8086 MICROPROCESSOR BASED EXPERIMENTATION & APPLICATIONS-A CASE STUDY

A DISSERTATION

submitted in partial fulfilment of the
requirements for the award of the degree
of
MASTER OF ENGINEERING
in
ELECTRICAL ENGINEERING
(System Engineering & Operational Research)

By
INDRA GUPTA

DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE - 247 667 (INDIA)
MARCH, 1986

DEDICATED
TO
PAPA & DIDI

CERTIFICATE

Certified that the dissertation entitled 8086 MICROPROCESSOR BASED EXPERIMENTATION AND APPLICATION- A CASE STUDY which is being submitted by Ms. Indra Gupta in partial fulfilment for the award of the degree of MASTER OF ENGINEERING in ELECTRICAL ENGINEERING (System Engineering and Operational Research) of the University of Roorkee, Roorkee, is a record of student's own work carried out by her under my supervision and guidance. The contents embodied in this dissertation has not been submitted for the award of any other degree or diploma.

This is further to certify that she has worked for a period of about 7 months, from Aug. 1985 to Feb.1986 for preparing this dissertation at this University.

M Vasantha

(M.K.VASANTHA)
Reader
Electrical Engineering Department
University of Roorkee
Roorkee-247667.

DATED MARCH 4, 1986.

Thought —

There has been a long trend of expressing gratitude by the students to persons concerned through acknowledgements, which now-a-days has become a convention.

So the author could not go beyond the limitations:though words seemed always insufficient for expressing thoughts to her.

ACKNOWLEDGEMENTS

Sri M.K.Vasantha, Reader, Elect. Engg. Deptt., U.O.R., Roorkee who stood by my side all through the dissertation work as my guide, deserves special mention in addition to immense gratitude.

Dr. P.Mukhopadhyay deserves special thanks for taking necessary action to procure the VMC-86 Microcomputer and stepper motors along with required units, firstly as the P.G.A.P.C. and later as the H.O.D., Elect. Engg. Deptt., U.O.R., Roorkee.

My sincere thanks should be paid to Dr. V.K.Verma, Prof. and Dr. J.D.Sharma, Prof. Elect. Engg.Deptt., U.O.R., Roorkee for fulfilling the requirements at various stages of work.

Mr. S.Batra Lect. and Mr. Bhatnager Lect., Mech. Engg. Deptt., U.O.R., Roorkee should be thanked for their kind advice/ directions time to time and Mr. Mukhtar of Mech. Engg. Deptt., U.O.R., Roorkee for sparing his valuable time and extending his help for mechanical fabrication.

I would do injustice if I fail to thank the group (SEOR) teachers, E.E., Deptt., U.O.R., Roorkee for their help as and when required.

A word of thanks goes to Mr. Rajender Singh and Mr. J.P. Sharma, workers in Microprocessor and Computer Lab. Elect. Engg. Deptt., U.O.R., Roorkee for their cooperation during lab. timings.

The author also wishes to thank the many individuals especially Ms. Anita Patra who deserve credit for assisting in preperation of this dissertation.

Indra

(INDRA GUPTA)

ABSTRACT

The present dissertation report has been fragmented into seven chapters. The history and introduction to microprocessor development is presented in chapter-0. The chapter-1 then discusses the main system: microcomputer VMC-86.

Further the man-machine communication is achieved via the CRT system interface, it is expanded upon along with the details of the interfacing circuitery in chapter-2. Chapter 3 deals with the development of some software program modules. Few Hardware Modules have also been tested on VMC-86 and this experimentation is considered in chapter 4, along with necessary software.

The Stepper Motor Control is being considered as the case study and is discussed in Chapter-5, in this the following controls have been achieved.

1. R.P.M.Control
2. Linear Displacement Control
3. S.H.M.with uniform velocity
4. Point selection on a x-y plane.

As every thing has its own limitations; this dissertation is no exception. So, the closing chapter-6 further on presents the limitations of the present work and areas of extension possible. However to provide additional information Appendices have been incorporated in this report as per the requirements.

CONTENTS

PAGE NO

CERTIFICATE

ACKNOWLEDGEMENTS

ABSTRACT

CHAPTER

0 HISTORICAL INTRODUCTION ... 1

1. SALIENT FEATURES OF VMC-86

1.1 Introduction ... 5

1.2 Hardware Description ... 6

1.21 CPU ... 6

1.22 Clock Generator ... 7

1.23 Wait State Generator ... 8

1.24 DISPLAY ... 8

1.25 Multibus Buffering ... 9

1.26 Interface ... 9

1.3 Memory Mapping ... 9

1.4 I/O Mapping ...10

1.5 RAM Memory Decoding ...15

1.6 I/O Decoding ...17

1.7 ROM Decoding ...17

1.8 VMC/86 In Multiprocessing Mode ...20

2. INTERFACING OF CRT WITH 8086 36 - 44

3. DEVELOPMENT OF SOFTWARE MODULES 45 - 81

4. EXPERIMENTATION WITH HARDWARE MODULES 82 - 95

5. CONTROLLING THE STEPPER MOTOR 97 - 119

6. DISCUSSION AND CONCLUSION 120-122

PAGE NO

APPENDICES

APPENDIX - A ... 123

APPENDIX - B ... 126

APPENDIX - C ... 131

APPENDIX - D ... 137

APPENDIX - E ... 138

APPENDIX - F ... 139

APPENDIX - G ... 141

APPENDIX - H ... 154

REFERENCES ... 168

BIBILOGRAPHY ... 169

CHAPTER - 0

HISTORICAL INTRODUCTION

In the year 1971 Intel Corporation took advantage of the LSI technology and the world's first microprocessor chip 4004 was introduced. While intel was working on MC-S-4 a parallel development project was under way, it would lead to introduction of the first 8 bit microprocessor the 8008, with 48 instructions.

With the introduction of intel 8080 in 1974 microprocessor took a major step forward. The 8080 was the first device with speed and power to make the microprocessor an important tool for the designer. With the 2 μs instruction cycle and 30 more instructions the 8080 offered a ten fold increase in throughput over 8008. It could directly address 64K bytes of memory versus the 16 K of 8008. The stack from within had been removed and put on to the memory, restrictions on subroutine nesting were thus removed and stack operations were expanded to include status and register saving, I/O ports were increased from 8/24 in 8008 respectively to 256 each in 8080. It could execute decimal and BCD arithmetic. The 8080 had better operand addressing modes, better interrupt processing and a 16 bit address bus, only 6 peripheral IC's were needed versus the 8008's 20 peripherals I.C.chips to make a minimum microcomputer. The 8080 was fabricated using only 5000 transistors, a highly efficient design, package size moved to 40 pin DIP from 18 pin of 8008.

Rapid acceptance and high demand for 8080 spawned many of the 8 bit microprocessors in small time interval, some of them are Zelog Z-80, 6800, Intel 8085. 8085 requires fewer peripherals than 8080 and offered additional features such as vectored interrupt and a serial I/O port:

Another 8 bit advance was the arrival late in the decade of processors such as TEXAS Instrumentation 9980, Intel's 8088 and Motorala's 6809. The above mentioned microprocessor offered 8 bit external buses but processed data internally in 16-bit words.

The approach permits full compatibility with 8 bit hardware- while also providing 16 bit software facility with little changes.

Today microprocessor era is in full swing, continuous development work is going on to design faster and better processors and some microprocessors reached the compaitibility of an average mini computer. The top of the microprocessors scale now features 16 bit data and even 32 bit microprocessors.

The most important aspect of a processor is the word length; the number of binary bits the processor can manipulate in parallel. It is quite correct to say that early 4 bit and 8 bit architectures have been surpassed by the newer 16 bit architectures. But after the development of 16 bit processors, the manufacture of small word size processors are not abondoned, since many applications, such as-

Appliance Control, traffic control,electronic games etc. do not require the added capability of a 16 bit processor and its associated cost. The most of the applications are handled comfortably by the 4 or 8 bit M/CS. For more complex control applications 16 bit processor may be essential and as such the 16 bit M/CS will be at the top of the line for next several years. However this does not mean that microprocessor development is in saturation but it clearly indicates that advancement will still be made but not by merely increasing the word length. IN BRIEF: THE DEVELOPMENT IS GOING ON TOWARDS THE IDEAL MICROPROCESSOR.

The following reasons indicate that the 16 bit m/c will dominate the high end of the microprocessor field for next few years.

1. The cost of capital equipment that is necessary for manufacture is increasing day by day.
2. In a processor based system, the CPU is becoming a less and less complex chip compared to chips that do the associated logic and control. A microprocessor is fraction of the total system into which it is incorporated; Hence with the development of a new CPU, a variety of support chips should also be assigned to provide a complete range of components for a wide variety of users. The more complicated the processor

becomes, the greater the need to expand the product line and each support device also becomes more complex.

3. Yet another reason why 16 bit processors are likely to dominate for some more years in their architecture. The 16 bit architectures are easier for the designer to work and has fewer limitations. It has many attributes of a Main Frame Computer and best suited for designing efficient compilers for working with H.L.L.

However 8086 has been considered in this dissertation. Brief discription of pin signals together with **its** architecture details are discussed in Appendix - G.

CHAPTER- 1

SALIENT FEATURES OF VMC -86

1.1 INTRODUCTION:

VMC-86 is Configured around the Intel's 16 bit microprocessor 8086 and can be used to control any industrial process and to develop software for 8086. For communicating with the outside world it has keyboard having 28 keys and 8 seven segment displayes. It also has the capacity of interfacing with teletypewritter, CRT terminal and an audio casette recorder.

The powerfull monitor has taken the space of 16K (ROM) +4K(RAM). The total memory on the board can be easily expanded to 64 K/128K bytes of EPROM and 32K/128K byte of CMOS RAM/IRAM using 6116/2186. The system has 72 programmable I/O lines through three 8255-A which can communicate with different I/O peripherals. The serial I/O communication is made possible through 8251A USART.

For control applications, three 16 bit Timer/Counter are available through 8253 timer I.C., For real time applications, the 8 level of interrupts are provided through 8259 interrupt controller. It also provides the on board facility of arithmetic processor 8231 that provides high performance fixed and floating point arithmetic and a variety of floating point trigonomatric and mathematical operations. Most beautiful thing is the battery back up facility for on board RAM. This saves the user's program in case of power failure.

The VMC-86 provides an on board EPROM programmer. The user can burn any of his/her program in 2716/2732/2732A/2764/27128.

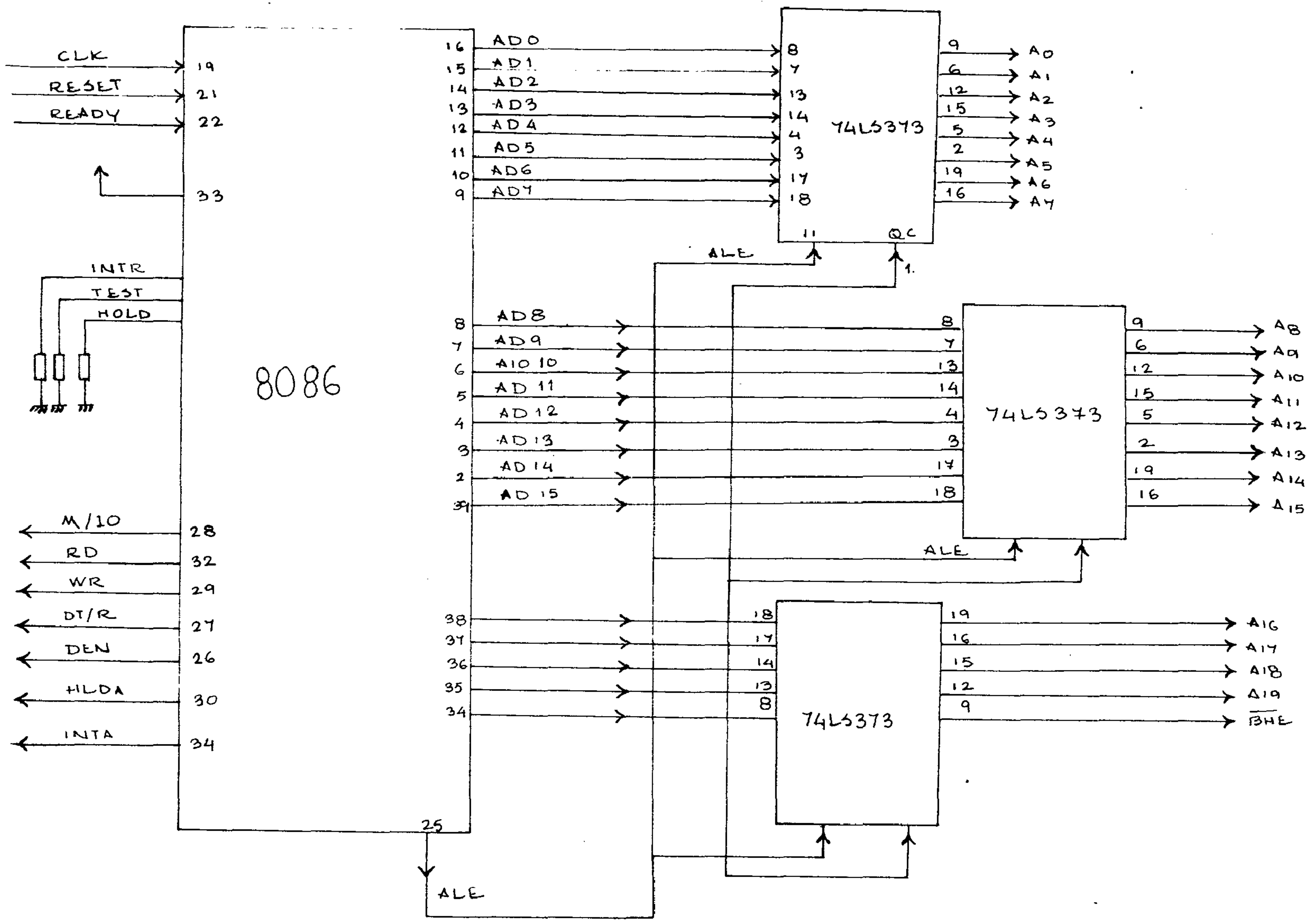
1.2 HARDWARE DESCRIPTION:

It is divided into following sections.

1.21 CPU :

8086 is a 16 bit 3rd generation micro-processor. 8086 is suitable for an exceptionally wide spectrum of microcomputer applications and this flexibility is one of its most outstanding characterstics. The 8086 has got 16 data lines and 20 address lines. The lower 16 address lines are time multiplexed with 16 data lines hence, it becomes necessary to latch the address lines. This is done by using 74LS373, as Shown in Fig. 1.1. As several of the CPU pins have dual functions that are selected by MN/$\overline{\text{MX}}$ pin. In this kit it is held logically high (minimum mode) and so these pins transfer control signals directly to memory and I/O devices. It has MULTIBUS edge connector.

The INTR, $\overline{\text{TEST}}$ and HOLD input to the 8086 are pulled down and brought out at PCB edge connector. The 8086's NMI input is connected to VCTINT key. The maskable interrupt INTR is available to the peripheral ckts through the expansion



8086
74LS373
74LS373
74LS373
CLK
RESET
READY
INTR
TEST
HOLD
M/10
RD
WR
DT/R
DEN
HLDA
INTA
ALE
ALE
ALE
ALE
QC
1.
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
AD8
AD9
A10 10
AD 11
AD 12
AD 13
AD 14
AD 15
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
$\overline{BHE}$
19
21
22
33
28
32
29
27
26
30
34
25
38
37
36
35
34

FIG 11

bus. To use the maskable interrupt an interrupt vector pointer must be provided on the data bus when $\overline{INTA}$ is active, an interrupt controller (using 8259) ckt is provided to take care of more than one source of interrupt.

1.22 CLOCK GENERATOR:

Intel 8284 clock generator/driver is being used for this purpose. The detailed pin connection description and logic symbolism of 8284 is given in Appendix-A. The input frequency to 8284 is taken from a crystal which operates at a fundamental frequency of 14.7456 MHz. The clock generator/ driver divides the crystal frequency by three to produce the 4.9 MHz.CLK signal required by CPU. In addition the clock generator performs the further division by two and outputs a 2.45 MHz. named as PCLK and is used as primary clock signal for remainder of the ckts. The two outputs of the CLOCK GENERATOR namely RDY and RST are internally synchronized to the 4.9 MHz CLK signal. RST is used to RESET the VMC-86 to an intialized state and occurs when the $\overline{RES}$ input to the CLOCK generator goes LOW i.e. when power is put ON or when system RESET key is pressed. The input RDY1 is discussed in next section (Wait State Generator).

The system can operate at either 4.9MHz or 2.5 MHz (peripheral clock). The clock is selected by jumper connection as shown in Fig. 1.2.

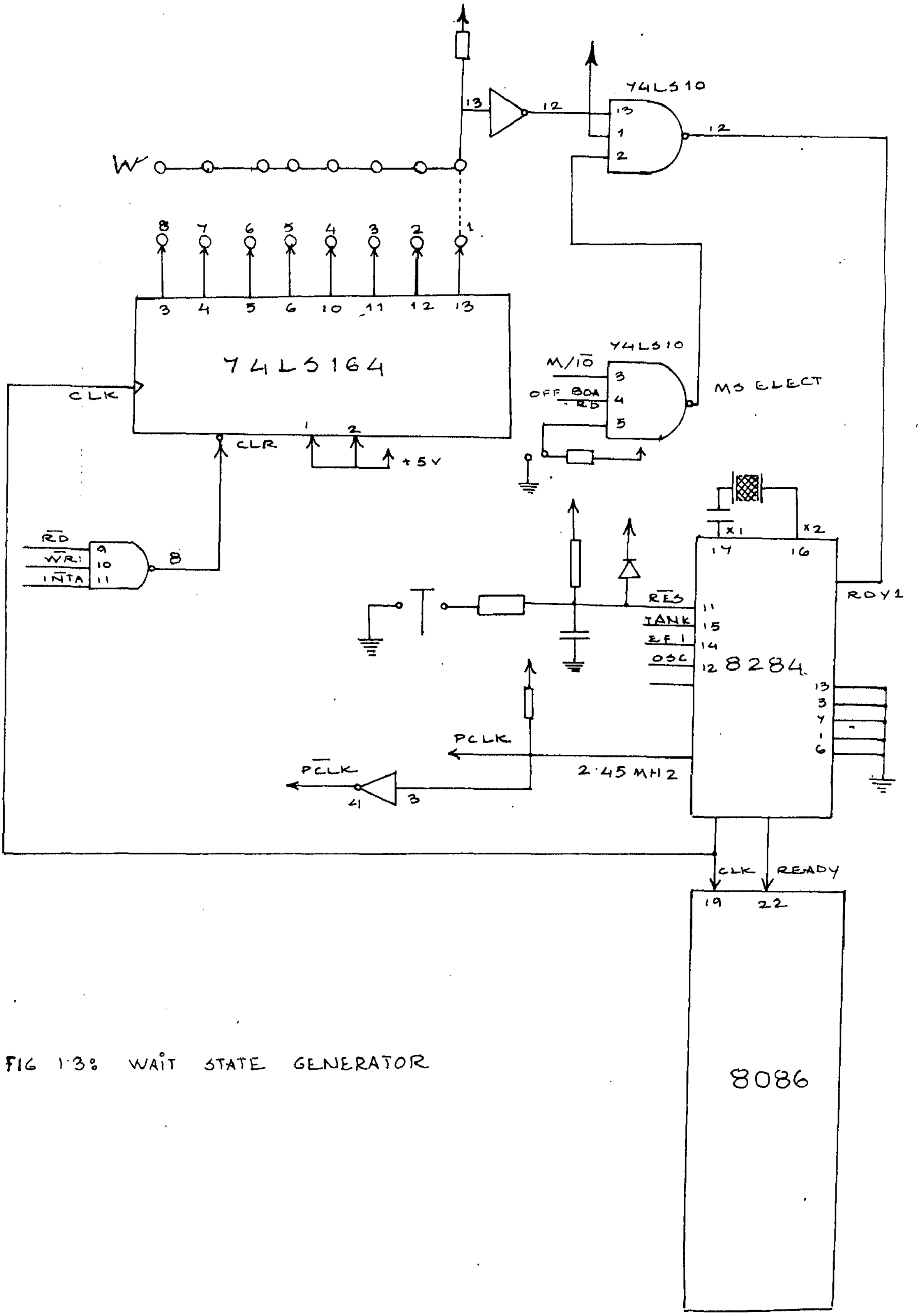


FIG 1.3: WAIT STATE GENERATOR

```
      C             B              A
      0             0 ------------ 0
   2.45 MHz        CLK          4.9 MHz
  ≈ 3 MHz                     ≈ 5 MHz
```

FIG: 1.2

At present it is in25 MHz.Configuration.

1.23 WAIT STATE GENERATOR:

The wait state ckt is provided to insert exact number of wait states into the CPU'S bus cycle to compensate for a slow peripheral I/O or memory ckt. that has been interfaced to the expansion bus or to allow on board memory and I/O operations to function correctly when the CPU is operated at 2.5 MHz. rate. The ckt diagram for WAIT state generation in VMC-86 is given in Fig. 1.3.

RDY1 input of the clock generator is ACTIVE HIGH; output of shift register (serial) is NAND'ed with MSELECT and after inverting given to RDY1. Hence as per the requirement of no. of wait states the output 1,2,3,4,5,6,7, or 8 are shorted with W through a jumper ; Input RDY1 to 8284 decides output READY of 8284; and hence input READY of 8086: hence no. of Wait states to be introduced.

1.24 DISPLAY :

It provides 8 digits of seven segment display. Four digits are for displaying address of any location or name of any register, Rest of the four digits are for displaying the contents of any Memory location or register.

1.25 MULTIBUS BUFFERING:

All data, address and control lines are available to user at PCB edge connector in the MULTIBUS CONFIGURATION. Sockets are provided on P.C.B. to buffer these lines before going to MULTIBUS.

In order to facilitate the multiprocessing operation, all address, data and necessary control lines are made bidirectional.

1.26 INTERFACE:

VMC-86 provides following interface

i. Cassete Recorder

ii. RS232C interface

iii. 20 mA Current loop interface

iv. Eprom Programmer interface (on board).

In this dissertation the RS32C interface has been used for interfacing the CRT for man-machine communication. The next chapter deals with the CRT interface in detail.

1.3 MEMORY MAPPING:

VMC-86 provides 4K Bytes of on-board RAM and 16K of EPROM Loaded with powerful monitor. The on board memory can be expanded to 128 K Byte of EPROM and 32K/128K of RAM.

It provides eight 28 pin sockets named as ROMO to ROM7 and sixteen 28 pin sockets named as RAMO to RAM15. ROM 0 to ROM 7 can be defined to have either 2716 EPROM (2K Byte each) or 2732 (4K Byte each) or 2764 EPROM (8K Byte each) or 27128 EPROM (16K Byte each). Similarly RAMO to RAM 15 can be defined to have either 6116 CMOS RAM (2K Byte each) or 2186 RAM (8K Byte each). The selection of different memory chips is achieved by making proper jumper connections in Black Box-I.

Memory mapping for present situation of VMC-86 is given in table 1.1. Therefore mapping given in table 1.1 is valid if ROMO to ROM7 are defined for 2764 and RAMO-RAM 15 are defined for 6116.. If 27128 is used for ROMO-ROM7, 2186 is used for RAMO-RAM15, the mapping will differ since 27128 is 16K 2186 is of 8K. The corresponding mapping is discussed in further sections.

1.4 I/O MAPPING:

All the I/O devices are interfaced as an isolated I/O device in VMC-86. I/O mapping is given in table 1.2. The following are the chips used.

1.41 8279 is used as Keyboard controller. It provides a scanned interface to 28 contact key matrix provided in VMC-86 and scanned displays. 8279 has got 16x8 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 micro processor. 8279 also refreshes the display RAM automatically.

TABLE 1.1 MEMORY MAPPING

ADDRESS IN HEX	MEMORY SIZE (BYTES)	ROM/RAM	IC USED	REMARKS
FE000-FFFFF	8K	ROM	2764	SYSTEM MONITOR
FC000-FDFFF	8K	ROM	2764	
FA000-FBFFF	8K	ROM	2764	
F8000-F9FFF	8K	ROM	2764	
F6000-F7FFF	8K	ROM	2764	
F4000-F5FFF	8K	ROM	2764	
F2000-F3FFF	8K	ROM	2764	
F0000-F1FFF	8K	ROM	2764	
08000-EFFFF	8K	ROM/RAM	-	Space available for further expansion through multibus connector.
07800-07FFF	2K	RAM	6116	
07000-077FF	2K	RAM	6116	
06800-06FFF	2K	RAM	6116	ON BOARD
06000-067FF	2K	RAM	6116	EXPANSION
05800-05FFF	2K	RAM	6116	OF
05000-067FF	2K	RAM	6116	RAM
04800-04FFF	2K	RAM	6116	
04000-047FF	2K	RAM	6116	

contd....

TABLE 1.1 contd...

03800-03FFF	2K RAM	6116	
03000-037FF	2K RAM		
02800-02FFF	2K RAM		
02000-027FF	2K RAM		
01800-01FFF	2K RAM		
01000-017FF	2K RAM		PRESENTLY
00800-00FFF	2K RAM		AVAILABLE
00200-007FF	1.5K RAM		
00000-001FF	0.5K RAM		AREA RESERVED FOR MONITOR

1.42. <u>8255:</u>

To interface peripheral equipments to the system bus 8255 (PPI) are provided which basically act as general purpose I/O component. 9 Input/Output ports each of 8 lines are provided using three 8255. The port addresses are given in table 1.2 .

1.43 <u>8253:</u>

It 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 multiplier, real time clock etc. It has three independent counter, each has a count rate of 2 MHz CLK, GATE and OUT signals of this timer are brought at J3 Euro connector at Right most top corner of the unit.

1.44 8251 :

This chip is provided to make possible the serial interface of CRT and TTY.

1.45 8259 :

The 8259 is a device specifically designed for use in real time, interrupt driven microcomputer systems. It manages eight levels of requests and has built in features for expandability to other8259'S. It is programmed by system's software as an I/O peripheral. A selection of priority modes is available to the programmer so that the manner in which the requests are processed by 8259 can be configured to match his/her system requirements. The priority modes can be changed or reconfigured dynamically at any time during the main program.

TABLE 1.2

I/O MAPPING

DEVICE NAME	PORT NAME	PORT ADDRESS	REMARKS
8255-I	PORT A PORT B PORT C CONTROL WORD	FFF9 FFFB FFFD FFFF	A_2A_1 Goes to A_1A_0 of 8255 and A_3A_0=11 along with A_4=1 is used for CS.

contd....

DEVICE NAME	PORT NAME	PORT ADDRESS	REMARKS
8255-2	PORT A	FFF8	A_2A_1 Goes to A_1A_0 of 8255-2 and A_3A_0=10 along with A_4=1 is used for CS
	PORT B	FFFA	
	PORT C	FFFC	
	CONTROL WORD	FFFF	
8255-3	PORT A	FFF1	A_2A_1 Goes to A_1A_0 of 8255-3 and A_3A_0=01 along with A_4=1 is used for CS
	PORT B	FFF3	
	PORT C	FFF5	
	CONTROL WORD	FFF7	
8279	DATA WORD	FFE8 or FFEC	A_2 is redundant,
	COMMAND WORD	FFEA or FFEE	A_1 goes to A_1 of 8279 A_3A_0=10 is used for CS
8253	COUNTER 0	FFE1	A_2A_1 goes to A_2A_1 of 8253 and A_3A_0=01 along with A_4=0 is used for CS
	COUNTER 1	FFE3	
	COUNTER 2	FFE5	
	CONTROL WORD	FFE7	
8251	DATA WORD	FFF0 OR FFF4	A_1 goes to $C/\overline{D}$ of 8251 A_3A_0=00 along with A_4=1 is used for CS
	COMMAND WORD	FFF2 OR FFF6	
8231	DATA WORD	FFE0 OR FFE4	A_1 goes to A_1 of 8231 A_3 A_0=00 along with A_4=0 is used for CS
	COMMAND WORD	FFE2 OR FFE6	
8259	DATA WORD	FFE9 OR FFED	A_1 goes to A_1 of 8259 A_3A_0=11 along with A_4=0 is used for CS
	COMMAND WORD	FFEB OR FFEF	

1.5 RAM MEMORY DECODING:

Refering to Ram memory mapping given in table 1.1 the address varies from 00000 - 07FFF. From table 1.1, it is very clear that A_{19}, A_{18}, A_{17}, A_{16} are zero always. Further A_{15}, A_{14}, A_{13}, A_{12}, make a combination varying from 0 to 7, which means A_{15} is also zero always, these lines in conjunction of $M/\overline{IO}$ are used to generate a single singnal names $\overline{\text{SELECT RAM}}$. The actual configuration is given in Fig.1.4.

A_{14}-A_{12} may have any combination varying from 0-7. From memory mapping addresses it is clear that this combination defines that, which RAM is being selected. Hence these lines are decoded through 3 lines to 8 line decoder and connected to the CS signals of the corresponding RAM's as shown in Fig. 1.5. The address line A_0 is being used in conjunction with $\overline{BHE}$ to generate $\overline{WR}$ signals for upper and lower bank namely $\overline{WEU}$, $\overline{WEL}$ $\overline{WEU}$ and $\overline{WEL}$ are generated as shown in Fig. 1.6.

These $\overline{WEL}$ and $\overline{WEU}$ signals are connected to the $\overline{WR}$ pin of all lower and upper banks Ram chips- respectively.

1.51 MEMORY DECODING FOR 2186 RAM:

2186 is 8K bytes integrated dynamic memory, the memory mapping in VMC-86 using 2186 is given in Table 1.3 Referring to the pin connection diagram of 2186 in Appendix B, it has 13 address lines i.e. A_0-A_{12}. A_0 is being used in conjunction with $\overline{BHE}$ to select upper or lower memory bank at a time, A_1-A_{13} are directly connected to A_0 - A_{12} .

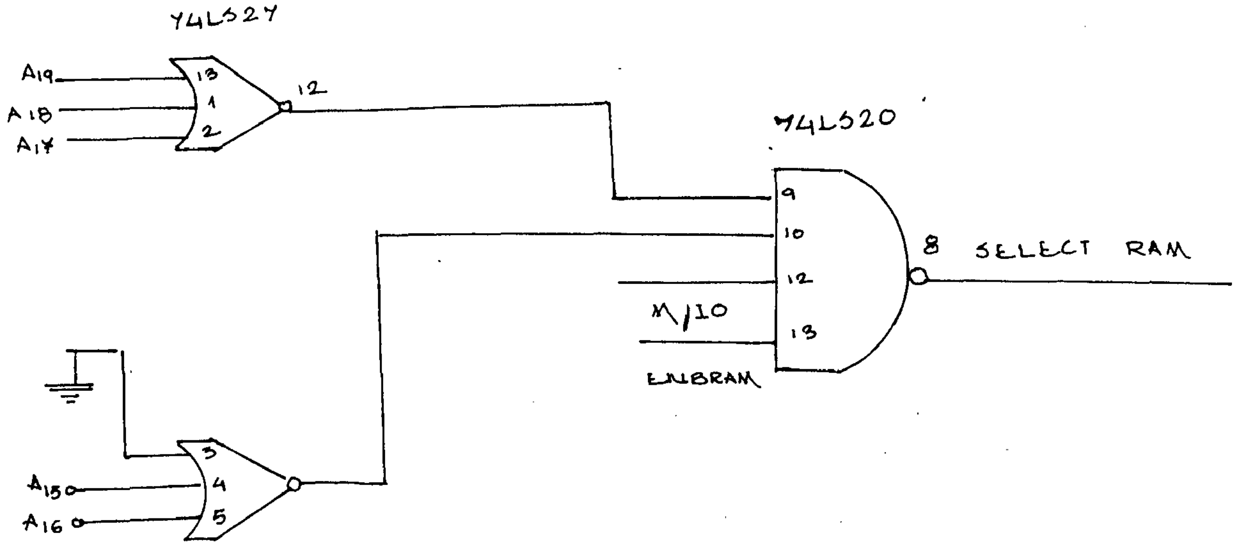


FIGURE - 1·4

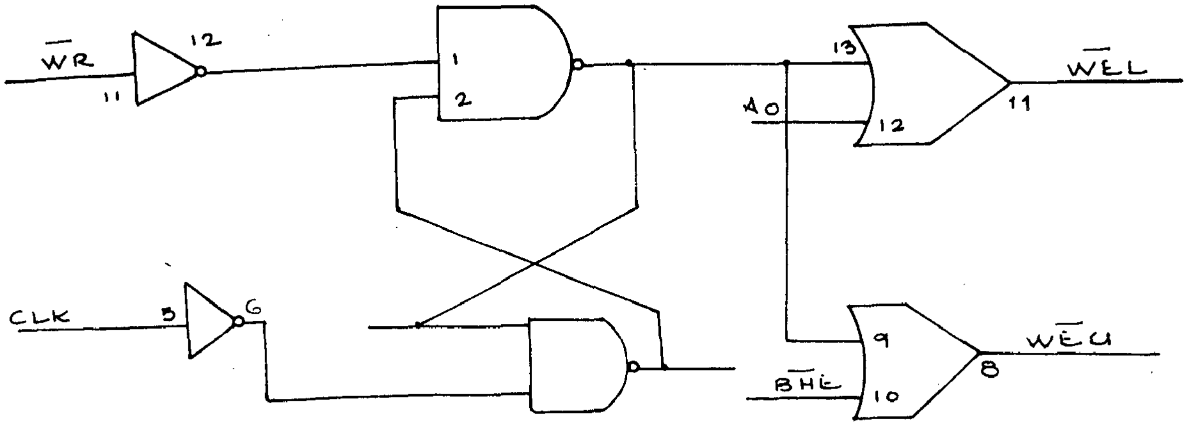


FIGURE 1·6

pins of 2186. A_{14}, A_{15}, A_{16} are decoded by 3 line to 8 line decoder 74LS138 in conjunction with SELECTRAM and connected to $\overline{CS}$ pin. Actually A_{14}, A_{15} and A_{16} lines are brought at pin no. 8,7,6 of B.B.I and also the input to 74 LS138 are at pin no. 1,2,3. As per the chip used i.e. 6116 or 2186 the connections are to be changed. It is important to note that $\overline{CS}$ for corresponding upper and lower memory bank is same.

TABLE 1.3

MEMORY MAPPING FOR 2186 RAM

ADDRESS	CORROSPONDING CHIP
00000 - 01FFF	RAM0
02000 - 03FFF	RAM1
04000 - 05FFF	RAM2
06000 - 07FFF	RAM3
08000 - 09FFF	RAM4
0A000 - 0BFFF	RAM5
0C000 - 0DFFF	RAM6
0E000 - 0FFFF	RAM7
10000 - 11FFF	RAM8
12000 - 13FFF	RAM9
14000 - 15FFF	RAM10
16000 - 17FFF	RAM11
18000 - 19FFF	RAM12
1A000 - 1BFFF	RAM 13
1C000 - 1DFFF	RAM14
1E000 - 1FFFF	RAM15

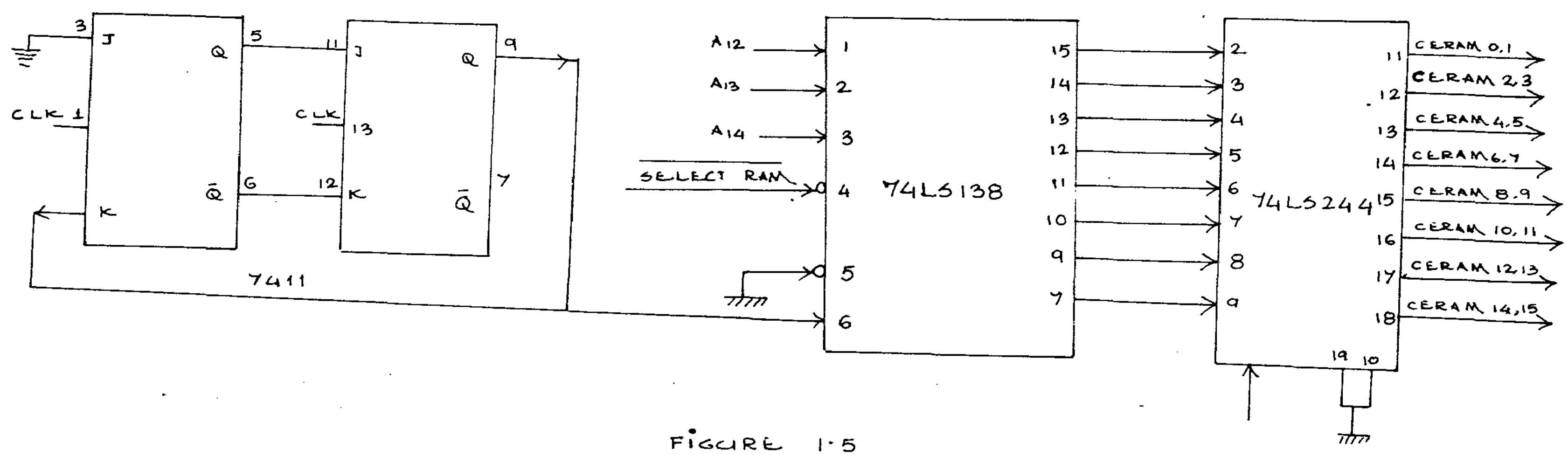

3
J
Q
5
CLK 1
Q̄
G
K
11
J
Q
9
CLK
13
12
K
Q̄
7
7411
A12
A13
A14
SELECT RAM
1
2
3
4
5
6
74LS138
15
14
13
12
11
10
9
7
2
3
4
5
6
7
8
9
74LS244
11
12
13
14
15
16
17
18
19
10
CERAM 0,1
CERAM 2,3
CERAM 4,5
CERAM 6,7
CERAM 8,9
CERAM 10,11
CERAM 12,13
CERAM 14,15

FIGURE 1·5

1.6 I/O DECODING :

In conjunction with Fig. 2.2, Table 1.2 and Fig. 1.7 give clear picture of I/O decoding. Since all the I/O ports for general purpose used in VMC-86 are 8 bit ports, hence $\overline{BHE}$ signal is used to permit the byte operation to be performed. From Fig. 1.7 it is clear that when $\overline{BHE}$ signal will be HIGH then only the CSI1, CSI3, CSI5 and CSI7 will be ACTIVE LOW.

1.7 ROM DECODING :

In VMC-86, at present, 2764 ROM chip is being used. Refering to pin connection diagram given in Appendix-B, 2764 has 13 address lines. Refering to memory mapping of ROM (2764) given in table 1.1, A_{16}, A_{17}, A_{18} and A_{19} are always 1. Two 74LS156 are being used to decode A_{13}, A_{14}, A_{15} along with A_{16}, thus it provides 16 blocks of 4K word each namely A to P (refering to Fig. 1.4). This slot of 4K word can be further divided into two slots of 2K word each in Fig. 1.4. 74LS-32 is being used to decode two blocks of 4K words alongwith A_{12} and $\overline{A}_{12}$ to get 3 blocks of 2K word each. The decoding circuitry is very clearly shown in Fig. 1.8.

In VMC-86 the total ROM area in general consists of 8 sockets. Since 2 chips should be selected at a time hence 4 OE signals have been generated namely OE1, OE2, OE3, and OE4. Refering to the manual, the points A to P are being brought to the Black Box II and also the desired pins of the ROM sockets, are being brought at Black Box II. According to, which ROM chip

(Amongst the mentioned one) is being used one has to make the connections accordingly.

For 2764 the connections in the Black Box 2 would be as given below:

14 - 17
13 - 16
and 12 - 15

TABLE 1.4

ROM MEMORY MAPPING FOR 2716

ADDRESS IN HEX	MEMORY SIZE IN BYTES	SOCKET NO
FF800-FFFFF	2K	0
FF000-FF7FF	2K	4
FE800-FEFFF	2K	1
FE000-FE7FF	2K	5
FD800-FDFFF	2K	2
FD000-FD7FF	2K	6
FC800-FCFFF	2K	3
FC000-FC7FF	2K	7

TABLE 1.5

ROM MEMORY MAPPING FOR 2732

ADDRESS IN HEX	MEMORY SIZE	SOCKET NO
FF000 - FFFFF	4K	0
FE000 - FEFFF	4K	4
FD000 - FDFFF	4K	1
FC000 - FCFFF	4K	5
FB000 - FBFFF	4K	2
FA000 - FAFFF	4K	6
F9000 - F9FFF	4K	3
F8000 - F8FFF	4K	7

TABLE 1.6

ROM MEMORY MAPPING FOR 27128

ADDRESS IN HEX	MEMORY SIZE	SOCKET NO
FC000 - FFFFF	16K	0
F8000 - FBFFF	16K	4
F4000 - F7FFF	16K	1
F0000 - F3FFF	16K	5
EC000 - EFFFF	16K	2
E8000 - EBFFF	16K	6
E4000 - E7FFF	16K	3
E0000 - E3FFF	16K	7

1.8 VMC-86 IN MULTIPROCESSING MODE: EXPANSION THROUGH MULTIBUS:

The multibus is a general purpose multiprocessing system bus. Any one,designing multiprocessing systems should consider building his/her systems around the multibus for two important reasons.

1. To save the time and costs associated with developing a new system.
2. To gain compatibility with a wide variety of products available for the multibus.

The multibus provides a versatile communications channel that can be used to coordinate a wide variety of computing modules. In VMC-86 address, Data and Control lines of 8086 are buffered and are made available at PCB edge connector,as per the MULTIBUS STANDARD. Thus the system can also be used for multiprocessing applications.

The tristate control logic of the buffers is brought separately on VMC-86 board at the left most top corner. This logic can be controlled from outside. The jumper connections required for operating the system in multiprocessing mode are given in Fig. 1.9.

F	O	MP
E	O	TSC
D	O	GND

Fig. 1.9

For multiprocessing application E should be shorted to F Otherwise E should be shorted to D. In fact MN/$\overline{\text{MX}}$ pin connection is brought to point E and F,D points are ground and supply points respectively. When using in minimum mode as is the case at present, MN/$\overline{\text{MX}}$ is connected to Vcc and in multiprocessing systems MN/$\overline{\text{MX}}$ would be grounded.

The pin signals of MULTIBUS are given in Table 2. A functional Description of the signals follows:

The multibus signal lines are divided into following sections:

1. INITIALIZATION SIGNAL LINE
2. ADDRESS AND INHIBIT LINES
3. BUS CONTENTION RESOLUTION LINES
4. INFORMATION TRANSFER PROTOCOL LINES
5. ASYNCHRONOUS INTERUPT LINES
6. POWER SUPPLY LINES
7. RESERVED LINES.

1. <u>INITIALIZATION SIGNAL LINE</u>

INIT : The initialization signal resets the entire system to a predetermined state. $\overline{\text{INIT}}$ may be supplied by one of the bus masters or by external logic.

2. ADDRESS AND INHIBIT LINES

(a) $\overline{\text{ADR0}}$ - $\overline{\text{ADR13}}$ 20 address lines are used to transmit the address of the memory location or I/O port to be acessed. $\overline{\text{ADR13}}$ is the most significant bit, while $\overline{\text{ADR0}}$ is the least significant bit. 8 bit bus masters use 16 address lines ($\overline{\text{ADR0}}$ - $\overline{\text{ADRF}}$) to address memory and 8 address lines $\overline{\text{ADR0}}$ - $\overline{\text{ADR7}}$ to select I/O ports. 16 bit bus masters address memory via all 20 address lines and select I/O ports via the low order 12 lines.

(b) $\overline{\text{INH1}}$: The inhibit RAM signal prevents RAM memory devices from responding to the address on the address bus. $\overline{\text{INH1}}$ allows ROM memory devices to override RAM devices when ROM and RAM memory are assigned the same memory space.

(c) $\overline{\text{INH2}}$: The inhibit ROM signal prevents ROM memory devices from responding to the address on the address bus. $\overline{\text{INH2}}$ allows auxiliary ROM to override ROM devices when ROM and auxiliary ROM memory are assigned the same memory space.

$\overline{\text{INH1}}$ and $\overline{\text{INH2}}$ may also be used to allow memory mapped I/O devices to override RAM and ROM devices respectively.

(d) $\overline{\text{BHEN}}$: $\overline{\text{BHEN}}$ is used to specify that data will be transferred on the high order 8 data lines of the multibus. This signal is used in systems that utilize 16 bit memory or I/O modules.

3. DATA LINES

(a) $\overline{\text{DAT0-DATF}}$: The 16 bidirectional data lines are used to exchange information with a memory location or I/O port. $\overline{\text{DATF}}$ is the most significant bit, although in 8 bit systems only lines $\overline{\text{DAT0-DAT7}}$ are used and $\overline{\text{DAT7}}$ becomes the most significant bit. $\overline{\text{DAT0}}$ is the least significant bit always.

4. BUS CONTENTION RESOLUTION LINES

(a) BCLK : The negative edge of the Bus clock is used to synchronize bus contention. $\overline{\text{BCLK}}$ is asynchronous with the CPU Clock. $\overline{\text{BCLK}}$ may be slowed, stopped or single stepped during debugging.

(b) CCLK : The constant clock provides a clock signal of constant unspecified frequency.

(c) BPRN : The Bus priority in signal tells a bus master that no higher priority device is requesting use of the system bus. $\overline{\text{BPRN}}$ is synchronized with $\overline{\text{BCLK}}$. This signal is 'daisy chained' if serial priority arbitration is used, when using parallel priority arbiteration, a bus arbiter generates $\overline{\text{BPRN}}$.

(d) $\overline{\text{BPRO}}$: This is a Bus Priority Out Signal. Like $\overline{\text{BPRN}}$, $\overline{\text{BPRO}}$ is 'daisy chained' when serial priority arbiteration is used; $\overline{\text{BPRO}}$ is fed to the $\overline{\text{BPRN}}$ input of the next lower priority module. When using parallel priority arbiteration, a bus arbiter must provide this signal. $\overline{\text{BPRO}}$ is synchronized with $\overline{\text{BCLK}}$.

(e) $\overline{BUSY}$: The bus busy signal is supplied by the current bus master to indicate that, the system bus is in use. $\overline{BUSY}$ is used by other devices to determine whether or not they may acquire control of system bus. $\overline{BUSY}$ is synchronized with $\overline{BCLK}$.

(f) BREQ : The Bus Request Signal is used by devices to indicate that they wish to become bus master. $\overline{BREQ}$ is synchronized with $\overline{BCLK}$; it is not bussed on the mother board.

(g) $\overline{CBRQ}$: $\overline{CBRQ}$ is used by all potential bus masters to inform the current bus master that another master wishes to use the bus, if $\overline{CBRQ}$ is high, the current bus master knows that no other device is requesting the bus therefore the present bus master is to retain the bus.

5. INFORMATION TRANSFER PROTOCOL LINES :

A bus master that has control of the system bus generates all data transfer control signals. All address signals (and data signal when a write is to occur) must be stable at least 50 ns prior to the transfer control signal pulse and must remain valid for at leat 50 ns after the control signal pulse is removed.

Information transfer protocol lines are not synchronous with $\overline{BCLK}$.

(a) $\overline{\text{MRDC}}$: The Memory Read Control indicates that the address of a memory location has been placed on the address lines and that the contents of the address location are to be placed on data lines.

(b) $\overline{\text{MWTC}}$: The Memory Write Control indicates that the address of the memory location has been placed on the address lines and data has been placed on the system data lines, the data is to be written into the addressed memory location.

(c) $\overline{\text{IORC}}$: The I/O Read Control indicates that the address of an input port has been placed on the system address lines, and the data at that input port is to be placed on the data lines.

(d) $\overline{\text{IOWC}}$: The I/O Write Control indicates that the address of an output port has been placed on the system data lines, the data is to be output to addressed port.

(e) $\overline{\text{XACK}}$: All exchanges involve handshaking. Therefore, the selected bus.slave must provide the bus master with an acknowledge signal in response to the transfer control signal. The transfer acknowledge signal is the required response that indicates that the specified operations has been completed.

(f) $\overline{\text{AACK}}$: The advanced acknowledge signal is used by 8080 A Microprocessors. $\overline{\text{AACK}}$ is an advance acknowledge that allows the CPU to complete a specified operation without entering a wait state. Bus slaves that provide $\overline{AACK}$ must also provide $\overline{XACK}$. This requirement must be met since not all bus masters will respond to the $\overline{AACK}$ signal.

6. ASYNCHRONOUS INTERRUPT LINES :

(a) INTO-INT7 : These priority interrupt request lines are used with parallel interrupt resolution circuitery. $\overline{INT7}$ has the lowest priority, $\overline{INTO}$ the highest priority.

(b) $\overline{INTA}$: $\overline{INTA}$ is used by a bus master to request that external logic place interrupt vector information on the Multibus Data Lines.

7. POWER SUPPLY LINES :

Various regulated power supply lines are provided on the bus. Each module must provide both bulk decoupling and high frequency decoupling local to the resident logic devices.

8. RESERVED LINES:

These are the lines left available for future intel definition.

How the above signals are brought to the PCB connector in VMC-86 is shown in Fig. 1.10 (a).

The multi bus requires a delay from valid address of 50ns before a Read Control signal can be transmitted to a selected device. The read control pulse must be at least 100ns wide, and the address must remain stable at least 50ns after the Read Control signal terminates. If the selected device requires more than the 100 ns read signal or the 150 ns minimum specified address access time, then the device may extend, the read cycle by using $\overline{\text{XACK}}$ (Transfer Acknowledge) signal. This signal is equivalent to the Ready signal connected to 8284 RDY input. $\overline{\text{XACK}}$ is'Normally not ready', it is driven active to tell the CPU that the device is ready to receive or transmit data and to allow termination of bus cycle. The Multibus specifies data setup and hold times relative to $\overline{\text{XACK}}$ signal, rather than a Read or Write Control Signal, to allow autonomous operations of the selected device in a multiprocessing systems with mixed CPU types.

The Write bus cycle is similar to the Read bus cycle Written data must be valid a minimum of 50ns prior to the Write Control signal and must be held valid a minimum of 50ns following the write control signal.

Master modules attached to the Multibus must not violate the minimum setup and hold times or control pulse widths. Many designs provide better than minimum margins when running at their maximum band width. Slave modules must be able to tolerate the minimum set up and hold times but may extend the access times if they delay return of $\overline{\text{XACK}}$ by an appropriate amount.

INTERRUPT HANDLING IN MULTIPROCESSING SYSTEM:

The multibus provides two basic interrupt handling methods. These are:

1. A method whereby interrupt vectors are not transferred on the bus. Rather they are generated by the bus master's interrupt controller. The slave that requests the interrupt must be part of the same module as the bus master. If the interrupting slave is part of another module, then the slave will use the Multibus interrupt request lines $\overline{\text{INT0}}$-$\overline{\text{INT7}}$ to request an interrupt; this interrupt will be processed by the bus master's interrupt controller.

2. A method where the interrupt vector is transferred on the bus. When a slave device requests an interrupt, interrupt control logic interrupts the processor. The processor acknowledge the interrupt by lowering the $\overline{\text{INTA}}$ line and locking the system bus. This allows an interrupt vector to be transferred. Following the initial $\overline{\text{INTA}}$ cucle, interrupt control logic determines the address of the highest priority slave currently requesting an interrupt. This address is placed on the address bus. The addressed slave responds by transmitting an interrupt vector address back to the master.

In addition to providing a standard asynchronous data transfer protocol and timing specifications for designing master and slave modules, the multibus provides a standard protocol which multiple masters use to exchange bus control. To allow asynchronous masters to share the bus, the multibus maintains its own clock signal independent of clock signals local to modules that might connect to the multibus. The multibus clock signal $\overline{\text{BCLK}}$ synchronizes asynchronous requests for bus access. This allows arbitration logic to resolve priorities and grant access to one master at a time.

The highest priority master has $\overline{\text{BPRN}}$ grounded. The parity enable output $\overline{\text{BPRO}}$ from each master is connected to the priority input $\overline{\text{BPRN}}$ of the next lowest priority master. If that master does not need the bus, it propogates its $\overline{\text{BPRN}}$ to $\overline{\text{BPRO}}$. A master that needs the bus, will output $\overline{\text{BPRO}}$ high. One more signal is included to indicate an idle or not busy status of the multibus i.e. $\overline{\text{BUSY}}$ and this signal is common to all buses.

TABLE 2

PIN ASSIGNMENT OF BUS SIGNALS ON MULTIBUS BOARD CONNECTOR

COMPONENT SIDE

	PIN	MNEMONIC	DESCRIPTION
POWER SUPPLIES	1	GND	Signal Ground
	3	+ 5V	+5V dc
	5	+ 5V	+5V dc
	7	+12V	+12V dc
	9	-5V	-5V dc
	11	GND	Signal GND

contd...

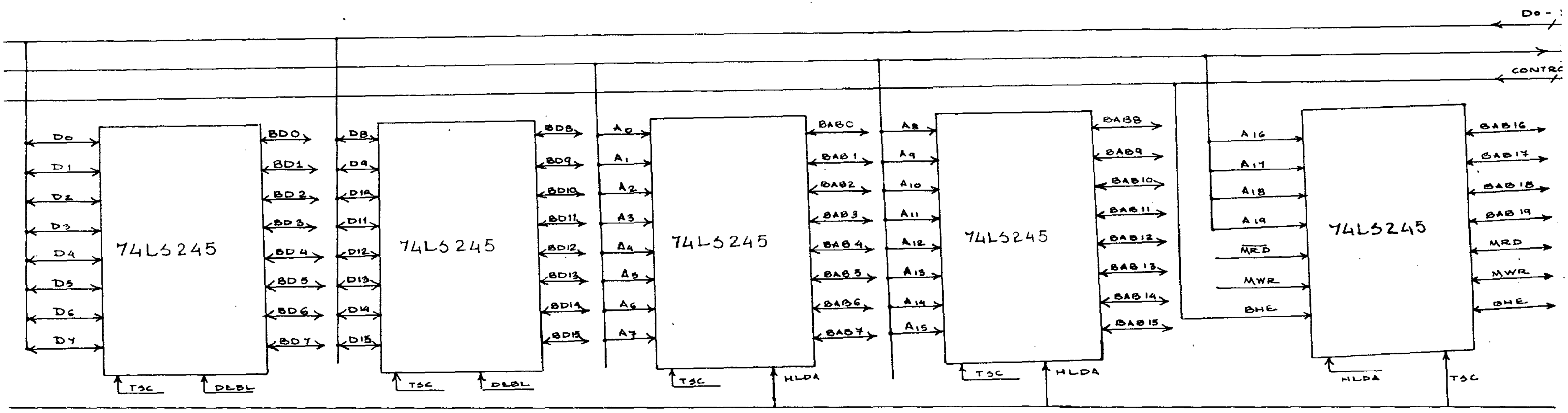

Do -
CONTRO
74LS245
74LS245
74LS245
74LS245
74LS245
D0
D1
D2
D3
D4
D5
D6
D7
BD0
BD1
BD2
BD3
BD4
BD5
BD6
BD7
D8
D9
D10
D11
D12
D13
D14
D15
BD8
BD9
BD10
BD11
BD12
BD13
BD14
BD15
A0
A1
A2
A3
A4
A5
A6
A7
BAB0
BAB1
BAB2
BAB3
BAB4
BAB5
BAB6
BAB7
A8
A9
A10
A11
A12
A13
A14
A15
BAB8
BAB9
BAB10
BAB11
BAB12
BAB13
BAB14
BAB15
A16
A17
A18
A19
$\overline{MRD}$
MWR
BHE
BAB16
BAB17
BAB18
BAB19
MRD
MWR
BHE
TSC
DEBL
TSC
DEBL
TSC
HLDA
TSC
HLDA
HLDA
TSC

FIGURE 1·10 - (a)

	PIN	MNEMONIC	DESCRIPTION
BUS CONTROLS	13	BCLK/	Bus Clock
	15	BPRN/	Bus Pri. Inhibitor
	17	BUSY/	Bus Busy
	19	MRDC/	Mem Read Cmd
	21	IORC/	I/O Read Cmd.
	23	XACK/	XFER Acknowledge
BUS CONTROLS AND ADDRESS	25		RESERVED
	27	BHEN/	BYTE HIGH ENABLE
	29	CBRQ/	COMMON BUS REQUEST
	31	CCLK/	CONSTANT CLOCK
	33	INTA/	INTR.ACKNOWLEDGE
INTERRUPTS	35	INT6/	Parallel Interrupt Requests.
	37	INT4/	
	39	INT2/	
	41	INT0/	
ADDRESS	43	ADRE/	Address Bus
	45	ADRC/	
	47	ADRA/	
	49	ADR8/	
	51	ADR6/	
	53	ADR4/	
	55	ADR2/	
	57	ADR0/	

	PIN	MNEMONICS	DESCRIPTION
DATA	59	DATE/	Data Bus
	61	DATC/	
	63	DATA/	
	65	DAT8/	
	67	DAT6/	
	69	DAT4/	
	71	DAT2/	
	73	DATO/	
	75	GND	Signal Ground
	77	-	Reserved
	79	-12V	-12 dc.
	81	+ 5V	+ 5V d.c.
	83	+ 5V	+ 5V d.c.
	85	GND	Signal Ground.

CIRCUIT SIDE

	PIN	MNEMONICS	DESCRIPTION
POWER SUPPLIES	2	GND	Signal Ground
	4	+5V	+5V d.c.
	6	+5V	+5V d.c.
	8	+12V	+12V d.c.
	10	-5V	-5V d.c.
	12	GND	Signal GND
BUS CONTROLS	14	INIT/	INITIALIZE
	16	BPRQ/	Bus Pri Cut
	18	BREQ/	Bus Request
	20	MWTC/	Mem. Write Cmd.
	22	IOWC/	I/O Write Cmd.
	24	INH1/	Inhibit 1 disable RAM

	PIN	MNEMONIC	DESCRIPTION
BUS CONTROLS AND ADDRESS	26	INH2/	Inhibit 2 disable PROM or ROM
	28	AD10/	Address bus
	30	AD11/	
	32	AD12/	
	34	AD13/	
INTERRUPT	36	INT7/	Parallel Interrupt Requests
	38	INT5/	
	40	INT3/	
	42	INT1/	
ADDRESS	44	ADRF/	Address Bus
	46	ADRD/	
	48	ADRB/	
	50	ADR9/	
	52	ADR7/	
	54	ADR5/	
	56	ADR3/	
	58	ADR1/	
DATA	60	DATF	Data Bus
	62	DATD	
	64	DATB	
	66	DAT9	
	68	DAT7	
	70	DAT5	
	72	DAT3	
	74	DAT1	
POWER SUPPLIES	76	GND	
	78	RESERVED	
	80	-12V	
	82	+5V	
	84	+5V	
	86	GND	

The details of the connector available in VMC-86 follows.

DETAILS OF CONNECTOR J1

PIN NO	DESCRIPTION	PIN NO	DESCRIPTION
1	GND	8	NC
2	Vcc(+5V)	9	-12V
3	GND	10	+12V
4	Vcc(+5V)	11	NC
5	Battery	12	Programming voltage (+24V/21V)
6	GND	13	NC
7	NC	14	

DETAILS OF CONNECTOR J2:

PIN	SIGNAL	PIN	SIGNAL
1	P1C4	14	P1B1
2	P1C5	15	P1A6
3	P1C2	16	P1A7
4	P1C3	17	P1A4
5	P1C0	18	P1A5
6	P1C1	19	P1A2
7	P1B6	20	P1A3
8	P1B7	21	P1A0
9	P1B4	22	P1A1
10	P1B5	23	P1C6
11	P1B2	24	P1C7
12	P1B3	25	GND
13	P1B0	26	GND

DETAILS OF CONNECTOR J3

PIN	SIGNAL	PIN	SIGNAL
1	Vcc	33	P3A4
2	Vcc	34	P3A5
3	GND	35	P3A6
4	GND	36	P3A7
5	P2A0	37	P3B0
6	P2A1	38	P3B1
7	P2A2	39	P3B2
8	P2A3	40	P3B3
9	P2A4	41	P3B4
10	P2A5	42	P3B5
11	P2A6	43	P3B6
12	P2A7	44	P3B7
13	P2B0	45	P3C0
14	P2B1	46	P3C1
15	P2B2	47	P3C2
16	P2B3	48	P3C3
17	P2B4	49	P3C4
18	P2B5	50	P3C5
19	P2B6	51	P3C6
20	P2B7	52	P3C7
21	P2C0	53	CLK0
22	P2C1	54	GATE0
23	P2C2	55	OUT0
24	P2C3	56	CLK1
25	P2C4	57	OUT1
26	P2C5	58	GATE1
27	P2C6	59	CLK2
28	P2C7	60	OUT2
29	P3A0	61	GATE2
30	P3A1	62	MP
31	P3A2	63	} Not used
32	P3A3	64	

DETAILS OF CONNECTOR J4

PIN	DESCRIPTION	
1	DTRL	(RS232C)
2	RTS	(RS232C)
3	CTS	(RS232C)
4	DSR	(RS232C)
5	TX+	(RS232C)
6	Rx+	(RS232C)
7	GND	
8	EAROUT	} Audio-Casette interface
9	MIC IN	
10	Rx -	(20 mA loop)
11	Tx-	(20 mA loop)
12	Tx^+	(20 mA loop)
13	Rx^+	(20 mA loop)

CHAPTER -2

INTERFACING OF CRT WITH 8086

INTRODUCTION:

Two major hardware operations that a microprocessor performs are reading data from an input device and writting data to an output device. In most microcomputer applications, the CPU must communicate with a variety of I/O devices. The information that passes between CPU and these peripheral devices can be classified as either data or control. Data are typically numeric and alphanumeric information encoded in some suitable binary code, such as straight binary, BCD or ASC11. Control information is usually one of several types commands from the CPU, requests for service from I/O devices, control codes from CPU or status code from IO devices. In all but the simplest microprocessor systems the transmission of this information between micro processor and I/O devices is the critical part of the system design. This chapter deals with the interface of CRT with CPU.

CRT INTERFACE :

A peripheral device can be interfaced in two ways:

1. I/O mapped (Isolated I/O)
2. Memory mapped

In VMC-86 Isolated I/O method is used for interfacing an CRT. Before discussing C.R.T. interface, explaination of few important terms often used in such interface are in order:-

PARALLEL AND SERIAL TRANSMISSION :

As mentioned above in various instances, it is desirable to transmit binary data from one system to other. In such situations the data can be transmitted using either parallel or serial transmission techniques.

In parallel transmission each bit of the binary data is transmitted over a separate wire or line at the same instant of time ,while in serial transmission only one line is used to transmit the complete binary data bit by bit. In this technique data are usually sent starting with the least significant bit. In order to differentiate amongst various bits, a clock signal is used. Typical I/O device which transmits and receivesdata serially are CRT, teletype, casette recorder and so on.

SERIAL I/O IS MORE COMMON THAN PARALLEL I/O. In many situation it may be uneconomical or impossible to use parallel data transfer. Parallel data transfer becomes costly when many remotely placed devices are connected to the processor, in such a situation a large number of costly cables would be required for data transmission. To avoid this, serial data transmission is employed. In serial data transfer since the data is transfered bit by bit on a single line and hence reducing no. of cables and also no. of drivers/receivers required (if any). In some cases the serial method is the only method for transmitting the data for e.g. if data are to be transmitted over transmission lines, parallel data will have to be suitably coded into a serial format and then transmitted.

Serial data transmission can be divided into two types, namely

Synchronous

Asynchronous

SYNCHRONOUS SERIAL DATA TRANSMISSION:

The basic feature of synchronous serial data transmission is that, the data are transmitted or received based on a clock signal. After deciding on a specific rate of data transmission, commonly known as baud rate (bits per second). The transmitting device sends a data bit at each clock pulse. In order to interpret data correctly the receiving end must know the start and end of each data unit. Therefore in synchronous data transmission. The receiver must know the number of data units to be transferred.Also the receiver must be synchronized with data boundries. Usually one or two synchronized characters are used to indicate the start of each synchronous data stream. The data unit may contain a parity bit.

The synchronous receiver waits in a hunt mode while looking for data. As soon it matches one or two synchronous charactersbased on the number of synchronous characters used, the receiver starts interpreting the data. However if the data to be transmitted are not ready, the transmitter will pad with synchronous characters until data are available.

Once the receiver matches the synchronous characters, it receives the specified number of data units and then goes into a hunt mode for matching the synchronous pattern for the next data.

It is important to note that data may consists of 5,6,7 or 8 bits, but receiver treat it as 8 bit and if data is less than 8 bits then the rest of the bits are ignored.

ASYNCHRONOUS SERIAL DATA TRANSMISSION:

In this type of data transfer, the transmitting device does need not to be synchronized to the receiving device. The transmitting device can send one or more data units when it has dat ready to be sent. Each data unit must be formatted. In other words each data unit must contain start and stop bits, indicating the begining and end of each data unit. An interface chip is required between the microcomputer and serial I/O device. The interface chip performs the following functions:

1. Converts an 8 bit parallel data unit from the microcomputer into serial data for transmitting them to a serial I/O device.
2. Converts serial data from the serial I/O device into 8 bit parallel data for transmitting to the microcomputer.

ASYNCHRONOUS SERIAL DATA FORMAT:

Each asynchronous serial data unit can be divided into equal time intervals called bit intervals. A data bit can be either HIGH or LOW during each bit interval. An 8 bit data will have eight bit intervals. Each data bit will corrospond to one of the eight bit intervals.

The format for asynchronous serial data contains the following information.

1. A LOW start bit.
2. 5-8 data bits denoting the actual data being transfered.
3. An optional parity bit for either odd or even parity.
4. Stop bits denoting the end of the data. Stop bits may be 1,$1\frac{1}{2}$ or 2 having HIGH LEVELS. Note that $1\frac{1}{2}$ stop bits means HIGH level with a duration of 1.5 times the bit interval. Fig. 2.1 shows an example of asynchronous serial data with a LOW start bit, 8 bit data, 1 odd parity bit and one stop bit.

SERIAL DATA RATE:

The serial data rate is known as baud rate. The BAUD RATE is defined as number of bits of data transfered per second. Since each bit is transmitted over a duration of one bit interval ,

Hence

$$\text{Baud Rate} = \frac{1}{\text{Bit Interval}} = \text{Bits/sec.}$$

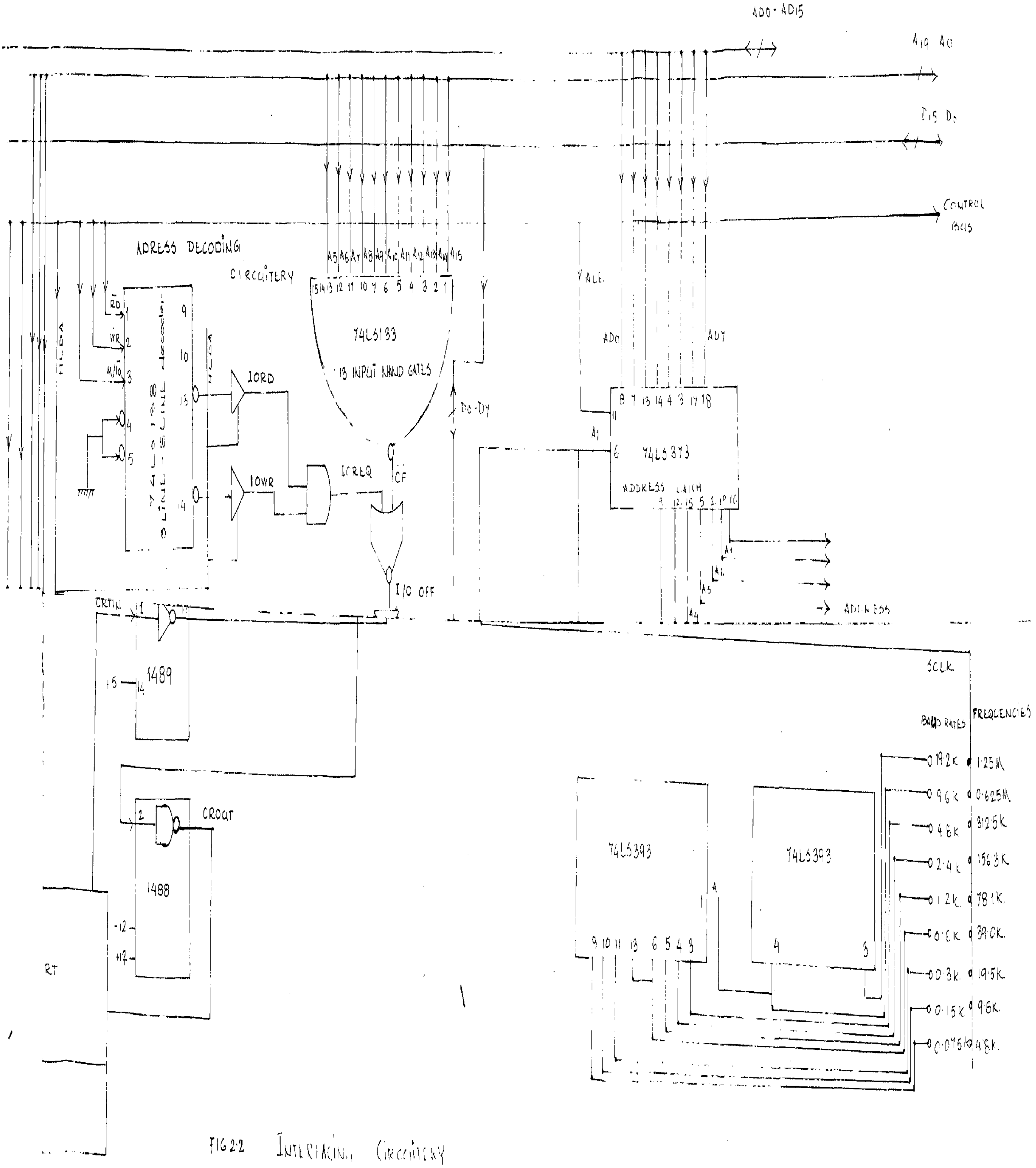


FIG 2.2 INTERFACING CIRCUITRY

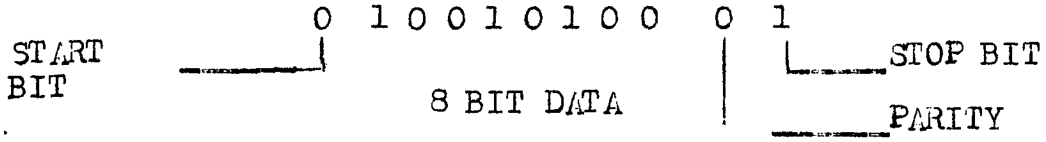


FIG. 2.1: ASYNCHRONOUS SERIAL DATA USING 8 BIT DATA, ODD PARITY AND 1 STOP BIT.

Microprocessor manufacturers typically provide the interfacing functions required by both synchronous and asynchronous serial transmission on a single chip called an USART. The intel 8251 is an example of an typical USART. If the chip contains only the asynchronous capability it is called as UART or an Asynchroncus Communication Interface Adapter (ACIA). The motorola 6850 is a typical example.

INTERFACING CKT DISCUSSION:

In this case however 8251 has been used for the purpose in Asynchronous mode. The interfacing diagram is given in Fig. 2.2.

Details of the different chips used are given in Appendixes as follows

USART 8251A..........	Appendix C
74LS393	Appendix D
1489	Appendix E
1488	Appendix F
8284	Appendix A

As shown in Fig. 2.2 two 74LS393 Dual 4 stage binary counters are used for generating the different frequencies from main CLK for different baud rates. The CLK frequency is 2.5 MHz. Further frequency division for different baud rates is very clear from Ckt. diagram/Appendix-D. The various frequencies for different values of baud rates are given in table 2 (a)

TABLE 2(a)

S.No.	BAUD RATES	FREQUENCIES
1.	19.2 K	1.25 M
2.	9.6 K	0.625 M
3.	4.8 K	312.5 K
4.	2.4 K	156.3 K
5.	1.2 K	78.1 K
6.	0.6 K	39.0 K
7.	0.3 K	19.5 K
8.	0.15 K	9.8K
9.	0.075	4.8K

So for selecting different baud rates the corresponding jumper as shown in Fig. 2.2 the point A is connected to SCLK of 8251 i.e. the receive and transmit Clock (RCLK and TRCLK). In this case both are same. So the 8251 changes the parallel data from 8086 to serial data for sending to CRT. But as the two systems namely microprocessor intel 8086 and CRT works at different logic levels the serial data thus obtained can not be directly send to CRT. CRT works at RS232C logic levels and intel 8086 at TTL logic levels. To convert the TTL logic levels to RS232C (i.e. +5V -0V to-12V to +12V) the 1488 is being used.

Similarly the data signal comming from the CRT (RS232C) is converted into TTL logic levels using 1489 IC; before transmitting it to 8251. After changing the logic levels the data signal goes to 8251, The USART changes the serial data into parallel and then transmit it to the microprocessor.

ADDRESS DECODING CIRCUITERY :

Referring to the I/O mapping, the addresses assigned to 8251 are the following

DATA WORD	FFF0 or FFF4
COMMAND WORD	FFF2 or FFF6

Writting these addresses in expanded form

1111 1111 1111 0000 or
1111 1111 1111 0100

1111 1111 1111 0010 or
1111 1111 1111 0110

So the chip to be selected $A_4, A_5, A_6 \dots A_{15}$ all should be 1, independent of whether data is transfered to/from DATA WORD or it is transfered to/from command word. Out of lower four bits A_3 and A_0 are zero always, A_2 is redundant to provide foldback addresses. A_1 decides that whether addresss is of DATA WORD or of COMMAND WORD hence A1 is directly connected to $C/\overline{D}$ input of 8251 and rest of the address bits are used to generate the chip select $\overline{CS}$ signal for 8251. Generation of $\overline{CS}$ signal for 8251 is very clearly shown in Fig. 2.2.

As shown in Fig. 2.2 A_{15}-A_5 are connected to a 13 input Nand gates 74LS133 so that when all the address bits A_{15}-A_5 are 1 then this NAND gate will give zero output, say $\overline{CF}$

I/O REQ signal is generated by decoding the three control lines namely $\overline{RD}$, $\overline{WR}$ and M/$\overline{IO}$. Whenever $\overline{RD}$ control signal and M/$\overline{IO}$ control signal are ACTIVE LOW and HLDA signal output from microprocessor is LOW. Then the $\overline{IORD}$ signal will will be ACTIVE LOW; whenever $\overline{WR}$ is ACTIVE LOW and IO/$\overline{M}$ is also LOW the $\overline{IOWR}$ will be LOW provided HLDA signal is LOW. Whenever either $\overline{IORD}$ is ACTIVE LOW OR $\overline{IOWR}$ is ACTIVE LOW, the $\overline{IOREQ}$ will be ACTIVE LOW. The $\overline{IOREQ}$ and $\overline{CF}$ are inputted to a NOR gate to get I/O OFF signal so when both $\overline{IOREQ}$ and $\overline{CF}$ are LOW, the I/OOFF signal will be HIGH and therefore $\overline{IOSELECT}$ signal will be LOW. The I/$\overline{O\ SELECT}$ signal along with address bits A_4, A_3, A_0 and control signal $\overline{BHE}$ is being used to generate the chip select signal for various I/O devices, are discussed in section 1.6 in previous chapter.

CHAPTER -3

DEVELOPMENT OF SOFTWARE MODULES

8086 has wide variety of instructions in its instruction set: Instruction set is given in Appendix. Just to elaborate the fact that how to use the instructions for a particular purpose and when to use which instruction, few software modules have been developed. Table 3.1 gives the brief description of the developed software modules.

TABLE 3.1

S.No.	FUNC.NAME	DESCRIPTION
1.	SOAP	SUM OF A.P.
2.	DNADD	DECIMAL NO. ADDITION
3.	BCDA	B.C.D. ADDITION
4.	SROOT	SQUARE ROOT
5.	SINE	TRIGANOMATRIC SINE
6.	BTGC	BINARY TO GRAY CODE CONVERSION
7.	OLBMV	OVERLAP BLOCK MOVE
8.	MULT	32 bit * 32 bit
9.	MATMUL	MATRIX MULTIPLICATION
10.	NOBIS	NUMBER OF BYTES IN A STRING
11.	DDIV	DECIMAL DIVISION
12	DMUL	DECIMAL MULTIPLICATION
13	ANBAS	ARRANGING NO IN ASCENDING ORDER
14	ANDS	ARRANGING NO IN DECENDING ORDER
15	GFIBC	GENERATE FIBONACCI NOS

S.No.	FUNC.NAME	DESCRIPTION
16.	MAXNIS	MAXIMUM NUMBER IN A STRING
17.	MINNIS	MINIMUM NUMBER IN A STRING
18.	FACTN	FACTORIAL OF A NUMBER
19.	HTDC	HEX TO DECIMAL CONVERSION

All the software modules have been written as an subroutines. The listing of the software modules along with proper documentations and brief description is contained in following pages.

FUNCTION NAME : SOAP

INPUT : n,d,a in same sequence in PARADR

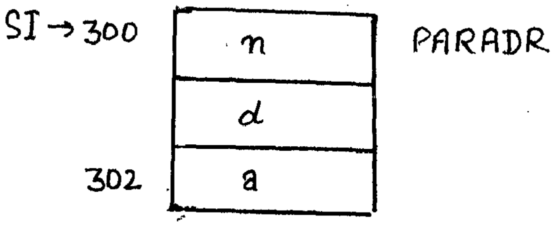
OUTPUT : SUM IN AX

CALLS : NONE

DESTROYS : AX, BX, SI,DX,CX

DESCRIPTION : $\text{SUM} = \frac{[2a + d\,(n-1)]n}{2}$

a,d,n should be in hex, series is ascending, Result is in Hex.



contd...

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	200	BE 00 03	MOV SI, PARADR	INITIALIZE THE PARAMETER POINTER
	203	8A 1C	MOV BL, [SI]	TAKE n
	205	46	INC SI	POINTER POINTING TOWARDS d
	206	FE CB	DEC BL	COMPUTE (n-1)
	208	8A 04	MOV AL, [SI]	
	20A	F6 E3	MUL AL,BL	COMPUTE d(n-1)
	20C	46	INC SI	POINTER POINTING TOWARDS a
	20D	50	PUSH AX	SAVE THE PRODUCT d(n-1)
	20E	B3 02	MOV BL,02	
	210	8A 04	MOV AL, [SI]	
	212	F6 E3	MUL Al,BL	COMUTE 2a
	214	5A	POP DX	RESTORE PRODUCT d(n-1)
	215	03 C2	ADD AX, DX	COMUTE 2a +d(n-1)
	217	4E	DEC SI	POINTER POINTING
	218	4E	DEC SI	TOWARDS n
	219	B5 00	MOV CH,00	
	21B	8A 0C	MOV CL,[SI]	
	21E	F7 E1	MOV AX,CX	
	220	BB 0200	MOV BX, 0002	
	223	F6 F3	DIV AX,BX	
	225	C3	RET	

FUNCTION NAME : DNADD (Decimal Number Addition)

INPUT : Decimal Numbers as Strings of ASCII number as shown in Fig. 3.2.

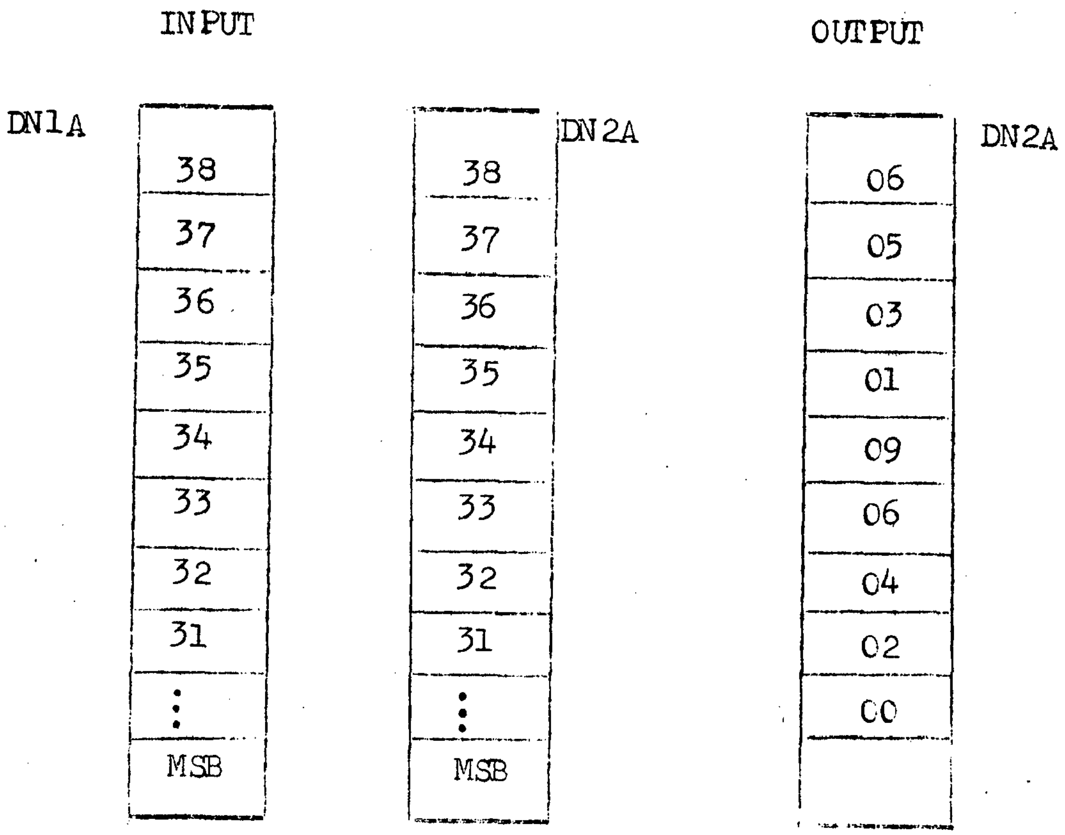
OUTPUT : SUM of two decimal numbers in location of the string pointed by DI as shown in Fig. 3.2

CALLS : None

DESTROYS : SI, DI, AX, CX, FL.

DESCRIPTION : Decimal Addition is achieved by ACSII Addition and then making proper adjustment.

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS.	COMMENTS
	200	BE 00 03	MOV SI, DN1A	
	203	BF 08 03	MOV DI, DN2A	
	206	B9 08 00	MOV CX, LOSONS	
LOOP	209	8A 04	MOV AL,[SI]	
	20B	46	INC SI	
	20C	12 05	ADC AL, [DI]	
	20E	37	AAA	
	20F	88 05	MOV [DI],AL	
	211	47	INC DI	
	212	49	DEC CX	
	213	75 F4	JNZ LOOP	
	215	73 07	JNC LAST	
		B0 00	MOV AL,00	
	217	12 05	ADC AL,[DI]	
	219	88 05	MOV [DI],AL	
	21B	C3	RET	



in ASCCI Strings

```
eg.      12345678
       + 12345678
      ------------
         24691356
      ------------
```

FIG. 3.2

FUNCTION NAME : BCD ADD

INPUT : [SI] → LSB of ST1, [DI] → LSB of ST2
[CX] = Length of String.

OUTPUT : SUM As BCD no. in memory locations of ST2.

CALLS : NONE

DESTROYS : FL, SI, DI, CX, AX

DESCRIPTION : Limitation is that SUM of any two corresponding numbers< 99.

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	200	F8	CLC	
	201	B9 08 00	MOV CX, SL	
	204	BE 00 03	MOV SI, ST1	
	207	BF 08 03	MOV DI, ST2	
LOOP	20A	8A 04	MOV AL, [SI]	
	20C	12 05	ADC AL, [DI]	
	20E	27	DAA	
	20F	88 05	MOV [DI], AL	
	211	46	INC SI	
	212	47	INC DI	
	213	49	DEC CX	
	214	75 F4	JNZ LOOP	
	216	C3	RET	

eg.

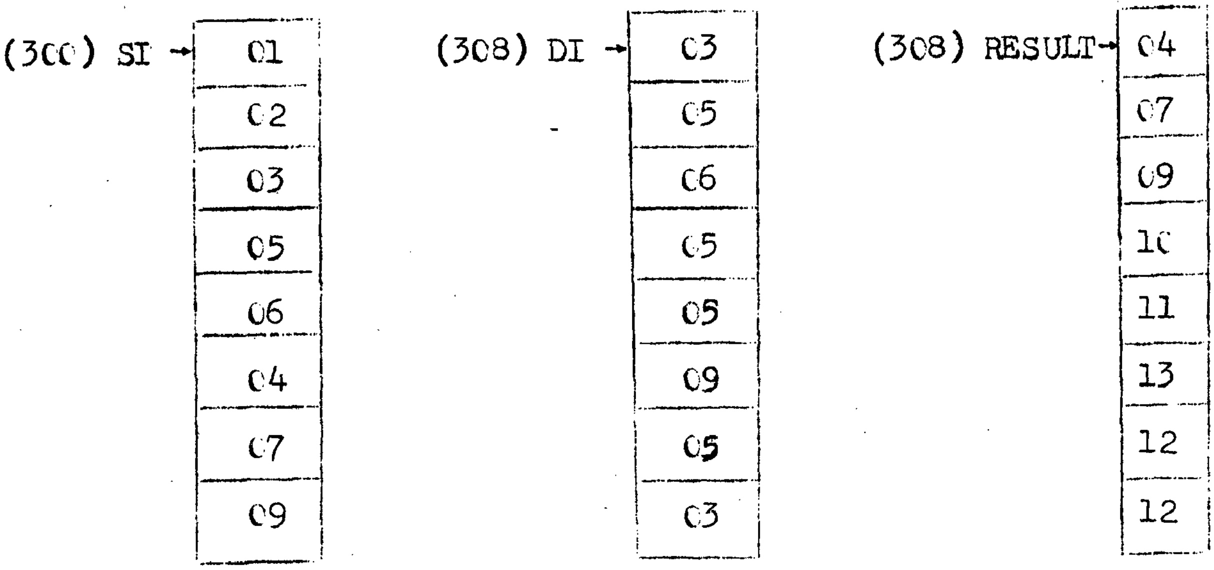


FIG. 3.3

FUNCTION NAME : BTGC

INPUT : THE BINARY NUMBER IN LOCATION POINTED BY SI
LOOK TABLE STARTING FROM LOCATION POINTED BYBX
AS SHOWN IN FIG. 3.4

OUTPUT : GRAY CODE CORRESPONDING TO BINARY NO. IN
LOCATION [[SI]+1].

CALLS : None

DESTROYS : BX, SI,AX

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	200	BB 00 03	MOV BX,300	
	203	BE 50 02	MOV SI,250	
	206	AC	LODSB	
	207	D7	XLAT	
	208	88 04	MOV [SI],AL	
	20A	C3	RET	

LOOK UP TABLE

ADDRESS	CONTENTS
300	00
301	01
302	02
303	03
304	04
305	05
306	06
307	07
308	0F
309	0E
30A	0D
30B	0C
30C	0B
30D	0A
30E	09
30F	08

FIG 3·4

FUNCTION NAME : OVLAPBMV

INPUT : NO OF WORDS IN CUNTR CX

END ADDR OF STRING TO BE MOVED AS POINTER1

END ADDR OF STRING AFTER MOVING AS POINTER2

LENGTH OF THE STRING TO BE MOVED AS COUNTER WORD

OUTPUT: STRING WILL BE MOVED TO DESIRED ADDR.

CALLS : NONE

DESTROYS: SI,DI,FR,CX

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	200	FD	STD	
	201	BE 1F 03	MOV SI,POINTER1	
	204	BF 30 03	MOV DI,POINTER2	
	207	B9 20 00	MOVCX, CUNTR ~~&~~ WORD	
	20A	F2 A5	RENZ:MOVSW	
	20C	C3	RET	

FUNCTION NAME : MUL

INPUT : TWO 32 BIT OPERANDS IN SEQUENTIAL MEMORY LOCATION POINTED BY Bx

OUTPUT : RESULT IN SEQUENTIAL MEMORY LOCATION STARTING FROM BX+8.

CALLS : NONE

DESTROYS : SI,BX,AX

DESCRIPTION:

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	2FD	BE 00 00	MOV SI,0000	INITIALIZE SI FOR SEGMENT
	300	BB 00 03	MOV BX,0300	
	303	8B 00	MOV AX,[BX]	MULTIPLY LOW-ORDER 16 BITS
	305	F7 60 04	MUL [BX+4]	BY LOW-ORDER 16 BITS
	308	89 40 08	MOV [BX+8],AX	SAVE RESULT,WHICH IS IN AX
	30B	89 50 0A	MCV [BX+10],DX	AND DX
	30E	8B 00	MCV AX [BX]	MULTIPLY LOW ORDER 16 BITS OF
	310	F7 60 06	MUL [BX + 6]	OPERAND B
	313	C1 40 CA	ADD [BX+1C],AX	ADD TC PREVIOUS RESULT
	316	11 50 CC	ADC [BX+12],DX	ASSUME RESULT BYTES
	319	77 03	JNC NEXTØ MUL	ARE INITIALLY ZERO

contd...

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
NEXTMUL	31B	FE 40 0E	INC [BX+14]	
	31E	8B 40 02	MOV AX,[BX+ 2]	MULTIPLY HIGH ORDER 16 BITS
	321	F7 60 04	MUL [BX+4]	OF OPERAND A BY HIGH ORDER
	324	01 40 0A	ADD [BX+10],AX	16 BITS OF OPERAND B
	327	11 50 0C	ADC [BX+12],DX	
	32A	77 02	JNC HI$ORDER$MUL	
HIOMUL	32C	FE 40 0E	INC [BX+14]	SAVE CARRY
	32F	8B 40 02	MOV AX, [BX+2]	MULTIPLY HIGH-ORDER 16 BITS OF OPERAND
	332	F7 60 06	MUL [BX+6]	ABY HIGH ORDER 16BITS OF OPERAND B
	335	01 40 0C	ADD [BX+12],AX	ADD TO PREVIOUS RESULT
	338	11 50 0E	ADC [BX+14],DX	ADD TO PREVIOUS RESULT
	33B	C3	RET	

FUNCTICN NAME : MATMUL

INPUT : ADDRESS CCRROSPCNDING TC PCINTERS MATA, MATB and RES PCINTER.

MAT ELEMENTS IN PRCPER AREA, shown in Fig. 3.5

CUTPUT : PRCDUCT MATRIX STARTING FRCM ADDRESS CF RESULT PCINTER.

CALLS : i. RESULT $ MEMCRY $ AREA $ INITIALIZATICN

ii. SUB $ ADD

iii. ADDITICN

iv. ASCII $ MUL

v. ASCII $ CCNV

vi. ELEMENT $ CNE

DESTRCYS : AX, SI,DI,CX,DX,BP,BX

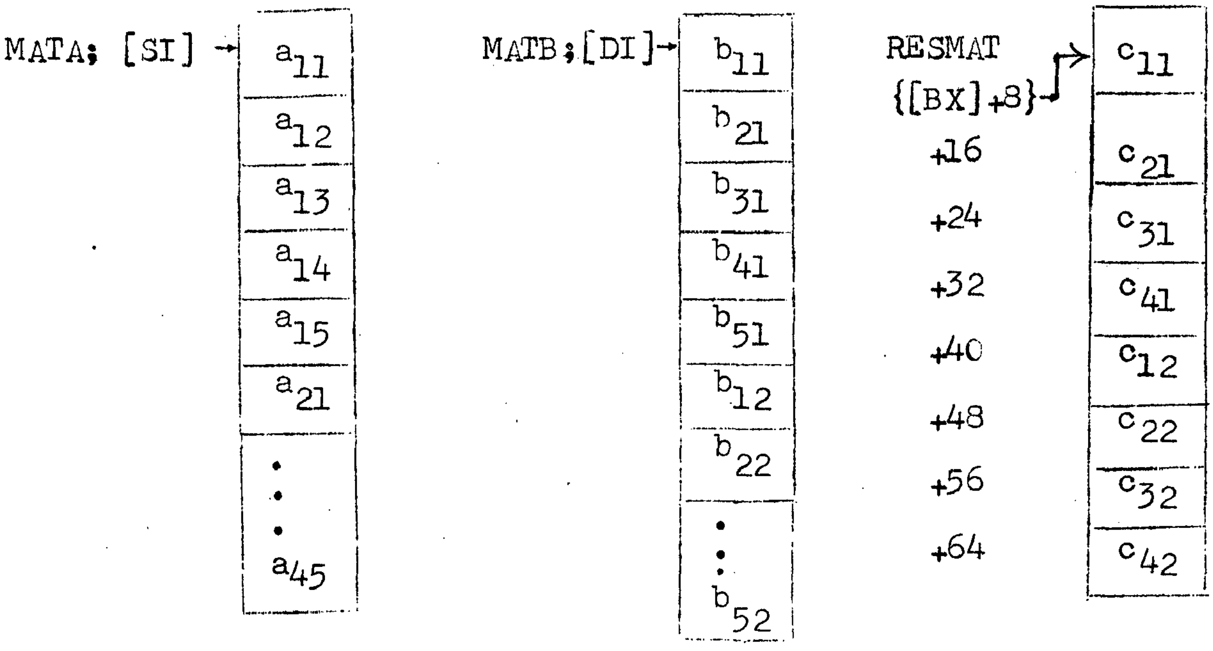
DESCRIPTICN : This Subroutine is for the multiplication of the following type.

[C] = [A] * [B]

Where A is 4 x 5 matrix

B is 5 x 2 matrix

with little changes this subroutine can be modified for any order of matrices multiplication.



LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	4F0	9A 00 02 00 00	CALL RESULT$ AREA $ INITIAZATION	TO INITIALIZE THE RESULT AREA WITH 00
	4F5	BD 00 00	MOV BP,0000	INITIALIZE THE STACK
	4F8	B8 00 00	MOV AX,00 00	SEGMENT AND INDEX
	4FB	8E D0	MOV SS,AX	AS 00
	4FD	BB 20 04	MOV BX,RES$ POINTER	INITIALIZE THE RESULT$ POINTER
	500	BE 00 03	MOV SI,MATA$ POINTER	INITIALIZE THE MATRIX. A POINTER
	503	BF 00 04	MOVDI,MATB$ POINTER	INITIALIZE THE MATRIX B POINTER
	506	B9 04 00	MOV CX,04	INITIALIZE NO.OF COLUMNS IN A
LOOP1	509	9A 90 06 00 00	CALL ELEMENT$ ONE	TO CALCULATE COMPONENTS OF C_{11}
	50E	BF 00 04	MOV DI,MAT A$ POINTER	TO CALCULATE COMPONENTS
	511	49	DEC CX	OF C_{21}, C_{31}, C_{41}
	512	75 F5	JNZ LOOP1	
	514	BF 05 04	MOV DI,405	
	517	B9 04 00	MOV CX,04	
	51A	BE 00 03	MOV SI,300	
LOOP2	51D	9A 90 06 00 00	CALL ELEMENT$ ONE	TO CALCULATE COMPONENTS OF C_{12}
	522	BF 05 04	MOV DI,405	TO CALCULATE THE
	525	49	DEC CX	COMPONENTS OF C_{22}, C_{32} and C_{42}
	526	75 F5	JNZ LOOP2	
	528	9A 10 04 00 00	CALL ASC11$CONV	TO CONVERT THE COMPONENTS INTO ASC11CODES
	52D	BF 6E 04	MOV DI,46E	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	530	BE 6C 04	MOV SI,46C	TO CALCULATE
	533	BA 04 00	MOV DX,04	C_{42}
	536	9A B0 06 00 00	CALL SUB $ADDITION	
	53B	80 0D 30	OR [DI],30	
	53E	57	PUSH DI	
	53F	47	INCR DI	
	540	80 0D 30	OR [DI],30	
	543	5F	POP DI	
	544	4A	DEC DX	
	545	75 EF	JNZ LOOP 1	
	547	BF 66 04	MOV DI, 466	
	54A	C6 05 00	MOV [DI]$_8$,00	
	54D	BF 64 04	MOV DI,464	
	550	BE 62 04	MOV SI,462	
	553	BA 04 00	MOV DX,04	
	556	9A B0 06 00 00	CALL SUB $ADDITION	TO CALCULATE
	55B	80 0D 30	OR [DI],30	C_{32}
	55E	57	PUSH DI	
	55F	47	INCR DI	
	560	80 0D 30	OR [DI] 30	
	563	5F	POP DI	
	564	4A	DEC DX	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	565	75 EF	JNZ LOOP2	
	567	BF 5C 04	MOV DI,45C	
	56A	C6 05 00	MOV [DI],00	
	56D	BF 5A 04	MOV DI,45A	
	570	BE 58 04	MOV SI,458	
	573	BA 04 00	MOV DX,04	
LOOP3	576	9A B0 06 00 00	CALL SUB$ADDITION	
	57B	80 0D 30	OR [DI],30	
	57E	57	PUSH DI	TO CALCULATE
	57F	47	INCR DI	C_{22}
	580	80 0D 30	OR [DI],30	
	583	5F	POP DI	
	584	4A	DEC DX	
	585	75 EF	JNZ LOOP 3	
	587	BF 52 04	MOV DI,452	
	58A	C6 05 00	MOV [DI]$_8$,00	
	58D	BF 50 04	MOV DI,450	TO CALCULATE C_{12}
	590	BE 4E 04	MOV SI, 44E	
	593	BA 04 00	MOV DX,04	
LOOP4	596	9A B0 06 00 00	CALL SUB$ ADDITION	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	59B	80 0D 30	OR [DI],30	
	59E	57	PUSH DI	
	59F	47	INCR DI	
	5A0	80 0D 30	OR [DI],30	
	5A3	5F	POP DI	
	5A4	4A	DEC CX	
	5A5	75 EF	JNZ LOOP4	
	5A7	BF 48 04	MOV DI,448	
	5AA	C6 05 00	MOV [DI],00	
	5AD	BF 46 04	MOV DI,446	
	5B0	BE 44 04	MOV SI,444	
	5B3	BA 04 00	MOV DX,0.4	
LOOP5	5B6	9A B0 06 00 00	CALL SUB$ ADDITION	TO CALCULATE C_{41}
	5BB	80 0D 30	OR [DI],30	
	5BE	57	PUSH DI	
	5BF	47	INCR DI	
	5C0	80 0D 30	OR [DI],30	
	5C3	5F	POP DI	
	5C4	4A	DEC DX	
	5C5	75 EF	JNZ LOOP5	
	5C7	BE 3E 04	MOV DI,43E	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	5CA	C6 05 00	MOV [DI],00	
	5CD	BF 3C 04	MOV DI,43C	
	5D0	BE 3A 04	MOV SI, 43A	
	5D3	BA 04 00	MOV DX,04	TO CALCULATE C_{31}
LOOP6	5D6	9A B0 06 00 00	CALL SUB ADDITION	
	5DB	80 0D 30	OR [DI],30	
	5DE	57	PUSH DI	
	5DF	47	INCR DI	
	5E0	80 0D 30	OR [DI],30	
	5E3	5F	POP DI	
	5E4	4A	DEC CX	
	5E5	75 EF	JNZ LOOP 6	
	5E7	BF 34 04	MOV DI,434	
	5EA	C6 05 00	MOV DI,00	
	5ED	BF 32 04	MOV DI,432	
	5F0	BE 30 04	MOV SI,430	
	5F3	BA 04 00	MOV DX,04	TO CALCULATE C_{21}
LOOP7	5F6	9A B0 06 00 00	CALL SUB$ADDITION	
	5FB	80 0D 30	OR [DI],30	
	5F3	57	PUSH DI	
	5FF	47	INCR DI	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	600	80 0D 30	OR [DI],30	
	603	5F	POP DI	
	604	4A	DEC CX	
	605	75 EF	JNZ LOOP 7	
	607	BF 2A 04	MOV DI,42A	
	60A	C6 05 00	MOV [DI],00	
	60D	BF 28 04	MOV DI,428	
	610	BE 26 04	MOV SI, 426	
	613	BA 04 00	MOV DX,04	
LOOP8	616	9A B0 06 00 00	CALL SUB$ ADDITION	
	61B	80 0D 30	OR [DI],30	
	61E	57	PUSH DI	
	61F	47	INCR DI	TO CALCULATE C_{11}
	620	80 0D 30	OR [DI],30	
	623	5F	POP DI	
	624	4A	DEC DX	
	625	75 EF	JNZ LOOP 8	
	627	C3	RET	

FUNCTION NAME : RESULT $ MEMORY $ AREA $ INITIALIZATION

INPUT : None

OUTPUT : Result memory are is initialized to 00

CALLS : None

DESTROYS: CX,SI,DI,FR

DESCRIPTION :

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	200	B9 28 00	MOV, CX,28	
	203	BF 20 04	MOV DI, 420	
LOOP	206	C7 05 00 00	MOV DI, 00	
	20A	47	INC DI	
	20B	47	INC DI	
	20C	49	DEC CX	
	20D	75 F7	JNZ LOOP	
	20F	C3	RET	

FUNCTION NAME : ASCII $ MUL

INPUT : INITIALIZE SI AND DI,POINTER, AND BX

OUTPUT : PRODUCT OF THE ASCII NUMBERS CONTAINED IN [SI] AND [DI] IN [BX]

CALLS : NONE

DESTROYS : SI, DI, BX, DL, AL

DESCRIPTION

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	350	8A 04	MOV AL, [SI]	
	352	8A 15	MOV DL, [DI]	
	354	F6 E2	MUL AL,DL	
	356	D4 0A	AAM	
	358	02 07	ADD AL, [BX]	
	35A	37	AAA	
	35B	46	INCSI	
	35C	47	INCDI	
	35D	C3	RET	

FUNCTION NAME : ADDITION

INPUT : SI,DI,POINTER TO POINT TWO ELEMENTS COMPONENT.

OUTPUT : ADDS TWO COMPONENTS AND SUM IS IN SEQUENTIAL MEMORY LOCATION POINTED BY DI

CALLS : NONE

DESTROYS : AL, CX, SI, DI

DESCRIPTION :

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	6D0	B9 02 00	MOV CX, 02	
LOOP	6D3	8A 04	MOV AL, [SI]	
	6D5	46	INC SI	
	6D6	12 05	ADC AL, [DI]	
	6D8	37	AAA	
	6D9	88 05	MOV [DI],AL	
	6DB	47	INC DI	
	6DC	49	DEC CX	
	6DD	75 F4	JNZ LOOP	
	6DF	73 06	JNC LAST	
	6E1	B0 00	MOV AL,00	
	6E3	12 05	ADC AL,[DI]	
	6E5	88 05	MOV [DI],AL	
LAST	6E7	C3	RET	

FUNCTION NAME : SUB $ ADDITION

INPUT : ADDRESS TO THE POINTER SI AND DI

OUTPUT : ADDS THE TWO COMPONENTS OF MATRIX ELEMENT C

CALLS : ADDITION

DESTROYS : BP,DI,SI,CX,AX

DESCRIPTION :

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	6B0	89 FD	MOV BP,DI	SAVE ELADDRESS
	6B2	9A D0 06 00 00	CALL ADDITION	ADD TWO COMPONENTS
	6B7	4E	DEC SI	
	6B8	4E	DEC SI	TO ACCESS NEXT
				COMPONENTS
	6B9	4E	DEC SI	
	6BA	4E	DEC SI	
	6BB	89 EF	MOV DI,BP	RESTORE THE
	6BD	C3	RET	ELADDRESS

FUNCTION NAME	: ASCII $ CONV
INPUT	: NONE
OUTPUT	: ASCII CODES (CORROSPONDING NUMBER) IN POINTER DI
CALLS	: NONE
DESTROYS	: DI, CX,
DESCRIPTION	:

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	410	BF 20 04	MOV DI, 420	
	413	B9 50 00	MOV CX, 50	
	416	80 0D 30	OR [DI],30	
	419	47	INC DI	
	41A	49	DEC CX	
	41B	75 F9	JNZ LOOP	
	41D	C3	RET	

FUNCTION NAME : NIDS

INPUT : CUNT LB } NUMBER OF BYTES TO BE STORED
CUNT+1 HB

CUNT+2 ST ADDR OF FIRST BYTE

OUTPUT : REARRANGED NUMBERS IN ASCENDING ORDER STARTING FROM CUNT+3

CALLS : NONE

DESTROYS : SI, BX,CX,AL,FR

LABEL	ADDRESS	CONTENTS	MNEMONIC AND OPERANDS	COMMENTS
	200	BE 00 03	MOV SI, CUNT	
	203	8A 1C	MOV BX [SI]	
	205	4B	DEC BX	
LOOP1	206	8A 0C	MOV CX,[SI]	
	208	49	DEC CX	
	209	BE 02 03	MOV SI,CUNT+2	
LOOP2	20C	8A 04	MOV AL, [SI]	
	20E	46	INC SI	
	20F	3A 04	CMP AL, [SI]	
	211	73 06	JAE LOOP 3	
	213	86 04	XCHG AL, [SI]	
	215	4E	DEC SI	
	216	88 04	MOV [SI],AL	
	218	46	INC SI	
LOOP3	219	E0 F1	LOOP NZ LOOP 2	

FUNCTION NAME : ELEMENT $ ONE

INPUT : SI, DI,POINTERS SHOULD POINT TOWARDS TWO CORROSPONDING ELEMENTS OF MATRICES.

OUTPUT : COMPONENTS OF ONE ELEMENT OF PRODUCT MATRIX IN BX POINTER.

CALLS : ASCII $ MUL

DESTROYS : BX, SI, DI, AX, DL

DESCRIPTION:

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	690	51	PUSH CX	SAVE PREVIOUS COUNTR
	691	B9 05 00	MOV CX,05	INITIALIZE NEW COUNTR
	694	9 50 03 00 00	CALL,ASCII$MUL	MULTIPLY 8 bit *8bit
	699	88 07	MOV [BX],AL	SAVE LOWER ORDR 8BIT OF PRODUCT
	69B	43	INC BX	
	69C	88 27	MOV [BX],AH	SAVE HIGHER ORDER8BIT OF PRODUCT
	69E	43	INC BX	
	69F	49	DEC CX	IS ALL ELENTS HAVE BEEN MULTIPLIED
	6A0	75 F2	JNZ LOOP	NO MULTIPLY NEXT
	6A2	59	POP CX	YES RESTORE COUNTR
	6A3	C3	RET	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	21B	4B	DEC BX	
	21C	BE 00 03	MOV SI, CUNT	
	21F	75 E5	JNZ LOOP 1	
	221	C3	RET	

FUNCTION NAME : GNFIBN

INPUT : TWO STARTING NO. OF SERIES IN NUMB1 and NUMB1+1
NUM B2: LENGTH OF SERIES IN COUNTR (CX)

OUTPUT : SERIES OF FIBONACCI NUMBERS STARTING FROM ADDR NUMB2.

CALLS : NONE

DESTROYS : AX,BX,CX,FR

DESCRIPTION

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	200	B8 00 00	MOV AX,00 00	
	203	BB 00 03	MOV BX,NUMB1	
	206	B9 10 CO	MOV CX,COUNTR	
	209	89 07	MOV [BX],AX	
	20B	40	INC AX	
	20C	89 47 02	MOV [BX+2] AX	
	20F	8B 07	MOV AX,[BX]	
LOOP	211	01 47 02	ADD,AX,[BX+2]	
	214	89 47 04	MOV [BX+4],AX	
	217	43	INC BX	
	218	43	INC BX	
	219	EO F4	LOOPNZ Loop	
	21B	C3	RET	

FUNCTION NAME : MAXNIS

INPUT : NO OF BYTES IN COUNTR, STRING STARTING FROM STADDR.

OUTPUT : Max. No. in AH

CALLS : None

DESTROYS : SI, CX, AX, FR

DESCRIPTION :

LABEL	ADDR	CONTENTS	MNEMONIC AND OPERANDS	COMMENTS
	200	BE 00 03	MOV SI,STADDR	Initialize the pointer
	203	B9 10 00	MOV CX,COUNTR	Initialize the COUNTR
	206	B4 00	MOV AH,00	
STRT	208	3A 24	CMP AH, [SI]	Byte > 0
	20A	73 02	JAE NEXT	No: check next Byte
	20C	8A 24	MOV AH, [SI]	Yes exchange Result with String Byte
NEXT	20E	46	INC SI	Increment COUNTR
	20F	E0 F7	LOOP NZ STRT	To Check next byte
	211	C3	RET	END;Results is in AH

FUNCTION NAME : MINNIS

INPUT : No. of Bytes in COUNTR; String Starting from STADDR

OUTPUT : Minimum No. in AH

CALLS : None

DESTROYS : SI, CX, AX, FR

DESCRIPTION

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	200	BE 00 03	MOV SI, STADDR	
	203	B9 10 00	MOV CX, COUNTR	
	206	B4 FF	MOV AH,FF	
STRT	208	38 24	CMP [SI], AH	
	20A	73 02	JAE NEXT	
	20C	8A 24	MOV AH, [SI]	
NEXT	20E	46	INC SI	
	20F	E0 F7	LOOP NZ STRT	
	211	C3	RET	

FUNCTION NAME	: FACTN
INPUT	: PARAMETERS n, in AL and n-1 in CX
OUTPUT	: Result n IN AX
CALLS	: NONE
DESTROYS	: AX, CX,BL,FL
DESCRIPTION	: NUMB IS one byte.

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	200	B9 03 00	MOV CX,N-1	
	203	BO 04	MOV AL,N	
	205	88 C3	MOV BL,AL	
L1	207	FE CB	DEC BL	
	209	F6 E3	MUL AL,BL	
	20B	49	DEC CX	
	20C	75 F9	JNZ L1	
	20E	C3	RET	

FUNCTION NAME : HTDC

INPUT : As shown in Fig. 3.6.

OUTPUT : RESULT STARTING FROM FIN$ RES$ ADDR,

CALLS : ASCII$STRING$ MUL, ASCII$DIVISITION, ASCII$ ADDITION

DESTROYS : DECIMAL NO. IN SEQUENTIAL MEMORY LOCATION starting from 500.

DESCRIPTION

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	200	B9 03 00	MOV CX,03	INITIALIZE COUNTR
	203	BB 00 03	MOV BX,HEX SADDR	
	206	43	INC BC	LEAVE LOWEST DIGIT
	207	8A 17	MOV DL, [BX]	TAKE NEXT WT 16^1DIGIT
	209	BE 04 03	MOV SI,WTADDR	INITIALIZE WEIGHT TABLE POINTER.
	20C	BF 00 05	MOV DI,RESADDR	INITIALIZE RESULT POINTER.
LOOP1	20F	9A 00 04 00 00	CALL ASCII$ STRING $ MUL	CALCULATE DECIMAL EQUIVALENT WEIGHT
	214	47	INC DI	
	215	43	INC BX	CONSIDER NEXT HIGHER DIGIT.
	216	8A 17	MOV DL,[BX]	
	218	49	DEC CX	ARE WEIGHTAGE OF ALL DIGITS HAVE BEEN CALCULATED
LOOP3	219	80 CD 30	OR [DI],30	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	21B	BF 00 05	MOV DI,RES $ ADDR	YES PROCEED TO ADD WEIGHTAGES
	21E	B9 0F 00	MOV CX,0F	
AGAIN	221	80 0D 30	OR [DI],30	CONVERT TH ELEMENTS IN PRODUCT TO TEIR CORROS PONDING ASCII CODE
	224	47	INC DI	
	225	49	DEC CX	
	226	75 F9	JNZ AGAIN	
	228	BE 05 05	MOV SI,L2	
	22B	BF 0A 05	MOV DI,L3	
	22E	9A 30 04 00 00	CALL ASCII$ ADDITION	ADD WEIGHTAGES OF TWO DIGITS
	233	BF 0A 05	MOV DI,L3	
	236	B9 06 00	MOV CX,06	
LOOP2	239	80 0D 30	OR [DI],30	
	23C	47	INC DI	
	23D	49	DEC CX	
	23E	75 F9	JNZ LOOP2	
	240	BE 00 05	MOV SI,L1	
	243	BF 0A 05	MOV DI,L3	
	246	9A 30 04 00 00	CALL ASCII$ ADDITION	ADD THE THIRD WEIGHTAGE
	24B	BF 0A 05	MOV DI,FIN$ RES$ADDR	INITIALIZE FINAL RESULT POINTER
	24E	B9 06 00	MOV CX,06	
LOOP3	251	80 0D 30	OR [DI],30	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	254	47	INC DI	
	255	49	DEC CX	
	256	75 F9	JNZ LOOP3	
	258	BB 00 03	MOV BX,300	
	25B	BE 00 05	MOV SI,500	
	25E	8A 07	MOV AX,[BX]	
	260	88 04	MOV [SI],AX	
	262	46	INC SI	
	263	B9 04 00	MOV CX,04	
	266	C6 04 30	MOV [SI],30	
	269	46	INC SI	
	26A	49	DEC CX	
	26B	75 F 9	JNZ LOOP 3	
	26D	BE 00 05	MOV SI,500	
	270	BF 0A 05	MOV DI,50A	
	273	9A 30 04 00 00	CALL ASCII $ ADDITION	ADD WEIGHTAGE OF 0th DIGIT
	278	C3	RET	

FUNCTION NAME : ASCII$ ADDITION

INPUT : TWO STRINGS TO BE ADDED POINTED BY SI and DI

OUTPUT : SUM OF TWO STRINGS IN LOCATIONS (SEQUENTIAL) POINTED BY DI

CALLS : None

DESTROYS : CX,SI,DI,AX

DESCRIPTION :

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	430	B9 05 00	MOV CX,05	
LOOP	433	8A 04	MOV AL, [SI]	
	435	46	INC SI	
	436	12 05	ADC AL,[DI]	
	438	37	AAA	
	439	88 05	MOV [DI],AL	
	43B	47	INC DI	
	43C	49	DEC CX	
	43D	75 F4	JNZ LOOP	
	43F	73 06	JNC LAST	
	441	B0 00	MOV AL,00	
	443	12 05	ADC AL,[DI]	
	445	88 05	MOV [DI],AL	
LAST	447	C3	RET	

FUNCTION NAME :ASCII $ STRING $ MUL

INPUT :SI,DI POINTING TOWARDS MULTIPLICAND AND MULTIPLIER

OUTPUT :PRODUCT IN DI LOCATIONS

CALLS :NONE

DESTROYS : DI,AX,DL,SI

DESCRIPTION

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	400	51	PUSH CX	
	401	B9 04 00	MOV CX,04	
	404	C6 05 00	MOV [DI],00	
	407	80 E2 OF	AND,DL,OF	
LOOP	40A	8A 04	MOV Al,[SI]	
	40C	46	INC SI	
	40D	80 E0 OF	AND AL,OF	
	410	F6 E2	MUL DL	
	412	D4 OA	AAM	
	414	02 05	ADD Al,[DI]	
	416	37	AAA	
	417	88 05	MOV [DI],AL	
	419	47	INC DI	
	41A	88 25	MOV [DI],AH	
	41C	49	DECCX	
	41D	75 EB	JNZ LOOP	
	41F	59	POP CX	
	420	C3	RET	

FUNCTION NAME : SROOT

INPUT : NUMBER (OPERAND) IN MEMORY NUMI

OUTPUT : SQUARE ROOT OF NO IN NUMI IN MEMORY NUMI+2

CALLS : NOTHING

DESTROYS : AX,BX,DX,DS,SI

DESCRIPTION : FIRST APPROXIMATION = $2+\frac{N}{200} = A_1$(say)

SECOND APPROXIMATION= $[\frac{N}{A_1}+A_1]\frac{1}{2} = A_2$(say)

THIRD APPROXIMATION = $[\frac{N}{A_2}+A_2]\frac{1}{2} = A_3$(say)

AND SO ON TILL $A_n \cdot A_n = N$

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
SROOT	200	C7C2-00 00	MOV DX,0000	DX=0 FOR DIVISION INSTRUCTION
	204	BB 04 03	MOV BX 0304	BASE ADDRESS OF NUMBERS
	207	B8 1000	MOV AX,0010	VALUE FOR THE DS REGISTERS
	20A	8E D8	MOV DS,AX	MOVE TO DS FROM AX
	20C	8B 07	MOV AX,[BX]	GET THE ORIGINAL NUMBER
	20E	F7 77 FE	DIV AX,[BX-02]	DIVIDE BY 200 (DECIMAL)
	211	05 02 00	ADD AX,0002H	ADD2 TO THE RESULT
LOOP	214	89 47 02	MOV[BX+02],AX	SAVE APPROXIMATION IN MEMORY
	217	F7 67 02	MUL AX,[BX+02]	MULT APPROXIMATION BY ITSELF.

contd....

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	21A	3B 07	CMP A1,[BX]	RESULT=ORIGINAL NUMBER
	21C	74 10	JZ DONE	YES THEN 8086 IS DONE
	21E	8B 07	MOV AX,[BX]	NO,GET ORIGINAL NUMBER.
	220	C7 C2 0000	MOV DX,0000	SET DX=0 FOR DIVISION
	224	F7 77 02	DIV AX,[BX+02]	DIVIDE ORIG.BY APPROXIMATION.
	227	03 47 02	ADD AX,[BX+02]	ADD APPROXIMATION TO RESULT.
	22A	D1 E8	SHR	DIVIDE RESULT BY TWO
	22C	EB E6	JMP LOOP	JMP TO SAVE NEW APPROXIMATION
DONE	22E	BE	MOV SI,MASDATB	
	231	9A FD 02 00 00	CALL MASDISP	

0302	C8 00		; Decimal 200 or Hex C8
0304	00 00	NUMI	; Number whose Root is to be determined
0306	00 00		; Square Root of number is stored here.

CHAPTER - 4

EXPERIMENTATION WITH HARDWARE MODULES

In order for the 8086 to communicate with peripheral devices, the peripheral must be wired to the microcomputer's data and address buses, as well as being wired to some of the microprocessor's control signals. Once this is done, the 8086 can communicate with the I/O device using either accumulator I/O or memory mapped I/O. The hardware interfaces for the two techniques are very similar.

In accumulator I/O the 8086 can communicate with up to 64 K eight bit devices. In order to actually transfer data between one of these devices and the 8086, the CPU has to execute either an IN or OUT instruction. There are two different types of these instructions. One type has the address of the peripheral device fixed in the instruction, much like an immediate data byte. This type of instruction can only be used to address the first 256 accumulator I/O devices starting at zero. This is the same type of I/O instruction, but different op-code that the 8080 and 8085 have. The other method uses the content of the DX register as the device address, thus gaining the ability to address 64K devices. Regardless of the type of IN or OUT instruction used, the AX register of the CPU will either transmit (output) data to the peripheral or receive (input) data from the peripheral.

In an accumulator I/O peripheral interface, the 16 bit device address would be present on AD0 through AD15. These address signals, along with $\overline{RD}$, $\overline{WR}$ and M/$\overline{IO}$ would then be gated togather and the resulting logic 1 or logic 0 pulse would be used to actually gate information off or to gate information onto the data bus.

The AX register can become the bottelneck in I/O intensive situation, simply because the data would have to be moved from memory to it, to be output, or it would have to be moved from AX to memory, after being imput. One solution to this problem is to use memory mapped I/O. In this type of I/O the microprocessor assumes that it is communicating with the memory location, when in fact it is communicating with peripheral device. The complete 20 bit address bits and 16 bit data bus are to be used, for interface design. The only advantage in using memory mapped I/O is the fact that any 8086 instruction that interfaces a memory location can now be used to communicate with a properly designed memory mapped I/O peripheral device thus we no longer have bottelneck in the AX register. Two disadvantages of memory mapped I/O are the facts that it uses up some of the address space normally reserved for memory, and it is slower than accumulator I/O simply because of the time required to calculate 20 bit address.

However in present case using accumulator I/O technique 9 I/O ports are being provided on VMC-86 using three 8255-A chips.

I/O mapping for this is given in Chapter-1. Using these I/O ports only experimentation is being performed with the following Hardware modules.

1. Key Board Simulator Module
2. Digital I/O
3. Stepper Motor Control Module

The last one is discussed in next chapter.
Description of the former two follows:

TEST MODULE 1 : KEY BOARD

The three main types of peripherals connected to a microprocessor system are input devices, output devices and mass memory or bulk memory devices.

In the microprocessor world the keyboard is the input device most frequently used. The simplest keyboard is the hexadecimal keyboard, a 16 key keyboard. Hexadecimal keyboards are very inexpensive and are found on most microprocessor equipped appliances. Today many applicances such as television sets or washing machines incorporate microprocessors in their control system using a keyboard as an input device.

In general keyboard and display are required in the instruments for the man to machine communication. The various types of switches used in instruments are ON/OFF switch, PUSH/ Release switch,Band switches, Keyboard switches etc.

In microprocessor based equipments 'normally Open' type of switches are used. These switches give the change over of the contact as soon as they are pressed. The bouncing time of the keys i.e. unstable state of contact is maximum of 10 m sec. Pressing of the key is monitored by the keyboard controller part or by software written as in this case and necessary action is taken.

Keyboard can be easily interfaced to micro-processor through programmable I/O lines of the 8255, 8155 or 8279 (the keyboard display encoder and decoder. These are the two ways of connecting the keys to I/O lines.

In the first technique, each of the key switches can be attached separately to a bit of an input port. This technique is beneficial if the number of keys are less than 8. If the keys are more, than more number of input lines are required and multibyte operation is required. This leads to inefficient use of I/O lines, if one key is to be monitored at a time. The number of input lines can be easily reduced using the second technique where the keys are organised in matrix format. Each of the key is connected between a particular row and a column.

While connecting the keyboard through 8279, the four scan lines SL0-SL3 are decoded and used with return lines RL0-RL7 into a matrix form. As any of the key is pressed an interrupt is generated which is given to microprocessor to service the keyboard routine.

The keyboard module in this case has 20 keys connected in a matrix form of 8 x 3 matrix through 8255 as shown in Fig. 4.1. As shown in Fig. 4.1 this keyboard interface is non-encode type i.e. Hardware recognizes the key closure and encode it. The row of the matrix is connected through Port C (bit 0,1,2) and columnsare returned to PORT A (bit 0-7). All the 8 bits of the input port A are pulled down by 56K resistance to avoid any noise interference. When no key is pressed, the microprocessor reads the input as 00. The keyboard scanning starts by giving HIGH signal at PC0 and LOW signal at PC1 and PC2. If any of the key connected to PC0 through port A is pressed, the corresponding column bit will be also made to high and will be detected by software. Similarly the rest of the keys are scanned in the same fashion.

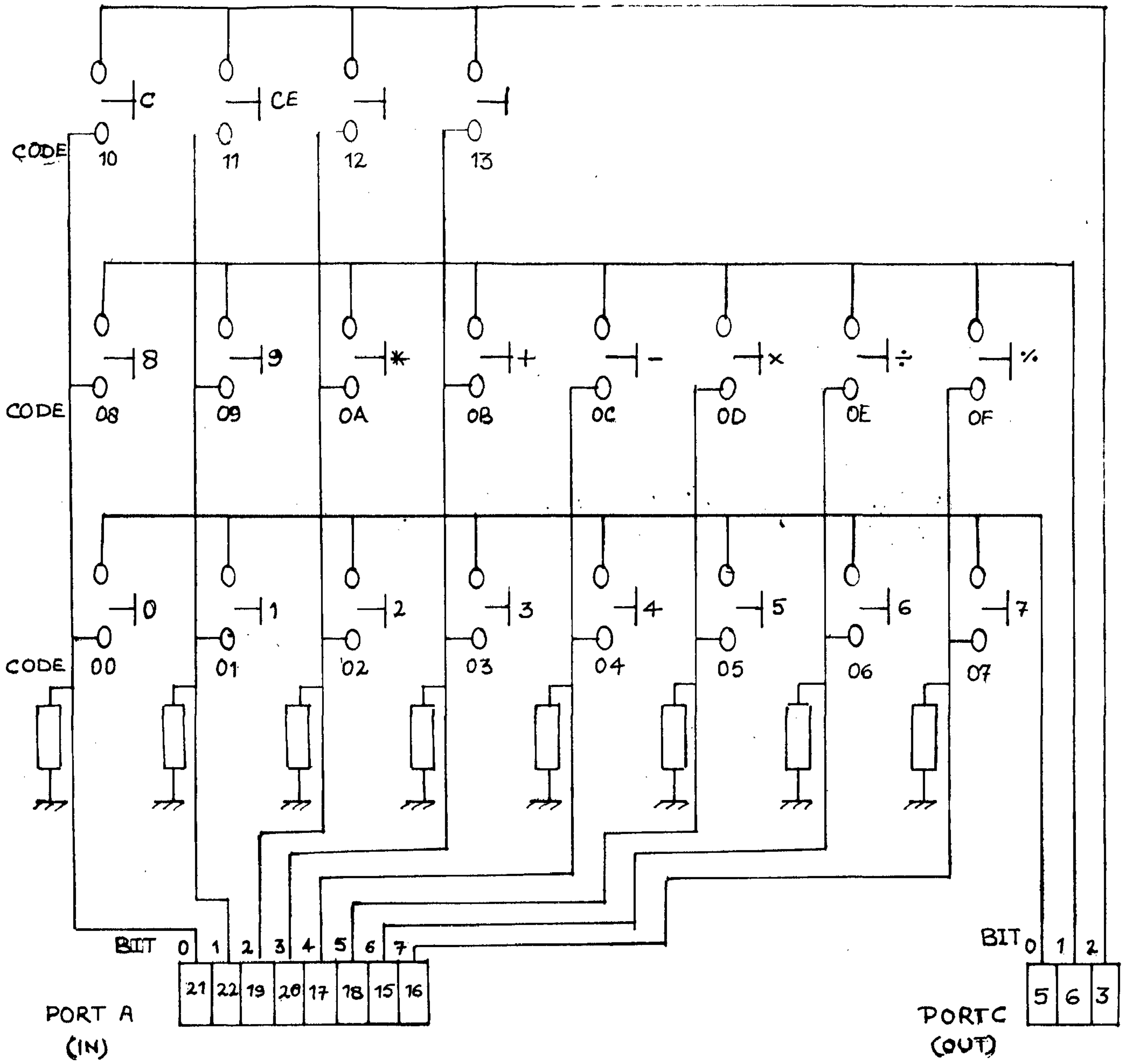
The following two experiments have been performed on this interface.

1. TO DISPLAY A CODE OF THE KEY PRESSED, IN THE DATA FIELD.THE CODE REMAINS IN THE DISPLAY TILL THE ACTIVATION OF THE NEXT KEY.

   note:- The Code assigned to each Key is given in the CKT diagram 4.2

2. TO EXECUTE A PROGRAM AS SOON AS A PARTICULAR KEY IS PRESSED.

   However one can also write a program to perform Decimal/Hex arithmetic as operated in calculator.



KEYBOARD INTERFACE

FUNCTION NAME : KBDR

INPUT : Nothing

OUTPUT : Code of the Key displayed in data field.

CALLS : CODE, DB OUTWORDS

DESTROYS : AX, BX,CX,DX,FL

DESCRIPTION : The port of 8255-1 is initialized in such a way that port A acts as an input port and port B and C as output ports. The three rows of the key are scanned one by one and process is repeated till the key is pressed. The information of code is then displayed and the monitor jumps back again to see if any other key is pressed.

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
KBDR	400	BA FF FF	MOV DX,FFFF	INITIALIZE PORT A- INPUTPORT $ PORT B,PORT C AS OUTPUT PORT
	403	B0 90	MOV AL,90	
	405	EE	OUT [DX],AL	
INIT	406	B7 00	MOV BH,00	INITIALIZE FINAL KEY CODE
	408	B3 01	MOV BL,01	INPUT WALKING PATTERN WITH LSB1
SCAN	40A	88 D8	MOV AL,BL	MOV THE WALKING PATTERN TO PORT C
	40C	BA FD FF	MOV DX,FFFD	
	40F	EE	OUT [DX],AL	

contd...

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	410	BA F9 FF	MOV DX, F9FF	READ THE STATE OF COLUMN THROUGH PORT A
	413	EC	IN AL,[DX]	
	414	E8 27 00	CALL CODE	CLASIFY THE 8 BIT WORD INTO 8 BITS
	417	3C 08	CMP AL,08	ANY KEY CLOSURE
	419	78 10	JS DISP	YES,GO TO DISPLAY IT
	41B	80 C7 08	ADD BH,08	INCREMENT PORTS CODE
	41E	80 FF 18	CMP BH,18	HAS PORT C CODE BECOME 18
	421	79 F3	JNS INIT	YES,START SCANNING FROM ROW 0
	423	88 D8	MOV AL,BL	NO, MOV THE WALKINING
	425	D0 D0	RCL AL,01	ONE TO SCAN NEXT LINE
	427	88 C3	MOV BL,AL	
	429	EB DF	JMP SCAN	CONTINUE SCANNING
DISP	42B	08 F8	OR AL,BH	
	42D	B4 00	MOV AH,00	
	42F	50	PUSH AX	
	430	B0 00	MOV AL,00	
	432	50	PUSH AX	
	433	B0 01	MOV A1,01	
	435	50	PUSH AX	
	436	50	PUSH AX	
	437	9A E0 CB 00 FF	CALL:DBOUTWORDS	DISPLAY THE CODE IN DATA FIELD GO TO SCAN THE KEYBOARD AGAIN
	43C	EB C8		

contd...

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
CODE	43E	08 CO	ORAL,AL	
	440	75 03	JNZ CODE2	
	442	BO 08	MOV AL,08	
	444	C3	RET	
CODE2	445	B5 00	MOV CH,00	
CODE5	447	DO C8	ROR AL,01	CHECK LSB
	449	72 04	JC CODE 10	IS LSB 1,YES: GOTO RETURN
	44B	FE C5	INC CH	NO,INCREMENT COUNTER
	44D	EB F8	JMP CODE5	CHECK THE NEXT BIT
CODE10	44F	88 E8	MOV AL,CH	
	451	C3	RET	

DIGITAL INPUT DIGITAL OUTPUT MODULE

This interface consists of simply 8 loggle switches and 8 outputs through LED'S. This interface demonstrates some of the basic programming techniques involved in the logic controllers or ladder networks. In ladder networks or logic controllers, the outputs are logical function of the inputs, performing functions by software and outputting the results. Thus the outputs are immediately set after a change in the inputs. The delay in time is equal to the time required to perform the logic function by software.

CIRCUIT DESCRIPTION:

The basic circuit diagram is shown in Fig. 4.3. As shown in Fig. 4.3 the system consists of eight input SPDT switches which give logic 0 or 1 signal to input lines of Port B. The output port A is buffered by open collector inverter, the 7406. The output LED's are connected to output buffer. The outputs of buffer are available on the top most left hand side of the connector, so that any device other than LED's can be connected.

The following experiments have been performed with the help of this test module.

1. Simple input port and output port checking.
2. Decade counter.

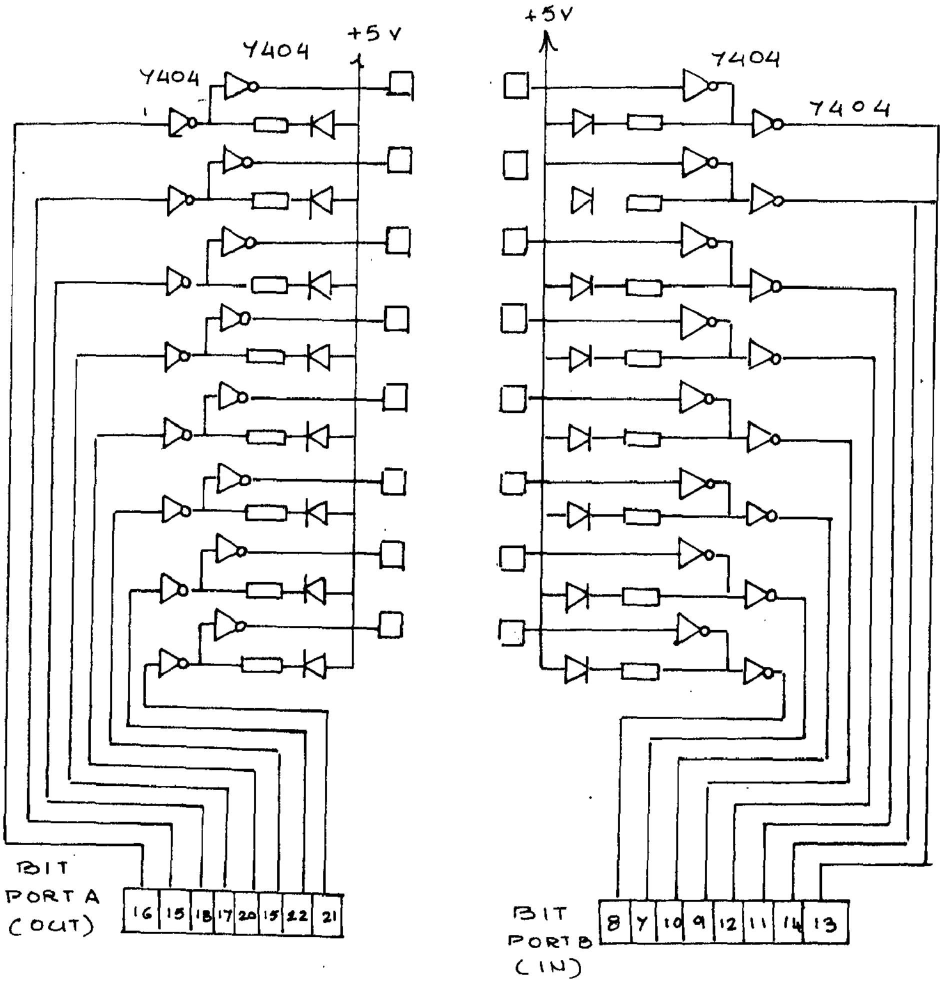


FIGURE 4·3 : DIGITAL I/O CARD

TEST PROGRAM-1

LADDER NETWORK SIMULATOR

FUNCTION NAME	:	LADDNS
INPUT	:	TO PORT FFFB
OUTPUT	:	FF TO OUTPUT PORT IF O.K.
CALLS	:	NONE
DESTROYS	:	Al,DX,BL
DESCRIPTION	:	The Ladder network Simulated is shown in Fig. 4.4

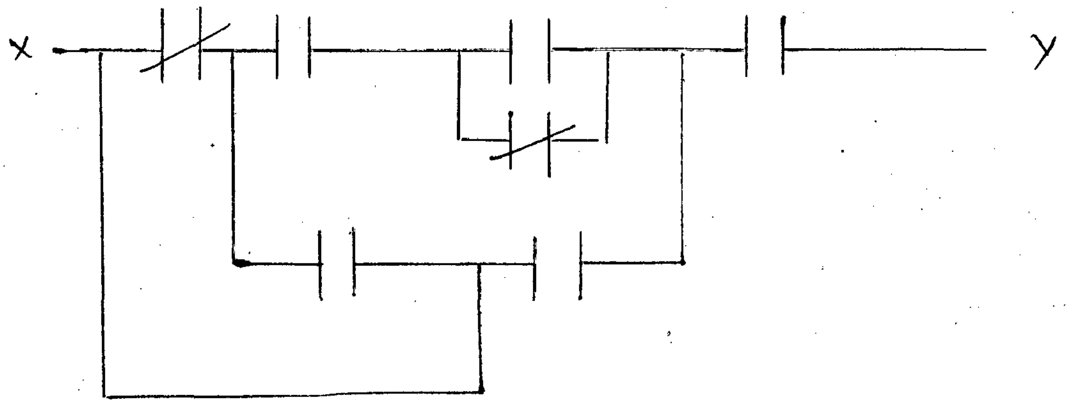


FIG 4.4

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS
	200	BO 82	MOV AL,82
	202	BA FF FF	MOV DX FFFF
	205	EF	OUT DX,AL
START	206	BA FB FF	MOV DX,FFFB
	209	EC	IN AL,DX
	20A	88 C3	MOV BL,AL
	20C	34 E4	XOR E4
	20E	3C OA	COMPOA
	210	74 1A	JZ SET
	212	88 D8	MOV AL,BL
	214	34 OF	XOR AL,OF
	216	3C OE	CMP OE
	218	74 12	JZ SET
	21A	88 D8	MOV AL,BL
	21C	34 96	XOR AL,96
	21E	3C 68	CMP AL,68
	220	74 OA	JZ SET
	222	88 D8	MOV AL,BL
	224	34 B7	XOR AL,B7
	226	3C 48	CMP AL,48
	228	74 02	JZ SET
	22A	EB 08	JMP LAST

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS
SET	22C	BO 01	MOV A1,01
	22E	BA F9 FF	MOV DX,FFF9
	231	EE	OUT DX,AL
	232	EB D2	JMP START
LAST	234	BO 00	MOV AL,00
	236	BA F9 FF	MOV DX, FFF9
	239	EE	OUT DX,AL
	23A	EB CA	JMP START

TEST PROGRAM-2 WITH DIGITAL I/O MODULE

FUNCTION NAME	:	DECUNTR
INPUT	:	Nothing/(Parameter for proper delay)
OUTPUT	:	Output at LED Port Varyinging from 0 to 9
CALLS	:	DELAY
DESTROYS	:	AX,CX,BX,FL,DX
DESCRIPTION	:	This program is written for Port of 8255-1 This program leads the output port (LED) to vary from 0 to 9 and repeating till again reset the up. The delay between two outputs can be varied as per need. Thus output of the program is decade counter.

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	300	BO 82	MOV HL,82	INITIALIZE THE I/O PORTS
	302	BA FF FF	MOV DX,FFFF	
	305	EE	OUT DX,AL	
	306	B1 OA	MOV CL,OA	
	308	BA F9 FF	MOV DX,FFF9	
	30B	BO 00	MOV AL,00	START COUNTING

conted.....

TABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	30D	EE	OUT DX,AL	
	30E	9A 00 05 00 00	CALL DELAY	WAIT TO NEXT COUNT
	313	FE C0	INC CL	
	315	FE C9	DEC CL	
	317	75 F4	JNZ LOOP	
	319	EB EB	JMP Repeat	
DELAY	500	BB FF FF	MOV BX,FFFF	DELY ELEMENT
DELY	503	4B	DEC BX	
	504	75 FD	JNZ DELY	TO WAIT
	506	C3	RET	

CHAPTER -5

CONTROLLING THE STEPPER MOTOR

INTRODUCTION:

Stepper motorcan be defined as an electromechanical transducer and is unique in digital electronics. It converts the digital pulses into precise angular steps of desired value or it can be defined as electromechanical transducer to convert digital information to positional information.

SMS have been developed as early as 1930s, but their expansion began in the 1960's with the advent of digital electronics. The main reason is that stepper motor itself is a digital device and can be easily interfaced with the digital systems.

The stepper motors are classified according to their internal characterstics, i.e. construction and operating principle rather than by their external characterstics like Torque Vs Speed, or number of steps per revolution. There are basically following types of stepper motors:

(a) Solenoid and Ratchet SM

(b) Permanent Magnet SM

(c) Variable Relluctance SM

(d) Harmonic Drive/Responsym SM

(e) Electrohydraulic SM.

The motor which is being used for the case study in this dissertation is PMBIFILAR WOUND SM, which is a special type of Permanent Magnet SM.

Few important terms often used in concern with the stepper motor are the following:

(i) STEP ANGLE: It is the angle through which the shaft of the unloaded stepper motor moves in response to a input pulse. θ_s is also called step size or resolution, and is determined by the number of teeth, excitation sequence etc. The common values are 7.5°, 6°, 4.5°, 3°, 1.6° and 1.2° per step.

(ii) ACCURACY : Tolerance is the maximum deviation from nominal value θ_s of the actual rotor displacement θ in response to a input pulse, under no load conditions. Three and fine percent are the usual accuracy tolerances. It is important to note that error in position of rotor is not cumulative, so that the error at the end N steps will still be 3 percent or 5 percent which ever the case may be.

(iii) TORQUE-DISPLACEMENT CURVE: The torque-displacement characterstics of an stepper motor are shown in FIG. 5.1. The curve is approximately sinusoidal in nature. Stable and unstable positions of the rotor are marked in the figure 5.1.

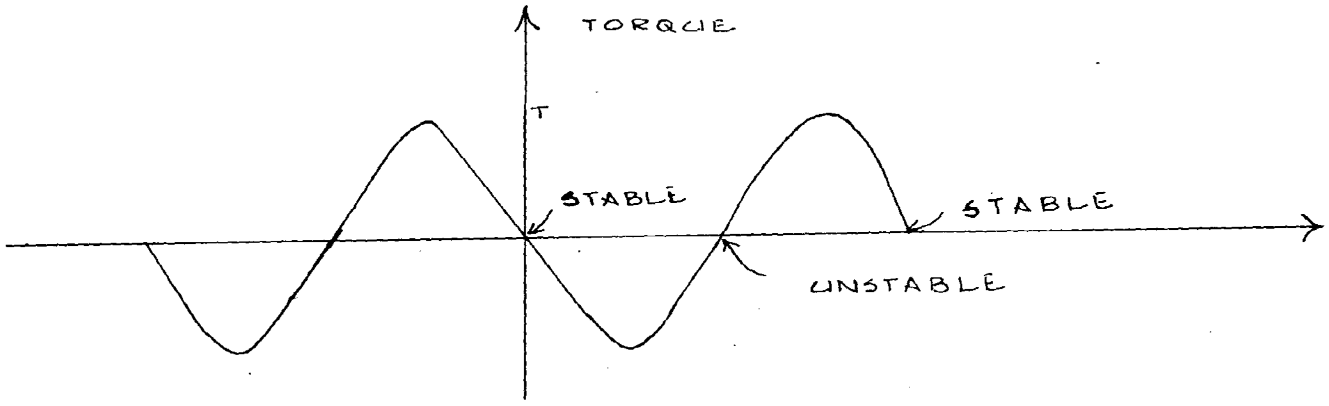


FIG -5.1 : TORQUE DISPLACEMENT CURVE FOR A STEEPER MOTOR

HOLDING TORQUE (TH)

It is the maximum load torque that can be applied to the rotor without causing the rotor to step from its stable equilibrium position.

DETENTE TORQUE :

It is the maximum rotor torque that can be applied to the unexcited motor without causing the rotor to step from its stable equilibrium position. It is much less than the HOLDING TORQUE.

TORQUE-STEPPING RATE CHARACTERSTICS:

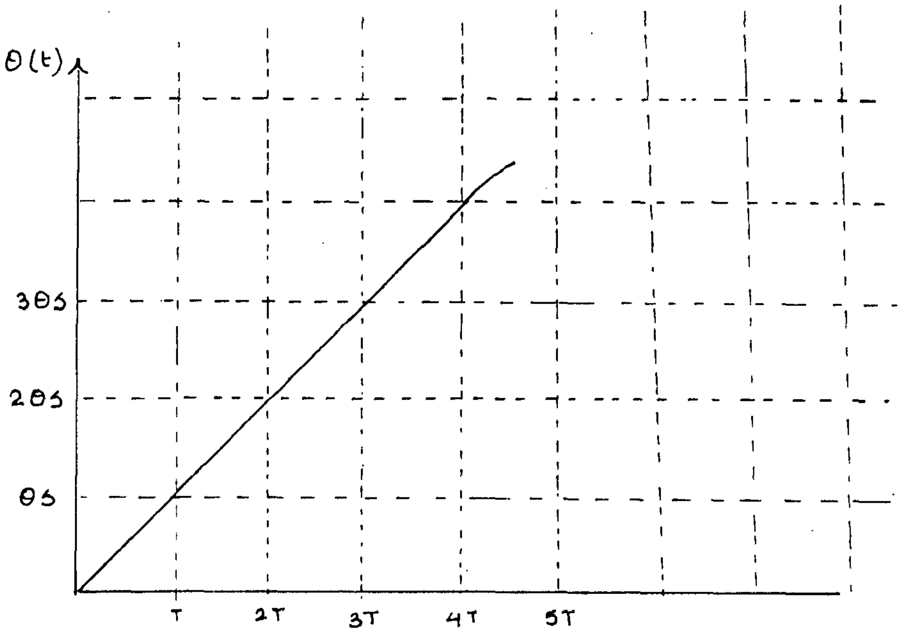
There are two modes of running the stepper motor.

(i) Single Stepping mode or start stop mode.

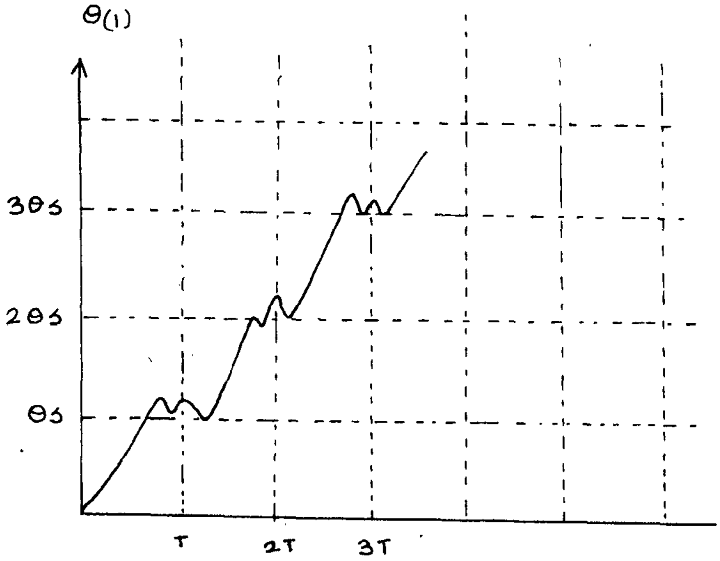
(ii)Slewing mode.

These two modes are given in Fig. 5.2. As shown in Fig. 5.2, in the single stepping mode, the rotor oscillates around the next equilibrium position and comes to rest before the next input pulse arrives. That is why the motor will start immediately on applying the input pulses, similarly it will stop without overshooting the mark the moment the input pulses are stopped.

On the other hand the rotor does not come to rest before the next input pulse arrives when it is slewing. In other words as the rotor is already in motion where the next pulse comes, the rotor can move to the next position much faster than



SLEWING MODE



SINGLE STEPPING MODE

FIGURE 5.2: MODES OF OPERATION OF S.M.

in the single stepping mode. However slewing means continuous running, As a consequence the motor will miss several steps if the input pulses are applied at slewing rate with the motor at rest. Similarly the motor will overshoot the mark by several steps if the input pulses are stopped suddenly.

PUL-IN TORQUE (TPI):

It is the maximum load torque against which the motor can start, stop or reverse without loosing a step when operating at a particular stepping rate.

PULL-OUT TORQUE (TPO) :

It is the maximum load torque against which the motor can slew at a given stepping rate without loosing a step.

PULL-IN RATE : It is the maximum stepping rate at which the motor will run in single stepping mode against a given load torque without missing a stOp.

PULL OUT RATE: It is the maximum stepping rate at which the motor will slew without missing a step against a given load torque.

RESPONSE RANGE:

It is the range of stepping rates in which the motor can start, stop or reverse against a given load torque without missing a step.

SLEWING RANGE:

It is the range of stepping rates in which the motor will stew without missing a step against a given load torque.

SALIENT FEATURES OF STEPPER MOTORS:

1. Instantaneous response to control pulses.
2. Holds on to the position infinitely in static condition.
3. No burn out due to locked rotor.
4. Speed can be varied over a wide margin.
5. High torque to inertial ratio. Can be overdriven without any damage.
6. Can be programmed for three parameters namely, speed, direction and number of steps.

Stepping motors differ from conventional Servomotors in following respects.

1. There is no control winding in stepping motors. Both windings are identical.
2. The stepping rate (speed of rotation) is governed by frequency of switching and not supply voltage.
3. A pulse input two phase Clock (instead of continuous pulses) will move the shaft of motor by one step for every pulse, thus number of steps to be moved can be precisely controlled.

4. When there is no pulse input, the rotor will remain locked up in the position in which the last step was taken. Since at any time the two windings are always energised which lock the rotor electromagnetically.

5. Stepping motors can be programmed in 3 parameters

(i) Direction

(ii) Speed

(iii) Number of steps

PRINCIPLE OF OPERATION:

The PM stepper motor can diagramatically be shown as in Fig. 5.3. It consists of two stator windings say A and B and a rotor having two magnetic poles N and S. When any one of the stator winding is energised the corresponding magnetic poles are generated in the stator. The rotor (permanent magnet) hence positions itself such that its poles look with the corresponding stator poles: When two windings are being energised, at the same time the rotor positions itself along the direction of resultant magnetic field.

The different combinations of excitation of stator windings along with the corresponding rotor position is shown in Fig. 5.4.

The motor which is being used in this case has 1.8^{o} step angle, it has 50 teeth on the rotor and 8 main poles on the stator. The step angle is given by

$$\therefore \quad \theta_s = \frac{360}{Nr.\ Kws}$$

Nr = no. of rotor teeth

Kws = excitation Sequence

The following 3 modes of operation of PM stepper motor are possible.

i. Single phase mode

ii. Two phase mode

iii. Hybrid mode

In single phase mode only one of the motor winding is excited at a time, There are 4 steps in excitation sequence. In two phase mode both the phases are excited at a time. In this mode also there are four steps in sequence excitation.

In both the above cases i.e. single and two phase mode the step angle $\theta_s = 90^\circ$ and excitation sequence factor is 2. However the rotor position is 45° away from those in the single phase mode.

HYBRID mode is the combination of single phase and two phase.

In the present case the HYBRID MODE has been used, in the hybrid mode of operation the voltage +V is applied during certain steps while voltage - V is also applied sometimes. So this requires a bipolar regulated supply and a pair of SPDT switches. To avoid this each of the stator winding is

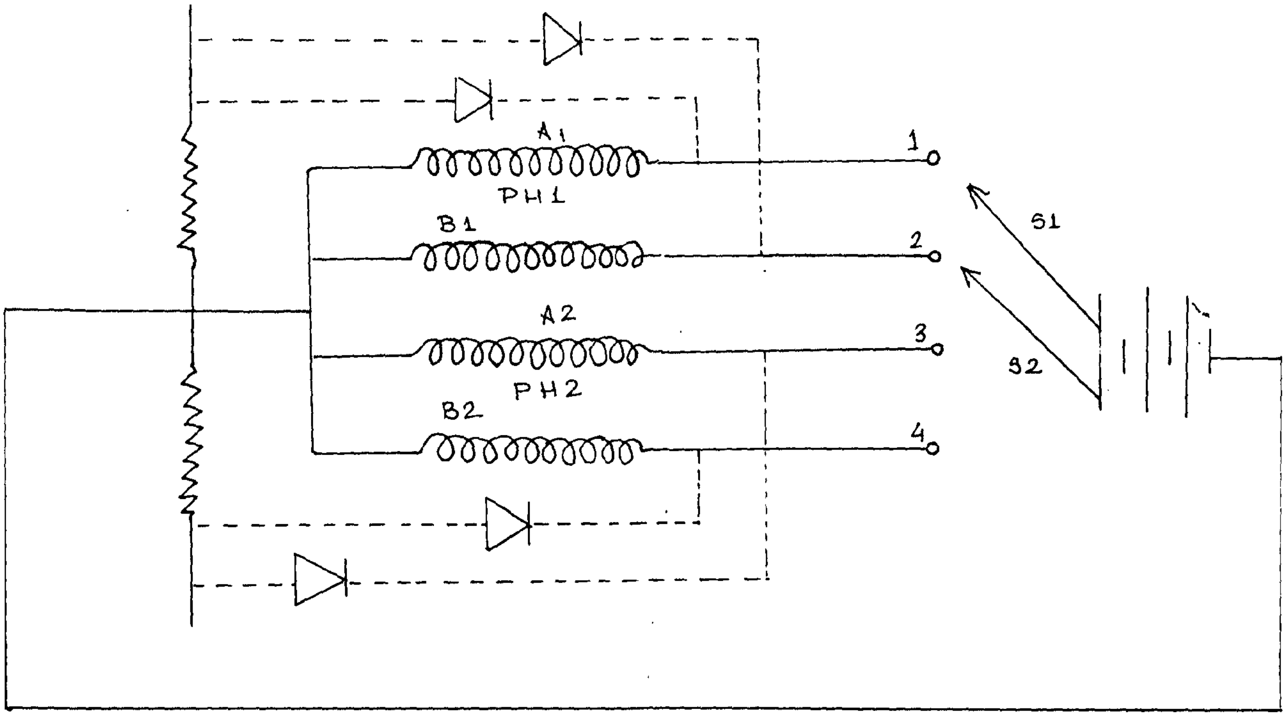


FIGURE - 5·4

SWITCHING DIAGRAM FOR EXCITATION OF WINDINGS splited into two sections A1,A2 and B1,B2. These sections are wound differentially. These winding sections can now be excited from a unipolar regulated supply. This type of construction of PM stepper motor is called bifilar winding construction. Advantage achieved by doing so is,reduced winding inductance and consequently improved torque stepping rate characterstics.

The motor being used in present case is PM bifilar Wound Stepper Motor operated in Hybrid mode.

Stepping motor being used is of bifiler wound with six leads. Each of the two phases has double winding with a centre tap. The advantage achieved by doing centre tapping is that switching the supply from one side to another of phase causes reversal of magnetic polarity without actually reversing the polarity of supply. Four step input sequence gives 1.8°(full) after and eight step input sequence give 0.9°(half) step function.

The switching sequence along with 4 and 8 step input sequence is given in Table 5.1. The switching sequence given in table 5.1 will move the shaft in one direction, to reverse the direction of movement one has to follow the reverse sequence i.e. from down to upward.

TABLE 5.1

SWITCHING SEQUENCE

4STEP INPUT SEQUENCE				8 STEP INPUT SEQUENCE			
PH-1		PH-2		PH-1		PH-2	
A-1	B-1	A-2	B-2	A-1	B-1	A-2	B-2
1	0	1	0	1	0	1	0
0	1	1	0	0	0	1	0
0	1	0	1	0	1	1	0
1	0	0	1	0	1	0	0
				0	1	0	1
				0	0	0	1
				1	0	0	1
				1	0	0	0

note: In above table positive logic has been adopted. However in present case the 4 step input sequence has been used.

NEED OF STABLIZED CURRENT :

As torque is directly proportional to the current in winding. The current in winding is governed by the d.c. resistance of the winding. As the switching sequence starts the inductive reactance of the winding which increases with frequency of switching, it opposes the rise of current to desired level within the time given for one step depending upon the frequency of stepping. This is mainly due to L/R time constant of winding. The drop in current level causes drop in torque as the speed increases. In order to improve torque at high speeds it is necessary to maintain current at the desired level. This can be achieved by one of the following methods.

1. By increasing supply voltage and introducing current limiting resistances in each phase. Introduction of resistances improves the time constant of the winding.
2. By using a constant current source with or without a chopper instead of using a constant voltage source which will give even better performance.

STARTING AND STOPPING UNDER LOAD:

There is a limit for every type of stepping motor as regards the speed at which it will start and stop without loosing step. The limit is due to load torque and load inertial To overcome this acceleration and decceleration techniques have to be employed. In this technique the stepping rate on switching should be very low and should increase to desired level gradually depending on inertia to be encountered, similarly while stopping the stepping rate should be decreased gradually.

Stepper motor Controller Card being used in present dissertation is shown in Fig. 5.5.

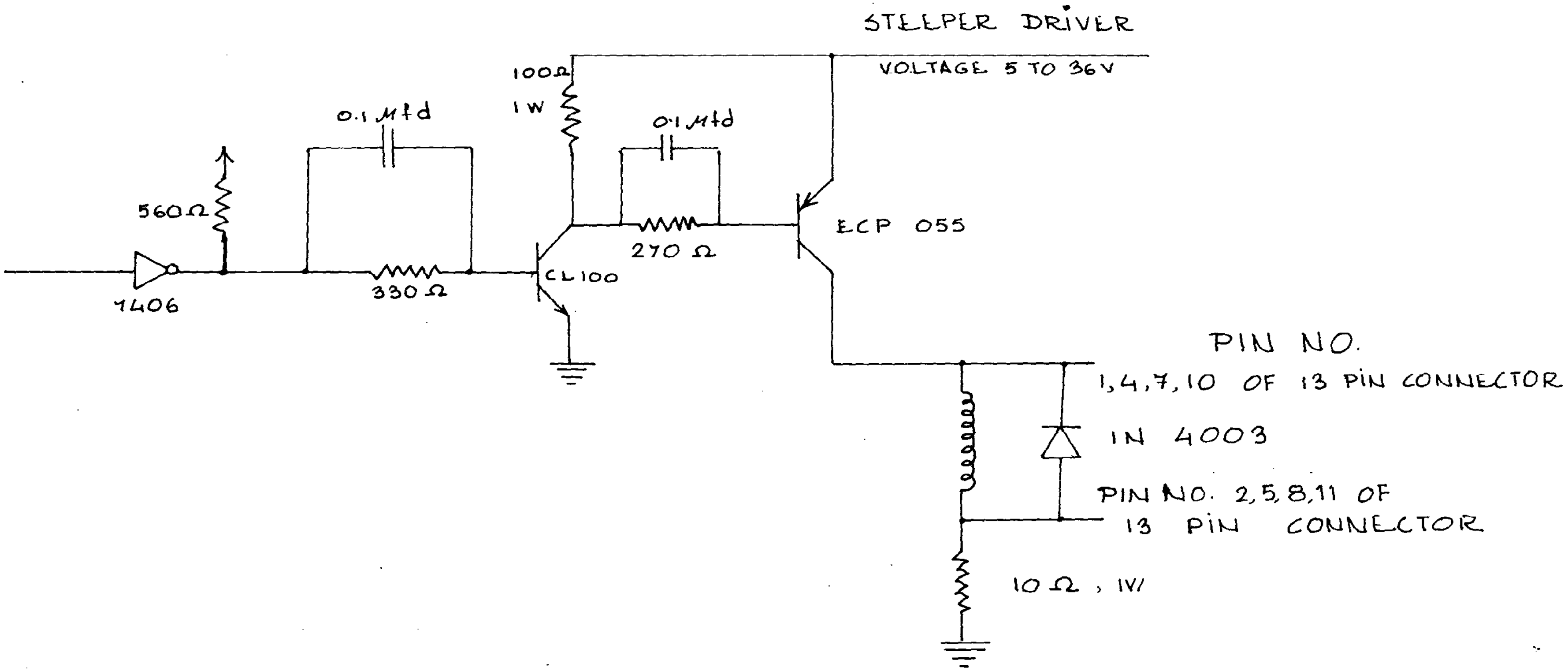


FIGURE 5·5 : STEPPER MOTOR _ CONTROLLER CARD

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
30B	30B	EE	OUT DX	
	30C	E8 21 00	CALL DELAYR	
	30F	BO -	MOV AL,CODE 2	
	311	EE	OUT DX	
	312	E8 1B 00	CALL DELAY R	
	315	BO -	MOV AL,CODE 3	
	317	EE	OUT DX	
	318	E8 15 00	CALL DELAYR	
	31B	BO -	MOV AL,CODE 4	
	31D	EE	OUT DX	
	31E	E8 OF 00	CALL DELAYR	
	321	EB E3	JMP START	
DELAYR	330	BB - -	MOV BX, WHT1	
	333	E8 07 00	CALL DELAY	
	336	BB - -	MOV BX,WHT2	
	339	E8 01 00	CALL DELAY	
	33C	C3	RET	
	33D	4B	DEC BX	
	33E	90	NOP	
	33F	90	NOP	
	340	90	NOP	

contd..

In present case following 4 experiments have been performed on stepper motor.

1. Velocity (r.p.m) control
2. Linear Displacement Control
3. S.H.M. with uniform Velocity
4. Selecting a point in x-y plane.

The required software for the above follows:

EXPERIMENT-1

FUNCTION NAME	:	SMVELC
INPUT	:	DELAY ELEMENT AS PER DESIRE CODES AS PER THE DIRECTION OF MOVEMENT
OUTPUT	:	DESIRED SPEED OF STEPPER MOTOR
CALLS	:	NONE
DESTROYS	:	ALDX
DESCRIPTION	:	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	300	BA FF FF	MOV DX, FFFF	
	303	BO 80	MOV Al, 80	
	305	EE	OUT DX	
START	306	BA F9 FF	DX, FFF9	
	309	BO	MOV AL,CODE 1	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	341	75 FA	JNZ DELAY	
	343	C3	RET	

NOTE :

1. The CODE 1, CODE2, CODE3 and CODE 4 are to be fed as follows:

   FOR CLOCKWISE MOVEMENT: CODI≡FA, CODE 2≡ F6, CODE 3 ≡ F5, CODE 4 ≡ F9.

   FOR ANTICLOCKWISE MOVEMENT: CODE 1≡ F9, CODE 2≡F5, CODE3 ≡ F6, CODE 4 ≡ FA.

2. WHT1 and WHT2 may be equal or may not be.

   The following readings have been taken.

S.No.	WHT1	WHT2	R.P.M.	S.No.	WHT 1	WHT 2	R.P.M.
1	100	100	50	5	1001	0100	6
2	200F	0F00	3	6	0A01	0001	10
3.	050F	000F	20				
4.	0F00	0F00	33.3				

EXPERIMENT -2

FUNCTION NAME : LDC

INPUT : None

OUTPUT : Linear displacement

CALLS : None

DESTROYS : AL, DX, BX, Cx

DESCRIPTION : The program has been written using Clockwise Codes for one dimensional movement, can be modified for reverse direction movement.

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	2FD	B9 -- --	MOV CX, COUNT	
	300	BA FF FF	MOV DX, FFFF	
	303	BO 80	MOV AL,80	
	305	EE	OUT DX	
	306	BA F9 FF	MOV DX, FFF9	
	309	BO FA	MOV AL, FA	
	30B	EE	OUT DX	
	30C	E8 21 00	CALL DELAYR	
	30F	BO F6	MOV AL, F6	
	311	EE	OUT DX	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	312	E8 10 00	CALL DELAYR	
	315	BO F5	MOV AL, F5	
	317	EE	OUT DX	
	318	E8 15 00	CALL DELAYR	
	31B	BO F9	MOV AL,F9	
	31D	EE	OUT DX	
	31E	E8 OF OO	CALL DELAYR	
	321	49	DEC CX	
	322	75 E2	JMP NZ STRT	
	324	F4	HLT	
DELAYR	330	BB 00 01	MOVBX,CONST	
	333	E8 07 00	CALL DELAY	
	336	BB 00 01	MOV BX,CONST	
	339	E8 01 00	CALL DELAY.	
	33C	C3	RET	
DELAY	33D	4B	DEC BX	
	33E	90	NOP	
	33F	90	NOP	
	340	90	NOP	
	341	75 FA	JNZ DELAY	
	343	C3	RET	

The following reading have been taken

COUNT VALUE	DISPLACEMENT
30	2.2 Cm
25	1.8 Cm
20	1.3 Cm
15	0.8 Cm
10	0.7 Cm

EXPERIMENT - 3

FUNCTION NAME	SHM
INPUT	NONE
OUTPUT	SHM
CALLS	DELAY
DESTROYS	CX, DX,AL,BX

DESCRIPTION

LABEL	ADDRESS	CONTENTS	MNEMONIC AND OPERANDS	COMMENTS
	300	B9 20 00	MOV CX, 20	≡ Amplitude
	303	BA FF FF	MOV DX,FFFF	Initialize 8255-1
	306	BO 00	MOV AL,80	ports
	308	EE	OUT DX	
	309	BA F9 FF	MOV DX, FFF9	
LOOP1	30C	BO FA	MOV AL,FA	
	30E	EE	OUT,DX	
	30F	E8 3E 00	CALL DELYR1	Delay Between two steps.
	312	BO F6	MOV AL,F6	
	314	EE	OUT, DX	
	315	E8 3800	CALL DELYR1	
	318	BO F5	MOV AL,F5	
	31A	EF	OUT, Dx	
	31B	E8 32 00	CALL DELYR1	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	31 E	BO F9	MOV AL,F9	
	320	EE	OUT, DX	
	321	E8 2C 00	CALL DELYR1	
	324	49	DEC CX	IS AMPLITUDE IS ACHIEVED.
	325	75 E5	JNZ LOOP 1	NO CONTINUE THE MOVEMENT.
	327	B9 40 00	MOV CX,40	YES INITIALIZE COUNTER FOR REVERSE MOVEMENT
	32A	BO F9	MOV AL,F9	
	32C	EF	OUT,DX	
	32D	E8 3E 00	CALL DELYR2	
	330	BO F5	MOV AL,F5	
	332	EE	OUT,DX	
	333	E8 38 00	CALL DELYR2	
	336	BO F6	MOV AL,F6	
	338	EE	OUT,DX	
	339	E8 32 00	CALL DELYR2	
	33C	BO FA	MOV AL,F A	
	33E	EE	OUT,DX	
	33F	E8 2C 00	CALL DELYR2	
	342	49	DECCX	IS AMPLITUDE IS ACHIEVED.

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
		75 E5	JNZ LOOP2	NO,CONTINUE MOVEMENT.
		B9 40 00	MOV CX,40	YES,INITIALIZE COUNTER.
		E8 C1 FF	JMP LOOP 1	REPEAT THE MOTION
DELYR1	350	BB FF FF	MOV BX,FFFF	
	353	E8 07 00	CALL DELY1	
	356	BB FF FF	MOV DX,FFFF	
	359	E8 01 00	CALL DELY1	
	35C	C3	RET	
DEIY1	35D	4B	DEC BX	
	35E	90	NOP	
	35F	90	NOP	
	360	90	NOP	
	361	75 FA	JNZ DELY1	
	363	C3	RET	
DELYR2	36E	BB FF FF	MOV BX,FFFF	
	371	E8 07 00	CALL DELY 2	
	374	BB FF FF	MOV BX, FFFF	
	377	E8 01 00	CALL DELY 2	
	37A	C3	RET	
DELY2	37B	4B	DEC BX	
	37C	90	NOP	
	37D	90	NOP	
	37E	90	NOP	
	37F	75 FA	JNZ DELY2	
	381	C3	RET.	

EXPERIMENT 4

FUNCTION NAME : SELAP

INPUT : None

OUTPUT : The point (desired) would be selected in plane.

CALLS : XMOV, YMOV

DESTROYS : AL,CX,DX,BX

DESCRIPTION :

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS.	COMMENTS
	400	B9 20 00	MOV CX,XM	FOR X-MOVEMENT
	403	9A 00 02 00	CALL XMOV	
	407	B9 20 00	MOV CX,YM	FOR y-MOVEMENT
	40A	9A 00 03 00	CALL YMOV	
	40E	C3	RET	STOP THE MOVEMENT.

FUNCTION NAME : XMOV
INPUT : NONE
OUTPUT : X-MOVEMENT OF PLANE
CALLS : NONE
DESTROYS : Al,CX,DX,BX
DESCRIPTION : According to the desired quadrant we have to vary the codes to be outputed or to vary intiliazation of the point 0,0.

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	200	BA FF FF	MOV DX,FFFF	INITIALIZE
	203	B0 80	MOV AL,80	8255-1 PORTS
	205	EE	OUT, DX	
REP	206	BA F9 FF	MOV DX FFF9	
	209	B0 FA	MOV AL,FA	
	20B	E8 21 00	CALL DELYR	DELAY BETWEEN TWO STEPS
	20E	B0 F6	MOV Al,F6	
	220	EE	OUT, DX	
	211	E8 1B 00	CALL, DELYR	
	214	B0 F5	MOV AL,F5	
	216	EF	OUT,DX	
	217	E8 15 00	CALL,DELYR	
	21A	B0 F9	MOV AL,F9	
	21C	EE	OUT,DX	
	21D	E8 0F 00	CALL DELYR	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	220	49	DEC CX	CHECK X, IF O.K.
	221	75 E2	JNZ REP	NO,CONTINUE MOVEMENT
	223	C3	RET	YES,STOP THE MOVEMENT
	230	BB 0001	MOV BX,100 CALL DELY	
	236	BB 00 01	MOV BX,100	
	239	E8 01 00	CALL,DELY	
	23C	C3	RET	
DELY	23D	4B	DEC BX	
	23E	90	NOP	
	23F	90	NOP	
	240	90	NOP	
	241	75 FA	JNZ DELY	
	243	C3	RET	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
322	75 E2			
	322	75 E2	JNZ REP1	
	324	C3	RET	
DELYR5	330	BB 00 01	MOV BX,100	
	333	E8 07 00	CALL DELY1	
	336	BB 00 01	MOV BX,100	
	339	E8 01 00	CALLDELY1	
	33C	C3	RET	
DELY1	33D	4B	DEC BX	
	33E	90	NOP	
	33F	90	NOP	
	340	90	NOP	
	341	75 FA	JNZ DELY 1	
	343	C3	RET	

FUNCTION NAME Y MOV

INPUT None

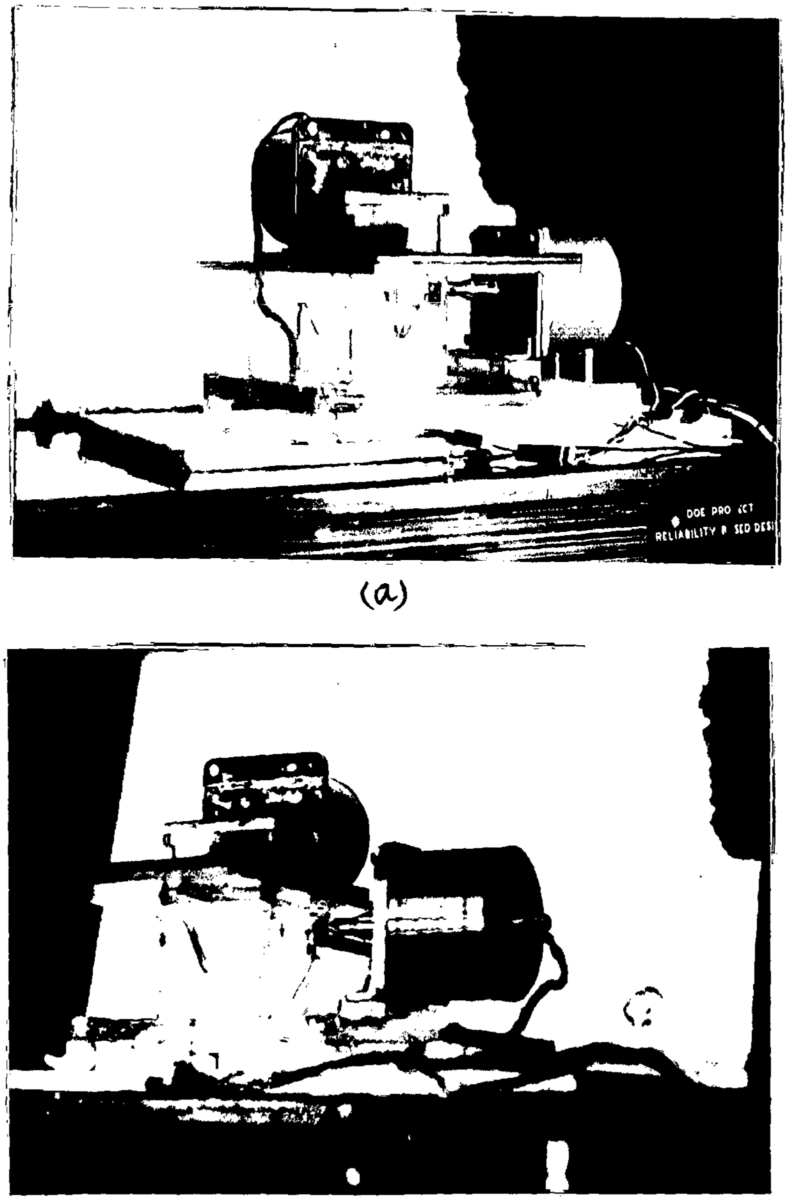
OUTPUT Y-MOVEMENT OF A PLANE

CALLS NONE

DESTROYS AL, CX,BX,DX

DESCRIPTION:

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	300	BB FF FF	MOV DX,FFFF	
	303	BO 80	MOV AL,80	
	305	EE	OUT DX	
REP1	306	BA F8 FF	MOV DX,FFF8	
	309	BO F9	MOV Al, F9	
	30B	EE	OUT DX	
	30C	E8 21 00	CALL DELYRS	
	30F	BO F5	MCV AL,F5	
	311	EE	CUT, DX	
	312	E8 1B 00	CALL DELYRS	
	315	BO F6	MCV AL,F6	
	317	EE	OUT, DX	
	318	E8 15 00	CALL ,DELYRS	
	31B	BO FA	MOV AL,FA	
	31D	EE	OUT,DX	
	31E	E8 OF 00	CALL DELYRS	
	321	49	DEC CX	



(a)

(b)

Fig 6.1

CHAPTER - 6

DISCUSSION AND CONCLUSION

DISCUSSION

The purpose of present dissertation is to understand basically the microprocessor 8086. So brief discussion of pin signals and its architecture is included in Appendix - G. In comparision to the earlier microprocessors, it has wide variety of instructions in its instruction set. To understand exactly what each instruction results into and what are its limitations requires experience. However to eloborate the above fact few examples have been considered for the development of the software modules and presented in Chapter-3.

To realize the I/O technique few Hardware modules have also been tested and presented along with the required software in Chapter -4.

Description of microcomputer VMC-86 presented in the very first chapter, Chapter-1, and it includes the memory decoding and I/O decoding available for 8086 which differs slightly from other microprocessors due to control signal $\overline{BHE}$.

To incorporate the application of 8086 in a process control application a simple model (mechanical) has been fabricated as shown in Fig. 6.1. With this model tha following experiments have been performed:

1. Linear Displacement Control
2. Simple Harmonic Motion in a plane with uniform velocity.

3. Selection of a point on an x-y plane.

Also the speed control of stepper m●tor has been achieved by interfacing an seperate stepper motor withe VMC-86 micro-computer. For demonstration purpose some commands for CRT- VTZ-10 have been developed as per given in appendix- H.

CONCLUSION: LIMITATIONS AND EXTENSION OF PRESENT WORK

Stepper motors control has been considered as the case study in the present dissertation but stability of the stepper motor has not been considered. Max. stepping rate which can be achieved and the selection of the stepper motors for a particular application could not be considered due to time factor, this can be considered as extension of present work.

The very first extension which can be done is the P.C.B. drilling M/C control, for drilling minute hol●s considering minute distances. For it the Numeric processor 8087 can also be encorporated to take care of various calculations so that the software could be reduced.

Curve tracing of following two types can be achieved.

1. Variation of a single parameter w.r.t time.
2. Variation of the one parameter w.r.t another variable not time as eg. Sine wave parabola, Hyperbola etc.

Marking on the metal sheets can also be achieved up to a certain accuracy greater than the accuracy which can be achieved by hand.

In extension of this a m/c can be designed to engrave the names on the metals. In this case software for different alphabet would be stored in monitor and monitor would have to be written in such a way that it would scan simply the Key pad provided.

In the present dissertation few examples have been considered to eloborate the fact that, how the 8086 could be applicable in the process control application. So many things could have been done but could not been due to availability of limited time. For more examples have been considered in the above paragraph which are possible as an extension of this work. As for as the extension of present work and applications of the microprocessor 8086 is concerned the sky is the only LIMIT.

APPENDIX-A

CLOCK GENERATOR: 8284

SALIENT FEATURES:

1. 18 pin package
2. Generates the System Clock for iAPX 86,88 processors.
3. Available in two versions

   8284A 5MHz to 8 MHz.

   8284A-1 10 MHz.

4. Uses a Crystal or TTL Signal for Freq. Source.
5. Provides Local READY and Multibus READY synchronization.
6. Single +5V Supply.
7. Also generates system Reset output from Schmitt trigger input.
8. Capable of Clock Synchronization with other 8284 As.

FUNCTIONAL DESCRIPTION OF PINS FOLLOWS:

SYMBOL	TYPE	NAME AND FUNCTION
$\overline{\text{AEN1}}$	1	Address Enable: is an ACTIVE LOW signal. AEN serves to qualify its respective Bus Ready Signal (RDY1 or RDY2). AEN1 validates RDY1 and AEN2 validates RDY2. Two AEN signal input are used in multiprocessor system.

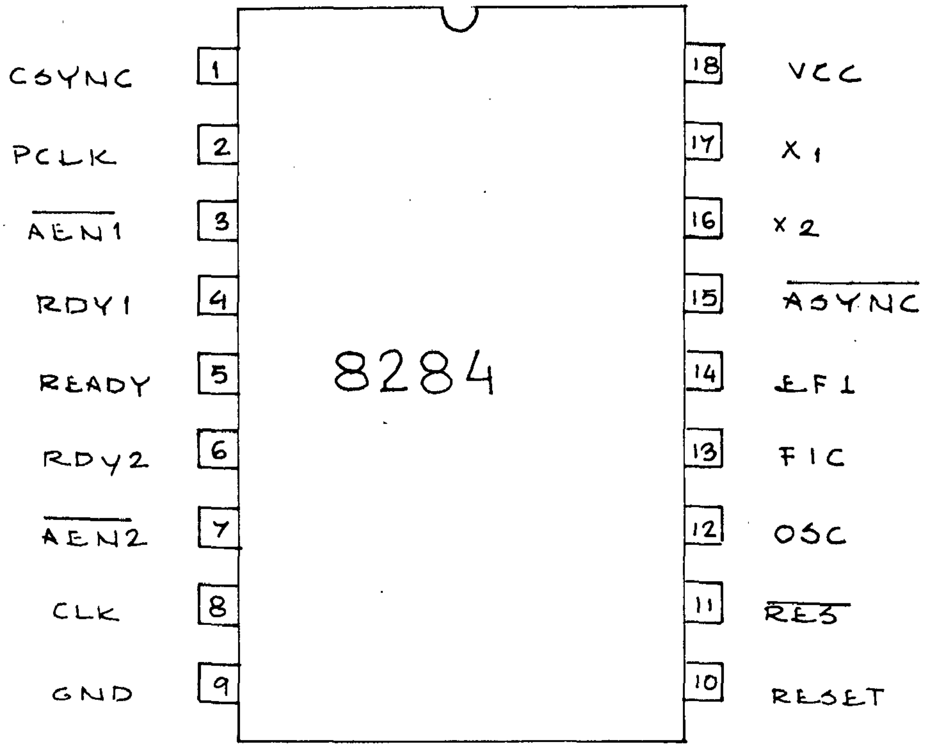


FIGURE 1(a): PIN CONFIGURATION

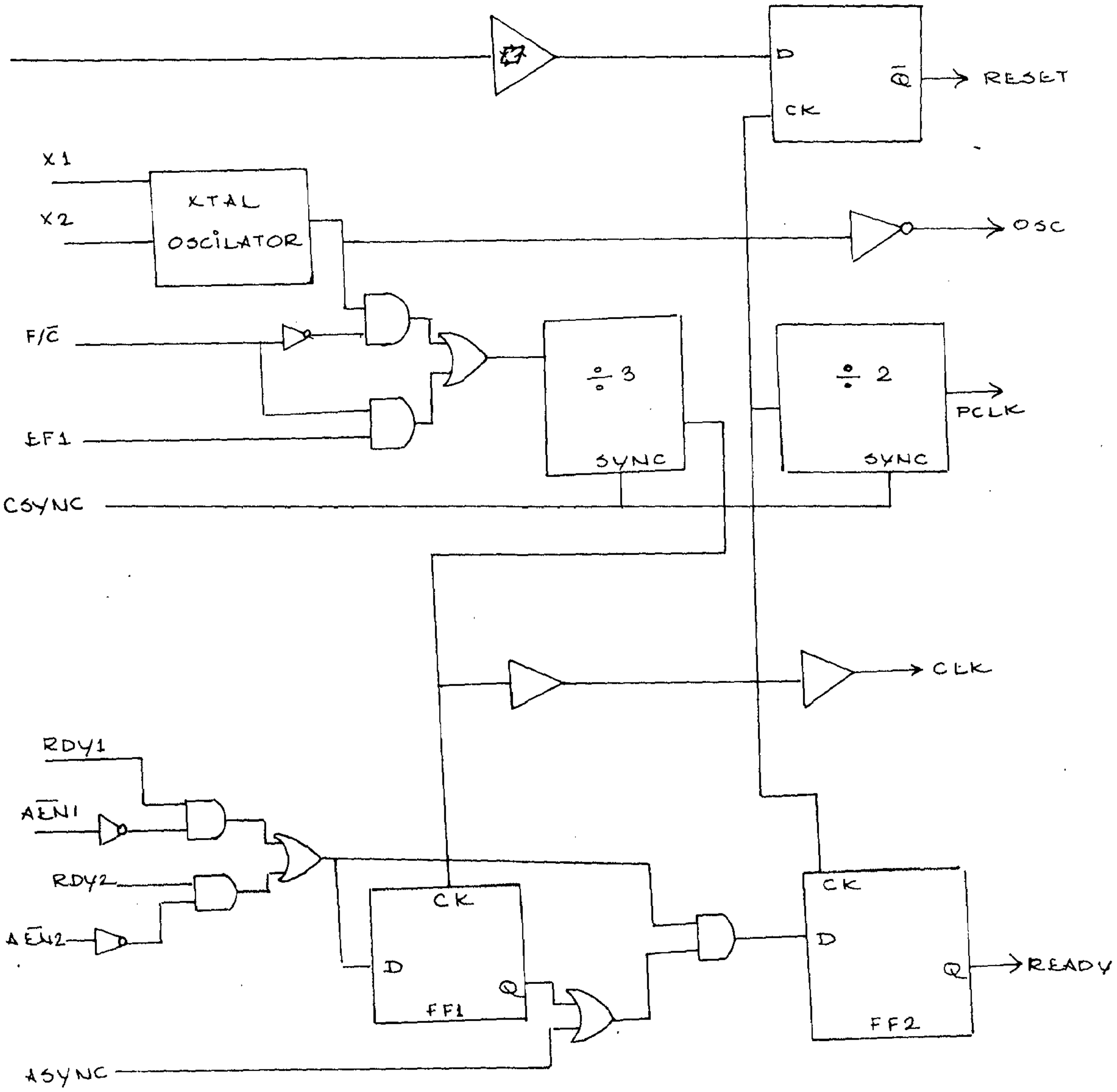


FIGURE 1(b) : INTERNAL LOGIC: 8284

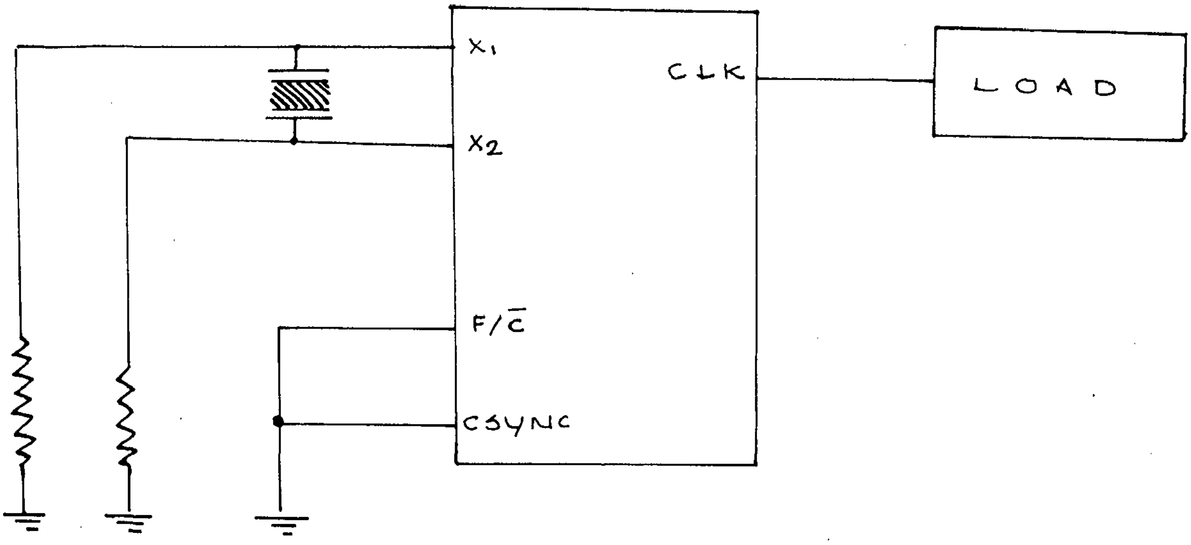


FIGURE 2 : CLOCK TIMING USING XTAL AT X1 X2

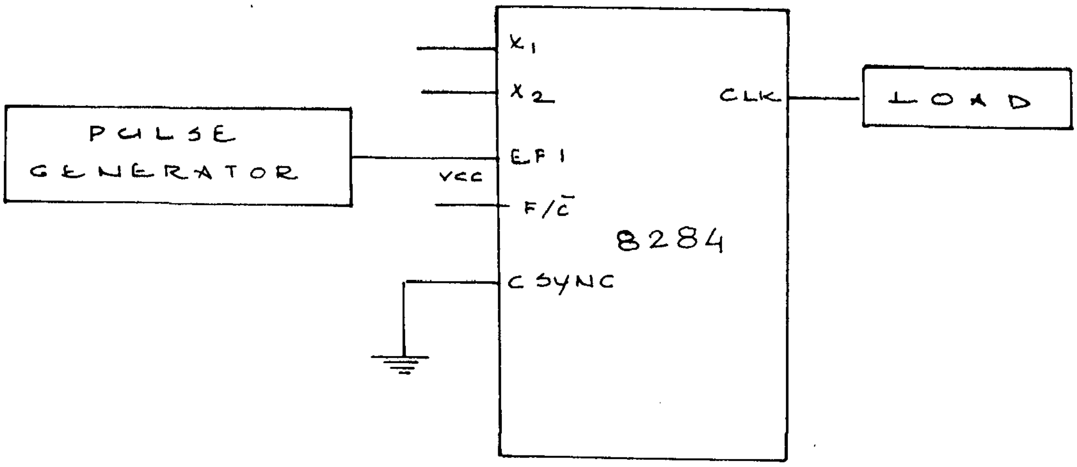


FIGURE 2 (b): CLOCK TIMING USING EFI INPUT

SYMBOL	TYPE	NAME AND FUNCTION
RDY1, RDY2	I	Bus Ready RDY is ACTIVE HIGH signal which is an indication from a device located on the system data bus that data has been received or is available. RDY1 is qualified by AEN1 and RDY2 by AEN2.
$\overline{ASYNC}$	I	Ready Synchronization Select: Its the input signal which defines the synchronization mode of the READY logic.
READY	0	Ready: READY is the ACTIVE HIGH input signal which is the synchronized RDY signal input.
X1,X2	I	Crystal ln: X_1 and X_2 are the pins to which the Xtal is attached. The Xtal frequency is 3 times the desired processor clock frequency.
$F/\overline{C}$	I	Frequency/Crystal Select: Its an Strapping option. When Strapped LOW F/C permits the processor clock to be generated by Xtal, if strapped HIGH the processor clock is generated by EFl input.
EFl	I	External frequency Input used to generate the Clock when FIC is strapped HIGH. EFI is the simply square wave of frequency 3 times the desired clock frequency.
CLK	0	Processor CLK - is the Clock output used by the processors and is directly connected to processor's CLK input.

SYMBOL	TYPE	NAME AND FUNCTION
PCLK	O	Peripheral Clock : is TTC level peripheral clock signal whose output freq. is 1/2 of the clock and has 50 percent duty cycle.
OSC	O	Oscillator output: is TTL level output of the internal oscillator whose freq. is equal to that of crystal frequency.
$\overline{RES}$	O	Reset ln : $\overline{RES}$ is an ACTIVE LOW signal which is used to generate RESET. The 8284 provides the Schmitt trigger input so that an RC connection can be used to establish the power up Reset of proper duration.
RESET	O	Its an ACTIVE HIGH signal, is generated using $\overline{RES}$ and is used to Reset the 8086 family processors.
CSYNC	I	Clock Synchronization: is an ACTIVE HIGH signal and is used to synchronize two or more 8284 (clock Generator).
GND	I	Ground
V_{cc}	I	Power : + 5 V supply.

APPENDIX - B

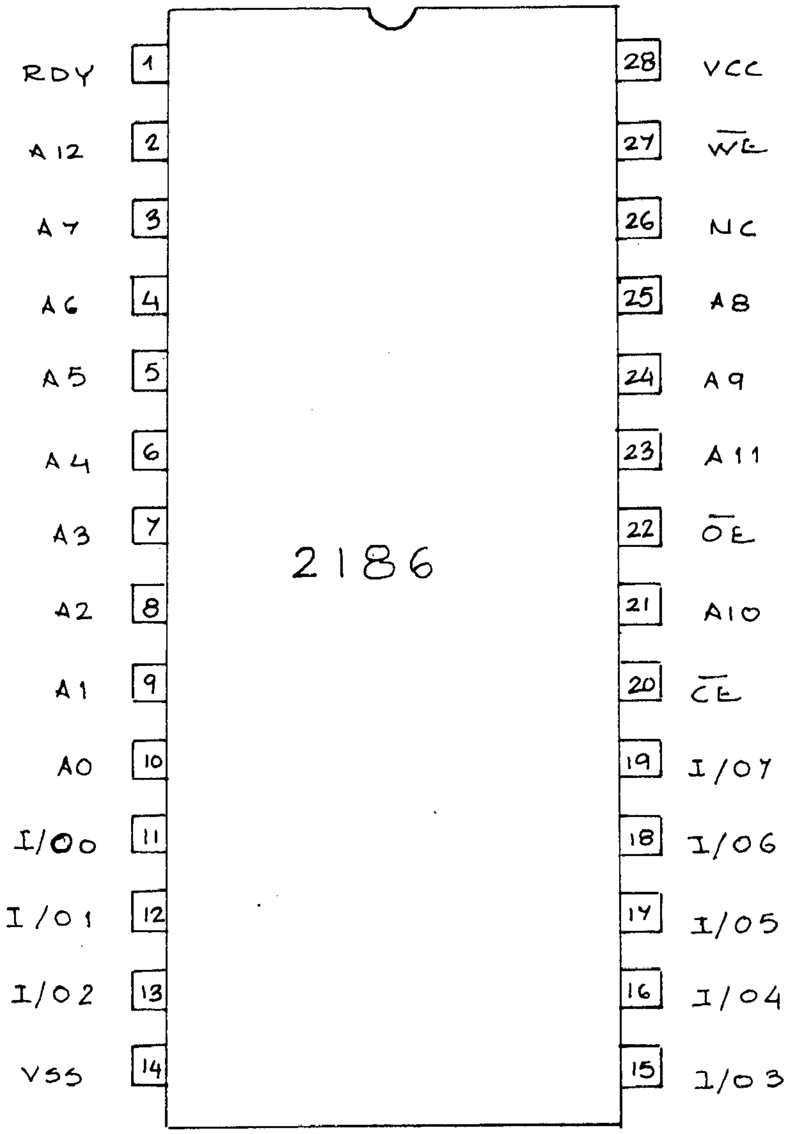


FIGURE 3: PIN CONNECTION DIAGRAM

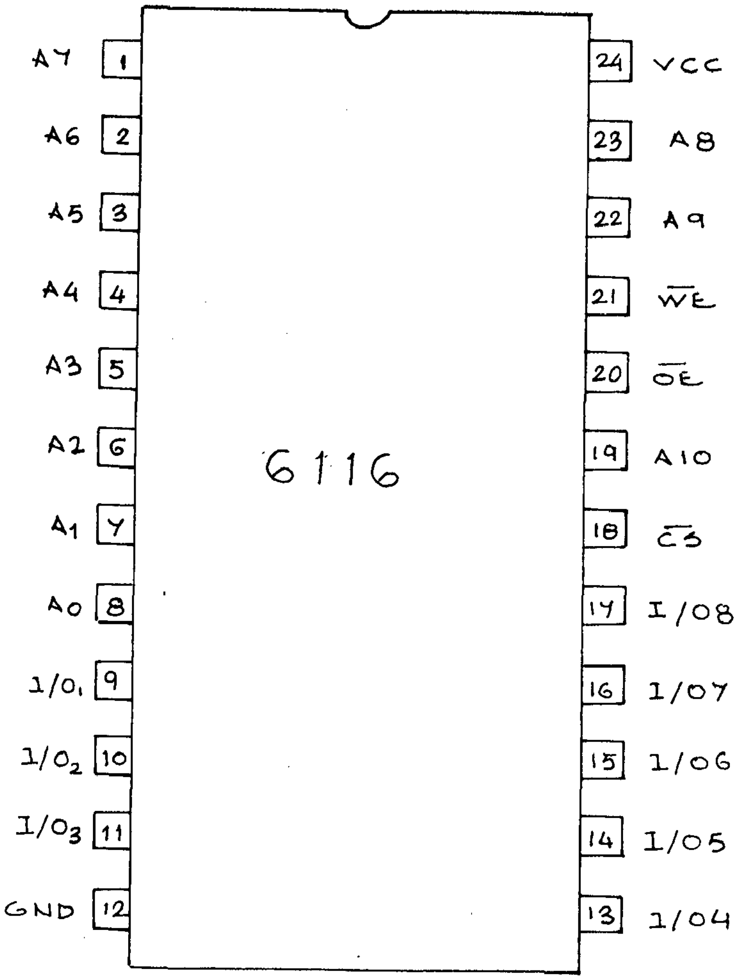


FIGURE 4: PIN CONNECTION DIAGRAM

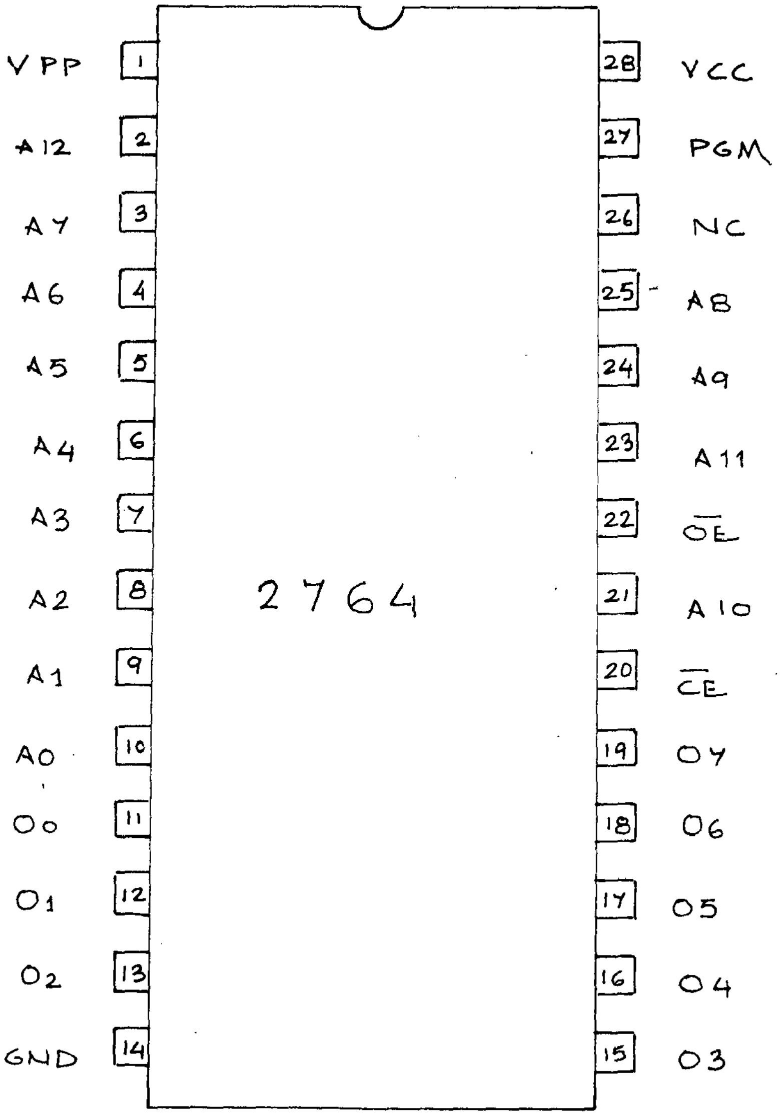


FIGURE 6 : PIN CONNECTION DIAGRAM

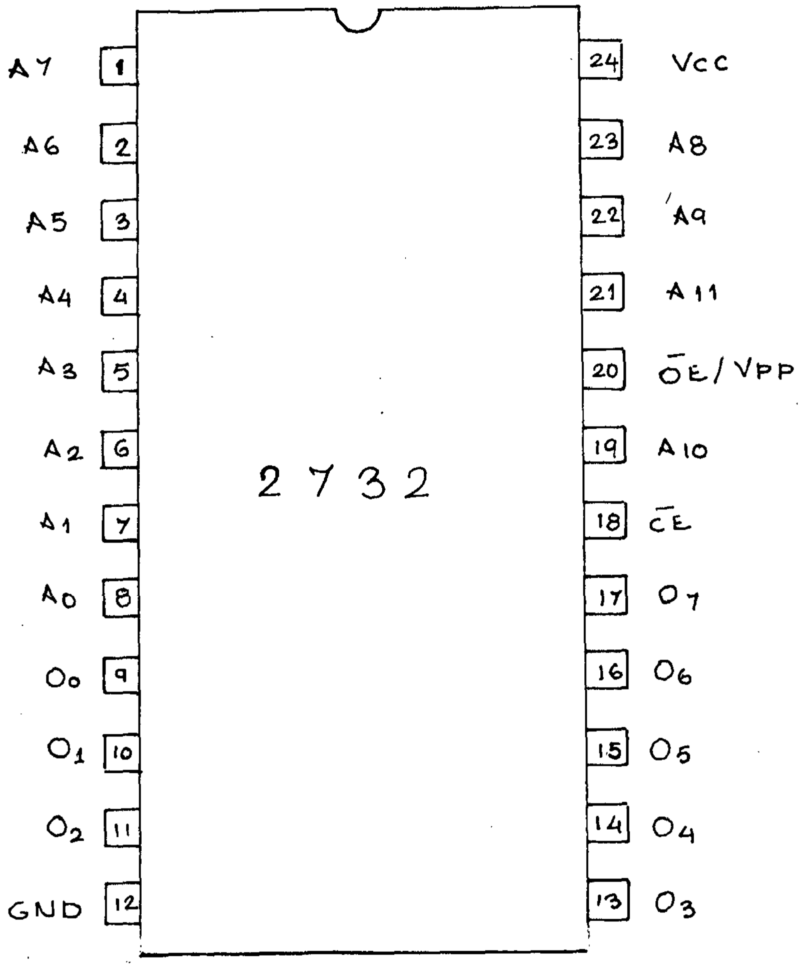


FIGURE 7

PIN CONNECTION DIAGRAM

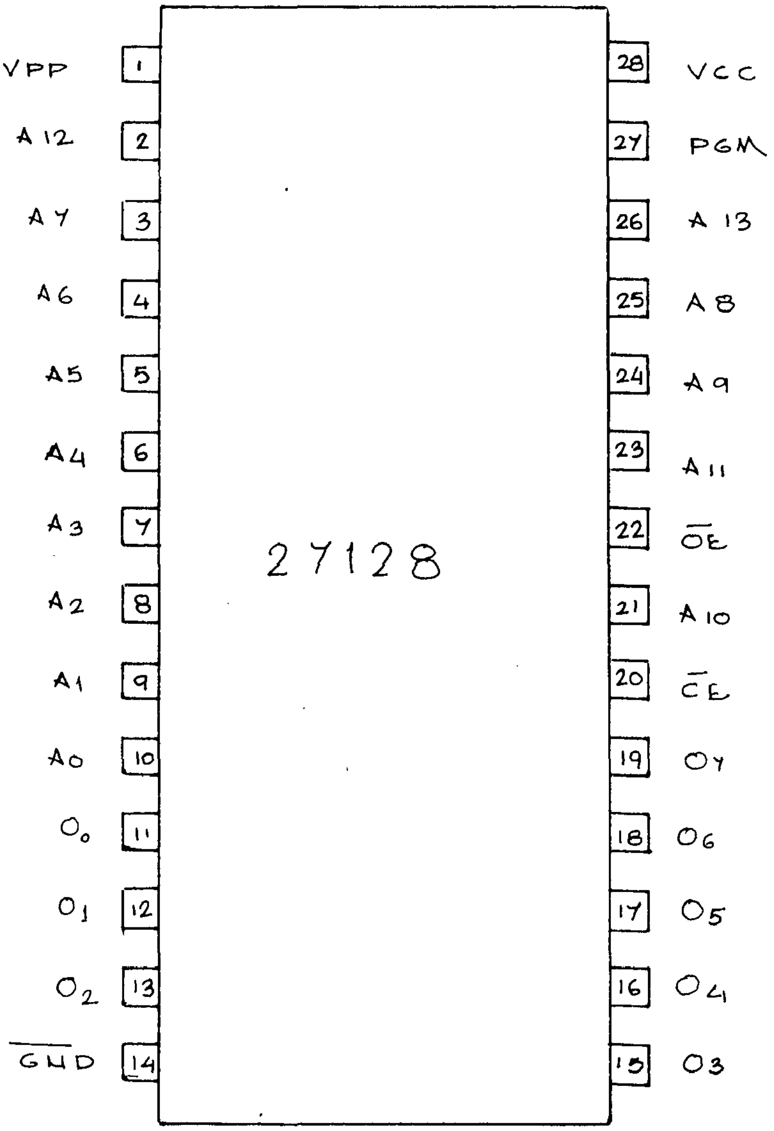


FIGURE 8 : PIN CONNECTION DIAGRAM

APPENDIX-C

FUNCTIONAL DESCRIPTION OF 8251

8251 is a Universal Synchronous/Asynchronous Receiver Transmitter for a wide range of intel microcomputers such as 8048, 8080, 8085, 8086 and 8088 Like other I/o devices its functional configuration is programmed by the system's software for maximum flexibility. Pin connection diagram is given in Fig. 9.1 [illegible]

RESET : A high on this input forces 8251 into an idle mode. The device will remain at idle until a new set of control words is written into 8251 A to program its functional definition.

CLK : CLK input is used to generate internal device timings frequency of the clock must be greater than 30 times the Receiver or Transmitter data bit rates.

$\overline{WR}$: Low on this input informs 8251 that CPU is writting data or control words to the 8251A.

$\overline{RD}$: A low on this input informs the 8251A that the CPU is reading data or status information from 8251. A

$C/\overline{D}$ (Control/Data): In conjunction with the $\overline{WR}$ and $\overline{RD}$ inputs this input informs the 8251A that the WORD on the data bus is a data character or a control word or status 1= CONTROL/STATUS , 0 = DATA.

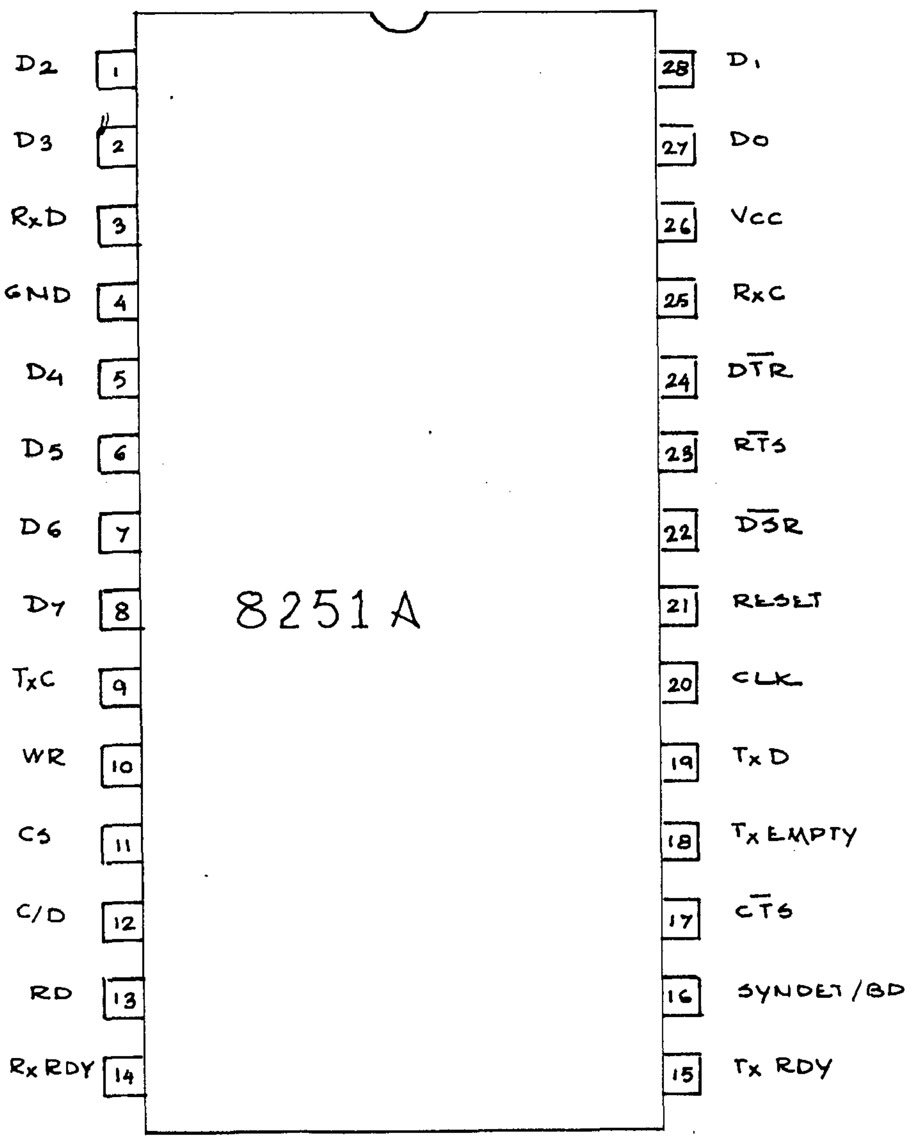

8251A
D2 1
D3 2
RxD 3
GND 4
D4 5
D5 6
D6 7
D7 8
TxC 9
WR 10
CS 11
C/D 12
RD 13
RxRDY 14
D1 28
D0 27
Vcc 26
RxC 25
DTR 24
RTS 23
DSR 22
RESET 21
CLK 20
TxD 19
TxEMPTY 18
CTS 17
SYNDET/BD 16
TxRDY 15

FIG. 9.1

$\overline{CS}$ (CHIP SELECT) : A low on this input selects the 8251A.

$\overline{DSR}$(DATA SET READY): The DSR input signal is a general purpose, 1 bit inverting input port, $\overline{DSR}$ input is normally used to test modem conditions.

DTR (Data Terminal Ready) : The $\overline{DTR}$ output signal is a general purpose, 1 bit inverting output port. It can be set low by programming the appropriate bit in the command instruction.

$\overline{RTS}$ (Request to Send) : is a general purpose, 1 bit inverting output port it can be set low by programming the appropriate bit in command register and is used fo? modem control.

$\overline{CTS}$ (clear tc send): A low on this ınput enables the 8251A to transmit serial data if the Tx Enable bit in the command byte is set to a one.

TRANSMITTER CONTROL:

TxRDY → (Transmitter Ready) This output signals the CPU that transmitter is ready to accept a data character. It can be checked by status read operation.

TXE (Transmitter Empty): When 8251 has no characters to send the TXE output goes HIGH to tell the CPU that transmitter is empty. It Remains HIGH when Transmitter is disabled.

$\overline{TXC}$ → (Transmitter Clock). It controls the rate at which the characters are to be transmitted In Asynchronous mode Band rate is a fraction of actual $\overline{TXE}$ and this fraction has to be selected by a portion of mode instruction. It can be 1, 1/16, 1/64, the

Falling end of $\overline{\text{TXC}}$ shifts the serial data out of the 8251.

RECEIVER CONTROL:

RXRDY (Receiver Ready) This output indicates that whether the 8251 A contains a character that is ready to be input to the CPU or not. In asynchronous mode to set RXRDY the Receiver must be enabled to sense a slart Bit and to transfer the complete assembled character to the Dat Output Register.

$\overline{\text{RXC}}$ (Receiver Clock) Receiver Clock Controls the rate at which characters are to be received. In asynchronous mode the Baud rate is the fraction of actual $\overline{\text{RXC}}$ frequency, this fraction factor has to be selected by a portion of mode instruction. It can be 1, 1/16 or 1/64 the $\overline{\text{RXC}}$.

SYNDET/BRKDET : This pin is used in synchronous mode and may be used as either input or output programmable through the control word, when used as an output the SYNDET pin will go high to indicate that 8251 has located the SYNC character in receive mode. When used as an input, a positive going signal will cause the 8251A to start assembling data characters on the rising edge of the next RxC. Once in SYNC the high input signal can be removed.

If used in double SYNC characters the SYNDET will go high in the middle of the last bit of the second SYNC character. SYNDET is automatically reset upon a status read operation.

BREAK (ASYNC MODE ONLY) :

This output will go high whenever the receiver remains low through two consecutive stop bit sequences (including start bit, data bits and parity bits. Break detect may also be read as status bit, it is reset only upon a master chip reset or Rx data returning to one state.

PROGRAMMING THE 8251 :

The complete functional defenition of the 8251A is programmed by the systems software. A set of control words must be sent out by the CPU to initialize the 8251A to support the desired communications format. These control words programme the BAUD RATE,CHARACTER LENGTH,NUMBER OF STOP BITS, SYNCHRONOUS or ASYNCHRONOUS OPERATION,EVEN/ODD/OFF PARITY etc. In Sync. mode options are also provided to select whether internal or external character synchronization. Once programmed, 8251 is ready to perform its communication functions. The TxRDY output is raised high to signal the CPU that 8251 is ready to receive a data character from the CPU. This output is Reset automatically when the CPU Writes a character into the 8251A.8251 receives the serial data from Modem or IO device. Upon receiving an entire character,the RxRDY output is raised high to signal the CPU that,the 8251 has a complete character ready for the CPU to fetch. RxRDY is Reset automatically upon the CPU data read operation.

The 8251 can not begin transmission until the Tx Enable (transmitter enable) bit is set in the command instruction and it has received a clear to send ($\overline{CTS}$) input. The TxD output will be held in the marking state upon Reset.

The Control words to be programmed are split into two formats:

1. Mode instruction
2. Command instruction

Mode instruction: 8251 can be used for either Asynchronous or Synchronous data communication. To understand how the mode instruction defines the functional operation of 8251A, the designer can best view the dévice as two seperate components, one synchronous and the other Asynchronous; sharing the same package. The format definition can be changed only after a master chip Reset. For explaination purpose the two format are seperately given in Fig. 9(d) and 9(e).

STATUS READ DEFINITION:

In data communication systems it is often necessary to examine the status of the active device to ascertain if errors have occurred or other conditions that require the processors attention. The 8251 A has facilities that allow

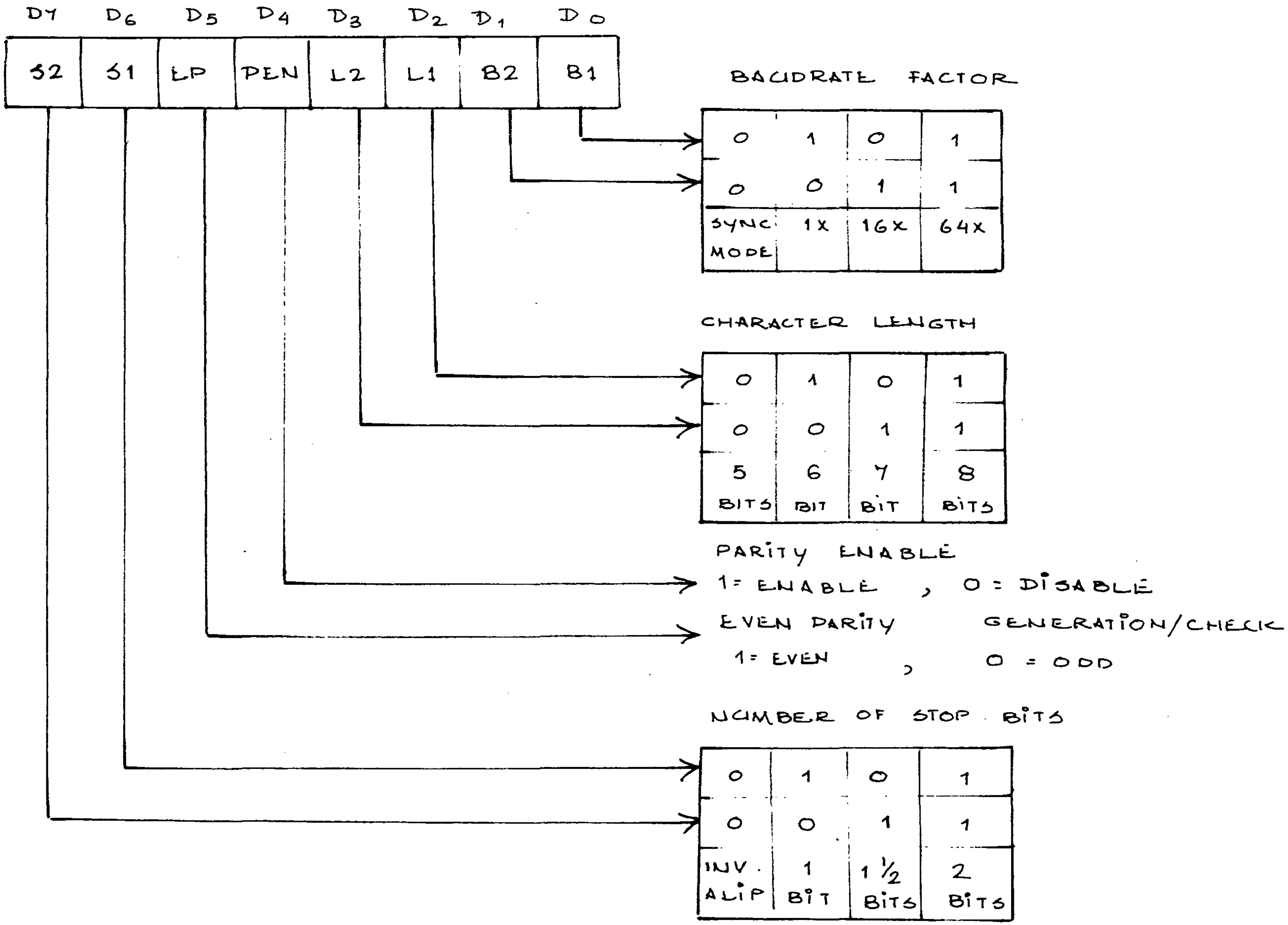


FIGURE 9 (c) : MODE INSTRUCTION FORMAT;
ASYNCHRONOS MODE

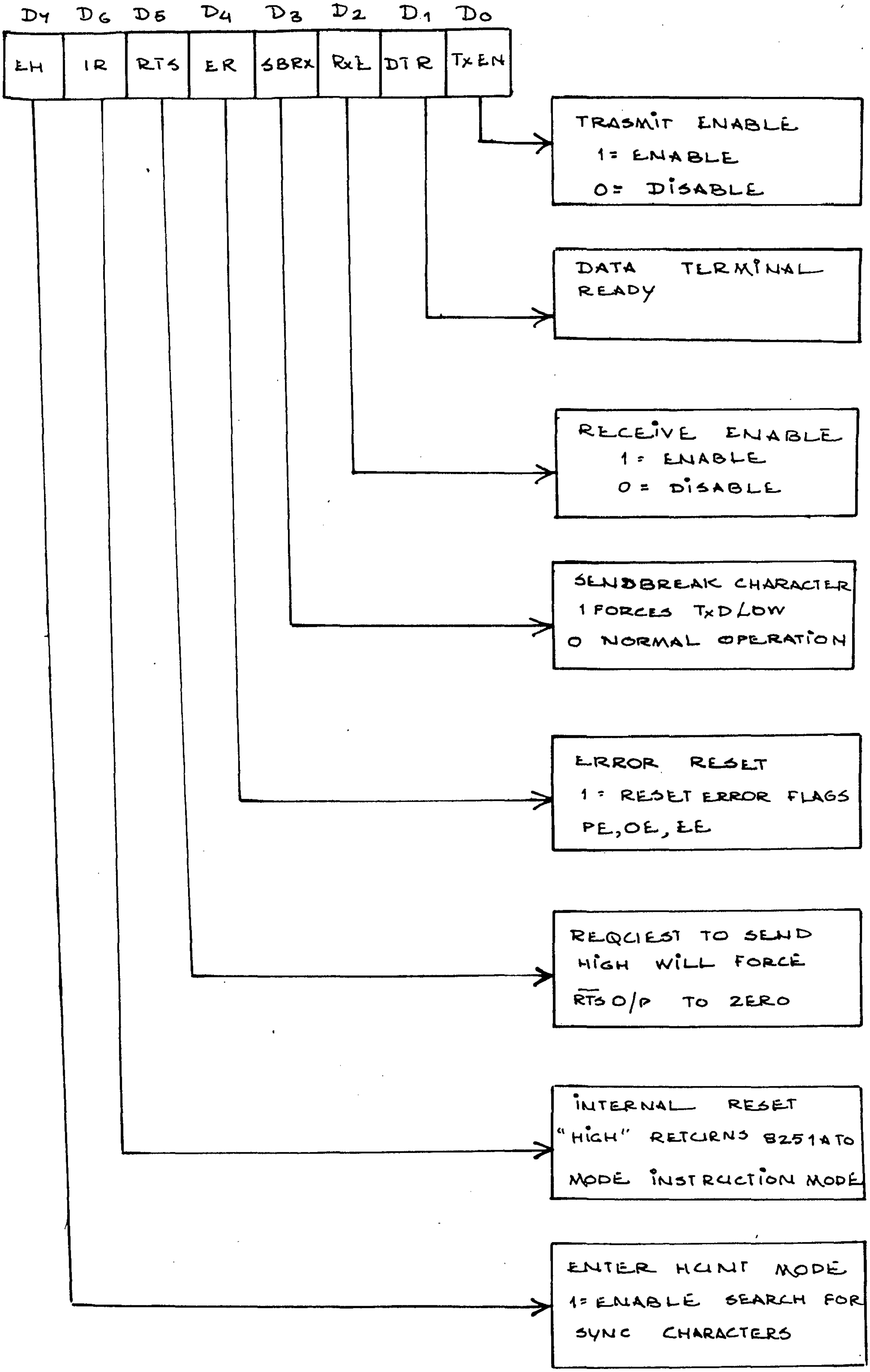
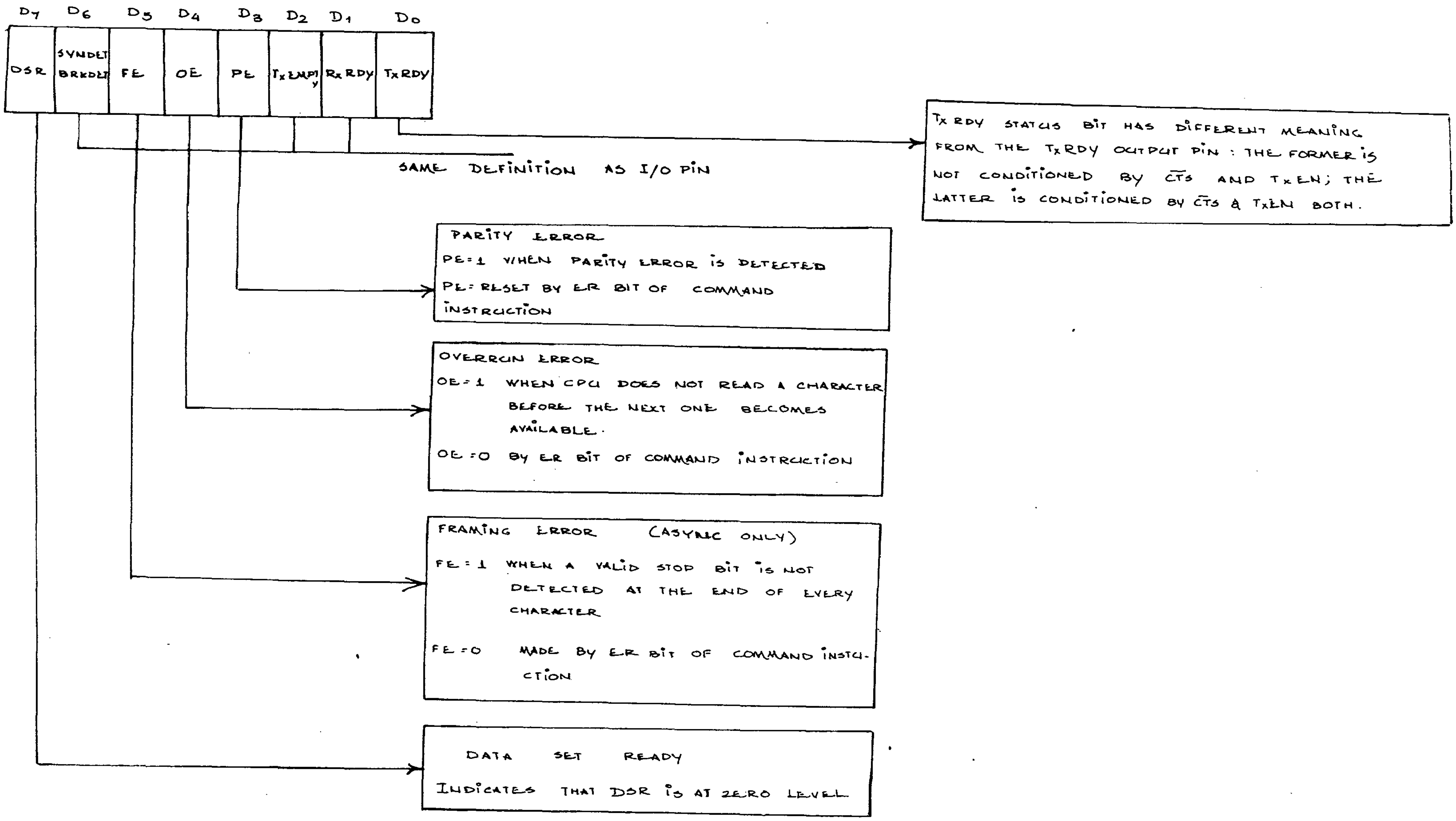


FIGURE 9(d): COMMAND INSTRUCTION FORMAT



TxRDY STATUS BIT HAS DIFFERENT MEANING FROM THE TxRDY OUTPUT PIN : THE FORMER IS NOT CONDITIONED BY $\overline{CTS}$ AND TxEN; THE LATTER IS CONDITIONED BY $\overline{CTS}$ & TxEN BOTH.

PARITY ERROR

PE=1 WHEN PARITY ERROR IS DETECTED

PE= RESET BY ER BIT OF COMMAND INSTRUCTION

OVERRUN ERROR

OE=1 WHEN CPU DOES NOT READ A CHARACTER BEFORE THE NEXT ONE BECOMES AVAILABLE.

OE=0 BY ER BIT OF COMMAND INSTRUCTION

FRAMING ERROR (ASYNC ONLY)

FE=1 WHEN A VALID STOP BIT IS NOT DETECTED AT THE END OF EVERY CHARACTER

FE=0 MADE BY ER BIT OF COMMAND INSTUCTION

DATA SET READY

INDICATES THAT DSR IS AT ZERO LEVEL

FIGURE 9(C) : STATUS READ FORMAT

the programmer to read the status of the device at any time during the functional operation (status update is inhibited during status read).

A normal read command is issued by CPU with $C/\overline{D}=1$ to accomplish thin function. Status read format is given in Fig. 9(e).

APPENDIX : D

74LS393 DUAL 4 STAGE (BIT) BINARY COUNTER:

Pin Connection diagram is given in Fig. 10.

It comprises of 8 master-slave Flip-Flops and additional gating to implement two individual 4 bit counters in a single package. Each counter has a CLEAR and a clock input. N bit binary counters can be implemented with each package providing the capability of divide by 256. 74LS393 has parallel outputs from each counter stage so that any submultiple of the input count frequency is available for system timing signals. Truth table and functional block diagram is given in Fig. 11.

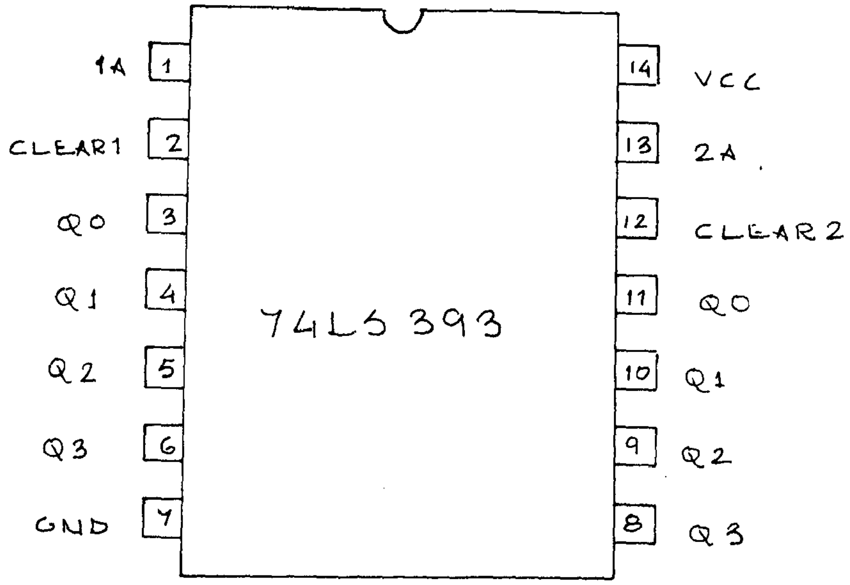


FIGURE 10: PIN CONNECTION DIAGRAM

CUNT	OUTPUT			
	QD	QC	QB	QA
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H
10	H	L	H	L
11	H	L	H	H
12	H	H	L	L
13	H	H	L	H
14	H	H	H	L
15	H	H	H	H

TRUTH TABLE FOR 74LS393

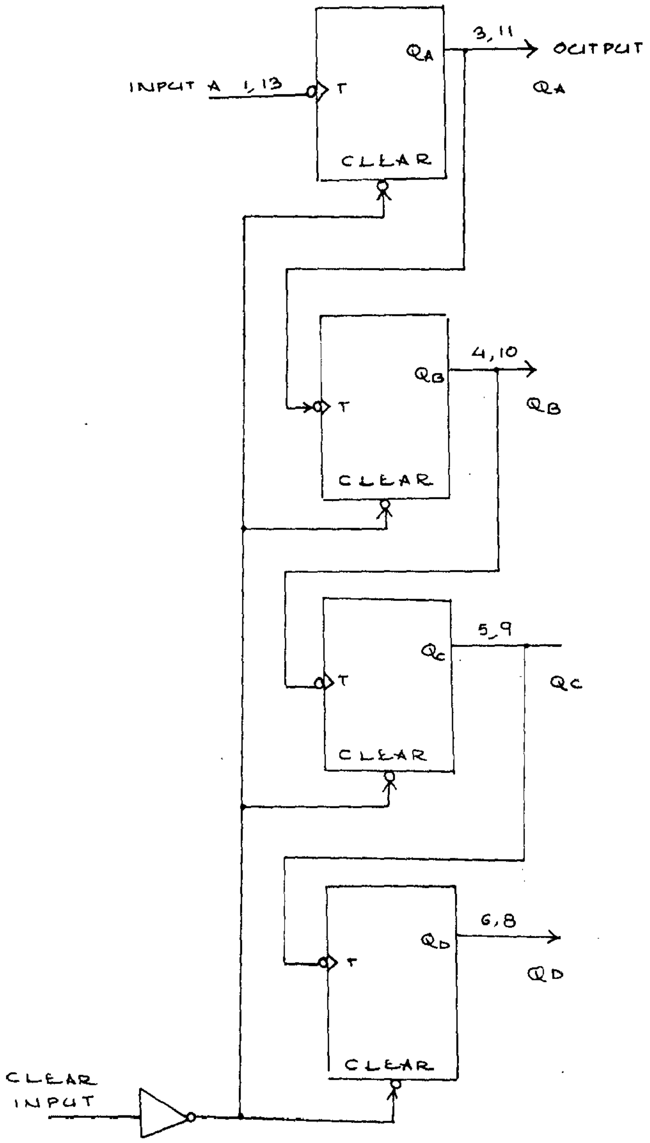


FIGURE - 11
FUNCTIONAL BLOCK DIAGRAM

EPPENDIX - E

X-R 1489 QUAD LINE RECEIVER

The XR 1489 is a monolithic quad line receiver specially designed for data bus interface. Each of the line receiver sections has adjustable hysteresis characterstics for improved noise rejection. The input and output levels of the ckt. are designed to provide direct interface between RS 232C data bus standard and the DTL or TTL logic levels. The XR 1489 quad line receiver along with its companion Ckt. XR 1488 quad line driver provides a complete interface between DTL and TTL logic levels and the RS 232C defined voltage and impedance levels. Pin connection diagram and functional block diagram is shown in Fig. 12.

<u>Features</u>	Direct Replacement of MC 1489A
	Current limited Output
	Compatible with DTL and TTL logic
	Meets EIA Standard RS 232C

APPLICATIONS:

1. Data Bus Interface
2. Micro-processor Interface
3. Remote terminal Interface
4. RS 232C interface.

Versions	XR 1489 AN	Ceramic	0-70°c
	XR 1489 AP	Plastic	0-70°C.

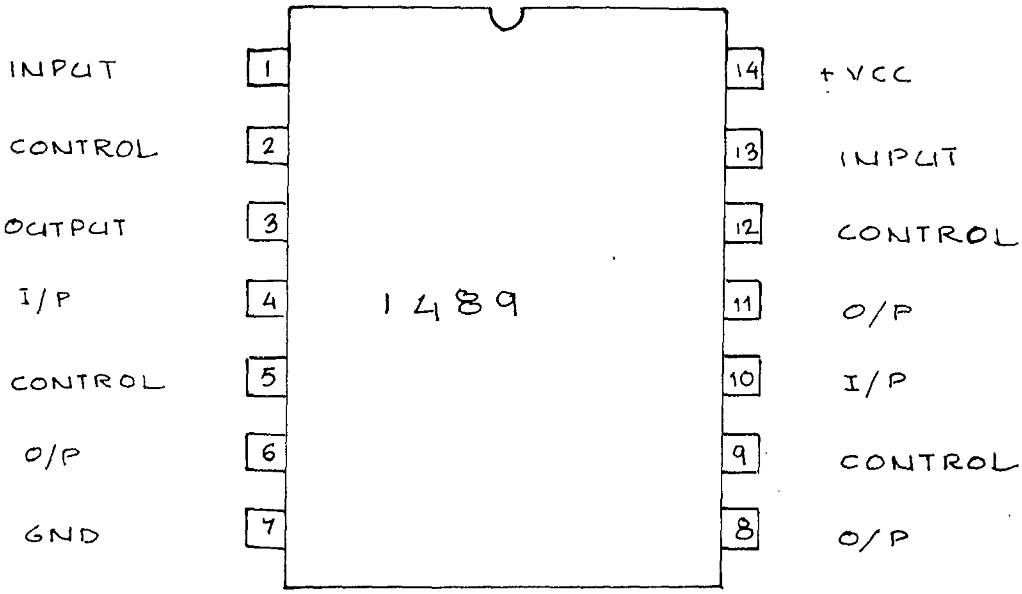


FIG. 12 (a)
PIN CONNECTION DIAGRAM

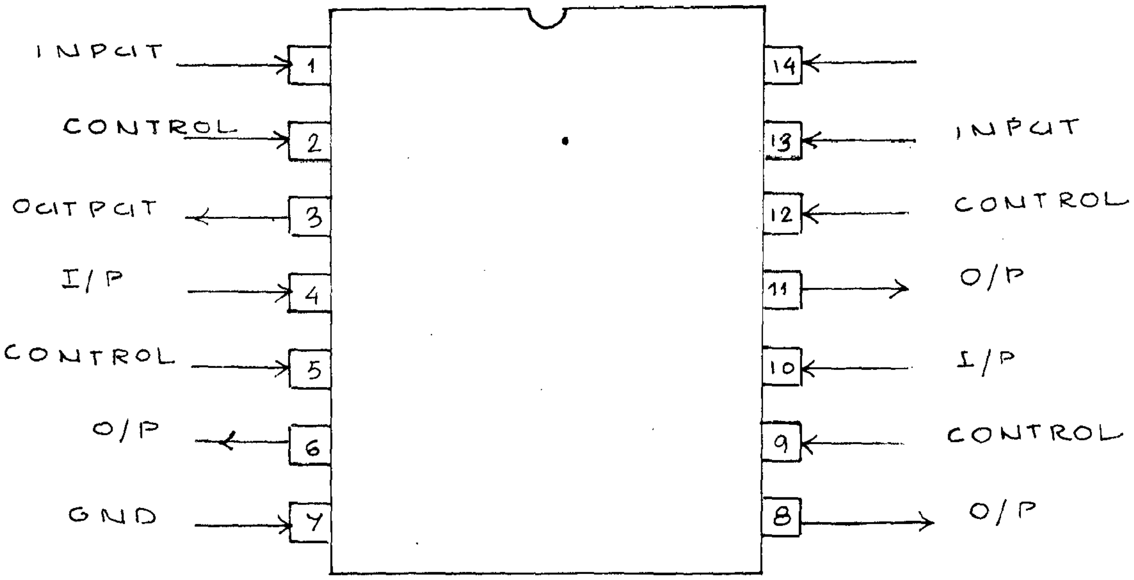


FIGURE 12(b) : FUNCTIONAL BLOCK DIAGRAM

APPENDIX-F

XR-1488 QUAD LINE DRIVER

XR 1488 is a monolithic quad line driver designed to interface data terminal equipment with data communications equipment in conformance with the specifications of EIA standard No. RS 232C. This extremely versatile integrated ckt can be used to perform a wide range of applications. The ckt features output current limiting circuitery and independent positive and negative power supply driving elements. Compatibility with all DTL and TTL families enhances the versality of the CKT.

The XR 1488 quad line driver along with its companion ckt the XR 1488 quad line receiver provides a complete interface system between DTL and TTL logic levels, and the RS 232C defined voltage and impedance level. The pin connection diagram is shown in Fig. 13(a).

FEATURES

Current limited output

Independent +ve Power Supply driving elements

Independent -ve Power Supply driving elements

Compatible with DTL and TTL logic familier

Data Terminal/Data Communication interface.

Conforms to EIA Standard No. RS 232C.

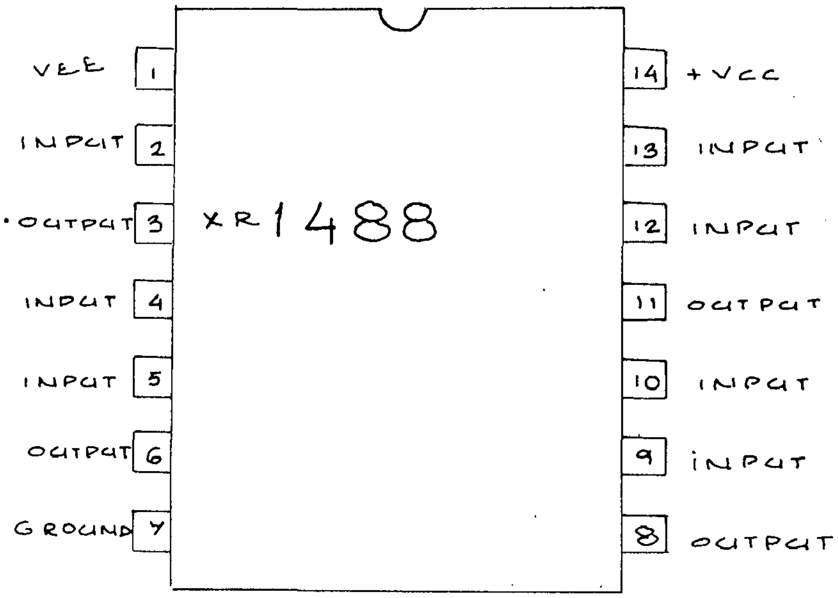


FIGURE 13(a): PIN CONNECTION DIAGRAM

Application : RS 232C Interface.

VERSIONS

XR 1488N	Ceramic	0 to 70°C
XR 1488P	Plastic	0°C to + 70°C.

APPENDIX - G

SALIENT FEATURES OF 8086

Intel 8086 introduced in June 1978 is the first of the high performance generation of 16 bit microprocessors. It is implemented in N channel depletion load silicon gate technology (HMOS) and packaged in 40 pin DIP package. It can address 1M byte of memory directly and can be used in multiprocessing system. The detail pin out and architecture is discussed in following lines:

(a) PIN OUT: Pin out [illegible] of 8086 is shown in Fig. 13.1.

like other microprocessors it has.

1. Address Lines
2. Data lines
3. Control and Status lines
4. Power and tuning lines.

The pin signals of 8086 are divided into 3 catagories.

1. Common Signals
2. Maximum mode Signals
3. Minimum mode signals.

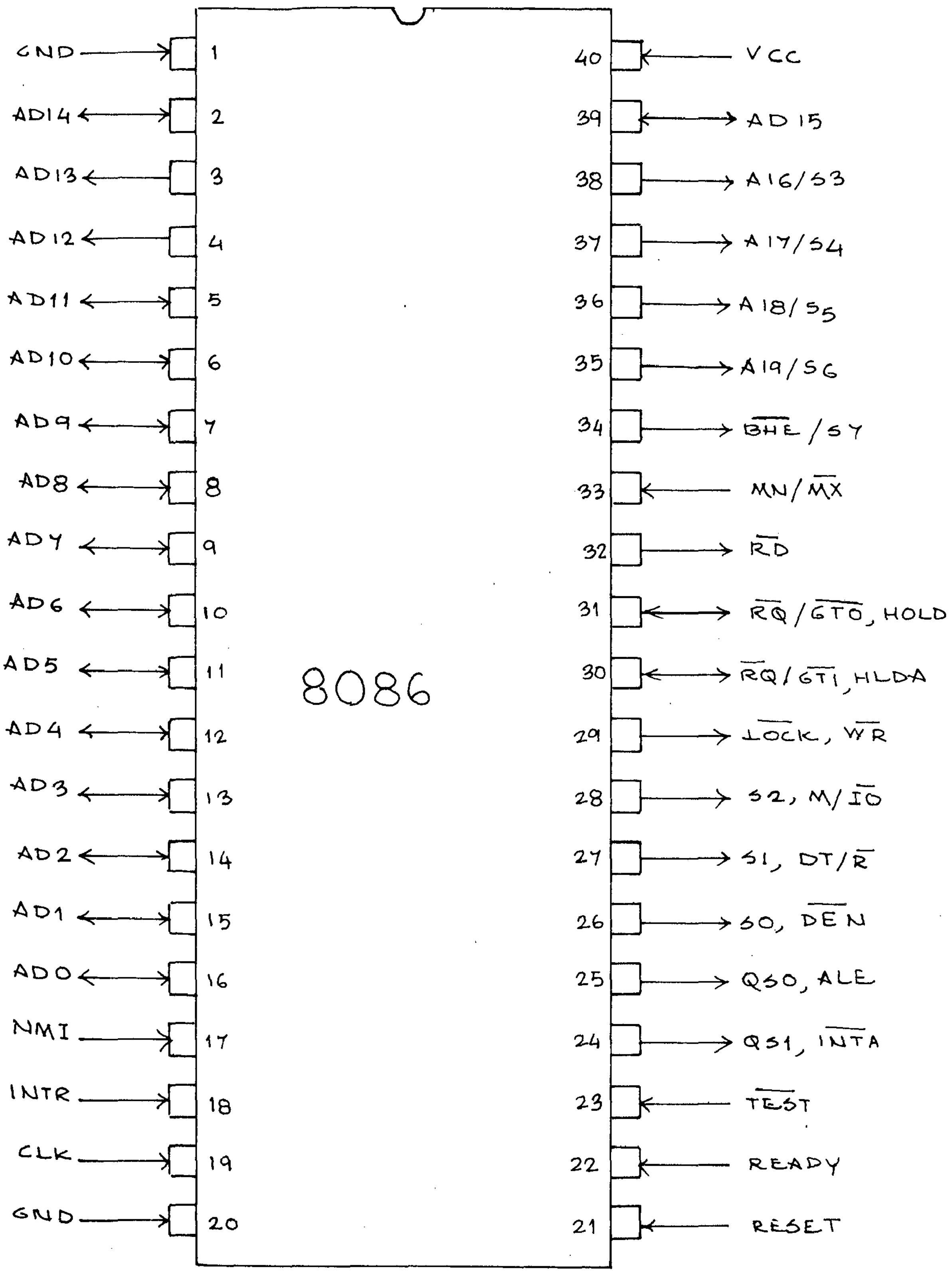


FIGURE 13·1: PIN CONNECTION DIAGRAM

(b) COMMON SIGNALS

(i) AD14-ADO at Pin No. 2-16 and AD15 at Pin No. 39:

Time multiplexed Address/data bus.

Memory/IO address (T_1) and data (T_2,T_3,T_w,T_4) bus.

(ii) A_{19}/S_6- A_{16}/S_3 at Pin No. 35-38 : Time multiplexed Address/Status lines.

During T_1 used as address lines for memory operations, LOW during I/O operations, During T_2,T_3,T_4 and T_w as status information is available on these lines. S_3 and S_4 indicate which of the segment register is used (to construct physical address) as follows:

S_4	0	0	1	1
S_3	0	1	0	1
SEG REG	ES	SS	CS	DS

S_5 reflects state of interrupt enable flag, S_6 is LOW except during HOLD ACKNOWLEDGE Clock periods.

(iii) $\overline{\text{BHE}}/\overline{S_7}$ at Pin no. 34 (OUT): BHE implies Bus High Enable. During the execution of READ,WRITE and Interrupt Acknowledge cycle this line acts as $\overline{\text{BHE}}$. $\overline{\text{BHE}}$ and is held LOW during first clock period of the instruction cycle. This signal is also used in conjunction with ADO line for select logic for memory banks. During the subsequent clock periods this pin maintains the output of first

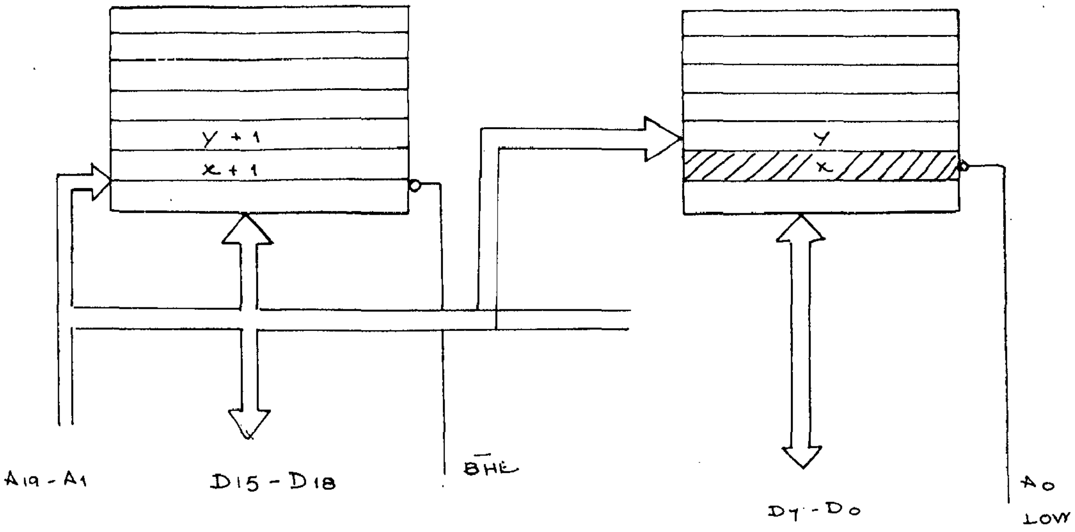


FIGURE 13·3 (a): EVEN ADDRESS BYTE TRANSFER

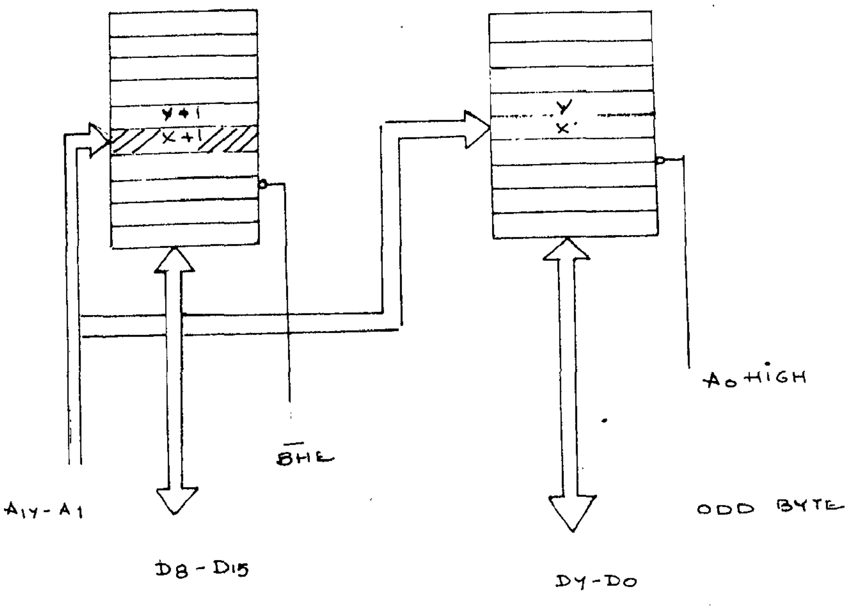


FIGURE 13·3(b): ODD ADDRESS BYTE TRANSFER clock period. Selection of memory banks using $\overline{\text{BHE}}$ is shown in Fig. 13.3(a) and 13.3(b).

S_7 is spare status line and its contents are undefined.

(iv) MN/MX at Pin no. 33 (IN):

Indicates the system configuration

if grounded - Maximum mode.

if +5V - Minimum mode.

(v) $\overline{\text{RD}}$ at Pin no. 32 (OUT) : Indicate that processor is performing an READ (MEMORY/IO) operation.

(vi) READY at Pin no. 22(IN): Control Signal used by the selected memory or IO device to indicate whether device is ready for data transfer operation or not.

(vii) $\overline{\text{TEST}}$ at Pin no. 23 (IN): Examined by the processor to come out of the idle state.

(viii) INTR at Pin no. 18 (IN): Maskable Interrupt Request line, speciality is that it can be masked by software.

(ix) NMI at Pin no. 17 (IN): Non maskable interrupt line and had higher priority over the INTR. It causes type 2 interrupt (predefined).

(x) RESET at Pin no. 21 (IN): When ACTIVE HIGH causes the processor to immediately terminates its present activity and starts execution from FFFF0.

(xi) CLOCK at Pin no. 19 (IN): It provides basic timings for processor. The maximum mode and minimum mode signals with the brief functional description is given in table 13.1.

TABLE 13.1

	MAXIMUM MODE		MINIMUM MODE	
PIN NO	SIGNAL	FUNCTION	SIGNAL	FUNCTION
26	$\overline{\text{SO}}$	Provide status information as per given in table 13.2(a)	$\overline{\text{DEN}}$	Provided as an output enable for data bus transceiver.
27	$\overline{\text{SI}}$		DT/$\overline{\text{R}}$	Data Transmit/ Receive. It is used to control the direction of data flow through the transceiver.
28	$\overline{\text{S2}}$		M/$\overline{\text{IO}}$	Implies Memory/IO accessing.
31,30	RQ/$\overline{\text{GTD}}$ RQ/$\overline{\text{GT1}}$	Request grant lines used in multiprocessing systems, RQ/$\overline{\text{GTO}}$ has higher priority than RQ/$\overline{\text{GT1}}$	HOLD, HOLDA	Both used in DMA operation HOLD is INPUT HOLDA is OUTPUT
25,24	Q50, And Q51	Imply queue status as per table 13.2(b)	ALE and $\overline{\text{INTA}}$	ALE is used to latch Higher order. 12 bits of address. INTA is used as write strobe for interrupt acknowledge Cycle.
29	$\overline{\text{LOCK}}$	It imply in multi processing system that system bus is locked.		Indicate that the processor is performing a ~~memory/~~ IO write cycle.

TABLE 13.2(a)

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	IMPLICATION
0	0	0	INTERRUPT ACKNOWLEDGE BUS CYCLE
0	0	1	I/O READ
0	1	0	I/O WRITE
0	1	1	HALT
1	0	0	INSTRUCTION FETCH
1	0	1	MEMORY READ
1	1	0	MEMORY WRITE
1	1	1	INACTIVE

TABLE 13.2(b)

QSO	QS1	IMPLICATION
0	0	NO OPERATION
0	1	THE FIRST BYTE OF INSTR IS BEING EXECUTED
1	0	THE QUE IS BEING EMPTIED
1	1	A SUBSEQUENT INSTR.BYTE IS BEING TAKEN FROM THE QUE

1.4 ARCHITECTURE:

It is divided into two sections as shown in Fig. 13.4.

1. BIU (Bus interface unit)
2. EU (Execution unit)

The execution unit executes the instructions. The bus interface unit fetches the instructions, reads operands, and writes results. The two units operate independently and in most cases overlap instruction fetch with execution, therefore the instruction fetch time is essentially eliminated. A 16-bit ALU in EU maintains the CPU status and control flags and manipulates the general registers and instruction operands. Register and data path in EU are 16 bits wide for faster internal transfers. The EU has no connection with system bus, but obtains its instructions from a queue maintained by the BIU. When an instruction requires accesss to memory or I/O the EU requests the BIU to obtain or store the data. The EU only manipulates 16 bit addresses but the BIU can perform address relocation giving the EU access to a full megabyte of memory space. The BIU performs all Bus operations for the EU. Data are transfered between CPU and peripheral devices when sc demanded by EU.

When the EU is executing instructions the BIU is fetching more instructions from memory. The instructions are

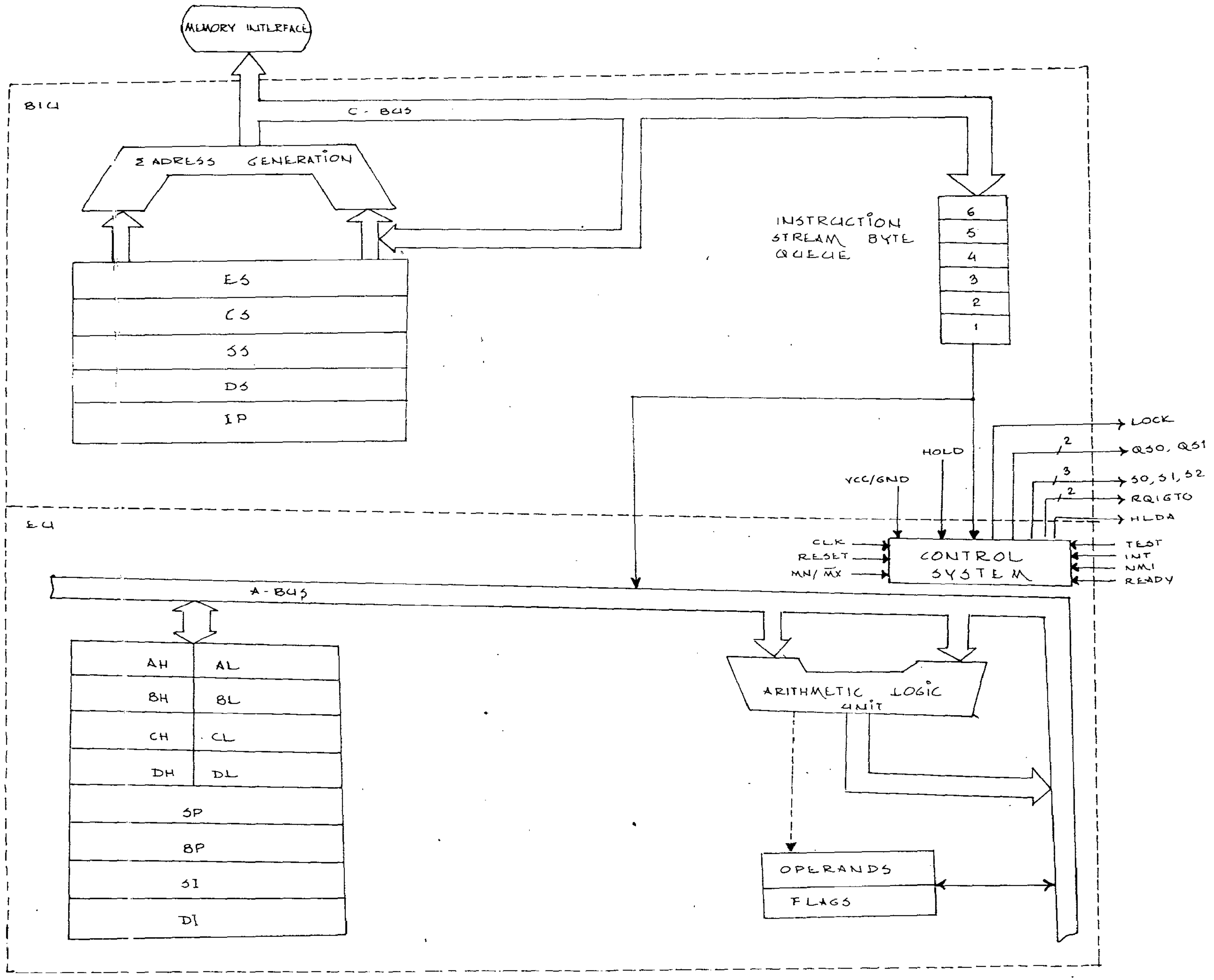

MEMORY INTERFACE
BIU
C - BUS
Σ ADRESS GENERATION
ES
CS
SS
DS
IP
INSTRUCTION STREAM BYTE QUEUE
6
5
4
3
2
1
HOLD
VCC/GND
CLK
RESET
MN/$\overline{MX}$
CONTROL SYSTEM
LOCK
2
QS0, QS1
3
S0, S1, S2
2
RQ/GT0
HLDA
TEST
INT
NMI
READY
EU
A - BUS
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
SI
DI
ARITHMETIC LOGIC UNIT
OPERANDS
FLAGS

FIG 13.4

stored in internal RAM array called the instruction stream queue, the length is of 6 Byte. The queue allows the BIU to keep the EU supplied with prefetched instructions under most conditions, without tying up the system bus. Under most circumstances, the queue contains at least one byte of the sequence of instructions so the EU does not have to wait for instructions to be fetched. Bytes in the queue are of the next logical instruction if execution proceeds serially. If an instruction transfers control to another location, the BIU will reset the queue and begin refilling after passing the new instruction to the EU.

There are eight 16 bit registers in EU. These registers are divided into two groups of four each:

1. Data registers
2. Pointer an index registers.

Each data register can be used as two 8 bit registers or a one 16 bit register. The other CPU registers are always acessed as 16 bit units. There are two pointer registers the stack pointer SP and Base pointer (BP). The SP contains the current stack address up to which it is full. BP is used in addressing memory. The two index registers, the source index (SI) and destination index (DI) are used in indexed addressing and also used in string operations.

The BIU contains four 16 bit segment registers and one 16 bit instruction pointer (IP), which is equivalent to PC. The 8086 megabyte memory space is divided in 4 segments of 64K bytes each Code segment (CS) register points to current Code segment from which instructions are fetched. The effective address of an instruction in memory is obtained by adding the contents of CS to the contents of IP in a particular way. The stack segment (SS) register points to the current stack segment i.e. Stack operations occur in this segment of memory. The effective address is obtained by adding the contents of SP to contents of SS. The DS register points towards current data segment where program variable are usually kept. Extra segment (ES) register points towards segment where data are typically stored.

8086 has 16 bit Flag register out of which only 9 bits are used. Actual format is shown in Fig. 13.6.

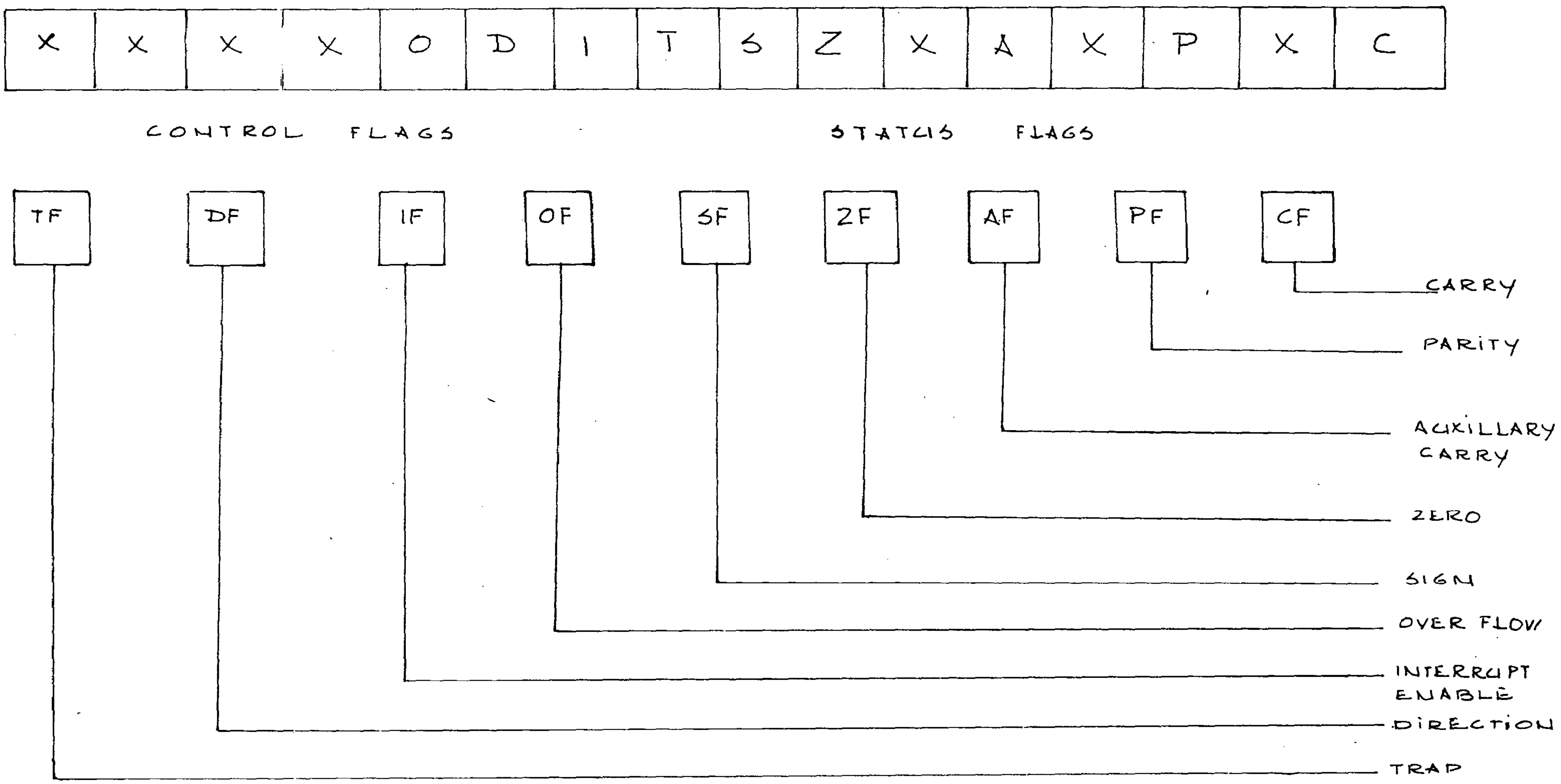

X X X X O D I T S Z X A X P X C
CONTROL FLAGS
STATUS FLAGS
TF
DF
IF
OF
SF
ZF
AF
PF
CF
CARRY
PARITY
AUXILLARY CARRY
ZERO
SIGN
OVER FLOW
INTERRUPT ENABLE
DIRECTION
TRAP

FIGURE 13·6

Implied uses of the registers are listed in table 13.1(c)

Implied uses of the Flags are listed in Table 13.1 (d).

TABLE 13.1(c)

REGISTER	OPERATIONS
AX	WORD MULTIPLY, WORD DIVDE,WORD I/O
AL	BYTE MULTIPLY, BYTE DIVIDE,BYTE I/O, TRANSLATE,DECIMAL ARITHMETIC
AH	BYTE MULTIPLY, BYTE DIVIDE
BX	TRANSLATE
CX	STRING OPERATIONS, LOOPS
CL	VARIABLE SHIPT AND ROTATE
DX	WORDMULTIPLY,WORD DIVIDE,INDIRECT I/O
SP	STACK OPERATIONS
SI	STRING OPERATIONS
DI	STRING OPERATIONS
FL	NOT GENERAL PURPOSE REGISTER

TABLE 13.1(d)

FLAG	FUNCTIONAL DESCRIPTION
AF	If the auxillary carry flag is set there has been a carry out of the low nibble into the high nibble or a borrow from high nibble into the low nibble of an 8 bit no. This flag is used by decimal arithmetic instructions.
CF	Is set if there has been a carry out of or a borrow into the high order bit of the result (8 bit or 16 bit). This flag is used by the instructions that add and subtract multibyte number.
OF	Is set if an arithmetic overflow has occured, that is, a significant digit has been lost because of the size of the result exceeds the capacity of its destination location.
SF	Is set if the higher order bit of the result is a 1, hence indicates the sign of the result.
PF	Can be used to check for data transmission errors.
ZF	Is set if the result of the operation is zero.
DF	It set causes autodecrement of SI/DI in string primitive instructions. If rest causes autoincrement of SI/DI in string primitive instructions
IF	It set allows the CPU to recognise external interrupt requests. If reset disables the external interrupts.
TF	Setting TF puts the processor into single step mode for debugging.

1.5 MEMORY:

1.5.1 STORAGE ORGANIZATION:

The 8086 can address up to 1M byte or 512 K words of memory directly, logically the memory is organized as a sequence of 2^{20} bytes but physically it is organized in two banks each of 512K bytes as shown in Fig. 13.8. One bank is connected to the lower half of the sixteen bit data bus (D_7-D_0) and contains even addressed bytes. The other bank is connected to the upper half of the sixteen bit data bus (D_{15}-D_8) and contains odd addressed bytes. A specific byte within each bank is selected by address lines A_{19}-A_1. The most significant address bit ADo and output control signal $\overline{BHE}$ are used to select the appropriate bytes to be read from or written into the memory.

Table 13.1(e) describes the implications of different combinations of $\overline{BHE}$ and A_0.

TABLE 13.1(e)

BHE	AO	IMPLICATION
0	0	One 16 bit word
0	1	One byte from/to odd address
1	0	One byte from/to even address
1	1	None

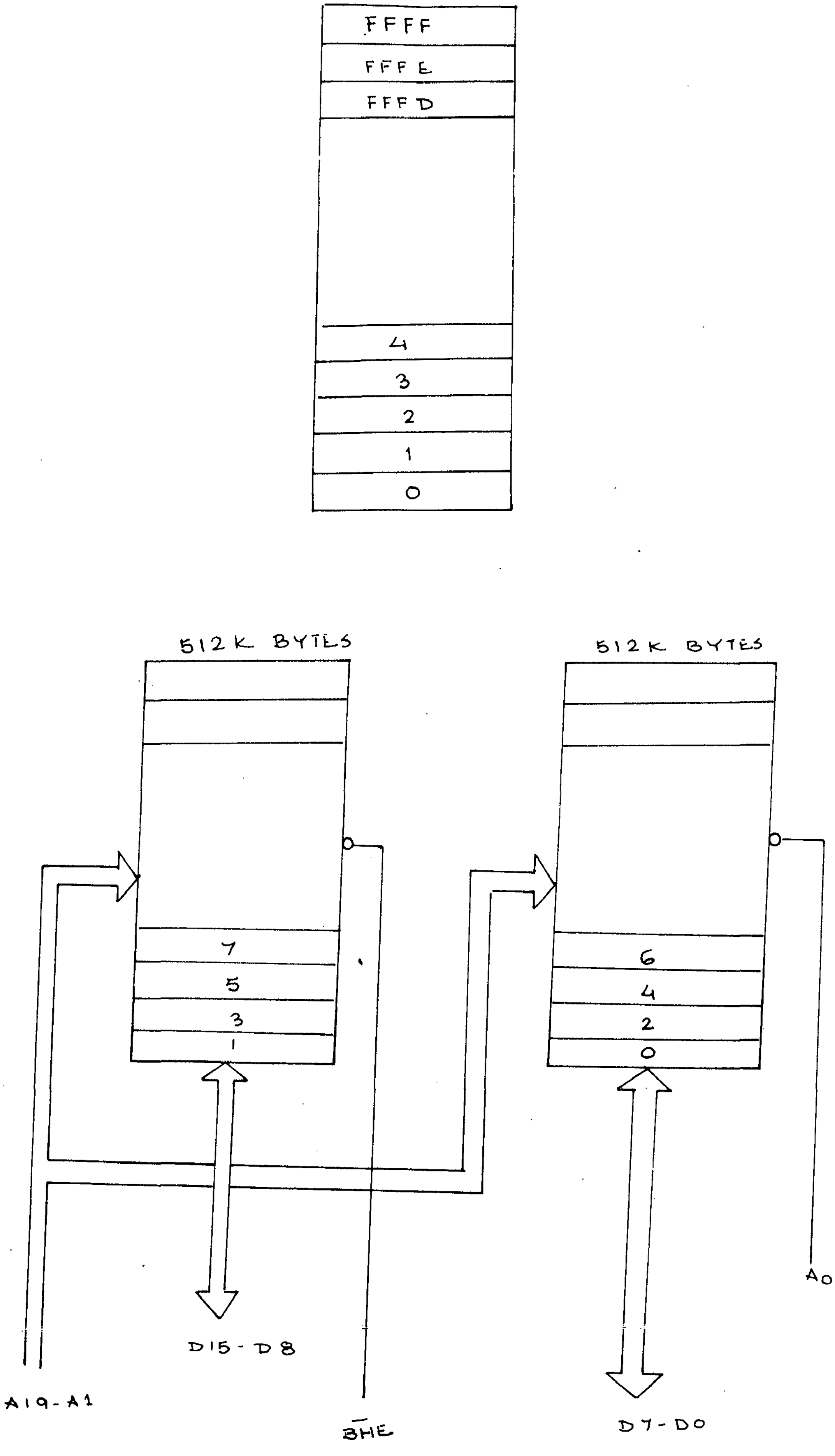


FIGURE 13·8: MEMORY ORGANIZATION

1.52 SEGMENTATION :

1M bytes of memory is divided into 4 segments namely Code Segment, Data segment, Stack Segment, and Extra segment. A segment is a logical unit of memory space that may be upto 64 K bytes long. Each segment is made up of contagious memory locations and these segment may be fully overlaped, partially overlapped or totally independent.

1.53 DYNAMICALLY RELOCATABLE CODE :

The segmented memory structure of 8086 makes it possible to write programs that are position independent or dynamically relocatable which also allows multistacking.

1.54 DEDICATED AND RESERVED MEMORY LOCATIONS:

Two areas in extream low and high memory are dedicated to specific processor function or are reserved by intel corporation for use by intel hardware and software products, the locations are OH to 7FH (128) bytes and FFFFO to FFFFF (16 bytes). These areas are used for interrupt processing and system RESET processing.

1.6 I/O SPACE:

The 8086 I/O space can accomodate up to 64K 8 bit ports or up to 32 K 16 bit ports. I/O space is not segmented.

1.61 RESTRICTED I/O LOCATIONS:

Locations F8H through FFH in the I/O space are reserved by Intel Corporation for use by future intel hardware and software products.

INSTRUCTION SET

Instruction is an basic operation which a μp can perform and set of such basic instruction is called as instruction set. 8086 provides a very wide instruction set consisting of 154 basic instructions. It is not easy to tell exact variations, one could realize it while going through the instruction set.

According to the function the instructions perform, the 8086 instructions can be grouped in following ways:

1. Data movement instructions
2. Arithmetic instructions
3. Logical instructions
4. String Primitive Instructions
5. Program Counter Control Instructions
6. Processor Control Instructions
7. I/O Instructions.
8. Rotate and Shift/instructions.

The instruction set briefly listed in Table g-1.

TABLE 9-1

Mnemonic and Description	Instruction Code			
Data Transfer				
MOV = Move:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory to/from register	1 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w 1
Immediate to register	1 0 1 1 w reg	data	data if w 1	
Memory to accumulator	1 0 1 0 0 0 0 w	addr-low	addr-high	
Accumulator to memory	1 0 1 0 0 0 1 w	addr-low	addr-high	
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 reg r/m		
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 reg r/m		
PUSH = Push:				
Register/memory	1 1 1 1 1 1 1 1	mod 1 1 0 r/m		
Register	0 1 0 1 0 reg			
Segment register	0 0 0 reg 1 1 0			
POP = Pop:				
Register/memory	1 0 0 0 1 1 1 1	mod 0 0 0 r/m		
Register	0 1 0 1 1 reg			
Segment register	0 0 0 reg 1 1 1			
XCHG = Exchange:				
Register/memory with register	1 0 0 0 0 1 1 w	mod reg r/m		
Register with accumulator	1 0 0 1 0 reg			
IN = Input				
Fixed port	1 1 1 0 0 1 0 w	port		
Variable port	1 1 1 0 1 1 0 w			
OUT = Output				
Fixed port	1 1 1 0 0 1 1 w	port		
Variable port	1 1 1 0 1 1 1 w			
XLAT=Translate byte to AL	1 1 0 1 0 1 1 1			
LEA=Load EA to register	1 0 0 0 1 1 0 1	mod reg r/m		
LDS=Load pointer to DS	1 1 0 0 0 1 0 1	mod reg r/m		
LES=Load pointer to ES	1 1 0 0 0 1 0 0	mod reg r/m		
LAHF=Load AH with flags	1 0 0 1 1 1 1 1			
SAHF=Store AH into flags	1 0 0 1 1 1 1 0			
PUSHF=Push flags	1 0 0 1 1 1 0 0			
POPF=Pop flags	1 0 0 1 1 1 0 1			
Arithmetic				
ADD = Add:				
Reg./memory with register to either	0 0 0 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s w=01
Immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w 1	
ADC = Add with carry:				
Reg./memory with register to either	0 0 0 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s w=01
Immediate to accumulator	0 0 0 1 0 1 0 w	data	data if w=1	
INC = Increment:				
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m		
Register	0 1 0 0 0 reg			
AAA=ASCII adjust for add	0 0 1 1 0 1 1 1			
DAA=Decimal adjust for add	0 0 1 0 0 1 1 1			
SUB = Subtract:				
Reg./memory and register to either	0 0 1 0 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s w=01
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w=1	
SBB = Subtract with borrow				
Reg./memory and register to either	0 0 0 1 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s w=01
Immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w=1	
DEC = Decrement:				
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m		
Register	0'1 0 0 1 reg			
NEG=Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m		

Mnemonic and Description	Instruction Code			
CMP = Compare:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory and register	0 0 1 1 1 0 d w	mod reg r/m		
Immediate with register/memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s;w=01
Immediate with accumulator	0 0 1 1 1 1 0 w	data	data if w=1	
AAS=ASCII adjust for subtract	0 0 1 1 1 1 1 1			
DAS=Decimal adjust for subtract	0 0 1 0 1 1 1 1			
MUL=Multiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m		
IMUL=Integer multiply (signed)	1 1 1 1 0 1 1 w	mod 1 0 1 r/m		
AAM=ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0		
DIV=Divide (unsigned)	1 1 1 1 0 1 1 w	mod 1 1 0 r/m		
IDIV Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m		
AAD ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0		
CBW Convert byte to word	1 0 0 1 1 0 0 0			
CWD Convert word to double word	1 0 0 1 1 0 0 1			
Logic				
NOT Invert	1 1 1 1 0 1 1 w	mod 0 1 0 r/m		
SHL/SAL Shift logical/arithmetic left	1 1 0 1 0 0 v w	mod 1 0 0 r/m		
SHR=Shift logical right	1 1 0 1 0 0 v w	mod 1 0 1 r/m		
SAR=Shift arithmetic right	1 1 0 1 0 0 v w	mod 1 1 1 r/m		
ROL=Rotate left	1 1 0 1 0 0 v w	mod 0 0 0 r/m		
ROR=Rotate right	1 1 0 1 0 0 v w	mod 0 0 1 r/m		
RCL Rotate through carry flag left	1 1 0 1 0 0 v w	mod 0 1 0 r/m		
RCR Rotate through carry right	1 1 0 1 0 0 v w	mod 0 1 1 r/m		
AND = And:				
Reg./memory and register to either	0 0 1 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w=1
Immediate to accumulator	0 0 1 0 0 1 0 w	data	data if w=1	
TEST = And function to flags, no result:				
Register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m		
Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w=1
Immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w=1	
OR = Or:				
Reg./memory and register to either	0 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w=1
Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w=1	
XOR = Exclusive or:				
Reg./memory and register to either	0 0 1 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w=1
Immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w=1	
String Manipulation				
REP=Repeat	1 1 1 1 0 0 1 z			
MOVS = Move byte/word	1 0 1 0 0 1 0 w			
CMPS = Compare byte/word	1 0 1 0 0 1 1 w			
SCAS = Scan byte/word	1 0 1 0 1 1 1 w			
LODS = Load byte/wd to AL/AX	1 0 1 0 1 1 0 w			
STOS = Stor byte/wd frm AL/A	1 0 1 0 1 0 1 w			
Control Transfer				
CALL = Call:				
Direct within segment	1 1 1 0 1 0 0 0	disp-low	disp-high	
Indirect within segment	1 1 1 1 1 1 1 1	mod 0 1 0 r/m		
Direct intersegment	1 0 0 1 1 0 1 0	offset-low	offset-high	
		seg-low	seg-high	
Indirect intersegment	1 1 1 1 1 1 1 1	mod 0 1 1 r/m		

Mnemonic and Description	Instruction Code		
JMP = Unconditional Jump:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Direct within segment	1 1 1 0 1 0 0 1	disp-low	disp-high
Direct within segment-short	1 1 1 0 1 0 1 1	disp	
Indirect within segment	1 1 1 1 1 1 1 1	mod 1 0 0 r/m	
Direct intersegment	1 1 1 0 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	1 1 1 1 1 1 1 1	mod 1 0 1 r/m	
RET - Return from CALL:			
Within segment	1 1 0 0 0 0 1 1		
Within seg adding immed to SP	1 1 0 0 0 0 1 0	data-low	data-high
Intersegment	1 1 0 0 1 0 1 1		
Intersegment, adding immediate to SP	1 1 0 0 1 0 1 0	data-low	data-high
JE/JZ=Jump on equal/zero	0 1 1 1 0 1 0 0	disp	
JL/JNGE=Jump on less/not greater or equal	0 1 1 1 1 1 0 0	disp	
JLE/JNG=Jump on less or equal/not greater	0 1 1 1 1 1 1 0	disp	
JB/JNAE=Jump on below/not above or equal	0 1 1 1 0 0 1 0	disp	
JBE/JNA=Jump on below or equal/not above	0 1 1 1 0 1 1 0	disp	
JP/JPE=Jump on parity/parity even	0 1 1 1 1 0 1 0	disp	
JO=Jump on overflow	0 1 1 1 0 0 0 0	disp	
JS=Jump on sign	0 1 1 1 1 0 0 0	disp	
JNE/JNZ=Jump on not equal/not zero	0 1 1 1 0 1 0 1	disp	
JNL/JGE=Jump on not less/greater or equal	0 1 1 1 1 1 0 1	disp	
JNLE/JG=Jump on not less or equal/greater	0 1 1 1 1 1 1 1	disp	
JNB/JAE=Jump on not below/above or equal	0 1 1 1 0 0 1 1	disp	
JNBE/JA=Jump on not below or equal/above	0 1 1 1 0 1 1 1	disp	
JNP/JPO=Jump on not par/par odd	0 1 1 1 1 0 1 1	disp	
JNO=Jump on not overflow	0 1 1 1 0 0 0 1	disp	

Mnemonic and Description	Instruction Code	
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
JNS=Jump on not sign	0 1 1 1 1 0 0 1	disp
LOOP Loop CX times	1 1 1 0 0 0 1 0	disp
LOOPZ/LOOPE=Loop while zero/equal	1 1 1 0 0 0 0 1	disp
LOOPNZ/LOOPNE=Loop while not zero/equal	1 1 1 0 0 0 0 0	disp
JCXZ Jump on CX zero	1 1 1 0 0 0 1 1	disp
INT - Interrupt		
Type specified	1 1 0 0 1 1 0 1	type
Type 3	1 1 0 0 1 1 0 0	
INTO=Interrupt on overflow	1 1 0 0 1 1 1 0	
IRET=Interrupt return	1 1 0 0 1 1 1 1	
Processor Control		
CLC Clear carry	1 1 1 1 1 0 0 0	
CMC Complement carry	1 1 1 1 0 1 0 1	
STC Set carry	1 1 1 1 1 0 0 1	
CLD Clear direction	1 1 1 1 1 1 0 0	
STD Set direction	1 1 1 1 1 1 0 1	
CLI Clear interrupt	1 1 1 1 1 0 1 0	
STI Set interrupt	1 1 1 1 1 0 1 1	
HLT Halt	1 1 1 1 0 1 0 0	
WAIT Wait	1 0 0 1 1 0 1 1	
ESC Escape (to external device)	1 1 0 1 1 x x x	mod x x x r/m
LOCK Bus lock prefix	1 1 1 1 0 0 0 0	

Notes

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

if s w = 01 then 16 bits of immediate data form the operand
if s.w = 11 then an immediate data byte is sign extended to form the 16-bit operand.
if v = 0 then "count" = 1; if v = 0 then "count" in (CL)
x = don't care
if v = 0 then "count" = 1; if v = 1 then "count" in (CL) register
z is used for string primitives for comparison with ZF FLAG

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table.

16-Bit (w = 1)		8-Bit (w = 0)		Segment	
000	AX	000	AL	00	ES
001	CX	001	CL	01	CS
010	DX	010	DL	10	SS
011	BX	011	BL	11	DS
100	SP	100	AH		
101	BP	101	CH		
110	SI	110	DH		
111	DI	111	BH		

Instructions which reference the flag register file as a 16-bit object use the symbol **FLAGS** to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

APPENDIX-H

I BEL	ADDRESS	CONTENTS	MNEMONICS AND CPERANDS	COMMENTS
START	300	EE 00 04	MOV SI,400	
	303	B9 04 00	MOV CX, 04	
CLOOP	306	9A 00 15 00 00	CALL CHRIN	Store the Command characters
	30B	88 04	MOV [SI],AL	in command Buffer.
	30D	46	INCSI	
	30E	49	DEC CX	
	30F	75 F5	JNZ CLOOP	
	311	4E	DEC SI	Point SI at Bottom of command Buffer
	312	BF 0A 04	MOV DI,CMD$ BUFFER	Standard table address.
	315	B9 04 00	MOV CX,04	4 Character command.
	318	FD	STD	Auto decrement SI and DJ
LOOP	319	A6	CMPS	To compare with standard character.
	31A	7402	JZ NEXT	The two matches, Yes Compare next char.
	31C	EB E2	JMP START	NO; Again Scan for Command (Proper).
NEXT	31E	49	DEC CX	
	31F	75 F8	JNZ LOOP	All Chr. compared No. Check /or next Che.
LOOP 1	321	9A 00 15 0000	CALL CHRIN	Yes, Scan the Return Key.
	326	3C 0D	CMP AL,0D	Is Return Key Pressed.
	328	75 F7	JNZ LOOP 1	No; Scanfor Return Key.
	32A	9A 2F 03 0000	CALL DESPLAY1	Yes, Desplay the standard Index with LOAD THE CMD.No.

contd...

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
DISPLAY1	32F	BE 10 04	MOV SI,DATA b̸INDEX	Indix pointer for index data
	332	BB 86 00	MOV BX,COUNT	
	335	E9 74 00	JMP DISPLAY	
REP	338	9A 00 03 00 00	CALL CHRIN	to scan CMD No.
	33D	BF DC 04	MOV DI,CMNb̸NOb̸ TABLE	Command no table pointer.
	340	B9 04 00	MOV CX,04	Initialize the Counter.
LOOP2	343	FC	CLD	Auto increment DI pointer.
	344	A6	SCAS	CMpare CMD No.
	345	75 02	JZ next	Matches. Yes go to display
NEXT	349	EB 15 90 90 90	JMP CHECK b̸ DISPLAY	
	34E	49	DEC CX,	No Decrement Counter
	34F	75 F2	JNZ LOOP 2	Are all CMD No. compared (Scanned) No, Check another.
	351	9A E0 03 00 00	CALL ERROR	No CMD no.matches with CMD No.TABLE DISPLAY Error.
	356	C3	RET	Again Display CMND No. etc.

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	360	88 C4	MOV AH,AL	SAVE CMD No.
CHECK and DISPLAY	362	24 CE	AND AL,FE	IS CMD NO IS 1
	364	74 02	JZ next 1	YES GO to display for CMD 1
	366	EB OB	JMP CMD2	No, Check for next CMD No.
NEXT1	368	BE EA 06	MOV SI, Addr ≡ X1	Index pointer to display CMD 01
	36B	BB 64 02	MOV Bx,COUNT1	
	36E	9A - - 90 90	JMP DISPLAY	To display
CMD2	373	88 E0	MOV AL,AH	
	376	24 CD	AND AL,FD	IS CMD NO 02
	379	74 02	JZ Next 2	Yes:go to display CMD 1
	37B	EB OB	JMP CMD3,	No: check for CMD3
NEXT 2	37D	BE - -	MOV SI,Addr ≡ X2 table.	
	380	BB 86 02	MOV BX,COUNT2	To display CMD2
	383	EB 29 90	JMP DISPLAY	
CMD3	386	88 E0	MOV AL,AH	
	389	24 CC	AND AL,FC	
	38B	74 02	JZ NEXT 3	
	38D	EB 08	JMP CMD 4	
NEXT3	38F	BE - -	MOV SI,ADDr ≡ X3 table	
	392	BB - -	MOV BX,COUNT3	
	395	EB 15	JMP DISPLAY	
CMD4	397	88 E0	MOV AL,AH	
	399	24 CB	AND AL,FB	
	39B	74 02	JZ Next 4	
	39D	EB 08	JMP NO CMD	
NEXT4	39F	BE - -	MOV SI ,Addr ≡ X4 table	
	A2	BB - -	MOV Bx.	

LABELS	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	3A5	EB C5	JMP DISPLAY	
NO CMD	3A7	E8 - -	CALL ERROR	All the CMD no checked, non matches, hence display error.
	3AA	EB 12	JMP LCMNDISP	
			RET	
DESPLAY	3AC	BA F2 FF	MOV DX,FFF2	
STRT	3AF	EC	IN AL,[DX]	To display the character matter corresponding to the CMD No. whose Address is already in SI register.
	3 BC	24 C1	AND AL,C1	
	3B2	74 FB	JZ STRT	
	3B4	8A 04	MOV AL,[SI]	
	3B6	BA FC FF	MOV DX,FFFC	
	3B9	EE	OUT [DX],AL	
	3BA	46	INC SI	
	3BB	4B	DEC BX	
	3BC	75 EE	JNZ DISPLAY	
LCMN-DISP	3BE	BE A6 C4	MOV SI $ DATA $ FOR LCMDNDISP	
	3C1	BB 16 00	MOV BX,16	
INT	3C4	BA F2 FF	MOV DX,FFF2	
STRT1	3C7	EC	IN AL [DX]	
	3C8	24 C1	AND AL,01	
	3CA	74 FB	JZ STRT1	

LABEL	ADDRESS	CONTENTS	MNEMONICS AND OPERANDS	COMMENTS
	3CC	8A 04	MOV AL,[SI]	To display LOAD THE CMD NO
	3CE	BA F0 FF	MOV DX,FFF0	
	3D1	EE	OUT [DX],AL	
	3D2	46	INC SI	
	3D3	4B	DEC BX	
	3D4	75 EE	JNZ INT	
	3D6	E9 5F FF	JMP REP	Scan another CMD No.
	3D9			
CHRIN	1500	BA F2 FF	MOV DX,FFF2	
	1503	EC	IN	
	1504	24 02	CMP 02	
	1506	74 F8	JZ LOOP	
	1508	BA F0 FF	MOV DX,FFF0	
	150B	EC	IN	
	150C	C3	RET.	

INDEX DATA

```
1B      45      1B      48
0D      0A      0A      31      2E      20      20
4D      45      4E      55
0D      0A      0A      32      2E      20      20
53      4F      46      54      57      41      52
45      20      20      44      45      56      45
4C      4F      50      4D      45      4E      54
21      53      20      20      4F      4E      20
56      4D      43      2D      38      36
0D      0A      0A      33      2E      20      20
45      58      50      45      52      49      4D
45      4E      54      41      54      49      50
4E      20      20      57      49      54      48
20      20      44      49      46      46      45
52      45      4E      54      20      20      4D
4F      44      55      4C      45      53
0D      0A      0A      34      2E      20      20
53      54      45      50      50      45      52
20      20      4D      4F      54      4F      52
20      20      43      4F      4E      54      52
4F      4C


ICMDNO →0D      0A      0A      0A
        4C      4F      41      44      20      20
        54      48      45      20      20      43
        4D      44      20      4E      4F      2E
```

DATA FOR MENU DISPLAY

```
         1B    45    1B    48    0D    0A    20    20
         20    20    20    47    45    54    20    20
         41    51    49    4E    44    54    45    44
         20    20    57    49    54    48    20    20
         54    48    45    20    20    53    59    53
         54    45    4D    20    20    56    4D    43
         2D    38    36    20    27    49    54    48
         20    56    54    5A    2D    31    30    20
         54    45    52    4D    49    4E    41    4C

0D OA0A0A      28    61    29    20    52    41
4D       20    41    52    45    41    20    3A
20       35    4B    20    42    59    54    45
53       20    4F    4E    20    4Z    4F    41
52       44    20    3B    43    61    6E    20
62       65    20    65    78    70    61    6E
64       65    64    20    74    6F    20    31
32       38    4B    20    62    79    74    65
73
0D       0A    28    62    29    20    52
4F       4D    20    41    52    45    41
20       3A    20    31    3C    4B    20
42       59    54    45    53    20    4F
4E       20    42    4F    41    52    44
28       4D    4F    4E    49    54    4F
52       29    20    3B    43    61    6E
20       62    65    20    65    78    70
61       6E    64    65    64    20    74
6F       20    31    32    38    4B    20
62       79    74    65    73
```

```
0D    0A    28    63        29    20
52    41    4D    20        3A    44
45    43    4F    44        49    4E
47    20    56    41        52    49
41    52    49    4F        4E    53
20    46    4F    52  2D    0D    0A
20    20    20    20        28    69
29    20    36    31        31    36
20    28    32    4B        20    62
79    74    65    73        20    65
61    63    68    20        29    0D
0A    20    20    20        20    28
69    69    29    20        36    32
36    34    20    28        20    38
4B    20    62    79        74    65
6B    20    65    61        63    68
20    29    0D    0A        20    20
20    20    28    69        69    69
29    20    32    31        38    36
20    38    4B    20        62    79
74    65    6B    20        65    61
63    68    20    29
0D    0A    20    28        64    29
20    52    4F    4D        20    3A
44    45    43    4F        44    49
4E    47    20    56        41    52
49    41    54    49        4F    4E
53    20    46    4F        52    2D
0D    0A    20    20        20    20
28    69    29    20        32    37
31    36    20    28        20    32
4B    20    62    79        74    65
73    20    65    61        63    68
20    29
```

DATA FOR MENU CONTD...

```
OD    OA    20    20     20    20    28
69    29    20    32     37    33    32
20    28    20    34     4B    20    62
79    74    65    73     20    65    61
63    68    20    29
OD    OA    20    20     20    20    28
69    69    29    20     32    37    36
34    20    28    20     38    4B    20
62    69    74    65     73    20    65
61    63    68    20     29
OD    OA    20    20     20    20    28
69    76    29    20     32    37    31
32    38    20    28     20    31    36
4B    20    62    69     74    65    73
20    65    61    63     68    20    29
OD    OA    28    65     29    20    52
53    20    32    33     32    43    3B
54    54    59    20     41    4E    44
20    56    65    64     69    6F    20
43    61    73    65     74    74    65
20    49    6E    74     65    72    66
61    63    65
OD    OA    28    66     29    20    54
68    72    65    65     20    38    32
35    35    20    3A     20    39    20
49    2F    4F    20     70    6F    72
74    73
OD    OA    28    67     29    20
4D    75    6C    74     69    20
42    75    73    20     50    72
6F    76    69    64     65    46
```

DATA FOR SOFTWARE DEVELOPMENTS

```
1B      45      1B      48
0D      0A      53      2E        4E      2E
20      20      20      20        46      55      4E
43      2E      20      4E        41      4D      45
1B      46      21      3F        44      45      53
43      52      49      50        54      49      4F
4E                      1B        46      24      21
31      2E      0D      0A        20      32      2E
0D      0A      20      33        2F      0D      0A
20      34      2E      0D        0A      20      35
2E      0D      0A      20        36      2E      0D
0A      20      37      2E        0D      0A      20
38      2E      0D      0A        20      39      2E
0D      0A      31      30        2E      0D      0A
31      31      2E      0D        0A      31      32
2E      0D      0A      31        33      2E      0D
0A      31      34      2E        0D      0A      31
35      2E      0D      0A        31      36      2E
                1B      46        24      24      53
4F      41      50                        1B      46
24      3F      53      55        4D      20      4F
46      20      41      2E        20      50      2E
                1B      46        25      24      44
4D      41      44      44        1B      46      25
3F      44      45      43        49      4D      41
4C      20      4E      4F        2E      20      41
44      44      49      54        49      4F      4E
1B      46      26      24        42      43      44
41      1B      46      26        3F      42      2E
43      2E      44      2E        41      44      44
49      54      49      4F        4E      1B      46
27      24      53      52        4F      4F      54
1B      46      27      3F        53      51      55
41      52      45      20        52      4F      4F
```

DATA FOR SOFTWARE DEVELOPMENTS: contd...

```
54    1B    46    28    24    53
49    4E    45    1B    46    28
3F    54    52    49    47    41
4E    4F    4D    41    54    52
49    43    20    53    49    4E
45    18    46    29    24    42
54    47    43    1B    46    29
3F    42    49    4E    41    52
59    20    54    4F    20    47
52    41    59    20    43    4F
44    45    1B    46    2A    24
4F    4C    42    4D    56    1B
46    2A    3F    4F    56    45
52    4C    41    50    50    49
4E    47    20    42    4C    4F
43    4B    20    4D    4F    56
45    1B    46    2B    24    4D
55    4C    1B    46    2B    3F
33    32    20    62    69    74
2A    33    32    20    62    69
74    1B    46    2C    24    4D
41    54    4D    55    4C    1B
46    2C    3F    4D    41    54
52    49    58    20    4D    55
4C    54    49    50    4C    49
43    41    54    49    4F    4E
1B    46    2D    24    4E    4F
42    49    53    1B    46    2D
3F    4E    4F    2E    20    4F
46    20    42    59    54    45
53    20    49    4E    20    41
20    53    54    52    49    4E
47
```

DATA FOR SOFTWARE DEVELOPMENTS contd...

```
1B    46    2E    24    44    44
49    56    1B    46    2E    3F
44    45    43    49    4D    41
4C    20    44    49    56    2E
1B    46    2F    24    44    4D
55    4C    1B    46    2F    3F
44    45    43    49    4D    41
4C    20    4D    55    4C    54
49    2E    1B    46    30    24
41    4E    42    41    53    1B
46    30    3F    41    52    52
41    4E    47    45    20    4E
4F    2E    20    49    4E    20
41    53    43    45    4E    44
49    4E    47    20    4F    52
44    45    52    1B    46    31
24    41    4E    44    53    1B
46    31    3F    2E    2E    2E
2E    20    20    20    4D    45
43    45    4E    44    49    4E
47    2E    2E    2E    2E    1B
46    32    24    47    46    49
42    43    1B    46    32    3F
47    45    4E    45    52    41
54    45    20    46    49    42
4F    4E    41    43    43    49
20    4E    4F    2E    1B    46
33    24    4D    41    58    4E
49    53    1B    46    33    3F
4D    41    58    2E    20    4E
4F    2E    49    4E    20    41
20    53    54    52    49    4E
47
```

CMD No.3

```
1B    45    1B    48
0D    0A    0A
54    48    45    20     46    45    4C
4F    57    49    4E     47    20    4D    4F
44    55    4C    45     53    20    48    41
56    45    20    42     45    45    4E    20
54    45    53    54     45    44
1B    46    24    24     31    2E    20
4B    65    79    62     6F    61    72
64    20    53    69     6D    75    6C
61    74    6F    72     20    4D    6F
6D    75    6C    65
1B    46    26    24     32    2E    20
44    69    67    69     74    61    6C
20    49    2F    4F     20    43    41
52    44    1B    46     28    24    33
2E    20    53    74     65
70    70    65    72     20    4D    6F
74    6F    72    20     43    6F    6E
74    72    6F    6C
```

CMD4

```
1B    45    1B    48    0D    0A    0A
54    68    65    20    46    6F    6C    6C    6F   77
69    6E    67    20    45    78    70    65    72
69    6D    65    6E    74    73    20    48    61   76
65    20    43    65    65    6E    20    50    65   72
66    6F    72    6D    65    64
1B    46    24    24    31    2E    20
4C    69    6E    65    61    12    20    44    69
73    70    6C    61    63    65    6D    65    6E
74    20    43    6F    6E    74    72    6F    6C
1B    46    26    24    32    2E    20    53    2E
48    2E    4D    2E    20    57    69    74    68
20    55    6E    69    66    6F    72    6D
20    56    65    6C    6F    62    69    74    79
1B    46    28    24    33    2F    20    50    6F
69    6E    74    20    53    65    6C    65    63
74    69    6F    6E    20    4F    6E    20    61
58    59    20    50    6C    61    6E    65
1B    46    2A    24    2E    20
56    65    6C    6F    63    74    79    20
43    6F    6E    74    72    6F    6C    0D    0A
```

REFERENCES

1. Russell Rector- George Alexy, The 8086 Book Aosborne/Mc Graw Hill.

2. Intel's 8086 Mannual.

3. Hardware details of 8086 M/S. Vinytics notes.

4. J.W.Coffron, Introduction to 8086.

5. Srinivas M.P., The 8086 Processor, Reprinted/ Jan. 1985, CEDT, I.I.Sc., Bangalore.

6. VMC-86 Microcomputer's Mannual.

7. TOCCI, Microprocessor and Microcomputers.

8. VTZ-10, CRT, Mannual.

9. Intel's Microsystems Compunents Mannual Vol. I and Vol. II.

10. Nagrath And Kothari, Electric Machines.

11. Electronics For You, March, April, May, 1985 edition.

12. Rodnay Zacks, An Introduction To Microprocessors from Chips to Systems.

13. Laventhal, Introduction To Microprocessor, 1982.

14. Mohamed Rafiquzzaman, Microcomputer Theory and Applications with the Intel SDK-85, 1982.

15. Takasi Kenjo, Stepping Motors and Their Microprocessor Controls, Clarendon Press Oxford, 1984.

BIBILOGRAPHY

Stepper Motors

[1] Mc Clelland, W. (1927). The Application of Electricity in Warships, JIEEE 65, 829-71.

[2] Kieburtz, R.B. (1964). The Step Motor- The Next Advance in Control Systems, IEEE Transactions on Automatic Control. Jan., pp. 98-104.

[3] Byrne, J.V. and Lacy, J.C. (1976). Characterstics of saturated stepper and Reluctance Motors, IEEE Conf. Pub. 136, pp. 93-6.

[4] Walker, C.L.(1919). Improvements in and connected with Electro Magnetic Step by step Signalling and Synchronous Rotation. U.K. Patent 137, 150.

[5] Chicken, C.B. and Thain, J.H.(1920). Electrical Signalling Appratus, U.S.Patent 1353, 025.

[6] Thomas, A.G. and Fleischauer, F.J.(1957). The Power Stepping Motor- A new digital actuator. Control Engineering 4, (Jan.), 74-81.

[7] Bailey S.J. (1960) Incremental Servos, Part I- Stepping vs Stepless Control, Control Engineering 7, (Nov.) 123-7.

[8] Bailey, S.J.(1960) Incremental Servos, Part II- Operation and Analysis, Ibid.7, (Dec.) 97-102.

[9] Bailey, S.J. (1961). Incremental Servos, Part III-How They've Been used Ibid 8(Jan), 85-8.

[10] Bailey, S.J. (1961). Intermental Servos, Part IV-Today's Hardware. Ibid 8, (March), 133-5.

[11] Proctor, J.(1963). Stepping Motors Move In. Product Engineering 4, (Feb.) 74-188.

[12] Feiertag, K.M. and Donahoo, J.T. (1952). Dynamoelectric Machine US Patent 2,589, 999.

[13] Pawletko, J.P. (1972). Approaches to Stepping Motor Controls. Proc. First Annual Sympsium on Incremental Motion Control Systems and Devices, Department of Electrical Engineering, University of Illions, pp. 431-63.

[14] Hinds, W.E. (1974), The Sawyer Linear Motor. Proc. Third Annual Symposium on Incremental Motion Control Systems and Devices. University of Illions, pp. WI-W10.

[15] Chai, H.D. and Pawletko, J.P. Serial Printer with linear Motor Drive US Patent 4, 044, 881.

[16] Singh, G.,Gerner, M., and Itzkowitz, H(1979). Motion Control Aspects in Motors and Systems, University of Leeds, pp. 6-12.

[17] Biscoe, G.I. and Mills, A.S. (1977). The rationalization and Standardization of stepping motors and their test methods. Proc. Sixth Annual Symposium on Incremental motion Control Systems and devices, Department of Electrical Engineering, University of Illinois, pp. 331-42.

[18] Kuo, B.C. (1979). Step Motors and Control Systems, Chapter -6 SRL Publishing Company, Champaign, Illinois.

APPLICATIONS OF STEPPER MOTORS

[19] Chai, H.D. and Pawletko, J.P. (1977). Serial Printer with linear motor drive U.S. Patent 4, 044,881.

[20] Singh, G.Gerner, M. and Itzkwitz, H. (1979) Motion Control aspects in the Oyx intellegent type writter Proc. International Conference on Stepping Motors and Systems, University of Leeds, pp. 6-12.

[21] Patterson, M.L. and Haselby, R.D. (1977) A microstepped XY Controller with adjustable phase current waveforms. Proc. Sixth Annual Symposium on Incremental Motion Control Systems and Devices, Department of Electrical Engineering, University of Illinois, pp. 163-8.

[22] Patterson, M.L., Haselby, R.D. and Kemplin, R.M. (1917) speed precision and smoothness characterize four color plotter pen drive system. Hewlett Packard Journal 29,(1), 13-19.

[23] Hinds, W.E. and Nicto, B. (1973). The Sawyer linear motor. Proc. Second Annual Symposium on Incremental motion control Systems and devices Department of Electrical Engineering, University of Illinois, pp. W1-10.

[24] Hughes, R.O. (1975). Dyanamics of Incremental motion devices associated with planetary exploration spacecraft Proc. Fourth Annual Symposium on Incremental motion Control systems and devices. Department of Electrical Engineering, University of Illinois, pp. BB 1-8.

[25] Tolivar, A.F. and Hughes, R.O. (1976). Science platform pointing Control law for a planetary exploration spacecraft. Proc. Fifth Annual Symposium on Incremental motion Control systems and devices, Dept. of Electrical Engineering, University of Illinois, pp. AA 1-2.