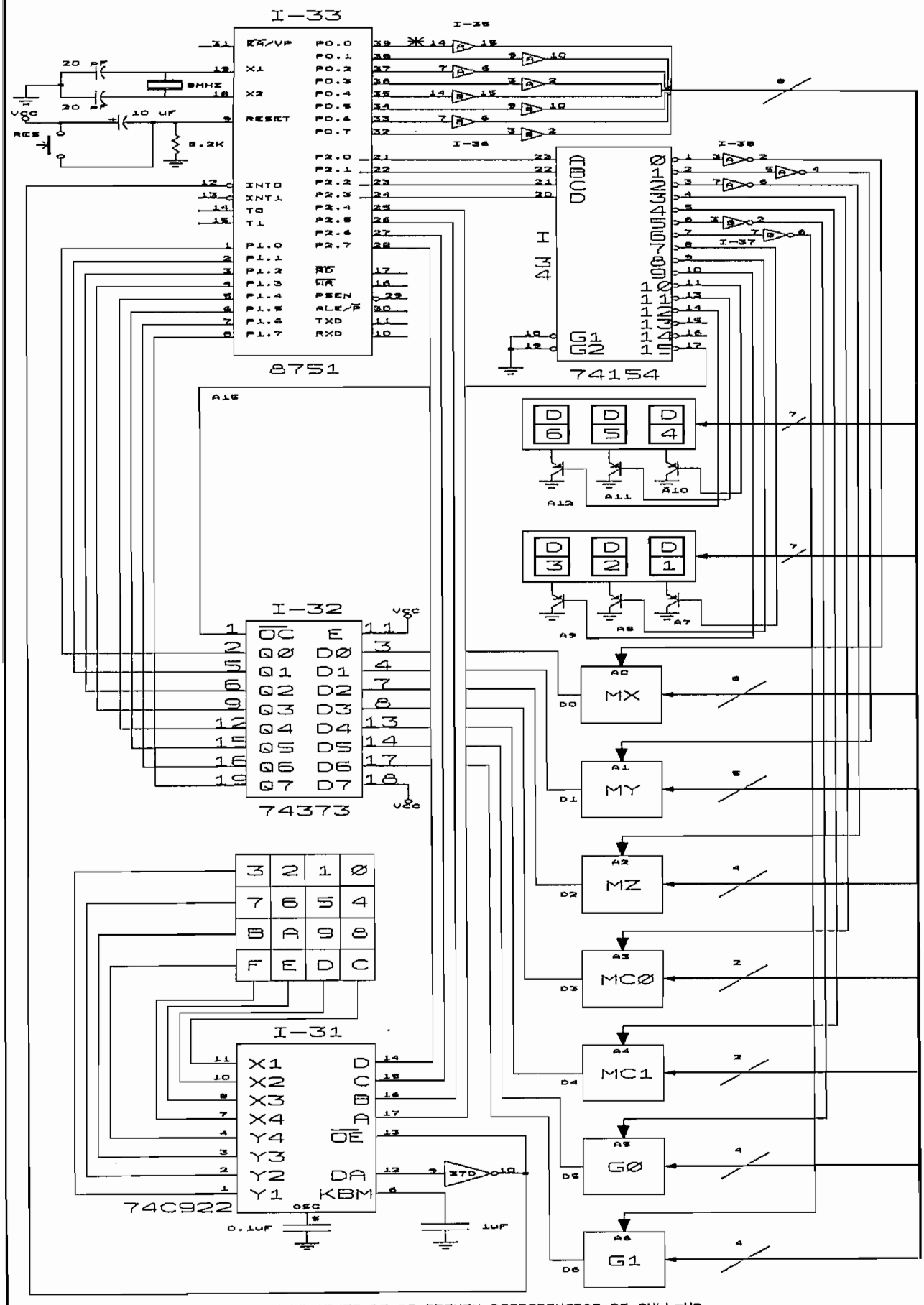
2.8 CIRCUITO PRINCIPAL: (Fig. 2.11).

El circuito principal se encuentra formado por todos los circuitos auxiliares, los elementos que lo forman son:

- 1 microcontrolador 8751H (I-33).
- 1 cristal de 8 Mhz.
- 1 decodificador de direcciones 74154 (I-34).
- 1 latch 74373 (I-32).
- 1 decodificador de teclado 74C922 (I-31).
- 2 buffers 4050 tipo CMOS (I-35 e I-36).
- 2 inversores 4049 tipo CMOS (I-37 e I-38).
- 1 inversor 7414 tipo TTL (I-37).
- 1 teclado de 16 teclas (se utilizarán solo 12 teclas).
- 1 circuito para los displays (Fig. 2.02).
- 1 circuito auxiliar para MX.
- 1 circuito auxiliar para MY.
- 1 circuito auxiliar para MZ.
- 1 circuito auxiliar para MC0 y MC1.
- 1 circuito auxiliar para G0 y G1.

El 8751H tiene el control de todo. Es preciso observar que no se ha utilizado ningún tipo de memoria externa pues no fue necesario. Para almacenar el programa se necesito 3.8K bytes de los 4K bytes que internamente dispone en el 8751H y para los datos también fue suficiente los 128 bytes de RAM que posee. Comencemos analizando la función que desempeña cada uno de los puertos del 8751H.

DIAGRAMA ELECTRONICO GENERAL DEL STM



* TODOS LOS PINES DE P0 TIENEN RESISTENCIAS DE PULL-UP

FIG. 2.11

Puerto cero. P0 (P0.0 = Pin 39, P0.1 = Pin 38, P0.2 = Pin 37, P0.3 = Pin 36, P0.4 = Pin 35, P0.5 = Pin 34, P0.6 = Pin 33, P0.7 = Pin 32). Este puerto es utilizado únicamente como puerto de salida, los integrados I-37 e I-38 sirven como amplificadores de corriente a la salida de este puerto, estos amplificadores al mismo tiempo protegen al microprocesador. Los 8 bits de P0 forman parte de lo que en adelante conoceremos como el BUS DE DATOS, el cuál lleva la información hacia los displays y los circuitos auxiliares.

Puerto uno. P1 (P1.0 = Pin 1, P1.1 = Pin 2, P1.2 = Pin 3, P1.3 = Pin 4, P1.4 = Pin 5, P1.5 = Pin 6, P1.6 = Pin 7, P1.7 = Pin 8). Es utilizado únicamente como puerto de entrada, I-32 es un latch tipo TTL. I-32 sirve como una interfase entre las respuestas que darán los circuitos auxiliares y el 8751H, este latch trabaja por estado bajo (en el pin 1). Cuando un cierto circuito auxiliar ha terminado la tarea que a él se le encomendó realizar colocará un 1L como respuesta en el respectivo pin de entrada del latch.

Nota: Es muy importante recordar que todos y cada uno de los puertos del 8751H y todos y cada uno de los pines de un puerto pueden ser utilizados como entradas siempre y cuando tengan escrito un 1L en su respectivo latch. Si un latch de un cierto pin tiene escrito un 0L y se lo utiliza como entrada el microcontrolador sufrirá un daño PERMANENTE cuando se lea un 1L por éste pin, puesto que internamente ocurrirá un cortocircuito.

Puerto dos. P2 (P2.0 = Pin 21, P2.1 = Pin 22, P2.2 = Pin 23, P2.3 = Pin 24, P2.4 = Pin 25, P2.5 = Pin 26, P2.6 = Pin 27, P2.7 = Pin 28). De este puerto, se utilizan los 4 pines menos significativos como salidas y los 4 pines más significativos como entradas. Sin embargo cuando leemos o sacamos un dato por este puerto necesariamente tenemos que hacerlo con todos los pines a la vez, por lo tanto cuando sacamos un cierto dato por los primeros 4 pines debemos sacar 1111 en los segundos 4 pines, la razón para esto fue descrita al explicar sobre P1, estos unos no son utilizados puesto que van a las salidas del decodificador de teclado y estas salidas se encuentran en alta impedancia.

Cuando lee un dato también lee los 4 primeros bits sin embargo no existe ningún problema. La función de los 4 primeros pines de P2 es seleccionar una de las salidas del decodificador de direcciones permitiendo que en la salida seleccionada aparezca un ØL. Al seleccionar una cierta salida el bus de datos (PØ) se vuelve útil solo para el circuito o display seleccionado, por lo tanto es evidente que tendremos una cierta dirección para cada uno de estos elementos, dichas direcciones son las siguientes:

P2	Dirección	Elemento a controlar
FØH	AØ	MX (Mecanismo X)
F1H	A1	MY (Mecanismo Y)
F2H	A2	MZ (Mecanismo Z)

F3H	A3	MC0 (Motor de T0)
F4H	A4	MC1 (Motor de T1)
F5H	A5	G0 (Grabadora 0)
F6H	A6	G1 (Grabadora 1)
F7H	A7	D1 (Display 1)
F8H	A8	D2 (Display 2)
F9H	A9	D3 (Display 3)
FAH	A10	D4 (Display 4)
FBH	A11	D5 (Display 5)
FCH	A12	D6 (Display 6)
FDH	A13	Libre
FEH	A14	Libre
FFH	A15	Latch (I-32)

En definitiva la función que realiza el decodificador de direcciones es permitir controlar más elementos que los que se puede controlar directamente con las salidas del puerto. Recordemos la sincronización que tiene que existir entre el bus de datos (P0) y la dirección enviada por P2.

Puerto tres. P3 (P3.0 = Pin 10, P3.1 = Pin 11, P3.2 = Pin 12, P3.3 = Pin 13, P3.4 = Pin 14, P3.5 = Pin 15, P3.6 = Pin 16, P3.7 = Pin 17). Este puerto es difiere de los 3 anteriores ya que tiene funciones especiales. Trabaja normalmente como puerto de salida pero cuando deseamos que trabaje como entrada primero se debe escribir 1L en el latch, en estas condiciones las entradas representan las funciones especiales que son las siguientes:

P3.Ø = RXD. Recepción de datos en forma serial.

P3.1 = TXD. Transmisión de datos en forma serial.

P3.2 = INTØ'. Ejecuta la subrutina de interrupción cero.

P3.3 = INT1'. Ejecuta la subrutina de interrupción uno.

P3.4 = TØ. Timer cero.

P3.5 = T1. Timer uno.

P3.6 = WR'. WRITE de la memoria RAM externa.

P3.7 = RD'. READ de la memoria RAM externa.

En nuestro caso solo ocupamos P3.2 (Pin 12) que corresponde a la interrupción Ø y la activaremos por transición.

El ØL presentado a la salida es invertido por medio de los integrados I-36 e I-37 esta inversión es necesaria puesto que solo en este momento aparece una transición positiva que es utilizada por los latches (74374) que poseen los circuitos MX, MY, MZ, MCØ, MC1, GØ y G1.

Los circuitos auxiliares que controla el circuito principal son los siguientes:

- Sistema de control del mecanismo X. MX
- Sistema de control del mecanismo Y. MY
- Sistema de control del mecanismo Z. MZ
- Sistema de control del mecanismo MCØ. MCØ
- Sistema de control del mecanismo MC1. MC1
- Sistema de control de la grabadora cero (GØ). GØ
- Sistema de control de la grabadora uno (G1). G1

Cada mecanismo posee su respectivo hardware, de este modo tenemos una cierta autonomía, los circuitos MX, MY y MZ son capaces de detenerse por si solos. Al probar los circuitos auxiliares lo podemos hacer sin el 8751H. A continuación se da el número de la figura, los integrados que ocupa, la función que realiza y la forma en que trabaja cada uno de los circuitos auxiliares.

2.9 CIRCUITOS IMPRESOS.

La fig. 2.12 no corresponde exactamente a un circuito impreso, ahí se encuentra la ubicación de todos los integrados utilizados en la construcción de la tarjeta electrónica del simulador.

En la fig. 2.13 encontramos el circuito impreso que ha permitido la implementación de la tarjeta electrónica del simulador. El circuito se encuentra en tamaño real, el número de perforaciones necesarias fue de 1250.

En la fig. 2.14 encontramos el circuito impreso para los displays, corresponde a una tarjeta electrónica completamente aparte del anterior, se encuentra a doble escala.

SILUETAS DE LOS INTEGRADOS EN EL CIRCUITO IMPRESO

NOTA: TODOS LOS INTEGRADOS SE ENCUENTRAN EN TAMAÑO REAL

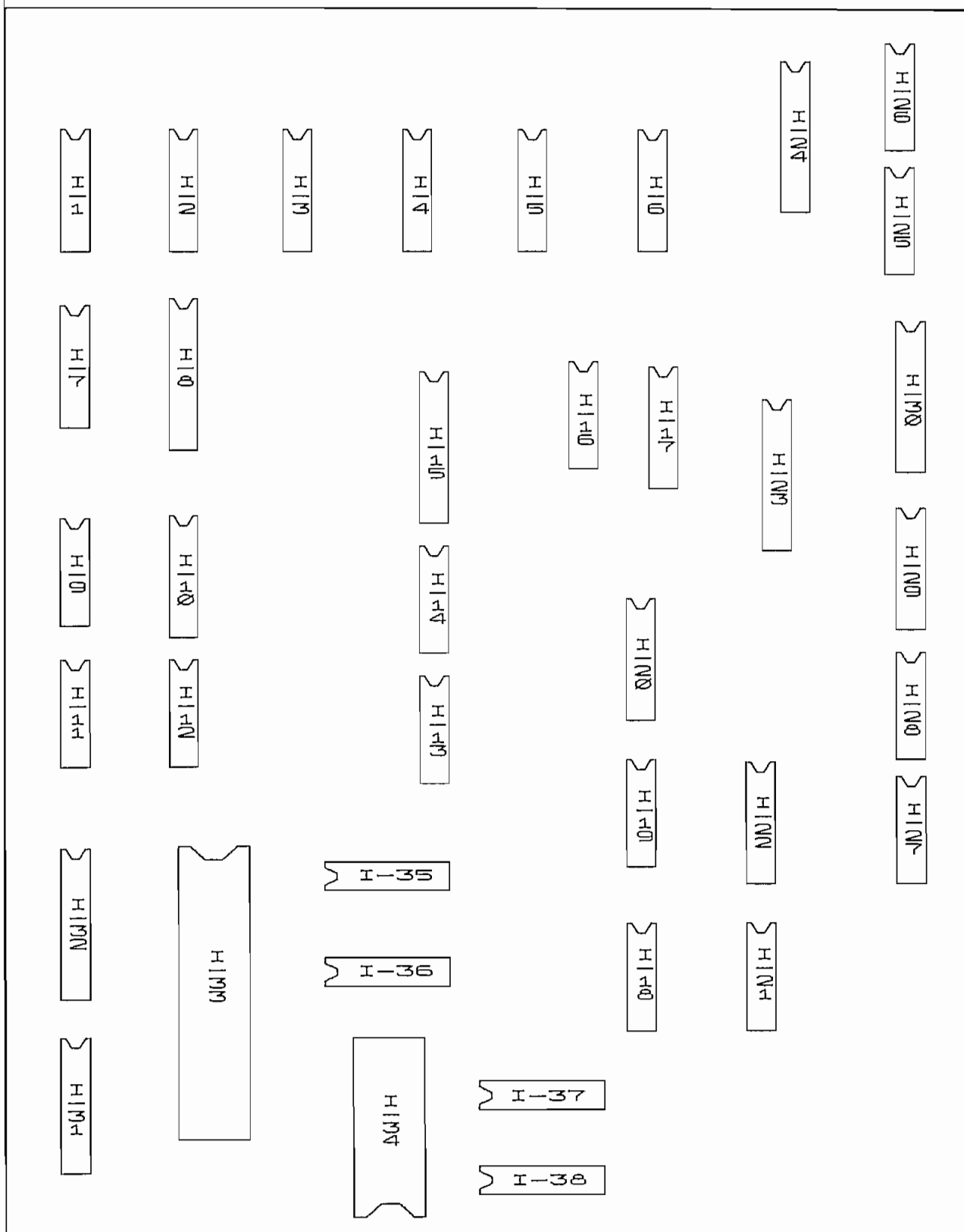


FIG. 2.12

CIRCUITO IMPRESO DEL STM

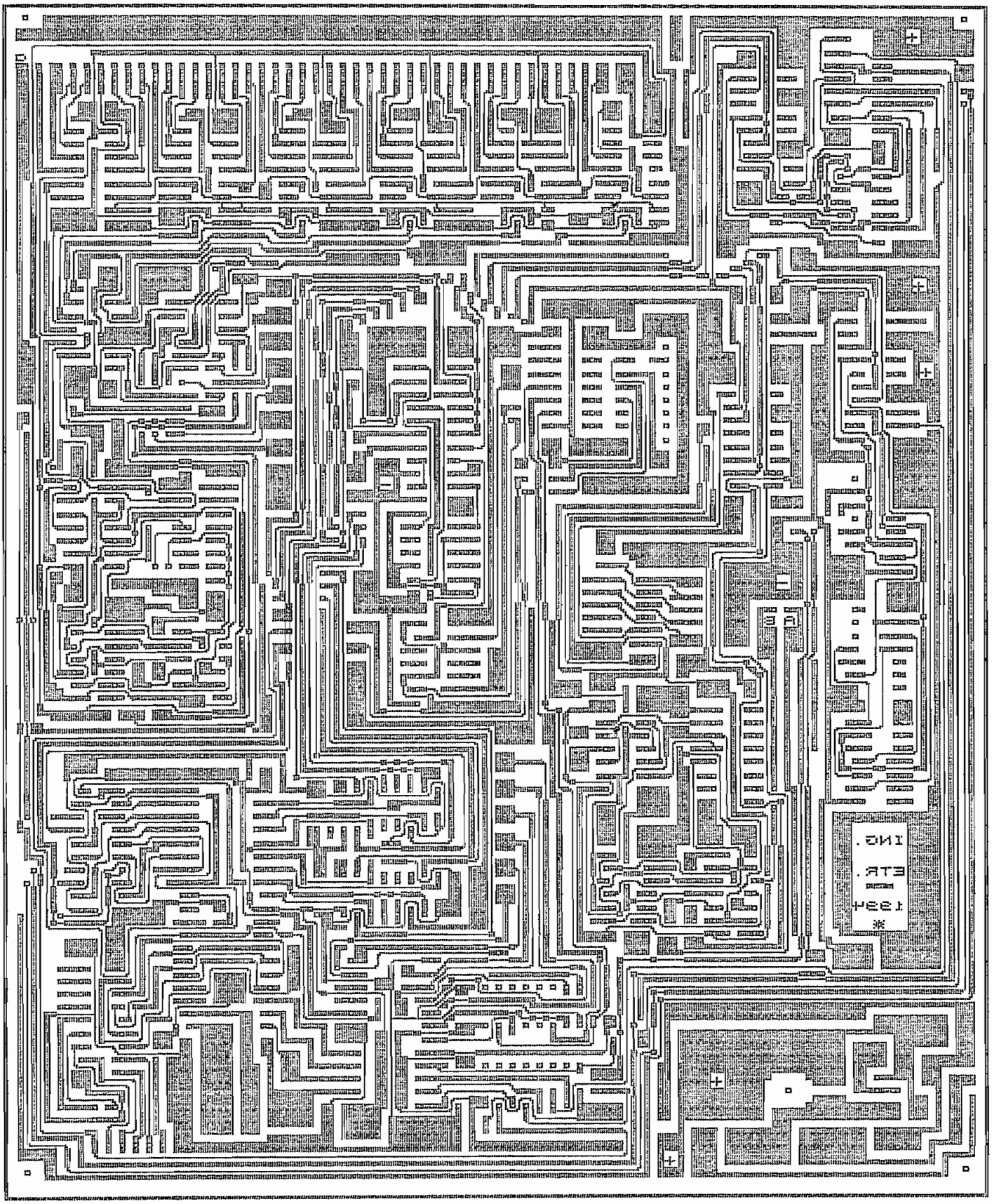


FIG. 2.13

CIRCUITO IMPRESO DE LOS DISPLAYS

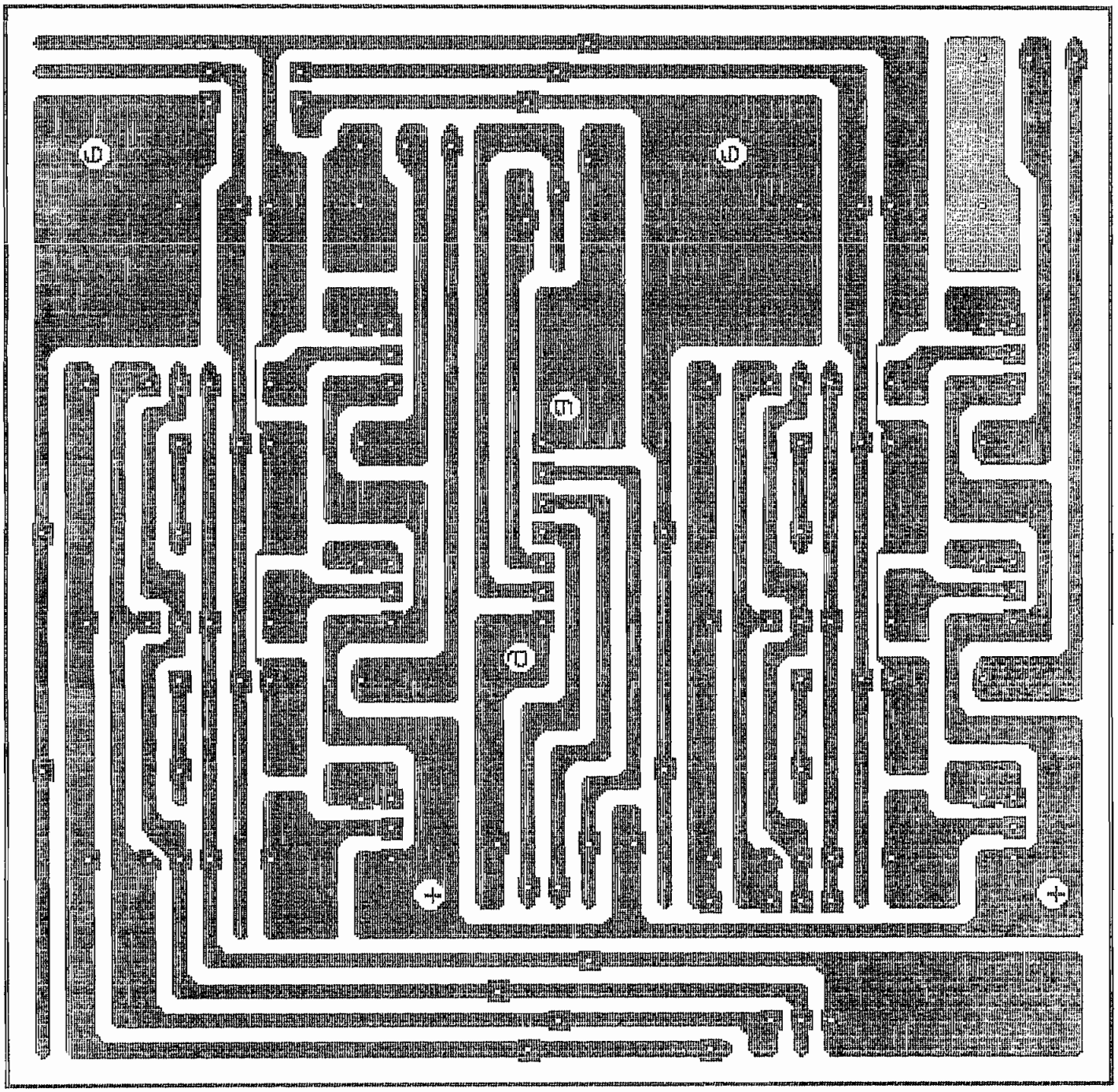


FIG. 2.14

CAPITULO III

SOFTWARE DEL STM

INTRODUCCION

En este capítulo se realiza un análisis sobre los datos necesarios para cada uno de los circuitos auxiliares, breve síntesis sobre el funcionamiento del STM, de esta manera se verá que datos son necesarios para cada una de las subrutinas. Se explica para que sirven todas y cada una de las localidades de memoria y a que subrutina pertenecen, la función que desempeña cada una de las subrutinas y el modo en que lo realiza.

En este capítulo se encuentra la explicación de todo el software, el programa que permite la simulación y todos los diagramas de flujo necesarios para el programa.

3.1 SINTESIS DEL FUNCIONAMIENTO DEL STM.

La siguiente explicación se la puede considerar casi innecesaria si se ha entendido los capítulos 1 y 2, sin embargo vamos a recordar lo principal de los dos capítulos anteriores.

El STM posee 120 casetes dispuestos en 3 filas de 40 casetes cada fila, los casetes se encuentran verticalmente con la parte mas ancha hacia arriba.

El mecanismo MX permite ubicarse dentro de cualquiera de las 42 columnas numeradas entre 0 y 41. Las numeraciones entre 1 y 40 son las correspondientes a las 40 columnas de casetes (3 casetes en cada columna). La posición adicional 0 ayuda ubicar el casete seleccionado frente a la grabadora cero (G0) y la segunda posición adicional 41 ayuda ubicar el casete seleccionado frente a la grabadora uno (G1).

El mecanismo MY es completamente similar a al mecanismo MX excepto que permite ubicarse sobre una determinada fila, el número de filas existentes es 5, numeradas entre 0 y 4.

Las posiciones 0, 2 y 4 son efectivamente las que corresponden a las 3 filas de casetes (cada fila tiene de 40 casetes), las posiciones 1 y 3 son filas intermedias las mismas que ayudan a depositar un determinado casete en una cierta casilla.

El mecanismo MZ, esta compuesto de una especie de pinza semicerrada, cuando el mecanismo baja el casete queda atrapado si en estas condiciones el mecanismo sube el casete subirá junto a él. Cuando se desea liberar el casete de MZ, el casete es bajado hasta la casilla que corresponda luego MZ se mueve en el sentido de las Y de una de las 3 filas de casetes a una fila intermedia quedando liberado el casete.

MCØ, es el mecanismo que permite ingresar un casete a GØ, tiene dos posiciones dentro y fuera de GØ. MC1, es similar que MCØ excepto que trabaja con G1.

GØ es reversible, esto implica que se puede seleccionar el lado del casete que se desea escuchar sin necesidad de dar la vuelta al casete. Posee 4 controles, PLAY (ESCUCHAR), FORWARD (AVANZAR), REWIND (REBOBINAR), y LADO (Ø o 1). G1 es completamente similar a GØ.

Nota: Es preciso recordar que todos los casetes siempre se encuentran con una posición fija que es el lado A frente a GØ y el lado B frente a G1. Ningún mecanismo podrá cambiarlos de esta posición.

Además la posición inicial de todos los casetes es rebobinado al lado A. Una vez que un casete ha sido escuchado previo a ser devuelto a su posición física original también la cinta del casete tiene que ser restituida a su posición original.

El siguiente es un resumen de los datos que son necesarios enviar a cada uno de los elementos y de la dirección que corresponde a dicho elemento.

El dato es enviado por P0 y la dirección es enviada en los 4 primeros bits de P2 (los otros 4 bits de P2 son utilizados como entradas razón por la cual cuando sacamos una dirección por los 4 primeros bits, en los otros bits tiene que ser escrito unos, esta es una condición para que un puerto pueda ser utilizado como entrada).

El siguiente paráfrasis permitirá aclarar ciertos puntos sobre lo que se acaba de exponer.

A0 hasta A12: Dirección de los elementos, está en los 4 primeros bits de P2. Corresponden desde F0 hasta FC respectivamente. FF es la dirección del latch de respuestas.

MX (A0). Utiliza los 8 bits de P0, los 6 primeros bits especifican la posición en la cual MX se detendrá, el bit 7 determina el movimiento hacia la izquierda del MX, y el bit 8 el movimiento hacia la derecha.

Xa es un sensor del extremo izquierdo y Xb es un sensor del extremo derecho, estos 2 sensores encuentran exclusivamente por seguridad. La posición inicial o de reposo de MX es la 20.

MY (A1). Utiliza los 5 primeros bits de P0, los 3 primeros especifican la posición a la cual debe moverse MY, el bit 4 el movimiento hacia arriba, y el bit 5 hacia abajo. Ya e Yb son sensores en la parte superior e inferior de las filas. La posición inicial de MY es la 2.

MZ (A2). Tiene 2 posiciones, utiliza los 4 primeros bits de P0. El dato necesario para que MZ ascienda es XXXX1001B y para que descienda es XXXX0110B. La posición inicial es el mecanismo en la parte superior.

MC0 y MC1 (A3 o A4). Cuando el dato sobre P0 es XXXX0001B el casete entra a G0, cuando es XXXX0010B el casete sale de G0, con XXXX0100B el casete entra a G1 y con XXXX1000B el casete sale de G1.

G0 y G1 (A5 o A6). Los primeros 4 bits de P0 son para el control de G0 y los segundos 4 bits por G1. El control que realizan es sobre el PLAY, FORWARD, REWIND Y LADO de cada grabadora. La posición inicial es 0000 0000B.

DISPLAYS (A7 hasta A12). Usa 7 bits de P0, que es el dato para 7 segmentos y con la dirección puesta sobre P2 activamos el display correspondiente.

LATCH (A15). Es activado por estado bajo se lo utiliza para leer las respuestas de los elementos anteriores excepto los displays).

3.2 SUBRUTINAS UTILIZADAS EN EL PROGRAMA PRINCIPAL.

La siguiente es una lista de las subrutinas utilizadas por el programa principal, el orden en que se encuentra corresponde al mismo orden en el cual las encontraremos en el programa. Se encuentra dividida en 4 grupos que son:

El primero corresponde a las subrutinas de movimiento, que controlan todo lo que son movimientos físicos es decir a MX, MY, MZ, MCØ y MC1.

El segundo grupo de subrutinas son de control de las grabadoras, dentro de este grupo se han incluído las subrutinas UBICAS y DEVCAS que no son exactamente de control de las grabadoras sin embargo se las admitió dentro de este grupo.

El tercer grupo corresponde a la subrutina MUESTRE, esta subrutina contiene 3 subrutinas y se la ha puesto aparte por ser la subrutina que más tiempo pasa en ejecución dentro del programa.

El último grupo corresponde a la subrutina TECLADO junto con su subrutina auxiliar DEC_BIN, este grupo constituyen el medio para obtener datos desde el exterior hacia el microcontrolador. La subrutina DEC_BIN permite transformar los datos decimales a hexadecimal para que pueda ser almacenado en 1 sola localidad de memoria.

3.2.1 SUBRUTINAS DE MOVIMIENTO.

SUB MOVXY

SUB MOVZA

SUB MOVZB

SUB MC0INP

SUB MC0OUT

SUB MC1INP

SUB MC1OUT

3.2.2 SUBRUTINAS DE CONTROL DE LAS GRABADORAS

SUB FRWG0

SUB RWDG0

SUB FRWG1

SUB RWDG1

SUB UBICASG0

SUB UBICASG1

SUB PLAYG0 (SUB. BIN_DEC)

SUB PLAYG1 (SUB. BIN_DEC)

SUB DEVCASG0

SUB DEVCASG1

3.2.3 SUBRUTINA MUESTRE (RES, PXY, DIS).

SUB RES

SUB PXY

SUB DIS

3.2.4 SUBRUTINA DE INGRESO DE DATOS DESDE EL TECLADO.

SUB TECLADO

SUB DEC_BIN

3.3 ORGANIZACION DEL STACK DE MEMORIA.

A continuación se muestra la distribución de la RAM.

Localidades utilizadas por el programa principal

00	01	02	03	04	05	06	07
08	09	0A	0B	0C	0D	0E	0F

Localidades utilizadas por la SUB-MUESTRE

10	11	12	13	14	15	16	17

Localidades ocupadas por la SUB-TECLADO (Int-0) SP

18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27
28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37
38	39	3A	3B	3C	3D	3E	3F

Pox	Poy		Smc0	Smc1	Py0	Py1	Lb0
40	41	42	43	44	45	46	47
Pfx	Pfy	Pfz					Lb1
48	49	4A	4B	4C	4D	4E	4F
Rx	Ry	Rz	Rm0	Rmc1	RG0	RG1	RE
50	51	52	53	54	55	56	57
#cas	Px	Py	Ld	#cas	Px	Py	Ld
58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67
68	69	6A	6B	6C	6D	6E	6F
CC0	CC1	Cas. seleccionando			Casete escuchando.		
70	71	72	73	74	75	76	77
CENT	DEC.	UNI.	BIN.	DATOS A SER ESCUCH.			NTS
78	79	7A	7B	7C	7D	7E	7F

A continuación se da una explicación detallada de la función que desempeña cada una de estas localidades.

00H hasta 0FH Son utilizadas por el programa principal, corresponden a los 2 primeros bancos de registros.

10H hasta 17H Son utilizadas por la subrutina MUESTRE, dentro de esta subrutina se encuentran tres subrutinas, la primera subrutina es RES, la segunda es PXY y la tercera es DIS.

18H hasta 1FH Las utiliza la subrutina TECLADO, esta subrutina es ejecutada por medio de la interrupción 0, cuando aparece una transición negativa en el pin 12 del microcontrolador. Es activada por transición no por estado.

20H hasta 3FH El SP podrá desplazarse de acuerdo a lo que se necesite dentro de estas 32 localidades, no será preciso utilizar todas.

40H y 41H Tienen los datos de columna y fila respectivamente en que se encuentra MZ.

45H y 46H Poseen las señales que permitirán conocer si las grabadoras G0 y G1 se encuentran en PLAY (45H para G0 y 46H para G1). Cuando existe un 01 la grabadora correspondiente a esa localidad significará que se encontrará en PLAY, caso contrario no.

47H y 4FH — Permitirán saber cuando una grabadora se encuentra libre (47H para G0 y 4FH para G1). Cuando existe un 01 la grabadora correspondiente a esa localidad no estará libre y cuando existe un 00 si lo estará. Cabe notar que las localidades 45H y 46H no tienen la misma función que las localidades 47H y 4FH, puesto que una determinada grabadora puede no encontrarse en PLAY pero puede estar ocupada ejecutando un REWIND o FORWARD o simplemente el casete aún no ha sido devuelto a su posición original.

48H y 49H — Poseen los valores de las posiciones de columna (X) y fila (Y) a las cuales tiene que desplazarse MZ.

4AH — Contiene el dato con el cual el MZ se desplazará hacia arriba (09H) o hacia abajo (06H).

4BH y 4CH — Poseen los datos de control de MC0 y MC1 cuando 4BH = 01 el casete es introducido en G0 y cuando 4BH = 02 el casete es retirado de G0. Cuando 4CH = 04 el casete es introducido en G1 y cuando 4CH = 08 el casete es retirado de G1. Los datos de 4BH y 4CH son sacados simultáneamente por P0.

4DH y 4EH Contienen los datos que controlarán a G0 y G1 respectivamente. Los datos de las dos localidades serán sacados simultáneamente.

50H, 51H y 52H Estas localidades son utilizadas para almacenar las respuestas de los elementos MX, MY y MZ respectivamente, cuando una de estas localidades contiene 01 significará que dicho elemento ha llegado a la posición solicitada.

53H y 54H Son destinadas a contener la respuesta de MC0 y MC1 respectivamente, un 00 en la localidad correspondiente determinará que el elemento ha llegado a la posición deseada.

55H y 56H En estas localidades se encuentran las respuestas de las grabadoras G0 y G1, cuando tenemos un 01 en una de éstas, significará que la grabadora ha terminado la tarea encomendada.

57H Contiene la respuesta de todos los elementos, esta respuesta se encuentra almacenada en hexadecimal, el bit 0 corresponde a MX, el bit 1 a MY, el bit 2 a MZ, el bit 3 a MC0, el bit 4 a MC1, el bit 5 a G0 y el bit 6 a G1. El bit 7 no posee absolutamente ninguna respuesta.

58H hasta 5BH Corresponde al buffer de G0, 58H contiene el número del casete solicitado, 59H contiene la posición X (columna) del casete solicitado, 5AH la posición Y (fila) del casete solicitado y 5BH el lado del casete.

5CH hasta 5FH Corresponde al buffer de G1, 5CH contiene el número del casete solicitado, 5DH contiene la posición X (columna) del casete solicitado, 5EH la posición Y (fila) del casete solicitado y 5FH el lado del casete.

60H hasta 6FH Estas 8 localidades se encuentran libres, esto claramente demuestra que los 128 bytes de RAM fueron suficientes para las variables del programa..

70H Esta localidad corresponde a la selección que será ubicada en G0.

71H Esta localidad corresponde a la selección que será ubicada en G1.

72H hasta 74H Contienen centenas, decenas y unidades de la selección que se encuentra REALIZANDO ese momento, estos dígitos serán mostrados en D6, D5, D4. En ningún momento se exhibe los ceros de la izquierda.

75H hasta 77H Contienen las centenas, decenas y unidades de la selección que se encuentra ESCUCHANDO ese momento, estos dígitos serán mostrados D3, D2, D1. En ningún momento se exhibe los ceros de la izquierda.

78H hasta 7AH Aquí serán ubicados los dígitos que van ingresando desde el teclado, estos dígitos son los mismos que irán apareciendo en D6, D5, D4.

7BH Contiene el equivalente hexadecimal del los tres dígitos anteriores, el cual posteriormente pasará a formar parte del buffer de selecciones siempre y cuando el dato sea valido.

7CH hasta 7EH Corresponden a lo que llamaremos el buffer de selecciones, aquí serán ubicados los códigos de las selecciones que serán escuchados. Al almacenar en hexadecimal estamos reduciendo de las 3 localidades que originalmente fueron necesarias a una.

7FH Contiene el número total de selecciones (NTS). Este número máximo puede ser F0H (240) puesto que el número máximo de casetes que puede contener el STM es de 120.

3.4 EXPLICACION DEL FUNCIONAMIENTO DEL PROGRAMA.

3.4.1 PROGRAMA PRINCIPAL. (Fig. 3.01). Primero recordemos que G0 y G1 se encuentran frente a frente, G0 al lado izquierdo y G1 al lado derecho. Según el diagrama general de flujo del STM, cuando G0 y G1 se encuentran libres tendrá prioridad G0, es decir si existen datos, el casete correspondiente a dicho dato será ubicado en G0, si los temas seleccionados se encuentran al lado A, G0 será puesto en PLAY-LADO 1, esto se debe a la ubicación relativa que posee el casete respecto a G0.

Si los temas seleccionados se encuentran en el LADO B, G0 será puesto directamente en PLAY-LADO 0 (este pequeño inconveniente no tendremos con G1).

Una vez que estamos escuchando el casete que se encuentra en G0 si existe otra selección el nuevo casete será colocado en G1 si la selección requiere poner en PLAY el lado 1, el casete será puesto primero en FORWARD y luego permanecerá listo para ser puesto en PLAY-LADO 1, una vez que termine de ser escuchado el casete que se encuentra en G0.

Si el casete que se encuentra en G0 terminó, entonces inmediatamente es puesto en PLAY el casete que se encuentra en G1, una vez que ha sido puesto en PLAY, el casete de G0 es devuelto a su posición original previo la reposición del estado original de la cinta del casete.

DIAGRAMA GENERAL DE FLUJO DEL STM

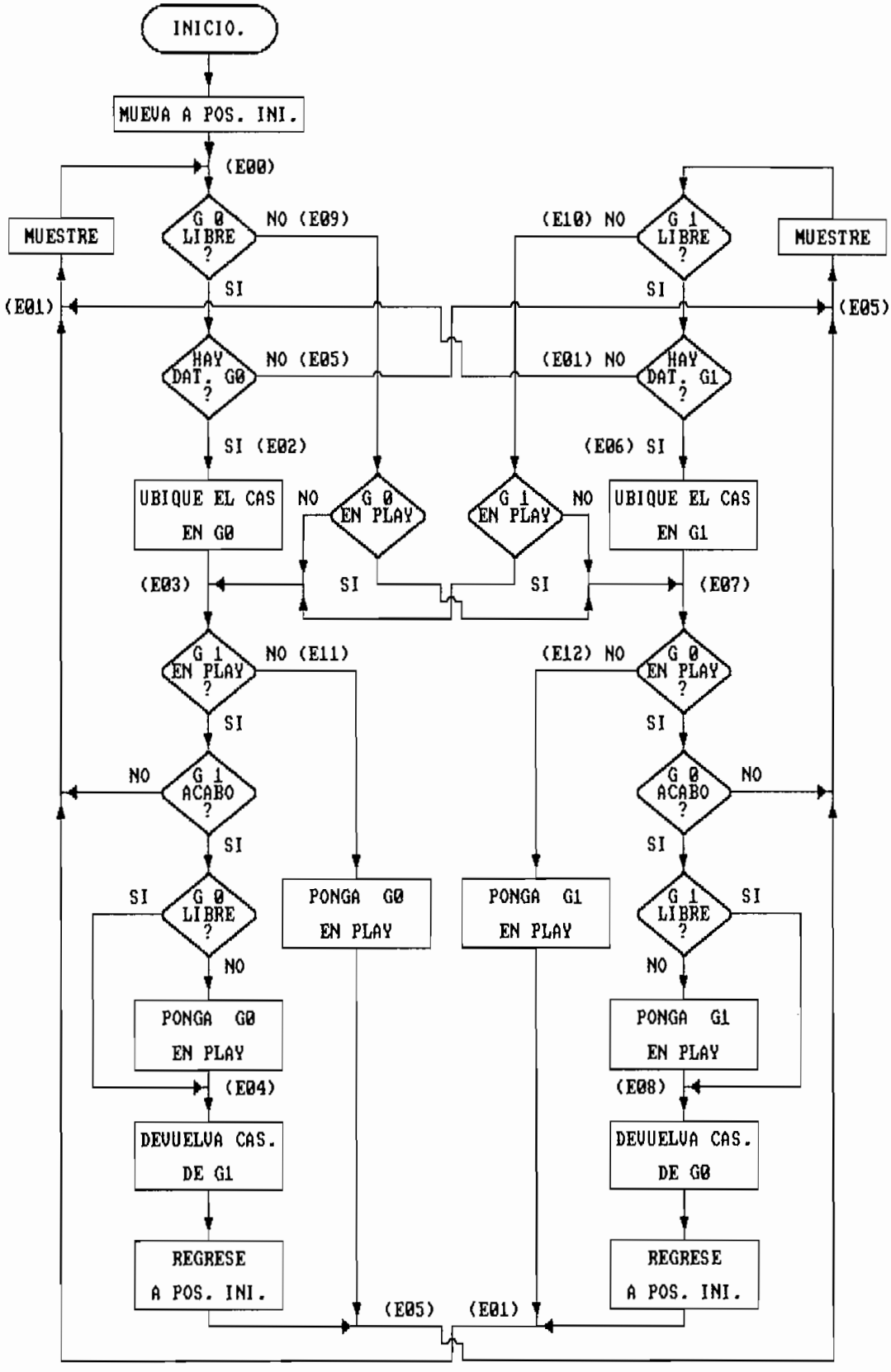


FIG. 3.01

En este momento se encuentra escuchando el casete de G1, si aparece una nueva selección se ubicará el casete seleccionado en G0 y esperara a que G1 termine para poder ser puesto en PLAY. Mientras exista una selección, dicho casete seleccionado será ubicado en la grabadora libre, si las dos grabadoras se encuentran libres será ubicado en G0.

Dentro del programa las señales de grabadora libre u ocupada lo tendremos por medio de un 00 ó 01 respectivamente. Las localidades utilizadas para este fin son la 47H para G0 y la 4FH para G1.

A continuación se encuentra una explicación detallada del funcionamiento de cada una de las subrutinas.

3.4.2 SUBRUTINA MOVXY. (Fig. 3.02). Esta subrutina permite mover el MZ desde una cierta posición inicial (Xo,Yo) especificada por las localidades de memoria 40H y 41H hasta una cierta posición final (Xf,Yf) especificada por las localidades de memoria 48H y 49H. Debemos tener en cuenta que los 2 movimientos pueden realizarse simultáneamente, algo que no puede suceder con el mecanismo MZ. Esta subrutina calcula el sentido de movimiento de MX, en base a las posiciones iniciales y finales si debe moverse hacia la derecha es colocado un 1 en el bit 7 antes de sacar este valor por P0, si el movimiento es a la izquierda se coloca un 1 en el bit 6. Un funcionamiento similar a MX posee MY, trabaja con las localidades 41H (Yo) y 49H (Yf).

SUB MOVXY

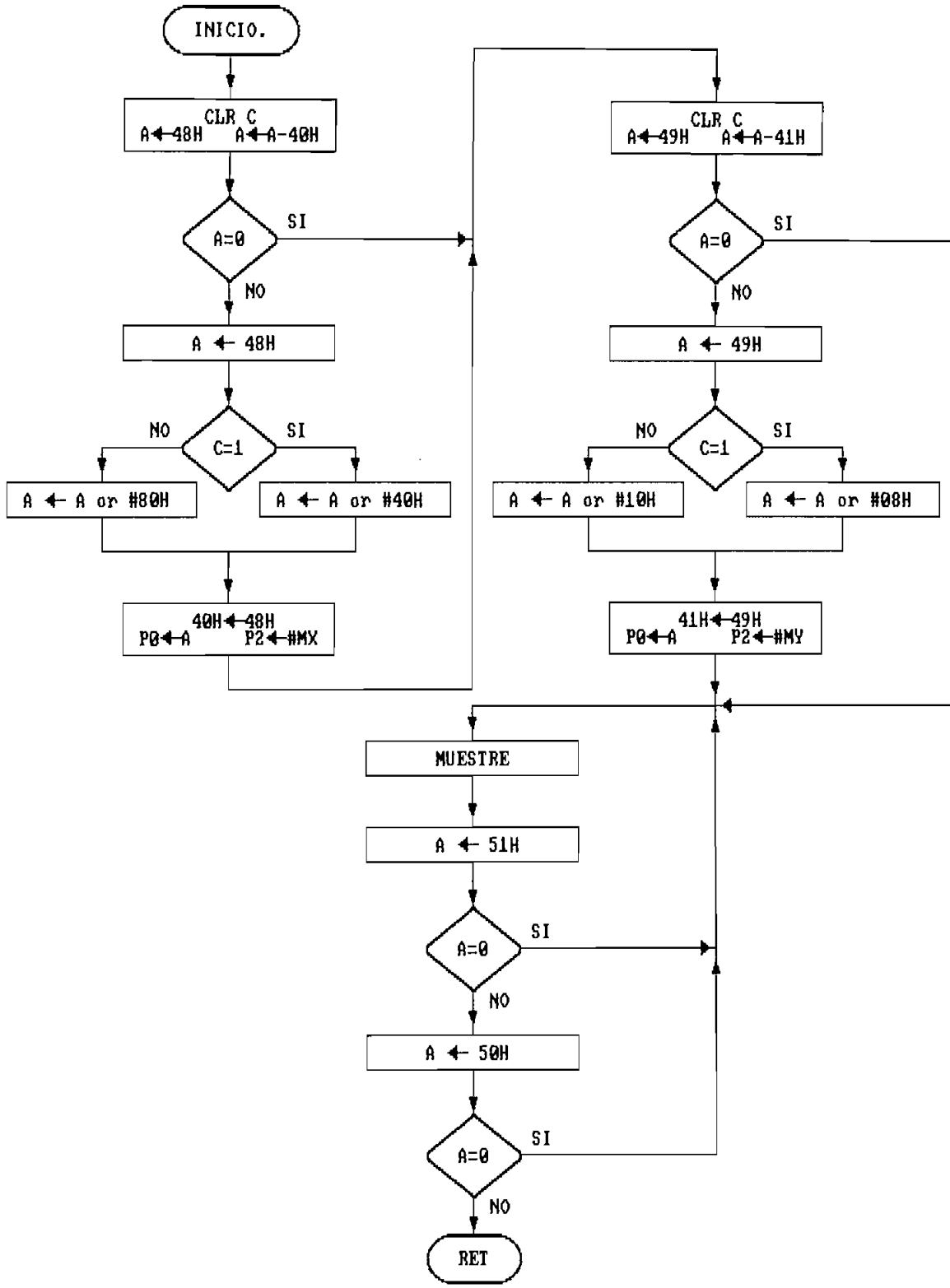


FIG. 3.02

La subrutina termina únicamente cuando se ha llegado a la posición deseada caso contrario no será posible que el STM realice ninguna otra actividad, esta respuesta de que el mecanismo ha alcanzado la posición deseada la obtenemos en las localidades 50H y 51H.

3.4.3 SUBRUTINAS MOVZA Y MOVZB (Fig. 3.03). Estas subrutinas permiten controlar a MZ, cuando es llamada como MOVZA (09H) asciende y cuando es llamada como MOVZB (06H) MZ desciende.

La respuesta de MZ lo encontramos en la localidad 52H, un 01H en esta localidad indicará que ha llegado a la posición solicitada, esta subrutina no terminará mientras el MZ no haya alcanzado dicha posición, razón por lo cual sería inútil esperar que se ejecute cualquier otra cosa mientras no aparezca la respuesta 01 de MZ.

3.4.4 SUBRUTINAS MC0INP Y MC0OUT. (Fig. 3.04). MC0INP permite la ubicación de un cierto casete desde la posición (0,2) hasta dentro de G0, cuando es llamada como MC0OUT la subrutina realiza la función opuesta a MC0INP, es decir saca el casete de G0 y ubica en la posición (00H,02H).

El programa permanecerá en esta subrutina mientras la respuesta no haya sido primero un 01H y luego 00H, el 01H indicará que el mecanismo ha llegado a la posición deseada en cambio el 00H indicará que MC0 queda listo para recibir en cualquier otro momento una nueva orden.

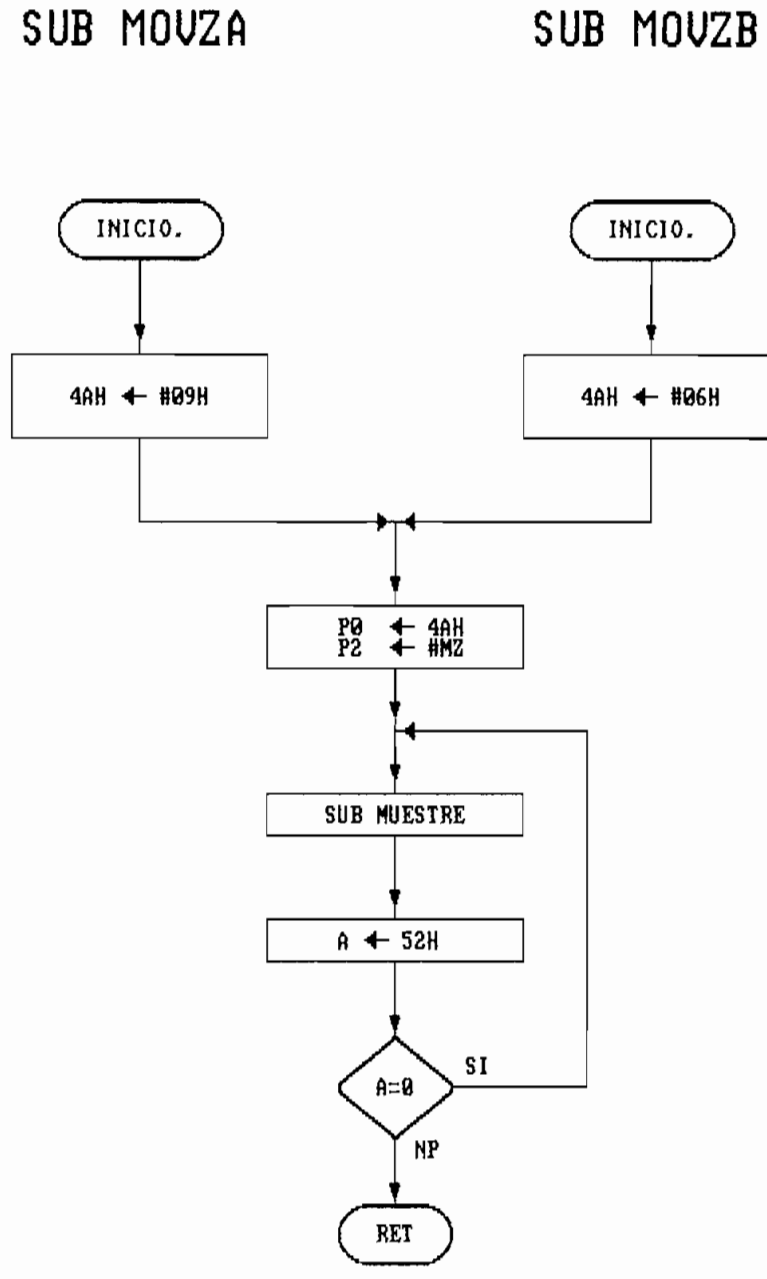

SUB MOVZA
SUB MOVZB
INICIO.
INICIO.
4AH ← #09H
4AH ← #06H
P0 ← 4AH
P2 ← #MZ
SUB MUESTRE
A ← 52H
A=0
SI
NP
RET

FIG. 3.03

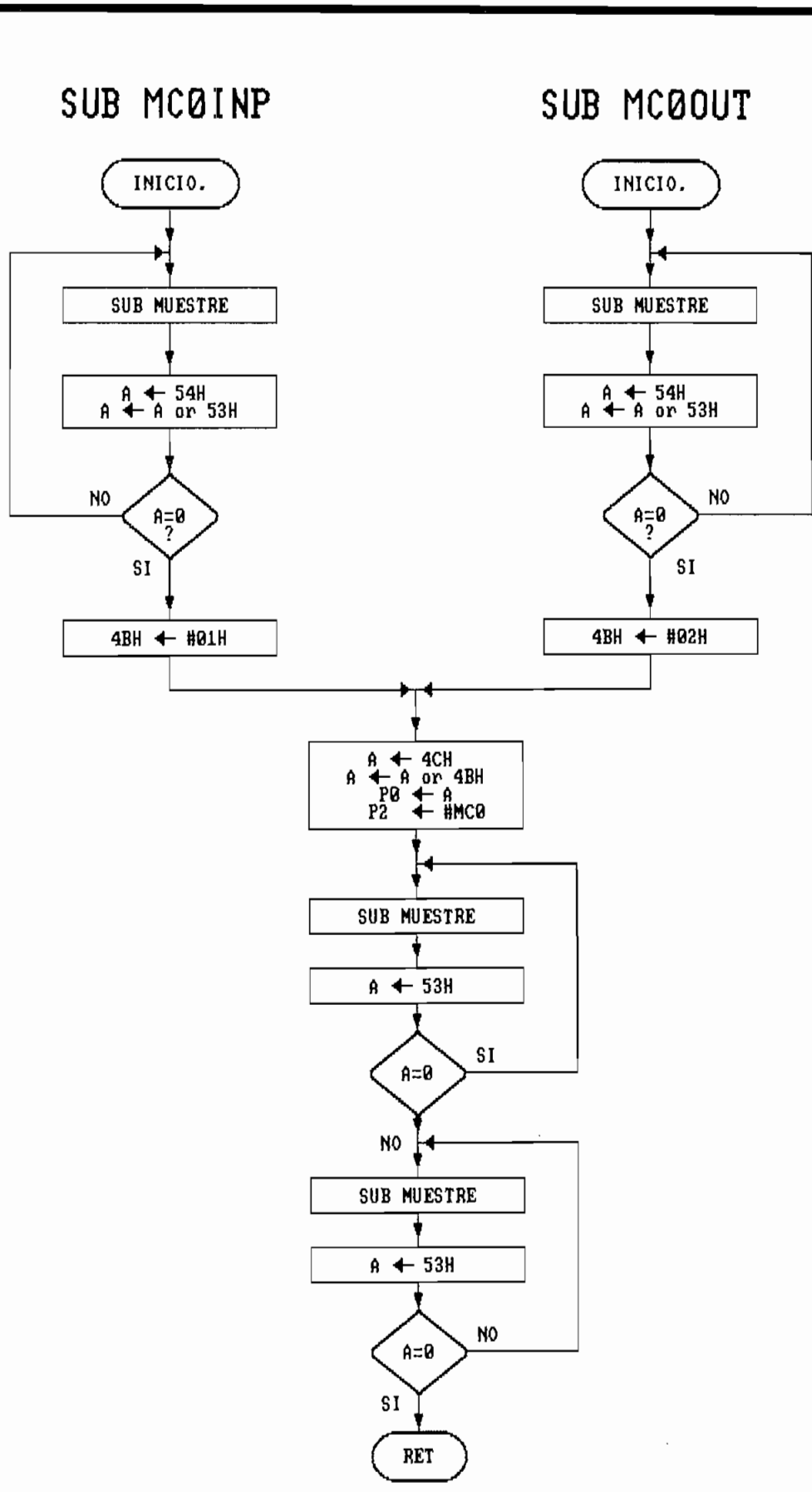

SUB MC0INP
INICIO.
SUB MUESTRE
A ← 54H
A ← A or 53H
A=0 ?
NO
SI
4BH ← #01H
SUB MC0OUT
INICIO.
SUB MUESTRE
A ← 54H
A ← A or 53H
A=0 ?
NO
SI
4BH ← #02H
A ← 4CH
A ← A or 4BH
P0 ← A
P2 ← #MC0
SUB MUESTRE
A ← 53H
A=0
SI
NO
SUB MUESTRE
A ← 53H
A=0
NO
SI
RET

FIG. 3.04

3.4.5 SUBRUTINAS MC1INP Y MC1OUT. (Fig. 3.05). MC1INP permite la ubicación de un cierto casete desde la posición (29H,02H) hasta dentro de G1, cuando es llamada como MC1OUT la subrutina realiza la función opuesta a MC1INP, es decir saca el casete de G1 y ubica en la posición (29H,02H). El programa permanecerá en esta subrutina mientras la respuesta no haya sido primero un 01H y luego 00H, el 01H indicará que el mecanismo ha llegado a la posición deseada en cambio el 00H indicará que MC1 queda listo para recibir en cualquier otro momento una nueva orden.

3.4.6 SUBRUTINA RES. (Fig. 3.06). Esta subrutina es la primera de tres partes de la subrutina MUESTRE, previamente expliquemos como se encuentra formada dicha subrutina. Se encuentra formada por tres subrutinas que en si son totalmente independientes entre ellas pero siempre son ejecutadas una a continuación de otra. Estas 3 subrutinas son RES (Fig. 3.06), PXY (Fig. 3.07) y DIS (Fig. 3.08). Expliquemos la función de cada una de estas 3 subrutinas y la razón de porque se ejecutan a la vez.

RES, es la encargada de obtener las respuestas de cada uno de los 7 elementos que conforman el STM, esta respuesta la obtiene en P0, la almacena en la localidad 57H y de ahí obtiene el bit cero y lo almacena en la localidad 50H, luego obtiene el bit 1 lo almacena en la localidad 51H y así sucesivamente hasta obtener el bit 6 y almacenarlo en la localidad 56H.

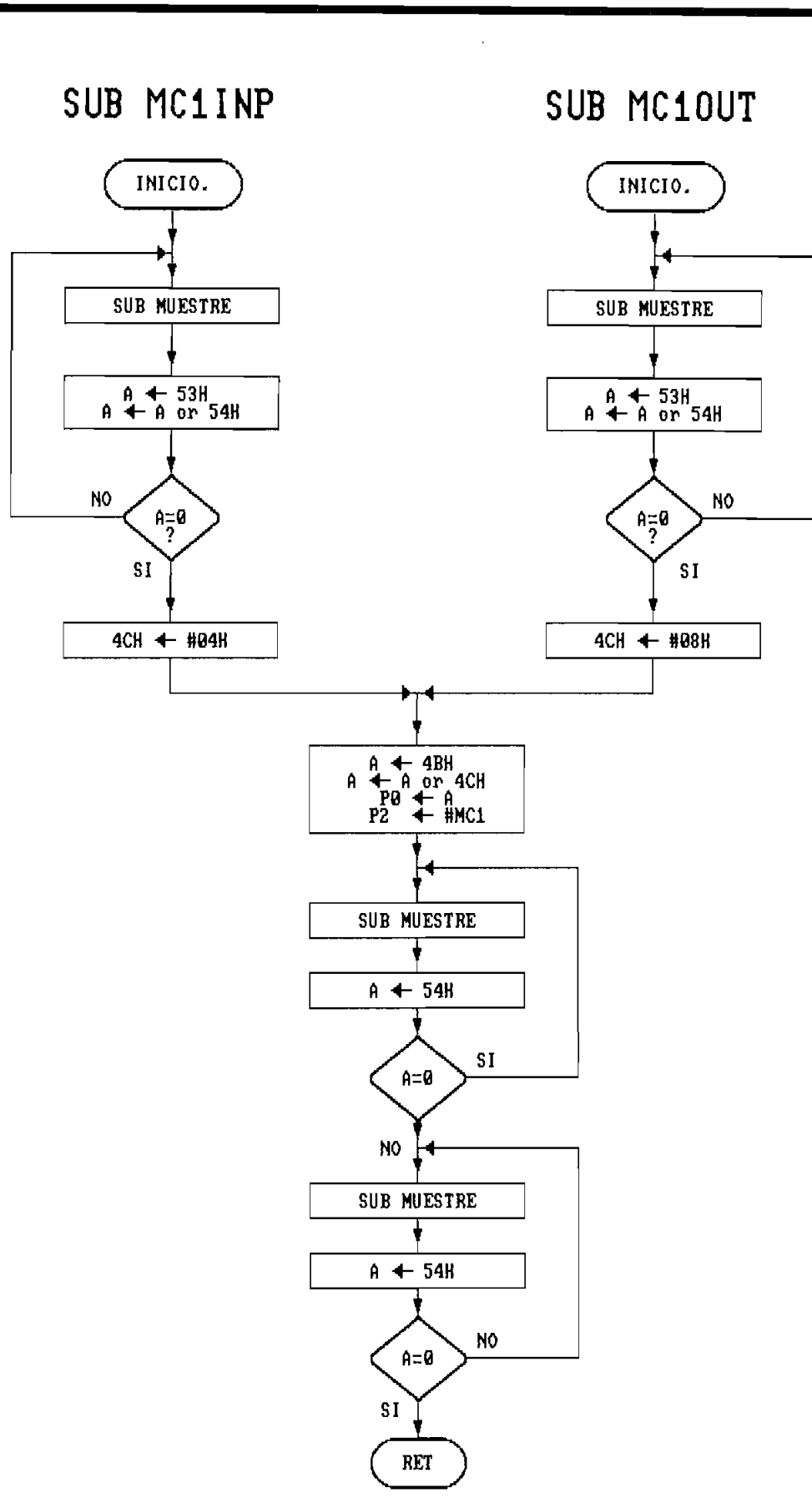

SUB MC1INP
SUB MC1OUT
INICIO.
INICIO.
SUB MUESTRE
SUB MUESTRE
A ← 53H
A ← A or 54H
A ← 53H
A ← A or 54H
NO
A=0 ?
SI
NO
A=0 ?
SI
4CH ← #04H
4CH ← #08H
A ← 4BH
A ← A or 4CH
P0 ← A
P2 ← #MC1
SUB MUESTRE
A ← 54H
A=0
SI
NO
SUB MUESTRE
A ← 54H
A=0
NO
SI
RET

FIG. 3.05

(MUESTRE) SUB RES

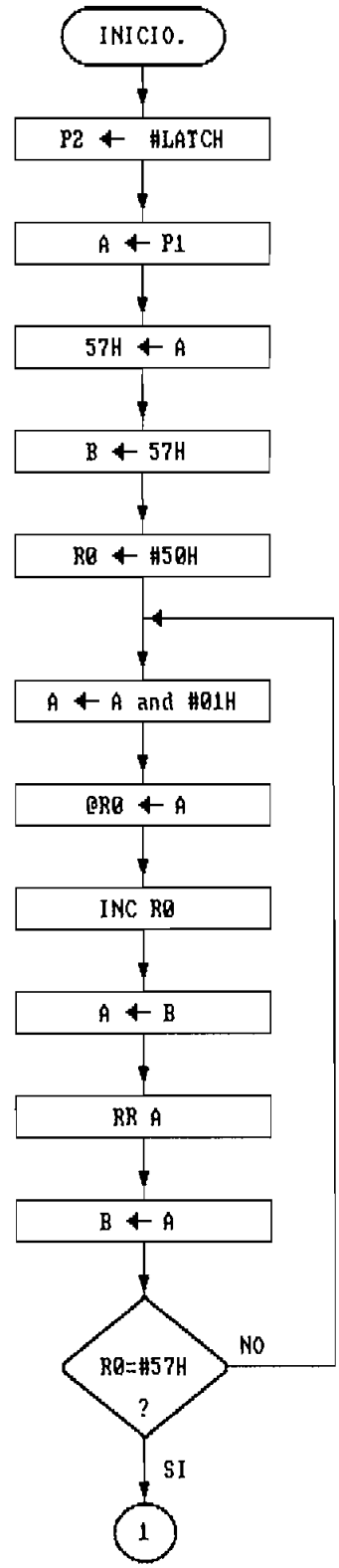


FIG. 3.06

Una explicación más detallada sobre que elemento le corresponde cada una de estas respuestas lo podemos ver en la sección 3.3.

3.4.7 SUBRUTINA PXY. (Fig. 3.0.7). Es la encargada de pasar los datos (si los hay) desde el buffer de selecciones (7DH, 7EH, 7EH), hasta los buffers de G0 y G1 (las localidades de estos buffers fueros descritas en la sección 3.5). Si existen datos lo primero que revisa es cual buffer se encuentra libre, puede darse que el buffer libre sea el de G0 o el de G1.

Si el buffer libre es el de G0, toma el dato y calcula el número del casete, almacena este dato en la localidad 58H, luego calcula la columna y almacena en 59H. Calcula la fila donde se encuentra ese casete y almacena dicho valor en la localidad 5AH, por último calcula el lado del casete que será escuchado y almacena en la localidad 5BH (almacena 00H si el lado es el A, y almacena 01H si el lado es el B).

Si el buffer libre es el de G1, toma el dato y calcula el número del casete al cual corresponde, almacena este dato en la localidad 5CH, luego calcula la columna y almacena este dato en la localidad 5DH. Calcula la fila donde se encuentra ese casete y almacena ese valor en la localidad 5DH, por último calcula el lado del casete que será escuchado y almacena en la localidad 5FH (almacena 00H si el lado es el A, y almacena 01H si el lado es el B).

(MUESTRE) SUB PXY

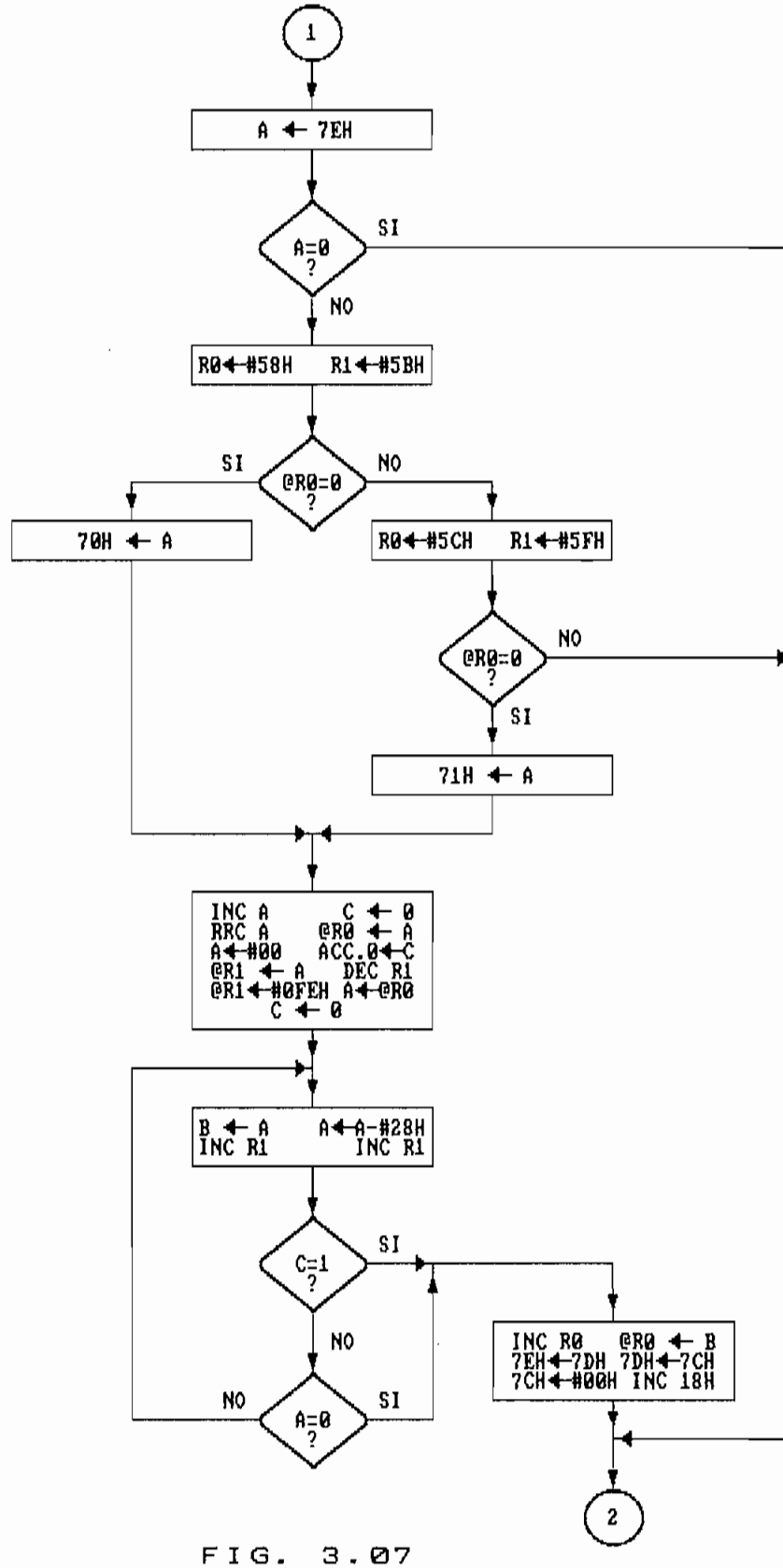


FIG. 3.07

Si el caso es que los 2 buffers se encuentran libres tendrá prioridad el buffer de GØ, esto claramente implica que cuando recién se enciende el STM y se realiza la primera selección, ésta será escuchada en GØ.

3.4.8 SUBRUTINA DIS. (Fig. 3.Ø.8). Es la tercera parte de la subrutina MUESTRE. DIS es la encargada de mostrear en los 3 primeros displays la selección que se encuentra realizando en ese momento es decir los datos que estamos ingresando desde el teclado. No exhibirá los ceros de la izquierda y cuando no se está realizando ninguna selección únicamente indicará el cero de la derecha. En los segundos 3 displays indicará el código de la selección que en ese momento estamos escuchando, de modo similar que para los 3 primeros displays no exhibirá los ceros de la izquierda, si ninguna selección ha sido realizada y no estamos escuchando nada, únicamente indicará el cero de la derecha.

La frecuencia de muestreo de cada display es de 1 Khz, esto se encuentra calculado según el cristal utilizado para el microcontrolador, para nuestro caso el cristal es de 8MHz. Esta frecuencia de 1KHz es la adecuada para realizar el muestreo puesto que si utilizamos una frecuencia menor se observará que los displays parpadean debido a la lenta frecuencia de barrido, si utilizamos una frecuencia muy alta en cambio corremos el riesgo de que los datos se confundan en los displays es decir que pese a que estamos mandando datos para un display aparezca en otro display.

(MUESTRE) SUB DIS

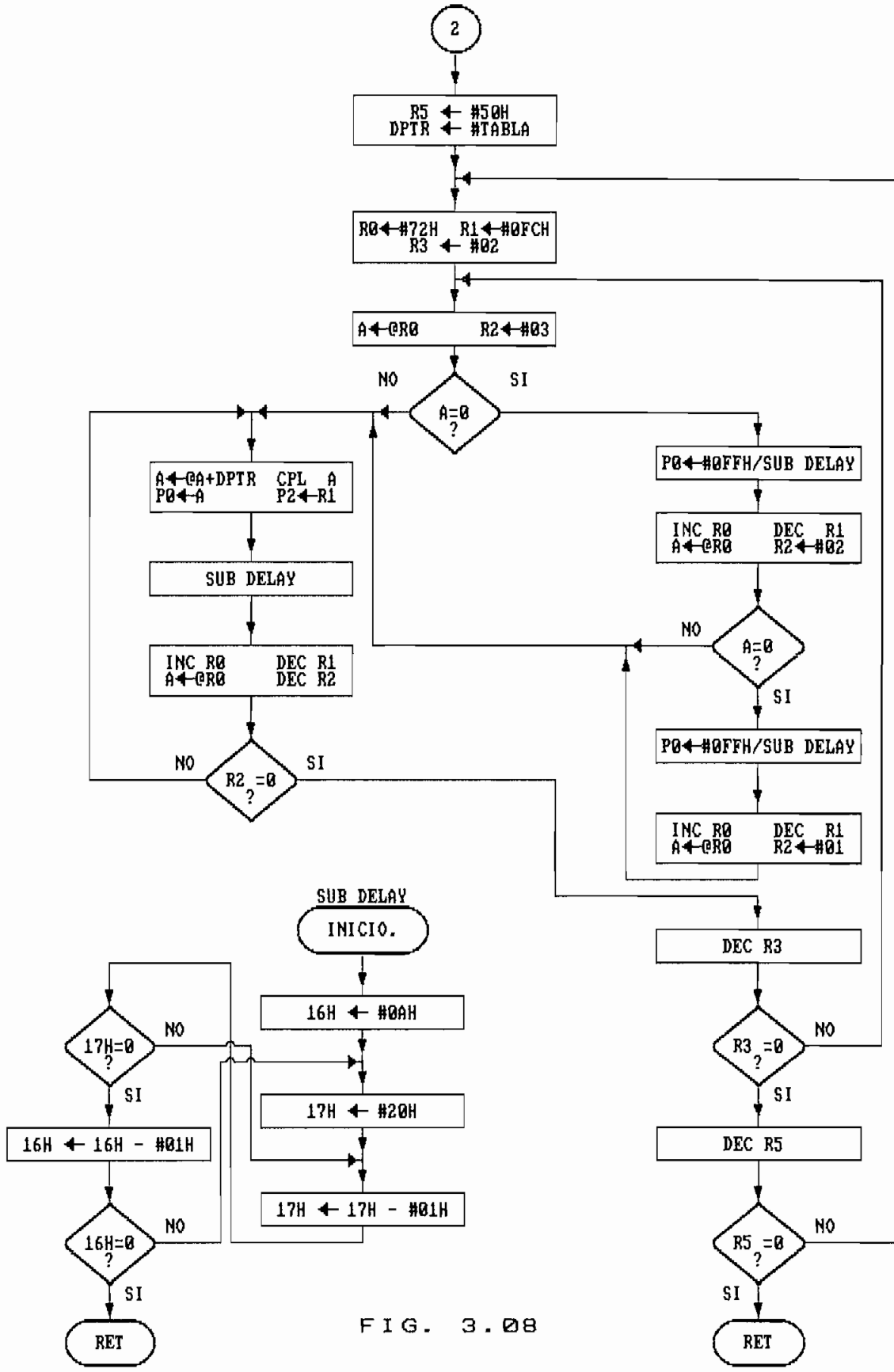


FIG. 3.08

Dentro de la subrutina MUESTRE, la subrutina DIS se ejecuta 80 veces mientras que la subrutinas RES y PXY solamente se ejecutan una vez. Cada vez que se ejecuta la subrutina MUESTRE el tiempo que se demora en terminar esta es aproximadamente 500 Ms. Podemos decir que durante el tiempo que se encuentra funcionando el STM, más del 99% del tiempo se encuentra ejecutando la subrutina MUESTRE.

3.4.9 SUBRUTINAS FRWG0 y RWDG0. (Fig. 3.09). FRWG0 permite colocar en FORWARD a G0 es decir que la cinta del casete que en ella se encuentra avance desde el lado izquierdo de G0 hacia el lado derecho, el dato necesario será #02H en la localidad 4EH.

Una vez que se ha ingresado a esta subrutina no podrá salir de ella mientras no aparezca primero un 01H en la localidad 55H y luego aparezca un 00H en la misma localidad, esta localidad contiene la respuesta de G0.

SUBRUTINA RWDG0, permite colocar en REWIND a G0 es decir que la cinta del casete que en ella se encuentra avance desde el lado derecho de G0 hacia el lado izquierdo, el dato necesario será #04H en la localidad 4EH. Del igual manera que para FRWG0, una vez que se ha ingresado a esta subrutina no se podrá salir de ella mientras no aparezca primero un 01H en la localidad 55H y luego aparezca un 00H en la misma localidad, esta localidad contiene la respuesta de G0 y es controlada por la subrutina RES.

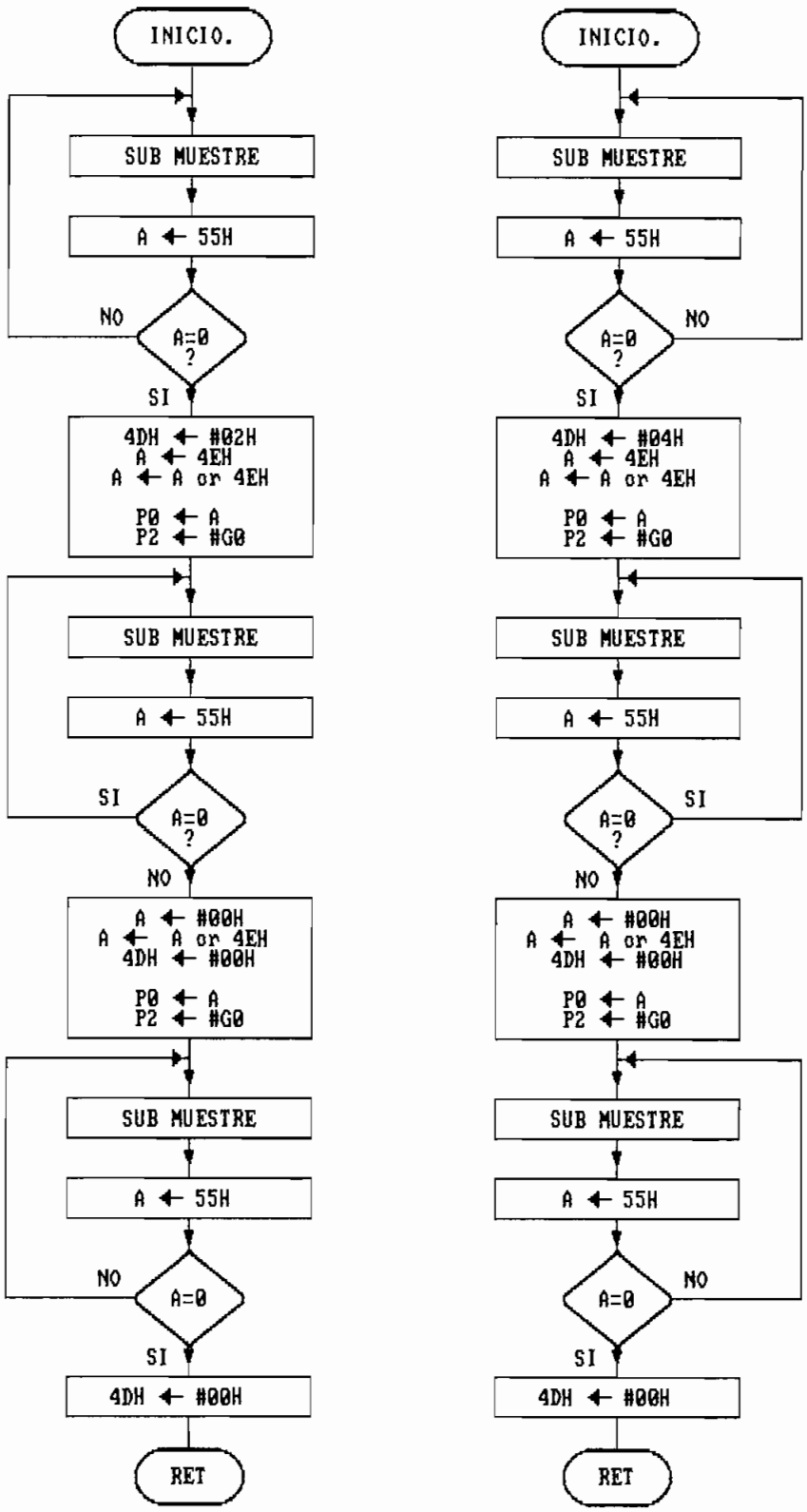

SUB FRWG0
INICIO.
SUB MUESTRE
A ← 55H
NO
A=0 ?
SI
4DH ← #02H
A ← 4EH
A ← A or 4EH
P0 ← A
P2 ← #G0
SUB MUESTRE
A ← 55H
SI
A=0 ?
NO
A ← #00H
A ← A or 4EH
4DH ← #00H
P0 ← A
P2 ← #G0
SUB MUESTRE
A ← 55H
NO
A=0
SI
4DH ← #00H
RET
SUB RWDG0
INICIO.
SUB MUESTRE
A ← 55H
A=0 ?
NO
SI
4DH ← #04H
A ← 4EH
A ← A or 4EH
P0 ← A
P2 ← #G0
SUB MUESTRE
A ← 55H
A=0 ?
SI
NO
A ← #00H
A ← A or 4EH
4DH ← #00H
P0 ← A
P2 ← #G0
SUB MUESTRE
A ← 55H
A=0
NO
SI
4DH ← #00H
RET

FIG. 3.09

3.4.10 SUBRUTINAS FRWG1 y RWDG1. (Fig. 3.10). FRWG1 permite colocar en FORWARD a G1 es decir que la cinta del casete que en ella se encuentra avance desde el lado izquierdo de G1 hacia el lado derecho, el dato necesario será #20H en la localidad 4DH. Una vez que se ha ingresado a esta subrutina no se podrá salir de ella mientras no aparezca primero un 01H en la localidad 56H y luego aparezca un 00H en la misma localidad, esta localidad contiene la respuesta de G1.

SUBRUTINA RWDG1, permite colocar en REWIND a G1 es decir que la cinta del casete que en ella se encuentra avance desde el lado derecho de G1 hacia el lado izquierdo, el dato necesario será #40H en la localidad 4DH.

De igual manera que para FRWG1, una vez que se ha ingresado a esta subrutina no se podrá salir de ella mientras no aparezca primero un 01H en la localidad 56H y luego aparezca un 00H en la misma localidad, esta localidad contiene la respuesta de G1.

3.4.11 SUBRUTINAS UBICASG0 Y UBICASG1. (Fig. 3.11). Estas 2 subrutinas permiten atrapar al casete de cualesquiera de las 120 posiciones (X,Y) y ubicarlo frente G0 (0,2) o G1 (41,2) dependiendo de cual de las dos subrutinas fue ejecutada.

La subrutina UBICASG0 servirá exclusivamente a G0 y la otra subrutina a G1. La función que realiza esta subrutina se limitan única y exclusivamente a lo mencionado.

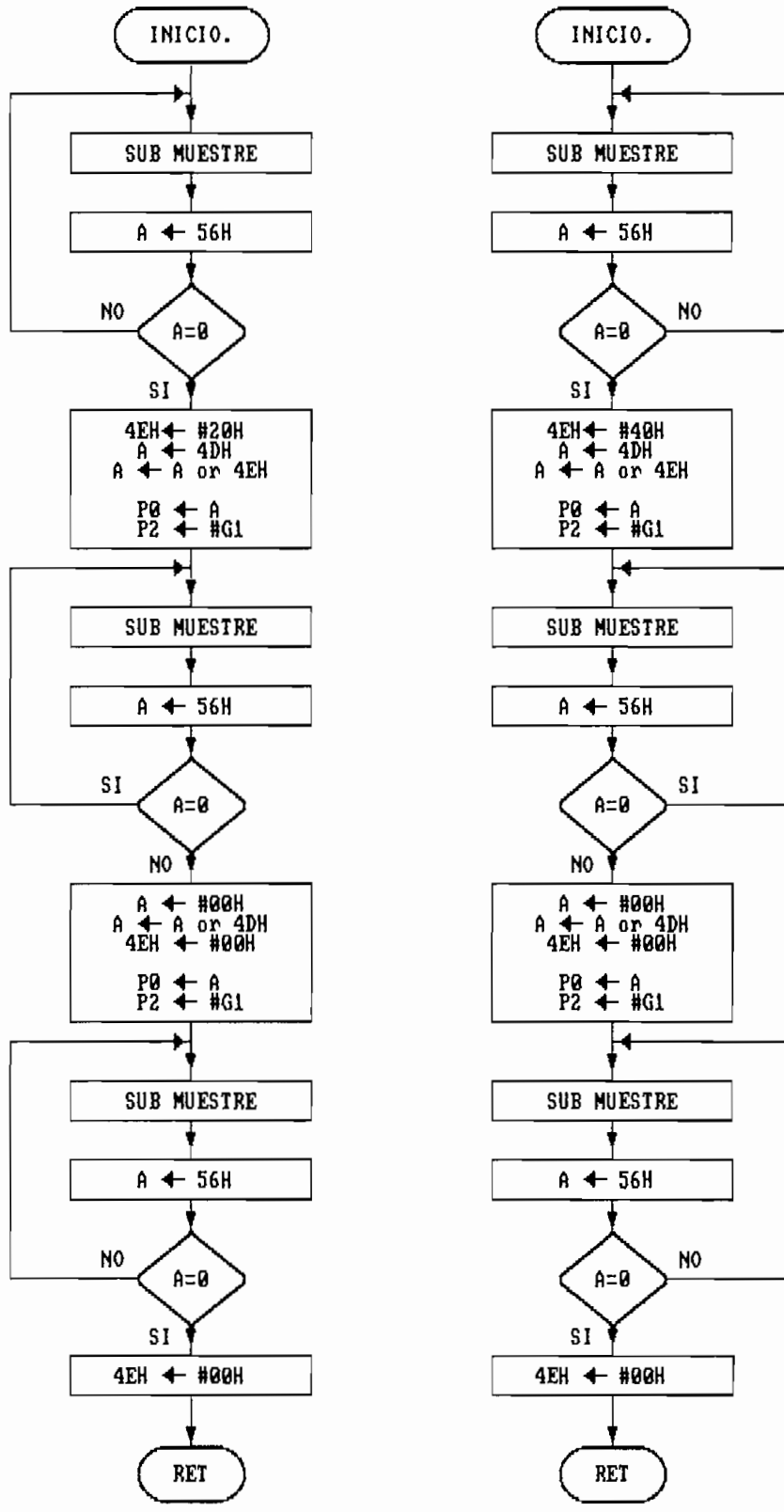

SUB FRWG1
INICIO.
SUB MUESTRE
A ← 56H
NO
A=0
SI
4EH ← #20H
A ← 4DH
A ← A or 4EH
P0 ← A
P2 ← #G1
SUB MUESTRE
A ← 56H
SI
A=0
NO
A ← #00H
A ← A or 4DH
4EH ← #00H
P0 ← A
P2 ← #G1
SUB MUESTRE
A ← 56H
NO
A=0
SI
4EH ← #00H
RET
SUB RWDG1
INICIO.
SUB MUESTRE
A ← 56H
NO
A=0
SI
4EH ← #40H
A ← 4DH
A ← A or 4EH
P0 ← A
P2 ← #G1
SUB MUESTRE
A ← 56H
SI
A=0
NO
A ← #00H
A ← A or 4DH
4EH ← #00H
P0 ← A
P2 ← #G1
SUB MUESTRE
A ← 56H
NO
A=0
SI
4EH ← #00H
RET

FIG. 3.10

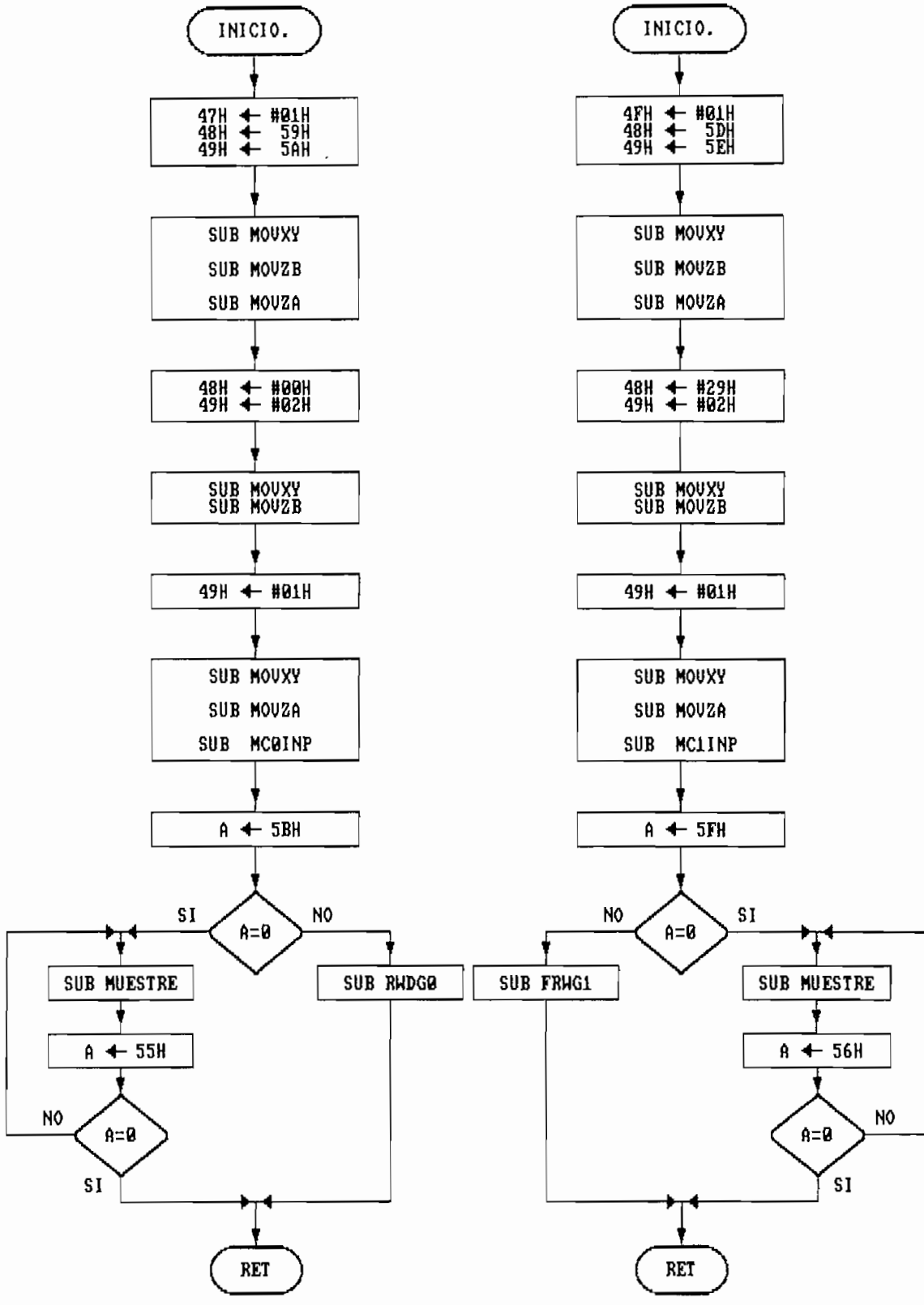

SUB UBICASG0
INICIO.
47H ← #01H
48H ← 59H
49H ← 5AH
SUB MOVXY
SUB MOVZB
SUB MOVZA
48H ← #00H
49H ← #02H
SUB MOVXY
SUB MOVZB
49H ← #01H
SUB MOVXY
SUB MOVZA
SUB MC0INP
A ← 5BH
SI
A=0
NO
SUB MUESTRE
A ← 55H
NO
A=0
SI
SUB RWDG0
RET
SUB UBICASG1
INICIO.
4FH ← #01H
48H ← 5DH
49H ← 5EH
SUB MOVXY
SUB MOVZB
SUB MOVZA
48H ← #29H
49H ← #02H
SUB MOVXY
SUB MOVZB
49H ← #01H
SUB MOVXY
SUB MOVZA
SUB MC1INP
A ← 5FH
NO
A=0
SI
SUB FRWG1
SUB MUESTRE
A ← 56H
A=0
NO
SI
RET

FIG. 3.11

Es necesario destacar que estas 2 subrutinas utilizan las siguientes subrutinas; MOVXY, MOVZA, MOVZB y MCOINP. Además la primera subrutina utiliza la subrutina RWDG0 y la segunda subrutina utiliza a FRWG1.

3.4.12 SUBRUTINAS PLAYG0 y PLAYG1. (Fig. 3.12). Al ejecutarse PLAYG0 primero se revisa que la señal de G0 se encuentre en 0 caso contrario se quedará al inicio de ésta.

Si pasa esta parte dependiendo de que lado del caso se necesita poner en play enviará el dato correspondiente, si el lado que se necesita escuchar es el "A" entonces pondrá el dato 09H en la localidad 4DH, si en cambio el lado que se necesita escuchar es el lado "B" pondrá 01H en la localidad 4DH, esto implica que se esta poniendo en PLAY el lado contrario al que fue solicitado, esto se debe a la ubicación relativa del casete respecto a G0 (para G1 no tendremos este problema).

Cuando se ejecuta PLAYG1, primero se revisa que la señal de G1 se encuentre en 0 caso contrario se quedará al inicio de ésta.

Si pasa esta parte dependiendo de que lado del caso se necesita poner en play enviará el dato correspondiente, si el lado que se necesita escuchar es el "A" entonces pondrá el dato 10H en la localidad 4EH, en cambio si el lado que se necesita escuchar es el B pondrá 90H en la localidad 4EH.

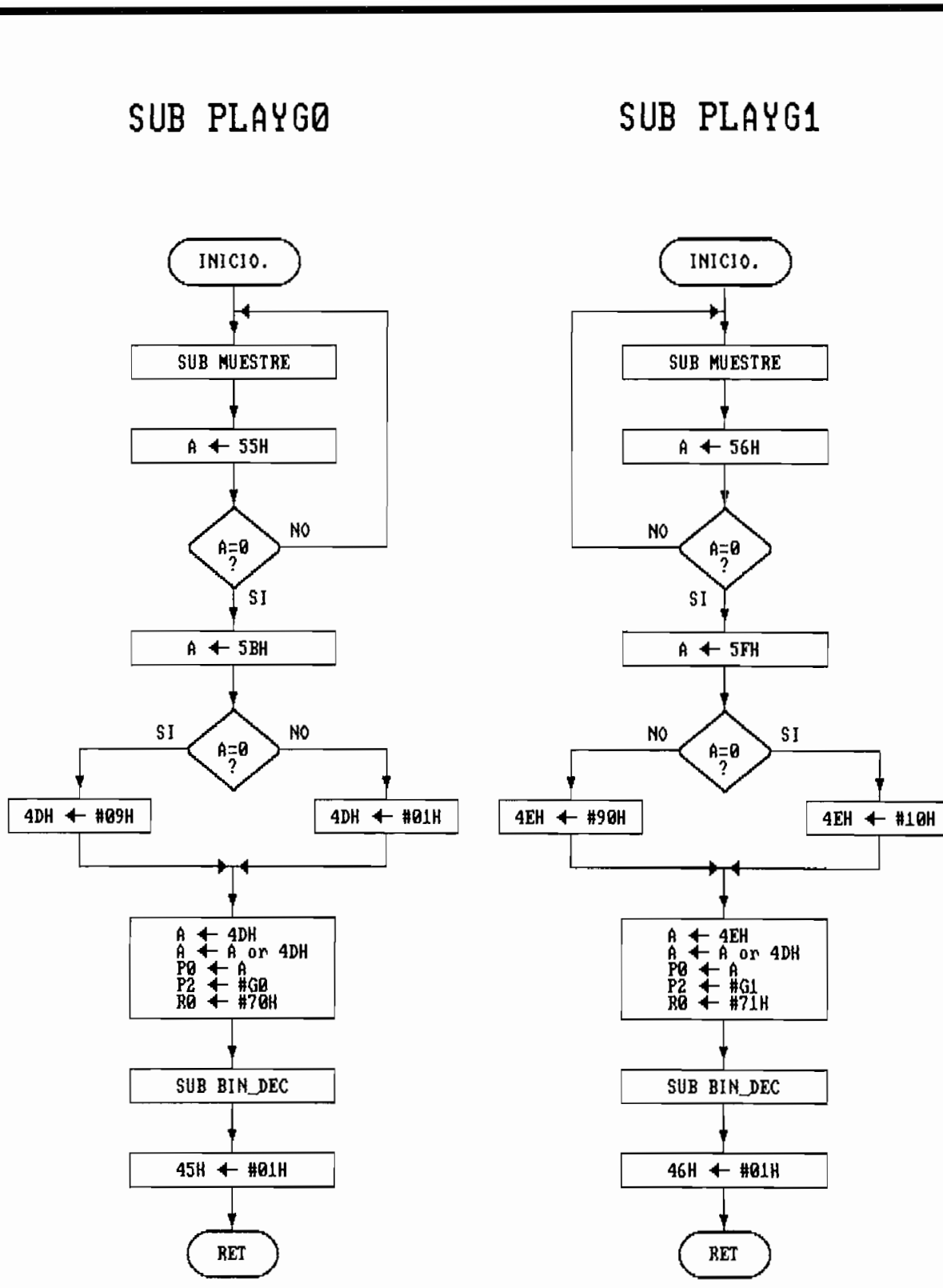

SUB PLAYG0
INICIO.
SUB MUESTRE
A ← 55H
A=0 ?
NO
SI
A ← 5BH
SI
A=0 ?
NO
4DH ← #09H
4DH ← #01H
A ← 4DH
A ← A or 4DH
P0 ← A
P2 ← #G0
R0 ← #70H
SUB BIN_DEC
45H ← #01H
RET
SUB PLAYG1
INICIO.
SUB MUESTRE
A ← 56H
NO
A=0 ?
SI
A ← 5FH
NO
A=0 ?
SI
4EH ← #90H
4EH ← #10H
A ← 4EH
A ← A or 4DH
P0 ← A
P2 ← #G1
R0 ← #71H
SUB BIN_DEC
46H ← #01H
RET

FIG. 3.12

Recordemos que debido a que G0 y G1 trabajan con el mismo latch, se sacará simultáneamente el dato que contenga la localidad 4DH y 4EH.

3.4.13 SUBRUTINA BIN_DEC. (Fig. 3.13). Esta subrutina es la encargada de convertir un dato que se encuentra en binario a tres dígitos decimales. Esta subrutina es utilizada únicamente por las subrutinas PLAYG0 y PLAYG1.

3.4.14 SUBRUTINA DEVCASG0. (Fig. 3.14). Esta subrutina realiza el trabajo contrario a la subrutina UBICASG0, una vez que ha terminado de ser escuchado el casete en G0 tiene que ser devuelto a su posición original previo a la reposición original de la cinta del casete. Esta subrutina se encargará de devolver la cinta su posición original, luego lo sacará de G0, moverá MX hasta la posición (00H,02H) y de ahí lo pasará a su posición original. Posteriormente borrará todas las localidades utilizadas por G0 dejando lista para una nueva selección.

3.4.15 SUBRUTINA DEVCASG1. (Fig. 3.15). Esta subrutina se encarga de devolver la cinta a su posición original, luego lo sacará de G1, moverá MZ hasta la posición (41,2) y de ahí lo pasará a su posición original. Posteriormente borrará todas las localidades de memoria utilizadas por el buffer de G1 dejando lista para una nueva selección. Esta subrutina realiza exactamente el trabajo contrario a la subrutina UBICASG1.

SUB BIN_DEC

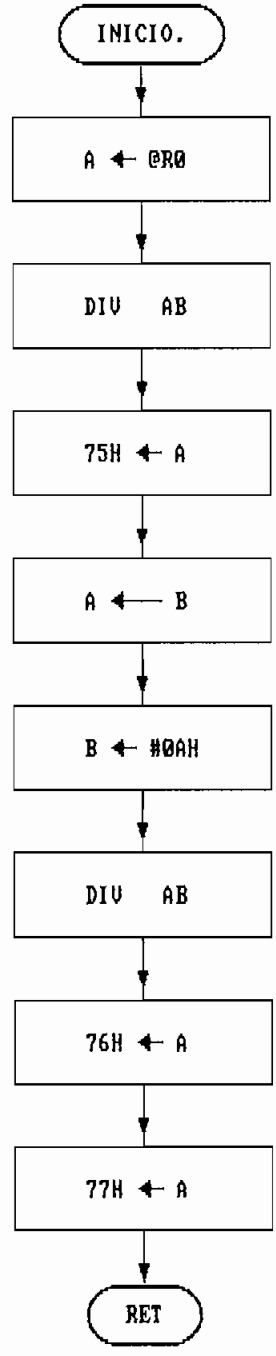


FIG. 3.13

SUB DEVCASG0

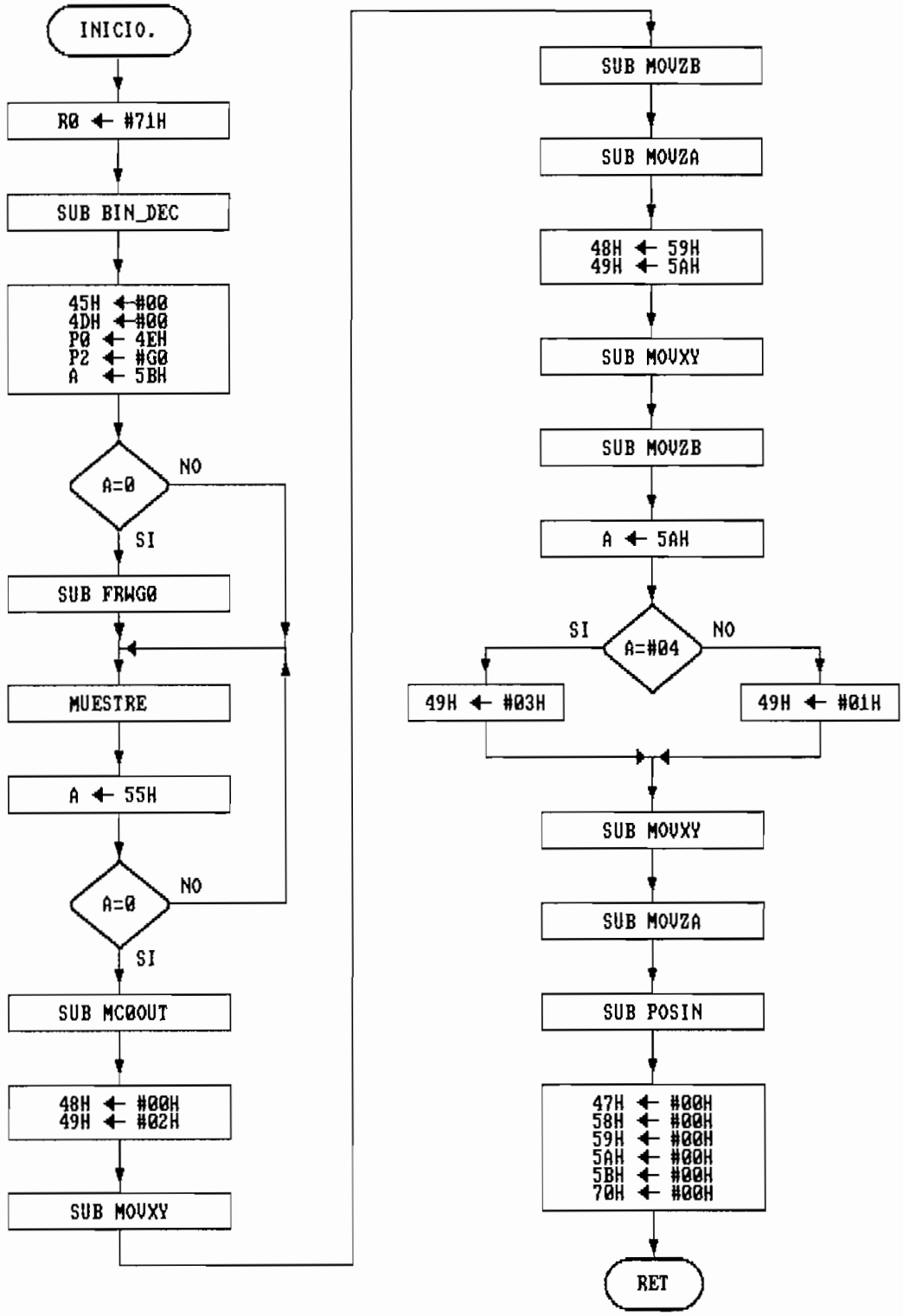


FIG. 3.14

SUB DEVCASG1

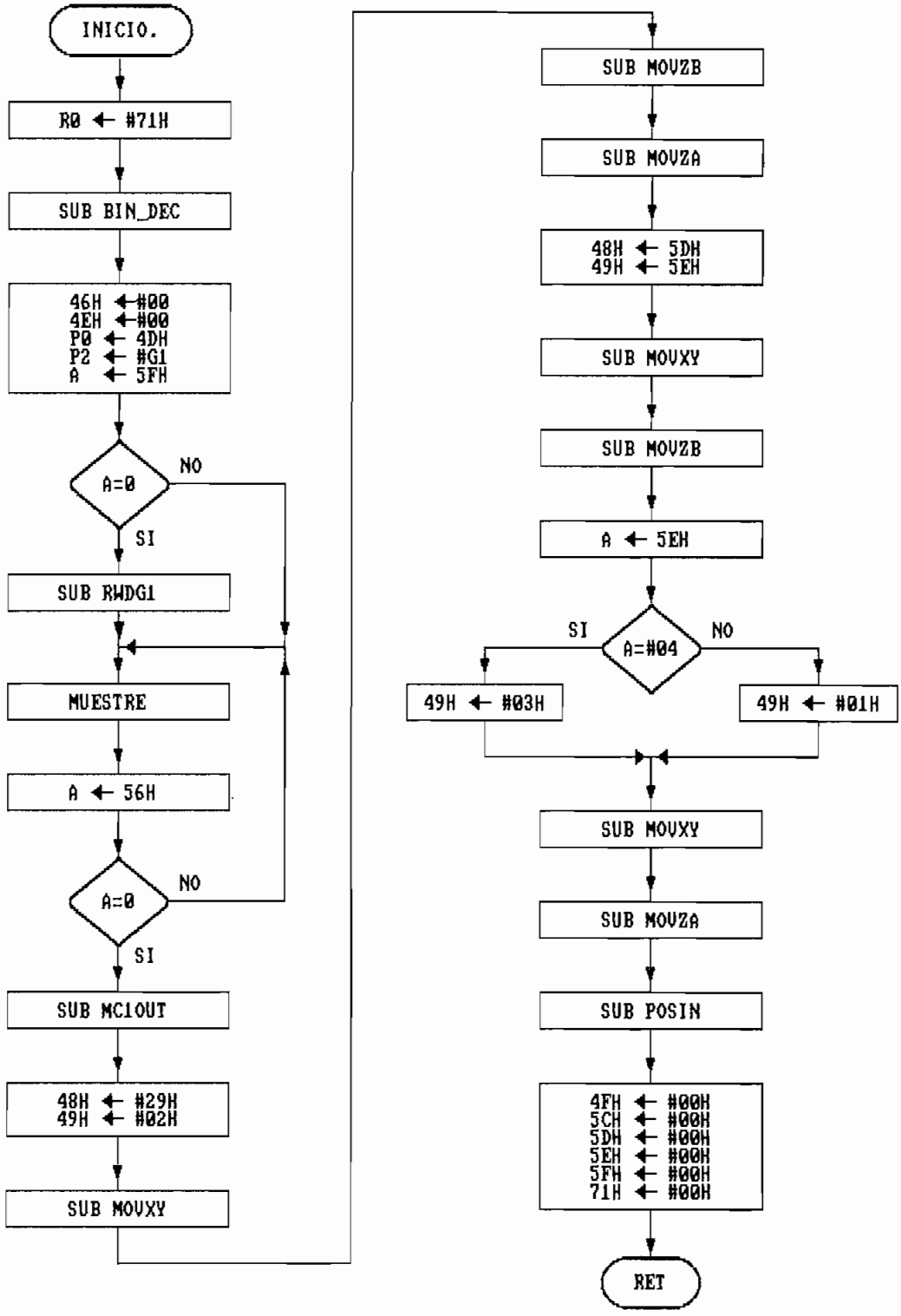


FIG. 3.15

3.4.16 SUBRUTINA TECLADO. (Fig. 3.16). Esta subrutina es controlada por la interrupción cero y es la encargada de permitir el ingreso de datos. Cuando recién energizamos el STM toda la memoria RAM del microcontrolador se encuentra con ceros, la primera selección que ingresemos corresponderá a 2 veces el número de casetes que posea el STM (por lo tanto máximo 240).

Cuando pulsamos el primer dígito, este dígito se escribe en el display de la derecha del primer grupo de 3 displays, el segundo dígito obligara a recorrer un display a la izquierda al primer dígito y se ubicará en el display de la derecha, el tercer dígito hará que el primer y segundo dígito recorran un display a la izquierda, los dígitos son aceptados siempre y cuando el número que forme la selección pedida no supere al número máximo de selecciones existentes caso contrario ni siquiera aparecerán sobre los displays (como la primera selección realizada corresponde al número máximo de selecciones que se podrá hacer, esta selección no podrá ingresar si es mayor a 240).

Una vez que el número correspondiente a la selección que nos encontramos haciendo se encuentra exhibido en el primer grupo de 3 displays, para que este número sea aceptado debemos pulsar "A" solo en este momento la selección realizada pasa a formar parte del buffer de selecciones, y aparecerá un cero en los displays que se encontraban exhibiendo el dato que estamos seleccionando.

SUBRUTINA TECLADO

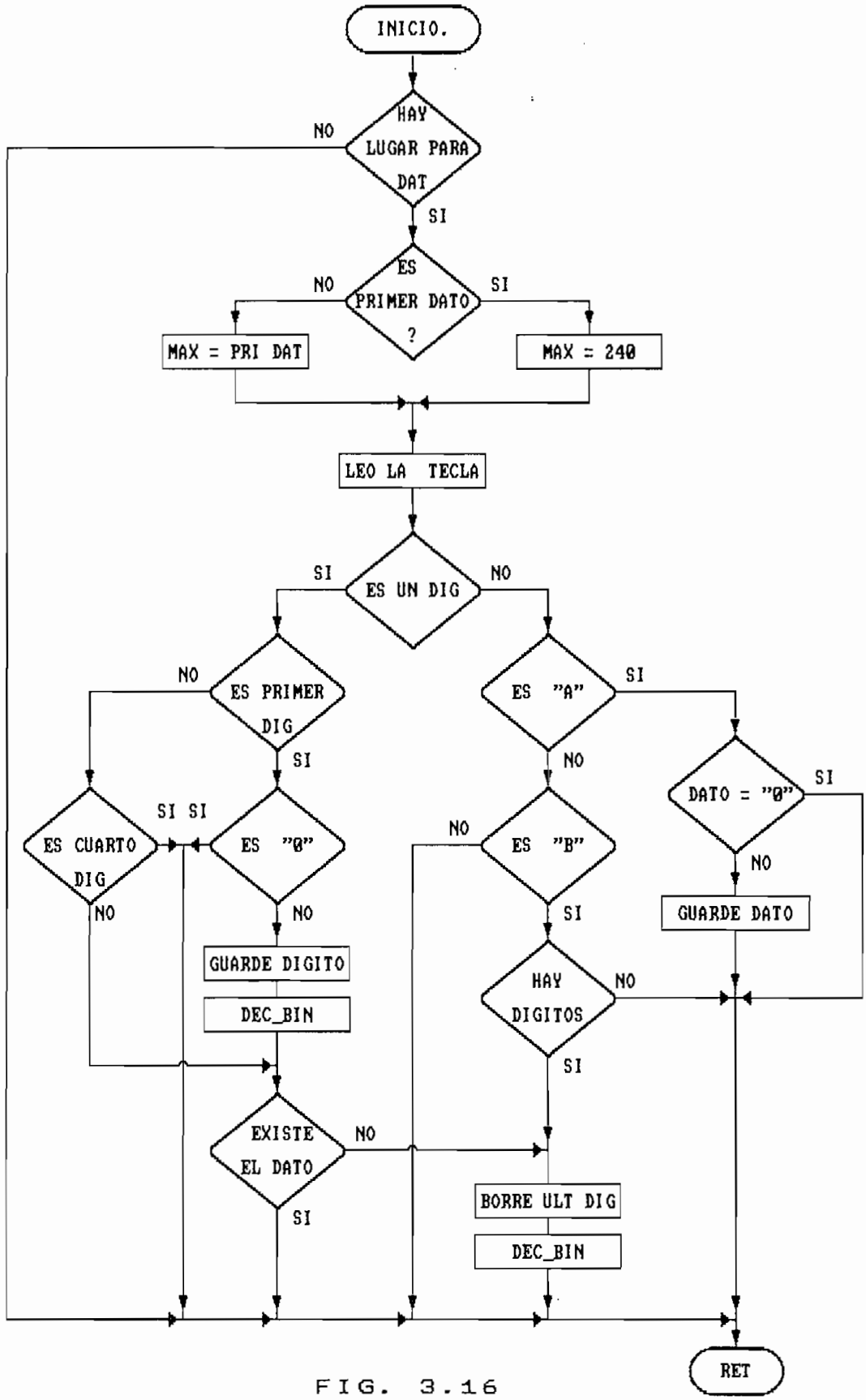


FIG. 3.16

Cuando se ha estado realizando una cierta selección y un dígito fue erróneo antes de pulsar la tecla "A" tenemos la posibilidad de pulsar la tecla "B", esta tecla borrará el dígito que se encuentre más a la derecha en ese momento sobre el primer grupo de displays.

El número máximo de selecciones que podemos pedir es de 5 (3 se almacenan en el buffer de selecciones, 1 en el buffer de G0 y 1 en el buffer de G1).

Se pudo haber colocado hasta 40 selecciones puesto que tenemos esa cantidad de localidades libres pero para nuestro caso no se requiere tanto pues lo que se necesita es verificar el funcionamiento del STM.

3.4.17 SUBRUTINA DEC_BIN. (Fig. 3.17). Esta subrutina se encarga de convertir los dígitos ingresados desde el teclado a su equivalente en hexadecimal y almacenarlos en la localidad 7BH, esta conversión es necesaria para poder verificar si el dato ingresado hasta ese momento existe o no. La subrutina DEC_BIN es utilizada exclusivamente por la subrutina TECLADO.

Nota: Si el dato no existe la última tecla presionada es denegada, hasta cuando sea un dato válido o hasta cuando se borre todos los dígitos que hasta ese momento se han ingresado por medio de la tecla "B". Si no existe ningún dígito la tecla "B" no hace absolutamente nada.

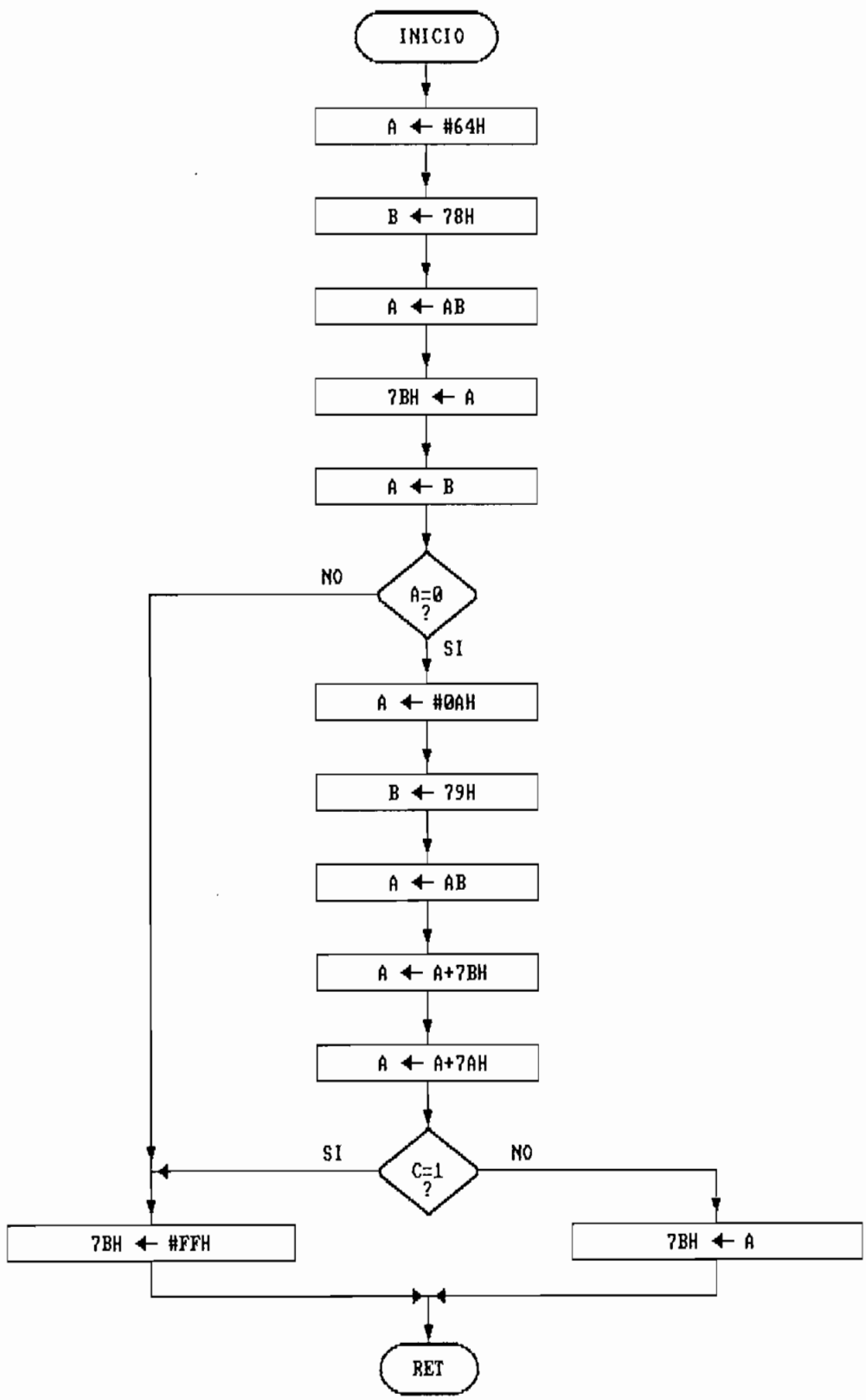

SUB DEC_BIN
INICIO
A ← #64H
B ← 78H
A ← AB
7BH ← A
A ← B
A=0 ?
NO
SI
A ← #0AH
B ← 79H
A ← AB
A ← A+7BH
A ← A+7AH
C=1 ?
SI
NO
7BH ← #FFH
7BH ← A
RET

FIG. 3.17

3.5 PROGRAMA.

El programa está en la misma secuencia que en 3.4.

```
; ************************************************************************
; ************************************************************************
; *******          PROGRAMA QUE REALIZA EL CONTROL DEL STM          ********
; ************************************************************************
; ************************************************************************

; DEFINICION DE ETIQUETAS PARA EL PROGRAMA PRINCIPAL Y SUBRUTINAS

MX      EQU     0F0H
MY      EQU     0F1H
MZ      EQU     0F2H
MC0     EQU     0F3H
MC1     EQU     0F4H
G0      EQU     0F5H
G1      EQU     0F6H
DISP1   EQU     0F7H
DISP2   EQU     0F8H
DISP3   EQU     0F9H
DISP4   EQU     0FAH
DISP5   EQU     0FBH
DISP6   EQU     0FCH
LATCH   EQU     0FFH
```

```
                ORG     0000H
                LJMP    INICIO
                ORG     0003H
                LCALL   TECLADO
                RETI
                ORG     000BH
                RETI
                ORG     0013H
                RETI
                ORG     001BH
                RETI
                ORG     0023H
                RETI

;               ************************
;               ************************
;               ** PROGRAMA PRINCIPAL **
;               ************************
;               ************************

INICIO:         MOV     R0,#7FH         ; ENCERA LA RAM
CLRAM:          MOV     @R0,#00
                DJNZ    R0,CLRAM

                MOV     IE,#81H         ; ACTIVO INT-0
                SETB    TCON.0          ; INT.0 ACTIVADA POR FLANCO
                MOV     SP,#1FH         ; MUEVO EL SP
                MOV     18H,#7FH        ; PUNT.  DE DATOS (SUB TECLADO)
```

```
PROGPRIN:   LCALL   POSIN       ; MUEVO A LAS POSICIONES INICI.
            LCALL   MC0OUT
            LCALL   MC1OUT
            AJMP    E00         ; INGRESO AL PROGRAMA.
E01:        LCALL   MUESTRE     ; SUB. RES, PXY, DIS.
E00:        MOV     A,47H       ; ESTA LIBRE G0?
            JNZ     E09         ; NO => E09.
            MOV     A,58H       ; SI => HAY DATOS PARA G0?
            JNZ     E02         ; SI HAY DATOS => E02.
            AJMP    E05         ; NO HAY DATOS => E05.
E02:        LCALL   UBICASG0    ; UBICO EL CASETE EN G0.
E03:        MOV     A,46H       ; G1 ESTA EN PLAY?
            JZ      E11         ; NO => E11.
            MOV     A,56H       ; SI => ACABO G1?
            JZ      E01         ; NO => E01.
            MOV     A,47H       ; SI => G0 ESTA LIBRE?
            JZ      E04         ; NO => E04.
            LCALL   PLAYG0      ; PONGA EN PLAY G0.
E04:        LCALL   DEVCASG1    ; REGRESO EL CASETE DE G1.
            LCALL   POSIN       ; MUEVA A POSICIONES INICIALES.
            AJMP    E05         ; VAYA A E05.
E09:        MOV     A,45H       ; G0 ESTA EN PLAY?
            JZ      E03         ; NO => E03.
            AJMP    E07         ; SI => E07.
E11:        LCALL   PLAYG0      ; PONGA EN PLAY G0.
            AJMP    E05         ; VAYA A E05.
E05:        LCALL   MUESTRE     ; SUB. RES, PXY, DIS.
            MOV     A,4FH       ; ESTA LIBRE G1?
```

```
                JNZ     E1Ø             ; NO => E1Ø.
                MOV     A,5CH           ; SI => HAY DATOS PARA G1?
                JNZ     EØ6             ; SI HAY DATOS => EØ2.
                AJMP    EØ1             ; NO HAY DATOS => EØ5.
EØ6:            LCALL   UBICASG1        ; UBICO EL CASETE EN G1.
EØ7:            MOV     A,45H           ; GØ ESTA EN PLAY?
                JZ      E12             ; NO => E11.
                MOV     A,55H           ; SI => ACABO G1?
                JZ      EØ5             ; NO => EØ5.
                MOV     A,4FH           ; SI => G1 ESTA LIBRE?
                JZ      EØ8             ; NO => EØ8.
                LCALL   PLAYG1          ; PONGA EN PLAY G1.
EØ8:            LCALL   DEVCASGØ        ; REGRESO EL CASETE DE GØ.
                LCALL   POSIN           ; MUEVA A POSICIONES INICIALES.
                AJMP    EØ1             ; VAYA A EØ1.
E1Ø:            MOV     A,46H           ; G1 ESTA EN PLAY?
                JZ      EØ7             ; NO => EØ7.
                AJMP    EØ3             ; SI => EØ3.
E12:            LCALL   PLAYG1          ; PONGA EN PLAY G1.
                AJMP    EØ1             ; VAYA A EØ1.

; Esta subrutina POSIN, permite colocar en las posiciones iniciales es-
; to implica la posición inicial en X, en Y, y en Z.
POSIN:          LCALL   MOVZA
                MOV     48H,#2Ø
                MOV     49H,#Ø2
                LCALL   MOVXY
                RET
```

```
; \/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
; \/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
; \/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
; \/\/                                                            \/\/
; \/\/                 AQUI EMPIEZA LAS SUBRUTINAS                \/\/
; \/\/                                                            \/\/
; \/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
; \/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
; \/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
; =====================================================================
; ==                                                                 ==
; ==                     SUBRUTINAS DE MOVIMIENTO                    ==
; ==               (AQUI SE ENCUENTRAN VARIAS SUBRUTINAS)            ==
; ==                                                                 ==
; ==  MOVXY: PERMITE UBICAR EL BRAZO DE SELECCION EN LA FILA ESPE-   ==
; ==         CIFICADA POR LA DIRECCION 49H Y EN LA COLUMNA ESPECI-   ==
; ==         FICADA POR LA DIRECCION 48H.                            ==
; ==         (EXISTEN 5 FILAS Y 42 COLUMNAS)                         ==
; ==                                                                 ==
; ==  MOVZA:  PERMITE SUBIR EL MECANISMO Z DE SUJECION DEL CASETE.   ==
; ==  MOVZB:  PERMITE BAJAR EL MECANISMO Z DE SUJECION DEL CASETE.   ==
; ==                                                                 ==
; ==  MC0INP:  PERMITE INTRODUCIR EL CASETE EN LA GRABADORA CERO.    ==
; ==  MC0OUT:  PERMITE RETIRAR EL CASETE DE LA GRABADORA CERO.       ==
; ==                                                                 ==
; ==  MC1INP:  PERMITE INTRODUCIR EL CASETE EN LA GRABADORA UNO.     ==
; ==  MC1OUT:  PERMITE RETIRAR EL CASETE DE LA GRABADORA UNO.        ==
; =====================================================================
```

```
; Esta subrutina permite colocar a MZ en una cierta posición X,Y (de-
; terminadas por las localidades 48H y 49H respectivamente). Determina
; el sentido de giro de cada uno de los motores en base a las posicio-
; nes iniciales (localidades 40H y 41H). El programa no saldrá de esta
; subrutina mientras no exista un 1 en las localidades 50H y 51H.

MOVXY:      CLR     C           ; Resta, posición inicial de X
            MOV     A,48H       ; de la posición final, si hay
            SUBB    A,40H       ; carry implica que la posición
            JZ      MOVY        ; deseada se encuentra a la iz-
            MOV     A,48H       ; quierda caso contrario se en-
            JNC     XDER        ; cuentra a la derecha. Si no
            ORL     A,#40H      ; hay carry y la respuesta es 0
            AJMP    OUTX        ; esta parte no se ejecuta.
XDER:       ORL     A,#80H      ; Movimiento a la derecha.
OUTX:       MOV     40H,48H     ; Movimiento a la izquierda.
            MOV     P0,A
            MOV     P2,#MX
MOVY:       CLR     C           ; Resta la posición inicial de
            MOV     A,49H       ; la posición final, si hay ca-
            SUBB    A,41H       ; rry mueve hacia arriba, caso
            JZ      WAITXY      ; contrario mueve hacia abajo.
            MOV     A,49H       ; Si la respuesta fue cero no
            JNC     YDOWN       ; se ejecuta esta subrutina y
            ORL     A,#08H      ; salta a la salida (WAITXY).
            AJMP    OUTY
YDOWN:      ORL     A,#10H      ; Movimiento hacia abajo.
OUTY:       MOV     41H,49H
```

```
                MOV     P0,A
                MOV     P2,#MY
WAITXY:         LCALL   MUESTRE         ; Sale de esta subrutina cuan-
                MOV     A,51H           ; do la respuesta de MX y MY
                JZ      WAITXY          ; son uno.
                MOV     A,50H
                JZ      WAITXY
                RET

; Esta subrutina controla el movimiento del motor Z. Si es llamada como
; MOVZA el mecanismo en Z sube, en cambio si es llamada como MOVZB el
; mecanismo en Z baja. El programa no saldrá de esta subrutina mientras
; en la localidad 52H no aparezca un 1 que es la respuesta de MZ indi-
; cando que ya llegó a la posición solicitada.

MOVZA:          MOV     4AH,#09H        ; Mecanismo Z hacia arriba.
                AJMP    OUTMZ
MOVZB:          MOV     4AH,#06H        ; Mecanismo Z hacia abajo.
OUTMZ:          MOV     P0,4AH
                MOV     P2,#MZ
REFMZ:          LCALL   MUESTRE         ; Permanece mostrando mientras
                MOV     A,52H           ; la respuesta no sea uno.
                JZ      REFMZ
                RET

; Esta subrutina permite introducir o retirar el casete de la grabadora
; G0, cuando llamamos a MC0INP el casete será introducido en la graba-
; dora y cuando llamamos a MC0OUT el casete será retirado de la graba-
```

```
; dora G0. El programa no saldrá de esta subrutina mientras en la loca-
; lidad 53H no aparezca un 1  que es el  que indicará que el  mecanismo
; MC0 ya llegó a la posición solicitada.

MC0INP:     LCALL   MUESTRE         ; Permanece mostrando  mientras
            MOV     A,54H           ; las respuestas  de  MC0 y MC1
            ORL     A,53H           ; no sean cero.
            JNZ     MC0INP
            MOV     4EH,#01         ; MC0 dentro de G0.
            AJMP    OUTMC0
MC0OUT:     LCALL   MUESTRE         ; Similar que para MC0INP.
            MOV     A,54H
            ORL     A,53H
            JNZ     MC0OUT
            MOV     4EH,#02         ; MC0 fuera de G0.
OUTMC0:     MOV     A,4CH           ; Saco el  dato  que tenía  MC1
            ORL     A,4EH           ; junto  con  el  nuevo dato de
            MOV     P0,A            ; MC0.
            MOV     P2,#MC0
REPMC0:     LCALL   MUESTRE         ; Espera que  la  respuesta sea
            MOV     A,53H           ; un uno.
            JZ      REPMC0
            MOV     P0,#00
            MOV     P2,#MC0
AGAIN0:     LCALL   MUESTRE         ; Espera que  la  respuesta  se
            MOV     A,53H           ; haga cero.
            JNZ     AGAIN0
            RET
```

```
; Esta subrutina hace  lo mismo que la subrutina  anterior pero para G1.
; El programa no saldrá de  esta subrutina  mientras 54H no aparezca un
; 01 que indica que MC0 ya llegó a la posición solicitada.

MC1INP:     LCALL   MUESTRE         ; Permanece  mostrando mientras
            MOV     A,53H           ; las respuestas de  MC1 y MC1
            ORL     A,54H           ; no sean cero.
            JNZ     MC1INP
            MOV     4CH,#04         ; MC1 dentro de G1.
            AJMP    OUTMC1
MC1OUT:     LCALL   MUESTRE         ; Similar que para MC1INP.
            MOV     A,53H
            ORL     A,54H
            JNZ     MC1OUT
            MOV     4CH,#08         ; MC1 fuera de G1.
OUTMC1:     MOV     A,4BH           ; Saca  el dato  que tenía  MC0
            ORL     A,4CH           ; junto con  el  nuevo  dato de
            MOV     P0,A            ; MC1.
            MOV     P2,#MC1
REPMC1:     LCALL   MUESTRE         ; Espera que  la respuesta  sea
            MOV     A,54H           ; un uno.
            JZ      REPMC1
            MOV     P0,#00
            MOV     P2,#MC1
AGAIN1:     LCALL   MUESTRE         ; Espera  que  la respuesta  se
            MOV     A,54H           ; haga cero.
            JNZ     AGAIN1
            RET
```

SUBRUTINAS DE CONTROL DE LAS GRABADORAS

(AQUI SE ENCUENTRAN VARIAS SUBRUTINAS)

UBICASG0: PERMITE UBICAR UN CIERTO CASETE QUE SE ENCUENTRA EN LA POSICION (X,Y) DENTRO DE LA GRABADORA G0.
POSICION X: LOCALIDAD 59H
POSICION Y: LOCALIDAD 5AH

UBICASG1: PERMITE UBICAR UN CIERTO CASETE QUE SE ENCUENTRA EN LA POSICION (X,Y) DENTRO DE LA GRABADORA G1.
POSICION X: LOCALIDAD 5DH
POSICION Y: LOCALIDAD 5EH

PLAYG0: PONE LA GRABADORA G0 EN PLAY.

PLAYG1: PONE LA GRABADORA G1 EN PLAY.

DEVCASG0: RETIRA EL CASETE DE LA GRABADORA G0 Y DEVUELVE A SU POSICION ORIGINAL.

DEVCASG1: RETIRA EL CASETE DE LA GRABADORA G1 Y DEVUELVE A SU POSICION ORIGINAL.

```
FRWG0:      LCALL   MUESTRE     ; No ingresa a esta subrutina mien-
            MOV     A,55H       ; tras la respuesta no sea cero.
            JNZ     FRWG0
            MOV     4DH,#02H    ; Pone FORWARD a  G0 y  realiza una
            MOV     A,4EH       ; OR con el dato que tenía G1 y sa-
            ORL     A,4DH       ; co por el puerto.
            MOV     P0,A
            MOV     P2,#G0
WAITFRWG0:  LCALL   MUESTRE     ; Permanece  mostrando  mientras no
            MOV     A,55H       ; haya un uno como respuesta.
            JZ      WAITFRWG0
            MOV     A,#00       ; Si la respuesta fue uno  entonces
            ORL     A,4EH       ; desactiva a G0 y pone el dato
            MOV     4DH,#00H    ; que tenía G1.
            MOV     P0,A
            MOV     P2,#G0
WAITSTEG0:  LCALL   MUESTRE     ; Permanece  mostrando  mientras la
            MOV     A,55H       ; respuesta no sea un cero.
            JNZ     WAITSTEG0
            RET

RWDG0:      LCALL   MUESTRE     ; No ingresa a esta subrutina mien-
            MOV     A,55H       ; tras la respuesta no sea cero.
            JNZ     RWDG0
            MOV     4DH,#04H    ; Pone REWIND a G0,  y  realiza una
            MOV     A,4EH       ; OR con el dato que tenía G1 y sa-
            ORL     A,4DH       ; ca por el puerto.
            MOV     P0,A
```

```
                MOV     P2,#G0
WAITRWDG0:      LCALL   MUESTRE     ; Permanece mostrando mientras no
                MOV     A,55H       ; haya un uno como respuesta.
                JZ      WAITRWDG0
                MOV     A,#00       ; Si la respuesta fue uno entonces
                ORL     A,4EH       ; desactiva a G0 y pone el dato que
                MOV     4DH,#00H    ; tenía G1.
                MOV     P0,A
                MOV     P2,#G0
HOLDSTBG0:      LCALL   MUESTRE     ; Permanece mostrando mientras la
                MOV     A,55H       ; respuesta no sea un cero.
                JNZ     HOLDSTBG0
                RET

FRWG1:          LCALL   MUESTRE     ; No ingresa a esta subrutina mien-
                MOV     A,56H       ; tras la respuesta no sea cero.
                JNZ     FRWG1
                MOV     4EH,#20H    ; Pone FORWARD a G1, realiza una OR
                MOV     A,4DH       ; con el dato que tenía G0 y saca
                ORL     A,4EH       ; por el puerto.
                MOV     P0,A
                MOV     P2,#G1
WAITFRWG1:      LCALL   MUESTRE     ; Permanece mostrando mientras no
                MOV     A,56H       ; haya un uno como respuesta.
                JZ      WAITFRWG1
                MOV     A,#00       ; Si la respuesta fue uno entonces
                ORL     A,4DH       ; desactiva a G1 y pongo el dato
                MOV     4EH,#00H    ; que tenía G0.
```

```
                MOV     P0,A
                MOV     P2,#G1
WAITSTBG1:      LCALL   MUESTRE     ; Permanece mostrando mientras la
                MOV     A,56H       ; respuesta no sea un cero.
                JNZ     WAITSTBG1
                RET

RWDG1:          LCALL   MUESTRE     ; No ingresa a esta subrutina mien-
                MOV     A,56H       ; tras la respuesta no sea cero.
                JNZ     RWDG1
                MOV     4EH,#40H    ; Pone REWIND a G1, realiza una OR
                MOV     A,4DH       ; con el dato que tenía G0 y saca
                ORL     A,4EH       ; por el puerto.
                MOV     P0,A
                MOV     P2,#G1
WAITRWDG1:      LCALL   MUESTRE     ; Permanece mostrando mientras no
                MOV     A,56H       ; haya un uno como respuesta.
                JZ      WAITRWDG1
                MOV     A,#00       ; Si la respuesta fue uno entonces
                ORL     A,4DH       ; desactiva a G1 y pongo el dato
                MOV     4EH,#00H    ; que tenía G0.
                MOV     P0,A
                MOV     P2,#G1
HOLDSTBG1:      LCALL   MUESTRE     ; Permanece mostrando mientras la
                MOV     A,56H       ; respuesta no sea un cero.
                JNZ     HOLDSTBG1
                RET
```

```
; Esta subrutina UBICASG0, permite pasar de una cierta posición (X,Y)
; hacia dentro de la G0. Y la segunda subrutina UBICASG1 hace exacta-
; mente lo mismo con la única diferencia que ubicará el casete en la
; grabadora G1.

UBICASG0:   MOV     47H,#01     ; Pone una señal de OCUPADA G0.
            MOV     48H,59H     ; Da ubicación del casete.
            MOV     49H,5AH     ; (Columna, Fila).
            LCALL   MOVXY       ; Mueve hacia esas posiciones.
            LCALL   MOVZB       ; Atrapa el casete solicitado.
            LCALL   MOVZA       ; Sube el casete solicitado.
            MOV     48H,#00     ; Va frente a la grabadora G0.
            MOV     49H,#02
            LCALL   MOVXY
            LCALL   MOVZB       ; Deposita el casete.
            MOV     49H,#01     ; Separa el casete de MZ.
            LCALL   MOVXY
            LCALL   MOVZA       ; Sube MZ.
            LCALL   MC0INP      ; Introduce el casete en G0.
            MOV     A,5BH       ; Si el lado del casete solici-
            JZ      OUTUBI0     ; tado es el A  =>  pone G0 en
            LCALL   RWDG0       ; REWIND.
OUTUBI0:    LCALL   MUESTRE     ; Mientras la señal de MC0 no
            MOV     A,53H       ; sea cero, no sale de esta
            JNZ     OUTUBI0     ; subrutina (UBICASG0).
            RET

UBICASG1:   MOV     4FH,#01     ; Pone una señal de OCUPADA G1.
```

```
                MOV     48H,5DH         ; Da ubicación del casete.
                MOV     49H,5EH         ; (Columna, Fila).
                LCALL   MOVXY           ; Mueve hacia esas posiciones.
                LCALL   MOVZB           ; Atrapa el casete solicitado.
                LCALL   MOVZA           ; Sube el casete solicitado.
                MOV     48H,#41         ; Va frente a la grabadora G1.
                MOV     49H,#02
                LCALL   MOVXY
                LCALL   MOVZB           ; Deposita el casete.
                MOV     49H,#01         ; Separa el casete de MZ.
                LCALL   MOVXY
                LCALL   MOVZA           ; Sube MZ.
                LCALL   MC1INP          ; Introduce el casete en G1.
                MOV     A,5FH           ; Si el lado del casete solici-
                JZ      OUTUBI1         ; tado es el B   =>  pone G1 en
                LCALL   FRWG1           ; FORWARD.
OUTUBI1:        LCALL   MUESTRE         ; Mientras la  señal  de MC1 no
                MOV     A,56H           ; sea  cero,  no  sale de  esta
                JNZ     OUTUBI1         ; subrutina (UBICASG1).
                RET

; De  las  dos  subrutinas  que se encuentran a continuación la primera
; permite colocar la grabadora G0 en PLAY y la segunda G1 en PLAY.

PLAYG0:         LCALL   MUESTRE         ; Verifica  si  la respuesta se
                MOV     A,55H           ; encuentra  en 0.  Si  lo está
                JNZ     PLAYG0          ; entonces continua.
                MOV     A,5EH           ; Si el lado que se desea escu-
```

```
                JZ      EDGEA           ; char es el A => por la ubica-
                MOV     4DH,#01H        ; ción  de G0 y del casete debe
                AJMP    OUTPLAYG0       ; poner el lado B en G0.
EDGEA:          MOV     4DH,#09H        ; Pone PLAY, LADO B.
OUTPLAYG0:      MOV     A,4DH           ; Realiza una OR con los  datos
                ORL     A,4EH           ; que posee G1, dado que traba-
                MOV     P0,A            ; jan con el mismo latch.
                MOV     P2,#G0
                MOV     R0,#70H         ; Pone  los  dígitos correspon-
                LCALL   BIN_DEC         ; dientes para que sean mostra-
                MOV     45H,#01         ; dos en los displays.
                RET

PLAYG1:         LCALL   MUESTRE         ; Verifica  si la respuesta  se
                MOV     A,56H           ; encuentra en 0.   Si lo está
                JNZ     PLAYG1          ; entonces continua.
                MOV     A,5FH           ; Si el lado que se desea escu-
                JZ      LADOA           ; char es el A => por la ubica-
                MOV     4EH,#90H        ; cion de G1 y del casete  debe
                AJMP    OUTPLAYG1       ; poner el lado A en G1.
LADOA:          MOV     4EH,#10H        ; Pone PLAY, LADO A.
OUTPLAYG1:      MOV     A,4EH           ; Realiza una OR con  los datos
                ORL     A,4DH           ; que posee G0, dado que traba-
                MOV     P0,A            ; jan con el mismo latch.
                MOV     P2,#G1
                MOV     R0,#71H         ; Pone  los  dígitos correspon-
                LCALL   BIN_DEC         ; dientes para que sean mostra-
                MOV     46H,#01H        ; dos en los displays.
```

```
                RET

BIN_DEC:        MOV     A,@R0           ; Esta subrutina convierte un
                MOV     B,#100          ; número binario ubicado en la
                DIV     AB              ; localidad indicada por R0 y
                MOV     75H,A           ; almacena:
                MOV     A,B             ; Centenas en la localidad 75H.
                MOV     B,#10           ; Decenas en la localidad 76H.
                DIV     AB              ; Unidades en la localidad 77H.
                MOV     76H,A
                MOV     77H,B
                RET

; Estas 2 subrutinas permiten devolver el casete a su posición original
; tanto en la ubicación física como en el estado de la cinta del casete
; La primera subrutina (DEVCASG0) trabaja solo para la grabadora G0.
; La segunda subrutina (DEVCASG1) trabaja solo para la grabadora G1.

DEVCASG0:       MOV     R0,#71H         ; Exhibe en los displays lo que
                LCALL   BIN_DEC         ; tiene G1.
                MOV     45H,#00H        ; Borra señal de PLAY de G0.
                MOV     4DH,#00H        ; Borra dato de G0.
                MOV     P0,4EH          ; Saca los datos que tenía G1 y
                MOV     P2,#G0          ; desactiva a G0.
                MOV     A,5BH           ; Si el lado escuchado fue el
                JNZ     SIDEB           ; "A" => antes de devolver el
                LCALL   FRWG0           ; casete activa FORWARD.
SIDEB:          LCALL   MUESTRE         ; Permanece mostrando mientras
```

```
                MOV     A,55H           ; no termine FORWARD.
                JNZ     SIDEB
                LCALL   MC2OUT          ; Retira el casete de G0.
                MOV     48H,#00H        ; Mueve MZ a la posición (0,2).
                MOV     49H,#02H
                LCALL   MOVXY
                LCALL   MOVZB           ; Atrapa el casete.
                LCALL   MOVZA           ; Sube el casete.
                MOV     48H,59H         ; Regresa el  casete a su posi-
                MOV     49H,5AH         ; ción original.
                LCALL   MOVXY
                LCALL   MOVZB
                MOV     A,5AH           ; Libera el casete (dependiendo
                CJNE    A,#04H,F12G0    ; de  la fila a que pertenece).
                MOV     49H,#03         ; Si fue la fila 4 me mueve ha-
                AJMP    OUTDEVG0        ; cia la 3,  fue  fila 0 ó 2 se
F12G0:          MOV     49H,#01H        ; mueve hacia fila 1.
OUTDEVG0:       LCALL   MOVXY
                LCALL   MOVZA           ; Libera el casete.
                LCALL   POSIN           ; Retorna a posición inicial.
ESPEREG0:       LCALL   MUESTRE         ; Permanece  mostrando mientras
                MOV     A,55H           ; no se ponga  en 0 la señal de
                JNZ     ESPEREG0        ; respuesta de G0.
                MOV     47H,#00         ; Pone en cero todos  las loca-
                MOV     58H,#00         ; lidades utilizadas por G0.
                MOV     59H,#00
                MOV     5AH,#00
                MOV     5BH,#00
```

```
                MOV     70H,#00
                RET

DEVCASG1:       MOV     R0,#70H         ; Exhibe en los displays lo que
                LCALL   BIN_DEC         ; tenga G1.
                MOV     46H,#00         ; Borra señal de PLAY de G1
                MOV     4EH,#00H        ; Borra dato de G1.
                MOV     P0,4DH          ; Saca los datos que tenía G0 y
                MOV     P2,#G1          ; desactiva a G1.
                MOV     A,5FH           ; Si  el lado  escuchado fue el
                JNZ     SIDEBG1         ; "A"  =>  antes de devolver el
                LCALL   FWDG1           ; casete activa REWIND.
SIDEBG1:        LCALL   MUESTRE         ; Permanece  mostrando mientras
                MOV     A,56H           ; no termine REWIND.
                JNZ     SIDEBG1
                LCALL   MC1OUT          ; Retira el casete de G1.
                MOV     48H,#29H        ; Mueve el brazo de selección
                MOV     49H,#02H        ; a la posición (41,02).
                LCALL   MOVXY
                LCALL   MOVZB           ; Atrapa el casete.
                LCALL   MOVZA           ; Sube el casete.
                MOV     48H,5DH         ; Regresa  el casete a su posi-
                MOV     49H,5EH         ; ción original.
                LCALL   MOVXY
                LCALL   MOVZB
                MOV     A,5EH           ; Libera el casete (dependiendo
                CJNE    A,#04H,F12G1    ; de la fila que corresponda).
                MOV     49H,#03         ; Si  fue la fila 4  mueve  ha-
```

```
                AJMP    OUTDEVG1        ; cia 3, fue fila 0 ó 2 se mue-
F12G1:          MOV     49H,#01H        ; ve hacia la fila 1.
OUTDEVG1:       LCALL   MOVXY
                LCALL   MOVZA           ; Libera el casete.
                LCALL   POSIN           ; Retorna a posición inicial.
ESPEREG1:       LCALL   MUESTRE         ; Permanece  mostrando mientras
                MOV     A,56H           ; no se ponga  en 0 la señal de
                JNZ     ESPEREG1        ; respuesta de G1.
                MOV     4FH,#00         ; Pone  en cero todos las loca-
                MOV     5CH,#00         ; lidades utilizadas por G1.
                MOV     5DH,#00
                MOV     5EH,#00
                MOV     5FH,#00
                MOV     71H,#00
                RET
```

```
;=============================================================
;=                                                           =
;=                        SUB   MUESTRE                      =
;=                                                           =
;=  RES                                                      =
;=  ACTUALIZA LA RESPUESTA DE LOS ELEMENTOS.                 =
;=                                                           =
;=  PXY                                                      =
;=  CALCULA LA FILA, COLUMNA Y LADO DE UN CASETE SOLICITADO. =
;=  EL DATO CALCULADO PONE EN EL BUFFER DE LA GRABADORA LIBRE =
;=                                                           =
;=  DIS                                                      =
;=  MOSTREA EN TRES DISPLAYS EL CODIGO DEL CASETE QUE SE ESTA =
;=  SELECCIONANDO Y EN OTROS TRES DISPLAYS EL CODIGO DEL CASETE =
;=  QUE SE ESTA ESCUCHANDO.                                  =
;=                                                           =
;=============================================================
; Inicializa las variables para poder utilizarlas en esta subrutina.
MUESTRE:    PUSH    ACC             ; Guarda el A, B, PSW.
            PUSH    B
            PUSH    PSW
            SETB    PSW.4           ; Selecciona el banco 2.
            CLR     PSW.3

; RES  actualiza el buffer de  respuesta de los elementos,  permitiendo
; conocer si un elemento ya ha terminado la tarea encomendada.
; La respuesta de un elemento que ha terminado la tarea pedida la obte-
; nemos en las siguientes localidades:
```

```
; Loc.     Elem.    Resp.
; 50H      MX       01
; 51H      MY       01
; 52H      MZ       01
; 53H      MC0      01
; 54H      MC1      01
; 55H      G0       01
; 56H      G1       01

RES:        MOV     P2,#LATCH       ; Activa el latch de respuestas
            MOV     A,P1            ; Lee la respuesta.
            MOV     57H,A           ; Inicializa la subrutina OTRO
            MOV     B,57H
            MOV     R0,#50H
OTRO:       ANL     A,#01H          ; Guarda las respuestas  en las
            MOV     @R0,A           ; localidades 50H hasta 56H.
            INC     R0              ; Las respuestas son 1L  cuando
            MOV     A,B             ; cuando ya ha terminado la ta-
            RR      A               ; rea  encomendada y  0L cuando
            MOV     B,A             ; aún no ha terminado.
            CJNE    R0,#57H,OTRO

; Esta segunda parte revisa si hay datos seleccionados,  si los hay re-
; visa si el buffer de G0  esta libre si no lo esta revisa si el buffer
; de G1 esta libre,  si ninguno de los dos  esta libre  salta toda esta
; segunda parte.  En cambio si uno de los dos buffers estuvo libre (te-
; niendo  prioridad el de G0)  calcula el casete,  la fila, la columna,
; el lado del casete y almacena el dato en el buffer respectivo.
```

```
; El buffer de G0  se encuentra en las localidades  58H, 59H, 5AH, 5BH.
; El buffer de G1  se encuentra en las localidades  5CH, 5DH, 5EH, 5FH.

PXY:        MOV     A,7EH           ; Si no hay datos no se ejecuta
            JZ      DIS             ; esta subrutina.
            MOV     R0,#58H         ; Inicializa para el buffer G0.
            MOV     R1,#5BH
            CJNE    @R0,#00H,LIB_G1 ; Esta libre G0?
            MOV     70H,A
            AJMP    FIL_COL         ; Si, calcula fila-columna.
LIB_G1:     MOV     R0,#5CH         ; Inicializa para el buffer G1
            MOV     R1,#5FH
            CJNE    @R0,#00H,DIS
            MOV     71H,A
FIL_COL:    INC     A               ; Calcula el lado a que corres-
            CLR     C               ; ponde. (0 = A y 1 = B).
            RRC     A
            MOV     @R0,A           ; Almacena el número de casete.
            CLR     A
            MOV     ACC.0,C
            MOV     @R1,A           ; Almacena el lado.
            DEC     R1
            MOV     @R1,#0FEH
            MOV     A,@R0
            CLR     C
RESTE:      MOV     B,A             ; Calcula fila y columna.
            SUBB    A,#28H
            INC     @R1
```

```
                INC     @R1             ; Guarda la posición Y.
                JC      GUARDE
                JZ      GUARDE
                AJMP    RESTE
GUARDE:         INC     R0
                MOV     @R0,B           ; Guarda la posición X.
                MOV     7EH,7DH         ; Retira un dato.
                MOV     7DH,7CH
                MOV     7CH,#00H
                INC     18H

; Esta  tercera parte visualiza dos  grupos de tres displays  cada uno.
; En el primer grupo el código del casete que se está seleccionando  y
; en el segundo grupo el código del casete que se está escuchando.
; No visualiza los ceros a la izquierda.
; La frecuencia de barrido para los displays es de 1 KHz.

DIS:            MOV     R5,#80          ; Esta  subrutina  se  repetirá
                MOV     DPTR,#TABLA     ; 80 veces.
DISP:           MOV     R0,#72H         ; Dirección del primer dígito.
                MOV     R1,#0FCH        ; Habilitación del primer disp.
                MOV     R3,#02H         ; Son  dos grupos de 3 displays
TRES:           MOV     A,@R0           ; Revisa si el dígito más  sig-
                MOV     R2,#03          ; nificativo no es cero.
                JZ      DOS
                AJMP    SHOW            ; Hay tercer dígito => muestra
DOS:            MOV     P0,#0FFH
                LCALL   DELAY           ; Espera 1mS.
```

```
        INC     R0              ; Da la dirección del próximo
        DEC     R1              ; dígito.
        MOV     A,@R0
        MOV     R2,#02H
        JZ      UN              ; Hay un solo dígito, muestra
        AJMP    SHOW            ; aún así sea cero.
UN:     MOV     P0,#0FFH
        LCALL   DELAY           ; Espera 1mS
        INC     R0
        DEC     R1
        MOV     A,@R0
        MOV     R2,#01H
SHOW:   MOVC    A,@A+DPTR       ; INDIQUE, indica el número de
        CPL     A               ; dígitos especificados por R2.
        MOV     P0,A            ; P0, contiene los segmentos
        MOV     P2,R1           ; del dígito.
        LCALL   DELAY           ; Permanece el dato 1 mS.
        INC     R0
        DEC     R1
        MOV     A,@R0
        DJNZ    R2,SHOW
        DJNZ    R3,TRES         ; Repite para el segundo grupo.
        DJNZ    R5,DISP
X:      POP     PSW             ; Restaura PSW, B, A.
        POP     B
        POP     ACC
        RET
```

```
DELAY:      MOV     16H,#0AH        ; Esta subrutina retarda 1 mS
WAIT:       MOV     17H,#20H
            DJNZ    17H,$
            DJNZ    16H,WAIT
            RET


TABLA:      DB      3FH     ; COD. 0
            DB      06H     ; COD. 1
            DB      5BH     ; COD. 2
            DB      4FH     ; COD. 3
            DB      66H     ; COD. 4
            DB      6DH     ; COD. 5
            DB      7DH     ; COD. 6
            DB      07H     ; COD. 7
            DB      7FH     ; COD. 8
            DB      6FH     ; COD. 9
```

```
;  ================================================================
;  =                                                              =
;  =                        SUB   TECLADO                         =
;  =      (ESTA SUBRUTINA ES CONTROLADA POR LA INTERRUPCION CERO)  =
;  =                                                              =
;  =  LEE LA TECLA PRESIONADA.                                    =
;  =                                                              =
;  =  NO ACEPTA SELECCIONES QUE NO EXISTEN.                       =
;  =                                                              =
;  =  EL PRIMER DATO INGRESADO ES EL NUMERO TOTAL DE SELECCIONES. =
;  =                                                              =
;  =  PERMITE BORRAR 1, 2 Ø 3 DIGITOS EN LA ULTIMA SELECCION  "B". =
;  =                                                              =
;  =  PARA QUE EL DATO SEA ACEPTADO SE DEBE PRESIONAR LA TECLA "A". =
;  =                                                              =
;  ================================================================

TECLADO:    PUSH    ACC             ; Guarda el A, B, PSW.
            PUSH    B
            PUSH    PSW
            SETB    PSW.4           ; Selecciona el banco 3.
            SETB    PSW.3
            CJNE    R0,#7BH,CONT    ; Comprueba  si hay  espacio en
            AJMP    OUT             ; el buffer. Si no hay sale.
CONT:       CJNE    R0,#7FH,DATO    ; Pone un  límite para  el dato
            MOV     R3,#240         ; que vaya a ingresar.
            AJMP    TECLA
DATO:       MOV     R3,7FH
```

```
TECLA:      MOV     A,P2            ; Lee la tecla presionada y al-
            SWAP    A               ; macena en R2.
            ANL     A,#0FH
            MOV     R2,A
            CLR     C               ; Determina que tecla es.
            SUBB    A,#0AH
            JC      ES_DIG          ; Hay carry   =>  es un dígito.
            JZ      ES_A            ; Es cero  =>  es "A".
            CJNE    R2,#0EH,OUT     ; Es otra tecla  =>  sale.
            AJMP    ES_B            ; Va a ES_B.
ES_DIG:     MOV     A,R1            ; Es primer dígito?
            JZ      PRI_DIG         ; Si => salta a PRI_DIG.
            CJNE    R1,#03,STR_DIG  ; Es el cuarto dígito?
            AJMP    OUT             ; Si => OUT. No => STR_DIG.
PRI_DIG:    MOV     A,R2            ; Es la primera tecla "0".
            JZ      OUT             ; Si => sale.
STR_DIG:    MOV     78H,79H         ; No => guarda.
            MOV     79H,7AH
            MOV     7AH,R2
            INC     R1              ; Increm. contador de dígitos.
            LCALL   DEC_BIN         ; Convierte a binario  el dato.
            CLR     C               ; Verifica si el dato existe.
            MOV     A,R3            ; C=0  =>  existe.
            SUBB    A,7EH           ; C=1  =>  no existe.
            JNC     OUT
ES_B:       CJNE    R1,#00,BORRE    ; Hay algún dígito para borrar?
            AJMP    OUT             ; No => sale.
BORRE:      MOV     7AH,79H         ; Si => borra último dígito.
```

```
                MOV     79H,78H
                MOV     78H,#00
                DEC     Ri              ; Decrementa contador de digit.
                LCALL   DEC_BIN         ; Convierte el dato a binario.
                AJMP    OUT
ES_A:           MOV     A,7BH           ; Hay un dato?
                CJNE    A,#00H,STR_DAT  ; Si => lo guarda.
                AJMP    OUT             ; No => sale.
STR_DAT:        MOV     @R0,7BH
                DEC     R0
                MOV     78H,#00         ; Encera  todas las localidades
                MOV     79H,#00         ; que  sirvieron para almacenar
                MOV     7AH,#00         ; los dígitos y el número bina-
                MOV     7BH,#00         ; rio equivalente de ellos.
                MOV     Ri,#00          ; Borra contador de dígitos.
OUT:            MOV     74H,7AH         ; Mueve los  dígitos  hacia las
                MOV     73H,79H         ; localidades donde  serán mos-
                MOV     72H,78H         ; trados.
                POP     PSW             ; Restaura el PSW, B, A.
                POP     B
                POP     ACC
                RET

; Esta subrutina convierte 1, 2 o 3 dígitos ingresados a su equivalente
; en binario, si el número es mayor a 255 entonces no se puede  almace-
; nar en 8 bits y devuelve #0FFH en la localidad 7BH.
; Esta subrutina es utilizada únicamente por la subrutina TECLADO.
DEC_BIN:        MOV     A,#100          ; Multiplica centenas por 100.
```

```
        MOV     B,78H
        MUL     AB
        MOV     7EH,A           ; Guarda el resultado
        MOV     A,B
        JNZ     OVER            ; Si es mayor a 255, termina.
        MOV     A,#10           ; Multiplica las decenas por 10
        MOV     B,79H
        MUL     AB
        ADD     A,7EH           ; Suma las unidades.
        ADD     A,7AH
        JC      OVER            ; Si es mayor a 255, termina.
        MOV     7EH,A
        RET
OVER:   MOV     7EH,#0FFH       ; Coloca #0FFH en el  resultado
        RET                     ; cuando es mayor a 255.
        END
```

CAPITULO IV

INSTRUCCIONES PARA EL USO DEL SIMULADOR DEL STM

INTRODUCCION

En este capítulo se encuentra la información necesaria para el uso del simulador por hardware del STM. Se explica el modo de ingresar los datos al simulador, la función de los displays y los leds, la forma en que trabajan los sensores y en detalle la secuencia de operaciones que irá realizando junto con el usuario.

El capítulo es relativamente simple sin embargo es preciso entenderlo perfectamente dado que para su correcto funcionamiento se requiere una interacción con el usuario, caso contrario va a resultar un tanto esotérico y no se va entender que función se encuentra realizando ni que sensor debería ser activado en el momento adecuado.

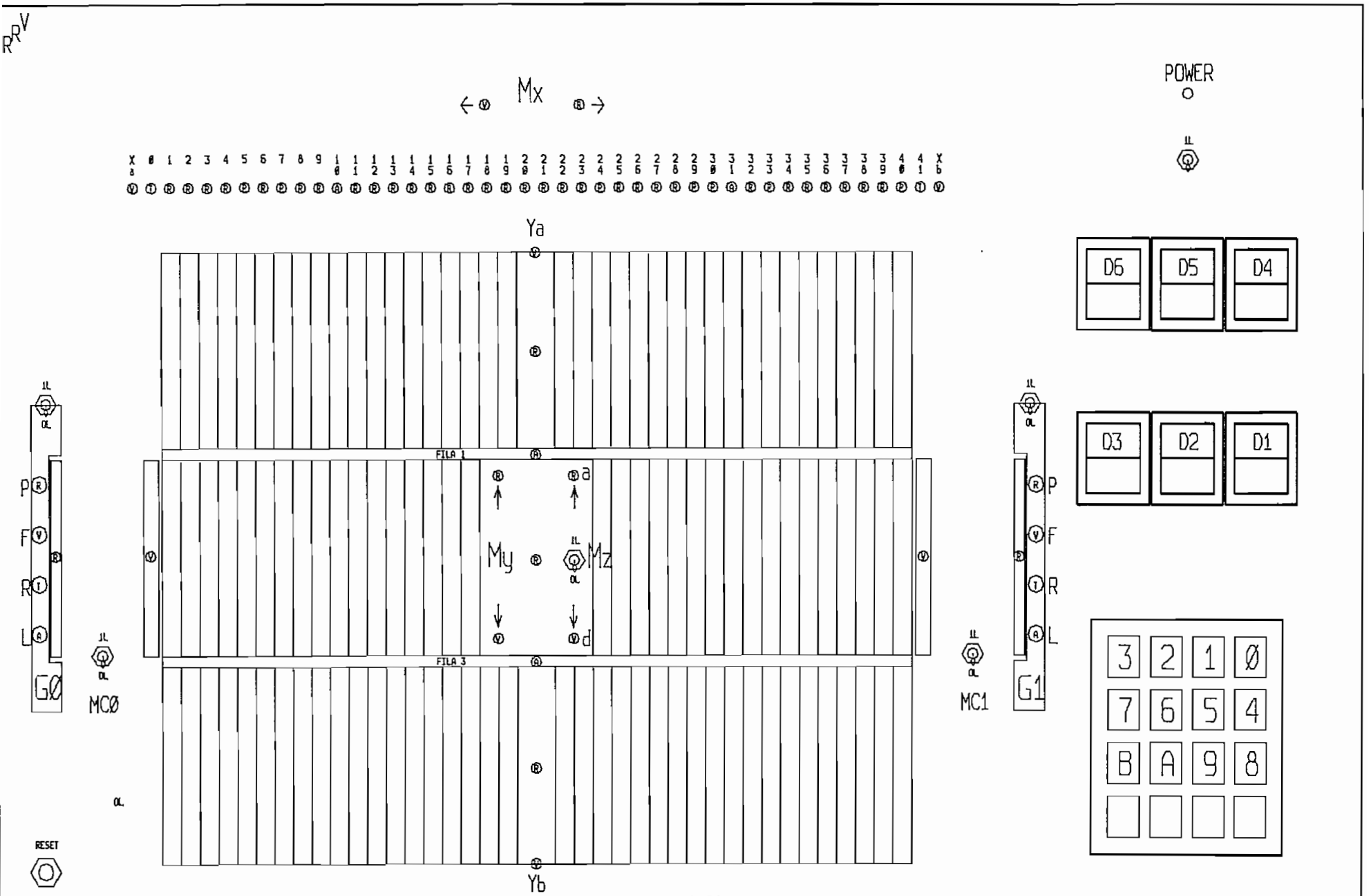
4.1 DESCRIPCION DEL PANEL DEL SIMULADOR. (Fig. 4.1)

Este panel está formado por los siguientes elementos:

- Interruptor principal y led.
- Pulsante (RESET).
- Displays de 7 segmentos.
- Teclado de 16 teclas (utilizadas solo 12).

- Leds para el sentido de giro de MX.
- Sensores para MX.
- Imán para los sensores de MX.
- Leds para los sensores de MX.
- Leds para el sentido de giro de MY.
- Sensores para MY.
- Imán para los sensores de MY.
- Leds para los sensores de MY.
- Leds para MZ.
- Interruptor para MZ.

- Leds para MC0.
- Interruptor para MC0.
- Leds para MC1.
- Interruptor para MC1.
- Leds para G0.
- Interruptor para G0.
- Leds para G1.
- Interruptor para G1.



EPN/94

FIG. 4.01

4.1.1 INTERRUPTOR PRINCIPAL Y LED. Este interruptor se encuentra en la parte superior derecha del STM, controla la corriente que circula en el primario del transformador, el mismo que se utiliza para obtener la fuente DC que dará la energía suficiente para todos los elementos del STM.

Se pudo haber controlado la corriente en el secundario del transformador pero esta forma de controlar presenta una desventaja que es la circulación de una pequeña corriente en el primario aun cuando éste interruptor no se encuentra conectado.

Una vez que el interruptor se encuentra cerrado el transformador genera un voltaje en el secundario, este voltaje luego de ser rectificado es regulado a 5VDC, el cual será utilizado para polarizar los integrados y para encender los leds. El led de la fuente que se encuentra justo al frente del interruptor principal, se encuentra conectado a través de una resistencia a la fuente de 5VDC. Por lo tanto la única condición para que este led se encienda es cerrar el interruptor principal.

4.1.2 PULSANTE (RESET). Este pulsante se encuentra en la esquina inferior izquierda, cuando lo pulsamos el microcontrolador saltará la ejecución del programa hasta la dirección de inicio o dirección 0000H. RESET no borra la RAM interna del microcontrolador, únicamente forza a que se ejecute el programa desde el inicio.

Para nuestro caso la función de reset y el interruptor principal realizan la misma función, puesto que al inicio del programa se encuentran 3 instrucciones las cuales borran completamente las 128 localidades de la RAM interna del microcontrolador.

4.1.3 DISPLAYS DE 7 SEGMENTOS. El simulador del STM tiene 6 displays de 7 segmentos, están divididos en 2 grupos de 3 displays cada uno. El primer grupo que se encuentra en la parte superior corresponde a los displays D6,D5,D4 y exhibirá el código del casete que se esta seleccionando. El segundo grupo corresponde a los displays D3,D2,D1 y muestra el código del casete que en ese momento se está escuchando. Ninguno de los 2 grupos mostrará los ceros de la izquierda, puesto que son innecesarios. La corriente con la cual trabajan cada uno de los segmentos es de 10 Ma, los displays son de cátodo común pero al tener la versatilidad para manipular los datos de P0 desde el programa se pudo haber utilizado displays de ánodo común sin que esto implique inconveniente alguno.

4.1.4 TECLADO. Es tipo matricial de 16 teclas a la salida de este teclado disponemos de 8 pines, 4 de los cuales representan las 4 filas y los otros 4 las columnas. Requiere de un decodificador de teclado el cual convierte los datos fila-columna en una información binaria de 4 bits. De este teclado para nuestro caso solamente son utilizadas 12 teclas, las últimas 4 teclas son ignoradas.

Las primeras 10 teclas numeradas desde el "0" hasta el "9" representan los dígitos, "A" es la tecla que permite que una cierta selección sea aceptada y sea puesta en la cola del buffer principal de selecciones y la tecla "B" permite borrar el dígito que se encuentra más a la derecha del primer grupo de displays, se puede borrar 1, 2 ó 3 dígitos dependiendo de cuantos se encuentran ingresados.

El buffer aceptará como máximo 5 selecciones, esta cantidad pudo ser ampliada hasta un número de 40 pero esto representaría una molestia al probar el simulador del STM pues tendríamos que esperar que pasen toda esta cantidad de selecciones hasta ver que sucede al final.

4.1.5 LEDS PARA EL SENTIDO DE GIRO DE MX. Existen 2 leds los cuales señalan el sentido de giro de MX, el led verde que se encuentra a la izquierda de la marca "MX" indica que el motor que controla la ubicación del brazo de selección en el sentido de X comenzó a girar hacia la izquierda, esto implica que la posición buscada se encuentra a la izquierda de la posición que en ese momento tenga el brazo de selección. Lo análogo sucede con el led rojo que se encuentra a la derecha de la marca "MX". El encendido de estos leds es controlado automáticamente por el microcontrolador.

4.1.6 SENSORES PARA MX. Existen 44 relés magnéticos colocados en el sentido de las X, cuando uno de ellos es activado el led correspondiente se encenderá.

El relé se encuentra en la parte interna del panel, colocado justo al frente del led y para ser activado se necesita un imán el cual cerrará los contactos del mismo haciendo que el led respectivo se encienda. Estos sensores tienen la siguiente nomenclatura; el primero es Xa, el último es Xb, desde el segundo hasta el penúltimo se encuentran numerados desde el 0 hasta 41. Las numeraciones desde 1 hasta 40 corresponden a las 40 columnas de casetes. La numeración 0 pertenece a la posición que ayuda a ubicar un cierto casete en G0, la posición 41 tiene la misma finalidad que la posición 0 pero para G1.

Cuando uno de los 44 relés es cerrado aparece una señal la misma que ingresa al circuito de control de MX y si esa posición pertenece a la que fue solicitada por el microcontrolador el led que indica la dirección en la cual se encontraba moviendo MX se apagará, entendiéndose que la búsqueda ha terminado.

4.1.7 IMAN PARA LOS SENSORES DE MX. El imán que se tiene que utilizar debe ser adecuado de modo que su campo magnético no sea muy disperso pues activaría más de 1 relé a la vez, por otro lado si su campo no es suficientemente intenso no podrá activar ningún relé puesto que la distancia que existe entre el imán y el relé es de 3mm, dicha distancia corresponde al espesor del acrílico que constituye la infraestructura del panel. El imán como es tendrá que ser desplazado sobre los relés por el usuario.

4.1.8 LEDS PARA LOS SENSORES DE MX. Estos leds indican cuando uno de los 44 sensores ubicados en el sentido X ha sido activado, en cualquier instante solo podrá estar encendido 1 led a la vez, es decir no puede estar activado más de un relé. La nomenclatura que poseen estos leds es la misma que la utilizada para sus respectivos sensores.

4.1.9 LEDS PARA EL SENTIDO DE GIRO DE MY. Existen 2 leds los cuales señalan el sentido de giro de MY, el led verde que se encuentra en la parte inferior de la marca "MY" e indica que el mecanismo MY empezó a moverse hacia abajo, esto implica que la posición deseada debe ser buscada en el rango inferior partiendo de la posición que tenga en ese momento.

Lo análogo sucede con el led rojo que se encuentra arriba de la marca "MY". El encendido de estos leds es controlado automáticamente por el microcontrolador.

4.1.10 SENSORES PARA MY. Existe un total de 7 relés magnéticos cada uno de estos trabajan junto con un led, estos sensores tienen la siguiente nomenclatura; el primero es Ya, el último es Yb, desde el segundo (empezando desde arriba) hasta el penúltimo se encuentran numerados desde el 0 hasta 4.

Las numeraciones 0, 2 y 4 corresponden a las 3 filas de casetes. Las numeraciones 1 y 3 corresponden a posiciones intermedias entre las filas 0,2 y 2,4 respectivamente.

Cuando un determinado relé ha sido activado el led correspondiente se encenderá. El relé se encuentra en la parte interna del simulador colocado justo al lado izquierdo del led y para ser activado precisa de un imán que permitirá cerrar los contactos haciendo que el led se encienda, esta señal ingresará al circuito de control de MY y si esa posición pertenece a la solicitada por el microcontrolador el led que indica la dirección en la cual se estuvo moviendo MY se apagará, entendiéndose que ha llegado a la posición buscada.

4.1.11 IMAN PARA LOS SENSORES DE MY. El imán que se debe utilizar tiene las mismas características que el utilizado para los relés de MX y fue descrito en 4.1.7. Este imán evidentemente también tendrá que ser movido por el usuario.

4.1.12 LEDS PARA LOS SENSORES DE MY. Indican cuando uno de los 7 sensores ha sido activado, en cualquier instante solo podrá estar encendido 1 led a la vez. Estos leds poseen la misma nomenclatura que su respectivo sensor.

4.1.13 LEDS PARA MZ. Existen 2 leds los cuales señalan el sentido el movimiento que tiene el mecanismo MZ. El led verde que se encuentra en la parte inferior de la marca "MZ" nos indica que MZ se está DESCENDIENDO y el led rojo que se encuentra arriba de la marca "MZ" indica que el mecanismo se encuentra ASCENDIENDO. Estos leds son encendidos automáticamente por el microcontrolador.

4.1.14 INTERRUPTOR PARA MZ. Este interruptor se encuentra en medio de los 2 leds que se acaba de mencionar. Tiene doble finalidad pues reemplaza a los 2 sensores que necesita MZ. Cuando el mecanismo MZ asciende, al final de su ascenso existe un microswitch (Za) el cual detiene al mecanismo. De modo similar cuando el mecanismo MZ desciende, al final de su descenso existe otro microswitch (Zb) que detiene al mecanismo. En tal virtud este interruptor cuando se encuentra en la parte superior estará simulando a Za y cuando se encuentra en la parte inferior a Zb.

4.1.15 LEDS PARA MCØ. Existen 2 leds los cuales señalan el sentido de movimiento del mecanismo MCØ. El led rojo se encuentra a la izquierda de la marca "MCØ" indica que el casete está siendo introducido en GØ. El led verde se encuentra a la derecha "MCØ" indica que el casete está siendo retirado de GØ por dicho mecanismo. Uno de estos 2 leds son encendidos automáticamente por el microcontrolador.

4.1.16 INTERRUPTOR PARA MCØ. Este interruptor se encuentra en la parte inferior central de los leds que se acaba de mencionar y es el encargado de reemplazar a los 2 sensores que necesita MCØ.

Cuando el MCØ ha terminado de introducir o retirar el casete en GØ, al final existe un microswitch el cual da una señal al microcontrolador para que detenga al motor que originó el movimiento del mecanismo MCØ.

Este interruptor debe enviar un pulso de 1L por lo menos 500 ms y tiene que regresar a 0L para que el programa pueda continuar con su ejecución. Este interruptor tiene que ser manipulado por el usuario.

4.1.17 LEDS PARA MC1. El led rojo se encuentra a la derecha de la marca "MC1" indica que el casete está siendo introducido en G1 por el mecanismo MC1.

El led verde se encuentra a la izquierda de la marca "MC0" indica que el casete está siendo retirado de G1 por dicho mecanismo. Uno de estos 2 leds son encendidos automáticamente por el microcontrolador.

4.1.18 INTERRUPTOR PARA MC1. Este interruptor se encuentra en la parte inferior central de los leds que se acaba de mencionar y reemplaza a los 2 sensores que necesita MC1. Funciona de modo similar que el INTERRUPTOR PARA MC0.

4.1.19 LEDS PARA G0. En la parte central-izquierda del panel existen 4 leds los cuales representan la función que se encuentra realizando dicha grabadora.

Led rojo, indica que G0 se encuentra en PLAY.

El led verde, indica que G0 se encuentra en FORWARD. La cinta del casete que se encuentra en G0 se encuentra desplazando desde el lado izquierdo hasta el derecho.

El led naranja, indica que la se encuentra en REWIND. La cinta del casete que se encuentra en G0 se encuentra desplazando desde el lado derecho de hasta el izquierdo.

Led amarillo, señala el lado que la grabadora está en PLAY, cuando esta apagado significa que el lado que esta siendo escuchado es el lado "A" de la grabadora y cuando esta encendido se esta escuchando el lado "B".

Nota: Solo para G0, el lado "A" de G0 coincide con el lado "B" del casete y el lado "B" de G0 con el lado "A" del casete. Para G1, el lado "A" de G1 coincide con el lado "A" del casete y el lado "B" de G1 con el lado "B" del casete. Todos los casetes SIEMPRE se encuentran con el lado "A" hacia G0.

4.1.20 INTERRUPTOR PARA G0. Se encuentra en la parte superior de los leds que se acaba de mencionar. Sirve para que cuando G0 haya terminado una determinada función (PLAY, FORWARD O REWIND), el usuario pueda enviar una señal (1L, mínimo 500ms) al microcontrolador el mismo que retirará la función que en ese momento posea G0. El programa no puede continuar hasta que este interruptor regrese a 0L.

4.1.21 LEDS PARA G1. En la parte central-derecha del panel existen 4 leds los cuales representan la función que se encuentra realizando dicha grabadora.

Led rojo, indica que G1 se encuentra en PLAY.

El led verde, indica que G1 se encuentra en FORWARD. La cinta del casete que se encuentra en G1 se encuentra desplazando desde el lado izquierdo hasta el derecho.

El led naranja, indica que la se encuentra en REWIND. La cinta del casete que se encuentra en G1 se encuentra desplazando desde el lado derecho de hasta el izquierdo.

Led amarillo, señala el lado que la grabadora está en PLAY, cuando esta apagado significa que el lado que esta siendo escuchado es el lado "Ø" de la grabadora y cuando esta encendido se esta escuchando el lado "1".

4.1.22 INTERRUPTOR PARA G1. Se encuentra en la parte superior de los leds que se acaba de mencionar. Sirve para que cuando G1 ha terminado una de las tareas encomendadas (PLAY, FORWARD O REWIND), el usuario debe enviar una señal poniéndolo en 1L, la cual será leída por el microcontrolador el mismo retirará la función que en ese momento posea G1. El programa no continua mientras este interruptor no regrese a su posición original que es ØL.

Los interruptores de MCØ, MC1, GØ y G1 envían las respectivas señales de modo similar, es decir un pulso de 1L durante un tiempo mínimo de 5ØØ mS.

4.2 CODIGOS.

Antes de poder ingresar los datos tenemos que conocer el código que posee cada una de las selecciones. Para poder asignar dichos códigos partamos de las siguientes premisas:

El número máximo de casetes que puede contener el STM es de 12Ø.

Todos lo casetes SIEMPRE se encuentran con el lado "A" a la izquierda (hacia GØ) y con el lado "B" a la derecha (hacia G1).

Al tener 12Ø casetes disponemos 12Ø lados "A" y 12Ø lados "B" por lo tanto tenemos 24Ø lados.

Todos los casetes se encuentran parados con la parte más ancha hacia arriba y distribuidos en 3 filas de 4Ø casetes cada fila.

Las casillas que contienen a los casetes se encuentran numeradas desde 1 hasta 12Ø, siendo la 1 la que se encuentra en la esquina superior-izquierda y la 12Ø la que está en la esquina inferior-derecha.

Sea el número de casetes que sea (obviamente máximo 12Ø) tienen que estar ubicados en casillas contiguas comenzando desde la casilla #1.

Así por ejemplo si existen 87 casetes tienen que ocupar las primeras 87 casillas, es decir las primeras 2 filas completas y 7 casetes en la tercera fila. No se admite casillas vacías entre casetes. Un casete que ocupa una determinada casilla en ningún momento podrá ocupar una diferente a esa.

A cada uno de los 240 lados le corresponde un código único compuesto completamente de dígitos. Los códigos van desde el número 1 hasta el 240.

Ej. 4.2.a Al casete que se encuentra en la casilla #1, le corresponde el código #1 para el lado "A" y el código #2 para el lado "B".

Ej. 4.2.b Al casete que se encuentra en la casilla #63, le corresponde el código #125 para el lado "A" y el código #126 para el lado "B".

Ej. 4.2.c Al casete que se encuentra en la casilla #117, por lo tanto le corresponde el código #233 para el lado "A" y el código #234 para el lado "B".

De estos ejemplos podemos inferir que estos códigos podemos escribirlo en forma de ecuación.

Código del lado "A" = (2 * # Casilla) - 1

Código del lado "B" = 2 * # Casilla

4.3 INGRESO DEL PRIMER DATO.

Una vez que el STM es energizado puede aceptar los datos que ingresan desde el teclado, estos datos pueden entrar en cualquier momento puesto que la ejecución de la subrutina que permite el ingreso de datos es controlada por la interrupción cero, siendo esta la única interrupción utilizada en el circuito estamos absolutamente seguros que en ningún instante la tecla que presionemos será ignorada.

El primer dato que ingresamos corresponde al código más alto que en adelante podremos introducir, este primer dato no corresponde a ninguna selección. Por ejemplo si tenemos 65 casetes en el STM el primer dato que debemos ingresar es el 130, en adelante no solo que no se podrá ingresar un dato que supere a 130. Como es evidente el primer dato tampoco será aceptado si se intenta ingresar un número superior a 240.

Ej. 4.3.a Analicemos paso a paso que sucede una vez que hemos energizado el simulador del STM e intentamos ingresar un dato superior a 240.

D6, D5, D4	_ _ 0
Tecla pres.	"2"
D6, D5, D4	_ _ 2
Tecla pres.	"4"
D6, D5, D4	_ 2 4

(error)	Tecla pres.	"7"
	D6, D5, D4	_ 2 4
(error)	Tecla pres.	"1"
	D6, D5, D4	_ 2 4
(error)	Tecla pres.	"3"
	D6, D5, D4	_ 2 4
	Tecla pres.	"0"
	D6, D5, D4	2 4 0
	Tecla pres.	"A"
	D6, D5, D4	_ _ 0

Ej. 4.3.b Empezamos a partir del momento en que fue energizado el STM. Tenemos 3 casetes disponibles. Por lo tanto el primer dato es el #6.

	D6, D5, D4	_ _ 0
(error, es 6)	Tecla pres.	"8"
	D6, D5, D4	_ _ 8
(borro el 8)	Tecla pres.	"B"
	D6, D5, D4	_ _ 0
	Tecla pres.	"6"
	D6, D5, D4	_ _ 6
	Tecla pres.	"A"
	D6, D5, D4	_ _ 0

Ej. 4.3.c Empezamos a partir del momento en que fue energizado el STM. Tenemos 47 casetes disponibles, por lo tanto el primer dato es el #94.

	D6, D5, D4	_ _ 0
(error, es 9)	Tecla pres.	"8"
	D6, D5, D4	_ _ 8
(error, es 4)	Tecla pres.	"7"
	D6, D5, D4	_ 8 7
(borro el 7)	Tecla pres.	"B"
	D6, D5, D4	_ _ 8
(borro el 8)	Tecla pres.	"B"
	D6, D5, D4	_ _ 0
	Tecla pres.	"9"
	D6, D5, D4	_ _ 9
	Tecla pres.	"4"
	D6, D5, D4	_ 9 4
	Tecla pres.	"A"
	D6, D5, D4	_ _ 0

La tecla "B" permite borrar el dígito de más a la derecha del dato que estemos ingresando, se puede borrar de forma continua 1, 2 o 3 dígitos.

La tecla "A", permite almacenar el dato en el buffer.

Una vez introducido el primer dato solo puede ser borrado cuando se apague el STM o se presione RESET.

Además, mientras permanezca encendido o no se pulse RESET, el STM no se podrá aceptar una selección superior al primer dato.

4.4 INGRESO DE SELECCIONES.

Una selección es un código elegido subjetivamente por el usuario. Se puede almacenar hasta 5 selecciones.

Para explicar esta sección lo más explícito es realizar un ejemplo completo en el cual podamos ver exactamente lo que sucede.

Ej. 4.4.a El STM posee 98 casetes (por lo tanto quedan las últimas 22 casillas vacías) y queremos escuchar las siguientes selecciones:

#57, #185, #158, #45, #192, #79.

Energizamos el STM e introducimos el primer dato que tiene que ser el #196 (este dato corresponde a 2 veces el número de casetes que dispongamos).

D6, D5, D4	_ _ 0
Tecla pres.	"1"
D6, D5, D4	_ _ 1
Tecla pres.	"9"
D6, D5, D4	_ 1 9
Tecla pres.	"6"
D6, D5, D4	1 9 6
Tecla pres.	"A"
D6, D5, D4	_ _ 0

Ingresamos la primera selección, #57.

D6, D5, D4	_ _ 0
Tecla pres.	"5"
D6, D5, D4	_ _ 5
Tecla pres.	"7"
D6, D5, D4	_ 5 7
Tecla pres.	"A"
D6, D5, D4	_ _ 0

Ingresamos la segunda selección, #185.

D6, D5, D4	_ _ 0
Tecla pres.	"1"
D6, D5, D4	_ _ 1
Tecla pres.	"8"
D6, D5, D4	_ 1 8
Tecla pres.	"5"
D6, D5, D4	1 8 5
Tecla pres.	"A"
D6, D5, D4	_ _ 0

Ingresamos la tercera selección, #158.

D6, D5, D4	_ _ 0
Tecla pres.	"1"
D6, D5, D4	_ _ 1
Tecla pres.	"5"

	D6, D5, D4	_ 1 5
	Tecla pres.	"8"
	D6, D5, D4	1 5 8
	Tecla pres.	"A"
	D6, D5, D4	_ _ 0

Ingresamos la cuarta selección, #45.

	D6, D5, D4	_ _ 0
(error, es 4)	Tecla pres.	"8"
	D6, D5, D4	_ _ 8
(borro el 8)	Tecla pres.	"B"
	D6, D5, D4	_ _ 0
	Tecla pres.	"4"
	D6, D5, D4	_ _ 4
	Tecla pres.	"6"
	D6, D5, D4	_ 4 5
	Tecla pres.	"A"
	D6, D5, D4	_ _ 0

Ingresamos la quinta selección, #192.

	D6, D5, D4	_ _ 0
	Tecla pres.	"1"
	D6, D5, D4	_ _ 1
	Tecla pres.	"9"
	D6, D5, D4	_ 1 9
	Tecla pres.	"2"

D6, D5, D4	1 9 2
Tecla pres.	"A"
D6, D5, D4	_ _ 0

Ingresamos la sexta selección, #79.

D6, D5, D4	_ _ 0
Tecla pres.	"7"
D6, D5, D4	_ _ 0
Tecla pres.	"9"
D6, D5, D4	_ _ 0
Tecla pres.	"A"
D6, D5, D4	_ _ 0

En la sexta selección ninguna tecla tiene aceptación pues el buffer de selecciones se encuentra lleno y no puede aceptar absolutamente ninguna otra selección mientras no se haya terminado de escuchar la primera, momento en el cual el microcontrolador borrará la primera selección del buffer dejando un espacio para una nueva selección. De estas 5 selecciones que se han ingresado la primera ha pasado a ocupar el buffer de G0 y la segunda el buffer de G1.

El buffer de G0 al cual se hace referencia corresponde a 4 localidades de memoria (58H, 59H, 5AH, 5BH), estas 4 localidades contienen el código del casete que será colocado en G0, este código está desensamblado y almacenado en estas localidades de la siguiente manera:

Primera localidad del buffer de G0. # del casete, el mismo que debe ser entre 1 y 120.

Segunda localidad del buffer de G0. # posición X, es el número de la columna donde se encuentra dicho casete, este valor está comprendido entre 1 y 40.

Tercera localidad del buffer de G0. # posición Y, es el número de la fila donde se encuentra dicho casete, es 0 si está en la primera fila, 2 si está en la segunda y 4 si se encuentra en la tercera.

Cuarta localidad del buffer de G0. Lado del casete al cual corresponde dicha selección, 0 si esa selección es impar es decir corresponde al lado "A" de ese casete y 1 si es par entonces es el lado "B".

Del mismo modo al buffer de G1 le corresponde las localidades 5CH, 5DH, 5EH, 5FH, estas localidades contienen el código del casete que será colocado en G1.

En síntesis.

Buffer de G0	Buffer de G1	Dato almacenado
58H	5CH	# casete.
59H	5DH	# columna.
5AH	5EH	# fila.
5BH	5FH	lado.

4.5 PRUEBA DEL SIMULADOR POR HARDWARE DEL STM.

4.5.1 NOMENCLATURA UTILIZADA. Partimos desde el momento en que el simulador fue energizado. Asumimos que lo primero que hizo el usuario fue introducir los datos correspondientes al ejemplo 4.4.a y no realiza ninguna otra selección hasta cuando se acabe de escuchar la última realizada, esto no implica que no se puede hacer lo que sucede es que complicaría la explicación que vamos a realizar. Los 4 interruptores que simulan los sensores de MCØ, MC1, GØ y G1 tienen que encontrarse en ØL. El interruptor MZ debe encontrarse enviando un ØL.

La siguiente nomenclatura se utilizará hasta el final de este capitulo.

Led verde de MX encendido (MICRO) => MX-IZQUIERDA.
Led rojo de MX encendido (MICRO) => MX-DERECHA.
Led verde de MY encendido (MICRO) => MY-BAJA.
Led rojo de MY encendido (MICRO) => MY-SUBE.

Led rojo de MZ encendido (MICRO) => MZ-ASCIENDE.
Led verde de MZ encendido (MICRO) => MZ-DESCIENDE.
Interruptor de MZ conectado (USUARIO) => INT-MZ = 1L
Interruptor de MZ desconec. (USUARIO) => INT-MZ = ØL

Led rojo de MCØ encendido (MICRO) =>INTRO. CASETE EN GØ
Led verde de MCØ encendido (MICRO) =>RETIRA CASETE DE GØ

Interrup. de MC0 conectado (USUARIO) => INT-MC0 = 1L

Interruptor de MC0 desconec (USUARIO) => INT-MC0 = 0L

Led rojo de MC1 encendido (MICRO) =>INTRO. CASETE EN G1

Led verde de MC1 encendido (MICRO) =>RETIRA CASETE DE G1

Interrup. de MC1 conectado (USUARIO) => INT-MC1 = 1L

Interruptor de MC1 desconec (USUARIO) => INT-MC1 = 0L

Interruptor de G0 conectado (USUARIO) => INT-G0 = 1L

Interruptor de G1 desconec (USUARIO) => INT-G1 = 0L

IMAN-X (x) (USUARIO) => El IMAN-X debe pasar a la posición x, donde x puede variar desde 0 hasta 41 ó x=Xa ó x=Xb. (Ver lit. 4.1.7).

IMAN-Y (y) (USUARIO) => El IMAN-Y debe pasar a la posición y, donde y puede variar desde 0 hasta 4 ó y=Ya ó y=Yb. (Ver lit. 4.1.11).

Si existe dudas en cuanto a la función que desempeña cada uno de los elementos que se acaba de mencionar es probable que la causa se deba a que no se ha leído detenidamente el literal 4.1, en el cual se encuentra claramente explicado dichas funciones. Sería recomendable volverlo a leer, pero más detenidamente.

4.5.2 EJECUCION DEL PROGRAMA. Los datos que tenemos en los buffers son:

```
Buffer de G0.   57     (29 29    0     0)
Buffer de G1.   185    (93 13    4     0)
Buffer princ.   158
                45
                192
```

Se pone en posiciones iniciales.

```
                MZ-ASCIENDE
(USUARIO)       INT-MZ = 1L (arriba)
                MX-DERECHA
                MY-BAJA
(USUARIO)       IMAN-X(20)
(USUARIO)       IMAN-Y(2)
                RETIRA CASETE DE G0
(USUARIO)       INT-MC0 = 1L
(USUARIO)       INT-MC0 = 0L
                RETIRA CASETE DE G1
(USUARIO)       INT-MC1 = 1L
(USUARIO)       INT-MC1 = 0L
```

Ubica el casete correspondiente a la selección #57 en G0.

```
                MX-DERECHA
                MY-SUBE
(USUARIO)       IMAN-X(29)
(USUARIO)       IMAN-Y(0)
```

```
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                MX-IZQUIERDA
                MY-BAJA
(USUARIO)       IMAN-X(0)
(USUARIO)       IMAN-Y(2)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MY-SUBE
(USUARIO)       IMAN-Y(1)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                INTRODUCE CASETE EN G0
(USUARIO)       INT-MC0 = 1L
(USUARIO)       INT-MC0 = 0L

                G0 <- PLAY (1)
```

Ubica el casete correspondiente a la selección #185 en G1.

```
                MX-DERECHA
                MY-BAJA
(USUARIO)       IMAN-X(13)
(USUARIO)       IMAN-Y(4)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MZ-ASCIENDE
```

```
(USUARIO)       INT-MZ (arriba)
                MX-DERECHA
                MY-SUBE
(USUARIO)       IMAN-X(41)
(USUARIO)       IMAN-Y(2)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MY-SUBE
                IMAN-Y(1)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                INTRODUCE CASETE EN G1
(USUARIO)       INT-MC1 = 1L
(USUARIO)       INT-MC1 - ØL
```

Regresa el casete correspondiente al código #57 a su lugar.

```
(USUARIO)       INT-GØ = 1L
                G1 <- PLAY (Ø)
                GØ <- SIN FUNCION
(USUARIO)       INT-GØ = ØL
                GØ <- FORWARD
                INT-GØ = 1L
                GØ <- SIN FUNCION
                INT-GØ = OL
                RETIRA CASETE DE GØ
(USUARIO)       INT-MCØ = 1L
(USUARIO)       INT-MCØ = ØL
                MX-IZQUIERDA
```

```
                MY-BAJA
(USUARIO)       IMAN-X(0)
(USUARIO)       IMAN-Y(2)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                MX-DERECHA
                MY-SUBE
(USUARIO)       IMAN-X(29)
(USUARIO)       IMAN-Y(0)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MY-BAJA
(USUARIO)       IMAN-Y(1)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                MX-IZQUIERDA
                MY-BAJA
(USUARIO)       IMAN-X(20)
(USUARIO)       IMAN-Y(2)

Buffer de G0.   158   (79 39   2    1)
Buffer de G1.   185   (13  4   0    0)
Buffer princ.   45
                192
                0
```

Ubica el casete correspondiente a la selección #158 en GØ.

```
                MX-DERECHA
(USUARIO)       IMAN-X(39)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                MX-IZQUIERDA
(USUARIO)       IMAN-X(Ø)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MY-SUBE
(USUARIO)       IMAN-Y(1)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                INTRODUCE CASETE EN GØ
(USUARIO)       INT-MCØ = 1L
(USUARIO)       INT-MCØ = ØL
                GØ <- REWIND
(USUARIO)       INT-GØ = 1L
                GØ <- SIN FUNCION
(USUARIO)       INT-GØ = ØL
(USUARIO)       INT-G1 = 1L


                GØ <- PLAY (Ø)
```

Regresa el casete correspondiente al código #185 a su lugar.

```
                G1 <- SIN FUNCION
```

```
(USUARIO)       INT-G1 = ØL
                G1 <- REWIND
                INT-G1 = 1L
                G1 <- SIN FUNCION
                INT-G1 = ØL
                RETIRA CASETE DE G1
(USUARIO)       INT-MC1 = 1L
(USUARIO)       INT-MC1 = ØL
                MX-DERECHA
                MY-BAJA
(USUARIO)       IMAN-X(41)
(USUARIO)       IMAN-Y(2)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                MX-IZQUIERDA
                MY-BAJA
(USUARIO)       IMAN-X(13)
(USUARIO)       IMAN-Y(4)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MY-SUBE
(USUARIO)       IMAN-Y(3)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                MX-DERECHA
                MY-SUBE
```

```
(USUARIO)       IMAN-X(20)
(USUARIO)       IMAN-Y(2)

Buffer de G0.   158    (79 39    2     1)
Buffer de G1.   45     (23 23    0     0)
Buffer princ.   192
                0
                0

Ubica #45 en G1.
                MX-DERECHA
                MY-SUBE
(USUARIO)       IMAN-X(23)
(USUARIO)       IMAN-Y(0)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                MX-DERECHA
                MY-BAJA
(USUARIO)       IMAN-X(41)
(USUARIO)       IMAN-Y(2)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MY-SUBE
                IMAN-Y(1)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
```

```
                    INTRODUCE CASETE EN G1
(USUARIO)           INT-MC1 = 1L
(USUARIO)           INT-MC1 - 0L
(USUARIO)           INT-G0 = 1L

                    G1 <- PLAY (1)
```

Regresa el casete correspondiente al código #158 a su lugar.

```
                    G0 <- SIN FUNCION
(USUARIO)           INT-G0 = 0L
                    RETIRA CASETE DE G0
                    INT-MC0 = 1L
                    INT-MC0 = 0L
                    MX-IZQUIERDA
                    MY-BAJA
(USUARIO)           IMAN-X(0)
(USUARIO)           IMAN-Y(2)
                    MZ-DESCIENDE
(USUARIO)           INT-MZ (abajo)
                    MZ-ASCIENDE
(USUARIO)           INT-MZ (arriba)
                    MX-DERECHA
(USUARIO)           IMAN-X(39)
                    MZ-DESCIENDE
(USUARIO)           INT-MZ (abajo)
                    MY-SUBE
(USUARIO)           IMAN-Y(1)
                    MZ-ASCIENDE
```

```
(USUARIO)       INT-MZ (arriba)
                MX-IZQUIERDA
                MY-BAJA
(USUARIO)       IMAN-X(20)
(USUARIO)       IMAN-Y(2)


Buffer de G0.   192    (96 16    4     1)
Buffer de G1.   45     (23 23    0     0)
Buffer princ.   0
                0
                0

Ubica #192 en G0.
                MX-IZQUIERDA
                MY-BAJA
(USUARIO)       IMAN-X(16)
(USUARIO)       IMAN-Y(4)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                MX-IZQUIERDA
                MY-SUBE
(USUARIO)       IMAN-X(0)
(USUARIO)       IMAN-Y(2)
                MZ-DESCIENDE
                INT-MZ (abajo)
```

```
                    MY-SUBE
                    IMAN-Y(1)
                    MZ-ASCIENDE
(USUARIO)           INT-MZ (arriba)
                    INTRODUCE CASETE EN G0
(USUARIO)           INT-MC0 = 1L
(USUARIO)           INT-MC0 = 0L
                    G0 <- REWIND
(USUARIO)           INT-G0 = 1L
                    G0 <- SIN FUNCION
(USUARIO)           INT-G0 = 0L

(USUARIO)           INT-G1 = 1L
                    G0 <- PLAY(0)
                    G1 <- SIN FUNCION
(USUARIO)           INT-G1 = 0L
```

Regresa el casete correspondiente al código #158 a su lugar.

```
                    G1 <- REWIND
                    INT-G1 = 1L
                    G1 <- SIN FUNCION
                    INT-G1 = 0L
                    RETIRA CASETE DE G1
(USUARIO)           INT-MC1 = 1L
(USUARIO)           INT-MC1 = 0L
                    MX-DERECHA
                    MY-BAJA
(USUARIO)           IMAN-X(41)
```

```
(USUARIO)       IMAN-Y(2)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                MX-IZQUIERDA
                MY-SUBE
(USUARIO)       IMAN-X(23)
(USUARID)       IMAN-Y(0)
                MZ-DESCIENDE
(USUARIO)       INT-MZ (abajo)
                MY-BAJA
(USUARIO)       IMAN-Y(1)
                MZ-ASCIENDE
(USUARIO)       INT-MZ (arriba)
                MX-IZQUIERDA
                MY-BAJA
(USUARIO)       IMAN-X(20)
(USUARIO)       IMAN-Y(2)

Buffer de G0.   192    (96 16    4     1)
Buffer de G1.   0      (0  0     0     0)
Buffer princ.   0
                0
                0

Regresa el casete correspondiente al código #192 a su lugar.
(USUARIO)       INT-G0 = 1L
```

```
                    G0 <- SIN FUNCION
(USUARIO)           INT-G0 = 0L
                    RETIRE CASETE DE G0
                    INT-MC0 = 1L
                    INT-MC0 = 0L
                    MX-IZQUIERDA
(USUARIO)           IMAN-X(0)
                    MZ-DESCIENDE
(USUARIO)           INT-MZ (abajo)
                    MZ-ASCIENDE
(USUARIO)           INT-MZ (arriba)
                    MX-DERECHA
                    MY-BAJA
(USUARIO)           IMAN-X(16)
(USUARIO)           IMAN-Y(4)
                    MZ-DESCIENDE
(USUARIO)           INT-MZ (abajo)
                    MY-SUBE
(USUARIO)           IMAN-Y(3)
                    MZ-ASCIENDE
(USUARIO)           INT-MZ (arriba)
                    MX-DERECHA
                    MY-SUBE
(USUARIO)           IMAN-X(20)
(USUARIO)           IMAN-Y(2)
```

CONCLUSIONES

- En primer lugar cabe anotar que un gran porcentaje del tiempo invertido en este trabajo se encuentra en los 3 intentos realizados por construir el sistema mecánico del STM.

- La falta de recursos como la maquinaria adecuada para construir la parte mecánica, considero que fue uno de los principales limitantes para que el STM no haya sido construido en su totalidad.

- Al intentar construir la parte mecánica se llegó a conocer distintos aspectos relacionados con esta rama, considero que este es un punto bastante positivo.

- El hardware implementado hubiese sido posible realizarlo con un menor número de integrados, de este modo se sacaría un mayor provecho al microcontrolador. Sin embargo esto requeriría un mayor número de pruebas en cuanto al programa que en él debe ser grabado. Muchos de los circuitos implementados es posible probarlos aún sin la necesidad del microcontrolador.

- Una de las partes complejas del presente trabajo fue establecer la idea clara de la forma física que debe tener el aparato, algo similar sucedío en cuanto al hardware.

- La parte en que más tiempo se empleó dentro de lo que es software fue la realización del diagrama general de flujo. El microcontrolador utilizado (8751H) posee grandes ventajas frente al Z8Ø, como por ejemplo un set de instrucciones mucho más simple, su EPROM interna, etc.

- Las aplicaciones que puede tener el 8751H son bastante grandes, dentro de lo que es la electrónica se podría decir que "dichas aplicaciones se encuentran limitadas solo por la imaginación".

- La presente tesis ha permitido conocer distintos paquetes de programas, considero que este es otro punto bastante importante.

- Creo que en la EPN debería existir la posibilidad de que una tesis pueda ser realizada en grupo de estudiantes de distinta especialidad, esto conduciría a la construcción de aparatos más complejos e interesantes.

BIBLIOGRAFIA

- TTL 1
- TTL 2
- ECG
- NTE
- VOLTAGE REGULATOR.
  (HANDBOOK) NATIONAL SEMICONDUCTOR.
- MANUAL DEL 8751.