FULLY SOFTWARE CONTROLLED PCB HOLE-POSITION PROCESSING SYSTEM

by

Tayfun Demir

B.S. in E.E., Boğaziçi University, 1981

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Boğaziçi University

1984

To Vildan

ACKNOWLEDGEMENTS

I am grateful to my thesis supervisor Doç.Dr. Okyay Kaynak,for his helps,guidance and cooperation,and especially acknowledge his encouraging supervision in our work to design and operate the realized system.

I would also like to express my thanks to Y.Doç.Dr. Vahan Kalenderoğlu,for his guiding helps on the mechanical construction and design of the scanner,and to Y.Doç.Dr. Ömer Cerid for his valuable suggestions on hardware problems of the system.

I also thank Vildan Polos for her encouragement in my works and her skillful helps in the typing of the manuscript.

ABSTRACT

The purpose of the thesis is to design and realize a microprocessor-based system to process and simulate the drilling of hole-positions in printed circuit boards under software control.

System is based on Z-80 microprosessor which controls a stepper motor driven mechanical moving stage scanner. Scan-control,detection and drilling of hole-positions are performed by the Z-80 microprocessor-based card which is connected to the drivers of the stepper motors through which the power requirements of the motors are supplied during acceleration,steering and deceleration of the mechanical stages.

In the developed system,hardware is minimized, giving all possible controls to the software.System can also be viewed as an intelligent system,since the detection of hole positions is done by optical means but not manually. Also,the drilling process utilizes optimum-path concepts, minimizing drilling time.

ÖZETÇE

Bu tezin amacı,baskı devreler üzerine açılacak delik yerlerinin,mikroişlemci denetiminde saptanması ve delinmesini simüle eden bir sistemin tasarım ve gerçekleştirilmesidir.

Sistem,Z-80 mikroişlemci kontrolunda olan adımlayıcı motorların sürdüğü iki boyutta hareketli bir mekanik tezgahtan oluşur.Tarama kontrolu,delik yerlerinin saptanması ve delimi,Z-80 mikroişlemci kart tarafından yapılır.Bu kart,aynı zamanda mekanik tezgaha hareket sağlayan motorlara,hızlanma,maksimum hızda sürme,yavaşlama sırasında gereken güç gereksinmelerini karşılayan sürücülere bağlıdır.

Geliştirilen bu sistemde donanım en aza indirilerek, mümkün bütün kontrollar yazılım denetimindedir.Delik yerlerinin sisteme girilmesi optik yollarla sağlanmış olup,delme işlemi sırasındada optimum yol kavramı doğrultusunda,zamanlama en aza indirilmiştir.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

I. INTRODUCTION

Even though, in today's technology Computer Aided Design (CAD) is becoming more important, in which case the drilling hole positions are known from the design process, presently many of the board layouts are drawn by hand. Thus at this moment, in order to make the drilling of these holes, either automatic drilling machines are used, or drilling is performed manually using a drill.

Both of the mentioned drilling operations have disadvantages. If an automatic drilling machine is used, hole positions must be manually stored on paper-tape or cassette or an input has to be made directly into the memory of the drilling machine which then performs the drilling operation according to the data recorded into its memory. In this case, alignment of the holes is at its maximum accuracy with minimum process time.

When drilling is done by hand using a drill, one can not talk about accuracy in alignment, or time taken in order to complete even a small card.

As can be understood, even generating the list of drilling positions or calculation of their coordinates, is a very substantial part of the total production time, and when drilling operation is included, the time required is twice as much.

In this thesis,a proto-type system is designed and realized which is capable of detecting the hole positions by itself,using its scanning and detection programs,then performing the drilling of these appropriately recorded hole positions under the control of its drilling program which utilizes a developed optimum-path algorithm written for this specific application,including the drilling accuracy of an automatic drilling machine with process time minimization.

II. SYSTEM LAYOUT

General architecture of the system is given in the below figure.

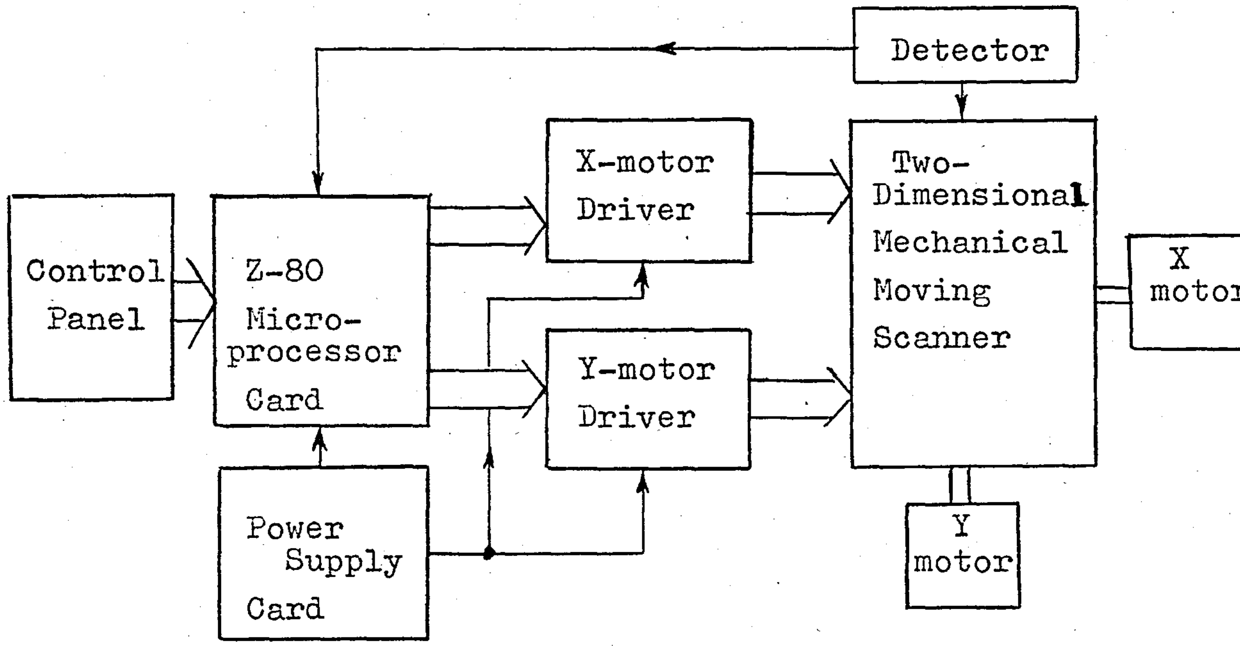


Figure 1.1 System Architecture

System scans the previously prepared dot mask, located onto the upper stage, in a meander pattern, taking data at the end of every 80 step of the stepper motors under the control of the related routine within the software of the system.

Control program is processed by the Z-80 microprocessor card and the required step sequence is generated which is then fed to the motor drivers connected to the stepper motors.Scanning and detection routines continue interactively until the whole layout is scanned.Then,control is transferred to the drilling program.

From this point on,the detector can be visualized as a drill,and drilling is performed repeatedly,according to the data obtained during the scanning and detection routines,waiting a few seconds on the dot to be drilled, simulating the drilling process.This program utilizes an optimum-path algorithm specifically written for this kind of application.

Detailed explanations and calculations are given under specific headings and chapters.

CHAPTER 2

I. STEPPER MOTOR BASICS

A stepper motor is a device that converts electronic signals into discrete mechanical motion.Each time the direction of the current in the motor windings is changed,the motor output shaft rotates a specific angular distance.The motor shaft can be driven in either clockwise or counter-clockwise direction and can be operated at very high stepping rates up to 20,000 steps per second.

Stepper motors offer many advantages as an actuator in a digitally controlled positioning system.They are easily interfaced with a microcomputer or a microprocessor in order to provide opening,closing,rotating,reversing,cycling and highly accurate positioning in a variety of applications. Mechanical components such as gears,clutches,brakes and belts are not needed since stepping is accomplished electronically.There is no need for any feedback device such as a tachometer or encoder.Because the system is open loop,the problems of feedback loop phase shift and resultant instability common to servo drives are eliminated.However,if desired,a minor loop may be closed around the stepper motor with an encoder for system performance enhancement.

Stepper motors are available in a range of frame sizes and with standard step angles of 0.72 ,1.8 ,5 ,15 , 18 degrees and 0.9, 2.5, 7.5, 9 degrees(half-angle) with step accuracies of 3 per cent or 5 per cent noncumulative.

A. Operation

Stepper motors operate on phase-switched d.c.power. If the motor is a 1.8 degree per step motor, the shaft advances 200 steps per revolution when a four-step input sequence (full-step mode) is used and 400 steps per revolution (0.9 degree per step) when an eight-step input sequence (half-step mode) is used.

STEP	SW1	SW2	SW3	SW4
1	ON	OFF	ON	OFF
2	ON	OFF	OFF	ON
3	OFF	ON	OFF	ON
4	OFF	ON	ON	OFF
1	ON	OFF	ON	OFF

(FULL-STEP MODE)

TABLE 2.1 Four-step input sequence

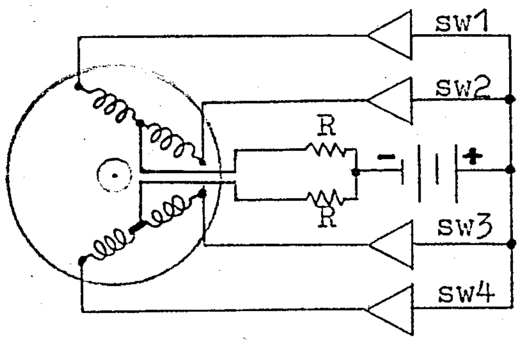


FIGURE 2.1 DC Stepping circuit

STEP	SW1	SW2	SW3	SW4
1	ON	OFF	ON	OFF
2	ON	OFF	OFF	OFF
3	ON	OFF	OFF	ON
4	OFF	OFF	OFF	ON
5	OFF	ON	OFF	ON
6	OFF	ON	OFF	OFF
7	OFF	ON	ON	OFF
8	OFF	OFF	ON	OFF
1	ON	OFF	ON	OFF

(HALF-STEP MODE)

TABLE 2.2 Eight-step input sequence

1-2-3-4-1 sequence in full-step mode, and 1-2-3-4-5-6-7-8-1 sequence in half-step mode provide clockwise rotation of the shaft of the motor. For counter-clockwise rotation of the shaft, switching steps are performed in the following order : 1-4-3-2-1 in full-step, 1-8-7-6-5-4-3-2-1 in half-step mode.

Apart from the Four-step and Eight-step drive methods, there is one more drive method which is called Wave Drive. Energizing only one winding at a time is the so-called Wave Excitation.

STEP	SW1	SW2	SW3	SW4
1	ON	OFF	OFF	OFF
2	OFF	OFF	OFF	ON
3	OFF	ON	OFF	OFF
4	OFF	OFF	ON	OFF

TABLE 2.3 Wave Drive

This type of excitation also produces the same increment as the four-step sequence.Since only one winding is on,the hold and running torque with rated voltage applied will be reduced 30 per cent.Within limits,the voltage can be increased to bring output power back to rated torque value.The advantage of this type of drive is increased efficiency while the disadvantage is decreased step accuracy.

Also in the multiple stepping case,the pulses can be timed to shape the velocity of the motion,slow during start,accelerate to maximum velocity,then decelerate to stop with minimum ringing.

B. Torque

1. Holding Torque

At standstill (zero step per second and rated current) the torque required to deflect the rotor a full step is called the holding torque.Normally,the holding torque is higher than the running torque and thus acts as a strong brake in holding a load.Since deflection varies with load, the higher the holding torque the more accurate the position will be held.

2. Residual Torque

The non-energized detent torque of a permanent-magnet stepper motor is called Residual Torque.As a result of the permanent magnet flux and bearing friction,it has a value of approximately 1/10 the holding torque.This characteristic of permanent magnet steppers is useful in holding a load in the proper position even when the motor is de-energized. The position however will not be held as accurately as when the motor is energized.

3. Dynamic Torque

A typical speed/torque characteristic curve is shown below.

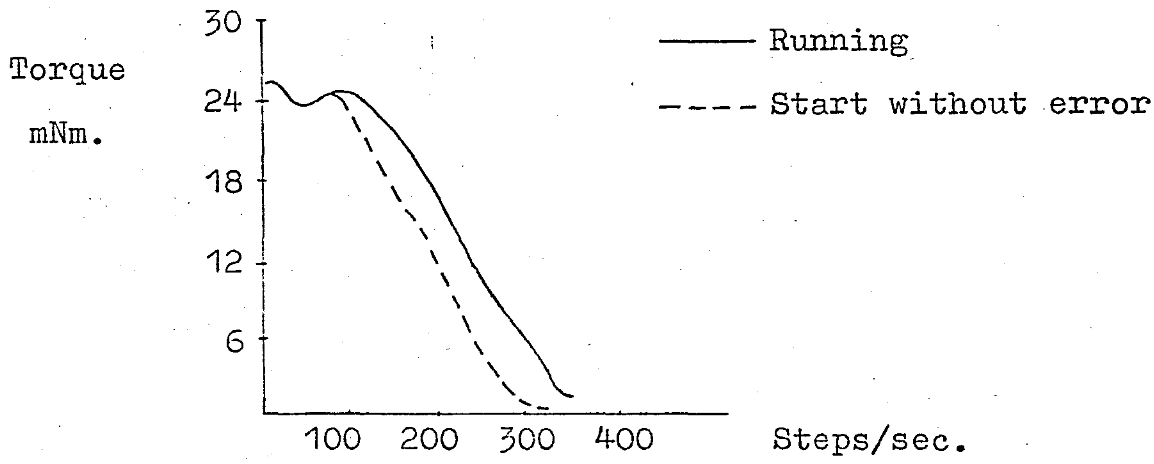


Figure 2.2 Airpax K82402 L/R Stepper Speed/Torque

The Start Without Error curve shows what torque load the motor can start and stop without loss of a step when started and stopped at a constant step or a pulse rate.

The Running curve is the torque available when the motor is slowly accelerated to the operating stepping rate. It is the actual dynamic torque produced by the motor.This curve is also called the SLEW curve.

The difference between the Running and the Start Without Error torque curves is the torque lost due to the accelerating the motor rotor inertia.

C. Resonance

When a stepper motor is operated at its natural frequency,typically 90 to 160 steps per second,depending on motor type,an increase in the audio and vibration level of the motor may occur.The frequencies at which this resonance will occur vary widely depending on the characteristics of the load.

In some cases,the motor can oscillate or loose steps, while in other applications,resonance may not be experienced. Where resonance does occur,an increase in inertial loading will usually allow operation at these frequencies.

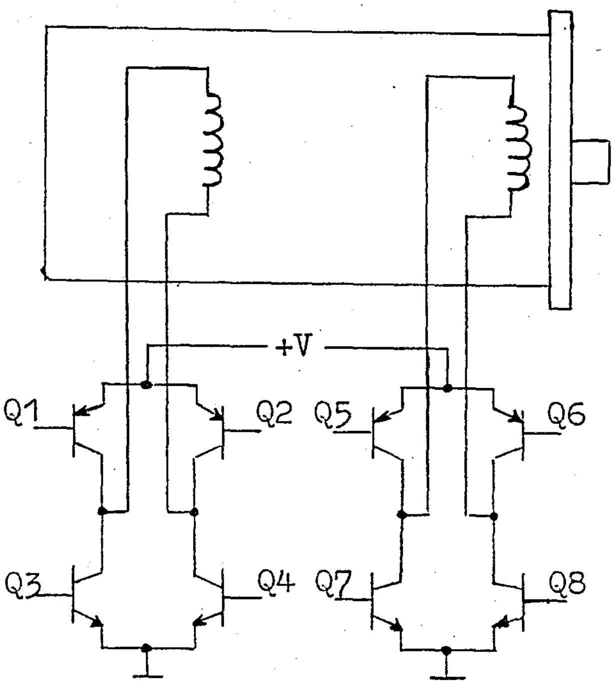
A permanent magnet stepper motor,however,will not exhibit the instability and loss of steps often found in variable reluctance stepper motors since the permanent magnet motors have higher rotor inertia and a stronger detent torque.

D. Bipolar & Unipolar Operation

There are steppers with either 2 coil bipolar or 4 coil unipolar windings.

The stator flux with bipolar winding is reversed by reversing the current in the winding.It requires a push-pull bipolar drive as shown in Figure 2.3.One must be careful in the design of the circuit so that the transistors in series do not short the power supply by turning on at the same time.Properly operated,the bipolar winding gives the optimum motor performance at low to medium step rates.

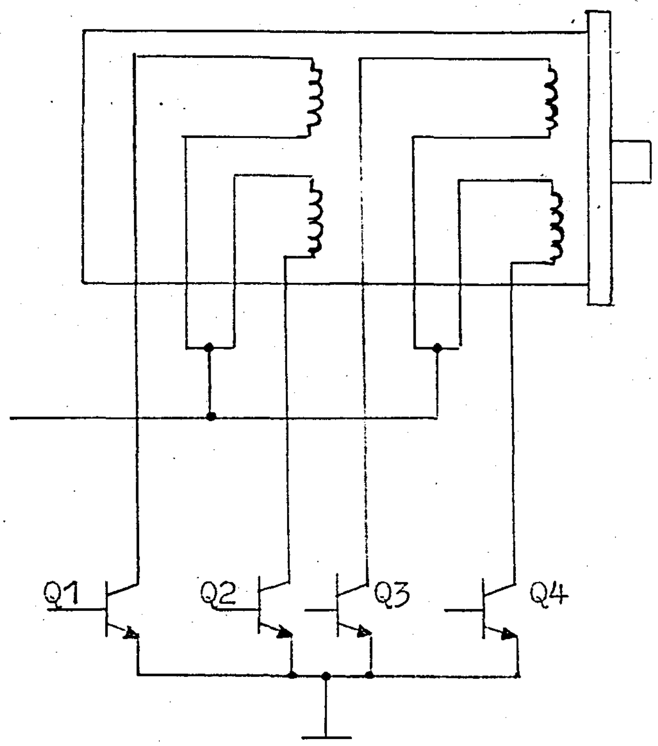
A unipolar winding has two coils wound on the same



	STEP	Q_{1-4}	Q_{2-3}	Q_{5-8}	Q_{6-7}	
CW Rotation ↓	1	ON	OFF	ON	OFF	CCW Rotation ↑
	2	ON	OFF	OFF	ON	
	3	OFF	ON	OFF	ON	
	4	OFF	ON	ON	OFF	
	1	ON	OFF	ON	OFF	

Figure 2.3 Bipolar Switching Sequence

bobin per stator half.Flux is reversed by energizing one coil or the other coil from a single power supply.Unipolar case allows the drive circuit to be simplified.Only four power switches are required,and the timing is not as critical as in the bipolar case,(refer to figure 2.4) to prevent the short through two transistors.



CW Rotation ↓ / CCW Rotation ↑

STEP	Q1	Q2	Q3	Q4
1	ON	OFF	ON	OFF
2	ON	OFF	OFF	ON
3	OFF	ON	OFF	ON
4	OFF	ON	ON	OFF
1	ON	OFF	ON	OFF

Figure 2.4 Unipolar Switching Sequence

For the unipolar motor to have the same number of turns per winding as the bipolar motor,the wire diameter must be decreased and therefore the resistance increased. This results 30 per cent less torque for the unipolar motor at low steprates.At higher rates,torque outputs are equal.

E. Performance Improvement & Drive Types

If a motor is operated at a fixed rated voltage and if its frequency (i.e. its step rate) is tried to be increased, the torque output decreases because of the rise time of the coil which limits the power delivered to the motor. And this effect is due to the inductance to resistance ratio of the circuit.

This may be compensated by raising the power supply voltage and adding a series resistor (as shown in figure 2.5) or by increasing the power supply voltage to obtain a constant current as the step rate increases.

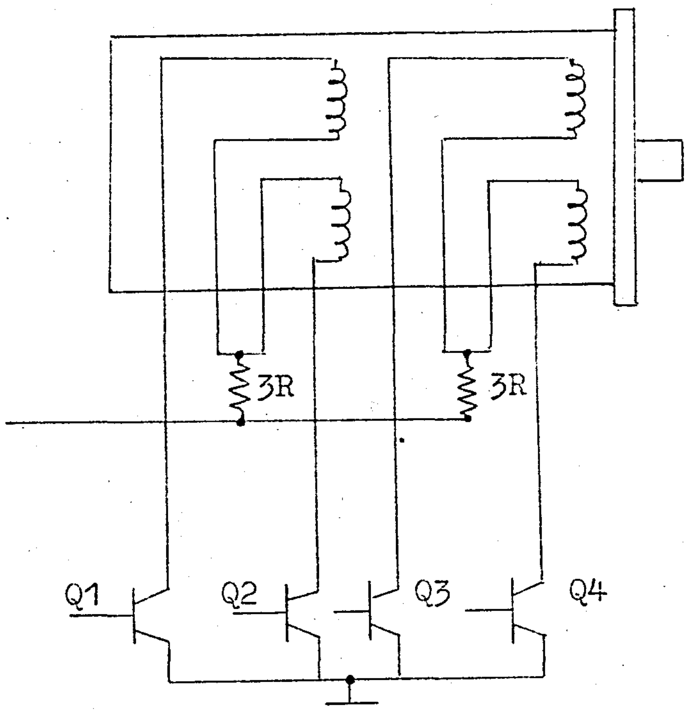


Figure 2.5 L / 4R Drive

For L/4R drive, series resistor is selected three times the motor winding resistance. Supply voltage is increased to four times the motor rated voltage. It can easily be understood that power supplied to the system also increases by a factor of four with respect to L/R drive.

For power minimization, bi-level or chopper drives may be selected.

1. Bi-level Drive

At zero step per second, this type of drive holds the motor at a lower voltage than rated voltage, and higher voltage when stepping. It is most efficient when operated at a fixed, constant stepping rate.

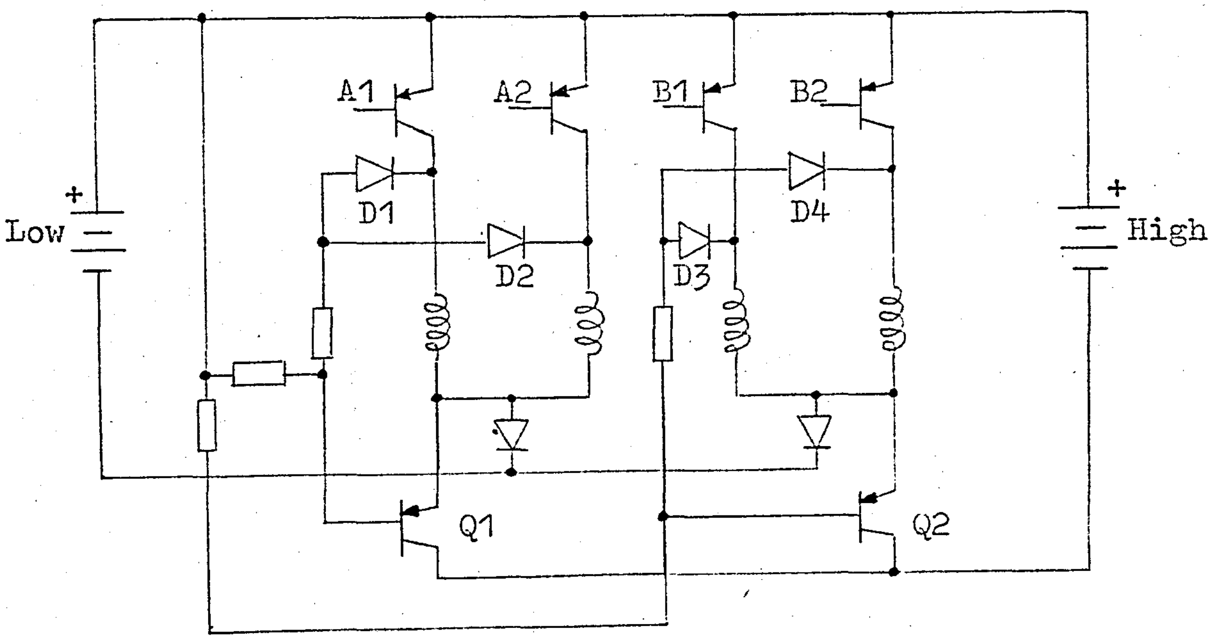


Figure 2.6 Bi-level Drive (Unipolar)

The high voltage source is put on through a current sensing resistor or by the circuit in figure 2.6 which uses the inductively generated turnoff current spikes to control the voltage.At standstill,the low voltage source energizes the motor windings.As the stepping sequence is fed to the windings,diodes $D_{1,2,3,4}$ are used to make the high voltage transistors $Q_{1,2}$ conduct.

2. Chopper Drive

Such a drive maintains an average current level, using a current sensing resistor to turn on the high voltage supply until an upper current level is obtained,and turn off the high voltage until a low level is sensed.Then it turns on the high voltage again.

This type of drive is best for fast acceleration and variable frequency applications,and more efficient than a constant current amplifier regulated supply.In a chopper drive,the voltage supply must be five to ten times the motor voltage rating.

F. Transient Voltage Suppression

Transient voltages are generated as current is switched through the windings during stepping.These voltages can cause faulty operation and damage the motor or drive components unless a means of limiting or removing them is provided.The most common method for suppressing transient voltages is to use shunting diodes across each winding.Since this reduces torque,voltage is allowed to rise to more than twice the supply voltage across the switching transistors.

In order to achieve this, a zener or a series resistor is added for faster induced field, faster current decay, better performance.

G. Performance Limitations

Increasing the voltage to a stepper motor at standstill or low stepping rates will produce a proportionally higher torque until the magnetic flux paths within the motor saturate. As the motor nears saturation, it becomes less efficient and thus does not justify the additional power input.

The maximum speed a stepper can be driven is limited by hysteresis and eddy current losses. At some rate, the heating effects of these losses limit any further effort to get more speed or torque output by driving the motor harder.

II. CHARACTERISTICS OF THE STEPPERS USED IN THIS PROJECT

Stepper motors used in the proposed project have the following specifications whose related performance charts are given in the appendix.

Manufacturer: ORIENTAL MOTORS

Type: PH296-03

Voltage: 14 V

Current per phase: 0.7 A/phase

Holding torque: 174 oz-in (123 N cm)

Resistance per phase: 20 ohms/phase

Inductance per phase: 60 mH/phase

Working temperature range: -10°C to 50°C (14°F to 122°F)

Temperature rise: 80°C (176°F)

Insulation type: Class B

Insulation resistance: 100 MΩ or more when megger reading is DC 500 V

Dielectric strength: Withstands in normal when impressing 0.5 kV at 60 Hz. between the windings and the frame for one minute

CHAPTER 3

SYSTEM HARDWARE

Hardware of the system consists of the following the blocks and explained as listed.

A. Z-80 microprocessor card

B. Mechanical Assembly of the Scanner

C. Stepper Motor Drivers

D. Detector

E. Power Supply

Although each heading above will be considered in detail ,main spec's of the hardware are;

i. Z-80 is used as the CPU,and a Z-80 PIO for input/output purposes.

ii.Control programs are about three Kbytes long and stored in two 2716 EPROMs.

iii.Maximum scanning area of the system is 220mm by 150 mm (x,y) mechanically.

iv.System scans the nodes of a grid pattern whose nodes are 1/20 inch (1.27mm) apart.

v. Since a two kilobyte random access memory is used in the system,memory available for storing the hole-position information limits the card size to 170mm by 140mm (x,y),because RAM is also used as a stack and a general purpose store area for program constants.

A. Z-80 Microprocessor Card

In the design of the system,Z-80 microprocessor is chosen as the CPU,whose block diagram of the internal structure is shown in the below figure.

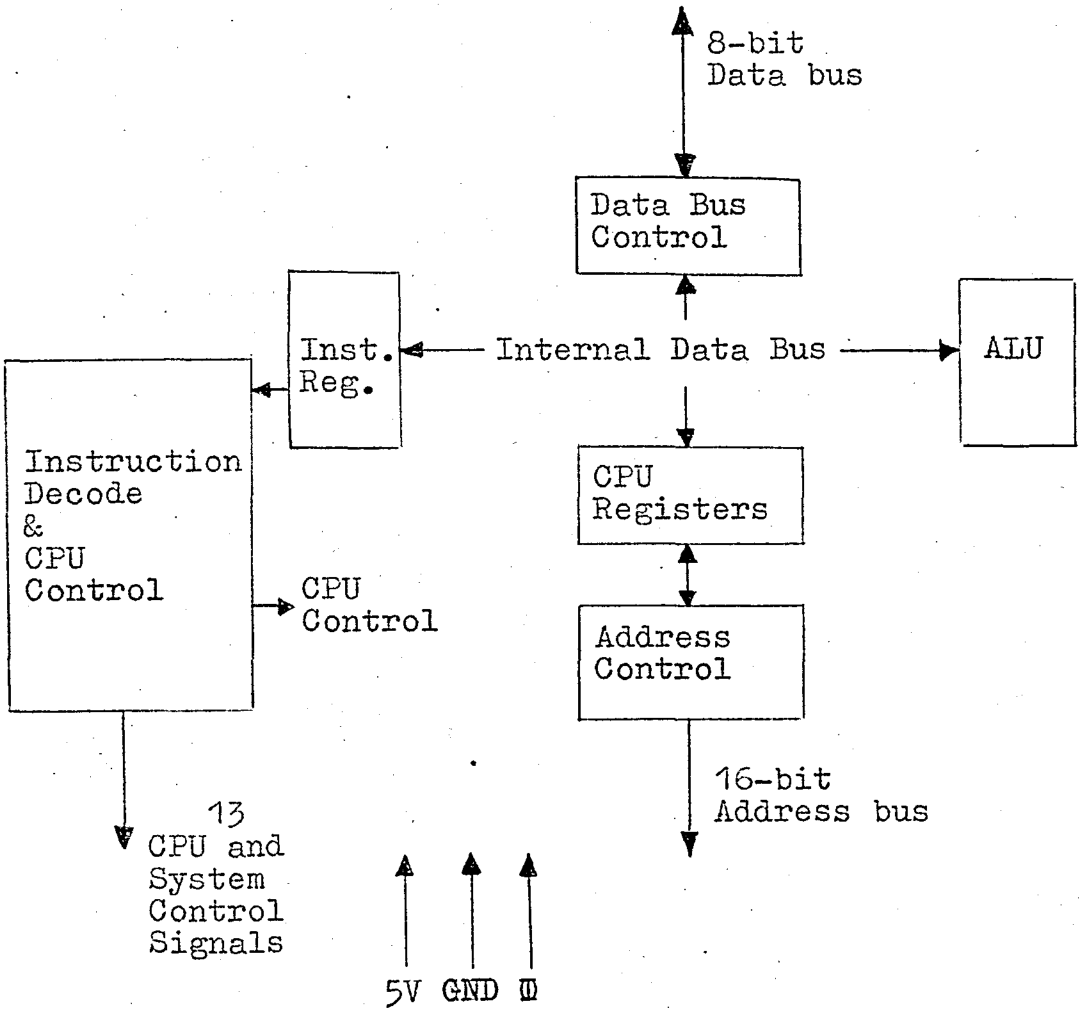


Figure 3.1 Z-80 CPU Block Diagram

Z-80 is an 8-bit processor with eighteen 8-bit registers,and four 16-bit registers.The registers include two sets of six general purpose registers that may be usedindividually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers.

Z-80 CPU can execute 158 different instruction types including all 78 of the 8080A CPU.The instructions can be broken down into the following major groups:

a. Load and Exchange

b. Block Transfer and Search

c. Arithmetic and Logical

d. Rotate and Shift

e. Bit Manipulation (set,reset,test)

f. Jump,Call and Return

g. Input/Output

h. Basic CPU Control

Also,the type of addressing modes avaliable in Z-80 CPU include; Immediate,Immediate Extended,Modified Page Zero, Relative,Extended,Indexed,Register,Implied,Register Indirect and Bit Addressing modes.

Apart from Non Maskable Interrupt,the CPU can be programed in any one of the three maskable interrupt modes, MODE 0,1,2.

Details of the above mentioned characteristics of the Z-80 CPU can be found in the Appendix.

Input & Output actions are done by Z-80 PIO Parallel I/O which has two parts and provides a TTL compatible interface between peripherals and Z-80 CPU.

Memory devices are; 2 EPROMS of 2716 type and a RAM, 6116 P-3, 2kx8 bit capacity.

System clock frequency is 2.0MHz which is obtained from 4.00 MHz. crystal,dividing this by 2 using a D-type flip-flop.

Address Decoding of the system is done as shown below:

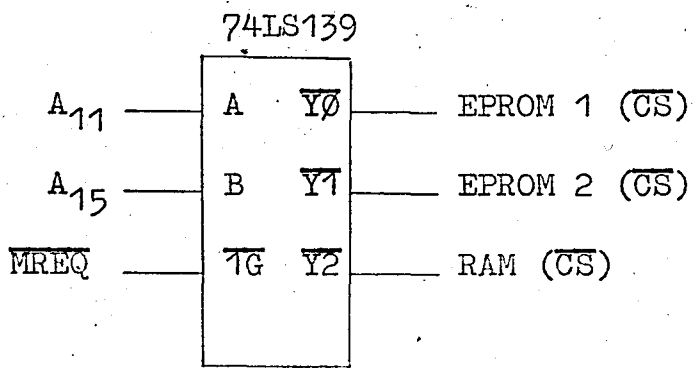


Figure 3.2 Address Decoding Circuity

This scheme of decoding gives such a memory layout;

ØØØØ - Ø7FF	EPROM1
Ø8ØØ - ØFFF	EPROM2
1ØØØ - 7FFF	Unclassified
8ØØØ - 87FF	RAM

Table 3.1 Memory Layout of the System

Mode 2 interrupt mode is selected in order to connect control switches to the system.Using this mode a table of 16-bit starting addresses is obtained for every input service routine.When an interrupt is accepted,a 16-bit pointer is formed to obtain the desired interrupt service routine starting address from the table.The upper 8 bits are loaded to the I register,where the lower eight bits of the pointer are supplied by the interrupting switches.But only 7 bits can be used,as the least significant bit must be zero.This is required since the pointer is used to get two adjacent bytes to form a complete 16-bit service routine starting address and the addresses must always start in even locations.

This mode is used together with the hardware below.

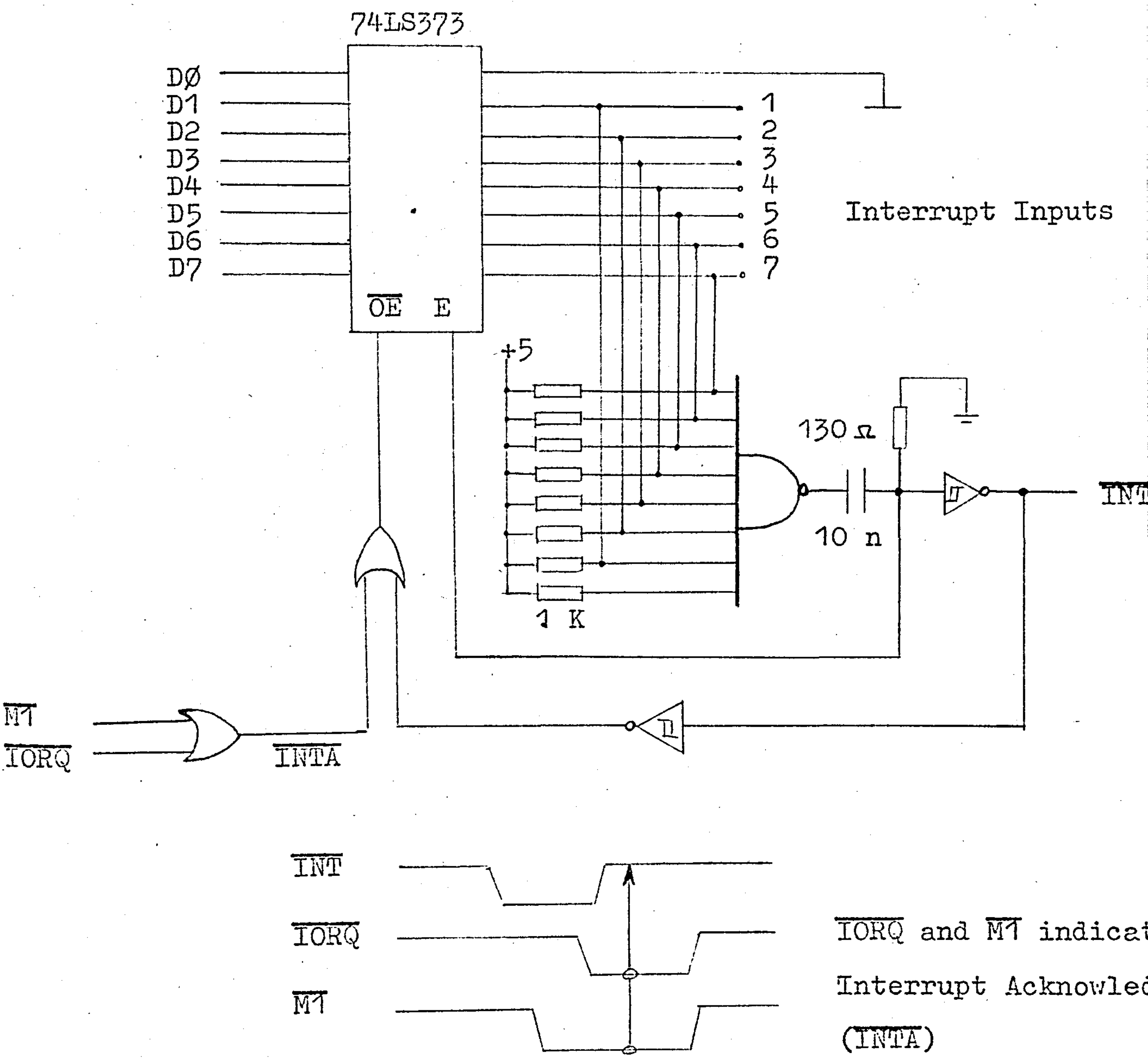


Figure 3.3 Interrupt Hardware

When one of the switches connected to interrupt inputs is drawn to ground,NAND gate output goes high and sends a 1.3 microseconds interrupt pulse through capacitor,resistor network,and at this moment,74LS373 Octal Transparent Latch is enabled and it latches the data on its input.Meanwhile the CPU generates an $\overline{\text{INTA}}$ signal which enables the output of the latch,placing the data taken from the switches to the data bus.This information supplies the lower 8-bits of the 16-bit address.Thus CPU jumps to the desired location to run the special program,this specific switch wants to run.

First,PortA of PIO is programmed as output,Port B as input.Then,after,detection and scanning program,Port B also, is programmed as output.

RESET ROUTINE

ØØØØ	31E587	LD SP,87E5H
ØØØ3	ED5E	IM 2
ØØØ5	3EØ7	LD A,Ø7
ØØØ7	ED47	LD I,A
ØØØ9	3EØF	LD A,ØFH
ØØØB	D3Ø2	OUT (Ø2),A
ØØØD	3E4F	LD A,4FH
ØØØF	D3Ø3	OUT (Ø3),A
ØØ11	AF	XOR A
ØØ12	D3ØØ	OUT (ØØ),A
ØØ14	FB	EI
ØØ15	76	HALT

Table 3.2 Reset Routine

This routine,sets stack pointer to 87E5H,chooses interrupt mode 2,loads I register with Ø7,programs A port of PIO as output and B port as input.

Interrupt Service Routine Starting Addresses:

START	Switch	- Ø7BE	:	1F
		Ø7BF	:	ØØ
STOP	Switch	- Ø7DE	:	25
		Ø7DF	:	Ø7
Pos-x	Switch	- Ø7FC	:	ØØ
		Ø7FD	:	Ø7
Neg-x	Switch	- Ø7FA	:	4Ø
		Ø7FB	:	Ø7
Pos-y	Switch	- Ø7F6	:	65
		Ø7F7	:	Ø7
Neg-y	Switch	- Ø7EE	:	DØ
		Ø7EF	:	Ø6

Table 3.3

NMI is connected to the emergency switch on the control panel.Output Port A is connected to the stepper-motor drivers:4-least significant pins to x-motor,4-most significant pins to y-motor.Input Port B is connected to the detector circuity (Pin BØ).The other six pins are grounded.But Pin B7 is connected as shown below:

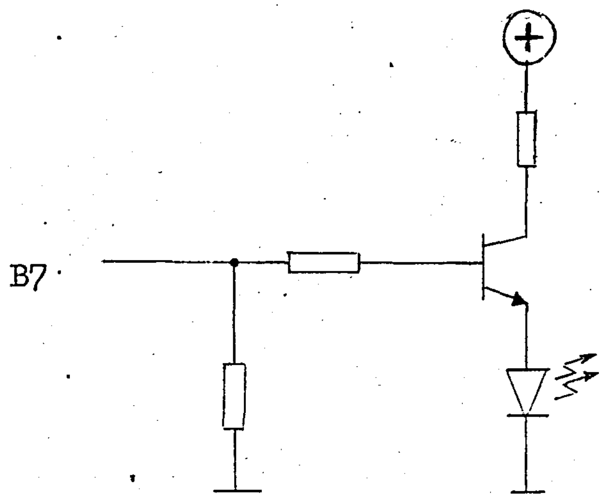


Figure 3.4 Drill Simulator

This circuit enables us to simulate the drilling process when the detector comes onto a dot,as if it is drilling the hole,the LED flashes.During these operations Port B is also programmed as output port.

B. Mechanical Assembly of the Scanner

Stage scanner used in this system is mainly composed of two stages mounted on top of each other with the following specifications.

Since for this very special application, in which very high precision motion is needed, the lead-screws and their corresponding nuts should not impose any backlash to the system. Although there are ways to avoid this phenomena such as coupling the lead-screws with adjustable nuts or using spring systems, in the prototype, backlash did not cause any trouble, which is one of the best spec's of the mechanical system.

Mechanical system, from bottom to up is made up of:

1) Base Plate (dim. 260 x 350 x 4mm)
2) y-stage (dim. 420 x 195 x 4mm)
3) x-stage (dim. 270 x 220 x 4mm)
4) Glass Plate (dim. 270 x 220 x 4mm)

Base plate holds the two vertical plates between which the lead-screw and the carrier-shafts are mounted that carry the y-stage.

Y-stage is on top of a carrier-sub-plate which is mounted onto the nut and the linear bearing housings by which a free slide on the shafts can be achieved. Y-stage holds the other two vertical plates. On these vertical plates, the other carrier-shaft pair and the other lead-screw is found. All stages and plates are aliminum.

X-stage, like y-stage is also carried by another sub-plate which is mounted to the nut of this lead-screw.

Mechanical system, being as mentioned above, uses linear motion bearings which gives the stages almost frictionless, smooth linear motion. Each stage has 4 of these bearings, (IKO, D=16 type) imposing a balanced load to the carrier-shafts Diameter of the shafts is 16mm.

Lead screws have 8 threads per inch. Meaning that, one revolution of the screw moves the nut 1/8 inch. Since the stepper motors, used in this system, have 1.8 degree step angle, giving one revolution with 200 pulses, as a result, one pulse to the steppers moves the stages 1/1600 inch. Since no-blacklash is experienced, this 1/1600 inch happens to be the resolution of the system described.

Coupling between the shafts of the steppers' and the lead-screws is done as shown below:

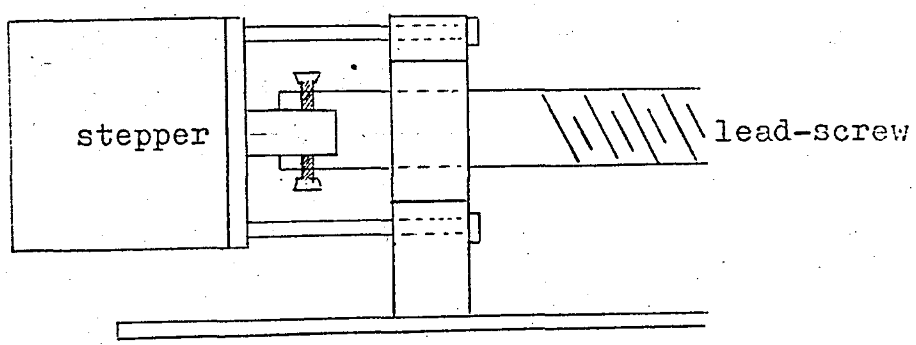


Figure 3.5 Motor Coupling

On top of the x-stage, a glass-plate is mounted on which the dot-mask is placed for processing. Glass plate, x-plate combination can be seen in the below figure:

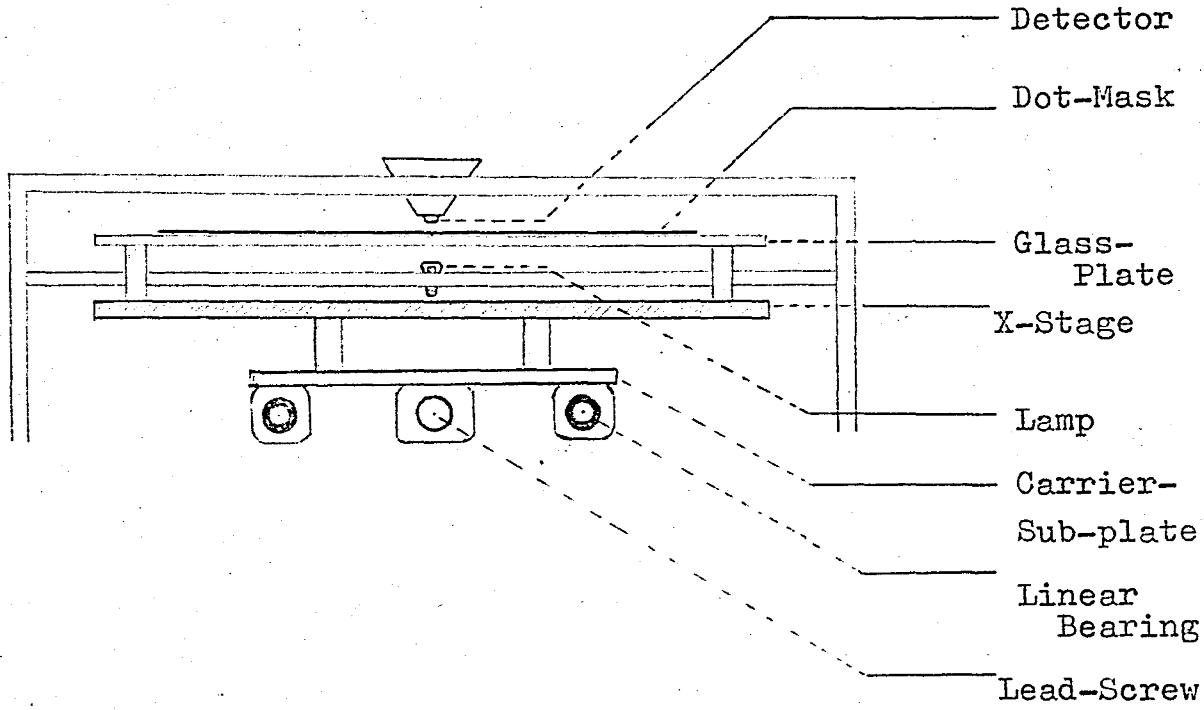


Figure 3.6 X-stage

The mechanical assembly, having the previously mentioned characteristics, gives a scanning area of 220 mm by 150 mm. (x , y)

C. Stepper Motor Drivers

Although the drive circuitry of a stepper motor has to be determined according to the requirements of the application (such as the amount of torque needed,speed,acceleration) stepper motors which are used in the system were available before the mechanical assembly (i.e. stage scanner) is realized.So,there was the obligation of making a suitable drive circuitry for these stepper motors considering how a scanner design may be the best for the application.

Since in this application,scanning is done with very small intervals (1.27mm),time is a very important factor in the process.So,motors have to be forced to be driven at the probable maximum speed,by acceleration,steering at that speed and deceleration finally.

Mechanical assembly is so designed that almost frictionless movement is obtained which means that there was no need for high torque output from the steppers.This characteristic of the scanner created a chance to speed up the stepper motors.

When speed vs torque characteristic of the steppers is examined,it is seen that as speed increases,torque output decreases accordingly.Since slow constant speed is also needed when acceleration is not possible because of short paths,studying the Start without Error Curve (the stepping rate which a motor can start and stop without losing a step) indicated that,a speed around 200 steps per second,will be good enough for the constant speed requirements.

Since high torque output is not needed for this application, it is decided to choose a drive circuitry which is best for fast acceleration and variable frequency operation; and a chopper drive is preferred after studying the drive types.

An inductor of inductance L, having resistance R, behaves according to the below formula when a voltage V is applied on its terminals:

$$V = L\frac{di}{dt} + Ri$$

Taking Laplace Transform, we get,

$$\frac{V}{s} = sLI(s) + RI(s)$$

$$I(s) = \frac{V}{s(sL+R)} = \frac{(V\ /\ R)}{s} - \frac{(V\ /\ R)}{s+\frac{R}{L}}$$

$$I(t) = \frac{V}{R}(1-\exp(-tR/L))$$

In our case we equate the above equation to 0.7 A rated current, and we obtain,

$$0.7 = \frac{V}{20}\left(1-\exp(-t/3x10^{-3})\right) \quad \text{where } R=20 \text{ ohms}, \ L=60 \text{ mH}$$

From here,

$$t = -3x10^{-3}\left(\ln\left(1-\frac{14}{V}\right)\right)$$

If V=30 V, then t=1.8 ms.

If V=42 V, then t=1.2 ms.

It is seen that increasing the voltage applied to a winding, increases the time to reach a specific current. From the above equation, we deduce that a voltage of approximately 42 volts can give an operation frequency of 834 Hertz. This fact is one of the important design criteria of the motor drivers.

Drive circuitry of one stepper motor is designed as in figure 3.7, and the operation of the circuit is as the following:

According to the coming excitation pulse sequence from the microprocessor card, to either one of the transistors Q_{2A}, Q_{5A}, Q_{2B}, Q_{5B}, corresponding motor winding is chosen and connected to the high voltage through power transistors, Q_{1A}, and Q_{1B}, until the rated current of the windings is reached.

This rated current is detected on current sensing resistors R_{5A} and R_{5B}, turning on transistors Q_{6A} and Q_{6B}, which draw Q_{1A} and Q_{1B} into cut-off. At this moment, the low voltage supply is connected to the windings in order to prohibit the power that will be dissipated on Q_{1A} and Q_{1B} when rated current is passing through.

Diodes D_{3A}, D_{4A}, D_{3B}, D_{4B} and zener diodes Z_A and Z_B are for voltage suppression purposes in order to protect the circuitry from high voltage inductive spikes that may be generated. Diodes D_{2A} and D_{2B} are to inhibit reverse current to low voltage supply, while diodes D_{5A}, D_{6A}, D_{5B}, D_{6B} are included in order to avoid negative spikes through windings.

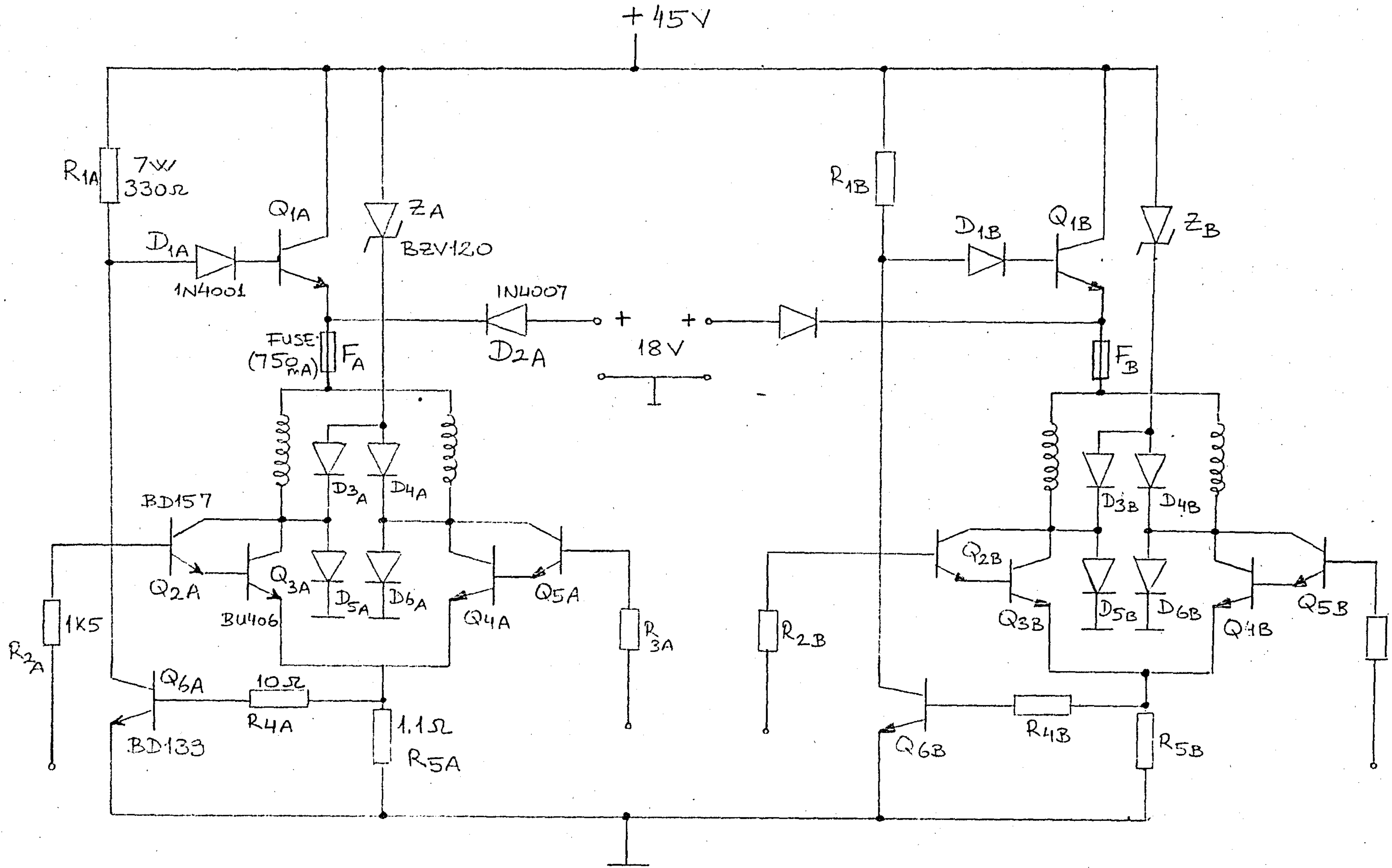


Figure 3.7 Motor Driver Circuitry

D. Detector

Detection system consists of the following components:

1- A photo-transistor as a detector

2- Lamp (12 V)

3- Comparator (LM 324)

Photo-transistor is connected in common-emitter mode, collector voltage being the output. Its base being excited by radiation, photo-transistor gives an output according to the light intensity falling onto the base. Thus, voltage seen on its collector vary with light.

In order to obtain sharp transitions and stable voltages, a comparator is connected to the collector of the photo-transistor as shown in figure 3.8.

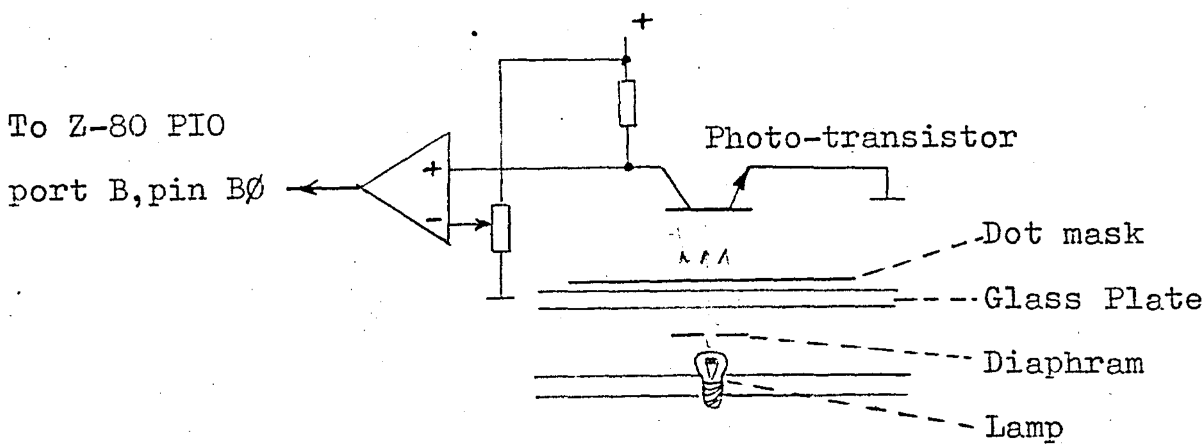


Figure 3.8 Detector Circuitry

By using the potentiometer,the offset voltage is set to 1.2 V,which means that,when the voltage at the collector of the photo-transistor rises to or above 1.2 V,the comparator raises its output to high (4 V).Below 1.2 volts,output is at low level (0.55 V).

In the Z-80 PIO specifications,input low voltage is given as 0.8 V,and input high voltage as minimum 2 V.Since comparator output is connected to Z-80 PIO pin BØ,a wrong data input is avoided,which can be encountered due to the oscillations.

This information coming from the detector through the comparator is used to decide whether there is a hole or not at that node.

Since the photo-transistor has a very narrow sensitivity angle,the lamp has to be located in perfect alignment with the photo-transistor.Also,a diaphram is put on to the lamp,to show only the filament to the photo-transistor.

E. Power Supply

Voltage levels required in the system are:

1) 42 V for the high voltage side of the drivers
2) 18 V for the low voltage side of the drivers
3) 10-12 V for the lamp
4) 5 V for the processor-card, detector, comparator

High voltage is obtained just by rectification through diodes, and filtering bycapacitors using a 32 V AC supply as shown in figure 3.9.

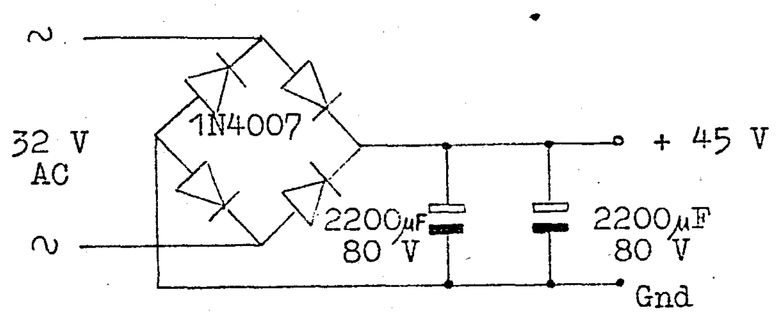


Figure 3.9 High Voltage Supply

Although the output of the above circuitry is around 45 volts at no load, when loaded output drops to 42 volts, which is sufficient to fulfill the high voltage requirement of the driver card.

Other voltage levels are obtained using voltage regulator IC's, as given in figure 3.10. Voltage levels, 18 and 5 volts are derived from 7818 and 7805 regulators with 1 A ratings. Lamp voltage is supplied by using a 723 IC (Variable Voltage Regulator). This supply is made variable in order to set the intensity of the lamp to a level so that the phototransistor output voltage can be at the optimum level.

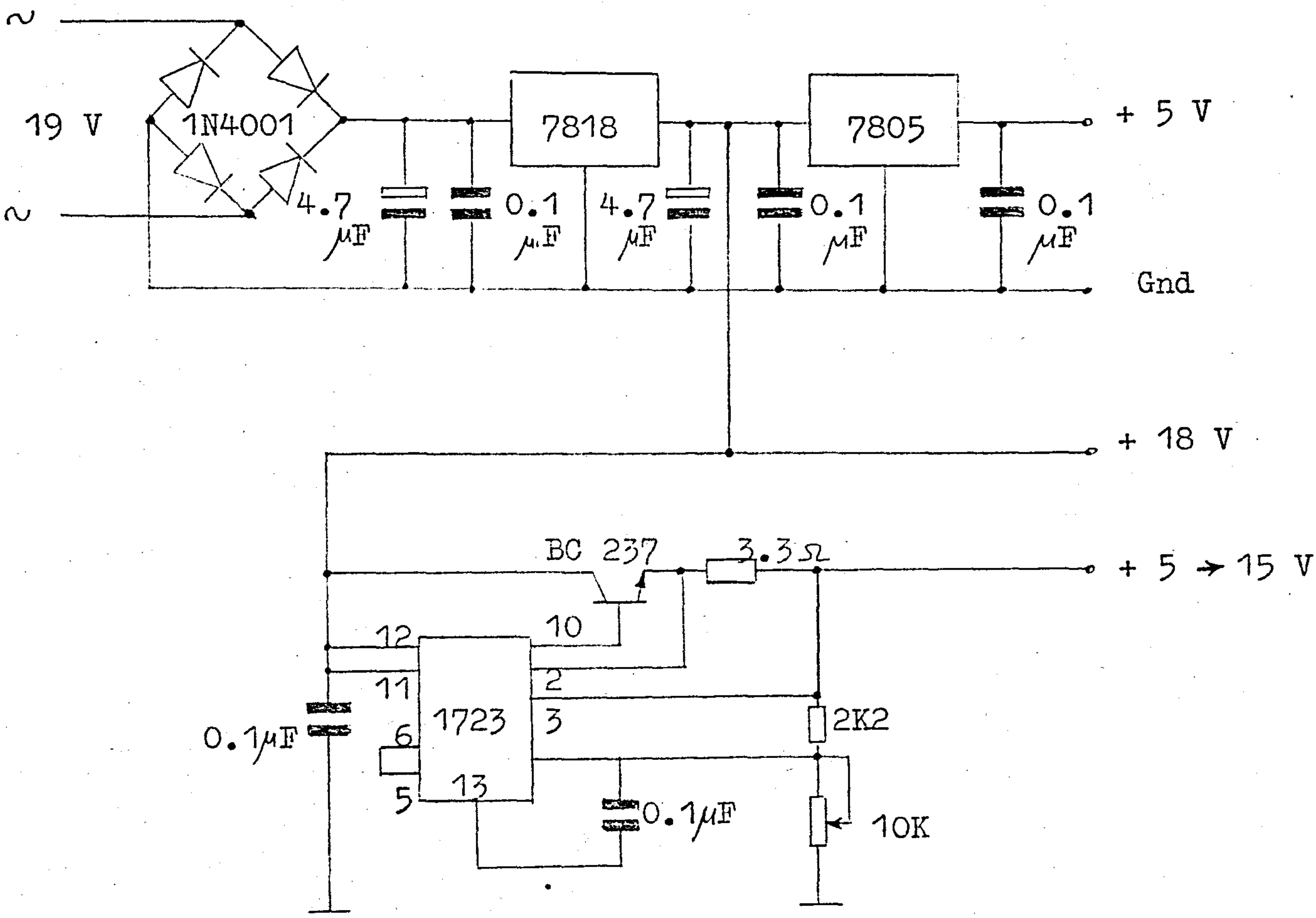


Figure 3.10 Low Voltage Supplies

Approximately 500 mA is drawn from the 45 V supply, and 700 mA from the above supply card (200-250 mA processor-card, 70-80 mA lamp, 350 mA driver).

CHAPTER 4

SYSTEM SOFTWARE

System software can be devided into four main program blocks:

i. Line Detection Program

ii. Frame-length Detection Program

iii. Scanning & Store Hole-Positions Program

iv. Drilling Program

There are also a few sub-programs which are used in each of the above mentioned programs,and these include;

a. Constant Delay & Variable Delay Programs

b. Acceleration & Deceleration subroutines

c. Constant Speed Subroutines

A. SUBROUTINES

I. Constant Delay

This delay routine is used at constant speed movement programs.(Routine DLY).It is 8017 T cycles long.Since system clock frequency is 2 MHz.(one T cycle is 0.5 microseconds),this gives us a delay of;

$$8017 \times 0.5 \times 10^{-6} = 4.0085 \text{ msec. } (249 \text{ Hz})$$

II. Variable Delay

The Variable Delay routine is used where a delay of different durations is needed,namely in the acceleration, maximum speed and deceleration routines (Routine VDLY).Its delay is given by the relation below:

$$((133 \times 0.5 \times 10^{-6})\ M\ ) + 5 \text{ microseconds}$$

Multiplier M,is loaded before the VDLY routine is called,and it determines the duration of the delay.

III. Constant Speed Subroutines

As mentioned before,in order to rotate the rotor of the stepper motor,a four-bit pattern must be sent to the appropriate windings in a special sequence.These patterns are A,9,5,6 in hexadecimal form (1Ø1Ø,1ØØ1,Ø1Ø1,Ø11Øin binary) for clockwise rotation; 6,5,9,A in hex. for counter-clockwise rotation.

'A' port of the PIO is used to output these patterns. The least significant four bits (03,02,01,0Ø) are connected to X-motor, the rest (07,06,05,04) to the Y-motor through the motor drivers.

Since one port is used for both of the stepper motors and one of the motors should be in standstill state while the other is rotating, the bit patterns of the motors should be stored in the memory as shown below for use in the routines.

Memory Address	Bit Pattern	
Ø7AØ H	ØA H	For X-Motor
Ø7A1 H	Ø9 H	
Ø7A2 H	Ø5 H	
Ø7A3 H	Ø6 H	
Ø7BØ H	AØ H	For Y-Motor
Ø7B1 H	9Ø H	
Ø7B2 H	5Ø H	
Ø7B3 H	6Ø H	

Table 4.1 Motor Constants

As it is seen from the table, most significant four bits of the x-motor bit patterns, and the least significant four bits of the y-motor bit patterns are zeroes, not to energize the other motor windings while one of them is rotating, therefore not drawing excess current at standstill.

Using the above mentioned bit patterns and delay routine DLY, the constant speed routines are written as shown in the following figure 4.1.

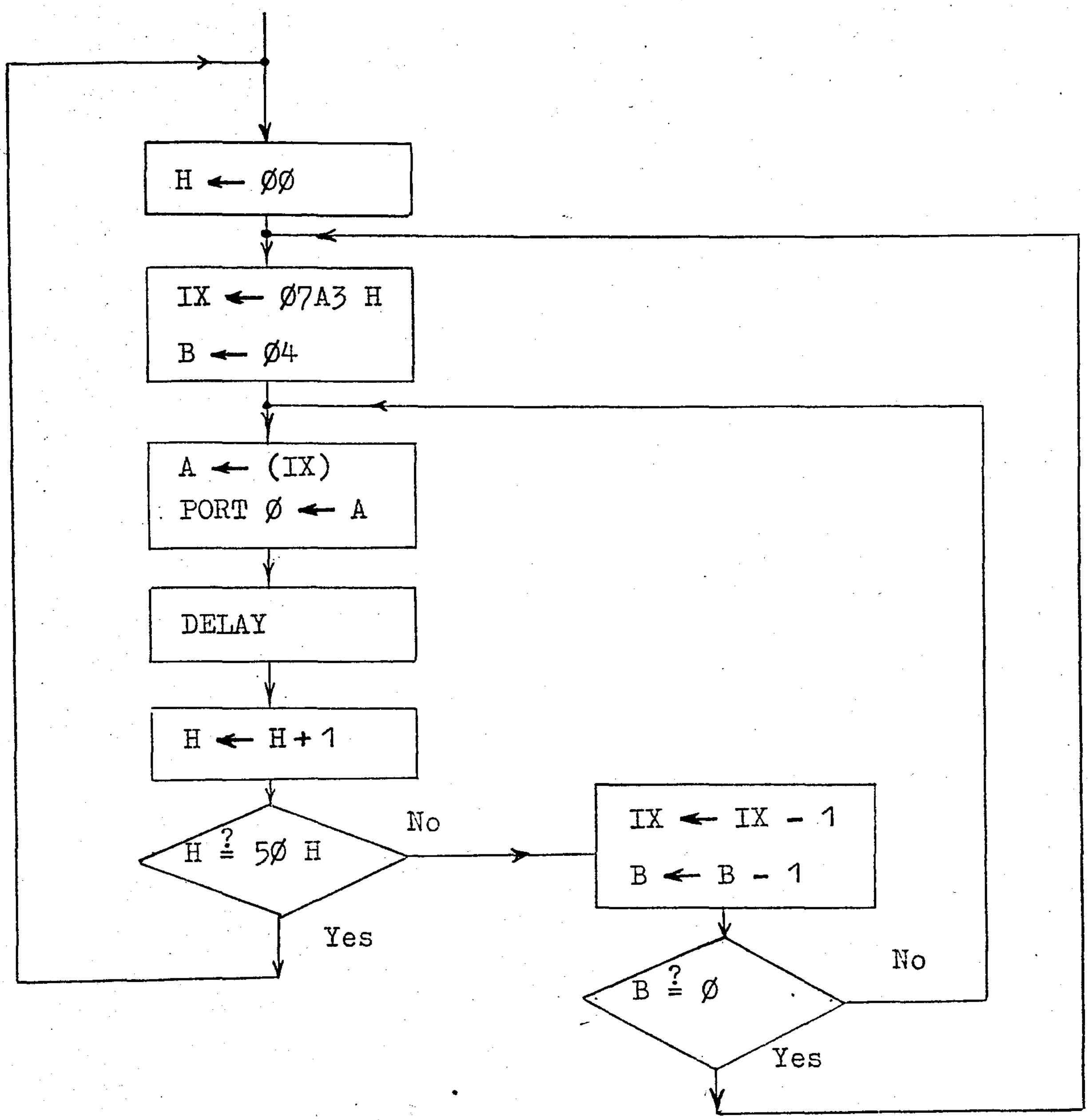


Figure 4.1 X-motor CCW Constant Speed Routine

Since one step of the stepper motor gives a 1/1600 inch linear motion;in order to move one grid length that is 1/20 inch,motor has to be stepped 80 times(80 Decimal = 50 Hex.).

In these constant speed routines H register of the processor is used as the step counter.At every 80 step count, the routine repeats itself to move one more grid length.

Figure 4.1 is the flowchart of the routine to move the x-motor such that the upper (x-table) stage moves in positive-x direction (i.e. away from the motor).

There are four routines, which are called by using the interrupt mode 2 of the Z-80 through the control panel switches, giving manual control of the X and Y stages by which the detector can be positioned on the dot mask anywhere desired. These are negative and positive X and Y direction movement routines, located at;

Ø7ØØ H : X-Stage, Positive-X (away from the motor)
Ø74Ø H : X-Stage, Negative-X (towards the motor)
Ø765 H : Y-Stage, Positive-Y (towards the motor)
Ø6DØ H : Y-Stage, Negative-Y (away from the motor)

These routines move the stages continiously until the STOP button is pressed on the control panel.

IV. Acceleration & Deceleration Routines

Acceleration and deceleration routines are written using the same logic of the constant speed routines, but varying the delay durations. Delays are determined according to the acceleration and deceleration constants determined experimentaly by running the stepper motors until the desired profile is reached, and these constants are listed in table 4.2.

Acceleration takes place in eleven steps, reaching a speed of 791 steps/sec at the final step giving a profile as shown in figure 4.2.

Address	Constant	Step Rate(steps/sec)
Ø7CØ	3F	238
Ø7C1	3D	245
Ø7C2	3A	260
Ø7C3	34	289
Ø7C4	2B	349
Ø7C5	25	406
Ø7C6	2Ø	470
Ø7C7	1A	578
Ø7C8	17	654
Ø7C9	15	716
Ø7CA	13	791

Table 4.2 Acceleration Constants

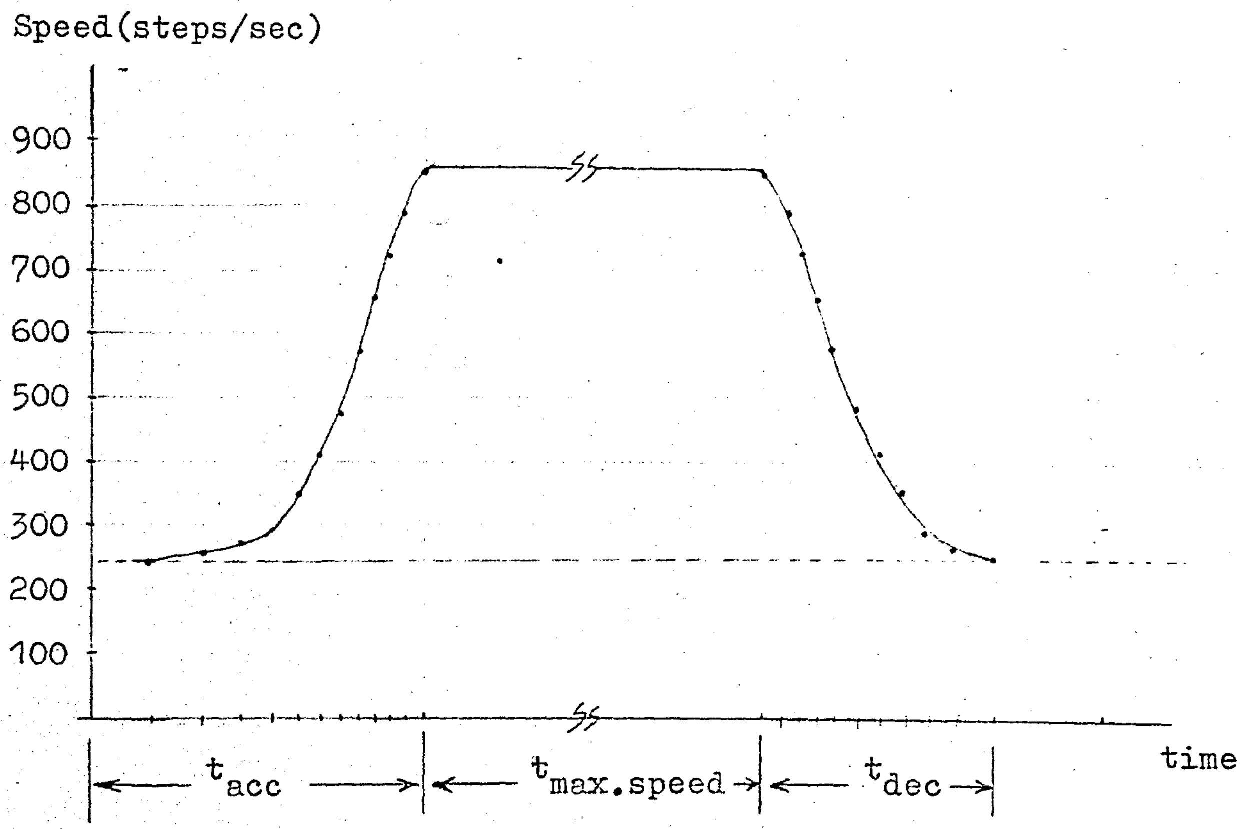


Figure 4.2 Acceleration/Deceleration Profile

Deceleration constants are the same as acceleration constants,first constant being the last one in table 4.2, and they are written into memory locations from Ø7DØ to Ø7D9.

The flowchart of an acceleration routine is given in figure 4.3,on the next page.

In the deceleration case,IY index register is loaded with Ø7DØ which is the starting address of the deceleration constants,and register C is loaded with ØAH because deceleration takes place in ten steps.

After acceleration,motors are fed with a 835 steps per second pulse rate,increasing the speed of the stages to their maximum velocity.

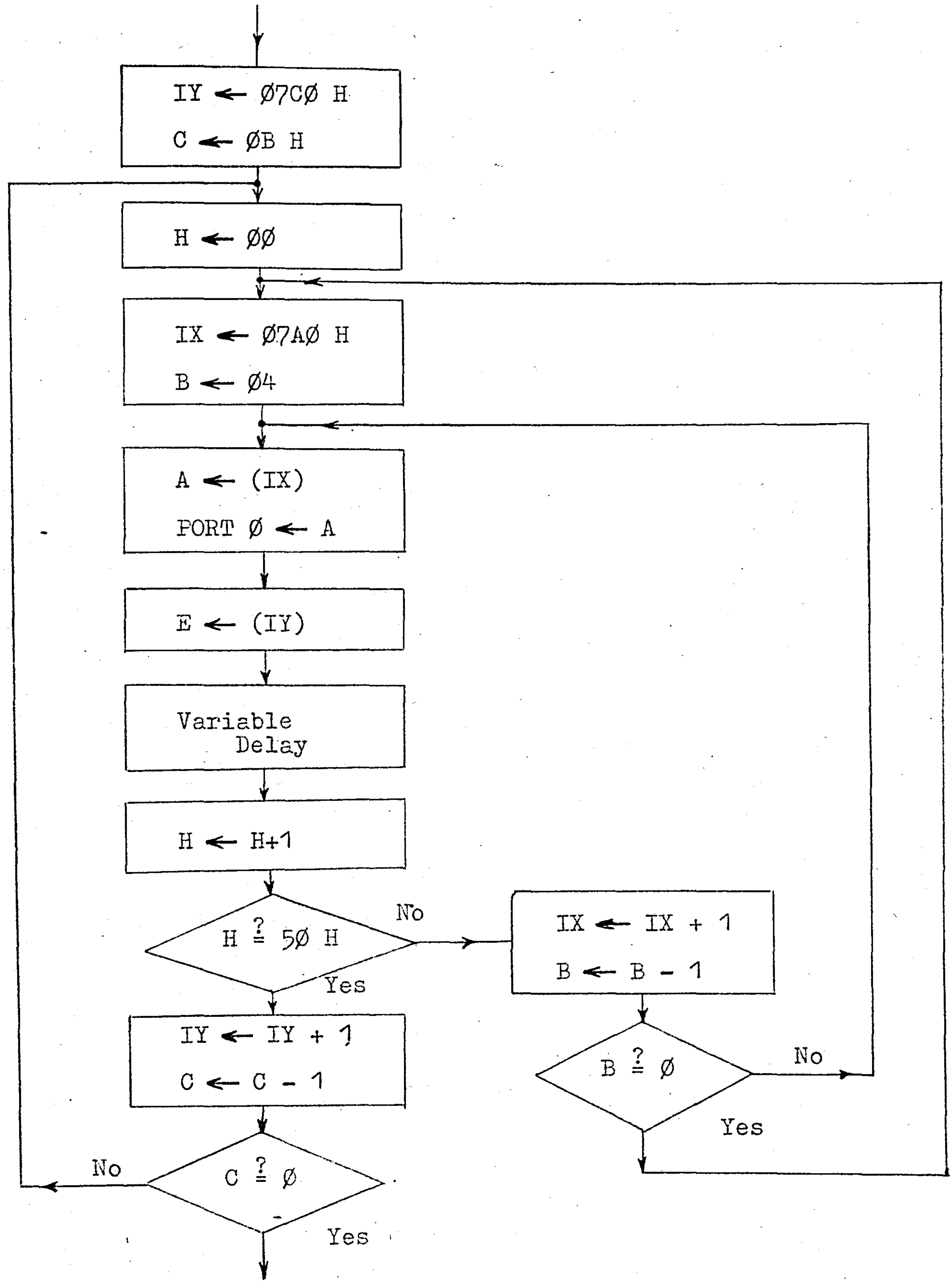


Figure 4.3 Acceleration Routine Flowchart

B. MAIN PROGRAMS

I. Line Detection Program

In general,this program searches the frame lines, and locates the detector on the top right corner of the ring mask.Program operates as follows:

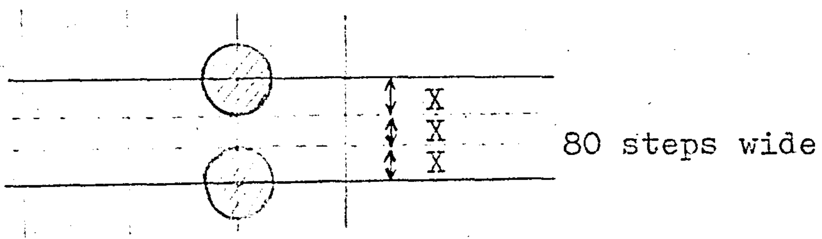
1). Detector moves up (y-stage moves in negative-y direction) in 4-step intervals and inputs data through port 1.

a. If input is zero (that is no hole),program returns to 1),in order to move 4 more steps.

b. If input is 'one' (that is hole-position or frame-line),detector is moved 68 more steps in order to measure the depth of the blackness,taking data at every 4-step.

i. If a zero is detected in one of these 4-step movements,program returns to 1). for further search.

ii. If a zero is not detected (meaning that all of these 4-step movements give high(one)),program decides that it has found the upper frame line of the card.Because the maximum possible diameter of a hole position is 54 steps.



3X = 80 , X = 27

Diameter of a dot = 2X = 54 steps

Figure 4.4 Dot dimensions

2). Then the detector steps back from the upper line by the amount it entered plus one grid length, giving a total of 150 steps.

3). After these operations, detector moves towards right (x-stage moves in negative-x direction, towards motor), inputing data at every 4-step.

a. If a zero is detected, detector continues its movement in the same manner as explained in 3).

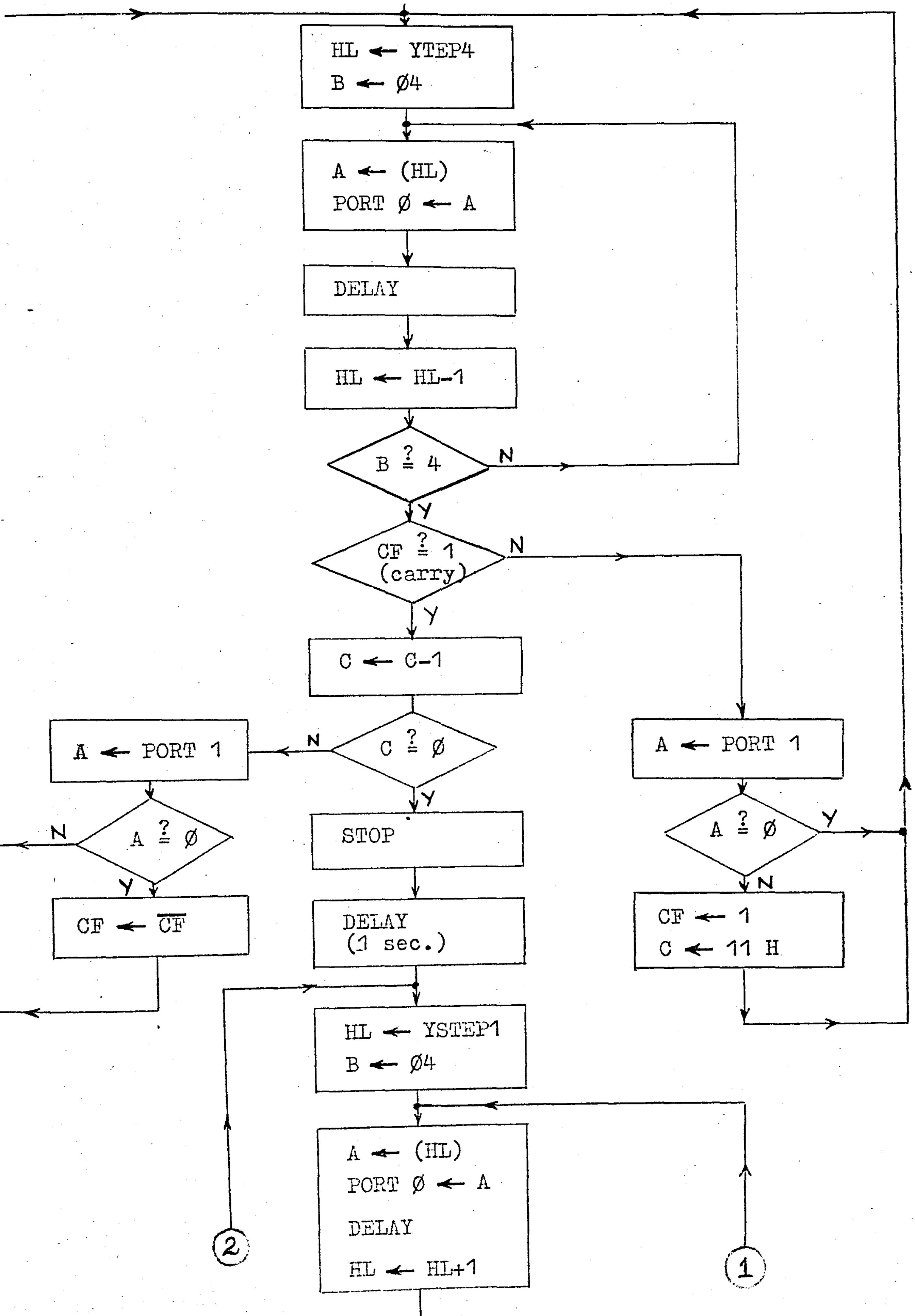
b. If a high level is detected, this is for sure the right side frame line, because as explained in the 'operating the system' section, there should not be any dots, one grid below the upper frame-line.

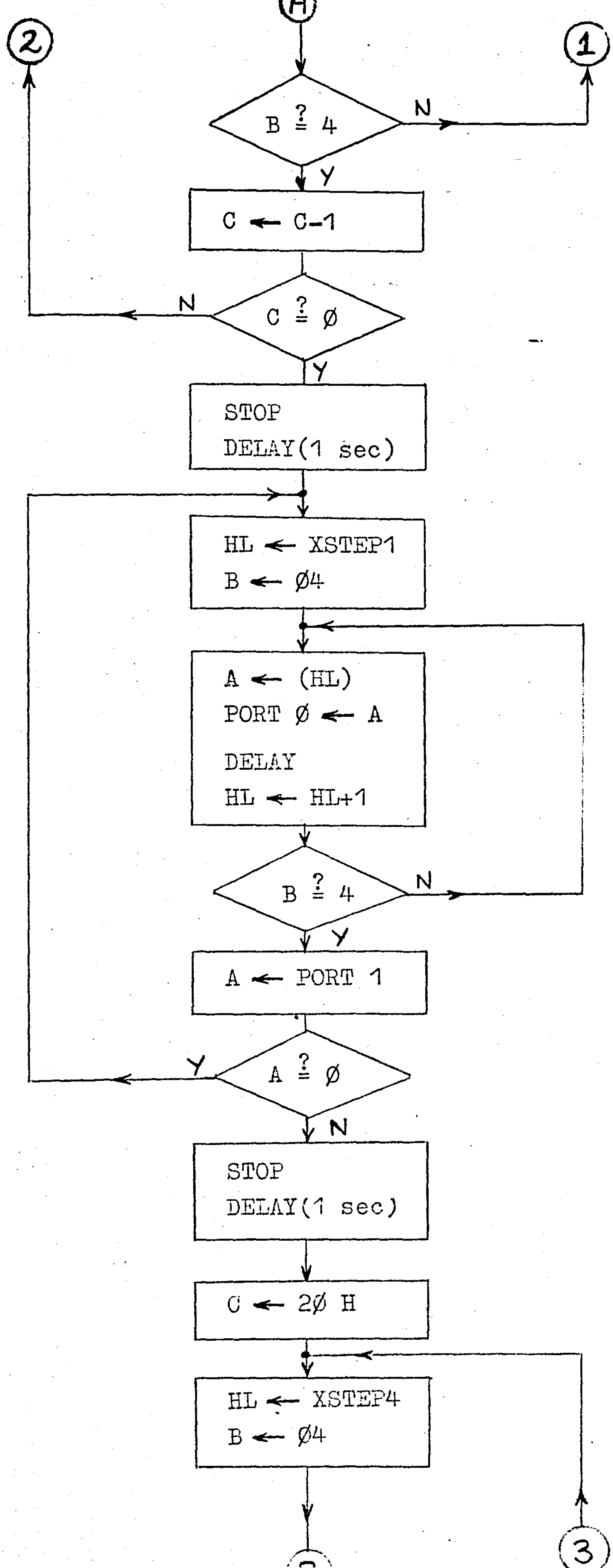
4). Detector then moves one grid length in the opposite direction (towards left), and stops a moment to continue with the next program.

This final location of the detector after this line-detection program, is the zero-position reference point, thereafter all calculations and length measurements are done according to this position.

Flowchart of this program is given in figure 4.5.

Frame Line Detection Program




A
2
1
B ? 4
N
Y
C ← C-1
C ? Ø
N
Y
STOP
DELAY(1 sec)
HL ← XSTEP1
B ← Ø4
A ← (HL)
PORT Ø ← A
DELAY
HL ← HL+1
B ? 4
N
Y
A ← PORT 1
A ? Ø
Y
N
STOP
DELAY(1 sec)
C ← 2Ø H
HL ← XSTEP4
B ← Ø4
3

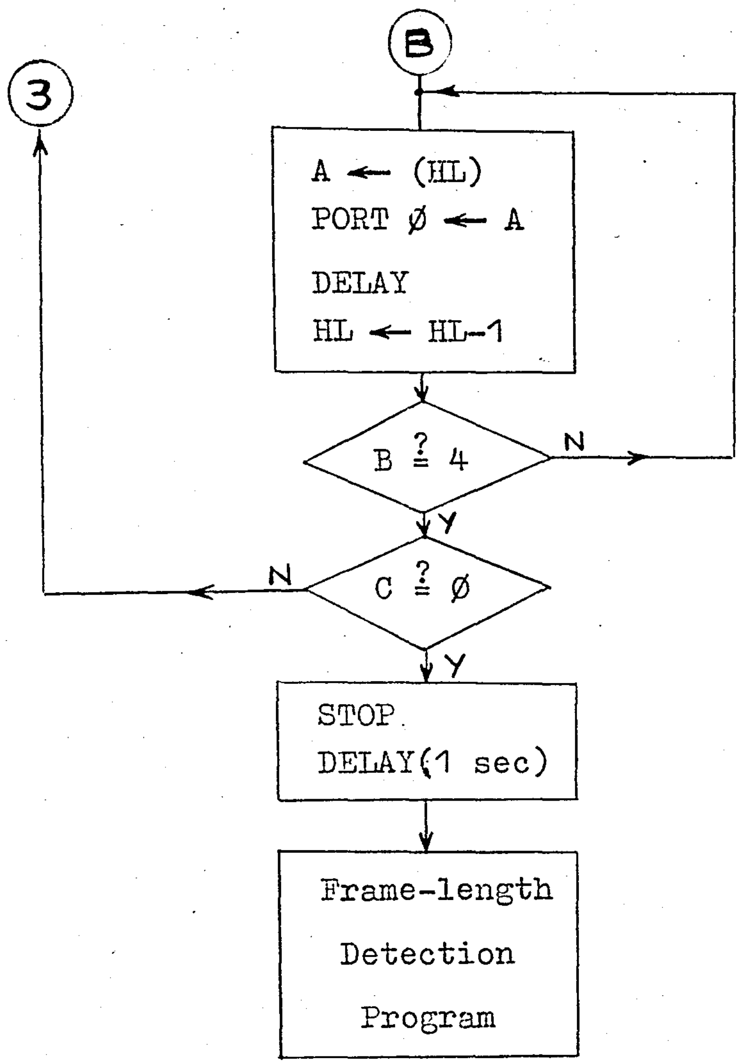


Figure 4.5 Flowchart of Line-Detection program

II. Frame-Length Detection Program

This program measures the x and y lengths of the frame of the card and operates as follows:

1). Detector moves towards left (x-stage moves in positive-x direction, away from the motor) with constant speed of 246 pulses/sec., incrementing the grid counter at every 80 step, until the detector observes the left-side line of the frame.

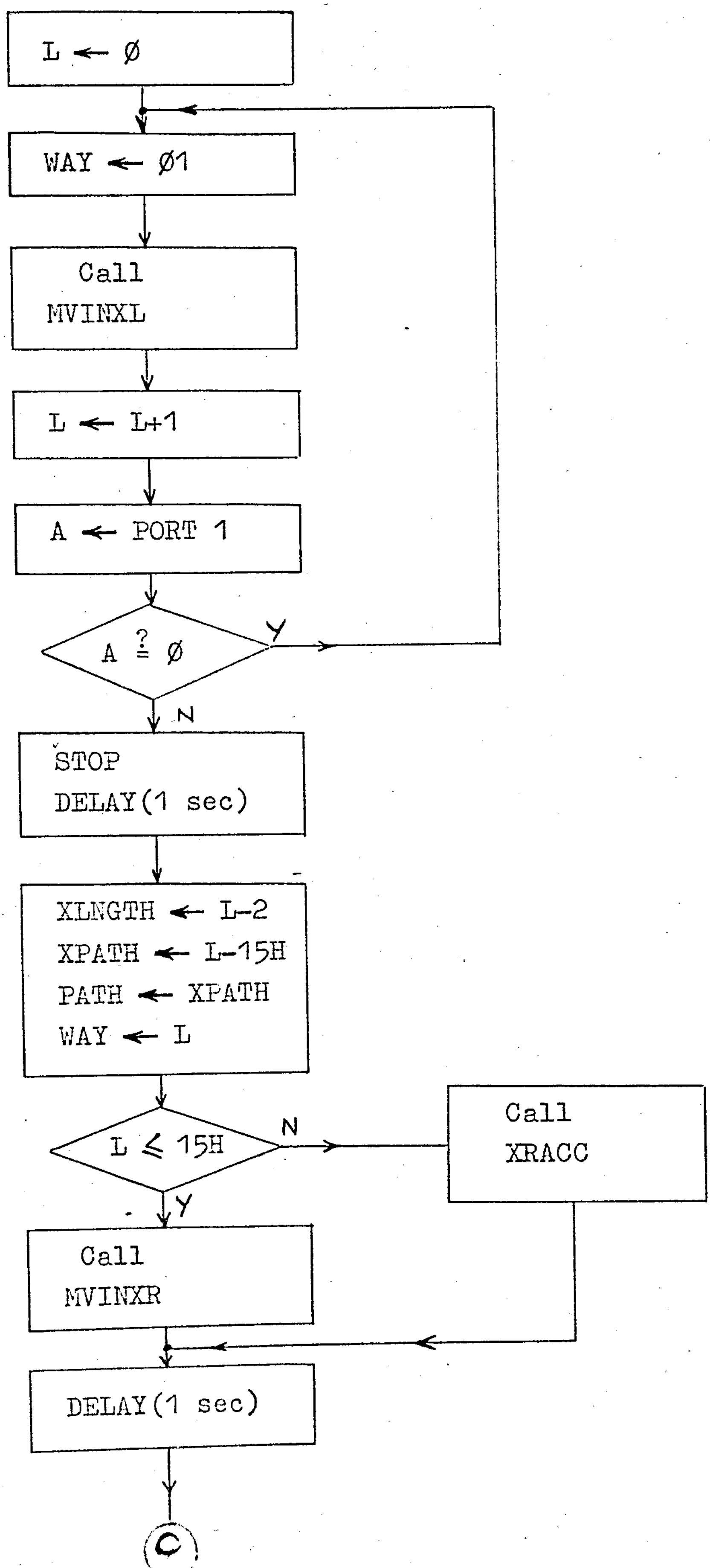
2). Then this length information in its grid counter is stored (by subtracting 2) into the memory location XLNGTH. Two is subtracted because detector could have entered the frame just at the end of the grid length, and the other grid is subtracted to leave one grid empty on every side of the card.

3). Since acceleration and deceleration steps add up to 21, (11 grids for acceleration, 10 for deceleration), this number is compared with the x-length measured by the system. If x-length is greater than 21 grids, system returns to zero position by accelerating to maximum speed. If smaller or equal to 21, then it returns with constant speed.

4). Y-length is also measured in the same way as explained in 1,2,3 above. This time, same control signals and step commands are send to the y-motor. Thus, bottom-stage moves and performs the same procedures. Y-length is stored into YLNGTH, and detector finally returns to zero-position.

Flowchart is given in figure 4.6.

Frame-length Detection Program



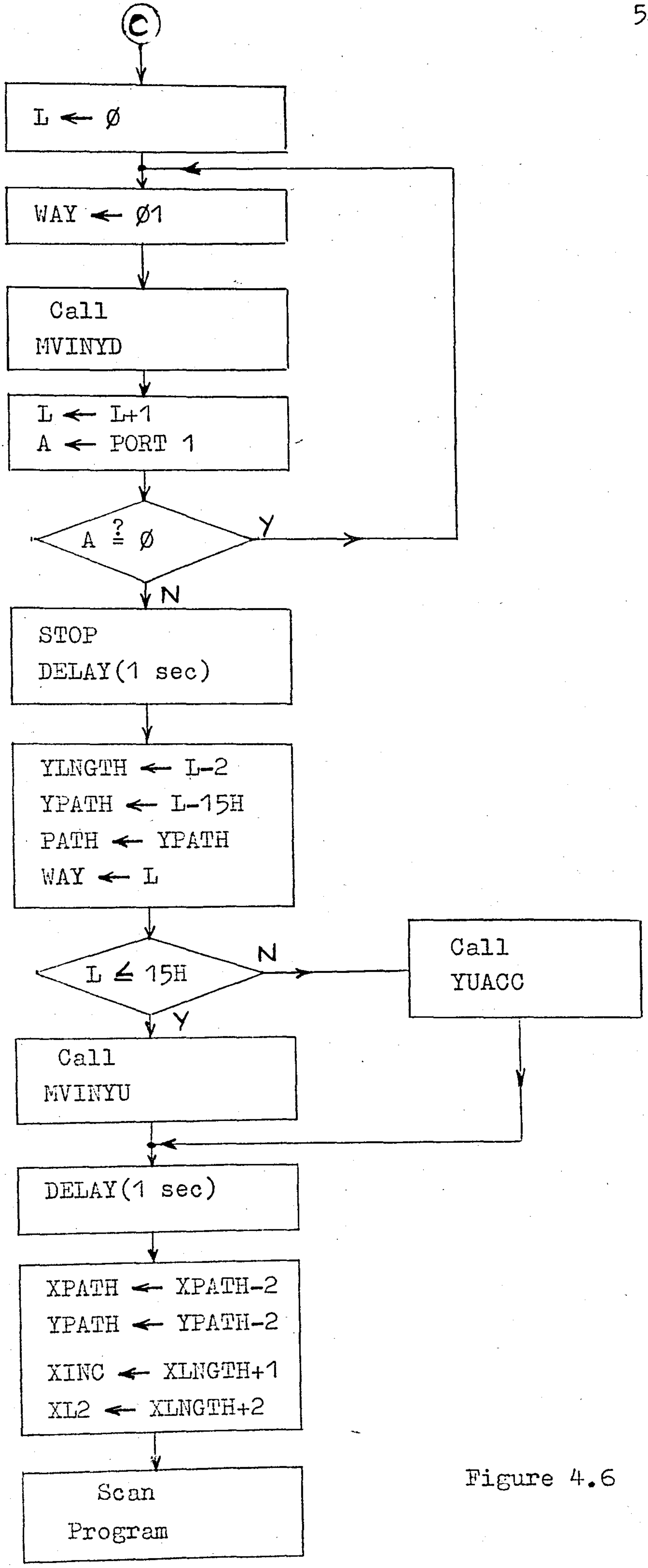

C
L ← Ø
WAY ← Ø1
Call
MVINYD
L ← L+1
A ← PORT 1
A ?= Ø
Y
N
STOP
DELAY(1 sec)
YLNGTH ← L-2
YPATH ← L-15H
PATH ← YPATH
WAY ← L
L ≤ 15H
N
Y
Call
YUACC
Call
MVINYU
DELAY(1 sec)
XPATH ← XPATH-2
YPATH ← YPATH-2
XINC ← XLNGTH+1
XL2 ← XLNGTH+2
Scan
Program

Figure 4.6

III. Scanning and Store Programs.

This program scans the whole card in meander pattern taking data at every 1/20 inch interval(which is the node separation of the grid),and storing the data taken according to a pattern given in the STORE routine for further processing in the DRILLING program.The operation of this program is as follows:

1). Program first determines whether the scanning will take place at constant speed or in the maximum speed mode. Decision is made according to the length in the x-direction(LNGTHX).If x-length is greater than 21 grids (sum of acceleration and deceleration steps),then program enters the maximum speed mode.If not,scanning is done at constant speed (scanning with CSCAN routine).

2). Scanning is done in the following manner when the maximum speed mode is selected due to XLNGTH.

a. Since XLNGTH is known,and sum of acceleration and deceleration steps is 21,length which will be travelled with maximum speed is calculated by subtracting 21 from XLNGTH and storing this value into memory location XPATH.

b. Scanning of the line,is done first by acceleration and taking data at every node by calling the STORE routine.Then stage reaches its maximum speed,travels at this speed by the length in XPATH (again taking data at each node by STORE routine)and decelerates in ten steps and stops at the end of the line.

Odd numbered lines are scanned from right to left,while the even numbered lines are scanned from left to right in the same format explained in 2.b.

c. At the end of every line, LASTST (Last Store) routine is called.Since data (hole-position information) are stored in 8-bit blocks into the memory,even though a byte is not completely filled at the end of a line,it is stored half-full ,to prepare the store routine for the new coming data belonging to the next line to be scanned.

3). After the above mentioned steps,the y-stage moves down one grid,and the y-length is decremented by one.

a. If y-length is not covered totally,that is if the scanning of the card is not finished,scanning procedure continues as explained in 2a,b.

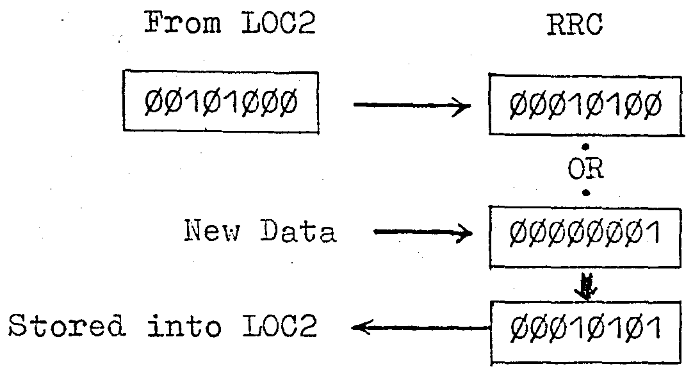
b. If the scanning of the card is finished at the end of an odd-numbered line,detector is returned to zero-position by a routine called SRTNZ1 which first makes the x-stage move by XLNGTH long (with or without acceleration depending on the x-length),and then the y-stage,YLNGTH long.

c. If the scanning is finished at the end of an even numbered line,program jumps to SRTNZ2 which is a part of the program SRTNZ1,to move the y-stage until the detector is again placed at the top right corner (zero position) of the card.(Returning to zero position is necessary for the drilling process).

4). Sub-routine STORE is the main part of the scanning and detection program,whose operation is as follows:

a.After exchanging registers,accumulator and flags currently used with the other register set,routine inputs data from port 1(which is the input port).

b. Then retrieves the previous hole-position information byte from memory location LOC2,rotates right and performs logical OR operation on this byte with the new input byte as shown below.



c. LOC1 holds the bit count.In the beginning,it contains 8,but decremented after each above mentioned operations are performed.When LOC1 is decremented to zero,meaning that 8 consecutive hole-position information is read,and one byte is filled.Then routine jumps to STM (Store to memory) routine within the STORE program.

d. STM routine retrieves the data from LOC2, rotate right one more time,and store this form into the memory location ,in which STAM (Starting Address of Memory) contains.Initially,STAM contains 8ØØØH,which is the first RAM location in the system.After each byte-store,STAM content is incremented by one,LOC2 is cleared,and LOC1 is loaded with Ø8,in order to process the new coming data.

e. LASTST routine (Last Store),within the routine STORE,stores whatever data are in LOC2,after rotating the content LOC1 times in order to make this data fit to the reading sequence.

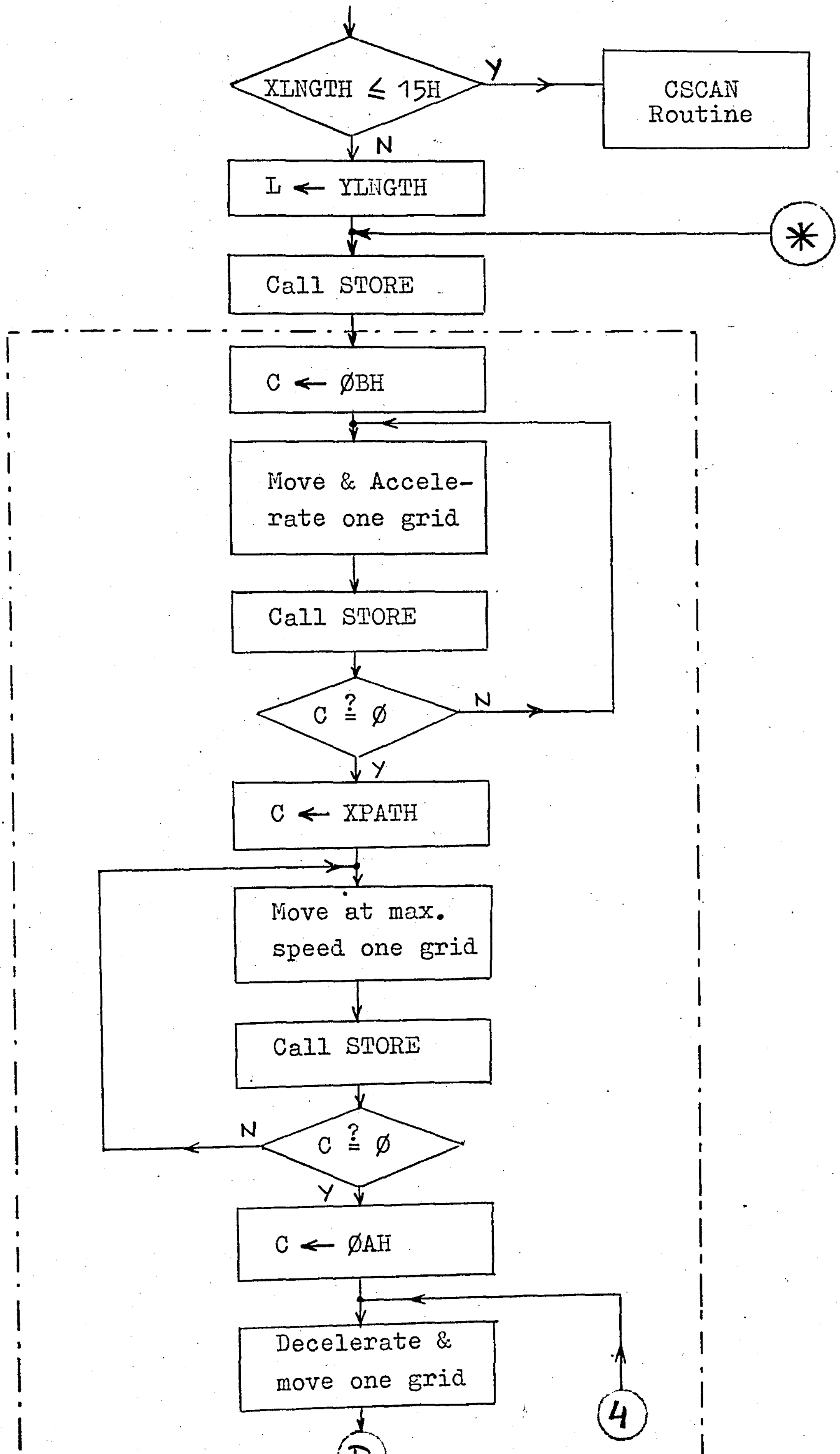
STORE routine takes 118 T cycles (59 micro-sec.) if a store to memory is not performed, and 233 T cycles, if performed (116.5 micro-sec.). This means that the routine is fast enough to function properly even when motors are running at their maximum speed without imposing any constraints to speed.

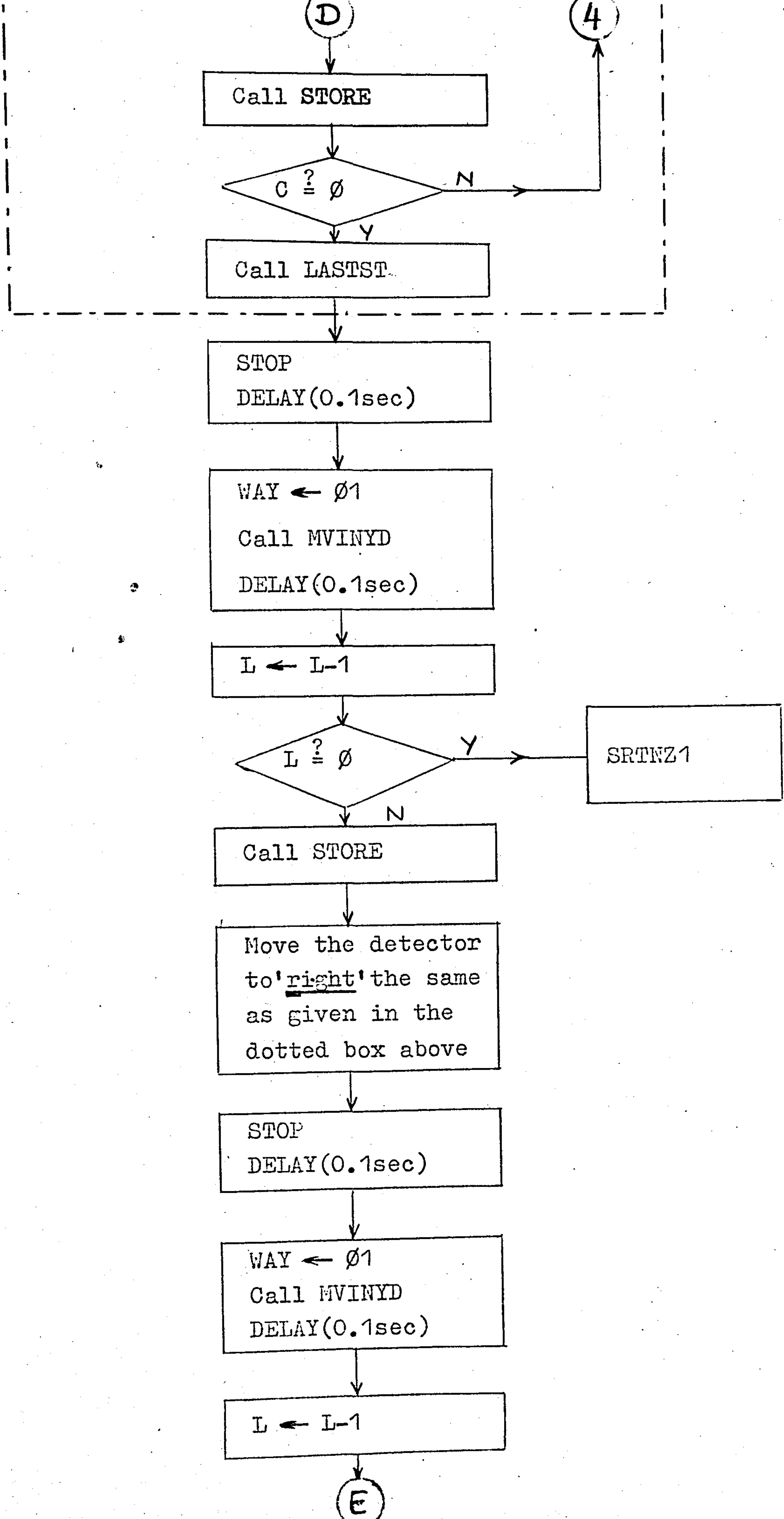
5). Constant speed scanning routine CSCAN uses MVINXL, MVINXR and MVINYD which are constant speed routines and the scanning of the card is the same as in the maximum speed case, also utilizing STORE routine for hole-position detection.

Flowcharts of these routines are given in figures 4.7, 4.8, 4.9.

In figure 4.11, detector movement in each program is given schematically.

Scanning Program




D
4
Call STORE
C ? ø
N
Y
Call LASTST
STOP
DELAY(0.1sec)
WAY ← ø1
Call MVINYD
DELAY(0.1sec)
L ← L-1
L ? ø
Y
SRTNZ1
N
Call STORE
Move the detector to 'right' the same as given in the dotted box above
STOP
DELAY(0.1sec)
WAY ← ø1
Call MVINYD
DELAY(0.1sec)
L ← L-1
E

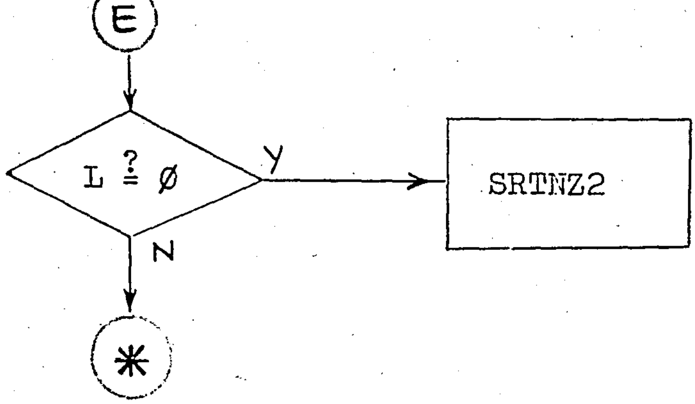
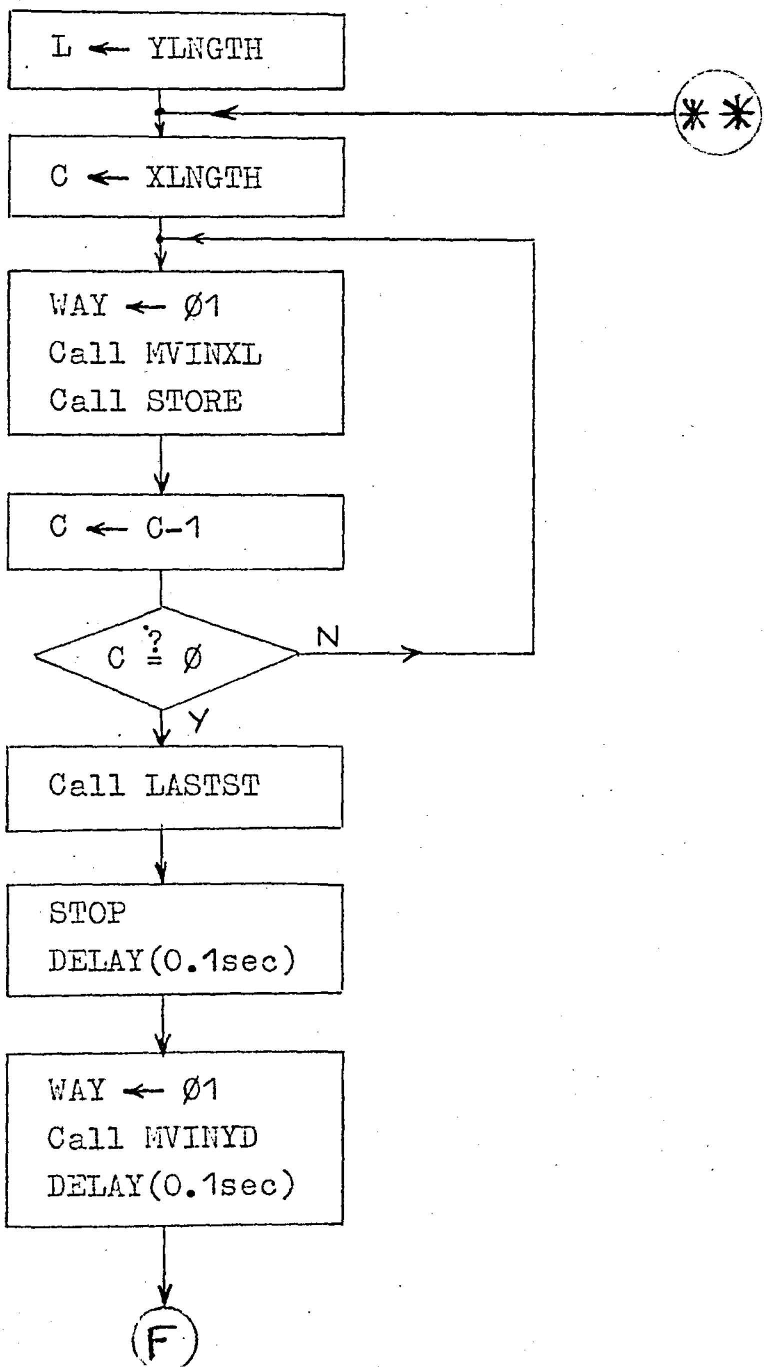


Figure 4.7 Scanning & Detection Program

CSCAN (Constant Speed Scanning) Routine



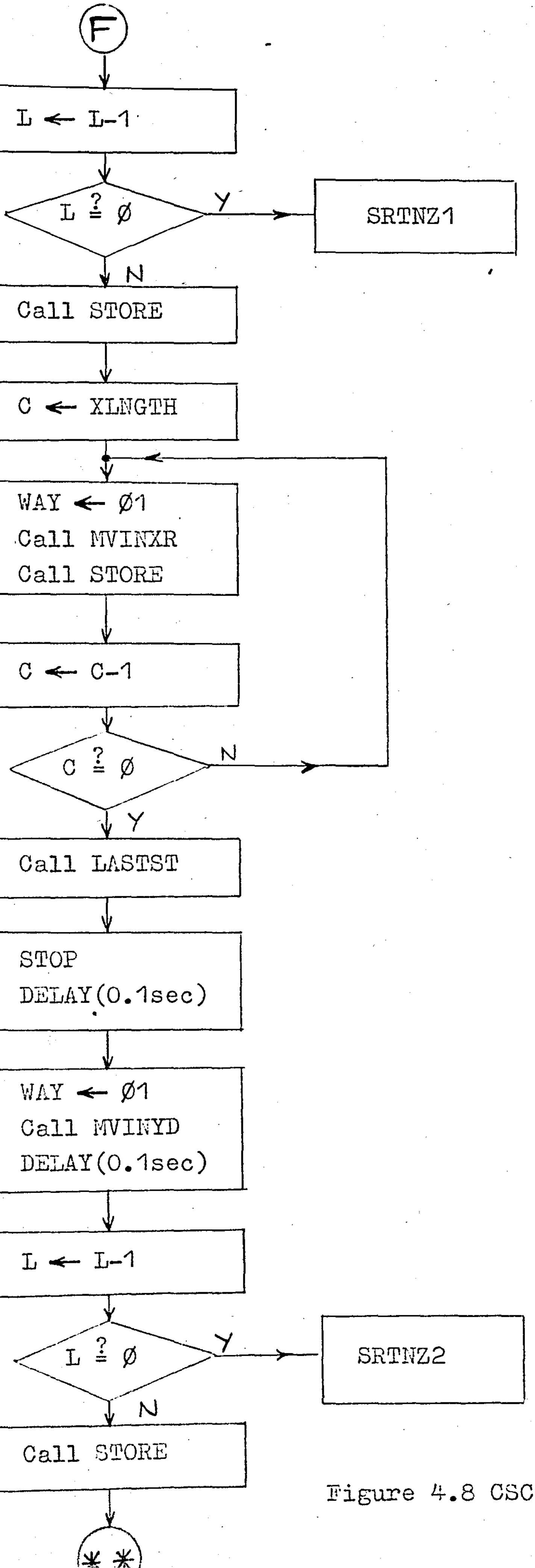


Figure 4.8 CSCAN Routin

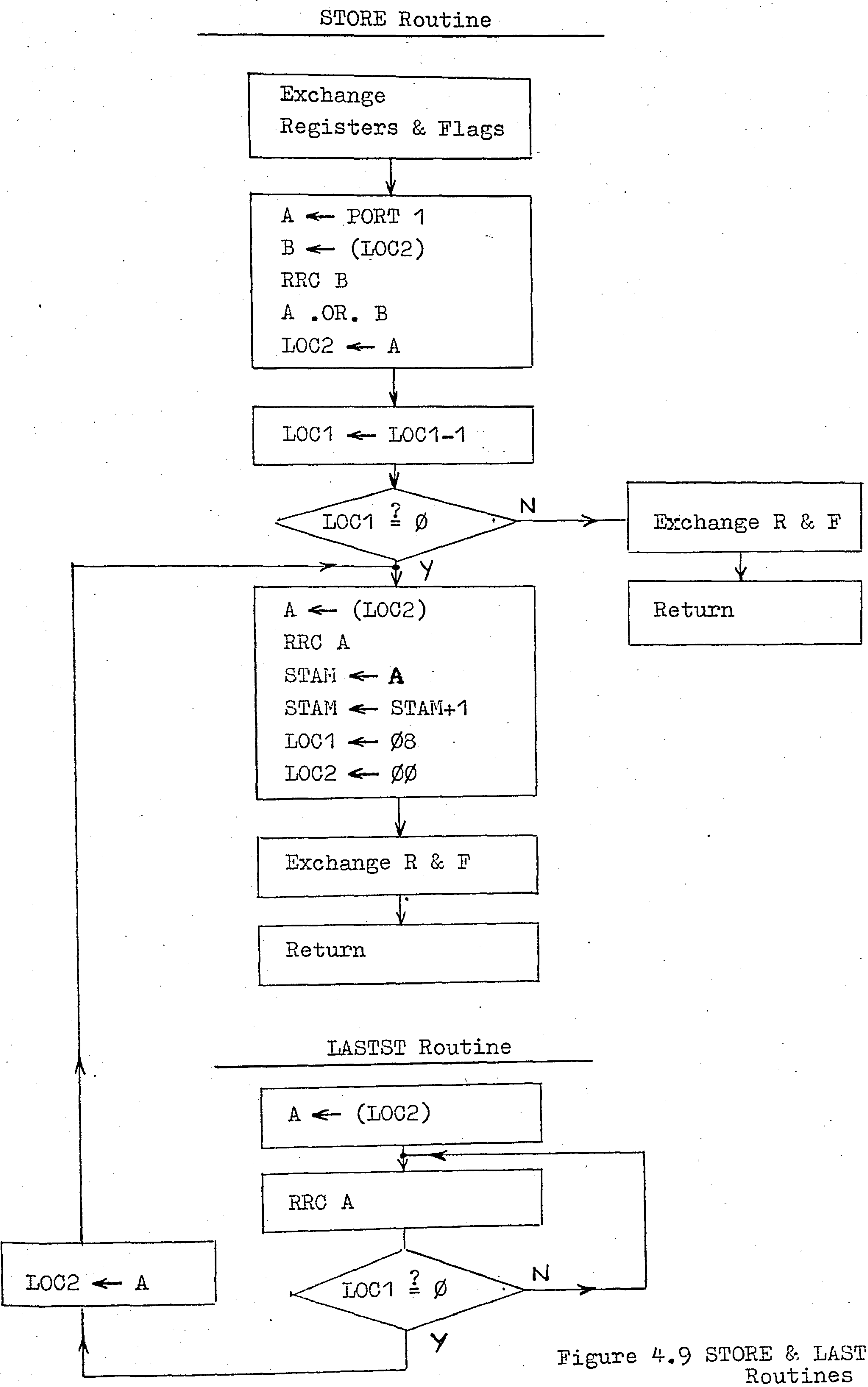


Figure 4.9 STORE & LASTST Routines

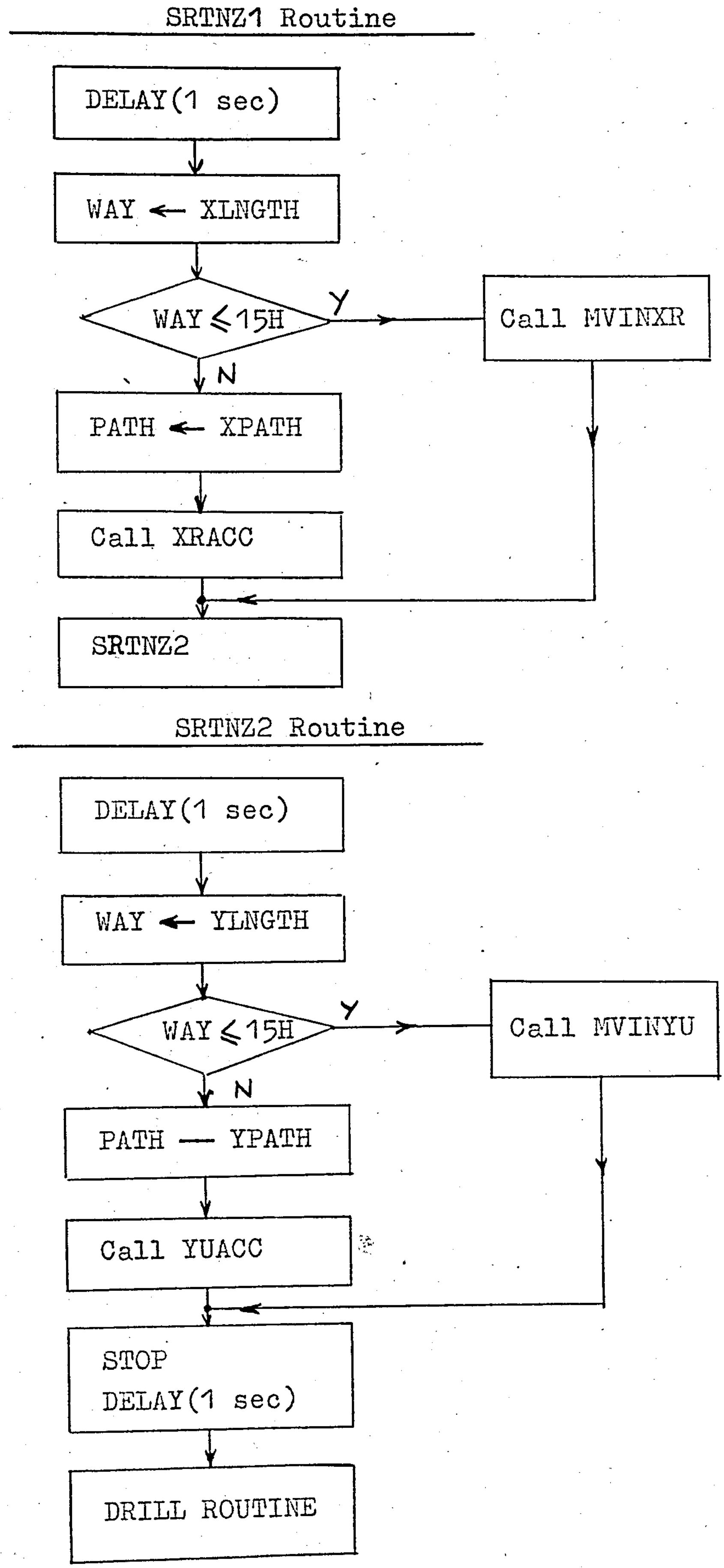
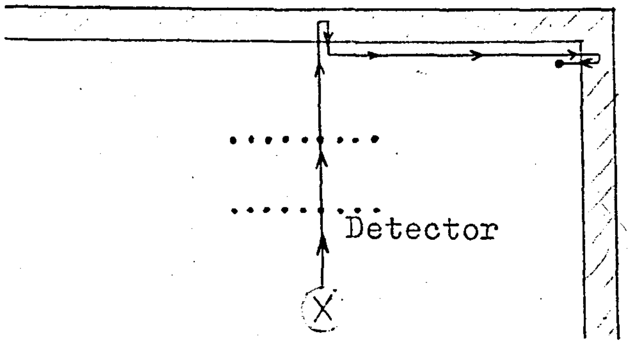
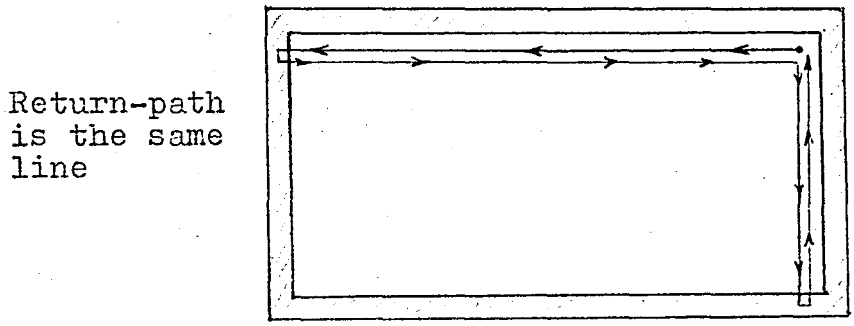


Figure 4.10 SRTNZ1 & SRTNZ2 Routines

I. Line Detection Program



II. Frame-length Measurement Program



III. Scanning Program (meander pattern)

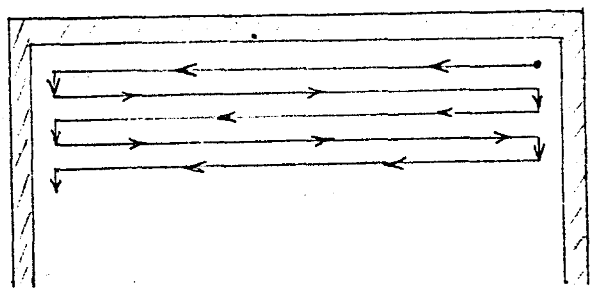


Figure 4.11 Detector Movements in each program

IV. DRILLING Program

This program retrieves the hole-position information from the memory and performs the drilling process according to these data using an optimum-path algoritm, which operates as follows:

1). Routine first clears certain areas of RAM which will be used during the program, and does some initialization by loading data to appropriate registers. Since the scanning of the card is done first from right to left, then from left to right, the drilling program also, operates in a different fashion when lines are odd-numbered or even. The data of an odd-numbered line are processed as described.

a. One byte of information is taken from memory and bit by bit a '1' is looked for. At every test, incremented x-length value is decremented (which is in fact the node number on this line). Zeros do not cause branching, but when a '1' is encountered, its location on that line is written in to the memory starting from FRSTIX (873ØH). By not considering zeroes in a byte, routine only records the locations of '1's till the line ends.

For example, if the x-length of the card is 100, this means there are 100 grids, but 101 nodes. There is the possibility of having 101 hole-positions on this line. This number is decremented every time a bit in a byte is tested. For instance, when a '1' is found in a byte and the register contains 69. This '1' is said to be at node 69, beginning being the 101st node whose position is at the rightmost side.

b. After this search till the end of the line, memory location of the last stored hole-position is put into LASTIX, and the address of next byte to be tested in the second line is saved in SAVEHL.

c. Then a test is done whether there exists a hole or not on this specific line. If there is not a hole, one grid length is moved down in the y-direction and the routine jumps to FTR, to trace the second line.

i. If there is a hole, it is tested whether there is only one hole position. If it is so, hole location is compared with the OLDX (which is the old location of the motor in the previous line. In the beginning, motor position is the XINC, incremented x-length value). Comparison decides whether hole is at the left or at the right of the old motor position. If they happen to be at the same location, drilling is made, (in fact simulated by turning on a LED for a few seconds). If the hole is at the right side or at the left side of the motor, the distance to be travelled is calculated and moved to the hole location with or without acceleration depending on the distance between the hole position and OLDX.

ii. If there is not only one hole on this specific line, a test is made whether the first hole is at the right of the old position of the motor. If it is not, meaning that it is at its left, motor moves left to this first hole-position and performs the drilling from this point onwards from right to left. If the first hole is at the right of OLDX, a comparison is made between the distance of the first hole and OLDX, and the distance between last hole and OLDX. If first hole is nearer, motor moves towards right on to the first hole and starts drilling from here (from right to left). If the last hole-position is nearer,motor moves towards left onto the last hole to perform the drilling from here (from left to right).

After the hole-positions are processed as explained and drilling is completed on this line,stage moves one grid in y-direction,and routine jumps to the second line trace section of the program.All of the above mentioned procedures are true only for odd numbered lines due to the fact that scanning and detection of hole positions are done in the same format and direction.

2). FTRACE routine searches the memory as in the STR routine,but it is for even numbered lines and hole locations are calculated differently.

a. When a '1' is found in a bit test,decremented value of XINC is subtracted from XL2 which is two-incremented x-length in order to make the grid node numbering system similar to the odd-lines.Because in the even-numbered line case,the first data are at the left side of the card whereas in the odd-lines,first data are at the rightmost side of the card.

b. After necessary store procedures are completed as in 1.b,a test is done whether there exists a hole or not on this line.If not,one grid is moved in y-direction,and control is passed to STRACE to trace the next line.

i. If there is only one hole,its location is compared with OLDX which determines whether hole is at the left or at the right of OLDX. And according to the direction decided by the comparison,motor is moved and the drilling is performed.

ii. If there are more than one hole,a test is made whether the first hole is on the left of OLDX.If not, motor moves towards right to drill this first hole,and continue with others from here on (from left to right).If there exists a hole at the left of OLDX,a comparison is made between the distance of first hole location and OLDX,with the distance of the last hole location and OLDX.

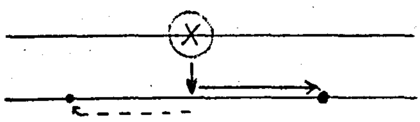
If the first hole is nearer,motor moves towards left to drill the first one and continue with others from left to right.If the last hole is nearer,it moves to right to drill the last hole,and drill the others from here to left.

After the drilling of all the holes on this line is completed,stage moves in y-direction one grid down, and checks whether the card is finished or not.If not,program jumps to the other line in order to continue the drilling. If the card is finished,detector (now being the drill), returns to zero-position and enters in a HALT state waiting for the command to drill the other card.

As it can be seen from the above explained drilling scheme,program utilizes an optimum-path algorithm which reduces the scanning time considerably during drilling.

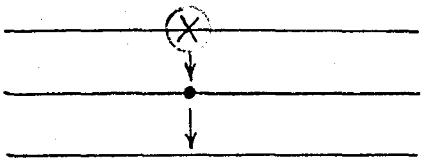
The schematic explanation of how drilling isperformed and the flowcharts are given in the following figures.

Drilling of holes on odd-numbered lines

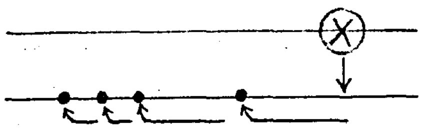


----Last line drilled

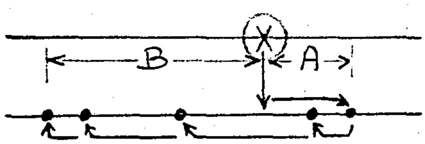
----Next line to be drilled(odd-no)



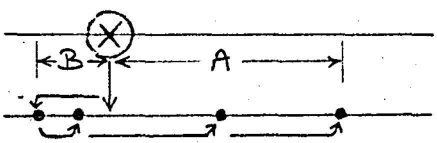
a). If there is only one hole to be drilled,drill moves directly towards it according to its location(right or left).If it is just on it,it drills and moves to the other line.



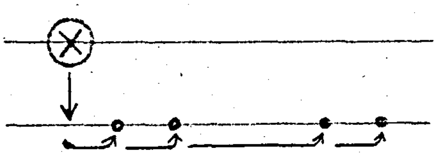
b). It searches right side,if no hole,it moves directly tothe nearest hole and continues.



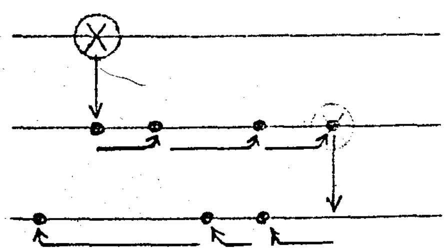
c). If there is a hole on the right, distances A and B are compared. If A is smaller,it goes to right first,then continues to left.



d). If B is smaller,it goes to left first,then drills towards right.



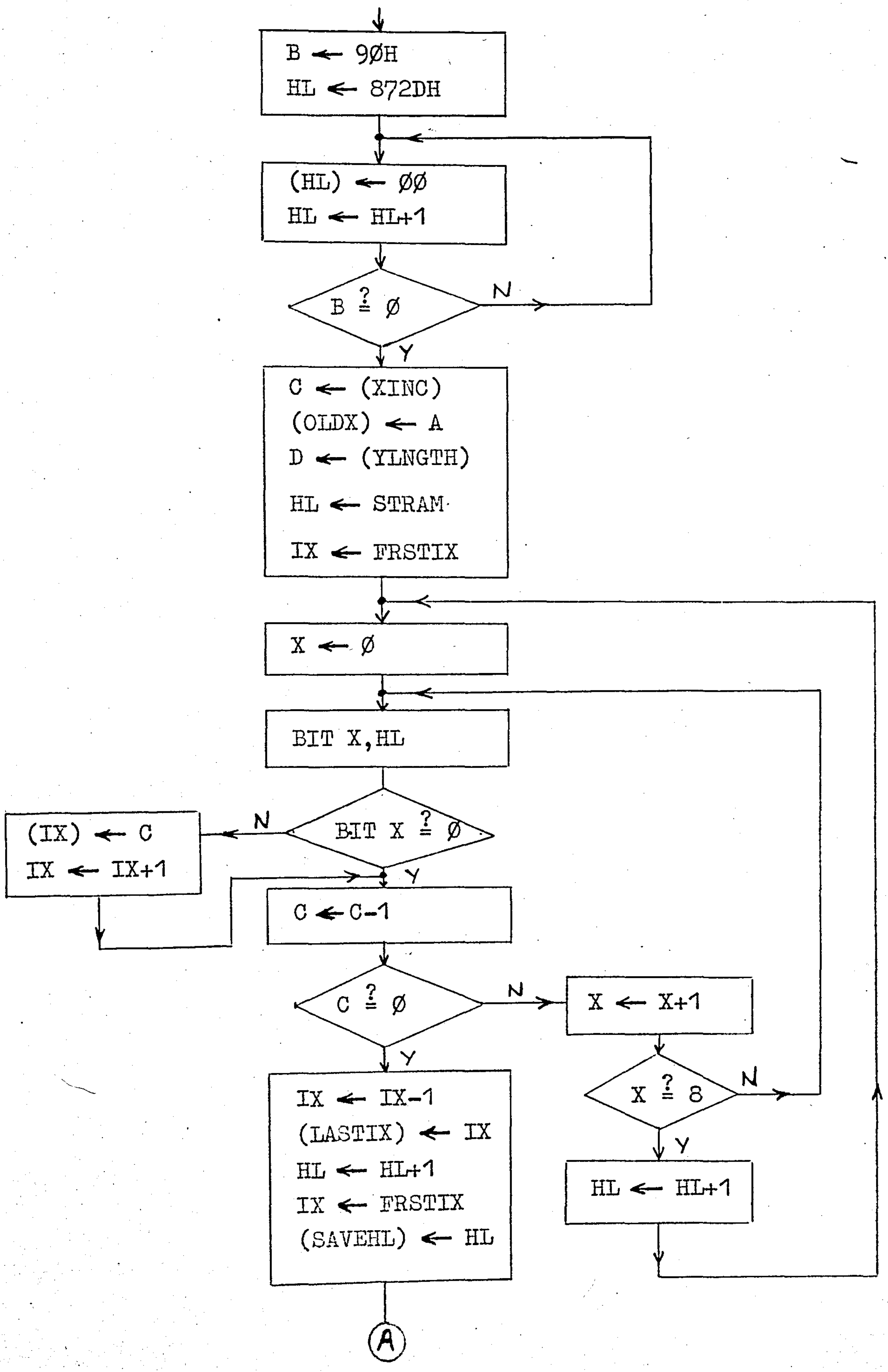
e). If all of the holes are on the right,it goes to the nearest one and drills towards right.

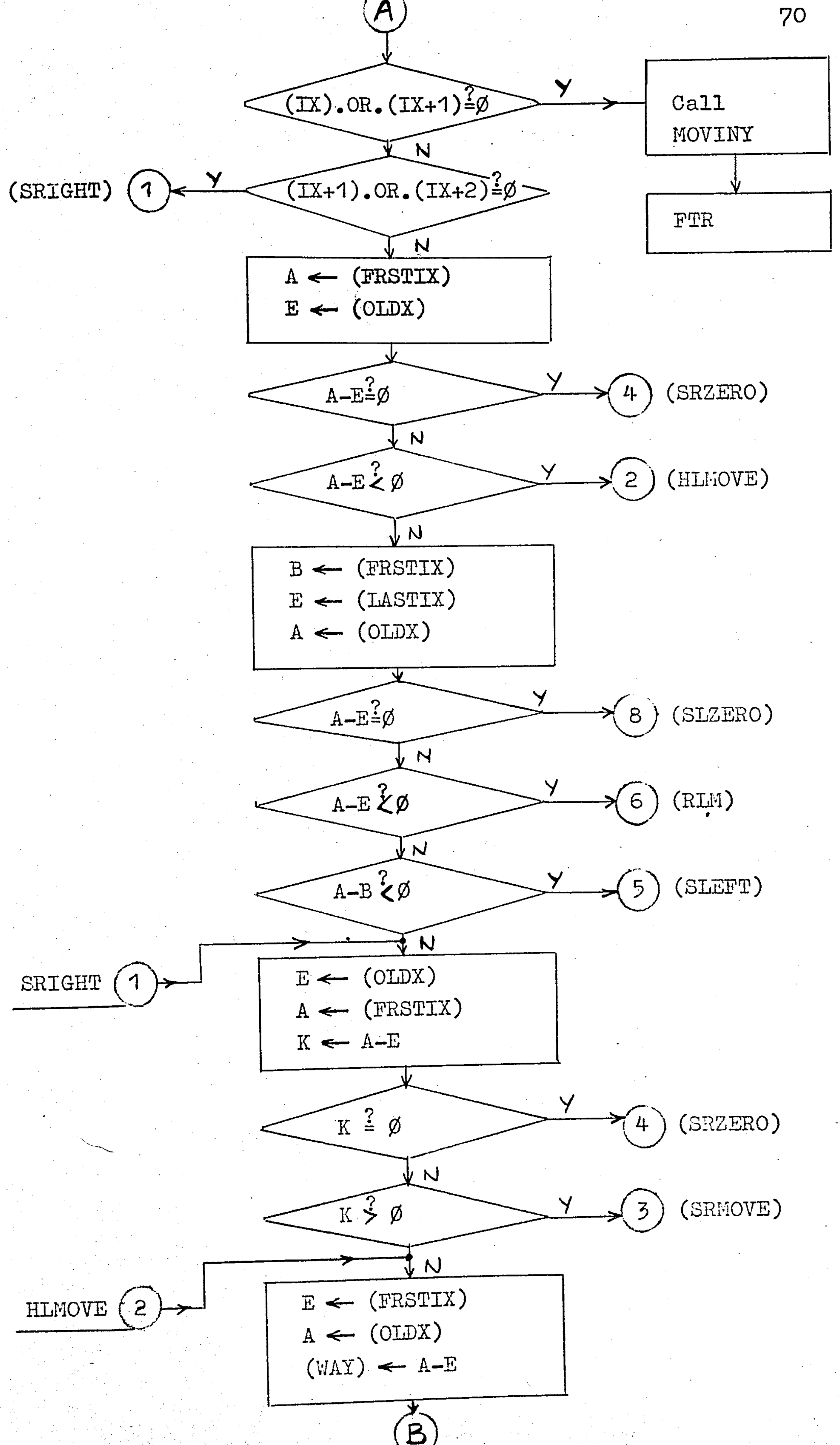


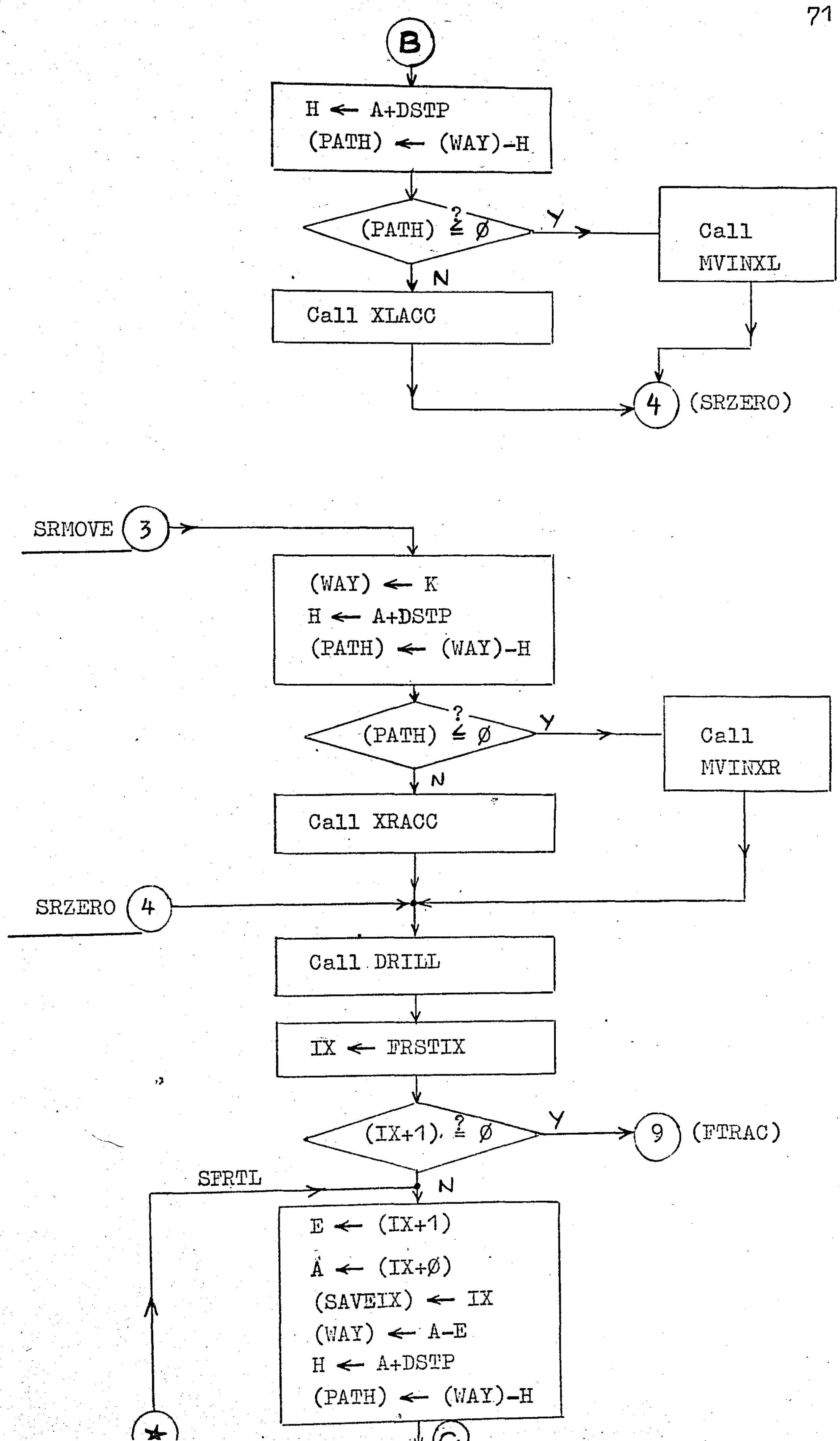
f). If it happens to be on a hole, and no holes on its left(or righ it drills this one,and continues drilling towards right (left)

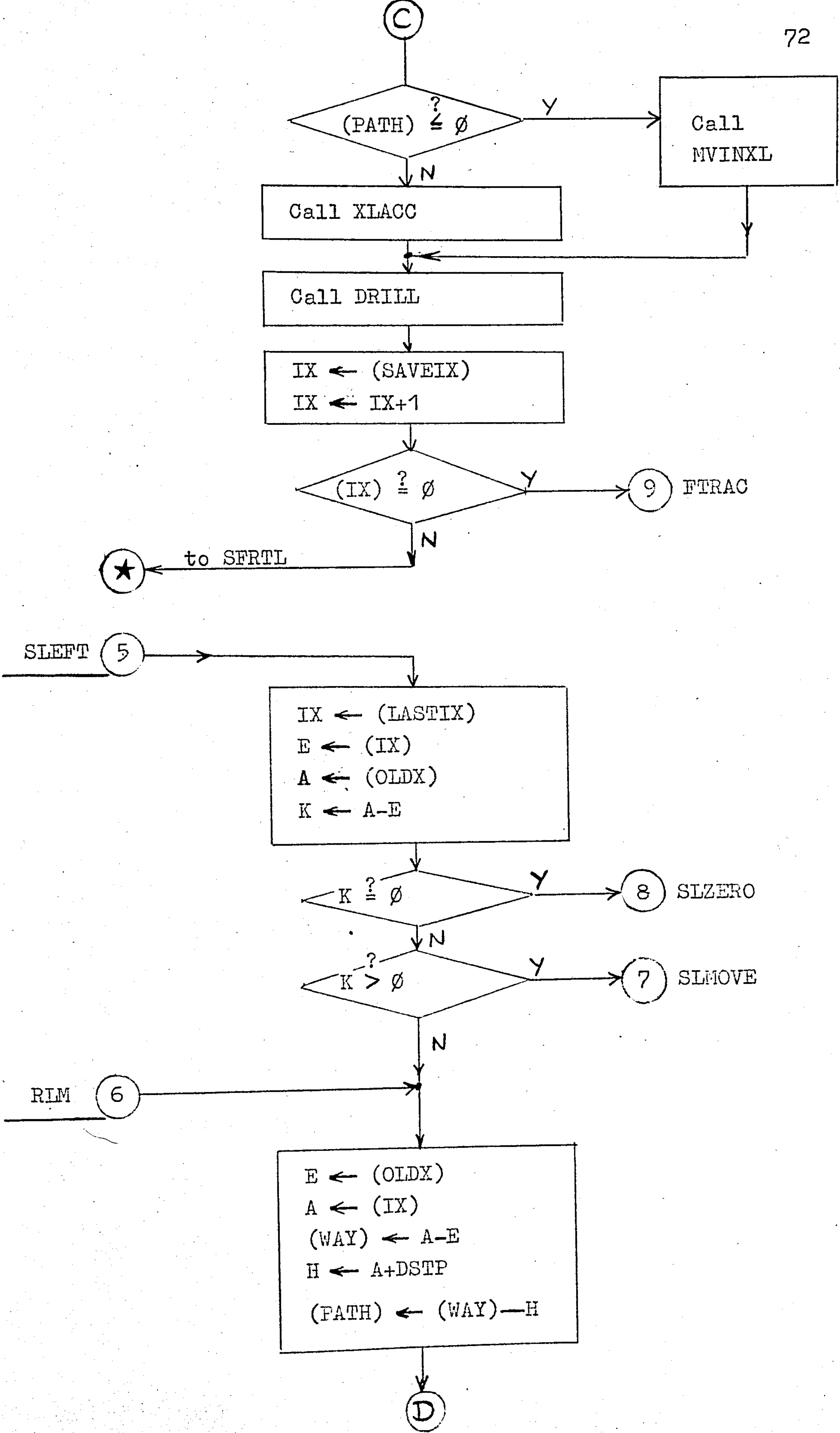
Drilling is the same for even-lines,but it first looks on its left side due to the scanning format.

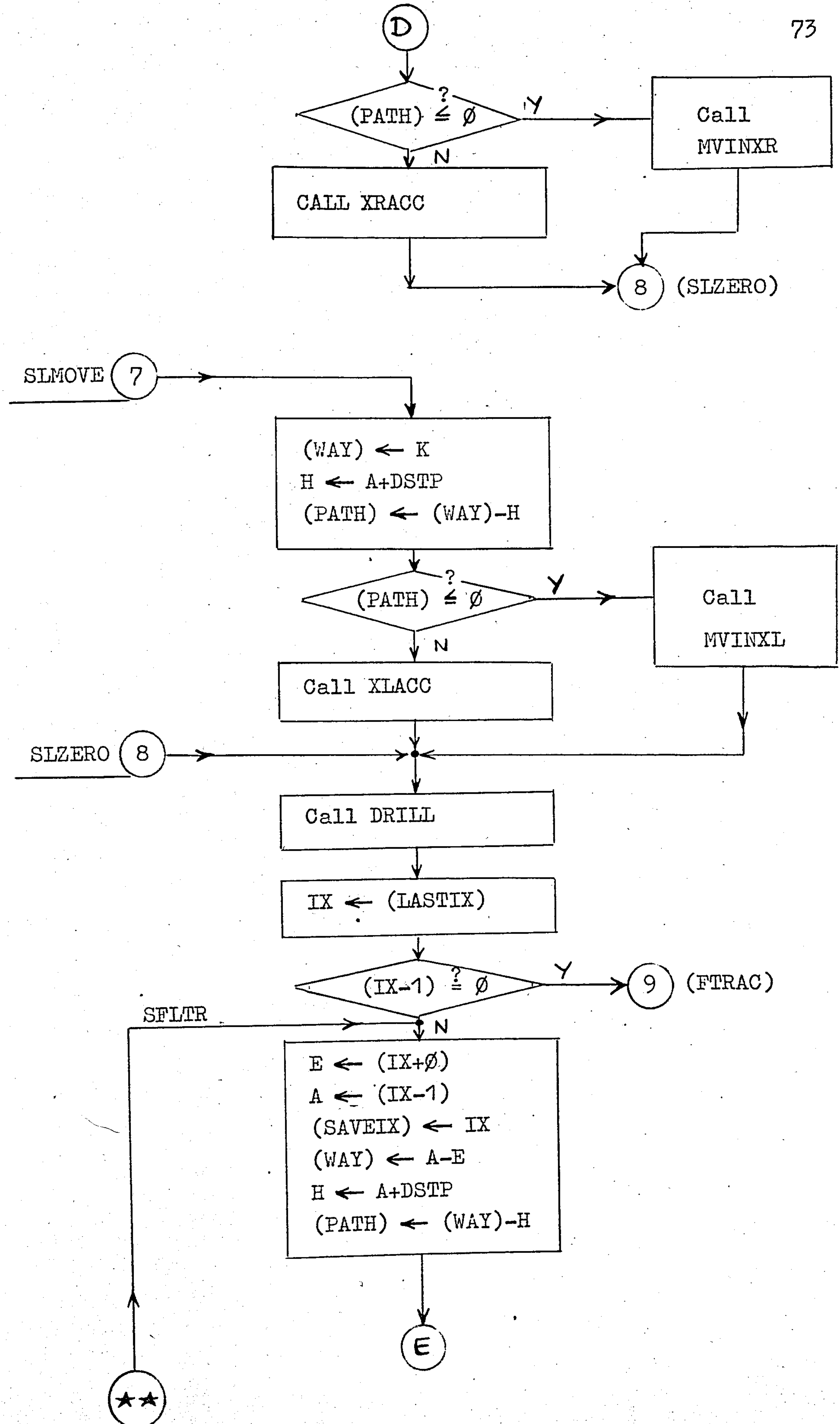
Drilling Program

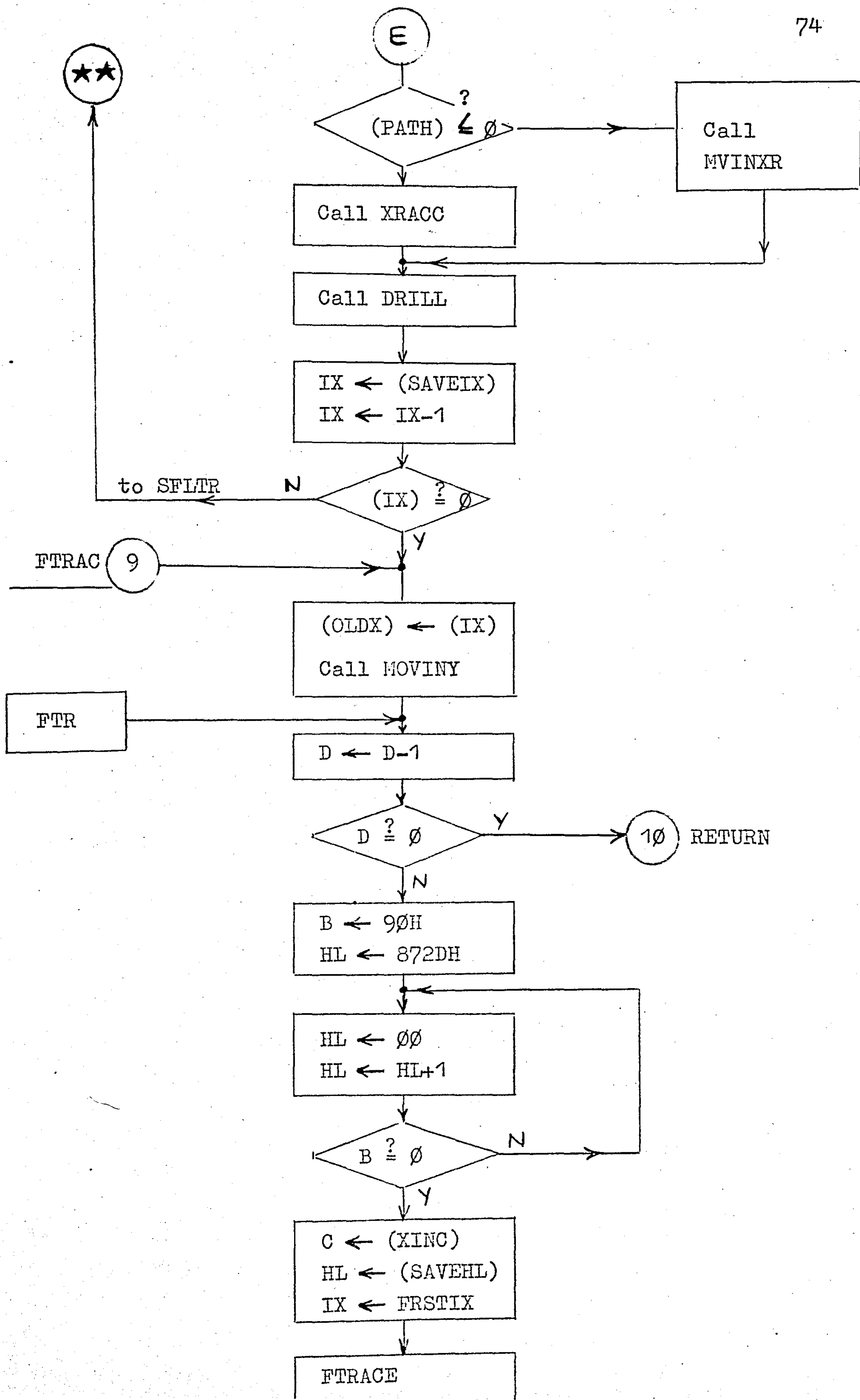



A
(IX).OR.(IX+1)?=Ø
Y
N
Call MOVINY
FTR
(SRIGHT)
1
(IX+1).OR.(IX+2)?=Ø
A ← (FRSTIX)
E ← (OLDX)
A−E?=Ø
4
(SRZERO)
A−E?<Ø
2
(HLMOVE)
B ← (FRSTIX)
E ← (LASTIX)
A ← (OLDX)
A−E?=Ø
8
(SLZERO)
A−E?<Ø
6
(RLM)
A−B?<Ø
5
(SLEFT)
SRIGHT
E ← (OLDX)
A ← (FRSTIX)
K ← A−E
K ?= Ø
(SRZERO)
K ?> Ø
3
(SRMOVE)
HLMOVE
E ← (FRSTIX)
A ← (OLDX)
(WAY) ← A−E
B


B
H ← A+DSTP
(PATH) ← (WAY)-H
(PATH) ≦? Ø
Y
N
Call MVINXL
Call XLACC
4 (SRZERO)
SRMOVE 3
(WAY) ← K
H ← A+DSTP
(PATH) ← (WAY)-H
(PATH) ≦? Ø
Y
N
Call MVINXR
Call XRACC
SRZERO 4
Call DRILL
IX ← FRSTIX
(IX+1) ≦? Ø
Y
9 (FTRAC)
SFRTL
N
E ← (IX+1)
A ← (IX+Ø)
(SAVEIX) ← IX
(WAY) ← A-E
H ← A+DSTP
(PATH) ← (WAY)-H
C


C
(PATH) ? ≤ Ø
Y
N
Call MVINXL
Call XLACC
Call DRILL
IX ← (SAVEIX)
IX ← IX+1
(IX) ? = Ø
Y
9 FTRAC
N
to SFRTL
★
SLEFT 5
IX ← (LASTIX)
E ← (IX)
A ← (OLDX)
K ← A-E
K ? = Ø
Y
8 SLZERO
N
K ? > Ø
Y
7 SLMOVE
N
RIM 6
E ← (OLDX)
A ← (IX)
(WAY) ← A-E
H ← A+DSTP
(PATH) ← (WAY)—H
D


D
(PATH) ? ≤ Ø
Y
N
Call MVINXR
CALL XRACC
8 (SLZERO)
SLMOVE 7
(WAY) ← K
H ← A+DSTP
(PATH) ← (WAY)-H
(PATH) ? ≤ Ø
Y
N
Call MVINXL
Call XLACC
SLZERO 8
Call DRILL
IX ← (LASTIX)
(IX-1) ? ≤ Ø
Y
9 (FTRAC)
SFLTR
N
E ← (IX+Ø)
A ← (IX-1)
(SAVEIX) ← IX
(WAY) ← A-E
H ← A+DSTP
(PATH) ← (WAY)-H
E
★★


E
★★
(PATH) ≤ ∅ ?
Call MVINXR
Call XRACC
Call DRILL
IX ← (SAVEIX)
IX ← IX-1
to SFLTR
N
(IX) ≟ ∅
Y
FTRAC
9
(OLDX) ← (IX)
Call MOVINY
FTR
D ← D-1
D ≟ ∅
Y
10
RETURN
N
B ← 9∅H
HL ← 872DH
HL ← ∅∅
HL ← HL+1
B ≟ ∅
N
Y
C ← (XINC)
HL ← (SAVEHL)
IX ← FRSTIX
FTRACE

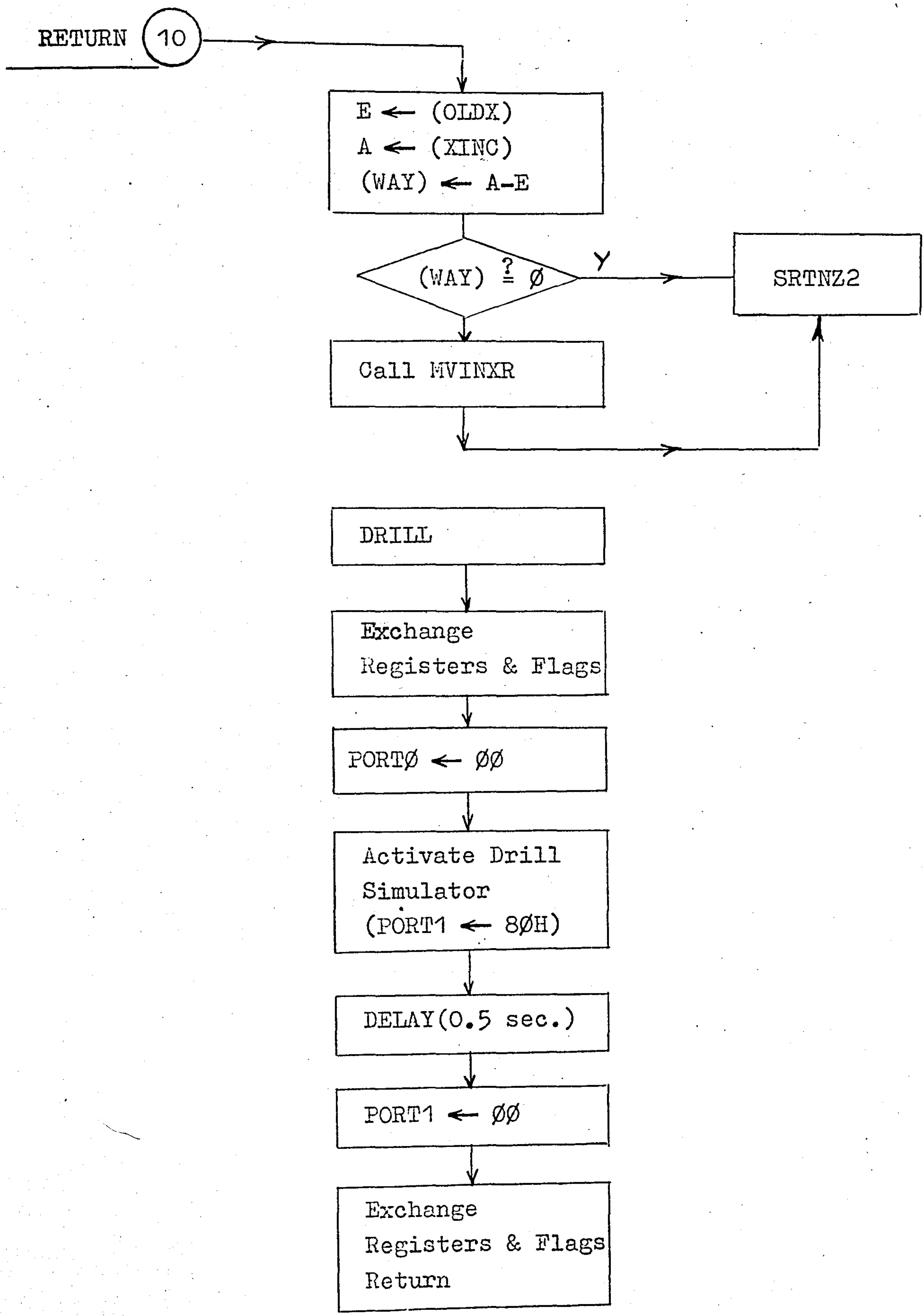


Figure 4.11 Flowchart of Drilling Program
(For odd-lines)

The flowchart of scanning and drilling the even-numbered lines, is about the same as given in Figure 4.11 with the differences as explained within the related parts of the drilling program section, and can be examined from the assembly language listing of the total program given in the appendix, from line number Ø511.

DISCUSSION & CONCLUSIONS

The stepper motors used in the system were high output torque,not fast steppers which were supplied before no calculation or work has been done,and they were not the best ones for this specific application where speed is a very important factor.Although,they are forced to be driven with a speed of 835 steps/sec,there are stepper motors that can operate with 4000 steps/sec,still supplying the torque needed in this system.Using such steppers could reduce the process time by a factor 5.

The construction of the mechanical stage scanner was so precise that either x or y direction stages had almost frictionless linear motion.This advantage of the assembly gave the chance of increasing the speed of the motors to their possible maximum stepping rate by losing some portion of the torque output.At maximum speed,which is 835 steps/sec stage moves with 1.325 cm/sec velocity (47.72 m/hr).At slow constant speed which is also the starting speed,246 step/sec, stage moves with 0.39 cm/sec velocity (14.05 m/hr).

In the beginning of the project,it was thought that the backlash of such a mechanical scanner could be the most probable danger for the precision of the whole system.But there were no backlash experienced even after complicated movement schemes.In case,there were methods advised such as spring system or adjustable nuts on lead-screws.

Although the system operation,especially,the dot-mask preperation seems to be tedious,it is in fact straight forward and there are some advantages coming from the scanning scheme due to this mask preperation.One of them is that, scanning is done only once,in order to determine the location of a hole-position and no need to calculate its origin,because y-coordinate of a hole is known when a line is about to be scanned,and only x-coordinate is determined for its exact location.

Drilling routine also brings a considerable amount of time reduction in the whole process by using the optimization algorithm.

After some mechanical modifications,system can be used as a real drilling machine utilizing its mentioned characteristics.Detector,lamp,glass plate combination can be designed to be replacable with the drill set,in order to perform the drilling process.

A P P E N D I C E S

APPENDIX A.

```
XLNGTH : EQU 87FØH  ;Memory location for X-length
XPATH  : EQU 87F1H  ;Memory location for X-path
YLNGTH : EQU 87F2H  ;Memory location for Y-length
YPATH  : EQU 87F3H  ;Memory location for Y-path
XSTEP1 : EQU Ø7AØH  ;Location of first X-motor constant
XSTEP4 : EQU Ø7A3H  ;Location of last X-motor constant
YSTEP1 : EQU Ø7BØH  ;Location of first Y-motor constant
YSTEP4 : EQU Ø7B3H  ;Location of last Y-motor constant
XINC   : EQU 87F4H  ;Location of incremented x-length
XL2    : EQU 87E8H  ;  "  of twice   "    x-length
LOC1   : EQU 87F5H  ;Location of byte count
LOC2   : EQU 87F6H  ;Location of incomplete store byte
STAM   : EQU 87FAH  ;Starting address location of memory
SAVEHL : EQU 87EEH  ;Location to save HL registers
SAVEIX : EQU 87FCH  ;Location to save IX register
LASTIX : EQU 87F8H  ;   "  to store last IX content
OLDX   : EQU 87ECH  ;   "  to store old position of motor
FRSTIX : EQU 873ØH  ;   "  to store first IX content
PATH   : EQU 87F7H  ;Location to store distance path
WAY    : EQU 87EBH  ;Location to store length way
STRAM  : EQU 8ØØØH  ;Starting address of RAM
A+DSTP : EQU 15H    ;Sum of acceleration+dec. steps
ACOUNT : EQU ØBH    ;Acceleration steps
DCOUNT : EQU ØAH    ;Deceleration steps
COUNT  : EQU Ø4H    ;Motor pulse sequence steps
BYTE   : EQU Ø8H    ;Bit count in a byte
STEP   : EQU Ø1H    ;One step length
GRID   : EQU 5ØH    ;One grid length,80D steps
MAXSPD : EQU 12H    ;Maximum speed constant
```

```
                LD IX,87F5H
                LD (IX+ØØ),Ø8H
                LD (IX+Ø1),ØØH
                LD HL,8ØØØH
                LD (STAM),HL
NGY         :   LD HL,YSTEP4      ;Frame Detection Program
                LD B,COUNT
OUT         :   LD A,(HL)         ;Move in 4-step sequence
                OUT (PORTØ),A     ;and input data
                CALL DLY
                DEC HL
                DJNZ OUT
                JP NC,IN1
                DEC C
                JP Z,STOP
                IN A,(PORT1)
                BIT Ø,A
                JP NZ,NGY
                CCF
                JP NGY
IN1         :   IN A,(PORT1)
                BIT Ø,A
                JP Z,NGY          ;If input is zero continue
                LD C,11H          ;to move,if not move 68 more
                SCF               ;steps to determine the depth
                JP NGY
STOP        :   XOR A
                OUT (PORTØ),A
                LD B,FFH
```

```
DEL1    : CALL DLY
          DJNZ DEL1
          LD C,29H          ;Move out from the upper frame
PSY     : LD HL,YSTEP1
          LD B,COUNT
OUT3    : LD A,(HL)
          OUT (PORTØ),A
          CALL DLY
          INC HL
          DJNZ OUT3
          DEC C
          JP NZ,PSY
          XOR A
          OUT (PORTØ),A
          LD B,FFH
DEL2    : CALL DLY
          DJNZ DEL2
NGX     : LD HL,XSTEP1      ;Search for the right side
          LD B,COUNT        ;frame line
OUT2    : LD A,(HL)
          OUT (PORTØ),A
          CALL DLY
          INC HL
          DJNZ OUT2
          IN A,(PORT1)
          BIT Ø,A
          JP Z,NGX
          XOR A
          OUT (PORTØ),A
```

```
                LD B,FFH
DEL3    :       CALL DLY
                DJNZ DEL3
                LD C,2ØH            ;Move out from the right side
PSX     :       LD HL,XSTEP4        ;frame
                LD B,COUNT
OUT4    :       LD A,(HL)
                OUT (PORTØ),A
                CALL DLY
                DEC HL
                DJNZ OUT4
                DEC C
                JP NZ,PSX
                XOR A               ;Stop on the zero position
                OUT (PORTØ),A       ;location
                LD B,FF
DEL4    :       CALL DLY
                DJNZ DEL4
FRLGTH  :       LD L,ØØ             ;Frame length program
FPSX    :       LD A,STEP           ;Measure x-length
                LD (WAY),A
                CALL MVINXL
                INC L
                IN A,(PORT1)
                BIT Ø,A
                JP Z,FPSX
                XOR A
                OUT (PORTØ),A
                LD B,FFH
```

```
DEL5   : CALL DLY
         DJNZ DEL5
         DEC L
         DEC L
         LD DE,XLNGTH
         LD A,L
         LD (DE),A
         INC L
         INC L
         LD A,L
         SUB A+DSTP
         LD (XPATH),A
         LD (PATH),A
         LD A,L
         LD (WAY),A
         CALL Z,MVINXR
         JP Z,DLX
         CALL C,MVINXR
         JP C,DLX
         CALL XRACC
DLX    : LD B,FFH
DEL6   : CALL DLY
         DJNZ DEL6
         LD L,ØØ
FPSY   : LD A,STEP        ;Measure y-length
         LD (WAY),A
         CALL MVINYD
         INC L
         IN A,(PORT1)
```

```
            BIT Ø,A
            JP Z,FPSY
            XOR A
            OUT (PORTØ),A
            LD B,FFH
DEL7     :  CALL DLY
            DJNZ DEL7
            DEC L
            DEC L
            LD DE,YLNGTH
            LD A,L
            LD (DE),A
            INC L
            INC L
            LD A,L
            SUB A+DSTP
            LD (YPATH),A
            LD (PATH),A
            LD A,L
            LD (WAY),A
            CALL Z,MVINYU
            JP Z,DLYY
            CALL C,MVINYU
            JP C,DLYY
            CALL YUACC
DLYY     :  LD B,FFH
DEL8     :  CALL DLY
            DJNZ DEL8
            LD A,(XPATH)
```

```
                DEC A
                DEC A
                LD (XPATH),A
                LD A,(YPATH)
                DEC A
                DEC A
                LD (YPATH),A
                LD A,(XLNGTH)
                INC A
                LD (XINC),A
                INC A
                LD (XL2),A
SCANPR      :   LD A,(XLNGTH)   ;Scanning Program
                SUB A+DSTP      ;Test for constant speed scanning
                JP Z,CSCAN      ;or by acceleration
                JP C,CSCAN
                LD A,(YLNGTH)
                LD L,A
                CALL STORE
SXPA        :   LD IY,ACONS     ;Start scanning the first line
                LD C,ACOUNT     ;by acceleration
SXPAH       :   LD H,ØØ
SXPAR       :   LD IX,XSTEP4
                LD B,COUNT
SXPAM       :   LD A,(IX+ØØ)
                OUT (PORTØ),A
                LD E,(IY+ØØ)
                CALL VDLY
                INC H
```

```
                LD A,GRID
                CP H
                JR Z+9
                DEC IX
                DJNZ SXPAM
                JP SXPAR
                INC IY
                CALL STORE
                DEC C
                JP NZ,SXPAH
                PUSH HL
                LD HL,XPATH
                LD C,(HL)
                POP HL
SXAPH         : LD H,ØØ
SXAPS         : LD IX,XSTEP4     ;Scanning with maximum speed
                LD B,COUNT
SXAPM         : LD A,(IX+ØØ)
                OUT (PORTØ),A
                LD E,MAXSPD
                CALL VDLY
                INC H
                LD A,GRID
                CP H
                JR Z,+9
                DEC IX
                DJNZ SXAPM
                JP SXAPS
                CALL STORE
```

```
                DEC C
                JP NZ,SXAPH
SXPD        :   LD IY,DCONS      ;Start to decelerate
                LD C,DCOUNT
SXPDH       :   LD H,ØØ
SXPDR       :   LD IX,XSTEP4
                LD B,COUNT
SXPDM       :   LD A,(IX+ØØ)
                OUT (PORTØ),A
                LD E,(IY+ØØ)
                CALL VDLY
                INC H
                LD A,GRID
                CP H
                JR Z,+9
                DEC IX
                DJNZ SXPDM
                JP SXPDR
                INC IY
                CALL STORE
                DEC C
                JP NZ,SXPDH
                EXX
                EX AF,AF'
                LD HL,LOC1
                LD C,(HL)
                LD A,BYTE
                AND C
                JP NZ,SX1
```

```
                CALL LASTST
                JP SP1
SX1     :       EX AF,AF'
                EXX
SP1     :       XOR A           ;Stop at the end of the line
                OUT (PORTØ),A
                LD B,1FH
SDEL1   :       CALL DLY
                DJNZ SDEL1
                LD A,STEP
                LD (WAY),A
                CALL MVINYD     ;Move one grid down
                LD B,1FH
SDEL2   :       CALL DLY
                DJNZ SDEL2
                DEC L           ;Test whether the card is
                JP Z,SRTNZ1     ;finished,if yes,return to
                CALL STORE      ;zero position
SXNA    :       LD IY,ACONS     ;if not,scan the next line
                LD C,ACOUNT
SXNAH   :       LD H,ØØ
SXNAR   :       LD IX,XSTEP1
                LD B,COUNT
SXNAM   :       LD A,(IX+ØØ)
                OUT (PORTØ),A
                LD E,(IY+ØØ)
                CALL VDLY
                INC H
                LD A,GRID
```

```
                CP H
                JR Z+9
                INC IX
                DJNZ SXNAM
                JP SXNAR
                INC IY
                CALL STORE
                DEC C
                JP NZ,SXNAH
                PUSH HL
                LD HL,XPATH
                LD C,(HL)
                POP HL
SXDPH       :   LD H,ØØ
SXDPS       :   LD IX,XSTEP1
                LD B,COUNT
SXDPM       :   LD A,(IX+ØØ)
                OUT (PORTØ),A
                LD E,MAXSPD
                CALL VDLY
                INC H
                LD A,GRID
                CP H
                JR Z+9
                INC IX
                DJNZ SXDPM
                JP SXDPS
                CALL STORE
                DEC C
```

```
                JP NZ,SXDPH
SXND      :     LD IY,DCONS
                LD C,DCOUNT
SXNDH     :     LD H,ØØ
SXNDR     :     LD IX,XSTEP1
                LD B,COUNT
SXNDM     :     LD A,(IX+ØØ)
                OUT (PORTØ),A
                LD E,(IY+ØØ)
                CALL VDLY
                INC H
                LD A,GRID
                CP H
                JR Z,+9
                INC IX
                DJNZ SXNDM
                JP SXNDR
                INC IY
                CALL STORE
                DEC C
                JP NZ,SXNDM
                EXX
                EX AF,AF'
                LD HL,LOC1
                LD C,(HL)
                LD A,BYTE
                AND C
                JP NZ,SX2
                CALL LASTST
```

```
                 JP SP2
        SX2    : EX AF,AF'
                 EXX
        SP2    : XOR A
                 OUT (PORTØ),A
                 LD B,1FH
        SDEL3  : CALL DLY
                 DJNZ SDEL3
                 LD A,STEP
                 LD (WAY),A
                 CALL MVINYD
                 CALL STORE
                 LD B,1FH
        SDELR  : CALL DLY
                 DJNZ SDELR
                 DEC L
                 JP Z,SRTNZ2
                 JP SXPA
        CSCAN  : LD A,(YLNGTH)  ;Constant Speed Scanning Program
                 LD L,A
                 CALL STORE
        CSTART : LD A,(XLNGTH)
                 LD C,A
        LCONT  : LD A,STEP
                 LD (WAY),A
                 CALL MVINXL
                 CALL STORE
                 DEC C
                 JP NZ,LCONT
```

```
              EXX
              EX AF,AF'
              LD HL,LOC1
              LD C,(HL)
              LD A,BYTE
              AND C
              JP NZ,CX1
              CALL LASTST
              JP CP1
CX1         : EX AF,AF'
              EXX
CP1         : XOR A
              OUT (PORTØ),A
              LD B,1FH
CDEL1       : CALL DLY
              DJNZ CDEL1
              LD A,STEP
              LD (WAY),A
              CALL MVINYD
              LD B,1FH
SDEL3       : CALL DLY
              DJNZ SDEL3
              DEC L
              JP Z,SRTNZ1
              CALL STORE
              LD A,(XLNGTH)
              LD C,A
RCONT       : LD A,STEP
              LD (WAY),A
```

```
                CALL MVINXR
                CALL STORE
                DEC C
                JP NZ,RCONT
                EXX
                EX AF,AF'
                LD HL,LOC1
                LD C,(HL)
                LD A,BYTE
                AND C
                JP NZ,CX2
                CALL LASTST
                JP CP2
CX2         :   EX AF,AF'
                EXX
CP2         :   XOR A
                OUT (PORTØ),A
                LD B,1FH
SDELR       :   CALL DLY
                DJNZ SDELR
                LD A,STEP
                LD (WAY),A
                CALL MVINYD
                DEC L
                JP Z,SRTNZ2
                CALL STORE
                JP CSTART
STORE       :   EXX                    ;Store Program
                EX AF,AF'
```

```
                IN A,(PORT1)
                LD HL,LOC2
                LD B,(HL)
                RRC B
                OR B
                LD HL,LOC1
                LD C,(HL)
                LD (LOC2),A
                DEC C
                LD (HL),C
                JP Z,STM
                EX AF,AF'
                EXX
                RET
       LASTST : LD A,(LOC1)      ;Last store
                LD B,A
                LD A,(LOC2)
       ROT    : RRC A
                DJNZ ROT
                LD (LOC2),A
       STM    : LD A,(LOC2)      ;Store to memory
                RRC A
                LD HL,(STAM)
                LD (HL),A
                INC HL
                LD (STAM),HL
                LD A,BYTE
                LD (LOC1),A
                LD A,ØØ
```

```
                 LD (LOC2),A
                 EX AF,AF'
                 EXX
                 RET
        SRTNZ1 : LD B,FFH       ;Return to zero position from
        SDELX  : CALL DLY       ;an odd numbered line
                 DJNZ SDELX
                 LD A,(XLNGTH)
                 LD (WAY),A
                 SUB A+DSTP
                 CALL  Z,MVINXR
                 JP Z,SRTNZ2
                 CALL C,MVINXR
                 JP C,SRTNZ2
                 LD A,(XPATH)
                 LD (PATH),A
                 CALL XRACC
        SRTNZ2 : LD B,FFH      ;Return to zero-position from
        SDEL4  : CALL DLY      ;an even-line
                 DJNZ SDEL4
                 LD A,(YLNGTH)
                 LD (WAY),A
                 SUB A+DSTP
                 CALL Z,MVINYU
                 JP Z,END
                 CALL C,MVINYU
                 JP C,END
                 LD A,(YPATH)
                 LD (PATH),A
```

```
                CALL YUACC
END     :       XOR A
                OUT (PORTØ),A
                LD B,FFH
SDEL5   :       CALL DLY
                DJNZ SDEL5
                LD B,9ØH          ;DRILLING Program
                LD HL,872DH
ZERØ    :       INC HL
                LD (HL),ØØ
                DJNZ ZERØ
                LD A,(XINC)
                LD C,A
                LD (OLDX),A
                SRL A
                LD (HALFX),A
                LD A,(YLNGTH)
                LD D,A
                LD HL,STRAM
                LD IX,FRSTIX
                JP STR
FTRACE  :       BIT Ø,(HL)        ;Routine to trace an even-line
                JP NZ,FLDØ
FTRØ    :       DEC C
                JP Z,NXTLIN
                BIT 1,(HL)
                JP NZ,FLD1
FTR1    :       DEC C
                JP Z,NXTLIN
```

```
                BIT 2,(HL)
                JP NZ,FLD2
FTR2    : DEC C
                JP Z,NXTLIN
                BIT 3,(HL)
                JP NZ,FLD3
FTR3    : DEC C
                JP Z,NXTLIN
                BIT 4,(HL)
                JP NZ,FLD4
FTR4    : DEC C
                JP Z,NXTLIN
                BIT 5,(HL)
                JP NZ,FLD5
FTR5    : DEC C
                JP Z,NXTLIN
                BIT 6,(HL)
                JP NZ,FLD6
FTR6    : DEC C
                JP Z,NXTLIN
                BIT 7,(HL)
                JP NZ,FLD7
FTR7    : DEC C
                JP Z,NXTLIN
                INC HL
                JP FTRACE
FLDØ    : LD A,(XL2)
                SUB C
                LD (IX+Ø),A
```

```
Ø548              INC IX
Ø549              JP FTRØ
Ø56Ø    FLD1    : LD A,(XL2)
Ø561              SUB C
Ø562              LD (IX+Ø),A
Ø563              INC IX
Ø564              JP FTR1
Ø565    FLD2    : LD A,(XL2)
Ø566              SUB C
Ø567              LD (IX+Ø),A
Ø568              INC IX
Ø569              JP FTR2
Ø57Ø    FLD3    : LD A,(XL2)
Ø571              SUB C
Ø572              LD (IX+Ø),A
Ø573              INC IX
Ø574              JP FTR3
Ø575    FLD4    : LD A,(XL2)
Ø576              SUB C
Ø577              LD (IX+Ø),A
Ø578              INC IX
Ø579              JP FTR4
Ø58Ø    FLD5    : LD A,(XL2)
Ø581              SUB C
Ø582              LD (IX+Ø),A
Ø583              INC IX
Ø584              JP FTR5
Ø585    FLD6    : LD A,(XL2)
Ø586              SUB C
```

```
                LD (IX+0),A
                INC IX
                JP FTR6
FLD7    :       LD A,(XL2)
                SUB C
                LD (IX+0),A
                INC IX
                JP FTR7
NXTLIN  :       DEC IX
                LD (LASTIX),IX
                INC HL
                LD (SAVEHL),HL
                LD IX,FRSTIX      ;Test whether there is a hole
                LD A,(IX+0)       ;on this line
                LD E,(IX+1)
                OR E
                CALL Z,MOVINY
                JP Z,STRACE
                LD A,(IX+1)       ;Test whether there is only
                LD E,(IX+2)       ;one hole
                OR E
                JP Z,FLEFT
                LD A,(OLDX)
                LD E,A
                LD IX,(LASTIX)
                LD A,(IX+0)
                SUB E
                JP Z,RZERO
                JP C,LRM
```

```
                LD B,A
                LD A,(FRSTIX)
                LD E,A
                LD A,(OLDX)
                SUB E
                JP Z,LZERO
                JP C,RM
                SUB B
                JP NC,FRIGHT
                JP FLEFT
FRIGHT        : LD A,(OLDX)     ;Routine to move right
                LD E,A
                LD IX,(LASTIX)
                LD A,(IX+Ø)
                SUB E
                JP Z,RZERO
                JP NC,RMOVE
LRM           : LD A,(IX+Ø)
                LD E,A
                LD A,(OLDX)
                SUB E
                LD (WAY),A
                LD H,A+DSTP
                SUB H
                CALL Z,MVINXL
                JP Z,RZERO
                CALL C,MVINXL
                JC C,RZERO
                LD (PATH),A
```

```
                CALL XLACC
                JP RZERO
FLEFT   :  LD A,(FRSTIX)    ;Routine to move left
                LD E,A
                LD A,(OLDX)
                SUB E
                JP LZERO
                JP NC,LMOVE
RM      :  LD A,(OLDX)
                LD E,A
                LD A,(FRSTIX)
                SUB E
                LD (WAY),A
                LD H,A+DSTP
                SUB H
                CALL Z,MVINXR
                JP Z,LZERO
                CALL C,MVINXR
                JP C,LZERO
                LD (PATH),A
                CALL XRACC
                JP LZERO
RMOVE   :  LD (WAY),A
RATEST  :  LD H,A+DSTP
                SUB H
                CALL Z,MVINXR
                JP Z,RZERO
                CALL C,MVINXR
                JP C,RZERO
```

```
                LD (PATH),A
                CALL XRACC
RZERO         : NOP
                CALL DRILL
                LD IX,(LASTIX)
                LD A,(IX-1)
                OR A
                JP Z,STRAC
FRRTL         : LD E,(IX-1)     ;Routine to move from right
                LD A,(IX+Ø)     ;to left on an evenline
                LD (SAVEIX),IX
                SUB E
                LD (WAY),A
                LD H,A+DSTP
                SUB H
                CALL Z,MVINXL
                JP Z,DL
                CALL C,MVINXL
                JP C,DL
                LD (PATH),A
                CALL XLACC
DL            : NOP
                CALL DRILL
                LD IX,(SAVEIX)
                DEC IX
                LD A,(IX-1)
                OR A
                JP Z,STRAC
                JP FRRTL
```

```
LMOVE  : LD (WAY),A
LATEST : LD H,A+DSTP
         LD A,(WAY)
         SUB H
         CALL Z,MVINXL
         JP Z,LZERO
         CALL C,MVINXL
         JP C,LZERO
         LD (PATH),A
         CALL XLACC
LZERO  : NOP
         CALL DRILL
         LD IX,FRSTIX
         LD A,(IX+1)
         OR A
         JP Z,STRAC
FRLTR  : LD E,(IX+0)      ;Routine to move from left
         LD A,(IX+1)      ;to right on an evenline
         LD (SAVEIX),IX
         SUB E
         LD (WAY),A
         LD H,A+DSTP
         SUB H
         CALL Z,MVINXR
         JP Z,DR
         CALL C,MVINXR
         JP C,DR
         LD (PATH),A
         CALL XRACC
```

```
DR      : NOP
          CALL DRILL
          LD IX,(SAVEIX)
          INC IX
          LD A,(IX+1)
          OR A
          JP Z,STRAC
          JP FRLTR
DRILL   : EXX               ;Drill simulating routine
          EX AF,AF'
          XOR A
          OUT (PORTØ),A
          LD A,8ØH
          OUT (PORT1),A
          LD B,7FH
DR2     : CALL DLY
          DJNZ DR2
          XOR A
          OUT (PORT1),A
          EX AF,AF'
          EXX
          RET
STRAC   : LD A,(IX+Ø)
          LD (OLDX),A
          CALL MOVINY
STRACE  : DEC D
          JP Z,RETURN
          LD B,9ØH
          LD HL,872DH
```

```
SZERO  : INC HL
         LD (HL),∅∅
         DJNZ SZERO
         LD A,(XINC)
         LD C,A
         LD HL,(SAVEHL)
         LD IX,FRSTIX
STR    : BIT ∅,HL        ;Routine to trace an odd-line
         JP NZ,S∅
R∅     : DEC C
         JP Z,SECLIN
         BIT 1,(HL)
         JP NZ,S1
R1     : DEC C
         JP Z,SECLIN
         BIT 2,(HL)
         JP NZ,S2
R2     : DEC C
         JP Z,SECLIN
         BIT 3,(HL)
         JP NZ,S3
R3     : DEC C
         JP Z,SECLIN
         BIT 4,(HL)
         JP NZ,S4
R4     : DEC C
         JP Z,SECLIN
         BIT 5,(HL)
         JP NZ,S5
```

```
R5      : DEC C
          JP Z,SECLIN
          BIT 6,(HL)
          JP NZ,S6
R6      : DEC C
          JP Z,SECLIN
          BIT 7,(HL)
          JP NZ,S7
R7      : DEC C
          JP Z,SECLIN
          INC HL
          JP STR
SØ      : LD (IX+Ø),C
          INC IX
          JP RØ
S1      : LD (IX+Ø),C
          INC IX
          JP R1
S2      : LD (IX+Ø),C
          INC IX
          JP R2
S3      : LD (IX+Ø),C
          INC IX
          JP R3
S4      : LD (IX+Ø),C
          INC IX
          JP R4
S5      : LD (IX+Ø),C
          INC IX
```

```
                 JP R5
S6             : LD (IX+0),C
                 INC IX
                 JP R6
S7             : LD (IX+0),C
                 INC IX
                 JP R7
SECLIN         : DEC IX
                 LD (LASTIX),IX
                 INC HL
                 LD (SAVEHL),HL
                 LD IX,FRSTIX
                 LD A,(IX+0)
                 LD E,(IX+1)
                 OR E
                 CALL Z,MOVINY
                 JP Z,FTR
                 LD A,(IX+1)
                 LD A,(IX+2)
                 OR E
                 JP Z,SRIGHT
                 LD A,(OLDX)
                 LD E,A
                 LD A,(FRSTIX)
                 SUB E
                 JP Z,SRZERO
                 JP C,HLMOVE
                 LD B,A
                 LD IX,(LASTIX)
```

```
                LD E,(IX+Ø)
                LD A,(OLDX)
                SUB E
                JP Z,SLZERO
                JP C,RLM
                SUB B
                JP C,SLEFT
       SRIGHT : LD A,(OLDX)     ;Routine to move right on
                LD E,A          ;an odd-line
                LD A,(FRSTIX)
                SUB E
                JP Z,SRZERO
                JP NC,SRMOVE
       HLMOVE : LD A,(FRSTIX)
                LD E,A
                LD A,(OLDX)
                SUB E
                LD (WAY),A
                LD H,A+DSTP
                SUB H
                CALL Z,MVINXL
                JP Z,SRZERO
                CALL C,MVINXL
                JP C,SRZERO
                LD (PATH),A
                CALL XLACC
                JP SRZERO
       SRMOVE : LD (WAY),A
                LD H,A+DSTP
```

```
                SUB H
                CALL Z,MVINXR
                JP Z,SRZERO
                CALL C,MVINXR
                JP C,SRZERO
                LD (PATH),A
                CALL XRACC
SRZERO        : NOP
                CALL DRILL
                LD IX,FRSTIX
                LD A,(IX+1)
                OR A
                JP Z,FTRAC
SFRTL         : LD E,(IX+1)      ;Routine to move from right
                LD A,(IX+Ø)      ;to left on an odd-line
                LD (SAVEIX),IX
                SUB E
                LD (WAY),A
                LD H,A+DSTP
                SUB H
                CALL Z,MVINXL
                JP Z,SDL
                CALL C,MVINXL
                JP C,SDL
                LD (PATH),A
                CALL XLACC
SDL           : NOP
                CALL DRILL
                LD IX,(SAVEIX)
```

```
Ø9Ø7                  INC IX
Ø9Ø8                  LD A,(IX+1)
Ø91Ø                  OR A
Ø911                  JP Z,FTRAC
Ø912                  JP SFRTL
Ø913     SLEFT    :   LD IX,(LASTIX)  ;Routine to move left on
Ø914                  LD A,(IX+Ø)     ;an odd-line
Ø915                  LD E,A
Ø916                  LD A,(OLDX)
Ø917                  SUB E
Ø918                  JP Z,SLZERO
Ø919                  JP NC,SLMOVE
Ø92Ø     RLM      :   LD A,(OLDX)
Ø921                  LD E,A
Ø922                  LD A,(IX+Ø)
Ø923                  SUB E
Ø924                  LD (WAY),A
Ø925                  LD H,A+DSTP
Ø926                  SUB H
Ø927                  CALL Z,MVINXR
Ø928                  JP Z,SLZERO
Ø929                  CALL C,MVINXR
Ø93Ø                  JP C,SLZERO
Ø931                  LD (PATH),A
Ø932                  CALL XRACC
Ø933                  JP SLZERO
Ø934     SLMOVE   :   LD (WAY),A
Ø935                  LD H,A+DSTP
Ø936                  LD A,(WAY)
```

```
                SUB H
                CALL Z,MVINXL
                JP Z,SLZERO
                CALL C,MVINXL
                JP C,SLZERO
                LD (PATH),A
                CALL XLACC
        SLZERO : NOP
                CALL DRILL
                LD IX,(LASTIX)
                LD A,(IX-1)
                OR A
                JP Z,FTRAC
        SFLTR  : LD E,(IX+Ø)        ;Routine to move from left
                LD A,(IX-1)         ;to right on an odd-line
                LD (SAVEIX),IX
                SUB E
                LD (WAY),A
                LD H,A+DSTP
                SUB H
                CALL Z,MVINXR
                JP Z,SDR
                CALL C,MVINXR
                JP C,SDR
                LD (PATH),A
                CALL XRACC
        SDR    : NOP
                CALL DRILL
                LD IX,(SAVEIX)
```

```
                DEC IX
                LD A,(IX-1)
                OR A
                JP Z,FTRAC
                JP SFLTR
FTRAC   :       LD A,(IX+Ø)
                LD (OLDX),A
                CALL MOVINY
FTR     :       DEC D
                JP Z,RETURN
                LD B,9ØH
                LD HL,872DH
ZR      :       INC HL
                LD (HL),ØØ
                DJNZ ZR
                LD A,(XINC)
                LD C,A
                LD HL,(SAVEHL)
                LD IX,FRSTIX
                JP FTRACE
RETURN  :       LD A,(OLDX)     ;Return to zero-position after
                LD E,A          ;drilling of the whole card is
                LD A,(XINC)     ;finished
                SUB E
                JP Z,SRTNZ2
                LD (WAY),A
                CALL MVINXR
                JP SRTNZ2
```

```
MOVINY : LD A,STEP          ;Routine to move one grid in
         LD (WAY),A         ;y-direction downwards.
         CALL MVINYD
         RET
DLY    : LD D,7DH           ;Constant delay routine.
LOOP   : LD E,Ø3H
         DEC E
         JR NZ,-1
         DEC D
         JP NZ,LOOP
         RET
VDLY   : LD D,Ø8H           ;Variable delay routine.
IND    : DEC D
         JP NZ,IND
         DEC E
         JP NZ,VDLY
         RET
MVINXL : EXX                ;Routine to move left in
         EX AF,AF'          ;x-direction with constant
         LD A,(WAY)         ;speed.
         LD C,A
RH     : LD H,ØØ
RLINE  : LD IX,XSTEP4
         LD B,COUNT
RNEXT  : LD A,(IX+ØØ)
         OUT (PORTØ),A
         CALL DLY
         INC H
         LD A,GRID
```

```
                CP H
                JR Z,+9
                DEC IX
                DJNZ RNEXT
                JP RLINE
                DEC C
                JP NZ,RH
                EXX
                EX AF,AF'
                RET
MVINXR  :       EXX                 ;Routine to move right in
                EX AF,AF'           ;x-direction with constant
                LD A,(WAY)          ;speed.
                LD C,A
LH      :       LD H,00
LLINE   :       LD IX,XSTEP1
                LD B,COUNT
LNEXT   :       LD A,(IX+00)
                OUT (PORT0),A
                CALL DLY
                INC H
                LD A,GRID
                CP H
                JR Z,+9
                INC IX
                DJNZ LNEXT
                JP LLINE
                DEC C
                JP NZ,LH
```

```
                EXX
                EX AF,AF'
                RET
ACONS   : EQU Ø7CØH        ;Acceleration constants
DCONS   : EQU Ø7DØH        ;Deceleration constants
YUACC   : EXX              ;Routine to move the detector
          EX AF,AF'        ;up in y-direction by accelera-
          LD IY,ACONS      ;tion.
          LD C,ACOUNT
YRTH    : LD H,ØØ
YARTN   : LD IX,YSTEP4
          LD B,COUNT
BMOVE   : LD A,(IX+ØØ)
          OUT (PORTØ),A
          LD E,(IY+ØØ)
          CALL VDLY
          INC H
          LD A,GRID
          CP H
          JR Z,+9
          DEC IX
          DJNZ BMOVE
          JP YARTN
          INC IY
          DEC C
          JP NZ,YRTH
          LD HL,PATH
          LD C,(HL)
YMAXH   : LD H,ØØ
```

```
YMAX   : LD IX,YSTEP4
         LD B,COUNT
YAMV   : LD A,(IX+ØØ)
         OUT (PORTØ),A
         LD E,MAXSPD
         CALL VDLY
         INC H
         LD A,GRID
         CP H
         JR Z,+9
         DEC IX
         DJNZ YAMV
         JP YMAX
         DEC C
         JP NZ,YMAXH
YDEC   : LD IY,DCONS
         LD C,DCOUNT
YDRTH  : LD H,ØØ
YDRTN  : LD IX,YSTEP4
         LD B,COUNT
YDMV   : LD A,(IX+ØØ)
         OUT (PORTØ),A
         LD E,(IY+ØØ)
         CALL VDLY
         INC H
         LD A,GRID
         CP H
         JR Z,+9
         DEC IX
```

```
        DJNZ YDMV
        JP YDRTN
        INC IY
        DEC C
        JP NZ,YDRTH
        XOR A
        OUT (PORTØ),A
        EX AF,AF'
        EXX
        RET
```

```
MVINYU : EXX            ;Routine to move the detector
         EX AF,AF'      ;up in y-direction with
         LD A,(WAY)     ;constant speed.
         LD C,A
         LD H,ØØ
ULINE  : LD IX,YSTEP4
         LD B,COUNT
UNEXT  : LD A,(IX+ØØ)
         OUT (PORTØ),A
         CALL DLY
         INC H
         LD A,GRID
         CP H
         JR Z,+9
         DEC IX
         DJNZ UNEXT
         DEC C
         JP NZ,ULINE
         EXX
         EX AF,AF'
         XOR A
         OUT (PORTØ),A
         RET
```

```
MVINYD : EXX              ;Routine to move the detector
         EX AF,AF'        ;down in y-direction with
         LD A,(WAY)       ;constant speed.
         LD C,A
         LD H,ØØ
DLINE  : LD IX,YSTEP1
         LD B,COUNT
DNEXT  : LD A,(IX+ØØ)
         OUT (PORTØ),A
         CALL DLY
         INC H
         LD A,GRID
         CP H
         JR Z,+9
         INC IX
         DJNZ DNEXT
         DEC C
         JP NZ,DLINE
         EXX
         EX AF,AF'
         XOR A
         OUT (PORTØ),A
         RET
```

```
XRACC  : EXX              ;Routine to move the detector
         EX AF,AF'        ;right in x-direction by
         LD IY,ACONS1     ;acceleration.
         LD C,ACOUNT      ;Acceleration starts.
XRTH   : LD H,ØØ
XARTN  : LD IX,XSTEP1
         LD B,COUNT
AMOVE  : LD A,(IX+Ø)
         OUT (PORTØ),A
         LD E,(IY+Ø)
         CALL VDLY
         INC H
         LD A,GRID
         CP H
         JR Z,+9
         INC IX
         DJNZ AMOVE
         JP XARTN
         INC IY
         DEC C
         JP NZ,XRTH
         LD HL,PATH
         LD C,(HL)
XNMXH  : LD H,ØØ          ;Maximum speed is reached,and
XNMAX  : LD IX,XSTEP1     ;the stage moves with that
         LD B,COUNT       ;speed PATH long.
XAMV   : LD A,(IX+Ø)
         OUT (PORTØ),A
```

```
                LD E,MAXSPD
                CALL VDLY
                INC H
                LD A,GRID
                CP H
                JR Z,+9
                INC IX
                DJNZ XAMV
                JP XNMAX
                DEC C
                JP NZ,XNMXH
XDEC      :     LD IY,DCONS1      ;Deceleration begins.
                LD C,DCOUNT
XDRTH     :     LD H,ØØ
XDRTN     :     LD IX,XSTEP1
                LD B,COUNT
XDMV      :     LD A,(IX+Ø)
                OUT (PORTØ),A
                LD E,(IY+Ø)
                CALL VDLY
                INC H
                LD A,GRID
                CP H
                JR Z,+9
                INC IX
                DJNZ XDMV
                JP XDRTN
                INC IY
```

```
                DEC C
                JP NZ,XDRTH
                XOR A           ;Deceleration lasts,and the
                OUT (PORTØ),A   ;x-stage stops.Then the routine
                EXX             ;returns to where it is called.
                EX AF,AF'
                RET
XLACC   :  EXX             ;Routine to move the detector
                EX AF,AF'       ;left in x-direction by
                LD IY,ACONS1    ;acceleration using the same
                LD C,ACOUNT     ;procedures described in XRACC
SXPAH   :  LD H,ØØ         ;routine.
SXPAR   :  LD IX,XSTEP4
                LD B,COUNT
SXPAM   :  LD A,(IX+Ø)
                OUT (PORTØ),A
                LD E,(IY+ØØ)
                CALL VDLY
                INC H
                LD A,GRID
                CP H
                JR Z,+9
                DEC IX
                DJNZ SXPAM
                JP SXPAR
                INC IY
                DEC C
                JP NZ,SXPAH
```

```
                LD HL,PATH
                LD C,(HL)
SXAPH   :       LD H,00
SXAPS   :       LD IX,XSTEP4
                LD B,COUNT
SXAPM   :       LD A,(IX+0)
                OUT (PORT0),A
                LD E,MAXSPD
                CALL VDLY
                INC H
                LD A,GRID
                CP H
                JR Z,+9
                DEC IX
                DJNZ SXAPM
                JP SXAPS
                DEC C
                JP NZ,SXAPH
SXPD    :       LD IY,DCONS1
                LD C,DCOUNT
SXPDH   :       LD H,00
SXPDR   :       LD IX,XSTEP4
                LD B,COUNT
SXPDM   :       LD A,(IX+0)
                OUT (PORT0),A
                LD E,(IY+0)
                CALL VDLY
                INC H
```

```
LD A,GRID
CP H
JR Z,+9
DEC IX
DJNZ SXPDM
JP SXPDR
INC IY
DEC C
JP NZ,SXPDH
EXX
EX AF,AF'
RET
```

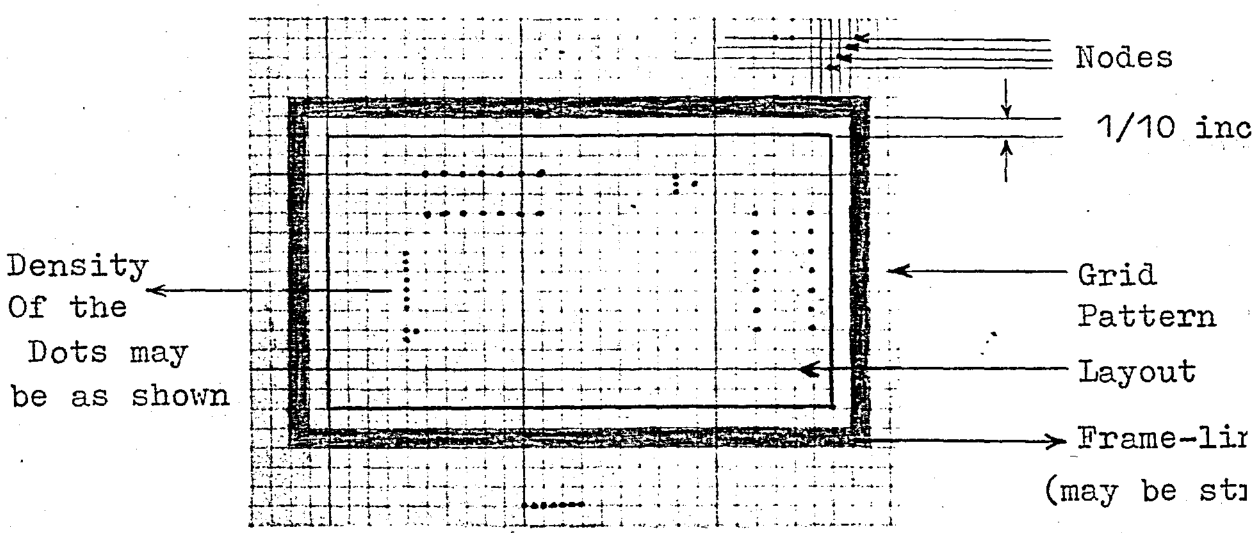
Appendix B.

OPERATING INSTRUCTIONS

1). PREPARATION OF THE DOT-MASK

i. After the printed circuit layout is drawn, it is placed on a grid pattern in which node separation is 1/10 inch as shown below. On top of these two, the clean transparency is located.

ii. The frame lines of the dot mask are drawn onto the transparency such that every side of the dot-mask is 1/10 inch greater than the frame of the circuit layout. Frame line thckness of 1/10 inch is enough. A 0.5mm drawing pen of black colour can be used. (RAPIDO 0.5). Or rather than drawing the frame lines, a strip of 2mm. thickness can also be used.



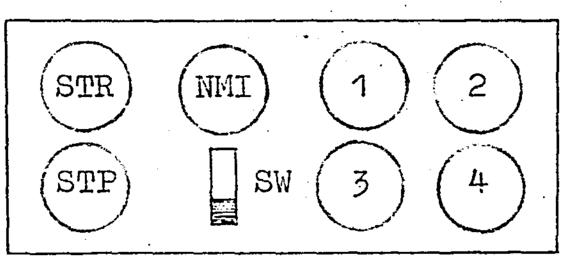
iii. Then hole-positions are marked on to the transparency by locating them at the nodes of the grid pattern. (one more hole can be marked between each node of the above grid pattern, because the scanning resolution of the system is 1/20 inch).

If there are hole-positions on the layout which do not coincide with the nodes of the grid pattern, they should be placed onto the nearest node (with an error less than 0.54 mm).

After the dot-mask is prepared, it is correctly placed onto the glass plate of the scanner, carefully matching the upper right corner of the mask frame to the right-angled marker on the glass.

2). RUNNING THE SYSTEM

i. System is turned on using the switch on the control panel as shown below. Power on reset, runs the reset routine and HALT LED is on, located on the microprocessor card. Then using the manual control switches 1,2,3,4, the detector is positioned somewhere in the dot-mask frame.



Control Panel

STR Button : START
STP " : STOP
NMI " : Emergency stop(when used system must be reset)
1 " : X-motor (stage moves towards the x-motor)
2 " : X-motor (stage moves away from the x-motor)
3 " : Y-motor (stage moves towards the y-motor)
4 " : Y-motor (stage moves away from the y-motor)

ii. Pressing the START button commences the programs.After the scanning of the whole card is finished, detector returns to the upper right corner (zero-position), and stops.HALT LED turns on again.

iii. This time START button starts the drilling program.(At each drilling hole position a LED flashes on for a few seconds to simulate drilling time.)

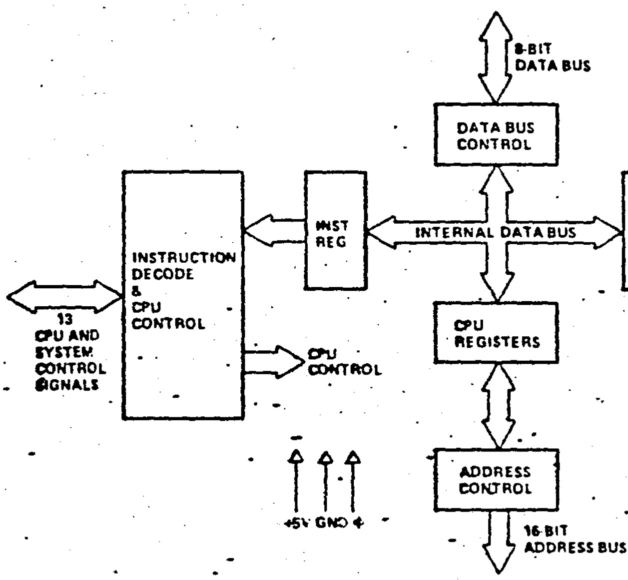
iv. After the completion of the drilling process, detector(now it is the drill) returns to zero-position and stops(again the HALT LED is on).Drilling process can be repeated as many times as desired by pressing the STR button.

v. For a new dot-mask scanning,a reset must be given to the system.Then continue from step ii.

APPENDIX C.

2.0 Z-80 CPU ARCHITECTURE

A block diagram of the internal architecture of the Z-80 CPU is shown in figure 2.0-1. The diagram shows all of the major elements in the CPU and it should be referred to throughout the following description.



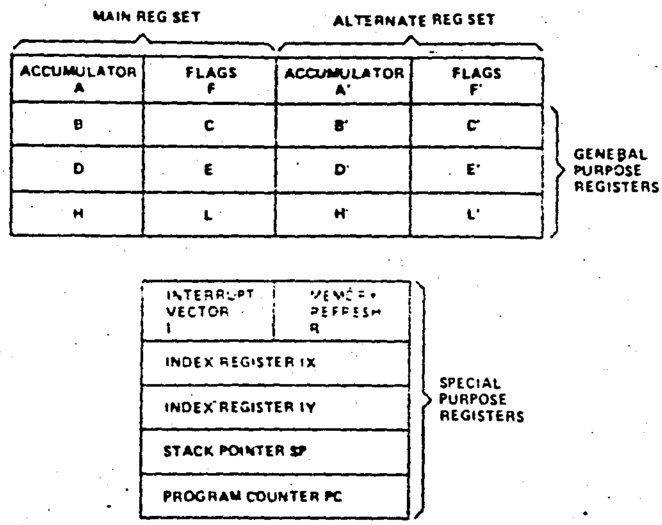
Z-80 CPU BLOCK DIAGRAM
FIGURE 2.0-1

2.1 CPU REGISTERS

The Z-80 CPU contains 208 bits of R/W memory that are accessible to the programmer. Figure 2.0-2 illustrates how this memory is configured into eighteen 8-bit registers and four 16-bit registers. All Z-80 registers are implemented using static RAM. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers.

Special Purpose Registers

1. Program Counter (PC). The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs the new value is automatically placed in the PC, overriding the incrementer.
2. Stack Pointer (SP). The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack into specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation.



Z-80 CPU REGISTER CONFIGURATION
FIGURE 2.0-2

3. **Two Index Registers (IX & IY).** The two independent index registers hold a 16-bit base address that is used in indexed addressing modes. In this mode, an index register is used as a base to point to a region in memory from which data is to be stored or retrieved. An additional byte is included in indexed instructions to specify a displacement from this base. This displacement is specified as a two's complement signed integer. This mode of addressing greatly simplifies many types of programs, especially where tables of data are used.

4. **Interrupt Page Address Register (I).** The Z-80 CPU can be operated in a mode where an indirect call to any memory location can be achieved in response to an interrupt. The I Register is used for this purpose to store the high order 8-bits of the indirect address while the interrupting device provides the lower 8-bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with absolute minimal access time to the routine.

5. **Memory Refresh Register (R).** The Z-80 CPU contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. Seven bits of this 8 bit register are automatically incremented after each instruction fetch. The eighth bit will remain as programmed as the result of an LD R, A instruction. The data in the refresh counter is sent out on the lower portion of the address bus along with a refresh control signal while the CPU is decoding and executing the fetched instruction. This mode of refresh is totally transparent to the programmer and does not slow down the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer. During refresh, the contents of the I register are placed on the upper 8 bits of the address bus.

Accumulator and Flag Registers

The CPU includes two independent 8-bit accumulators and associated 8-bit flag registers. The accumulator holds the results of 8-bit arithmetic or logical operations while the flag register indicates specific conditions for 8 or 16-bit operations, such as indicating whether or not the result of an operation is equal to zero. The programmer selects the accumulator and flag pair that he wishes to work with with a single exchange instruction so that he may easily work with either pair.

General Purpose Registers

There are two matched sets of general purpose registers, each set containing six 8-bit registers that may be used individually as 8-bit registers or as 16-bit register pairs by the programmer. One set is called BC, DE and HL while the complementary set is called BC', DE' and HL'. At any one time the programmer can select either set of registers to work with through a single exchange command for the entire set. In systems where fast interrupt response is required, one set of general purpose registers and an accumulator/flag register may be reserved for handling this very fast routine. Only a simple exchange commands need be executed to go between the routines. This greatly reduces interrupt service time by eliminating the requirement for saving and retrieving register contents in the external stack during interrupt or subroutine processing. These general purpose registers are used for a wide range of applications by the programmer. They also simplify programming, especially in ROM based systems where little external read/write memory is available.

2.2 ARITHMETIC & LOGIC UNIT (ALU)

The 8-bit arithmetic and logical instructions of the CPU are executed in the ALU. Internally the ALU communicates with the registers and the external data bus on the internal data bus. The type of functions performed by the ALU include:

Add	Left or right shifts or rotates (arithmetic and logical)
Subtract	Increment
Logical AND	Decrement
Logical OR	Set bit
Logical Exclusive OR	Reset bit
Compare	Test bit

2.3 INSTRUCTION REGISTER AND CPU CONTROL

As each instruction is fetched from memory, it is placed in the instruction register and decoded. The control sections performs this function and then generates and supplies all of the control signals necessary to read or write data from or to the registers, control the ALU and provide all required external control signals.

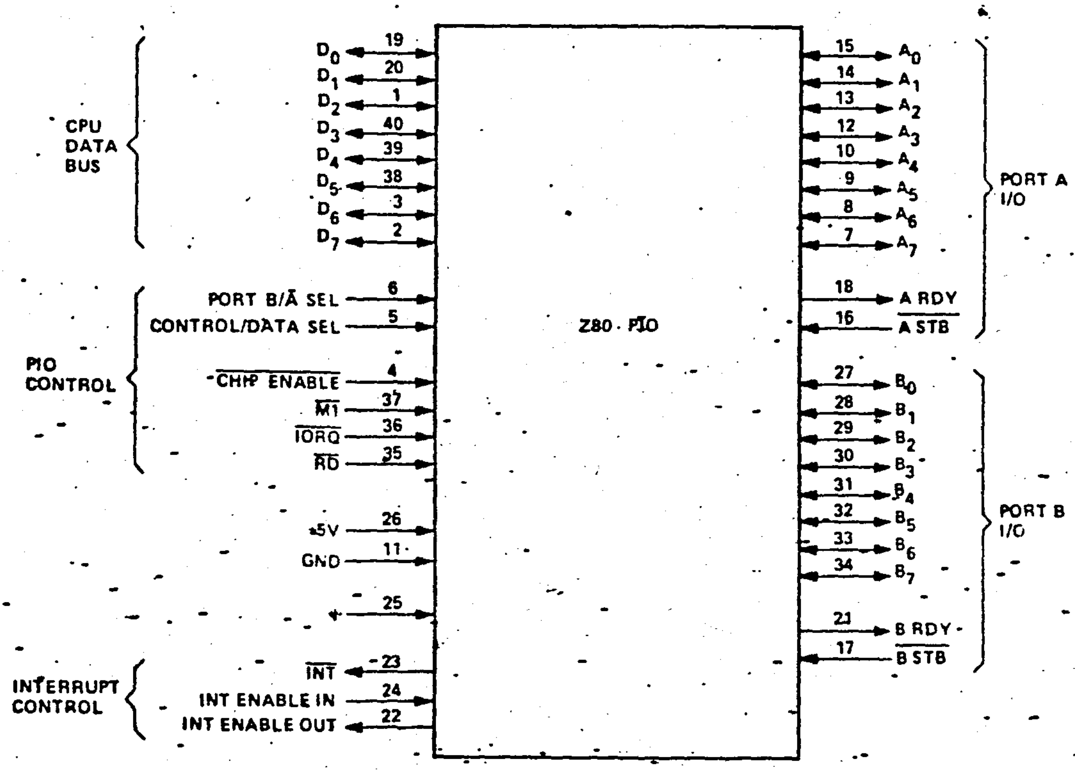


FIGURE 3.0-1
PIO PIN CONFIGURATION

2.0 PIO ARCHITECTURE

A block diagram of the Z80-PIO is shown in Figure 2.0-1. The internal structure of the Z80-PIO consists of a Z80-CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic. The CPU bus interface logic allows the PIO to interface directly to the Z80-CPU with no other external logic. However, address decoders and/or line buffers may be required for large systems. The internal control logic synchronizes the CPU data bus to the peripheral device interfaces (Port A and Port B). The two I/O ports (A and B) are virtually identical and are used to interface directly to peripheral devices.

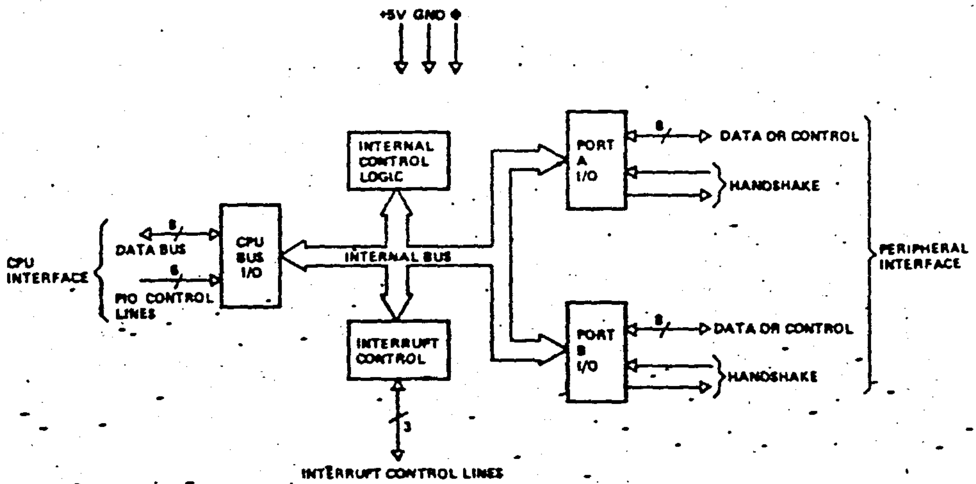


FIGURE 2.0-1
PIO BLOCK DIAGRAM

The Port I/O logic is composed of 6 registers with "handshake" control logic as shown in Figure 2.0-2. The registers include: an 8 bit data input register, an 8 bit data output register, a 2 bit mode control register, an 8 bit mask register, an 8 bit input/output select register, and a 2 bit mask control register.

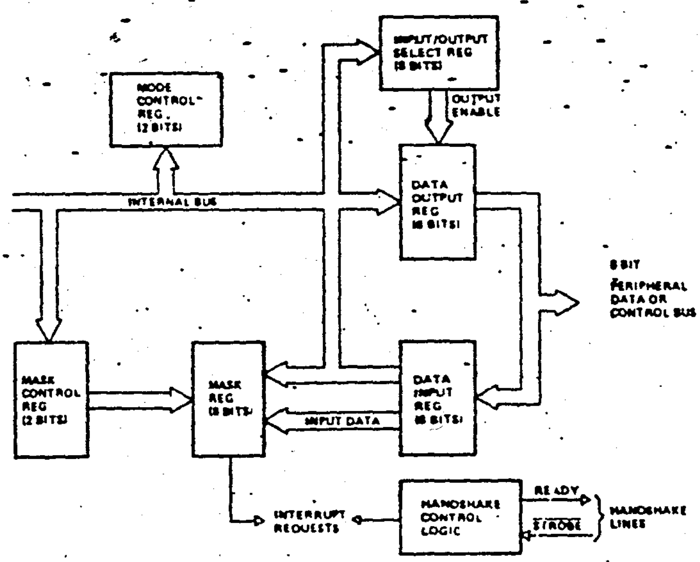


FIGURE 2.0-2
PORT I/O BLOCK DIAGRAM

The 2-bit mode control register is loaded by the CPU to select the desired operating mode (byte output, byte input, byte bidirectional bus, or bit control mode). All data transfer between the peripheral device and the CPU is achieved through the data input and data output registers. Data may be written into the output register by the CPU or read back to the CPU from the input register at any time. The handshake lines associated with each port are used to control the data transfer between the PIO and the peripheral device.

he 8-bit mask register and the 8-bit input/output select register are used only in the bit control mode. In this mode any of the 8 peripheral data or control bus pins can be programmed to be an input or an output as specified by the select register. The mask register is used in this mode in conjunction with a special interrupt feature. This feature allows an interrupt to be generated when any or all of the unmasked pins reach a specified state (either high or low). The 2-bit mask control register specifies the active state desired (high or low) and if the interrupt should be generated when *all* unmasked pins are active (AND condition) or when *any* unmasked pin is active (OR condition). This feature reduces the requirement for CPU status checking of the peripheral by allowing an interrupt to be automatically generated on specific peripheral status conditions. For example, in a system with 3 alarm conditions, an interrupt may be generated if any one occurs or if all three occur.

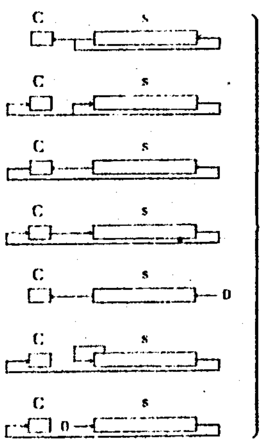
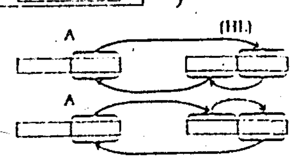
The interrupt control logic section handles all CPU interrupt protocol for nested priority interrupt structures. The priority of any device is determined by its physical location in a daisy chain configuration. Two lines are provided in each PIO to form this daisy chain. The device closest to the CPU has the highest priority. Within a PIO, Port A interrupts have higher priority than those of Port B. In the byte input, byte output or bidirectional modes, an interrupt can be generated whenever a new byte transfer is requested by the peripheral. In the bit control mode an interrupt can be generated when the peripheral status matches a programmed value. The PIO provides for complete control of nested interrupts. That is, lower priority devices may not interrupt higher priority devices that have not had their interrupt service routine completed by the CPU. Higher priority devices may interrupt the servicing of lower priority devices.

When an interrupt is accepted by the CPU in mode 2, the interrupting device must provide an 8-bit interrupt vector for the CPU. This vector is used to form a pointer to a location in the computer memory where the address of the interrupt service routine is located. The 8-bit vector from the interrupting device forms the least significant 8 bits of the indirect pointer while the I Register in the CPU provides the most significant 8 bits of the pointer. Each port (A and B) has an independent interrupt vector. The least significant bit of the vector is automatically set to a 0 within the PIO since the pointer must point to two adjacent memory locations for a complete 16-bit address.

The PIO decodes the RETI (Return from interrupt) instruction directly from the CPU data bus so that each PIO in the system knows at all times whether it is being serviced by the CPU interrupt service routine without any other communication with the CPU.

Operation	MNE	OP	Description For s =	n	I or R	A	r	(nn)	(BC)	(DE)	(HL)	(IX + e)	(IY + e)	Flags C Z S P/V
Move, bit	SET	b, s	Set b[th] bit of s			2/8	2/8				2/15	4/23	4/23	— — — —
	RES	b, s	Reset b[th] bit of s			2/8	2/8				2/15	4/23	4/23	— — — —
Move, register	LD	r, s	r ⟵ s	2/7		1/4	1/4				1/7	3/19	3/19	— — — —
	LD	s, r	s ⟵ r			1/4	1/4				1/7	3/19	3/19	— — — —
	LD	s, n	s ⟵ n			2/7	2/7				2/10	4/19	4/19	— — — —
	LD	A, s	A ⟵ s (see text for note on flags)	2/7	2/9	1/4	1/4	3/13	1/7	1/7	1/7	3/19	3/19	— — — —
	LD	s, A	s ⟵ A		2/9	1/4	1/4	3/13	1/7	1/7	1/7	3/19	3/19	— — — —
Block move	LDI		(DE) ⟵ (HL); Inc. DE, HL; Dec. BC							2/16				— x x ↕
	LDIR		(DE) ⟵ (HL); Inc. DE, HL; Dec. BC; Repeat until BC = 0							2/21BC				— x x 0
	LDD		(DE) ⟵ (HL); Dec. DE, HL, BC							2/16				— x x ↕
	LDDR		(DE) ⟵ (HL); Dec. DE, HL, BC; Repeat until BC = 0							2/21BC				— x x ↕
Input/Output	IN	A, (n)	A ⟵ Device(n).			2/10								— — — —
	IN	r, (C)	r ⟵ Device(C)			2/11	2/11							— ↕ ↕ ↕
	OUT	(n), A	Device(n) ⟵ A			2/11								— — — —
	OUT	(C), r	Device(C) ⟵ r			2/12	2/12							— — — —
Block I/O	INI		(HL) ⟵ Device(C); Inc. HL; Dec. B								2/15			— ↕ x x
	INIR		(HL) ⟵ Device(C); Inc. HL; Dec. B; Repeat until B = 0								2/20B			— 1 x x
	IND		(HL) ⟵ Device(C); Dec. HL, B								2/15			— ↕ x x
	INDR		(HL) ⟵ Device(C); Dec. HL, B; Repeat until B = 0								2/20B			— 1 x x
	OUTI		Device(C) ⟵ (HL); Inc. HL; Dec. B								2/15			— ↕ x x
	OTIR		Device(C) ⟵ (HL); Inc. HL; Dec. B; Repeat until B = 0								2/20B			— 1 x x
	OUTD		Device(C) ⟵ (HL); Dec. HL, B								2/15			— ↕ x x
	OTDR		Device(C) ⟵ (HL); Dec. HL, B; Repeat until B = 0								2/20B			— 1 x x
Increment	INC	s	s ⟵ s + 1			1/4	1/4				1/11	3/23	3/23	— ↕ ↕ ↕
Decrement	DEC	s	s ⟵ s − 1			1/4	1/4				1/11	3/23	3/23	— ↕ ↕ ↕
Complement A	CPL		A ⟵ $\overline{A}$			1/4								— — — —
	NEG		A ⟵ 00 − A			2/8								↕ ↕ ↕ ↕
Add/Subtract	ADD	s	A ⟵ A + s	2/7		1/4	1/4				1/7	3/19	3/19	↕ ↕ ↕ ↕
	ADC	s	A ⟵ A + s + C	2/7		1/4	1/4				1/7	3/19	3/19	↕ ↕ ↕ ↕
	SUB	s	A ⟵ A − s	2/7		1/4	1/4				1/7	3/19	3/19	↕ ↕ ↕ ↕
	SBC	s	A ⟵ A − s − C	2/7		1/4	1/4				1/7	3/19	3/19	↕ ↕ ↕ ↕
	DAA		Correct BCD addition and subtraction			1/4								↕ ↕ ↕ ↕

Z-80 INSTRUCTION SET

Operation	MNE	OP	Description	n	I or R	A	r	(nn)	(BC)	(DE)	(HL)	(IX + e)	(IY + e)	Flags C Z S P/V
AND	AND	s	A ⟵ A AND s	2/7		1/4	1/4				1/7	3/19	3/19	0 ↕ ↕ ↕
Exclusive-OR	XOR	s	A ⟵ A ⊕ s	2/7		1/4	1/4				1/7	3/19	3/19	0 ↕ ↕ ↕
OR (inclusive)	OR	s	A ⟵ A OR s	2/7		1/4	1/4				1/7	3/19	3/19	0 ↕ ↕ ↕
Bit test	BIT	b, s	Set (reset) Z flag if b[th] bit of s is equal to 0 (1)			2/8	2/8				2/12	4/20	4/20	— ↕ x x
Compare	CP	s	A − s (A not changed)	2/7		1/4	1/4				1/7	3/19	3/19	↕ ↕ ↕ ↕
Block compare	CPI		A − (HL); Inc. HL; Dec. BC			2/16								— ↕ x ↕
	CPIR		A − (HL); Inc. HL; Dec. BC; Repeat until A = (HL) or BC = 0			2/21BC								— ↕ x ↕
	CPD		A − (HL); Dec. HL, BC			2/16								— ↕ x ↕
	CPDR		A − (HL); Dec. HL, BC; Repeat until A = (HL) or BC = 0			2/21BC								— ↕ x ↕
Rotate	RLC	s	 Z, S, and P/V are not affected for s = A			1/4	2/8				2/15	4/23	4/23	↕ ↕ ↕ ↕
	RRC	s				1/4	2/8				2/15	4/23	4/23	↕ ↕ ↕ ↕
	RL	s				1/4	2/8				2/15	4/23	4/23	↕ ↕ ↕ ↕
	RR	s				1/4	2/8				2/15	4/23	4/23	↕ ↕ ↕ ↕
	SLA	s				1/4	2/8				2/15	4/23	4/23	↕ ↕ ↕ ↕
	SRA	s				1/4	2/8				2/15	4/23	4/23	↕ ↕ ↕ ↕
	SRL	s				1/4	2/8				2/15	4/23	4/23	↕ ↕ ↕ ↕
Rotate digits	RLD										2/18			— ↕ ↕ ↕
	RRD										2/18			— ↕ ↕ ↕

Definitions:
n = 8 bits of immediate data
r = any of the eight-bit CPU registers A, B, C, D, E, H, L
(nn) = the memory location pointed to by the second and third bytes of the instruction
(HL) = the memory location pointed to by the HL register pair

(Continued)

Figure A8-2. (Continued)

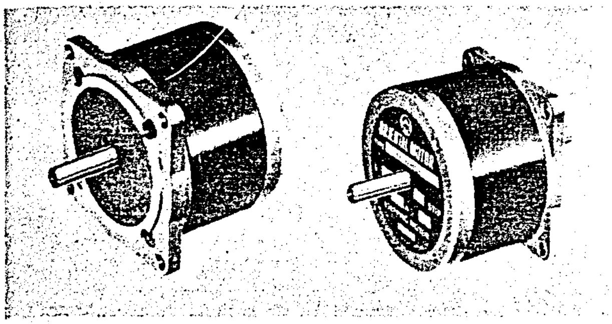
Operation	MNE	OP	Description	AF	SP	BC	DE	HL	IX	IY	Flags C Z S P/V
			For ss =	← Bytes/Cycles						→	
Move, register-pair	LD	ss, nn	ss ⟵ nn		3/10	3/10	3/10	3/10	4/14	4/14	— — — —
	LD	ss, (nn)	ss ⟵ (nn)		4/20	4/20	4/20	3/16	4/20	4/20	— — — —
	LD	(nn), ss	(nn) ⟵ ss		4/20	4/20	4/20	3/16	4/20	4/20	— — — —
	LD	SP, ss	SP ⟵ ss					1/6	2/10	2/10	— — — —
	PUSH	ss	Stack ⟵ ss	1/11		1/11	1/11	1/11	2/15	2/15	— — — —
	POP	ss	ss ⟵ Stack	1/10		1/10	1/10	1/10	2/14	2/14	— — — —
Exchange, register-pair	EX	DE, HL	DE ⟷ HL				1/4				— — — —
	EX	AF, AF'	AF ⟷ AF'	1/4							— — — —
	EXX		BC ⟷ BC'; DE ⟷ DE'; HL ⟷ HL'			1/4					— — — —
	EX	(SP), ss	Stack ⟷ ss					1/19	2/23	2/23	— — — —
Increment, register-pair	INC	ss	ss ⟵ ss + 1		1/6	1/6	1/6	1/6	2/10	2/10	— — — —
	DEC	ss	ss ⟵ ss − 1		1/6	1/6	1/6	1/6	2/10	2/10	— — — —
Double add/subtract	ADD	HL, ss	HL ⟵ HL + ss		1/11	1/11	1/11	1/11			↕ — — —
	ADD	IX, ss	IX ⟵ IX + ss		2/15	2/15	2/15		2/15		↕ — — —
	ADD	IY, ss	IY ⟵ IY + ss		2/15	2/15	2/15			2/15	↕ — — —
	ADC	HL, ss	HL ⟵ HL + ss + C		2/15	2/15	2/15	2/15			↕ ↕ ↕ ↕
	SBC	HL, ss	HL ⟵ HL − ss − C		2/15	2/15	2/15	2/15			↕ ↕ ↕ ↕

Operation	MNE	OP	Description	Bytes/Cycles	Flags C Z S P/V
Set carry	SCF		C ⟵ 1	1/4	1 — — —
Complement carry	CCF		C ⟵ $\overline{C}$	1/4	↕ — — —
Enable interrupts	EI		IFF ⟵ 1	1/4	— — — —
Disable interrupts	DI		IFF ⟵ 0	1/4	— — — —
Select interrupt mode	IM	0	Select 8080 interrupt mode	2/8	— — — —
	IM	1	Select ≈6800 interrupt mode (vector to address 0038H)	2/8	— — — —
	IM	2	Select Z80 interrupt mode (vector through table)	2/8	— — — —
Halt	HALT		Stop	1/4	— — — —
No operation	NOP		PC ⟵ PC + 1	1/4	— — — —
Jump unconditionally	JP	nn	PC ⟵ nn (jump anywhere)	3/10	— — — —
Branch unconditionally	JR	e	PC ⟵ PC + e (jump within −126 and +129 bytes from the present location)	2/12	— — — —
Jump conditionally	JP	cc, nn	PC ⟵ nn if cc = 1 for cc = C, NC, Z, NZ, M, P, PE, PO	3/10	— — — —
Branch conditionally	JR	cc, e	PC ⟵ PC + e if cc = 1 for cc = C, NC, Z, NZ	2/7, 12	— — — —
Jump indirect	JP	(HL)	PC ⟵ HL	1/4	— — — —
	JP	(ss)	PC ⟵ ss for ss = IX, IY	2/8	— — — —
Decrement and jump	DJNZ	, e	DEC. B; PC ⟵ PC + e if B ≠ 0	2/8, 13	— — — —
Call subroutine	CALL	nn	Stack ⟵ PC; PC ⟵ nn	3/17	— — — —
	RST	p	Stack ⟵ PC; PC ⟵ p for p = 00, 08, 10, 20, 28, 30, or 38 (hex)	1/11	— — — —
	CALL	cc, nn	Stack ⟵ PC; PC ⟵ nn if cc = 1 for cc = C, NC, Z, NZ, M, P, PE, PO	3/10, 17	— — — —
Return from subroutine	RET		PC ⟵ Stack	1/10	— — — —
	RET	cc	PC ⟵ Stack if cc = 1 for cc = C, NC, Z, NZ, M, P, PE, PO	1/5, 11	— — — —
Return from interrupt	RETI		PC ⟵ Stack; Reset interrupting peripheral interface chip	2/14	— — — —
	RETN		PC ⟵ Stack; Restore IFF as it was before this non-maskable interrupt	2/14	— — — —

1.8° STEP ANGLE HYBRID TYPE

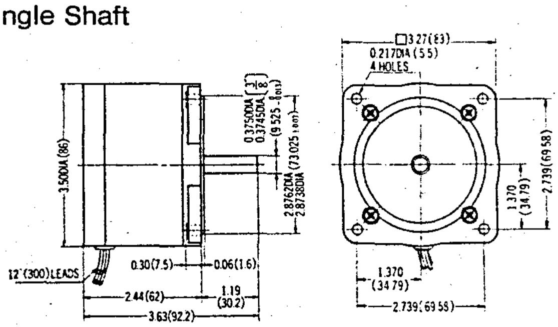
APPENDIX D.

PH296-□□□

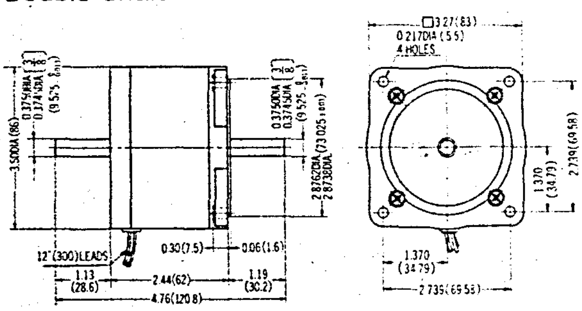


DIMENSIONS

Single Shaft

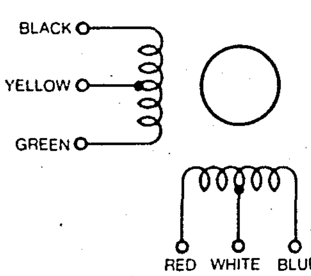


Double Shaft

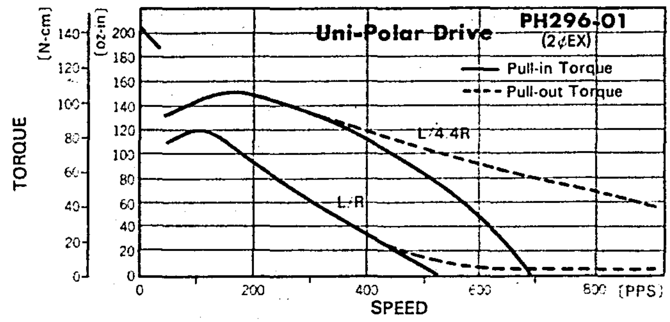


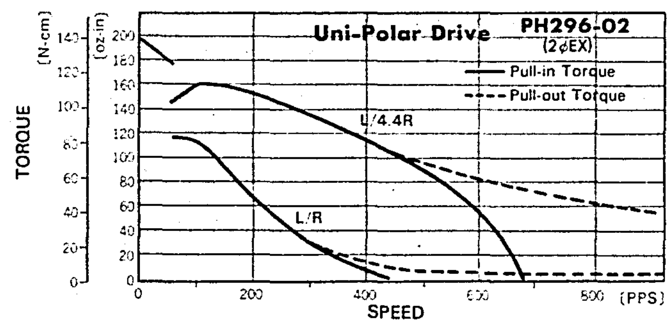
scale 1:4, unit=inch (mm)

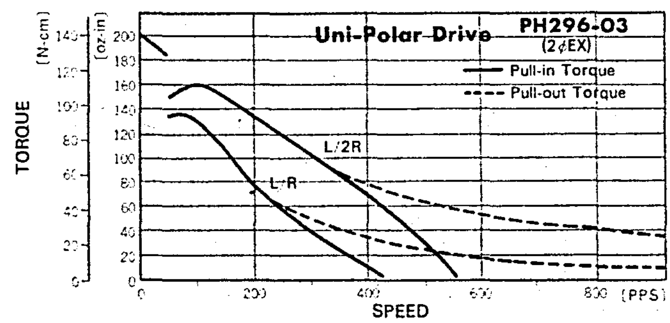
COLORS OF LEAD WIRES



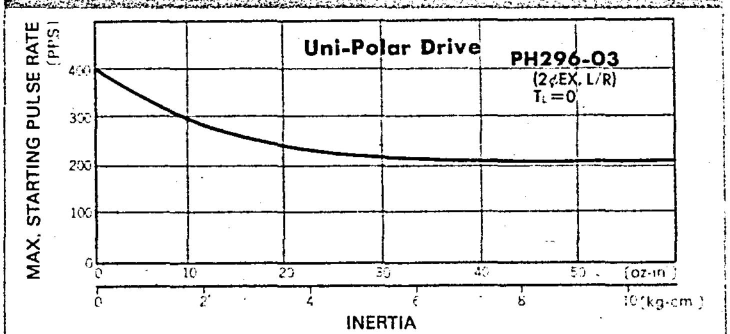
SPEED VS. TORQUE CHARACTERISTICS







INERTIA VS. STARTING PULSE RATE CHARACTERISTICS

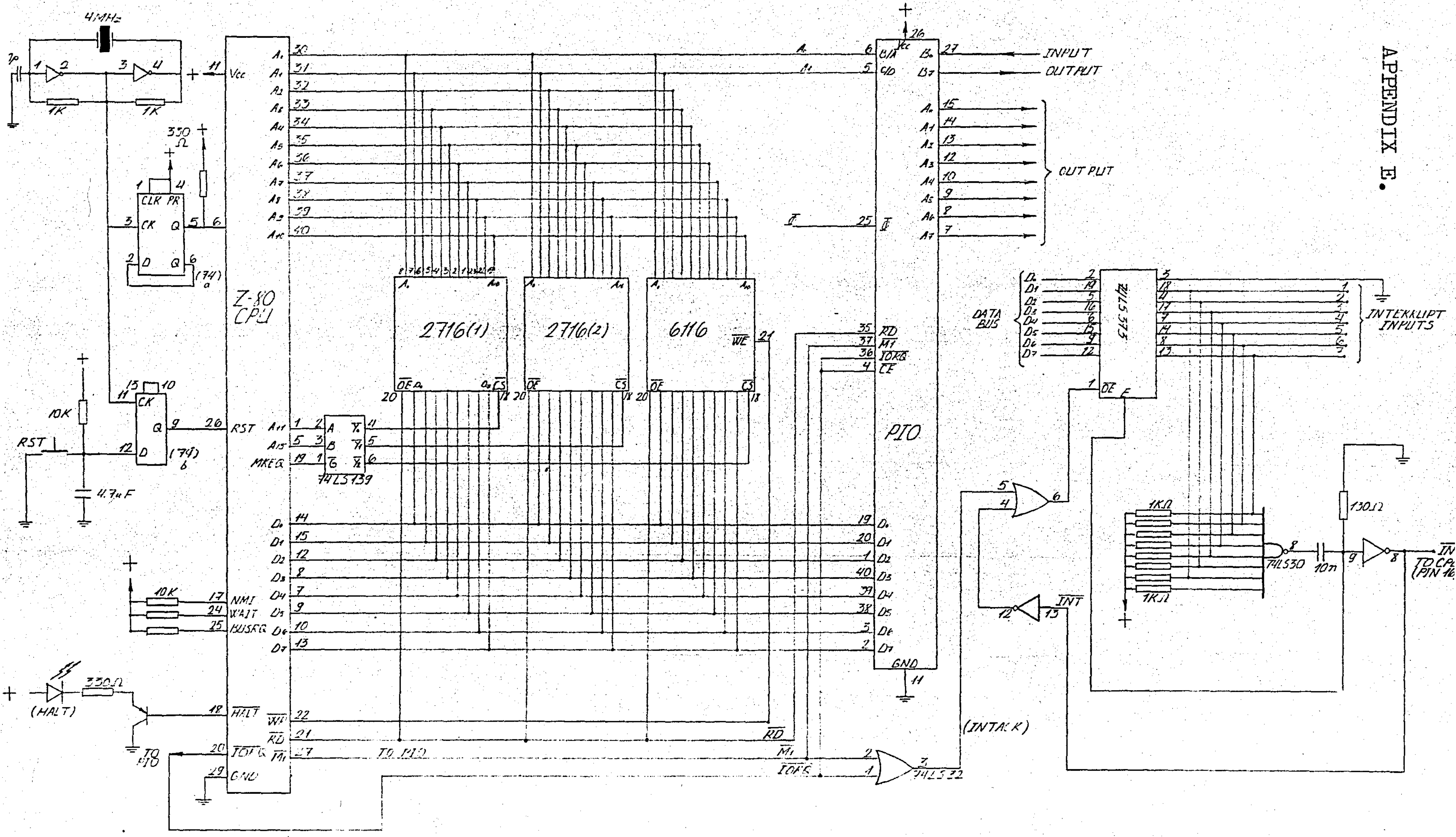


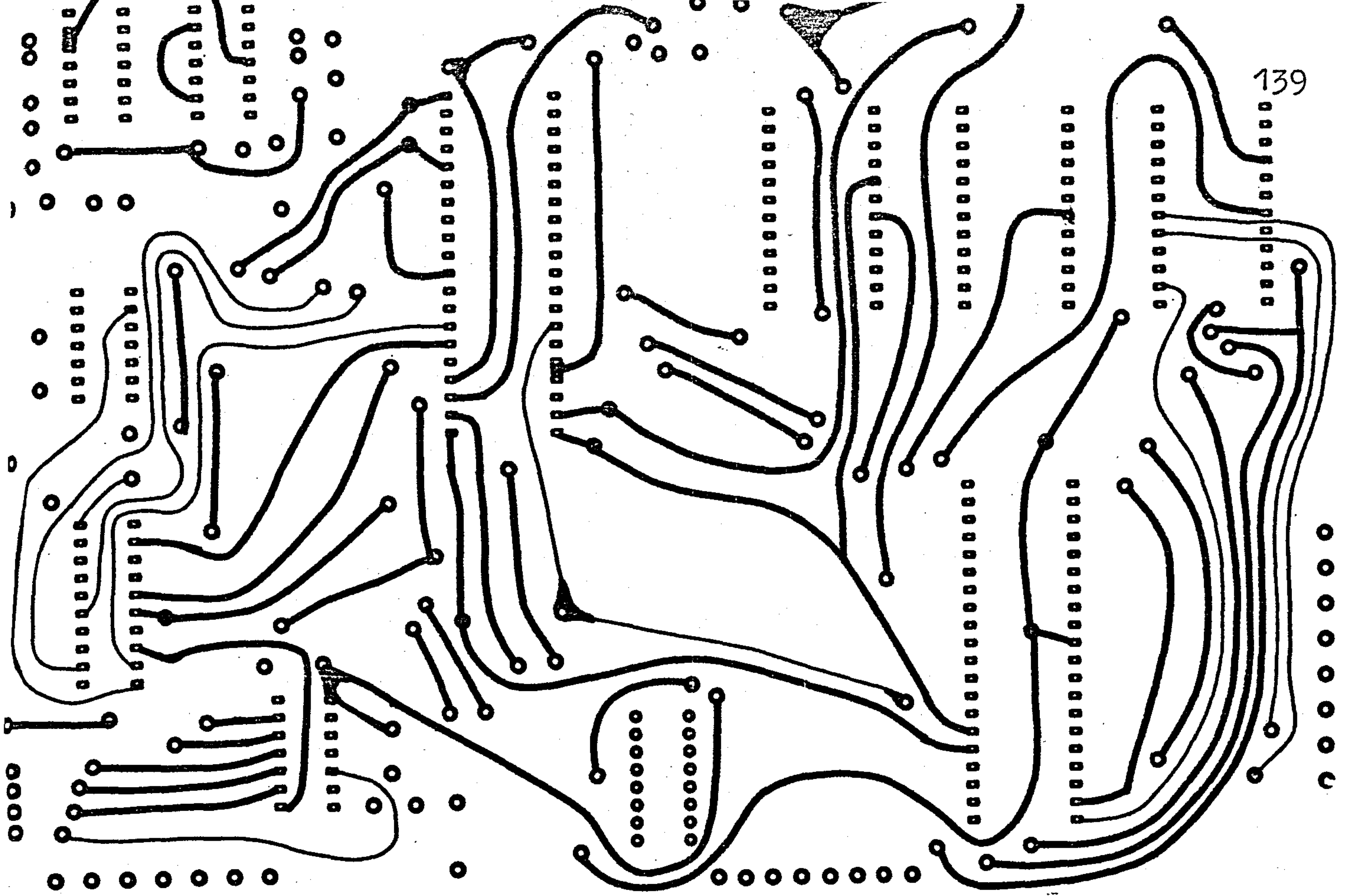
SPECIFICATIONS (2-phase full-step)

Motor type		Voltage	Current per phase	Holding Torque		Resistance per phase	Inductance per phase
Single Shaft	Double Shaft	V	A/phase	oz-in	N·cm	ohm/phase	mH/phase
PH296-01	**PH296-01B**	1.8	4.5	174	123	0.4	1.4
PH296-02	**PH296-02B**	5.5	1.25	174	123	4.4	14
PH296-03	**PH296-03B**	14	0.7	174	123	20	60

● Rotor Inertia 3.1 oz-in² (560g-cm²) ● Weight 3.3lbs (1.5 kg)

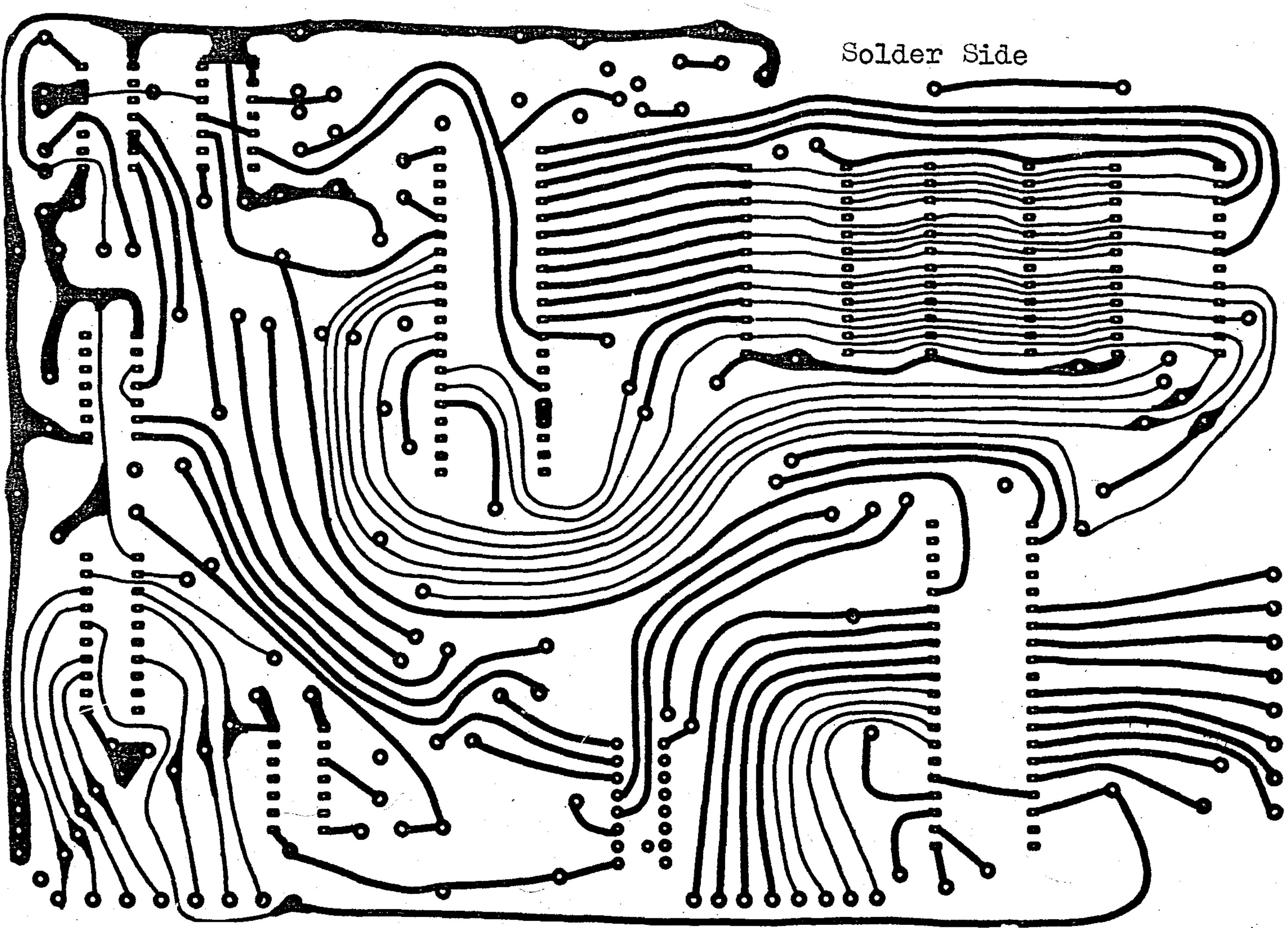
APPENDIX E.


4MHz
1K
1K
330 Ω
CLK PR
CK Q
D Q
(74) a
Z-80 CPU
Vcc
2716(1)
2716(2)
6116
WE
74LS139
10K
RST
4.7uF
(74) b
NMI
WAIT
BUSRQ
(HALT)
330Ω
HALT
TO PIO
GND
TO PIO
PIO
INPUT
OUTPUT
OUTPUT
DATA BUS
74LS373
INTERRUPT INPUTS
1KΩ
1KΩ
74LS30
10n
130Ω
INT
(INTACK)
74LS32

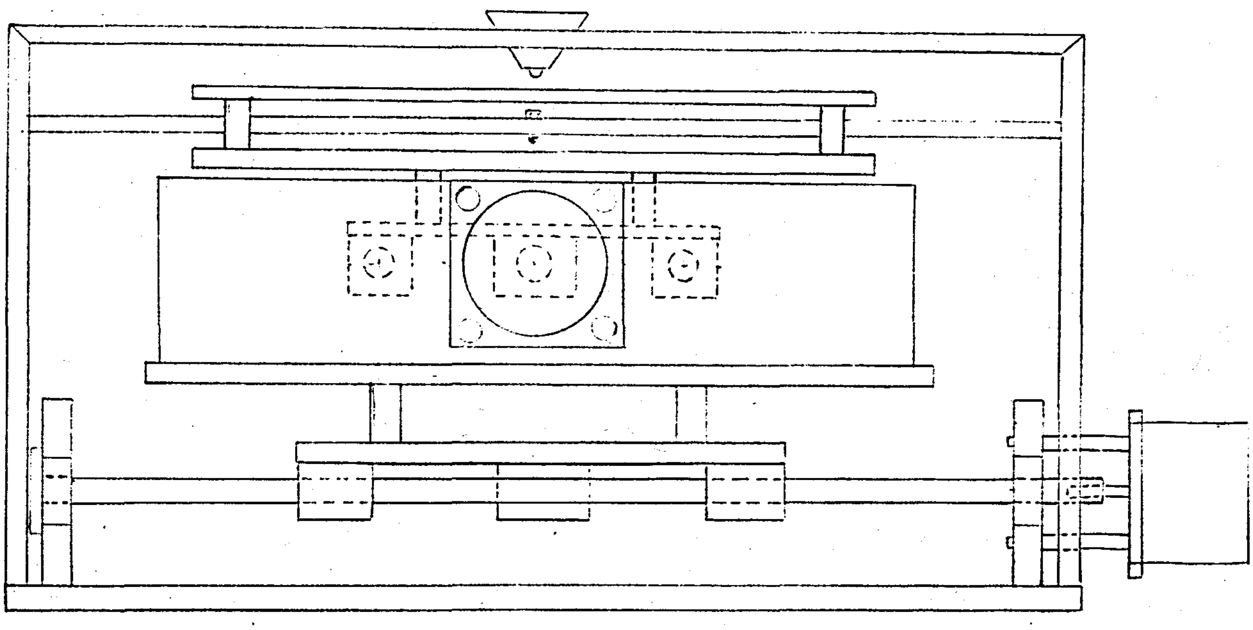


Component Side

Solder Side



APPENDIX F.



Drawing of mechanical scanner
(not to scale)

BIBLIOGRAPHY

1. Zilog Component Data Catalog,1981.
2. National Component Data Book,1976.
3. Telefunken Opto-Electronic Data Book,1978.
4. Motorola Transistor Data Manual,1979.
5. TI Linear and Interface Circuits Applications Data Book,1981.
6. SHARP Z-80 CPU/PIO/CTC Technical Manual,1978.
7. AIRPAX Stepper Motor Handbook,1979.
8. BODINE Electric Company Stepper Motors and Controls,1981
9. Slo-Syn DC Stepping Motors Handbook,1982.
10. Modulynx Motion Controls for Stepping Motors,1981.
11. Slo-Syn Stepping Motor Controls,1980.
12. Leventhal,Lance. Introduction to Microprocessors.Prentice-Hall Editions,1978.
13. Peatman,John B. Microcomputer-Based Design. Mc.Graw Hill Book Co.,1977.
14. Millman,Jacob. Microelectronics.International Student Edition,Mc.Graw-Hill Co.,1983.
15. Halkias,Christos. Electronic Applications. Mc.Graw-Hill Co.,1976