www.vinyticsppl.com

(VMC-8501/8502/8503/8503A)
MICROPROCESSOR TRAINING CUM-DEVELOPMENT KIT BASED ON 8085

Vinytics PERIPHERALS PVT. LTD.

C-241, G8 PANDAV NAGAR, DELHI-110092
Tel. : 011-22486937
Fax : 22515127
Email : vinyticsindia@gmail.com,
vinyticsppl@rediffmail.com
Website : www.vinyticsppl.com

TABLE OF CONTENTS

Chapter-1 ..... SYSTEM INTRODUCTION

GENERAL DESCRIPTION 1
SYSTEM SPECIFICATIONS 2
SYSTEM CAPABILITIES 3

Chapter-2 ..... HARDWARE DESCRIPTION

GENERAL 4
MEMORY 4
I/O DEVICES 4
8279, 8255, 8253, 8155 DISPLAY

Chapter-3 ..... COMMAND DESCRIPTION

KEY BOARD DESCRIPTION 7
LIST OF COMMANDS 9
COMMAND DESCRIPTION 10

Chapter-4 ..... ON BOARD INTERFACES (OPTIONAL)

ONBOARD RS-232-C INTERFACE 30
CRT TERMINAL INTERFACE 30
HYPER TERMINAL (WINDOWS UPLOAD & DOWNLOAD) 31

Chapter-5 ..... SERIAL I/O DEVICE COMMANDS

GENERAL 33
LIST OF COMMANDS 33
COMMAND DESCRIPTION 34
LIST A MEMORY BLOCK 34
EXAMINE/MODIFY MEMORY 34
ENTER A MEMORY BLOCK 35
EXAMINE/MODIFY REGISTER 36
SINGLE INSTRUCTIONS 37
GO COMMAND (G) 38
BLOCK MOVE COMMAND (B) 38
INSERT COMMAND (I) 38
DELETE COMMAND (D) 39
INSERT DATA (N) 39
DELETE DATA (O) 40
FILL (F) 40
RELOCATE (H) 41
MEMORY COMPARE (J) 41
STRING (K) 42

Chapter-7 ..... SYSTEM EXPANDABILITY
ON BOARD EXPANSION 72
DETAILS OF CONNECTIONS 73
SIGNALS AT CONNECTOR C1 (BUS) 73
SIGNALS AT CONNECTOR C2 (TIMER) 74
SIGNALS AT POWER SUPPLY CONNECTOR 74
SIGNALS AT CONNECTOR C6 (8155) 74
SIGNALS AT CONNECTOR C4 (8255) 75
SIGNALS AT CONNECTOR C5 (RS232C) 75

APPENDIX-A

INTERRUPTS IN 8085 KIT 77

APPENDIX-B

INSTRUCTION SET OF 8085 80

APPENDIX-C

ASSEMBLER & DISASSEMBLER OF 8085 84

APPENDIX-D

SYNTAX OF 8085 ASSEMBLER & DISASSEMBLER 88
MAINTENANCE TIPS FOR VMC-850X 90
REFERENCES 90

BLOCK DIAGRAM FOR VMC-8501 91
BLOCK DIAGRAM FOR VMC-8502 92

***

Chapter-1

SYSTEM INTRODUCTION

GENERAL DESCRIPTION

VMC-850X series Kit is a single board **MICROPROCESSOR TRAINING/ DEVELOPMENT KIT** configured around the most widely used Microprocessor of todays world. Based on 8085 Microprocessor, it can be used to train engineers to control any industrial process and to develop software for 8080 and 8085 based systems.

The VMC-850X communicates with the outside world through a key board having 28 keys and seven segment hexadecimal display. The kit also has the capability of interacting with CRT Terminal and IBM PC compatible computer system through the serial interface provided on the board.

VMC-850X provides 8K/32K bytes of RAM and 8K bytes of EPROM. The total on board memory can be very easily expanded to 64K bytes in an appropriate combination of RAM and ROM. The monitor is incorporated from 0000-1FFF and the necessary 8K bytes of RAM has an address of 2000-3FFF.

The Input/Output structure of VMC-8501 provides 24 programmable I/O lines using 8255. It has got 16 bit programmable Timer/Counter for generating any type of counting etc. The Input/ Output structure of VMC-8502 is same as VMC-8501 and it has additional feature of 22 I/O lines and a 14 bit Timer/Counter facility using 8155.

The on board residents system monitor software is very powerful and provides various software utilities. The kit provides various powerful software commands like INSERT, DELETE, BLOCK MOVE, RELOCATE, STRING, FILL & MEMORY COMPARE etc. which are very helpful in debugging/developing the software.

VMC-850X is configured around the internationally adopted STD Bus, which is the most popular bus for process control and real time applications. All the address, data and control lines are available at the edge connector. The Kit is fully expandable for any kind of application.

SYSTEM SPECIFICATION FOR VMC-8501

CPU	-	8 bit Microprocessor, the 8085
MEMORY	-	Total on board capacity of 64K bytes
RAM	-	8K/32K bytes and space for further expansion
ROM	-	8K bytes of EPROM loaded with powerful program
TIMER	-	16 bit programmable timer/counter using 8253
I/O	-	24 I/O lines using 8255
KEYBOARD	-	10 keys for command 16 keys for hexadecimal data entry 1 key for vector interrupt & 1 key for reset
LED DISPLAY	-	6 seven segment display 4 for address field & 2 for data field
BUS	-	All data, address and control signals (TTL compatible available at FRC connector
INTERFACE	-	RS-232-C through SID/SOD lines (optional)
POWER SUPPLY REQUIREMENT	-	+5V, 1.5Amp for the kit ±12V +5%, 250mA for CRT/PC interface
OPERATING TEMPERATURE	-	0 to 50°C

ADDITIONAL SPECIFICATION FOR VMC-8502

I/O	-	24 I/O lines using 8255 & additional 22 I/O lines & a 14 bit Timer/Counter using 8155

ADDITIONAL SPECIFICATION FOR VMC-8503

I/O	-	48 I/O lines using 02 nos. of 8255

ADDITIONAL SPECIFICATION FOR VMC-8503A

SERIAL COMM.	-	Through RS-232C port using 8251.
REAL TIME CLOCK	-	Using IC 6242

SYSTEM CAPABILITIES (KEYBOARD MODE)

1. Examine the contents of any memory location.
2. Examine/Modify the contents of any of the uP internal register.
3. Modify the contents of any of the RAM location.
4. Move a block of data from one location to another location.
5. Insert one or more instructions in the user program.
6. Delete one or more instructions from the user program.
7. Relocate a program written for some memory area to some other memory area.
8. Find out a string of data lying at a particular address.
9. Fill a particular memory area with a constant.
10. Compare two block of memory.
11. Insert one or more data bytes in the user's program/data area.
12. Delete one or more data bytes from the user's program/data area.
13. Execute a program at full clock speed.
14. Execute a program in single step i.e. instruction by instruction.

SYSTEM CAPABILITIES (SERIAL MODE)

Most of the commands mentioned above can also be used in the serial mode. Please refer to the chapter-5.

Chapter-2

HARDWARE DESCRIPTION

GENERAL

The system has got 8085 as the Central Processing Unit. The clock frequency for the system is 3.07 MHz and is generated from a crystal of 6.14 MHz.

8085 has got 8 data lines and 16 address lines. The lower 8 address lines and 8 bit data lines are multiplexed. Since the lower 8 address bits appear on the bus during the first clock cycle of a machine cycle and the 8 bit data appears on the bus during the 2nd and 3rd clock cycle, it becomes necessary to latch the lower 8 address bits during the first clock cycle so that the 16 bit address remains available in subsequent cycles. This is achieved using a latch 74-LS-373.

MEMORY

VMC-850X provides 8/32K bytes of RAM using 6264/62256 chip and 8K bytes of EPROM for monitor. There is one memory space provided on VMC-850X. This one space can be defined any address slots from 8000 - DFFF depending upon the size of the memory chip to be used. Total onboard memory can be extended to 64K bytes.

I/O DEVICES

The various I/O chips used in VMC-8501 are 8279, 8255 & 8253 and VMC-8502 are 8279, 8255, 8253 & 8155. The functional role of all these chips is given below:

8279 (Keyboard & Display Controller)

8279 is a general purpose programmable keyboard and display I/O interface device designed for use with the 8085 microprocessor. It provides a scanned interface to 28 contact key matrix provided in VMC-850X and scanned interface for the six seven segment displays. 8279 has got 16 x 8 display RAM which can be loaded or interrogated by the CPU. When a key is pressed, its corresponding code is entered in the FIFO queue of 8279 and can now be read by the microprocessor. 8279 also refreshes the display RAM automatically.

8255 (Programmable Peripheral Interface)

8255 is a programmable peripheral interface (PPI) designed to use with 8085 Microprocessor. This basically acts as a general purpose I/O device to interface peripheral equipments to the system bus. It is not necessary to have an external logic to interface with peripheral devices since the functional configuration of 8255 is programmed by the system software. It has got three Input/Output ports of 8 lines each (PORT-A, PORT-B & PORT-C). Port C can be divided into two ports of 4 lines each named as Port C upper and Port C lower. Any Input/Output combination of Port A, Port B, Port C upper and lower can be defined using the appropriate software commands. The port addresses for these ports are given in Chapter-6. VMC-850X provides 24 Input/Output ports using 8255 chips.

8253 (Programmable Internal Timer)

This chip is a programmable interval Timer/Counter and can be used for the generation of accurate time delays under software control. Various other functions that can be implemented with this chip are programmable rate generator, Even Counter, Binary rate Multiplier, Real Time Clock etc. This chip has got three in dependent 16 bit counters each having a count rate of up to 2KHz. The first Timer/Counter (i.e. Counter 0) is being used for Single Step operation. However, its connection are also brought at connector space C4. For single step operation CLK0 signal of Counter 0 is getting a clock frequency of 1.535 MHz. The counter 1 is used to generate clock for 8251. Counter 1 & Counter 2 are free for the user. Clock for the CLK1, CLK2 is to be given externally.

8155 (Programmable I/O Port & Timer Interface) Optional (only in Model VMC-8502)

8155 is a programmable I/O ports and timer interface designed to use with 8085 Microprocessor. The 8155 includes 256 bytes of R/W memory, three I/O ports and a Timer. This basically acts as a general purpose I/O device to interface peripheral equipments to the system bus. It is not necessary to have an external logic to interface with peripheral devices since the functional configuration of 8155 is programmed by the system software. It has got two 8-bit parallel I/O port (Port-A, Port-B) and one 6-bit (Port-C). Ports A & B also can be programmed in the handshake mode, each port using three signals as

handshake signals from Port-C. The timer is a 14 bit down counter and has four modes. VMC-8502 optionally provides 22 I/O ports & a 14 bit timer/counter.

8251 (USART)

This chip is a programmable communication interface and is used as a peripheral device. This device accepts data characters from the CPU in parallel format and then converts them into serial data characters for the CPU. This chip will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character from the CPU. The CPU can read the complete status of it at any time. One such chip is used in VMC-8503A Kit and these can be used for interfacing any serial device. The connections of 8251 is brought at connector C5.

DISPLAY

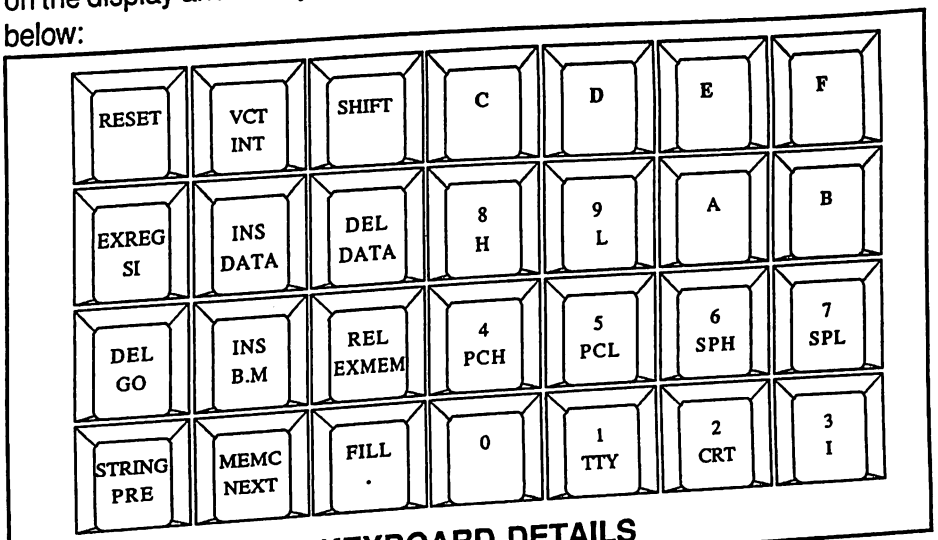
VMC-850X provides six digits of seven segment display. Four digits are for displaying the address of any location or name of any register, whereas the rest of the two digits are meant for displaying the contents of a memory location or of a register. All the six digits of the display are in hexadecimal notation.

Chapter-3

COMMAND DESCRIPTION

KEYBOARD DESCRIPTION

VMC-850X has 28 keys and six-seven segment display to communicate with the outside world. As VMC-850X is switches on; a message -'UP 85' is displayed on the display and all keys are in command mode. The key board is as shown below:



KEYBOARD DETAILS

RESET - Reset the system.

VCT INT - Hardware interrupt via keyboard, RST 7.5.

SHIFT - Provides a second level command to all keys.

GO - To execute the program.

SI - To execute the program in single step mode.

EXREG - Examine Register; allows user to examine and modify the contents of different registers.

EXMEM - Examine Memory; allows user to examine any memory location and modify any RAM location.

PRE - Previous is used as an intermediate terminator in case

	of Examine Memory. It decrements the PC contents and writes the contents of data field to the address displayed in the address location.
NEXT	- Increment is used as a intermediate terminator in case of Examine Memory, Examine Register etc. It increments the PC Contents and writes the data lying in data field at the location displayed at address field.
"."	- Terminator is used to terminate the command and write the data in data field at the location displayed in address field.
DEL	- Delete the part of program or data, with relocation by one or more bytes.
INS	- Inserts the part of the program or data with relocation, by one or more bytes.
B.M.	- Allows user to move a block of memory to any RAM area.
FILL	- Allows user to fill RAM area with a constant.
REL	- Relocates a program written for some memory area and to be transferred to other memory area.
INS DATA	- Inserts one or more data bytes in the user's program/ data area.
DEL DATA	- Deletes one or more data bytes from the user's program/ data area.
STRING	- Finds out the string of data lying at a particular address or addresses.
MEMC	- Memory Compare: Compares two blocks of memory for equality.
0 - F	- Hexadecimal Keys.

A '-' on the MSD of address display indicates that system is waiting for a command. If, instead of a valid command, the user gives a data, the system will display '-Err'. A dot on the LSD of address field indicates that the system expects an address. Whenever the data of any memory location is changed, a dot is displayed on the LSD of Data Field.

The VMC-850X accepts all data and address in hexadecimal form as given in the table - 1.

TABLE-1

HEXADECIMAL	DECIMAL	BINARY	LED DISPLAY
0	0	0000	0
1	1	0001	1
2	2	0010	2
3	3	0011	3
4	4	0100	4
5	5	0101	5
6	6	0110	6
7	7	0111	7
8	8	1000	8
9	9	1001	9
A	10	1010	A
B	11	1011	b
C	12	1100	C
D	13	1101	d
E	14	1110	E
F	15	1111	F

LIST OF COMMANDS

1. RESET
2. EXAMINE/MODIFY REGISTER
3. EXAMINE/MODIFY MEMORY

4. GO
5. SINGLE INSTRUCTION
6. BLOCK MOVE
7. DELETE
8. INSERT
9. RELOCATE
10. FILL
11. STRING
12. MEMORY COMPARE
13. INSERT DATA
14. DELETE DATA

COMMAND DESCRIPTION

RESET

This key initializes the VMC-850X Kit and displays '- UP85' on the display. A '-' on the left most end of display indicates that the system is expecting a valid command.

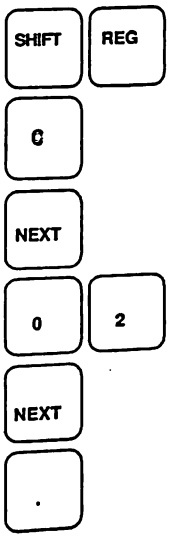
EXAMINE/MODIFY REGISTER (EXREG)

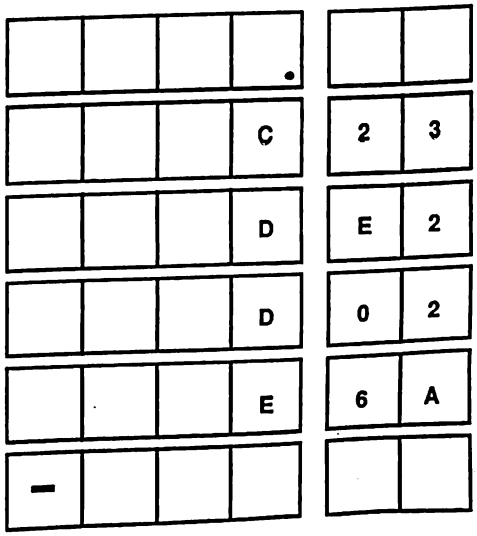
This command is used to examine/modify any internal register of the CPU. If one wants to examine the contents of all the registers, one can start from `A' Reg. and examine all the registers by pressing next key. Whereas if some specific registers is to be examined, then the key for that register can be entered directly. The contents of any register can be changed.

Exercise

Examine the contents of C Reg. and D Reg. and change the contents of D Reg. to 02.

HEX KEY PAD





On pressing SHIFT and REG key, a dot is displayed in the address field. Enter the reg. identifier i.e. C. The contents of C reg. is displayed. Press NEXT to see the content of D Reg. E2 is displayed. Press 0 Key and 2 key and then NEXT to enter 02 in D Register. Terminate the command by pressing "." key.

Note: The data 23 and E2 displayed for C and D Registers are just some arbitrary data and taken for example only.

EXAMINE/MODIFY MEMORY (EXMEM)

This command is used to examine the contents of any memory location and modify the contents of the RAM area.

On pressing this key, a dot is displayed in the end of address field. On can now enter the address of any location one wants to examine. Enter the desired address and press NEXT. The contents of this location is displayed in the data field. If one wants to examine the contents of next location, just press NEXT and the address in the address field will be incremented by one and its contents will

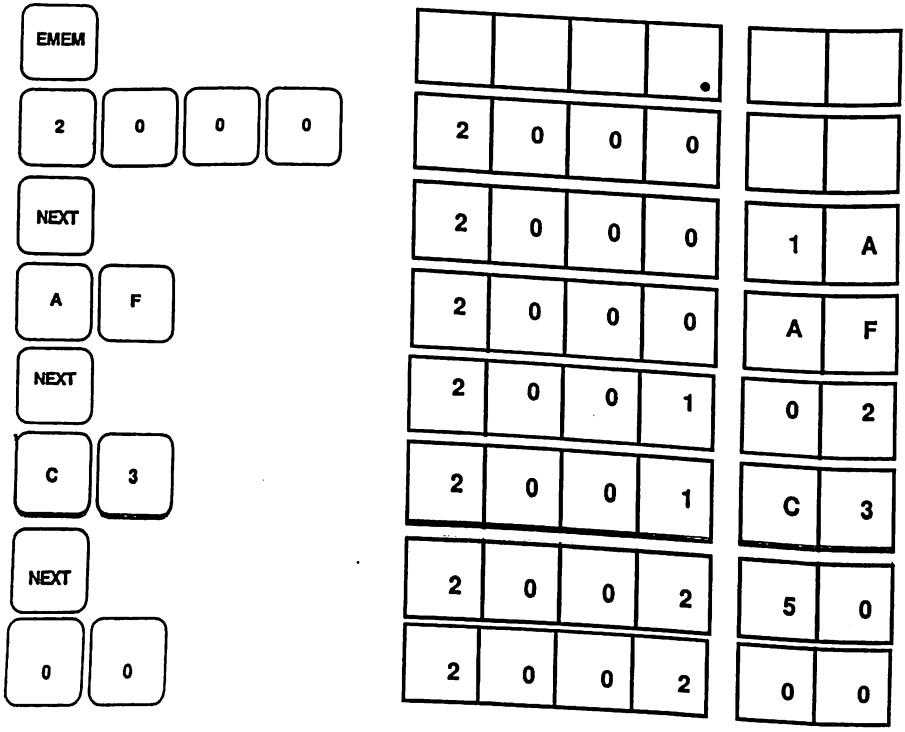
be displayed in the data field. Same way if one wants to examine the content of previous location just press PRE key and the address in the address field will be decremented by one and its contents will be displayed in the data field.

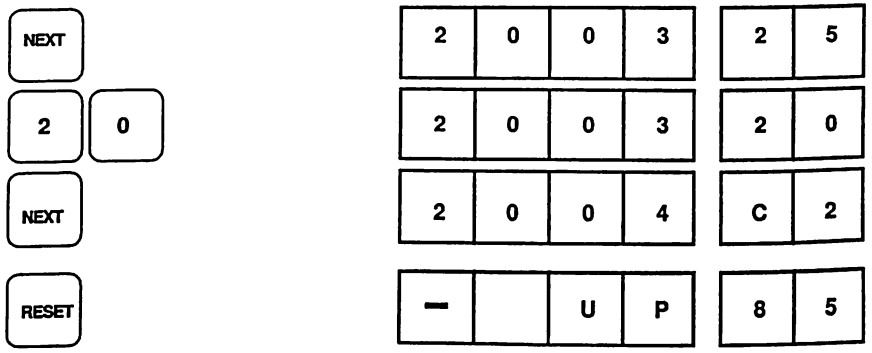
If one wants to modify the contents of any RAM location, then enter the data and press NEXT. The data field will be written in the address displayed in the address field and simultaneously the contents of next location will be displayed.

Exercise

Enter the following program:

Address	Data	Comments
2000	AF	XRA,A
2001	C3	JMP 2000
2002	00	
2003	20	





Verify that the program is entered properly.

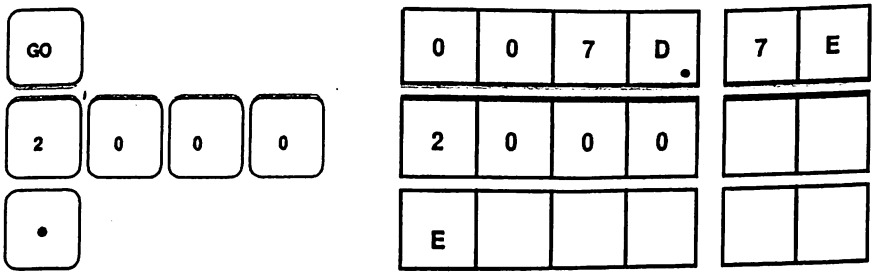
Note: Please note that data displayed on pressing NEXT i.e. 1A, 02, 50, 25 and C2 is some arbitrary data and need not be same, every time the process is repeated.

GO

This command is used to execute the program in full clock speed. On pressing this key, the program counter contents are displayed in the address field with the data in the data field. A dot in the address field indicates that the address can be changed, if so desired. Enter the starting address of the program. On entering this address, the data field gets blanked out. Press Terminate (.) key. The CPU will start executing the program and E will be displayed in the address field.

Exercise

Execute the program entered earlier using examine memory command.



SINGLE INSTRUCTIONS (SI)

This command is used to execute the program instruction by instruction. On pressing SI, the program counter content is displayed on the address field and its data in the data field. If one wants to modify the address, one can do that. After entering the address, press NEXT, the contents of the entered address is displayed. On pressing NEXT, one instruction will be executed and the address of the next instruction will be displayed with its data in the data field. Each time NEXT is pressed, one instruction is executed. If one wants to terminate and command at any stage, one can do that using (.) key. On pressing (.) key, a '- ' is displayed in the address field. One can now examine any internal register of CPU or any memory location and modify it if desired.

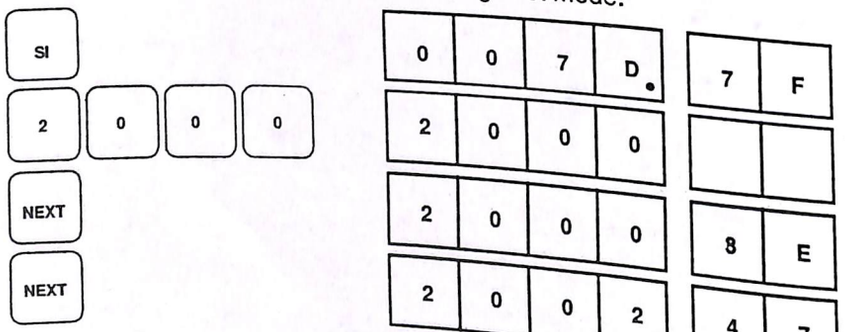
Exercise

Enter the following program and run in SI Mode.

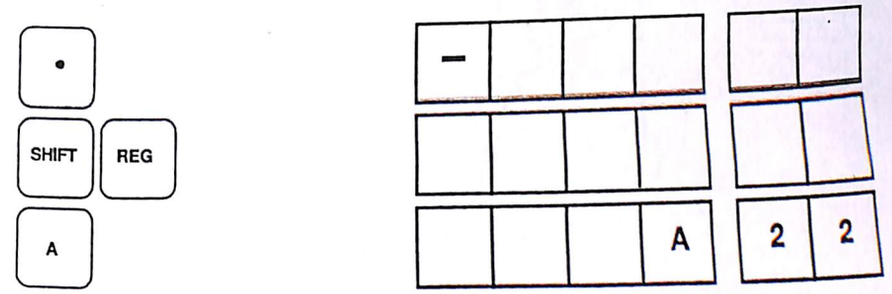
Address	Data	Mnemonic	Comments
2000	3E 22	MVI A,22	Move 22 in a Register
2002	47	MOV A,B	Move the Contents of B
Register into A Register			
2003	EF	RST 5	Software Breakpoint

Note: Enter the above program using examine memory command.

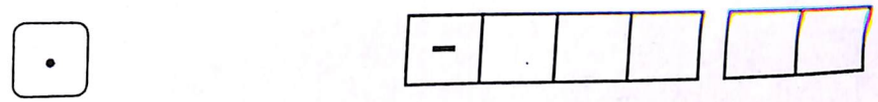
The following is the procedure of executing in SI mode.



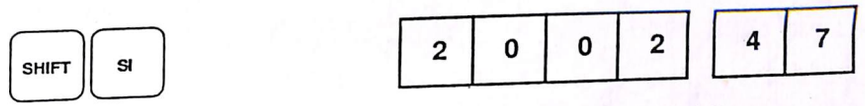
The first command is executed i.e. A Register has been loaded with 22. Let us examine the content of A Register. For this we will have to terminate the command here.



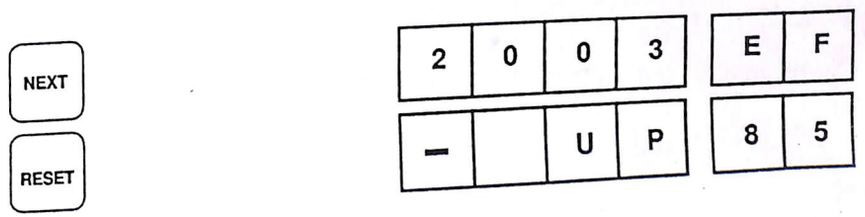
Terminate the command again.



To run the program further press SHIFT and SI.



The program came back at the same address where we left it.



The program has gone back to monitor.

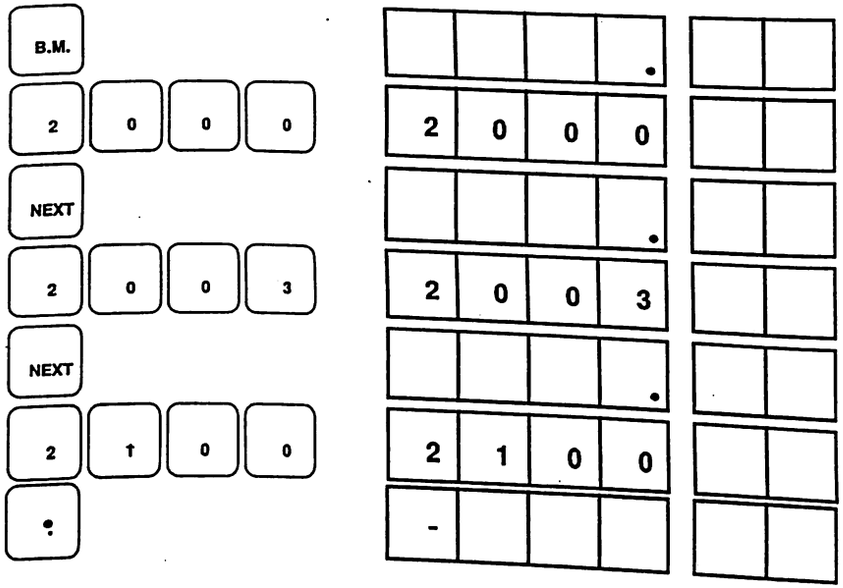
BLOCK MOVE (B.M.)

This command allows the user to move the block of data from one memory location to another memory location. On pressing this key a dot is displayed at the end of address field. Enter the starting address of the block to be moved and press NEXT. Again a dot is displayed. Now enter the end address of the

block and press NEXT. Again a dot is displayed. Now enter the destination address and press Terminate (.) key. A '-' is displayed in the display.

Exercise

Block Move the program lying from 2000 to 2003 (in the earlier exercise) to 2100.



Verify that the program has moved to 2100 using examine memory command.

DELETE (DEL)

This command allows the user to delete one or more instructions from the user's program. In this command all the memory referenced instructions also get modified accordingly to keep the logic of the program same. The following information is to be entered:

1) Starting address of the user program.
2) End address of the user program.
3) Address of the location from where onwards the bytes are to be deleted.
4) Address of the location till where the bytes are to be deleted.

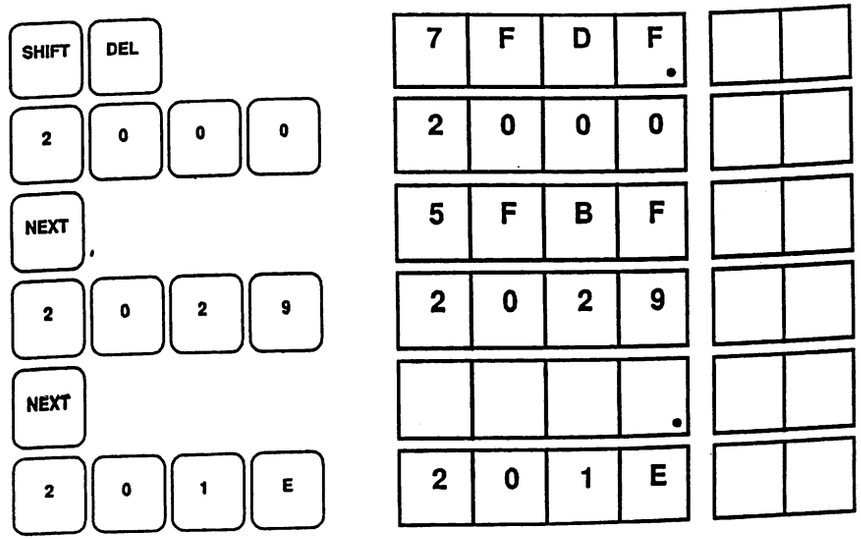
Exercise

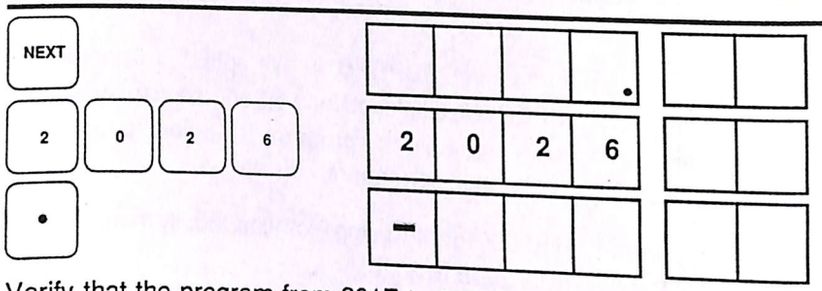
Take the program for flashing **'SUPERB'** in example 9 of Chapter-6. In this program the word **'SUPERB'** is displayed for 0.5 Sec., the display is cleared for 0.5 Sec. and the logic is repeated. In this program, if the clear routine is deleted, the word **'SUPERB'** will remain permanently displayed.

Enter this program from 2000 to 2029 using Examine/Modify memory command and delete the data from 201E to 2026.

Run this program using **GO** command before deleting the data. You will see that **'SUPERB'** is being flashed on the display. To delete the clear routine do the following:

On pressing SHIFT and DELete Key, some address is displayed. Enter the starting address of the program and press NEXT. Now enter the end address and press NEXT. A dot is displayed at the end of the Address field. Now enter the starting address from where the bytes are to be deleted and press NEXT. Again a dot is displayed at the end of address field. Enter the end address till where the bytes are to be deleted and press Terminator (.) key. A '-' will be displayed in the address field indicating that the system is ready to accept the new command.





Verify that the program from 201E to 2026 has been deleted. Execute the program and see that the word 'SUPERB' is displayed permanently.

INSERT (INS)

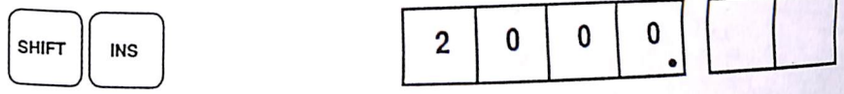
This command allows the user to insert one or more instructions in the user's program with automatic modification of the memory referenced instructions. This following information is required to be entered.

1) Starting address of the program.
2) End address of the program.
3) Address from where the bytes are to be entered.
4) No. of bytes to be entered.
5) Data.

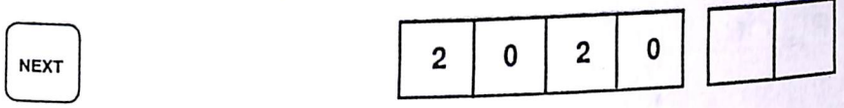
Exercise

Insert the bytes again, which are deleted in the above exercise of flashing **'SUPERB'**.

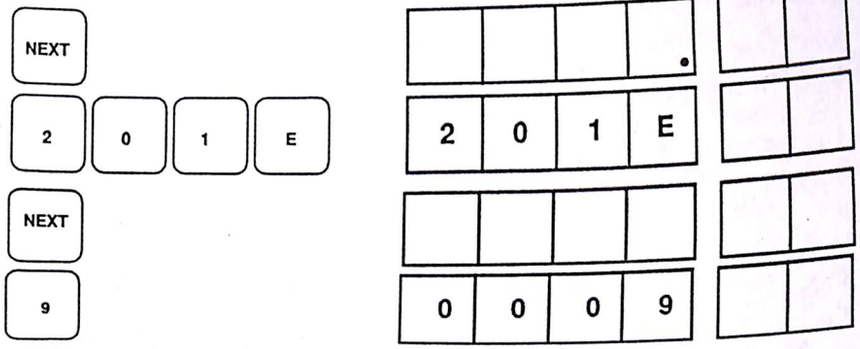
On pressing SHIFT and INSert, an address is displayed on the address field. Enter the starting address of the program and press NEXT. Again an address is displayed. Now enter the end address of the program and press NEXT. A dot is displayed at the end of the address field. Enter the address at which the bytes are to be entered and press NEXT. A dot is displayed again. Now enter the number of bytes to be entered and press NEXT. The system will display the address where you wish to enter the bytes with its current data in the data field. Enter the bytes you want to insert using NEXT key. When all the bytes are entered, a '-' will be displayed indicating that the system is ready to accept a new command.



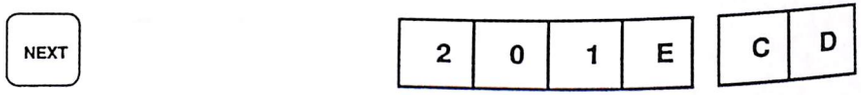
Since we have just executed this program, the program address are not disturbed. So we directly press NEXT.



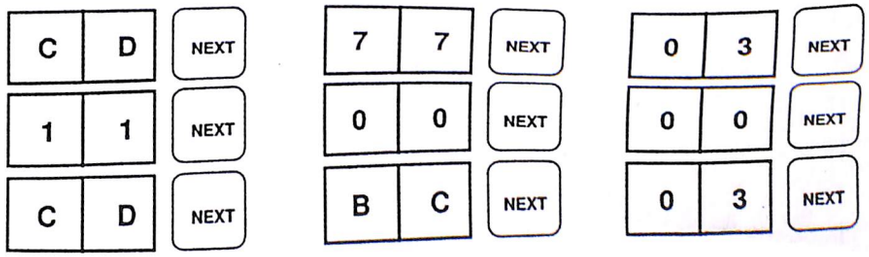
Our last address is also 2020, so we just press NEXT.



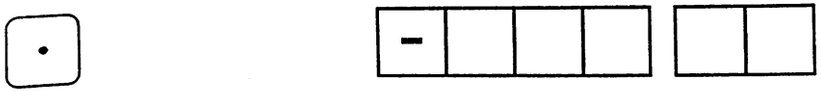
Since 9 bytes have to be entered,



the contents of 20 1E is displayed.



a '-' will be displayed.



Verify that the bytes have been inserted and execute the program from 2000 to note the flashing 'SUPERB' on the display.

RELOCATE(REL)

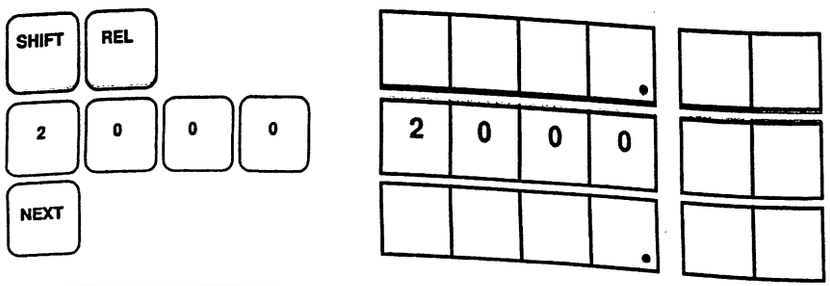
This command allows the user to relocate program written for some memory area, to some other memory area. The information required to be entered are:

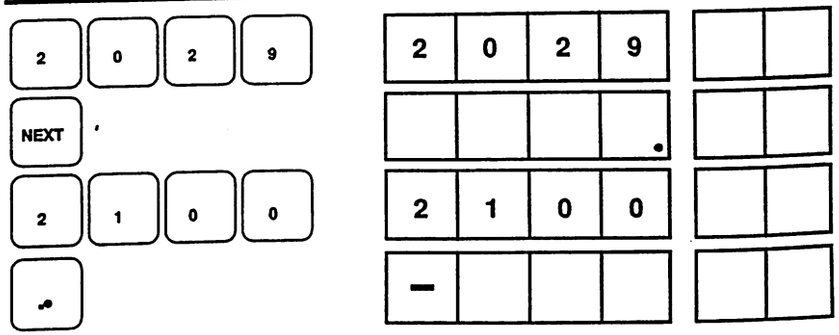
1. Starting address of the program.
2. End address of the program.
3. Destination address where the program has to be relocated.

The relocate command can be best understood with the earlier example of flashing **'SUPERB'**. This program is written for 2000 memory area. So that jump instruction for looping at the end is with reference to 2000 only. Suppose you want this program to be executable for 2100 area, then this can not be done by Block Move because when we Block Move the program from 2000-2029 to 2100, the contents of 2127 onwards will be 2127 - C3, 2128 - 06, 2129 - 20. Where as in order to execute this program from 2100, the content of 2129 should be 21.

This can be done using Relocate command.

On pressing SHIFT and RELocate, a dot is displayed in the address field. Enter the starting address of the program i.e. 2000 and press NEXT. A dot is displayed again. Now enter the end address of the program i.e. 2029 and press NEXT. Again a dot is displayed. Now enter the destination address i.e. 2100 and press (.).





Now verify that 2129 has got 21. Execute the program from 2100 and note that the 'SUPERB will be flashing on the display.

FILL

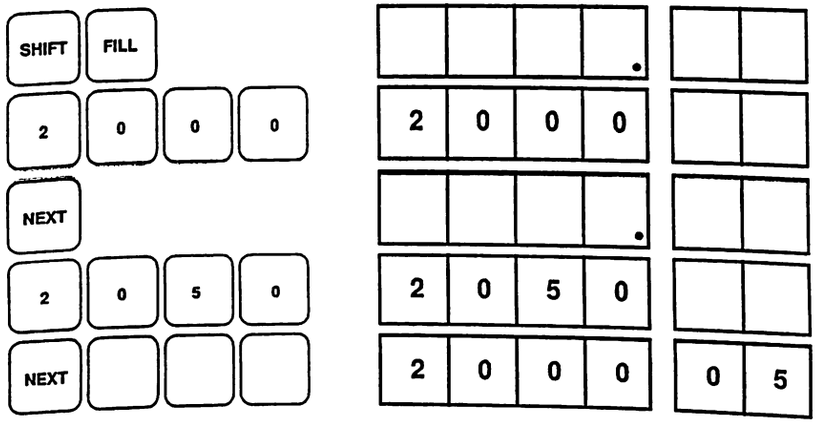
This command allows the user to fill a memory area (RAM) with a constant. The following information is required to be entered.

1) Starting address of the memory area from where the data should be stored.
2) End address of the memory area till were the data should be stored.
3) The constant with which the data should be done i.e. 22.

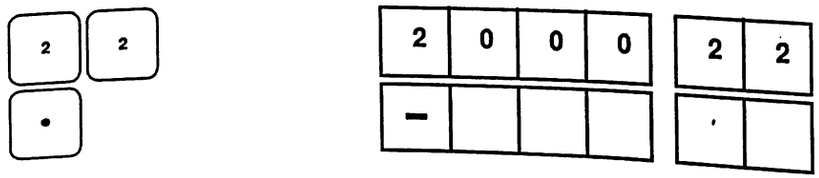
Exercise

Fill the RAM area from 2000 to 2050 with 22.

Press SHIFT and FILL. A dot will be displayed at the end of the address field. Enter the starting address and press NEXT. Again a dot will be displayed. Now enter the end address and press NEXT. The present contents of end address will be displayed. Enter the content and press (.).



05 is just some random data which was lying at 2000. Press '2' key twice and press Terminate (.) key.



Verify that 22 is filled from 2000 to 2050.

STRING

This command allows the user to find the address or addresses at which a particular string of Data is lying within a specified program. The word string here means a few bytes of Data lying consecutively one after another.

The following information needs to be entered.

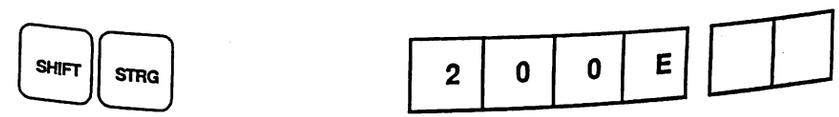
1) Starting address of the program.
2) End address of the program.
3) Address of the location at which the first byte of the string lies.
4) Address of the location at which the last byte of the string lies.

Suppose in the earlier example of flashing 'SUPERB', you want to find out the

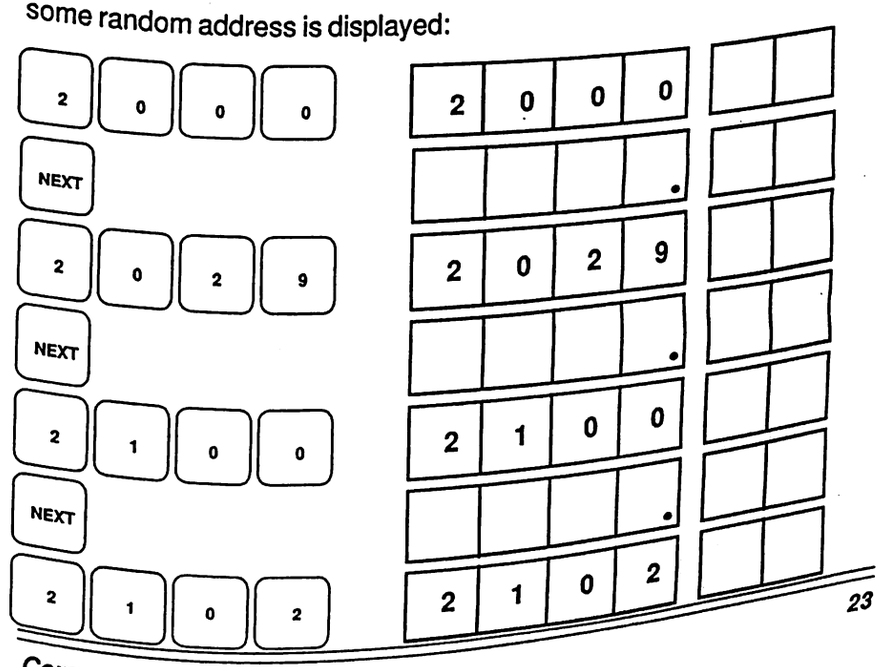
addresses at which CALL OUTPUT instruction is lying.

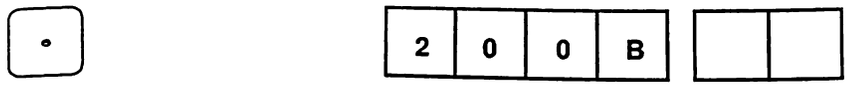
Press shift and string key. Some address is displayed. Enter the starting address of your program and press NEXT. A dot is displayed at the end of the address field. Now enter the end address of the program and press NEXT. A dot is displayed again. Enter the address where the first byte of the string lies and press NEXT. Again a dot is displayed. Now enter the addresses at which the last byte of the string has been stored and press Terminate (.) key. The system will display the first address (within the specified program) at which the string lies. Press NEXT to see the next address. This way using NEXT key you can see all the address at which the string of Data is lying. A '-' is displayed if all the addresses (at which the string or Data is lying) have been displayed.

Note: If you don/t know the first and last addresses where the string lies, then the string can be stored in another area and these addresses can be given. In this example the string is first stored at 2100.

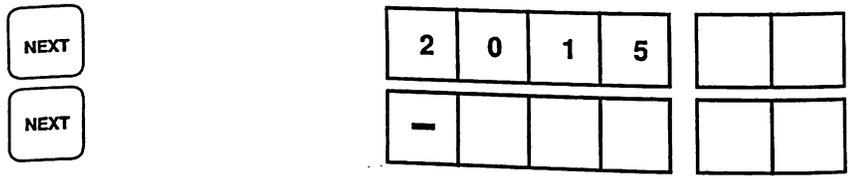


some random address is displayed:





first string address is displayed.



In the above example note the two address at which **'Call Output'** is lying. These address will be 200B and 2015. Verify by Examine memory command that CD D0 05 is lying at 200B and 2015 onwards.

MEMORY COMPARE (MEMC)

This command allows the user to compare two blocks of memory for equality. If they are not equal, the address of the first block at which there is a discrepancy will be displayed. The flowing information needs to be entered.

1) Starting address of the first Block.
2) End address of the first Block.
3) Starting address of the second Block.

Exercise

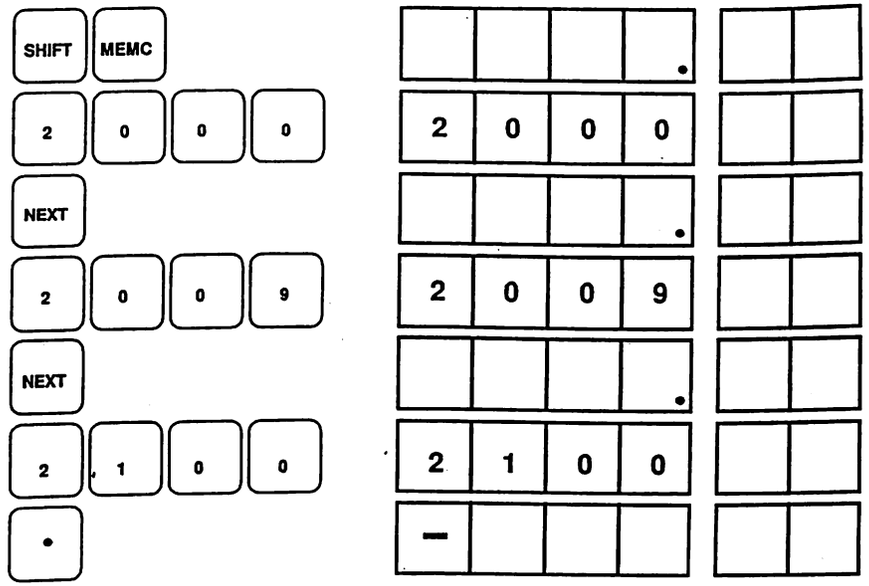
Enter the following Data using Examine Memory command:

2000	-	00	2005	-	55
2001	-	11	2006	-	66
2002	-	22	2007	-	77
2003	-	33	2008	-	88
2004	-	44	2009	-	99

Now Block Move this block to 2100 using B.M command. Now use Memory Compare command as follows:

Press SHIFT and Memory Compare. A dot is displayed at the end of address

field. Enter the starting address of the first block and press NEXT. Again a dot is displayed. Enter the end address of the first block and press NEXT. A dot is displayed. Now enter the starting address of the second block and press Terminate (.) key.



Since the two blocks are identical a '-' will be displayed.

Now change the content of 2005 to 50 and that of 2008 to 68. Again use the Memory Compare command as mentioned above. You will se that an address 2005 will be displayed on the address field and its contents in the data field. Press NEXT and now 2008 will be displayed with the contents. On pressing NEXT, '-' will be displayed indicating that the two blocks are identical.

INSERT DATA (INSD)

This command is exactly same as INSert except that in this command the relocation is not done after inserting the bytes.

This command allows the user to insert one or more instructions in the user's

program without automatic modification of the memory referenced instructions. This following information is required to be entered.

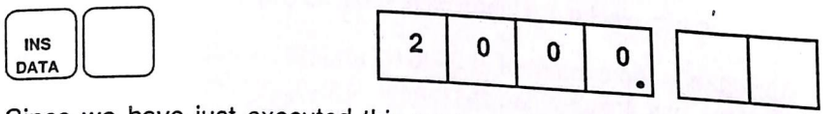
1) Starting address of the program.
2) End address of the program.
3) Address from where the bytes are to be entered.
4) No. of bytes to be entered.
5) Data.

Exercise

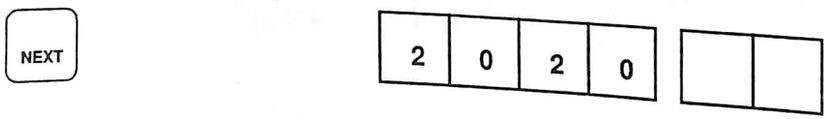
Insert the bytes again, which are deleted in the above exercise of flashing

'SUPERB'.

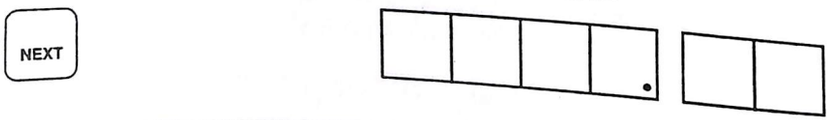
On pressing INSert, an address is displayed on the address field. Enter the starting address of the program and press NEXT. Again an address is displayed. Now enter the end address of the program and press NEXT. A dot is displayed at the end of the address field. Enter the address at which the bytes are to be entered and press NEXT. A dot is displayed again. Now enter the number of bytes to be entered and press NEXT. The system will display the address where you wish to enter the bytes with its current data in the data field. Enter the bytes displayed indicating that the system is ready to accept a new command.

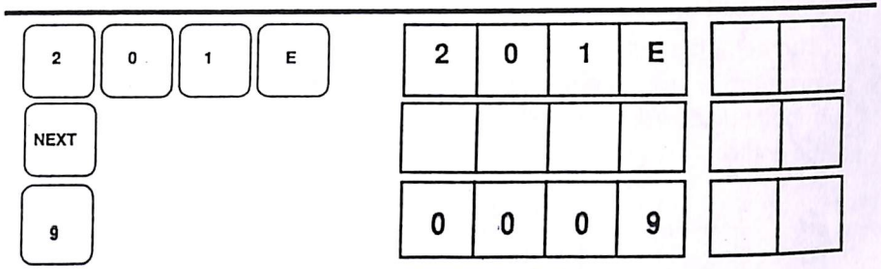


Since we have just executed this program, the program address are not disturbed. So we directly press NEXT.

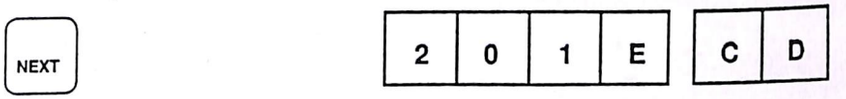


Our last address is also 2020, so we just press NEXT.

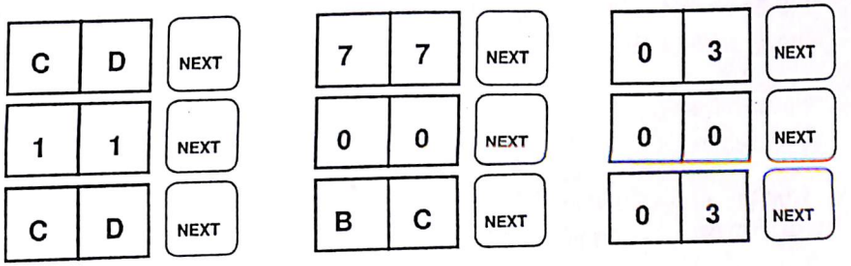




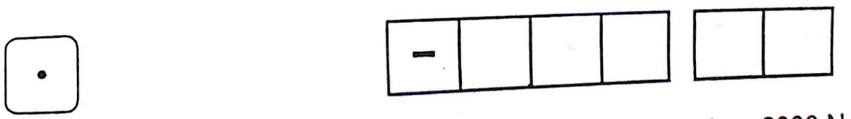
Since 9 bytes have to be entered,



the contents of 20 1E is displayed.



a '-' will be displayed.



Verify that the bytes have been inserted. Execute the program from 2000. Note that the flashing **'SUPERB'** will not display on the seven segment display. It is because the automatic modification of the routins has not done. Tis is the difference from **INSERT (INS).**

DELETE (DEL)

This command is exactly same as delete except that in this command the relocation is not done after deleting the bytes.

This command allows the user to delete one or more instructions from the user's program. In this command all the memory referenced instructions will not get modified according to the logic of the program. The following information is to be entered:

1) Starting address of the user program.
2) End address of the user program.
3) Address of the location from where onwards the bytes are to be deleted.
4) Address of the location till where the bytes are to be deleted.

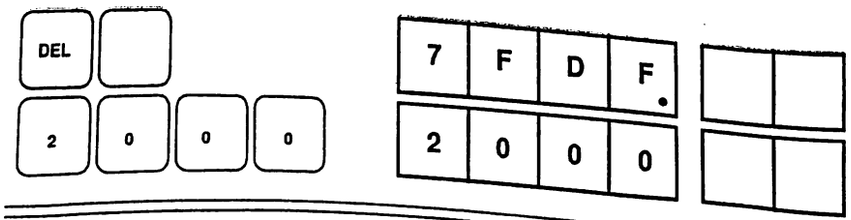
Exercise

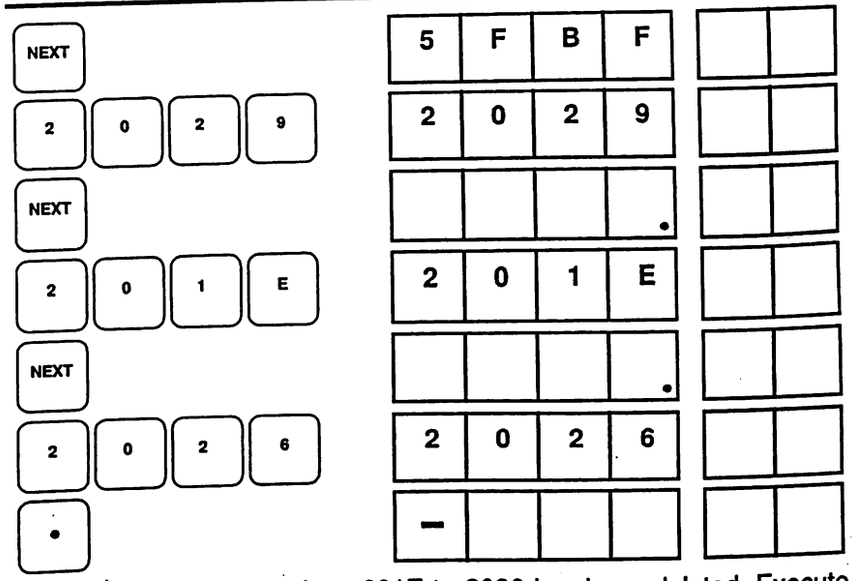
Take the program for flashing '**SUPERB**' in example 9 of Chapter-6. In this program the word '**SUPERB**' is displayed for 0.5 Sec., the display is cleared for 0.5 Sec. and the logic is repeated. In this program, if the clear routine is deleted, the word '**SUPERB**' will remain permanently displayed.

Enter this program from 2000 to 2029 using Examine/Modify memory command and delete the data from 201E to 2026.

Run this program using **GO** command before deleting the data. You will see that '**SUPERB**' is being flashed on the display. To delete the clear routine do the following:

On pressing SHIFT and DELete Key, some address is displayed. Enter the starting address of the program and press NEXT. Now enter the end address and press NEXT. A dot is displayed at the end of the Address field. Now enter the starting address from where the bytes are to be deleted and press NEXT. Again a dot is displayed at the end of address field. Enter the end address till where the bytes are to be deleted and press Terminator (.) key. A '-' will be displayed in the address field indicating that the system is ready to accept the new command.





Verify that the program from 201E to 2026 has been deleted. Execute the program. See that the word '**SUPERB**' will not be displayed permanently. It is due to the modification has not done at the corresponding routine. This is the difference from **DELETE (DEL.)**

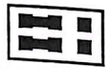
Chapter-4

ON BOARD INTERFACE

(OPTIONAL)

ONBOARD RS-232-C INTERFACE

To enhance the capabilities of VMC-850X, onboard interfaces for RS-232C have been provided. The CRT terminal provides RS-232-C interface and so a CRT terminal can be connected to VMC-850X through the standard interfaces. The RS-232-C interface is provided through SID & SOD lines of 8085 microprocessor.

Sl. No.	Interface at Connector C5	Jumper Setting at J2
1.	RS-232-C with SID/SOD lines	
2.	RS-232-C using USART	

CRT TERMINAL INTERFACE

1) Connect CRT cable to the terminal.
2) Switch ON the power supply. Press RESET key and then press 2 CRT key of the Kit.

The kit is now ready to interact with the terminal at the baud rate set earlier on the terminal.

PROCEDURE OF INTERFACING BETWEEN 8085 KIT & PC USING HYPER TERMINAL FACILITY OF WIN 95/98/ME/2000/XP.(OPTIONAL)

1. SWITCH ON PC AS WELL AS KIT.CONNECT RS-232 CABLE ON CRT CONNECTOR (C5) OF KIT TO PC COM PORT 01 OR 02.
2. IN CASE OF WITHOUT BUILT IN POWR SUPPLY KIT CONNECT PROPERLY +5,+12,-12 AND GND FROM PS-III POWER SUPPLY.
3. FOR INBUILT POWER SUPPLY NO NEED TO CONNECT THE ABOVE SUPPLIES.
4. ENABLE **HYPER TERMINAL** FROM WINDOW APPLICATION (START/PROGRAMS/ACCESSORIES/COMMUNICATIONS/HYPER TERMINAL)
5. CLICK ON **HYPERTRM** ICON AND GIVE ANY NAME (FOR SETTING PARAMETERS) AND CLICK OK.
6. CHOOSE DIRECT **TO COM1 OR COM2** AND CLICK OK.
7. SET PARAMETER AS FOLLOWS :

FOR 8085 KIT		**KEY TO BE PRESSED ON 8085 KIT**
BITS PER SECOND	**4800**	**PRESS** 2 CRT **KEY**
DATA BITS	**8**	
PARITY	**NONE**	
STOP BITS	**1**	
FLOW CONTROL	**NONE**	

8. GO ON MENU BAR SELECT FILE—PROPERTIES—SETTING—ASCII SETUP...

ASCII SENDING:	**LINE DELAY**	**20**
	CHARACTER DELAY	**20**
ASCII RECEVING:	**ENABLE**	FORCE INCOMING...
		WRAP LINES THAT...

9. AFTER PRESSING 2 CRT KEY OF KIT AND PRESS ENTER KEY

OF PC KAYBOARD, THE DISPLAY OF KIT WILL DISAPPEAR. AND "*" WILL APPEAR ON HYPER TERMINAL SCREEN.

UPLOADING PROCEDURE FROM KIT TO PC FOR 8085 KIT

1. FOLLOW THE PROCEDURE OF PREVIOUS FOR COMMUNICATION BETWEEN AND PC KIT.
2. WRITE L2000 200F (STARTING AND ENDING ADDRESS OF RAM)
3. ON MENU BAR SELECT TRANSFER—CAPATURE TEXT...
4. CLICK ON IT AND GIVE ANY NAME WITH **TXT** EXTENSION (PATH: FILES\ACCESSORIES\HYPER TERMINAL\XXX.TXT) AND CLICK ON START.
5. AFTER THAT PRESS SHIFT+$ KEY OF PC KEYBOARD, THE ADDRESS AND DATA FIELD WILL APPEAR ON HYPER TERMINAL SCREEN.
6. GO IN MENU BAR SELECT TRANSFER—CAPATURE TEXT... — STOP—FILE—SAVE.
7. BY DOING THIS YOU CAN SAVE ANY DATA IN GIVEN PATH AS ABOVE.

DOWNLOADING PROCEDURE FORM PC TO KIT FOR 8085 KIT

1. FOLLOW THE PROCEDURE OF PREVIOUS FOR COMMUNICATION BETWEEN AND PC KIT.
2. WRITE T (IN CAP. MODE) GO IN MENU BAR SELECT TRANSFER—SEND TEXT FILE...

   CLICK ON XXX.TXT FILE AND OPEN IT.
4. AFTER SOME SECOND " " WILL APPEAR AGAIN ON PRESSING ENTER KEY OF PC KEYBOARD, IT INDICATES XXX.TXT FILE IS SUCCESSFULLY LOAD IN KIT MEMORY

Chapter-5
SERIAL I/O DEVICE COMMANDS

GENERAL

The VMC-850X responds to the serial I/O device when the system mode is changed from the key board mode to a serial I/O device mode (SIOD Mode). The monitor responds by outputting a carriage return, a line feed and a prompt character (.) on the I/O device. Convey command is in the form of a single alphabetic character specifying the command, parameters are entered as hexadecimal numbers.

The NEXT command in the key board mode can be executed by space (SP), a comma (,) and a carriage return (CR). Similarly $ sign is equivalent to a EXEC key (.) in the keyboard mode. The code for each key as it is pressed on the SIOD is just echoed back to the SIOD before monitor takes any action. CR is echoed as CR and line feed (LF). Semicolon (;) is a delimiter character used in ENTER command only. A space or a carriage return can also be used as a delimiter.

The SIOD works on hexadecimal nos. all the information is to be entered in hexadecimal form. The SIOD will prompt a '.' on the outputting device for any error condition. The error conditions are similar to the key board error conditions. A carriage return, a line feed and the prompt character is given out and a new command can be entered now.

LIST OF COMMANDS

1. **L** List a memory block
2. **M** Examine/Modify Memory
3. **E** Enter a memory block
4. **R** Examine/Modify Register
5. **S** Single Step
6. **G** Go
7. **B** Block Move
8. **I** Insert
9. **D** Delete
10. **N** Insert Data

11. **O** Delete Data
12. **F** Fill
13. **H** Relocate
14. **J** Memory Compare
15. **K** String

The following pages explain these commands in detail.

COMMAND DESCRIPTION

LISTA MEMORY BLOCK

L command outputs on the SIOD device a formatted listing of memory block.

Format

L Low Address, High Address $

Type **L** followed by the starting address of the memory block to be listed, followed by a comma (,) and then the end address of the memory block followed by $.

The outputting format will be as given below:

Example

Suppose you want to list the data from 2000 to 2018.

L 2000, 2018 $

2000: AB AE CD BC AA BB BC AF CD DE

2001: AB CD DC DD EE AA BC AF BB AA

2014: BC CD AA AB FC $

EXAMINE/MODIFY MEMORY

The M command allows you to examine and modify memory locations individually. The command functions as follows:

Format

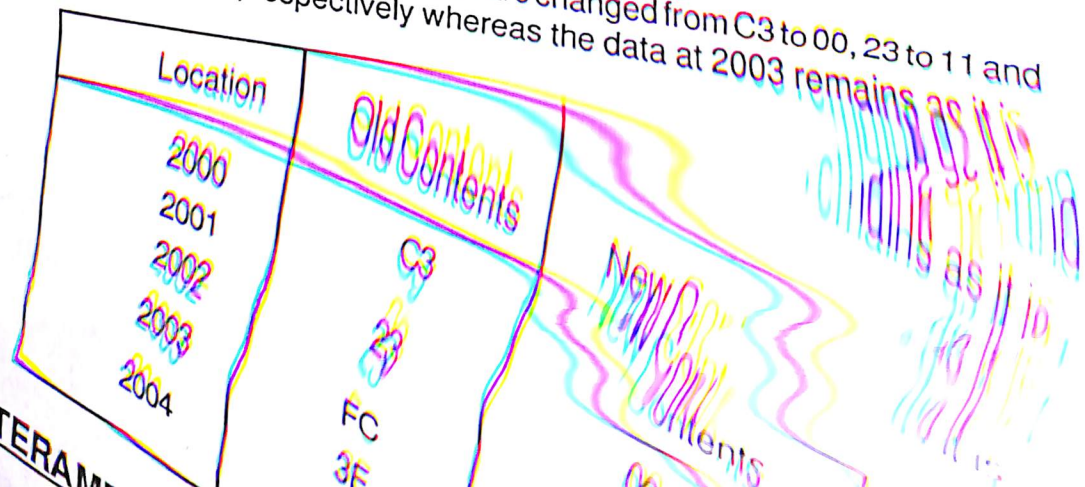
M Address, (Data),.....$

1. Type M, followed by the hexadecimal address of the first memory location you wish to examine, followed by a space or comma.
2. The contents of the location are displayed followed by a dash '-'.
3. To modify the contents of the location displayed, type the new data, followed by a comma. If you do not wish to modify the location, type only the comma. The higher memory location will automatically be displayed as the step (s).
4. A $ at any stage terminates the command.

Example

M 2000, C3-00, 23-11, FC-22, 3E, 21-44 <4>

The contents of 2000 to 2002 and 2004 are changed from C3 to 00, 23 to 11 and FC to 22 & 21 to 44, respectively whereas the data at 2003 remains as it is.



Format:

E address : data, data.....$

1. Type E followed by the starting address of the memory block to be entered, followed by a colon (:).
2. Each byte followed by a comma as it is entered from the SIOD is deposited in the consecutive location in the memory.
3. In case the terminator is colon (:) the proceeding parameter is taken as a fresh address and the subsequent data bytes are stored in memory location starting from the fresh address.
4. A $ terminates the command.

Example

E 2000: 3E, 11, 11, 08, FC $.

The memory contents as stored are shown below:

2000 3E, 11, 11, 08, FC.

EXAMINE/MODIFY REGISTER

Display & modification of CPU register is accomplished via R command.

R (Register identifier.....$.)

1. Type R followed by a single alphabet register identifier. The contents may now be changed if so desired. In case you do not want to modify the contents, just enter a comma. The contents of the next register will be printed. The register identifiers for various CPU registers are given below:

Register Identifier	Register
A	Register A
B	Register B
C	Register C
D	Register D

Register Identifier	Register
E	Register E
F	Flag byte
I	Interrupt Mask
H	Register H
L	Register L
S	Stack Point MSB & LSB
P	Program Counter MSB & LSB

SINGLE INSTRUCTIONS

This command allows to execute the program one instruction at a time.

Format

S (Starting Address) ,.....$

1. Pressing of S key will list the PC and first byte of the program. In case one want to modify it, one has to enter the new address and then press comma. The new address will be entered and pressing of comma again will execute on instruction. The new PC and the next instruction will be listed. In this way one can execute the program in single instruction mode.

Example

The following program is to be executed in single instruction mode:

Address	Op Code	Instruction
2000	11 50 20	LXI D, 2050
2003	3E 27	MVI A, 27
2005	21 70 20	LXI H, 2070
2008	77	MOV M, A

On executing S command S 2000 : 11/, 2003 : 3E/, 2005 : 21/. If one wants to execute further, one presses ',' otherwise one presses $.

GO COMMAND (G)

This command execute the program.

Format

The format for this command will be as follows:

G Starting address $.

Pressing of G key will display the PC content and the first byte of the instruction. To modify it, enter the desired address & then press comma, the PC will be modified with new contents & the corresponding data will be listed. When $ key is pressed, CPU starts executing the program. Suppose the program starts from 2000 then the format will be G 2000 $.

BLOCK MOVE COMMAND (B)

This command has the same functions as explained in BLOCK MOVE under keyboard description. The format for the command is as follows:

B

Starting address of the source

,

End address of the source

,

Starting address of the destination

$

INSERT COMMAND (I)

This command has the same functions as explained in INSERT under key board description. The format for this command is as follows:

I

Starting address of the program

,

End address of the program

,

Address from where the byte or bytes are to be entered

,

No. of bytes

,

Data bytes separated by

$

DELETE COMMAND (D)

This command has the same function as explained in DELETE under key board description. the format for this command is as follows:

D

Starting address of the program

,

End address of the program

,

Starting address from where the bytes are to be deleted

,

End address till where the bytes are to be deleted

$

INSERT DATA (N)

This command has the same function as explained in INSERT DATA under key board description. The format for this command is as follows:

N

Starting address of the program/data area,

,

End address of the program

,

Starting address at which the bytes are to be entered

,

No. of bytes

,

Data bytes separated by ','

,

Example

Suppose the RAM area 2000 to 2007 has the following data. 2000 : AF 02 00 25 57 DE 47 BC and we need to insert AA BB at 2002 onwards then the format will be N XXXX-2000, XXXX-2002, 2, ZZ-AA, BB, On pressing the last ',' the command is executed and a prompt character again appear indicating that the new command can be entered.

DELETE DATA (O)

This command has the same function as explained in DELETE DATA under key board description. The format for this command is as follows:

O
Starting address of the program/Data area,
,
End address of the program/data area
,
Starting address from where the deletion should start
,
End address till where bytes are to be deleted
$

Suppose the bytes inserted in the above example are now to be deleted. Then the format will be:

O XXXX - 2000, XXXX - 2009, 2002, 2003 $

One can verify that the bytes have been deleted.

FILL (F)

This command has the same function as explained in Chapter-3. The format for this command in serial mode will be:

F
Starting address of area
,
End address
,
Data to be filled
$

Example

Suppose the RAM area, from 2000 to 2010 is to be filled with FF. The the format will be:

F 2000, 2010, XX - FF $

XX is some random data displayed at that stage.

RELOCATE (H)

The command has the same function as explained in Chapter-3. The format for this command would be:

H
Starting address of the program
,
End address of the program
,
Destination address of the program
$

Suppose the program relocated in Chapter-3 under RELOCATE is to be relocated in serial mode, the format for this will be:

H 2000, 2029, 2100 $

MEMORY COMPARE (J)

format for this command is as follows:

J

Starting Address of first block

,

End address of first block

,

Starting address of second block

$

If the example explained earlier in Chapter - 3 for RELOCATE is to be executed in serial mode. The format will be:

J 2000, 2009, 2100 $

STRING (K)

This command has the same function as exampled in Chapter-3. The format for this command is as follows:

K

Starting address of the program

,

End address of the program

,

Address of the location at which first byte of string lies

,

Address of the location at which last byte of string lies

$

If the example of Chapter-3 under STRING is to be executed in serial mode, the format for this will be:

K XXXX-2000, XXXX-2029, 2100, 2102 $ 200B, 2015

P.S.

1) *XXXX indicates the address outputted by the system.*
2) *ZZ indicate the date outputted by the system.*

Chapter-6

SYSTEM SOFTWARE

MEMORY MAPPING

VMC-850X provides 8/32K byte of RAM and 8K byte of of EPROM. The total onboard memory can be expanded to 64K bytes. There is one memory socket provided on the board to extend o board memory.

Each socket can be defined to have any address between 0000-FFFF. The address definition and the selection of chip for each block is done by the jumpers selection on the board of VMC-850X.

Two important points should be kept in mind while defining these memory sockets for address or chips. For the system operation the monitor should start from address 0000. A minimum of 2K RAM should be there on the board with the starting address as 2000. The monitor uses certain portion of this RAM for temporary use. This area is from 2770 to 27FF. The user is advised not to use this area for storing program.

The detail of this RAM area is given here.

LOCATION	CONTENTS
2770 - 277D	For Kit Expansion
277E	SHIFT
2780	FLAG
27BD - 27BF	RST 7.5
27BA - 27BC	RST 7
27B7 - 27B9	RST 6.5
27B4 - 27B6	RST 6
27B1 - 27B3	RST 4.0
27AE - 27B0	RST 2.0
27AB - 27AD	RST 1.0
27AA - 2781	Stack
27DA	E Register
27DB	D Register

LOCATION	CONTENTS
27DC	C Register
27DD	B Register
27DE	Flags
27DF	A Register
27E0	L Register
27E1	H Register
27E2	Interrupt Mask
27E3	Program Counter LSB
27E4	Program Counter MSB
27E5	Stack Pointer LSB
27E6	Stack Pointer MSB
27E7	SP Shift
27E8 - 27E9	No. of Bytes
27EA - 27EB	Source Address I
27EC - 27ED	Destination Address I
27EE - 27EF	Offset Address
27F0 - 27F1	Lower Limit
27F2 - 27F3	Higher Limit
27F4 - 27F5	Current Address
27F6	Current Data
27F7 - 27FA	Output Buffer
27FB	Input Buffer
27FC	Temporary Location
27FD - 27FE	Half bit time
27FF	System mode

I/O MAPPING

I/O MAPPING FOR VMC-8501

Various Chips used in I/O mapped mode in VMC-8501 are 8255, 8253 & 8279.

The address for these I/O devices are given below:

Device No.	Active range Port Addresses	Port Numbers	Selected Device
8255	00-07		PPI
		00 and 04	Port A
		01 and 05	Port B
		02 and 06	Port C
		03 and 07	Control Word
8253	10-17		PIT
		10 and 14	Counter 0
		11 and 15	Counter 1
		12 and 16	Counter 2
		13 and 17	Control Word
8279	18-1F		Keyboard/ Display Controller
		18 and 1C	Mode Selector
		19 and 1D	Control Word

I/O MAPPING FOR VMC-8502

Various Chips used in I/O mapped mode in VMC-8502 are 8255, 8155, 8253 & 8279.

The address for these I/O devices are given below:

Device No.	Active range Port Addresses	Port Numbers	Selected Device
8255	00-07		PPI
		00 and 04	Port A
		01 and 05	Port B
		02 and 06	Port C
		03 and 07	Control Word
8155	08-0F		PPI & TIMER
		08	Control Word
		09	Port A
		0A	Port B
		0B	Port C
		0C	LSB Timer
		0D	MSB Timer
8253	10-17		PIT
		10 and 14	Counter 0
		11 and 15	Counter 1
		12 and 16	Counter 2
		13 and 17	Control Word
8279	18-1F		Keyboard/ Display Controller
		18 and 1C	Mode Selector Control Word
		19 and 1D	

USEFUL SOFTWARE ROUTINES

VMC-850X monitor program needs various utility functions while using the onboard keyboard and display CRT Terminals as a console. These utility functions (routines) resides in the 4K Bytes of Monitor and so can also be used by the user to simplify his programming task. Some of these utilities are given here.

Address of routine	Label	Description	Registers Affected
030B	CH INP	**Character Input** This routine takes one 8 bit byte from the serial I/o Port and return it to the calling routine in A register. The 16 bit number stored in 27FD (LSB) and 27FE (MSB) decides the baud rate. The number decides the half bit time and is the argument of subroutine DELAY.	A,B,C,D,E, H,L & F/FS
032F	CLEAR	This routine is for keyboard only. Input : B-Dot,Flat = 1 for a dot 8 in address field O fo not dot Output : None	A,B,C,D,E, H,L & F/FS
033F	CLEAR DISPLAY	This routine clears te display and terminate the comand. The inputs and outputs in this routine are one	All
0366	CNASC	This routine converts a hex nibble to ASCII. The various inputs and outputs in this routine are: Input: C-hexcode to be converted to ASCII. Output: C-ASCII Code	A,B,C,H&L

Address of routine	Label	Description	Registers Affected
036E	CNABN	This routine converts an ASCII character to its Binary Value. The various Inputs and outputs required are Reg. A-7 Bit ASCII code with parity bit = 0, Output Reg. A : 4 Bit Hex Nibble when Input is any hex numeric 0-F. Reg. A: 10 when Input is $ Reg. A: 11 when Input is SP/ CR Reg A: 3A when Input is:, for any other Input it jumps to error.	All
038A	CHOUT	**Character Output:** This routine takes one byte (8 bit) passed on by the calling routine in register C on the I/O Port. The baud rate serial is decided by the 16 bit number stored in 27FD (LSB) and 27FE (MSB). The number determines the half bit time and is the argument of the subroutine DELAY.	A,B,C,D,E, H,L & F/FS
03B0	DCRNB	**Decrements a Byte** This routine decrements a Byte and if the decremented value is zero it sets the zero flag. Input: None Output: the zero flag gets set if the decremented value = 0 otherwise it is reset.	A & F/FS

Address of routine	Label	Description	Registers Affected
03BC	DELAY	The routine is used to provide delays. It stores the number in FS register pair D, counts it down to zero & comes back to the calling routine. Total time delay introduced by the routine is (24N + 17) x basic machine cycle. N # ()	A,D,E & F
03C3	DISPLPC	**DISPLAY PC CONTENT:** This routine displays the PC Content and the first byte of the instruction stored. The inputs and outputs in this case are none.	All
03E2'	DMDT	**DISPLAY/MODIFY DATA:** This routine display/modify the data. The Inputs and outputs in this routine are: Input: HL - 1 If a memory address was received. - O If no data was received A - Termination Character DE - The data received HL - Same as Input	A,B,C,D,E & F/FS
03FA	ECHO	This routine is used in SIOD mode, when the keyboard is in active. This routine outputs a single character to the user on	A,B,D & E

Address of routine	Label	Description	Registers Affected
		SIOD. The Inputs and Outputs are Reg. C. Character to ECHO to the terminal. Output: Reg. C: Same as Input.	
0417	EMM	This routine examine/modify a (Common) memory Block. The Inputs for this routine are HL-First memory location to be examined. Outputs: None.	A,B,C,H,L & F/FS
0455	ERROR	This routine displays an error on the display if the system is in Key Board Mode. And if the system is in SIOD Mode, it lists a * and goes to command recognizer. The various Inputs and Outputs are: Input : None Output : None	All
0468	RECHEX	**Receive Hex Digits:** If the system is in Keyboard mode, this routine accepts, a	All

Address of routine	Label	Description	Registers Affected
		hexadecimal number from it, displays it on the display and returns it as a 16 bit number. The routine displays error in case of an invalid terminator. The valid terminators are EXECUTE, NEXT AND PREVIOUS. RST 5.5 should be unmasked first in this operation. The various Inputs and Outputs are Reg. Reg.B: 0 - Use address field. Reg.B: 1 - Use data field. Outpus: A - The last character received from the Keyboard. D,E - last 4 hex digits received from the keyboard. Carry - Set if atleast one valid hex digit was received else reset. If the system is in SIOD mode then the number is received from the appropriate SIOD and the number is displayed. Valid terminators are $, '.' space and ':'.	
04C5		**HILO (COMMON) DE = DE - HL:**	A,D,E,F/ FS

Address of routine	Label	Description	Registers Affected
		In this routine DE Reg. contents become equal to DE Reg. Content - HL Reg. Contents. CRY Set if DE was <HL and reset if DE was> HL.	
04EE	RCVCHR	**Receive a character from SIOD** This routine waits for a character to be typed on the SIOD. When the character is typed, it sends it back to the SIOD and returns the ASCII code value to the calling routine in Register A. Selection of any of the device can be done by changing the content of the location - 27FF. 27FF Mode 00 Invalid 01 TTY 02 CRT If the MS bit of this code is set, ten there will not be any echo of the character, location 27FD stores half bit time for proper baud rate output - ASCII code received from SIOD.	A,B,C,D,E, H,L & F/FS
0519	INSDG	**Insert Hex Digit:** This routine allows to insert hex digit. The Inputs and Outputs in this routine are:	A,D,E,H,L & F/FS

Address of routine	Label	Description	Registers Affected
		Input: Reg. A - Hexdigit to be Inserted. Reg. DE - Hex Value. Output: Reg. DE - Hex value with digit inserted.	
0552	LSTBTE	**List a byte on SIOD** This routine takes a byte from SIOD and converts it into two hexadecimal digits for listing on SIOD. The byte to be listed is stored in Reg. A and the SIOD is selected by changing the contents of the location 27FF. Contents of location (Mode) 27FF = 00 invalid 01 TTY 02 CRT Location 27FF stores half bit time for proper baud rate.	A,B,C,D,E & F/FS
05AF	New Ad	**Read New Address:** This routine reads keyboard/SIOD to find out if there is a new address. If the terminator is EXEC, it jumps to clear display. In case of any terminator other than EXEC it gives error. The various Inputs and Outputs are: Input: H1 - Old address. Output CRY: 01 if no new value is received else CRY is	All

Address of routine	Label	Description	Registers Affected
		resetted. HL: New value of address if received otherwise 00.	
05D0	OUTPUT	**Display the Character** This routine is used to display a character on the display. If the system is in SIOD Mode, the routine returns without any operation. The various inputs to this routine are given below: Reg.A: 0 - Use Address Field. 1 - Use Data Field. Reg. B: 0 - No dot to be displayed 1 - Dot at the right edge of the field. Reg. H.L. - Starting address of the Character Code. Character Displayed / Character Code 0 00 1 01 2 02 3 03 4 04 5 05 6 06 7 07	A',B,C,D,E, F,L&F/FS

Address of routine	Label	Description	Registers Affected
		Character Displayed / Character Code 8 08 9 09 A 0A B 0B C 0C D 0D E 0E F 0F H 10 L 11 P 12 I 13 R 14 U 15 BLANK 16	
0629	RD KB	**Read the Keyboard** This routine scans the key board for any key to be pressed. When a key is pressed, its value is returned to the calling routine in register A RST 5.5 should be first un-masked for the proper operation. Input: A - The byte to be searched. DE - No. of bytes is in each entry. HL - Starting address of the table. C - No. of entries in the table.	A,H,L & F/FS

Address of routine	Label	Description	Registers Affected
		Output: HL - address where the comparison was made. C - decremented value of the counter.	
06E3	MOD AD	**Modify Address** If the system is under the control of keyboard, it displays 16 bit number stored in 27F4 and 27F5. Input: Reg.B - 0 no dot. 1 dot at the right edge of the field. If the system is in SIOD mode, 4 hex digits are listed on SIOD.	A,B,C,D,E, H,L & F/FS
06FA	MODDT	**Modify Data** If the system is under the control of keyboard, it displays 8 bit number stored in 27F6. Input: Reg.B - 0 no dot. 1 dot at the right edge of the field. If the system is in SIOD mode, 2 hex digits are listed on the SIOD.	A,B,C,D,E, H,L & F/FS

Address of routine	Label	Description	Registers Affected
0639	RD HILO	**Read HI and LO** This routine reads the lower limit and higher limit of addresses from the keyboard. If no new address are entered, then the lower and higher limit address are not disturbed. Input: None Output: None	All
0653	REL	**Relocate a Program** This routine relocates a program written for one memory area to another memory area by adding a fixed 16 bit number to all the address references between and including (27EA, 27EB) and (27EC, 27ED). Lower Limit (27F0, 27F1) should point to the starting address of the routine to be relocated and the higher limit (27F2, 27F3) should point to the last location.	All
06CB	RD2AD	**Read two addresses** This routine reads two addresses from key board or SIOD. The inputs and outputs are: Input: None Output: Reg.BC - Ist address Reg.DE - IInd address	A,B,C,D,E, H,L & F/FS

Address of routine	Label	Description	Registers Affected
		Reg.A - Terminator of the IInd address	
06D0	TSRCH	**Search a Table** This routine searches a table for a byte and if the search fails, it gives error.	C,H,L & F/ FS

DEVELOPING/DEBUGGING SOFTWARE

VMC-850X provides software features like Relocate, String, Insert, Delete etc. which find extensive application in developing/debugging software. The various steps involved in developing software are:

1. Define the problem in the form of a flow chart.
2. Write the program in Assembly Language of 8085.
3. Assemble the program using the Reference card provided with the Manual.
4. Enter the program into RAM area and Run it. For running the program use GO Command. It is likely that the program may not run in one shot because some mistake is there in it. The process of finding this mistake and removing it is called the debugging of the program.

One way of finding the mistake in the program is to run the program in single instruction mode and after each step compare what the program is doing and what is supposed to do. In the process of this one might have to examine the contents of memory locations or the contents of µP internal registers after the execution of each instruction.

BREAKPOINT SETTING OF VMC-850X

During this process of debugging some time one might just like to examine the status of the program at a particular point. If this point is near the beginning of the program, one can reach this point by single instruction facility. But, if this point is quite far from the beginning of the program,

it is time saving to make use of break point facility. **For this introduce a RST5 instruction (EF) at the point to be examined and run the program at full speed using GO Command. When, during the execution of the program, this instruction is encountered, the control of the processor is transferred to the monitor. The monitor saves the user registers and displays a sign - UP 85 on the display. Now one can examine the status of any memory location or nay internal register. One can change the content of memory location or register if necessary.IT IS VERY IMPORTANT NOT TO PRESS RESET KEY AFTER GO COMMAND. USE SHIFT & EXEM REG KEY TO READ/ MODIFY THE RESISTORS.**

Sometimes while debugging one may find that certain instructions are to be added to the program or to be deleted from the program. This can be done by using insert or delete instruction. The programs written for one memory area can be made operative for some other area using the RELOCATE command. For details of Insert, Delete and Relocate command refer to Chapter-3 of Keyboard Description.

SAMPLE PROGRAMS

PROGRAM-1

Hexadecimal additions of two numbers.

DESCRIPTION

The program takes the content of 2000, adds it to 2001 and stores the result back at 2002. **Excute the prorgram from 2003 location.**

Steps

1. Initialize HL Reg. pair with address where the first number is lying.
2. Store the number in accumulator.
3. Get the second number.
4. Add the two numbers and store the result in 2002.
5. Go back to Monitor. **Excute from 2003 location.**

Address	Code	Label	Mnemonic	Operand	Comments
2000	DATA				1st no. to be added
2001	DATA				2nd no. to be added
2002	RESULT				Result
Step-1					
2003	21 00 20	START	LXI	H,2000	Point to 1st no.
Step-2					
2006	7E		MOV	A,M	Load the accumulator
Step-3					
2007	23		INX	H	Advance pointer
Step-4					
2008	86		ADD	M	Add IInd number
2009	23		INX	H	Advance pointer
200A	77		MOV	MA	Store result
Step-5					
200B	EF		RST	5	

Example

Address	Data
2000	1A
2001	18
2002	32 Data in Hex Number

PROGRAM-2

The decimal addition of two decimal numbers. The result should not be greater than 199.

DESCRIPTION

The program will add two decimal numbers lying at 2000 and 2001. The result will be stored (in decimal) at 2002.

Steps

1. Load the contents of first location in Acc.
2. Add it with the contents of second location.
3. Adjust the decimal.
4. Store the result and go back to monitor. **Excute from 2003 location.**

Address	Code	Label	Mnemonic	Operand	Comments
2000		DATA			Two decimal nos. to be added.
2001		DATA			
2002		RESULT			Result
Step-1					
2003	21 00 20		LXI	H,2000	Point to 1st no.
2006	7E		MOV	A,M	Load it in to accumulator
Step-2					
2007	23		INX	H	Point to 2nd no.
2008	86		ADD	M	Add the two nos.

Step-3

2009	27	DAA		Convert to decimal

Step-4

200A	23	INX	H	Point to storage
200B	77	MOV	M,A	Store it
200C	EF	RST	5	

Example

Address	Data	
2000	23	Data in decimal
2001	32	Data in decimal
2002	55	Answer in decimal

PROGRAM-3

Add two sixteen bit numbers.

DESCRIPTION

This program will add two sixteen bit numbers lying at 2000, 2001, 2002 and 2003. The result is stored at 2004 and 2005. Carry is neglected.

Steps

1. Load the first number in HL.
2. Load the second number.
3. Add the two and store result.
4. Go back to monitor. **Excute from 2006 location.**

Address	Code	Label	Mnemonic	Operand	Comments
2000	1st no.	LSB			1st sixteen bit number
2001	1st no.	MSB			
2002	2nd no.	LSB			2nd sixteen bit number
2003	2nd no.	MSB			
2004	RESULT	LSB			Result of addition
2005	RESULT	MSB			

Step-1

2006	2A 00 20	START	LHLD	2000	Load the HL Reg. with 1st number
2009	EB		DCHG		Exchange the HL and DE register.

Step-2

200A	2A 02 20		LHLD	2002	Load the HL Reg. with 2nd number.

Step-3

200D	19		DAD		Add HL & DE Registers
200E	22 04 20		SHLD	2004	Store the result in 2004.

Step-4

2011	EF		RST	5	

Example

Address	Data	
2000	CA	LSB1
2001	A7	MSB1
2002	6B	LSB2
2003	B9	MSB2
2004	35	LSB1 + LSB2 (CA + 6B)
2005	61	MSB1 + MSB2 (A7 + B9)

PROGRAM-4

Addition of a 8 bit number series neglecting the carry generated. This program adds 'N' No. of hexadecimal numbers lying from 2101 onwards. The 2100 has the number of hexadecimal bytes to be added. The result is stored at 2100 location.

Steps

1. Load the contents from 2100 location, give how many bytes are to be added.
2. Initialize Acc. as the result will be stored in Acc.
3. Let the memory to point the number of the bytes to be added into partial register Acc.
4. Decrement the counter having no. of bytes.
5. Check if zero - No. repeat from point 3.
6. Store the result at 2100 location.
7. Go back to monitor. **Excute from 2000 location.**

Address	Code	Label	Mnemonic	Operand	Comments
Step-1					
2000	21 00 21	START	LXI	H,2100	Point to first no.
2003	46		MOV	B,M	Load count into B Register
Step-2					
2004	AF		XRA	A	Clear A Register
Step-3					
2005	23	LOOP	INX	H	Point to the 1st number
2006	86		ADD	M	Add memory to total
Step-4					
2007	05		DCR	B	Substract from count
Step-5					
2008	C2 05 20		JNZ	LOOP	Test to see if done
Step-6					
200B	32 00 21		STA	2100	Save the result
Step-7					
200E	EF		RST	5	

Example

Address	Data	
2100	04	The no. of hexadecimal no. to be added
2101	10	First Hex No.
2102	02	Second Hex No.
2103	08	Third Hex No.
2104	04	Fourth Hex No.

After the program executes the answer will be stored in 2100.

2100	1E	Answer is in Hexadecimal.

PROGRAM-5

Separation of hexadecimal number into two digits. This program breaks the byte of data stored at 2100 into two nibbles. The MS nibble and LS nibble are stored at 2102 and 2101 location respectively.

Steps

1. Load the byte into Acc.
2. Clear the MS nibble and store it at 2101.
3. Load the byte from 2100.
4. Clear the LS nibble and store it at 2102.
5. GO BACK TO Monitor. **Excute from 2000 location.**

Address	Code	Label	Mnemonic	Operand	Comments
Step-1					
2000	3A 00 21	START	LDA	2100	Get the number
Step-2					
2003	E6 0F		ANI	0F	Mask off the first four bits
2005	32 01 21		STA	2101	Store result 1
Step-3					
2008	3A 00 21		LDA	2100	Get the number again

Step-4

200B	E6 F0	ANI	F0	Mask off the last four bits
200D	32 02 21	STA	2102	Store result 2

Step-5

2010	EF	RST	5	

Example

Address	Data	
2100	AF	Number stored
2101	00	Memory before program execution
2102	00	Memory before program execution

After the program executes:

2100	AF	Number stored
2101	0F	LSB separated from AF
2102	A0	MSB separated from AF

PROGRAM-6

Combination of two hex nibbles to from one byte number. The program takes two nibbles from 2100, 2101 and combines to form a byte. The nibbles from 2100 is taken as MS nibble of the byte to be formed.

Steps

1. Initialize memory pointer with 2100 and load MS nibble into Acc.
2. Shift the four bits towards the left, the shifted information is stored in Acc.
3. Point to next location 2101 and 'OR' its contents with the contents of Acc.
4. Increment the memory pointer to store the result at 2102.
5. Go back to Monitor. **Excute from 2000 location.**

Address	Code	Label	Mnemonic	Operand	Comments
Step-1					
2000	21 00 21	START	LXI	H,2100	Point to 1st number
2003	7E		MOV	A,M	Get it
Step-2					
2004	07		RLC		
2005	07		RLC		Move it to MSB
2006	07		RLC		
2007	07		RLC		
Step-3					
2008	23		INX	H	Point to next number
2009	B6		ORA	M	OR the two numbers together
Step-4					
200A	23		INX	H	
200B	77		MOV	M,A	Store the result
Step-5					
200C	EF		RST	5	

Example

Address	Data	
2100	04	MSB
2101	05	LSB
2102	00	Memory before program execution

After the program execution:

2100	04	MSB
2101	05	LSB
2102	45	Memory after program execution

PROGRAM-7

This program checks the hex number stored in location 2010 for odd or even parity. If the parity is odd, 00 will be stored in location 2011. Otherwise EE is stored in 2011.

Steps

1. Set the memory counter (Register pair HL) to the data location 2010 and bring its contents to Acc.
2. "OR" the contents of Accumulator with itself.
3. Check the parity register for odd or even parity and store 00 or EE in location 2011 depending upon whether it is odd or even parity.
4. Go back to Monitor. **Excute from 2000 location.**

Address	Code	Label	Mnemonic	Operand	Comments
Step-1					
2000	21 10 20		LXI	H,2010	Set the memory counter
2003	7E		MOV	A,M	Get the first no. in Acc.
Step-2					
2004	B7		ORA	A	Set the flag.
Step-3					
2005	E2 0C 20		JPO	ODD	If the parity is odd, jump to ODD.
2008	2C		INR	L	Point to memory location for result
2009	36 EE		MVI	M,EE	Store EE in 2011
200B	EF		RST	5	Go to Monitor
200C	2C	ODD	INR	L	Point to memory location for result
200D	36 00		MVI	M,00	Store 00 in 2011
Step-4					
200F	EF		RST	5	Go to Monitor

2010	Data
2011	Result

Example

Address	Data	
2010	10	Before execution 2011-00
2010	30	After execution 2011-EE

PROGRAM-8

Multiplication by two, employing bit rotation. The program multiplies a hex number stored in location 200A with two and stores the result in location 200B. It uses the bit rotation technique.

Steps

1. Load the data of location 200A in the Acc.
2. Set the carry flag to zero.
3. Rotate the Acc. through carry.
4. Store the contents of Acc. in 200B.
5. Go back to monitor. **Excute from 2000 location.**

Address	Code	Label	Mnemonic	Operand	Comments
Step-1					
2000	3A 0A 20	START	LDA	200A	Load the number in Acc.
Step-2					
2003	37		STC		Set the carry flag to 1
2004	3F		CMC		Complement the carry (Store 0 in the carry flag).
Step-3					
2005	17		RAL		Rotate Acc. to left, through Carry (Multiply by 2).

Step-4

2006	32 0B 20	STA	200B	Store the result in 200B.

Step-5

2009	EF	RST	5	Go to Monitor
200A	Data			
200B	Result			

Example

Address	Data
Before execution	
200A	1A
After execution	
200B	34

PROGRAM-9

Display flashing 'SUPERB'

This program will display a flashing 'SUPERB' in address and data field. The flashing rate is 0.5 sec.

Steps

1. Initialize the stack pointer.
2. Clear the display.
3. Point to the data of SUPERB and display SUPE in the address field and RB in the data field.
4. Wait for 0.5 Sec.
5. Clear the display and wait for 0.5 sec.
6. Jump to START.

Address	Code	Label	Mnemonic	Operand	Comments
Step-1					
2000	31 FF 20		LXI	SP,20FF	Initialize stack pointer

Step-2

2003	CD 47 03	CALL	CLEAR		Clear the display
Step-3					
2006	AF	START	XRA	A	A is 00 to display character in the address field.
2007	47		MOV	B,A	No dot to be displayed in the data field.
2008	21 50 20		LXI	H,2050	Starting address where the display is to be started.
200B	CD D0 05	CALL	OUTPUT		Call OUTPUT routine to display four characters in address field.
200E	3E 01		MVI	A,01	Initialize A,B for data to be displayed in data field.
2010	06 00		MVI	B,00	
2012	21 54 20		LXI	H,2054	HL points the address where the data to be displayed is lying.
2015	CD D0 05	CALL	OUTPUT		Display last two characters of SUPERB in data field.
2018	11 00 00		LXI	D,0000	Display SUPERB for about 0.5 seconds
201B	CD BC 03	CALL	DELAY		
Step-5					
201E	CD 47 03	CALL	CLEAR		Clear the display
2021	11 00 00		LXI	D,0000	Clear display for 0.5 sec.
2024	CD BC 03	CALL	DELAY		
2027	C3 06 20	JMP	START		
Step-6					
2050	**05 15 12**				Data for display
2053	**0E 14 0B**				

Chapter-7

SYSTEM EXPANDABILITY

ONBOARD EXPANSION

VMC-850X provides 8/32K bytes of RAM and 8K bytes of EPROM. The total memory on the board can be expanded to 64K bytes in a suitable combination of RAM/EPROM. Each socket can be defined to have any of the Chip 6116/ 6264/2764/27128/27256 and depending upon the chip selected, the total on board capacity will vary. The details about how a address can be allocated to any memory space or how a chip can be selected for any memory space is discussed in this Chapter.

The VMC-8501 provides only 24 I/O lines and VMC-8502 provides 24 I/ O lines using 8255 which can be expanded to 46 I/O lines using 8255 & 8155. It has a 14 bit timer/counter facility on the board.

The kit address data are available on 50 Pin FRC STD Bus which can be used to expand the Kit with other STD BUS Cards.

MEM0	0000 - 1FFF E000 - FFFF
MEM1	2000 - 3FFF
MEM2 (OPTIONAL)	8000 - FFFF

DETAILS OF CONNECTIONS

SIGNALS AT CONNECTOR-C1 - BUS CONNECTOR

PINS	SIGNALS	PINS	SIGNALS
1	+5V DC	2	+5V DC
3	GND	4	GND
5	D3	6	D7
7	D2	8	D6
9	D1	10	D5
11	D0	12	D4
13	A7	14	A15
15	A6	16	A14
17	A5	18	A13
19	A4	20	A12
21	A3	22	A11
23	A2	24	A10
25	A1	26	A9
27	A0	28	A8
29	WR	30	RD
31	IORQ	32	MEMRQ
33	RST 6.5	34	RST 6.5
35	ALE	36	ALE
37	S1	38	S0
39	BUSACK	40	BUSRQ
41	INTA	42	INTR
43	READY	44	NMI
45	RESET OUT	46	RESET IN
47	CLOCK OUT	48	I/O CS 28-2F
49	RAM CS	50	8000-8FFF

SIGNALS AT CONNECTOR-C2 - TIMER

PINS	SIGNALS	PINS	SIGNALS
1	GATE0	2	GND
3	OUT0	4	OUT1
5	CLK0	6	GATE1
7	GATE2	8	CLK1
9	OUT2	10	CLK2

SUPPLY AT CONNECTOR
POWER SUPPLY CONNECTOR

PINS	SIGNALS
1	+5V
2	GND
3	+12V
4	-12V

SIGNALSAT CONNECTOR-C6
8155 TIMER, I/O CONNECTOR FOR 8502

PINS	SIGNALS	PINS	SIGNALS
1	P1C4	2	P1C5
3	P1C2	4	P1C3
5	P1C0	6	P1C1
7	P1B6	8	P1B7
9	P1B4	10	P1B5
11	P1B2	12	P1B3
13	P1B0	14	P1B1
15	P1A6	16	P1A7

PINS	SIGNALS	PINS	SIGNALS
17	P1A4	18	P1A5
19	P1A2	20	P1A3
21	P1A0	22	P1A1
23	TIMER IN	24	TIMER OUT
25	GND	26	GND

SIGNALS AT CONNECTOR-C4 8255-I - PPI CONNECTOR

PINS	SIGNALS	PINS	SIGNALS
1	P2C4	2	P2C5
3	P2C2	4	P2C3
5	P2C0	6	P2C1
7	P2B6	8	P2B7
9	P2B4	10	P2B5
11	P2B2	12	P2B3
13	P2B0	14	P2B1
15	P2A6	16	P2A7
17	P2A4	18	P2A5
19	P2A2	20	P2A3
21	P2A0	22	P2A1
23	P2C6	24	P2C7
25	GND	26	GND

SIGNALS AT CONNECTOR-C5 - RS-232-C
CONNECTOR FOR 850X

PINS	SIGNALS
1	CRT IN
2	CRT OUT
3	GND
4	GND

APPENDIX-A

INTERRUPTS IN 8085 KIT

The 8085 has 5 interrupt inputs: INTR, RST 5.5, RST 6.5, RST 7.5, and TRAP. Each of three RESTART inputs, 5.5, 6.5, 7.5, has a programmable mask. TRAP is also a RESTART interrupt except it is non-maskable.

The three RESTART interrupts cause the internal execution of RST (saving the program counter in the stack and branching to the RESTART address) if the interrupts are enabled and if the interrupt mask is not set. The nonmaskable TRAP causes the internal execution of a RST independent of the state of the interrupt enable or masks.

Name	RESTART ADDRESS (HEX)	
TRAP	24_{16}	(used in 8253 timer for Single Step Instruction execution)
RST 5.5	$2C_{16}$	(used in keyboard controller 8279)
RST 6.5	34_{16}	(Free for the user and its address is in RAM location is 27B7 to 27B9)
RST 7.5	$3C_{16}$	(used in keyboard and address is 27BD to 27BF)

There are two different types of inputs in the restart interrupts. RST 5.5 and RST 6.5 are high level-sensitive like INTR (and INT on the 8080) and are recognized with the same timing as INTR. RST 7.5 is rising edge-sensitive. For RST 7.5, only a pulse is required to set an internal flip flop which generates the internal interrupt request. The RST 7.5 request flip flop remains set until the request is serviced. Then it is reset automatically.

The status of the three RST interrupt masks can only be affected by the SIM instruction and RESET IN.

The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows: TRAP - highest priority. RST 7.5, RST 6.5, RST 5.5, INTR - lowest priority. This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt RST 5.5 can interrupt a RST 7.5 routine if the interrupts were re-enabled before the end of the RST 7.5 routine.

The TRAP interrupt is useful for catastrophic errors such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the highest priority. It is not affected by any flag or mask. The TRAP input is both edge and level sensitive. The TRAP input must go high and remain high to be acknowledged, but will not be recognized again until it goes low, then high again. This avoids any false triggering due to noise or logic glitches.

Example-1

The following examples illustrate the use of RST 5.5 used in keyboard controller 8279 to get the scan code and display in the data field.

Address	Code	Label	Mnemonic	Operand	Comments
2000	3E 0E		MVI	A,0E	Initialize RST 5.5
2002	30		SIM		Set interrupt mask
2003	FB		EI		Enable interrupt
2004	CD 29 06		CALL	RD KB	Read keyboard
2007	32 F6 27		STA	27F6	Store current data
200A	CD FA 06		CALL	MODDT	Display the scan code on data field
200D	C3 00 20		JMP	2000	Loop

Execute from 2000 location. Whenever any key is pressed, it displays the scan code of that particular key.

Example-2

The following program will be demonstrate the use of RST 7.5 as VEC INT key on the keyboard. The program will display '**Err**' whenever the vector interrupt key is pressed.

Address	Code	Label	Mnemonic	Operand	Comments
2000	3E 0B		MVI	A,0B	Initialize RST 7.5
2002	30		SIM		Set interrupt mask
2003	FB		EI		Enable interrupt
2004	C3 04 20		JMP	2004	Loop

Now write the program for interrupt service routine at 2100 location

2100	CD 55 04	CALL	Err	This displays Err on the display field
2103	76	HLT		

Now enter the jump address of interrupt service routine at 27BD.

27BD	C3 00 21	JMP	2100	Jump to interrupt service routine.

Execute the program from 2000 and press vector interrupt (VEC INT) key and displays the service routine Err.

APPENDIX-B

INSTRUCTION SET FOR 8085 MICROPROCESSOR

HEX	MNEMONIC		HEX	MNEMONIC	
CE	ACI	8-Bit	3F	CMC	
8F	ADC	A	BF	CMP	A
88	ADC	B	B8	CMP	B
89	ADC	C	B9	CMP	C
8A	ADC	D	BA	CMP	D
8B	ADC	E	BB	CMP	E
8C	ADC	H	BC	CMP	H
8D	ADC	L	BD	CMP	L
8E	ADC	M	BE	CMP	M
87	ADD	A	D4	CNC	16-Bit
80	ADD	B	C4	CNZ	16-Bit
81	ADD	C	F4	CP	16-Bit
82	ADD	D	EC	CPE	16-Bit
83	ADD	E	FE	CPI	8-Bit
84	ADD	H	E4	CPO	16-Bit
85	ADD	L	CC	CZ	16-Bit
86	ADD	M	27	DAA	
C6	ADI	8-Bit	09	DAD	B
A7	ANA	A	19	DAD	D
A0	ANA	B	29	DAD	H
A1	ANA	C	39	DAD	SP
A2	ANA	D	3D	DCR	A
A3	ANA	E	05	DCR	B
A4	ANA	H	0D	DCR	C
A5	ANA	L	15	DCR	D
A6	ANA	M	1D	DCR	E
E6	ANI	8-Bit	25	DCR	H
CD	CALL	16-Bit	2D	DCR	L
DC	CC	16-Bit	35	DCR	M
FC	CM	16-Bit	0B	DCX	B
2F	CMA		1B	DCX	D

HEX	MNEMONIC		HEX	MNEMONIC	
2B	DCX	H	01	LXI	B, 16-Bit
3B	DCX	SP	11	LXI	D, 16-Bit
F3	DI		21	LXI	H, 16-Bit
FB	EI		31	LXI	SP, 16-Bit
76	HLT		7F	MOV	A,A
DB	IN	8-Bit	78	MOV	A,B
3C	INR	A	79	MOV	A,C
04	INR	B	7A	MOV	A,D
0C	INR	C	7B	MOV	A,E
14	INR	D	7C	MOV	A,H
1C	INR	E	7D	MOV	A,L
24	INR	H	7E	MOV	A,M
2C	INR	L	47	MOV	B,A
34	INR	M	40	MOV	B,B
03	INX	B	41	MOV	B,C
13	INX	D	42	MOV	B,D
23	INX	H	43	MOV	B,E
33	INX	SP	44	MOV	B,H
DA	JC	16-Bit	45	MOV	B,L
FA	JM	16-Bit	46	MOV	B,M
C3	JMP	16-Bit	4F	MOV	C,A
D2	JNC	16-Bit	48	MOV	C,B
C2	JNZ	16-Bit	49	MOV	C,C
F2	JP	16-Bit	4A	MOV	C,D
EA	JPE	16-Bit	4B	MOV	C,E
E2	JPO	16-Bit	4C	MOV	C,H
CA	JZ	16-Bit	4D	MOV	C,L
3A	LDA	16-Bit	4E	MOV	C,M
0A	LDAX	B	57	MOV	D,A
1A	LDAX	D	50	MOV	D,B
2A	LHLD	16-Bit	51	MOV	D,C

HEX	MNEMONIC
52	MOV D,D
53	MOV D,E
54	MOV D,H
55	MOV D,L
56	MOV D,M
5F	MOV E,A
58	MOV E,B
59	MOV E,C
5A	MOV E,D
5B	MOV E,E
5C	MOV E,H
5D	MOV E,L
5E	MOV E,M
67	MOV H,A
60	MOV H,B
61	MOV H,C
62	MOV H,D
63	MOV H,E
64	MOV H,H
65	MOV H,L
66	MOV H,M
6F	MOV L,A
68	MOV L,B
69	MOV L,C
6A	MOV L,D
6B	MOV L,E
6C	MOV L,H
6D	MOV L,L
6E	MOV L,M
77	MOV M,A
70	MOV M,B

HEX	MNEMONIC
71	MOV M,C
72	MOV M,D
73	MOV M,E
74	MOV M,H
75	MOV M,L
3E	MVI A, 8-Bit
06	MVI B, 8-Bit
0E	MVI C, 8-Bit
16	MVI D, 8-Bit
1E	MVI E, 8-Bit
26	MVI H, 8-Bit
2E	MVI L, 8-Bit
36	MVI M, 8-Bit
00	NOP
B7	ORA A
B0	ORA B
B1	ORA C
B2	ORA D
B3	ORA E
B4	ORA H
B5	ORA L
B6	ORA M
F6	ORI 8-Bit
D3	OUT 8-Bit
E9	PCHL
C1	POP B
D1	POP D
E1	POP H
F1	POP PSW
C5	PUSH B
D5	PUSH D

HEX	MNEMONIC
E5	PUSH H
F5	PUSH PSW
17	RAL
1F	RAR
D8	RC
C9	RET
20	RIM
07	RLC
F8	RM
D0	RNC
C0	RNZ
F0	RP
E8	RPE
E0	RPO
0F	RRC
C7	RST 0
CF	RST 1
D7	RST 2
DF	RST 3
E7	RST 4
EF	RST 5
F7	RST 6
FF	RST 7
C8	RZ
9F	SBB A
98	SBB B
99	SBB C
9A	SBB D
9B	SBB E
9C	SBB H

HEX	MNEMONIC
9D	SBB L
9E	SBB M
DE	SBI 8-Bit
22	SHLD 16-Bit
30	SIM
F9	SPHL
32	STA 16-Bit
02	STAX B
12	STAX D
37	STC
97	SUB A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
96	SUB M
D6	SUI 8-Bit
EB	XCHG
AF	XRA A
A8	XRA B
A9	XRA C
AA	XRA D
AB	XRA E
AC	XRA H
AD	XRA L
AE	XRA M
EE	XRI 8-Bit
E3	XTHL

NOTE: 1-Byte Instructions: Operand R,M or implicit
2-Byte Instructions: Operand 8-Bit
3-Byte Instructions: Operand 16-Bit

APPENDIX-C

ASSEMBLER & DISASSEMBLER (OPTIONAL)

In **VMC-850X** there is an onboard facility of assembler/ disassembler. This mode will work in Serial Interface mode. The installation of Serial mode is given in Chapter-4. After installing the Serial Interface, the command for Assembler & Disassembler is shown below:

A=Assemble Mode
C=Disassemble Mode

ASSEMBLER MODE

On pressing the key 'A' VMC-850X comes in to the assembler mode. As soon as 'A' key is pressed, kit asks RAM address. This will be the starting address of the program to be entered. After entering the starting address, press <CR> key, it displays the entered starting address in the monitor. Now it waits for mnemonics entry.

One can enter all the valid mnemonics of 8085 and the Pseudo commands. If the entered alphabets do not form a valid mnemonic of a Pseudo command, the carriage goes to the same line and prints the address of the previous line. Hence the entry of the wrong mnemonic is indicated by giving the same line to the user.

Entry of a space completes one field of entry; and processing of that field is done immediately by the command. By field, we mean, mnemonic as one field, operand or operand label as another field.

On pressing 'A' key, VMC-850X waits for the RAM address.

Format

RAM ADR : STARTING ADDRESS

ADDRESS MNEMONICS <CR>

SINGLE BYTE INSTRUCTIONS

NNNN Mnemonic Registered One or RST No. (if needed) <CR>

If the mnemonic entered is a single byte instruction with single operand, then the command waits for operand entry. No message is given by the command in this field. The user enters the required operand which is normally a register name of the two registers. When single register or RST number operand is needed, the command accepts the entry preceded by block delimiter (a space). That means if the operand name is wrongly entered, one needs to enter the correct name and follow it by delimiter. This completes the entry of the operand. if the operand name or number is not a valid character (A,B,C,D,E,H,L,0 to 7) then a '*' is printed to convey the error. The command further waits for valid operand to be entered.

Even after an error in operand entry is conveyed the validity of mnemonic still holds. The '*' conveys an error in operand field only. If mnemonic is entered in this field by mistake and the last character of the mnemonic happens to be valid Register name (e.g. ADD, ADC, DAA, DAD, SUB, SBB, etc.) then the command will accept the last character as the valid register name and will process it. This is because the command takes only the last character for processing, not the total entry.

For MOV instructions, the operand entry is r1,r2. If any of register names is not a valid name, the '*' is printed and command waits for a new entry of the operands i.e. both the register name are required to be given. Here again the '*' does not cancel the mnemonic entry, it only cancels the operand entry.

Note: NNNN means address

TWO BYTE INSTRUCTIONS

NNNN Mnemonic data operand <CR> (for single operand instruction)

NNNN Mnemonic Register name data operand <CR> (for two operand instruction)

There are two types of two bytes instructions in 8085. The first requires only one operand, i.e. a data byte. The second types requires two operand, one is a register name, second is a data byte.

In the data operand field, the data byte is entered as two hex numbers. One can enter any number of hex, digits, the command takes only last two digits as the data. Hence errors can be corrected by re-entering the numbers. If by mistake, a character other than hex is entered, then the error is flagged by printing a '*' and command waits for a fresh data entry. The data entered before the mistake is fully rejected by the command.

Register name entry and address label entry is similar to as explained in the single bytes instructions.

THREE BYTE INSTRUCTION

NNNN Mnemonic O-Operand <CR>

In case of three byte instruction 'O' will be displayed, one has to enter two byte address/two byte data depending upon instruction.

Example

Program 1 given on Chapter-6 is entered by using assembler/ disassembler software as follows:

After entering in the Serial Interface mode, press 'A' key for assembler mode.

```
RAM ADR  2000 <CR>
2000 LXI  H  O-2050 <CR>        ; 3 byte instruction
2003 MOV  A , M <CR>            ; 1 byte instruction
2004 INX  H <CR>
2005 ADD  M <CR>
2007 INX  H <CR>
2008 MOV  M , A <CR>
2009 EF   RST5 <CR>
200A END <CR>                   ; end pseudo command is
                                  used to come out of the
                                  assembler
```


ERROR CODE

1) An error in entering **wrong opcode** will erase complete opcode and display the same memory locatin to continue.

Example

2050 C ALT <CR>

2050-

2) An error in entering **wrong operand** will display a '*'. Now one has to enter the complete operand.

Example

2020 MVI A,45 <CR> *-

DISASSEMBLER

'C' command disassembles the program as specified by the starting address and end address. Incase one wants to proceed further, press <CR> key, otherwise <Esc> key will exit from the disassembly mode. After entering in the Serial mode press 'C' key, VMC-850X comes into the disassembly mode.

Format

PROG ADR (STARTING ADDRESS) (ENDING ADDRESS) <CR>

LIST ADR <CR>

Example

To disassemble the previous program entered through assembler command.

```
PROG    ADR     2000    200A    <CR>
LIST    ADR     2000    <CR>
2000    21      50      20
LXI     * H     2050    *-
2003    7E
MOV     A,M             -
```

If you want to see further press <CR> key or Esc to exit from Disassembler mode.

APPENDIX-D

SYNTAX OF 8085 ASSEMBLER & DISASSEMBLER

The following is the syntax for using Single Line Assembler:

DATA TRANSFER GROUP		SYNTAX EXAMPLE	
MOV	R1,R2	MOV	A,B
MOV	R,M	MOV	A,M
MOV	M,R	MOV	M,A
MVI	R,DATA	MVI	AbXX
MVI	M,DATA	MVI	MbXX
LXI	RP,DATA16	LXI	HbO-XXXX
LDA	ADDR	LDA	O-XXXX
STA	ADDR	STA	O-XXXX
LHLD	ADDR	LHLD	O-XXXX
SHLD	ADDR	SHLD	O-XXXX
LDAX	RP	LDAX	B
STAX	RP	STAX	B
XCHG		XCHG	

ARITHMETIC GROUP			
ADD	R	ADD	B
ADD	M	ADD	M
ADI	DATA	ADI	XX
ADC	R	ADC	A
ADC	M	ADC	M
ACI	DATA	ACI	XX
SUB	R	SUB	C
SUB	M	SUB	M
SUI	DATA	SUI	XX
SBB	R	SBB	A
SBB	M	SBB	M
SBI	DATA	SBI	XX
INR	R	INR	B
INR	M	INR	M
DCR	R	DCR	C
DCR	M	DCR	M
INX	RP	INX	H
DCX	RP	DCX	B
DAD	RP	DAD	D
DAA		DAA	

LOGICAL	GROUP		
ANA	R	ANA	D
ANA	M	ANA	M
ANI	DATA	ANI	XX
XRA	R	XRA	D
XRA	M	XRA	M
XRI	DATA	XRI	XX
ORA	R	ORA	C
ORA	M	ORA	M
ORI	DATA	ORI	53
CMP	R	CMP	B
CMP	M	CMP	M
CPI	DATA	CPI	XX
RLC		RLC	
RRC		RRC	
RAL		RAL	
RAR		RAR	
CMA		CMA	
CMC		CMC	
STC		STC	

BRANCH GROUP			
JMP	ADDR	JMP	O-XXXX
JCONDITION	ADDR	JC	O-XXXX
CALL	ADDR	CALL	O-XXXX
CONDITION	ADDR	CC	O-XXXX
RET		RET	
RCONDITION		RC	
RST	N	RST	5
PCHL		PCHL	

STACK I/O & M/C CONTROL			
PUSH	RP	PUSH	B
PUSH	PSW	PUSH	PSW
POP	RP	POP	B
POP	PSW	POP	PSW
XTHL		XTHL	
SPHL		SPHL	
IN	PORT	IN	XX
OUT	PORT	OUT	XX
EI		EI	
DI		DI	
HLT		HLT	
NOP		NOP	

Note: *XX* = *Data*
XXXX = *16 bit Data/Address*

MAINTENANCE TIPS FOR VMC-850X

VMC-850X with 8/32K bytes of RAM and 8K ROM draws about 0.6 Amp. of current. However, with maximum on board expandability of 64K of memory and all buffer chips it draws about 1.5 Amps. But a safe value of 2.00 Amps is recommended in this case.

VMC-850X is supplied to the customer either just as a board or in a proper ABS PLASTIC cabinet.

On connection/switching on the power supply the system checks itself for the proper operation and displays a message '**- UP 85**'. If the display is blank on switching on the power supply, then check for the following:

a) Check the fuse of the Power Supply.

b) If the fuse is all right, then disconnect the +5V supply from the board and check for the presence of +5V at the wire terminal.

c) If the +5V is not there, then check backward to rectify the power supply.

d) On the other hand if the +5V is there at the wire terminal, then check for the following on the board of VMC-850X:

i) Shorting of the +5V Zener diode.

ii) Shorting of the 1000 UF capacitor on the left most side of the board.

iii) Shorting of the Vcc and ground on the board due to some metal burr. etc.

If the message - UP 85 is there on the display but the system does not accept any command check for the following:

i) Overloading of the power supply.

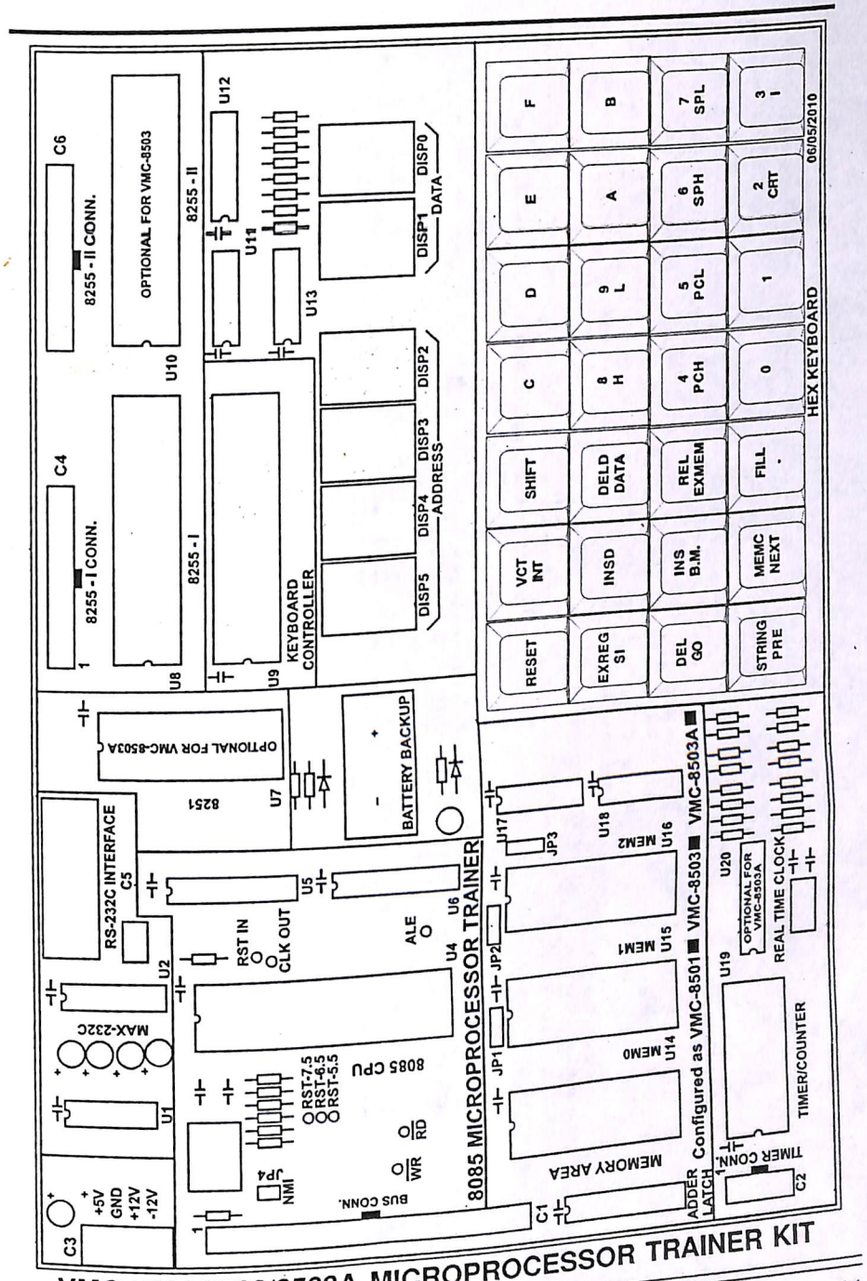
ii) Check that the supply voltage has not gone below 4.8V.

iii) Check for Vcc at the respective pins of all IC's.

If a fault other than mentioned above occurs, please contact our Service Engineer at the nearest service centre.

REFERENCES

1. B. Ram, "Fundamentals of Microprocessors and Microcomputers"; Dhanpat Rai Publications (P) Ltd., 2000.
2. A.P. Mathur, "An Introduction to Microprocessors";3rd Edition, Tata McGraw-Hill, 1989.
3. John P. Hayes, "Digital System Design and Microprocessors";McGraw-Hill, 1985.
4. R.S. Gaonker, "Microprocessor Architecture, Programming and Applications"; Wiley Eastern Ltd., 1986.
5. D.V. Hall, "Microprocessors and Digital System"; McGraw Hill, 1983.
6. M. Rafiquzzaman, "Microprocessors and Microcomputer Based System Design"; CRC Press, 1990.
7. Intel's Microporcessor and Peripherals Handbook, Volume 1 and II, 1989.
8. Intel Microprocessors Handbook, 1990.



VMC-8501/8503/8503A MICROPROCESSOR TRAINER KIT

Block Diagram for VMC-8501/8503/8503A

VMC-8502/8502AD MICROPROCESSOR TRAINER KIT
C3
+5V
GND
+12V
-12V
U1
MAX-232C
U2
RS-232C INTERFACE
C5
8251
OPTIONAL
U7
C4
8255 CONN.
C6
8155 CONN.
U8
8255
U10
8155
U9
KEYBOARD CONTROLLER
U11
U12
U13
JP4
NMI
RST IN
CLK OUT
RST-7.5
RST-6.5
RST-5.5
BUS CONN.
8085 CPU
U4
U5
U6
ALE
WR
RD
BATTERY BACKUP
DISP5
DISP4
DISP3
DISP2
ADDRESS
DISP1
DISP0
DATA
8085 MICROPROCESSOR TRAINER
JP1
JP2
JP3
U17
U18
C1
MEMORY AREA
MEM0
MEM1
MEM2
U14
U15
U16
ADDER LATCH
Configured as VMC-8502 VMC-8502AD
RESET
VCT INT
SHIFT
C
D
E
F
EXREG SI
INSD
DELD DATA
8 H
9 L
A
B
DEL GO
INS B.M.
REL EXMEM
4 PCH
5 PCL
6 SPH
7 SPL
STRING PRE
MEMC NEXT
FILL .
0
1
2 CRT
3 I
HEX KEYBOARD
07/05/2010
C2
TIMER CONN.
U19
U20
OPTIONAL
REAL TIME CLOCK
TIMER/COUNTER