M.Sc. (Electronics)

Semester- II

Microprocessors and Microcontrollers (Core)

List Of Experiments

1. To ADD two Binary numbers each 8 Bytes long.
2. To find the maximum no. in a given string (16 bytes long) and store it in location 0510.
3. To sort a string of a no. of bytes in descending order.
4. To multiply an ASCII string of eight numbers by a single ASCII digit. The result is a string of unpacked BCD digits.
5. To Divide a string of unpacked ASCII digits.
6. A data string of no. of bytes (to be specified in CX reg.) is located from the starting address 0500. This data string is to be converted to its equivalent 2'S complement form and the result is to be stored from 0600 onwards.
7. Flashing display of superb on seven segment display on executing this program from address 2000h, "superb" mes- sage flashes on the display of the kit.
8. Addition of 2 numbers and stored result at 3012 address.
9. Subtraction of 2 numbers and stored result at 3012 address.
10. Division of 2 numbers and stored result at 3012 address.
11. Multiplecation of 2 numbers and stored result at 3012 address.

MICROPROCESSOR 8086

INTEL 8086 ASSEMBLY LANGUAGE OPCODES

43. INT - Interrupt
44. INTO - Interrupt on Overflow
45. INVD - Invalidate Cache
46. INVLPG - Invalidate Translation Look-Aside Buffer Entry
47. IRET/IRETD - Interrupt Return
48. JA/JNBE - Jump Above / Jump Not Below or Equal
49. JAE/JNB - Jump Above or Equal / Jump on Not Below
50. JB/JNAE - Jump Below / Jump Not Above or Equal
51. JBE/JNA - Jump Below or Equal / Jump Not Above
52. JC - Jump on Carry
53. JCXZ/JECXZ - Jump if Register (E)CX is Zero
54. JE/JZ - Jump Equal / Jump Zero
55. JG/JNLE - Jump Greater / Jump Not Less or Equal
56. JGE/JNL - Jump Greater or Equal / Jump Not Less
57. JL/JNGE - Jump Less / Jump Not Greater or Equal
58. JLE/JNG - Jump Less or Equal / Jump Not Greater
59. JMP - Unconditional Jump
60. JNC - Jump Not Carry
61. JNE/JNZ - Jump Not Equal / Jump Not Zero
62. JNO - Jump Not Overflow
63. JNS - Jump Not Signed
64. JNP/JPO - Jump Not Parity / Jump Parity Odd
65. JO - Jump on Overflow
66. JP/JPE - Jump on Parity / Jump on Parity Even
67. JS - Jump Signed
68. LAHF - Load Register AH From Flags
69. LAR - Load Access Rights
70. LDS - Load Pointer Using DS
71. LEA - Load Effective Address
72. LEAVE - Restore Stack for Procedure Exit
73. LES - Load Pointer Using ES
74. LFS - Load Pointer Using FS
75. LGDT - Load Global Descriptor Table
76. LIDT - Load Interrupt Descriptor Table
77. LGS - Load Pointer Using GS
78. LLDT - Load Local Descriptor Table
79. LMSW - Load Machine Status Word
80. LOCK - Lock Bus
81. LODS - Load String
82. LOOP - Decrement CX and Loop if CX Not Zero
83. LOOPE/LOOPZ - Loop While Equal / Loop While Zero
84. LOOPNZ/LOOPNE - Loop While Not Zero / Loop While Not Equal
85. LSL - Load Segment Limit
86. LSS - Load Pointer Using SS
87. LTR - Load Task Register
88. MOV - Move Byte or Word
89. MOVS - Move String
90. MOVSX - Move with Sign Extend
91. MOVZX - Move with Zero Extend
92. MUL - Unsigned Multiply

93. NEG - Two's Complement Negation
94. NOP - No Operation
95. NOT - One's Compliment Negation
96. OR - Inclusive Logical OR
97. OUT - Output Data to Port
98. OUTS - Output String to Port
99. POP - Pop Word off Stack
100. POPA/POPAD - Pop All Registers onto Stack
101. POPF/POPFD - Pop Flags off Stack
102. PUSH - Push Word onto Stack
103. PUSHA/PUSHAD - Push All Registers onto Stack
104. PUSHF/PUSHFD - Push Flags onto Stack
105. RCL - Rotate Through Carry Left
106. RCR - Rotate Through Carry Right
107. REP - Repeat String Operation
108. REPE/REPZ - Repeat Equal / Repeat Zero
109. REPNE/REPNZ - Repeat Not Equal / Repeat Not Zero
110. RET/RETF - Return From Procedure
111. ROL - Rotate Left
112. ROR - Rotate Right
113. SAHF - Store AH Register into FLAGS
114. SAL/SHL - Shift Arithmetic Left / Shift Logical Left
115. SAR - Shift Arithmetic Right
116. SBB - Subtract with Borrow
117. SCAS - Scan String
118. SETAE/SETNB - Set if Above or Equal / Set if Not Below
119. SETB/SETNAE - Set if Below / Set if Not Above or Equal
120. SETBE/SETNA - Set if Below or Equal / Set if Not Above
121. SETE/SETZ - Set if Equal / Set if Zero
122. SETNE/SETNZ - Set if Not Equal / Set if Not Zero
123. SETL/SETNGE - Set if Less / Set if Not Greater or Equal
124. SETGE/SETNL - Set if Greater or Equal / Set if Not Less
125. SETLE/SETNG - Set if Less or Equal / Set if Not greater or Equal
126. SETG/SETNLE - Set if Greater / Set if Not Less or Equal
127. SETS - Set if Signed
128. SETNS - Set if Not Signed
129. SETC - Set if Carry
130. SETNC - Set if Not Carry
131. SETO - Set if Overflow
132. SETNO - Set if Not Overflow
133. SETP/SETPE - Set if Parity / Set if Parity Even
134. SETNP/SETPO - Set if No Parity / Set if Parity Odd
135. SGDT - Store Global Descriptor Table
136. SIDT - Store Interrupt Descriptor Table
137. SHR - Shift Logical Right
138. SHLD/SHRD - Double Precision Shift
139. SLDT - Store Local Descriptor Table
140. SMSW - Store Machine Status Word
141. STC - Set Carry
142. STD - Set Direction Flag

143. STI - Set Interrupt Flag
144. STOS - Store String
145. STR - Store Task Register
146. SUB - Subtract
147. TEST - Test For Bit Pattern
148. VERR - Verify Read
149. VERW - Verify Write
150. WAIT/FWAIT - Event Wait
151. WBINVD - Write-Back and Invalidate Cache
152. XCHG - Exchange
153. XLAT/XLATB - Translate
154. XOR - Exclusive OR

BLOCK DIAGRAM OF INTEL 8086

The 8086 CPU is divided into two independent functional units:

1. Bus Interface Unit (BIU)

2. Execution Unit (EU)

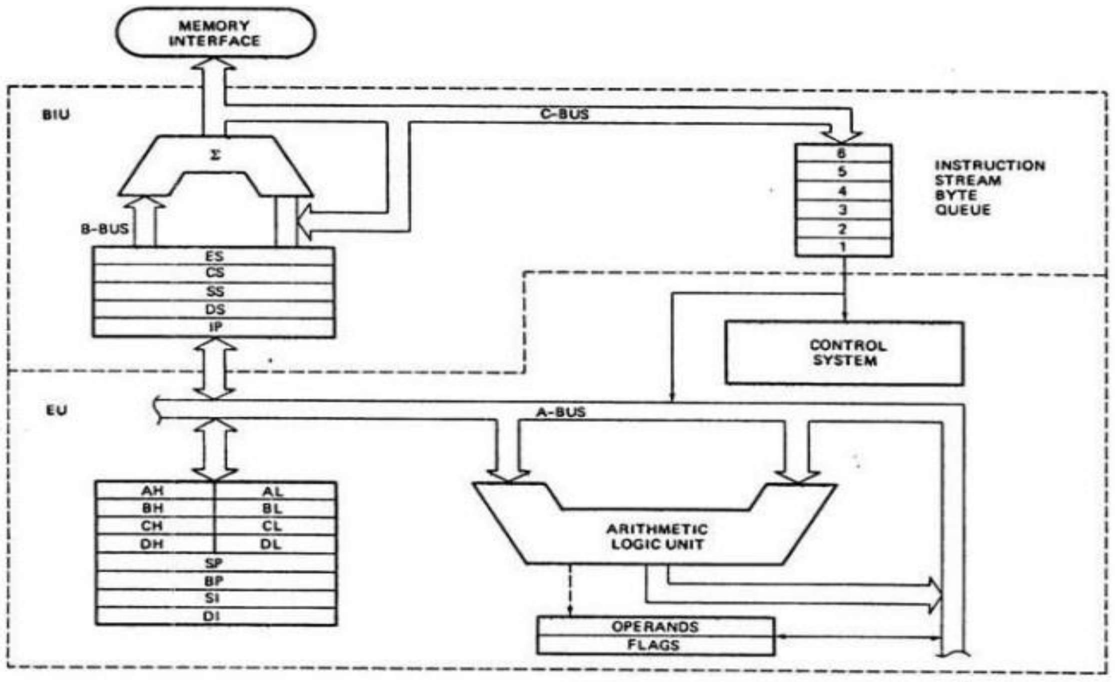


Fig. 1 Block diagram of intel 8086

Features of 8086 Microprocessor:

1. Intel 8086 was launched in 1978.

2. It was the first 16-bit microprocessor.

3. This microprocessor had major improvement over the execution speed of 8085.

4. It is available as 40-pin Dual-Inline-Package (DIP).

5. It is available in three versions:

a. 8086 (5 MHz)

b. 8086-2 (8 MHz)

c. 8086-1 (10 MHz)

6. It consists of 29,000 transistors.

Bus Interface Unit (BIU)

The function of BIU is to:

- Fetch the instruction or data from memory.
- Write the data to memory.
- Write the data to the port.
- Read data from the port.

Instruction Queue

1. To increase the execution speed, BIU fetches as many as six instruction bytes ahead to time from memory.

2. All six bytes are then held in first in first out 6 byte register called instruction queue.

3. Then all bytes have to be given to EU one by one.

4. This pre fetching operation of BIU may be in parallel with execution operation of EU, which improves the speed execution of the instruction.

Execution Unit (EU)

The functions of execution unit are:

- To tell BIU where to fetch the instructions or data from.
- To decode the instructions.
- To execute the instructions.

The EU contains the control circuitry to perform various internal operations. A decoder in EU decodes the instruction fetched memory to generate different internal or external control signals required to perform the operation. EU has 16-bit ALU, which can perform arithmetic and logical operations on 8-bit as well as 16-bit.

General Purpose Registers of 8086

These registers can be used as 8-bit registers individually or can be used as 16-bit in pair to have AX, BX, CX, and DX.

1. AX Register: AX register is also known as accumulator register that stores operands for arithmetic operation like divided, rotate.

2. BX Register: This register is mainly used as a base register. It holds the starting base location of a memory region within a data segment.

3. CX Register: It is defined as a counter. It is primarily used in loop instruction to store loop counter.

4. DX Register: DX register is used to contain I/O port address for I/O instruction.

Segment Registers

Additional registers called segment registers generate memory address when combined with other in the microprocessor. In 8086 microprocessor memory is divided into 4 segments as follow:

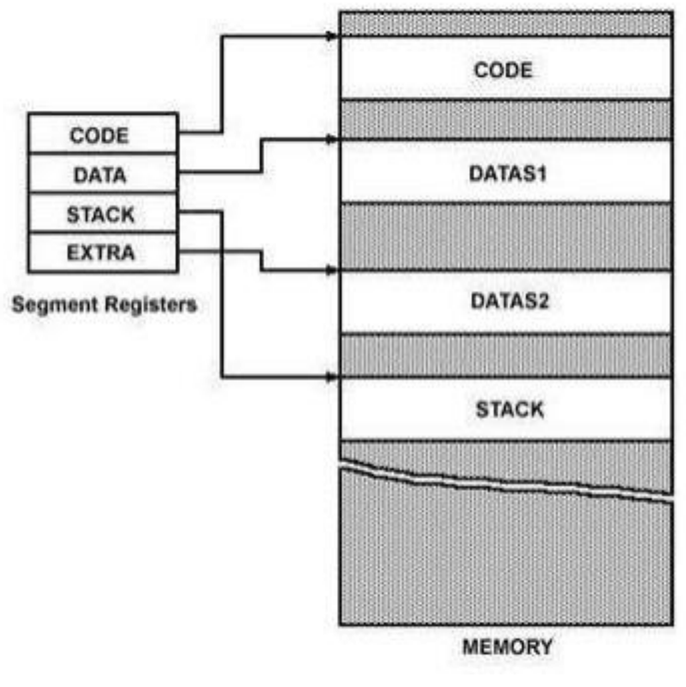


Fig.2 Memory segments of 8086

1. Code Segment (CS): The CS register is used for addressing a memory location in the Code Segment of the memory, where the executable program is stored.

2. Data Segment (DS): The DS contains most data used by program. Data are accessed in the Data Segment by an offset address or the content of other register that holds the offset address.

3. Stack Segment (SS): SS defined the area of memory used for the stack.

4. Extra Segment (ES): ES is additional data segment that is used by some of the string to hold the destination data.

Flag Registers of 8086

Flag register in EU is of 16-bit and is shown in fig. 3:

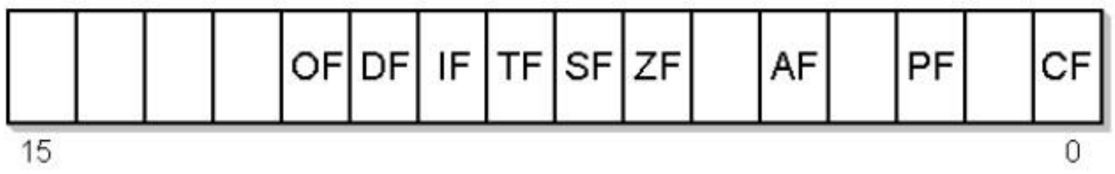


Fig.3 Flag register of 8086

Flags Register determines the current state of the processor. They are modified automatically by CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program. 8086 has 9 flags and they are divided into two categories:

1. Conditional Flags
2. Control Flags

Conditional Flags

Conditional flags represent result of last arithmetic or logical instruction executed. Conditional flags are as follows:

• **Carry Flag (CF):** This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic.

• **Auxiliary Flag (AF):** If an operation performed in ALU generates a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), the AF flag is set i.e. carry given by D3 bit to D4 is AF flag. This is not a general-purpose flag, it is used internally by the processor to perform Binary to BCD conversion.

• **Parity Flag (PF):** This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1's, the Parity Flag is set and for odd number of 1's, the Parity Flag is reset.

• **Zero Flag (ZF):** It is set; if the result of arithmetic or logical operation is zero else it is reset.

• **Sign Flag (SF):** In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.

• **Overflow Flag (OF):** It occurs when signed numbers are added or subtracted. An OF indicates that the result has exceeded the capacity of machine.

Control Flags

Control flags are set or reset deliberately to control the operations of the execution unit. Control flags are as follows:

1. Trap Flag (TP):

a. It is used for single step control.

b. It allows user to execute one instruction of a program at a time for debugging.

c. When trap flag is set, program can be run in single step mode.

2. Interrupt Flag (IF):

a. It is an interrupt enable/disable flag.

b. If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled.

c. It can be set by executing instruction sit and can be cleared by executing CLI instruction.

3. Direction Flag (DF):

a. It is used in string operation.

b. If it is set, string bytes are accessed from higher memory address to lower memory address.

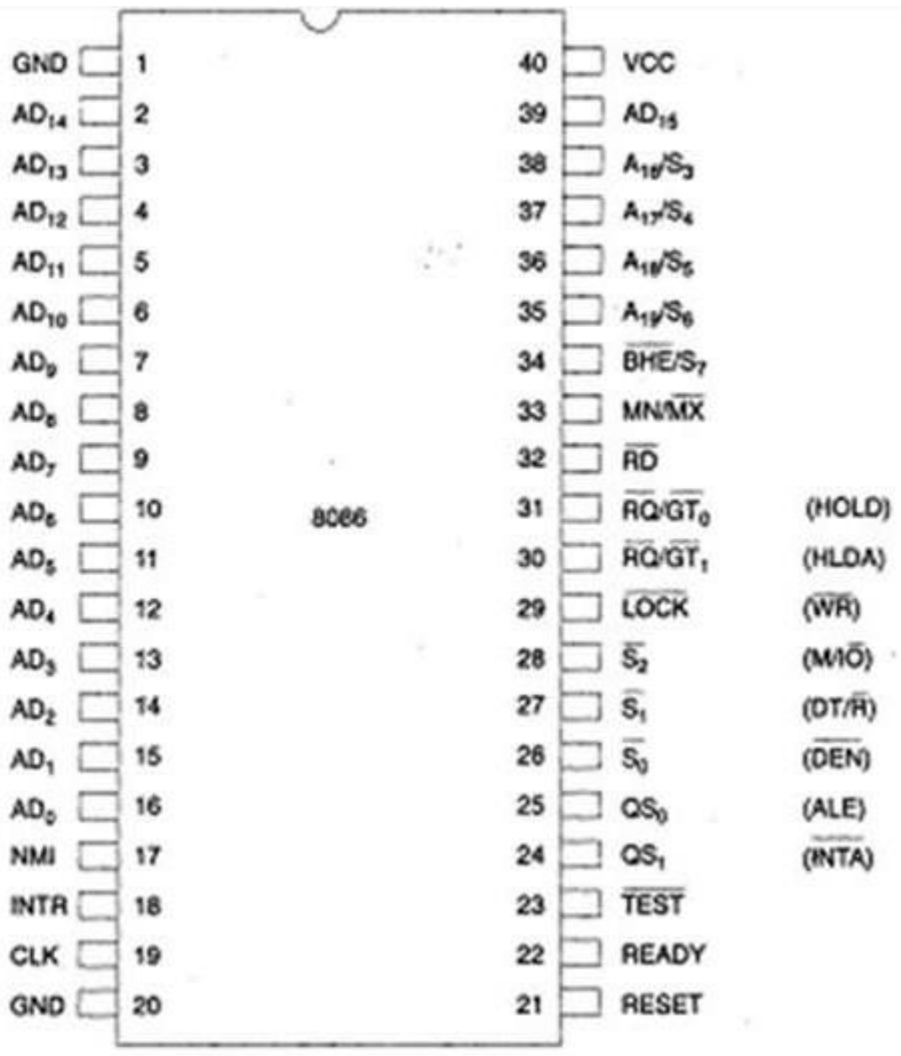
c. When it is reset, the string bytes are accessed from lower memory address to higher memory address.

Microprocessor - 8086 Pin Configuration

8086 was the first 16-bit microprocessor available in 40-pin DIP (Dual Inline Package) chip. Let us now discuss in detail the pin configuration of a 8086 Microprocessor.

8086 Pin Diagram

Here is the pin diagram of 8086 microprocessor –



Let us now discuss the signals in detail −

Power supply and frequency signals

It uses 5V DC supply at V_{CC} pin 40, and uses ground at V_{SS} pin 1 and 20 for its operation.

Clock signal

Clock signal is provided through Pin-19. It provides timing to the processor for operations. Its frequency is different for different versions, i.e. 5MHz, 8MHz and 10MHz.

Address/data bus

AD0-AD15. These are 16 address/data bus. AD0-AD7 carries low order byte data and AD8AD15 carries higher order byte data. During the first clock cycle, it carries 16-bit address and after that it carries 16-bit data.

Address/status bus

A16-A19/S3-S6. These are the 4 address/status buses. During the first clock cycle, it carries 4-bit address and later it carries status signals.

S7/BHE

BHE stands for Bus High Enable. It is available at pin 34 and used to indicate the transfer of data using data bus D8-D15. This signal is low during the first clock cycle, thereafter it is active.

Read(*$\overline{RD}$*)

It is available at pin 32 and is used to read signal for Read operation.

Ready

It is available at pin 22. It is an acknowledgement signal from I/O devices that data is transferred. It is an active high signal. When it is high, it indicates that the device is ready to transfer data. When it is low, it indicates wait state.

RESET

It is available at pin 21 and is used to restart the execution. It causes the processor to immediately terminate its present activity. This signal is active high for the first 4 clock cycles to RESET the microprocessor.

INTR

It is available at pin 18. It is an interrupt request signal, which is sampled during the last clock cycle of each instruction to determine if the processor considered this as an interrupt or not.

NMI

It stands for non-maskable interrupt and is available at pin 17. It is an edge triggered input, which causes an interrupt request to the microprocessor.

$\overline{TEST}$

This signal is like wait state and is available at pin 23. When this signal is high, then the processor has to wait for IDLE state, else the execution continues.

MN/*$\overline{MX}$*

It stands for Minimum/Maximum and is available at pin 33. It indicates what mode the processor is to operate in; when it is high, it works in the minimum mode and vice-aversa.

INTA

It is an interrupt acknowledgement signal and id available at pin 24. When the microprocessor receives this signal, it acknowledges the interrupt.

ALE

It stands for address enable latch and is available at pin 25. A positive pulse is generated each time the processor begins any operation. This signal indicates the availability of a valid address on the address/data lines.

DEN

It stands for Data Enable and is available at pin 26. It is used to enable Transreceiver 8286. The transreceiver is a device used to separate data from the address/data bus.

DT/R

It stands for Data Transmit/Receive signal and is available at pin 27. It decides the direction of data flow through the transreceiver. When it is high, data is transmitted out and vice-a-versa.

M/IO

This signal is used to distinguish between memory and I/O operations. When it is high, it indicates I/O operation and when it is low indicates the memory operation. It is available at pin 28.

WR

It stands for write signal and is available at pin 29. It is used to write the data into the memory or the output device depending on the status of M/IO signal.

HLDA

It stands for Hold Acknowledgement signal and is available at pin 30. This signal acknowledges the HOLD signal.

HOLD

This signal indicates to the processor that external devices are requesting to access the address/data buses. It is available at pin 31.

Assembly Language Program Process

1. Kit on.
2. Press reset button.
3. Press enter key.
4. Press A for selection of Addressing mode and type starting location.
5. Type instruction like MOV SI 0500

6. Next line of program [memory location 0400 to 0413 I (INT 03)].
7. Press F7.
8. Press D for entering input data.
9. Press 0500 this is starting location of input data.
10. Continue give input up to 050F address.
11. Press F7 for main menu.
12. Press G for execution.
13. Press enter key.
14. Press f7 for saving result.
15. Press D.
16. Press 0510. This is out stored memory location.
17. We get result 0510 memory location is 15.
    15 is the largest number in storing.

SAMPLE PROGRAM

The monitor software of M86-02 resides in 16K Byte of EPROM. The sys- tem 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 Subroutine chapter.

EXAMPLE

The following sample programs are given here to make the user familiarise with the operation of M86-02.

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.

PROGRAM-1: To ADD two Binary numbers each 8 Bytes long:

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0400	F8	CLC	Clear Carry Flag.

0401	B9 04 00	MOV CX,0004	Load Counter register with no. of times addition to be per- formed (i.e. Initialize the counter register).
0404	BE 00 05	MOV SI,0500	Load source Index Reg. with starting address of Ist Binary no. (LSBs array)
0407	BF 08 05	MOV DI, 0508	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.
040A	8B 04	MOV AX, [SI]>	Load Data bytes (which are in location 0500 and 0501 in 16 Bit ACC.
			i.e. (0500) – AH
			(0501)- AL
040C	11 05	ADC [DI],AX	Add the contents (MS Bytes) of 0508, 0509 with the con- tents (LS Bytes) of 0500 + 0501 and store the result in location 0508 onwards.
040E	46	INC SI	Point at 0502 LOCN (Next rele-
040F	46	INC SI	vant source LOCN).
0410	47	INC DI	Point at next relevant LOCN. i.e.
0411	47	INC DI	0504
0412	49	DEC CX	Decrement the counter.
0413	75 F5	JNE 040A	If not zero (i.e. CX = 0000) then continue addition.
0415	F4/CC	HLT/INT 03	Else, Halt.

For example

		After Execution
0500 : 01	0508 : 0A	0508 : 0B
0501 : 02	0509 : 0B	0509 : 0D
0502 : 03	050A : 0C	050A : 0F
0503 : 04	050B : 0E	050B : 12
0504 : 05	050C : 0F	050C : 14
0505 : 06	050D : 10	050D : 16
0506 : 07	050E : 11	050E : 18
0507 : 08	050F : 12	050F : 1A

PROGRAM-2: To find the maximum no. in a given string (16 bytes long) and store it in location 0510.

ADDRESS	**OP CODE**	**MNEMONIC**	**COMMENTS**
0400	BE 00 05	MOV SI, 0500	Lond SI reg. with starting address of string.
0403	B9 10 00	MOV CX,0010	Initialize Counter register (with the length of string i.e. no. of bytes).
0406	B4 00	MOV AH,00	Initialize the 8 bit ACC.
0408	3A 24	CMP AH, [SI]	The first data byte of the string with ‘00’.
040A	73 02	JNB 040E	If both bytes match (above is equal) then branch to (I).
040C	BA 24	MOV AH, [SI]	Else, move the contents of (0500) into 8 bit ACC, i.e. a real no. in AH.

040E	46	INC SI	Point at the address of string.
040F	E0 F7	LOOPNE 0408	Decrement the counter value, if not zero, continue processing (searching to the Max. No. continued)
0411	98 24	MOV [SI], AH	Max. no. in 0510 address.
0413	F4/CC	HLT/INT 03	Halt.

For example

						After Execution		
0500	:	01	0508	:	12	0510	:	15
0501	:	02	0509	:	08			
0502	:	03	050A	:	09			
0503	:	04	050B	:	0A			
0504	:	05	050C	:	0B			
0505	:	06	050D	:	0E			
0506	:	15	050E	:	0C			
0507	:	07	050F	:	0D			

PROGRAM-3: To sort a string of a no. of bytes in descending order:

ADDRESS	**OP CODE**	**MNEMONIC**	**COMMENTS**
0400	BE 00 05	MOV SI,0500	Initialize SI reg. with Mem LOCN0500
0403	8B 1C	MOV BX, SI	BX has the no. of bytes (to be used for sorting) LOCNS 0500 & 0501.
0405	4B	DEC BX	Decrement the no. of bytes by one.
0406	8B 0C (3)	MOV CX, [SI]	Also CX has the no. of bytes in LOCNS 0500 and 0501.
0408	49	DEC CX	Decrement the no. of bytes by one
0409	BE 02 05	MOV SI,0502	Initialize SI reg. with the starting address of string (having data bytes).
040C	8A 04 (2)	MOV AL, [SI]	Move the first data byte of string into AL.
040E	46	INC SI	Point at the next bytes of the string.
040F	3A 04	CMP AL, [SI]	Compare the two bytes of string.
0411	73 06	JNB 0419	If two bytes are equal or 1st byte is above that the second byte branch to (1).
0413	86 04	XCHG [SI],AL	Else.
0415	4E	DEC SI	Second byte is less than first byte and swap the two bytes..
0416	88 04	MOV [S;],AL	
0418	46	INC SI	Point at the next LOCN of the string.
0419	E2 F1 (1)	LOOP 040O	Loop of CX is not zero (i.e. continue processing till z=0.

041B	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.
041C	BE 00 05	MOV SI, 0500	
041F	75 E5	JNE 0406	
0421	F4	INT 03	Halt.

For example

			After Execution		
0500	:	05	0502	:	28
0501	:	00	0503	:	25
0502	:	20	0504	:	20
0503	:	25	0505	:	15
0504	:	28	0506	:	07
0505	:	15			
0506	:	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
0400	BE 00 05	MOV SI,0500	Lond SI reg. with starting address of string.
0403	BF 08 05	MOV DI,0508	Lond DI reg. with starting address of result LOCNS.
0406	B2 34	MOV DL,34	Load DL with the multiplier ASCII digit.
0408	B9 08 00	MOV CX, 0008	Load counter reg. with the no. of bytes in the string.
040B	C6 05 00	MOV DI, 0050	
040E	80 E0 0F	AND DL,0F	MS nibble of multiplier is zeroed.
0411	8A 04 (a)	MOV AL, [SI]	First ASCII no. of string in AL.
0413	46	INC SI	Point at the next LOCN in string (of ASCII Nos.)
0414	80 E0 0F	AND AL,0F	MS nibble at multiplier no gap and is also zeroed.
0417	F6 E2	MUL DL	Perform the fn. AX = AL*DL.
0419	D4 0A	AAM	Perform the fn. AH = AL/DA. AL = remainder
041B	02 05	ADD AL,[DI]	The contents of AL (remainder obtained by performing the above operation.)
041D	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(=0508).

041E	88 05	MOV [DI],AL	
0420	47	INC DI	Point at the next Dest. LOCN.
0421	88 25	MOV [DI], AH	Contents of AH (quotient got in AAM operation) are moved in next best LOCN (0509)
0423	49	DEC CX	Decrement the counter reg.
0424	75 EB	JNE 0411	If not zero continue multiply and storing unpacked BCD digits, ELSE.
0426	F4	HLT	HALT.

For example

After Execution

(Unpacked BCD digits)

0500	:	31	0508	:	04	80
0501	:	32	0509	:	08	03
0502	:	33	050A	:	02	A6
0503	:	34	050B	:	07	00
0504	:	35	050C	:	01	06
0505	:	36	050D	:	06	10
0506	:	31	050E	:	06	10
0507	:	32	050F	:	08	00

PROGRAM-5: To Divide a string of unpacked ASCII digits:

ADDRESS	OP CODE	MNEMONIC	COMMENTS
0400	B2 36	MOV DL,36	DL having the divisor, a single 8 bit ASCII digit.
0402	BE 00 05	MOV SL,0500	Lond SI reg. with starting address of ASCII string.
0405	BF 08 05	MOV DI,0508	Load DI with the starting address of the result LOCNS.
0408	B9 08 00	MOV CX, 0008	Initialize the counter Reg. with the no. of bytes in the string.
040B	80 E2 0F	AND DL,0F	MS nibble of DL contents is zeroed.
040E	32 E4	XOR AH,AH	Initialize the 8 bit ACC (=00).
0410	AC (1)	LODSB	Load AL with the contents of address accessed by SI reg. and increment SI reg. i.e. point at the next address LOCN.
0411	80 E0 0F	AND AL,0F	MS nibble of AL contents is also zeroed.
0414	D5 0A	AAD	Perform the fn. AL=(AH*OA + AL), AH = 00.
0416	F6 F2	DIV DL	Perform the fn. AD/DL AL = Quotient and AH = remainder.
0418	AA	STOSB	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).
0419	E0 F5	LOOPNE 0410	Continue dividing the unpacked ASCII digits if the contents of C are not zeroed; else.
041B	F4	INT 03	Halt.

For example

		After Execution	
0500	31	0508	00
0501	32	0509	02
0502	33	050A	00
0503	34	050B	05
0504	35	050C	07
0505	36	050D	06
0506	31	050E	00
0507	32	050F	02

PROGRAM-6:

A data string of no. of bytes (to be specified in CX reg.) is located from the starting address 0500. This data string is to be converted to its equivalent 2'S complement form and the result is to be stored from 0600 onwards.

ADDRESS	**OP CODE**	**MNEMONIC**	**COMMENTS**
0400	BE 00 05	MOV SI,0500	Lond SI reg. with starting address of data string.
0403	BF 00 06	MOV DI,0600	Lond DI reg. with starting address of result LOCNS.
0406	B9 10 00	MOV CX,0010	Load CX with the no. of bytes in the string.
0409	AC (1)	LODSB	Load AL with data byte accessed by SI reg. and increment the address LOCN in SI reg.
040A	F6 D8	NEG AL	The contents of AL are 2'S complemented.

040C	AA	STOSB	Store AL contents in LOCN pointed to by DI ref. & increment in the current location in DI reg.
040D	E0 FA	LOOPNE 0409	If CX = 0000,continue 2'S complementing the data in string else;
040F	F4	INT 03	Halt.

For example

			After Execution		
0500	:	01	0600	:	FF
0501	:	02	0601	:	FE
0502	:	03	0602	:	FD
0503	:	04	0603	:	FC
0404	:	05	0604	:	FB
0405	:	06	0605	:	FA
0406	:	07	0606	:	F9
0407	:	08	0607	:	F8
0408	:	09	0608	:	F7
0409	:	0A	0609	:	F6
040A	:	0B	060A	:	F5
040B	:	0C	060B	:	F4
040C	:	0D	060C	:	F3
040D	:	0E	060D	:	F2
040E	:	0F	060E	:	F1
040F	:	10	060F	:	F0

Table of Contents

SYSTEM INTRODUCTION

GENERAL DESCRIPTION

VMC-8031/8051/89C51 is a single board **MICROCONTROLLER TRAINING/DE-VELOPMENT KIT** configured around the most popular Intel's 8051/8051, 8 bil singlo chip micro controller. VMC-8031/8051/19651 can be used to train Engineer's about the architecture, instruction set and capabilities of the 8031 chip and also for actually controlling any industrial process etc.

The kit communicates with the outside world through a key board having 28 hex keys and seven segment hexadecimal display. The kit has the capability of interacting with an IBM PC compatible PCXT/AT system

VMC-8031/8051/89C51 provides 32K bytes of RAM and 32K bytes of EPROM. The total on board program & data memory can be very easily expanded to 128K bytes in an appropriate combination of RAM and RIOM. The monitor is incorporated from 0000-1FFF and the necessary 32K bytes of RAM has an address of 2000-3FFF.

The Input/Output structure of VMC-8031/8051/89C51 provides 48 programmable I/ O lines. It has got 16 bit programmable Timer/Counter for generating any type of counting etc.

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

The system also provides a serial monitor covering most of the command avail- able in keyboard mode. A help menu makes the monitor more users friendly.

VMC-8031/8051/89C51 is configured around the internationally adopted Bus, which is the most popular bus for process control and real time applications. All the address, data and control lines are available at the FRC connector. The Kit is fully expandable for any kind of application.

The kit has onboard battery backup to store the program incase of power failure.

SYSTEM SPECIFICATION

CPU……………………………………………………………	8031/8051/89C51
MEMORY……………………………………………………	Total on board capacity of 128K bytes
RAM………………………………………………..……..	32K bytes and space for further expan- sion

ROM	32K bytes of EPROM-leaded with powerful program.
TIMER	16 bit programmable timer/counter using 8253
V.O	48 vO lines using 8255 PPI
KEYBOARD	10 keys for command, 16 keys for hexadecimal data entry 1 key for vector interrupt & 1 key for reset
LED DISPLAY	6 seven segment display (4 for address field & 2 for data field)
BUS	All data, address and controlsignals (TTL compatible available at FRC connector)
INTERFACE	RS-232-C through 8251.
POWER SUPPLY REQUIREMENT	+5V, 1.5Amp for the kit
OPERATING TEMPERATURE.	0 to 50°C
REAL TIME CLOCK	(OPTIONAL)
BATTERY BACKUP	For RAM

SYSTEM CAPABILITIES

1. Examine the contents of any memory location.
2. Examine/Modify the contents of register.
3. Modify the contents of any of the RAM location
4. Move a block of data memory to another data memory.
5. Move a block of data memory to program memory.
6. Move a block of program memory to program memory.
7. Insert one or more instructions in the user program.
8. Delete one or more instructions from the user program.
9. Fill a particular memory area with a constant.

10. Execute a program at full clock speed.

11. Execute a program in single step i.e. instruction by instruction.

12. Communicate with PC through RS232C port.

13. Upload/Download program to/fro PC.

14. Store the program incase of power failure during operation.

15. Interfacing of I/O lines through 8255.

16. Real Time Clock (Optional)

17. Set/Clear Break Point.

18. Enable/Disable Break Point.

19. Display Break Point

HARDWARE DESCRIPTION

GENERAL
The system has got 8031/8051/89C51 as the Central Processing Unit. The clock frequency for the system is 10 MHz and is generated from a crystal of 10 MH

MEMORY

VMC-8031/8051/89C51 provides 32K bytes of RAM using 62256 chip and 32K bytes of EPROM for monitor. The various chips which can be used are 2732, 2764, 27128, 27256, 6116 and 6264. There is one memory space provided on VMC-8031/8051/89C51. This one space can be defined any address slots from 3000FFFF depending upon the size of the memory chip to be used.

I/O DEVICES

The various I/O chips used in VMC-8031/8051/89C51 are 8279, 8255, 8251 & 8253. The functional role of all these chips is given below:

8279 (Keyboard & Display Controller)

6279 is a general purpose progra…

ports are given in Chapter-5, VMC-8031/8051/89C51 provides 48 Input/Output ports using 8255 chips.

8251 (USART)

This chip is a programmable communication interface and is used as a peripheral device. This device accepts data characters from the CPU, in parallel format ang then converts them into serial data characters for the CPU. This chip will signal the CPU whenever it can accept a new character for the CPU. The CPU cor the complete status of it at any time. 8251 has been utilized in VMC-8031/805

89C51 for RS-232C interface.

8253 (Programmable Internal Timer)

This chip is a programmable interval Timer/Counter and can be used for the generation of accurate time delays under software control. Various other functions that can be implemented with this chip are programmable rate generator Even Counter, Binary rate Multiplier, Real Time Clock etc. This chip has got three in dependent 16 bit counters each having a count rate of up to 2KHz. The first Timer/Counter (l.e. Counter 0) is being used for Single Step operation. However, its connection is also brought at connector space CN4. For single step operation CLKO signal of Counter 0 is getting a clock frequency of 1.575 MHz. The counter 1 is used to generate clock for 8251. The clock 1 is also feed with 1.535 Mhz.

DISPLAY

VMC-8031/8051/89C51 provides six digits of seven segment display. Four digits are for displaying the address of any location or name of any register, whereas the rest of the two digits are meant for displaying the contents of a memory location or of a register. All the six digits of the display are in hexadecimal notation.

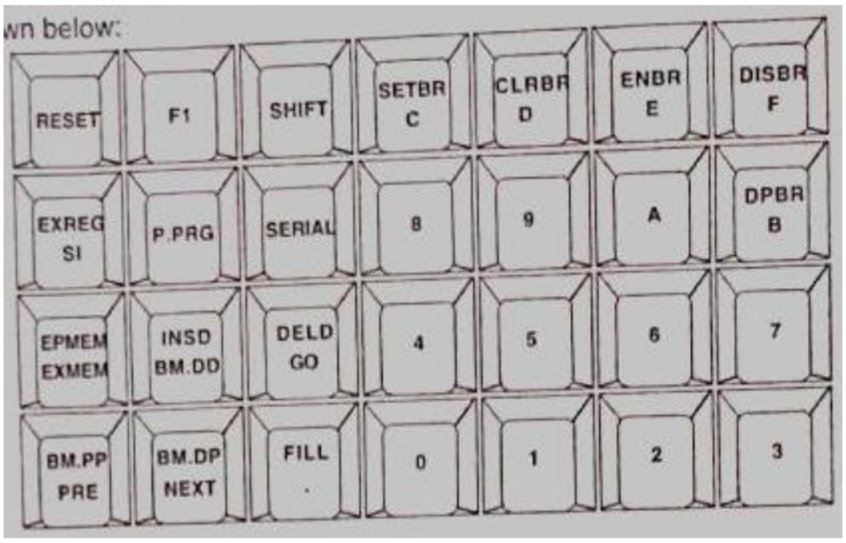
BATTERY BACK-UP

The VMC-6031/8051/89C51 provides a battery back-up for the RAM area. A rechargeable cell is provided onboard for the storing of program in this RAM.

COMMAND DESCRIPTION

KEYBOARD DESCRIPTION

VMC-8031/8051/89C51 has 28 keys and six-seven segment display to communicate with the outside world. As VMC-8031/8051/89C51 is switches on, a message -UP 51' is displayed on the display after pressing Reset The key board is as shown below



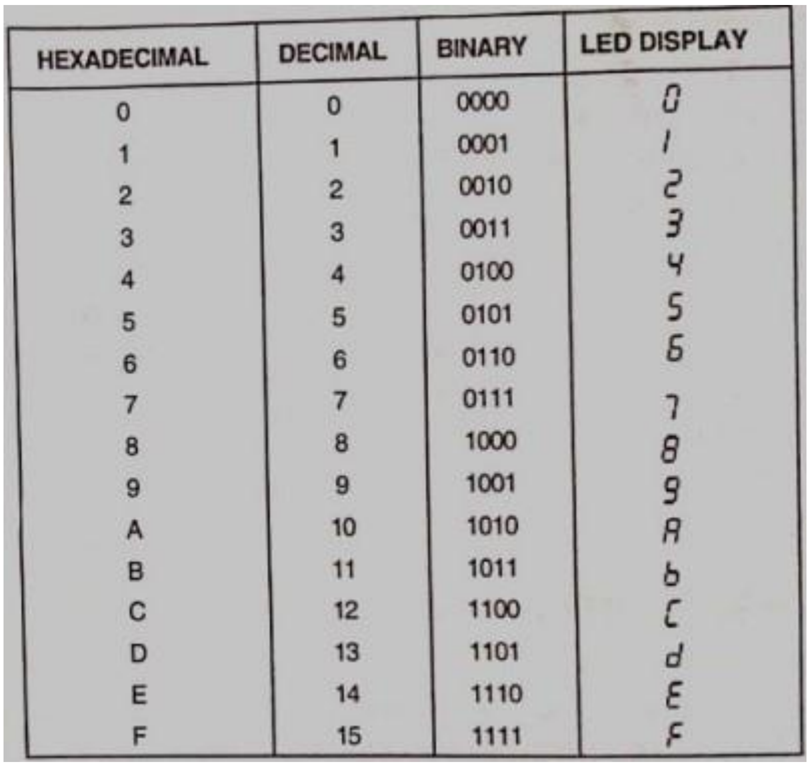
RESET	Reset the system
SHIFT	Provides a second level command to some keys.
GO	To execute the program.
S.I.	To execute the program in single step mode.
EXREG	Examine Register; allows user to examine and modify the contents of different registers.
EXMEM	Examine Program Memory; allows user to examine/modify any data memory location.
PRE	Previous is used as an intermediate terminator in case of Examine Memory. It decrements the PC contents and writes the contents of data field to the address displayed in the address location
NEXT	Increment is used as a intermediate terminator in Increment is used Examine Register etc. It increments the Examine Memand writes the data tying in data field at the location displayed at address field.
"-"	Terminator is used to terminate the command and write the data in data field at the location displayed in address field.
BM.DD	Allows user to move a block of data memory to another data memory.
BM.PP	Allows user to Move a block of Program memory to another program memory.
BM.DP	Allows user to Move a block of Data memory to program memory.
FILL	Allows user to fill RAM area with a constant.

INS	Inserts one or more data bytes in the user's program/data area.
DELD	Deletes one or more data bytes from the user's program/data area.
SETBR	Set Break point allows user to set a break point anywhere in the user program.
CLRBR	Clear Break point allows user to clear a break point any- where in the user program.
ENBR	Enable Break point allows user to enable a break point any- where in the user program.
DPBR	Display Break point allows user to see the address where the break point was set.
P.PRG	This key is for further expansion
SERIAL	This key is used for Serial Communication with PC.

All commands are followed by a set of numeric parameters separated by PREV. NEXT &" (Execute) to work as delimiters

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

The VMC-8031/8051/89C51 accepts all data and address in hexadecimal form as given in the table – 1



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

LIST O F COMMANDS

1. RESET
2. EXAMINE MODIFY REGISTER
3. EXAMINE MODIFY DATA MEMORY
4. EXAMINE MODIFY PROGRAM MEMORY
5. GO
6. SET BREAKPOINT
7. ENABLE BREAKPOINT
8. DISABLE BREAKPOINT
9. DISPLAY BREAKPOINT
10. CLEAR BREAKPOINT
11. SINGLE INSTRUCTION
12. BLOCK MOVE FROM PROGRAM MEMORY TO PROGRAM MEMORY
13. BLOCK MOVE FROM DATA MEMORY TO PROGRAM MEMORY
14. BLOCK MOVE FROM DATA MEMORY TO DATA MEMORY
15. DELETE DATA
16. INSERT DATA
17. FILL

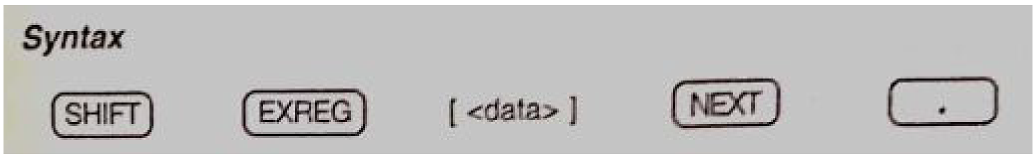
COMMAND DESCRIPTION

RESET

This key initializes the VMC-8031/8051/89C51 Kit and displays '- UP51' on the display. A “– “ on the left most end of display indicates that the system is expect- ing a valid command.

EXAMINE/MODIFeY REGISTER (EXREG)

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



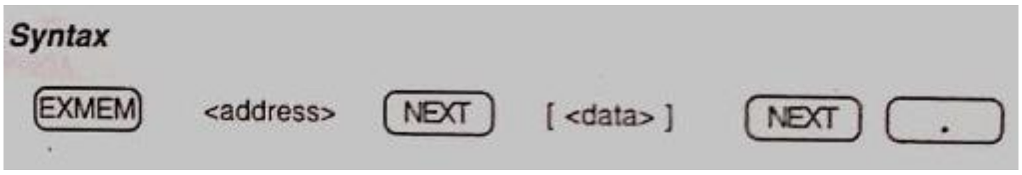
Syntax

SHIFT EXREG [<data>] NEXT .

On pressing SHIFT then EXREG key, the contents of Areg. are displayed in the data field. One can modify the contents by entering the data and pressing NEXT or else just press NEXT to see the contents of next registers. On pressing NEXT every time, the system wil display the contents of al the registers. The registers are displayed ni the sequence of A, B, PCH, PCL, DPH, DPL, SP, PS, TLO, THO, TL1, TH1, TOD, SCO, IE, RO ot R7. fI at any stage one wants to terminate the command, just press"." key, the system wil display "-“.

EXAMINE/MODIFY DATA MEMORY(EXMEM)

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



Syntax

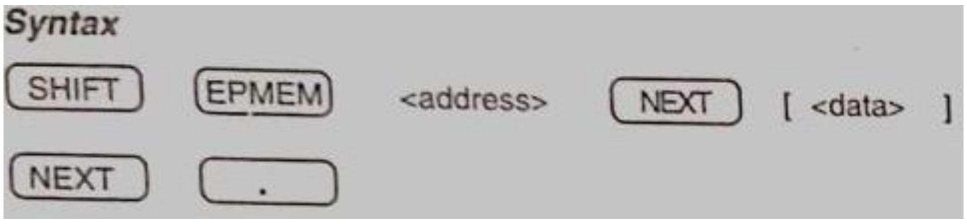
EXMEM <address> NEXT [<data>] NEXT .

On pressing this key, a dot is displayed in the address field. One can now enter the address of any location which have to examine, and press NEXT. The con- tents of this location is displayed in the data field. If one wants to examine the contents of next location, just press NEXT and the address in the address field will be incremented by one and its contents will be displayed in the data field. Same way if one wants to examine the content of previous location just press PRE key and the address in the address field will be decremented by one and its contents wil be displayed in the data field.

IF one wants to modify the contents of any RAM location, then enter the data and press NEXT. The data field will be written in the address displayed in the address field and simultaneously the contents of next location will be displayed. If at any stage one ants to terminate the command, just press".* key, the system will display"-".

EXAMINE/MODIFYPROGRAM MEMORY(EPMEM)

This command is used to examine the contents of any of the program memory location.



Syntax

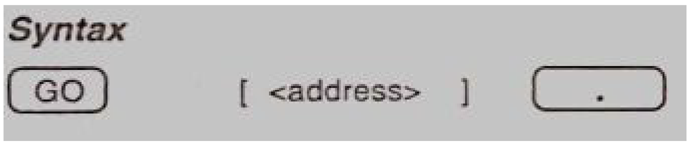
SHIFT EPMEM <address> NEXT [<data>]

NEXT .

The rest of the operation is same as explained in EXMEM command.

GO

This command is used to execute the program in full clock speed. On pressing this key, the program counter contents are displayed in the address field with the data ni the data field. Enter the starting address of the program, from where the program is to be execute. Press Terminate (-) key. The CPU wil start executing the program and E wil be displayed ni the address field.

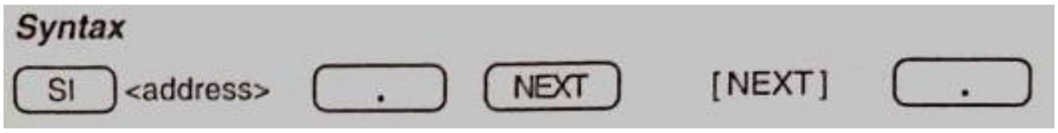


Syntax

GO [<address>] .

SINGLE INSTRUCTIONS

This command is used to execute the program instruction by instruction. On pressing SI, the program counter content is displayed on the address field and its data in the data field. If one wants to modify the address, one can do that. After entering the address, press NEXT, the contents of the entered address is dis- played. On pressing NEXT, one instruction wil be executed and the address of the next instruction will be displayed with its data in the data field. Each time NEXT is pressed, one instruction is executed. If one wants to terminate and command at any stage, one can do that using (.) key. On pressing (.) key, a - is displayed in the address field. One can now examine any internal register of CPU or any memory location without pressing Reset.

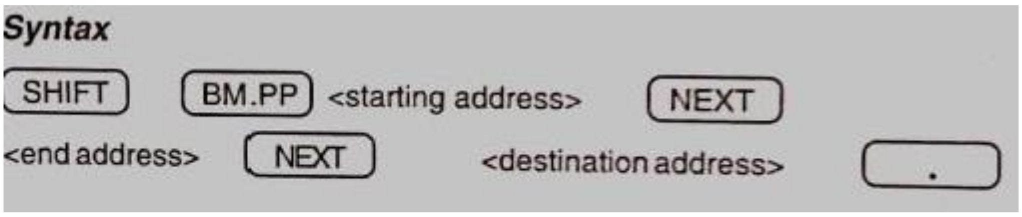


Syntax

SI <address> . NEXT [NEXT] .

BLOCK MOVE PROGRAM MEMORY TO PROGRAM MEMORY (BM.PP)

This command allows the user to move the block of program from the program memory location to another program memory location. On pressing SHIFT and BM.PP key a dot is displayed in the address field. Enter the starting address of the block to be moved and press NEXT. Again a dot is displayed. Now enter the end address of the block and press NEXT. Again a dot is displayed. Now enter the destination address and press Terminate (.) key. A - is displayed in the display



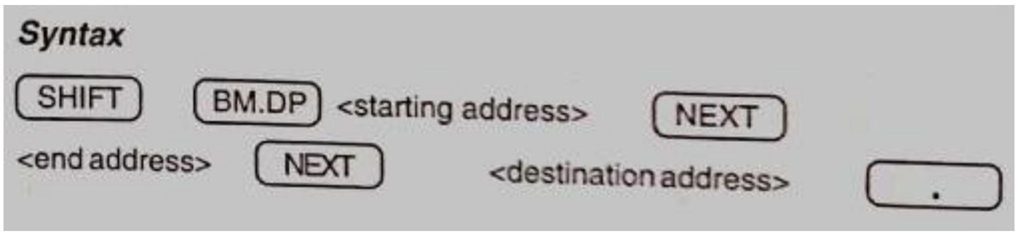
Syntax

SHIFT BM.PP <starting address> NEXT

<end address> NEXT <destination address> .

BLOCK MOVE DATA MEMORY TO PROGRAM MEMORY (BM.DP)

Its works the same way as explained above except that the movement of information is from data to program area.

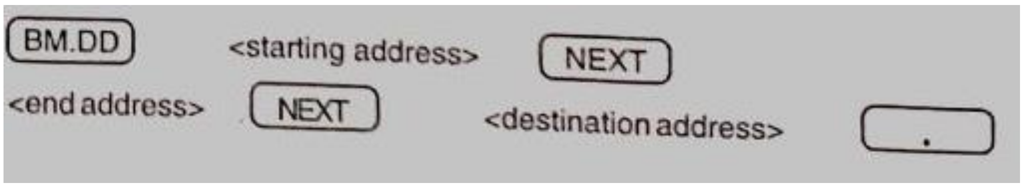


Syntax

SHIFT BM.DP <starting address> NEXT

<end address> NEXT <destination address> .

BLOCK MOVE DATA MEMORY TO DATA MEMORY (BM.DD)



BM.DD <starting address> NEXT

<end address> NEXT <destination address> .

DELETE (DELD)

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

1) Starting address of the user program.

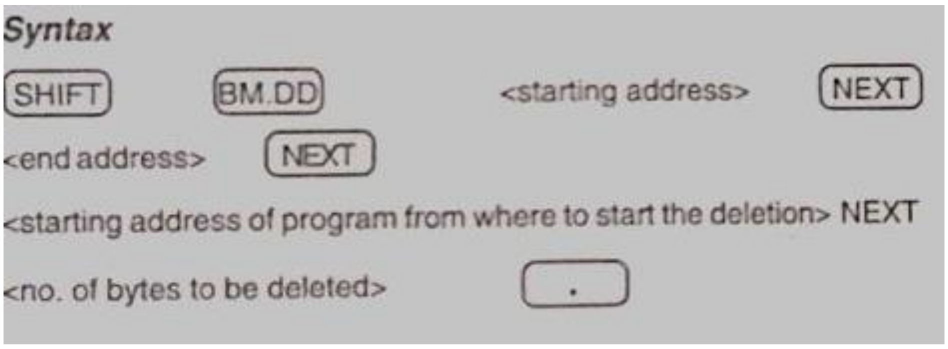
2)End address of the user program.

3)Address of the location from where onwards the bytes are to be deleted.

4)Address of the location til where the bytes are to be deleted.

It the delete iperation is performed ni a program, the jump & call address of memory referenced instructions need ta be modified.

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



Syntax

SHIFT BM.DD <starting address> NEXT
<end address> NEXT
<starting address of program from where to start the deletion> NEXT
<no. of bytes to be deleted> .

INSERT (INSD)

This command allows the user to insert one or more instructions in the user's program. This following information is required to be entered.

1) starting address of the program.

2) End address of the program.

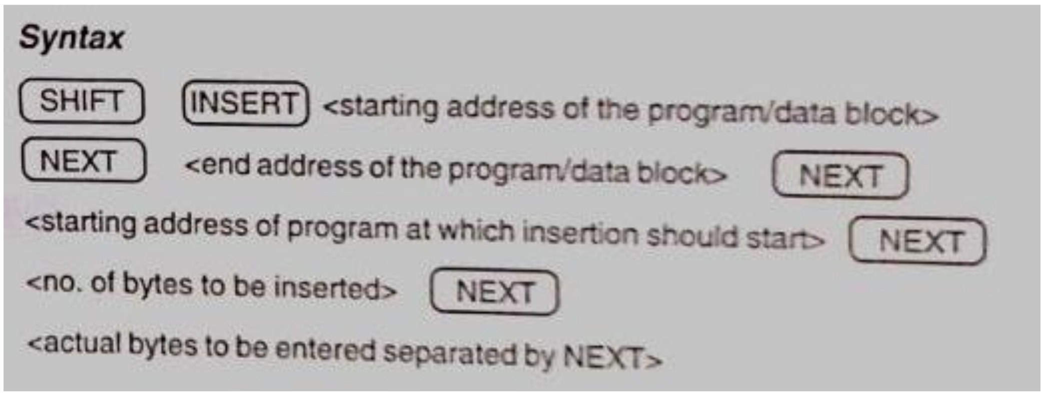
3) Address from where the bytes are to be entered.

4) No. of bytes to be entered.

5) Data.

On pressing SHIFT and INSD, a ' ' is displayed in the address field. Enter the starting address of the program from where you want to insert and press next. again a i s displayed. Now enter the end address of the program and press NEXT.

Now enter the address of the program where the yes are to be inserted and press NEXT. The system wil display t h e ' ' again. Now enter the no. of bytes to be inserted and press NEXT. The system wil display the address where you wish to enter the bytes. With its current data in the data field. Enter the bytes you want to insert using NEXT key. When all the bytes are entered, a ?wil be displayed indicating that the Insert is completed.



Syntax

SHIFT INSERT <starting address of the program/data block>
NEXT <end address of the program/data block> NEXT
<starting address of program at which insertion should start> NEXT
<no. of bytes to be inserted> NEXT
<actual bytes to be entered separated by NEXT>

Syntax

‹starting address of program from where to start the deletion> NEXT ‹no. of bytes to be deleted>

FILL

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

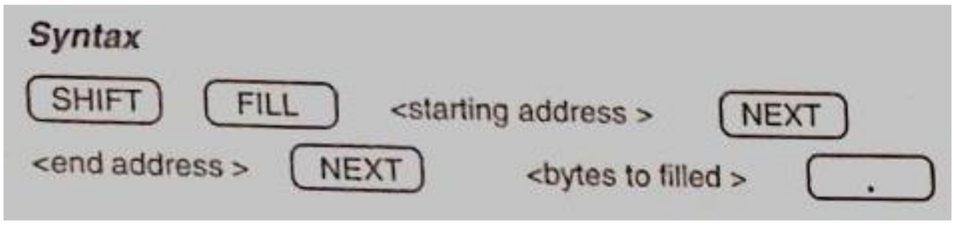
1) Starting address of the memory area from where the data should be stored.

2) End address of the memory area till were the data should be stored.

3) The constant with which the data should be done i.e. 22

Press SHIFT and FILL. A ',’ wil be displayed ni the address field. Enter starting address and press NEXT. Again a',’ will be displayed. Now enter the end address till where the filing to be done and press NEXT.

Again a ‘,’ will be displayed. Now enter byte to be filled and press ‘,’ key. The system will display ‘,’.

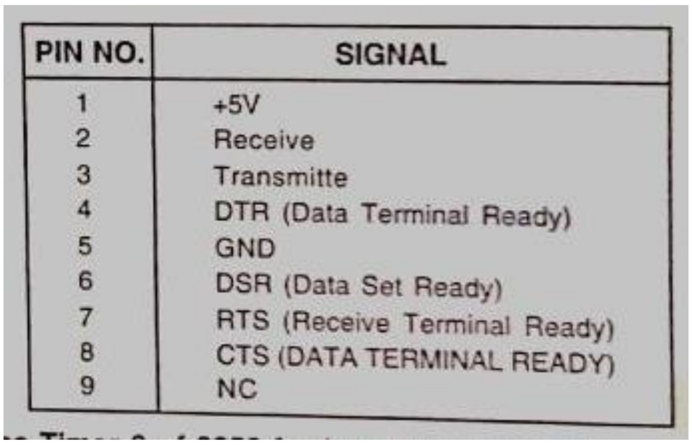


ONBOARD INTERFACES

GENERAL DETAILS OF INTERFACES
VMC-8031/8051/89C51 provides the following on-board interfaces as mentioned earlier.

)1 RS-232C interface through 8251 USART chip. 2) Serial Mode Interface.

RS-232C INTERFACE
The RS-232C interface is provided on VMC-8031/8051/89C51 through Intel's USART chip 8251 (Universal Synchronous Asynchronous Receiver Transmitter). It is a seven line interface with all seven signal lines being brought out at connector space CN7, which is a 9 pin D type Male Connector. The pin detail for this interface is as follows:



PIN NO.	SIGNAL
1	+5V
2	Receive
3	Transmitte
4	DTR (Data Terminal Ready)
5	GND
6	DSR (Data Set Ready)
7	RTS (Receive Terminal Ready)
8	CTS (DATA TERMINAL READY)
9	NC

The 8251 uses Timer 0 of 8253 for baud rate generation.

The VMC-8031/8051/89C51 monitor contain a software for connecting VMC-8031/ 8051/89C51 through this interface to the serial port of IBM-PC compatible computer. The commands are provided for stering the contents of system memory to the floppy drive in Intel's HEX format and same way the Intel HEX files can be loaded into the system memory.

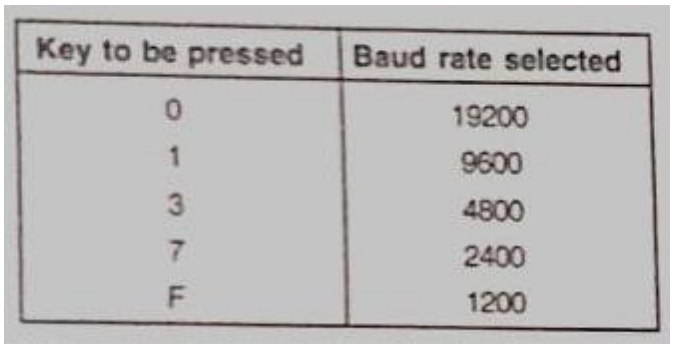
The interface to the serial port of the IBM-PC Compatible Computer can be achieved at any of the baud rate viz. 19200, 9600, 4800, 2400 & 1200 which can be selected as explained ni the chapter -ni Serial Monitor commands. However for operating at 19200 baud, only PC/AT should be used.

BAUD RATE SELECTION

The user can set the VMC-8031/8051/89C51 kit at the desired baud rate by following the procedure given below:

1) Press serial key on kit keyboard.

2) Press the required key from the table given below:



Key to be pressed	Baud rate selected
0	19200
1	9600
3	4800
7	2400
F	1200

Press FIL key. Amessage SEr wil be displayed on the Seven Segment display of kit and fi the computer/terminal si also set at the same baud rate, a message:

VMC-51> SERIAL COMM. COMMAND>

appears on the computer screen.

SERIAL MODE INTERFACE

Utility of this kit is that it can be interfaced with PC through serial port for operations with the help of PC Keyboard.

1) Connect the kit to the Serial port of PC with the help of cable of be connected at CN6.

2) Switch ON the kít &PC.

3) Run the HYPER TERMINAL and set the COM port &Baud rate.

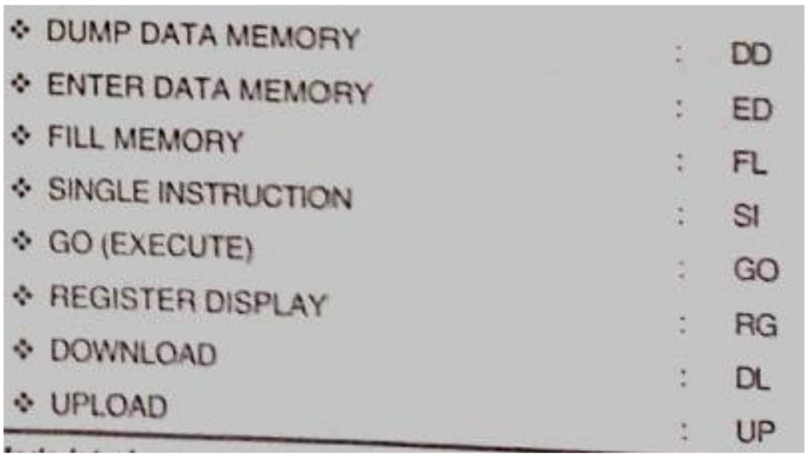
4) Press RESET, SERIAL and the desired numeric keyof the kit and FILL.

5) The Display on monitor wil be as follows.

VMC-51 SER COMM. COMMAND>

SERIAL COMMANDS
VMC-8031/8051/89C51 has a Serial Interface which allow the user to execute most of the key board commands through the ASCIl Keyboard. VMC-8031/8051/ 89C51 has a serial interface through USART 8251. The system can be interface with the serial port of a IBM-PC/XT/AT compatible Computer. Program having capture facility.

The following commands are available through Serial Mode. Use only CAPITAL letters for executing the commands.



❖ DUMP DATA MEMORY	:	DD
❖ ENTER DATA MEMORY	:	ED
❖ FILL MEMORY	:	FL
❖ SINGLE INSTRUCTION	:	SI
❖ GO (EXECUTE)	:	GO
❖ REGISTER DISPLAY	:	RG
❖ DOWNLOAD	:	DL
❖ UPLOAD	:	UP

DUMP DATA MEMORY (DD)
This command dumps the data memory between and including two specified address separated by a space.

Syntax

DD «starting address> <end address> ‹CR>

The starting address should not be higher than the end address.

ENTER DATA MEMORY (ED)

This command is used for entering data bytes in to data memory area.

Syntax

ED «starting address> ‹CR>

On pressing ‹CR> the PC content is incremented and on pressing ‹Sapce bar> the PC content is decremented which enables the user to examine/modify the data of the location shown. For coming out of this command press Esc.

FILL MEMORY (FL)

This command is used to fill a memory area with a constant.

Syntax

FL «starting address> «end address> ‹byte to be filled> ‹CR>

SINGLE INSTRUCTION (SI)

This command is used for running a program ni single instruction mode i.e. instruction by instruction.

Syntax

SI<starting address> ‹CR›

After the execution of an instruction, the contents of all the registers are dis- played and system waits for a Carriage Return to be pressed for the execution of the next instruction. For coming out of this command press Esc key.

EXECUTION IN FULL S P E E D (GO)

This command is used ot execute a program ni ful clock speed mode.

Syntax

GO starting address> ‹CR>

REGISTER DISPLAY (RG)

This command displays the contents of al the registers of 8031 controller.

Syntax

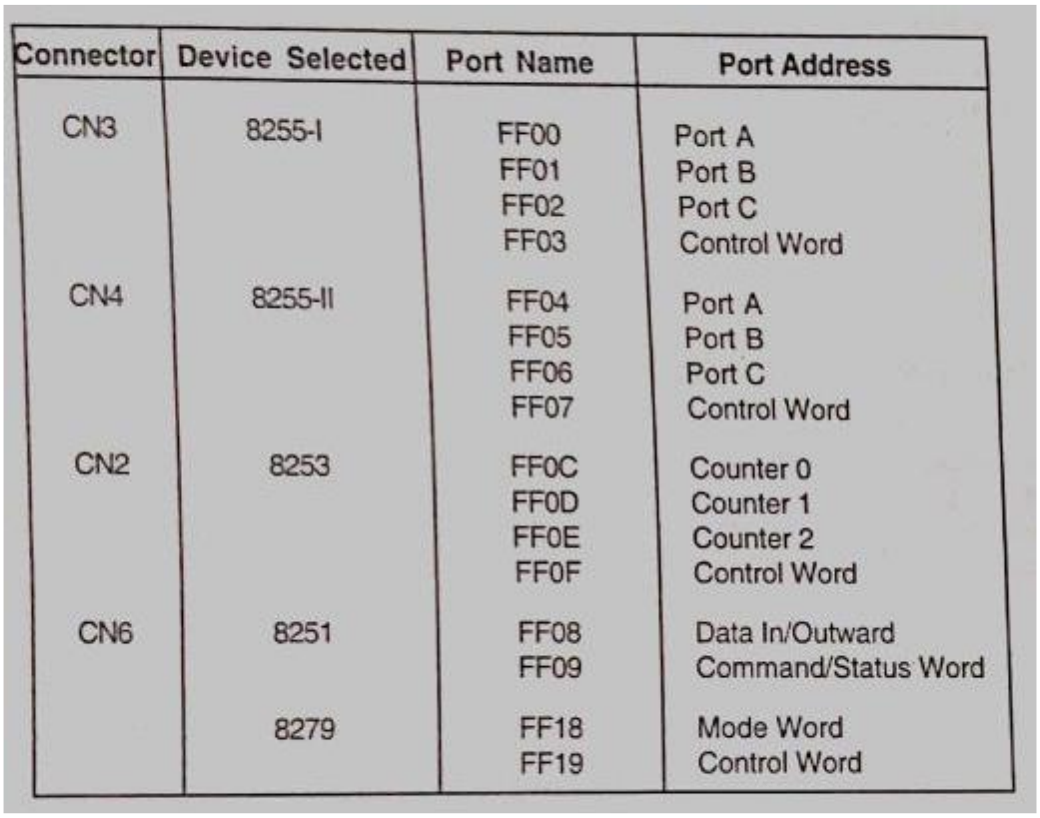
RG <CR>

MEMORY MAPPING

The 8031/8051 chip supports 64K bytes of program memory and 64K bytes of data memory. That means the total memory which the 8031 can address physi- caly is 128K bytes. Since ni a practical situation, the user may need the same memory area for Data as well as programs, the system VMC-8031/8051/89C51 has been designed in a way that the same memory chip can be used as program/data memory.

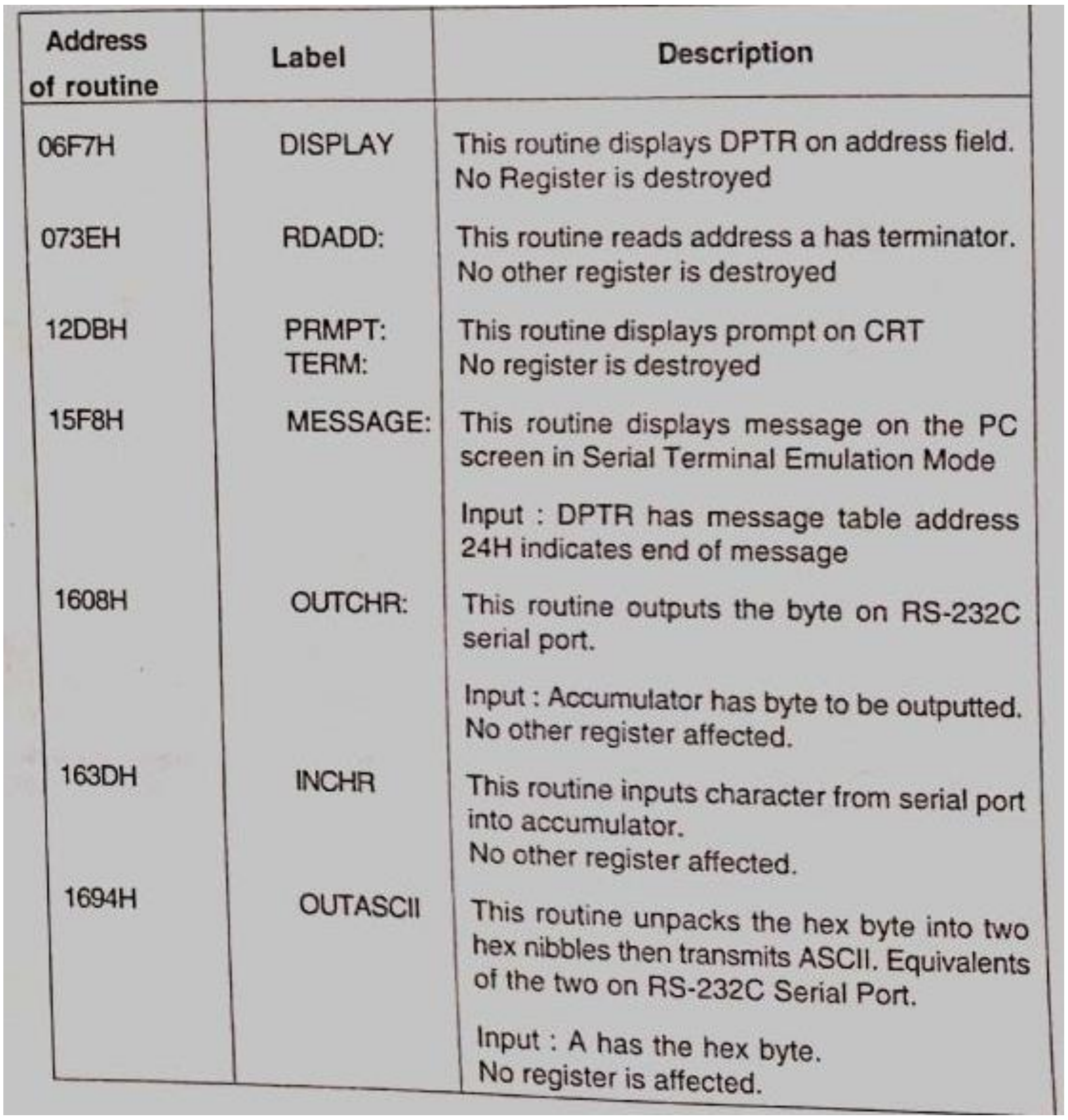
There are three memory sockets provided on the kit. The two sockets are pro- vided with the memory R(AM &Monitor ROM) and one socket is free for the memory expanded by user.

The RAM area provided with the address range from 2000-3FFF. The addres of memory si from 6000-7FFF.

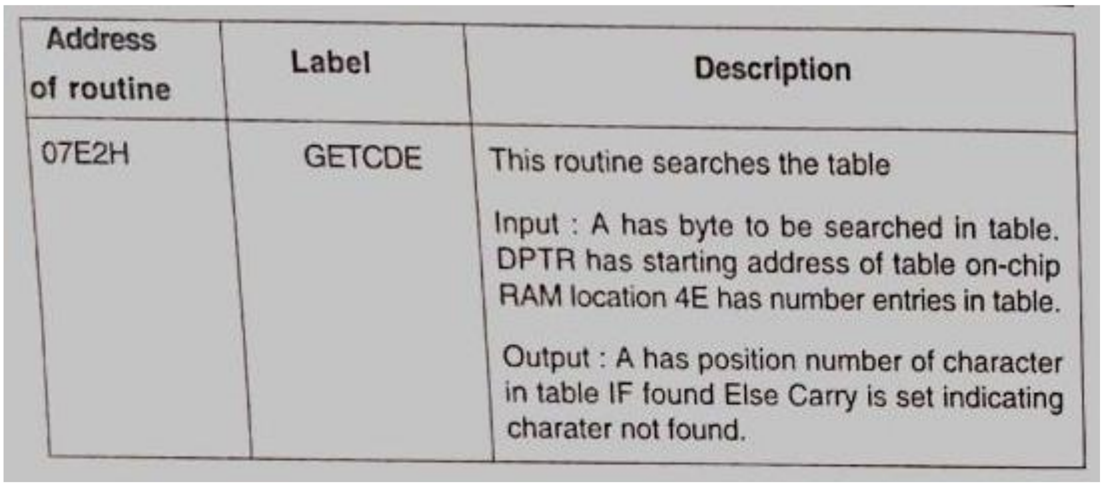


Connector	Device Selected	Port Name	Port Address
CN3	8255-I	FF00	Port A
		FF01	Port B
		FF02	Port C
		FF03	Control Word
CN4	8255-II	FF04	Port A
		FF05	Port B
		FF06	Port C
		FF07	Control Word
CN2	8253	FF0C	Counter 0
		FF0D	Counter 1
		FF0E	Counter 2
		FF0F	Control Word
CN6	8251	FF08	Data In/Outward
		FF09	Command/Status Word
	8279	FF18	Mode Word
		FF19	Control Word

VMC-8031/8051/89C51 monitor uses certain subroutines for its operation, which can also be used by the user of his programs. The addresses of these routines and their descriptions are given here



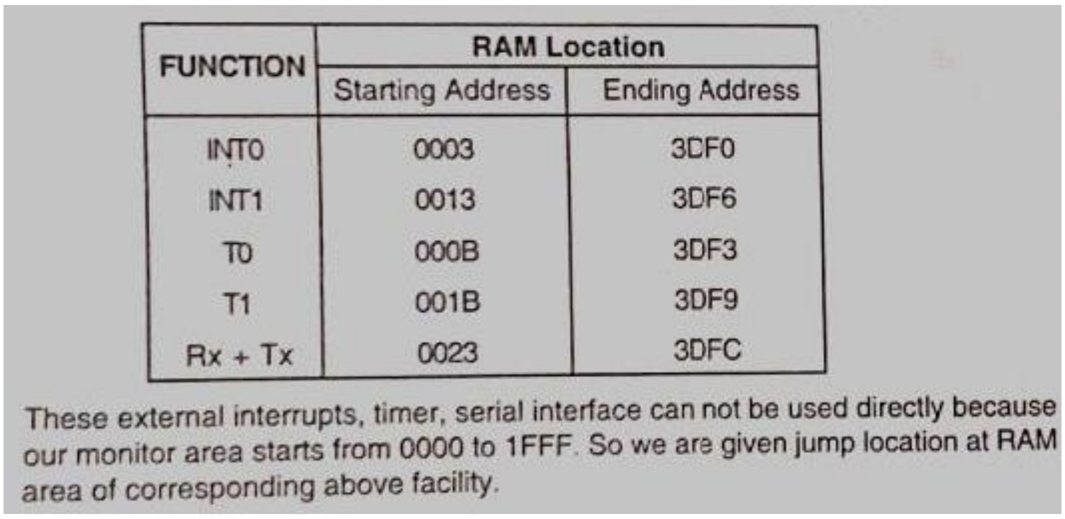
Address of routine	Label	Description
06F7H	DISPLAY	This routine displays DPTR on address field. No Register is destroyed
073EH	RDADD:	This routine reads address a has terminator. No other register is destroyed
12DBH	PRMPT: TERM:	This routine displays prompt on CRT No register is destroyed
15F8H	MESSAGE:	This routine displays message on the PC screen in Serial Terminal Emulation Mode Input : DPTR has message table address 24H indicates end of message
1608H	OUTCHR:	This routine outputs the byte on RS-232C serial port. Input : Accumulator has byte to be outputted. No other register affected.
163DH	INCHR	This routine inputs character from serial port into accumulator. No other register affected.
1694H	OUTASCII	This routine unpacks the hex byte into two hex nibbles then transmits ASCII. Equivalents of the two on RS-232C Serial Port. Input : A has the hex byte. No register is affected.



Address of routine	Label	Description
07E2H	GETCDE	This routine searches the table Input : A has byte to be searched in table. DPTR has starting address of table on-chip RAM location 4E has number entries in table. Output : A has position number of character in table IF found Else Carry is set indicating charater not found.

<u>**INTERRUPT**</u>

There are two external interrupts called INTO & INT are coming from CPU. Interrupt INTO is used for keyboard and interrupt INT is free for the user. The different interrupts, Timer, Serial Interface addresses are as follows



FUNCTION	RAM Location	
	Starting Address	Ending Address
INTO	0003	3DF0
INT1	0013	3DF6
T0	000B	3DF3
T1	001B	3DF9
Rx + Tx	0023	3DFC

These external interrupts, timer, serial interface can not be used directly because our monitor area starts from 0000 to 1FFF. So we are given jump location at RAM area of corresponding above facility.

SAMPLE PROGRAMS
Sample Programs are given here to the user to understand the programming techniques of 8031/8051/89C51 microcontrollers.

Program-1

"FLASHING DISPLAY OF SUPERB ON SEVEN SEGMENT DISPLAY ON EXECUTING THIS PROGRAM FROM ADDRESS 2000H, "SUPERB" MES- SAGE FLASHES ON THE DISPLAY OF THE KIT".

ADDRESS	CODE	LABEL	MNEMONIC	OPERAND	COMMENTS
2000	90 20 2E	HERE:	MOV	DPTR,#202E	;SUPERB message
2003	12 06 F7		LCALL	06F7	;display routine
2006	7B 00		MOV	R3, #0	
2008	7A 00	LOOP2:	MOV	R2, #0	
200A	DA FE	LOOP1:	DJNZ	R2,	;delay code
200C	DB FA		DNJZ	R2, 2008	
200E	90 20 34		MOV	DPTR,#2034	;blank message
2011	12 06 F7		LCALL	06F7	;display routine
2014	7B 00		MOV	R3, #0	
2016	7A 00	LOOP4:	MOV	R2, #0	
2018	DA FE	LOOP3:	DJNZ	R2, 2018	;delay code

```
201A          DB FA                 DJNZL        R3, 2016
201C          80 E2                 SJMP         2000
202E          49 83 31 61 F5    C1ADRI: DFB      49,83,31, 61,F5,C1      ;data superb
2034          FF FF FF FF FF   FFADR2: DFB       FF, FF, FF, FF, FF, FF   ; blank
display
```

Program- 2:

Addition of 2 numbers and stored result at 3012 address.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
3000	78 04	MOV R0, #04H	;1ST number
3002	79 05	MOV R1, #05H	;2nd number
3004	E8	MOV A, R0	
3005	29	ADD A, R1	;adding result at A reg.
3006	90 30 12	MOV DPTR, #3012H	;result at 3012 address
3009	F0	MOVX @DPTR, A	
300A	80 FE	SJMP 300A	

The result at 3012 address will be 09H.

Program- 3:

Subtraction of 2 numbers and stored result at 3012 address.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
3000	78 05	MOV R0, #05H	;1ST number
3002	79 04	MOV R1, #04H	;2nd number
3004	C3	CLR C	

3005	E8	MOV A, R0	
3006	99	SUBB A, R1	;adding result at A reg.
3007	90 30 12	MOV DPTR, #3012H	;result at 3012 address
300A	F0	MOVX @DPTR, A	
300B	80 FE	SJMP 300B	

The result at 3012 address will be 01H.

Program- 4:

Division of 2 numbers and stored result at 3012 address.

ADDRESS	OP CODE	MNEMONIC	COMMENTS
3000	78 08	MOV R0, #08H	;1ST number
3002	79 02	MOV R1, #02H	;2nd number
3004	E8	MOV A, R0	
3005	89 F0	MOV B, R1	
3007	84	DIV AB	;divide
3008	90 30 12	MOV DPTR, #3012H	;div result at 3012 address
300B	F0	MOVX @DPTR, A	
300C	80 FE	SJMP 300C	

The result at 3012 address will be 04H.

Program- 5:

Multiplecation of 2 numbers and stored result at 3012 address.

ADDRESS	**OP CODE**	**MNEMONIC**	**COMMENTS**
3000	78 08	MOV R0, #03H	;1ST number
3002	79 02	MOV R1, #02H	;2nd number
3004	E8	MOV A, R0	
3005	89 F0	MOV B, R1	
3007	A4	MUL AB	;multiplecation
3008	90 30 12	MOV DPTR, #3012H	;mul. result at 3012 address
300B	F0	MOVX @DPTR, A	
300C	80 FE	SJMP 300C	

The result at 3012 address will be 06H.