ESA 85 - 2
USER MANUAL

Electro Systems Associates Pvt Ltd
www.esaindia.com

PREFACE

This is the user's manual for ESA85-2 microprocessor trainer. This manual describes the hardware and software components of ESA85-2 and gives the interface information necessary for expanding the system.

This manual describes in detail the facilities offered by the Keyboard Monitor Program, Serial Monitor Program, the Text Editor and 2-pass Assembler Disassembler Packages. The on-board facilities, Audio Cassettee Interface, EPROM Programmer Interface, Centronics Parallel Printer Interface are also described in this manual. Communication with a host computer is also described. Complete schematics and related drawings are provided in the appendices.

Please note that this volume is a user's guide for ESA85-2 and as such does not deal elaborately with the features of 8085 processor and related peripherals and their programming. Details regarding these can be obtained from the following INTEL Publication.

Micro systems Component Handbook Vol I and II

While every effort has been made to present the information in an accurate and simple fashion, we do welcome suggestions for improving the quality and usefulness of this manual.

Please address all your correspondence to :

ELECTRO SYSTEMS ASSOCIATES PVT LTD.,
4215 J.K. Complex, First Main Road, Subramanyanagar
P.O. Box No. 2139 BANGALORE - 560 021 INDIA
Fax : 91-80-3325615 Phone : 3323029 3322924
email : esaindia@vsnl.com,
www.esaindia.com

HOW TO USE THIS MANUAL

This manual is organized as follows :

Chapter 1	**INTRODUCTION,** gives a general description of ESA85-2 system (both hardware and software features.)
Chapter 2	**CONFIGURATION AND INSTALLTION,** describes the configuration of the system to user requirements and the installation of ESA85-2 for serial or keyboard mode of operation.
Chapter 3	**KEYBOARD MONITOR,** describes in detail various commands available to the user when the keyboard monitor is in control of the system (operated via on-board hexadecimal keyboard).
Chapter 4	**SERIAL MONITOR,** describes the commands available to user when the serial monitor is in control of the system.
Chapter 5	**HARDWARE,** describes the hardware organization of ESA85-2 and provides complete information for expanding the system. All connector details are provided in this chapter.
Chapter 6	**MONITOR ROUTINES ACCESSIBLE TO USER,** describes different monitor routines available to user from both serial and keyboard monitor and lists them with calling conventions and register utilisation.
Chapter 7	**AUDIO TAPE INTERFACE,** describes the features of the on-board audio tape interface, its installation, its operation from both keyboard and serial mode and the data formats.
Chapter 8	**PARALLEL PRINTER INTERFACE,** describes the features of the on-board centronics compatible parallel printer interface, its installation, operation and connector requirements.
Chapter 9	**PROM PROGRAMMER SYSTEM,** decribes the features of the on-board EPROM Programmer, its installation and its operation from both keyboard and serial mode.

Chapter 10	**TEXT EDITOR,** describes the line-oriented text-editor faciilty provided in serial mode of operation.
Chapter 11	**ASSEMBLER,** describes the operation of the powerful, PROM-resident 2-pass assembler, provided in serial mode of operation.
Chapter 12	**DISASSEMBLER,** describes the 2-pass disassembly facility provided in serial mode of operation.
Chapter 13	**COMMUNICATION WITH A HOST COMPUTER SYSTEM,** describes how ESA85-2 can be operated from a host computer system like an IBM PC/XT/AT compatible system.
Chapter 14	**PROGRAMMING EXAMPLES,** presents a variety of programming examples to be executed on ESA85-2. These examples range from simple familiarization programs to more elaborate ones illustrating the use of monitor routines, interrupts etc.

The **APPENDICES** contain the complete Schematics of ESA85-2, Component layout, connector details and some useful reference information.

CONTENTS

CHAPTER 1

INTRODUCTION

ESA85-2 is an extremely powerful microprocessor trainer based on the popular 8085 CPU. It can be used as a flexible instructional aid in academic institutions. Sophisticated features like powerful 2-pass Assembler and Text Editor allow the use of ESA85-2 for serious microcomputer developmental work in research institutions and R&D laboratories.

On-board facilities like EPROM programmer, Audio Tape Interface and Centronics compatible Parallel Printer Interface enhance the power and utility of ESA85-2 system.

Following are the main features of ESA85-2:

* ESA85-2 can be operated either from on-board keyboard or from a CRT terminal through its RS 232 C interface.
* Keyboard and serial monitor programs support the entry of user programs, editing and relocation, debug facilities like breakpoints and single-stepping, direct port input/output and full speed execution of user programs.
* Text editor permits line-oriented editing facilities for ASCII files and saving and restoring of files from audio cassette.
* 2-pass Assembler can assemble any memory-resident assembly language program. Supports all the standard INTEL mnemonics, useful directives like ORG, DB, DW, DS, EQU, SET etc. Supports labels of upto six characters.
* 2-Pass Disassembler disassembles the object code into standard INTEL mnemonics, converts all branch and subroutine addresses into lablels for easy reference.

* On-board PROM Programmer features a 28-pin ZIF socket to program all standard PROMs 2716 through 27512. Intelligent Programming Algorithm is used wherever possible resulting in significant reduction in programming time (for eg. less than 2 minutes for programming a 27128!).
* On-board Audio Tape Interface provides MIC and EAR sockets for operation with an Audio Tape recorder. Reliable storing/loading of assembly language source programs, object codes, data etc is possible.
* On-board interrupt controller 8259A to service upto 8 interrupts.
* 32 K Bytes of CMOS static RAM is provided with battery back-up option. Total on-board memory can be upto 64K Bytes.
* Allows multi-processor system design by supporting the HOLD and HLDA signals.
* STD bus compatible signals available on the bus connector for easy expansion.

Options Available

a. Interface Modules for training purpose (Calculator Keyboard, Elevator, Display, ADC with DAC, Dual Slope ADC, Dual DAC, Logic Controller, Traffic Lights, 8253 Demo, RTC, Tone Generator, Stepper Motor, Numerical Printer, etc.)
b. 8-bit, 16 Channel ADC
c. 5 1/4" Floppy Disk Drive Interface.
d. 26 Core Ribbon Cable Connector Set.
e. RS-232C Interface Cable for terminal (25 pin D type male connector on both sides of the cable)
f. XT852: A communication package that can be used with a PC/XT/AT or compatible systems with connecting cable (25 pin D type male on one end and 25 pin D type female on the other end)
g. Ni-Cd battery

SPECIFICATIONS :

CPU	**:**	8085 Operated at 3.072 MHz
Memory	**:**	Three 28-pin JEDEC sockets offer 64K Bytes of memory as follows: 16K Bytes of firmware in one 27128 32K Bytes of static RAM Using one 62256 with optional battery backup 16K Bytes of PROM/RAM (Optional) through Jumper selections allow this socket to accommodate 2764/27128/6264/62256.

Firmware :

Serial and Keyboard Monitors.

Text Editor, Assembler (2pass), Disassembler package (can be used in serial mode only)

Audio tape interface driver software.

Centronics printer interface driver software.

PROM Programming software.

Peripherals

8279-5: To control 32 keys keyboard and 6-digit, 0.5" seven segment LED display.

8253-5: 3 Programmable interval timers
Timer 0 is used for implementing single-step facility, Timer 1 is used for generating baud clock, Timer 2 is available to the user (Through jumper option, user can use Timer 1 also, if he does not use it for baud clock).

8251 A: For serial communication supporting all standard bauds from 110 to 19,200. (Baud is selected through on-board DIP switch)

8259 A: Programmable Interrupt Controller accepts 8 Interrupt signals from the auxillary system connector.

8255 -5: (4 numbers) Two are used by the system to implement PROM Programmer, Audio Cassette Interface and Parallel Printer interface. The other two are available to user giving 48 programmable I/O Lines.

Bus Expansion

STD bus Compatible signals are available on a strip connector.

Interface Signals

Parallel I/O: 48 lines (2 X 8255-5) of TTL compatible bus brought-out to two Spectra-strip type ribbon cable connectors.

Serial I/O: RS-232C with standard MODEM control signals through on-board 25 pin D-type female connector.

Cassette Interface Signals : Available on MIC and EAR sockets.

Parallel Printer Signals : Centronics compatible Parallel Printer Interface signals available on a 25 pin D-type female connector in IBM PC/XT Compatible pin configuration.

Interrupts

All interrupts except TRAP (used for single-step implementation) are available to user.

Power Supply (Optional)

  5V, (± 0.1V), 3.0A
+ 12V, (± 1.0V), 250mA
- 12V, (± 1.0V), 100mA
  30V, (± 2.0V), 100mA

CHAPTER 2

CONFIGURATION AND INSTALLATION

2.1 CONFIGURATION OF ESA85-2

ESA85-2 Microprocessor trainer is versatile and can be configured in number of ways as determined by the settings of a DIP switch and other jumpers (refer to the component layout diagram in Appendix C to locate the DIP switch and the jumpers). This chapter describes all the configuration options and the installation procedures.

2.1.1 OPERATIONS MODE SELECTION

ESA85-2 can be operated either in the hexadecimal keypad mode or in the serial mode. In the hexadecimal keypad mode the trainer is operated from the hexadecimal keyboard/display unit. In the serial mode, the trainer is connected to a CRT terminal or to a host computer system (like IBM PC/XT/AT Compatibles) through as RS 232 C interface. In either mode of operation, the system provides a variety of commands for program development/debugging. However, several features of ESA85-2 like editor, disassembler, assembler etc., are available only in the serial mode of operation. The selection of the desired mode of operation is done as follows:

Switch 4 of the DIP switch	Operational mode
OFF	hexadecimal keypad mode.*
ON	Serial mode

(* factory installed option)

Chapter 3 describes the commands available in keyboard mode and chapter 4 describes the commands available in serial mode.

2.1.2 PRINTER ENABLE/DISABLE

ESA85-2 firmware includes the driver program for centronics compatible parallel printer interface. This driver can be enabled/disabled as shown below:

Switch 6 of the DIP switch	Printer driver
OFF	Disabled.*
ON	Enabled.

(* factory installed option)

Chapter 8 describes the parallel printer interface in detail.

2.1.3 BAUD RATE SELECTION

In serial mode of operation, ESA85-2 configures the on-board 8251A USART as follows:

* asynchronous mode
* 8 Bit character length
* 2 stop Bits
* no parity
* Baud Rate factor of 16X

Timer 1 of the on-board 8253A provides the transmit and receive Baud clocks for the USART. (Refer to chapter 5 for a detailed discussion of the Hardware). This timer is initiated by the system firmware to provide proper Baud clock based on the settings of the DIP switch as shown below:

Baud rate selection

S3	S2	S1	Baud rate
ON	ON	ON	110
ON	ON	OFF	300
ON	OFF	ON	600
ON	OFF	OFF	1200
OFF	ON	ON	2400
OFF	ON	OFF	4800
OFF	OFF	ON	9600*
OFF	OFF	OFF	1,9200

(* factory installed option)

2.1.4 MEMORY SELECTION

ESA85-2 has three 28-Pin sockets labelled U7,U8,U9 for memory. Two of these sockets are populated and the third one is for expansion by the user. System firmware (16K bytes) is supplied in a 27128 EPROM at the socket U7. 32K bytes of static RAM is provided by a 62256 at the socket U9. The other socket, U8 is for user expansion. This socket can be configured, through jumper settings JP6,JP7 and JP8, to accept 2764 or 27128 or 6264 or 62256 to provide 8K or 16K bytes of PROM/RAM. As installed at the factory, the jumpers are set for 27128 option. But the socket is left unpopulated. The configuration can be set as shown below:

Device	Address Range	JP6	JP7	JP8
2764	8K EPROM (4000H-5FFFH) (6000H-7FFFH)	A	A	A
6264	8K RAM (4000H-5FFFH) (6000H-7FFFH)	A	B	A
27128	16K EPROM (4000H-7FFFH)	B	A	A*
62256	16K RAM (4000H-7FFFH)	B	B	B

(*Default Jumper setting)

2.1.5 INTERRUPT AND RESETIN SELECTION

The sources for the vectored interrupts, RST5.5, RST6.5, RST7.5 and the TRAP can be selected to be either on-board signals or off-board signals. However, please note that the source for TRAP is configured to be the STEP signal from the timer. This selection is necessary for implementing the single-step facility provided by both the keyboard monitor and serial monitor. Further, the reset input to the CPU can be selected to come from the on-board RESET key or from an off-board source. The jumper settings, their interpretation and their default settings are shown below:

JP1=	OPEN	:RST7.5 source in external signal RST75[*]
	CLOSED	:RST7.5 source is on-board signal KBINT
JP2=	OPEN	:RST6.5 source in external signal RST65[*]
	CLOSED	:RST6.5 source is on-board signal RXRDY
JP3=	OPEN	:RST5.5 source in external signal RST55[*]
	CLOSED	:RST5.5 source is on-board signal 8279INT
JP4-A		:TRAP source is the external signal NMIRQ[*]
JP4-B		:TRAP source is the on-board signal STEP[*]
JP5-A		:RESET-IN* source is the on-board signal KBRESETIN*[*]
JP5-B		:RESET-IN* source is the off-board signal PBRESETIN*

[*] Default factory setting.

Notes:

1. The off-board signals RST55, RST65, RST75 are brought to the auxillary system connector P2. The signals NMIRO* and PBRESETIN* are brought to the main system connector P1. (Refer the component layout diagram in Appendix C and chapter 5 on Hardware.)
2. TRAP source must be the on-board signal STEP if the single-step feature of the monitor is required.
3. RESET-IN* source must be on-board signal KBRESETIN* if the on-board key RESET is to be made use of.

2.2 INSTALLATION OF ESA85-2

To install ESA85-2, the following accessories are required.

a) Power Supply 5V, 1.2 AMP Additionally + 12V, at 250mA -12V at 100mA for serial mode of operation 30V, 100mA if PROM Programmer system is being used. (Refer chapter 9 for details of this interface)

b) For Serial mode of operation: CRT terminal with RS 232 C interface OR Host system (like an IBM PC/XT/AT Compatible) with the driver software for host system. (Refer chapter 13 for details).

2.2.1. INSTALLATION PROCEDURE FOR SERIAL MODE OF OPERATION:

a) Select Serial mode of operation (Ref. Section 2.1.1)

b) Select printer if required (Ref. Section 2.1.2)

c) Set the desired baud rate (Ref. Section 2.1.3)

d) Select interrupt sources if required (Ref. Section 2.1.5)

e) Select memory configuration if necessary (Ref. Section 2.1.4)

f) Connect ESA85-2 to the CRT terminal/Host system through RS 232 C cable (Appendix E describes the RS 232 C interface requirements) over the connector J5. (Refer Appendix C for locating the connectors). If a terminal is being used, turn on the terminal. If a computer system is being used, turn on the system and execute the driver program. (Ref. Chapter 13 for details).

g) Connect the Power Supply of required capacity to ESA85-2 and turn on the power.

h) Press the RESET Key on ESA85-2.

Now the following sign-on message should appear on the screen 'ESA85-2 SERIAL MONITOR V x.y'

(V x.y indicates version x and revision y)

The sign-on message is followed by the command prompt, "." in the next line.

Now ESA85-2 is ready for operation in Serial mode.

NOTE:

The Display module will display "Serial"

2.2.2 NO RESPONSE IN SERIAL MODE

If there is no response from ESA85-2 in serial mode, after installing it is described in the previous section, check the following items:

a) Check the RS 232 C cable connections at both the ends. (Appendix E describes the interface in detail)

b) Check the power supply connections and voltage levels.

c) Check the handshake signals of RS 232 C interface (Ref.Appendix E)

d) Check the baud rates of ESA85-2 and the device connected to it.

e) If a computer system is the controlling device, check that the driver program is running, the RS 232 C cable is connected to the correct port and that the port is working

f) Check the configuration of ESA85-2 again. (DIP Switch settings, jumpers).

NOTE:

DIP Switch status is read only at power ON/RESET. If you change the settings, either press the RESET key or switch off and then switch on the power supply.

If the problem still persists, please contact the manufacturer.

2.2.3. INSTALLATION PROCEDURE FOR KEYBOARD MODE OF OPERATION

a) Select Keyboard mode of operation (Ref. Section 2.1.1)

b) Select interrupt sources if required (Ref. Section 2.1.5)

c) Set the memory configuration if necessary (Ref. Section 2.1.4)

d) Connect the power supply of required capacity to ESA85-2 and switch on the power.

e) Press the RESET key of the Keyboard/Display module.
Now the following sign-on message will appear on the seven-segment display.

-ESA 85

Now ESA85-2 is ready for operation in the keyboard mode.

2.2.4. NO RESPONSE IN KEYBOARD MODE

If correct sign-on message does not appear in the keyboard mode, check the following items.

a) If seven-segment display is totally blank, check the power supply connections and voltages.

b) If seven-segment display shows random pattern, check the configuration settings once again.

NOTE:

DIP Switch is read only at power ON/RESET. If you change the settings, either press the RESET key or switch off and then switch on the power supply.

If the problem persists, please contract the manufacturer.

CHAPTER 3

KEYBOARD MONITOR

3.1 INTRODUCTION

This chapter describes the commands supported by the keyboard monitor program. In the keyboard mode, the user enters the commands and data by pressing the appropriate keys on the keypad. Responses are displayed by the system on the six-digit 7-segment LED display.

Whenever the monitor expects a command, the display shows a dash(“-”) at the left edge of the address field, possibly along with an error message or with the sign-on message upon reset. Thus, it should be noted that irrespective of the characters appearing in the rest of the display, a dash at the left edge of the display always is a command prompt. When the monitor expects a parameter, a decimal point will be displayed at the right edge of the field into which the parameter is to be placed. The parameter will be either an address to be entered in the address field or a byte of data to be entered in the data field (The only exception, explained later, occurs in the use of the EXAM REG command). The valid range for an address parameter is from 1 to 4 hexadecimal digits and the valid range for a data parameter is from 1 to 2 hexadecimal digits. Longer numbers may be entered but such numbers are evaluated modulo 64K or 256 respectively, i.e. only the last four or the last two digits entered will be accepted.

The RESET key causes a hardware reset and restarts the monitor. The Monitor displays the sign-on message (-ESA85) across the address and data fields of the display. Now the monitor will be ready to accept the commands from the user. But the monitor does not save the information about the state (register values etc) of any previous user program. However, contents of the user portion of the RAM area are not disturbed by the monitor. User stack pointer will be set to EFFFH.

3.2 RAM USAGE

The System monitor utilizes 512 bytes of RAM, from FE00H to FFFFH as scratch pad area for System Stack and variables. User program should not disturb this area, otherwise the results are unpredictable.

3.3 KEYBOARD AND DISPLAY

As noted already, the display consists of 6 seven- segment LED displays, separated into two fields. The left filed, called the address field consists of 4 digits and the right field called the data filed consists of 2 digits.

The 32 key keyboard consists of the following groups of keys.

a. Hexpad: 16 keys representing the hexadecimal digits 0 through F. In the use of the EXAM REG command, some of these represent register names also, as indicated by the legends on the keytops. This usage is further explained in the description of the EXAM REG command. Further, six of these keys serve the functions of command keys also. The context of operation defines the meaning of the key.

b. Command group: 11 command keys one of which is USER DEFINED. These commands are in addition to the six commands mentioned above.

c. Delimiter group: 3 keys (NEXT, PREV, EXEC) which serve as the delimiters while entering the commands/data.

d. System operation keys: RESET and KBINT keys. RESET key, as already noted causes a hardware reset of the system. KBINT key can be connected to the RST 7.5 input. It’s use is explained in chapter 5

3.4 MONITOR COMMANDS

The keyboard monitor is capable of executing seventeen individual commands, summarized in Table 3.1. Each command is represented by exactly one key with appropriate legend on the keytop. The commands are described in detail in the sections which follow. In both the table and in the individual command descriptions, the following notation is used:

Upper case letters and numbers represent keyboard keys:

[A] indicates that 'A' is an optional entry;

A/B indicates that either 'A' or 'B' must be entered;

[A]* indicates zero or more optional occurrences of 'A';

<B*> indicates a 1 or 2 bytes parameter to be entered by the user.

TABLE 3.1 KEYBOARD MONITOR COMMAND SUMMARY

COMMAND		FUNCTION/FORMAT
EXAMINE/MODIFY MEMORY	:	Displays/Modifies the contents of a memory location. EXAM MEM <address> NEXT [[Data] NEXT/PREV]* EXEC.
EXAMINE/MODIFY REGISTER	:	Displays/Modifies 8085 register contents. EXAM REG <Reg key> [[Data] NEXT/PREV]* EXEC
SINGLE STEP	:	Executes a single user program instruction SINGLE STEP [<Start addr] NEXT [[Start addr] NEXT]* EXEC.
GO	:	Transfers control from monitor to user program Go [<addr>]Next [<breakpoint 1>] Next [<breakpoint 2>] Next [<breakpoint 3>] Next [<breakpoint 4>] EXEC

COMMAND		FUNCTION/FORMAT	
BLOCK MOVE	:	Moves a block of data from one portion to another	
		BLK MOVE	<start addr>
		NEXT	<end addr>
		NEXT	<dest addr>
		EXEC	
INSERT	:	Inserts one or more instructions in the user program	
		INSERT	[<Low Limit>]
		NEXT	[<High Limit>]
		NEXT	[<Low insert addr>]
		NEXT	<No. of bytes>
		NEXT	[<Data>]
		EXEC	
DELETE	:	Deletes one or more instructions in the user program.	
		DELETE	[Low Limit]
		NEXT	[High Limit]
		NEXT	<Low delete addr>
		NEXT	<High Delete addr>
		EXEC	
INPUT BYTE	:	Inputs a byte from the specified port	
		INBYTE	<Port address>
		NEXT	[NEXT]*
		EXEC	
OUTPUT BYTE	:	Outputs A Byte to the Specified Port	
		OUT BYTE	<Port address>
		NEXT	<Data>
		NEXT	[<data>]*
		EXEC	
COMPARE MEMORY	:	Compares two blocks of memory	
		COMP	<Start address>
		NEXT	<end address>
		NEXT	<dest addr>
		EXEC	

COMMAND		FUNCTION/FORMAT
FILL MEMORY	:	Fills a block of memory with a byte constant FILL <Start address> NEXT <end address> NEXT <data> EXEC
PROG	:	Programs an EPROM. Refer chapter 9
BLNK	:	Blank checks an EPROM. Refer chapter 9
VRFY	:	Verifies an EPROM. Refer chapter 9
PRRD	:	Reads an EPROM. Refer chapter 9
TPWR	:	Saves data on audio tape. Refer chapter 7
TPRD	:	Loads data from audio tape. Refer chapter 7

3.4.1. EXAMINE/MODIFY MEMORY COMMAND

FUNCTION

This command is used to examine the contents of selected memory locations. The contents can be optionally modified if the memory location is in RAM area.

FORMAT

EXAM MEM <address> Next [[data] NEXT/PREV] * EXEC

OPERATION:

1. To use this command, press the EXAM MEM key when prompted for a command. When this key is pressed, the display is cleared and a dot appears at the right edge of the address field indicating that an address entry is required.
2. Enter the memory address of the byte to be examined. (As already noted, memory addresses are evaluated modulo 64K). This value is displayed in the address field of the display.
3. After entering the address value, press the NEXT key. The data byte at the addressed memory location will be displayed in the data field and a decimal point (a dot) appears at the right edge of the data field indicating that data can be updated.

4. If the contents of the address memory location are only to be examined, press the EXEC key to terminate the command, or press the NEXT key to examine/modify the next consecutive memory location or the PREV key to examine/modify the previous memory location.
5. To modify the contents of an addressed memory location, enter the new data from the key board (note that data values are evaluated modulo 256). However, this new value, displayed in the data field is not entered into the addressed memory location till NEXT or EXEC or PREV is pressed.

ERROR CONDITIONS:

Error conditions occurs while attempting to modify a non-existent or Read-Only Memory (ROM) location. Note that the error is not detected until the PREV or NEXT or EXEC key is pressed. When an error is detected, the characters 'Err' are displayed along with the command prompt character (-) in the address field.

EXAMPLES:

Example 1: Examine a series of memory locations starting from 000H (the start of keyboard Monitor).

Key Pressed	**Display**		**Comments**
	Address Field	**Data Field**	
RESET	-ESA	85	System Reset
EXAM MEM			Examine Memory Command.
0	0000.		First memory location to be examined 0000H.
NEXT	0000	F3.	Contents of this location.
NEXT	0001	3E.	Next location and its contents.
NEXT	0002	0F.	Next location and its contents.
PREV	0001	3E.	Previous locations, its contents.
EXEC	-		Command termination/prompt.

Example 2 : Examine and modify the contents of memory location 8D00H

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	System Reset
EXAM MEM			Examine Memory Command
8	0008.		
D	008D.		Memory location to be examined & modified.
0	08D0.		
0	8D00.		
NEXT	8D00	XX.	Contents of this location.
A	8D00	0A.	New data to be entered
F	8D00	AF.	
EXEC	-		Command termination prompt after updating the data.

To check that data was updated successfully, press EXAM MEM key and enter memory address 8D00H and note that "AF" is displayed in the data field when NEXT key is pressed.

Example 3: Trying to modify ROM location.

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	System Reset
EXAM MEM	.		Examine memory Command.
A	000A.		Address of location to be examined.
NEXT	000A	10.	Contents of this location.
F	000A	0F.	
A	000A	FA.	New data to be entered
NEXT	-Err		Error message

You tried to modify the contents of a Read only Memory location. Hence, the error message along with command prompt character was displayed. You can repeat the above sequence of keys to see that the contents of this location are unaltered.

3.4.2. EXAMINE/MODIFY REGISTER COMMAND

FUNCTION

The examine Register Command is used to examine and optionally modify the contents of any of the 8085’s registers.

FORMAT

EXAM REG <reg key> [<data>] NEXT/PREV* EXEC

OPERATION

1. To use this command, press the EXAM REG key when prompted for a command. Now, the display is cleared and a decimal point appears at the right edge of the address field. However, unlike in EXAM MEM command, this prompt now means that a register name entry is required. Thus the next hexadecimal key board entry will be interpreted as a register name.

TABLE 3.2 PROCESSOR REGISTERS

Register Name	Register identifier key	Display Abbreviation
Register A	A	A
Register B	B	B
Register C	C	C
Register D	D	D
Register E	E	E
Flags Register	F	F
Interrupt Mask	3/I	I
Register H	8/H	H
Register L	D/L	L
Stack pointer High byte	4/SPH	SPH
Stack pointer Low byte	5/SPL	SPL
Program Counter High byte	6/PCH	PCH
Program Counter Low byte	7/PCL	PCL

FORMAT OF INTERRUPT MASK I:

0	0	0	IE	M7.5	M6.5	M6.5

^ Interrupt Masks

^ Interrupt Enable Flag

FORMAT OF THE FLAG BYTE F:

S	Z	X	AC	X	P	X	C
Sign	Zero		Auxiliary Carry		Parity		Carry

Fig 3.1 : Format of I and F Registers

2. When the hexadecimal key is pressed, the corresponding register abbreviation is displayed in the address field and the contents of this register are displayed in the data field and a data update prompt (dot) appears at the right edge of the data field.

   Table 3.2 defines the 8085 register names, the hexadecimal keyboard acronyms, the abbreviations appearing in the address field of the display and the sequence in which the registers are displayed. The formats of the flag byte F and interrupt mask I are shown in Figure 3.1.

3. When the register contents are displayed (with the data update prompt), the contents of this register can be modified if desired. To do this, enter the new value from the keyboard. This new value will be displayed in the data field and the register contents are updated when either the PREV or NEXT or EXEC key is pressed.

4. After examining and optionally modifying the contents, if the EXEC key is pressed, the command is terminated. If the NEXT key is pressed, the above abbreviation and contents of the "next register" (according to the order shown in Table 3.2) are displayed and opened for modification. Note that the sequence is not cyclic and thus pressing the NEXT key when the PCL is displayed will terminate the command. If the PREV key is pressed, the abbreviation and contents of the "previous register" (according to the order shown in Table 3.2) are displayed and opened for modification.

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	System Reset
EXAM MEM	.		Examine memory Command.
8/H	H	xx.	Contents of Register
8	H	08	New Data
NEXT	L	xx.	H Register updated
			Next Register's contents displayed
EXEC	.		Command termination/prompt.

3.4.3 INPUT BYTE COMMAND

FUNCTION.

The INPUT BYTE command is used to input (accept) a byte of data from an input port.

FORMAT

IN BYTE <port address> NEXT [NEXT] * EXEC

OPERATION:

1. To use this command, press the IN BYTE key when prompted for command. When this key is pressed, the display is cleared and a decimal point appears at the right edge of the address field indicating that an address entry is required.
2. Enter the address of the port to be read. Note that the port address can be in the range 0-255 (decimal) only. Thus only 2 hex digits are required to specify the port address. If longer value is entered, the last four digits entered are displayed until the NEXT key is pressed. Then only the last two digits are accepted as the port address. In any case, this port address is duplicated in the most significant two digits of the address field of the display. After entering the port address, press the NEXT key. The addressed port is read and the data is displayed in the data field of the display.

3. Pressing the NEXT key again, updates the data field display with the current data byte at the addressed input port. Pressing the EXEC key terminated the command and the command entry prompt (‘-’) appear.

NOTE:

The I/O ports provided on ESA 85-2, their addresses and usage are described in detail in chapter 5.

Example:

Input the byte from the port 70H (i.e. baud rate switch status)]

Key Pressed	**Display**		**Comments**
	Address Field	**Data Field**	
RESET	-ESA	85	System Reset
IN BYTE	.		Input Byte Command
7	0007.		Port Address
0	0070.		
NEXT	7070	xx	Input Data Byte
NEXT	7070	xx	Current Data Byte
NEXT	7070	xx	Current Data Byte
EXEC	-		Command termination/prompt

3.4.4. OUTPUT BYTE COMMAND

FUNCTION

The OUTPUT BYTE command is used to output a byte of data to an output port.

FORMAT

OUTBYTE<port address> NEXT<data> [NEXT <data>] * EXEC

OPERATION

1. TO use this command, press the OUTBYTE key when prompted for command. When this key is pressed, the display is clear and a decimal point appears at the right edge of the address field indicating that an address entry is required.
2. Enter the desired port address. Note that the port address can be in the range 0-255 (decimal) only. Thus only 2 hex digits are required to specify the port address. If longer value is entered, the last four digits entered are displayed until the NEXT key is pressed. Then only the last two digits are accepted as the port address. In any case, this port address is duplicated in the most significant two digits of the address field of the display. After entering the desired port address, press the NEXT key.
3. Now a decimal point appears at the right edge of the data field indicating that the data byte to be output can now be entered using the hexadecimal keyboard; enter the data byte to be output.
4. After entering the data, press the EXEC key to output the byte to the port and to terminate the command, or press the NEXT key if additional data is to be output to the addressed port.

NOTE: As mentioned in the previous section, the I/O ports provided on ESA85-2, their addresses and usage are explained in detail, in chapter 5.

Example: Output a string of ASCII characters (ESA) to the data port of the USART (8251) to be transmitted to a CRT terminal through the on board RS 232 C interface.

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	System Reset
OUT BYTE	.		Output Byte Command
2	0002.		
0	0020.		Port address
NEXT	2020	.	
4	2020	04.	
5	2020	45.	Send ‘E’
NEXT	2020	.	
5	2020	05.	
3	2020	53.	Send ‘S’
NEXT	2020	.	.
4	2020	04	
1	2020	41	Send ‘A’
EXEC	-		Command termination/prompt.

3.4.5 GO COMMAND

FUNCTION

The GO command is used to transfer control of the system from the monitor to user’s program with optional breakpoints.

FORMAT

```
Go, [<address>],               NEXT [<breakpoint 1>]
                               NEXT [<breakpoint 2>]
                               NEXT [<breakpoint 3>]
                               NEXT [<breakpoint 4>]EXEC
```

OPERATION

1. To use this command, press the GO key when prompted for command entry. When this key is pressed, the current contents of the User Program Counter are displayed in the address field and the contents of the location addressed by the PC are displayed in the data field. A dot also appears at the right edge of the address field indicating that the user can update the value of User Program Counter if required..

2. To begin Program execution, press EXEC key. Now the address and data fields are cleared, and “E” is displayed at the left edge of the address field and control is then transferred to the user program at the current value of the User program counter.

3. To return control to the monitor, you can write the RST3 instruction (opcode, DFH) at the end of your program. When this instruction is executed, Monitor regains control of the system and full information about the user program is saved. Another way to exist from a running program and return control to the monitor is to press RESET key. However, in this case, all register information about user program is lost. Note that in any case the contents of the user portion of the RAM area are not disturbed by the Monitor.

4. Another method of breaking the program execution at a specific address is to set breakpoints. After optionally modifying the user PC, instead of pressing EXEC, user can now press NEXT key and enter a breakpoint address. User can now press EXEC or press NEXT key and enter another breakpoint address. Like this user can enter a maximum of four breakpoints and then press EXEC key to initiate program execution.

The monitor now transfers control to user program. When any of the specified breakpoints is encountered, control returns to the monitor which will then save the user context and issue the command prompt.

NOTES:

1. Breakpoints are implemented by replacing the instructions at the breakpoint addresses with RST 3 instructions. Upon encountering any one breakpoint, the instructions at all the breakpoint addresses are replaced with the original user instructions.
2. From the above description, it is clear that breakpoints can be set only in RAM.
3. If none of the breakpoints are encountered and user regains control via REST key, then the monitor cannot restore the original user instructions!

ERROR CONDITIONS

1. Specifying a breakpoint in PROM area
2. Specifying more than four breakpoints.

Examples:

Example 1: Suppose the following program has been entered in the memory by EXAM MEM command.

Location	Object Code	Mnemonic
8800	3E	MVI A,42
8801	42	
8802	4f	MOV C,A
8803	DF	RST 3

To run this program, press the keys according to the following sequence.

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	System Reset
GO	xxxx.	xx	GO command; current PC and the byte at this PC are displayed.
8	0008.		New starting address
8	0088.		
0	0880.		
0	8800.		

Key Pressed	Display		Comments
	Address Field	Data Field	
EXEC	-		Control transferred to the program at 8800H. Control returned to the monitor through the execution of RST3 Instruction.

Example 2: Transfer control to program assembled from the location 8000H with a breakpoint at 80E0H.

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	System Reset
GO	xxxx.	xx	GO command; Current PC and the byte at this PC are displayed.
8	0008.		New value of the PC
0	0080.		
0	0800.		
0	8000.		
NEXT	.		Enter breakpoint address.
8	0008.		
0	0080.		
E	080E.		
0	80E0.		
EXEC	-		Control returns because of break point.

3.4.6. SINGLE STEP COMMAND

FUNCTION

This command is used to execute a program, one instruction at a time. With each instruction executed, control is returned to the monitor. Thus this command is an extremely useful debugging tool.

FORMAT

SINGLE STEP [<start address>] NEXT[[<start address>]NEXT]* EXEC

OPERATION

1. To use this command, press the SINGLE STEP key when prompted for command. Now the contents of the user program counter are displayed in the address field and the byte at this location is displayed in the data field. A dot also appears at the right edge of the address field indicating that the user can modify the value of the program counter if desired, by entering the new address.
2. To execute the one instruction at the current value of the program counter, press the NEXT key. When this key is pressed, the instruction at the displayed address is executed, and the new value of user PC is displayed in address field and its associated instruction byte is displayed in the data field. The contents of the PC are again opened for optional modification by the user.
3. To terminate command, press the EXEC key. (Note that after terminating the SINGLE STEP Command, if you again press the SINGLE STEP key, control returns exactly to the point where you pressed the EXEC key).

EXAMPLES

Example 1. Suppose the program given as Example 1 to illustrate the GO Command has been entered in the memory. Now this program can be single-stepped as follows.

Key Pressed	**Display**		**Comments**
	Address Field	**Data Field**	
RESET	-ESA	85	System Reset
SINGLE STEP	xxxx.	xx	Current value of PC and the byte at this PC
8	0008.		
8	0088.		
0	0880.		New value of PC
0	8800.		
NEXT	8802.	4F	Instruction at 8800 is executed. Instruction to be executed is displayed.
NEXT	8803	DF	Next instruction
EXEC	-		Command termination

3.4.7 BLOCK MOVE COMMAND

FUNCTION:

This command is used to move a block of data from one area of the memory to another area.

FORMAT:

BLK MOVE <start address> NEXT <end address> NEXT <destination address> EXEC

OPERATION:

1. To use this command, press the BLK MOVE key when prompted for command entry. When this key is pressed, display field is cleared and decimal point appears at the right edge of the address field.
2. New enter the starting address of the block of data to be moved. Press the NEXT key. Now the display field is again cleared and again a decimal point appears at the right edge of the address field. Now enter the ending address of the block of data to be moved. Press the NEXT key. Monitor clears the display field and a decimal point appears at the right edge of the address field. Now enter the starting address of the area into which the block of data is to be moved. This address is called the destination address. Press the EXEC key to start the command execution.
3. Monitor now moves the block of data from Start Address to End Address, to the area beginning at the Destination Address. After moving the data monitor displays the command prompt sign.

ERROR CONDITIONS:

1. Specifying a value for the end address of the source block which is less than the value of the start address of the source block.
2. Trying to move the data into non-existent or Read-only Memory.

EXAMPLES:

Example 1: Moving the block of program code listed in the example for the GO command.

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	System Reset
BLK MOVE	.		Block move command
8	0008.		Start address of source block
8	0088.		
0	0880.		

Key Pressed	Display		Comments
	Address Field	Data Field	
0	8800.		
NEXT	.		
8	0008.		End address of source block.
8	0088.		
0	0880.		
3	8803.		
NEXT	.		
8	0008.	Destination address	
8	0088.		
2	0882.		
0	8820.		
EXEC	-		Block moved. Command prompt.

Now using the EXAM MEM key observe the contents of the locations 8820H, 8821H, 8822H and 8823H and verify that the code has indeed been moved into the destination block.

Example 2:

Trying to move into PROM area

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	System Reset
BLK MOVE	.		Block Move command
0	0000.		start address
NEXT	.		
F	000F.		End address
NEXT	.		
2	0002.		Destination address
0	0020.		
EXEC	-Err		Attempt to move data into PROM area resulted in error message. Command prompt.

Notes:

BLOCK MOVE, moves a block from one area to another. It does not consider whether the block being moved consists of program code or data. Thus no relocation of program code occurs and the block is simply moved.

The system determines if the destination block overlaps the source block. It moves the data from either the starting address or from the ending address as required when overlap exists.

3.4.8 INSERT COMMAND

FUNCTION

This command is used to insert one or more instructions into the user program.

FORMAT

INSERT [<Low-Limit>] NEXT [[<High-Limit>] Next <Low Insert Address> Next <No. of bytes> NEXT [<data>]* EXEC

OPERATION

This command together with the next command to be described viz. DELETE, provides the user mini-editing facilities.

INSERT command allows the user to insert the desired number of bytes into a program at a specified address. The address references in the program, which change became of this insertion, are automatically corrected by the monitor. To do such corrections and to relocate the program, the monitor must know

*1. Low Limit: The starting address of the user program

*2. High Limit: Present ending address of the user program.

*3. Low-Insert Address: The address from where the bytes are to be inserted.

*4. No. of bytes: The number of bytes to be inserted (in hexadecimal notation and

*5. The actual bytes to be inserted.

a) To use this command, press the INSERT key when prompted for command entry. When this key is pressed, “Low Limit” (i.e. the starting address of the program as remembered by the monitor from previous operations) is displayed in the address field. This value can be changed by the user by entering a new starting address and then pressing the NEXT key. If the value displayed is not to be changed, simply press the NEXT key.

b) Now the monitor displays the assumed High Limit (Present end of the user program). To change this value, enter the new value and press NEXT key. Otherwise, simply press the NEXT key.

c) Now the monitor clears the display and a dot appears at the right edge of the address field indicating that an address entry is required. Now enter the "Low Insert Address" (i.e. the address starting from which insertions are to be made) and press the NEXT key.

d) Again the display is cleared and a dot appears at the right edge of the address field. This time, enter the number of bytes to be inserted, in hexadecimal notation, followed by the NEXT key.

e) The monitor now displays the "Low Insert Address" in the address field and a dot appears at the right edge of the data field prompting a data entry. Enter a byte value to be inserted. Press the NEXT key if more bytes are to be entered. After entering the last byte, or after entering a byte followed by the EXEC key, the command is terminated and a command prompt appears at the left edge of the display. Thus this step is similar to EXAM MEM command operation.

The monitor inserts the new bytes into the user program and makes appropriate changes to the address references that are to be changed because of the insertion.

NOTES:

1. The monitor treats the entire program segment from LOW-LIMIT to HIGH-LIMIT as relocatable and adjusts all memory reference. Hence, if the inserted program fragment refers to locations within itself, then care must be taken to see that "No. of bytes" is subtracted from all such references before entering them.
2. As the size of the program grows, when a number of bytes are inserted, the user should ensure that there is sufficient free memory available beyond the current end address of the user program.
3. Relocation is explained in greater detail in chapter 14. on Programming Examples.

ERROR CONDITIONS

1. Specifying a "Low Insert Address" value that is greater than the value of the "High Limit".
2. Trying to insert instruction in non-existent or Read-Only Memory.

Example :

Example on the usage of INSERT command is provided in the next section, after describing the DELETE Command.

3.4.9. DELETE COMMAND

FUNCTION

This command is used to delete one or more instructions from the user program.

FORMAT

DELETE [<Low Limit>] NEXT [<High Limit>]NEXT <Low Delete Address> NEXT <High Delete Address> EXEC

OPERATION

This command allows the user to delete one or more instructions from a program. The address references to be changed because of this deletions process, are changed automatically by the monitor. The interpretation of the parameters appearing in the format of the command is same as in the INSERT command, described in the previous section.

1. To use this command, press the DELETE key when prompted for command entry. Now the LOW-LIMIT is displayed in the address field. The LOW-LIMIT can optionally be changed by the user. Then the user must press the NEXT key. Now the monitor displays the HIGH-LIMIT. After modifying this if desired, press the NEXT key.
2. Now the display is cleared and a dot appears at the right edge of address field indicating that an address entry is required. Enter the "LOW DELETE ADDRESS" (i.e. the address starting from which instructions are to be removed) and press the NEXT key.
3. Again the display is cleared and a dot is displayed at the right edge of the address field. Enter the "HIGH DELETE ADDRESS" (i.e. the address of the last byte to be removed from the user program) and press the EXEC key.
4. Monitor deletes all the bytes from "LOW DELETE ADDRESS" to "HIGH DELETE ADDRESS". It then makes appropriate changes to all address reference in the program from "LOW LIMIT" to "HIGH LIMIT">

ERROR CONDITIONS

Specifying a "LOW DELETE ADDRESS" which is greater than "HIGH LIMIT" or "HIGH DELETE ADDRESS".

EXAMPLE: This example illustrates the use of INSERT and DELETE commands.

Assume that the following program has already been entered into the memory using the EXAM MEM Command.

LOCATION	CONTENTS	LABEL	INSTRUCTION
8830	C5		PUSH B
8831	0B	DELAY:	DCX B
8832	78	MOV A,B	
8833	B1	ORA C	
8834	C2	JNZ DELAY	
8835	31		
8836	88		
8837	C1	POP B	
8838	D1	POP D	
8839	E1	POPH	
883A	C9	RET	

Now, suppose, you discovered that you did not initialise BC register pair before starting the DELAY. You can insert one instruction, say LXI B, 3030H starting at the location 8831 by the following key sequence.

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	System Reset
INSERT	xxxx.		Current Low-Limit
8	0008.		The LOW LIMIT of the user program.
8	0088.		
3	0883.		
0	8830.		
NEXT	xxxx.		Current High-Limit
8	0008.		The HIGH-LIMIT of the program.
8	0088.		
3	0883.		
A	883A.		

Key Pressed	Display		Comments
	Address Field	Data Field	
NEXT	.		
8	0008		Low insert address
8	0088.		
3	0883.		
1	8831.		
NEXT	.		
3	0003.		Insert one instruction of 3 bytes
NEXT	8831	xx.	Start entering the instruction.
1	8831	01	
NEXT	8832	xx.	
3	8832	03.	
0	8832	30.	
NEXT	8833	xx.	
3	8833	03.	
0	8833	30.	
EXEC	-		Instruction inserted, Return to command Prompt

To see that the instruction has indeed been inserted, you can use the EXAM MEM command to observe the contents of the locations 8830H to 883DH (883A+3=883D). Also note that the address reference for the JNZ instruction has changed from 8831 to 8834.

After inserting the instruction and after examining the program, suppose you discover that you have two extra instruction POP D and POP H. You can delete these instructions by the following key sequence.

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	System Reset
DELETE	8830.		Current Low-Limit. You do not want to change it. So press NEXT.
NEXT	883A.		Current HIGH-LIMIT
8	0008.		New HIGH LIMIT
8	0088.		

Key Pressed	Display Address Field	Display Data Field	Comments
3	0883.		
D	883D.		
NEXT			Enter Low DELETE ADDRESS. Note that POP D is now at 883BH and not at 8838H.
8	0008.		
8	0088.		
3	0883.		
B	883B.		
NEXT	.		Enter HIGH DELETE ADDRESS
8	0008.		
8	0088.		
3	0883.		
C	883C.		
EXEC	-		The two bytes are deleted. Command prompt appears.

To check operation, use EXAM MEM key to observe the contents location 8830H to 883BH. They should be C5, 01, 30, 30, 0B, 78 B1, C2, 34, 88, C1, C9.

3.4.10 COMPARE COMMAND

FUNCTION

This command is used to compare two blocks of memory.

FORMAT

COMP <start address> NEXT <end address> NEXT

<destination address> EXEC

OPERATION

1. To use this command, press the COMP key when prompted for command entry. When this key is pressed, display filed is cleared and a decimal point appears at the right edge of the address field.

2. Now enter the starting address of the block of data to be compared. Press the NEXT key. Now the display field is again cleared and a decimal point appears at the right edge of the address field. Enter the ending address of the block of data to be compared and press the NEXT key. This display filed is cleared and a decimal point appears at the right edge of the address field. Now enter the starting address of the area with which the specified block of data is to be compared. This address is called the destination address. Press the EXEC key to start the command execution.

3. The monitor now compares the data in the specified block against the data in the destination block. If no mismatch is detected in the entire address range specified, the command is terminated and the command prompt is displayed.

If any mismatch is detected, the source address and data are displayed and the system waits for user's input. If user presses EXEC key, the command is terminated and the command prompt is displayed. If user presses the NEXT key, the system proceeds with the comparison from the next location.

ERROR CONDITIONS

1. Specifying a value for the end address of the source block which is less than the value of the start address of the source block.

EXAMPLES:

Example 1: Using Examine Memory command, enter the following data into memory locations 8800H to 8807H.

8800: 00, 02, 03, 04, 05, 06, 07

Now use the Block Move command to move this data into the destination block starting at 8810H. Now compare these two blocks of memory.

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	System Reset
COMP	.	85	Compare command prompt for source start.
8	0088.		
8	0088.		

Key Pressed	Display		Comments
	Address Field	Data Field	
0	0880.		
0	8800.		
NEXT	.		Prompt for source end.
8	0008.		
8	0088.		
0	0880		
7	8807.		
NEXT	.		Prompt for destination start.
8	0008.		
8	0088.		
1	0081.		
0	8810.		
EXEC	-		Match. Command termination/ command prompt.

Now using Examine Memory command, alter the contents of location 8805H to 55H and issue the compare command again. The key sequence will be exactly same as described already. Hence only the interaction after pressing EXEC key is shown below:

Key Pressed	Display		Comments
	Address Field	Data Field	
EXEC	8805	55	Mismatch. waiting for user input
NEXT	-		continue comparison, command termination/command prompt.

3.4.11 FILL COMMAND

FUNCTION

This command is used to fill a block of memory with a constant.

FORMAT

FILL <start address> NEXT <end address> NEXT <data> EXEC

OPERATION:

1. To use this command, press the FILL key when prompted for command entry. When this key is pressed, display field is cleared and decimal point appears at the right edge of the address field.

2. Now enter the starting address of the block of memory to be filed. Press the NEXT key. Now the display field is again cleared and again a decimal point appears at the right edge of the address field. Now enter the ending address of the block. Press the NEXT key. Monitor clears the display field and a decimal point appears at the right edge of the address field. Now enter the constant. Press the EXEC key to start the command execution.

3. Monitor now fills the block of memory from Start Address to End Address with the specified constant. Then the monitor displays the command prompt sign.

ERROR CONDITIONS:

1. Specifying a value for the end address of the source block which is less than the value of the start address of the source block.

2. Trying to fill the data in non-existent or Read-only Memory.

EXAMPLES:

Example 1: Filling the block of memory from 8800H to 880FH with a constant 55H.

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	System Reset
FILL	.		Fill command
8	0008.		Start address
8	0088.		
0	0880.		
0	8800.		
NEXT	.		
8	0008.		End address of block
8	0088.		
0	0880.		
F	880F.		

Key Pressed	Display		Comments
	Address Field	Data Field	
NEXT	.		
5	0005.		Constant value
5	0055.		
EXEC	-		Block filled. Command prompt.

Now using the EXAM MEM key observe the contents of the locations 8800H to 880FH and observe the data to be 55H.

3.5 USER DEFINED FUNCTION:

ESA85-2 System has one command key whose functions can be defined by the user. This key labeled F1 is in addition to the standard command keys described already.

When the User-defined function key is pressed, monitor transfers control to a prespecified location in RAM, FE15H.

Following Power on/Rest, the monitor initializes this area with the instruction JMP Err.

Thus after Power on/Reset, if you press the function key, Error message along with the command prompt (-Err) is displayed.

MAKING USE OF THE FUNCTION KEY

To use the user-defined function key, the user must write appropriate instructions in the locations reserved for the user function key. (These instructions can be written using the EXAM MEM command). Note that as only 3 byes (Starting from FE15H) are reserved for F1 key, the user instruction will generally be a JMP instruction.

EXAMPLE

Assume that the user has written the software for programming a specific type of PROM. Assume that this program is assembled from location 4000H (expansion ROM area). One way to invoke this routine would be to use the GO command to transfer control to this program. The second way is to use the function key. Assume that the user has decided to use the key F1. Thus, when F1 is pressed, the program at 4000H should gain control. To do this, user can set up a JMP instruction at FE15H as follows:

Location	Byte value
FE15H	C3
FE16H	00
FE17H	40

Now, whenever the F1 key is pressed, the PROM programming routine at 4000H gets control of the system.

NOTES:

1. If he user inadvertently uses the locations reserved for the user function key, the results of pressing the user defined function key are unpredictable.
2. Note that the monitor initializes the location reserved for the key F1 with the JMP Err instruction, following either power ON or RESET. Thus if the user writes his/her instructions in these reserved locations, and subsequently presses the RESET key, then his/her instructions will be overwritten.

CHAPTER 4

SERIAL MONITOR

4.1 INTRODUCTION

This chapter describes the commands supported by the Serial Monitor Program. The Serial Monitor allows ESA85-2 to be operated from a CRT terminal / Host computer connected through RS 232C serial interface (Refer to chapter 5 on Hardware and Appendix E on RS 232 C connector Details).

The system must be configured for serial mode of operation as described in section 2.2.1. The PROM Programmer System commands (P,R,V,B), Audio Tape Interface Commands (L.W).Software Development Commands (E,A,Z) are described in later chapters. The remaining commands are described in this chapter.

When the system enters serial mode of operation, the sign-on message "ESA85-2 SERIAL MONITOR V x. y" is displayed (x is the current version number and y is the revision number) on one line and a period "." on the next line indicating that the monitor is ready to accept commands from the user. With the exception of RESET and KBINT keys, the keyboard is completely disabled.

4.2. RAM USAGE

The system monitor utilizes 512 bytes of RAM, from FE00H to FFFFH as scratch pad area for system stack and variables. User programs should not alter this area, otherwise the results are unpredictable.

Further, the Text Editor assumes the RAM from F000H to F9FFH as the default Text Buffer area. The Assembler and Disassembler packages assume the default Label Table to be from FA00H to FDFFH. User should take these assumptions into account while developing his/her programs. (Refer chapters 10, 11 and 12 on Text Editor, Assembler and Disassembler.

4.3 STRUCTURE OF MONITOR COMMANDS

Whenever the monitor is ready to accept a command from the user, it outputs a period (‘.’) as the command prompt character at the beginning of a new line.

The commands entered by the user consists of a single character command mnemonic followed by a list of command parameters. This list may consist of upto four parameters depending on the particular command being used. When more than one parameter is required, a (‘,’) is used between the parameters as a seperator.

A command is terminated either by a Carriage Return or by a comma, depending on the command itself. Commands are executed one at a time and only one command is allowed within one command line.

PARAMETER ENTRY

All numeric parameters are to be entered as hexadecimal numbers. The valid range for one byte parameters is 00 to FF and if more than 2 digits are entered, only the last two digits are valid (leading zeros may be omitted). Thus all one byte values are interpreted modulo 256 (decimal). The valid range for 2-byte parameters is 0000 to FFFF and longer values are evaluated modulo 64K (i.e. only the last four digits are valid).

All the commands except the X (examine/modify register) command require only hexadecimal values as parameters. The register name abbreviation entries required by the X command are described later while describing the X command in detail.

RESPONSE TO ERRORS

Whenever an error is detected by the monitor (either in the command entry or in the command execution) the command is aborted, the symbol (‘?’) is output on the command line, a carriage return and a linefeed are issued and the command prompt character (‘.’) is output at the beginning of a new line. (The possible error conditions are described while illustrating the individual commands.)

Command execution occurs only after a valid delimiter (a comma or a carriage return depending on the command) is entered. Hence a command entry can be cancelled anytime before the delimiter is entered by “committing an error”. That is, enter any character that is not legal for the expected entry. The monitor detects this error, aborts the command, displays `?’ symbol and returns to command entry mode.

4.4. MONITOR COMMANDS

Each command described in this chapter consists of a single character, followed by apropriate parameters and data. These commands are summarized in Table 4.1. and are described in detail in the following sections. In the table as well as in the subsequent descriptions, the following notation is used:

A capital letter indicates a command mnemonic:

[a] indicates an optional parameter ‘a’

a/b indicates either ‘a’ or ‘b’ to be entered

[b]* indicates the nature of a 1 or 2 byte hex value to be enterered by the user as a parameter or data; <CR> indicates a Carriage Return is to be entered.

Further, when describing the individual commands with example, output from the system is underlined.

These symbols are used only to clarify the command formats and they are to be neither entered by the user nor output by the system.

TABLE 4.1 SUMMARY OF SERIAL MONITOR COMMANDS

COMMAND		FUNCTION/FORMAT
A (Assemble)	:	Assembler (Refer Chapter 11)
(Blank check)	:	Blank check an EPROM (Refer Chapter 9)
(Compare Memory)	**:**	Compares 2 blocks of memory <Start address>, <end address>, <destination address> <CR>
D (Display Memory)	**:**	Displays memory contents in line formatted output D <Start address>, [<end address>]<CR>
E (Editor)	:	Editor (Refer Chapter 10))
F (Fill Memory)	:	Fill a block of memory with a constant byte F <start address> , <end address>, <constant <CR>
G (GO)	**:**	Transfers the processor control from the monitor to user program with optional breakpoints. G [<Start address>] [, <breakpoint address 1>] [, <breakpoint address 2>] [, <breakpoint address 3>] [, <breakpoint address 4>] <CR>
H (Help)	:	Help. List all the commands supported by the Serial Monitor H <CR>
I (Input Byte)	:	Input a byte from an specified port I I <port address >, [,] *<CR>
L (Read Tape)	:	Load data from Tape (Refer Chapter 7)
M (Move Memory)	**:**	Moves a block of memory contents M <Start address> , <end address> <destination address> <CR>

N (Single Step)	:	Execute one instruction of user program N [<start address>], [<start address>,]* <CR>
O (Output Byte)	:	Output a byte to an output port O <port address> <data> [, <data>,] * <CR>
P (Program)	:	Program an EPROM (Refer chapter 9)
R (Read)	:	Read an EPROM (Refer chapter 9)
S (Substitute Memory)	:	Displays/Modifies memory locations S <address> , [[<New Data>], -]* <CR>
V (Verify)	:	Verify an EPROM (Refer chapter 9)
W (Write onto tape) :		Write data onto tape (Refer chapter 7)
X (Examine/Modify Registers)	:	Displays/Modifies the processor registers X [<reg>] [[<new data>],] * <CR>
Z (Disassemble)	:	Disassembler (Refer chapter 12)

4.4.1 S (SUBSTITUTE MEMORY) COMMAND

FUNCTION

The S (Substitute Memory) command is used to examine the contents of specified memory locations. Further, if the locations are in RAM, their contents can be altered if desired.

FORMAT

S <address>, [[new data>], / - /] * <CR>

OPERATION

1. Enter S followed by the address of the memory location to be examined and then enter a comma. The monitor will now output the contents of that location followed by a dash ‘-’. Note that in Serial monitor mode a ‘-’ is always a prompt for data entry, while a “.” is the prompt for command entry.
2. To modify the contents of this location, the user can enter the new value now.
3. Enter a comma, either immediately after the ‘-’ prompt by the system or after the entry of a new value, to examine/modify the next sequential location. A carriage return, instead of the comma terminates the command and returns the monitor to the command entry mode.

ERROR CONDITIONS:

1. Trying to modify the contents of non-existent or PROM locations.

Example 1: Examine the PROM location 11H .

```
S11, D3- <CR>
```

Example 2: Examine a series of RAM locations starting at 8820H and modify the contents of the location 8822H.

```
.S 8820,xx-,
8821 xx-,
8822 xx- 23 <CR> .
```

Example 3: Examine & alter PROM location OOH.

```
.S0,F3- FF, ? .
.
```

When you try to modify the contents of a PROM location, the monitor displays the error sign (a question mark) and returns the command prompt. To see that the contents of the addressed location remain unaltered, you can use the S command again to examine the location 0H.

4.4.2 D (DISPLAY MEMORY) COMMAND

FUNCTION

This command is used to display the contents of a block of memory.

FORMAT

D <Start address> , <end address> <CR>

OPERATION

1. To use this command, enter D when prompted for command entry. After entering D, enter the starting address of the memory block whose contents are to be displayed, then enter a comma, enter the end address of the memory block followed by carriage return.
2. Now the monitor will output the starting address, the contents of the location from this address to the specified end address. The display appears in formatted lines with 16 bytes/line. The number of bytes displayed on the first line are so adjusted that if the second line is present, its first location has address with the last nibble as zero. The ASCII equivalents of the displayed data values are also shown on each line . The non-displayable characters are shown as periods (".").

Example 1: To display the contents of 5 bytes from location 8800H.

```
      .D8800,8804
      00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F              ASCII
8800  41  42  43  44  31                                                          ABCD1 .
```

4.4.3M (MOVE MEMORY) COMMAND

FUNCTION

This command is used to move a block of data from one area of the memory to another area.

FORMAT

M <start address>, <end address>, <destination address>   <CR>

OPERATION

1. To use this command, enter M when prompted for command entry. Follow it with the starting address of the source block to be moved ("start address"), a comma, the ending address of the source block ("end address"), another comma, and then the starting address of the area into which the source block is to be moved ("destination address"). Now enter the carriage return.

This operation moves the contents of memory locations from "start address" to "end address" to consecutive memory locations starting from the "destination address".

The system determines if there is any overlap between source and destination blocks and accordingly tranfers the data beginning either at the "start address" or at the "end address".

ERROR CONDITIONS:

1. Specifying an "end address" value which is less than the value of the "start address".
2. Trying to move data into non-existent or Read Only Memory locations.

EXAMPLES:

Example 1 : Move the contents of the locations 800H through 80FH to the memory block beginning at 8840H.

```
. M 800, 80F, 8840 <CR>

.
```

Example 2 : Move the contents of the locations 800H through 80FH to the memory block beginning at 200H.

```
. M 800, 80F, 200 ?

.
```

An attempt to move data into PROM locations produces the error message.

4.4.4. F (FILL MEMORY) COMMAND

FUNCTION :

This command is used to fill a block of memory with a specified constant.

FORMAT :

F <Start address>, <end address>, <constant> <CR>

OPERATION :

1. To Use this command enter F when prompted for command entry.
2. Now enter the starting address of the block of memory to be filled. Enter a comma. Now enter the ending address of the block. Again enter a comma. Now enter the constant. Press the Cariage Return key to start the command execution.
3. Monitor now fills the block of memory form start address to end address with the speficied constant Then the monitor displays the command prompt sign.

ERROR CONDITIONS :

1. Specifying a value for the end address of the source block which is less than the value of the start address of the source block.
2. Trying to fill the data in non-existent or Read -only memory.

Examples :

Example 1 : Filling the block of memory with a constant 55H

Now you can use the D command to examine the block of memory to see that it is filled with the constant 55H.

4.4.5 C (COMPARE) COMMAND

FUNCTION

Compare command can be used to compare the contents of one memory block with the contents of another memory block.

FORMAT

C <start address > , <end address >   <Destination address > <MS><CR>

OPERATION

1. To use this command, enter C when prompted for command entry. Then enter starting address of the first block, a comma, ending address of the first block, another comma and then the starting address of the second block followed by the carriage return.

2. The monitor now compares the contents of location beginning at start address of block1 with the contents of location beginning at start address of block2. This process continues till the contents of end address are compared with those of the corresponding location in the 2nd block. Any differences detected are displayed.

EXAMPLES:

1. Compare the contents of memory locations 8000H to 8FFFH with those of a memory block begining at 9000H.

```
C 8000, 8FFF, 9000 <CR>
```

. (This response showed that there is no mismatch)

2. Compare the contents of memory locations A000H to AFFFH with those of a memory block beginning at 8000H

```
. C A000, AFFF, 8000 <CR>
ABC0=00    8BC0=FF
AED8=48    8ED8-54
```

. (This response showed that there is mismatch at two locations).

4.4.6 X (EXAMINE/MODIFY REGISTERS) COMMAND

FUNCTION

This command is used to examine and optionally modify the contents of the registers.

FORMAT

```
X [<reg>,]   [[<new data>] , ] <CR>
```

OPERATION

1. To examine the contents of all the registers, enter X followed by carriage return when prompted for command entry. The monitor will now display the contents of all the registers.
2. If you wish to examine/modify the contents of a particular register, then enter X (when prompted for command) followed by the register name abbreviation. The register name abbreviations are shown in Table 4.2. Now the monitor will output an equal sign ('='), the current contents of the specified register and data prompt character ("-"). The contents of this register can be changed now by entering the new data value, followed by a valid terminator (a comma or the Carriage Return). If the terminator is the Carriage Return, the command is terminated. If the terminator is not the carriage return the next "sequential" register is displayed and opened for optional modification. The sequence in which registers are displayed is also shown in Table 4.2. (Note that this sequence is not circular and if a comma is entered after the contents of the "last" register (i.e. PC) are examined/modified, the command is automatically terminated.

TABLE 4.2

Register name	Abbreviation
Accumulator	Ab/
Register B	Bb/
Register C	Cb/
Register D	Db/
Register E	Eb/
Flags Register	Fb/
Interrupt Mask Register	Ib/
Register H	Hb
Register L	Lb/
Memory Pointer	HL
Stack Pointer	SP
Program Counter	PC

Note:

b/ means a single blank character (ASCII code 20H) generated when the SPACE BAR is depressed.

EXAMPLES :

Example 1 : Examine the contents of registers.

```
    . X
A=FF   B=EE   C=00   D=21   E=34   F=A0   I=07   H=06   L=45   HL=0645   SP=8FE0   PC=8825
```

Example 2 : Examine and alter reg C and then examine reg D.

```
    . XC, =52H- 34, D=63H-<CR>

    .
```

4.4.7 I (INPUT BYTE) COMMAND

FUNCTION:

The INPUT BYTE command is used to input (accept) a byte of data from an inpt port.

FORMAT :

I <port address>, [,] * <CR>

OPERATION :

1. To use this command, enter I when prompted for command.

2. Enter the address of the port to be read. Note that the port address can be in range 0-255 (decimal) only. Thus only 2 hex-digits are required to specify the port address. If longer value is entered. only the last two digits are valid as the port address. After entering the port address, enter "," followed by <CR>. The addressed port is read and the data is displayed.
3. Pressing the "," key again the display shows the current data byte at the addressed input port. Pressing the carriage return key terminates the command and the command entry prompt appears.

Notes:

The I/O ports provided on ESA85-2, their addresses and usage are described in detail in chapter 5 on Hardware.

EXAMPLE :

Input the byte from the port 70H (i.e. baud rate switch status)

. I 70, xx, xx <CR>

.

4.4.8 O (OUTPUT BYTE) COMMAND

FUNCTION :

The OUTPUT BYTE command is used to output a byte of data to an output port.

FORMAT :

O <port address>, <data>, [<data>,] * <CR>

OPERATION:

1. To use this command, enter O when prompted for command
2. Enter the desired port address. Note that the port address can be in range 0-255 (decimal) only. Thus only 2 hex-digits are required to specify the port address. If longer value is entered. only the last two digits are valid as the port address. After entering the desired port address, enter "," and then enter the data byte to be output.
3. After entering the data, press the carriage return key to output the byte to the port and to terminate the command, or press the "," key if additional data is to be output to the addressed port.

NOTE:

As mentioned in the previous section, the I/O ports provided on ESA85-2, their addresses, and usage are explained in detail, in chapter 5.

Example : Output 80H to be command port of user 8255.

. 043, 80, <CR>

.

4.4.9 G (GO) COMMAND

FUNCTION :

The GO command is used to transfer the control of the system from monitor to the user's program, with optional breakpoints.

FORMAT :

```
G [Start address],        [, <breakpoint address 1>]
                          [, <breakpoint address 2>]
                          [, <breakpoint address 3>]
                          [, <breakpoint address 4>] <CR>
```

OPERATION :

1. To use this command, enter G when prompted for command entry and then enter <CR>. The monitor will now display the current value of user program counter, the instruction byte stored at this location, and the data entry prompt character ("-").

2. Now, if you wish to modify the value of the PC (i.e. the address to which control is to be transferred), enter the new value followed by carriage return. Now the user context is restored and control is transferred to the program starting at the current value of the user program counter.

Breakpoints:

A powerful debugging tool-Breakpointing a program- is available to the user. To use this facility, enter a comma after optionally modifying the PC, enter a maximum of four breakpoints (seperated by commas) and then enter carriage return.

Now the control is transferred to the program starting at the current PC value. Upon reaching any one of the specified breakpoint addresses, control is returned to the monitor. Monitor saves the complete user context, displays the current PC value and then issues a command prompt.

NOTES:

1. When breakpoint addresses are specified, the monitor saves the instructions shown by breakpoint addresses and replaces them with RST3 instruction. When any one of the breakpoints is reached, control is returned to the monitor, which after saving the registers, replaces all the breakpointed instructions with their original values. Hence, breakpoint addresses must be specified each time a program to be breakpointed is executed.

2. Specifying more than one breakpoint address is useful when debugging a program section containing branch instructions.

Example 1:

Enter the program presented as example 1 for the GO command from the keyboard monitor (section 3.4.5). You can execute it as shown below:

```
. G xxxx=xx - 8800 <CR>

.
```

Example 2:

Transfer control to a user program assembled from location 8000H with breakpoints at 804EH and 8124H.

```
. G xxxx=xx - 8000, 804E, 8124 <CR>

.
```

4.4.10 N (SINGLE STEP) COMMAND

FUNCTION :

The command is used to execute a program one instruction at a time. With each instruction executed, control is returned to the monitor. Thus this command is an extremely useful debugging tool.

FORMAT :

N [<start address>] , [<start address> ,] * <CR>

OPERATION:

1. To use this command, enter N when prompted for command and then enter <CR>. Now the contents of the user program counter are displayed along with the contents of all the user registers. The first byte of the instruction at User PC is also displayed.
2. To execute the one instruction at the current value of the program counter, press the “,” key. To change the PC , enter a new value and then press “,”. When this key is pressed, the instruction at the displayed address is executed and then all the register values are displayed. The contents of PC are again opened for optional modification by the user.
3. To terminate command, press the carriage return key. (Note that after terminating the N command, if if you again enter N, control returns exactly to the point where you pressed the carriage return key).

Examples :

Example 1: Suppose the program given as example 1 to illustrate the GO command has been entered in the memory. Now this program can be single stepped as follows :

```
. N
A=XX  B=XX  C=XX  D =XX  E=XX  F=XX  I=XX  H=XX  L=XX  HL=XXXX  SP = XXXX  PC = XXXX
XXXX=XX=8800,
A=42  B=XX  C=XX  D=XX  E =XX  F=XX  I=XX  H=XX  L=XX  HL=XXXX  SP= XXXX  PC = 8802
8802 = 4F <CR>
```

4.4.11 H (HELP) COMMAND

FUNCTION :

The HELP command is used to list all the commands supported by the serial monitor.

FORMAT :

H

OPERATION :

As soon as H is entered by the user, in response to the command prompt, the system lists in alphabetical order all the commands supported by the serial monitor. The display appears as shown below:

COMMAND SUMMARY

A : ASSEMBLE	B : BLANK CHECK EPROM	C : COMPARE MEMORY
D : DISPLAY MEMORY	E : TEXT EDITOR	F : FILLMEMORY
G : EXECUTE A PROGRAM	H : DISPLAY ALL COMMANDS	I : INPUT FROM A PORT
J : ERROR, RESERVED	K : ERROR RESERVED	L : LOAD FROM THE TAPE
M : BLOCK MOVE MEMORY	N : SINGLE STEP A PROGRAM	O : OUTPUT TO A PORT
P : PROGRAM AN EPROM	Q : ERROR, RESERVED	R : READ FROM AN EPROM
S : SUBSTITUTE MEMORY	T : ERROR, RESERVED	U : ERROR, RESERVED
V : VERIFY AN EPROM	W : WRITE ONTO TAPE	X : ALTER REGISTERS
Y : ERROR, RESERVED	Z : DISASSEMBLE MEMORY	

CHAPTER 5

HARDWARE

5.1 INTRODUCTION

This chapter describes the hardware design details of ESA85-2. Appendix A gives the complete schematics, Appendix B gives the connector details and Appendix C has the component layout diagram. The design details are discussed in the following order:

a) CPU, Address Bus, Data Bus and Control Signals
b) Memory Addressing
c) I/O Addressing
d) Keyboard/Display Interface
e) Programmable Interval Timer and Serial Interface
f) programmable Peripheral Interface Devices.
g) Wait State Logic
h) Interrupts and Hold
i) Bus expansion
j) Connector details

5.2 CPU, ADDRESS, DATA AND CONTROL SIGNALS

ESA85-2 Uses 8085A CPU operated with a 6.144 MHZ crystal. The on-board RESET key can provide a RESETIN* signal to the CPU. Alternatively, the on-board RESET key can be disabled and an external RESETIN* signal provided via the connector P1. This selection is made via the jumper JP7. (Refer section 2.1.5). The RESETOUT from CPU is used to reset rest of the system. Also this signal is available after inversion, on connector P1, for resetting any off-board peripherals.

The clock out from CPU is buffered (by 74 LS 244 at U1) and is available on connector P1. This clock is divided by two (by 74 LS 74 at U11) to provide the peripheral clock PCLK.

The lower address bus is demultiplexed using a 74 LS 373 at U14 and the upper address bus is buffered using 74 LS 244 at U27. The data bus is buffered using a 74 LS 245 at U26. The control signals RD*,WR*,IO/M*,ALE,INTA*,HLDA and RESETOUT are buffered by 74 LS 244 at U1. All these buffered signals are available on the system connector P1. (connector details are given at the end of this chapter.) The "enable" of these buffers is controlled by BHOLDA signals. Thus these buffers are automatically disabled when another bus master gains control (through HOLD signal) to drive these signals.

5.3 MEMORY ADDRESSING

ESA85-2 has three 28-pin JEDEC compatible slots (U5,U6 and U7) for accepting memory devices. The socket at U5 is populated with a 27128 which contains the system firmware. The socket at U7 is populated with a 62256 to provide 32K bytes of static RAM. Memory from FE00H to FFFFH is utilized by the system and the rest is available to the user. The socket at U6 is unpopulated. This socket can be configured to accept 2764/6264/27128/62256 via jumpers JP1,JP2 and JP3. (Refer section 2.1.4). The memory map is as follows:

TABLE 5.1 Memory Map

Device	Address range
27128 at U5	0000-3FFF
62256 at U7	8000-FFFF
2764 at U6	4000-5FFF (6000-7FFF)
6264 at U6	4000-5FFF (6000-7FFF)
27128 at U6	4000-7FFF
62256 at U6	4000-7FFF

Note that when 2764 or 6264 is installed at U6, because of address foldback, the device responds to two address ranges i.e. 4000H to 5FFFH and 6000H to 7FFFH. Also, when 62256 is installed, only the lower 16K bytes are accessible and the upper 16K bytes are unused.

The three chip select signals are derived from BIO/M*, BA15 and BA14. The logic is implemented by 74 LS 32 at U24 and 74 LJ 00 at U25.

Battery Option:

The 62256 provided at U7 can be backed up by an optional battery. The terminals of connecting the battery be brought out as +B and GND.

5.4 I/O ADDRESSING

I/O decoding is implemented using a 74 LS 138 at U23. BIO/M*,BA7,BA6,BA5 and BA4 only are used a derive the chip select signals. Thus foldback exists over the unused address lines. The I/O devices, their addresses and their usages is summarized below:

TABLE 5.2 I/O ADDRESS MAP

I/O Device	Address	Usage
8255A : 1 at U40		
(Programmable Peripheral Interface)		
Port A	00H	Available to user.
Port B	01H	
Port C	02H	The signals are available
Control Port	03H	on connector J1
8255A:2 at U35		
(Programmable Peripheral Interface)		
Port A	40H	Available to user
Port B	41H	The signals are available
Port C	42H	on connector J2
Control Port	43H	
8253-5 at U8		
(Programmable Interval Timer)		
Timer 0	10H	Timer 0 is required for single Step facility.

I/O Device	Address	Usage
Timer 1	11H	Timer 1 is used for baud clock generation.
Timer 2	12H	Timer 2 is available to user. The signals are available on connector P2.
Control Port	13H	
8251A at U2		
(Programmable Communication Interface)		
Data Port	20H	Used for implementing serial.
Command port	21H	Communication
8279-5 at U22		
(Programmable keyboard/Display Interface)		
Data Port	30H	Used for implementing
Command Port	31H	Keyboard/Display interface
8259A at U4		
(Programmable Interrupt Controller)		
Data Port	51H	Available to user. Provides
Command port	56H	support for upto 8 interrupts
8255A:3 at U36		
(Programmable Peripheral Interface)		
Port A	60H	Used for implementing
Port B	61H	Prom programmer system
Port C	62H	
Control Port	63H	
8255:4 at U12		
(Programmable Peripheral Interface)		
Port A	70H	Used for reading the DIP switch,
Port B	71H	and for implementing Parallel
Port C	72H	Printer interface and Audio Tape interface.
Control Port	73H	

5.5. KEYBOARD/DISPLAY INTERFACE

The Keyboard/Display section of ESA85-2 is controlled by 8279.

The port addresses are given in section 5.4. The 8279 is configured for the following operation mode:

* Encoded scan keyboard with 2-key lock out

* 8 digits, 8-bit, left entry display

The keyboard reading is implemented by polling the command/status port of 8279. User can read the keyboard in interrupt driven mode by shorting the jumper JP 12 and by writing suitable RST 5.5 service routine. (Refer section 2.1.5).

The codes assigned to the keys of the on-board keypad are listed below:

TABLE 5.3

SERIAL No. (also the No. shows code in drawings)	**Key**	**Corresponding code**
1	0/FILL	00H
2	1	01H
3	2	02H
4	3/1	03H
5	4/SPH	04H
6	5/SPL	05H
7	6/PCH	06H
8	7/PCL	07H
9	8/H	08H
10	9/L	09H
11	A/PROG	0AH
12	B/BLNK	0BH
13	C/VRFY	0CH
14	D//PRRD	0DH
15	E/TPRD	0EH
16	F/TPWR	0FH
17	F1	10H

SERIAL No. (also the No. shows code in drawings)	Key	Corresponding code
18	BLK MOVE	11H
19	COMP	12H
20	INSERT	13H
21	IN BYTE	14H
22	DELETE	15H
23	OUT BYTE	16H
24	PREV	17H
25	SINGLE STEP	18H
26	GO	19H
27	EXAM MEM	1AH
28	EXAM REG	1BH
29	NEXT	1CH
30	EXEC	1DH

NOTE:

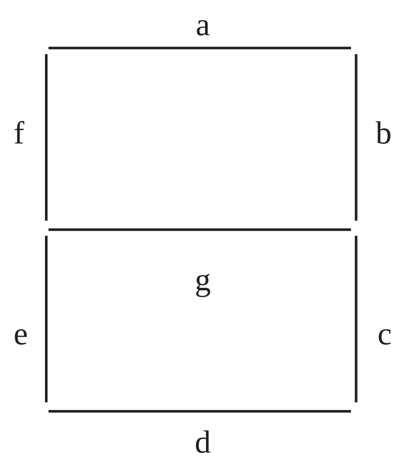
RESET and KBINT keys are not connected to key-board controller.

The signal generated by pressing the RESET key can be used to drive the RESETIN* signal to the CPU, the shorting JP7-A (the default setting.) The signal generated by pressing the KBINT key can be used to drive the RST 7.5 input to the CPU by shorting JP 10. (The default setting leaves JP 10 open.)

Display Drive:

The segment drive outputs of 8279 (A0 through A3 and B0 through B3) from a single 8-bit parallel output driving the display element segments. The correspondence between the data bus, segment outputs of 8279 and the display element segments is shown below:

TABLE 5.4



Display segments

Display segment Control

CPU DATA BUS	D7	D6	D5	D4	D3	D2	D1	D0
8279 OutPut	A3	A2	A1	A0	B3	B2	B1	B0
Segment enabled	d	c	b	a	dp	g	f	e

Bit	= 1 Corresponding segment is	ON
	= 0 Copressponding segment if	OFF

Example:

To display "E", the data format requires that segment a,f,g,e and d should be On and other segments should be OFF. So the data should be 1001 0111=97H. Display codes for other patterns can be worked out similarly.

Display Position:

To display characters in the address field, 8279 must be instructed to start from display position 0 (by sending 90H to the command port of 8279) and to display characters in the data field, 8279 must be instructed to start from display position 4 (by sending 94H to the command port of 8279.). For further details regarding display codes and display procedures, user can refer the listing of Monitor program supplied alongwith ESA85-2.

5.6 PROGRAMMABLE INTERVAL TIMER

ESA85-2 has an on-board programmable interval timer 8253-5 at socket position U8. Its I/O addresses can be found in Table 5.2 in section 5.4.

8253 has one command/status port and three data ports called Timer0, Timer1 and Timer2 to provide three programmable timers. Of these,Timer 0 is utilized to drive the TRAP input of the CPU for implementing the Single Step facility (by shorting JP6-B). SOD output from the CPU controls the gate input of this Timer 0. Timer 1 is utilized to generate Transmit and Receive clocks of 8251A (Programmable communication interface.) however, these signals as well as the signals related to Timer 2 are available on auxillary system connector P2. Thus, if user does not require serial interface, he/she can make use of Timer 1 (in addition to Timer 2, which is always available to user) if required. Connector details are provided in the last section of this chapter.

5.7 SERIAL INTERFACE

As 8251A (Programmable communication interface) at position U2, is used for implementing RS 232 C compatible serial interface. This device is programmed for asynchronous operation, 2 stop bits, no parity, data length of 8 bits and baud scale factor of 16X. As already noted, Timer 1 of 8253 is used to generate the required Transmit and Receive clocks of 8251A based on the setting of the DIP switch. (Refer section 2.1.3 on Baud Rate selection).

The I/O address of 8251 A can be found in Table 5.2 in section 5.4

8251A is operated in a polled mode. However, character input routine can be made interrupt driven by shorting JP11 and writing suitable RST 6.5 service routine. (Refer section 2.1.5)

The TXD,DTR,RTS outputs of 8251A are shifted to RS 232C compatible voltage levels by 1488 at U9 and the RXD,DSR and CTS inputs are shifted to TTL levels required by 8251A by 1489 at U15. The RS 232C compatible signals are available on 25-pin D type female connector J5. (Refer the last section of this chapter for connector details).

NOTE: If the handshake signals are not required, user can short DTR,DSR and RTS,CTS.

5.8 PROGRAMMABLE PERIPHERAL INTERFACE DEVICES

ESA85-2 has four numbers of 8255As (Programmable peripheral interface devices.). Each 8255A consists of a command port and three 8-bit programmable input/output Ports called Port A, port B and Port C. The Port addresses of these devices can be found in Table 5.2 in section 5.4. This section describes the use of these devices.

One 8255A at U36 is used for implementing Prom Programmer System. The system monitor initializes this device for mode 0 operation, with Port A and Port C as output ports and port B as input port.

The 8255A at U12 is used for implementing Audio Tape interface (through port lines PC1 and PC6), Centronics compatible parallel printer interface (through port B, PC0 and PC7) and for reading the DIP switch (through port A). The system monitor initializes this device for mode 0 operation, with Port A and upper part of Port C as input ports and port B and lower part of port C as output ports.

The remaining two 8255As at U40 and U35 are completely available to user. The port signals are available on connectors J1 and J2 (Refer the last section of this chapter for connector details.)

5.9 WAIT STATE LOGIC

ESA85-2 accesses on-board memory and I/O devices with zero wait states. However, the READY input to the CPU is available on system connector P1 and user can drive this signal low to introduce wait states if required.

NOTE: READY is pulled up to VCC through a 4.7 Kohm resistor on ESA85-2.

5.10 INTERRUPTS AND HOLD

ESA85-2 provides an on-board programmable interrupt controller 8259A at U4. The addresses of this device can be found in Table 5.2 in section 5.4. The interrupt output of 8259A is connected to the INTR input of the CPU and the INTA* output of the CPU, (after buffering) is connected to the INTA* input of 8259A. The 8 interrupt inputs to 8259A can be driven by user through auxillary system connector P2. (Refer the last section of this chapter for connector details.)

RST5.5, RST6.5 and RST7.5 inputs can be driven by either on-board signals 8279INT,RXRDY and KBINT or by off-board signals RST5.5, RST6.5, and RST7.5 via the connector P2. (Refer section 2.1.5). The system monitor does not make use of these vectored interrupts. Thus while on the monitor, these interrupts are masked and disabled.

When these vectored interrupts are recognized, the CPU saves the PC and starts execution from specific memory locations. To allow the use of these interrupts by the user, the monitor has JMP instructions at these locations. The target of these JMP instructions is RAM area. For example, when RST 7.5 is recognized, CPU pushes the current PC onto the stack and executes the instruction at 3CH. The instruction at 3CH is JMP FE12H. Thus, now the control comes to FE12H. Three bytes are available to user from FE12H. Thus user should enter another JMP instruction at FE12H (using for example, Examine Memory command) and this JMP must be to the service routine written by the user. The RAM locations corresponding to different RST instructions and vectored interrupts are shown below:

TABLE 5.5 RAM addresses for interrupts.

Interrupt	Corresponding RAM Location
RST 4	FE00H
RST 5	FE03H
RST 5.5	FE06H
RST 6	FE09H
RST 6.5	FE0CH
RST 7	FE0FH
RST 7.5	FE12H

Chapter 14 on Programming Examples includes an example illustrating the use of RST 7.5 interrupt. In a similar way, user can make use of the other interrupts.

TRAP input can be driven either by off-board signal NMIRQ* via connector P1 or by on-board signal STEP (Timer 0 output from 8253 (Refer section 2.1.5). However, ESA85-2 uses this non-maskable interrupt for implementing single-step facility and hence the default jumper setting allows STEP signal to drive TRAP. If user wishes to drive this input from an off-board source, then he/she cannot use single-step facility.

ESA85-2 does not make use of HOLD signal. It is grounded through a 4.7K Ohm resistor. An off-board signal can drive this line via the connector P1. As already noted in section 5.2, HOLDA from CPU disables on-board buffers. Hence user can implement multiprocessor designs (for eg. DMA systems) without any problems.

5.11 BUS EXPANSION

ESA85-2 permits easy expansion of the system by providing all the necessary signals on two connectors, P1 and P2. The signals are STD bus compatible and thus user can easily expand the capabilities of ESA85-2.

5.12 CONNECTOR DETAILS

There are six connectors no ESA85-2 in addition to the power connector J4. Of these P1 is the system connector (providing address, data and control signals) and P2 is the auxillary system connector (providing additional signals, for system expansion). These two are strip connectors. J5, a 25-pin, female D-type connector provides the signals for RS 232 C Compatible serial interface. J3, another 25pin, female D-type connector provides the signals for centronics compatible parallel printer interface. Two Spectra-Strip type 26 pin connectors, J1 and J2 provide the signals for the two 8255As available to user.

The signals definitions on all these connectors are listed below. (This information is available in Appendix B also.

CONNECTORS J1 & J2

PIN NO	8255	PIN	PIN NO	8255	PIN
1.	13	PC4	14.	19	PB1
2.	12	PC5	15.	38	PA6
3.	16	PC2	16.	37	PA7
4.	17	PC3	17.	40	PA4
5.	14	PC0	18.	39	PA5
6.	15	PC1	19.	2	PA2
7.	24	PB6	20.	1	PA3
8.	25	PB7	21.	4	PA0
9.	22	PB4	22.	3	PA1
10.	23	PB5	23.	11	PC6
11.	20	PB2	24.	10	PC7
12.	21	PB3	25.	26	Vcc
13.	18	PB0	26.	7	GND

CONNECTOR J5

PIN NO	FUNCTION
1.	GND
2.	TXD
3.	RXD
4.	RTS
5.	CTS
6.	DSR
7.	GND
20.	DTR

All the remaining pins are not connected.

Note: If your terminal does not support hand-shaking signals, loop RTS & CTS and DSR & DTR. Also remember, ESA85-2 should be connected to terminal RXD and so on.

P1 SYSTEM CONNECTOR (50 PIN BERG STICK)

PIN NO	SIGNAL	PIN NO	SIGNAL
1	+5V	2	+5V
3	GND	4	GND
5	BD3	6	BD7
7	BD2	8	BD6
9	BD1	10	BD5
11	BD0	12	BD4
13	BA7	14	BA15
15	BA6	16	BA14
17	BA5	18	BA13
19	BA4	20	BA12
21	BA3	22	BA11
23	BA2	24	BA10
25	BA1	26	BA9
27	BA0	28	BA8
29	BWR*	30	BRD*
31	BIO*	32	BMEM*
33	SOD	34	BALE*
35	SI*	36	S0*
37	HLDA*	38	HOLD*
39	BINIAK*	40	INTRQ*
41	WAITRQ*	42	NMIRQ*
43	RESETOUT*	44	PBRESETIN*
45	CLOCKOUT*	46	PCLOCK
47	GND	48	GND
49	+12V	50	-12V

P2 AUXILLARY CONNECTOR

PIN NO	SIGNAL	PIN NO	SIGNAL
1	+5V	2	+5V
3	GND	4	GND
5	SID	6	SOD
7	RST 7.5	8	RST 6.5
9	RST 5.5	10	GND

PIN NO	SIGNAL
11	CLOCK 1
13	CLOCK 2
15	OUT 1
17	GND
19	INT 0
21	INT 2
23	INT 4
25	INT 6

PIN NO	SIGNAL
12	GATE 1
14	GATE 2
16	OUT 2
18	GND
20	INT 1
22	INT 3
24	INT 5
26	INT 7

J3 PRINTER ADAPTER

PIN NO	SIGNAL
1	STROBE*
2	DAT0
3	DAT1
4	DAT2
5	DAT3
6	DAT4
7	DAT5
8	DAT6
9	DAT7
10	NC
11	BUSY*
12	NC
13	NC

PIN NO	SIGNAL
14	NC
15	NC
16	NC
17	NC
18	GND
19	GND
20	GND
21	GND
22	GND
23	GND
24	GND
25	GND

J4 POWER CONNECTOR

PIN NO	SIGNAL
1	+5 V
2	GND
3	-12 V
4	+12 V
5	+30 V

CHAPTER 6

MONITOR ROUTINES ACCESSIBLE TO USER

ESA85-2 Monitor offers several user-callable routines both in the keyboard and serial modes of operation, details of which are given below. These routines can be used to considerably simplify the program development work.

NOTE: User should, as a general rule, save the registers of interest before calling the monitor routines and restore them after returning from the monitor routines.

6.1 KEYBOARD MONITOR ROUTINES ACCESSIBLE TO USER

Calling Address	Mnemonic	Functions
056CH	UPDAD	Updates Address field of the display. The contents of the locations (CURAD),FE73H & FE74H are displayed in the address field. The contents of all the CPU registers and flags are affected. If Reg. B=1, dot at the right edge of the field; if B=0, no dot.
0578H	UPDDT	Updates Data field of the display. The contents of the location (CURDT),FE75H are displayed in the data field. The contents of all CPU registers and flags are affected. If Reg. B=1, dot at the right edge of the field; if B=0, no dot.

04E3H	OUTPUT	Outputs characters to display. The parameters for this routine are as follows : Reg A=0 - Use address field Reg A=1 - Use data field Reg B=1 - Dot at the right edge of the field. =0 - No dot. Reg HL = Starting address of character string to be displayed.

0D75H	HILO	Computes the difference between two 16-bit values. Parameters are set up as follows:
		Reg DE = Value 1;
		Reg HL = Value 2;
		
		The difference between the two values is available in DE and carry is set as follows:
		Carry = 1 if value 1 Value 2
		= 0 otherwise
		DE = Value 1 - Value 2

0412H	CLEAR	Clears the display. This routine blanks the entire display field. Parameter is: REG B=1 - dot at the right edge of the address field. = 0 - No dot.
0514H	RDKBD	Reads keyboard. This routine waits until a character is entered from the system keyboard and upon return, it places the character in the A register. The register A and F/F's are affected.
049DH	HXDSP	Expand Hex digits for display. This routine expands hex digits and stores them in the specified buffer. OUTPUT routine can now be used to display this number, parameters: Reg DE = Hex value to be expanded. Registers A,H, L and flags are affected.
0630	RELCT	Refer Section 14.4

6.2 SERIAL MONITOR ROUTINES ACCESSIBLE TO USER

Calling Address	Mnemonic	Function/Description
0B97H	GETCH	Gets one character from the USART. Input parameters None. Output : C = Character (ASCII) received from USART. Regs. A,C and flags are affected.
0BB9H,	SOUTPT	Outputs one character to the USART. Inputs : C = Character (ASCII) to be output to USART. Reg. A,C, and flags are affected.
0C11H,	NMOUT	Outputs one byte as two hex digits to the serial I/O device. Inputs : A = Byte to be output Regs. A,B,C, and flags are affected.
0B0FH,	PRVAL	Gets the ASCII code corresponding to a hex digit. Inputs : C = Hex digit. Outputs : C=Corresponding ASCII code. Regs. B,C,H and L are affected.
0B04H,	DISPM	Displays a string of characters. The string should be terminated by character Zero which is not output. Inputs: HL = Starting address of the string of characters. Regs: A,C,H,L, and flags are affected.
0B69H,	VALDG	Checks if the character in C is a valid hex digit. Inputs : C = Character to be checked. Outputs : Carry = 1, if the character is a valid hex digit (0-F) Carry = 0 otherwise. Reg A and flags are affected.
32CDH,	DLY1MS	Introduces a delay of 1 millisecond. Reg A and flags are affected.
06BCH	SERIAL	Invokes serial monitor program.

CHAPTER 7

AUDIO TAPE INTERFACE

7.1 INTRODUCTION

Audio tape is an economical, large capacity, non-volatile storage medium. During the program development, the user can store the necessary information (Programs and data) on the audio tape and at a later point of time, he can reload the saved information into the system memory. Also, the Text Editor package utilizes this interface for loading/saving text files. (Refer chapter 10 on Text Editor). This relieves the user from the burden of manually entering large volume of information every time the system is powered on. Audio cassette recorder is an attractive solution as back-up storage device, in comparison with the more versatile Floppy Disks, primarily because of the lowcost. Any ordinary commercial tape recorder can be used.

The on-board Audio Tape interface consists of the necessary hardware and software to allow the user to store data and read data from a commercial audio tape recorder. The data is stored and read as named files.

7.2 INSTALLATION

The interface consists of two sockets - one marked as MIC and other as EAR. (Refer the Component Placement Diagram in Appendix C). Use any standard audio cable(supplied along with the trainer) to connect MIC or EAR socket to the tape recorder.

7.2.1 OPERATIONAL HINTS

1) Connect only one socket MIC or EAR at any time to tape recorder. Connecting both the sockets permanently to the tape recorder may short circuit both the signal lines if the recorder common point connection is different from the convention followed in ESA85-2.
2) The volume and tone controls of your tape recorder may have to be adjusted to sufficiently high levels for reliable operation of this interface.
3) The reliability of the recording can be increased by using a recorder and tape of good quality.
4) The interface can accomodate normal variation in the tape speed. Ensure that your recorder does not produce too significant changes in the tape speed.
5) Avoid storing relatively large files. If the file is really large, split it into multiple files of smaller size. Though this scheme reduces the utilization of the tape space, it improves the chances of restoring the complete file correctly.

7.3 OPERATION FROM THE KEYBOARD MONITOR

The software allows the user to store data on the tape and read the recorded data from the tape, as named files.

7.3.1 STORING DATA ONTO TAPE

TPWR command implements this facility. Thus to invoke this facility, press TPWR key when prompted for command entry. After typing TPWR, set the recorder in record mode and now enter the parameters as shown below:

<Filename> NEXT <starting address> NEXT <Ending address> EXEC

OPERATION

As soon as TPWR key is pressed, the display is cleared with a dot at the right edge of the address field. Enter the file name as a string of hexadecimal digits. Any number of digits can be entered, but the last 4 digits entered (currently displayed in the address field) are treated as valid. After entering the filename press the NEXT key. The display is again cleared with a dot at the right edge of the address field. Now enter the starting address of the block of data to be transferred to the tape and press the NEXT key. The display is again cleared with a dot in the address field. Now enter the ending address of the block of data to be transferred to the tape.

After entering the ending address, connect the microphone input of the recorder to the MIC jack of the tape interface hardware, press PLAY and REC keys of the recorder and then press the EXEC key.

Now data is transferred onto the tape. After writing the data, the system records a checksum byte also. During the transfer, the display will show only dots. After transferring the data, control returns to the Monitor.

IMPORTANT NOTE

If the recorder is not ready and if you press the EXEC key, data would be sent out from the interface and this data would not be recorded on tape. Hence ensure that the recorder is ready and in record mode before pressing the EXEC key.

Example: To store the contents of locations 8800 - 88FF on tape with a file name 123F.

Key pressed	**Display**		**Comment**
	Address field	**Data field**	
RESET	- E S A	8 5	System Reset
TPWR			Tape write command
1	0 0 0 1.		
2	0 0 1 2.		
3	0 1 2 3		Filename = 123F
F	1 2 3 F		
NEXT			Prompt for starting address
8	0 0 0 8		
8	0 0 8 8		
0	0 8 8 0		
0	8 8 0 0		Starting address=8800H
NEXT			Prompt for ending address
8	0 0 0 8		
8	0 0 8 8		
F	0 8 8 F		
F	8 8 F F		Ending address=88FFH Connect the microphone of the recorder to MIC. Press REC and PLAY keys of the recorder.
EXEC	. . . .		Data is being recorded on the tape Display is all dots. control returns to Monitor after completing the transfer.

7.3.2 READING DATA FROM TAPE

TPRD command implements this facility. Thus to invoke this facility, press TPRD key when prompted for command entry.

After pressing TPRD, enter the desired filename followed by EXEC i.e. << file name>>EXEC

Then set the recorder in PLAY mode.

OPERATION

As soon as TPRD key is pressed, the display is cleared with a dot at the right edge of the address field. The user must now enter the desired filename (As starting and ending addresses are already stored on the tape, there is no need for the user to enter these values).

Connect the Earphone of the recorder to EAR jack of the Tape Interface Hardware. Position the tape well ahead of the approximate start of the desired file. Now press the EXEC key and then press the PLAY key of the recorder.

The display will be dashes in all the digit positions while the search is on for the desired file. As and when a filename is read by the system, it is displayed in the address filed. When the specified file is found, the entire display will show dots. Data is transferred from the tape in to the memory. As noted already, the starting and ending addresses of the memory block are read from the tape itself. A checksum also, recorded on the tape during tape write, is read and is compared with the checksum of the actual data transferred. If they do not match, or if the data transfer is not correct, the system displays error message - Err and control returns to the monitor. If the data is transferred without any errors, control returns to the monitor which will display the command prompt dash.

EXAMPLE: To read the contents of the file with the filename of 123F.

Key pressed	**Display**		**Comment**
	Address field	**Data field**	
RESET	- E S A	8 5	System Reset
TPRD.			Tape Read command.
1	0 0 0 1		
2	0 0 1 2		
3	0 1 2 3		
F	1 2 3 F		Filename 123F
			Connect recorder (using earphone jack) to the EAR of theTape Interface Hardware. Press play of the recorder.

Key pressed	Display		Comment
	Address field	Data field	
EXEC	- - - -	- -	Initial display. All dashes
	x x x x		First filename found and displayed.
	x x x x		Second filename found; search continues.This process repeats until the required file is found.
	1 2 3 F		Specified file found. the entire display shows dots. Data is transferred from the tape into memory. Completion of data transfer successfully.

7.4 OPERATION FROM THE SERIAL MONITOR

The software allows the user to store data on the tape and read the recorded data from the tape, as named files.

7.4.1 STORING DATA ONTO TAPE

"W" command implements this facility. To use this command, type W, when the system prompts for a command (with the dot prompt.)

The system will now prompt for a file name as shown below:

FILE NAME =

Enter the file name followed by RETURN. A valid file name consists of a sequence of hexadecimal digits. User may enter any number of digits but the system retains only the last four digits. If an invalid file name is entered, the operation is aborted with an error message and control returns to the monitor.

Now the system will prompt for the starting address of the memory block to be saved on the tape:

START ADDRESS =

Enter the starting address followed by RETURN.

Now the system will prompt for the Ending address as follows:

END ADDRESS =

Ensure that the Recorder is connected to the interface via the MIC jack. Set up the Recorder in the record mode.

Now enter the ending address followed by RETURN. If either of the address entries is not a valid hexadecimal number, the operation is aborted and control returns to the monitor. Further, the ending address must be greater than or equal to the starting address. Otherwise, the system will display the error message **“END START”** and prompt again for the ending address.

The system will now display the message, **“WRITING ON TO TAPE”** and begin transferring the specified data onto the tape. After the data, a checksum byte is also written onto the tape. After completing the transfer, the system will display the message **“FINISHED WRITING ONTO TAPE”** and then control returns to monitor.

IMPORTANT NOTE

Data is sent out once the user enters the RETURN after the Ending Address. If the RECORDER is really not ready at this stage, the data is still sent out and this will not get recorded on the tape!

EXAMPLE

Store the contents of locations 8000H to 8FFFH, with a file name 14B3, on the tape.

```
.W
FILE NAME = 14B3
STARTADDRESS = 8000
Now connect the RECORDER via the MIC jack and set it in RECORD mode.
END ADDRESS = 8FFF
WRITING ONTO TAPE
FINISHED WRITING ONTO TAPE
```

7.4.2 READING DATA FROM TAPE

“L” command implements this facility. To use this facility, type L followed by <<CR>>, when the system prompts for a command (with the dot prompt.)

The system will now prompt for the filename as shown below:

FILE NAME =

Enter the filename followed by RETURN. (The rules regarding the filename are same as the ones explained in the W command.) Note that the addresses of the memory block are already stored on the tape under the specified filename. So user should not enter them again. Now the system will display the message, “SEARCHING THE FILE”.

Connect the RECORDER to the interface via the EAR jack. Position the tape well ahead of the approximate starting point of the desired file. Set the RECORDER in the play mode.

As and when the system finds a file, it displays the filename as shown below:

FILE NAME = X X X X

If the file is not the specified one, the search continues. When the specified file is found, it displays the message, “READING DATA FROM THE FILE” and begings loading the data into memory.

As already noted, the start and end addresses of the memory block are read from the tape itself. Further, the checksum byte recorded on the tape during Tape Write operation, is also read. This checksum byte is compared with the checksum calculated from the actual data read from the tape and if they do not match, it is treated as error condition.

If the file is loaded without any errors, it displays the message, “FILE LOADED” and control returns to the monitor. If any errors occur either in read or in writing into the memory, it displays the message, “READ FAIL” and control returns to the monitor.

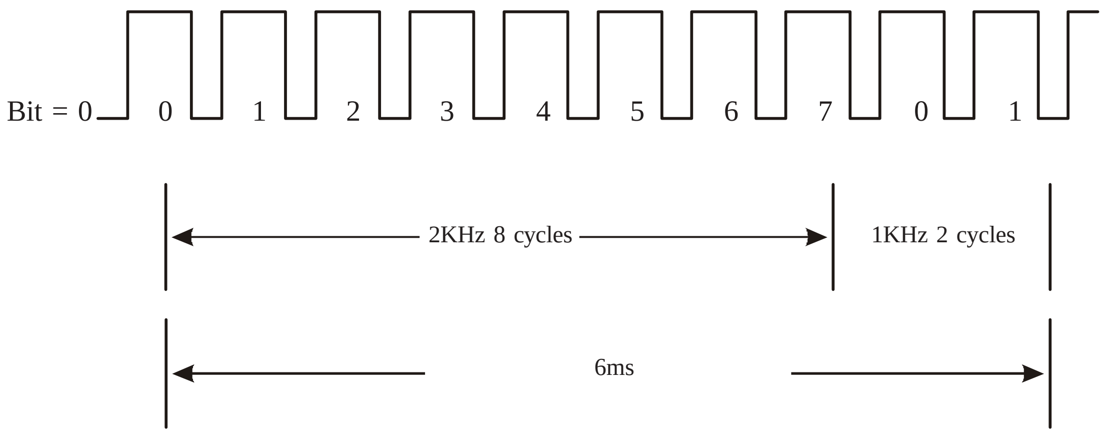
Example: Read the contents of the file with filename 14B3.

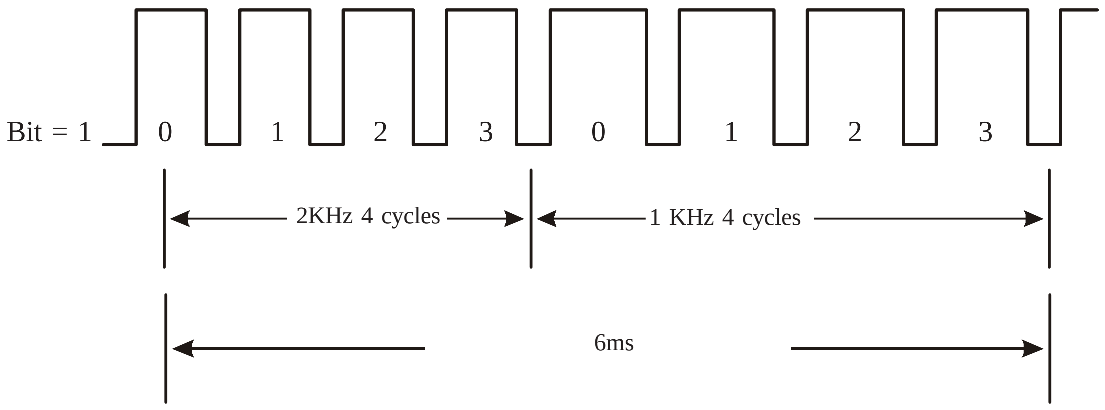
```
.L

FILE NAME = 14B3 <<CR>>
SEARCHING THE FILE
FILE NAME = X X X X
FILE NAME = X X X X
FILE NAME = 14B3
LOADING THE DATA FROM THE FILE
FILE LOADED
```

7.5 DATA RECORDING FORMATS

7.5.1 BIT FORMAT:

Both 0 and 1 are recorded as combination of some high frequency (2KHz) signals and some low frequency (1KHz) signals as shown below:





7.5.2 BYTE FORMAT :

A byte is recorded as one start bit (a zero), 8 data bits and one stop bit.

Start Stop

0:	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	1
60 ms									

7.5.3 FILE FORMAT:

Before recording the file, a lead synchronization signal of 1 KHz frequency is recorded for 4 seconds. Then a seven byte header is recorded. The file header consists of filename (2 Bytes), start address (2 bytes), end address (2 bytes) and checksum (1 byte). Following this, a mid synchronization signal is recorded. This consists of a 2KHz signal for 2 sec and 1KHz signal for 0.8 secs. Now the file data is recorded. After this, a tail synchronization signal of 2KHz frequency is recorded for 2 seconds. Thus the file format is:

Lead Sync	File Name	Start addr	End addr	Check sum	Mid sync	Data	Tail sync
1KHz 4 sec	2 Bytes	2 Bytes	2 Bytes	1 Byte	2KHz 1KHz 2sec + 0.8 sec		2KHz 2 sec

7.6 STORAGE CAPACITY

As is obvious from the above description of the Data formats, the exact amount of data that can be stored on a tape depends upon the total number of files created. The larger the number, the smaller is the amount of data that can be stored (because of the space consumed by the larger number of synchronization signals). Assuming a typical number of 25 files for a standard tape of 90 minutes duration, we see that approximately 10 minutes (after allowing for about 15 seconds of inter-file gap during recording) may be taken up by synchronization signals. This allows us to record approximately 72 KB of data on one such cassette. However, if data is recorded in files of smaller size (say 128 or 256 bytes/file), the number of files stored will increase and total data storage capacity may fall to about 40K bytes.

CHAPTER 8

PARALLEL PRINTER INTERFACE

8.1 INTRODUCTION

ESA85-2 trainer supports Centronics Compatible Parallel Printer Interface. The interface makes use of BUSY signal for hand shaking and strobe pulses for synchronization. Using this facility the user can obtain hard copy on any centronics compatible printer. However to get properly formated listing it is advisable to use 80/132 column printer. The on-board 8255A is made use of, to implement this interface.

8.2 INSTALLATION

To install the printer interface

a) Switch OFF the power supply.

b) Connect one end of the printer cable to J3 (25pin ‘D’ type connector) of ESA85-2. (Refer the component layout diagram in Appendix C to locate the connector J3).

c) Connect the other end of the printer cable to the printer.

d) Configure the system for serial mode of operation by setting SW4 of the on-board DIP switch to ON position.

e) Enable the printer interface by setting the SW6 of the on-board DIP switch to ON position.

f) Switch on the power to the printer and ESA85-2.

Note: The necessary printer cable could be obtained from Electro Systems Associates (P) Ltd as an optional accessory. However, the connector details are given in section 8.5 and the user can make use of these details to make a suitable cable if desired. Please note that the cable must be short enough to be driven by the on- board 8255A. We suggest a maximum length of 3 feet for reliable operation.

8.3 OPERATION

When the printer interface is installed and enabled as described above, any character sent to the console is sent to the Printer also. For example, to obtain a hard copy of the contents of a block of memory locations, user can issue the D (Display Memory) command from the serial monitor. The contents of the specified memory block are printed exactly as they appear on the screen. Note that the D command itself is also printed.

Notes :

1. All control and non-printable ASCII characters are printed as "." (ASCII Code 2EH).
2. If any errors occur during printing (for eg: printer is not in ON-LINE, paper-out error etc), the system will be looping indefinitely in the print character routine. To recover, user may have to press the RESET key.

8.4 DIRECT OUTPUT TO PRINTER

As already described, when the printer interface is enabled, any character sent to the console is sent to the printer also. This facility is available in the serial mode of operation only. However, user can directly access a routine "print character" to print a single character. This routine can be called from the user's program when the system is operating in either of the two modes-keyboard or serial. Further, this routine prints a character independent of the setting of SW6. Thus this routine can be used to print the desired information when the system is running in the keyboard mode. Even in the serial mode of operation, this routine can be used to print information which may not be sent to the console. The details of this routine are given below:

Name of the Routine :PRINT
Function : Print a character (Non-printable one is printed as ".").
Calling address : 0BCAH
Input : Register C=ASCII code of the character to be printed.
Destroys: Register A and Flags.

Example :

The following program prints the message "ESA". The program can be executed either from the Keyboard monitor or from the Serial monitor. (NOTE: Ensure that the printer is connected to the system and that it is in ONLINE mode).

LOCATION	CONTENTS	MNEMONIC	COMMENTS
8000	21 0E 80	LXI HMSG	;Point to the message area
8003	4E LOOP:	MOV C, M	;Get the character
8004	CD CA 0B	CALL PRINT	;and print it.
8007	7E	MOV A, M	The message is assumed to be ;terminated by a null character.
8008	23	INX H	;Point to next character.
8009	B7	ORA A	;End of the message?
800A	C2 03 80	JNZ LOOP	;No-continue printing
800D	DF	RST 3	;Yes-return to the monitor

MSG : DFB “ESA”, 0DH,0AH,00H

8.5 CONNECTOR DETAILS

The signal definitions on the 25-pin, female D type connector used for parallel printer interface are given below :

PIN NO ON J3	SIGNAL	DIRECTION FROM ESA85-2	DESCRIPTION	PIN NO ON CENTRONICS CONNECTOR
1	STROBE	O/P	STROBE* pulse to the printer ,	1
2	Data 0	O/P	These signals	2
3	Data 1	O/P	represent 8 bits	3
4	Data 2	O/P	of parallel data	4
5	Data 3	O/P	High = 1	5
6	Data 4	O/P	Low = 0	6
7	Data 5	O/P		7
8	Data 6	O/P		8
9	Data 7	O/P		9
11	BUSY	I/P	A high indicates that Printer cannot receive data The signal becomes high in following cases a) During data entry b) During Printing operation c) In OFF-LINE state d) During printer error status,	11
18-25	GND		Signal ground	19

CHAPTER 9

PROM PROGRMMER SYSTEM

9.1 INTRODUCTION

ESA85-2 PROM programmer system is a powerful and easy to use facility provided on-board ESA85-2. This chapter describes the use of this PROM Programmer System.

NOTE: 30V, and +12V power supply must be connected when the PROM Programmer System is being used. If a separate 30V, +12V power supply is being used, ensure that its common is connected to the common of the system power supply.

The system permits the user to program, verify, blank check and read any of the popular EPROMs - 2716 through 27512. The system consists of the necessary hardware and software. The software can be invoked either from the keyboard monitor or from the serial monitor. A 28 pin ZIF socket is provided for placing the EPROMs. When a 24-pin EPROM is to be placed, it must be aligned with the bottom row i.e. top two rows of pins are to be left blank.

The system uses Intelligent Programming Algorithm whenever possible which reduces the programming time significantly.

The devices supported by the system and the type number to be entered by the user are listed below:

Device		Type number to be entered by the user
2716	(@25V)	2716
27C16	(@25V)	2716
2732A	(@21V)	732A
27C32A	(@21V)	732A
27C32	(@25V)	2732
2732	(@25V)	2732
2764A	(@12.5V)	764A
27C64D	(@12.5V)	764A
27C64	(@21V)	2764
2764	(@21V)	2764
27128A	(@12.5V)	128A
27C128	(@12.5V)	128A
27C128	(@21V)	0128
27128	(@21V)	0128
27256	(@12.5V)	0256
27C256	(@12.5V)	0256
27C256	@21V)	2256
27512	(@12.5V)	0512

The device selection is totally software-controlled and no hardware changes or jumper settings are necessary for selecting any of the above listed devices.

9.2 OPERATION FROM SERIAL MONITOR:

The Serial Monitor provides the following 4 commands to support the PROM Programmer SYSTEM:

P - Program Command
V - Verify Command
B - Blank Check Command
R - Read Command

Enter the appropriate command, when prompted for command entry by the serial Monitor.

Aborting a Command:

Once a specific command is issued, further prompts will depend on the command itself. However, if the user enters ESC whenever the system is looking for an entry from the user, the current operation is aborted and control returns to the warm start of the Serial Monitor.

9.2.1 P COMMAND:

This command is used to program a PROM.

This command required the following four parameters:

PROM TYPE	=	PROM Type : should be one of the types listed above in section 9.1
BUFFER START	=	Starting address of the source of data
BUFFER END	=	Ending address of the source of data.
PROM START	=	Absolute starting address of the PROM (from where programming is to begin)

As soon as P is typed, the system displays each parameter value and prompts for new value. User can enter a new value followed by Carriage Return or simply enter Carriage Return if the displayed value is not to be changed.

Note that the parameters must satisfy certain conditions as listed below.

i) PROM type can be only one of the valid types listed in section 9.1

ii) Buffer end address must be greater than or equal to the Buffer start address.

iii) The PROM must have enough space to accommodate all the bytes specified by the Buffer start address and Buffer end address. In other words, the following relation must be satisfied.

Prom Start + (Butter and Address-Buffer Start Address) < = Highest Absolute Address of the PROM.

For example, suppose PROM type is 2764. Then its highest absolute address is 1FFF H. Suppose the other parameters are as follows:

Buffer Start=8000
Buffer End=9FFF
PROM Start=100

The 100 + (9FFF-8000)=20FF>1FFF. So this combination of parameters is invalid.

* As user enters the parameter values, the above mentioned constraints are checked and if any of the constraints are violated, the error message ‘What ???’ is displayed and user is again prompted for a value for the offending parameter.

After optional modification of the parameter values by the user, the system checks the PROM for blank values (0FFH) in the required zone. If the PROM is not blank, the following prompt appears:

PROM IS NOT BLANK. OK? (Y/N)

If user types, N, the command is aborted and control returns to command prompt of the Serial Monitor.

If the user enters Y, the system proceeds further. Any other character results in error message and repetition of the same prompt.

* Now the following message appears:

PROGRAMMING IN PROGRESS...

The system proceeds with programming and verification on a byte by byte basis. Intelligent Programming Algorithm is used if the PROM can support it. This results in considerable reduction n programming time required for such devices.

* If the complete programming is successful, the system will display a 16-bit checksum and control will return to the Serial Monitor.

If the programming is unsuccessful, the following information is displayed:

PROGRAMMING FAILURE @ XXXX

Where XXXX is the PROM address where programming failed.

NOTE: During programming and verification from serial mode, the location in PROM and the corresponding data that is being programmed are displayed in the Keyboard/Display unit’s display fields continuously.

After programming the specified range, the system performs another verification cycle for the entire range. If any mismatch is found, it displays the message.

VERIFY FAILURE @ XXXX

where XXXX is the PROM location where verify failed. Then control returns to the monitor.

9.2.2 V COMMAND

This command is used to verify the contents of a PROM against a source.

The parameters and their interpretation is completely similar to that of the P Command.

If the verification is successful, the 16-Bit checksum is displayed and the Serial Monitor command prompt appears on the next line.

If the verification fails, a message and parameters at the point of failure are displayed as shown below.

VERIFY FAILURE @ XXXX

Where XXXX is the PROM address where verification has failed. Then control returns to the Serial Monitor.

Thus the operation of this command is quite similar to the operation of the P command, except that here the PROM is just verified, not programmed.

9.2.3 B COMMAND

This command is used to check if a specified range in the PROM is blank (contains FFH)

This command requires the following three parameters:

PROM TYPE	:	Same as for P command
PROM START	:	The absolute starting address of the PROM
PROM END	:	The absolute ending address of the PROM.

The parameters must satisfy the following relations:

i)	PROM START	<=	Absolute Last Address of the PROM
ii)	PROM END	<=	Absolute last Address of the PROM and
iii)	PROM END	>=	PROM Start.

The parameter display and modification procedures are menu-driven and are similar to those of the P command

The PROM is checked for blank values in the specified range. The PROM addresses and data read are displayed in the Keyboard/Display unit’s display fields. If it is blank then the following message is displayed:

O.K. PROM IS BLANK

Then control returns to the Serial Monitor.

If a location in the specified range is not blank, the following message is displayed.

PROM IS NOT BLANK @ XXXX

Where XXXX is the absolute PROM address of the first non-blank location. Then control returns to the Serial Monitor.

Note that only the first non-blank location address is displayed. Subsequent locations may also be non-blank locations.

9.2.4 R COMMAND

This command is used to transfer the contents of the PROM into the ESA85-2 memory space.

This command requires the following four parameters:

TYPE	:	Same as for P command
PROM START	:	Same as for B command
PROM END	:	Same as for B command
BUFFER START	:	Starting address in ESA85-2 memory space.

The parameter display and modification procedures are menu driven and are completely similar to the ones described for P command.

The starting and ending addresses of the PROM must satisfy the relations described for the B (Blank check) command.

After the optional modification of the parameters, the contents of the PROM, in specified range are transferred into ESA85-2 memory, starting at the specified Buffer starting address. The PROM addresses and data read are displayed in the Keyboard/Display unit's address and data fields respectively.

During the transfer, as each byte is written into memory, it is read back and verified. If the write is successful for all the locations, a 16-bit checksum is displayed and control returns to the Serial Monitor.

If an error occurs during transfer (i.e. unsuccessful write into a location), the following message is displayed.

WRITE FAILURE @ MEMORY LOCATION XXXX

where XXXX is the address of the location where write failure occurred. Then control returns to the Serial Monitor.

9.3 OPERATION FROM THE KEYBOARD MONITOR

The Keyboard of ESA85-2 has four keys which can be used to invoke the functions of the PROM Programmer System. The keys and their functions are

Key	Function
PROG	**Program a PROM**
VRFY	**Verify a PROM**
BLNK	**Blank Check a PROM**
PRRD	**Read a PROM**

Parameter Entry:

The following information is common to all the commands.

Whenever a parameter entry is required, the system will display a message in the data field which indicates the type of parameter to be entered and it will display the default value in the address field with a dot. To retain this value, user can press NEXT or EXEC, as required. Otherwise, user can enter the new value and then press NEXT or EXEC as required. All entries are made into the address field only. The different messages which can appear in the data filed and their meanings are given below:

Message in the data field	Meaning
Pt	The type number of the PROM
bS	Buffer Start Address
bE	Buffer End Address
PS	PROM Starting Address
PE	PROM Ending Address

All the parameter values are evaluated modulo 64K and a parameter entry is terminated by any valid delimiter. (Valid delimiters are: NEXT and EXEC). If an invalid entry is made, the message '-Err' is flashed in the address field and the prompt for the offending parameter is displayed again so that the user can enter the correct value. Pressing the 'NEXT' key takes the user to the next parameter if one is required - Otherwise it is treated as error condition.

Pressing the 'EXEC' key completes the parameter entry process of the command.

9.3.1 PROG COMMAND:

This command is used to program a PROM,

This command requires the following four parameters in that order.

PROM TYPE	PROM type (should be one of the type listed in section 9.1)
BUFFER START	Starting address of the source of data.
BUFFER END	Ending address of the source of data.
PROM START	Absolute Starting address of the PROM (from where programming is to begin).

* Note that these parameters must satisfy certain conditions as explained in the section 9.2.1 (P Command).

* Once the correct parameter values are available, the system checks the PROM for blank values (FFH) in the required zone. If the PROM is blank in the required zone, the system proceeds further. Otherwise, it displays the message FULL in the address field with a dot and waits for user input. Now if the user presses 'EXEC' key, the system returns control to the Keyboard Monitor. If user presses "NEXT" key, the system proceeds further. Any other key will result in flashing of error message and reprompt.

* After this, the system proceeds with programming and verification on a byte by byte basis. As each location in the PROM gets programmed, the PROM address is displayed in the address field and the programmed data is displayed in the data field.

* The system utilizes Intelligent Programming Algorithm wherever applicable and this reduces the programming time significantly.

* If the complete programming and verification is successful, the system will display a 16-bit checksum in the address field and then the system waits for user input. If user presses "EXEC" control returns to the Keyboard Monitor. Pressing any other key (except RESET) has no effect.

If the programming is unsuccessful, the address of the failed PROM location is displayed in the address field and ROM data is displayed in the data field. The system now waits for user input. If user presses "EXEC" key, control returns to the keyboard Monitor. Pressing any other key (except RESET) has no effect.

After programming the specified range, the system performs another verification cycle over the entire range. If any mismatch is found, it displays the PROM address and data and waits for user input. If user presses “EXEC” control returns to the Keyboard Monitor. Pressing any other key (except RESET) has no effect.

9.3.2 VRFY COMMAND

* This command is used to verify the contents of a PROM against a source.
* The parameters and their interpretation is completely similar to that of the PROG command.
* If the verification is successful, the 16-bit checksum is displayed in the address field. Now press “EXEC” key to return control to the monitor. Pressing any other key (except RESET) has no effect.
* If the verification fails, then the parameters at the failed location are displayed as in the PROG command.

(i.e. PROM address and data are displayed).

Now if ‘EXEC’ key is pressed, control returns to the Monitor. Pressing any other key (except RESET) has no effect.

9.3.3. BLNK COMMAND

* This command is used to check if a specified range in the PROM is blank (consists 0ffh).

This command requires the following three parameters in that order:

PROM TYPE	:	Same as in PROG Command
PROM START	:	The absolute starting address of the PROM
PROM END	:	The absolute ending address of the PROM

* The parameters must satisfy certain relations as explained in the section 9.2.3 on B command..
* Once valid parameter values are available, the PROM is checked for blank state in the specified range.
* If the PROM is blank in the specified range, the message-PAS is displayed in the address field. and control returns to the Keyboard Monitor. Otherwise, the address of the first non-bank location is displayed in the address field and the PROM data is displayed in the data field. The system now waits for user input. If “EXEC” is pressed, control returns to the monitor. Pressing any other key (except RESET) has no effect.

9.3.4 PRRD COMMAND

* This command is used to transfer the contents of the PROM into the memory space of ESA85-2.

This command requires the following four parameters in that order:

PROM TYPE	:	Same as in PROG Command
PROM START	:	Same as in Blank Check Command
PROM END	:	Same as in Blank Check Command
BUFFER START	:	Starting address in ESA85-2 memory space.

* These parameters must satisfy certain conditions as explained in the section 9.2.4 on R command.

* Once correct parameter values are available, the system reads the PROM and transfers the contents to successive locations starting from the specified Buffer start. The PROM address and data are displayed in address and data fields respectively.

* If the transfer is successful, a 16-bit checksum is displayed. Otherwise the address of the offending location is displayed in the address field and PROM data is displayed in the data field. The system now waits for user input. If "EXEC" is pressed control returns to monitor. Pressing any other key (except RESET) has no effect.

9.4. EXAMPLES

Example 1: From keyboard, read the contents of a 2732 PROM into memory locations 8000H to 8FFFH.

Key Pressed	Display		Comments
	Address Field	Data Field	
RESET	-ESA	85	
PRRD	2732.	Pt	Transfer command. Type prompt
NEXT	0000.	PS	PROM Start
NEXT	0FFF.	PE	PROM End

Key Pressed	Display Address Field	 Data Field	Comments
NEXT	0000.	bS	Buffer Start
8	0008.	bS	
0	0080.	bS	
0	0800.	bS	
0	8000.	bS	
EXEC	1234		successful transfer. Checksum
EXEC	-ESA	85	Return to monitor

Example 2: From Serial monitor, program the contents of locations 8000H to 8FFFH into a 2764, starting at 1000H. (User’s entry is underlined only for clarity)

P

PROM TYPE	2732 - 2764	<CR>
BUFFER START	0000 - 8000	<CR>
BUFFER END	0000 - 8FFF	<CR>
PROM STAT	0000 - 1000	<CR>

PROGRAMMING IN PROGRESS...

Checksum=1724

9.5 SUPPORT FOR EEPROM 2816A

The Prom Programmer System of ESA85-2 supports all the popular UV-Erasable EPROMs as described already. Further, it supports the Electrically-Erasable EPROM 2816A also. The type number should be specified as shown below:

Device	Type Number to be entered
2816A	816A

Program, Read, Verify and Blank check commands can be used exactly as described in the earlier sections.

BYTE ERASE:

As 2816A is electrically erasable, you can erase a specific location by programming FFH in that location. Thus enter a data value of FFH in the source buffer locations corresponding to the EEPROM locations to be erased and then issue the normal P command. This procedure will erase the selected locations.

CHIP ERASE:

The entire chip can be erased by issuing the P command and specifying a source buffer of 2K bytes capacity with each location containing a value of FFH. For example, assume that each location in the range 8000H to 87FFH contains the value FFH. Now you can erase the entire chip by issuing the P command with the following parameters:

PROM TYPE	=	816A
BUFFER START	=	8000
BUFFER END	=	87FF
PROM START	=	0000

The system recognizes this situation and instead of programming each location with 0FFH, it erases the entire chip using the chip Erase feature of 2816A.

9.6 SUPPORT FOR 8751 FAMILY MICROCONTROLLERS

The EPROM Programmer section of ESA 85-2 trainer also supports Microcontrollers of 8751 family with an ADDITIONAL HARDWARE and 8751 adapter to the PROGRAMMER.

The 8751 adapter module has a 28-pin long-lead IC type connector. Insert this connector into the 28-pin ZIF socket on the ESA 85-2 trainer. (Be careful to ensure that Pin 1 of this connector is towards the lever of the trainer ZIF socket.) The adapter module has a 40-pin ZIF socket to house the 8751 series of microcontrolles.

The devices supported and the type number to be specified are given below:

DEVICE	TYPE number to be entered
8751/8751H - 4K Bytes (21.0V)	8751
87C51 - 4K Bytes (12.5V)	0C51
87C51FA - 8K Bytes (12.5V)	C51A
87C51FB - 16K Bytes (12.5V)	C51B

Program, Read, Verify and Blank check commands are same and can be used exactly as described in the earlier sections. Care should be taken about the jumper positions on the adapter. Jumper plug JPI should be SHORT and JP2 OPEN while Program/Read/Blank check/Verify.

CAUTION:

8751 adapter should be installed only when the system is ON and the device selected in 8751 family Microcontroller. With the adapter installed, if the device selected is other than 8751 family Microcontrollers, the equipment and the device can suffer severe damage!!!

9.6.1 SECURITY BIT PROGRAMMING

In the above listed devices 8751 has one security bit. When this bit is programmed, external access to the microcontroller is denied. If such a device is read, the data appears as "FF", as if it is a blank device.

To program the security bit

OPEN the Jumper plugs JPI and JP2
Program any location with any data.

87C51/87C51FA/87C51FB have two security bits called SECURE EXTERNAL bit and SECURE VERIFY bit. Programming both the bits denies any external access to the on-chip Program memory.

If only SECURE EXTERNAL EXECUTION bit is programmed and then a 32x8 SECURITY TABLE is programmed, the chip can maintain the verify capability. The data that appears on the Port 0 during the secured verification is scrambled by the SECURITY TABLE contents.

Erasing the EPROM erases the SECURITY TABLE as well as the security bits. The device can again be Programmed/Read/Verified.

ESA 85-2 Trainer supports programming of both the security bits

However, it does not support programming of SECURITY TABLE.

To Program Security Bit 1 (External Execution)

OPEN the Jumper plugs JPI and JP2.
Program any location with any data.

To Program Security Bit 2 (Verify Control)

OPEN the Jumper plug JP1 and CLOSE JP2.
Program any location with any data.

CHAPTER 10

TEXT EDITOR

10.1 INTRODUCTION

The firmware of ESA85-2 includes a powerful, line-oriented Text Editor for creating, editing and saving ASCII text files. This utility can be invoked from the Serial Monitor only. Assembly language source files created and edited using this Text Editor can be assembled using the resident 2-pass Assembler. Further the source files can be saved on/loaded from the tape using the on-board Audio Tape Interface. Thus these utilities can be used to considerably simplify the task of program development.

The Text Editor is a line-oriented one, i.e. the basic unit of text for all editor operations is a line. A line of text consists of a sequence of ASCII characters terminated by the Carriage Return (ASCII code 0DH). A line is limited to a maximum length of 72 characters, adequate for all practical applications. A file consists of a sequence of lines terminated by the end-of-file marker 1AH. The file has a name which consists of a maximum of 4 hexadecimal digits. Thus the conventions are same as those assumed by the Audio Tape Interface software (Ref. chapter 7 or Audio Tape Interface).

10.2 E (EDIT) COMMAND

To invoke the Text Editor, type E when prompted for command by the Serial Monitor.

The Text Editor then displays a sign-on message, followed by the list of commands supported by the Text Editor.

The display appears as shown below:

```
ESA85-2 Text Editor VX.Y

N=NEW

D=Delete

I=Insert

L=List

R=Replace

X=Exit

LOAD-
```

The X and Y in the sign-on message refer to the version and release numbers. As can be seen from the above list, the Text Editor supports six commands which are described in detail in the subsequent sections.

After listing the commands, the system displays “LOAD-” which is the prompt for loading a file from the Tape. If a file is to be loaded from the Audio Tape, enter the file name followed by Carriage Return. Ensure that the Tape Recorder is connected to ESA85-2 via the EAR jack, the tape is positioned properly and the Recorder is set in the PLAY mode. This operation, and the system responses are exactly similar to the ones described for the TR command. (Ref chapter 7 on Audio Tape Interface). When the specified file is loaded, the system displays the fo\lowing message:

Starting Address=XXXX

Ending Addrers=XXXX

Buffer End=XXXX

The Starting Address and Ending Address refer to the start and end of the text file as recorded on the tape. The Buffer End displayed is the current end of the text buffer. Now the system prompts for a new buffer end as follows :

BUFFER END =

Af the end of the text buffer is to be different from the displayed Buffer End address, enter a new value followed by Carriage Return; otherwise enter only a Carriage Return. Now the Text Editor displays its command prompt "," indicating that it is ready to accept commands from the user.

If the response to the "LOAD-" prompt is only a Carriage Return (i.e. no file name is specified by the user), the system assumes that no file is to be loaded from the tape. It then searches the text buffer to determine if there is a valid text file in the buffer. If there is no text file (which will be the case if the Text Editor is being invoked for the first time after power-on condition), the system displays the following message and prompt:

Text Buffer (0F000-0F9FF)

Starting Address=-

As shown above, the Text Editor assumes the text buffer to be located, by default, from POOOH to F9FFH. User can change these addresses if required. The next prompt is for the Starting Address of the text buffer. If this value is to be different from the default value of F000H, then enter a new value followed by Carriage Return; otherwise enter only Carriage Return. Then the system prompts for the ending address of the text buffer as shown below :

Ending Address=-

If the ending address is to be different from the default value of F9FFH, enter a new value followed by Carriage Return; otherwise enter only Carriage Return. In any case, note that the Ending Address must be greater than the Starting Address; otherwise the system displays an error message and prompts again for the Starting Address and Ending Address. After obtaining valid value, the system displays the command prompt of the Text Editor, a ">" on a new line, indicating that the Text Editor is ready to accept commands from the user.

If user responds with only a Carriage Return to the "LOAD-" prompt and the system finds a valid text file in the text buffer (created/loaded in an earlier invocation of the Text Editor,) the system will display the starting and ending addresses of the text file, and the ending address of the text buffer. Then, it will prompt for a new Buffer End Address. Thus the sequence of displays is exactly same as the one resulting from a file being loaded from the tape. In other words if the system finds valid text file, it proceeds as if the file is loaded from the tape as explained already.

10.3 EDITOR COMMANDS

As already noted, the Text Editor supports six commands -N,D,I,L,R and X. These commands are described in details in following subsections.

Each of these commands requires zero, one or two parameters. When two parameters are requried, they must be separated by a comma. A command entry is terminated by the Carriage Return. Thus the system begins the interpretation of the command only after Carriage Return is entered.

During the entry of command, parameter or text, the system supports the CTRL-H (Back space, ASCII code 08H) operation.

As already noted, the Text Editor is a line-oriented one. The line numbers are maintained as Hexadecimal numbers and the first line has the line number of 0001. User also must specify the line numbers as hexadecimal numbers.

10.3.1 I (INSERT) COMMAND

FUNCTION :

This command allows the user insert one or more lines of text into the file currently being edited.

FORMAT :

I <CR> Or I # <CR> In <CR>

OPERATION :

This command is invoked by typing I when prompted for command by the Text Editor. I can be optionally followed by the hash cahracter ("#") or by a line number. Terminate the command entry with Carriage Return.

I without any parameters, allows the user to insert text at the beginning of the file. If the file is opened for the first time, the system displays the message "NEW FILE". As text is being inserted at the beginning of the file, the first line of text inserted would have the line number 0001. Thus the prompt would be :

0001

Now user can insert the desired text. At the end of the insertion process, enter CTRL-A to terminate the I command. Then the Text Editor issues its command prompt on a new line.

I # and In operate in a similar fashion in that they allow the insertion of text at specific position in the text file. I # allows insertion of text at the end of the current file, while In allows the insertion of text just before the line with line number n. If there is no file (i.e. the file is opened for the first time), it will be treated as error condition. Then the Text Editor issues the error message and returns its command prompt. Spcification of a line number which is greater than the value of the last line number in the file is also an error condition and treated in the same way.

If no errors are detected, the system displays the appropriate line number and the user can now enter the trext. (Note that the displayed line number is one greater than the last line number in the case of I # and it is n in the case of I n). As in the case of I option, at the end of the text entry, user can type CTRL-A to terminate the I command. The Text Editor then issues its command prompt on a new line.

NOTES :

1. The line number gets incremented by one every time user enters a Carriage Return.
2. The line numbers of the text following the inserted text are automatically adjusted based on the number of lines of text inserted by the user.

10.3.2. D (DELETE) COMMAND

FUNCTION :

This command allows the to delete one or more lines of text from the file currently being edited.

FORMAT :

D n <CR> or D no, n2 <CR>

OPERATION :

This command is invoked by typing D when prompted for the command by the Text Editor. "D" must be followed by one or two parameters followed by the Carriage Return.

D n <CR> deletes one line with the line number n from the text file. If n is greater than the line number of the last line in the file, an error message is displayed and then the Text Editor displays its command prompt on a new line.

D n1, n2 <CR> deletes all the lines with line numbers from n1 to n2. If n1 is greater than the line number of the last line in the file, an error message is displayed followed by the prompt of the Text Editor on a new line. If n2 is greater than the line number of the last line in the file, then n2 is replaced with the line number of the last line. If n2 is less than n1, an error message is displayed followed by the command prompt of the Text Editor on a new line.

If no errors are detected, the specified number of lines are deleted and at the end of the deletion process, the following message is displayed.

m - LINES DELETED

Where m is the number of lines deleted. Then the Text Editor's command prompt appears on the next line.

10.3.3 R (REPLACE) COMMAND

FUNCTOIN :

This command allows the user to replace exactly one line of text with a new line of text.

FORMAT :

Rn <CR>

OPERATION :

This command is invoked by typing R when prompted for command by the Text Editor. "R" should be followed by the line number of the line of text to be replaced. The line number must be followed by the Carriage Return.

If the line number specified is greater than the line number of the last line of the text, or if there is no text, the error message is displayed followed by the command prompt of the Text Editor on a new line. If no errors are detected, the systrem displays the line of text with the specified line number and prompts for new text as shown below:

```
n <present contents of this file>

n
```

User can now enter the new text. When user terminates the text entry with Carriage Return, the new line replaces the old line. Note that the length of the new line can be less than or equal to or greater than the length of the old line. The system automatically adjusts the rest of the text based on the length of the new line. After replacing the line, the Text Editor displays its command prompt on a new line.

NOTE S :

1. The Text Editor is a line-oriented one. Thus individual characters in a line can not be replaced. The entire line has to be replaced to effect the necessary changes.
2. User can abort the command by typing **CTRL-A** before typing Carriage Return. In this case the old contents remain unaltered.

10.3.4 L (LIST) COMMAND

FUNCTION :

This command allows the user to display one or more lines of text.

FORMAT :

L <CR> or Ln <CR> or L n1, n2 <CR>

OPERATION :

This command is invoked by typing L when prompted for command by the Text Editor. L can be optionally followed by one or two line numbers. The command entry is terminated by Carriage Return.

When L command is issued without any parameters, the system displays the complete file i.e it lists all the lines in the text file. If there is no file, error message is displayed and the prompt of the Text Editor is displayed on a new line.

If one optional line number is specified in the L command (L n <CR>), the system will display one line of text which has the specified line number is greater than the line number of the last line of text, an error message is displayed and the prompt of the Text Editor appears on a new line.

If two line numbers are specified (separated by a comma) in the L command (L n1, n2 <CR>), the system displays all the lines with line numbers from n1 to n2. If n2 is less than n1, an error message is displayed and the prompt of the Text Editor appears on a new line. If n2 greater than the line number of the last line, n2 is replaced by the line number of the last line.

Notes :

1. The display can be stopped temporarily by typing **CTRL-S** and it and be resumed by typing **CTRL-Q.**
2. The display can be aborted and the command terminated by typing **CTRL-A.**
3. If a Printer is attached and is enabled (Ref. section 2.1.2), then the listing appears on the Printer also.

10.3.5 N (NEW) COMMAND

FUNCTION :

This command allows the user to reinitialize the Text Editor.

FORMAT :

N <CR>

OPERATION :

This command is invoked by typing N followed by Carraige Return when prompted for command by the Text Editor.

When this command is involked, the Text Editor is reinitialized and any old file present is marked as not present. Thus the effect is an if the Text Editor is invoked for the first time after power-on condition.

10.3.6 X (EXIT) COMMAND

FUNCTION :

This command is used to terminate the operation of the Text Editor and return control to the Serial Monitor.

FORMAT :

X <CR>

OPERATION :

This command is invoked by typing X followed by Carraige Return when prompted for command by the Text Editor.

If there is no file in the next buffer when X command is issued, control immediately returns to the Serial Monitor which will issue its command prompt (".") on a new line.

If there is a file in the text buffer, the Text Editor will first display the current name of the text file and prompt for a new name as shown below :

FILE NAME = XXXX -

User can type Carriage Return to retain the name displayed or can enter a new file name followed by Carriage Return. (Ref section 10.1 for rules regarding file name.)

The system will then display the starting and ending addresses of the text file and the ending address of the text buffer as shown below :

```
STARTING ADDRESS    =  XXXX
ENDING ADDRESS      =  XXXX
BUFFER END          =  XXXX
```

This will be followed by the following query :

SAVE? (Y/N)

If the file is not to be saved on the tape, enter N. The control will then return to the Serial Monitor.

If the file is to be saved on the tape, enter Y. Note that before typing "Y", you must connect the Audio Tape Recorder to the on-board Tape Interface via the MIC jack, position the tape to the desired position and set up the Recorder in the record mode. (Refer chapter 7 on Audio Tape Interface for detals regarding the Tape Write operations.) The system will then display the following message :

WRITING ONTO TAPE

The system will record the file on the tape using the on-board Tape Interface and after completing the Tape Write operation, will return control to the Serial Monitor. The Serial Monitor will then issue its command prompt (".") on a new line.

10.4 A SAMPLE SESSION WITH THE TEXT EDITOR

Presented below is a sample session with the Text Editor, which illustrates the use of various Editor commands described above :

```
.E
ESA85-2 Text Editor V2.0

N=NEW
D=Delete
I=Insert
L=List
R=Replace
X=Exit

LOAD-

Text Buffer [0F000-0F9FF]
            Starting Address =-8000
            Ending Address =-80FF

>I
```

```
NEW FILE
0001 ORG 9000H
0002 MVI A,0AAH
0003 MOV B,A
0004 END
0005^C
```

(NOTE: CTRL-C is displayed as ^C)

```
>L

0001 ORG 9000H
0002 MVI A,0A,AH
0003 MOV B,A
0004 END

>X

FILE NAME=E7FFAAA1
        Starting Address =8000
        Ending Address =8021

BUFFER END=80FF
SAVE?(Y/N) Y
WRITING ONTO TAPE

.E

ESA85-2 Text Editor V2.0
N=NEW
D=Delete
I=Insert
L=List
R=Replace
X=Exit

LOAD-AAA1
```

```
SEARCHING THE FILE...

FILE NAME=2222
FILE NAME=3333
FILE NAME=AAA1

READING DATA FROM THE FILE

Starting Address = 8000
Ending Address = 8021

BUFFER END=80FF
BUFFER END=--

>L

0001 ORG 9000H
0002 MVI A,0AAH
0003 MOV B,A
0004 END

>R2
0001 MVI A,0AAH
0002 MVI A,55H

>L2
0002 MVI A,55H

>L1,5

0001 ORG 9000H
0002 MVI A,55H
0003 MOV B,A
0004 END

<I3
```

```
0003 NOP

0004^C

>L

0001 ORG 9000H
0002 MVI A,55H
0003 NOP
0004 MOV B,A
0005 END

>D3

0001 - LINES DELETED

>L2,5

0002 MVI A, 55H
0003 MOV B,A
0004 END

>X

FILE NAME=AAA1 -
        Starting Address = 8000
        Ending Address = 8020

BUFFER END = 80FF

SAVE? (Y/N) N
```

CHAPTER 11

EXAMPLE PROGRAMS

11.1 Program to display ESA P LTD in trainer display. Execute the Program in KEYBOARD mode only.

OUTPUT EQU 0255H

ADDRESS	OBJECT	MNENOMIC	COMMENTS
8000	C2 D5	CLR 0D5	To select Program Memory.
8002	90 80 50	MOV DPTR, #8050	
8005	12 02 55	LCALL OUTPUT	Output routine to display string of 7 characters.
8008	02 80 00	LJMP 8000	

Keep data from 8050 to 8056 Program Memory.

8050 - D6
8051 - 77
8052 - 37
8053 - 83
8054 - 87
8055 - ED
8056 - 97

The program is in a continuous loop. Press RESET key to come out of the program.

11.2 Program to perform Multiplication of two numbers. Execute this program either in KEYBOARD or in SERIAL MODE. Taking 2 numbers in 9000 & 9001 Data Memory, and storing the result in 9002 & 9003 Data Memory.

ADDRESS	OBJECT	MNENOMIC		COMMENTS
8000	90 90 01	MOV	DPTR, #9001	Keep Data in 9000 & 9001 Data Memory location.
8003	E0	MOVX	A,@DPTR	
8004	F5 F0	MOV	0F0, A	
8006	90 90 00	MOV	DPTR, #9000	
8009	E0	MOVX	A,@DPTR	
800A	A4	MUL	AB	Perform Multiplication operation.
800B	90 90 02	MOV	DPTR, #9002	Store the result in 9002 & 9003 Data Memory
800E	F0	MOVX	@DPTR,A	
800F	A3	INC	DPTR	
8010	E5 F0	MOV	A,0F0	
8012	F0	MOVX	@DPTR,A	
8013	75 A0 E1	MOV	0A0,#0E1	
8016	78 02	MOV	R0,#02	
8018	E2	MOVX	A,@R0	
8019	20 E3 03	JB	0E3H,801F	Check for Dip Switch
801C	20 00 00	LJMP	0	Jump to SER Monitor
801F	02 04 FD	LJMP	04FD	Else to K.B. Prompt

11.3 Program to perform Division of 2 numbers. Execute the program either in KEYBOARD mode or in SERIAL mode. Taking 2 numbers in 9000 & 9001 Data Memory. After division operation store the result in 9002 & 9003 Data Memory.

ADDRESS	OBJECT	MNENOMIC		COMMENTS
8000	90 90 01	MOV	DPTR, #9001	Keep Data in 9000 & 9001 Data Memory location.
8003	E0	MOVX	A, @DPTR	
8004	F5 F0	MOV	0F0,A	

ADDRESS	OBJECT	MNENOMIC	COMMENTS
8006	90 90 00	MOV DPTR,#9000	
8009	E0	MOVX A,@DPTR	
800A	84	DIV AB	Perform division operation.
800B	90 90 02	MOV DPTR, #9002	Store the result in 9002 & 9003 Data Memory
800E	F0	MOVX @DPTR,A	
800F	A3	INC DPTR	
8010	E5 F0	MOV A,0F0	
8012	F0	MOVX @DPTR,A	
8013	75 A0 E1	MOV 0A0,#0E1	
8016	78 02	MOV R0,#02	
8018	E2	MOVX A,@R0	
8019	20 E3 03	JB 0E3H,801F	Check for Dip Switch
801C	02 00 00	LJMP 0	Jump to SER Monitor
801F	02 04 FD	LJMP 04FD	Else to be K.B. Prompt

11.4 Program to Display

ELECTRON SYSTEMS ASSOCIATES PVT LTD
BANGALORE on the console

Execute this program in SERIAL mode only.

	DISPM	EQU	164BH
ADDRESS	**OBJECT**	**MNENOMIC**	**COMMENTS**
8000	C2 05	CLB 005	
8002	90 80 50	MOV DPTR, #8050	Keeping the Data in 8050 Program Memory.
8005	12 16 4B	LCALL 164B	Routine to display string of characters to console.
8008	02 00 00	LJMP 0	

```
       0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
8050:  20  20  20  20  20  0A  20  20  20  20  20  45  4C  45  43  54
8060:  52  4F  20  53  59  53  54  45  4D  53  20  41  53  53  4F  43
8070:  49  41  54  45  53  20  50  56  54  20  4C  54  44  2E  0A  0D
8080:  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20
8090:  20  20  20  20  42  41  4E  47  41  4C  4F  52  45  2E  0A  0D
80A0:      0A  0D  00
```

11.5 Program to display ASCII Character on the console. Execute this program in serial mode only.

SOUTPUT EQU 160EH

ADDRESS	**OBJECT**	**MNENOMIC**		**COMMENTS**
8000	90 90 00	MOV	DPTR,#9000	Keep the ASCII value in 9000 data Memory.
8003	E0	MOVX	A,@DPTR	
8004	12 16 0E	LCALL	SOUTPUT	
8007	02 00 03	LJMP	3	; return to serial monitor

11.6 Program to convert ASCII to its HEX equivalent on the trainer display. If the code is less than 40, then 30 is subtracted from the code to get its binary equivalent. If the code is greater than 40, then the equivalent no. lies between A & F. Execute this program in KEYBOARD mode only.

ADDRESS	**OBJECT**	**MNENOMIC**		**COMMENTS**
8000	90 89 00	MOV	DPTR, # 8900	Keep the HEX eq in 8900 Data Memory
8003	E0	MOVX	A, @DPTR	
8004	F9	MOV	R1,A	
8005	C3	CLR	C	
8006	94 0A	SUBB	A,#0A	
8008	40 0B	JC	8015	Check for carry
800A	E9	MOV	A,R1	
800B	24 37	ADD	A,#37	
800D	78 60	MOV	R0,#60	Data field addess
800F	F6	MOV	@R0,A	
8010	12 01 9B	LCALL	091B	Routine to display in data field.
8013	80 EB	SJMP	8000	
8015	E9	MOV	A,R1	
8016	C3	CLR	C	
8017	24 30	ADD	A,#30	Add Accumulator content with 30.
8019	02 80 0D	LJMP	800D	

11.7 In the given byte checking the 5th bit is ‘1’ or ‘0’. If the 5th bit is ‘1’, 00 should be stored in 8901 Data Memory or if it is '0', ffh should be stored in 8901H data memory. Execute the program in KEYBOARD mode or in SERIAL mode.

ORG 8000H

ADDRESS	OBJECT	MNENOMIC		COMMENTS
8000	90 89 00	MOV	DPTR, #8900	Store the given byte in 8900 Data Memory
8003	E0	MOVX	A, @DPTR	
8004	A3	INC	DPTR	
8005	33	RLC	A	Rotate left three times.
8006	33	RLC	A	
8007	33	RLC	A	
8008	40 06	JC	8010	Check for carry
800A	74 FF	MOV	A,#0FF	Move FF in to accumulator
800C	F0	MOVX	@DPTR,A	
800D	02 80 13	LJMP	8013	
8010	74 00	MOV	A,#00	Move 00 in to Accumulator.
8012	F0	MOVX	@DPTR,A	
8013	75 A0 E1	MOV	0A0,#0E1	Check for DIP switch
8016	78 02	MOV	R0,#02	
8018	E2	MOVX	A,@R0	
8019	20 E3 03	JB	0E3,801F	
801C	02 00 03	LJMP	3	Jump to SFR monitor
801F	02 04 FD	LJMP	04FD	Else to K.B. Prompt

11.8 Program to display largest number among ‘N’ numbers. Execute the program either in KEYBOARD mode or in SERIAL mode.

```
            ORG 8000H
PUTBYTE     EQU     185EH
UPDDT       EQU     019BH
```

ADDRESS	OBJECT	MNENOMIC		COMMENTS
8000	90 89 00	MOV	DPTR,#8900	The total no (‘N’) of data bytes is stored in 8900 data memory.

ADDRESS	OBJECT	MNENOMIC	COMMENTS
8003	E0	MOVX A, @DPTR	
8004	FA	MOV R2,A	Reg R2 is used as counter.
8005	90 89 01	MOV DPTR,#8901	Data will be put from
8008	E0	MOVX A,@DPTR	8901 onwards.
8009	1A	DEC R2	Decrement counter.
800A	F9	MOV R1,A	
800B	A3	INC DPTR	Increment data memory.
800C	E0	MOVX A,@DPTR	
800D	FB	MOV R3,A	
800E	99	SUBB A,R1	Comparing 2 numbers.
800F	50 14	JNC 8025	
8011	DA F8	DJNZ R2,800B	
8013	75 A0 E1	MOV 0A0,#0E1	Check for DIP switch.
8016	78 02	MOV R0,#02	
8018	E2	MOVX A,@R0	
8019	20 E3 0E	JB 0E3,802A	
801C	E9	MOV A,R1	
801D	F5 71	MOV 71H,A	Display the largest number on console.
801F	12 18 5E	LCALL PUTBYTE	
8022	02 00 03	LJMP 03	
8025	EB	MOV A,R3	
8026	F9	MOV R1,A	
8027	02 80 11	LJMP 8011	
802A	E9	MOV A,R1	
802B	78 60	MOV R0,#60	Display the largest number on
802D	F6	MOV @R0,A	trainer display.
802E	12 01 9B	LCALL UPDDT	

```
8031                 02  80  00          LJMP     8000
```

11.9 Program to display decimal count 0 to 20. Execute the program either in KEYBOARD mode or in SERIAL mode.

```
                         SOUTPT EQU 160EH
                         PUTBYTE EQU 185EH
                         UPDDT EQU 019BH
                         ORG 8000H

ADDRESS   OBJECT              MNEMONIC                     COMMENTS

8000      75  A0  E1          MOV     0A0H,#0E1H      Check for DIP switch
8003      78  02              MOV     R0,#02H
8005      E2                  MOVX    A,@R0
8006      20  E3  22          JB      0E3,KBD
8009      7A  00              MOV     R2,#00H         Store count 00 in R2
800B      EA         RPT :    MOV     A,R2
800C      F5  71              MOV     71H,A
800E      12  18  5E          LCALL   PUTBYTE         Routine to display on console.
8011      7B  03              MOV     R3,#03H
8013      74  08     LOOP:    MOV     A,#08H
8015      12  16  0E          LCALL   SOUTPT
8018      DB  F9              DJNZ    R3,LOOP
801A      12  80  43          LCALL   DELAY           Delay routine.
801D      12  80  43          LCALL   DELAY
8020      EA                  MOV     A,R2
8021      24  01              ADD     A,#01H
8023      D4                  DA      A               Perform Decimal Adjust.
8024      FA                  MOV     R2,A            Accumulator operation.
8025      BA  21  E3          CJNE    R2,#21H,RPT     Compare the count with 21.
8028      02  00  00          LJMP    0000H
802B      7A  00     KBD      MOV     R2,#00H
802D      EA         RPT1:    MOV     A,R2
```

ADDRESS	OBJECT				MNEMONIC		COMMENTS
802E	F5	60			MOV	60H,A	Routine to display.
8030	12	01	9B		LCALL	UPDDT	on Data field of trainer.
8033	12	80	43		LCALL	DELAY	
8036	12	80	43		LCALL	DELAY	
8039	0A				INC	R2	
803A	EA				MOV	A,R2	
803B	D4				DA	A	
803C	FA				MOV	R2,A	
803D	BA	21	ED		CJNE	R2,#21H,RPT1	
8040	02	04	FD		LJMP	04FDH	
8043	7C	FF		DELAY:	MOV	R4,#0FFH	Delay routine.
8045	7B	FF		BACK2:	MOV	R3,#0FFH	
8047	1B			BACK1:	DEC	R3	
8048	BB	00	FC		CJNE	R3,#00H,BACK1	
804B	1C				DEC4		
804C	BC	00	F6		CJNE	R4,#00H,BACK2	
804F	22				RET		

11.10 Program to display 24 hours digital clock in keyboard MODE. Execute the program from 8900. To change the Min, Hours change the data in the location 8903 and 8901 respectively.

```
UPDAD    EQU      020BH
UPDDT    EQU      019BH
```

ADDRESS	OBJECT				MNEMONIC		COMMENTS
8900	7D	23			MOV	R5,#23H	Store Hours in R5 Reg.
8902	7F	58			MOV	R7,#58H	Store Minutes in R7 Reg.
8904	ED			LOOP:	MOV	A,R5	
8905	F5	61			MOV	61H,A	
8907	EF				MOV	A,R7	
8908	F5	60			MOV	60H,A	
890A	12	02	0B		LCALL	UPDAD	Routine to Invoke Address field.

```
890D    79  00       UP :  MOV    R1,#00H         Store Seconds in R1 Reg.
890F    E9           RPT   MOV    A,R1
8910    FE                 MOV    R6,A
8911    F5  60             MOV    60H,A
8913    12  01  9B         LCALL  UPDDT           Routine to Invoke Data
                                                  field.
8916    12  89  44         LCALL  DELAY
8919    EE                 MOV    A,R6
891A    E9                 MOV    A,R1
891B    24  01             ADD    A,#01H
891D    D4                 DA     A
891E    F9                 MOV    R1,A
891F    B9  60  ED         CJNE   R1,#60H,RPT     compare 60 sec is over or not.
8922    74  00             MOV    A,#00H
8924    F5  60             MOV    60H,A
8926    12  01  9B         LCALL  UPDDT
8929    EF                 MOV    A,R7
892A    24  01             ADD    A,#01H
892C    D4                 DA     A
892D    FF                 MOV    R7,A
892E    BF  60  26         CJNE   R7,#60H,MIN     Compare 60 Minutes is
                                                  over or not.
8931    74  00             MOV    A,#00H
8933    F5  60             MOV    60H,A
8935    ED                 MOV    A,R5
8936    24  01             ADD    A,#01H
8938    D4                 DA     A
8939    FD                 MOV    R5,A
893A    BD  24  23         CJNE   R5,#24H,HRS     Compare 24 hours is
                                                  over or not.
893D    7D  00             MOV    R5,#00H
893F    7F  00             MOV    R7,#00H
```

```
8941    02 89 04            LJMP    LOOP            Jump to loop
8944    7A 04       DELAY:  MOV     R2,#04H         To provide 1 sec delay
8946    7C FF       BACK3:  MOV     R4,#0FFH
8948    7B FF       BACK2:  MOV     R3,#0FFH
894A    1B          BACK2:  DEC     R3
894B    BB 00 FC            CJNE    R3,#00,BACK1
894E    1C                  DEC     R4
894F    BC 00 F6            CJNE    R4,#00,BACK2
8952    1A                  DEC     R2
8953    BA 00 F0            CJNE    R2,#00,BACK3
8956    22                  RET
8957    EF            MIN:  MOV     A,R7            subroutine to display
                                                    minutes.
8958    F5 60               MOV     60H,A
895A    12 02 0B            LCALL   UPDAD
895D    02 89 0D            LJMP    UP
8960    ED            HRS:  MOV     A,R5            Subroutine to display hours
8961    F5 61               MOV     61H,A
8963    12 02 0B            LCALL   UPDAD
8966    7F 00               MOV     R7,#00H
8968    02 89 0D            LJMP    UP
```

11.11 Program to display 24 hours digital clock in SERIAL mode. Execute the program from 8000h. To change theMIN, HOURS change the data in location 8009h & 8007h respectively.

```
PUTBYTE EQU 185EH
SOUTPT EQU 160EH
DISPM EQU 164BH
```

```
ADDRESS OBJECT              MNENOMIC                COMMENTS

8000    90 80 DD            MOV     DPTR,#DAT       Keep string 'HRS MIN SEC'
                                                    in program memory.
8003    12 16 4B            LCALL   DISPM           dispaly routine.
8006    7F 23       START:  MOV     R7,#23H         Store hours in R7 reg.
```

```
8008      7E  58              MOV    R6,#58H        Stores Minutes in R6 Reg.
800A      EF          LOOP:   MOV    A,R7
800B      F5  71              MOV    71H,A
800D      12  18  5E          LCALL  PUTBYTE        Routine to put character on
                                                    console.
8010      74  20              MOV    A,#20H         Provide space.
8012      12  16  0E          LCALL  SOUTPT
8015      EE                  MOV    A,R6
8016      F5  71              MOV    71H,A
8018      12  18  5E          LCALL  PUTBYTE
801B      74  20              MOV    A,#20H
801D      12  16  0E          LCALL  SOUTPT
8020      7A  00      BEGIN:  MOV    R2,#00H        Keep seconds 00
8022      EA          SEC:    MOV    A,R2           In R2 reg.
8023      F5  71              MOV    71H,A
8025      12  18  5E          LCALL  PUTBYTE
8028      12  80  89          LCALL  DELAY          Delay routine.
802B      12  80  89          LCALL  DELAY
802E      7B  03              MOV    R3,#03H
8030      74  08      RPT:    MOV    A.#08H
8032      12  16  0E          LCALL  SOUTPT
8035      DB  F9              DJNZ   R3,RPT
8037      EA                  MOV    A,R2
8038      24  01              ADD    A,#01H
803A      D4                  DA     A
803B      FA                  MOV    R2,A
803C      BA  60  E3          CJNE   R2,#60H,SEC    Check for 60 sec over.
                                                    or not.
803F      74  00              MOV    A,#00H
8041      F5  71              MOV    71H,A
8043      12  18  5E          LCALL  PUTBYTE
8046      EE                  MOV    A,R6
```

```
8047    24  01              ADD     A,#01H
8049    D4                  DA      A
804A    FE                  MOV     R6,A
804B    BE  60  52          CJNE    R6,#60H,MIN      Check for 60 minutes over
                                                     or not.
804E    7B  07              MOV     R3,#07H
8050    74  08      RPT1:   MOV     A,#08H
8052    12  16  0E          LCALL   SOUPT
8055    DB  F9              DJNZ    R3,RPT1
8057    74  00              MOV     A,#00H
8059    F5  71              MOV     71H,A
805B    12  18  5E          LCALL   PUTBYTE
805E    7B  07              MOV     R3, #07H
8060    74  08      RPT2:   MOV     A,#08H
8062    12  16  0E          LCALL   SOUTPT
8065    DB  F9              DJNZ    R3, RPT2
8067    EF                  MOV     A,R7
8068    24  01              ADD     A,#01H
806A    D4                  DA      A
806B    FF                  MOV     R7,A
806C    BF  24  52          CJNE    R7,#24H,HRS      check for 24 hours
                                                     over or not.
806F    7B  04              MOV     R3,#04H
8071    74  08      RPT3:   MOV A,#08H
8073    12  16  0E          LCALL   SOUTPT
8076    DB  F9              DJNZ    R3, RPT3
8078    74  00              MOV     A,#00H
807A    F5  71              MOV     71H,A
807C    12  18  5E          LCALL   PUTBYTE
807F    7F  00              MOV     R7,#00H
8081    7E  00              MOV     R6,#00H
8083    12  80  B7          LCALL   CURSOR
```

```
8086        02  80  0A             LJMP   LOOP
8089 : Delay subroutine
8089        7D  C0      DELAY:     MOV    R5,#0C0H         This routine will provide
                                                           1sec delay.
808B        7C  FF      BACK2:     MOV    R4,#0FFH
808D        90  00  00             MOV    DPTR,#0H
8090        A3          GOBACK:    INC    DPTR             Delay routine.
8091        E5  82                 MOV    A,82H
8093        45  82                 ORL    A,82H
8095        70  F9                 JNZ    GOBACK
8097        1C          BACK1:     DEC    R4
8098        BC  00  FC             CJNE   R4,#00H,BACK1
809B        1D                     DEC    R5
809C        BD  00  EC             CJNE   R5,#00H,BACK2
809F        22                     RET
80A0 : Subroutine to display minutes.
80A0        7B  07        MIN:     MOV    R3,#07H
80A2        74  08        RPT4:    MOV    A,#08H
80A4        12  16  0E             LCALL  SOUTPT
80A7        DB  F9                 DJNZ   R3,RPT4
80A9        EE                     MOV    A,R6
80AA        F5  71                 MOV    71H,A
80AC        12  18  5E             LCALL  PUTBYTE
80AF        74  20                 MOV    A,#20H
80B1        12  16  0E             LCALL  SOUTPT
80B4        02  80  20             LJMP   BEGIN
80B7 : Subroutine to move the cursor three times back.
80B7        7B  03      CURSOR:    MOV    R3,#03H
80B9        74  08      RPT5:      MOV    A,#08H
80BB        12  16  0E             LCALL  SOUTPT
80BE        DB  F9                 DJNZ   R3,RPT5
80C0        22                     RET
```

80C1 : Subroutine to display hours.

```
80C1        FF                HRS:  MOV     R7,A
80C2        F5   71                 MOV     71H,A
80C4        12   18   5E            LCALL   PUTBYTE
80C7        74   20                 MOV     A,#20H
80C9        12   16   0E            LCALL   SOUTPT
80CC        74   00                 MOV     A,#00H
80CE        F5   71                 MOV     71H,A
80D0        12   18   5E            LCALL   PUTBYTE
80D3        7E   00                 MOV     R6,#00H
80D5        74   20                 MOV     A,#20H
80D7        12   16   0E            LCALL   SOUTPT
80DA        02   80   20            LJMP    BEGIN
```

80DD : Subroutine to display HRS,MIN SEC on the console.

```
80DD      48 52 53 20 4D        DAT:    DB 48H, 52H, 53H, 20H, 4DH
80E2      49 4E 20 53 45                DB 49H, 4EH, 20H, 53H, 45H
80E7      43 0D 0A 00                   DB 43H, 0DH, 0AH, 00H
80EB      02 80 06                      LJMP START
                                        END
```

11.12 Program to perform Addition of two numbers. This progran can be executed either in SERIAL mode or in KEYBOARD mode. Two numbers are taken from locations 9000 & 9001 of Data Memory. They are added and the result is stored in the location 9002 of Data Memory.

```
ADDR   OBJECT         LABLE   MNEMONIC                      COMMENTS

                              SYMBOLS
8000                          ORG 8000H
8000   90   90   00   START:  MOV      DPTR,#9000H      ;   KEEP DATA IN 9000
8003   E0                     MOVX     A,@DPTR              AND 9001 LOCATIONS
8004   F5   F0                MOV      0F0H,A               OF DATA MEMORY.
8006   90   90   01           MOV      DPTR,#9001H
8009   E0                     MOVX     A,@DPTR
800A   25   F0                ADD      A,0F0H           ;   ADD THEM.
800C   90   90   02           MOV      DPTR,#9002H      ;   STORE THE RESULT IN
800F   F0                     MOVX     @DPTR,A          ;   9002 LOCATION OF
8010   75   A0   E1           MOV      0A0H,#0E1H       ;   DATA MEMORY.
8013   78   02                MOV      R0,#02
```

```
8015    E2                       MOVX    A,@R0
8016    20  E3  03               JB      0E3H,L1         ;  READ DIP SWITCH
8019    02  00  00               LJMP    0               ;  JUMP TO SER. MONITOR
801C    02  04  FD         L1:   LJMP    04FDH           ;  ELSE TO K.B.PROMPT.
801F                             END
```

11.13 Program to perform subtraction of two numbers. This progran can be executed either in SERIAL mode or in KEYBOARD mode. Two numbers are taken from locations 9000 & 9001 of Data Memory. They are subtracted and the result is stored in the location 9002 of Data Memory.

```
ADDR    OBJECT          LABLE    MNEMONIC                   COMMENTS

                                 SYMBOLS
8000                             ORG     8000H
8000    90  90  01      START:   MOV     DPTR,#9001H     ;  KEEP DATA IN 9000
8003    E0                       MOVX    A,@DPTR         ;  AND 9001 LOCATIONS
8004    F5  F0                   MOV     0F0H,A             OF DATA MEMORY.
8006    90  90  00               MOV     DPTR,#9000H
8009    E0                       MOVX    A,@DPTR
800A    95  F0                   SUBB    A,0F0H          ;  SUBSTRACT THEM.
800C    90  90  02               MOV     DPTR,#9002H     ;  STORE THE RESULT IN
800F    F0                       MOVX    @DPTR,A         ;  9002 LOCATION OF
8010    75  A0  E1               MOV     0A0H,#0E1H      ;  DATA MEMORY.
8013    78  02                   MOV     R0,#02
8015    E2                       MOVX    A,@R0
8016    20  E3  03               JB      0E3H,L1         ;  READ DIP SWITCH
8019    02  00  00               LJMP    0               ;  JUMP TO SER. MONITOR
801C    02  04  FD         L1:   LJMP    04FDH           ;  ELSE TO K.B.PROMPT.
801F                             END
```

CHAPTER 12

DISASSEMBLER

12.1 INTRODUCTION

Disassembly is an extremely useful technique, often employed during debugging. A Disassembler converts machine language codes into Assembly language mnemonics, making it easy for the user to understand/verify the program.

The Disassembler provided by ESA85-2 is available only when the kit is in serial mode of operation. This disassembler is a two- pass one. During the first pass, all the addresses to which JMPs/CALLs are made,are identified and converted to labels. The label consists of the letter "L" immediately followed by the address represented by four ASCII characters. During the second pass, the codes are disassembled and displayed in mnemonic form. Each instruction to which JMP/ CALL is made will have a label name followed by a colon in the label field. The operand field of instructions referring to such labelled instructions will have the label name (instead of the hexadecimal address value). For example:

```
L8120 :          DCR      A
                 JNZ      L8120
```

12.2 OPERATION

To use this facility, type Z when prompted for command by the Serial Monitor. Immediately, the system displays the following sign-on message:

ESA85-2 Disassembler Vx.y

Then it prompts for the starting address of the program to be disassembled.

Starting Address = ---

Type in the starting address for the assembled code that has to be disassembled. Terminate the address with the RETURN key. Now you will be prompted with

Ending Address = ---

Enter the program's ending address followed by RETURN.

The default location of the label is FA00H to FDFFH. Note that each label requires 2 bytes of storage. Thus the default allocation allows 512 labels. User can alter the location and size of the Label Table if required. Otherwise the default values can be choosen by typing RETURN to the prompts for Start and End addresses of the Label Table. The prompts appear as shown below:

Label Table [FA00-FDFF]

Starting Address = ---

Enter RETURN to retain the default address of FA00H or enter a new starting address followed by RETURN. The system now prompts:

Ending Address = ---

Enter RETURN to retain the default address of FDFFH or enter a new ending address followed by RETURN. Note that if the user enters new values, then the Ending Address must be greater than or equal to the starting address; otherwise the system prompts again for the starting and Ending Addresses.

After obtaining the relevant parameters as described above, the system begins the disassembly operation. First it displays the following message:

Starting Pass 1

After completing pass1, it displays the following message:

Starting Pass 2

After completing pass 2, the system will display the display column headers as

ADDRESS CODE LINE LABEL MNEMONIC OPERAND

The disassembled code is then displayed according to the above format.

The display can be halted at any point by **CTRL-S** and restarted by **CTRL-Q.** The disassembly can be aborted at any time by **CTRL- A.**

At the end the system will display

End of Disassembly

and the control will be returned to the Serial Monitor.

NOTE :

If the disassembly of the last instruction requires reading of data from locations beyond the specified end address, the system will read them to complete the disassembly. For example, if the specified End address is 81FFH and the code at 81FFH is C2H (which is a 3-byte instruction), the system will read the required data from locations 8200H and 8201H to complete the disassembly.

ERROR MESSAGES

1. **Label Table Exhausted :** This is displayed before the start of Pass - 2 When the number of labels is greater than what can be accommodated in the Label Table. The Pass - 1 is aborted as soon as this occurs , but Pass-2 goes on with some of the internal references going unlabelled. This message stays on the screen for about 10 seconds and Pass 2 starts thereafter.
2. **Invalid code XX @ XXXX H:** This is displayed if an invalid opcode is encountered. The message lists the illegal opcode and its location. This message is first displayed in Pass-1 and then pass-1 is aborted and Pass-2 is started. The same message is again displayed in Pass-2, with disassembled codes displayed upto the address where the invalid code is encountered and then the disassembly process is aborted and control returns to the Serial Monitor.

EXAMPLE 1 : Assume the contents of locations 8000H to 8006H are:

3EH, 50H, 3DH, 00H, C2H, 02H, 80H.

```
. Z

      ESA85-2 Disassembler Vxy

                  Starting Address = 8000<CR>
                  Ending Address = 8006<CR>
```

Label Table [FA00-FDFF]

Starting Address = <CR>
Ending Address = <CR>

Starting Pass-1
Starting Pass-2

ADDRESS	CODE	LINE NUMBER	LABEL	MNEMONIC	OPERAND
8000	3E,50	0001		MVI	A,50H
8002	3D	0002	L8002	DCR	A
8003	00	0003		NOP	
8004	C2, 02, 80	0004		JNZ	L8002

End of Disassembly.

EXAMPLE 2 : Assume the contents of locations 9000H to 900AH are 21H, 00H, 88H, 31H, 00H, C0H, 08H, 00H, 01H, 60H, 88H.

. Z

ESA85-2 Disassembler Vx.y

Starting Address = --- 9000 <CR>
Ending Address = --- 900A <CR>

Label Table [FA00-FDFF]

Starting Address = --- <CR>
Ending Address = --- <CR>

Starting Pass-1

Invalid code 08H @ 9006H

Starting Pass-2

Invalid code 08H @ 9006H

ADDRESS	CODE	LINE NUMBER	LABEL	MNEMONIC	OPERAND
9000	21,00,88	0001		LXI	H,8800H
9003	31,00, C0	0002		LXI	SP,C000H

End of Disassembly

CHAPTER 13

COMMUNICATION WITH A HOST COMPUTER SYSTEM

13.1 INTRODUCTION

As already noted, ESA 85-2 operating in the serial mode, can be connected to either a CRT terminal or a host computer system. When a computer system is the controlling element, it must be executing a driver software to communicate with ESA 85-2.

XT852 is such an optional communication package which allows the user to establish a communication link between the ESA 85-2 trainer and a PC/XT/AT compatible computer system. The link is established between asynchronous serial ports of the computer(COM1/COM2) and ESA 85-2.

XT852 is supplied as a '.EXE' file on a 3½" DSHD diskette and can be executed on a PC/XT/AT compatible computer system under PC-DOS/MS-DOS operating system. A suitable RS 232 C cable has to be used for connecting ESA 85-2 to a PC/XT/AT compatible system.

XT852 fully supports all the commands of ESA 85-2. Further, it allows the contents of a disk file to be downloaded from the computer system into the memory of ESA 85-2. User can develop assembly language programs on the PC/XT/AT compatible computer system, cross-assemble them using a suitable cross- assembler to generate object code files and then use XT852 to download these object code files into ESA 85-2 for execution. Thus the extensive development facilities available on the PC/XT/AT compatible systems can be used to complement the facilities available on ESA 85-2.

Further XT852 allows uploading of data from the memory of ESA 85-2 to the disk file of the computer. Thus this facility can conveniently be used to save user programs.

13.2 INSTALLATION

NOTE:

Make sure that you have made a backup copy of XT 852.EXE before proceeding with the installation.

The detailed installation procedure is as follows:

a) Configure ESA 85-2 for serial mode of operation and set the serial port of ESA 85-2 for 9600 Baud and No parity (Refer sections 2.1.1 and 2.1.3).

b) Connect the computer system to ESA 85-2 over the COM1/COM2 serial port (Refer to Technical Manual of your system for details regarding the signal definitions on COM ports). The signal definitions of the RS 232 C port of ESA 85-2 can be found in Appendix E.

c) Insert the diskette containing the file XT852.EXE into the available drive and run the program by typing XT852 or XT852/B to select Black and White mode if computer system has a CGA monitor.

d) Now the following message appears on the screen.

XT852 Version 1.1

ELECTRO SYSTEMS ASSOCIATES PVT LTD
BANGALORE

Alt+S	---	Set Communication Parameters
Ctrl+F1	---	Help
Alt+F1	---	Command Help
<Esc>	---	Clear Command
<F1>	---	Previous Command Character
<F3>	---	Command Recall
Ctrl+U	---	Upload Command
Ctrl+D	---	Download Command

! Command	---	DOS Shell/Command
Alt+X	---	Exit

Press any key to continue

e) XT852 checks for the presence of communication ports COM1 and COM2. If both ports are not available it will display the message.

No serial port present as reported by BIOS

and exits to DOS. Otherwise XT852 will read the communication parameters from file "XT852.INS" and initializes the communication port. XT852 searches current directory for file "XT852.INS". If search fails, it will search the path given by the DOS environment variable INIT. If the file is not present, following message is displayed

XT852.INS file not found !

Serial parameters are set to COM1, 9600, 8, 2, None

Do you want to change?

Yes No

If option "No" is selected the communication parameters: Serial Port COM1, Baud 9600, Data bits 8, Stop bits 2, Parity None are set. If option "Yes" is selected the communication parameters can be interactively modified as described in section 13.4.6.

Now XT852 attempts to establish communication between the computer and ESA 85-2. If successful, the command prompt "." appears on the screen. Subsequently during the power-on or RESET of the trainer, the sign-on message "ESA 85-2 SERIAL MONITOR Vx.y" appears on the screen followed by command prompt ("."). The word "SErIAL" will be displayed on the trainer's keyboard display. Otherwise it will display the message.

Unable to transmit the data

Retry or Ignore

If ESA 85-2 is not powered on, power it on and press <<R>> to retry to establish the communication. Pressing <I> will exit XT852 to DOS. Now check for the following :

a) Ensure that ESA 85-2 is connected to the correct COM port and that the COM port is functioning properly.
b) Ensure that ESA 85-2 is functioning properly and configured correctly.
c) Check the RS 232 C cable and its connections.

Since the communication package utilizes the hardware handshake signal DTR while communicating with ESA 85-2, the interfacing cable must support this signal also.

Note :

XT852 utilizes an interrupt driven routine for reading characters from the COM port. Thus it is possible for XT852 to miss some characters if the system has any resident programs which are interrupt driven. (For example, many systems include a CLOCK program in the AUTOEXEC file, to display the time on the upper right corner of the screen.) Hence it is desirable not to run any such resident programs while XT852 is running.

If the problem persists, Please contact the manufacturer.

13.3 RETURNING TO DOS

User can terminate XT852 and return control to DOS by typing Alt+X when the program is waiting for a keyboard input.

13.4 OPERATIONAL DETAILS

The complete command set of the ESA 85-2 is transparent and is fully supported by XT852 (Refer to chapter 4 for the serial monitor mode commands). Press F1 for help command.

In addition, XT852 supports the file download, file upload and other commands which are explained below.

NOTE: During parameter entry, the system expects the alphabetic characters to be in upper case. Thus it is convenient to use the keyboard with the CAPS LOCK on.

13.4.1 DOWNLOAD OPERATION:

This feature allows downloading of the contents of an object code file into the memory of ESA 85-2.

NOTE: The object code file must be a ".HEX" file with records in INTEL 8-Bit HEX format. Please refer to the relevant INTEL manuals for the definition of INTEL 8-Bit HEX format. Most of the cross assemblers for 8085 do produce object code files which are ".HEX" files with records in INTEL 8-Bit HEX format.

To perform download operation, press Ctrl+D in response to the command prompt (".").

The system will now prompt for the name of the disk file, from which the information is to be downloaded. The prompt is as follows:

Download filename [.HEX]:

Enter the file name with extension, terminated by <CR>. If the filename is invalid, it displays "File not found!" and prompts again for the filename. If the path specified is invalid, it displays a message "Path not found!" and prompts again for the filename. If none of the above errors occur, the system will prompt for the starting address of the program as follows:

Start Address :

Enter the starting address followed by <CR>. A maximum of four hex digits are allowed for the starting address. Now the system will prompt for the ending address as follows:

End Address :

Enter the starting address followed by <CR>. A maximum of four hex digits are allowed for the ending address. Now the system will prompt for the load offset value as follows:

Load offset value :

If the user wishes to download the file to an address range different from the actual address range of the file, then a suitable offset value can be entered to enable the file to be downloaded to the desired address range. For Example if the starting address of a file is 8000H and Ending address is 80FFH and if the user wishes to download it to an address range starting at C000H, the user has to enter an offset value of C000H-8000H=4000H. In case the user wishes to download to the same address range an offset value of 0000 has to be entered or simply press <CR>.

After obtaining the Filename, Starting address, the Ending address and the Load offset value, the system will read the file, gather the data in the specified address range, reformat the data for compatibility with the protocol required by ESA 85-2 and send the data to ESA 85-2.

While the downloading is going on, the system will display the following message:

Downloading in Progress..XXXX.

After downloading is over, the system returns the command prompt of ESA 85-2. It also displays the starting address of each record being downloaded.

13.4.2 UPLOAD OPERATION :

This feature allows uploading of the data from the memory of ESA 85-2 to the computer system and saves the data in the specified disk file in INTEL 8-Bit HEX format.

To perform upload operation, press Ctrl+U in response to the command prompt (".").

The system will now prompt for the name of the disk file, into which the information is to be uploaded. The prompt is as follows:

Upload filename [.HEX] :

Enter the file name with extension, terminated by <CR>. If the file already exists, then the system will display

File already exists!

Overwrite?

<u>Y</u>es <u>N</u>o <u>A</u>ppend

Select the first option by pressing <Y> to overwrite the contents of the existing file. Pressing <N> will let the user specify another filename. Select the third option <A> to append to the contents of the existing file.

If no error occurs, the system will prompt for starting address of the program as follows:

Start Address =

Enter the starting address terminated by <CR>. A maximum of four hex digits are allowed for the starting address.

Now the system will prompt for the ending address as follows:

End Address =

Enter the ending address terminated by <CR>. A maximum of four hex digits are allowed for the ending address.

After obtaining the filename, starting address and the ending address, the system will gather the data in the specified address range of the ESA 85-2, reformat the data into INTEL 8-bit HEX records and store the data in the specified file.

Uploading in progress XXXX

While the uploading is going on, the system will display the starting address (XXXX) of each record being uploaded. Once the uploading is complete XT852 will let the user specify another address range. User may specify a new address range or enter <Esc> to terminate uploading operation.

The following error messages may appear during upload and download operations.

1. Invalid function number !

- This is XT 852 internal error, therefore contact ESA technical support for Assistance.

2. File not found !
3. Path not found !
4. No more files !
5. Access denied !
6. Invalid file handle !
7. Insufficient Disk space !
8. Unable to continue upload !
9. Colon is not present at the start of the Record !
10. Invalid data in (source file) the following Record !
11. Checksum Error in the following Record !

13.4.3 DOS COMMANDS :

At the command prompt "." any valid DOS command can be entered preceded by "!". XT852 environment is saved and the DOS command is executed. Then XT852 environment is restored and XT852 command prompt is displayed again.

13.4.4 BOTTOM LINE :

During the session, XT852 displays many of the XT852 commands at the bottom line in reverse video for the convenience of user. The bottom line is displayed as

Ctrl+F1 Help, Alt+F1 CmdHelp, Alt+S Commn, <Esc> ClrCmd, Alt+X Exit, F1, F3, ↑, ↓

13.4.5 COMMAND RECALL :

This feature facilitates recall of the commands already entered by the user, upto a maximum of 16 commands. Press <F3>to recall the previous command. All the commands are kept in a circular buffer, User may use Up-arrow and Down-arrow keys to traverse through the sequence of commands in backward or forward direction respectively in a circular fashion. User may recall just the previous command, character by character, by pressing <F1>desired number of times. Current command being entered can be cleared by using <Esc>key any time before pressing <CR>.

13.4.6 COMMUNICATION :

Communication parameters can be set during the session by pressing Alt+S. List of parameters and their current values will appear on the screen. Select the desired parameter with the help of arrow keys and keep the space bar <SP> pressed till the desired value appears. The parameters allowed to be set are Communication Port (COM1/COM2), Baud Rate (110 / 150 / 300 / 600 / 1200 / 2400 /4800 / 9600), Number of Data bits (7/8), Number of Stop bits (1/2) and Parity (NONE/ODD/EVEN). After selecting the desired values press <CR> to set the parameters or press <Esc> to ignore the values.

Communication parameters can also be modified, while user is in DOS by editing the file XT852.INS. This file contains single data line, having five integers separated by blanks, representing various communication parameters. These five integers represent serial communication port, baud rate, number of data bits, number of stop bits and parity, in sequence. Table 13.1 shows details of the integer values and corresponding parameters.

Commn. Port	int1	Baud Rate	int2	Data Bits	int 3	Stop Bits	int 4	Parity	int
COM1	0	110	0	7	0	1	0	odd	0
COM2	1	150	1	8	1	2	1	none	1
		300	2					even	2
		600	3						
		1200	4						
		2400	5						
		4800	6						
		9600	7						

Table 13.1

13.4.7 HELP:

On-line help is available for all ESA 85-2 monitor commands and specific commands of XT852. Help facility can be selected by Ctrl+F1. A menu of commands is displayed from which desired command can be selected by using arrow keys and help information about that command is displayed in the remaining part of the screen. Context sensitive help is available using Alt+F1. This facility can be used if more information is desired about the command being entered against command prompt.

CHAPTER 14

PROGRAMMING EXAMPLES

14.1 INTRODUCTION

In this chapter, we will describe some programming examples which can be run on the ESA85-2 System. The first set of examples are designed to illustrate the use of various command keys and the second set of examples are designed to illustrate the use of monitor routines. The user, encountering a microprocessor trainer for the first time, is strongly urged to go through this chapter in detail, load and execute the programs as indicated. The experienced user can skip this section and directly refer to section 14.3 to become familiar with the use of monitor routines.

14.2 PROGRAMMING EXAMPLES

EXAMPLE 1: The following program is a very short and simple one. This program loads the registers B and C with specific values and stores a specific value in a RAM location. Assume the program is to be loaded from 8800 H. The program and the hand assembly in shown below:

LOCATION	CONTENTS	MNEMONIC	COMMENTS
8800	06 22	MVI B, 22H	;Initialize Reg B
8802	0E 82	MVI C, 82H	;Initialize Reg C
8804	78	MOV A,B	
8805	32 40 88	STA 8840	;Load the RAM location
8808	DF	RST 3	;Return control to the monitor

Thus the sequence of 9 bytes to be stored from location 8800 H through 8808 H is - 06, 22, 0E, 82, 78, 32, 40, 88, DF. Enter these using EXAM MEM command. After entering the last byte into the location 8808H, press the EXEC as the delimiter. Note that EXEC is only a delimiter key and is not used for executing the user program. The command to be used for executing user programs is the GO command. Using this command, execute the above program. After execution, control returns to the monitor. Note that this happens because of the RST3 instruction. When control returns to the monitor via the RST3 instruction, the monitor first saves the complete user context.

Now use the EXAM REG key to observe the contents of the registers B and C. They will be 22 and 82. Using the EXAM MEM command, observe the contents of the location 8840 H to be 22 H indicating the successful execution of the program.

Now press the RESET key and observe the contents of the locations 8808H and 8840 H. The contents remain undisturbed. Pressing the RESET key does not disturb the user portion of the RAM. But now, examine the B and C registers. It is likely that they will not contain the values 22H and 82 H. If you press the RESET key, user register values are not guaranteed to remain unaltered.

Example 2: The following program converts two BCD digits stored in memory to a binary number and stores the binary number in memory. It is assumed that the most significant BCD digit is at the location 8840H and next BCD digit is in the next location i.e. in 8841 H. The equivalent binary number is to be stored in location 8842 H.

The program first multiplies the most significant BCD digit by 10 and then adds the second BCD digit to get the Binary number. The multiplication by 10 is implemented as repeated addition:

10xa = (8xa) + (2xa) = [(4xa) + (4xa)] + (2xa)

	MNEMONIC	COMMENTS
BCDBIN:	LXI H, 8840	;Point in BCD string
	MOV A,M	;Get most significant
		;BCD digit
	ADD A	;MSD x 2
	MOV B,A	;Save it
	ADD A	;MSD x 4
	ADD A	;MSD x 8
	ADD B	;(MSD x 8) + (MSD x 2)
	INX H	
	ADD M	;Add next BCD digit to get
		;binary equivalent
	INX H	
	MOV M,A	;and store it
	RST 3	;Return to monitor.

Suppose you decide to hand-assemble this program to start from location 8800 H. Assume we get the following sequence of bytes:

21, 40, 88, 7E, 87, 47, 87, 80, 23, 86, 23, 77, DF. Enter these 13 (decimal) bytes into locations 8800 H through 880C H.

Set up two BCD digits in locations 8840 H and 8841 H -say 4 and 5.

Now execute the program using the GO command. After the execution of the program, control returns to the monitor and a command prompt dash will be displayed. Now using EXAM MEM command, examine the contents of the location 8842 H. Surprise, Instead of the expected 2D, we find 1D. So we should again check the program.

After going through the program again, we find the logic to be correct. The next step is to check the coding. Now, we find (fortunately) that, while there are two ADD A instructions after MOV B, A instruction, our code sequence has only one ADD A instruction. (After opcode 47H in location 8805 H, there should be 87 H, 87 H and then 80 H; but we have only one 87 H and then 80 H). So instead of multiplying MSD by 10, we multiplied it by 6 (=4+2). To correct this, we must insert an instruction of one-byte length (ADD A) at location 8807 H. We can use the INSERT command for this purpose. The required key sequence would be

```
INSERT <8800> NEXT <880C> NEXT <8807>
```

NEXT <1> NEXT <87> EXEC.

Now using EXAM MEM command, observe the contents of the locations 8800 H through 880D H and verify that the program has been entered correctly. Now execute it. Examine the location 8842 H. Fine, we get the correct result of 2D.

Example 3: 8085 has no instruction for multiplication. One simple way to implement multiplication is by repeated addition. Here, if you want to multiply m by n, then you clear the result and add the multiplicand m to the result n times, where n is the multiplier. Thus 2 x 3 is implemented as [[(0+2) + 2] +2] = 6. The following program multiplies any two one byte numbers, but assumes that the numbers are unsigned.

; Name :	USMULT (Unsigned Multiplication)
; Function:	This routine performs the multiplication of two unsigned one- byte integers and returns the two-byte result.
; Inputs:	(C) = multiplicand
	(B) = multiplier
; Outputs:	(H,L) = product
; Registers Destroyed:	A,E,H,L, Flags

```
USMULT:         LXI H,0000H         ;Clear the result
                MOV E,B             ;Save Multiplier and
                MVI B,00H           ;clear B. This is done, so
                                    ;that we can use DAD
                                    ;instruction.
MULIO:          MOV A,E
                ORA A               ;Multiplier - 0?
                JZ OVER             ;Yes - Jump to finish
                DAD B               ;No - Add multiplicand again
                DCR E               ;Decrement multiplier and
                                    ;repeat
                JMP MULIO
OVER:           RST 3               ;Return to monitor
```

Now you can select the starting location for this program and hand-assemble it. Suppose, we want to load this program from 8800 H. Then assembling the above program, let us say, we get the following machine codes.

LOCATION	CONTENTS	LOCATION	CONTENTS	LOCATION	CONTENTS
8800	21	8801	00	8802	00
8803	58	8804	06	8805	00
8806	79	8807	B7	8808	CA
8809	10	880A	88	880B	09
880C	1D	880D	C3	880E	06
880F	88	8810	DF		

Now, using the EXAM MEM command enter the above program. After entering the last byte into location 8810 H, you press the EXEC key. Note that EXEC key is only a delimiter and it is not a command to start the execution of user program. When you press the EXEC key, after entering the last byte, the EXAM MEM command is terminated and monitor returns to the command entry mode i.e. the display is cleared and a dash appears in the left most digit position of the display.

Now we have to set up the parameters for this program. The parameters are the multiplicand and the multiplier. Suppose, we select their values as 1A H and 1 A H. They must be entered into the registers B and C. Use EXAM REG command to enter these values.

Now, we are ready to execute the program. Note that if this program is part of a bigger program, which usually will be the case, then the last instruction would have been a RET instruction. But here, we write RST3 instruction, so that control is returned to the monitor. Then, we can examine the results. To start the execution of the program use GO key and provide the starting address - 8800.

Now the display is cleared, and E appears in the left- most digit of display. This always indicates that a user program is running. But, now what is happening? The ‘E’ has remained there. We know that when control returns to the monitor via RST3 instruction, the monitor will display the command prompt and sign-on message. This present program is extremely short one and should not take more than a few milliseconds. Thus, E should be displayed only for a few milliseconds and then a dash should appear. In fact, we should not be able to notice E at all, because it appears for such a short period. Thus it is clear that the program is caught in infinite loop and control is never returning to monitor. Now is the time for some debugging. The first thing to do of course, is to recover from the erroneous program i.e. allow the monitor to gain control. The only way to do it now would be to press the RESET key. (Note that this type of situation is almost the only situation warranting the use of RESET key, once you initialise the system).

Now go through the code carefully to see if there are any coding mistakes. Assume we find one.

The next approach can be the single step command. So reinitialize the parameters (multiplicand and multiplier) and use the single step command to step through the program. After executing the first three instructions check the registers B, E, H and L and observe their contents to be O.K. Now step through the next instruction. Reg A should be equal to the multiplier 1A H. It is! Now step through the instructions until JMP instruction, checking the register contents after each instruction. After stepping through the JMP instruction at the location 880C H, we come again to the first instruction of the loop. Step through it. Now Reg A should contain 19 H (1A H-1). But what is it containing? It is still 1A H though we have decremented the E register which is holding the multiplier value. Checking E register shows that the multiplier has indeed been decremented as Reg E now contains 19H. So the culprit must be the MOV A,E instruction. As it is logically the correct instruction, the error must be one of coding and we must have done a poor job while desk-checking the opcodes. Looking up the opcodes, we find that, indeed, instead of entering 7B H (opcode for MOV A,E), we entered 79H (opcode for MOV A,C) in location 8806 H. Thus the multiplier is never found to be zero and an infinite loop occurs. Correct this opcode and run the program again. Now it works.

This, rather trivial example was used only to illustrate the use of the step command in debugging the programs.

With experience, the user will certainly find many more ways in which the commands can be combined to develop useful programs.

14.3 USE OF MONITOR ROUTINES

The user can considerably simplify his program development if he makes suitable use of the several useful routines available in the system Monitor. Chapter 6 provides the descriptions and calling addresses of such useful routines available in the keyboard monitor and in the serial monitor.

The programming examples which follow, illustrate the use of monitor routines.

EXAMPLE 4: This example waits for the user to press a key and then displays the corresponding internal key code in the data filed. The program is an infinite loop and to recover from this program, you must press the RESET key.

Note: The Keys RESET and KBINT are not connected to the keyboard controller. So the program cannot recognize these two keys.

LOCATION	CONTENTS	LABEL	MNEMONIC	COMMENTS
8800	CD,14,05	LOOP:	CALL RDKBD	;Get the key code
8803	57		MOV D,A	;Reg D=Hex value to ;be expanded
8804	CD,9D,04		CALL HXDSP	;Expand the digits
8807	06,00		MVI B,00H	;No Dot
8809	3E,01		MVI A,01H	;use data field for ;display
880B	CD,E3,04	CALL OUTPUT		;Display the keycode
880E	C3,00,88	JMP LOOP		

Enter this program using EXAM MEM command and execute it using GO command. Whenever key is pressed (any key other than RESET and KBINT keys), its internal keycode is displayed and you can compare this value with the values shown in Table 5.2 (in chapter 5 on Hardware).

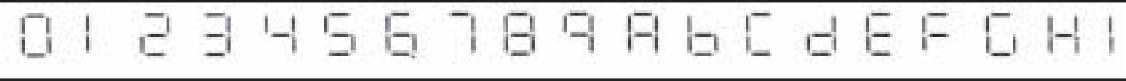
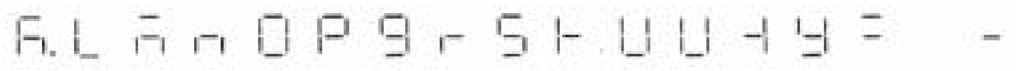
Example 5: If you objective is to display a numerical value only, as in the above example, a much simpler way would be to use the routines UPDAD, UPDDT. The above program can be written as follows, if we use the routine UPDDT.

LOCATION	CONTENTS	LABEL	MNEMONIC	COMMENTS
8800	CD 14 05	LOOP:	CALL RDKBD	;Get key code
8803	06 00		MVI B,00	;No Dot
8805	32 75 FE		STA FE75H	;Store key code in ;reserved location
8808	CD 7805	CALL UPDDT		;to Display it
880B	C3 00 88	JMP LOOP		

Example 6: This example flashes the two fields of display alternately. This program uses a subroutine to produce delay between the displays.

LOCATION	CONTENTS	LABEL	MNEMONIC	COMMENTS
8800	3E 00	AGAIN:	MVI A,00H	;Use address field
8802	06 00	MVI B,00		;No dot
8804	21 40 88	LXI H, 8840 H		;Use character string starting at location 8840 H
8807	CD E3 04		CALL OUTPUT	;and display Fire
880A	3E 01		MVI A,01H	;Use data field
880C	06 00		MVI B,00	;No dot
880E	21 44 88		LXI H,8844 H	;Character string starts at 8844 H.
8811	CD E3 04		CALL OUTPUT	;Display blanks in ;data field
8814	CD 31 88		CALL DELAY	;Introduce a delay
8817	3E 00		MVI A, 00H	;Use address field
8819	06 00		MVI B, OOH	;No dot
881B	21 46 88		LXI H, 8846 H	;Display ‘HELP’ in
881E	CD E3 04		CALL OUTPUT	;address field
8821	3E 01		MVI A, 01	Use data field
8823	06 00		MVI B, 00	;No dot
8825	21 4A 88		LXI H, 884AH	;Message start
8828	CD E3 04		CALL OUTPUT	;Display “US” ?
882B	CD 31 88		CALL DELAY	;Introduce a delay
882E	C3 00 88		JMP AGAIN	;Repeat the sequence
8831	11 FF FF	DELAY:	LXI D, FFFF H	;Delay subroutine
8834	1B	DLOOP:	DCX D	
8835	7A		MOV A,D	
8836	B3		ORA E	
8837	C2 34 88		JNZ DLOOP	
883A	C9		RET	
			ORG 8840 H	
8840	OF 13 14 0E 16 16 10 0E 11 12 15 05		DB 0F 0E 16 16 10 13 14 0E 11 12 15 05	

Enter the program from 8800 H to 883A H and enter the data from 8840 H to 884B H, using the EXAM MEM command. Using the GO command execute the program. You should see alternate displays of the messages “FIrE”, “HELP US”. The delay between the messages can be changed by altering the value initially loaded into D,E register pair, in the DELAY subroutine. We can change the displays message, by entering the different codes. The codes for the each display is given below.

Code	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	B2	10	13
DATA	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I
DISP,																			
CODE	71	11	FA	BD	00	12	AF	14	05	2E	15	15	73	AD	1D	16	34		
DATA	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z		-		
DISP,																			

Example 7 : This example illustrates the use of KBINT key. As explained in chapter 5 on Hardware, KBINT key can be connected to the RST7.5 pin of 8085 processor by installing the jumper JP1. In such a case, when this key is pressed, control is transferred to a prespecified location in memory (3CH) as RST7.5 is a vectored interrupt. The monitor has a JMP FE 12 instruction starting at this location. Thus when KBINT key is pressed, control first goes to the location 3CH and by executing the instruction here, comes to location FE12H. In this region also, there are not many bytes available for a service routine, so we will write a further JMP at FE12 H, to user RAM area and in that area we must write the service routine.

The program described below is a random number generator. The main program consists of a simple loop to increment a one-byte counter. The count after reaching FFH, become 0 again with one further increment and thus is always a value between 0 and FFH. Whenever the user presses the KBINT key, the interrupt service routine fetches the counter value, masks off the most significant 6 bits, displays the value (thus the number displayed is always, 00,01,02 or 03) and returns to the main program which continues with updating the counter value.

LOCATION	CONTENTS	LABEL	MNEMONIC	COMMENTS
8800	3E 0B		MVI A,0BH	;unmask RST 7.5
8802	30		SIM	;Enable interrupts
8803	FB		EI	;Initialize interrupts

LOCATION	CONTENTS	LABEL	MNEMONIC	COMMENTS
8804	AF	XRA A		
8805	3C	LOOP:	INR A	;Infinite loop of
8806	C3 05 88		JMP LOOP	;updating the counter

Notice that, we must first unmask RST7.5 interrupt using the SIM instruction and then we must enable the Interrupt system using the EI instruction. Enter the above program using EXAM MEM command.

Now, we must write the service routine. Assume we want to load it from 8820 H. So we must write a JMP 8820 H instruction at location FE 12 H.

LOCATION	CONTENTS	MNEMONIC	COMMENTS
FE12	C3 20 88	JMP R75SR	;Jump to RST7.5 service routine

Enter these bytes using EXAM MEM command.

Now at 8820 H, we must write the service routine.

LOCATION	CONTENTS	LABEL	MNEMONIC	COMMENTS
8820	F5		PUSH PSW	;save counter
8821	E6 03		ANI03	;Mask off the most ;significant 6 bits
8823	32 75 FE	LOOP;	STAFE75H	
8826	06 00		MVI B,00H	;No Dot
8828	CD 78 05		CALL UPDDT	;Display the number
882B	F1		POP PSW	;restore counter
882C	FB		EI	;Enable interrupts
882D	C9		RET	;Return to main program.

Notice the EI instruction, before the RET instruction. This is required, as interrupts are disabled once an interrupt is recognized. As, we want to recognize the interrupts again, we must enable them before returning to the main program. Load the above service routine.

Execute the program at location 8800 H. Now whenever the KBINT key is pressed, a value between 0 and 3 is displayed in the data field.

As this program is an infinite loop, to recover, you must press the RESET key.

Example 8: This example illustrates the use of serial monitor routine to display a message on the console. In the following program it displays ‘ELECTRO SYSTEMS’ on the console.

LOCATION	CONTENTS	LABEL	MNEMONIC	COMMENTS
8800	21 00 89		LXI H, 8900H	;Start of the first message
8803	CD 04 0B		CALL DISPM	;Displays the first word ‘ELECTRO’
8806	06 02		MVI B,02	;No of blanks after the first word
8808	CD 41 0C		CALL NSPOUT	;Displays two blanks after this word
880B	21 08 89		LXI H, 8908	;Start of the second message
880E	CD 04 0B		CALL DISPM	;Displays the second word ‘SYSTEMS’
8811	DF		RST E	;Return to monitor
8900	45 4C 45 43 54 52 4F 00			
8908	53 59 53 54 45 4D 53 00			

Enter the program in locations 8800H to 8811H and data from 8900 to 890F using ‘S’ command. And using ‘G’ command execute the program. Then ‘ELECTRO SYSTEMS’ will be seen on the console.

NOTE: 1. This program as well as the earlier programs can be entered using the Text Editor and assembled using the resident Assembler. Disassemble the code using the Disassembler before executing the programs to see the machine codes generated.

14.4 RELOCATION OF PROGRAMS

What is Relocation?

Suppose you have a program assembled from a given location. If you move the program code to a different location, generally it will not work properly. This is because of the fact that all jump, call and direct load instructions will be having address references pointing to the old locations. As an example consider the following DELAY subroutine.

LOCATION	INSTRUCTION	LABEL	MNEMONIC
8840	01 FF 07		LXI B, 07FFH
8843	0B	DLY:	DCX B
8844	78		MOV A,B
8845	B1		ORA C
8846	C2 43 88		JNZ DLY
8849	C9		RET

If you move the above block of code to another location, say 8850H, then it will not work correctly. The instruction at 8846H will be JNZ 8843 while it should be JNZ 8853 as the value of the label DLY is now 8853H. All such address references must be adjusted appropriately, for the moved program to work properly. Such a process of adjustment is called Relocation. (Of course this is a simplistic view, but adequate for the present discussion).

Such a relocation feature is an extremely convenient tool for debugging. As an example suppose you want to debug a program located in a PROM. As the program is in a PROM, you can’t conveniently introduce any code patches to try out alternatives. If you can relocate such a program into RAM area, it will be a simple matter to introduce the necessary code patches. This is just one example. There are many other situations where a Relocation facility can be put to good use.

ESA85-2 provides the user with a convenient tool for such a relocation. This is done through a monitor routine RELCT whose calling address id 0630H. Now we describe the use and limitations of this routine.

Using the Relocation routine RELECT

To use this routine, the following parameters must be set to appropriate values:

MEMORY	LOCATION	PARAMETER DESCRIPTION
FE6D,	FE6E	Relocation offset (OFSTAD) Relocation offset = New starting Address - Present starting address.

MEMORY	LOCATION	PARAMETER DESCRIPTION
FE6F,	FE70	Starting address of the program (LOLMT)
FE71	FE72	Ending address of the program (HILMT) (All the instructions between the starting address and ending address are relocated)
FE67	FE68	Low-Limit of the range (SADD1)
FE6B	FE6C	High-Limit of the range (DADD1) (All memory references to the addresses which are within this range are adjusted)

Operation of RELCT routine.

This monitor routine will examine each instruction, from starting address to ending address. If the instruction is not a memory reference instruction, it will proceed to examine the next instruction. On the otherhand, if it is a memory reference instruction, then it will check if the referred address is within the range Low-Limit, High-Limit. If it is out of range the instruction is not disturbed. Otherwise, the Relocation offset is added to the reference address of this instruction. For example, assume the instruction is 78H (MOV A,B). It will be left undisturbed. Suppose the instruction is C2,43,88 (JNZ 8843H) i.e. a memory reference instruction. Suppose the Low Limit is 8840H and High-Limit is 8849H. Then 8843 is within this range. Now assume that the offset is 0010H. Then it will and 0010H to 8843H and this new value of 8853H will replace 8843H. Thus the instruction will now appear as C2,53,88. This process is repeated with all the instructions from Start address to End address.

Note:

All three byte instructions are of memory reference type, but for the following 4 instructions; LXI B, Value; LXI D, Value; LXI H, Value; LXI SP, Value.

Steps involved in using the RELCT routine

In summary the steps involved in using the RELCT routine are as follows:

Step 1 : Use the Block Move command to move the program code from the original area to the desired area.

Step 2 : Set up the parameters required for the RELCT routine as explained above.

Step 3 : Find 4 bytes of free RAM area and enter the following program.

```
CALL        RELCT       ; Call the monitor routine to
                          relocate the code

RST         3           ; Return to monitor
```

Step 4 : Execute the above program using the GO command.

Step 5 : If required, alter addresses which are not memory references, if any.

Example 9: Consider the following simple subroutine which implements a delay. (Assume the program is assembled from 8840 H).

LOCATION	INSTRUCTION	LABEL	MNEMONIC
8840	01 FF 07		LXI B,07FF H
8843	0B	DLY:	DCX B
8844	78		MOV A,B
8845	B1		ORA C
8846	C2 43 88		JNZ DLY
8849	C9		RET

Let us assume that we want to relocate this program to the address 8850H. As already noted the first step is to move the code using the BLOCK MOVE command. The required key sequence is:

BLKMOVE <8840>NEXT<8849> NEXT<8850> EXEC

Note that the locations 8856, 8857 and 8858 contain the values C2,43,88. Thus the target address of this "JNZ" instruction is incorrect. It should be 8853 and not 8843. Hence this value must be adjusted suitably. In a larger program, there might be several such memory references which have to be adjusted. This can be done conveniently by using the routine RELCT.

We have to now set up the parameters for the RELCT routine as shown below.

Relocation offset:

As we want to move the program from 8840H to 8850H, the offset is 8850 - 8840 =0010H.

Starting address = 8850H (and not 8840H) Ending address = 8859H (and not 8849H) Low-Limit and High Limit:

As the original program contains reference to addresses in the range 8840-8849H, we have to specify this range for our relocation, so:

Low-Limit = 8840 H
High-Limit = 8849H

Enter these parameters into the appropriate memory locations using the EXAM MEM Command.

Now we must 'CALL' the RELCT routine. For this purpose, we have to write small program. Find 4 bytes of unused RAM area (say 8C00-8C03H) and enter the following program and execute it using the GO command.

LOCATION	CONTENTS	COMMENTS
8C00	CD	; Call the Relocate Routine
8C01	30	
8C02	06	
8C03	DF	; Return to monitor

Now you can use EXAM MEM command to examine the locations 8850-8859H and verify that indeed the program has been relocated. (8857H location will contain 53H and not the incorrect 43H).

Exercise: If you have installed a 62256 in the expansion PROM/RAM (4000H-7FFFH) try to relocate the Monitor (0000H to 3FFFH) to start from 4000H.

Note that RELCT routine will adjust only memory reference. This may necessitate further adjustments as shown in the following example.

Example 10: Consider the following program which outputs 4 characters stored in a table to the key keyboard/display controller 8279.

LOCATION	CONTENTS	LABEL	MNEMONIC
8800	21 0E 88		LXI H, 880E
8803	0E 04		MVI C,04
8805	7E	D10:	MOV A,M
8806	D3 30		OUT DATA79
8808	23		INX H
8809	0D		DCR C
880A	C2 05 88		JNZ D10
880D	DF		RST 3
880E	37 37 60 E3		

Assume you want to relocate the program to start from 8820H and you want to move the program as well as the table of 4 bytes (which starts at 880E H).

Now if you use BLKMOVE command, then write the program fragment to relocate as in Example 9, you will have all memory references properly adjusted. But the first instruction will still be 21 0E 88. In otherwords, though the table has been moved from 880EH to 882EH, the value loaded into H,L register paid will remain unaltered. This is because of the fact that, as already noted, RELCT routine adjusts only memory reference addresses and the instruction LXI H, value does not involve memory reference. Hence you have to adjust such values after relocation.

Summarizing this example, if constant data is being moved, reference to such data items must be adjusted by the user.