8085-BASED UNIVERSAL EPROM PROGRAMMER

BY

KHANDAKER ATWAR HOSSAIN

A PROJECT REPORT

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING, BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE

DEGREE OF

MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

JANUARY, 1985.

CERTIFICATE

This is to certify that this project work was done by me and it has not been submitted elsewhere for the award of any degree or diploma.

Countersigned

(Dr. Syed Mahbubur Rahman)
Supervisor.

Signature of candidate

Kh. Atwar Hossain.
(Khandaker Atwar Hossain)

Accepted as satisfactory for the partial fulfilment of the requirements for the degree of Master of Engineering in Electrical and Electronic Engineering.

EXAMINERS

1. ....
(Dr. Syed Mahbubur Rahman) — Supervisor
Assistant Professor,
Computer Engineering Department
Bangladesh University of Engineering
and Technology, Dhaka, Bangladesh

2. ....
(Dr. Shamsuddin Ahmed) — Member
Dean
Faculty of Electrical and
Electronic Engineering,
Bangladesh University of Engineering
and Technology, Dhaka, Bangladesh

3. Rahman 14/1/85
(Dr. Syed Fazl-i Rahman) — Member
Professor and Head,
Department of Electrical and
Electronic Engineering,
Bangladesh University of Engineering
and Technology, Dhaka, Bangladesh

4. ....
(Dr. Mahfuzur Rahman) — Member
Professor and Head,
Computer Engineering Department
Bangladesh University of Engineering
and Technology, Dhaka, Bangladesh.

DHAKA
JANUARY, 1985

ALL PRAISES TO ALLAH, THE MERCIFUL

ACKNOWLEDGEMENT

The author expresses his indebtness and deep sense of gratitude to Dr. Syed Mahbubur Rahman, Assistant Professor of Computer Engineering Department, Bangladesh University of Engineering and Technology, Dhaka for his continuous guidance encouragement and help and continous suggestions all along the course of this work.

The author wishes to express his great indebtness to Dr. Shamsuddin Ahmed, Dean of the Faculty of Electrical and Electronic Engineering for his interest in progressing the project work.

The author also expresses his greatfulness and thanks to Dr. S.F. Rahman, Head of the Department of Electrical and Electronic Engineering and to Dr. Mahfuzur Rahman, Head of the Department of Computer Engineering, who were kind enough to provide all the facilities of the Departments for this work.

The author expresses his indeptness to Mr. Abdul Hannan, Principal, Khulna Engineering College, for his keen interest and encouragement in pursuing the post graduate studies.

ABSTRACT

Microprocessor based designs require to store their operating and additional programs in a permanent memory such as an EPROM. EPROM programmers are thus an essential requirement for any microprocessor based design. The present project work deals with the development of a 8085-based EPROM programmer, which can be used in Program, Verify, Read, Display, Copy and Print Mode. Programs have been developed for the Intel 2716, 2732, 2764 and 27128 EPROMs. Since the programmer is software controlled, any type of EPROM can be programmed just with a little change in the control program of the EPROM programmer. The project work, besides, greatly encouraging microcomputer based designs, will also save buying of costly manufactured programmers available in foreign market.

CONTENTS

CONTENTS (Contd.)

CHAPTER - I

INTRODUCTION

1.1 General Introduction

The advent of discrete semiconductors and the technological follow-on, e.g., the monolithic integrated circuits, have had significant impact on the performance, cost, reliability, maintainability, physical size, and architecture of central processing units of digital computer and allvarieties of digital systems.

The use of semiconductors has now started to spread to peripheral equipment and memory. When the memory elements are organized and combined in a particular arrangement, they are referred to as memory system.

1.2 Non-Volatile Memories

A memory section for storing permanent data, e.g., list of constants, tables of data conversion like trigonometric sine tables etc., and fixed computer programs like operating instructions, editor, monitor or utility programs, subroutins, keyboard encoder, character generator etc., do not require a write capability once it is loaded; thus time and cost may be saved by omitting the write feature. Such a storage device is called a Read-only Memory or ROM. The information stored in these devices will remain unchanged even if power is removed from the system which offers a considerable convenience. Moreover for data only occassionally changed, there are even cases between full write capability and the ROM. One such type is erasable programmable read-only memory or EPROM.

EPROM has quickly found its popularity as a non-volatile temporary storage medium. More likely responsible is its attractive combination of field programmability, high density, and low power consumption when compared to bipolar fusible-link memories. Such devices, built with either p- or n- channel metal-oxide-semiconductor technology, have served well in prototyping microprocessor based system . There is an extra advantage for the user to see the actual silicon chip through the quartz window that lets the ultraviolet light for erasure.

1.3 Some Available EPROM Programmers

To program the EPROMs some devices are required known as EPROM programmer. Various types of EPROM programmer are available in the market. Performance of these programmers varies with the cost. These programmers are sometimes ON-line connected with the computer and sometimes independent units. Some available programmers are listed below along with their approximate prices.

i. "Universal PROM programmer" by Intel corporation. It can program all kinds of EPROMs using different personality cards and programming adapter socket. It can be used as an independant unit and also micro-computer based with RS-232 cable. Price is around £ 1000.

ii."PROM Programmer M980[7]" by Pro-Log Corporation. They state "Built to handle the Programmable devices of today .... and tomorrow". Cost is above £ 1000.

iii. "EPROM Emulator and Programmer EP4000[8]" produced by G.P. Industrial Electronics Ltd. It is a microprocessor controlled programmer which can program all the popular EPROMs of Intel Corporation. Cost is about £ 600.

iv. Rockwell International introduced an EPROM programmer with their microcomputer AIM-65[9]. It can program only the Intel 2716 EPROM. Duplication is not possible here.

v. H. Muller (Switzerland) designed an EPROM Programmer for the popular 2716 & 2732 EPROMs using timer, logic gate and other accessories. There is no scope of duplication. Material cost is around £ 25[10].

vi. H.S.Lynes (England) designed an EPROM Programmer to accommodate 2708, 2716 & 2732 EPROMs using drivers, latches etc. It is to be interfaced with microcomputer. Material cost is around £ 20[11].

1.4 Aim of the Project:

The EPROM Programmer to be designed is microcomputer controlled. It can be used for programming Intel 2716, 2732, and 2764 and 27128 EPROM and the EPROMs within the 8048H & 8087H microcomputer chips. An additional advantage over the available EPROM programmers is that with little modification it can be used to program any type of EPROM available in the market. The 40 output ports should be connected to proper pins of the EPROM and a software with a little change should be written for the particular EPROM to be programmed.

EPROM is an essential part in all microcomputer based designs. As such the design of such a flexible EPROM programmer will greatly help the research works on the microcomputer designs. It will also replace the need for investment of quite a large amount of foreign exchange required for this purpose. With industrial backup commercial manufacturing may also be taken up.

CHAPTER - II

SEMICONDUCTOR MEMORY

2.1 Read-Only Memory

Semiconductor memories fall into two basic categories. Read-only memories (ROMs) and Randam-access read/write memories (RAMs). ROM forms an important part of memory requirement in most microcomputer systems. This type of memory are also called fixed memory, permanent memory or read-only store (ROS).

In principle it is simply a special combination circuit because an input signal combination (address and entry) defines a unique output combination. The ROM can take a digital code at its input terminals and provide a unique digital code on its output terminals. The relationship between its input and output codes are relatively fixed, usually alterable only by slow techniques, and for this reason it is termed "read-only". The difference between read-only memory and read/write memory is the level of difficulty in changing the stored information.

The ROM, being a fixed memory. is non-volatile: i.e., loss of power or system malfunction does not change the contents of memory. The ROMs have the feature of random access, which means that the access time for a given memory location is the same as that for all other locations.

As a result of the recent advances in IC technology, ROMs exist in many forms. The technique employed for storing information in the ROM (called programming) provides a convenient method for classifying all ROMs into one of the following three categories[4].

1. Programmed during manufacture
2. One-time programmable after manufacture by the user
3. Programmable after manufacture with provision for erasing and reprogramming.

Provision for programming any of these three types commonly involves some sort of link, which may be opened or closed, between each row-select line and an input line of each OR gate. These units are then specified according to the information pattern that is to be programmed into the unit in a manner that depends upon the units type.

2.1.1 Manufacture-Programmed Read-Only Memory (ROM): Programming Characteristics.

ROMs are programmed usually as one of the final steps of their manufacture. The links are simply gaps that may be bridged by a final metalization pattern that is placed onto the circut according to the desired information to be programmed. This is done with a mask that determines the precise pattern, which is custom-made for a particular application. The mask is expensive to make but may be used to program any number of units. So this type of ROM is best suited for mass production.

An example of this first type of ROM unit using pn junction devices is shown in Fig.2.1. A diode is provided for each bit location in the unit. As can be seen a gap occurs in each path between a diode and the common line for a column. A value of "O"

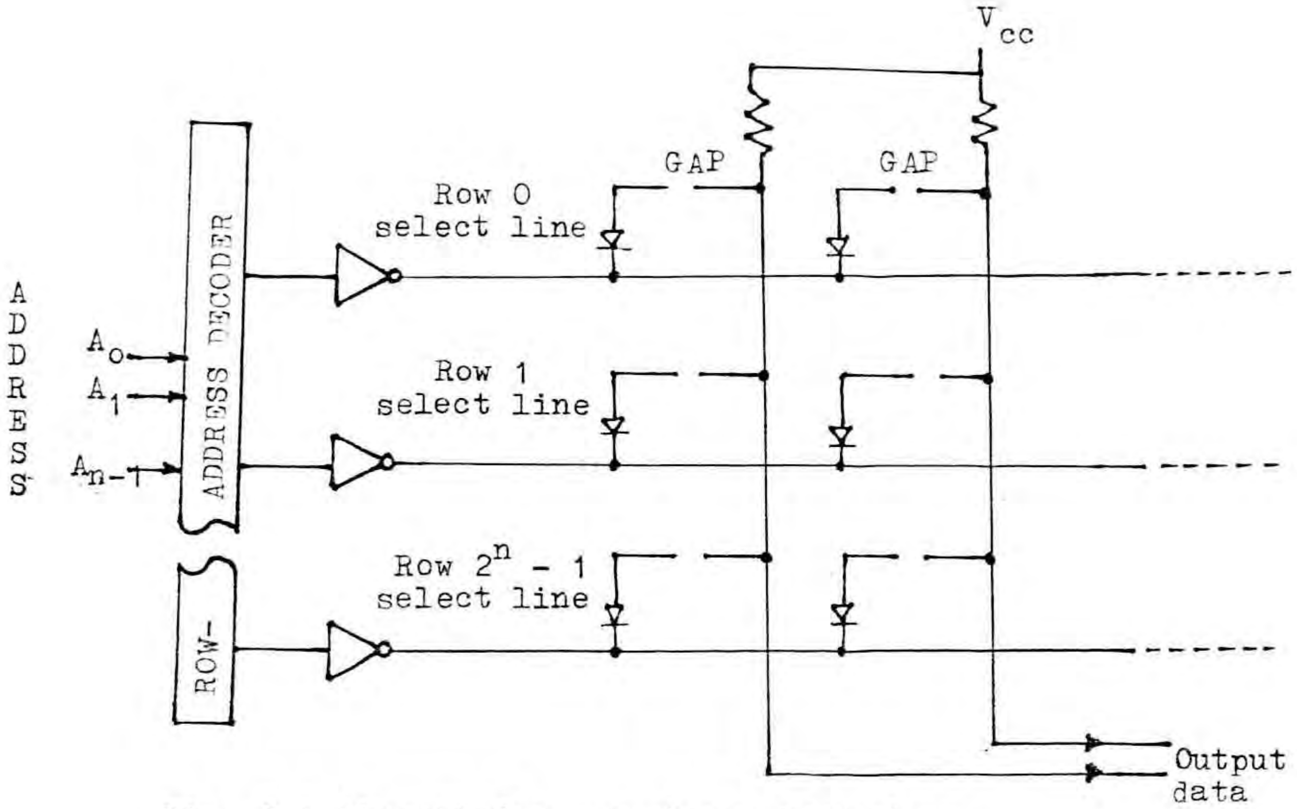


Fig. 2.1 Circuit for a mask-programmed pn junction read-only memory unit.

is programmed into a particular bit position by bridging the corresponding gap, otherwise, the value of that bit is 1.

From the figure 2.1 it can be noticed that the common line of a column is normally pulled high by a resistor and that only the select line corresponding to the specified address can pull it low. If the gap to the diode connected to that select line is left open, then the common line remains high, corresponding to an output of "1". If the gap is closed on the otherhand, then the common line is pulled low by the select line, corresponding to an output of "0".

2.1.2 Programmable Read-Only Memory (PROM): Programming Characteristics

This type of ROM can be programmed one time by the user. It has generally a link that, after manufacture, can be altered one way, either from closed to open or vice verse. The most common way of doing this is to make the link from a fusible metal, such as NiCr, which can be selectively "blown" (i.e., open circuited) by supplying an external current of sufficient magnitude. An example of such a ROM unit is shown in Figure 2.2. The unit is similar to the previous example of a pn junction ROM unit, except that NiCr fuses are placed across the gaps and that there is provision for each of these fuses to be blown.

There are two modes of operation for this type of ROM:

i. Read mode

ii. Program mode

The mode at any time is determined by the value of the power-supply voltage V_{cc}. For the read mode, V_{cc} is placed at its normal value (e.g., 5 V); for the program mode, V_{cc} is raised to higher voltage (e.g., 10 V). A threshold circuit is connected to V_{cc} to generate the signal labled "Read Mode" which is logic-1 when the ROM unit is in the read mode of operation. This signal, Read Mode, is used in conjunction with an external signal, Chip Enable, to affect both the enabling of the row-select lines and the enabling of the output data lines.

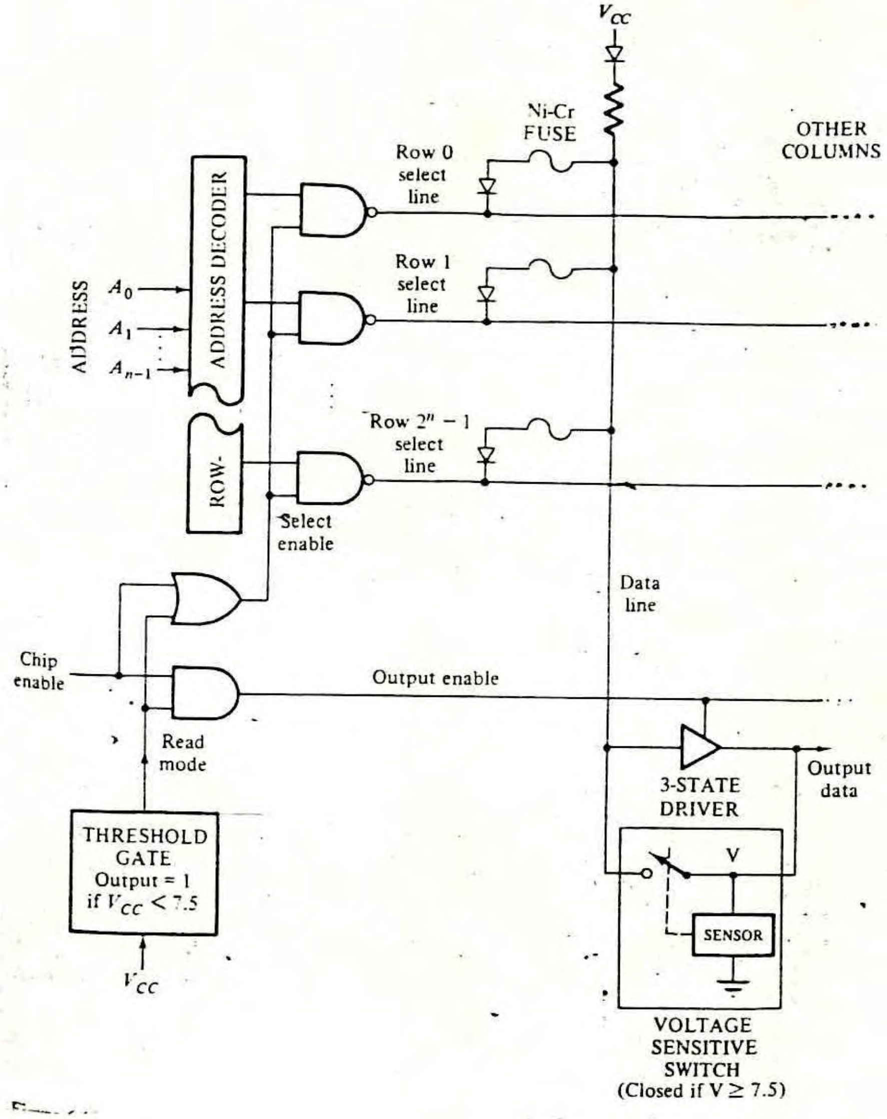


Fig. 2.2 Circuit for a fusble-link programmable read-only memory unit.

Read mode:- In this mode the row-select lines behave presisely the same way as in the preceding example of a pn junction ROM. As can be seen in Fig. 2.2, NAND gates are inserted into the row-select lines. The second input line of each of these NAND gates, which is driven by the OR of Chip Enable and the signal Read Mode, will be at logic-1, since Read Mode is at logic-1. In this case the NAND gates simply invert the decoder outputs. The row-select line corresponding to the specified address is at logic-0, and all other row-select lines are at logic-1.

If the fuse is intact for a column and the selected row. then the data line for that column is forced low (logic-0); otherwise, the data line is pulled high (logic-1) by the pull-up resistor. A 3-state driver is used to couple each column data line to an overall output line labelled "Output Data". These 3-state drivers are controlled by a signal labeled "Output Enable", which is the AND of Chip Enable and the signal Read Mode.

Program mode:- In this mode of operation, V_{cc} is raised to a higher voltage (e.g., 10V). This affects the circuit in several ways. It causes the signal read mode to go to logic-0, which has two effects. First, the signal Select Enable is made to depend upon the external signal Chip Enable, thereby making the row-select line corresponding to the specified address to be logic-0 only if Chip Enable is logic-1. Second, it causes the signal Output Enable to be logic-0, which disables the 3-state output drivers independent, of the value of Chip Enable. In addition and most important, the row-select lines that do not correspond to the specified address are raised to higher voltage (e.g.10V), since the gates that drive them use the power-supply voltage V_{cc} to determine the logic-1 voltage level.

To cause a fuse to be blown, the desired address is specified, the Chip Enable line is then activated to enable the row-select lines, and then a specified current (approximately 50mA) is supplied from a current source to the appropriate Output Data line. This current passes through a voltage-sensitive switch to the data line of the corresponding column. The switch is a special circuit that closes when the voltage V applied by the current source exceeds a value somewhat higher than the maximum logic-1 level of 5V. The applied voltage will be sufficiently high if the specified current is made to flow. Once past the switch, the applied current flows through the fuse and diode that are connected to the row-select line that is low (logic-0). All other row-select lines are at approximately 10V and therefore draw an insignificant amount of applied current. After a short time the current causes the fuse to melt away. When a sufficient time has passed to allow for this, the Chip Enable line is deactivated. The process may then be repeated for other fuses that are to be blown in the same row and then for other rows.

2.1.3 Erasable Programmable Read-Only Memory (EPROM): Programming Characteristics.

The third type of read-only memory unit, which can be programmed, erased, and reprogrammed, is refferred to as "erasable programmable read-only memory" or in short EPROM. This type of unit generally makes use of a link that can be placed on one condition (say closed) on a selective basis and into the other condition (say open) on a collective basis. Programming such a unit consists of first placing all links collectively into a specific condition, which amounts to erasing any previous information content, and then placing desired links into the opposite state, one at a time.

A notable example of a link that is used in EPROM units consists of a special type of MOS transistor having what is called a floating gate. A specific form of such a transistor is that of a p-channel normally-off (enhancement-mode) MOS transistor with its gate electrode surrounded by insulation, as shown is Fig.2.3. The gate has no lead attached to it but may, however, be made to acquire a negative charge, as will be seen. When charged negatively, the gate induces a positive charge into the channel, just as it would if a negative potential with respect to the substrate were applied to the gate. With a positive charge in the channel the transistor is conductive. On the otherhand, if the gate has no negative charge, then no positive charge is induced into the channel and the transistor is not conductive.

Negative charge is supplied to the gate by injecting electrons from the drain through the insulation to the gate. This is done by applying a relatively high positive voltage to the source with respect to the drain.

The resulting electric field induces a relatively high positive voltage on the gate with respect to the drain.

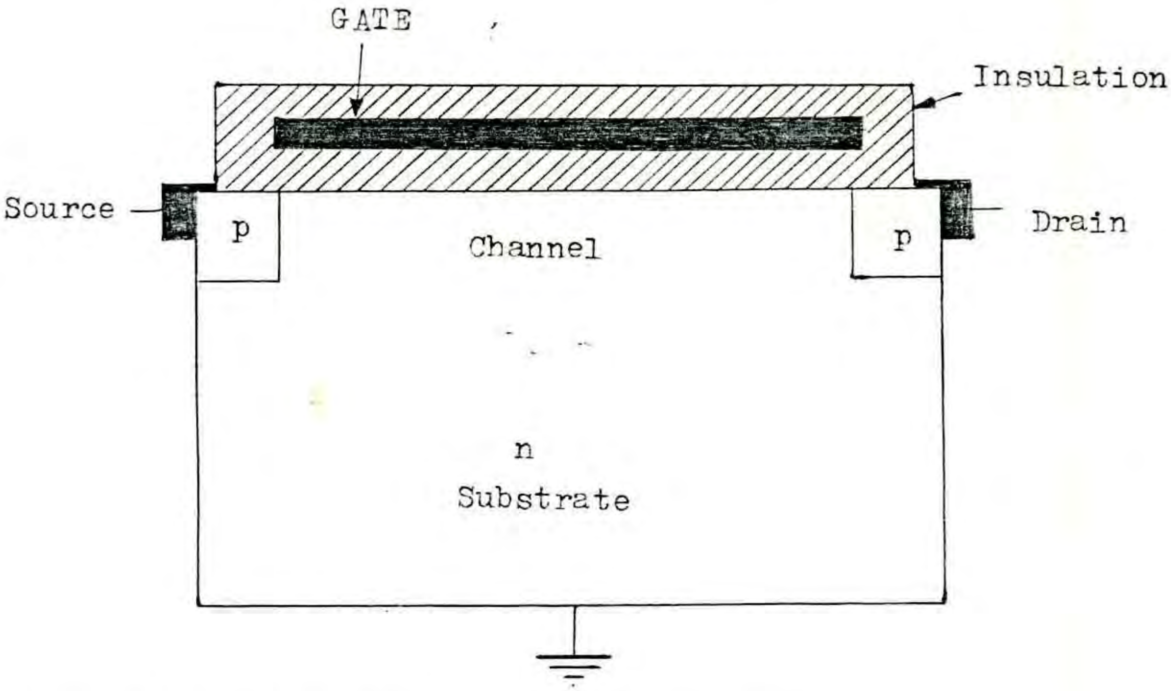


Fig. 2.3 Floating-gate PMOS enhancement-mode transistor for EPROM units.

Roughly speaking, this voltage causes a breakdown to occur at the junction between the drain and the insulation, resulting in a flow of rather energetic electrons from the drain into the insulation. Owing to their energetic state, the electrons are able to drift through the insulation to the gate electrode. This electron charge on the gate will remain there almost indefinitely unless specifically removed. Charge is removed from the gate by irradiating the area with ultraviolet light, which imparts sufficient energy to each electron to bring it into a conductive energy band in the insulation. The electrons are then able to flow away from the gate owing to their mutual repulsion.

A read-only memory unit using these special transistors can be structured in a manner similar to the previous example of a MOSROM, which was shown in Fig.2.1. In this case, each gap in the unit is bridged with one of these floating-gate transistors. In addition, circuitry is included that allows each floating-gate transistor to be selected for application of the necessary voltage to charge the gate. This selection process involves the address and output-data lines.

To program the resulting EPROM, all links are first opened (corresponding to logic-1) by irradiating the unit with ultraviolet light of sufficient intensity and duration. Then, for those bit positions that are to contain logic-0, the corresponding links are closed one at a time.

2.2 Erasure Characteristics of the EPROMs

The erasure characteristics of the Intel 27-series EPROMs and the EPROMs within the 8748 & 8048 micro-computer chips are such that erasure begins to occur when exposed to light with wave lengths shorter than approximately 4000 Angstrom (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000 Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical EPROMs approximately in 3 years while it would take approximately one week to cause erasure when exposed to direct sunlight. If the EPROMs are to be exposed to these types of lighting conditions for extented periods of time, opaque labels are available which should be placed over the chip window to prevent unintentional erasure.

The recommended erasure procedure for the EPROMs is exposure to shortwave ultraviolet lightwhich has a wavelength of 2537Å. The integrated dose (i.e., UV intensity X exposure time) for erasure should be a minimum of 15 w-sec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000 μV/cm^2 power rating. The chip should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure[5].

CHAPTER - III

MICROCOMPUTER

3.1 Basic Microcomputer Organization

A microcomputer is a bus-oriented system of subassemblies that are implemented utilizing the technology of LSI. Functions of these subassemblies provide for manipulation of information, ordering the sequence of instruction execution, interpretation of instructions, control and timing of bus operation, storage of instructions and data, and communication between the computer and the external environment. The first four of these functions are frequently handled by a single subassembly known as a microprocessor. The storage function is handled by a memory subassembly. This subassembly may consists of ROM and/or RAM. The external communication is performed by a set of subassemblies known as input and output ports. Each port provides an interface between the microprocessor and some external device.

Figure 3.1 shows the general layout of a microcomputer and Fig.3.2 shows the basic blocks of an 8-bit microcomputer system. The various buses which connect these blocks are also shown. There are three buses:-

1. The address bus
2. The data bus
3. The control bus

These buses connect the microprocessor (CPU) to each of the ROM, RAM and I/O elements so that information transfer between the microprocessor and any of the other elements can take place.

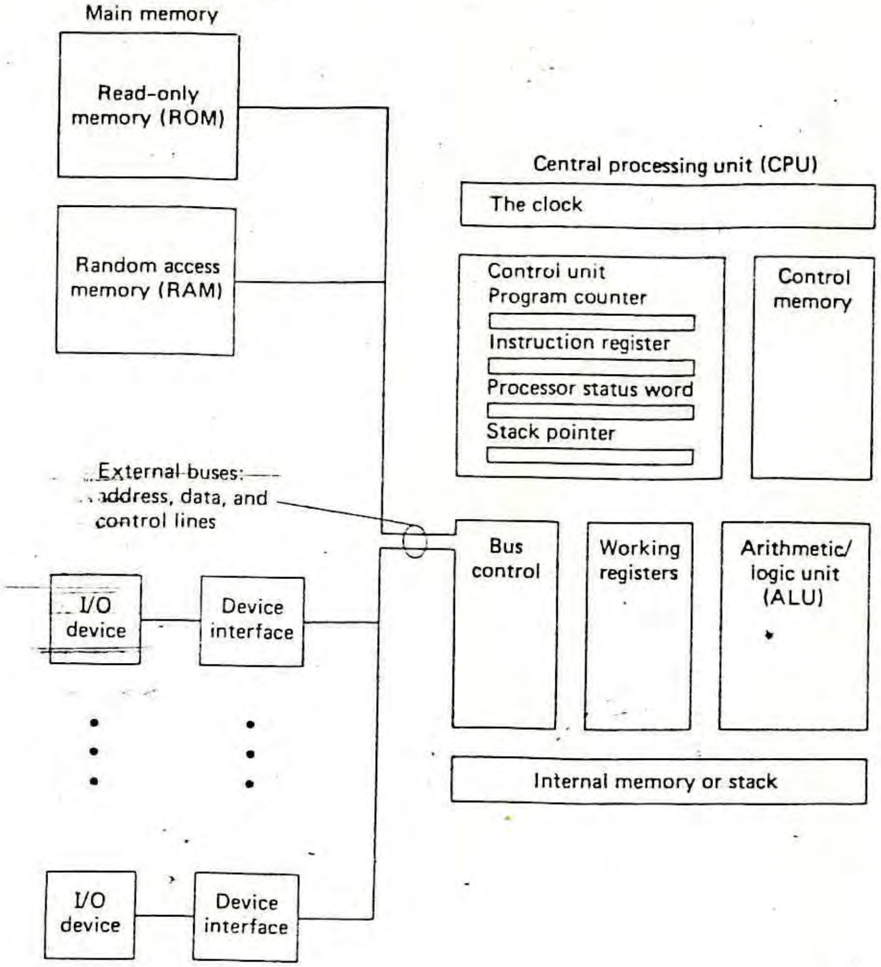


Fig. 3.1 General layout of a microcomputer.

Note: The I/O device may be the Keyboard (Input), video display (Output), EPROM Programmer (Output).

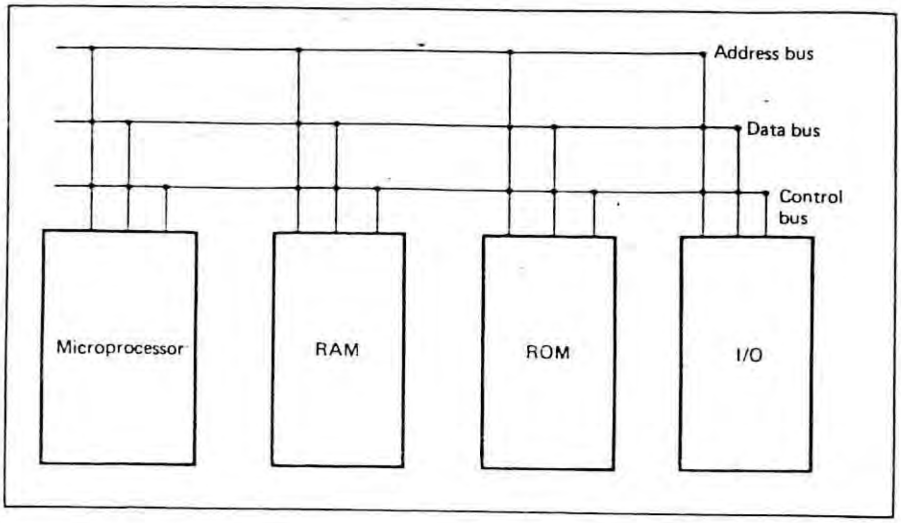


Fig.3.2 Block Diagram of a Typical Microcomputer

Address bus is used for transferring information from microprocessor to memory or I/O elements in one direction only. When the microprocessor wants to transfer information between itself and a certain memory location or I/O device, it generates the 16-bit address from an internal register on its 16 address pins A_0 - A_{15}, which then appears on the address bus. These 16 address bits are then decoded to determine the desired memory location or I/O device. The decoding process normally requires hardware (decoder).

The data bus is a bidirectional bus. In 8085 microprocessor, the 8 data pins are used to sand lower 8 address bits in addition to data. That is the data pins are time-shared or multiplexed. When address bits are sent the address latch enable (ALE) pin becomes HIGH.

The control bus consists of a number of signals that are used to synchronize the operation of the individual microcomputer element. The microprocessor sands some of these signals to the other elements to indicate the type of operation being performed.

3.2 The Intel 8085 Microprocessor

The Microprocessor is the CPU of the microcomputer and is the combination of control unit and arithmatic and logic units. Therefore, the power of the microcomputer is determined by the capabilities of the microprocessor and its clock frequency determines the speed of the microcomputers. Microprocessor consists of logical components that enable it to function as a programmable logic processor. Some of the components, i.e., the program counter, stack and instruction register, provide for the management of a program. Other components, i.e., the ALU, Carry flip-flop, scratchpad register and data-address register provide for the manipulation of data. The remaining components, i.e., the decoder, timing and control unit specify and coordinate the operation of the other components. Internal pathways interconnect the components to provide for transferring data between designeated components. Connection of the microprocessor to other units (memory and I/O devices) is done with the address, data, and control buses.

3.2.1 The 8085 Architecture and its Pin Function

Intel 8085 8-bit microprocessor is housed in a 40-pin dual-in-line (DIP) package. Fig. 3.3 shows the Intel 8085 pin diagram and Fig.3.4 shows the functional block diagram of the Intel 8085 microprocessor.

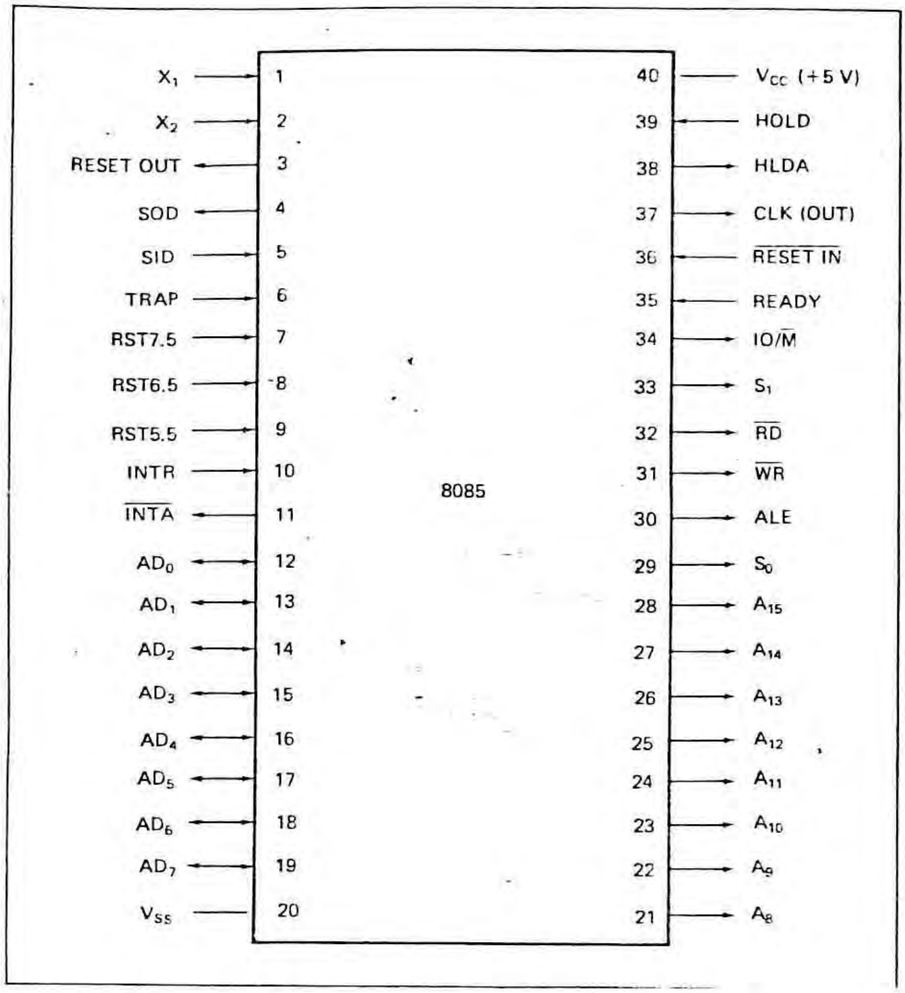
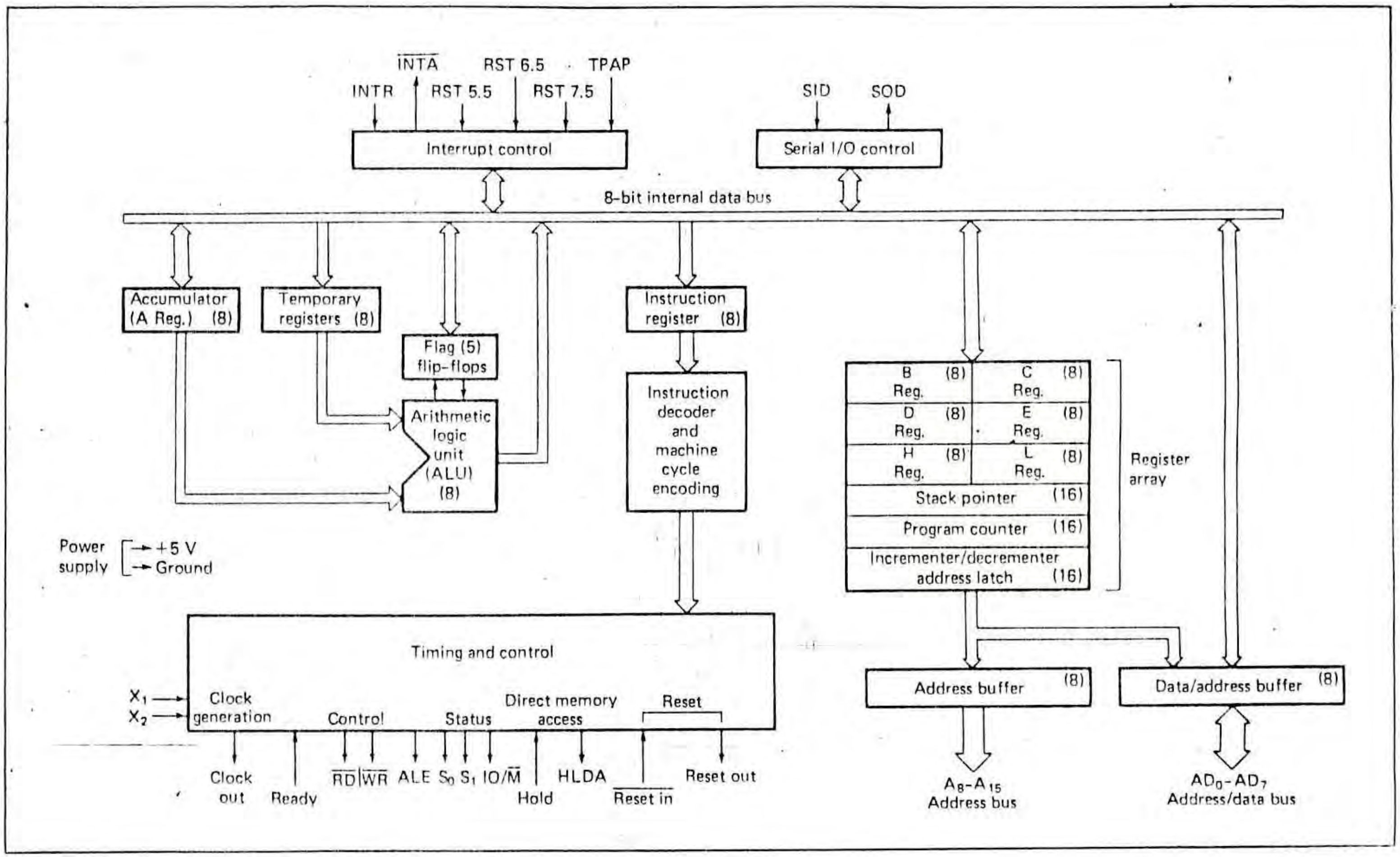


Fig. 3.3 Microprocessor signals and Pin Assignments

The internal organization or architecture of the Intel 8085 microprocessor is briefly discussed below.

It has a 16-bit program counter and address latch which

Figure 3.4 8085 Microprocessor Functional Block Diagram (Courtesy of Intel Corporation)



feed the dedicated address bus (A_{15} - A_8) and the dual purpose address/data bus (AD_7 - AD_0). Parallel data enters and leaves the MPU via the multiplexed address/data bus (AD_7 - AD_0). The address/data bus transmits an address when the ALE control line is HIGH and data when the ALE line is LOW.

The 8-bit internal data bus carries input or output data throughout the unit. The data can flow from the internal data bus to the 8-bit accumulator or temporary register, flags, instruction register, interrupt control unit, serial I/O control unit, any of the general purpose registers (B,C,D,E,H and L), 16-bit stack pointer, 16-bit program counter, or 8-bit data/address buffer. The arithmetic-logic unit (ALU) is being fed by two 8-bit registers (accumulator and temporary register). The flag flip-flops have five status indicator as shown in Fig. 3.5. below.

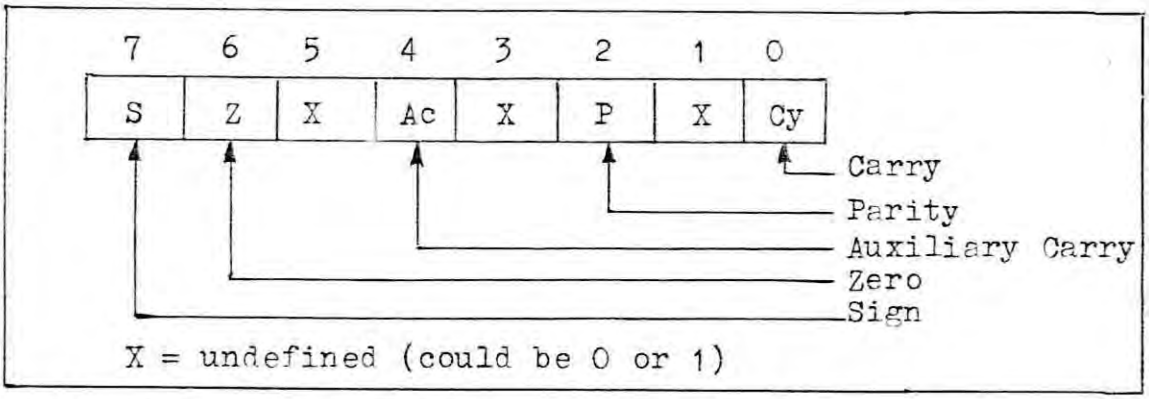


Fig. 3.5 Format of Processor Status Word

The instruction register feeds the instruction decoder. This instruction decoder interprets the current instruction and determines the microprogram to be followed or the machine cycle encoding. The instruction decoder then instructs the timing and control section as to the sequence of events to be followed. The timing and control section coordinates actions of both processor and the peripheral.

The RD pin signal is outputted LOW during a memory or I/O READ operation. Similarly, the WR pin signal is outputted LOW during a memory or I/O WRITE. The $IO/\overline{M}$ signal is outputted high to indicate an I/O operation and is outputted low during a memory operation. The $IO/\overline{M}$, S_0, and S_1 are outputted during its internal operations as shown in Fig. 4.2. (Chapter-IV).

In the present project we have a microcomputer with the 16-bit address bus, 8-bit data bus, and with the necessary control signals ($\overline{IORD}$, $\overline{IOWR}$). The designed EPROM programmer is interfaced to these buses. The interfacing designs are described in the subsequent Chapters.

CHAPTER - IV

INTERFACING WITH PERIPHERAL DEVICE

In regard to microcomputer systems, interfacing can be separated into two areas of concern. One area involves the connection of the components, such as memory units and input/output registers, to the buses of a microprocessor. Such interfacing is primarily concerned with the timing and control of the buses and selection of a component so as to effect a data transfer at a given time between the selected component and the microprocessor.

The other area of concern involves interfacing components external to the microcomputer, such as peripheral devices, data channels etc. Such interfacing does not directly involve the buses of the microprocessor; so it is less structured. It is concerned with converting signals associated with the external components, which might be of any nature (including analog), to signals compatible with the buses and vice versa.

An input or output operation is the act of transferring data to or from a selected peripheral device. The microprocessor is the focus of all operation, so an input will mean data flows into the MPU whereas an output will mean data flows out of the MPU. Those locations where data is input from or output to are usually called input or output ports.

From the 8085 instruction sets, it appears that it uses the IN and OUT instructions for transferring data to and from I/O ports. These data transfer instructions are illustrated in Fig. 4.1. The output instruction is represented by OUT mnemonic in assembly language program, while the input instruction uses the IN mnemonic. The instruction formats for these operations are also reproduced in the Fig. 4.1 showing the opcode followed by a device number or port address. The byte long port address can select one of 256 (2^8) ports. Mostly the most significant 8 address lines (A_8 - A_{15}) are used for the port address.

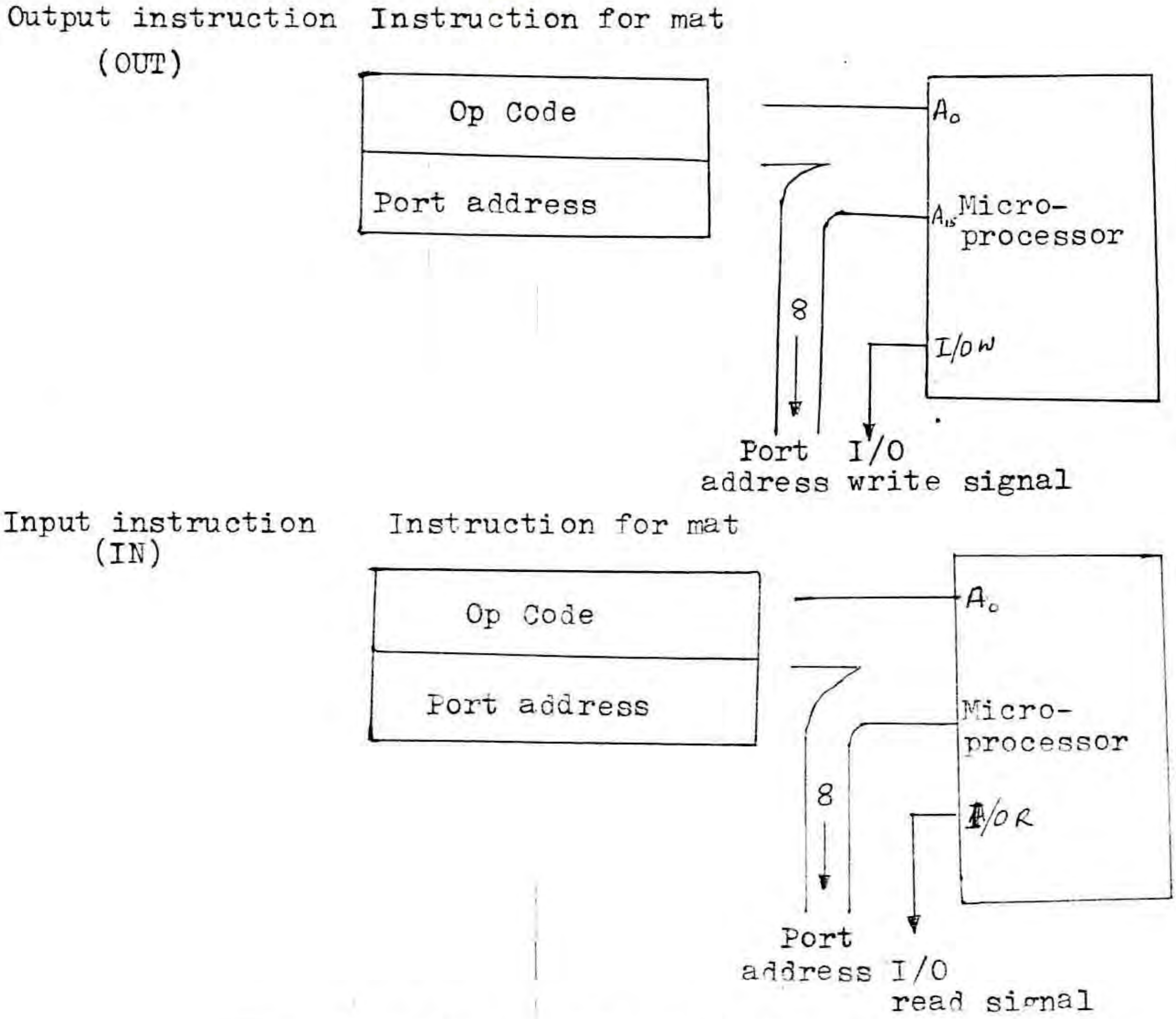


Fig. 4.1 I/O operation of 8085

Fig.4.1 also shows two additional output control signals added to the microprocessor. When using the OUT operation, a special input/output write ($\overline{I/OW}$) signal is used. The IN operation also requires the use of a special output signal called input/output read ($\overline{I/OR}$) signal. Both of these output signals are active LOW signals and are illustrated in Fig.4.1. The status of the S_0, S_1 and IO/$\overline{M}$ pins during various read/write operations are shown in Fig. 4.2.

8085 control signals			Function
IO/M	S_1	S_0	
0	0	1	Memory write
0	1	0	Memory read
1	0	1	I/O write
1	1	0	I/O read
0	1	1	Op code fetch
1	1	1	Interrupt acknowledge

Fig. 4.2 Control Signal table for I/O.

The data transfer using the IN and OUT instructions are classed as program-controlled I/O. Program instructions are controlling the transfer of data during IN and OUT operations. Program controlled I/O is divided into two techniques:

i. Standared I/O

ii. Memory Mapped I/O

The memory-mapped I/O technique is the most common and can be used with any microprocessor. The standard I/O techniques can be used only with microprocessors that have separate IN and OUT instructions as in the 8085 microprocessor.

It is common to refer to an output as "an output to a peripheral device". In actual practice however, the output from the microprocessor is not directly to a peripheral device but to a memory device which stores the data for the peripheral unit. The intermediate blocks in Fig. 4.3 are the memory devices known as "input interface adapter" or "output interface adapter". It is common for I/O interface adaptor to have characteristics other than memory also.

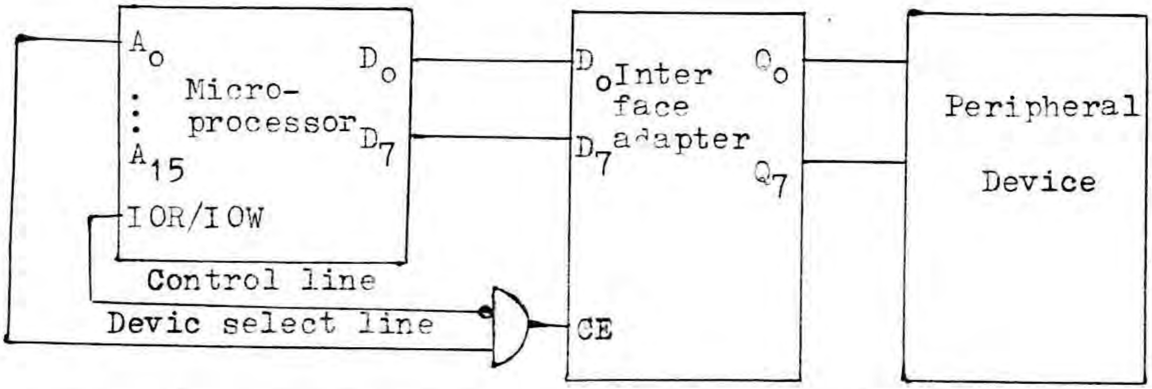


Fig. 4.3 Connection with microprocessor of the peripheral

In the designed EPROM programmer, the peripheral is the EPROM to be programmed housed in the 40-pin socket. The address decoder together with the buffers and latches form the output interface adapter through which address, data and control singnal pass to the EPROM. The buffers with the decoder form the 'input interface adapter' through which data to be read or verified is input to the microcomputer (Fig. Chapter-VI).

CHAPTER - V

PROGRAMMING CHARACTERISTICS OF INTEL EPROMs

5.1 Comparative Study of the Intel EPROMs

The EPROM Programmer can be used for programming the following type of EPROMs:-

i. 16k (2k x 8) UV Erasable PROM: Intel 2716

ii. 32k (4k x 8) UV Erasable PROM: Intel 2732

iii. 64k (8k x 8) UV Erasable PROM: Intel 2764

iv. 128k(16k x 8) UV Erasable PROM: Intel 27128

v. 1k x 8 PROM within the Intel 8048H single component 8 bit microcomputer.

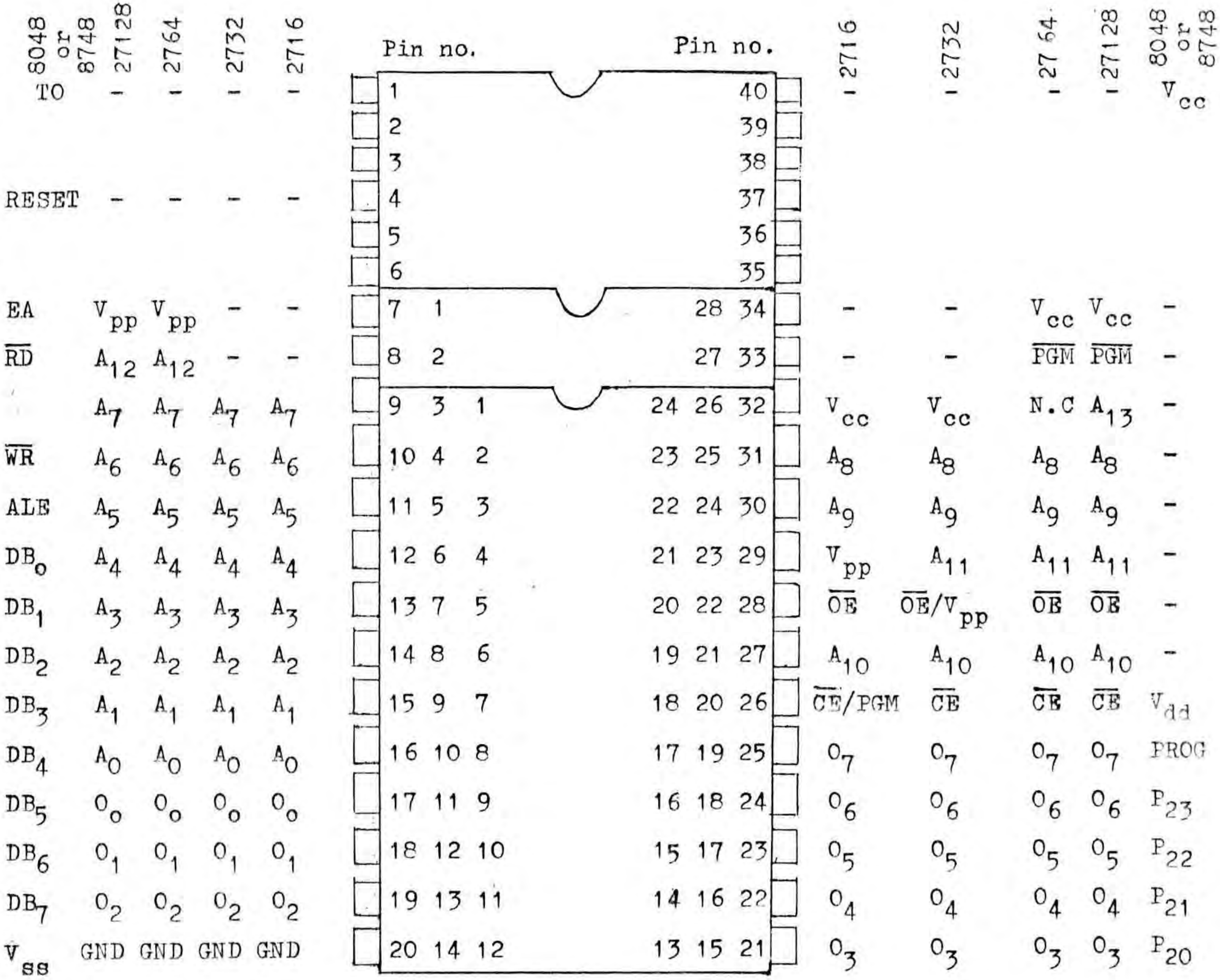
vi. 1k x 8 PROM within the Intel 8748H single-chip microcomputer.

A comparative pin diagram of all the EPROMs is shown in Fig.5.1. Individual pin diagrams, block diagrams and logic symbols of each of the above EPROMs are given in the appendix.

5.2 27-Series EPROM Operation

The EPROMs have the features of fast signle-address location programming and have, an access time from 250 ns to 650 ns and is ideal for use with the high-performance microprocessors. A block diagram is shown in Fig. 5.2 to represent the 27-series EPROMs.

The EPROMs have five modes of operation as listed in Table 5.1 the modes are briefly discussed below which will be helpful in the hardware design.



8048 or 8748	27128	2764	2732	2716	Pin no.			Pin no.			2716	2732	27 64	27128	8048 or 8748
TO	-	-	-	-	1					40	-	-	-	-	V_{cc}
					2					39					
					3					38					
RESET	-	-	-	-	4					37					
					5					36					
					6					35					
EA	V_{pp}	V_{pp}	-	-	7	1			28	34	-	-	V_{cc}	V_{cc}	-
$\overline{RD}$	A_{12}	A_{12}	-	-	8	2			27	33	-	-	$\overline{PGM}$	$\overline{PGM}$	-
	A_7	A_7	A_7	A_7	9	3	1	24	26	32	V_{cc}	V_{cc}	N.C	A_{13}	-
$\overline{WR}$	A_6	A_6	A_6	A_6	10	4	2	23	25	31	A_8	A_8	A_8	A_8	-
ALE	A_5	A_5	A_5	A_5	11	5	3	22	24	30	A_9	A_9	A_9	A_9	-
DB_o	A_4	A_4	A_4	A_4	12	6	4	21	23	29	V_{pp}	A_{11}	A_{11}	A_{11}	-
DB_1	A_3	A_3	A_3	A_3	13	7	5	20	22	28	$\overline{OE}$	$\overline{OE}/V_{pp}$	$\overline{OE}$	$\overline{OE}$	-
DB_2	A_2	A_2	A_2	A_2	14	8	6	19	21	27	A_{10}	A_{10}	A_{10}	A_{10}	-
DB_3	A_1	A_1	A_1	A_1	15	9	7	18	20	26	$\overline{CE}$/PGM	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	V_{dd}
DB_4	A_0	A_0	A_0	A_0	16	10	8	17	19	25	O_7	O_7	O_7	O_7	PROG
DB_5	O_o	O_o	O_o	O_o	17	11	9	16	18	24	O_6	O_6	O_6	O_6	P_{23}
DB_6	O_1	O_1	O_1	O_1	18	12	10	15	17	23	O_5	O_5	O_5	O_5	P_{22}
DB_7	O_2	O_2	O_2	O_2	19	13	11	14	16	22	O_4	O_4	O_4	O_4	P_{21}
V_{ss}	GND	GND	GND	GND	20	14	12	13	15	21	O_3	O_3	O_3	O_3	P_{20}

Fig. 5.1 Pin diagram of Intel EPROM

5.2.1 Read Mode:- The 27-series EPROMs have two control functions, both of which must be satisfied in order to obtain data at the outputs. Chip Enable ($\overline{CE}$) is the power control and should be used for device selection. Output Enable ($\overline{OE}$) is the output control and should be used to gate data to the output pins, independent of device selection.

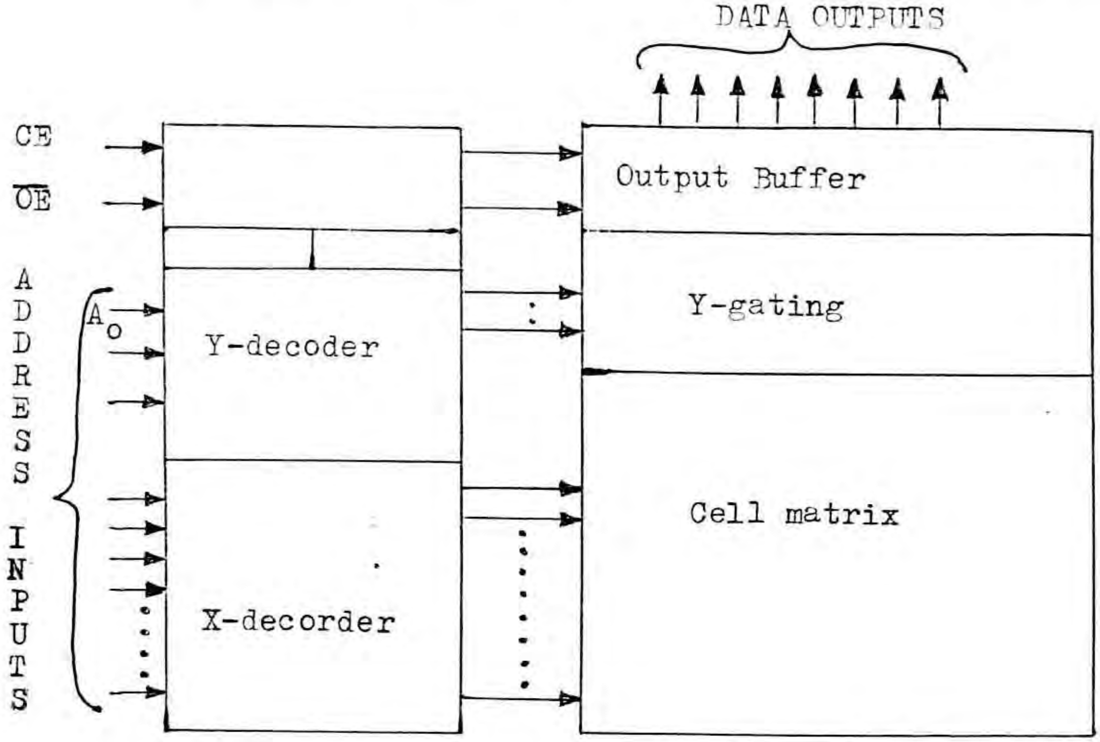


Fig. 5.2 Block diagram of EPROMs.

5.2.2 Standby Mode:- The 27-series are placed in the standby mode by applying a TTL high signal to the $\overline{CE}$ input. When in standby mode, the outputs are in a high impedance state, independent of $\overline{OE}$ input.

TABLE 5.1 COMPARATIVE STUDY OF MODE SELECTIONS FOR INTEL EPROMs

(28 pin configuration: EPROMs will be housed in the lower side of socket).

Mode	Pins Type of EPROMs	CE/PGM (20)	PGM (27)	OE/V_{pp} (22)	V_{pp} (23)	V_{cc} (24)	V_{cc} (28)	V_{pp} (1)	Outputs 11-13, 15-19
READ	2716	V_{IL}	-	V_{IL}	+5	+5	-	-	D_{out}
	2732 2732A	V_{IL}	-	V_{IL}	-	+5	-	-	
	2764	V_{IL}	V_{IH}	V_{IL}	-	-	+5	+5	
	27128	V_{IL}	V_{IH}	V_{IL}	-	-	+5	+5	
STAND-BY	2716	V_{IH}	-	X	+5	+5	-	-	High Z
	2732 2732A	V_{IH}	-	X	-	+5	-	-	
	2764	V_{IH}	X	X	-	-	+5	+5	
	27128	V_{IH}	X	X	-	-	+5	+5	
PRO-GRAM	2716	Pulsed V_{IL} to V_{IH}	-	V_{IH}	+25	+5	-	-	D_{in}
	2732 2732A	V_{IH}	-	25 21	-	+5	-	-	
	2764	V_{IL}	V_{IL}	V_{IH}	-	-	+5	+21	
	27128	V_{IL}	V_{IL}	V_{IH}	-	-	+5	+21	
PRO-GRAM VERIFY	2716	V_{IL}	-	V_{IL}	+25	+5	-	-	D_{out}
	2732 2732A	V_{IL}	-	V_{IL}	-	+5	-	-	
	2764	V_{IL}	V_{IH}	V_{IL}	-	-	+5	+21	
	27128	V_{IL}	V_{IH}	V_{IL}	-	-	+5	+21	
PRO-GRAM INHIB-IT	2716	V_{IL}	-	V_{IL}	+25	+5	-	-	High Z
	2732 2732A	V_{IH}	-	25 21	-	+5	-	-	
	2764	V_{IH}	X	X	-	-	5	21	
	27128	V_{IH}	X	X	-	-	5	21	

Contd. Table 5.1

1. X ----- don't care
2. Exceeding 22V on pin 1 (V_{pp}) will damage the 2764 and 27128 while programming
3. In the program mode the 2732A OE/V_{pp} input is pulsed from a TTL low level to 21V (25V for 2732). Exceeding 22V will damaged the 2732A.
4. V_{cc} must remain within 5 ± 5% Volts.
5. V_{IL} = Input low voltage = 0.1V to 0.8V
6. V_{IH} = Input high voltage = 2V to 6V
7. V_{cc} must be applied simultaneously or before V_{pp} and removed simultaneously or after V_{pp}.

5.2.3 Program Inhibit Mode:- Programming of multiple EPROMs in parallel with different data is also accomplished. Except for $\overline{CE}$/PGM or $\overline{CE}$ or $\overline{CE}$(or PGM)all like inputs (including $\overline{OE}$) of the parallel 27 -EPROMs may be common. An active signal to the $\overline{CE}$ pin of a particular EPROM will program that EPROM and others are inhibited from being programmed.

5.2.4 Program Verify Mode:- A verify should be performed on the programmed bits to determine that they were correctly programmed. The verify may be performed with V_{pp} at 25V (or 21V as the case may be). In case of 2716 V_{pp} should be at 5V except during programming and program verify. In case of 2732, 2764 & 27128 the verify is accomplished with $\overline{OE}$ & $\overline{CE}$ at V_{IL} with PGM at V_{IH}.

5.2.5 Programming Mode:- Initially after each erasure, all bits of the EPROMs are in the "1" state. Data is introduced by selectively programming "O's" into the desired bit locations. Although "O's" will be programmed, both "1's" and "O's" may be present in the data word.

a. Intel 2716 EPROM:

The 2716 is in the programming mode when the power supply

V_{pp} = 25V and

$\overline{OE}$ = V_{IH}

When the address and data are stable, a 50 msec, active-high, TTL program pulse is applied to the $\overline{CE}$/PGM input.

b. The 2732 EPROMs:

The 2732 & 2732A are in the programming mode when $\overline{OE}/V_{pp}$ = 21V.

When the address and data are stable, a 50 msec, active-low, TTL program pulse is applied to the $\overline{CE}$ input. The 2732A must not be programmed with a DC signal applied to the $\overline{CE}$ input.

c. The 2764 & 27128 EPROMs :

The 2764 & 27128 are in the programming mode when

V_{pp} = 21V

$\overline{CE} = \overline{PGM} = V_{IL}$

For programming, CE should be kept TTI low at all times while V_{pp} is kept at 21V. When address and data are stable, a 50 msec, active low, TTL program pulse is applied to $\overline{PGM}$ input.

For all the above EPROMs the data to be programmed is applied 8 bits in parallel to the data output pins. The levels required for the address and data inputs are all TTL. A program pulse must be applied at each address location to be programmed. Any location can be programmed at any time-either individually, sequentially or at random. The program pulse has a maximum width of 55 msec[5].

5.3 Programming of 8748H & 8048H EPROM

The programming process consists of:

i. activating the program mode

ii. applying an address

iii. latching the address

iv. applying data

v. applying program pulse.

Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a description of their functions:

Pins	Function
XTAL 1	Clock input 1 to 3 MH_z
Reset	Initialization and address latching
Test 0	Selection of program or Verify mode
EA	Activation of Program/Verify mode
BUS	Address and data input,data output during verify
P_{20}-P_{22}	Address input
V_{DD}	Programming power supply
PROG	Program pulse input

The program/verify sequence is:

1. V_{DD} = 5V, clock applied or internal oscillator operating $\overline{\text{Reset}}$ = 0V, TEST 0= 5V, EA = 5V, BUS and PROG floating, P_{10} and P_{11} must be tied to ground.

2. Insert 8748H in programming socket.

3. TEST 0= 0V (select program mode).

4. EA = 18V (activate program mode)

5. Address applied to BUS and P_{20} - P_{22}.

6. $\overline{\text{RESET}}$ = 5V (latch address)

7. Data applied to BUS

8. V_{DD} = 21V (programming power)

9. PROG = OV followed by one 50 msec pulse to 18V

10. V_{DD} = 5V

11. TEST 0 = 5V (verify mode)

12. Read and verify data on BUS

13. TEST 0 = OV

14. RESET= OV and repeat from step 5

15. Programmer should be at conditions of step 1 when the EPROMs are removed from socket.

CHAPTER - VI

HARDWARE DESIGN OF EPROM PROGRAMMER

6.1 Functional description of EPROM Programmer

A simplified block diagram of the EPROM porgrammer is shown in Fig. 6.1 and consists of the following major blocks.

a. The address decoder (ADDEC)

b. The I/O data buffers (DABUF)

c. The expansion buffer (EXBUF)

d. The address latches

i. Lower address latch (ADL)

ii. Higher address latch (ADH)

e. The data latch

i. EPROM data latch (DR)

ii. Printer data latch (PRT)

f. The control latches

i. Port controlling latch (PCR)

ii. EPROM controlling latch (CR)

The address decoder is used to select the address, data and control latches and the I/O buffers and the expansion buffer. The B8 to BF (Hexadecimal) address range is used by the address decoder for the selection purpose.

The I/O data Buffers are used for transferring data to and from the microcomputer. The expansion buffer is reserved for future expansion of the system.

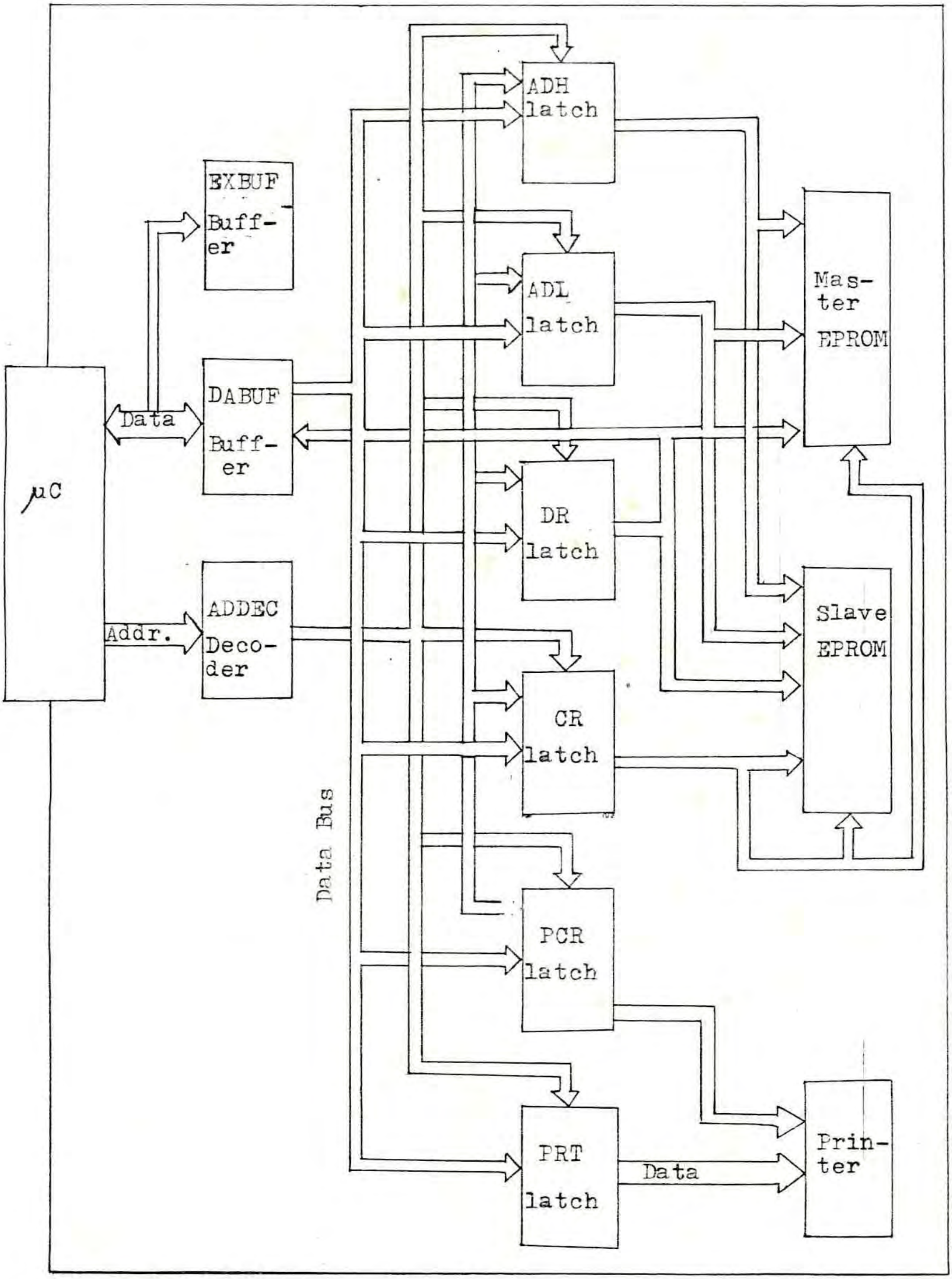


Fig. 6.1 Block Diagram of EPROM Programmer

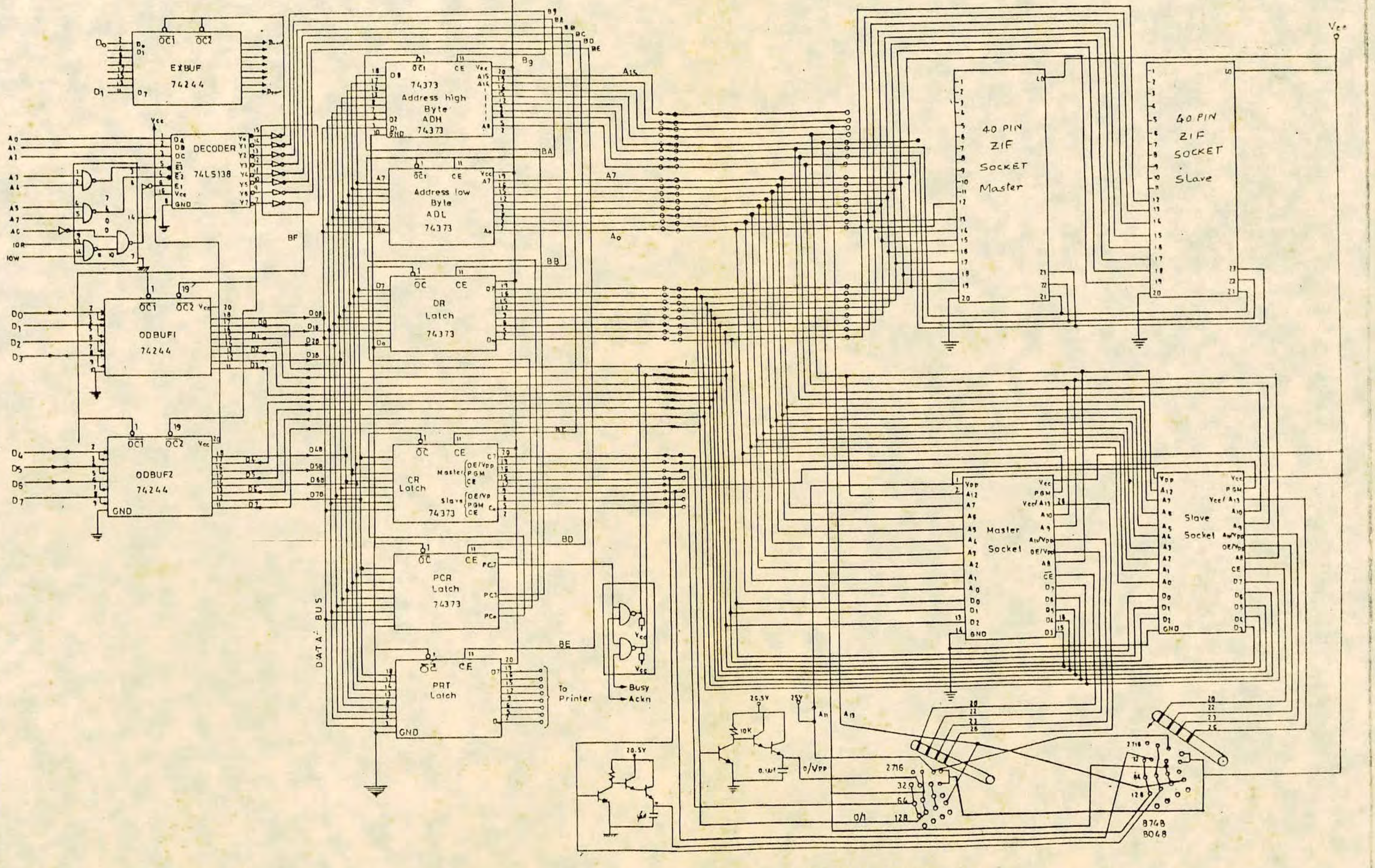


Fig. 6.2 Circuit diagram of EPROM Programmer

The control latches are used for sending various signals to the EPROMs for the mode selections and also for controlling the remaining latches. Control lines are program-controlled so that the Intel EPROM series (2716, 2732,2764, 27128 etc.) can be programmed.

The data latches are used to latch the data to be programmed and upon receiving output control ($\overline{OC}$) signal they send the data to the desired location.

The address latches are used to latch the address of the location of the EPROM to be programmed and upon receiving the output control signal they put the address on the desired pins of the EPROM.

6.2 Detail Circuit Diagram of the EPROM Programmer

A detailed schematic diagram of the designed EPROM programmer is shown in Fig. 6.2. The description of each functional unit is given below.

6.2.1 The address decoder (ADDEC)

There are six latches and two sets of buffers in the designed programmer. Their address are to be B8 to BF. This is accomplished with the help of the 3-line to 8-line decoder (74LS138) IC Chip in conjunction with the 2-input NAND gates. The pins D_A, D_B, D_C of the decoder are connected to the A_8,A_9 and A_{10} lines respectively of the address bus of the microcomputer.

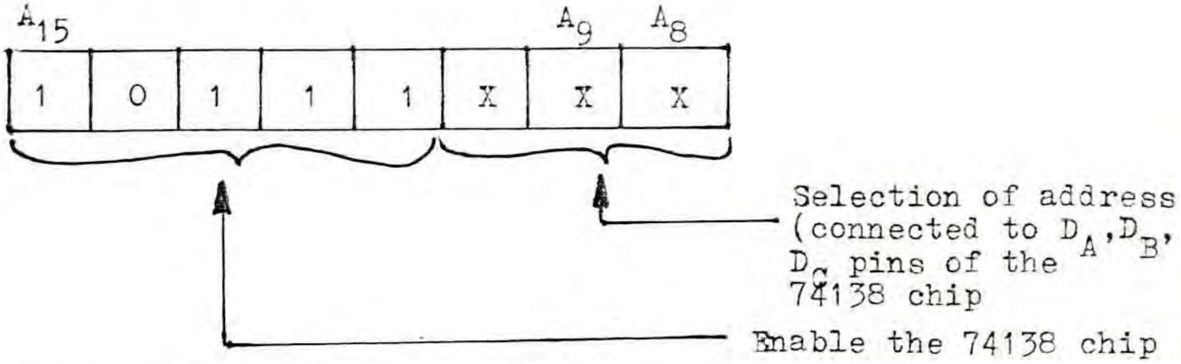
The six outputs ($\overline{Y}_1$ to $\overline{Y}_6$) bearing the address B9 to BE, of the decoder are connected to the chip Enable (CE) pin of the six latches, after being inverted by the 7404 inverter, as because the Chip Enable of the latches are active High. The addresses of all the latches and buffers are given in Table 6.1.

Table 6.1 Address of the Latches

Sl.No.	Name of the latch/Buffer	Address designated (in hexadecimal)	CE/$\overline{OC}$(Buffer) connection with decoder pin
1.	I/O Data Buffer(DABUF1 & DABUF2)	B8	$\overline{Y}_0$(Pin 15)
2.	Higher address latch(ADH)	B9	Y_1(Pin 14)
3.	Lower address latch(ADL)	BA	Y_2(Pin 13)
4.	EPROM Data latch(DR)	BB	Y_3(Pin 12)
5.	EPROM controlling latch (CR)	BC	Y_4(Pin 11)
6.	Port control latch(PCR)	BD	Y_5(Pin 10)
7.	Printer data latch(PRT)	BE	Y_6(Pin 9)
8.	Expansion Buffer(EXBUF)	BF	$\overline{Y}_7$(Pin 7)

If the CE pin of a particular latch is HIGH, the contents of the data bus will be latched in the register. The CE pin will be HIGH when the specified latch is addressed. This data will be outputted only when the output control ($\overline{OC}$) pin is LOW. This $\overline{OC}$ pin is controlled by the port control register (PCR).

Using this decoder the following binary format of the high address bus (Fig. 6.3) may be used for address selection between B8 to BF.



Selection of address (connected to D_A, D_B, D_C pins of the 74138 chip

Enable the 74138 chip

Fig.6.3 Addressing format of the address decoder.

The decoder should be enabled with the above address only during input/output operation i.e., when $\overline{IOR} = 0$ or $\overline{IOW} = 0$.

The decoder for decoding the address has three enabling pins, $\overline{E}_3$, $\overline{E}_2$, which are active low and E_1 which is active high. The decoder will output LOW at one of its 8 output pins depending on the value of D_A, D_B, D_C inputs when all three pins $\overline{E}_3$, $\overline{E}_2$ and E_1 are activated.

The input to $\overline{E}_3$ pin is from the output pin of a NAND gate whose inputs are A_{11} and A_{12} lines of the address bus and that to $\overline{E}_2$ pin is from the output of another NAND gate whose inputs are A_{13} and A_{15} lines of the address bus. So $\overline{E}_3$ and $\overline{E}_2$ are selected only when $A_{11} = 1$, $A_{12} = 1$, $A_{13} = 1$ and $A_{15} = 1$. Another logic circuit is necessary to select E_1 such that E_1 will be high only when $A_6 = 0$ and $IOR = 0 \overset{\text{or}}{/} IOW = 0$. Accordingly the input to E_1 pin will come from a logic circuit which will follow the truth table and/its Karnaugh map involving the don't care conditions as given in Fig. 6.4

IOW	IOR	A_{14}	Output E_1
0	0	0	d
0	0	1	d
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Fig. 6.4(a) Truth table.

	$\overline{IOR}\,\overline{IOW}$	$\overline{IOR}IOW$	$IORIOW$	$IOR\overline{IOW}$
$\overline{A_{14}}$	d	1		1
A_{14}	d			

Fog. 6.4(b) Karnaugh map

The boolean function of the above truth table will be

$$E_1 = \overline{A_{14}}\,\overline{IOW} + \overline{A_{14}}\,\overline{IOR}$$

$$= \overline{\overline{\overline{A_{14}}(\overline{IOW} + \overline{IOR})}}$$

$$= \overline{\overline{\overline{A_{14}}} + \overline{\overline{IOW} + \overline{IOR}}}$$

$$= \overline{A_{14} + \overline{\overline{IOW}}.\overline{\overline{IOR}}}$$

$$= \overline{A_{14} + IOW.IOR}$$

$$= \overline{A_{14}}\ (\overline{IOW.IOR})$$

$$= \overline{\overline{\overline{A_{14}}.\overline{IOW.IOR}}}$$

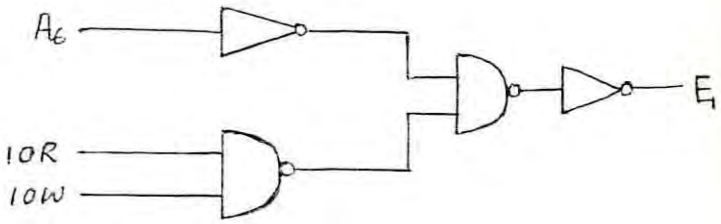


Fig. 6.5 Logic Circuit for the above truth table

Therefore the logic circuit becomes as shown in Fig.6.5.

6.2.2 I/O and Expansion Buffers (DABUF1, DABUF2,EXBUF)

74LS244 buffers are used which has two control pins. Each control pin controls four bits of data. The $\overline{OC1}$ controls four output bits and $\overline{OC2}$ controls four input bits. Two buffers are used for handling eight outgoing and eight incoming bits. The two buffers DABUF1 and DABUF2 are used for transferring data to microcomputer received from the EPROM in case of reading and verifying the EPROM content and for receiving data from the microcoumpter for onward transmission to different latches via the data bus. The address of these two bus buffers are B8. These buffers are designed to be always in the output mode by connecting the $\overline{Y}_o$ output of the decoder after being inverted by the inverter to $\overline{OC1}$ pins of the buffers. The output control pins $\overline{OC1}$ and $\overline{OC2}$ are active LOW.

For all OUT instructions (OUT B9 to OUT BF), the decoder outtput pin $\overline{Y}_o$ (pin 15 of 74138) will be HIGH and after being inverted it will be LOW. Therefore these buffers will be always in the output mode and data will be outputted to the data bus from the microcomputer. The outtut pin $\overline{Y}_o$ is directly connected to the output control $\overline{OC2}$ pin of the buffers. So that when IN instruction to address B8 is executed only then the pin $\overline{Y}_o$ is LOW and these buffers are in the input mode as $\overline{OC2}$ is directly connected to $\overline{Y}_o$ of decoder and data from the specified address of the EPROM will be inputted to the buffers and will be transmitted to the microprocessor. The output pin $\overline{Y}_7$ is directly connected to the expansion buffer (EXBUF) and here 8-bit data can only be outputted and cannot be inputted.

6.2.3 The Address High and Address Low Registers(ADL & ADH)

The address high latch (ADH) is used for latching the high byte address (A_8-A_{15}) and the latch ADL is used for latching low byte address (A_0-A_7) of the EPROM to be programmed. At first the high byte address of the location of the EPROM is loaded in the accumulator of the microprocessor. Then the content of the accumulator is transferred to the ADH latch using the instruction OUT B9. When OUT B9 instruction is used the $\overline{OC}$1 pins of the two data buffers (DABUF1, DABUF2) is LOW and the CE pin of the latch ADH is high and hence the content of the accumulator is transferred to the ADH latch and is latched there. In the same manner the LOW byte address of the location of the EPROM is latched in the ADL latch. Here the instruction transferring the low byte address is OUT BA. The $\overline{OC}$ pin remains high so that output is 3-stated. The $\overline{OC}$ pins of all the latches is controlled by the port control register (PCR) latch.

6.2.4 The Data latches (DR & PRT Latches)

The data to be transferred to the location specified by the content of the ADH and ADL latches is loaded in the accumulator and is transferred and latched in the Data register (DR)using the instruction OUT BB. Its output control ($\overline{OC}$) pin also remains high. The output control $\overline{OC}$ is activated by the second bit of the PCR latch.

In a similar way data is transferred to the latch PRT using the instruction OUT BE. It's $\overline{OC}$ pin is grounded so that it is of the latch PCR.

Port control latch (PCR) controls the output pin ($\overline{OC}$) of the remaining latches. Control information is transferred to the PCR from accumulator using the instruction OUT BD. Output control pin ($\overline{OC}$) is grounded so that whenever its chip enable (CE) pin is high, the content of the data bus is latched in the PCR and is outputted at the same time. The format of the Port control latch for output selection of the remaining latches are shown in Fig. 6.6.

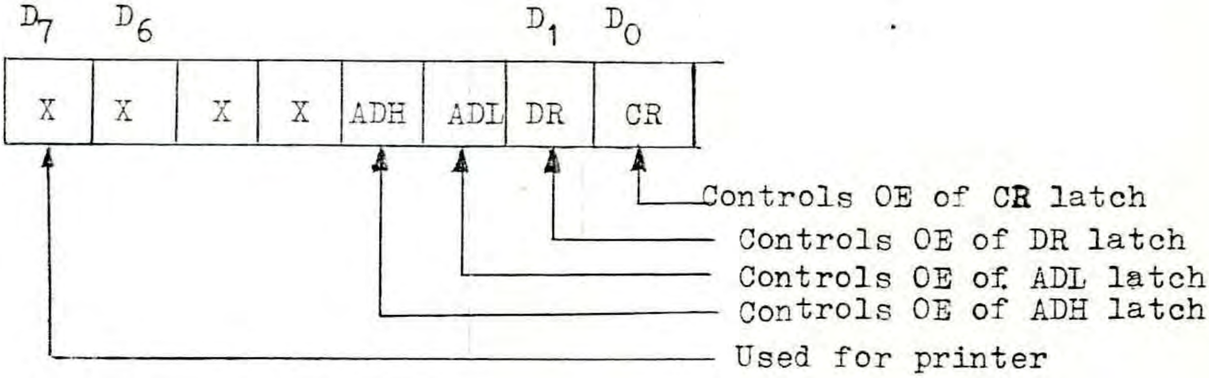


Fig. 6.6 Format of Port control latch

When the output Q is Low, the corresponding latch outputs the latched data in it.

Table 6.2 Control Commands for Various Functions on the EPROMs

Mode	Type of EPROM	Control Signal								Hex
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
READ MASTER	2716 & 2732	1	X	1	0	X	0	X	X	
	2764 & 27128	1	X	1	0	1	0	X	X	
READ SLAVE	2716 & 2732	0	X	0	1	X	1	X	X	
	2764 & 27128	0	1	0	1	X	1	X	X	
PROGRAM MASTER	2716	1	X	1	1	X	Pulsed V_{IL} to V_{IH}	X	X	
	2732	1	X	1	0	X	1	X	X	
	2764 128	1	1	1	1	0	0	X	X	
PROGRAM SLAVE	2716	1	X	Pulsed V_{IL} to V_{IH}	1	X	1	X	X	
	2732	1	X	1	1	1	1	X	X	
	2764 128	1	0	0	1	1	1	X	X	
PROGRAM VERIFY	2716	1	X	1	0	X	0	X	X	
	2732	1	X	1	0	X	0	X	X	
	2764 128	1	X	1	0	1	0	X	X	
PROGRAM VERIFY SLAVE	2716	0	X	0	1	X	1	X	X	
	2732	0	X	0	1	X	1	X	X	
	2764 128	0	1	0	1	X	1	X	X	

6.3 Connection of the EPROM Pins

From a comparative study of the pin diagrams of all the EPROMs (Fig. 5.4) it can be seen that the 27-series EPROMs pin diagrams are almost similar. The difference in the pin connections are shown in Table 6.3. A MASTER IC socket (40 pin) and a SLAVE IC socket (40 pin) with zero insertion force (ZIF) are used for placing any of the EPROMs type during programming. The EPROMs are always placed at the bottom of the socket as shown in Fig. 5.1. Since the 8048H and 8748H microcomputer EPROMs differ significantly from the 27-series, a different connecter is used for programming but with the same ZIF socket. The two connecter connection can be seen from the detailed circuit diagram in Fig. 6.2.

The address pins A_0-A_7 are connected to the 8 output pins D_0-D_7 of the ADL latch. The next three address pins A_8-A_{10} and A_{12} are also connected permanently to the three lower output pins D_0-D_2 and D_4 lines of the ADH latch. The A_{11} and A_{13} pins are connected to the D_3 and D_5 pins of the ADH latch via the selector switch because the connection of these pins are variable for diff. EPROMs.

The data pins D_0-D_7 of EPROMs are connected to the output pins D_0-D_7 of DR latch and also they are connected directly to the eight input pins of the buffers DABUFI & DABIUF2 so that the data can be inputted to the μC directly when needed. The data of either slave of Master EPROM will be inputted to the

Table 6.3 Difference in the Pin connections of different EPROM

Pin No. 28-Pin Configuration	Pin no. 24-Pin configuration	2716	2732	2764	27128	Comment
1	-	-	-	V_{pp}	V_{pp}	May be connected permanently to socket
2	-	-	-	A_{12}	A_{12}	
3-19	1 - 17	$(A_0 - A_7)$and $(O_0 - O_2)$and GND and$(O_3 - O_5)$				Same for all EPROMs
20	18	CE/PGM	CE	CE	CE	Variable on type of EPROM So a switch is to be used
22	20	OE	OE/V_{pp}	OE	OE	
23	21	V_{pp}	A_{11}	A_{11}	A_{11}	
26	24	V_{cc}	V_{cc}	n.c	A_{13}	
27	-	-	-	PGM	PGM	May be connected permanently to socket.
28	-	-	-	V_{cc}	V_{cc}	

microcomputer via the buffers depending on whose $\overline{OE}$ enable is activated. The programming voltage is supplied to the programming pins in case of programming an EPROM at either Master or Slave location from an outside source of 25V or 21V as the case may be. Since the +25V V_{pp} required by the EPROM must be off except when programming or verifying, the switch is designed to be in the off state when the processor is turned on. A high on the switch input turns it off.

With the switch off, the diode to +5V turns on and supplies the V_{pp} (for 2716) pin with the voltage required for normal read operation. This input is controlled by the CR register output. When the switch to 25V is on, the diode is reverse biased. The switching circuit is shown in Fig. 6.7.

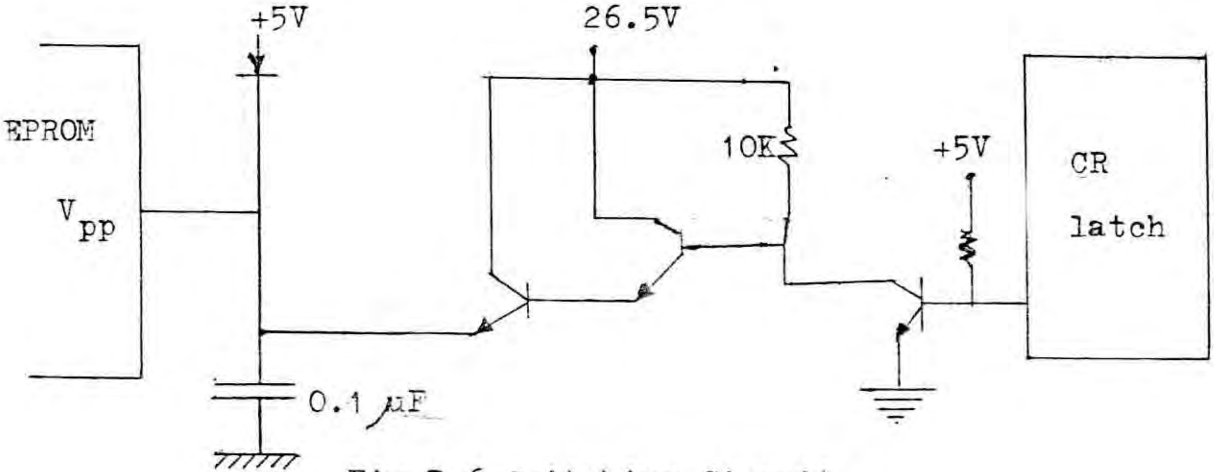


Fig.7.6 Switching Circuit

Due to voltage drop across transistor a higher voltage than 25V(or 21V) must be used as the switch supply. A 0.1 µF capacitor on the V_{pp} pin helps prevent overshoot which might destory the EPROM during switching.

CHAPTER - VII

SOFTWARE DEVELOPMENT

7.1 Functions of the Programmer

The following operation will be performed by the microprocessor-based programmer.

1. Loading the data to be programmed from a selected input device (Tape, EPROM etc.) into the micro-computer memory.

2. Programming a segment of a PROM with the data which are stored begining at a specified address in the microcomputer memory.

3. Displaying the contents of a segment of a PROM at the Master or Slave location or of the main memory on the monitor screen.

4. Moving a block of data from one memory location to another memory location.

5. Transferring a block of data in a PROM into the microcomputer memory so.that the contents of the PROM may be verified or examined through the system and may also be used to duplicate a PROM.

6. Transferring a block of data from a PROM to the printer to make a hardcopy of the content.

7. Comparing a block of data in a PROM with the contents of a segment of memory (Program Verification).

8. Verifying a segment of a PROM to see whether the segment is erased.

9. Printing the contents of a segment of memory.

Hardware has been designed so that data to be programmed into the EPROM can come from another ROM/EPROM or from RAM via CPU.

7.2 Program Development for the Programmer

Flow diagram of the program to perform the functions described above is shown in Fig. 7.1. To perform the operations as shown in the flow diagram seven subprograms are used. These subprograms are called by the main program (Main Menu) as desired by the user. The names and functions of subprograms are as follows:

i. READ: Used to read the contents of an EPROM at Master location or at slave location and store it to a desired memory location.

ii. MOVE: Used for moving a block of data from a certain memory location to another location.

iii. DISPLAY: Used to display the contents of an EPROM at Master location or at Slave location.

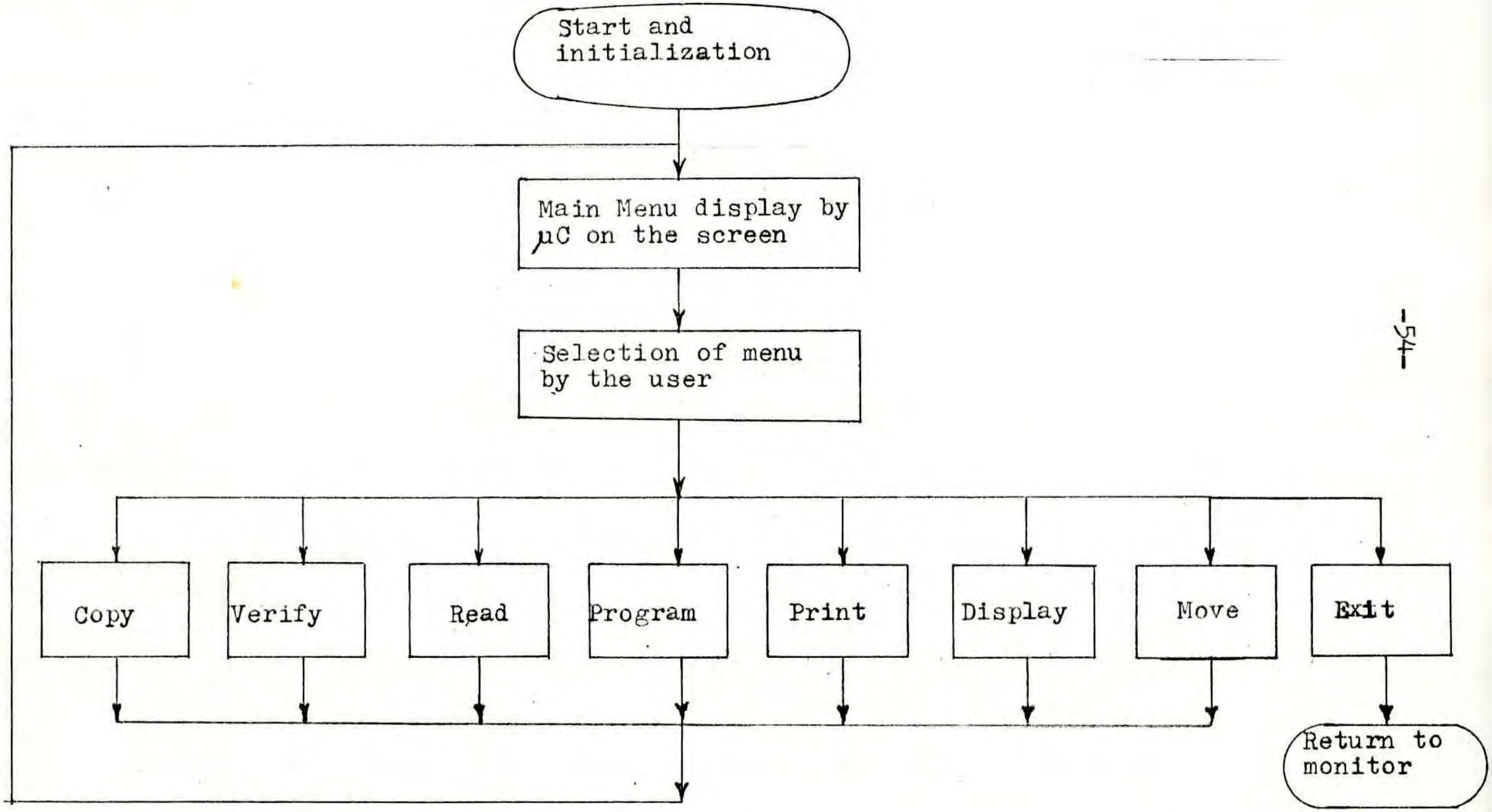


Fig. 7.1 Flow diagram of the program to perform different functions.

iv. VERIFY: Used to verify if any desired segment of the EPROM at Slave or at Master location erased.

v. COPY: Used for copying an EPROM in Slave location from an EPROM at Master location.

vi. PROGRAM: Used for programming any segment of the EPROM at Master location from the main memory.

vii. PRINT: Used for printing the data from the main memory or from EPROM at Master or Slave location.

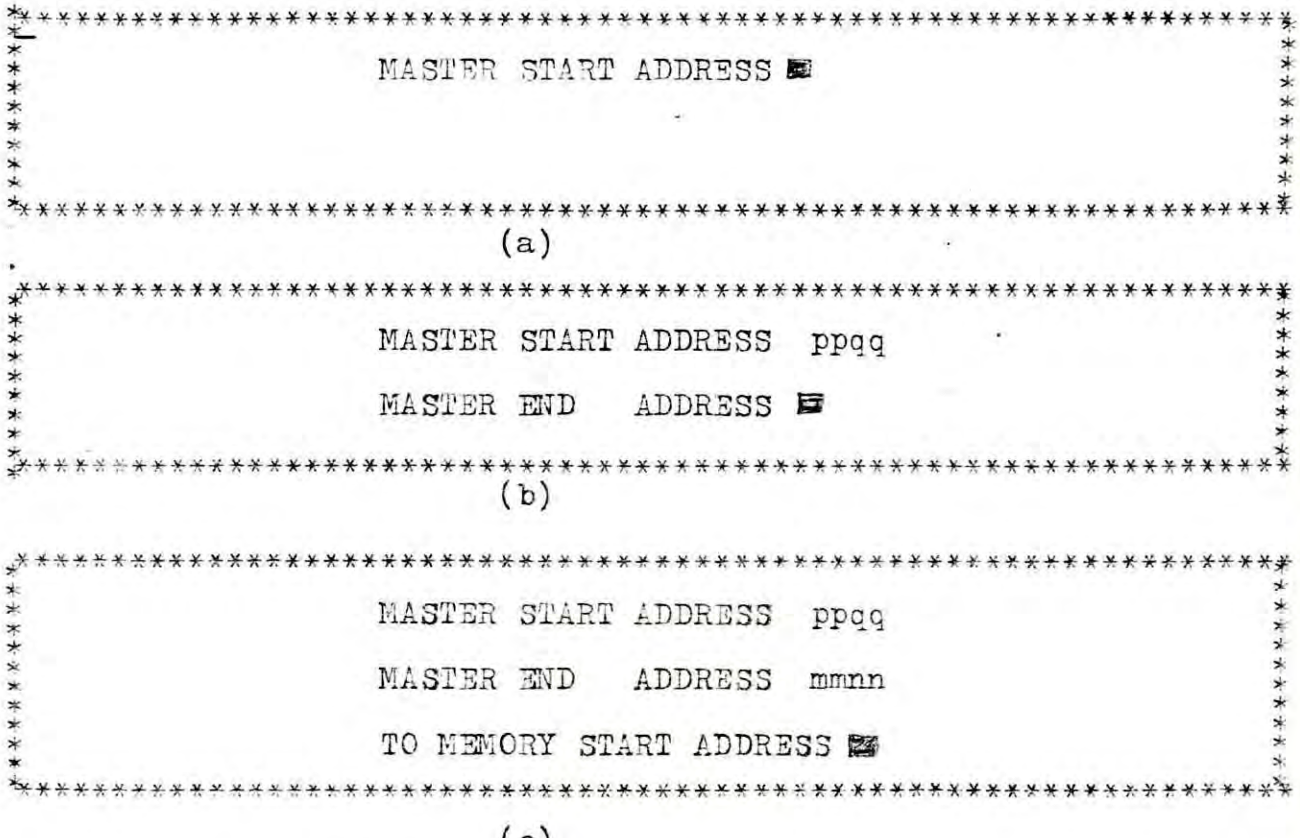
The different variable names used in these subprograms are given in Table 7.1.

When the program is executed main Menu will be shown on the monitor screen of the system. The format of the main Menu as will be shown on the screen is shown in Fig.7.2. There are seven choices of operations to be performed as shown in the figure and for each choice there is a corresponding key to be pressed on the key board. After pressing a particular key the corresponding operation format for the corresponding function will be displayed on the screen. Now the different operations will be discussed in the following sections.

7.2.1 READ Subprogram

Read subprogram is used to read the contents of an EPROM at Slave or Master location and store it to a desired memory location. After the main Menu is displayed if the

Once the source EPROM is selected, the microcomputer waits for the start and end address of the EPROM and also for the destination address in the memory of the microcomputer starting from which data block from the EPROM will be stored. Each of the address will be entered by the user from the key board interactively The interaction of the user and the computer for transferring data from the Master EPROM is shown in Fig. 7.4(a), (b), (c). As each line of the prompt is displayed the corresponding address is to be entered with carriage return at the end. Next line appears only after the previous line is entered. The typed address can be corrected using the cursor movement.



```
MASTER START ADDRESS ■
```

(a)

```
MASTER START ADDRESS  ppqq

MASTER END   ADDRESS ■
```

(b)

```
MASTER START ADDRESS  ppqq

MASTER END   ADDRESS  mmnn

TO MEMORY START ADDRESS ■
```

(c)

Fig. 7.4 Interactive displays of Master EPROM for READ operation.

An overall flow chart of the Read program is shown in Fig. 7.5(a).

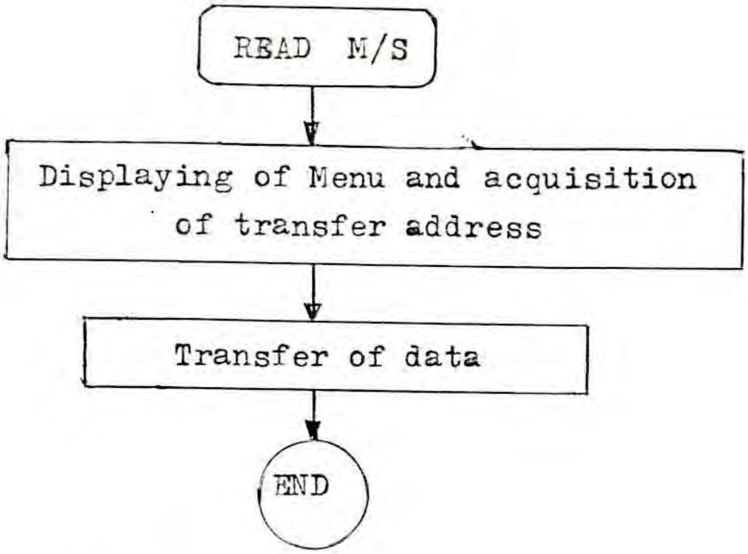


Fig. 7.5 (a) Overall Flow Chart for READ Program

The detailed flow chart of read display menu. is shown in Fig. 7.5(b) and that of the actual data reading is given in Fig. 7.5(c). The detail program is given in the appendix.

7.2.2 Conversion of an ASCII-coded hexadecimal number to its binary eouivalent

Data to be read from Master from a certain address,to be programmed in Slave from certain address etc. are entered by punching the number on the key board manually. This number is to be stored in the specified address. A chart for the specified address and thier functions are given in the Table 7.2.

When a key containing a number is pressed its ASCII code is loaded in the accumulator .. This ASCII code should be converted

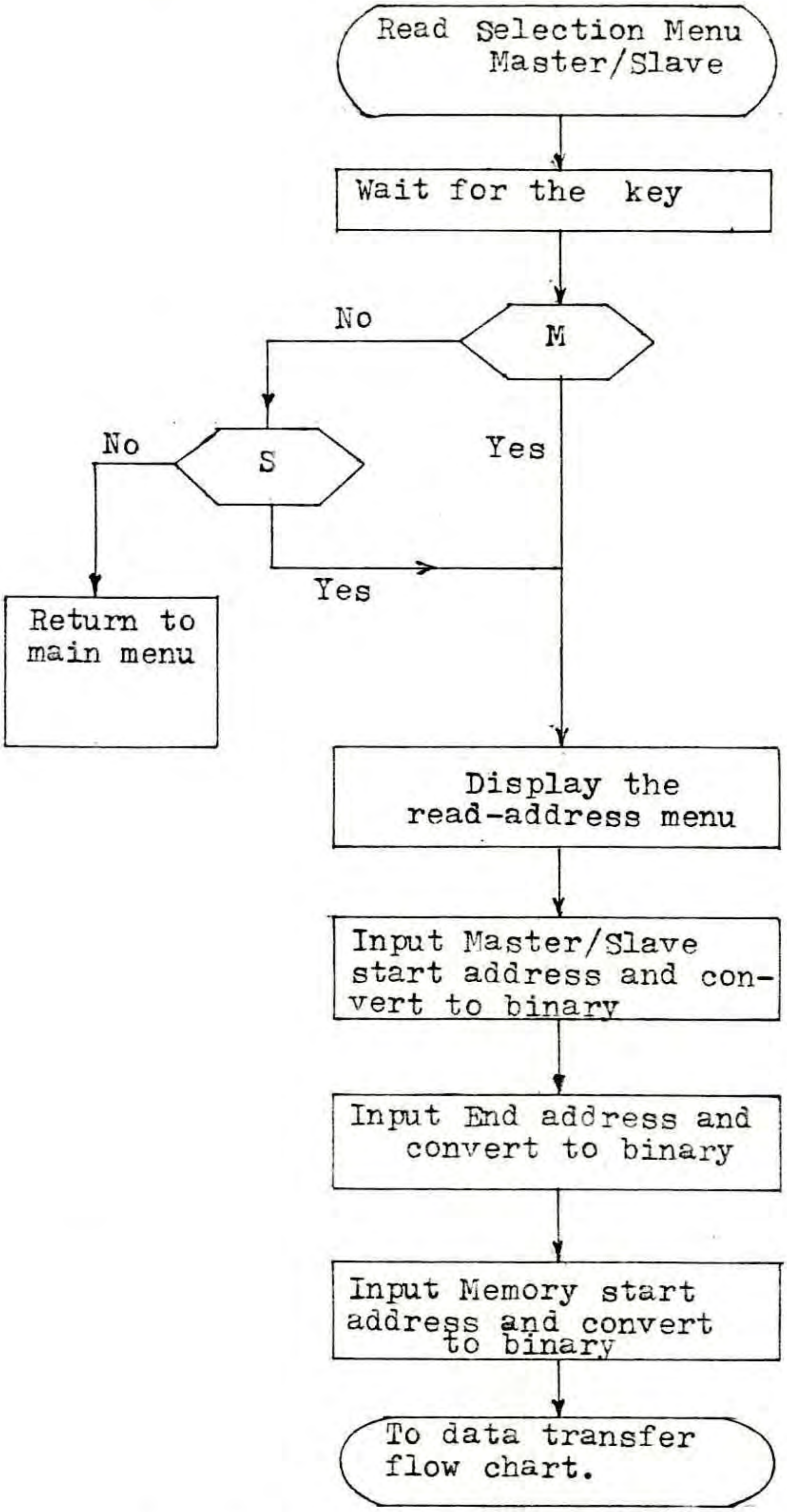


Fig. 7.5(b) Display of Read Menu and acquisition of address flow chart.

READ an EPROM at Master/Slave location

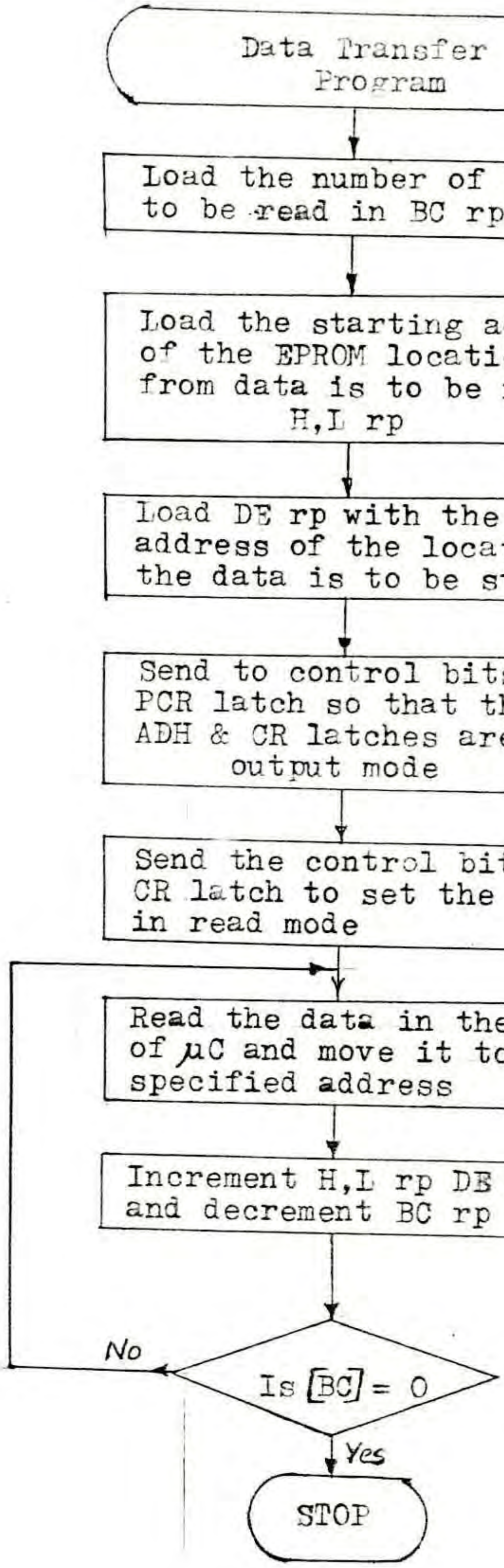

Data Transfer Program
Load the number of bytes to be read in BC rp.
Load the starting address of the EPROM location where from data is to be read in H,L rp
Load DE rp with the starting address of the location where the data is to be stored
Send to control bits to the PCR latch so that the ADL, ADH & CR latches are in the output mode
Send the control bits from CR latch to set the EPROM in read mode
Read the data in the Acc. of µC and move it to a specified address
Increment H,L rp DE rp and decrement BC rp
No
Is [BC] = 0
Yes
STOP

Fig. 7.5(c) Flow chart for Data Transfer for READ operation.

to the corresponding number and then stored in the specified memory location. One byte should be stored in one memory location.

It can be seen from the Table that ASCII Code for the number 0 through 9 are 30 Hex to 39 Hex and A through F are 41 Hex through 46 Hex. So to convert 30H through 39H to number 0 through 9, 30H is to be subtracted or four high bits are to be cleared, and to make 41H to 46H to Hex number A through F, 37_{16} is to be subtracted.

The flow chart for ASCII-code to binary conversion routine required for each entry of the address is shown in Fig.7.6.

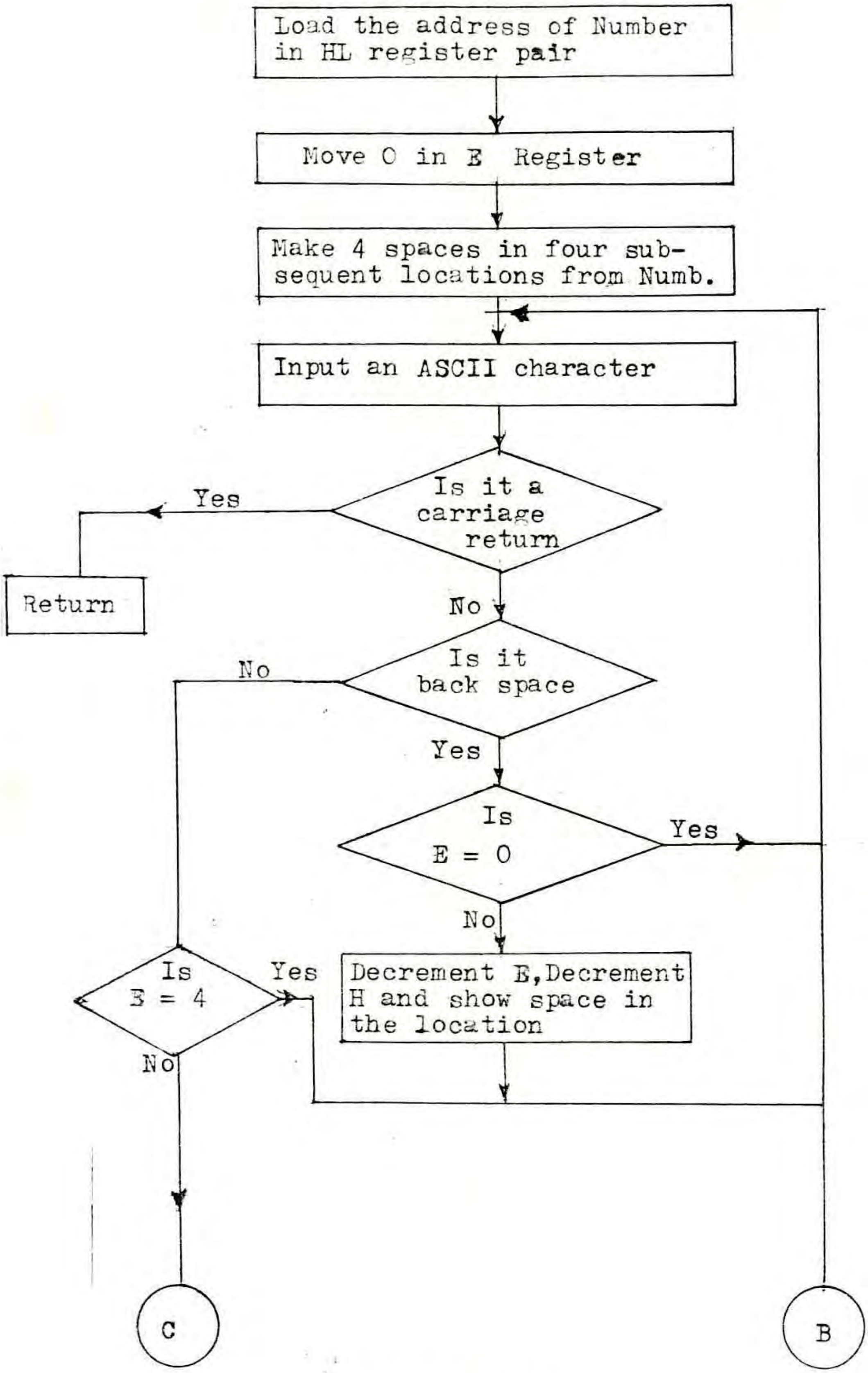

Load the address of Number in HL register pair
Move 0 in E Register
Make 4 spaces in four subsequent locations from Numb.
Input an ASCII character
Is it a carriage return
Yes
Return
No
Is it back space
No
Yes
Is E = 0
Yes
No
Decrement E, Decrement H and show space in the location
Is E = 4
Yes
No
C
B

Fig.7.6 (Contd.)

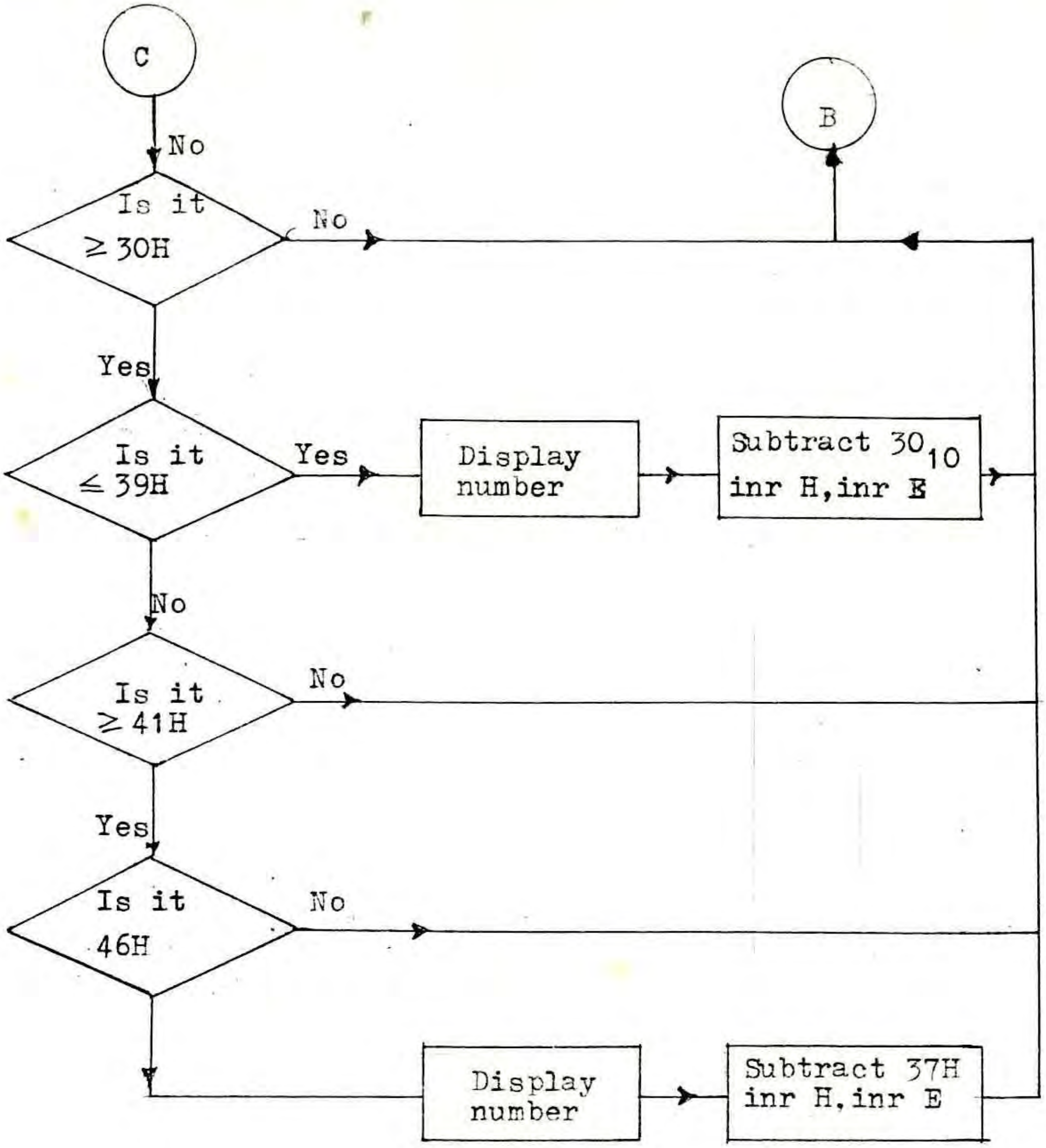


Fig. 7.6 Flow chart for ASCII to binary conversion

7.2.3 MOVE Subprogram

It is used for transferring a block of data from a certain memory location to another memory location. After the main Menu is displayed on the screen, if the key M is pressed, the screen will display the MOVE Menu as shown in Fig. 7.7(a)

```
*****************************************************************
*                                                               *
*                                                               *
*         M : MOVE_DATA_FROM_MEMORY_TO_MEMORY                   *
*                                                               *
*                                                               *
*                 MEMORY START ADDRESS █                        *
*                                                               *
*                                                               *
*****************************************************************
```

Fig. 7.7(a)

The cursor will be at the position shown. The starting address will have to be entered manually. After entering the address if carriage return key is pressed the format shown in Fig.7.7(b) will be displayed with the cursor shown at the position. Ending address will have to be entered manually.

```
*****************************************************************
*                                                               *
*                                                               *
*        M : MOVE_DATA_FROM_MEMORY_TO_MEMORY                    *
*                                                               *
*                                                               *
*                 MEMORY START ADDRESS  ppqq                    *
*                                                               *
*                                                               *
*                            UNTIL  █                           *
*                                                               *
*****************************************************************
```

Fig.7.7(b) Interactive Displays of Move Menu

The CR key will have to be pressed to get the next instruction and the farmat shown in Fig. 7.7(c) will be displayed. The memory starting address wherefrom the data is to be stored will have to be entered manually.

```
M : MOVE DATA FROM MEMORY TO MEMORY

          MEMORY START ADDRESS  ppqq

                         UNTIL  mmnn

       TO MEMORY START ADDRESS  ■
```

Fig. 7.7(c) Interactive Displays of Move Menu

The flow chart consists of mainly two parts:

i. Dispalying of the operating format and entering the starting and ending addresses.

ii. Transferring the data from one location to another location.

The flow charts are shown in Fig.7.8(a) and 7.8(b).

7.2.4 DISPLAY Subprogram

It is used to display the contents of an EPROM at Master location or at Slave location. In this operation the data is to be fetched from the EPROM at Master or slave location and is to be displayed on the screen instead of storing in the memory with the difference that in DISPLAY routine the contents are displayed instead of being stored and in READ operation the data are stored in memory.

The operation and flow chart are similar to the READ subprogram excepting that instead of moving the fetched data to the memory location it is displayed.

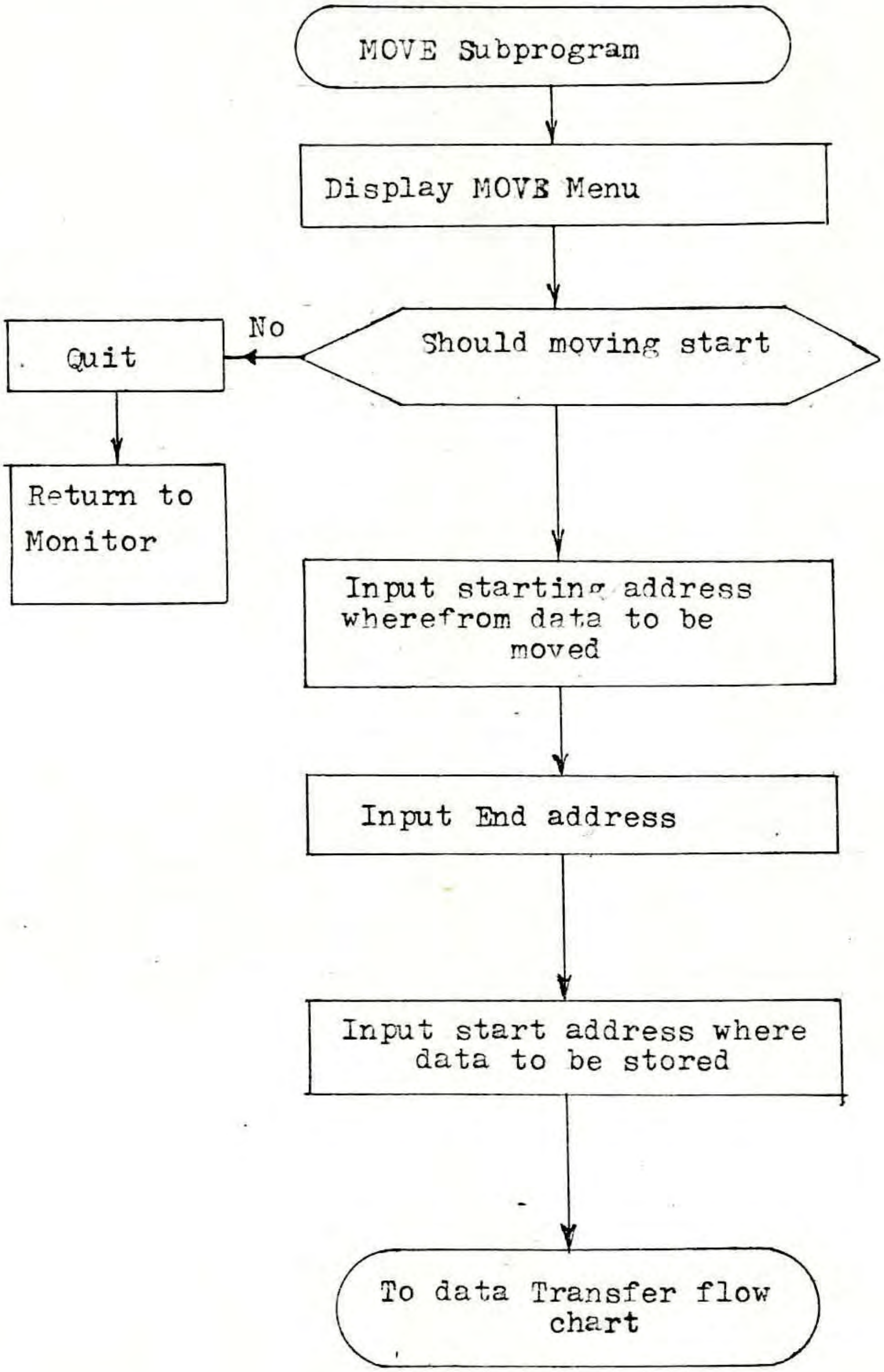

MOVE Subprogram
Display MOVE Menu
Should moving start
No
Quit
Return to Monitor
Input starting address wherefrom data to be moved
Input End address
Input start address where data to be stored
To data Transfer flow chart

Fig. 7.8(a) Flow chart for MOVE Menu display and address entry.

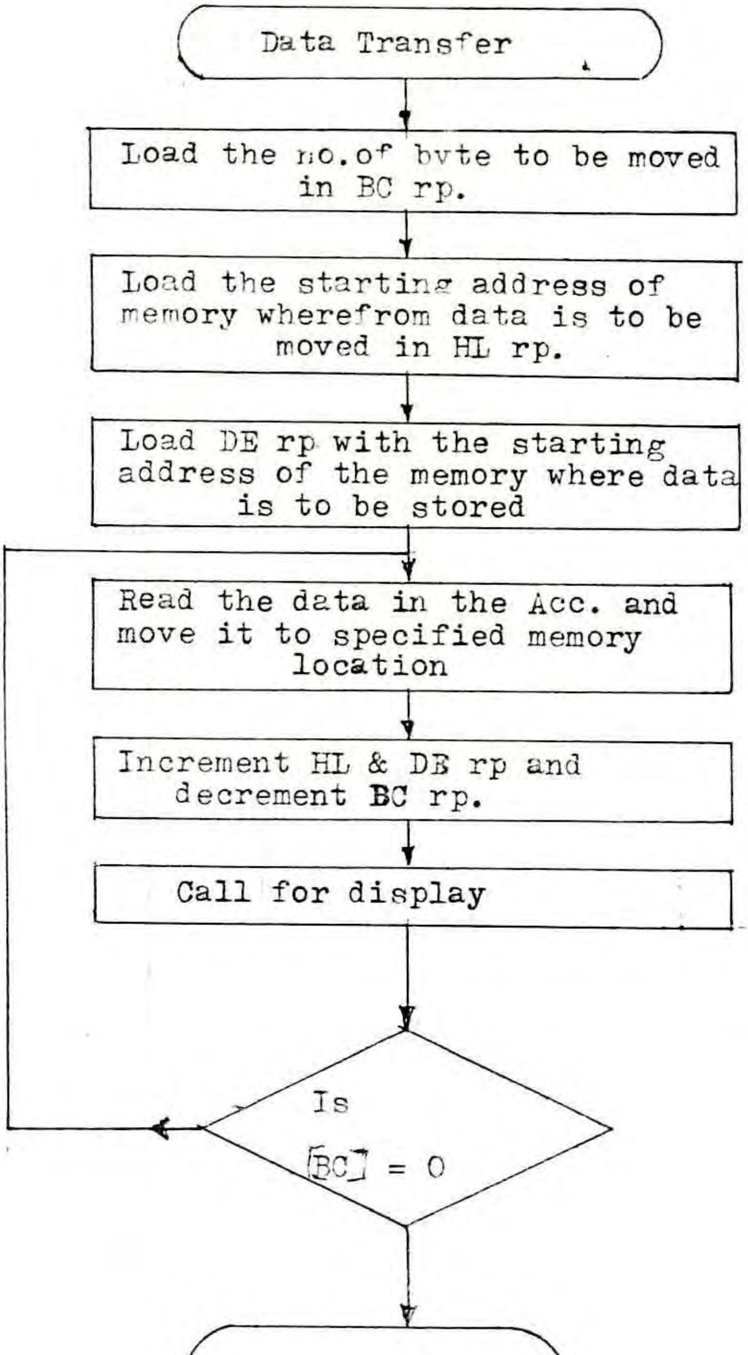
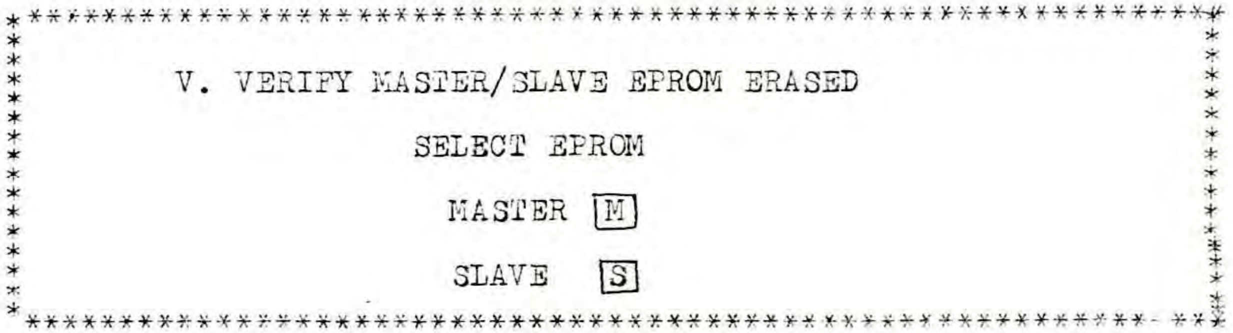

Data Transfer
Load the no.of byte to be moved in BC rp.
Load the starting address of memory wherefrom data is to be moved in HL rp.
Load DE rp with the starting address of the memory where data is to be stored
Read the data in the Acc. and move it to specified memory location
Increment HL & DE rp and decrement BC rp.
Call for display
Is [BC] = 0
STOP

Fig. 7.8(b) Flow Chart for Transfer of Data from Memory to Memory.

7.2.5 VERIFY Subprogram

It is used to verify whether any segment of an EPROM at Master or Slave location is erased. Here data is to be fetched from the EPROM and read and displayed on the screen to see whether EPROM is erased. After the main menu (Selection menu) is displayed on the screen, if the key V is pressed, verify Menu will be displayed as shown in Fig. 7.9(a).



```
V. VERIFY MASTER/SLAVE EPROM ERASED

           SELECT EPROM

             MASTER [M]

             SLAVE  [S]
```

Fig. 7.9(a) Format of Verify Menu

After the verify menu is displayed, if M is pressed the EPROM at Master location will be verified and if S is pressed the EPROM at slave location will be verified.

As in the case of READ, once the source EPROM is selected the microcomputer waits for the start and end address of the block of EPROM to be verified.

The figures that will be displayed after pressing the $\boxed{M}$ key is shown in Fig. 7.10(a) and (b). Each of the addresses will have to be entered by the user from the key board interactively. Only four figures can be entered and may be corrected by shifting the cursor. Next line will be displayed only after the previous

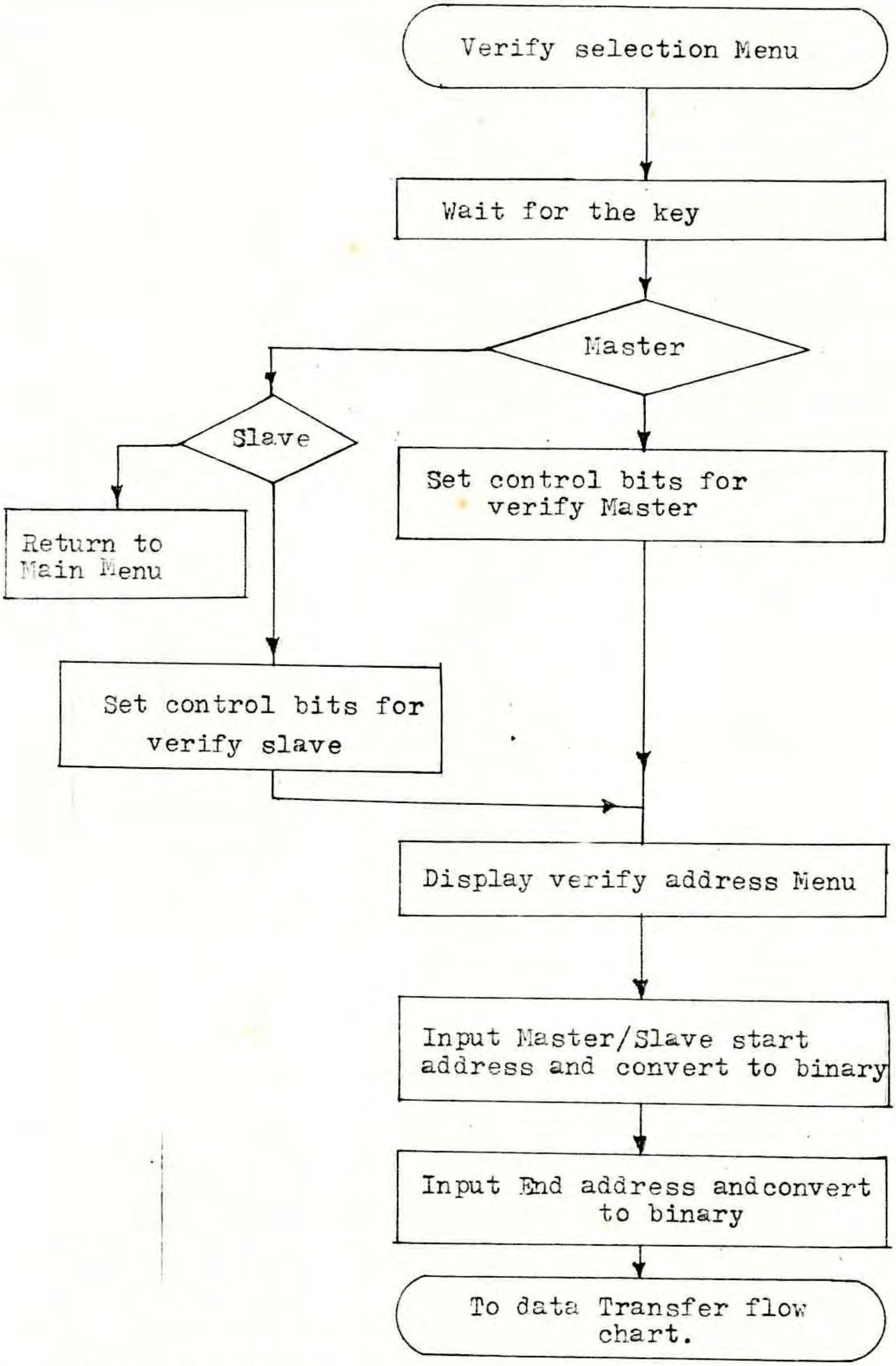

Verify selection Menu
Wait for the key
Master
Slave
Set control bits for verify Master
Return to Main Menu
Set control bits for verify slave
Display verify address Menu
Input Master/Slave start address and convert to binary
Input End address andconvert to binary
To data Transfer flow chart.

Fig.7.11(a) Flow chart for displaying menu and acquisition of addresses for VERIFY.

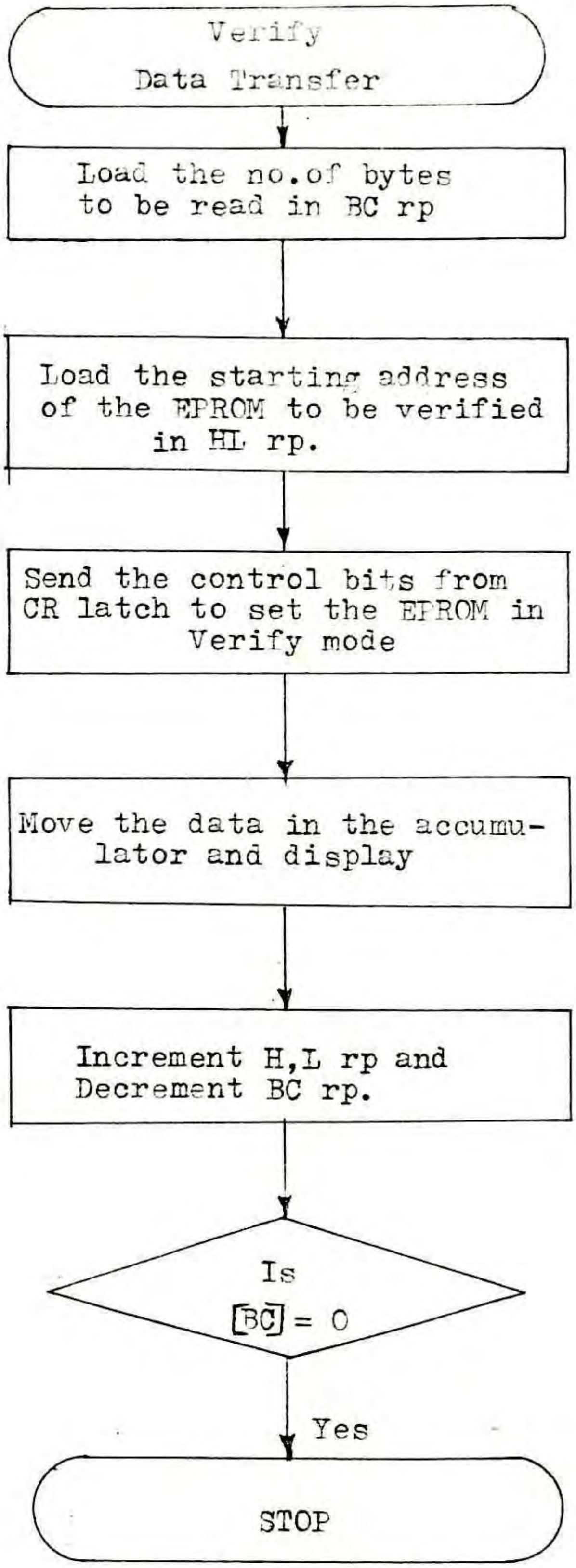

Verify
Data Transfer
Load the no.of bytes to be read in BC rp
Load the starting address of the EPROM to be verified in HL rp.
Send the control bits from CR latch to set the EPROM in Verify mode
Move the data in the accumulator and display
Increment H,L rp and Decrement BC rp.
Is [BC] = 0
Yes
STOP

Fig. 7.11(b) Flow Chart for Transferring data for VERIFY.

```
**************************************************************
*                                                            *
*                                                            *
*                 MASTER START ADDRESS ■                     *
*                                                            *
*                                                            *
*                                                            *
*                                                            *
*                                                            *
*                                                            *
*                                                            *
*                                                            *
**************************************************************
```

(a)

```
**************************************************************
*                                                            *
*                                                            *
*                 MASTER START ADDRESS  ppqq                 *
*                                                            *
*                 MASTER END ADDRESS  ■                      *
*                                                            *
*                                                            *
*                                                            *
*                                                            *
*                                                            *
**************************************************************
```

(b)

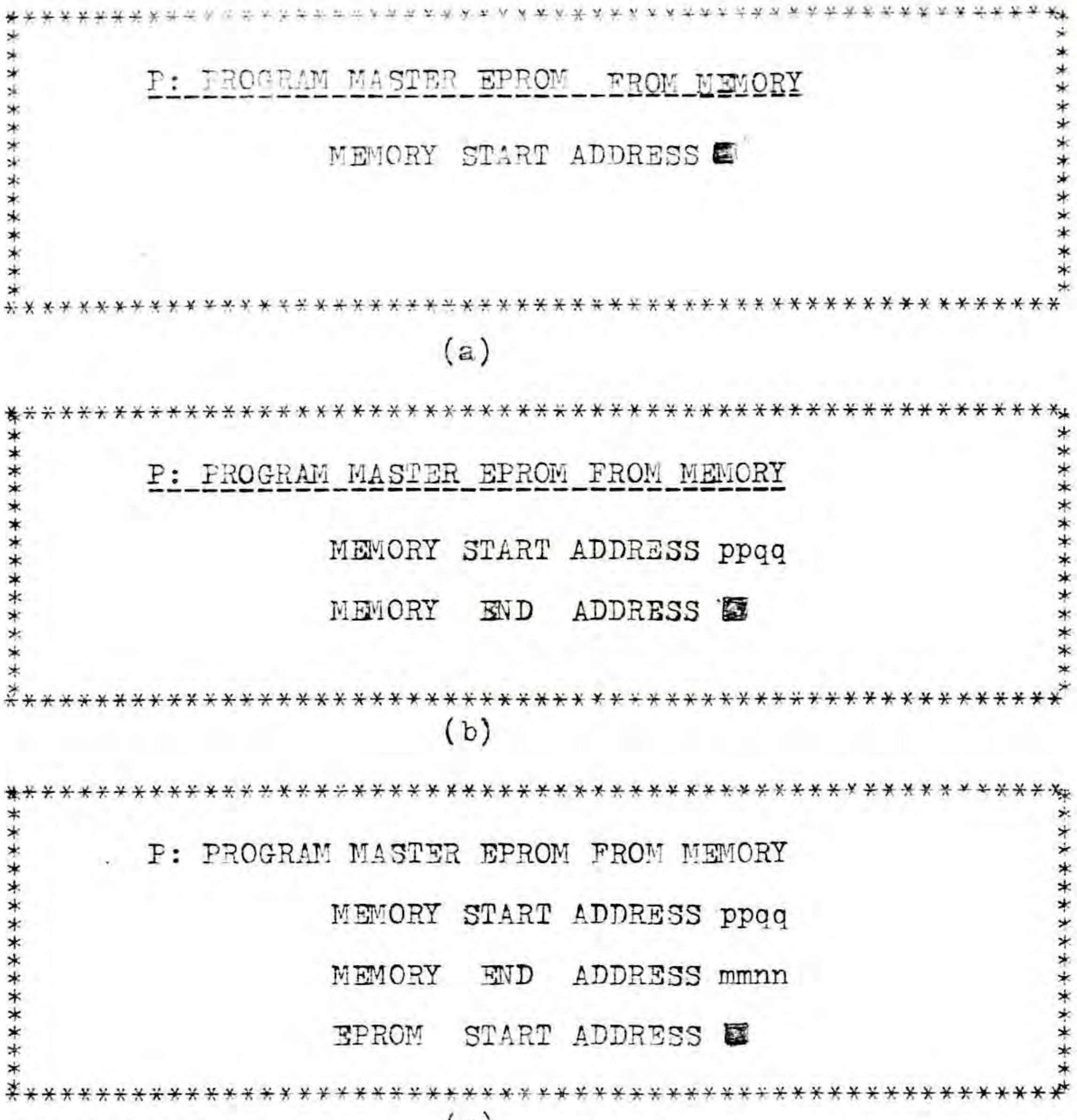
Fig. 7.10 Interactive Displays of Verify Menu

line is entered with carriage return at the end.

The detailed flow chart for displaying and address entering is shown in Fig.7.11(a) and that of data transfer is shown in Fig.7.11(b).

7.2.6 PROGRAM Subprogram

This program is used for programming any segment of the EPROM at Master location from the data stored at a certain memory location. After the main menu is displayed if key P is pressed, program menu as shown in Fig.7.12(a),(b) and (c) will be displayed on the screen one after the other.



```
P: PROGRAM MASTER EPROM  FROM MEMORY

           MEMORY START ADDRESS ■
```

(a)

```
P: PROGRAM MASTER EPROM FROM MEMORY

           MEMORY START ADDRESS ppqq

           MEMORY  END  ADDRESS ■
```

(b)

```
P: PROGRAM MASTER EPROM FROM MEMORY

           MEMORY START ADDRESS ppqq

           MEMORY  END  ADDRESS mmnn

           EPROM  START ADDRESS ■
```

(c)

Fig. 7.12 Interactive Displays of Program Menu

As in the previous cases, here also the start address will have to be entered manually and only after the start address has been entered with carriage return at the end, the next line will appear with the cursor at the position shown. The typed data can be corrected by moving the cursor.

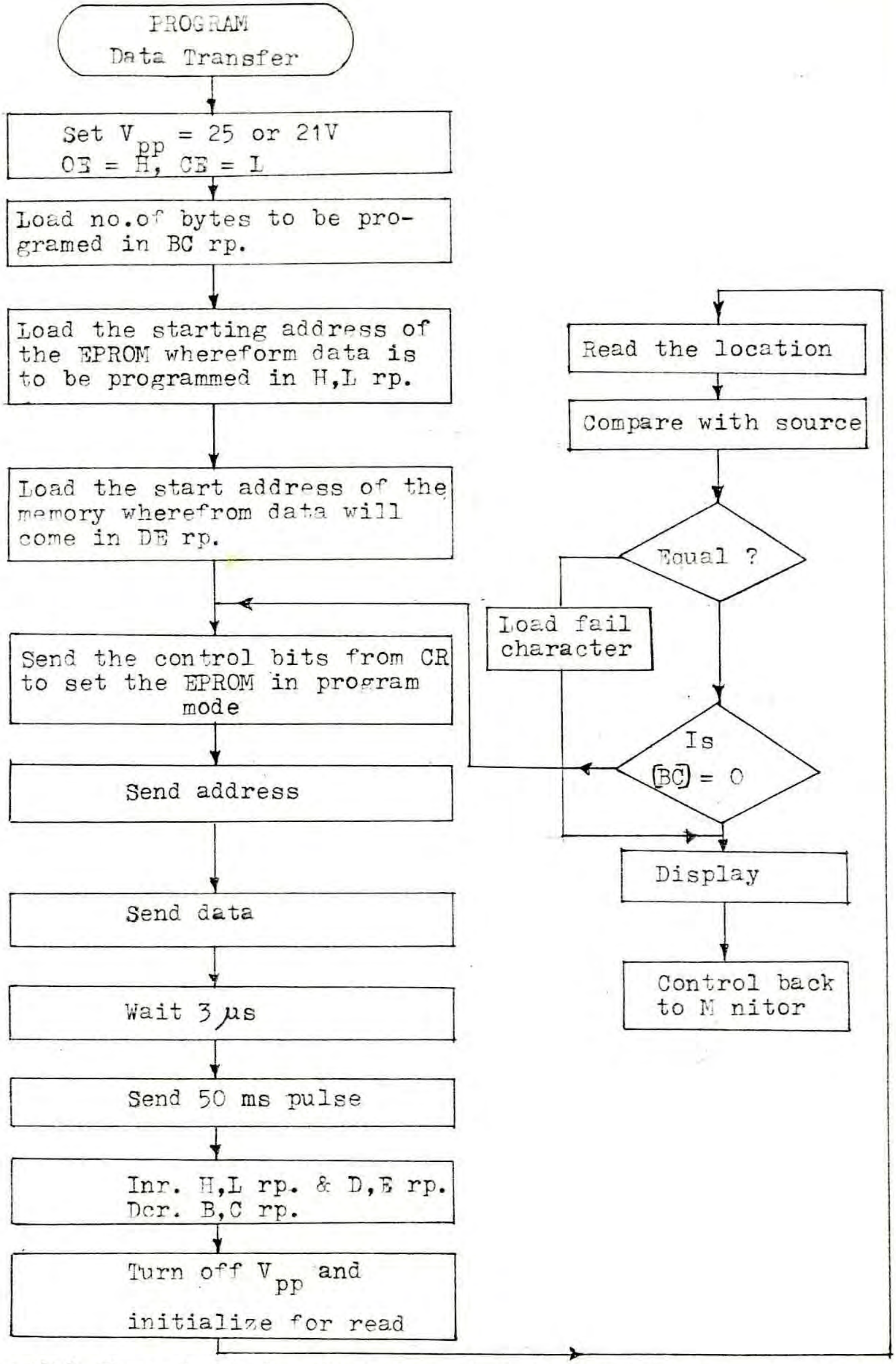


Fig.13(b) Flow chart for transfer of data for program operation.

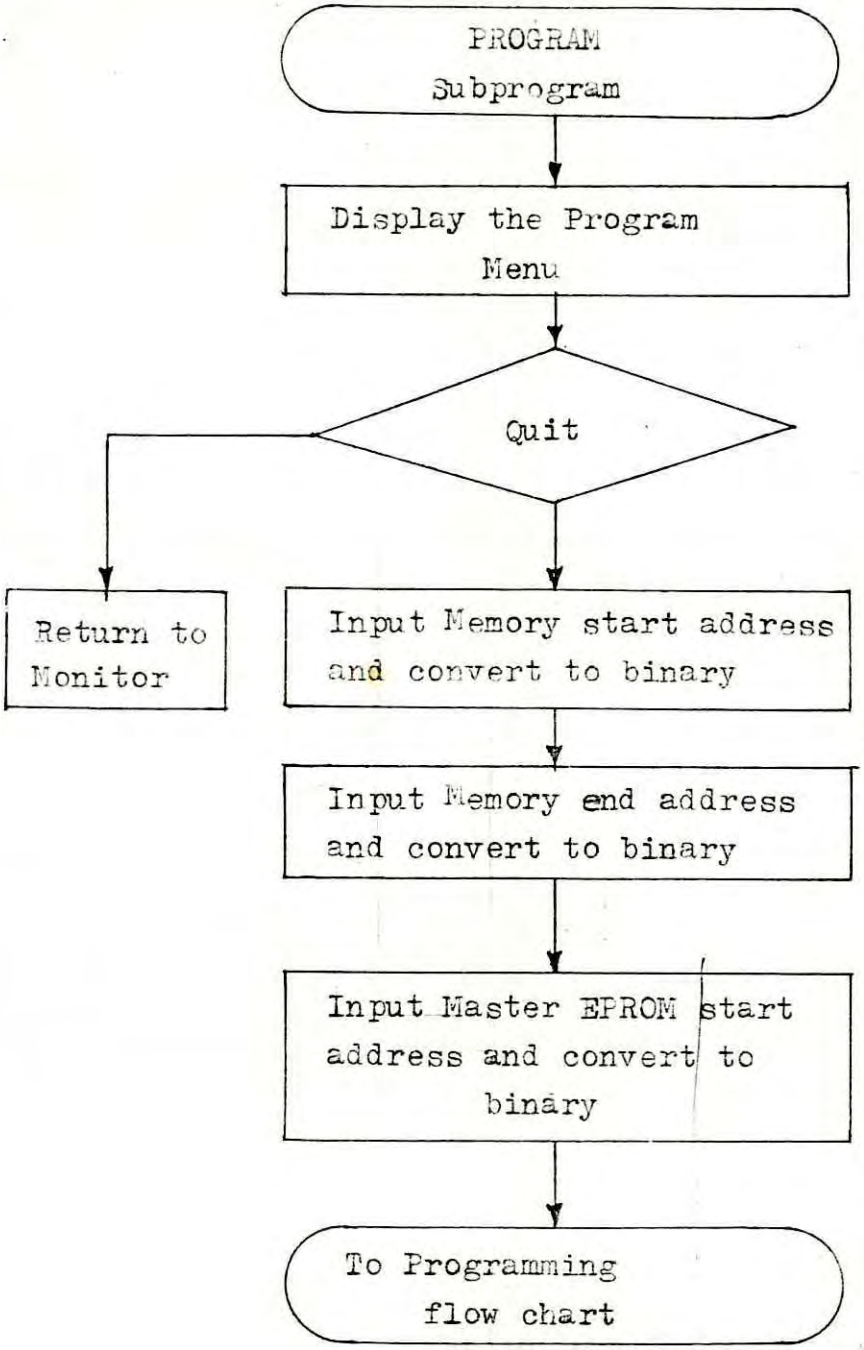

PROGRAM Subprogram
Display the Program Menu
Quit
Return to Monitor
Input Memory start address and convert to binary
Input Memory end address and convert to binary
Input Master EPROM start address and convert to binary
To Programming flow chart

Fig.7.13(a) Flow Chart for Display of Program Menu and Acquisition Start and End Address.

As in the other cases, here also the flow chart can be divided in two sections, (i) Displaying the Program Menu and entering the start and end addresses of the memory where the data to be programmed is stored and the start address of the EPROM wherefrom programming is to begin and (ii) programming the EPROM.

The detailed flow charts are shown in Fig.7.13(a) and(b). Programming waveforms and programming characteristics of Intel EPROM are given in APPENDIX-B. To program the device, V_{pp} is first raised to 25V or 21V as the case may be. The desired address is applied to the device. and $\overline{\text{OE}}$ is raised to a high state. The data byte is then applied to the device. At least 2μs after the data is stable, a 50 ms program pulse is applied. Data must be held low for at least 2μs after the program pulse goes low. After a location is programmed, $\overline{\text{OE}}$ can be taken low and location read to see if the programming was successful. It $\overline{\text{OE}}$ is left high, the data and address for the next location can be sent and then another 50 ms programming pulse applied to the $\overline{\text{CE}}$/PGM pin. The address will be incremented after each location is programmed.

7.2.7 COPY Subprogram

COPY subprogram is used to copy a block of data in an EPROM in Master location into an EPROM at Slave location. Data is first fetched from the EPROM in Master location by activating its output enable pin and then sending the data to the EPROM at Slave location in programming mode and then verifying whether the EPROM at slave location is correctly programmed. After the main menu is displayed if key C is pressed, Copy Menu as shown in Fig.7.15(a),(b) and (c) will be displayed on the screen one after another.

```
     C: COPY INTO SLAVE FROM MASTER
     ------------------------------

           MASTER START ADDRESS ■
```

(a)

```
     C: COPY INTO SLAVE FROM MASTER
     ------------------------------

           MASTER START ADDRESS   ppqq

           MASTER  END  ADDRESS   ■
```

(b)

```
     C: COPY INTO SLAVE FROM MASTER
     ------------------------------

           MASTER START ADDRESS   ppqq

           MASTER  END  ADDRESS   mmnn

           SLAVE START  ADDRESS   ■
```

(c)

Fig.7.15 Interactive Display of COPY Menu.

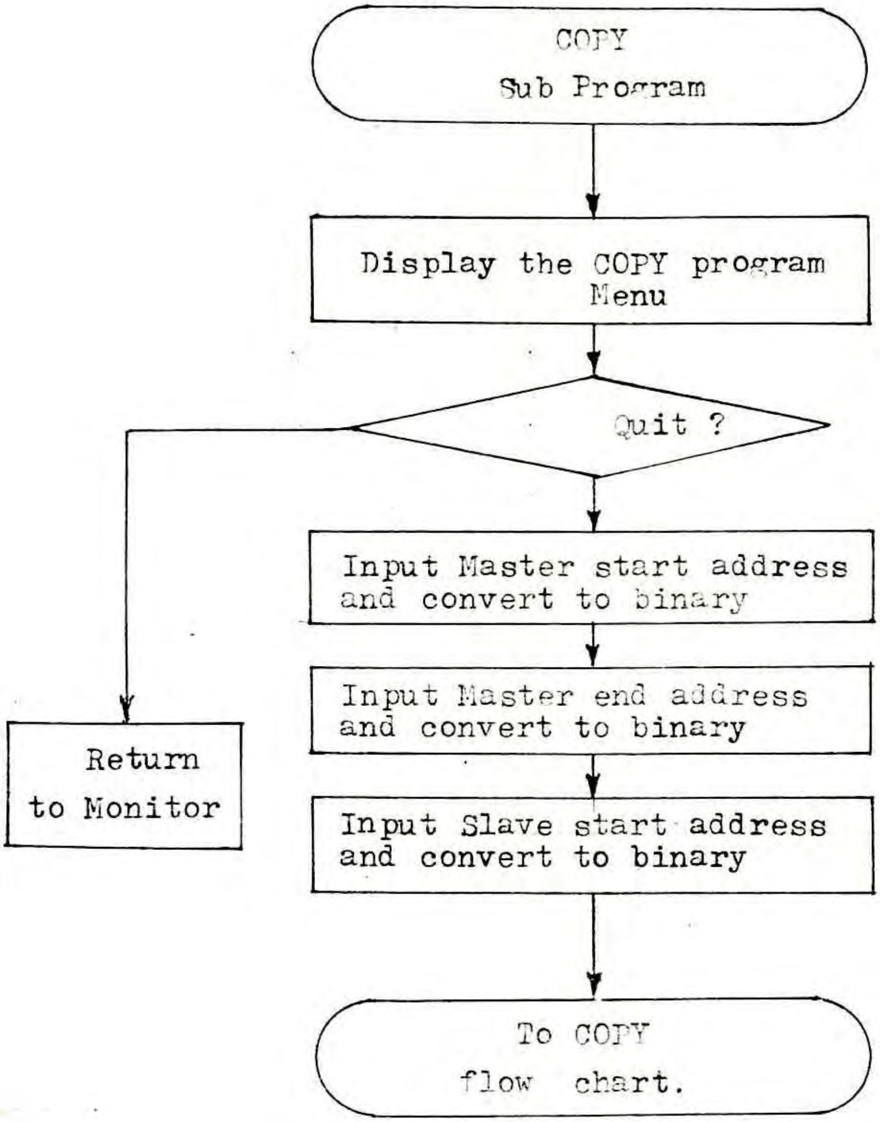

COPY
Sub Program
Display the COPY program Menu
Quit ?
Return to Monitor
Input Master start address and convert to binary
Input Master end address and convert to binary
Input Slave start address and convert to binary
To COPY flow chart.

Fig.7.16 (a) Flow chart for displaying and acquisition of address in COPY mode.

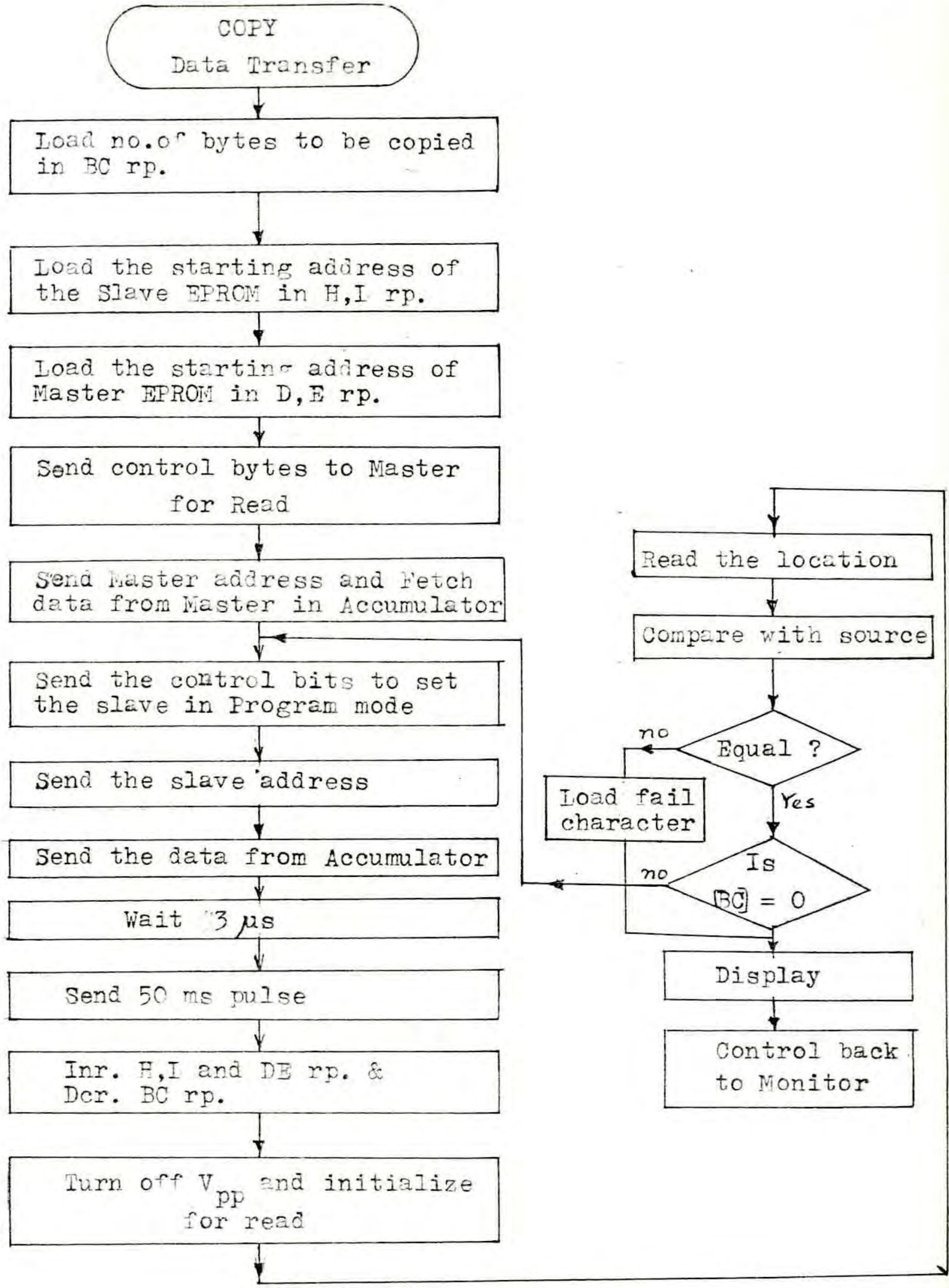

COPY
Data Transfer
Load no.of bytes to be copied in BC rp.
Load the starting address of the Slave EPROM in H,L rp.
Load the starting address of Master EPROM in D,E rp.
Send control bytes to Master for Read
Send Master address and Fetch data from Master in Accumulator
Send the control bits to set the slave in Program mode
Send the slave address
Send the data from Accumulator
Wait 3 µs
Send 50 ms pulse
Inr. H,L and DE rp. & Dcr. BC rp.
Turn off V_{pp} and initialize for read
Read the location
Compare with source
Equal ?
no
Yes
Load fail character
Is [BC] = 0
no
Display
Control back to Monitor

Fig.7.16(b) Flow Chart for Transfer of Data for COPY.program.

The addresses are to be entered manually from the key board interactively. Here also the next line will appear on the screen only after the previous address is entered with carriage return at the end. The typed address can be corrected using the cursor movement.

The flow chart is shown in Fig.7.15(a) and (b). The first flow chart is for entering the addresses and displaying the Menu and the second flow chart is for fetching the content of Master EPROM and then programming into Slave EPROM and then verifying it.

7.2.8 PRINT Subprogram

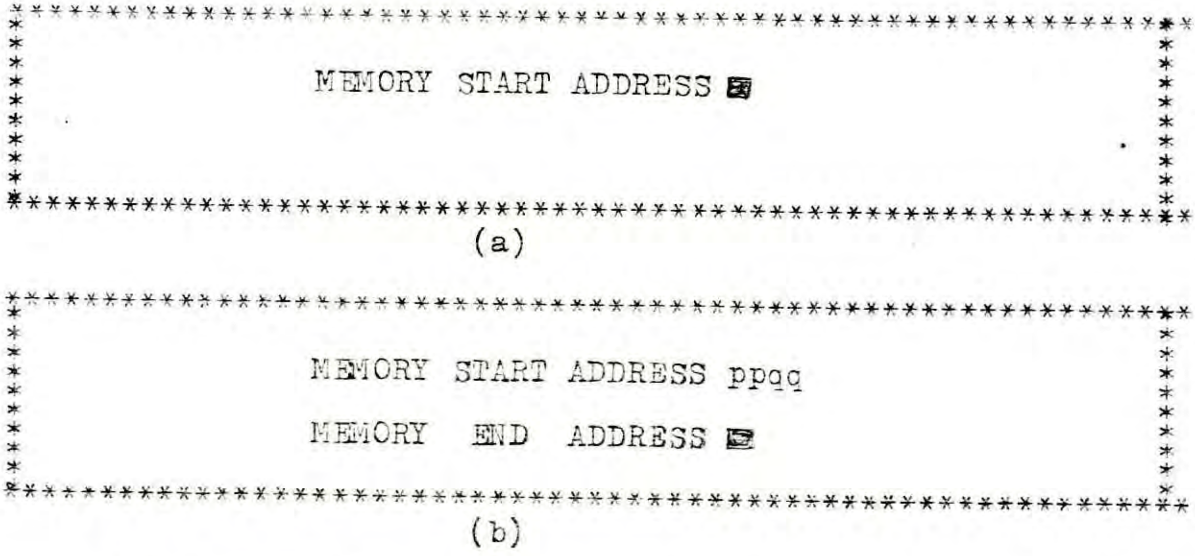
This program is used for printing the data from the main memory or from EPROM at Master or Slave location. After the main menu is displayed on the screen, if the key T is pressed, the system will enter the PRINT mode and the PRINT menu as shown in Fig.7.17 will be displayed.

```
***********************************************************
*                                                         *
*     T: PRINT DATA FROM MEMORY, MASTER, SLAVE            *
*                                                         *
*                 FROM MEMORY  [M]                        *
*                                                         *
*                 FROM MASTER  [T]                        *
*                                                         *
*                 FROM SLAVE   [S]                        *
*                                                         *
***********************************************************
```

Fig.7.17 Format of Print Menu

After PRINT Menu is displayed if $\boxed{M}$ is pressed the data from memory will be printed, if $\boxed{T}$ is pressed, data from Master EPROM will be printed and if $\boxed{S}$ is pressed data from Slave EPROM will be printed.

Once the source is selected the microcomputer waits for the start and end addresses of the block of data to be printed. Each of the addresses will be entered interactively. Interaction of the user and the computer for transferring data from Memory is shown in Fig. 7.18(a) and (b). The data is to be entered manually by the user. Next line appears only after the previous line is entered.



```
MEMORY START ADDRESS ■
```

(a)

```
MEMORY START ADDRESS ppqq
MEMORY  END  ADDRESS ■
```

(b)

Fig.7.18 The Interactive Displays of Memory for PRINT.

The flow chart for the PRINT program can also be divided into two blocks.

1. Dispalying of Menu and acquisition of address
2. Transfering of data to printer.

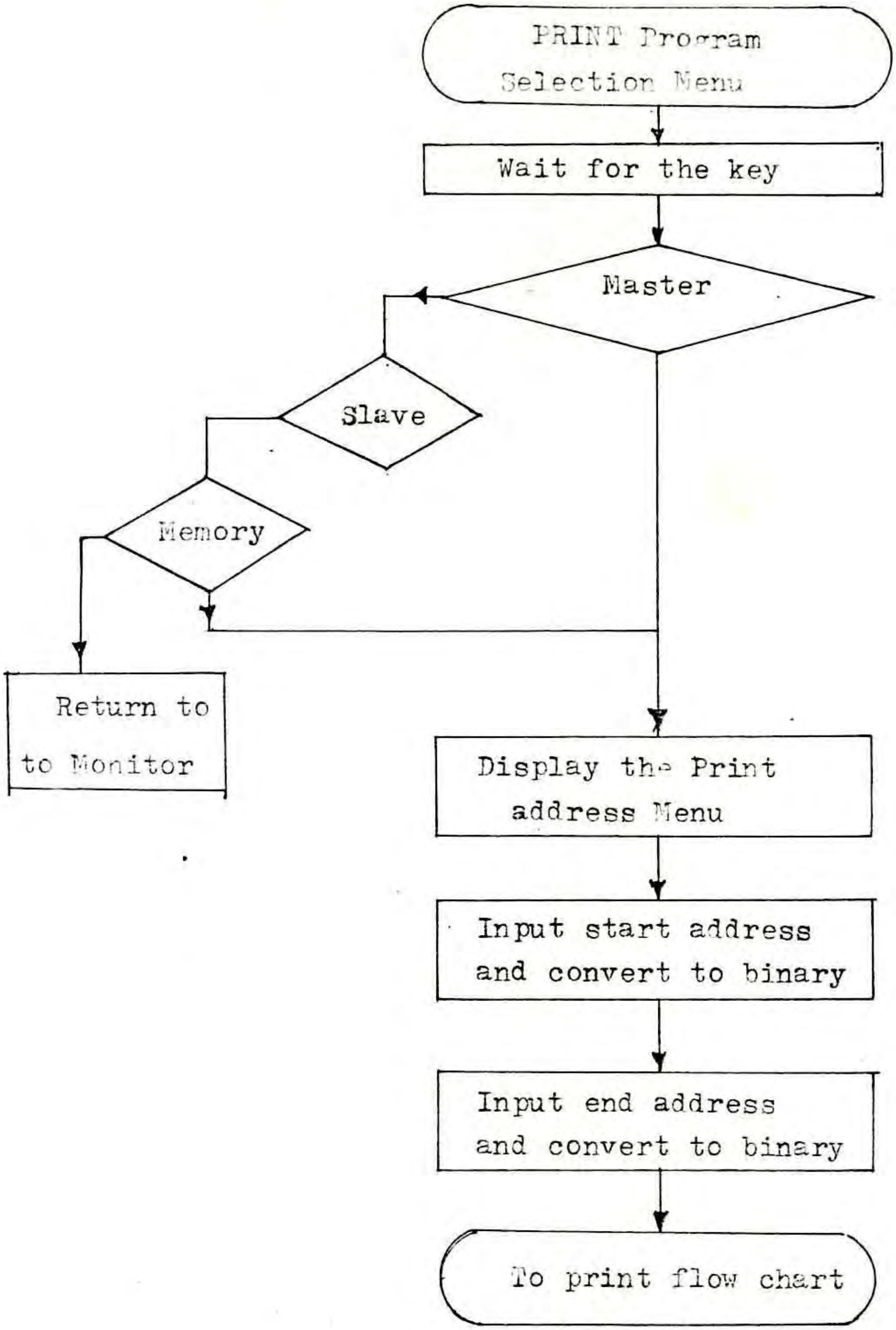


Fig.7.18(a) Flow Chart for Displaying of Print Menu and Acquisition of Address.

The printer is a slow device. So it cannot print data as fast as data can be supplied from memory. To overcome this difficulty the printer has two signals (i) busy (ii) acknowledge . When one byte of data enters the printer, it sends the busy signal indicating that it cannot accept any more data and when printer accepts one byte of data it sends acknowledge signal that it has received data. These signals are examined by the control unit and sends data only when the printer is not busy.

CHAPTER - VIII

DISCUSSIONS AND CONCLUSIONS

The EPROM programmer designed by us is microprocessor controlled Three buffers, six latches and decoder, NAND gates, inverters were used in the design.A single sided PCB was designed and constructed with in-house facilities. Since the board was single sided as such there was many cross connections, which is very cumbersome. This is due to the unavailability of a PCB laboratory. The design would be easier if integrated chip like 8155 was available. Due to non-availability of the LSI integrated chip, we had to use the simpler type IC's.

This programmer can be used to Program, Verify, Read, Display, Copy and Print. With its 40 output terminals, with proper connections and software any type of EPROM can be programmed.

The designed Programmer can be used for programming Intel 2716, 2732, 2764 and 27128 EPROMs and the EPROMs within Intel 8748, 8049 chips. Since the programmer is software controlled, as such it can be used to program any type of EPROM available, just with a little change in the control program of the EPROM programmer. This programmer will also save the buying of costly manufactured programmer available in the foreign market and it will prove to be helpful in research works on the microcomputer based design.

REFERENCE

1. Steve Ciarcia, "Build your own Z80 Computer", BYTE Book/A McGraw Hill Publication/Peterborough, New Hampshire.

2. Roger L. Tokheim, "Microprocessor Fundamentals," Schaum's Outline series in computers, McGraw Hill Book Company, New York, 1980.

3. Millman J., Halkias C.C., "Integrated Electronics: Analog and digital circuits and systems", McGraw Hill Book Company, New York.

4. Givone, D.D., Roesser, R.P., "Microprocessors/Microcomputers: An Introduction," McGraw Hill Book Company, New York.

5. "Intel Data catalog", 1983.

6. Hall, Douglas V., "Microprocessors and digital systems" McGraw Hill Book Company, New York.

7. "Pro-Log Corporation". M 980, Electronics, July 31,1980.

8. "G.P. Industrial Electronics", EP4000, Wireless World, April 1983.

9. H.Muller, "Simple EPROM Programmer" International Centre for theoretical physics, Technology and Application in physics, 18 April - 13 May, 1983.

10. "AIM-65 User's Guide", Rockwell International, 1978.

11. H.S.Lynes,"EPROM Programmer", Wireless World, April, May, 1982.

APPENDIX - A

Soft-ware programme

```
*************************************************************
*     THE DISPLAY MONITOR HAS THE ADDRESS STARTING FROM
*     E800H TO EFFFH.IF ANY DATA IS MOVED TO A PARTICULAR
*     LOCATION WITHIN THIS ADDRESS RANGE.THE DATA WILL BE
*     DISPLAYED AT THAT LOCATION.
*     THIS IS THE MAIN PROGRAM WHICH WILL DISPLAY THE MAIN
*     MENU ON THE MONITOR FOR THE SELECTION OF SUBPROGRAM.
*************************************************************

   0100 START LXI D,0E8CH      PUT 44 STAR(*) FROM
   0110  CALL STAR             THE LOCATION  0E8C OF MONITOR
   0120  LXI D,0EDCAH          PUT 44 * AT 0EDC OF MONITOR.
   0130  CALL STAR
   0140  LXI D,0E951H
   0150  LXI H,SEL             WRITE SELECTION MENU
   0160  CALL DSPT             FROM E951H LOCATION.
   0170  LXI D,0E990H
   0180  CALL DASH             UNDERLINE BY DASH.
   0190  LXI D,0EA10H          WRITE
   0200  LXI H, COP            COPY
   0210  CALL DSPT
   0220  MVI A,3A
   0230  STA 0EA18H
   0240  LXI D,0EA1AH          MASTER
   0250  CALL DMST
   0260  CALL TO               TO
   0270  SHLD 0EA23H
   0280  LXI D,0EA27H
   0290  CALL DSLV             SLAVE
   0300  MVI A,3A
   0310  STA 0EA2C
   0320  MVI A,03               C
   0330  STA 0EA2EH          --------------
   0340  LXI D,0EA9D           WRITE
   0350  CALL DPRG             PROGRAM
   0360  MVI A,3AH
   0370  STA 0EA98H
   0380  LXI D,EA9AH              .
   0390  CALL DMEM             MEMORY
   0400  CALL TO
   0410  SHLD DEAA3H           TO
   0420  LXI D,0EAA7H
   0430  CALL DMST             MASTER
   0440  MVI A,3A
   0450  STA 0EAACH
   0455  MVI A,10H              P
   0460  STA 0EAAEH          --------------
   0470  LXI D,EB10H           WRITE
   0490  CALL DVER             VERIFY
   0500  MVI A,3AH
   0510  STA 0EB18H
   0520  LXI D,EB1AH
   0530  CALL DMS              MASTER,SLAVE
   0540  LXI D,0EB27H
   0550  CALL DERS             ERASED
   0560  MVI A,3A
   0570   STA 0EE2CH
   0580  MVI A,16H              V
   0590  STA 0EB2EH          --------------
   0600  LXI D,EB90H           WRITE
   0610  CALL DRD              READ
   0620  MVI A,3AH
   0630  STA 0EB98H
   0640  LXI D,0EB9AH
   0650  CALL DMS              MASTER,SLAVE
   0660  MVI A,3AH
   0670  STA 0EBACH
   0680  MVI A,12H              H
   0690  STA 0EBAEH          --------------
```

```
0700  LXI D,0EC10H            WRITE
0710  CALL DDIS               DISPLAY
0720  MVI A,3AH
0730  STA 0EC1,H
0740  LXI D,0EC1AH
0750  CALL DMSM               MASTER,SLAVE,MEMORY
0760  MVI A,3AH
0770  STA 0EC2CH
0780  MVI A,04                  D
0790  STA 0EC2EH            ---------------
0800  LXI D,0EC90H            WRITE
0810  CALL DMOV               MOVE
0820  MVI A,3A
0830  STA 0EC98H
0840  LXI D,0EC9AH
0850  CALL DMEM               MEMORY
0760  CALL TO
0870  SHLD 0ECA3H             TO
0880  LXI D,0ECA7H
0890  CALL DMEM               MEMORY
0900  MVI A,3A
0910  STA 0ECACH
0920  MVI A,0DH                 M
0930  STA 0ECAEH            ---------------
0940  LXI D,0EC10H            WRITE
0950  CALL DPRT               PRINT
0960  MVI A,3AH
0970  STA 0EC18H
0980  LXI D,0EC1AH
0990  CALL DMSM               MASTER,SLAVE,MEMORY
1000  MVI A,3A
1010  STA 0EC2CH
1020  MVI A,14                  T
1030  STA 0EC2EH            ---------------
*    THIS IS THE PORTION OF MAIN MENU WHICH WILL
*    DETECT WHETHER ANY KEY IS PRESSED.IF ANY KEY
*    IS PRESSED THE PROGRAM WILL JUMP TO THE SUBPROGRAM
*    CORRESPONDING TO THE KEY PRESSED.
1040 MMKPRS CALL 0F6H         MAIN MENU KEY PRESSED
1050  CPI 43H                 IS KEY 'C'
1060  JZ COPY                 THEN JUMP TO COPY SUBPRG.
1070  CPI 50H                 IS IT 'P'
1080  JZ PROGRAM              JUMP TO PROGRAM SUBPRG.
1090  CPI 56H                 IS IT 'V'
1100  JZ VERIFY               JUMP TO VERIFY SUBPRG.
1110  CPI 52H                 IS IT 'R'
1120  JZ READ                 JUMP TO READ SUBPRG.
1130  CPI 44H                 IS IT 'D'
1140  JZ DISPLAY              JUMP TO DISPLAY SUBPRG.
1150  CPI 54H                 IS IT 'T'
1130  JZ PRINT                JUMP TO PRINT SUBPRG.
1170  CPI 4DH                 IS IT 'M'
1180  JZ MOVE                 JUMP TO MOVE SUBPRG.
1190  JMP MMKPRS
*
*    READ SUBPROGRAM STARTS FROM HERE.
*    IT DISPLAYS THE READ SUB MENU.
*
1200 READ CALL CLEAR          CLEAR MAIN MENU DISPLAY.
1210  MVI A,52H               WRITE
1220  STA 0E951H
1230  MVI A,3AH               R
1240  STA  0E953H
1250  LXI D,0E955H
1260  CALL DRD                READ
1270  LXI D,0E95AH
1280  CALL DMS                MASTER,SLAVE
1290  LXI D,0E957H
1300  CALL DEPR               EPROM
1310  ACLL TO
1320  SHLD 0E95DH             TO
1330  LXI D,0E961H
1340  CALL DMEM               MEMORY
1350  CALL SEMSD              MASTER  M  SLAVE  S
```

```
1360 RDMENU CALL 0F6H       CHECK FOR READ KEY PRESSED.
1370   CPI 4DH             IS KEY M THEN
1380   JZ RMST             JUMP TO RMST
1390   CPI 54H             IS IT S THEN
1400   JZ RSLV             JUMP TO RSLV
1410   CPI 51              IF KEY IS Q THEN RETURN
1420   JZ START            TO MAIN MENU.OTHERWISE
1430   JMP RDMENU          REPEAT THE LOOP.
*     ACQUISITION OF START & END ADDRESSES FOR READ
*     SUBPRG. AND READING OF DATA FROM MASTER START HERE
1460 RMST LXI D,0EA10H
1470   CALL SPACE          CLEAR 'SELECT EPROM'
1480   CALL ESEADD         ENTER START& END ADDRESS.
1490   CALL LMEM3          WRITE'MEMORY'
1500   CALL ETSADD         ENTER MEMORY START ADDRESS.
1510   CALL SUBTR          LOAD RESULT OF  SUBTRACTION
1520   MVI A,0A0H              IN B,C R.P.
1530   OUT 0BCH            SET MASTER IN READ MODE.
1540   MVI A,04
1550   OUT 0BDH            SET THE ADH, ADL & CR IN OUTPUT MODE.
1560   CALL DERP           LOAD START ADDR.OF MEM.IN D,E
1570   CALL HLRP           RP.&THAT OF EPROM IN H,L RP.
1580 MRLOP MOV A,H
1590   OUT 0B9H            SEND HIGH BYTE ADDRESS
1600   MOV A,L
1610   OUT 0BAH            SEND LOW BYTE ADDRESS.
1620   IN 0B8H             READ DATA IN ACCUMULATOR.
1630   STAX D
1640   INX H
1650   INX D
1660   MOV A,B
1670   ORA C               IF (BC) IS NOT 0 REPEAT
1680   JNZ MRLOP           MASTER READ LOOP(MRLOP).
1690   HLT
1700 RSLV LXI D,0EA10H      SLAVE READING STARTS HERE.
1710   CALL SPACE
1720   CALL LSLV1
1730   CALL ESEADD
1740   CALL LMEM3
1750   CALL ETSADD
1760   CALL SUBTR
1762   MVI A,04
1765   OUT 0BDH            SET THE ADH,ADL,CR IN OUTPUT MODE.
1770   MVI A,14H           SET SLAVE IN READ MODE
1780   OUT 0BCH
1790   CALL DERP           SET THE STARTING ADDRESS
1800   CALL HLRP           OF SLAVE & MEMORY
1810 SRLOP MOV A,H
1820   OUT 0B9H            SEND THE HIGH BYTE ADDRESS
1830   MOV A,L
1840   OUT 0BAH            SEND LOW BYTE ADDRESS
1850   IN 0B8H             INPUT THE DATA
1860   STAX D              STORE DATA IN MEMORY
1870   INX H
1880   INX D
1890   MOV A,B
1900   ORA C               IF (BC) IS NOT 0 REPEAT
1910   JNZ SRLOP           SLAVE READ LOOP(SRLOP)
1920   HLT
*
*     DISPLAY SUBPROGRAM STARTS HERE.IT DISPLAYS CONTENT
*      OF EPROM AT SLAVE OR MASTER LOCATION ON MONITOR.
*
1930 DISPLAY CALL CLEAR       CLEAR MAIN MENU
1940   MVI A,04H           WRITE
1950   STA 0E951H
1960   MVI A,3AH           D
1970   STA 0E953H
1980   LXI D,0E955H
1990   CALL DDIS           DISPLAY
2000   LXI D,0E95DH
2010   CALL DMSM           MASTER,SLAVE,MEMORY
2020   CALL SEMSD          MASTER  M  SLAVE  S
2060   CALL LMEM3          WRITE'MEMORY'
2340   MVI A,4DH
2350   STA 0EB9CH           M
```

```
2360 DMENU CALL 0F6H          CHECK FOR DISPLAY KEY PRESSED
2370  CPI 4DH                 IS IT M THEN GO TO
2380  JZ DSMST                DSMST
2390  CPI 53                  IS IT S THEN GO TO
2400  JZ DSSLV                DSSLV
2410  CPI 52H                 IS IT R THEN GO TO
2420  JZ DSMEM                DSMEM
2430  CPI 51                  IS IT Q THEN RETURN TO
2440  JZ START                MAIN MENU OTHERWISE LOOP BACK
2450  JMP DMENU               TO SEARCH FOR KEY PRESSED
2460 DSMST LXI D,0EA10H       CLEAR'SELECT EPROM'
2470  CALL SPACE
2480  CALL ESEADD             ENTER ADDRESS
2490  CALL SUBTR              RESLT OF SUBTRACTION IN BC RP.
2500  MVI A,0A0H              SET MASTER IN READ MODE.
2510  OUT 0BCH
2520  MVI A,04                SET ADH,ADL & CR IN OUT MODE.
2530  OUT 0BDH
2540  CALL HLRP               START ADDR. IN HL
2550  CALL DSLOP              DISPLAY THE CONTENT
2555  HLT
2560 DSSLV LXI 0EA10H
2570  CALL SPACE
2580  CALL LSLV1
2590  CALL ESEADD             ENTER ADDRESS
2610  CALL SUBTR
2620  MVI A,04
2630  OUT 0BDH
2640  MVI A,14H
2645  OUT 0BCH                SET SLAVE IN OUTPUT MODE
2650  CALL HLRP
2660  CALL DSLOP            DISPLAY
2665  HLT
2670 DSMEM LXI 0EA10H
2680  CALL SPACE
2690  CALL LMEM1
2700  CALL ESEADD             ENTER START,END ADDRESSES
2710  CALL SUBTR              RESULT OF SUBTRACTION IN B,C RP
2720  LHLD STRT1          .   LOAD STARTING ADDRESS IN H,L
2730 DSLOPM MOV A,M
2740  PUSH B
2750  MOV B,A
2760  CALL 103H               DISPLAY CONTENT
2770  POP B
2780  INX H
2790  DCX B
2800  MOV A,B
2710  ORA C                   IS (BC) IS NOT 0 THEN
2820  JNZ DSLOPM              LOOP BACK
2830  HLT
*
*    'PROGRAM' SUBPROGRAM STARTS FROM HERE. IT IS USED TO
*    PROGRAM AN EPROM AT MASTER OR SLAVE LOCATION
*    FROM MAIN MEMORY.
*
2890 PROGRAM CALL CLEAR          CLEAR MAIN MENU
2900  MVI A,16H               WRITE
2910  STA 0E951H              P
2920  MVI A,3AH
2930  STA 0E953H
2940  LXI D,0E955H
2950  CALL DPRG               PROGRAM
2960  LXI D,0E95DH
2970  CALL DMST               MASTER
2980  LXI D,0E965H
2990  CALL DEPR               EPROM
2995  LXI D,0E96CH
2996  LXI H,FRM               DISPLAY 'FROM'
2997  CALL DSPT
```

```
 2998  LXI D,0E972H
 2999  LXI H,MEM
 3000  CALL DSPT             DISPLAY'MEMORY'
 3001  CALL LMEM             DISPLAY'MEMORY'AND
 3002  CALL LSAD            'START ADDRESS' ON SAME LINE
 3005  CALL LEAD             DISPLAY'END ADDRESS'IN NEXT LINE
 3010  CALL ESEADD           ENTER START & END ADDRESSES
 3015  CALL LMST2            DISPLAY 'MASTER'AND
 3020  CALL LSAD3            'START ADDRESS'ON THIRD LINE
 3025  CALL ETSADD           ENTER MASTER START ADDRESS
 3040  CALL SUBTR
 3050 PLOP MVI A,DD          SET EPROM IN PROGRAM MODE
 3055  OUT 0BCH
 3060  MVI A,04
 3070  OUT 0BDH              SET ADL,ADH,CR IN OUT MODE
 3080  CALL DERP             LOAD PROGRAM START ADDR.OF EPROM
 3090  CALL HLRP             LOAD MEMORY START ADDR. IN H,L RP
 6100  MOV A,H
 3110  OUT 0B9H              SEND HIGH BYTE ADDRESS
 3120  MOV A,L
 3130  OUT 0BAH              SEND LOW  BYTE ADDRESS
 3135  NOP
 3140  MVI A,D4              SEND PROGRAMMING PULSE
 3150  OUT 0BCH              SEND PROGRAMMING PULSE
 3180  CALL DELAY            WAIT 50 MS.
 3190  INX H
 3200  DCX B
 3220  MVI A,0ADH            SET MASTER IN READ MODE.
 3230  OUT 0BCH
 3240  IN 0B8H               INPUT THE PROGRAMMED BYTE.
 3245  PUSH D
 3250  MOV D,A
 3260  MOV A,M               LOAD ACC. THE BYTE THAT WAS TO BE
 3270  CMP D                 PROGRAMMED & COMPARE WITH PROGRAMMED
 3280  JZ PLOP1              BYTE.IF EQUAL JUMP TO PLOP1
 3290  JNZ PLOP2             OTHERWISE TO PLOP2
 3300 PLOP1 PUSH B
 3310  MOV B,M
 3320  CALL 103H             DISPLAY PROGRAMMED BYTE
 3330  POP B
 3335  POP D
 3340  MOV A,B
 3350  ORA C                 IS (BC)=0
 3360  JNZ PLOP              IF NOT  REPEAT LOOP
 3365  HLT
 3370 PLOP2 PUSH B
 3380  MVI B,FF
 3390  CALL 103H             DISPLAY FF, LOAD FAIL CHARACTER.
 3400  POP B
 3405  POP D
 3410  HLT
**************************************************************
*        SUBROUTINES CALLED BY THE MAIN PROGRAM START FROM
*        HERE.THE FIRST SUBROUTINE IS FOR ENTERING THE ADDRESS
*        MANUALLY FROM KEYBOARD.ONLY FOUR BYTES CAN BE ENTERED
*        AND WRONG ENTRY CAN BE CORRECTED BY CURSOR SHIFTING.
*        NAME OF THIS SUBROUTINE IS 'LDADR'
**************************************************************
 4000 LDADR PUSH H           LOAD ADDRESS S.R.STARTS HERE
 4010  PUSH D
 4020  PUSH B
 4030  MVI E,0
 4040  LXI H,BYTE3
 4050  MVI D,4
 4060 SP MVI M,20H           MAKE FOUR SPACES AFTER CURSOR.
 4070  DCX H
 4080  DCR D
```

```
     4090  JNZ SP
     4100 NUM1 CALL 0F6H
     4110  CPI 0DH                      IS IT CARRIAGE RETURN.
     4120  JZ NUM2
     4130  CPI 8                        IS IT BACK SPACE. IF NOT
     4140  JNZ CHAR                     THEN JUMP TO CHAR
     4150  MOV A,E
     4160  CPI 0
     4170  JZ NUM1
     4180  DCR E
     4190  DCX H
     4200  MVI B.8
     4210  CALL 103H
     4220  JMP NUM1
     4230 CHAR MOV B,A                  SAVE ACCUMULATOR.
     4240  MOV A,E
     4250  CPI 4
     4260  JZ NUM1
     4270  MOV A.B
     4280  CPI 30H                      3. IS ASCII CODE FOR 0
     4290  JM NUM1                      IS CHAR. LESS THAN 30H.
     4300  CPI 3AH                      IS IT LESS THAN 9 THEN
     4310  JM HEXL                      JUMP TO HEXL TO ENTER IT
     4320  CPI 41H
     4330  JM NUM1
     4340  CPI 47H
     4350  JP JUM1
     4360  MOV B,A
     4370  CALL 103H                    DISPLAY THE NUMBER AND THEN
     4380  SUI 37H                      SUTRACT 37H TO MAKE IT HEX NO.
     4385 HEXH MOV M,A                  AND STORE NO.ADDRESSED BY H.L RP
     4390  INR E
     4400  INX H
     4410  JMP NUM1
     4420  MOV B,A
     4430 HEXL CALL 103H
     4440  SUI 30H
     4450  JMP HEXH
     4460 NUM2 LHLD BYTE1
     4470  MOV A,L
     4480  RLC
     4490  RLC
     4500  RLC
     4510  RLC
     4520  ADD H                        FIRST TWO DIGITS ARE STORED AT
     4530  STA MEMR1                    LOCATION 'MEMR1'
     4540  LHLD BYTE2
     4550  MOV A.L
     4560  RLC
     4570  RLC
     4580  RLC
     4590  RLC
     4600  ADD H                        NEXT TWO DIGITS ARE STORED AT
     4610  STA MEMR2                    LOCATION 'MEMR2'
     4620  POP B
     4630  POP D
     4640  POP H
     4650 BYTE1 DS 2
     4660 BYTE2 DS 2
     4670 BYTE3 DS 1
     4680 MEMR1 DS 1
     4690 MEMR2 DS 1
     4700  RET
*    SUBTRACT SUBROUTINE-  THIS ROUTINE SUBTRACTS
*     (STRT1) FROM (END1) & STORES IT IN B,C RP
*     STRT1 AND END1 ARE STARTING AND ENDING ADDRESSES OF THE
*     LOCATIONS  RESPECTIVELY WHEREFROM DATA TO BE READ OR PROGRAMMED
**************************************************
    4710 SUBTR LHLD END1
    4720  XCHG
    4730  LHLD STRT1
    4740  MOV A,H
    4750  SUB D
    4760  MOV C,A
    4770  MOV A,L
    4780  SBB E
```

```
   4790  MCV B.A
   4800  RET
************************************************************
*     THIS IS SUBROUTINE FOR
*     SHIFTING DATA FROM TWO MEMORY LOCATION TO D,E RP
************************************************************
   4810 DERP LHLD STRT2        STARTING ADDRESS WHERE DATA  TO BE
   4820  MOV D.L               SHIFTED OR PROGRAMMED IS LOADED
   4830  MOV E.H               IN D,E RP
   4840  RET
************************************************************
*
*     THIS IS SUBROUTINE FOR
*     SHIFTING DATA FROM TWO MEMORY LOCATION TO H.L RP
   4850 HLRP LHLD STRT1        STARTING ADDRESS WHEREFROM DATA
   4860  MOV A.L               TO BE SHIFTED IS LOADED IN H.L RP
   4870  MCV L.H
   4880  MOV H.A
   4890  RET
**************************************************************
*     SUBROUTINE FOR WRITING START.END ADDRESS & THEN ENTERING
*     THE ADDRESS MANUALLY FROM KEYBOARD FOUR DIGITS CAN BE
*     ENTERED AND ALSO CORRECTED BY CURSOR SHIFTING
**************************************************************
   4900 ESEADD CALL LSAD1      WRITE START END ADDRESS.
   4910  LXI H,0EAA0H
   4920  CALL CURSD            PLACE CURSOR AT DESIRED POSITION
   4930  CALL LDADR            LOAD ADDRESS
   4932  LHLD MEMR1            SHIFT THE ENTERED ADDRESS
   4935  SHLD STRT1            TO STRT1
   4940  CALL LEAD             WRITE END ADDRESS
   4950  LXI H,0EB20H
   4960  CALL CURSD            PLACE CURSOR AT POSITION
   4970  CALL LDADR
   4980  LHLD MEMR1            SHIFT THE ENTERED ADDRESS
   4990  SHLD END1             TO END1
   5000  RET
************************************************************
*     THIS IS SUBROUTINE FOR ENTERING ADDRESS WHERE DATA TO BE
*     STORED OR PROGRAMMED
************************************************************
   5010 ETSADD CALL LSAD3      WRITE'START ADDRESS' ON 3RD LINE
   5020  LXI H,0EBA0H
   5030  CALL CURSD
   5040  CALL LDARD
   5050  LHLD MEMR1            TRANSFER THE ENTERED ADDRESS
   5060  SHLD STRT2            TO STRT2
   5070  RET
************************************************************
*   SUBROUTINE FOR CLEARING THE MAIN MENU
************************************************************
   5080 CLEAR LXI D,00E950H
   5090  CALL SPACE
   5100  LXI D,0EA10H
   5110  CALL SPACE
   5120  LXI D,0EA90H
   5130  CALL SPACE
   5140  LXI D,0EB10H
   5150  CALL SPACE
   5160  LXI D,0EB10H
   5170  CALL SPACE
   5180  LXI D,0E90H
   5190  CALL SPACE
   5200  LXI D,0EC10H
   5210  CALL SPACE
   5220  LXI D,0EC90H
   5230  CALL SPACE
   5240  RET
```

```
********************************************************
*      THIS IS 'DELAY' SUBROUTINE WHICH MAKES A 50 MILLI SECOND
*      LOOP AT A CLOCK FREQUENCY OF 3 MEGA HERTZ
********************************************************
   5242 DELAY PUSH D
   5244  LXI D,1800H           1800H IS FOR 3 MEGA HERTZ CLOCK
   5246 DLLOP DCX D
   5248  MOV A,D
   5250  ORA E
   5252  JNZ DLLOP              LOOP 1800H TIMES
   5254  POP D
   5256  RET
********************************************************
*      FOLLOWING ARE THE SUBROUTINES FOR DISPLAYING DIFFERENT
*      WORDS AT PARTICULAR LOCATIONS. LOCATIONS ARE LOADED
*      IN D,E RP USING INSTRUCTION LXI D,ADDR
********************************************************
   5258 LMST1 LXI D,0EA94H   DISPLAY'MASTER'IN FIRST LINE
   5260  CALL DMST
   5270  RET
   5280 LMST2 LXI D 0EB14H   DISPLAY'MASTER'IN SECOND LINE
   5290  CALL DMST
   5300  RET
   5331 LMEM1 LXI D,0EA94H   DISPLAY'MEMORY'IN FIRST LINE
   5340  CALL DMEM
   5350  RET
   5360 LMEM3 LXI D,0EB94H   DISPLAY'MEMORY'IN SECOND LINE
   5370  CALL DMEM
   5380  RET
   5390 LSAD1 LXI D,0EA9BH   DISPLAY'START ADDRESS'IN
   5400  LXI H,SAD            FIRST LINE
   5410  CALL DSPT
   5420  RET
   5430 LEAD LXI D,0EB1BH    DISPLAY'END ADDRESS'
   5440  LXI H,EAD
   5450  CALL DSPT
   5470  RET
   5480 LSAD3 LXI D,0EB9BH   DISPLAY 'START ADDRESS'IN
   5490  LXI H,SAD            THIRD LINE
   5500  CALL DSPT             .
   5510  RET
   5520 LSLV1 LXI D,0EA94H   DISPLAY'SLAVE'IN FIRST LINE
   5580  CALL DSLV
   5590  RET
   5600 LSLV2 LXI D,0EB14H
   5610  CALL DSLV
   5620  RET
********************************************************
*      FOLLOWING ARE THE SUBROUTINES FOR DISPLAYING DIFFERENT
*      WORDS AT DIFFERENT LOCATIONS.WORDS ARE LOADED IN H,L RP
*      AND LOCATIONS ARE LOADED IN D,E RP.
********************************************************
   5630 DMST LXI H,MST        DISPLAY'MASTER'
   5640  CALL DSPT
   5650  RET
   5660 DSLV LXI H,SLV        DISPLAY'SLAVE'
   5670  CALL DSPT
   5680 RET
   5690 DPRG LXI H,PRG        DISPLAY'PROGRAM'
   5700  CALL DSPT
   5710  RET
   5720 DMEM LXI H,MEM        DISPLAY'MEMORY'
   5730  CALL DSPT
   5740  RET
   5750 DVER LXI H,VER        DISPLAY'VERIFY'
   5740  CALL DSPT
   5750  RET
   5760 DMS LXI H,MSE         DISPLAY'MASTER,SLAVE EPROM'
   5770  CALL DSPT
   5780  RET
   5790 DERS LXI H,ERS          DISPLAY'EPROM ERASED'
   5800  CALL DSPT
   5810 RET
   5820 DRD LXI H,RD            DISPLAY'READ'
   5830  CALL DSPT
   5840  RET
   5850 TO MVI L,4FH            LOAD ASCII CODE OF T AND O
   5860  MVI H,54H              IN H,L RP
   5870  RET
   5880 DDIS LXI H,DIS          DISPLAY 'DISPLAY'
   5890  CALL DSPT
   5900  RET
   5910 DMSM LXI H,MSM          DISPLAY'MASTER,SLAVE,MEMORY'
   5920  CALL DSPT
   5930  RET
   5940 DMOV LXI H,MOV          DISPLAY'MOVE'
   5950  CALL DSPT
   5960  RET
   5970 DPRT LXI H,PRT          DISPLAY 'PRINT'
   5980  CALL DSPT
   5990  RET
   6000 DEPR LXI H,EPR          DISPLAY'EPROM'
   6010  CALL DSPT
   6020  RET
   6030 DSLC LXI H,SLC          DISPLAY'SELECT'
   6040  CALL DSPT
   6050  RET
********************************************
*     SUBROUTINE'LOP'FOR LOOPING UNTIL CONTENT OF B,C RP IS 0
********************************************
   6060 LOP STAX D
   6070  INX D
   6080  DCR B
   6090  MOV A,B
   6100  CPI 00
   6110  RET
********************************************
*     SUBROUTINE FOR PUTTING 44 STARS(*) IN A LINE
********************************************
   6120 STAR MVI B,2CH
   6130 LOP1 MVI A,2AH          2A IS ASCII CODE FOR *
   6140  CALL LOP
   6150  JNZ LOP1
   6160  RET
********************************************
*     SUBROUTINE FOR PUTTING 40 SPACES IN A LINE
********************************************
   6170 SPACE MVI B,28H
   6180 LOP2 MVI A,20H          20H IS ASCII CODE FOR SPACE
   6190  CALL LOP
   6200  JNZ LOP2
   6210  RET
********************************************
*     SUBROUTINE FOR DISPLAYING 40 DASHES IN A LINE
********************************************
   6220 DASH MVI B,20H
   6230 LOP3 MVI A,2DH
   6240  CALL LOP
   6250  JNZ LOP3
   6260  RET
*     SUBROUTINE FOR MAKING 7 SPACES **************
   6270 SVNSP MVI B,07H
   5280 LOP4 MVI A,20H
   6290  CALL LOP
   6300  JNZ LOP4
   6310  RET
```

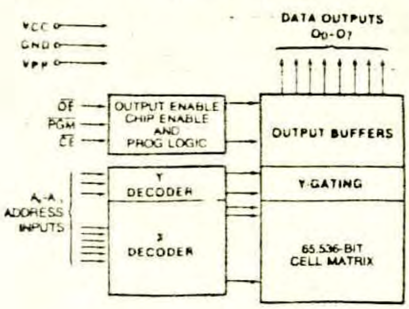
```
*     SUBROUTINE FOR PLACING CURSOR AT A PARTICULAR LOCATION
*     ADDRESS OF LOCATION IS LOADED IN H,L RP
***********************************************************
   6320 CURSD SHLD 0FFFEH
   6330  MVI B,20H
   6340  CALL 103H
   6350  RET
***********************************************************
*     DISPLAY SUBROUTINE. THE ADDRESS OF THE WORD TO BE
*     DISPLAYED IS LOADED IN H,L RP
***********************************************************
   6360 DSPT MOV A,M
   6370  CPI 0FFH
   6380  RZ
   6390  STAX D
   6400  INX H
   6410  INX D
   6420  JMP DSPT
   6430  RET
***********************************************************
*     SUBROUTINE FOR FETCHING THE CONTENT OF AN EPROM AND
*     DISPLAYING IT
***********************************************************
   6440 DSLOP MOV A,H
   6450  OUT 0B9H                SEND HIGH BYTE ADDRESS
   6460  MOV A,L
   6470  OUT 0BAH                SEND LOW BYTE ADDRESS
   6480  IN 0B8H                 INPUT THE CONTENT
   6490  PUSH B
   6500  MOV B,A
   6510  CALL 103H
   6520  POP B
   6530  INX H
   6540  MOV A,B
   6550  ORA C
   6560  JNZ DSLOP
   6590  RET
***********************************************************
*     DATA FIELD STARTS FROM HERE.THE BYTES ARE ASCII
*     CODE OF THE WORD WRITTEN AT THE RIGHT SIDE
***********************************************************
   6700 SAD DW 1314H             START ADDRESS
   6710  DW 1112H
   6720  DW 1420H
   6730  DW 0104H
   6740  DW 0412H
   6750  DW 0D13H
   6760  DW 1320H
   6770  DB 0FFH
   6780 EAD DW 2020H             END ADDRESS
   6790  DW 050EH
   6800  DW 0420H
   6810  DW 0404H
   6820  DW 0412H
   6830  DW 0513H
   6840  DW 1320H
   6850  DB 0FFH
   6860 COP DW 0305H             COPY
   6870  DW 1019H
   6880  DB 0FFH
   6890 FRM DW 0612H             FROM
   6900  DW 0F0DH
   6910  DB 0FFH
   6920 MST DW 0D01H             MASTER
   6930  DW 1314H
   6940  DW 0512H
   6950  DB 0FFH
```

```
6960 SLV DW 130CH        SLAVE
6970  DW 0116H
6980  DW 0520H
6990  DB 0FFH
7010 SLC DW 1205H        SELECT
7020  DW 0C05H
7030  DW 0314H
7040  DB 0FFH
7050 PRG DW 1012H        PROGRAM
7060  DW 0F07H
7070  DW 1201H
7080  DW 0D20H
7090  DE 0FFH
7100 UTL DW 150EH        UNTIL
7110  DW 1409H
7120  DW 0C20H
7130  DE 0FFH
7140 MEM DW 0D05H        MEMORY
7150  DW 0D0FH
7160  DW 1219H
7170  DB 0FFH
7180 SEL DW 1220H        SELECTION MENU
7190  DW 0520H
7200  DW 0C20H
7210  DW 0520H
7220  DW 0320H
7230  DW 1420H
7240  DW 0920H
7250  DW 0F20H
7260  DW 0E20H
7270  DW 2020H
7080  DW 0D20H
7290  DW 0520H
7300  DW 0E20H
7310  DW 1520H
7320  DE 0FFH
7330 DIS DW 0409H        DISPLAY
7340  DW 1310H
7350  DW 0C01H
7360  DW 1920H
7370  DB 0FFH
7380 VER DW 1605H        VERIFY
7390  DW 1509H
7400  DW 1209H
7410  DE 0FFH
7420 EPR DW 0510H        EPROM
7430  DW 120FH
7440  DW 0D20H
7450  DB 0FFH
7460 ERS DW 0510H        ERASED
7470  DW 0113H
7480  DW 0504H
7490  DE 0FFH
7500 RD DW 1205H         READ
7510  DW 0104H
7520  DE 0FFH
7530 MOV DW 0D0FH        MOVE
7540  DW 1605H
7550  DB 0FFH
7560 PRT DW 1020H        PRINT
7570  DW 097EH
7580  DW 1420H
7590  DE 0FFH
7600 MS DW 0D01H
7610  DW 1314H
7620  DW 0512
7630  DW 2013H
7640  DW 0C01H
7650  DW 1605H
7660  DB 0FFH
```

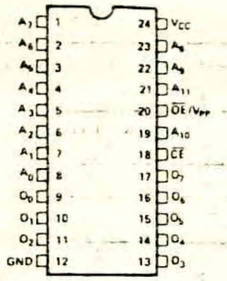
APPENDIX-B

Pin Diagrams and Programming Waveforms

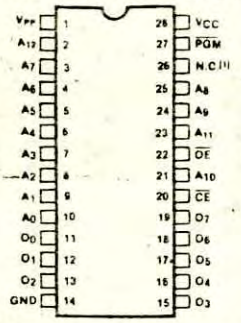
BLOCK DIAGRAM



2732A PIN CONFIGURATION



2764 PIN CONFIGURATION



[1] For upgradability to JEDEC approved 128K EPROMs, provide an address line to pin 26. For compatibility with the 2732A and 32K ROMs, provide a trace from V_{CC} to pin 26.

MODE SELECTION

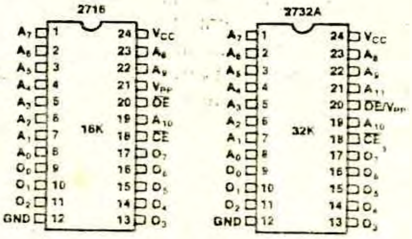
PINS / MODE	$\overline{CE}$ (20)	$\overline{OE}$ (22)	$\overline{PGM}$ (27)	V_{PP} (1)	V_{CC} (28)	Outputs (11-13, 15-19)
Read	V_{IL}	V_{IL}	V_{IH}	V_{CC}	V_{CC}	D_{OUT}
Standby	V_{IH}	x	x	V_{CC}	V_{CC}	High Z
Program	V_{IL}	x	V_{IL}	V_{PP}	V_{CC}	D_{IN}
Program Verify	V_{IL}	V_{IL}	V_{IH}	V_{PP}	V_{CC}	D_{OUT}
Program Inhibit	V_{IH}	x	x	V_{PP}	V_{CC}	High Z

x can be either V_{IL} or V_{IH}

PIN NAMES

A_0-A_{12}	ADDRESSES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}$	OUTPUT ENABLE
O_0-O_7	OUTPUTS
$\overline{PGM}$	PROGRAM
N.C.	NO CONNECT

*HMOS is a patented process of Intel Corporation.



Refer to 2732A data sheet for specifications.

PIN NAMES

A_0-A_{10}	ADDRESSES
$\overline{CE}$/PGM	CHIP ENABLE/PROGRAM
$\overline{OE}$	OUTPUT ENABLE
O_0-O_7	OUTPUTS

Figure 1. Pin Configuration

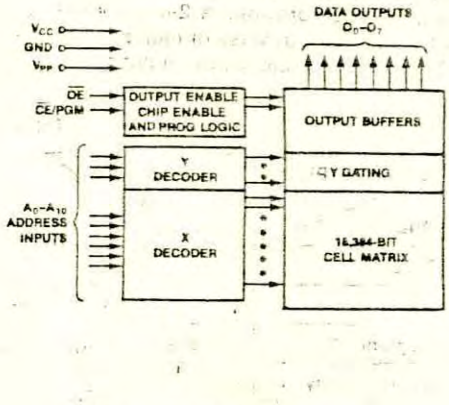
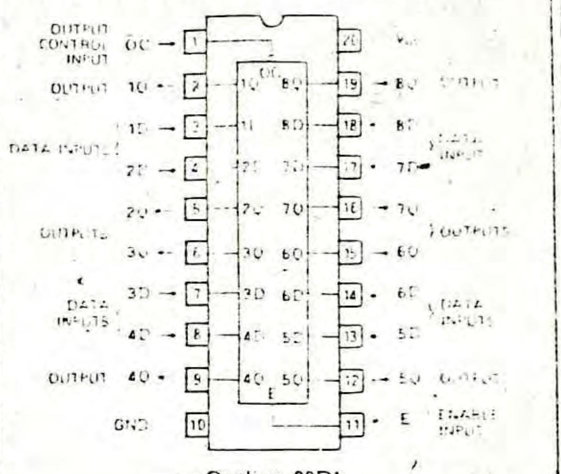


Figure 2. Block Diagram

Pin diagram of intel EPROMs.

PIN CONFIGURATION (TOP VIEW)



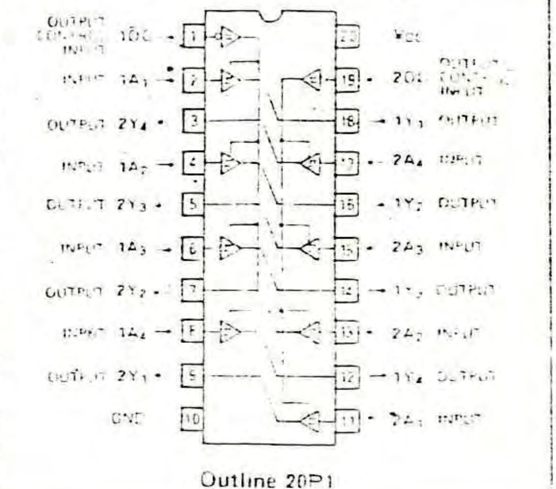
Outline 20P1

FUNCTIONAL DESCRIPTION

Since the 8 D-type latches use pnp transistor input for the output control input OC and enable input E, which are common to all 8 circuits, the input load factor is small. With a hysteretis of 400mV (typical) specially given to the input circuit E, noise margin is high. When E is high, the information from the data input D appears in the output Q. When the D signal changes, the signal that appears in Q also changes. When E changes from high to low, the status of D immediately before the change is latched. While E is low, the status of Q does not change even if the D is changed. When $\overline{OC}$ is high, 1Q – 8Q are all put in the high impedance state irrespective of other input signals. Since all outputs have high fan-out, this device is suitable for use as a buffer register, I/O ports or bi-directional bus driver. For application, see M74LS374P.

Pin diagram of Intel74LS373.

PIN CONFIGURATION (TOP VIEW)



Outline 20P1

FUNCTIONAL DESCRIPTION

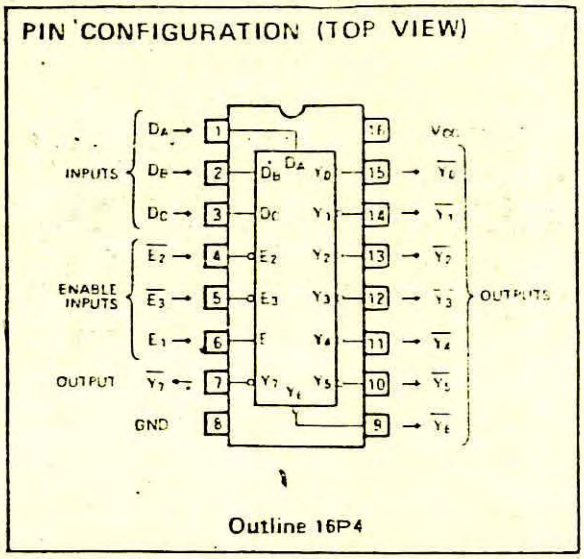
The use of pnp transistors in the input circuit has enabled the achievement of small input load factor. With hysteresis characteristics, the buffer has a 3-state noninverted output with high noise margin.

When output control input $\overline{OC}$ is low, the output Y is low if input A is low and Y is high if A is high. When $\overline{OC}$ is high, all of Y_1, Y_2, Y_3, and Y_4 are in the high-impedance state, irrespective of the status of A.

By connecting $1\overline{OC}$ with $2\overline{OC}$, it becomes possible to control the output of all 8 circuits simultaneously. Output can be terminated by a load resistor of 133Ω or over.

For standard characteristics, see M74LS241P.

Pin diagram of 74LS244.

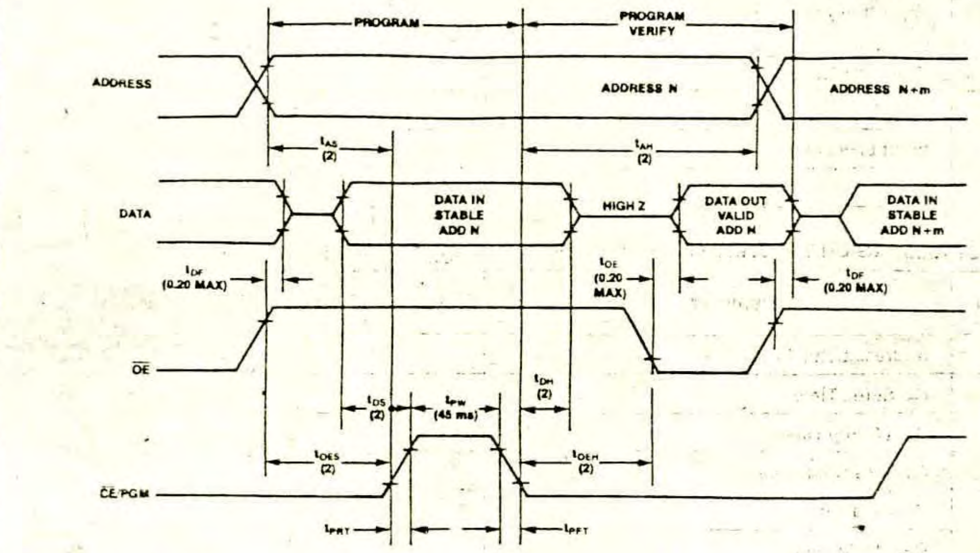


FUNCTIONAL DESCRIPTION

For use as a decoder, specify inputs D_A./D_B, and D_C in 3-bit binary code. In the case of decoding function, the E_1 is kept in high state while $\overline{E_2}$ and $\overline{E_3}$ are kept low. If E_1, $\overline{E_2}$ and $\overline{E_3}$ are not in these conditions, all the outputs become high, irrespective of the status of $D_A \sim D_C$. For use as a demultiplexer, $\overline{E_1}$, $\overline{E_2}$ and E_3 are used as data inputs and D_A, D_B, and D_C as selection inputs This forms a 1-line to 8-line demultiplexer.

Pin diagram of 74LS138.

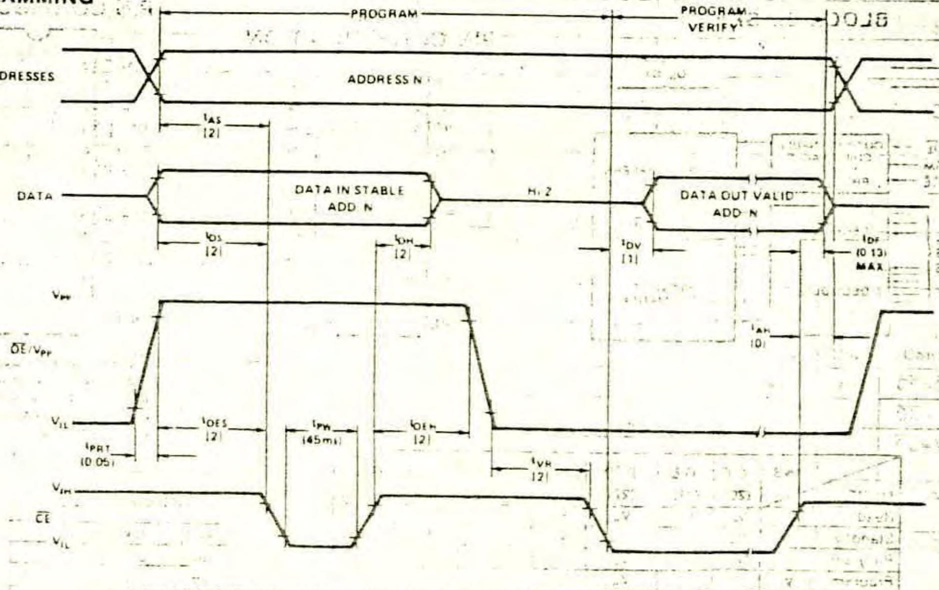
PROGRAMMING WAVEFORMS (V_{PP} = 25V ±1V, V_{CC} = 5V ±5%)



NOTE:
ALL TIMES SHOWN IN PARENTHESES ARE MINIMUM TIMES AND ARE μSEC UNLESS OTHERWISE NOTED.

Programming Waveform of Intel 2716

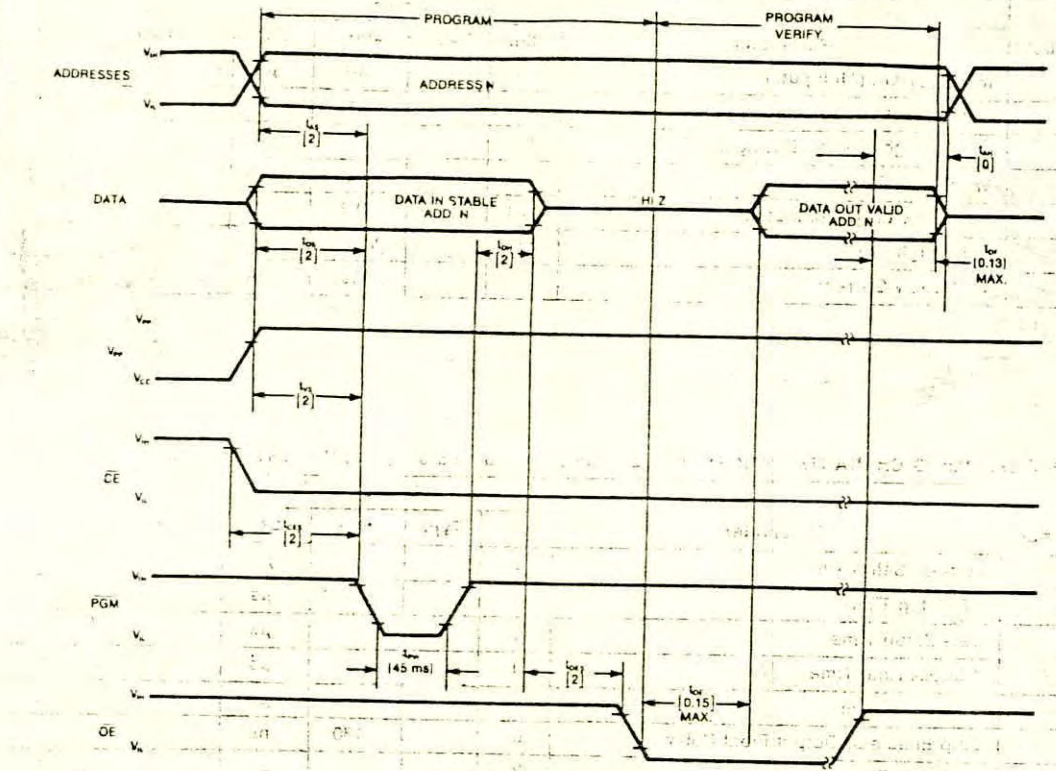
PROGRAMMING



1. ALL TIMES SHOWN IN ( ) ARE MINIMUM AND IN μSEC UNLESS OTHERWISE SPECIFIED
2. THE INPUT TIMING REFERENCE LEVEL IS 1V FOR A V_{IL} AND 2V FOR A V_{IH}

Programming Waveform of Intel 2732

PROGRAMMING WAVEFORMS

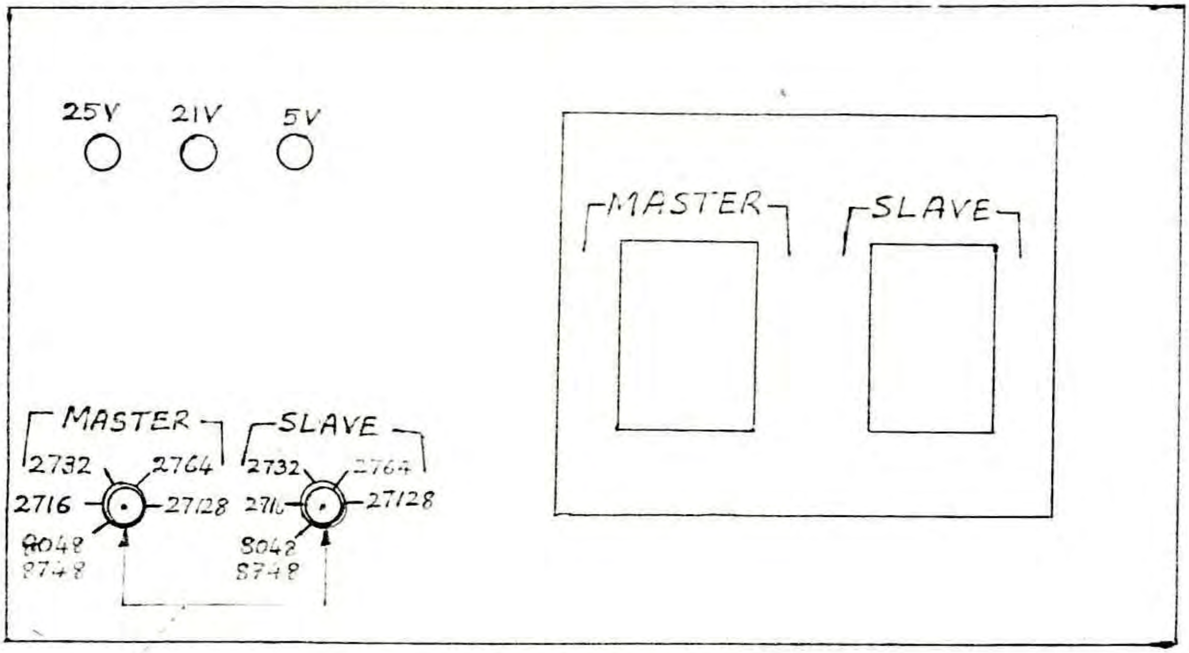


1. ALL TIMES SHOWN IN [ ] ARE MINIMUM AND IN µSEC UNLESS OTHERWISE SPECIFIED.
2. THE INPUT TIMING REFERENCE LEVEL IS 1V FOR A V_{IL} AND 2V FOR A V_{IH}.
3. t_{OE} AND t_{DF} ARE CHARACTERISTICS OF THE DEVICE BUT MUST BE ACCOMMODATED BY THE PROGRAMMER.

Programming Waveforms of 2764 &27128.

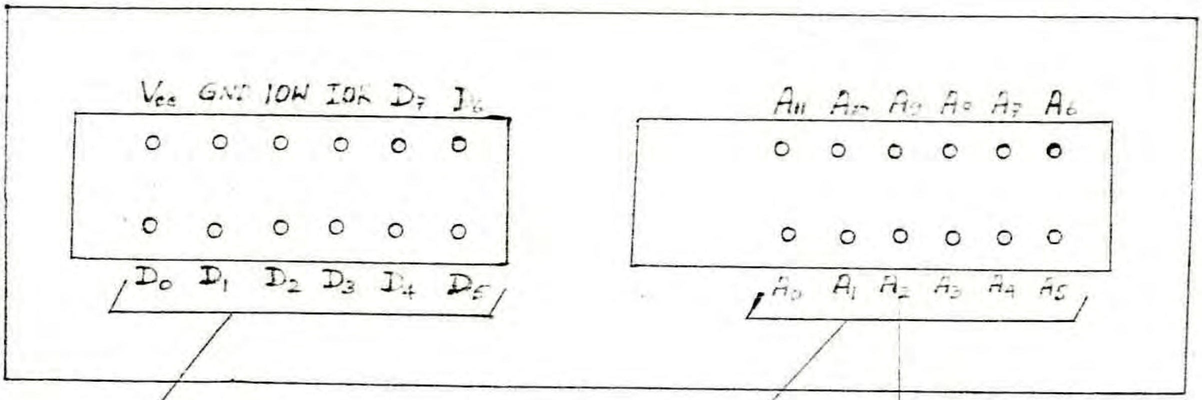
APPENDIX - C

Printed Circuit Board Designs



→Selection knob for connection to different type of EPROM

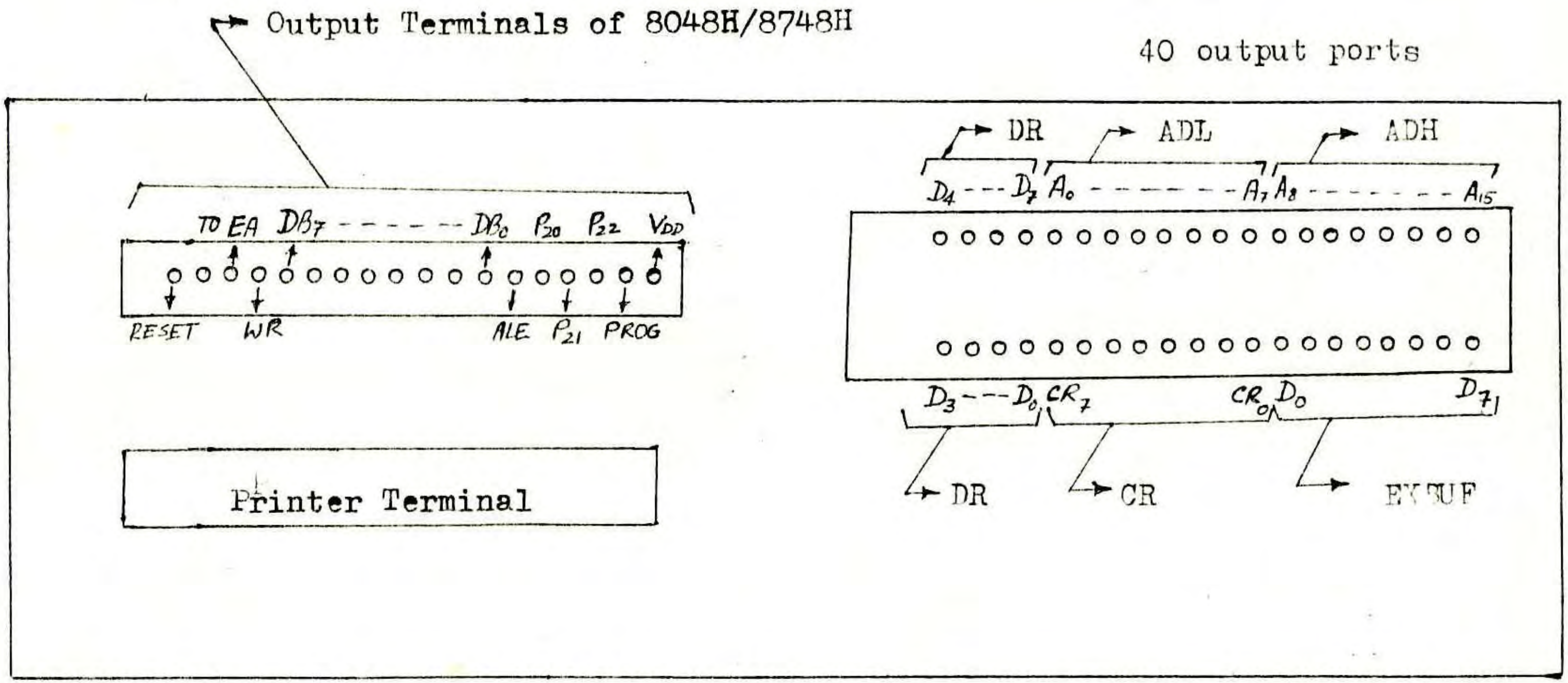
Top side view of the EPROM rogrammer box.(above)


V_{cc} GND IOW IOR D_7 D_6
D_0 D_1 D_2 D_3 D_4 D_5
A_{11} A_{10} A_9 A_8 A_7 A_6
A_0 A_1 A_2 A_3 A_4 A_5

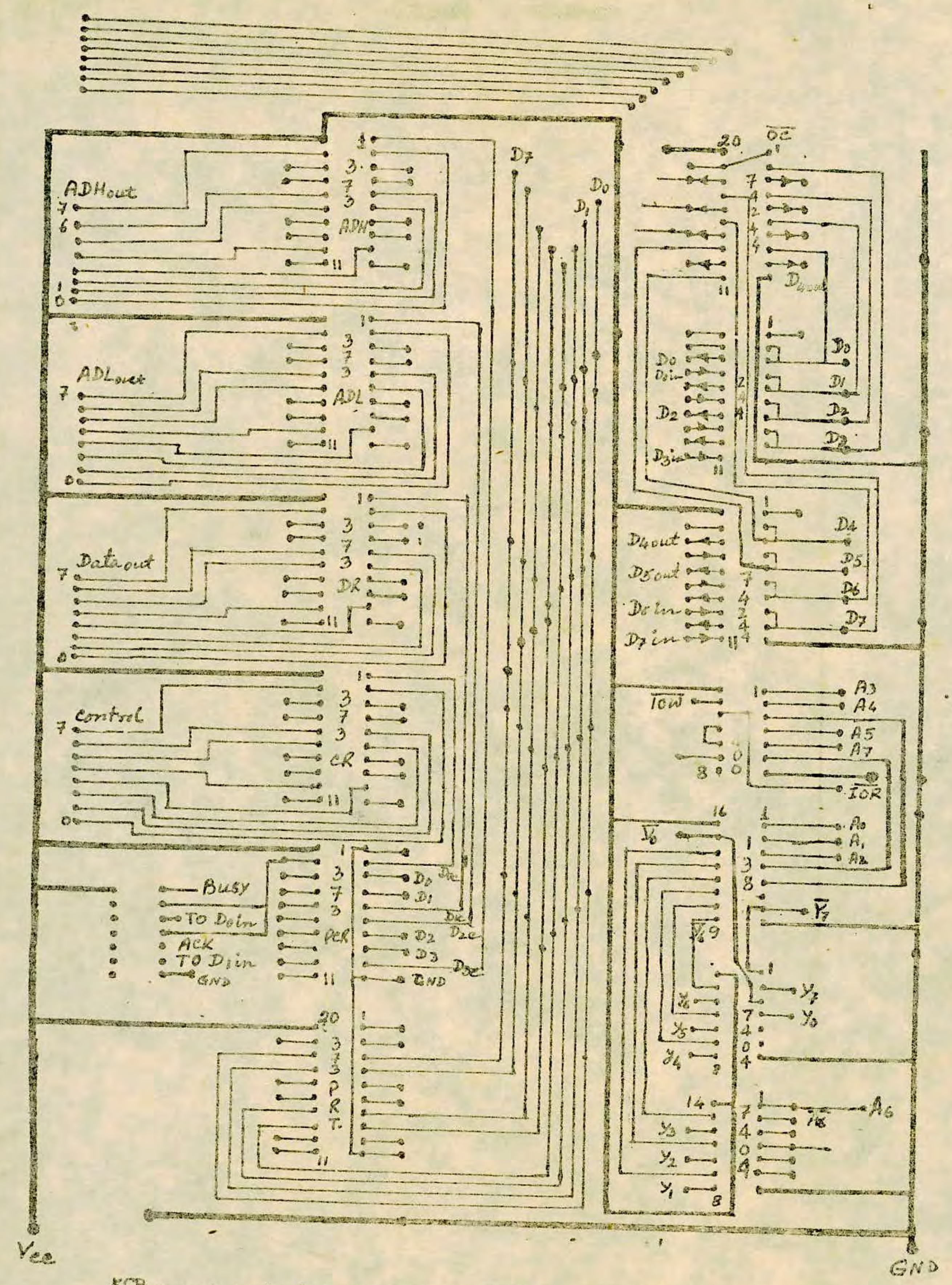
→To Data bus of μC

→To Address bus of μC

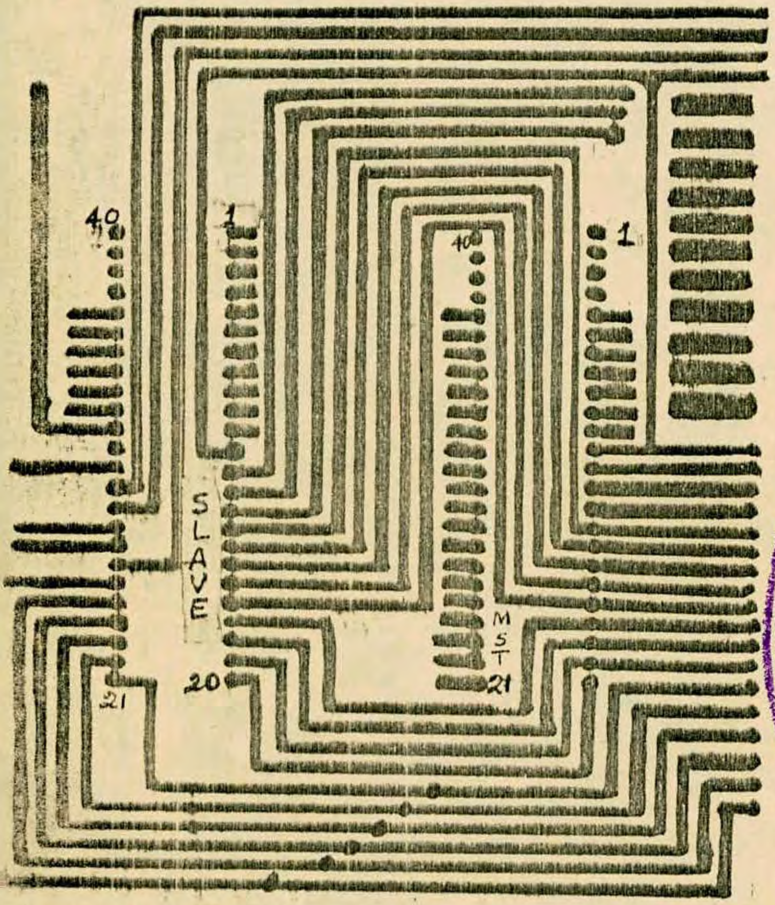
Front view of the EPROM Programmer Box.



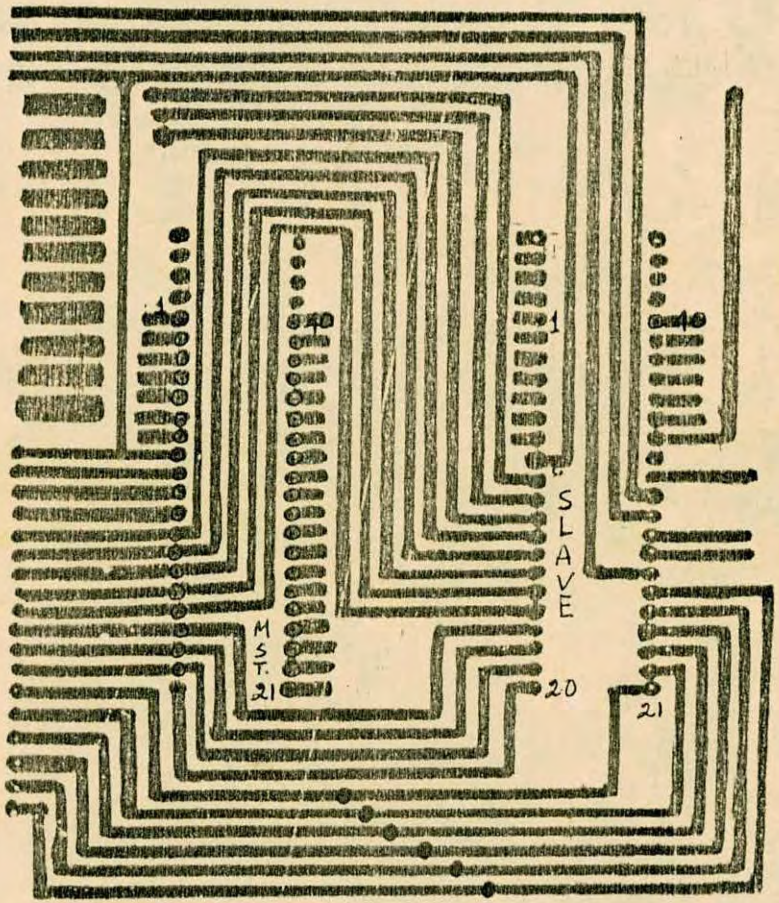
Terminal outputs on the back side



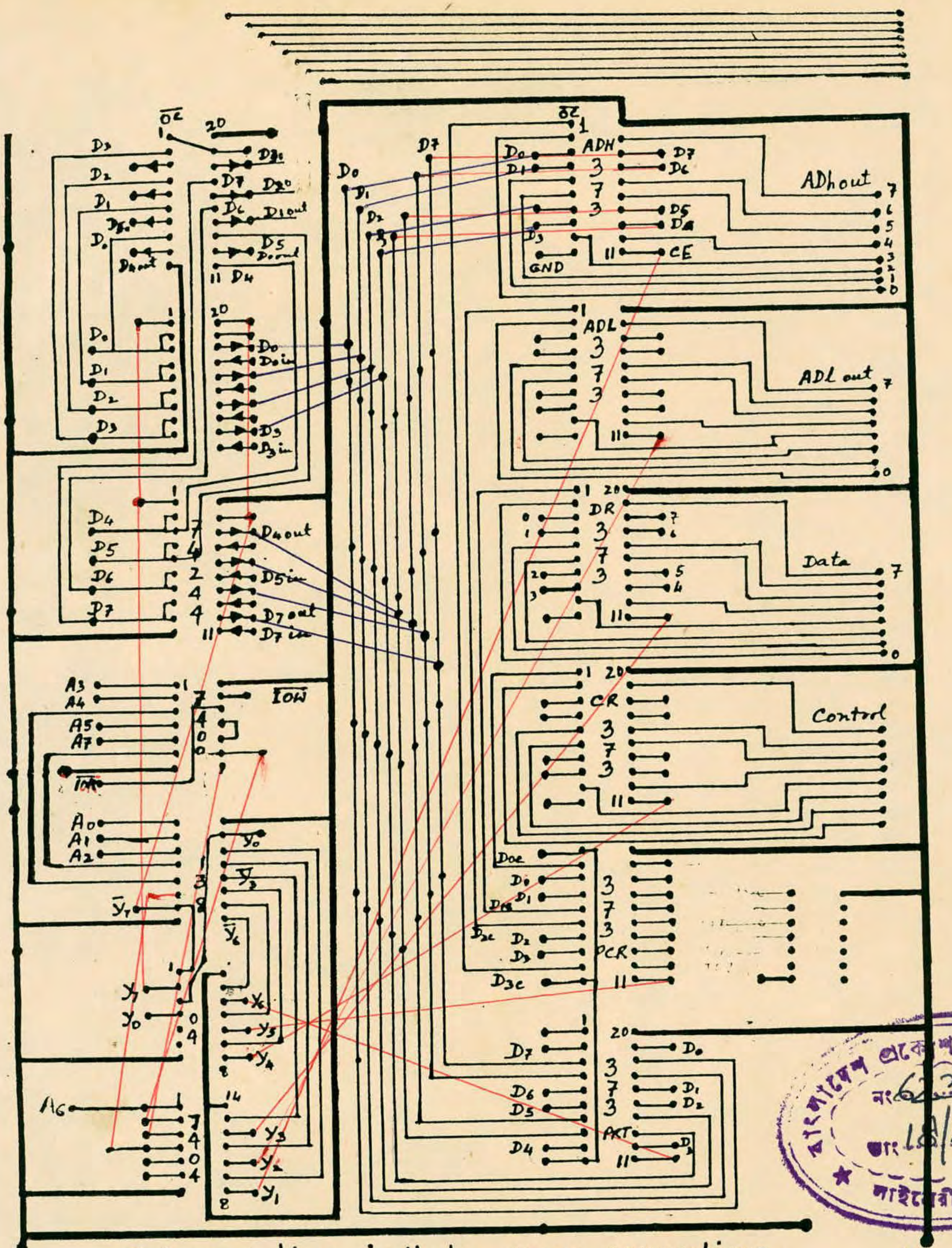
PCB Board design of the Programmer(Reverse side)



PCB of Master & slave connection(Reverse side)



PCB of Master & slave connection(component side)



Coloured lines indicates cross connections.
ADH - Address high registers ; CR - Control register.
ADL - ,, Low ,, ; DR - Data register.
PCR - Port Control ,, ; PRT - Printer ,, .

PCB Board design of the Programmer(component side)