EXPERIMENT NO.1

Aim:

A. To study Memory organization of 8051 microcontroller .
B. Write 8051Program for simple programs on memory transfer.

Objective:

1. To be familiar with data transfer between 8051 microcontroller and internal / external memory

Apparatus: PC. Keil simulator

Theory:

Memory Organization

The 8051 has two types of memory and these are Program Memory and Data Memory. Program Memory (ROM) is used to permanently save the program being executed, while Data Memory (RAM) is used for temporarily storing data and intermediate results created and used during the operation of the microcontroller. Depending on the model in use (we are still talking about the 8051 microcontroller family in general) at most a few Kb of ROM and 128 or 256 bytes of RAM is used. However All 8051 microcontrollers have a 16-bit addressing bus and are capable of addressing 64 kb memory. It is neither a mistake nor a big ambition of engineers who were working on basic core development. It is a matter of smart memory organization which makes these microcontrollers a real "programmers' goody".

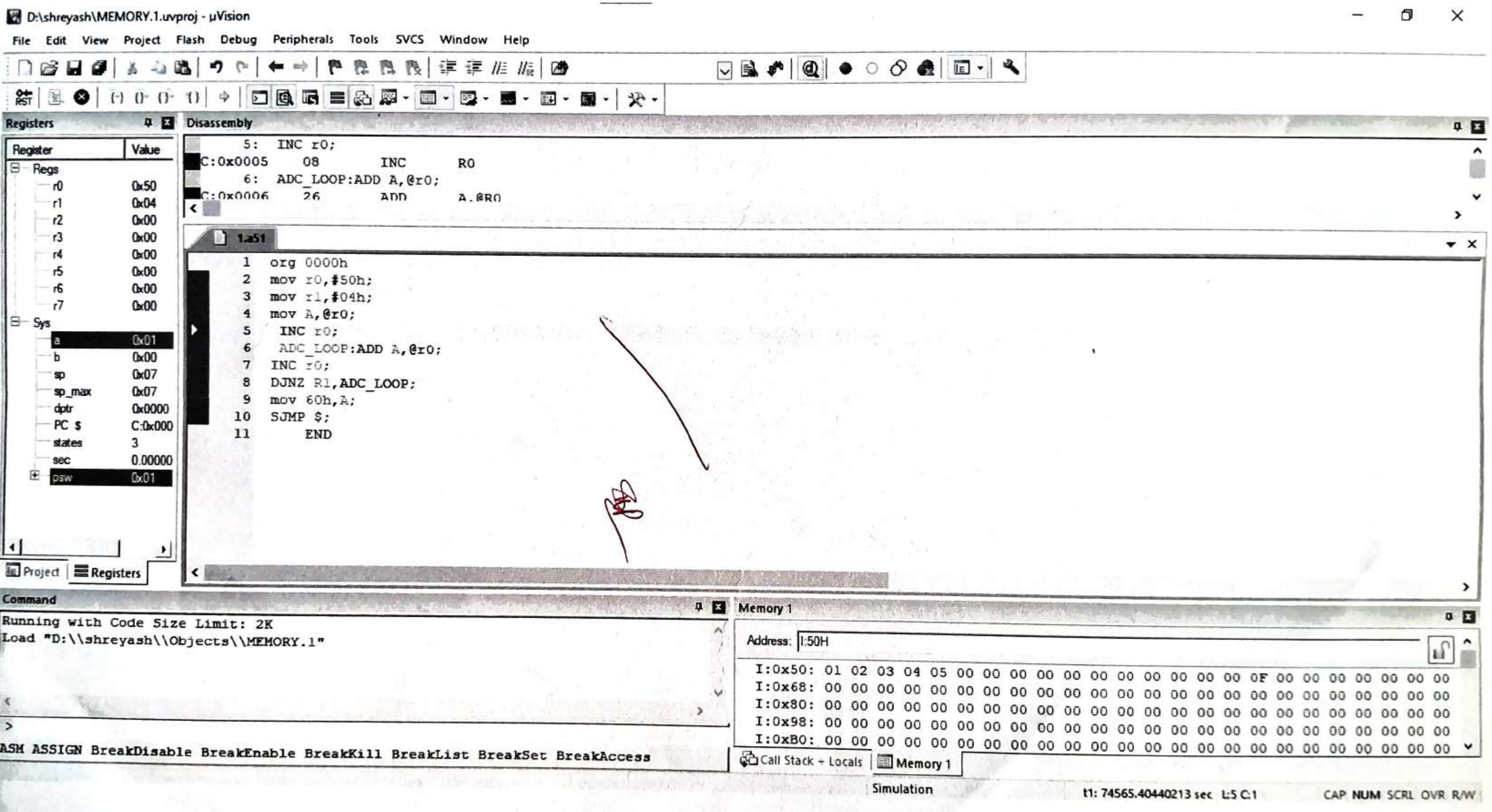
Program Memory

The first models of the 8051 microcontroller family did not have internal program memory. It was added as an external separate chip. These models are recognizable by their label beginning with 803 (for example 8031 or 8032). All later models have a few Kbyte ROM embedded. Even though such an amount of memory is sufficient for writing most of the programs, there are situations when it is necessary to use additional memory as well. A typical example is so called lookup tables. They are used in cases when equations describing some processes are too complicated or when there is no time for solving them. In such cases all necessary estimates and approximates are executed in advance and the final results are put in the tables (similar to logarithmic tables).

EA=0 In this case, the microcontroller completely ignores internal program memory and executes only the program stored in external memory.

EA=1 In this case, the microcontroller executes first the program from built-in ROM, then the program stored in external memory.

In both cases, P0 and P2 are not available for use since being used for data and address transmission. Besides, the ALE and PSEN pins are also used.


D:\shreyash\MEMORY.1.uvproj - µVision

Register	Value
Regs	
r0	0x50
r1	0x04
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x01
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x000
states	3
sec	0.00000
psw	0x01

```
     5:   INC r0;
C:0x0005     08         INC      R0
     6:   ADC_LOOP:ADD A,@r0;
C:0x0006     26         ADD      A,@R0
```

1.a51

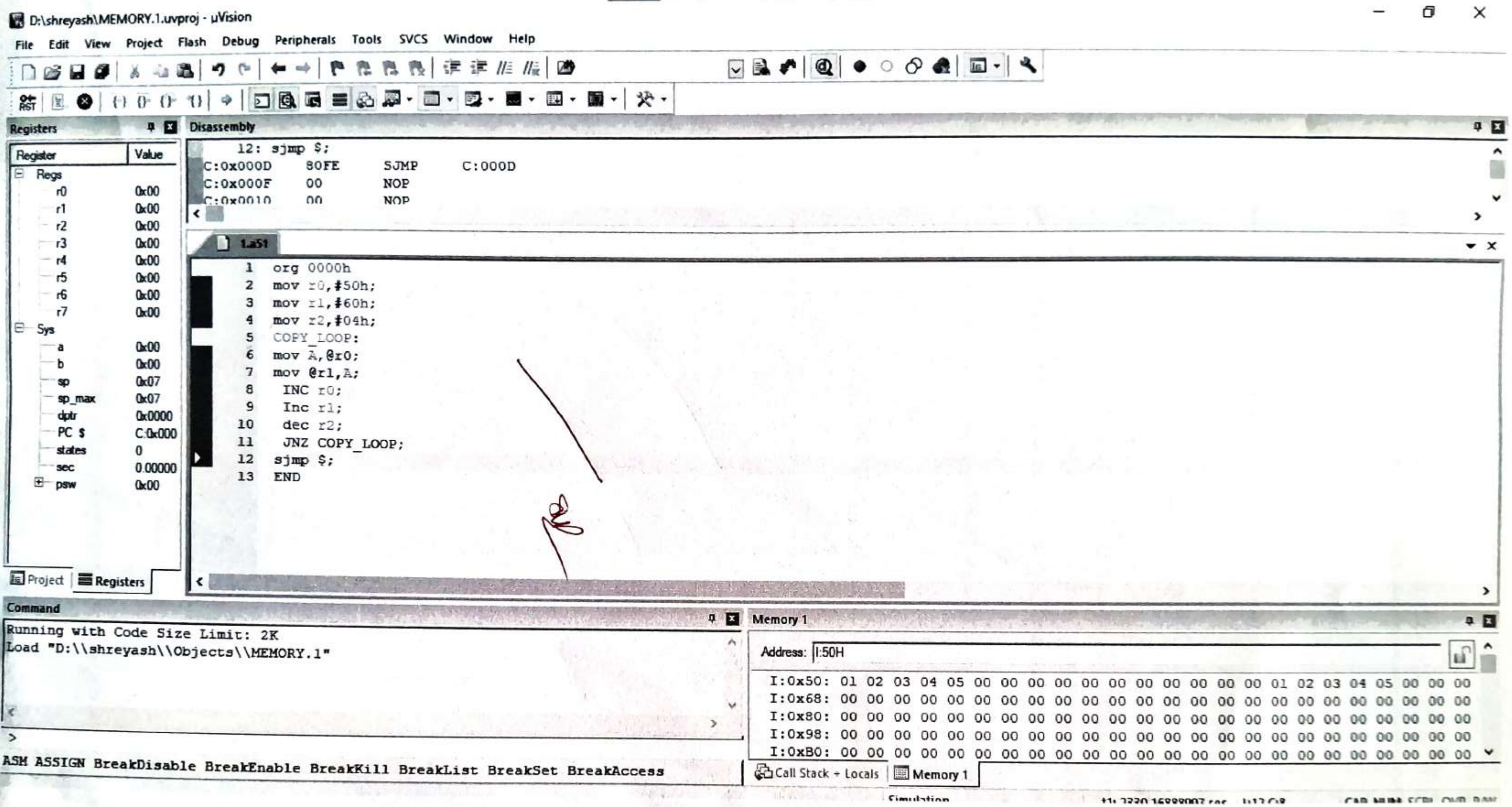
```
org 0000h
mov r0,#50h;
mov r1,#04h;
mov A,@r0;
 INC r0;
 ADC_LOOP:ADD A,@r0;
INC r0;
DJNZ R1,ADC_LOOP;
mov 60h,A;
SJMP $;
    END
```

```
Running with Code Size Limit: 2K
Load "D:\\shreyash\\Objects\\MEMORY.1"
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

Address: I:50H

```
I:0x50: 01 02 03 04 05 00 00 00 00 00 00 00 00 00 00 00 0F 00 00 00 00 00 00 00
I:0x68: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x98: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

t1: 74565.40440213 sec   L:5 C:1   CAP NUM SCRL OVR R/W


D:\shreyash\MEMORY.1.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
1.a51
Project
Registers
Command
Memory 1
Call Stack + Locals
Memory 1
Simulation

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x000
states	0
sec	0.00000
psw	0x00

```
    12: sjmp $;
C:0x000D    80FE    SJMP    C:000D
C:0x000F    00      NOP
C:0x0010    00      NOP
```

```
org 0000h
mov r0,#50h;
mov r1,#60h;
mov r2,#04h;
COPY_LOOP:
mov A,@r0;
mov @r1,A;
 INC r0;
 Inc r1;
 dec r2;
 JNZ COPY_LOOP;
sjmp $;
END
```

```
Running with Code Size Limit: 2K
Load "D:\\shreyash\\Objects\\MEMORY.1"
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

```
Address: I:50H
I:0x50: 01 02 03 04 05 00 00 00 00 00 00 00 00 00 00 00 01 02 03 04 05 00 00 00
I:0x68: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x98: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0xB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

[illegible]

9.Stop

Conclusion:
The experiment helped in Understanding the memory organisation of 8051 microcontroller by writing simple program for memory transfer. We gained practical knowledge of internal and external memory access and data movement between various memory type using 8051.

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
10	10	10	08	07	45
Sub Teacher Sign: [illegible]					

EXPERIMENT NO. 02

:

A. To study Port Structure of 8051 microcontroller
B. To write Embedded C Program for Parallel port interacting of LEDS—Different programs(flashing, Counter, BCD, HEX, Display of Characteristic)

ective:

. To be familiar with software delay calculations using 8051 microcontroller
2. To find functions of P0,P1,P2,P3 of 8051 Microcontroller

paratus: PC, Keil simulator

eory: Examining Figure 2.1, note that of the 40 pins, a total of 32 pins are set aside for the four ts P0, P1, P2, and P3 where each port takes 8 pins. The rest of the pins are designated as V_{cc}, GND, AL1, XTAL2, RST, EA, PSEN, and ALE.

PDIP/Cerdip

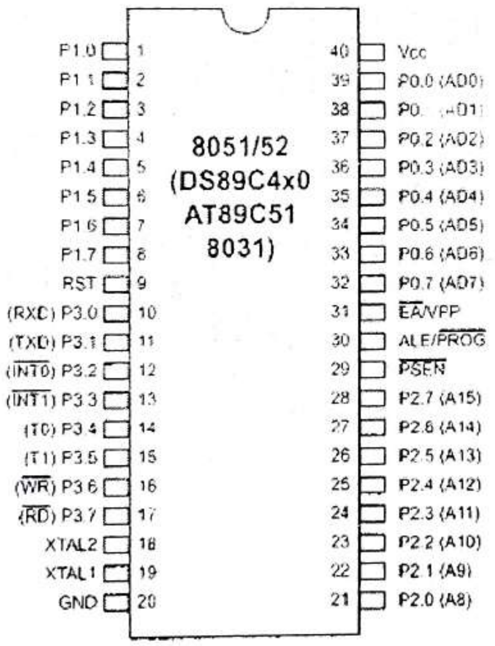


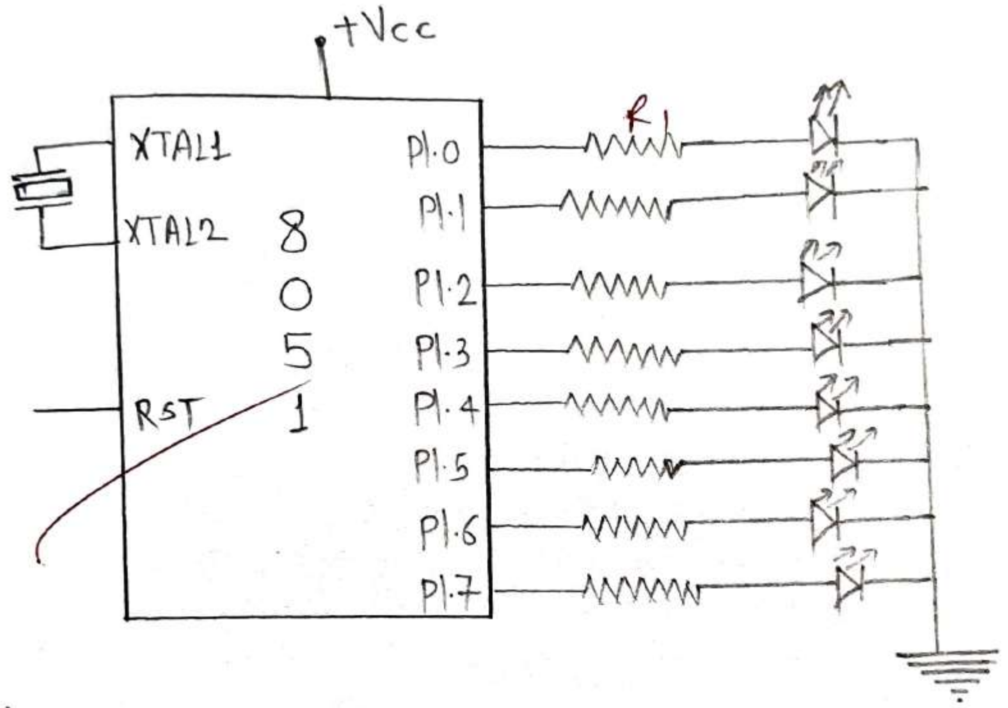
Figure 2.1 Pin Diagram of 8051 Microcontroller

Of these pins, six (V_{cc}, GND, XTAL1, XTAL2, RST, and EA) are used by all members of the 8051 and 8031 families. In other words, they must be connected in order for the system to work, regardless of whether the microcontroller is of the 8051 or 8031 family. The other two pins, PSEN and ALE, are used mainly in 8031-based systems. We first describe the function of each pin. Ports are discussed separately.

Vcc- Pin 40 provides supply voltage to the chip. The voltage source is +5V.

GND-Pin 20 is the ground.

Interfacing Diagram: LED to 8051



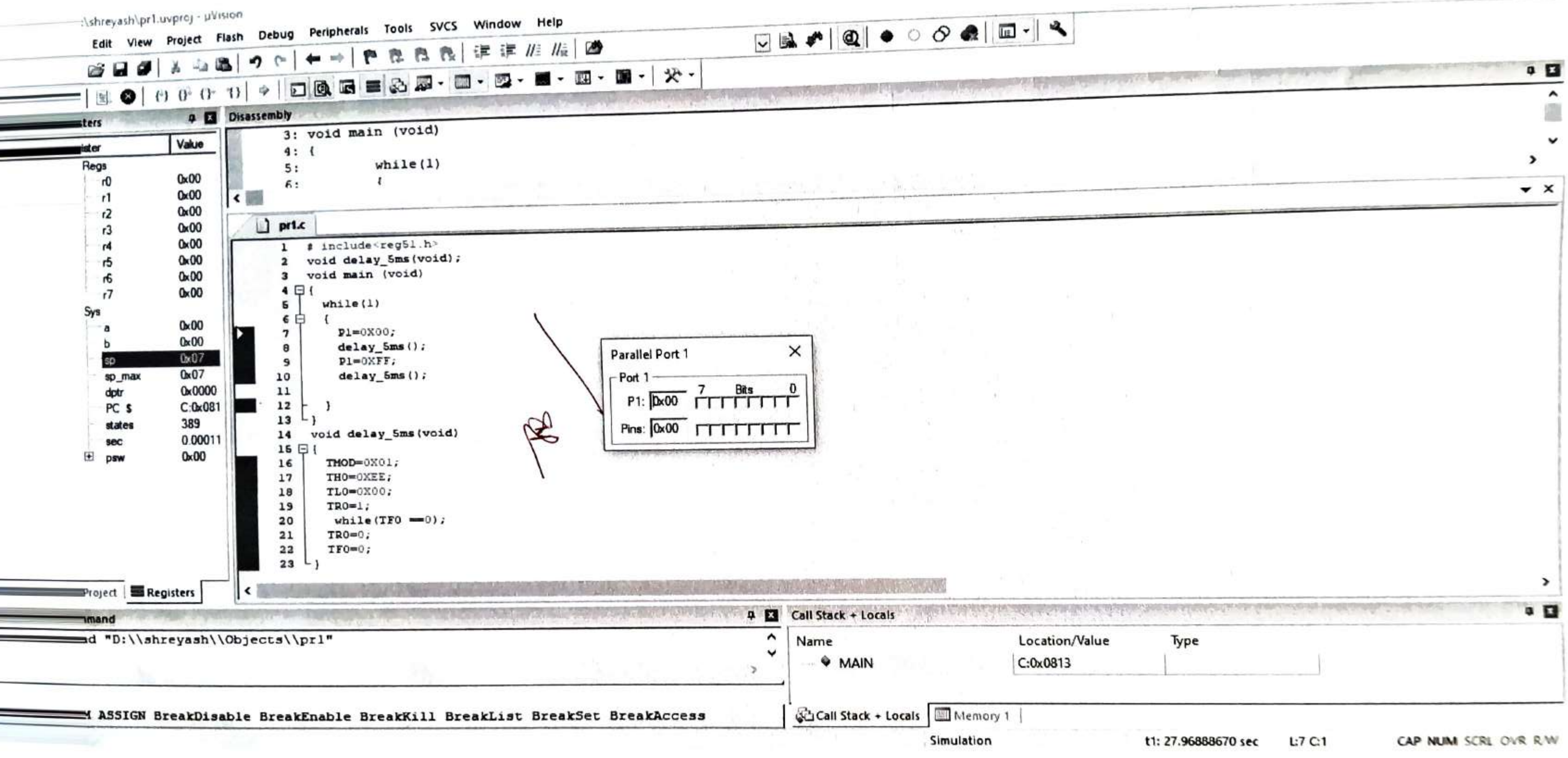
Algorithm:

A] Main Program:

1. Initialize port as output
2. Set all port pins to logic high
3. Call Delay
4. Clear all Port pins
5. Call Delay
6. Jump to step 2

B] Delay Subroutine:

1.Initialize Timer 0/1 selecting required Mode
2.Load value of TH0 and TL0
3.Clear TF flag
4. Start timer
5. Check TF flag .If it is not set keep checking.
6.Clear TF and stop timer
7. Return back to Main Program



:\shreyash\pr1.uvproj - µVision

Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Disassembly

```
     3: void main (void)
     4: {
     5:          while(1)
     6:          {
```

pr1.c

```
# include<reg51.h>
void delay_5ms(void);
void main (void)
{
  while(1)
  {
    P1=0X00;
    delay_5ms();
    P1=0XFF;
    delay_5ms();
  }
}
void delay_5ms(void)
{
  THOD=0X01;
  TH0=0XEE;
  TL0=0X00;
  TR0=1;
   while(TF0 ==0);
  TR0=0;
  TF0=0;
}
```

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x081
states	389
sec	0.00011
psw	0x00

Parallel Port 1

Port 1

P1: 0x00   7 Bits 0

Pins: 0x00

[illegible]

Command

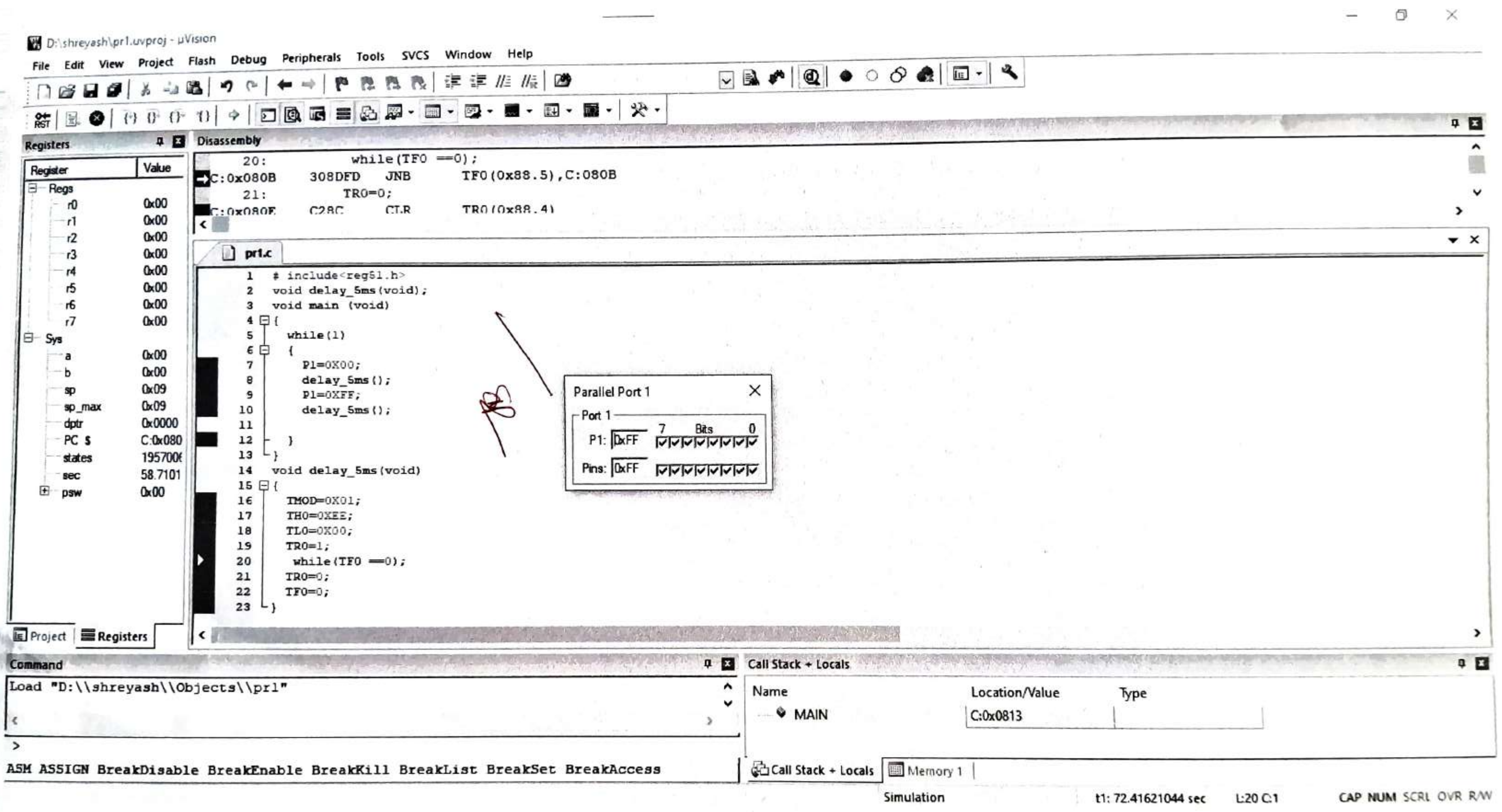
```
d "D:\\shreyash\\Objects\\pr1"
```

```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

Call Stack + Locals

Name	Location/Value	Type
MAIN	C:0x0813	

Simulation   t1: 27.96888670 sec   L:7 C:1   CAP NUM SCRL OVR R/W


D:\shreyash\pr1.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
pr1.c
Project
Registers
Command
Call Stack + Locals
Memory 1

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x09
sp_max	0x09
dptr	0x0000
PC $	C:0x080
states	195700(
sec	58.7101
psw	0x00

```
    20:          while(TF0 ==0);
C:0x080B    308DFD   JNB      TF0(0x88.5),C:080B
    21:           TR0=0;
C:0x080E    C28C     CLR      TR0(0x88.4)
```

```
# include<reg51.h>
void delay_5ms(void);
void main (void)
{
  while(1)
  {
    P1=0X00;
    delay_5ms();
    P1=0XFF;
    delay_5ms();

  }
}
void delay_5ms(void)
{
 TMOD=0X01;
 TH0=0XEE;
 TL0=0X00;
 TR0=1;
  while(TF0 ==0);
 TR0=0;
 TF0=0;
}
```

Parallel Port 1
Port 1
P1: 0xFF 7 Bits 0 ✓✓✓✓✓✓✓✓
Pins: 0xFF ✓✓✓✓✓✓✓✓

[illegible]

```
Load "D:\\shreyash\\Objects\\pr1"
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

Name	Location/Value	Type
MAIN	C:0x0813	

Simulation   t1: 72.41621044 sec   L:20 C:1   CAP NUM SCRL OVR R/W

Conclusion:

We studied the port structure of 8051 microcontroller and verified the functions of ports P_0 - P_3 simple embedded C program were executed for LED interfacing counters and display operation.

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
10	10	10	08	08	46
Sub Teacher Sign: [illegible]					

EXPERIMENT NO. 03 A

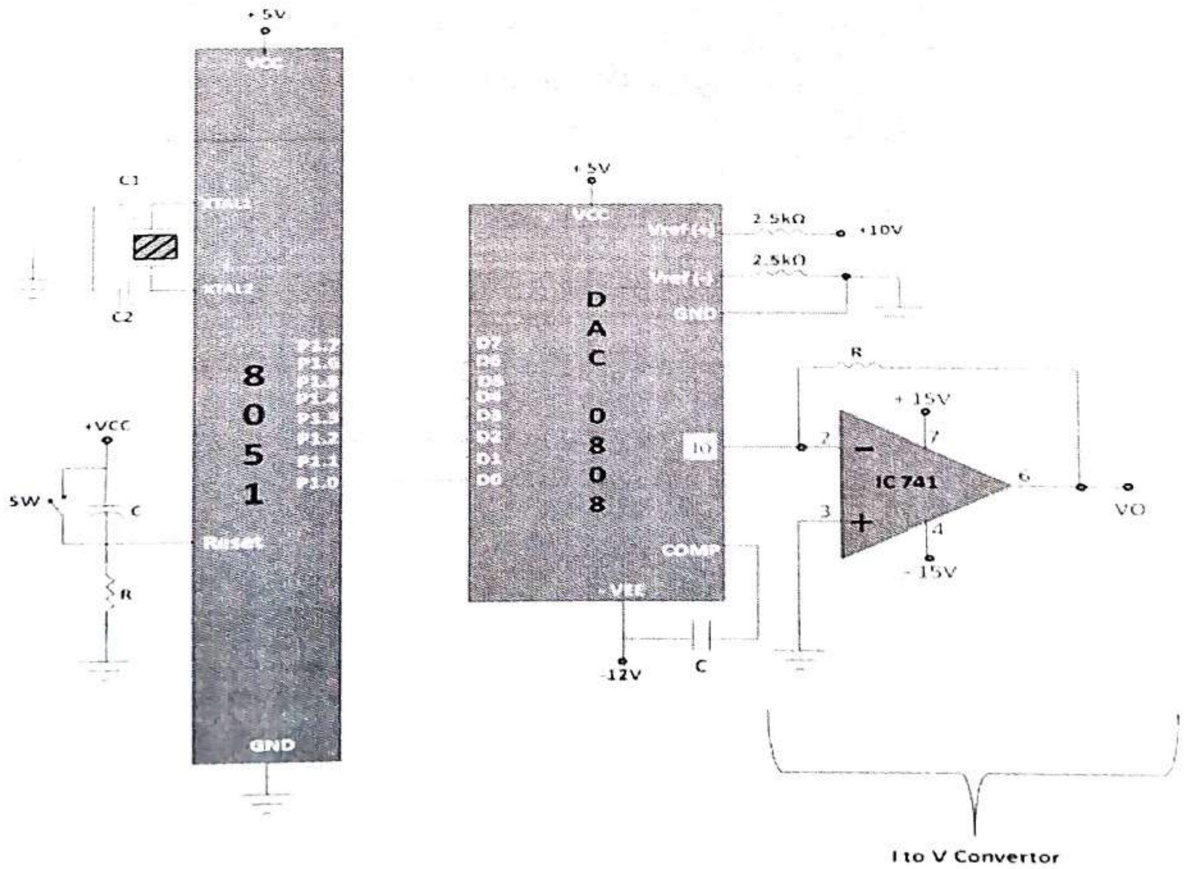
Aim: Waveform Generation using DAC 0808

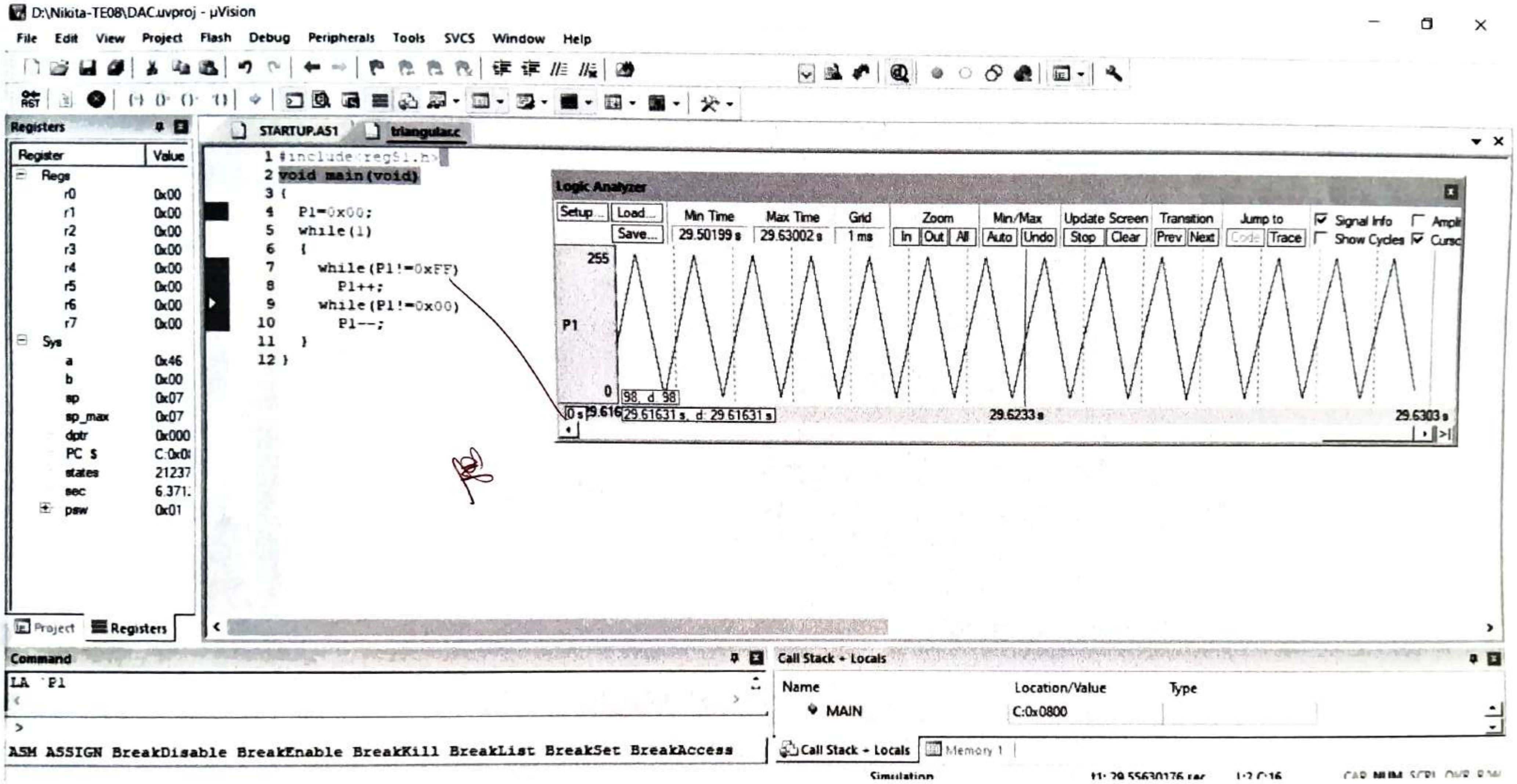
Objective:

1. To write assembly program to interface DAC 0808 with microcontroller 8051 to generate square wave and triangular wave.

Apparatus: PC. keil simulator. 8051 controller kit. LED Kit. Flash magic software

Theory: Microcontroller are used in wide variety of applications like for measuring and control of physical quantity like temperature, pressure, speed, distance, etc.
In these systems microcontroller generates output which is in digital form but the controlling system requires analog signal as they don't accept digital data thus making it necessary to use DAC which converts digital data into equivalent analog voltage.
In the figure shown, we use 8-bit DAC 0808. This IC converts digital data into equivalent analog Current. Hence we require I to V converter to convert this current into equivalent voltage.





```
#include<reg51.h>
void main(void)
{
  P1=0x00;
  while(1)
  {
    while(P1!=0xFF)
      P1++;
    while(P1!=0x00)
      P1--;
  }
}
```

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x46
b	0x00
sp	0x07
sp_max	0x07
dptr	0x000
PC $	C:0x0
states	21237
sec	6.371:
psw	0x01

Min Time	Max Time	Grid
29.50199 s	29.63002 s	1 ms

```
P1   255
     0    98, d: 98
0 s  29.616  29.61631 s, d: 29.61631 s      29.6233 s      29.6303 s
```

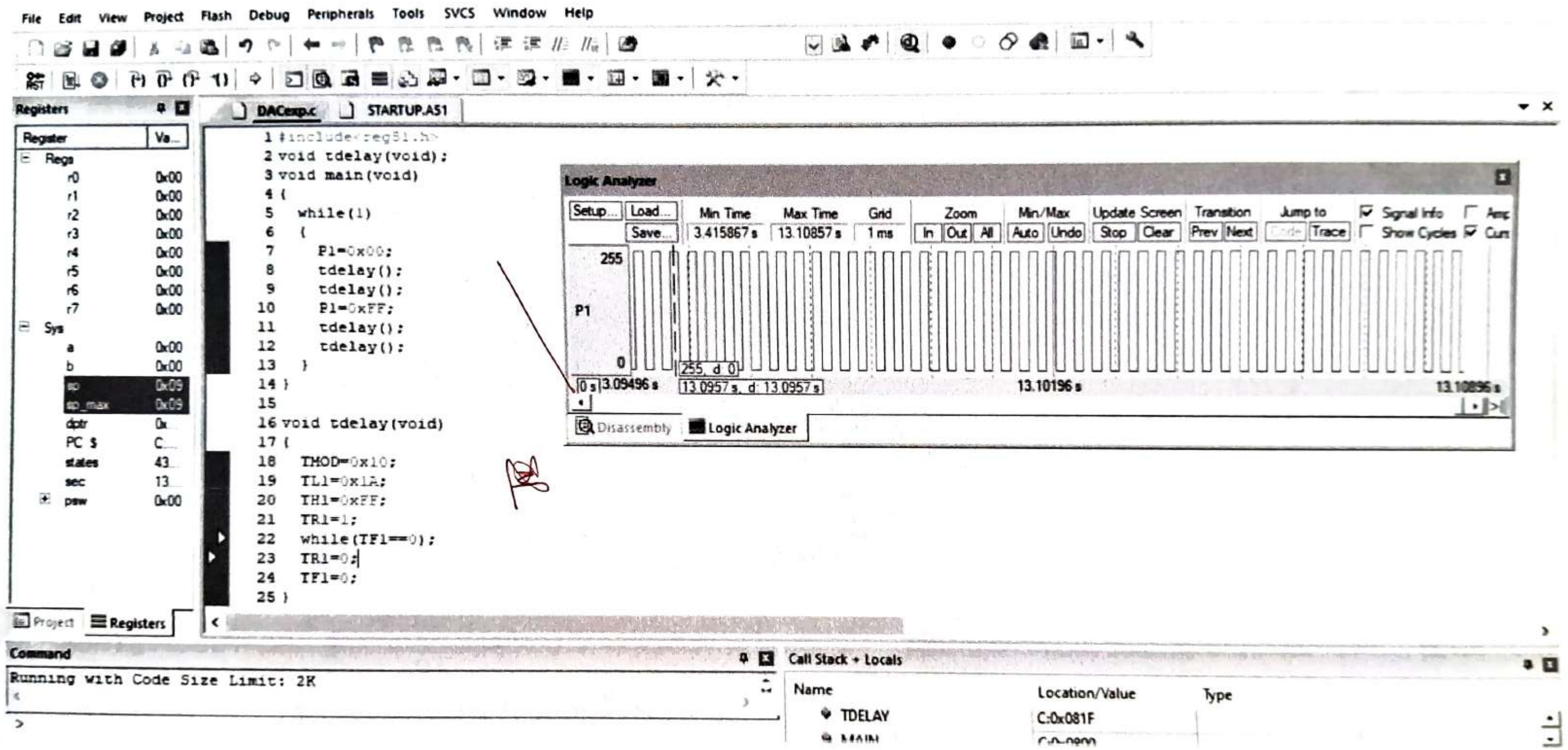
Command

```
LA `P1
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

Name	Location/Value	Type
MAIN	C:0x0800	

Simulation   t1: 29.55630176 sec   L:2 C:16   CAP NUM SCRL OVR R/W

[illegible]



```
#include<reg51.h>
void tdelay(void);
void main(void)
{
  while(1)
  {
    P1=0x00;
    tdelay();
    tdelay();
    P1=0xFF;
    tdelay();
    tdelay();
  }
}

void tdelay(void)
{
 TMOD=0x10;
 TL1=0x1A;
 TH1=0xFF;
 TR1=1;
 while(TF1==0);
 TR1=0;
 TF1=0;
}
```

Register	Va...
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x09
sp_max	0x09
dptr	0x...
PC $	C...
states	43...
sec	13...
psw	0x00

```
Running with Code Size Limit: 2K
```

Name	Location/Value	Type
TDELAY	C:0x081F	

[illegible]

B] To Generate Triangular Wave

1]Initialize Port as Output
2] Send Data 00H to PORT
3]Increment Data by 1
4] Jump to Step 2

Conclusion:

We successfully interfaced the DAC0808 with the 8051 microcontroller and generated analog waveforms such as square, triangular, and sine waves. This experiment demonstrates the conversion of digital signals into corresponding analog output using DAC.

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
10	10	10	08	07	45
Sub Teacher Sign: [illegible]					

EXPERIMENT NO. 03 B

Aim:

A .To study working of stepper motor.

B. Write Embedded C Program to interface 8051 with Stepper motor to rotate motor in clockwise and Anticlockwise direction.

Objective:

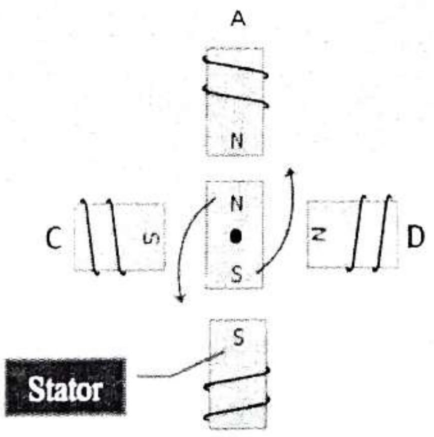
1 To be familiar with interfacing of stepper motor with 8051 microcontroller
2 To be familiar with the calculations of different step angles and speeds of stepper motor

Apparatus: PC keil simulator. 8051 controller kit. Flash magic software . Stepper motor and driver kit.

Theory:

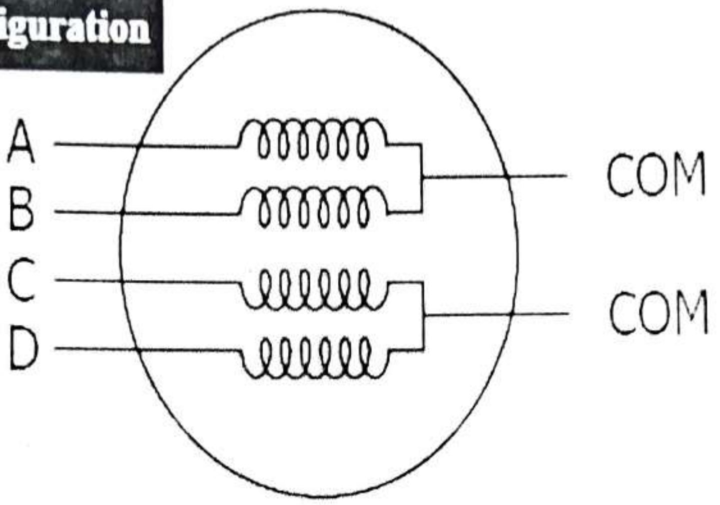
Stepper Motor

• A stepper motor is a widely used device that translates electrical pulses into mechanical movement
• The stepper motor is used for position control in applications such as disk drivers. dot matrix printers. and robotics, etc.
• Every stepper motor has a permanent magnet rotor (also called the shaft) surrounded by a stator



The most common stepper motors have four stator windings that are paired with a center-tapped common.
• This type of stepper motor is commonly referred to as a four-phase stepper motor
• The center tap allows a change of current direction in each of two coils when a winding is grounded, thereby resulting in a polarity change of the stator

Stator Windings Configuration



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 to each stator winding, the rotor will rotate.

Normal 4-Step Sequence

Clockwise ↓ / Counter-Clockwise ↑

Step #	Winding A	Winding B	Winding C	Winding D
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

The step angle is the minimum degree of rotation associated with a angles for various motors

• The step per revolution is the total number of steps needed to rotate one complete rotation or 360 degrees. To allow for finer resolutions, all stepper motors allow what is called an 8-step switching sequence.

• It's also called half-stepping, since each step is half of the normal step angle

Half-Step 8-Step Sequence

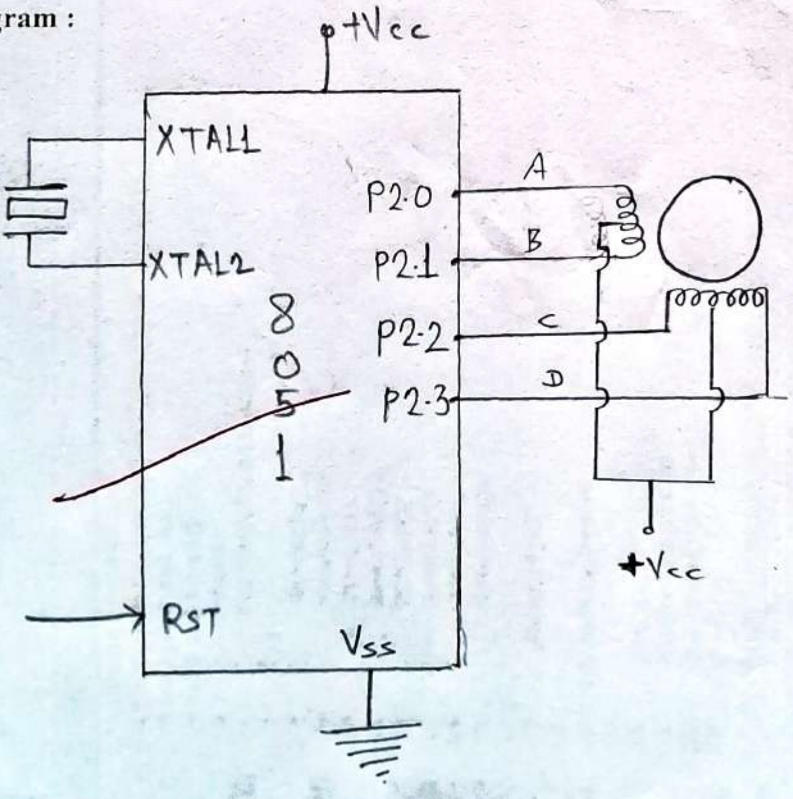
Clockwise ↓ / Counter-Clockwise ↑

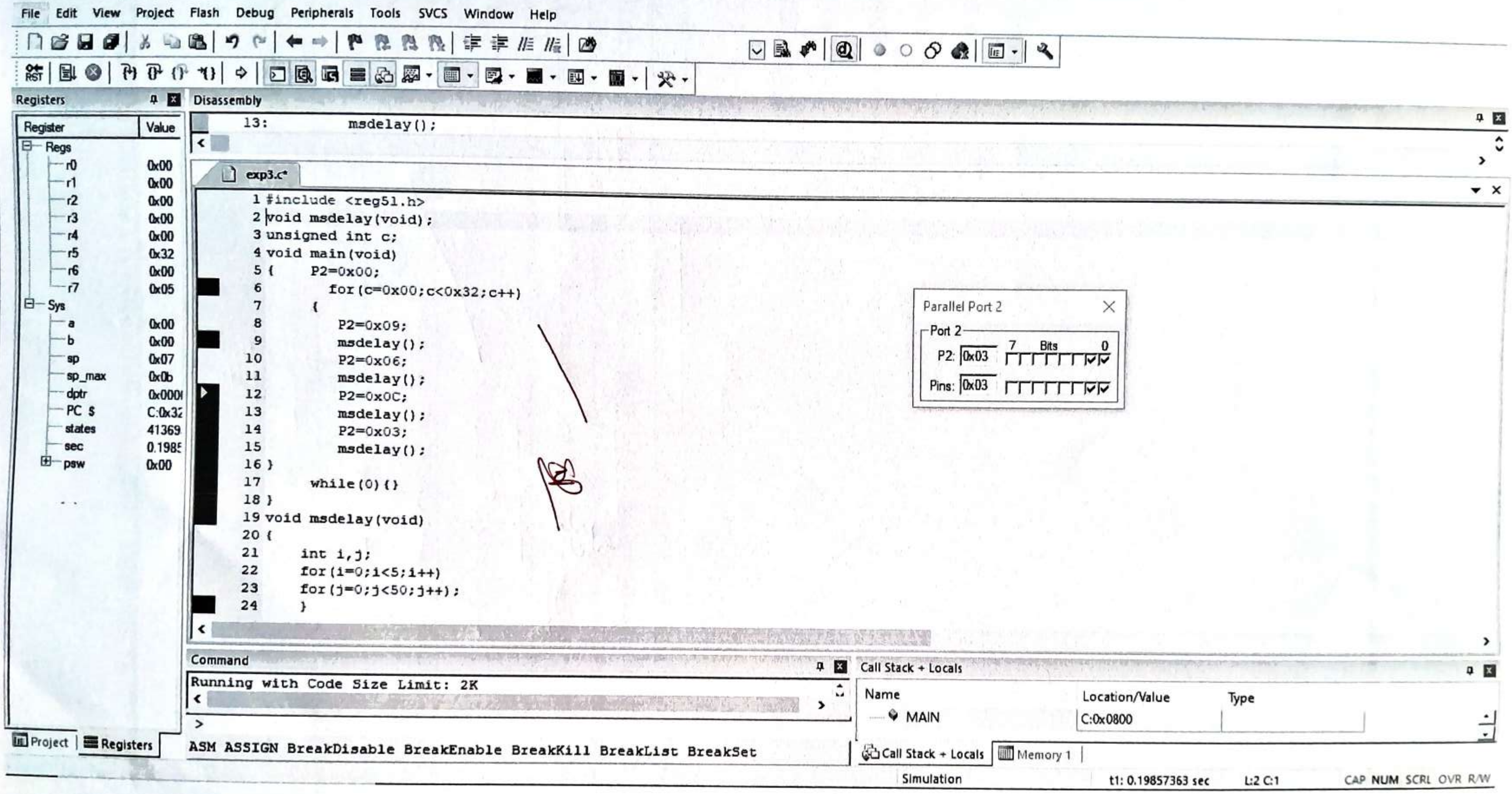
Step #	Winding A	Winding B	Winding C	Winding D
1	1	0	0	1
2	1	0	0	0
3	1	1	0	0
4	0	1	0	0
5	0	1	1	0
6	0	0	1	0
7	0	0	1	1
8	0	0	0	1

The relation between RPM (revolutions per minute), steps per revolution, and steps per second is as follow

- Steps/Sec = RPM x Steps per revolution / 60
- After completing every four steps, the rotor moves only one tooth pitch
- The smaller the step angle, the more teeth the motor passes
- ex. In a stepper motor with 200 steps per revolution, its rotor has 50 teeth since 4 x 50 = 200 steps are needed.

Interfacing Diagram :





```
Register        Value
Regs
  r0            0x00
  r1            0x00
  r2            0x00
  r3            0x00
  r4            0x00
  r5            0x32
  r6            0x00
  r7            0x05
Sys
  a             0x00
  b             0x00
  sp            0x07
  sp_max        0x0b
  dptr          0x000
  PC $          C:0x32
  states        41369
  sec           0.198
  psw           0x00
```

```
13:         msdelay();
```

```
#include <reg51.h>
void msdelay(void);
unsigned int c;
void main(void)
{     P2=0x00;
         for(c=0x00;c<0x32;c++)
        {
            P2=0x09;
            msdelay();
            P2=0x06;
            msdelay();
            P2=0x0C;
            msdelay();
            P2=0x03;
            msdelay();
}
      while(0){}
}
void msdelay(void)
{
      int i,j;
      for(i=0;i<5;i++)
      for(j=0;j<50;j++);
      }
```

```
P2:   0x03   7  Bits  0   [ ][ ][ ][ ][ ][ ][✓][✓]
Pins: 0x03                [ ][ ][ ][ ][ ][ ][✓][✓]
```

```
Running with Code Size Limit: 2K
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

Name	Location/Value	Type
MAIN	C:0x0800	

```
Simulation          t1: 0.19857363 sec     L:2 C:1     CAP NUM SCRL OVR R/W
```

[illegible]

Calculation Of Timer count Value :

Algorithm:

1]Start

2] Initialize PORT as Output

3]Send value of Sequence

4] Call Delay

5] jump to step 3

Conclusion:

We successfully interface the stepper motor with the 8051 microcontroller and controlled rotation in both clock-wise and anticlockwise directions. This experiment stepwise control of motor angle and speed using embedded C-program.

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
10	10	10	08	07	45
Sub Teacher Sign: [illegible]					

EXPERIMENT NO. 04

Aim: write a program for interfacing button, LED, relay & buzzer.

Objective: To write a embedded C program for interfacing button, LED, relay & buzzer as follows

A. when button 1 is pressed relay and buzzer is turned ON and LED's start chasing from left to right

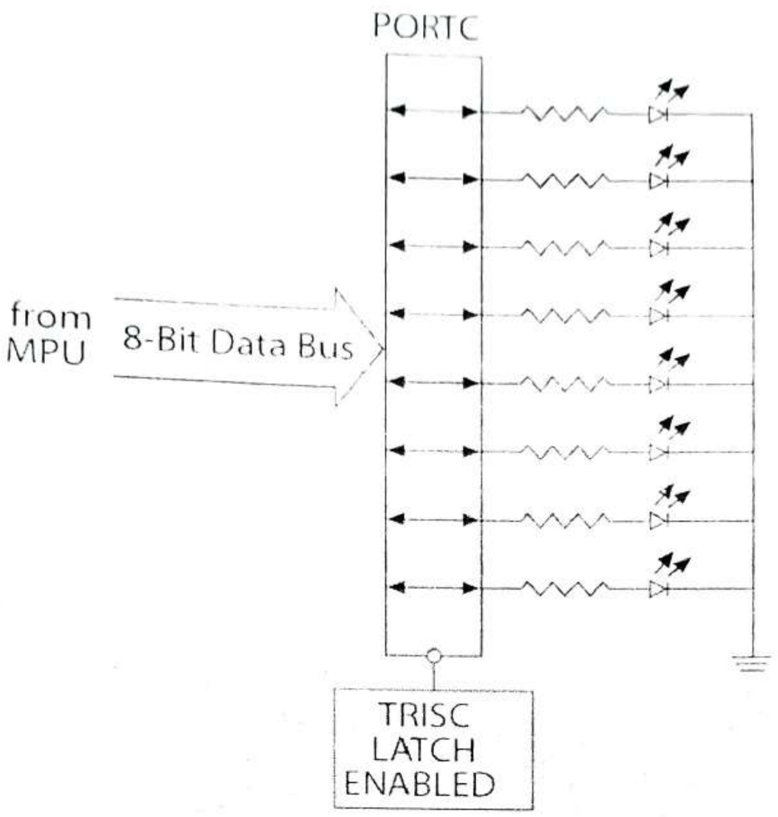
B. when button 2 is pressed relay and buzzer is turned OFF and Led start chasing from right to left

Apparatus: MPLAB Simulator, C18, PC, PIC18f Development KIT, LED KIT, USB cable, mikrobootloader.

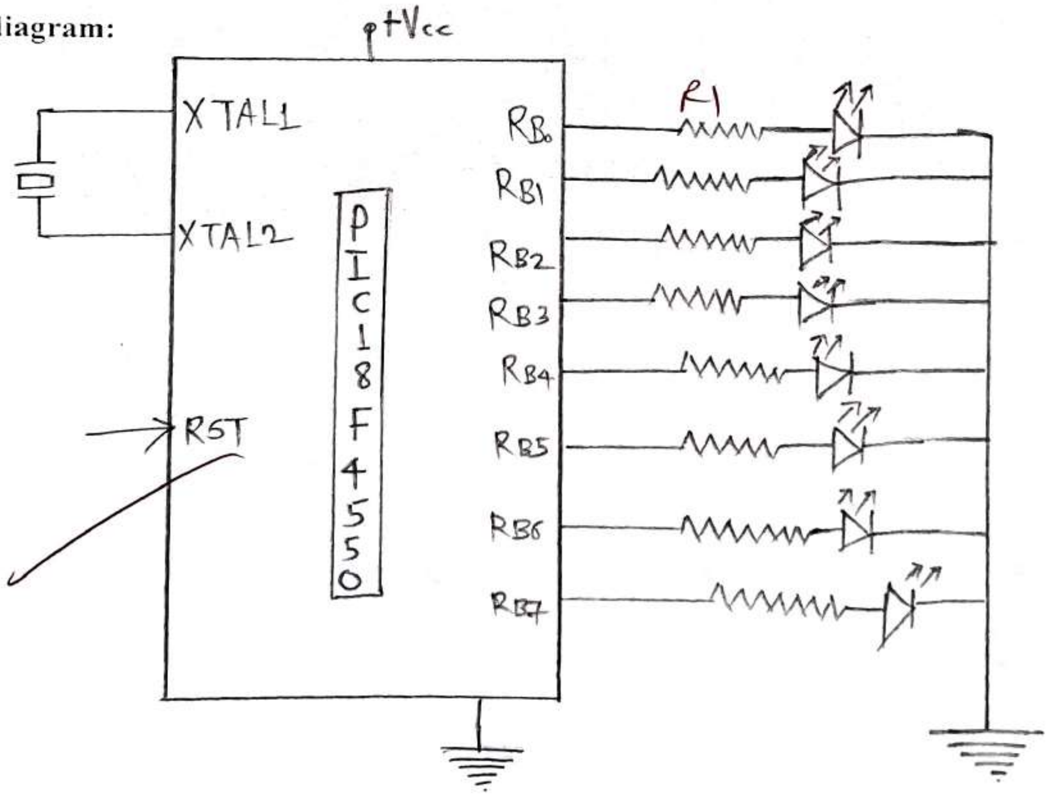
Features of PIC18F4550:

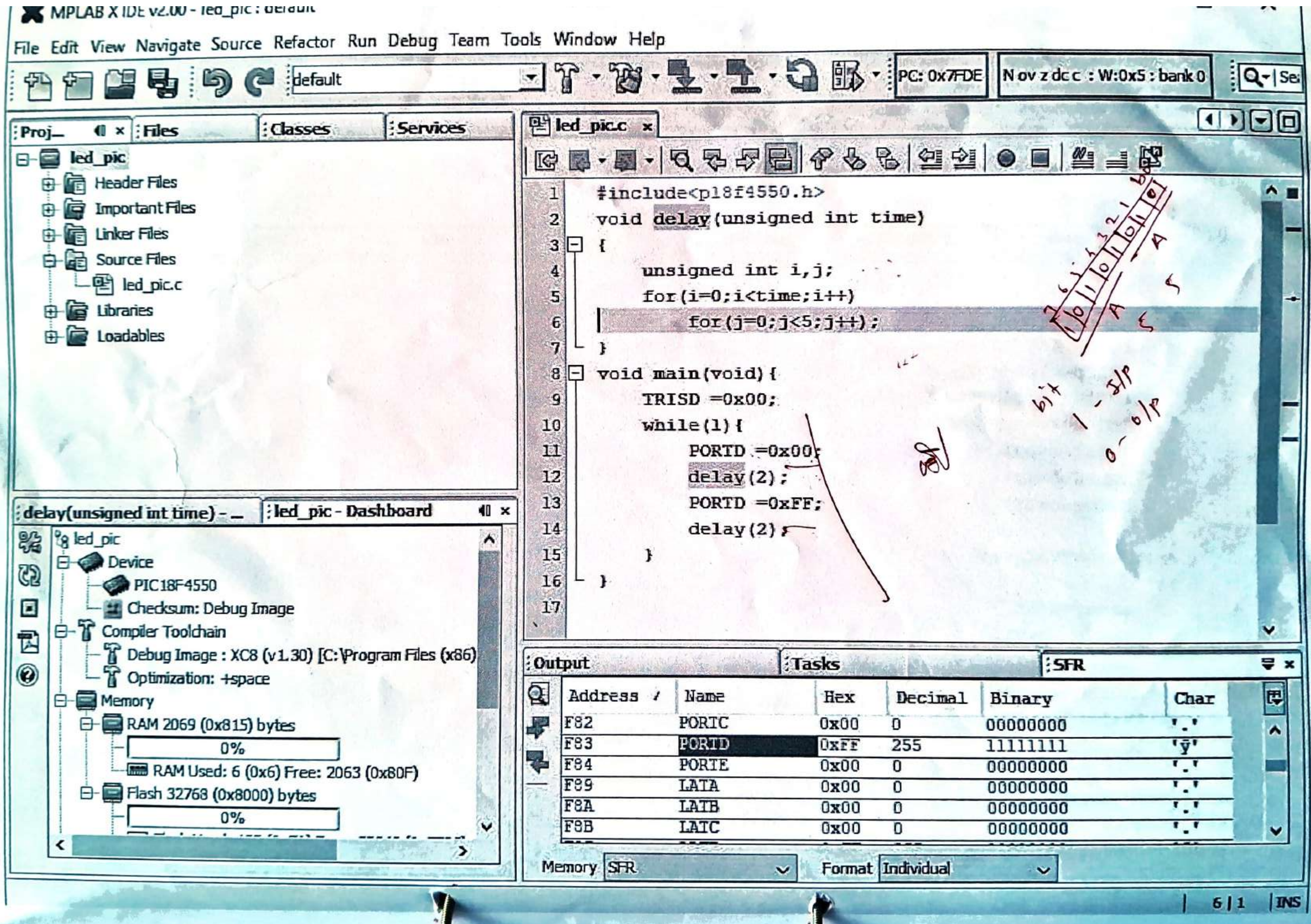
- PIC18F4550 belongs to the PIC18F family; PIC18F4550 is an 8bit microcontroller and uses RISC architecture. PIC18F4550 has 40 pins in PDIP (dual in line package) and 44 pin in TQFP (Quad flat package).
- 32KB flash memory, 2048 bytes of SRAM (synchronous Random Access memory), EEPROM (Electrically Erasable Program Read Only Memory) of 256 bytes are embedded in the PIC18F4550.
- It has 35 I/O pins for interfacing and communication with other peripherals, 13channel of 10bit analog to digital converters which are used for interfacing and communicating the analog peripherals (DC motor, LDR, etc.).
- It has 2 CCP and 1 ECCP module that is enhanced capture and compare module which is mainly used for modulation and waveform generation functions. CCP module is of 16bit register works as 16 capture bit register, 16 compare bit register, and PWM and duty cycle register.
- PIC18F4550 has SPI (serial peripheral interface) and i2c (inter integrated circuit) for master and slave modes. It has SPP (Streaming Parallel Port) for USB streaming transfer.
- PIC18F4550 is embedded with 4 timer modules (timer0 to timer3), 2 comparator modules and 3 external interrupt. It has Dual Oscillator options allow microcontroller and USB module to run at different clock speeds. It can operate in 2.0V to 5.5V

Peripheral Interface Controller (PIC) is microcontroller developed by Microchip. PIC microcontroller is fast and easy to implement program when we compare other microcontrollers like 8051. The ease



Interfacing diagram:




MPLAB X IDE v2.00 - led_pic : default
File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
default
PC: 0x7FDE
N ov z dc c : W:0x5 : bank 0
Proj_ Files Classes Services
led_pic
Header Files
Important Files
Linker Files
Source Files
led_pic.c
Libraries
Loadables
led_pic.c
Output Tasks SFR
Memory SFR
Format Individual
6 | 1 INS

```
#include<p18f4550.h>
void delay(unsigned int time)
{
    unsigned int i,j;
    for(i=0;i<time;i++)
        for(j=0;j<5;j++);
}
void main(void){
    TRISD =0x00;
    while(1){
        PORTD =0x00;
        delay(2);
        PORTD =0xFF;
        delay(2);
    }
}
```

delay(unsigned int time) - led_pic - Dashboard

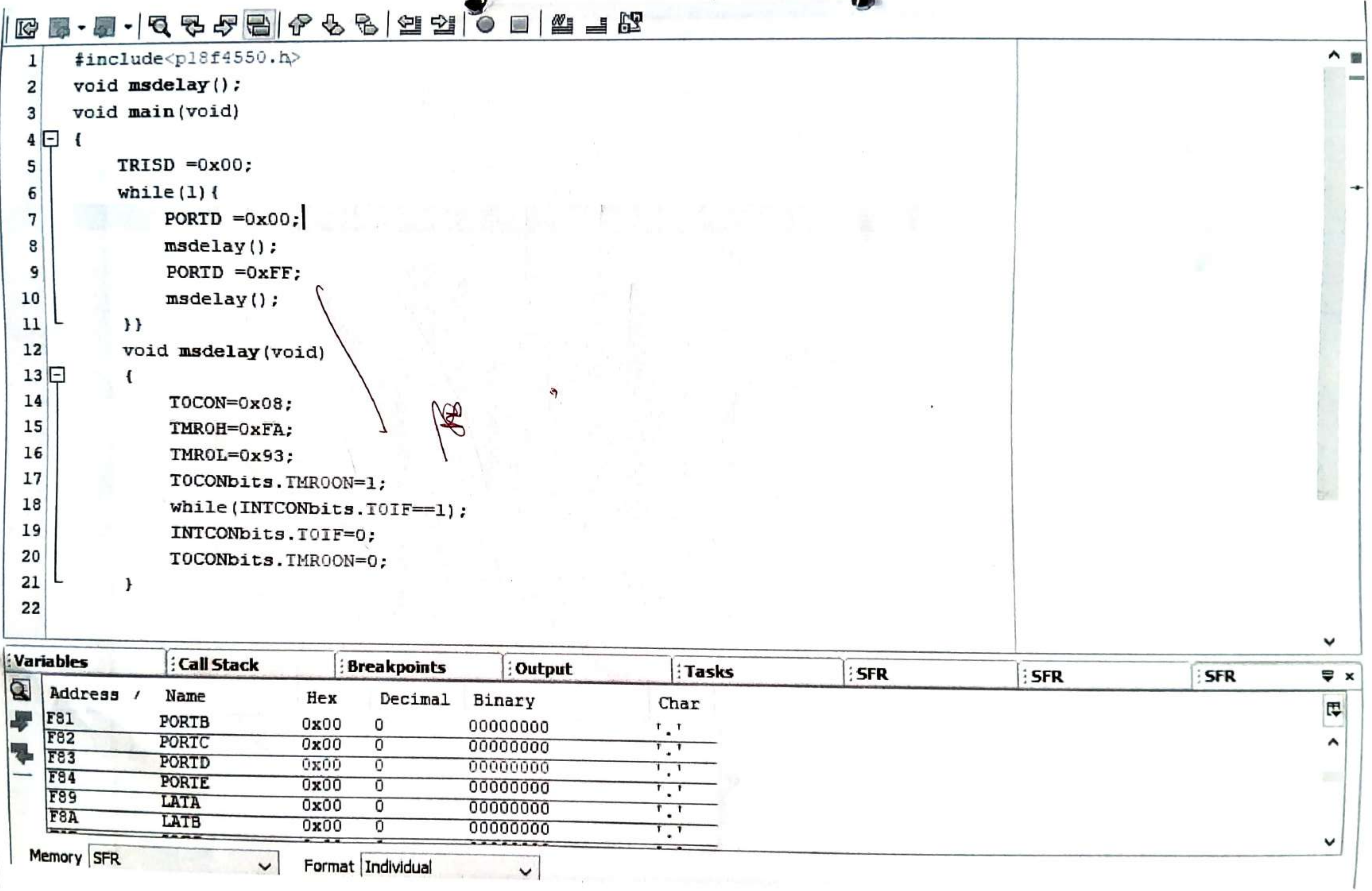
```
led_pic
  Device
    PIC18F4550
    Checksum: Debug Image
  Compiler Toolchain
    Debug Image : XC8 (v1.30) [C:\Program Files (x86)
    Optimization: +space
  Memory
    RAM 2069 (0x815) bytes
      0%
      RAM Used: 6 (0x6) Free: 2063 (0x80F)
    Flash 32768 (0x8000) bytes
      0%
```

Address	Name	Hex	Decimal	Binary	Char
F82	PORTC	0x00	0	00000000	'.'
F83	PORTD	0xFF	255	11111111	'ÿ'
F84	PORTE	0x00	0	00000000	'.'
F89	LATA	0x00	0	00000000	'.'
F8A	LATB	0x00	0	00000000	'.'
F8B	LATC	0x00	0	00000000	'.'

7	6	5	4	3	2	1	0
1	0	1	0	1	0	1	0

A 5 A 5

bit
1 - I/P
0 - O/P



```c
#include<p18f4550.h>
void msdelay();
void main(void)
{
     TRISD =0x00;
     while(1){
          PORTD =0x00;
          msdelay();
          PORTD =0xFF;
          msdelay();
     }}
     void msdelay(void)
     {
          T0CON=0x08;
          TMR0H=0xFA;
          TMR0L=0x93;
          T0CONbits.TMR0ON=1;
          while(INTCONbits.T0IF==1);
          INTCONbits.T0IF=0;
          T0CONbits.TMR0ON=0;
     }
```

[illegible]

```
Address /  Name    Hex    Decimal  Binary     Char
F81        PORTB   0x00   0        00000000   '.'
F82        PORTC   0x00   0        00000000   '.'
F83        PORTD   0x00   0        00000000   '.'
F84        PORTE   0x00   0        00000000   '.'
F89        LATA    0x00   0        00000000   '.'
F8A        LATB    0x00   0        00000000   '.'
```

Memory SFR   Format Individual

Algorithm:

1. Start
2. Initialize port B as input and port D, port A as output (RA2= relay, RB5= buzzer, RB0= row0, RB3= row 3, RB4= column 0, RB8=column 3)
3. Initialize integer as code for checking key status
4. Take data from port B into code variable
5. Check key 1 is pressed or not
6. If key 1 is pressed make relay on, buzzer on and go for LED part 1
7. Check key 2 is pressed or not
8. If key 2 is pressed make buzzer and relay off and go for led part 2
9. End

Algorithm for LED part 1

1. Start
2. Initialize integer i
3. Declare for loop with initial values i=0x1, final value 0x80, shift towards left with each iteration
4. Send 0x0 to port D. Send value of i to port D
5. Call delay
6. Stop

Algorithm for LED part 2

1. Start
2. Initialize integer i
3. Declare for loop with initial values i=0x80, final value 0x1, shift towards right with each iteration
4. Send 0x0 to port D. Send value of i to port D
5. Call delay
6. Stop

Conclusion: In this experiment we studied interfacing of various input device such as buttons and output devices such as LED, relay and buzzer with PIC18f4550 microcontroller and also observe when input button 1 is pressed relay and buzzer is turned on and LED changing from left to right. When input button 2 is pressed relay and buzzer turn off LED changing from right to left

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
10	10	10	08	07	45
Sub Teacher Sign: [illegible]					

EXPERIMENT NO. 05

Aim: write a program for interfacing LCD to PIC18FXX

Objective: To write a embedded C program for interfacing LCD with PIC18FXX
A. Display Message on 1st Line 4Th Position

Apparatus: MPLAB Simulator, , PC, PIC18fxx Development KIT, LCD KIT, USB cable, mikrobootloader.

Theory:

LCD is finding widespread use replacing LEDs

1. The declining prices of LCD
2. The ability to display numbers, characters, and graphics
3. Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD
4. Ease of programming for characters and graphics

Pin Descriptions for LCD

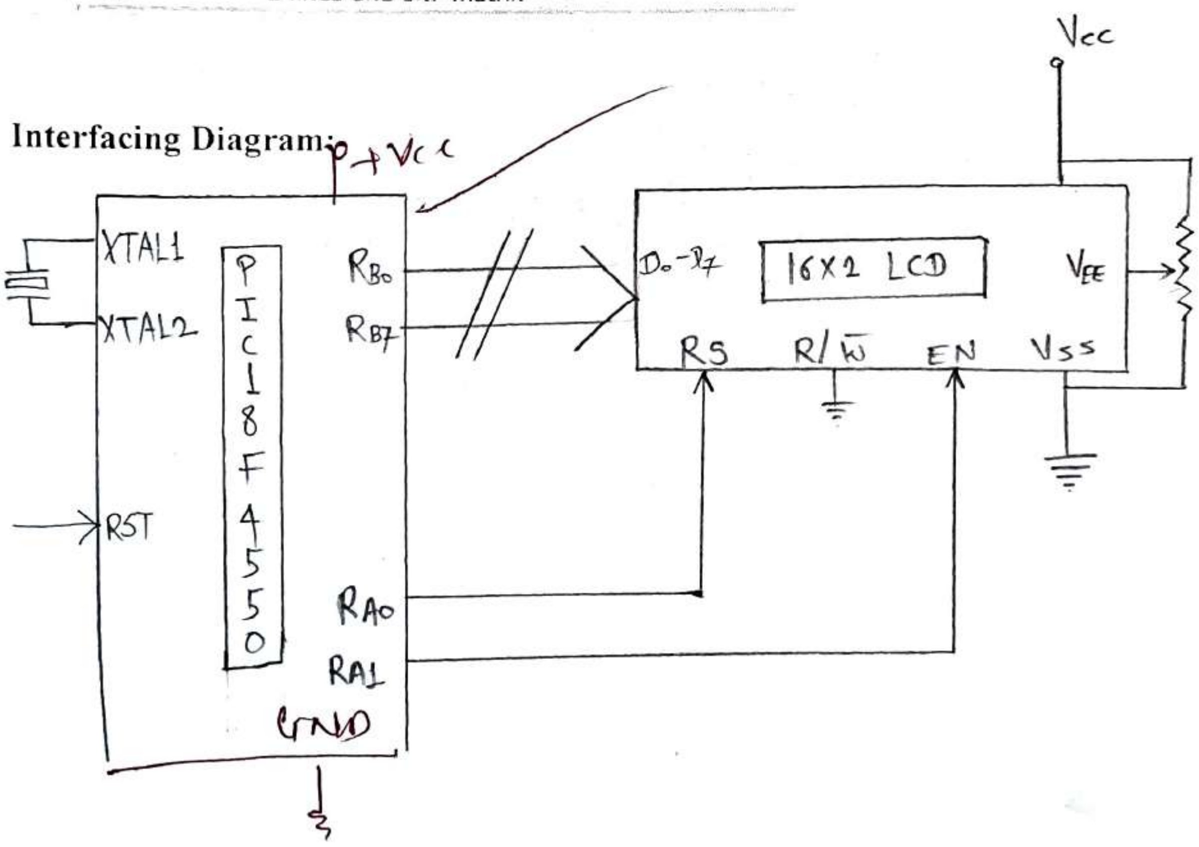
Pin	Symbol	I/O	Descriptions
1	VSS	--	Ground
2	VCC	--	+5V power supply
3	VEE	--	Power supply to control contrast
4	RS	I	RS=0 to select command register, RS=1 to select data register
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

used by the LCD to latch information presented to its data bus

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 blinking
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
80	Force cursor to beginning to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5x7 matrix

Interfacing Diagram:



Experiment No. 5

Name : Dnyanesh Agale

Class : TE E& TC

Roll No. :

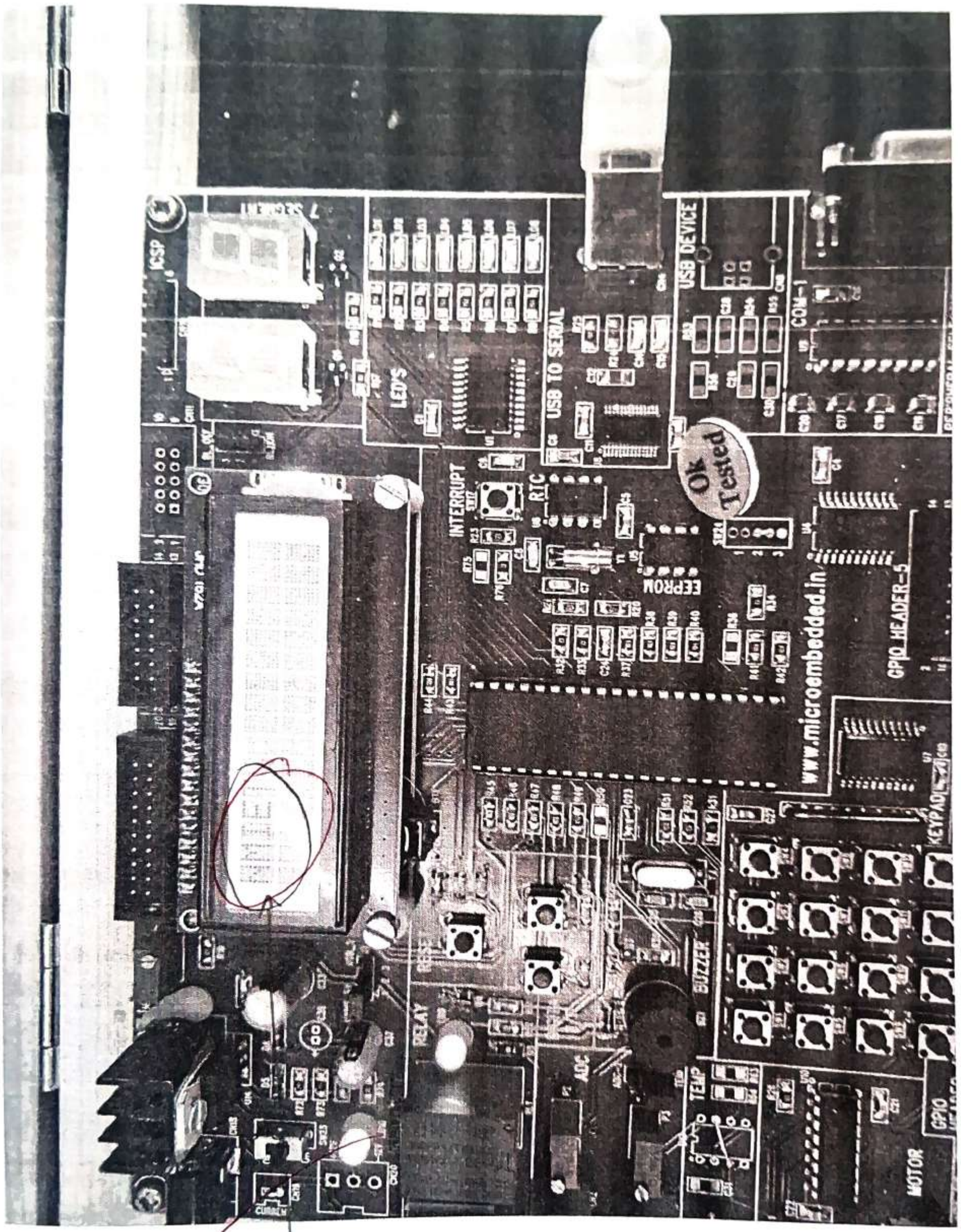
```
#include<p18f4550.h>
#define EN LATAbits.LA1
#define RS LATAbits.LA0
#define Dataport LATB

void msdelay(void);
void LCD_CMD(unsigned char value);
void LCD_DATA(unsigned char value);
void main(void)
{

  TRISB =0x00;
 TRISAbits.RA0 = 0; //RS pin
  TRISAbits.RA1 = 0; // EN pin
  LCD_CMD(0x38);
  msdelay();

  LCD_CMD(0x06);
  msdelay();
  LCD_CMD(0x0C);
  msdelay();
  LCD_CMD(0x01);
  msdelay();
```

[illegible]


7 SEGMENT
LED'S
USB TO SERIAL
USB DEVICE
COM-1
INTERRUPT
RTC
EEPROM
Ok Tested
www.microembedded.in
GPIO HEADER-5
KEYPAD
RELAY
BUZZER
ADC
TEMP
MOTOR

Output

Algorithm:

A]Main Program

```
1] Start
2] Initialize Data PORT  and  Control PORT  as output
3] Initialize LCD in 8 bit mode.
4]Send command 01H.
5]Call Delay
6]Send command 06H.
7]Call Delay
8]Send command 0EH
9]Call Delay
10]Send Data on Port1
11]Call Delay
12]STOP
```

B]Subroutine 1:

```
1]Transfer command to Data PORT
2]EN= H to L
3] RS=0
4]R/W=0
5]Return
```

B]Subroutine 2:

```
1]Transfer Data to Data PORT
2]EN= H to L
3] RS=1
4]R/W=0
5]Return
```

B]Subroutine 3:

```
1] Generate Delay
2]Return
```

Conclusion:
In this experiment we demonstrated the interfacing of LCD to PIC18FXX, wrote an embedded C program for interfacing and displayed message on 1st line 4th position

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
10	10	10	08	08	46
Sub Teacher Sign: [illegible]					