MICROPROCESSOR CONTROLLED DETECTION AND DRILLING OF PCB HOLES UTILIZING AN X-Y STAGE SCANNER

by

IBRAHIM OZYALCIN

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

MICRO-PROCESSOR CONTROLLED DETECTION AND DRILLING OF PCB HOLES UTILIZING AN X-Y STAGE SCANNER

APPROVED BY

Doç.Dr. Okyay Kaynak
(Thesis Supervisor)

Y.Doç.Dr. Ömer Cerid

Y.Doç.Dr. Vahan Kalenderoğlu

DATE OF APPROVAL: 31th July, 1984

To My Mother

ACKNOWLEDGEMENTS

It is indeed a great privilege to acknowledge those people who have extended their utmost support to me throughout both my undergraduate and graduate careers.

I would especially like to express my sincere gratitude and deepest appreciation for the understanding, co-operation and the extremely helpful guidance provided by my thesis advisor Doc. Dr. Okyay Kaynak.

I would like to thank Doc. Dr. Ömer Cerid for his valuable suggestions concerning the hardware of the system and Doc. Dr. Yorgo Istefanopulos, my undergraduate advisor, for his guidance, which he has extended to me throughout my course.

I am also very grateful to Doc. Dr. Vahan Kalenderoglu for his indispensable assistance on behalf of the mechanical design and the realization of the thesis.

ABSTRACT

The purpose of the thesis is to design and realise a system to detect and simulate the drilling of drilling-hole positions in printed circuit board masks making use of a stepper motor driven mechanical moving stage scanner under the control of microprocessor.

Determination of the print outline and the effective frame length, detection of the dots at proper drilling-hole positions, scanning of the dot mask in a meander pattern, generation of switching sequence of the unipolar stepper motors are under the control of software and achieved by interfacing the stage scanner, drive circuitry of the steppers and the detection system to a Z-80 microprocessor card.

Since all the control actions are performed by the microprocessor, the prototype can be considered as an intelligent system. The drilling part of the software minimizes the drilling process time making use of optimum path algorithm.

ÖZETÇE

Günümüzde, baskılı devrelerin delik delme işlemi devreler deneme safhasındayken miktarların azlığı nedeniyle önemli bir problem oluşturmakta ve kalıp-pin yöntemi masraflı olduğundan delikler el ile delinmektedir. Tezin amacı bu soruna çözüm getirmektir.

Yapilan prototipte baskılı devre üzerindeki delik yerleri ışık geçiren bir filme işaretlenerek bu filmin mikro işlemci denetiminde optik yöntemle taranmasıyla saptanmakta ve delme işlemi bir ışıkla simüle edilmektedir.

Sistemde kullanılan mekanik tezgah X ve Y yönlerinde adımlayıcı motorlarla hareket ettirilmekte ve tarama işleminde satır yöntemi kullanılmaktadır. Optik sistemin ve motorların tüm kontrolu Z-80 mikroişlemcisi ile gerçekleştirilmiş bir kart tarafından sağlanmaktadır. Geliştirilen sistemde yazılım, delme işleminde zamanı kısaltmak için optimum yol algoritmasını içermektedir.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

I. INTRODUCTION

The automatic drilling machines for drilling holes in printed circuit boards at positions stored on cassettes are widely used, but the determination of the hole positions often has to be done by hand. For a small series of printed circuit boards, generating the list of drilling positions is a very substantial part of the total production time. Another method but far more primitive, is drilling holes manually, which consumes more time and is less accurate than the aforementioned.

On the other hand, in the near future, Computer Aided Design will become very important, at which time the drilling positions will be known from the design process. However presently many of the layouts are made by hand. So the best way of handling holes of printed circuit boards is by the automatic determination of drilling positions, which is very cost-effective, where a small series of boards is concerned.

The developed prototype has mainly two aspects: One of them is the automatic scanning of the transparent layout with an optic sensor. The other is the simulation of the drilling procedure. For this purpose, a LED is used instead of a drill. Software provides all controls i.e, scanning, detection, the driving of the stepper motors and simulation of the drilling operation. Thus the system does not need any manual work for operation except pushing the start button and provides high accuracy due to precision stepper motors and high resolution optic sensor.

SYSTEM CONFIGURATION

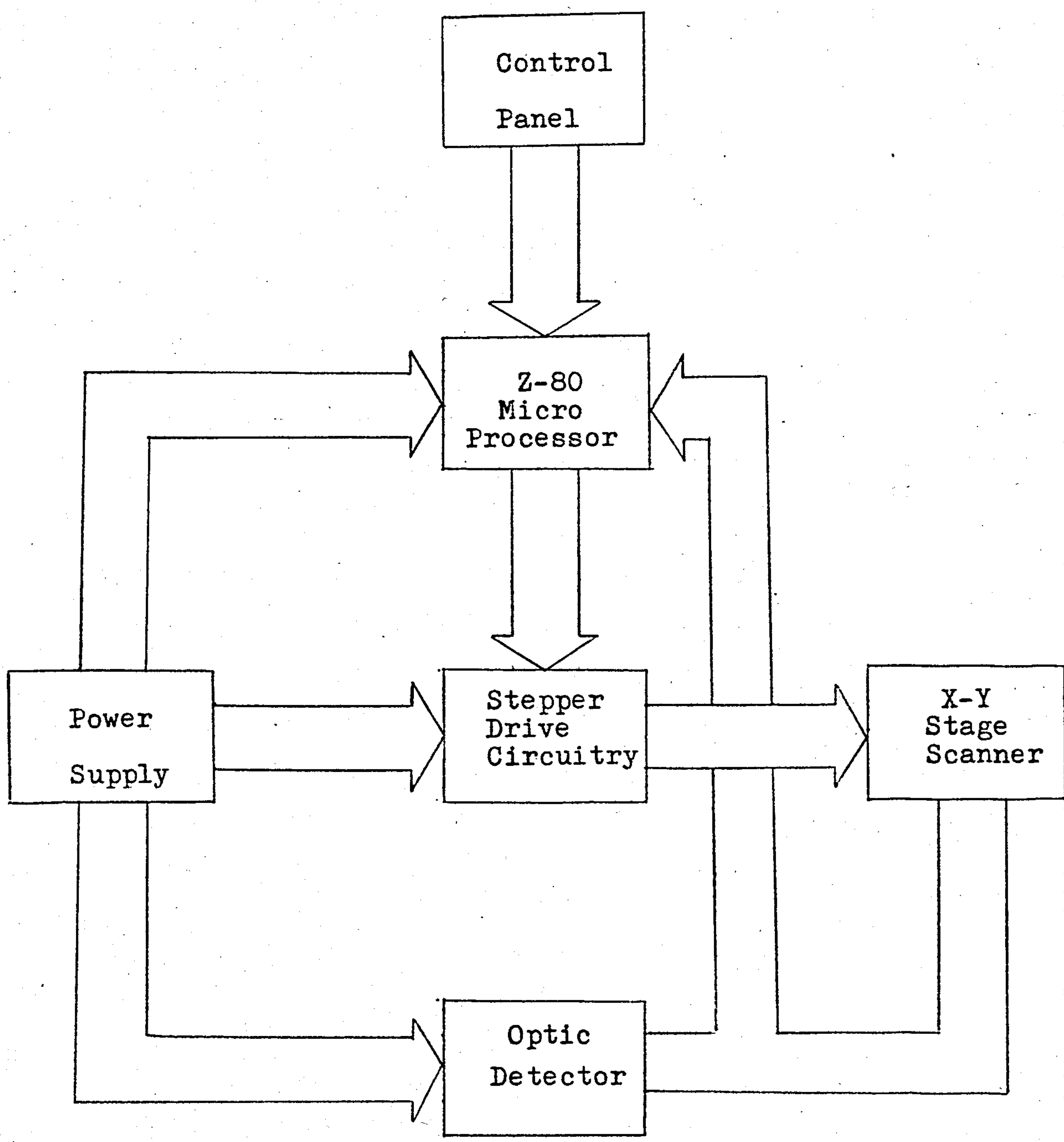


Figure 1.1

System has six main units. Those are :

1. Control panel
2. Z-80 microprocessor based card
3. Stepper drive circuitry
4. Mechanical assembly
5. Power supply unit
6. Optic detector

The above units will be explained in detail in the system hardware section.

II. STEPPER MOTORS

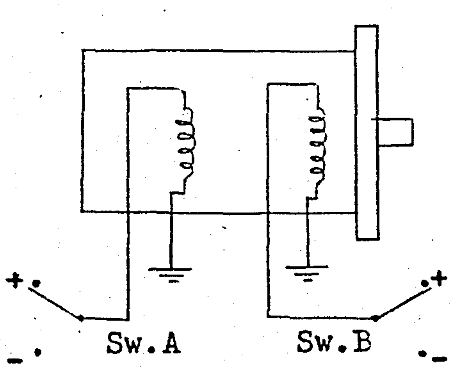
The stepping motor is a device which translates electrical pulses into mechanical movements.The output shaft rotates or moves through a specific angular rotation per each incoming pulse or excitation.This angle or displacement per movement is repeated precisely with each succeeding pulse translated by appropriate drive circuitry.The results of this precise,fixed and repeatable movement is the ability to accurately position.As opposed to a conventioal motor which has a free running shaft,the step motor shaft rotation is in fixed, repeatable,known increments.The stepping motor therefore allows load control ability of velocity, distance and direction.Initial positioning accuracy of a load being driven by a stepping motor is excellent.The repeatability(the ability to position through the same pattern of movements a multiple number of times) is even greater. The only system error introduced by the stepping motor is its single step error,and this is generally less than five percent of one step. Most significantly this error is non-cumulative, regardless of distance positioned or number of times repositioning takes place.

A. Construction and Operation

In a typical motor, electrical power is applied to two coils. Two stator cups formed around each of these coils with pole pairs mechanically displaced by half a pole pitch, become alternately energized north and south magnetic poles. Between the two stator-coil pairs the displacement is one fourth of a pole pitch.

The permanent magnet rotor is magnetized with the same number of pole pairs as contained by one stator-coil section. Interaction between the rotor and stator (opposite poles attracting and likes repelling) causes the rotor to move one fourth of a pole pitch per winding polarity change. A two phase motor with 12 pole pairs per stator-coil section would thus move 48 steps per revolution or seven and a half per step.

The normal electrical input is a four step switching sequence as shown in figure 2.1



CW ROT.

STEP	COILA	COILB
1	+	+
2	+	-
3	-	-
4	-	+
1	+	+

CCW ROT.

Figure 2.1

Continuing the sequence causes the rotor to rotate forward. Reversing the sequence reverses the direction of rotation. Thus, the stepper motor can be easily controlled by a pulse input drive which can be a two flip-flop logic circuit operated either open or closed loop.

Hereafter some specific names will be used for the stepper motors, so the below terminology will be needed :

1. Step Angle

The motor shaft rotates its specific angular increment each time the winding polarity is changed. This specific degree of rotation or increment is called the step angle. It is specified in degrees.

2. Step Accuracy

Defined as positional accuracy tolerance. This figure is generally expressed in percent and indicates the total error introduced by the stepping motor in a single movement. The error is noncumulative i.e it does not increase as additional steps are taken. In a linear positioning, with a resolution of .001 inches a three percent motor would introduce a maximum of .00003 inches error into the system. This total error would not accumulate nor increase with total distance moved or number of movements made. A particular step condition of the four step sequence repeatedly uses the same coil , magnetic polarity and flux path. Thus the most accurate movement would be to step in multiples of four since electrical and magnetic inbalances are eliminated. Increased accuracy also results from movements which are multiples of two steps. So, in positioning applications it is better to use two or four steps or multiples thereof for each desired measured increment.

3. Torque

The torque produced by a specific stepper motor depends on several factors :

i. The step rate
ii. The drive current supplied to the windings
iii. The drive design

a. Holding Torque. At standstill (i.e zero steps 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.

b. Residual Torque. The non-energized detent torque of a PM stepper motor is called residual torque. A result of the permanent magnet flux and bearing friction, it has a value of approximately one tenth 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.

4. Step Response

When given a command to take a step, the motor will respond within a specific time period. This step response or time for a single step is a function of the torque to inertia ratio of the motor and of the characteristics of the electronic drive system. Ratings are given for no-load conditions with time generally expressed in milliseconds. Single step response is shown in figure 2.2

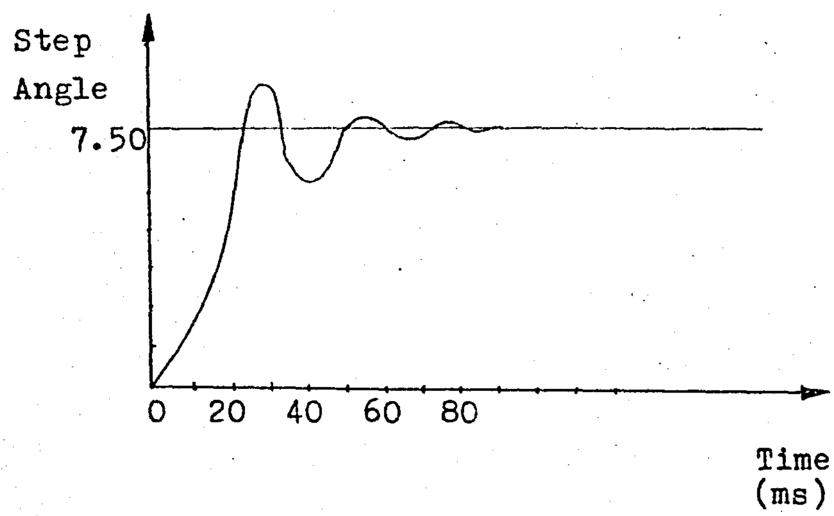


Figure 2.2

5. Resonance

Stepper motors are a spring-mass system and, as such, have certain natural frequency characteristics. When a motor's natural frequency or resonance is reached, an increase in the audible level of the motor's operation can be detected. In cases of severe resonant condition, the motor may lose steps and/or oscillate about a point. The frequency at which this resonance occurs varies, depending on the motor and the load. In many applications it may not occur to any perceptible degree; however, it is felt that the designer should realize that this condition can exist and specific facts about the resonant characteristics of an individual motor should be obtained from the manufacturer.

6. Translator

An electronic control with circuitry to convert pulses into the proper switching sequence, resulting in one motor step taken for each pulse received.

7. Preset Indexer

An electronic control which includes the translator function plus additional circuitry to control the number of steps taken as well as direction and velocity.

8. Ramping

The process of controlling pulse frequency to accelerate the rotor from zero speed to maximum speed as well as to decelerate the rotor from maximum speed to zero speed. Ramping increases the capability of driving the motor and load to higher speed levels, particularly with large inertial loads. A typical acceleration control frequency plot for an incremental movement with equal acceleration and deceleration time would be as shown in figure 2.3

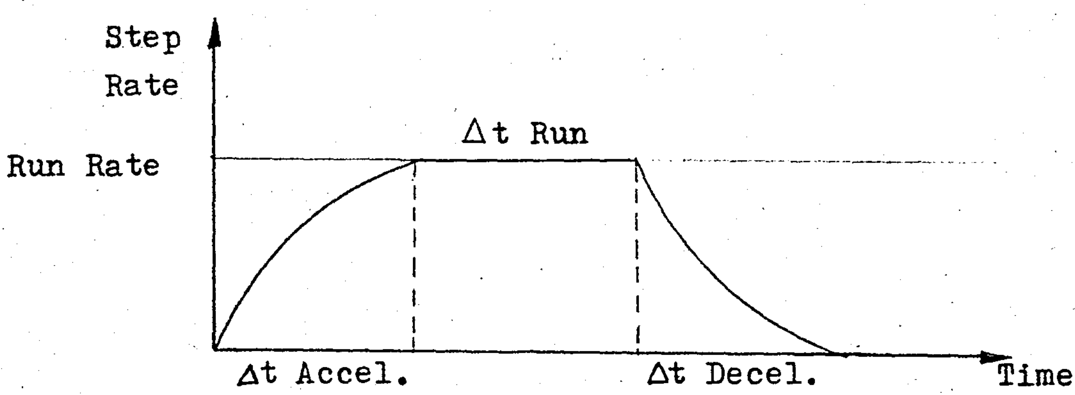


Figure 2.3

Ramping acceleration or deceleration control time allowed :

$$T_j \text{ (Torque mNm)} = J_t \cdot \frac{\Delta V}{\Delta t} \cdot K$$

Where J_t = Rotor inertia ($g.m^2$) plus load inertia

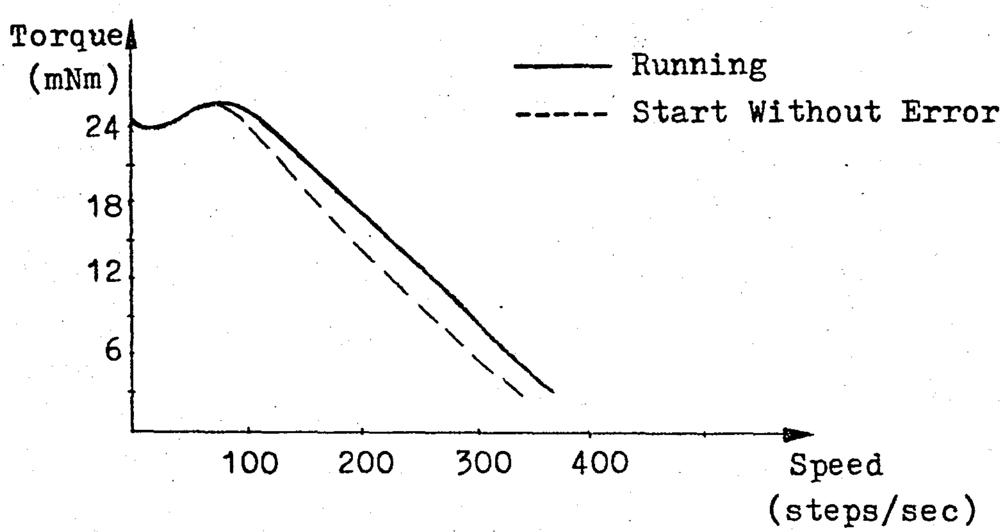
ΔV = Step rate change

Δt = Time allowed for acceleration in sec.

$K = \frac{2\pi}{\text{Steps/rev}}$

9. Start / Stop Without Error

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 pulse rate. The running curve is the torque available when the motor is slowly accelerated to the operating rate. It is thus the actual dynamic torque produced by the motor. This curve is sometimes called the slew curve. The difference between the running and the start without error torque curves is the torque lost due to accelerating the motor rotor inertia. A typical torque versus step rate characteristics curve is shown in figure 2.4.



Speed-Torque Curves

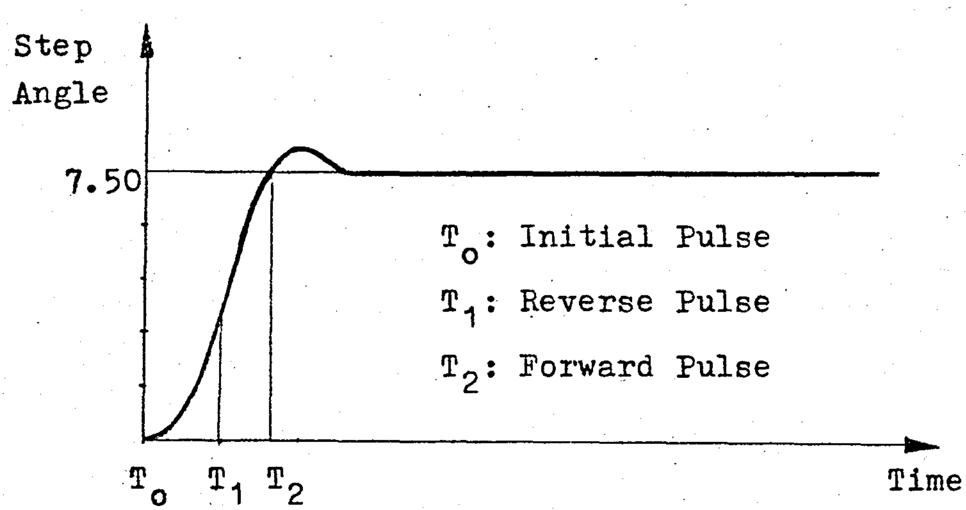
Figure 2.4

10. Slew Rate

An area of high speed operation where the motor can run unidirectionally in synchronism. However it cannot instantaneously start, stop or reverse. A stepping motor is brought up to a slewing rate using acceleration and is then decelerated to a stop under conditions where no step loss can be tolerated.

11. Damping

The reduction or elimination of step overshoot is defined as damping. It is used in applications where settling down time is important. In figure 2.5 electronically damped response is shown.



Damped Response

Figure 2.5

B. Drive Methods

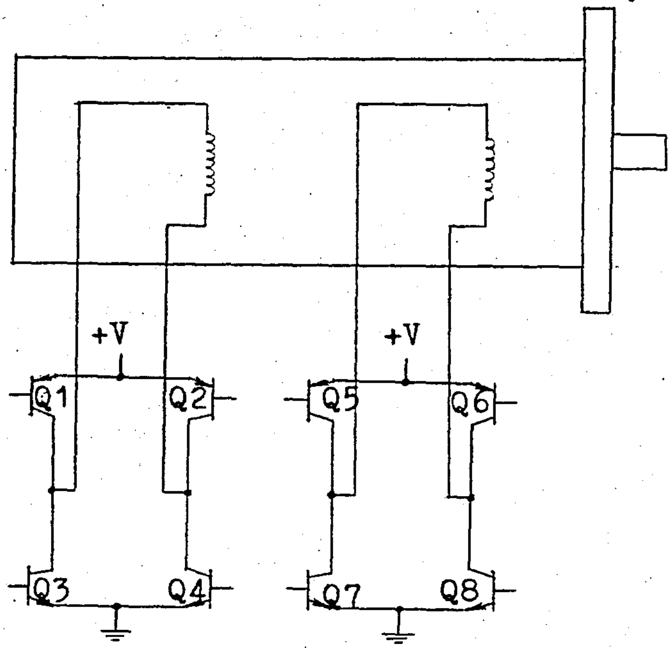
1. Bipolar

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

2. Unipolar

A unipolar winding has two coils wound on the same bobbin per stator half. Flux is reversed by energizing one coil or the other coil from a single power supply. The use of a unipolar winding, sometimes called a bifilar winding allows the drive circuit to be simplified. Not only are half as many power switches required (four vs. eight) but the timing is not as critical to prevent a current short through two transistors as is possible with bipolar drive. For a unipolar motor to have the same number of turns per winding as a bipolar motor, the wire diameter must be decreased and therefore the resistance increased. As a result unipolar motor have 30 percent less torque at low speeds. However at higher rates the torque outputs are equivalent.

SCHEMATIC BIPOLAR SWITCHING SEQUENCE



Normal 4 Step Sequence

Step	Q_1-Q_4	Q_2-Q_3	Q_5-Q_8	Q_1-Q_7
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

1/2 Step 8 Step Sequence (CW ROTATION ↓, CCW ROT. ↑)

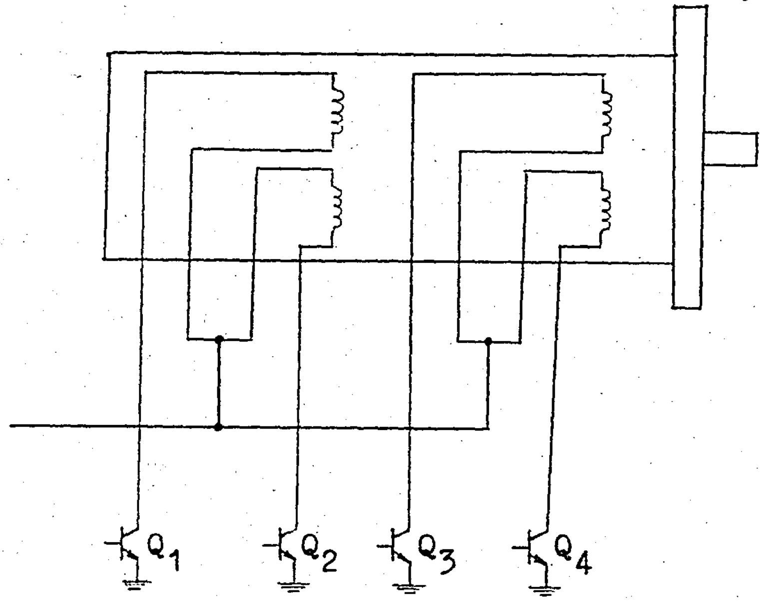
1	ON	OFF	ON	OFF
2	ON	OFF	OFF	OFF
3	ON	OFF	OFF	ON
4	OFF	OFF	OFF	ON
5	OFF	ON	OFF	ON
6	OFF	ON	OFF	OFF
7	OFF	ON	ON	OFF
8	OFF	OFF	ON	OFF
1	ON	OFF	ON	OFF

Wave Drive 4 Step Sequence

1	ON	OFF	OFF	OFF
2	OFF	OFF	OFF	ON
3	OFF	ON	OFF	OFF
4	OFF	OFF	ON	OFF
1	ON	OFF	OFF	OFF

Figure 2.6

SCHEMATIC UNIPOLAR SWITCHING SEQUENCE



Normal 4 Step Sequence

Step	Q_1	Q_2	Q_3	Q_4
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

1/2 Step 8 Step Sequence (CW ROTATION ↓ / CCW ROT. ↑)

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

Wave Drive 4 Step Sequence

1	ON	OFF	OFF	OFF
2	OFF	OFF	OFF	ON
3	OFF	ON	OFF	OFF
4	OFF	OFF	ON	OFF
1	ON	OFF	OFF	OFF

Figure 2.7

3. L / R Drive

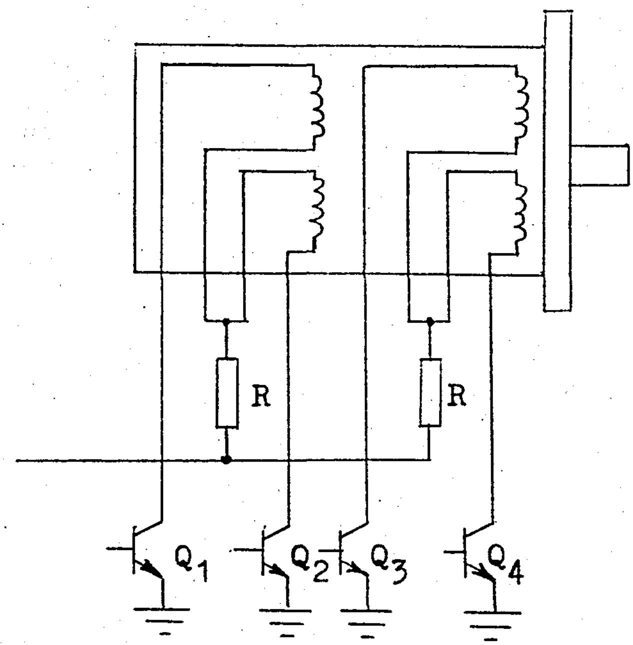
A motor operated at a fixed rated voltage has a decreasing torque curve as the frequency or step rate increases. This is due to the fact that the rise time of the coil limits the percentage of power actually delivered to the motor. This effect is governed by the inductance to resistance ratio of the circuit (L/R). Compansation for this effect can be by either increasing the power supply voltage to maintain a constant current as the frequency increases,or by raising the power supply voltage and adding a series resistor as is shown in Fig.2.8 . As the L/R is changed, more total power is used by the system. The series resistors, R, are selected for the L/R ratio desired. For L/4R they are selected to be 3 times the motor winding resistance with a:

$$\text{Watts rating} = (\text{Current per winding})^2 \times R$$

The power supply voltage is increased to 4 times motor rated voltage so as to maintain rated current to the motor. The power supplied will thus be 4 times that of a L/R drive. The unipolar motor which has a higher coil resistance thus has a better L/R ratio than a bipolar motor.

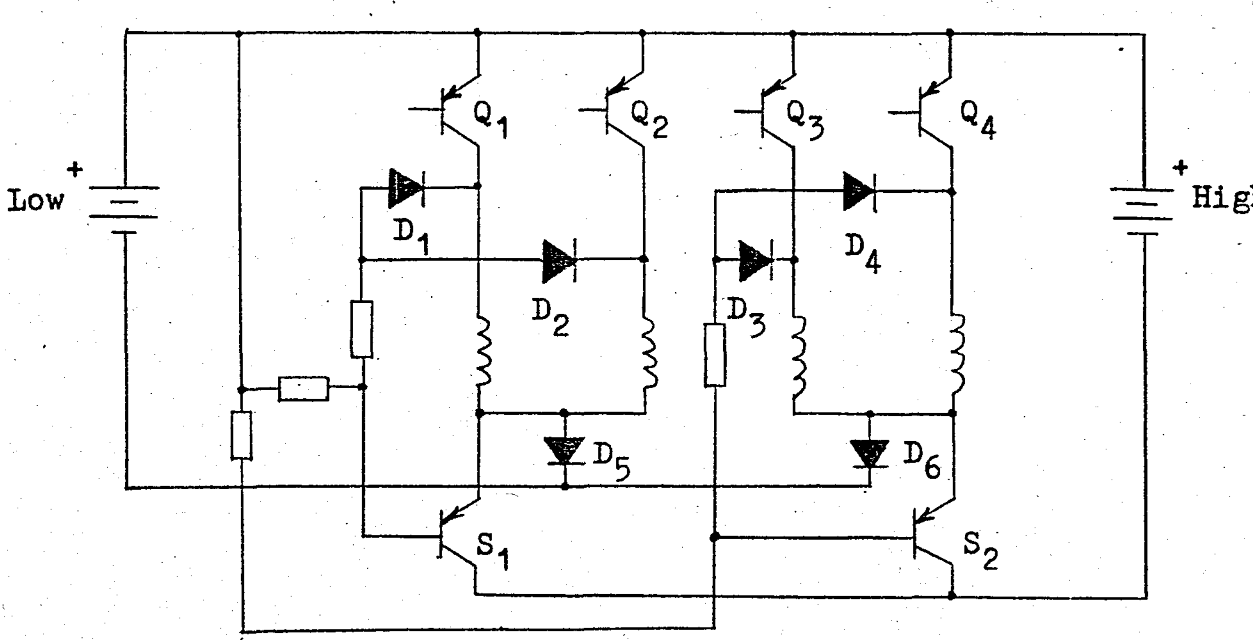
4. Bi-Level Drive

The bi-level drive allows the motor at zero step per second to hold at a lower than rated voltage, and when stepping to run at a higher than rated voltage. The high voltage may be switched on through the use of a current sensing resistor or a circuit which uses the inductively generated turnoff current spikes to control the voltage, is used.



L/R Drive

Figure 2.8



Unipolar Bi-Level Drive

Figure 2.9

5. Chopper Drive

A chopper drive maintains an average current level through the use of a current sensor which turns on a high voltage supply until an upper current value is reached. It then turns off the voltage until a low level limit is sensed where it turns on again. A chopper is best for fast acceleration and variable frequency applications. It is more efficient than a constant current amplifier regulated supply. The supply voltage of choppers is generally five to ten times the motor voltage rating.

6. Wave Drive

Energizing one winding at a time is called wave excitation. It 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 thirty percent. Within limits, the voltage can be increased to bring output power back to near rated torque value. The advantage of this type of drive is increased efficiency while the disadvantage is decreased step accuracy.

C. Characteristics of Steppers Used In The System

In the prototype system stepper motors are supplied from Oriental Motor Company. Specifications of the motors are as follows :

Type	:	PH296-03
Voltage	:	14 V
Current per phase	:	0.7 A/phase
Holding Torque	:	123 N-cm
Resistance per phase	:	20 ohm/phase
Inductance per phase	:	60 mH/phase
Rotor Inertia	:	560 g-cm^2
Weight	:	1.5 kg
Step Angle	:	1.8
Construction Type	:	Hybrid
No. of phase	:	2
Shaft Type	:	Single
Temperature Range	:	-10° C to 50° C
Temperature Rise	:	80 C or less
Insulation Type	:	Class B
Insulation Resistance	:	100 Mohm at 500 VDC
Dielectric Strength	:	Withstands in normal when impressing 0.5 kV at 60 Hz between the windings and the frame for one minute.

The other characteristics and graphs are in Appendix B.

III. SYSTEM HARDWARE

The developed system mainly consists of five units :

1. Mechanical assembly
2. Stepper motor drive circuitry
3. Z-80 Microprocessor based card
4. Power supply unit
5. Optic detector system

A. Mechanical Assembly

The realization in recent times of the need to implement efficient usage of manpower through improvement in pro - duction processes by automation has led to the development of devices such as the X - Y table.

The X - Y stage scanner facilitates the motion of the object to be positioned in either or both the X and Y axes. Obviously there are many methods to design a right angled motion table. Many factors need becarefully considered before deciding on a particular design. The factors are important in the sense that they determine the accuracy and reliability of the equipment. The motions are controlled by motors and the precision required in this application is obtained by using stepper motors which are driven by pulses that can easily be generated by drive circuits receiving single pulses from Z-80 microprocessor card.

Stepper motors give a very high precision motion depending on the lead - screw used to drive the table. The pitch of the lead - screw determines the amount of linear motion of the table per step of the motor. The main disadvantage of using lead - screw mechanism is backlash. Ofcourse there are some methods to prevent this disadvantage e.g using adjustable nut or spring system.

In the developed system backlash did not cause any problem so compensating methods were not applied.

B. Mechanical Design

1. Base Plate

The base plate supports the entire assembly. Aluminium is preferred due to strength and light property of this metal. Dimensions of this plate are : 260mm x 350mm x 4mm . The base plate supports two other plates at right angles and those plates have shafts and lead-screw mounted on them.

2. Intermediate Y - Plate

This is also made of aluminium. Its dimensions are as follows :

330 mm x 110 mm x 4 mm . This plate is placed on the linear motion bearings and the nut which are in turn mounted on shafts and lead-screw. This intermediate plate supports the other plate (Y - Table) which has dimensions : 420 mm x 195 mm x 4 mm and this supports the other two plates at right angles which are connected to the shafts and the lead screw.

3. Intermediate X - Plate

This plate is supported on the nut and the linear motion bearings which are mounted on shafts and the lead-screw. The dimensions of this plate are : 110 mm x 195 mm x 4 mm . This plate supports the top X-Plate which has dimensions 290 mm x 220 mm x 4 mm .

4. Linear Motion Bearings with Cylindrical Shafts

The best way is to use linear bearings to provide smooth movements. These bearings move along their shafts and are designed to give a smooth movement, which has extremely low friction. The X and Y tables are straight away mounted on these bearings and the lead-screws move the tables on those bearings giving perfect motion.

In order to have a robust system two shafts for the Y - table and two shafts for the X - table are used. Four bearings support each table. This means two bearings on each shaft. The shafts are made out of steel. The length of the X-table shafts is 400 mm and that of the Y-table is 320 mm. The diameter of both shafts is 16 mm.

Iko linear motion bearings (No. D-16) were selected and placed inside of aluminium blocks to support the X and Y tables. The shafts of both the X and the Y tables are supported by two other blocks at each end.

5. Lead — Screw and Its Nut

The most significant part of the X - Y table is the realization of the lead-screw and its nut. The pitch of the lead-screw determines the linear distance through which the X - Y stage scanner moves. The lead-screw moves in a nut and in the prototype, together they have virtually no backlash. The lead-screw has eight threads per inch, therefore the motor shaft needs to make eight revolutions to linearly displace the given X-Y table by one inch. Every revolution of the motor is made as a series of 200 steps i.e 1600 steps of the motor produce a linear displacement of one inch or one step produces 1/1600 inch displacement of the table i.e 0.015875 mm. This is the resolution of the prototype X-Y positioning table.

The nut is fixed under both the X and Y tables and the rotating lead-screw will move through the nut. Since the lead-screw motion is restricted to rotation only, it will cause the nut and hence the table to move linearly along the axis of the lead-screw. Since the length of the lead-screw is too long it is safer to give it some kind of radial bearing support. These bearings allow the lead-screws to rotate within them i.e the lead-screw will move within the bearing and at the same time will be supported by the bearing support. The lead-screw is suitably machined so that the support bearing fits on to the lead-screw perfectly. These bearings are fixed with some supports that are produced in the workshop of the university.

6. Motor Shaft, Lead - Screw Couplings

The power from the motor is transmitted to the lead-screw by coupling the motor shaft into the lead-screw and the combination is tightened by two screws which have housings on the lead-screw shaft.

The X and Y table motors are mounted to the block where two shafts and lead-screw are mounted. For this purpose four screws for each motor are used through the metal spacing units. At this point the important thing is the adjustment of the levels of the motor shaft and the lead-screw. In the developed system this levelling did not cause any problem due to design.

7. Glass Plate

This is the top part of the X-Y table which is placed on the X plate by using metal spacing units at all the four corners of the X-table. The aim of this spacing is to install the light source under the sample film to be scanned. So, the order of the units from top to bottom is : optic detector, sample film, glass plate and the X-plate. The distance between the glass plate and the X-table is 32 mm. The dimensions of the glass plate are 270 mm x 220 mm x 4 mm.

8. Operation

Every pulse input from the microprocessor card to the drive circuit makes the stepper motor rotate through one step or 1.8 degrees. Hence a total of 200 pulses, input to the drive circuit cause the motor shaft to make one revolution.

The scanning area of the X-Y table is defined by the length of the shafts. The developed system can scan an area which has an X-length of 220 mm and a Y-length of 150 mm.

The details related to the mechanical assembly are given in the appendix C.

C. Stepper Motor Drive Circuitry

The drive methods for stepper motors are explained in chapter 2 . It the second chapter. It can be easily understood that the drive circuitry affects the speed-torque characteristics of the steppers.

In order to have a proper drive, first the mechanical characteristics of the system must be considered, and then the selection of right stepper motors drive circuitry.

The best way to approach this problem, is to examine the speed-torque characteristics curves, and then settle the load specifications. If no acceleration is needed and the load is frictional, start-without-error curve should be used. The running curve, in conjunction with the equation :

$$T = I\alpha$$

where

T = Torque

α = Angular acceleration

I = Inertia

must be considered when the load is inertial and/or acceleration control is needed.

In the prototype system, X-Y stage scanner moves on linear motion bearings which have very little friction, hence the system does not need large amount of torque output from the motor, except that it needs high speed due to time consuming high resolution scanning.

Since the stepper motors are supplied before the realization of X-Y stage scanner, they are ordered from the powerful series. Thus it was a must to have a powerful drive stage to run these steppers at the required high speed.

The speed versus torque characteristics show that the steppers used in this project have the highest torque output at approximately hundred pulses per second. In addition to this, from the inertia versus starting-pulse-rate characteristics, it is seen that the maximum starting-pulse-rate is around two hundred pulses per second, and sometimes it is advisable to start with half of this speed in order not to lose any steps.

Under these circumstances, when all the drive methods are examined, the most suitable drive type seems to be the chopper drive.

In the chopper drive, current is sensed by a current sensing resistor which turns on a high voltage supply as soon as the current reaches 0.7 amperes and turns it off when the current falls below this value.

The prototype system as mentioned above needs acceleration, high speed and deceleration to save time. For this purpose it is appropriate to give some explanation about stepper's behaviour at variable frequency applications.

The torque output of the stepper motors needed to move the X-Y table is directly proportional to the current that passes through the coils of stepper motor. So, at high speed applications current cannot easily reach its rated value when normal supply voltage is used. It is very logical from the basic equation $V = I Z$ that, in order to have high currents through a coil showing an impedance Z, voltage must attain higher values.

The rise of current depends on the L/R time constant. This can be expressed by the following formula :

$$I = I_f \left(1 - e^{-\frac{t_1 R}{L}} \right)$$

where

I shows the value of current at time t_1 , and

I_f represents the final value of current

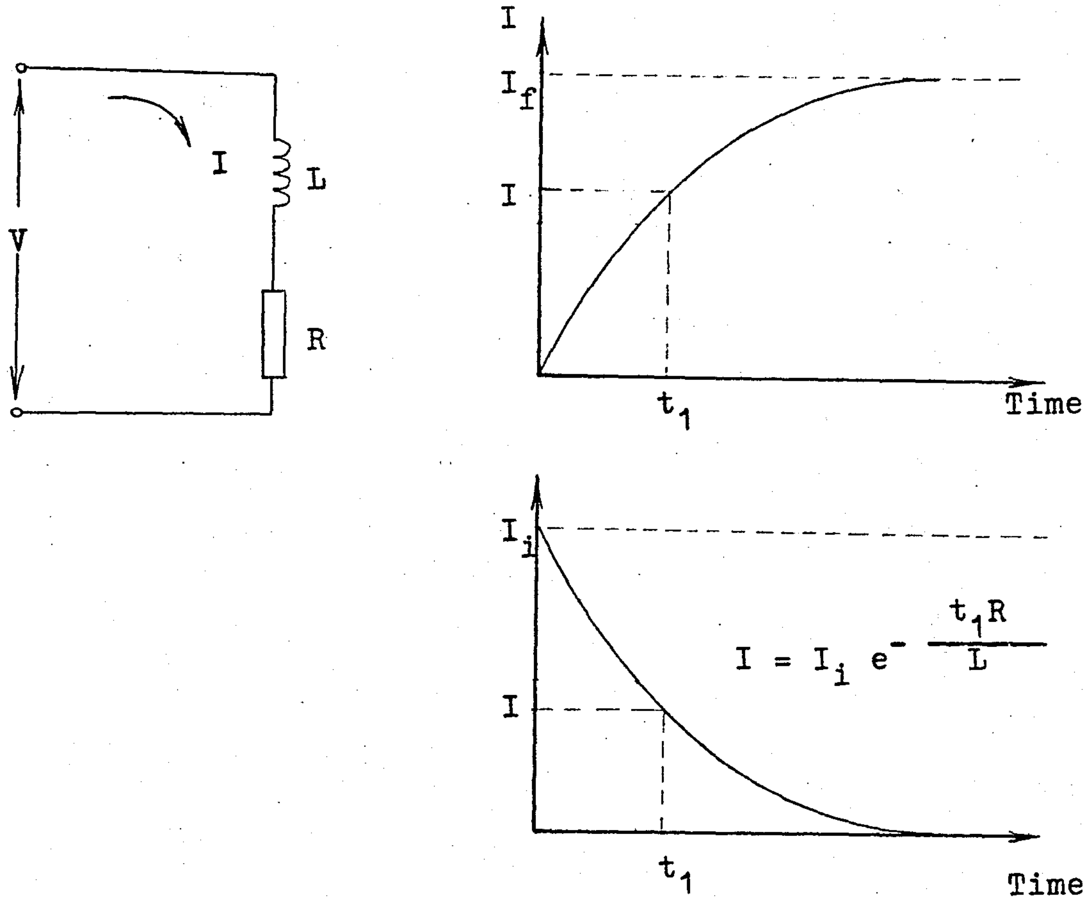


Figure 3.1

Current reaches around 60 percent of its final value at time t=L/R during charging. Discharge process can be viewed as the opposite of charging process. For example in the prototype the rated voltage of stepper motors is 14 V and their resistance per phase is 20 ohms and inductance per phase is 60 mH.

According to these data, the current reaches its 60 percent of its final value at time t=L/R .

$$I_f = \frac{V}{R} = \frac{14}{20} = 0.7 \text{ A}$$

$$I = 0.42 \text{ A} \quad \text{at } t=L/R$$

On the other hand, if high voltage (e.g 42 V in the prototype) supply is used :

$$I_f = \frac{42}{20} = 2.1 \text{ A}$$

$$I = 1.26 \text{ A} \quad \text{at } t=L/R$$

It can be seen from these rough calculations that the current can reach three times high a value when it is driven from the higher voltage supply in spite the fact that the time interval does not change i.e $t=L/R$.

Considering all these advantages a modified version of chopper drive is used. It somewhat behaves like a bi-level drive which is also mentioned in chapter two.

In the developed system :

$V = 42$ V

$R = 20$ ohm

$L = 60$ mH

$I = 0.7$ A

So the time required to attain a current of 0.7 A is 0.0012 s. As a result , a speed of 835 pulse per second is obtained.

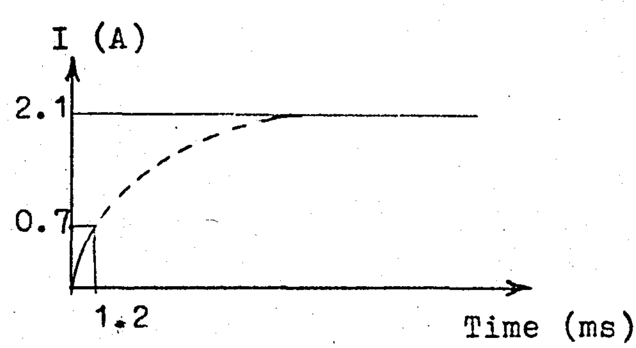


Figure 3.2

Referring to the figure 3.3 where two different supply levels are seen, the high voltage supply (42 V) is connected to the stepper motor windings through Q_1 and Q_2 power transistors. These transistors stay on, till the final value of winding current (0.7 A) is reached. Then Q_{11}, Q_{12} conduct and Q_1 and Q_2 go into cut off and some power is dissipated through the R_1 and R_2. At this time, the second low voltage (18 V) supply takes over the current and dissipation of Q_1 and Q_2 is avoided.

The diodes D_1 and D_3 are used to prevent the reverse biasing of Q_1 and Q_2 . There can be seen two fuses (F_1,F_2) to protect the system. D_2 (zener), D_4 (zener), D_5, D_6, D_7, D_8 are used for voltage suppression. Also D_9, D_{10},D_{11},D_{12}prohibits the negative pulses that may come from the windings due to the magnetic field created by the rotating permanent magnet rotor. Whenever winding current is turned off, a high voltage inductive spike will be generated which could damage the drive circuit switching transistors.

The above mentioned inductive spike results as per the inductor voltage equation :

$$V = L \frac{dI}{dt}$$

where a high value of rate of change of current is encountered due to an extremely small switching time interval.

The normal method used to suppress these spikes is to put a diode (Free Wheeling diode) across each winding. This, however, will reduce the torque output of the motor unless the voltage across the switching transistors is allowed to build up to at least twice the supply voltage. The higher this voltage the faster the induced field and current will collapse and thus the better performance. In the prototype circuit 120 V zener diodes are used for this purpose. Diodes D_{13} and D_{14} avoid the reverse currents to the low level supply (18 V).

The second supply is set to 18 V, since there will be a 14 V drop on the motor windings, 0.7 V on the diodes (D_{13} and D_{14}), 2.5 V collector to emitter voltage drop on the transistors (Q_4, Q_5, Q_8, Q_9) and 0.7 V on the current sensing resistors (R_9 and R_{10}).

The pulse sequence from the microprocessor card taken as output from the peripheral device (PIO) is fed to the bases of Q_3, Q_6, Q_7, Q_{10} through the base resistors R_3, R_4, R_5 and R_6.

Ofcourse there are some limitations due to the stepper motor characteristics beside the drive method. 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 motor can be driven is limited by hysteresis and eddy current losses. At some rate, the heating effects of these losses limits any further effort to get more speed or torque output by driving the motor harder.

The realised driving cicuitry improved the speed characteristics of the stepper motors which are not very suitable for high speed applications. During scanning a speed of 835 pulse per second is reached by means of this modified chopper drive circuit.

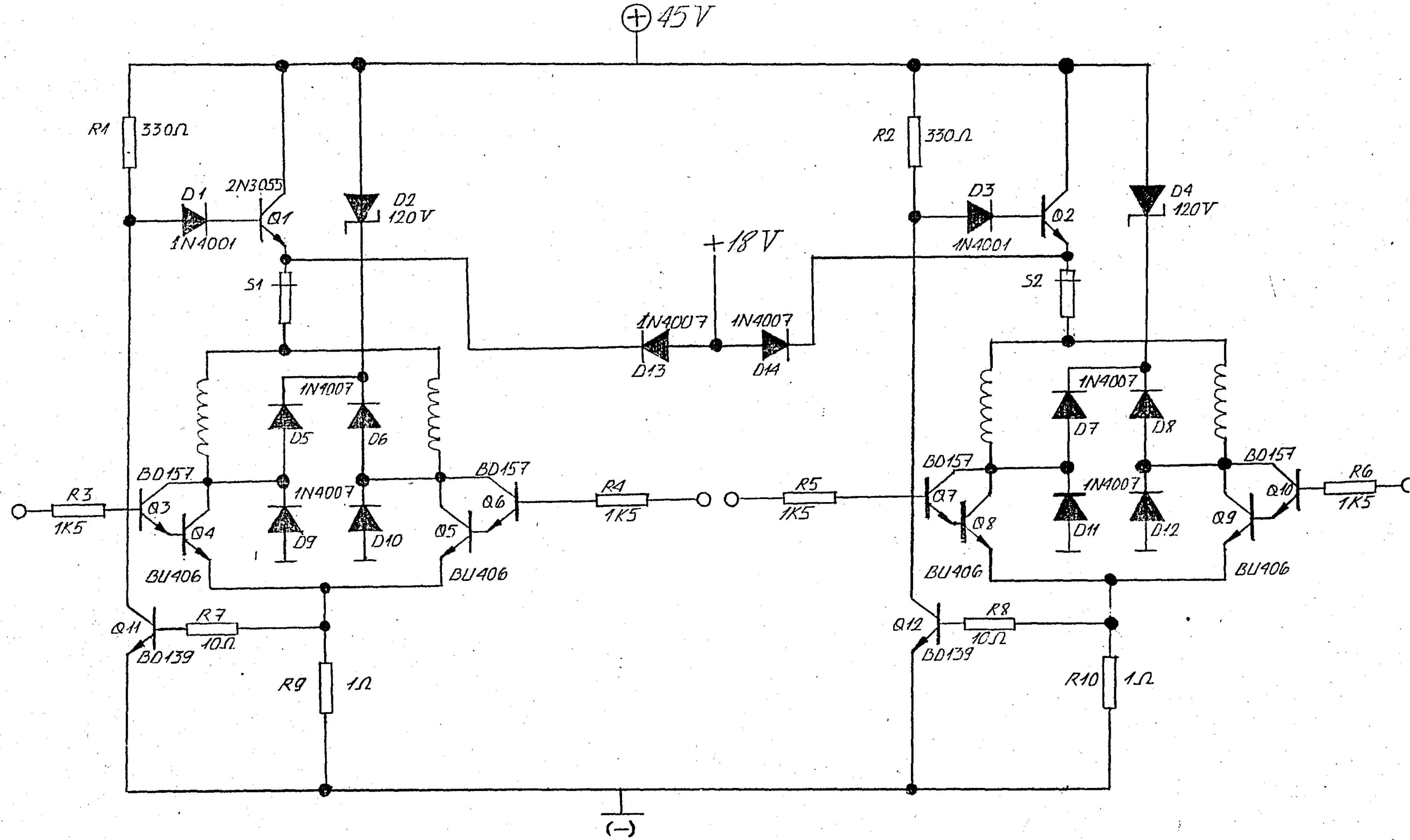


Figure 3.3 Stepper Motor Driving Circuitry

D. Z-80 Microprocessor Based Card

This card consists of four main parts :

1. Z-80 CPU
2. Z-80 PIO
3. Memory Section
4. Other IC's

1. Z-80 CPU

All controls are carried out by this unit. It works with a 2 MHz clock. This frequency obtained from a D-type flip-flop which divides the 4 MHz crystal frequency. The Reset input of the CPU is used through a D-type flip-flop also. The non-maskable interrupt, Wait and Bus Request inputs are connected to the V_{cc} by the pull-up resistors. The Halt state is shown by driving a PNP transistor which is used to turn on the LED.

2. Z-80 PIO

The Z-80 parallel input-output circuit is a programmable , two port device which provides a TTL compatible interface between peripheral devices and the Z-80 CPU. The CPU can configure the Z-80 PIO to interface with a wide-range of peripheral devices. In the prototype an 8 bit output is used to give the step sequence of motors, 4 least significant bits are controlling the X-motor, while the most significant

4 bits controlling the Y-motor.

The other port (B) is used as an input port during the scanning process and as an output port while simulation of the drill process. The circuit shown in the figure 3.4 is used for simulation making the LED flash when the detector (simulating the drill) comes on to the dot which should be drilled.

3. Memory

System uses three memory units. Two of them are 2k x 8 bit EPROMS (2716 type) and the other is a 2k x 8 bit static RAM (6116 P-3 type).

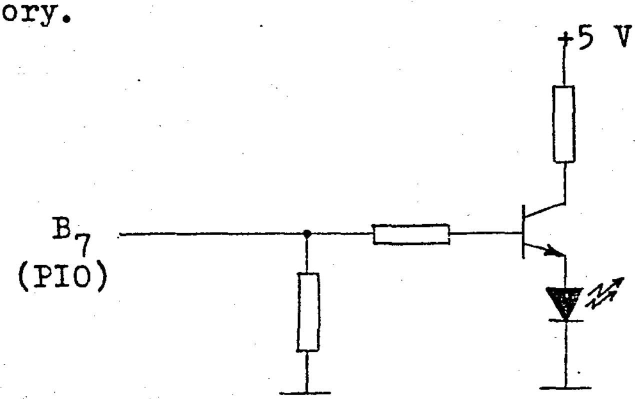
The storage size of the 2k RAM limits the card size to 170 mm x 140 mm despite that the maximum scanning area is 220 mm x 150 mm (X,Y) mechanically. Along with the hole position storage, a part of the RAM acts as a stack and also as a scratch-pad for the program.

The memory addresses of the system are decoded as follows :

EPROM 1 .... ØØØØ - Ø7FF
EPROM 2 .... Ø8ØØ - ØFFF
RAM .... 8ØØØ - 87FF

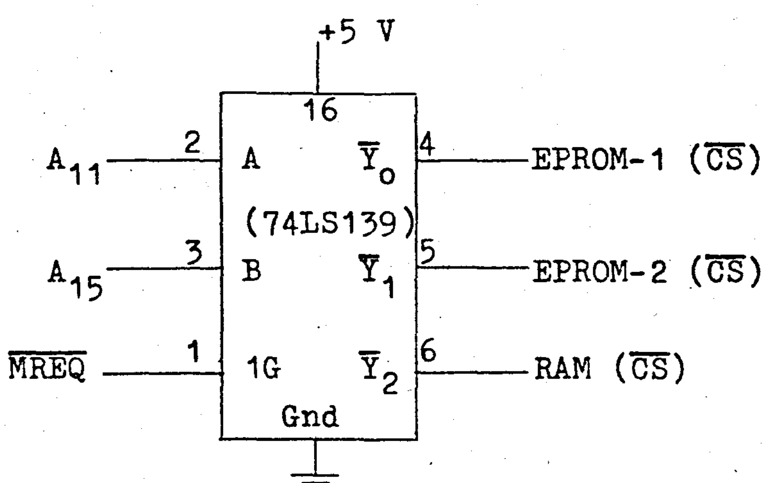
This decoding is performed by one half of a 74LS139 (Decoder/Demux). Inputs of this IC are supplied from the address pins (A_{11} and A_{15}) and $\overline{MREQ}$ pin of CPU. $\overline{Y}_0$, $\overline{Y}_1$, $\overline{Y}_2$ outputs of the decoder select the EPROM 1, EPROM 2 and RAM respectively, as shown in the figure 3.5

The output enable pins of EPROMS and RAM are connected to the $\overline{RD}$ pin of CPU. The RAM output enable pin has connection with the $\overline{WR}$ pin of CPU to perform writing operations into the memory.



Drill Simulator

Figure 3.4



Address Decoding

Figure 3.5

4. Other IC's

The other IC's perform the interrupt action as follow System interrupt mode is set to 2 to have control inputs for the X-Y table movements. In mode 2 system creates 16 bit starting address of the service routine. The programmer maintains a table of 16 bit starting addresses for every interrupt service routine. This table may be located anywhere in the memory. When an interrupt is accepted a 16 bit pointer must be formed to obtain the desired interrupt service routine's starting address from the table. The upper 8 bits of this pointer are formed from the content of the I-register. The I-register must have been previously loaded with the desired value by the programmer. In the developed board , the lower 8 bits of the pointer are supplied by the interrupting switches. The point to note is that only 7 bits can be used leaving the least significant bit zero. This is needed since the pointer is used to get two adjacent byte to form a complete 16 bit service routine's starting address and the addresses must always start in even locations.

In the Z-80 hardware layout given in the appendix, this interrupt mode works as follows : When one of the switche connected to interrupt inputs is drawn to ground, the NAND gate output goes high and sends a 1.3 microsecond interrupt pulse through the capacitor and resistor network, meanwhile the Octal Transparent Latch (74LS373) is enabled and it latches the data on its input. Meanwhile the CPU generates an $\overline{\text{INTA}}$ signal which enables the latch output, hence the data is loaded on the data bus. As mentioned before, this data form the lower 8 bits of the 16 bit starting address of the service routine. Then the CPU executes the program from that address on.

E. Power Supply Unit

This unit is made up of four main parts :

1. The High Voltage Supply

This supply uses 32 VAC input. After rectification and capacitive filtering, this unit is connected to the high voltage pin of the drive card. For this part of the supply no regulator is used since inductive spikes that are created due to high speed switching of the drive card, may damage the regulator circuit besides that there is no need for such regulation as far as drive cicuit structure is concerned. The output of this supply is 45 V at no load conditions, and at loading it falls to 42 V that is sufficient for the drive card.

2. The Low Voltage Supply

After rectification and capacitive filtering 25 VDC is fed to the input of integrated voltage regulator (7818). The output (18 VDC) is used as the low level supply and approximately 350 mA current is drawn by the drive circuitry. This second supply is needed to have bi-level drive for the stepper motors, so it provides current at low speeds (246pps) mostly and prevents the dissipation of transistors (Q_1 and Q_2) due to high voltage, by keeping them at the cut off.

If the low voltage supply were not included in the power supply unit, then, after the inductor current reaches its rated value (0.7 A), there would be a voltage on the transistors (V_{CE}) Q_1 and Q_2 which is equal to 42-17=25 V where

42 V = The high voltage supply

17 V = Voltage drop on the motor windings + V_{CE} of switching transistors (Q_4, Q_5, Q_8 and Q_9) + Voltage drop on the current sensing resistors (R_9 and R_{10})

Thus this voltage (25 V) with the rated current of motor windings, would cause approximately 17.5 W power dissipation on the transistors Q_1 and Q_2.

3. + 5 Volt Supply

The output of 18 V voltage regulator is also taken as the input for +5 V regulator (7805).

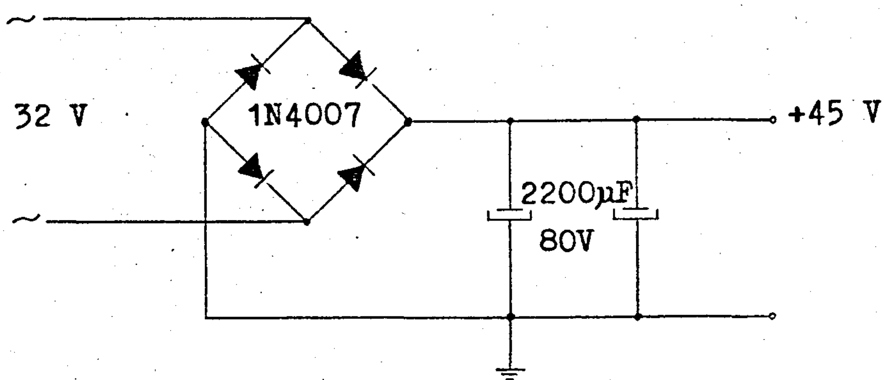
+5 V supply is used for the Z-80 microprocessor card, for the drill simulator circuit and also for the detection circuitry. The amount of current that is drawn from this supply is approximately 250 mA.

4. Adjustable Voltage Supply

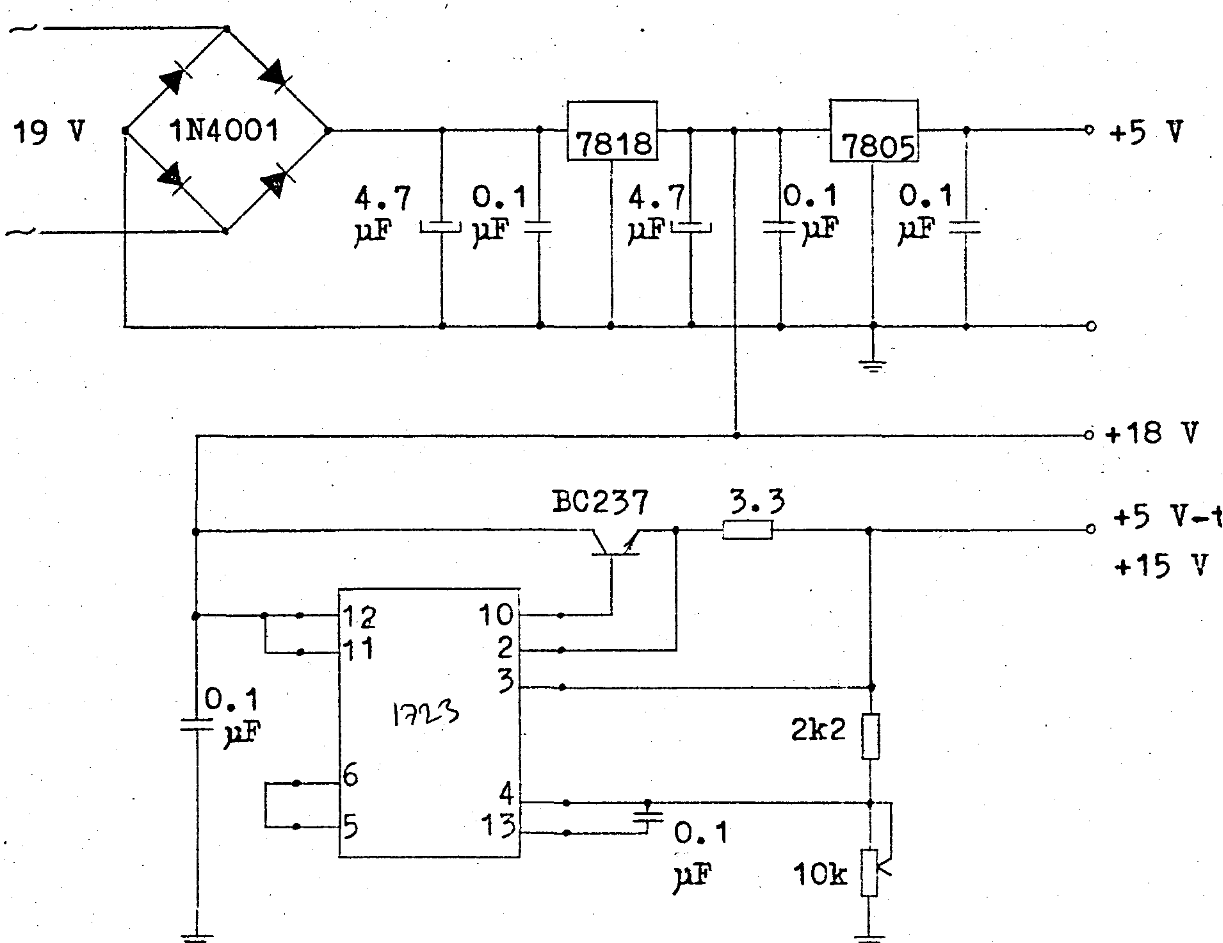
This adjustable voltage regulator (723) also uses 18 V regulator output as input and gives output to the lamp of the detection circuit. The voltage regulator is so set that its output can be adjusted from +5 V to +15 V. Adjustable sourc is needed because of the critical trigger levels of optic sensc output. Lamp needs approximately 70 mA.

The related figures of the power supply unit are shown on the following page.

High Voltage Supply



Low Voltage & +5 V Supply



Adjustable Supply

Figure 3.6

F. Optic Detector System

The optic detection is one of the most important feature of this thesis. Presently, there are many different kinds of optical sensors. In the developed system, the resolution of two consecutive points is 1.27 mm. This is not a very small distance for the sensor which is chosen for scanning procedure.

The optic sensor is not sensitive to the side lights due to its lens. In application, a small light source is used and the detector is centered on top of it, keeping a distance for the glass plate and the film.

For detection process, the operator places the dark solid dots on the transparent film. Then this film is scanned line by line. The sensor which is a transistor changes its state depending on the intensity of the light. The transistor is used in the common emitter configuration and the collector voltage is taken as output. Due to the mechanical movement and other disturbances, the best results can be achieved by using a comparator. For this purpose, LM 324 is used in the prototype. So the triggering level can be determined by adjusting the potentiometer i.e comparator inverting input voltage level is adjusted.

The comparator is essential in the circuitry in order to generate a +5 V signal for even a small input from the optical sensor. Direct coupling between the sensor and the microprocessor would not be appropriate because of the weak signal from the sensor, not being able to trigger the input port of the PIO of the Z-80 microprocessor.

Consequently, the output switches between high (+5V) and low (OV) levels. Those voltages are applied to the input port of the PIO of the Z-80 microprocessor card and this optical information is written to the memory, according to the scan program.

The detection circuitry and the detection mechanism are shown in figure 3.7 and figure 3.8 respectively.

Detection Circuitry

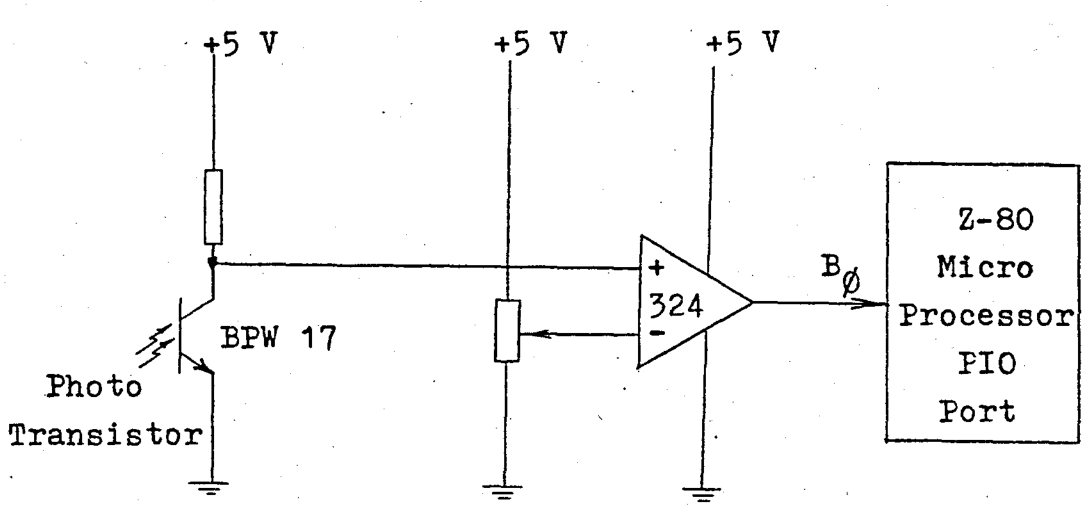


Figure 3.7

Detection Mechanism

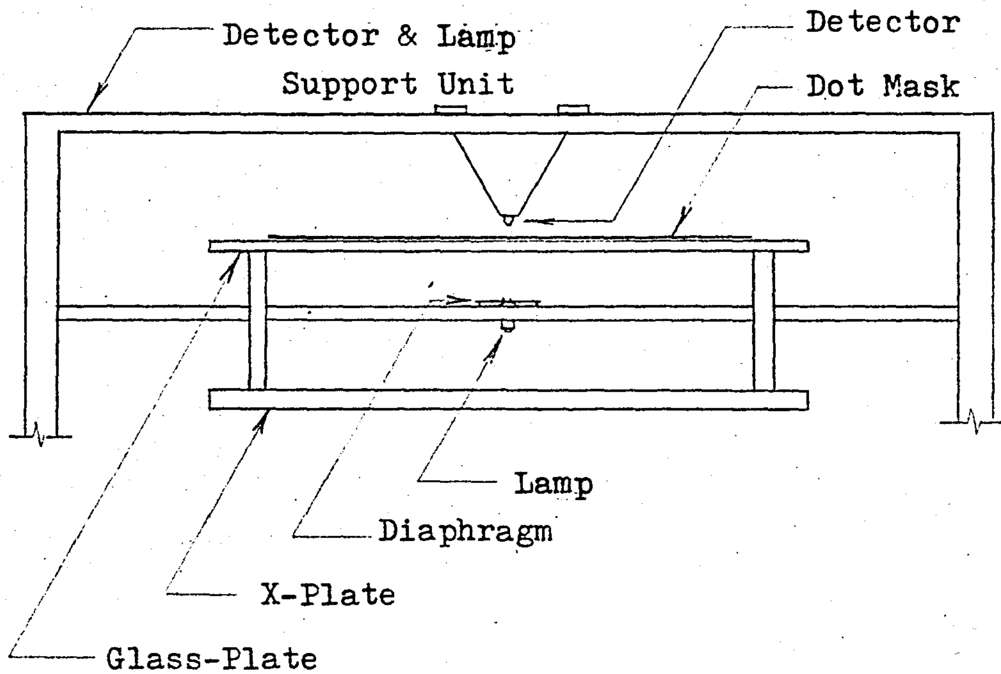


Figure 3.8

IV. SYSTEM SOFTWARE

The software of the system consists of mainly two sections and each program contains subprograms.

A. Detection Program
   1. Line Detection Program
   2. Frame-length Detection Program
   3. Scanning and Storing Program

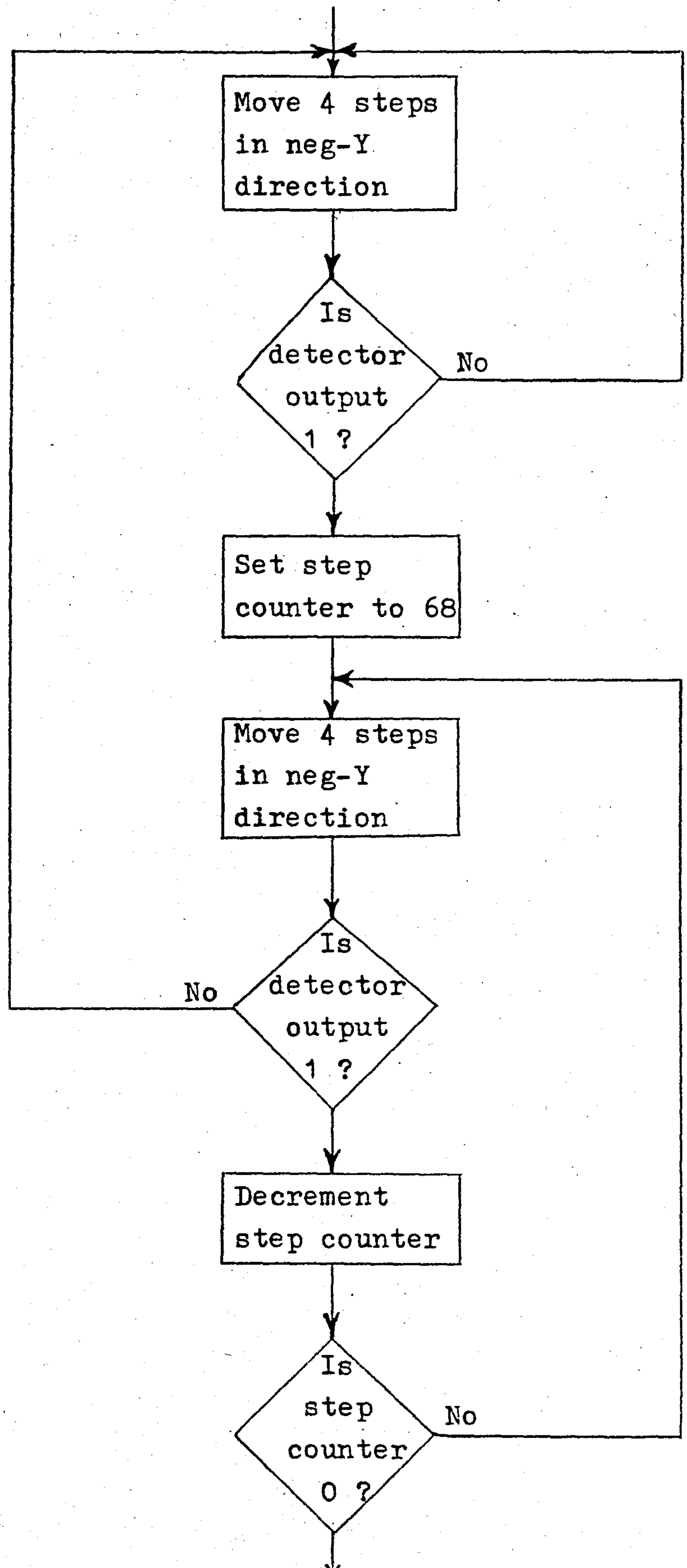
B. Drilling Program

C. Subprograms
   1. Reset Routine
   2. Interrupt Service Routine
   3. Acceleration and Deceleration Subroutines
   4. Constant Speed Subroutines
   5. Delay Subroutines

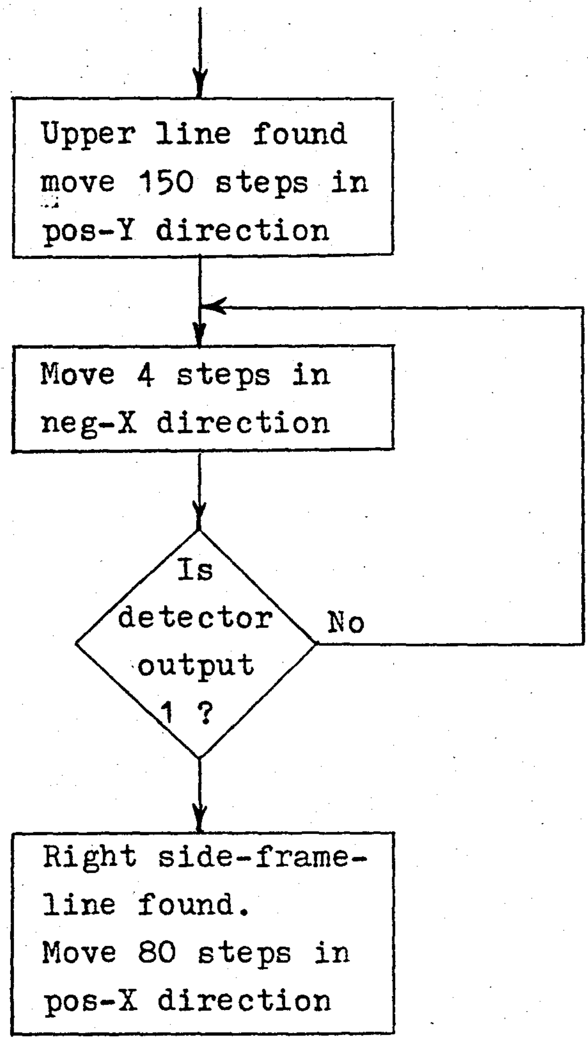
A. Detection Program

1. Line Detection Program

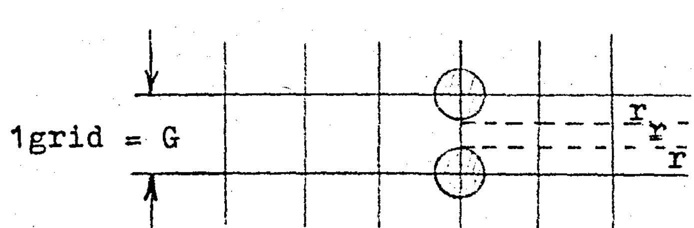
This routine searches the frame lines and locates the detector on the top right corner of the dot mask. Program flow is shown below :



Continued in the next page



The number of steps written in the flowchart are found according to the dimension of dots and grid length i.e dots have a radius of 27 steps where each step is equal to 0.015875 mm. and the length of a grid is 80 steps. Thus, two consecutive dots have a spacing distance of 26 steps as shown in Figure 4.1 .

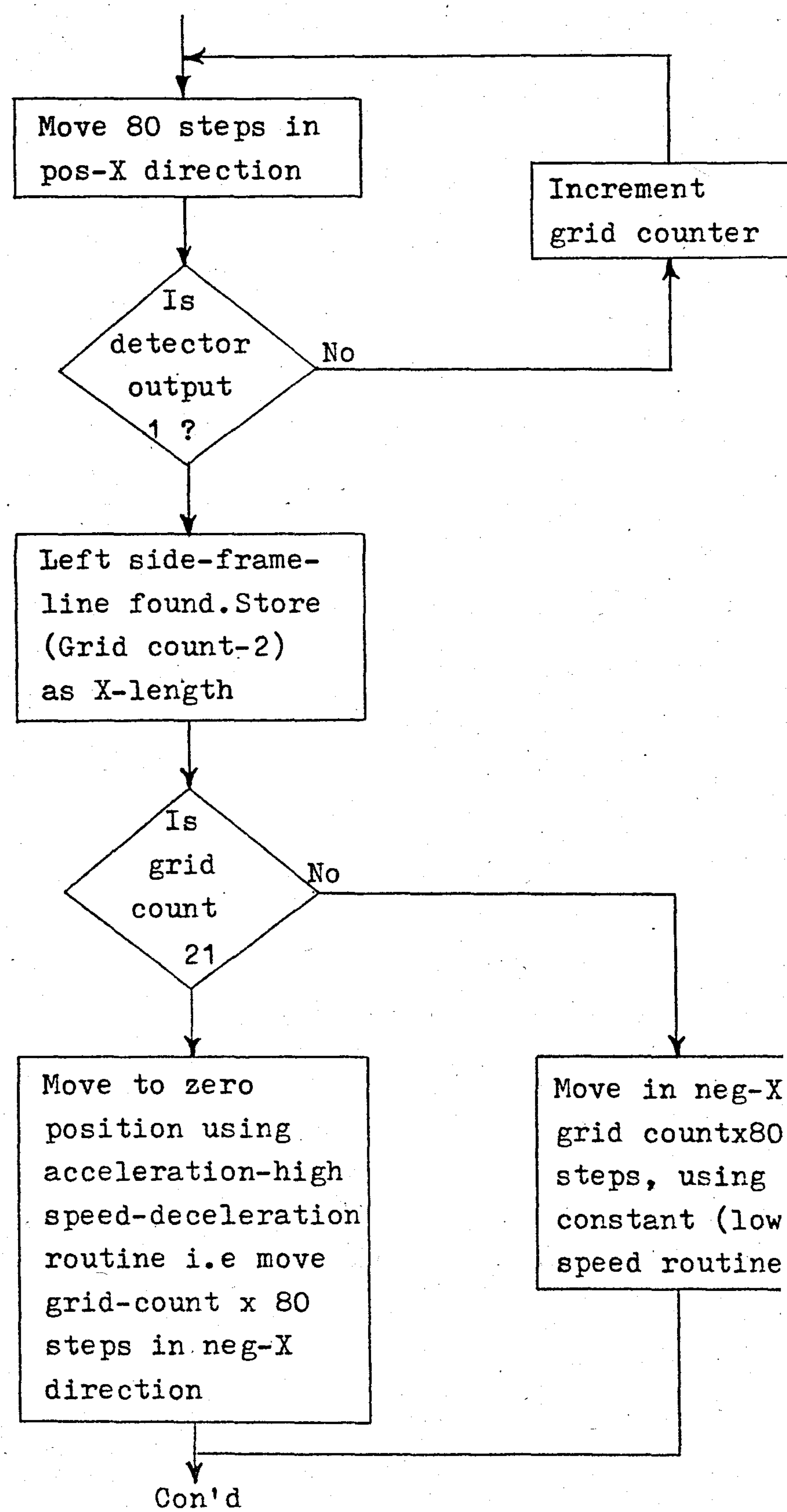


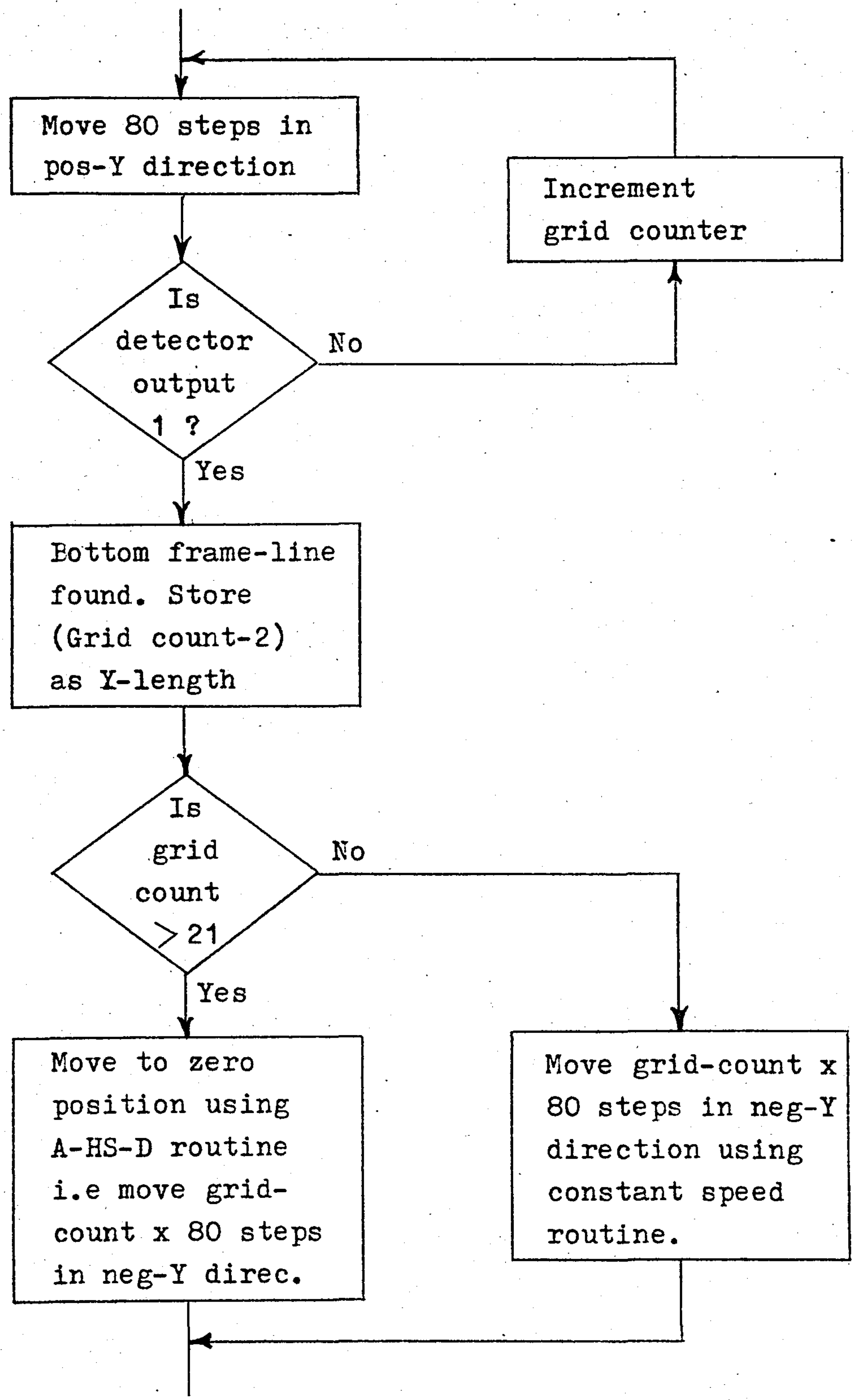
Dot dimensions

Figure 4.1

2. Frame Length Detection Program

This program measures the X and Y lengths of the frame of dot mask.





In this program the grid counter is compared with 21 since 11 grids are needed for acceleration and 10 for deceleration (1 grid = 80 steps).

In the flowchart A-HS-D is used to express the acceleration-high speed-deceleration routine in short.

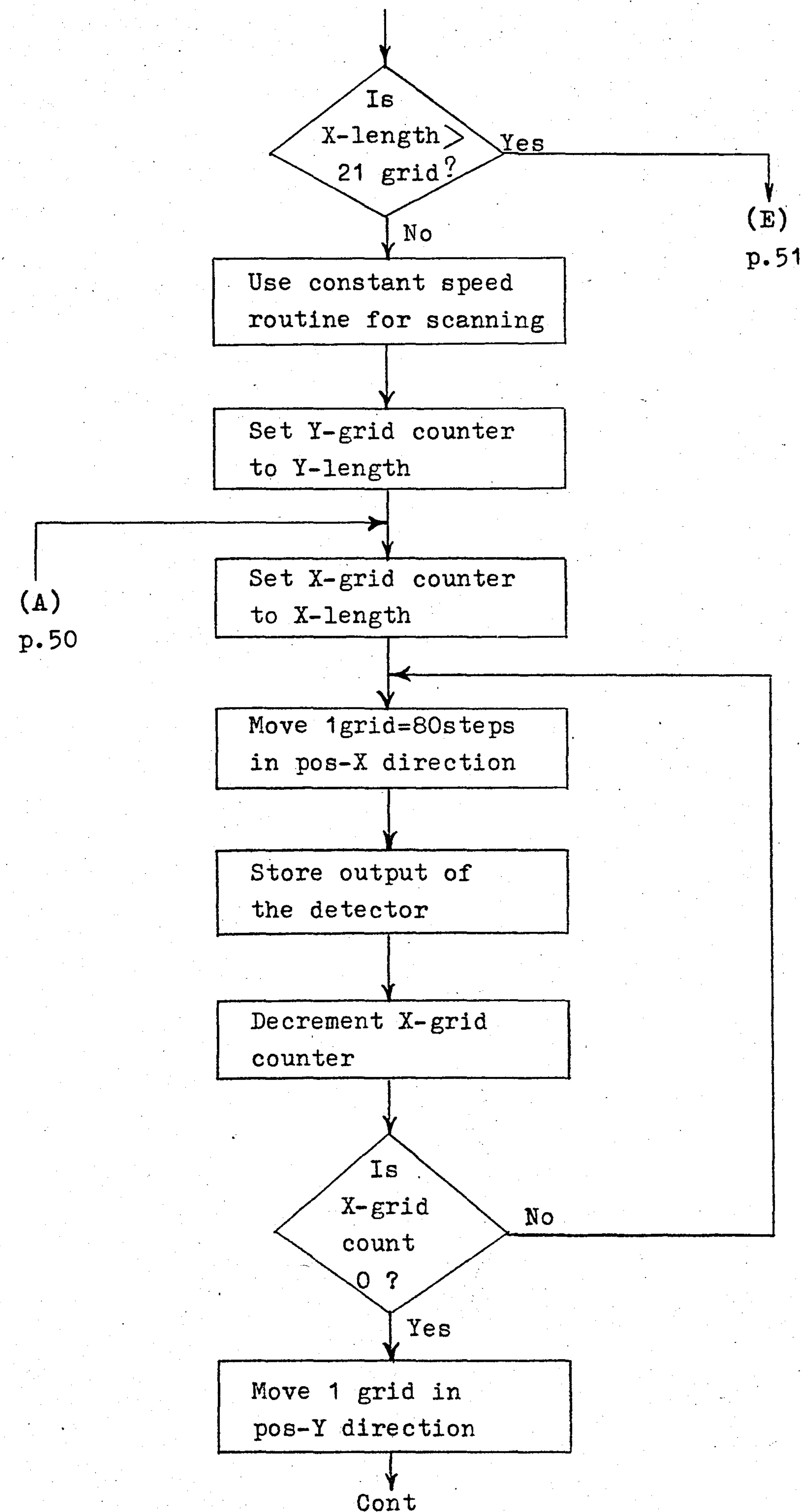
3. Scanning and Storing Program

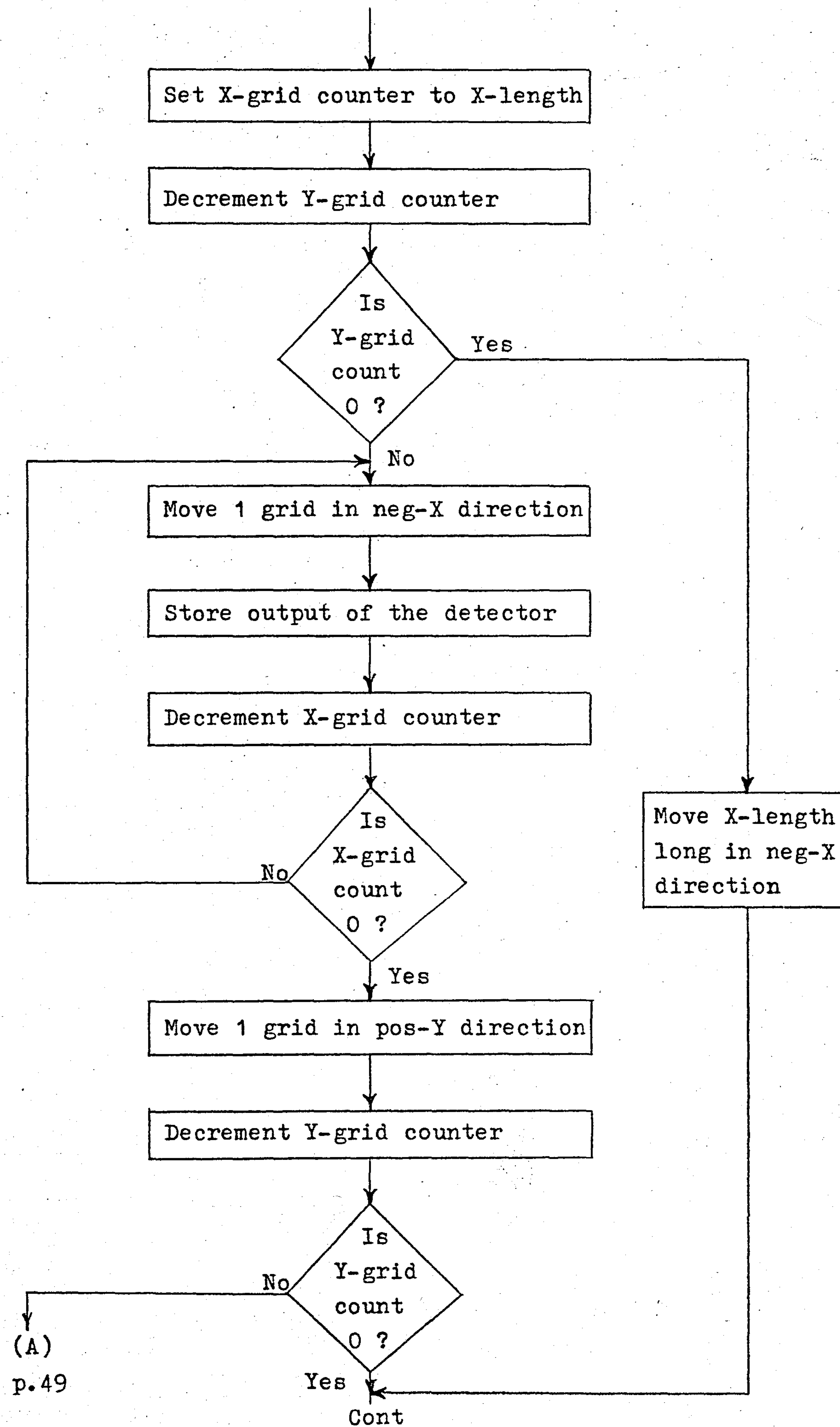
This program scans the dot mask whose dimensions are specified in the previous programs, in a meander pattern i.e odd numbered lines are scanned from right to left while even numbered lines are scanned in the opposite direction.

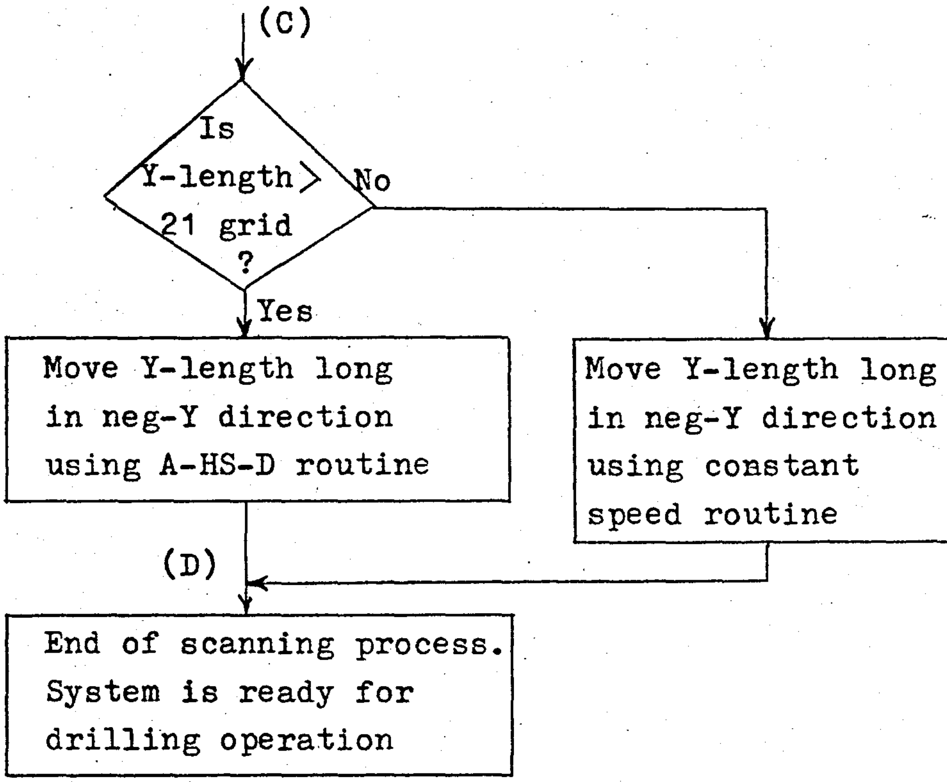
The store routine tests the output of the detector at every grid-node where spacing between two grids is 80 steps which is equal to 1/20 inch. Each test result (0 or 1) is stored in a temporary memory location and when 8 consecutive hole-position information is read (i.e one byte is filled) then this datum(byte) is stored in the memory starting from the address 8000 H.

The scan program can proceed with two different speeds. One of them is constant speed (246 steps/sec) scanning routine and the other is acceleration-high speed-deceleration routine. Those cases will be explained respectively.

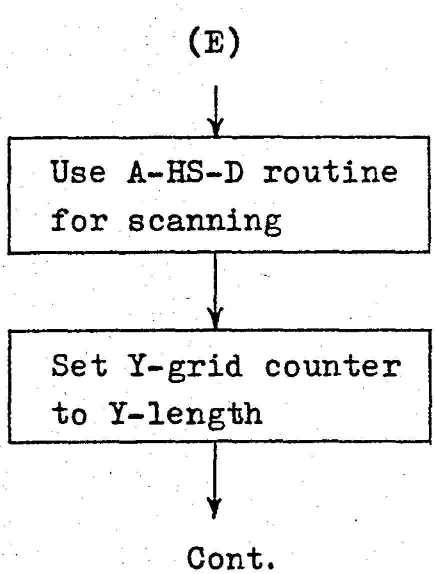
<u>a. Constant Speed Scanning and Detection.</u> This routine is selected if the X-length of the dot mask is equal or smaller than the distance reqnired for acceleration,high speed and deceleration. This distance is equal to 21/20 inch. The flowchart of this routine is in the following pages.

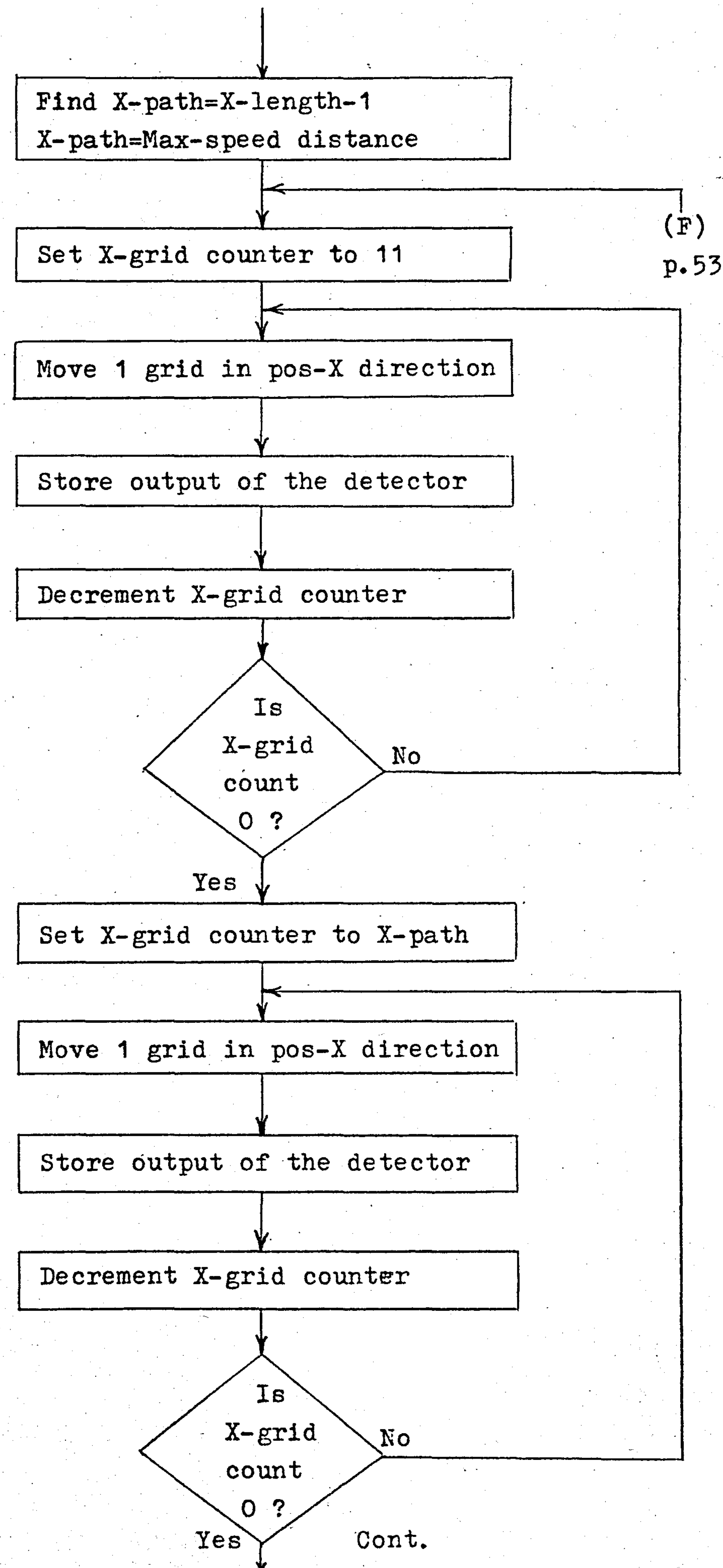

Is
X-length > 21 grid?
Yes
(E)
p.51
No
Use constant speed routine for scanning
Set Y-grid counter to Y-length
(A)
p.50
Set X-grid counter to X-length
Move 1grid=80steps in pos-X direction
Store output of the detector
Decrement X-grid counter
Is X-grid count 0 ?
No
Yes
Move 1 grid in pos-Y direction
Cont

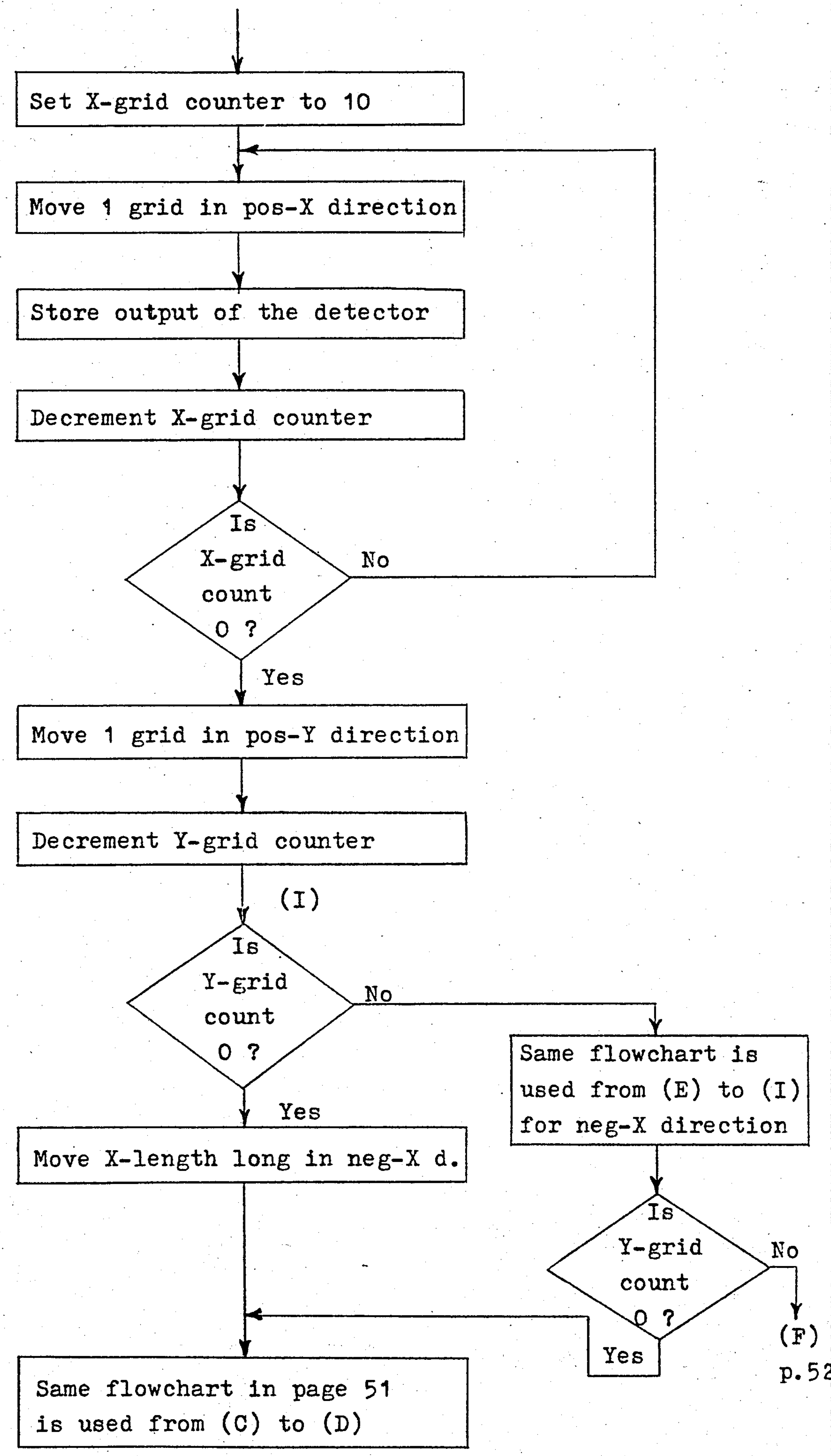

Set X-grid counter to X-length
Decrement Y-grid counter
Is Y-grid count 0 ?
Yes
No
Move 1 grid in neg-X direction
Store output of the detector
Decrement X-grid counter
Is X-grid count 0 ?
No
Yes
Move X-length long in neg-X direction
Move 1 grid in pos-Y direction
Decrement Y-grid counter
Is Y-grid count 0 ?
No
(A)
p.49
Yes
Cont



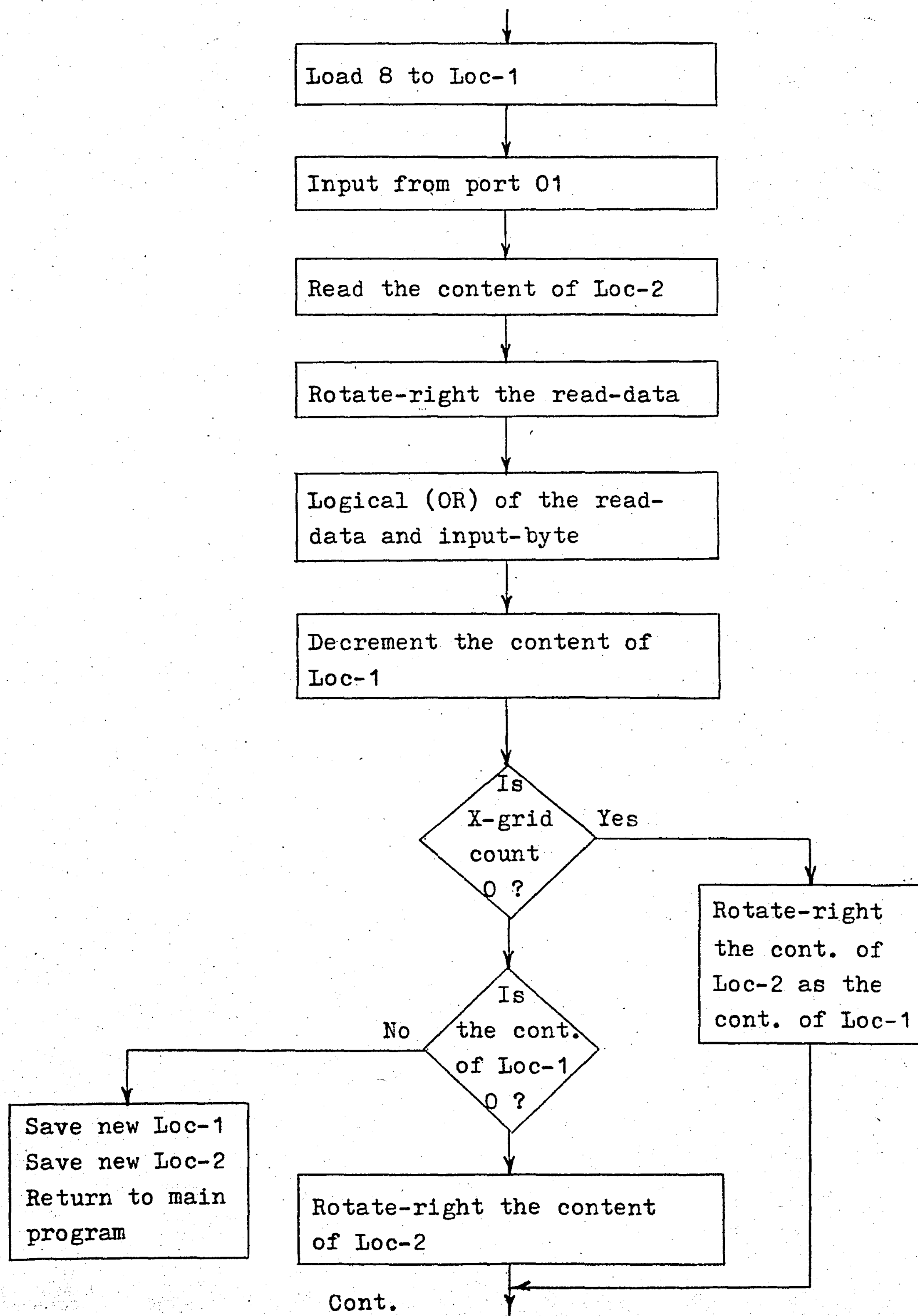
<u>b. High Speed Scanning and Detection.</u> When the X-length of the frame of dot mask, is greater than 21/20 inch, this routine is selected, so a better performance (i.e a shorter scanning time) is obtained due to acceleration--high speed (835 steps/sec)-deceleration profile instead of constant speed (246 steps/sec) scanning. Program flow is as follows :

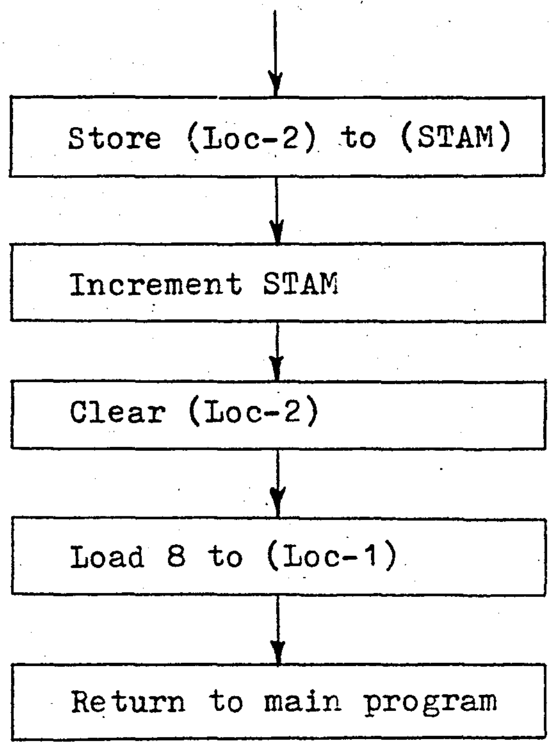



Find X-path=X-length-1
X-path=Max-speed distance
(F)
p.53
Set X-grid counter to 11
Move 1 grid in pos-X direction
Store output of the detector
Decrement X-grid counter
Is X-grid count 0 ?
No
Yes
Set X-grid counter to X-path
Move 1 grid in pos-X direction
Store output of the detector
Decrement X-grid counter
Is X-grid count 0 ?
No
Yes
Cont.


Set X-grid counter to 10
Move 1 grid in pos-X direction
Store output of the detector
Decrement X-grid counter
Is X-grid count 0 ?
No
Yes
Move 1 grid in pos-Y direction
Decrement Y-grid counter
(I)
Is Y-grid count 0 ?
No
Yes
Same flowchart is used from (E) to (I) for neg-X direction
Move X-length long in neg-X d.
Is Y-grid count 0 ?
No
(F) p.52
Yes
Same flowchart in page 51 is used from (C) to (D)

<u>c. Store Program.</u> By this program data which hold the drilling-hole-position information, are stored in the memory sequentially. Flowchart of the program is shown below:





Expressions used in the above flowchart are:

Loc-1 : Holds the bit count. Initially Loc-1 contains 8 and decremented at each test of the output of detector (at each grid-node). Its content reaches zero when 8 data are taken i.e a byte of hole-position information is ready to be stored.

Loc-2 : Keeps the data till a byte of information is formed. Then that byte is stored in the memory.

STAM : Means starting address of memory. The data (byte) in Loc-2 is stored to the memory location that STAM shows. In the beginning STAM contains 8000 H which is the first RAM location in the memory unit. STAM is incremented after each byte-store.

B. Drilling Program

This program reads the hole-position data from the memory and drives the X-Y stage scanner to perform the drilling process. Path optimization algorithm is used to minimize the drilling procedure time.

In the program, hole-position data is read from the memory as one byte at a time and every byte is tested bit by bit. A position counter (register) is set to X-length value and it is decremented at each bit test. If the tested bit contains'1' i.e that is a hole information then the value of position counter is stored in the memory starting from the address FirstIX=8730H. Zeroes which are encountered during bit test are not taken into consideration. When position counter reaches zero then it means that, line is finished and drilling of the holes on that line can be performed.

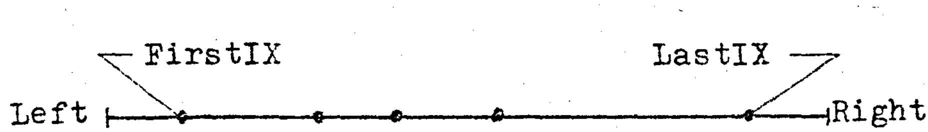
The flowchart in the following page is written for the odd numbered lines. In the even numbered lines hole-positions are listed (stored) from left to right. This can be simply shown as follows:

<u>Odd numbered line</u>

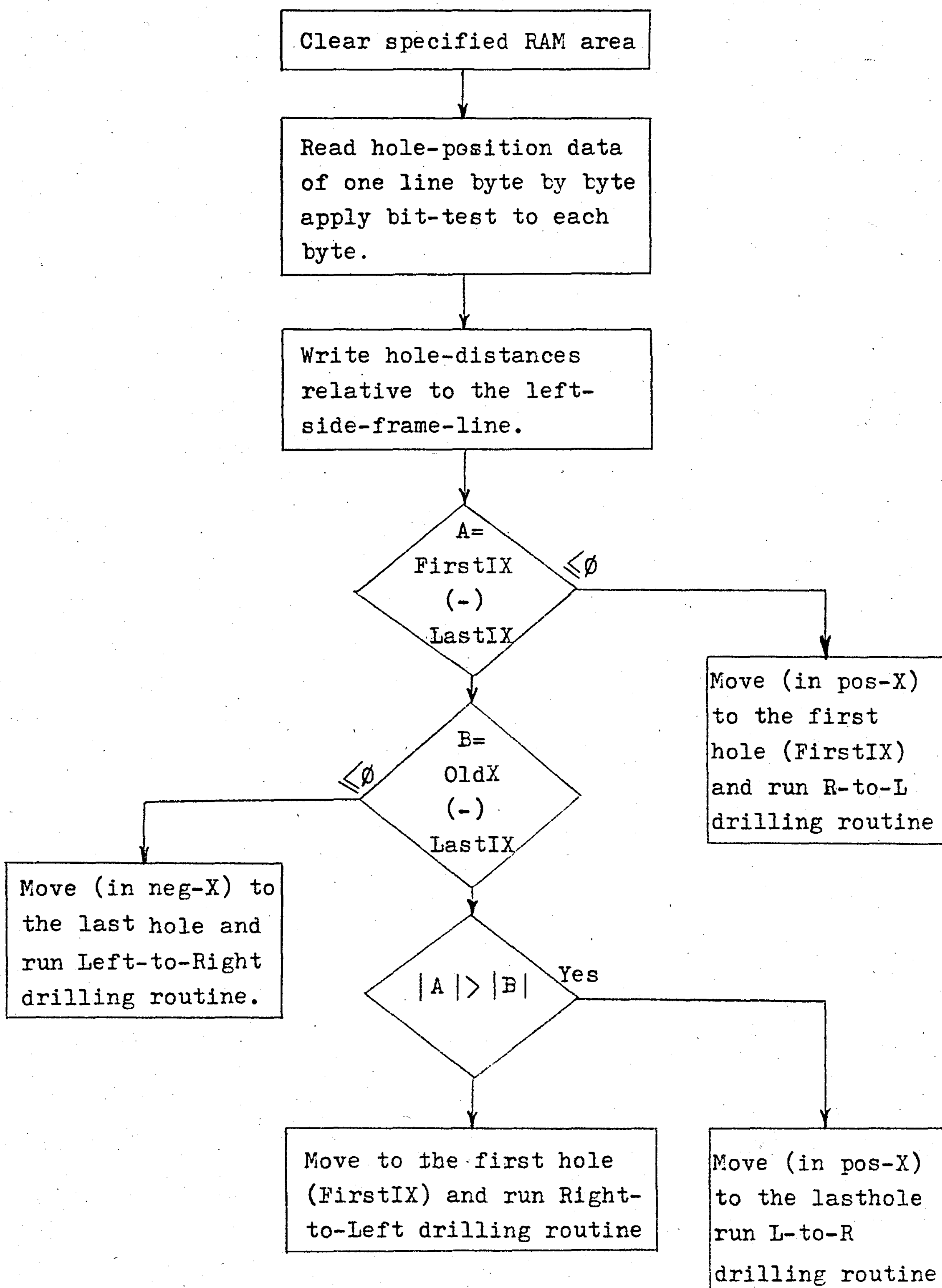
FirstIX (The first hole location)

LastIX (The last hole location)

<u>Even numbered line</u>



Flowchart of the drilling program for the odd numbered lines:



In the drilling program, OldX contains the last location of the X-motor. Locations of the holes are specified according to the distance from the left-frame-line i.e hole-position numbers increase from left to right.

After the execution of one line, the next line data are tested. If there is no data in any line then stage moves one grid (80 steps) in the positive-Y direction.

The movement of the stage from one hole position to another is performed due to calculation of relative distance between two subsequent hole-positions. If the distance between two hole-positions is greater than 21 grids then acceleration-maximum speed-deceleration routine is run otherwise the X-Y stage scanner moves with constant speed.

At the end of drilling program, return routine takes place and moves the X-Y table to its original zero position for another drilling operation.

C. Subprograms

1. Reset Routine

This subroutine :

a. Sets the stack pointer to 87E5 H

b. Chooses interrupt mode 2

c. Loads (I) interrupt page address register with Ø7

d. Programs A-port of PIO as output port and,

B-port of PIO as input port

Reset Routine

```
ØØØØ    31 E5 87    LD SP,87E5 H
ØØØ3    ED 5E       IM 2
ØØØ5    3E Ø7       LD A, Ø7
ØØØ7    ED 47       LD I, A
ØØØ9    3E ØF       LD A, ØF H
ØØØB    D3 Ø2       OUT (Ø2), A
ØØØD    3E 4F       LD A, 4F H
ØØØF    D3 Ø3       OUT (Ø3), A
ØØ11    AF          XOR A
ØØ12    D3 ØØ       OUT (ØØ), A
ØØ14    FB          EI
ØØ15    76          HALT
```

Table 4.1

2. Interrupt Service Routine

Control of the X-Y stage scanner is provided by this routine. Start switch starts the scanning and then the drilling programs. The other four switches (pos-X, neg-X, pos-Y, neg-Y) are used to move the X-Y table in both X and Y directions. Stop switch is used to break the above mentioned programs when they are running.

This routine utilizes the interrupt feature of Z-80 microprocessor. The Z-80 CPU is so operated that an indirect call to any menory location can be achieved in response to an interrupt. For this purpose the I register is loaded with Ø7 which is the high order 8-bits of the indirect address. The lower 8-bits of the address are provided by the interrupting switch.

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 4.2

3. Acceleration and Deceleration Subroutines

In the second section, driving conditions are examined for stepper motors. The necessary pulse sequences are supplied from the microprocessor card and the look-up table for bit patterns (shown in table 4.3) is created in the memory. Since stepper motors cannot start at high speeds, they are driven at low speed in the beginning and speed is increased step by step, changing the delays between pulse-patterns. For the above mentioned reasons, a specific distance is needed to accelerate and decelerate, which is 11 grids for acceleration and 10 grids for deceleration (1Grid=80steps). In the prototype, by changing delay constants, speed is changed from 238 steps/sec to 791 steps/sec in 11 stages.

Stepper Look-up Table

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

Table 4.3

These bit patterns (shown in Table 4.3) are sent to drivers through PIO unit. For clockwise rotation, bit patterns are in the form of A, 9, 5, 6 and opposite order for the counterclockwise rotation.

4. Constant Speed Subroutines

These routines are used almost in all programs. They provide constant delay between pulse patterns to be sent to stepper motor drivers. Motors obtain a speed of 246 steps/sec by these routines, and a register of the processor is used as the step counter. At every 80 step routine repeates itself.

5. Delay Subroutines

In all programs those routines are used either in constant form or in variable form.

For the stepper motor's pulse sequence, a delay of 8017 T-cycles long, is used between two consecutive pulses. Where, every T-cycle takes 0.5 microsecond due to 2 MHz clock frequency.

For the variable delay, a constant,in the delay routine,is changed. This kind of delay is used in the acceleration-maximum speed-deceleration routine to provide different speeds.

Line Detection

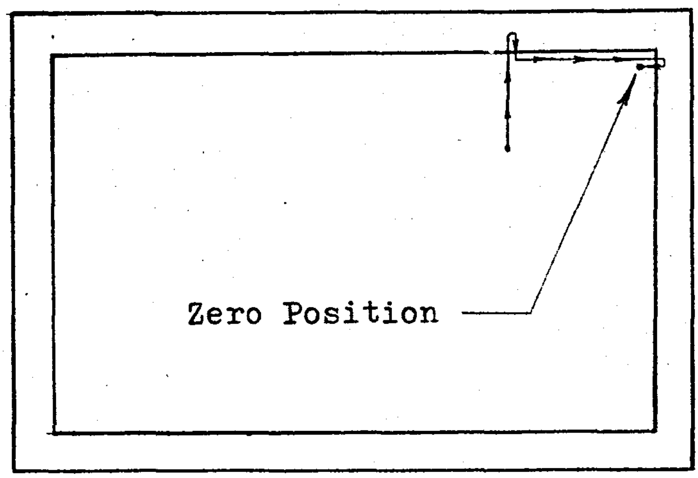


Figure 4.2

Frame Detection

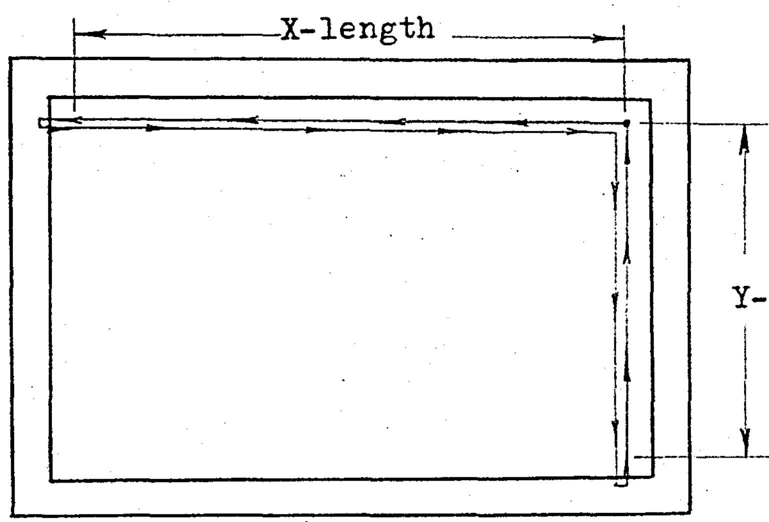


Figure 4.3

Scanning (Meander Pattern)

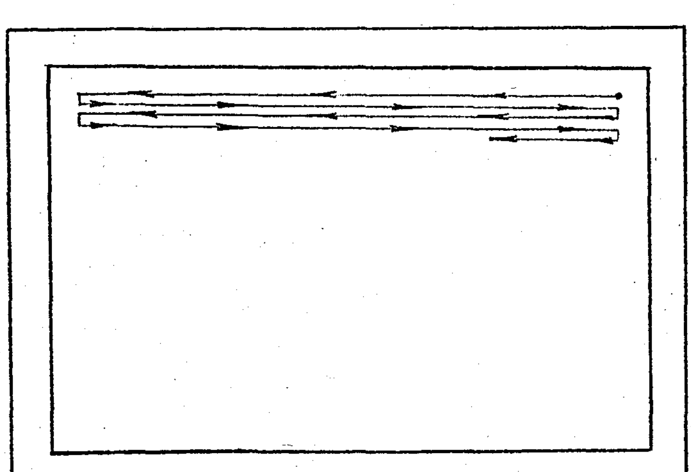


Figure 4.4

V . CONCLUSION

Presently the automation has led to the improvement in production processes, thereby needing minimum manpower. The advantages are, that the whole process once set up, can be executed repeatedly at higher speeds than is possible with an operator, mechanically doing the settings.

During the realization of the prototype, some problems were encountered but all of them were eliminated, for instance, for the mechanical assembly the most important thing was the backlash of the system, which was eliminated by a great deal of precision on the part of the mechanic. Also,in the same case certain methods to prevent backlash (e.g spring system, adjustable nut) could be utilized. Keeping in mind that it was better if the bearings at each end of the lead-screws were of the type that can move in their housing freely, bearing of such a nature could have been used. In addition to these, the alignment of the mechanical assembly had to have some adjustment points to provide smooth movement. In the developed system this was solved by the expertise of the mechanic.

The other important thing was the selection of stepper motors. Since , prior to this selection it was not known how much torque, a mechanical assembly would need, the steppers were chosen from powerful series which generally have low speed, high torque characteristics. This aspect resulted in success by means of the designed drive circuitry.

The designed system shows an effective use of microprocessors in the industrial field. The objective was to realize a system that needs minimum interaction of the user and to give accurate results. The system performance provides the above mentioned properties.The user can design his/her circuits freely and also due to system resolution, it can fit the mask pattern. The scan time, though, could be reduced considerably if motors of high speed (motors with a speed of upto 4000 steps/sec, rather than the ones used with a speed of 835 steps/sec) more suitable,were selected. The scanning process for a Eurocard takes about 17 minutes, which is considerably short a time due to the right and efficient usage of the software. By using the optimum path concept, this reduction in time is achieved.

The system can work as a drilling machine with some mechanical modifications such as the detector, the lamp and the glass plate can be replaced by a drill set to perform the drilling operation.

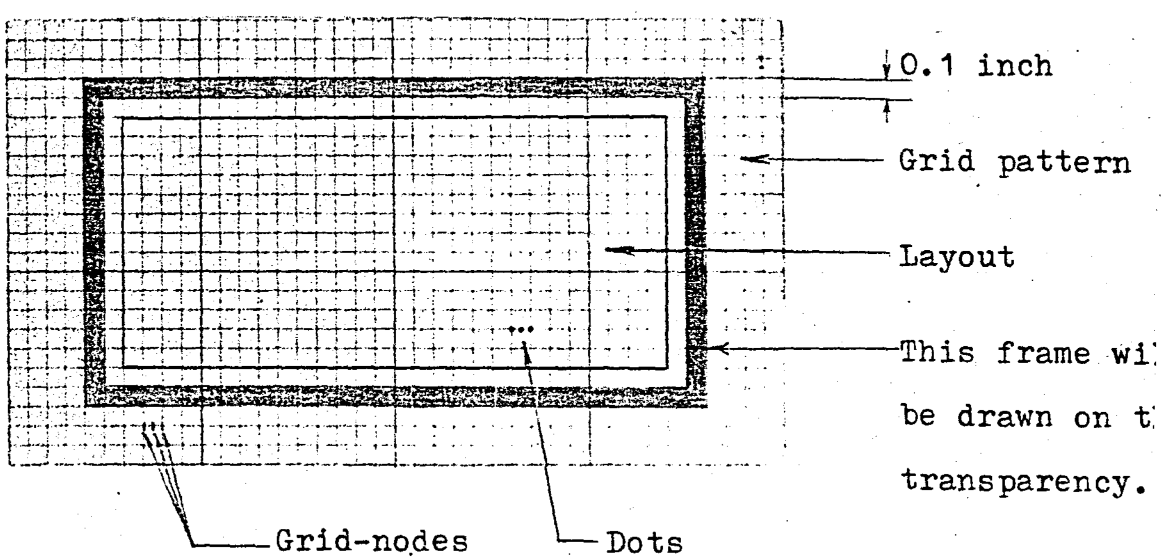
A P P E N D I C E S

APPENDIX A

OPERATING INSTRUCTIONS

1. How to Prepare The Dot Mask

i. After the printed circuit layout is drawn, it is placed on a paper which has 0.1 inch divisions (as shown below fig.). On these two, the clean transparency is located.



ii. The frame lines of the dot mask are drawn such that every side of the mask frame is 0.1 inch greater than the frame of the layout. Frame-line thickness of 0.1 inch is considered to be enough and it is advisable to use a drawing-pen with black ink (e.g Rapido 0.5 mm).

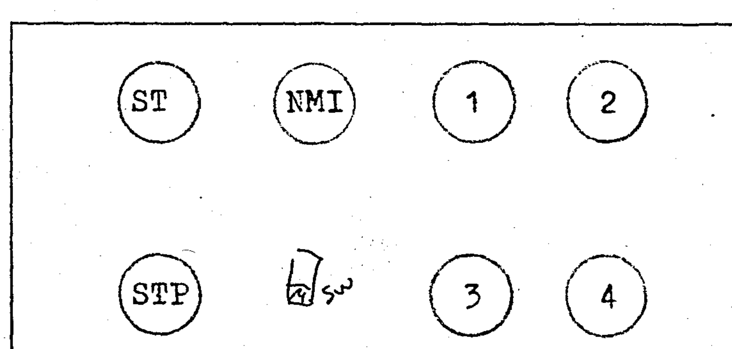
iii. Hole positions are marked on the transparency by locating them at the nodes of the grid pattern. One more hole can be marked between each node of the grid pattern shown in the previous page, because the scanning resolution of the system is 1/20 inch.

If there are any hole positions on the layout which do not coincide with the nodes of the grid pattern, they should be placed on to the closest node.

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

2. Running The System

i. System is switched on. Power on-reset runs the reset program and Halt LED is turned on at the end of this program. Then using the manuel control switches, the detector is placed somewhere in the dot-mask frame (i.e X-Y table is positioned in such a way that Detector points somewhere in the dot-mask frame).



Control Panel

The functions of the buttons are as follows:

Button ST : START

Button STP : STOP

Button NMI : Emergency stop

Button 1 : X-Motor control (stage moves in neg-X direction)

Button 2 : X-Motor control (stage moves in pos-X direction)

Button 3 : Y-Motor control (stage moves in pos-Y direction)

Button 4 : Y-Motor control (stage moves in neg-Y direction)

ii. Pressing the ST (Start) button starts the scanning and detection program and at the end, stage takes its original zero position (at the upper right corner) and HALT LED turns on.

iii. Now ST button commences the drilling program. At each drilling-hole-position, stage stops, a LED flashes for a few seconds to simulate the drilling operation.

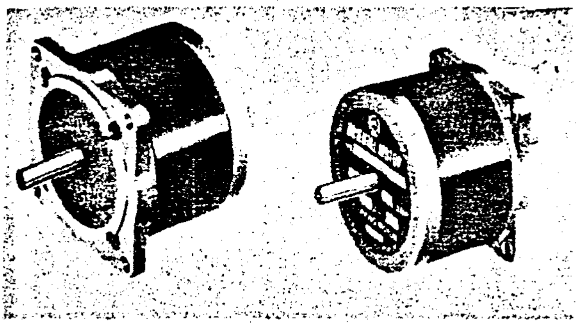
iv. After completing the drilling process of all holes, stage moves to zero-position and stops. Drilling process can be repeated as desired by pressing the ST button.

v. For a new dot-mask scanning process, RST (Reset) must be given to the system, then same order (from ii to iv) is followed.

In case of emergency, NMI button should be pressed.

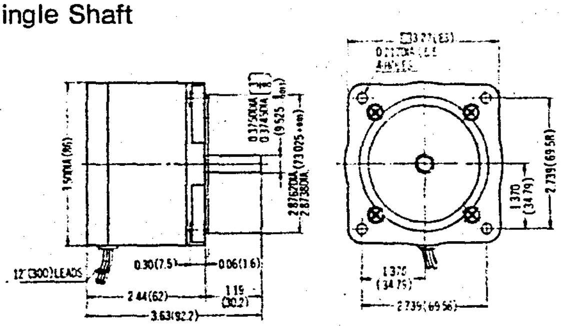
1.8° STEP ANGLE HYBRID TYPE

PH296-□□

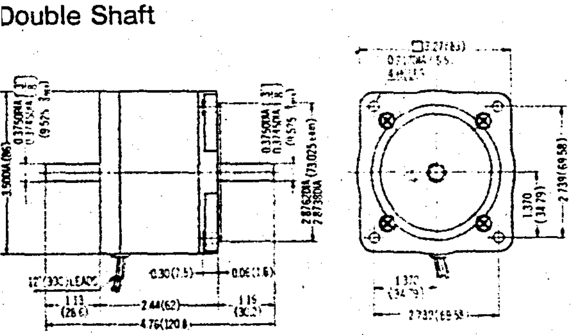


DIMENSIONS

Single Shaft

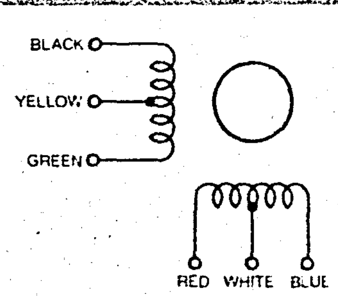


Double Shaft

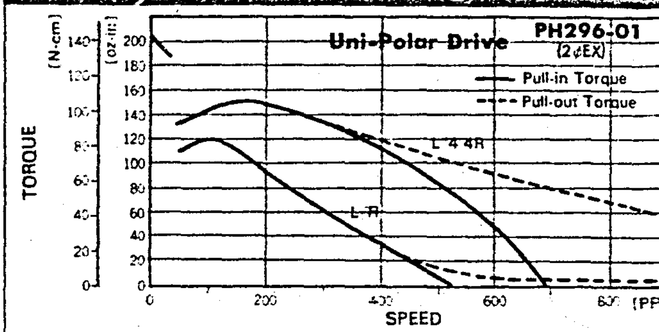


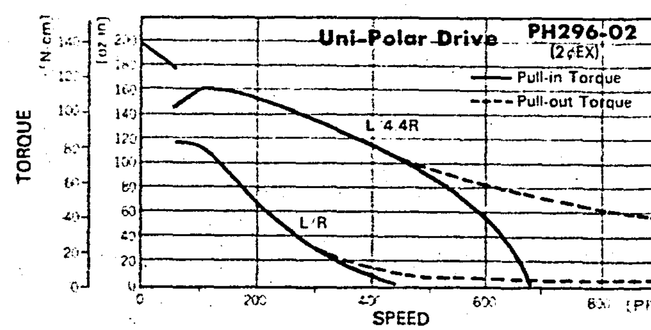
scale 1:4, unit=inch (mm)

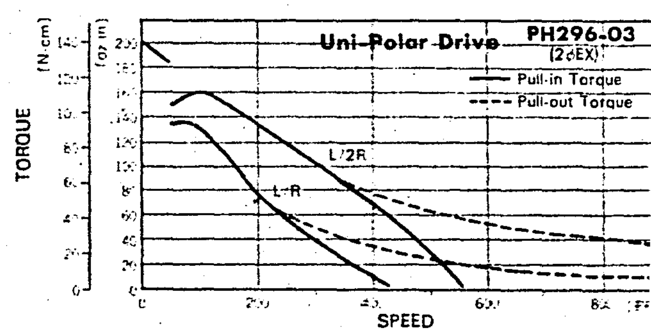
COLORS OF LEAD WIRES



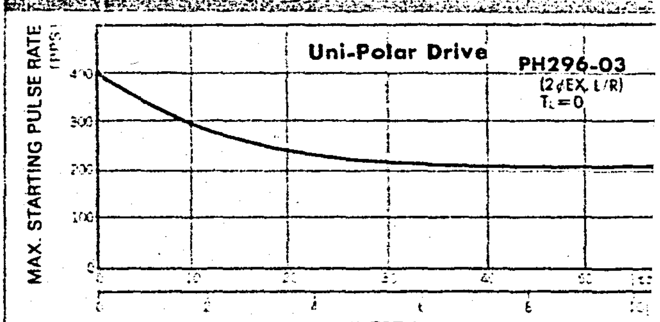
SPEED VS. TORQUE CHARACTERISTICS







INERTIA VS. STARTING PULSE RATE CHARACTERISTICS



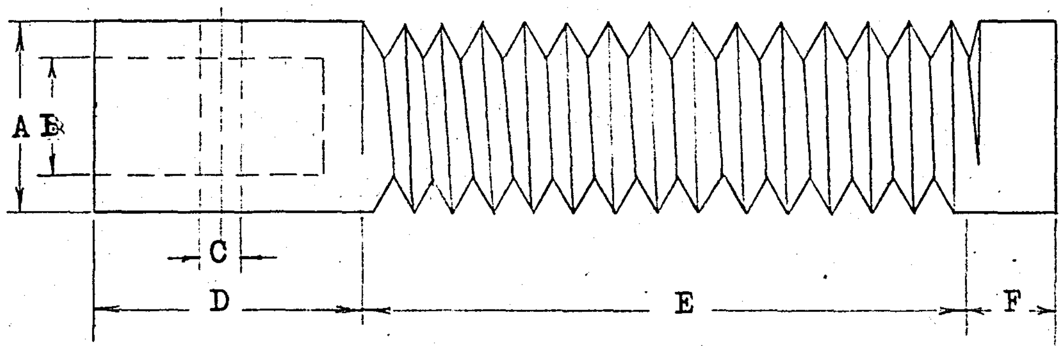
SPECIFICATIONS (2-phase full-step)

Motor type		Voltage	Current per phase	Holding Torque		Resistance per phase	Induct per pl
Single Shaft	Double Shaft	V	A/phase	oz-in	N·cm	ohm/phase	mH/p
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)

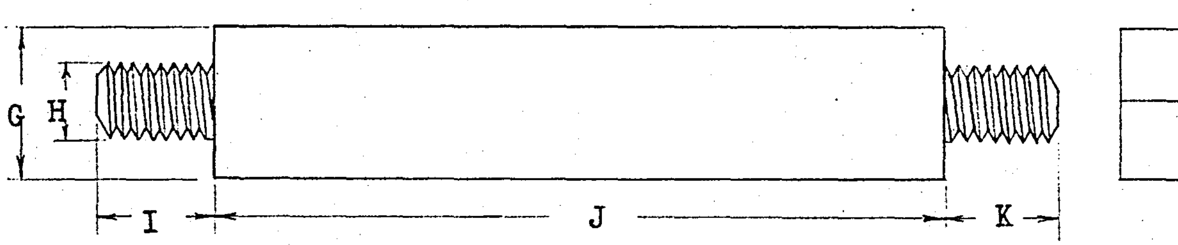
APPENDIX C

LEAD-SCREW



	A	B	C	D	E	F
X-Table	16	95	3	35	370	10
Y-Table	16	95	3	35	290	10

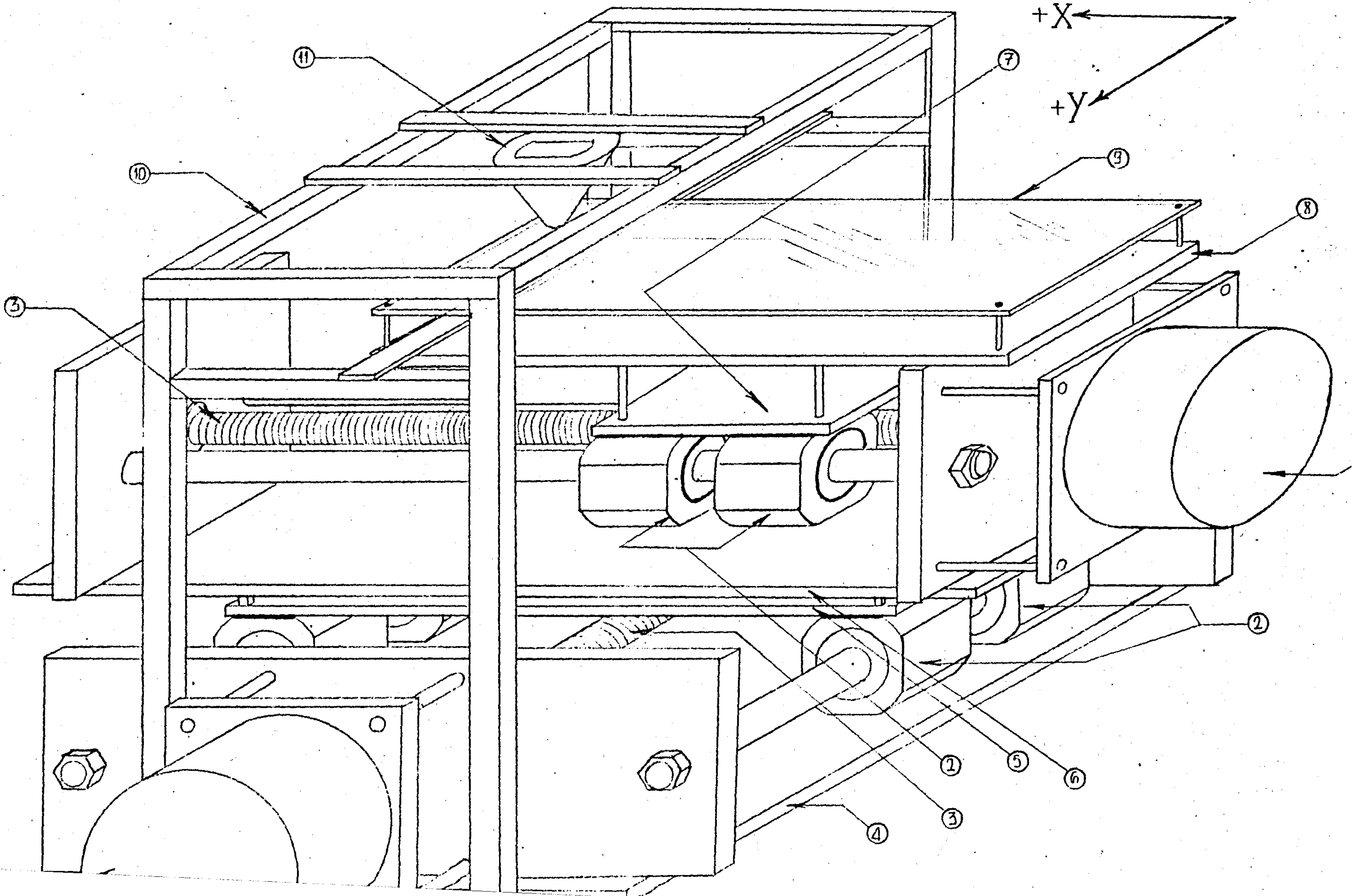
Cylindrical Shaft



	G	H	I	J	K
X-Table	16	10	15	370	15
Y-Table	16	10	15	290	15

i. Drawings are not to scale.

ii. All dimensions are in millimeters.



+X
+Y
11
7
10
9
8
3
2
1
5
6
3
4

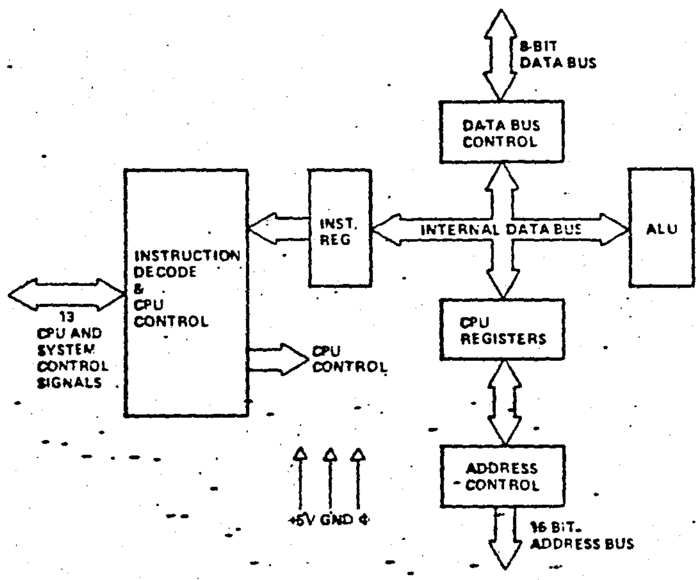
Part List of X-Y Stage Scanner

1. Stepper motors
2. Linear motion bearings
3. Lead-screws
4. Base plate
5. Intermediate Y-plate
6. Y-plate
7. Intermediate X-plate
8. X-plate
9. Glass plate
10. Support for lamp and photo-detector
11. Photo-detector system

APPENDIX D

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.



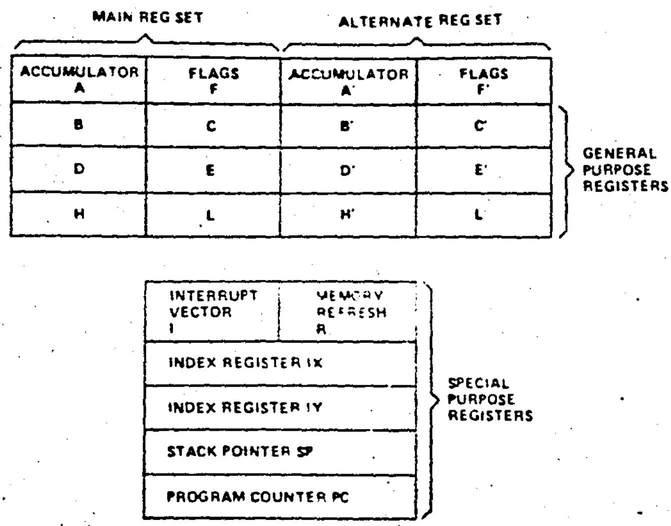
Z-80 CPU BLOCK DIAGRAM
FIGURE 2.0-1

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.

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:

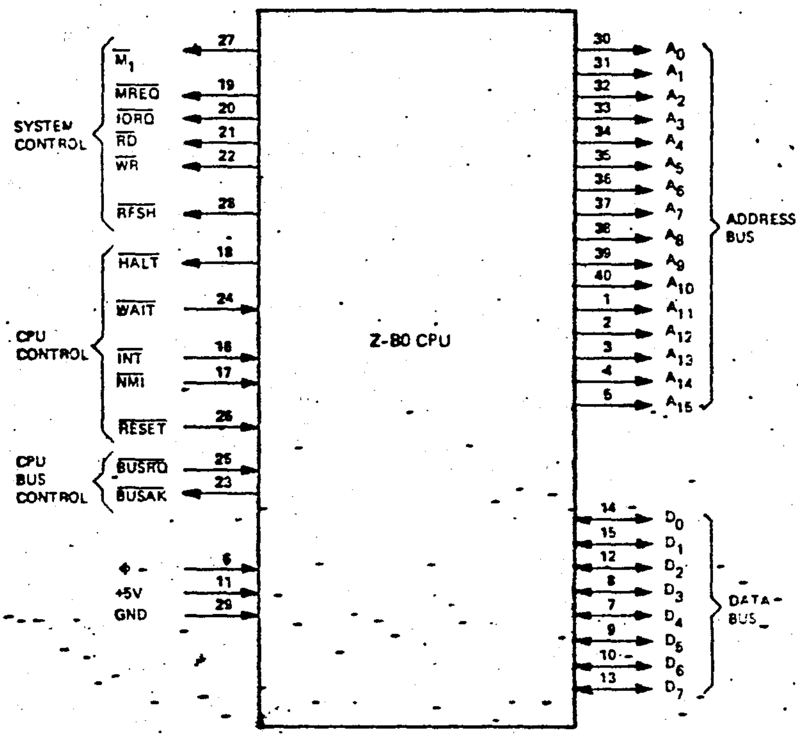
Add	Left or right shifts or rotates (arithmetic and logical)
Subtract	Increment
Logical AND	Decrement
Logical OR	Set bit
Logical Exclusive OR	Reset bit
Compare	Test bit

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.

Z-80 CPU PIN DESCRIPTION

The Z-80 CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in figure 3.0-1 and the function of each is described below.



Z-80 PIN CONFIGURATION
FIGURE 3.0-1

A_0-A_{15} (Address Bus)

Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges. I/O addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.

D_0-D_7 (Data Bus)

Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$ (Machine Cycle one)

Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution. Note that during execution of 2-byte op-codes, $\overline{M1}$ is generated as each op code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH or FDH. $\overline{M1}$ also occurs with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.

$\overline{MREQ}$ (Memory Request)

Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

APPENDIX E

INTRODUCTION

The Z-80 Parallel I/O (PIO) Circuit is a programmable, two port device which provides a TTL compatible interface between peripheral devices and the Z80-CPU. The CPU can configure the Z80-PIO to interface with a wide range of peripheral devices with no other external logic required. Typical peripheral devices that are fully compatible with the Z80-PIO include most keyboards, paper tape readers and punches, printers, PROM programmers, etc. The Z80-PIO utilizes N channel silicon gate depletion load technology and is packaged in a 40 pin DIP. Major features of the Z80-PIO include:

- Two independent 8 bit bidirectional peripheral interface ports with 'handshake' data transfer control
- Interrupt driven 'handshake' for fast response
- Any one of four distinct modes of operation may be selected for a port including:

  Byte output

  Byte input

  Byte bidirectional bus (Available on Port A only)

  Bit control mode

  All with interrupt controlled handshake
- Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic
- Eight outputs are capable of driving Darlington transistors
- All inputs and outputs fully TTL compatible
- Single 5 volt supply and single phase clock are required.

One of the unique freatures of the Z80-PIO that separates it from other interface controllers is that all data transfer between the peripheral device and the CPU is accomplished under total interrupt control. The interrupt logic of the PIO permits full usage of the efficient interrupt capabilities of the Z80-CPU during I/O transfers. All logic necessary to implement a fully nested interrupt structure is included in the PIO so that additional circuits are not required. Another unique feature of the PIO is that it can be programmed to interrupt the CPU on the occurrence of specified status conditions in the peripheral device. For example, the PIO can be programmed to interrupt if any specified peripheral alarm conditions should occur. This interrupt capability reduces the amount of time that the processor must spend in polling peripheral status.

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.

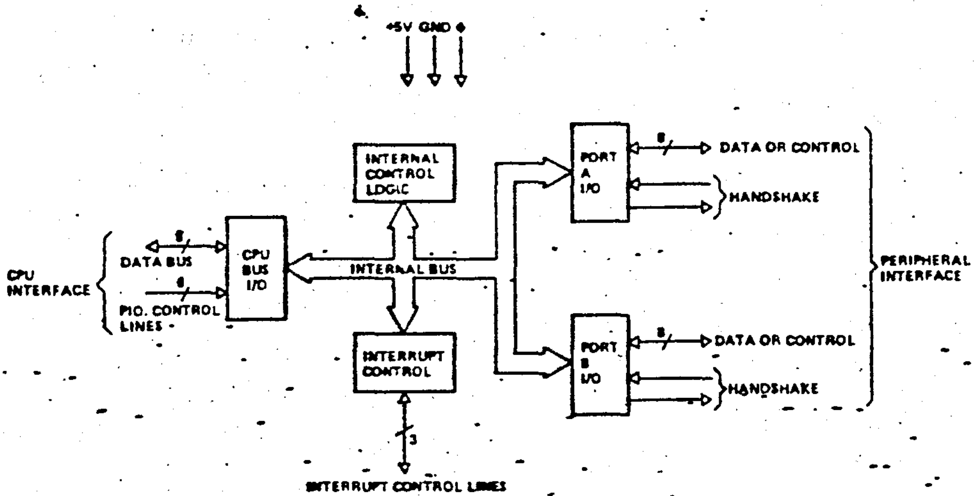


FIGURE 2.0-1
PIO BLOCK DIAGRAM

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.

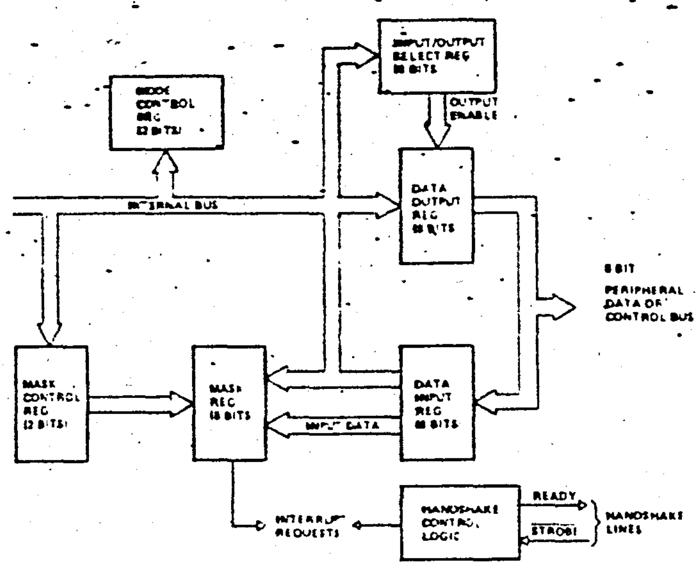


FIGURE 2.0-2
PORT I/O BLOCK DIAGRAM

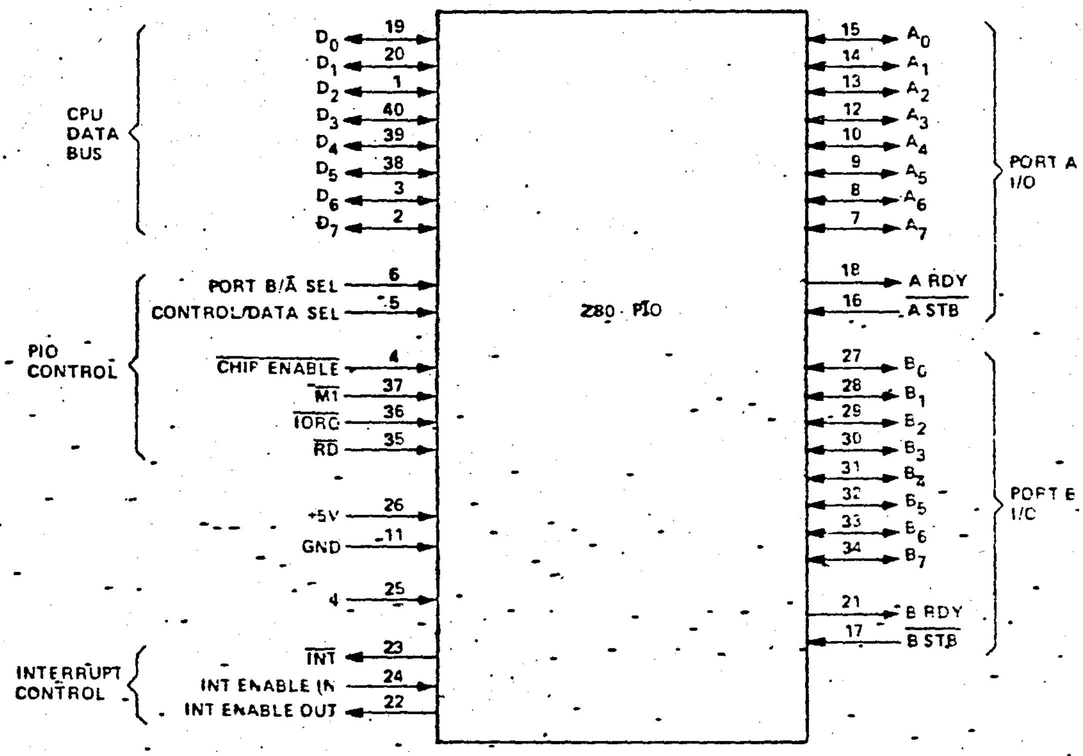
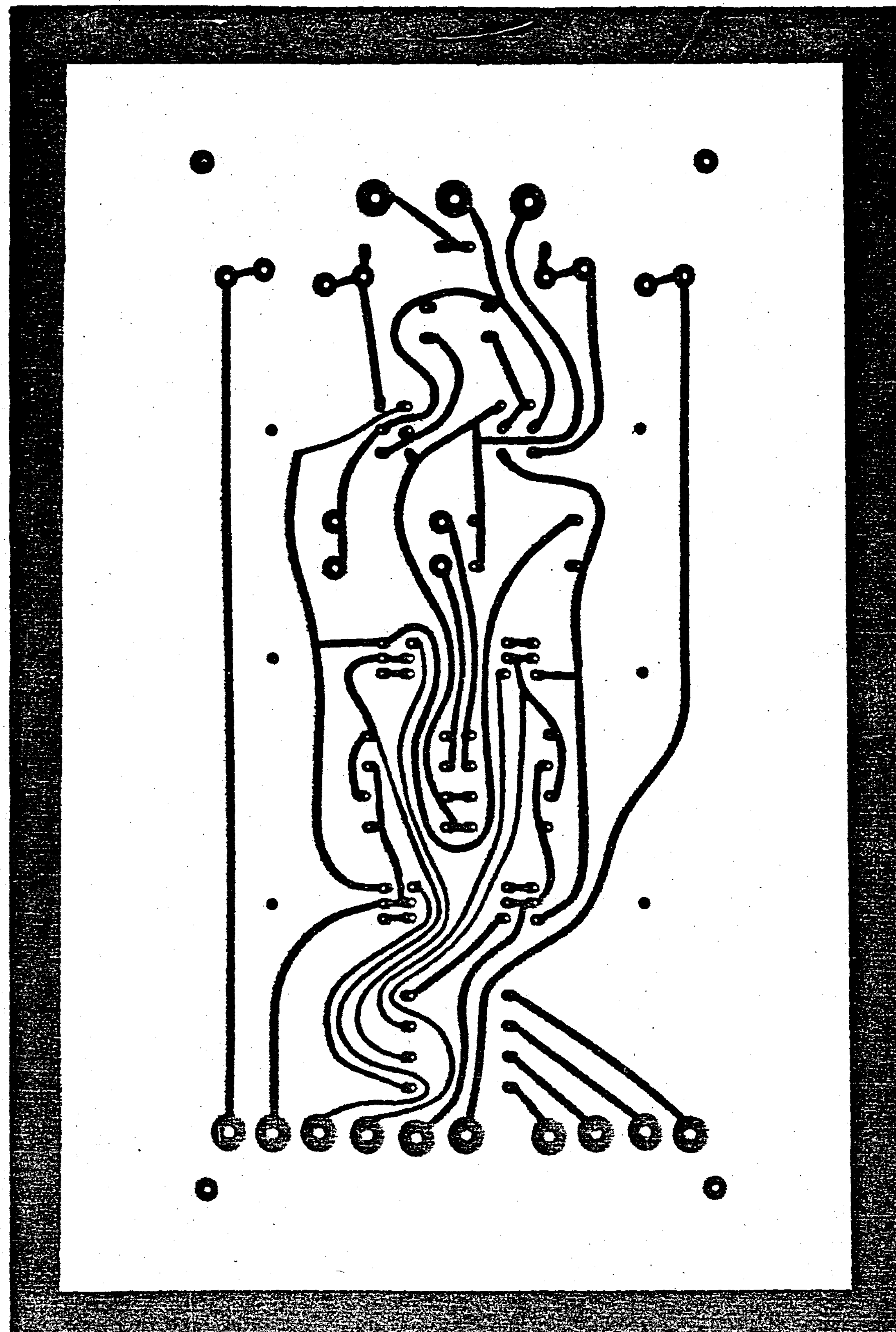
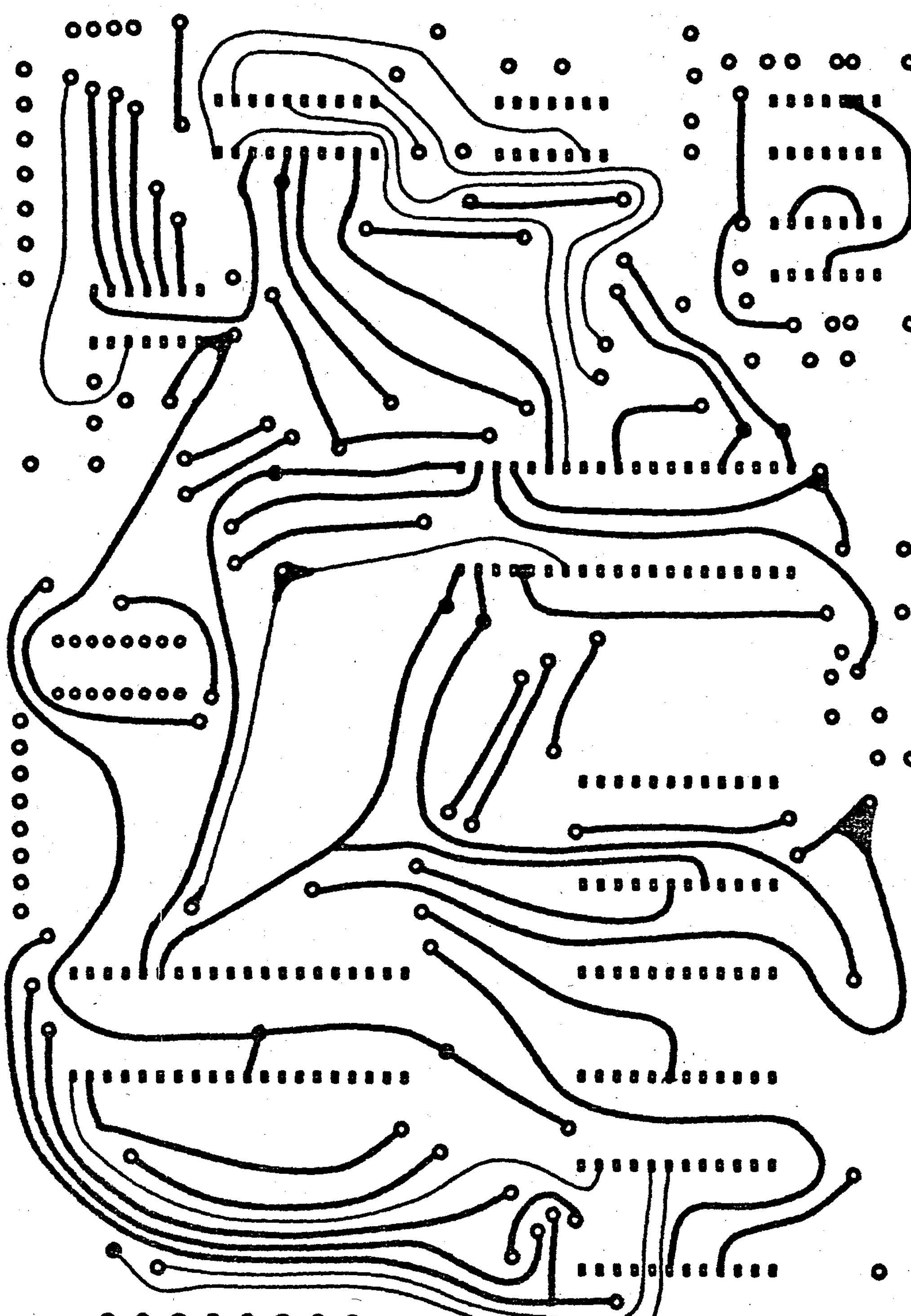


FIGURE 3.0-1
PIO PIN CONFIGURATION

APPENDIX F

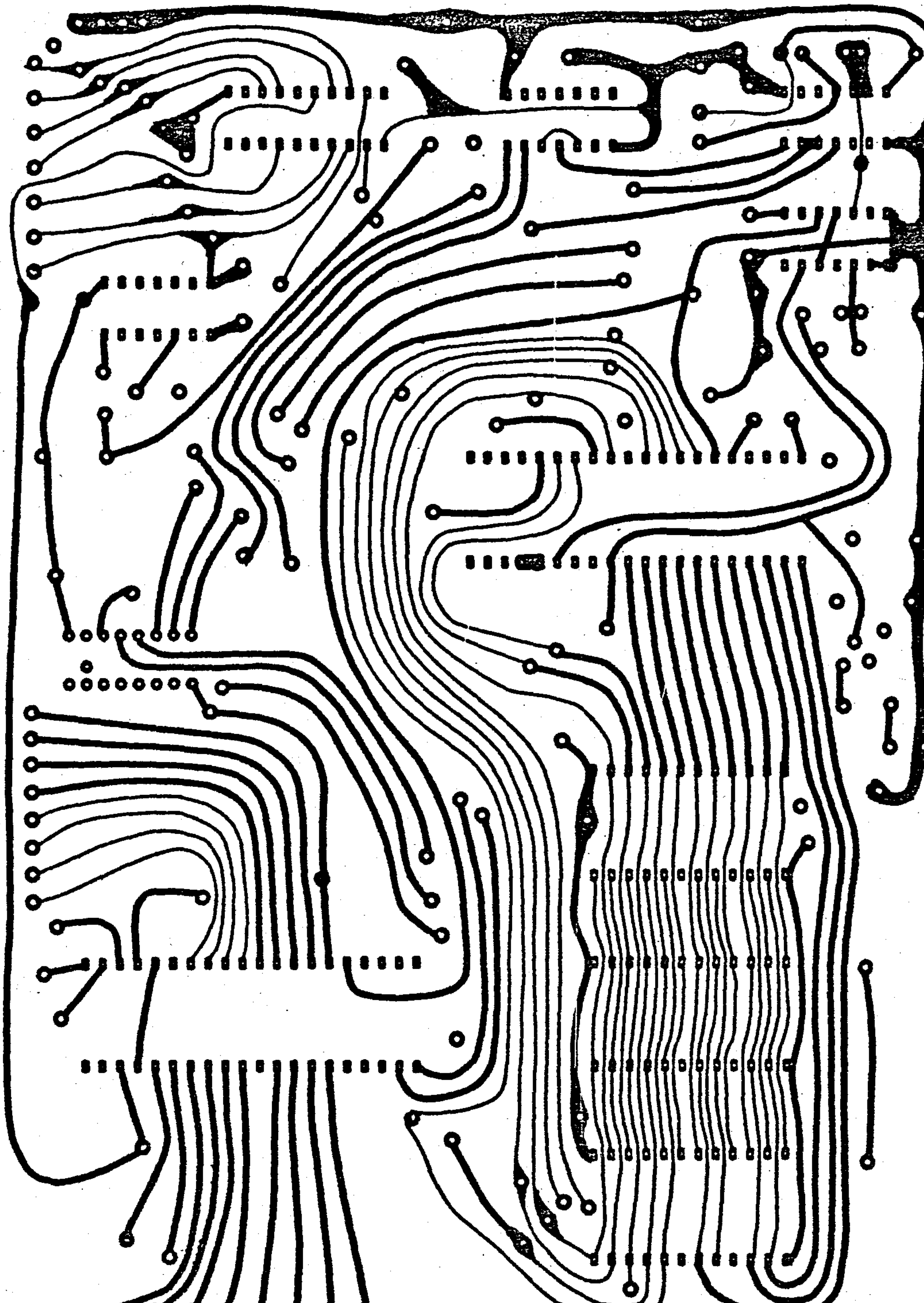


Stepper Motor Drive Card



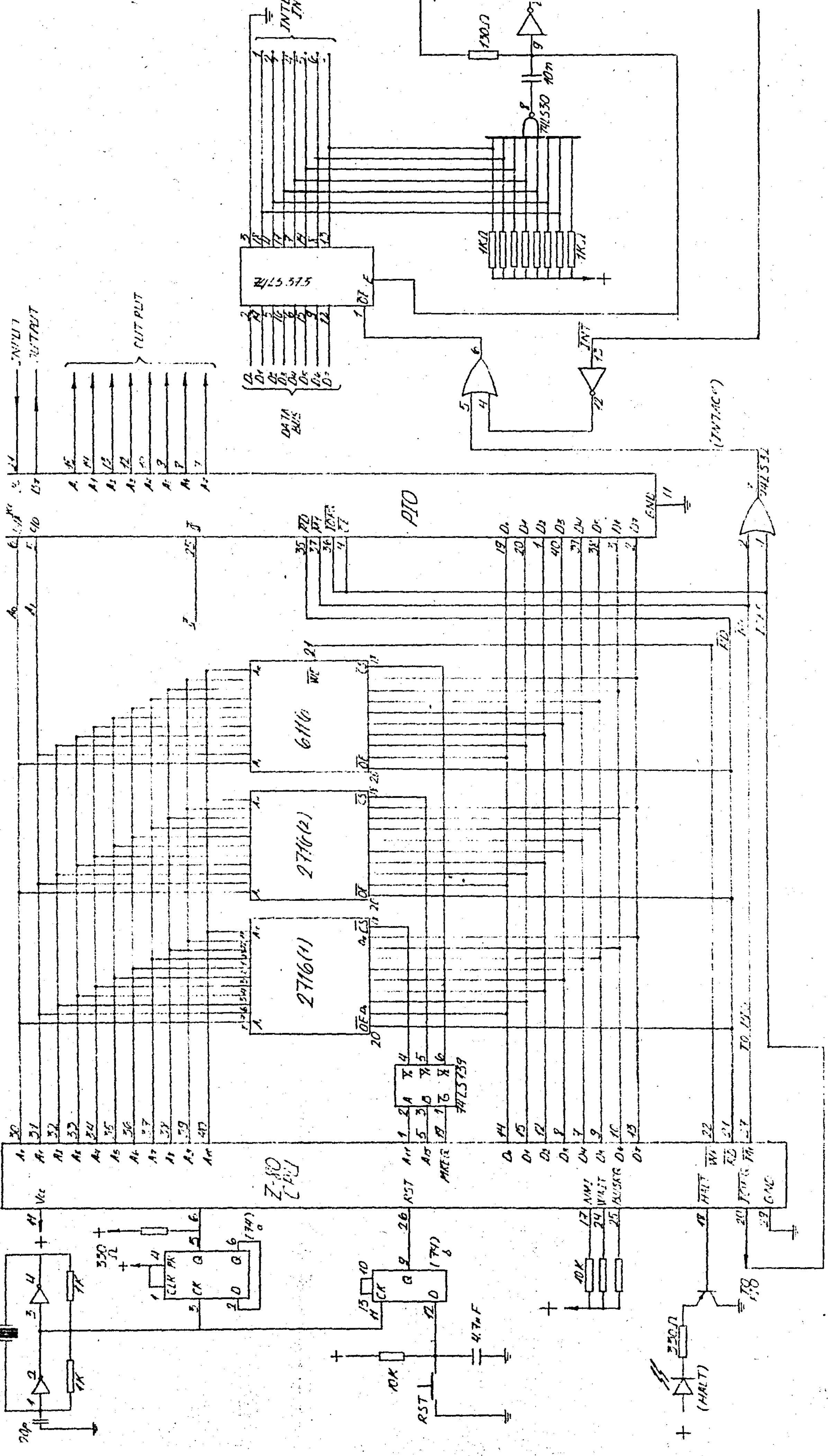
(Component side)

Z-80 Microprocessor Card



(Solder side)

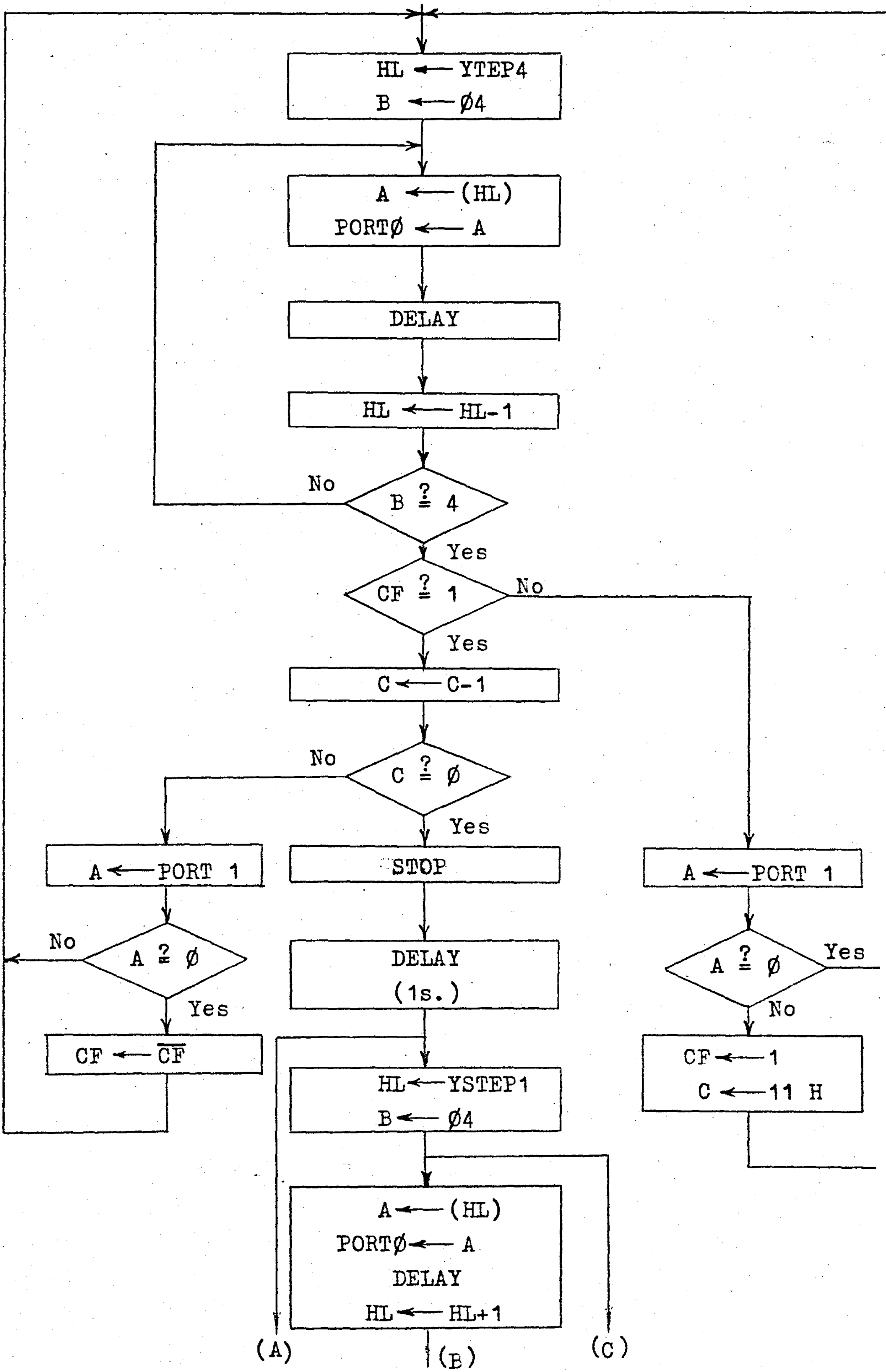
Z-80 Microprocessor Card

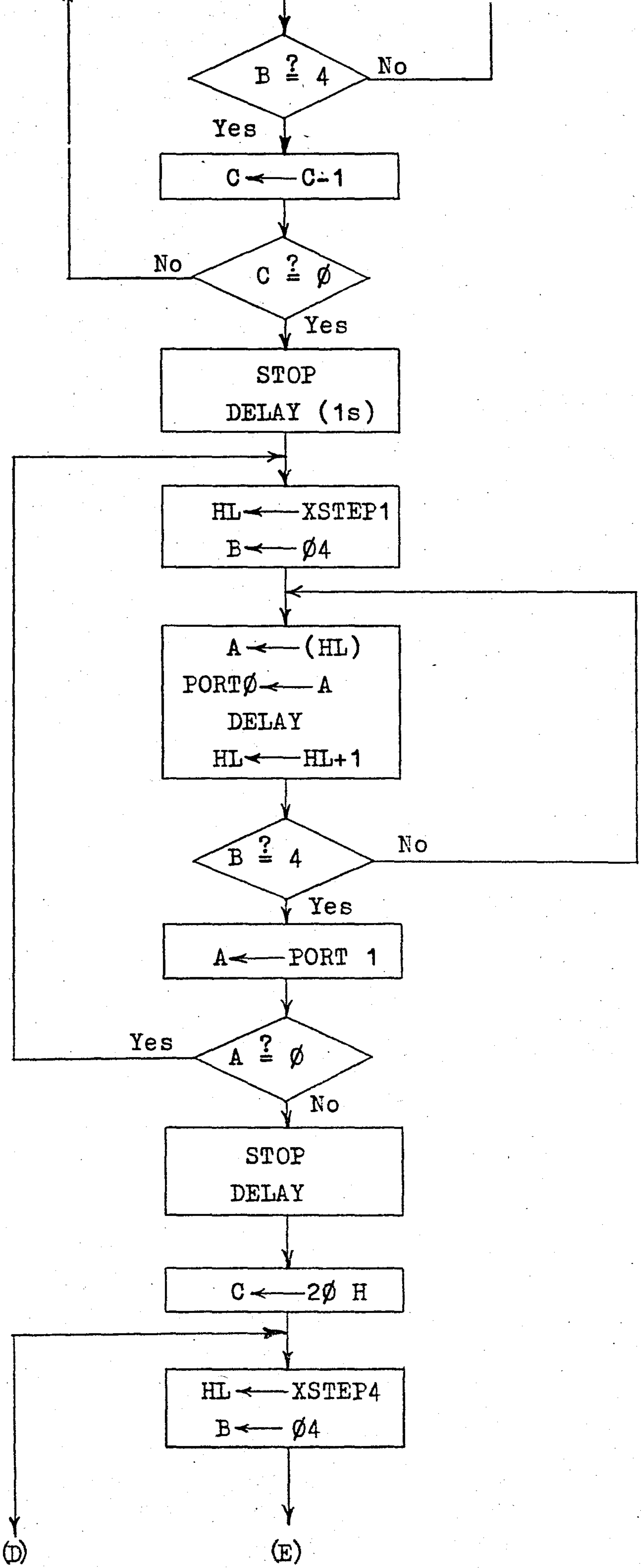


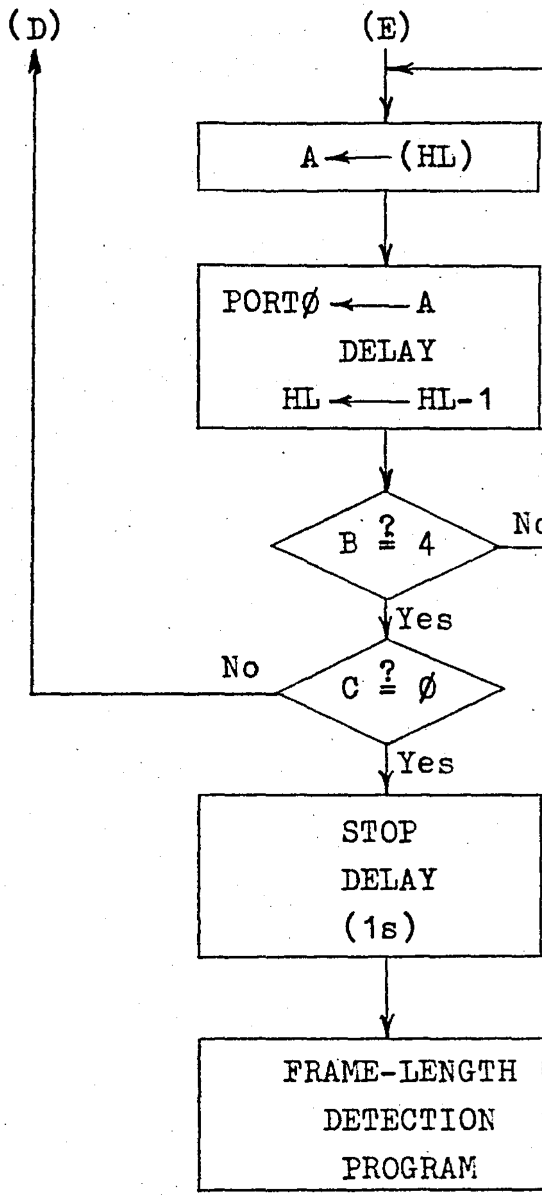
Z-80 MICRO PROCESSOR CARD

APPENDIX H

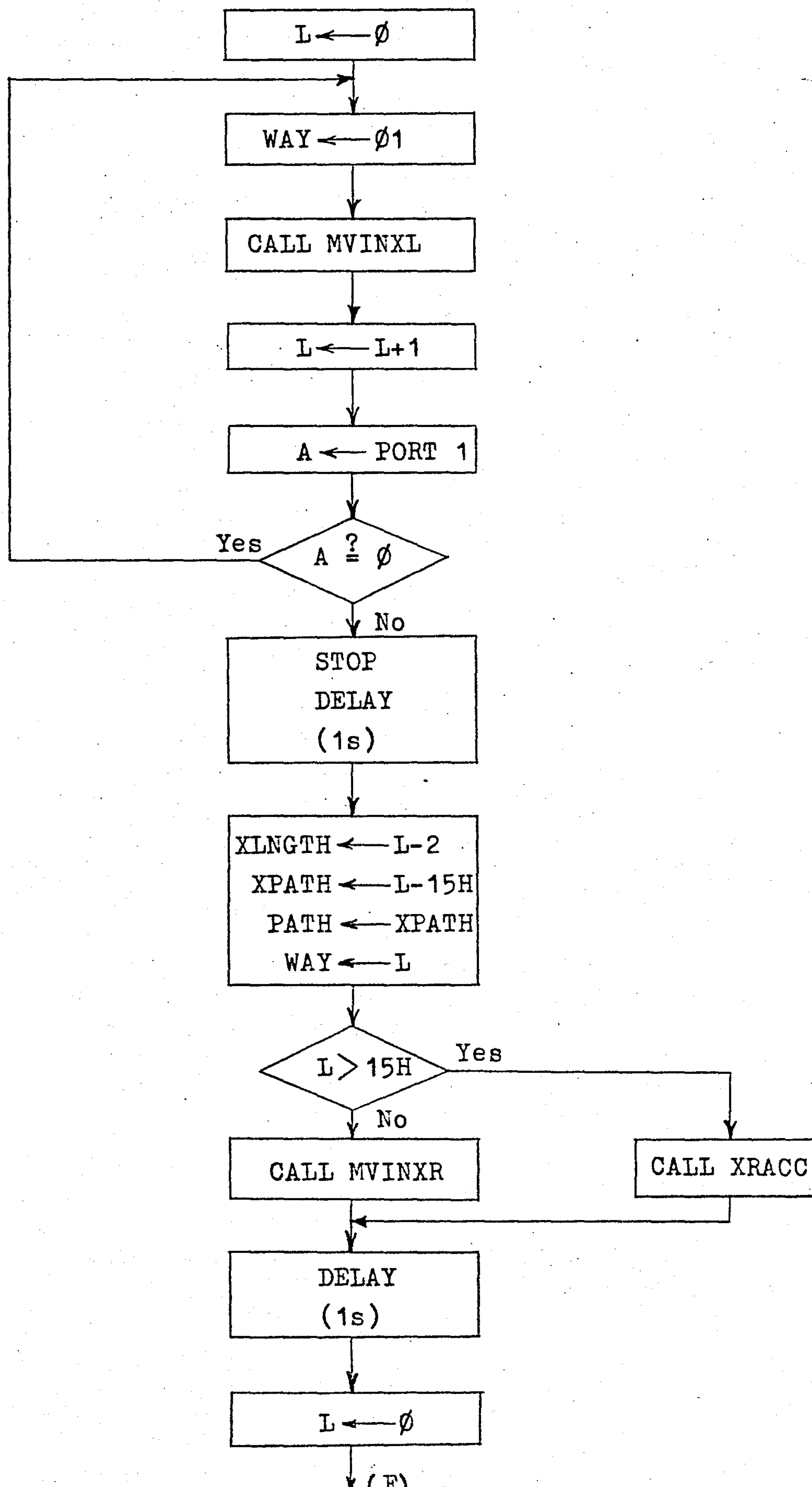
Frame Line Detection Program

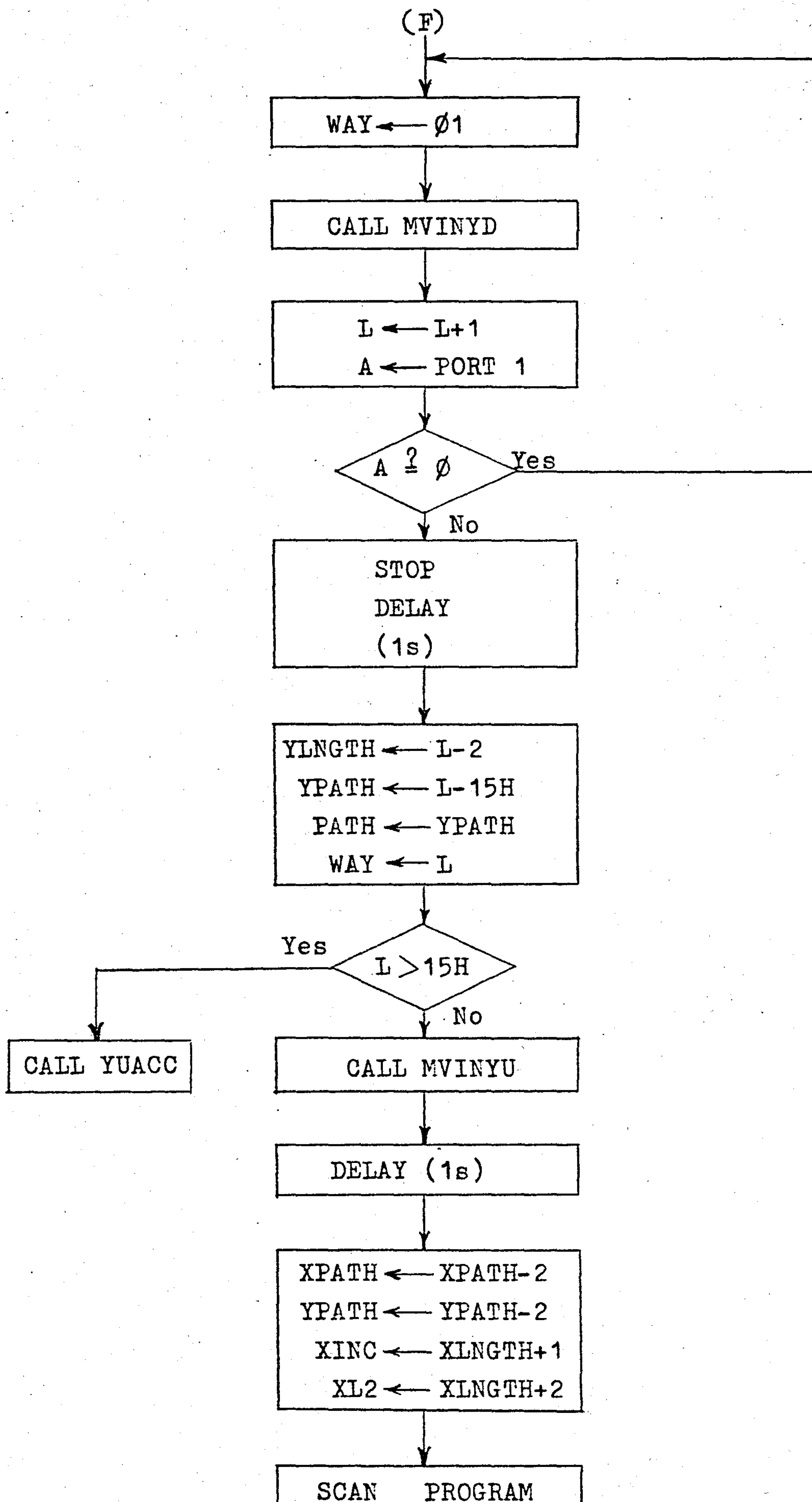



B ? 4
No
Yes
C ← C-1
No
C ? Ø
Yes
STOP
DELAY (1s)
HL ← XSTEP1
B ← Ø4
A ← (HL)
PORTØ ← A
DELAY
HL ← HL+1
B ? 4
No
Yes
A ← PORT 1
Yes
A ? Ø
No
STOP
DELAY
C ← 2Ø H
HL ← XSTEP4
B ← Ø4
(D)
(E)

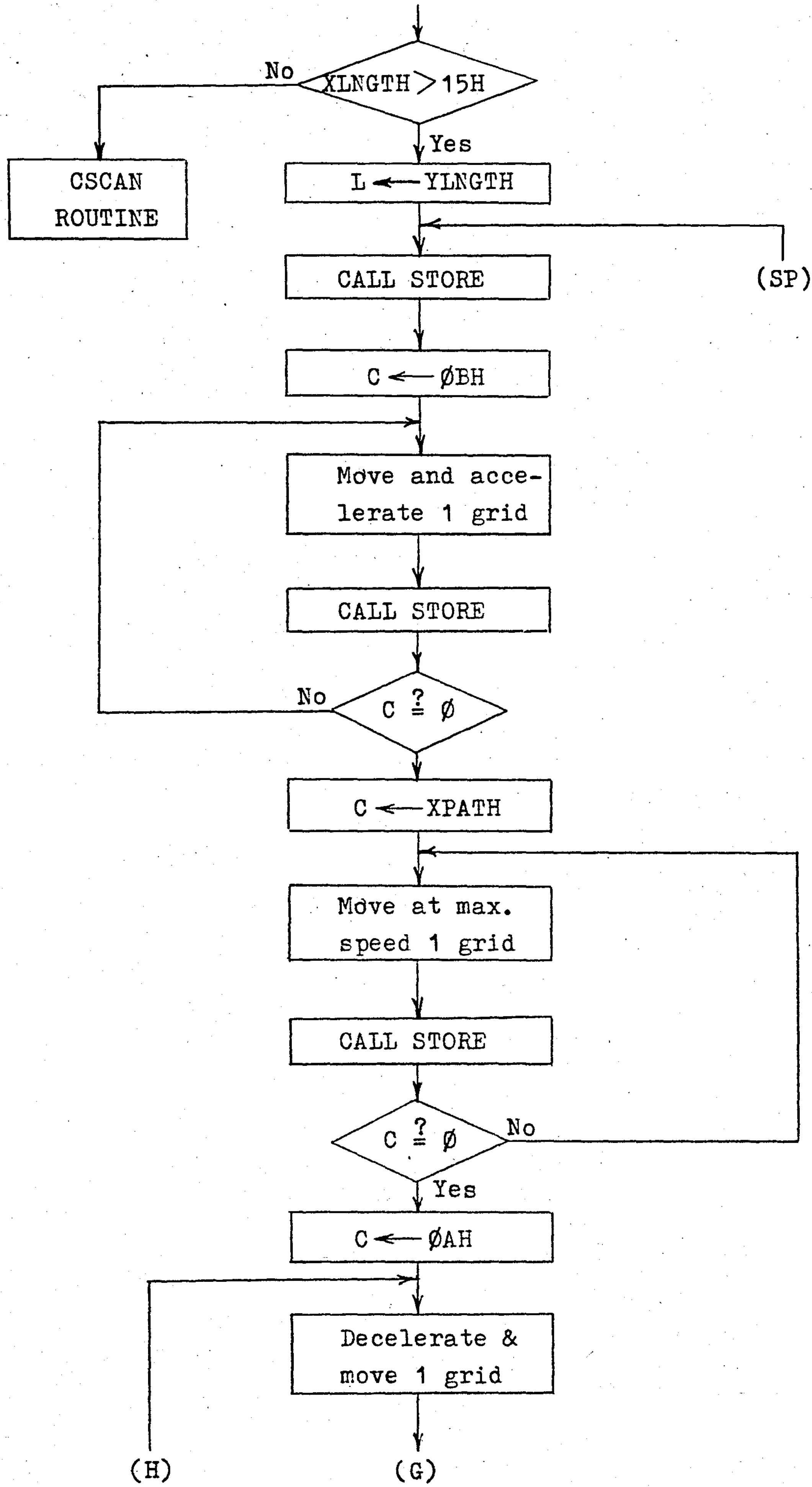

(D)
(E)
A ← (HL)
PORTØ ← A
DELAY
HL ← HL-1
B ? = 4
No
Yes
C ? = Ø
No
Yes
STOP
DELAY
(1s)
FRAME-LENGTH
DETECTION
PROGRAM

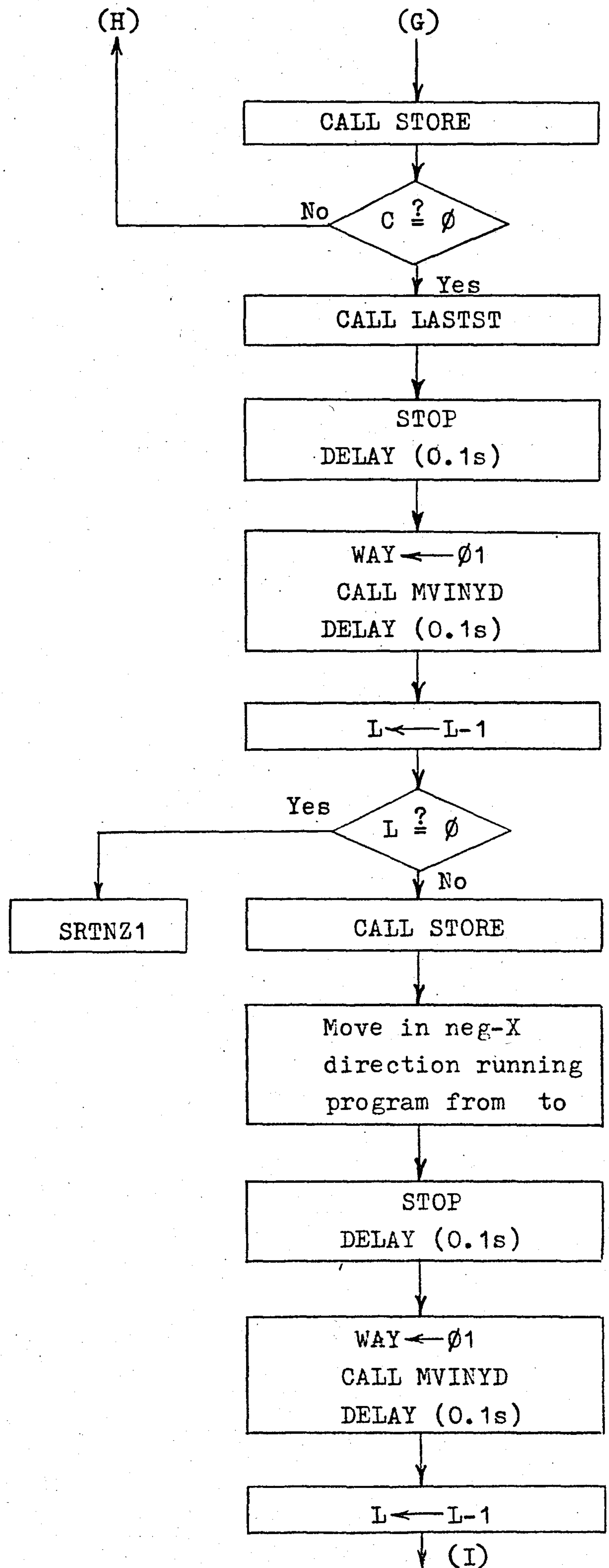
Frame-length Detection Program

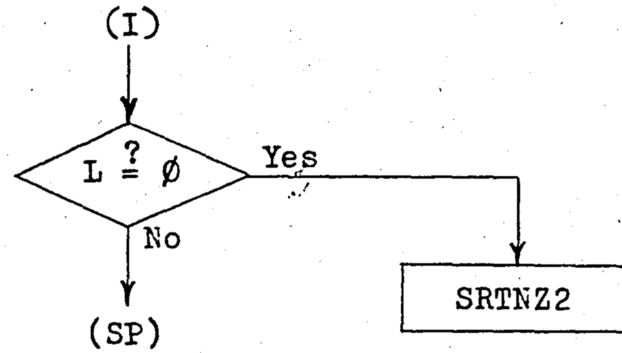



(F)
WAY ← Ø1
CALL MVINYD
L ← L+1
A ← PORT 1
A ≟ Ø
Yes
No
STOP
DELAY
(1s)
YLNGTH ← L-2
YPATH ← L-15H
PATH ← YPATH
WAY ← L
L > 15H
Yes
No
CALL YUACC
CALL MVINYU
DELAY (1s)
XPATH ← XPATH-2
YPATH ← YPATH-2
XINC ← XLNGTH+1
XL2 ← XLNGTH+2
SCAN PROGRAM

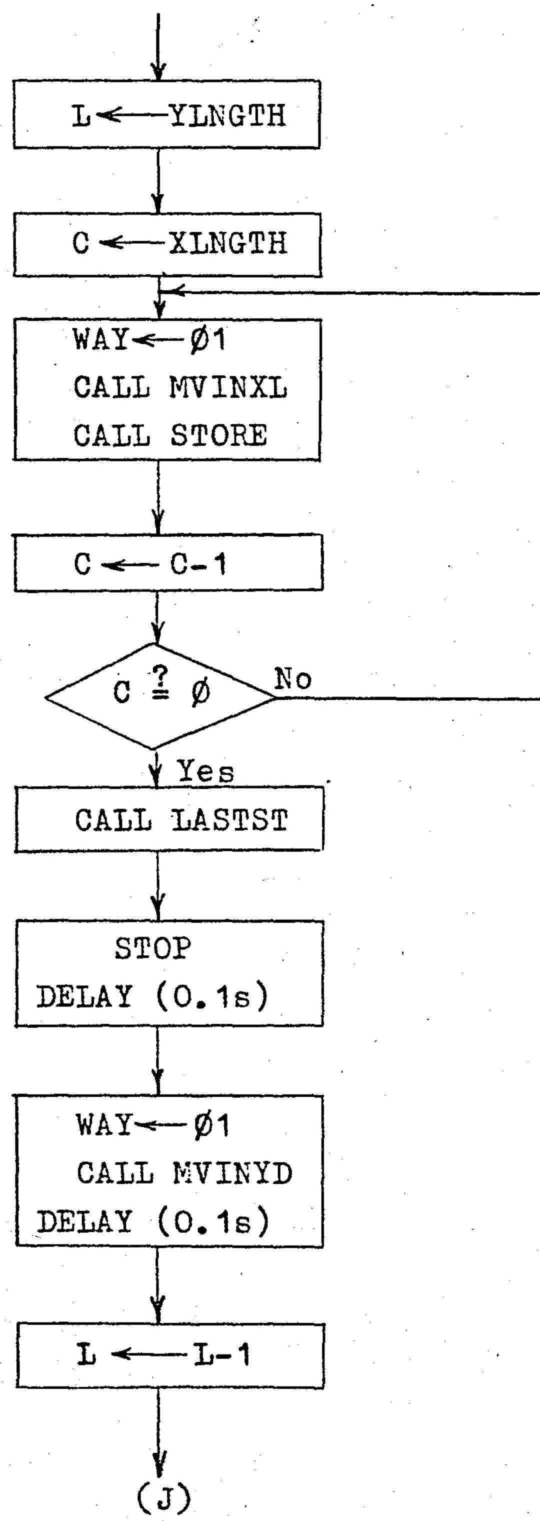
Scanning Program

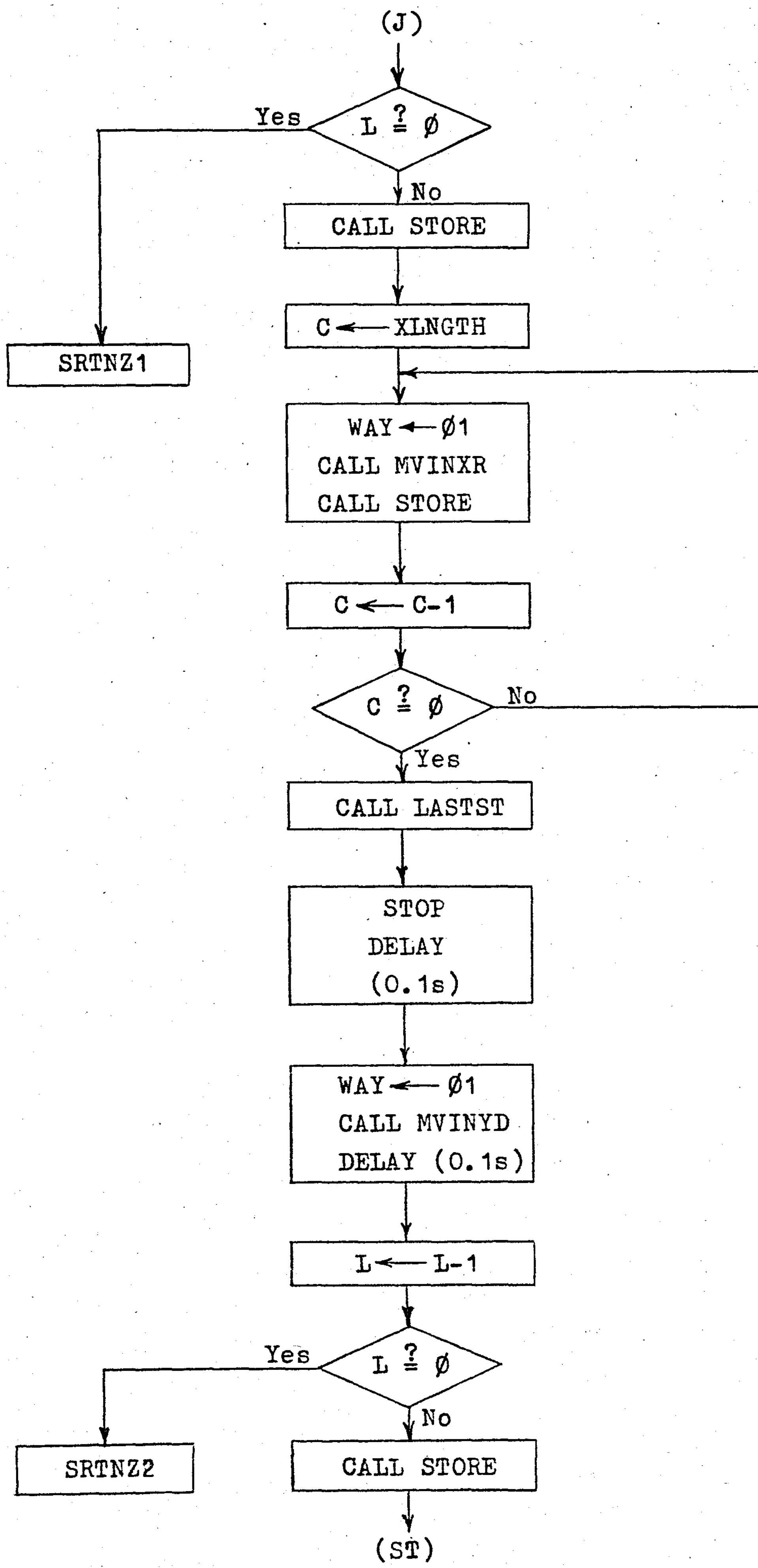



(H)
(G)
CALL STORE
C ?= ∅
No
Yes
CALL LASTST
STOP
DELAY (0.1s)
WAY ← ∅1
CALL MVINYD
DELAY (0.1s)
L ← L-1
L ?= ∅
Yes
No
SRTNZ1
CALL STORE
Move in neg-X
direction running
program from to
STOP
DELAY (0.1s)
WAY ← ∅1
CALL MVINYD
DELAY (0.1s)
L ← L-1
(I)

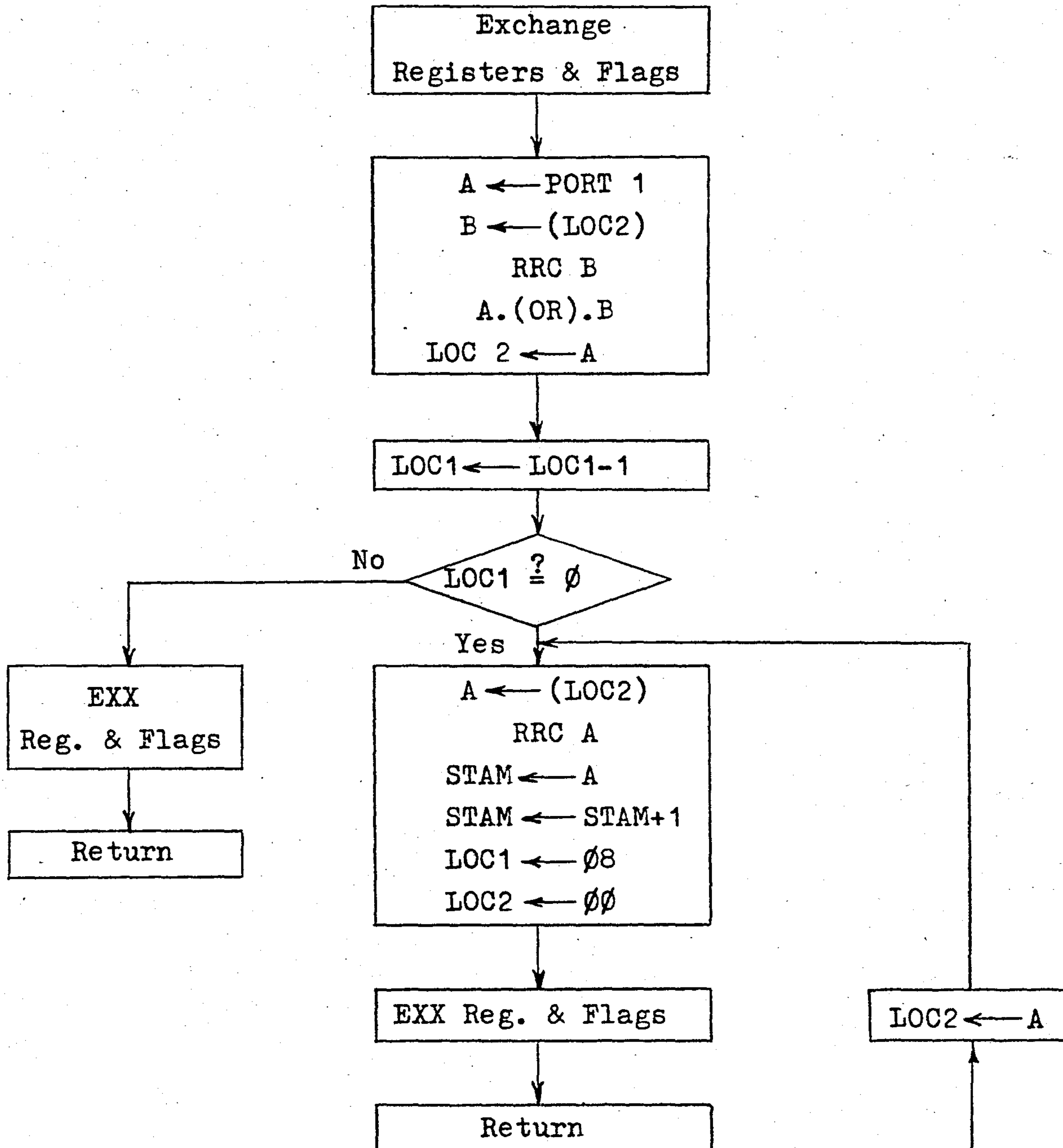


Constant speed scanning (CSCAN) Routine

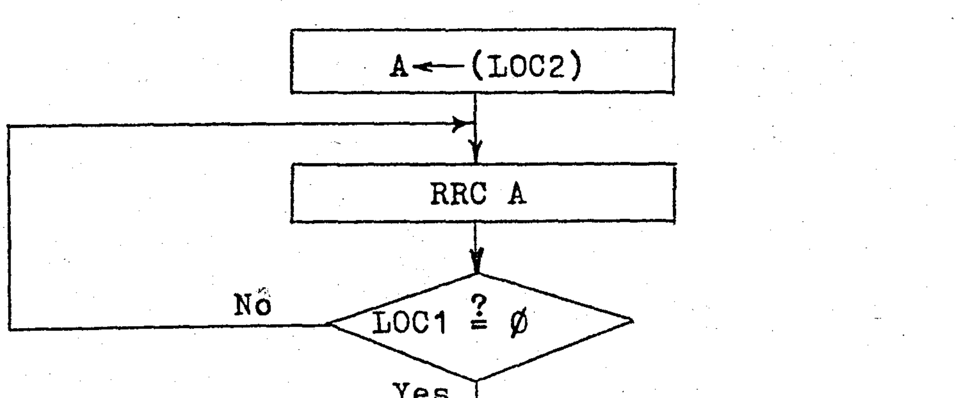



(J)
Yes
L ? ∅
No
CALL STORE
C ← XLNGTH
SRTNZ1
WAY ← ∅1
CALL MVINXR
CALL STORE
C ← C-1
C ? ∅
No
Yes
CALL LASTST
STOP
DELAY
(0.1s)
WAY ← ∅1
CALL MVINYD
DELAY (0.1s)
L ← L-1
Yes
L ? ∅
No
SRTNZ2
CALL STORE
(ST)

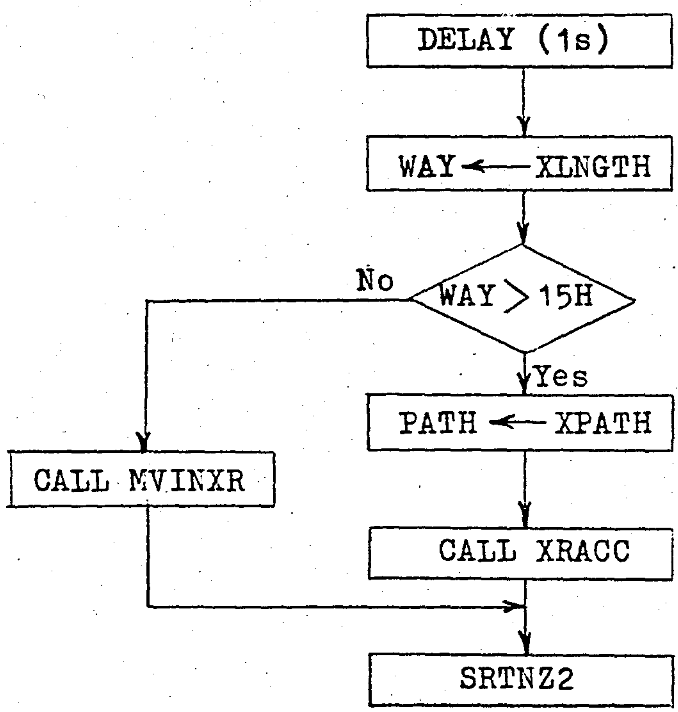
Store Routine



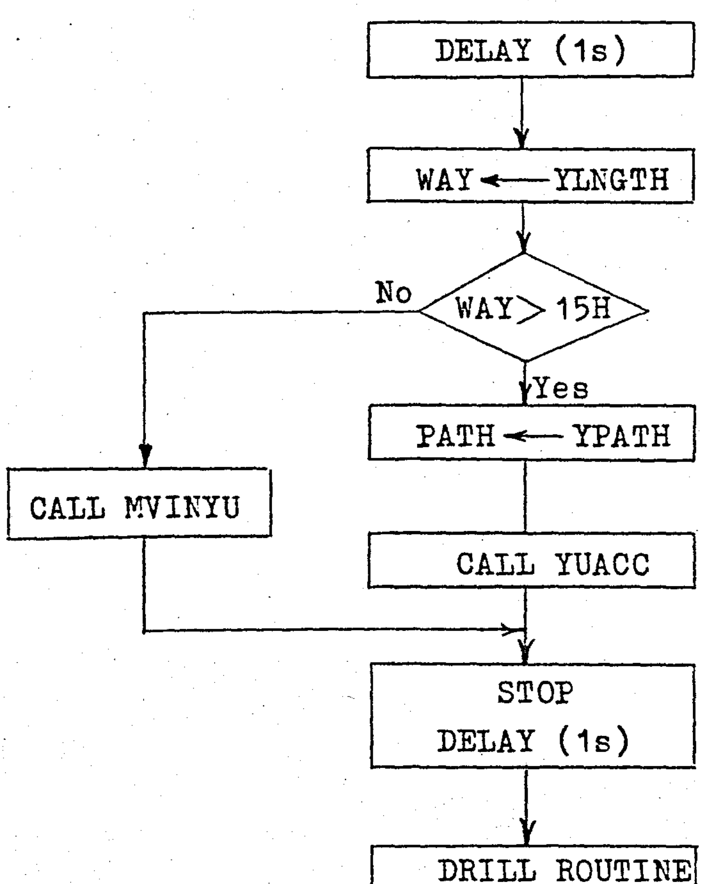
LASTST ROUTINE



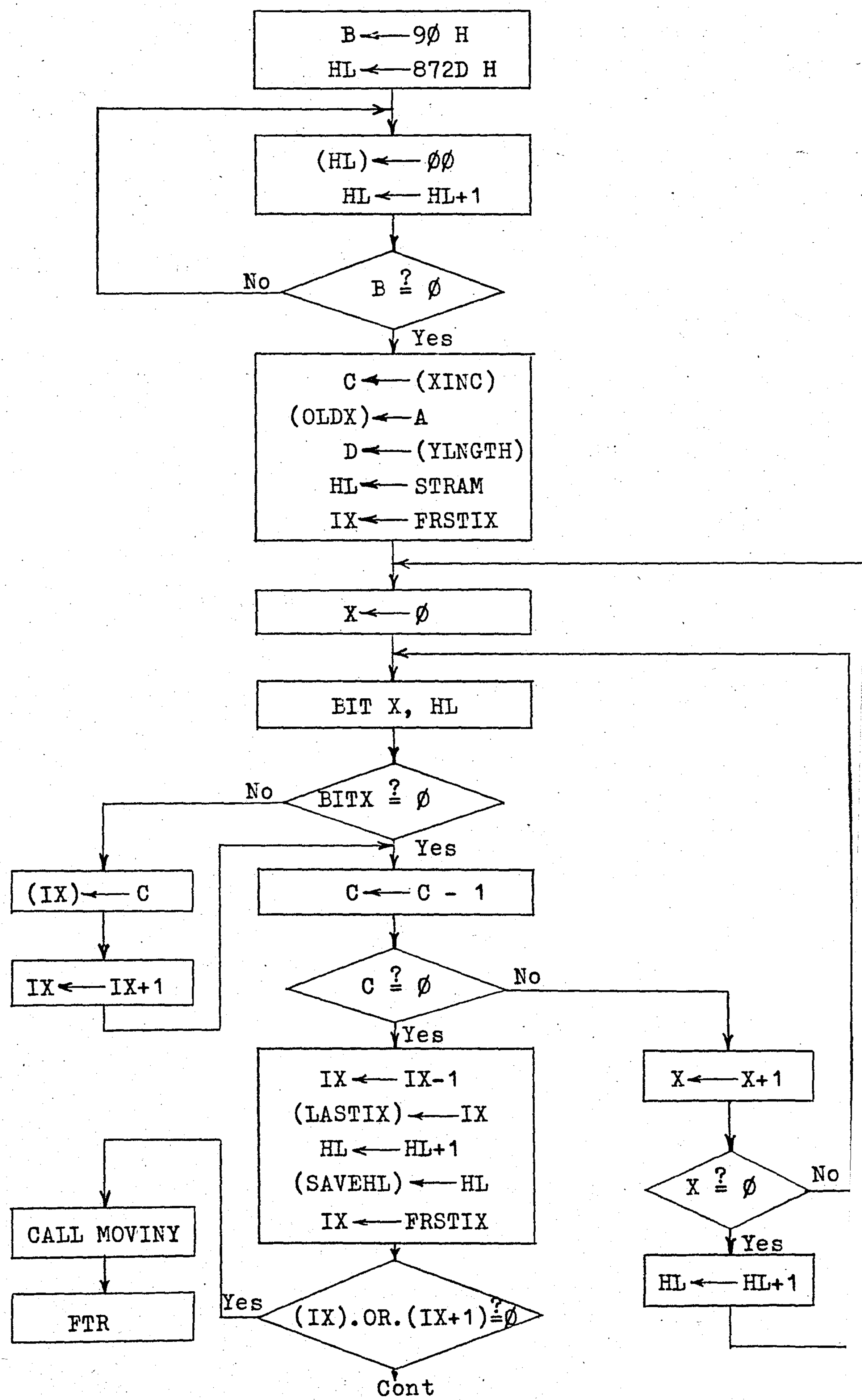
SRTNZ1 Routine

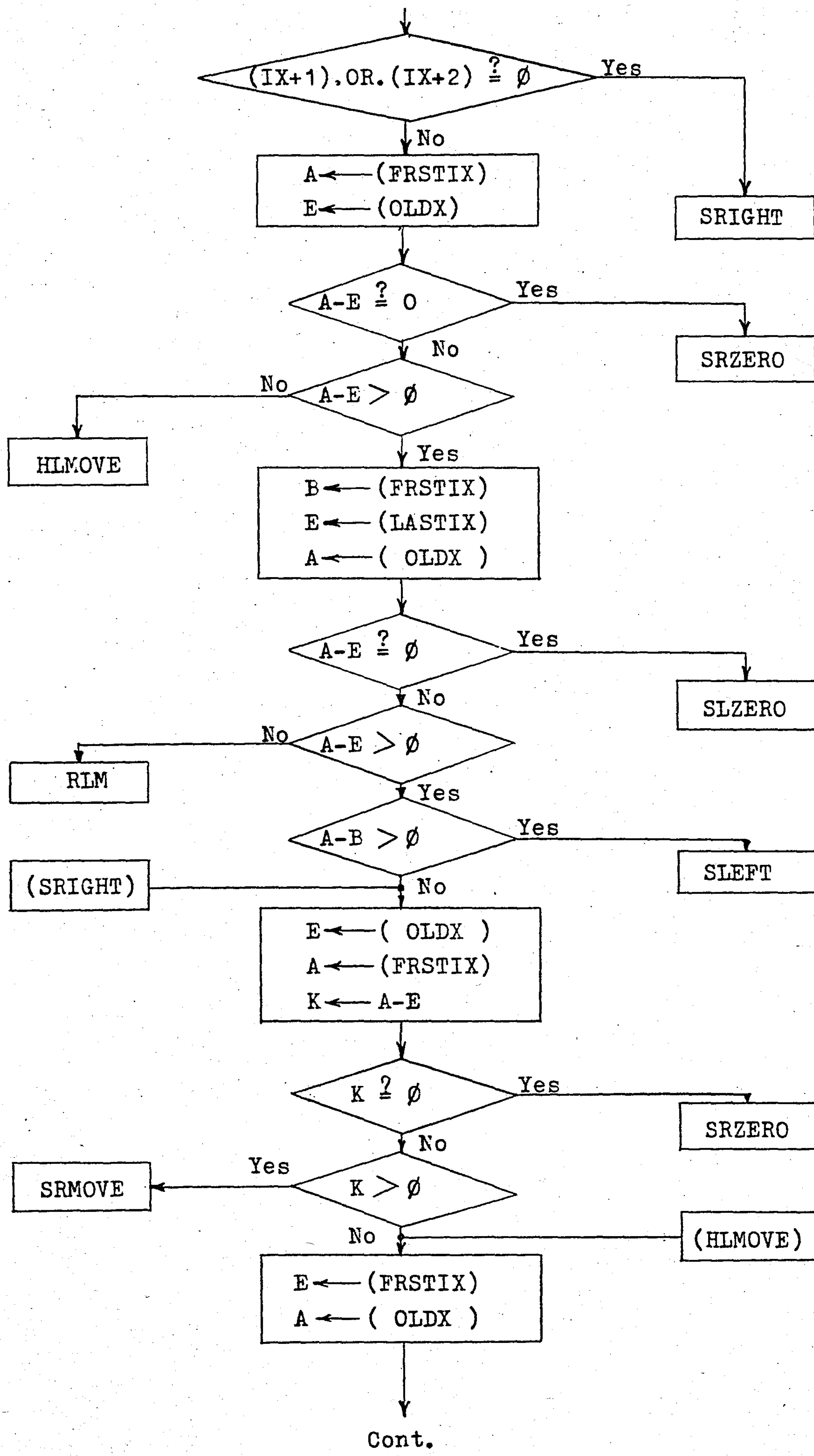


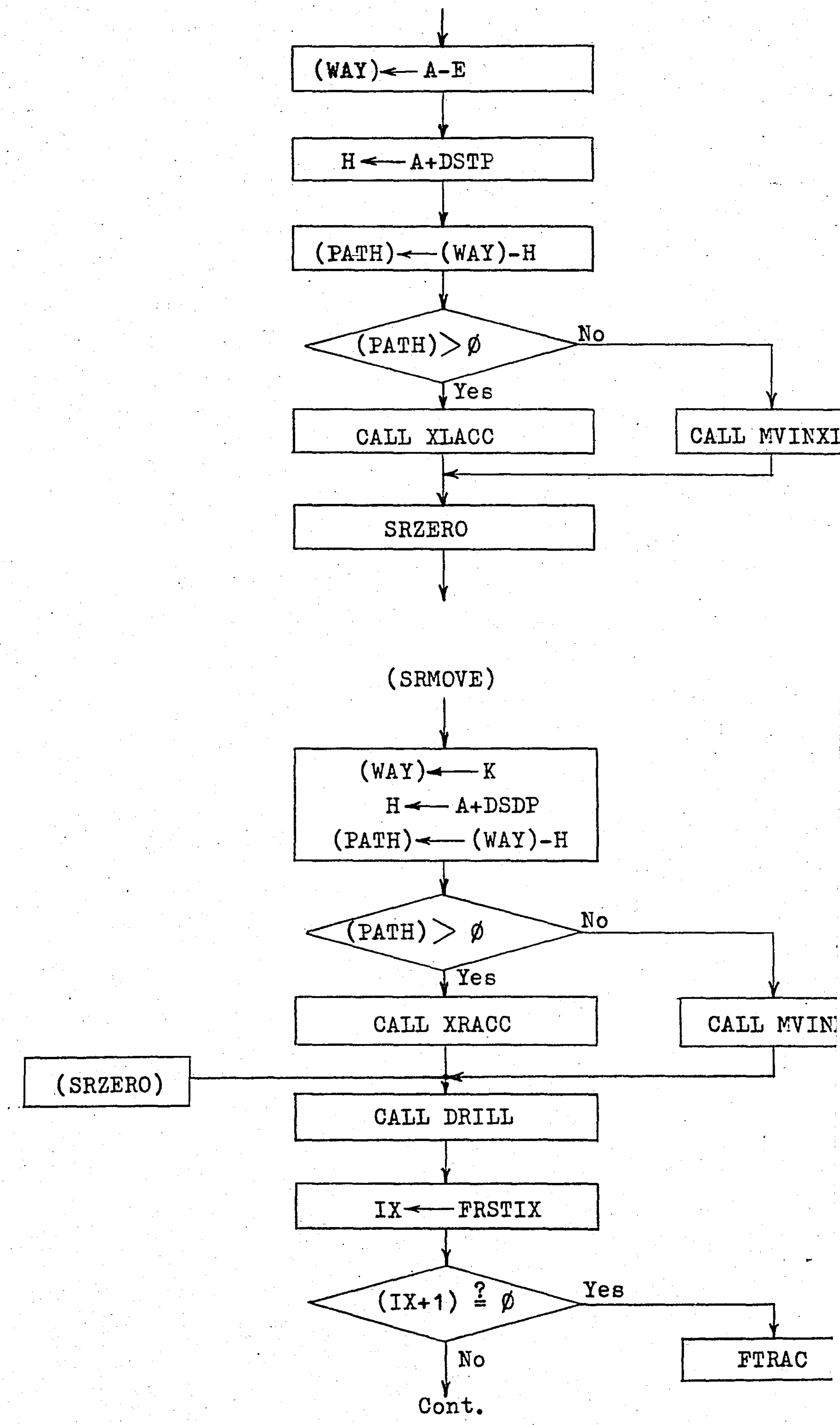
SRTNZ2 Routine



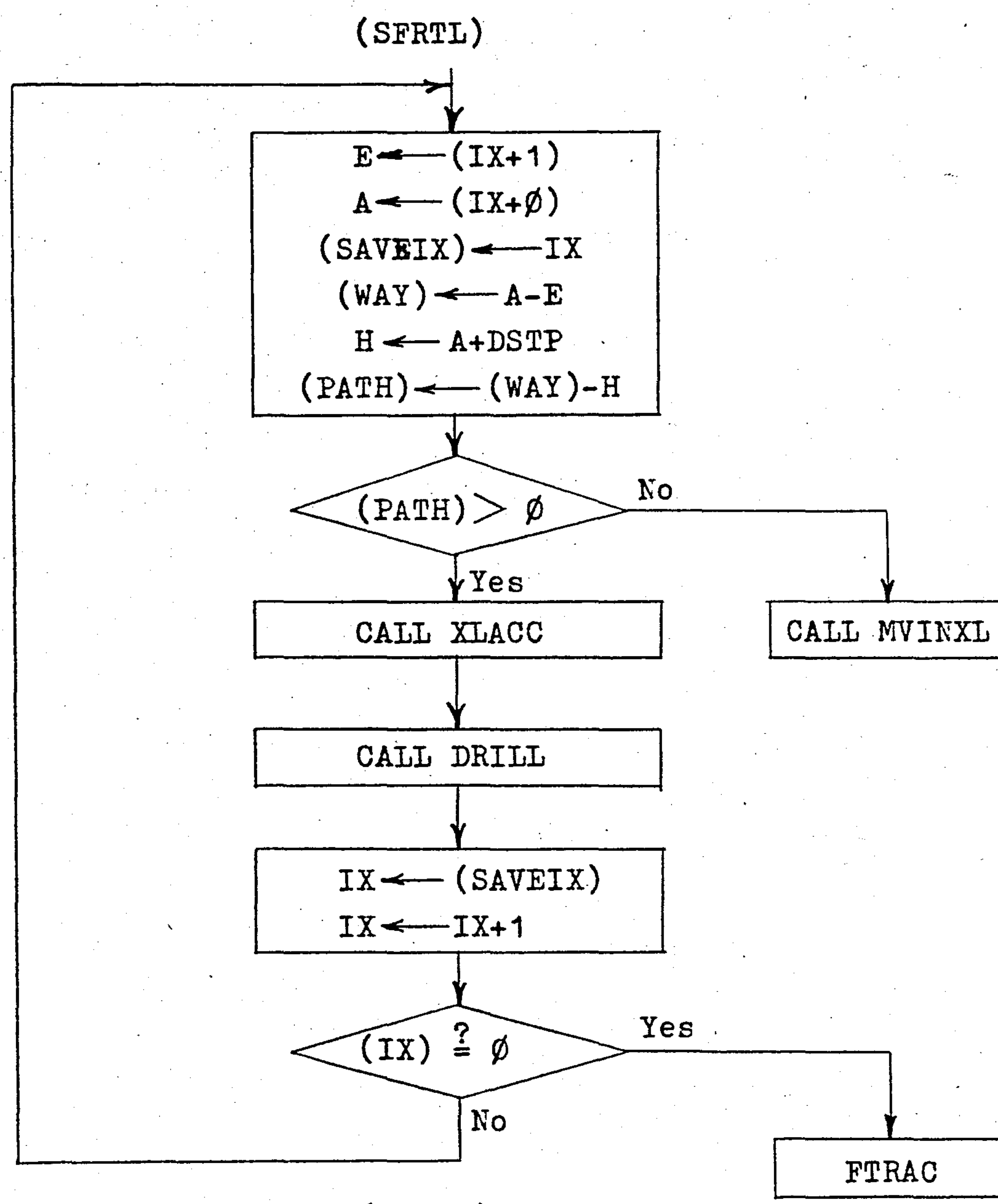
Drill Program (For odd lines)

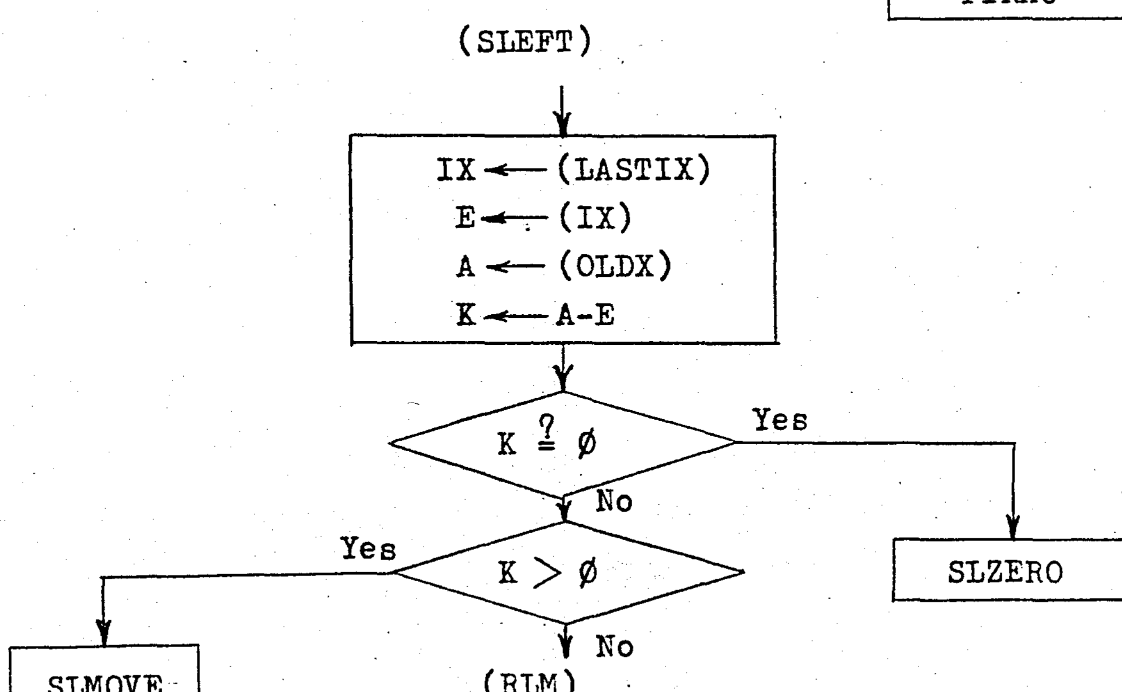


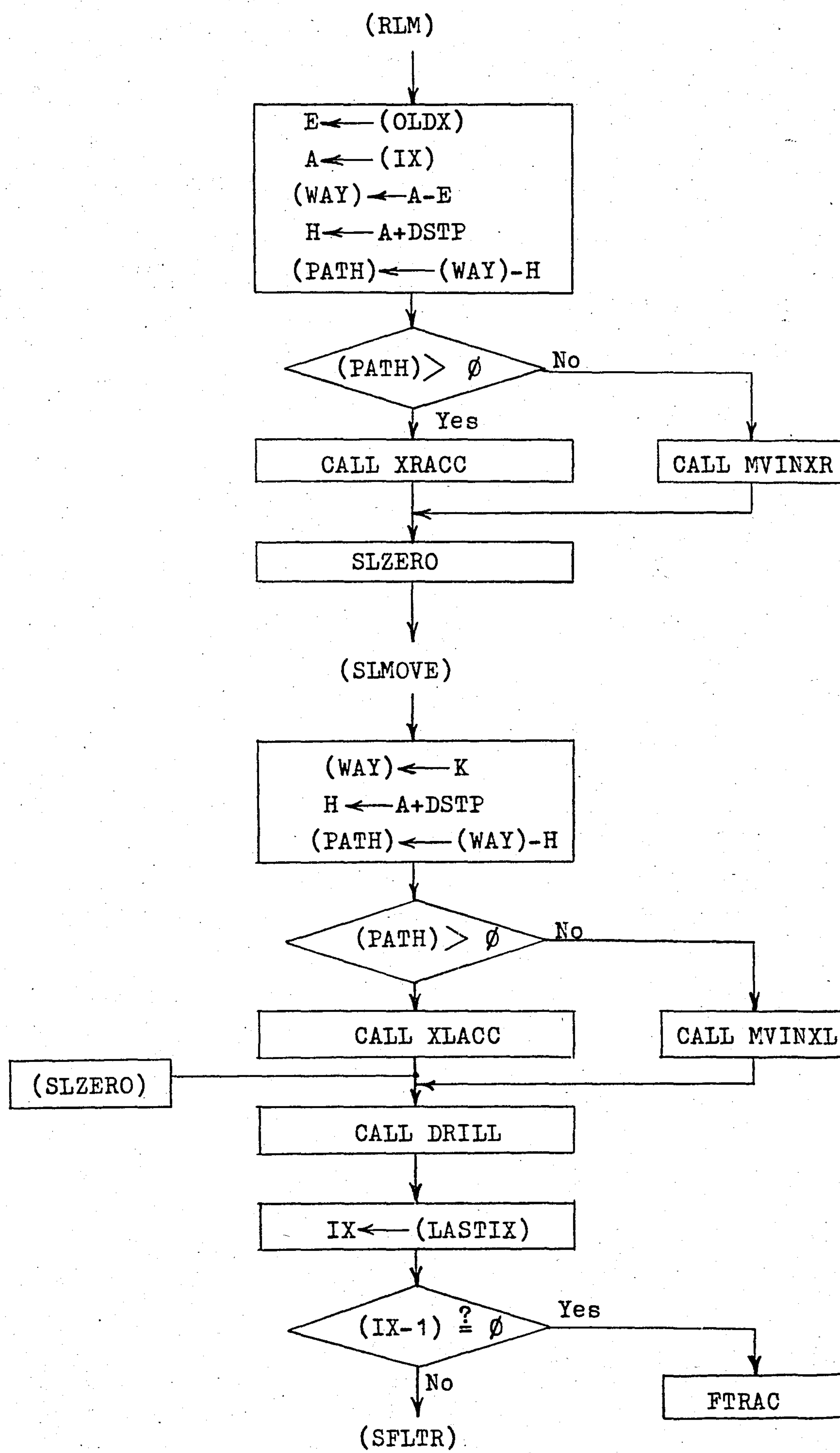

(IX+1).OR.(IX+2) ? = Ø
Yes
No
SRIGHT
A ← (FRSTIX)
E ← (OLDX)
A-E ? = 0
SRZERO
A-E > Ø
HLMOVE
B ← (FRSTIX)
E ← (LASTIX)
A ← (OLDX)
A-E ? = Ø
SLZERO
A-E > Ø
RLM
A-B > Ø
SLEFT
(SRIGHT)
E ← (OLDX)
A ← (FRSTIX)
K ← A-E
K ? = Ø
SRZERO
K > Ø
SRMOVE
(HLMOVE)
E ← (FRSTIX)
A ← (OLDX)
Cont.

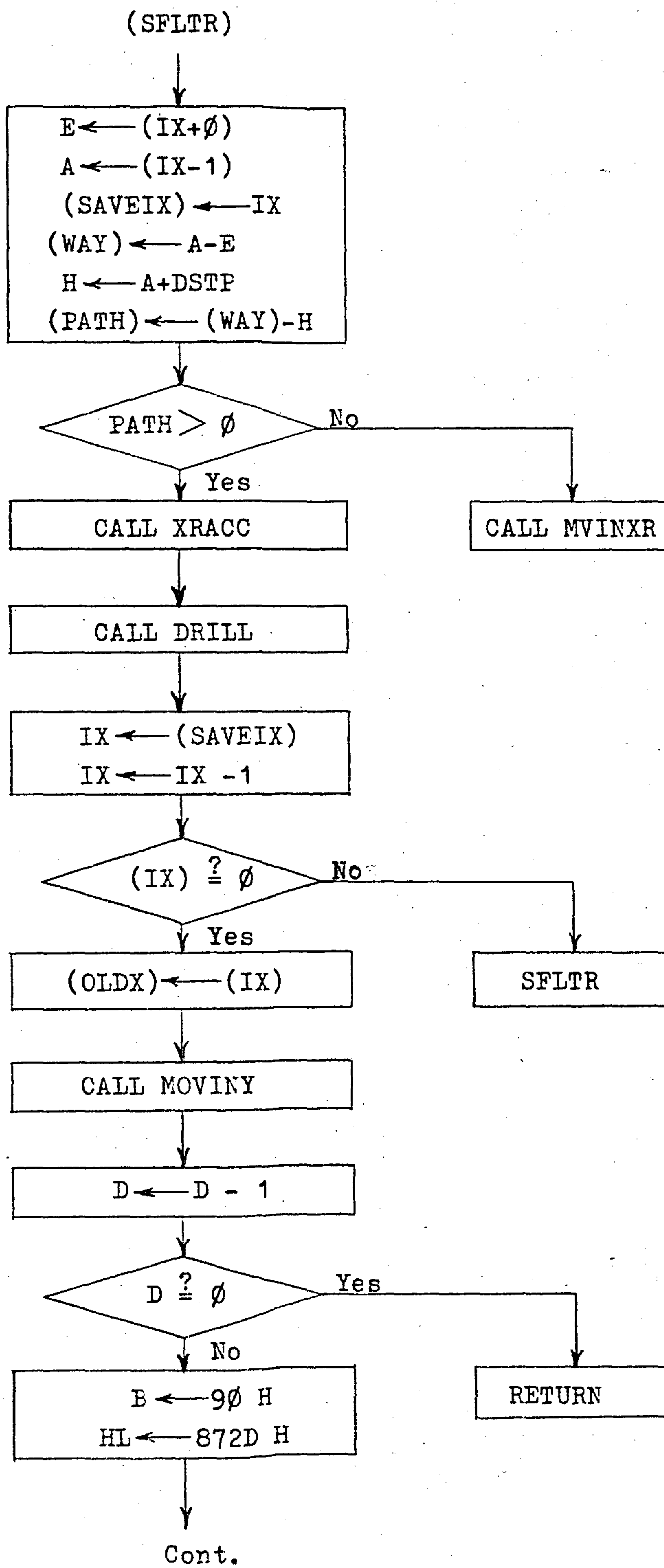


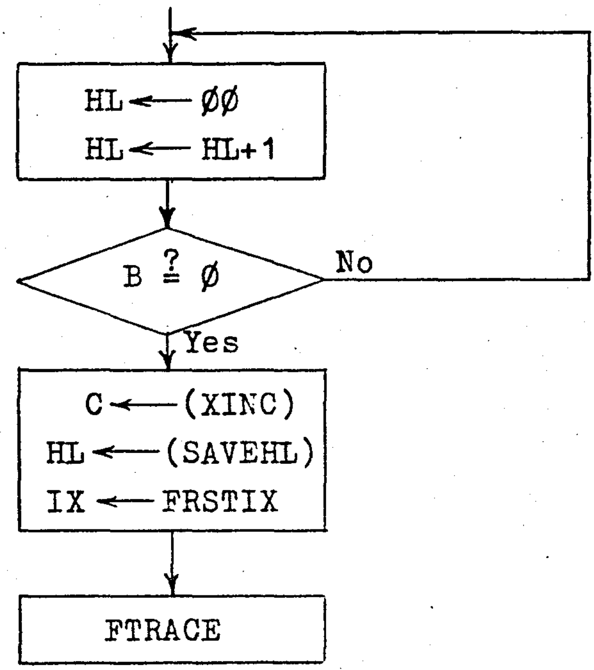
(WAY)←A-E
H←A+DSTP
(PATH)←(WAY)-H
(PATH)>Ø
No
Yes
CALL XLACC
CALL MVINXL
SRZERO
(SRMOVE)
(WAY)←K
H←A+DSDP
(PATH)←(WAY)-H
(PATH)>Ø
No
Yes
CALL XRACC
CALL MVIN
(SRZERO)
CALL DRILL
IX←FRSTIX
(IX+1) ?= Ø
Yes
No
FTRAC
Cont.

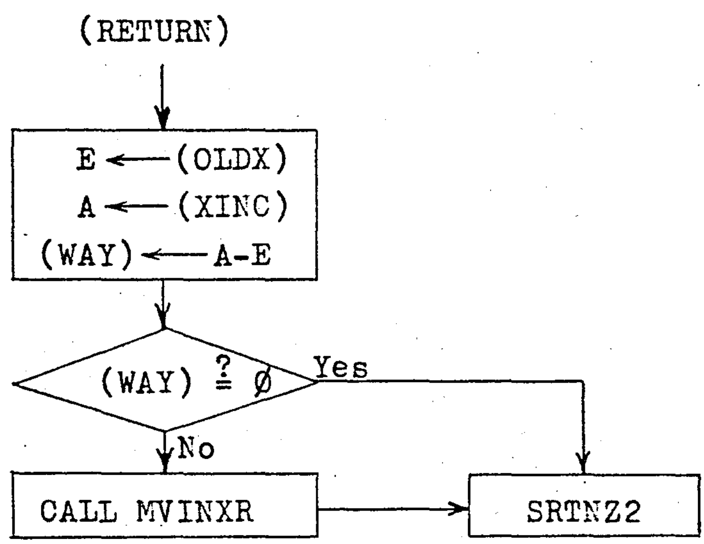

(SFRTL)
E ← (IX+1)
A ← (IX+Ø)
(SAVEIX) ← IX
(WAY) ← A-E
H ← A+DSTP
(PATH) ← (WAY)-H
(PATH) > Ø
No
Yes
CALL XLACC
CALL MVINXL
CALL DRILL
IX ← (SAVEIX)
IX ← IX+1
(IX) ?= Ø
Yes
No
FTRAC


(SLEFT)
IX ← (LASTIX)
E ← (IX)
A ← (OLDX)
K ← A-E
K ?= Ø
Yes
No
K > Ø
Yes
No
SLZERO
SLMOVE
(RLM)


(RLM)
E ← (OLDX)
A ← (IX)
(WAY) ← A-E
H ← A+DSTP
(PATH) ← (WAY)-H
(PATH) > Ø
No
Yes
CALL XRACC
CALL MVINXR
SLZERO
(SLMOVE)
(WAY) ← K
H ← A+DSTP
(PATH) ← (WAY)-H
(PATH) > Ø
No
CALL XLACC
CALL MVINXL
(SLZERO)
CALL DRILL
IX ← (LASTIX)
(IX-1) ?= Ø
Yes
No
FTRAC
(SFLTR)


(SFLTR)
E ← (IX+Ø)
A ← (IX-1)
(SAVEIX) ← IX
(WAY) ← A-E
H ← A+DSTP
(PATH) ← (WAY)-H
PATH > Ø
No
Yes
CALL XRACC
CALL MVINXR
CALL DRILL
IX ← (SAVEIX)
IX ← IX -1
(IX) ?= Ø
No
Yes
(OLDX) ← (IX)
SFLTR
CALL MOVINY
D ← D - 1
D ?= Ø
Yes
No
B ← 9Ø H
HL ← 872D H
RETURN
Cont.



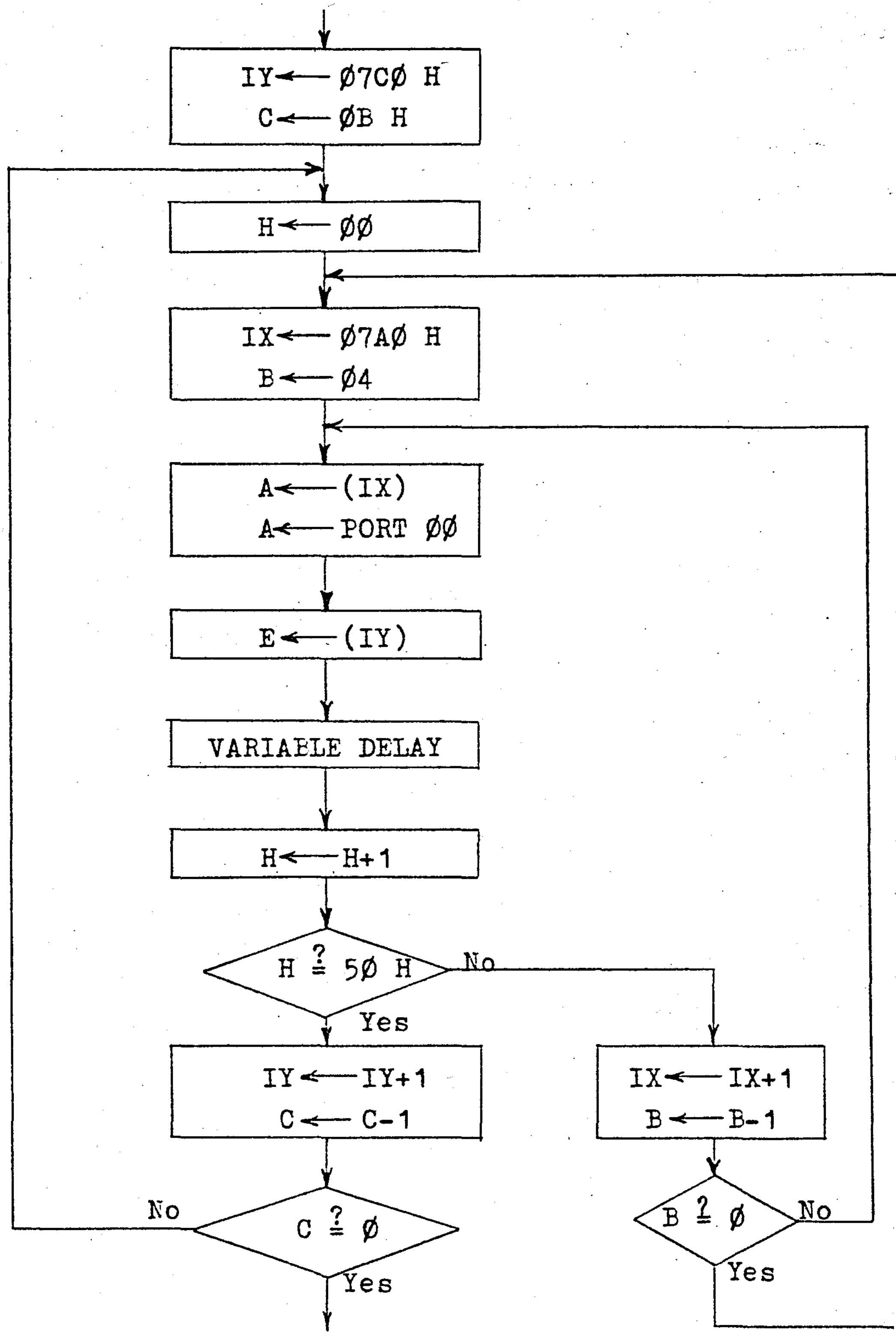


Drill Program

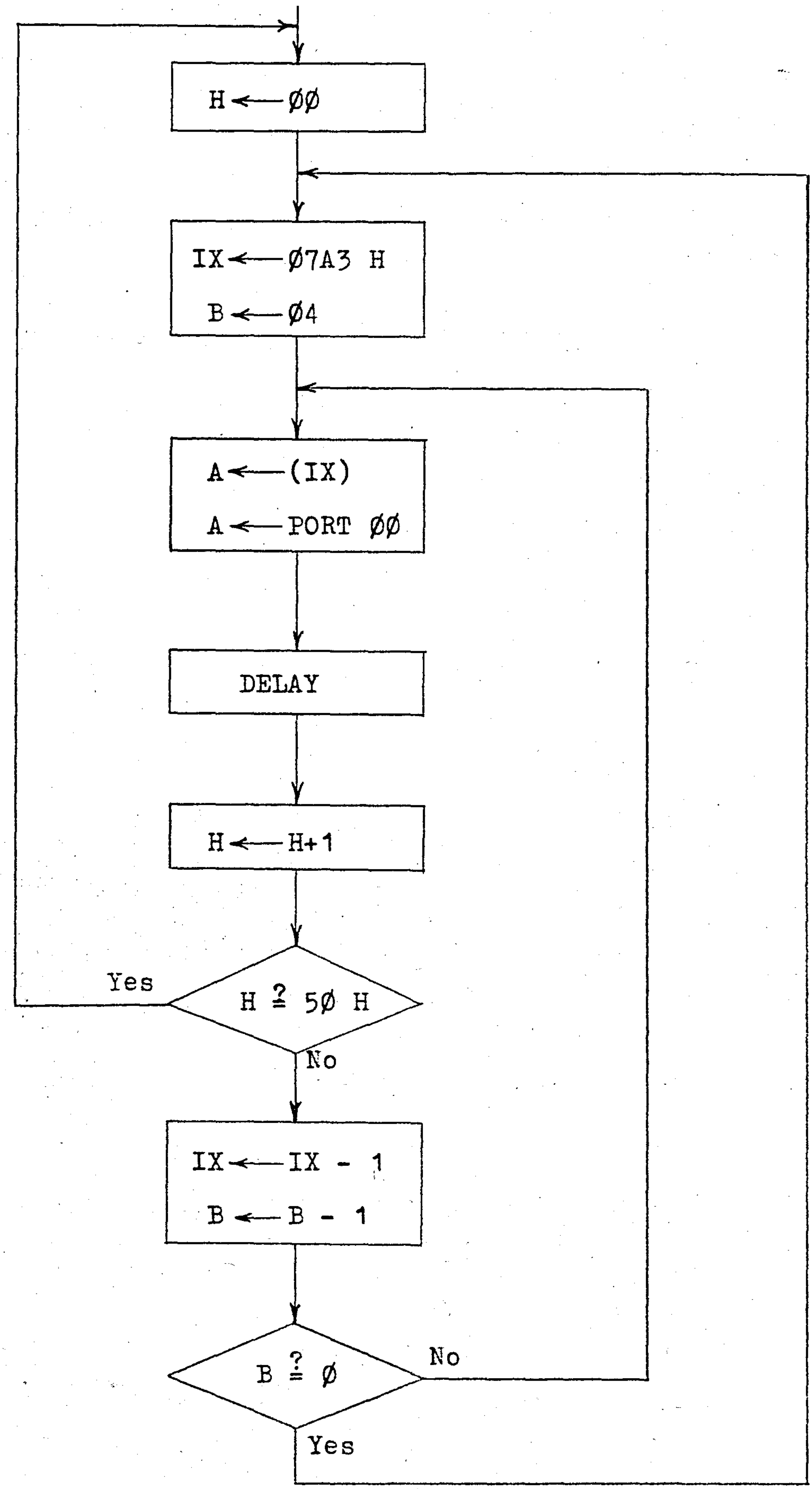
(For even lines)

The same flowchart (which is written for odd lines) can be used, with certain changes, for the even lines also. These changes can be understood by looking at the related sections of the drill program listing.

Acceleration Routine



Constant Speed Routine



```
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 const
XSTEP4 : EQU Ø7A3H   ;Location of last X-motor consta
YSTEP1 : EQU Ø7BØH   ;Location of first Y-motor const
YSTEP4 : EQU Ø7B3H   ;Location of last Y-motor consta
XINC   : EQU 87F4H   ;Location of incremented x-lengt
XL2    : EQU 87E8H   ;  "  of twice   "   x-length
LOC1   : EQU 87F5H   ;Location of byte count
LOC2   : EQU 87F6H   ;Location of incomplete store by
STAM   : EQU 87FAH   ;Starting address location of me
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
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 mo
                SCF               ;steps to determine the de
                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 scann
            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,00
SXDPS       :   LD IX,XSTEP1
                LD B,COUNT
SXDPM       :   LD A,(IX+00)
                OUT (PORT0),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 Progr
                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-li
                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+Ø),A
                INC IX
                JP FTR6
        FLD7  : LD A,(XL2)
                SUB C
                LD (IX+Ø),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+Ø)      ;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+Ø)
                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+∅)       ;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+Ø)      ;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),ØØ
          DJNZ SZERO
          LD A,(XINC)
          LD C,A
          LD HL,(SAVEHL)
          LD IX,FRSTIX
STR     : BIT Ø,HL          ;Routine to trace an odd-line
          JP NZ,SØ
RØ      : 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+Ø),C
                 INC IX
                 JP R6
S7       :       LD (IX+Ø),C
                 INC IX
                 JP R7
SECLIN   :       DEC IX
                 LD (LASTIX),IX
                 INC HL
                 LD (SAVEHL),HL
                 LD IX,FRSTIX
                 LD A,(IX+Ø)
                 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)
```

```
                INC IX
                LD A,(IX+1)
                OR A
                JP Z,FTRAC
                JP SFRTL
SLEFT  : LD IX,(LASTIX)   ;Routine to move left on
                LD A,(IX+Ø)       ;an odd-line
                LD E,A
                LD A,(OLDX)
                SUB E
                JP Z,SLZERO
                JP NC,SLMOVE
RLM    : LD A,(OLDX)
                LD E,A
                LD A,(IX+Ø)
                SUB E
                LD (WAY),A
                LD H,A+DSTP
                SUB H
                CALL Z,MVINXR
                JP Z,SLZERO
                CALL C,MVINXR
                JP C,SLZERO
                LD (PATH),A
                CALL XRACC
                JP SLZERO
SLMOVE : LD (WAY),A
                LD H,A+DSTP
                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 afte
                LD E,A          ;drilling of the whole card i
                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 i
         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,ØØ
LLINE   :  LD IX,XSTEP1
                LD B,COUNT
LNEXT   :  LD A,(IX+ØØ)
                OUT (PORTØ),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 detect
                EX AF,AF'   ;up in y-direction by accel
                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 detect
         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 detec
         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 detec
         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
XNMAX  : LD IX,XSTEP1     ;the stage moves with tha
         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,ØØ
SXAPS     :     LD IX,XSTEP4
                LD B, COUNT
SXAPM     :     LD A,(IX+Ø)
                OUT (PORTØ),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,ØØ
SXPDR     :     LD IX,XSTEP4
                LD B, COUNT
SXPDM     :     LD A,(IX+Ø)
                OUT (PORTØ),A
                LD E,(IY+Ø)
                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
```

BIBLIOGRAPHY

1. Leventhal, Lance A. Introduction to Microprocessors Prentice/Hall, Inc., 1978.
2. Mano, M.Morris. Computer Logic Design , Prentice/Hall Inc. 1978.
3. H.Taub/D.Schilling. Digital Integrated Electronics McGraw-Hill, Inc., 1977.
4. Millman/Halkias. Electronic Fundamentals and Applications McGraw-Hill, Inc.,1976.
5. George M.Chute/Robert D.Chute. Electronics In Industry McGraw-Hill, Inc.,1979.
6. Edminister, Joseph A. Electric Circuits, McGraw-Hill, Inc. 1972.
7. Earle, James H. Design Drafting , Addison-Wesley press, 1972.
8. Telefunken Optoelectronic Devices data book, 1976.
9. Philips Electric Motors and Accessories data handbook, 1977.
10. Gilder, Jules H. "Focus on Stepping Motors" , Electronic Design , Vol.25,No.22, pp. 48-57, 1977.
11. Superior Electric handbook "Design Engineer's guide to DC stepping motors", 1976.
12. Airpax Stepper Motor Handbook, 1979.
13. National Linear Data Book, 1978.
14. Motorola Transistor Data Book, 1975.
15. Motorola TTL Data Manual, 1981.
16. Intel Component Data Catalog, 1982.
17. Zilog Z-80 technical manual, 1977.
18. Motorola Memory Data Manual, 1981.
19. Intel Peripheral Design Handbook, 1979.