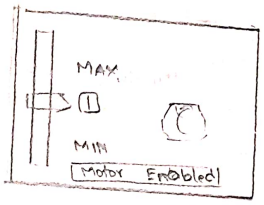


P_3	0x00
P_2	0x00
P_1	0x33
P_0	0x00

IP	0x00
IE	0x00
PCON	0x00
DPH	0x00
DPL	0x53
SP	0x0A

Pins	bits	
0xF5	0xF5	P_3
0xFF	0xFF	P_2
0xC0	0xC0	P_1
0xFF	0xFF	P_0

1	PSW	0	0	1	0	0	0	0	1

Date : 20/3/23

Experiment No. :

Aim: Program to interface stepper motor

Description: A stepper motor is an electromechanical device it converts electrical power into mechanical power. Also it is synchronous electric motor

Program

```
MOV TMOD, #50;
SETB TR1;
MOV DPL, #LOW(LEDCodes);
MOV DPH, #HIGH(LEDCodes);
CLR P3.4;1
CLR P3.3;1
again:
CALL setDirection;
MOV A,TL1;
CJNE A,#10,skip;
CALL clearTimer;
skip:
MOVC A,@A+DPTR;
MOV C,F0;
MOV ACC.7,C;
MOV P1,A;
JMP again;
SETDirection:
PUSH ACC;
PUSH 20H;
CLR A;
MOV 20H,#0;
MOVC P2.0;
MOV ACC.0,C;
MOV C,F0;
MOV 0,C;
CJNE A,20H,changeDir;
JMP finish; changeDir:
CLR P3.0;1 CLR P3.1;1
CALL clearTimer; MOV C,P2.0;
MOV F0.C; CPLC;
MOV P3.1,C;
finish:
POP 20H;
POP ACC;
RET;
clearTimer:
CLR A; CLR TR1;
MOV TL1,#0; SETB TR1;
RET
LEDCodes:
DB 11000000B, 11111001B, 10100100B,
10110000B, 10010010B, 10000010B,
11111000B, 10000000B, 10010000B
```

Aim: To simulate a basic elevator system using 8051 controller with LED's. where LED's represent different floors. The elevator moves up and down between 4 floors and each floor is indicated by an LED connected to port1

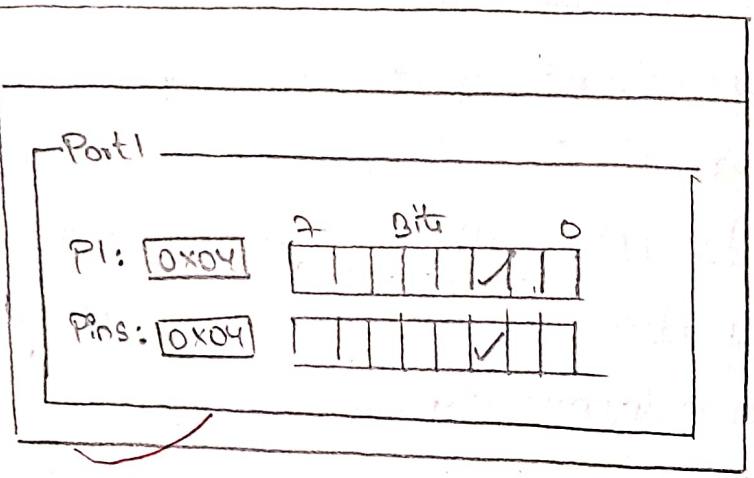
Hardware requirements: 8051 microcontroller, 4 LED's connected to port 1.0 to 1.3

Software requirements: Keiluvision5, protues for graphical simulation

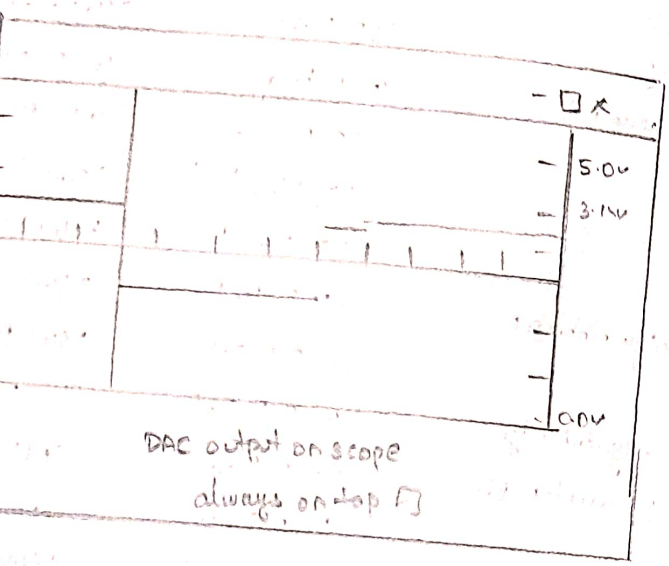
Working principle: Elevator starts at floor '0'. Elevator moves upto the above floors P1.3 LED will on, when elevator reaches top floor. It changes its direction downwards. Changing the direction of the elevator continues in infinite loop

Code:

```
#include<Reg51.h>
void delay_ms(unsigned int ms)
{ unsigned int i,j;
   for(i=0;j=ms;i++)
   for(j=0;j=1275;j++)
}
main()
{ unsigned char floors = 0;
      bit direction =0;
   while(1)
      { P1=0x00;
        P1<<floor;
        delay_ms(1000)
        if(floor==3)
```




```
        direction = 0;
    if (floor == 0)
        direction = 1;
    else if (direction)
        floor++
    else (direction)
        floor--;
    }
}
```



DAC output on scope
always on top

Date : 12/9/25

Aim: Program to interface ADC (Analog to Digital converter)

```
AP ORG 0;
JMP main;
ORG 3;
JMP extOISR;
ORG 0BH;
JMP timer OISR;
ORG 30H;
main:
SETB IT0;
SET EX0;
CLR P0.7;
MOV TMOD,#2;
MOV TH0,#-50;
;1
MOV TL0,#-50;
SETB TR0;
SETB ET0;
SETB EA;
JMP $;
;
;
timer OISR:1
CLR P3.6;
SETB P3.6;
RETI;
;
extOISR
CLR P3.7;
MOV P1,P2;
SETB P3.7;
RETI;
```

Date :

Aim: Program to interface LCD to display A-BC

Program ; put data in RAM

```
MOV 30H, #'A'
MOV 31H, #'B'
MOV 32H, #'C'
MOV 33H, #0;;;
CLR P1.3;; clear RS
CLR P1.7;1
CLR P1.6;1
SETB P1.5;1
CLR P1.4;1 high nibble set
SETB P1.2;1
CLR P1.2;1 negative edge
CALL delay;;; wait for BF
SETB P1.2;1
CLR P1.2;1
; SETB P1.2;
CLR P1.2;; negative edge
CALL delay;;
CLR P1.7;1
CLR P1.6;1
CLR P1.5;1
CLR P1.4;1
SETB P1.2;1
SETB P1.5;1
SETB P1.2;1
CLR P1.2;1
CLR P1.4;1
SETB P1.2;1
CLR P1.2;1
CALL delay;;
SETB P1.3;
MOV R1, #30H;
loop:
MOV A,@R1; 32 finish;
INC R1; JMP loop;
finish:
JMP$
Send character:
MOV C, ACC.7;1
MOV P1.7, C;1
MOV C, ACC.6;1
MOV P1.6, C;1
MOV C, ACC.4;1
MOV P1.4, C;1
SETB P1.2;1
CLR P1.2;1
MOV C, ACC.3;1
MOV P1.5, C;1
MOV C, ACC.0;1
MOV P1.4, C;1
SETB P1.2;1
CALL delay;
delay:
MOV R0, #50
DJNZ R0, $
RET
```

O/P: