CHAPTER 4

SYLLABUS:

4.1 Memory interfacing :-Program and data memory

4.2 I/O Interfacing:-LED, relays, keyboard,LCD,seven segment display,Stepper motor.

4.3 Interfacing DAC - 0808 with 8051 and itssimple programming

4.4 Interfacing ADC - 0808/09 with 8051 and its simple programming

4.1 Memory Interfacing

ROM (Program Memory)

ROM is a type of memory that does not lose its contents when the power is turned off. ROM is also called nonvolatile memory. There are different types of read-only memory.

- PROM
- EPROM
- EEPROM
- Flash EPROM
- Mask ROM

RAM (Data Memory)

RAM memory is called volatile memory since cutting off the power to the IC will result in the loss of data. Sometimes RAM is also referred to as RAM (read and write memory), in contrast to ROM, which cannot be written to. There are three types of RAM

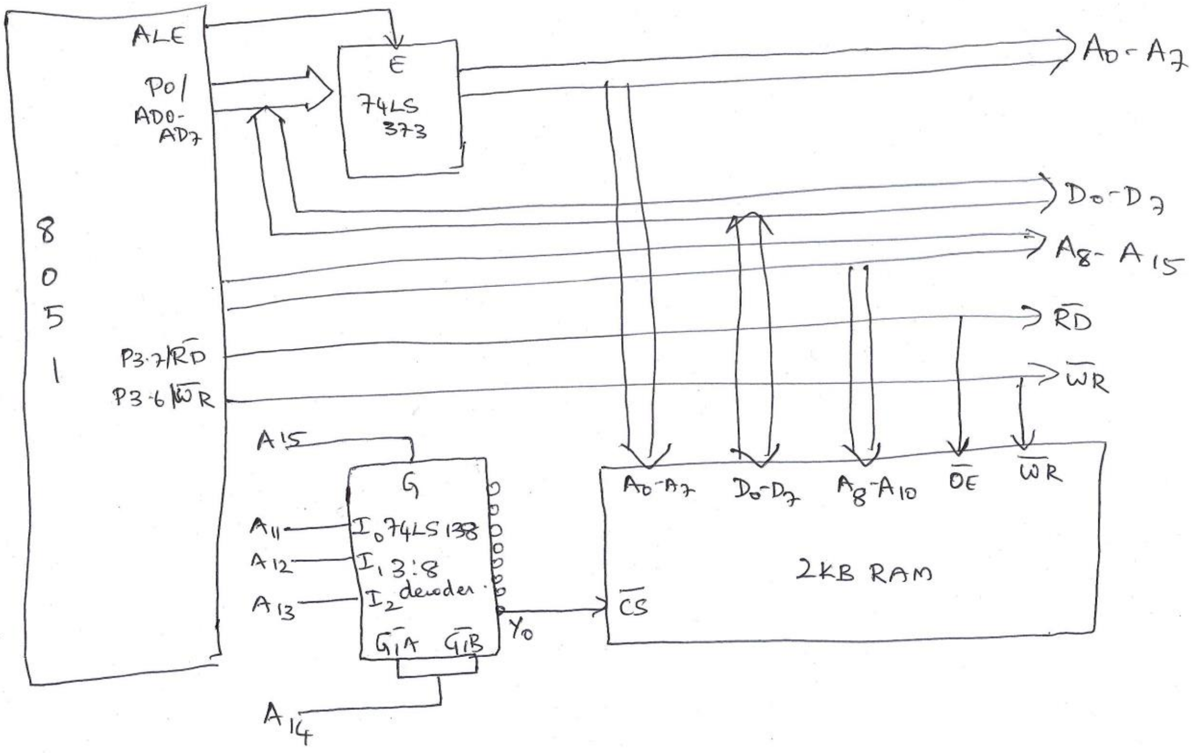
- Static RAM (SRAM)
- NV-RAM (nonvolatile RAM)
- Dynamic RAM (DRAM)

In connecting a memory chip to the CPU, note the following points

 The data bus of the CPU is connected directly to the data pins of the memory chip
 Control signals RD (read) and WR (memory write) from the CPU are connected to the OE (output enable) and WE (write enable) pins of the memory chip
 In the case of the address buses, while the lower bits of the address from the CPU go directly to the memory chip address pins, the upper ones are used to activate the CS pin of the memory chip

1. **Interface 2kb of RAM with 8051**

   To interface 2kB of memory number of address lines required is 11 (2^{11} = 2kB).
   ie. A0 to A10.
   The remaining address lines can be used for decoding using 3:8 decoder

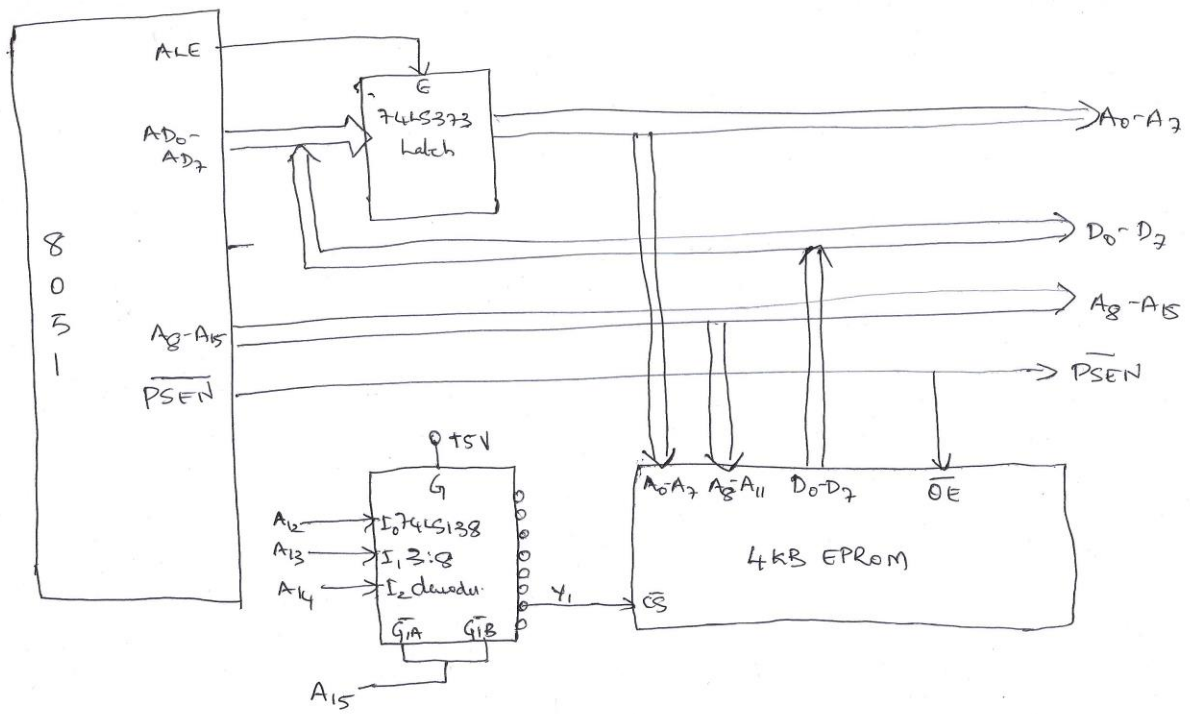


Memory Mapping:

A 15	A 14	A 13	A 12	A 11	A 10	A 9	A 8	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	Address
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Start address = 8000H
1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	End address = 87FFH

2. Interface 4kb of EPROM with 8051

To interface 4kB of memory number of address lines required is 12 (2^{12} = 2kB).
ie. A0 to A11.
The remaining address lines can be used for decoding using 3:8 decoder

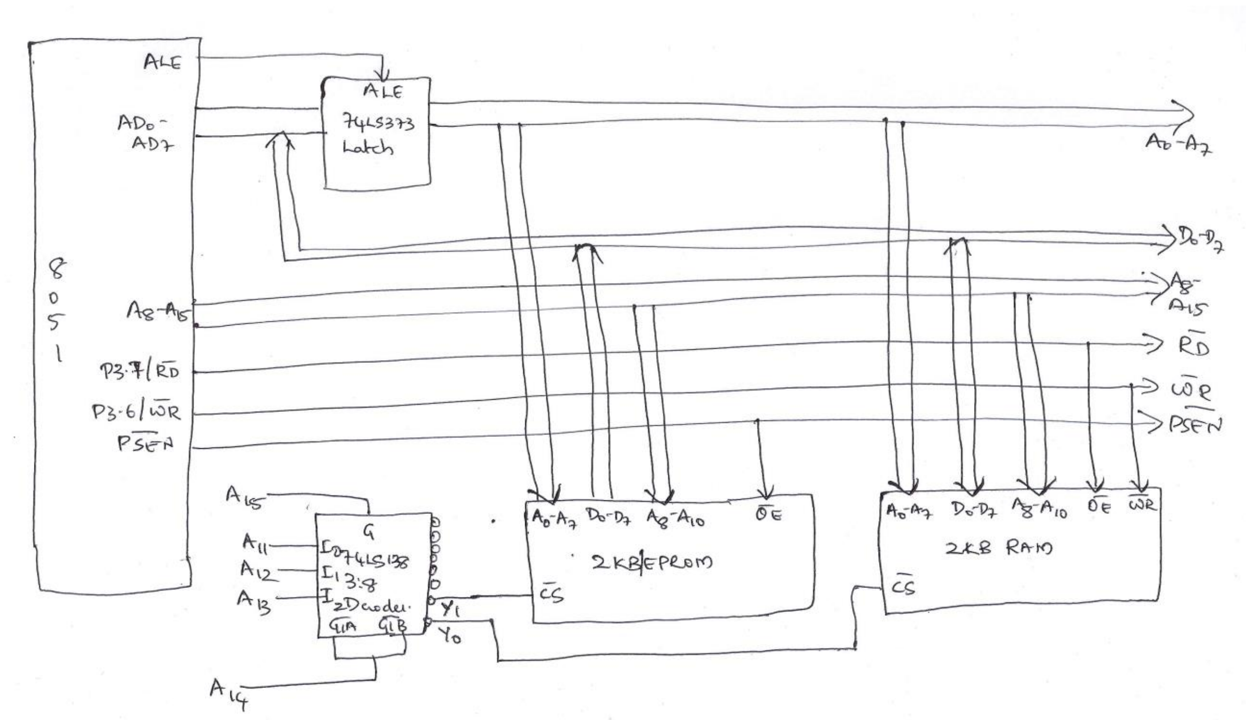


Memory Mapping:

A 15	A 14	A 13	A 12	A 11	A 10	A 9	A 8	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	Address
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	Start address =1000H
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	End address = 1FFFH

3. Interface 2kB 0f RAM and 2kB of EPROM with 8051

To interface 2kB of both memory, number of address lines required is 11 (2^{11} = 2kB).
ie. A0 to A10.
The remaining address lines can be used for decoding using 3:8 decoder.



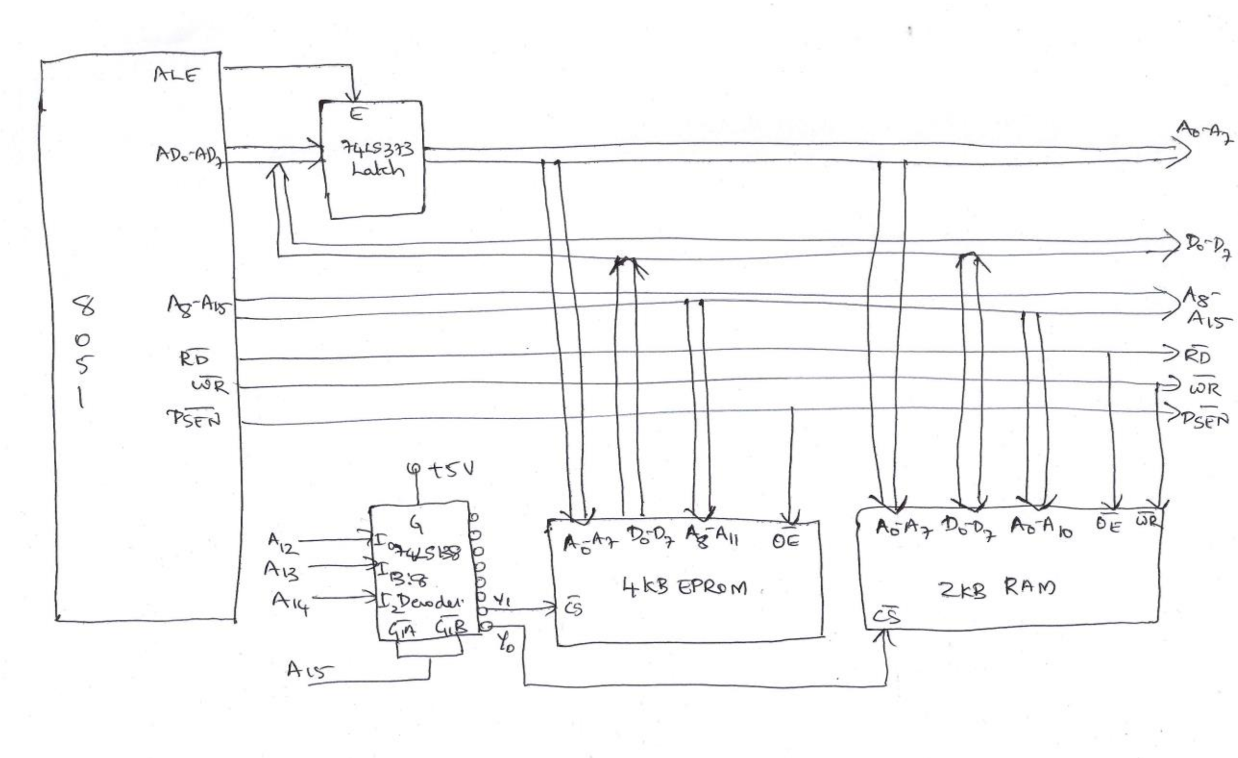
Memory Mapping:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Address
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Ram Start address =8000H
1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	Ram end address =87FFH
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	Eprom start address =8800H
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	Eprom end address =8FFFH

4. Interface 2kB of RAM and 4kB of EPROM with 8051

To interface 2kB of RAM number of address lines required is 11 (2^{11} = 2kB).

ie. A0 to A10.
To interface 4kB of EPROM number of address lines required is 12 (2^{12} = 2kB).
ie. A0 to A11.



Memory Mapping:

A 15	A 14	A 13	A 12	A 11	A 10	A 9	A 8	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	Address
0	0	0	0	X	0	0	0	0	0	0	0	0	0	0		Ram Start address =0000H
0	0	0	0	X	1	1	1	1	1	1	1	1	1	1	1	Ram end address =07FFH
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	Eprom start address =1000H
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	Eprom end address= 1FFFH

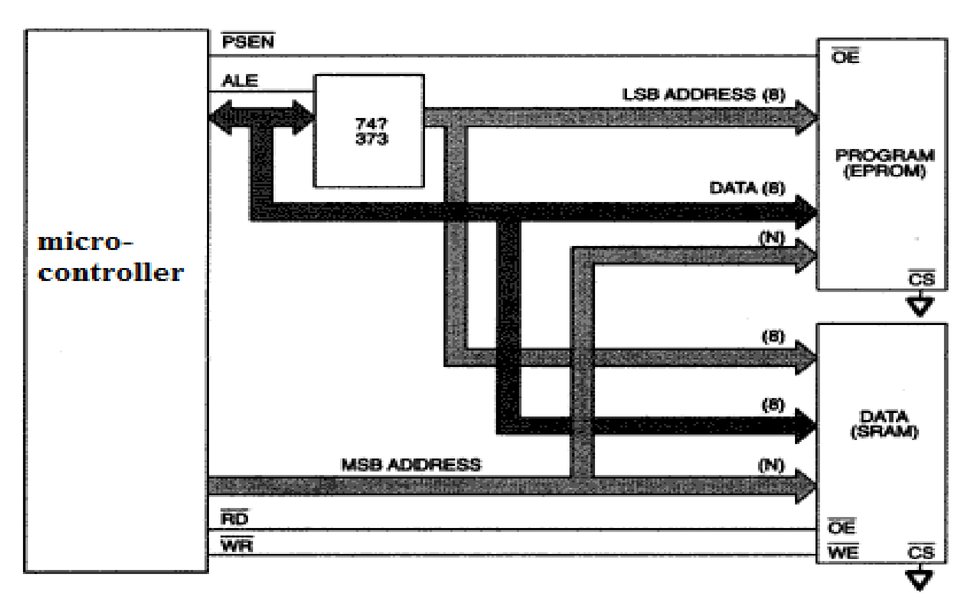
Absolute Decoding:-

All address lines are used to specify memory location to point physical memory location.

- Each physical memory location is identified by a unique address.
- The memory chip is selected only for specified logic level on higher order address lines, no other logic levels can select the chip.
- In this decoding , we get only one address for any 1 RAM/ROM.
- It increases the cost of decoding circuit.
- It is used in large memory system.

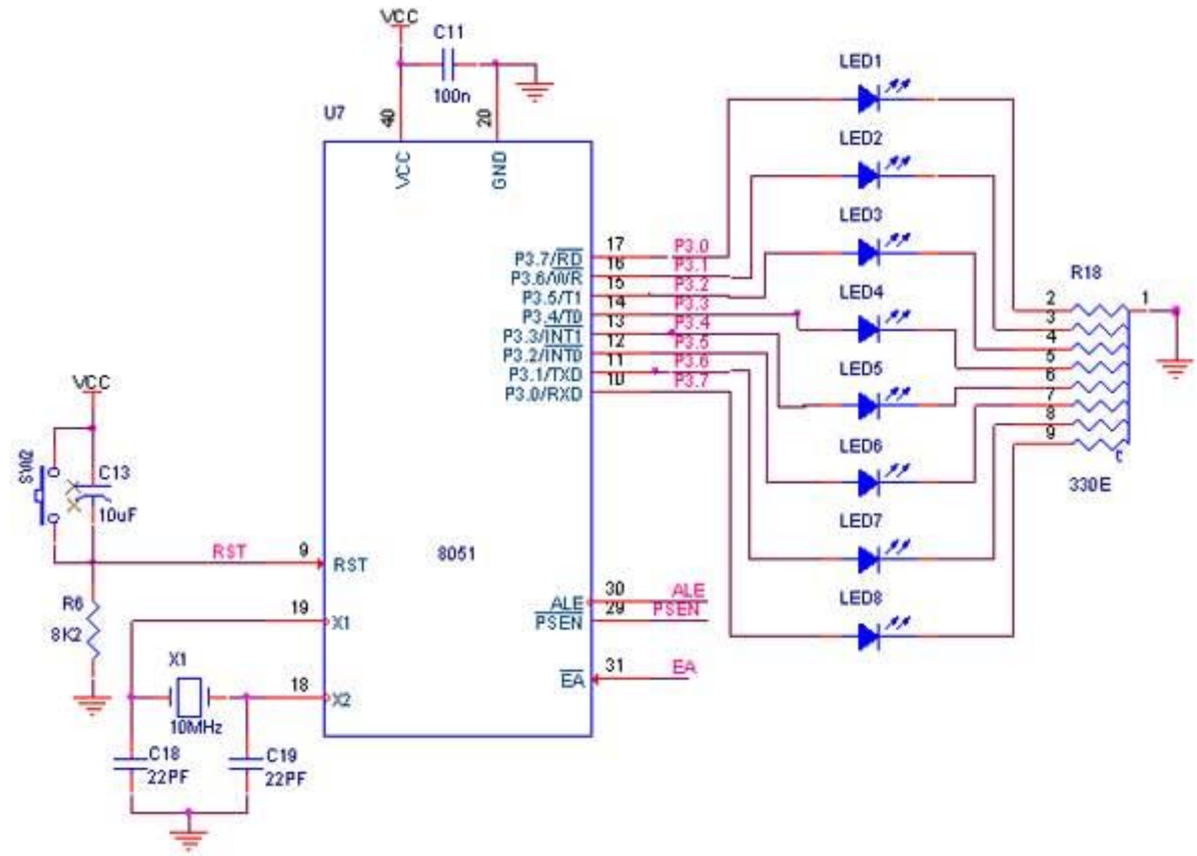
Partial Decoding:-

- The decoding in which all available address line are not used for decoding resulting in multiple address for same port is called partial decoding.
- Not all memory locations are used , only a subset of address lines are needed to point the physical memory location.
- Each physical memory is identified by several possible address.
- Hardware for decoding logic can be eliminated by using individual higher-order address lines to select memory chips.
- It reduces the cost of decoding circuit.
- It is used in small memory system.



4.2 I/O Interfacing:-LED, relays, keyboard, LCD, seven segment display, Stepper motor.

1) Interfacing of LEDs



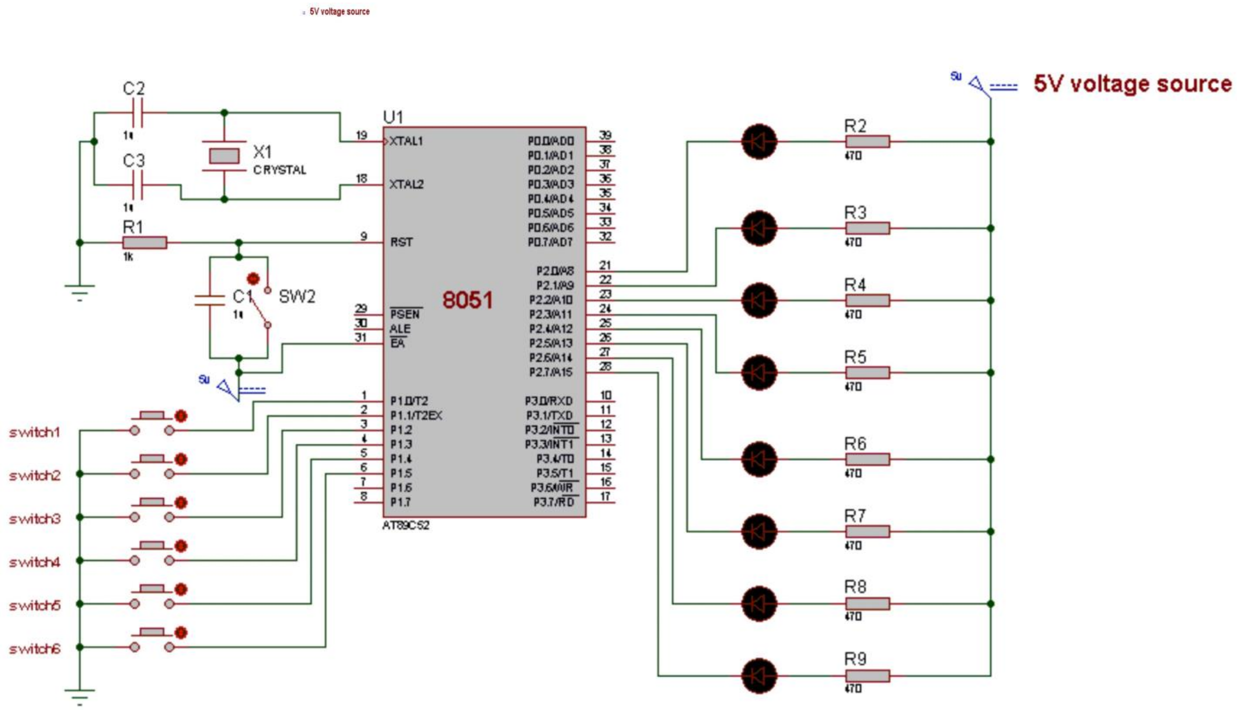
The Anode is connected through a resistor to GND & the Cathode is connected to the **Microcontroller** pin. So when the Port Pin is HIGH the LED is OFF & when the Port Pin is LOW the **LED** is turned ON.

Program to make the all the LEDs on and off

```
L1:MOV A,#FF
MOV P3,A
LCALL DELAY
MOV A,#00
MOV P3,A
LCALL DELAY
SJMP L1
DELAY: MOV R5,#05
H3      MOV R4,#FF
H2      MOV R3,#FF
```

```
H1:     DJNZ R3,H1
DJNZ R4,H2
DJNZ R5,H3
RET
```

Interfacing switches and LEDs



Write a program to switch on the LED connected to P2 when the corresponding switch connected to P1 is pressed.

Whenever a switch is pressed, the corresponding port 1 line will be at logic 0. If this is transferred to P2, the corresponding LED will glow.

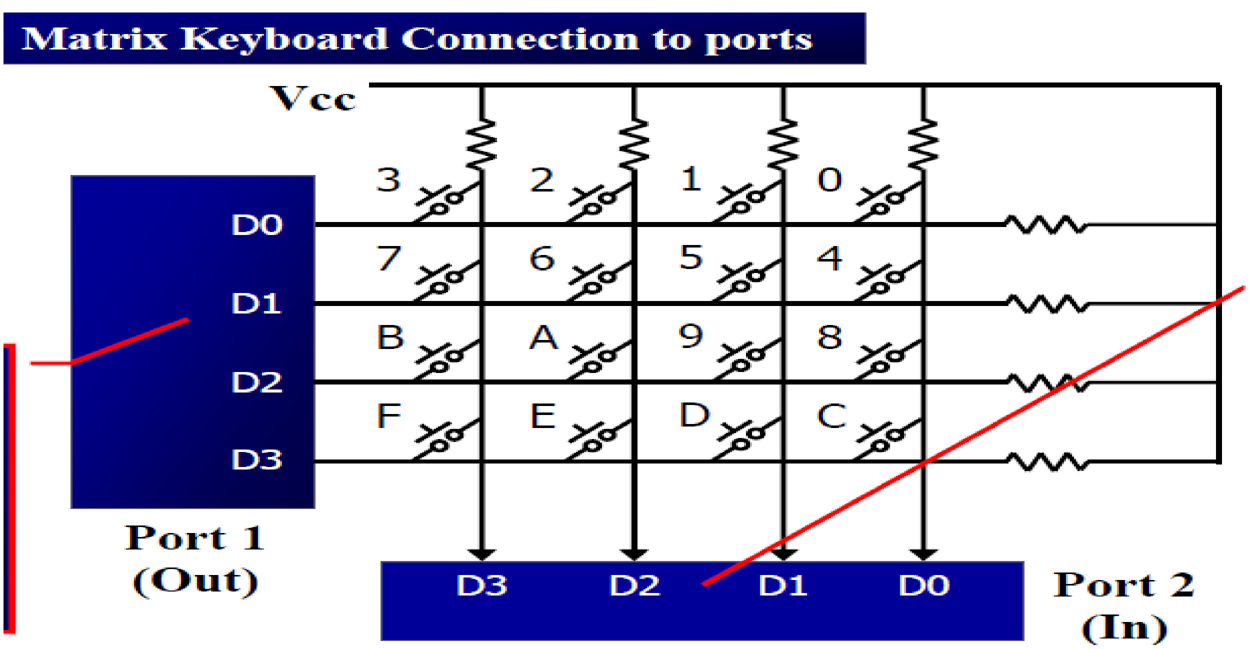
Program:

```
MOV P1,#0FFH                          ;P1 AS INPUT PORT

L1: MOV A,P1                          ; MOVE SWITCH POSITION TO A FROM P1

MOV P2,A                              ; MOVE SWITCH POSITION TO P2 SO THAT THE
CORRESPONDING LED  GLOW

SJMP L1                               ; CONTINUE
```

Write a program to on and off (blink) an LED connected to P2.0 when switch connected to P1.0 is pressed

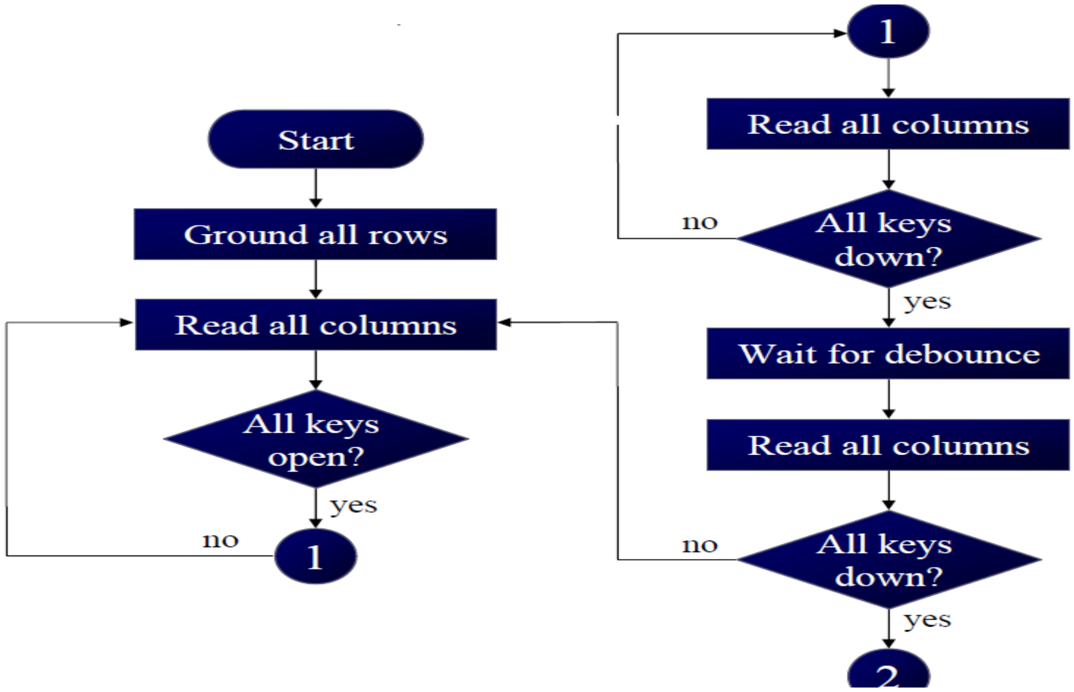
```
SETB P1.0                    ;P1.0 AS INPUT LINE
L1: JB P1.0, L1              ;CHECK WHETHER SWITCH CONNECTED TO P1.0 IS PRESSED
CLR P2.0                     ;ON THE LED CONNECTED TO P2.0 IF P1.0 = 0 (SWITCH IS PRESSED)
ACALL DELAY                  ; DELAY SUBROUTINE
SETB P2.0                    ; LED OFF AFTER SOMETIME
ACALL DELAY                  ; DELAY SUBROUTINE
SJMP L1                      ; CONTINUE
DELAY: MOV R5,#05
H3 :   MOV R4,#FF
H2 :    MOV R3,#FF
H1:     DJNZ R3,H1
DJNZ R4,H2
DJNZ R5,H3
RET
```

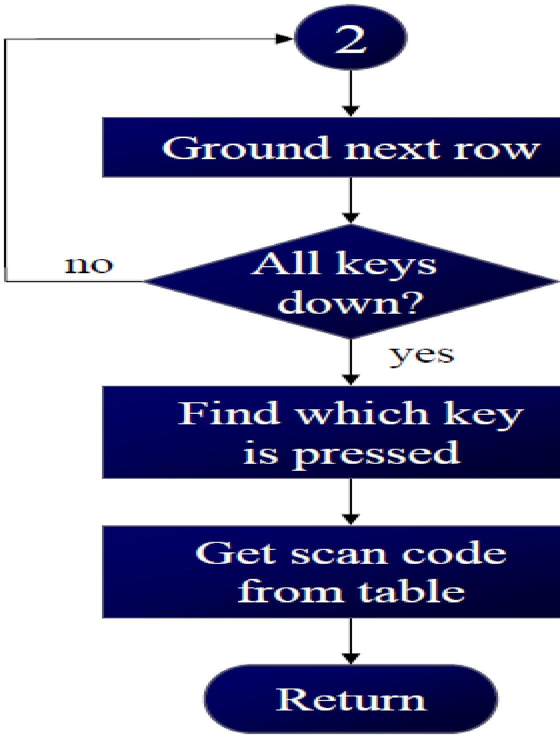
2. Interfacing of Matrix keyboard



Keyboards are organized in a matrix of rows and columns. The CPU accesses both rows and columns through ports. Therefore, with two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microcontroller. When a key is pressed, a row and a column make a contact. Otherwise, there is no connection between rows and columns.

Flow chart for matrix keyboard





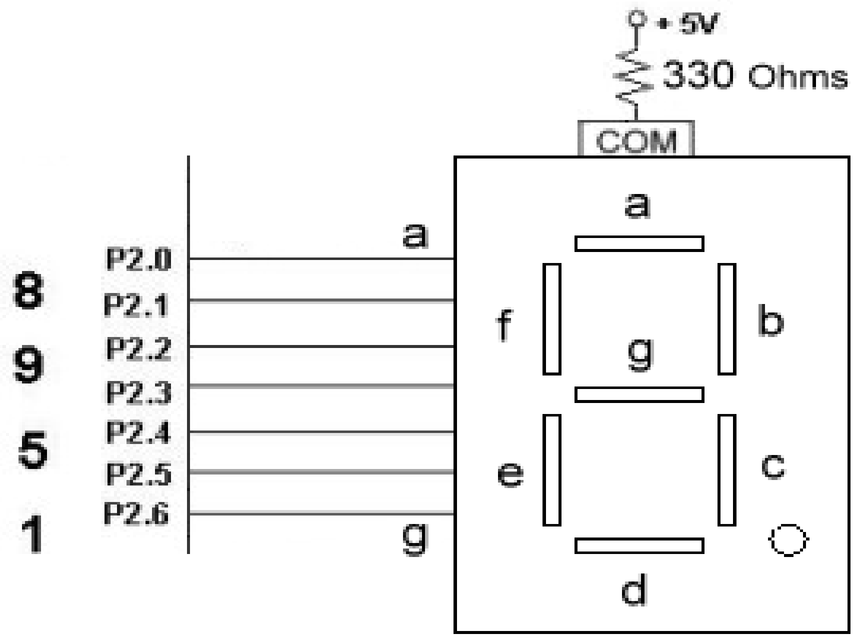
3. Interfacing of seven segment display (common anode)

Seven segment displays are of two types,*common cathode and common anode.*

In common cathode type , the cathode of all LEDs are tied together to a single terminal which is usually labeled as ‘com‘ and the anode of all LEDs are left alone as individual pins labeled as a, b, c, d, e, f, g & h (or dot) .

In common anode type, the anodes of all LEDs are tied together as a single terminal and cathodes are left alone as individual pins.

In common Anode in order to turn ON a segment the corresponding pin must be set to 0. And to turn it OFF it is set to 1.



Hex Number	Seven Segment conversion								Seven Segment equivalent
	dot	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	1	0	0	0	98

Write a program to display a counter on the seven segment LED(0 to 9)

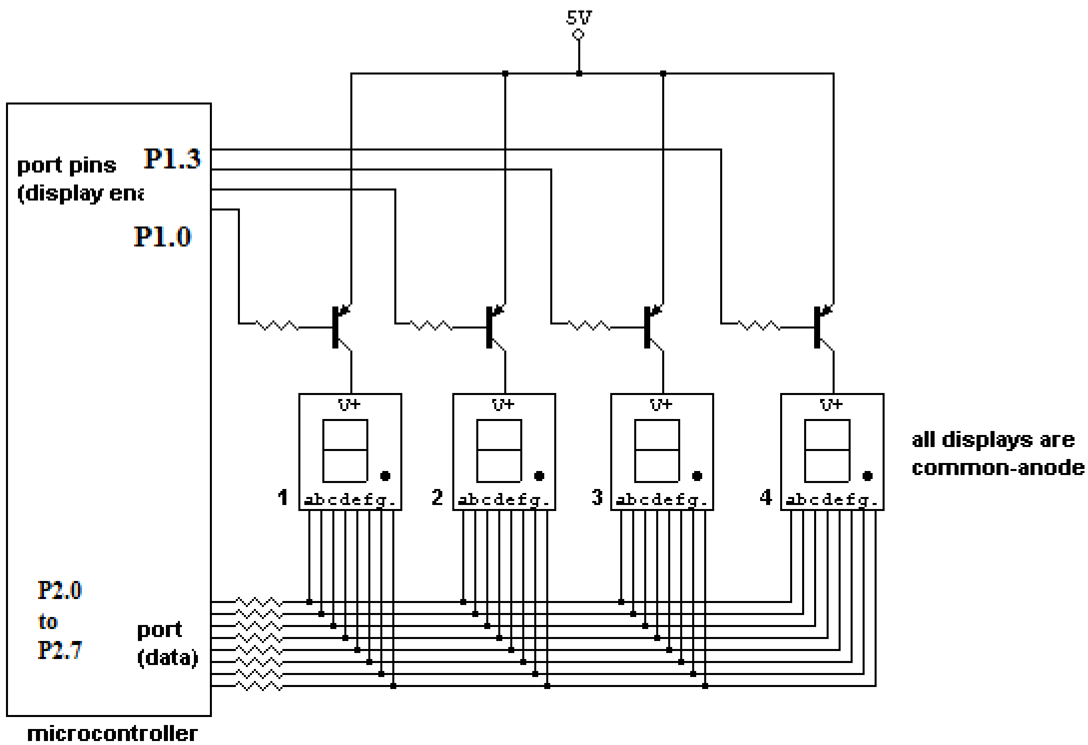
P2 is connected to the segments. We have to move the seven segment codes as per the table above to display 0 to 9

```
ORG 0000H
L1 : MOV P2.#0C0H;
ACALL DELAY
MOV P2.#0F9H;
ACALL DELAY
MOV P2.#0A4H;
ACALL DELAY
MOV P2.#0B0H;
ACALL DELAY
MOV P2.#99H;
ACALL DELAY
MOV P2.#92H;
ACALL DELAY
MOV P2.#82H;
ACALL DELAY
MOV P2.#0F8H;
ACALL DELAY
MOV P2.#80H;
ACALL DELAY
MOV P2.#98H;
ACALL DELAY
SJMP L1
```

Multiplexed display

Here only one 7-seg display is enabled at a given time through com signal. Inputs a-h are connected together to a common port. Hence total port pins needed are 8 +number of digits. In the figure shown above, which is for 4 digits, we require one port (8 lines) + 4 lines for common anode signal of each display.

The diagram below shows the interfacing of 4 digit common anode display with 8051.



TO DISPLAY 1234 CONTINUOUSLY

ORG 0000h

MOV P2,#0ff ;all data line at logic 1

MOV P1,#00h ; all select lines at logic 0

loop: SETB P1.0 ;SELECT DISP 1

MOV P2,#0F9h ; SEVEN SEGMENT CODE FOR 1

ACALL delay

CLR P1.0

SETB P1.1 ;SELECT DISP 2

MOV *P2,#0A4h* ;CODE FOR 2

ACALL delay

CLR P1.1

SETB P1.2 ;SELECT DISP 3

MOV P2,#0B0h ;CODE FOR 3

ACALL delay

CLR P1.2

SETB P1.3 ;SELECT DISP 4

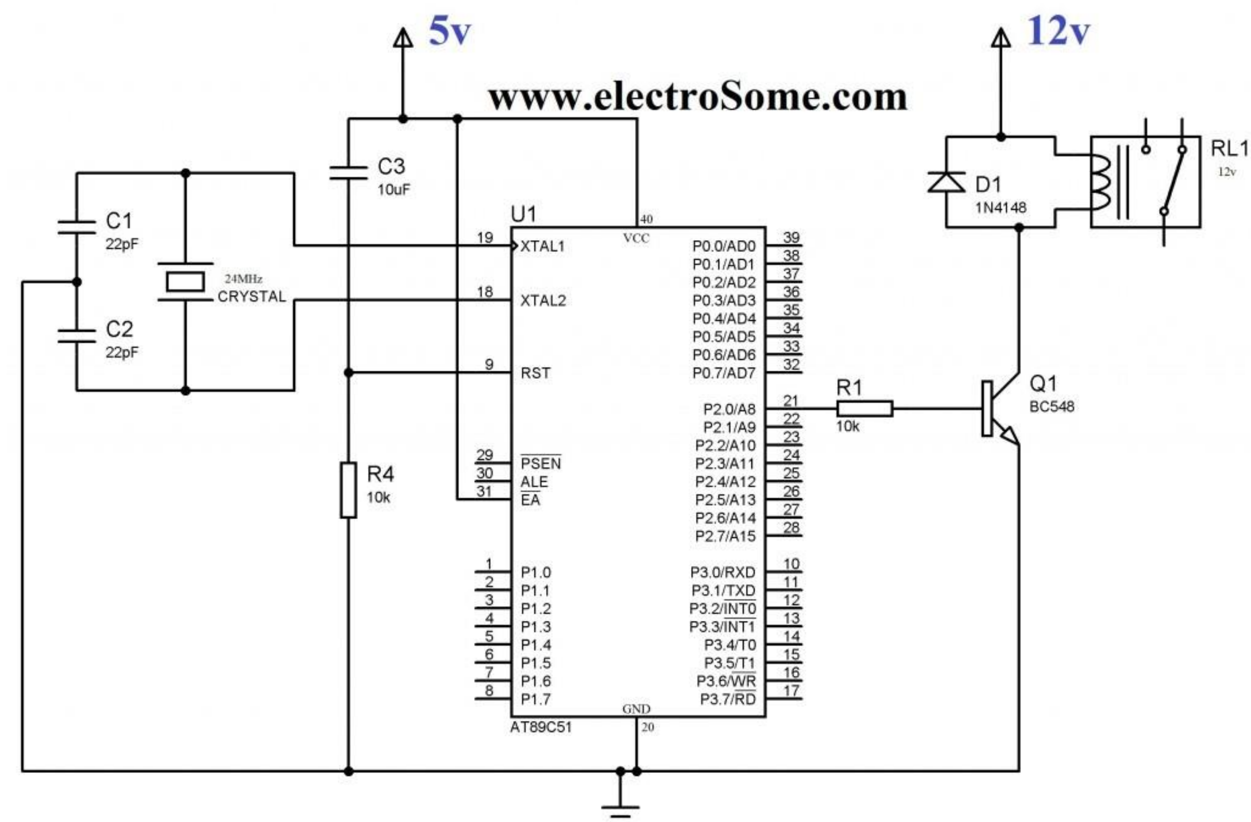
MOV P0,#99h ;CODE FOR 4

ACALL delay

CLR P1.3

SJMP loop

4. INTERFACING OF RELAY:



When P2.0 is high, the transistor Q1 is ON and the relay also will be on. When p2.0 is low, the transistor is off and hence relay will be off.

Program to On and OFF a relay, with a delay

ORG 0000H

UP: SETB P2.0

ACALL DELAY

CLR P2.0

ACALL DELAY

SJMP UP

DELAY: MOV R7,#100

L3: MOV R6,#255

L2:MOV R5,#255

L1: DJNZ R5,L1

DJNZ R6,L2

DJNZ R7,L3

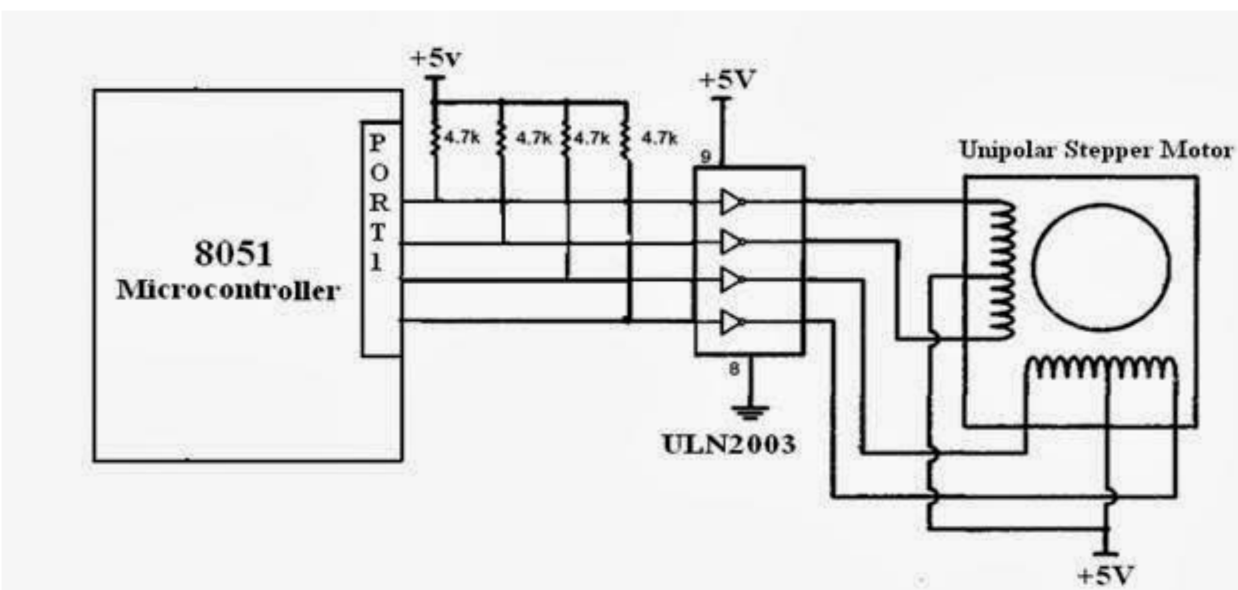
RET

END

5. **Interfacing of Stepper motor**

The stepper motor discussed here has a total of 6 leads, 4 leads representing the four stator windings, 2 commons for the center tapped leads.

As the sequence of power is applied as per the table given below to each stator winding, the rotor will rotate in clockwise direction. To rotate in anticlockwise, the sequence has to be given in reverse pattern.



P1.3	P1.2	P1.1	P1.0
A	C	B	D
1	0	0	1
1	1	0	0
0	1	1	0
0	0	1	0

Program:

Program for continuous rotation (clock wise)

org 0000H

```
       MOV P1, #09H
        ACALL DELAY
        MOV P1, #0CH
        ACALL DELAY
        MOV P1, #06H
        ACALL DELAY
        MOV P1, #03H
        ACALL DELAY
        SJMP MAIN
DELAY:
        MOV R7,#4
WAIT2:  MOV R6,#0FFH
WAIT1:  MOV R5,#0FFH
WAIT:   DJNZ R5,WAIT
        DJNZ R6,WAIT1
        DJNZ R7,WAIT2
        RET
        END
```

OR

ORG 0000H

LJMP MAIN

MAIN: MOV A, #99H

AGAIN: MOV P1,A

ACALL DELAY

RR A

SJMP AGAIN

```
DELAY:
        MOV R7,#4
WAIT2:  MOV R6,#0FFH
WAIT1:  MOV R5,#0FFH
WAIT:   DJNZ R5,WAIT
        DJNZ R6,WAIT1
        DJNZ R7,WAIT2
        RET
        END
```

Program to rotate anti clockwise

ORG 0000H

LJMP MAIN

MAIN: MOV A, #99H

AGAIN: MOV P1,A

ACALL DELAY

RL A

SJMP AGAIN

```
DELAY:
        MOV R7,#4
WAIT2:  MOV R6,#0FFH
WAIT1:  MOV R5,#0FFH
WAIT:   DJNZ R5,WAIT
        DJNZ R6,WAIT1
        DJNZ R7,WAIT2
        RET
        END
```

Program to rotate stepper motor for 360^0

Step angle = 1.8^0

Number of steps required = 360/1.8 = 200d =C8H

ORG 0000H

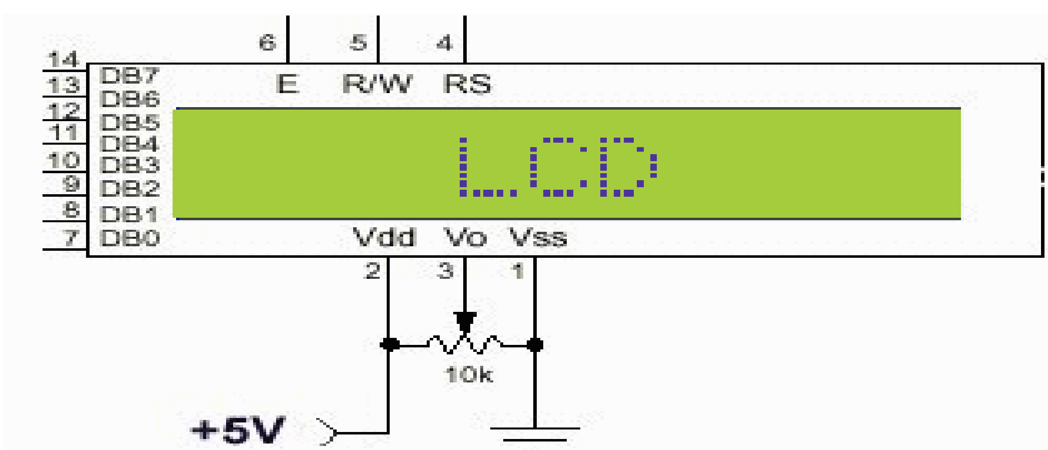
```
MOV R0,#0C8H

MOV A,#99H

L1: MOV P1,A

ACALL DELAY

RR A

DJNZ R0,L1

DELAY:
        MOV R7,#4
WAIT2:  MOV R6,#0FFH
WAIT1:  MOV R5,#0FFH
WAIT:   DJNZ R5,WAIT
        DJNZ R6,WAIT1
        DJNZ R7,WAIT2
        RET
        END

END
```

6. Interfacing of LCD

We use LCD display for the displaying messages in a more interactive way to operate the system or displaying error messages etc.

Commonly used *ALPHANUMERIC* displays are *1x16* (Single Line & 16 characters), *2x16* (Double Line & 16 character per line) & *4x20* (four lines & Twenty characters per line).

16×2 Liquid Crystal Display which will display the 32 characters at a time in two rows (16 characters in one row). Each character in the display is of size 5×7 pixel matrix. This matrix differs for different 16×2 LCD modules, if you take JHD162A, this matrix goes to 5×8. There are 16 pins in the LCD module, the pin configuration us given below



PIN NO	NAME	FUNCTION
1	VSS	Ground pin
2	VCC	Power supply pin of 5V
3	VEE	Used for adjusting the contrast commonly attached to the potentiometer.
4	RS	RS is the register select pin used to write display data to the LCD (characters), this pin has to be high when writing the data to the LCD. During the initializing sequence and other commands this pin should low.
5	R/W	Reading and writing data to the LCD for reading the data R/W pin should be high (R/W=1) to write the data to LCD R/W pin should be low (R/W=0)
6	E	Enable pin is for starting or enabling the module. A high to low pulse of about 450ns pulse is given to this pin.
7	DB0	DB0-DB7 Data pins for giving data(normal data like numbers characters or command data) which is meant to be displayed
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	LED+	Back light of the LCD which should be connected to Vcc
16	LED-	Back light of LCD which should be connected to ground.

LCD Command register

The LCD’s internal controller can accept several commands and modify the display accordingly. These commands are written to the command register of LCD by making RS as 0. These commands would be things like:

- Clear screen
- Return home
- Decrement/Increment cursor

After writing to the LCD, it takes some time for it to complete its internal operations. During this time, it will not accept any new commands or data. Hence we need to insert time delay between any two commands or data sent to LCD. The following table shows some of the common commands used in LCD.

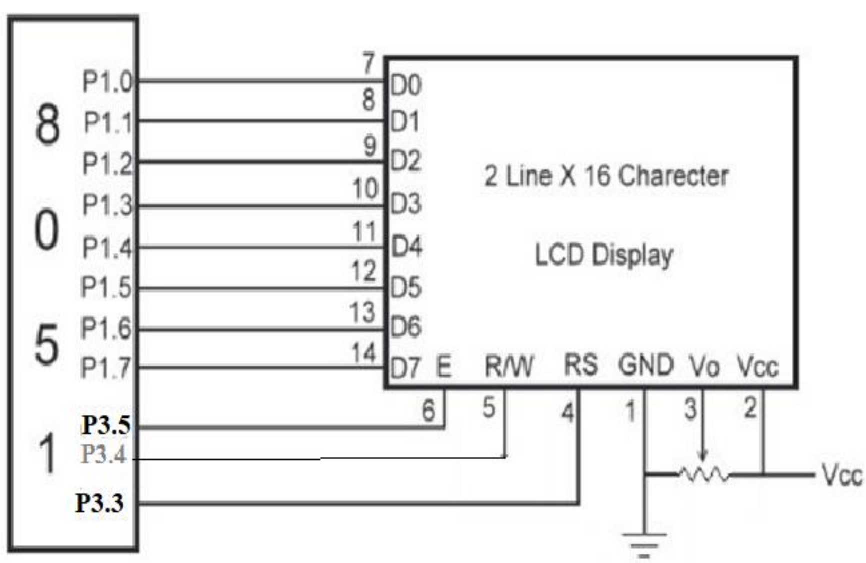
Table 4-8: LCD Command Codes

Code (hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor on
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
C0	Force cursor to beginning of 2nd line
38	2 lines and 5x7 matrix

Note: This table is extracted from Table 4-10.

LCD Data Display RAM

Display Data Ram (DDRAM) stores the display data. So when we have to display a character on LCD we basically write it into DDRAM. For a 2x16 LCD the DDRAM address for first line is from 80h to 8fh & for second line is 0c0h to 0cfh. So if we want to display 'H' on the 7^{th} position of the first line then we will write it at location 87h.



Programming LCD

Coming to the programming you should follow these steps:

- **STEP1:** Initialization of LCD by sending commands
- **STEP2:** Writing the data to LCD.

Initializing LCD

To initialize LCD to the 8051 the following instruction and commands are to be embed in to the functions

- 38H is used for 8-bit data initialization, 5X7 matrix display.
- 0EH for making LCD display on and cursor on.
- 01H for clearing the display of the LCD..
- 06H to shift cursor right
- 80H for positioning the cursor at first line, give the address as per the required position of first character as per the table given below

ROW	1st char addr	2nd char addr	3rd char addr				15th char addr	16th char addr
1st	80H	81H	82H	-----------	-----------	-----------	8EH	8FH
2nd	COH	C1H	C2H	-----------	-----------	-----------	CEH	CFH

Sending Commands to the LCD (LCDCMD Subroutine)

- Make R/W low.
- Make RS=0 if data byte is a command
- Place Command on P1 which will be written to command register of LCD.
- Pulse E from high to low.
- Repeat above steps for sending another command.

Writing the Data to the LCD (LCDDATA subroutine)

- Make R/W low.
- Make RS=1 if the data byte is a data to be displayed.

- Place data byte on P1 which will be written to the data register of LCD.
- Pulse E from high to low.
- Repeat above steps for sending another data.

```
 ORG 0000H
MOV A, #38H                    ; INITIALIZE, 2-LINES, 5X7 MATRIX.
LCALL  LCDCMD
LCALL DELAY
MOV A, #0EH                    ; LCD ON, CURSOR ON
LCALL  LCDCMD
LCALL DELAY
MOV A, #01H                    ; CLEAR LCD SCREEN
LCALL  LCDCMD
LCALL DELAY
MOV A, #06H                    ; SHIFT CURSOR RIGHT
LCALL  LCDCMD
LCALL DELAY
MOV A, #80H                    ; START ADDRESS
LCALL  LCDCMD
LCALL DELAY
MOV A,#‘M'
LCALL LCDDATA
LCALL DELAY
MOV A,#‘S'
LCALL LCDDATA
LCALL DELAY
MOV A,#‘B'
LCALL LCDDATA
LCALL DELAY
MOV A,#‘T'
LCALL LCDDATA
LCALL DELAY
MOV A,#‘E'
LCALL LCDDATA
LCALL DELAY
L1: SJMP L1

LCDCMD:
```

```
 MOV P1, A            ;MOVE ACC. LCDDATA TO PORT
 CLR P3.3              ;RS=0 FOR CMD
 CLR P3.4             ;RW=0 FOR WRITE
 SETB P3.5            ; H->L PULSE ON E

MOV R7,#255

HERE:    DJNZ  R7,HERE     ; small delay
CLR P3.5
 RET
```

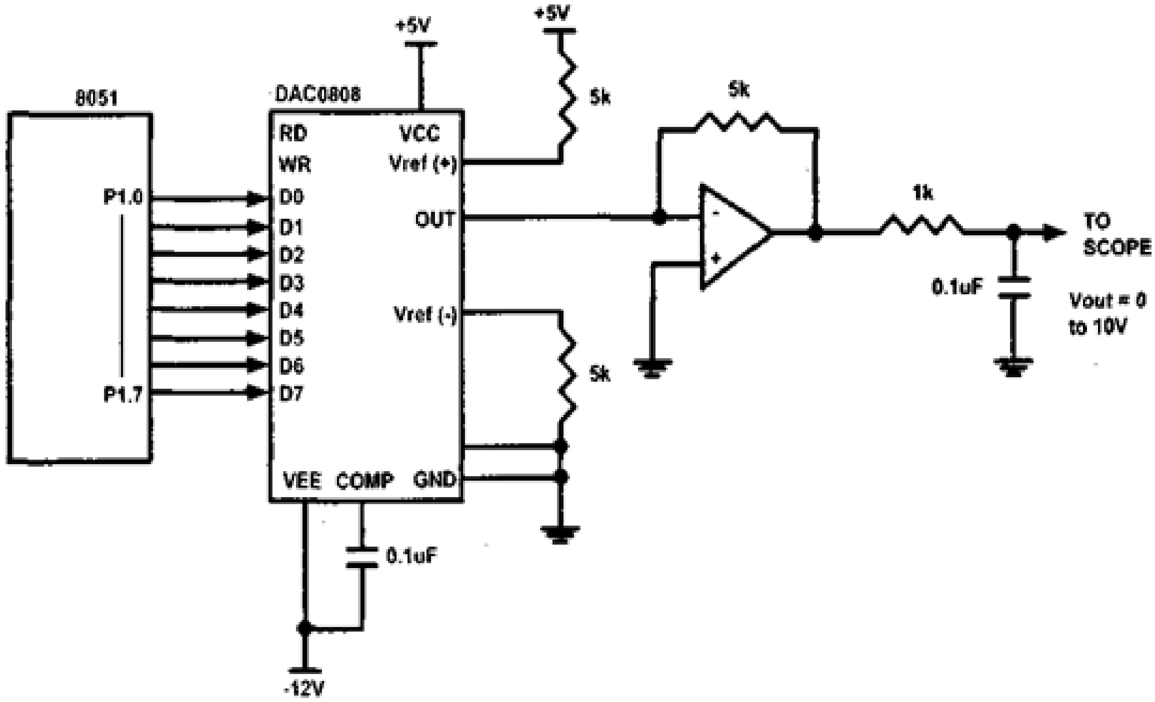
LCDDATA:

```
MOV P1, A           ;MOVE ACC. LCDDATA TO PORT
SETB P3.3           ;RS=1 LCDDATA
CLR P3.4            ;RW=0 FOR WRITE
SETB P3.5           ;H->L PULSE ON E

MOV R7,#255

HERE:    DJNZ  R7,HERE      ; small delay
CLR P3.5
RET
```

DELAY:

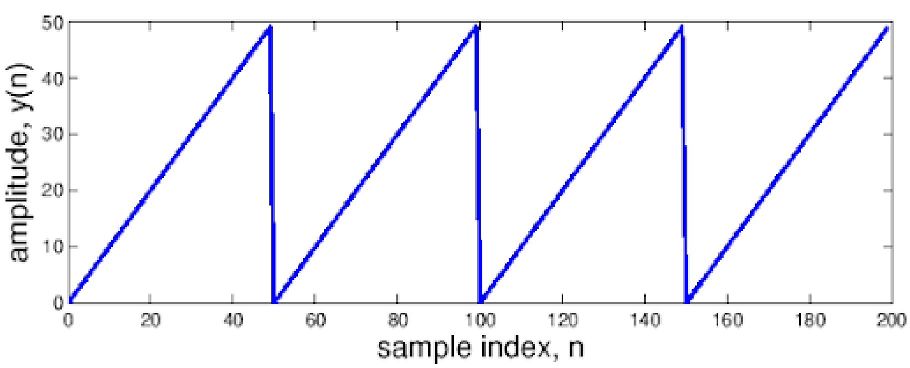
```
MOV R3,#50;

HERE1:  MOV R4,#255

HERE:    DJNZ  R4,HERE

DJNZ  R3,HERE1

 RET

 END
```

4.3 Digital to analog converter 0808



DAC 0808 is an 8 bit R-2R type digital to analog converter. The analog output is the current signal at Iout. This is converted to voltage using I to V converter.

Program for Ramp waveform generation



ORG 0000H

CLR A

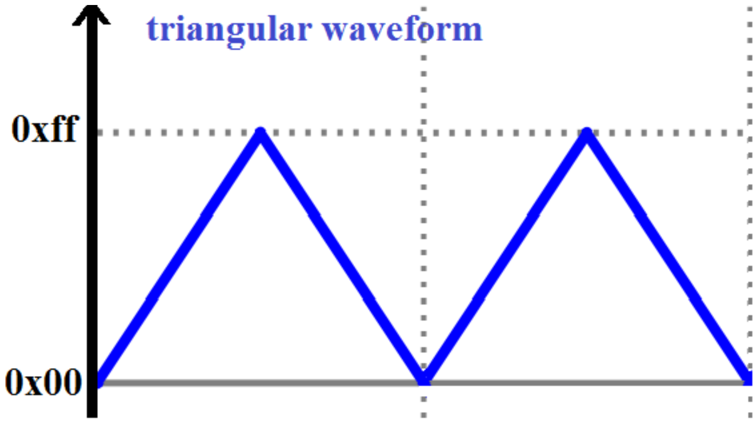
LOOP1: MOV P1,A

INC A

SJMP LOOP1

END

Program for Triangular wave generation



ORG 0000H

CLR A

LOOP1: MOV P1,A

INC A

CJNE A, #0FFH, LOOP1

LOOP2: DEC A

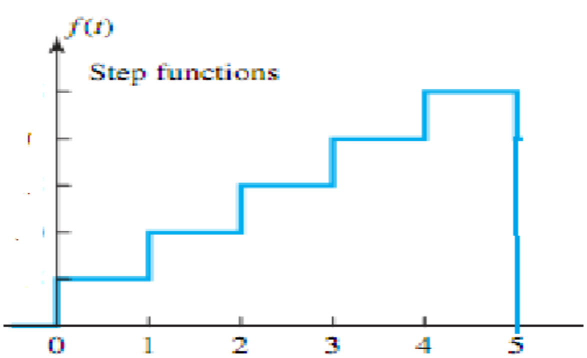
MOV P1,A

CJNE A, #00H,LOOP2

SJMP LOOP1

END

Staircase waveform generation



```
ORG 0000H
START:  CLR A
MOV P1,A
LCALL DELAY
REPEAT: ADD A,#51
MOV P1,A
LCALL DELAY
CJNE A,#255,REPEAT
MOV P1,A
LCALL  DELAY
SJMP START
```

Sine Wave

Considering output range of 0 to 10V,

Vout is represented in sine magnitude form as

Vout = 5V+5Vsinθ

For example for 30°

Vout = 5V+5Vsin30 = 5+5 X 0.5 = 7.5V

For calculating digital value to be given for 7.5V

256(max i/p) count corresponds to 10V(max o/p)

Hence 1V corresponds to 25.6 count change,

for 7.5V,   7.5 X 25.6 = 192D is the count

This way count is calculated for various angles and tabulated as below

θ	sinθ	Vout	Digital input
0	0	5	128
30	0.5	7.5	192
60	0.866	9.33	238
90	1.0	10	255

120	0.866	9.33	238
150	0.5	7.5	192
180	0	5	128
210	-0.5	2.5	64
240	0.866	0.669	17
270	-1.0	0	0
300	-0.866	0.669	17
330	-0.5	2.5	64
360	0	5	128

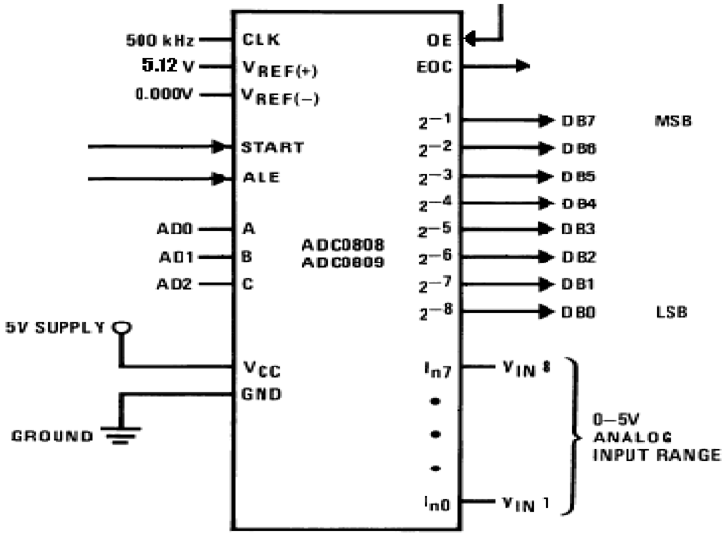
Program:

```
AGAIN:  MOV DPTR,#TABLE
        MOV R2,#COUNT
BACK:   CLR A
        MOVC A,@A+DPTR
        MOV P1,A
        INC DPTR
        DJNZ R2,BACK
        SJMP AGAIN
        ORG 300
TABLE:  DB 128,192,238,255,238,192 ;see Table 13-7
        DB 128,64,17,0,17,64,128
```

4.4 ADC 0808

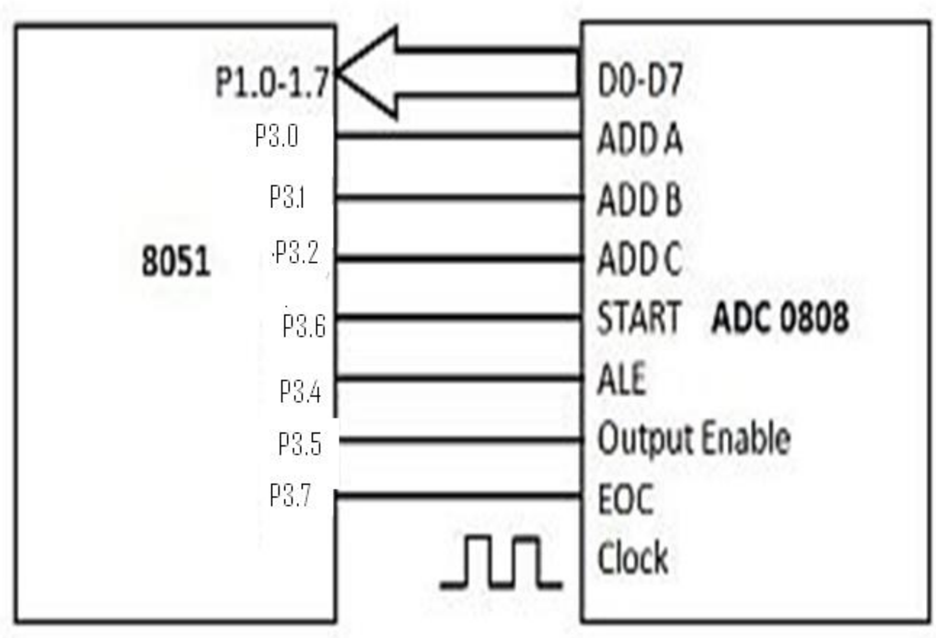
- ADC808/809 Chip with 8 analog channel. This means this kind of chip allows to monitor 8 different transducers.
- ADC804 has only ONE analog input: Vin(+).
- ALE: Latch in the address
- Start : Start of conversion (same as WR in 804)

- OE: output enable (same as RD in 804)
- EOC: End of Conversion (same as INTR in 804)



Channel	C B A
IN0	000
IN1	001
IN2	010
IN3	011
IN4	100
IN5	101
IN6	110
IN7	111

Interfacing of 0808 with 8051



Algorithm

1. Select the channel.
2. A Low – High transition on ALE to latch in the address.
3. A Low – High transition on START to reset SAR..
4. A High – Low transition on ALE.
5. A High – Low transition on start to start the conversion. (EOC should become low)
6. Wait for End of conversion (EOC) pin to become high.
7. Make Output Enable pin High.
8. Take Data from the ADC’s output
9. Make Output Enable pin Low.
10. Move the digital data to the desired location

Program

ADC_A BIT P3.0

ADC_B BIT P3.1

ADC_C BIT P3.2

ADC_START BIT P3.6

```
ADC_ALE BIT P3.4
ADC_OE BIT P3.5
ADC_EOC BIT P3.7
    ORG 0000H
    MOV P1,#0FFH                    ; P1 as input port
    SETB ADC_EOC                    ;P3.7 as input line
        CLR ADC_ALE
        CLR ADC_START
        CLR ADC_A                   ;
        CLR ADC_B                   ;SELECT CHANNEL 0
        CLR ADC_C                   ;
         SETB ADC_ALE               ; A Low – High transition on ALE to latch in the address.
        SETB ADC_START              ; A Low – High transition on START to reset SAR.
        LCALL DELAY_SMALL
        CLR ADC_ALE                 ;   A High – Low transition on ALE.
        CLR ADC_START               ; A High – Low transition on start to start the conversion.
                                            (EOC should become low)
        AGAIN: JB ADC_EOC, AGAIN
         AGAIN1: JNB ADC_EOC, AGAIN1     ; Wait for End of conversion (EOC) pin to
                                              Become high.
        SETB ADC_OE                 ;Make OE high
        MOV A,P1                    ; Read Data
        MOV P2,A
        CLR ADC_OE                  ; Clear OE
        LOOP:  SJMP LOOP
    END
```