DESIGN AND FABRICATION OF GENERAL PURPOSE 8085 CARD AND DEVELOP VARIOUS INTERFACING CARDS USEFUL IN EDUCATION OF ELECTRONICS.

Thesis

Submitted to the Saurashtra University, Rajkot
For
The Degree of Doctor of Philosophy
(Science)
(Electronics)

By
Darshan G. Vyas
Lecturer in Physics,
Hemchandracharya North Gujarat University,
Patan-384265(North Gujarat)

Research Supervisor
Dr. H. N. Pandya
Associate Professor,
Department of Electronics / Physics
Saurashtra University,
Rajkot-360005
India

April 2005

Affectionately dedicated to
My parents
Retd. Prof. G. P. Vyas &
Mrs. Hansaben G. Vyas
and
my younger brother
Dhaval G. vyas

Statement under O.Ph.D. of Saurashtra University

The content of this thesis is my own work carried out under the supervision of Dr. H. N. Pandya and leads to some contribution in Electronics supported by necessary references.

(D.G.Vyas)

This is to certify that the present work submitted by Mr. Darshan G. Vyas for the Ph.D. degree of Saurashtra University, Rajkot has been the result of about 5 years of work under my supervision and is a valuable contribution in the field of Electronics.

Dr. H. N. Pandya,
Associate Professor,
Department of Electronics / Physics,
Saurashtra University,
Rajkot- 360 005

Acknowledgement

It is impossible to express words of thanks for my 'GURU' Dr.H.N.Pandya, but I am trying to express my sincere thanks to him. The name of Ph.D. student is bound with the name of his guide for lifetime, and it is impossible to obtain this most valuable degree without the proper guidance. Dr.H.N.Pandya helped me everywhere. He not only solved my technical problems but also helped me in the development of my educational carrier. For me and my family this name is unforgettable.

I am highly indebted to my parents Retd. Prof.Ghanshyambhai P.Vyas and Mrs. Hansaben G. Vyas who have brought up me with immense love and helped me to get education to this stage. Their loving care and all time worries have been a main source of my inspiration. My younger brother Dhaval Vyas also boosted my morals and helped me at every stage of this work. During my stay in Rajkot and crucial working hours my friend like elder brother Mr. Jitubhai Vyas has played a key role in helping me and maintain my time schedule by adjusting his work schedule.

In the early stage of my research work the engineer Mr. Maheshbhai Bhatt (Physics Department, Gujarat University, A'bad) has been kind enough in clearing my some of the basic doubts.

I am also thankful to Prof. B.J.Mehta, Prof. N.N.Jani for giving me moral support. I feel happy to say that I got chance to work in the company of Dr.M.N.Jivani and Dr.N.A.Shah.

The moral support and affectionate concern of Mrs. Kiranben H. Pandya in the critical moments of my research work and during my disappointments have soothed my soul and made me to resume my work with eternal enthusiasm.

I am thankful to my colleague Ph.D. students Mr. Bimal Vyas and Mr. Maulin Nanavati for helping me at various stages of my research work. I wish to express my loving thanks to my friends Mr. Manishbhai Pandya and Mr. Taresh Bhatt for sharing all good and "stressed" moment of my work. The help extended by Mr. Bhaskar Anand and Harshad Chirutkar is also appreciated.

I am also thankful to Mr. Pareshbhai M. Patel, Lab. Technician, Physics Department, Hem. N. G. University, Patan for helping me in printing and binding of my thesis. I am also thankful to our respectable Prof. K.N.Patel, Co-ordinator Physics, Hem. N.G.University, Patan for boosting my enthusiasm.

I am highly thankful to my all professors of Gujarat University, A'bad, for giving me a knowledge and information about the subject.

I am thankful and humbly appreciate the technical help of N.K.Modi and C.K.Panchal, P.R.L., A’bad,

Finally I am thankful to "GOD" and my all well wishers.

Contents

Abstract

At present in colleges, the universities and other technical institutions the microprocessor as a subject is inevitably taught. A large section of student learners and hobbyists wish to gain in depth knowledge in the field of microprocessor.

We have considered a very versatile Intel microprocessor 8085 for the present work. The designing of the basic microprocessor CPU card is conceived and developed. The complete hardware and related software have been developed. This has culminated into a basic trainer kit. The interfacing aspects of buffer chips, memory chips, I/O chips along with keyboard and display devices have been presented in detail. The P.C.B. designing was also carried out using in-house facility. The detailed assembly language program for this system called monitor program is indigenously developed and successfully run on the system. The monitor program supports all basic features of a typical microprocessor trainer kit. In this kit the commands like Exam memory, Block move, Go and Execute, Single step, Exam register, Next and Previous are included.

We have used 8085A, 8279, 8255, 74373, 74245, 74155, 74138, FNDs, push button switches and few discrete components in the present kit. The system resources can be used for interfacing other circuits by using kit expansion connectors. The present work gives a complete idea of developing 8085 based basic system.

To create various applied types of interfacing gadgets, we have presented a number of interfacing modules. In these, the interfacing design circuits and related softwares have been discussed. This would very helpful to learners to develop their own interfacing modules based on 8085.

Following is the list of the interfacing modules developed by us.

- Digital-to-analog converter
- Analog-to-digital converter
- Running character display
- An intelligent control panel
- Analog voltage measurement without using ADC
- Logic controller
- Interfacing Timer 8253
- Interfacing USART 8251
- Interfacing PIC 8259

Thus the present work would be very useful to the learners because it represents every hardware detail with the complete radiate software.

CHAPTER – 1 INTRODUCTION

1.1 Objective of the present work.

The present advance technology has made the microprocessors a routine element in the industrial & domestic applications. Today's most of the "Intelligent" electronic devices use microprocessors.

The increasing usage of microprocessor in the society has aroused an atmosphere to know & learn about microprocessors and its applications. This has resulted in the inclusion of detailed study of microprocessors as a part of regular & routine syllabus in the colleges, technical institutions & universities.

Such increasing demand and inquisitiveness among the learners sprouted this work. In essence, we planned to design and fabricate an 8085 based system, which will guide the learners to understand the microprocessors right from the scratch. In addition to this we also planned to present how one can utilize the basic concepts of microprocessor in developing various interfacing circuits through representative interfacing examples.

In sum up, we can say that the objectives of the present work are:

1. To design the circuit of 8085 based CPU card.

2. To fabricate the 8085 based CPU card as per the above circuit design. This will be consisting of
   - 8085 CPU & Supporting chips.
   - Memory made up of EPROM 2764 & RAM 6264.
   - I/O section comprised of 8255.
   - Key board & Display interfacing section using 8279.

3. To design & fabricate various interfacing modules as follows.
   - An intelligent control panel.
   - Analog voltage measurement without using ADC.
   - Digital to analog converter.
   - Analog to digital converter.
   - USART 8251.
   - Programmable interval timer/counter 8253.
   - Interrupt controller – 8259.
   - Logic controller.
   - Running character display.

Some times very interesting question arises, as why people go for the 8085 based systems? The answer to this question in our context is as follows.

The reasons of selecting 8085 are as follows.

1. It is a chip which includes all basic features of general microprocessors.
2. It is very popular.
3. Its literature is available in abundance.
4. Many colleges and university departments teach 8085.
5. Large number of students and learners are interested to gain systematic knowledge of 8085.

1.2 Literature survey

There are number of books available serving as a text-book or reference books on the 8085 microprocessors. Also, data books and manuals of various manufacturers are available. The list of manufacturer is as follows.

List of manufacturers

1. EMM semiconductor
   Hawthorne, CA 90250, USA
   .
2. MOS technology
   Norristown, PA 19401, USA.
3. Texas instruments
   Houston, TX 77001, USA.
4. Mostek
   Carolton, TX 75006, USA.
5. SGS – ATES
   Waltham, MA 02154, USA.
6. General Instruments (GI)
   Hicksville, NY 16002, USA.
7. Advanced Micro Devices (AMD)
   Sunnyvale, CA 94608, USA.
8. Rockwell international
   Anaheim, CA 92803, USA.
9. RCA
   Sommerville, NJ 08876, USA.
10. Intel corporation
    Santa Clara, CA 95051, USA.

11. American Microsystem (AMI)
    Santa Clara, CA 95051, USA.

12. Signetics
    Scottsdale, AZ 85252, USA.

13. Fairchild
    San Jose, CA 95110, USA.

14. Western digital corporation
    Newport Beach, CA 92663, USA.

15. Zilog
    Cupertino, CA 95014, USA.

16. Sharp
    Paramus, NJ 07652, USA.

17. Motorola
    Phoenix, Arizona 85006, USA.

Some of the useful books are listed as follows.

[1] Microprocessor architecture, programming and applications by R.S.gaonkar

[2] Microprocessor data handbook, Revised Edition, BPB Publications

[3] 0000 to 8085 Introduction to microprocessors for engineers and scientists by P.K.Ghosh and P.R.Shridhar, second edition, PHI Publications.

[4] Printed Circuit Board by Dr.H.N.Pandya, Published by Gujarat Granth Nirman Board, Ahmedabad,India.

The list of research publications is as follows.

[1] Study of interfacing module to establish communications between a PC and a 8085 based microprocessor kit. LE, Lab Experiments, Volume-4,No-3,September-2004.

[2] Study of 8255 through experiments using microprocessor kit. LE, Lab Experiments, Volume-4,No-2,June-2004

[3] 8085 based novel software technique for V to F type analog to digital conversion. Electronic Maker, June-2004

[4] Understanding interrupts of 8085 using Logic Analyzer. ETA-2004,25th & 26th FEB.-2004, Computer Department, Saurashtra University,Rajkot.

[5] Design and construction of analog to digital conversion interfacing module for 8085 microprocessor kit. 18th Gujarat Science congress, Physics Department, Saurashtra University, Rajkot dated 13th March,2004.

[6] An intelligent control panel : A novel microprocessor based system. ETA-2003, 11th , 12th ,13th July,2003, Computer Department, Saurashtra University, Rajkot.

[7] An interfacing module for 8085 based systems for solving Boolean equations".(communicated)

[8] Demystifying running character display.(communicated)

The list of web-sites of interest is as given below.

[1] www.standardproducts.philips.com

[2] www.qsl.net

[3] www.xs4all.nl

[4] www.cpu-world.com

[5] www.pearson.ch

[6] www.svnit.ac.in

[7] www.pearsoned.co.uk

[8] bookweb.kinokuniya.co.jp

[9] www.ses.co.il

[10] www.iinf.polsl.gliwice.pl

In the present work the microprocessor manufactured by Intel Corporation is planned to be used. In the present work, the processor of 8085 family is selected.

To design an 8085 based education microprocessor kit, the technical information of the processor 8085 and its family chips are required. This literature is available from the Intel Corporation.

Many industrial microprocessor kits are available, but these kits are having some limitations from educational point of view.

The websites surfed are as given below.

CHAPTER-2 BASIC UNDERSTANDING OF 8085 MICROPROCESSOR

The Intel corporation launched its 8-bit microprocessor series starting with 8080.[1,2] After 8080, it introduced a series of 8085 microprocessor. This series is available as 8085, 8085A, 8085AH, 8085AH-1, 8085AH-2.

The operating frequencies are 3MHz, 5MHz and 6MHz. Various versions of 8085 differ only in power consumption, speed and in timing signals. To site the difference 8085AH group has following instruction cycle differences.

8085AH - 1.3Microsecond instruction cycle.
8085AH-2 - 0.8Microsecond instruction cycle.
8085AH-1 - 0.67Microsecond instruction cycle.

We have concentrated on 8085A. But for all practical purposes the other versions can be used. The salient features of 8085A are as follows.

1. It needs only +5V power supply.
2. It uses single clock.
3. It is a 40-pin device.
4. It has a 16-address lines, which are divided into two groups. AD_0 to AD_7 and A_8 to A_{15}. The lower 8-lines AD_0 to AD_7 are multiplexed lines, i.e. at one time they work as address line A_0 to A_7 and at other time they work as data lines D_0 to D_7.
5. It has 8-addressable Registers, A, B, C, D, E, H, L, F and two 16-bit registers SP and PC.
6. It has three status lines: $IO/\overline{M}$, S_1, S_0 and three control lines: $\overline{RD}$, $\overline{WR}$ and $\overline{INTA}$.
7. For DMA operation it provides two lines, HOLD and HLDA.
8. Two lines $\overline{RESET\ IN}$ and RESET OUT are available for Reset operation.
9. It provides five hardware interrupts that is TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.
10. For serial communication two lines SID and SOD are provided.
11. For slower devices a synchronizing line READY is available.

2.1 The Block diagram

For the general understanding purpose the Intel has provided the internal working details of 8085A in turms of some basic units. We call this pictorial presentation of units as block diagram of 8085A.

In Figure 2.1.1 details of the block diagram of 8085A is depicted.[3]

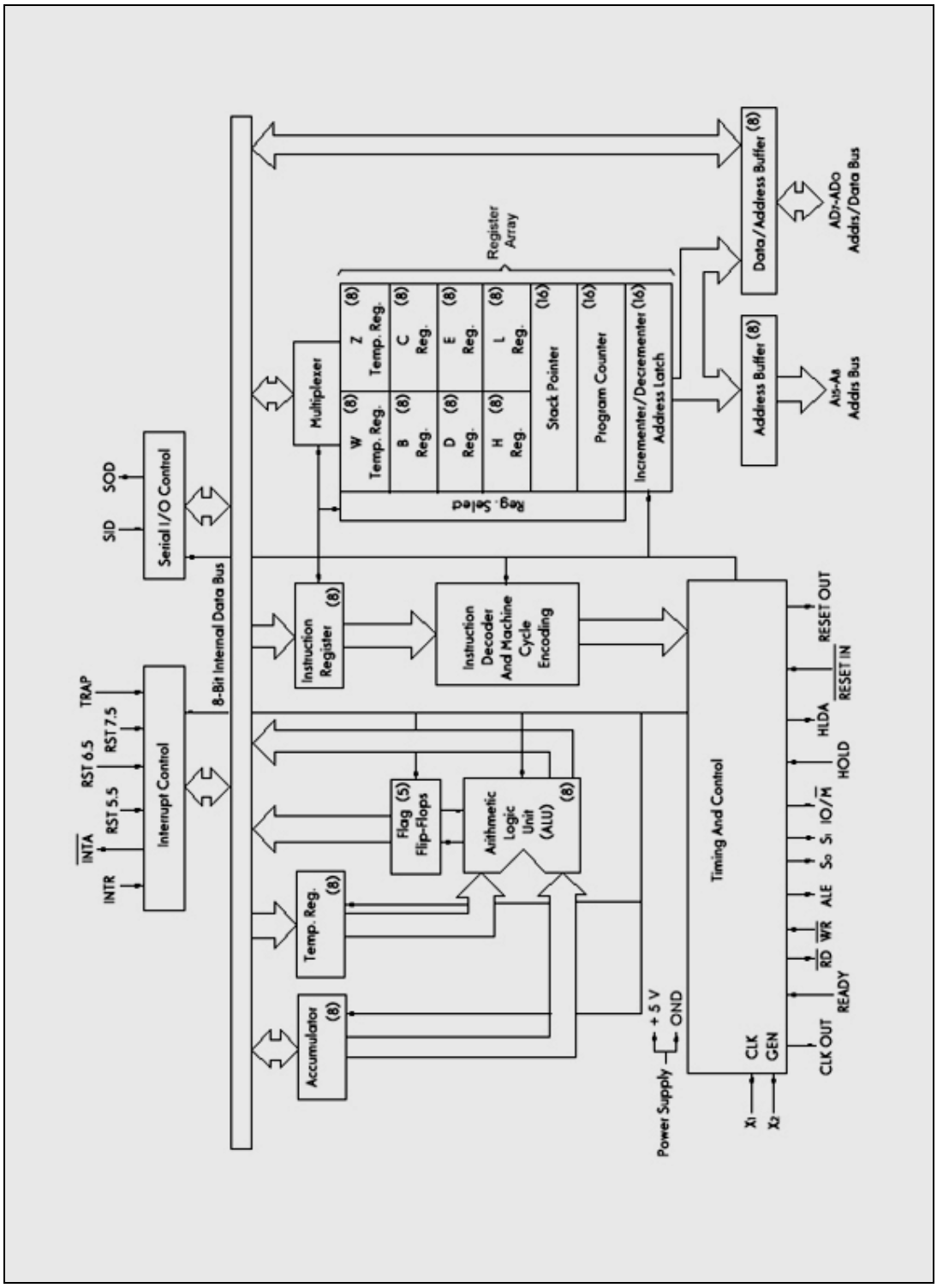


Figure-2.1.1 Block diagram of 8085

The block diagram is divided into various groups as follows.

1. ALU unit with its family registers.
2. Internal Registers.

3. Instruction Decoder and Machine cycle encoder along with IR (Instruction Register).
4. Timing and control unit with associated pins.
5. Interrupt control block.
6. Serial I/O control.
7. Address Data Buffers.

1. ALU unit with its family registers

The ALU unit that is Arithmetic and Logic unit is an 8-bit unit. It is useful for performing all arithmetic and logical operations. It uses two 8-bit registers accumulator and a temporary register as input registers. The output of this unit generally goes to accumulator through internal bus for certain instructions. It uses flag register to declare the outcome of arithmetic and logical operations by Setting or Resetting the certain flags.

2. Internal Registers

8085A has eight 8-bit registers, these are A, B, C, D, E, H, C and flag. These are used as general purpose registers. They are used to store data temporarily. The flag register has following flag bits.

D7	D6	D5	D4	D3	D2	D1	D0
S	**Z**	**X**	**AC**	**X**	**P**	**X**	**CY**

X means don't care.

FIGURE-2.1.2 Flag register bits

1. Carry flag (bit D0)

This is D0 bit of flag register. When arithmetic or logical operations are performed such that they generate Carry, this bit is set, otherwise remains Reset. In other words, for Carry CY=1 and for NO-Carry CY=0.

2. Parity flag (P - Bit D2)

If the number of 1's in accumulator are even, parity flag is set (P=1). If the number of 1's is odd the parity flag is cleared (P=0).

3. Auxiliary Carry (AC - bit - D_4)

If Carry is generated while adding bit 3 of accumulator, AC is set otherwise reset. This is used in BCD arithmetic.

4. Zero flag (Z - bit - D_6)

If the result of execution of an instruction is zero this flag is set (Z=1) other wise cleared (Z=0)

5. Sign flag (S - bit - D_7)

It provides the sign of data. If S = 0 the data is positive and if S = 1 the data is negative.

Note that accumulator is used as one of the default input for arithmetic and logical instructions. It stores the result of such instructions. For IN and OUT instructions ,it used as default destination and source respectively.

The two 16-bit registers SP and PC are special purpose registers. SP takes care of the stack management, while PC always point to the instruction codes in the memory.

In addition to these registers, register unit has a facility to increment or decrement the contents of SP and PC.

3. Instruction Decoder and Machine cycle encoder along with IR (Instruction Register)

When the codes of the instructions are fetched from the memory they go first to the instruction register within 8085A, one at a time. After that a series of decoder circuitry “Interprets” the codes. This interpretation will decide how many machine cycles and which types of machine cycles are needed for the instruction under consideration. This information is passed to timing and control unit for proper action.

4. Timing and control unit with associated pins

The signals received from instruction decoder and machine cycle encoder are analyzed and proper circuits are activated with reference to the clock and with the use of various control signals to execute the instruction. This unit controls the over all functions of all other units and maintains the timing among them selves. Following pins are directly associated with this unit.

X_1, X_2, CLK OUT, READY, $\overline{RD}$, $\overline{WR}$, ALE, S_0, S_1, IO/$\overline{M}$, HOLD, HLDA, $\overline{\text{RESET IN}}$ and RESET OUT.

5. Interrupt Control Block

This block controls the over all management of hardware interrupts of 8085. It receives interrupts from TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR. It provides INTA signal to device interrupting on INTR. A better understanding of interrupts can be obtained from the literature. [15]

6. Serial I/O Control Block

8085A provides a primary facility to perform serial communication using its SID and SOD pins. Data bits are received on SID and transmitted on SOD. For this 8085A uses SIM and RIM instructions.

7. Address Data Buffers

Address on address lines is provided by PC. Before O’s and 1’s of address appear on address lines they are buffered by address and data / address buffers.

PIN Diagram of 8085

Here each pin has its own importance. But before use of this IC, we must know the function of each pin. The pin diagram of 8085 is shown in Figure-2.1.3.[4]

- Pin No – 1 & 2 : X1 and X2

Generally crystal is connected between these two pins as shown in figure 2.1.4. We can also connect LC or RC network instead of crystal. The external clock source can also be given.

Here 20pf capacitors are required, if crystal frequency is bellow 4MHz.The body of the crystal is also grounded to ground electromagnetic pickup.

This typical value of capacitance given in data book of 8085, and the reason for these capacitors is to assure the oscillator start up at the correct frequency. This is shown in Figure 2.1.4.

- Pin – 3 : Reset out

This signal is used to reset the peripheral ICs. When microprocessor resets this signal activates.

- Pin – 4 & 5 : SOD & SID

These two pins are used for serial communication of data.
SOD means serial output data and SID means serial Input data.
SID is used to get data in and SOD is used to put data out serially.

- Pin – 6 :TRAP

This trap is a non-maskable interrupt. When low to high transition takes place on this pin and high status is received till interrupt is recognized corresponding interrupt routine will be immediately called.

- Pin – 7, 8, 9, 10 : RST 7.5, RST 6.5, RST 5.5 and INTR

These all can be considered as maskable interrupt pins but the priority is different.

The priority of Trap is highest then RST 7.5, RST 6.5, RST 5.5 and then INTR. When interrupt signal is received at any interrupt pin & if interrupt is enabled, its corresponding routing will be called.

- Pin – 11 : $\overline{INTA}$

This pin is known as interrupt acknowledge. When high status is received at pin-10 (INTR) the microprocessor sends low signal on this pin. This signal acknowledges that the interrupt is accepted by microprocessor.

- Pin – 12 to 19 : $AD_0 – AD_7$

These eight pins are known as multiplexed address/data lines. Here AD_0 means initially the pin is an address A_0 and after sometime it becomes data D_0.To use Address and Data, demultiplexing is required, whenever we are going to use 8085 in any system.

- Pin – 20 : Vss

This pin should be connected with ground.

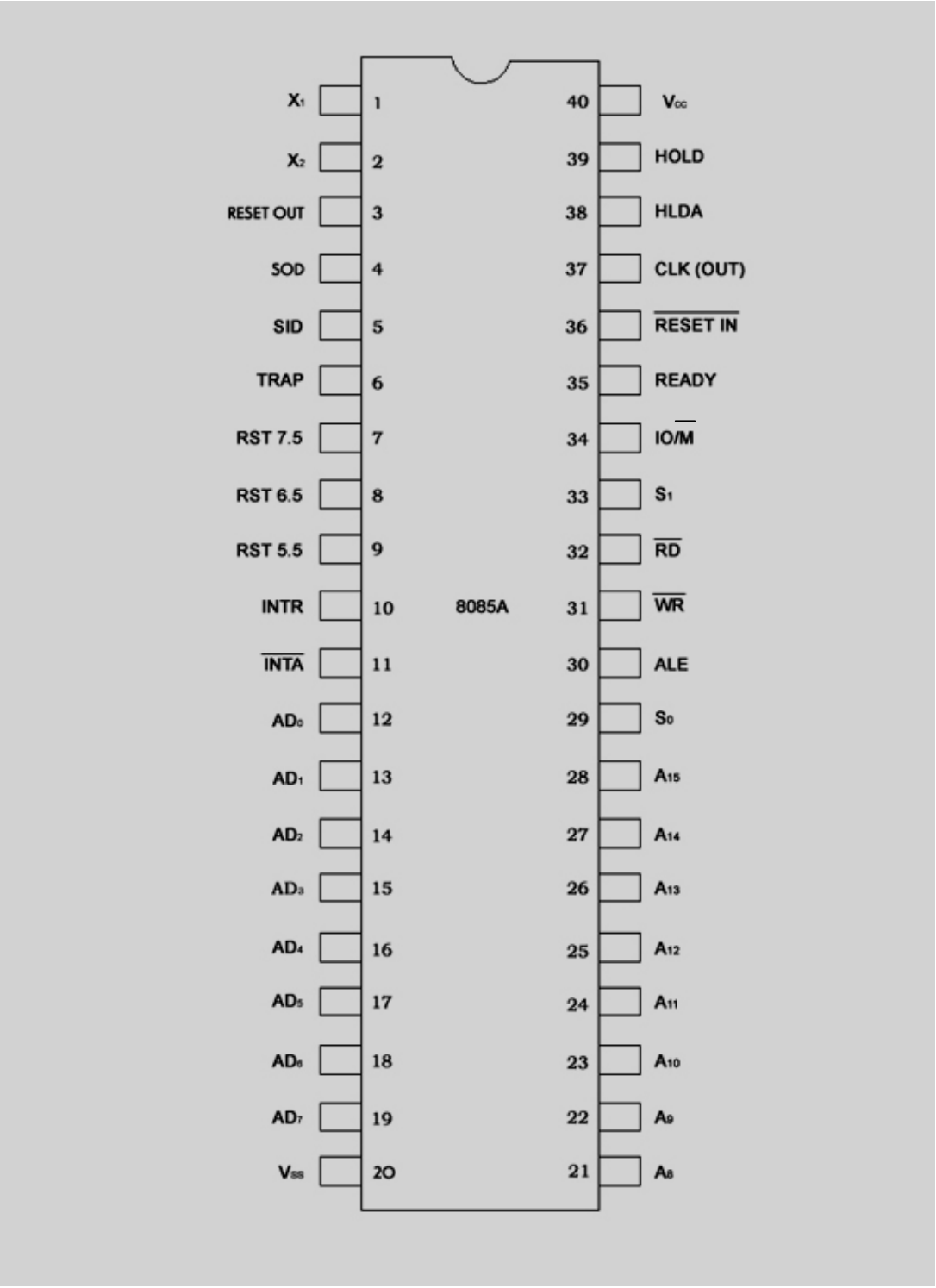


Figure 2.1.3 : PIN DIAGRAM OF 8085A

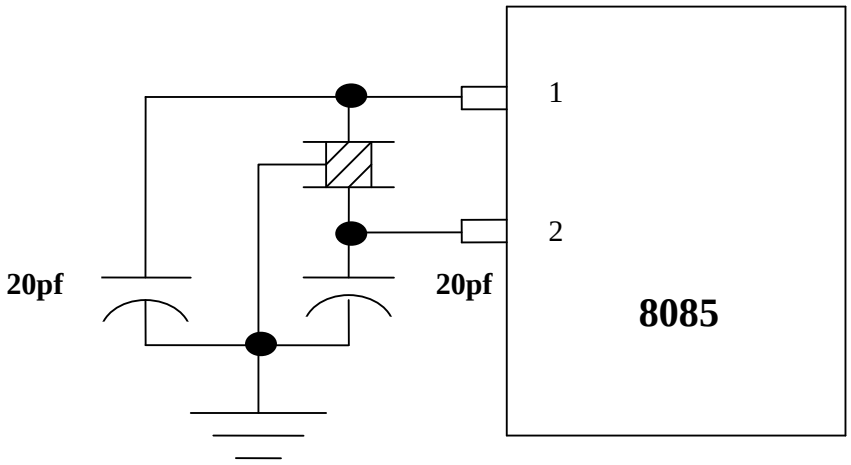


FIGURE – 2.1.4 CRYSTAL CONNECTION

- Pin 21 to 28 : A_8 to A_{15}

These 8-pins are known as upper order address pins.
Address bus is having total 16-bits and these are upper 8-bits, 8-lower bits are available after demultiplexing by the latch.

- Pin – 29 : S_0

This is lower status pin. It is rarely used in system.

- Pin – 30 : ALE

The full form of ALE is Address Latch Enable this.
ALE signal is used for demultiplexing of AD_{0-7}.

- Pin – 31 : $\overline{WR}$

This is known as write signal. This pin goes low when something is going to write into either memory or output port.

- Pin – 32 : $\overline{RD}$

This is known as Read signal. This pin goes low when something is going to read from either memory or input.

- Pin – 33 : S_1

This is also a status signal. It is rarely used in system.

- Pin – 34 : IO/$\overline{M}$

This pin is used as input-output or memory selection. When this pin has high logic microprocessor may perform Input or Output port related operation. If this pin has low logic means microprocessor may perform memory related operation. Sometimes it may be in high impedance state means neither high nor low. This high impedance state is also known as tri-state.

- The following Table – 1 shows the Status on Status signals IO/M, S_1 and S_0 and status on control signal RD, WR and INTA.

Table – 1

Operation	Status Signals			Control Signals		
	IO/$\overline{M}$	S1	S0	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
Opcode Fetch	0	1	1	0	1	1
Memory Read	0	1	0	0	1	1
Memory Write	0	0	1	1	0	1
Input Read	1	1	0	0	1	1
Output Write	1	0	1	1	0	1
Interrupt Acknowledge	1	1	1	1	1	0
Halt (Low on ready)	Tri-state	0	0	Tri-state	Tri-state	1
Hold	Tri-state	X	X	Tri-state	Tri-state	1
RESET	Tri-state	X	X	Tri-state	Tri-state	1

- Pin – 35 : Ready

If this pin is high microprocessor is working normally. But when this pin will be low, microprocessor will go into wait state for integer number of clock pulses.

- Pin – 36 : $\overline{\text{Reset in}}$

This is active low pin. When signal on this pin will be low the microprocessor will be reset. On Reset the program counter starts execution from address 0000H.

- Pin – 37 : CLK -OUT

Internal operating frequency is available on this pin. We can give this signal to any peripheral by using this pin for synchronization.

- Pin – 38 : HLDA

When HOLD signal is received, an acknowledgement is issued by this pin.

- Pin-39 : HOLD

A high on this pin suspends normal CPU operation, and the CPU relinquishes hold of buses.

- Pin-40 : Vcc

This pin should be connected with +5 volt.

2.2 Interrupt

Interrupts are primarily provided to reduce the burden on processor. In this scheme processors are not supposed to continuously check the I/O devices. Whenever I/O device gets ready to communicate with processor, It intimates the same through any of the interrupt lines.8085 has following interrupts.

1. TRAP
2. RST 7.5
3. RST 6.5
4. RST 5.5
5. INTR

They are listed as per their priorities .INTR, RST 5.5 and RST 6.5 are level sensitive, while RST 7.5 is rising edge sensitive. TRAP is sensitive to rising edge and level both.

RST 5.5, RST 6.5 and RST 7.5 are maskable interrupts, while TRAP is non maskable interrupt. The former are affected by EI and DI while letter is not affected by EI and DI.

The vectored addresses of interrupts are as follows.

Interrupt	**ISR address**
TRAP	0034h
RST 7.5	003Eh
RST 6.5	0034h
RST 5.5	002Ch

2.3 Serial Communication

We know that in serial communication bits are communicated one by one with every clock pulse. In other words for every clock pulse one bit is communicated. This means either input or output over one channel can be performed.

In 8085A this is achieved in a slightly different manner. Two special pins SID and SOD are provided, for inputting and outputting data respectively. These pins function with the help of execution of special instructions: RIM for inputting the data and SIM for outputting the data. Thus contrary to usual serial communication each bit on SID or SOD is communicated every time when RIM or SIM is executed not with reference to clock.

Whenever RIM instruction is executed the data on SID pin (either '0' or '1') gets transferred to the 7^{th} bit of accumulator i.e. A_7. Similarly, When SIM instruction is executed the data in 7^{th} bit of accumulator (A_7) gets transferred at SOD pin.

CHAPTER-3 INSTRUCTION SET

Every microprocessor understands a set of predefined command which are nothing but typical combination of 0's and 1's i.e. binary numbers. The collection of all these binary numbers or codes makes a complete set of instructions.

The manufacturers of the microprocessors classify this set into various groups, based on the common functionality of the instructions. The instruction set of 8085A also can be divided into various groups as follows.

1. Data transfer group.
2. Arithmetic group.
3. Branch group.
4. Logic group.
5. Stack input/output and machine control group.

The instructions of 8085A also can be categorized based on the number of bytes needed for concerned instruction. These categories are

1. 1-byte instruction.
2. 2-byte instruction.
3. 3-byte instruction.

In microprocessors the arithmetic and logical operations are performed on operands which we can call as Data bytes. We know that actual operation is performed in arithmetic and logic unit of microprocessor.

For that operand needed are to be brought to this unit. The operands may be stored at various possible sources, that is they can be in registers of microprocessors or they can be in external memory or they can be in I/O device. Now to bring them to ALU, manufacturers provide different ways (through different instructions). These ways are nothing but different "Addressing modes".

3.1 Addressing modes

The 8085A has following five addressing modes.

1. Immediate.
2. Direct.
3. Register.
4. Register Indirect.
5. Inherent or implied.

1. Immediate addressing

In this mode required operand (Data) is available in the memory location "Immediately" after the OP-CODE location in the memory.

2. Direct addressing

In this addressing mode address of the operand is provided along with the instruction.

For example LDA 5020H, in this instruction 5020H is addressing memory where required operand is stored.

3. Register addressing

In this type of addressing mode operand is contained in the register or register pair. The register or register pair are part of the instruction. For example MOV B,C, INX H etc.

4. Register Indirect addressing

In this addressing mode the address of the operand is not given by direct means but indicated indirectly i.e. address of the operand is represented by a pair of registers. For example LDAX B. In this instruction accumulator is to be loaded with the operand which is stored at certain location of the memory, whose address is available in register pair BC.

5. Inherent addressing or Implied addressing mode

In this addressing mode no operand is required. The instructions are self sufficient regarding the operations. For example STC. In this instruction it is clearly implied that there is no need of 8-bit operand. But operation is to be performed on a carry bit.

3.2 Instruction Format

In 8085A the general format of instruction can be classified considering the number of bytes needed for a given instruction.

These are

1. One byte instruction.
2. Two byte instruction.
3. Three byte instruction.

In single byte format the concern byte is an op-code byte. In two byte instruction format, the first byte is an op-code and the next is a data byte. In three byte instruction first byte is op-code byte, second byte is lower address byte of data and third byte is higher address byte of data.

Tables showing the instruction summary:

- **Move, Load and Store instructions**

Mnemonics	Instruction code							
	D7	D6	D5	D4	D3	D2	D1	D0
MOV R1, R2	0	1	D	D	D	S	S	S
MVI R	0	0	D	D	D	1	1	0
LXI B	0	0	0	0	0	0	0	1
LXI D	0	0	0	1	0	0	0	1
LXI H	0	0	1	0	0	0	0	1
STAX B	0	0	0	0	0	0	1	0
STAX D	0	0	0	1	0	0	1	0
LDAX B	0	0	0	0	1	0	1	0
LDAX D	0	0	0	1	1	0	1	0
STA	0	0	1	1	0	0	1	0
LDA	0	0	1	1	1	0	1	0
SHLD	0	0	1	0	0	0	1	0

LHLD	0	0	1	0	1	0	1	0
XCHG	1	1	1	0	1	0	1	1

- **Stack related instructions**

Mnemonics	Instruction code							
	D7	D6	D5	D4	D3	D2	D1	D0
PUSH B	1	1	0	0	0	1	0	1
PUSH D	1	1	0	1	0	1	0	1
PUSH H	1	1	1	0	0	1	0	1
PUSH PSW	1	1	1	1	0	1	0	1
POP B	1	1	0	0	0	0	0	1
POP D	1	1	0	1	0	0	0	1
POP H	1	1	1	0	0	0	0	1
POP PSW	1	1	1	1	0	0	0	1
XTHL	1	1	1	0	0	0	1	1
SPHL	1	1	1	1	1	0	0	1
LXI SP	0	0	1	1	0	0	0	1
INX SP	0	0	1	1	0	0	1	1
DCX SP	0	0	1	1	1	0	1	1

- **Jump instructions**

Mnemonics	Instruction code							
	D7	D6	D5	D4	D3	D2	D1	D0
JMP	1	1	0	0	0	0	1	1
JC	1	1	0	1	1	0	1	0
JNC	1	1	0	1	0	0	1	0
JZ	1	1	0	0	1	0	1	0
JNZ	1	1	0	0	0	0	1	0
JP	1	1	1	1	0	0	1	0
JM	1	1	1	1	1	0	1	0
JPE	1	1	1	0	1	0	1	0
JPO	1	1	1	0	0	0	1	0
PCHL	1	1	1	0	1	0	0	1

- **Call instructions**

Mnemonics	Instruction code							
	D7	D6	D5	D4	D3	D2	D1	D0
CALL	1	1	0	0	1	1	0	1
CC	1	1	0	1	1	1	0	0
CNC	1	1	0	1	0	1	0	0
CZ	1	1	0	0	1	1	0	0
CNZ	1	1	0	0	0	1	0	0
CP	1	1	1	1	0	1	0	0

CM	1	1	1	1	1	1	0	0
CPE	1	1	1	0	1	1	0	0
CPO	1	1	1	0	0	1	0	0

- **Return instructions**

Mnemonics	Instruction code							
	D7	D6	D5	D4	D3	D2	D1	D0
RET	1	1	0	0	1	0	0	1
RC	1	1	0	1	1	0	0	0
RNC	1	1	0	1	0	0	0	0
RZ	1	1	0	0	1	0	0	0
RNZ	1	1	0	0	0	0	0	0
RP	1	1	1	1	0	0	0	0
RM	1	1	1	1	1	0	0	0
RPE	1	1	1	0	1	0	0	0
RPO	1	1	1	0	0	0	0	0

- **Restart, Input and output instructions**

Mnemonics	Instruction code							
	D7	D6	D5	D4	D3	D2	D1	D0
RST N	1	1	A	A	A	1	1	1
IN	1	1	0	1	1	0	1	1
OUT	1	1	0	1	0	0	1	1

- **Increment and decrement instructions**

Mnemonics	Instruction code							
	D7	D6	D5	D4	D3	D2	D1	D0
INR R	0	0	D	D	D	1	0	0
DCR R	0	0	D	D	D	1	0	1
INX RP	0	0	D	D	0	0	1	1
DCX RP	0	0	D	D	1	0	1	1

- **Addition and subtraction instructions**

Mnemonics	Instruction code							
	D7	D6	D5	D4	D3	D2	D1	D0
ADD R	1	0	0	0	0	S	S	S
ADC R	1	0	0	0	1	S	S	S
ADI	1	1	0	0	0	1	1	0
ACI	1	1	0	0	1	1	1	0
DAD RP	0	0	D	D	1	0	0	1
SUB R	1	0	0	1	0	S	S	S
SBB R	1	0	0	1	1	S	S	S
SUI	1	1	0	1	0	1	1	0

SBI	1	1	0	1	1	1	1	0

- **Logical instructions**

Mnemonics	Instruction code							
	D7	D6	D5	D4	D3	D2	D1	D0
ANA R	1	0	1	0	0	S	S	S
XRA R	1	0	1	0	1	S	S	S
ORA R	1	0	1	1	0	S	S	S
CMP R	1	0	1	1	1	S	S	S
ANI	1	1	1	0	0	1	1	0
XRI	1	1	1	0	1	1	1	0
ORI	1	1	1	1	0	1	1	0
CPI	1	1	1	1	1	1	1	0

- **Rotate instructions**

Mnemonics	Instruction code							
	D7	D6	D5	D4	D3	D2	D1	D0
RLC	0	0	0	0	0	1	1	1
RRC	0	0	0	0	1	1	1	1
RAL	0	0	0	1	0	1	1	1
RAR	0	0	0	1	1	1	1	1

- **Special instructions**

Mnemonics	Instruction code							
	D7	D6	D5	D4	D3	D2	D1	D0
CMA	0	0	1	0	1	1	1	1
STC	0	0	1	1	0	1	1	1
CMC	0	0	1	1	1	1	1	1
DAA	0	0	1	0	0	1	1	1

- **Control instructions**

Mnemonics	Instruction code							
	D7	D6	D5	D4	D3	D2	D1	D0
EI	1	1	1	1	1	0	1	1
DI	1	1	1	1	0	0	1	1
NOP	0	0	0	0	0	0	0	0
HLT	0	1	1	1	0	1	1	0
RIM	0	0	1	0	0	0	0	0
SIM	0	0	1	1	0	0	0	0

Here DDD, SSS, AAA are three binary bits from 000 to 111. DDD shows destination register code and SSS shows source register code. AAA shows interrupt number. DD shows 00 to 11 as destination register pair code.

3.3 All Instructions

In this chapter in the beginning we have divided the whole instruction set of 8085A into several groups. We will discuss the instructions of 8085A as per these groups.[5]

1. Data Transfer Group

The data can be transferred by the following types of instructions. The flags are not affected by these instructions.

MVI r1 Data -	The Data is loaded into register r.
MVI M1 Data -	The Data is immediately loaded into memory whose address is in HL pair.
MOV r1, r2 -	The Data of register r2 is transferred into r1.
MOV M, r -	The Data of register is transferred into memory whose address is given in HL.
MOV r, M -	The Data of memory whose address is in HL is transferred into register.
LXI rp1 Data -	The 16-bit data is immediately loaded into register pair rp.
STA address -	Store content of accumulator at given address.
LDA address -	Load content into accumulator from given address.
SHLD address -	Store HL register pair directly at the given address and address +1 locations.
LHLD address -	Load HL register pair directly from the given address and address +1 locations.
LDAX Rp -	Load the accumulator with the contents of the memory whose address is given into register pair Rp.
STAX Rp -	Store the content of accumulator into the memory whose address is given into register pair Rp.
XCHG -	Exchange the contents of register pair HL with DE.

2. Arithmetic Group

This arithmetic group can be classified into following categories.

1. Increment & Decrement group.
2. Addition group.
3. Subtraction group.
4. Decimal Adjust Accumulator group.

- **Increment & Decrement group**

INR r -	The content of register is incremented by one only carry flag is not affected.
INR M -	Increment the content of the memory location by one whose address is in HL Register pair only carry flag is not affected.

INX rp - Increment the content of the register pair rp by one. No flags are affected.

DCR r - The content of register is decremented by one only carry flag is not affected.

DCR M - Decrement the content of the memory location by one whose address is in HL Register pair only carry flag is not affected.

DCX rp - Decrement the content of the register pair rp by one. No flags are affected.

- **Addition group**

ADD r - Addition of the contents of accumulator and register r. The answer is stored into accumulator. All flags are affected.

ADD M - Addition of the contents of accumulator and memory whose address is in HL pair. The answer is stored into accumulator. All flags are affected.

ADI Data - Addition of the contents of accumulator with the Data. The answer is stored into accumulator. All flags are affected.

ADC r - Addition of the carry flag with the contents of accumulator and register. The answer is stored into accumulator. All flags are affected.

ADC M - Addition of the carry flag with the contents of accumulator and memory (whose address is given in HL pair). The answer is stored into accumulator. All flags are affected.

ACI Data - Addition of the carry flag with the content of accumulator and Data. The result is stored into accumulator. All flags are affected.

DAD rp - Addition of the contents of register pair rp with the contents of the HL pair. The result is stored into HL. Only the carry flag is affected.

- **Subtraction group**

In this group of instructions borrow is used for carry flag.

SUB r - Subtract the content of register r from accumulator. The result is stored into accumulator. All flags are affected.

SUB M - Subtract the content of memory (whose location is given in HL pair) from accumulator. The result is stored into accumulator. All flags are affected.

SUI Data - Subtract the data from accumulator. The result is stored into accumulator. All flags are affected.

SBB r - Subtract the borrow and content of register r rom the content of accumulator. The result is stored into accumulator. All flags are affected.

SBB M - Subtract the borrow and the content of memory (whose address is given in HL pair) from the content of accumulator. The result is stored into accumulator. All flags are affected.

SBI Data - Subtract the borrow and the data from the content of accumulator. The result is stored into accumulator. All flags are affected.

- **Decimal Adjust group**

DAA - A Decimal Adjust Accumulator.
This instruction converts the binary content of accumulator into B.C.D. form.

3. Branch group

JMP address - This instruction jumps at address unconditionally. All flags are unaffected.

J cond. address - This instruction jumps at address conditionally. The instructions are as given below with conditions.

JC	Jump	if	C=1
JNC	Jump	if	C=0
JP	Jump	if	S=0
JM	Jump	if	S=1
JPE	Jump	if	P=1
JPO	Jump	if	P=0
JZ	Jump	if	Z=1
JNZ	Jump	if	Z=0

All flags remain unaffected.

PCHL- This instruction copies the contents of HL pair into PC. All flags remain unaffected.

CALL address - This instruction calls the subroutine unconditionally, after storing content of PC on stack. All flags remain unaffected.

C cond. address - This instructions call the subroutine conditionally, after storing content of PC on stack. The instructions are as given below with conditions.

CC -	Call	if	C=1
CNC-	Call	if	C=0
CP-	Call	if	S=0
CM-	Call	if	S=1
CPE-	Call	if	P=1
CPO-	Call	if	P=0
CZ-	Call	if	Z=1
CNZ-	Call	if	Z=0

All flags remain unaffected.

- Return

RET - This instruction returns from the subroutine unconditionally. The address is retrieved from the stack. All flags remain unaffected.

R cond - This instruction returns from the subroutine conditionally. The address is retrieved from the stack. All flags remain unaffected.

These conditional return instructions are as given below with conditions.

RC	Return	if	C=1
RNC	Return	if	C=0
RP	Return	if	S=0
RM	Return	if	S=1
RPE	Return	if	P=1
RPO	Return	if	P=0
RZ	Return	if	Z=1
RNZ	Return	if	Z=0

RST n - This instruction is similar to a call instruction for fixed address.

The fixed address are as given below.

Instruction	Destination Address
RST 0 -	0000H
RST 1 -	0008H
RST 2 -	0010H
RST 3 -	0018H
RST 4 -	0020H
RST 5 -	0028H
RST 6 -	0030H
RST 7 -	0038H

4. Logic group

The logical operations can be performed by using the following instructions.

ANA r - The content of accumulator is logically ANDed with content of register r. The result is stored into accumulator. All flags are affected. The carry will be cleared and the auxiliary carry will be set.

ANA M - The content of accumulator is logically ANDed with content of memory (whose address is given in HL pair). The result is stored into accumulator. All flags are affected. The carry flag is cleared and the auxiliary carry will be set.

ANI Data - The content of accumulator is logically ANDed with Data. The result is stored into accumulator. All flags are affected. The carry flag is cleared and the auxiliary carry will be set.

XRA r - The content of accumulator is logically exclusive 0Red with the content of register r. The result is stored into accumulator. All flags are affected. The carry and auxiliary carry flags are reset.

XRA M - The content of accumulator is logically exclusive 0Red with the content of memory (location in HL pair). The result is stored into accumulator. All flags are affected. The carry and auxiliary carry flags are reset.

XRI Data - The content of accumulator is logically exclusive ORed with the data. The result is stored into accumulator. All flags are affected. The carry and auxiliary carry flags are reset.

ORA r - The content of accumulator is logically ORed with the content of register r. The result is stored into accumulator. All flags are affected. The carry and auxiliary carry flags are reset.

ORA M - The content of accumulator is logically 0Red with the content of memory (whose address is given in HL pair). The result is stored into accumulator. All flags are affected. The carry and auxiliary carry flags are reset.

ORI Data - The content of accumulator is logically ORed with the data. The result is stored into accumulator. All flags are affected. The carry and auxiliary carry flags are reset.

CMP r - This instruction compares the content of accumulator with content of register r. If both are same zero flag is set. If content of accumulator is less then content of register so carry flag is set. All fags are affected.

CMP m - This instruction compares the content of accumulator with content of memory (whose address is given in HL pair). If both are same zero flag is set. If content of accumulator is less then content of memory carry flag is set. All flags are affected.

CPI Data - This information compares the content of accumulator with data. If both are same so zero flag is set. If the content of accumulator is less then Data, so carry flag is set. All flags are affected.

STC - The carry flag can be set by this instruction.

CMC - The carry flag is complimented by this instruction.

CMA - The content of accumulator is complimented by this instruction.

- Rotate

RLC - Rotate content of accumulator left by one position. The content of carry flag is lost.

RRC - Rotate content of accumulator right by one position. The content of carry flag is lost.

RAL - Rotate content of accumulator left by one position through carry.

RAR - Rotate content of accumulator right by one position through carry.

5. Stack, I/O and machine control Instructions

PUSH rp - This instruction pushes(copies) the content of register pair on the stack. The content of stack pointer is decremented by two. Flags are unaffected.

PUSH PSW- The content of accumulator and flags(PSW means Program Status Word) are pushed on the stack. The content of stack pointer is decremented by two. Flags are unaffected.

POP Rp - The data from the top of stack is retrieved into register pair Rp. The stack pointer is incremented by two. Flags are unaffected.

POP PSW - The contents of accumulator and flags are retrieved from top of the stack. The stack pointer is incremented by two. Flags are unaffected.

XTHL - The data at locations pointed by SP and SP+1 are exchanged with registers L and H respectively. Flags are unaffected.

SPHL - The content of register pair H gets copied into stack pointer. Flags are unaffected.

Input / Output

IN Port - This instruction receives 8-bit data from port and stores into accumulator Flags are unaffected.

OUT Port- This instruction puts 8-bit data from Accumulator to the port. Flags are unaffected.

Machine Control

EI - This instruction sets the internal interrupt flip-flop of microprocessor. Flags are unaffected.

DI - This instruction resets the internal interrupt flip-flop of microprocessor. Flags are unaffected.

NOP - This instruction is not doing anything. It is also known an No-Operation. Flags are unaffected.

HLT - The microprocessor halts by this instruction. It will be restarted by interrupt or reset. Flags are unaffected.

SIM - The full form of this instruction is Set Interrupt Mask. This instruction works with the content of

accumulator. The meaning of all eight bits of accumulator is given below.

* D0,D1 and D2 mask RST 5.5,RST 6.5 and RST 7.5 respectively. ‘1’ means mask and ‘0’ means unmask.
* D3 should be ‘1’ to make D0,D1 and D2 effective.
* D4 resets RST 7.5 flip-flop.
* D5 is not used.
* D6 is serial output enable. It should be ‘1’ to enable serial output data.
- D7 bit transfers at SOD pin if D6 is ‘1’.

RIM - The full form of this instruction is Read Interrupt Mask. The status is copied into accumulator. The meaning of all eight bits of accumulator is as given below.

* D0,D1 and D2 set if RST 5.5,RST 6.5 and RST 7.5 respectively are masked.
* D3 sets if IE(interrupt enable)flag is set.
* D4,D5 and D6 set if RST 5.5,RST 6.5 and RST 7.5 respectively are pending.
* D7 reads the logic at SID(serial input data)pin.

CHAPTER-4 INTERFACING OF EPROM AND RAM

The 8085A processor has only a few internal register which can store data. In real life any user’s program cannot be accommodated in such internal register memory. Hence 8085A requires external memory.

Processor alone cannot make a useful system. It requires memory and input output facility. To be a useful microprocessor based system certain user's commands are to be a remembered permanently, so that every time system is initialized it can perform execution of few user defined commands.

For this system has to employ a sort of memory which can permanently store these commands, EPROM is such type of memory. Hence, it becomes very interesting to know that how one can interface such EPROMs with the processors.

On the other hand microprocessor based system (e.g. microprocessor trainer kit) does not required to remember permanently users program. For this purpose it requires some temporary memory which can perform read write operation, in order to execute user programs. This type of memory is available as RAM memory. So it is also interesting to know how to interface such memory chips with processor.

4.1 EPROM 2764A

It is a 8-bit wide memory. In the number 2764A, 64 indicates that it has total 64 Kbit memory, In bytes, it becomes 8Kx 8 memory size.[4]

4.1.1 BLOCK DIAGRAM AND FUCTIONAL DESCRIPTION

The block diagram of 2764A is shown in Figure 4.1.1.1.

The heart of the chip is a cell matrix of 65536 bits. This cell matrix is internally so arranged that it gives us 8KX8 memory. The 8K bytes require total 13-address lines. In 2764A this lines are A_0 to A_{12}. These address lines are internally applied to Y-decoder and X-decoder, which select the particular cell of cell matrix. The output of the selected cells is passed to output buffers. These buffers are again controlled by output enable pin. When $\overline{OE}$ is low the output buffers send the data on the output pins, O_0 to O_7.

The chip 2764A has four special pins V_{PP}, $\overline{OE}$, $\overline{PGM}$ and $\overline{CE}$.

The functions of above pins together decides the various modes of the operation of 2764A. Table-4.1.1.1. describes the detail of various modes of 2764A.

Table-4.1.1.1

Mode	Pins					
	$\overline{CE}$	$\overline{OE}$	$\overline{PGM}$	V_{PP}	V_{CC}	Output
Read	V_{IL}	V_{IL}	V_{IH}	V_{CC}	5.0V	D out
Output disable	V_{IL}	V_{IH}	V_{IH}	V_{CC}	5.0V	High – Z
Stand by	V_{IH}	X	X	V_{CC}	5.0V	High – Z
Programming	V_{IL}	V_{IH}	V_{IL}	12V	5.0V	D In
Program verify	V_{IL}	V_{IL}	V_{IH}	12V	5.0V	D out
Program inhibit	V_{IH}	X	X	12V	5.0V	High – Z

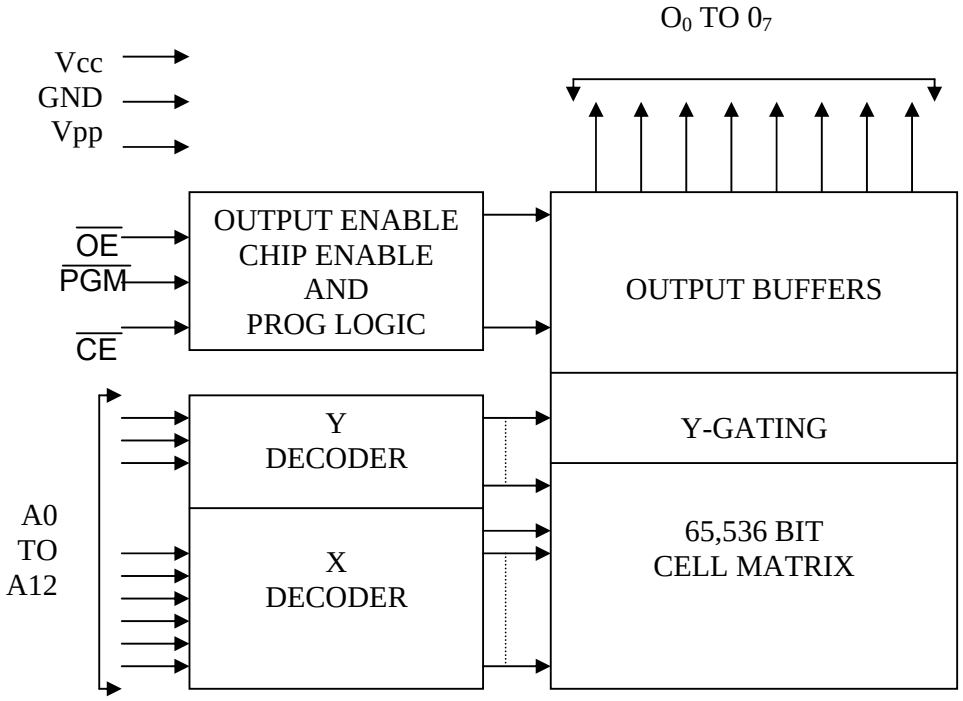


Figure 4.1.1.1 INTERNAL BLOCK DIAGRAM OF EPROM 2764A

In above table V_{IL} is the input low level voltage and V_{IH} is the input high level voltage.
(V_{IL} – 0.1V to 0.8V and V_{IH} = 2V to + V_{CC})

To program the 2764A we have to select the chip (CE = V_{IL}), inhibit the reading operation. (OE = V_{IH}), and PGM = V_{IL}. At the same time V_{PP} should be given +12V supply and V_{CC} should be given +5V.

Figure 4.1.1.2. shows the pin diagram of 2764A.

4.1.2 INTERFACING OF 2764A IN THE PRESENT SYSTEM

In the present microprocessor system design 8K byte of EPROM was considered sufficient to store the monitor program.

For interfacing 2764A with the 8085A a decoder 74155 and a latching IC 74L5373 are used. The interfacing circuit for the same is shown in Figure 4.1.2.1. Note that this circuit is a part of the detailed circuit diagram of the microprocessor trainer kit designed by us.

IC 74373 demultiplexes the AD0 to AD7 lines of 8085A, when ALE becomes high and it enables IC 74L5373 through its pin no.11(G), to latch the address A0 to A7. Active low enable pin no.1, that is OC of IC 74373 is permanently grounded. The output Q0 to Q7 of IC 74L5373 (i.e. A0 to A7 address lines.) are connected with the address lines A0 to A7 of EPROM 2764A. The rest of the address lines of 2764A, that is A8 to A12 address lines of 8085A.

The EPROM IC has a pin named $\overline{OE}$ which helps to read data out. This $\overline{OE}$ pin is directly connected with the $\overline{RD}$ of 8085A. Its $\overline{PGM}$ and Vpp are tied up with Vcc

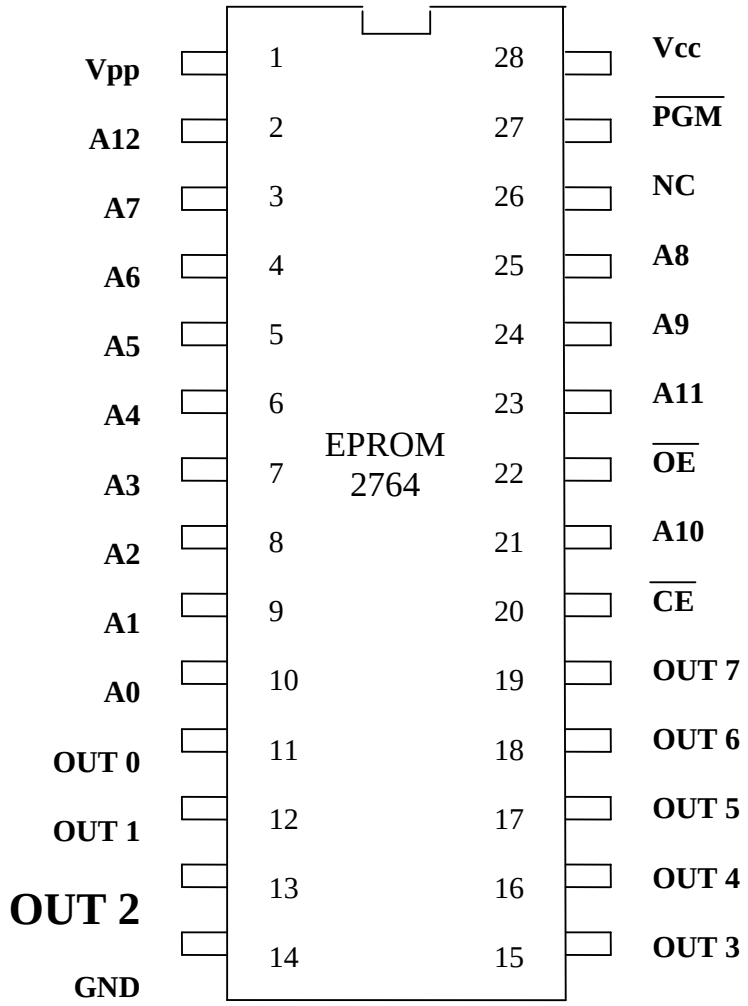


Figure 4.1.1.2 PIN DIAGRAM OF EPROM 2764A

The chip selection is controlled by the decoder 74155. In the present system design this 74155 is used to select the memory chips 2764A, 6264, and I/O chips 8279 and 8255.

To separate the memory and I/O devices the output pins of 74155 are divided into two groups that is 1Y0, 1Y1, 1Y2 and 1Y3 are for I/O device selection and 2Y0, 2Y1, 2Y2 and 2Y3 are for memory chip selection.

74155 has four enable pins out of which three are active low and one active high. Out of three active low pins two are permanently grounded. The remaining two (where one is active low and another is active high) are connected with IO/$\overline{M}$ pin of 8085A. Here active high is connected with enabling of outputs of internal decoder 1 of 74155. While active low is

connected with the internal decoder-2 of 74155. Hence whenever IO/M pin of 8085A is high, I/O devices are selected, and when IO/M is low, memory chips are selected.

Now the differentiation of I/O devices and memory chips is done through the input lines A and B of 74155, these A and B input lines of 74155 are directly connected with the A14 and A15 address lines of 8085A respectively. Following Table-4.1.2.1 explains the I/O device and memory chips selection details.

TABLE-4.1.2.1

IO/$\overline{M}$	**A (A14)**	**B (A15)**	**Selection**
0	0	0	EPROM 2764A (2Y0)
0	0	1	Unused (2Y2)
0	1	0	RAM 6264 (2Y1)
0	1	1	Unused (2Y3)
1	0	0	PPI 8255 (1Y0)
1	0	1	External card (1Y1)
1	1	0	KDI 8279 (1Y1)
1	1	1	Unused (1Y3)

Figure: 4.1.2.1 CONNECTION OF EPROM WITH 8085

4.2 STATIC RAM 6264

It is 8-bit wide read/write memory. Its capacity is 8KX8 bits.[4]

4.2.1 BLOCK DIAGRAM AND FUCTIONAL DESCRIPTION OF 6264

Figure 4.2.1.1 shows the internal block diagram of 6264 RAM chip. It has total 13-address lines i.e. A0 to A12.

As shown in the block diagram the 65536 bits are arranged in three dimensions as 8 Matrices of 256 X 32 rows & columns respectively. The address lines A_2, A_3, A_4, A_5, A_7, A_8, A_9 and A_{11} drive the row decoder while A_0, A_1, A_6, A_{10} and A_{12} drive the column decoder. There is unit called input data control which accepts 8-bit input on the lines DQ_0 to DQ_7 under the control of $\overline{E_1}$, E_2 and $\overline{W}$. The reading of the chip is done under the control of $\overline{E_1}$, E_2, $\overline{W}$ and $\overline{G}$.

Table 4.2.1.2. explains the various modes of the chip MCM 6264C.

Table – 4.2.1.2

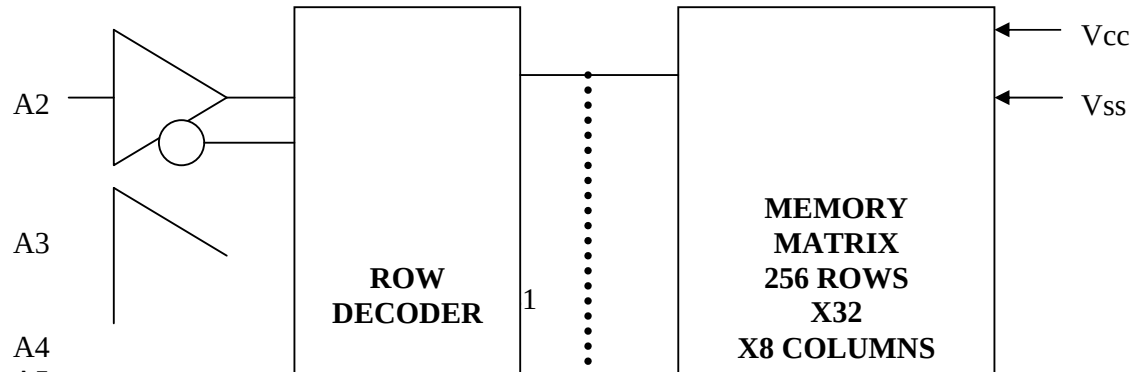
$\overline{E1}$	E2	$\overline{G}$	$\overline{W}$	Mode	Output
H	X	X	X	Not selected	High - Z
X	L	X	X	Not selected	High - Z
L	H	H	H	Output disabled	High - Z
L	H	L	L	Read	D out
L	H	X	X	Write	High - Z

Figure 4.2.1.2. shows the pin assignment of the 6264C.

4.2.2 INTERFACING OF 6264 IN THE PRESENT SYSTEM

In the present system, the interfacing circuit for RAM 6264 involves 74LSs373 and 74155 and 8085. ICs 74373 and 74155 are having the same connection details with the 8085A as discussed in section 4.1.2. The only difference in this one is that the A and B input lines of 74155 selects 6264 through output pin 2Y, by accepting A15, A14=0,1.

The detailed circuit diagram is shown in Figure 4.2.2.1 Note that since this is Read/Write memory one should connect RD and WR pins with 6264. In the present design WR is connected with WR and RD is connected with OE of 6264. 6264 has two chips select lines. CS1, and CS2. CS1 is driven by 74155 while CS2 is connected with +5volt.



—

—

—

Figure 4.2.1.1 INTERNAL BLOCK DIAGRAM OF RAM 6264

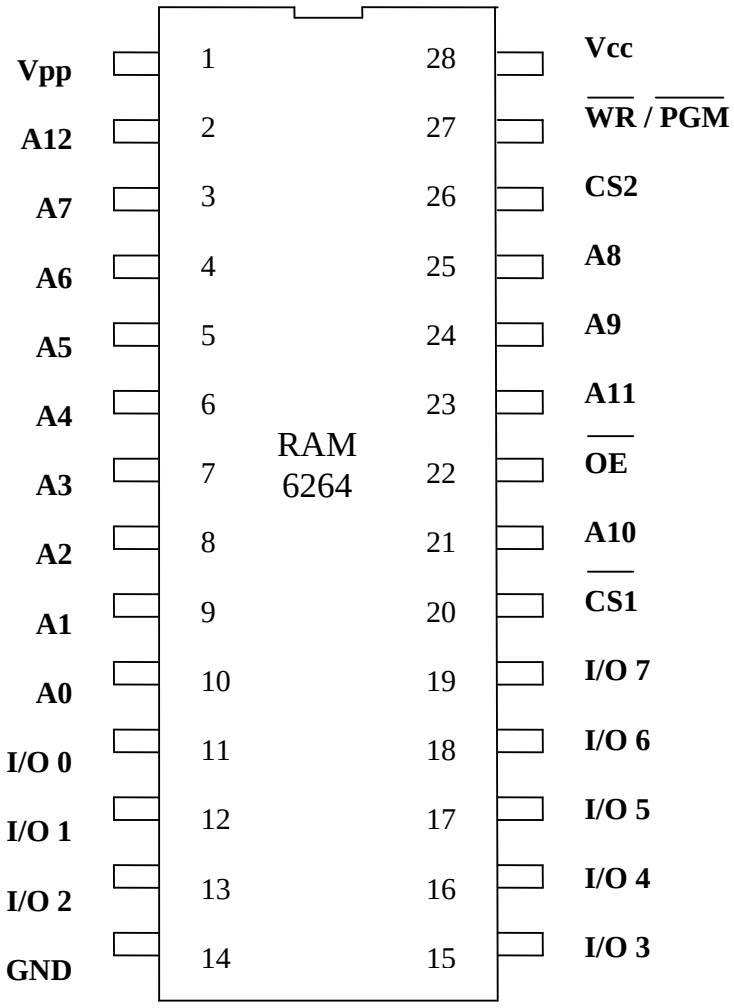


Figure 4.2.1.2. The pin assignment of the 6264

Figure 4.2.2.1 : CONNECTION OF RAM WITH 8085

CHAPTER – 5 I/O INTERFACING

For a given microprocessor system to execute the user's program, there should be a provision to allow the user to store these programs into microprocessor system's memory. This is generally done by interfacing a hex key pad with the microprocessor. User also needs to check the results of the execution of his program, for this purpose output device is used. Generally seven segment LEDs or LCD display are used in microprocessor trainer kits.

8085A based system is not just a calculating machine. It is also used for control applications. For this some I/O port lines are needed to communicate the control signals to and fro between microprocessor and the device.

Such interfacing of input device, output device and I/O port lines with the processors is called I/O interfacing.

In the present work the keyboard and display controller chip 8279 is used for interfacing the hex keyboard and seven segment display. Also the programmable peripheral interface chip 8255 is employed for interfacing I/O port lines with 8085A.

5.1 Programmable keyboard display interface 8279

This chip is useful to interface the keyboard and display with 8085 microprocessor. It has been introduced by Intel. [6]

The advantage of interfacing 8279 with 8085A over other techniques are as follows.

- It relives microprocessor from scanning keyboard & refreshing displays.
- It also takes care about debouncing of key and blanking display while scanning.
- It can perform simultaneous keyboard display operation.
- The scan time for display is programmable.
- It can operate in different modes.

5.1.1. Block diagram of 8279

Block diagram of 8279 is shown in Figure 5.1.1.
The block diagram is divided into following units.

1. Data buffer.
2. I/O control.
3. FIFO/sensor RAM status.
4. Display address Registers.
5. 16X8 Display RAM.
6. 8X8 FIFO/sensor RAM.
7. Scan counter.
8. Return.
9. Keyboard Debounce & control.
10. Display registers.
11. Control & Timing registers.
12. Timing & Control.

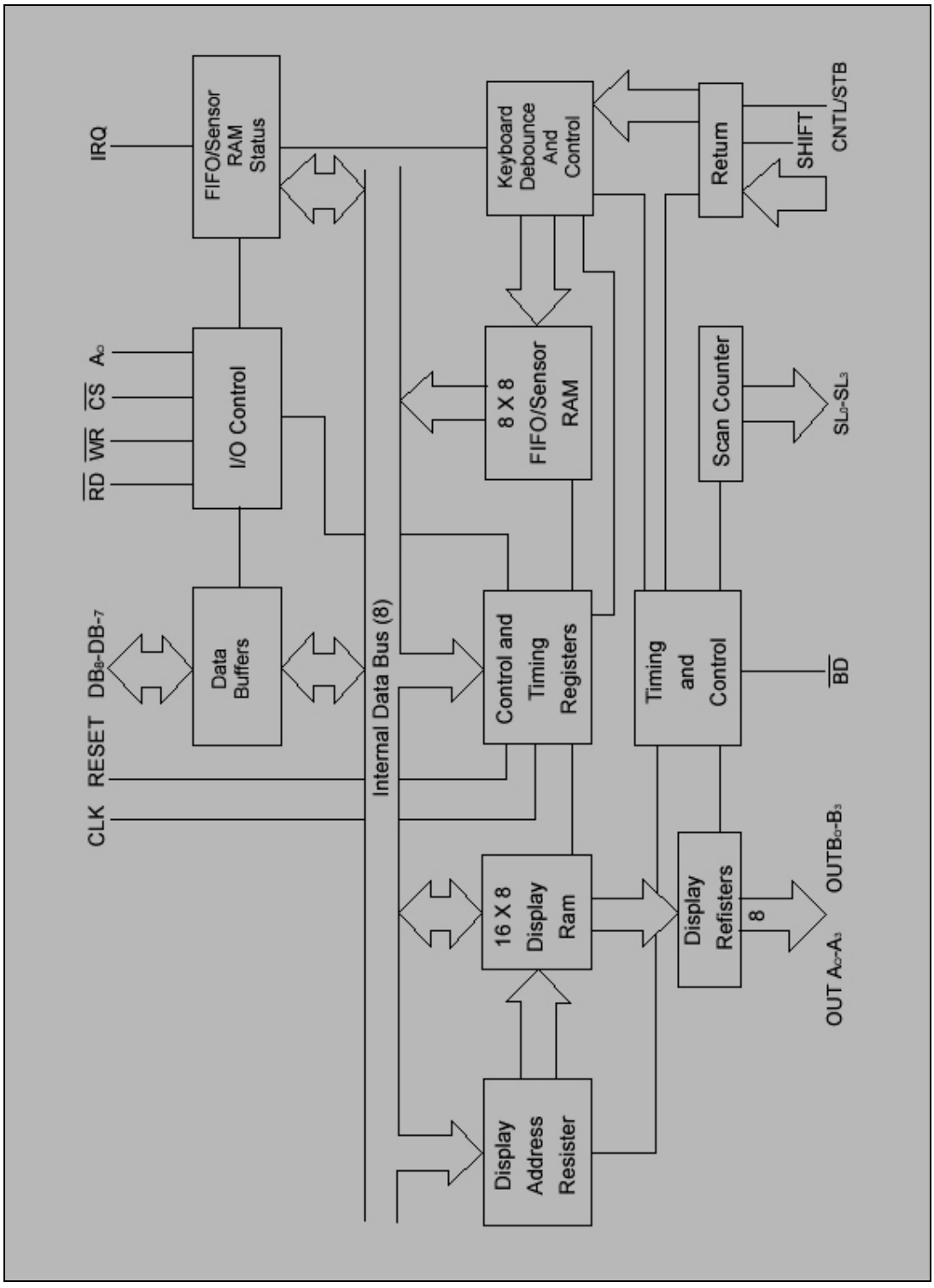


Figure 5.1.1 Block Diagram of 8279

1. Data buffer

This block interfaces the data lines of 8279 with the system data bus. It takes control words from 8085A to 8279 as well as data to and fro. It is represented by directional arrow.

2. I/O control

In this unit the pins $\overline{CS}$, A_0, $\overline{RD}$ and $\overline{WR}$ are associated. The combinations of low or high status of these signals determine the direction of data to and fro between 8085A and 8279.

Following Table 5.1.1. explains the various conditions of input output data communication between 8085A and 8279.

Table – 5.1.1

Signal Conditions				Meaning
$\overline{CS}$	A_0	$\overline{RD}$	$\overline{WR}$	
0	0	0	0	No meaning
0	0	0	1	System data bus <= D_0 – D_7 of 8279
0	0	1	0	No meaning
0	0	1	1	No meaning
0	1	0	0	Status word => system data bus
0	1	0	1	Control words from system data bus => D_0 – D_7 of 8279
0	1	1	1	No meaning
1	X	X	X	Chip not selected

3. FIFO/sensor RAM status

With this block only one pin of 8279 is associated and that is IRQ (interrupt request).

This block stores the status of FIFO/sensor RAM. It keeps the track of number of entries in FIFO RAM and provides and IRQ signal when the FIFO is not empty. In other words whenever any key is depressed and its corresponding code gets stored into FIFO RAM, and IRQ is generated. This is normally connected with any RST interrupt pins of 8085A. Proper service routine is executed on IRQ, which read the FIFO RAM. As the FIFO RAM is read the IRQ goes low.

The status of the FIFO RAM can be checked by reading of the status word.

4. Display address Registers

8279 has a capacity to store the display codes (8-bit) of the characters to be displayed on particular type of display. For this it contains an internal RAM memory of 16-bytes, exclusively for display character codes. Each location of this memory is addressed by one of the register “Display address registers”. We can say that this block contains a group of registers, which contains the address of various locations of display RAM. Note that these addresses are supplied by user through software programs by using the control words of 8279.

5. 16X8 display RAM

It is the internal read / write memory of 8279. It has 16-locations of 8-bits. It stores the character codes for display purpose.

6. 8X8 FIFO/sensor RAM

8279 is able to take in the codes generated by the key depression of the keyboard, as well as any change of sensor matrix, in lieu of keyboard. This block stores the consecutive 8-entries of keyboard/sensor matrix. This is a particular type of memory called FIFO.

The content of this FIFO can be read either by status check method or by interrupt method.

7. Scan counter

This unit has four pins associated with it. These are scan lines SL_0 to SL_3. This unit is working just like a counter and output of this counter appears on the lines SL_0 to SL_3. This counter is a programmable and works as encoded counter or decoded counter. When programmed as encoded counter the lines SL_0 to SL_3 can be connected as input to external decoder chip.

When it is programmed as decoded counter the decoding is done internally in the 8279 and the lines SL_0 to SL_3 behave as the output of internal decoder. It has another interpretation in encoded mode 16 displays (e.g. FNDs) can be interfaced where as in decoded mode only four display devices can be interfaced.

These lines are used to scan the keyboard and display.

8. Return

This block takes the status of the keys of the keyboard or any change of the sensor element of sensor matrix. There are ten different pins associated with this block, these are eight return lines,RL_0 to RL_7 and control and shift pins. RL_0 to RL_7 lines are generally connected with the columns of the keyboard matrix. The status of all the columns is stored as three bit code in FIFO RAM. Shift and control lines are helpful to double the keycodes of the keyboard that means if shift key is pressed then all the keys of the keyboard have different key codes generated. Similarly control/strobe pin also can be used. Note that control/strobe is working as the strobed line when keyboard is used in strobed input mode.

9. Keyboard debounce and control

This block transfers the code of the key depressions with proper debounce logic.

10. Display registers

This block contains two registers A and B, which store the display codes.

Each register stores the four bit code. For 8-bit code they can be combined. The contents of this block appear on the output pins, named as out A_0 to out A_3 and out B_0 to out B_3.

11. Control and Timing registers

This block contains the different types of control registers, which are helpful to program the 8279.

12. Timing and Control

This block controls the overall function of all the blocks along with the control timing registers block. A special pin called BD is associated with this block. This pin is useful for blinking the display under the program control.

The programmable peripheral interface 8279 is a 40-pin device.

These 40-pins are divided into four categories.

1. **CPU interface pins.**
2. **Keyboard input pins.**
3. **Display output pins.**
4. **Scan pins.**

1. CPU interface pins

It consists of IRQ, DB_O-DB_7, $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A_O, RESET and CLK.

IRQ: This is output line and known as interrupt request. This signal is activated whenever the FIFO RAM is not empty in keyboard mode or a change in a sensor is defected in sensor matrix mode. This line is used for interrupt driven $^I/_O$ system.

DB_O-DB_7: These lines are bi-directional lines. All data bytes and command bytes between microprocessor and 8279 are transferred through these lines.

$\overline{RD}$: This is an input line. When this pin becomes low the microprocessor can read a byte from 8279.

$\overline{WR}$: This is an input line. When this pin becomes low the microprocessor can write a byte into 8279.

$\overline{CS}$: This is an input line. When this pin becomes low the microprocessor can read or write a byte.

A_O: This is an input line. When A_0=0 means work with Data register. And when A_0=1 means work with control/status register.

RESET: This is an input line. A high signal on this pin resets the 8279. After being reset, the IC 8279 is forced to select following modes.

1. 16 digit, 8-bit character display-left entry.
2. Encoded scan keyboard-2 key lock out.
3. The clock pre-scaler or divisor is set to 31_{10}.
4. The clear code is set to 0000 0000. (Common Cathode Display).

CLK: This is an input line. The clock is given from microprocessor to generate internal timing.

2. Keyboard Input Lines

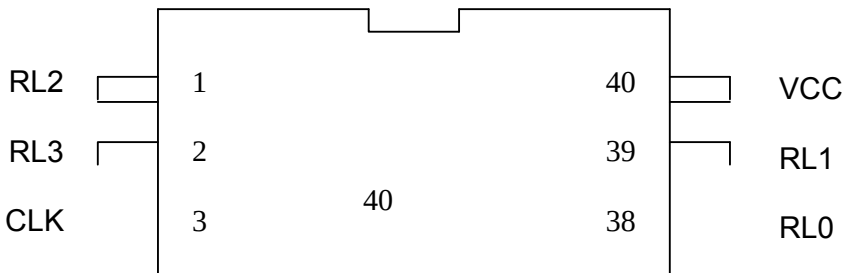
This keyboard Input line category consists RL_0-RL_7, shift and CNTL/STB lines. These lines are connected to the keyboard.

RL_0-RL_7: These are the return lines. These lines are connected to the scan lines through the keys or sensor switches. It is having an internal pull up registers so it remains high till the key remains unpressed. They also serve as an 8-bit input in the strobed input mode.

Shift: This is an input line. In the scanned keyboard modes, the status on this pin is also strobed with the key position on key closure. It has an active internal pull-up. Hence key closure should pull it low.

CNTL/STB: For keyboard modes this line is used as a control input and strobed like status pin on key closure.

In the strobed input mode, this line acts as STB input line. The data on RL_0-RL_7 is entered into FIFO at the rising edge of this signal.



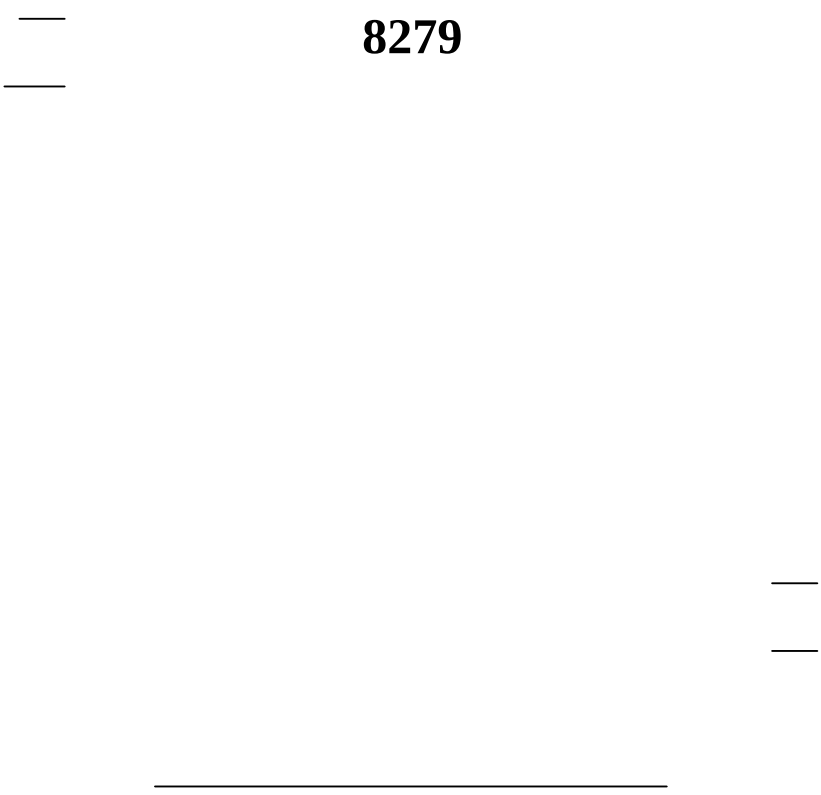


Figure-5.1.2(a) Pin diagram of 8279

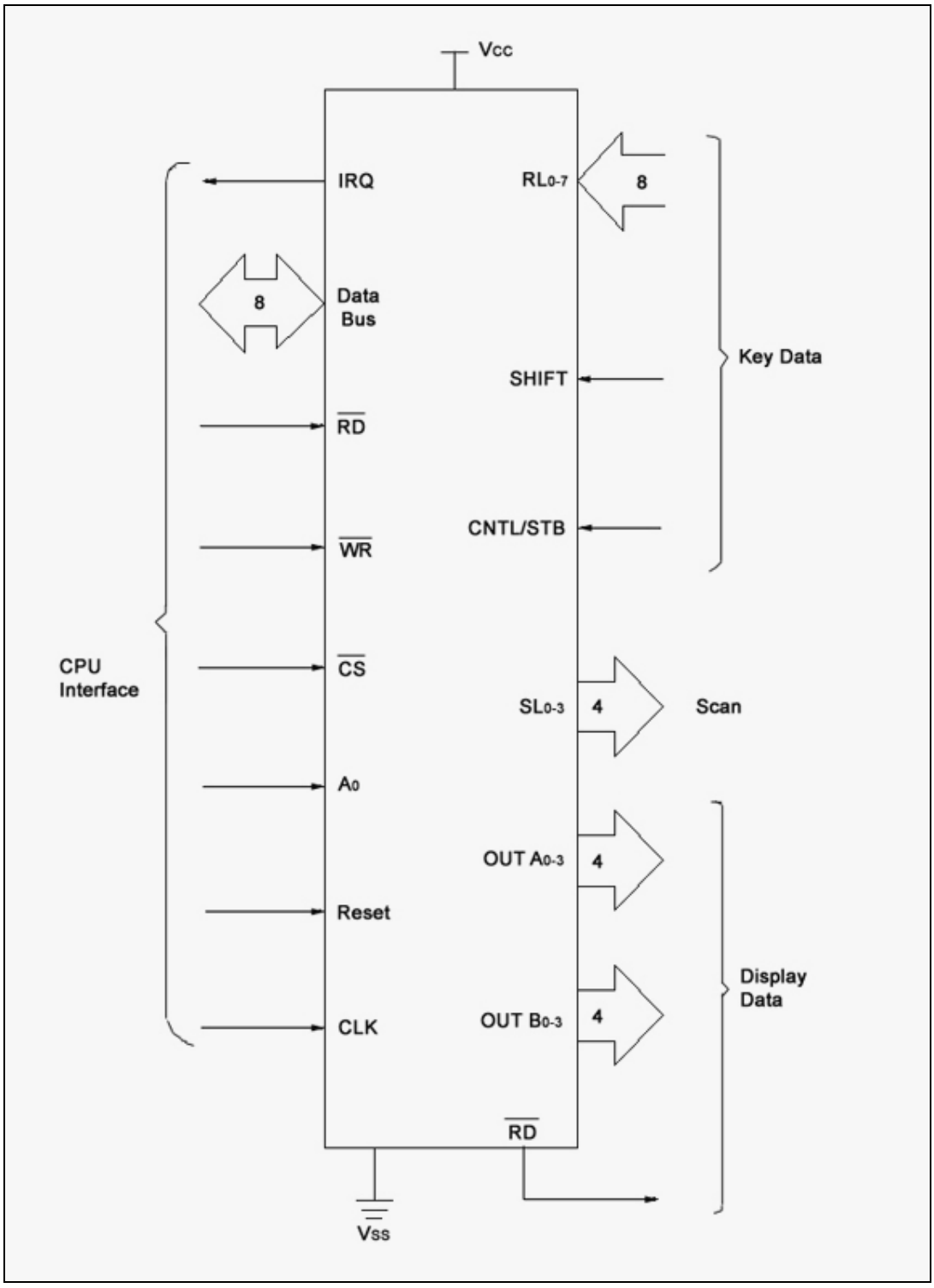


Figure-5.1.2(b) The logic symbol of 8279

3. Display output lines

This display output lines category consists of OUT A_{0-3}, OUT B_{0-3} and BD lines. These lines are connected to display.

OUT A_{0-3} & OUT B_{0-3}

These lines are output lines. These are two, four bit output ports of the 16-byte display RAM. The data from these outputs is synchronized to the scan lines (SL_0-SL_3) for multiplexed digit displays. These two ports can be individually blanked or inhibitated.

$\overline{BD}$: This is an output line. It is used to blank the display. This pin is activated during digit switching or by a display blanking command.

4. Scan Lines

SL_0-SL_3: These are output lines. It is used to scan keyboard or sensor matrix and display.

These lines can be either encoded (1 of 16) or decoded (1 of 4 SL_0 to SL_3).

5.1.3. Programming part of 8279

The programming of 8279 can be done using the following control word.

1. Key board/Display mode set

MSB							LSB
0	0	0	D	D	K	K	K

D D: Display Mode

0 0: Eight 8-bit character display – left entry
0 1: Sixteen 8-bit character display – left entry
1 0: Eight 8-bit character display – right entry
1 1: Sixteen 8-bit character display – right entry

K K K: Keyboard Mode

0 0 0: encoded scan keyboard – 2 key lockout
0 0 1: decoded scan keyboard – 2 key lockout
0 1 0: encoded scan keyboard – N-key rollover
0 1 1: decoded scan keyboard – N-key rollover
1 0 0: encoded scan sensor matrix
1 0 1: decoded scan sensor matrix
1 1 0: strobed input, encoded display scan

2. Program clock

MSB							LSB
0	0	1	P	P	P	P	P

This command causes the external clock to be divided by a prescaler PPPPP (which may have values in the range 2 to 31) to generate internal timing and multiplexing signals.

3. Read FIFO/Sensor RAM

MSB							LSB
0	1	0	AI	X	A	A	A

AI = Auto-Increment
AAA = RAM Address Bits
X = Don't care

This sets up the 8279 for a read of the FIFO/sensor RAM. All subsequent reads will be from successive locations in the FIFO (if the AI flag is set) until another command is issued. In the Sensor Matrix mode, AAA represents one of the eight rows of the sensor RAM. If AI = 1, subsequent Read operations are from successive FIFO locations.

4. Read display RAM

MSB							LSB
0	1	1	AI	A	A	A	A

AI = Auto-Increment
AAAA = Row Address of Display RAM that is to be read

5. Write display RAM

After writing this command word with A_0 = 1, further write operations with A_0 = 0 will be to the display RAM. The role of the AI flag is the same as in the Read Display RAM and FIFO/Sensor RAM.

MSB							LSB
1	0	0	AI	A	A	A	A

6. Display write inhibit/Blanking

MSB							LSB
1	0	1	X	IW	IW	BL	BL

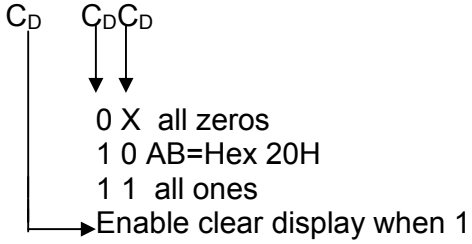
IW = Inhibit Write Flag
BL = Blank Display Flag

IW bits can be used to mask nibble A or nibble B when set to '1'. The BL bits, if set to '1' cause the A and B nibbles to be blanked.

7. Clear

MSB							LSB
1	1	0	C_D	C_D	C_D	C_F	C_A

C_D = Clears all rows of Display RAM to a selectable blanking code as follows:



C_F = FIFO status cleared, IRQ line reset
C_A = Clear All (combined C_D and C_F)

8. End Interrupt/ Error Mode Set

MSB							LSB
1	1	1	E	X	X	X	X

In the N-key rollover mode, if the E bit is programmed to '1', the chip operates in the special Error mode. In the Sensor Matrix mode, this command lowers the IRQ line and thus permits further writing into RAM.

9. Status word

MSB							LSB
DU	S/E	O	U	F	N	N	N

D7: DU : Display Unavailable
D6: S/E : Sensor closure/ Error flag for multiple closure.
D5: O : Over-run Error
D4: U : Under-run Error
D3: F : FIFO Full
D2: N :
D1: N : **Number of characters in FIFO**
D0: N :

The FIFO status word indicates whether there is an underrun (when the CPU attempts to read an empty FIFO) or overrun (which occurs during the attempted entry of an additional character when the FIFO is full), and the number of characters in the FIFO RAM. It also provides information about Sensor Closure, any error flag for multiple closures, and the status of the availability of the display.

DU = Display Unavailable
S/E = Sensor Closure\Error Flag for Multiple Closures
O = error Over-run
U = error Under-run
F = FIFO Full
NNN = Number of characters in FIFO

5.1.4. Interfacing of 8279 in the present system

The interfacing of 8279 in the present microprocessor trainer kit is explained in two steps.

The first step describes the interfacing of 8279 with 8085A, while the second step describes the interfacing of 8279 with 24-keys hex pad and 8-FND displays.

⇨ STEP – 1 – CPU side interfacing of 8279

Figure – 5.1.4.1(A) shows the circuit details of interfacing the 8279 with 8085A through 74LS373 and 74155. The functions of Latch IC 74LS373 and decoder 74155 are explained earlier. The Read, Write, Clock, Reset and IRQ pins of 8279 are directly connected with the $\overline{RD}$, $\overline{WR}$, CLK OUT, Reset out and RST5.5 respectively.

The A_0 pin of 8279 is connected to the AD_0 pin of 8085 through a Latch IC 74373. This suggests that for every T_1 clock of every machine cycle 74LS373 connects the address line A_0 with the A_0 pin of 8279.

Note that address lines A_{14} and A_{15} are responsible for generating chip select for 8279.

The following signal conditions generate the chip select for 8279.

Pin	A15	A14	IO/$\overline{M}$
Logic state	0	1	1

The port addresses of 8279 are decided as shown in Table 5.1.4.1.

Table – 5.1.4.1

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	Selection
A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	1	X	X	X	X	X	1	Control/Status register
0	1	X	X	X	X	X	0	Data register

In the hardware we have interfaced 8279 as I/O device by using IO/$\overline{M}$ such that CS of 8279 becomes valid only when IO/$\overline{M}$ = 1. For this reason we have to use I/O related instructions for accessing ports of 8279. We know that when we use I/O addressing the lower byte of address is duplicating on higher address lines. For this reason in Table 5.1.4.1 We have taken A_0-A_8, A_1-A_9……A_7-A_{15}, as common address lines. Taking this into consideration address line A_8, A_{14} and A_{15} decide the port addresses of 8279. Unconnected addressing lines can be taken as '0' or '1'. This generates many duplicate addresses of 8279 ports. But in our work we have taken unconnected lines as zeros. So port addresses of 8279 are as follows.

Control/Status register = 41H
Data register = 40H

⇨ Step – 2 I/O devices side interfacing of 8279

Figure – 5.1.4.1(B) illustrates the circuit details of the interfacing of the 24-keys hex keypad and 8-FND display with 8279. The 8279 is configured as follows.

1. For key board, encoded scan keyboard with 2-key lockout.
2. For display, 8-digit, 8-character display with right entry.

This configuration for key board and display makes the KB/DISP mode set word to be 10H.

The clock for the 8279 is programmed to be system clock divided by 31 hence 3MHz/31 = 96.77KHz is the clock for 8279. This makes the program clock word of 8279 to be 3Fh.

We have used here common anode type seven segment FND display so to make individual display element (diode) glow, the corresponding output line of 8279 has to be zero. Keeping this in mind to clear the all display FNDs (all eight) we have to output "all once", on OA_0 to OA_3 and OB_0 to OB_3 output lines of 8279. For this all rows of display RAM should be made 1's. At the same time we need not clear the FIFO. All these all together frames the "Clear word" to be DDh.

The output lines of 8279 are connected to the individual elements of all FNDs through 74LS245. To make a one way direction in 74LS245 we have made the DIR pin of this IC to be high. The enable gate pin G which is active low has been connected permanently to the ground.

Another important aspect of 8279 is scanning. For this four scan lines are provided, Viz. SL_0 to SL_3. But due to manufacturers guideline the SL_3 line is not used. SL_0 to SL_2 lines are connected to the decoder 74LS138, which is permanently enabled by connecting its active high enable pin G_1 to V_{CC} and active low pins G_{2A} and G_{2B} with the ground.

The outputs of SL_0 to SL_2 change from 000 to 111, and selects the outputs of 74LS138 from Y_0 to Y_7 in sequence. These outputs Y_0 to Y_7 are connected with common anodes of individual eight FNDs. Thus when SL_0 to SL_2 are 000 it makes display-0 to be selected. The Y_0 output of 74138 is also connected with Row-2 of the matrix key board. Where in keys 4, 5, 6, 7, REG-CHK & Sing SIP are "returned" to the return lines RL_1 to RL_6 of 8279. In similar way the outputs are connected with the remaining Rows of key board & corresponding FNDs of display. Y_4, Y_5, Y_6 & Y_7 are connected with only FNDs.

Figure – 5.1.4.1(A) The circuit details of interfacing the 8279 with 8085A

Figure – 5.1.4.1(B) The circuit details of the interfacing of the 24-keys hex keypad and 8-FND display with 8279

Circuit Function

The important aspects of this circuit function can be divided as follows.

1. How the circuit identifies the key code.
2. How the circuit displays a display character.

1. Key code identification

The key codes are identified by the arrangement of the key matrix 8279 can identify the key from maximum 8X8 key matrix. It has also facility of assigning four different meaning to individual keys by the use of shift and control keys. We have not used this facility.

We have designed 4X6 matrix key board. Here rows and columns are in decoded form in a sense that whenever any key is pressed its corresponding position number will be transferred to FIFO RAM in encoded form. For example, if in the present circuit “Execute” key is pressed then its corresponding position i.e. Row-2 and Column-4 will be encoded as 100 for return lines and 010 for scan lines as shown bellow.

CNT	SHIFT						
		←	SCAN	→	←	RETURN	→
1	1	0	1	0	1	0	0

Code = D4H

The complete key codes of key board are given in following Table-5.1.4.2..

Table – 5.1.4.2

Keys	Scan Line number	Return Line number	Codes in Binary & HEX
0	1	6	1100 1110=CE
1	1	5	1100 1101=CD
2	1	1	1100 1001=C9
3	1	2	1100 1010=CA
4	0	6	1100 0110=C6
5	0	5	1100 0101=C5
6	0	1	1100 0001=C1
7	0	2	1100 0010=C2
8	2	6	1101 0110=D6
9	2	5	1101 0101=D5
A	2	1	1101 0001=D1
B	2	2	1101 0010=D2
C	5	6	1110 1110=EE
E	5	1	1110 1101=ED
F	5	2	1110 1001=E9
FM	5	3	1110 1010=EA
NEXT	1	4	1110 1011=EB
PREV	1	3	1100 1011=CB
GO	5	4	1110 1100=EC
EXEC	2	4	1101 0100=D4

BM	2	3	1101 0011=D3
ER	0	3	1100 0011=C3
SS	0	4	1100 0100=C4

3. Display of character

Which ever character we want to display on FND display its equivalent FND code has to be generated. Following figure explains the procedure.

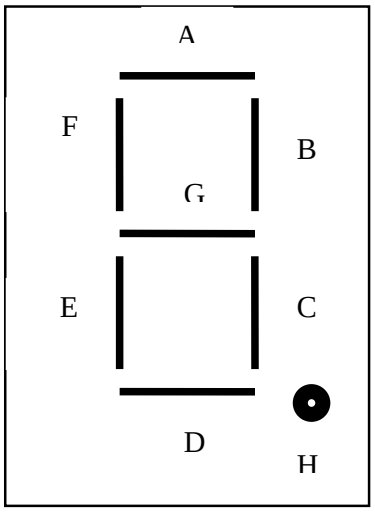


Figure. 5.1.4.2. (a) The diode elements of FND

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
A	B	C	D	E`	F	G	H

Figure. 5.1.4.2. (b) code generation

As shown in Figure. 5.1.4.2.(a). the diode elements of FND LT542 are corresponded with data bits D_7 to D_0 as shown in Figure. 5.1.4.2.(b).

Suppose we want to display '1' we have to make the diode element b and c low and others high, because this is common anode type FND. This makes the code.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
1	0	0	1	1	1	1	1

The code is 9FH. By following this procedure the necessary numerals and alphabets are generated as shown below.

The below list given in Table-5.1.4.3 containts only the character codes which we have used.

Table – 5.1.4.3

Character	Code	Character	Code
0	03	F	71
1	9f	D	85
2	25	0	03
3	0d	E	60
4	99	S	49
5	49	U	82
6	41	8	01
7	1f	5	49
8	01	H	91
9	09	L	E3
A	11	P	31
B	C1	N	D5
C	63	T	E1
D	85	G	08
E	61	P	30

All these codes are stored in EPROM. Now to display them they are brought into the display RAM of 8279. Suppose we want to display the start up message "—doe. su. 85". Now the following instructions will bring this code into display RAM of 8279.

```
START:    LXI SP,5EFFH

          MVI A,10H      ;8-DIGIT,8-CHARACTER RIGHT ENTRY
          OUT 41H        ;ENCODED SCAN K.B. 2-KEY LOCKOUT

          MVI A,3FH      ;CLK DIVIDED BY 31(11111b).
          OUT 41H
          MVI A,DDH      ;COMMON ANODE CF=0 CA=1
          OUT 41H

            MVI A,00H
            LXI H,TEMP_BUF
            MVI B,25H
LOOP1:      MOV M,A
            INX H
            DCR B
            JNZ LOOP1

            CALL BLK_DISP   ;BLANK ALL DISPLAY MEMORY
            LXI D,FFFFH
            CALL DELAY
```

```
        MVI A,0EH
        SIM

        MVI A,40H       ;READ FIFO IN FIXED ADDRESS MODE
        OUT 41H
        MVI A,90H       ;TO WRITE DISPLAY RAM
        OUT 41H
        MVI B,08H
        LXI H,FND_CODE+19H

LOOP21: MOV A,M
        OUT 40H
        DCX H
        DCR B
        JNZ LOOP21


START1: EI
        HLT
```

When codes of the start up message are transferred the scan counter will generate the binary count starting from 000 to 111 and will display the codes stored in display RAM.

5.2 Programmable peripheral Interface 8255

8255 is a general purpose $^{I}/_{O}$ device, to perform parallel communication. It has three $^{I}/_{O}$ ports named Port-A, Port-B and Port-C. Each consisting of eight bits. These ports can be used in three different modes that is mode-0, mode-1 and mode-2.

In mode-0 all three ports are used as simple input or output ports. In mide-1, port-A and/or port-B are used as input/output under the control of the hand shaking signals which are obtained through certain pins of port-C. In mode-2, port-A can be used as bi-directional port while port-B can be used in mode-0 or in mode-1. Certain pins of port-C are used by port-A as handshaking signals.[6]

5.2.1. Block diagram of 8255

The block diagram and pin diagram of 8255 are shown in Figure 5.2.1.1 and Figure 5.2.1.2 respectively. The block diagram of 8255 can be divided into following blocks.

1. Data bus buffer.
2. Read/Write control logic.
3. Group-A and Group-B control.
4. Ports A, B and C.

1. Data bus buffer

The pins D_0 to D_7 are associated with this block. It is a three state bi-directional 8-bit buffer. It is used to interface 8255 to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instruction of the CPU. Control Word and the Status information are also transferred through the data bus buffer.

2. Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

$\overline{(CS)}$

Chip Select. A "low" on this input pin enables the communication between the 8255A and the CPU.

$\overline{(RD)}$

Read. A "low" on this input pin enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255A.

$\overline{(WR)}$

Write. A "low" on this input pin enables the CPU to write data or control words into the 8255A.

(A_0 and A_1)

Port Select 0 and Port Select 1. These input signals, in conjunction with the $\overline{RD}$ and $\overline{WR}$ inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A_0 and A_1).

Table-5.2.1.1 - 8255A BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Input Operation (READ)
0	0	0	1	0	Port A – Data Bus
0	1	0	1	0	Port B – Data Bus
1	0	0	1	0	Port C – Data Bus
					Output Operation (WRITE)
0	0	1	0	0	Data Bus – Port A
0	1	1	0	0	Data Bus – Port B
1	0	1	0	0	Data Bus – Port C
1	1	1	0	0	Data Bus – Control
					Dissable Function
X	X	X	X	1	Data Bus – 3-State
1	1	0	1	0	Illegal Condition
X	X	1	1	0	Data Bus – 3-State

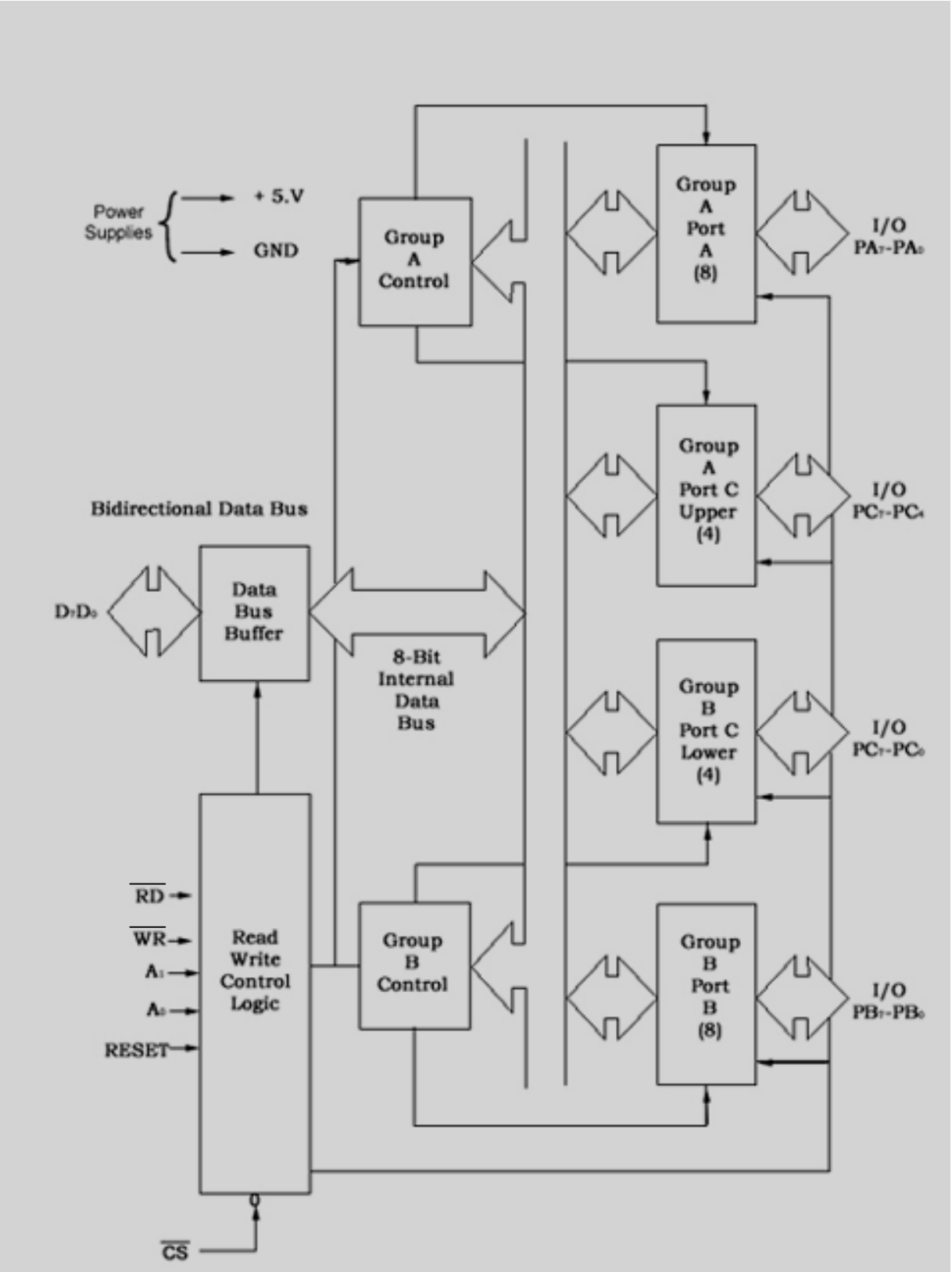


Figure 5.2.1.1. Block diagram of 8255.

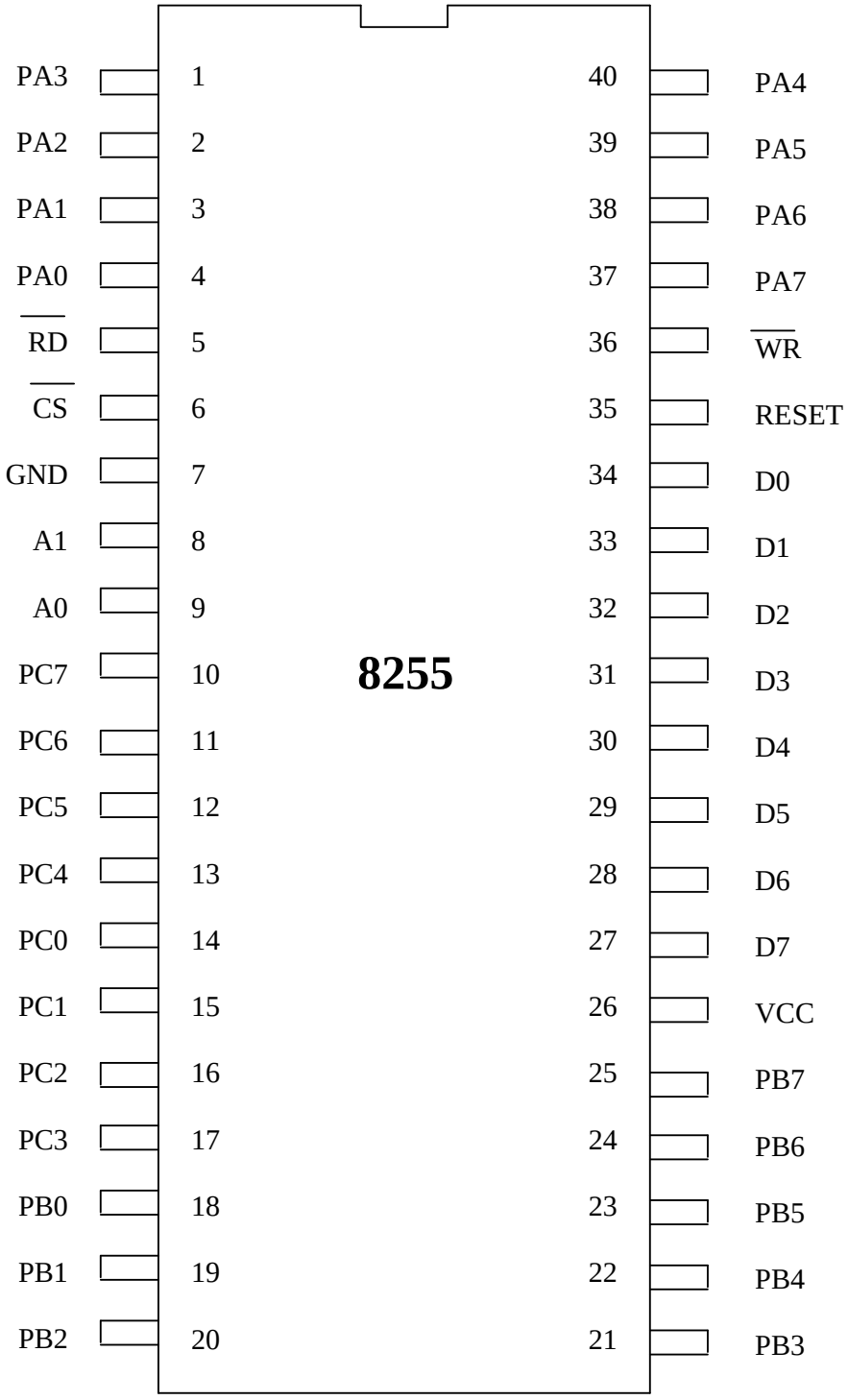


Figure 5.2.1.2. Pin diagram of 8255.

(RESET)

A “high” on this input clears the control register and all ports (A, B, C) are set to the input mode.

4. Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU “outputs” a control word to the 8255A. The control word contains information such as “mode”, “bit set”, “bit reset”, etc., that initializes the functional configuration of the 8255A.
Each of the Control blocks (Group A and Group B) accepts “commands” from the Read/Write Control Logic, receives “control words” from the internal data bus and issues the proper commands to its associated ports.

Control Group A-Port A and Port C upper (C7-C4)

Control Group B-Port B and Port C lower (C2-C0)

The Control Word Register can only be written into. No Read operation of the Control Word Register is allowed.

5. Ports A, B, and C

The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or “personality” to further enhance the power and flexibility of the 8255A.

Port A: One 8-bit data output latch/buffer and one 8-bit data input latch.

Port B: One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C: One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal input in conjunction with ports A and B.

5.2.2 MODE – Description

MODE-0 (Basic Input output Mode)

In this mode all three ports are used either as input or output ports. In this mode outputs are latched but inputs are not latched. 16-different $^{I}/_{O}$ configurations are possible in this mode.

MODE-1 (Strobed Input/ output)

This mode provides a means for transferring $^{I}/_{O}$ data to or from a specified port in conjunction with strobes or hand shaking signals. Port-A and Port-B use the lines of Port-C, to generate or accept handshaking signals. In this mode inputs and outputs are latched.

Mode-1 describes Ports-A & B in two ways input and output.

- **Ports A and B as input ports in Mode-1**

Figure 5.2.2.1. shows the Port-A and Port-B configuration in Mode-1. In this input mode following signals are necessary to understand.

$\overline{\text{STB}}$: (Strobed input): A “low” on this input loads data into the input latch.

IBF: (Input Buffer Full F/F): A “high” on this output indicates that data has been loaded into the input latch; in essence, an acknowledgement IBF is set by $\overline{\text{STB}}$ input being low and is reset by the rising edge of the $\overline{\text{RD}}$ input.

INTR: (Interrupt Request): A high on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{\text{STB}}$ is a “one”, IBF is a “one” and INTE is a “one”. It is reset by the falling edge of $\overline{\text{RD}}$. this procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A: Controlled by bit set/reset of PC_4.

INTE B: Controlled by bit set/reset of PC_2.

Figure 5.2.2.1. (b) explains the timing relation of above described hand shaking signals.

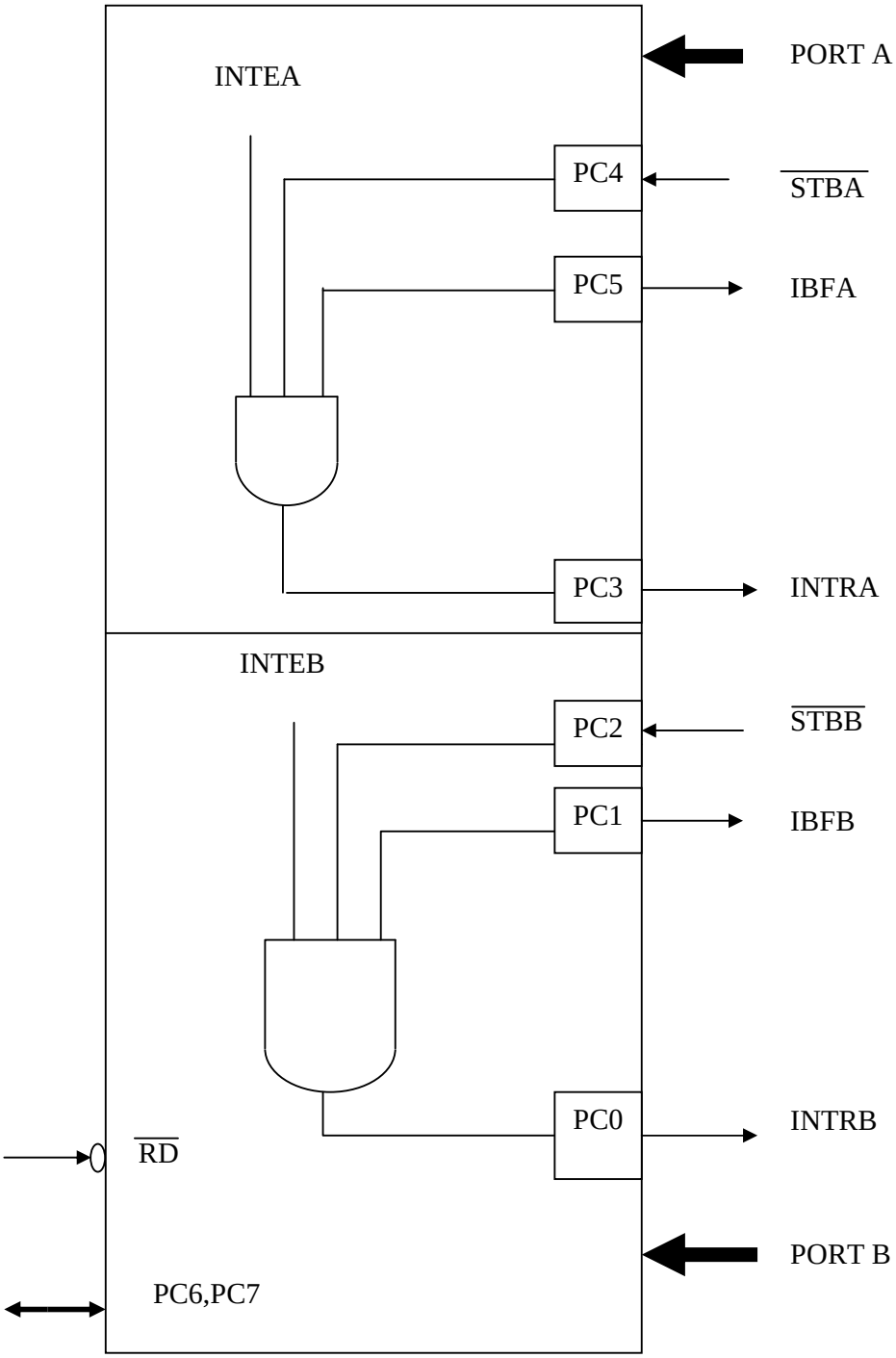


Figure – 5.2.2.1(a)-Mode-1 INPUT

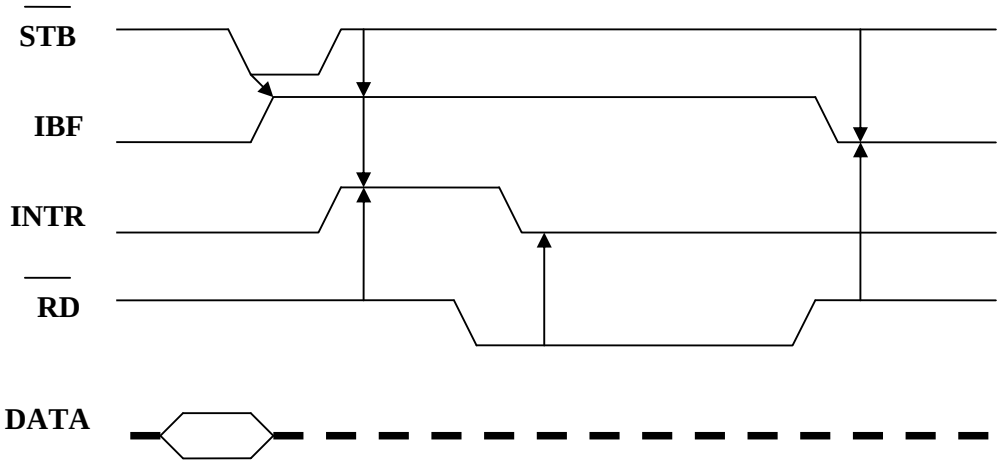


Figure – 5.2.2.1(b)-Mode-1 input waveform

- **Port-A & B as output ports in Mode-1**

Figure 5.2.2.2. (a) describes the port-A and B configured as output ports in mode-1, while figure 5.2.2.2.(b) Shows the Waveforms for the same. The hand shaking signals in the output mode are as follows.

$\overline{\text{OBF}}$ (Output Buffer full F/F): The $\overline{\text{OBF}}$ output will go "low" to indicate that the CPU has written data out to the specified port. The $\overline{\text{OBF}}$ F/F will be set by the rising edge of the $\overline{\text{WR}}$ input and reset by $\overline{\text{ACK}}$ input being low.

$\overline{\text{ACK}}$ (Acknowledge Input): A "low" on this input informs the 8255A that the data from port A or port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt request): A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when $\overline{\text{ACK}}$ is a "one", $\overline{\text{OBF}}$ is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{\text{WR}}$.

INTE A: Controlled by bit set/reset of PC_6.

INTE B: Controlled by bit set/reset of PC_2.

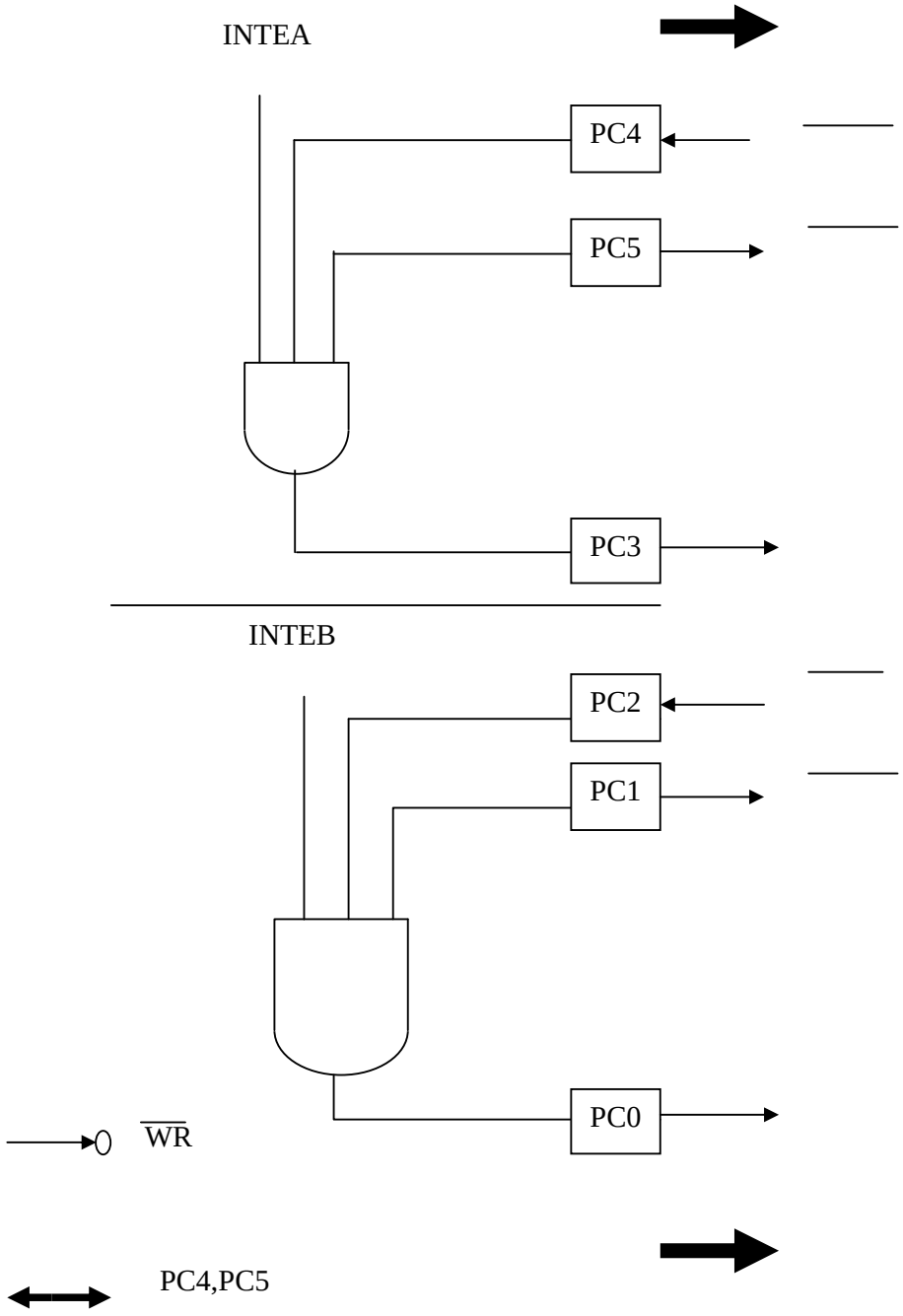


Figure – 5.2.2.2(a)-Mode-1 OUTPUT

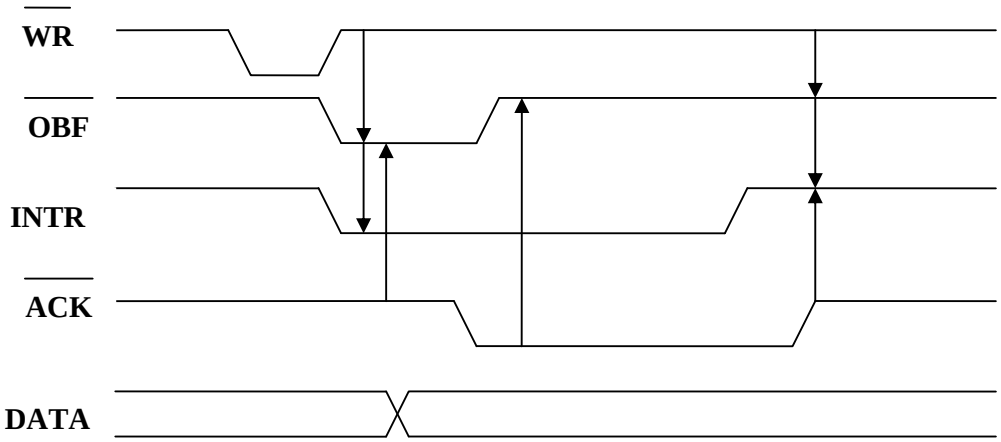


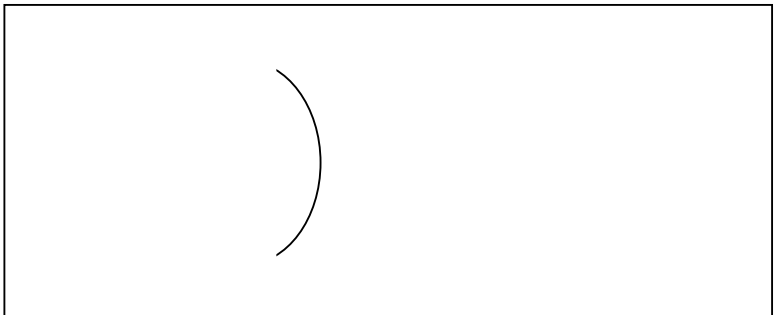
Figure 5.2.2.2.(b) Waveform

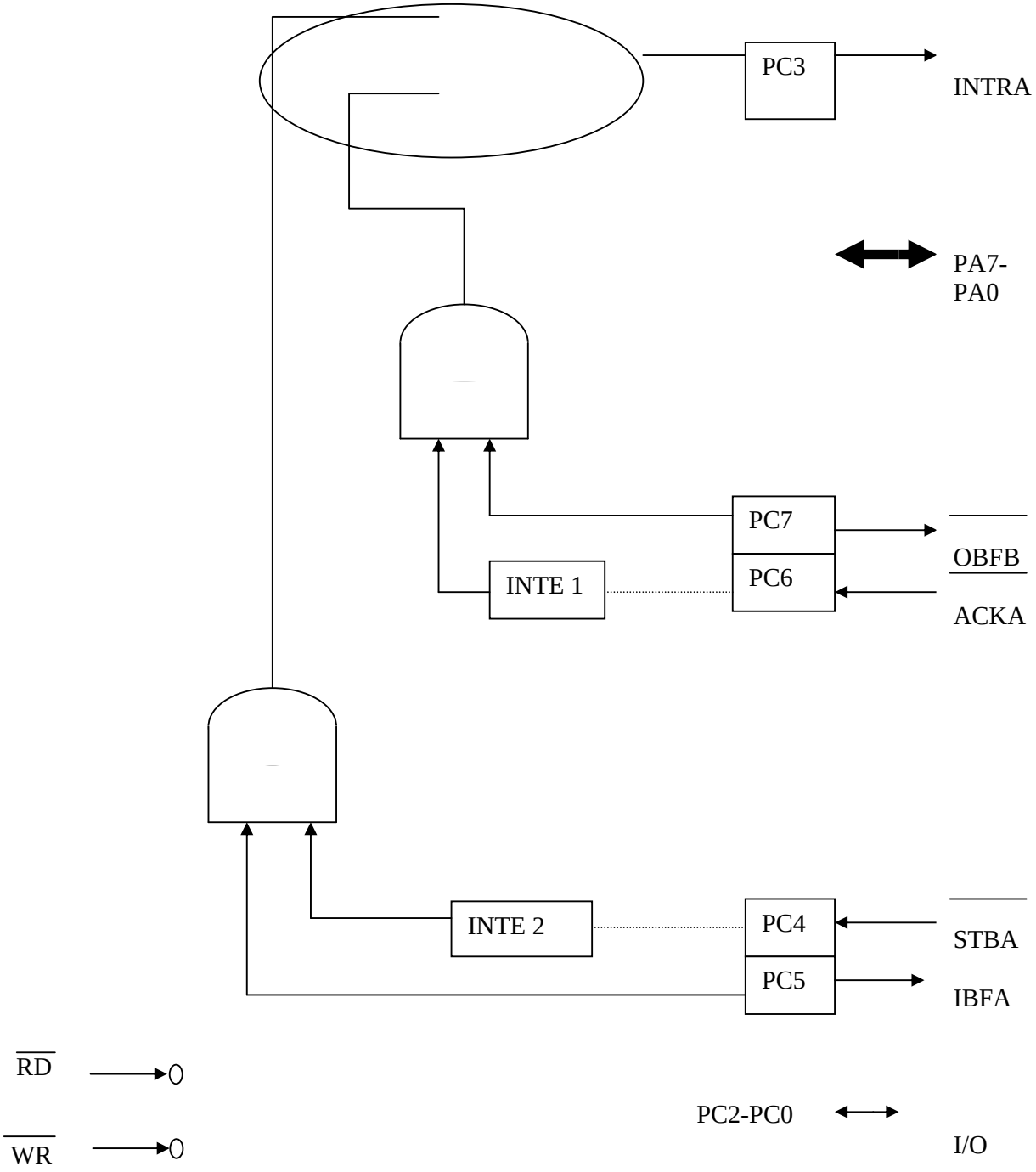
MODE-2 (Strobed bidirection bus $^{I}/_{O}$)

This mode provides means for transmitting and receiving data using hand shaking signals for Port-A only.

Both inputs and outputs are latched. The Port-A configuration for mode-2 is shown in Figure. 5.2.2.3 (A) and its waveforms in Figure 5.2.2.3(B)..

The hand shaking signals $\overline{OBF}$, $\overline{ACK}$, $\overline{STB}$, IBF have their usual meanings. INTE-1 is controlled by PC6, while INTE-2 is controlled by PC4. Note that in mode-2. Port-B can be programmed independently.





Fligure-5.2.2.3(A) mode-2 configuration

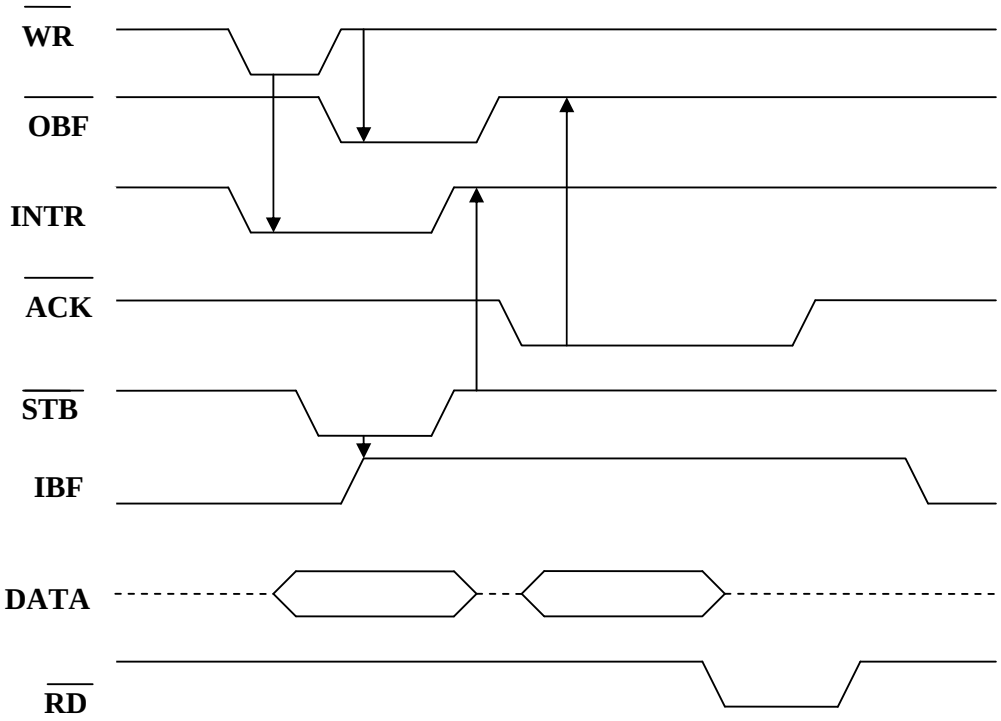


Figure-5.2.2.3(B) mode-2 waveforms

5.2.3. Programming of 8255

The basic control words which are used to program 8255 are mode definition format and bit Set/Reset format.

Mode definition format

The Figure 5.2.3.1 shows the bit description of mode definition format.

CONTROL WORD

D7	D6	D5	D4	D3	D2	D1	D0
1	**M1**	**M0**	**PA**	**PCU**	**M**	**PB**	**PCL**

D0: PCL PORT C (LOWER)
1=INPUT
0=OUTPUT
D1: PB PORT B
1=INPUT
0=OUTPUT
D2: M MODE SELECTION FOR GROUP-B
0=MODE-0
1=MODE-1
D3: PCU PORT C (UPPER)
1=INPUT
0=OUTPUT

D4:PA PORT A
1=INPUT
0=OUTPUT
D5,D6: M0 AND M1 MODE FOR GROUP-A

M1	M0	MODE SELECTION
0	0	MODE-0
0	1	MODE-1
1	0	MODE-2
1	1	MODE-2

D7: THIS BIT SHOULD BE '1'.

Figure 5.2.3.1. – Control word.

BSR CONTROL WORD

While Figure 5.2.3.2 shows the bit description for BSR format. Note that BSR Format is applicable to Port-C bits only.

D7	D6	D5	D4	D3	D2	D1	D0
0	**X**	**X**	**X**	**B2**	**B1**	**B0**	**S/R**

D0: S/R SER/RESET
1=SET
0=RESET
D1,D2,D3:B0,B1,B2 TO ACTIVATE LINE OF PORT-C

B2	B1	B0	PORT-C LINE
0	0	0	LINE-0
0	0	1	LINE-1
0	1	0	LINE-2
0	1	1	LINE-3
1	0	0	LINE-4
1	0	1	LINE-5
1	1	0	LINE-6
1	1	1	LINE-7

D4,D5,D6: THIS BITS ARE DON'T CARE BITS.

D7: THIS BIT SHOULD BE '0' TO SELECT BSR MODE.

Figure 5.2.3.2 – BSR Control word.

Status Word

The status words are available for Mode-1 & Mode-2.

Figure 5.2.3.3. describes the bit definitions for Mode-1 status word. While Figure 5.2.3.4. explains the bit definitions for mode-2 status word.

D7	D6	D5	D4	D3	D2	D1	D0
I/O	**I/O**	**IBFA**	**INTEA**	**INTRA**	**INTEB**	**IBFB**	**INTRB**

Figure 5.2.3.3 (a) $^{I}/_{P}$ Mode-1 Status Word.

D7	D6	D5	D4	D3	D2	D1	D0
$\overline{\text{OBFA}}$	**INTEA**	**I/O**	**I/O**	**INTRA**	**INTEB**	**$\overline{\text{OBFB}}$**	**INTRB**

Figure 5.2.3.3 (b) O/P Mode-1 Status Word.

D7	D6	D5	D4	D3	D2	D1	D0
$\overline{\text{OBFA}}$	**INTE1**	**IBFA**	**INTE2**	**INTRA**	**X**	**X**	**X**

Figure 5.2.3.4 Status word of Mode-2

5.2.4 Interfacing of 8255 in the present system

The interfacing of 8255 is shown in Figure 5.2.4.1. The CPU side pins of 8255 are connected with corresponding pins of 8255 through Latch and decoder ICs, as the case may be on the other hand the port pins of 8255 are connected directly with a 40-pin connector.

The D_0 to D_7 lines of 8255 are directly connected with AD_0 to AD_7 lines of 8085 respectively. The $\overline{RD}$ and $\overline{WR}$ of 8255 are directly connected with $\overline{RD}$ and $\overline{WR}$ pins of 8085. A_0 and A_1 of 8255 are connected with A_0 and A_1 of 8085 using Latch IC 74LS373. A Reset pin is directly connected with Reset out. CS is connected with 1Y0 of decoder 74155. It is the same decoder which we have discussed earlier.

This suggests that to select 8255 ports and control word following becomes port addresses.

Table – 5.2.4.1

Address	Port
00H	Port – A
01H	Port – B
02H	Port – C
03H	CWR

Which port pin is connected with which pin of 40-pin FRC connector is quite clear from Figure 5.2.4.1.

If the user wants to use 8255 he has to design his circuit as per the details of connector.

Figure-5.2.4.1 Interfacing of 8255 in present system

Connector details are given in Table-5.2.4.2.

Table-5.2.4.2

PB2	PB3
PB1	PB4
PB0	PB5
PC3	PB6
PC2	PB7
PC1	
PC0	
PC4	
PC5	
PC6	
PC7	
GND	GND
	VCC
PA0	PA4
PA1	PA5
PA2	PA6
PA3	PA7

5.2.5 Verification of some concepts of 8255

To test and verify the various modes of the 8255 we have developed the small circuits. [14] For this we have used the in built facility of an 8085 based microprocessor kit. We have made Port-A as output port and port B as input port and configured 8255 in Mode-1. Then we have established the interconnections of control signals using port-c and inverter chip. Similarly, port-A was configured as bi-directional port and port-c was used for generating proper control signals. Then for both the cases proper softwares were prepared and tested successfully.

CHAPTER – 6 P.C.B. DESIGNING AND FABRICATION

6.1 Schematic preparation

There are number of softwares available for schematic preparation. The well known names are ORCAD, Smart Work, Protel, Easy PCB, ES – Route, Cadence, etc.

The first step to design the PCB is the schematic preparation.

The software is having various in – built library functions. Various components are available in the library. All TTL Devices are available in TTL library, all CMOS Devices are available in CMOS library. All memory chips are available in memory library.

The list of components can be seen while opening the library. We can easily place and remove any component, connection, power symbol, Ground symbol bus, junction, entry, text, etc. These facilities may be given by different names in different softwares.

Important commands used are given below.

Component:

This command is used for placement of component on the current work sheet. During placement the component may be rotated or mirrored. The component library must be included in the library list.

We can add or remove library from the library list. One can also use component browse for searching component.

Wire:

This command is used to place electrical wire connections on the current schematic worksheet. This wire can have sub – commands like begin, end, new, etc.

Power symbol & Ground symbol:

This command places power port on the schematic worksheet. This symbol shows that the connection has been established with power (V_{CC}) or ground. The symbols are as shown bellow.

Power symbol Ground symbol

Orientation can be changed by either 0 or 90 or 180 or 270 degree.

Bus:

This command is used to place a graphical bus line on the current worksheet. A bus line is used to represent a common path way for multiple signals on a worksheet.

Buses can be attached to ports or sheet symbols, for connection to other schematic sheets. This bus has a sub – commands like begin, end, new, etc.

Entry or Bus – entry:

This command is used to place a graphical bus entry object on the current worksheet. A bus entry is used to connect wires to a bus line.

Junction:

In schematic circuit diagram, when two wires are crossing each other, two situations are possible:.

1 – Both wires are contacting each other.

2 – Both wires are not contacting each other.

If user wants to join both wires, the junction should be placed as shown in figure.

No contact contact

The shape of junction is generally square or circular.

Text:

The user can write text at any place in the schematic worksheet.

Block:

The rectangular block can be defined to copy, delete, move, etc.

It is to be noticed that the preparation of schematic circuit is generally done in smallest sized sheet. If the circuit is large enough so the user can go for large sheet or the user can put a connector between two sheets to connect both circuits. Schematic can be saved and printed. The detailed schematic diagram of 8085 based educational microprocessor kit is shown in Figure – 6.1.1.

Figure 6.1.1 Schematic circuit diagram of 8085 trainer kit

6.2 P.C.B. Layout Preparation

Preparing layout means to design the layout of track, components etc. on the P.C.B.

After preparing schematic worksheet the P.C.B. layout can be prepared either manually or automatically. In the present work all P.C.B.s are manually prepared.

The track size, via size, pad size, layer, color for track, etc. can be changed by different commands. The scale and grid can be adjusted in mils or mm.

The double sided P.T.H.-P.C.B. is designed in present work. The footprints of some components are stored in computer, while some footprints were designed. The footprint of any component can be designed and stored into computer in addition to the available footprints.

Three layers bottom layer, top layer and overplay are used. The copper tracks, pads and vias are placed on bottom layer and top layer. The actual size of components have been drawn in overlay for proper spacing.

The Figures-6.2.1, 6.2.2 and 6.2.3 given below show the bottom layer, top layer & overlay.

Figure-6.2.1 Bottom layer of 8085 based trainer kit

Figure- 6.2.2 Top layer of 8085 based trainer kit

Figure – 6.2.3 Overlay layer of 8085 based trainer kit

6.3 PCB fabrication

The process to develop single sided PCB is as follows.[7]

To fabricate the PCB, we need to follow the given steps given below. We need copper clad laminate and negative of the PCB layout.

Step – 1: Clean the copper clad laminate.

The copper clad laminate should be washed by water. Then clean it by glass – Wool. After cleaning process the copper plate should be shining.

Step – 2: Make a thin layer of Photoresist + Thinner on a clean copper clad laminate & dry it.

Mix photo-resist and thinner and pour it on the clean and shiny copper plate. By using cotton make a thin layer of this mixer.

Note: Photo resist is light sensitive liquid so do this process in dark room.

Now dry the sheet in dark room for 15 minutes.

Step – 3: Put negative & expose under ultraviolet light.

Now put negative on the copper sheet and expose it under ultraviolet light. If ultraviolet light source is not available, we can work with ordinary tube light or sunlight. This ultraviolet light will do polymerization of the photo resist under light.

The light cannot pass through opaque region of negative, so no polymerization takes place in this region.

Step – 4: Developer Wash.

Now give developer wash. In developer the un-polymerized photoresist will dissolve.

The polymerized pattern will be same as the PCB layout.

Step – 5: Etching.

Then etch this PCB in liquid $FeCl_3$. After etching we can see the designed layout on the copper sheet.

Step – 6: Drilling.

Now the last step is drilling, means make holes, to insert components.

6.4 General aspect.

6.4.1 List of components.

The list of components in 8085 base educational kit is as given bellow.

- IC 8085.
- IC 8279.
- IC 8255.
- IC 74155.
- IC 74245.
- IC 74373.
- IC 74138.
- IC 2764A.
- IC 6264.
- Crystal 6 MHz.
- 8 – FNDS.
- 8 – BC 177 PNP transistors.
- Push button – switches.
- Few resistors & capacitors.
- ZIF Socket for 2764.
- IC Sockets for all ICs.

6.4.2 costing

The cost factor is one of the most important factor in electronics instruments. The cost of this educational kit is approximately Rs.2500/- (Excluding power supply).

This cost can vary as per demand and supply of different components in electronics market.

CHAPTER-7 : MONITOR PROGRAM

```
RESET:        .EQUAL  1700H
INT55:        .EQUAL  1670H
INT1:         .EQUAL  1600H
TEMP_BUF:     .EQUAL  5F00H   ;8 DISPLAYS 5F00H TO 5F07H
DISP_COU:     .EQUAL  5F08H
DATA_MEM:     .EQUAL  5F09H
BYTE_3:       .EQUAL  0050H
BYTE_2:       .EQUAL  0080H
FND_CODE:     .EQUAL  00A0H
BYTE_ONE:     .EQUAL  00E0H
FM_RAM:       .EQUAL  5F0AH    ;FILL MEMORY
SS_RAM:       .EQUAL  5F0BH    ;SINGLE STEP
ER_RAM:       .EQUAL  5F0CH    ;EXAM REGISTER
GO_RAM:       .EQUAL  5F0DH    ;GO TO ADDRESS
EXEC_RAM:     .EQUAL  5F0EH    ;EXECUTE THE PROGRAM
BM_RAM:       .EQUAL  5F0FH    ;BLOCK MOVE KEY
NEXT_RAM:     .EQUAL  5F10H    ;NEXT KEY
PREV_RAM:     .EQUAL  5F11H    ;PREVIOUS KEY
TEMP1:        .EQUAL  5F12H
DISPDATA:     .EQUAL  5F13H;8  DISPLAY SO 5F13H TO
                              ;5F1AH
DA_OR_AD:     EQUAL   5F1BH;INTITIALLY 00H BUT AFTER
                              ;NEXT KEY 01H
REG_PC:       .EQUAL  5F1CH;PC IS 16-BIT SO 5F1CH &
                              ;5F1DH  ARE USED
```

```
BM_CONT:      .EQUAL  5F1EH;BM IS INCREMENTED 4 TIMES

BM_S_ADD:     .EQUAL  5F1FH;BLOCK MOVE STARTING
                              ;ADDRESS(2 BYTES)

BM_E_ADD:     .EQUAL  5F21H;BLOCK MOVE ENDING
                              ;ADDRESS(2 BYTES)

BM_D_ADD:     .EQUAL  5F23H;BLOCK MOVE DESTINATION
                              ;ADDRESS(2 BYTES)

ER_COUNT:     .EQUAL  5F25H   ;EXAM REGISTER COUNT

REGPSW:       .EQUAL  5F26H   ;(2 BYTES FLAG & ACC)

REGBC:        .EQUAL  5F28H   ;(2 BYTES C & B)

REGDE:        .EQUAL  5F2AH   ;(2 BYTES E & D)

REGHL:        .EQUAL  5F2CH   ;(2 BYTES L & H)

REGSP:        .EQUAL  5F2EH   ;(2 BYTES SPH & SPL)

REGPC:        .EQUAL  5F30H   ;(2 BYTES PCH & PCL)

REGTEMP:      .EQUAL  5F32H   ;(2 BYTES MSB & LSB)

SS_EXE:       .EQUAL  5F34H   ;(6 BYTES 34H TO 39H)

CHK_CARY:     .EQUAL  5F35H   ;TO CHECK CARRY FLAG

D0:           .EQUAL  CEH
D1:           .EQUAL  CDH
D2:           .EQUAL  C9H
D3:           .EQUAL  CAH
D4:           .EQUAL  C6H
D5:           .EQUAL  C5H
D6:           .EQUAL  C1H
D7:           .EQUAL  C2H
D8:           .EQUAL  D6H
D9:           .EQUAL  D5H
DA:           .EQUAL  D1H
DB:           .EQUAL  D2H
DC:           .EQUAL  EEH
DD:           .EQUAL  EDH
DE:           .EQUAL  E9H
DF:           .EQUAL  EAH
 FM:          .EQUAL  EBH
NEXT:         .EQUAL  CCH
PREV:         .EQUAL  CBH
```

```
 GO:          .EQUAL  ECH
EXEC:         .EQUAL  D4H
 BM:          .EQUAL  D3H
 ER:          .EQUAL  C3H
 SS:          .EQUAL  C4H


    ORG 0000H
      JMP START

    ORG 0008H      ;RST 1 LOCATION

     JMP RST_1

    ORG 002ch

     jmp int5_5

     ORG BYTE_3

     DB 0C3H  ;JMP    1
     DB 0C2H  ;JNZ    2
     DB 0CAH  ;JZ     3
     DB 0D2H  ;JNC    4
     DB 0DAH  ;JC     5
     DB 0E2H  ;JPO    6
     DB 0EAH  ;JPE    7
     DB 0F2H  ;JP     8
     DB 0FAH  ;JM     9
     DB 0CDH  ;CALL   A
     DB 0C4H  ;CNZ    B
     DB 0CCH  ;CZ     C
     DB 0D4H  ;CNC    D
     DB 0DCH  ;CC     E
     DB 0E4H  ;CPO    F
     DB 0ECH  ;CPE    10
     DB 0F4H  ;CP     11
     DB 0FCH  ;CM     12
     DB 01H   ;LXI B  13
     DB 11H   ;LXI D  14
     DB 21H   ;LXI H  15
     DB 31H   ;LXI SP 16
     DB 2AH   ;LHLD   17
     DB 22H   ;SHLD   18
     DB 3AH   ;LDA    19
     DB 32H   ;STA    1A
```

ORG BYTE_2

```
DB 3EH  ;MVI A        1
DB 06H  ;MVI B        2
DB 0EH  ;MVI C        3
DB 16H  ;MVI D        4
DB 1EH  ;MVI E        5
DB 26H  ;MVI H        6
DB 2EH  ;MVI L        7
DB 36H  ;MVI M        8
DB 0C6H ;ADI          9
DB 0CEH ;ACI          A
DB 0D6H ;SUI          B
DB 0DEH ;SBI          C
DB 0E6H ;ANI          D
DB 0EEH ;XRI          E
DB 0F6H ;ORI          F
DB 0FEH ;CPI          10
DB 0DBH ;IN           11
DB 0D3H ;OUT          12
```

ORG FND_CODE

```
db 03h ;0
db 9fh ;1
db 25h ;2
db 0dh ;3
db 99h ;4
db 49h ;5
db 41h ;6
db 1fh ;7
db 01h ;8
db 09h ;9
db 11h ;A
db c1h ;B
db 63h ;C
db 85h ;D
db 61h ;E
db 71h ;F
db FEH ;.      10
db FFH ;BLANK  11
db fdh ;-      12
db 85h ;d      13
db 03h ;O      14
db 60h ;E.     15
db 49h ;S      16
db 82h ;U.     17
db 01h ;8      18
db 49h ;5      19
```

```
                db 91h ;H       1A
                db e3h ;L       1B
                db 31h ;P       1C
                db d5h ;n       1D
                db e1h ;t       1E
                db 08H ;g.      1F
                db 30H ;P.      20

                ORG BYTE_ONE

                db c9h  ;RET  1
                db c0h  ;RNZ  2
                db c8h  ;RZ   3
                db d0h  ;RNC  4
                db d8h  ;RC   5
                db e0h  ;RPO  6
                db e8h  ;RPE  7
                db f0h  ;RP   8
                db f8h  ;RM   9
                db e9h  ;PCHL A


                ORG INT1

RST_1:          SHLD REGTEMP    ;SAVE HL IN TEMP MEMORY
                POP H           ;GET ADDRESS FROM WHERE RST 1 COMES
                PUSH H          ;AGAIN DECREMENT STACK POINTER
                SHLD REGPC      ;STORE ADDRESS OF PC
                LHLD REGTEMP    ;GET BACK ORIGINAL DATA OF HL

                CALL REG_SAVE   ;IT WILL SAVE ALL REGISTERS
                                        ;EXCEPT PC.
                JMP START
REG_SAVE:  SHLD REGHL

                PUSH D
                POP H
                SHLD REGDE

                PUSH B
                POP H
                SHLD REGBC

                PUSH PSW
                POP H
                SHLD REGPSW

                LXI H,0000H
                DAD SP
                INX H
```

```
                INX H
                INX H
                INX H
                SHLD REGSP

                RET
REGSTORE:  SHLD REGHL

                PUSH D
                POP H
                SHLD REGDE

                PUSH B
                POP H
                SHLD REGBC

                PUSH PSW
                POP H
                SHLD REGPSW

                RET

REG_LOAD:   LHLD REGPSW
                PUSH H
                POP PSW
                LHLD REGBC
                PUSH H
                POP B
                LHLD REGDE
                PUSH H
                POP D

                LHLD REGHL

                RET


                ORG INT55

int5_5:         PUSH PSW
                IN 40H
                STA DATA_MEM
                POP PSW
                RET
```

```
                ORG RESET

START:          LXI SP,5FFFH

                MVI A,10H       ;8-DIGIT,8-CHARACTER RIGHT ENTRY
                OUT 41H         ;ENCODED SCAN K.B. 2-KEY LOCKOUT

                MVI A,3FH       ;CLK DIVIDED BY 31(11111b).
                OUT 41H

                MVI A,DDH       ;COMMAN ANODE CF=0 CA=1
                OUT 41H

                MVI A,00H
                LXI H,TEMP_BUF
                MVI B,25H

LOOP1:          MOV M,A
                INX H
                DCR B
                JNZ LOOP1


                CALL BLK_DISP   ;BLANK ALL DISPLAY MEMORY
                LXI D,FFFFH
                CALL DELAY
                MVI A,0EH
                SIM


                MVI A,40H       ;READ FIFO IN FIXED ADDRESS MODE
                OUT 41H

                MVI A,90H       ;TO WRITE DISPLAY RAM
                OUT 41H

                MVI B,08H
                LXI H,FND_CODE+19H

LOOP21:         MOV A,M
                OUT 40H
                DCX H
                DCR B
                JNZ LOOP21

START1:         EI
                HLT
                LDA DATA_MEM
                CALL SU1
                JMP START1
```

```
BLK_DISP:   LXI H,DISPDATA   ;TO BLANK ALL DISPLAY MEMORY
            MVI B,08H
            MVI A,11H

LOOP2:      MOV M,A
            INX H
            DCR B
            JNZ LOOP2
            RET

SU1:        CPI FM           ;IF FILL MEMORY KEY IS PRESSED
            JNZ NO_0      ;JNZ NO_0
            CALL F_M_SR      ;FILL MEMORY SUBROUTINE
            RET

NO_0:       CPI NEXT      ;IF NEXT KEY IS PRESSED
            JNZ NO_1
            CALL NEXT_SUB
            RET

NO_1:       CPI PREV       ;IF PREVIOUS KEY IS PRESSED
            JNZ NO_2
            CALL PREV_SUB
            RET

NO_2:       CPI GO         ;IF GO KEY IS PRESSED
            JNZ NO_3
            CALL GO_SUB
            RET

NO_3:       CPI EXEC
            JNZ NO_4
            CALL EXEC_SUB   ;IF EXECUTE KEY IS PRESSED
            RET

NO_4:       CPI BM         ;IF BLOCK MOVE KEY IS PRESSED
            JNZ NO_5
            CALL BM_SUB
            RET

NO_5:       CPI ER         ;IF EXAM REGISTER KEY IS PRESSED
            JNZ NO_6
            CALL ER_SUB
            CALL DATADISP   ;TO DISPLAY DATA
            MVI A,00H      ;TO STORE 00H IN SS_RAM WHEN
                                ;REGISTERS
            STA SS_RAM      ;ARE EXAMINED.
            RET
```

```
NO_6:       CPI SS
            JNZ NO_7
            CALL SS_SUB
            RET

NO_7:       CALL CHK_0TOF
            RET

SS_SUB:     LDA SS_RAM
            CPI SS
            JZ OUT_SS1      ;IF IT CALLS SECOND TIME SO GO OUT
                            ;SS1
            MVI A,SS
            STA SS_RAM      ;NOW STORE CODE OF SING. STEP INTO
                            ;SS_RAM
            CALL DISP_SS    ;TO DISPLAY SING. STEP.
            RET

OUT_SS1:    LXI H,DISPDATA+5        ;TO TEST N OF SING. STEP.
            MOV A,M
            CPI 1DH         ;N IS STORED AT 1DH IN FND_CODE
            JZ OUT_SS2      ;IF ADDRESS IS NOT GIVEN SO GET
                            ;CALL GET_ADD
            CALL GET_ADD
            SHLD REGPC      ;PREVIOUS ADDRESS

OUT_SS2:    LHLD REGPC      ;GET ADDRESS OF SING.STEP.
            MOV A,M         ;GET THE DATA TO BE EXECUTED INTO
                            ;ACC.
            LXI H,SS_EXE    ;RAM LOCATION WHERE ONE BYTE IS
                            ;STORED
            MOV M,A         ;TO EXECUTE

            LXI H,BYTE_2    ;TO CHECK 2-BYTE INSTRUCTION
            MVI B,12H       ;12 2-BYTE INSTRUCTIONS ARE IN 8085

SS_LOOP1:   CMP M
            JZ O_SS_2       ;OUT SING.STEP. 2-BYTE
            INX H
            DCR B
            JNZ SS_LOOP1

            LXI H,BYTE_3
            MVI B,1AH

SS_LOOP2:   CMP M
            JZ O_SS_3       ;OUT SING.STEP. 3-BYTE
            INX H
            DCR B
            JNZ SS_LOOP2
```

```
                JMP BYTE_1

O_SS_2:     LHLD REGPC
            INX H          ;DATA OF 2-BYTE INSTRUCTION
            SHLD REGPC     ;STORE AFTER INCREMENT
            MOV A,M        ;GET DATA INTO ACC.
            LXI H,SS_EXE+1  ;POINTER TO SS_EXE+1
            MOV M,A
            INX H
            JMP RUN_IT

BYTE_1:     LXI H,BYTE_ONE
            MVI B,0AH

SS_LOOP3:   CMP M
            JZ O_SS_1      ;OUT SING.STEP. 1-BYTE
            INX H
            DCR B
            JNZ SS_LOOP3

            LXI H,SS_EXE    ;IF 1-BYTE INSTRUCTION SO THEN JUMP
            INX H          ;THEN JUMP BACK AT COMEBACK

RUN_IT:     MVI M,C3H
            LXI D,COMEBACK  ;ADDRESS OF COMEBACK
            INX H
            MOV M,E
            INX H
            MOV M,D
            CALL REG_LOAD
            SHLD REGHL
            LXI H,0000H
            DAD SP
            SHLD REGTEMP

            LHLD REGPSW  ;CARRY IS MODIFIED BY
                         ;INSTRUTION(DAD SP)
            PUSH H
            POP PSW

            LHLD REGSP
            SPHL
            LHLD REGHL
            JMP SS_EXE

COMEBACK:   SHLD REGHL
            LXI H,CHK_CARY
            JNC NO_CARY
            MVI M,01H
            JMP OUT_CARY
```

```
NO_CARY:    MVI M,00H
OUT_CARY:   LXI H,0000H
            DAD SP
            SHLD REGSP
            LHLD REGTEMP
            SPHL
            LHLD REGHL
            CALL REGSTORE
            LXI H,CHK_CARY
            MOV A,M
            CPI 01H
            JNZ OUT10
            LHLD REGPSW
            MOV A,L
            ORI 01H
            MOV L,A
            SHLD REGPSW
OUT10:      LHLD REGPC
            INX H
            SHLD REGPC;BACK FROM BYTE-3 SUBROUTINE
BF_BYTE3:   LHLD REGPC
            CALL GET_DATA   ;THE DATA OF ADDRESS IN HL IN
                                    ;D(MSB)
            LXI H,DISPDATA  ;AND E(LSB)
            MOV M,E
            INX H
            MOV M,D
            INX H
            MVI M,12H       ;CODE TO DISPLAY '-'.
            INX H
            MVI M,12H       ;CODE TO DISPLAY '-'
            LXI D,DISPDATA+7
            LHLD REGPC
            MOV A,H
            ANI 0F0H
            RRC
            RRC
            RRC
            RRC
            STAX D
            DCX D
            MOV A,H
            ANI 0FH
            STAX D
            DCX D
            MOV A,L
            ANI 0F0H
            RRC
            RRC
            RRC
```

```
                RRC
                STAX D
                DCX D
                MOV A,L
                ANI 0FH
                STAX D

                CALL DATADISP


                RET

O_SS_1:         MOV A,M
                CPI C9H         ;RET
                JNZ SS_1_O1     ;SING.STEP.1-BYTE OUT-1
RET_YES:        LHLD REGSP      ;SP INCREMENTED BY 2 & NEW ADD.IS
                                ;IN HL
                MOV A,M
                STA REGPC
                INX H
                MOV A,M
                STA REGPC+1
                INX H
                SHLD REGSP
                JMP BF_BYTE3
SS_1_O1:        CPI C0H         ;RNZ
                JNZ SS_1_O2     ;SING.STEP.1-BYTE OUT-2
                LHLD REGPSW
                MOV A,L
                ANI 40H         ;TO CHECK ZERO FLAG
                JZ RET_YES
RET_NO:         LHLD REGPC
                INX H
                SHLD REGPC
                JMP BF_BYTE3    ;BACK FROM BYTE-3


SS_1_O2:        CPI C8H         ;RZ
                JNZ SS_1_O3     ;SING.STEP.1-BYTE OUT-3
                LHLD REGPSW
                MOV A,L
                ANI 40H         ;TO CHECK ZERO FLAG
                JNZ RET_YES
                JMP RET_NO


SS_1_O3:        CPI D0H         ;RNC
                JNZ SS_1_O4     ;SING.STEP.1-BYTE OUT-4
                LHLD REGPSW
```

```
                MOV A,L
                ANI 01H         ;TO CHECK CARRY FLAG
                JZ RET_YES
                JMP RET_NO


SS_1_O4:        CPI D8H         ;RC
                JNZ SS_1_O5     ;SING.STEP.1-BYTE OUT-5
                LHLD REGPSW
                MOV A,L
                ANI 01H         ;TO CHECK CARRY FLAG
                JNZ RET_YES
                JMP RET_NO


SS_1_O5:        CPI E0H         ;RPO
                JNZ SS_1_O6     ;SING.STEP.1-BYTE OUT-6
                LHLD REGPSW
                MOV A,L
                ANI 04H         ;TO CHECK CARRY FLAG
                JZ RET_YES
                JMP RET_NO


SS_1_O6:        CPI E8H         ;RPE
                JNZ SS_1_O7     ;SING.STEP.1-BYTE OUT-7
                LHLD REGPSW
                MOV A,L
                ANI 04H         ;TO CHECK CARRY FLAG
                JNZ RET_YES
                JMP RET_NO


SS_1_O7:        CPI F0H         ;RP
                JNZ SS_1_O8     ;SING.STEP.1-BYTE OUT-8
                LHLD REGPSW
                MOV A,L
                ANI 80H         ;TO CHECK CARRY FLAG
                JZ RET_YES
                JMP RET_NO


SS_1_O8:        CPI F8H         ;RM
                JNZ SS_1_O9     ;SING.STEP.1-BYTE OUT-9
                LHLD REGPSW
```

```
                MOV A,L
                ANI 80H         ;TO CHECK CARRY FLAG
                JNZ RET_YES
                JMP RET_NO


SS_1_O9:        SHLD REGPC
                JMP BF_BYTE3

O_SS_3:         MOV A,M
                CPI C3H         ;JMP
                JNZ SS_3_O1     ;SING.STEP. 3-BYTE OUT-1
JUMP_YES:       LHLD REGPC
                INX H
                MOV A,M
                STA REGPC
                INX H
                MOV A,M
                STA REGPC+1
                JMP BF_BYTE3    ;BACK FROM BYTE-3

SS_3_O1:        CPI C2H         ;JNZ
                JNZ SS_3_O2     ;SING.STEP.3-BYTE OUT-2
                LHLD REGPSW
                MOV A,L
                ANI 40H         ;TO CHECK ZERO FLAG
                JZ JUMP_YES
JUMP_NO:        LHLD REGPC
                INX H
                INX H
                INX H
                SHLD REGPC
                JMP BF_BYTE3    ;BACK FROM BYTE-3
SS_3_O2:        CPI CAH         ;JZ
                JNZ SS_3_O3     ;SING.STEP.3-BYTE OUT-2
                LHLD REGPSW
                MOV A,L
                ANI 40H         ;TO CHECK ZERO FLAG
                JNZ JUMP_YES
                JMP JUMP_NO

SS_3_O3:        CPI D2H         ;JNC
                JNZ SS_3_O4     ;SING.STEP.3-BYTE OUT-2
                LHLD REGPSW
                MOV A,L
                ANI 01H         ;TO CHECK CARRY FLAG
                JZ JUMP_YES
                JMP JUMP_NO
```

```
SS_3_O4:    CPI DAH        ;JC
            JNZ SS_3_O5    ;SING.STEP.3-BYTE OUT-2
            LHLD REGPSW
            MOV A,L
            ANI 01H        ;TO CHECK CARRY FLAG
            JNZ JUMP_YES
            JMP JUMP_NO

SS_3_O5:    CPI E2H        ;JPO(PARITY FLAG IS RESET)
            JNZ SS_3_O6    ;SING.STEP.3-BYTE OUT-2
            LHLD REGPSW
            MOV A,L
            ANI 04H        ;TO CHECK CARRY FLAG
            JZ JUMP_YES
            JMP JUMP_NO

SS_3_O6:    CPI EAH        ;JPE(PARITY FLAG IS SET)
            JNZ SS_3_O7    ;SING.STEP.3-BYTE OUT-2
            LHLD REGPSW
            MOV A,L
            ANI 04H        ;TO CHECK CARRY FLAG
            JNZ JUMP_YES
            JMP JUMP_NO

SS_3_O7:    CPI F2H        ;JP(SIGN FLAG IS RESET)
            JNZ SS_3_O8    ;SING.STEP.3-BYTE OUT-2
            LHLD REGPSW
            MOV A,L
            ANI 80H        ;TO CHECK CARRY FLAG
            JZ JUMP_YES
            JMP JUMP_NO

SS_3_O8:    CPI FAH        ;JM(SIGN FLAG IS SET)
            JNZ SS_3_O9    ;SING.STEP.3-BYTE OUT-2
            LHLD REGPSW
            MOV A,L
            ANI 80H        ;TO CHECK CARRY FLAG
            JNZ JUMP_YES
            JMP JUMP_NO

SS_3_O9:    CPI CDH        ;CALL
            JNZ SS_3_OA    ;SING.STEP.3-BYTE OUT-2
CALL_YES:   LHLD REGPC
            INX H
            MOV A,M
            STA REGPC
            INX H
            MOV A,M
            STA REGPC+1
```

```
                INX H
                XCHG
                LHLD REGSP
                DCX H
                MOV M,D
                DCX H
                MOV M,E
                SHLD REGSP
                JMP BF_BYTE3    ;BACK FROM BYTE-3

SS_3_OA:        CPI C4H         ;CNZ
                JNZ SS_3_OB     ;SING.STEP.3-BYTE OUT-2
                LHLD REGPSW
                MOV A,L
                ANI 40H         ;TO CHECK ZERO FLAG
                JZ CALL_YES
CALL_NO:        LHLD REGPC
                INX H
                INX H
                INX H
                SHLD REGPC
                JMP BF_BYTE3    ;BACK FROM BYTE-3

SS_3_OB:        CPI CCH         ;CZ
                JNZ SS_3_OC     ;SING.STEP.3-BYTE OUT-2
                LHLD REGPSW
                MOV A,L
                ANI 40H         ;TO CHECK ZERO FLAG
                JNZ CALL_YES
                JMP CALL_NO

SS_3_OC:        CPI D4H         ;CNC
                JNZ SS_3_OD     ;SING.STEP.3-BYTE OUT-2
                LHLD REGPSW
                MOV A,L
                ANI 01H         ;TO CHECK CARRY FLAG
                JZ CALL_YES
                JMP CALL_NO

SS_3_OD:        CPI DCH         ;CC
                JNZ SS_3_OE     ;SING.STEP.3-BYTE OUT-2
                LHLD REGPSW
                MOV A,L
                ANI 01H         ;TO CHECK CARRY FLAG
                JNZ CALL_YES
                JMP CALL_NO
```

```
SS_3_OE:    CPI E4H       ;CPO
            JNZ SS_3_OF   ;SING.STEP.3-BYTE OUT-2
            LHLD REGPSW
            MOV A,L
            ANI 04H       ;TO CHECK CARRY FLAG
            JZ CALL_YES
            JMP CALL_NO

SS_3_OF:    CPI ECH       ;CPE
            JNZ SS_3_OG   ;SING.STEP.3-BYTE OUT-2
            LHLD REGPSW
            MOV A,L
            ANI 04H       ;TO CHECK CARRY FLAG
            JNZ CALL_YES
            JMP CALL_NO

SS_3_OG:    CPI F4H       ;CP
            JNZ SS_3_OH   ;SING.STEP.3-BYTE OUT-2
            LHLD REGPSW
            MOV A,L
            ANI 80H       ;TO CHECK CARRY FLAG
            JZ CALL_YES
            JMP CALL_NO

SS_3_OH:    CPI FCH       ;CM
            JNZ SS_3_OI   ;SING.STEP.3-BYTE OUT-2
            LHLD REGPSW
            MOV A,L
            ANI 80H       ;TO CHECK CARRY FLAG
            JNZ CALL_YES
            JMP CALL_NO

SS_3_OI:    CPI 01H       ;LXI B,
            JNZ SS_3_OJ   ;SING.STEP.3-BYTE OUT-2
            LHLD REGPC
            INX H
            MOV A,M
            STA REGBC
            INX H
            MOV A,M
            STA REGBC+1
            INX H
            SHLD REGPC
            JMP BF_BYTE3  ;BACK FROM BYTE-3
```

```
SS_3_OJ:    CPI 11H        ;LXI D,
            JNZ SS_3_OK    ;SING.STEP.3-BYTE OUT-2
            LHLD REGPC
            INX H
            MOV A,M
            STA REGDE
            INX H
            MOV A,M
            STA REGDE+1
            INX H
            SHLD REGPC
            JMP BF_BYTE3   ;BACK FROM BYTE-3

SS_3_OK:    CPI 21H        ;LXI H,
            JNZ SS_3_OL    ;SING.STEP.3-BYTE OUT-2
            LHLD REGPC
            INX H
            MOV A,M
            STA REGHL
            INX H
            MOV A,M
            STA REGHL+1
            INX H
            SHLD REGPC
            JMP BF_BYTE3   ;BACK FROM BYTE-3

SS_3_OL:    CPI 31H        ;LXI SP,
            JNZ SS_3_OM    ;SING.STEP.3-BYTE OUT-2
            LHLD REGPC
            INX H
            MOV A,M
            STA REGSP
            INX H
            MOV A,M
            STA REGSP+1
            INX H
            SHLD REGPC
            JMP BF_BYTE3   ;BACK FROM BYTE-3

SS_3_OM:    LXI D,SS_EXE   ;LHLD,SHLD,STA,LDA
            LHLD REGPC
            MOV A,M
            STAX D
            INX H
            INX D
            MOV A,M
            STAX D
            INX H
            INX D
            MOV A,M
```

```
                STAX D
                SHLD REGPC
                LXI H,SS_EXE+3
                JMP RUN_IT

DISP_SS:        LXI H,DISPDATA+7
                MVI M,05H       ;TO DISPLAY S
                DCX H
                MVI M,01H       ;TO DISPLAY I
                DCX H
                MVI M,1DH       ;TO DISPLAY n
                DCX H
                MVI M,1FH       ;TO DISPLAY g.
                DCX H
                MVI M,05H       ;TO DISPLAY S
                DCX H
                MVI M,1EH       ;TO DISPLAY t
                DCX H
                MVI M,0EH       ;TO DISPLAY E
                DCX H
                MVI M,20H       ;TO DISPLAY P.
                DCX H
                CALL DATADISP
                RET



ER_SUB:         STA ER_RAM      ;EXAM REGISTER SUBROUTINE
                LDA ER_COUNT    ;HOW MANY TIMES THIS KEY IS
                                        ;PRESSED.
                CPI 00H         ;IS IT PRESSED FIRST TIME.
                JNZ ER_OUTBC    ;EXAM REGISTER OUTPUT B & C
                CALL A_AND_F    ;DISPLAY CONTENT OF ACC & FLAGS

INC_ERC:        LDA ER_COUNT
                INR A
                STA ER_COUNT
                RET

ER_OUTBC:       CPI 01H
                JNZ ER_OUTDE    ;EXAM REGISTER OUT TO DISPLAY D &
                                        ;E
                CALL B_AND_C    ;DISPLAY CONTENT OF B & C
                JMP INC_ERC     ;INCREMENT EXAM REGISTER COUNT

ER_OUTDE:       CPI 02H
                JNZ ER_OUTHL    ;EXAM REGISTER OUT TO DISPLAY H &
                                        ;L
                CALL D_AND_E    ;DISPLAY CONTENT OF D AND E
```

```
                JMP INC_ERC     ;INCREMENT EXAM REGISTER COUNT

ER_OUTHL:       CPI 03H
                JNZ ER_OUTSP    ;EXAM REGISTER OUT TO DISPLAY
                                ;SP(STACK POINTER)
                CALL H_AND_L    ;DISPLAY CONTENT OF H AND L
                JMP INC_ERC     ;INCREMENT EXAM REGISTER COUNT


ER_OUTSP:       CPI 04H
                JNZ ER_OUTPC    ;EXAM REGISTER OUT TO DISPLAY
                                ;PC(PROGRAM COUNTER)
                CALL SP_DISP    ;DISPLAY CONTENT OF STACK POINTER
                JMP INC_ERC     ;INCREMENT EXAM REGISTER COUNT


ER_OUTPC:       CALL PC_DISP    ;DISPLAY CONTENT OF PROGRAM
                                ;COUNTER
                XRA A           ;TO CLEAR ACC.
                STA ER_COUNT    ;RESET COUNT TO ZERO
                RET

A_AND_F:        LHLD REGPSW     ;LOAD HL BY PSW
                CALL HL_DISP    ;HL INTO DISPDATA+3 TO DISPDATA
                MVI A,11H       ;CODE FOR BLANK DISPLAY
                STAX D          ;AFTER CALLING HL_DISP
                INX D
                MVI A,0FH
                STAX D
                INX D
                MVI A,11H
                STAX D
                INX D
                MVI A,0AH
                STAX D
                RET


B_AND_C:        LHLD REGBC      ;LOAD HL BY BC
                CALL HL_DISP    ;HL INTO DISPDATA+3 TO DISPDATA
                MVI A,11H       ;CODE FOR BLANK DISPLAY
                STAX D          ;AFTER CALLING HL_DISP
                INX D
                MVI A,0CH
                STAX D
                INX D
                MVI A,11H
                STAX D
                INX D
                MVI A,0BH
```

```
                STAX D
                RET

D_AND_E:        LHLD REGDE      ;LOAD HL BY DE
                CALL HL_DISP    ;HL INTO DISPDATA+3 TO DISPDATA
                MVI A,11H       ;CODE FOR BLANK DISPLAY
                STAX D          ;AFTER CALLING HL_DISP
                INX D
                MVI A,0EH
                STAX D
                INX D
                MVI A,11H
                STAX D
                INX D
                MVI A,0DH
                STAX D
                RET

H_AND_L:        LHLD REGHL      ;LOAD HL BY HL
                CALL HL_DISP    ;HL INTO DISPDATA+3 TO DISPDATA
                MVI A,11H       ;CODE FOR BLANK DISPLAY
                STAX D          ;AFTER CALLING HL_DISP
                INX D
                MVI A,1BH       ;1B IS CODE TO DISPLAY L
                STAX D
                INX D
                MVI A,11H
                STAX D
                INX D
                MVI A,1AH       ;1A IS CODE TO DISPLAY H
                STAX D
                RET

SP_DISP:        LHLD REGSP      ;LOAD HL BY SP
                CALL HL_DISP    ;HL INTO DISPDATA+3 TO DISPDATA
                MVI A,11H       ;CODE FOR BLANK DISPLAY
                STAX D          ;AFTER CALLING HL_DISP
                INX D
                MVI A,11H
                STAX D
                INX D
                MVI A,1CH       ;1C IS CODE TO DISPLAY P
                STAX D
                INX D
                MVI A,05H       ;TO DISPLAY S
                STAX D
                RET

PC_DISP:        LHLD REGPC      ;LOAD HL BY PC
                CALL HL_DISP    ;HL INTO DISPDATA+3 TO DISPDATA
```

```
        MVI A,11H       ;CODE FOR BLANK DISPLAY
        STAX D          ;AFTER CALLING HL_DISP
        INX D
        MVI A,11H
        STAX D
        INX D
        MVI A,0CH
        STAX D
        INX D
        MVI A,1CH       ;1C IS CODE TO DISPLAY P
        STAX D
        RET



HL_DISP:    LXI D,DISPDATA  ;FILL UP LAST FOUR LOCATIONS
        MOV A,L         ;AND POINTED TO DISPDATA+4
        ANI 0FH
        STAX D
        INX D
        MOV A,L
        ANI F0H
        RRC
        RRC
        RRC
        RRC
        STAX D
        INX D
        MOV A,H
        ANI 0FH
        STAX D
        INX D
        MOV A,H
        ANI F0H
        RRC
        RRC
        RRC
        RRC
        STAX D
        INX D
        RET



BM_SUB:     STA BM_RAM
```

```
WRO_ADD:    LDA BM_CONT
            CPI 00H       ;FIRST TIME KEY IS PRESSED.(SADD)
            JNZ BM_OUT1
            CALL STAT_ADD
            JMP INC_RET    ;JUMP TO INCREMENT & RETURN

BM_OUT1:    CPI 01H       ;SECOND TIME KEY IS PRESSED(EADD)
            JNZ BM_OUT2
            CALL GET_ADD
            SHLD BM_S_ADD  ;BLOCK MOVE START ADDRESS
            CALL END_ADD
            JMP INC_RET

BM_OUT2:    CPI 02H
            JNZ BM_OUT3
            CALL GET_ADD
            SHLD BM_E_ADD  ;BLOCK MOVE END ADDRESS
            CALL DIST_ADD  ;THIRED TIME KEY IS PRESSED(DADD)
INC_RET:    LDA BM_CONT
            INR A
            STA BM_CONT
            CALL DATADISP
            RET

BM_OUT3:    CALL GET_ADD
            SHLD BM_D_ADD  ;BLOCK MOVE DESTINATION
                                ;ADDRESS
            MVI A,00H
            STA BM_CONT

            LHLD BM_S_ADD  ;GET STARTING ADDRESS
            XCHG           ;PUT STARTING ADDRESS INTO DE PAIR
            LHLD BM_E_ADD  ;GET ENDING ADDRESS

            MOV A,L
            SUB E
            MOV C,A
            MOV A,H
            SBB D
            JC WRO_ADD     ;WRONG ADDRESS IS GIVEN SO GET
                                ;AGAIN
            MOV B,A
            LHLD BM_D_ADD  ;GET DESTINATION ADDRESS INTO HL
            XCHG           ;PUT DESTINATION ADDRESS INTO DE
            LHLD BM_S_ADD  ;GET STARTING ADDRESS

BM_LOOP1:   MOV A,M
            STAX D
            INX H
            INX D
```

```
                DCR C
                JNZ BM_LOOP1
                MOV A,B
                CPI 00H
                JZ BM_END1
                DCR B
                JNZ BM_LOOP1
BM_END1:        CALL BLK_DISP
                MVI A,0EH
                STA DISPDATA
                CALL DATADISP

                RET



STAT_ADD:       LXI H,DISPDATA+7    ;STARTING ADDRESS OF BLOCK
                                    ;MOVE
                MVI M,05H       ;5 IS EQUIVALENT TO S
                DCX H
                MVI M,0AH       ;TO DISPLAY A
                DCX H
                MVI M,0DH       ;TO DISPLAY D
                DCX H
                MVI M,0DH       ;TO DISPLAY D
                RET


END_ADD:        LXI H,DISPDATA+7  ;END ADDRESS OF BLOCK MOVE
                MVI M,0EH       ;TO DISPLAY E
                DCX H
                MVI M,0AH       ;TO DISPLAY A
                DCX H
                MVI M,0DH       ;TO DISPLAY D
                DCX H
                MVI M,0DH       ;TO DISPLAY D
                RET

DIST_ADD:       LXI H,DISPDATA+7   ;DESTINATION ADDRESS OF BLOCK
                                    ;MOVE
                MVI M,0DH       ;TO DISPLAY D
                DCX H
                MVI M,0AH       ;TO DISPLAY A
                DCX H
                MVI M,0DH       ;TO DISPLAY D
                DCX H
                MVI M,0DH       ;TO DISPLAY D
                RET
```

```
EXEC_SUB:   STA EXEC_RAM
            CALL GET_ADD
            SHLD REG_PC
            CALL BLK_DISP
            MVI A,0EH
            STA DISPDATA
            CALL DATADISP
            LHLD REG_PC
            PCHL          ;ONLY RESET CAN STOP EXECUTION
            NOP
            NOP
            HLT

GO_SUB:     STA GO_RAM
            MVI A,90H
            OUT 41H

            MVI A,FDH     ;CODE TO DISPLAY '_'
            MVI C,08H


G_S_L1:     OUT 40H
            DCR C
            JNZ G_S_L1    ;GO SUBROUTINE LOOP 1

            MVI A,00H     ;NEXT/PREVIOUS KEY WILL BE
            STA NEXT_RAM  ;PRESSED FIRST TIME AFTER
            STA PREV_RAM  ;THIS GO KEY

            MVI A,00H
            STA DA_OR_AD;NOW ADDRESS NEEDS TO ROTATE FOR
                        ;0 TO F
            RET

PREV_SUB:   MOV B,A
            MVI A,01H
            STA DA_OR_AD;NOW DATA NEEDS TO ROTATE FOR
                        ;0 TO F

            LDA NEXT_RAM;IF KEY IS PRESSED AFTER NEXT SO
            CMP B     ;JUST DECREMENT ADDRESS & DISPLAY
                        ;DATA
            JZ P_S_OUT3 ;OTHERWISE DISPLAY DATA WITHOUT
                        ;DECREMENT
            LDA PREV_RAM
            CMP B
            JNZ P_S_OUT1  ;PREVIOUS SUBROUTINE OUT1
```

```
P_S_OUT3:   LXI H,DISPDATA+4
            MVI A,0FFH
            DCR M
            CMP M
            JNZ P_S_OUT2    ;JNZ TO PREVIOUS SUBROUTINE OUT2

            MVI M,0FH
            INX H
            DCR M
            CMP M
            JNZ P_S_OUT2

            MVI M,0FH
            INX H
            DCR M
            CMP M
            JNZ P_S_OUT2

            MVI M,0FH
            INX H
            DCR M
            CMP M
            JNZ P_S_OUT2

            MVI A,0FH
            STA DISPDATA+4
            STA DISPDATA+5
            STA DISPDATA+6
            STA DISPDATA+7

P_S_OUT2:   CALL GET_ADD    ;TO STORE DATA INTO MEMORY
            INX H           ;PREVIOUS DATA NEEDS TO STORE
            LDA DISPDATA+1  ;GET MSB DATA
            RRC
            RRC
            RRC
            RRC
            MOV C,A
            LDA DISPDATA
            ORA C
            MOV M,A         ;DATA WILL BE STORED AT PREVOUS
                            ;ADDRESS
P_S_OUT1:   MOV A,B
            STA PREV_RAM
            LDA FM_RAM   ;CHECK WHETHER FILL MEMORY KEY
                         ;WAS PRESSED
            CPI FM       ;OR NOT?

            JNZ P_S_E1      ;JNZ TO PREVIOUS SUBROUTINE END1
```

```
                CALL GET_ADD    ;CALL GET ADDRESS FROM DISPDATA
                                ;NOW ADDRESS IS IN HL PAIR

                CALL GET_DATA   ;GET DATA OF ADDRESS IN HL INTO
                                ;ACC
                                ;& INTO D(MSB ONLY) & E(LSB ONLY)
                MOV A,E
                STA DISPDATA
                MOV A,D
                STA DISPDATA+1
                CALL DATADISP
P_S_E1:         RET


NEXT_SUB:       MOV B,A
                MVI A,01H
                STA DA_OR_AD;NOW DATA NEEDS TO ROTATE FOR
                                ;0 TO F
                LDA NEXT_RAM
                CMP B
                JNZ N_S_OUT1    ;NEXT SUBROUTINE OUT1

                LXI H,DISPDATA+4
                MVI A,10H
                INR M
                CMP M
                JNZ N_S_OUT2    ;JNZ TO NEXT SUBROUTINE OUT2

                MVI M,00H
                INX H
                INR M
                CMP M
                JNZ N_S_OUT2

                MVI M,00H
                INX H
                INR M
                CMP M
                JNZ N_S_OUT2

                MVI M,00H
                INX H
                INR M
                CMP M
                JNZ N_S_OUT2

                MVI A,00H
                STA DISPDATA+4
                STA DISPDATA+5
```

```
                STA DISPDATA+6
                STA DISPDATA+7

N_S_OUT2:    CALL GET_ADD    ;TO STORE DATA INTO MEMORY
                DCX H          ;PREVIOUS DATA NEEDS TO STORE
                LDA DISPDATA+1  ;GET MSB DATA
                RRC
                RRC
                RRC
                RRC
                MOV C,A
                LDA DISPDATA
                ORA C
                MOV M,A        ;DATA WILL BE STORED AT PREVOUS
                                   ;ADDRESS

N_S_OUT1:    MOV A,B
                STA NEXT_RAM
                LDA FM_RAM   ;CHECK WHETHER FILL MEMORY KEY
                                   ;WAS PRESSED
                CPI FM       ;OR NOT?

                JNZ N_S_E1      ;JNZ TO NEXT SUBROUTINE END1
                CALL GET_ADD    ;CALL GET ADDRESS FROM DISPDATA
                           ;NOW ADDRESS IS IN HL PAIR

                CALL GET_DATA   ;GET DATA OF ADDRESS IN HL INTO
                                         ;ACC  & INTO D(MSB ONLY) &
                                         ;E(LSB ONLY)
                MOV A,E
                STA DISPDATA
                MOV A,D
                STA DISPDATA+1
                CALL DATADISP
N_S_E1:         RET

GET_DATA:    MOV A,M
                MOV B,A
                ANI 0FH
                MOV E,A     ;LSB ONLY

                MOV A,B
                ANI 0F0H
                RRC
                RRC
                RRC
                RRC
                MOV D,A      ;MSB ONLY

                MOV A,M      ;BOTH NIBBLES (BYTE)IS IN ACC
```

```
                RET

GET_ADD:    LDA DISPDATA+7
            CPI 11H
            JNZ OUT_GA1     ;OUT GET ADDRESS 1
            MVI A,00H

OUT_GA1:    RRC
            RRC
            RRC
            RRC
            MOV H,A

            LDA DISPDATA+6
            CPI 11H
            JNZ OUT_GA2     ;OUT GET ADDRESS 2
            MVI A,00H

OUT_GA2:    ORA H
            MOV H,A

            LDA DISPDATA+5
            CPI 11H
            JNZ OUT_GA3     ;OUT GET ADDRESS 3
            MVI A,00H

OUT_GA3:    RRC
            RRC
            RRC
            RRC
            MOV L,A

            LDA DISPDATA+4
            CPI 11H
            JNZ OUT_GA4     ;OUT GET ADDRESS 4
            MVI A,00H

OUT_GA4:     ORA L
            MOV L,A

            RET

F_M_SR:     STA FM_RAM
            CALL BLK_DISP

            MVI A,90H
            OUT 41H

            MVI A,FEH       ;CODE TO DISPLAY '.'
            MVI C,08H
```

```
F_M_L_1:    OUT 40H
            DCR C
            JNZ F_M_L_1     ;FILL MEMORY LOOP 1

            MVI A,00H       ;NEXT/PREVIOUS KEY WILL BE
            STA NEXT_RAM    ;PRESSED FIRST TIME AFTER
            STA PREV_RAM    ;THIS FILL MEMORY KEY

            MVI A,00H
            STA DA_OR_AD;NOW ADDRESS NEEDS TO ROTATE FOR
                        ;0 TO F

            RET


DATADISP:   MVI A,90H
            OUT 41H
            MVI B,08H
            LXI H,DISPDATA

D_C_1:      LXI D,FND_CODE
            MOV A,M
            ADD E
            MOV E,A
            LDAX D
            OUT 40H
            INX H
            DCR B
            JNZ D_C_1

            RET

CHK_0TOF:   LDA DATA_MEM
            MVI B,00H
            CPI D0          ;IS IT ZERO?
            JNZ OUT0

            JMP LAST_1

OUT0:       INR B
            CPI D1      ;IS IT ONE?
            JNZ OUT1

            JMP LAST_1

OUT1:       INR B
            CPI D2
            JNZ OUT2
```

```
                JMP LAST_1

OUT2:           INR B
                CPI D3
                JNZ OUT3

                JMP LAST_1

OUT3:           INR B
                CPI D4
                JNZ OUT4

                JMP LAST_1

OUT4:           INR B
                CPI D5
                JNZ OUT5

                JMP LAST_1

OUT5:           INR B
                CPI D6
                JNZ OUT6

                JMP LAST_1

OUT6:           INR B
                CPI D7
                JNZ OUT7

                JMP LAST_1

OUT7:           INR B
                CPI D8
                JNZ OUT8

                JMP LAST_1

OUT8:           INR B
                CPI D9
                JNZ OUT9

                JMP LAST_1

OUT9:           INR B
                CPI DA
                JNZ OUTA

                JMP LAST_1
```

```
OUTA:       INR B
            CPI DB
            JNZ OUTB

            JMP LAST_1

OUTB:       INR B
            CPI DC
            JNZ OUTC

            JMP LAST_1

OUTC:       INR B
            CPI DD
            JNZ OUTD

            JMP LAST_1

OUTD:       INR B
            CPI DE
            JNZ OUTE

            JMP LAST_1

OUTE:       INR B

LAST_1:     LDA DA_OR_AD    ;DATA OR ADDRESS
            CPI 00H         ;IF ADDRESS NEEDS TO ROTATE SO
                                ;DA_OR_AD=00H
            JNZ NEXT1       ;IF DATA NEEDS TO ROTATE SO
                                ;DA_OR_AD=01H
            CALL ADD_ROT
            JMP LAST1

NEXT1:      CALL DATA_ROT

LAST1:      CALL DATADISP
            RET


DATA_ROT:   LDA DISPDATA
            STA DISPDATA+1

            MOV A,B
            STA DISPDATA
            RET

ADD_ROT:    LDA DISPDATA+6
            STA DISPDATA+7
```

```
                LDA DISPDATA+5
                STA DISPDATA+6

                LDA DISPDATA+4
                STA DISPDATA+5

                MOV A,B
                STA DISPDATA+4

                RET

DELAY:          PUSH PSW
DEL1:           DCX D
                MOV A,E
                ORA D
                JNZ DEL1
                POP PSW
                RET


                .END
```

CHAPTER – 8 DIGITAL TO ANALOG CONVERTER

There are various situations where we needs to convert the digital data into equivalent analog voltages. Such a converter chip is called digital to analog converter, or briefly DAC. There are various DAC chips available in the market. We have selected DAC – 0808 for interfacing with our microprocessor trainer kit.[8]

We have developed this circuit to illustrate that how one can interface the popular DACs like DAC – 0808.

In the present interfacing circuit we have converted the 8-bit digital input to a maximum +5V analog output.

8.1 DAC – 0808 Chip

It is a 16-pin device, available in DIP (Dual Inline Package) and SOP (Small outline Package) packages. It is an 8-bit monolithic digital to analog converter chip, providing current as a output. In this chip reference current (I_{ref}) trimming is not required, as the full scale output current is typically $\pm$1 LSB of 256 I_{ref}/ 256. The power supply currents of the chip are independent of bit codes, and exhibits essentially constant device characteristics over the entire supply voltage range.

This chip accepts TTL, DTL or CMOS logic levels. The features of DAC – 0808 are as follows.

- Relative accuracy: $\pm$ 0.19% error maximum.
- Full scale current match: $\pm$ 1 LSB typical
- 7 and 6-bit accuracy available
- Fast setting time: 150 ns typical
- Non inverting digital inputs are TTL and CMOS compatible
- High speed multiplying input slew rate: 8mA / microsecond
- Power supply voltage range: $\pm$ 4.5V to $\pm$ 18V
- Low power consumption: 33mW @ $\pm$ 5V

Block diagram

Figure 8.1.1. shows the block diagram of DAC – 0808.

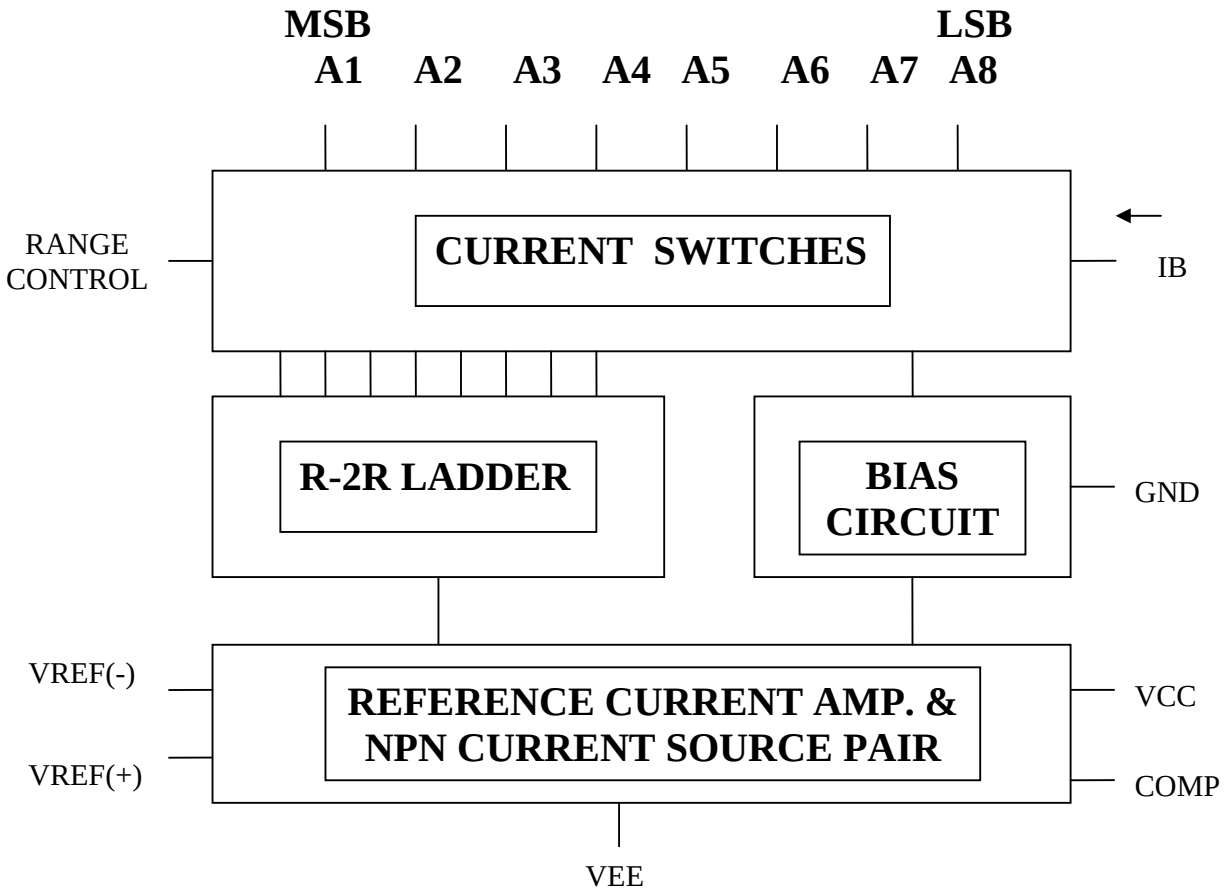


Figure-8.1.1 BLOCK DIAGRAM OF DAC-0808

As shown in the block diagram the binary 8-bit data is to be applied at pins A_1 to A_8, such that MSB appears at A_1 and LSB appears at A_8.

The block “current switches” takes care of the various types of digital low level and high level voltages. The R-2R ladder unit provides the weighted contribution of each input bit. In association with + V_{ref} and – V_{ref} the R-2R ladder generates proper value of equivalent I_{out}.

This is obtained with the help of NPN current source pair and reference current amplifier. The bias circuit takes care of proper biasing conditions of the internal components.

Figure 8.1.2 shows the pin diagram of DAC – 0808.

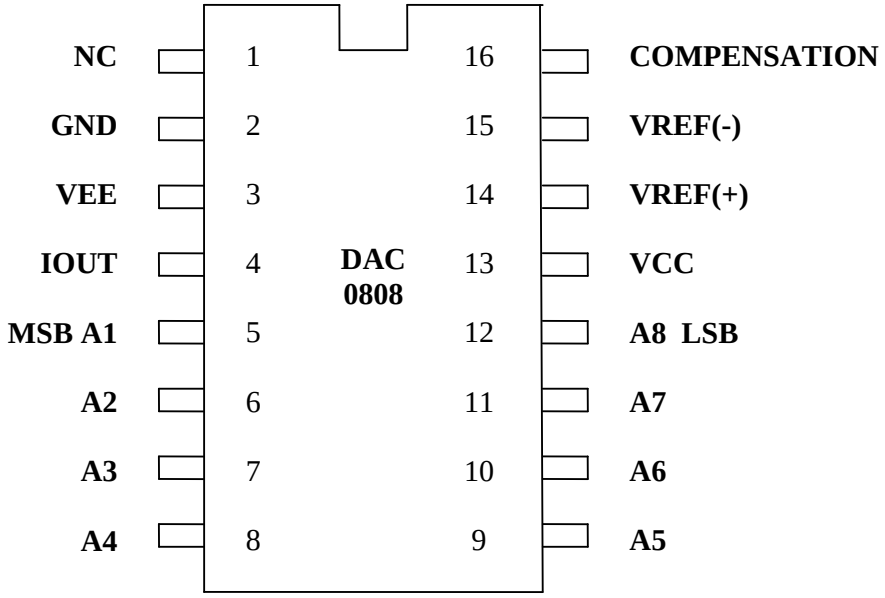


Figure-8.1.2 PIN DIAGRAM OF DAC-0808

Pin – 1 NC:

This is not used in the circuit. So the user can leave it unused.

Pin – 2 GND:

This is ground pin all potentials are measured with respect to this pin.

Pin – 4 Iout:

This pin gives the output current. This current is proportional to the input digital data.

Pin – 5 to 12 A_1 to A_8:

These are the eight input pins. The digital data is received at these eight pins. These pins are capable to accept TTL based logical signals.

Pin – 13 V_{CC}:

This pin is positive supply pin. The designer needs to supply positive voltage with respect to ground at this pin.

Pin – 14 Vref+:

This is positive reference pin. By varying the current at this pin we can calibrate the output current.

Pin – 15 Vref-:

This is negative reference pin. This pin should be connected at ground potential.

Pin – 16 Comp:

The capacitor of 0.01 microfarad is connected between this pin and negative supply pin. The output current can be converted into voltage by adding current to voltage converter.

8.2 Basic interfacing circuit with P.C.B. aspect

The interfacing details of DAC – 0808 is shown in Figure – 8.2.1. The proper voltage conditions required and + V_{ref} are obtained by external means through a 9-pin D-type connector. The voltages at these pins are V_{EE} = -15Volt and V_{CC} = +15Volt. In this application we need positive output voltage so, -V_{ref} is grounded. The 8-bit digital input is obtained from Port-A through I/O connector, where PA0 connects with A_8 (LSB) & PA_7 connects with A_1 (MSB).

For compensation of phase a capacitor C_1 of value 0.01Mf is connected between COMP (pin no-16) and V_{EE} (pin no-3). The resultant output current at pin no-4 (I_{out}) is inputted to the inverting pin of OP-AMP 741 which is designed as current to voltage converter. The adjustment of the V_{out} can be done with the help of feed back resistor R_3.

Circuit Function

In the present interfacing circuit digital input data is supplied through 8255 located in the microprocessor kit. In the software 8255 is configured such that Port-A becomes output port. The desired data is output through Port-A at the input pins of DAC – 0808. This gets immediately converted into equivalent analog output, which can be measured by a Multimeter.

P.C.B. aspects

The track layout pattern for the DAC-0808 module is designed using computer. The track layout for the same(bottom layer) is shown in Figure-8.2.2(a) and its overlay drawing is shown in Figure 8.2.2(b).

Software analysis

To get the full scale output voltage the below given program can be used.

```
M V I A, 80 H ; Control word
O U T 03 H
M V I A, FF H
O U T 00 H
H L T
```

The control word 80h will initialize port-A as output port, then send 00h/ffh at port-A, then by potentiometer adjust 0v/5v at output.

Figure 8.2.1 SCHEMATIC CIRCUIT OF DAC-0808 CARD

FIGURE 8.2.2(A) BOTTOM LAYER OF P.C.B. LAYOUT

FIGURE 8.2.2(B) OVERLAY OF P.C.B. LAYOUT

8.3 Examples.

1. You are given the microprocessor kit and DAC – 0808 based interfacing module. After applying proper power supplies develop program to get 2.5V at output. Fill up the below given table.

Input digital data	Output Voltage
00H	
01H	
02H	
04H	
10H	
20H	
40H	
80H	

2. You are given the microprocessor kit and DAC – 0808 based interfacing module. After applying proper power supplies develop program to generate square Wave. The low Voltage will be zero and high Voltage will be 4.5V and show it on CRO.

3. You are given the microprocessor kit and DAC – 0808 based interfacing module. After applying proper power supplies develop program to generate triangular wave and show it on CRO.

CHAPTER – 9 ANALOG TO DIGITAL CONVERTER

Many physical entities are now measured by means of digital equipments. For examples, digital readout of temperature, pressure etc. Such entities are sensed through its corresponding transducer. The transducer generates the analog voltage or current. To obtain digital representation of these entities their transducer output analog voltage or current are to be converted into digital format.

There are various means to convert analog voltage or current into equivalent digital outputs. The devices which perform this are called Analog to digital converter. There are few different techniques being used for different ADCs.

Basically, the ADCs sample the data, quantaize and encode then into digital format. The very famous technique, which is used as a part of conversion process is successive approximation.

Now a day ADC ICs are available, which are microprocessor compatible.

9.1 ADC – 0808 Chip.

In the present work we have considered ADC – 0808.[8] It is an 8-bit analog to digital converter with 8-channel in built multiplexers. It is the monolithic CMOS device manufactured by the National Semiconductor.

The 8-bit A/D converter uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register.

The 8-channel multiplexer can directly access any of 8-single-ended analog signals.

The device eliminates the need for external zero and full scale adjustments. Easy interfacing to microprocessor is provided by the latched and decoded multiplexer address inputs and latched TTL TRI-STATE outputs.

The design of the ADC – 0808 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. The ADC – 0808 offers high speed, high accuracy, minimal temperature dependence, excellent long-term accuracy and repeatability, and consumes minimal power. These features make this device ideally suited to applications from process and machine control to consumer and automotive applications.

Features

- Easy interface to all microprocessors
- Operates ratiometrically or with 5 V_{DC} or analog span adjusted voltage reference
- No zero or full scale adjustment required
- 8-channel multiplexer with address logic
- 0V to 5V input range with single 5V power supply
- Outputs meet TTL voltage level specifications
- Standard hermetic or molded 28-pin DIP package
- 28-pin molded chip carrier package

Key Specifications

- Resolution — 8 Bits
- Total Unadjusted Error — ± ½ LSB and ± 1 LSB
- Single Supply — 5 V_{DC}
- Low Power — 15 mW
- Conversion Time — 100 us

Block diagram

Figure 9.1.1. illustrate the block diagram of ADC – 0808.
ADC – 0808 consists of 8-main parts.

1. 8-channel multiplexing analog switches
2. Address Latch and Decoder
3. Comparator
4. Control and Timing
5. Successive Approximation register
6. Switch tree
7. 256R resistor ladder
8. Tri-state output Latch buffer

1. **8-channel multiplexing analog switches**.

This unit receives analog voltages to be converted into digital on eight different pins called Channel. This input can be single ended analog signal.

2. **Address latch and decoder.**

This block has four inputs in which three are address lines and one is ALE (Address Latch Enable) pin. Following Table 8.1.1. shows the input states for the address lines to select any channel. The address is latched into the decoder on the low to high transition of the ALE Signal.

Table 9.1.1

Selected Analog Channel	Address lines		
	C	**B**	**A**
IN 0	L	L	L
IN 1	L	L	H
IN 2	L	H	L
IN 3	L	H	H
IN 4	H	L	L
IN 5	H	L	H
IN 6	H	H	L
IN 7	H	H	H

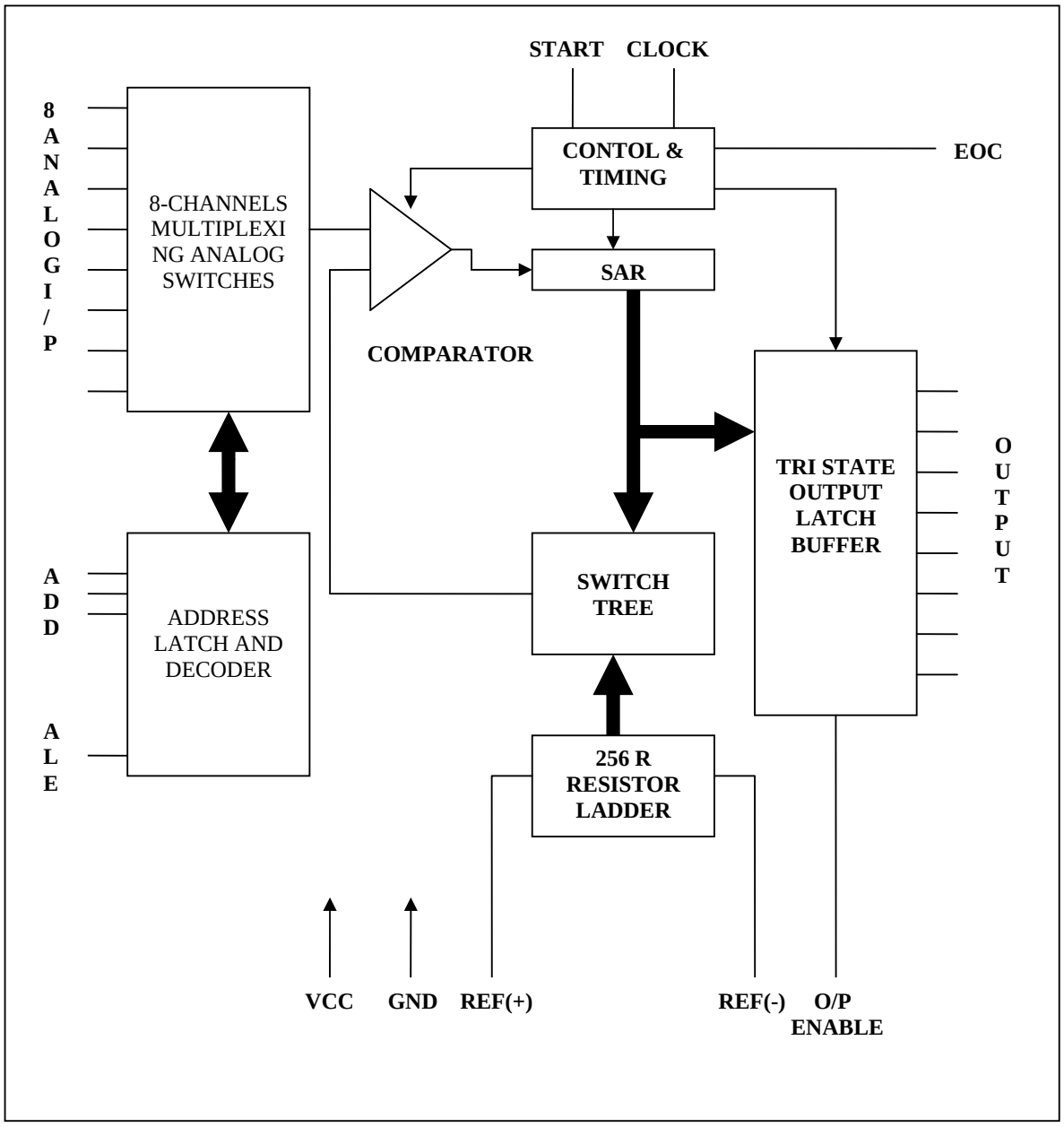


Figure 9.1.1. Block diagram of ADC – 0808

3. **Comparator**

This unit compares the analog input on any one channel with the intermediate analog voltage converted by successive approximation.

4. **Control and Timing**

This unit controls the various functions of other units, it has start, clock and EOC lines. It controls the comparator, SAR and output latch buffer.

5. **Successive Approximation Register**

This is an eight bit register which performs eight iterations to approximate the input voltage.

It is reset on the positive edge of the start conversion (SC) pulse. The conversion begins on the falling edge of the start conversion pulse. A conversion in process will be interrupted by receipt of a new start conversion pulse. Continuous conversion pulse may be accomplished by tying the End of Conversion (EOC) out to the A.C. input, if used in this mode an external start conversion pulse should be applied after power-up. EOC will go low between zero and eight clock pulses after the rising edge of start conversion (SC).

6. **Switch tree**

This unit consists of number of in built switches. These switches pass the voltages of 256R resistor ladder. It gives output voltage to the comparator. This output voltage is the result of ladder network voltage under the control of S.A.R. Figure 9.1.2. shows the register ladder and switch tree.

7. **256-R resistor ladder**

This unit is a network of 256 resistors. Hence, it is called 256R resistor ladder. It is better then R-2R ladder because of its inherent monotonocity. Which guarantees no missing digital codes, it does not cause low variations on the reference voltage. The network is shown in Figure – 8.1.2. The bottom resistor and the top resistor of the ladder network are not the same value as the remainder of the network. The difference in these resistors causes the output characteristics to be symmetrical with the zero, and full scale points of the transfer curve.

8. **Tri-state output latch buffer**

This unit is controlled by Control and Timing unit and Output Enable (OE) pin. It outputs the final contents of S.A.R. i.e. the converted digital data on the eight different output pins of the chip.

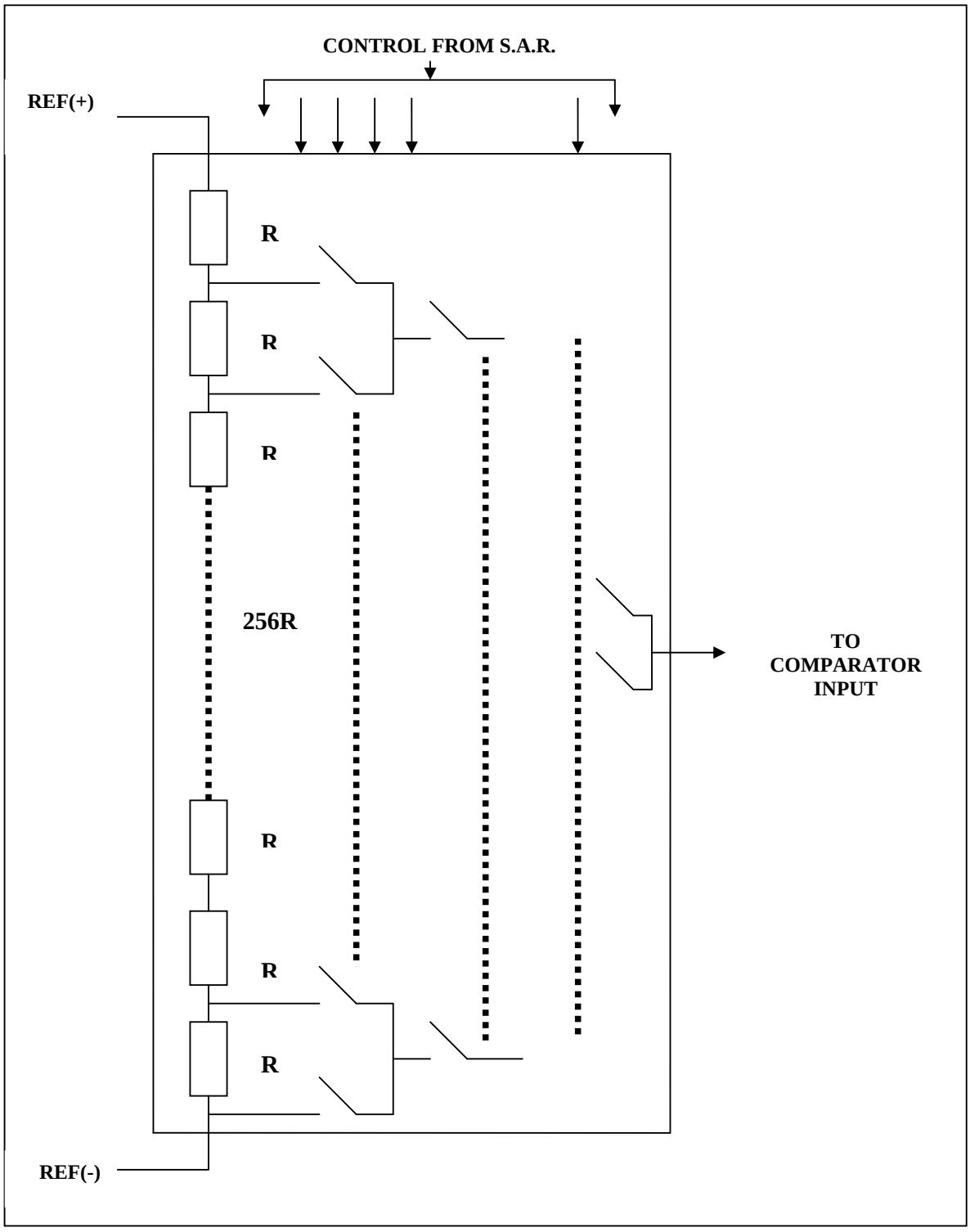


Figure 9.1.2 Resistor ladder and switch tree

Pin diagram & Timing diagram

The pin diagram of ADC – 0808 is shown in Figure 9.1.3. In pin diagram IN_0 to IN_7 are 8-input pins to take in eight different analog voltages. The selection of any of these input channel is controlled by the address pins ADD A, ADD B and ADD C. The start pin accepts a pulse for starting conversion. EOC pin provides the high status when conversion is over. OE pin when receives high causes the digital data stored in latch buffer to output on the output lines. Here output lines are 2^{-1} (MSB), 2^{-2}….2^{-8} (LSB). ALE is used to latch the address on ADD A, ADD B and ADD C into the internal latch. $V_{Ref(+)}$ and $V_{Ref(-)}$ accepts the positive and negative voltage respectively for the internal 256R ladder network.

Pin – 1, 2, 3, 4, 5, 26, 27, 28 : IN0 to IN7

These pins are input channel pins. The voltage between this pin and ground can be converted into digital.

Pin – 6 : START

The analog to digital conversion Starts after receiving a pulse at this pin.

Pin – 7 : E0C

This pin is known as End of conversion pin. This pin goes low when conversion Starts and becomes high when conversion ends.

Pin – 8, 14, 15, 17, 18, 19, 20, 21 : Data lines

These are the data pins. The converted digital data appear at these pins. The pin numbers and its labels are Shown in Figure 9.1.3..

Pin – 9 : 0E

This pin is known as output Enable. After conversion the digital data will be read if this pin is at high logic.

Pin – 10 : CLK

This is Clock pin. To convert analog data into digital form we need to apply Clock. The conversion time is dependent upon the frequency of the Clock.

Pin – 11 : V_{CC}

This is the supply pin +5V is provided at this pin through 40 – pin FRC connector.

Pin – 12 : REF+

This is positive reference pin. This pin is shorted with V_{CC} pin.

Pin – 13 : GND

This pin is the ground pin. The voltage at all points have been measured with respect to this pin.

Pin – 16 : REF-

This is negative reference pin. This pin is shorted with ground pin.

Pin – 22 : ALE

This pin is known as Address Latch Enable. The status on A_0, A_1 & A_2 of DAC – 0808 will be latched by pulse at this pin.

Pin – 23, 24 & 25 : A_2, A_1 & A_0

These are the three address pins. These pins select analog input channel.

Timing diagram

As shown in Figure – 9.1.4, first of all analog inputs are to be made stable at the input channels. There after from the microprocessor proper address is provided on address lines. This address is available for very short time in its stable form, so a signal on ALE pin of ADC – 0808 is to be applied during this time i.e. during the first clock pulse appearing at the clock pin of ADC – 0808. After that microprocessor is supposed to provide a start conversion pulse.

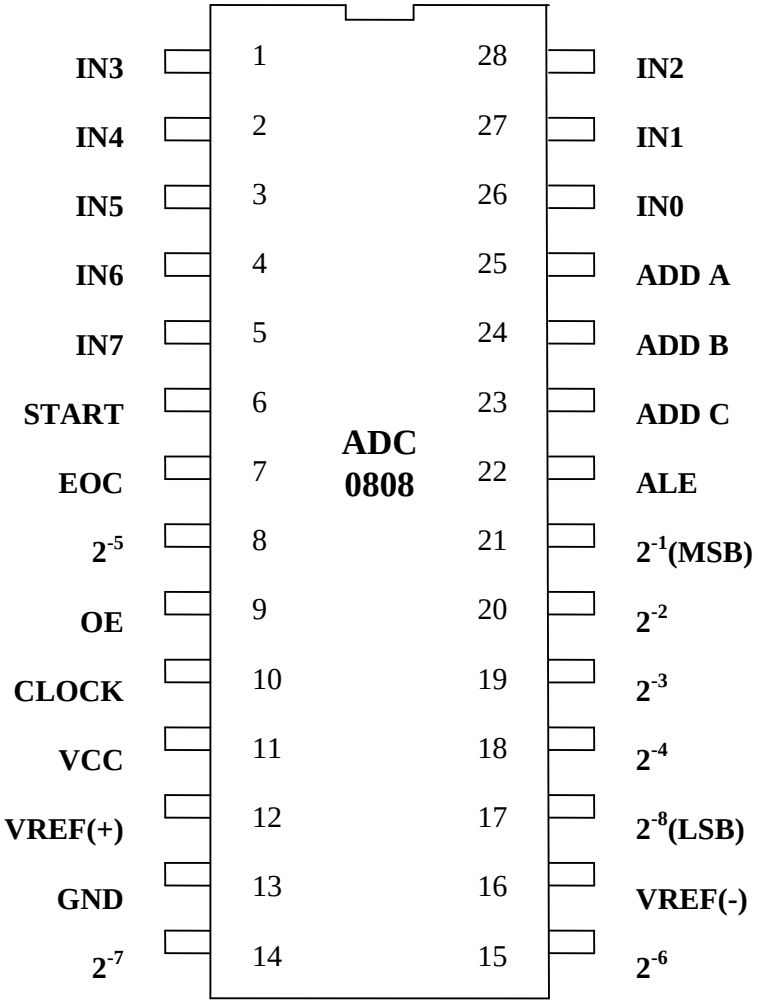


Figure 9.1.3. – Pin diagram of ADC – 0808

When conversion is over EOC pin goes low for certain time. After this time EOC again becomes high microprocessor should sense this last

transition of EOC, and then should send a positive pulse on OE pin. This will put the converted data on to the output line of ADC – 0808.

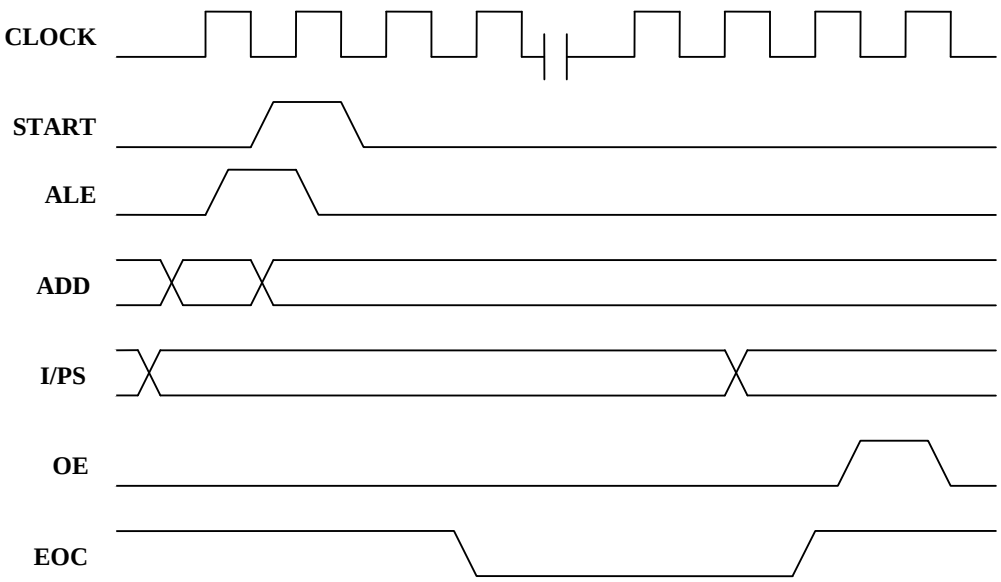


FIGURE 9.1.4 TIMING DIAGRAM

9.2. Basic interfacing circuit with P.C.B. aspects

We have designed and developed an interfacing circuits to illustrate the use of ADC-0808 with the 8085A microprocessor.[9]

The interfacing circuit for ADC – 0808 is shown in Figure 9.2.1.The 40 pin FRC connector connects all the pins of ADC – 0808 with 8255, V_{CC} and ground which is on board on the 8085 card except input channels.

Port-A and Port-B are used as input port, while Port-C is used as output port of 8255. Port-C generates the timing pulses for OE, ALE, Start and CLK. It also generates address on A_0, A_1 and A_2 pin of ADC – 0808. In the software the proper address is generated by Set-Reset of PC_5, PC_6 & PC_7.

There after, ALE is generated which latches the address into ADC – 0808. $Ref_{(+)}$ and $Ref_{(-)}$ are connected with V_{CC} and ground respectively. In the present design we connected on channel-7 the +5V supply and its converted data was read by Port-A. The EOC signal is sensed by Port-B. When EOC becomes valid the software generates OE through PC_2 to read the data on Port-A. One can use any channel by considering the corresponding proper address on A_0, A_1 and A_2.

FIGURE 9.2.1. SCHEMATIC CIRCUIT OF ADC

P.C.B. aspects

The track layout pattern for the ADC-0808 module is designed using computer. The track layout for the same (bottom layer) is shown in Figure-9.2.2(a) and its overlay drawing is shown in Figure 9.2.2(b). Note that in overlay the straight lines show jumper wires.

This interfacing module can be connected with microprocessor kit through 40 – pin FRC connector. The tested program is given below. This software can be loaded into memory of the kit.

Software

```
HL_DISP:        .EQUAL      1C1CH
DATADISP:       .EQUAL      1E65H
DELAY:          .EQUAL      1F40H
ADC:            .EQUAL      0200H

            ORG ADC

        LXI SP, 5EFFH
        MVI A, 92H      ;CONTROL WORD
        OUT 03H

        MVI A, 0FH      ;TO SET PC7 CHANNAL ADDRESS
        OUT 03H
        MVI A, 0DH      ;TO SET PC6 CHANNAL ADDRESS
        OUT 03H
        MVI A, 0BH      ;TO SET PC5 CHANNAL ADDRESS
        OUT 03H

START:  MVI A, 03H      ;TO SET PC1 ALE PIN
        OUT 03H

        MVI A, 01H      ;TO SET PC0 START PIN
        OUT 03H
        NOP
        NOP
        NOP
        MVI A, 02H      ;TO RESET PC1 ALE PIN
        OUT 03H

        MVI A, 00H      ;TO RESET PC0 START PIN
        OUT 03H

LOOP1:  MVI A, 07H      ;TO SET PC3 CLOCK PIN
        OUT 03H

        NOP
        NOP
        NOP
        MVI A, 06H      ;TO RESET PC3 CLOCK PIN
```

```
OUT 03H

IN 01H
ANI 80H        ;EOC PIN IS CONNECTED WITH PB7
JZ LOOP1

MVI A, 05H     ;TO SET OUTPUT ENABLE PIN
OUT 03H

NOP
NOP
IN 00H
MOV L, A

MVI A, 04H     ;TO RESET OUTPUT ENABLE PIN
OUT 03H

MVI H, 00H
CALL HL_DISP
CALL DATADISP

LXI D, FFFFH
CALL DELAY
LXI D, FFFFH
CALL DELAY
JMP START
.END
```

Result

This ADC – 0808 is capable to convert given analog data into 8- bit digital data. To set any one bit out of eight bits, we should apply proper data, these types of observations are shown in Table – 9.2.1.

Table-9.2.1

Observation No.	Analog input voltage	Digital output
1	0volt	00000000b
2	0.024 volt	00000001b
3	0.04 volt	00000010b
4	0.081 volt	00000100b
5	0.16 volt	00001000b
6	0.32 volt	00010000b
7	0.61 volt	00100000b
8	1.2 volt	01000000b
9	2.4 volt	10000000b
10	4.82 volt	11111111b

Here it is shown that the weight of next MSB is almost double then any bit.

FIGURE 9.2.2(A) BOTTOM LAYER OF ADC-0808 CARD

FIGURE 9.2.2(B) OVERLAY OF ADC-0808 CARD

Figure-9.2.3 Captured signals by Logic analyzer

The logic analyzer can be used to study the waveforms. The timing signals captured by logic analyzer are shown in Figure. 9.2.3. Here we have captured CLOCK, START, ALE, ADD_0 – 2 OE, EOC and D_0 – D_7.

One can see that ALE and START are generated at the beginning. ADD – 0 to ADD – 2 are kept at high logic to select channel – IN_7. The clock signals are applied when end of conversion signals goes low, from this point conversion starts.

The microprocessor continuously checks the logic at EOC pin. When EOC becomes high then OE signal is made low. The converted data FFH is read on port – A and displayed on FND display.

The conversion time, which is specified in the data sheet of ADC – 0808 is 100 micro – second (for CLOCK frequency of 640 KHz) and the same is verified through logical state analyzer, and it is 1705 micro – second (for CLOCK frequency of 40.84 KHz).

9.3 Examples

Example-1

You are given a microprocessor kit and an interfacing card of ADC – 0808. Write software program to read analog voltage of Channel – IN_7 and store this digital data at some memory location.

Example-2

You are given a microprocessor kit and an interfacing card of ADC – 0808. Write software program to see the digital data on FND displays and find the Weightage of all bits.

CHAPTER – 10 RUNNING CHARACTER DISPLAY

In the series of Interfacing Cards developed with our present microprocessor trainer kit “The Running Character Display” has been included to elucidate the fascination of numbers and alphabets running on the display. This type of display is heavily used in commercial advertisements. Various types of display modules are available for this. There can be verities of displays for this purpose. But among all, the basic working principals are almost same.

In the present running character display design, we have considered 5x7 display module for interfacing 8085A. To drive the display LEDS of the modules buffering was considered.

10.1. Understanding basic Circuit and P.C.B. aspects.

Let us first understand the organization of 5x7 matrix. Figure-10.1.1. shows the arrangement of 5X7 matrix display modules.

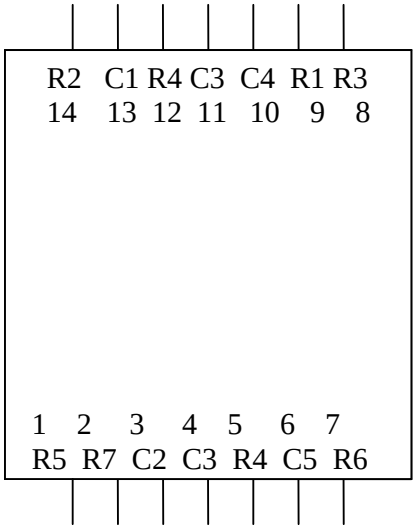


FIGURE-10.1.1 PIN CONFIGURATION OF 5X7

For buffering all LEDS of these three display modules all together, we have used three 8-bit buffers, consisting of ICs 74245. All input lines of these buffer ICs are terminated to a 40-pin FRC connector.

The detailed interfacing circuit for running character display interfacing module is shown in Figure 10.1.2. There are three 5x7 matrix modules, which allow the maximum three characters to be displayed at a time.

Figure 10.1.2:Schematic diagram of Running Character Display

The connector pin details are shown in following Table – 10.1.1.

Table – 10.1.1.

FRC pin no.	Connecting pin from 8255.	Buffer IC pin no.	5x7 matrix module.
1	PB_2	U3 – A6	C_2
2	PB_3	U3 – A5	C_3
3	PB_1	U3 – A7	C_1
4	PB_4	U3 – A4	C_4
5	PB_0	U3 – A8	C_0
6	PB_5	U3 – A3	C_5
7	PC_3	U2 – A5	C_{11}
8	PB_6	U3 – A2	C_6
9	PC_2	U2 – A6	C_{10}
10	PB_7	U3 – A1	C_7
11	PC_1	U2 – A7	C_9
12	--	--	--
13	PC_0	U2 – A8	C_8
14	--	--	--
15	PC_4	U2 – A4	C_{12}
16	--	--	--
17	PC_5	U2 – A3	C_{13}
18	--	--	--
19	PC_6	U2 – A2	C_{14}
20	--	--	--
21	PC_7	--	--
22	--	--	--
23	--	--	--
24	--	--	--
25	GND	GND	--
26	GND	GND	--
27	--	--	--
28	--	--	--
29	--	--	--
30	V_{CC}	V_{CC}	--
31	--	--	--
32	--	--	--
33	PA_0	U1 – A1	R_1
34	PA_7	--	--
35	PA_1	U1 – A2	R_2
36	PA_6	U1 – A7	R_3
37	PA_2	U1 – A3	R_4
38	PA_5	U1 – A6	R_5
39	PA_3	U1 – A4	R_6
40	PA_4	U1 – A5	R_7

This FRC connector is further connected on 8085A CPU Card on I/O connector, which connects all the 24 I/O lines of 8255 and V_{CC}, ground. Thus ultimately the peripheral IC 8255 controls the display modules.

Note that the rows of all the three 5x7 matrix module are interconnected which is not shown in Figure. 10.1.2.

Thus to glow for example the matrix element LED positioned at C_0R_0 we have to supply +5V to R_0 and we have to connect C_0 with the ground through current limiting register. Which demands that in the case PA_0 should be high and PB_0 should be low. This makes the partial coading character code.

Figure – 10.1.3. (a) and (b)shows the organization of the 1.2" DM DOT matrix 5x7 display module. The product No. is KLP 1157I. It is column cathode part supplied by Kwality photonics private Limited.[10]

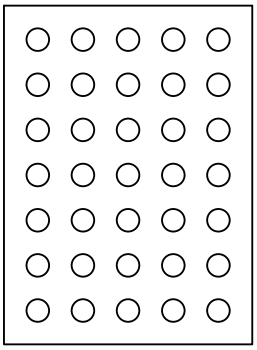


Figure-10.1.3(a) Top view of 5x7 matrix

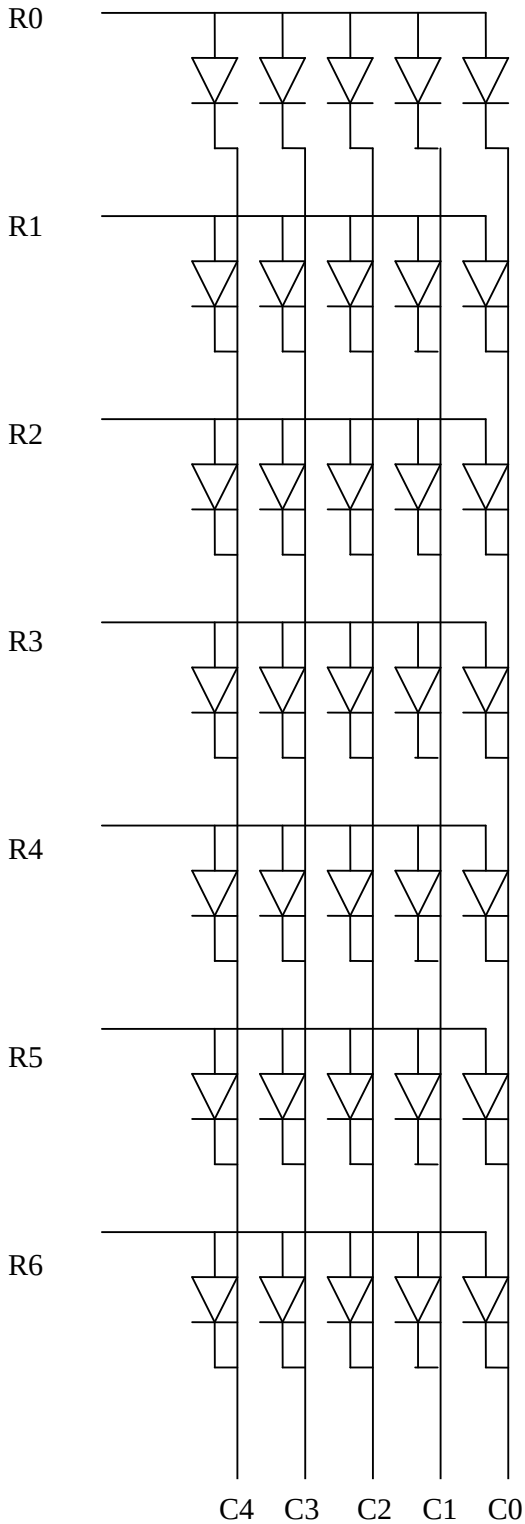


Figure-10.1.3(b)
Internal LEDarrangement

Now the technique to display a character on this 5X7 matrix display is illustrated in Figure 10.1.4.

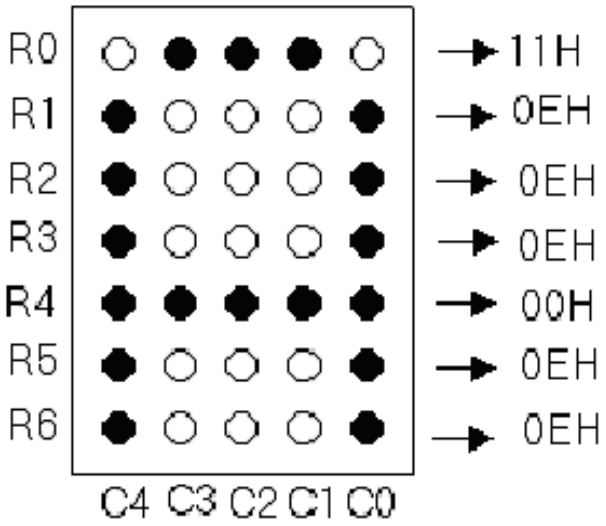


FIGURE 10.1.4 Character 'A' on 5x7 matrix display

In this Figure-10.1.4 how to display a letter 'A' is explained. In this block solid circles are to be glown. For that respective row should be supplied with high signal and columns should be grounded.

Code generation

In running character display a character is displayed in a given display module for very small time. There after a new character is displayed on the same module. This way all the display modules display different character at different time. A software program stored in microprocessor kit controls this changing character and time.

To display any single character on a single display module, the microprocessor sends a typical high or low signal to the corresponding LEDs of module. This combination of high and low frames a binary word called character codes. As shown in Figure-10.1.4 a single character will need seven such five bit binary codes. In Figure-10.1.4 the codes for letter 'A' are also shown. Note that for first row R0, the code is 11H. This is because five LEDs correspond to eight-bit data as follows.

because five LEDs correspond to eight-bit data as follows.

x x x 1 0 0 0 1 = 11H

For other rows one can generate codes referring to Figure-10.1.4

These codes are sent by microprocessor through the 8255 ports to display module. In present case the code 11H will display the three center LEDs of R0 of display module. Next microprocessor will send the code 0E to display the two LEDs at the ends of R1. Like these all codes are supplied sequentially and rapidly. So for a viewer the character display is stationary.

Up to this we have discussed for a single display module. For multi character display the various codes of various characters are supplied to the different display matrices maintaining the synchronization, to generate running environment of the characters.

PCB aspects

The track layout pattern for the running character display module is prepared using in house laboratory facility. The track layout drawing for the same (bottom layer) is shown in Figure 10.1.5. (a) and its overlay drawing is shown in Figure 10.1.5. (b).

Note that in Figure 10.1.5. (b) the small line segments indicate the jumper wires.

10.2. Program techniques and example

We know from Figure 10.1.2. that rows of all the three modules together are controlled by Port-A and columns are controlled by Port-B and C, where Port-B controls columns C_0 to C_7 and Port-C controls columns C_8 to C_{14}. Note that PC_7 bit is not connected as there in no C_{15} column.

Figure-10.1.5(a): Bottom layer of P.C.B.

Figure 10.1.5(b) : overlay of P.C.B.

In the program logic following points are very well conceived.

1. For display purpose one row at a time is selected i.e. When PA_0 is made high the row R_0 for all the three modules is selected.

2. Displaying starts from the Right most 5x7 display matrix module.

3. For display purpose instead of selecting group of LEDs as per the row code the individual bit vise LEDs are selected one by one.
   This is done to reduce the power requirement at a time.
   To understand this concept consider Figure 10.1.4. In this figure if we want to display character 'A' on 5x7 matrix-0, we have to first output data 11 H on Port-B, for R_0. If we do so then the middle three LEDs of row-1 of 5x7 matrix-0 will glow. That means at one time all the three LEDs will draw the current. To avoid this we have designed the software such that only one LED of the row of the character code is displayed at a time. For this the character codes for the row have been masked for allowance of only one valid bit to glow corresponding LED. This masked data is sent to the Port-B (And for Port-C as well, as per the requirement). This will glow only one LED at a time. Thus power requirement is reduced. The whole row code is displayed by rotating and masking the row code. This technique provides uniform and bright illumination of the character on the display.

4. All necessary character codes are stored in EPROM. Then they are transferred to the RAM.

5. All three modules are now treated as 7x15 single matrix. This is further divided into two: 7x7 (R_0, R_1, R_2, R_3, R_4, R_5, R_6, C_{14}, C_{13}, C_{12}, C_{11}, C_{10}, C_9, C_8) and 7x8 (R_0 to R_6 and C_7 – C_0) for software purpose. 7x7 is controlled by Port-C and 7x8 is controlled by Port-B.

6. In software to display on 7x8, the other part 7x7 is kept blank by outputting FFh on Port-C and similarly vice-versa.

7. Initially the software displays on arrow as shown in Figure – 10.2.1.
   At the beginning of the program an arrow is shown because it shows the direction of running, as well as to eliminate display of random data.

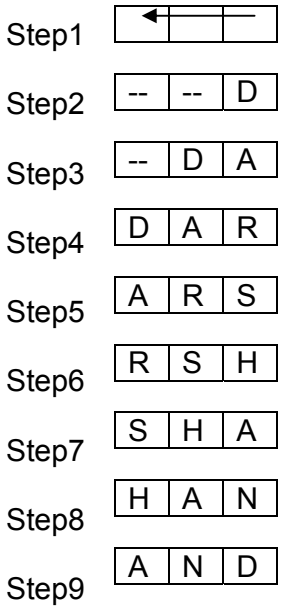


Figure-10.2.1: shifting of the character 'DARSHAN'

8. Now we will consider how the display looks running. Consider Figure 10.2.2. in this figure 7x15 display module is represented in six different steps.

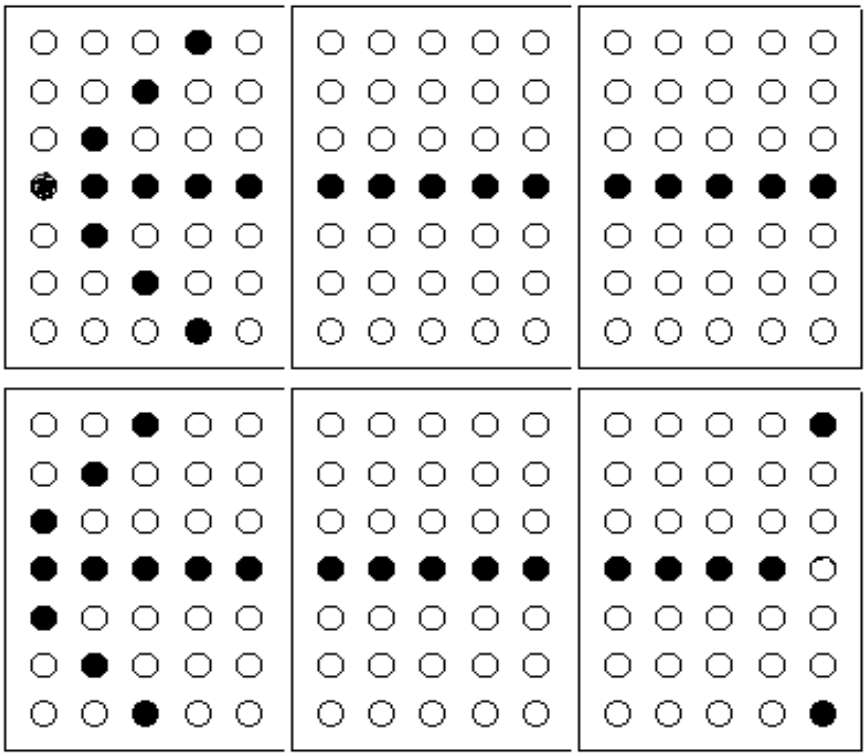


Figure-10.2.2: shifting of the character 'D'(P.T.O.)

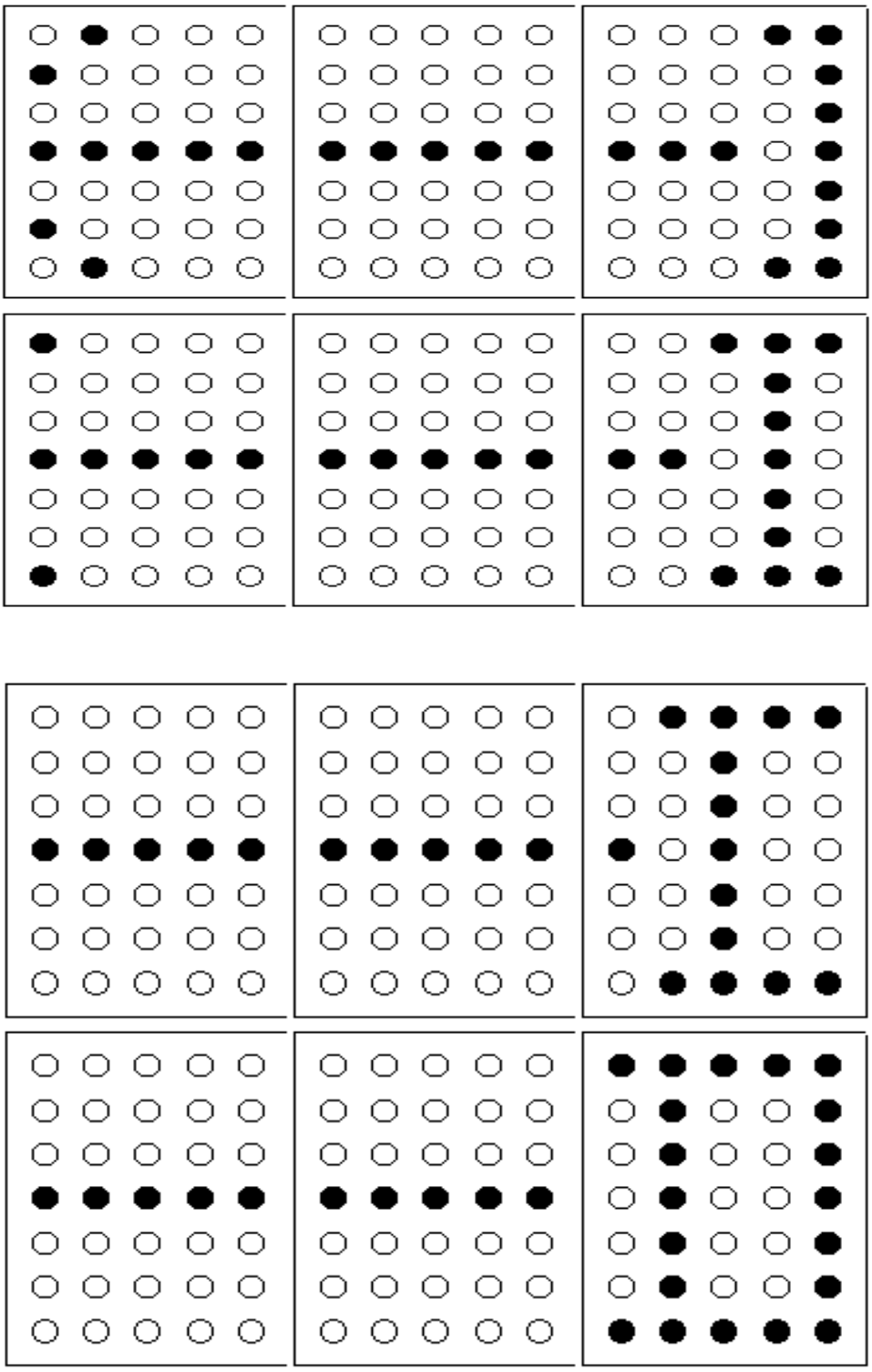


Figure-10.2.2: shifting of the character ‘D’

In step-1 an arrow is shown on display. In step-2 the whole display is shown shifted left by one column, one can see the two LEDs of character ‘D’ being displayed in step-2. In step-3 next column of character-‘D’ gets

displayed with a one more column wise left shifting makes the display “running”. Note that program is written such that once arrow gets shifted completely it does not reappear. Only the message “DARSHAN” gets rotated.

The complete logic of the present software is illustrated in the form of flow chart in Figure 10.2.3.

Following is the detailed listing of the software written for “running character Display” interface.

```
DELAY:      .EQUAL   1F3FH
POINTER:    .EQUAL   5F50H
COUNT:      .EQUAL   5F40H     ;5F40H
COUNT1:     .EQUAL   5F41H     ;5F41H
ROT_COU:    .EQUAL   5F42H     ;Rotate counter
TEMP_HL:    .EQUAL   5F43H     ;5F43H & 5F44H
TEMP_HL1:   .EQUAL   5F45H     ;5F45H & 5F46H
COUNT2:     .EQUAL   5F47H
RCD:        .EQUAL   0850H     ;Running char.disp.
COD_SEQ:    .EQUAL   1500H     ;code seq. in EPROM


    ORG RCD

        LXI SP,5EFFH
        MVI A,80H       ;CONTROL WORD
        OUT 03H

        LXI H,POINTER ;FROM 5F40H TO 5F4EH
        MVI M,FFH       ;CODES TO DISPLAY AN ARROW
        INX H
        MVI M,FFH
        INX H
        MVI M,FFH
        INX H
        MVI M,00H
        INX H
        MVI M,FFH
        INX H
        MVI M,FFH
        INX H
        MVI M,FFH
        INX H
        MVI M,F7H       ;POINTER 7 LOCATION
        INX H
        MVI M,EFH
        INX H
        MVI M,DFH
        INX H
        MVI M,80H
        INX H
```

```
                MVI M,DFH
                INX H
                MVI M,EFH
                INX H
                MVI M,F7H

AGAIN:          MVI A,0CH       ;DARSHAN HAS 7 CHARECTERS
                STA COUNT2          ;Initialize counter 2
                LXI H,COD_SEQ   ;source pointer to code ;sequence in
                                ;EPROM.
                SHLD TEMP_HL1   ;temporary storage
NEX_CHAR:  LHLD TEMP_HL1
                MVI B,07H   ;counts to count 7 codes of ;each character
                LXI D,POINTER+14;Destination pointer to ;next character to be
                                ;displayed
SECOND:     MOV A,M             ;code of character
                STAX D      ;store at pointer + 14 ;location
                INX H       ;increment source
                INX D       ;increment destination
                DCR B
                 JNZ SECOND             ;if all codes are not ;transferred,
                                        ;repeat
                SHLD TEMP_HL1   ;now point to next character ;group


                LXI H,COUNT1    ;COUNTER TO ROTATE
                                                ;CHARECTERS
                MVI M,05H

BEGIN:          LXI H,ROT_COU   ;COUNTER TO DISPLAY
                                                ;CHARECTERS
                MVI M,20H   ;counts to get stable display


START:          MVI B,01H       ;ROW SELECTOR
                MVI C,FEH       ;TO MASK BYTE FOR
                                ;BITWISE ;GLOWING

                LXI H,POINTER
LOOP2:          MOV D,M         ;SAVE DATA IN D

                MOV A,B         ;GET ROW NO.
                OUT 00H     ;ACTIVATE ROW FOR ALL 3 DISPLAYS

LOOP1:          MOV A,D     ;GET CODE IN ACCUMULATOR
                ORA C       ;MASK CODE
                OUT 01H     ;PUT ON PORT-B

                MOV A,C         ;TO ROTATE MASKING BYTE
                RLC
```

```
        MOV C,A

        PUSH D      ;DELAY TO GLOW SELECTED LED
        LXI D,0015H
        CALL DELAY      ;FOR FINITE TIME
        POP D
        CPI FEH     ;ARE ALL LEDS IN SELECTED ROW
                    ;ILLUMINATED
        JNZ LOOP1;IF NOT, REPEAT

        MVI A,FFH   ;TO OFF ALL LEDS OF PORT-B
        OUT 01H    ;TO OFF ALL LEDS OF PORT-B


        INX H ;INCREMENT TO POINT NEXT BYTE

        MOV A,B     ;GET ROW AND ROTATE (NEXT ROW)
        RLC
        MOV B,A
        CPI 01H     ;WHETHER ALL 7 ROWS HAVE BEEN
                    ;ACTIVATED OR NOT?
        JNZ LOOP2


        LXI H,POINTER+7 ;POINTER WHICH USES PORT-C
LOOP21: MOV D,M     ;SAVE DATA IN D

        MVI A,FFH   ;TO OFF ALL LEDS OF PORT-C
        OUT 02H

        MOV A,B     ;GET ROW NO.
        OUT 00H

LOOP11: MOV A,D     ;GET CODE IN ACCUMULATOR
        ORA C       ;MASK CODE
        OUT 02H     ;PUT ON PORT-C

        MOV A,C     ;TO ROTATE MASKING BYTE
        RLC
        MOV C,A
        PUSH D      ;TO STORE DE
        LXI D,0015H ;TO DELAY
        CALL DELAY
        POP D       ;GET-BACK DE
        CPI FEH     ;ARE ALL LEDS IN SELECTED ROW
                    ;ILLUMINATED.
        JNZ LOOP11

        MVI A,FFH
        OUT 02H     ;TO OFF ALL LEDS OF PORT-C
```

```
        INX H      ;INCREMENT TO POINT NEXT BYTE
        MOV A,B ;GET ROW AND ROTATE
        RLC
        MOV B,A
        CPI 01H
        JNZ LOOP21

        LXI H,ROT_COU
        DCR M
        JNZ START

        LXI H,COUNT    ;TO SHIFT RAM DATA BY 1-BIT
                                              ;IN ROW
        MVI M,07H
        LXI H,POINTER+7+7 ;POINTER OF NEXT CHAR.

        LXI D,POINTER ;POINTER OF PORT-B
        LXI B,POINTER+7       ;POINTER OF PORT-C

        SHLD TEMP_HL
LOOPB:  LHLD TEMP_HL
        PUSH B
        LDA COUNT1    ;FIVE BITS NEED TO SHIFT
        MOV B,A


        MOV A,M ;GET CODE OF NEXT CHARECTER
LOOPA:  RAR       ;GET PROPER BIT OF NEXT CHAR. IN CARRY
        DCR B
        JNZ LOOPA


        POP B
        LDAX D
        RAL                   ;GET CARRY AS LAST BIT OF PORT-B &
                              ;PUT MSB IN CARRY
        STAX D
        LDAX B
        RAL             ;GET CARRY AS LSB OF PORT-C
        STAX B

        INX B
        INX D
        INX H
        SHLD TEMP_HL

        LXI H,COUNT    ;TOTAL 7 BYTES NEED TO ROTATE
        DCR M
```

```
JNZ LOOPB

LXI H,COUNT1  ;NOW 5 TO 4, 4 TO 3 AND SO ON
DCR M
JNZ BEGIN

LXI H,COUNT2  ;TOTAL 7 CHARACTERS(DARSHAN)
                                ;NEED TO DISPLAY
DCR M
JNZ NEX_CHAR
JMP AGAIN

ORG COD_SEQ
      db 00h  ;D
      db 16h
      db 16h
      db 16h
      db 16h
      db 16h
      db 00h

      db 11h  ;A
      db 0Eh
      db 0Eh
      db 00h
      db 0Eh
      db 0Eh
      db 0Eh

      db 00h  ;R
      db 0Eh
      db 0Eh
      db 00h
      db 0Bh
      db 0Dh
      db 0Eh

      db 00h  ;S
      db 0Fh
      db 0Fh
      db 00h
      db 1Eh
      db 1Eh
      db 00h

      db 0Eh  ;H
      db 0Eh
      db 0Eh
      db 00h
      db 0Eh
```

```
db 0Eh
db 0Eh

db 11h  ;A
db 0Eh
db 0Eh
db 00h
db 0Eh
db 0Eh
db 0Eh

db 0Eh  ;N
db 0Eh
db 06h
db 0Ah
db 0Ch
db 0Eh
db 0Eh
.END
```

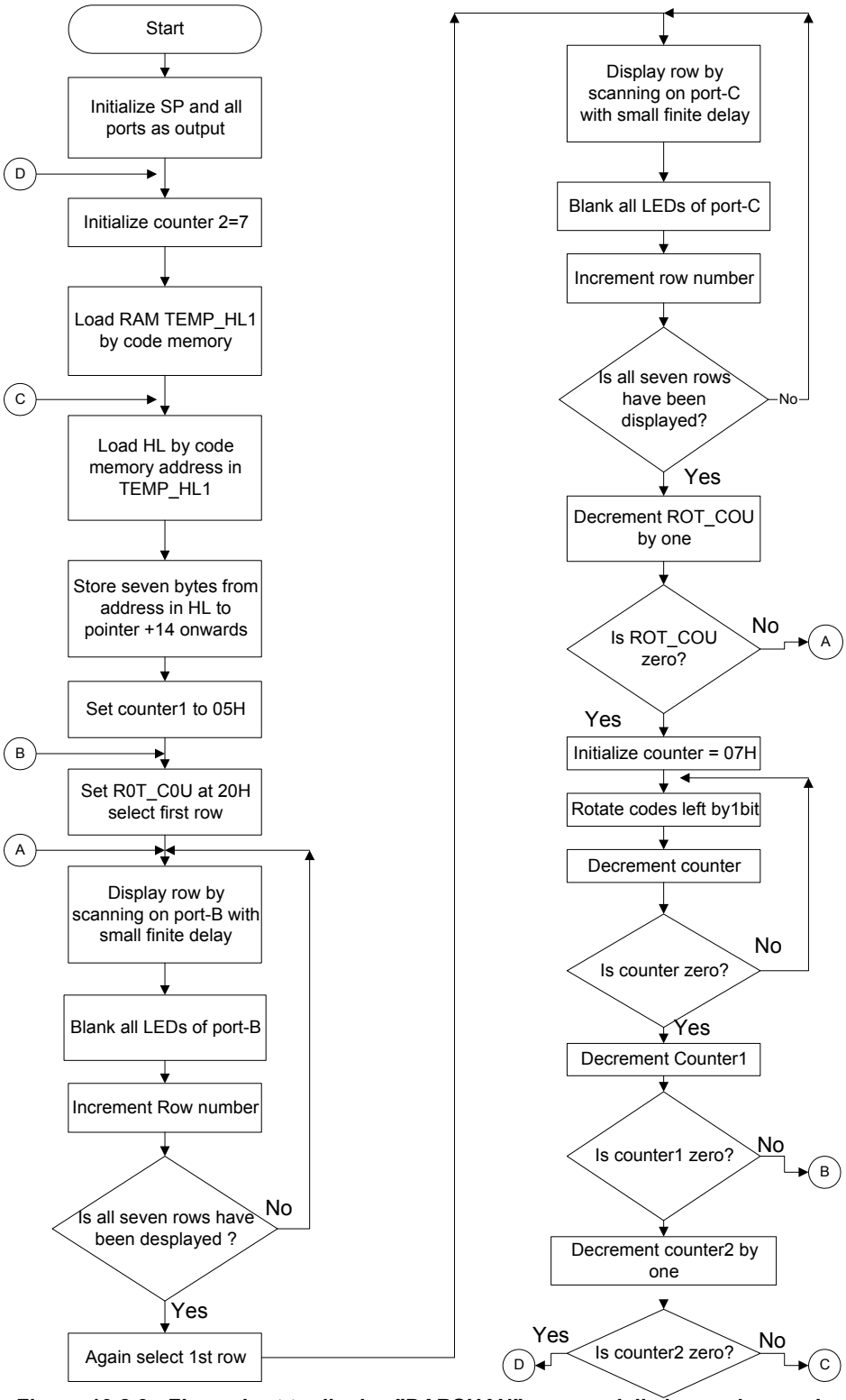


Figure-10.2.3 : FLow chart to display "DARSHAN" sequencially in running mode

10.3 EXAMLPES

Example – 1

You are given an interfacing card of “Running Character Display” with its circuit diagram and a microprocessor kit. Write program to switch ON the LED R1-C0, R1-C1.

Example – 2

You are given an interfacing card of “Running Character Display” with its circuit diagram and a microprocessor kit. Write the program to switch ON all LEDs of ROW-1.

Example – 3

You are given an interfacing card of “Running Character Display” with its circuit diagram and a microprocessor kit. Write the software program to display character-‘A’ stationary.

Example – 4

You are given an interfacing card of “Running Character Display” with its circuit diagram and a microprocessor kit. Write software program to display running of character-‘A’.

Example – 5

You are given an interfacing card of “Running Character Display” with its circuit diagram and a microprocessor kit. Write software program to display running of massage “DARSHAN”.

CHAPTER – 11 AN INTELLIGENT CONTROL PANEL

This is one more example of exemplifying the interfacing of basic trainer kit with some useful interfacing circuit. In the present interfacing module we have designed and developed an 8085A based intelligent control panel.[11] Such panels are sued in the environment where extremely high security and safety are of prime importance.

11.1. Basic circuit understanding and P.C.B. aspects

The interfacing board circuit details are shown in Figure 11.1.1. It consists of eight code lock switches, four relays, four driving transistors, eight LEDs and 40-pin FRC connector. The code lock switches are of push-button type. All eight code lock switches are connected to Port-A of PPI 8255, while indicator LEDs are connected with Port-B. Four ON-OFF relays are connected to upper four pins of Port-C (i.e. PC_4 to PC_7) through four PNP driving transistors, one for each relay. The relays need +12 Volt supply while LEDs required +5 Volt supply.

Note that +12 Volt supply is taken from external power supply.

We have earlier discussed that EPROM 2764A occupies 0000H to 1FFFH locations while RAM 6264 occupies 4000H to 5FFFH locations. In earlier section we have noted that 8255 port addresses are 00H, 01H, 02H and 03H for Port-A, Port-B, Port-C and CWR respectively.

The code lock switches are named as A, B, C, D, E, F, G and Enter. These switches get connected with indicator LEDs through software. The connections of port pins of 8255 with other circuit port is shown in Table 11.1.1.

One terminal of all the port switches are grounded while other terminals are connected to Port-A through pull-up resistors. Hence, when no key is pressed Port-A reads the data FFH. The key codes of code lock keys are shown in Table 11.1.2.

FIGURE 11.1.1 SCHAMATIC DIAGRAM OF AN ITELLIGENT CONTROL PANEL

Table-11.1.1

Port Pin	Connected to
PA0	ENTER
PA1	G
PA2	F
PA3	E
PA4	D
PA5	C
PA6	B
PA7	A
PB0	Cathode of LED D8 through 330 ohm resistor
PB1	Cathode of LED D7 through 330 ohm resistor
PB2	Cathode of LED D6 through 330 ohm resistor
PB3	Cathode of LED D5 through 330 ohm resistor
PB4	Cathode of LED D4 through 330 ohm resistor
PB5	Cathode of LED D3 through 330 ohm resistor
PB6	Cathode of LED D2 through 330 ohm resistor
PB7	Cathode of LED D1 through 330 ohm resistor
PC0	NOT CONNECTED
PC1	NOT CONNECTED
PC2	NOT CONNECTED
PC3	NOT CONNECTED
PC4	BASE OF T4 THROUGH 10K OHM RESISTOR
PC5	BASE OF T3 THROUGH 10K OHM RESISTOR
PC6	BASE OF T2 THROUGH 10K OHM RESISTOR
PC7	BASE OF T1 THROUGH 10K OHM RESISTOR

Table-11.1.2

KEY NAME	KEY CODE
A	7FH
B	BFH
C	DFH
D	EFH
E	F7H
F	FBH
G	FDH
ENTER	FEH

In the software design ON-OFF Control of all four relays is obtained by the key combinations of the code-lock keys as shown in Table 11.1.3.

Table-11.1.3

RELAY NO.	ON-OFF CODES	
	ON	**OFF**
RL1	G, G, G, ENTER	E, E, E, ENTER
RL2	F, F, F, ENTER	A, A, A, ENTER
RL3	C, C, C, ENTER	B, B, B, ENTER
RL4	E, G, F, ENTER	F, G, E, ENTER

From Table 11.1.3. we learnt that if we want to turn ON RL_3, we have to press the key “C” three times and then “Enter” key once. The software collects these four codes and compares with the stored data. If the match occurs Port-C is activated to turn-ON Relay NO-3. Note that user presses any key its corresponding LED will glow.

The PPI Chip 8255 is used in BSR and I/O modes in this system. One can generate codes of one’s own choice by proper changes in the software.

P.C.B. Aspect

The schematic drawing circuit of an intelligent control panel was used to create the P.C.B. layout. The track layout was prepared by manual facility. The track layout is shown in Figure 11.1.2. for bottom side. The overlay drawing for the same is shown in Figure 11.1.3.

FIGURE 11.1.2 BOTTOM LAYER OF AN INTELLIGENT CONTROL PANEL

FIGURE 11.1.2 OVERLAY OF AN INTELLIGENT CONTROL PANEL

11.1.1. Assembly language program for intelligent control panel

The software initializes 8255 such that Port-A works as input port and Port-B & Port-C work as output port. To indicate the activation of hardware and software the LEDs corresponding to switches B, D, F and Enter are glown through initial software port. Thereafter all relays are turned OFF as initial condition.

The software waits for the key pressing through a subroutine named Key-Chk. When valid key pressing has found, it compares the key code and stores in the RAM. A counter is used to check matching of consecutive three key depressions. If match takes place then particular routine is called to turn ON or OFF the respective relay.

The detailed assembly language program is as given bellow.

SOFTWARE

```
RESET:      .EQUAL  1C00H
COD_BUF:    .EQUAL  5F00H
COUNT:      .EQUAL  5F10H
TEMP:       .EQUAL  5F11H

        ORG RESET
START:   LXI SP,5EFFH;initialize SP
         MVI A,90H  ;initialize 8255;PA as I/P,PB and PC as O/P
         OUT 03H
         MVI A,AAH  ;turn on LEDs corresponding to B,D,F and Enter
         OUT 01H
         MVI A,00H
         STA COUNT       ;RESET counter
         STA TEMP
         MVI A,FFH       ;turn off all relays
         OUT 02H
NEXT1:   CALL KEY_CHK    ;To check key pressing
         LDA TEMP
         CPI FEH       ;TO CHECK ENTER
         JZ OUT1
         LXI H,COD_BUF
         LDA COUNT
         ADD L
         MOV L,A
         LDA TEMP
         MOV M,A

         LDA COUNT
         INR A
         STA COUNT
LOOP2:   IN 00H
         CPI FFH
         JNZ LOOP2     ;WHETHER KEY IS RELEASED OR NOT?
         MVI A,FFH
```

```
                OUT 01H
                JMP NEXT1
OUT1:           CALL PASS_CHK   ;PASSWORD CHECK
                MVI A,00H
                STA COUNT
                JMP NEXT1

PASS_CHK:   LXI H,COD_BUF
            CALL CHK_1ON

            LXI H,COD_BUF
            CALL CHK_1OFF

            LXI H,COD_BUF
            CALL CHK_2ON

            LXI H,COD_BUF
            CALL CHK_2OFF

            LXI H,COD_BUF
            CALL CHK_3ON

            LXI H,COD_BUF
            CALL CHK_3OFF

            LXI H,COD_BUF
            CALL CHK_4ON
            LXI H,COD_BUF
            CALL CHK_4OFF
            RET

CHK_1ON:    MOV A,M
            CPI FDH         ;1ST DIGIT
            RNZ
            INX H
            MOV A,M
            CPI FDH         ;2ND DIGIT
            RNZ
            INX H
            MOV A,M
            CPI FDH         ;3RD DIGIT
            RNZ
            MVI A,0EH       ;BSR CW TO RESET PC7 SO RLY1=ON
            OUT 03H
            RET

CHK_1OFF:   MOV A,M
            CPI F7H         ;1ST DIGIT
            RNZ
            INX H
```

```
                MOV A,M
                CPI F7H         ;2ND DIGIT
                RNZ
                INX H
                MOV A,M
                CPI F7H         ;3RD DIGIT
                RNZ
                MVI A,0FH       ;BSR CW TO SET PC7 SO RLY1=OFF
                OUT 03H
                RET


CHK_2ON:        MOV A,M
                CPI FBH         ;1ST DIGIT
                RNZ
                INX H
                MOV A,M
                CPI FBH         ;2ND DIGIT
                RNZ
                INX H
                MOV A,M
                CPI FBH         ;3RD DIGIT
                RNZ
                MVI A,0CH       ;BSR CW TO RESET PC6 SO RLY2=ON
                OUT 03H
                RET

CHK_2OFF:       MOV A,M
                CPI 7FH         ;1ST DIGIT
                RNZ
                INX H
                MOV A,M
                CPI 7FH         ;2ND DIGIT
                RNZ
                INX H
                MOV A,M
                CPI 7FH         ;3RD DIGIT
                RNZ
                MVI A,0DH       ;BSR CW TO SET PC6 SO RLY2=OFF
                OUT 03H
                RET

CHK_3ON:        MOV A,M
                CPI DFH         ;1ST DIGIT
                RNZ
                INX H
                MOV A,M
                CPI DFH         ;2ND DIGIT
                RNZ
                INX H
```

```
                MOV A,M
                CPI DFH         ;3RD DIGIT
                RNZ
                MVI A,0AH       ;BSR CW TO RESET PC5 SO RLY3=ON
                OUT 03H
                RET

CHK_3OFF:       MOV A,M
                CPI BFH         ;1ST DIGIT
                RNZ
                INX H
                MOV A,M
                CPI BFH         ;2ND DIGIT
                RNZ
                INX H
                MOV A,M
                CPI BFH         ;3RD DIGIT
                RNZ
                MVI A,0BH       ;BSR CW TO SET PC5 SO RLY3=OFF
                OUT 03H
                RET

CHK_4ON:        MOV A,M
                CPI F7H         ;1ST DIGIT
                RNZ
                INX H
                MOV A,M
                CPI FDH         ;2ND DIGIT
                RNZ
                INX H
                MOV A,M
                CPI FBH         ;3RD DIGIT
                RNZ
                MVI A,08H       ;BSR CW TO RESET PC4 SO RLY4=ON
                OUT 03H
                RET

CHK_4OFF:       MOV A,M
                CPI FBH         ;1ST DIGIT
                RNZ
                INX H
                MOV A,M
                CPI FDH         ;2ND DIGIT
                RNZ
                INX H
                MOV A,M
                CPI F7H         ;3RD DIGIT
                RNZ
                MVI A,09H       ;BSR CW TO SET PC4 SO RLY4=OFF
                OUT 03H
```

```
                RET

KEY_CHK:    IN 00H
            CPI FFH
            JZ KEY_CHK
            STA TEMP

            MVI B,FFH
LOOP1:      DCR B
            JNZ LOOP1

            IN 00H
            CPI FFH
            JZ KEY_CHK

            MOV B,A       ;STORE IN B TO COMP. WITH
                                ;TEMP MEMORY
            LDA TEMP
            CMP B
            JNZ KEY_CHK
            OUT 01H       ;TO ILLUMINATE LED.
            RET
            .END
```

11.2 Examples.

Example – 1

You are given an 8085 based microprocessor trainer kit and an interfacing card of an intelligent control panel with its circuit diagram. Write software program to switch ON the corresponding LED till the switch remain pressed.

Example – 2

You are given an 8085 based microprocessor trainer kit and an interfacing card of an intelligent control panel with its circuit diagram. Write the software program to switch ON relay – 1 by pressing switch B.

Example – 3

You are given an 8085 based microprocessor trainer kit and an interfacing card of an intelligent control panel with its circuit diagram. Write the software program to switch ON RL1, RL2, RL3 & RL4 by pressing switches A, B, C and D respectively, and to switch – OFF by E, F, G and Enter respectively.

Example – 4

You are given an 8085 based microprocessor trainer kit and an interfacing card of an intelligent control panel with its circuit diagram. Write a software program to set passwords as per your choice.

CHAPTER – 12 ANALOG VOLTAGE MEASUREMENT WITHOUT USING ADC

Introduction:

On this interfacing module we have demonstrated the use of VFC chip to measure analog voltage. We have avoided the use of ADC chips.

12.1. VFC Chip

LM 331 is a famous V to F converter chip.[8] It is useful for analog to digital conversion, precision frequency to voltage conversion, long term integration, linear frequency modulation and demodulation and many other functions. The output, when used as a voltage to frequency converter is a pulse train at a frequency precisely proportional to the applied input voltage. It is ideally suited for use in digital systems at low power supply voltages. Hence, it is suitable for microprocessor based system. It utilizes a new temperature compensated band gap reference circuit to provide excellent accuracy over the full operating temperature range, at power supplies as low as 4Volt. The precision timer circuit has low bias currents without degrading the quick response necessary for 100KHz voltage to frequency conversion.

The output is capable of driving three TTL Loads, or a high voltage output up to 40V, yet is short circuit proof against V_{CC}. The features of LM 331 are as follows.

Features

- Guaranteed linearity 0.01% max.
- Improved performance in existing voltage-to-frequency conversion applications.
- Split or single supply operation.
- Operates on single 5V supply.
- Pulse output compatible with all logic forms.
- Excellent temperature stability, + 50 ppm/°C max.
- Low power dissipation, 15 mW typical at 5V.
- Wide dynamic range, 100 dB min at 10 kHz full scale frequency.
- Wide range of full scale frequency, 1 Hz to 100 kHz.
- Low cost.

Simplified Block diagram

The LM 331 is a monolithic circuit designed for accuracy and versatile operation when applied as a voltage-to-frequency (V-to-F) converter or as a frequency-to-voltage (F-to-V) converter. A simplified block diagram of the LM 331 is shown in Figure 12.1.1 and consists of a switched current source, input comparator, and one-shot timer.

The operation of these blocks is best understood by going through the operating cycle of the basic V-to-F converter, Figure 12.1.1, which consists of the simplified block diagram of the LM 331 and the various resistors and capacitors connected to it.

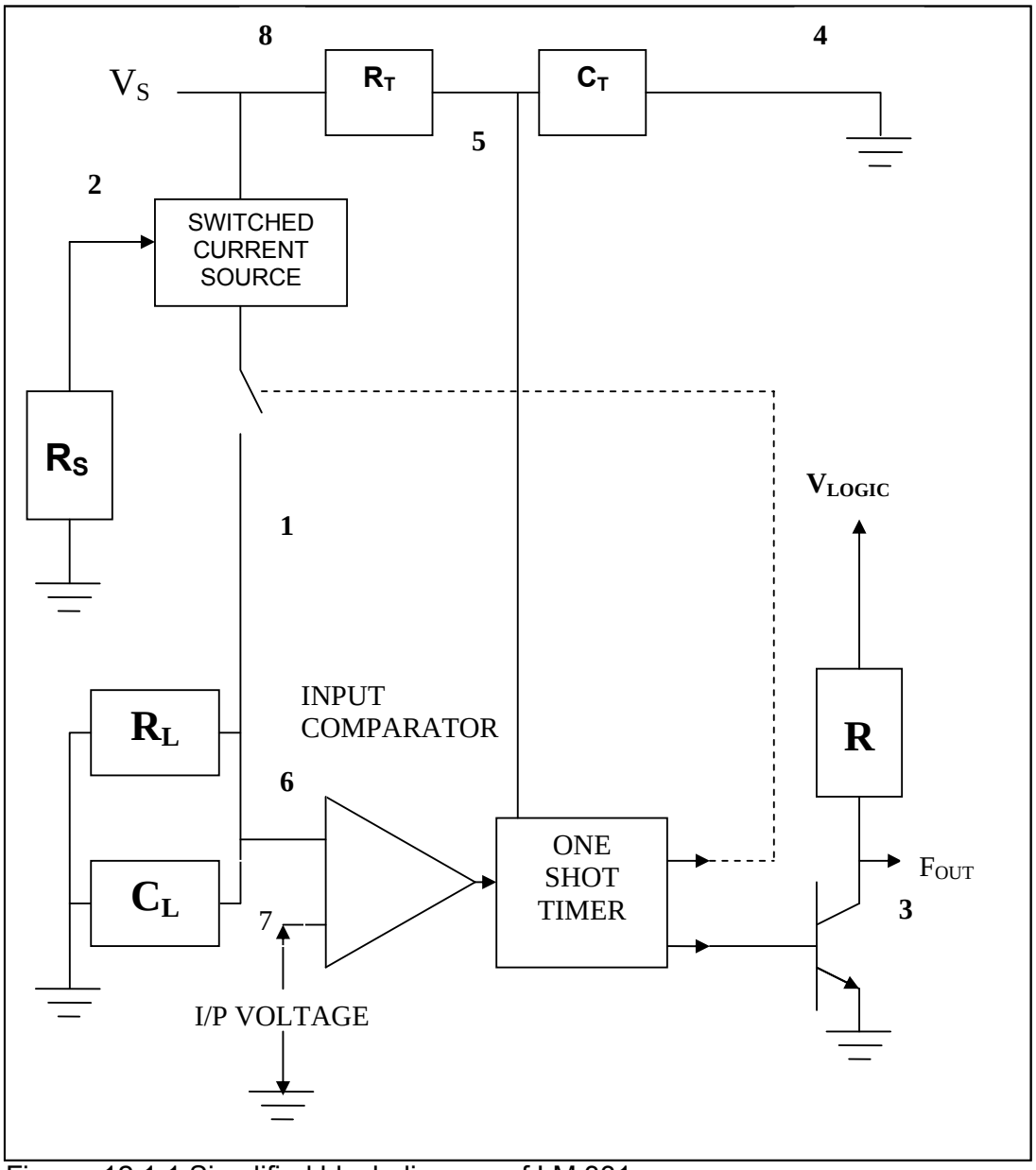


Figure: 12.1.1 Simplified block diagram of LM 331

The voltage comparator compares a positive input voltage V1, at pin 7 with respect to the voltage, V_X, at pin 6. If V1 is greater, the comparator will trigger the one-shot timer. The output of the timer will turn ON both the frequency output transistor and the switched current source for a period t=1. 1 R_1C_1. During this period, the current I will flow out of the switched current source and provide a fixed amount of charge. Q=I x t. into the capacitor, C_L. This will normally charge V_X up to a higher level than V1. At the end of the timing period, the current I will turn OFF, and the timer will reset itself.

Now there is no current flowing from pin 1, and the capacitor C_L will be gradually discharged by R_L until V_X falls to the level of V1. Then the comparator will trigger the timer and start another cycle.

The current flowing into C_L is exactly I_{AVE}=I x (1.1 x R_1C_1) x f, and the current flowing out of C_L is exactly V_X/R_L or V_{IN}/R_L. If V_{IN} is doubled, the frequency will double to maintain this balance. Even a simple V-to-F converter can provide a frequency precisely proportional to its input voltage over a wide range of frequencies.

12.2 Basic interfacing circuit with P.C.B. aspect

12.2.1 Introduction

Using LM 331 we have designed a circuit and tested the same for converting analog voltages into digital form.[12]

In the present work I.C. LM 331 is used as voltage to frequency converter. The output of this chip is connected with port pin PA0 of PPI I.C. 8255, which is an integral part of a microprocessor kit. The ON-time of the frequency at PA0 is measured using a software counter.

The accumulated counts for specific input voltage are displayed, which should be noted for each suitable input voltage values as observations. Input voltage (Vin) versus counts are plotted, which will give a non-linear graph. This graph is segmented into consecutive linear pieces. For each linear segment corresponding slope values and their corresponding constants are calculated and stored as data in two different look-up tables respectively.

Now for any random value of input voltage within the range 0v to 3v, its corresponding count will fall in any one linear segment.

The software will convert this corresponding data(which is in hexadecimal) to decimal value and display as actual input voltage.

12.2.2 Details of the circuit used

The circuit is as shown in Figure – 12.2.2.1.The basic device used is V to F converter I.C. LM 331. The passive components used are considered as per data book of National Semiconductor. The gain resistor R_s is taken 10K ohm as fixed resistor. The offset voltage adjustment is not used. The input voltage range is taken from 0v to 3v.

The supply voltage +5v is taken from the microprocessor kit through FRC interfacing cable. The output of LM 331 is connected to PA0 pin of 8255 of microprocessor kit through the same interfacing cable.

12.2.3 P.C.B. aspect

The P.C.B. making was done in three steps.
STEP-1 : Schematic preparation
STEP-2 : Track layout designing
STEP-3 : P.C.B. manufacturing

The schematic was prepared using computer software which is shown as Figure 12.2.2.1 , the track layout (Bottom layer)was done manually and is shown in Figure 12.2.3.1. The overlay designed is reproduced in Figure 12.2.3.2.

Figure:12.2.2.1 Schematic diagram of VFC card

12.2.4 Data collection and Analysis

Initially the input voltage (vin) is applied in steps of 0.25v from 0v to 3v. Each voltage input displayed the output frequency waveform on the oscilloscope whose ON-time and OFF-time were measured and are shown in Table-12.2.4.1. From Table-12.2.4.1 the graphs of Vin versus F and Vin versus T exhibit the linear and non-linear behavior respectively and are shown in Figure–12.2.4.1 and Figure-12.2.4.2.

One can also learn from Table-12.2.4.1 that the OFF-time of the frequency almost remains constant. So in the microprocessor interface software only ON-time is considered.

As the ON-time of the frequency represents the equivalent input voltage, the same is measured by using the microprocessor kit. For this special software is executed, which constantly checks the high state of pin PA0 or in other words the ON-time of the frequency.

Till the PA0 pin is high the counts in the counter of the software is kept on increasing. The moment PA0 becomes low the counts of the counter is displayed which a user is supposed to note as an observation. Thus, different values of counts for different input voltages are observed as shown in Table-12.2.4.2.

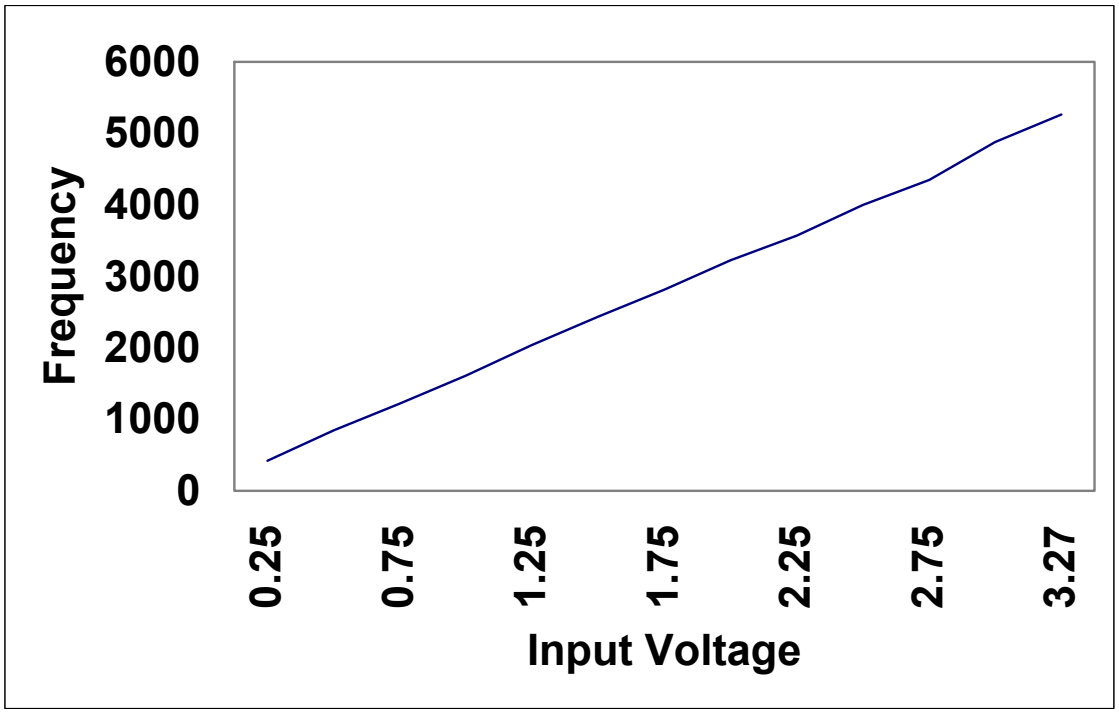
FIGURE-12.2.3.1 BOTTOM LAYER OF VFC

FIGURE-12.2.3.1 OVERLAY LAYER OF VFC

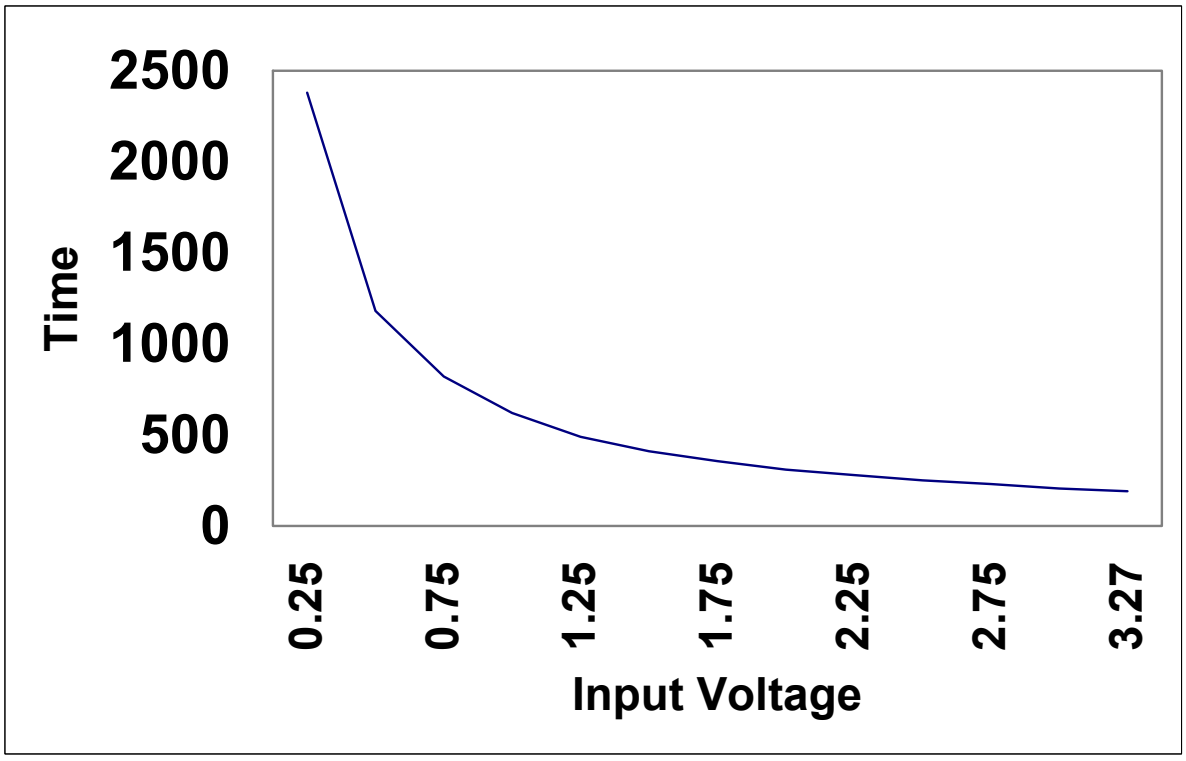
The graph of Vin versus count as shown in Figure – 12.2.4.3, exhibits non-linear behavior. It is necessary to convert its non-linear behavior such that any input voltage can be directly represented as decimal equivalent value in the display.

Table-12.2.4.1

Input Voltage Volts	ON-time μsec.	OFF-time μsec.	Total Time μsec.	Frequency Hertz
0.25	2300	80	2380	420.16
0.5	1100	80	1180	847.45
0.75	740	80	820	1219.5
1	540	80	620	1612.9
1.25	410	80	490	2040.81
1.5	330	80	410	2439.02
1.75	275	80	355	2816.9
2	230	80	310	3225.8
2.25	200	80	280	3571.42
2.5	170	80	250	4000
2.75	150	80	230	4347.82
3	125	80	205	4878.04
3.27	110	80	190	5263.15



Figure–12.2.4.1 Input voltage Vs Frequency



Figure–12.2.4.2 Input voltage Vs Time

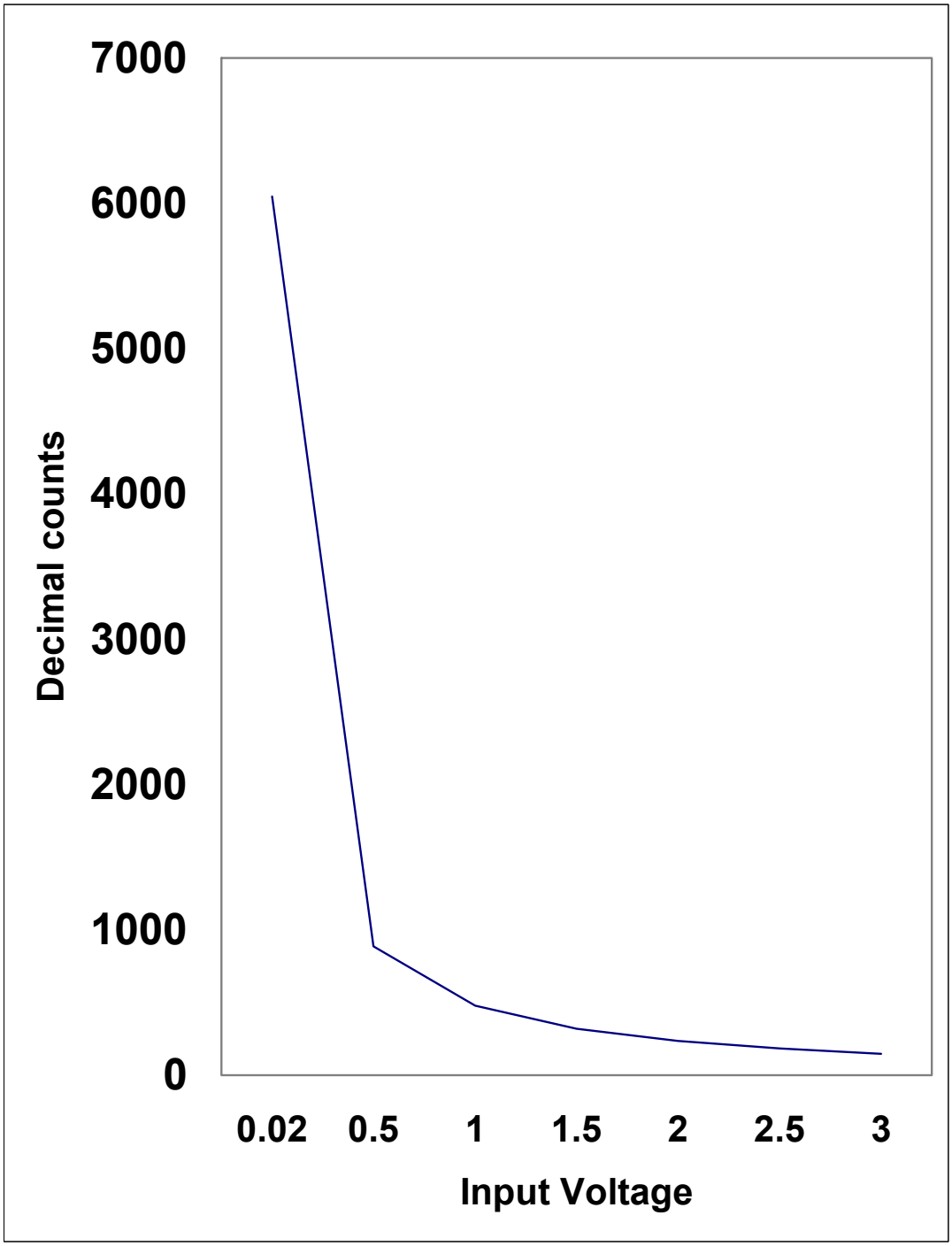


Figure – 12.2.4.3 : Input voltage Vs Counts

Table-12.2.4.2

Input Voltage Volts	Hexadecimal Counts	Decimal counts
0.02	17A0H	6048
0.5	376H	886
1	1DDH	477
1.5	140H	320
2	EBH	235
2.5	B8H	184
3	93H	147

For this the graph of Fig. – 12.2.4.3 is divided in segments as 0.02v to 0.5v,0.5v to 1.0v,1.0v to 1.5v,1.5v to 2.0v,2.0v to 2.5v and 2.5v to 3.0v. Here each segment is assumed to be linear so each segment can be represented with straight line equation

Y=mX + c
Where
Y=Input Voltage
m = slope
X=count value
c = line constant

From Table – 12.2.4.2 we can find-out the slope and constant of all segments. One representative calculation for 1st segment is shown as follows.

m1 =Y2 – Y1 / X2- X1
=17120H – 4E20H / 376H – 17A0H
=- 75300 / 142A
m1 =-5CH

For X1 and Y1,

Y1 =m1X1 + C1
4E20H =(-5CH)(17A0H) + C1
4E20H + (5CH)(17A0H) =C1
C1 =8CBA0H

Table – 12.2.4.3 shows various values of slopes and constants for various segments. The values of slopes and constants are stored in two different look-up tables as per software convenience.

Table-12.2.4.3

Segment	Slope	Constant
20,000μv to 5,00,000μv	m1 = -5Ch	c1 = 8CBA0h
5,00,000μv to 10,00,000μv	m2 = -4C6h	c2 = 182664h
10,00,000μv to 15,00,000μv	m3 = -C70h	c3 = 266EF0h
15,00,000μv to 20,00,000μv	m4 = -16Fah	c4 = 339BE0h
20,00,000μv to 25,00,000μv	m5 = -264Bh	c5 = 41AB59h
25,00,000μv to 30,00,000μv	m6 = -34C9h	c6 = 4C1618h

12.2.5 Software logic and software program

The graph of Fig. – 12.2.4.3 is now considered as calibration graph i.e. if the user keeps any input voltage value its corresponding count value will fall in this graph.

The software which is represented will take this count value as input data and will check, in which segment of look up table it falls. Then software will multiply this count with the slope of that segment and the product will be subtracted from the constant(because slope is negative). This will give the number in hexadecimal which is nothing but voltage itself. This hexadecimal number is then converted into decimal number for display of the input voltage.

The software uses typical routines of the microprocessor kit, which has been indigenously developed by us.

The detailed software is as given.

SOFTWARE

```
HL_DISP:      .EQUAL      1C1CH

DATADISP:     .EQUAL      1E65H

DELAY:        .EQUAL      1F40H

COUNTER:      .EQUAL      5F40H   ;5F40H TO 5F43H

X_POINT:      .EQUAL      5F44H   ;5F44H TO 5F45H
```

```
ADD1_BFR:     .EQUAL      5F46H   ;5F46H TO 5F49H
ADD2_BFR:     .EQUAL      5F4AH   ;5F4AH TO 5F4DH
ADD_ANS:      .EQUAL      5F4EH   ;5F4EH TO 5F51H
ADD_TEMP:     .EQUAL      5F52H   ;5F52H & 5F53H
MUL1_BFR:     .EQUAL      5F54H   ;5F54H & 5F55H
MUL2_BFR:     .EQUAL      5F56H   ;5F56H & 5F57H
BCD_BFR:      .EQUAL      5F58H   ;5F58H TO 5F5CH
SEG_NO:       .EQUAL      5F5DH   ;5F5DH TO FIND SEGMENT
TEMP_BFR:     .EQUAL      5F5EH   ;5F5EH & 5F5FH
DISPDATA:     .EQUAL      5F13H
VFC:          .EQUAL      0B00H
              ORG VFC
              LXI SP,5EFFH
              MVI A,90H
              OUT 03H
START:        LXI H,COUNTER
              MVI B,04H
LOOP1:        MVI M,00H
              INX H
              DCR B
              JNZ LOOP1
              LXI H,COUNTER
              MVI B,20H
LOOP2:        IN 00H
              ANI 01H
```

```
                JZ LOOP2
                LXI D,FFFFH
AGAIN:          DCR E
                JNZ NO_ZERO
                DCR D
                JNZ NO_ZERO
                MVI B,04H       ;IT EXECUTES AT 0 VOLT ONLY
ZERO_VLT:       MVI M,00H
                INX H
                DCR B
                JNZ ZERO_VLT
                JMP DISP_FFH
NO_ZERO:        IN 00H
                ANI 01H
                JZ OUT1
                MVI A,01H
                ADD M
                MOV M,A
                JNC AGAIN
                INX H
                MVI A,01H
                ADD M
                MOV M,A
                DCX H
                JNC AGAIN
```

```
        INX H
        INX H
        MVI A,01H
        ADD M
        MOV M,A
        DCX H
        DCX H
        JNC AGAIN
        INX H
        INX H
        INX H
        INR M
        DCX H
        DCX H
        DCX H
        JMP AGAIN
OUT1:   DCR B
        JNZ LOOP2
        CALL CONT_DIV
        CALL CONT_DIV
        CALL CONT_DIV
        CALL CONT_DIV
        CALL CONT_DIV
        LXI H,COUNTER
        MOV E,M
```

```
                INX H
                MOV D,M
                LXI H,X_PARA
                SHLD X_POINT
NXT_PONT:  LHLD X_POINT
                MOV C,M
                INX H
                MOV B,M
                INX H
                SHLD X_POINT
                CALL MINUS      ;BC-DE=HL
                JC OUT2
                JMP NXT_PONT
OUT2:           LHLD X_POINT
                DCX H
                DCX H
                DCX H
                DCX H
                SHLD ADD1_BFR
                LXI H,ADD1_BFR+2
                MVI C,0AH
LOOP_A1:    MVI M,00H
                INX H
                DCR C
                JNZ LOOP_A1
```

```
LXI H,X_PARA
SHLD ADD2_BFR
CALL SUB_TION
LHLD ADD_ANS
DCX H
DCX H   ;TO AVOID FFFFH (ZER0 VOLT)
MOV A,L
STA SEG_NO
LXI H,SLOPE
ADD L
MOV L,A
MVI A,00H
ADC H
MOV H,A
MOV E,M
INX H
MOV D,M
XCHG
SHLD MUL1_BFR
LHLD COUNTER
SHLD MUL2_BFR
CALL MULTIPLY
LHLD ADD_ANS
SHLD ADD2_BFR
LHLD ADD_ANS+2
```

```
        SHLD ADD2_BFR+2
        LDA SEG_NO
        STC
        CMC
        RAL     ;TO DOUBLE THE CONTENT BECAUSE CONSTANT
IS 4-                   BYTE
        LXI H,CONSTANT
        ADD L
        MOV L,A
        MVI A,00H
        ADC H
        MOV H,A
        MOV E,M
        INX H
        MOV D,M
        SHLD TEMP_BFR
        XCHG
        SHLD ADD1_BFR
        LHLD TEMP_BFR
        INX H
        MOV E,M
        INX H
        MOV D,M
        XCHG
        SHLD ADD1_BFR+2
```

```
                CALL SUB_TION
                CALL HEX_BCD
                LHLD BCD_BFR
                CALL HL_DISP
                LHLD BCD_BFR+2
                CALL HL1_DISP
                JMP OUT3
DISP_FFH:   LHLD COUNTER
                CALL HL_DISP
                LHLD COUNTER+2
                CALL HL1_DISP
OUT3:           CALL DATADISP
                LXI D,FFFFH
                CALL DELAY
                JMP START
HL1_DISP:   LXI D,DISPDATA+4  ;FILL UP FIRST FOUR LOCATIONS
                MOV A,L         ;AND POINTED TO DISPDATA+4
                ANI 0FH
                STAX D
                INX D
                MOV A,L
                ANI F0H
                RRC
                RRC
                RRC
```

```
                RRC
                STAX D
                INX D
                MOV A,H
                ANI 0FH
                STAX D
                INX D
                MOV A,H
                ANI F0H
                RRC
                RRC
                RRC
                RRC
                STAX D
                INX D
                RET
CONT_DIV:   LXI H,COUNTER+3
                STC
                CMC
                MOV A,M
                RAR
                MOV M,A
                DCX H
                MOV A,M
                RAR
```

```
                MOV M,A
                DCX H
                MOV A,M
                RAR
                MOV M,A
                DCX H
                MOV A,M
                RAR
                MOV M,A
                RET
MINUS:          STC
                CMC
                MOV A,C
                SUB E
                MOV L,A
                MOV A,B
                SBB D
                MOV H,A
                RET
X_PARA:         DB FFH
                DB FFH
                DB A0H
                DB 17H
                DB 76H
                DB 03H
```

```
                DB DDH
                DB 01H
                DB 40H
                DB 01H
                DB EBH
                DB 00H
                DB B8H
                DB 00H
                DB 93H
                DB 00H
                DB 00H
                DB 00H
ADDITION:  LXI H,ADD_ANS  ;Z=X+Y X=ADD1_BFR Y=ADD2_BFR
Z=ADD_ANS
                MVI D,04H
AD_LOOP1:  MVI M,00H
                INX H
                DCR D
                JNZ AD_LOOP1
                LXI H,ADD2_BFR
                LXI D,ADD1_BFR
                LXI B,ADD_ANS
                STC          ;TO SET CARRY
                CMC          ;TO RESET CARRY(BY COMPLIMENT)
                PUSH PSW     ;TO STORE CARRY
```

```
                MVI A,04H       ;COUNTER TO ADD 4-BYTES
AD_LOOP2:  STA ADD_TEMP
                POP PSW
                LDAX D
                ADC M
                STAX B
                PUSH PSW
                INX H
                INX B
                INX D
                LDA ADD_TEMP
                DCR A
                JNZ AD_LOOP2
                POP PSW         ; PUSH PSW EXECUTED 1 MORE TIME
                RET
SUB_TION:   LXI H,ADD_ANS  ;Z=X-Y X=ADD1_BFR Y=ADD2_BFR
Z=ADD_ANS
                                ;SAME MEMORY IS USED FOR ADD & SUB
                MVI D,04H       ;TO STORE 00H IN ANSWER
SU_LOOP1:   MVI M,00H
                INX H
                DCR D
                JNZ SU_LOOP1
                LXI D,ADD1_BFR  ;X IN X-Y
                LXI H,ADD2_BFR  ;Y IN X-Y
```

```
                LXI B,ADD_ANS   ;TO STORE ANSWER
                STC
                CMC
                PUSH PSW
                MVI A,04H
                SU_LOOP2:           STA ADD_TEMP
                POP PSW
                LDAX D
                SBB M
                STAX B
                PUSH PSW
                INX H
                INX B
                INX D
                LDA ADD_TEMP
                DCR A
                JNZ SU_LOOP2
                POP PSW         ; PUSH PSW EXECUTED 1 MORE TIME
                RET
MULTIPLY:  LXI H,ADD1_BFR  ;MUL1_BFR X MUL2_BFR = ADD_ANS
                                ;ADDITION SUBROUTINE WILL USE
                MVI B,0CH       ;TO STORE 00H IN ALL MEMORY WHICH
                                ;ARE USED IN ADDITION SUBROUTINE
MU_LOOP1: MVI M,00H
                INX H
```

```
                DCR B
                JNZ MU_LOOP1
                LHLD MUL1_BFR   ;GET DATA1 IN HL
                SHLD ADD2_BFR   ;STORE HL(DATA1)IN ADD2_BFR
                MVI B,10H       ;COUNTER TO MULTIPLY 16-BITS
MU_LOOP2:  MVI C,04H      ;TO ROTATE LEFT 4-BYTES
                LXI H,ADD_ANS   ;STARTING ADDRESS OF BUFFER
                LDA ADD_ANS+3   ;GET MSB IN ACC.('CONT. OF C'-1)
                RAL             ;GET D7 OF MSB IN CARRY
                CALL  ROT_LEFT
                LHLD ADD_ANS
                SHLD ADD1_BFR
                LHLD ADD_ANS+2
                SHLD ADD1_BFR+2
                MVI C,02H       ;TO ROTATE LEFT 2-BYTES
                LXI H,MUL2_BFR  ;DATA2 IS GOING TO ROTATE
                LDA MUL2_BFR+1  ;GET D7 OF MSB IN CARRY
                RAL
                CALL ROT_LEFT
                PUSH B          ;TO SAVE BC,IN ADDITION ALL REGISTERS
                                ;WILL BE USED.
                CC ADDITION
                POP B
                DCR B
                JNZ MU_LOOP2
```

```
                RET
ROT_LEFT:   MOV A,M
                RAL
                MOV M,A
                INX H
                DCR C
                JNZ ROT_LEFT
                RET
HEX_BCD:    MVI B,20H
                MVI C,05H
                LXI H,BCD_BFR          ;5-BYTES
HB_LOOP1:  MVI M,00H
                INX H
                DCR C
                JNZ HB_LOOP1
HB_LOOP3:  MVI C,04H
                LXI H,ADD_ANS
                LDA ADD_ANS+3
                RAL
                CALL ROT_LEFT
                LXI H,BCD_BFR
                MVI C,05H
HB_LOOP2:   MOV A,M
                ADC M
                DAA
```

```
                MOV M,A
                INX H
                DCR C
                JNZ HB_LOOP2
                DCR B
                JNZ HB_LOOP3
                RET
SLOPE:          DB 5CH  ;2-BYTES
                DB 00H
                DB C6H
                DB 04H
                DB 70H
                DB 0CH
                DB FAH
                DB 16H
                DB 4BH
                DB 26H
                DB C9H
                DB 34H
CONSTANT: DB A0H   ;4-BYTES
                DB CBH
                DB 08H
                DB 00H
                DB 64H
                DB 26H
```

```
DB 18H

DB 00H

DB F0H

DB 6EH

DB 26H

DB 00H

DB E0H

DB 9BH

DB 33H

DB 00H

DB 59H

DB ABH

DB 41H

DB 00H

DB 18H

DB 16H

DB 4CH
DB 00H

.END
```

12.3 Examples

Example – 1

You are given a microprocessor kit and a VFC module. Vary the input voltages from 0V to 3V in steps of 0.25V and measure ON – Time and OFF – Time by using either C.R.O. or Logic Analyzer.

Example – 2

You are given a microprocessor kit and a VFC module. Vary the input voltages from 0V to 3V in steps of 0.5V and display the counts of ON – Time on the display of microprocessor kit.

Example – 3

You are given a microprocessor kit and a VFC module. Vary the input voltages from 0V to 3V in steps of 0.5V and note down the counts of ON – Time as observations and find the slope and constants for different segments of 0.5V and calculate voltage for some value of count.

CHAPTER-13 LOGIC-CONTROLLER

Solving Boolean equations using microprocessor kit is one of the popular way of understanding microprocessor capabilities. In the present paper one interfacing circuit is designed, constructed and tested for such purpose. This interfacing module accepts four variable inputs at a time and can display output of eight variables, but with proper software any number of input variables can be assigned as well as any number of output variables can be displayed. The actual Boolean equations are solved by 8085 based software which user can develop him self.

13.1 Understanding of basic circuit and P.C.B. aspects

13.1.1 : Introduction

The interfacing module is designed using PPI 8255 ,74LS245 bi-directional buffer, five push button switches and eight LEDs. The inputs are given through four switches S1, S2, S3 and S4. Pressing these switches each time toggles the input logic state. Switch S5 allows the input to be loaded into the processor. These switches are interfaced with port-A of 8255. The port-C of 8255 feeds the output to the LEDs through buffer chip 74LS245.

In the present module user can input any number of inputs through four switches S1, S2, S3 and S4. For example, if user wants twelve variables to input, he has to do so in three steps. First, set the switches S1, S2, S3 and S4 and then press enter(S5) switch. The user software now should wait for second set of data. Now as second set, user should set switches S1, S2, S3 and S4 as per his variables and then press enter(S5).

Similarly, third set of input variables is to be entered. Same is true for displaying the output.

13.1.2 : Circuit and Discussion with P.C.B. aspects

The detailed interfacing circuit is shown in Figure-13.1.2.1. The 8255 interfaces the microprocessor system (kit) with the logic control circuit. At the CPU side the RD, WR, RESET and data bus pins of 8255 are directly connected with the corresponding pins of 8085. The port-C of 8255 is connected with the input pins of buffer chip 74LS245. The output of this buffer chip is connected with the LEDs through current limiting resistors. The switches are directly connected with the port-A of 8255. The same information is shown in Table-13.1.2.1 and Table-13.1.2.2.

Here 40-pin FRC connector interfaces the microprocessor kit and the module. The P.C.B. layout of the circuit shown in Fig.-13.1.2.1 is shown in Fig.-13.1.2.2. (bottom layer) and Fig.-13.1.2.3(overlay). In order to understand the function of the module, consider the following illustration.

Table-13.1.2.1 :Connection details of I/O LEDs and Port C of 8255

LED	Port pin
OUT3	PC0
OUT2	PC1
OUT1	PC2
OUT0	PC3
IN3	PC4
IN2	PC5
IN1	PC6
IN0	PC7

Table-13.1.2.2 : Connection details of switches with Port A of 8255

Switch No.	Port pin
S1	PA3
S2	PA2
S3	PA1
S4	PA0
S5 (ENTER)	PA4

Suppose we want to solve the following Boolean equations.

$Y0 = A0 + B0$

$Y1 = A1 \bullet B1$

$Y2 = A2 \oplus B2$

$Y3 = \overline{A3}$

In above Boolean equations we have seven input variables as A0, B0, A1, B1, A2, B2 AND A3. Suppose we want all the inputs to be '1'. To enter them into the module, press switches S1, S2, S3 and S4 such that they all represent logic '1', this situation is displayed on the input LEDs by glowing.

Now press enter switch S5. This process will transfer the high states of input variables A0, A1, A2 and A3 to the program. Now again rearrange S1, S2 and S3 such that they represent input variables B0, B1 and B2 to their high state and then press enter switch S5.

Now user should develop a program which will perform above logical function and displaying the result on LEDs. For guiding purpose a representative flow-chart of above Boolean equations is given in Fig.-13.1.2.4. Note that it is for the above equations only. One can take any Boolean equations and develop a software. In flow-chart, the rectangular box containing Boolean equations of the above program needs to be modified for other sets of equation.

Note that one has to use port addresses and memory location addresses as per one's microprocessor kit.

FIGURE-13.1.2.1 : SCHEMATIC CIRCUIT OF LOGIC-CONTROLLER

FIGURE-13.1.2.2 : BOTTOM LAYER OF LOGIC CONTROLLER

FIGURE-13.1.2.3 : OVERLAY OF LOGIC CONTROLLER

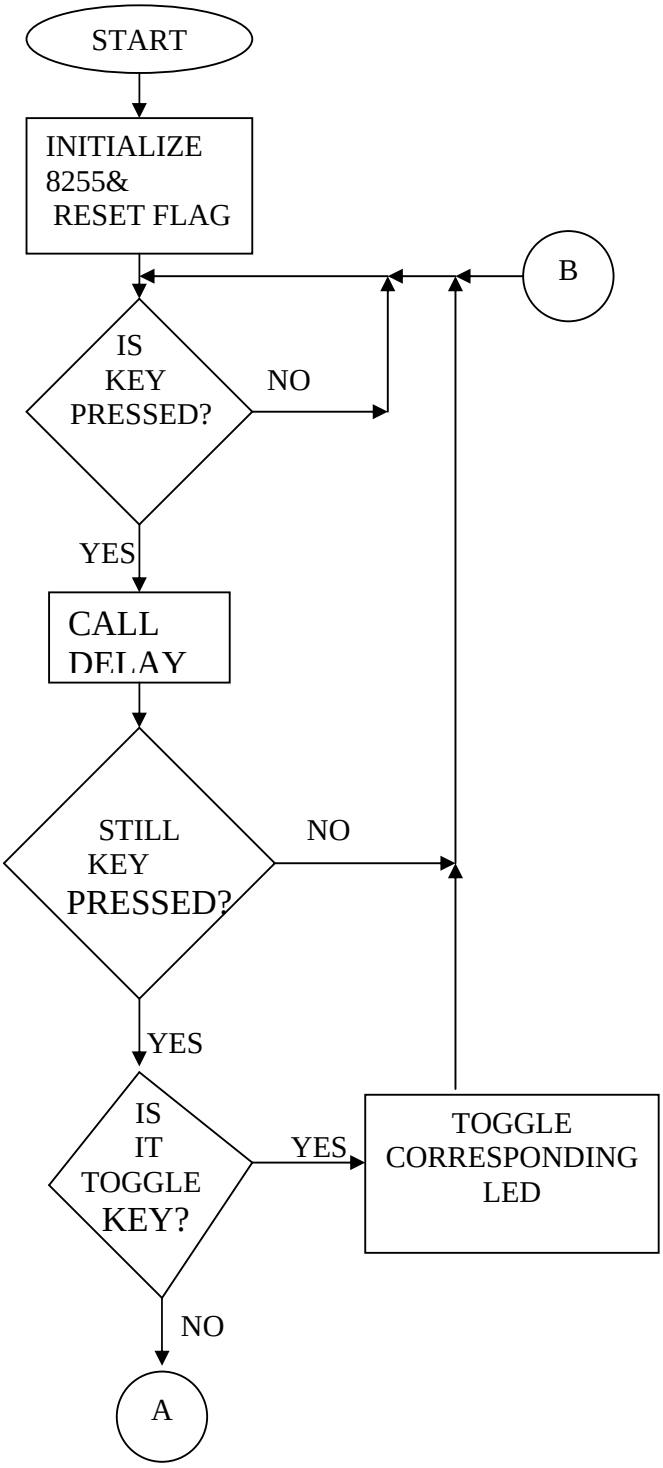


FIGURE- 13.1.2.4 (A) FLOW-CHART OF SOLUTION

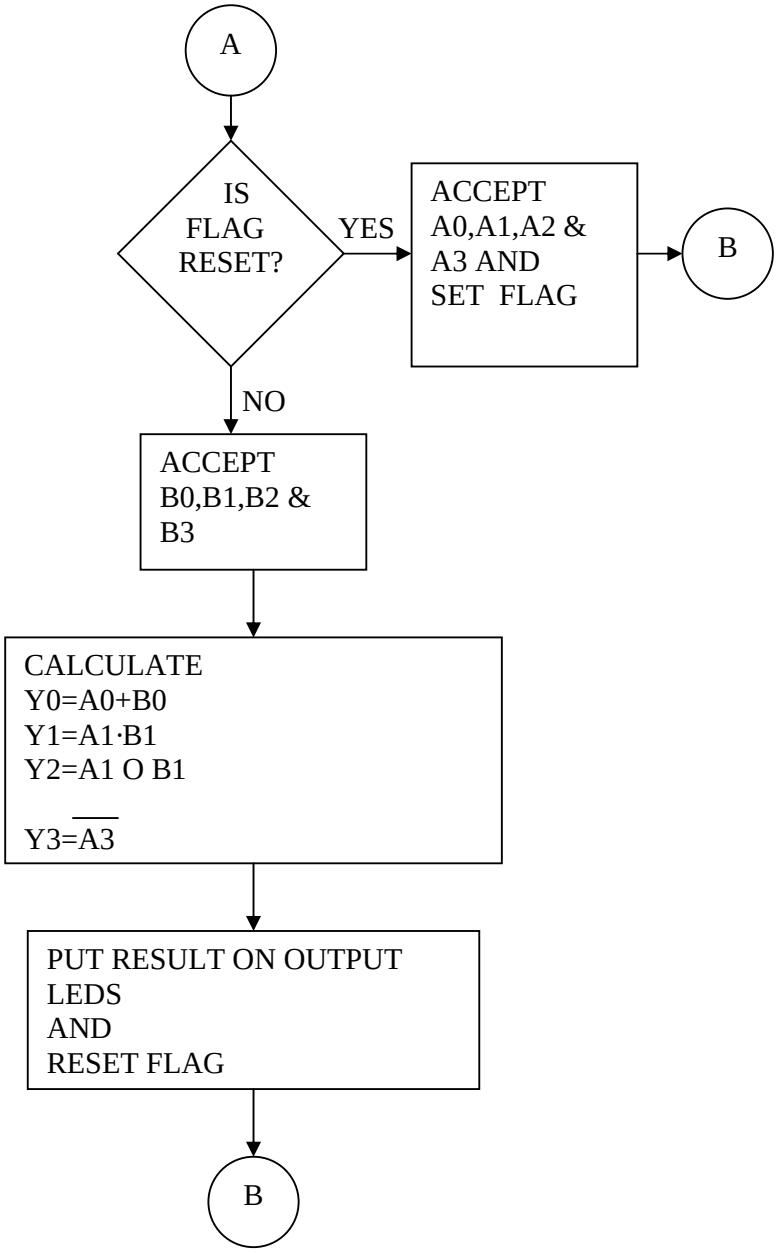


FIGURE- 13.1.2.4 (B) FLOW-CHART OF SOLUTION

13.2 Usage of logic controller card

The present interfacing module is useful in different ways. It helps to understand the capability of microprocessor to solve various logical problems. Apart from this the circuit design of this module explains how one can use the input output functions of a typical microprocessor based system. This is clarified by switches S1 to S4, which are interfaced as input. Similarly, LED's interfacing helps to understand the output device connecting concepts.

13.3 Programming aspects of logic controller

The general concepts of programming are understood with the help of flow-chart illustrated in Figure-13.1.2.4.

Initially the flag is reset and the 8255 is initialized, then program waits for the key depression. After receiving a valid key, the program checks, whether it is toggle key or enter key, if it is toggle key, program toggles the corresponding LED by changing the data. And if it is enter key the program checks for flag. If flag is reset A0,A1,A2 and A3 are accepted and the flag is set , otherwise it would accept the data as B0,B1,B2 and B3.

Then the Boolean equation is solved by the program and the result is put on output LEDs. The flag is reset again to repeat the process.

The software program is as given below.

SOFTWARE

```
DATA1:  .EQUAL  5F40H
DATA2:  .EQUAL  5F41H
PORTC:  .EQUAL  5F42H
FLAG:   .EQUAL  5F43H
DELAY:  .EQUAL  1F40H
LC:     .EQUAL  1100H

ORG   LC
        LXI SP,5EFFH    ;Initialize stack pointer
        LXI H,DATA1     :Initialize register pair HL
        MVI A,90H       :Initialize register A by 90h

        OUT 03H         ;send this 90h at 03h
        MVI A,00H       ;TO RESET ALL BITS OF PORT C
        OUT 02H         ;send this 00h at 02h
        MVI A,00H       ;FLAG
        STA FLAG        ;store 00h in flag
        STA PORTC           ;store 00h in portc
LOOP1:  IN 00H          ;read porta
        ANI 1FH         ;mask by 1fh
        CPI 1FH         ;compare with 1fh
        JZ LOOP1        ;if zero flag is set so go to loop1
```

```
        LXI D,FFFFH             ;Initialize register pair by ffffh
        CALL DELAY      ;call delay subroutine

        IN 00H      ;read porta
        ANI 1FH     ;mask by 1fh
        CPI 1FH     ;compare with 1fh
        JZ LOOP1    ;if zero flag is set so go to loop1
        MVI C,04H       ;COUNTER
        MVI B,10H       ;TO SEND DATA ON PC4
LOOP2:  RAR                 ;rotate right through carry
        JNC SW_PRESSED      ;if carry is reset so jump to its
                            ;subroutine
        MOV D,A         ;to store [A] temporarily
        CMC             ;compliment carry
        MOV A,B         ;to manipulate port c data
        RAL             ;rotate left through carry
        MOV B,A         ;get back after rotation
        MOV A,D         ;retrieve stored data
        DCR C           ;decrement counter
        JNZ LOOP2       ;if counter is zero so go out of the
                        ;loop
        LDA FLAG        ;get flag into accumulator
        RAL             ; TO CHECK WHETHER DATA1 OR
                        ;DATA2
        JC OUT1         ;if flag is set so go to out1
        LDA PORTC       ;get entered variables
        ANI F0H      ;NOW WE NEED ONLY A,B,C & D
        RRC         ; four time rotate to interchange nibbles
        RRC
        RRC
        RRC

        MOV M,A      ;H&L HAVE ADDRESS OF DATA1
        LXI H,DATA2    ;NOW 2ND DATA WILL IN DATA2
        MVI A,80H
        STA FLAG            ;SET FLAG FOR 2ND DATA
        MVI A,00H
        STA PORTC               ;put 00h in portc
        OUT 02H       ;ALL LEDS ARE OFF AFTER ACCEPTING
                            ;DATA
        JMP LOOP1               ;go back to loop1
OUT1:   LDA PORTC               ;get entered data
        ANI F0H       ;NOW WE NEED ONLY A,B,C & D NOT FLAG
        RRC         ; four time rotate to interchange nibbles
        RRC
        RRC
        RRC
        MOV M,A         ;to store [A] temporary
        LXI H,PORTC     ;pointer to portc
        MVI M,00H      ;ANS WILL BE IN PORTC
```

```
                LDA DATA2               ;get second set of data
                MOV C,A         ;to store [A] into [C]
                LDA DATA1               ;get first set of data
                MOV B,A         ;to store [A] into [B]
                ORA C       ;TO CALCULATE A0+B0
                ANI 01H      ;Y0=A0+B0
                ORA M     ;to put Y0
                MOV M,A       ;STORE ANS IN PORTC
                MOV A,B   ;get first data
                ANA C        ;TO CALCULATE A1*B1
                ANI 02H      ;Y1=A1*B1
                ORA M     ;to put Y1
                MOV M,A       ;STORE ANS IN PORTC
                MOV A,B       ;get first data
                XRA C        ;TO CALCULATE A2 EX-OR B2
                ANI 04H      ;Y2=A2 EX-OR B2
                ORA M     ;to put Y2
                MOV M,A   ;STORE ANS IN PORTC
                MOV A,B   ;get first data
                CMA       ;Y3=compliment of A3
                ANI 08H      ;TO MASK Y3 ONLY
                ORA M        ;TO GET ALL 4 BITS
                OUT 02H      ;send answer
                MVI A,00H    ;FOR NEXT DATA1 2ND CYCLE
                STA FLAG        ;reset flag
                LXI H,DATA1     ;to get first data in second cycle
                JMP LOOP1               ;jump to loop1
SW_PRESSED:
               LDA PORTC        ;get all bits
                XRA B           ;ex-or with 1 to toggle
                STA PORTC               ;store result
                OUT 02H         ;send result to 02h
                JMP LOOP1               ;jump to loop1
                .END
```

13.4 Examples

Example – 1

You are given a logic controller module and 8085 based microprocessor kit. Write software to ON – LED by pressing its corresponding key.

Example – 2

You are given a logic controller module and 8085 based microprocessor kit. Write software program to toggle LED by each press of its corresponding switch.

Example – 3

You are given a logic controller module and 8085 based microprocessor kit. Write software to solve following Boolean equation and develop program to get output.

$$Y_0 = \overline{A_0}$$

$$Y_1 = \overline{A_1}$$

$$Y_2 = \overline{A_2}$$

$$Y_3 = \overline{A_3}$$

Example – 4

You are given a logic controller module and 8085 based microprocessor kit. Write software to solve following Boolean equations and develop program to get output.

$$Y_0 = A_0 + B_0$$

$$Y_1 = \overline{A_1 \bullet B_1}$$

$$Y_2 = \overline{A_2 + B_2}$$

$$Y_3 = A_3$$

CHAPTER – 14 INTERFACING TIMER 8253

To generate accurate time delays and different types of rectangular waveforms this Timer IC 8253 is used. The user can vary the delay by changing software. This IC 8253 can work in six different modes, and it is having three independent counters. The counters are able to count in binary or in BCD.[6]

14.1 Basics of 8253.

This programmable Interval Timer/counter 8253 is designed by Intel corporation. This 8253 is compatible with 8085 microprocessor. Each 16-bit counter can work with a count rate up to 2MHz. All modes of operation are software or hardware programmable.

Because of this facility in IC 8253, the microprocessor becomes free from loops in system software. This facility allows the CPU to carry out other tasks in the mean time.

Figure 14.1.1 and 14.1.2 show the functional block diagram and the pin diagram of 8253 respectively.

Data bus Buffer:

To interface 8253 data bus with system bus this tristate bi-directional 8-bit buffer is used. The direction of data buffer is decided by read / write control signals.

When write signal is activated, this buffer receives data from system data bus. When read is activated, this buffer transmits data to system data bus.

Read/Write Logic:

This Read/Write Logic block has five input signals, these signals are Read, Write, Chips select, A_1 and A_0.

This block accepts input from system control bus and address bus. A_1 and A_0 are activated specific part of 8253 and Read or Write decides whether data is to be read or written respectively.

Control Word Register:

This register of 8253 gets selected when A_0=1 and A_1=1. It is used to write command word, which specifies the counter to be used, its mode of operation and the data transfer to be used i.e. read or write the data bytes (LSB, MSB or LSB & MSB). The data cannot be read from control register.

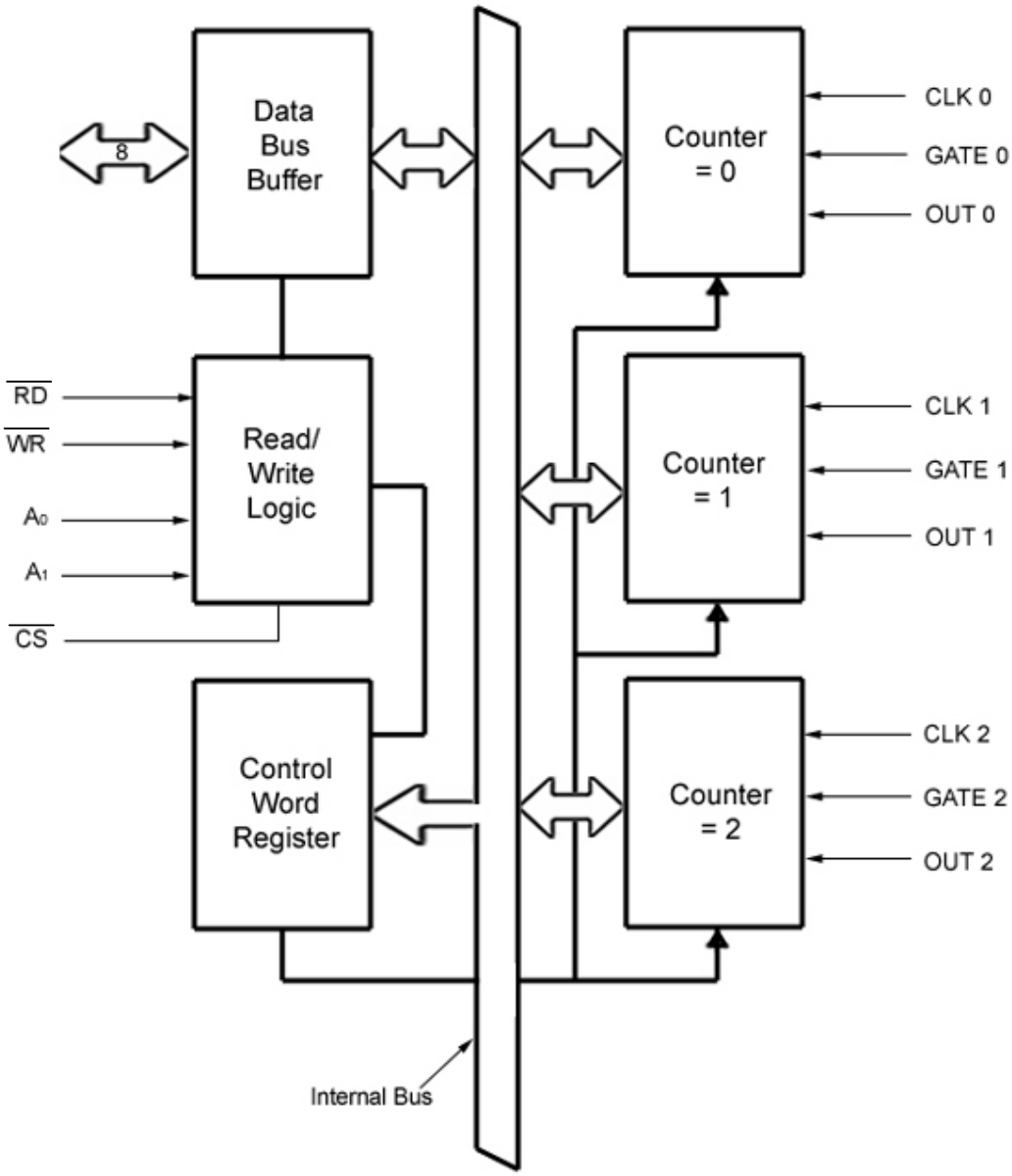


Figure – 14.1.1 Functional block diagram of 8253.

Table-14.1.1 Counter/Control Register Selection

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A1	A0	SELECTION
0	1	0	0	0	LOAD COUNTER NO. 0
0	1	0	0	1	LOAD COUNTER NO. 1
0	1	0	1	0	LOAD COUNTER NO. 2
0	1	0	1	1	WRITE MODE WORD
0	0	1	0	0	READ COUNTER NO. 0
0	0	1	0	1	READ COUNTER NO. 1
0	0	1	1	0	READ COUNTER NO. 2
0	0	1	1	1	NO-OPERATION (TRI-STATE)
1	X	X	X	X	DISABLE(TRI-STATE)
0	1	1	X	X	NO-OPERATION(TRI-STATE)

Counter – 0, 1 and 2:

Here these all are independent 16-bit down counters. These counters can be programmed separately. Each counter is having three pins Clock, Gate and Output.

The loaded value in the counter will be decremented by counter at each clock input pulse.

Pin Description

D_0-D_7 Data bus:

To transfer data between 8085 and 8253 these 8-bit data bus is used. These 8-bit data bus should be connected with 8-bit system data bus of 8085.

$\overline{CS}$: Chip Select:

To select 8253 IC this pin should be low. If this pin is high, 8253 will be in de-active state.

$\overline{RD}$: Read:

This is an active low input signal. This signal is used in co-ordination with other signals to transfer data into 8085 from 8253.

$\overline{WR}$: Write:

This is an active low input signal. This signal is used in co-ordination with other signals to transfer data into 8253 from 8085.

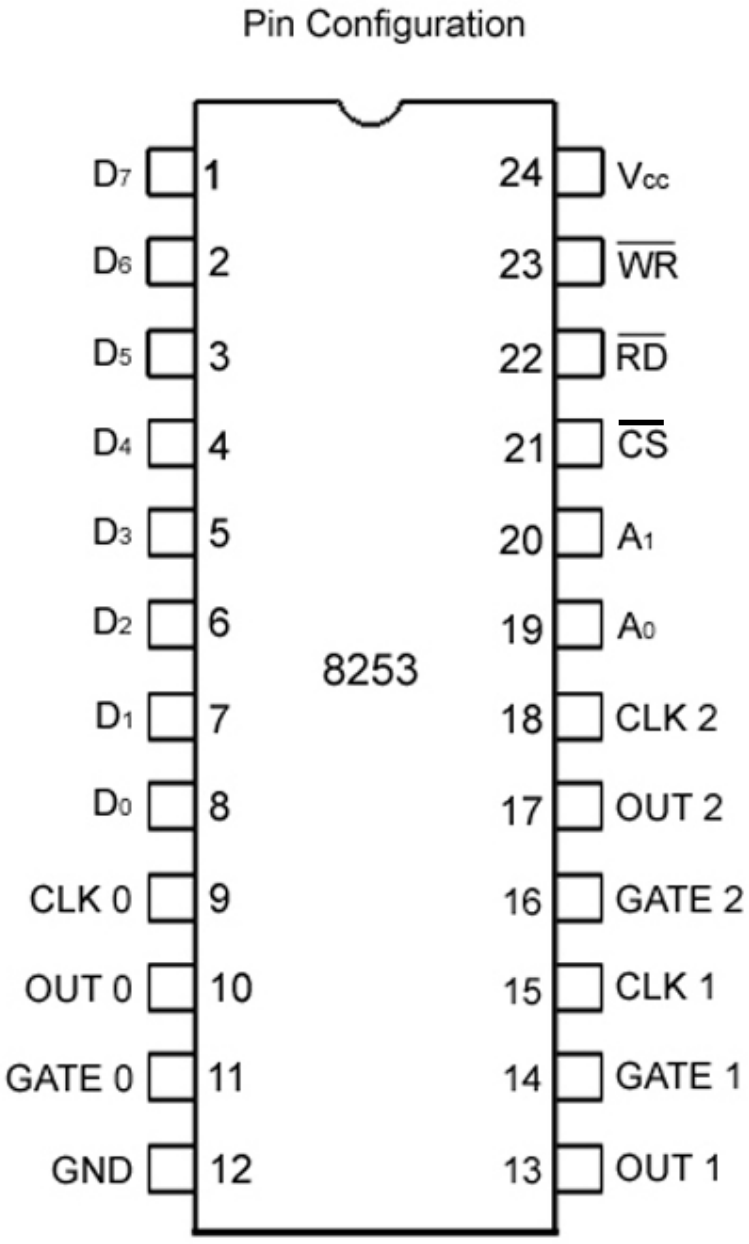


Figure – 14.1.2 Pin diagram of 8253

A_0-A_1: Address Lines:

These are active high input lines. To distinguish different parts of 8253 these address lines are used. These lines are internally used by 8253 to generate addresses as follows.:

A_1	A_0	Select Part
0	0	Counter – 0
0	1	Counter – 1
1	0	Counter – 2
1	1	Control Register

CLK_0, CLK_1, CLK_2: Clock input:

These lines are clock input to counters. CLK_0, CLK_1 and CLK_2 are input clocks of Counter-0, Counter-1 & Counter-2 respectively. Respective counters will count the pulses applied at its pin.

Gate-0, Gate-1, Gate-2: Gate Control:

By using this pin, an external hardware can control the counter. The Gate-0, Gate-1 and Gate-2 are control pins for Counter-0, Counter-1 and Counter-2 respectively. These pins are having different functions in different modes.

OUT 0, OUT 1, OUT 2: Output Pins:

These lines are active high output lines, used to give output of the counters. The OUT-0, OUT-1 and OUT-2 are output pins for Counter-0, Counter-1 and Counter-2 respectively.

14.2 Interfacing circuits with P.C.B. aspects.

Figure 14.2.1 shows the expansion 40-pin FRC connector with pin label. This connector is used to connect the circuit of 8253 with microprocessor kit.

The detailed circuit diagram of this peripheral card is given in Figure – 14.2.2.

In Figure – 14.2.2 we can see three buffer ICS 74LS245, one programmable Interval Timer IC 8253, nine LEDS and nine current limiting resistors, one 40-pin FRC male connector and some socket pins to provide external signals or to read the status through logic analyzer.

Here the data bus buffer IC 74245 is directed by RD signal. When RD becomes low, the direction will be from B to A otherwise A to B.

All the signals like RD, WR, A_0, A_1, CS and AD_0-AD_7 are buffered by using buffer IC 74LS245.

The same type of buffer IC also buffers the clock, gate & output signals.

AD7	A8
AD6	A9
AD5	A10
AD4	A11
AD3	A12
AD2	A13
AD1	A14
AD0	A15
$\overline{\text{INTA}}$	ALE
$\overline{\text{RD}}$	$\overline{\text{WR}}$
$\overline{\text{IO/M}}$	VCC
RESET OUT	INTR
$\overline{\text{CS}}$	GND
	RST 6.5
	RST 7.5
CLK	TRAP
A7	A6
A5	A4
A3	A2
A1	A0

Figure – 14.2.1 Expansion FRC connector of microprocessor kit.

Figure – 14.2.2 The schematic circuit diagram of 8253

Figure-14.2.3(A) The bottom layer of interfacing card of timer IC 8253
(inserted on page no. 212_1)

Figure-14.2.3(A) The overlay layer of interfacing card of timer IC 8253
(inserted on page no. 212_1)

The port addresses are as given in Table-14.2.1.

Table – 14.2.1

No.	Counter/CWR	Address
1	Counter – 0	80H
2	Counter – 1	81H
3	Counter – 2	82H
4	CWR	83H

Figure-14.2.3(A) is showing the PCB layout of the bottom layer of the Figure–14.2.2.

Figure-14.2.3(B) is showing the top view (overlay) of the interfacing card and all the straight lines are showing jumper wires.

Input & Output Observation:

The input means the clock and gate signals and output means out pin of the counter.

The slow changes can be observed on LEDS but the fast changes can be observed either by CRO or Logic analyzer.

To test this interfacing card we have captured the waveforms of all the six-modes. By analyzing this waveform the user can get clear idea about different modes.

- Procedure to work with this card
  - ⇨ Connect this card with 8085 based microprocessor kit.
  - ⇨ Connect any pin of port-B of 8255 (microprocessor kit) at gate-0 pin, and any pin of port-C at Clock-0 pin.
  - ⇨ Load the given software in RAM/EPROM area of microprocessor kit .
  - ⇨ Execute the software in single-step to see the LED ON-OFF States or execute directly software and capture the waveforms by using CRO or Logic analyzer.

Here in the present experiment we are generating clock by using software in microprocessor kit. In software the clock is generated continuously, and a high signal is provided to gate.

14.3 Programming aspect of 8253

⇨ **Control Word & explanation.**

The control word and its explanation is shown in Figure-14.3.1.

D7	D6	D5	D4	D3	D2	D1	D0
SC1	**SC0**	**RL1**	**RL0**	**M2**	**M1**	**M0**	**BCD**

SC1	SC0	SELECTION
0	0	COUNTER-0
0	1	COUNTER-1
1	0	COUNTER-2
1	1	ILLEGAL

M2	M1	M0	MODE
0	0	0	0
0	0	1	1
X	1	0	2
X	1	1	3
1	0	0	4
1	0	1	5

RL1	RL0	OPERATION
0	0	COUNTER LATCHING OPERATION
0	1	READ/LOAD LSBYTE ONLY
1	0	READ/LOAD MSBYTE ONLY
1	1	READ/LOAD LSBYTE FIRST AND THEN MSBYTE

BCD	MEANING
0	BINARY COUNTER 16-BIT
1	BCD COUNTER 4-DECADES

Figure-14.3.1 control word with explanation

⇨ Mode, explanation and Timing diagram.

Detailed Mode Description

In this section the six modes are described in detail; mode setting and count loading operations are described.

Mode 0: Interrupt On Terminal Count

Figure 14.3.2 shows the operation of a counter in Mode 0.

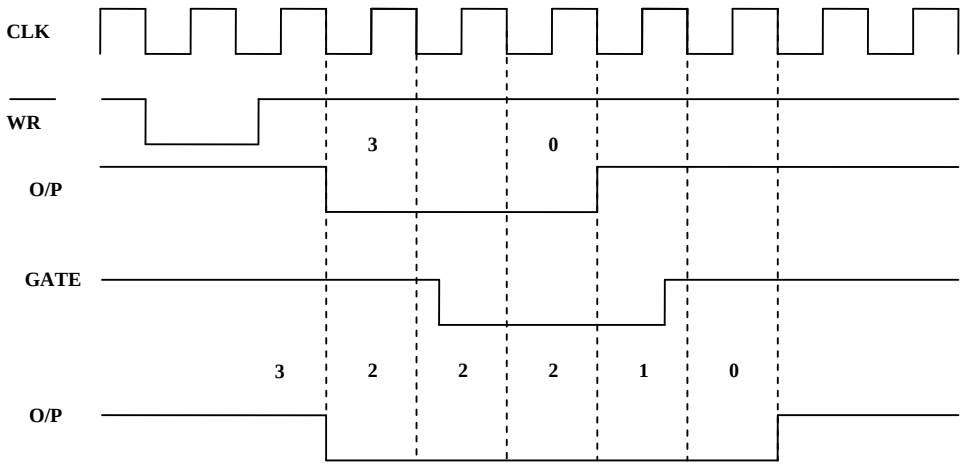


Figure-14.3.2 :MODE-0 :Interrupt On Terminal Count

OUTPUT

After the mode is set, the output becomes low and remains low while the selected counter is loaded with the count. The counter decrements the loaded count starting from the falling edge of the clock pulse which occurs just after the completion of the loading operation (WR goes high at the end of the loading operation) and continues to decrement the count at each subsequent clock pulse. When the terminal count is reached, the output becomes high and stays high till the count register is loaded with a new count or the mode of operation is changed.

GATE

If the Gate pin is made low while the counter decrements, the counting stops, and then current contents are held. The process resumes only after the Gate pin is made high again. The counter decrement starts again from the time of the falling edge of the clock pulse subsequent to the rising edge of the Gate signal.

Further; if the count register is reloaded while the counting is on, then (i) after the first byte of the count is written, the current counting stops, and (ii) after the second byte of the count is written, the counting restarts with the new count number.

Mode 1: Programmable One-Shot (A Negative Pulse of Controllable Width)

Fig. 14.3.3 shows the operation of a counter in Mode 1.

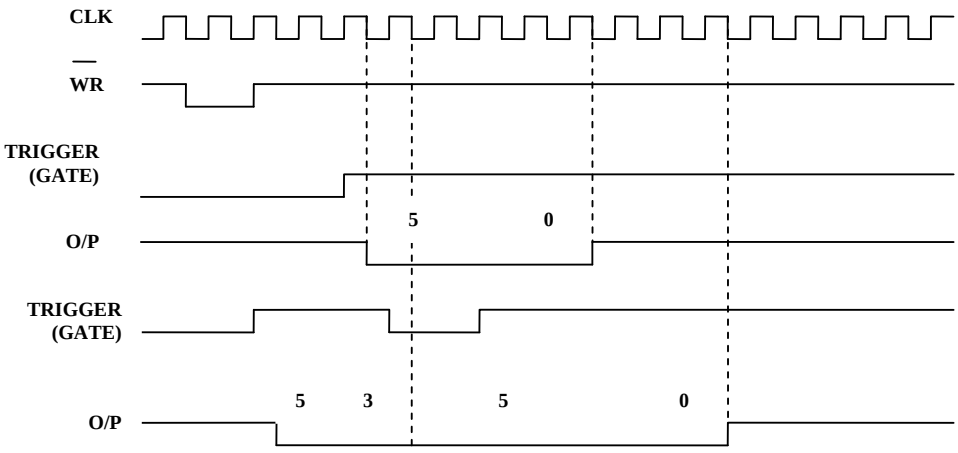


Figure-14.3.3 :MODE 1: Programmable One-Shot (A Negative Pulse Of Controllable Width)

OUTPUT

The output remains high while the mode is set and the count is loaded. It goes low after the rising edge of the Gate input (as shown in Fig. 14.3.3, the trigger goes high between the 4th and 5th clock pulse, while the output goes low on the falling edge of the 5th clock pulse). The output remains low till the terminal count is reached at which it becomes high again.

TRIGGER

If a trigger (Gate) signal appears while the output is low, the count is automatically reloaded into the counter and it is decremented from the full count again. The output remains low for the full count after an intermediate rising edge of the Gate input.

A new count may be added while the output is low, but it will not affect the duration of the one-shot pulse until the next trigger. Also, the current count can be read at any time and it will not affect the one-shot pulse.

Mode 2: Rate Generator

Fig. 14.3.4 shows the operation of a counter in Mode 2.

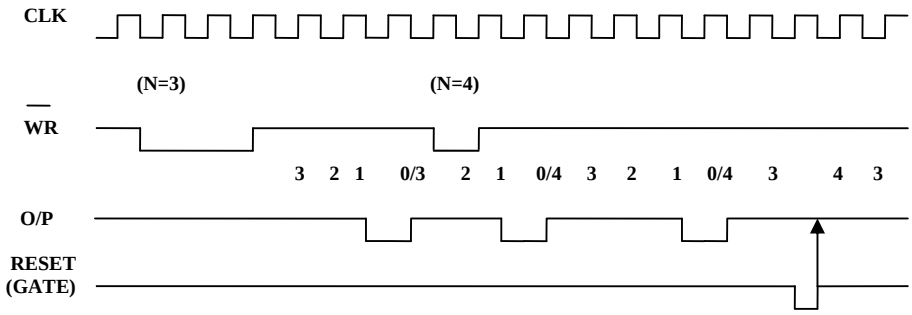


Figure-14.3.4 :MODE 2 :Rate Generator

OUTPUT

In this mode, depending on what is loaded for n as the count, after n pulses, the output goes low for one clock period (clock ON time + clock OFF time). Then it again becomes high for n pulses and low for one clock period and so on. It therefore works as divide by n counter. If the counter is working in this mode and a new n value is loaded, the current output pulse timing is not affected; the next one is, according to the new value of n.

GATE

When the Gate input (Reset) goes low, it forces the output to go high and inhibits the counting. When the Gate goes high, the output starts its count from that point onwards. The Gate input can therefore be used to synchronize the counter with the Gate input pulse. The output can also be synchronized through software. The output goes high as soon as the mode is set. The count needs to be loaded and it starts counting only after the count is loaded.

Mode 3: Square Wave Rate Generator

This is similar to Mode 2 with the exception that the output remains low for half the count and is high for the other half (for even n).

There are two possibilities depending on whether n is even or odd.

(I) If n is even

The counter is decremented by two on the falling edge of each clock pulse till the Terminal Count is reached. The state of the output changes at this time (high to low or vice-versa), the count is then reloaded with the full count and the process is again repeated. Figure 14.3.5 shows the output of counter in Mode 3.

(ii) If n is odd

When the output is high, the first clock pulse decrements the count by one and subsequent clock pulses decrement the count by two till the Terminal Count is reached. The output then is made low and the full count is reloaded. When the output is low, the first clock pulse decrements the count by three and subsequent clock pulses decrement the count by two till the Terminal Count is reached (see Figure 14.3.5).

In summary, if the count is odd, for (n+1)/2 pulses the output is high, and for (n-1)/2 pulses the output is low.

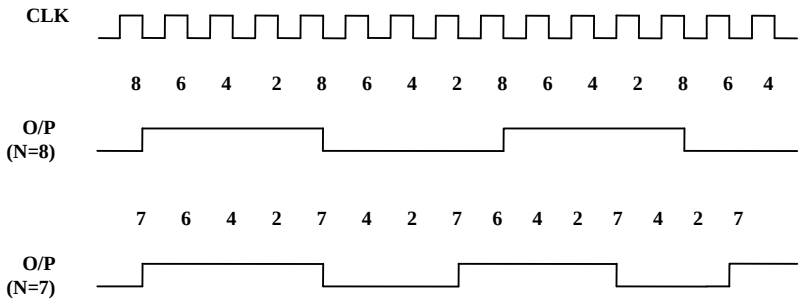


Figure-14.3.5 :MODE 3: Square Wave Rate Generator

Mode 4: Software Triggered Strobe

OUTPUT

A software-controlled delayed negative pulse of one clock period duration with or without synchronization is generated in this mode. Figure 14.3.6 shows a counter operation in Mode 4. In this mode, after the mode is set, the output becomes high. Counting starts after the counter is loaded. On reaching Terminal Count, the output goes low for one clock period and goes high again. If during counting, the count register is reloaded, the new count is loaded during the next clock pulse.

GATE
The counting is inhibited for the time the Gate input stays low. The Gate input can therefore be used for synchronization.

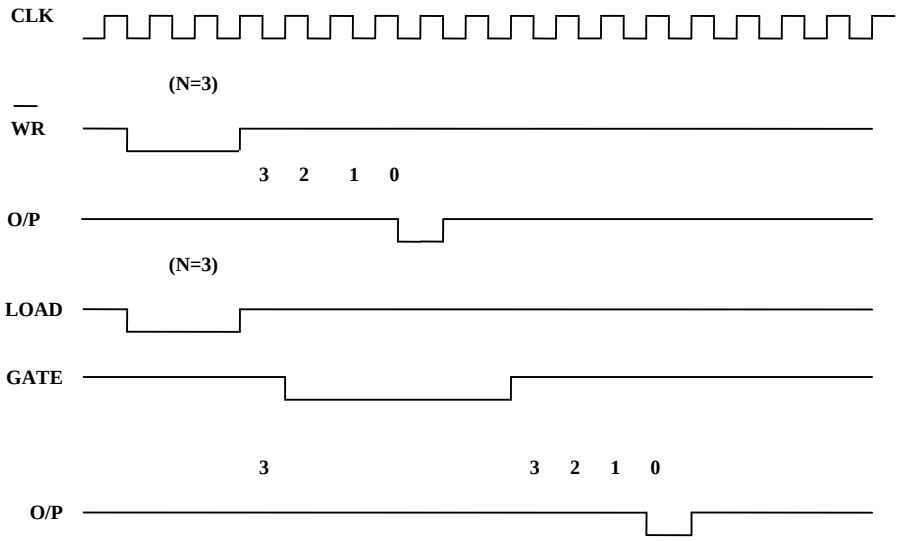


Figure 14.3.6 :MODE 4 :Software Triggered Strobe

Mode 5: Hardware Triggered Strobe

In this mode of operation, a delayed negative pulse with a width of one clock period is generated following a positive going trigger input at the Gate. Figure 14.3.7 shows the output of a counter in Mode 5.

OUTPUT
The counting starts after the rising edge of the Gate input, and on Terminal Count, the output goes low for one clock period. The counter can be re-triggered. After the rising edge of an trigger, the output does not go low until the full count.

The control words are different for different modes. The modes and its corresponding control words are given in Table 14.3.

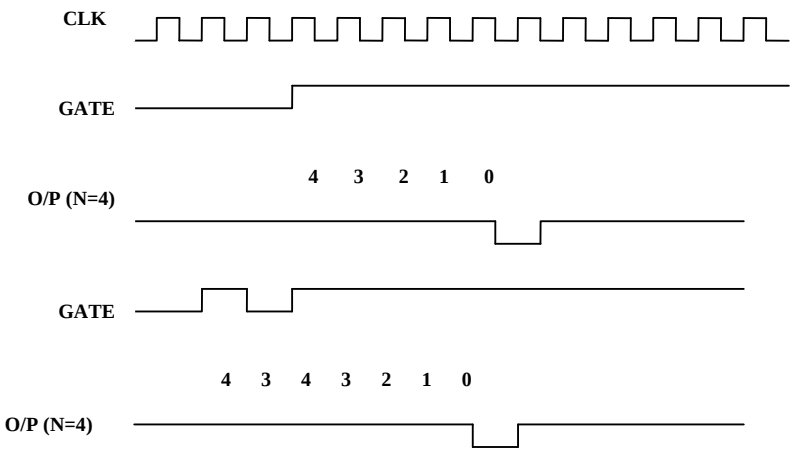


Figure-14.3.7 :MODE 5: Hardware Triggered Strobe

Table-14.3

Mode	Control Word
Mode-0	10H
Mode-1	12H
Mode-2	14H
Mode-3	16H
Mode-4	18H
Mode-5	1AH

The software for all these modes can be verified by capturing and comparing the waveforms.

Here clock is also generated by the software so WR signal is come so many times, so in captured waveform we have connected CS pin in place of WR. Figure 14.3.8 to Figure 14.3.13 shows output on logic analyzer of this interfacing card. Software for Mode-0 to Mode-5.

```
        MOV A,80H  ;CONTROL WORD OF 8255 ;CLOCK
        OUT 03H
        MVI A,10H/12H/14H/16H/18H/1AH;CONTROL WORD ;OF 8253
        OUT 83H
        MVI A,04H    ;COUNTS
        OUT 80H
        MVI A,FFH
        OUT 01H            ;FOR GATE PIN
LOOP:   OUT 00H            ;FOR CLOCK
        CMA
        NOP
        JMP LOOP
        .END
```

Figure 14.3.8 Mode-0 output on logic analyzer

Figure 14.3.9 Mode-1 output on logic analyzer

Figure 14.3.10 Mode-2 output on logic analyzer

Figure 14.3.11 Mode-3 output on logic analyzer

Figure 14.3.12 Mode-4 output on logic analyzer

Figure 14.3.13 Mode-5 output on logic analyzer

14.4 Examples

Experiment – 1:

You are given a microprocessor kit and an interfacing module of counter IC 8253. Write programs to check working of counter_____ (0, 1 or 2) in mode_____ (0 to 5) and capture waveforms by using Logic analyzer.

Experiment – 2:

You are given a microprocessor kit and an interfacing module of counter IC 8253. Write programs to check working of counter_____ (0, 1 or 2) in modes_____ (0 to 5) and see logics of all pins (Gate, Clock & Output) of the counter on LEDS.

CHAPTER-15 INTERFACING USART 8251

This USART (Universal Synchronous / Asynchronous Receive Transmit) 8251 chip is used for serial communication. [6] The COM port of the PC can be used to interchange data between a PC and our microprocessor trainer kit through this interfacing card.

15.1 Basics of 8251

Features of USART 8251

- It is compatible with 8085 MPU.
- It supports both synchronous & asynchronous modes of operation.
- It needs single +5V power supply.
- It can detect an error like parity, over run and framing.
- Synchronous baud rate is possible from DC to 64 K.
- Asynchronous baud rate is possible from DC to 19.2 K.
- All inputs / outputs are TTL compatible.
- 5 to 8 bit data transmission is being possible.

The block diagram of USART 8251 is given in Figure 15.1.2.

The block diagram in Figure – 15.1.2 consists of following blocks.

1. Data bus buffer.
2. Read / Write control logic.
3. Transmitter section.
4. Receiver section.
5. Modem Control.

1. Data bus buffer

This is a tri – state bi-directional buffer. This buffer is used to connect internal data bus of USART 8251 to the system bus of microprocessor 8085. The direction of buffering is decided by Read or Write signal.

2. Read / Write Control Logic:

This block is controlling all blocks. This block accepts different control signals like Read, Write, Control / Data, Chip Select, Clock etc.

The Table – 15.1 summarizes the functions of different control signals.

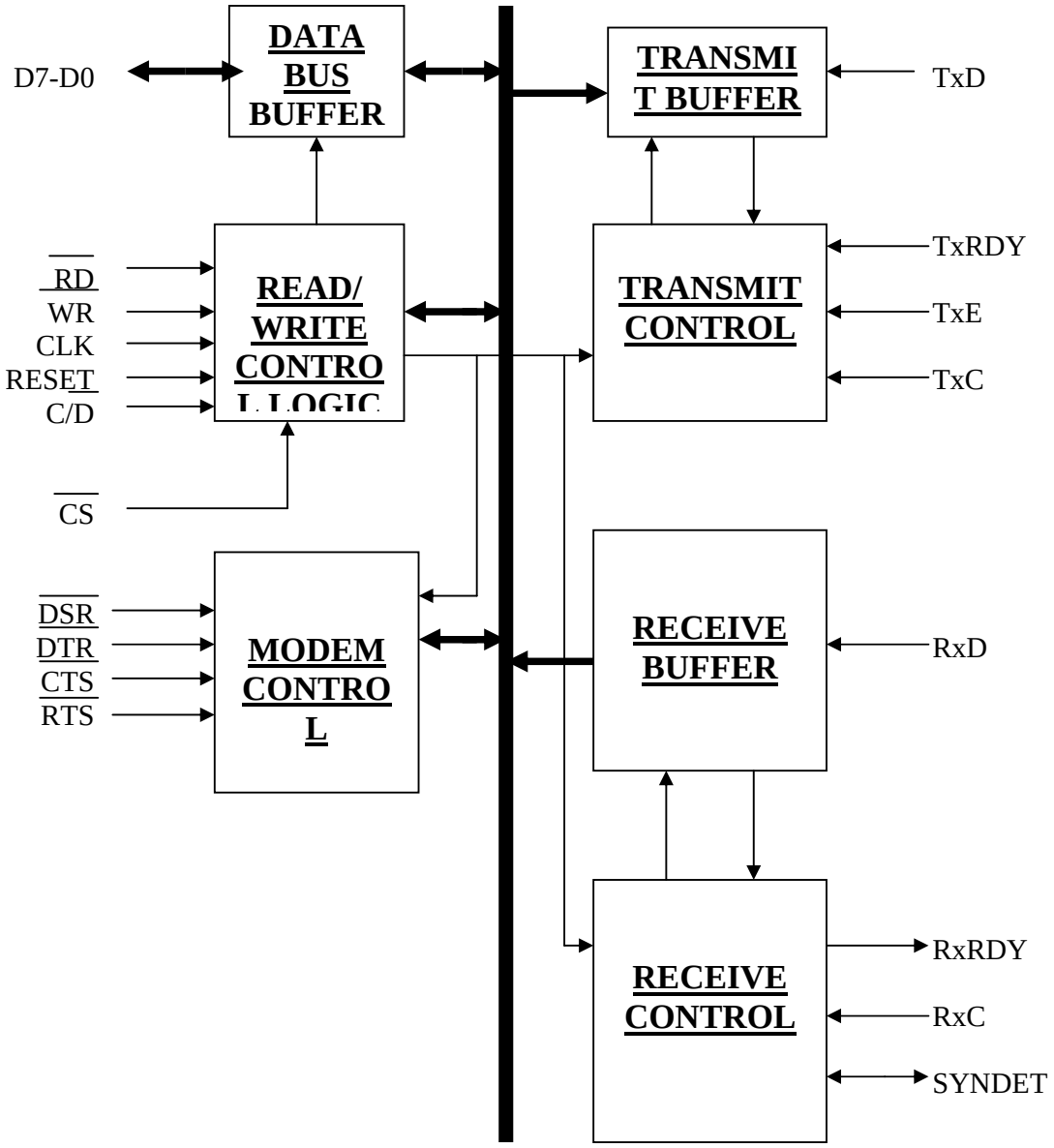


Figure 15.1.2 Block diagram of USART 8251

3. Transmitter Section

Figure – 15.1.3 is showing blocks of Transmitter Section.

This Transmitter Section consists of the transmitter buffer register, an output shift register and Transmit Control logic.

The transmitter buffer register accepts parallel data from the data bus buffer. The contents of the transmitter buffer are automatically transferred to the serial output register, when the output register becomes empty.

The control on both these blocks are provided by Transmit Control logic.

Table – 15.1

C / $\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Function
0	0	1	0	8251 Data port to Data bus
0	1	0	0	Data Bus

				contents to 8251 Data port
1	0	1	0	Status read
1	1	0	0	Control register write
X	1	1	0	Data Bus Tri-stated
X	X	X	1	Data Bus Tri-stated

4. Receiver Section

Figure – 15.1.4 is showing block diagram of receiver section.

This Receiver section consists of a Receiver buffer register, a serial input register and control logic. The USART accepts serial data in input register. The input register block also converts this serial data into parallel form. This parallel data is loaded into receiver buffer register under influence of receiver control logic block.

5. Modem Control

When data needs to be sent on telephone line, it becomes necessary to convert digital data into analog form while transmission and vise versa while reception. MODEM means Modulation and Demodulation of signals.

The pins $\overline{DTR}$, $\overline{DSR}$, $\overline{RTS}$ and $\overline{CTS}$ are used to control modem by 8251. These pins are also used to communicate with personal computer.

The pin diagram of 28 – pin DIP USART 8251 is shown in Figure 15.1.1

Pin Description of USART 8251

Data Bus: $D_0 – D_7$ Pin numbers: 27, 27, 1, 2, 5, 6, 7, 8

These are bi-directional data lines connected to data bus of microprocessor 8085. The microprocessor can send or receive data and/or command from 8251 on these lines.

RxD: Receive data – pin no.- 3

This is an input pin. This pin is used to receive the serial stream of data. This pin is generally connected with the transmitter pin of the transmitter.

GND: Ground – pin no.- 4

This pin is connected with ground terminal of the power supply.

TxC: Transmitter Clock – pin no.- 9

This input pin is used to provide clock to the transmitter block of USART. This pin decides the rate at which data is transmitted.

In Synchronous mode the baud rate is same as the frequency at this pin.

In Asynchronous mode the baud rate is variable, means the baud rate is frequency divisible by either 64 or 16 or 1.In both the modes the data bit is transmitted on falling edge of TxC Clock.

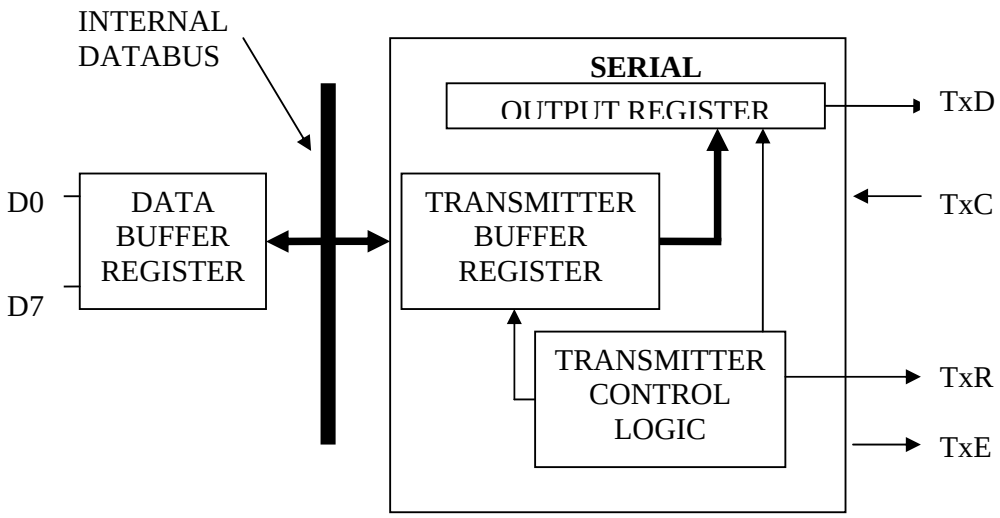


Figure: 15.1.3 Transmitter section

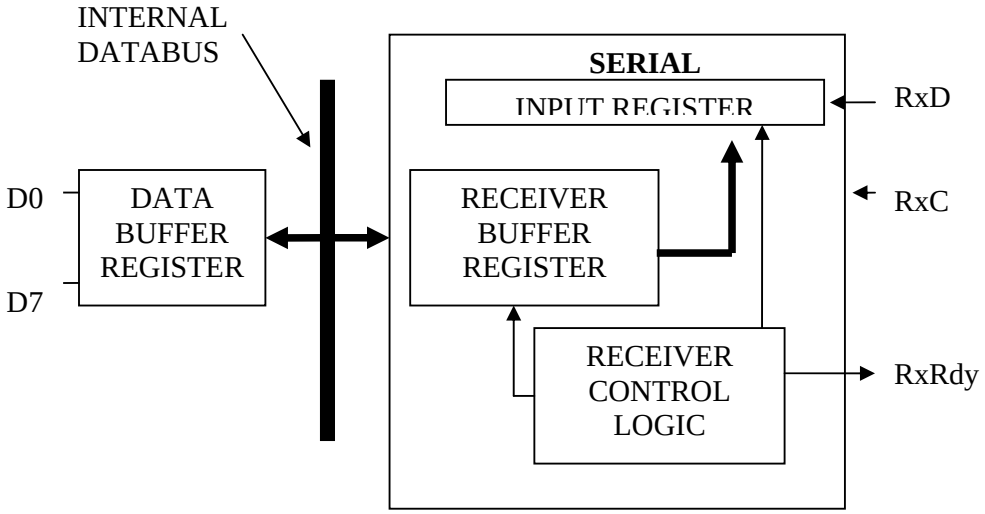


Figure:15.1.4 Receiver scetion

$\overline{\text{WR}}$: Write pin no.- 10

This is an input pin. This is used to write data into 8251. When this pin becomes low, the data bus buffer accepts data from system data bus. This pin is connected with write output signal of microprocessor 8085.

$\overline{CS}$: Chip Select

This is an input pin. The signal on this pin is used to select 8251 chip. When signal on this pin is low, so 8251 is selected, and if this pin high 8251 is not selected.

C / $\overline{D}$: Control / Data – pin no.- 12

This is an input pin. This pin is used to differentiate between control section and Data Section. When this pin is at logic high, the Control Section is selected. Similarly, when this pin is at logic low, the Data Section is selected. Generally, this pin is connected to the address line A_0 of microprocessor kit.

$\overline{RD}$: Read: pin no.- 13

This pin is an input pin. By using this pin microprocessor kit can read data from 8251. When this pin is low, the data bus buffer accepts data from selected part and sends it to microprocessor kit.

RxRDY: Receiver Ready pin no.- 14

This is an output signal. This signal is used to interrupt CPU. When this pin is enabled means the data is ready in 8251 to transfer in microprocessor 8085. If this pin is not connected with interrupt, then microprocessor can get information about the status of this pin by status read. In both asynchronous and synchronous modes when data is transferred to data output register then RxRDY is set.

TxRDY: Transmitter Ready pin no.- 15

This is an output signal. This signal is used to inform the microprocessor 8085 that the transmitter is ready to accept data. Generally, this TxRDY pin is connected with interrupt of 8085. If this pin is not connected with interrupt, then microprocessor can get information about the status of this pin by status read. This pin is automatically reset by write signal.

SYNDET / BD: Synchronous detect or break detect: pin no.- 16

This pin is bi-directional pin. The direction of this pin is decided by the mode selection.

In synchronous mode this pin is used as synchronous detection. When 8251's internal synchronous detection circuit is used, so this pin behaves as output pin. If external synchronous detection circuit is used, this pin behaves as input pin. This is programmed through mode word. When external SYNDET is connected, then the internal SYNDET circuit will be disabled.

In Asynchronous mode this pin behaves as output pin. It is now known as break detect. The output of this pin is activated to indicate break in character.

The status of this pin can also be read by status word read.

$\overline{CTS}$: Clear To Send pin.- 17

This is an input pin. If TxE bit in command byte is set to one and the 8251 is enabled to transmit serial data. If this pin becomes high, 8251 will stop transmitting data.

TxE: Transmitter Empty: pin no.- 18

This is an output signal. This signal indicates that a transmitter has no character to transmit. The TxE is automatically reset upon receiving a character from microprocessor.

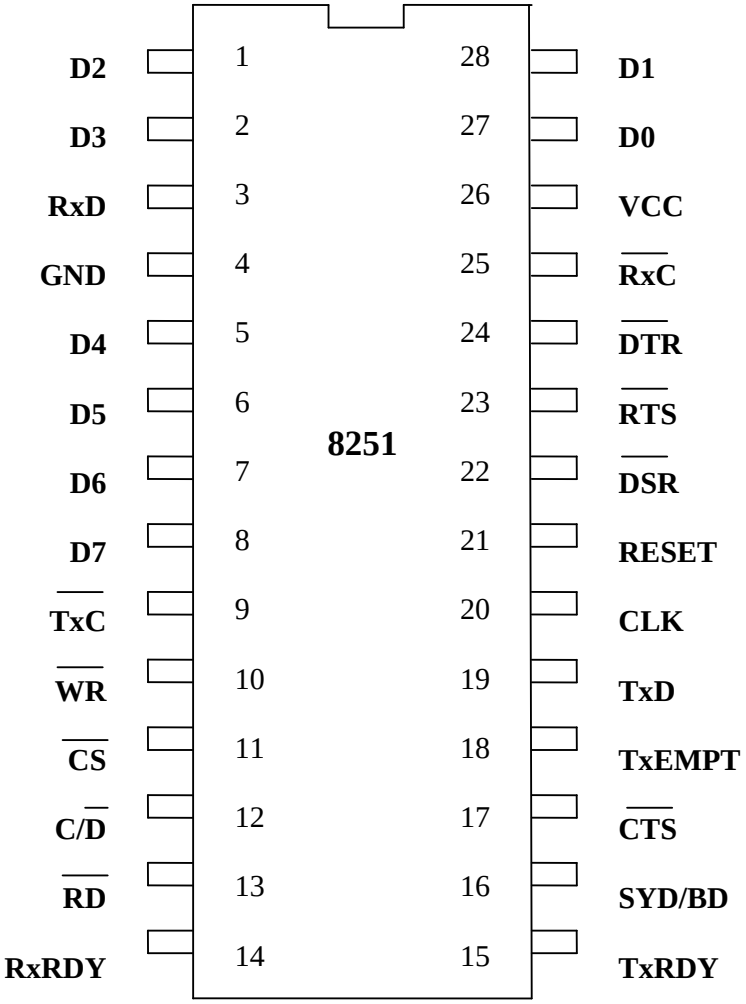


Figure 15.1.1 PIN DIAGRAM OF 8251

TxD: Transmit Data: pin no.- 19

This is an output pin. This pin is used to output the serial stream of data. When data is not transmitting, then this pin is held in high logic.

CLK: Clock pin no.- 20

This is an input pin. This clock input is used for communication between microprocessor 8085 and USART 8251. This clock frequency must be greater then 30 times the receiver and transmitter clock frequencies.

RESET: Reset pin no.- 21

This is an input pin. This pin is used to reset 8251. Generally, this pin is connected with Reset out pin of 8085. To reset 8251, the status on this pin should remain low for at least 6 – clock pulses.

$\overline{DSR}$: Data Set Ready pin no.- 22

This is an input signal. This is general purpose, 1 – bit inverting input port. This pin is used to test MODEM, whether Data set is ready or not? The information about the logic at this pin can also achieve by status read.

$\overline{RTS}$: Request to Send pin no.- 23

This is an output pin. This pin is used to signal the MODEM that 8251 has data which is to be transmitted. The status of this pin can be controlled by setting or resetting bit D_5 of the command instruction word.

$\overline{DTR}$: Data Terminal Ready: pin no.- 24

This is an output pin. This pin is used to signal the MODEM about 8251's readiness to accept or transmit serial data. The status of this pin can be controlled by setting or resetting bit D_1 of the command instruction word.

RxC: Receiver Clock: pin no.- 25

This is an input pin. This pin is used to give clock to receiver. The receiving baud rate is decided by frequency given at this pin. In a synchronous mode the baud rate is either divided by 64 or 16 or 1 to the Receiver Clock Frequency. While in synchronous mode baud rate is same as receiver clock frequency.

V_{CC}: Supply Voltage pin no.- 26

This pin should be connected with +5V with respect to ground.

15.2 Interfacing circuits with P.C.B. aspects

For interfacing 8251 and creating a useful application a circuit was designed to establish communication between a PC and 8085 microprocessor based trainer kit.[13] The software for the communication is prepared using C-language.

To establish Serial Communication between microprocessor kit and Personal Computer this interfacing circuit is designed. This circuit includes Max – 232 IC and USART 8251 with few connectors and capacitors.

Figure 15.2.1 is showing as interfacing module as a schematic circuit diagram.

This module is based on serial communication protocols. Here we can see that the module connected with microprocessor kit via 8251 and module connected with PC via MAX – 232.

The port addresses for USART 8251 are 80H and 81H for data and control registers respectively.

Figure 15.2.1 Schematic diagram of an interfacing module of USART 8251

Figure 15.2.2 Schematic circuit diagram of on-board clock

Figure 15.2.3 Bottom layer of USART 8251 module

Figure 15.2.4 Overlay layer of USART 8251 module

Figure 15.2.5 Pin-to Pin connection of 25-pin D-type connector

The interfacing of module with PC is done through a D – type 25 – pin connector.

Here 1 to 1 cable is used between module to PC and module to kit.

Here in Figure 15.2.1 the transmitter and receiver clocks are not shown connected with any clocks. For this an option is given that user can use any external clock of his choice or can use on board clock. The circuit diagram of on – board clock is given in Figure 15.2.2.

Figure – 15.2.3 is the bottom layer of actual sized P.C.B. layout of the module, and Figure 15.2.4 is the overlay of actual sized P.C.B. All the straight lines in overlay are showing jumpers.

On the module if user wants on board clock, he should insert IC 555 at its location. If user wants to use external clock then he should remove IC 555 and should connect external clock at points A and B in Figure – 15.2.3.

The pin – to – pin connection of interfacing module and communication port is as shown in Figure 15.2.5.

The Figures 15.2.1 and 15.2.2 are put on the same P.C.B., which is shown in Figure 15.2.3.

15.3 Programming aspect of 8251.

To transmit or receive data mode and command instructions must be loaded into USART 8251.

The Mode – instruction format for an asynchronous operation is shown in Figure 15.3.1.

D7	D6	D5	D4	D3	D2	D1	D0
S2	**S1**	**EP**	**PEN**	**L2**	**L1**	**B2**	**B1**

Figure:15.3.1 Mode Instruction Format : Asynchronous Operation

Explanation of Figure 15.3.1 is as given below.

D0 AND D1: BAUD RATE FACTOR

D1	D0	MEANING
0	0	SYNCHRONOUS MODE
0	1	CLK/1
1	0	CLK/16
1	1	CLK/64

D2 AND D3: CHARACTER LENGTH

D3	D2	BITS PER CHARACTER
0	0	5-BITS
0	1	6-BITS
1	0	7-BITS
1	1	8-BITS

D4: PARITY ENABLE(PEN)

1=PARITY ENABLE

0=PARITY DISABLE

D5: EVEN PARITY GENERATION/CHECK(EP)
1=EVEN PARITY
0=ODD PARITY

D6 AND D7: NUMBER OF STOP BITS

D7	D6	NUMBER OF STOP BITS
0	0	INVALID
0	1	1-BIT
1	0	1 ½ -BITS
1	1	2-BITS

The mode instruction format for synchronous operation is shown in Figure 15.3.2.

D7	D6	D5	D4	D3	D2	D1	D0
SCS	**ESD**	**EP**	**PEN**	**L2**	**L1**	**0**	**0**

Figure:15.3.2 Mode Instruction Format : Synchronous Operation

Explanation of Figure 15.3.2 is as given below.

D0 AND D1: BOTH MUST BE 0.

D2 AND D3: CHARACTER LENGTH

D3	D2	BITS PER CHARACTER
0	0	5-BITS
0	1	6-BITS
1	0	7-BITS
1	1	8-BITS

D4: PARITY ENABLE(PEN)
1=PARITY ENABLE
0=PARITY DISABLE

D5: EVEN PARITY GENERATION/CHECK(EP)
1=EVEN PARITY
0=ODD PARITY

D6: EXTERNAL SYNCHRONOUS DETECTION(ESD)
1= SYNDET IS AN INPUT
0=SYNDET IS AN OUTPUT

D7: SIGNAL CHARACTER SYNCHRONIZATION
1=SINGLE SYNCHRONIZATION CHARACTER
0=DOUBLE SYNCHRONIZATION CHARACTER

The command instruction Format is shown in Figure – 15.3.3.

D7	D6	D5	D4	D3	D2	D1	D0
EH	**IR**	**RTS**	**ER**	**SBRK**	**RxE**	**DTR**	**TxEN**

Figure: 15.3.3 COMMAND INSTRUCTION FORMAT

Explanation of Figure 15.3.3 is as given below.

D0: TRANSMIT ENABLE
1=ENABLE
0=DISABLE

D1: DATA TERMINAL READY
HIGH WILL FORCE DTR OUTPUT TO ZERO

D2: RECEIVE ENABLE
1=ENABLE
0=DISABLE

D3: SEND BREAK CHARACTER
1= IT FORCES TxD LOW
0= NORMAL OPERATION

D4: ERROR RESET
1=IT RESETS ERROR FLAGS PE,OE AND FE
0=NORMAL OPERATION

D5: REQUEST TO SEND
HIGH WILL FORCE RTS OUTPUT TO ZERO.

D6: INTERNAL RESET
HIGH RETURNS 8251 TO MODE INSTRUCTION FORMAT

D7: ENTER HUNT MODE
1 = ENABLES SEARCH FOR
SYNCHRONIZATION CHARACTERS.
0 = NORMAL OPERATION
IT HAS NO EFFECT IN ASYNCHRONOUS MODE.

The status word format is shown in Figure – 15.3.4.

D7	D6	D5	D4	D3	D2	D1	D0
DSR	**SYNDET BRKDET**	**FE**	**OE**	**PE**	**TxEMPTY**	**RxRDY**	**TxRDY**

Figure: 15.3.4 STATUS WORD FORMAT

Explanation of Figure 15.3.4 is as given below.

D0: TxRDY STATUS BIT HAS DIFFERENT MEANING FROM THE TxRDY OUTPUT PIN. THE FORMER IS NOT CONDITIONED BY CTS AND TxEN; THE LATTER IS CONDITIONED BY BOTH CTS AND TxEN.

i.e. TxRDY STATUS BIT= DB BUFFER EMPTY

TxRDY PIN OUT= DB BUFFER EMPTY · (CTS=0) · (TxEN=1)

D1: RxRDY

THIS IS USED TO INTERRUPT CPU. WHEN THIS IS ENABLED MEANS THE DATA IS READY IN 8251 TO TRANSFER IN MICROPROCESSOR 8085.

D2: TxEMPTY

THIS INDICATES THAT A TRANSMITTER HAS NO CHARACTER TO TRANSMIT. THE TxEMPTY IS AUTOMATICALLY RESET UPON RECEIVING A CHARACTER FROM MICROPROCESSOR.

D3: PARITY ERROR

THE PE FLAG IS SET WHEN A PARITY ERROR IS DETECTED. IT IS RESET BY THE ER OF THE COMMAND INSTRUCTION. PE DOES NOT INHIBIT OPERATION OF THE 8251.

D4: OVER-RUN ERROR

THE OE FLAG IS SET WHEN THE CPU DOES NOT READ A CHARACTER BEFORE THE NEXTONE BECOMES AVAILABLE. IT IS RESET BY THE ER OF THE COMMAND INSTRUCTION. OE DOES NOT INHIBIT OPERATION OF THE 8251. HOWEVER, PREVIOUSLY OVERRUN CHARACTER IS LOST.

D5: FRAMING ERROR

THE FE FLAG IS SET WHEN A VALID STOP BIT IS NOT DETECTED AT THE END OF EVERY CHARACTER. IT IS RESET BY THE ER OF THE COMMAND INSTRUCTION. FE DOES NOT INHIBIT OPERATION OF THE 8251.

D6: SYNDET/BRKDET

IN SYNCHRONOUS MODE THIS IS USED AS SYNCHRONOUS DETECTION. WHEN 8251'S INTERNAL SYNCHRONOUS DETECTION CIRCUIT IS USED, SO THIS PIN BEHAVES AS OUTPUT PIN. IF EXTERNAL SYNCHRONOUS DETECTION CIRCUIT IS USED, SO THIS PIN BEHAVES AS INPUT PIN. THIS IS PROGRAMMED THROUGH MODE WORD. WHEN EXTERNAL SYNDET IS CONNECTED, THEN THE INTERNAL SYNDET CIRCUIT WILL BE DISABLED.

IN ASYNCHRONOUS MODE THIS PIN BEHAVES AS OUTPUT PIN. IT IS NOW KNOWN AS BREAK DETECT.

D7: DATA SET READY

INDICATES THAT THE DSR IS AT A ZERO LEVEL.

To check working of this module, two test experiments have been performed. In these experiments C – compiler have been used.

The programming aspect of this 8251 card with C – programming is given in detail in test experiments 1 & 2.

1. To establish communication from PC to 8085 based microprocessor kit.
2. To establish the communication from 8085 based microprocessor kit to PC.

1. <u>To establish communication from PC to 8085 based microprocessor kit.</u>

In this experiment the user is supposed to enter the desired 8 bit data (Hex format) from the ASCII keyboard of PC and let the same data be displayed on the seven segment display of the microprocessor kit. For this user has to load proper C programming in the computer and the matching program written in the assembly language in the microprocessor kit. These programs are as follows

<u>PCTOKT1.C</u>

```
#include <bios.h>
#include <conio.h>
#include<stdio.h>

#define COM1      1
#define DATA_READY 0x30
#define SETTINGS   ( 0x20 | 0x00 | 0x04| 0x03)

int main(void)
{
  int in, out, status, DONE = 0xdd;

  bioscom(0, SETTINGS, COM1);
  cprintf("... BIOSCOM [ESC] to exit ...\n");
  loop:
    status = bioscom(3, 0, COM1);
    printf("status=%x",status);
    if (status & DATA_READY)
                {
                   bioscom(1,DONE, COM1);
                }

 printf("Enter hex no. & 111 to come out of the program\n");
scanf(" %x",&DONE);
if(DONE!=0x111)
        goto loop;

  return 0;
}
```

PC_TO_KT.ASM

```
HL_DISP:        .EQUAL      1C1BH
DATADISP:       .EQUAL      1E64H
DELAY:          .EQUAL      1F3FH
A8251:          .EQUAL      0390H

        ORG A8251
            MVI A,00H
            OUT 81H
            OUT 81H
            OUT 81H
            MVI A,40H
            OUT 81H             ;FOR MASTER RESET
            MVI A,CEH   ;2-STOPBITS,NOPARITY,;8-BIT
                        ;DATA,BAUD=CLK/16
            OUT 81H
            MVI A,36H   ;RTS=0,ER=0,RxEN=1,DTR=0,TxEN=0
            OUT 81H

LOOP:       IN 81H              ;READ STATUS
            ANI 02H             ;IF RxRDY=0 WAIT
             JZ LOOP
            IN 80H              ;READ DATA
            MOV L,A
            CALL HL_DISP        ;TO UNPACK DATA
            CALL DATADISP       ;TO DISPLAY DATA
            JMP LOOP
            END
```

Discussion of the program

The program PCTOKT1.C written in C-language used a very important function called

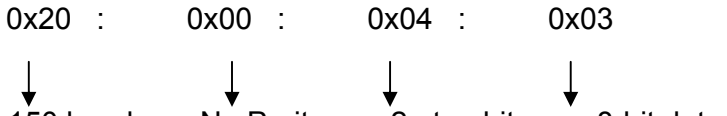
bioscom (int cmd, char abyte, int port)

The detail of this function is as explained bellow.

cmd

Value	Exlanation
0	Sets the communication parameters like baud rate, parity, stop bits and Number of data bits.
1	Sends the character in abyte out over the communication line
2	Receives a character from the communication line
3	Returns current status of the communications port.

abyte



port

0 = COMPORT – 1
1 = COMPORT – 2

here we have used com port –2.

In the program PCTOKT1.C the statement

bioscom(0,SETTING,COM1);

is used to set the communication parameters of comport-1, as define in the statement.

define SETTING (0x20 | 0x00 | 0x04 | 0x03);

the the statement

status=bioscom(3,0,COM1);

read the current status of comport1 and assigned this value to variable status.

If the status and the Data-ready values are matched, the statement bioscom (1.DONE,COM1);

Sends the character from com1 to the communication channel.

To stop the execution of this program enter 111H from the keyboard.

Now the program PC-TO-KT reads the character sent by PC by proper configuration of 8251 and display on the seven segment display.

Note that in this program (PC-TO-KT) subroutines and HL-DISP and DATADISP have been used as per our microprocessor kit. The user should use such routines as per his kit.

procedure

- Connect the interfacing module with the proper cable.
- Connect interfacing module with PC on COM2.
- Load program PC-TO-KT in the microprocessor kit. And execute it.
- Load the program PCTOKT1.C and execute it.
- At this moment the seven segment display of microprocessor kit will show the character DD.
- And the monitor of PC will display “Enter hex no. 111 to come out of the program”
- Now the system is ready for communication

- Enter the 8-bit hex data of your choice and press enter key to observe it on the seven segment of the kit.

Thus you have performed the communication

3. To establish the communication from 8085 based microprocessor kit to personal computer**.**

In this experiment more than one data byte is transferred from the memory of the microprocessor kit to PC. For this, user is supposed to enter the required data byte in the memory of microprocessor kit, before executing the programs in kit as well as in the PC.

The user has to enter the following programs in the microprocessor kit and PC.

KTTOPC1.C

```
#include <bios.h>
#include <conio.h>

#define COM1       1
#define DATA_READY 0x100
#define TRUE       1
#define FALSE      0

#define SETTINGS ( 0x20| 0x00 | 0x04 | 0x03)

int main(void)
{
  int in, out, status, DONE = FALSE,a[10],i=0;

  bioscom(0, SETTINGS, COM1);
  clrscr();
  cprintf("... BIOSCOM [ESC] to exit ...\n");
 bioscom(1, 0x00, COM1);
  while (!DONE)
  {
    status = bioscom(3, 0, COM1);
    if (status & DATA_READY)
    {
     out = bioscom(2, 0, COM1);
          a[i++]=out;
         bioscom(1, 0x55, COM1);
          if(i==5)
          goto end;
       }

  }
  end:
```

```
  for(i=0;i<5;i++)
  printf("\ndata=%x\n",a[i]);
  getch();
  return 0;
}
```

KT_TO_PC.ASM

```
HL_DISP:        .EQUAL    1C1BH
DATADISP:       .EQUAL    1E64H
DELAY:          .EQUAL    1F3FH
A8251:          .EQUAL    0290H

        ORG A8251
    MVI A,00H
    OUT 81H
    OUT 81H
    OUT 81H
    MVI A,40H  ;FOR SOFTWARE RESET
    OUT 81H
    MVI A,CEH ;2 STOP BITS,NO PARITY,8-BIT DATA,BAUD=CLK/16
    OUT 81H
    MVI A,37H  ;RTS=0,ER=0,RxEN=1,DTR=0,TxEN=1
    OUT 81H

    LXI H,4200H
    MVI B,05H

LOOP:
     IN 81H          ;READ STATUS
    ANI 01H          ;IF TxRDY=0 WAIT
    JZ LOOP

    MOV A,M          ;TRANSMIT DATA
    OUT 80H

LOOP2:
     IN 81H          ;READ STATUS
    ANI 02H          ;IF RxRDY=0 WAIT
    JZ LOOP2

    IN 80H           ;READ DATA
    STA 4300H;STORE DATA
    INX H
    DCR B
    JNZ LOOP
    RST 1

    END
```

Discussion of the program

This program is written in C-language uses the bioscom() as the main tool. Which is explained earlier in detail. This program prepares the PC to receive the data bytes from the microprocessor kit with the proper parameter.

SETTINGS decides this parameter. The bioscom(), when encountered first time in program configure the COM1. at the second time of its appeareance of bioscom() the dummy byte is transferred. Then the third statement

status = bioscom(3,0,COM1);

Check the status while the statement

out = bioscom(2,0,COM1);

receives the byte from the communication channel.

Lastly the statement bioscom(1,0x55,COM1); transmit dummy character to take care of the handshake signals.

Program KT-TO-PC.ASM written in assembly language configures 8251 in asynchronous mode for 2-stop bits, No-parity, 8-bit data and baud = CLK / 16 . then after checking the proper protocol signals the program transmits the data stored from 4200H onward.

In the last few instructions the program stores the dummy character transmited by PC to location 4300H.

After the data communication RST1 instruction hands over the control to the monitor routine of the microprocessor kit.

15.4 Examples

Example-1

You are given the microprocessor kit and module of USART 8251. Write a program to transmit 55H and Catch the Waveforms by logic analyzer.

Example-2

You are given the microprocessor kit and module of USART 8251. The personal computer with C – compiler is also given. Send five data from RAM of microprocessor kit and show it on screen of PC. To do this write programs in assembly and in C.

Example-3

You are given the microprocessor kit and module of USART 8251. The personal computer with C – compiler is also given. Develop program in kit and in PC by which entered 8 – bit hex data from key board of PC will be displayed on FNDS of microprocessor kit.

CHAPTER – 16 INTERFACING PIC 8259

PIC means Priority Interrupt Controller. The IC 8259 works as PIC. It manages the interrupts in interrupt driven system environment. Basically, it receives interrupts from peripheral devices, resolves their priorities and generates an interrupt to the 8085A on INTR pin. On receiving interrupt on INTR, 8085A generates three $\overline{\text{INTA}}$ signals: one for code of the CALL instruction and next two for the address of the subroutine for the interrupting device.

A single 8259 can handle eight different interrupts. The eight interrupting pins of 8259 can be connected with eight different 8259 chips. This is known as cascading. Full cascading provides maximum 64-levels of interrupts.[5]

The 8259 is used by proper programming. For this a set of initialization command words (ICWs) is used to configure 8259 as per users specification. The ICWs are used to specify addresses for concern interrupts, single or cascaded, level or edge triggered mode, master or slave, call address interval etc.

Operational command words (OCWs) are used to operate the 8259 in various modes. Which can be fully nested mode, rotating priority mode, special mask mode or polled mode. OCWs are also useful to mask specific interrupts and status read operation.

16.1. Basics of 8259

Basically 8259 is used to enhance the interrupt handing capacity of the system. It is a 28-pin device.

The internal block diagram of 8259 is shown in Figure – 16.1.1.

Interrupts and Control Logic Section

This section consists of: (a) Interrupt Request Register (IRR), (b) In-Service Register (ISR), (c) Priority Resolver, (d) Interrupt Mask Register (IMR), and (e) Control Logic Block.

(a) Interrupt Request Register (IRR)

The eight interrupt inputs set corresponding bits of the Interrupt Request Register. The IRR is used to store information about the interrupt inputs requesting service.

(b) In-Service Register (ISR)

The In-Service Register is used to store information about the interrupts currently being serviced.

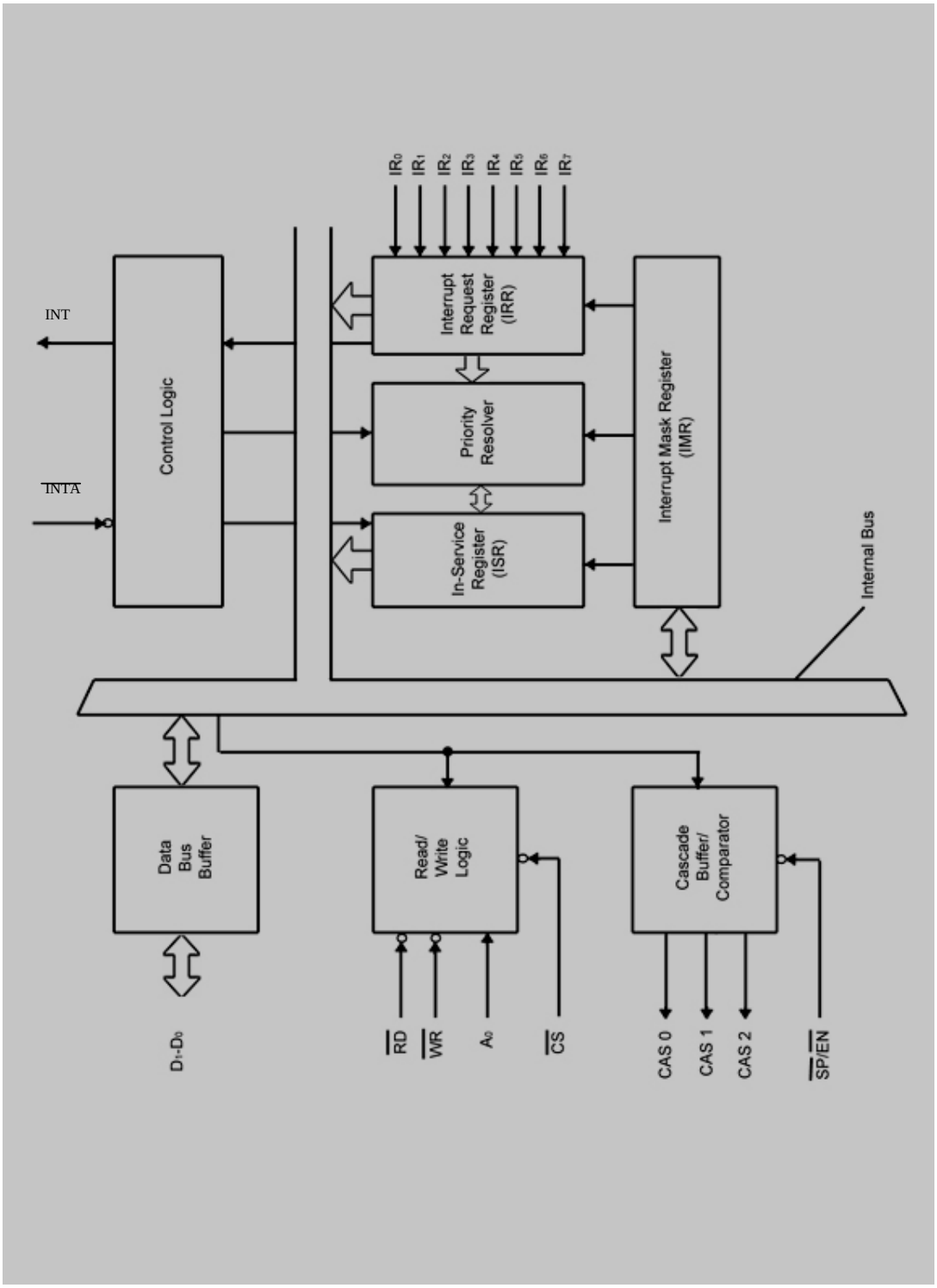


Figure 16.1.1 The internal block diagram of 8259

(c) Priority Resolver

This determines the priorities of the interrupts requesting service (which set corresponding bits of the IRR). The resolver determines the priorities as dictated by the priority mode set by the OCWs. The bit

corresponding to the highest priority interrupt input is set in the ISR during the $\overline{\text{INTA}}$ input.

(d) Interrupt Mask Register (IMR)

This register can be programmed by an OCW to store the bits which mask specific interrupts. The IMR operates on the IRR. An interrupt which is masked by software (by programming the IMR) will not be recognized and serviced even if it sets the corresponding bit in the IRR.

(e) Control Logic

This block has an input and an output line. The 8259, after resolving its input interrupt request priorities, puts out an interrupt request to the CPU, on the INT output. This is directly connected to the CPU interrupts input. In the 8085A, the INT output is connected to the INTR. The CPU responds to the request by putting out an $\overline{\text{INTA}}$. This signal is given to the 8259 on the $\overline{\text{INTA}}$ input. The 8259 then places the operation code for the CALL instruction on the data bus. This is read by the CPU and perceives that two additional INTAs are required to read the address (vectoring data) of the service routine. The 8259 places the two address bytes on the data bus when the two additional $\overline{\text{INTA}}$ signals are received.

Data Bus Buffer

This 8-bit bi-directional tri-state buffer is used to interface the 8259 to the system data bus. Control words for the 8259, status words, and vectoring data are all passed through the data bus buffer.

Read/Write Control Logic Section

This contains the Initialization Command Word Registers (ICW registers) and the Operation Command Word Registers (OCW registers) which are programmed by the CPU to set up the 8259, and to operate it in various modes. This section also accepts Read commands from the CPU to permit the CPU to read status words. The pins associated with this section are described below.

Chip Select $\overline{\text{CS}}$

This is an active low input which is used to select the device.

WRITE ($\overline{\text{WR}}$)

This is an active low input and is used to write OCWs and ICWs onto the 8259.

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

This is also an active low input. It is used by the CPU to read the status of the IRR, ISR, IMR or the interrupt level.

Cascade Buffer Comparator

This section generates control signals necessary for cascade operations. It also generates Buffer-Enable signals. As stated earlier, the 8259 can be cascaded with other 8259s in order to expand the interrupt handling capacity to sixty-four levels. In such a case, the former is called a master, and the latter are called slaves. The 8259 can be set up as a master or a slave by the $\overline{\text{SP}}/\overline{\text{EN}}$ pin in the non-buffered mode, or by software if it is to be operated in the buffered mode of operation (buffered and non-buffered modes of operation are describes later in this Section).

CAS 0-2

For a master 8259, the CAS0-CAS2 pins are outputs, and for slave 8259s, these are inputs. When the 8259 is a master (that is, when it accepts interrupt requests from other 8259s), the CALL opcode is generated by the

Master puts out an identification code of three-bits (to select one out of the eight possible slave 8259s) on the CAS0-CAS2 lines. The slave 8259s accept these three signals as inputs (on their CAS0-CAS2 pins) and compare the code put out by the master with the codes assigned to them during initialization. The slave thus selected (which has originally placed an interrupt request to the master 8259) then puts out the address of the interrupt service routine during the second and third $\overline{INTA}$ pulses from the CPU.

$\overline{SP}/\overline{EN}$ (Slave Program/Enable Buffer)

This pin is used to specify whether the 8259 is to act as a master or a slave. If this pin is kept at 5V the 8259 understands that it is to function as a master, and if it is kept at 0V, the 8259 understands that it is to function as a slave.

In large systems where buffers are used to drive the data bus, the data put out by the 8259 is response to $\overline{INTA}$ cannot be accessed by the CPU (due to the data bus buffer being disabled). If an 8259 is used in the buffered mode (buffered or non-buffered modes of operation can be specified at the time of initializing the 8259), the $\overline{SP}/\overline{EN}$ pin is used as an output which can be used to enable the system data bus buffer whenever the 8259's data bus outputs are enabled (when it is ready to put out data).

To summarize, in non-buffered mode, the $\overline{SP}/\overline{EN}$ pin of an 8259 is used to specify whether the 8259 is to operate as a master or as a slave, and in the buffered mode, the $\overline{SP}/\overline{EN}$ pin is used as an output to enable the data bus buffer of the system.

Figure 16.1.2 shows the pin diagram of 8259A.

16.2. Interfacing circuit with P.C.B. aspect

To understand the important features of 8259, we developed an interfacing circuit. This circuit detail is shown in Figure – 16.2.1.

To understand the concept of cascading we have considered two 8259A out of which one is working as MASTER and another is working as SLAVE.

We have used the nomenclature for this two 8259 as "MASTER" for the master 8259A and "SLAVE" for the slave 8259A.

The master is connected with the 8085 through a buffer IC 74LS245 and a 40-pin FRC connector. A_0, $\overline{RD}$ and $\overline{WR}$ are connected with A_0, $\overline{RD}$ and $\overline{WR}$ of 8085A. $\overline{SP}/\overline{EN}$ pin is connected with V_{CC} for MASTER. The INT and $\overline{INTA}$ pins are connected with INTR and $\overline{INTA}$ pins of 8085A respectively through buffer. The chip selects of MASTER and SLAVE are obtained through decoder 74155.

The input lines of decoder A and B are connected with address lines A_4 and A_5 of 8085A. Out of three active low enable inputs of 74155, two are connected with the $\overline{CS}$ output of decoder 74155 of the CPU.

This decoder of CPU circuit decodes the addresses as shown in the following Table 16.2.1.

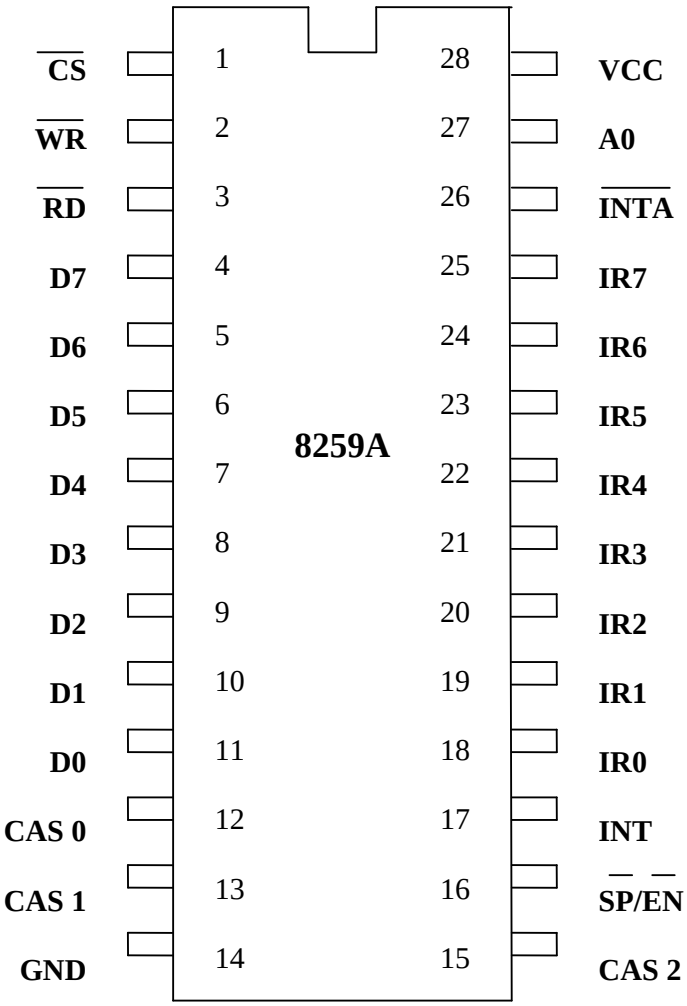


Figure 16.1.2 PIN DIAGRAM OF 8259A

Table 16.2.1:CPU decoder information

A15	A14	A13	A12	A11	A10	A9	A8	Used Hex addresses	Selection
A7	A6	A5	A4	A3	A2	A1	A0		
0	0	X	X	X	X	X	X	00H to 03H	8255
0	1	X	X	X	X	X	X	40H & 41H	8279
1	0	X	X	X	X	X	X	80H, 81H, 90H, 91H	External Peripheral

From the Table – 16.2.1, it becomes clear that the Port - addresses for MASTER are 80H and 81H. While for SLAVE Port – addresses are 90H and 91H.

The device side connection of MASTER includes pins IR_0 to IR_7 and CS_0 to CS_2. In the present circuit IR_0 to IR_7 are connected to eight external push button type switches through NAND gates.

The switches SW_9 to SW_{15} constitute one input of each respective NAND gate. The MASTER switch becomes the second input to the all NAND gates. Note that the NAND gate connected with IR_7 interrupt line is getting as input the INT of SLAVE through an inverter made of NPN transistor.

For the SLAVE data lines D_0 to D_7 are connected with AD_0 to AD_7 through 40-pin FRC connector, just like the MASTER. $\overline{RD}$, $\overline{WR}$ and $\overline{INTA}$ are connected to $\overline{RD}$, $\overline{WR}$ and $\overline{INTA}$ pins of 8085A through buffer. The $\overline{SP}/\overline{EN}$ pin is grounded to suggest that it is a SLAVE. The cascade lines are directly connected with cascade lines of MASTER. The interrupt lines of SLAVE are connected to eight different push-button type switches through NAND gates. These eight switches SW_1 to SW_8 become one of the inputs for respective NAND gates. The other inputs of all the NAND gates are connected to a switch called SLAVE switch.

Circuit Function

To understand the 8259 in Master mode switches SW_9 to SW_{15} can be used, to generate interrupts on the IR_0 to IR_6 lines. Similarly to understand the 8259 in slave mode switches SW_1 to SW_8 can be used to generate the interrupts on IR_0 to IR_7 lines of SLAVE. The INT line of SLAVE is connected to the IR_7 level of master through an INVERTER and a NAND gate. This NAND gate is controlled by master switch and status of INT pin. In the present design we have selected the level sensitive types of interrupts.

To understand the concepts of priority and masking, either for MASTER or SLAVE one can use the MASTER switch or Slave switch respectively as the case may be. If master switch is pressed, it makes one of inputs of all NAND gates low. This in consequence makes interrupt pin IR_0 to IR_7 high. In other word by pressing the master switch we generate simultaneous interrupt requests on all interrupt lines. This forces us to decide the priority. One can selectively mask these interrupts. The same is true for SLAVE 8259. if one wants to generate interrupts on special interrupt line, individual switch can be pressed, saving slave and master switches.

Figure – 16.2.1. Schematic circuit of PIC 8259A card

In the present software we have chosen default priority mode (Fully Nested mode) for the MASTER i.e. When we press master switch IR_0 line will assume highest priority. The subroutine associated with IR_0 will get

executed. In the present case the subroutine for IR_0 is supposed to display the characters “0000”. In other words, for MASTER 8259 if one presses the master switch the kit will display “0000”. For rest of the interrupts, whenever interrupt takes place it will display numbers as indicated in the following Table 16.2.2.

Table – 16.2.2: Corresponding display for interrupt lines.

MASTER		SLAVE	
Interrupt line	Display data	Interrupt line	Display data
IR0	0000	IR0	DD00
IR1	1111	IR1	DD11
IR2	2222	IR2	DD22
IR3	3333	IR3	DD33
IR4	4444	IR4	DD44
IR5	5555	IR5	DD55
IR6	6666	IR6	DD66
IR7	7777	IR7	DD77

Note that to understand the priority concept we have selected the specific rotation mode for SLAVE 8259, where the control word (0CW2) is selected as C4H, which makes the IR_4 line to assume lowest priority. In this mode the immediate next higher interrupt line assumes highest priority. So in present case when a slave switch is pressed, it generates all eight interrupts simultaneous, on slave 8259, but because of the priority selected in software IR_5 line will assume as highest priority and hence, its subroutine will be excited to display the message “DD55”.

The present interfacing module can be used to test almost all concepts of 8259 through proper control words.

P.C.B. aspects

The schematic for the present circuit was prepared using computer software, there after the track layout was prepared manually using the software. The schematic and track layout design is shown in Figure 16.2.1. and Figure 16.2.2.

Here Figure 16.2.2 shows the bottom layer and figure 16.2.3 shows the overlay of P.C.B. design.

Here straight lines in overlay show the jumpers.

Figure 16.2.2 Bottom layer of PIC-8259 card

Figure 16.2.3 overlay of PIC-8259 card

8259 Programming

The 8259 is programmed by the CPU by loading a set of

(i) Initialization Command Words, and

(ii) Operation Command Words.

Each 8259 in the system must first be initialized by loading a set of ICWs. OCWs can be loaded after initialization.

Initialization Command Word 1 (ICW1)

A write command issued to 8259 with A_0=0 and D_4=1 is interpreted as ICW1, which starts the initialization sequence. During ICW1, the following steps occur:

(a) The edge-sense circuit is reset. This implies that after initialization, an interrupt must make a low-to-high transition for it to perceived by the 8259.

(b) The IMR is cleared (all interrupts are disabled).

(c) The IR7 input is assigned the lowest priority.

(d) The slave mode address is set to 7.

(e) Status Read is set to IRR, the Special Mask Mode is cleared, and

(f) If bit D_0 in ICW1 (IC4) is set to '0', all functions in ICW4 are set to '0'.

The format of the byte to be loaded for ICW1 is shown in Figure. 16.2.4

A_5-A_7

A_0-A_4 of the vectoring address are automatically inserted by the 8259 for all the IR inputs for an interval spacing of four; A_0-A_5 are inserted automatically for an interval spacing of eight A_5-A_7 are programmable as set by bits D_5-D_7 of ICW1.

LTIM

This bit determines if the interrupts are to be recognized in the level-triggered mode or in the edge-triggered mode.

ADI

This sets the CALL address interval to either four or eight.

SINGL

This is used to inform the 8259 if it is the only 8259 in the system, or if additional 8259s are present.

IC4

This is used to specify whether ICW4 is required or not required. If it is set '0', the 8259 is set in the non-buffered mode, non-Auto EOI, and for operation with 8080/8085 systems by default.

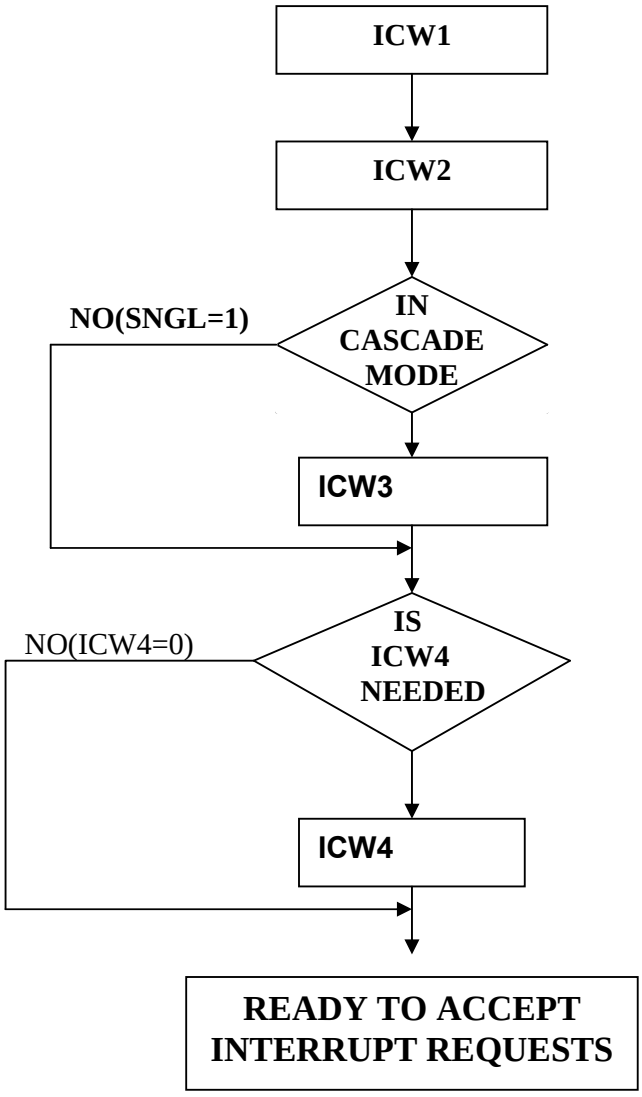


Figure-16.2.4: 8259 Initialization flow chart

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	**A7**	**A6**	**A5**	**1**	**LTIM**	**ADI**	**SNGL**	**IC4**

D0: IC4
1=ICW4 IS NEEDED.
0=ICW4 IS NOT NEEDED.
D1: SNGL
1=SINGLE MODE
0=CASCADE MODE
D2: ADI (CALL ADDRESS INTERVAL)
1=INTERVAL OF 4-ADDRESSES
0=INTERVAL OF 8-ADDRESSES
D3: LTIM (LEVEL TRIGGERED MODE)
1=LEVEL TRIGGERED MODE
0=EDGE TRIGGERED MODE
D4: THIS BIT SHOULD BE '1'.

D5,D6,D7: A5,A6 AND A7 BITS OF INTERRUPT VECTOR ADDRESS.

A0: THIS ADDRESS LINE SHOULD BE LOW TO SELECT ICW1

Figure-16.2.5(a) : initialization command word 1

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	**A15**	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**

D7 to D0: A15 TO A8 OF INTERRUPT VECTOR ADDRESSES.

A0: **THIS ADDRESS LINE SHOULD BE HIGH TO SELECT ICW2**.

Figure-16.2.5(b) initialization command word-2

Initialization Command Word 2 (ICW2)

A write command following ICW1, with A_0=1, is interpreted as ICW2. The format of the byte to be loaded as ICW2 is shown in Figure-16.2.5(b).This is used to load the high-order byte of the interrupt vector address of all the interrupts. It should be noted that this byte is common for all the interrupts.

Initialization Command Word 3 (ICW3)

ICW3 is required only if there is more than one 8259 and if they are cascaded. An ICW3 operation loads a slave register in the 8259. The format of the byte to be loaded as an ICW3 for master 8259 or a slave 8259 is shown in figure16.2.6.

⇨ **ICW3 FOR MASTER DEVICE**

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	**S7**	**S6**	**S5**	**S4**	**S3**	**S2**	**S1**	**S0**

D7 TO D0: S7 TO S0

1=IR INPUT HAS A SLAVE.

0=IR INPUT DOESNOT HAVE A SLAVE.

A0: THIS ADDRESS LINE SHOULD BE HIGH TO SELECT ICW3.

ICW3 FOR SLAVE DEVICE

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	**0**	**0**	**0**	**0**	**0**	**ID2**	**ID1**	**ID0**

D0 TO D2: ID0 TO ID2

ID2	ID1	ID0	SLAVE ID
0	0	0	SLAVE –0
0	0	1	SLAVE –1
0	1	0	SLAVE –2
0	1	1	SLAVE –3
1	0	0	SLAVE –4
1	0	1	SLAVE –5
1	1	0	SLAVE –6
1	1	1	SLAVE –7

D3 TO D7: THESE BITS SHOULD BE ZERO.

A0: THIS ADDRESS LINE SHOULD BE '1'.

Figure-16.2.6 Initialization Command Word-3

MASTER MODE ICW3

If the 8259 currently being initialized is a master (as determined by its $\overline{SP}/\overline{EN}$ pin being high in a non-buffered environment, or by M/S = 1 in ICW4 in a buffered environment), each bit in ICW3 is used to specify to the master whether it has a slave 8259 attached to it on its corresponding IR (Interrupt Request) input. A '1' is used to specify the presence of a slave at that input, and a '0' is used to specify the absence of a slave.

Later, after initialization, if one of the slave 8259s (say, the slave 8259 whose INTR output is connected to the IR2 input of the master 8259) raises its INTR output, the master generates the CALL instruction Opcode, and puts out the identification number of the slave (in this case, 010) on the CAS0-CAS2 lines; all the slaves compare the code on their common CAS0-CAS2 lines (the CAS0-CAS2 outputs of the master are connected to the CAS0-CAS2 inputs of all the slave 8259s for cascaded operation) with the Slave Identification code loaded into them during initialization. The slave which had originally placed the interrupt request to the master is enabled by the CAS0-CAS2 lines and it releases the appropriate vector address during the second and third $\overline{INTA}$ cycles.

SLAVE MODE ICW3

If the 8259 currently being initialized is a slave (as determined by its $\overline{SP}/\overline{EN}$ pin being low in a non-buffered system environment, or by M/S=0 in ICW4 in a buffered environment), bits D_0-D_2 of ICW3 are used to assign a slave identification code (slave ID) to the 8259. It should be noted that the slave ID is equivalent to the master IR input to which the INTR output of the slave is connected. Later, after initialization, the slave compares this code with the CAS0-CAS2 inputs in order to release the address vector.

Initialization Command Word 4 (ICW4)

As shown in Figure-16.2.5(A), ICW4 only if the D_0 bit of ICW1 (IC4) is set. The format of ICW4 is shown in Figure-16.2.7.

SFNM

If this bit is set, the Special Fully Nested Mode is programmed.

BUF

This is used to indicate to the 8259 whether it is in a buffered or non-buffered environment. If BUF=1, the $\overline{SP}/\overline{EN}$ pin is used as an output to enable the data bus buffer of the system. The 8259 can be configured as a master or a slave by the M/S bit.

M/S

In the buffered mode of operation, M/S=1 sets up the 8259 being initialized as a master, and M/S=0 sets it up as a slave. In the non-buffered mode, M/S has no significance as the master/slave selection is dependent on the status of the SP/EN pin.

AEOI

If the AEOI mode is set to '1', the Automatic End Of Interrupt mode is programmed.

PM

This is used to specify whether the 8259 is to operate in an 8080/8085A environment or in an 8088/8086 environment.

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	**0**	**0**	**0**	**SFNM**	**BUF**	**M/S**	**AEOI**	**MPU**

D0: MPU

1=8086/8088 MODE
0=8085/8080 MODE

D1: AEOI

1=AUTOMATIC EOI
0=NORMAL EOI

D2 & D3: M/S & BUF

BUF=0 NON BUFFERED MODE

BUF=1 BUFFERED MODE
M/S=1 MASTER
M/S=0 SLAVE

D4: SFNM

1=SPECIAL FULLY NESTED MODE
0= NOT SPECIAL FULLY NESTED MODE

Figure-16.2.7 Initialization Command Word-4

Operation Command Words (OCWs)

After initialization, the 8259 is ready to process interrupt requests. However, during operation, it might be necessary to change the manner of processing the interrupts. Operation Command Words (OCWs) are used for this purpose. They may be loaded anytime after the 8259's initialization to dynamically alter the priority modes. Figure-16.2.8. shows the format of the Operation Control Words.

OCW1

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	**M7**	**M6**	**M5**	**M4**	**M3**	**M2**	**M1**	**M0**

OCW2

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	**R**	**SL**	**EOI**	**0**	**0**	**L2**	**L1**	**L0**

OCW3

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	**0**	**ESMM**	**SMM**	**0**	**1**	**P**	**RR**	**RIS**

Figure-16.2.8 Operation Command Words

Operation Command Word 1 (OCW1)

A Write command to the 8259 with A_0=1 (after ICW2) is interpreted as OCW1. OCW1 is used for enabling or disabling the recognition of specific interrupt requests by programming the IMR. M=1 indicates that the interrupt is to be masked, and M=0 indicates that it is to be unmasked as shown in figure 16.2.9(A).

Operation Command Word 2 (OCW2)

A Write command with A_0=0, and D_4D_3=00 is interpreted as OCW2. The R (Rotate), SL (Select-Level), EOI bits control the Rotate and End Of Interrupt Modes, and combinations of two. Figure16.2.9(B) shows the Operation Command Word format. L_2-L_0 are used to specify the interrupt level to be acted upon when the SL bit is active.

OCW1

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	**M7**	**M6**	**M5**	**M4**	**M3**	**M2**	**M1**	**M0**

INTERRUPT MASK

1=MASK SET

0=MASK RESET

Figure 16.2.9(A) Format of OCW1

OCW2

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	**R**	**SL**	**EOI**	**0**	**0**	**L2**	**L1**	**L0**

D0 TO D2: IR LEVEL TO BE ACTED UPON

L2	L1	L0	IR LEVEL
0	0	0	LEVEL–0
0	0	1	LEVEL–1
0	1	0	LEVEL–2
0	1	1	LEVEL–3
1	0	0	LEVEL–4
1	0	1	LEVEL–5
1	1	0	LEVEL–6
1	1	1	LEVEL–7

D5,D6 & D7: EOI,SL & R

R	SL	EOI	MEANING
0	0	0	ROTATE IN AEOI MODE(CLEAR)
0	0	1	NON SPECIFIC EOI COMMAND
0	1	0	NO OPERATION
0	1	1	SPECIFIC EOI COMMAND
1	0	0	ROTATE IN AEOI MODE(SET)
1	0	1	ROTATE IN NON SPECIFIC EOI COMMAND
1	1	0	SET PRIORITY COMMAND
1	1	1	ROTATE ON SPECIFIC EOI COMMAND

A0: THIS ADDRESS LINE SHOULD BE LOW TO SELECT OCW2

Figure 16.2.9(B) Format of OCW2

Operation Command Word 3 (OCW3)

OCW3 is used to read the status of the registers, and to set or reset the Special Mask and Polled modes as shown in Figure-16.2.10.

8259 INTERRUPT MODES

The various modes of operation of the 8259 are:

(i) Fully Nested Mode,
(ii) Rotating Priority Mode,
(iii) Special Masked Mode, and
(iv) Polled Mode.

Fully Nested Mode (FNM)

This is the default mode setting after initialization. The 8259 continues to operate in the Fully Nested Mode until the mode is changed through Operation Command Words.

In this mode, the highest priority (Priority 0) is assigned to IR0 and the lowest priority (Priority 7) is assigned to IR7. When as interrupt request is acknowledged, the 8259 determines the interrupt of the highest priority and sets its corresponding bit in the ISR. The vector address corresponding to this interrupts is then put out. The ISR bit remains set until an End Of Interrupt

OCW3

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	**0**	**ESMM**	**SMM**	**0**	**1**	**P**	**RR**	**RIS**

D0 & D1:RIS AND RR

D1(RR)	D0(RIS)	MEANING
0	0	NO ACTION
0	1	NO ACTION
1	0	READ **IRR**
1	1	READ **ISR**

D2: POLL COMMAND

1=POLL COMMAND
0=NO POLL COMMAND

D3: THIS BIT SHOULD BE '1'.
D4: THIS BIT SHOULS BE '0'.
D5 & D6: SMM ESMM

D6(ESMM)	D5(SMM)	MEANING
0	0	NO ACTION
0	1	NO ACTION
1	0	**RESET** SPECIAL MASK
1	1	**SET** SPECIAL MASK

D7: THIS BIT SHOULD BE '0'.
A0: THIS ADDRESS LINE SHOULD BE LOW TO SELECT OCW3.

Figure16.2.10 Format of OCW3

(EOI) command through an OCW is issued by the CPU before exiting from the interrupt service routine. However, if the AEOI mode is set in ICW4 during

initialization, the ISR bit is automatically reset on the trailing edge of the last $\overline{\text{INTA}}$ pulse.

While as ISR bit is set, all lower interrupt levels and another interrupt of the same level are inhibited. However, an interrupt level higher than the ISR bit currently being serviced will cause the 8259 to generate an INTR. This interrupt can however be acknowledged only if the Interrupt Enable flip-flop of the microprocessor has been enabled through software (in the interrupt service routine of the current interrupt).

The EOI and AEOI commands are described below.

End Of Interrupt (EOI)

The IS bit can be reset by an End Of Interrupt command issued by the CPU, usually just before exiting from the interrupt routine.

In the Fully Nested Mode, the highest level in the ISR would necessarily correspond to the last interrupt acknowledged and serviced. In such a case, a non-specific EOI command may be issued by the CPU (by an OUT instruction to the 8259, with A_0=0, $D_7D_6D_5D_4D_3$ being 00100, $D_2D_1D_0$ being of no significance, as can be derived from OCW2 in figure-16.2.9) before exiting from the routine. The 8259 then resets the highest level IS bit (among those that are set).

However, if the FNM is not used, the 8259 may not be able to determine the last interrupt acknowledged. In such a case, a specific EOI command will have to be issued by the CPU (by an OUT instruction to the 8259 with A_0=0, $D_7D_6D_5D_4D_3$ being 00100, $D_2D_1D_0$ specifying the level on which the EOI command is to act, as shown in OCW2 in figure-16.2.9) before exiting from the interrupt service routine. The 8259 then resets the IS bit of the level specified by the EOI command.

It should be noted that in the cascade mode, the EOI command must be issued twice, once for the master and once for the slave.

Automatic End Of Interrupt (AEOI)

The AEOI mode is set by ICW4. If the AEOI mode is set, the 8259 will perform a non-specific EOI on its own on the trailing edge of the third $\overline{\text{INTA}}$ pulse. The AEOI mode can only be used for a master 8259 and not for a slave. If the interrupt system is enabled (by software) in the service routine, it is open to fresh interrupts from any level (as dictated by the ISR) since the bit corresponding to the interrupt being serviced is reset in the ISR.

Special Fully Nested Mode (SFNM)

In the FNM, on the acknowledgement of an interrupt, further interrupts from the same level are disabled. However, in large systems which use cascaded 8259s and where the interrupt levels within each slave have to be considered, this creates a problem. An interrupt input to a slave, in turn causes the slave to place an interrupt request to the master on one of the master's inputs. Further interrupts to the slave will cause the slave to place requests to the master on the same input to the master, but these will not be recognized because further interrupts on the same input level are disabled by the master.

The Special Fully Nested Mode (SFNM) is used to surmount this problem. The SFNM is set up by ICW4 during initialization. It is similar to the FNM except for the following differences:

(a) When an interrupt request from a slave is being serviced, the slave is allowed to place further requests (these requests are of a higher priority than the request currently being serviced). These interrupts are recognized by the master and it initiates interrupt requests to the CPU.

(b) Before exiting from the interrupt service routine, a non-specific EOI must be sent to the slave and its ISR must be read to determine if it was the only interrupt to the slave. If the ISR is empty, a non-specific EOI command can be sent to the master. If it is not empty, it implies that the same IR level input to the master is to be serviced again due to more than one interrupt being presented to the slave, and an EOI must not be sent to the master.

Rotating Priority Mode

The Rotating Priority mode can be set in (a) Automatic Rotation, and (b) Specific Rotation.

(a) Automatic Rotation

This mode is used in an interrupt structure where the interrupts must be assigned equal priority. In this mode the last serviced interrupt level IRX (X can vary from 0 to 7) is automatically assigned the lowest priority and the interrupt level IR (X + 1) is assigned the highest priority. For example, if the ISR status and the priority status are as shown in Figure-16.2.11(a) (IR3 and IR5 are being serviced, but IR3 service routine is being executed, as it is of a higher priority), after an EOI from the IR3 routine, the priorities will be as shown in Figure-16.2.11(b). Automatic rotation is possible by a non-specific EOI or by an AEOI. For automatic rotation on non-specific EOI, and OCW2 with R=1, SL=0, and EOI=1 must be written into the 8259. Automatic rotation mode on an AEOI is set by loading an OCW2 with R=1, SL=0, and EOI=0, and is reset by loading an OCW2 with R=0, SL=0, and EOI=0 (see Figure 16.2.9).

BEFORE ROTATE (IR3, IR5 BEING SERVICED,IR3 ACTOVE)

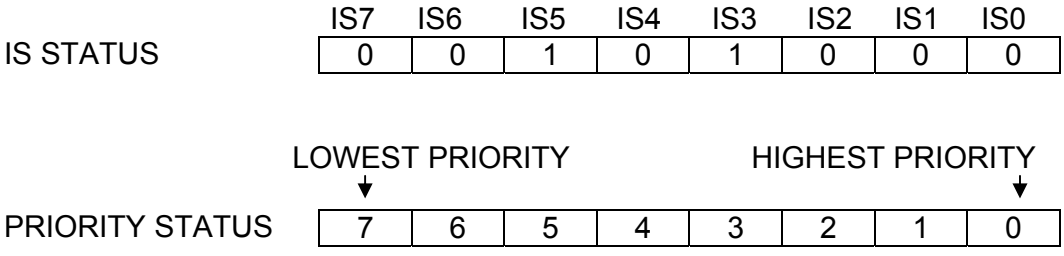


Figure-16.2.11(a) : Priority status before rotation

AFTER EOI FROM IR3 ROUTINE

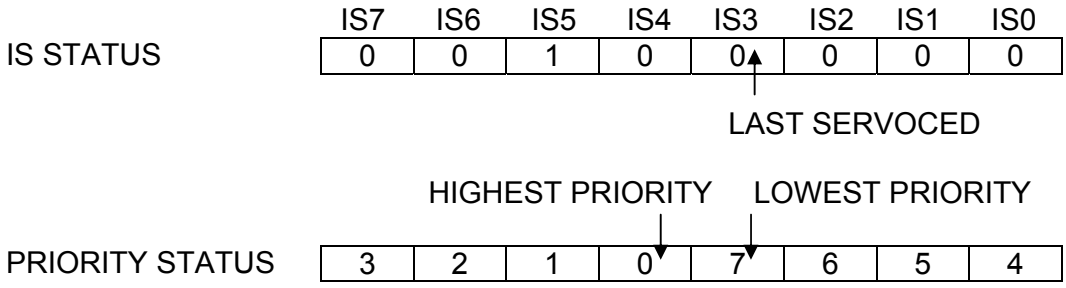


Figure-16.2.11(B) : Priority status after rotation

(b) Specific Rotation

In the Automatic Rotation mode, the interrupt request last serviced is assigned the lowest priority, whereas in the Specific Rotation mode, the lowest priority can be assigned to any level IRX (X can vary from 0 to 7) as specified by OCW2. I(X + 1) is then assigned the highest priority.

This mode is set by the CPU by issuing an OUT instruction with A_0=0, $D_7D_6D_5D_4D_3$ set to 11000, with D_2-D_0 specifying the interrupt level IRX that is to be assigned the lowest priority. It should be noted that this operation is independent of an EOI command. However, specific rotation can be accomplished by using the Rotate on Specific EOI option in OCW2 (R=1, SL=1, EOI=1, L_2-L_0 specify the interrupt level IRX that is to assigned the lowest priority on an EOI: see figure 13.13). In such a case, the priority changes are automatically made after the EOI command.

Special Mask Mode

It may be sometimes desirable to selectively enable lower priority interrupts. Usually, if an EOI command is not given to the 8259, the IS bit of the last serviced interrupt is not reset, and consequently all lower priority interrupts are kept disabled.

The Special Mask Mode can be set by making the ESMM and SMM bits '1' in OCW3. When a mask bit is set in OCW1, all further interrupts at that level are inhibited, while interrupts on all other levels that are not masked by OCW1 (both lower and higher) are enabled. It is thus possible to selectively enable interrupts by programming the mask register.

The Special Mask Mode can be cleared by loading an OCW3 with ESMM=1 and SMM=0.

Polled Mode

In the Polled Mode of operation, the INT output of the 8259 is either not connected to the INTR input of the 8085A, or the system interrupts are kept disabled by software. The devices are then serviced by the 8085A by polling the interrupt requests.

The Polled mode is set by making P=1 in OCW3. A subsequent Read command issued to the 8259 (with $\overline{RD}$=0, and $\overline{CS}$=0) is treated as an $\overline{INTA}$ by the 8259. It then sets the ISR bit corresponding to the highest level interrupt in the IRR and puts out a byte (the format of this type is shown in Figure-16.2.12) on the data bus. The 8085A can examine the status of D_7 to check if an interrupt needs to be serviced (D_7=1 implies that an interrupt is to be serviced, and D_7=0 implies that no interrupt has occurred). W_0-W_2 give the code of the highest priority interrupt level requesting service.

If an interrupt is detected, the software must include a CALL to an interrupt service routine (the address is dependent on the level of the interrupt, as detected by W_0-W_2). Since the INTR line is not used in this mode, more than one 8259 may be connected in the master mode; it is thus possible to have more than sixty-four levels of interrupts in this mode.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I	-	-	-	-	W_2	W_1	W_0

Figure 16.2.12 Polled Mode Output Word

8259 STATUS READ OPERATION

The status of the Interrupt Request Register, the In-Service Register, and the Interrupt Mask Register of the 8259 may be read by issuing appropriate Read commands as described below.

IRR Status Read

An OCW3 with RR=1 (Read Register) and RIS=0 (Read ISR) sets up the 8259 for a status read of the Interrupt Request Register. A subsequent Read command issued to the 8259 (with $\overline{RD}$, $\overline{CS}$=0) causes the 8259 to put out the contents of the IRR. When the 8259 is not in the Polled mode, after it is set up for an IRR status read operation, all Read commands with A_0=0 cause the 8259 to put out the IRR status word (the OCW3 needs to be written onto the 8259 only once).

ISR Status Read

An OCW3 with RR=1 and RIS=1 sets up the 8259 for a status read of the In-Service Register. A subsequent read command issued to the 8259 will cause the 8259 to put out the contents of the ISR onto the data bus. When the 8259 is not in the Polled Mode, after it is set up for an ISR status read operation, all Read commands with A_0=0 cause the 8259 to put out the ISR status word (the OCW3 needs to be written onto the 8259 only once).

IMR Status Read

A Read command issued to the 8259 with A_0=1 (with $\overline{RD}$, $\overline{CS}$=0) causes the 8259 to put out the contents of the Interrupt Mask Register. OCW3 is not required for a status read of the IMR.

It should be noted that after initialization, the default status read operation involves the IRR, unless the ISR status read is set up. Also, if P=1, RR=1 in OCW3, the status read operation is overridden by the Poll operation.

DEFAULT IR7 ROUTINE

If an interrupt input does not remain high until after the falling edge of the first $\overline{INTA}$ pulse, it is treated as an invalid input (this may occur due to a noise glitch). The 8259 then generates a default CALL to the IR7 routine. A valid IR7 input generates a normal CALL to the IR7 routine. A default IR7 operation does not set the corresponding ISR bit while a normal IR7 operation sets it.

The IR7 service routine is therefore generally used to execute only a RET instruction (in order to return the Stack Pointer to its initial value). If however, the IR7 routine is needed for other purposes, a default IR7 operation may be checked by including a status read operation of the ISR at the beginning of the interrupt service routine for IR7. If the bit corresponding to the IR7 input is set, it implies that it is a valid interrupt; and an invalid interrupt otherwise.

SOFTWARE

```
HL_DISP:      .EQUAL       1C1bH
DATADISP:     .EQUAL       1E64H
DELAY:        .EQUAL       1F3fH
MASTER:       .EQUAL       0E00H
SLAVE:        .EQUAL       0F00H
PROG:         .EQUAL       1000H

              ORG MASTER
                   LXI H,0000H      ;TO DISPLAY 0000H
                   JMP DISP
              ORG MASTER+8
                   LXI H,1111H      ;TO DISPLAY 1111H
                   JMP DISP
              ORG MASTER+16
                   LXI H,2222H      ;TO DISPLAY 2222H
                   JMP DISP
              ORG MASTER+24
                   LXI H,3333H      ;TO DISPLAY 3333H
                   JMP DISP
              ORG MASTER+32
                   LXI H,4444H      ;TO DISPLAY 4444H
                   JMP DISP
              ORG MASTER+40
                   LXI H,5555H      ;TO DISPLAY 5555H
                   JMP DISP
              ORG MASTER+48
                   LXI H,6666H      ;TO DISPLAY 6666H
                   JMP DISP
              ORG MASTER+56
                   LXI H,7777H      ;TO DISPLAY 7777H

DISP:         CALL HL_DISP
              CALL DATADISP
```

```
        LXI D,FFFFH
        CALL DELAY
        MVI A,20H       ;END OF INTERRUPT
        OUT 80H
        RET


         ORG SLAVE
        LXI H,DD00H     ;TO DISPLAY DD00H
        JMP DISP1
    ORG SLAVE+8
        LXI H,DD11H     ;TO DISPLAY DD11H
        JMP DISP1
    ORG SLAVE+16
        LXI H,DD22H     ;TO DISPLAY DD22H
        JMP DISP1
    ORG SLAVE+24
        LXI H,DD33H     ;TO DISPLAY DD33H
        JMP DISP1
    ORG SLAVE+32
        LXI H,DD44H     ;TO DISPLAY DD44H
        JMP DISP1
    ORG SLAVE+40
        LXI H,DD55H     ;TO DISPLAY DD55H
        JMP DISP1
    ORG SLAVE+48
        LXI H,DD66H     ;TO DISPLAY DD66H
        JMP DISP1
    ORG SLAVE+56
        LXI H,DD77H     ;TO DISPLAY DD77H

DISP1:  CALL HL_DISP
        CALL DATADISP
        LXI D,FFFFH
        CALL DELAY
        MVI A,20H       ;END OF INTERRUPT FOR MASTER
        OUT 80H
        MVI A,20H       ;END OF INTERRUPT FOR SLAVE
        OUT 90H
        RET

    ORG PROG
        LXI SP,5EFFH

        MVI A,18H       ;ICW1 FOR MASTER
        OUT 80H
        LXI H,MASTER
        MOV A,H         ;ICW2 FOR MASTER
        OUT 81H
        MVI A,80H       ;ICW3 FOR MASTER
```

```
        OUT 81H

        MVI A,18H      ;ICW1 FOR SLAVE
        OUT 90H
        LXI H,SLAVE
        MOV A,H        ;ICW2 FOR SLAVE
        OUT 91H
        MVI A,07H      ;ICW3 FOR SLAVE
        OUT 91H

        MVI A,C4H ;TO SET HIGHEST PRIORITY TO IR5 OF SLAVE
        OUT 90H


LOOP:   LXI H,FFFFH       ;TO DISPLAY FFFFH
        CALL HL_DISP
        CALL DATADISP
        LXI D,FFFFH
        CALL DELAY
        EI
        HLT
        JMP LOOP

        .END
```

16.4 Examples.

Experiment – 1

You are given a microprocessor kit and a peripheral module of PIC 8259. Write a program to see 0000 on display initially, and when we press any key of MASTER PIC means SW9, SW10….SW15 or 'MASTER', so FFFF will be displayed for few second.

Experiment – 2

You are given a microprocessor kit and a peripheral module of PIC 8259. Write a program to see FFFF on display initially. When we press any key of MASTER PIC means SW9, SW10….SW15 so on display we can see 0000, 1111….6666 respectively.

Experiment – 3

In experiment – 2 add the program to prove that when we press 'MASTER' key so PIC accepts IR_7. Explain about default priority of PIC 8259.

Experiment – 4

You are given a microprocessor kit and a peripheral module of PIC 8259. Write a program to see FFFF on display initially. When we press any key of SLAVE PIC means SW1, SW2…. SW8 so on display we can see DD00, DD11….DD77. Set highest priority for IR_5 and show by pressing 'SLAVE' key. CDD55 should display when we press 'SLAVE' key.

References

[1] "8080 microcomputer user's manual." Intel corporation, Santa Clara, CA, September 1975.

[2] Intel corporation "MCS - 80/80 family user's manual", Santa Clara, CA, 1979.

[3] Microprocessor architecture, programming and applications by R.S.gaonkar

[4] Microprocessor data handbook, Revised Edition, BPB Publications

[5] 0000 to 8085 Introduction to microprocessors for engineers and scientists by P.K.Ghosh and P.R.Shridhar, second edition, PHI Publications.

[6] Intel Component Data Catalog, 1979, Santa clara, California 95051.

[7] Printed Circuit Board by Dr.H.N.Pandya, Published by Gujarat Granth Nirman Board, Ahmedabad,India.

[8] Special Purpose Linear Devices, Databook, National Semiconductor.

[9] Design and construction of analog to digital conversion interfacing module for 8085 microprocessor kit. 18th Gujarat Science congress, Physics Department, Saurashtra University, Rajkot dated 13th March,2004.

[10] Data sheet 38, 1.2" DM DOTMATRIX 5x7 KLP1157I-CC, Kwality Photonics Pvt. Ltd., Hyderabad, INDIA.

[11] An intelligent control panel : A novel microprocessor based system. ETA-2003, 11th , 12th ,13th July,2003, Computer Department, Saurashtra University, Rajkot.

[12] 8085 based novel software technique for V to F type analog to digital conversion. Electronic Maker, June-2004

[13] Study of interfacing module to establish communications between a PC and a 8085 based microprocessor kit. LE, Lab Experiments, Volume-4,No-3,September-2004.

[14] Study of 8255 through experiments using microprocessor kit.LE, Lab Experiments, Volume-4,No-2,June-2004

[15] Understanding interrupts of 8085 using Logic Analyzer. ETA-2004,25th & 26th FEB.-2004, Computer Department, Saurashtra University, Rajkot.