MICRO-86 / 88 LCD

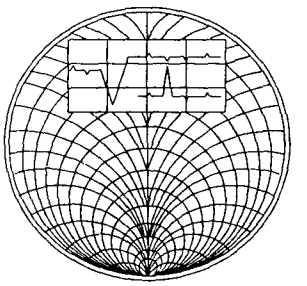
Technical Reference

Version 2.0

Technical Clarification / Suggestion :

✍ / ☎

Technical Support Division,
Vi Microsystems Pvt. Ltd.,
Chennai - 600 096.
Ph: 044-2496 3142, 044-5201 5899
E-Mail :service@vimicrosystems.com
Website: vimicrosystems.com



Vi Microsystems Pvt. Ltd.,
Chennai - 96

PREFACE

The *"MICRO-86/88 LCD trainer Technical Reference"* manual provides you with all the basic details of the trainer, its I/O & memory mapping, command function, connector details, cable configuration & the circuit diagram.

The manual starts with and introduction to the installation procedure for the trainer, with separate sections for 8086. Following this is a separate detailing with explanatory diagrams, the capabilities of the trainer with regard to additional development features.

CHAPTER 1 briefs the hardware & Software features of the trainer in a nutshell. This chapter will make you aware of the facilities that are available with the trainer. The forth-coming chapters elaborate the features mentioned in this chapter.

CHAPTER 2 deals with the hardware details of the trainer in detail. A knowledge of memory mapped I/O addresses of the various peripherals used, memory configuration of the functional block diagram can be had from this chapter.

CHAPTER 3 explains all the commands involve in M - 86/88LCD trainer trainer which can be given from an IBM- PC / AT keyboard.

CHAPTER 4 introduces you to the actual implementation of programs with our trainer, the ways in which you can execute your program & debug it.

CHAPTER 5 illustrates the details of the various connectors in the trainer & this will aid you in interfacing your trainer with an external system.

CHAPTER 6 is an introduction to the serial monitor facility provided by the monitor. However more detailed explanation out the serial monitor is given in a separate manual.

CHAPTER 7 is left for monitor system calls.

We hope this manual will be of immense use to you. We shall be grateful to consider suggestions for further improvement of this manual.

Write to:

The Customer-Support Division,
Vi Microsystems Pvt. Ltd.,
Plot No.75, Electronics Estate,
Perungudi, Chennai - 600 096.
Phone: (044) 2496 1842, 2496 1852.
Fax: (044) 2496 1536.
E-mail: service@vimicrosystems.com

CONTENTS

3) *SOFTWARE FEATURES*

4) *USER PROGRAM EXECUTION*

5) *CONNECTOR DETAILS*

6) *SERIAL MONITOR*

7) *MONITOR SYSTEM CALLS*

THE CHAPTER IS LEFT FOR FUTURE DEVELOPMENT.

LIST OF APPENDICES

INTRODUCTION

The MICRO - 86 / 88 LCD Trainer which you have acquired is a Microprocessor trainer which can work on 8086 CPU and hence the name with this tainer kit you can learn about the most popular 16 bit microprocessor of intel namely 8086, and a host of Intel peripherals. The trainer can assist you in learning the instruction set of these powerful processors, Assembly language programming, the interfacing of the peripherals and so on.

The MICRO - 86 / 88 LCD Trainer can operate with 16 bit processor 8086 in minimum mode. In the follwing section, the procedure to be followed in installing the kit has been outlined.

MICRO - 86 / 88 LCD Trainer has the power supply of built-in with multi out power connection, and can directly operate at 230V AC/50Hz mains. Hence installing the trainer does not require any further complicated procedure.

With unpacking the package, you will notify the following.

i) MICRO - 86 / 88 LCD trainer kit.
ii) MICRO - 86 / 88 LCD Tech Ref.
iii) MICRO - 86 / 88 LCD User Ref.
iv) PC- Power Chord.

INSTALLATION PROCEDURE:

The trainer works with 8086 CPU in the minimum mode for installing the kit with 8086 processor do as follows.

1. Power-on the trainer.

2. The trainer is issued a power-up -reset and the reset message is shown in Fig (i) is displayed. press RES key, if the display is not proper.

3. If you encounter any further problems please contact our customer support division before proceeding further.

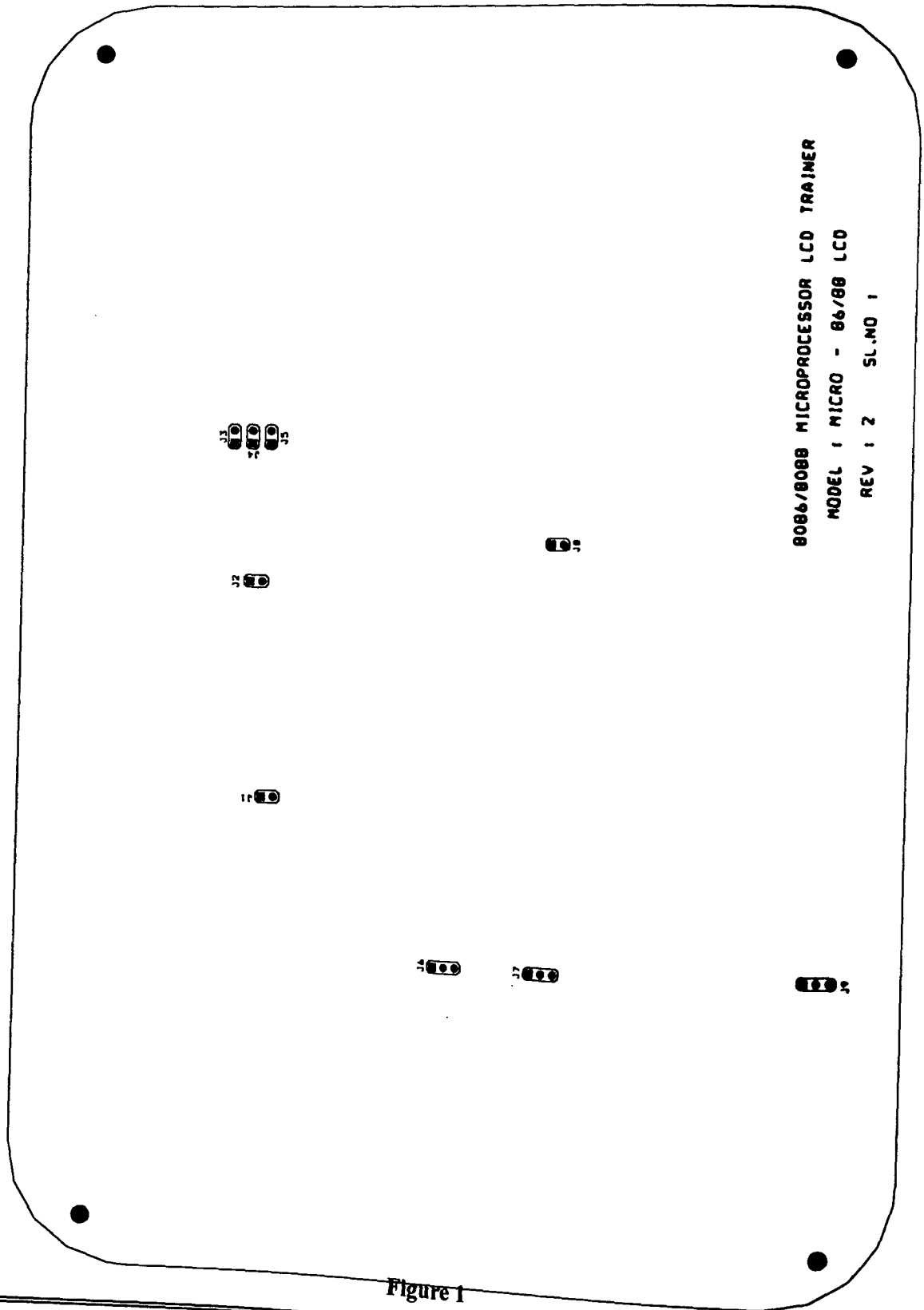
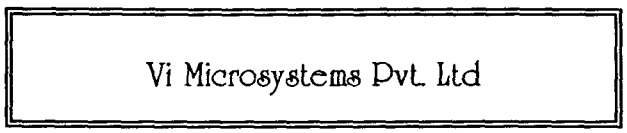

8086/8088 MICROPROCESSOR LCD TRAINER
MODEL : MICRO - 86/88 LCD
REV : 2 SL.NO :

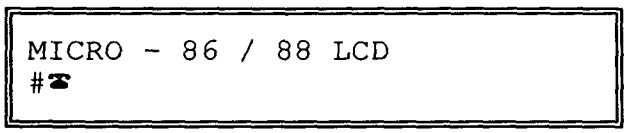
Figure 1

RESET MESSAGE:

INITIAL MESSAGE



FINAL MESSAGE



In order to perform the features provided with the trainer to your maximum advantage you are requested to confirm to certain basic norms and are requested to abstain from the following.

CAUTION:

1. Please don't tamper with any of the components in the trainer.

2. Please don't solder or anything else, when the power is up.

3. Don't atempt to service the trainer yourself in case of problems.

CHAPTER-1

GENERAL INFORMATION

1.1 INTRODUCTION:

This chapter highlights in brief M - 86/88 LCD trainer, the hardware and software facilities available on our trainer kit. Detailed information about the features mentioned in briefly in this chapter has been provided in the following chapters.

1.2 SPECIFICATIONS:

1.2.1 HARDWARE SPECIFICATIONS:

1. CPU PROCESSOR AND CLOCK FREQUENCY:

Intel 8086/8088 CPU at 5/4.77 MHz clock rate [Optionally 8 MHz]

2. MEMORY:

MONITOR EPROM	:	F000:0000 - 3FFF for 16 K
EPROM EXPANSION	:	F000:0000 - 1FFF for 64 K
SYSTEM RAM	:	0000:1000 - 3FFF for 16 K
SYSTEM RAM EXPANSION	:	0000:1000 - FFFF for 64 K
INTERRUPT VECTORS	:	0000:0000 - 03FF
STACK /DATA AREA (MONITOR, ASM/DSM, EDITOR)	:	0000:0400 - 0FFF

3. PERIPHERALS:

PARALLEL I/O	:	48 I/O lines using two numbers of 8255.
SERIAL	:	One number of RS232C - Serial Interface using 8251A USART.

TIMER	:	3 Channel 16-bit Programmable Timer 8253. Channel 0 used as baud rate generator for the Serial Port.

4. DISPLAY:

16×2 LCD DISPLAY (with Back light provision) **(or) 20 × 4 LCD** Display

5. KEYBOARD:

IBM PC-AT KEYBOARD

6. AUDIO CASSETTE INTERFACE with file management.

7. ON-BOARD BATTERY BACKUP: (Optional)

Onboard battery backup facility is optionally provided for RAM (U11 & U12).

8. POWER CONSUMPTION:

+ 5 V : 1.5 Amps

9. POWER SUPPLY SPECIFICATIONS:

Model	:	SMPS-01
Mains	:	230 Volts AC at 50 Hz
Input	:	230V AC at 50H
Output	:	1) + 5 Volts, 3 amps regulated 2) + 12 Volts, 50 mA regulated 3) - 12 Volts, 50 mA regulated

10. PHYSICAL CHARACTERISTICS:

M - 86/88 LCD trainer PCB : 260 mm x 680 mm

11. BUS EXPANSION:

A new concept of VXT_Bus has been incorporated in M - 86/88 LCD trainer which facilitates addition of extra hardware on to M - 86/88 LCD trainer. An unlimited number of Add-On Boards can be add to this Expansion Bus. All buffered address, data and control signals are brought out to this bus. The trainer provision for one VXT_Bus connectors.

1.2.2 SOFTWARE SPECIFICATIONS:

M - 86/88 LCD trainer contains a high performance 16K monitor program. It is designed to respond to User input, RS232 communications, Printer interface, parallel port interface and so on. The simple description of the command functions supported by this monitor is as follows:

COMMAND KEY FUNCTION MEM0RY:

RES

The RES key allows the user to terminate any activity and return the M - 86/88 LCD trainer to an initialised state. When pressed, the **"Vi Microsystems Pvt. Ltd"** message appears in the display for a few seconds and then the monitor will come to the command prompt "TECH- 8086" Only after this prompt is displayed, the commands will be accepted.

INT

The INT key allows the user to interrupt any activity and return to the command prompt.

CHAPTER-2

HARDWARE DETAILS

2.1. INTRODUCTION:

The hardware description of M - 86/88 LCD trainer provided in this chapter, highlights on all technical aspects including memory allocation, peripheral interfacing and power supply.

2.1.1 FUNCTIONAL BLOCK DIAGRAM DESCRIPTION:

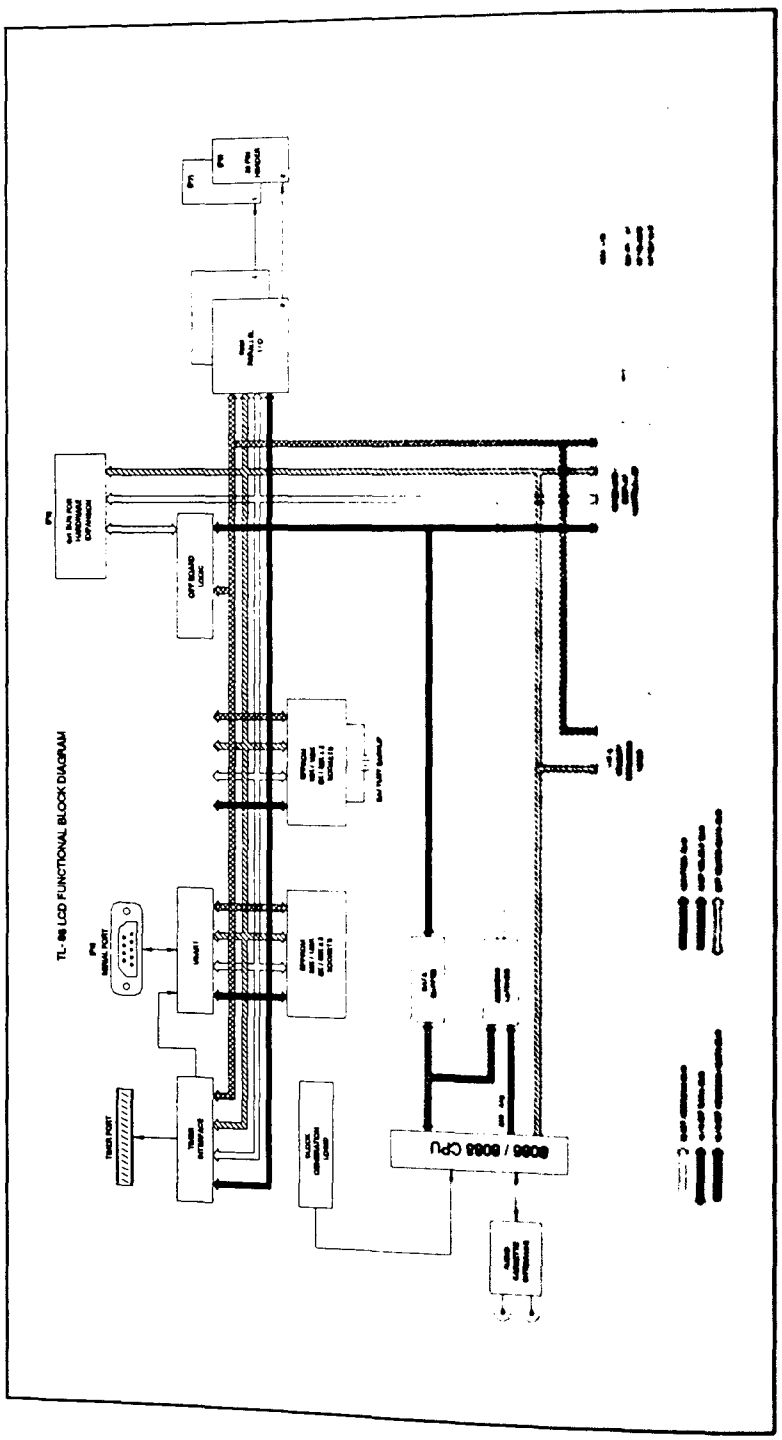
The Functional Block Diagram of M - 86/88 LCD trainer is given in Figure below. Referring to that figure, the following explanation outlines the working of the trainer as a complete system. To locate the individual components please refer to the Component Layout.

i. INPUTS AND OUTPUTS OF THE CPU:

The CPU gets the clock from the clock generator 8284 which uses a crystal at 15 or 14.318 MHz and if it is an 8 MHz trainer, 24 MHz. The Reset, Interrupt lines, and Data lines are also inputs to the CPU. The CPU outputs comprise the Address lines, Data lines and Control lines.

ii. ADDRESS AND DATA BUS:

The 16-bit 8086 bidirectional multiplexed Address/Data lines and the higher order 4 address lines are brought to latches and buffers. The address lines are latched using ALE and thus demultiplexed from the data lines. The data lines are buffered through octal transceivers. The outputs of these ICs comprise the 20 bit address and 16/8 bit data bus as the case may be. The direction of data transfer is decided by the direction select input of the transceiver which is the DT/R* output of the CPU.



BASIC FUNCTIONAL BLOCK DIAGRAM OF MICRO-86LCD TRAINER

iii. CONTROL BUS:

The other bus is the control bus. The control signals required for proper operation of the system are the IOR* (I/O Read), IOW* (I/O Write), MR* (Memory Read) and MW* (Memory Write). The peripherals on the trainer are all I/O mapped and hence to input from or output to a peripheral, IOR* and IOW* are utilised. The memory read and memory write signals are used to output enable an EPROM & RAM and write into a RAM respectively. These signals are generated from the IO/M* or M/IO* and WR*, RD* signals in minimum mode operation.

iv. CHIP SELECT LOGIC:

The selection of any peripheral or memory device requires a CS* (Chip Select) to enable that particular device. This requires address decoding, both memory and I/O where in decoding is separately achieved for EPROM, RAM and I/O devices.

All the above signals-address, data, control and chip select are routed to all the peripherals and memory devices in the trainer.

v. KEYPAD AND DISPLAY:

The block diagram shows a AT keyboard and LCD display. This is the unit with which the user communicates with the system 16 ×2 LCD module is used for displaying the command and data. IBM-PC-AT keyboard is used for entering data and control commands.

vi. MEMORY:

The Block Diagram shows 2 sockets for RAM both Odd and Even in the case of 8086 and 2 RAM sockets for contiguous locations in memory in the case of 8088. Two sockets are provided for RAM expansion. The basic EPROM capacity available is 16 KB arranged as Odd and Even banks in the case of 8086 and continuous locations in the case 0f 8088. The remaining EPROM capacity which is 16 KB can be availed as an optional facility. Each of the EPROM sockets can support either 8/16/32 KB memory devices and the RAM sockets 8/32 KB devices.

vii. PERIPHERALS:

The peripherals depicted in the Block Diagram include:

a) 8253 Timer for Baud Clock Generation.

b) 8251A USART for RS232C Serial Communication with associated drivers.

c) 8259 - Programmable Interrupt Controller.

d) Audio Cassette Interface.

e) 8255 PPI for Parallel TTL I/O.

System expansion is facilitated by the expansion slots available on board. The data bus to the expansion slots is sent through a second layer of buffering, the buffers being degated for addresses of all devices present on board.

2.2 COMPONENT LAYOUT AND FUNCTION BLOCK DIAGRAM

The above figure shows the functional block diagram for Micro-86LCD trainer. The functional block diagram provides the complete system design in blocks. Have a careful look at the components layout for the exact location for the connectors, IC's and so on. The component layout is provided in the Appendix for user reference.

2.3 MEMORY CONFIGURATION:

This section explains the memory mapping facilities available on the Micro-86/88LCD trainer. Fig F2.2. shows the complete Memory Allocation of Micro-86/88 LCD trainer. As shown in the figure, the Monitor reserves some portion of the total memory capacity of the CPU for its internal use and some for the user interface. The remaining portion is left for the user for their own development. This is indicated in the diagram as **"LEFT FOR USER DEVELOPMENT"**.

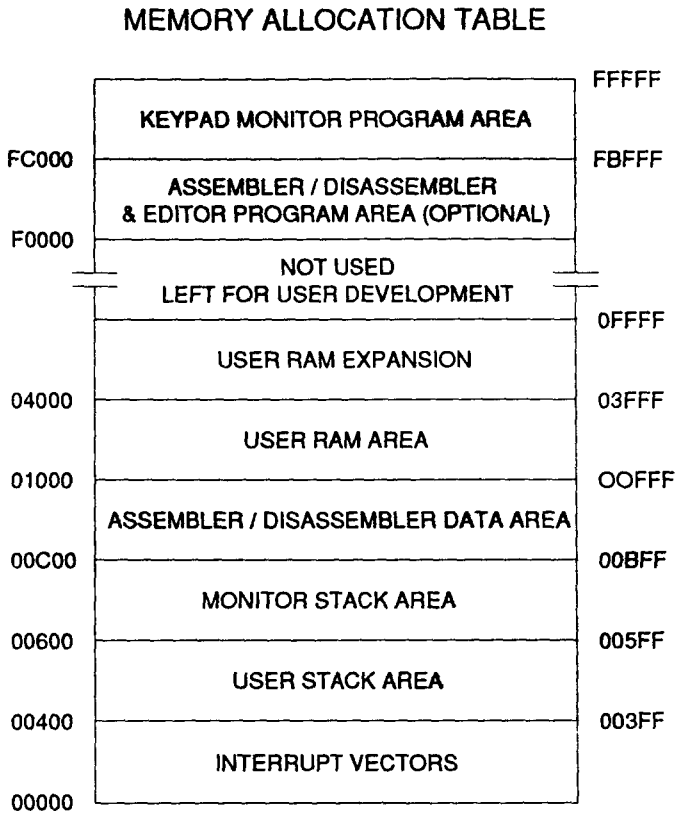


Figure 2.2 Memory Allocation Table for Micro-86LCD trainer

2.3.1 ALLOCATION OF EPROM:

The trainer has the Monitor EPROMs which control the complete system operation. The main software for responding to the user requests is programmed in these EPROMs. In case of 27256 EPROMs they will have the Two Pass Assembler, Disassembler and Editor which are optional. The following table gives the memory address map for Monitor EPROMs. The Monitor EPROMs of 16K and 64K capacity can be clearly distinguished by the Identification Code (I.D) labelled on the EPROM.

2.3.2 EPROM EXPANSION:

Out of the aggregate 16K [i.e., FC000 to FFFFF or F000:C000 to F000:FFFF], U1 occupies all EVEN addresses like FC000,FC002, and so on. U3 occupies all ODD addresses like FC001,FC003, and so on. Hence EPROM expansion can be done only in multiples of 16K Bytes.

TABLE : 2.3.1

SL.No	START ADDRESS	END ADDRESS	SOCKET NUMBER	IC USED	TOTAL CAPACITY	MONITOR EPROM I.D
1.	F0000	FFFFF	U5, U6	27256×2	64K Bytes	

2.3.3 ALLOCATION OF RAM AND RAM EXPANSION:

The RAM memory sockets have been designed to take either 6264 or 62256 ICs.

The basic capacity of read write memory available is 16K using two numbers of 6264. Expansion can be done by placing 62256 (in sockets U2 and U4) and placing J2 in position 2.

TABLE 2.3.2

SL. NO.	STARTING ADDRESS	ENDING ADDRESS	SOCKET NUMBER	IC USED	TOTAL CAPACITY
1	00000	03FFF	U11, U12	6264 × 2	16K Bytes
2	00000	0FFFF	U11, U12	62256 × 2	64K Bytes

2.4. BATTERY BACKUP: (OPTIONAL FACILITY)

Battery Backup is provided optionally for 16K bytes of RAM (U11 and U12) from address 00000 to 03FFF. If RAM expansion has been availed using 62256 ICs, the battery backup works for 64 KB. A 3.6V Ni-Cd Battery is used, which provides stand by power to RAM during power-off and gets charged during power-on. The data in RAM can be retained to a maximum of 24 hours if the battery has been charged to its maximum extent.

The above sections have dealt with the memory allocation of M - 86/88 LCD trainer . The forthcoming sections shall detail the peripheral mapping of the trainer, in the same fashion.

2.5. I/O ALLOCATION:

The complete I/O Allocation table of the peripherals of M - 86/88 LCD trainer is given in Figure F2.3. As seen from the table, the peripherals occupy I/O addresses from 0000 to 0060. The I/O addresses from 0081 to 00FF are used for Add-On Boards. Apart from these reserved addresses, the rest are available to users for their own development purposes.

The following sections briefly describe the various peripherals and interfaces available on M - 86/88 LCD trainer, emphasizing on the component used, its features, the connector termination and addresses. The Connector Pin Assignments can be had from the Chapter "CONNECTOR DETAILS".

I/O ALLOCATION TABLE

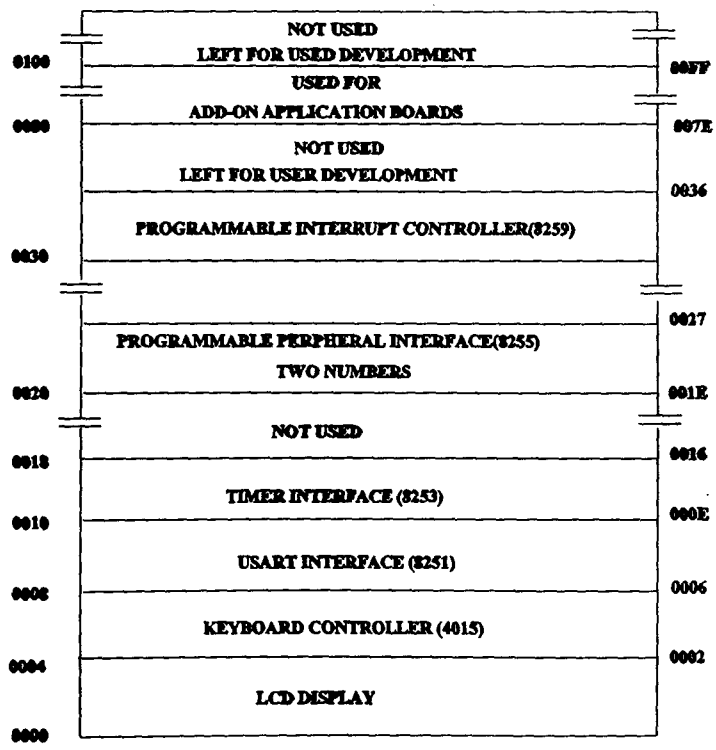


Figure 2.3

2.6. TIMER INTERFACE: (U30)

i) DEVICE USED : Intel 8253 - PROGRAMMABLE INTERVAL TIMER.

ii) KEY FEATURES OF 8253

a) 3 independent 16-bit counters.

b) Programmable counter modes.

c) Input clock upto 2Mhz.

d) Programable count from 2 to 2^{16}

iii) SYSTEM MAPPING : I/O mapped I/O

iv) CHANNEL 0

Input Clock : 1.25 / 1.19 MHz (or) User input clock TTL levels and frequency less than 2Mhz.

Output Clock : 153 KHz if using system clock.

Used for : Baud rate generation for the serial port.

v) CHANNEL 1 & 2

Input Clock : TTL level clock of rate less 2MHz can be given by user

Output Clock : User determined.

Used for : Available for user development.

vi) I/O MAPPED I/O ADDRESS:

FUNCTION	ADDRESS	SOCKET NO.
Control Register	0016	U30
Channel 0	0010	U30
Channel 1	0012	U30
Channel 2	0014	U30

2.7. RS232C SERIAL INTERFACE: (U8),(U9)

i) DEVICE USED : Intel 8251A-UNIVERSAL SYNCHRONOUS/ ASYNCHRONOUS RECEIVER/TRANSMITTER (USART).

ii) KEY FEATURES OF 8251A :

a) Both Synchronous and Asyn-chronous operations.

b) Full Duplex, Double buffered, transmitter and receiver

c) Parity, Overrun and Framing Error detection.

iii) SYSTEM MAPPING : I/O mapped I/O

iv) INPUT CLOCK OF 8251A : 2.5 / 2.35 MHz

v) BAUD CLOCK : Baud clock for the 8251A (U26) provided by Channel 0 of 8253 (U17)

vi) INITIALISED TO : 9600 Baud Rate, 8 Data bits, 1 Stop bit and No Parity upon Power-ON.

vii) DRIVER USED : MAX232 (U27) for RS232C Serial Transmitter and Reception.

viii) CONNECTOR TERMINATION : The RTS*, CTS*, RxD and TxD lines are the connector P3 and the connector is an AMPHENOL STANDARD 9-PIN D TYPE MALE.

ix) I/O MAPPED I/O ADDRESS:

FUNCTION	ADDRESS	CONNECTOR NO.	SOCKET NO.
8251A Control/ Status	000A	P11	U8
8251A Data	0008	P11	U8

2.8. PARALLEL INTERFACE: (U13, U23)

i) DEVICE USED : Intel 8255 - PROGRAMMABLE PERIPHERAL INTERFACE

ii) KEY FEATURES OF 8255

a) 24 Programmable I/O lines confi-gured as three 8-bit ports.

b) Direct Bit SET/RESET capability.

c) Three modes of operation.

- BASIC I/O
- STROBED I/O
- BIDIRECTIONAL BUS

iii) SYSTEM MAPPING : I/O mapped I/O

iv) 8255 PORT 1 (U30) : All 24 I/O lines are available to users.

v) 8255 PORT 2 (U31) : All 24 I/O lines are available to users.

vi) CONNECTOR TERMINATION : The 48 TTL I/O lines of both the 8255 ICs, U30 & U31 are terminated at the two **26-pin IDC connectors** P7(U30) and P8 and(U30) respectively. All these terminations are as per the Dio-1 bus stard.

viii) I/O MAPPED I/O ADDRESS:

FUNCTION	ADDRESS	CONNECTOR NO.	SOCKET NO.
I 8255 Control Register	0026		U13
I 8255 Port A	0020	P4	U13
I 8255 Port B	0022	P4	U13
I 8255 Port C	0024	P4	U13
II 8255 Control Register	0056		U23
II 8255 Port A	0050	P5	U23
II 8255 Port B	0052	P5	U23
II 8255 Port C	0054	P5	U23

2.9 LCD INTERFACE:

i) DEVICE USED : 16 × 2 LCD Module, IC 74LS174.

ii) SYSTEM MAPPING : I/O mapped I/O Address

I/O ADDRESS:

FUNCTION	ADDRESS	CONNECTOR NO.	SOCKET NO.
Display control/Data	04	LCD 1	LCD 1
RS and IORW	06	LCD 1	LCD 1

2.10 IBM/PC/AT KEYBOARD INTERFACE (U33, U34):

DEVICE USED : CD4015B (Keyboard controller)

SYSTEM MAPPING : Memory mapped I/O

KEYBOARD USED : IBM-PC Keyboard

I/O ADDRESS:

FUNCTION	ADDRESS	CONNECTOR NO.	SOCKET NO.
Data Read Keyboard	10	P10	U33
Data Read Keyboard	10	P10	U34

2.11 AUDIO CASSETTE INTERFACE: (U18, U21, U32)

i) DEVICES USED : 74LS04(U18) With Filtering and Wave shaping networks for Tape Read and Write operations.

ii) KEY FEATURES : File Handling Capability with a file name specification of 16-bits.

iii) SYSTEM MAPPING : I/O mapped I/O

iv) TAPE WRITE : This operation is done using 7404 (U18).

v) TAPE READ : This operation is done using PAL16L8 (U21) (as inverter) and 74LS125 (U32).

vi) CONNECTOR TERMINATION : The Tape Write Output is terminated at MIC socket (P4) and Tape Read Input is terminated at EAR socket (P5)

vii) I/O MAPPED I/O ADDRESS:

FUNCTION	ADDRESS	CONNECTOR NAME	SOCKET NO.
Tape Write	0024	MIC	U18
Tape Read	001A	EAR	U21,U32

2.12 BUS EXPANSION:

In M-86/88 LCD trainer, hardware expansion facility has been provided through a bus based architecture called Bus extender P6.

All the CPU Address lines, Data lines and Control signals are brought out through Address latches, Data transceivers and Control signal buffers and terminated at the P6 connector. The data bus to the expansion slots is double buffered and the buffers are deselected for addresses of memory and peripheral devices on the trainer.

Such bus based expansion helps in the easy plugging of a range of our add-on experiment boards for further experimentation and also makes it convenient for the users to build their own add-on modules. The P6 connector is brought out to 50 pin FRC connector.

2.13 POWER SUPPLY :[SMPS - 001A]

Model LPOW 001A power supply used in M - 86/88LCD trainer, is a high efficiency, multi output, switch mode power supply. This power supply has stable outputs with different current ratings.

POWER SUPPLY SPECIFICATIONS:

INPUT	:	230 V Ac 50 Hz.	
OUTPUT	:	DC outputs.	
		+5V / 3A	Regulated
		+12V / 150mA	Regulated
		-12V / 150mA	Un regulated

CIRCUIT DESCRIPTION

For the circuit diagram of this SMPS- 001A please refer the Appendix at the back.

The power supply as a whole has two sections, viz.,

(i) Rectifier section.

(ii) Regulator section.

CHAPTER 3

SOFTWARE FEATURES

3.1 INTRODUCTION:

This chapter describes the commands involve in M - 86/88 LCD trainer. Which can be given from an IBM-PC/AT keyboard. The M - 86/88 LCD trainer can accept any command related to the following short listed functions once in the command prompt mode, indicated by a "#" in the left most position of the second row in the LCD module. The function that can be performed by using the command are as follows.

* Line Assembler and disassembler command.
* Display and substitute memory locations.
* Dump memory contents.
* Display and modified registers of the CPU.
* Enter and intiate execution of your own programs.
* Debug your program through the trace facility provided by the monitor.
* Fill a block of a RAM memory.
* Move a block of memory to RAM or within RAM.
* Compare two blocks of memory.
* Insert bytes into RAM memory.
* Delete bytes from RAM memory.
* Input a byte or word from an input port.
* Output a byte or word to an output port.
* Transmit or received a block of data through the serial port.
* Print a block of memory.

In M - 86/88 LCD trainer command can be given from an IBM-PC/ AT keyboard. So the key board standard is universal. Each command is dealt in detail in the following sections.

The reset message in M - 86/88 LCD trainer is as follows.

```
Micro-8086 EB
#☎
```

The commands can be entered from the # prompt.

The syntax notation used in the following descriptions are as follows,

1) <Addr>, <Start Addr>, <End Addr>, <Dest Addr> <Program End Addr>, <Insert Addr> -- 16 bit hex address to be entered by user (16 bit in case ofword operation).

2) <Data> -- 8/10 bit data to be entered by user.

3) <File Name> -- An 16-bit hex data awhich specifies the file name in Tape commands.

4) <Port Addr> -- An 8-bit hex data (16-bit in cases of word that specifies an I/O port.

5) <CR> -- Carriage return.

NOTE:

1. A command should be entered immediately after the # prompt without blank space.

2. The display used is 16 × 2 LCD. If more than 16 characters are enetered, the display will be left shifted to accomodate the new entry in the following examples, the extra character (the commands which involved more than 16 characters) are given in the thrid line for illustration.

3.2 COMMAND LINE EDITOR FEATURES:

i) Single line editor can process upto 40 characters.

ii) Valid key functions are:

Enter	-- To validate an aentry. To increment memory location. To select from menu.
------ __	-- To decrement memory location. To select from menu.
> .	-- To terminate a command.
Back space	-- To delete a character and comeback one position.
Space Bar	-- To provide space in a command or data.
0-9	-- Numeric characters from 0 to 9
A - Z	-- Alphabetic Character from A to Z.

3.3. ASSEMBLER COMMAND:

FUNCTION:

This command is used to enter the mnemonics of 8086 and it gives the opcode for the mnemonics.

SYNTAX:

```
#A     <CR>
A     - Line Assembler Command.
```

PROCEDURE:

1. After entering the 'A' command, the kit is displayed as follows

2. Then, it asks the starting address

Now, you entered the user RAM address and programs, for every mnemonics for 8086 entered, it gives the opcode.

3.4 DIS-ASSEMBLER COMMAND:

FUNCTION :

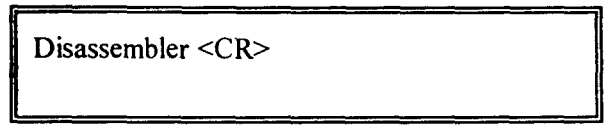
This command gives the mnemonics and corresponding opcode in the given address.

SYNTAX:

```
#U          <CR>
U     -     Dis Assembler Command.
```

PROCEDURE:

1. After entering the 'U' command, the kit is displayed as follows.



2. Then, if you enter the starting address.

3.5 SUBSTITUTE MEMORY COMMAND:

FUNCTION :

This command is used to examine the containts of selected memory location and modify the RAM contents if desired.

SYNTAX:

i) #SB <ADDR> <CR>

SB	-	Substitute byte command word
<ADDR>	-	User can enter the address either in segment: offset form or only offset form
<CR>	-	keyboard Return

ii) #SW <ADDR> <CR>

SW	-	Substitute word command word.
<ADDR>	-	Address either segment: offset or only.
<CR>	-	Keyboard return.

3.3. ASSEMBLER COMMAND:

FUNCTION:

This command is used to enter the mnemonics of 8086 and it gives the opcode for the mnemonics.

SYNTAX:

```
#A     <CR>
A     - Line Assembler Command.
```

PROCEDURE:

1. After entering the 'A' command, the kit is displayed as follows

2. Then, it asks the starting address

Now, you entered the user RAM address and programs, for every mnemonics for 8086 entered, it gives the opcode.

3.4 DIS-ASSEMBLER COMMAND:

FUNCTION :

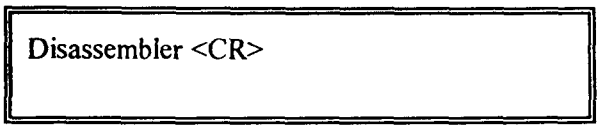
This command gives the mnemonics and corresponding opcode in the given address.

SYNTAX:

```
#U          <CR>
U     -     Dis Assembler Command.
```

PROCEDURE:

1. After entering the 'U' command, the kit is displayed as follows.



2. Then, if you enter the starting address.

3.5 SUBSTITUTE MEMORY COMMAND:

FUNCTION :

This command is used to examine the containts of selected memory location and modify the RAM contents if desired.

SYNTAX:

i) #SB <ADDR> <CR>

SB	-	Substitute byte command word
<ADDR>	-	User can enter the address either in segment: offset form or only offset form
<CR>	-	keyboard Return

ii) #SW <ADDR> <CR>

SW	-	Substitute word command word.
<ADDR>	-	Address either segment: offset or only.
<CR>	-	Keyboard return.

PROCEDURE:

1) After entering a substitute command, the contents of that address are displayed as follows.

```
Micro-8086 EB
=SB <Addr> <CR>
```

2) Use enter to examine the next location or to increment the memory location and the '--' key to view previous locations. ie. to decrement the memory location.

3) To modify the contents, enter the new data and press enter key.

4) To termiante this command press a dot (.)

```
Substitute byte
0000 : 1002 -00☎
```

ERROR CONDITIONS:

1) Attempting to modify the contents of a read only non exident memory location.

2) Invalid hex Address

3) Invalid hex data.

EXAMPLE:

1. To modify the contents of 0000 : 1000 to 3E and 0000 : 1001 to 22.

```
Micro-8086 EB
#SB 1000 <CR>
```

Enter the substitute command as snown in this figure.

```
Substitute byte
0000 : 1000 - 00 3E <CR>
```

The data 3E is entered and carraige return is pressed. Now 3E is entered in 1000 location.

```
Substitute byte
0000 : 1000 - 00 22 <CR>
```

The data 22 is entered in the location 1001.

```
Substitute byte
0000 : 1002 - 00 . <CR>
```

To terminate this command press dot (.)

```
Micro-8086 EB
#
```

Now, Micro-8086EB is at command prompt.

2. To modify the content of 0000 : 1100 to 5678

```
Micro-8086 EB
#SW 0000 : 1100 <CR>
```

Enter the substitute word command as shown

```
SW 0000 : 1100 - XXXX 5678 <CR>
```

The data 5678 is entered and carriage return is pressed. Now 5678 ios entered in 1100 location.

```
SW 0000 : 1102 - 0100.
```

To terminate this command, press dot (.)

```
Micro-8086 EB
#☎
```

Back to command prompt.

3.6: REGISTER COMMAND:

FUNCTION:

To examine and modify the register contents of the CPU.

SYNTAX:

#R <R>

R - Register view / modify command word.
<CR> - Carriage return.

PROCEDURE:

i) Enter register command when prompted for entry.

ii) Intially "AX" register is displayed

```
Register view
AX = 0000:☎
```

iii) Press enter or "-" key to view the subsequent or the previous register in the order as given in the table below.

Register	Description
AX	A Register
BX	B register
CX	C Register
DX	D Register
SP	Stack pointer
BP	Base pointer
SI	Source index
DI	Destination index
CS	Code segment
DS	Data segment
SS	Stock segment
ES	Extra segment
IP	Instruction pointer
FL	flag Register

iv) Change data to any register if desired and press carraige return

v) The sequence is circular and there fore apressing enter key after the last register order, will again display the first one.

vi) To terminater this command press dot (.)

EXAMPLE:

To modify the contents of DX register to 55, follow the sequences given below

```
Micro - 8086 EB
#R <CR>
```

Enter register command.

```
Register view
AX = 0000:<CR>
```

Skipping "AX" register by pressing enter

```
Register view
BX = 0000:<CR>
```

Skipping "BX" register by pressing enter.

```
Register view
CX = 0000:<CR>
```

Skip "CX" register

```
Register view
DX = 0000: *55 <CR>
```

Modify "DX" contents to 55H.

```
Register view
SP = 0800 : <->
```

Decrement and again view the contents of "DX" register.

```
Register view
DX = 0055 : .
```

```
Micro - 8086 EB
#☎
```

Back to command prompt.

3.7 GO & EXECUTE COMMAND:

3.7.1 GO & EXEC COMMAND:

FUNCTION:

The GO command us used to Run a program. This command transfers control of the 8086A CPU from the monitor program to user programs.

SYNTAX:

```
# GO <ADDR> <CR>
```

GO	-	GO Command.
<ADDR>	-	16 bit address. either segment : offset or offset only)
<CR>	-	Carraige return.

PROCEDURE:

i) Type GO command with address from where execution should start.

ii) Now, the control is transfered to the address entered by you and the display will be as shown below.

```
Executing...
```

iii) To exit from the execution and to return control to the monitor, press reset key or interrupt key in the kit.

iv) By using INT-2 instruction with opcodes CD01, you can transfer control to monitor without saving any registers. CD, 02 should be placed at the end of the program.

NOTE:

Before transfering control from monitor program to user programs, the register contents are updated using their respective buffers.

For instance, if you initialise AX = 23, using register command, and then use the GO command. "AX" will be updated to 23 before executing the user program.

EXAMPLE:

To start the execution of a user program at 0000 : 1000

```
Micro - 8086 EB
#Go 0000 : 1000 <CR> (or)#Go 1000 <CR>
```

Enter GO command

```
Executing ...
```

Transfer control to user program.

3.8 GO & EXEC WITH BREAK - POINT:

FUNCTION:

This command executes a block of program whose start and end address are specified.

SYNTAX:

#GB <Start Addr> <End Addr> <CR>

GB	-	GO with break command.
<Start Addr>	-	Program start Address.
<EndAddr>	-	Program end Address (ie) break-point address.
<CR>	-	Carraige Return.

PROCEDURE:

i) Enter GO command with the above said syntax.

ii) The program is executed only till the break-point. After reaching break-point, the registers are all saved up and break address with its contents are displayed. After sometime the control automatically returns to command prompt. After reaching the break point the display will be as follows.

```
Break Pt. Reached ..
<Break Pt. Addr> - <Data>
```

NOTE:

i) You cannot break-point an instruction in a ROM (Read Only memory Location).

ii) The Break point once set will be cleared up after reaching it. So, break-point address must be specified each time the program is to be executed with break point.

iii) The break-point address specified must contain data which will be taken by the processor as an opcode and not as an operand.

ERROR CONDITIONS:

Attempting to break-point a ROM location or Non-Existent memory location.

EXAMPLE:

To break-point a program at 1025 whose execution starts at 1000.

```
Micro - 8086EB
#GB 1000 1025 <CR>
```

Enter program block address.

```
Break point reached ...
0000 : 1205 - B0
```

Break point reached indication.

```
Micro - 8086 EB
#☎
```

Back to command prompt.

TRACE COMMAND:

FUNCTION:

This command helps the user to execute programs in steps i.e instrcution. This command will be very helpful while debugging programs.

SYNTAX:

#TR <Addr> <CR>

TR	-	Trace command word.
<Addr>	-	Program starting Address. (segment : offset of offset only)
<CR>	-	Carraige Return.

PROCEDURE:

i) Enter the Trace command with the above syntax.

ii) Now the first instruction is executed an the registers are updated.

iii) The next instruction to be executed and the memory location are displayed.

iv) Since, registers are updated after each individual instruction, you can view the register contents after any execution of individual instruction.

v) Press dot (.) to return to monitor and view registers or memory locations are desired.

vi) Continue single stepping after viewing register contents by agian executing trace command

```
Single step..
0000 : <Addr> - <Data>
```

3.9 FILL COMMAND:

FUNCTION:

This command permits a block of RAM memory to be filled with desired data byte.

SYNTAX:

i) #FB <Start Addr> <End Addr> <Byte Data> <CR>

FB	-	Fill Byte Command word.
<Start Addr>	-	Block starting address.(segment: offset or offset only) <End Addr>
<Addr>	-	Block ending address(segment : offset only)
<Data>	-	Word data to be filled.
<CR>	-	Carraige return.

ii) #FW <Start Addr> <End Addr> <Data> <CR>

FW	-	Fill word command word.
<Start Addr>	-	Block start address (Segment : Offset or offset only)
<End Addr>	-	Block end Address (offset only)
<Data>	-	Word data to be filled.
<CR>	-	Carraige return.

PROCEDURE:

i) Enter the Fill Command with the above syntax.

ii) Now, the block will be filled with the specified byte and control returns to monitor.

```
Break point reached ...
0000 : 1205 - B0
```

Break point reached indication.

```
Micro - 8086 EB
#☎
```

Back to command prompt.

TRACE COMMAND:

FUNCTION:

This command helps the user to execute programs in steps i.e instrcution. This command will be very helpful while debugging programs.

SYNTAX:

#TR <Addr> <CR>

TR	-	Trace command word.
<Addr>	-	Program starting Address. (segment : offset of offset only)
<CR>	-	Carraige Return.

PROCEDURE:

i) Enter the Trace command with the above syntax.

ii) Now the first instruction is executed an the registers are updated.

iii) The next instruction to be executed and the memory location are displayed.

iv) Since, registers are updated after each individual instruction, you can view the register contents after any execution of individual instruction.

v) Press dot (.) to return to monitor and view registers or memory locations are desired.

vi) Continue single stepping after viewing register contents by agian executing trace command

```
Single step..
0000 : <Addr> - <Data>
```

3.9 FILL COMMAND:

FUNCTION:

This command permits a block of RAM memory to be filled with desired data byte.

SYNTAX:

i) #FB <Start Addr> <End Addr> <Byte Data> <CR>

FB	-	Fill Byte Command word.
<Start Addr>	-	Block starting address.(segment: offset or offset only) <End Addr>
<Addr>	-	Block ending address(segment : offset only)
<Data>	-	Word data to be filled.
<CR>	-	Carraige return.

ii) #FW <Start Addr> <End Addr> <Data> <CR>

FW	-	Fill word command word.
<Start Addr>	-	Block start address (Segment : Offset or offset only)
<End Addr>	-	Block end Address (offset only)
<Data>	-	Word data to be filled.
<CR>	-	Carraige return.

PROCEDURE:

i) Enter the Fill Command with the above syntax.

ii) Now, the block will be filled with the specified byte and control returns to monitor.

ERROR CONDITIONS:

i) If the start address > end address.

ii) Attempting to fill ROM location or a Non-existent memory location.

EXAMPLE:

i) To fill a block from 0000:1000 to 1100 with data 33.

```
Micro - 8086 EB
#FB 1000 1100 33 <CR>
```

Enter Fill command

```
Fill Byte
```

(appears) flashes for a while.The filling operation is being performed.

```
Micro - 8086 EB
#☎
```

Back to the prompt.

ii) To fill a block from 0000 : 1200 to 1300 with data 1234.

```
Micro - 8086 EB
#FW 1200 1300 1234 <CR>
```

Enter fill word command

```
Fill Word
```

(appears for a moment).

The filling operation is being performed.

```
Micro - 8086 EB
#
```

Back to the prompt.

3.10 BLOCK MOVE COMMAND :

FUNCTION:

This command moves the contents of a specified block of memory to another block whose starting address is specified [to RAM]

SYNTAX:

i) #MB <Start Addr> <End Addr> <Dest Addr> <CR>

MB	-	Block move byte command word.
<Start Addr>	-	Block starting address.(segment : offset or offset only)
<End Addr>	-	Block Ending Address.(offset only)
<Dest Addr>	-	Destination Address.(segment : offset or offset only)
<CR>	-	Carraige reutrn.

ii) #MB <Start Addr> <End Addr> <Dest Addr> <CR>

MB	-	Block move word command .
<Start Addr>	-	Block starting address.(segment : offset or offset only)
<End Addr>	-	Block Ending Address.(offset only)
<Dest Addr>	-	Destination Address.(segment : offset or offset only)
<CR>	-	Carraige reutrn.

PROCEDURE:

i) Enter th Block Move Command with the above syntax.

ii) Now, the data is moved from source bnock to destination nlock and the control is transferred to monitor program.

iii) Address can be monitored in the display throughout the move block function.

NOTE:

i) Actually this command copies the specified block to the destination block byte by byte in the case of MB command and word by word in the case of MW command. The contents of the source block are not altered.

ii) So, if the destination address spoecified falls with in the source block, then it does not do the wark of the MOVE, the memory locations overlap and filling of locations with a similar data happens.

iii) The EPROM locations can be copied from, but cannot be coppied into.

iv) While entering the move command the display automatically will shift left when the command exceeds 16 bytes.

ERROR CONDITIONS:

i) Start address > end address of the source block.

ii) Attempting to move data to EPROM or Non-Existent RAM locations.

EXAMPLE:

i) To move the functions of the block from 1000 to 1200 to the block starting at 1300.

```
Micro - 8086 EB
#MB 1000 1200 1300 <CR>
```

Enter Move byte command

```
Move byte
```

Flashes for fraction of second
Move block is in progress.

```
Micro - 8086 EB
#
```

Back to the command prompt.

```
Micro - 8086 EB
#MW 1000 1200 1300 <CR>
```

Enter move word command.

```
Move word
```

Flashes for a fraction of a second
Move blocks is in progress.

```
Micro - 8086 EB
#
```

Back to command prompt.

3.11 BLOCK COMPARE COMMAND:

FUNCTION:

This command compared the contents of two block of memory locations and displays whose contents are also be modified.

SYNTAX:

```
#CB  <Start Addr>   <End Addr>  <Dest Addr>  <CR>
CB                -    Block compare Byte command word.
<Start Addr>      -    Block starting Address.
<End Addr>        -    Block Ending Address.
<Dest Addr.       -    Destination Address.
<CR>              -    Carraige return.
```

PROCEDURE:

i) Enter the block compare command with the above syntax.

ii) Now, both the blocks will be compared.

iii) If both the blocks contains identical data, then the system returns to prompt to indicate that both the block are equal.

iv) if unequal data is found anywhere then comparitive display is made as shown below.

```
<Start Addr> <Data> / <Dest Addr>
<Data>
```

Now the destination data contains can be modified, or, just enter key can be pressed for further comparison to be done.

v) So press enter key to continue compare block data, or press (.) dot to skip comparison.

vi) The system displays command prompt, if the rest of the blocks are equal, or after it has finished the full comparison between the two blocks. In the same example given here, first fill two blocks with a similar data and then compare them both.

Now, change the data within the blocks and check once again using the compare command. This is a much easier way to understand the block compare command.

EXAMPLE - 1:

First, fill a block of memory from 1000 to 1500 with hex data say 66, and compare the block from 1000 to 1200 with the block from 1300 to 1500. They should be equal.

```
Micro-8086 EB
#CB 1000 1200 1300 <CR>
```

Issue Block compare command.

```
Block compare
```

Flashes for a fraction of a second.

The two blocks are being compared.

```
Block compare
```

End address is reached.

```
Micro - 8086 EB
#
```

Now, micro - 8086 EB is at command prompt.

Since the two blocks specified contains the same data at respective address location (i.e) (1000) = (1300) ; (1001) = (1301) etc., the compare command returns to the command prompt, to indicate that the blocks are identical.

EXAMPLE 2:

Now, substitute memory location 1300's contents to 77 and issue the Block compare command.

```
Micro - 8086 EB
#CB 1000 1200 1300 <CR>
```

Issue Block compare command.

```
0000 : 1000 - 66
0000 : 1300 - 77
```

Since, the data differs both contents are displayed for comparative study and data can now be modified, or, just enter key can be pressed for further comparison.

```
Block compare
```

Flashes for a fraction of second. Since rest of the data in the blocks are identical address and data of the destination block is displayed.

```
Block compare
```

Flashes for a fraction of a second.
End address is reached.

```
Micro - 8086 EB
#☎
```

Now micro-8086 EB is at the command prompt.

3.12 INSERT COMMAND:

FUNCTION:

This command inserts the specified file bytes in the desired memory location.

SYNTAX:

i) #ISB <Insert Addr> < Program End Addr> <CR>

ISB	-	Insert Byte command prompt
<Insert Addr>	-	Insert Address.(segment : offset or offset only)
<Program End Addr>	-	End address of the block. (offset only)
<CR>	-	Carraige return.

ii) #ISW <Insert Addr> <Program End Addr> <CR>

ISW	-	Insert word command.

PROCEDURE:

i) Enter the insert command in the above given syntax.

ii) Now, the display will be as follows.

```
Micro - 8086 EB
#ISB <Ins. Addr> <Prg.end Addr> <CR>
```

iii) Now, enter the data to be inserted against the address followed by enter key every time.

iv) To terminate this command at any time, press dot (.)

v) The data enterd are written into memory starting from the insert address and therefore the entire block is shifted below one by one to make space for bytes to be inserted.

ERROR CONDITIONS:

i) Inserting into Read - only or Non-existent memory locations.

ii) If the insert address > program end address.

iii) Invalid data or address.

EXAMPLE:

To insert three bytes 11,22 and 33 starting at address 1000, sthe specified block ending at 1200.

```
Micro - 8086 EB
#ISB 1000 1200 <CR>
```

Enter insert command.

```
Insert Byte
0000 : 1000 - 1200 <CR>
```

Insert byte "11" at 1000

```
Insert Byte
0000 : 1001 - 22 <CR>
```

Insert byte "22" at 1001

```
Insert Byte
0000 : 1002 - 33 <CR>
```

Insert byte "33" at 1002

```
Insert Byte
0000 : 1003 - .
```

To terminate insert byte command.

```
Micro - 8086 EB
#☎
```

After three valid insertions, control is transferred to command prompt.

NOTE:

i) To insert bytes into memory, the block from insert address +1 to the end address specified is shifted down by one. Now a memory location is free to enter the data. This is what is actually done in the insert command.

ii) For each and every data to be inserted, the same procedure is followed. If you are to inser within a program you have entered, check the jump addresses to ensure that they are correct since you have inserted data into the program.

3.13 DELETE COMMAND:

FUNCTION:

i) This command deletes a block of bytes from a memory.

SYNTAX:

i) #DEB <Start Addr> < End Addr> <Program End> <CR>

DEB	-	Delete Byte command word
<Start Addr>	-	Delete block start Address.(segment : offset or offset only)
<End Addr>	-	Delete block End address (offset only)
<Program End Addr>	-	Program end address(offset only)
<CR>	-	Carraige return.

ii) #DEW <Start Addr> <End Addr> <Program End Addr> <CR>

DEW	-	Delete word command word

PROCEDURE:

i) Enter the Delete command in the above given syntax.

ii) Now, the block starting from block end address + 1 to the program end address, will be moved to the block starting from block start address.

ERROR CONDITIONS:

i) Attempting to delete bytes from Read-only or Non-Existent memory location.

ii) Block start address > Block end address.

EXAMPLE:

Delete the block from 1000 to 1050. The end address of the program that contains the block is 1200

```
Micro - 8086 EB
#DEB 1000  1050 1200 <CR>
```

Enter delete block command

```
Delete byte
```

Flashes for a fraction of second.

Block from 1051 to 1100 will be moved to 1000, so that previous contents are Deleted.

```
Micro - 8086 EB
#☎
```

Control is transferred to command prompt.

NOTE:

Now if you view location 1000, the data at 1051 would have been moved up here. Similarly, data from 1052 to 1001 and so on till 1200.

3.14 INPUT COMMAND :

FUNCTION :

This command inputs data from the desired port.

SYNTAX:

i) #IB <Port Address> <CR>

IB	-	Input Byte command word.
<Port Addr>	-	Valid Input Address. (i.e 0000 to FFFFH).
<CR>	-	Carraige return.

ii) #ISW <Port Addr> <CR>

IW	-	Input word command

PROCEDURE:

i) Enter IB command in the above stated syntax.

ii) Now, the data would be read and displayed as shown below.

```
Micro - 8086 EB
#IB<port Address>
```

<Data> - Data read from the input port.

iii) You can also modify the port assress.

iv) Press dot (.) to terminate the command.

EXAMPLE:

Input data from port Address 20.

```
Micro - 8086 EB
#IB 20 <CR>
```

Enter input command

```
Input Byte 55
IB 0020
```

The data from the port is displayed

```
Input data 55
IB . <CR>
```

To terminate this command press dot (.) and carraige return.

```
Micro - 8086 EB
#☎
```

Back to the prompt.

3.15 OUTPUT COMMAND:

FUNCTION:

This command outputs data to the desired port

SYNTAX:

i) #OB <port Addr> <Data> <CR>

OB	-	output byte command word
<port Addr>	-	Output port address (0000 to FFFFH)
<Data>	-	Output data (00 to FFH)
<CR>	-	Carriage return

ii) #OW <Port Addr> <Data> <CR>

PROCEDURE:

i) Enter the output command in the above syntax.

ii) Now the data entered is output to the port specified and the displayed will be as follows.

```
Output byte <Data>
OB <port Addr> <Data>
```

iii) Now again you are in out command mode that you can continuously execute out command or this command terminated with a dot (.) followed by a carriage return.

EXAMPLE:

To output data 45 to port address 60 and 55 to port address 70.

```
Micro - 8086 EB
OB 60 45 <CR>
```

Enter output command.

```
output Byte 45
OB 0060
```

Now, data 45H is output to port 60.

```
Output Byte 45
OB 7045 <CR>
```

Modify the port address and data.

```
Output Byte 55
OB 0070
```

Now, data 55 is output to port 70

```
Output Byte 55
OB. <CR>
```

Termiante this command using dot (.) followed by carriage return.

```
Micro - 8086 EB
#☎
```

Back to the command prompt.

3.16 SERIAL INPUT COMMAND:

FUNCTION:

This command inputs data from the 8251 port of M - 86/88 LCD trainer asynchronously.

SYNTAX:

#SI <Start Addr> <CR>

SI	-	Serial input command word.
<Start Addr>	-	Address to store the serial input data.
<CR>	-	Carriage return.

PROCEDURE:

i) Enter the serial input command in the above said format.

ii) Now, transmit data from the host system.(Pls refer to 'Note -2' for further discussion contact DATA TRANSMISSIOIN FROM HOST SYSTEMS)

iii) The M - 86/88 LCD trainer recevies data and stores it from the given starting address and returns to the command prompt on receiving the EOF MARK.(Ref Note-2)

iv) Using substitute memory command check for the data from the starting address specified.

NOTE - 1

i) An RS232C serial cable should be connected between the two systems.

ii) Ensure that the Baud rate setting of the host system is identical to that set in Micro - 8086EB.

iii) In M - 86/88 LCD trainer, the serial data format is set as undermentioned. It should be similar at the host system too.

Serial data format forM - 86/88 LCD trainer

Data	:	8 Bit
Stop bit (s)	:	1
Parity	:	None.

NOTE - 2:

i) The EOF mark is a data starting consisting of 5 "?" (3F)

ii) The host may be any system having RS232C serial port. It can be IBM PC / XT / AT, a kit etc.,

On a PC to kit communication the "DATACOM" pacakge available with us, directly supports the EOF Mark.

On a Kit to Kit communication, The EOF data is directly supported.

While transmitting the serial data from in other system ensure that the tansmitting system adds 5 "?" data at the end of transmission.

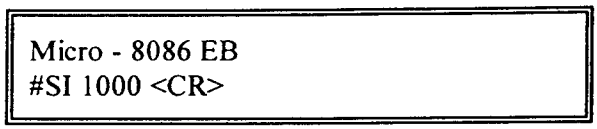
ERROR CONDITIONS:

i) writing into a Read-only or a Non -existent RAM locations.

ii) Due to mismatch reception, which can occur if the baud rates, the other communication parameters are not identical in both the transmitting and receiving system.

EXAMPLE:

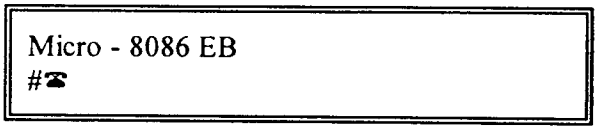
To serial input from USART starting from address 1000.



Enter the serial input command

Flashes for a fraction of a second.

Waiting for data at serial port.



Back to command prompt.

NOTE:

Transmit data from the transmitting system only after making the kit into resume mode. If the trainer receives five 3F, if automatically comes to prompt, else reset the kit.

3.17 SERIAL OUTPUT COMMAND:

FUNCTION:

This command outputs data serially through the 8251 of the M - 86/88 LCD trainer asymchrounously.

SYNTAX:

```
#SO  <Start Addr>   <End Addr>  <CR>
SO          -    Serial output command word.
<Start Addr>    -    Start address of the transmit data
                     block.(Segment : offset or offset
                     only).
<End Addr>      -    End Address of the transmit data block
                     (offset only)
<CR>            -    Carriage return.
```

PROCEDURE:

i) Enter the serial output command in the above said format.

ii) Ensure that the const system is set in the receive mode.

iii) Now, data will be transmitted serially. M - 86/88 LCD trainer returns to command prompt after data transmission is over.

NOTE:

i) If serial port of the receiving system is not proper, then the trainer waits with the message.

ii) Make sure your end of file contains five 3F.

EXAMPLE :

To serial output data from 1000 to 1200.

```
Micro - 8086 EB
#SO 1000 1200 <CR>
```

Enter the serial output command.

```
Serial output
```

Flashes for a fraction of a second.

Transmitting data from 1000 to 1200 and address can be monitored throughout the process.

```
Micro - 8086 EB
#☎
```

Back to the command prompt.

3.18 TAPE WRITE COMMAND:

FUNCTION:

This command writes data from a specified block of memory on to an audio tape.

SYNTAX:

#TO <Start Addr> <End Addr> <CR>

To	-	Tape write command word.
<Start Addr>	-	Starting address of the blocks of data to be saved
<End Addr>	-	End Address of the block
<File name>	-	Valid file name (i.e 0000 To FFFF)
<CR>	-	Carriage reutrn

PROCEDURE:

i) The file name is a hex data from 0000 to FFFF.

ii) Keep the taspe recorder ready. Insert audio cassette. Keep treble, bass and volume controls to maximum position.

iii) Press the play and record key in the tape recorder (i.e., keep the tape recorder in record mode).

iv) Enter the Tape write command in the above said format.

v) Now Data will be transmitted from the specified block along with the start address, end address. The file name and a checksum of all data in the block.

NOTE:

i) Prior to execute the tape write command, ensure that the EAR of tape recorder is connected to the MIC of the kit, through the cable.

ii) It is available to wait for one second after pressing record key in the tape, for the tape motor to revolve at the constant speed.

EXAMPLE:

To write a block of data from 1000 to 1200 on to an audio cassette under the file name AA

```
Micro - 8086 EB
#TO 1000  1200 <CR>
```

Enter tape write command. Keep tape recorder ready in record mode before pressing enter key. Now the prompt will be,

```
Tape out
File - AA <CR>
```

Enter the file name

```
Tape out
saving..
```

Saving file on to the tape.

```
Micro -8086 EB
#
```

Back to the command prompt.

3.19 TAPE READ COMMAND

FUNCTION:

This command inputs data from the output of a audio taperecorder where data were stored using a tape write command.

SYNTAX:

```
#TI <CR>
TI        -    Tape read command word.
<CR>      -    Carriage return.
```

PROCEDURE:

i) Enter the TI command, now the prompt will be as follows

```
Tape in
File ?
```

ii) Then enter the file name

```
Tape IA
File - <File name>  <CR>
```

iii) You are not prompted for any addresses, intead the start address of the block will be taken from the file that is read. (Remember it was written onto tape along with the data.)

iv) Rewine the tape and connect the LINEOUT of tape recorder to EAR of the kit. Now press the PLAY key.(i.,e keep the tape recorder in the PLAY mode)

v) The kit reads data fro m the tape and starts checking for a file.

vi) If the file is found, the data is loaded, a separate checksum is found an compared with the cheksum read from tape.

vii) In both the checksum match, the system displays the address till which the data is stroed and returns to comand prompt.

viii) if no matching file is found the system repeats getting data from tape. To terminate the function reset M - 86/88 LCD trainer.

ERROR CONDITIONS:

i) The checksum read from the tape calculated checksum are not equal.

ii) Invalid file name.

EXAMPLE:

To read a file named AA from an audio tape. Assume that the file AA ois the third file in the suido cassette you have recorded . Now, if the audio tape as the rewoudfully, you should read only the thrid file as the required one.

```
Micro-8086EB
#TI <CR>
```

Enter tape input command

```
Tape IA
File - AA <CR>
```

Enter input file name

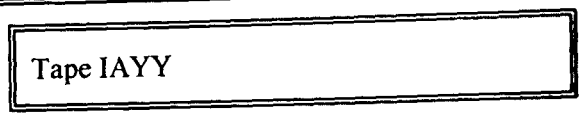
```
Tape IA
Load ...
```

Searching for a file

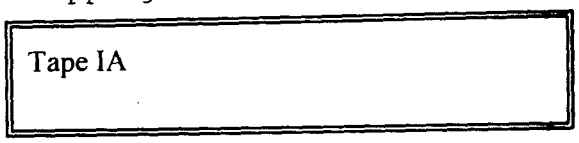
```
Tape IA XX
```

```
Tape in
```

Again searching for a file AA

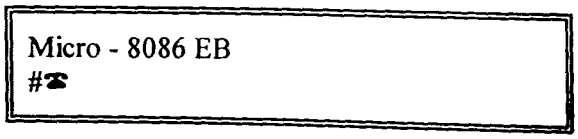


Skipping YY file



Searching for file AA

File AA detected, Reading data from the address where data is stored.



Back to the command prompt

3.20 PRINT COMMAND

FUNCTION:

This command prints the contents of a block of memory onto a centronics comaptible parallel printer.

SYNTAX:

```
#PR  <Start Addr>   <End Addr>   <CR>
     PR             -    Print command word.
     <Start Addr>   -    Print  blocks start address(segment :
                         offset or offset only)
      <End Addr>    -    Print  block end Address (offset only)
      <CR>          -    Carriage reutrn
```

PROCEDURE:

i) Enter print command in the above format. If a parallel printer is kept connected to the printer port, the printing will be done else printer error will reported.

ii) The printing will be in the following format. Address.... 16 data in hex..... ASCII codes for the 16 data.

ERROR CONDITIONS:

i) If the printer is not ready or if the interface is not centronics compatible.

ii) If the start address is said > end address.

NOTE:

Use printer adapter cable connect the printer to the parallel port connector (P12).

EXAMPLE :

To print the contents of locations from 1000 to 1200.

```
Micro-8086EB
#PR 1000 1200 <CR>
```

Enter print command

```
Printing
```

Printing the progress, address can be monitored.

```
Printing done
```

End address reached

```
Micro-8086EB
#
```

Back to the command prompt.

NOTE:

If the printer is not available, or if there is no paper sensed, error message is displayed.

3.21 SERIAL MONITOR COMMAND:

FUNCTION:

Using these serial monitor command you can operate your trainer through computer. before that you will connect RS232C cable (proper configuration) to the computer to any one of the serial port.

```
Micro-8086EB
#SM
```

and press enter and displayed like this.

```
Serial monitor
```

NOTE:

A detailed discussion of the serial monitor, which includes the procedure for invoking the same and the command supported as been provided in a separate chapter (chapter 6).

CHAPTER - 4

USER PROGRAM EXECUTION

4.1 INTRODUCTION:

A Program is a sequence of binary values, comprised by the opcodes and the operands. If the user knows all the opcodes and the operands, then the hex values can be directly entered into the trianer,. For entering the program, the usage of command keys SUB, Go, EXEC must be known. Explanation regarding the function of all these command keys has been detailed in chapter 3 "key function"

Most often it is not possible to remember all the opcodes permitted for any CPU. Hence it is customary to use a utility called "ASSEMBLER", which converts the program for mnemonics to opcode form. The assembled program has to be linked and converted into absolute form before it is presented for execution to the CPU.

All these utilities are available as standard program accompany MS-DOS. For storing the program, any word processor can be employed. The M-86/88 LCD trainer also has an Editor program, which can also be used to store source programs in the mnemonic form.

The next step is to assemble the program. This can be done using MASM which is the Assembler provided by MS-DOS, or by using the Vi-MDS 86 two pass Assembler which is an optional faciltiy that can be availed with the M-86/88 LCD trainer.

programs assembled with the Vi- MDS Assembler can straight away be executed without further modifications, while programs assembled using MASM have to linked using the MS-DOS link utility, converted to executable form and then dumped to the trainer serially before asking the CPU to execute the program

This chapter explains both the creation of the program. in the opcode form and then entering the program iun the trainer, top executed by the CPU. Througout this chapter we shall refer to an "executable program" as a "Binary file".

4.2 WRITING A PROGRAM:

The sample program taken is a 16-bit addition program. Let the program can be entered from origin 0 : 1000. The manual M-86/88 LCD trainer user has a number of programs which can be tried in a similar manner. The program given below can be entered in the mnemonics or source form using in the editor has explained earlier, or using any other editor utility. Let thefile be called test. ASM.

```
org  1000
mov  ax,1234
mov  bx,5678
add  ax,bx
mov  [1200],ax
hlt
```

4.3 ASSEMBLING THE PROGRAM:

The micro - 86/88 EB itself supports a two pass Assembler and disassembler and text editor. The user can use this option and can write porograms directly into the micro-86/88 EB memory in 8086 mnenomics. If the user is employing the MASM utility then the listing produced by the assemble is shown below. Usage of the Vi-MDS 86 two pass Assembler is explained a separate manual.

To use MASM, at the DOS prompt enter

MASM test,,,, where test is the primary file name of the program. The file name extension should always be "asm"

This will create a list file which has the extension as "lst", an object file which has the extension as "obj", and a file with extension "crf", the primary file name being "test".

The "obj", file is the input to the linker and the ".lst", file is a program listing with mnemonics and object codes and can be used for reference.

MASM LISTING:

```
                                ; Program to illustrate the use of
                                ; an assembler output format.
                                ;
= 0025                          value 1 equ          25th
= 0025                          value 2 equ          25th
0000                            code segment
                                assume    cs:code,ds:code
                                ;
1000                             org 1000h
                                ;
1000                start:
1000 B0   25                    mov  al,value1
1002 B3   25                    mov  bl value2
1004 02   C3                    add  al,bl
1006 A2   0A   10               mov  ds: [result], al
1009 F4                         hlt
                                ;
100A 00                         result    db         00
                                ;
100B                            code ends
                                end
```

The assembler format listed above uses a lot of assembler directives. Thesignifications of a few commonly used directives is mentioned below.

org	:	It is the directive used to indicate the address from which assembling to be done. Specifying no such "org" will initiate the assembler to start from 0000H. The 'H' succeeding the numeral denotes that the number is denoted in Hexadecimal.
equ	:	It assigns values to initialised variables used in the program.
end	:	It indicates to the assembler to stop assembly. It usually marks the end of a program and it is a must
;	:	A semi-colon at the start of a line denotes that the line denotes that the line will not be taken for assembly. so it is used to write commands.

segment : Indicates the start segment. it is denoted by a unique name which is not reserved one . If a program has no "segment" statement and error message will be displayed.

ends : Indicates a end of segment and it is a must.

assume : Indicates to the assembler that the symbols in the segment can be assesed using this segment registered. The valid segment register entries are CS, DS, ES, SS.

db : It is used to define a byte at a location. This is genrally used to create lookup tables simialr to this are the

dw - Define word
dd - Define double word
dq - Define Quad word
dt - Define ten bytes.

4.4 LINKING THE PROGRAM:

The .obj file output of the assembler cannot be understood by the CPU. to generate proper lable reference the .obj has to be linked.

To used link, at the DOS prompt enter

Link test;

The output of the linker is an executable file which can be run on the system under MS-DOS but not in the kit, because the kit does not have the monitor program that understand exe file formats and header structures. So this exe file should be converted to plan binary code which can be done as below.

4.5 CONVERTING TO BINARY CODE:

The "link" program creates the file "test.exe". This file has to be converted to a binary file before being downloaded to the trainer. This is acheived by employing the exe.2bin utility, which convers exe files to binary files.

To use EXE2BIN, at the DOS prompt enter,
exe2bin test test.bin
This creates the file test.bin which can be used straight away to be executed by the CPU in the M-86/88 LCD trainer:

4.6 ENTERING THE PROGRAM INTO THE TRAINER:

To download the program to the traiener, use can be made of the "DATACOM" utility provided by us. The kit should be in the serial input mode and DATACOM should be in the TRANSMIT menu. Syntax for using the serial input command has been provided in the previous chapter.

To enter the object codes manually in the kit, use the "SB" (substitute memory) command. Enter thedata starting form 0000 :1000. The data in the memory locations from 1000 is as given in the following table:

ADDRESS	OPCODE	OPERAND	MNEMONIC	OPERANDS
1000	B8	34 12	mov	ax,1234
1002	BB	78 56	mov	bx,5678
1004	01	D8	add	ax,bx
1006	A3	00 12	mov	[1200],ax
1009	F4		hlt	

PROCEDURE:

The message displayed as follows

```
Vi Microsystems Pvt. Ltd.,
```

Flashes for a fraction of second and displayed as

```
Micro-8086EB
#
```

End program starting Address.

```
Micro-8086EB
#SB 1000 <CR>
```

```
Substitute Byte
0000 : 1000-XX BB <CR>
```

```
Enter the data '34'
```

```
Substitute Byte
0000 : 1001-XX 34 <CR>
```

```
Enter the data '12'
```

```
Substitute Byte
0000 : 1001 - XX 12 <CR>
```

```
Enter the data 'BB'
```

```
Substitute Byte
0000 : 1003-XX <CR>
```

```
Enter the data '78'
```

```
Substitute Byte
0000 : 1004-XX 78 <CR>
```

```
Enter the data '56'
```

```
Substitute Byte
0000 : 1005-XX 56 <CR>
```

```
Enter the data '01'
```

```
Substitute Byte
0000 : 1006-XX 01 <CR>
```

Enter the data 'D8'

```
Substitute Byte
0000 : 1003-XX D8<CR>
```

Enter the data 'A3'

```
Substitute Byte
0000 : 1008-XX A3 <CR>
```

Ente the data '00'

```
Substitute Byte
0000 : 1009-XX 00 <CR>
```

Enter the data '12'

```
Substitute Byte
0000 : 100A-XX 12 <CR>
```

Enter the data 'F4'

```
Substitute Byte
0000 : 100B-XX F4<CR>
```

Now the opcode for addition program is entered.

```
Substitute Byte
0000 : 100C-XX <CR>
```

Press ' .' (dot) for command prompt.

```
Micro - 8086 EB
#☎
```

4.7 EXECUTION OF THE PROGRAM:

To execute the program, make use of 'GO' Command and check the result at 1200 location. The procedure is given below.

Enter the 'GO' comand and starting address of program.

```
Micro-8086 EB
#GO 1000 <CR>
```

```
Executing
```

Program is executed. Reset the system.

Substitute the result in address 1200.

```
Micro-8086 EB
#SB 1200 <CR>
```

```
Substitute Byte
0000 : 1200-AC <CR>
```

```
Substitute Byte
0000 : 1201-68 <CR>
```

Press ' . ' (dot) for back to command prompt.

4.7.1 USE OF SINGLE STEP COMMAND:

The single step command can be used to execute your program in steps, i.e., instruction by instruction and facilitates debugging the program easily since the results or program flow can be examined after each instruction.

```
Micro-86 EB ]
#TR 1000 <CR>
```

Move the data 1234 to the reg. AX .

```
Single step ....
0000 : 1003 - BB <CR>
```

Move the data 5678 to reg BX.

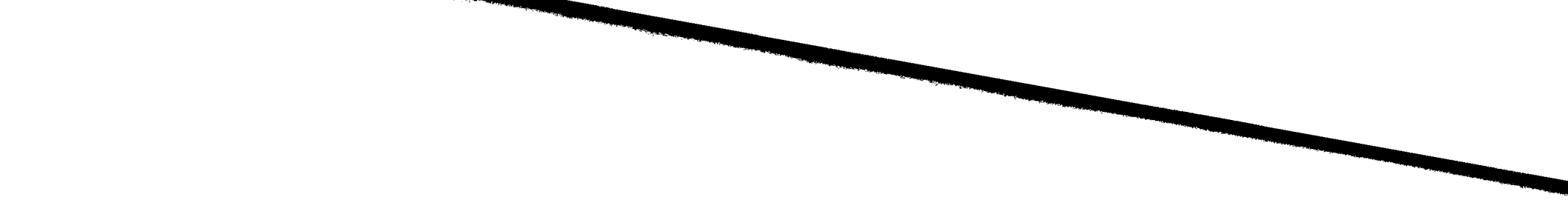
```
Single step ....
0000 : 1006 - 01 <CR>
```

Add the ax and bx reg. and result load to reg. ax.

```
Single step ....
0000 : 1008 - A3 <CR>
```

The data in reg. ax load to address 1200.

```
Single step ....
0000 : 100B -F4 <CR>
```



CHAPTER 5

CONNECTOR DETAILS

5.1 INTRODUCTION:

The following is the list of all the connectors available on M - 86/88 LCD trainer. The ensuing sections present more details on the pin Assignments of the connectors listed below.

P1	-	Power connectors
P4, P5	-	Parallel port connectors
P2	-	EAR input(Audio interface)
P3	-	MIC output (Audio interface)
P6	-	Bus expansion connector
P11	-	Seriasl port connector
P7	-	VXT_Bus connectors
LCD1	-	IBM_KBD & LCD Interface.
P10	-	Timer Port Connector
P8	-	IBM_KBD & LCD Interface.
P9	-	Interrupt port connector.

NOTE:

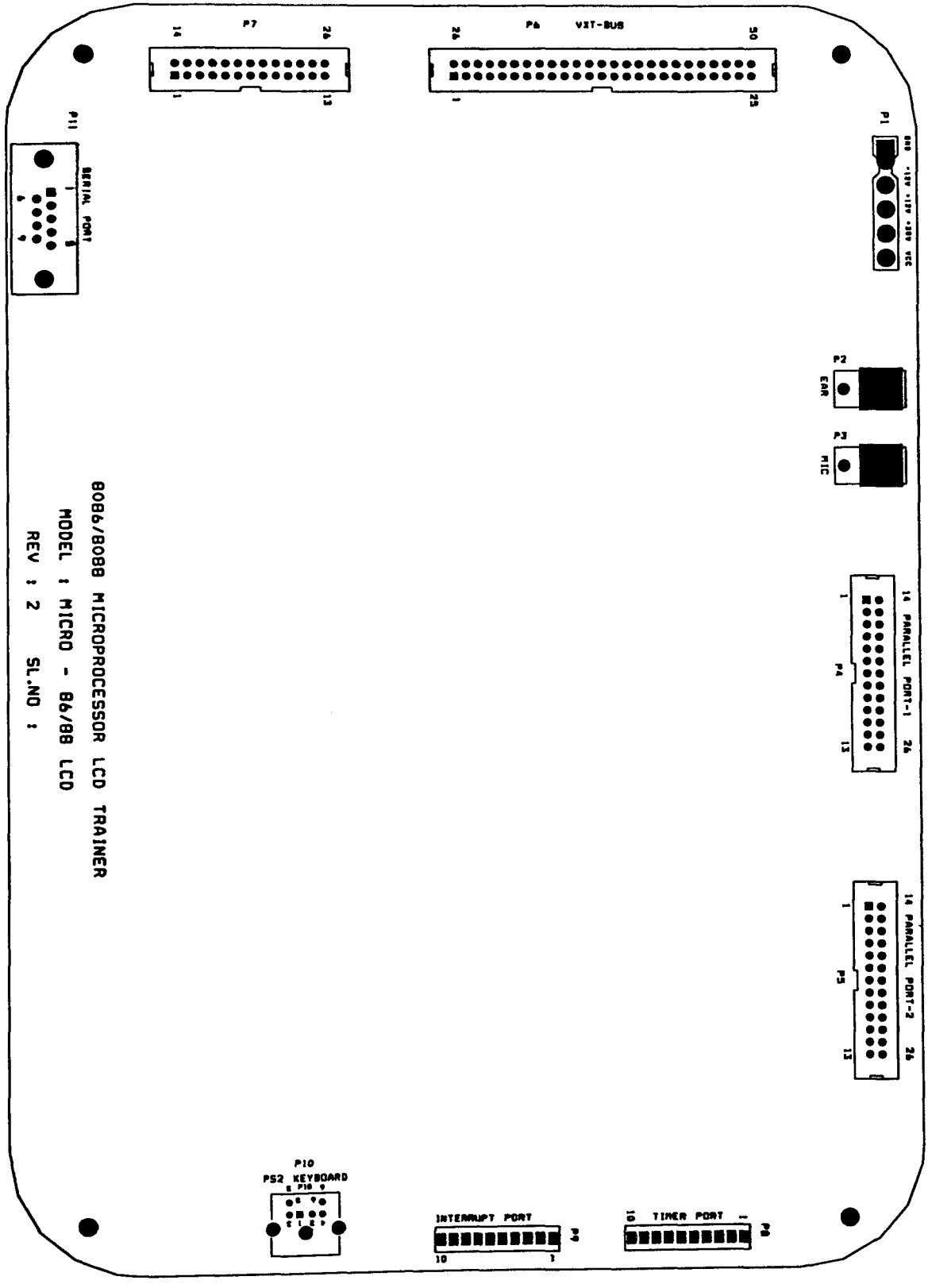
a) The signals NC, VCC and GND have common definitions for all connectors described in the succeeding sections.

NC	=	No conncetions.
VCC	=	+5V Power supply.
GND	=	0V Reference Ground.

5.2 CONNECTOR LAYOUT:

Fig F5.1 shows the positioning of the various connectors on the M - 86/88LCD trainer. All the connectors listed above are shown in the connector layout. Please refer to the figure while following the ensuing description regarding connector details. Each of the sections contains information about theconnector used, the signal description, the sginal definition and detail about themating connector to be used.

It is once again reiterated that direct soldering from any of the connectors available on thetrainer orfrom the trainer itself is not advisable. Users are requested to follow the guidelines given for matting connectors and use only the advocated matting connectors, while employing any of the peripherals on the trainer.


P7
P6 VIT-BUS
P11
SERIAL PORT
P1
P2
EAR
P3
MIC
P4
14 PARALLEL PORT-1 26
P5
14 PARALLEL PORT-2 26
P10
PS2 KEYBOARD
INTERRUPT PORT
TIMER PORT
8086/8088 MICROPROCESSOR LCD TRAINER
MODEL : MICRO - 86/88 LCD
REV : 2 SL.NO :

5.3 POWER CONNECTOR:

CONNECTOR USED:

UNICON 5 pin Male conector

- Spacing between pins 2,3,4,5 = 5mm
- Spacing between pins 1 and 2 = 7.5mm

SIGNAL DESCRIPTION:

PIN	DETAILS
1	GND
2	-12V
3	+12V
4	NC
5	VCC

MATING CONNECTOR:

UNICON 5- Pin female connector.

- With the same spacing as said above.

5.4 PARALLEL PORT CONNECTORS:(P4,P5)

CONNECTOR USED:

26 Pin IDC Male connector

- 26 pins arranged in two rows of 13 each
- pin to pin pitch distance = 2.54mm

SIGNAL DEFINITION:

PIN	DETAILS	PIN	DETAILS
1	PA0	14	PB5
2	PA1	15	PB6
3	PA2	16	PB7
4	PA3	17	PC0
5	PA4	18	PC1
6	PA5	19	PC2
7	PA6	20	PC3
8	PA7	21	PC4
9	PB0	22	PC5
10	PB1	23	PC6
11	PB2	24	PC7
12	PB3	25	GND
13	PB4	26	VCC

SIGNAL DESCRIPTION:

PA0 - PA7 = 8255 Port A I/O lines
PB0 - PB7 = 8255 port B I/O lines
PC0 - PC7 = 8255 port C I/O lines

MATTING CONNECTOR:

26 Pin Dual row connector.

- Pitch = 2.54mm
- The flat cable used should be of pitch 1.27mm

5.5 TIMER PORT CONNECTOR (P8)

CONNECTOR USED

S-401 10 Pin Male Connector

- Pins arranged in a single row.
- Pitch = 2.5 mm

SIGNAL DEFINITION

PIN	DETAILS
1	OUT1
2	OUT0
3	CLK1
4	CLK0
5	OUT2
6	CLK2
7	NC
8	NC
9	GND
10	VCC

SIGNAL DESCRIPTION

OUT0 - Channel 0 Output
CLK0 - Channel 0 Input
OUT1 - Channel 1 Output
CLK1 - Channel 2 Output
OUT2 - Channel 2 Input

MATING CONNECTOR

S - 401 10 pin Female Connector

- With the same spacing as said above.

5.6 INTERRUPT PORT CONNECTOR (P9)

CONNECTOR USED

S-401 10 Pin Male Connector

- Pins arranged in a single row.
- Pitch = 2.5 mm

SIGNAL DEFINITION

PIN	DETAILS
1	IR0
2	IR1
3	IR2
4	IR3
5	IR4
6	IR5
7	IR6
8	IR7
9	GND
10	VCC

SIGNAL DESCRIPTION

IRO - IR7 - 8 Interrupt Request Inputs to 8259.

MATING CONNECTOR

S401 10-Pin Female Connector with the same specifications.

5.7 AUDIO CONNECTORS: (P2,P3)

CONNECTOR USED:

Ear Phone Socket

- Socket Radius = 2.5mm

SIGNAL DESCRIPTION:

2.5mm Ear Phone Jack.

- The jack should be used with a two lead cable.

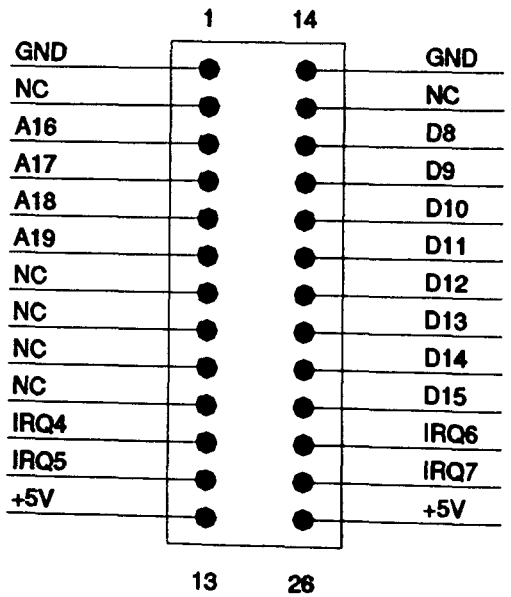
5.8 BUS EXPANSION CONNECTOR (P7)

CONNECTOR USED:

(26 pin IDC Male connector

- Pins arranged in two rows of 13 each.
- Pin to pin pitch distance = 2.54mm

SIGNAL DEFINITION:



MATING CONNECTOR:

26 Pin FRC Dual row connector

- Pitch = 2.54mm.
- The flat cable used should be of pitch 1.27mm

5.9 SERIAL PORT CONNECTOR (P7)

9 Pin D type Male connector.

- Pins arranged in two rows of 5 and 4 pins.
- Grid pitch is 2.76mm * 2.84mm
- The connector is AMPHENOL standard.

SIGNAL DEFINITION:

PIN	DETAILS
1	NC
2	TxD
3	RxD
4	RTS*
5	CTS*
6	NC
7	GND
8	NC
9	NC

SIGNAL DESCRIPTION:

TxD - Trnasmit Data

RxD - Received Data

CTS* - Clear to send

RTS* - Request to send

MATING CONNECTOR:

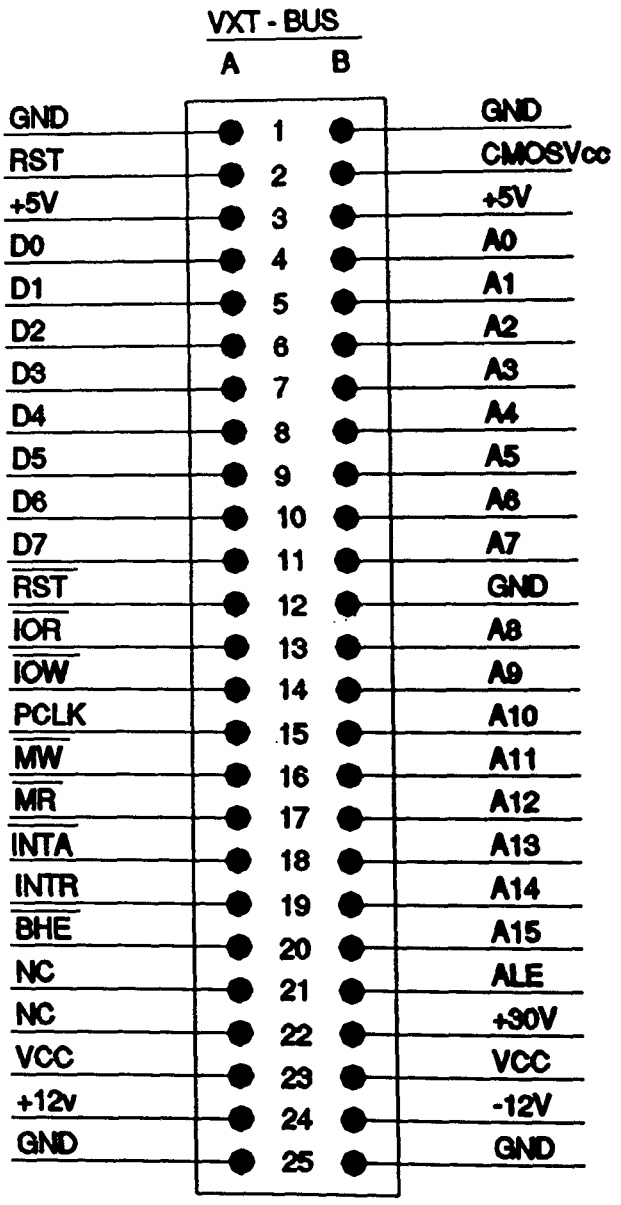
9 Position D type female connector withthe same grid pitch.

5.10 VXT BUS CONNECTOR: (P11)

CONNECTOR USED:

50 Pin Connector

- Provided for interfacing ADC / DAC / STEPPER motor interfacing boards.



SIGNAL DEFINITION:

MATING CONNECTOR:

50 Pin Connector

5.9 IBM - KEYBOARD & LCD INTERFACE CONNECTOR (LCD1 & P10)

CONNECTOR USED:

20 Pin RMC Connector

- Pin arranged in single row of 20 pins.

SIGNAL DEFINITION

PIN	DETAILS
1	GND
2	VCC
3	Voltage Input for LCD display
4	RS
5	DIOW
6	DISP
7	D0
8	D1
9	D2
10	D3
11	D4
12	D5
13	D6
14	D7
15	CS1
16	RST-
17	GND
18	CS2
19	NC
20	NC

MATING CONNECTOR

- Pitch = 2.54mm.
- The flat cable used should be of pitch 1.27mm.

CHAPTER 6:

SERIAL MONITOR

INTRODUCTION:

The serial Monitor can be invoked in two modes, namely the terminal mode based in the serial port or the CRT mode based on 6845 CRT controller using a separate Add-on Board . The syntax to invoke the serial Monitor was already given in chapter 3 "key functions". Entering SM in trainer using keyboard the serial port based serial monitor, while pressing 'A' key at command prompt invokes the CRT based serial Monitor.

The following is a simple description of the commands available with the serial monitor. The notation employed in the description makes use of certain symbolic representations, which are listed below.

XXXX	-	Denotes source start address
YYYY	-	Denotes source end address
ZZZZ	-	Denotes Destination start address
SSSS	-	Denotes segment address
PPPP	-	Denotes port address
DD	-	Denotes data type
GG	-	Denotes valid register.

"A" (ASSEMBLE) COMMAND:

FUNCTION:

This command activates the 8086 - Two pass Assembler, if present in the monitor, since it is an OPTIONAL FACILITY.

SYNTAX:

```
#A <CR>
```

If the Assembler is not present a message says "The Assembler" is available OPTIONALLY". If it is present, the editor file will be taken up for assembly.

"B" (BASIC) COMMAND:

This command invokes the GBASIC interpreter, if available.

SYNTAX:

#b <CR>

EXAMPLE:

#B<CR>

If the GBASIC Interpreter is not present, then the message, "The GBASIC INTERPRETER is available OPTIONALLY" is displayed.

If the CBASIC Interpreter is present, then the system switches to GBASIC Interpreter whose usage can be understood using the "GBASIC USER MANUAL".

"D" (DUMP) COMMAND:

FUNCTION:

This key is used to display memory contents in HEX as well as in ASCII. This function allows 5 types of syntax.

SYNTAX:

#d <CR>
#dXXXX <CR>
#dSSSS:XXXX <CR>
#dXXXX YYYY <CR>
#dSSSS : XXXX YYYY <CR>

1. The first syntax dumps 80 (hex) bytes of memory contents in the screen starting from previous address dumped. If this command is used immediately after going to the serial monitor, the monitor will display addresses and contents of the memory locations 0000 to 007f.

2. The second syntax permits specifying the starting address. This will also dump 80 (hex) characters starting from the address entered.

3. The third syntax is similar to the second one and in addition this allows the user to change the segment also. This will start dumping, taking both segment and offset from the user.

4. The fourth and fifth syntax allow the user to specify the end address also. Incase the user wants to dump from 1000 to 1010 only, the starting address and the ending address can be specified as 1000 and 1010 respectively . This syntax allows the user to dump the memory contents within a segment.

"E" (EDITOR) COMMAND:

FUNCTION:

This command activates the text editor. The file created using the Editor can be assembled only using the Two-pass Assembler provided alongwith.

SYNTAX:

```
#E  <CR>
```

"F" (FILL) COMMAND:

FUNCTION:

This command fills a block of memory with the data given.

SYNTAX:

```
#FXXXX  YYYY DD <CR>
#FSSSS : XXXX  YYYY  DD <CR>
```

Any of the above specified syntax can be used to fill a memory block with the desired data. If the specified block of memory is available as read/write memory, then the block is filled and the monitor returns to the prompt "#". Else an appropriate error message is displayed.

"G" (GO) COMMAND:

FUNCTION:

This command enables the user programs to be executed. This loads all the register with appropriate values and jumps to the location pointed by CS:IP. The user is allowed to enter the segment and offset in the command line itself. IF the user is entering only the offset then care should be taken to check that the value of "CS" is exactly matching the segment in which the user program resides.

SYNTAX:

```
#GSSSS : XXXX <CR>
#GSSSS : XXXX  YYYY <CR>
#GXXXX <CR>
#GXXXX YYYY <CR>
```

The GO command allows the break address also to be specified. But the limitation is that the user cannot enter a different address for the BREAK ADDRESS. Execution is stopped at the break address and the register contents are displayed.

"H' (HELO) COMMAND:

FUNCTION:

This command displays the help menu in the screen.

SYNTAX:

```
#H <CR>
```

A short description of all the commands in a single line is provided in the help menu.

"I" (INPUT) COMMAND:

FUNCTION:

This command reads a byte from a port.

SYNTAX:

```
#IPPPP <CR>
```

Execution of the above command causes the CPU to read from the port address specified as PPPP and the data input is displayed in the next line on the screen and the monitor returns to the prompt.

"M" (MOVE) COMMAND:

FUNCTION:

This command copies a block of memory onto another block, the destination addresses being in RAM.

SYNTAX:

```
#MXXXX YYYY YYYY <CR>
#MSSSS :  XXXX YYYY ZZZZ<CR>
#MSSSS :  XXXX YYYY SSSS : ZZZZ <CR>
```

1. The first syntax copies the contents of location DS:XXXX to DS:ZZZZ and so on upto DS:YYYY.

2. The second syntax permits specifying the segment address for the source starting address. The destination segment is the same as the source segment.

3. The third syntax is similar to the second one and in addition this allows the user to specify different segment address for the destination.

"O" (OUTPUT) COMMAND:

FUNCTION:

This command outputs a byte to a aport specified by the user.

SYNTAX:

#OPPPP DD <CR>

Execution of this command causes the data byte DD to the output to the port at address PPPP and then the monitor returns to the prompt mode "#".

"Q" (QUIT) COMMAND:

FUNCTION:

This command is used to quit the serial Monitor either from terminal mode or from the CRT mode and switch back to the keypad monitor, ie. 33 keys keyboard and 8 digit display. Before going back, the serial monitor displays a message "GOOD BYE".

SYNTAX:

```
#R <R>
#RGG <CR>
```

1. The first syntax is used to view all the registers at the same time. By doing this the monitor displays the contents of the register in the following order.:

   AX, BX, CX, DX, SP, BP, DI, CS, ES, SS, IP, FL

2. The user can vary contents of the individual register by using the second index. As per the second index "GG" can be any of the CPU registers so if the user deserves to load "AX" register with a data of "3256", then upon entering "RAX" at the prompt, the current contents of AX register is displayed and changes can be made by just typing in 3256 followed by a carriage return to confirm the data change. The register contents are altered and the monitor displays the "#" prompt in the following line. Invalid data will be rejected with an appropriate error message.

"S" (SUBSTITUTE) COMMAND:

FUNCTION:

This command used to substitute the memory contents with the desired data. This command is used to enter the programs into the memory directly.

SYNTAX:

```
#SXXXX <CR>
#SSSSS : XXXX <CR>
```

When SXXXX is entered at the prompt, the monitor reads the contents of the location XXXX and displays the data by reading the screen along with the address as DS :[XXXX] : DD. The user will have three choices now.

i) Enter the new data.
ii) Skip this location and go to the next location.
iii) Exit from the Substitute command.

If changes of data desired then type in new data and press carriage return.

If change of data is not required and if next location is to be viewed then simply press carriage return. This will not change the data in the first displayed location but displays the next location and its contents and waits for user inputs.

To Exit from the substitute command press "." (period) followed by a carriage return which will quit this command and display "#".

To cancel and entry the user can use the Backspace.

"T" (TRACE) COMMAND :

FUNCTION:

This similar to single step command in the Trainer as well as the Debug in MS-DOS. This command single steps to the program starting from the current "IP" contents. This command will execute only one instruction at any time. After executing the instruction it will display the contents of all registers. Here the registers can see that the IP will be pointing to the next instruction to be executed.

SYNTAX:

```
# t <CR>
# t XXXX <CR>
```

The user can keep tracing the program by pressing "t" continuously and this can also see, whether the register are loaded properly or not.

"U" (UNASSEMBLE) COMMAND:

SYNTAX:

```
#U <CR>
```

The Disassembler is an OPTIONAL facility and hence if you are to invoke the disassembler and if it is not present a message saying "The Disassembler is available OPTIONALLY" is displayed.

"V" (VERIFY) COMMAND:

FUNCTION:

This command is used to compare the contents of two block of memory. The output will be printed in the screen.

SYNTAX:

```
#VXXXX  YYYY  ZZZZ <CR>
#VSSSS : XXXX YYYY  ZZZZ<CR>
#VSSSS : XXXX YYYY  SSSS : ZZZZ <CR>
```

The command compares the two blocks of memory and checks the whether the data in location XXXX is equal to the data in the location ZZZZ. If both the blocks are same, in other words if the data in all the location of the source block is equal to all the data in all the locations in the destination block, then the monitor comes back to the command prompt. Otherwise the locations at which the comparison failed are displayed.

CHAPTER - 7

MONITOR SYSTEM CALLS

THIS CHAPTER IS LEFT FOR FUTURE DEVELOPMENT

APPENDIX - A

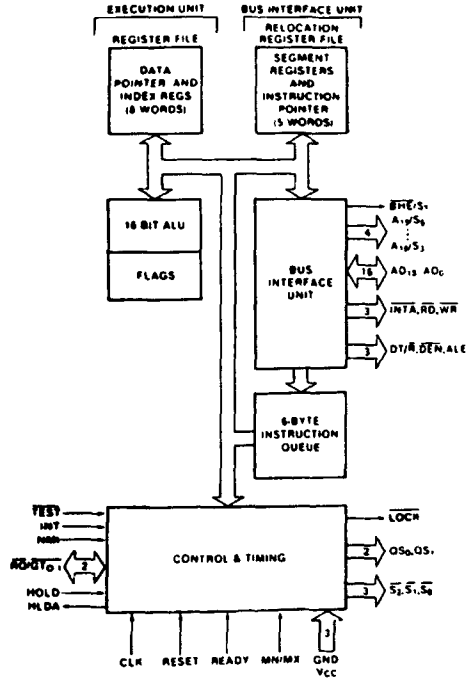
8086 DATA SHEETS

intel.

8086
16-BIT HMOS MICROPROCESSOR
8086/8086-2/8086-1

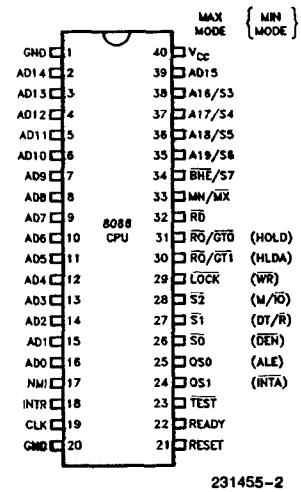
- **Direct Addressing Capability 1 MByte of Memory**
- **Architecture Designed for Powerful Assembly Language and Efficient High Level Languages**
- **14 Word, by 16-Bit Register Set with Symmetrical Operations**
- **24 Operand Addressing Modes**
- **Bit, Byte, Word, and Block Operations**
- **8 and 16-Bit Signed and Unsigned Arithmetic in Binary or Decimal Including Multiply and Divide**
- **Range of Clock Rates: 5 MHz for 8086, 8 MHz for 8086-2, 10 MHz for 8086-1**
- **MULTIBUS System Compatible Interface**
- **Available in EXPRESS**
  - **Standard Temperature Range**
  - **Extended Temperature Range**
- **Available in 40-Lead Cerdip and Plastic Package**
  (See Packaging Spec. Order #231369)

The Intel 8086 high performance 16-bit CPU is available in three clock rates: 5, 8 and 10 MHz. The CPU is implemented in N-Channel, depletion load, silicon gate technology (HMOS-III), and packaged in a 40-pin CERDIP or plastic package. The 8086 operates in both single processor and multiple processor configurations to achieve high performance levels.



231455-1

Figure 1. 8086 CPU Block Diagram



231455-2

40 Lead

Figure 2. 8086 Pin Configuration

Table 1. Pin Description

The following pin function descriptions are for 8086 systems in either minimum or maximum mode. The "Local Bus" in these descriptions is the direct multiplexed bus interface connection to the 8086 (without regard to additional bus buffers).

Symbol	Pin No.	Type	Name and Function
AD_{15}–AD_0	2–16, 39	I/O	**ADDRESS DATA BUS:** These lines constitute the time multiplexed memory/IO address (T_1), and data (T_2, T_3, T_W, T_4) bus. A_0 is analogous to $\overline{BHE}$ for the lower byte of the data bus, pins D_7–D_0. It is LOW during T_1 when a byte is to be transferred on the lower portion of the bus in memory or I/O operations. Eight-bit oriented devices tied to the lower half would normally use A_0 to condition chip select functions. (See $\overline{BHE}$.) These lines are active HIGH and float to 3-state OFF during interrupt acknowledge and local bus "hold acknowledge".
A_{19}/S_6, A_{18}/S_5, A_{17}/S_4, A_{16}/S_3	35–38	O	**ADDRESS/STATUS:** During T_1 these are the four most significant address lines for memory operations. During I/O operations these lines are LOW. During memory and I/O operations, status information is available on these lines during T_2, T_3, T_W, T_4. The status of the interrupt enable FLAG bit (S_5) is updated at the beginning of each CLK cycle. A_{17}/S_4 and A_{16}/S_3 are encoded as shown. This information indicates which relocation register is presently being used for data accessing. These lines float to 3-state OFF during local bus "hold acknowledge."

A_{17}/S_4	A_{16}/S_3	Characteristics
0 (LOW)	0	Alternate Data
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data
S_6 is 0 (LOW)		

$\overline{BHE}/S_7$	34	O	**BUS HIGH ENABLE/STATUS:** During T_1 the bus high enable signal ($\overline{BHE}$) should be used to enable data onto the most significant half of the data bus, pins D_{15}–D_8. Eight-bit oriented devices tied to the upper half of the bus would normally use $\overline{BHE}$ to condition chip select functions. $\overline{BHE}$ is LOW during T_1 for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. The S_7 status information is available during T_2, T_3, and T_4. The signal is active LOW, and floats to 3-state OFF in "hold". It is LOW during T_1 for the first interrupt acknowledge cycle.

$\overline{BHE}$	A_0	Characteristics
0	0	Whole word
0	1	Upper byte from/to odd address
1	0	Lower byte from/to even address
1	1	None

$\overline{RD}$	32	O	**READ:** Read strobe indicates that the processor is performing a memory or I/O read cycle, depending on the state of the S_2 pin. This signal is used to read devices which reside on the 8086 local bus. $\overline{RD}$ is active LOW during T_2, T_3 and T_W of any read cycle, and is guaranteed to remain HIGH in T_2 until the 8086 local bus has floated. This signal floats to 3-state OFF in "hold acknowledge".

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
READY	22	I	**READY:** is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The READY signal from memory/IO is synchronized by the 8284A Clock Generator to form READY. This signal is active HIGH. The 8086 READY input is not synchronized. Correct operation is not guaranteed if the setup and hold times are not met.
INTR	18	I	**INTERRUPT REQUEST:** is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.
$\overline{TEST}$	23	I	**$\overline{TEST}$:** input is examined by the "Wait" instruction. If the $\overline{TEST}$ input is LOW execution continues, otherwise the processor waits in an "Idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.
NMI	17	I	**NON-MASKABLE INTERRUPT:** an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.
RESET	21	I	**RESET:** causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the Instruction Set description, when RESET returns LOW. RESET is internally synchronized.
CLK	19	I	**CLOCK:** provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.
V_{CC}	40		**V_{CC}:** +5V power supply pin.
GND	1, 20		**GROUND**
MN/$\overline{MX}$	33	I	**MINIMUM/MAXIMUM:** indicates what mode the processor is to operate in. The two modes are discussed in the following sections.

The following pin function descriptions are for the 8086/8288 system in maximum mode (i.e., MN/$\overline{MX}$ = V_{SS}). Only the pin functions which are unique to maximum mode are described; all other pin functions are as described above.

$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$	26–28	O	**STATUS:** active during T_4, T_1, and T_2 and is returned to the passive state (1, 1, 1) during T_3 or during T_W when READY is HIGH. This status is used by the 8288 Bus Controller to generate all memory and I/O access control signals. Any change by $\overline{S_2}$, $\overline{S_1}$, or $\overline{S_0}$ during T_4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T_3 or T_W is used to indicate the end of a bus cycle.

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ (Continued)	26–28	O	These signals float to 3-state OFF in "hold acknowledge". These status lines are encoded as shown.

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Characteristics
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	Halt
1 (HIGH)	0	0	Code Access
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive

Symbol	Pin No.	Type	Name and Function
$\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$	30, 31	I/O	**REQUEST/GRANT:** pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}/\overline{GT_0}$ having higher priority than $\overline{RQ}/\overline{GT_1}$. $\overline{RQ}/\overline{GT}$ pins have internal pull-up resistors and may be left unconnected. The request/grant sequence is as follows (see Page 2-24): 1. A pulse of 1 CLK wide from another local bus master indicates a local bus request ("hold") to the 8086 (pulse 1). 2. During a T_4 or T_1 clock cycle, a pulse 1 CLK wide from the 8086 to the requesting master (pulse 2), indicates that the 8086 has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge". 3. A pulse 1 CLK wide from the requesting master indicates to the 8086 (pulse 3) that the "hold" request is about to end and that the 8086 can reclaim the local bus at the next CLK. Each master-master exchange of the local bus is a sequence of 3 pulses. There must be one dead CLK cycle after each bus exchange. Pulses are active LOW. If the request is made while the CPU is performing a memory cycle, it will release the local bus during T_4 of the cycle when all the following conditions are met: 1. Request occurs on or before T_2. 2. Current cycle is not the low byte of a word (on an odd address). 3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence. 4. A locked instruction is not currently executing. If the local bus is idle when the request is made the two possible events will follow: 1. Local bus will be released during the next clock. 2. A memory cycle will start within 3 clocks. Now the four rules for a currently active memory cycle apply with condition number 1 already satisfied.
$\overline{LOCK}$	29	O	**$\overline{LOCK}$:** output indicates that other system bus masters are not to gain control of the system bus while $\overline{LOCK}$ is active LOW. The $\overline{LOCK}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active LOW, and floats to 3-state OFF in "hold acknowledge".

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
QS_1, QS_0	24, 25	O	**QUEUE STATUS:** The queue status is valid during the CLK cycle after which the queue operation is performed. QS_1 and QS_0 provide status to allow external tracking of the internal 8086 instruction queue.

QS_1	QS_0	Characteristics
0 (LOW)	0	No Operation
0	1	First Byte of Op Code from Queue
1 (HIGH)	0	Empty the Queue
1	1	Subsequent Byte from Queue

The following pin function descriptions are for the 8086 in minimum mode (i.e., MN/$\overline{MX}$ = V_{CC}). Only the pin functions which are unique to minimum mode are described; all other pin functions are as described above.

M/$\overline{IO}$	28	O	**STATUS LINE:** logically equivalent to S_2 in the maximum mode. It is used to distinguish a memory access from an I/O access. M/$\overline{IO}$ becomes valid in the T_4 preceding a bus cycle and remains valid until the final T_4 of the cycle (M = HIGH, IO = LOW). M/$\overline{IO}$ floats to 3-state OFF in local bus "hold acknowledge".
$\overline{WR}$	29	O	**WRITE:** indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the M/$\overline{IO}$ signal. $\overline{WR}$ is active for T_2, T_3 and T_W of any write cycle. It is active LOW, and floats to 3-state OFF in local bus "hold acknowledge".
$\overline{INTA}$	24	O	**$\overline{INTA}$:** is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T_2, T_3 and T_W of each interrupt acknowledge cycle.
ALE	25	O	**ADDRESS LATCH ENABLE:** provided by the processor to latch the address into the 8282/8283 address latch. It is a HIGH pulse active during T_1 of any bus cycle. Note that ALE is never floated.
DT/$\overline{R}$	27	O	**DATA TRANSMIT/RECEIVE:** needed in minimum system that desires to use an 8286/8287 data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically DT/$\overline{R}$ is equivalent to $\overline{S_1}$ in the maximum mode, and its timing is the same as for M/$\overline{IO}$. (T = HIGH, R = LOW.) This signal floats to 3-state OFF in local bus "hold acknowledge".
$\overline{DEN}$	26	O	**DATA ENABLE:** provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver. $\overline{DEN}$ is active LOW during each memory and I/O access and for INTA cycles. For a read or $\overline{INTA}$ cycle it is active from the middle of T_2 until the middle of T_4, while for a write cycle it is active from the beginning of T_2 until the middle of T_4. $\overline{DEN}$ floats to 3-state OFF in local bus "hold acknowledge".
HOLD, HLDA	31, 30	I/O	**HOLD:** indicates that another master is requesting a local bus "hold." To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement in the middle of a T_4 or T_I clock cycle. Simultaneous with the issuance of HLDA the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor will LOWer the HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. Hold acknowledge (HLDA) and HOLD have internal pull-up resistors. The same rules as for $\overline{RQ}$/$\overline{GT}$ apply regarding when the local bus will be released. HOLD is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the setup time.

FUNCTIONAL DESCRIPTION

General Operation

The internal functions of the 8086 processor are partitioned logically into two processing units. The first is the Bus Interface Unit (BIU) and the second is the Execution Unit (EU) as shown in the block diagram of Figure 1.

These units can interact directly but for the most part perform as separate asynchronous operational processors. The bus interface unit provides the functions related to instruction fetching and queuing, operand fetch and store, and address relocation. This unit also provides the basic bus control. The overlap of instruction pre-fetching provided by this unit serves to increase processor performance through improved bus bandwidth utilization. Up to 6 bytes of the instruction stream can be queued while waiting for decoding and execution.

The instruction stream queuing mechanism allows the BIU to keep the memory utilized very efficiently. Whenever there is space for at least 2 bytes in the queue, the BIU will attempt a word fetch memory cycle. This greatly reduces "dead time" on the memory bus. The queue acts as a First-In-First-Out (FIFO) buffer, from which the EU extracts instruction bytes as required. If the queue is empty (following a branch instruction, for example), the first byte into the queue immediately becomes available to the EU.

The execution unit receives pre-fetched instructions from the BIU queue and provides un-relocated operand addresses to the BIU. Memory operands are passed through the BIU for processing by the EU, which passes results to the BIU for storage. See the Instruction Set description for further register set and architectural descriptions.

MEMORY ORGANIZATION

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory is logically divided into code, data, extra data, and stack segments of up to 64K bytes each, with each segment falling on 16-byte boundaries. (See Figure 3a.)

All memory references are made relative to base addresses contained in high speed segment registers. The segment types were chosen based on the addressing needs of programs. The segment register to be selected is automatically chosen according to the rules of the following table. All information in one segment type share the same logical attributes (e.g. code or data). By structuring memory into relocatable areas of similar characteristics and by automatically selecting segment registers, programs are shorter, faster, and more structured.

Word (16-bit) operands can be located on even or odd address boundaries and are thus not constrained to even boundaries as is the case in many 16-bit computers. For address and data operands, the least significant byte of the word is stored in the lower valued address location and the most significant byte in the next higher address location. The BIU automatically performs the proper number of memory accesses, one if the word operand is on an even byte boundary and two if it is on an odd byte boundary. Except for the performance penalty, this double access is transparent to the software. This performance penalty does not occur for instruction fetches, only word operands.

Physically, the memory is organized as a high bank (D_{15}–D_8) and a low bank (D_7–D_0) of 512K 8-bit bytes addressed in parallel by the processor's address lines A_{19}–A_1. Byte data with even addresses is transferred on the D_7–D_0 bus lines while odd addressed byte data (A_0 HIGH) is transferred on the D_{15}–D_8 bus lines. The processor provides two enable signals, $\overline{BHE}$ and A_0, to selectively allow reading from or writing into either an odd byte location, even byte location, or both. The instruction stream is fetched from memory as words and is addressed internally by the processor to the byte level as necessary.

Memory Reference Need	Segment Register Used	Segment Selection Rule
Instructions	CODE (CS)	Automatic with all instruction prefetch.
Stack	STACK (SS)	All stack pushes and pops. Memory references relative to BP base register except data references.
Local Data	DATA (DS)	Data references when: relative to stack, destination of string operation, or explicitly overridden.
External (Global) Data	EXTRA (ES)	Destination of string operations: explicitly selected using a segment override.

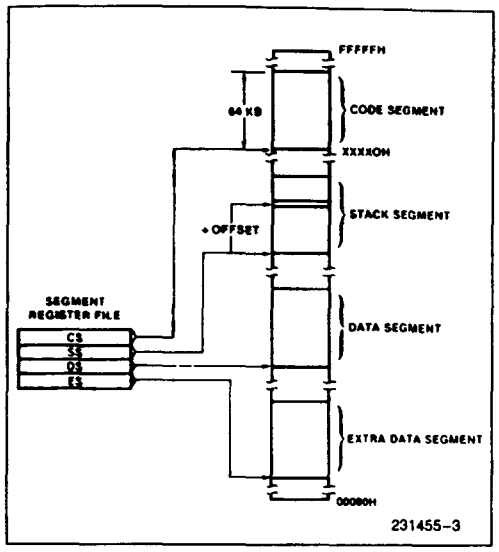


Figure 3a. Memory Organization

In referencing word data the BIU requires one or two memory cycles depending on whether or not the starting byte of the word is on an even or odd address, respectively. Consequently, in referencing word operands performance can be optimized by locating data on even address boundaries. This is an especially useful technique for using the stack, since odd address references to the stack may adversely affect the context switching time for interrupt processing or task multiplexing.

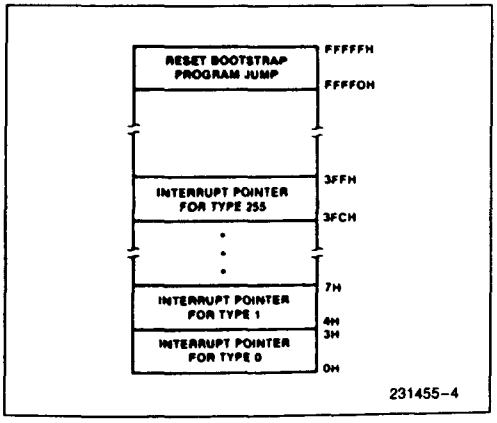


Figure 3b. Reserved Memory Locations

Certain locations in memory are reserved for specific CPU operations (see Figure 3b). Locations from address FFFF0H through FFFFFH are reserved for operations including a jump to the initial program loading routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be. Locations 00000H through 003FFH are reserved for interrupt operations. Each of the 256 possible interrupt types has its service routine pointed to by a 4-byte pointer element consisting of a 16-bit segment address and a 16-bit offset address. The pointer elements are assumed to have been stored at the respective places in reserved memory prior to occurrence of interrupts.

MINIMUM AND MAXIMUM MODES

The requirements for supporting minimum and maximum 8086 systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 8086 is equipped with a strap pin (MN/$\overline{MX}$) which defines the system configuration. The definition of a certain subset of the pins changes dependent on the condition of the strap pin. When MN/$\overline{MX}$ pin is strapped to GND, the 8086 treats pins 24 through 31 in maximum mode. An 8288 bus controller interprets status information coded into $\overline{S_0}$, $\overline{S_2}$, $\overline{S_2}$ to generate bus timing and control signals compatible with the MULTIBUS architecture. When the MN/$\overline{MX}$ pin is strapped to V_{CC}, the 8086 generates bus control signals itself on pins 24 through 31, as shown in parentheses in Figure 2. Examples of minimum mode and maximum mode systems are shown in Figure 4.

BUS OPERATION

The 8086 has a combined address and data bus commonly referred to as a time multiplexed bus. This technique provides the most efficient use of pins on the processor while permitting the use of a standard 40-lead package. This "local bus" can be buffered directly and used throughout the system with address latching provided on memory and I/O modules. In addition, the bus can also be demultiplexed at the processor with a single set of address latches if a standard non-multiplexed bus is desired for the system.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T_1, T_2, T_3 and T_4 (see Figure 5). The address is emitted from the processor during T_1 and data transfer occurs on the bus during T_3 and T_4. T_2 is used primarily for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "Wait" states (T_W) are inserted between T_3 and T_4. Each inserted "Wait" state is of the same duration as a CLK cycle. Periods

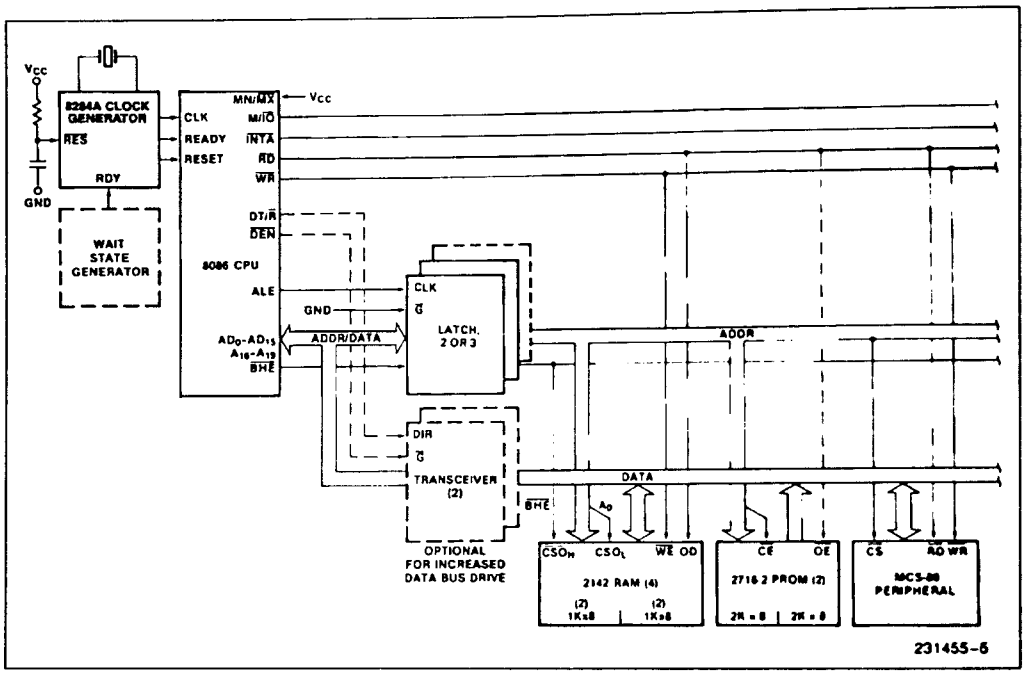


Figure 4a. Minimum Mode 8086 Typical Configuration

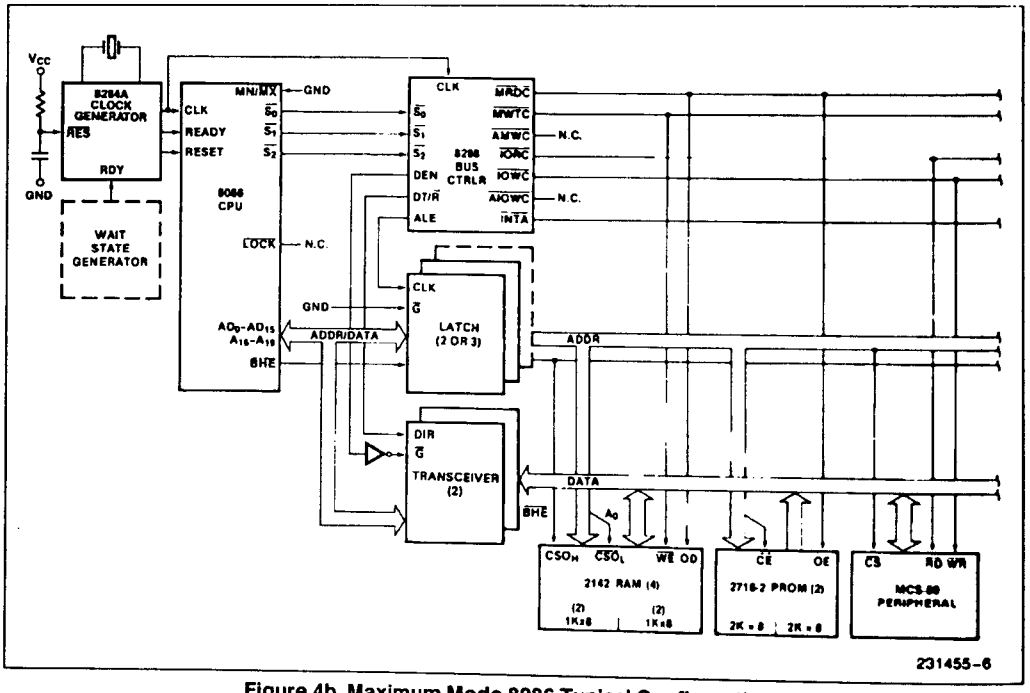


Figure 4b. Maximum Mode 8086 Typical Configuration

can occur between 8086 bus cycles. These are referred to as "Idle" states (T_i) or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

During T_1 of any bus cycle the ALE (Address Latch Enable) signal is emitted (by either the processor or the 8288 bus controller, depending on the MN/$\overline{MX}$ strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

Status bits $\overline{S_0}$, $\overline{S_1}$, and $\overline{S_2}$ are used, in maximum mode, by the bus controller to identify the type of bus transaction according to the following table:

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Characteristics
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

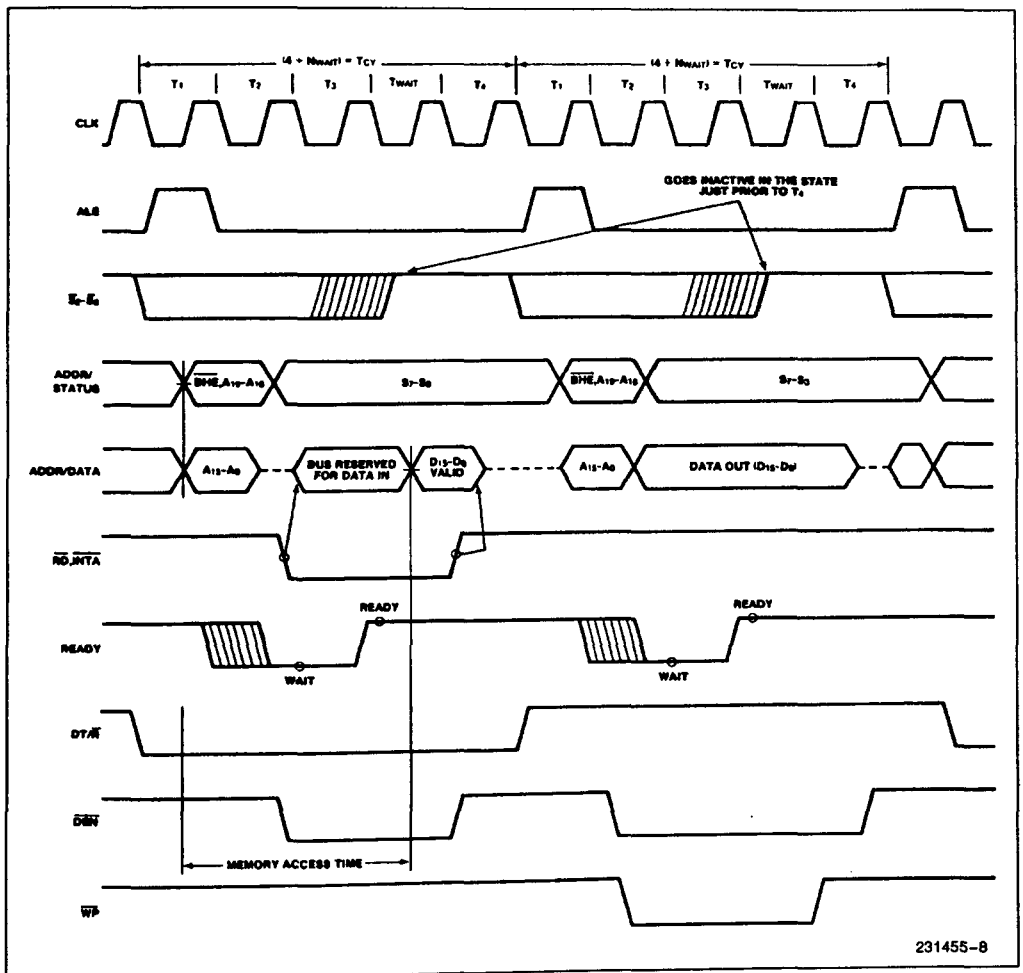


Figure 5. Basic System Timing

Status bits S_3 through S_7 are multiplexed with high-order address bits and the $\overline{BHE}$ signal, and are therefore valid during T_2 through T_4. S_3 and S_4 indicate which segment register (see Instruction Set description) was used for this bus cycle in forming the address, according to the following table:

S_4	S_3	Characteristics
0 (LOW)	0	Alternate Data (extra segment)
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

S_5 is a reflection of the PSW interrupt enable bit. S_6 = 0 and S_7 is a spare status bit.

I/O ADDRESSING

In the 8086, I/O operations can address up to a maximum of 64K I/O byte registers or 32K I/O word registers. The I/O address appears in the same format as the memory address on bus lines A_{15}–A_0. The address lines A_{19}–A_{16} are zero in I/O operations. The variable I/O instructions which use register DX as a pointer have full address capability while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space.

I/O ports are addressed in the same manner as memory locations. Even addressed bytes are transferred on the D_7–D_0 bus lines and odd addressed bytes on D_{15}–D_8. Care must be taken to assure that each register within an 8-bit peripheral located on the lower portion of the bus be addressed as even.

External Interface

PROCESSOR RESET AND INITIALIZATION

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 8086 RESET is required to be HIGH for greater than 4 CLK cycles. The 8086 will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 10 CLK cycles. After this interval the 8086 operates normally beginning with the instruction in absolute location FFFF0H (see Figure 3b). The details of this operation are specified in the Instruction Set description of the MCS-86 Family User's Manual. The RESET input is internally synchronized to the processor clock. At initialization the HIGH-to-LOW transition of RESET must occur no sooner than 50 μs after power-up, to allow complete initialization of the 8086.

NMI asserted prior to the 2nd clock after the end of RESET will not be honored. If NMI is asserted after that point and during the internal reset sequence, the processor may execute one instruction before responding to the interrupt. A hold request active immediately after RESET will be honored before the first instruction fetch.

All 3-state outputs float to 3-state OFF during RESET. Status is active in the idle state for the first clock after RESET becomes active and then floats to 3-state OFF. ALE and HLDA are driven low.

INTERRUPT OPERATIONS

Interrupt operations fall into two classes; software or hardware initiated. The software initiated interrupts and software aspects of hardware interrupts are specified in the Instruction Set description. Hardware interrupts can be classified as non-maskable or maskable.

Interrupts result in a transfer of control to a new program location. A 256-element table containing address pointers to the interrupt service program locations resides in absolute locations 0 through 3FFH (see Figure 3b), which are reserved for this purpose. Each element in the table is 4 bytes in size and corresponds to an interrupt "type". An interrupting device supplies an 8-bit type number, during the interrupt acknowledge sequence, which is used to "vector" through the appropriate element to the new interrupt service program location.

NON-MASKABLE INTERRUPT (NMI)

The processor provides a single non-maskable interrupt pin (NMI) which has higher priority than the maskable interrupt request pin (INTR). A typical use would be to activate a power failure routine. The NMI is edge-triggered on a LOW-to-HIGH transition. The activation of this pin causes a type 2 interrupt. (See Instruction Set description.)

NMI is required to have a duration in the HIGH state of greater than two CLK cycles, but is not required to be synchronized to the clock. Any high-going transition of NMI is latched on-chip and will be serviced at the end of the current instruction or between whole moves of a block-type instruction. Worst case response to NMI would be for multiply, divide, and variable shift instructions. There is no specification on the occurrence of the low-going edge; it may occur before, during, or after the servicing of NMI. Another high-going edge triggers another response if it occurs after the start of the NMI procedure. The signal must be free of logical spikes in general and be free of bounces on the low-going edge to avoid triggering extraneous responses.

MASKABLE INTERRUPT (INTR)

The 8086 provides a single interrupt request input (INTR) which can be masked internally by software with the resetting of the interrupt enable FLAG status bit. The interrupt request signal is level triggered. It is internally synchronized during each clock cycle on the high-going edge of CLK. To be responded to, INTR must be present (HIGH) during the clock period preceding the end of the current instruction or the end of a whole move for a block-type instruction. During the interrupt response sequence further interrupts are disabled. The enable bit is reset as part of the response to any interrupt (INTR, NMI, software interrupt or single-step), although the FLAGS register which is automatically pushed onto the stack reflects the state of the processor prior to the interrupt. Until the old FLAGS register is restored the enable bit will be zero unless specifically set by an instruction.

During the response sequence (Figure 6) the processor executes two successive (back-to-back) interrupt acknowledge cycles. The 8086 emits the LOCK signal from T_2 of the first bus cycle until T_2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle. In the second bus cycle a byte is fetched from the external interrupt system (e.g., 8259A PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector lookup table. An INTR signal left HIGH will be continually responded to within the limitations of the enable bit and sample period. The INTERRUPT RETURN instruction includes a FLAGS pop which returns the status of the original interrupt enable bit when it restores the FLAGS.

HALT

When a software "HALT" instruction is executed the processor indicates that it is entering the "HALT" state in one of two ways depending upon which mode is strapped. In minimum mode, the processor issues one ALE with no qualifying bus control signals. In maximum mode, the processor issues appropriate HALT status on $\overline{S}_2$, $\overline{S}_1$, and $\overline{S}_0$; and the 8288 bus controller issues one ALE. The 8086 will not leave the "HALT" state when a local bus "hold" is entered while in "HALT". In this case, the processor reissues the HALT indicator. An interrupt request or RESET will force the 8086 out of the "HALT" state.

READ/MODIFY/WRITE (SEMAPHORE) OPERATIONS VIA LOCK

The $\overline{\text{LOCK}}$ status information is provided by the processor when directly consecutive bus cycles are required during the execution of an instruction. This provides the processor with the capability of performing read/modify/write operations on memory (via the Exchange Register With Memory instruction, for example) without the possibility of another system bus master receiving intervening memory cycles. This is useful in multi-processor system configurations to accomplish "test and set lock" operations. The $\overline{\text{LOCK}}$ signal is activated (forced LOW) in the clock cycle following the one in which the software "LOCK" prefix instruction is decoded by the EU. It is deactivated at the end of the last bus cycle of the instruction following the "LOCK" prefix instruction. While $\overline{\text{LOCK}}$ is active a request on a RQ/GT pin will be recorded and then honored at the end of the LOCK.

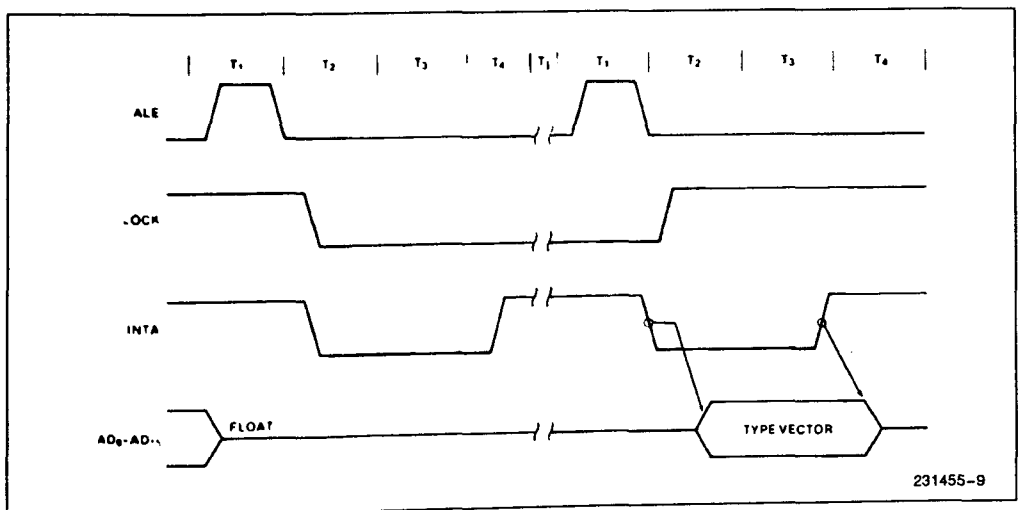


Figure 6. Interrupt Acknowledge Sequence

EXTERNAL SYNCHRONIZATION VIA TEST

As an alternative to the interrupts and general I/O capabilities, the 8086 provides a single software-testable input known as the $\overline{TEST}$ signal. At any time the program may execute a WAIT instruction. If at that time the $\overline{TEST}$ signal is inactive (HIGH), program execution becomes suspended while the processor waits for $\overline{TEST}$ to become active. It must remain active for at least 5 CLK cycles. The WAIT instruction is re-executed repeatedly until that time. This activity does not consume bus cycles. The processor remains in an idle state while waiting. All 8086 drivers go to 3-state OFF if bus "Hold" is entered. If interrupts are enabled, they may occur while the processor is waiting. When this occurs the processor fetches the WAIT instruction one extra time, processes the interrupt, and then re-fetches and re-executes the WAIT instruction upon returning from the interrupt.

Basic System Timing

Typical system configurations for the processor operating in minimum mode and in maximum mode are shown in Figures 4a and 4b, respectively. In minimum mode, the MN/$\overline{MX}$ pin is strapped to V_{CC} and the processor emits bus control signals in a manner similar to the 8085. In maximum mode, the MN/$\overline{MX}$ pin is strapped to V_{SS} and the processor emits coded status information which the 8288 bus controller uses to generate MULTIBUS compatible bus control signals. Figure 5 illustrates the signal timing relationships.

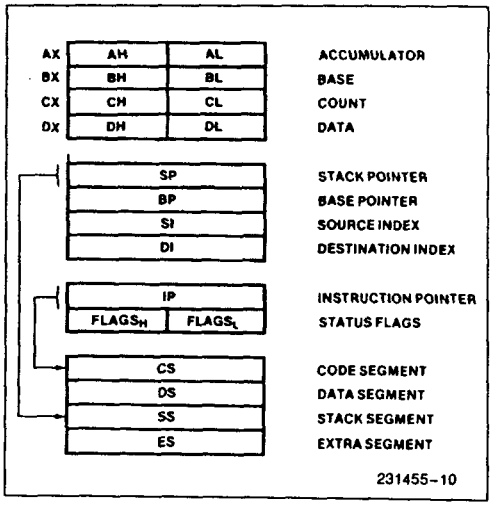


Figure 7. 8086 Register Model

SYSTEM TIMING—MINIMUM SYSTEM

The read cycle begins in T_1 with the assertion of the Address Latch Enable (ALE) signal. The trailing (low-going) edge of this signal is used to latch the address information, which is valid on the local bus at this time, into the address latch. The $\overline{BHE}$ and A_0 signals address the low, high, or both bytes. From T_1 to T_4 the M/$\overline{IO}$ signal indicates a memory or I/O operation. At T_2 the address is removed from the local bus and the bus goes to a high impedance state. The read control signal is also asserted at T_2. The read ($\overline{RD}$) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later valid data will be available on the bus and the addressed device will drive the READY line HIGH. When the processor returns the read signal to a HIGH level, the addressed device will again 3-state its bus drivers. If a transceiver is required to buffer the 8086 local bus, signals DT/$\overline{R}$ and $\overline{DEN}$ are provided by the 8086.

A write cycle also begins with the assertion of ALE and the emission of the address. The M/$\overline{IO}$ signal is again asserted to indicate a memory or I/O write operation. In the T_2 immediately following the address emission the processor emits the data to be written into the addressed location. This data remains valid until the middle of T_4. During T_2, T_3, and T_W the processor asserts the write control signal. The write ($\overline{WR}$) signal becomes active at the beginning of T_2 as opposed to the read which is delayed somewhat into T_2 to provide time for the bus to float.

The $\overline{BHE}$ and A_0 signals are used to select the proper byte(s) of the memory/IO word to be read or written according to the following table:

$\overline{BHE}$	A0	Characteristics
0	0	Whole word
0	1	Upper byte from/to odd address
1	0	Lower byte from/to even address
1	1	None

I/O ports are addressed in the same manner as memory location. Even addressed bytes are transferred on the D_7–D_0 bus lines and odd addressed bytes on D_{15}–D_8.

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge signal ($\overline{INTA}$) is asserted in place of the read ($\overline{RD}$) signal and the address bus is floated. (See Figure 6.) In the second of two successive INTA cycles, a byte of information is read from bus

lines D_7–D_0 as supplied by the inerrupt system logic (i.e., 8259A Priority Interrupt Controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into an interrupt vector lookup table, as described earlier.

BUS TIMING—MEDIUM SIZE SYSTEMS

For medium size systems the MN/$\overline{MX}$ pin is connected to V_{SS} and the 8288 Bus Controller is added to the system as well as a latch for latching the system address, and a transceiver to allow for bus loading greater than the 8086 is capable of handling. Signals ALE, DEN, and DT/$\overline{R}$ are generated by the 8288 instead of the processor in this configuration although their timing remains relatively the same. The 8086 status outputs ($\overline{S_2}$, $\overline{S_1}$, and $\overline{S_0}$) provide type-of-cycle information and become 8288 inputs. This bus cycle information specifies read (code, data, or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 8288 thus issues control signals specifying memory read or write, I/O read or write, or interrupt acknowledge. The 8288 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence data isn't valid at the leading edge of write. The transceiver receives the usual DIR and $\overline{G}$ inputs from the 8288's DT/$\overline{R}$ and DEN.

The pointer into the interrupt vector table, which is passed during the second INTA cycle, can derive from an 8259A located on either the local bus or the system bus. If the master 8259A Priority Interrupt Controller is positioned on the local bus, a TTL gate is required to disable the transceiver when reading from the master 8259A during the interrupt acknowledge sequence and software "poll".

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin with
Respect to Ground.....−1.0V to +7V
Power Dissipation.....2.5W

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS

(8086: T_A = 0°C to 70°C, V_{CC} = 5V ±10%)
(8086-1: T_A = 0°C to 70°C, V_{CC} = 5V ±5%)
(8086-2: T_A = 0°C to 70°C, V_{CC} = 5V ±5%)

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	+0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.5	V	(Notes 1, 2)
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = − 400 μA
I_{CC}	Power Supply Current: 8086 8086-1 8086-2		340 360 350	mA	T_A = 25°C
I_{LI}	Input Leakage Current		±10	μA	0V ≤ V_{IN} ≤ V_{CC} (Note 3)
I_{LO}	Output Leakage Current		±10	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
V_{CL}	Clock Input Low Voltage	−0.5	+0.6	V	
V_{CH}	Clock Input High Voltage	3.9	V_{CC} + 1.0	V	
C_{IN}	Capacitance of Input Buffer (All input except AD_0–AD_{15}, $\overline{RQ}/\overline{GT}$)		15	pF	fc = 1 MHz
C_{IO}	Capacitance of I/O Buffer (AD_0–AD_{15}, $\overline{RQ}/\overline{GT}$)		15	pF	fc = 1 MHz

NOTES:
1. V_{IL} tested with MN/$\overline{MX}$ Pin = 0V. V_{IH} tested with MN/$\overline{MX}$ Pin = 5V. MN/$\overline{MX}$ Pin is a Strap Pin.
2. Not applicable to $\overline{RQ}/\overline{GT0}$ and $\overline{RQ}/\overline{GT1}$ (Pins 30 and 31).
3. HOLD and HLDA I_{LI} min = 30 μA, max = 500 μA.

A.C. CHARACTERISTICS

(8086: T_A = 0°C to 70°C, V_{CC} = 5V ± 10%)
(8086-1: T_A = 0°C to 70°C, V_{CC} = 5V ± 5%)
(8086-2: T_A = 0°C to 70°C, V_{CC} = 5V ± 5%)

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

Symbol	Parameter	8086		8086-1		8086-2		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TCLCL	CLK Cycle Period	200	500	100	500	125	500	ns	
TCLCH	CLK Low Time	118		53		68		ns	
TCHCL	CLK High Time	69		39		44		ns	
TCH1CH2	CLK Rise Time		10		10		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10		10		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	30		5		20		ns	
TCLDX	Data in Hold Time	10		10		10		ns	
TR1VCL	RDY Setup Time into 8284A (See Notes 1, 2)	35		35		35		ns	
TCLR1X	RDY Hold Time into 8284A (See Notes 1, 2)	0		0		0		ns	
TRYHCH	READY Setup Time into 8086	118		53		68		ns	
TCHRYX	READY Hold Time into 8086	30		20		20		ns	
TRYLCL	READY Inactive to CLK (See Note 3)	−8		−10		−8		ns	
THVCH	HOLD Setup Time	35		20		20		ns	
TINVCH	INTR, NMI, TEST Setup Time (See Note 2)	30		15		15		ns	
TILIH	Input Rise Time (Except CLK)		20		20		20	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		12		12		12	ns	From 2.0V to 0.8V

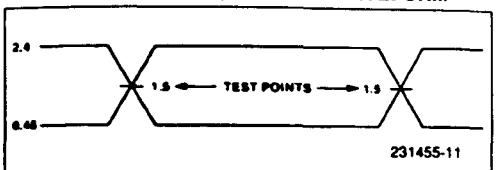
A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

Symbol	Parameter	8086		8086-1		8086-2		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TCLAV	Address Valid Delay	10	110	10	50	10	60	ns	
TCLAX	Address Hold Time	10		10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	10	40	TCLAX	50	ns	
TLHLL	ALE Width	TCLCH-20		TCLCH-10		TCLCH-10		ns	
TCLLH	ALE Active Delay		80		40		50	ns	
TCHLL	ALE Inactive Delay		85		45		55	ns	
TLLAX	Address Hold Time	TCHCL-10		TCHCL-10		TCHCL-10		ns	
TCLDV	Data Valid Delay	10	110	10	50	10	60	ns	*C_L = 20–100 pF for all 8086 Outputs (In addition to 8086 selfload)
TCHDX	Data Hold Time	10		10		10		ns	
TWHDX	Data Hold Time After WR	TCLCH-30		TCLCH-25		TCLCH-30		ns	
TCVCTV	Control Active Delay 1	10	110	10	50	10	70	ns	
TCHCTV	Control Active Delay 2	10	110	10	45	10	60	ns	
TCVCTX	Control Inactive Delay	10	110	10	50	10	70	ns	
TAZRL	Address Float to READ Active	0		0		0		ns	
TCLRL	$\overline{RD}$ Active Delay	10	165	10	70	10	100	ns	
TCLRH	$\overline{RD}$ Inactive Delay	10	150	10	60	10	80	ns	
TRHAV	$\overline{RD}$ Inactive to Next Address Active	TCLCL-45		TCLCL-35		TCLCL-40		ns	
TCLHAV	HLDA Valid Delay	10	160	10	60	10	100	ns	
TRLRH	$\overline{RD}$ Width	2TCLCL-75		2TCLCL-40		2TCLCL-50		ns	
TWLWH	$\overline{WR}$ Width	2TCLCL-60		2TCLCL-35		2TCLCL-40		ns	
TAVAL	Address Valid to ALE Low	TCLCH-60		TCLCH-35		TCLCH-40		ns	
TOLOH	Output Rise Time		20		20		20	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		12		12		12	ns	From 2.0V to 0.8V

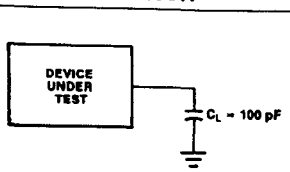
NOTES:
1. Signal at 8284A shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T2 state. (8 ns into T3).

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 1.5V for both a Logic "1" and "0".

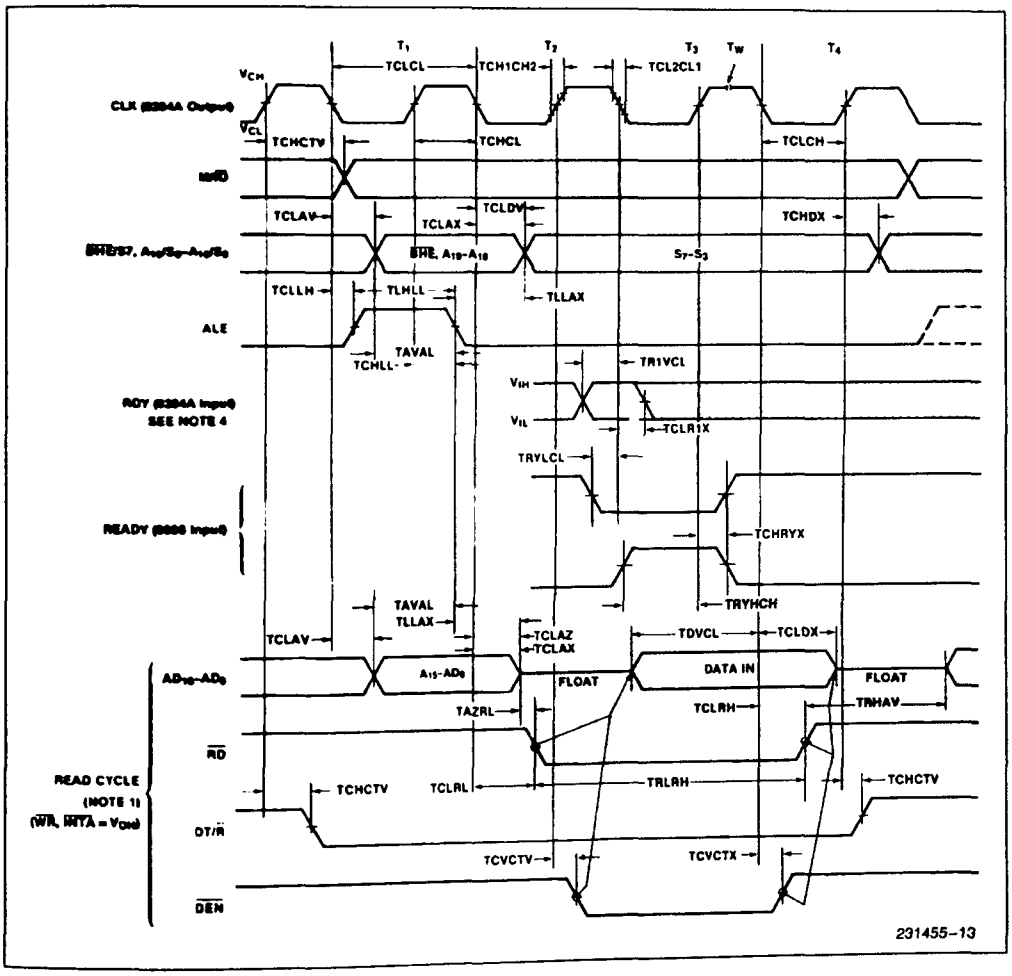
A.C. TESTING LOAD CIRCUIT



C_L Includes Jig Capacitance

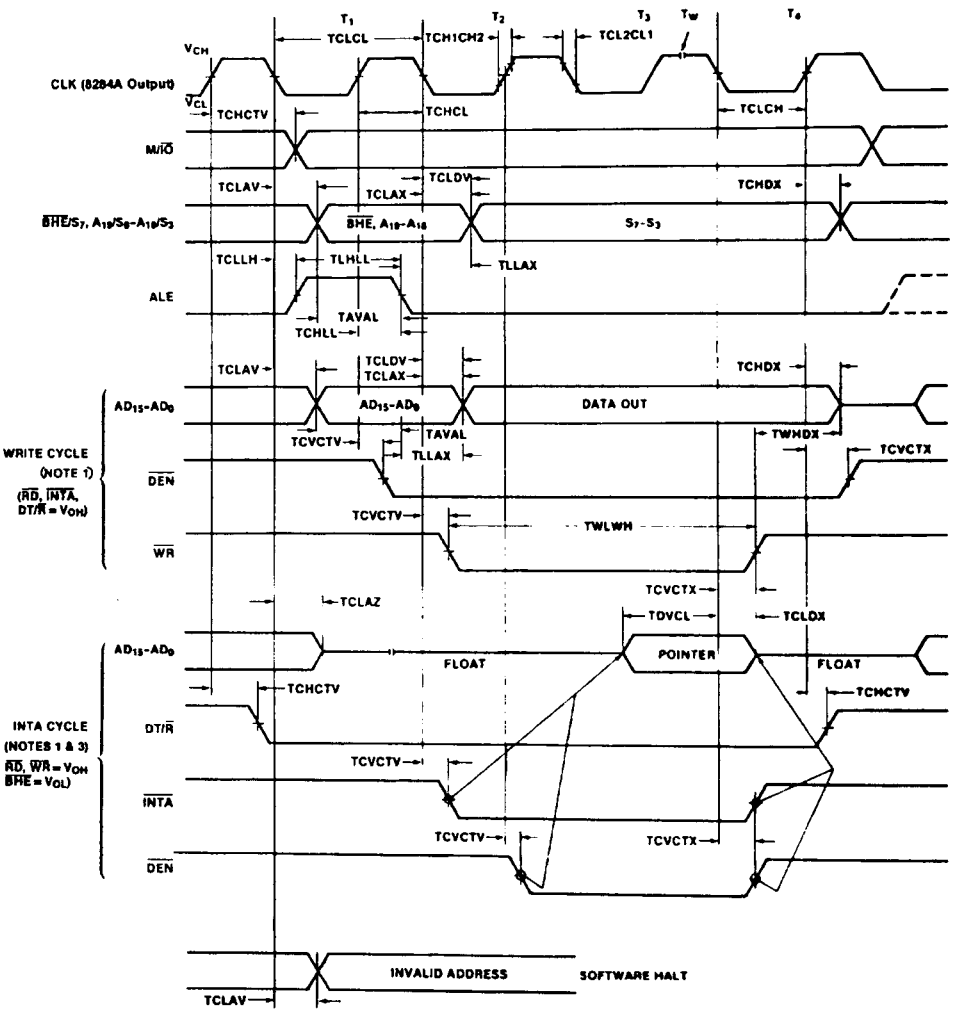
WAVEFORMS

MINIMUM MODE



WAVEFORMS (Continued)

MINIMUM MODE (Continued)



SOFTWARE HALT—
RD, WR, INTA = V_{OH}
DT/R = INDETERMINATE

231455-14

NOTES:
1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Two INTA cycles run back-to-back. The 8086 LOCAL ADDR/DATA BUS is floating during both INTA cycles. Control signals shown for second INTA cycle.
4. Signals at 8284A are shown for reference only.
5. All timing measurements are made at 1.5V unless otherwise noted.

A.C. CHARACTERISTICS

MAX MODE SYSTEM (USING 8288 BUS CONTROLLER)
TIMING REQUIREMENTS

Symbol	Parameter	8086		8086-1		8086-2		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TCLCL	CLK Cycle Period	200	500	100	500	125	500	ns	
TCLCH	CLK Low Time	118		53		68		ns	
TCHCL	CLK High Time	69		39		44		ns	
TCH1CH2	CLK Rise Time		10		10		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10		10		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	30		5		20		ns	
TCLDX	Data in Hold Time	10		10		10		ns	
TR1VCL	RDY Setup Time into 8284A (Notes 1, 2)	35		35		35		ns	
TCLR1X	RDY Hold Time into 8284A (Notes 1, 2)	0		0		0		ns	
TRYHCH	READY Setup Time into 8086	118		53		68		ns	
TCHRYX	READY Hold Time into 8086	30		20		20		ns	
TRYLCL	READY Inactive to CLK (Note 4)	−8		−10		−8		ns	
TINVCH	Setup Time for Recognition (INTR, NMI, $\overline{TEST}$) (Note 2)	30		15		15		ns	
TGVCH	$\overline{RQ}/\overline{GT}$ Setup Time (Note 5)	30		15		15		ns	
TCHGX	$\overline{RQ}$ Hold Time into 8086	40		20		30		ns	
TILIH	Input Rise Time (Except CLK)		20		20		20	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		12		12		12	ns	From 2.0V to 0.8V

A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

Symbol	Parameter	8086		8086-1		8086-2		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TCLML	Command Active Delay (See Note 1)	10	35	10	35	10	35	ns	
TCLMH	Command Inactive Delay (See Note 1)	10	35	10	35	10	35	ns	
TRYHSH	READY Active to Status Passive (See Note 3)		110		45		65	ns	
TCHSV	Status Active Delay	10	110	10	45	10	60	ns	
TCLSH	Status Inactive Delay	10	130	10	55	10	70	ns	
TCLAV	Address Valid Delay	10	110	10	50	10	60	ns	
TCLAX	Address Hold Time	10		10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	10	40	TCLAX	50	ns	
TSVLH	Status Valid to ALE High (See Note 1)		15		15		15	ns	
TSVMCH	Status Valid to MCE High (See Note 1)		15		15		15	ns	
TCLLH	CLK Low to ALE Valid (See Note 1)		15		15		15	ns	C_L = 20–100 pF for all 8086 Outputs (In addition to 8086 self-load)
TCLMCH	CLK Low to MCE High (See Note 1)		15		15		15	ns	
TCHLL	ALE Inactive Delay (See Note 1)		15		15		15	ns	
TCLMCL	MCE Inactive Delay (See Note 1)		15		15		15	ns	
TCLDV	Data Valid Delay	10	110	10	50	10	60	ns	
TCHDX	Data Hold Time	10		10		10		ns	
TCVNV	Control Active Delay (See Note 1)	5	45	5	45	5	45	ns	
TCVNX	Control Inactive Delay (See Note 1)	10	45	10	45	10	45	ns	
TAZRL	Address Float to READ Active	0		0		0		ns	
TCLRL	RD Active Delay	10	165	10	70	10	100	ns	
TCLRH	RD Inactive Delay	10	150	10	60	10	80	ns	

A.C. CHARACTERISTICS (Continued)

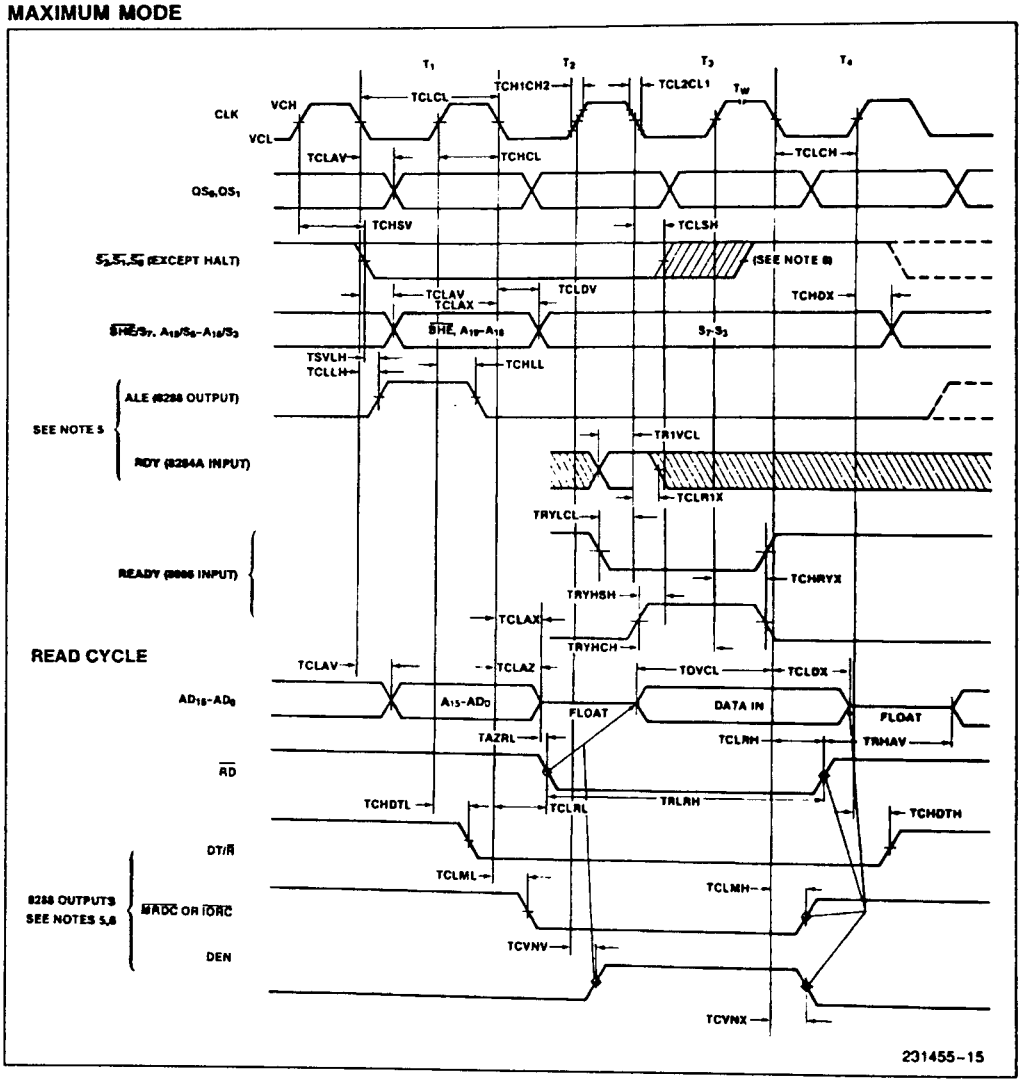
TIMING RESPONSES (Continued)

Symbol	Parameter	8086		8086-1		8086-2		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TRHAV	RD Inactive to Next Address Active	TCLCL-45		TCLCL-35		TCLCL-40		ns	C_L = 20–100 pF for all 8086 Outputs (In addition to 8086 self-load)
TCHDTL	Direction Control Active Delay (Note 1)		50		50		50	ns	
TCHDTH	Direction Control Inactive Delay (Note 1)		30		30		30	ns	
TCLGL	GT Active Delay	0	85	0	38	0	50	ns	
TCLGH	GT Inactive Delay	0	85	0	45	0	50	ns	
TRLRH	RD Width	2TCLCL-75		2TCLCL-40		2TCLCL-50		ns	
TOLOH	Output Rise Time		20		20		20	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		12		12		12	ns	From 2.0V to 0.8V

NOTES:
1. Signal at 8284A or 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).

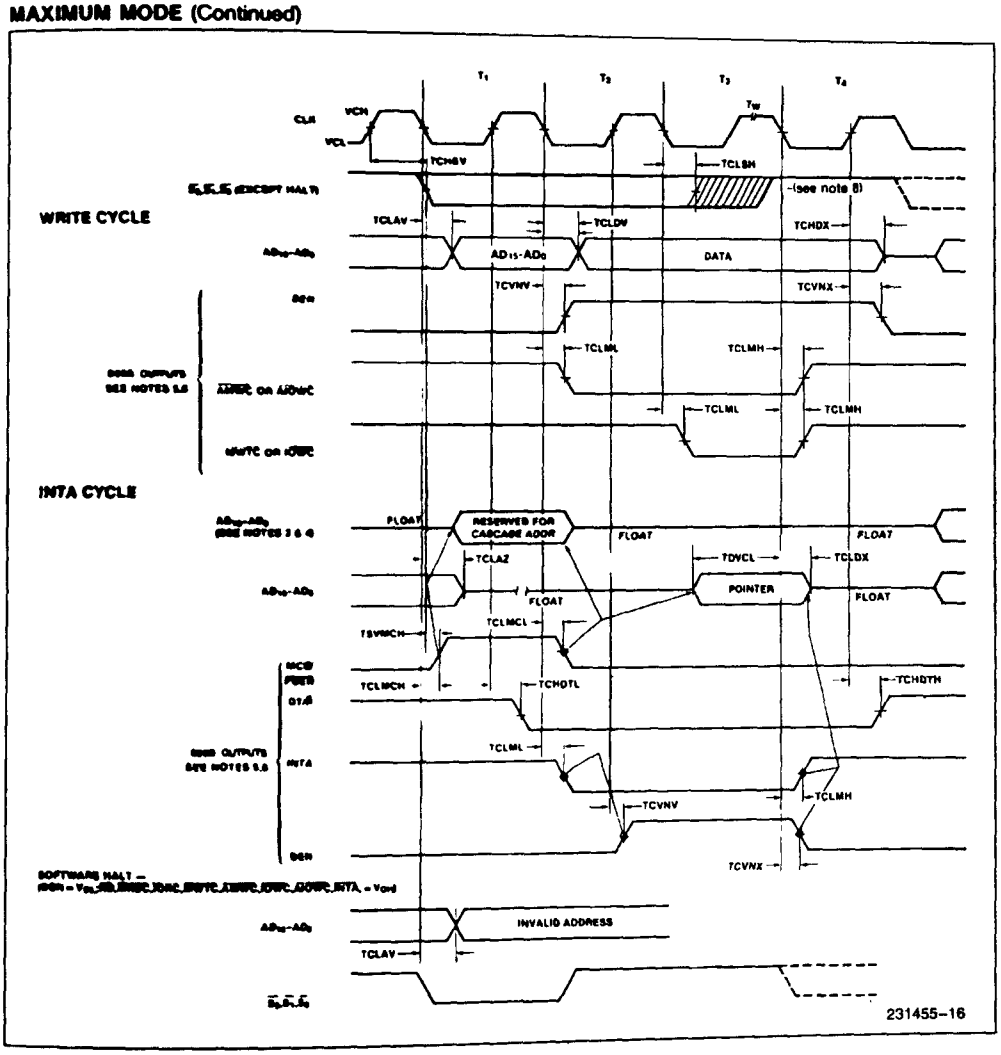
WAVEFORMS

MAXIMUM MODE



WAVEFORMS (Continued)

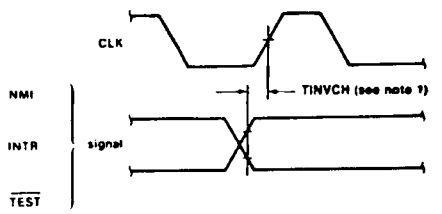
MAXIMUM MODE (Continued)



NOTES:
1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Cascade address is valid between first and second INTA cycle.
4. Two INTA cycles run back-to-back. The 8086 LOCAL ADDR/DATA BUS is floating during both INTA cycles. Control for pointer address is shown for second INTA cycle.
5. Signals at 8284A or 8288 are shown for reference only.
6. The issuance of the 8288 command and control signals ($\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$, $\overline{INTA}$ and DEN) lags the active high 8288 CEN.
7. All timing measurements are made at 1.5V unless otherwise noted.
8. Status inactive in state just prior to T_4.

WAVEFORMS (Continued)

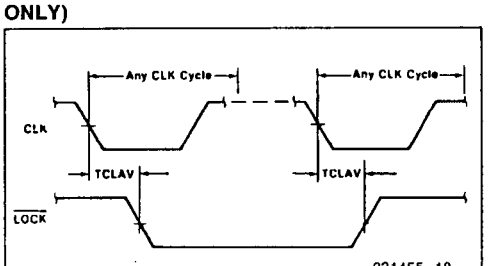
ASYNCHRONOUS SIGNAL RECOGNITION



231455-17

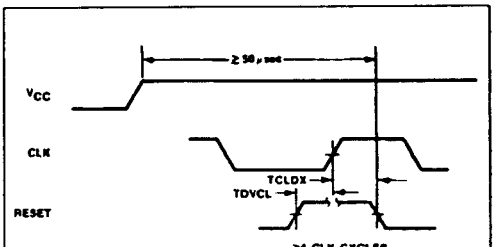
NOTE:
1. Setup requirements for asynchronous signals only to guarantee recognition at next CLK.

BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)



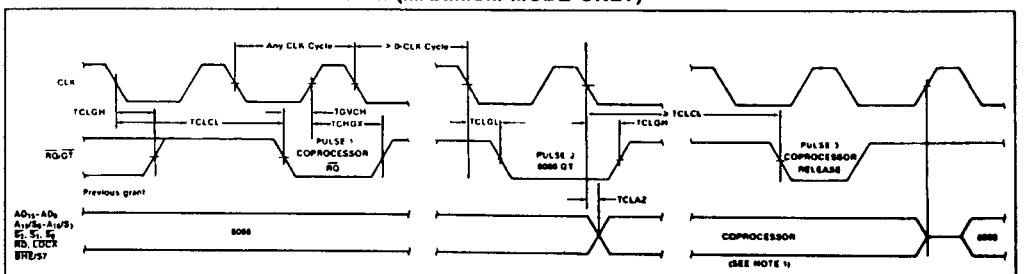
231455-18

RESET TIMING



231455-19

REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)



231455-20

NOTE:
The coprocessor may not drive the buses outside the region shown without risking contention.

WAVEFORMS (Continued)

HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)

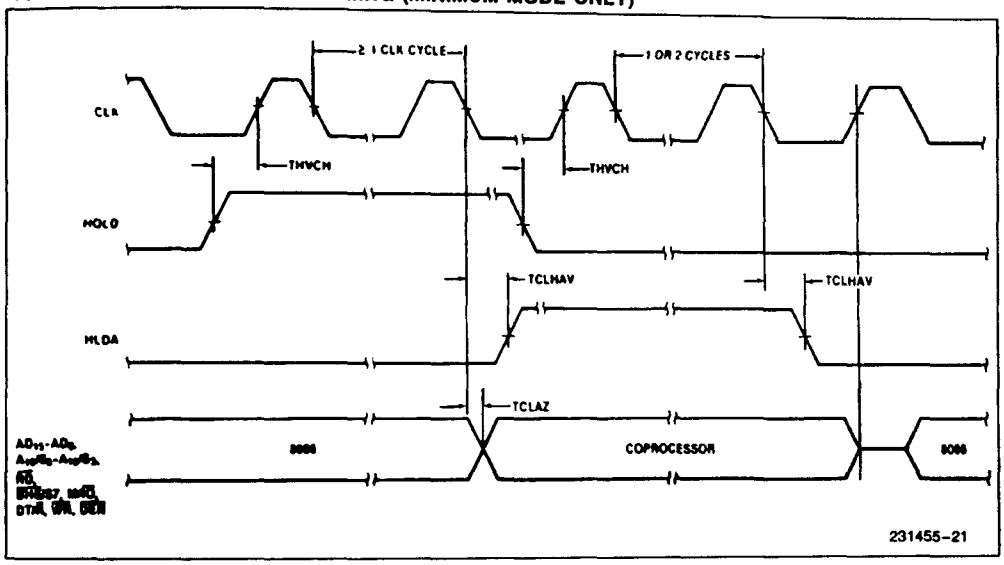


Table 2. Instruction Set Summary

Mnemonic and Description	Instruction Code			
DATA TRANSFER				
MOV = Move:	76543210	76543210	76543210	76543210
Register/Memory to/from Register	100010dw	mod reg r/m		
Immediate to Register/Memory	1100011w	mod 000 r/m	data	data if w = 1
Immediate to Register	1011 w reg	data	data if w = 1	
Memory to Accumulator	1010000w	addr-low	addr-high	
Accumulator to Memory	1010001w	addr-low	addr-high	
Register/Memory to Segment Register	10001110	mod 0 reg r/m		
Segment Register to Register/Memory	10001100	mod 0 reg r/m		
PUSH = Push:				
Register/Memory	11111111	mod 110 r/m		
Register	01010 reg			
Segment Register	000 reg 110			
POP = Pop:				
Register/Memory	10001111	mod 000 r/m		
Register	01011 reg			
Segment Register	000 reg 111			
XCHG = Exchange:				
Register/Memory with Register	1000011w	mod reg r/m		
Register with Accumulator	10010 reg			
IN = Input from:				
Fixed Port	1110010w	port		
Variable Port	1110110w			
OUT = Output to:				
Fixed Port	1110011w	port		
Variable Port	1110111w			
XLAT = Translate Byte to AL	11010111			
LEA = Load EA to Register	10001101	mod reg r/m		
LDS = Load Pointer to DS	11000101	mod reg r/m		
LES = Load Pointer to ES	11000100	mod reg r/m		
LAHF = Load AH with Flags	10011111			
SAHF = Store AH into Flags	10011110			
PUSHF = Push Flags	10011100			
POPF = Pop Flags	10011101			

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
ARITHMETIC	76543210	76543210	76543210	76543210
ADD = Add:				
Reg./Memory with Register to Either	000000dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 000 r/m	data	data if s: w = 01
Immediate to Accumulator	0000010w	data	data if w = 1	
ADC = Add with Carry:				
Reg./Memory with Register to Either	000100dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 010 r/m	data	data if s: w = 01
Immediate to Accumulator	0001010w	data	data if w = 1	
INC = Increment:				
Register/Memory	1111111w	mod 000 r/m		
Register	01000reg			
AAA = ASCII Adjust for Add	00110111			
BAA = Decimal Adjust for Add	00100111			
SUB = Subtract:				
Reg./Memory and Register to Either	001010dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 101 r/m	data	data if s w = 01
Immediate from Accumulator	0010110w	data	data if w = 1	
SSB = Subtract with Borrow				
Reg./Memory and Register to Either	000110dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 011 r/m	data	data if s w = 01
Immediate from Accumulator	0001111w	data	data if w = 1	
DEC = Decrement:				
Register/memory	1111111w	mod 001 r/m		
Register	01001 reg			
NEG = Change sign	1111011w	mod 011 r/m		
CMP = Compare:				
Register/Memory and Register	001110dw	mod reg r/m		
Immediate with Register/Memory	100000sw	mod 111 r/m	data	data if s w = 01
Immediate with Accumulator	0011110w	data	data if w = 1	
AAS = ASCII Adjust for Subtract	00111111			
DAS = Decimal Adjust for Subtract	00101111			
MUL = Multiply (Unsigned)	1111011w	mod 100 r/m		
IMUL = Integer Multiply (Signed)	1111011w	mod 101 r/m		
AAM = ASCII Adjust for Multiply	11010100	00001010		
DIV = Divide (Unsigned)	1111011w	mod 110 r/m		
IDIV = Integer Divide (Signed)	1111011w	mod 111 r/m		
AAD = ASCII Adjust for Divide	11010101	00001010		
CBW = Convert Byte to Word	10011000			
CWD = Convert Word to Double Word	10011001			

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
LOGIC	76543210	76543210	76543210	76543210
NOT = Invert	1111011w	mod010r/m		
SHL/SAL = Shift Logical/Arithmetic Left	110100vw	mod100r/m		
SHR = Shift Logical Right	110100vw	mod101r/m		
SAR = Shift Arithmetic Right	110100vw	mod111r/m		
ROL = Rotate Left	110100vw	mod000r/m		
ROR = Rotate Right	110100vw	mod001r/m		
RCL = Rotate Through Carry Flag Left	110100vw	mod010r/m		
RCR = Rotate Through Carry Right	110100vw	mod011r/m		
AND = And:				
Reg./Memory and Register to Either	001000dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod100r/m	data	data if w = 1
Immediate to Accumulator	0010010w	data	data if w = 1	
TEST = And Function to Flags, No Result:				
Register/Memory and Register	1000010w	mod reg r/m		
Immediate Data and Register/Memory	1111011w	mod000r/m	data	data if w = 1
Immediate Data and Accumulator	1010100w	data	data if w = 1	
OR = Or:				
Reg./Memory and Register to Either	000010dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod001r/m	data	data if w = 1
Immediate to Accumulator	0000110w	data	data if w = 1	
XOR = Exclusive or:				
Reg./Memory and Register to Either	001100dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod110r/m	data	data if w = 1
Immediate to Accumulator	0011010w	data	data if w = 1	
STRING MANIPULATION				
REP = Repeat	1111001z			
MOVS = Move Byte/Word	1010010w			
CMPS = Compare Byte/Word	1010011w			
SCAS = Scan Byte/Word	1010111w			
LODS = Load Byte/Wd to AL/AX	1010110w			
STOS = Stor Byte/Wd from AL/A	1010101w			
CONTROL TRANSFER				
CALL = Call:				
Direct within Segment	11101000	disp-low	disp-high	
Indirect within Segment	11111111	mod010r/m		
Direct Intersegment	10011010	offset-low	offset-high	
		seg-low	seg-high	
Indirect Intersegment	11111111	mod011r/m		

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code		
JMP = Unconditional Jumps	76543210	76543210	76543210
Direct within Segment	11101001	disp-low	disp-high
Direct within Segment-Short	11101011	disp	
Indirect within Segment	11111111	mod 1 0 0 r/m	
Direct Intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	11111111	mod 1 0 1 r/m	
RET = Return from CALL:			
Within Segment	11000011		
Within Seg Adding Immed to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment Adding Immediate to SP	11001010	data-low	data-high
JE/JZ = Jump on Equal/Zero	01110100	disp	
JL/JNGE = Jump on Less/Not Greater or Equal	01111100	disp	
JLE/JNG = Jump on Less or Equal/ Not Greater	01111110	disp	
JB/JNAE = Jump on Below/Not Above or Equal	01110010	disp	
JBE/JNA = Jump on Below or Equal/ Not Above	01110110	disp	
JP/JPE = Jump on Parity/Parity Even	01111010	disp	
JO = Jump on Overflow	01110000	disp	
JS = Jump on Sign	01111000	disp	
JNE/JNZ = Jump on Not Equal/Not Zero	01110101	disp	
JNL/JGE = Jump on Not Less/Greater or Equal	01111101	disp	
JNLE/JG = Jump on Not Less or Equal/ Greater	01111111	disp	
JNB/JAE = Jump on Not Below/Above or Equal	01110011	disp	
JNBE/JA = Jump on Not Below or Equal/Above	01110111	disp	
JNP/JPO = Jump on Not Par/Par Odd	01111011	disp	
JNO = Jump on Not Overflow	01110001	disp	
JNS = Jump on Not Sign	01111001	disp	
LOOP = Loop CX Times	11100010	disp	
LOOPZ/LOOPE = Loop While Zero/Equal	11100001	disp	
LOOPNZ/LOOPNE = Loop While Not Zero/Equal	11100000	disp	
JCXZ = Jump on CX Zero	11100011	disp	
INT = Interrupt			
Type Specified	11001101	type	
Type 3	11001100		
INTO = Interrupt on Overflow	11001110		
IRET = Interrupt Return	11001111		

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code	
	76543210	76543210
PROCESSOR CONTROL		
CLC = Clear Carry	11111000	
CMC = Complement Carry	11110101	
STC = Set Carry	11111001	
CLD = Clear Direction	11111100	
STD = Set Direction	11111101	
CLI = Clear Interrupt	11111010	
STI = Set Interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to External Device)	11011xxx	mod x x x r/m
LOCK = Bus Lock Prefix	11110000	

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 = 1 then "count" in (CL)
x = don't care
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)

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -004 data sheet. Please review this summary carefully.

1. The Intel 8086 implementation technology (HMOS) has been changed to (HMOS-III).
2. Delete all "changes from 1985 Handbook Specification" sentences.

8086 Instruction Set Summary

The 8086 instruction set can be broken into some simple categories.

Transport Commands

- IN - Read data from input ports
- OUT - Write data to output ports
- MOV - Copy data from source to destination
- LAHF - Copy flags to AH register
- SAHF - Copy to flags from AH to flags
- PUSH - Copy to stack
- POP - Copy and remove from stack (Restore from stack)
- PUSHF - Copy flags to stack
- POPF - Restore flags from stack
- XCHG - Exchange two items
- XLAT - Byte translation from table pointed to by BX
- LEA - Load effective address into pointer
- LDS - Load DS and pointer
- LES - Load ES and pointer

String Commands (modify pointer based upon size and direction flag)

- MOVSB - Move string bytes
- MOVSW - Move string words
- CMPSB - Compare string bytes
- CMPSW - Compare string words
- SCASB - Scan for a byte in a string
- SCASW - Scan for a word in a string
- LODSB - Load AL from a string
- LODSW - Load AX from a string
- STOSB - Store byte from AL in string
- STOSW - Store word from AX in string

Arithmetic Commands

- ADD - Add
- ADC - Add with carry in
- AAA - ASCII adjust after addition
- DAA - BCD adjust after addition
- INC - Increment
- SUB - Subtract
- SBB - Subtract with borrow
- AAS - ASCII adjust after subtraction
- DAS - BCD adjust after subtraction
- DEC - Decrement
- NEG - Negate
- CMP - Compare
- MUL - Signed multiply
- IMUL - Unsigned multiply
- DIV - Signed divide
- IDIV - Unsigned divide
- AAM - ASCII adjust after multiplication
- AAD - ASCII adjust before division

- CBW - Sign extend byte to word
- CWD - Sign extend word to double

Bit Manipulation

- AND - Bit by bit logical AND
- OR - Bit by bit logical OR
- XOR - Bit by bit logical exclusive or
- NOT - Invert all bits
- TEST - Non-destructive AND
- SHL - Shift unsigned bit pattern left
- SAL - Shift signed bit pattern left
- SHR - Shift unsigned bit pattern right
- SAR - Shift signed bit pattern right
- ROL - Rotate bit pattern to the left
- ROR - Rotate bit pattern to the right
- RCL - Rotate bit pattern to the left including the C bit
- RCR - Rotate bit pattern to the right including the C bit

Program Control

- CALL - Call subroutine (variants are NEAR and FAR)
- RET - Return from subroutine
- INT - Software interrupt
- INTO - Interrupt on overflow
- IRET - Return from interrupt
- LOOP - Repeat section of code if CX not zero
- LOOPE or LOOPZ - Repeat isection of code f CX not zero and Z flag set
- LOOPNE or LOOPNZ - Repeat section of code if CX not zero and Z flag clear
- JMP - Go to label or address provided
- JCXZ - Go to label if CX equal zero
- JG or JNLE - Jump to label if signed result greater than, or not less than or equal
- JGE or JNL - Jump to label if signed result greater than or equal, or not less than
- JE or JZ - Jump to label if equal
- JLE or JNG - Jump to label if signed result less than or equal, or not greater than
- JL or JNGE - Jump to label if signed result less than, or not greater than or equal
- JA or JNBE - Jump to label if unsigned result above, or not below or equal
- JAE or JNB - Jump to label if unsigned result above or equal, or not below
- JBE or JNA - Jump to label if unsigned result below or equal, or not above
- JB or JNAE - Jump to label if unsigned result below, or not above or equal
- JC - Jump to label if carry set
- JNC - Jump to label if carry clear
- JO - Jump to label if overflow
- JNO - Jump to label if no overflow
- JP or JPE - Jump to label if parity set (even parity)
- JNP or JPO - Jump to label if parity clear (odd parity)
- JS - Jump to label if negative (sign)
- JNS - Jump to label if positive (no sign)

Miscellaneous

- STD - Autodecrement pointer in string commands
- CLD - Autoincrement pointer in string commands
- STC - Set carry flag

- CLC - Clear carry flag
- CMC - Complement carry flag
- STI - Set interrupt flag (enable interrupts)
- CLI - Clear interrupt flag (disable interrupts)
- HLT - Wait until interrupt
- WAIT - Wait for /TEST pin to go low
- ESC - Give bus to coprocessor
- LOCK - Prevent bus contention during next instruction
- NOP - No operation (do nothing - a pad instruction)

Last Modified June 23, 1997 by Kurt Nalty
Back to MP1 Schedule
Back to top for Electronics
Top Level for Austin Community College

Questions? Send e-mail to: *knalty@austin.cc.tx.us*

APPENDIX - B

8086 INSTRUCTION SET

This summary is presented only for reference use. For those instructions that can have a variety of operands, the different possibilities are listed. Please refer to outside documentation for more detailed explanations of the exact use of these instructions.

Data Transfer Instructions

A. MOV - Move:

- register or memory location to or from register
- immediate to register or memory location
- immediate to register
- memory location to accumulator
- accumulator to memory location
- register or memory location to segment register
- segment register to register or memory location

B. PUSH - Push:

- register or memory location
- register
- segment register

C. POP - Pop:

- register or memory location
- register
- segment register

D. XCHG - Exchange:

- register or memory location with register
- register with register

E. IN - Input from:

- fixed port
- variable port

F. OUT - Output to:

- fixed port
- variable port

G. XLAT - Translate byte to AL:

H. LEA - Load EA (Effective Address) to register

I. LDS - Load pointer value to DS and register

J. LES - Load pointer value to ES and register

K. LAHF - Load AF with flags

L. SAHF - Store AH into flags

M. PUSHF - Push flags

N. POPF - Pop flags

Arithmetic Instructions

A. ADD - Add:

- register or memory location with register with result stored in either
- immediate to register or memory location
- immediate to accumulator

B. ADC - Add with carry:

- register or memory location with register with result stored in either
- immediate to register or memory location
- immediate to accumulator

C. INC - Increment

- register or memory location
- register

D. AAA - ASCII adjust for addition

E. DAA - Decimal adjust for addition

F. SUB - Subtract:

- register or memory location with register with result stored in either
- immediate to register or memory location
- immediate to accumulator

G. SBB - Subtract with borrow:

- register or memory location with register with result stored in either
- immediate to register or memory location
- immediate to accumulator

H. DEC - Decrement:

- register or memory location
- register

I. NEG - Negate the contents of a specified register or memory location

J. CMP - Compare:

- register or memory location with register
- immediate with register or memory location
- immediate with accumulator

K. AAS - ASCII adjust for subtract

L. DAS - Decimal adjust for subtract

M. MUL - Multiply (unsigned) accumulator by register or memory location

N. IMUL - Integer multiply (signed) accumulator by register memory location

O. AAM - ASCII adjust for multiplication

P. DIV - Divide (unsigned) accumulator by register or memory location

Q. IDIV - Integer divide (signed) accumulator by register or memory location

R. ADD - ASCII adjust for division

S. CBW - Convert byte to word and perform sign extension from AL to AX

T. CWD - Convert word to doubleword and perform sign extension from AX to DX

Logical Instructions:

A. NOT - Ones complement of register or memory location

B. SHL - Logical left shift of register

C. SAL - Arithmetic left shift of register

D. SHR - Logical right shift of register

E. SAR - Arithmetic right shift of register

F. ROL - Rotate register left

G. ROR - Rotate register right

H. RCL - Rotate register left through carry flag

I. RCR - Rotate register right through carry flag

J. AND - Logical and:

- register or memory location with register with result stored in either
- immediate to register or memory location
- immediate to accumulator

K. TEST - Logical AND test with result stored in flags

- register or memory location with register
- immediate data and register or memory location
- immediate data and accumulator

L. OR - Logical or:

- register or memory location with register with result stored in either
- immediate to register or memory location
- immediate to accumulator

M. XOR - Exclusive or:

- register or memory location with register with result stored in either
- immediate to register or memory location
- immediate to accumulator

String Manipulation Instructions

A. REP - Repeat

B. MOVS - Move byte or word

C. CMPS - Compare byte or word

D. SCAS - Scan byte or word

E. LODS - Load byte or word to AL or AX

F. STDS - Store byte or word from AL or AX

Control Transfer Instructions

A. CALL - Call Subroutine:

- direct within segment
- indirect within segment
- direct intersegment
- indirect intersegment

B. JMP - Unconditional jump:

- direct within segment
- direct within segment-short
- indirect within segment
- direct intersegment
- indirect intersegment

C. RET - Return from CALL:

- within segment
- within segment and add immediate to stack pointer
- intersegment
- intersegment and add immediate to stack pointer

D. JE or JZ - Jump on equal to zero **

** The jump instructions that are listed in pairs are exactly identical and can be used inter-changeable.

E. JL or JNGE - Jump on less or not greater or equal

F. JLE or JNG - Jump on less or equal or not greater

G. JB or JNAE - Jump on below or not above or equal

H. JBE or JNA - Jump on below or equal or not above

I. JP or JPE - Jump on parity or parity even

J. JO - Jump on overflow

K. JS - Jump on sign

L. JNE or JNZ - Jump on not equal or not zero

M. JNL or JGE - Jump on not less or greater or equal

N. JNLE or JG - Jump on not less or equal or greater

O. JNB or JAE - Jump on not below or above or equal

P. JNBE or JA - Jump on not below or equal or above

Q. JNP or JPO - Jump on not parity or parity odd

R. JNO - Jump on not overflow

S. JNS - Jump on not sign

T. JCXZ - Jump on CX zero

U. LOOP - Subtract one from CX and jump if greater than zero

V. LOOPZ or LOOPE - Loop while zero or equal

W. LOOPNZ or LOOPNE - Loop while not zero or equal

X. INT - Interrupt
- Type specified

Y. INTO - Interrupt on overflow

Z. IRET - Return from interrupt

Processor Control Instructions

A. CLC - Clear carry

B. CMC - Complement Carry

C. STC - Set Carry

D. CLD - Clear Direction

E. STD - Set Direction

F. CLI - Clear Interrupt

G. STI - Set Interrupt

H. HLT - Halt

I. WAIT - Wait

J. ESC - Escape to external device

K. LOCK - Bus lock prefix

ADDRESSING MODE BYTE:

The 8086 obviously offers an extensive selection of addressing modes. The next questions is: how are these addressing modes implemented in the object code? The 8086 specifies most data memory addressing modes in an instruction's object code using one byte of object code, known as the addressing mode byte. The addressing mode byte may have one or two additional displacement bytes associated with it. The addressing mode byte is always the second byte of be illustrated has been included prior to the initial object code. The addressing mode byte may be illustrated as follows:

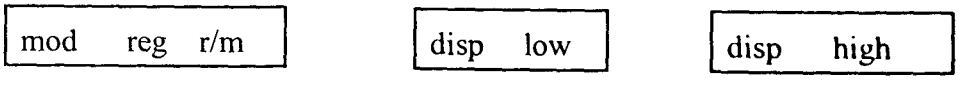
MOD

00 Memory addressing mode. r/m specifies the exact addressing option. There are no displacement bytes.

01 Memory addressing mode. r/m specifies the exact addressing option. There is one displacement byte. This displacement bytes is viewid as a single number in the range -127 to -128. When this number is used in the memory address calculation, the number is signed extended to 16 bits. In this case, the addressing mode bytes can be illustrated as follows.

where mod = 01 and disp is the 8-bit signed displacement value.

10 Memory addressing mode. r/m specifies the addressing option. There are two displacement bytes. The first displacement byte is the low- order eigth bits of the displacement. When this number is used in the memory address calculation, the number is treated as an unsigned 16-bit number. In this case, the addressing mode bytes can be ilustrated as follows.



where mod= 10, disp low is the low-order eight bits of the displacement, and disp high is the high-order eigth bits of the displacement.

11 register addressing mode. r/m specifies a register. Used in conjunction with the w bit to determine if an 8-or 16-bit register is selected.

reg reg is used in conjunction with another bit, the w bit, in the selection of the register to be used in the operation. The w bit, which is part of the instruction op-code, selects whether an 8-or 16-bit operation is performed.

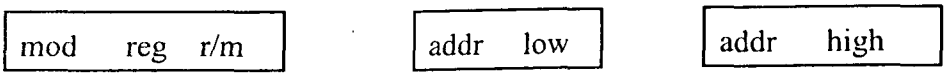
REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Certain instructions- those which require only a single register or memory operand (NOT, NEG, etc), have one implied operand (MOV immediate, DIV, MUL, etc.), or use addressing modes to calculate a target address (JMP, CALL, etc.)- use the reg fields as an extension to the opcode byte to specify the desired instruction. see the descripton of ADC on page 3-59 as an example.

r/m r/m specifies the addressing mode in conjunction with mod, as follows:

r/m	mod - 00	mod -01	mod - 10	mod - 11	
				w =0	w = 1
000	BX+SI	BX+SI+DISP	BX+SI+DISP	AL	AX
001	BX+DI	BX+DI+DISP	BX+DI+DISP	CL	CX
010	BP+SI	BP+SI+DISP	BP+SI+DISP	DL	DX
011	BP+DI	BP+DI+DISP	BP+DI+DISP	BL	BX
100	SI	SI+DISP	SI+DISP	AH	SP
101	DI	DI+DISP	DI+DISP	CH	BP
110	Direct Address	BP+DISP	BP+DISP	DH	SI
111	BX	BX+DISP	BX+DISP	BH	DI

This table is self-explanatory, with the exception of direct address. When mod is 00 and r/m is 110, the offset address is tekan directly from the two bytes that follow the addressing mode byte, this can be illustrated as follows.



where mod is 00, r/m is 110, addr-high is the high-order 8 bits of the offset address and addr-low is the low- order 8 bits of the offset address.

SEGMENT OVERRIDE:

Every addressing moda has a standard default segment register. In most cases you can select an alternative segment register by using a segment override prefix. To use the prefix, place the following byte in front of the instruction whose default segment register assignment is to be overridden.

DIAGRAM

In three cases, the segment override may not be used. They are:

1. Stack reference instructions (e.g., PUSH and CALL) that use the stack pointer (SP register) to compare the offset always use the SS register as the register.

2. String instructions that use the DI register always use the ES register as the segment register. In a string operation where both SI and DI are used (e.g., MOVS or CMPS), a segment override prefix, if present, overrides the SI offset's segment register.

3. Segment override prefixes cannot be used with program memory addressing. All instruciton fetches are relative to the CS segment register.

MEMORY ADDRESSING TABLES:

r/m	mod = 00	mod = 01	mod = 10
000	Base Relative Indexed BX+SI	Base Relative Indexed BX+SI+DISP	Base Relative Direct Index BX+SI+DISP
001	Base Relative Indexed BX+DI	Base Relative Direct Indexed BX+DI+DISP	Base Relative Direct Indexed BX+DI+DISP
010	Base Relative Indexed Stack BP+SI	Base Relative Direct Indexed Stack BP+SI+DISP	Base Relative Direct Indexed Stack BP+SI+DISP
011	Base Relative Indexed Stack BP+DI	Base Relative Direct Indexed BP+DI+DISP	Base Relative Direct Indexed Stack BP+DI+DISP
100	Implied SI	Direct, Indexed SI+DISP	Direct, Indexed SI+DISP
101	Implied DI	Direct, Indexed DI+DISP	Direct, Indexed DI+DISP
110	Direct Direct Address	Base Relative Direct Stack BP+DISP	Base Relative Direct Stack BP+DISP
111	Base Relative BX	Base Relative Direct BX+DISP	Base Relative Direct BX+DISP

Note that two operand instructions will very frequenctly access one operand out of memory, while the other operand is in a CPU register. Also, both operands will frequently be accessed out of CPU registers. The 8086 does not allow both operands to be accessed out of memory, with the exception of several special data string manipulation instructions. The following options are available.

Source Operand	Destination Operand	Result
CPU Register	CPU Register	CPU Register
Memory Location	CPU Register	CPU Register
CPU Register	Memory Location	Memory Location

INSTRUCTION SET MNEMONICS

In the following section, each 8086 assembly language instruction is discussed. The format of each description is composed of six distinct parts:

1. The instruction mnemonic and the various operands associated with it. Variable operands are signified by lower-case letters. The mnemonics itself and any fixed operands are signified by capital letters. Here is an example.

2. Adscription of the instructions operation.

3. The machine language encoding of the instruction.

4. An example of the instructions operation. This is not present for some very simple instruction.

5. A diagram of the instructions execution, which shows the effect the instruction has on the 8086 flags, registers, and memory.

6. A notes section that includes assorted information such as short examples of how the instruction might be used, or related instructions that might be more effective in particular instances.

ABBREVIATIONS:

Theses are abbreviations used for the operands described with the mnemonics:

ac Either the AL register, if an 8-bit operation is specified, or the AX register, if a 16-bit operation is specified. This will be represented in an 8086 assembly language instruction by Al or AX.

Addr An 8086 address composed of two 16-bit addresses, a 16 bit of offset address and a 16-bit segment address. Typically, this is represented in an 8086 assembly language instruction.

count Either 1 or the contents of the CL register. This will be represented by 1 or CL in an 8086 assembly language instruction.

data 8 or 16 bits of immediate data. This can appear as any of a wide selection of numeric representation of expressions in an 8086 assembly language statement.

disp 8-bit signed binary displacement used by the jump and jump-on condition instructions. Invariably this will be represented by a lable in an 8086 assembly language instruction.

disp16 16-bit binary displacement used by the call, jump, and return instructions. When used in the call and jump instructions, this is almost represented by a label. The return instruction will typically use a numeric expression to represent disp16. Its use with the return instruction will be shown.

mem Memory operand. The addressing mode used to select the operand is specified by the addressing mode byte. This will typically be represented by a lable, in which case the assembler will select the appropriate addressing mode byte, or a sequence of symbols that allows the selection of a specific addressing mode byte.

mem/rg Memory or register operand. Consult descriptions for menu and rag.

Port An I/O port. This will be represented by a numeric representation or an expression. The port number must be between 0_{16} and FF_{16} .

reg Register AH, AL, BH, BL, CH, CL, DH or DL if an 8 bit operation is specified. register AX, BX, CX, DX, SP, BP, SI, or DI if a 16-bit operation is specified.

segreg Register CS, DS, ES or SS.

These are the abbreviations used in describing the instruction's encoding.

c One bit used in the shift and rotate instructions selecting either 1 or the contents of the CL register to be the number of shifts/ rotates to be performed.

d One bit used to specify the direction in which an operation is performed.

disp 8 bits used as a signed binary displacement by the jump and jump-on condition instructions.

jj Two hexadecimal digits, used to represent immediate data or part of a 16-bit displacement.

kk Two hexadecimal digits, used to represent immediate data or part of a 16-bit displacement.

mod reg
r/m 8-bit addressing mode byte that is described in earlier in this chapter.

rrr Threee bit selecting one of the 8086 general-purpose registers

IF

an 8-bit operation is specified
rrr = 000 for AL
001 for CL
010 for DL
011 for BL
100 for AH
101 for CH
110 for DH
111 for BH

16-bit operation is specified
rrr = 000 for AX

S One bit indicating whether or not immediate data is to be sign extended. If a 16-bit operation with immediate data is specified, it is possible that the immediate operand can be expressed using just one byte of program memory space, s is interpreted as follows.

s=0, Two bytes are necessary for the immediate data, no sign extension is performed.
s=1, One byte of immediate data is present. To form the sixteen bits of immediate data necessary for the operation data byte.

ss Two bits selecting one of the 8086 segment register

ss = 00 for ES
01 for CS
10 for SS
11 for DS

V one bit indicating the location to which a software interrupt should be vectored. If v= 0, then the interrupt service routine is located at the address specified at location 0000C16 otherwise the address is determined by the succeding byte.

w One bit indicating whether an 8-or 16-bit operation is performed.

w=0 8-bit operation
w=1 16-bit operation

xxx Three don't care bits

yy two hexadecimal digits indicating the I/O port number to be used by the instruction.

The following symbols are used in the example use of instructions.

H The will appear at the end of a group of digits of specify that the digits be treated as hexadecimal digits

[ ] These are used indicate the contents of the memory location addressed by the expression inside the brackets. Suppose that the BX register contains $054A_{16}$ in the current data segment.

x,y,z

hexadecimal digit. For example.

[BX]

data element.

address.

EA appears in calculations for the number of execution cycles required by individual instructions. EA specifies addressing mode execution cycles, which must be

	ADD 6 cycles
d Addressing	ADD 9 cycles
	ADD 5 cycles
ng	ADD 5 cycles
ddressing	ADD 9 cycles
Addressing	ADD 7 or 8 cycles
Direct	
	ADD 11 or 12 cycles

mode cycles must be added as follows:

efix is present	ADD 2 cycles
ssed and the word	
ory address	ADD 4 cycles

modes require one

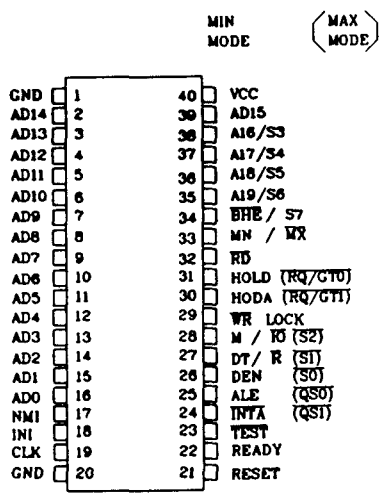
than BP+DI and BX+SI modes

APPENDIX - C

IC PINOUTS

APPENDIX–C

8086 – 16 BIT MICROPROCESSOR



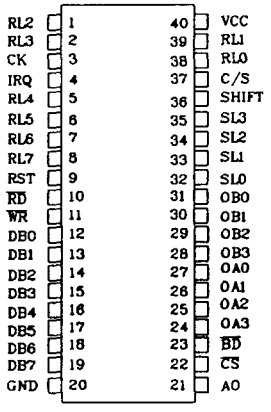
SIGNALS	INPUT/ OUTPUT	DESCRIPTION
AD15–AD0	I/O	ADDRESS/DATA BUS
NMI	I	NON–MASKABLE INTERRUPT (EDGE TRIGGERED)
INTR	I	INTERRUPT REQUEST (LEVEL TRIGGERED)
CLK	I	CLOCK INPUT
RESET	I	READY INPUT
TEST	I	ACTIVE LOW TEST INPUT CPU IS IDLE IF THIS PIN IS LOW
RD	O	ACTIVE LOW READ
BHE /S7	O	BUS HIGH ENABLE /STATUS 7
MN/ MX	I	MINIMUM/MAXIMUM INDICATES MODE IN WHICH PROCESSOR IS TO OPERATE
A19/S6 A18/S5 A17/S4 A16/S3	O	ADDRESS LINES/STATUS LINES
VCC		+5V POWER SUPPLY
GND		GROUND

PINS WITH DIFFERENT FUNCTIONS IN THE 2 MODES

MAXIMUM MODE	INPUT/ OUTPUT	DESCRIPTION
QS1.QS0	O	QUEUE STATUS
S0, S1, S2	O	STATUS LINES
LOCK	O	LOCK OUTPUT
RQ/GT1	I/O	REQUEST/BUS GRANT
RQ/GT0	I/O	REQUEST/BUS GRANT WITH HIGHER PRIORITY THAN RQ/GT1

MAXIMUM MODE	INPUT/ OUTPUT	DESCRIPTION
INTA	O	INTERRUPT ACKNOWLEDGE
ALE	O	ADDRESS LATCH ENABLE
DEN	O	DATA ENABLE
DT/R	O	DATA TRANSFER/RECEIVE
M/IO	O	MEMORY I/O
WR	O	WRITE/IO
HLDA	O	HOLD ACKNOWLEDGE
HOLD	I	HOLD INPUT

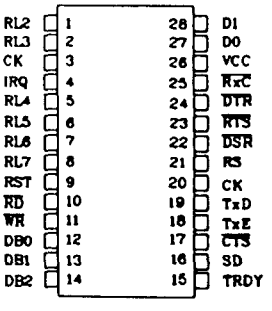
8279 - PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE



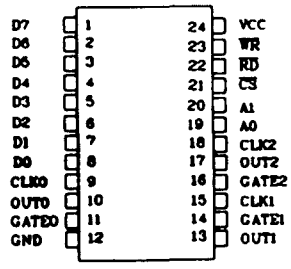
SIGNALS	INPUT/ OUTPUT	DESCRIPTION
DB0-DB7	I/O	DATA BUS
CLK	I	CLOCK INPUT
RESET	I	RESET
$\overline{CS}$	I	CHIP SELECT
$\overline{RD}$	I	READ
$\overline{WR}$	I	WRITE
A0	I	BUFFER ADDRESS
IRQ	O	INTERRUPT REQUEST
SL0-SL3	O	SCAN LINES
SHIFT	I	SHIFT
CNTL/ST3	I	CONTROL/STROBED INPUT MODE
OUTA0-OUTA13 OUTB0-OUTB13	O	OUTPUT FOR DISPLAY REFRESH REGISTERS
$\overline{BD}$	O	BLANK DISPLAY
VCC		+5V POWER SUPPLY
GND		GROUND

8251 - PROGRAMMABLE COMMUNICATION INTERFACE

SIGNALS	INPUT/ OUTPUT	DESCRIPTION
DB0-DB7	I/O	DATA BUS
RESET	I	RESET
CLK	I	CLOCK INPUT
C/D	I	CONTROL/DATA
$\overline{RD}$	I	READ
$\overline{CS}$	I	CHIP SELECT
$\overline{WR}$	I	WRITE
$\overline{DSR}$	I	DATA SET READY
$\overline{DTR}$	O	DATA TERMINAL READY
$\overline{CTS}$	I	CLEAR TO SEND
$\overline{RTS}$	O	REQUEST TO SEND
TXD	O	TRANSMIT DATA
TXRDY	O	TRANSMITTER READY
TXE	O	TRANSMITTER EMPTY
)TXC	I	TRANSMITTER CLOCK
RTX	I	RECEIVE DATA
RXRDY	O	RECEIVE READY
$\overline{RXC}$	I	RECEIVE CLOCK
SYNDET	I/O	SYNC/BREAK/DEFECT



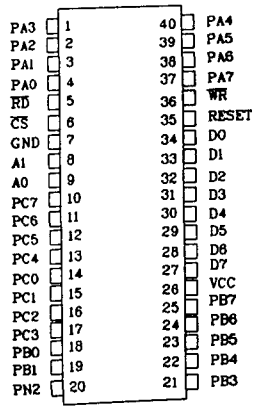
8253 - PROGRAMMABLE INTERVAL TIMER



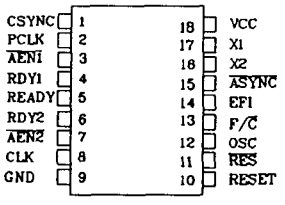
SIGNALS	INPUT/ OUTPUT	DESCRIPTION
D0-D7	I/O	DATA BUS
WR	0	WRITE
RD	0	READ
CS	I	CHIP SELECT
A0,A1	I	COUNTER SELECT INPUTS
CLK 0 CLK 1 CLK 2	I	CLOCK INPUTS FOR 3 CHANNELS
GATE 0 GATE 1 GATE 2	I	GATE INPUTS FOR 3 CHANNELS
OUT 0 OUT 1 OUT 2	0	OUTPUTS FOR 3 CHANNELS
VCC	I	+5V POWER SUPPLY
GND	I	GROUND

8255 - PROGRAMMABLE PHERIPHERAL INTERFACE

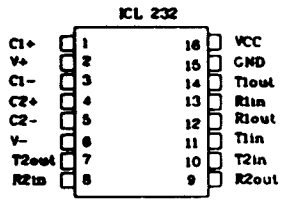
SIGNALS	INPUT/ OUTPUT	DESCRIPTION
DB0-DB7	I/O	DATA BUS
RESET	I	RESET
RD	I	READ
CS	I	CHIP SELECT
WR	I	WRITE
A0, A1	I	PORT SELECT 0, PORT SELECT 1.
PA0-PA7	I/O	PORT A
PB0-PB7	I/O	PORT B
PC0-PC7	I/O	PORT C
VCC	I	+5V POWER INPUT
GND	I	GROUND

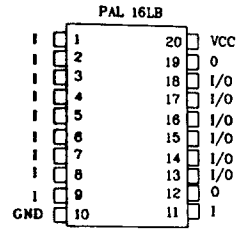


8284 – CLOCK GENERATOR & DRIVER

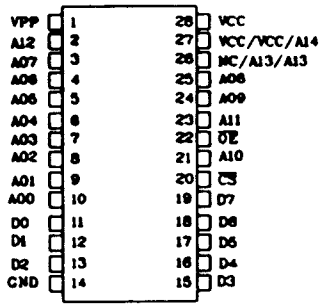


SIGNALS	INPUT/ OUTPUT	DESCRIPTION
$\overline{AEN1}$, $\overline{AEN2}$	I	ADDRESS BUS ENABLE QUALIFIES RDY1/RDY2 RESPECTIVELY
RDY1, RDY2	I	BUS READY- WHEN HIGH INDICATES THAT DATA IS AVAILABLE ON DATA BUS FROM EXTERNAL DEVICE
$\overline{ASYNC}$	I	READY SYNCHRONISATION SELECT DEFINES SYNCHRONISATION MODE OF READY LOGIC
READY	O	READY OUTPUT
X1,X2	I	CRYSTAL INPUT TERMINALS CRYSTAL FREQUENCY MUST BE THRICE THE PROCESSOR'S REQUIRED FREQUENCY
F/C	I	EXTERNAL FREQUENCY / CRYSTAL SELECT
EFI	I	EXTERNAL FREQUENCY INPUT
CLK	O	PROCESSOR CLOCK
PCLK	O	PERIPHERAL CLOCK- HALF THE FREQUENCY OF PROCESSOR CLOCK
OSC	O	OSCILLATOR OUTPUT
$\overline{RES}$	I	ACTIVE LOW RESET INPUT
RESET	O	ACTIVE LOW RESET OUTPUT
CSYNC	I	CLOCK SYNCHRONISATION- ALLOWS MULTIPLE 8284'S TO BE SYNCHRONISED TO PROVIDE CLOCK IN PHASE
VCC		+5V POWER SUPPLY
GND		GROUND





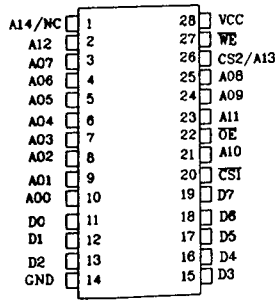
2764/128/256 - 8K/16K/32K UVEPROM



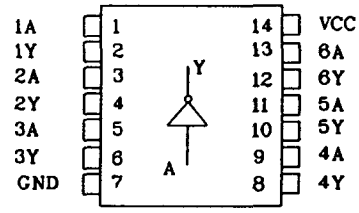
SIGNALS	DESCRIPTION
A0-A14	ADDRESS BUS
D0-D7	DATA BUS
$\overline{OE}$	OUTPUT ENABLE
$\overline{CS}$	CHIP SELECT
Vpp	PROGRAMMING VOLTAGE
NC	NO CONNECTION FOR 2764
VCC	+5V POWER INPUT
GND	GROUND

6264/256 - 8K/32K STATIC RAM

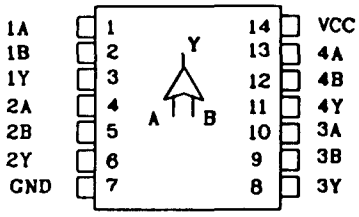
SIGNALS	DESCRIPTION
A0-A14	ADDRESS BUS
D0-D7	DATA BUS
$\overline{OE}$	OUTPUT ENABLE
$\overline{CS}$	CHIP SELECT
Vpp	PROGRAMMING VOLTAGE
NC	NO CONNECTION FOR 2764
VCC	+5V POWER INPUT
GND	GROUND



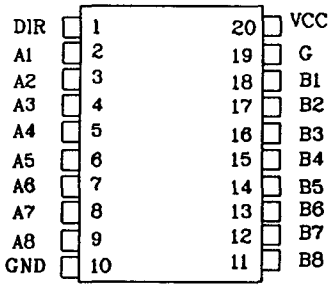
HEX INVERTOR
7404



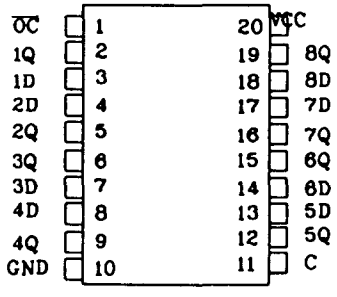
QUAD 2 I/P OR GATE
7432



OCTAL BUS TRANSCEIVERS
743245



OCTAL D LATCHES
74373



APPENDIX - D

SYSTEM CALLS

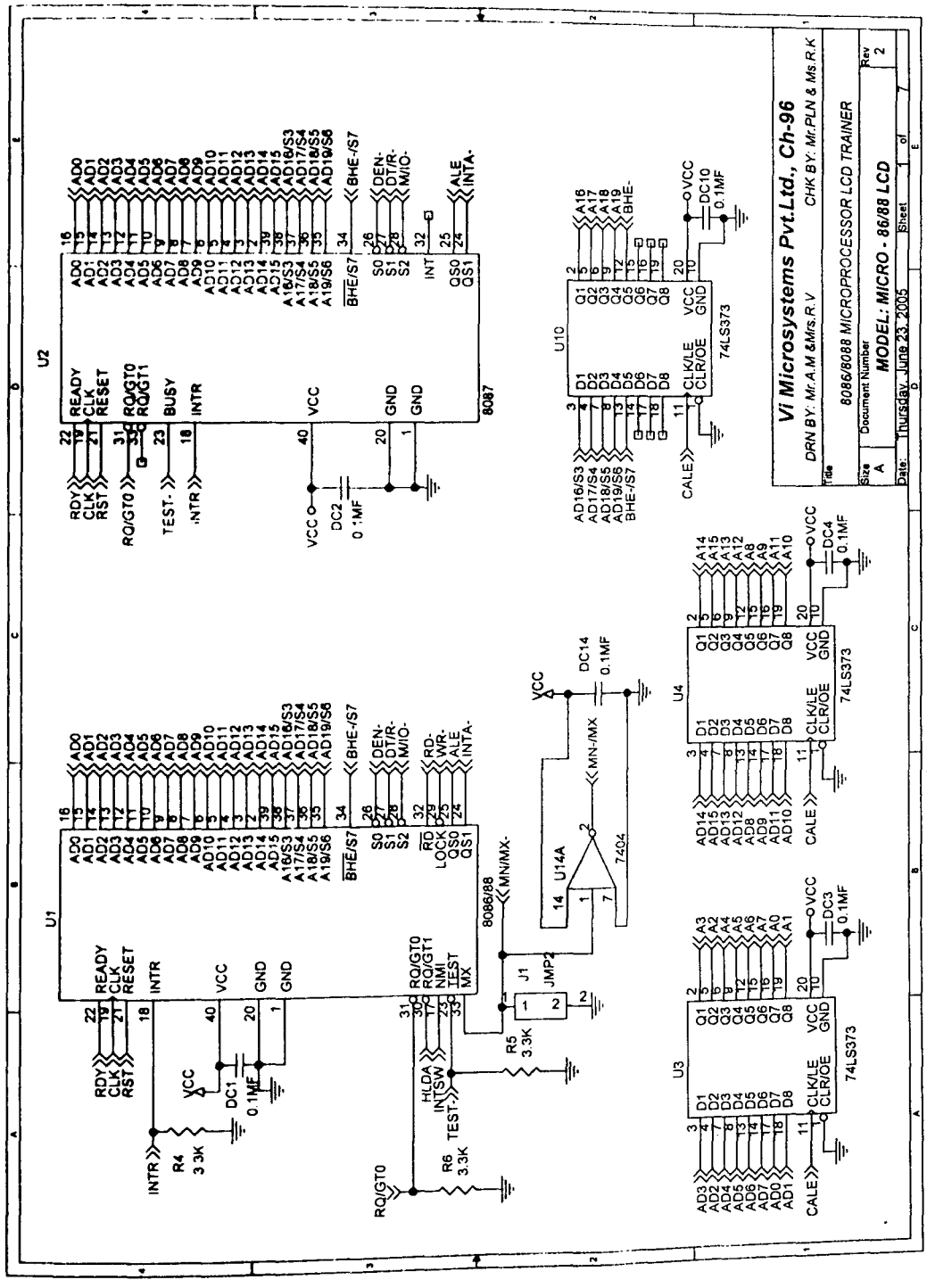
QUICK REFERENCE CARD

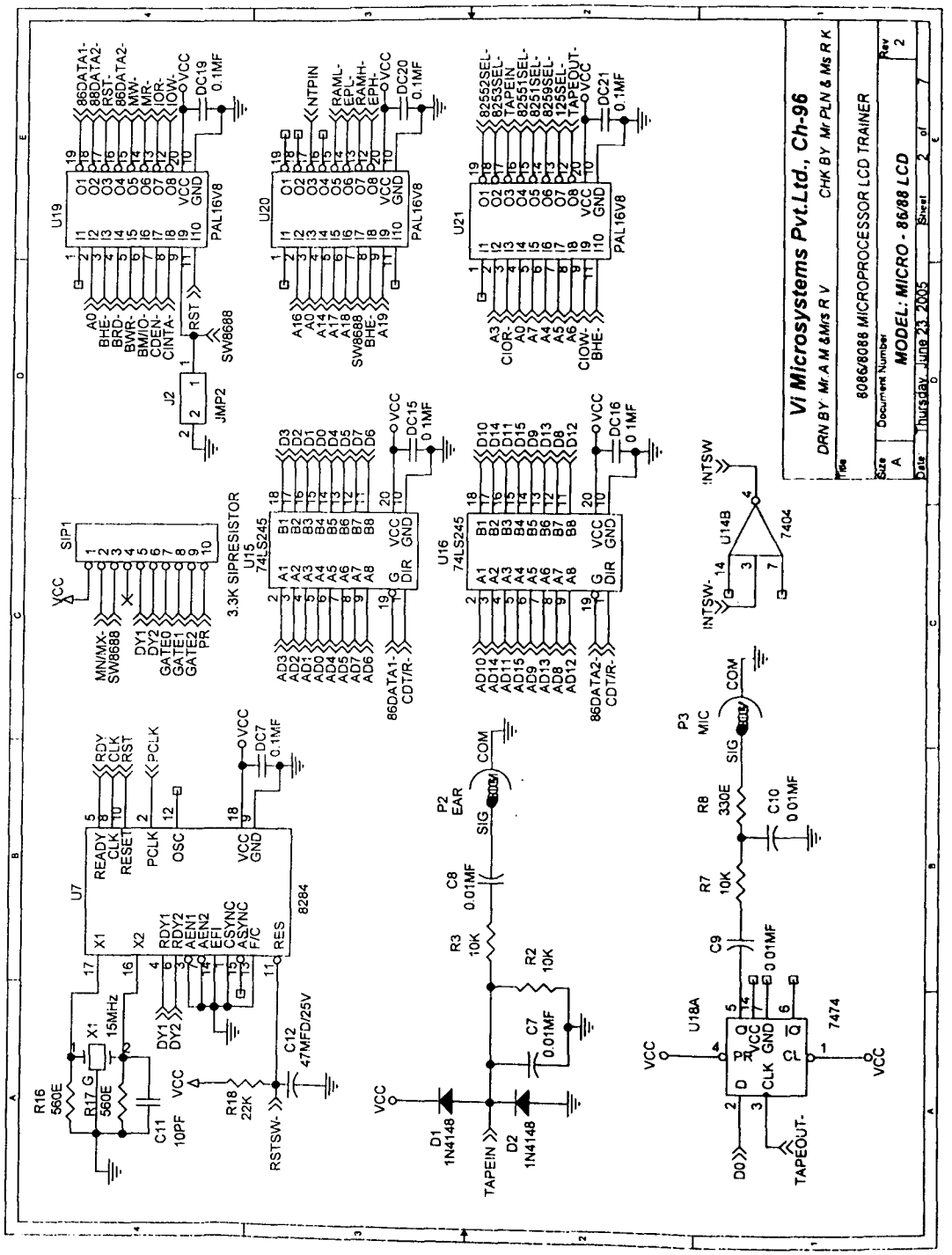
AL	AH	DX	OUTPUT	FUNCTION
X	00	X	X	Kit Reset
X	01	X	AL = Keycode	RDKBD
LN 0-Addr 1-Data HN 0-Word 1-Byte	02	X	Data in DX	GETHEX
LN 0-Addr 1-Data HN 0-Word 1-Byte	03	Data	X	PUTHEX
LN 0-Addr 1-Data 2-Both HN 0-Blank 1-Prompt	04 05	Memory Pointer X	X X	Message Display Addrblk Datablk addrpmt Datapmt
LN 0-Addr 1-Data X	06	X	A = Data from the serial port	SERIN

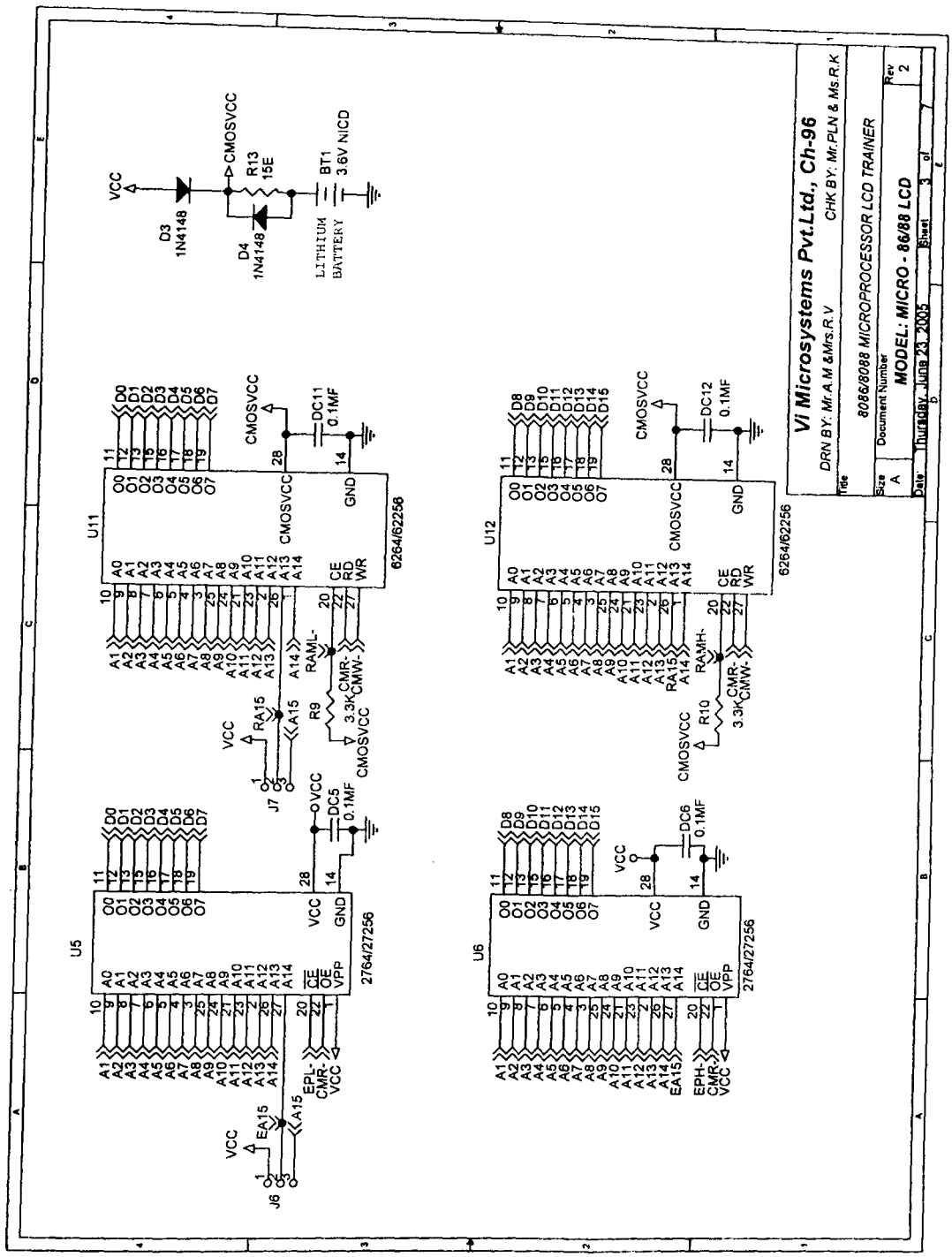
X	-	Dont care
Addr	-	Address
HN	-	Higher Nibble
LN	-	Lower Nibble
Datablk	-	Data Field Blank
Addrpmt	-	Address Field Prompt
Addrblk	-	Address Field Blank
Datapmt	-	Data Field Prompt
RDKBD	-	Read a key from the keyboard
GETHX	-	Get a Hex data from the keyboard
PUTHEX	-	Display the Hex data in the display
SERIN	-	Input a byte from the serial port

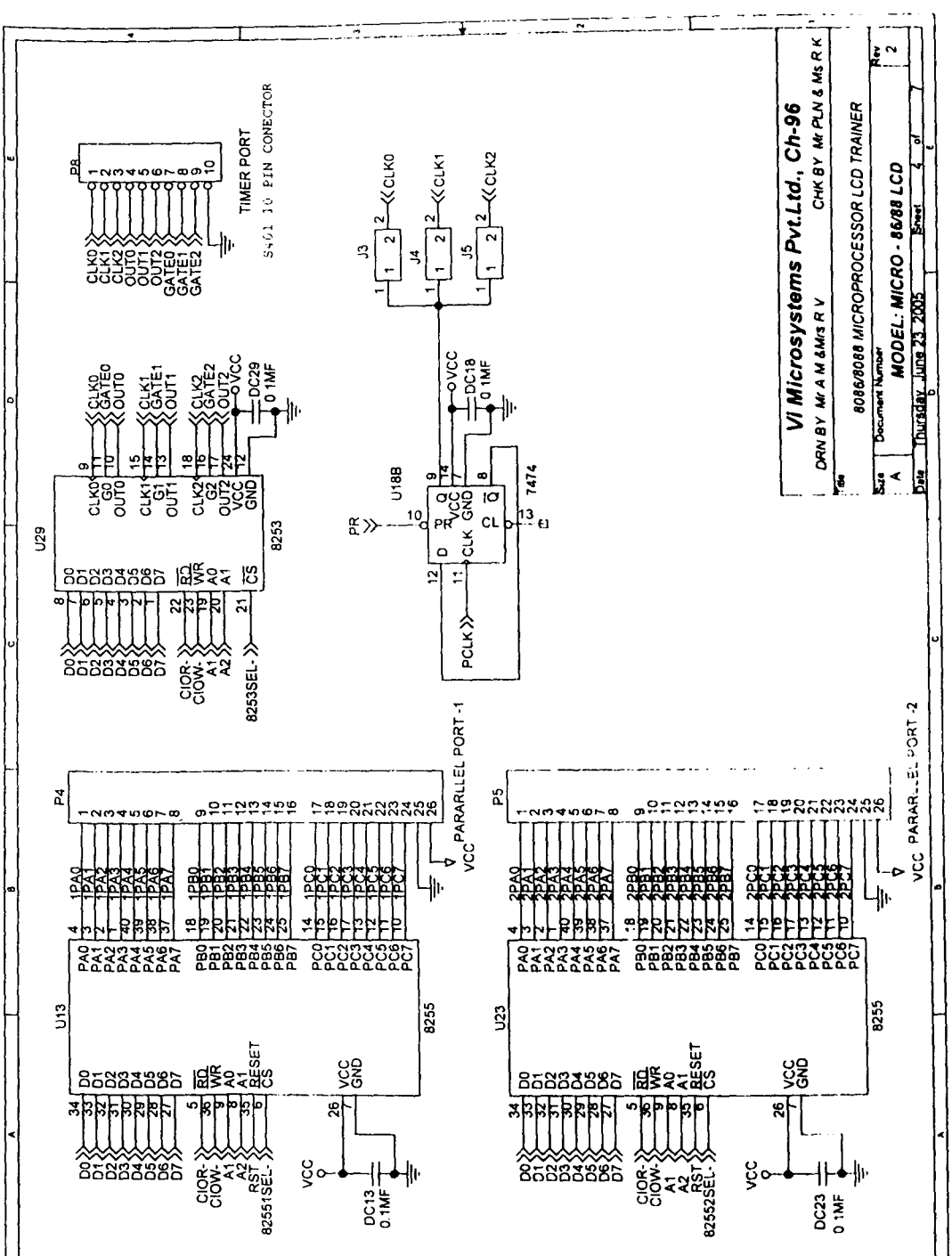
APPENDIX - F

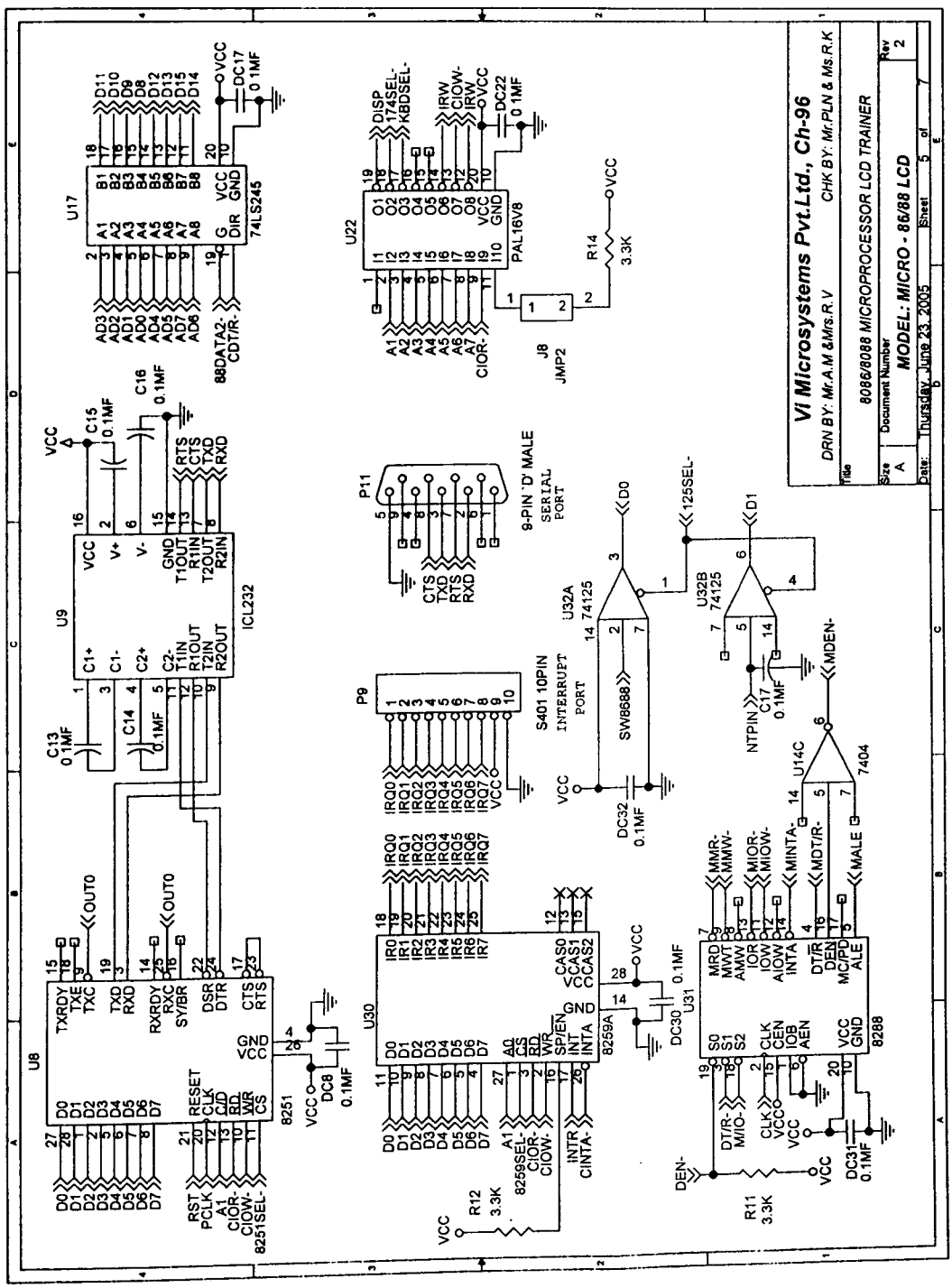
CIRCUIT DIAGRAM

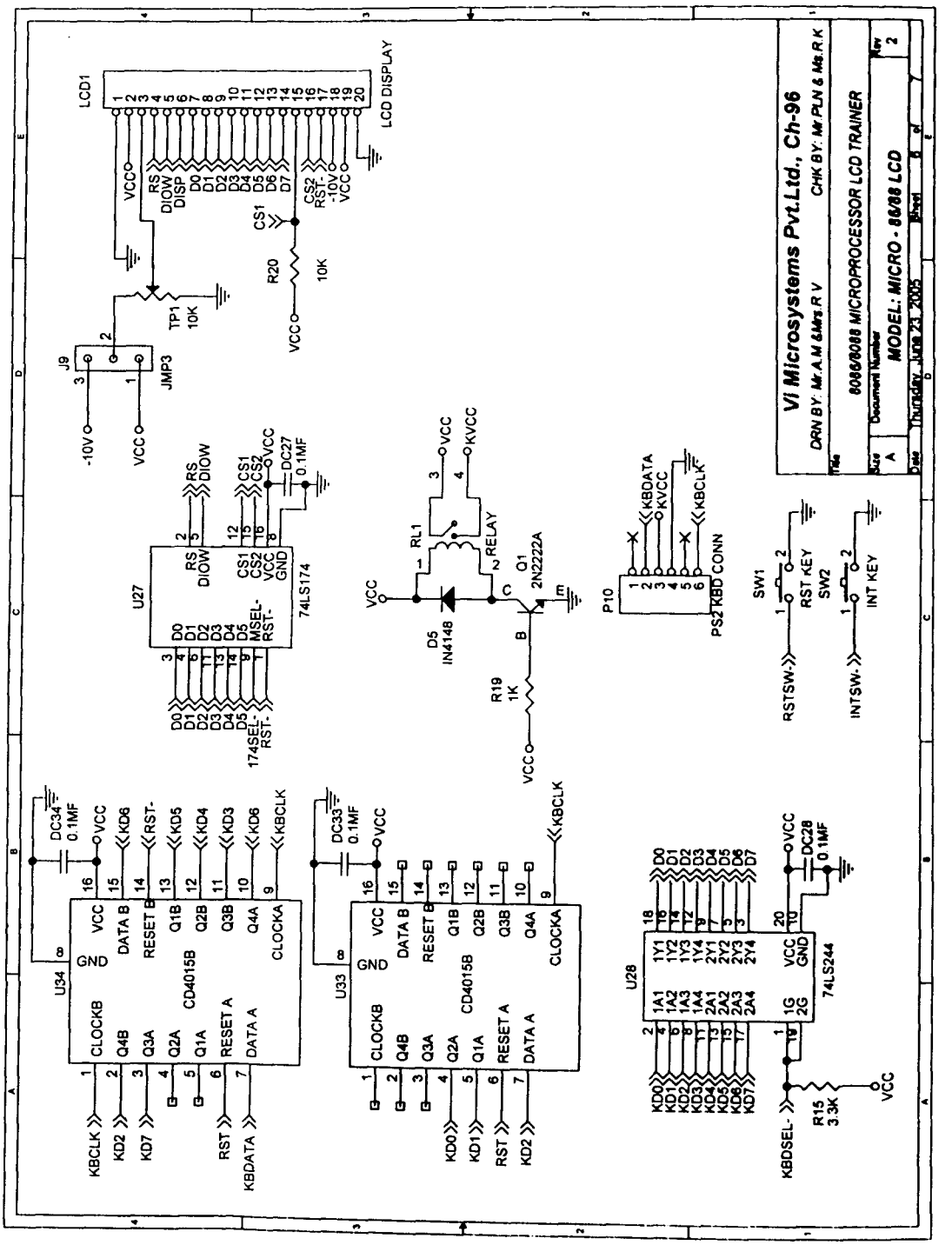


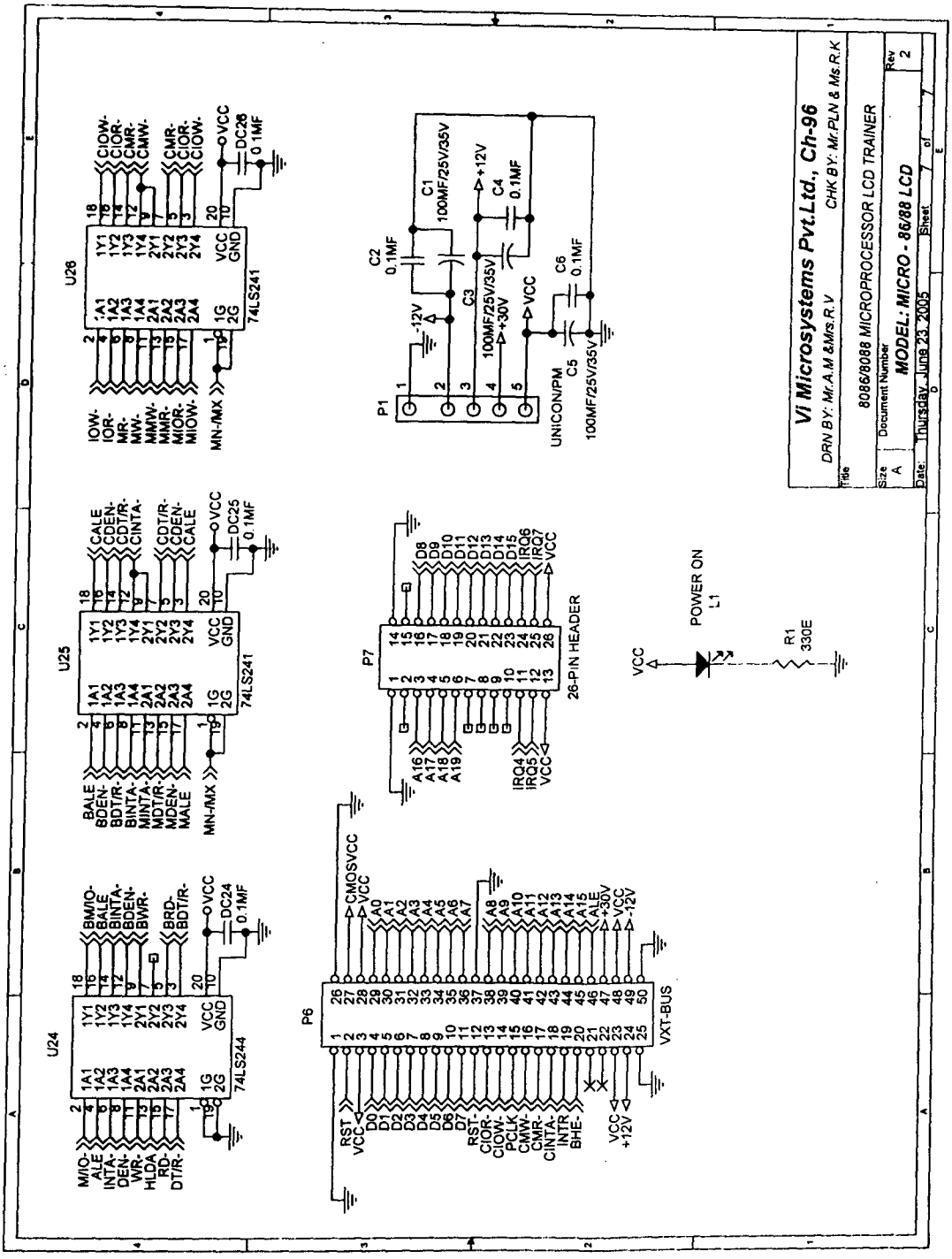

Vi Microsystems Pvt.Ltd., Ch-96
DRN BY Mr A M & Mrs R V
CHK BY Mr PLN & Ms R K
8086/8088 MICROPROCESSOR LCD TRAINER
MODEL: MICRO - 86/88 LCD
Thursday, June 23, 2005
U7
8284
U15
74LS245
U16
74LS245
U18A
7474
U19
PAL16V8
U20
PAL16V8
U21
PAL16V8
SIP1
3.3K SIPRESISTOR
U14B
7404
P2
EAR
P3
MIC
X1
15MHz
R16
560E
R17
560E
R18
22K
C11
10PF
C12
47MFD/25V
C7
0.01MF
C8
0.01MF
C9
0.01MF
C10
0.01MF
R2
10K
R3
10K
R7
10K
R8
330E
D1
1N4148
D2
1N4148
J2
JMP2
SW8688
DC7
0.1MF
DC15
0.1MF
DC16
0.1MF
DC19
0.1MF
DC20
0.1MF
DC21
0.1MF


Vi Microsystems Pvt.Ltd., Ch-96
DRN BY: Mr.A.M &Mrs.R.V
CHK BY: Mr.PLN & Ms.R.K
8086/8088 MICROPROCESSOR LCD TRAINER
MODEL: MICRO - 86/88 LCD
Thursday, June 23, 2005
VCC
CMOSVCC
R13 15E
BT1 3.6V NICD
LITHIUM BATTERY
D3 1N4148
D4 1N4148
U11
U12
U5
U6
6264/62256
2764/27256
DC11 0.1MF
DC12 0.1MF
DC5 0.1MF
DC6 0.1MF
R9 3.3K
R10 3.3K
RAML-
RAMH-
CMR-
CMW-
EPL-
EPH-
J6
J7


Vi Microsystems Pvt.Ltd., Ch-96
DRN BY Mr A M &Mrs R V
CHK BY Mr PLN & Ms R K
8086/8088 MICROPROCESSOR LCD TRAINER
MODEL: MICRO - 86/88 LCD
Thursday, June 23, 2005
TIMER PORT
PARARLLEL PORT -1
PARARLLEL PORT -2
U29
8253
U13
8255
U23
8255
U18B
7474
P4
P5
P8
J3
J4
J5
CLK0
CLK1
CLK2
DC29 0.1MF
DC18 0.1MF
DC13 0.1MF
DC23 0.1MF
8253SEL->
82551SEL->
82552SEL->
PCLK>>

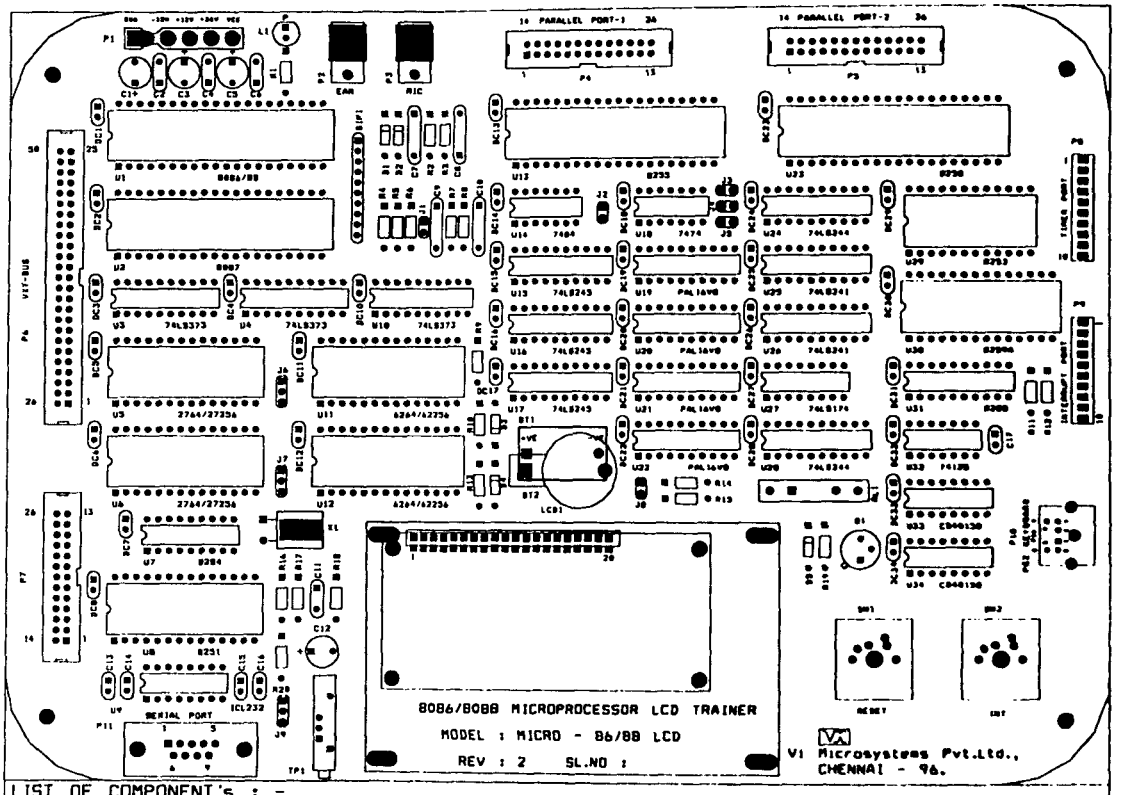

Vi Microsystems Pvt.Ltd., Ch-96
DRN BY: Mr.A.M &Mrs.R.V
CHK BY: Mr.PLN & Ms.R.K
8086/8088 MICROPROCESSOR LCD TRAINER
MODEL: MICRO - 86/88 LCD
U8
U9
ICL232
U17
74LS245
U22
PAL16V8
U30
8259A
U31
8288
U32A 74125
U32B 74125
U114C 7404
P11
9-PIN 'D' MALE SERIAL PORT
P9
S401 10PIN INTERRUPT PORT
J8 JMP2
8251
8251SEL
88DATA2-
CDT/R-
DISP
174SEL-
K8DSEL-
IRW
CIOW-
MDEN-
NTPIN
125SEL-
SW8688
VCC
GND


Vi Microsystems Pvt.Ltd., Ch-96
DRN BY: Mr.A.M & Mrs.R.V
CHK BY: Mr.PLN & Ms.R.K
8086/8088 MICROPROCESSOR LCD TRAINER
MODEL: MICRO - 86/88 LCD
LCD1
LCD DISPLAY
TP1 10K
R20 10K
JMP3
U27 74LS174
RL1 RELAY
D5 IN4148
Q1 2N2222A
R19 1K
P10 PS2 KBD CONN
SW1 RST KEY
SW2 INT KEY
U34 CD4015B
U33 CD4015B
U28 74LS244
R15 3.3K


Vi Microsystems Pvt.Ltd., Ch-96
DRN BY: Mr.A.M &Mrs.R.V
CHK BY: Mr.PLN & Ms.R.K
8086/8088 MICROPROCESSOR LCD TRAINER
MODEL: MICRO - 86/88 LCD
Thursday, June 23, 2005
U24
74LS244
U25
74LS241
U26
74LS241
P1
UNICON/PM
P6
VXT-BUS
P7
26-PIN HEADER
C1 100MF/25V/35V
C2 0.1MF
C3 100MF/25V/35V
C4 0.1MF
C5 100MF/25V/35V
C6 0.1MF
DC24 0.1MF
DC25 0.1MF
DC26 0.1MF
R1 330E
POWER ON
L1
VCC
GND
+12V
-12V
+30V

APPENDIX - E

COMPONENT LAYOUT



LIST OF COMPONENT's : -

IC's :-

U1 = 8086/8088 (40 PIN BASE)
U2 = 8087 (40 PIN BASE)
U3,U4,U10 = 74LS373 (20 PIN BASE)
U5,U6 = 2764/27256 (28 PIN BASE)
U7 = 8284 (18 PIN BASE)
U8 = 8251 (28 PIN BASE)
U9 = ICL232 (16 PIN BASE)
U11,U12 = 6264/62256 (28 PIN BASE)
U13,U23 = 8255 (40 PIN BASE)
U14 = 7404 (14 PIN BASE)
U15,U16,U17 = 74LS245 (20 PIN BASE)
U18 = 7474 (14 PIN BASE)
U19,U20,U21,U22 = PAL16V8 (20 PIN BASE)
U24,U28 = 74LS244 (20 PIN BASE)
U25,U26 = 74LS241 (20 PIN BASE)
U27 = 74LS174 (16 PIN BASE)
U29 = 8253 (24 PIN BASE)
U30 = 8259A (28 PIN BASE)
U31 = 8288 (20 PIN BASE)
U32 = 74125 (14 PIN BASE)
U33,U34 = CD4015B (16 PIN BASE)

CONNECTOR's :-

P1 = 5 PIN UNICON MLAE
P2,P3 = EP SOCKET
P4,P5,P7 = 26 PIN FRC CONN
P6 = 50 PIN FRC CONN
P8,P9 = S401 10 PIN CONN
P10 = PS2 KEYBOARD CONN
P11 = 9 PIN 'D' MALE CONN

RESISTOR's :-

R1,R8 = 330E
R2,R3,R7,R20 = 10K
R13 = 15E
R4,R5,R6,R9,R10,R11,R12,R14,R15 = 3.3K
R16,R17 = 560E
R18 = 22K
R19 = 1K

CAPACITOR's :-

C1,C3,C5 = 100MF/25V/35V/E
C2,C4,C6,C13,C14,C15,C16,C17 = 0.1MF/C
ALL DC = 0.1MF/C
C7,C8,C9,C10 = 0.01MF/C
C11 = 10PF
C12 = 47MF/25V/E

JUMPER's :-

J1,J2,J3,J4,J5,J8 = 2 PIN JUMPER
J6,J7,J9 = 3 PIN JUMPER

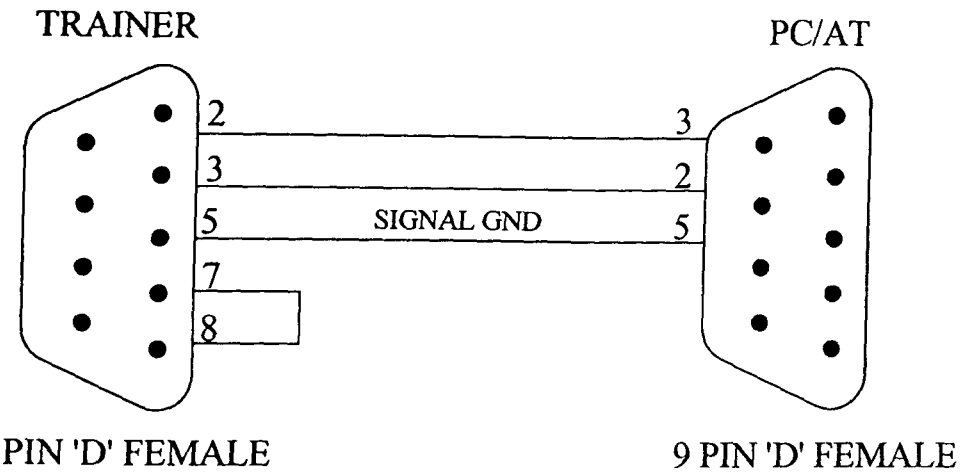
OTHER COMPONENT's :-

BT1 = 3.6V NICD BATTERY (Or)
BT2 = LITHIUM BATTERY
D1,D2,D3,D4,D5 = IN4148 DIODE
LCD1 = 16x2 (Or) 20x4 LCD DISPLAY
L1 = 5MM RED LED
Q1 = 2N2222A TRANSISTOR
RL1 = 5V SIP RELAY
TP1 = 10K TRIMPOT (V Or H)
SIP1 = 3.3K 10 PIN SIPRESISTOR
SW1,SW2 = RRE KEY
X1 = 15MHz CRYSTAL

APPENDIX - G

RS232 CABLE

CONFIGURATION FOR PC/ AT

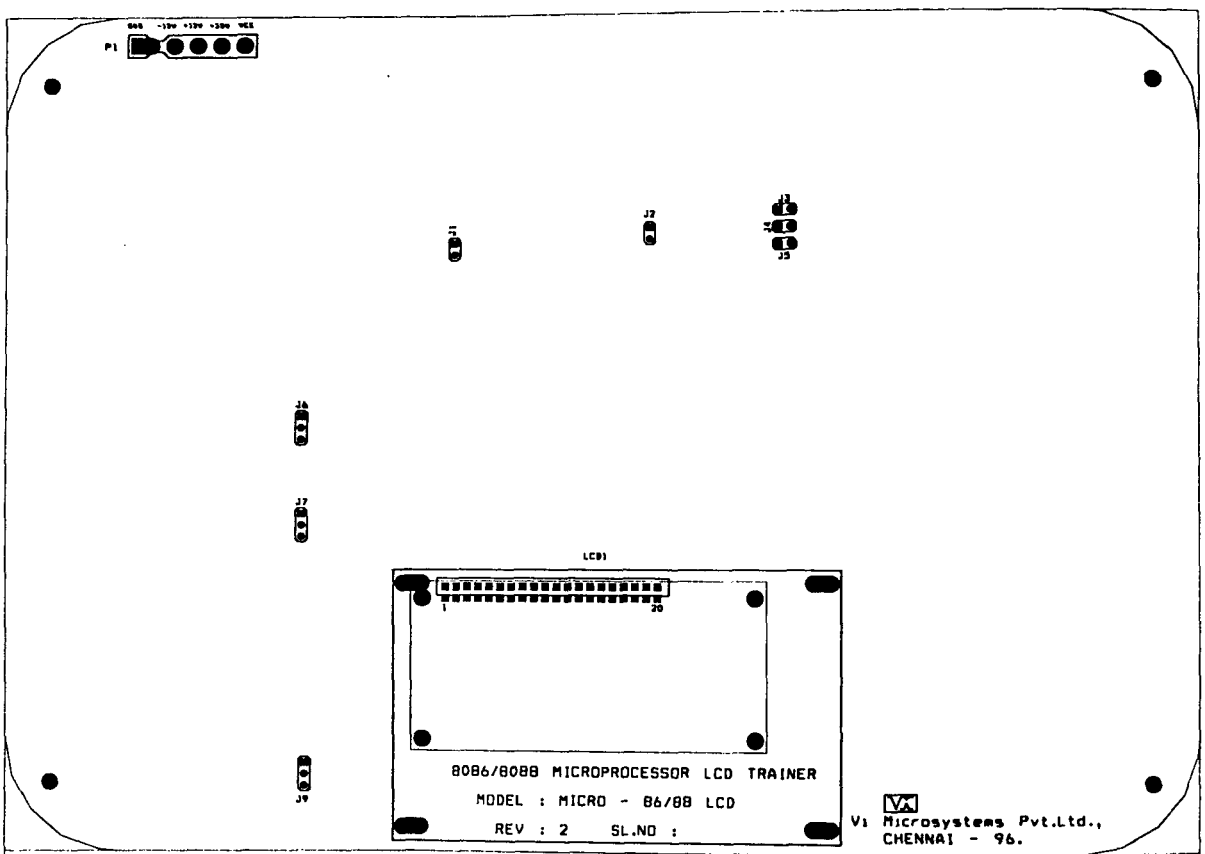


IN THE TRAINER THE CONNECTOR IS A 9 PIN 'D' MALE CONNECTOR

IN THE PC/AT THE CONNECTOR IS A 9 PIN 'D' MALE CONNECTOR

APPENDIX - H

JUMPER DETAILS



JUMPER DETAILS:-

J1 = MAX/MIN MODE SELECT
J2 = MEMORY SELECTION(C000)
J3,J4,J5 = CLOCK I/P TO TIMER
J6 = MEMORY SECTION FOR EPROM
J7 = MEMORY SECTION FOR RAM
J9 = VOLTAGE SELECT TO LCD DISPLAY