USER'S MANUAL

FOR

VMC-8603

8086/88 BASED

MICROPROCESSOR TRAINING

CUM-DEVELOPMENT KIT

(NEW MODEL - 1.02)

Table of Contents

Table of Contents

Table of Contents

1

SYSTEM INTRODUCTION

GENERAL DESCRIPTION

VMC-8603 is a single board MICROPROCESSOR TRAININGIDEVELOPMENT KIT configured around the INTEL's 16 bit Microprocessor 8086. This kit can be used to train engineers, to control any industrial process and to develop software for 8086 systems.

The kit has been designed to operate in the max. mode. Co-processor 8087 and 110 Processor 8089 can be added on board. 8086 CPU can also be replaced by 8088 CPU.

The Kit communicates with the outside world through a keyboard having 28 keys and eight seven segment displays.

VMC-8603 is packed up with powerful monitor in 16K Bytes of factory programmed EPROMS and 16K Bytes of ReadlWrite Memory. The total memory on the board can be easily expanded to 256K Bytes of EPROM and 128K Byte of CMOS RAM. The system has 72 programmable I/O lines. The serial I/O Communication is made possible through 8251.

For control applications, three 16 bit Timer/Counters are available through 8253. VMC-8603 provides onboard battery back up for onboard RAM. This saves the user's program in case of power failure.

VMC-8603 provides an onboard EPROM Programmer which enables the user to burn his program in any of the eproms 2764/27128/27256. The onboard resident system monitor software is very powerful. It provides various software commands like BLOCK MOVE, INSERT, DELETE, FILL etc. which are helpful in debugging/developing software. An onboard line assembler (optional) is also provided on VMC-8603

SYSTEM SPECIFICATIONS

PROCESSORS

Central Processor	:	8086, 16 bit Microprocessor operating in max. mode or 8088 8 bit Microprocessor.
Co-Processor	:	8087 Numeric Data Processor.
110	:	8089 I/O Processor.

MEMORY

EPROM	:	16K Bytes of EPROM Loaded with monitor expandable to 1024 Bytes using 27256.
RAM	:	16K bytes of CMOS RAM expandable to 256K Bytes using 6264/62256.

INPUT/OUTPUT

Parallel	:	24 I/O lines expandable to 72 lines (3 nos. of 8255A).
Serial	:	EIA RS-232-C (Main).
TIMER/COUNTER	:	Three 16 bit Timer/Counter through 8253.
Other Interfaces	:	EPROM PROGRAMMER for 2764/27128/27256.
Keyboard & Display	:	28 keys and 8 Seven Segment display.
BUS	:	All address, data and control signals (TIL Compatible) available at edge connector as per Multi Bus. The kit also has its own Resident Bus.
Physical Size	:	9.5" x 12.5".
Power Supply	:	5V, 2.5 Amps for kit, +_12V, 250mA for CRT & +24/21V for EPROM Programmer
Operating Temp.	:	0 to 50·C.

SYSTEM CAPABILITIES

KEYBOARD MODE

1. Examine/Modify the memory byte locations.
2. Examine/Modify the memory word locations.
3. Examine/Modify the contents of any of internal register of 8086.
4. Move a block of Data/Program 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. To Write/Read directly to/from the I/O Port.
8. Fill a particular memory area with a constant.
9. Check the contents of an EPROM for blank (Blank Check).
10. List the Contents of an EPROM in to RAM area.
11. Verify the contents of an EPROM with any memory area.
12. Program/Duplicate an EPROM.
13. To execute the program in full clock speed.
14. To execute one program instruction at a time.

SERIAL MODE

1, Display/Modify memory location.
2. Display/Modify internal registers of 8086.
3. Display Block of memory data.
4. Move a block of Data/Program from one location to another location.
5. Execute the program in full clock speed mode.
6. To execute one program instruction at a time.
7. Port Input
8. Port Output
9. Read Hex file
10. Write Hex file

SYSTEM INSTAIIATION

2

KEYBOARD MODE

To install VMC-8603 in keyboard mode, the following additional things are required.

1. Power Supply (+5V/2.5 Amp.)

INSTALLATION:

1. Select the memory chips (EPROM and RAM) as per your requirement by changing the jumper connections as mentioned in Chapter - 3 under the heading Memory. If no change is to be done, go ahead.
2. Connect the Power Supply wires (GND & +5V) of the supply conneetor marked for various voltage.
3. Switch on the Power Supply.
4. A message - UP 86 will come on display (PRESS RESET if you do not get - UP 86) and can now use the system.

SERIAL MODE

To install VMC-8603 in serial mode, the following additional things are required.

1. Power Supply +5V12.5 Amp., +_ 12V/250mA
2. CRT Terminal with RS-232-C serial interface.
3. SERIAL CABLE from kit to Terminal

INSTALLATION:

Steps 1 and 2 are same as mentioned earlier in keyboard mode.

3. Connect the Power Supply wires (GND, +5V, +12V and -12V) of the kit supply connector marked for various voltages.
4. Connect the cable from the CRT Terminal to the Main RS-232-C interface of the kit marked as J3 Connector.
5. Set the Baud rate of the kit to the desired value by changing the jumper position as mentioned in Chapter - 5.
6. Switch on VMC-8603 and Press Reset.
7. Switch ON the terminal and set the same baud rate in the terminal as selected for VMC-8603.

8. Press CRT Key on VMC-8603. A message Serial will be displayed on the kit and a message VMC-8603 version 1.2 will come on the screen and in the next line it displays a prompt character '.', indicating that now the command can be entered from the terminal.. The detail about the connector J3 are given in Chapter - 5.

One can now give the commands as mentioned in Chapter-6 under the heading "Serial I/O Commands".

3

HARDWARE DESCRIPTION

CPU

8086 is a 16 bit, third generation microprocessor and is suitable for an exceptionally wide spectrum of microcomputer applications. This flexibility is one of most outstanding characteristics.

8086 has got 16 data lines and 20 address lines. The lower 16 address lines are mUltiplexed with 16 data lines. Hence it becomes necessary to latch the address lines. This is done by using 74 LS 373. In fact several of the 40 CPU pins have dual functions that are selected by a strapping pin. In this kit 8086 is used in the max. mode (MN/Mx input held logically low).

The 8088 is designed with an 8-bit external path to memory and *110.* Except that the 8086 can transfer 16 bits at a time, the two processors & software are identical in almost every respect. Software identical in almost every respect. Software written for one CPU will execute on the other without alteration. The two processors are designed to operate with the 8089 I/O processors and other processors in multiprocessing and distributed processing systems.

The INTR, TEST & Hold Inputs to 8086 are pulled down and are brought out at PCB FRC connector. The 8086's NMI Input is connected to the VCT INT Key.

The maskable interrupt INTR is available to the peripheral circuits through the expansion Bus. To use the maskable interrupt an interrupt vector pointer must be provided on the data bus when INTA is active. An interrupt Controller Circuit is provided to take care of more than one source of interrupt.

CO-PROCESSOR 8087

The 8087 Co-processor "hooks" have been designed into the 8086 and 8088 so that this types of processor can be accommodated in the future. A co-processor differs from an independent processor in that it obtains its instructions from another processor, called a host. The co-processor monitors instructions fetched by the host and recognizes certain of these as its own and executes them. A co-processor, in effect, extends the instruction set of its host computer.

I/O PROCESSOR 8089

The 8086 and 8088 are designed to be used with the 8089 in high performance 1/O applications. The 8089 inconceptually resembles a microprocessor with two DMA channels and an instruction set specifically tailored for I/O operations. Unlike simple DMA controllers, the 8089 can service I/O devices directly, removing this task from the CPU. In addition, it can transfer data on its own bus or on

the system bus, can match 8-bit or 16-bit peripherals to 8-bit or 16-bit buses, and can transfer data from memory to memory and from I/O devices to I/O device. 8089 has been used here in local mode.

CLOCK GENERATION

The clock generator circuit is an Intel's 8284 clock generator/driver. The circuit accepts a crystal input which operates at a fundamental frequency of 14.7456 MHz. (14.7456 MHz

was selected since this frequency is a multiple of the baud rate clock and also provides a suitable frequency for the CPU). The clock generator/driver divides the crystal frequency by three to produce the 4.9 MHz ClK signal required by the CPU. Additionally, the clock generator performs a further divide-by-two output called PCLK (peripheral clock) which is the primary clock signal used by the remainder of the circuits.

The clock generator/driver provides two control signal outputs which are synchronized (internally) to the 4.9 MHz ClK signal; RDY (ready) and RST (reset). RST is used to reset the VMC-8603 to an initialized state that occurs when the RES input goes low (when power first is applied or when the SYSTEM RESET key is pressed).

The system can operate at either 4.9 MHz or 2.45 MHz. This is selected by a set of jumpers JP1 on the right hand side of the 8284 clock generator as shown below:

filil	1	4.9 MHz
~	2	2.45 MHz (lOWER)

The VMC-8603 is suppled in 4.9 MHz configuration.

BUS CONTROLLER

The 8288 is a Bus Controller which decodes status signals output by an 8089, or a maximum mode 8086 or 8088. When these signals indicate that the processor is to run a bus cycle, the 8288 issues a bus command that identifies the bus cycle as memory read, memory write, I/O read, I/O write, etc. It also provides a signal that strobes the address into latches. The 8288 provides the drive level needed for the bus control lines in medium to large systems.

BUS ARBITERS

The 8289 is a Bus Arbiter that controls the access of a processor to a multimaster system resources (typically memory) that is shared by two or more microprocessors (masters). Arbiters for each master may use one of several priority-resolving techniques to ensure that only one master drives the shared Bus.

MEMORY

VMC-8603 provides 16K Bytes of EPROM loaded with monitor and 16K bytes of CMOS RAM. The total onboard memory can be configured as follows:

EPROM - 512 bytes of EPROM using 27C010, 1024 bytes using 27C020 & 256K bytes using 62C1024.

RAM - 64K Bytes of RAM using '62256.

The system provides four *28/32* Pin sockets for the EPROM area named as MEM1 to MEM3 and four *28/32* Pin sockets for the RAM area named as MEM2 to MEM4. EPROMO to EPROM3 can be defined to have either of the EPROM *27128/27256/27512/27C010/27C020* and the MEM2 to MEM4 can be defined to have either of 6264/62256/62C1024. This selection is done by changing the jumper connections on the board of the kit.

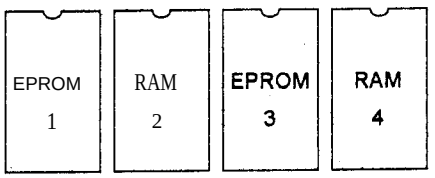
With the 20 bit address of 8086, a total of 1 Mega Bytes of memory can be addressed with the address slot as 00000 to FFFFF. The actual memory area will depend upon the memory chips selected and accordingly the addresses of the various memory sockets will change.

On VMC-8603 the 16K Bytes of RAM area is provided the bottom most slot and the 16K byte of EPROM area is provided the top most slot.

SELECTION OF MEMORY CHIPS

In VMC-8603 there are four memory sockets namely MEM1 to MEM4. Memory sockets MEM1 & MEM3 are meant for EPROMS and sockets MEM2,MEM3,MEM6 & MEM7 are meant for RAM area. Sockets 1& 3 can be defined to have either of 27128/27256/27512 EPROM in a continuous mapping. Same way sockets 2 & 4 can be defined to have either of 6264/62256/621024 RAM in a continuous mapping. This selection of Chips of different type and capacity in done by selecting a suitable jumper.

The RAM area in the kit has got the bottom most slot of the 1M which is addressable and the EPROM area has got the top most slot. The various sockets position is shown here:



The following are jumpers selection and the memory mapping in different combinations.

1) If all EPROM sockets are to be defined for 27128:

JP7	JP9
SHORT 1A &'16	SHORT 1 & 2

The memory mapping in this case for EPROM Area will be as given below:

EPROM1 - F8DDD to FFFFE
EPROM3 - F8DD1 to FFFFF

2) If all EPROM sockets are to be defined as 27256:

JP7	JP9
SHORT 1A & 16	SHORT 1 to 2

The memory mapping in this case for EPROM area will be as given below:

EPROM1 - FDDDD to FFFFE
EPROM3 - FDDD1 to FFFFF

3) If all the EPROM sockets are to be defined as 27512:

JP7	JP9
SHORT 1A & 16	SHORT 2 & 3

The memory mapping in this case for EPROM area will be as given below:

EPROM1 - EDODD to FFFFE
EPROM3 - EOOD1 to FFFFF

4) If all the EPROM sockets are to be defined as 27CD1 D:

JP7	JP9
SHORT 1A S 2A	SHORT 2 & 3

The memory mapping in this case for EPROM area will be as given below:

EPROM1 - COOOO to FFFFE
EPROM3 - C0001 to FFFFF

5) If all the EPROM sockets are to be defined as 27C020:

JP7	JP9
SHORT 1A & 2A	SHORT 2& 3

The memory mapping in this case for EPROM area will be as given below:

EPROM1 - 80000 to FFFFE
EPROM3 - 80001 to FFFFF

6) When the two memory sockets 2 & 4 are to be defined for 6264:

JP3	JP4
SHORT 16 & 26	SHORT 46 & 4H

The memory mapping for this case will be:

MEM2 - 00000 to 03FFE
MEM4 - 00001 to 03FFF

7) When the two memory sockets 2& 4 are to be defined for 62256

JP3	JP4
SHORT 16 & 1H	SHORT 46 & 4H

The memory mapping for this case will be:

RAMO - 00000 to OFFFE
RAM1 - 00001 to OFFFF

8) When the memory sockets are to be defined for 62C1024:

JP3	JP5
SHORT 16 & 1H	SHORT 76 & 66

, The memory mapping for this case will be:

RAMO	-	00000	to	OFFFE
RAM1	-	00001	to	OFFFF

110 DEVICES

8279

8279 is a general purpose programmable keyboard and display I/O interface device designed for use with the 8086 microprocessor, It provides a scanned interface to 28 contact key matrix provided in VMC-8603 and scanned 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

8255 is a programmable peripheral interface (PPI) designed to use with 8086 Microprocessor. This basically acts as a general purpose I/O component to interface peripheral equipments to the system bus. It is not necessary to have an external logic 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 and 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 Port-C lower can be defined using the appropriate software commands, The Port addresses for these ports are given here. VMC-8603 provides nine Input/Output ports of 8 lines each using three 8255 chips, These ports are brought out at connectors C10, C9, C11.

8253

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. Event Counter, Binary rate rnultipller, real time clock etc. This chip has got three independent 16 bit counters each having a count rate of up to 2 MHz, The CLK, GATE & OUT signals of these timers are brought out at the C5 connector.

8251

This chip is a programmable communication interface and is used as a peripheral device. This device accepts data characters from the CPU in parallel form and then converts them into a continuous serial data stream for transmission. Simultaneously it can receive serial data stream and converts them into parallel

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 for the CPU. The CPU can read the complete status of it at any time. 8251 has been utilized in VMC-8603 for Main/Aux. RS-232-C interface and 20mA current loop.

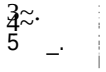
DISPLAY

VMC-8603 provides eight digits of seven segment display. Four digits are for displaying the address of any location or name or any register, whereas the rest of the four digits are meant for displaying the contents of memory location or of a register. All the eight digits of the display are in hexadecimal notation.

BATTERY BACK UP

The VMC-8603 provides a battery back up for the onboard RAM area. The battery back up circuit is based around LM-393. The Vcc points of all the sockets for the RAM area l.e, MEM2 to MEM4 are joined together to the common point, If the RAM area is not to be backed up by battery this point should be shorted to the +5V pt. otherwise to CMOS Vee as shown below:

28 Pin (For Battery)



G H

JP4

28 Pin (For VCC)

G H

JP4

32 Pin (For Battery)

G H

JP5

32 Pin (For VCC)

G H

JP5

PORT ADDRESS

The Port addresses of the various I/O devices used in VMC-8603 are given below:

DEVICE NAME	PORT NAME	PORT ADDRESS
8255-1	PORT A	FFF8
	PORT B	FFFA
	PORT C	FFFC
	CONTROL WORD	FFFE
8255-2	PORT A	FFEO
	PORT B	FFE2
	PORT C	FFE4
	CONTROL WORD	FFE6
8255-3	PORT A	FFF9
	PORT B	FFFB
	PORT C	FFFD
	CONTROL WORD	FFFF
8279	DATA WORD	FFE8 OR FFEC
	COMMAND WORD	FFEA OR FFEE
8253	COUNTER 0	FFD8
	COUNTER 1	FFDA
	COUNTER 2	FFDC
	CONTROL WORD	FFDE
8251 (Main)	DATA WORD	FFFO OR FFF4
	COMMAND WORD	FFF2 OR FFF6
8259 (Only in VMC-8603A)	DATA WORD	FFC8 OR FFCC
	COMMAND WORD	FFCA OR FFCE
8089	CHANNEL 1	FFCO
	CHANNEL 2	FFC1

KEYBOARD DESCRIPTION

4

KEYBOARD DESCRIPTION

The VMC-8603 has 28 keys and eight seven segment displays to communicate with outside world. As the power is turned on and

Reset key is pressed, a message -up 86 is displayed on the display and all the keys are in command mode. The keyboard is shown below:

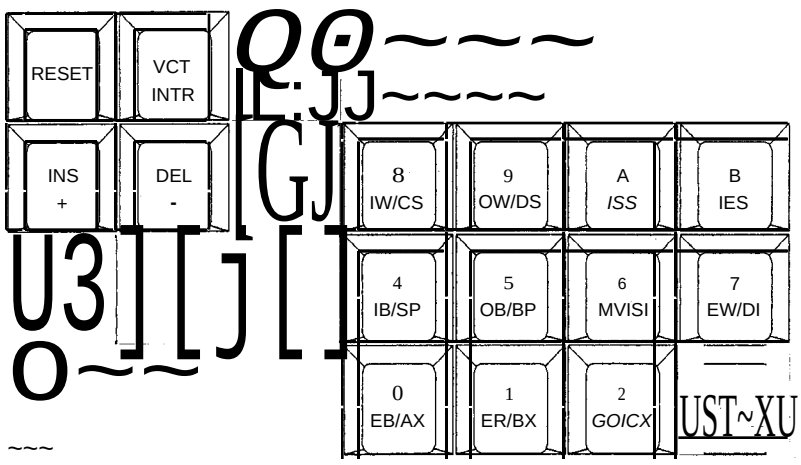


FIG - 4.1

With the keypad monitor program, you enter both commands and data by pressing individual keys of the keypad. (The Monitor communicates with you through the display). As shown on Figure-4.1, the Keypad is divided into two logical gr()lp~_l~ H3 hexadeqimclj 1s.~~~mJh~hLb.and-!icje!1nd thetwelveJLlnc::tio.n • keys on the left hand sid~:_

Most of the hexadecimal keys have combined functions as noted by their individual legends. The small letters appearing under the hexadecimal key values are acronyms for individual monitor commands and 8086 register names. Acronyms to the left of the slash sign are monitor commands, and acronyms to the right of the §.lash §ign!ire 8086 register n~lD_~_ş.The function of a hex~de6fin_al -keyst anyone time is dependent" on the current state of the monitor and what the monitor i~_expectinga.s..l,nput. Table 4.-::1tjefines both the commands and registers associated with the hexadecimal keYs.

TABLE 4.1 HEXADECIMAL KEYPAD LEGEND INTERPRETATION

Hexadecimal Key	Acronym	Command Name	Acronym	Register Name
0 EBIAX	EB	Examine Byte	AX	Accumulator
1 ERlBX	ER	Examine Register	BX	Base
2 GO/CX	GO	GO	CX	Count
3 ST/DX	ST	(Single) step	OX	Data
4 IBISP	18	Input Byte	SP	Stack Pointer
5 OBlBP	08	Output Byte	BP	Base Pointer
6 MV/SI	MV	Move	SI	Source Index
7 EW/DI	EW	Examine Word	01	Destination Index
8 IW/CS	IW	Input Word	CS	Code Segment
9 OW/DS	OW	Output Word	DS	Data Segment
A ISS	—	—	SS	Stack Segment
B IES	—	—	ES	Extra Segment
C BC/IP	BC	Blank Check	IP	Instruction Pointer
D LS/FL	LS	Listing	FL	Flag
E VRJ	VR	Verify	None	N/A
F PRGI	PRG	Program	None	N/A

The individual operation of the twelve function keys is defined in Table - 4.2.

TABLE·4.2 FUNCTION KEY OPERATION

Function Key	Operation
RESET	The SYSTEM RESET key allows you to terminate any present activity and to return your VMC~8603 to an initialized state. When pressed, the 8086 sign-on message appears in the display and the monitor is ready for command entry.
VCT INTR	The INTR (Interrupt) key is used to generate an immediate non/maskable type 2 interrupt (NMI). The NMI interrupt vector is initialized on power up or system reset to point to a routine within the monitor which causes all of the 8086's registers to be saved. Control is returned to the monitor for subsequent command entry.
INS +	INSERT Keys allows insertion of one or more Bytes in the user's program. The + (pius) key allows you to add two hexadecimal valves. This function simplifies relative addressing by allowing you to readily calculate an address location relative to a base address.
DEL -	DELETE key allows deletion of one or more Bytes from the user's program. The - (minus) key allows you to subtract one hexadecimal value from another.
REG	The REG (Register) key allows you to use the contents of any of the 8086's registers as an address or data entry.
CRT NEXT	CRT Key is used for entering into CRT mode. NEXT key is used to separate keypad entries and to increment the address field to the next consecutive memory location.
TIY	TTY key is used for entering into TTY interface. The (Period) key is the command terminator. When pressed, the current command is executed. Note that when using the GO command, the 8086 begins program execution at the address specified when the '0' key is pressed.
FILL PRV	FILL key allows you to Fill any Block of RAM area with a constant. PRV (Previous) key is used to decrement the address field to previous memory location.

Function Key	Operation
F1	User definable key.
F2	User definable key.
F3	User definable key.

Your VMC-8603 Kit uses the eight digit display to communicate with you. Depending on the current state of the monitor, the information displayed will be the:

- ~ Current contents of a register or memory location.
- ~ An "echo" of a hexadecimal key entry.
- ~ A monitor prompt sign.
- ~ An information of status message.

The display itself is divided into two groups of four characters. The group on the left is referred to as the "address field", and the group on the right is referred to as the "data field". All values displayed are in hexadecimal and follow the format shown in Table - 4.3.

TABLE - 4.3 HEXADECIMAL DISPLAY CHARACTERS

Hexadecimal Value	Display Format	Hexadecimal Value	Display Format
0	0	8	8
1	1	9	9
2	2	A	A
3	3	B	b
4	4	C	C
5	5	D	d
6	6	E	E
7	7	F	F

GENERAL OPERATION

When using the keypad monitor, you will be prompted through the display as to the input required. Whenever the monitor is expecting a command entry, a minus sign or dash appears in the most significant display digit of the address field. Pressing one of the hexadecimal command keys (Keys O-F) when the dash is displayed is interpreted as a command entry. When the key is pressed, the dash disappears and a decimal point (or decimal points) appears in the least significant display digit (or least significant digits) of the address field to indicate that subsequent keypad entry will be directed to the address field. Note that depending on the command, characters also may appear within the address and data fields. Monitor operation from this point is determined by the actual command entered. The description of each command is given later on in this chapter.

Following power on or whenever the system RESET key is pressed, the monitor initializes the VMC-8603and displays, the monitor sign-on message (UP in the two least signifi'Cantdigits of the data field) and the command prompt character (dash) in the most significant digit of the address field. Note that in the following command descriptions, the monitor always calculates a 20-bit physical memory .eddresa from a 16 bit segment address value and a 16 bit offset adqle.!>s',~!ue. The segment address value is entered first, the ":"keyis pressed (to separate the two entries), and then the offset address value js anterelL"The two values entered, which are displaced from one another by four bits, are added together to form the 20-bit physical memory address. If only one address value is enterecj (the colon must be omitted), it is interpreted by the rrionlfor as an offset' address value enter, and the current contents 0.1 the codesegment(CS) register are' used as the segment address value. The CS register contents and offset address value entered are combined to form the 20-bit physical address...

COMMANDDEsc~pnON

The various commands that can be executed by the monitor are listed below:

- EXAMINE BYTE
- EXAMINE WORD
- EXAMINE REGISTER
- INPUT BYTE
- INPUT WORD
- OUTPUT BYTE
- OUTPUT WORD
- GO
- MOVE
- STEP

- INSERT
- DELETE
- FILL
- BLANK CHECK
- VERIFY
- LIST
- PROGRAM/DUPLICATE

During the description of the commands the following notations are used:

@	indicate a keyboard key
[A]	indicate that" A" Is optional
[A].0'	indicate one or more optional occurrences of" A"
<B>	indicate that "B" is a variable

EXAMINE BYTE/EXAMINE WORD

Function

The Examine Byte (EB) and Examine Word(EW) commands examine the contents of selected memory locations. If the memory location can be modified (e.g., a location in RAM), the contents additionally can be updated.

Syntax

@	<addr>	NEXT	[[	<data>	]	NEXT	.	[.J
~	<addr>	NEXT	[[	<data>	]	NEXT	.	[.J

Operation

To use either command, press the EB key (Examine byte) or EW Key (Examine Word) when the command prompt character (-) is displayed. When either key is pressed, the decimal point at the right of the address field will light (the rest of the display will be blanked) to indicate the entry from the keyboard will be directed to the address field. From the keypad, enter the memory address of the byte or word to be examined, most significant character first. Note that all memory addresses consist of both a segment value and an offset value. When no segment value is specified, the default segment value is the current contents of the code segment (CS) register. When a segment value is specified, the first address entry is the segment value, a colon (:) is entered as a separator, and the second address entry is the offset value. The capacity of an address field entry is limited to four characters and, if more than four characters are entered for either a segment or offset value, only the last four characters entered (the four characters, currently displayed) are valid. After the address is entered, press the 'NEXT key.

The data byte or word contents of the addressed memory location will be displayed in the data field, and a decimal point will appear at the right of the data field to indicate that any subsequent hexadecimal keypad entry will be directed to the data field. Note that when using the Examine Word Command, the byte contents of the memory location displayed appear in the two least significant digits of the data field, and the byte contents of the next consecutive memory location (memory address + 1) appear in the two most-significant digits of the data field.

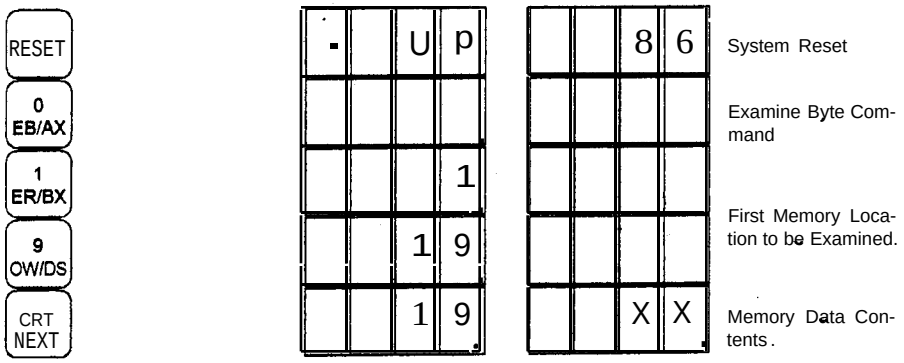
If the contents of the memory location addressed only are to be examined press the "key to examine the next consecutive memory location (Examine Byte command) or the next two consecutive memory locations (Examine Word Command). One can also press "PRV" key to examine the previous memory location/locations. To modify the contents of an address memory location, key-in the new data field is limited to either two (examine byte) or four (examine word) characters and that if more characters are entered, only the characters currently displayed in the field are valid. The data displayed is not updated in memory until either the ". or "NEXT" key is pressed. If the". key is pressed, the command is terminated, and the command prompt character is displayed in the address field. If the" NEXT" key is pressed, the offset address and data contents of the next consecutive memory locations (Examine Word Command) are displayed.

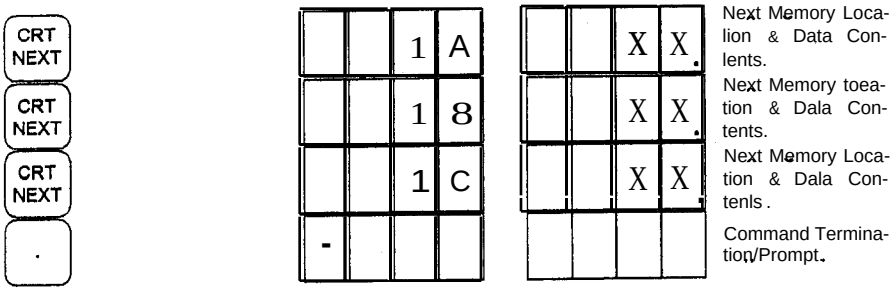
Error Conditions

Attempting to modify a non-existent or read-only (e.g. ROM or PROM) memory location. Note that the error is not detected until the "NEXT" or "." or "PRV" Key is pressed. When an error is detected, the characters "Err" are displayed with the command prompt character in the address field.

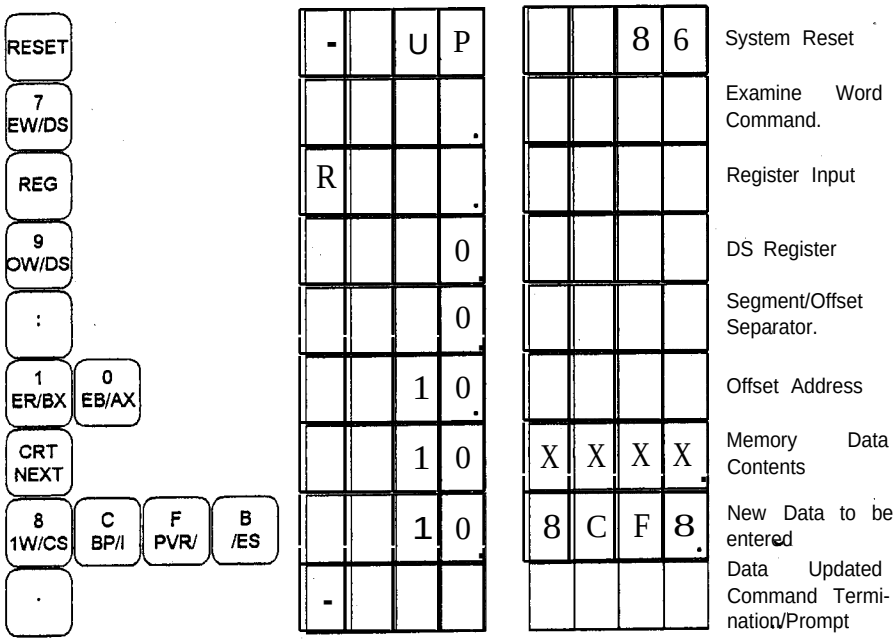
Examples

Example 1: Examine a Series of Memory Byte Locations Relative to the CS Register.





Example 2: Examining & Modifying Memory Word Location 10H relative to the OS Register.



To check that the data was updated successfully, press the EW key and enter the memory address (OS: IOH). Press the "NEXT" key and note that "8CFb" is displayed in the data field.

EXAMINE REGISTER

Function

The Examine Register (ER) command is used to examine and, if desired, to modify the contents of any of the 8086's registers.

Syntax

@ <reg key> [[<data>] NEXT * [0]

Operation

To examine the contents of a register. press the ER key when prompted for a command entry. When the key is pressed. the decimal point at the right of the address field will light. Unlike the Examine Byte Command. subsequent hexadecimal keypad entry will be interpreted as the register name (the acronym to the right of the slash sign on the key face) rather than its hexadecimal value. When the hexadecimal key is pressed, the corresponding register abbreviation will be displayed in the address field. the 16-bit contents of the register will be displayed in the data field and the decimal point on the right of the data field will light. Table 4-4 defines the 8086 register name. the hexadecimal keypad acronym. the display abbreviation and the sequence in which the registers are examined.

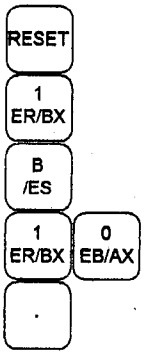
TABLE - 4.4 8086 REGISTERS

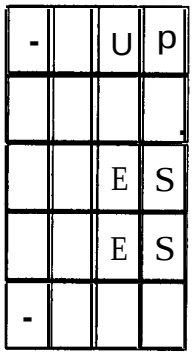
Register Name	Keypad Acronym	Display Abbreviation
Accumulator	AX	A
Base	eX	B
Count	CX	C
Data	OX	0
Stack Pointer	SP	SP
Base Pointer	BP	bp
Source Index	SI	SI
Destination Index	01	dl
Code Segment	CS	CS
Data Segment	OS	dS
Stack Segment	SS	S8
Extra Segment	ES	ES
Instruction Pointer	IP	IP
Flag	FL	FL

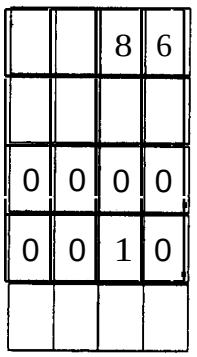
When the register contents are displayed (when the decimal point on the right of the data field lights), the register's contents can be modified or the register is said to be "open" for input. Keying in a value from the hexadecimal keypad will be echoed in the display's data field. and the register contents will be updated with the data value displayed when either the "NEXT" Key when the flag (FL) register is displayed will terminate the examine register command and return to the command prompt character.

Examples

Example - 1: Examining and Modifying a Register.







System Reset

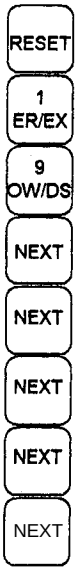
Examine Register Command.

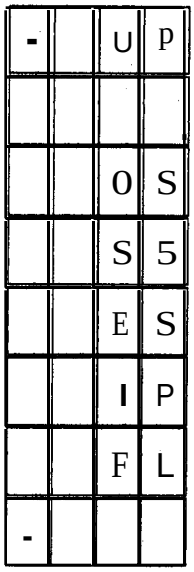
Exira Segment Register Contents.

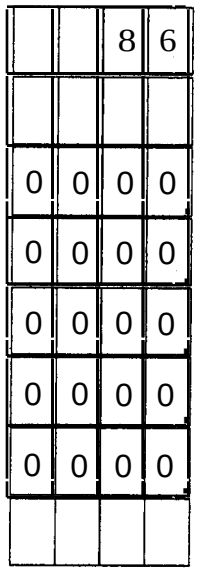
New Register Contents.

Register Updated Command Termination/Prompt..

Example-2: Examining a Series of Registers.







System Reset

Examine Register Command.

Data Segment Register Contents.

Stack Segment Register Contents.

Exira Segment Register Contents.

Instruction Pointer Register Contents

Flat Register Contents

Command Termination/Prompt

INPUT BYTE AND INPUT WORD

Function

The Input Byte (IB) and Input Word (IW) commands are used to input (accept) an 8-bit byte or 16-bit word from an input port.

Syntax

@ <port addr> (NEXT) [(NEXT)] . O

@ <port addr> (NEXT) [(NEXT)] " O

Operation

To use either the Input Byte or Input Word command press the corresponding hexadecimal key when prompted for command entry. When either the IB or IW key is pressed, the decimal point on the right of the address field will light to indicate that a port address entry is required. Using the hexadecimal keypad, enter the port address of the port to be read. Note that since I/O addressing is limited to 64 K (maximum address FFFFH), no segment value is permitted with the port address.

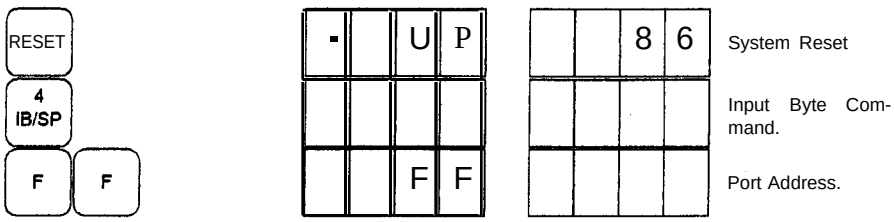
After the port address has been entered, press the "NEXT" key. The input byte or word at the addressed port will be displayed in the data field. Again pressing the "NEXT" key updates the data field display with the current data byte word at the addressed input port. Pressing the "." key terminates the command and prompts for command entry.

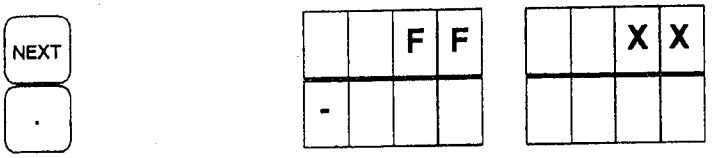
The VMC-8603 includes three 8255A parallel I/O interface which can be used with the Input Bytes and Input Word commands to input data from peripheral devices. Each 8255, in turn, consists of three individual 8-bit ports which are designated port A, Port B, and Port C. Each port operates independently during byte operations. During word operations, a pair of ports (I.e. a port of I and II 8255 operate together to form the 16-bit wide data word with lind 8255 corresponding to the low order byte. The IIIrd 8255 can be used only for byte operation and cannot be addressed in pair with some other port.

The Parallel I/O port circuits are programmed for input on power-up or whenever the system RESET key is pressed. If the circuit(s) previously has been programmed for output press the system RESET key (before pressing the command key) or, referring to the next section, output the appropriate byte or word value to the circuit's control port to program the port(s) for input.

Example

Example - 1 Single Byte Input from Port OFFH*.



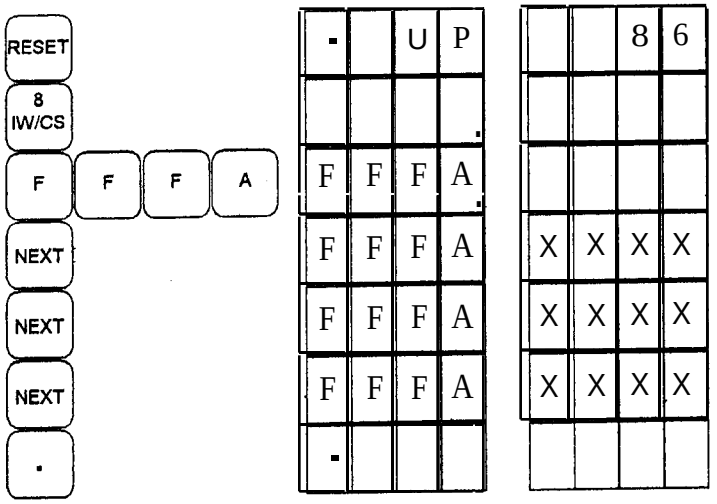


Input Data Byte.

Command Termination/Prompt.

* Port OFFH is not provided on the VMC-8603.

Example-2: Multiple Word Input from Parallel I/O Ports 18 and 28.



System Reset (Initializes Ports for Input).

Input Word Command.

Port B Address.

Input Data Word From Port.

Input Data Word Again.

Input Data Word Again.

Command Termination/Prompt.

OUTPUT BYTE AND OUTPUT WORD

Function

The Output Byte (OB) and Output Word (OW) commands are used to output a byte or word to an output port.

Syntax

(08) <port addr> (NEXT) <data> [(NEXT) <data>] * O

(OW) <port addr> (NEXT) <data> [(NEXT) <data>] * O

Operation

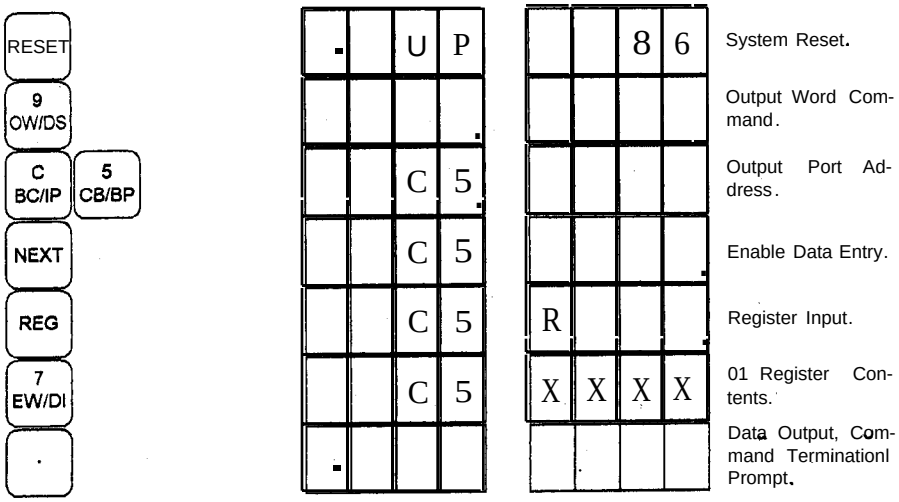
To use either command, press the corresponding hexadecimal key when prompted for command entry. When either the 08 or OW key is pressed, the decimal point on the right of the address field will light to indicate that a port address entry is required. Like the Input 8yte and Input Word commands, I/O addresses are limited to 64K, and no segment value is permitted. After the port address is entered, press the "NEXT" key. The decimal point on the right of the data field

will light to indicate that the data byte or word to be output now can be entered. Using the keypad, enter the byte or word to be output. After the data is entered press the "." key to output the byte or word to the port and to terminate the command or press the "NEXT" key if additional data is to be output to the addressed port.

As mentioned In the previous section, the Output Byte and Output Word commands can be used to program the 8255A parallel *1/0* port circuits for input or output as well as to output data to the individual ports. The *1/0* port circuits are programmed for input on power up or system reset and consequently first must be programmed for output (by outputting the appropriate data byte or word to the circuit's control port) before data can be output to the associated ports..

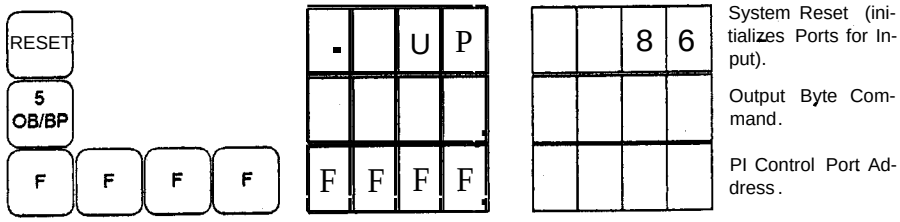
Examples

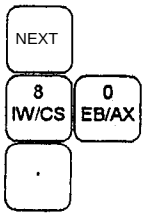
Example-1: Output 01 Register contents to Output Port OC5H*

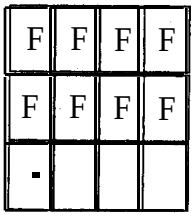


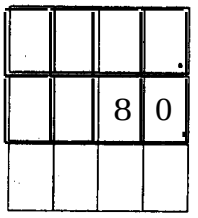
* Port OC5H is not provided on the VMC-8603.

Example-2: Programming 8255-1 for Output (the Control Word in FFFF).









Enable Data Entry.

Control Byte for Output.

Control Byte Output, Command Termination/Prompt.

GO

Function

The Go (Go) command is used to transfer control of the 8086 from the keypad monitor program to a user's program in memory.

Syntax

(GO) [<addr>] [(NEXT) <breakpoint>] O

Operation

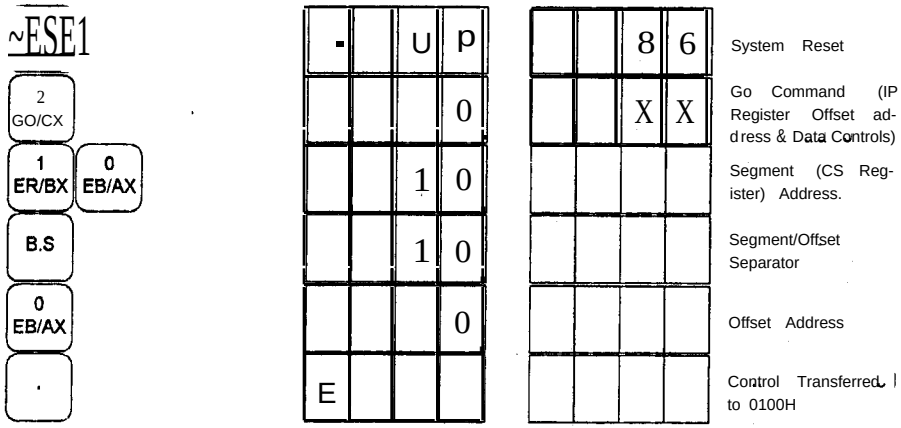
To use the GO command, press the GO key when prompted for command entry. When the key is pressed, the current IP (Instruction Pointer) register contents are displayed in the address field, the byte contents of the memory location addressed by the IP register are displayed in the data field and the decimal point at the right of the address entry can be entered. If an alternate starting address is required, enter the address from the keypad. (when an address is entered, the data field is blanked). To begin program execution (at the current instruction or alternate program address, press the "." key. When the key is pressed, the monitor displays an "E" in the most significant digit of the address field before transferring control to the program.

The Go command optionally permits a "breakpoint address" to be entered. A breakpoint address has the same effect as pressing the INTR while a program is being executed. To enter a breakpoint address, press the "NEXT" key after entering the starting address and enter the breakpoint address. Note that when specifying a breakpoint address, the default segment value is either the starting address segment value (if specifiedL of the current CS register contents (if a segment value is not specified with the starting address). When the "." key is pressed, the monitor replaces the instruction and saves the "break pointed" instruction before transferring control to the user's program. When the program reaches the breakpoint address, control is returned to the monitor, the breakpointed instruction is restored in the program, all registers are saved, and the monitor prompts for command entry to allow any of the registers to be examined. Note that since the braakpointed instruction is restored when control is returned to the monitor, the breakpoint address must be specified each time the program is to be executed with a breakpoint.

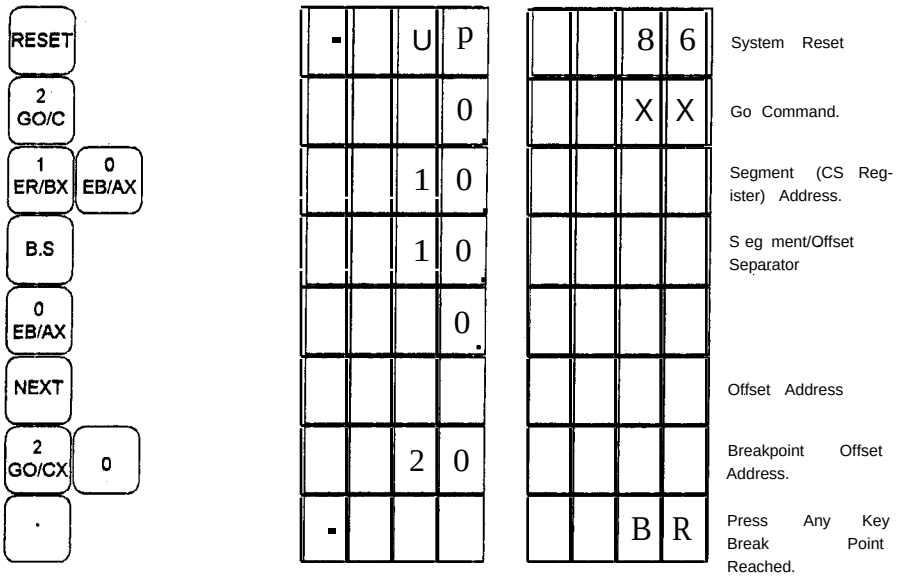
Error Conditions

Attempting to breakpoint an instruction in read-only memory.

Example-1: Transferring Control to the Sample Program.



Example-2: Entering and Executing a Breakpoint in the program. Suppose the breakpoint is to be entered at 0120.



MOVE

Function

The Move (MV) command permits a block of data to be moved within memory.

Syntax

(MV) <start addr> (NEXT) <end addr> (NEXT) <destination addr> O

Operation

The format of the Move Command is unique in that three successive entries are made in the address field. To use the Move Command, press the MV key when prompted for command entry. When the key is pressed, three decimal points appear in the address field to indicate that three entries are required. Each time an entry is made, the left most decimal point goes out so that the number of decimal points lit at anyone time indicate the number of entries still required. The entries are, in order:

1. The starting memory address of the block of data to be moved.
2. The ending memory address of the block of data to be moved.
3. The starting memory address (destination address) into which the block of data is to be moved.

Note that no segment value is permitted with an ending address and that block moves consequently are limited to 64K bytes.

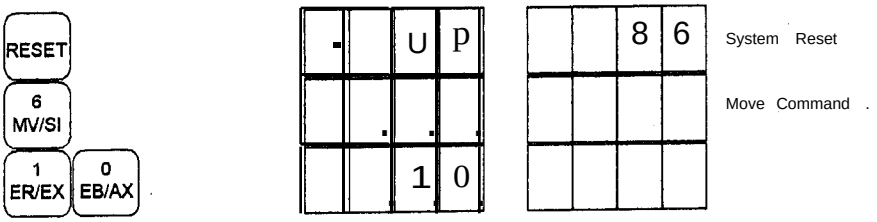
When the "." key is pressed, the data is moved and the command prompt sign is displayed. Note that when the block of data is moved, the data contained in the original (source) memory locations is not altered (unless the destination address falls within the original block of data in which case the overlapping memory locations will be overwritten by the data moved).

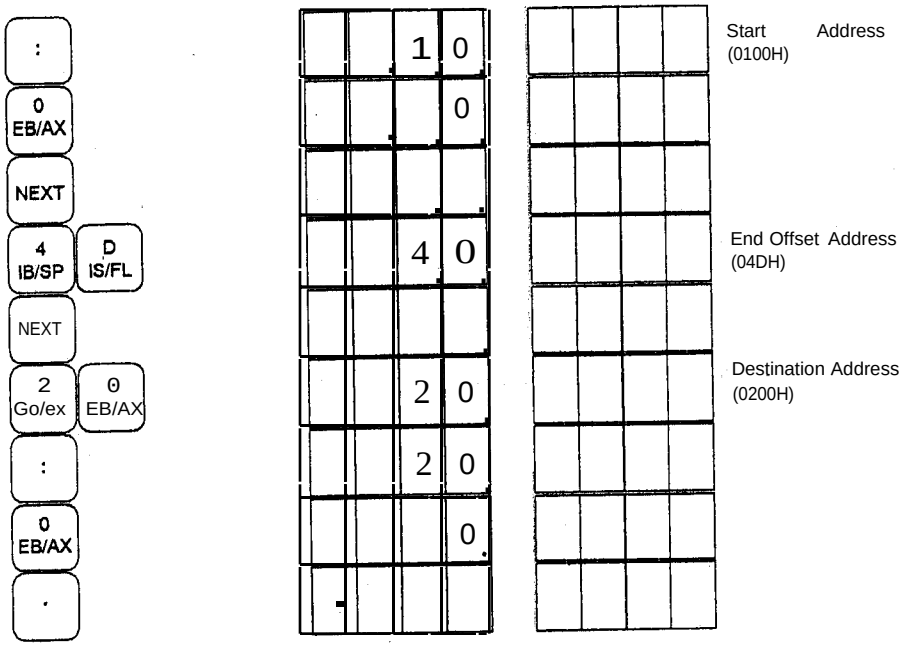
Error Conditions

Attempting to move data into read-only or non-existent memory.

Examples

Example-1: Moving the Program from 0100 to 014d to location 0200H





STEP

Function

The Step (ST) command permits program instructions in memory to be executed individually. With each instruction executed, control is returned to the monitor from the program.

Syntax

(ST) [<start addr>] (NEXT) [[<start addr>] (NEXT) O

Operation

To use the step command press the ST key when prompted for command entry. If a starting address other than the address displayed is required, enter the desired address. When the "NEXT" key is pressed, the instruction addressed is executed and the offset address of the next instruction to be executed is displayed in the address field and its associated instruction byte is displayed in the data field. Again pressing the "NEXT" key execute the current instruction and steps the program to the next instruction to be executed.

In the example which follows, the first few instructions of the "rolling dice" program are executed using the step command. The following table represents a listing of the beginning of that program. (A complete listing of the "rolling dice" program is included at the end of this chapter).

LOCATION	CONTENTS	SYMBOLIC	COMMENTS
00	8CC9	MOVCX,CS	;DEFINE DATA SEGMENT REGISTER EQUAL
02	8ED9	MOVCX,CS	;TO CODE SEGMENT REGISTER
04	6AEAFF	MOVDX,OFF·	;CLEAR DISPLAYCONTROL WORD.
07	60D3	MOVAL,OD3H	;TO 8279.
09	EE	OUT OX	
		READKEY	
OA	BAEAFF	MOV DX,OFFEAH	;SET UP 8279 COMMAND PORT.
00	EC	IN DX	;READ 8279 FIFO STATUS
OE	240F	ANDAL,OFH	;MASK ALL BUT FIFO COUNT.
10	74FB	JZ REAOKEY+3	;KEEP READING IF ZERO
12	E82800	CALL READ	;DUMMY READ TO UNLOAD PRESSED
		ATTA	;KEY FROM FIFO.

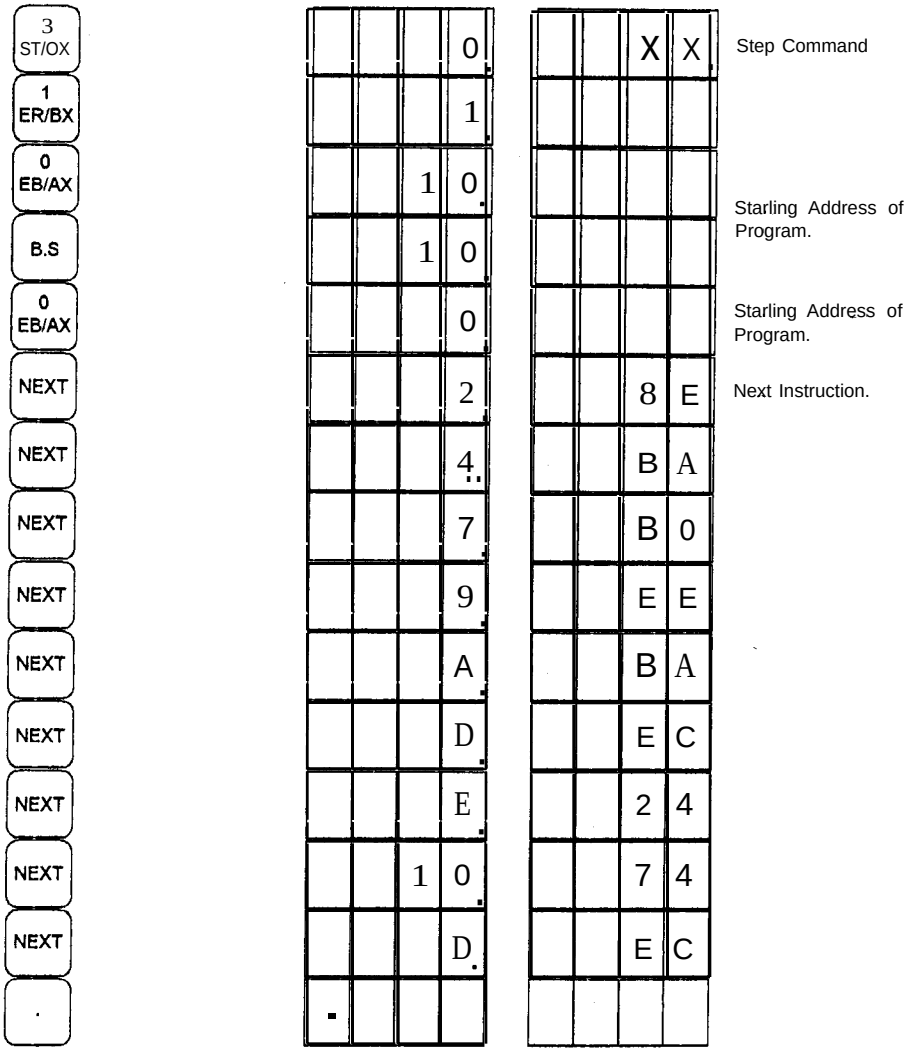
Note that when the program is stepped from the instruction at 1DH, the instruction at DOH is executed again and the instruction at 12H is not executed. This is caused by JZ (Jump Zero) instruction at 1DH which, since no key can be pressed to "roll the dice" (the monitor is in control of the keyboard). Jumps back to the instruction at OOH. Continuing to press the "NEXT" key will repeat the three instruction sequence (DOH, OEH, 1DH).

Restrictions

1. If an interrupt occurs prior to the completion of single stepped instruction or if a single-stepped instruction generates an interrupt, when the monitor is re-entered, the C8 and IP registers will contain the address of the interrupt service routine. Consequently, a type 3 (breakpoint interrupt instruction (OCCH or OCDH)) should not be single-stepped since its execution would step into the monitor.
2. An instruction that is part of a sequence of instructions that switches between stack segments (Le. changes the 88 and 8P register contents) cannot be single-stepped.

Examples

Example-1: Program Stepping



Program Listing

The following is the complete program listing for the "rolling dice" program used in the previous command examples.

LOCATION	CONTENTS	SYMBOLIC	COMMENTS
		ASSUME OS: DICE, CS: DiCE SEGMENT AT 100	
200	8CC9	MOVCX,CS	;DEFINE DATA SEGMENT
202	8E139	MOVDS,CX	;REGISTER EQUAL TO CODE SEGMENT REGISTER
204	BAEAFF	MOV DX,OFFEAH	;CLEAR DISPLAY CONTROL
207	BOD3	MOVAL,OD3H	;WORD TO 8279
209	BE	OUTDX	
LOCATION	CONTENTS	SYMBOLIC	COMMENTS
		READKEY	
20A	BAEAFF	MOV DX,OFFEAH	;SET UP 8279 COMMAND PORT
203	EC	IN DX	;READ 8279 FIFO STATUS
20E	240F	AND AL,OFH	;MASK ALL BUT FIFO COUNT
210	74FB	JZREADKEY+3	;KEEP READING IF ZERO
212	E82800	CALL READATA	;DUMMY READ UNLOAD ;UNLOAD THE PRESSED KEY FROM FIFO
		ZERO	
215	BBooOO	MOV BX,OOOOH	;INITIALIZE DICE COUNT TO ZERO.
		START	
218	43	INC BX	;INCREMENT DICE COUNT
219	80FB07	CMP BL,07H	;IF COUNT EQUALS 7,
21C	74F7	JZ ZERO	;RESET COUNT TO 0
21E	8BFB	MOVDI,BX	;PUT COUNT IN DI REGISTER
220	8A4D4702	MOV CL,COTBL (01-1)	;PUT7 -SEGMENT,DISPLAY ;CODE INCL REGISTER. ;USE DI AS POINTER INTO CODE TABLE.
224	BAEAFF	MOV DX,OFFEAH	;OUTPUT "WRITE DISPLAY"
227	B087	MOVAL,087H	;CONTROL WORD TO 8279
229	BAE8FF	MOV DX,OFFE8H	;OUTPUT THE DICE COUNT
22D	8ACI	MOVAL,CL	;TO 8279 DATA PORT
22F	BE		;
230	BAEAFF	MOV DX,OFFEAH	;READ 8279 FIFO STATUS
233	EC	IN DX	
234	240F	ANDAL,OFH	;MASK ALL BUT FIFO COUNT
236	74EO	JZ START	;KEEP COUNTING IF NO KEY PRESSED
238	E80200	CALL READATA	;DUMMY READ TO UNLOAD ;THE PRESSED KEY FROM FIFO

LOCATION	CONTENTS	SYMBOLIC	COMMENTS
23B	EBCD	JMP READXKEY	;GO READ NEXT KEY PRESSED, ;THEN START COUNT AGAIN
		READATA	
230	BAEAFF	MOV DX,OFFEAH	;OUTPUT "READ DATA"
240	B040	MOVAL,040H	;CONTROL WORD TO 6279
242	EE	OUT OX	
243	BAE8FF	MOV OX, OFFE6H	;SET UP AND READ 6279
246	EC	IN OX	;DATA PORT, RESULT IN AL REG-ISTER
247	C3	RET	;RETURN
		CDTBL	
246	06	DB 06H	;7-SEGMENT CODE FOR "1"
249	5B	DB 05BH	;7-SEGMENT CODE FOR "2"
24A	4F	DB 04FH	;7-SEGMENT CODE FOR "3"
~B	66	DB 066H	;7-SEGMENT CODE FOR "4"
24C	60	DB 06DH	;7-SEGMENT CODE FOR "5"
240	70	DB 07DH	;7-SEGMENT CODE FOR "6"
		DICE ENDS	

DELETE

The Delete Command (DEL) permits the deletion of one or more Bytes from the user's program.

Syntax

@ <start address> (NEXT) <End address> (NEXT) <Start Address for deletion> (NEXT) <End address of Deletion>

Operation

When DEL Key is pressed, four decimal points appear in the address field indicating that four address entries are to be made. Each time an entry is made, the left most decimal pt. goes out. The entries are in the following order.

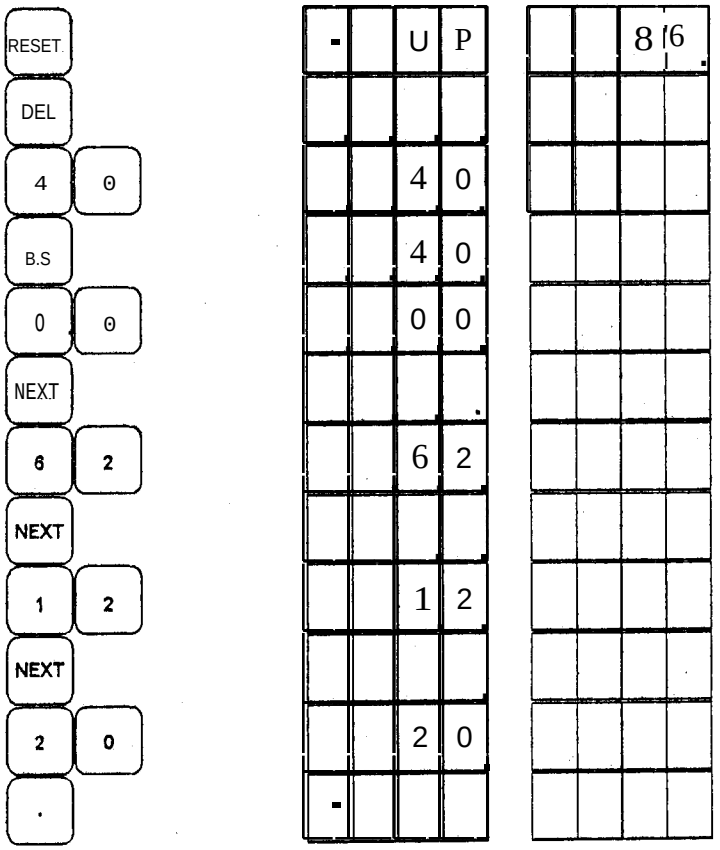
1) The starting address of the program.
2) End address of the program.
3) Starting address from where the deletion should start.
4) End address till where the deletion is to be done.

When "." key is pressed, the command prompt sign is displayed.

Error Condition

1) The starting and end address of the deletion are not within the boundary of the program.
2) The end address of the deletion is tess than the start address of deletion.

Example-1: Delete the portion from 0412H to 0420 from a program lying from 0400 to 0462.



You can verify that the program below 0420 has shifted upward.

INSERT

The INSERT (INS) command permits the insertion of one or more bytes in the user's program.

Syntax

@ <start address> (NEXT) <End address> (NEXT)

<address from which Insertion is to be done> (NEXT) <no. of bytes> (TTY)

Operation

When INS is pressed, four decimal points appear in the address field indicating that four address entries are to be made. The followWing is the order of entries

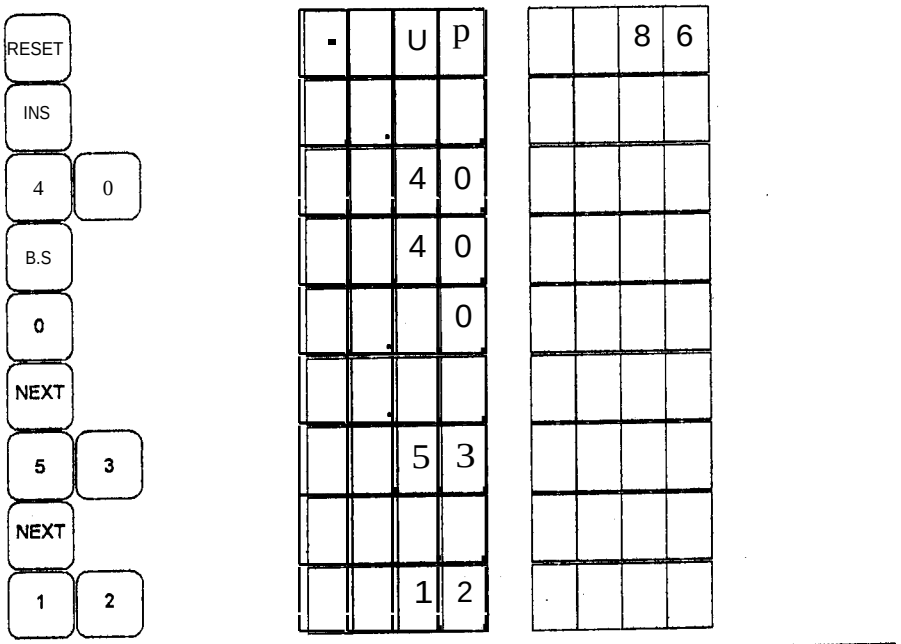
1) The starting address of the program.
2) End address of the program.
3) Address at which bytes are to be inserted.
4) Number of bytes to be inserted.

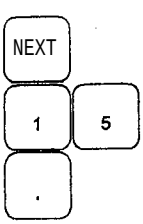
When "." key is pressed, the command prompt sign is displayed.

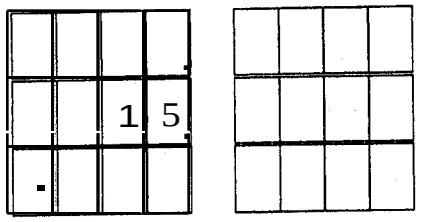
Error Condition

The address at which insertion is to be done, does not lie within the program limits.

E;)ampe-L: .:etus h:Ertthe sam em. ofbytes wh±h We:IE c:E:e'tEd. :h eo.rl:i=+
E*Iffip1=i.e.i1Eert15bytesat0412:h 1hep~ .¥bgret'Wem
0400 to 0453.







A space for 15 bytes has been created from address 0412 onward. We can now enter the byte at 0412 onward using the Examine Byte command.

ON BOARD INTERFACES

5

GENERAL DETAILS OF INTERFACES

To enhance the capabilities of the system on board interfaces for RS-232-C and EPROM Programmer have been provided. Most of the CRT terminals provide RS-232-C and (0-20mA) loop interface and so a CRT terminal can be connected to it through any of the two standard interfaces. The RS-232-C provided through Intel's 8251A (USART).

The two interfaces can be used at any of the baud rate 110, 150, 300, 600, 1200, 2400, 4800, 9600 or 19.2K. These baud rates are generated hardware wise using counters. The baud rate is selected by DIP Switch S1 on the board of VMC-8603 marked Baud rate. This is shown hereunder:

MAIN RS-232-C	
1	19.2K
2	9600
3	4800
4	2400
5	1200
6	600
7	300
8	150
0	110

The on board EPROM programmer interface can be used to store the developed program or data permanently into an EPROM. This interface is provided using I/O lines of 8255 provided on the board.

The relevant details for interfaces are also given below:

S.No.	Device to be Connected	Interface	Connection Details	Connector
1.	CRT Terminal	RS-232-C (Main)	Pin No.1 - NC Pin No.2 - RxD Pin No.3 - TxD Pin No.4 - DTR Pin No.5 - GND Pin No.6 - DSR Pin No.7 - RTS Pin No.8 - CTS Pin No.9 - NC	J3
2.	EPROM Programmer	Through 48 I/O lines of 8255I & II.		

Please note that actually RS-232-C (Main) interface provided on VMC-8603 is a seven line interface. For details of other lines see the detail of connector C3 in Chapter-7.

CRT TERMINAL INTERFACE

A CRT Terminal can be connected to VMC-8603 either through RS-232-C interface. The RS-232-C interface provided is with handshake (7 lines). For using RS-232-C interface connect the three lines CRT OUT (Tx+), CRT IN (Rx+) and GND of VMC-8603 to Rx+, Tx+ and GND of the Terminal respectively. In case of hand shake other signals like DTR, RTS, CTS & DSR can also be connected to the respective signals of terminal.

Select the baud rate of the terminal and set the baud rate of the VMC-8603 interface (RS-232-C Main) to the same value by choosing a proper jumper connection/DIP switch as mentioned earlier.

Press CRT key on VMC-8603. The following should occur.

1) A prompt message "SEriAL" is displayed on the display of VMC-8603.
2) A message "VMC-8603 MONITOR V1.2" is displayed on the terminal on a new line.
3) A prompt character "." is displayed on the next line.

The kit is now ready to interact with the terminal at the baud rate set earlier. The key board of the kit remains disabled. To bring back the kit in the key board mode press RESET key on the kit board.

EPROM PROGRAMMER

VMC-8603 provides onboard EPROM PROGRAMMER for the 2764/27128/27256 EPROMS. The following commands can be used in the EPROM-Programmer mode.

1) BLANK CHECK
2) LIST
3) VERIFY
4) PROGRAM (DUPLICATE)

Since 8086 is a 16 bit processor and the memory chips available are of 8 bits, the odd and even bytes of 16 bit Data have to be separated out. And so while using list, verify and program command, the user should specify whether the operation is to be done on odd bytes or even bytes or continuous bytes. This is done by using different terminators for the Commands.

a 'o' is used as a terminator for continuous Bytes.
a '1' is used as a terminator for odd Bytes.
a '2' is used as a terminator for even Bytes.

The 20 bit address of 8086 is divided into two parts, the segment address and the offset address. The starting address of any EPROM in the ZIF socket is 0000 and the maximum address will depend upon the capacity of the chip. In case of 27128 chip, the maximum address would be 3FFF. The address of the EPROM in the ZIF is also divided into segment address and offset address. In this case the segment address should be always given as 0000 followed by the offset address which would be the actual address. In case of LIST, VERIFY & PROGRAM Commands, three addresses are to be entered. Two addresses correspond to the memory on the board of VMC-8603 and the third address correspond to the EPROM in the ZIF socket.

It is important to note that in case of the Even Byte operation, the two addresses corresponding to the Board of VMC-8603 should both be even addresses and in case of the Odd Byte operation, both the addresses should be odd addresses. A deviation from above i.e. a combination of Even and Odd address with the terminator as 1 or 2 would result in Error condition and in this case, the command does not terminate at all unless Reset Key is pressed.

BLANK CHECK

Blank check command is used to check the EPROM placed in the ZIF (Zero Insertion Force) socket for blank.

Syntax

(BC/IP) <start address> (NEXT) <End address> (TTY)

Operation

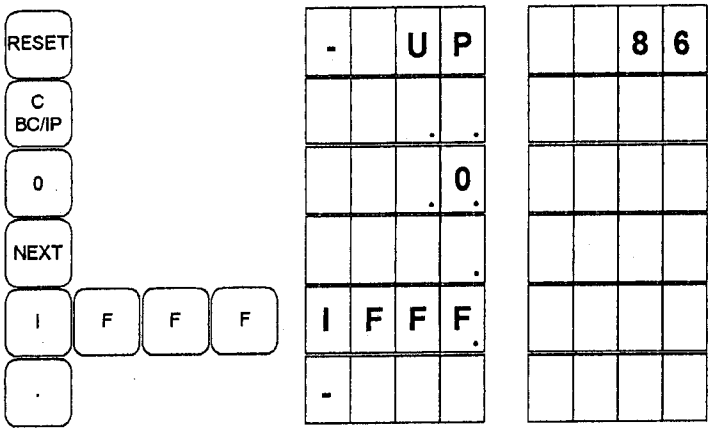
When BC Key is pressed, two decimal points appear in the address field to indicate that two address entries are required. Each time an entry is made, the left most decimal point goes out. The entries are in order

1) The starting address of the EPROM from where the blank check should start.
2) The End address of the EPROM till where the system should check for blank.

Blank check can be performed on 2764/27128/27256. Please see the note at the end of this chapter.

When the "." key is pressed, the command prompt sign "." is displayed if the EPROM area being checked is blank. If any location is not blank, its address will be displayed in the address field and its contents in the data field. If this is the only location which is not blank, then on pressing NEXT key, the command prompt sign will appear otherwise the address of the next location which is not blank will be displayed and so on.

Example-1: Perform a blank check on a blank 2764 EPROM (entire 8K).



VERIFY

This command is used to verify the content of the EPROM put in the ZIF Socket with any memory area on the board of VMC-86/3 (RAM OR ROM). Please see the note at the end of this chapter.

Syntax

(E VR/) <start address of onboard memory>(NEXT)

<end address of onboard memory> (NEXT)

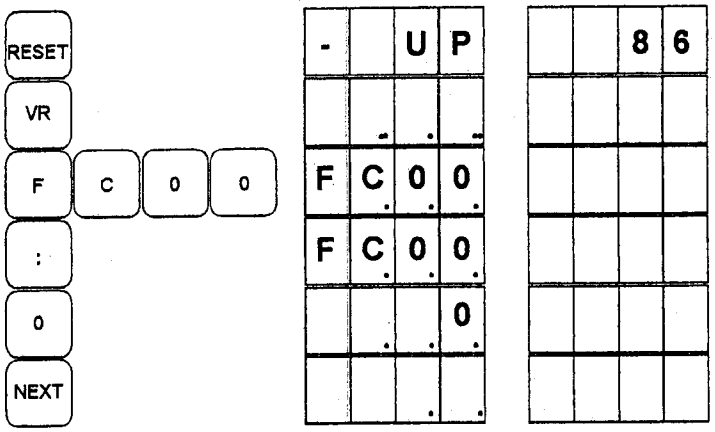
<start address of EPROM IN ZIF> (NEXT)

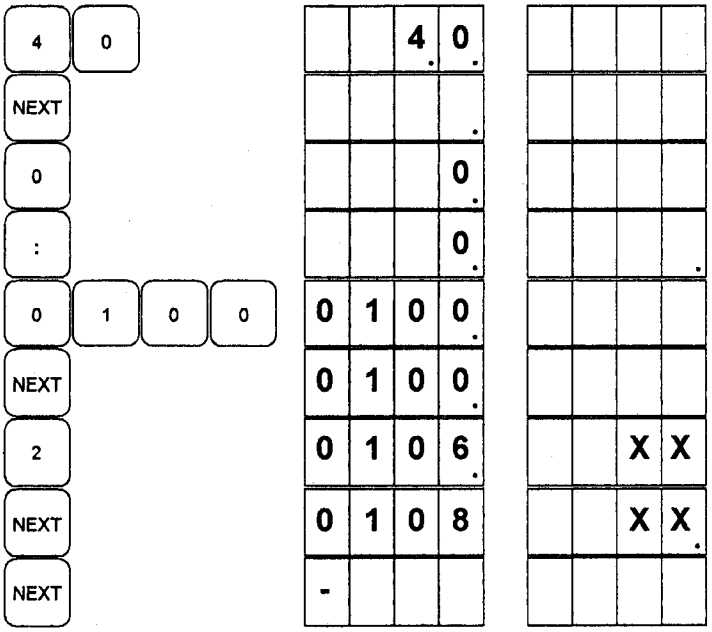
Operation

On pressing VR key, three decimal points will appear in the address field indicating that 3 addresses are required to be entered. The entries are to be made in the following order:

1) Starting address of the memory block on VMC-8603 with which the contents of the EPROM in the ZIF is to be verified.
2) End address of the memory block on VMC-8603.
3) Starting address of the EPROM in the ZIF socket. When '0', '1' or '2' key is pressed, the two blocks of memory are verified for equality. If they are equal, the command prompt sign is displayed. If they are not equal, the address at which the discrepancy occurs and its contents are displayed. If this is the only location where the discrepancy exist, a command prompt sign is displayed on pressing NEXT key otherwise, the next location address where there is a difference will be displayed.

Example-1: Verify the contents of 2764 EPROM from location 0100 to 0120 (Even Byte) with the Monitor of VMC-8603 starting from FC000 to FC040. We assume that the two blocks are not same and there is a discrepancy at FC006 and FC008.





LIST

This command is used to list the contents of an EPROM in the ZIF socket, in the system RAM area. Please see the note at the end of this chapter.

Syntax

(LS) <start address> (NEXT) <end address of RAM> (NEXT)

<start address of EPROM> (NEXT)

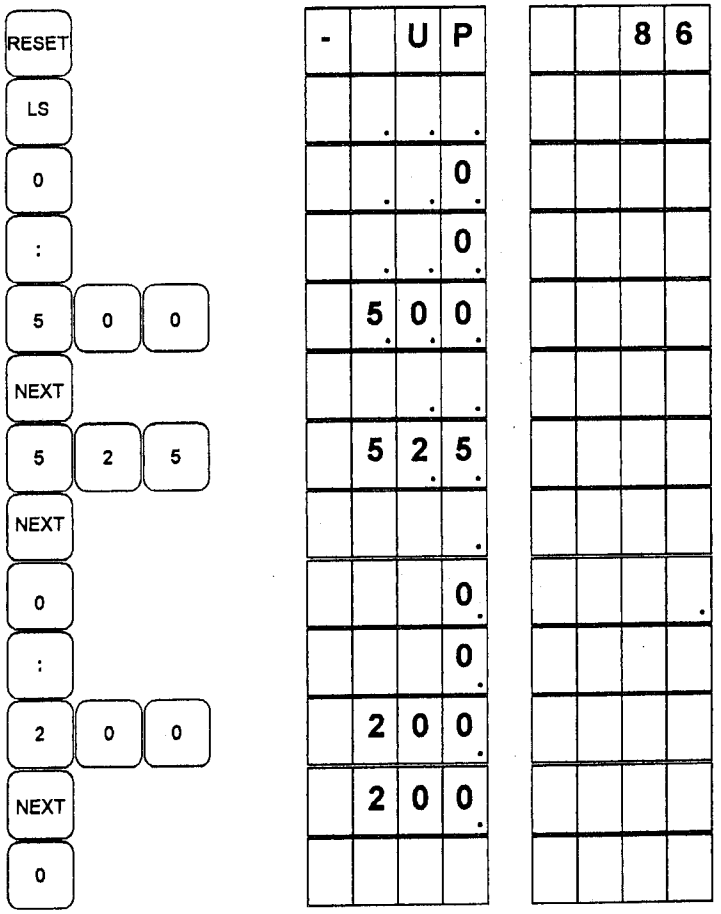
Operation

On pressing LS key, three decimal points are displayed in the address field indicating that three addresses are required to be entered. The entries are to be made in the following order.

1) Starting address of the RAM where the listing is to be made.
2) End address of the RAM where the listing is to be made.
3) Starting address of the EPROM in the ZIF socket from where the listing is to be made in the RAM area.

On pressing '0' or '1' or '2' key, the command prompt character is displayed.

Example-1: LIST the contents of EPROM 2764 from 0200 in to RAM area 00500 to 00525 (Continuous Bytes).



PROGRAM

This command is used to program any of the earlier mentioned EPROM with the data lying any where on the board of VMC-8603 (ROM or RAM). Please see the note at the end of this chapter.

Syntax

(PRG) <starting address of onboard memory> (NEXT)

<End address of onboard memory> (NEXT)

<starting address of EPROM on ZIF> (NEXT)

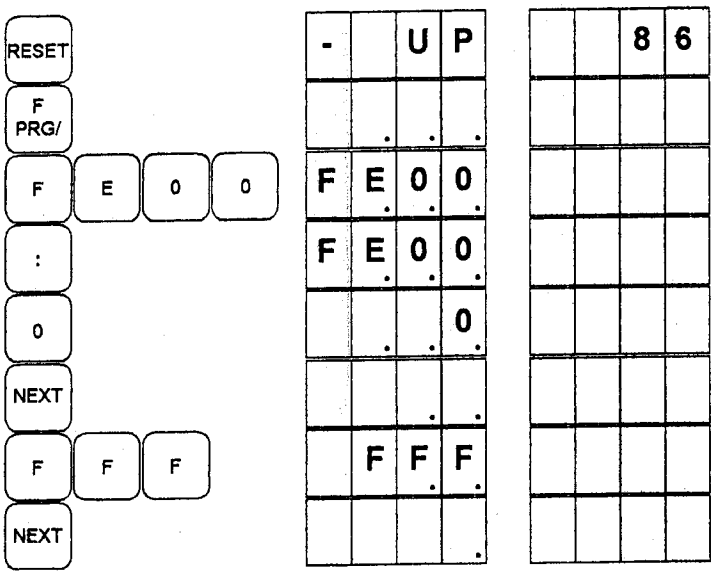
Operation

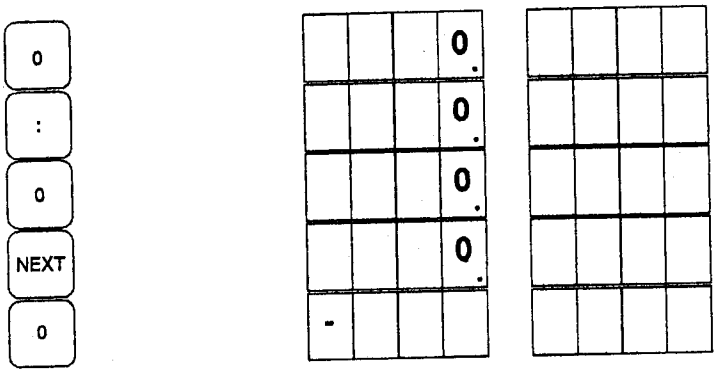
On pressing PRG Key, three decimal points are displayed in the address field indicating that three address entreis are to be made. The entries are to be made in the following order.

1) Starting address of the onboard memory from where the data is to be taken.
2) End address of the onboard memory till where the data is to be taken.
3) Starting address of the EPROM in the ZIF socket from where the programming should start.

On pressing '0'or'1'or'2' key, the programming will start. This is indicated by blinking PRG IND LED. When the programming is over, a command prompt character is displayed in the address field.

Example-1: Program 2732 EPROM with the EPROM lying in VMC-8603 at the address FE000 to FFFFF (continuous Byte)





Note 1. *Before performing any of the EPROM Programmer Commands, ensure that the required Personality module has been put in the socket respectively.*

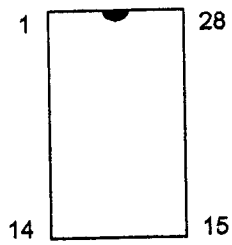
2. Before programming, the pink wire of the power connector (C6) should be connected according to the programming voltage required for the EPROM.

EPROM	Prog. Voltage
2764	21V
27C64	12.5V
27128	21V
27C128	12.5V
27256	21V
27C256	12.5V
2716	25V
2732	21V

*(Programming Voltage is selected by switch given at the back in the model VMC-8603P)

* For all CMOS (27C....) EPROMs it is 12.5V only.

3. *Notch should point in the upper direction, base to be fixed at Pin No.14 (GND).*



SERIAL I/O DEVICE COMMANDS

6

GENERAL

It has been mentioned earlier that the VMC-8603 can be connected to serial devices like Teletypewriter or CRT Terminal through 20mA current loop or RS-232-C interface. User can enter in the serial mode from the keyboard of VMC-8603 using TTY key or CRT key depending upon which serial interface is being used.

On entering into serial mode a prompt message "SEriAL" is displayed on the address and Data field of VMC-8603 DISPLAY. ("SEri" is displayed in the address field and "AL" is displayed on the data field). At the same time a message "VMC-8603 MONITOR V1.2" is displayed on the serial device on one line and a prompt character "." on the next line to indicate that it is ready to accept command entry.

This line is referred to as the "command line" and consists of either a one-or-two-character command mnemonic followed by one to three command parameters or "arguments". (If desired for visual separation, a space can be entered between the command mnemonic and the first argument). When more than one argument is required, a single comma (",") is used between arguments as a delimiter.

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

With the exception of the register abbreviations associated with the X (Examine/ Modify Register) command, all arguments are entered as hexadecimal numbers. The valid range of hexadecimal values is from 00 to FF for byte entries and from 0000 to FFFF for word entries (leading zero's may be omitted). If more than two (byte entries) or four (word entries) digits are entered, only the last two or four digits entered are valid. Address arguments consist of segment value & an offset value. If a segment value is not entered, the default segment value is the current contents of the code segment (CS) register unless specified otherwise in the command description. When both a segment value and an offset value are entered as an address argument, entry is the offset value. A colon (":") is entered the first entry is the segment value, and the second between the entreis as a separator.

Since command execution occurs only after a command terminator is entered, a command entry can be cancelled any time before the terminator is entered by pressing any character that is not legal for the entry expected. When a command is cancelled, the number symbol ('#') is output on the command line, a carriage return and a line feed are issued and the command prompt character ('.') is output on the new line.

XTALK is a simple terminal emulator software for IBM-PC/XT/AT compatible computers. It allows the user to communicate with the computer through serial port with the facility of downloading & uploading of the data between the computer and the other serial devices. The various communication parameters like baud rate (speed), number of data bits, stop bits, parity etc. can be changed. The package communicates through COM1: as well as COM2: ports of the IBM-PC/XT/AT system.

INSTALLATION

a) Insert the diskette containing XTALK in your A: drive.
b) If you have hard disk, make a directory in the name of XTALK in your c: drive.
c) Copy the diskette in your directory as given below:
   A> COPY *.* C:\XTALK <CR>
d) Execute XTALK as given below:
   XTALK <CR> (either from A: or C:)

SECTION OF COMMUNICATION PARAMETERS

For setting the parameters, press <HOME> key. The system will show you a 'COMMAND? Prompt which means that the system is asking you to give commands. Write 'SP-9600' to set the baud rate at 9600.

COMMAND? SP 9600

The baud rates available in XTALK are: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200 etc. If you want to set the baud rate at say 4800, then in "COMMAND?" Prompt, write SP-9600 and press <CR>, Rest of the parameters viz. Parity, Duplex Mode, Data Parity, Full Duplex Mode, 8 Data Bits and one Stop Bit. If you want to change any of the parameters write the first two letters e.g., `PA for Parity, `DU' for Duplex Mode and then write as per requirement. Suppose you want to change Parity, you should write in the COMMAND? Prompt PA EVEN, for odd Parity write PA ODD. Similarly one can change Duplex mode, Port etc.

After setting the parameters, write GO LO in COMMAND? Prompt. It will show you that "Local data link is now active. Press <CR> and on screen you will get a prompt to start with.

COMMUNICATION BETWEEN XTALK SOFTWARE & VMC-8603 KIT

a) Connect the Kit and Power Supply. Make sure that the +5V, GND, +12V & -12V are connected properly.
b) Connect the serial port of PC with the RS-232-C (9 pin connector) of the kit.

g) Your PC & Kit are now linked for communication and all your instructions can now be operated from your PC Keyboard.

All serial commands mentioned in this chapter will be fed through PC Keyboard. Uploading & Downloading can also be done through this Kit.

NOTE: The standard buffers used in RS232C interface are 1488 and 1489 IC's. In case of any problem in serial communication, one scan suspect these going bad. The RS232C port also does not work in case of failure of +12V or -12V supply. The voltages can be checked at pin No.1 of IC 1488 and pin No.14 of IC 1488 to be -12V and +12V respectively with respect to ground.

LIST OF MONITOR COMMANDS

VMC-8603 serial monitor is capable of executing the following commands:

1) Substitute Memory (S)
2) Examine/Modify Register (X)
3) Display Memory (D)
4) Move (M)
5) Port Input (I)
6) Port Output (O)
7) Go (G)
8) Single Step (N)
9) Read Hex file (R)
10) Write Hex file (W)
11) Uploading Data (W Starting Address , Ending Address)
12) Downloading Data (R)

While explaining the commands the following syntax is used:

[A] indicates that "A" is optional

[A]* indicates one or more occurrences of "A"

<B> indicates that "B" is a variable

<Cr> indicates that a carriage return is entered.

c) Baud Rate Selection on the VMC-8603:

The two interfaces can be used at any of the baud rate 110, 150, 300, 600, 1200, 2400, 4800, 9600 or 19.2K. These baud rates are generated hardware wise using counters. The baud rate is selected by a DIP Switch S1 on the board of VMC-8603 marked Baud rate. This is shown hereunder:

MAIN RS-232-C	
1	19.2K
2	9600
3	4800
4	2400
5	1200
6	600
7	300
8	150
0	110

d) Switch on the Computer and the VINYTICS VMC-8603 Kit..

e) After loading XTALK, set the parameters as shown in the table below:

PARAMETERS TO SET	COMMANDS TO GIVE IN XTALK MODE			
SPEED	9600 BAUDRATE SP		9600	
DUPLEX MODE	FULL		DU	FULL
PARITY	NONE		PA	NONE
STOP BITS	1		ST	1
DATA BITS	8		DA	8
CW AIT	DELAY	20	CW	20
LW AIT	DELAY	20	LW	20
EMIATION	NONE		EM	NONE

Write GO LO in COMMAND prompt and press <CR>

To save the above settings to a file, give save command. SAVE VMC-8603 and press <CR>. After this command when one uses XTALK again the parameters can be baded either using LOAD 'LO' command or as soon as XTALK is executed, it displays all the transmission files which has extension *XTK. One can enter the number of file directly. Now the kit is ready for serial communication.

f) Press CRT Key of the kit and Serial Monitor Ver 1.2 will be displayed indicates that communication is done between kit and IBM PC.

COMMAND DESCRIPTION

SUBSTITUTE MEMORY

Function

The substitute Memory (S) command is used to examine the byte (S) or word (SW) contents of select memory locations. If the contents of the memory location can be modified (e.g., a RAM location), the contents additionally can be updated with a new data value entered from the console device.

Syntax

S [W] <ADDR>, [[<NEW CONTENTS>],]* <CR>

Operation

To use the Substitute Memory command, enter S or SW when prompted command entry and enter the address of the memory location to be examined. Note if a segment address value is not specified, the current contents of the code segment (CS) register are used by default. After the address is entered, enter a comma. The monitor location followed by a dash (the monitor's data entry prompt character) and a space to indicate that the addressed location is open for update. Note that when using the SW command, the byte contents of the next consecutive memory location (memory address + 1) are outputted first, followed by the byte contents of the actual location addressed. Similarly, when updating memory contents using the SW command, first the next consecutive memory location, and the second byte entry (next two digits) will be written into the addressed memory location.

If only one memory location is to be examined enter a carriage return to terminate the command. (If new data was entered, it is updated when the carriage return is entered). If a series of continuous memory locations are to be examined and/or updated, enter a comma to advance to the next consecutive memory location (S Command) or next two consecutive memory locations (SW command). Again, if the data contents are not to be updated, enter a comma to examine the next memory location or enter the new data followed by a comma to update the current location and to examine the next location. Entering a carriage return terminates the command.

Error Conditions

Attempting to modify a non-existent or read-only (e.g.., ROM or PROM) memory location.

Examples

Example-1: Examine ROM location OFF00H

S FF00:0, 90-<CR>

Example-2: Examine RAM Location 050H, relative to the DS register, and update the contents of location 051H to 0F7H

S DS : 50, E4-,
0051 A4 -F7 <CR>

EXAMINE/MODIFY REGISTER

Function

The Examine/Modify Register (X) command is used to examine and, if desired, to modify any of the 8086's individual register or to examine the contents of all of the 8086 registers.

Syntax

x [<REG> [[<NEW CONTENTS>],] * <CR>

Operation

To use the Examine/Modify Register Command, enter X when prompted for command entry. If you only wish to examine the current contents of the registers, enter a carriage return. (The contents of all fourteen registers will be output.) If you wish to examine and optionally to modify the contents of an individual register, enter the register's abbreviation according to Table - 6.1

TABLE - 6.1 REGISTER ABBREVIATIONS

Register Name	Abbreviation
Accumulator	AX
Base	BX
Count	CX
Data	DX
Stack Pointer	SP
Base Pointer	BP
Source Index	SI
Destination	DI
Code Segment	CS
Data Segment	DS
Stack Segment	SS
Extra Segment	ES
Instruction Pointer	IP
Flag	FL

When a register abbreviation is entered, the monitor outputs an equal sign ("="), the current register contents, the data prompt character ("-") and a space. If you wish to change the register's contents, enter the new contents followed by a comma (to advance to the next" sequential" register) or a carriage return (to terminate the command). The register sequence is in the order shown in Table 5.1. Note that the sequence is not circular and that if a comma is entered after the contents of the flag (FL) register are examined or a modified, the monitor returns to the command mode. When a carriage return is entered, the register is updated (if new contents were entered) and the monitor returns to the command mode.

Examples

Example-1: Examine the 8086's registers

X <CR>

AX =89D3	BX =0002	CX =0010	DX =FFEA	SP =0100
BP =D3EB	SI =9295	DI =0002	CS =0010	DS =0010
SS =0000	ES =0000	IP =000D	FL =F046	

Example-2: Modify the CS register and examine the next two registers.

X CS= 000-20,
DS= 0010-,
SS= 0000-<CR>

DISPLAY MEMORY

Function

The Display Memory (D) command is used to output the contents of a block of memory to the console device.

Syntax

D [W] <start addr> [<end addr>] <CR>

Operation

The command provides a line-formatted output of the memory block bounded by start address and end address (inclusive). Since end address is relative to the segment address value specified or implied with start address (CS register contents if a segment value is not specified with start address), no segment value is permitted with end address and block transfers consequently are limited to 64K bytes or 32K words with each command execution.

To use the Display Memory Command, enter D (for byte output) or DW (for word output) prompted for command entry) and then enter start address of the memory data block. If only one byte or word is to be displayed, enter a carriage return, while if a block of memory is to be displayed, enter end address and a carriage return. The monitor will output, beginning on the next line, the starting offset address, the data contents of that location end, when end address is specified, the data contents of a number of consecutive memory locations, each separated by a space. The line format output is arranged so that any subsequent lines (if required) will begin with the offset address of the first byte or word in the line and will consist of a maximum of either sixteen byte entries or eight word entries per line.

The Display Memory command can be cancelled or the output can be stopped at any time by entering control characters from the console device. Control C immediately terminates the command and returns the monitor to the command entry mode. Control-S Stops the output, but does not terminate the command. Control-Q resumes output that has been stopped. The only allowed console input following a Control-S is either a Control-Q or Control C.

Error Condition

End address less than the offset value of start address.

Example-1: Display contents of locations 09H through 02AH relative to the DS register.

```
D DS:9, 2A <CR>
0009  EE BA EA FF EC 24 OF
0010  74 FB E8 27 00 00 00 43 80 FB 07 F7 8B FB
0020  8A 4D 46 BA EA FF B0 87 EE BA E8
```

Example-2: Display contents of locations FF000H through FF02AH in word mode.

```
DW FF00:0, 2A <CR>
0000  E990  0098  0072  0088  4328  2029  3931  3837
0010  4920  544E  4C45  4320  524F  4050  0000  4000
0020  7F00  007D  8600  5006  0000  4000
```

MOVE

Function

The Move Command is used to move a block of data within memory.

Syntax

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

Operation

When using the Move command, the contents of the memory block bounded by start address and end address (inclusive) are moved to consecutive memory locations beginning at destination address. As with the D (Display Memory) command, end address is relative to the segment address (if no segment value is implied with start address the CS register contents are used). Consequently, no segment value is permitted with end address, and block moves are limited to 64K bytes.

Since a move is performed one at a time, the Move Command can be used to fill a block of memory with a pre-defined constant. This is accomplished by specifying a destination address which is one greater than start address. The block of memory locations from start address to end address + 1 are filled with the value contained in start address. (The S command is used prior to the Move command to define the constant at start address).

Error Conditions

Attempting to move data to a read-only (e.g., ROM OR PROM) or non-existent memory location.

Specifying an end address value which is less than the offset value of start address.

Examples

Example-1: Move the contents of locations 0200H through 0250H, relative to the DS register, to the memory block starting at the destination address defined by a segment value equal to the ES register plus 010H and an offset value of 02CH.

M DS: 200, 250, ES+10:2C <CR>

Example-2: Move the contents of locations 0100H through 014CH to the memory block beginning at 0500H relative to the CS register.

M 100, 14C, 500 <CR>

PORT INPUT

Function

The Port Input (I) Command is used to display a byte or word at an input port.

Syntax

I[W] port addr , [.]* <CR>

Operation

The Port Input Command inputs a byte (I Command) or word (IW Command) from the port specified by port address and displays the byte or word value on the console device. Since I/O addressing is limited to 64K I/O byte addresses, no segment value is permitted with port address. After port address is entered, a comma is required to cause the byte or word at the input port to be displayed at the console. Each subsequent comma entered causes the current data at the addressed input port to be displayed on a new line. A carriage return terminates the command and causes the monitor to prompt for command entry.

When using the port Input command to input data from the 8255A parallel I/O port make sure that they have been defined earlier as input port.

Note that these circuits are programmed for input on power-on/reset and that if they were last programmed for output, they must be reprogrammed for input.

When using word input (IW Command), the port P2 (low-order byte) address is entered as port address.

Example

Example-1: Input single word from parallel I/O ports P1A and P2A.

IW FFF8,
C7C5 <CR>

Example-2: Input multiple bytes from port 02FAH

I 2FA,
FF,
FC,
00,
B7,
3F,<CR>

PORT OUTPUT

Function

The Port Output (O) Command is used to output a byte or word to an output port.

Syntax

O [W] <port addr> , <data> [, <DATA>]* <CR>

Operation

The Port Output command outputs the byte (O Command) or word (OW Command) entered as data to the output port specified by port address. Like the Port Input Command I/O addressing is limited to 64K I/O byte addresses, and no segment value is permitted with port address. After entering port address, a comma and the data to be output, a carriage return is entered to cause the data to be output to the port and to terminate the command, or a comma is entered to permit subsequent data output to the addressed port. Data can be output repetitively to the port by entering new data followed by a comma. A carriage return following a data entry outputs the data and terminates the command.

As mentioned in the previous section, the two 8255A parallel I/O Port circuits are programmed for input on power-on or system reset. Consequently, to use the Port Output Command to output data from the parallel I/O Ports, the circuits first must be programmed for output. This is accomplished by using the Port Output Command to output a control byte (or word) to the 8255A circuit's control port.

Examples

Example-1: Program parallel I/O port P2 for output

O FFFF, 80 <CR>

Example - 2: Output multiple words to I/O port 020F0H

OW 20F0, BAEA,
- 4CFF,
- B0AE
- EE47,
- F9D3 <CR>

Example-3: Output single byte to parallel I/O port PIC

O FFFC, 3 C <CR>

GO

Function

The Go (G) command is used to transfer control of the 8086 from the serial monitor program to a user's program in memory.

Syntax

G [<start addr>] [,<breakpoint addr>] <CR>

Operation

To use the GO Command, enter G when prompted for command entry. The current IP (Instruction Pointer) register contents, the data entry prompt character and the byte contents of the location addressed by the IP register are outputted. If an alternate starting address is required, enter start address. To transfer control from the monitor to the program and to begin program execution, enter a carriage return.

The Go Command optionally permits a "breakpoint address" to be entered. Note that when specifying breakpoint address, the default segment value is either the start address segment value (if specified) or the current CS register contents (if a segment value is not specified with start address). When breakpoint address is specified, the monitor replaces the instruction at the addressed location with an interrupt instruction and saves the "breakpoint" instruction. When the program reaches breakpoint address, control is returned to the monitor, the breakpointed instruction is replaced in the program, all registers are saved, and the monitor outputs the following message followed by a command prompt to allow any of the registers to be examined.

BR @ aaaa:bbbbb

In the above message, "aaaa" is the current CS register value, and "bbbb" is the current IP register value. (The combined register value is the address of the breakpointed instruction). If a subsequent Go command is entered, execution resumes at the replaced breakpointed instruction. Note that since the breakpointed instruction is replaced when control is returned to the monitor, breakpoint address must be specified each time a program to be breakpointed is executed.

Error Conditions

Attempting to breakpoint an instruction in read-only memory.

Examples

Example-1: Transfer control to the program at 04C0H, relate to the CS register.

G 0000D-Ec 4C0 <CR)

Example-2: Transfer control to the program at 10:0H and break at the instruction in location 10:37H cr

G 010E-24 10:0, 37 <CR>
BR @ 0010 : 0037

SINGLE STEP

Function

The Single Step (N) command is used to execute a single user-program instruction. With each instruction executed, control is returned to the monitor to allow evaluation of the instruction executed.

Syntax

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

Operation

To use the Single Step command, enter N when prompted for command entry. The monitor will output the current instruction pointer (IP) register contents (the current instruction pointer (IP) register contents (the offset address of the next instruction to be executed) and the instruction byte pointed to by the IP (and CS) register. If execution of an instruction at another address is desired, enter start address. If start address includes a segment value, both the CS and IP registers are modified. When the comma is entered, the instruction addressed its executed and control is returned to the monitor. The monitor saves all of the register contents and outputs the address (IP register contents) and instruction byte contents of the next instruction to be executed on the following line. Each time a comma is entered, the addressed instruction is executed and the address and instruction byte contents of the next instruction to be executed are output on a new line.

While using the Single Step command to step through a program, a new start address can be entered without repeating the command entry. When the comma is entered, the instruction addressed is executed and the next instruction's address and byte contents are output. A carriage return terminates the command.

Restrictions

1. If an interrupt occurs prior to the completion of a single-stepped instruction generates an interrupt, when the monitor is re-entered, the CS and IP registers will contain the address of the interrupt service routine. Consequently, a type 3 (breakpoint) interrupt instruction (0CCH or 0CDH) should not be single-stepped since its execution would step into the monitor.

2. An instruction that is part of a sequence of instructions that switches between stack segments (i.e., changes the SS and SP register contents) cannot be single-stepped.

Examples

Example-1: Single Step a series of instructions beginning at 0100H, relative to the CS register.

N 0000- 00 100,
0102 - 8E,
0104 - BA,
0107 - B0,
0109 - EE,
010A - BA <CR>

READ HEX FILE

Function

The Read Hex File (R) command allows the monitor to read an 8086 or 8080 hexadecimal object file from a paper tape and to load the data read from the file into memory.

Syntax

R [<bias number>] <CR>

Operation

To use the Read Hex File Command, enter R when prompted for command entry. When the tape is loaded in the reader and ready; enter a carriage return. The data read from the file will be written into memory beginning at each record's load address. If the file is in the 8086 format and includes an execution start address record, the CS and IP registers will be up dated with the execution address specified in that record. If the file is in 8080 format and includes an EOF (end - of - file) record, the IP register is updated with the execution address specified in the E0F record. Note that a segment address value is not used with the 8080 file format; the data ready is written into memory locations relative to a segment value of zero and, when an E0F record execution address is specified, the CS register is not changed.

When an optional bias number is specified, it is added to each record's load address to offset the file in memory.

Error Conditions

Tape check sum error.

Attempting to load data into non-existent or read-only memory.

Examples

Example-1: Read a file and load the data into memory 256 (decimal) bytes above the load addresses specified in the file.

R 100 <CR>

WRITE HEX FILE

Function

The Write Hex File (W) command allows a block of memory to be written (output), in either 8086 or 8080 hexadecimal object file format, to a paper tape punch.

Syntax

W [X] <start addr> , <end addr> [, <exec addr>] <CR>

Operation

To use the Write Hex file command, enter W for 8086 file format or WX for 8080 file format and enter start address and end address of the memory block to be output. Note that no segment address value is permitted with end address (the start address value is specified with start address, the current CS register value is used. When the carriage return is entered, the following information is punched on the paper tape:

- ✧ Six inches of leader (60 null characters)
- ✧ An extended address record (8086 format only)
- ✧ The data contents of the memory block bounded by start address and end address (inclusive).
- ✧ An end-of-file (EOF) record.
- ✧ Six inches of trailer (60 null characters).

Optionally, an execution address can be specified prior to entering the carriage return. This is the memory address that is loaded into the CS and IP registers (IP register only with 8080 format) when the tape is read with the R command. Depending on the format selected, when execution start address record containing execution address is punched immediately following the tape leader (8086 format) or the offset address value of execution address is punched in the EOF record (8080 format).

When using the 8086 format (W command), the start address segment value (CS register value if a segment value is not specified) is entered (punched) in the extended address record, and the start address offset value is entered in the load address field of the first data record. The segment and offset address values of execution address are entered in the execution start address record (CS register contents if a segment address value is not specified with execution address).

When using the 8080 format (WX command), the start address offset value is punched in the load address offset value is punched in the load address field of the first data record. Execution address, if specified, is punched in the EOF record. Note that a segment address value is not permitted with execution address or end address and that the start address segment value is used only to define the starting address of the memory block and that it is not punched on the tape. The Write Hex File command can be cancelled or stopped at any time by entering control characters from the console device. Control C cancels the command and prompts for new command entry. Control S stops the output, but does not cancel the command. Control-Q resumes output that has been stopped. The only console input allowed following a Control-S is either Control-Q or Control-C.

Error Conditions

Specifying a value for end address that is less than offset value of start address.

Examples

Example-1: Output the memory block bounded by 04H and 06DDH, relative to the current CS register, to an 8086 file with an execution address of CS 040H.

W 4, 6DD, 40 <Cr>

UPLOADING

XTALK provides a feature by which the data stored in VMC-8603 can be stored in the floppy diskette of PC/XT/AT. This can be achieved by following the instructions given as below:

a) Write W 0: 200, 288

b) Press Home Key of the Computer Keyboard, **COMMAND?** will be displayed in the 25th line.

c) Write CA ON

d) Press <CR> thrice and data from AD1 to AD2 will be displayed

e) Again press Home key.

f) Write CA OFF and press <CR>

g) The computer will show you on the screen as follows:

"Information is still in the capture.

Do you want to save it (Y/N)".

If you want to save the data, press Y. After pressing 'Y' it will ask you:

"Write capture buffer to what file? _____"

Name the file in which you want to save the data following .DAT e.g., ABC.DAT and press <CR>.

i) After pressing <CR> on the screen, you will get:

"File successfully written, Press <Enter>".

j) Data captured will be stored in the file as defined by the file name.

DOWNLOADING THE ABOVE FILE FROM PC/XT/AT TO 8085 KIT

The following procedure is to be adopted for downloading the above file from PC/XT/AT to VMC-8603.

The 'R' command loads the data from your diskette/PC to the memory of the Kit.

a) Press 'R' key.

b) Press 'Home key' of the PC Keyboard, **COMMAND?** will appear on the 25th line.

c) Name the file in which you want to send data from floppy diskette to memory.
Write SEND ABC.DAT in command prompt and press <CR> key twice.

d) The programs stored in the file will be loaded into memory of your kit at the address specified in the program.

ABORTING THE OPERATION

For aborting the XTALK, write "QUIT" at COMMAND? prompt.

a) A "." prompt is displayed on the screen.

b) Press Home key on the computer keyboard. "COMMAND?" will be displayed on the 25th line.

c) Write `QU' and XTALK operation is aborted.

SAMPLE PROGRAMS

7

INTRODUCTION

The monitor software of VMC-8603 resides in 16K Byte of EPROM having the address from FC000 to FFFFF. The system software has certain useful routines, which can be utilised by the user for developing his programs. The address of these routines are given in the appendix at the end of the manual.

PROGRAMMING EXAMPLE

The following sample programs are given here to make the user familiarise with the operation of VMC-8603.

1) Addition of two binary number of 8 byte length.
2) Find the largest number in a given string.
3) Sort a string of bytes in descending order.
4) ASCII multiplication.
5) Divide a string of unpacked ASCII digits.
6) Calculate the no. of bytes in a string of data.
7) Convert the string of data to its compliment form.
8) Move a Block of data upward.
9) Binary to gray conversion.

PROGRAM-1

TO ADD two Binary numbers each 8 Bytes long:

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0200	F8	CLC	Clear Carry Flag.
0201	B9 04 00	MOV CX,0004	Load Counter register with no. of times addition to be performed (i.e. Initialize the counter register).
0204	BE 00 03	MOV SI,0300	Load source Index Reg. with starting address of Ist Binary no. (LSBs array).

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0207	BF 08 03	MOV DI, 0308	Load Destination Index Reg. with Dest. Address (where the result of add. is to be started storing). Also it's the starting address of MSBs of array.
020A	8B 04	MOV AX,[SI]	Load Data bytes (which are in location 0300 and 0301 in 16 bit ACC. i.e. (0300) - AH (0301) - AL
020C	11 05	ADC [DI],AX	Add the contents (MS Bytes) of 0308, 0309 with the contents (LS Bytes) of 0300 + 0301 and store the result in location 0308 onwards.
020F	46	INC SI	Point at 0302 LOCN (Next relevant source LOCN).
020F	46	INC SI	
0211	47	INC DI	Point at next relevant LOCN, i.e. 0304.
0211	47	INC DI	
0212	49	DEC CX	Decrement the counter.
0213	75 F5	JNZ 020A	If not zero (i.e. CX =0000) then continue addition.
0215	F4	HLT	Else, Halt.

For example

						AFTER EXECUTION		
0300	:	01	0308	:	0A	0308	:	0B
0301	:	02	0309	:	0b	0309	:	0D
0302	:	03	030A	:	0C	030A	:	0F
0303	:	04	030B	:	0E	030B	:	12
0304	:	05	030C	:	0F	030C	:	14
0305	:	06	030D	:	10	030D	:	16
0306	:	07	030E	:	11	030E	:	18
0307	:	08	030F	:	12	030F	:	1A

PROGRAM - 2

To find the maximum no. in a given string (16 Bytes long) and store it in location 0310.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0200	BE 00 03	MOV SI, 0300	Load SI reg. with starting address of string.
0203	B9 10 00	MOV CX, 0010	Initialize Counter Reg. (with the length of string i.e. no. of bytes).
0206	B4 00	MOV AH, 00	Initialize the 8 bit Acc.
0208	3A 24	CMP AH,[SI]	The 1st data byte of the string with '00'.
020A	73 02	JAE 020E	If both bytes match (above is equal) then branch to (I).
020C	8A 24	MOV AH,[SI]	Else, move the contents of (0300) into 8 bit ACC, i.e., a real no. in AH.
020E	46	INC SI	Point at the next address of string.
020F	E0 F7	LOOP NZ [AI]	Decrement the counter value, if not zero, continue processing (searching to the Max. No. continued.)
0211	88 24	MOV[SI],AH	Max. No. in 0310 address.
0213	F4	HLT	Halt.

For Example

0300	:	01	0308	:	12
0301	:	02	0309	:	08
0302	:	03	030A	:	09
0303	:	04	030B	:	0A
0304	:	05	030C	:	0B
0305	:	06	030D	:	0E
0306	:	15	030E	:	0C
0307	:	07	030F	:	0D

AFTER EXECUTION

0310 : 15

PROGRAM - 3

To sort a string of a no. of bytes in descending order:

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0200	BE 00 03	MOV SI,0300	Initialize SI Reg. with Mem LOCN 0300.
0203	8B 1C	MOV BX, [SI]	BX has the no. of bytes (to be used for sorting) LOCNS 0300 & 0301.
0205	4B	DEC BX	Decrement the no. of bytes by one.
0206	8B 0C	MOV CX [SI]	Also CX has the no. of bytes in LOCNS 0300 and 0301.
0208	49	DEC CX	Decrement the no. of bytes by one.
0209	BE 02 03	MOV SI, 0302	Initialize SI reg. with the starting address of string (having data bytes).
020C	8A 04	MOV AL, [SI]	Move the first data byte of string into AL.
020E	46	INC SI	Point at the next bytes of the string.
020F	3A 04	COMP AL,[SI]	Compare the two bytes of string.
0211	73 06	JAE 0219	If two bytes are equal or 1st byte is above that the second Byte Branch to (1).
0213	86 04	XCHG AL, [SI]	Else
0215	4E	DEC SI	Second byte is less than first byte and swap (interchange) the two bytes.
0216	88 04	MOV [SI],AL	
0218	46	INC SI	Point at the next LOCN of the string.
0219	E2 F1	LOOP 020C	Loop if CX is not zero (i.e. continue processing till z=0)

ADDRESS	OP CODE	MNEMONIC	COMMENTS
021B	4B	DEC BX	At this juncture, first sorting will be over i.e. first no. is logically compared with the rest of the nos. For the correct sorting, all the nos. must be compared with each other logically, i.e. above processing should be carried out no. of bytes times.
021C	BE 00 03	MOV SI,0300	
021F	75 E5	JNZ 0206	
0221	F4	HLT	Halt.

For Example

						AFTER EXECUTION	
0300	:	05	i.e. 5 nos. are to be placed in decending order.	0302	:	28
0301	:	00		0303	:	25
0302	:	20		0304	:	20
0303	:	25		0305	:	15
0304	:	28		0306	:	07
0305	:	15				
0306	:	07				

PROGRAM - 4

ASCII MULTIPLICATION

To multiply an ASCII string of eight numbers by a single ASCII digit. The result is a string of unpacked BCD digits.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0200	BE 00 03	MOV SI, 0300	Load SI reg. with starting address of string.
0203	BF 08 03	MOV DI, 0308	Load DI reg. with the starting address of result LOCNS.
0206	B2 34	MOV DL, 34	Load DL with the Multiplier ASCII Digit.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0208	B9 08 00	MOV CX,0008	Load counter reg. with the no. of bytes in the string.
020B	C6 05 00	MOV [DI],00	
020E	80 E2 0F	ANDL, 0F	MS nibble of Multiplier is ze-roed.
0211	8A 04	MOV AL, [SI]	First ASCII no. of string in AL.
0213	46	INC SI	Point at the next LOCN in string (of ASCII Nos.)
0214	80 E0 0F	AND AL, 0F	MS nibble at Multiplier no gap and is also zeroed.
0217	F6 E2	MUI DL	Perform the fn. AX = AL* DL
0219	D4 0A	AAM	Perform the fn. AH=AL/0A, AL=remainder.
021B	02 05	ADD AL,[DI]	The contents of AL (remainder obtained by performing the above operation)
021D	37	AAA	Added with 00 which are in 1st Dest. LOCN. The contents of AL are unpacked Decimal no. and are stored in 1st Dest. LOCN (=0308).
021E	88 05	MOV [DI],AL	
0220	47	INC DI	Point at the next Dest. LOCN.
0221	88 25	MOV [DI],AH	Contents of AH (Quotient got in AAM operation) are moved in next best. LOCN (0309).
0223	49	DEC CX	Decrement the Counter reg.
0224	75 EB	JNZ 0211	If not zero, continue multiply and storing unpacked BCD digits, ELSE.
0226	F4	HLT	HALT.

For Example

				AFTER EXECUTION (Unpacked BCD Digits)		
0300	:	31		0308	:	04
0301	:	32		0309	:	08
0302	:	33		030A	:	02
0303	:	34		030B	:	07
0304	:	35		030C	:	01
0305	:	36		030D	:	06
0306	:	31		030E	:	06
0307	:	32		030F	:	08

PROGRAM - 5

To Divide a String of Unpacked ASCII Digits:

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0200	B2 36	MOV DL, 36	DL having the divisor, a single 8 bit ASCII Digit.
0202	BE 00 03	MOV SL, 03 00	Load SI with the starting address of ASCII string.
0205	BF 08 03	MOV DI, 0308	Load DI with the starting address of the result LOCNS.
0208	B9 08 00	MOV CX, 0008	Initialize the counter Reg. with the no. of bytes in the string.
020B	80 E2 0F	AND DL, 0F	MS nibble of DL contents is zeroed.
020E	32 E4	XOR AH,AH	Initialize the 8 bit ACC (=00)
0210	AC	LODS B	Load AL with the contents of address accessed by SI reg. and increment SI reg. i.e. point at the next address LOCN.
0211	80 E0 0F	AND AL,OF	MS nibble of AL contents is also zeroed.
0214	D5 0A	AAD	Perform the fn. AL=(AH * OA) + AL), AH = 00.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0216	F6 F2	DIV DL	Perform the fn. AD/DL AL = Quotient and AH = reminder.
0218	AA	STOS B	The contents of AL are stored in the Address pointed to by the DI reg. and next address LOCN in DI reg. is pointed (i.e. current address LOCN of DI reg. is incremented by one).
0219	E0 F5	LOOP NZ 0210	Continue dividing the unpacked ASCII digits if the contents of C are not zeroed; else.
021B	F4	HLT	Halt.

For Example

			AFTER EXECUTION		
0300	:	31	0308	:	00
0301	:	32	0309	:	02
0302	:	33	030A	:	00
0303	:	34	030B	:	05
0304	:	35	030C	:	07
0305	:	36	030D	:	06
0306	:	31	030E	:	00
0307	:	32	030F	:	02

PROGRAM - 6

To calculate the no. of bytes in a string starting from 0302 up to an identifier (data byte) placed in AL reg. The actual count will be in LOCN 0300 & 0301.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0200	BE 00 03	MOV SI, 0300	Load SI reg. with the starting Address where the result is to be stored.
0203	B9 FF FF	MOV CX, FFFF	Initialize the counter register.
0206	BF 02 03	MOV DI, 0302	Load DI reg. with the starting address of string.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0209	B0 03	MOV AL, 03	Store the identifier in ASL.
020B 020C	F2 AE AE	REPNZ SCASB	Data byte accessed by DI reg.compared with identifier without altering either of the contents, this comparing continuous with the incrementing of DI contents (Address LOCN) till the two bytes match. With each scanning the contents of CX go on decrement by one.
020D 020F	F7 D1 89 0C	NOT CX MOV [SI], CX	Z'S complemented CX & Move CX 89 OC MOV (si),CX 020F contents into 0300.
0211	F4	HLT	Halt.

For Example

Let (309) = 03 Let (0302) = 03

Then after executing the program CX = 0008 & (0300) = 08 (0301) = 00

PROGRAM - 7

A Data string of no. of bytes (to be specified in CX reg.) is located from the starting address 0300. This data string is to be converted to its equivalent 2' S complement Form and the result is to be stored from 0400 on wards.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0200	BE 00 03	MOV SI, 0300	Load SI reg., with the starting address of data string.
0203	BF 00 04	MOV DI, 0400	Load DI with the starting address of result LOCNS.

For Example

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0206	B9 10 00	MOV CX, 0010	Load CX with the no. of bytes in the string.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0209	AC	LODSB	Load AL with Data byte accessed by SI reg. and increment the address LOCN in SI reg.
020A	F6 D8	NEG AL	The contents of AL are 2's complemented.
020C	AA	STOS B	Store AL contents in LOCN pointed to by DI ref. & increment the current Location in DI reg.
020D	E0 FA	Loop NZ 0209	If CX = 0000, continue 2's complementing the data in string else;
020F	F4	HLT	Halt.

For Example

			AFTER EXECUTION		
0300	:	01	0400	:	FF
0301	:	02	0401	:	FE
0302	:	03	0402	:	FD
0303	:	04	0403	:	FC
0304	:	05	0404	:	FB
0305	:	06	0405	:	FA
0306	:	07	0406	:	F9
0307	:	08	0407	:	F8
0308	:	09	0408	:	F7
3039	:	0A	0409	:	F6
030A	:	0B	040A	:	F5
030B	:	0C	040B	:	F4
030C	:	0D	040C	:	F3
030D	:	0E	040D	:	F2
030E	:	0F	040E	:	F1
030F	:	10	040F	:	F0

PROGRAM - 8

A Block of 20 H words is located in 0300 to 031F. This block of data is to be moved upwards in memory so as to occupy 0310 to 032F. No. of characters (bytes) in the string is in CX Reg.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0200	FD	STD	Direction Flag is set in order to auto decrement the SI and DI index regs.
0201	BE 1F 03	MOV SI,031F	Load SI with the end address of Block of data.
0204	BF 2F 03	MOV DI, 032F	Load DI with the end address of Block where the Block of data is to be moved.
0207	B9 20 00	MOV CX,0020	Initialize count reg. with the no. of bytes in the string.
020A	F2 A5	REPNE:MOVSW	It transfer the words from the SI LOCNS to DI string till CS does not become zero (with the simultaneous updating of the SI & DI LOCNS).
020C	F4	HLT	

For Example

			AFTER EXECUTION			RESULT
0300	:	01	030F	:	10	0310 to 031F were found to have same contents as the LOCNS 0300 - 030F.
0301	:	02	0310	:	01	
0302	:	03	0311	:	02	
0303	:	04	0312	:	03	
0304	:	05	0313	:	04	
0305	:	06	0314	:	05	
0306	:	07	0315	:	06	
0307	:	08	0316	:	07	
0308	:	09	0317	:	08	
0309	:	0A	0318	:	09	
030A	:	0B	031A	:	0A	
030B	:	0C	031B	:	0B	
030C	:	0D	031C	:	0C	
030D	:	0E	031D	:	0D	
030E	:	0F	031D	:	0E	
			031E	:	0F	
			031F	:	10	

PROGRAM - 9

Binary to gray code conversion for number 00 to 0F using Translate instruction.

The address of the B. Number whose equivalent gray code is to be found is in SI register. The start address of the loop up table for the binary to gray code is in BX register. The result will be moved to (SI + 1) Location.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0200	BB 00 03	MOV BX, 0300	Load BX with the starting address of look up Table. Load SI with the address of Binary No. whose gray code is to be found.
0203	BE 50 01	MOV SI, 01 50	
0206	AC	LODSB	Load AL with the data byte accessed by SI reg. and increment its current address by one, i.e. SI = 0151.
0207	D7	XLAT	AL = (BX + AL) i.e. the contents of Address accessed by Bx + AL is stored in AL.
0208	88 04	MOV[SI],AL	The contents of AL are moved into LOCN pointed by SI, i.e., 0151 contains the Result.
020A	F4	HLT	Halt.

LOOK UP TABLE

0300 : 00
0301 : 01
0302 : 02
0303 : 03
0304 : 04
0305 : 05
0306 : 06
0307 : 07
0308 : 0F
0309 : 0E
030A : 0C

030B : 0D
030C : 08
030D : 09
030E : 0B
030F : OA

For Example

Let (0150)	:	05	Let (0150)	:	09
Result (0151)	:	05	Result (0151	:	0E

USE OF 8087 PROCESSOR

The following programs illustrate the use of 8087 processor.

PROBLEM: 1

Write a program to calculate N3= $\sqrt{N12 + N22}$ 32 bit integer value stored at 300 H and 304H respectively. The result should be store at 308H.

Steps required:

1) Load N1 and calculate its $N1^2$ value
2) Load N2 and calculate its $N2^2$ value
3) Add the two.
4) Take the square root and store it in 308 location.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
Step-1			
0200	9B BB 00 03	MOV BX,0300H	BX POINTS TO N1
0204	DB 07	FLD [BX]	Load N1
0206	9B DA 0F	FMUL [DX]	Give wait calculate $N1^2$
Step-2			
0209	9B DD D9	FSTP ST (1)	Load N2 in ST (1)
020C	9B BB 04 03	MOV BX, 0304	H Load Bx to point N2
0210	DB 07	FLD [Bx]	Load N2
0212	9B DA 0F	FMUL [Bx]	Get $N2^2$

ADDRESS	OP CODE	MNEMONIC	COMMENTS
Step-3			
0215	9B D8 C1	FADD ST (1)	Add the $N1^2 + N2^2$ the result is on TOP of stack.
Step-4			
0218	9B D9 FA	FSQRT	Calculate square root of $N1^2 + 2^2$
021B	9B BB 08 03	MOV BX,0308	Load BX with 0308 for storing result.
021F	DB 17	FST (BX)	Store the result (TOS)
0221	9B	FWAIT	ON 0308.
0222	CC	INT 3	Give Break Point.

Steps required for running the program:

1) Enter the program from 0000 : 0200 location.
2) Set up the data in memory location 300 and 304.

N1	0300	:	09	00	00	00
N2	0304	:	04	00	00	00

3) Run the program by GO Command and see the result as follows:

N3	0308	:	0A	00	00	00

Problem

Write a program to calculate Sin (z) where Z - is defined in degrees.

Solution: Sin (z) is calculated using tangent function of 8087.

$$\tan (Z/2) = Y/X$$

$$\text{then Sin (z)} = \frac{2XY}{X+Y}$$

The FPTAN instruction has to be given data in radians. The value of degrees is to be converted in radian. On execution of FPTAN the result is return in Y/X format where X is written on top of stack and Y is the next to top of stack. Further the argument must satisfy 0< argument < P1/4. This is the reason we restrict Z to satisfy 0< Z < 90.

This restriction can be eliminated, by more computation. If Z is unrestricted, then the use of FPREM function of 8087 to reduce the argument to the required range and use the relation Sin (P1 + Z) = -Sin z

The above problem is solved by calculating Z/2, then converting the degrees into radians and then calculating the tagents. The resulting values of X and Y are used for calculating Sin (Z).

The following steps are required to implement this program.

1) Calculating the Z/2
2) Calculate the angle in radians.
3) Calculate tan (Z/2) = X/Y format.
4) Calculate Sin Z = $\frac{2XY}{X2+Y2}$
5) Get the integer value argument - 4.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
Step-1			
0200	B4 00	MOV AH, 00	Clear AH
0202	D1 E8	SHR AX, 01	Z = Z/2.
0204	A3 00 03	MOV, W ARGI,AX	Load AX in Arg.1
0207	DF 06 00 03	FILD ARGI	Loading Z\2 on TOS
Step-2			
020B	C7 06 02 03	MOVWARG 2,00B4	ARG 2 I is loaded with B4 i.e. 180° value in hex.
020F	B4 00		
0211	9B DF 06 02 03	FILD ARG 2	Insert F Wait and load AR G2.
0216	9B DE F9	FDIVRP	Wait and divide ST (O) = z/ 180.
0219	9B D9 EB	FILDP	ST (O) = PI
021C	9B DE C9	FMULRP	ST (O) = PI X Z/180 **TOTAL ANGLE IN RADI-ANS**
Step-3			
021F	9B D9 F2	FTAN	Calculate tan Z/2 as Y/X.
0222	9B D9 F0	FILD ST(O)	Duplicate X on top of stack.
Step-4			
0225	9B D8 C9	FMUL ST,ST(1)	ST (O) =X * X
0228	9B D9 C2	FILD ST (2)	ST (O) = Y.
022B	9B D8 CB	FMUL ST (O),ST 3	ST (O) = Y * Y

ADDRESS	OP CODE	MNEMONIC	COMMENTS
022E	9B DE C1	FADDP ST (1),ST(O)	YXY is popped of and top of stack ST(O) = X_2 + Y_2.
0231	9B DE F9	FDIVRP	ST (O) = x/x2 + Y2
0234	9B DE C9	FMULRP	ST (O) = XY/X2+Y2
0237	9B D9 C0	FLD ST (O)	Duplicate the top of stack. $\frac{2XY}{X2+Y2}$
023A	9B DE C1	FADDP ST(I),ST(O)	ST(O)= =Sing(Z).
Step-5			
023D	C7 06 04 03 10 27	MOV,W ARG E 2710H	Load ARG3 with 10,000 value.
0243	9B DF 06 04 03	FILD ARG 3	
0248	9B DE C9	FMURP	ST (0)=10,000*SIN(Z)
024B	9B D9 FC	FRND INT	Round the integer.
024E	9B DF 36 06 03	FSTP ARG4	Store it in BCD form.
0253	9B	FWAIT	
0254	A1 06 03	MOV,AX,ARG4	The result is stored at 0306 and Ax. Go to monitor.
0257	CC	CC	Go to monitor.

In the addition to above, the stack operation on each instruction execution is shown. This will help to understand the program in details. This way of writing will help user to gain confidence in writing the program for 8087.

Steps required to implement the program.

1) Enter the above program.
2) Do not use any memory location from 0300 to 0308 as they are required for implementing the program.
3) Enter the value of angle in AL, say for calculating Sin 30, the AL should be 1EH.
4) Execute the program and observe the value in AX, the result will be AX=5000H. The output in packed BCD form is available in the register AX. A decimal point is to be assumed before the first BCD digit.

5) Execute the program for various values of Sin Z

Input (AL)	Output (AX)	Sin Z
10	2756	Sin 16
2E	7193	Sin 46
3C	8660	Sin 60
56	9976	Sin 86

The stack operation on execution of each instruction.

1) FLDI ARGI

ST	(0)	—>	————
			ARGI z
			————

2) FILD ARG2

ST	(0)	——	ARG 180°
ST	(1)	——	ARGI Z

3) FDIVRP

ST	(0)	——	Z/180

4) FLDPI

ST	(0)	—>	PI
ST	(1)		Z/180

5) FMULRP

ST	(0)	——	PI * Z/180
ST	(1)		Z/180

6) FPTAN

ST	(0)	——	X
ST	(1)		Y
ST	(2)		Z/180

7) FLX ST (0)

ST	(0)	X
ST	(1)	X
ST	(2)	Y
ST	(3)	Z/180

8) FMUL ST (0), ST (1)

ST	(0)	X^2
ST	(1)	X
ST	(2)	Y
ST	(3)	Z/180

9) FLD ST (2)

ST	(0)	Y
ST	(1)	X^2
ST	(2)	X
ST	(3)	Y
ST	(4)	Z/180

10) FLD ST (0), ST (3)

ST	(0)	X^2
ST	(1)	X^2
ST	(2)	X
ST	(3)	Y
ST	(4)	Z/180

11) FADDP ST (1) ST (0)

ST	(0)	$X^2 + Y^2$
ST	(1)	X
ST	(2)	Y
ST	(3)	Z/180

12) FDIVRP

ST	(0)	$X/X^2 + Y^2$
ST	(1)	Y
ST	(2)	Z/180

13) FMULRP

ST	(0)	$XY/X^2 + Y^2$
ST	(1)	Z/180

14) FLD ST (0)

ST	(0)	$XY/X^2 + Y^2$
ST	(1)	$XY/X^2 + Y^2$
ST	(2)	Z/180

15) FADD PST (1), ST (0)

ST	(0)	$2XY/X^2 + Y^2$
ST	(0)	=SIN (Z)

16) FLDI ARG 3

ST	(0)	10,000
ST	(1)	SIN(Z)

17) FMUR P

ST	(0)	10,000 X SIN Z

18) FRNDINT

ST	(0)	Rounding the integer

19) FSTP AR G 4
The argument has the result.

PROGRAM - 10

This is a sample program for Port A of 8255-I generating square wave output at Connector C10.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0200	BA FE FF	MOV DX,FFFE	;initialization of CWR
0203	B0 80	MOV AL,80	
0205	EE	OUT DX,AL	
0206	BA F8 FF	MOV DX,FFF8	;selection of Port A
0209	B0 55	MOV AL,55	
020B	EE	OUT DX,AL	;out data 55 at Port A
020C	B0 AA	MOV AL,AA	
020E	EE	OUT DX,AL	;out data AA at Port A
020F	EB F8	JMP 0209	

PROGRAM - 11

This is a sample program for 8259 interrupt '4' generating square wave output at Connector J5 (Port A-Pin No.21 & 22) by giving GND at Pin No.6 of Connector J13.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0200	BA C8 FF	MOV DX,FFC8	;selection of data word
0203	B0 17	MOV AL,17	
0205	EE	OUT DX,AL	
0206	BA CA FF	MOV DX,FFCA	;selection of command word
0209	B0 C0	MOV AL,C0	
020B	EE	OUT DX,AL	
020C	B0 03	MOV AL,03	
020E	EE	OUT DX,AL	;selection of interrupt 4
020F	B0 FE	MOV AL,FE	
0211	EE	OUT DX,AL	
0212	FB	STI	;set interrupt
0213	EB EE	JMP 0203	
0310	00 04		
0312	00 00		;interrupt address at 0000 : 0400
0400	BA FE FF	MOV DX,FFFE	;initialization of CWR
0403	B0 80	MOV AL,80	

```
0405    EE          OUT DX,AL
0406    BA F8 FF    MOV DX,FFF8   ;selection of Port A
0409    B0 55       MOV AL,55
040B    EE          OUT DX,AL     ;out data 55
040C    B0 AA       MOV AL,AA
040E    EE          OUT DX,AL     ;out data AA
040F    EB F8       JMP 0409
```

PROGRAM - 12

This is a sample program for 8253 for generating square wave output at Counter/ Timer 0 of Connector C5.

```
0200    BA DE FF    MOV DX,FFDE   ;initialisation of CWR
0203    B0 36       MOV AL,36
0205    EE          OUT DX,AL
0206    BA D8 FF    MOV DX,FFD8   ;selection of Counter/Timer 0
0209    B0 FF       MOV AL,FF
020B    EE          OUT DX,AL     ;out data 55
020C    B0 00       MOV AL,00
020E    EE          OUT DX,AL     ;out data 00
020F    EB F8       JMP 0209
```

CONNECTOR DETAILS

8

INTRODUCTION

The details of various connectors and jumpers on the board of VMC-8603 are as follows:

DETAILS OF CONNECTOR J1

PIN	SIGNAL
1	+5VDC
2	+5VDC
3	GND
4	GND
5	LD3
6	LD7
7	LD2
8	LD6
9	LD1
10	LD5
11	LD0
12	LD4
13	A8
14	A16
15	A7
16	A15
17	A6
18	A14
19	A5
20	A13
21	A4
22	A12
23	A3
24	A11
25	A2

PIN	SIGNAL
26	A10
27	A1
28	A9
29	$\overline{\text{L1WR}}$
30	$\overline{\text{LIRD}}$
31	S2
32	S2
33	$\overline{\text{LMWR}}$
34	$\overline{\text{LMRD}}$
35	LALE
36	LALE
37	S1
38	S0
39	HLDA
40	HOLD
41	$\overline{\text{LINTA}}$
42	INTR
43	RD12
44	NMI
45	MRST
46	RESETIN
47	PCLK
48	$\overline{\text{8089CS}}$
49	NC
50	CE ELEVEN RAM

DETAILS OF CONNECTOR J2

PIN	SIGNAL
1	LD8
2	LD9
3	LD10
4	LD11
5	LD15
6	LD12
7	LD14
8	LD13
9	DRQ2
10	DRQ1
11	EXT1
12	EXT2
13	BCLK

PIN	SIGNAL
14	$\overline{\text{BUSY}}$
15	BPRN
16	$\overline{\text{BREQ}}$
17	NC
18	BRD
19	NC
20	NC
21	NC
22	NC
23	NC
24	NC
25	NC
26	NC

DETAILS OF CONNECTOR J3

PIN	SIGNAL
1	NC
2	RxD
3	TxD
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	NC

DETAILS OF CONNECTOR J4

PIN	SIGNAL
1	+5VDC
2	+5VDC
3	GND
4	GND
5	+12VDC
6	-12VDC
7	+24VDC
8	+B (NC)

DETAILS OF CONNECTOR J5

PIN	SIGNAL
1	1PC4
2	1PC5
3	1PC2
4	1PC3
5	1PC0
6	1PC1
7	1PB6
8	1PB7
9	1PB4
10	1PB5
11	1PB2
12	1PB3
13	1PB0

PIN	SIGNAL
14	1PB1
15	1PA6
16	1PA7
17	1PA4
18	1PA5
19	1PA2
20	1PA3
21	1PA0
22	1PA1
23	1PC6
24	1PC7
25	GND
26	GND

DETAILS OF CONNECTOR J6

PIN	SIGNAL
1	2PC4
2	2PC5
3	2PC2
4	2PC3
5	2PC0
6	2PC1
7	2PB6
8	2PB7
9	2PB4
10	2PB5
11	2PB2
12	2PB3
13	2PB0

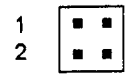
PIN	SIGNAL
14	2PB1
15	2PA6
16	2PA7
17	2PA4
18	2PA5
19	2PA2
20	2PA3
21	2PA0
22	2PA1
23	2PC6
24	2PC7
25	GND
26	GND

DETAILS OF CONNECTOR J7

PIN	SIGNAL
1	3PC4
2	3PC5
3	3PC2
4	3PC3
5	3PC0
6	3PC1
7	3PB6
8	3PB7
9	3PB4
10	3PB5
11	3PB2
12	3PB3
13	3PB0

PIN	SIGNAL
14	3PB1
15	3PA6
16	3PA7
17	3PA4
18	3PA5
19	3PA2
20	3PA3
21	3PA0
22	3PA1
23	3PC6
24	3PC7
25	GND
26	GND

DETAILS OF JUMPER JP1



1E - Pin 8 of 8284
2E - Pin 2 of 8284
1F - PCLK
2F - PCLK

DETAILS OF JUMPER JP2

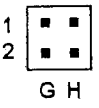
1 - +24VP
2 - +24VDC
3 - +21VP

DETAILS OF CONNECTOR J13

(Only for VMC-8603A)

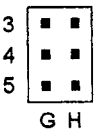
PIN	SIGNAL
1	GND
2	Vcc
3	IRQ7
4	IRQ6
5	IRQ5
6	IRQ4
7	IRQ3
8	IRQ2
9	IRQ1
10	IRQ0

DETAILS OF JUMPER JP3



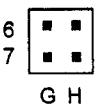
1G - Pin 28 of MEM2, MEM4
2G - RAMCBK
1H - A14
2H - NC

DETAILS OF JUMPER JP4



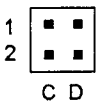
3G - NC
4G - RAMBK
5G - NC
3H - RAMCBK
4H - Pin 30 of MEM2, MEM4
5H - +5VDC

DETAILS OF JUMPER JP5



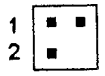
6G - RAMBK
7G - Pin 32 of MEM2, MEM4
6H - NC
7H - +5VDC

DETAILS OF JUMPER JP6



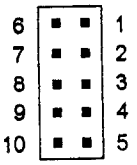
1C - BPRN
2C - BPRN
1D - GND
2D - NC

DETAILS OF JUMPER JP7



1A - Pin 30 of MEM1, MEM3
2A - A18
1B - +5VDC

DETAILS OF JUMPER JP8



6J - OUT2
7J - GATE2
8J - CLK1
9J - GATE1
10J - OUT1
1K - CLK2
2K - NC
3K - CLK0
4K - OUT0
5K - GATE0

DETAILS OF JUMPER JP9

1 - +5VDC
2 - Pin 3 of MEM1, MEM3
3 - A16

ASSEMBLER/DISASSEMBLER 9

INSTALLATION

The following steps are to be followed to work on 'ASSEMBLER/DISASSEMBLER' [from now-on abbreviated as 'ASS/DIS-BLER']:

a) Make sure that the VMC-8603 Microprocessor Training Kit has ASS/DIS-BLER PROM.

b) Connect +5V, -12V and +12V Supply to VMC-8603.

c) Set the BAUD RATE SWITCH of the main port as per your CRT-TERMINAL or IBM-PC, connected through a RS-232C port.

NOTE: Baud rate and other transmission parameters of both the kit and CRT-TERMINAL must be same for perfect communication.

The baud rate selection at VMC-8603 is done by switching-on one of the switch of main-port DIP Switch.

d) To select a desired BAUD RATE the DIP SWITCH CONFIGURATION is given as follows:

SWITCH POSITION	BAUD RATE
1 - ON	9600
2 - ON	4800
3 - ON	2400
4 - ON	1200
5 - ON	600
6 - ON	300
7 - ON	150
8 - ON	75
9 - ON	110

The other Transmission Parameters to be selected are as follows:

(1) 8 - BIT

(2) 1 - STOP BIT

(3) NO PARITY

(4) CRT/TTY switch on VMC-8603 should be on "CRT's side".

e) Connect VMC-8603 kit to CRT-Terminal through a cable having a 3 pin male connector on one side and 25 pin D type male/female connector on the other side. The connections of these two connectors are as follows:

Pin Nos. Connector	25 pin D type Connector	3 pin
1 & 7	GND	GND
2	Tx	Rx
3	Rx	Tx
4 & 5	SHORT	

f) Switch ON the VMC-8603 and CRT-Terminal and set the Transmission Parameters.

Here make sure that both CRT-Terminal and VMC-8603 have the same Transmission Parameters.

g) Press CRT/NXT Key of the VMC-8603. A message:

"VMC-8086/88M MONITOR, V1.2"

will come on the screen.

h) To come to Assembler/Disassembler mode, press G - F800:00 <CR>. A message:

Vinytics 8086/8088 Assembler Ver 1.0
By - Kishore.....
Copyrights Vinytics 1986
0600:000

will come on the screen.

i) Now one can enter/display 8086/8088 Assembler Programmes.

COMMANDS

This assembler is a Single Pass Kit-Assembler specially developed for Vinytics. This Assembler accepts a free format Intel 8086 Assembler Language from Terminal connected to 8086 Kit through Key-Board.

All the instructions of 8086/8088 Microprocessor have same format as Intel's except for some Pseudo Opcodes, specially introduced in this Single Pass Assembler. Following are the total Pseudo commands available.

***ORG Segment:Offset** ..... This command changes segment and offset address

Example

0600:0000 ORG 0000:200
0000:0200 — This command changes the segment and offset address to 0000:0200

***ORG Offset** ..... This command changes offset address.

Example

0000:0200 ORG 300 — This command changes the offset address to 0200

***DIS** ..... This command disassembles next 10 lines of the programme as per offset address.

Example

DIS — This command disassembles net 10 lines of the programme as per current segment and offset address.

To have a more detailed account of the action of this command refer to the Programmer-2 of the User's Manual, wherein it was disassembled as:

```
0000:0200              DIS
0000:0200  BE0003      MOV     CX,0010
0000:0203  B91000      MOV     CX,0010
0000:0206  B400        MOV     AH,00
0000:0208  3A24        CMP     AH,[SI]
0000:020A  7302        JNC     020E
0000:020C  8A24        MOV     AH,[SI]
0000:020E  46          INC     SI
0000:020F  E0F7        LOOPNZ  0208
0000:0211  8824        MOV     [SI],AH
0000:0213  F4          HLT
```

***DIS OFFSET** ..... This command will disassemble next 10 lines as per the Offset Address given.

To have a more detailed account of the action of this command refer to the Programme-1 of the User's Manual wherein it was disassembled as:

```
0000:0222              DIS     207
0000:0207  BF0803      MOV     DI,0308
0000:020A  8B04        MOV     AX,[SI]
0000:020C  1105        ADC     [DI],AX
0000:020E  46          INC     SI
0000:020F  46          INC     SI
0000:0210  47          INC     DI
0000:0211  47          INC     DI
0000:0212  49          DEC     CX
0000:0213  75F5        JNZ     020A
0000:0215  F4          HLT
```

CLEAR from, to..... This command fills NOP instructions (Offset addresses ofcourse!) in specified area.

LABEL..... In all Jump, Call and other memory reference instructions, one has to define the Address by a Label of maximum four characters, starting with an alphabet.

The following examples give you an idea of how one has to define the LABEL while entering the programme. It is important to note that this ASS/DIS-BLER is a Single Line Assembler, so one has to predefine the LABEL WITH its Address before using it.

Example

* A LABEL at a particular address as:

```
0000:0206      SER:                    ;here, SER label is defined
0000:0206              MOV     AX,DX   ;at the address: 0206
```

* One can do labelling in the following ways also:

```
0000:0213              ORG     0219    ;to define LABEL at the
0000:0219      SER3:                   ;Offset address 0219 from
0000:0219              ORG     213     ;the offset address 0213
0000:0213              JNZ     SER3
```

SHOW ..... This command will show still undefined labels in your assembly input.

PURGE..... This command clears current symbol-table

END ..... This command ends the Assembler input and will branch back to monitor commands mode.

***DW Value** ..... This command stores 16 bit value at Current Address.

***DW Offset symbol** ..... This command stores 16 bit value at the given offset symbol at the current address.

***DB Value** ..... This command stores 8 bit value at current address.

***DB LOW Value** ..... These commands store 8 bit

***DB HIGH Value** ..... MSB/LSB Byte as the case may be.

Also please refer to the examples as given in the next section of this manual, as per the programmes given in the user's manual of Vinytics VMC-8603 microprocessor training kit, for different commands of Assembler/Disassembler programmes.

EDITING COMMANDS

This Assembler provides editing facilities on previous line, so that small errors can be corrected with little efforts. The Assembler accepts Free Format Input and all with both LOWER and UPPER case letters to be inputted.

Only control characters are set for editing purpose as the Terminal connected to the kit may have different escape sequence translation in their own computer or special settings to transmit control characters for a proper escape sequence.

Following are the Control Characters for editing:

^B ..... Copies rest of the line and cursor set to end of line

^D ..... Inserts one character in current line keeping last line intact.

^E ..... Deletes current character from last line

^H ..... Moves cursor backwards

<CR> ..... Sends the line to Assembler

x ..... Any other character other than above will replace the character of the last line being edited.

```
VMC-8603 MONITOR V1.2
G 0000-BC F800:00      ;'G' is GO command & enter F800:00
                       ;to go into ASS/DIS-BLER

Vinytics 8086/8088 Assembler Ver1.0
By - Kishore
Copyrights Vinytics 1986

            ;THIS IS THE EXAMPLE OF ENTERING PROGRAMME USING
            ;"VINYTICS 8086/8088 ASS/DIS-BLER" AS GIVEN IN THE
            ;PROGRAMME-1 OF THIS MANUAL

0600:0000           ORG  0000:200     ;to define STARTING ADDRESS
                                      ORIGIN
0000:0200           CLC
0000:0201           MOV  CX,0004
0000:0204           MOV  SI,0300
0000:0207           MOV  DI,0308
0000:020A   START:                    ;to define LABEL 'START'
0000:020A           MOV  AX,[SI]
0000:020C           ADC  [DI],AX
0000:020E           INC  SI
0000:020F           INC  SI
0000:0210           INC  DI
0000:0211           INC  DI
0000:0212           DEC  CX
0000:0213           JNZ  START
0000:0215           HLT
0000:0216           ORG  0300         ;to change the origin to 300H for
                                      ;DATA ENTRY to enter data us-
                                      ing
                                      ;'DB' command
0000:0300           DB   01
0000:0301           DB   02
0000:0302           DB   03
0000:0303           DB   04
0000:0304           DB   05
0000:0305           DB   06
0000:0306           DB   07
0000:0307           DB   08
0000:0308           DB   0A
```

```
0000:0309              DB      0B
0000:030A              DB      0C
0000:030B              DB      0E
0000:030C              DB      0F
0000:030D              DB      10
0000:030E              DB       11
0000:030F              DB      12
0000:0310              ORG     0000:200    ;change ORIGIN to STARTING
                                           ;ADDRESS
0000:0200              DIS                 ;to display the ENTERED
                                           ;PROGRAMME
0000:0200  F8          CLC                 ;starting of the Programme
0000:0201  B90400      MOV     CX,0004
0000:0204  BE0003      MOV     SI,0300
0000:0207  BF0803      MOV     DI,0308
0000:020A  8B04        MOV     AX,[SI]
0000:020C  1105        ADC     [DI],AX
0000:020E  46          INC     SI
0000:020F  46          INC     SI
0000:0210  47          INC     DI
0000:0211  47          INC     DI
0000:0212  49          DEC     CX
0000:0213  75F5        JNZ     020A
0000:0215  F4          HLT                 ;end of the programme
0000:0223  END                             ;END command to EXIT
                                           ;ASS/DIS-BLER

G 0000- BC F800:0      ;'G' is GO command & enter F800:00
                       ;to go into ASS/DIS-BLER

Vinytics 8086/8088 Assembler Ver 1.0
By - Kishore.....
Copyrights vinytics 1986

            ;THIS IS THE EXAMPLE OF ENTERING PROGRAMME USING
            ;"VINYTICS 8086/8088 ASS/DIS-BLER" AS GIVEN IN THE
            ;PROGRAMM-2 OF THIS MANUAL

0600:0000       ORG  0000:200              ;to define starting address
0000:0200            MOV     SI,0300
0000:0203            MOV     CX,0010
0000:0206            MOV     AH,00
0000:0208            SER:                ;to define the Label as SER
0000:0208            CMP     AH,[SI]
0000:020A            ORG     20E
```

```
0000:020E      SER1:                          ;to define the Label as SER1
0000:020E              ORG     20A
0000:020A              JAE     SER1
0000:020C              MOV     AH,[SI]
0000:020E              INC     SI
0000:020F              LOOPNZ SER
0000:0211              MOV     [SI],AH
0000:0213              HLT
0000:0214              ORG     300             ;to change the origin to 300H for
                                               ;DATA ENTRY to enter data us-
                                               ing ;'DB' command
0000:0300              DB      01
0000:0301              DB      02
0000:0302              DB      03
0000:0303              DB      04
0000:0304              DB      05
0000:0305              DB      06
0000:0306              DB      15
0000:0307              DB      07
0000:0308              DB      12
0000:0309              DB      08
0000:030A              DB      09
0000:030B              DB      0A
0000:030C              DB      0B
0000:030D              DB      PE
0000:030E              DB      0C
0000:030F              DB      0D

0000:0310              ORG     200             ;change the ORIGIN to display
                                               the entered programme
0000:0200              DIS                     ;DIS command to display the
                                               programme
0000:0200  BE0003      MOV     SI,0300
0000:0203  B91000      MOV     CX,0010
0000:0206  B400        MOV     AH,00
0000:0208  3A24        CMP     AH,[SI]
0000:020A  7302        JNC     020E
0000:020C  8A24        MOV     AH,[SI]
0000:020E  46          INC     SI
0000:020F  E0F7        LOOPNZ 0208
0000:0211  8824        MOV     [SI],AH
0000:0213  F4          HLT

0000:0214 END;END command to EXIT ASS/DIS-BLER
```

```
VMC-8603 MONITOR V1.2
.G 0000- FF F800:00     ;'G' is GO command & enter F800:00
                        ; to go into ASS/DIS-BLER

Vinytics 8086/8088 Assembler Ver 1.0
By - Kishore.....
Copyrights vinytics 1986

            ;THIS IS THE EXAMPLE OF ENTERING PROGRAMME USING
            ;"VINYTICS 8086/8088 ASS/DIS-BLER" AS GIVEN IN THE
            ;PROGRAMME-3 OF THIS MANUAL

0600:0000           ORG     0000:200    ;to define STARTING ADDRESS
                                        ORIGIN
0000:0200           MOV     SI,0300
0000:0203           MOV     BX,[SI]
0000:0205           DEC     BX
0000:0206   SER3:                       ;to define LABEL 'SER3'
0000:0206           MOV     CX,[SI]
0000:0208           DEC     CX
0000:0209           MOV     SI,0302
0000:020C   SER2:                       ;to define LABEL 'SER2'
0000:020C           MOV     AL,[SI]
0000:020E           INC     SI
0000:020F           CMP     AL,[SI]

0000:0211           ORG     219         ;to change origin to define LA-
                                        BEL
0000:0219   SER1:                       ;to define LABEL 'SER1'
0000:0219           ORG     211         ;to change origin to enter
                                        programme
0000:0211           JAE     SER1
0000:0213           XCHG    AL,[SI]
0000:0215           DEC     SI
0000:0216           MOV     [SI],AL
0000:0218           INC     SI
0000:0219           LOOP    SER2
0000:021B           DEC     BX
0000:021C           MOV     SI,0300
0000:021F           JNZ     SER3
```

```
0000:0221                HLT
0000:0222                ORG     300        ;to change the origin to 300H for

0000:0300                DB      05         ;DATA ENTRY to enter data us-
                                            ing 'DB' command
0000:0301                DB      00
0000:0302                DB      20
0000:0303                DB      25
0000:0304                DB      28
0000:0305                DB      15
0000:0306                DB      07
0000:0307                ORG     200
0000:0200                DIS                ;to display the ENTERED
                                            ;PROGRAMME

0000:0200  BE0003        MOV     SI,0300    ;starting of programme
0000:0203  8B1C          MOV     BX,[SI]
0000:0205  4B            DEC     BX
0000:0206  8B0C          MOV     CX,[SI]
0000:0208  49            DEC     CX
0000:0209  BE0203        MOV     SI,0302
0000:020C  8A04          MOV     AL,[SI]
0000:020E  46            INC     SI
0000:020F  3A04          CMP     AL,[SI]
0000:0211  7306          JNC     0219
0000:0218                DIS

0000:0218  D40A          AAM     0A
0000:021A  0205          ADD     AL,[DI]
0000:021C  37            AAA
0000:021D  8805          MOV     [DI],AL
0000:021F  47            INC     DI
0000:0220  8825          MOV     [DI],AH
0000:0222  49            DEC     CX
0000:0223  75EC          JNZ     0211
0000:0225  F4            HLT
```

```
VMC-8603 MONITOR V1.2
.G 0000- FF F800:00     ;'G' is GO command & enter F800:00
                        ; to go into ASS/DIS-BLER

Vinytics 8086/8088 Assembler Ver 1.0
By - Kishore.....
Copyrights vinytics 1986

            ;THIS IS THE EXAMPLE OF ENTERING PROGRAMME USING
            ;"VINYTICS 8086/8088 ASS/DIS-BLER" AS GIVEN IN THE
            ;PROGRAMME-4 OF THIS MANUAL

0600:0000           ORG     0000:200
0000:0200           MOV     SI,0300
0000:0203           MOV     DI,0308
0000:0206           MOV     DL,34
0000:0208           MOV     CX,0800
0000:020B           MOV     BYTE PTR [DI],00
0000:020E           AND     DL,0F
0000:0211           SER:                ;to define LABEL 'SER'
0000:0211           MOV     AL,[SI]
0000:0213           INC     SI
0000:0214           AND     AL,0F
0000:0216           MUL     DL
0000:0218           AAM
0000:021A           ADD     AL,[DI]
0000:021C           AAA
0000:021D           MOV     [DI],AL
0000:021F           INC     DI
0000:0220           MOV     [DI],AH
0000:0222           DEC     CX
0000:0223           JNZ     SER
0000:0225           HLT
0000:0226           ORG     300         ;to change the origin to 300H for
                                        ;DATA ENTRY to enter data us-
                                        ing
                                        ;'DB' command
0000:0300           DB      31
0000:0301           DB      32
0000:0302           DB      33
0000:0303           DB      34
0000:0304           DB      35
0000:0305           DB      36
0000:0306           DB      31
0000:0307           DB      32
```

```
0000:0308              END
0000:0000              ORG     0000:200
0000:0200              DIS

0000:0200  BE0003      MOV     SI,0300
0000:0203  BF0803      MOV     DI,0308
0000:0206  B234        MOV     DL,34
0000:0208  B90008      MOV     CX,0800
0000:020B  C60500      MOV     BYTE PTR [DI],00
0000:020E  80E20F      AND     DL,0F
0000:0211  8A04        MOV     AL,[SI]
0000:0213  46          INC     SI
0000:0214  240F        AND     AL,0F
0000:0216  F6E2        MUL     DL

0000:0218  DIS

0000:0218  D40A        AAM     0A
0000:021A  0205        ADD     AL,[DI]
0000:021C  37          AAA
0000:021D  8805        MOV     [DI],AL
0000:021F  47          INC     DI
0000:0220  8825        MOV     [DI],AH
0000:0222  49          DEC     CX
0000:0223  75EC        JNZ     0211
0000:0225  F4


VMC-8603 MONITOR V1.2
.G 0000- E9 F800:00    ;'G' is GO command & enter F800:00
                       ; to go into ASS/DIS-BLER

Vinytics 8086/8088 Assembler Ver 1.0
By - Kishore.....
Copyrights vinytics 1986

            ;THIS IS THE EXAMPLE OF ENTERING PROGRAMME USING
            ;"VINYTICS 8086/8088 ASS/DIS-BLER" AS GIVEN IN THE
            ;PROGRAMME-5 OF THIS MANUAL

0600:0000              ORG     0000:200    ;to define STARTING ADDRESS
                                           ;ORIGIN
0000:0200              MOV     DL,36
0000:0202              MOV     SI,0300
```

```
0000:0205           MOV     DI,0308
0000:0208           MOV     CX,0008
0000:020B           AND     DL,0F
0000:020E           XOR     AH,AH
0000:0210           SER                 ;to define LABEL 'SER'
0000:0210           LODSB
0000:0211           AND     AL,0F
0000:0213           AAD
0000:0215           DIV     DL
0000:0217           STOSB
0000:0218           LOOPNZ SER
0000:021A           HLT
0000:021B           ORG     300         ;to change the origin to 300H for
                                        ;DATA ENTRY to enter data
                                        ;using 'DB' command
0000:0300           DB      31
0000:0301           DB      32
0000:0302           DB      33
0000:0303           DB      34
0000:0304           DB      35
0000:0305           DB      36
0000:0306           DB      31
0000:0307           DB      32
0000:0308           ORG     200
0000:0200           DIS                 ;to display the ENTERED
                                        ;PROGRAMME

0000:0200 B236      MOV     DL,36
0000:0202 BE0003    MOV     SI,0300
0000:0205 BF0803    MOV     DI,0308
0000:0208 B90800    MOV     CX,0008
0000:020B 80E20F    AND     DL,0F
0000:020E 30E4      XOR     AH,AH
0000:0210 AC        LODSB
0000:0211 240F      AND     AL,0F
0000:0213 D50A      AAD     0A
0000:0215 F6F2      DIV     DL
0000:0217 DIS
0000:0217 AA        STOSB
0000:0218 E0F6      LOOPNZ 0210
0000:021A           F4      HLT
```

```
VMC-8603 MONITOR V1.2
.G 0000- FF F800:00     ;'G' is GO command & enter F800:00
                        ; to go into ASS/DIS-BLER

Vinytics 8086/8088 Assembler Ver 1.0
By - Kishore.....
Copyrights vinytics 1986

            ;THIS IS THE EXAMPLE OF ENTERING PROGRAMME USING
            ;"VINYTICS 8086/8088 ASS/DIS-BLER" AS GIVEN IN THE
            ;PROGRAMME-6 OF THIS MANUAL

0600:0000               ORG     0000:200    ;to define STARTING ADDRESS
                                            ;ORIGIN
0000:0200               MOV     SI,0300
0000:0203               MOV     CX,0FFFF
0000:0206               MOV     DI,0302
0000:0209               MOV     AL,03
0000:020B               REPNZ
0000:020C               SCASB
0000:020D               NOT     CX
0000:020F               MOV     [SI],CX
0000:0211               HLT
0000:0212               ORG     309
0000:0309               DB      03
0000:030A               ORG     200
0000:0200               DIS                 ;to change the origin to 30H for
                                            ;DATA ENTRY to enter data
                                            ;using 'DB' command
0000:0200  BE0003       MOV     SI,0300     ;starting of programme
0000:0203  B9FFFF       MOV     CX,FFFF
0000:0206  BF0203       MOV     DI,0302
0000:0209  B003         MOV     AL,03
0000:020B  F2           REPNZ
0000:020C  AE           SCASB
0000:020D  F7D1         NOT     CX
0000:020F  890C         MOV     [SI],CX
0000:0211  F4           HLT                 ;end of the programme
```

VMC-8603 MONITOR V1.2

```
.G 0000- BC F800:00     ;'G' is GO command & enter F800:00
                        ; to go into ASS/DIS-BLER
```

Vinytics 8086/8088 Assembler Ver 1.0
By - Kishore.....
Copyrights vinytics 1986

```
                ;THIS IS THE EXAMPLE OF ENTERING PROGRAMME USING
                ;"VINYTICS 8086/8088 ASS/DIS-BLER" AS GIVEN IN THE
                ;PROGRAMME-7 OF THIS MANUAL

0600:0000               ORG     0000:200    ;to define STARTING ADDRESS
                                            ;ORIGIN
0600:0200               MOV     SI,
0600:0200               ORG     0000:200
0000:0200               MOV     SI,300
0000:0203               MOV     DI,400
0000:0206               MOV     CX,0010
0000:0209       SER:                        ;to define LABEL 'SER'
0000:0209               LODSB
0000:020A               NEG     AL
0000:020C               STOSB
0000:020D               LOOPNZ SER
0000:020F               HLT
0000:0210               ORG     300         ;to change the origin to 300H for
                                            ;DATA ENTRY to enter data us-
                                            ing ;'DB' command
0000:0300               DB      01
0000:0301               DB      02
0000:0302               DB      03
0000:0303               DB      04
0000:0304               DB      05
0000:0305               DB      06
0000:0306               DB      07
0000:0307               DB      08
0000:0308               DB      09
0000:0309               DB      0A
0000:030A               DB      0B
0000:030B               DB      0C
0000:030C               DB      0D
0000:030D               DB      0E
```

```
0000:030E              DB      0F
0000:030F              DB      10
0000:0310              ORG     200
0000:0200              DIS                     ;to display the ENTERED
                                               ;PROGRAMME
0000:0200  BE0003      MOV     SI,0300         ;start of programme
0000:0203  BE0004      MOV     DI,0400
0000:0206  B91000      MOV     CX,0010
0000:0209  AC          LODSB
0000:020A  F6D8        NEG
0000:020C  AA          STOSB
0000:020D  E0FA        LOOPNZ 0209
0000:020F  F4          HLT                     ;end of the programme
```

```
VMC-8603 MONITOR V1.2
.G 0000- BC F800:00    ;'G' is GO command & enter F800:00
                       ; to go into ASS/DIS-BLER
```

```
Vinytics 8086/8088 Assembler Ver 1.0
By - Kishore.....
Copyrights vinytics 1986
```

```
        ;THIS IS THE EXAMPLE OF ENTERING PROGRAMME USING
        ;"VINYTICS 8086/8088 ASS/DIS-BLER" AS GIVEN IN THE
        ;PROGRAMME-8 OF THIS MANUAL

0600:0000              ORG     0000:200        ;to define STARTING ADDRESS
                                               ;ORIGIN
0000:0200              STD
0000:0201              MOV     SI,031F
0000:0204              MOV     DI,032F
0000:0207              MOV     CX,0020
0000:020A              REPNZ
0000:020B              MOVSW
0000:020C              HLT
0000:020D              ORG     300             ;to change the origin to 300H for
                                               ;DATA ENTRY to enter dat us-
                                               ing ;'DB' command
0000:0300              DB      01
0000:0301              DB      02
0000:0302              DB      03
0000:0303              DB      04
0000:0304              DB      05
```

```
0000:0305              DB      06
0000:0306              DB      07
0000:0307              DB      08
0000:0308              DB      09
0000:0309              DB      0A
0000:030A              DB      0B
0000:030B              DB      0C
0000:030C              DB      0D
0000:030D              DB      0E
0000:030E              DB      0F
0000:030F              ORG     200
0000:0200              DIS                     ;to display the ENTERED
                                               ;PROGRAMME

0000:0200              FD      STD             ;starting of programme
0000:0201  BE1F03      MOV     SI,031F
0000:0204  BF2F03      MOV     DI,032F
0000:0207  B92000      MOV     CX,0020
0000:020A  F2          REPNZ
0000:020B  A5          MOVSW
0000:020C  F4          HLT                     ;end of the programme
```

```
VMC-8603 MONITOR V1.2
.G 0000- BC F800:00    ;'G' is GO command & enter F800:00
                       ; to go into ASS/DIS-BLER

Vinytics 8086/8088 Assembler Ver 1.0
By - Kishore.....
Copyrights vinytics 1986

            ;THIS IS THE EXAMPLE OF ENTERING PROGRAMME USING
            ;"VINYTICS 8086/8088 ASS/DIS-BLER" AS GIVEN IN THE
            ;PROGRAMME-9 OF THIS MANUAL

0600:0000              ORG     0000:200
0000:0200              MOV     BX,0300
0000:0203              MOV     SI,0150
0000:0206              LODSB
0000:0207              XLAT
0000:0208              MOV     [SI],AL
0000:020A              HLT
```

```
0000:020B              ORG   300      ;to change the origin to 300H for
                                      ;DATA ENTRY to enter data us-
                                      ing ;'DB' command
0000:0300              DB    00
0000:0301              DB    01
0000:0302              DB    02
0000:0303              DB    03
0000:0304              DB    04
0000:0305              DB    05
0000:0306              DB    06
0000:0307              DB    07
0000:0308              DB    0F
0000:0309              DB    0E
0000:030A              DB    0C
0000:030B              DB    0D
0000:030C              DB    08
0000:030D              DB    09
0000:030E              DB    0B
0000:030F              DB    0A
0000:0310              ORG   200
0000:0200              DIS           ;to display the ENTERED
                                      ;PROGRAMME

0000:0200  BB0003      MOV   BX,0300 ;start of programme
0000:0203  BE5001      MOV   SI,0150
0000:0206  AC          LODSB
0000:0207  D7          XLAT
0000:0208  8804        MOV   [SI],AL
0000:020A  F4          HLT           ;end of programme
```

PRINTER INTERFACE (OPTIONAL) 8603

The printer interface is provided using 8255-III. The 8255 is configured by case is follows.

25 PIN TYPE (D TYPE FEMALE)	SIGNAL	PORT
1	DATA STORE	PB0
2	DATA 0	PA0
3	DATA 1	PA1
4	DATA2	PA2
5	DATA3	PA3
6	DATA4	PA4
7	DATA5	PA5
8	DATA6	PA6
9	DATA7	PA7
10	ACK	PC4
11	BUSY	PC5
12	P END	PC6
13	SELECT	PC7
14	AUTO FEED	PB1
16	INIT. PRN	PB2
17	SELECT INPUT	PB3
18-25	GND	

NOTE: IBM PC PRINTER CABLE CAN BE DIRECTLY USED.

EXAMPLE FOR DOT MATRIX PRINTER

This routine is used to print certain message stored at message pointer using a Dot Matrix Printer. Control and data lines of printer are connected through 8255. One has to store in CX register the no. of charecter to be printed.

PROGRAM

```
400   BA E6 FF          MOV   DX,FFE6   ;For 8255 initialization
403   B0 89             MOV   AL,89
405   EE                OUT   DX,AL
406   B3 40             MOV   BL,40     ;This printer initializa-
408   9A 50 04 00 00    CALL  0:0450    tion routine is needed
                                        for some printers.
40D   BE A0 04          MOV   SI,04A0   ;Location where string
                                        is stored (message
                                        pointer)
410   B9 14 00          MOV   CX,0014   ;No. of character to be
                                        printed
413   8A 1C       LOOP: MOV   BL,[SI]
415   56                PUSH  SI
416   51                PUSH  CX
417   9A 50 04 00 00    CALL  00:0450   ;Printer routine
41C   59                POP   CX
41D   5E                POP   SI
41E   46                INC   SI
41F   49                DEC   CX
420   75 F1             JNZ   LOOP
422   90                NOP
423   90                NOP
424   90                NOP
425   90                NOP
426   90                NOP
427   B3 0A             MOV   BL,0A     ;For line feed
429   9A 50 04 00 00    CALL  00:0450
42E   B3 0D             MOV   BL,0D     ;For cariage return
430   9A 50 04 00 00    CALL  00:450
435   CC                              ;Breakpoint
```

PRINTER ROUTINE

;This routine is used to out the data which is to be printed.

```
450   BA E2 FF          MOV   DX,FFE2   ;AUTOPE,      Data
                                        Strobe, INIT high
453   B0 0F             MOV   AL,0F
455   EE                OUT   DX,AL
456   BA E4 FF  LOOP1:  MOV   DX,FFE4   ;Read busy low
459   EC                IN    AL,DX
45A   F6 D0             NOT   AL
45C   24 20             AND   AL,20
45E   74 F6             JZ    LOOP1:
460   88 D8             MOV   AL,BL     ;OUT data for printing
462   BA E0 FF          MOV   DX,FFE0
465   EE                OUT   DX,AL
466   90                NOP
467   90                NOP
468   90                NOP
469   90                NOP
46A   90                NOP
46B   90                NOP
46C   BA E2 FF          MOV   DX,FFE2   ;Make strobe low
46F   B0 0E             MOV   AL,0E
471   EE                OUT   DX,AL
472   90                NOP
473   90                NOP
474   90                NOP
475   90                NOP
476   90                NOP
477   90                NOP
478   B0 0F             MOV   AL,0F     ;Make strobe high
47A   EE                OUT   DX,AL
47B   B9 01 03          MOV   CX,0301   ;Delay
47E   49          LOOP2:DEC   CX
47F   75 FD             JNZ   LOOP2
481   CB                RET
```

MESSAGE POINTER:

	CODE	CHARACTER
4A0	56	V
4A1	49	I
4A2	4E	N
4A3	59	Y
4A4	54	T
4A5	49	I
4A6	43	C
4A7	53	S
4A8	20	SPACE
4A9	50	P
4AA	45	E
4AB	52	R
4AC	49	I
4AD	50	P
4AE	48	H
4AF	45	E
4B0	52	R
4B1	41	A
4B2	4C	L
4B3	20	SPACE