#GPIO PIN

```
#include"delay.h"
#include<LPC21XX.h>
main()
{
IODIR0|=1<<0;
while(1)
{
IOPIN0|=(1<<0);
delay_s(1);
IOPIN0&=~(1<<0);
delay_s(2);
}
}
```

```
#include "delay.h"
//#include <LPC214X.H>
#include<LPC21XX.h>
#include "defines.h"
main()
{
   SETBIT(IODIR0,9);
        SETBIT(IODIR0,10);
        while(1)
        {
                SETBIT(IOPIN0,9);//pin 9
                SETBIT(IOPIN0,10);
                delay_ms(500);
                CLRBIT(IOPIN0,9);//0
           CLRBIT(IOPIN0,10);       //0
                delay_ms(500);
        }
}
```

```
#include "delay.h"
#include<LPC21XX.h>
#include "defines.h"
main()
{
   SETBIT(IODIR0,0);
        SETBIT(IODIR0,1);
                SETBIT(IODIR0,2);
                        SETBIT(IODIR0,3);
        while(1)
        {          delay_ms(500);
                SETBIT(IOPIN0,0);
                delay_ms(500);
```

```
                SETBIT(IOPIN0,1);
                 delay_ms(500);
                 SETBIT(IOPIN0,2);
                delay_ms(500);
                SETBIT(IOPIN0,3);
                 delay_ms(500);
                CLRBIT(IOPIN0,0);
                delay_ms(500);
                CLRBIT(IOPIN0,1);
                delay_ms(500);
                CLRBIT(IOPIN0,2);
                delay_ms(500);
                CLRBIT(IOPIN0,3);
                delay_ms(500);
        }
}
```

```
#include "delay.h"
#include<LPC21XX.h>
#include "defines.h"

main()
{
        u8 t=255;
   WRITEBYTE(IO0DIR,0,0xff);
        while(1)
        {
                        t=~t;
                WRITEBYTE(IO0PIN,0,t);
                delay_ms(100);
        }
}
```

```
#include "defines.h"
#include"types.h"
#include<lpc21xx.h>
#define IPIN0 0 //P0.0
#define OPIN7 7 //P0.7
main()
{
        u8 t;
        //cfg p0.7 as output pin
        SETBIT(IO0DIR,OPIN7);
```

```
        while(1)
        {
                t=READBIT(IOPIN0,IPIN0);
                WRITEBIT(IOPIN0,OPIN7,t);
        }
}
```

/* main_toggle_pin_2.c */

```
#include "delay.h"
#include<LPC21XX.h>
#include "defines.h"
main()
{
   SETBIT(IODIR0,9);
        while(1)
        {
                SETBIT(IOPIN0,9);
                delay_ms(100);
                CLRBIT(IOPIN0,9);
                delay_ms(100);
        }
}
```

```
#include "delay.h"
#include<LPC21XX.h>
#include "defines.h"
main()
{
   SETBIT(IODIR0,7);
        while(1)
        {
                SSETBIT(IOSET0,7);
                CLRBIT(IOPIN0,7);
                delay_ms(100);
        }
}
```

/* main_toggle_pin_4.c */

```
#include "delay.h"
#include<LPC21XX.h>
#include "defines.h"

main()
```

```
{
        u8 t=0;
   SETBIT(IODIR0,7);
        while(1)
        {
                WRITEBIT(IOPIN0,7,t=!t);
                delay_ms(100);
        }
}
```

/*gpio*/

```
#define IPIN0 0 //P0.0
#define OPIN7 7 //P0.7
main()
{
        u8 t;
        //cfg p0.7 as output pin
        SETBIT(IODIR0,OPIN7);
        while(1)
        {
                t=READBIT(IOPIN0,IPIN0);
                WRITEBIT(IOPIN0,OPIN7,t);
        }
}
```

/* gpio_io_1.c */

```
#include "delay.h"
#include "defines.h"
#include<LPC21XX.h>

#define IPIN0 0 //P0.0
#define OPIN7 7 //P0.7
main()
{
        u8 t;
        //cfg p0.7 as output pin
        SETBIT(IODIR0,OPIN7);
        while(1)
        {
                t ? SETBIT(IOPIN0,OPIN7):CLRBIT(IOPIN0,OPIN7);
        }
}
```

/* gpio_io_2.c */

```
#include "delay.h"
#include "defines.h"
#include<LPC21XX.h>
```

```
#define IPIN 0 //P0.0
#define OPIN 7 //P0.7
main()
{
	u8 t;
	//cfg p0.7 as output pin
	SETBIT(IODIR0,OPIN);
	while(1)
	{
		READWRITEBIT2(IOPIN0,OPIN,IOPIN0,IPIN);
	}
}
```

/* gpio_io_3.c */

```
#include "delay.h"
#include "defines.h"
#include<LPC21XX.h>

#define IPIN 0  //P0.0
#define OPIN 7 //p0.7
main()
{
	u8 t;
	//cfg p0.7 as output pin
	SETBIT(IODIR0,OPIN);
	while(1)
	{
		READWRITEBIT2(IOPIN0,OPIN,IOPIN0,IPIN);
	}
}
```

/* gpio_io_p1.c*/

```
#include"types.h"
#include"defines.h"
#include<LPC21XX.h>

#define IPIN_1 0 //P.0
#define IPIN_2 1 //P0.1
#define OPIN   7 //PO.7

int main()
{
	u8 t;
	//cfg p0.7 as output pin
	SETBIT(IODIR0,OPIN);
	while(1)
	{
		t=(READBIT(IOPIN0,IPIN_1)|READBIT(IOPIN0,IPIN_2));
```

```
                WRITEBIT(IOPIN0,OPIN, t);
        }
}
```

//gpio_io_p2.c

```
#include"types.h"
#include"defines.h"
#include<LPC21XX.h>

#define IPIN_1 0
#define         IPIN_2 1
#define IPIN_3 2
#define IPIN_4 3

#define         OPIN_1 4
#define         OPIN_2 5
#define         OPIN_3 6
#define OPIN_4 7
```

////**gpio_io_p3.c**

```
int main()
{
        u8 t;
        //cfg p0.4 to p0.7 as output pin
        SETBIT(IODIR0,OPIN_1);
        SETBIT(IODIR0,OPIN_2);
        SETBIT(IODIR0,OPIN_3);
        SETBIT(IODIR0,OPIN_4);
        while(1)
        {
                READWRITEBIT(IOPIN0,OPIN_1,IPIN_1);
                READWRITEBIT(IOPIN0,OPIN_2,IPIN_2);
                READWRITEBIT(IOPIN0,OPIN_3,IPIN_3);
                READWRITEBIT(IOPIN0,OPIN_4,IPIN_4);
        }
}
```

//gpio_io_p4.c

```
#include"types.h"
#include"defines.h"
#include<LPC21XX.h>

#define IPIN_1 0
#define         IPIN_2 1
#define IPIN_3 2
#define IPIN_4 3
```

```
#define		OPIN_1 4
#define		OPIN_2 5
#define		OPIN_3 6
#define OPIN_4 7

int main()
{
	u8 t;
	//cfg p0.4 to p0.7 as outpins
	for(i=0;i<4;i++)
	SETBIT(IODIR0,OPIN_1+i);
	while(1)
	{
		for(i=0;i<4;i++)
		READWRITEBIT(IOPIN0,OPIN_1+i,IPIN_1+i);
	}
}
```

//gpio_io_p5.c

```
#include"types.h"

#include"defines.h"
#include<LPC21XX.h>

#define		IPINS_4NO 0 //p0.0
#define		OPINS_4NO 4 //po.4

int main()
{
	u8 t;
	//cfg p0.4 to p0.7 as outpins
	WRITENIBBLE(IODIR0,OPINS_4NO,15);
	while(1)
	{
		t=READNIBBLE(IOPIN0,IPINS_4NO);
		WRITENIBBLE(IOPIN0,OPINS_4NO,t);
	}
}
```

#LED

/*assuming port0.0 pin connected to active low LED & port0.7 pin connected to active high LED. write an ECP to toggle active low LED 10 times @100ms and active high LED 10times @200ms */

```
#include <LPC21XX.h>
#include "defines.h"
#include "types.h"

#define LED_AL 0 //P0.0
#define LED_AH 7 //P0.7

int main()
{
	u8 i;
	SETBIT(IODIR0,LED_AL);
	SETBIT(IODIR0,LED_AH);
	for(i=0;i<10;i++)
	{
		CPLBIT(IOPIN0,LED_AL);
		delay_ms(100);
	}
	for(i=0;i<10;i++)
	{
		CPLBIT(IOPIN0,LED_AH);
		delay_ms(200);
	}
	while(1);
}
```

/*same above code using WRITEBYTE */

```
#include <LPC21XX.h>
#include "defines.h"
#include "delay.h"
#include "types.h"

#define LED_AL 0 //P0.0
#define LED_AH 7 //P0.7

int main()
{
	u8 i,t=0;
	SETBIT(IODIR0,LED_AL);
	SETBIT(IODIR0,LED_AH);
	{
```

```
        for(i=0;i<20;i++)
        {
                WRITEBIT(IOPIN0,LED_AL, t=!t);
                delay_ms(1000);
        }
        for(i=0;i<20;i++)
        {
                WRITEBIT(IOPIN0,LED_AH, t=!t);
                delay_ms(2000);
        } while(1);
}
}
```

//sw_al_led_ah.c

```
#include<LPC21XX.h>
#include"types.h"
#include"defines.h"
                        c

#define SW_AL 0 //P0.0 connected to sw
#define LED_AH 7 //P0.7 CONNETED TO LED

int main()
{
        u8 t;
        SETBIT(IODIR0,LED_AH);
        while(1)
        {
          t=READBIT(IOPIN0,SW_AL);
          WRITEBIT(IOPIN0,LED_AH,~t);
        }
}


/*power of 2*/
#include<LPC21XX.H>
#include"delay.h"
#include"types.h"
#include"defines.h"
int main()
{
u8 n,num=2;
SETBYTE(IODIR0,0,0xff);
{
for(n=0;n<=8;n++)
{
if(n!=0)
{
num=2*num;
}
```

```
WRITEBYTE(IOPIN0,0,(num^0x0f));
delay_s(5);
}
}
while(1);
}
```

/*flash an led connected to any part line at the rate of 1se for for 10 times & stop*/

```
typedef unsigned char u8;
typedef unsigned int u32;
#include<LPC21XX.H>
#define SETBYTE(WORD,BITPOS,BTYE) (WORD|=BYTE<<BITPOS)
void delay_s(u32 delays)
{
delays*=12000000;
while(delays--);
}
#define SETBIT(WORD,BIT) (WORD|=1<<BIT)
#define CLRBIT(WORD,BIT) (WORD&=~(1<<BIT))
#define SETBYTE(WORD,BITPOS,BYTE) (WORD=|(BYTE<<BITPOS))
#define CLRBYTE(WORD,BITPOS,BYTE) (WORD&=~(BYTE<<BITPOS))
//writenibble(word,startbitpos,nibble) (word=((word&~(15<<startbitpos))|
(nibble<<startbitpos)))
#define PIN 0
main()
{
u8 i;
SETBYTE(IODIR0,PIN,255);
for(i=0;i<=31;i++)
{
SETBYTE(IOPIN0,PIN,0x0f);
delay_s(3);
CLRBYTE(IOPIN0,PIN,255);
delay_s(3);
}

while(1);

}
```

```
/*
-----------------------------------------------------------------
implement a 2 digit
up and down counter use two switeches and 8 LED's as mentioned.
sw1 for incrementing count,sw2 for decremebting count and
display a updated count on LED's.
-----------------------------------------------------------------
*/
#include<LPC21xx.h>
#include"defines.h"
#include"delay.h"
#define LED 0
#define sw1 8
#define sw2 9
int main()
{
int cnt=0;
WRITEBYTE(IODIR0,LED,0xff);
WRITEBIT(IODIR0,sw1,1);
WRITEBIT(IODIR0,sw2,1);
WRITEBYTE(IOPIN0,LED,0x0f);
while(1)
if(READBIT(IOPIN0,sw1)==0)
{
while(READBIT(IOPIN0,sw1)==0);
{
cnt++;
if(cnt<=255)
WRITEBYTE(IOPIN0,LED,(cnt^0x0f));
}
if(READBIT(IOPIN0,sw2)==0)
{
while(READBIT(IOPIN0,sw2)==0);
cnt--;
if(cnt>=0)
WRITEBYTE(IOPIN0,LED,(cnt^0x0f));
}
}
}
```

/*odd_digit_led*/

```
#include<LPC21XX.H>
#include"delay.h"
#include"types.h"
#include"defines.h"
int main()
{
```

```
u8 i,n=255;
SETBYTE(IODIR0,0,0xff);
while(1)
{
for(i=1;i<=n;i++)
{
if(i%2==1)
{
WRITEBYTE(IOPIN0,0,(i^0x0f));
delay_s(2);
}
}
}
}
```

/*second highest number in led al ah*/

```
#include<LPC21XX.H>
#include"types.h"
#include"delay.h"
#include"defines.h"
int main()

{
	u32 i,max,s1max;
	u32 arr[10]={2,70,6,10,6,8,90,3,2,2};
	max=s1max=arr[0];
			SETBYTE(IODIR0,0,0xff);
			while(1)
			{
	for(i=1;i<10;i++)
	{
		if(arr[i]>max)
		{
			s1max=max;

			max=arr[i];

		}


	}
			delay_s(2);
			WRITEBYTE(IOPIN0,0,s1max^0x0f);
}}
```

```
/*
-----------------------------------------------------------------
write an ecp to display binary equivalent of switch press count
on 8 LED's (4AH and 4AH)
-----------------------------------------------------------------
*/

#include<LPC21xx.h>
#include"defines.h"
#include"delay.h"
#define LED 0
#define sw 8
int main()
{
int cnt=0;
WRITEBYTE(IODIR0,LED,0xff);
WRITEBYTE(IOPIN0,LED,0x0f);
while(1)
{
if(READBIT(IOPIN0,sw)==0)
{
while(READBIT(IOPIN0,sw)==0);
cnt++;
WRITEBYTE(IOPIN0,LED,(cnt^0x0f));
}
}
}
```


#7SEG

```
//rand function1
#include<lpc21xx.h>
#include"types.h"
#include"defines.h"
#include"delay.h"

#define S EG7 0
#define SEL1 8
#define SEL2 9

int main()
{
u16 a[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
u8 i,m;
//initialization
```

```
        WRITEBYTE(IO0
                . DIR0,SEG7,0xff);
        SETBIT(IODIR0,SEL1);
                SETBIT(IODIR0,SEL2);

while(1)
{


        for(i=0;i<10;i++)
        {
        a[i]=rand()%10+48;

           for(m=0;m<=100;m++)
                                {
                                        WRITEBYTE(IOPIN0,SEG7,(a[i/10]));
                                        WRITEBIT(IOSET0,SEL1,1);
                                         delay_ms(10);

                                        WRITEBIT(IOCLR0,SEL1,1);

                                        WRITEBYTE(IOPIN0,SEG7,(a[i%10]));
                                        WRITEBIT(IOSET0,SEL2,1);
                                        delay_ms(10);
                                        WRITEBIT(IOCLR0,SEL2,1);
                                }
                        }
           delay_s(1);
           }
}
```

//rand function2

```
 #include<lpc21xx.h>
#include"types.h"
#include"defines.h"
#include"delay.h"
#define SEG 0
#define SEL1 8
#define SEL2 9
int main()
{
        u16 a[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90},i,m;
//initialization


        WRITEBYTE(IODIR0,SEG,0xff);
        SETBIT(IODIR0,SEL1);
        SETBIT(IODIR0,SEL2);
```

```
while(1)
{

//        srand(getpid());
          for(i=0;i<=10;i++)
          {
                    a[i]=rand()%6+10;
                              for(m=0;m<=100;m++)
                              {
          WRITEBYTE(IOPIN0,SEG,a[i/10]);
          WRITEBIT(IOSET0,SEL1,1);
          delay_ms(1);
          WRITEBIT(IOCLR0,SEL1,1);

          WRITEBYTE(IOPIN0,SEG,a[i%10]);
       WRITEBIT(IOSET0,SEL2,1);
       delay_ms(1);
       WRITEBIT(IOCLR0,SEL2,1);
                    delay_s(5);
          }
}
}
}
```

//rand function3

```
#include<lpc21xx.h>
#include"types.h"
#include"defines.h"
#include"delay.h"

#define SEG7 0
#define SEL1 8
#define SEL2 9
#define SW 10
int main()
{
u16 a[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
u8 i;
//initialization


       WRITEBYTE(IODIR0,SEG7,0xff);
       SETBIT(IODIR0,SEL1);

while(1)
{


       for(i=0;i<10;i++)
       {
```

```
        a[i]=rand()%9;
           if(READBIT(IOPIN0,SW)==0)
           {
           while(READBIT(IOPIN0,SW)==0);
        WRITEBYTE(IOPIN0,SEG7,a[i]);
        WRITEBIT(IOSET0,SEL1,1);
        delay_ms(1000);
        WRITEBIT(IOCLR0,SEL1,1);
           }
           //delay_s(1);
           }
}
}
```

//Isprime

```
#include<lpc21xx.h>
#include"delay.h"
#include"types.h"
#include"defines.h"
#include<math.h>
#define SEG7 0
#define SEL1 8
#define SEL2 9
u8 a[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
u8 i;
int Isprime(u32 num)
{
u8 i;
          for(i=2;i<=sqrt(num);i++)
                    if(num%i==0)
                    {
                    return 0;
                    }
                    return 1;

}

int main()
{

//initialize
u8 n,m;
WRITEBYTE(IODIR0,SEG7,0xff);
SETBIT(IODIR0,SEL1);
SETBIT(IODIR0,SEL2);
while(1)
{
                    for(n=2;n<=99;n++)
                    {
```

```
                        if(Isprime(n)==1)
                        {
                                for(m=0;m<=100;m++)
                                {
                                        WRITEBYTE(IOPIN0,SEG7,(a[n/10]));
                                        WRITEBIT(IOSET0,SEL1,1);
                                         delay_ms(1);
                                        WRITEBIT(IOCLR0,SEL1,1);

                                        WRITEBYTE(IOPIN0,SEG7,(a[n%10]));
                                        WRITEBIT(IOSET0,SEL2,1);
                                        delay_ms(1);
                                        WRITEBIT(IOCLR0,SEL2,1);
                                }
                        }
                        delay_ms(1000);
                }
}
}
```

//.fabonacci

```
#include<lpc21xx.h>
#include"delay.h"
#include"types.h"
#include"defines.h"
#define SEG7 0
#define DSEL1 8
#define DSEL2 9
u32 a[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
int main()
{
unsigned int x,y,z;
unsigned int i,n=12;
WRITEBYTE(IODIR0,SEG7,0Xff);
WRITEBIT(IODIR0,DSEL1,1);
WRITEBIT(IODIR0,DSEL2,1);
while(1)
{
x=0;
y=1;
WRITEBIT(IOPIN0,DSEL1,1);
WRITEBYTE(IOPIN0,SEG7,a[x]);
delay_ms(1000);
WRITEBYTE(IOPIN0,SEG7,a[y]);

for(i=1;i<=n;i++)
{
z=x+y;
x=y;
```

```
y=z;

WRITEBYTE(IOPIN0,SEG7,a[z/10]);
WRITEBIT(IOSET0,DSEL1,1);
delay_s(1);
WRITEBIT(IOCLR0,DSEL1,1);
WRITEBYTE(IOPIN0,SEG7,a[z%10]);
WRITEBIT(IOSET0,DSEL2,1);
delay_s(1);
WRITEBIT(IOCLR0,DSEL2,1);
}
}
}
```

/*test2_7seg.c*/

```
#include<lpc21xx.h>
#include"delay.h"
#include"types.h"
#include"defines.h"
#define DSEL1 8
#define DSEL2 9
#define LED7SEG 0
#define dp 0x7f
int main()
{
u8 i;
u16 j;
u8 a[14]={0xcf,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xc8,0x80,0x90};
WRITEBYTE(IODIR0,LED7SEG,0xff);
SETBIT(IODIR0,DSEL1);
SETBIT(IODIR0,DSEL2);
while(1)
{
for(i=0;i<=10;i++)
{
SETBIT(IOPIN0,DSEL1);
WRITEBYTE(IOPIN0,LED7SEG,a[i/10]&dp);
CLRBIT(IOPIN0,DSEL1);
delay_s(1);
}
for(j=0;j<=10;j++)
{
SETBIT(IOPIN0,DSEL1);
WRITEBYTE(IOPIN0,LED7SEG,a[i%10]);
CLRBIT(IOPIN0,DSEL1);
delay_s(1);
}
}}
```

//seg.c

```
#include <LPC21xx.h>
#include"defines.h"
#include"delay.h"
#include"seg.h"
#include"types.h"
u8
seglut[14]={0xc0,0xf9,0xa4,0xb0,0x88,0x99,0x92,0x82,0x83,0xc6,0xf8,0x80,0x90,
0xa1};
#define CA7SEG_2_MUX  0
#define DSEL1        8
#define DSEL2        9

void Init_CA7SEG_2_MUX(void)
{
WRITEBYTE(IODIR0,CA7SEG_2_MUX,0XFF);
SETBIT(IODIR0,DSEL1);
SETBIT(IODIR0,DSEL2);
}

void dispi_2_mux_ca7seg(u8 i)
{
WRITEBYTE(IOPIN0,CA7SEG_2_MUX,seglut[i/10]);
SSETBIT(IOSET0,DSEL1);
delay_ms(1);
SCLRBIT(IOCLR0,DSEL1);

WRITEBYTE(IOPIN0,CA7SEG_2_MUX,seglut[i%10]);
SSETBIT(IOSET0,DSEL2);
delay_ms(1);
SCLRBIT(IOCLR0,DSEL2);
}
```

//check7seg.c

```
#include"LPC21XX.h"
void delay()
{
        unsigned int i,j;
        for(i<0;i<2000;i++)
                for(j<0;j<2000;j++);
}
int main()
{
        IODIR0=0XFFFFFFFF;
        while(1)
        {
                        IOSET0=0XFFFFFFFF;
                        IOCLR0=0XFFFFFFFF;
                        delay();
```

```
                    IOSET0=0XFFFFFFFF;
                    IOCLR0=0XFFFFFFFF;
                    delay();
        }
}
```

/*7seg Even Odd*/

```
#include<LPC21XX.h>
#include"delay.h"
#include"defines.h"
#include"types.h"
#define LED 0
#define DSEL1 8
#define DSEL2 9
u32 a[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
int main()
{
u8 i;
WRITEBYTE(IODIR0,LED,0xff);
SETBIT(IODIR0,DSEL1);
SETBIT(IODIR0,DSEL2);
while(1)
{
WRITEBYTE(IOPIN0,LED,0xff);
for(i=1;i<10;i++)
{
if(i%2==0)
{
SETBIT(IOPIN0,DSEL1);
WRITEBYTE(IOPIN0,LED,a[i]);
delay_s(1);
CLRBIT(IOPIN0,DSEL1);
}
else if(i%2==1)
{

 SETBIT(IOPIN0,DSEL2);
WRITEBYTE(IOPIN0,LED,a[i]);
delay_s(1);
CLRBIT(IOPIN0,DSEL2);
}
}
}
}
```

RAHUL SHRIDHAR BANSOD

/*00-99*/

```
#include<LPC21XX.h>
#include"delay.h"
#include"defines.h"
#include"types.h"
#define LED 0
#define DSEL1 8
#define DSEL2 9
#define dp 0x7f
u32 a[10]={0xcf,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
int main()
{
u8 i;
WRITEBYTE(IODIR0,LED,0xff);
SETBIT(IODIR0,DSEL1);
SETBIT(IODIR0,DSEL2);
while(1)
{
for(i=0;i<=99;i++)
{

WRITEBYTE(IOPIN0,LED,0xff);
{
SETBIT(IOPIN0,DSEL1);
WRITEBYTE(IOPIN0,LED,a[i/10]);
delay_s(1);
for(i=0;i<=9;i++)
{
SETBIT(IOPIN0,DSEL2);
WRITEBYTE(IOPIN0,LED,a[i%10]);
delay_s(1);
CLRBIT(IOPIN0,DSEL2);
}
}
}        }
}
```

/*
5. PART2: Implement up and down counter. Use two AL switches and 8 leds (4-active
high leds and 4-active low leds) as mentioned:
As long as sw1 is pressed increment the count value with respect to 1 sec
As long as sw2 is pressed decrement count value with respect to 1 sec
And display updated count on leds.
If both switches are pressed at a time, don't do any operation on count.
Note: if count value is 0, at this time if sw2 pressed it should display 0 only on leds and if
count value is 255 then if sw1 pressed it should display 255 only on leds.
***/**

```
#include<lpc21xx.h>
#include"types.h"
#include"delay.h"
#include"defines.h"
#define LED 0
#define SW1 8
#define SW2 9

int main()
{
s8 cnt=0;
WRITEBYTE(IODIR0,LED,255);
WRITEBYTE(IOPIN0,LED,0x0f);
while(1)
{
if(cnt!=-1 && cnt!=256)
{
if(READBIT(IOPIN0,SW1)==1)
{
delay_s(1);
cnt++;

WRITEBYTE(IOPIN0,LED,(0x0f^cnt));
}
if(READBIT(IOPIN0,SW2)==0)
{
while(READBIT(IOPIN0,SW2)==0);
delay_s(1);
cnt--;


WRITEBYTE(IOPIN0,LED,(0x0f^cnt));
}
if(READBIT(IOPIN0,SW1)==0 && READBIT(IOPIN0,SW2)==0)
{
```

```
delay_s(1);
//WRITEBYTE(IOPIN0,LED,0xf0^cnt);
}
}
}
}
```

#keypad

//kaypad.c

```
#include"types.h"
#include"defines.h"
#include"keypad.h"
#include <LPC21xx.H>
#define ROW0 16

#define ROW1 17

#define ROW2 18

#define ROW3 19

#define COL0 20

#define COL1 21

#define COL2 22

#define COL3 23


u8 keypadLUT[4][4]=

{

{1,2,3,4},{5,6,7,8},{9,0,'*','#'},{'!','@','$','&'}

};


void InitRows(void)

{

WRITENIBBLE(IODIR1,ROW0,0xf);
```

```
}

void InitCols(void)

{

//WRITENIBBLE(IODIR1,ROW0,0xf);

}

u8 colscan()

{

u8 t;

t=(READNIBBLE(IOPIN1,COL0));

t=(t<15)?0:1;

return t;

}

u8 keyscan(void)

{

u8 rNo,cNo,keyval=0;

while(colscan())

{

keyval++;

}

WRITENIBBLE(IOPIN1,ROW0,0xE);

if(!colscan())

{

rNo=0;
```

```
goto colcheck;
}

WRITENIBBLE(IOPIN1,ROW0,0xD);
if(!colscan())
{
rNo=1;
goto colcheck;
}

WRITENIBBLE(IOPIN1,ROW0,0xB);
if(!colscan())
{
rNo=2;
goto colcheck;
}

WRITENIBBLE(IOPIN1,ROW0,7);
if(!colscan())
{
rNo=3;
goto colcheck;
}

colcheck:
WRITENIBBLE(IOPIN1,ROW0,0);
if(READBIT(IOPIN1,COL0)==0)          cNo=0;
```

```
else if(READBIT(IOPIN1,COL1)==0)      cNo=1;

else if(READBIT(IOPIN1,COL2)==0)      cNo=2;

else if(READBIT(IOPIN1,COL3)==0)      cNo=3;


keyval=keypadLUT[rNo][cNo];

return keyval;

}
```

//keypad.h

```
#include"types.h"
void InitRows(void);
void InitCols(void);
u8 colscan(void);
u8 keyscan(void);
```

//sms_keypad.c

```
#include"types.h"
#include"defines.h"
#include<LPC21xx.h>
#include"kpm_defines.h"
const u8 kpm_LUT[4][4][4]=
{
        {'7','8','9','%','4','5','6','*','1','2','3','-','c','0','=','+',},
        {'a','d','g',' ','j','m','p',' ','s','v','y',' ',' ',' ',' ',' ',' ',},
        {'b','e','h',' ','k','n','q',' ','t','w','z',' ',' ',' ',' ',' ',' ',},
        {'c','f','i',' ',' ','l','o','r',' ',' ','u','x',' ',' ',' ',' ',' ',' ',' ',' ',}

};
const u8 Scancode[4]={14,13,11,7};
s32 presskey,khcount;
void InitKpm(void)
{
WRITENIBBLE(IODIR1,Row0,15);
}
u8 ColScan(void)
{
u8 t;
t=(READNIBBLE(IOPIN1,Col0)<15)?0:1;
return t;
}
u8 Colcheck()
{
```

```
u8 i,cNo;
for(i=0;i<4;i++)
{
if(READNIBBLE(IOPIN1,Col0==Scancode[i]))
{
        cNo=i;
        break;
}
}
 return cNo;
 }
 u8 Rowcheck()
 {
 u8 i,rNo;
 for(i=0;i<4;i++)
 {
 WRITENIBBLE(IOPIN1,Row0,Scancode[i]);
 if(!ColScan())
 {
 rNo=i;
 break;
 }
 }
 WRITENIBBLE(IOPIN1,Row0,0);
 return rNo;
 }
 u8 KeyScan()
 {
 u8 row=0,col=0;
 static u8 prow=0,pcol=0;
 row=Rowcheck();
 col=Colcheck();
 if(row!=prow)
 {
 khcount=0;
 prow=row;
 }
 if(col!=pcol)
 {
 khcount=0;
 pcol=col;
 }
  return kpm_LUT[khcount][row][col];
 }
```

//sms_keypad.h

```
#ifndef __SMS_KEYPAD_H__
#define __SMS_KEYPAD_H__

u8 ColScan(void);
u8 InitKpm(void);
u8 RowCheck(void);
u8 ColCheck(void);
u8 KeyScan(void);
#endif
```

/*
main.c
interfacing keypad and LCD
***/**

```
#include"types.h"
#include"keypad.h"
u8 key __attribute__((at(0x40000101)));
main()
{
InitRows();
InitCols();
InitLCD();
while(1)
{
key=keyscan();
U32LCD(key);

while(!colscan());
}
}
```

//main_sms_kpm_lcd.c

```
#include<LPC21XX.h>
#include"types.h"
#include"defines.h"
#include"lcd.h"
#include"kpm_defines.h"
#include"sms_keypad.h"
#include"lcd_defines.h"
#include"delay.h"
#define LED 19//p0.19
extern s32 presskey,khcount;
main()
{
u32 dly_ms=250;
```

```
s32 pos=-1;
InitLCD();
InitKpm();
StrLCD("sms keypad:");
while(1)
{
khcount=-1;
while(ColScan());
for(dly_ms=12000*250;dly_ms>0;dly_ms--)
{
WRITENIBBLE(IOPIN1,Row0,0);
if(!ColScan())
 {
 if(khcount==3)
 khcount=-1;
 else if (khcount==-1)
 {
 pos++;
 if(pos>15)
 {
 CmdLCD(GOTO_LINE2_POS0);
 StrLCD("                ");
 CmdLCD(0XC0);
 pos=0;
 }
 }
 khcount++;
 presskey=KeyScan();
 CmdLCD((0xC)+pos);
 CharLCD(presskey);
 while(!ColScan())
 dly_ms=12000*250;
 }
 }
 }
 }
```

RAHUL SHRIDHAR BANSOD

#LCD

//lcd.h

```
#include"types.h"
void WriteLCD(u8 Dat);
void CmdLCD(u8 Cmd);
void InitLCD(void);
void CharLCD(u8 C);
void StrLCD(u8 *s);
void U32LCD(u32 n);
void S32LCD(s32 n);
void F32LCD(f32 n,u8 nDp);
void BuildCGRAM(u8 *p,u8 nBytes);
//void BuildCGRAM(u8 *p,u8 nBytes);
void HexLCD(u32);
//void BinLCD(u32 n,u8 nBd);

void hexLCD (u32 n,u8 choice);
void OctLCD(u32 num);
```

//lcd_defines.h

```
#define LCD_DATA 8
#define LCD_RS 16
#define LCD_RW 18
#define LCD_EN 17

#define CLRSCR 0X01
#define DISP_ON_CUR_OFF 0X0C
#define DISP_ON_CUR_ON 0X0e
#define DISP_ON_CUR_BLK 0X0f

#define GOTO_LINE1_POS0 0X80
#define GOTO_LINE2_POS0 0XC0
#define SHIFT_CUR_RIGHT 0X06
#define GOTO_CGRAM      0X40
```

//lcd.c

```
#include "types.h"
#include "lcd_defines.h"
#include "defines.h"
```

```
#include<LPC21xx.h>
#include "delay.h"

void WriteLCD(u8 Dat)
{
	SCLRBIT(IOCLR0,LCD_RW);
	WRITEBYTE(IOPIN0,LCD_DATA,Dat);
	SSETBIT(IOSET0,LCD_EN);
	delay_us(1);
	SCLRBIT(IOCLR0,LCD_EN);
	delay_ms(2);
}
void CmdLCD(u8 cmd)
{
	SCLRBIT(IOCLR0,LCD_RS);
	WriteLCD(cmd);
}

/*
void WriteLCD(u8 dat)


{


	CLRBIT(IOPIN0,LCD_RW); //cfg for writing to LCD


	WRITEBYTE(IOPIN0,LCD_DATA,Dat);


	SETBIT(IOPIN0,LCD_EN); //high to...


	delay_us(1);


	CLRBIT(IOPIN0,LCD_EN); //low pulse


	delay_ms(2);  //atleast 2ms between consecutive writes


		//for syn bt/w cpu & LCD


}

*/
```

```
void InitLCD(void)
{
	delay_ms(15);
	WRITEBYTE(IODIR0,LCD_DATA,0xff);
	SETBIT(IODIR0,LCD_RS);
	SETBIT(IODIR0,LCD_RW);
	SETBIT(IODIR0,LCD_EN);

	CmdLCD(0x30);
	delay_ms(100);
	CmdLCD(0x30);
	delay_ms(100);
	CmdLCD(0x30);
	delay_ms(100);
	CmdLCD(0x38);
	CmdLCD(0x08);
	CmdLCD(0x01);
	CmdLCD(0x06);
	CmdLCD(0x0e);//0x0E,0x0F
	CmdLCD(GOTO_LINE1_POS0);
	CmdLCD(DISP_ON_CUR_BLK);
	CmdLCD(CLRSCR);
	CmdLCD(SHIFT//lcd.c_CUR_RIGHT);
}

 void CharLCD(u8 asciiV)
{
	SSETBIT(IOSET0,LCD_RS);
	WriteLCD(asciiV);
}

void StrLCD(s8 *s)
{

    while(*s)
	CharLCD(*s++);


}

void U32LCD(u32 n)
{
	s8 i=0;
	u8 a[10];
	SSETBIT(IOSET0,LCD_RS);
	if(n==0)
	{
		WriteLCD('0');
```

```
	}
	else
	{
		while(n)
		{
			a[i++]=(n%10)+48;
			n/=10;
		}
		for(--i;i>=0;i--)
		{
			WriteLCD(a[i]);
		}
	}
}

void S32LCD(s32 n)
{
	u32 t;
	SSETBIT(IOSET0,LCD_RS);
	if(n<0)
	{
		WriteLCD('-');
		t=-n;
	}
	U32LCD(t);
}

void F32LCD(f32 f,u8 nDP)
{
	u32 ipart;
	u8 i;
	if(f<0.0)
	{
		CharLCD('-');
		f=-f;
	}
	ipart=f;
	U32LCD(ipart);
	CharLCD('.');
	for(i=0;i<nDP;i++)
	{
		f=(f-ipart)*10;
		ipart=f;
		CharLCD(ipart+48);
	}
}
void Rotate_Array(u8 *p,u8 dir)
{
s8 i,temp;
u8 a[17];
```

```
if(dir=='r')
{
temp=a[15];
for(i=15;i>0;i--)
{
a[i]=a[i-1];
}
a[0]=temp;
}
else if(dir=='l')
{
temp=a[0];
for(i=0;i<15;i++)
{
a[i]=a[i+1];
}
a[15]=temp;
}
}
void BuildCGRAM(u8 *p,u8 nBytes)
{
u8 i;
//point to cgram
CmdLCD(GOTO_CGRAM);
for(i=0;i<nBytes;i++)
{
//write into cgram
CharLCD(p[i]);
}
//relocate tp DDRAM
//CmdLCD(GOTO_LINE1_POS0);
CmdLCD(GOTO_LINE1_POS0);
}
void HexLCD(u32 n)
{
u8 a[8]={'0'};
s8 i=0,t;
while(n)
{
t=(n%16);
a  [i++]=(t>9)?((t-10)+'A'):(t+48);
n/=10;
}
for(--i;i>=0;i--)
CharLCD(a[i]);
}
void hexLCD (u32 n,u8 choice)
{
u8 a[8]={'0'};
s8 i=0,t;
```

```
if(choice=='b')
{
while(n)
{
t=(n%16);
a[i++]=t>9?(t>10)+'A':(t+48);
n=16;
}
}
}
void OctLCD(u32 num)
{
	u32 octnum[100];
	u32 i=0;
		s32 j;
	while(num!=0)
	{
		octnum[i]=num%8;
		num=num/8;
		i++;
	}
	for(j=i-1;j>=0;j--)
	{
		U32LCD(octnum[j]);
	}

}
void BinLCD(u32 n,u8 nBD)
{
s8 i;
for(i=(nBD-1);i>=0;i--)
{
CharLCD(((n>>i)&1)+48);
}
}
```

/*
1. Write an ECP to develop the driver for 16*2 alphanumeric LCD.
a) To display a character
b) To display a string
c) To display an integer
d) To display float number up to three decimal places.
e) To display any custom character.
***/**

//main.c

```
#include "types.h"
#include "lcd_defines.h"
#include "defines.h"
#include<LPC21xx.h>
#include "delay.h"
#include "lcd.h"

u8
a[16]={0X1F,0X04,0X02,0X1F,0X06,0X04,0X02,0X00,0x00,0x11,0x0a,0x1f,0x0a,
0x11,0x00,0x00};
int main()
{
        InitLCD();
        CharLCD('A');
        delay_s(1);
        CmdLCD(CLRSCR);
        F32LCD(-12345.6789,3);
        delay_s(1);
        CmdLCD(CLRSCR);
        U32LCD(12345);
        delay_s(1);
        CmdLCD(CLRSCR);
        S32LCD(-12345);
        delay_s(1);
        CmdLCD(CLRSCR);
   StrLCD("CUTE CACTUS");
        delay_s(1);
        CmdLCD(CLRSCR);
        BuildCGRAM(a,16);//swastik character
        CharLCD(0);
        CharLCD(1);
        delay_s(1);
        hexLCD('A',8);
        while(1);
}
```

/*
4. Write an ECP display the basic time (HH:MM:SS) on LCD.
Note: don't use RTC registers
***/**

```
#include "types.h"
#include "lcd_defines.h"
#include "defines.h"
#include<LPC21xx.h>
#include "delay.h"
#include "lcd.h"

int main()
{
u8 H,M,S;
        InitLCD();
        while(1)
        {
        CmdLCD(0x01);
  for(H=0;H<=24;H++)
  {


  for(M=0;M<=60;M++)
  {

  for(S=0;S<=60;S++)
         {
         CmdLCD(0xC0);
         StrLCD("HH:MM:SS");
         CmdLCD(0x80);

         U32LCD(H);
         CmdLCD(0x82);
         CharLCD(':');
         CmdLCD(0x83);
         U32LCD(M);
         CmdLCD(0x85);
         CharLCD(':');
         CmdLCD(0x86);
         U32LCD(S);
         CmdLCD(0x88);
        delay_s(1);

         }
        }
        }
        }
        }
```

/*
9. Write a program to display the message “VECTOR” on the first Line and “Institute”
on the second line of a 2x16 LCD. Then make “Institute” scroll from right to left on
second line of LCD screen.

***/**

```
#include<lpc21xx.h>
#include"delay.h"
#include"defines.h"
#include"types.h"
#include"lcd.h"
#include"lcd_defines.h"
int main()
{
u8 i,j;
InitLCD();
while(1)
{

for(i=0,j=17;i<=17,j>0;j--,i++)
{
CmdLCD(0x80+j);
StrLCD("VECTOR");
CmdLCD(0xC0+i);
StrLCD("INSTITUTE");
delay_ms(500);
CmdLCD(0x01);
}
for(i=0,j=17;i<=17,j>0;j--,i++)
{
CmdLCD(0x80+i);
StrLCD("VECTOR");
CmdLCD(0xC0+j);
StrLCD("INSTITUTE");
delay_ms(500);
CmdLCD(0x01);
}
}
}
```

/*

3. Write an ECP to rotate a string on LCD (From right to left)

***/**

```
#include<lpc21xx.h>
#include"delay.h"
#include"defines.h"
#include"types.h"
#include"lcd.h"
#include"lcd_defines.h"

int main()
{
u8 i;
InitLCD();
while(1)
{
for(i=0;i<17;i++)
{
CmdLCD(0x80+i);
StrLCD("VECTOR INSTITUTE");
delay_ms(500);
CmdLCD(0x01);
}

/*delay_s(1);
for(i=17;i>0;i--)
{
CmdLCD(0x80+i);
StrLCD("VECTOR INSTITUTE");
CmdLCD(0x01);
}
*/
}
}
```

/*

6. Write an ECP to take 20 numbers randomly in an array and find the prime numbers in
the list of numbers to display on LCD.
Note: must use rand() function
***/**

```
#include<lpc21xx.h>
#include<math.h>
#include"delay.h"
#include"defines.h"
```

```
#include"types.h"
#include"lcd.h"
#include"lcd_defines.h"

 int Isprime(u32 num)
{
u8 i;
        for(i=2;i<=sqrt(num);i++)
                if(num%i==0)

                return 0;

                return 1;

}

int main()
{
while(1)
{
u8 j,a[30];
InitLCD();
a[j]=rand()%100;
for(j=0;j<=30;j++)
{
        if(Isprime(a[j]==1))
        {
        U32LCD(a[j]);
        delay_s(1);
        CmdLCD(0x01);
        }
  }
}}
```

/*

2. Write a program to display the message “VECTOR” on the first Line and “Institute”
on the second line of a 2x16 LCD. Then make “Institute” flash at the rate of 1sec for 5
times, then clear the LCD screen

***/**

```
#include<lpc21xx.h>
#include"delay.h"
#include"defines.h"
#include"types.h"
#include"lcd.h"
#include"lcd_defines.h"
```

```
int main()
{
u8 i;
InitLCD();
while(1)
{
CmdLCD(0x80);
StrLCD("VECTOR");
for(i=0;i<=5;i++)
{
delay_ms(5);
StrLCD("INSTITUTE");
CmdLCD(0x01);
}
CmdLCD(0x01);
}
}
```

/*

Write an ECP to take ten 3-digit numbers randomly in an array and find palindrome
numbers in the list of numbers to display on LCD.
Note: must use rand() function

***/**

```
#include<lpc21xx.h>
#include"delay.h"
#include"defines.h"
#include"types.h"
#include"lcd.h"

int main()
{
        int i,min=1,max=999,rev=0,rem,temp;
        InitLCD();
        while(1)
        {
        for(i=min;i<=max;i++)
        {
                temp=i;
                rev=0;
                while(temp)
                {
                        rem=temp%10;
                        rev=rev*10+rem;
                        temp=temp/10;
                }
```

```
                if(i==rev)
                {
                        CmdLCD(0xC0);
                        U32LCD(i);
                        delay_ms(200);
                        CmdLCD(0x01);
                }
                CmdLCD(0x80);
                StrLCD("palindrome interger");

        }
        }
}
```

//palindrome

```
#include<lpc21xx.h>
#include"delay.h"
#include"defines.h"
#include"types.h"
#include"lcd.h"
#include<string.h>
#include<stdlib.h>
int main()
{
InitLCD();
while(1)
{
      char str[100]="malayalam";

      int i,j,k;
                for(k=0;str[k];k++)
                {
                CmdLCD(0x80+k);
                CharLCD(str[k]);
                }
      if(str[strlen(str)-1]==10)
      {
            str[strlen(str)-1]=0;
      }
      for(i=0,j=strlen(str)-1;i<=j;i++,j--)
            if(str[i]!=str[j])
                  break;
      if(i>j)
                  {
                  CmdLCD(0xC0);
            StrLCD("palindrome");
                  }
                   else
                                    {
                                    CmdLCD(0xC0);
```

```
StrLCD("not palindrom");
}
```

```
}
}
```

/*

10. Write an ECP to convert first character of each word in to upper case and display the
final result on LCD.
Note: Connect two switches. Based on first switch press, take “vector india” as input and
do the operation; Based on second switch press, take “vector Hyderabad” as input and do
the operation. In both cases print input and output on lcd screen.
Embedded-C AssignmentsOutput:
When Switch1 is pressed, display “vector india” on the 1 st line of LCD and when the
switch1 is released display “Vector India” on the 2 nd line of LCD.
When Switch2 is pressed, display “vector hyderabad” on the 1 st line of LCD and when
the switch2 is released display “Vector Hyderabad” on the 2 nd line of LCD.
If none of the switch is pressed, display “waiting for input” on the 1 st line of LCD.

***/**

```
#include "types.h"
#include "lcd_defines.h"
#include "defines.h"
#include<LPC21xx.h>
#include "delay.h"
#include "lcd.h"
#include<string.h>
#include<stdlib.h>

#define SW1 0
#define SW2 1

int main()
{
InitLCD();
while(1)
{
u8 str[30],i;
```

```
str[30]="rahul bansod";
if(READBIT(IOPIN0,0)==0)
{
while(READBIT(IOPIN0,0)==0);
CmdLCD(0x80);
StrLCD("rahul bansod");
}
delay_ms(20);
if(READBIT(IOPIN0,1)==0)
{
while(READBIT(IOPIN0,1)==0);
CmdLCD(0xC0);
StrLCD("waiting for output...");
 if(str[0]>=97&&str[0]<=122)
        {
                str[i]-=32;
                i++;
        }
        while(str[i])
        {
                if(str[i]==32)
                {
                        str[i+1]-=32;

                        if(islower(str[i+1]!=0))
                        {
                                str[i+1]-=32;
                        }
                }

                  }
                  CmdLCD(0xC0);
                  StrLCD(str);
}
}
}
```

/*
11. Write an ECP to display the ASCII table information for A-Z, a-z & 0-9 on LCD
screen with respect to 5 seconds.
For example,
1 st line: A D H O
2 nd line: A 65 41 101
1 st line: A D H O
2 nd line: B 66
After 5 secs
42 102 and so on ....

***/**

```
#include "types.h"
#include "lcd_defines.h"
#include "defines.h"
#include<LPC21xx.h>
#include "delay.h"
#include "lcd.h"

int main()
{
u8 i;

InitLCD();
CmdLCD(0x80);
StrLCD("A  D  H    O");
/*CmdLCD(0x80);
CharLCD('A');
delay_ms(200);
//InitLCD();
CmdLCD(0x82);
CmdLCD('A');
delay_ms(200);
//InitLCD();
CharLCD(0x85);
CmdLCD('A');
delay_ms(200);
//InitLCD();
CmdLCD(0x80+11);
CharLCD('O');
  */
while(1)
{
StrLCD("A  D  H    O");
for(i=48;i<=57;i++)
{
CmdLCD(0xC0);
CharLCD(i);
CmdLCD(0xC2);
U32LCD(i);
CmdLCD(0xC6);
HexLCD(i);
CmdLCD(0xC0+11);
OctLCD(i);
delay_s(1);
//CmdLCD(0x01);
}
for(i=65;i<=90;i++)
{
CmdLCD(0xC0);
CharLCD(i);
```

```
CmdLCD(0xC2);
U32LCD(i);
CmdLCD(0xC6);
HexLCD(i);
CmdLCD(0xC0+11);
OctLCD(i);
delay_s(1);
//CmdLCD(0x01);
}
for(i=97;i<=122;i++)
{
CmdLCD(0xC0);
CharLCD(i);
CmdLCD(0xC2);
U32LCD(i);
CmdLCD(0xC6);
HexLCD(i);
CmdLCD(0xC0+11);
OctLCD(i);
delay_s(1);
//CmdLCD(0x01);
}
delay_s(1);
CmdLCD(0x01);
}
}
```


/*
13. Write an ECP to convert first and last character of each word in to upper case and display the final result on LCD.
Note: Connect two switches. Based on first switch press, take "vector india" as input and do the operation; Based on second switch press, take "vector Hyderabad" as
input and do the operation. In both cases print input and output on lcd screen.
Output:
When Switch1 is pressed, display "vector india" on the 1st line of LCD and when the switch1 is released display "VectoR IndiA" on the 2nd line of LCD.
When Switch2 is pressed, display "vector hyderabad" on the 1st line of LCD and when the switch2 is released display "VectoR HyderabaD" on the 2nd line of LCD.
If none of the switch is pressed, display "waiting for input" on the 1st line of LCD.
***/**

```
#include "types.h"
#include "lcd_defines.h"
#include "defines.h"
#include<LPC21xx.h>
```

```
#include "delay.h"
#include "lcd.h"
#include<string.h>
#include<stdlib.h>

#define SW1 0
#define SW2 1

int main()
{
u8 i=0,cnt;
char str1[30]="vector india";
char str2[30]="vector hyderabad";
InitLCD();
while(1)
{
if(READBIT(IOPIN0,SW1)==0)

{
while(READBIT(IOPIN0,SW1)==0);
cnt=0;
for(i=0;i<=strlen(str1)-1;i++)
{
CmdLCD(0x80+cnt);
cnt++;
CharLCD(str1[i]);
}
//CmdLCD(0x01);
i=0;
if(str1[0]>=97&&str1[0]<=122)
	{
		str1[0]-=32;
		i++;
	}
	while(str1[i])
	{
		if(str1[i]==32)
		{
			str1[i+1]-=32;
			str1[i-1]-=32;
		}

                                        i++;

		 }
		 if(str1[i]==0)
                                        {
			str1[i-1]-=32;
		}
		 //CmdLCD(0xC0);
```

```
                    //CmdLCD(CLRSCR);
                    cnt=0;
                    for(i=0;i<=strlen(str1)-1;i++)
                    {

                    CmdLCD(0xC0+cnt);
                    cnt++;
                    CharLCD(str1[i]);
                    //CmdLCD(0x01);
                    }
delay_s(2);
CmdLCD(0x01);
}


if(READBIT(IOPIN0,SW2)==0)
{
while(READBIT(IOPIN0,SW2)==0);

//CmdLCD(0x80);
cnt=0;
for(i=0;i<=strlen(str2)-1;i++)
{
CmdLCD(0x80+cnt);
cnt++;
CharLCD(str2[i]);
}
//CmdLCD(0x01);

  i=0;
 if(str2[0]>=97&&str2[0]<=122)
       {
             str2[0]-=32;
             i++;
       }
       while(str2[i])
       {
             if(str2[i]==32)
             {
                   str2[i+1]-=32;
                                                str2[i-1]-=32;
                                 }


                                 i++;

                    }
                    if(str2[i]==0)
                                 {
                   str2[i-1]-=32;
```

```
        }
              // CmdLCD(0xC0);
               //CmdLCD(CLRSCR);
               cnt=0;
               for(i=0;i<=strlen(str2)-1;i++)
               {

               CmdLCD(0xC0+cnt);
               cnt++;
               CharLCD(str2[i]);
               //CmdLCD(0x01);
               }
               delay_s(2);
CmdLCD(0x01);
}

}
}
```

/*
14. Write an ECP to remove extra spaces in a given string and display the final result
on LCD.
Note: Connect two switches. Based on first switch press, take " An Apple a day" as
input and do the operation; Based on second switch press, take " vector stu !" as
input and do the operation. In both cases print input and output on lcd screen.
Output:
When Switch1 is pressed, display " An Apple a day" on the 1st line of LCD and when
the switch1 is released display "An Apple a day" on the 2nd line of LCD.
When Switch2 is pressed, display " vector stu !" on the 1st line of LCD and when
the switch2 is released display "vector stu !" on the 2nd line of LCD.
If none of the switch is pressed, display "waiting for input" on the 1st line of LCD.
***/**

```
#include<lpc21xx.h>
#include<string.h>
#include<stdlib.h>
#include "types.h"
#include "lcd_defines.h"
#include "defines.h"
#include "delay.h"
#include "lcd.h"
#define SW1 0
#define SW2 1
```

```
int main()
{
s8 str1[]="An  Apple  a day" ;
s8 str2[]="vector   stu   !";
      u32 i=0,cnt;

InitLCD();
while(1)
{


if(READBIT(IOPIN0,SW1)==0)
 {
 //while(READBIT(IOPIN0,SW2)==0);
  cnt=0;
for(i=0;i<=strlen(str1)-1;i++)
{
CmdLCD(0x80+cnt);
cnt++;
CharLCD(str1[i]);
}
i=0;
      while(str1[0]==32)
            memmove(str1,str1+1,strlen(str1+1)+1);
      while(str1[i])
      {
            if((str1[i]==32)&&(str1[i+1]==32))
            {
                  memmove(str1+i,str1+i+1,strlen(str1+i+1)+1);
            i--;
      }
      i++;
      }

      if(str1[strlen(str1)-1]==32)
            str1[i-1]=0;
      //CmdLCD(CLRSCR);
                  cnt=0;

                  for(i=0;i<=strlen(str1)-1;i++)
                  {

                  CmdLCD(0xc0+cnt);
                  cnt++;
                  CharLCD(str1[i]);
                  }
                 // CmdLCD(0xC0);
                  //StrLCD(str2);
                  delay_s(2);
}
```

```
if(READBIT(IOPIN0,SW2)==0)
 {
 //while(READBIT(IOPIN0,SW2)==0);
  cnt=0;
for(i=0;i<=strlen(str2)-1;i++)
{
CmdLCD(0x80+cnt);
cnt++;
CharLCD(str2[i]);
}
i=0;
	while(str2[0]==32)
		memmove(str2,str2+1,strlen(str2+1)+1);
	while(str2[i])
	{
		if((str2[i]==32)&&(str2[i+1]==32))
		{
			memmove(str2+i,str2+i+1,strlen(str2+i+1)+1);
		i--;
	}
	i++;
	}

	if(str2[strlen(str2)-1]==32)
		str2[i-1]=0;
	//CmdLCD(CLRSCR);
			cnt=0;

			for(i=0;i<=strlen(str2)-1;i++)
			{

			CmdLCD(0xc0+cnt);
			cnt++;
			CharLCD(str2[i]);
			}
			delay_s(2);
			// CmdLCD(0xC0);
			 //StrLCD(str2);
}

CmdLCD(0x01);
StrLCD("waiting 4 input");
delay_s(1);
}

}
```

#UART

//test_uart0.c

```
#include "types.h"
#include "defines.h"
#include<LPC21XX.H>
#include "uart_defines.h"
#include"uart.h"
s8 rDat[20] __attribute__((at(0x40000100)));
void InitUART0(void)
{
SETBIT(IODIR0,Tx_LED);
SETBIT(IODIR0,Rx_LED);
//cfg p0.0 and p0.1 are uart0tx,rx,fun
PINSEL0=TxD0_PIN_EN|RxD0_PIN_EN;
//cgf U0LCR reg
U0LCR=1<<DLAB_BIT|WORD_LEN;
//cfg BAUDRATE 9600
U0DLL=DIVISOR;
U0DLM=DIVISOR>>8;
//clr DLAB bit
CLRBIT(U0LCR,DLAB_BIT);
}
void U0TXCHAR(u8 sDat)
{
//load data in tx_buffer
U0THR=sDat;
//wait until transmitter(PISO reg)empty
while(READBIT(U0LSR,TEMT_BIT)==0);
//status for user visibility
CPLBIT(IOPIN0,Tx_LED);
}
void U0TXSTR(u8 *p)
{
while(*p)
U0TXCHAR(*p++);
}
void U0TXU32(u32 n)
{
u32 a[10];
s32 i=0;
if(n==0)
U0TXCHAR('0');
else
{
while(n)
{
a[i]=(n%10)+48;
```

```
i++;
n=n/10;
}
for(--i;i>=0;i--)
U0TXCHAR(a[i]);
}
}
void U0TXS32(s32 n)
{
if(n<0)
{
U0TXCHAR('-');
n=-n;
}
U0TXU32(n);
}
void U0TXF32(f32 f,u8 nDP)
{
u32 i;
s32 j;
if(f<0.0)
{
U0TXCHAR('-');
f=-f;
}
i=f;
U0TXU32(i);
U0TXCHAR('.');
for(j=0;j<nDP;j++)
{
f=(f-i)*10;
i=f;
U0TXCHAR(i+48);
}
}
u8 U0RXCHAR(void)
{
while(READBIT(U0LSR,DR_BIT)==0);
CPLBIT(IOPIN0,Rx_LED);
return U0RBR;
}
s8 * U0RXSTR(void)
{
static s8 a[20];
u32 i=0;
while(1)
{
a[i]=U0RXCHAR();
U0TXCHAR(a[i]);
if(a[i]=='\r')
```

```
{
a[i]='\0';
break;
}
i++;
}
U0TXSTR("\n\r");
return a;
}
```

//uart_lcd.c

```
#include<lpc21xx.h>
#include"lcd.h"
#include"lcd_defines.h"
#include"uart.h"
#include"delay.h"
u8 ch;
int main()
{
InitUART0();
InitLCD();

while(1)
{
ch=U0RXCHAR();
U0TXCHAR(ch);
delay_s(1);
CmdLCD(0x80);
CharLCD(ch);
delay_ms(1000);
}
}
```



```
#include<lpc21xx.h>
#include"lcd.h"
#include"lcd_defines.h"
#include"uart.h"
#include"delay.h"

int main()
{
u8 i,j,n;
u8 ch,k;
InitUART0();
```

```
InitLCD();
//u8 i,j,k;

while(1)
{
for(i=1;i<=n;i++)
{
for(j=1;j<=n+1-i;j++)
{
u32 k=n+1-j;
//U0TXStr("");
//U0TXCHAR((k/10)+48);
//U0TXCHAR(U0RXCHAR(k%10)+48);
delay_s(1);
CmdLCD(0x80);
//ch=U0RXCHAR((k%10)+48);
k=U0TXCHAR((k%10)+48);
U32LCD(k);
delay_ms(1000);
}
//U0TXSTR("\n\r");
delay_ms(100);
}

}
}
```

/* display in hyper uart char string integer float*/

```
#include<lpc21xx.h>
#include"delay.h"
#include"types.h"
#include"defines.h"
#include"uart.h"
#include"uart_defines.h"
u8 rDat __attribute__((at(0x40000010)));
main()
{
InitUART0();
U0TXCHAR('A');
delay_ms(100);
U0TXSTR("\n\r hello RAHUL S. BANSOD \n\r");
delay_ms(100);
U0TXSTR("\n\r");
U0TXS32(-12345);
```

```
U0TXSTR("\n\r");
U0TXU32(12345);
U0TXSTR("\n\r");
U0TXF32(12.345,3);
U0TXSTR("\n\r");
while(1);
}
```

/*
uart_pattern

*****
******
****
****
****

***/**

```
#include<lpc21xx.h>
#include"defines.h"
#include"types.h"
#include"delay.h"
```

//uart_defines.h

```
//u0LCR sfr defines
#define LEN_8BITS 3
#define WORD_LEN LEN_8BITS
#define DLAB_BIT 7
```

//u0LCR SFR defines

```
#define DR_BIT 0
#define THRE_BIT 5
#define TEMT_BIT 6
#define BAUD 9600
#define FOSC 12000000
#define CCLK (FOSC*5)
#define PCLK CCLK/4
#define DIVISOR (PCLK/(16*BAUD))
#define TxD0_PIN_EN 0x00000001
#define RxD0_PIN_EN 0X00000004
#define Tx_LED 6//p0.6
#define Rx_LED 7//P0.7
s8 a[20];
void InitUART(void)
{
SETBIT(IODIR0,Tx_LED);
```

```
SETBIT(IODIR0,Rx_LED);
PINSEL0=TxD0_PIN_EN|RxD0_PIN_EN;

U0LCR=1<<DLAB_BIT|WORD_LEN;
U0DLL=DIVISOR;
U0DLM=DIVISOR>>8;
CLRBIT(U0LCR,DLAB_BIT);
}
void U0TXCHAR(u8 sDat)
{
U0THR=sDat;
while(READBIT(U0LSR,TEMT_BIT)==0);
CPLBIT(IOPIN0,Tx_LED);
}
u8 U0RXChar(void)
{
while(READBIT(U0LSR,DR_BIT)==0);
CPLBIT(IOPIN0,Rx_LED);
return U0RBR;
}
void U0TXStr(u8 *p)
{
while(*p)
U0TXCHAR(*p++);
}
s8 *U0RXStr(void)
{

u32 i=0;
while(1)
{
a[i]=U0RXChar();
U0TXCHAR(a[i]);
if(a[i]=='\r')
{
a[i]='\0';
break;
}i++;
}
U0TXStr("\n\r");
return a;
}
s8 rDat[20] __attribute__((at(0x40000100)));
main()
{
while(1)
{
unsigned int i,j;
InitUART();
for(i=0;i<=5;i++)
```

```
{
for(j=1;j<=i;j++)
U0TXCHAR((i%10)+48);   //j
U0TXStr("\r\n");
delay_ms(100);
}
}
}
```

```
/*
ABCDEF FDCBA
ABCD      DCBA
ABC         CBA
AB            BA
A               A
*/

#include<lpc21xx.h>
#include"defines.h"
#include"types.h"
#include"delay.h"

//uart_defines.h
//u0LCR sfr defines
#define LEN_8BITS 3
#define WORD_LEN LEN_8BITS
#define DLAB_BIT 7
//u0LCR SFR defines
#define DR_BIT 0
#define THRE_BIT 5
#define TEMT_BIT 6
#define BAUD 19200
#define FOSC 12000000
#define CCLK (FOSC*5)
#define PCLK CCLK/4
#define DIVISOR (PCLK/(16*BAUD))
#define TxD0_PIN_EN 0x00000001
#define RxD0_PIN_EN 0X00000004
#define Tx_LED 6//p0.6
#define Rx_LED 7//P0.7
s8 a[20];
void InitUART(void)
{
SETBIT(IODIR0,Tx_LED);
SETBIT(IODIR0,Rx_LED);
PINSEL0=TxD0_PIN_EN|RxD0_PIN_EN;

U0LCR=1<<DLAB_BIT|WORD_LEN;
U0DLL=DIVISOR;
```

```
U0DLM=DIVISOR>>8;
CLRBIT(U0LCR,DLAB_BIT);
}
void U0TXCHAR(u8 sDat)
{
U0THR=sDat;
while(READBIT(U0LSR,TEMT_BIT)==0);
CPLBIT(IOPIN0,Tx_LED);
}
u8 U0RXChar(void)
{
while(READBIT(U0LSR,DR_BIT)==0);
CPLBIT(IOPIN0,Rx_LED);
return U0RBR;
}
void U0TXStr(u8 *p)
{
while(*p)
U0TXCHAR(*p++);
}
s8 *U0RXStr(void)
{

u32 i=0;
while(1)
{
a[i]=U0RXChar();
U0TXCHAR(a[i]);
if(a[i]=='\r')
{
a[i]='\0';
break;
}i++;
}
U0TXStr("\n\r");
return a;
}
s8 rDat[20] __attribute__((at(0x40000100)));
int main()
{
int n=6,a,b,B;
InitUART();
 for(a=0;a<=n;a++)
 {
 for(b=-n;b<=n;b++)
 {
 B=(b<0)?-b:b;
 if(B<a)
   {
```

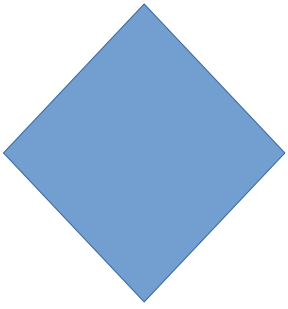
```
  U0TXCHAR(' ');
  delay_ms(10);
  }
  else
  {
  U0TXCHAR(64+n+1-B);
  delay_ms(10);
  }

  }
  U0TXStr("\n\r");
  }
  while(1);  }

  /*
******
*****
****
***
**
*

*/
s8 rDat[20] __attribute__((at(0x40000100)));
main()
{
unsigned int i,j,n=5;
InitUART();
for(i=1;i<=n;i++)
{
for(j=1;j<=n+1-i;j++)
U0TXCHAR('*');
U0TXStr("\n\r");
delay_ms(100);
}
}

/*
```



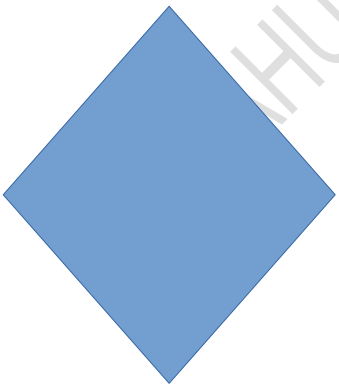
```
*/
```

```
s8 rDat[20] __attribute__((at(0x40000100)));
main()
{
unsigned int i,j,n=5;
InitUART();
for(i=1;i<=n;i++)
{
for(j=1;j<=n-i;j++)
U0TXCHAR(' ');

for(j=1;j<=2*i-1;j++)
U0TXCHAR('*');
U0TXStr("\n\r");
}
n--;
for(i=1;i<=n;i++)
{
for(j=1;j<=i;j++)
U0TXCHAR(' ');

for(j=1;j<=2*(n-i)+1;j++)
U0TXCHAR('*');
U0TXStr("\n\r");
//delay_ms(100);
}

//U0TXStr("\n\r");
delay_ms(10000);
}
```



```
s8 rDat[20] __attribute__((at(0x40000100)));
main()
{
unsigned int i,p,j,n=5;
InitUART();
for(i=1;i<=n;i++)
{
for(j=1;j<=n-i;j++)
```

```
U0TXCHAR(' ');
p=n;
for(j=1;j<=i;j++)
U0TXCHAR(((p++)%10)+48);
p=p+2;

for(j=1;j<=i;j++)
U0TXCHAR(((p--)%10)+48);

U0TXStr("\n\r");
}

for(i=1;i<=n;i++)
{
for(j=1;j<=i;j++)
U0TXCHAR(' ');
p=n;
for(j=1;j<=n-i+1;j++)
U0TXCHAR(((p++)%10)+48);
p=p+2;
for(j=1;j<=n-i;j++)
U0TXCHAR(((p++)%10)+48);
U0TXStr("\n\r");
//delay_ms(100);
}

//U0TXStr("\n\r");
delay_ms(1000);
}

/*
12345
 2345
  345
   45
    5
*/
s8 rDat[20] __attribute__((at(0x40000100)));
main()
{
while(1)
{
unsigned int i,j,n=5;
InitUART();
for(i=1;i<=5;i++)
{
U0TXStr("\n\r");
for(j=1;j<=5;j++)
{
if(j<i)
```

```
{
U0TXCHAR(' ');
U0TXCHAR(' ');
}
else
{
U0TXCHAR((j%10)+48);
U0TXCHAR(' ');
}
}
delay_s(1);
}}}
```

/* main_toggle_pin_3.c */

```
s8 rDat[20] __attribute__((at(0x40000100)));
main()
{
while(1)
{
unsigned int i,j,n=5;
InitUART();
for(i=1;i<=5;i++)
{
U0TXStr("\n\r");
for(j=1;j<=5;j++)
{
if(j<i)
{
U0TXCHAR(' ');
U0TXCHAR(' ');
}
else
{
U0TXCHAR((j%10)+48);
U0TXCHAR(' ');
}
}
delay_s(1);
}}}
```

#reference files

//types.h

```
typedef unsigned char u8;
typedef signed char s8;
typedef unsigned short u16;
typedef signed short s16;
typedef unsigned int u32;
typedef signed int s32;
typedef float f32;
typedef double f64;
typedef volatile unsigned int vu32;
```

// delay.h

```
#include "types.h"
void delay_us(u32 dlyus);
void delay_s(u32 dlys);
void delay_ms(u32 dlyms);
```

//delay.c

```
#include"types.h"
void delay_us(u32 dlyus)
{
dlyus*=12;
while(dlyus--);
}
void delay_ms(u32 dlyms)
{
dlyms*=12000;
while(dlyms--);
}
void delay_s(u32 dlys)
{
dlys*=12000000;
while(dlys--);
}
```

//defines.h

```
#define SSETBIT(WORD,BITPOS) (WORD=1<<BITPOS)
#define SETBIT(WORD,BITPOS) (WORD|=1<<BITPOS)
#define SETBYTE(WORD,BITPOS,BYTE) (WORD|=BYTE<<BITPOS)
#define SETNIBBLE(WORD,BITPOS,NIBBLE) (WORD|=NIBBLE<<BITPOS)
```

```
#define SCLRBIT    SSETBIT


#define CLRBIT(WORD,BITPOS)(WORD&=~(1<<BITPOS))
#define CLRNIBBLE(WORD,BITPOS,NIBBLE)
(WORD&=~(NIBBLE<<BITPOS))
#define CLRBYTE(WORD,BITPOS,BYTE)(WORD&=~(BYTE<<BITPOS))


#define CPLBIT(WORD,BITPOS)(WORD^=1<<BITPOS)
#define CPLNIBBLE(WORD,BITPOS,NIBBLE)(WORD^=NIBBLE<<BITPOS)
#define CPLBYTE(WORD,BITPOS,BYTE)(WORD^=BYTE<<BITPOS)


#define SET(WORD,BITPOS,BIT) (WORD|=(BIT<<BITPOS))
#define CLR(WORD,BITPOS,BIT) (WORD&=~(BIT<<BITPOS))
#define CPL(WORD,BITPOS,BIT) (WORD^=(BIT<<BITPOS))
#define WRITE(WORD,BITPOS,BIT) (WORD=((WORD&=~(BIT<<BITPOS))|
(BIT<<BITPOS)))


#define WRITEBIT(WORD,BITPOS,BIT) (WORD=((WORD&~(1<<BITPOS))|
(BIT<<BITPOS)))

#define READBIT(WORD,BITPOS)((WORD>>BITPOS)&1)

#define READWRITEBIT(WORD,DBIT,SBIT)(WORD=(WORD&~(1<<DBIT))|
(((WORD>>SBIT)&1)<<DBIT))

#define READWRITEBIT2(DWORD,WBIT,SWORD,RBIT)
(DWORD=((DWORD&~(1<<WBIT))|(((SWORD>>RBIT)&1)<<WBIT)))

#define WRITENIBBLE(WORD,BITPOS,NIBBLE)
(WORD=((WORD&~(15<<BITPOS))|(NIBBLE<<BITPOS)))

#define READNIBBLE(WORD,BITPOS)((WORD>>BITPOS)&15)

#define WRITEBYTE(WORD,BITPOS,BYTE)
(WORD=((WORD&~(255<<BITPOS))|(BYTE<<BITPOS)))

#define READBYTE(WORD,BITPOS) ((WORD>>BITPOS)&225)
```

//seg.c

```
#include <LPC21xx.h>
#include"defines.h"
#include"delay.h"
#include"seg.h"
#include"types.h"
```

```
u8
seglut[14]={0xc0,0xf9,0xa4,0xb0,0x88,0x99,0x92,0x82,0x83,0xc6,0xf8,0x80,0x90,
0xa1};
#define CA7SEG_2_MUX  0
#define DSEL1        8
#define DSEL2        9

void Init_CA7SEG_2_MUX(void)
{
WRITEBYTE(IODIR0,CA7SEG_2_MUX,0XFF);
SETBIT(IODIR0,DSEL1);
SETBIT(IODIR0,DSEL2);
}

void dispi_2_mux_ca7seg(u8 i)
{
WRITEBYTE(IOPIN0,CA7SEG_2_MUX,seglut[i/10]);
SSETBIT(IOSET0,DSEL1);
delay_ms(1);
SCLRBIT(IOCLR0,DSEL1);

WRITEBYTE(IOPIN0,CA7SEG_2_MUX,seglut[i%10]);
SSETBIT(IOSET0,DSEL2);
delay_ms(1);
SCLRBIT(IOCLR0,DSEL2);
}
```

//keypad.h

```
#include"types.h"
void InitRows(void);
void InitCols(void);
u8 colscan(void);
u8 keyscan(void);
```

//keypad.c

```
#include"types.h"
#include"defines.h"
#include"keypad.h"
#include <LPC21xx.H>
#define ROW0 16

#define ROW1 17

#define ROW2 18

#define ROW3 19

#define COL0 20
```

```
#define COL1 21

#define COL2 22

#define COL3 23


u8 keypadLUT[4][4]=

{

{1,2,3,4},{5,6,7,8},{9,0,'*','#'},{'!','@','$','&'}

};


void InitRows(void)

{

WRITENIBBLE(IODIR1,ROW0,0xf);

}


void InitCols(void)

{

WRITENIBBLE(IOSET1,COL0,0xf);

}


u8 colscan()

{

u8 t;

t=(READNIBBLE(IOPIN1,COL0));

t=(t<15)?0:1;

return t;

}
```

```
u8 keyscan(void)

{

u8 rNo,cNo,keyval=0;

while(colscan())
{
keyval++;
}

WRITENIBBLE(IOPIN1,ROW0,0xE);

if(!colscan())

{

rNo=0;

goto colcheck;//jumping statement

}


WRITENIBBLE(IOPIN1,ROW0,0xD);

if(!colscan())

{

rNo=1;

goto colcheck;

}


WRITENIBBLE(IOPIN1,ROW0,0xB);

if(!colscan())

{

rNo=2;

goto colcheck;

}
```

```
WRITENIBBLE(IOPIN1,ROW0,7);

if(!colscan())

{

rNo=3;

goto colcheck;

}

//jumping lable
colcheck:

 WRITENIBBLE(IOPIN1,ROW0,0);

if(READBIT(IOPIN1,COL0)==0)          cNo=0;

else if(READBIT(IOPIN1,COL1)==0)     cNo=1;

else if(READBIT(IOPIN1,COL2)==0)     cNo=2;

else if(READBIT(IOPIN1,COL3)==0)     cNo=3;


keyval=keypadLUT[rNo][cNo];

return keyval;

}
```

//sms_keypad.h

```
#ifndef __SMS_KEYPAD_H__
#define __SMS_KEYPAD_H__

u8 ColScan(void);
u8 InitKpm(void);
u8 RowCheck(void);
u8 ColCheck(void);
u8 KeyScan(void);
#endif
```

//sms_keypad.c

```
#include"types.h"
#include"defines.h"
#include<LPC21xx.h>
#include"kpm_defines.h"
const u8 kpm_LUT[4][4][4]=
```

```
{
        {'7','8','9','%','4','5','6','*','1','2','3','-','c','0','=','+',},
        {'a','d','g',' ','j','m','p',' ','s','v','y',' ',' ',' ',' ',' ',' ',},
        {'b','e','h',' ','k','n','q',' ','t','w','z',' ',' ',' ',' ',' ',' ',},
        {'c','f','i',' ','l','o','r',' ','u','x',' ',' ',' ',' ',' ',' ',' ',}

};

const u8 Scancode[4]={14,13,11,7};
s32 presskey,khcount;
void InitKpm(void)
{
WRITENIBBLE(IODIR1,Row0,15);
}
u8 ColScan(void)
{
u8 t;
t=(READNIBBLE(IOPIN1,Col0)<15)?0:1;
return t;
}
u8 Colcheck()
{
u8 i,cNo;
for(i=0;i<4;i++)
{
if(READNIBBLE(IOPIN1,Col0==Scancode[i]))
{
        cNo=i;
        break;
}
}
 return cNo;
 }
 u8 Rowcheck()
 {
 u8 i,rNo;
 for(i=0;i<4;i++)
 {
 WRITENIBBLE(IOPIN1,Row0,Scancode[i]);
 if(!ColScan())
 {
 rNo=i;
 break;
 }
 }
 WRITENIBBLE(IOPIN1,Row0,0);
 return rNo;
 }
 u8 KeyScan()
 {
```

```
u8 row=0,col=0;
static u8 prow=0,pcol=0;
row=Rowcheck();
col=Colcheck();
if(row!=prow)
{
khcount=0;
prow=row;
}
if(col!=pcol)
{
khcount=0;
pcol=col;
}
 return kpm_LUT[khcount][row][col];
}
```

//lcd.h

```
#include"types.h"
void WriteLCD(u8 Dat);
void CmdLCD(u8 Cmd);
void InitLCD(void);
void CharLCD(u8 C);
void StrLCD(u8 *s);
void U32LCD(u32 n);
void S32LCD(s32 n);
void F32LCD(f32 n,u8 nDp);
void BuildCGRAM(u8 *p,u8 nBytes);
//void BuildCGRAM(u8 *p,u8 nBytes);
void HexLCD(u32);
//void BinLCD(u32 n,u8 nBd);

void hexLCD (u32 n,u8 choice);
void OctLCD(u32 num);
```

//lcd_defines.h

```
#define LCD_DATA 8
#define LCD_RS   16
#define LCD_RW   18
#define LCD_EN   17

#define CLRSCR         0X01
#define DISP_ON_CUR_OFF 0X0C
#define DISP_ON_CUR_ON  0X0e
#define DISP_ON_CUR_BLK 0X0f

#define GOTO_LINE1_POS0 0X80
#define GOTO_LINE2_POS0 0XC0
#define SHIFT_CUR_RIGHT 0X06
#define GOTO_CGRAM      0X40
```

//lcd.c

```
#include "types.h"
#include "lcd_defines.h"
#include "defines.h"
#include<LPC21xx.h>
#include "delay.h"

void WriteLCD(u8 Dat)
{
	SCLRBIT(IOCLR0,LCD_RW);
	WRITEBYTE(IOPIN0,LCD_DATA,Dat);
	SSETBIT(IOSET0,LCD_EN);
	delay_us(1);
	SCLRBIT(IOCLR0,LCD_EN);
	delay_ms(2);
}
void CmdLCD(u8 cmd)
{
	SCLRBIT(IOCLR0,LCD_RS);
	WriteLCD(cmd);
}

/*
void WriteLCD(u8 dat)


{
	CLRBIT(IOPIN0,LCD_RW); //cfg for writing to LCD
	WRITEBYTE(IOPIN0,LCD_DATA,Dat);
	SETBIT(IOPIN0,LCD_EN); //high to...

	delay_us(1);
	CLRBIT(IOPIN0,LCD_EN); //low pulse
	delay_ms(2);  //atleast 2ms between consecutive writes
		//for syn bt/w cpu & LCD


}

*/



void InitLCD(void)
{
	delay_ms(15);
	WRITEBYTE(IODIR0,LCD_DATA,0xff);
	SETBIT(IODIR0,LCD_RS);
```

```
	SETBIT(IODIR0,LCD_RW);
	SETBIT(IODIR0,LCD_EN);

	CmdLCD(0x30);
	delay_ms(100);
	CmdLCD(0x30);
	delay_ms(100);
	CmdLCD(0x30);
	delay_ms(100);
	CmdLCD(0x38);
	CmdLCD(0x08);
	CmdLCD(0x01);
	CmdLCD(0x06);
	CmdLCD(0x0e);//0x0E,0x0F
	CmdLCD(GOTO_LINE1_POS0);
	CmdLCD(DISP_ON_CUR_BLK);
	CmdLCD(CLRSCR);
	CmdLCD(SHIFT_CUR_RIGHT);
}

 void CharLCD(u8 asciiV)
{
	SSETBIT(IOSET0,LCD_RS);
	WriteLCD(asciiV);
}

void StrLCD(s8 *s)
{

    while(*s)
	CharLCD(*s++);


}

void U32LCD(u32 n)
{
	s8 i=0;
	u8 a[10];
	SSETBIT(IOSET0,LCD_RS);
	if(n==0)
	{
		WriteLCD('0');
	}
	else
	{
		while(n)
		{
			a[i++]=(n%10)+48;
			n/=10;
```

```
		}
		for(--i;i>=0;i--)
		{
			WriteLCD(a[i]);
		}
	}
}

void S32LCD(s32 n)
{
	u32 t;
	SSETBIT(IOSET0,LCD_RS);
	if(n<0)
	{
		WriteLCD('-');
		t=-n;
	}
	U32LCD(t);
}

void F32LCD(f32 f,u8 nDP)
{
	u32 ipart;
	u8 i;
	if(f<0.0)
	{
		CharLCD('-');
		f=-f;
	}
	ipart=f;
	U32LCD(ipart);
	CharLCD('.');
	for(i=0;i<nDP;i++)
	{
		f=(f-ipart)*10;
		ipart=f;
		CharLCD(ipart+48);
	}
}
void Rotate_Array(u8 *p,u8 dir)
{
s8 i,temp;
u8 a[17];
if(dir=='r')
{
temp=a[15];
for(i=15;i>0;i--)
{
a[i]=a[i-1];
}
```

```
a[0]=temp;
}
else if(dir=='l')
{
temp=a[0];
for(i=0;i<15;i++)
{
a[i]=a[i+1];
}
a[15]=temp;
}
}
void BuildCGRAM(u8 *p,u8 nBytes)
{
u8 i;
//point to cgram
CmdLCD(GOTO_CGRAM);
for(i=0;i<nBytes;i++)
{
//write into cgram
CharLCD(p[i]);
}
//relocate tp DDRAM
//CmdLCD(GOTO_LINE1_POS0);
CmdLCD(GOTO_LINE1_POS0);
}
void HexLCD(u32 n)
{
u8 a[8]={'0'};
s8 i=0,t;
while(n)
{
t=(n%16);
a  [i++]=(t>9)?((t-10)+'A'):(t+48);
n/=10;
}
for(--i;i>=0;i--)
CharLCD(a[i]);
}
void hexLCD (u32 n,u8 choice)
{
u8 a[8]={'0'};
s8 i=0,t;
if(choice=='b')
{
while(n)
{
t=(n%16);
a[i++]=t>9?(t>10)+'A':(t+48);
n=16;
```

```
}
}
}
void OctLCD(u32 num)
{
	u32 octnum[100];
	u32 i=0;
		s32 j;
	while(num!=0)
	{
		octnum[i]=num%8;
		num=num/8;
		i++;
	}
	for(j=i-1;j>=0;j--)
	{
		U32LCD(octnum[j]);
	}

}
void BinLCD(u32 n,u8 nBD)
{
s8 i;
for(i=(nBD-1);i>=0;i--)
{
CharLCD(((n>>i)&1)+48);
}
}
```

//uart.h

```
#include "types.h"
void InitUART0(void);
void U0TXCHAR(u8);
void U0TXSTR(u8 *p);
void U0TXU32(u32);
void U0TXS32(s32);
void U0TXF32(f32, u8);
u8 U0RXCHAR(void);
s8* U0RXSTR(void);
```

//uart_defines.h

```
#define _8BITS 3
#define WORD_LEN  _8BITS
#define DLAB_BIT 7
//u0LCR SFR defines
#define DR_BIT 0
#define THRE_BIT 5
#define TEMT_BIT 6
#define BAUD 9600
#define FOSC 12000000
```

```
#define CCLK (FOSC*5)
#define PCLK CCLK/4
#define DIVISOR (PCLK /(16*BAUD))
#define TxD0_PIN_EN 0X00000001
#define RxD0_PIN_EN 0X00000004
#define Tx_LED 6//p0.6
#define Rx_LED 7//P0.7
```

//test_uart0.c

```
#include "types.h"
#include "defines.h"
#include<LPC21XX.H>
#include "uart_defines.h"
#include"uart.h"
s8 rDat[20] __attribute__((at(0x40000100)));

void InitUART0(void)
{
SETBIT(IODIR0,Tx_LED);
SETBIT(IODIR0,Rx_LED);
//cfg p0.0 and p0.1 are uart0tx,rx,fun
PINSEL0=TxD0_PIN_EN|RxD0_PIN_EN;  //0x05
//cgf U0LCR reg
U0LCR=1<<DLAB_BIT|WORD_LEN;  //0x83
//cfg BAUDRATE 9600
U0DLL=DIVISOR;
U0DLM=DIVISOR>>8;
//clr DLAB bit
CLRBIT(U0LCR,DLAB_BIT);
}
void U0TXCHAR(u8 sDat)
{
//load data in tx_buffer
U0THR=sDat;
//wait until transmitter(PISO reg)empty
while(READBIT(U0LSR,TEMT_BIT)==0);
//status for user visibility
CPLBIT(IOPIN0,Tx_LED);
}
void U0TXSTR(u8 *p)
{
while(*p)
U0TXCHAR(*p++);
}
void U0TXU32(u32 n)
{
u32 a[10];
s32 i=0;
if(n==0)
U0TXCHAR('0');
```

```
else
{
while(n)
{
a[i]=(n%10)+48;
i++;
n=n/10;
}
for(--i;i>=0;i--)
U0TXCHAR(a[i]);
}
}
void U0TXS32(s32 n)
{
if(n<0)
{
U0TXCHAR('-');
n=-n;
}
U0TXU32(n);
}
void U0TXF32(f32 f,u8 nDP)
{
u32 i;
s32 j;
if(f<0.0)
{
U0TXCHAR('-');
f=-f;
}
i=f;
U0TXU32(i);
U0TXCHAR('.');
for(j=0;j<nDP;j++)
{
f=(f-i)*10;
i=f;
U0TXCHAR(i+48);
}
}
u8 U0RXCHAR(void)
{
while(READBIT(U0LSR,DR_BIT)==0);
CPLBIT(IOPIN0,Rx_LED);
return U0RBR;
}
s8 * U0RXSTR(void)
{
static s8 a[20];
u32 i=0;
```

```
while(1)
{
a[i]=U0RXCHAR();
U0TXCHAR(a[i]);
if(a[i]=='\r')
{
a[i]='\0';
break;
}
i++;
}
U0TXSTR("\n\r");
return a;
}
```

/* i2c_defines.h */

```
#ifndef __I2C_DEFINES_H__
#define __I2C_DEFINES_H__

//defines for pin function selection
#define SCL_EN  0x00000010
#define SDA_EN  0x00000040

//defines for I2C_SPEED Configuration
#define CCLK      60000000  //Hz
#define PCLK      CCLK/4    //Hz
#define I2C_SPEED 100000    //Hz
#define LOADVAL   ((PCLK/I2C_SPEED)/2)
//bit defines for I2CONSET sfr
#define   AA_BIT 2
#define   SI_BIT 3
#define  STO_BIT 4
#define  STA_BIT 5
#define I2EN_BIT 6

#endif
```

//i2c.h

```
#ifndef __I2C_H__
#define __I2C_H__
#include "typ.h"

void init_i2c(void);
void i2c_start(void);
void i2c_stop(void);
void i2c_restart(void);
void i2c_write(u8);
void i2c_eeprom_write(u8,u16,u8);
u8   i2c_eeprom_read(u8,u16);
```

```
u8   i2c_read(void);
u8 i2c_ack(void);
u8 i2c_nack(void);
u8 i2c_masterack(void);

#endif
```

//i2c.c

```
#include <LPC21xx.h>
#include "typ.h"
#include "delay.h"
#include "i2c_defines.h"

void init_i2c(void)
{
          //Configure I/O pin for SCL & SDA functions using PINSEL0
          PINSEL0 |= (SCL_EN|SDA_EN);
          //Configure Speed for I2C Serial Communication
          //Using I2CSCLL
          I2SCLL=LOADVAL;
  //& I2CSCLH
          I2SCLH=LOADVAL;
         //I2C Peripheral Enable for Communication
          I2CONSET=1<<I2EN_BIT;
}


void i2c_start(void)
{
         // start condition
 I2CONSET=1<<STA_BIT;
         //wait for start bit status
         while(((I2CONSET>>SI_BIT)&1)==0);
         // clear start condition
         I2CONCLR=1<<STA_BIT;
}


void i2c_restart(void)
{
         // start condition
         I2CONSET=1<<STA_BIT;
         //clr SI_BIT
         I2CONCLR=1<<SI_BIT;
         //wait for SI bit status
         while(((I2CONSET>>SI_BIT)&1)==0);
         // clear start condition
         I2CONCLR=1<<STA_BIT;
}
```

```
void i2c_write(u8 dat)
{
        //put data into I2DAT
        I2DAT=dat;
        //clr SI_BIT
        I2CONCLR = 1<<SI_BIT;
        //wait for SI bit status
        while(((I2CONSET>>SI_BIT)&1)==0);
}


void i2c_stop(void)
{
        // issue stop condition
        I2CONSET=1<<STO_BIT;
  // clr SI bit status
        I2CONCLR = 1<<SI_BIT;
        //stop will cleared automatically
        //while(((I2CONSET>>STO_BIT)&1));
}


u8 i2c_nack(void)
{
        I2CONSET = 0x00; //Assert Not of Ack
    I2CONCLR = 1<<SI_BIT;
        while(((I2CONSET>>SI_BIT)&1)==0);
        return I2DAT;
}


u8 i2c_masterack(void)
{
        I2CONSET = 0x04; //Assert Ack
    I2CONCLR = 1<<SI_BIT;
        while(((I2CONSET>>SI_BIT)&1)==0);
        I2CONCLR = 0x04; //Clear Assert Ack
        return I2DAT;
}
```

/* i2c_eeprom.h */

```
#ifndef __I2C_EEPROM_H__

#define __I2C_EEPROM_H__


#include "types.h"
```

```
void i2cDevWrite(u8 slaveAddr,u8 wBuffStartAddr,u8 *p,u8 nBytes);

void i2cDevRead(u8 slaveAddr,u8 rBuffStartAddr,u8 *p,u8 nBytes);


#endif
```

/* i2c_eeprom.c */

```
#include <LPC21xx.h>
#include "typ.h"
#include "i2c.h"
#include "delay.h"


void i2c_eeprom_write(u8 slaveAddr,u16 wBuffAddr,u8 dat)
{
  i2c_start();
  i2c_write(slaveAddr); //slaveAddr + w
  i2c_write((wBuffAddr>>8)&0xFF);    //wBuffAddr
  i2c_write((wBuffAddr&0xFF));
  i2c_write(dat);    //wBuffAddr
  i2c_stop();
  delay_ms(10);
}


u8 i2c_eeprom_read(u8 slaveAddr,u16 rBuffAddr)
{
        u8 dat;
        i2c_start();
    i2c_write(slaveAddr); //slaveAddr + w
        i2c_write((rBuffAddr>>8)&0xFF);    //rBuffAddr
        i2c_write((rBuffAddr&0xFF));
        i2c_restart();
        i2c_write(slaveAddr|1); //slaveAddr + r
    dat=i2c_nack();
        i2c_stop();
        return dat;
}
```

//uart_interrupt

```
#include <LPC21xx.H>
/* LPC21xx definitions         */
#include <string.h>

#include "uart.h"
```

```
#define UART_INT_ENABLE 1

//defines for UART
#define RXD0_EN   1<<0
#define TXD0_EN   1<<2
/*
UART0:
BAUD RATE = PCLK/(16*DIVISOR);
DIVISOR = (U0DLM*256) + U0DLL;
*/
#define FOSC      12000000
#define CCLK            5*FOSC
#define PCLK            CCLK/4
#define BAUD            9600
#define DIVISOR   (PCLK/(16 * BAUD))

//bit defines for U0LCR
#define DLAB_BIT 7
//bit defines for U0LSR
#define RDR_BIT  0
#define THRE_BIT 5//recheck
#define TEMT_BIT 6

//char buff[6]="hello",dummy;
char buff[20];
char dummy;
unsigned char i=0,rx,flag,t=0;

void UART0_isr(void) __irq
{
 if((U0IIR & 0x04)) //check if receive interrupt
 {
                rx =  U0RBR; /* Read to Clear Receive Interrupt */
                if(rx==0X02)
                        flag=1;
                else if((flag == 1) && (rx!=0X03))
                {
                        buff[i++] = rx;
                }
                else
                {
                        buff[i] = '\0';
                        i=0;
                        flag = 2;
                }
 }
 else
 {
    dummy=U0IIR; //Read to Clear transmit interrupt
 }
```

```
   VICVectAddr = 0; /* dummy write */
}

void Init_UART0 (void) /* Initialize Serial Interface       */
{
  PINSEL0 |= 0x00000005; /* Enable RxD0 and TxD0              */

  U0LCR = 0x83;         /* 8 bits, no Parity, 1 Stop bit     */
  U0DLL = 97;           /* 9600 Baud Rate @ CCLK/4 VPB Clock  */
  U0LCR = 0x03;         /* DLAB = 0  */

  #if UART_INT_ENABLE > 0

  VICIntSelect = 0x00000000; // IRQ
  VICVectAddr0 = (unsigned)UART0_isr;
  VICVectCntl0 = 0x20 | 6; /* UART0 Interrupt */
  VICIntEnable = 1 << 6;   /* Enable UART0 Interrupt */
 // U0IIR = 0xc0;
 // U0FCR = 0xc7;
  U0IER = 0x03;       /* Enable UART0 RX and THRE Interrupts */

  #endif
}

 void UART0_Tx_char(unsigned char ch)  /* Write character to Serial Port    */
{
  while (!(U0LSR & 0x20));
  U0THR = ch;
}
unsigned char UART0_Rx_char(void)

{

  while(((U0LSR>>RDR_BIT)&1)==0);
          return U0RBR;

}


void UART0_Tx_str(char *s)
{
   while(*s)
      UART0_Tx_char(*s++);
}
```

#CAN

//mainnodetx2.c
/*fuel*/

```
#include <lpc21xx.h>
#include"adc_defines.h"
#include "types.h"

#include"delay.h"

#include "can.h"

#include"lcd.h"

#include"lcd_defines.h"

#include"adc.h"

f32 AR1;

u8 fuel_per;


int main(void)

{

        struct CAN_Frame txFrame;

        txFrame.ID=3;   txFrame.vbf.RTR=0; //data frame

        txFrame.Data1=0x87654321; txFrame.Data2=0x38392635;

        Init_CAN1();
        InitLCD();

        Init_ADC();

        CmdLCD(0x80);
        StrLCD("fuel_per:");
        while(1)
        {
                AR1=Read_ADC(CH1);
                fuel_per=(AR1-0.470)/(3.299-0.470)*100;

                                CmdLCD(0x01);
                CmdLCD(0x80);
```

```
                StrLCD("fuel_per:");
                U32LCD(fuel_per);

                        CmdLCD(0xC0);

                        F32LCD(AR1,3);
                delay_ms(200);
                txFrame.Data1=fuel_per;
                txFrame.vbf.DLC=1;
                CAN1_Tx(txFrame);
                delay_ms(200);

        }

}
```

//main_rx.c

```
#include <LPC21xx.h>

#include"can.h"

#include"can_defines.h"

#include"lcd.h"

#include"lcd_defines.h"

#include"delay.h"

main()

{
        f32 ch;

        struct CAN_Frame rxFrame;

        Init_CAN1();
        InitLCD();

        while(1)
        {

      CAN1_Rx(&rxFrame);

                ch=(rxFrame.Data1&0xFF);

                if(rxFrame.ID==2)
                {
```

```
                CmdLCD(0x80);
                StrLCD("temp:");
                F32LCD(ch,2);

        }
        else if(rxFrame.ID==3)

        {

                CmdLCD(0xc0);

                StrLCD("fuel_per:");

                F32LCD(ch,2);
        }
        delay_ms(500);
    }
}
```

//can.h

```
#ifndef __CAN_H__

#define __CAN_H__

#include "types.h"

struct CAN_Frame

{

    u32 ID;

    struct BitField

    {

        u8 RTR : 1;

        u8 DLC : 4;

    }vbf;

    u32 Data1,Data2;//8-bytes

};


void Init_CAN1(void);
```

```
void CAN1_Tx(struct CAN_Frame);

void CAN1_Rx(struct CAN_Frame *);


#endif
//can_defines.h
#define QUANTA     16

#define BRP        (PCLK/(BIT_RATE*QUANTA))

#define SAMPLE_POINT  (0.7 * QUANTA)

#define TSEG1        ((int)SAMPLE_POINT-1)  //TSEG1=prop_seg+Tph1_seg


#define TSEG2        (QUANTA-(1+TSEG1))   //TSEG2=Tph2_seg

#define SJW          ((TSEG2 >= 5)   ?   4 : (TSEG2-1))

#define SAM          0 //0 or 1 ,sample bus 1 or 3 time(s)

#define BTR_LVAL    (SAM<<23|(TSEG2-1)<<20|(TSEG1-1)<<16|(SJW-1)<<14|
(BRP-1))


//defines for C1CMR bit set


#define TR_BIT_SET   1<<0

#define RRB_BIT_SET  1<<2

#define STB1_BIT_SET 1<<5



//defines for C1GSR bit check


#define RBS_BIT_READ  1<<0

#define TBS1_BIT_READ 1<<2

#define TCS1_BIT_READ 1<<3
```

```
//defines for C1CMR bit set


#define TR_BIT_SET   1<<0

#define RRB_BIT_SET  1<<2

#define STB1_BIT_SET 1<<5


//defines for C1GSR bit check


#define RBS_BIT_READ  1<<0

#define TBS1_BIT_READ 1<<2

#define TCS1_BIT_READ 1<<3
```

//can.c

```
#include <lpc21xx.h>
#include "types.h"
#include"can.h"
#include"can_defines.h"


/*CAN Controller 1 Initialization : (defined in can.c )*/

void Init_CAN1(void)
{
   //cfg p0.25 pin as CAN1_RX pin(RD1),TD1 is exclusive
   PINSEL1|=RD1_PIN; //using defines from can_defines.h


                  // #define RD1_PIN 0x00040000 ,


                        //as RD1/ (i.e CAN1_RX)/p0.25
   //Reset CAN1 controller
   C1MOD=1;
   //All received messages are accepted
   AFMR=2;
   //Set baud Rate for CAN
   C1BTR=BTR_LVAL; //using defines from can_defines.h
   //Enable CAN1 controller
   C1MOD=0;
}
```

```
void CAN1_Tx(struct CAN_Frame txFrame)
{
        // Checking that the TX buffer is empty in C1GSR
    while((C1GSR&TBS1_BIT_READ)==0); //if status is 1 then empty

 // place 11-bit tx id in C1T1D1
        C1TID1=txFrame.ID;
 // place cfg whether data/remote frame & no of data bytes in message
   C1TFI1=txFrame.vbf.RTR<<30|txFrame.vbf.DLC<<16;
        // For Data Frame place 1 to 8 bytes data into Data Tx Buffers
        if(txFrame.vbf.RTR!=1)
   {
      //Place data bytes 1-4 in C1TDA1
                C1TDA1= txFrame.Data1;
      //Place data bytes 5-8 in C1TDB1
        C1TDB1= txFrame.Data2;
        }
   //Select Tx Buf1 & Start Xmission using
    C1CMR=STB1_BIT_SET|TR_BIT_SET;
        //monitor tx status in C1GSR
    while((C1GSR&TCS1_BIT_READ)==0);
}

void CAN1_Rx(struct CAN_Frame *rxFrame)
{
//wait for CAN frame recv status
        while((C1GSR&RBS_BIT_READ)==0);
//read 11-bit CANid of recvd frame.
        rxFrame->ID=C1RID;
// read & extract data/remote frame status
        rxFrame->vbf.RTR=(C1RFS>>30)&1;
//read & extract data length
        rxFrame->vbf.DLC=(C1RFS>>16)&0x0f;


//check if recvd frame is data frame,extract data bytes
if(rxFrame->vbf.RTR==0)
        {
//read 1-4 bytes from C1RDA
                rxFrame->Data1=C1RDA;
//read 5-8 bytes from C1RDB
        rxFrame->Data2=C1RDB;
}

// Release receive buffer
     C1CMR=RRB_BIT_SET;

}
```

//ADC.c

```
#include <LPC214x.h>

#include "types.h"
#include "defines.h"
#include "adc_defines.h"
#include "delay.h"

#define ADC_FUNC 0x01

void Init_ADC(void)
{
        PINSEL1 |= (ADC_FUNC<<24);  //configure P0.28 as ADC input
  AD0CR=PDN_BIT|CLKDIV|CHANNEL_SEL;
}

f32 Read_ADC(u8 chNo)
{
  u16 adcVal=0;
        f32 eAR;
        WRITEBYTE(AD0CR,0,chNo);
        SETBIT(AD0CR,ADC_START_BIT);
        delay_us(3);
        while(!READBIT(AD0GDR,DONE_BIT));
        CLRBIT(AD0CR,ADC_START_BIT);
        adcVal=(AD0GDR>>6)&0x3FF;
        eAR=((adcVal*3.3)/1023)*100;
        return eAR;
}
```

/*adc_defines.h*/

```
//defines for ADCR
#define CH0             0x01
#define CH1             0x02
#define CH2             0x04
#define CH3             0x08
#define CHANNEL_SEL         0
#define FOSC            12000000
#define CCLK            (5*FOSC)
#define PCLK            (CCLK/4)
#define ADCLK           3750000
#define CLKDIV          (((PCLK/ADCLK)-1)<<8)
#define PDN_BIT         (1<<21)
#define ADC_START_BIT       24
```

```
//defines for ADDR
#define DONE_BIT            31
```

/* adc.c */

```
#include"adc_defines.h"
#include"types.h"
#include<LPC21xx.h>
#include"adc.h"
#include"delay.h"
#include"defines.h"

void initadc(void)
{
ADCR=PDN_BIT|CLKDIV|CHANNEL_SEL;
}
f32 readadc(u8 chno)
{
u16 adcdval;
f32 eAR;
WRITEBYTE(ADCR,0,chno);
SETBIT(ADCR,ADC_START_BIT);
delay_us(3);
while(READBIT(ADDR,DONE_BIT)==0);
CLRBIT(ADCR,ADC_START_BIT);
adcdval=((ADDR>>6)&1023);
eAR=(adcdval*(3.3/1023));
return eAR;
}
```

/* adc.h */

```
void initadc(void);
f32 readadc(u8 channelno);
```

/* main_adc_potentiometer.c */

```
#include"types.h"
#include"delay.h"
#include"lcd_definees.h"
#include"lcd.h"
#include"adc.h"
#include"adc_defines.h"
f32 ar0;
int main(void)
{
initlcd();
initadc();
strlcd("ON-CHIP ADC:");
while(1)
```

```
{
ar0=readadc(CH0);
cmdlcd(GOTO_LINE2_POS0);
strlcd("              ");
cmdlcd(GOTO_LINE2_POS0);
/*if((ar0>0.48)&&(ar0<1.022))
strlcd("simulation error");
else */
f32lcd(ar0,3);
delay_ms(100);
}
```

#lm35_tempterature sensor

```
//lm35.h
#include"types.h"
f32 RdTempLM35(void);
f32 RdTempLM35_NP(void);
```

//main_adc_lm35.c

```
#include"lcd.h"
#include"lcd_definees.h"
#include"types.h"
#include"delay.h"
#include"lm35.h"

int main(void)
{
initlcd();
strlcd("LM35 READING:");
while(1)
{
cmdlcd(GOTO_LINE2_POS0);
strlcd("              ");
cmdlcd(GOTO_LINE2_POS0);
f32lcd(RdTempLM35_NP());
strlcd("degC");
delay_ms(100);
}
}
```

//lm35.c

```
#include”type.h”
#include”adc_defines.h”
#include”adc.h”
```

```
f32RdTempLM35(void)
{
static u8 flag;
if(flag==0)
{
InitADC();
flag++;
}
return(ReadADC(CHO)*100);
}
f32 RdTempLM35_NP(void)
{
static u8 flag;
f32 aR0,aR1;
if(flag==0)
{
InitADC();
flag++;
}
aR0=RaedADC(CHO);
aR1=ReadADC(CH1);
return((aR0-aR1)*100;
}
```

#distance sensor

//main_adc_gp2d12.c

```
#include"lcd.h"
#include"lcd_defines.h"
#include"types.h"
#include"delay.h"
#include"gp2d12.h"

int main(void)
{
Initlcd();
strlcd("OBJECT RANGE");
while(1)
{
cmdlcd(GOTO_LINE2_POS0);
strlcd("              ");
cmdlcd(GOTO_LINE2_POS0);
u32lcd(RdRangeGP2D12());
strlcd("on");
delay_ms(100);
}
}
```

//gp2d12.c

```
#include"types.h"
#include"adc_defines.h"
#include"adc.h"
u8 RdRangeGP2D12(void)
{
f32 aR;
u16 adcDval;
u8 rangecm;
static u8 flag;
if(flag==0)
{
InitADC();
flag++;
}
aR=readADC(ch0);
adcDval=(1024/5.0)*aR;
rangecm=((6787/(adcDval-3))-4);
return rangecm;
}
```

//gp2d12.h

```
#ifndef __GP2D12_H__
#define __GP2D12_H__
#include"types.h"
extern u8 RdRangeGP2D12(void);
#endif
```

//onelinekpm.c

```
#include"types.h"
#include"adcdefines.h"
#include"adc.h"
#include"onelinekpm.h"

f32 keyscan(void)
{
f32 aR;
static u8 flag;
if(flag==0)
{
InitADC();
flag++;
}
aR=readADC(ch0);
return mapkey(aR);
}
u32 mapkey(f32 aR)
```

```
{
u32 keyV;
if(aR>0.12 && aR<0.15)
keyV='0';
else if(aR>0.28 && aR<=0.32)
keyV='1';
else if(aR>=0.33 && aR<=0.40)
keyV='2';
else if(aR>=0.45 && aR<=0.65)
keyV='3';
else if(aR>=0.75 && aR<=0.85)
keyV='4';
else if(aR>=1.25 && aR<=1.50)
keyV='5';
else if(aR>=2.00 && aR<=2.20)
keyV='6';
else if(aR>=2.23 && aR<=2.30)
keyV='7';
else if(aR>=2.50 && aR<=2.85)
keyV='8';
else if(aR>=2.86 && aR<=2.88)
keyV='9';
else if(aR>=2.89 && aR<=2.92)
keyV='A';
else if(aR>=2.98 && aR<=3.20)
keyV='B';
return keyV;
}
```

//onelinekpm.h

```
#ifndef __ONELINEKPM_H__
#define __ONELINEKPM_H__
#include"types.h"
f32 keyscan(void);
u32 mapkey(f32);
#endif
```

//main_file_onelineKPM.c

```
#incluede“types.h”
#include”delay.h”
#include”lcd_defines.h”
#include”lcd.h”
#include”onelineKPM.h”
main()
{
InitLCD();
StrLCD(“one line KPM”);
while(1);
{
CmdLCD(GOTO_LINE2_POS0);
```

```
StrLCD(“                “);
Cmd(GOTO_LINE2_POS0);
//F32LCD(keyscan(),2);
CharLCD(keyscan());
delay_ms(100);
}
}
```

//external_interrupt_lcd.c

```
#include<lpc21xx.h>
#include"types.h"
#include"defines.h"
#include"lcd_defines.h"
#include"lcd.h"
#include"delay.h"

#define EINT1_PIN 3
#define FUN4 3
#define EINT1_PIN_FUN FUN4
#define CFGPIN(WORD,BITPAIRPOS,FUN)
WORD=((WORD&(~(3<<BITPAIRPOS)))|(FUN<<BITPAIRPOS))

 void eint1_isr(void)__irq
            {
            cmdLCD(GOTO_LINE2_POS0);
            strLCD("IN ISR");
            delay_ms(1000);
            cmdLCD(GOTO_LINE2_POS0);
            strLCD("              ");
            cmdLCD(GOTO_LINE2_POS0);
            strLCD("EXIT ISR");
            EXTINT=0x02;
            VICVectAddr=0;
            }
void Enable_EINT1(void)
            {
            CFGPIN(PINSEL0,6,EINT1_PIN_FUN);
            EXTMODE=0X02;
            EXTPOLAR=0X02;
            VICIntEnable=1<<15;
            VICVectCntl0=1<<5|15;
            VICVectAddr0=(unsigned)eint1_isr;
            }
             u32 count;
             main()
             {
             InitLCD();
             Enable_EINT1();
```

```
while(1)
{
cmdLCD(GOTO_LINE2_POS0);
strLCD("IN MAIN");
delay_ms(2000);
cmdLCD(CLRSCR);
count++;
}
}
```

//external_interrupt_test.c

```
#include<lpc21xx.h>
#include"types.h"
#include"defines.h"
#define EINT0_PIN 1
#define EINT0_LED 16
#define FUN4      3
#define EINT0_PIN_FUN  FUN4
#define CFGPIN(WORD,BITPAIRPOS,FUN)
WORD=((WORD&(~(3<<BITPAIRPOS)))|(FUN<<BITPAIRPOS))

void eint0_isr(void) __irq
{
CPLBIT(IOPIN0,EINT0_LED);
EXTINT=0X01;
VICVectAddr=0;
}
void enable_EINT0(void)
{
SETBIT(IODIR0,EINT0_LED);
PINSEL0=0X0000000C;
EXTMODE=0X01;
EXTPOLAR=0X01;
VICIntEnable=1<<14;
VICVectCntl0=1<<5|14;
VICVectAddr0=(unsigned)eint0_isr;
}
u32 count;
main()
{
enable_EINT0();
while(1)
{
count++;
}
}
```

//Testing_lcd_interrupt_1.c

```
#include<LPC21XX.h>
#include”types.h”
#include”defines.h”
#include”delay.h”

#define EINT0_PIN 1
#define EINT1_PIN 3
#define EINT0_PIN 16
#define EINT01_PIN 17

#define FUN4 3

#define EINT0_PIN_FUN FUN4
#define  EINT1_PIN_FUN FUN4
#define CFGPIN(WORD,PIN,FUN)\
WORD=((WORD&(~(3<<(PIN*2))))|(FUN<<(PIN*2)))
void enit0_isr(void) __irq
{
CPLBIT(IOPIN0,EinT0_LED);
EXTINT=0x01;
VICVectAddr=0;
delay_ms(500);
}
void eint1_isr(void) __irq
{
CPLBIT(IOPIN0,EINT1_LED);
EXTINT=0x02;
VICVectAddr=0;
dealy_ms(100);
}
void Enable_ENITS(void)
{
SETBIT(IODIR0,EINT0_LED);
SETBIT(IODIRR0,EINT1_LED);
//cfg gpio p0.1 pin as eint0 input pin

CFGPIN(PINSEL0,1,EINT0_PIN_FUN);
 //cfg gpio p0.3 pin as eint1 input pin

CFGPIN(PINSEL0,3,EINT1_PIN_FUN);
//cfg external interrupt peripheral
//EXTINT=0x00;
//cfg eint0,eint1 to taken as E_TRIG

EXTMODE=0x03;
//cfg eint0,eint1 to be taken as Redge
```

```
EXTPOLAR=0x03;
//VICIntselect=0; //all irqs
//enable eint 0,eint1 channel in VIC

VICIntEnable=1<<14|1<<15;
//allow isr address to load into VICVectAddrN
//With req interrupt source

VICVectcntl0=1<<5|14;
VICVectAddr0=(unsigned)eint0_isr;
}
u32 count;
main()
{
Enable_EINTS();
while(1)
{
count++;
}
}

/*
//eints_debug.ini
define button"TGL_EINT_PINS","TogglePortPIn()"//*(1,3)

FUNC void TogglePortPins(void) //(unsigned pin1,unsigned pin2)
{
PORT0^=1<<1|1<<3;
PORT0^=1<<1|1<<3;
}
//eints_debug.ini
define button "TGL_EINT_PINS",TogglePortPins(1,3)"

FUNC void ToggelPortPins(unsigned p1, unsigned p2)

PORT0^=1<<pin1|1<<pin2
PORT0^=1<<pin1|1<<pin2
}
*/
```