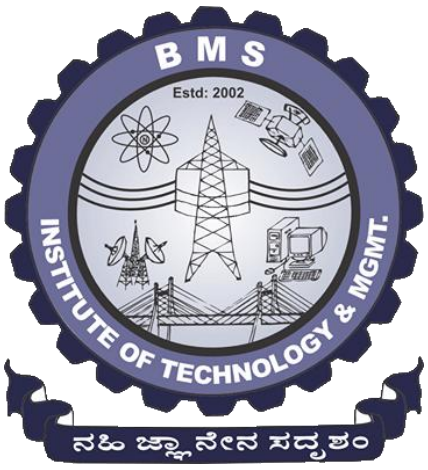
B.M.S INSTITUTE OF TECHNOLOGY & MANAGEMENT



Department of Computer Science & Engineering

LAB MANUAL

MICROPROCESSOR - HARDWARE PART (8086)

Sub Code: 10CSL48

4th Semester CSE

Prepared by:

Shankar. R

Under the Guidance:

Dr. G Thippeswamy
Professor & Head, CSE
BMSIT&M

Programs

1. b) Read the status of 8 input bits from the Logic Controller check parity
2. b) BCD Up-Down Counter on the Logic Controller Interface
3. b) Read the status of two 8-bit inputs (X & Y) from the Logic Controller Interface and display X*Y.
4. b) Display messages FIRE and HELP alternately on a 7-segment display
5. b) Assume any suitable message of 12 characters length and display it in the rolling fashion on a 7-segment display interface .
6. b) Convert a 16-bit binary value to BCD and display on a 7-segment display interface.
7. b) Scan a 8 x 3 keypad for key closure and display on screen.
8. b) Drive a Stepper Motor interface clockwise & counter wise
9. b) Generate the Sine Wave using DAC interface
10. b) Generate a Half Rectified Sine wave form using the DAC interface
11. b) Generate a Fully Rectified Sine waveform using the DAC interface
12. b) Alp to drive an elevator interface

The 8086 Microprocessor Pin Diagram

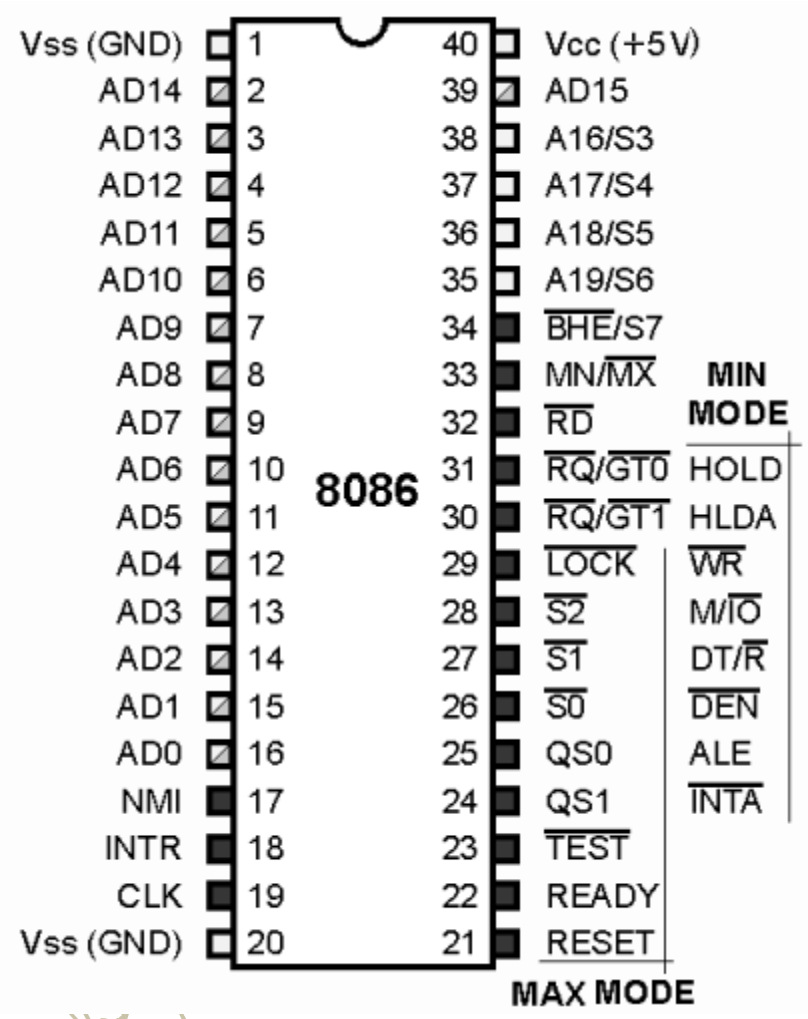
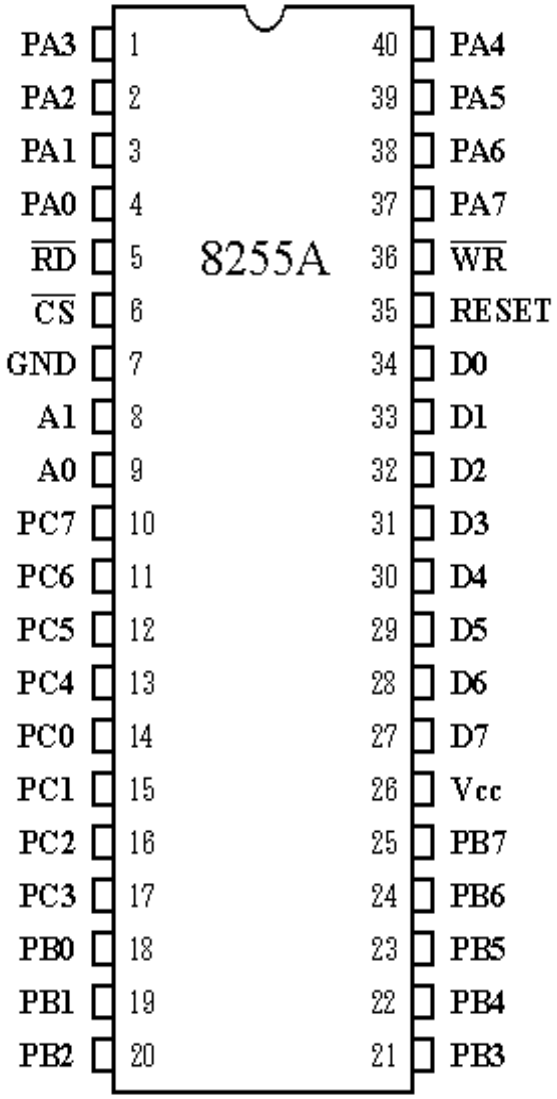


Fig:- The 8086 Microprocessor Pin Diagram

8255 Programmable Peripheral Interface (PPI)



Data Bus Buffer

This three-state bi-directional 8-bit buffer is used to interface the 8255 to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

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.

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

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

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

(A0 and A1) Port Select 0 and Port Select 1. These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word register. They are normally connected to the least significant bits of the address bus (A0 and A1).

(RESET) Reset. A "high" on this input initializes the control register to 9Bh and all ports (A, B, C) are set to the input mode.

A1	A0	SELECTION
0	0	PORT A
0	1	PORT B
1	0	PORT C
1	1	CONTROL

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 8255. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255. 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.

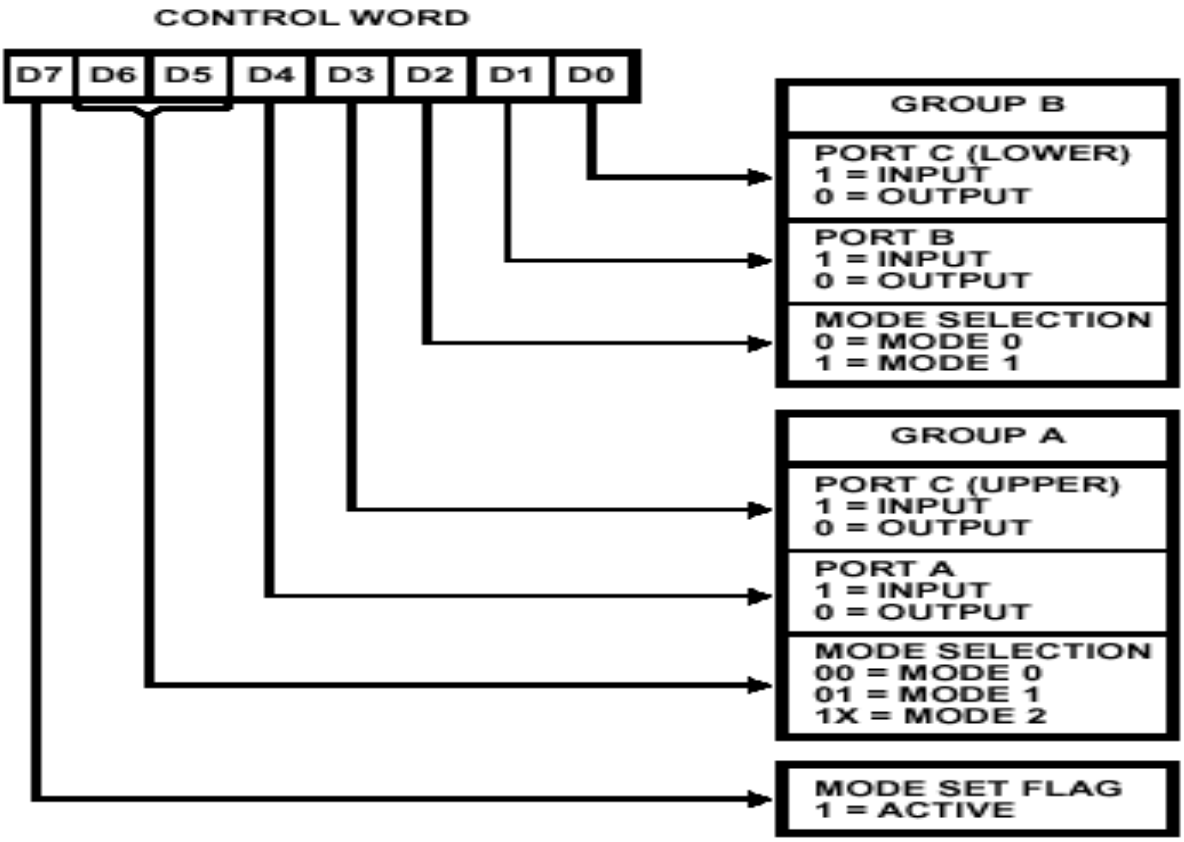
Ports A, B, and C

The 8255 contains three 8-bit ports (A, B, and C). All can be configured to 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 8255.

Port A One 8-bit data output latch/buffer and one 8-bit data input latch. Both "pull-up" and "pull-down" bus-hold devices are present on Port A.

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 output and status signal inputs in conjunction with ports A and B.



Examples and Basics: *Read these thoroughly*

1. If we want to take input and give output, then we can make any port (out of 3 ports) as input and any other port as output port. So, as an example, let us make **Port A as output and Port B as input.**

Now, fill the above table accordingly for this case. It will be 82h. Well, It's done like this.

;PORT A as output & PORT B as input

82 means → **8** **2**

1	0	0	0	0	0	1	0
D7	D6	D5	D4	D3	D2	D1	D0

always **1** for I/O mode	always **0** for mode 0	always **0** for mode 0	**0 →PORT A output**	0 →PORT C (upper) output	always **0** for mode 0	**1 →PORT B input**	0 →PORT C (lower) output

2. If we want to get 2 outputs, no input at all, then we can make any 2 ports as output ports. So let us make **Port B as output and Port C as output.**

Now, fill the table accordingly for this case. It will be 80h. It's done like this.

;PORT B as output & PORT C as output as well.

80 means → **8** **0**

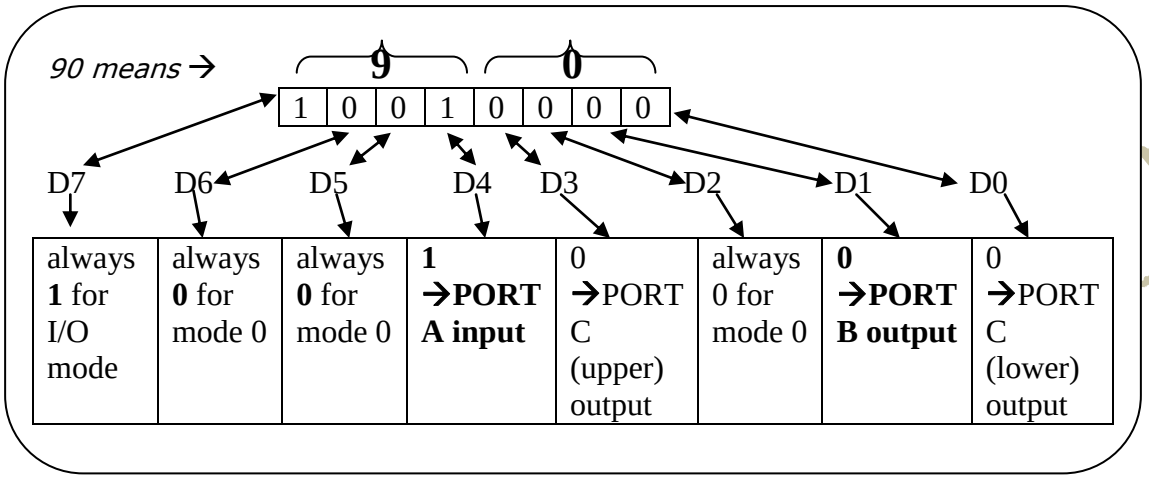
1	0	0	0	0	0	0	0
D7	D6	D5	D4	D3	D2	D1	D0

always **1** for I/O mode	always **0** for mode 0	always **0** for mode 0	0 →PORT A output	**0 →PORT C (upper) output**	always 0 for mode 0	**0 →PORT B output**	**0 →PORT C (lower) output**

3. Moving on, let us make **Port A as input and Port B as output.**

Now, fill the table accordingly for this case. It will be 90h. It's done like this.

; Port A as input and Port B as output.

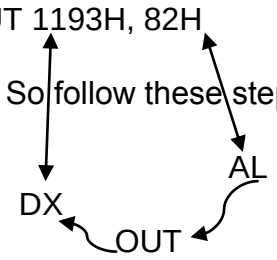


always **1** for I/O mode	always **0** for mode 0	always **0** for mode 0	**1 →PORT A input**	0 →PORT C (upper) output	always 0 for mode 0	**0 →PORT B output**	0 →PORT C (lower) output

All these are called as Control Word data. In order to use this, you must **initialize your 8255**. So the following 3 lines help you in initializing 8255 PPI.

Essentially, all you need to do is give this control word (82h or 80h or 90h or whatever) to the control word's address. Let's say that your control word is 82h and the address of it is 1193h. So you should give 82h to 1193h. Technically, you need to do this.

But, the problem is you can't give directly. So follow these steps.



Step1: **MOV AL, 82H**
Step2: **MOV DX, 1193H**
Step3: **OUT DX, AL**

Once you decide which port is what and initialize your 8255, next task is to take input from a port or/and give an output via some port which again depends on your requirement.

How to give input from a port?

Answer: **MOV DX, *port address***
IN AL, DX

E X A M P L E

```
.data
PB EQU 1191H
MOV DX, PB
IN AL, DX
```

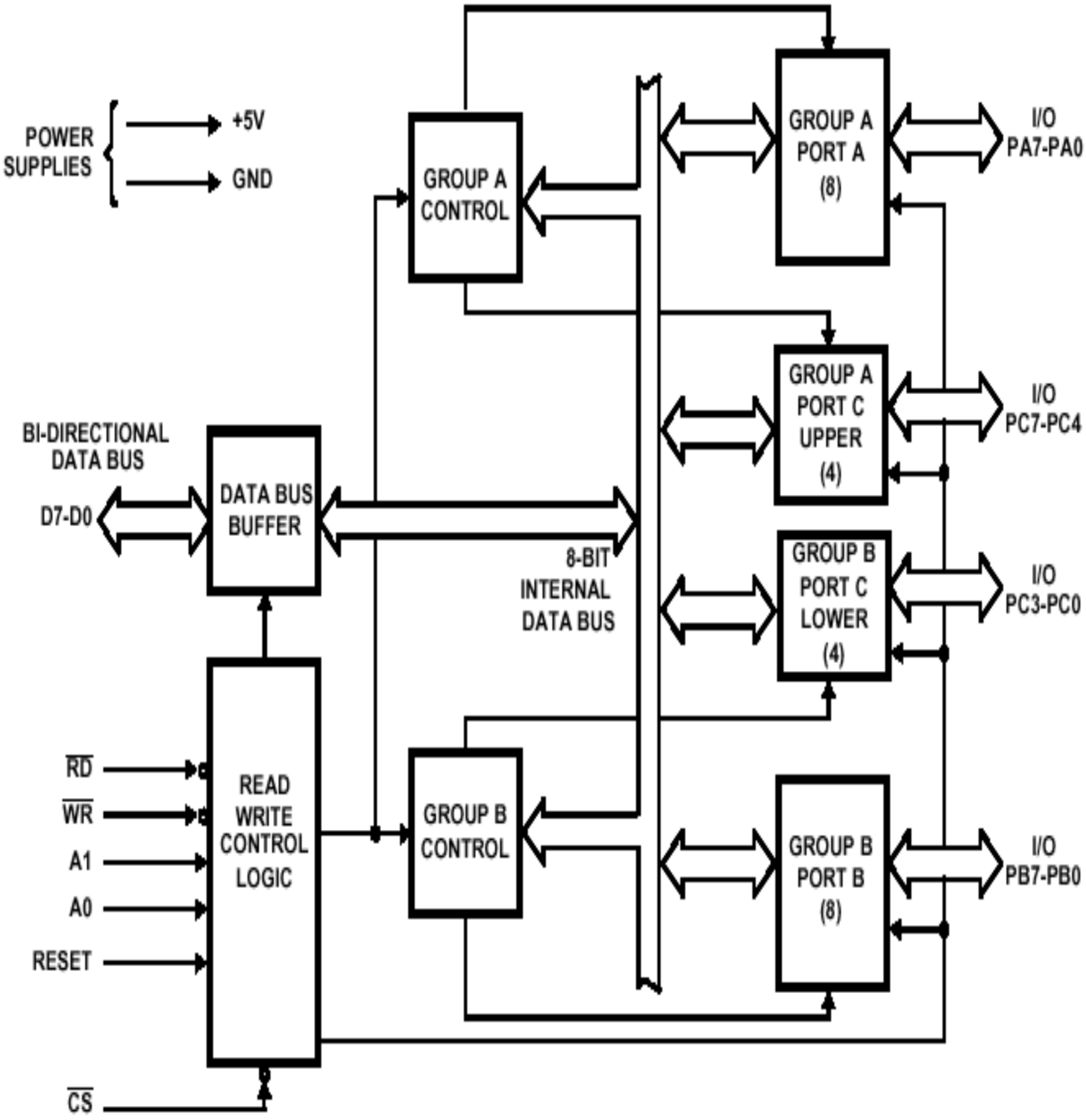
How to give output to a port?

Answer: **MOV DX, *port address***
OUT DX, AL

E X A M P L E

```
.data
PA EQU 1190H
MOV DX, PA
OUT AL, DX
```

8255 PPI Internal Architecture



MICROPROCESSORS LABORATORY
(Common to CSE & ISE)

Subject Code : 10CSL48	**I.A. Marks : 25**
Hours/Week : 03	**Exam Hours: 03**
Total Hours : 42	**Exam Marks: 50**

Notes:

- **Develop and execute the following programs using 8086 Assembly Language. Any suitable assemblerlikeASM,TASM etc may be used.**
- **Program should have suitable comments.**
- **The board layout and the circuit diagram of the interface are to be provided to the student during the examination.**

1. a) Search a key element in a list of 'n' 16-bit numbers using the Binary search algorithm.
   b) Read the status of eight input bits from the Logic Controller Interface and display 'FF' if it is the parity of the input read is even; otherwise display 00.

2. a) Write two ALP modules stored in two different files; one module is to read a character from the keyboard and the other one is to display a character. Use the above two modules to read a string of characters from the keyboard terminated by the carriage return and print the string on the display in the next line.

   b) Implement a BCD Up-Down Counter on the Logic Controller Interface.

3. a) Sort a given set of 'n' numbers in ascending order using the Bubble Sort algorithm.
   b) Read the status of two 8-bit inputs (X & Y) from the Logic Controller Interface and display X*Y.

4. a) Read an alphanumeric character and display its equivalent ASCII code at the center of the screen.
   b) Display messages FIRE and HELP alternately with flickering effects on a 7-segment display interface for a suitable period of time. Ensure a flashing rate that makes it easy to read both the messages (Examiner does not specify these delay values nor is it necessary for the student to compute these values).

5. a) Reverse a given string and check whether it is a palindrome or not.
   b) Assume any suitable message of 12 characters length and display it in the rolling fashion on a 7-segment display interface for a 30 suitable period of time. Ensure a flashing rate that makes it easy to read both the messages. (Examiner does not specify these delay values nor is it necessary for the student to compute these values).

6. a) Read two strings, store them in locations STR1 and STR2. Check whether they are equal or not and display appropriate messages. Also display the length of the stored strings.

b) Convert a 16-bit binary value (assumed to be an unsigned integer) to BCD and display it from left to right and right to left for specified number of times on a 7-segment display interface.

7. a) Read your name from the keyboard and display it at a specified location on the screen after the message "**What is your name?"** You must clear the entire screen before display.
b) Scan a 8 x 3 keypad for key closure and to store the code of the key pressed in a memory location or display on screen. Also display row and column numbers of the key pressed.

8. a) Compute **nCr** using recursive procedure. Assume that 'n' and 'r' are non-negative integers.
b) Drive a Stepper Motor interface to rotate the motor in specified direction (clockwise or counter-clockwise) by N steps (Direction and N are specified by the examiner). Introduce suitable delay between successive steps. (Any arbitrary value for the delay may be assumed by the student).

9. a) Read the current time from the system and display it in the standard format on the screen.
b) Generate the Sine Wave using DAC interface (The output of the DAC is to be displayed on the CRO).

10. a) Write a program to simulate a Decimal Up-counter to display 00-99.
b) Generate a Half Rectified Sine wave form using the DAC interface. (The output of the DAC is to be displayed on the CRO).

11. a) Read a pair of input co-ordinates in BCD and move the cursor to the specified location on the screen.
b) Generate a Fully Rectified Sine waveform using the DAC interface. (The output of the DAC is to be displayed on the CRO).

12. a) Write a program to create a file (input file) and to delete an existing file.
b) Drive an elevator interface in the following way:
i. Initially the elevator should be in the ground floor, with all requests in OFF state.
ii. When a request is made from a floor, the elevator should move to that floor, wait there for a couple of seconds (approximately), and then come down to ground floor and stop. If some requests occur during going up or coming down they should be ignored.

Note: In the examination *each* student picks one question from the lot of *all* 12 questions.

Microprocessor Hardware Interface Devices

LOGIC CONTROLLER INTERFACE (for 1b, 2b, 3b pgms)

THEORY

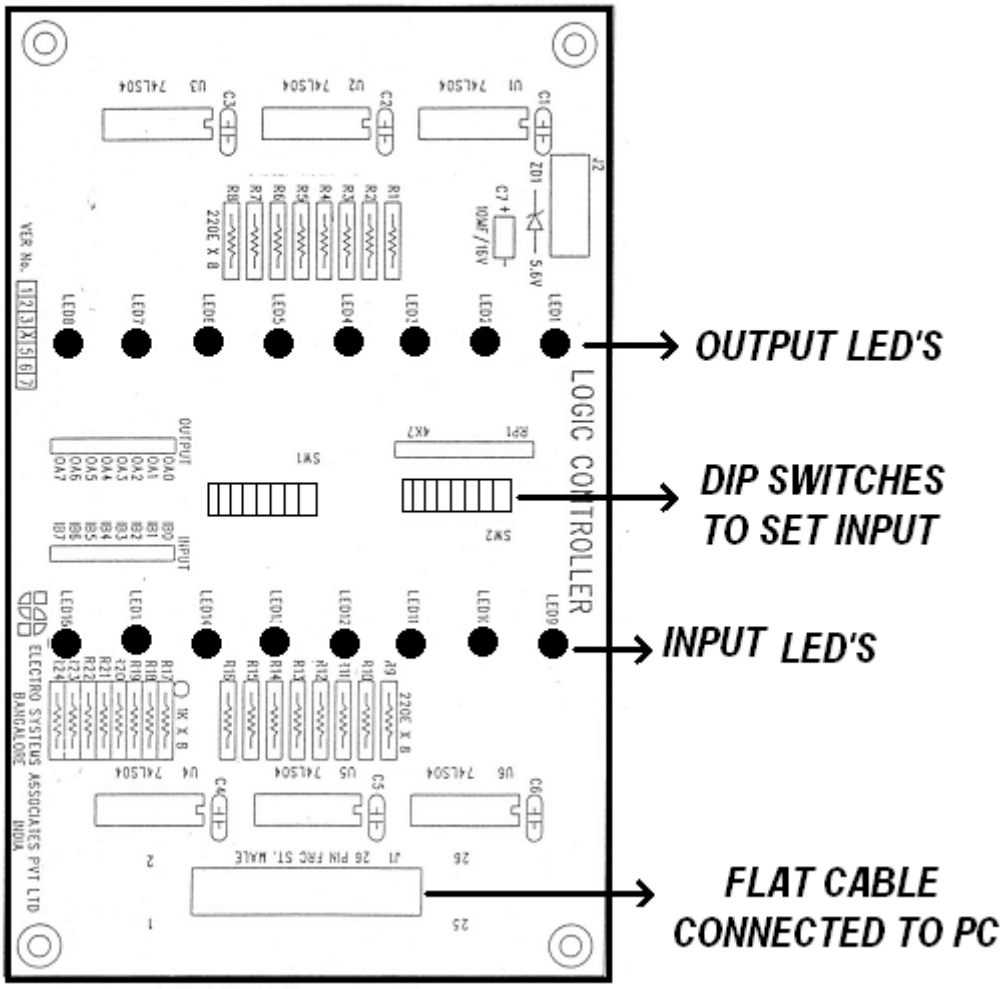
The interface consists of 8 TTL buffered outputs and 8 TTL buffered inputs. The logic state of each input/output is indicated by a corresponding LED (ON/OFF). The inputs can be read through PORT -B and the outputs can be controlled through PORT A. The inputs and outputs brought to 26-pin controller. Inputs LED's are controlled through Dipswitches.

This interface allows the user to perform experiments to understand some of the basic programming techniques involved in a logic controller. The software exercises could include combinational controllers, sequential controllers, programmable counters, etc.

OPERATION

Logic controller module is used as an input and output device. The 26-pin line plat ribbon cable (FRC) from the DIO card is connected to this module. Port A pins of 8255 are connected to 8 LEDs. When the Port A is high (1), than LED glows and when it is low (0), then LED is switched OFF. The port B-of 8255 is connected to logic controller toggle (DIP) switch. The toggle switch in turn is connected to an LED to indicate the state of the switch. When the switch is opened the Led is turned OFF and when switch is closed, the LED is ON.

LOGIC CONTROLLER HARDWARE MODULE DIAGRAM



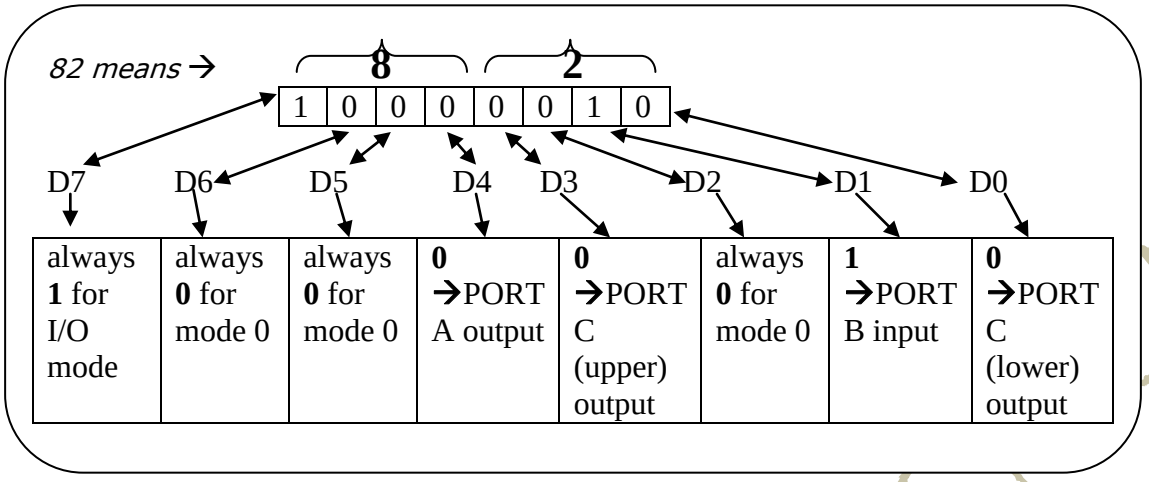
1.B) READ THE STATUS OF EIGHT INPUT BITS FROM THE LOGIC CONTROLLER INTERFACE AND DISPLAY 'FF' IF IT IS EVEN PARITY BITS OTHERWISE DISPLAY 00. ALSO DISPLAY NUMBER OF 1'S IN THE INPUT DATA.

```
.MODEL SMALL
INITDS MACRO
        MOV AX,@DATA        ;Initialization of data segment
        MOV DS,AX             (to access data)
ENDM

INIT8255 MACRO
        MOV AL,CW           ;Initialization of 8255 using control word
        MOV DX,CR           by passing 82h to control reg.
        OUT DX,AL            (to make port A as output & port B as input)

ENDM

EXIT MACRO
        MOV AH,4CH          ;Terminating the program
        INT 21H
ENDM
```

Common to all the programs

```
INPB MACRO
        MOV DX,PB           ; Initialization of PORT A as input
        IN AL,DX
ENDM
OUTPA MACRO
        MOV DX,PA           ; Initialization of PORT A as output
        OUT DX,AL
ENDM

.STACK
.DATA
        PA EQU 1190H
        PB EQU 1191H        ; Setting the port address for PORT A, PORT B & Control reg.
        CR EQU 1193H
```

```
CW DB 82H          ;82h is the value in control word 10000010, which makes
                   ;PORT  A as output & PORT B as input
```




```
.CODE
        INITDS          ;initializing data segment
        INIT8255        ;initializing 8255 processor
        INPB            ;reads input from logic controller through port B  in the form of
                        ;binary i.e. ex:00011100
        MOV BL,AL       ;take a copy to BL which we later use to check for parity
        MOV BH,0        ;counter
        MOV CX,8        ;8 bits of data
NEXTBIT:ROR AL,1        ;input is rotated right by one bit, if the carry is generated it
        JNC NEXT        ;means that bit is 1. So in that case keep incrementing BH
        INC BH
NEXT:   LOOP NEXTBIT    ;decrement CX by 1, if it's not zero, go to nextbit.
        MOV AL,BL       ;copy back from BL into AL.
        OR AL,AL        ;after OR operation it effects the parity flag in FLAG reg
        JPE DISPFF      ;(JPE- jump if parity even) if even parity, then jump to dispff
        MOV AL,00H      ;if the parity is odd, you should turn OFF the lights. So make
                        ; AL to hold zero (all 0's – eight 0's)
DISP:   OUTPA           ;Resultant 00 or ff is displayed on logic controller module in the
                        ;form of LED output
        CALL DELAY      ;see the output. By the time you are seeing, let it do something
        ADD BH,30H      ;convert your counter BH to ASCII to display
        MOV DL,BH       ;To display a single character, you need to have the printable
        MOV AH,2                character's ASCII value in DL. Then fill 2 in AH, Interrupt
        INT 21H                 number should be 21H.
```

```
        EXIT                    ;you are done with everything, so just exit.
DISPFF: MOV AL,0FFH             ; if the parity is even, you should turn ON the lights. So make
                                ; AL to hold 0FF (all 1's – eight 1's)
        JMP DISP

DELAY PROC
        MOV AX,07FFH            ;Delay can be increased or decreased according to our
B2:     MOV CX,0FFFFH           requirement
B1:     LOOP B1                 ;Decrement CX until CX becomes 0.
        DEC AX                  ;Now decrement AX, go to B2, fill cx again with a huge value
        JNZ B2                  ;do this till AX becomes 0.
        RET                     ;go back to wherever you called delay procedure.
DELAY ENDP
END
```

OUTPUT: Input is set from switches in the module & output is observed on the module. The counter of 1's can be observed in the screen.

2. B) PERFORM THE FOLLOWING FUNCTIONS USING THE LOGIC CONTROLLER INTERFACE.
1. BCD UP COUNTER 2. BCD DOWN COUNTER.

```
.MODEL SMALL

INITDS MACRO
        MOV AX,@DATA        ;  Initialization of data segment
        MOV DS,AX              (to access data)
ENDM


 INIT8255 MACRO
       MOV AL,CW            ; Initialization of 8255 using control word
       MOV DX,CR            by passing 82h to control reg.
       OUT DX,AL             (to make port A as output & port B as input)

ENDM


OUTPA MACRO
       MOV DX,PA            ; Initialization of PORT A as output
       OUT DX,AL
ENDM

PRINTF MACRO MSG
       MOV AH,9
       LEA DX,MSG
       INT 21H
ENDM

EXIT MACRO
       MOV AH,4CH           ;Terminating the program
       INT 21H
ENDM

.STACK
.DATA
```

```
        PA EQU 1190H        ; Setting the port address for PORT A, PORT B & Control reg.
        CR EQU 1193H
        CW DB 82H
        SELECT DB 10,13,"SELECT 1: UP COUNTER       2: RING COUNTER $"
        EXITMSG DB 10,13,"PRESS ANY KEY TO EXIT $"

.CODE
        INITDS                  ; initialize data segment
        INIT8255                ; initialize 8255
        PRINTF SELECT           ; print the choice
        MOV AH,1                ; input a single key. Gets stored in AL automatically
        INT 21H
        CMP AL,'1'              ; if your input is 1, go to upcounter
        JE UPCOUNTER
        CMP AL,'2'              ; if your input is 2, go to downcounter
        JE DOWNCOUNTER
        EXIT                    ; well, upon any other input, just exit.

UPCOUNTER: MOV AL,0             ; initial value of up counter is 0
    UP: OUTPA                   ; this display the contents of Al on the logic module
        CALL DELAY              ; have some delay
        CALL KBHIT              ; if you press any key, then exit.
        ADD AL,1                ; adding 1 to AL by this it increments the count
        DAA                     ; DAA-decimal adjust after addition
        CMP AL,99H              ; compares with 99 in order to count till 99
        JNE UP                  ; upon adding 1, if not equal to 99, go to up
        EXIT                    ; if it crosses 99, exit.

DOWNCOUNTER: MOV AL,99H         ; initial value of down counter is 99

  DOWN: OUTPA                   ; down counter starts
        CALL DELAY              ; have some delay
        CALL KBHIT              ; if you press any key, then exit.
        SUB AL,1                ; by subtracting 1, it will decrement the count
        DAS                     ; DAA-decimal adjust after SUBTRACTION
        CMP AL,0                ; compares with 0 in order to count till 0
```

```
        JNE DOWN                ; upon subtracting 1, if not equal to 0, go to down
        EXIT                    ; if it crosses 0, exit.

    DELAY PROC
        MOV BX,04FFH
    B2: MOV CX,0FFFFH
    B1: LOOP B1
        DEC BX
        JNZ B2
        RET
    DELAY ENDP

    KBHIT PROC
        PUSH AX                 ;save your precious AX value
        MOV AH,1                ;checks if any key is pressed in between the count
        INT 16H                 ;if you press any key, it becomes non-zero. So go to
        JNZ DONE                    done and exit.
        POP AX                  ;if you don't press any key, it becomes zero. So take out
                                    your precious value and return.
        RET
    DONE: EXIT                  ;so you have pressed a key, go to exit.
    KBHIT ENDP
      END*********************************************
```

OUTPUT:

-g

Select choice 1 or 2 for BCD or ring counter

Corresponding choice output is observed on the module

NOTE:

DAA means Decimal Adjust after Addition. Let's say AL=0. Keep adding 1 to it. It becomes 1,2,3,,,,9, after 9 it becomes A. But we don't want A, we want 10. So do DAA once it crosses 9. So it will now be 10. Now keep adding 1 to it. It becomes 11,12,13,14,,,,,19, after 19 it becomes 1A. So do DAA now because we don't want 1A, we want 20. So in a nutshell, once AL crosses 9, adjust the decimal after addition i.e. do DAA.

Similarly, **DAS** – Decimal adjust after Subtraction.

3. B) READ THE STATUS OF TWO 8-BIT INPUTS (X & Y) FROM THE LOGIC CONTROLLER INTERFACE AND DISPLAY X*Y.

```
.MODEL SMALL
 INITDS MACRO
        MOV AX,@DATA        ; Initialization of data segment
        MOV DS,AX             (to access data)
ENDM

  INIT8255 MACRO
        MOV AL,CW           ; Initialization of 8255 using control word
        MOV DX,CR           by passing 82h to control reg.
        OUT DX,AL            (to make port A as output & port B as input)

ENDM

INPB MACRO
        MOV DX,PB           ; Initialization of PORT B as input
        IN AL,DX
ENDM

OUTPA MACRO
        MOV DX,PA           ; Initialization of PORT A as output
        OUT DX,AL
ENDM
PRINTF MACRO MSG
        MOV AH,9
        LEA DX,MSG
        INT 21H
ENDM

EXIT MACRO
        MOV AH,4CH          ;Terminating the program
        INT 21H
ENDM
```

```
.DATA
        ASKX DB 10,13,"SET VALUE FOR X,      THEN PRESS ANY KEY $"
        ASKY DB 10,13,"SET VALUE FOR Y,      THEN PRESS ANY KEY $"

        PA EQU 1190H
        PB EQU 1191H        ;Setting the port address for PORT A ,PORT B & Control reg.
        CR EQU 1193H
        CW DB 82H
.CODE
        INITDS
        INIT8255

        PRINTF ASKX         ; ASK X
        MOV AH,8            ; reads the input without any echo(displaying)
        INT 21H
        INPB                ; reads 1st value i.e. X, which is set through logic controller
                              Module, value will be automatically stored in AL
        MOV BL,AL           ; contents of AL is copied to BL in order to store the value

        PRINTF ASKY         ; ASK Y
        MOV AH,8            ; reads the input without any echo(displaying)
        INT 21H
        INPB                ; reads 2nd value i.e. Y, which is set through logic controller
                              Module, value will be automatically stored in AL

        MUL BL              ; the contents of AL is multiplied with contents of BL
        OUTPA               ; the result of multiplication is stored in AX reg i.e. AL wil be
        CALL DELAY            having first 8 bit result. It is displayed first on the output
        MOV AL,AH             module, after some delay rest 8 bits which are in AH is copied
        OUTPA                 to AL and is displayed on the module
        EXIT

DELAY PROC
        MOV BX,0FFFH
B2:     MOV CX,0FFFFH
B1:     LOOP B1
```

```
        DEC BX
        JNZ B2
        RET
DELAY ENDP
END
```

OUTPUT: INPUTS X & Y IS SET THROUGH MONDULE AND OUTPUT IS OBSERVED

SEVEN SEGMENT DISPLAY (for 4b, 5b, 6b pgms)

INTRODUCTION:

This interface provides a four digit 7-segmcnt display driven by the output of four cascaded shift registers. Data to be displayed is transmitted serially (bit by bit) to the interface. Each bit is clocked into the shift register by providing a common clock through a port line (PC4 in ALS module). Thus one port line PB0 provides the data.

This interface allows the user to study seven segment display control technique, code conversion methods, etc. the software exercise could include procedures for table lookup, a variety of bit manipulation operations, etc.

CIRCUIT DECSRIPTION:

This interface allows display of up to 4 digits. The technique adopted uses four 8 bit serial in parallel out (SIPO) shift registers. The 8 outputs of the shift register arc connected to the seven-segment display through register. Each character is represented by an 8 bit 5 code and shifting is done by applying the MSB of the code corresponding to the last digit on the right to the data input of the 1st shift register and applying a clock pulse. The next most significant bit follows this till all the bits for that digit are exhausted. The MSB of the next digit from the right is then fed to the data input and this process is continued till all the digits are displayed. A total of 32 clock pulses are required to display the four digits.

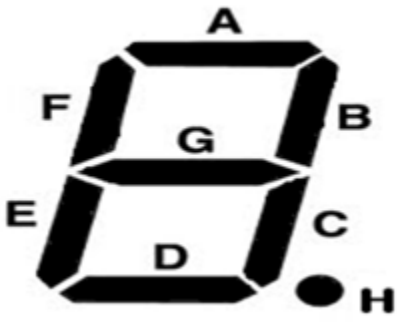
The code corresponding to the four digits can be stored in consecutive locations in RAM to be accessed by the output routine. A look up-table is used to convert these characters to their corresponding output code. The codes for the characters 0 to 9 and A to F are given in the table. With this scheme it is possible to output any special characters as the user can very easily write the corresponding output code.

This interface uses MAN72 Common Anode seven-segment display. Hence low (0) inputs must be given to each segment to glow (ON) and high to blank (OFF). The circuit diagram of the seven-segment display interface is provided at the beginning.

Design of Seven-Segment Code (SSC)

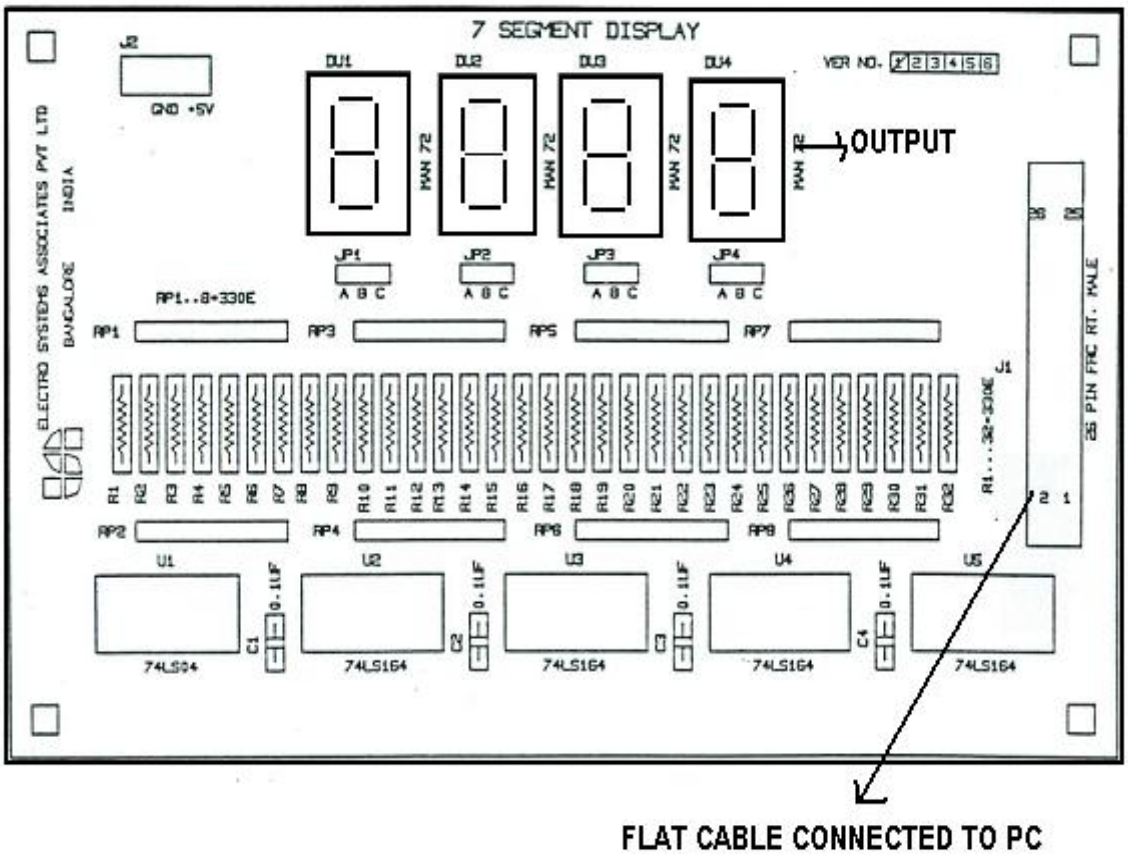
O=ON, 1=OFF.
Each display has seven segment and a dot (a, b, c, d, c, f, g and h). It is as shown in fig. For displaying any character, the corresponding segment must be given low (0) inputs.
Seven Segment Code to display Hex-digits are given in the table.

Fig: Seven Segment Display



Angka	PC.7	PC.6	PC.5	PC.4	PC.3	PC.2	PC.1	PC.0	Hex
	h	g	f	e	d	c	b	a	
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	0	0	0	0	90

7-SEGMENT DISPLAY HARDWARE MODULE DIAGRAM



4.B) DISPLAY MESSAGES FIRE AND HELP ALTERNATELY WITH FLICKERING EFFECTS ON A 7-SEGMENT DISPLAY INTERFACE FOR A SUITABLE PERIOD OF TIME. ENSURE A FLASHING RATE THAT MAKES IT EASY TO READ BOTH THE MESSAGES (EXAMINER DOES NOT SPECIFY THESE DELAY VALUES NOR IT IS NECESSARY FOR THE STUDENT TO COMPUTE THESE VALUES).

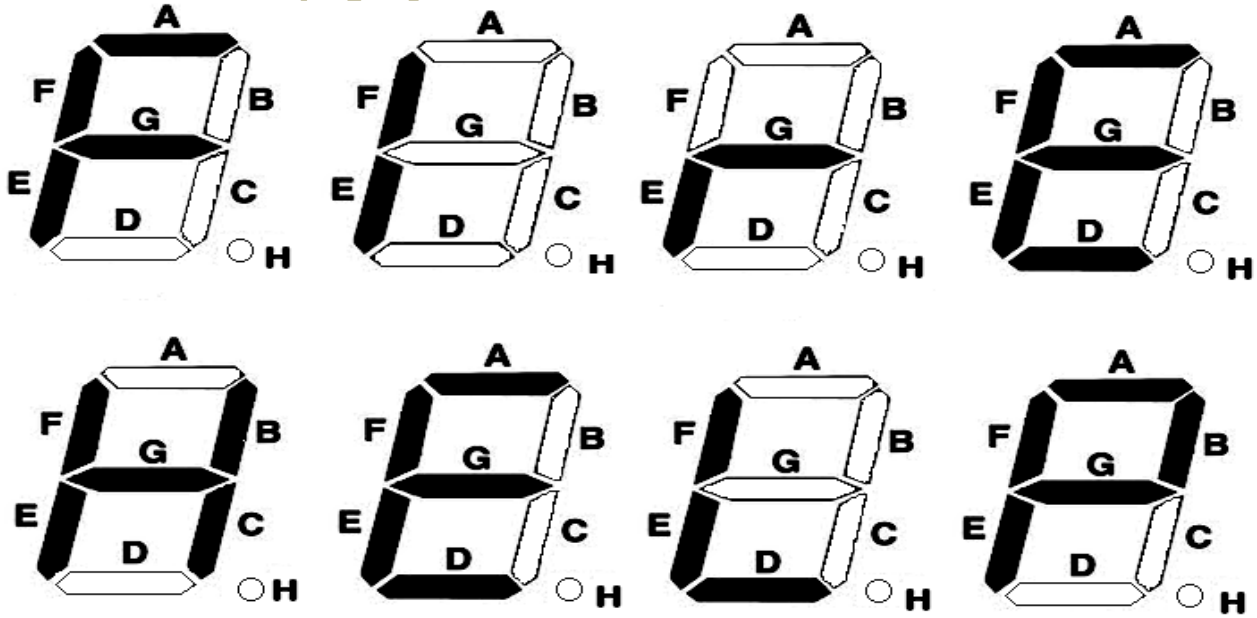
```
.MODEL SMALL
 INITDS MACRO
      MOV AX,@DATA        ; Initialization of data segment
      MOV DS,AX             (to access data)
      ENDM

 INIT8255 MACRO
      MOV AL,CW           ; Initialization of 8255 using control word
      MOV DX,CR           by passing 80h to control reg.
      OUT DX,AL           (to make port B as output & port C as output)
ENDM                                  to display    for clock
OUTPB MACRO
      MOV DX,PB           ; Initialization of PORT B as output
      OUT DX,AL
ENDM
OUTPC MACRO
      MOV DX,PC           ; Initialization of PORT C as output
      OUT DX,AL
ENDM
PRINTF MACRO MSG
      MOV AH,9
      LEA DX,MSG
      INT 21H
ENDM
EXIT MACRO
      MOV AH,4CH          ;Terminating the program
     INT 21H
ENDM

.MODEL SMALL
.STACK
```

```
.DATA
        PB EQU 1191H
        PC EQU 1192H        ;Setting the port address for PORT A ,PORAT C & Control reg.
        CR EQU 1193H
        CW DB 80H           ; 80h is the value in control word 10000000, which makes
                            ;PORT B as output & PORT C as out put

        M1 DB 10,13,"PRESS ANY KEY TO EXIT $"

                ; F     I     r   E
        FIRE DB 8Eh,0CFh,0AFh,86h       ; table FIRE & HELP contains hex
                                            decimal values for FIRE & HELP to display
                ; H     E     L     P     on 7-segment display module. You can
        HELP DB 89H,86H,0C7H,8CH            even try displaying your name with hexa
                                            decimal values .

.CODE
        INITDS
        INIT8255
        PRINTF M1           ; Displays M1
START:  LEA SI,FIRE         ; loads the address of FIRE to SI
        CALL DISP_MSG       ; Displays the contents of table form FIRE
        CALL DELAY
        LEA SI,HELP         ; loads the address of HELP to SI
        CALL DISP_MSG       ; Displays the contents of table form HELP
        CALL DELAY
        MOV AH,1            ; Checks if any key from key board is pressed
        INT 16H
        JZ START
        EXIT                ; terminate program


DISP_MSG PROC               ; displaying char starts from this proc
        MOV CX,4            ; count is taken 4 b'coz of 4 char in first string i.e. FIRE
NEXT_CHAR: MOV BL,8         ; BL indicates 8 bits in single char
        MOV AL,[SI]         ; char is moved to AL from SI which is in the form of 8-bit data
NXTBIT: ROL AL,1            ; rotate left will sends data out bit by bit
        OUTPB               ; sends bit by bit to output module
        PUSH AX             ; keeps copy of AX in stack b'coz next instruction changes it.
```

As capital R cannot be displayed we are considering small r (if you want R, go ahead in terms of A - 88H)

```
        MOV AL,00H        ; clock pulse 0 given to drive the bits on led through Port C
        OUTPC
        MOV AL,11H        ; clock pulse 1 given to drive the bits on led through Port C
        OUTPC
        POP AX            ; copy is retrieved from stack
        DEC BL            ; decrements the bit count
        JNZ NXTBIT        ; repeats until bit count becomes to 0
        INC SI            ; SI is pointed to next char
        LOOP NEXT_CHAR    ; automatically decrements char count (CX)
        RET               ; returns the control to called instruction
DISP_MSG ENDP

DELAY PROC
        MOV BX,0FFFH
B2:     MOV CX,0FFFFH
B1:     LOOP  B1
        DEC BX
        JNZ B2
        RET
DELAY ENDP
END
```

OUTPUT ON THE MODULE:



5.B) ASSUME ANY SUITABLE MESSAGE OF 12 CHARACTERS LENGTH AND DISPLAY IT IN THE ROLLING FASHION ON A 7-SEGMENT DISPLAY INTERFACE FOR A SUITABLE PERIOD OF TIME. ENSURE A FLASHING RATE THAT MAKES IT EASY TO READ BOTH THE MESSAGES. (EXAMINER DOES NOT SPECIFY THESE DELAY VALUES NOR IT IS NECESSARY FOR THE STUDENT TO COMPUTE THESE VALUES).

```
.MODEL SMALL
INITDS MACRO
        MOV AX,@DATA        ; Initialization of data segment
        MOV DS,AX             (to access data)
        ENDM

 INIT8255 MACRO
        MOV AL,CW           ; Initialization of 8255 using control word
        MOV DX,CR           by passing 80h to control reg.
        OUT DX,AL            (to make port B as output & port C as output)

ENDM                                          to display   for clock

OUTPB MACRO
        MOV DX,PB           ; Initialization of PORT B as output
        OUT DX,AL
ENDM

OUTPC MACRO
        MOV DX,PC           ; Initialization of PORT C as output
        OUT DX,AL
ENDM

PRINTF MACRO MSG
        MOV AH,9
        LEA DX,MSG
        INT 21H
ENDM

EXIT MACRO
        MOV AH,4CH          ;Terminating the program
        INT 21H
ENDM
```

```
.DATA
        PB EQU 1191H
        PC EQU 1192H        ;Setting the port address for PORT B ,PORAT C & Control reg.
        CR EQU 1193H
        CW DB 80H
        M1 DB 10,13,"PRESS ANY KEY TO EXIT $"

                ; B   A  9  8  7  6  5  4  3  2  1  0
                 ↓   ↓  ↓  ↓  ↓  ↓  ↓  ↓  ↓  ↓  ↓  ↓
        TXT DB 80H,88H,90H,80H,0F8H,82H,92H,99H,0B0H,0A4H,0F9H,0C0H

                ; Above table contains hex decimal value of 12 char from 0 to 9 & A,B
.CODE
        INITDS
        INIT8255
        PRINTF M1

START:  LEA SI,TXT          ; 12 char TXT address is loaded to SI
        MOV CX,12           ;count is taken as 12 b'coz of 12 chars in TXT
NEXT_CHAR:MOV AL,[SI]       ; the content of SI is loaded to AL in order to display on module
        CALL DISP_CHAR      ; control is moved to displaying char after display it comes back
        CALL DELAY
        CALL KBHIT          ; checks if any key is pressed
        INC SI              ; SI is pointed to next Char
        LOOP NEXT_CHAR      ; loop repeats until CX becomes 0 i.e. all the 12 char displayed
        JMP START

KBHIT PROC
        MOV AH,1
        INT 16H             ; to check if any key is pressed
        JNZ DONE
        RET                 ;if you haven't pressed any key, then return
DONE: EXIT                  ;now that you pressed any key, just exit
KBHIT ENDP
```

Why 12? b'coz your syllabus says to take any 12 characters.

```
DISP_CHAR PROC      ;display procedure starts
      MOV BL,8      ; count is taken as 8 b'coz of 8 bits in a single char
NXTBIT: ROL AL,1    ; rotate left by 1 bit ,contents of AL is shifted & sent out
      OUTPB         ; bits are sent out through PORT B
      PUSH AX       ; copy of AX is kept in stack b'coz next instruction changes it.
      MOV AL,00H    ; clock pulse 0 given to drive the bits on led through Port C
      OUTPC
      MOV AL,11H    ; clock pulse 1 given to drive the bits on led through Port C
      OUTPC
      POP AX        ; copy of AX is retrieved from Stack
      DEC BL        ; count of 8-bits are decremented
      JNZ NXTBIT    ; loop repeats until all the 8-bits are displayed
      RET
DISP_CHAR ENDP

DELAY PROC
      MOV BX,07FFH
B2:   MOV DI,0FFFFH
B1:   DEC DI
      JNZ B1
      DEC BX
      JNZ B2
      RET
DELAY ENDP
END
```

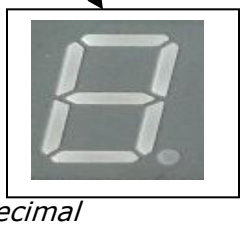
OUTPUT:

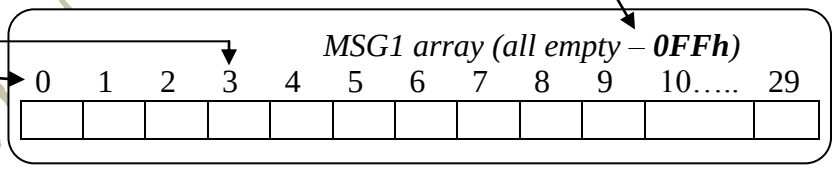
All the 12 char 0 to 9 & AB is displayed on the 7-segmant display it is shifted from right to left

6.B) CONVERT A 16-BIT BINARY VALUE (ASSUMED TO BE AN UNSIGNED INTEGER) TO BCD AND DISPLAY IT FROM LEFT TO RIGHT AND RIGHT TO LEFT FOR SPECIFIED NUMBER OF TIMES ON A 7-SEGMENT DISPLAY INTERFACE.

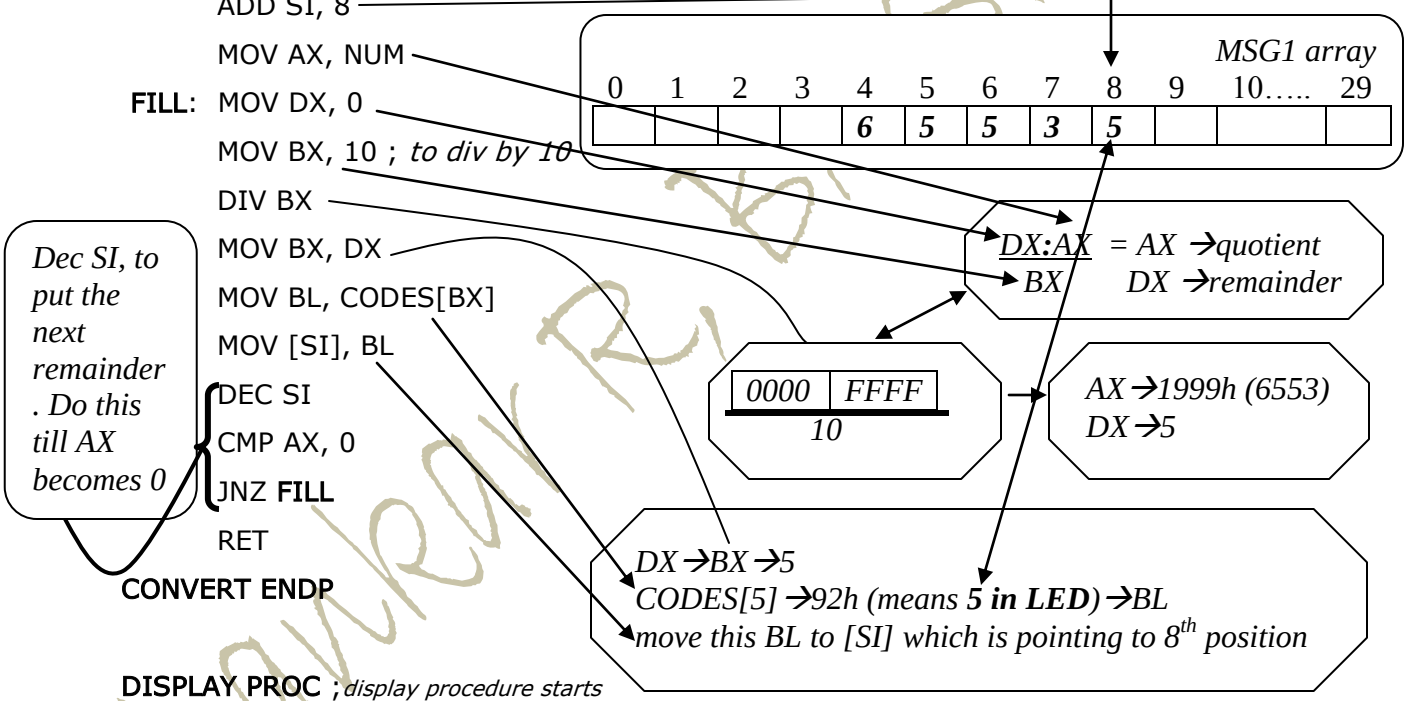
```
.MODEL SMALL
INITDS MACRO
        MOV AX,@DATA        ; Initialization of data segment
        MOV DS,AX             (to access data)
ENDM

INIT8255 MACRO
        MOV AL,CW           ; Initialization of 8255 using control word
        MOV DX,CR           by passing 80h to control reg.
        OUT DX,AL           (to make port B as output & port C as output)
                                          to display   for clock
ENDM

OUTPB MACRO
        MOV DX,PB           ; Initialization of PORT B as output
        OUT DX,AL
ENDM

OUTPC MACRO
        MOV DX,PC           ; Initialization of PORT C as output
        OUT DX,AL
ENDM

PRINTF MACRO MSG
        MOV AH,9
        LEA DX,MSG
        INT 21H
ENDM

EXIT MACRO
        MOV AH,4CH          ;Terminating the program
        INT 21H
ENDM
```

```
.DATA
        PB EQU 1191H
        PC EQU 1192H        ;Setting the port address for PORT B ,PORT C & Control reg.
        CR EQU 1193H
        CW DB 80H           ; 80h to make port B as output & port C as output

            ; 0    1    2    3    4    5    6    7    8    9

CODES DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H

  MSG1 DB 30 DUP (0FFh)  ;duplicate all 30 bytes with EMPTY character (0FFh means empty)

  NUM DW 0FFFFh         ; 0FFFFh is hexa decimal value for which we have to find decimal equivalent

.CODE
        INITDS
        INIT8255

        CALL CONVERT        ; call procedure to convert num (hexa) to decimal
        MOV CX, 11          ; number of times to display (you can change this number)
        LEA SI, MSG1        ; string where display codes are stored
        ADD SI, 3           ;

AGAIN: PUSH CX   ; save CX
          PUSH SI    ; save SI
          MOV CX, 4         ; to display four display codes
NEXTBIT:CALL DISPLAY        ; call display procedure to display 1 display code left wise
          DEC SI          ; so you have filled 1 character, now dec SI to fill the next character
          DEC CX            ; 1 character filling is done, fill the rest
          JNZ NEXTBIT       ; for all CX (4) bits
          CALL DELAY        ; introduce delay after four codes are sent to interface
          POP SI            ; take out your SI from stack
          POP CX            ; take out your CX from stack
          INC SI            ; point to next
          DEC CX            ; remaining CX number of times
          JNZ AGAIN
```





```
        MOV CX, 11      ;so we are done with left to right. Now let's do all over again from right to left
        LEA SI, MSG1    ; point SI again to MSG1
        ADD SI, 10      ; point to last (coz we have to display from right to left)
NEXTBIT1: CALL DISPLAY  ; call display procedure to display 1 display code right wise
        CALL DELAY      ; introduce delay after one display code sent
        DEC SI          ; you are coming from right to left
        DEC CX          ; remaining locations
        JNZ NEXTBIT1

        EXIT    ; you are done with everything (left to right & right to left), so just exit.

CONVERT PROC            ; Hexa to Decimal conversion starts
        ADD SI, 8
        MOV AX, NUM
FILL:   MOV DX, 0
        MOV BX, 10 ; to div by 10
        DIV BX
        MOV BX, DX
        MOV BL, CODES[BX]
        MOV [SI], BL
        DEC SI
        CMP AX, 0
        JNZ FILL
        RET
CONVERT ENDP

DISPLAY PROC ;display procedure starts
        MOV BL, 8       ; count is taken as 8 b'coz of 8 bits in a single char
        MOV AL, [SI]    ; store display code to al
NXT_BIT: ROL AL, 1      ; rotate left by 1 bit, contents of AL is shifted & sent out
        OUTPB           ; bits are sent out through PORT B
        PUSH AX         ; copy of AX is kept in stack b'coz next instruction changes it.
        MOV AL,00H      ; clock pulse 0 given to drive the bits on led through Port C
        OUTPC
        MOV AL,11H      ; clock pulse 1 given to drive the bits on led through Port C
        OUTPC
```



```
        POP AX          ; copy of AX is retrieved from Stack
        DEC BL          ; count of 8-bits are decremented
        JNZ NXT_BIT     ; loop repeats until all the 8-bits are displayed
        RET
DISPLAY ENDP

DELAY PROC
        PUSH CX
        PUSH BX
        MOV CX, 055FFH
OUTER:  MOV BX, 0FFFFH
INNER:  DEC BX
        JNZ INNER       ; repeat inner loop 0ffff times
        LOOP OUTER      ; repeat outer loop 55ffh times
        POP BX
        POP CX
        RET
DELAY ENDP
```

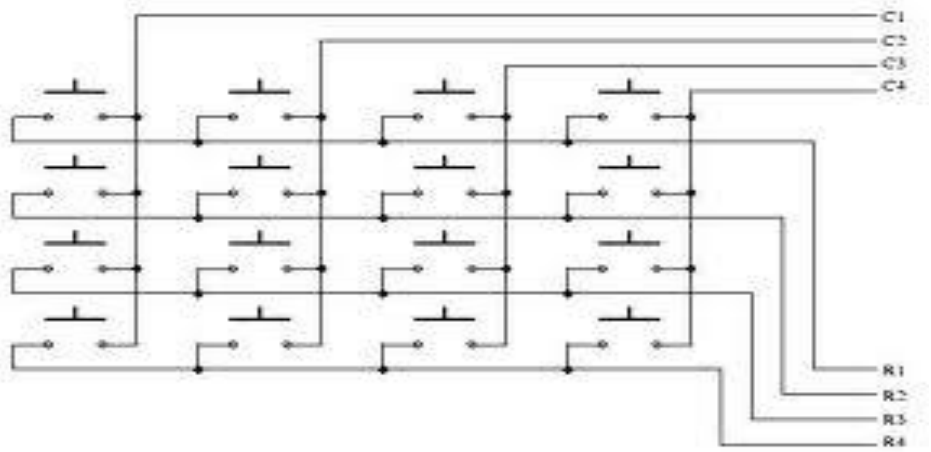
OUTPUT: *Output is 65535 for 0FFFFh. It is displayed on 7 segment display from left to right and from right to left.*

KEYBOARD INTERFACE (for 7b)

INTRODUCTION

This interface provides a calculator type keyboard consisting of the keys 0 to 9,+,-,x,l,%,AC,CE,CIIK,=.1\IC,MR,M,tvl+. These keys arranged in 3x8 matrixes. The row lines are driven by PC0, PC1 and PC2. The columns lines arc read through port A. When no key is pressed, all the return lines (col) are low. The rows are driven high one after another in sequence. When a row is driven high, pressing a key in that row causes the corresponding return lines (col) to be read as high. Then it scan for the column for which key is depressed. The row and column position can then be used to encode the key. As the scanning of the rows occurs at very high speed compared to human reaction times, there is no danger of missing a key depression. Further issues like **debouncing**, etc have to be handled through appropriate software routines.

The interface module has a 26-pin connector at one edge of the card. This is used for connecting the inter bee to the IBM-PC through DIO card with a flat cable connector set. The +5 DC power required by this interface is drawn from the IBM-PC via the flat cable connector set.

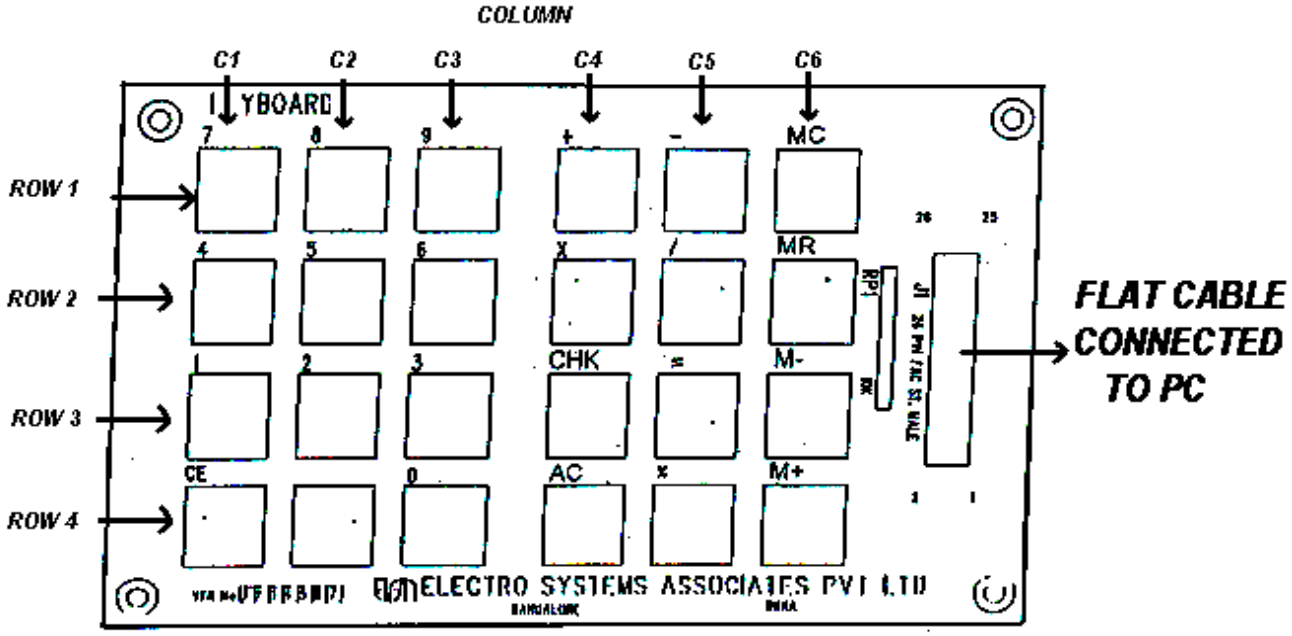


The TABLE 1 gives the corresponding HEX value for the keys.

TABLE 1

LABLE ON THE KEYTOP	HEX CODE	LABLE ON THE KEYTOP	HEX CODE
0	0	-	0C
1	1	X	0D
2	2	/	0E
3	3	/	0F
4	4	AC	10
5	5	CE	11
6	6	CHK	12
7	7	=	13
8	8	MC	14
9	9	MR	15
.	0A	M	16
+	0B	M+	17

KEY PAD INTERFACE HARDWARE MODULE DIAGRAM



7.B) SCAN A 8 X 3 KEYPAD FOR KEY CLOSURE AND TO STORE THE CODE OF THE KEY PRESSED IN A MEMORY LOCATION OR DISPLAY ON SCREEN. ALSO DISPLAY ROW AND COLUMN NUMBERS OF THE KEY PRESSED.

```
.MODEL SMALL
INITDS MACRO
        MOV AX,@DATA            ;initialization of data segment
        MOV DS,AX                  ;(to access data)
ENDM

INIT8255 MACRO
        MOV AL,CW                ;initialization of 8255 using control word
        MOV DX,CR               ;  by passing 90h to control reg.
        OUT DX,AL                ; (to make port A as input & port C as output)

ENDM                                  for reading key          for activating columns

OUTPC MACRO
        MOV DX,PC
        OUT DX,AL
ENDM

INPA MACRO
        MOV DX,PA
        IN AL,DX
ENDM

PRINTF MACRO MSG
        MOV AH,9
        LEA DX,MSG
        INT 21H
ENDM
EXIT MACRO
        MOV AH,4CH              ;terminating the program
        INT 21H
ENDM
```

```
.DATA
    DISPLAYABLECHARACTER DB '0123456789.+-*/%CEK=MRSA'
    KEYPRESSEDMSG DB 'KEY PRESSED IS : '        ; no $ here
    KEY DB ? ,10,13,'$'
    ROWMSG DB 'THE ROW NO IS : '              ; no $ here
    ROW DB ? ,10,13,'$'
    COLMSG DB 'THE COLUMN NO IS : '           ; no $ here
    COL DB ? ,'$'

    CR EQU 01193H
    PC EQU 01192H
    PA EQU 01190H
    CW DB 90H

.CODE
    INITDS
    INIT8255

REPEAT:                          ; ACTIVATING/EXCITING 1st COL
    MOV AL,01H                   ; 01(0001) for first col
    OUTPC                        ;checks if key is pressed in first col

    INPA                         ;reads the pressed input key in port A & stores it in AL
    CMP AL ,00                   ;if key is not pressed, AL will be 00 in this 1st column.
JNZ FIRSTCOL          ;if key is pressed in this 1st col, then goto FIRSTCOL, else goto next line
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                 ; ACTIVATING/EXCITING 2nd COL
    MOV AL,02H                   ; 02(0010) for second col
    OUTPC                        ;checks if key is pressed in second col

    INPA                         ;reads the pressed input key in port A & stores it in AL
    CMP AL ,00                   ;if key is not pressed, AL will be 00 in this 2nd column.
JNZ SECONDCOL    ;if key is pressed in this 2nd col, then goto SECONDCOL, else goto next line
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
                                ; ACTIVATING/EXCITING 3rd COL
        MOV AL,04               ; 04(0100) for third col
        OUTPC                   ;checks if key is pressed in third col

        INPA                    ;reads the pressed input key in port A & stores it in AL
        CMP AL ,00              ;if key is not pressed AL will be 00 in this 3rd column.
JNZ THIRDCOL        ;if key is pressed in this 3rd col, then goto THIRDCOL, else goto next line
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        JMP REPEAT      ; well, if you don't press any key, then repeat the entire task
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

FIRSTCOL: LEA SI,DISPLAYABLECHARACTER  ;SI points to displayablecharacter array
        MOV COL,31H             ;control has come here means, the key that we have pressed
                                  happens to be in first column
        MOV CL,30H              ;0th row in first col

NEXT1:                          ; to find the corresponding row in first col.
        INC CL                  ; increment the value of CL to make first row in first col.
        SHR  AL,01              ; shift right AL by 1 bit
        JC DISPLAY              ;carry flag (CF) is set when 1 is moved to carry flag from AL
        INC SI                  ;else inc SI and repeat till 1 (carry - CF) is found
        JMP NEXT1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SECONDCOL: LEA SI,DISPLAYABLECHARACTER      ;SI points to displayablecharacter array
        ADD SI,08               ; adding 8 to SI to make SI point to second column
        MOV COL,32H             ; control has come here means, the key that we have pressed
                                  happens to be in second column
        MOV CL,30H              ;0th row in second col

NEXT2:                          ; to find the corresponding row in second col.
        INC CL                  ; increment the value of CL to make first row in second col.
        SHR AL,01               ; shift right AL by 1 bit
        JC DISPLAY              ; carry flag (CF) is set when 1 is moved to carry flag from AL
        INC SI                  ; else inc SI and repeat till 1 (carry - CF) is found
        JMP NEXT2
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
THIRDCOL: LEA SI,DISPLAYABLECHARACTER     ;SI points to displayablecharacter array
    ADD SI,16            ;adding 16 to SI to make SI point to third column
    MOV COL,33H          ; control has come here means, the key that we have pressed
                             happens to be in third column
    MOV CL,30H           ; 0th row in third col

NEXT3:                   ; to find the corresponding row in third col.
    INC CL               ; increment the value of CL to make first row in third col.
    SHR AL,01            ; shift right AL by 1 bit
    JC DISPLAY           ; carry flag (CF) is set when 1 is moved to carry flag from AL
    INC SI               ; else inc SI and repeat till 1 (carry - CF) is found
    JMP NEXT3
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

DISPLAY:
    MOV DL,[SI]          ;SI points to whatever key that we have pressed
    MOV KEY,DL           ; pressed key is moved to the variable called key
    PRINTF KEYPRESSEDMSG ;keypressedmsg is displayed along with key pressed

    MOV ROW,CL           ;to display row msg with its number
    PRINTF ROWMSG

    PRINTF COLMSG        ;to display col msg with its number

    EXIT
END
```

OUTPUT : According to the given diagram below, whichever key is pressed, it displays that key with the corresponding row and column numbers.

Sample output: Assuming you pressed 0, the output should be,

```
KEY PRESSED IS=  0
THE ROW NUM IS= 1
THE COLUMN NUM IS=  1
```

Column	01(0001)	02(0010)	04(0100)
ROW	COL1	COL2	COL3
ROW1	0	8	c
ROW2	1	9	e
ROW3	2	.	k
ROW4	3	+	=
ROW5	4	-	m
ROW6	5	*	r
ROW7	6	/	s
ROW8	7	%	a

What do MC, MR, MS, M+, and M- in calculators do?

3 Answers

Joseph Lurie, logician and philosopher in the tradition of Aristotle (who wrote on everything)

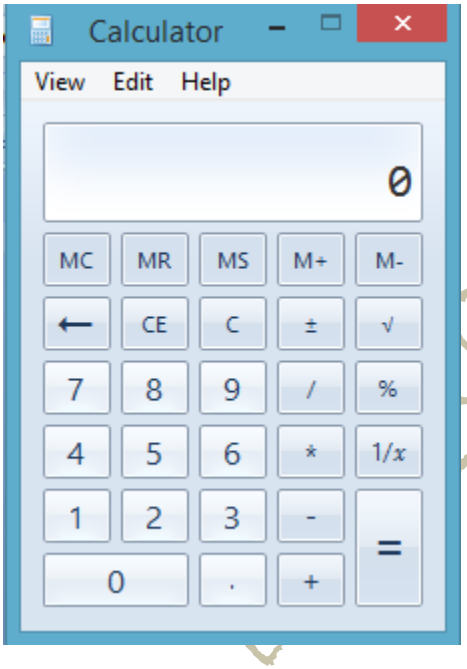
12.3k Views • Joseph is a Most Viewed Writer in Calculators.

Most calculators have the ability to store a number in memory in addition to the number that appears directly on the screen. The ms (memory store) button copies the number from the screen to the memory. The mr (memory recall) button brings the memory value back to the screen. M+ and m- add or subtract the screen value from the memory value (hiding the result in the memory, so you need to be careful not to double-press the button). Mc (memory clear) resets the memory value to zero. The mc and ms functions are functionally redundant, so some calculators will only provide one of them.

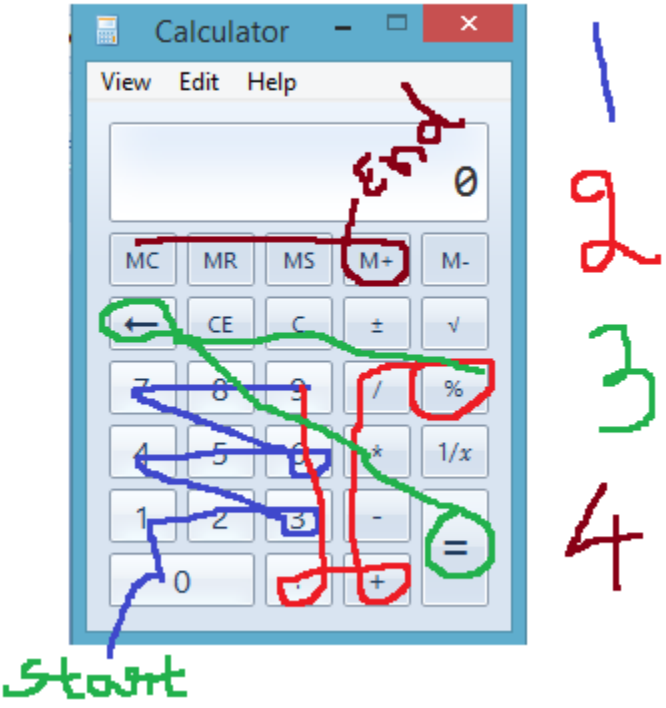
Written May 14, 2015 • View Upvotes • Answer requested by Amany Amany

How to remember the above table?

Well, you open up calculator which looks like this.



and try to follow this pattern (I know it's lame, but works).



STEPPER MOTOR INTERFACE (for 8b)

Data actuation and control represents the most popular applications of Microprocessor. Stepper motor control is a very popular application of control area, as stepper motors are capable of accepting pulses directly from the microprocessor and move accordingly.

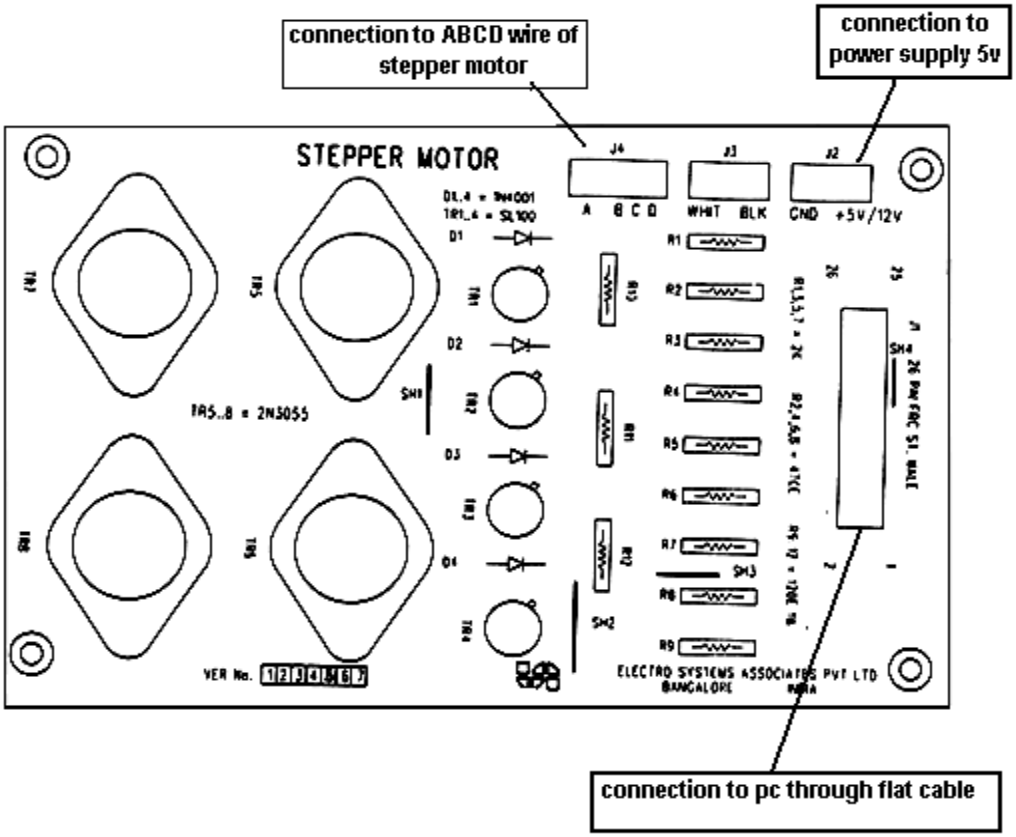
There are several areas of stepper motor applications like instrumentation, computer peripherals and machine tool drives. Tiny stepper motors are used in quartz analog electronics watches for driving the second, minute and hour hands. These motors operate directly within the button cells used in these electronic watches. Bigger stepper motors are used for driving the hands of slave clocks on railway platforms, bus stations, offices, factories , etc computer peripherals form an important areas of stepper motor applications. Card readers/punches, papers tape readers/punches, Teleprompters and teletypes represents the first application's areas of stepper motors. Digital X-Y plotters and dot matrix printers uses stepper motors for driving the arm and pen, and the paper respectively. Stepper motors find application in line printers to drive the paper advance mechanism. Floppy disks and hard/Winchesters disks have their magnetic reading/writing heads positioned by stepper motors.

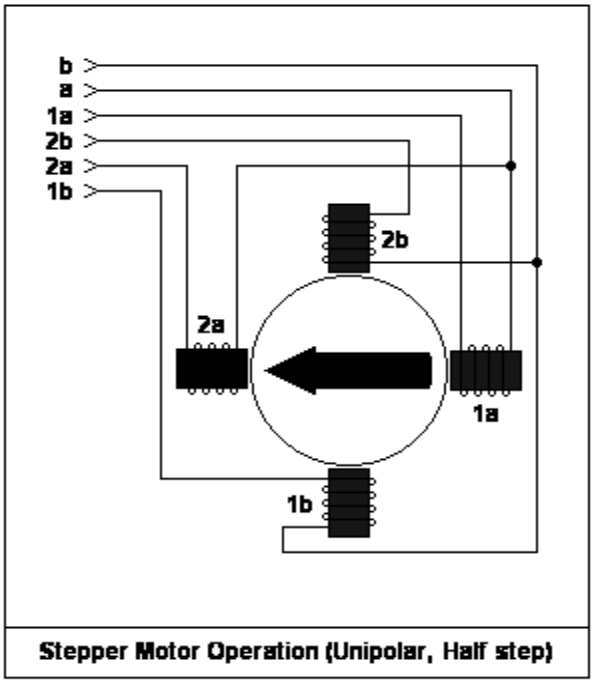
Then main applications areas of stepper motor are in Numerical Control (NC) systems for machine tools. Here they are utilized for driving the cutting tool along x, y, z directions. Another applications in this area is the co-ordinate table. Indexing mechanisms used in multi station machine tools employ stepper motors for moving either work piece or cutting tools.

SPECI FICATION OF THE STEPPER MOTOR USED

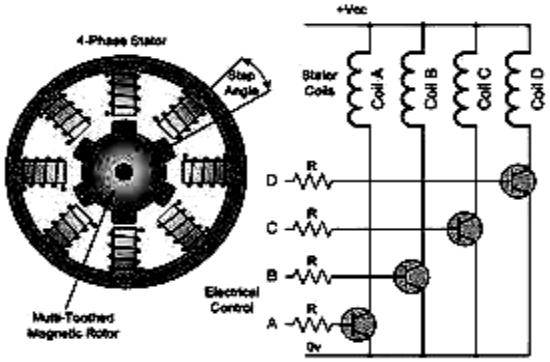
The stepper motor is reversible one with a torque of 3 kg-cm. The power requirement is +5DC @1.2 A current per winding at full torque. The step angle is 1.8 degree i.e. tor the every single excitation, the motor shafts rotates by 1.8 degree. for the motor to rotate one full revolution, number of steps required is 200. The stepper motor used has four stator windings which are brought out through colored wires terminated at a 4 pin polarized female connector.

STEPPER MOTOR HARDWARE MODULE DIAGRAM





Stepper Motor Operation (Unipolar, Half step)



8.B) DRIVE A STEPPER MOTOR INTERFACE TO ROTATE THE MOTOR BY N STEPS LEFT DIRECTION AND N STEPS RIGHT DIRECTION (N IS SPECIFIED BY THE EXAMINER). INTRODUCE SUITABLE DELAY BETWEEN SUCCESSIVE STEPS. (ANY ARBITRARY VALUE FOR THE DELAY MAY BE ASSUMED BY THE STUDENT).

```
.MODEL SMALL
INITDS MACRO
        MOV AX,@DATA        ; Initialization of data segment
        MOV DS,AX             (to access data)
ENDM

INIT8255 MACRO
        MOV AL,CW           ; Initialization of 8255 using control word
        MOV DX,CR           by passing 82h to control reg.
        OUT DX,AL           (to make port A as output)

ENDM

OUTPA MACRO                 ; one is enough
        MOV DX,PA
        OUT DX,AL
ENDM

EXIT MACRO
        MOV AH,4CH          ;Terminating the program
        INT 21H
ENDM

.DATA
        PA EQU 1190H        ; one is enough
        CR EQU 1193H
        CW DB 82H
.CODE
        INITDS
        INIT8255
        MOV AL,88H          ; setting value in Al 88=10001000
        MOV CX,200          ; taking count as 200
```

```
ROTATE:  OUTPA              ; perform rotation on port A
         CALL DELAY         ; have some delay in between the steps.
         ROR AL,01          ;CLOCKWISE DIRECTION- rotate right contents of AL i.e.
         DEC CX               10001000 is rotated towards right by 1 bit. This
       JNZ ROTATE             makes the stepper motor to rotate clock wise direction.
                            This loop repeats until step size becomes 00
         EXIT               ; once the count becomes 0, call exit macro

DELAY PROC
         PUSH AX            ; save AX as we change in next lines
         PUSH CX            ; save CX as we change in next lines
         MOV AX,0CFFH       ; speed of the stepper motor can be controlled by
OUTER:   MOV CX,0FFFFH        changing count value
INNER:   DEC CX
         JNZ INNER
         DEC AX
         JNZ OUTER
         POP CX
         POP AX
         RET
DELAY ENDP
END
```

For anti-clockwise rotation, do rotate AL towards left by 1 bit. i.e
ROL AL,01

You can take any other registers like SI & DI, BX etc. Just make sure your selection of registers doesn't affect the overall program.

If you want to take the same registers as you use in your main program like in this case, just push them on to stack, do whatever you want, and then pop them.

OUTPUT:

Stepper motor is rotated clock wise according to the count.

Formula to find step angle:

$$\text{Step angle} = \frac{360}{Rp * Sp} = \frac{360}{50*4} = 1.8\ ^\circ$$

Rp --- no of rotor poles
Sp --- no of stator poles

DUAL DAC INTERFACE (for 9b, 10b, 11b)

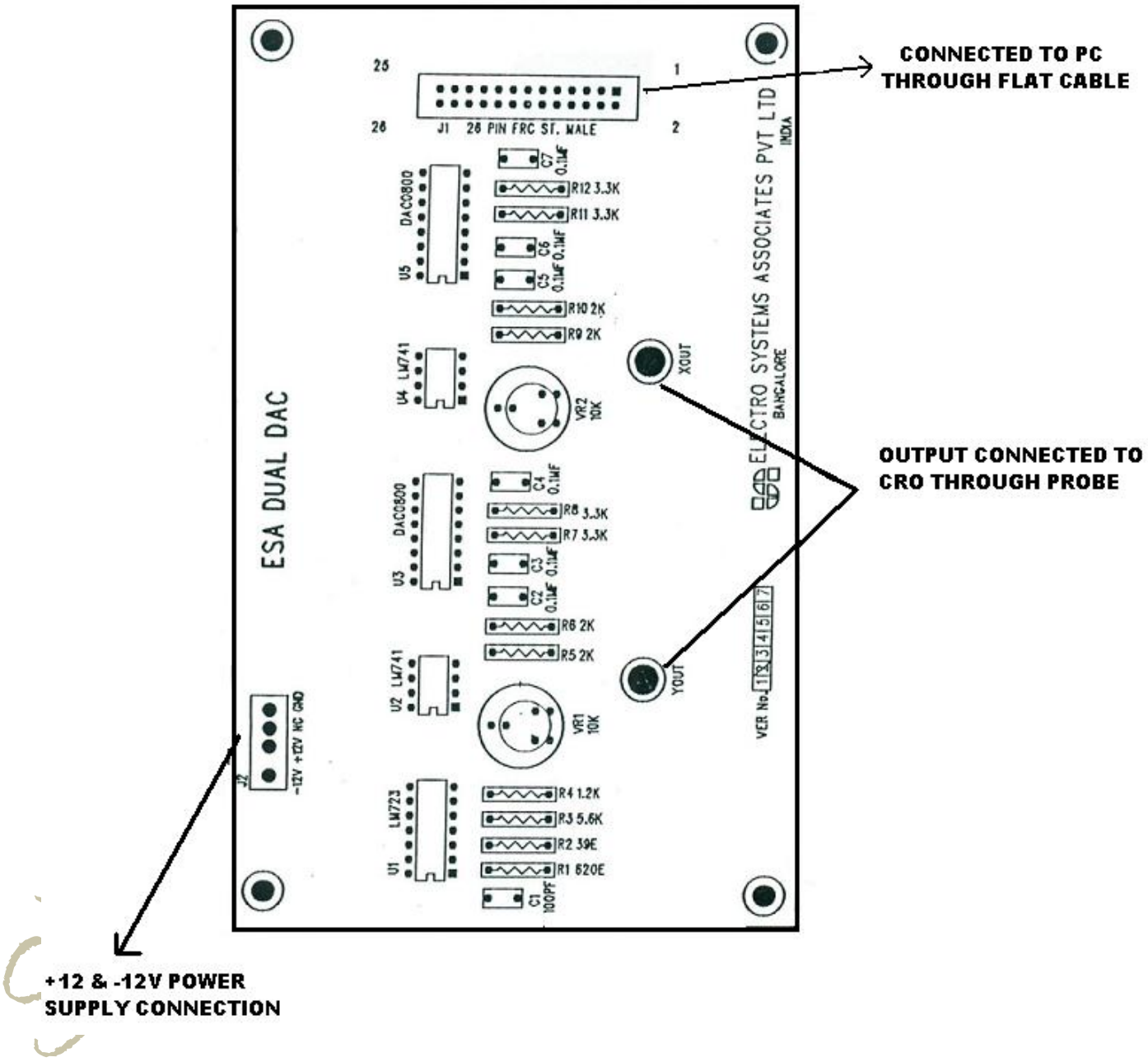
INTRODUCTION

The Dual DAC interface can be used to generate different interesting waveforms using microprocessor. These are two eight-bit digital to analog converters provided based all DAC 0800. The digital inputs to these DAC's arc provided through the port A and port B of 8255 used as output ports. The analog output from the DAC is given to operational amplifiers which act as current to voltage converters.

DESCRIPTION OFTHE CIRCUIT

The port A and port B of 8255 programmable peripheral interfaces are used as output ports. The digital inputs to the DAC s are provided through the port A and port B of 8255. The analog outputs of the DAC s are connected to the inverting inputs of the opamps 741 which act as current to voltage converters. The outputs from the opamps are connected to pins marked Xout & Yout at which the waveforms are observed on a CRO. The reference voltage for the DAC s is derived from an on-board voltage regulator 723. it generates a voltage of about 8V. The offset balancing of the opamps is done by making use of the two 10K pots provided. The output waveforms are observed at Xout & Yout on an oscilloscope.

DUAL DAC INTERFACE MODULE DIAGRAM



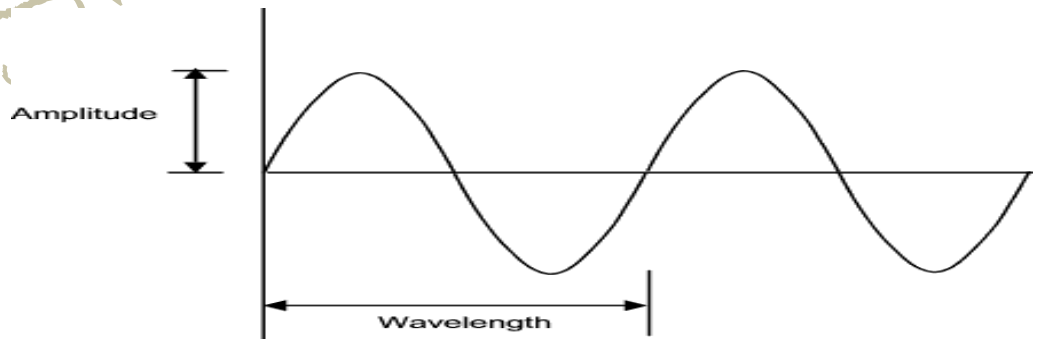
9.B) GENERATE THE SINE WAVE USING DAC INTERFACE (THE OUTPUT OF THE DAC IS TO BE DISPLAYED ON THE CRO).

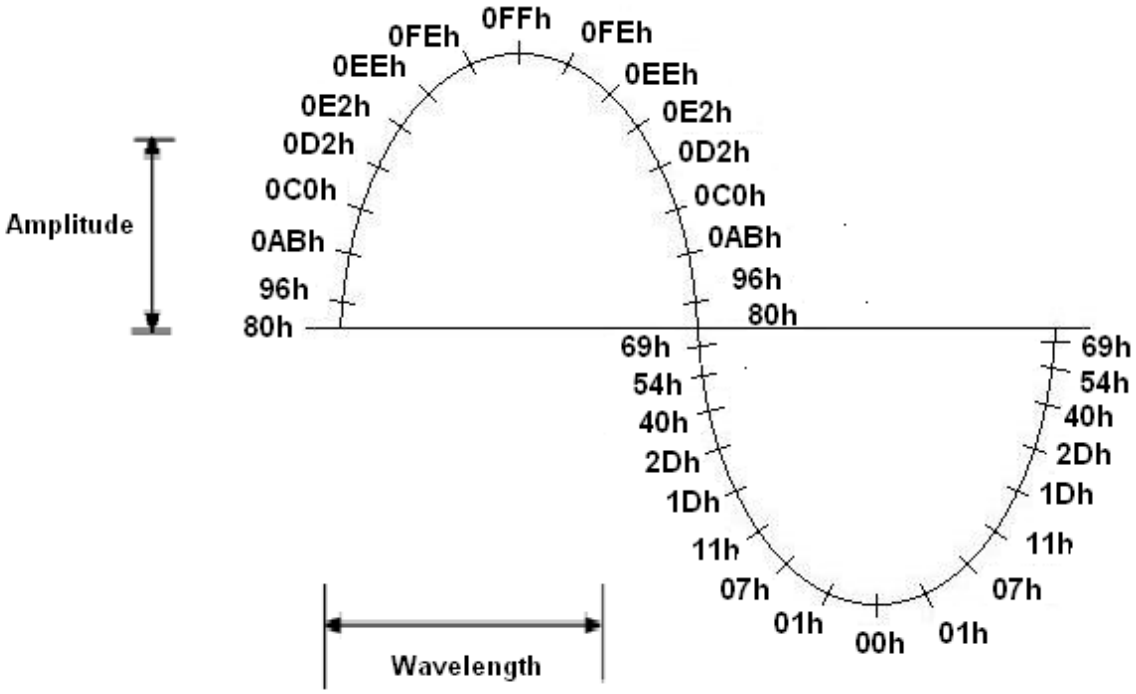
```
INITDS MACRO
        MOV AX,@DATA        ; Initialization of data segment
        MOV DS,AX             (to access data)
        ENDM

INIT8255 MACRO
        MOV AL,CW           ; Initialization of 8255 using control word
        MOV DX,CR           by passing 82h to control reg.
        OUT DX,AL            (to make port A as output)

ENDM

OUTPA MACRO             ; one is enough
        MOV DX,PA
        OUT DX,AL
ENDM

EXIT MACRO
        MOV AH,4CH          ;Terminating the program
        INT 21H
ENDM

.MODEL SMALL
.STACK
.DATA
        PA EQU 1190H        ;one is enough
        CR EQU 1193H
        CW DB 82H
        TABLE DB 80H,96H,0ABH,0C0H,0D2H,0E2H,0EEH,0F8H,0FEH,0FFH  ;+ve 1st half
              DB 0FEH,0F8H,0EEH,0E2H,0D2H,0C0H,0ABH,96H,80H       ;+ve 2nd half
              DB 69H,54H,40H,2DH,1DH,11H,07H,01H,00H              ;-ve 1st half
              DB 01H,07H,11H,1DH,2DH,40H,54H,69H,80H              ;-ve 2nd half
        MSG DB 10,13,"PRESS ANY KEY TO EXIT $"
```

Look at the conversion table at the end of this program. Then you will understand these

```
.CODE
        INITDS
        INIT8255
        LEA DX,MSG
        MOV AH,9
        INT 21H                          ; or   MOV CX,25H

START:  MOV CX,37                ;count value is taken 37 bcz the table contains 37 values
        LEA SI,TABLE             ; table address is loaded to SI
BACK:   MOV AL,[SI]              ; the contents of SI is moved to AL i.e. single value of
                                   table is moved
        OUTPA                    ;moved value is sent to hardware module through PORT A
        CALL DELAY
        INC SI                   ; SI is pointed to the next value of table
        LOOP BACK   ; loop repeats until all the contents of table is moved (till CX becomes 0)

        MOV AH,1
        INT 16H                  ; checks if any key is pressed in keyboard. If you
        JZ START                   haven't, then go to START

        EXIT                     ; if you press any key, just call exit macro
DELAY PROC
        MOV BX,0FFFH             ; NOTE: single loop delay is enough
L2:     DEC BX
        JNZ L2
        RET
DELAY ENDP
END
```

OUTPUT:





conversion table for producing sin wave							
Formula V=128*128 sin θ							
+ve Values				-ve Values			
θ	sinθ	v=128+(128*sinθ)	Hex	θ	sinθ	v=128+(128*sinθ)	Hex
0	0	128	**80h**	0	0	128	**80h**
10	0.1736	150.22	**96h**	-10	-0.1736	105.78	**69h**
20	0.342	171.78	**0Abh**	-20	-0.342	84.22	**54h**
30	0.5	192.00	**0C0h**	-30	-0.5	64.00	**40h**
40	0.6428	210.28	**0D2h**	-40	-0.6428	45.72	**2Dh**
50	0.766	226.05	**0E2h**	-50	-0.766	29.95	**1Dh**
60	0.866	238.85	**0Eeh**	-60	-0.866	17.15	**11h**
70	0.9397	248.28	**0F8h**	-70	-0.9397	7.72	**07h**
80	0.9848	254.05	**0Feh**	-80	-0.9848	1.95	**01h**
90	1	255.00	**0FFh**	-90	-1	255.00	**0FFh**

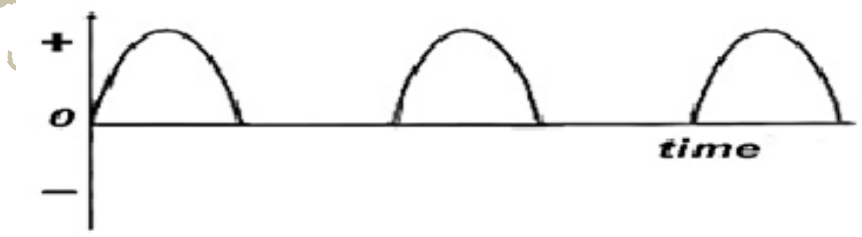
10.(B) GENERATE A HALF RECTIFIED SINE WAVE FORM USING THE DAC INTERFACE. (THE OUTPUT OF THE DAC IS TO BE DISPLAYED ON THE CRO).

```
INITDS MACRO
        MOV AX,@DATA        ; Initialization of data segment
        MOV DS,AX             (to access data)
        ENDM

INIT8255 MACRO
        MOV AL,CW           ; Initialization of 8255 using control word
        MOV DX,CR           by passing 82h to control reg.
        OUT DX,AL            (to make port A as output)

ENDM

OUTPA MACRO             ; one is enough
        MOV DX,PA
        OUT DX,AL
ENDM

EXIT MACRO
        MOV AH,4CH          ;Terminating the program
        INT 21H
ENDM

.MODEL SMALL
.STACK
.DATA
        PA EQU 1190H        ;one is enough
        CR EQU 1193H
        CW DB 82H
        TABLE DB 80H,96H,0ABH,0C0H,0D2H,0E2H,0EEH,0F8H,0FEH,0FFH  ;+ve 1st half
              DB 0FEH,0F8H,0EEH,0E2H,0D2H,0C0H,0ABH,96H,80H       ;+ve 2nd half
              DB 80H,80H,80H,80H,80H,80H,80H,80H,80H              ; all zeros (T-OFF)
              DB 80H,80H,80H,80H,80H,80H,80H,80H,80H              ; all zeros (T-OFF)
        MSG DB 10,13,"PRESS ANY KEY TO EXIT $"
```

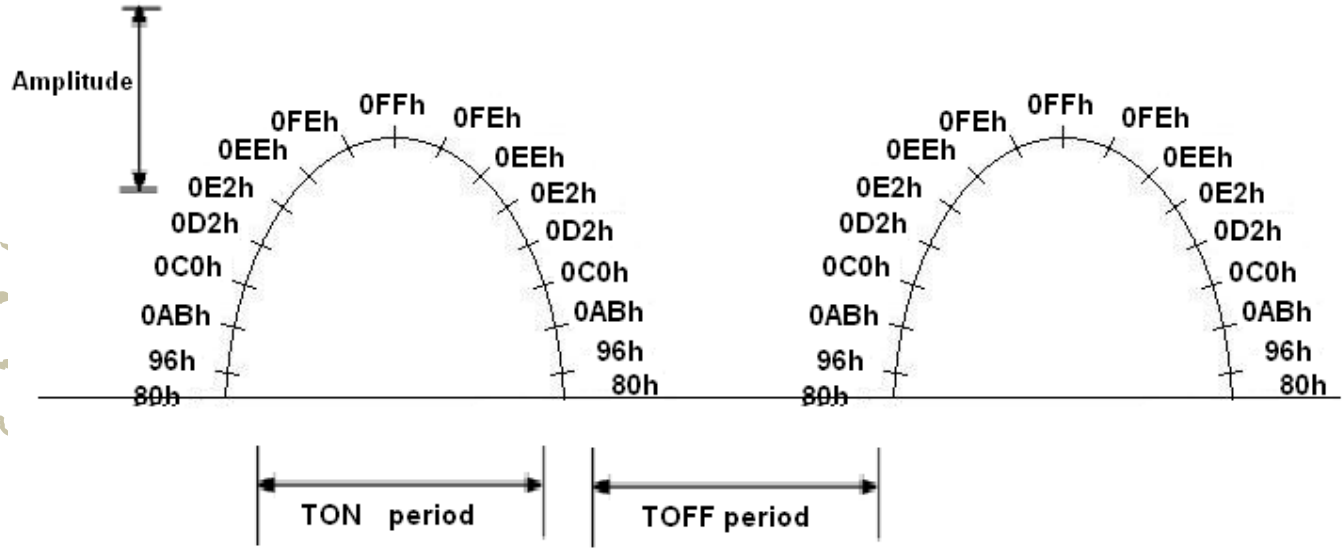
Look at the conversion table at the end of this program. Then you will understand these

```
.CODE
        INITDS
        INIT8255
        LEA DX,MSG
        MOV AH,9
        INT 21H                    ; or   MOV CX,25H

START:  MOV CX,37                  ;count value is taken 37 bcz the table contains 37 values
        LEA SI,TABLE               ; table address is loaded to SI
BACK:   MOV AL,[SI]                ; the contents of SI is moved to AL i.e. single value of
                                     table is moved
        OUTPA                      ;moved value is sent to hardware module through PORT A
        CALL DELAY
        INC SI                     ; SI is pointed to the next value of table
        LOOP BACK   ; loop repeats until all the contents of table is moved (till CX becomes 0)

        MOV AH,1
        INT 16H                    ; checks if any key is pressed in keyboard. If you
        JZ START                     haven't, then go to START

        EXIT                       ; if you press any key, just call exit macro
DELAY PROC
        MOV BX,0FFFH               ; NOTE: single loop delay is enough
L2:     DEC BX
        JNZ L2
        RET
DELAY ENDP
END
```

OUTPUT:



Converssion table for producing sin wave			
Formula V=128*128 sin θ			
θ	**sinθ**	**v=128+(128*sinθ)**	**Hex**
0	0	128	80h
10	0.1736	150.22	96h
20	0.342	171.78	0ABh
30	0.5	192.00	0C0h
40	0.6428	210.28	0D2h
50	0.766	226.05	0E2h
60	0.866	238.85	0EEh
70	0.9397	248.28	0F8h
80	0.9848	254.05	0FEh
90	1	255.00	0FFh



11. B) GENERATE A FULLY RECTIFIED SINE WAVEFORM USING THE DAC INTERFACE. (THE OUTPUT OF THE DAC IS TO BE DISPLAYED ON THE CRO).

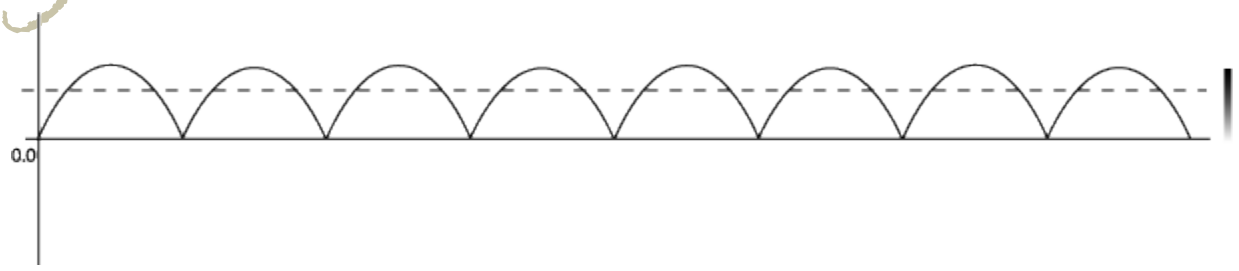
```
INITDS MACRO
        MOV AX,@DATA        ; Initialization of data segment
        MOV DS,AX             (to access data)
        ENDM

INIT8255 MACRO
        MOV AL,CW           ; Initialization of 8255 using control word
        MOV DX,CR           by passing 82h to control reg.
        OUT DX,AL            (to make port A as output)

ENDM

OUTPA MACRO             ; one is enough
        MOV DX,PA
        OUT DX,AL
ENDM

EXIT MACRO
        MOV AH,4CH          ;Terminating the program
        INT 21H
ENDM

.MODEL SMALL
.STACK
.DATA
        PA EQU 1190H        ;one is enough
        CR EQU 1193H
        CW DB 82H
        TABLE DB 80H,96H,0ABH,0C0H,0D2H,0E2H,0EEH,0F8H,0FEH,0FFH   ;+ve 1st half
              DB 0FEH,0F8H,0EEH,0E2H,0D2H,0C0H,0ABH,96H,80H        ;+ve 2nd half
              DB 96H,0ABH,0C0H,0D2H,0E2H,0EEH,0F8H,0FEH,0FFH       ;+ve 1st half
              DB 0FEH,0F8H,0EEH,0E2H,0D2H,0C0H,0ABH,96H,80H        ;+ve 2nd half
        MSG DB 10,13,"PRESS ANY KEY TO EXIT $"
```

Look at the conversion table at the end of this program. Then you will understand these

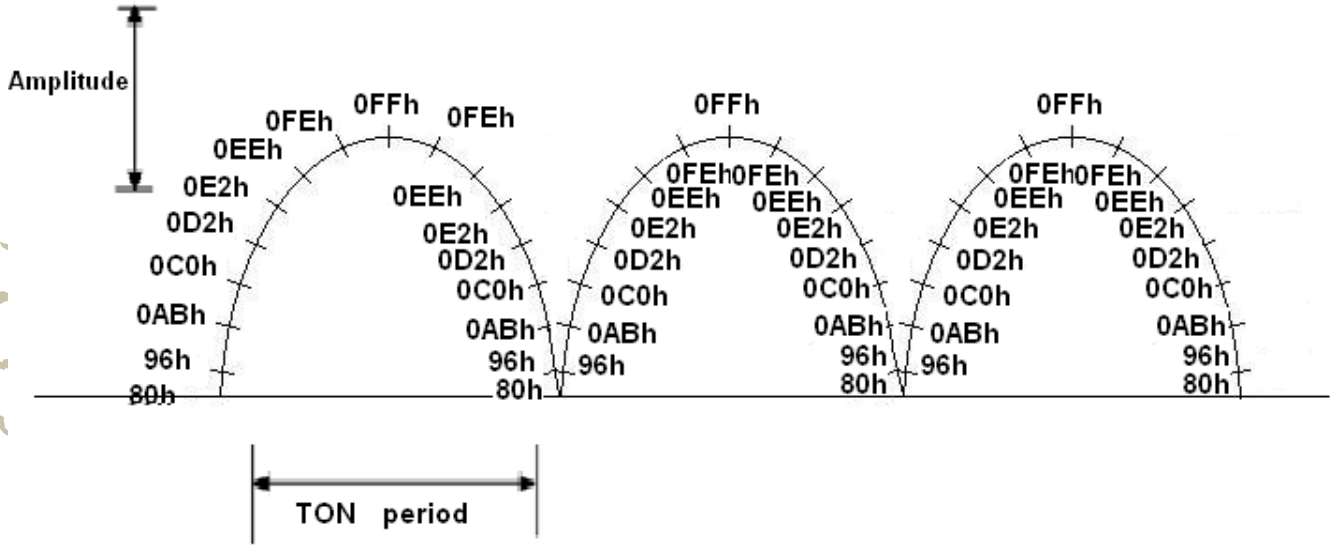
Copy & Paste

```
.CODE
        INITDS
        INIT8255
        LEA DX,MSG
        MOV AH,9
        INT 21H                    ; or   MOV CX,25H

START:  MOV CX,37                  ;count value is taken 37 bcz the table contains 37 values
        LEA SI,TABLE               ; table address is loaded to SI
BACK:   MOV AL,[SI]                ; the contents of SI is moved to AL i.e. single value of
                                     table is moved
        OUTPA                      ;moved value is sent to hardware module through PORT A
        CALL DELAY
        INC SI                     ; SI is pointed to the next value of table
        LOOP BACK   ; loop repeats until all the contents of table is moved (till CX becomes 0)

        MOV AH,1
        INT 16H                    ; checks if any key is pressed in keyboard. If you
        JZ START                     haven't, then go to START

        EXIT                       ; if you press any key, just call exit macro
DELAY PROC
        MOV BX,0FFFH               ; NOTE: single loop delay is enough
L2:     DEC BX
        JNZ L2
        RET
DELAY ENDP
END
```

OUTPUT:



Converssion table for producing sin wave			
Formula V=128*128 sin θ			
θ	**sinθ**	**v=128+(128*sinθ)**	**Hex**
0	0	128	80h
10	0.1736	150.22	96h
20	0.342	171.78	0ABh
30	0.5	192.00	0C0h
40	0.6428	210.28	0D2h
50	0.766	226.05	0E2h
60	0.866	238.85	0EEh
70	0.9397	248.28	0F8h
80	0.9848	254.05	0FEh
90	1	255.00	0FFh

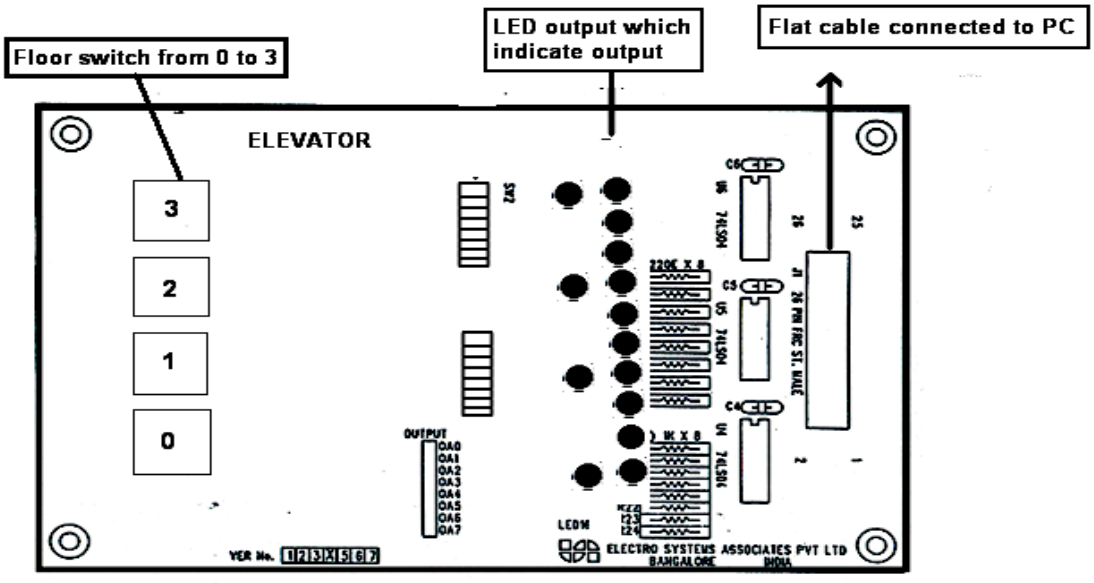


ELEVATOR INTERFACE

INTRODUCTION

The control and operation of an elevator can be simulated by the use of elevator interface. It is designed for four floors. A column of LEDs indicates the position of the elevator. The turning on of the successive LEDs one at a time in fixed time intervals also indicates the motion of the elevator. The port A of 8255 which is used as the output port controls these LEDs.

The request at any of the floors is indicated by means of request button. These are also LEDs placed at each floor, which gets turned on when there is a request from the corresponding floor. This request is latched by a flip-flop. The request can be recognized by the processor through port B or 8255 which acts as the input port. After the request is recognized by the processor it can be reset and then the request indicating LED turns off. When there is a request from more than one floor, priorities can be set and the elevator can be moved to the floor, which placed its request first, and then it can service the other floors in consecutive sequence. The location of the floor whose request button was depressed first determines the direction in which the elevator starts moving.



ELEVATOR HARDWARE MODULE DIAGRAM

Read this thoroughly

We use Port B for input

- If you press 0 in the above elevator interface, the input in AL will be (in PORT B7 – B0)

B7	B6	B5	B4	B3	B2	B1	B0
1	1	1	1	1	1	1	0

(FE)

Look at the position of 0. Where it is?? It is at the **0th bit (bit b0).** So it understands that the request is from 0th floor.

- If you press 1 in the above elevator interface, the input in AL will be (in PORT B7 – B0)

B7	B6	B5	B4	B3	B2	B1	B0
1	1	1	1	1	1	0	1

(FD)

Look at the position of 0. Where it is?? It is at the **1st bit (bit b1).** So it understands that the request is from 1st floor.

- If you press 2 in the above elevator interface, the input in AL will be (in PORT B7 – B0)

B7	B6	B5	B4	B3	B2	B1	B0
1	1	1	1	1	0	1	1

(FB)

Look at the position of 0. Where it is?? It is at the **2nd bit (bit b2).** So it understands that the request is from 2nd floor.

- If you press 3 in the above elevator interface, the input in AL will be (in PORT B7 – B0)

B7	B6	B5	B4	B3	B2	B1	B0
1	1	1	1	0	1	1	1

(F7)

Look at the position of 0. Where it is?? It is at the **3rd bit(bit b3).** So it understands that the request is from 3rd floor.

- On the other hand, If you don't press any number in the above elevator interface, AL will be (in PORT B3 – B0)

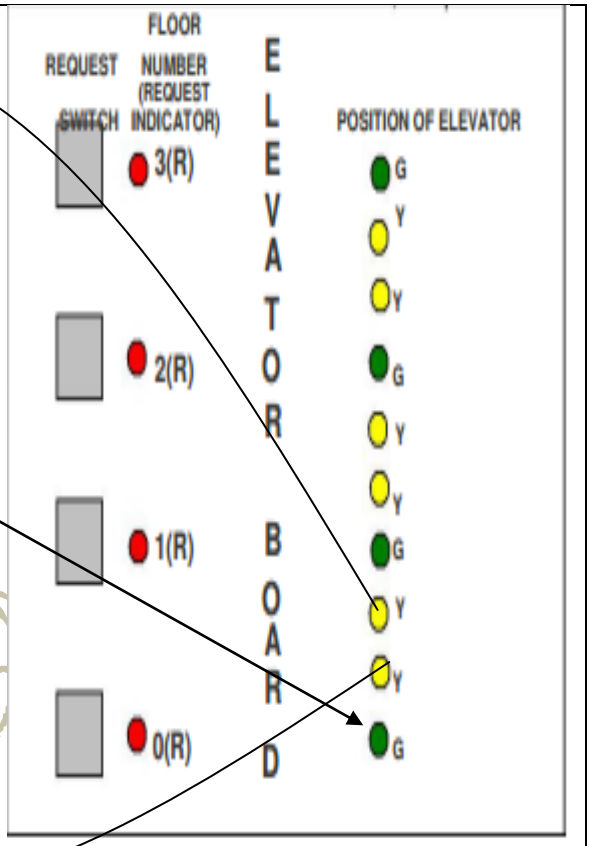
B7	B6	B5	B4	B3	B2	B1	B0
1	1	1	1	1	1	1	1

(FF)

Look at the position of 0. Where it is?? It is not found. So it understands that there is no request from any floor.

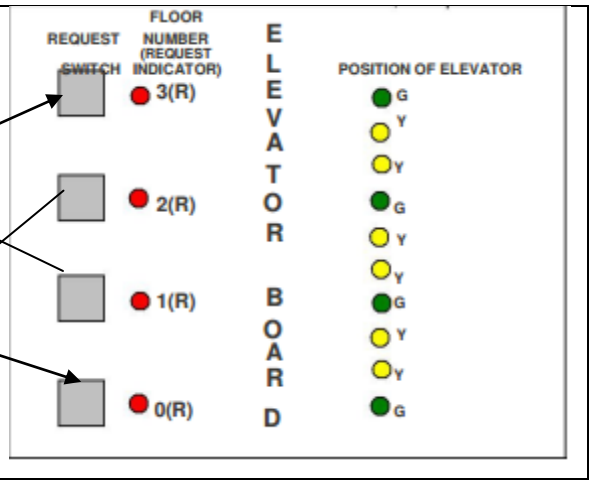
The BCD value on PA3-PA0 (Output) indicates the position of elevator (0 to 9). Lift position LEDs (Green) indicates the position of Lift at any time.

LED	PA3-PA0 (Output)			
0th LED	0	0	0	0
1st LED	0	0	0	1
2nd LED	0	0	1	0
3rd LED	0	0	1	1
4td LED	0	1	0	0
5th LED	0	1	0	1
6th LED	0	1	1	0
7th LED	0	1	1	1
8th LED	1	0	0	0
9th LED	1	0	0	1



The bits in the PA7-PA4 (Output) can be used to RESET Request Floor LED (Red). The bits corresponding to a floor must be reset to logic – 0 to turn off corresponding Request Floor LED.

FLOOR NUMBER	PA7-PA4 (Output)			
0th FLOOR	1	1	1	0
1st FLOOR	1	1	0	1
2nd FLOOR	1	0	1	1
3rd FLOOR	0	1	1	1



12. B) DRIVE AN ELEVATOR INTERFACE IN THE FOLLOWING WAY:

I. INITIALLY THE ELEVATOR SHOULD BE IN THE GROUND FLOOR, WITH ALL REQUESTS IN OFF STATE.

II. WHEN A REQUEST IS MADE FROM A FLOOR, THE ELEVATOR SHOULD MOVE TO THAT FLOOR, WAIT THERE FOR A COUPLES OF SECONDS, AND THEN COME DOWN TO GROUND FLOOR AND STOP. IF SOME REQUESTS OCCUR DURING GOING UP OR COMING DOWN THEY SHOULD BE IGNORED.

```
.MODEL SMALL
INITDS MACRO
        MOV AX,@DATA        ;Initialization of data segment
        MOV DS,AX             (to access data)
ENDM
INIT8255 MACRO
        MOV AL,CW           ;Initialization of 8255 using control word
        MOV DX,CR           by passing 82h to control reg.
        OUT DX,AL            (to make port A as output & port B as input)
ENDM

EXIT MACRO
        MOV AH,4CH          ;Terminating the program
        INT 21H
 ENDM

INPB MACRO
        MOV DX,PB           ; Initialization of PORT A as input
        IN AL,DX
ENDM

OUTPA MACRO
        MOV DX,PA           ; Initialization of PORT A as output
        OUT DX,AL
ENDM
PRINTF MACRO MSG
LEA DX,MSG
MOV AH,09H
INT 21H
ENDM
```

```
.DATA
        PA EQU 1190H
        PB EQU 1191H
        CR EQU 1193H
        CW DB 82H          ;82h is the value in control word 10000010, which makes
                           ;PORT  A as output & PORT B as input
        ANYKEYTOEXIT DB 10,13,"PRESS ANY KEY FOR EXIT $"
        FLOOR DB 00H,03H,06H,09H,0E0H,0D3H,0B6H,79H ;table which contains floor
                                                     numbers
.CODE
        INITDS             ;initializing data segment
START:
        INIT8255           ;initializing 8255 processor
        PRINTF ANYKEYTOEXIT
        MOV BL,0
INITIAL:MOV AL,BL          ;sets the initial position of lift to 0
        OR AL,0F0H
        OUTPA
        LEA SI,FLOOR
FLOORINPUT: CALL KBHIT     ;upon any key, just exit
        INPB               ; takes the input from elevator i.e which ever key is pressed
        OR AL,0F0H         and checks for the floor num  (see the above notes)
        MOV CL,AL          ; copy your floor value to CL
        SUB AL,0FFH        ; well if you don't press any number, AL will be FFh
        JZ FLOORINPUT            so in that case, go take the input again.
        MOV AL,CL          ; take back the floor value which you had copied to CL
        SUB AL,0FEH        ; if you press 0, since you are already in ground floor initially,
        JZ RESET                 just go to reset, turn off that 0 floor.

FLOORCHECK:
        MOV AL,CL
        ROR AL,1
        MOV CL,AL
        JNC DECIDE
        INC SI
        JMP FLOORCHECK
```

*CHECK REQUEST FROM ANY FLOOR. Here the logic is to **rotate right the request by 1 position**. If carry is not set, then request will be from ground floor, else **rotate** right the request by 1 more position. If carry is not set, then request will be from 1st floor, else **rotate** right the request by 1 more position. If carry is not set, then request will be from 2nd floor, else **rotate** right the request by 1 more position. If carry is not set, then request will be from 3rd floor.*

```
DECIDE:  CALL DELAY
         MOV AL,[SI]          ;decide floor value n loads to AL reg
         SUB AL,BL            ; BL initially holds 0
         JZ RESET     ; now you're done with up, go to reset, turn off and start moving down.
   UP:INC BL          ; BL will now hold the number of LED's to glow
         MOV AL,BL    ; turn on from bottom. Keep going up till it reaches your floor
         OR AL,0F0H
         OUTPA
         JMP DECIDE

 RESET:ADD SI,4       ;decides which floor and sends the corresponding output
         MOV AL,[SI]   to the module
         OUTPA

DOWN1:CALL DELAY
         MOV AL,BL    ; since BL holds the number of LED's to glow, copy that to AL
         OR AL,0F0H      and turn on that LED. Means you start coming down
         OUTPA           from that floor to ground floor
         DEC BL       ; coming down.
         JS START     ; (Jump if Sign) you can't go beyond ground floor. So start all over again.
         JNZ DOWN1    ; till you reach ground floor - 1.
         JMP INITIAL  ; see, if you reach ground floor, go to INITIAL where we turn
                         on the ground floor.

 KBHIT PROC
          MOV AH,1
          INT 16H
          JNZ DONE
          RET
   DONE: EXIT
  KBHIT  ENDP

 DELAY PROC
         PUSH CX
         PUSH BX
         MOV BX,0FFFH
```

```
        B2:MOV CX,0FFFFH
        B1:LOOP B1
           DEC BX
           JNZ B2
           POP BX
           POP CX
           RET
           DELAY ENDP
END
```

Some of the Repeated Macros

```
EXIT MACRO
        MOV AH,4CH          ;Terminating the program
        INT 21H
ENDM
READKB MACRO
        MOV AH,01H          ;Reading single character
        INT 21H
ENDM
Display MACRO
        MOV AH,02H          ;Displaying single character (you sh'd have
        INT 21H                          loaded printable char's ASCII value to DL)
ENDM
PRINTF MACRO MSG
        MOV AH,09H          ;Printing string
        LEA DX,MSG
        INT 21H
ENDM
INITDS MACRO
        MOV AX,@DATA        ;Initialization of data segment
        MOV DS,AX            (to access data)
        ENDM
 INIT8255 MACRO
        MOV AL,CW           ;Initialization of 8255 using control word
        MOV DX,CR
        OUT DX,AL
ENDM
INPB MACRO
        MOV DX,PB           ; Initialization of PORT A as input
        IN AL,DX
ENDM
OUTPA MACRO
        MOV DX,PA           ; Initialization of PORT A as output
        OUT DX,AL
ENDM
```

Procedures programs:

```
DELAY PROC
        PUSH AX
        PUSH CX
        MOV AX,0CFFH
OUTER:  MOV CX,0FFFFH
INNER:  LOOP INNER
        DEC AX
        JNZ OUTER
        POP CX
        POP AX
        RET
DELAY ENDP


KBHIT PROC
        MOV AH,1
        INT 16H
        JNZ DONE
        RET
DONE: EXIT
KBHIT  ENDP
```

Commonly used interrupt functions

Read functions:

```
MOV AH,01H          ;Reading single character
INT 21H

MOV AH,0AH          ;Reading string
INT 21H
```

Display functions:

```
MOV AH,02H          ;Displaying single character
INT 21H

MOV AH,09H          ;Displaying string
INT 21H
```

File functions:

```
MOV AH,3CH                  ; Creating file
MOV DX,OFFSET FILE NAME
INT 21H

MOV AH,41H                  ;Deleting file
MOV DX,OFFSET FILE NAME
INT 21H
```

Read time:

```
MOV AH,2CH
INT 21H
```

Set cursor position function:

```
MOV AH,02H
INT 10H
```

Character input with out echo

```
MOV AH,08H
INT 21H
```

To check Key board hit

```
MOV AH,1
INT 16H
```

Terminate program

```
MOV AH,4CH
INT 21H

INT 3
ALIGN 16    ; Aligns all the registers at the output screen
```

********** *All the best guys!* **********