Software experiments	
#	**Introduction**
1.	In a class of strength 10 students, I need to find the tallest guy in the class to reach the projector of the classroom. Help me to identify the tallest guy.
2.	To determine the number of ways to distribute 'N' marbles to a group of children you need to find factorial of N. Hence find factorial of a number.
3.	To illustrate the working of lookup table in ARM processor, find the square of a number (1 to 10) stored in look-up table.
4.	RAM is an important component of microcontroller. To understand how it can be accessed, add an array of 16 bit numbers and store the 32 bit result stored in internal RAM.
5.	Write an ALP to count the number of ones and zeros in two consecutive memory locations

Hardware experiments	
#	**Introduction**
1.	In the retail shop, the owner wants to know how many customers have been using the service of the shop. A device is used to keep count of incoming customers in the shop. Program the device to carry out the counting. Also the device should reset back to ZERO by down counting. Implement the above using Logic Controller Interface.
2.	In a petrol bunk, if a fire occurs accidently you need to display messages FIRE and HELP alternately on a 7-segment display interface to alarm the people. Implement the above scenario.
3.	A toy car uses a Stepper Motor interface to rotate the motor in specified direction. Program the toy to rotate (Clockwise or Counter-Clockwise) by N steps. Introduce suitable delay between successive steps. (Any arbitrary value for the delay may be assumed by the student).
4.	Generate Sine Wave using DAC interface (The output of the DAC is to be displayed on the CRO).
5.	LCD Interface
6.	Scan a 4X4 keypad for key pressed and display the key pressed on LCD screen.

Software experiments

1. In a class of strength 10 students, I need to find the tallest guy in the class to reach the projector of the classroom. Help me to identify the tallest guy.

```
        area big,code,readonly
        ldr r0,=tallest
        mov r1,#10
        ldr r3,[r0]
rpt      cmp r1,#1
        beq store
        sub r1,r1,#1
        add r0,r0,#4
        ldr r2,[r0]
        cmp r2,r3
        blt rpt
        mov r3,r2
        b rpt
store    b store

tallest         DCD 0X00000041
                DCD 0X00000024
                DCD 0X00000049
                DCD 0X00000030
                DCD 0X00000019
                DCD 0X00000080
                DCD 0X00000031
                DCD 0X00000040
                DCD 0X00000070
                DCD 0X00000056
                end
```

2. To determine the number of ways to distribute 'N' marbles to a group of children you need to find factorial of N. Hence find factorial of a number.

```
;PROGRAM TO FIND FACTORIAL OF A GIVEN NUMBER

; In this example we have taken n=7
; Check the result in R0/R3 register =13B0H (5040)
                AREA  fct, CODE, READONLY
                MOV r0, #7                      ; STORE FACTORIAL NUMBER IN R0
                MOV r1,r0                       ; MOVE THE SAME NUMBER IN R1
FACT
                CMP r1, #1                      ; COMPARISON
                BEQ XSS
                SUBS r1, r1, #1                 ; SUBTRACTION
                MUL r3,r0,r1            ; MULTIPLICATION
                MOV    r0,r3                                    ; Result
                b FACT                  ; BRANCH TO THE LOOP
XSS             END                                     ;Mark end of file
```

3. To illustrate the working of lookup table in ARM processor, find the square of a number (1 to 10) stored in look-up table.

```
        ;Assembly Program to find square of Number
        ;GIVEN NUMBER IS 6 (R1) THEN RESULT IS IN R3=24H(36)

                area data,code, readonly
                ldr r0,=TABLE1
                mov r1,#6
        loop    cmp r1,#1
                beq store
                sub r1,r1,#1
                add r0,r0,#4
                b loop

        store   ldr r3,[r0]

        XSS     B XSS

        ;Lookup table contains Squares of nos from 0 to 10 (in hex)

TABLE1          DCD 0X00000001  ;       SQUARE OF 1=1
                DCD 0X00000004  ;       SQUARE OF 2=4
                DCD 0X00000009  ;       SQUARE OF 3=9
                DCD 0X00000010  ;       SQUARE OF 4=16
                DCD 0X00000019  ;       SQUARE OF 5=25
                DCD 0X00000024  ;       SQUARE OF 6=36
                DCD 0X00000031  ;       SQUARE OF 7=49
```

```
DCD 0X00000040      ;       SQUARE OF 8=64
 DCD 0X00000051     ;       SQUARE OF 9=81
 DCD 0X00000064     ;       SQUARE OF 10=100
 end
```

4. RAM is an important component of microcontroller. To understand how it can be accessed, add an array of 16 bit numbers and store the 32 bit result stored in internal RAM.

```
;PROGRAM TO ADD an array of 16BIT NUMBERS & STORE IN INTERNAL RAM
;ARRAY OF 6 NUMBERS      0X1111,0X2222,0X3333,0XAAAA,0XBBBB,0XCCCC
; THE SUM IS 29997H THE RESULT CAN BE VIEWED IN LOCATION 0X40000000 & ALSO IN
R0

        AREA  ADDITION , CODE, READONLY

        MOV R5,#6                       ; INTIALISE COUNTER TO 6(i.e. N=6)
        MOV R0,#0                       ; INTIALISE SUM TO ZERO
        LDR R1,=VALUE1          ; LOADS THE ADDRESS OF FIRST VALUE
LOOP
        LDRH R2,[R1]            ; LOAD HALF WORD
        ADD R0,R0,R2            ; ADD THE ELEMENTS
        SUBS R5,R5,#1                   ; DECREMENT COUNTER
        ADD R1,R1,#2
        CMP R5,#0
        BNE LOOP                        ;  LOOK BACK TILL ARRAY ENDS
        mov R4,#0x40000000              ; LOADS THE ADDRESS OF RESULT
        STR R0,[R4]                     ; STORES THE RESULT

XSS     B XSS

VALUE1 DCW          0X1111,0X2222,0X3333,0XAAAA,0XBBBB,0XCCCC
        ; ARRAY OF 16 BIT NUMBERS(N=6)
        END                     ; Mark end of file
```

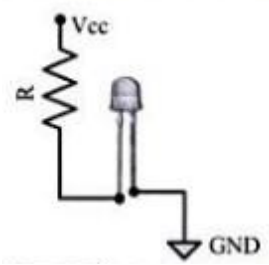
5. Write an ALP to count the number of ones and zeros in two consecutive memory locations

```
;PROGRAM TO COUNT THE NUMBER OF ONES & ZEROS IN TWO CONSECUTIVE
;MEMORY LOCATIONS
;CHECK THE RESULT IN R2 FOR ONES & R3 FOR ZEROS
                                */
                AREA  ONEZERO , CODE, READONLY

                MOV R2,#0                       ; COUNTER FOR ONES
                MOV R3,#0                       ; COUNTER FOR ZEROS
                MOV R7,#2                       ; COUNTER TO GET TWO WORDS
                LDR R6,=VALUE                   ; LOADS THE ADDRESS OF VALUE

LOOP            MOV R1,#32                      ; 32 BITS COUNTER
                 LDR R0,[R6],#4                 ; GET THE 32 BIT VALUE

LOOP0           MOVS R0,R0,ROR #1       ; RIGHT SHIFT TO CHECK CARRY BIT (1's/0's)
                BHI ONES                                        ; IF CARRY BIT IS 1
                                        ;GOTO ONES BRANCH OTHERWISE NEXT

ZEROS           ADD R3,R3,#1            ; IF CARRY BIT IS 0 THEN INCREMENT THE
                                        ;COUNTER BY 1(R3)
                B LOOP1                                 ; BRANCH TO LOOP1

ONES            ADD R2,R2,#1            ; IF CARRY BIT IS 1 THEN INCREMENT THE
                                        ;COUNTER BY 1(R2)

LOOP1           SUBS R1,R1,#1                   ; COUNTER VALUE DECREMENTED
                                                 ;BY 1
                BNE LOOP0                       ; IF NOT EQUAL GOTO TO LOOP0
                                                ;CHECKS 32BIT

                SUBS R7,R7,#1                   ; COUNTER VALUE DECREMENTED
                                                 ;BY 1
                CMP R7,#0                       ; COMPARE COUNTER R7 TO 0
                BNE LOOP                        ; IF NOT EQUAL GOTO TO LOOP


XSS             B XSS

VALUE           DCD 0X3,0X2                     ;   TWO VALUES IN AN ARRAY
                END                     ; Mark end of file
```

Hardware Interfacing Programs

1. Up-down counter on the Logic Controller Interface

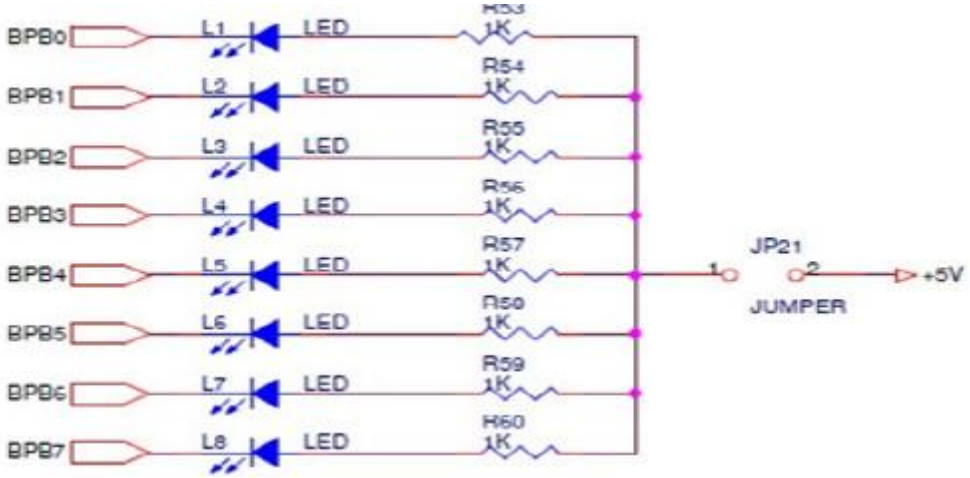
Light Emitting Diode (LED):-



The LED is a p-n junction diode, made up of semiconductors. LED is an output device for visual indication in any electronic device. LED can be interfaced to the port pins of a microcontroller in the following way:

The cathode of the LED is connected to the port pin and the anode to the supply voltage through resistors. The LED is on when the port pin is at logic 0 (Active Low).

Light Emitting Diodes (LED's) are components most commonly used for displaying the port line status. There are 8 LEDs on the board; these lines are connected to the Port lines P0.16 (PB0)(LSB) to P0.23 (PB7) (MSB) through buffer.



Active LOW output

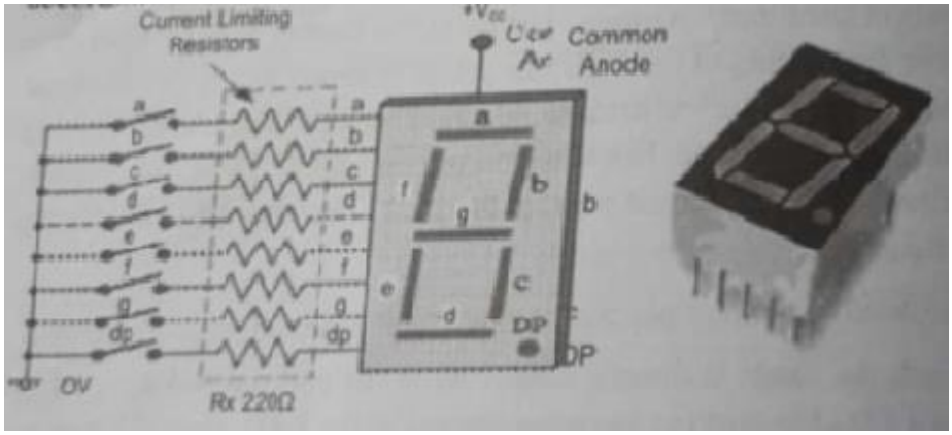
Result: Observe all the LED's L1 to L8 toggling with some delay when JP21 is shorted.

Delay: 1ms=3200 ticks

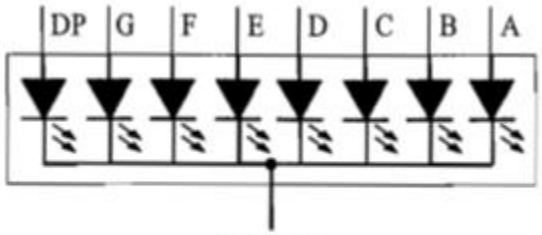
```
#include <LPC21xx.h>

void delay(void);
long hv;
int main ()
{
unsigned  nhv=0;

        PINSEL0 = 0x00000000;
        IO0DIR = 0x00FF0000;//enable for out port

        while(1)
        {
                for(hv=0; hv<= 0x20; hv++)
                {
                        nhv= (~hv);             // for incrementing display from 00 to 0x20
                        nhv=nhv& 0x000000ff;
                        IO0CLR=0x00ff0000;
                        IO0SET=(nhv<< 16);
                        delay();
                }
                for(hv=0x20; hv>= 0; hv--)
                {
                        nhv= (~hv);             // for decrementing display from 0x20 to 00
                        nhv=nhv& 0x000000ff;
                        IO0CLR=0x00ff0000;
                        IO0SET=(nhv<< 16);
                        delay();
                }

        }
}

void delay(void)
{
        unsigned  count;

for(count=0; count< 1385000; count++)
        {}
}
```

2. Display messages FIrE and HELP alternatively on a Seven Segment Display Interface

7-segment LED display is an output device for display of alpha numeric characters. It contains 7 LED segments in a special form to display characters. LED segments are named A to G and DP.



The following figure shows the common cathode configuration, where in cathode are common and 8 pins are connected to 8 segments.

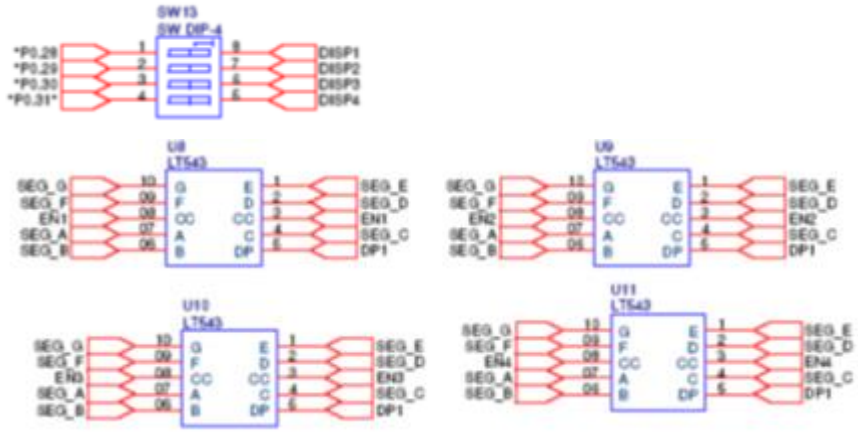


Common Cathode LED Display

There are four multiplexed 7-segment displays TL543 (U8 - U11) on the board. Each display has 8-inputs SEG_A (Pin-7), SEG_B (Pin-6), SEG_C (Pin-4), SEG_D (Pin-2), SEG_E(Pin-1), SEG_F (Pin-9), SEG_G (Pin-10) and DP (Pin-5). The remaining pins pin-3 & 8 is Common Cathode. The port lines P0.28 to P0.31 are used to select one of the 4 digits as shown in the table below. The port lines P0.16 to P0.23 are used as segment lines for the EIGHT digits through the 74HCT244 buffer (U7).

Selection Of seven segment displays:

P0.28	P0.29	P0.30	P0.31	Display unit selected
1	0	0	0	U8
0	1	0	0	U9
0	0	1	0	U10
0	0	0	1	U11



Keep all pins of SW13 in ON position and un short the jumper JP21. Short jumper JP2/ 1-2

Program

```
/*
a
  ----
f|  g |b
|----|
e|    |c
  ----  . dot
d
      a = P0.16
          b = P0.17
          c = P0.18
          d = P0.19
          e = P0.20
          f = P0.21
          g = P0.22
          dot = P0.23
Select lines for four 7 Segments
DIS1 P0.28
DIS2 P0.29
DIS3 P0.30
DIS4 P0.31
*/

#include<lpc21xx.h>
long delay;
int scroll,p;
int fire[4] = {0X00710000,0X00060000,0X00500000,0X00790000}; //FIrE
int help[4] = {0X00760000,0X00790000,0X00380000,0X00730000}; //HELP
int dispsel[4]={0x10000000,0x20000000,0x40000000,0x80000000};
```

```
int main (void)
 {
PINSEL0 = 0x00000000;
IO0DIR  = 0xF0FF0000;
while(1)
{
for(scroll =0,p=0;scroll<4;scroll++,p++)
{
IO0CLR  = 0xf0ff0000;
IO0SET = dispsel[scroll];
IO0SET = fire[p];   // display the values FIRE one after the other from right to left
for(delay=0;delay<100000;delay++)
{}

}

for(delay=0;delay<100000;delay++)
{}

for(scroll =0,p=0;scroll<4;scroll++,p++)
{
IO0CLR  = 0xf0ff0000;
IO0SET = dispsel[scroll];
IO0SET = help[p];   // display the values HELP one after the other from right to left
for(delay=0;delay<100000;delay++)
{}

}

}

 }
```

3. Drive a Stepper Motor interface to rotate the motor in clock-wise or counter-clockwise by N steps.

A stepper motor is a type of DC motor that rotates in steps, converts digital pulses into mechanical shaft rotation. Every revolution of the stepper motor is divided into a discrete number of steps, and the motor must be sent a separate pulse for each step. When electrical signal is applied to it, the motor rotates in steps and the speed of rotation depends on the rate at which the electrical signals are applied and the direction of rotation is dependent on the pattern of pulses.
Unipolar stepper motor contains 2 windings per phase. The direction of rotation of a stepper motor is controlled by changing the direction of current flow. Current in one direction flows in one coil and the opposite direction flows through in other coil.

Table 17-4 shows some step angles for various motors. In Table 17-4, notice the term *steps per revolution.* This is the total number of steps needed to rotate one complete rotation or 360 degrees (e.g., 180 steps x 2 degrees = 360).

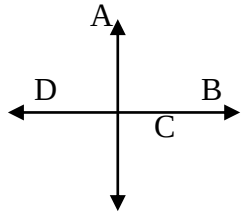
Step angle

Table 17-4: Stepper Motor Step Angles

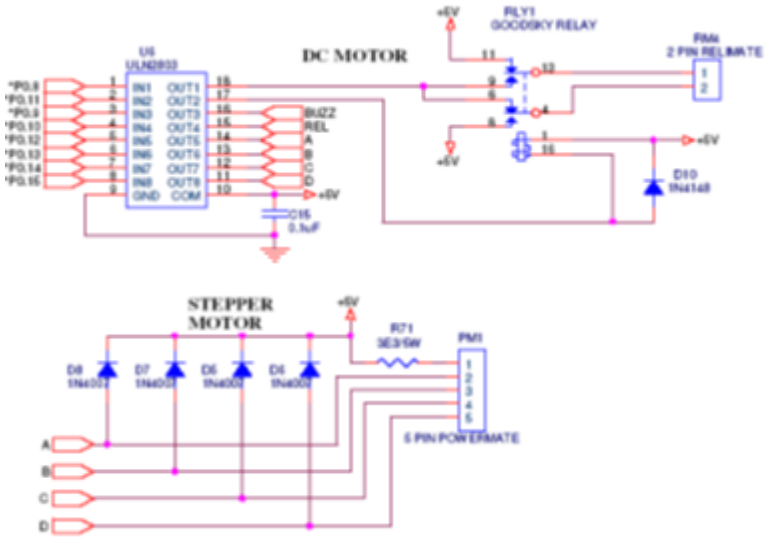
Step Angle	Steps per Revolution
0.72	500
1.8	200
2.0	180
2.5	144
5.0	72
7.5	48
15	24

The Stepper motor can be interfaced to the board by connecting it into the Power Mate PM1. The rotating direction of the stepper motor can be changed through software. Port lines used for Stepper motor are P0.12 – P0.15.

Connect the Female Powermate of the stepper motor to the male Powermate PM1 present on the board. Short JP13.
Result: The stepper motor rotates one rotation Clockwise & other rotation Anti Clockwise direction. This process is continuously in loop.



Current pass through A,B,C,D for clockwise direction and through D,C,B,A for anticlockwise direction.




```
/* A stepper motor direction is controlled by shifting the voltage across
the coils. Port lines : P0.12 to P0.15*/

 #include <LPC21xx.H>
Void clock_wise(void);
Void anti_clock_wise(void);
unsigned long int var1,var2;
unsigned int i=0,j=0,k=0;

int main(void)
 {
        PINSEL0 = 0x00000000;
        IO0DIR = 0x0000F000
```

```
while(1)

{
                for(j=0;j<50;j++)      // 200 times in Clock wise Rotation
                        clock_wise();

                for(k=0;k<65000;k++);   // Delay to show  anti_clock Rotation

                for(j=0;j<50;j++)      // 200 times in  Anti Clock wise Rotation
                        anti_clock_wise();
                for(k=0;k<65000;k++);   // Delay to show clock Rotation

        }                                                       // End of while(1)
}                                                               // End of main


Void clock_wise(void)
{
        var1 = 0x00000800;              //For Clockwise
for(i=0;i<=3;i++)           // for A B C D Stepping
        {
                var1 = var1<<1;         //For Clockwise
    IO0CLR=0x0000f000;
                IO0SET=var1;

for(k=0;k<3000;k++);     //for step speed variation
    }
}
Void anti_clock_wise(void)
{
        var1 = 0x00010000;      //For Anticlockwise
for(i=0;i<=3;i++)           // for A B C D Stepping
    {
     var1 = var1>>1;
                IO0CLR=0x0000f000;
                IO0SET=var1;
                for(k=0;k<3000;k++);     //for step speed variation

    } }
```

4. DAC controller

DAC0800 is used to convert the digital data into analog signals. Digital data from specified port lines is given to DAC input. Amplitude of output waveform can be varied by varying POT3 (5K Pot) that is by varying the reference voltage of DAC0800 when JP3 is closed. For Bipolar mode open jumper JP11. For Unipolar mode short jumper JP11. Port lines used for DAC are P0.16 – P0.23.

Sine Wave:

Result: Press the reset switch to run the program. Make sure that JP3 is shorted. Observethe Analog output waveform at the Pin-1 of RM3 using Oscilloscope (CRO) with respect to GND pin-2 of RM3. When jumper JP11 is shorted it is Unipolar mode whenjumper JP11 is opened it is Bipolar mode.

Vout=5+5sin(ω)
DAC i/p(convert to hex)=25.6 X Vout
ω=0,8,16,...90 for 0 to 90 degree(aprox.13 values), reverse for 90 to 0 degree; -8,-16,-24,...-90 for 0 to -90 degree(aprox.12 values), reverse for -90 to 0 degree.

ω	0	8	16	24	32	40	48	56	64	72	80	90	80	72	64	-	0	-8	-	-90
DAC i/p	80	91									FE	100	FE	-	-	-	80	6E		0

```
#include <LPC21xx.h>
int count,sinevalue,value;
unsigned int sine_tab[50]=
{0x80,0x91,0xA3,0xb4,0xC3,0xd2,0xdF,0xEA,0xF3,0xF9,0xFE,0x100,0xFE,0xF9,0xF3,

        0xEA,0xdF,0xd2,0xc3,0xb4,0xA3,0x91,0x80,0x6E,0x5c,0x4b,0x3c,0x2d,0x20,0x15,
                0x0c,0x06,0x01,0x00,0x01,0x06,0x0c,0x15,0x20,0x2d,0x3c,0x4b,0x5c,0x6E,0x80};
        int main(void)
{
        PINSEL0 = 0x00000000 ;                  // Configure P0.0 to P0.31 as GPIO
IO0DIR  = 0x00FF0000 ;
                while(1)
        {
                for(count=0;count<45;count++)
                {
                        sinevalue = sine_tab[count];
                        value=  (sinevalue<< 16);
                        IO0PIN = value;
                }
        }
}
```

CRO :- Adjust hold off, y-pos, x-pos, GD, ch i/ii, volts/div, time/div

5. LCD interface

LCD:

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments).
A **16x2 LCD** means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.
The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD.

LCD commands:

No	HEX Value	COMMAND TO LCD
1	0x01	Clear Display Screen
2	0x30	Function Set: 8-bit, 1 Line, 5x7 Dots
3	0x38	Function Set: 8-bit, 2 Line, 5x7 Dots
4	0x20	Function Set: 4-bit, 1 Line, 5x7 Dots
5	0x28	Function Set: 4-bit, 2 Line, 5x7 Dots
6	0x06	Entry Mode
7	0x08	Display off, Cursor off
8	0x0E	Display on, Cursor on
9	0x0C	Display on, Cursor off
10	0x0F	Display on, Cursor blinking
11	0x18	Shift entire display left
12	0x1C	Shift entire display right
13	0x10	Move cursor left by one character
14	0x14	Move cursor right by one character
15	0x80	Force cursor to beginning of 1st row
16	0xC0	Force cursor to beginning of 2nd row

A 16X2 Alphanumeric LCD Display.
The LCD is interfaced using 4 – bit mode.
RS = 0 for sending Command to the LCD, controlled by port P0.2
RS = 1 for sending Data to the LCD, controlled by port P0.2
R/W = 1 for reading from the LCD
R/W = 0 for writing to the LCD, normally it is grounded
EN = 0 for disabling the LCD
EN = 1 for enabling the LCD, controlled by port P0.3

D4= P0.4
D5= P0.5
D6= P0.6
D7= P0.7

```
/*Program to demonstrate
displaymesage on LCD screen*/
#include<lpc21xx.h>
#include<stdio.h>

voidlcd_init(void);
voidclr_disp(void);
voidlcd_com(void);
voidlcd_data(void);
voidwr_cn(void);
voidwr_dn(void);
void delay(unsigned int);
voidinit_port(void);
unsigned char temp,temp1;
unsignedint r,r1;

unsigned char *ptr,disp[] = "SHASHIDHARA";
unsigned char disp0[] = "MCE HASSAN";

int main()
{
        init_port();        //port intialisation
        delay(3200);                //delay for 1ms

        lcd_init();         //lcdintialisation
        delay(3200);                //delay
clr_disp();                         //clear display
        delay(500);          //delay

        ptr = disp;
        temp1 = 0x80;               // Display starting address, 1st line
        lcd_com();
        delay(800);

        while(*ptr!='\0')
   {
        temp1 = *ptr;
lcd_data();
        ptr ++;
   }
```

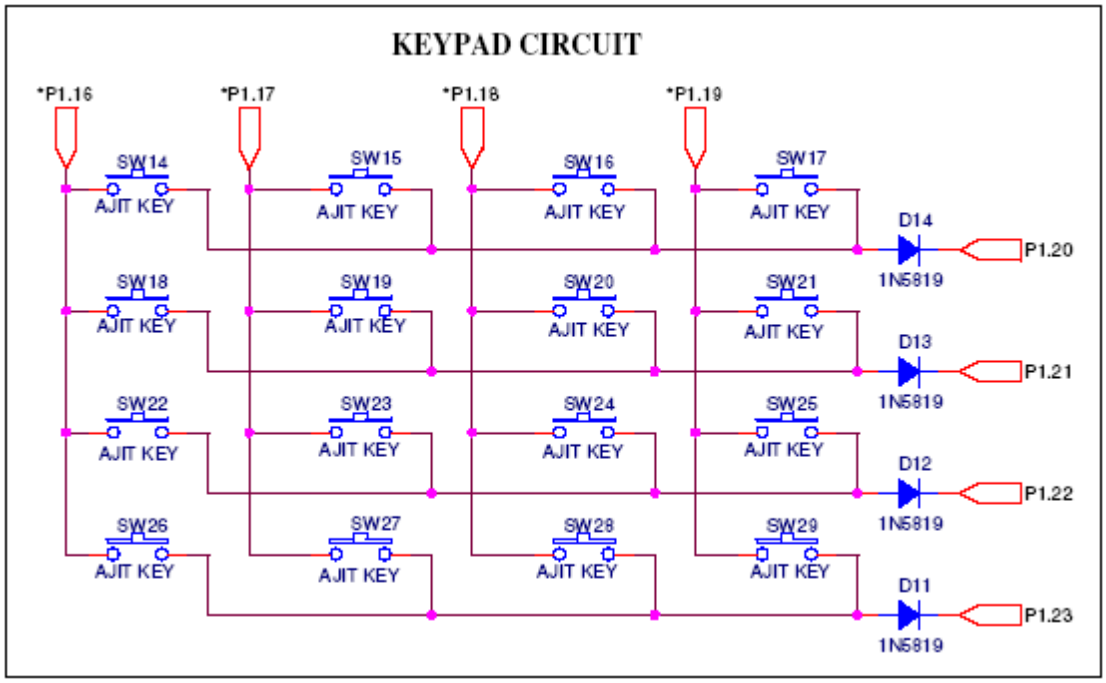
```
        ptr = disp0;
        temp1 = 0xc0;                   // Display starting address, 2nd line
        lcd_com();
        delay(800);

        while(*ptr!='\0')
   {
        temp1 = *ptr;
lcd_data();
        ptr ++;
   }
                }

void lcd_init (void)
{
// load command for lcd function setting with lcd in 4 bit mode,
// 2 line and 5x7 matrix display

        temp1 = 0x28;//Function Set: 4-bit, 2 Line, 5x7 Dots
        lcd_com();
        delay(3200);

// load a command for display on, cursor on and blinking off
        temp1 = 0x0C;          //Display on Cursor off
        lcd_com();
        delay(800);

// command for cursor increment after data dump
        temp1 = 0x06;
        lcd_com();
        delay(800);

        /*temp1 = 0x80;
        lcd_com();
        delay(800); */
}
void lcd_data(void)
{
temp = temp1 & 0xf0;
wr_dn();
temp= temp1 & 0x0f;
temp= temp << 4;
wr_dn();
delay(100);     }
```

```
void wr_dn(void)                    ////write data reg
{
        IO0CLR  = 0x000000FC;       // clear the port lines.
        IO0SET = temp;                      // Assign the value to the PORT lines
        IO0SET = 0x00000004;        // set bit  RS = 1
        IO0SET = 0x00000008;
delay(10);      // E=1, enable
        IO0CLR = 0x00000008;    //e=0, disable
}

void lcd_com(void) //send 4 bits of 2 data
{
temp = temp1 & 0xf0;//write 1st digit
wr_cn();//write 1st digit
temp = temp1 & 0x0f;
temp = temp << 4;
wr_cn();//write 2nd digit
delay(500);
}
void wr_cn(void)            //write command reg
{
        IO0CLR  = 0x000000FC;               // clear the port lines.
        IO0SET          = temp;                                 // Assign the value to the
PORT lines
        IO0SET          = 0x00000008;
delay(10);      // E=1 enable lcd
                IO0CLR  = 0x00000008; //disable lcd
}

void clr_disp(void)
{
// command to clear lcd display
   temp1 = 0x01;
lcd_com();

}
void delay(unsigned int r1)
{
        for(r=0;r<r1;r++);
}

void init_port()
{
        IO0DIR = 0x000000FC;        //configure o/p lines for lcd

}
```

6. 4 X 4 keyboard interface:

The switches SW14 to SW29 are organised as 4 rows X 4 columns matrix. One endof all the switches are connected to port lines P1.20 – P1.23, which is configured as rows.The other end of the matrix is connected to the port lines P1.16 – P1.19 which is configuredas columns.

The 4X4 matrix keypad interfacing with microcontroller ports is as below. The buttons on a keypad are arranged in rows and columns. Intersection of row and column is brought out to single button, for a total of 16 buttons on a 4X4 keypad. Pressing a button closes the switch between a column and a row, allowing current to flow between a column pin and a row pin. The key press in matrix keyboard is identified with row—column scanning technique.



Key Scan

	1	1	1	1	
1					Rows initially high
1					
1					
1					
Cols initially high					

Row 4 set to 0
After D key pressed

Row		1	0	1	1	
0	1	0	1	2	3	Rows initially high
1	1	4	5	6	7	
2	1	8	9	A	B	
3	0	C	D	E	F	
		0	1	2	3	Cols initially high

Key scan pattern: row=0111, col=1101

```
#include<lpc21xx.h>
#include<stdio.h>

/******* FUNCTION PROTOTYPE*******/

void lcd_init(void);
void clr_disp(void);
void lcd_com(void);
void lcd_data(void);
void wr_cn(void);
void wr_dn(void);
void scan(void);
void get_key(void);
void display(void);
void delay(unsigned int);
void init_port(void);

unsigned long int scan_code[16]= {0x00EE0000,0x00ED0000,0x00EB0000,0x00E70000,
                                  0x00DE0000,0x00DD0000,0x00DB0000,0x00D70000,
                                  0x00BE0000,0x00BD0000,0x00BB0000,0x00B70000,
                                  0x007E0000,0x007D0000,0x007B0000,0x00770000};


unsigned char ASCII_CODE[16]= {'0','1','2','3',
                               '4','5','6','7',
                               '8','9','A','B',
                               'C','D','E','F'};

unsigned char  row,col;

unsigned char temp,flag,i,result,temp1;
unsigned int r,r1;
unsigned long int var,var1,var2,res1,temp2,temp3,temp4;
int main()
```

```
{
        init_port();          //port intialisation
        delay(3200);                  //delay for 1ms
        lcd_init();          //lcd intialisation
        delay(3200);                  //delay
   clr_disp();                        //clear display
        delay(500);           //delay

        while(1)
        {
        get_key();
      display();
        }

}  //end of main()

void get_key(void)              //get the key from the keyboard
{
        unsigned int  i;
        flag = 0x00;
   IO1SET=0x000f0000;  //set 1s for p16 to p19 (cols i/p)
        while(1)
        {
        for(row=0X00;row<0X04;row++) //Writing one for rows, row is found
      {
        if( row == 0x03)
        {
                temp3=0x00700000;               }
        else if(row == 0x02)
        {
                temp3=0x00B00000;
                   }
                   else if(row == 0x01)
                   {
                temp3=0x00D00000;
                   }
        else if(row ==0x00)
                   {
                temp3=0x00E00000;
                   }
                        var1 = temp3;
         // each time var1 value is put to port1
                IO1SET = var1;
        IO1CLR =~var1;                  // Once again Conforming (clearing all other bits)
        scan();//find column
```

```
delay(100);                         //delay
        if(flag == 0xff)
        break;
        } // end of for
                if(flag == 0xff)
                break;
        }  // end of while

   for(i=0;i<16;i++)
   {
        if(scan_code[i] == res1)      //equate the scan_code with res1
        {
                        result =  ASCII_CODE[i];    //same position value of ascii code
                        break;                  //is assigned to result
        }
   }
}// end of get_key();

void scan(void)
{
   unsigned long int t;
   temp2 = IO1PIN;                      // status of port1, key pressed value
   temp2 = temp2 & 0x000F0000;              // Verifying column key

      if(temp2 != 0x000F0000)             // Check for Key Press or Not, store the value
in res1
      {
         flag = 0xff;
         res1 = temp2;
            res1=res1 | temp3;//final scan value is stored in res1
      }
      else
      {
         flag = 0x00;
      }
   }  // end of scan()

void lcd_init (void)
{

// load command for lcd function setting with lcd in 4 bit mode,
// 2 line and 5x7 matrix display

        temp1 = 0x28;//Function Set: 4-bit, 2 Line, 5x7 Dots
```

```
        }

cd_com();
        delay(3200);

// load a command for display on, cursor on and blinking off
        temp1 = 0x0C;          //Display on Cursor off
        lcd_com();
        delay(800);

// command for cursor increment after data dump
        temp1 = 0x06;
        lcd_com();
        delay(800);

        /*temp1 = 0x80;
        lcd_com();
        delay(800); */
}

void lcd_data(void)
{
temp = temp1 & 0xf0;
wr_dn();
temp= temp1 & 0x0f;
temp= temp << 4;
wr_dn();
delay(100);
}

void wr_dn(void)                        ////write data reg
{
        IO0CLR  = 0x000000FC;      // clear the port lines.
        IO0SET = temp;                      // Assign the value to the PORT lines
        IO0SET = 0x00000004;       // set bit  RS = 1
        IO0SET = 0x00000008;
delay(10);      // E=1, enable
        IO0CLR = 0x00000008;    //e=0, disable
}

void lcd_com(void) //send 4 bits of 2 data
{
temp = temp1 & 0xf0;//write 1st digit
wr_cn();//write 1st digit
temp = temp1 & 0x0f;
temp = temp << 4;
```

```
wr_cn();//write 2nd digit

delay(500);
}

void wr_cn(void)              //write command reg
{
        IO0CLR  = 0x000000FC;               // clear the port lines.
        IO0SET          = temp;             // Assign the value to the PORT lines
        IO0SET          = 0x00000008;
delay(10);      // E=1 enable lcd
                IO0CLR  = 0x00000008; //disable lcd
}

void clr_disp(void)
{
// command to clear lcd display
   temp1 = 0x01;
lcd_com();

}


void display(void)
{
temp1=0x80;//lcd command display  address for key value, from 1st line column 1
        lcd_com();
   temp1 = result;
   lcd_data();
}



void delay(unsigned int r1)
{
        for(r=0;r<r1;r++);
}

void init_port()
{
   IO0DIR = 0x000000FC;     //configure o/p lines for lcd
        IO1DIR = 0XFFF0FFFF; //configure i/p columns,o/p rows for key board
}
```