Visvesvaraya National Institute
of
Technology, Nagpur

Embedded Systems (ECP403)

Lab Report

Submitted By :
Suraj Kaple (BT18ECE021)
Semester 5
Electronics and Communication Engineering Dept.

Submitted To :
Dr. Ankit A Bhurane
Course Instructor

Contents

Experiment-1:

1.1 Indefinitely to and fro running LEDs:

Aim: Glow the LEDs indefinitely to and fro.

Problem Statement: Write an assembly language program and a C language program for 8051 micro-controller so that the 8 LEDs should glow one by one from right to left and then from left to right or vice versa.

Code: Assembly language program for 8051 micro-controller:

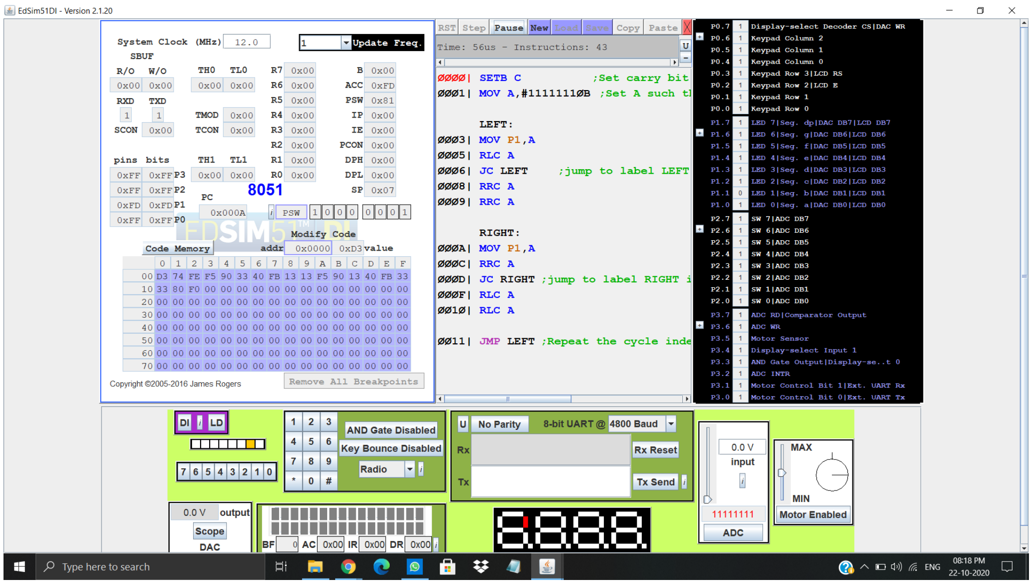
```
SETB C                      ;Set carry bit to 1
MOV A,#11111110B              ;Set A such that rightmost LED glows first

LEFT:
MOV P1,A
RLC A
JC LEFT         ;jump to LEFT if left-most LED is not reached

RRC A
RRC A

RIGHT:
MOV P1,A
RRC A
JC RIGHT         ;jump to RIGHT if right-most LED is not reached

RLC A
RLC A

JMP LEFT         ;Repeat the cycle indefinitely
```

C program for 8051 micro-controller:

```
#include<reg51.h>
void main(){
        B = 0xFE;
        P1 = B;
        while(1){
                unsigned int X = 2, i;
                B = 0xFD;
                for(i=0; i<7; i++){
                        P1 = B;
                        B = B + X;
                        if(X < 128){
                                X = 2*X;
```

```
                    B = B - X;
                }
            }
            B = 0xBF;
            X = 64;
            for(i=0; i<7; i++){
                P1 = B;
                B = B + X;
                X = X/2;
                B = B - X;
            }
        }
}
```

Output: The LEDs were successfully interfaced with 8051 micro-controller.EdSim51 and Keil μvision were used as simulators. The LEDs were then used to display a to and fro running pattern.



```
Time: 56us - Instructions: 43

0000| SETB C             ;Set carry bit
0001| MOV A,#11111110B  ;Set A such th
      LEFT:
0003| MOV P1,A
0005| RLC A
0006| JC LEFT      ;jump to label LEFT
0008| RRC A
0009| RRC A
      RIGHT:
000A| MOV P1,A
000C| RRC A
000D| JC RIGHT ;jump to label RIGHT i
000F| RLC A
0010| RLC A

0011| JMP LEFT ;Repeat the cycle inde
```

Figure 1: An instance during the indefinite running of LEDs.

Discussion and Observation: The LEDs are connected to P1 in common anode fashion. To glow any particular LED, the corresponding pin in P1 should be operated in active low mode.

Conclusion: It can be concluded that the LEDs can be interfaced with 8051 micro-controller and can be used to display patterns, output pin information for some other programs, etc.

1.2 Using switch to glow LED:

Aim: Using switches to glow specific LEDs.

Problem Statement: Write an assembly language program and C language program for 8051 micro-controller so that the LED corresponding to the switch pressed should glow until the switch is pressed again.

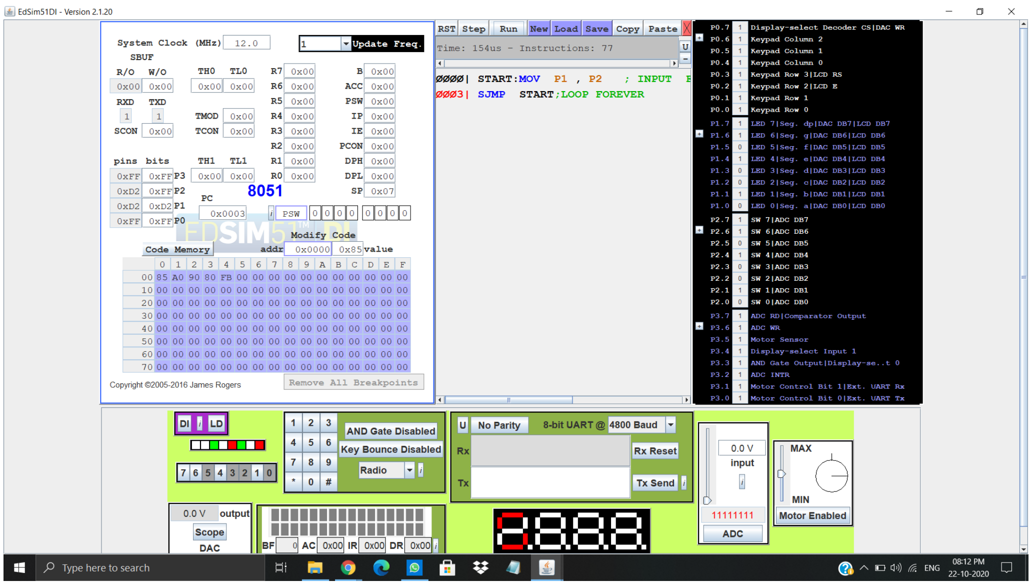
Code: The assembly language program for 8051 micro-controller:

```
START:MOV P1,P2      ;INPUT FROM SWITCHES GOES TO THE LEDS
JMP START                ;LOOP FOREVER
```

C program for 8051 micro-controller:

```
#include<reg51.h>
void main(){
        while(1){
                P1 = P2;
}
```

Output: The switch bank and LEDs were successfully interfaced with 8051 micro-controller.EdSim51 and Keil μvision was used as simulators. The LEDs then glow according to the switches pressed.



```
Time: 154us - Instructions: 77

ØØØØ| START:MOV  P1 , P2   ; INPUT F
ØØØ3| SJMP  START;LOOP FOREVER
```

Figure 2: The LEDs corresponding to the pressed switches glow.

Discussion and Observation: The LEDs are connected to P1 in common anode fashion. To glow any particular LED, the corresponding pin in P1 should be operated in active low mode. This operation is done by using the switches connected to P2.

Conclusion: It can be concluded that the LEDs can be glowed manually as well by using the switches as input.

Experiment-2: Interfacing DAC- Generate sine wave using DAC

Aim: Generate sine wave using Digital-to-Analog Converter (DAC).

Problem Statement: Write an assembly language program and C language program for 8051 micro-controller to indefinitely display an approximate sine signal with peak-to-peak amplitude as 5V on the scope of the simulator using a lookup table of minimum 40 samples saved in ROM.

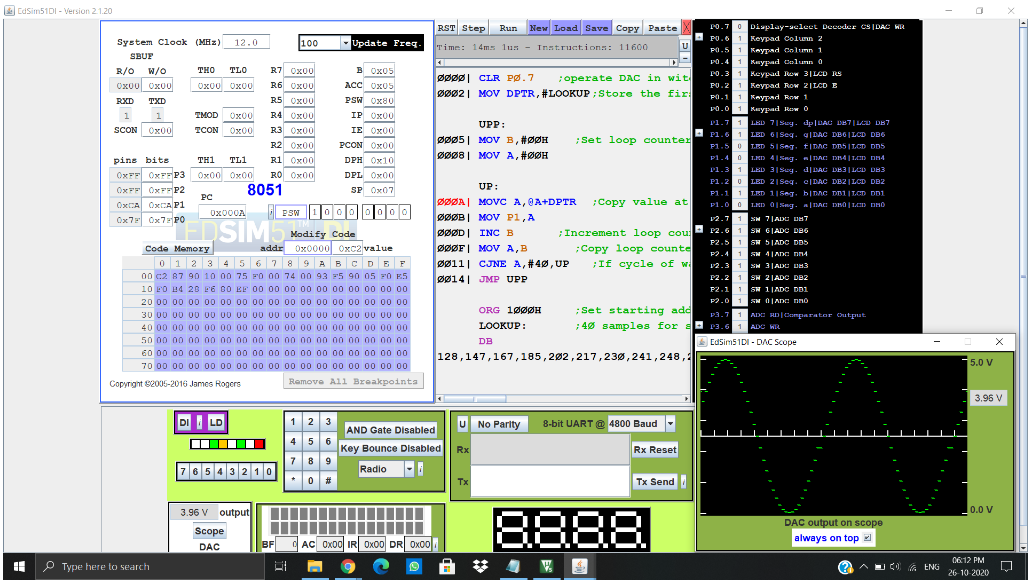
Code: The assembly language program for 8051 micro-controller:

```
CLR P0.7                ;operate DAC in wite mode (active low pin)
;Store the first address of lookup table in data pointer
MOV DPTR,#LOOKUP

UPP:
MOV B,#00H                ;Set loop counter to 0
MOV A,#00H

UP:
;Copy value at (A+1)th position in lookup table to accumulator
MOVC A,@A+DPTR
MOV P1,A
INC B                   ;Increment loop counter by 1
MOV A,B                   ;Copy loop counter to accumulator
CJNE A,#40,UP           ;If cycle of wave is not complete, jump to UP
JMP UPP

ORG 1000H               ;Set starting address of lookup table to 1000H
LOOKUP:                   ;40 samples for sine wave
DB 128,147,167,185,202,217,230,241,248,253,255,253,248,241,230,217,
202,185,167,147,128,108,88,70,53,38,25,14,7,2,1,2,7,14,25,38,53,70,
88,108
```

C program for 8051 micro-controller:

```
#include<reg51.h>
unsigned char lookup[] = {128,147,167,185,202,217,230,241,248,253,255,
                          253,248,241,230,217,202,185,167,147,128,108,
                          88,70,53,38,25,14,7,2,1,2,7,14,25,38,53,70,
                          88,108};
unsigned int i;
void main(){
        P0 = 0x00;
        while(1){
                for(i=0; i<40; i++){
                        P1 = lookup[i];
                }
        }
}
```

Output: The sine wave was successfully plotted with the help of 40 samples saved in look up table.



```
Time: 14ms 1us - Instructions: 11600

ØØØØ| CLR PØ.7      ;operate DAC in wit
ØØØ2| MOV DPTR,#LOOKUP;Store the fir

      UPP:
ØØØ5| MOV B,#ØØH      ;Set loop counter
ØØØ8| MOV A,#ØØH

      UP:
ØØØA| MOVC A,@A+DPTR   ;Copy value at
ØØØB| MOV P1,A
ØØØD| INC B          ;Increment loop cou
ØØØF| MOV A,B         ;Copy loop counte
ØØ11| CJNE A,#4Ø,UP    ;If cycle of wa
ØØ14| JMP UPP

      ORG 1ØØØH       ;Set starting add
      LOOKUP:         ;4Ø samples for s
      DB
128,147,167,185,2Ø2,217,23Ø,241,248,2
```

Figure 3: Snap of Simulator.

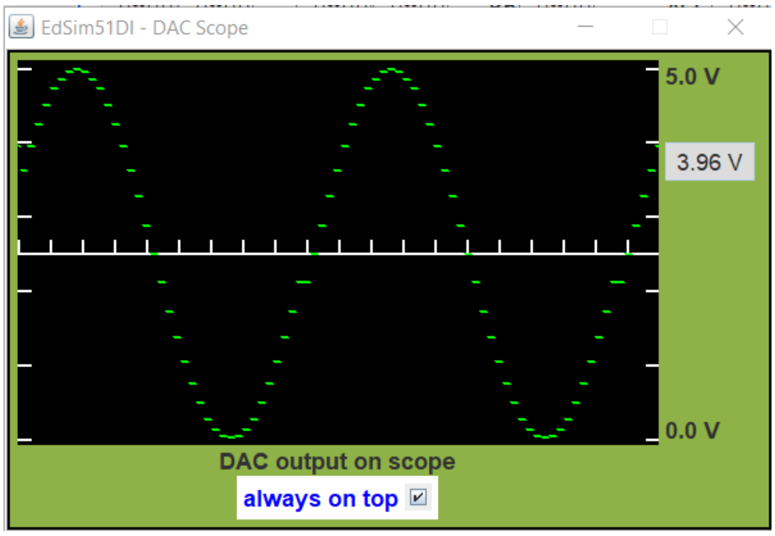


Figure 4: Sinusoidal wave as displayed on the scope.

Discussion and Observation: The most important part here is to construct the look-up table. Once this is done, we only need to display those values one after

other on the scope using the Digital to Analog Converter (DAC).
The 40 sample values can be determined from the formula

$$i^{th} sample = 128 + 127 \sin(\frac{2\pi}{40} i).$$

where i varies from 0 to 39 (*i.e.,* 40 samples).
The smoothness of the sine wave can be increased by increasing the number of samples.

Conclusion: It can be concluded that a micro-controller can be used to generate a sinusoidal wave. One of the ways is to store required values in the look-up table and then using it for displaying the wave on the scope.

Experiment-3: Interfacing keypad- Detecting key pressed on keypad and indicating unique binary pattern

Aim: Interfacing of 4x3 Keypad: Detect the key pressed and indicate as unique binary pattern on LED array with default connections.

Problem Statement: Write an assembly language program and C language program for 8051 micro-controller to detect a key press on the keypad and uniquely identify the key by representing its unique binary representation on the LED bar.

Code: The assembly language programs for 8051 micro-controller:

```
MOV P1,#0FFH         ;MAKE ALL LEDS OFF

START:
MOV P0,#0FFH
MOV B,#03H             ;NUMBER_OF_COLUMNS (3 IN OUR KEYPAD)
MOV A,#00H             ;ROW_NUMBER (INITIALISED TO 0)
MOV R0,#00H          ;COLUMN_NUMBER (INITIALISED TO 0)

COLCHECK:
CLR P0.6
MOV R0,#01H         ;SET COLUMN_NUMBER TO 1
LCALL ROWCHECK
LCALL DISPLAY
SETB P0.6

CLR P0.5
MOV R0,#02H         ;SET COLUMN_NUMBER TO 2
LCALL ROWCHECK
LCALL DISPLAY
SETB P0.5

CLR P0.4
MOV R0,#03H         ;SET COLUMN_NUMBER TO 3
LCALL ROWCHECK
LCALL DISPLAY
SETB P0.4

JMP COLCHECK         ;FOREVER LOOP

ROWCHECK:
JB P0.3,ROW2         ;IF NOT IN ROW1, CHECK IN ROW2
MOV A,#01H           ;IF IN ROW1, SET ROW_NUMBER TO 1
RET
```

```
ROW2:
JB P0.2,ROW3          ;IF NOT IN ROW2, CHECK IN ROW3
MOV A,#02H            ;IF IN ROW2, SET ROW_NUMBER TO 2
RET

ROW3:
JB P0.1,ROW4          ;IF NOT IN ROW3, CHECK IN ROW4
MOV A,#03H            ;IF IN ROW3, SET ROW_NUMBER TO 3
RET

ROW4:
JB P0.0,RETURN1          ;IF NOT IN ROW4, RETURN TO COL_CHECK
MOV A,#04H                ;IF IN ROW4, SET ROW_NUMBER TO 4
RET

RETURN1: RET

DISPLAY:
;KEY = (A-1)*B + R0
;RETURN TO COLCHECK IF KEY IS NOT FOUND IN THE CURRENT COLUMN
JZ RETURN2
DEC A       ;A = A - 1
MUL AB      ;A = A * B
ADD A,R0    ;A = A + R0
CPL A
MOV P1,A
JMP START   ;START OVER AGAIN

RETURN2: RET
```

C program for 8051 micro-controller:

```
#include<reg51.h>
sbit c1 = P0^6;
sbit c2 = P0^5;
sbit c3 = P0^4;
unsigned char b, a, r, key;
void init(){
        P0 = 0xFF;
        b = 0x03;
        a = 0x00;
        r = 0x00;
}
void display(){
        if(a == 0)
                return;
        key = (a - 1)*b + r;
        P1 = key;
        init();
}
void rowcheck(){
        if(!P0^3)
                a = 0x01;
        else if(!P0^2)
                a = 0x02;
        else if(!P0^1)
                a = 0x03;
        else if(!P0^0)
                a = 0x04;
}
void main()
{
        P1 = 0xFF;
        init();
        while(1){
                c1 = 0;
                r = 0x01;
                rowcheck();
                display();
                c1 = 1;

                c2 = 0;
                r = 0x02;
                rowcheck();
                display();
                c2 = 1;

                c3 = 0;
                r = 0x03;
                rowcheck();
                display();
                c3 = 1;
        }
}
```

Output: The keypad was successfully interfaced with the 8051 micro-controller using EdSim51 as simulator. The key-press was successfully detected and the binary equivalent of the unique key value was then displayed on the LED array.

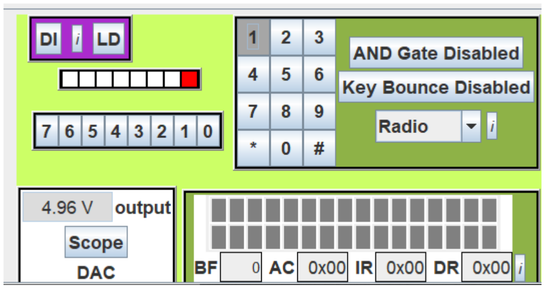


Figure 5: Output 1

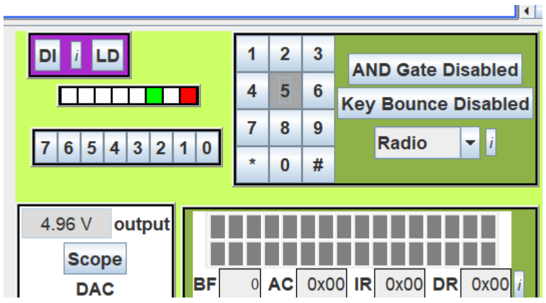


Figure 6: Output 2

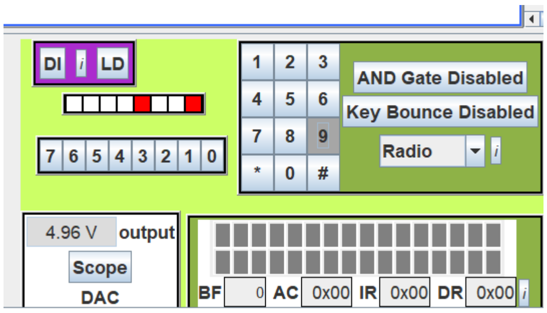


Figure 7: Output 3

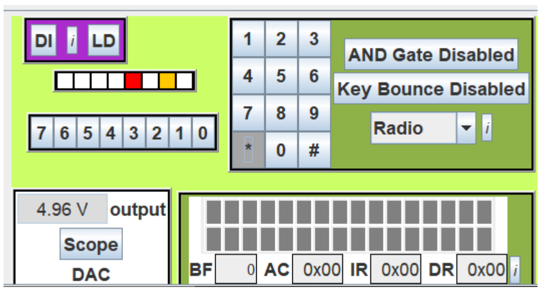
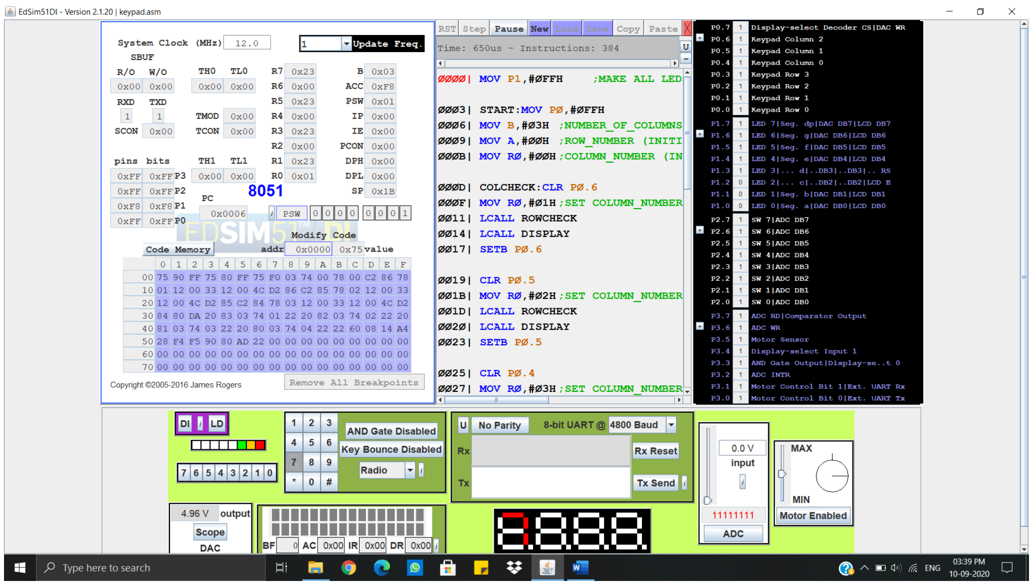


Figure 8: Output 4



```
Time: 650us - Instructions: 384

ØØØØ| MOV P1,#ØFFH      ;MAKE ALL LED

ØØØ3| START:MOV PØ,#ØFFH
ØØØ6| MOV B,#Ø3H  ;NUMBER_OF_COLUMNS
ØØØ9| MOV A,#ØØH  ;ROW_NUMBER (INITI
ØØØB| MOV RØ,#ØØH ;COLUMN_NUMBER (IN

ØØØD| COLCHECK:CLR PØ.6
ØØØF| MOV RØ,#Ø1H ;SET COLUMN_NUMBER
ØØ11| LCALL ROWCHECK
ØØ14| LCALL DISPLAY
ØØ17| SETB PØ.6

ØØ19| CLR PØ.5
ØØ1B| MOV RØ,#Ø2H ;SET COLUMN_NUMBER
ØØ1D| LCALL ROWCHECK
ØØ2Ø| LCALL DISPLAY
ØØ23| SETB PØ.5

ØØ25| CLR PØ.4
ØØ27| MOV RØ,#Ø3H ;SET COLUMN_NUMBER
```

Figure 9: Snap of simulator

Discussion and Observation: The keypad is used in radio mode, i.e., only one key is pressed at a time. The keypad used in EdSim51 simulator is as shown below-

1	2	3
4	5	6
7	8	9
*	0	#

The unique value for corresponding keys has been assigned as follows-

01	02	03
04	05	06
07	08	09
10	11	12

Indexing of rows and columns starts from 1. So, there are rows with indices 1, 2, 3, 4 and columns with indices 1, 2, 3.
Registers and what do they store:
Accumulator(A) - Row index (from 1 to 4), also A = 0 indicates that key was not found in the current column.
Register B - Total number of columns in the keypad (here, 3)
R_0 - Column index

Unique key value say x, for any key can be calculated by using the formula

$$x = ((A - 1) * B) + R_0 \tag{1}$$

Example:
Let the key pressed be 6
So, A = 2, R_0 = 3, B = 3
Hence, unique key value is

$$x = ((2 - 1) * 3) + 3 = 3 + 3 = 6$$

Then binary equivalent of this unique key value is then displayed on the LED array.

All pins of P0 are made HIGH. The column to be checked is made LOW and then the state of the rows is checked. Any row with a LOW state is noted in accumulator. So, now we know both the column and the row which have matching (LOW) state, hence the particular key is known.
The program continuously checks whether a key is pressed or not starting with all the keys in column#1, then in column#2 and finally in column#3. When A $\neq$ 0, then it indicates that a key has been pressed and the key is identified using (1). When A = 0, then it indicates that no key-press has been detected in the current column.

Conclusion: It can be concluded that a keypad can be interfaced with 8051 micro-controller. This keypad can be used to take input from the user in the form of numeric values and characters like '*' and '#'.

Experiment-4: Interfacing LCD- Displaying name on LCD

Aim: To interface LCD and displaying student name on LCD using 8051 micro-controller

Problem Statement: Write an assembly language program for 8051 micro-controller to display the name of student on the LCD peripheral.

Code: The assembly language program for 8051 micro-controller:

```
CLR P0.3          ;set Register Select (RS) = 0
MOV P1,#38H          ;(1)
LCALL WAIT          ;wait 1us
MOV P1,#0EH          ;(2)
LCALL WAIT          ;wait 1us
MOV P1,#06H          ;(3)
LCALL WAIT          ;wait 1us

SETB P0.3          ;set Register Select (RS) = 1
MOV P1,#'S'          ;write 'S'
LCALL WAIT          ;wait 1us
MOV P1,#'U'          ;write 'U'
LCALL WAIT          ;wait 1us
MOV P1,#'R'          ;write 'R'
LCALL WAIT          ;wait 1us
MOV P1,#'A'          ;write 'A'
LCALL WAIT          ;wait 1us
MOV P1,#'J'          ;write 'J'
LCALL WAIT          ;wait 1us
JMP ENDING

WAIT:
SETB P0.2          ;set enable (E) high
LCALL DELAY          ;generate delay
CLR P0.2          ;set enable (E) low
RET

DELAY:                ;a delay function to create 1us delay
MOV R0,#030H
DJNZ R0,$
RET

ENDING:
SJMP $
```

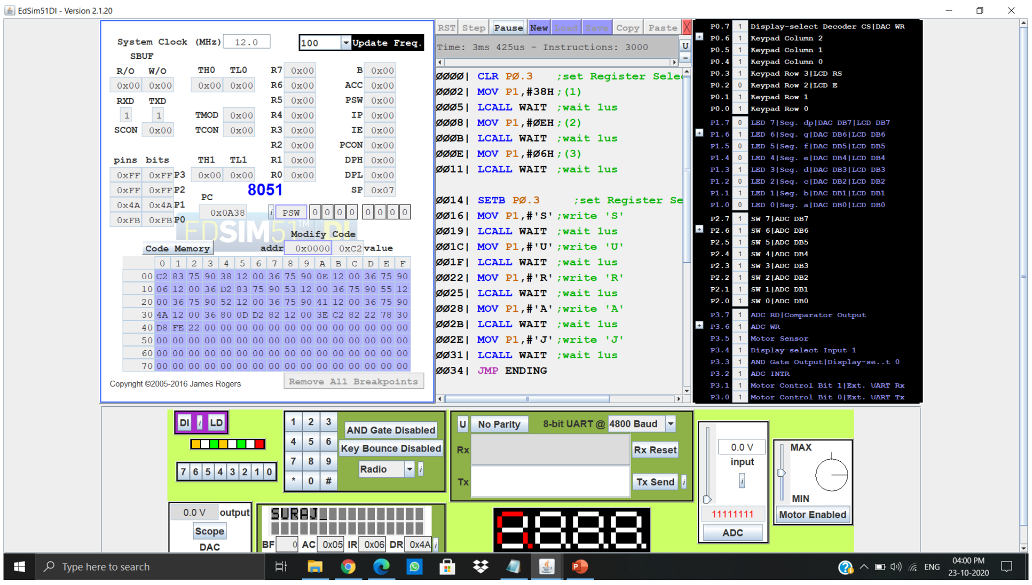
(1)Function set- Sets to 8-bit operation and selects 2-line display and 5 × 8 dot character font.

(2)Display on/off control- Turns on display and cursor. All display is in space mode because of initialization.
(3)Entry mode set- Sets mode to increment the address by one and to shift the cursor to the right at the time of write to the DD/CGRAM. Display is not shifted.

C program for 8051 micro-controller:

```c
#include<reg51.h>
unsigned int i, x;
void delay(){
        x = 0x30;
        while(x--){}
}
void wait(){
        P2 = 0x04;
        delay();
        P2 = 0x00;
}
void main(){
        P3 = 0x00;
        P1 = 0x38;
        wait();
        P1 = 0x0E;
        wait();
        P1 = 0x06;
        wait();

        P3 = 0x08;
        P1 = 'S';
        wait();
        P1 = 'U';
        wait();
        P1 = 'R';
        wait();
        P1 = 'A';
        wait();
        P1 = 'J';
        wait();
}
```

Output: The student name was successfully displayed on the LCD peripheral using EdSim51 as simulator.



```
Time: 3ms 425us - Instructions: 3000

ØØØØ| CLR PØ.3    ;set Register Sele
ØØØ2| MOV P1,#38H ;(1)
ØØØ5| LCALL WAIT  ;wait 1us
ØØØ8| MOV P1,#ØEH ;(2)
ØØØB| LCALL WAIT  ;wait 1us
ØØØE| MOV P1,#Ø6H ;(3)
ØØ11| LCALL WAIT  ;wait 1us

ØØ14| SETB PØ.3      ;set Register Se
ØØ16| MOV P1,#'S' ;write 'S'
ØØ19| LCALL WAIT  ;wait 1us
ØØ1C| MOV P1,#'U' ;write 'U'
ØØ1F| LCALL WAIT  ;wait 1us
ØØ22| MOV P1,#'R' ;write 'R'
ØØ25| LCALL WAIT  ;wait 1us
ØØ28| MOV P1,#'A' ;write 'A'
ØØ2B| LCALL WAIT  ;wait 1us
ØØ2E| MOV P1,#'J' ;write 'J'
ØØ31| LCALL WAIT  ;wait 1us
ØØ34| JMP ENDING
```

Figure 10: Snap of Simulator.

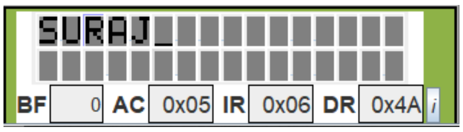


Figure 11: Student name as displayed on LCD.

Discussion and Observation: Three things are needed to be done before writing data to the CGRAM:
1. Function set
2. display on/off control
3. Entry mode set

Trigger function- The LCD is a negative edge triggered device, i.e., it changes its state only when the input signal becomes low.
This is the reason why the enable pin is set and reset (consuming 1 μs) after each operation whether it be the initialization of LCD or writing to the CGRAM.

Conclusion: It can be concluded that a LCD can be interfaced to 8051 micro-controller and can be used as a very effective way to output desired information from the micro-controller.

Experiment-5: Interfacing Motor- Counting number of revolutions:

Aim: Count the number of rotations made by the motor.

Problem Statement: Write an assembly language program for 8051 micro-controller to interface the motor with 8051 and count the number of rotations made by the motor.

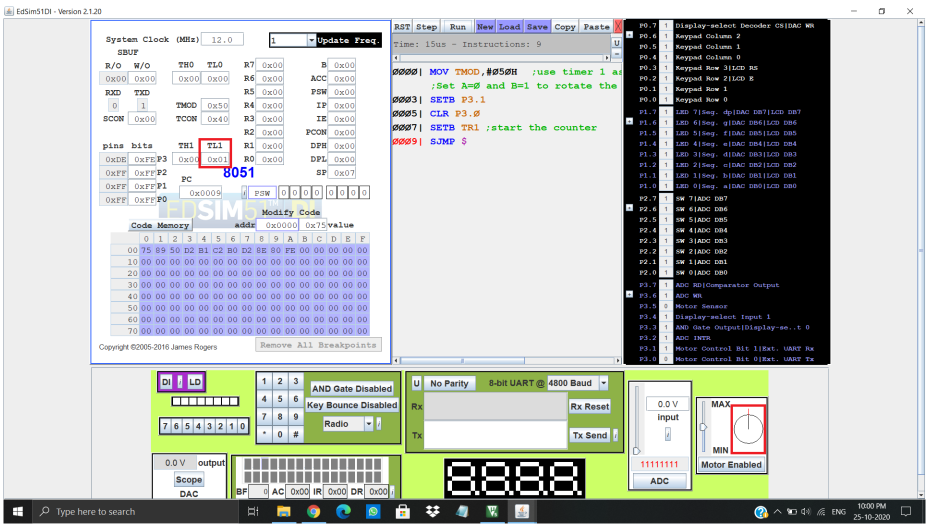
Code: The assembly language program for 8051 micro-controller:

```
MOV TMOD,#050H        ;use timer 1 as counter
;Set A=0 and B=1 to rotate the motor in forward direction
SETB P3.1
CLR P3.0
SETB TR1          ;start the counter
SJMP $
```

C program for 8051 micro-controller:

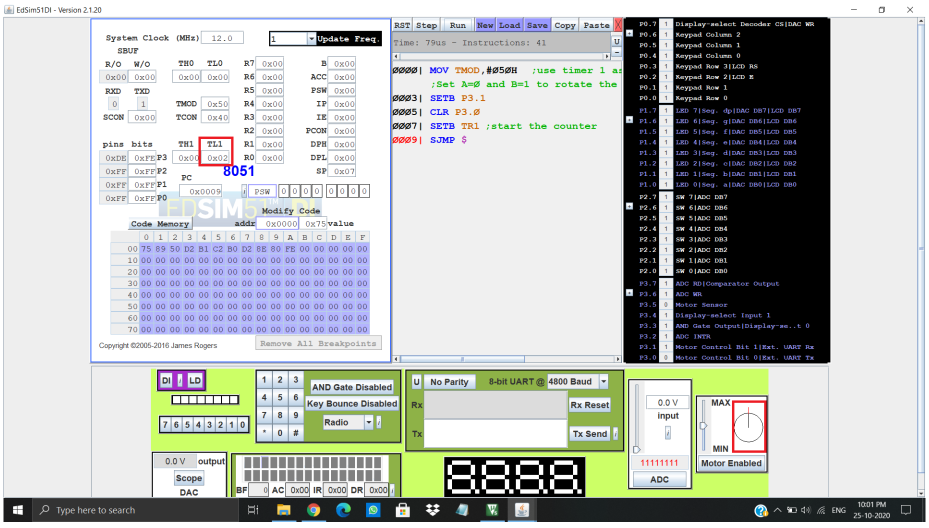
```
#include<reg51.h>
unsigned int x, i;
void main(){
        P3 = 0x02;
        TMOD = 0X50;
        while(1){}
}
```

Output: After every rotation, the count of number of rotations is updated in the counter 1 (TH1 and TL1).



```
System Clock (MHz) 12.0        1  Update Freq.
SBUF
R/O   W/O     TH0   TL0    R7 0x00    B    0x00
0x00  0x00    0x00  0x00   R6 0x00    ACC  0x00
RXD   TXD                  R5 0x00    PSW  0x00
0     1       TMOD  0x50   R4 0x00    IP   0x00
SCON  0x00    TCON  0x40   R3 0x00    IE   0x00
                           R2 0x00    PCON 0x00
pins  bits    TH1   TL1    R1 0x00    DPH  0x00
0xDE  0xFE P3 0x00  0x01   R0 0x00    DPL  0x00
0xFF  0xFF P2                         SP   0x07
0xFF  0xFF P1  PC 0x0009        PSW 0 0 0 0 0 0 0 0
0xFF  0xFF P0
Code Memory          Modify Code addr 0x0000  0x75 value
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 75 89 50 D2 B1 C2 B0 D2 8E 80 FE 00 00 00 00 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Time: 15us - Instructions: 9
0000| MOV TMOD,#050H   ;use timer 1 as
      ;Set A=0 and B=1 to rotate the
0003| SETB P3.1
0005| CLR P3.0
0007| SETB TR1 ;start the counter
0009| SJMP $
```

Figure 12: After 1 rotation, TL1 = 1.



```
Time: 79us - Instructions: 41

0000| MOV TMOD,#050H  ;use timer 1 as
      ;Set A=0 and B=1 to rotate the
0003| SETB P3.1
0005| CLR P3.0
0007| SETB TR1 ;start the counter
0009| SJMP $
```

Figure 13: After 2 rotations, TL1 = 2.

Discussion and Observation: The number of rotations made by the motor are directly reflected at counter 1.The pins P3.0 and P3.1 are applied to a dual bridge driver, the outputs of which are applied to a bi-directional DC motor. The truth table for the bridge and its effect on the motor is:

A	**B**	**motor**
0	0	stop
0	0	forward
1	0	reverse
1	1	stop

Conclusion: It can be concluded that the the motor can be interfaced with the 8051 micro-controller and the information regarding the number of rotations of the motor can be retrieved by the user while having control over its rotation.

Experiment-6: Display a square wave of frequency 1kHz using timers

Aim: Generate square wave of frequency 1kHz using timers.

Problem Statement: Write an assembly language program and a C language program for 8051 micro-controller to display a square wave of frequency 1kHz using timers.

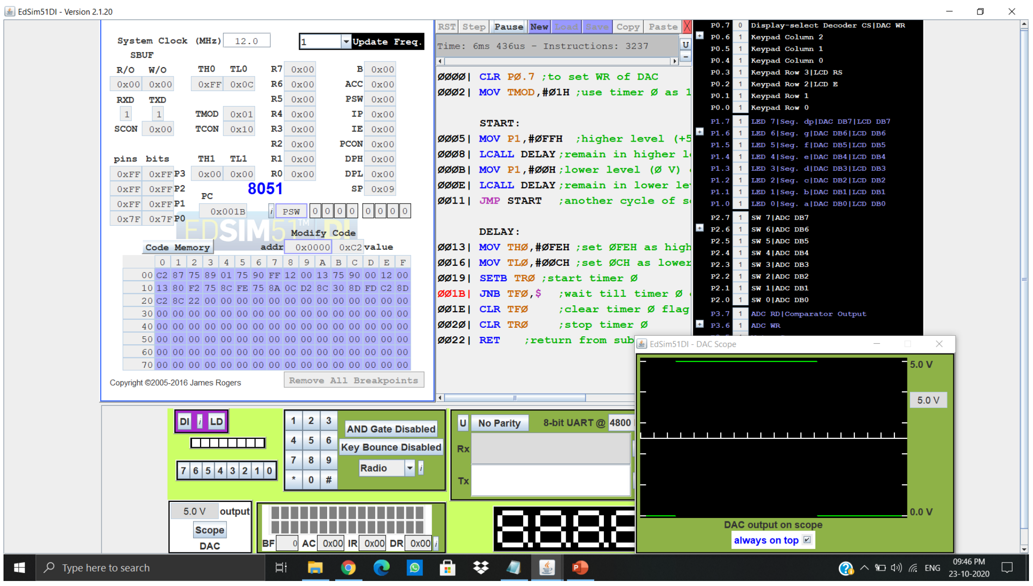
Code: The assembly language program for 8051 micro-controller:

```
CLR P0.7          ;operate DAC in wite mode (active low pin)
MOV TMOD,#01H          ;use timer 0 as 16bit timer

START:
MOV P1,#0FFH          ;higher level (+5 V) of square wave
LCALL DELAY          ;remain in higher level for 500 £\mu£s
MOV P1,#00H          ;lower level (0 V) of square wave
LCALL DELAY          ;remain in lower level for 500 £\mu£s
JMP START          ;another cycle of square wave

DELAY:
MOV TH0,#0FEH          ;set 0FEH as higher byte of timer 0
MOV TL0,#00CH          ;set 0CH as lower byte of timer 0
SETB TR0          ;start timer 0
JNB TF0,$           ;wait till timer 0 overflows (TF0 becomes 1)
CLR TF0              ;clear timer 0 flag
CLR TR0          ;stop timer 0
RET              ;return from subroutine
```

C program for 8051 micro-controller:

```
#include<reg51.h>
void delay(){
        TMOD = 0x01;
        TH0 = 0xFE;
        TL0 = 0x0C;
        TR0 = 1;
        while(TF0 == 0){}
        TR0 = 0;
        TF0 = 0;
}
void main(){
        P0 = 0x00;
        while(1){
                P1 = 0xFF;
                delay();
                P1 = 0x00;
                delay();
        }
}
```

Output: The square wave was successfully plotted with the help of timer 0.



```
Time: 6ms 436us - Instructions: 3237

0000| CLR P0.7 ;to set WR of DAC
0002| MOV TMOD,#01H ;use timer 0 as 1

      START:
0005| MOV P1,#0FFH  ;higher level (+5
0008| LCALL DELAY ;remain in higher le
000B| MOV P1,#00H ;lower level (0 V)
000E| LCALL DELAY ;remain in lower le
0011| JMP START   ;another cycle of s

      DELAY:
0013| MOV TH0,#0FEH ;set 0FEH as high
0016| MOV TL0,#00CH ;set 0CH as lower
0019| SETB TR0 ;start timer 0
001B| JNB TF0,$   ;wait till timer 0
001E| CLR TF0     ;clear timer 0 flag
0020| CLR TR0     ;stop timer 0
0022| RET    ;return from sub
```

Figure 14: Snap of Simulator.

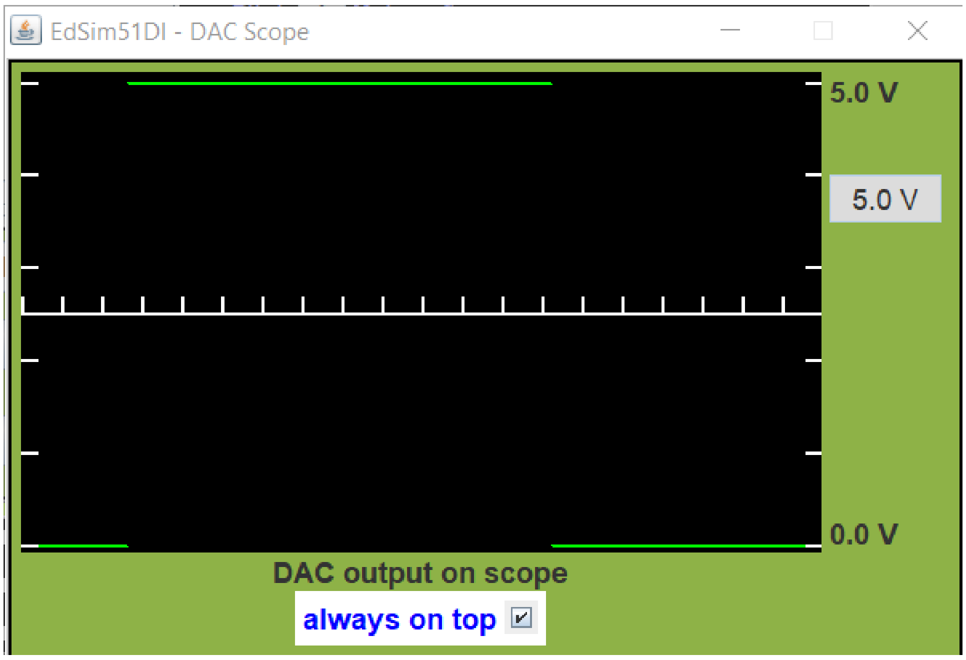


Figure 15: An instance of square wave as displayed on the scope.

Discussion and Observation: The processor is clocked at 12 MHz. So, the timer clock input will be

$$12MHz/12 = 1MHz$$

Hence, the time taken for the timer to make one increment will be

$$1/1MHz = 10^{-6}s = 1\mu s$$

Frequency f is given as

$$f = 1KHz$$

So, Time period T will be

$$T = 1/1KHz = 10^{-3}s = 1ms = 1000\mu s$$

Half of time period will be the time for which the wave is in either high or low level. So,

$$T_{half} = T/2 = 1000/2 = 500\mu s$$

So, for 500 μs the number of increments is 500.
Since the timers are of 16 bits, the maximum increments that can be made are

$$2^{16} = 65536$$

So, to produce a delay of 500 μs set the timer to the value

$$65536–500 = 65036$$

so that after 500 increments the timer overflows as a result the TF0 bit (timer 0 flag) gets SET.

$$(65036)_D = (FE0C)_H$$

So,

$$TH0 = (FE)_H \quad \text{and} \quad TL0 = (0C)_H$$

Conclusion: It can be concluded that a delay can be produced by using the timers by setting the timer value appropriately.

Experiment-7: Display a square wave of frequency 2kHz using counters

Aim: Generate square wave of frequency 2kHz using counters.

Problem Statement: Write an assembly language program for 8051 micro-controller to display a square wave of frequency 2kHz using counters.

Code: The assembly language program for 8051 micro-controller:

```
CLR P0.7        ;operate DAC in wite mode (active low pin)
MOV TMOD,#51H  ;use timer 0 as 16 bit timer & timer 1 as 16 bit counter

START:
MOV P1,#0FFH        ;higher level (+5 V) of square wave
LCALL DELAY        ;remain in higher level for 250£\mu£s
MOV P1,#00H        ;lower level (0 V) of square wave
LCALL DELAY        ;remain in lower level for 250£\mu£s
JMP START        ;another cycle of square wave

DELAY:
MOV TH0,#0FFH        ;set 0FEH as higher byte of timer 0
MOV TL0,#006H        ;set 006H as lower byte of timer 0
SETB TR1            ;start counter 1
SETB TR0            ;start timer 0

COUNTER:
SETB P3.5
CLR P3.5        ;(1)
JNB TF0,COUNTER        ;wait till timer 0 overflows, indicated by TF0

MOV R0,TL1
CLR TF0                ;clear timer 0 flag
CLR TR0                ;stop timer 0
CLR TR1                ;stop counter 1
MOV TL1,#00H        ;clear counter
RET                ;return from subroutine
```

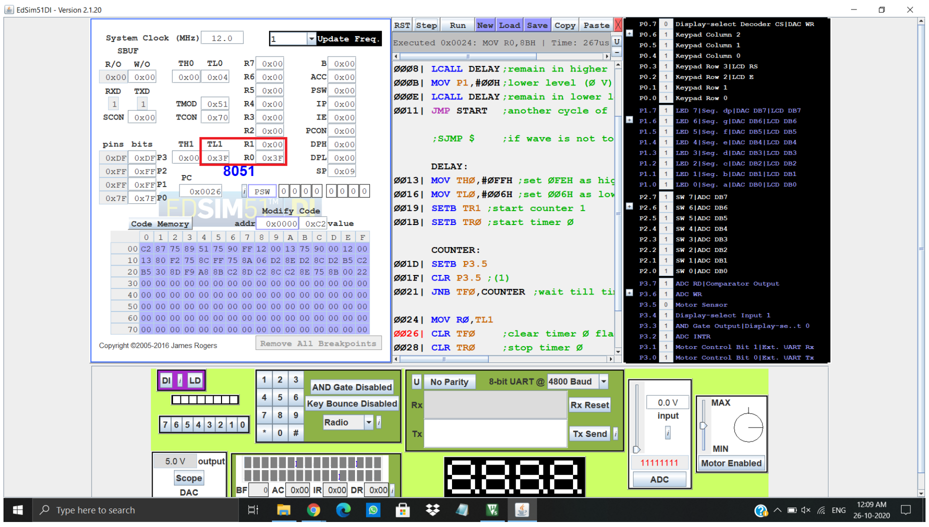
(1) counter increments by 1 when the T1 pin (P3.5) transits from 1 to 0 (high to low)

C program for 8051 micro-controller:

```
#include<reg51.h>
unsigned int x;
void delay(){
        TMOD = 0X51;
        TH0 = 0xFE;
        TL0 = 0x06;
        TR0 = 1;
        TR1 = 1;
        while(TF0 == 0){
                P3 = 0x20;
                P3 = 0x00;
        }
        x = TL1;
        TF0 = 0;
        TR0 = 0;
        TR1 = 0;
        TL1 = 0;
}
void main(){
        P0 = 0x00;
        while(1){
                P1 = 0xFF;
                delay();
                P1 = 0x00;
                delay();
        }
}
```

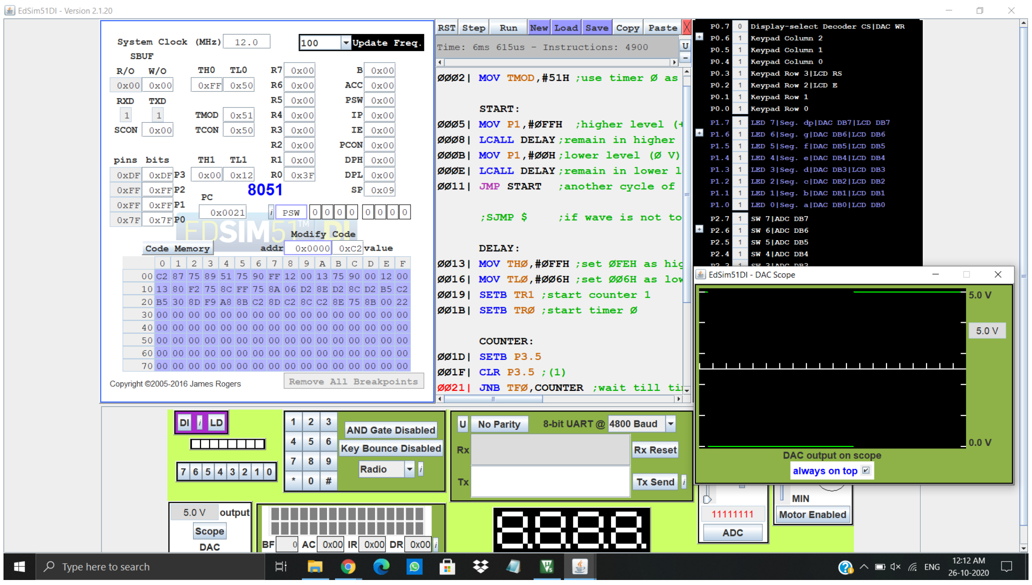
The output of counter is stored in x.

Output: The square wave was successfully plotted with the help of timer 0.



```
Executed 0x0024: MOV R0,8BH | Time: 267us
ØØØ8| LCALL DELAY ;remain in higher
ØØØB| MOV P1,#ØØH ;lower level (Ø V)
ØØØE| LCALL DELAY ;remain in lower l
ØØ11| JMP START   ;another cycle of

      ;SJMP $     ;if wave is not to

      DELAY:
ØØ13| MOV THØ,#ØFFH ;set ØFEH as hig
ØØ16| MOV TLØ,#ØØ6H ;set ØØ6H as low
ØØ19| SETB TR1 ;start counter 1
ØØ1B| SETB TRØ ;start timer Ø

      COUNTER:
ØØ1D| SETB P3.5
ØØ1F| CLR P3.5 ;(1)
ØØ21| JNB TFØ,COUNTER ;wait till ti

ØØ24| MOV RØ,TL1
ØØ26| CLR TFØ     ;clear timer Ø fla
ØØ28| CLR TRØ     ;stop timer Ø
```

Figure 16: Counter output stored in register R0



```
Time: 6ms 615us - Instructions: 4900

0002| MOV TMOD,#51H ;use timer 0 as

      START:
0005| MOV P1,#0FFH  ;higher level (+
0008| LCALL DELAY ;remain in higher
000B| MOV P1,#00H ;lower level (0 V)
000E| LCALL DELAY ;remain in lower l
0011| JMP START   ;another cycle of

      ;SJMP $     ;if wave is not to

      DELAY:
0013| MOV TH0,#0FFH ;set 0FEH as hig
0016| MOV TL0,#006H ;set 006H as low
0019| SETB TR1 ;start counter 1
001B| SETB TR0 ;start timer 0

      COUNTER:
001D| SETB P3.5
001F| CLR P3.5 ;(1)
0021| JNB TF0,COUNTER ;wait till ti
```

Figure 17: Snap of Simulator.

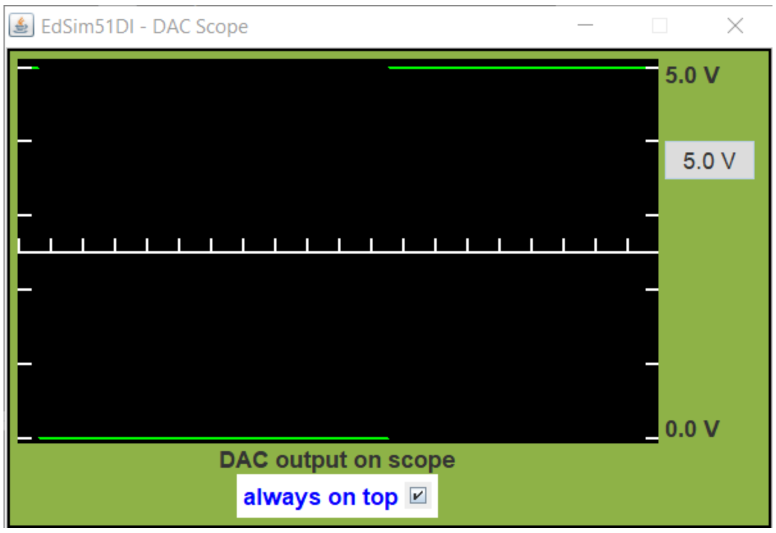


Figure 18: An instance of square wave as displayed on the scope.

Discussion and Observation: The processor is clocked at 12 MHz. So, the timer clock input will be

$$12MHz/12 = 1MHz$$

Hence, the time taken for the timer to make one increment will be

$$1/1MHz = 10^{-6}s = 1\mu s$$

Frequency f is given as

$$f = 2KHz$$

So, Time period T will be

$$T = 1/2KHz = 0.5 \times 10^{-3}s = 0.5ms = 500\mu s$$

Half of time period will be the time for which the wave is in either high or low level. So,

$$T_{half} = T/2 = 500/2 = 250\mu s$$

So, for 250 μs the number of increments is 250.
Since the timers are of 16 bits, the maximum increments that can be made are

$$2^{16} = 65536$$

So, to produce a delay of 250 μs set the timer to the value

$$65536–250 = 65286$$

so that after 250 increments the timer overflows as a result the TF0 bit (timer 0 flag) gets SET.

$$(65286)_D = (FE06)_H$$

So,

$$TH0 = (FE)_H \quad \text{and} \quad TL0 = (06)_H$$

The final thing required here is verifying whether 250μs delay has been correctly produced or not. This can be done by using a counter. This counter increments by 1 every time before checking the status of the TF0 flag.
The counter increments by 1 when the T1 pin (P3.5) transits from 1 to 0 (high to low). This transition is achieved by setting and resetting the T1 pin (hence, the transition consumes 2 μs). Then the status of TF0 flag is checked (this consumes 2 μs).
The counter should have incremented by 1 every 1 μs, whereas it is incremented every 4 μs.
So, the value received from the counter has to be multiplied by 4 to get the number of increments which in-turn tells the amount of delay produced (in μs) between high and low states.
The output of counter is stored in R0 register, which comes out to be $(3\mathrm{F})_\mathrm{H}$.

$$(3F)_H = (63)_D$$

$$63 \times 4 = 252\mu s$$

So, actual time-period T_{actual} is

$$T_{actual} = 2 \times 252 = 504\mu s$$

Hence, actual frequency f_{actual} is

$$f_{actual} = 1/T_{actual} = 1/(504 \times 10^{-6}) = 1.984kHz$$

<u>Conclusion</u>: It can be concluded that the square wave is produced of frequency 1.984 kHz with an error of 0.016 kHz.

Experiment-8: Display a sine wave using C language without using a look-up table

Aim: Generate sine wave without using a look-up table.

Problem Statement: Write a C language program for 8051 micro-controller to generate an approximate sine wave with peak-to-peak amplitude as 5V.

Code: C program for 8051 micro-controller:

```
#include <reg51.h>
#include <math.h>
unsigned int i;
float w = (2*3.14159)/40, s;
void main(){
        P0 = 0x00;
        while(1){
                for(i=0; i<40; i++){
                        s = 128 + 127*sin(w*i);
                        P1 = (int)s;
                }
        }
}
```

Output: The values obtained at P1 were observed and confirmed with the values used in the look-up table of Experiment-2.

Look-up table from Experiment-2:
128,147,167,185,202,217,230,241,248,253,255,253,248,241,230,217,202,185,167,147,
128,108,88,70,53,38,25,14,7,2,1,2,7,14,25,38,53,70,88,108.

Values obtained from C code:
128,147,167,185,202,217,230,241,248,253,255,253,248,241,230,217,202,185,167,147,
128,108,88,70,53,38,25,14,7,2,1,2,7,14,25,38,53,70,88,108.

Discussion and Observation: Every sample value can be determined from the formula

$$i^{th} sample = 128 + 127\sin(\frac{2\pi}{40}i).$$

where i varies from 0 to 39 (*i.e.,* 40 samples). More number of samples can also be taken but I preferred using 40 samples so as to verify this result with look-up tables of Experiment-2.

Conclusion: It can be concluded that a micro-controller can be used to generate a sinusoidal wave. One of the ways is to use the *sin()* function from *math.h* header file. Simultaneously every sample of the wave is generated and fed to Port 1 (P1).

Experiment-9: Serial Communication: Sending name to serial port

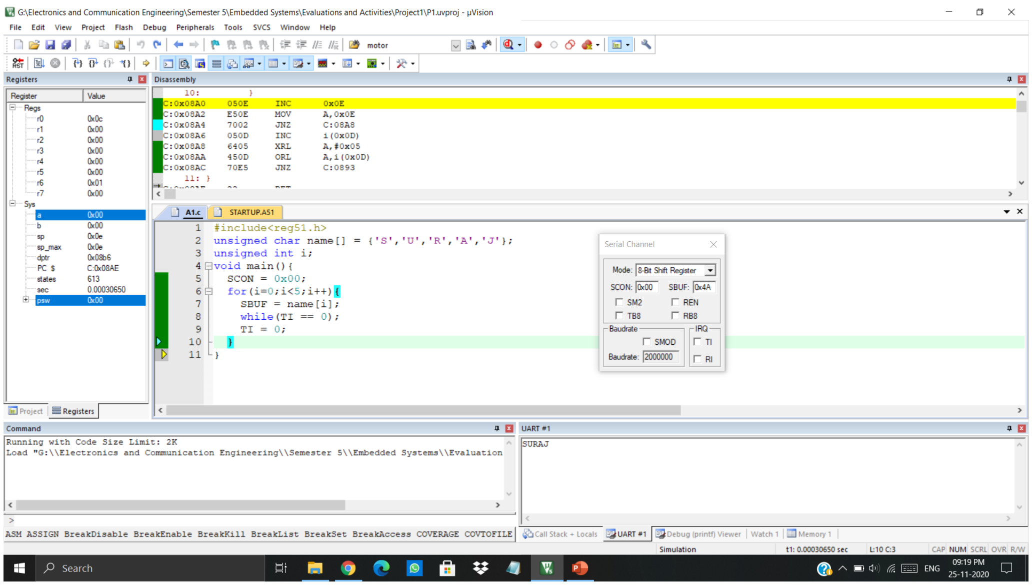
Aim: Serial Communication: Sending name to serial port.

Problem Statement: Write a C language program for 8051 micro-controller to send student name to serial port.

Code: C program for 8051 micro-controller:

```c
#include<reg51.h>
unsigned char name[] = {'S','U','R','A','J'};
unsigned int i;
void main(){
        SCON = 0x00;                            // set mode 0
        for(i=0;i<5;i++){
                SBUF = name[i];         // send data to SBUF register
                while(TI == 0);         // wait for completion
                TI = 0;         // clear TI before sending next
        }
}
```

Output: Student name was successfully sent to the serial port and can be seen on the UART#1 serial window in Keil μVision.



G:\Electronics and Communication Engineering\Semester 5\Embedded Systems\Evaluations and Activities\Project1\P1.uvproj - µVision

Registers

Register	Value
Regs	
r0	0x0c
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x01
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x0e
sp_max	0x0e
dptr	0x08b6
PC $	C:0x08AE
states	613
sec	0.00030650
psw	0x00

Disassembly

```
    10:         }
C:0x08A0    050E     INC      0x0E
C:0x08A2    E50E     MOV      A,0x0E
C:0x08A4    7002     JNZ      C:08A8
C:0x08A6    050D     INC      i(0x0D)
C:0x08A8    6405     XRL      A,#0x05
C:0x08AA    450D     ORL      A,i(0x0D)
C:0x08AC    70E5     JNZ      C:0893
    11: }
```

A1.c

```c
#include<reg51.h>
unsigned char name[] = {'S','U','R','A','J'};
unsigned int i;
void main(){
  SCON = 0x00;
  for(i=0;i<5;i++){
    SBUF = name[i];
    while(TI == 0);
    TI = 0;
  }
}
```

Serial Channel

Mode: 8-Bit Shift Register
SCON: 0x00 SBUF: 0x4A
☐ SM2 ☐ REN
☐ TB8 ☐ RB8
Baudrate: ☐ SMOD; Baudrate: 2000000
IRQ: ☐ TI ☐ RI

Command

```
Running with Code Size Limit: 2K
Load "G:\\Electronics and Communication Engineering\\Semester 5\\Embedded Systems\\Evaluation
```

UART #1

```
SURAJ
```

Simulation t1: 0.00030650 sec L:10 C:3

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
motor
STARTUP.A51
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE
Call Stack + Locals
Debug (printf) Viewer
Watch 1
Memory 1
Project
Registers
Search
ENG
09:19 PM
25-11-2020

Figure 19: Snap of Simulator.

Discussion and Observation: It is observed that there is some waiting time while sending data to the SBUF register. This is because data is being fed serially (one bit after the other). Hence, there comes the need of the TI flag which signals that the line is available to transmit a new byte.

Conclusion: It can be concluded that the need of multiple wires can be reduced by introducing the concept of serial communication.

Experiment-10: I2C Communication: Writing to a slave

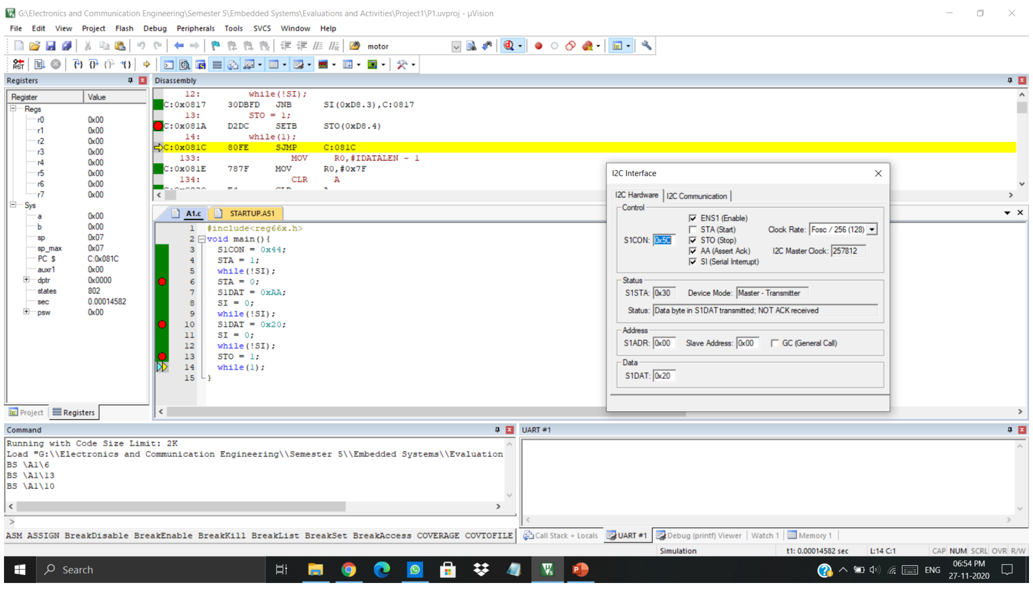
Aim: I2C Communication: Writing to a slave.

Problem Statement: Write a C language program for 8051 micro-controller to write 0010 0000 (20H) to a slave with the address 1010101 (7 bit only!, the 8th bit is reserved for Read/Write bit).

Code: C program for 8051 micro-controller:

```c
#include <reg66x.h>
void main(){
        S1CON = 0x44;
        STA = 1;
        while(!SI);
        STA = 0;
        S1DAT = 0xAA;
        SI = 0;
        while(!SI);
        S1DAT = 0x20;
        SI = 0;
        while(!SI);
        STO = 1;
        while(1);
}
```

Output: The master slave communication gave the following results.



G:\Electronics and Communication Engineering\Semester 5\Embedded Systems\Evaluations and Activities\Project1\P1.uvproj - µVision

Registers

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
PC $	C:0x081C
auxr1	0x00
dptr	0x0000
states	802
sec	0.00014582
psw	0x00

Disassembly

```
    12:         while(!SI);
C:0x0817    30DBFD   JNB      SI(0xD8.3),C:0817
    13:         STO = 1;
C:0x081A    D2DC     SETB     STO(0xD8.4)
    14:         while(1);
C:0x081C    80FE     SJMP     C:081C
   133:                 MOV      R0,#IDATALEN - 1
C:0x081E    787F     MOV      R0,#0x7F
   134:                 CLR      A
```

A1.c

```c
#include<reg66x.h>
void main(){
  S1CON = 0x44;
  STA = 1;
  while(!SI);
  STA = 0;
  S1DAT = 0xAA;
  SI = 0;
  while(!SI);
  S1DAT = 0x20;
  SI = 0;
  while(!SI);
  STO = 1;
  while(1);
}
```

I2C Interface — I2C Hardware

Control
S1CON: 0x5C
[✓] ENS1 (Enable)
[ ] STA (Start)
[✓] STO (Stop)
[✓] AA (Assert Ack)
[✓] SI (Serial Interrupt)
Clock Rate: Fosc / 256 (128)
I2C Master Clock: 257812

Status
S1STA: 0x30 Device Mode: Master - Transmitter
Status: Data byte in S1DAT transmitted; NOT ACK received

Address
S1ADR: 0x00 Slave Address: 0x00 [ ] GC (General Call)

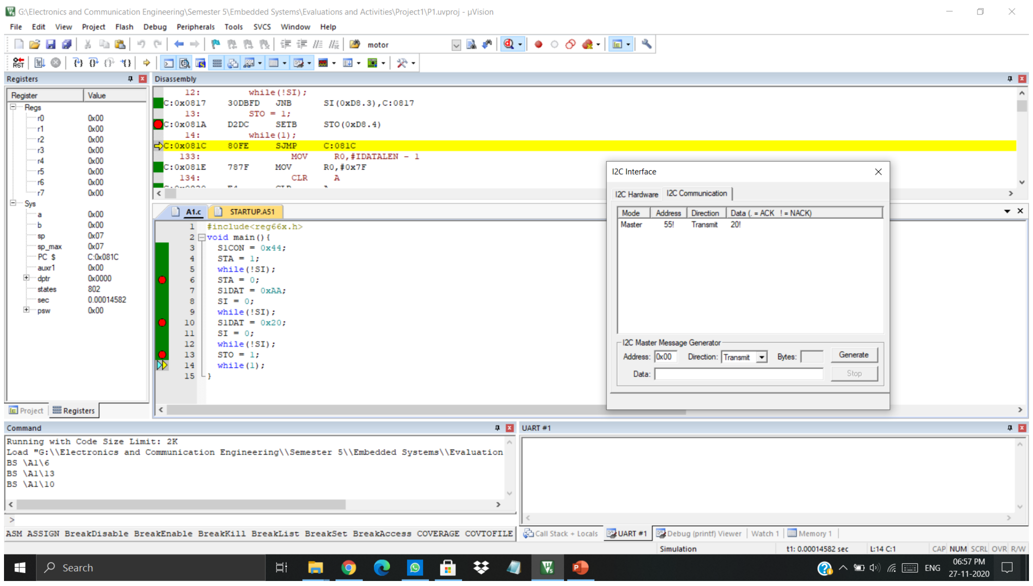
Data
S1DAT: 0x20

Command

```
Running with Code Size Limit: 2K
Load "G:\\Electronics and Communication Engineering\\Semester 5\\Embedded Systems\\Evaluation
BS \A1\6
BS \A1\13
BS \A1\10
```

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
motor
STARTUP.A51
I2C Communication
UART #1
Project Registers
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE
Call Stack + Locals UART #1 Debug (printf) Viewer Watch 1 Memory 1
Simulation t1: 0.00014582 sec L:14 C:1
Search
ENG 06:54 PM 27-11-2020

Figure 20: Snap of Simulator.


G:\Electronics and Communication Engineering\Semester 5\Embedded Systems\Evaluations and Activities\Project1\P1.uvproj - µVision

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
PC $	C:0x081C
auxr1	0x00
dptr	0x0000
states	802
sec	0.00014582
psw	0x00

```
    12:         while(!SI);
C:0x0817    30DBFD   JNB      SI(0xD8.3),C:0817
    13:         STO = 1;
C:0x081A    D2DC     SETB     STO(0xD8.4)
    14:         while(1);
C:0x081C    80FE     SJMP     C:081C
   133:                 MOV      R0,#IDATALEN - 1
C:0x081E    787F     MOV      R0,#0x7F
   134:                 CLR      A
```

A1.c

```
#include<reg66x.h>
void main(){
  S1CON = 0x44;
  STA = 1;
  while(!SI);
  STA = 0;
  S1DAT = 0xAA;
  SI = 0;
  while(!SI);
  S1DAT = 0x20;
  SI = 0;
  while(!SI);
  STO = 1;
  while(1);
}
```

I2C Interface

Mode	Address	Direction	Data (. = ACK ! = NACK)
Master	55!	Transmit	20!

I2C Master Message Generator — Address: 0x00 Direction: Transmit Bytes: Data:

```
Running with Code Size Limit: 2K
Load "G:\\Electronics and Communication Engineering\\Semester 5\\Embedded Systems\\Evaluation
BS \A1\6
BS \A1\13
BS \A1\10
```

Figure 21: Snap of Simulator.

<u>Discussion and Observation</u>: Such type of master-slave communication can be used in operations like adjusting screen brightness, contrast, hue, volume control, etc.

<u>Conclusion</u>: Key points like Modes of Operation of SIO1 logic and Master Transmitter Mode were studied from the P89C66X datasheet and accordingly the task was completed.