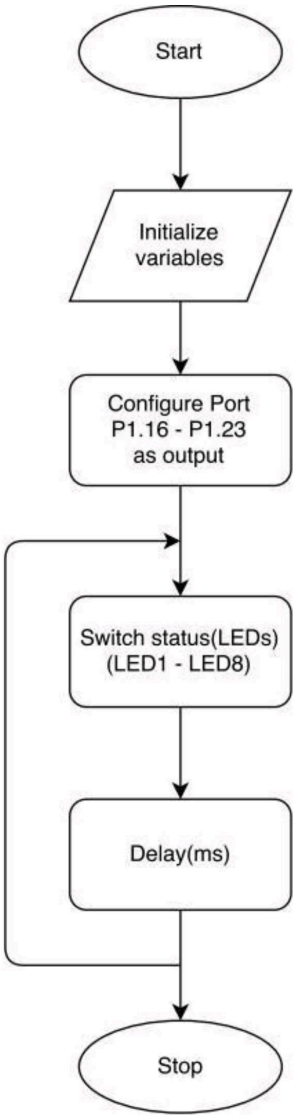
Expt no: 2 **INTERFACING LED**

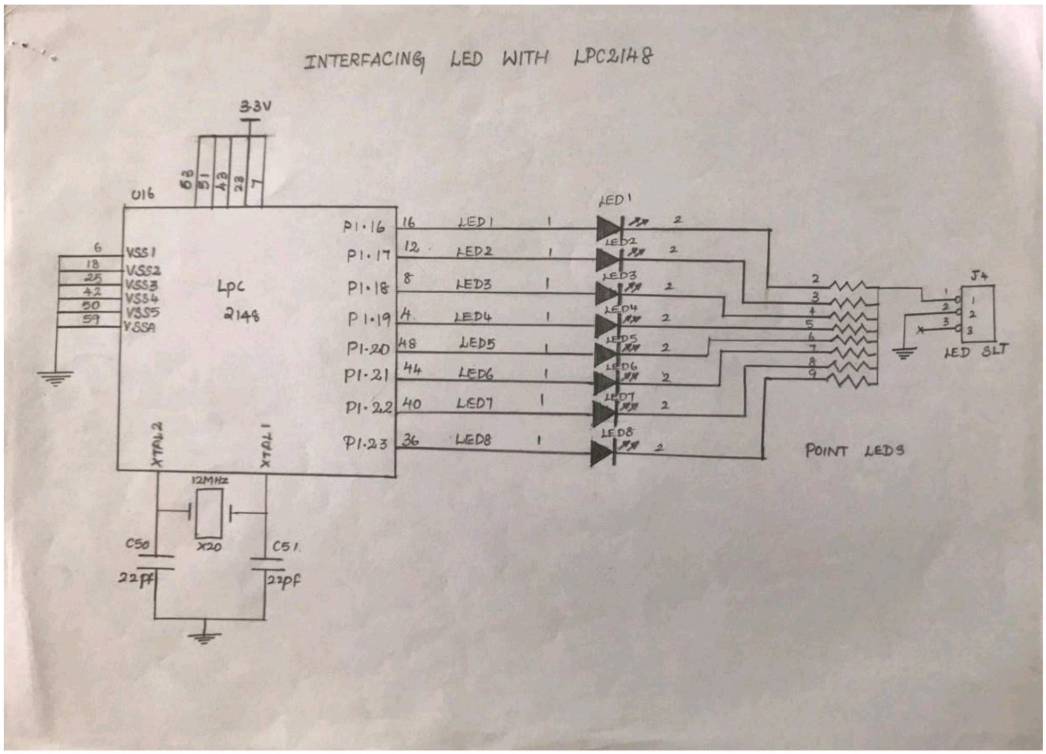
Aim: To interface LED with ARM LPC2148

Apparatus Required: LPC2148 ARM evaluation kit, KeiluVision, Flash Magic

Flowchart:



Interfacing Diagram:



Program:

```
#include <lpc214x.h>
#include <stdio.h>

#define LED_ROW 0xFF0000                    // LEDs are connected fromP1.16 to P1.23

void Delay(unsigned int n);

int main(void)
{
        IODIR1  = LED_ROW;                  //Configure P1.16 to P1.23 as Output

        while(1)                            //loop forever
        {
                IOSET1          = LED_ROW; //Switch on the LEDs
                Delay(20);              //Give a delay of 20ms
```

```
                        IOCLR1          = LED_ROW; //Switch off the LEDs
                        Delay(20);                 //Give a delay of 20ms
                }
}

void Delay(unsigned int n)
{
        int i,j;
        for(i=0;i<n;i++)
        {
                for(j=0;j<120000;j++)
                {;}
        }
}
```

Result:

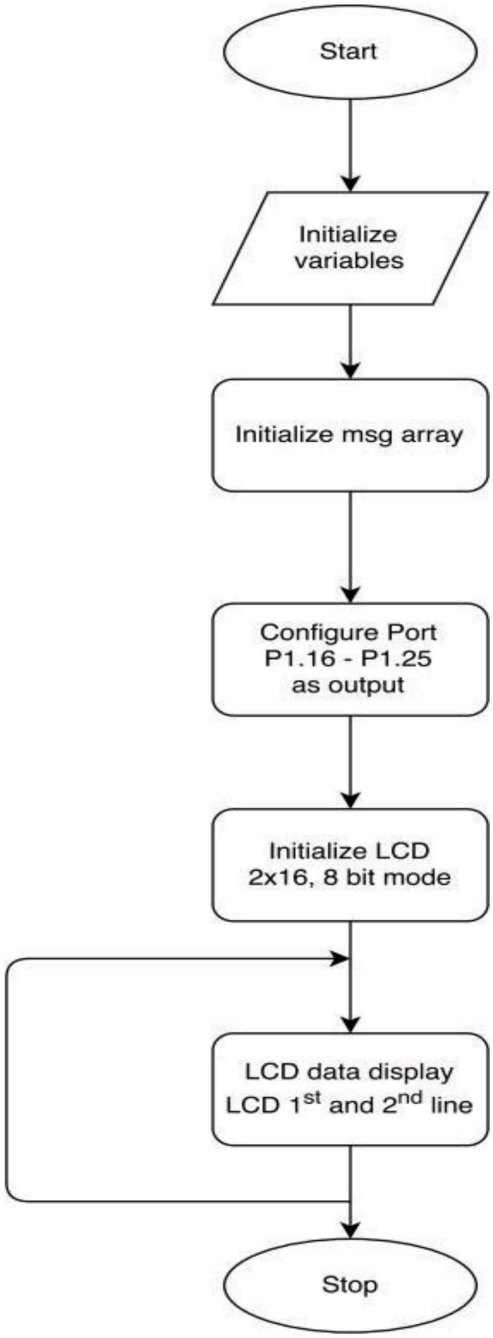
Thus the LEDs are interfaced with ARM LPC2148.

Expt no: 3 INTERFACING LCD

Aim: To program the ARM processor to display a message in LCD.

Apparatus Required: LPC2148 ARM evaluation kit, 2X16 LCD display, KeiluVision, Flash Magic

Flowchart



Program

```
#include <LPC214x.H>
#define RS      1<<16
#define EN      1<<17
void lcd_Init (void);
void lcd_cmd (char);
void lcd_data (char);
void lcd_display (void);
void delay(int);
char cmd[8] = {0x33,0x32,0x38,0x0C,0x06,0x01};
char msg[] = {" ARM7 TYRO VER4 "};
char msg1[]= {"  > LCD DEMO <  "};

int  main(void)
{
        IO1DIR =        0x3FF0000;
        lcd_Init();

        while(1)
        {
                lcd_cmd(0x01);
                lcd_display();
                delay(300);
        }
}

void lcd_Init(void)                                     // LCD Initialize
{
        unsigned int i;
        for(i=0;i<6;i++)
        {
                lcd_cmd(cmd[i]);
                delay(10);
        }
}

void lcd_cmd(char data)                                 // LCD Command Mode
{
        IO1CLR = 0xFFFFFFFF;
        IO1CLR= RS;
        IOPIN1 = (IOPIN1 | data << 18);
        IOSET1 = EN;
        delay(5);
        IOCLR1= EN;
}
```

```
void lcd_data (char data)                    // LCD Data Mode
{
        IO1CLR = 0xFFFFFFFF;
        IOSET1 = RS;
        IOPIN1 = (IOPIN1 | data << 18);
        IOSET1 = EN;
        delay(5);
        IOCLR1= EN;
}

void lcd_display (void)
{
        unsigned int i;

        lcd_cmd(0x80);                       // Display First Line
        delay(10);

        for(i=0;msg[i]!='\0';i++)
        {
                lcd_data(msg[i]);
                delay(40);
        }

        lcd_cmd(0xC0);                       // Display Second Line
        delay(10);

        for(i=0;msg1[i]!='\0';i++)
        {
                lcd_data(msg1[i]);
                delay(40);
        }
}

void delay(int n)
{
        unsigned int i,j;
        for(i=0;i<n;i++)
        {
                for(j=0;j<12000;j++)
                {;}
        }
}
```

Result:

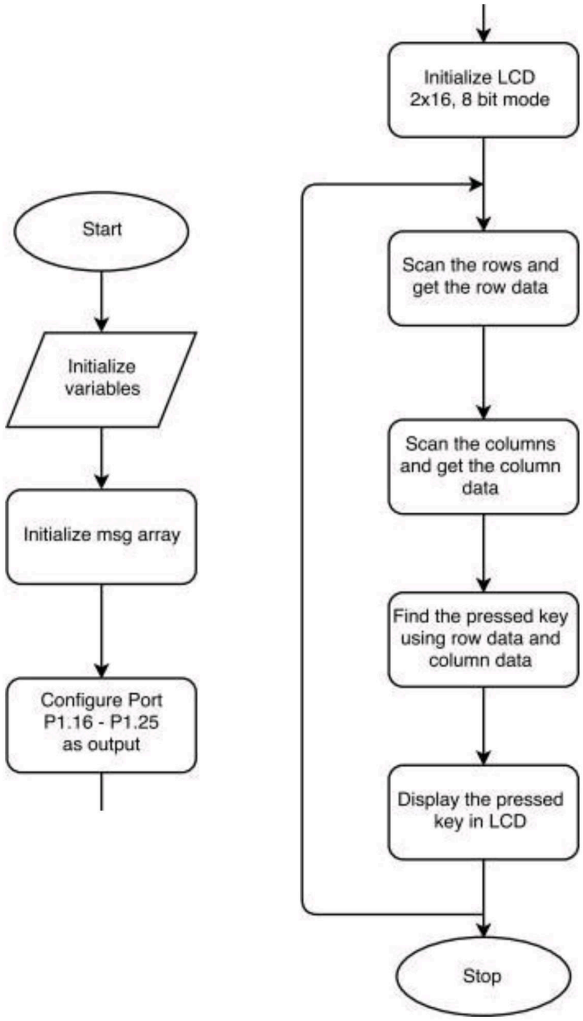
Thus the LCD is interfaced with ARM LPC2148.

Expt no 4: INTERFACING KEYPAD

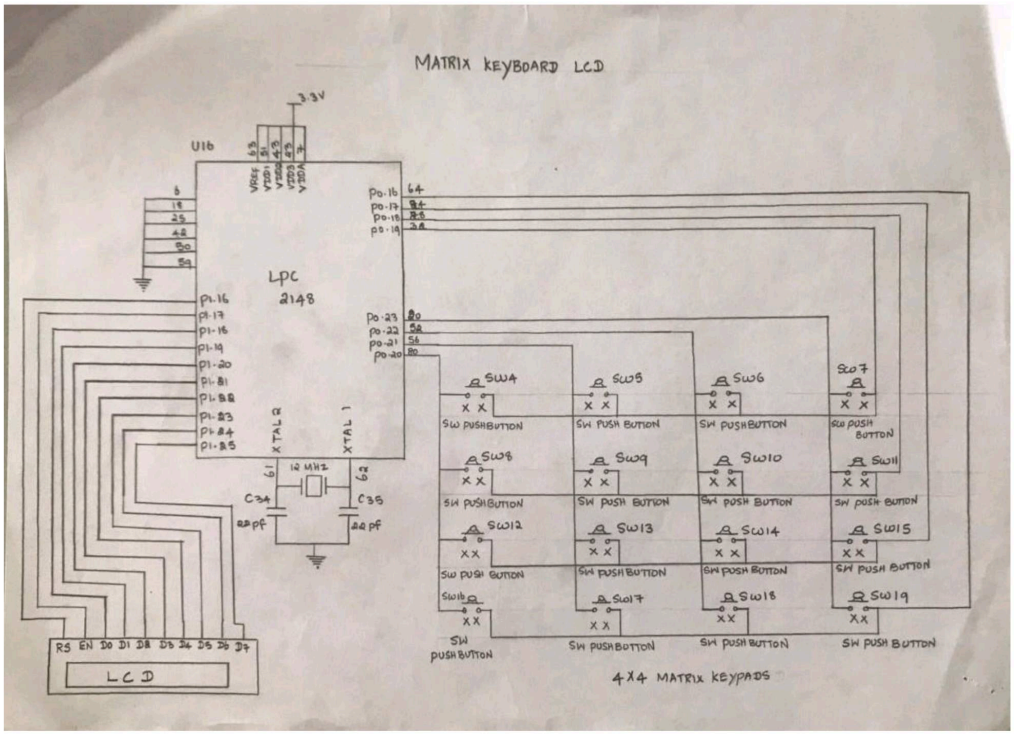
Aim: To program the ARM processor to display the message typed in keypad on LCD.

Apparatus Required: LPC2148 ARM evaluation kit, 2X16 LCD display, 4x4 matrix keypad, KeiluVision, Flash Magic

Flowchart



Interfacing Diagram



Program

```
#include <LPC214x.h>
#include <stdio.h>

#define SW      16

#define RS      1<<16
#define EN      1<<17

void lcd_Init (void);
void lcd_cmd (char);
void lcd_data (char);
void lcd_display (void);
void delay(int);

void Delay_Key(void);
void KeyScan(void);
```

```
unsigned char cmd[8] = {0x33,0x32,0x38,0x0C,0x06,0x01};
unsigned char msg[] = {" MATRIX KEY DEMO"};

unsigned char msg1[]= {" PRESSED KEY :  "};
unsigned char M,N,Row_Data, Col_Data;
unsigned char Msg[4][4] =      { 'C','D','E','F',

                                 '8','9','A','B',

                                 '4','5','6','7',

                                 '0','1','2','3' };

int main(void)
{
        IO1DIR =        0x3FF0000;
        lcd_Init();
        lcd_display();
        while (1)
        {
                Delay_Key();
                KeyScan();
                /* Call KeyScan to Scan Row & Column  */

                if (Row_Data< 4 &&Col_Data< 4)
                {
                        lcd_cmd(0xCE);
                        lcd_data (Msg[Col_Data][Row_Data]);
                        Delay_Key();
                }
        }
}

void Delay_Key(void)
{
        unsigned int i,j;
        for(i=0;i<350;i++)
                for(j=0;j<1234;j++);
}

void KeyScan()
{

        Delay_Key();
        IODIR0 = (0x0F << SW);
        // Configuring Rows as Input && Colum as OutPut     (P0.16 – P0.23)
        IO0PIN = (0xF0 << SW);
```

```
// Push Column Values to LOW so as to get ROW value

while (((IOPIN0>>SW)&0x00F0) == 0xF0);

M = IOPIN0 >> SW;

if (M == 0xE0)
{
        Row_Data = 0;
}
else if (M == 0xD0)
{
        Row_Data = 1;
}
else if (M == 0xB0)
{
        Row_Data = 2;
}
else if (M == 0x70)
{
        Row_Data = 3;
}
else
        Row_Data = 4;

Delay_Key();
Delay_Key();

/*^^^^^^^^^^^^^^^^^^^ Scanning of Column ^^^^^^^^^^^^^^^^^^^^^^^^*/
IOPIN0 =        0x0F << SW;
IODIR0 =        (0xF0 << SW);
// Configure Column as Input and Rows as OutPut (P0.16 – P0.23)

IOPIN0 =        (0x0F << SW);
// Push LOW to Rows to get the Column value of Key Press

while (((IOPIN0>>SW)&0x000F) == 0x0F);

N = (IOPIN0 >> SW);

if (N == 0x0E)
{
        Col_Data = 0;
}
else if (N == 0x0D)
{
        Col_Data = 1;
```

```
        }
        else if (N == 0x0B)
        {
                Col_Data = 2;
        }
        else if (N == 0x07)
        {
                Col_Data = 3;
        }
        else
                Col_Data = 4;

        Delay_Key();
        IOPIN0 =        0xF0 << SW;
        Delay_Key();

}

void lcd_Init(void)                                     // LCD Initialize
{
        unsigned int i;
        for(i=0;i<6;i++)
        {
                lcd_cmd(cmd[i]);
                delay(10);
        }
}

void lcd_cmd(char data)                                 // LCD Command Mode
{
        IO1CLR = 0xFFFFFFFF;
        IO1CLR= RS;
        IOPIN1 = (IOPIN1 | data << 18);
        IOSET1 = EN;
        delay(5);
        IOCLR1= EN;
}

void lcd_data (char data)                       // LCD Data Mode
{
        IO1CLR = 0xFFFFFFFF;
        IOSET1 = RS;
        IOPIN1 = (IOPIN1 | data << 18);
        IOSET1 = EN;
        delay(5);
        IOCLR1= EN;
}
```

```
void lcd_display (void)
{
        unsigned int i;

        lcd_cmd(0x80);                  // Display First Line
        delay(10);

        for(i=0;msg[i]!='\0';i++)
        {
                lcd_data(msg[i]);
                delay(40);
        }

        lcd_cmd(0xC0);                  // Display Second Line
        delay(10);

        for(i=0;msg1[i]!='\0';i++)
        {
                lcd_data(msg1[i]);
                delay(40);
        }
}

void delay(int n)
{
        unsigned int i,j;
        for(i=0;i<n;i++)
        {
                for(j=0;j<12000;j++)
                {;}
        }
}
```

Result:

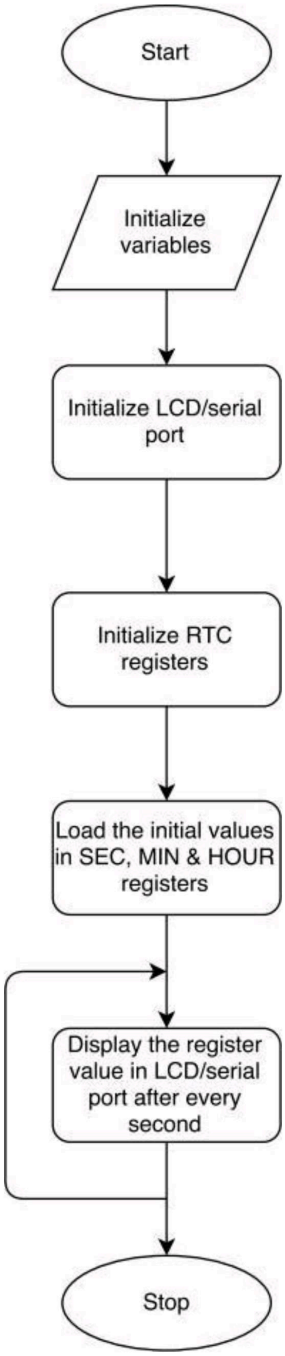
Thus the keyboard and LCD is interfaced with ARM LPC2148.

Expt no 5: REAL TIME CLOCK

Aim: To program the ARM processor to display RTC in LCD and serial port window

Apparatus Required: LPC2148 ARM evaluation kit, KeiluVision, Flash Magic

Flowchart



Program 5a

Real Time Clock with LCD

```
/*Inbuilt RTC*/

#include <lpc214x.h>

unsigned char flag=0;

#define RS      1<<16
#define EN      1<<17

char cmd[8] = {0x33,0x32,0x38,0x0C,0x06,0x01};

void lcd_cmd(unsigned char);
void lcd_data(unsigned char);
void lcd_Init(void);
void lcd_display(void);
void lcd_Convert(unsigned char);
void init_rtc(void);
void delay(int);


int main(void)
{
        VPBDIV = 0x02;
        DelayMs(1000);

        IO1DIR =        0x3FF0000;

        lcd_Init();
        init_rtc();

        while(1)
        {
                                lcd_cmd(0xC4);
                                flag=0;
                                lcd_data(HOUR/10 + '0');
                                lcd_data(HOUR%10 + '0');
                                lcd_data(':') ;
                                lcd_data(MIN/10 + '0');
                                lcd_data(MIN%10 + '0');
                                lcd_data(':');
                                lcd_data(SEC/10 + '0');
                                lcd_data(SEC%10 + '0');
```

```
        }
}

void init_rtc(void)
{
        ILR = 0x01;
        CCR = 0x13;
        CCR = 0x11;
        CIIR = 0x01;
        SEC = 0x00;
        HOUR = 0x00;
        MIN  = 0x00;

}

void lcd_Init(void)                                  // LCD Initialize
{
        unsigned int i;
        for(i=0;i<6;i++)
        {
                lcd_cmd(cmd[i]);
                delay(10);
        }
}

void lcd_cmd(char data)                              // LCD Command Mode
{
        IO1CLR = 0xFFFFFFFF;
        IO1CLR= RS;
        IOPIN1 = (IOPIN1 | data << 18);
        IOSET1 = EN;
        delay(5);
        IOCLR1= EN;
}

void lcd_data (char data)                    // LCD Data Mode
{
        IO1CLR = 0xFFFFFFFF;
        IOSET1 = RS;
        IOPIN1 = (IOPIN1 | data << 18);
        IOSET1 = EN;
        delay(5);
        IOCLR1= EN;
}

void delay(int n)
{
```

```
        unsigned int i,j;
        for(i=0;i<n;i++)
        {
                for(j=0;j<12000;j++)
                {;}
        }
}
```

Program 5b:

Real time clock with Serial port

```
/*Inbuilt RTC*/

#include <lpc214x.h>

void UART0_Init(void);
void UART0_Txmt(unsigned char);
unsigned char UART0_Recv(void);
void DelayMs(unsigned int ms);
void init_rtc(void);

int main(void)
{
        VPBDIV = 0x02;
        DelayMs(1000);

        UART0_Init();
        init_rtc();

        while(1)
        {

                                UART0_Txmt(HOUR/10 + '0');
                                UART0_Txmt(HOUR%10 + '0');
                                UART0_Txmt(':');
                                UART0_Txmt(MIN/10 + '0');
                                UART0_Txmt(MIN%10 + '0');
                                UART0_Txmt(':');
                                UART0_Txmt(SEC/10 + '0');
                                UART0_Txmt(SEC%10 + '0');
                                UART0_Txmt('\r');

        }
}
```

```
void init_rtc(void)
{
        ILR = 0x01;
        CCR = 0x13;
        CCR = 0x11;
        CIIR = 0x01;
        SEC = 0x00;
        HOUR = 0x00;
        MIN  = 0x00;

}

void UART0_Init(void)          /* Initialize Serial Interface       */
{
  PINSEL0       |= 0x00000005;     /* Enable RxD1 and TxD1              */
  U0LCR          = 0x00000083;     /* 8 bits, no Parity, 1 Stop bit     */
  U0DLL          = 0x000000C3;     /* 9600 Baud Rate @ 30MHz VPB Clock  */
  U0LCR          = 0x00000003;     /* DLAB = 0                          */
}


void UART0_Txmt(unsigned char Chr)       /* Write character to Serial Port*/
{
  while (!(U0LSR & 0x20));
  U0THR = Chr;
}

void DelayMs(unsigned int ms)                /*Delay at 30Mhz frequency*/
{
        long i,j;
        for (i = 0; i<ms; i++ )
        for (j = 0; j < 6659; j++ );
}
```

Result:

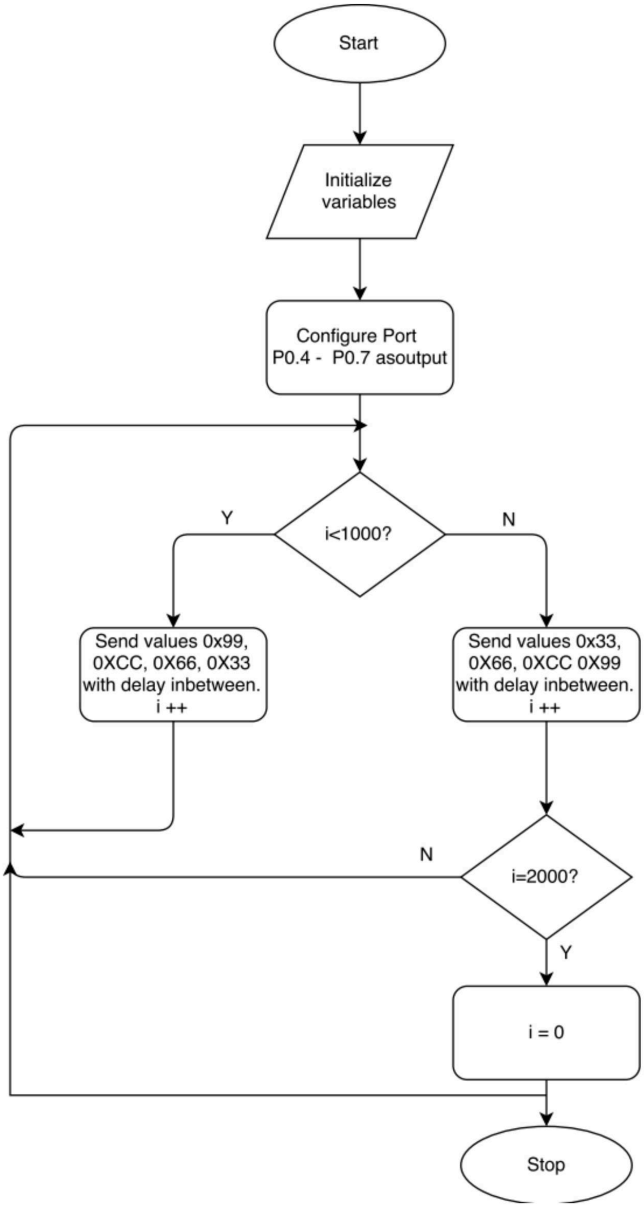
Thus the RTC value is displayed in LCD and serial port using LPC2148.

Expt no: 6 INTERFACING STEPPER MOTOR

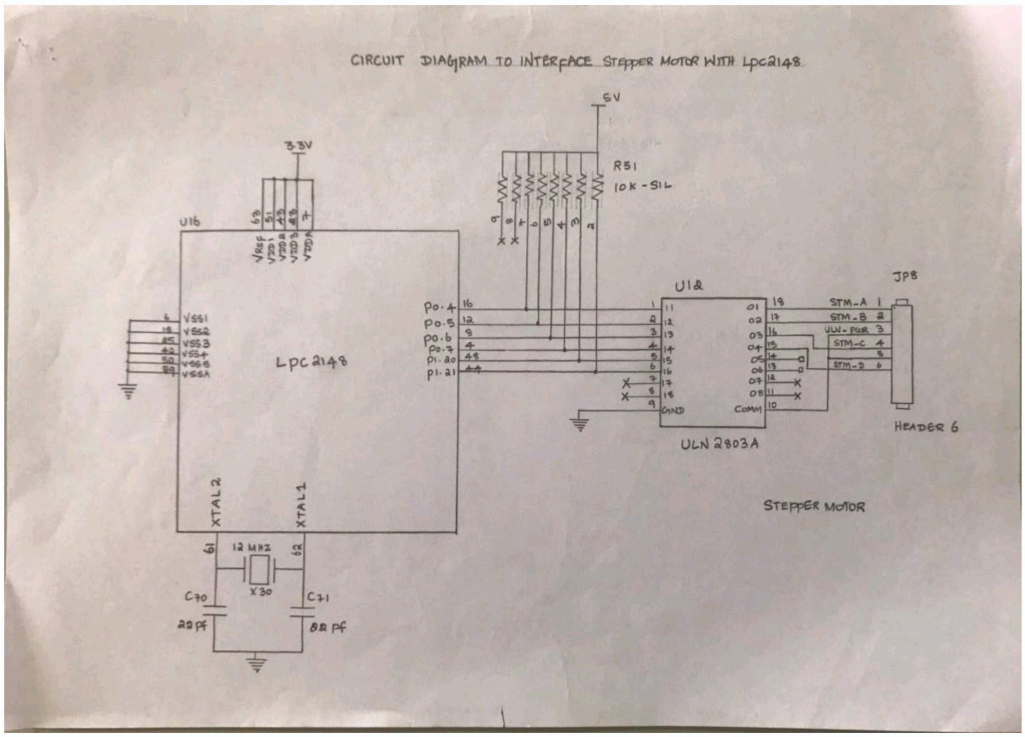
Aim: To interface stepper motor with ARM Processor and program it to rotate in different directions

Apparatus Required: LPC2148 ARM evaluation kit, stepper motor interface board, stepper motor, KeiluVision, Flash Magic

Flowchart



Interfacing Diagram



Program

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


void delay(int n)
{
        int i,j;
        for(i=0;i<n;i++)
        {
                for(j=0;j<0x3FF0;j++)
                {;}
        }
}

int main(void)
{
```

```
        unsigned int i=0;
        IODIR0 = 0x000000F0;                    //Configure P0.4 - P0.7 as Output Pins

        while(1)
        // Loop forever.....
        {
                if(i<1000)
                {
                        IOPIN0=0x99;
                        delay(10);
                        IOPIN0=0xcc;
                        delay(10);
                        IOPIN0=0x66;
                        delay(10);
                        IOPIN0=0x33;
                        delay(10);
                        i++;
                }
                else if(i>=1000 &&i<2000)
                {
                        IOPIN0=0x33;            //Stepper Squence 1001,1100,0110,0011
                        delay(10);
                        IOPIN0=0x66;
                        delay(10);
                        IOPIN0=0xcc;
                        delay(10);
                        IOPIN0=0x99;
                        delay(10);
                        i++;
                }
                else if(i==2000) i=0;
        }
}
```

Result:

Thus the stepper motor is interfaced with ARM controller and is made to rotate in different directions.

Expt no 7: PWM GENERATION USING ARM

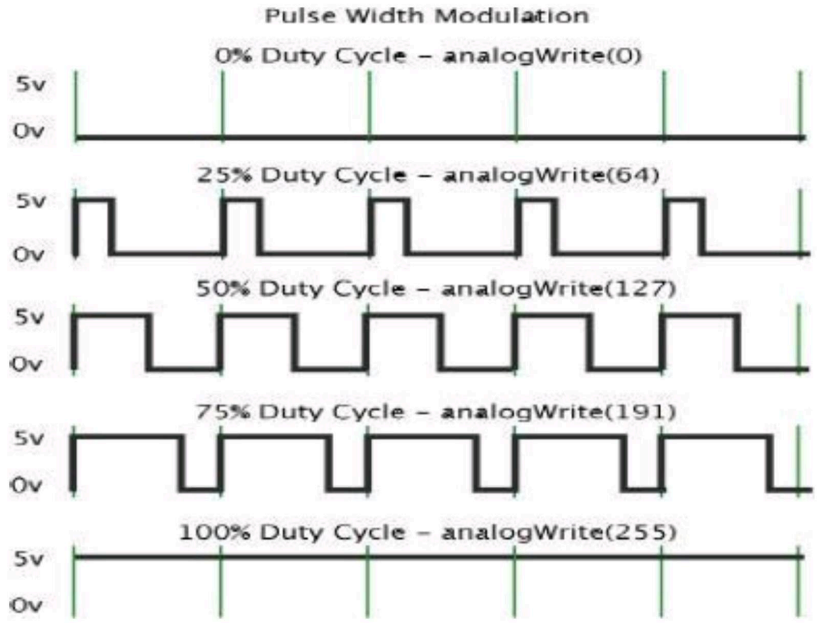
Aim: To program ARM processor to generate PWM using on board features.

Apparatus Required: LPC2148 ARM evaluation kit, KeiluVision, Flash Magic

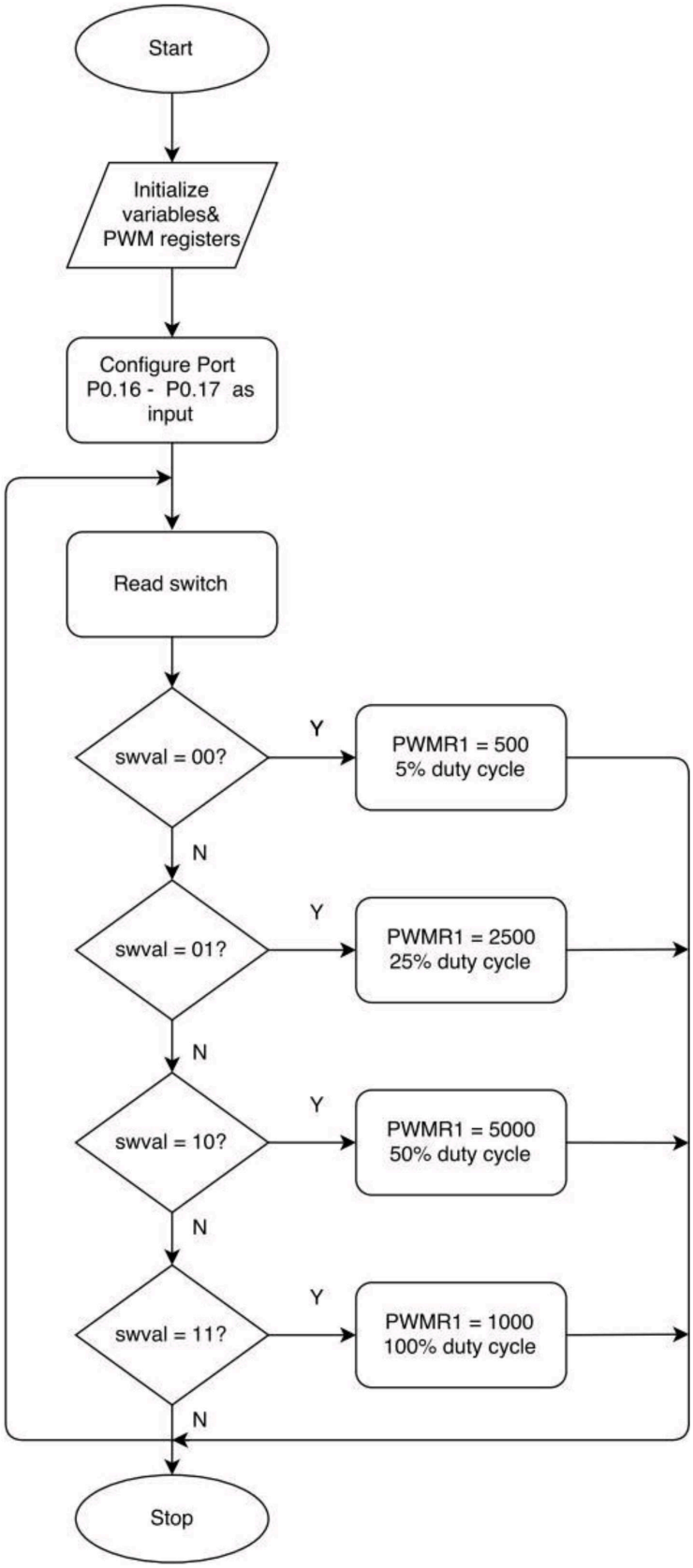
Interfacing PWM

Pulse width modulation (PWM) is a powerful technique for controlling analog circuits with a processor's digital outputs. PWM is employed in a wide variety of applications, ranging from measurement and communications to power control and conversion.

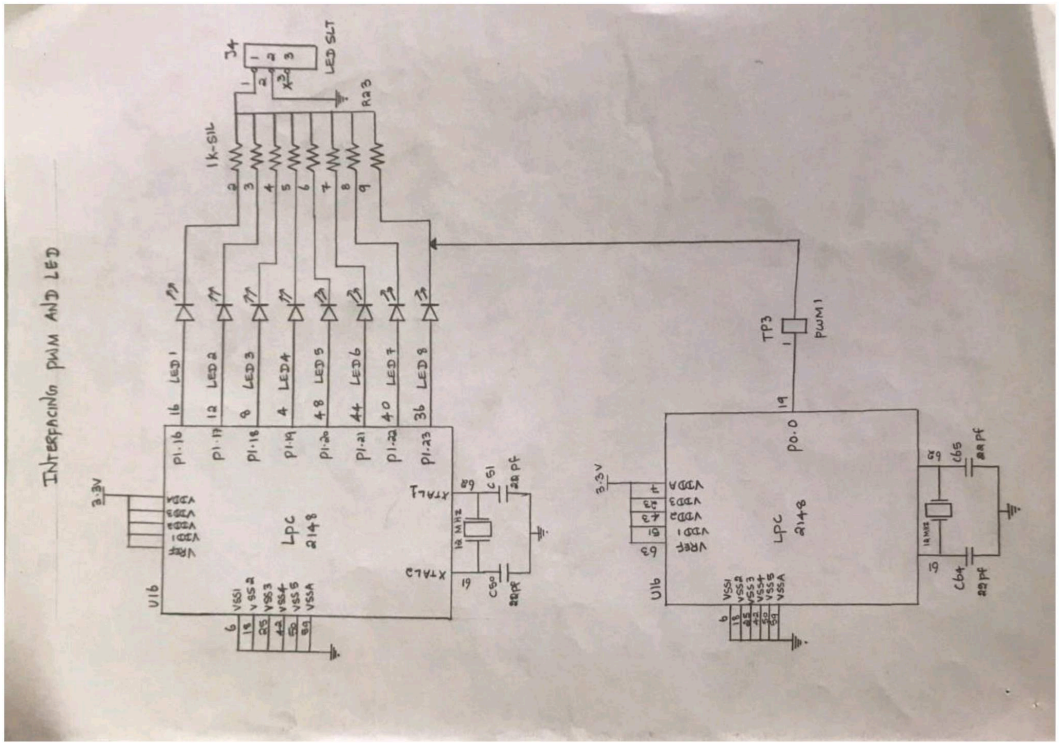
The figure shows four different PWM signals. One is PWM output at a 25% duty cycle. That is, the signal is on for 25% of the period and off the other 75%. Next shows PWM output at 50%, 75% and 100% duty cycles, respectively. These three PWM outputs encode three different analog signal values, at 10%, 50%, and 90% of the full strength.



Flowchart


Start
Initialize variables& PWM registers
Configure Port P0.16 - P0.17 as input
Read switch
swval = 00?
Y
PWMR1 = 500
5% duty cycle
N
swval = 01?
Y
PWMR1 = 2500
25% duty cycle
N
swval = 10?
Y
PWMR1 = 5000
50% duty cycle
N
swval = 11?
Y
PWMR1 = 1000
100% duty cycle
N
Stop

Interfacing Diagram



Program

```
#include <lpc214x.h>

#define PWMPRESCALE 30     //30 PCLK cycles to increment TC by 1 i.e 1 Micro-second

void InitPWM(void);
void DelayMs(unsigned int ms);
unsigned char Read_Switch(void);

unsigned char swval=0;

int main(void)
{
        VPBDIV = 0x02;
        DelayMs(1000);
```

```
        PINSEL1          = 0x00000000;        // PIN P0.16 to P0.17 as GPIO
        IO0DIR          &= 0xFFFCFFFF;                // PIN P0.16 to P0.17 as Input
        InitPWM();                           //Initialize PWM

 while(1)
 {
            swva  = Read_Switch();
            switch(swval)
            {
                case 0: PWMMR1       =  500; PWMLER = (1<<1); break;
                //Frequency (20*(1 /10Mhz)  = 2us  = 500KHz


                case 1: PWMMR1       = 2500; PWMLER = (1<<1); break;
                //Frequency (25*(1 /10Mhz)  = 2.5us = 400KHz

                case 2: PWMMR1       = 5000; PWMLER = (1<<1); break;
                //Frequency (50*(1 /10Mhz)  = 5us  = 200KHz

                case 3: PWMMR1       =10000; PWMLER = (1<<1); break;
                //Frequency (100*  1/10Mhz) = 10us  = 100KHz
            }
        }
}

void InitPWM(void)
{
        PINSEL0 = (1<<1);                            // Select PWM1 output for Pin P0.0
        PWMPCR  = 0x0;
        // Select Sin le Edge PWM - by de  ault its single Edged so this line can be removed
        PWMPR  = P     MPRESCALE-1;                  // 1 micro-second resolution
        PWMMR0 =  10000;                             // 10ms period duration
        PWMMR1  = 2500;
        // 2.5ms - pu lse duration i.e widtl  (Brigtness level)
        PWMMCR  = (1<<1);
        // Reset PWMTC on PWMMR0 ma tch
        PWMLER = (   <<1) | (1<<0);  // up date MR0 and MR1
        PWMPCR =  (1<<9);                                    // enable PWM1 output
        PWMTCR  = (1<<1) ;                                   // Reset PWM TC & PR

        PWMTCR = (  <<0) | (1<<3);  // en able counters and PWM Mode
}

unsigned char Read_  witch(void)
{
        unsigned ch   sw=0;
        sw = (IO0PIN >>16) & 0x00000003
```

```
        return sw;
}

void DelayMs(unsigned int ms)              //Delay at 30Mhz frquency
{
        long i,j;
        for (i = 0; i<ms; i++ )
        for (j = 0; j < 6659; j++ );
}
```

Result:

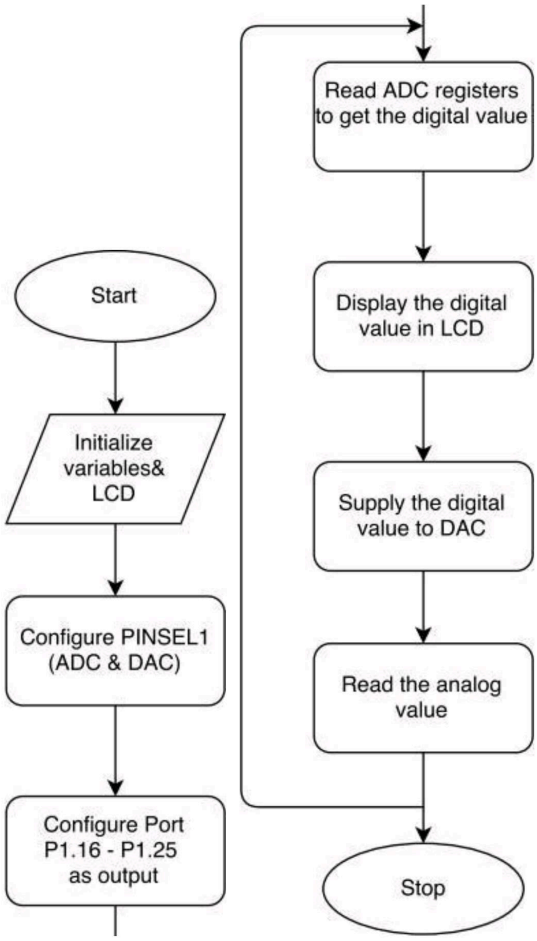
Thus PWM signal is generated using ARM LPC2148.

Expt no 8: INTERFACING ADC AND DAC

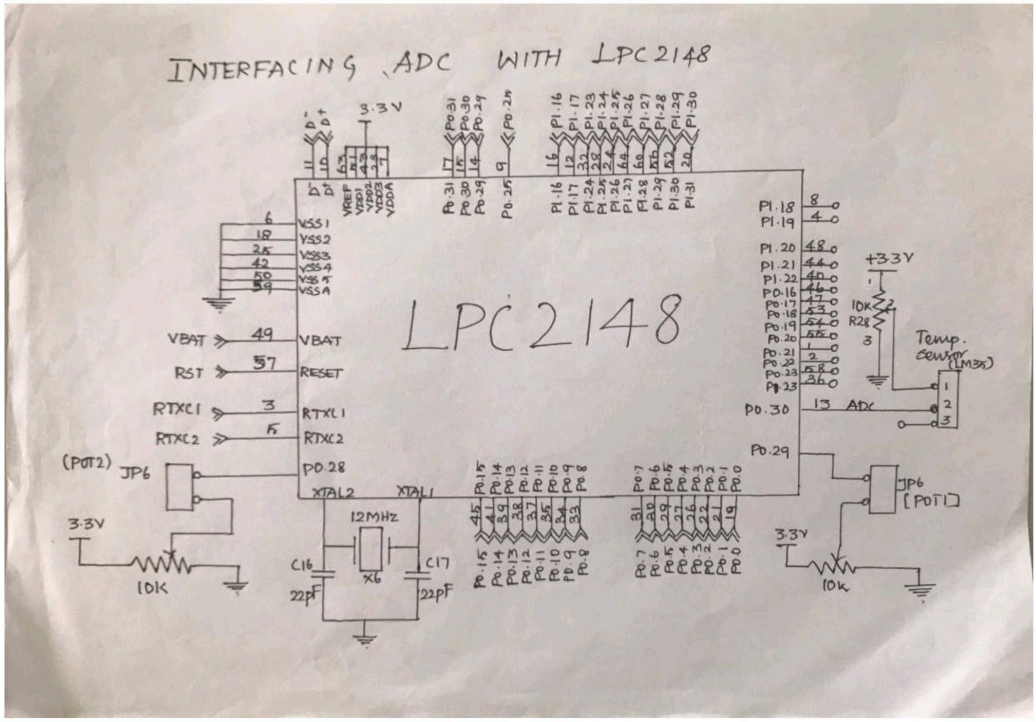
Aim: To Interface ADC and DAC with ARM Processor and display the room temperature.

Apparatus Required: LPC2148 ARM evaluation kit, LM 35 Temperature Sensor, KeiluVision, Flash Magic

Flowchart



Interfacing Diagram



Program

```
#include <lpc214x.h>
#include <string.h>

#define DONE1       1<<1
#define DONE2       1<<2
#define DONE3       1<<3
#define START       1<<24
#define RS          1<<16
#define EN          1<<17


void ADC_Init (void);
void ADC_Conversion(void);
void delay(int n);
void Convert_Data(unsigned int ADC_Val, unsigned char *string);
void DacInit(void);
void DacOut(unsigned int val);
```

```
void lcd_cmd (unsigned char);
void lcd_data (unsigned char);
void lcd_Init (void);
void lcd_display (void);
void delay(int);
void lcd_puts(unsigned char *);

char cmd[8] = {0x33,0x32,0x38,0x0C,0x06,0x01};

unsigned char Msg[] = "POT2  POT1  LM35",ADC_Result[5];
unsigned int ADC1_OUT,ADC2_OUT,ADC3_OUT;

int main(void)
{
		IO1DIR =	0x3FF0000;
		lcd_Init();
		ADC_Init ();
		DacInit();
		lcd_puts(Msg);
		while(1)
		{
			ADC_Conversion();
			DacOut(ADC1_OUT);
		}
}

void ADC_Init ()
{
		VPBDIV	=	0x02;
		PINSEL1	|=	0x01 << 24;		//Enable ADC0.1
		PINSEL1	|=	0x01 << 26;
		PINSEL1	|=	0x01 << 28;
		AD0CR	=	((0xE<<0)|(0x06<<8)|(0x1<<16)|(0x1<<21));
}

void ADC_Conversion()
{
		while ((AD0STAT & DONE1)== 0);
		ADC1_OUT = (AD0DR1 >> 6) & 0x3FF ;
		while ((AD0STAT & DONE2)== 0);
		ADC2_OUT = (AD0DR2 >> 6) & 0x3FF ;
		while ((AD0STAT & DONE3)== 0);
		ADC3_OUT = (AD0DR3 >> 6) & 0x3FF ;

		Convert_Data(ADC1_OUT,ADC_Result);
		lcd_cmd(0xC0);
```

```
                lcd_puts(ADC_Result);
                Convert_Data(ADC2_OUT,ADC_Result);
                lcd_cmd(0xC6);
                lcd_puts(ADC_Result);
                Convert_Data(ADC3_OUT,ADC_Result);
                lcd_cmd(0xCC);
                lcd_puts(ADC_Result);
}

void Convert_Data(unsigned int ADC_Val, unsigned char *string)
{
        string[0] = ADC_Val/1000 + 0x30;
        ADC_Val  = ADC_Val%1000;
        string[1] = ADC_Val/100 + 0x30;
        ADC_Val = ADC_Val%100;
        string[2] = ADC_Val/10 + 0x30;
        string[3] = ADC_Val%10 + 0x30;
        string[4] = '\0';
}

void DacInit(void)
{
   PINSEL1 |= 0x00080000;     //set P0.25 to DAC output
}


void DacOut(unsigned int val)
{
   DACR = (val<< 6) | (0<<16);
}

void lcd_Init(void)                                    // LCD Initialize
{
        unsigned int i;
        for(i=0;i<6;i++)
        {
                lcd_cmd(cmd[i]);
                delay(10);
        }
}

void lcd_cmd(char data)                                // LCD Command Mode
{
        IO1CLR = 0xFFFFFFFF;
        IO1CLR= RS;
        IOPIN1 = (IOPIN1 | data << 18);
        IOSET1 = EN;
```

```
        delay(5);
        IOCLR1= EN;
}

void lcd_data (char data)                    // LCD Data Mode
{
        IO1CLR = 0xFFFFFFFF;
        IOSET1 = RS;
        IOPIN1 = (IOPIN1 | data << 18);
        IOSET1 = EN;
        delay(5);
        IOCLR1= EN;
}

void lcd_puts(unsigned char *string)
{
        while(*string)
                lcd_data(*string++);
}

void delay(int n)
{
        unsigned int i,j;
        for(i=0;i<n;i++)
        {
                for(j=0;j<12000;j++)
                {;}
        }
}
```

Result:

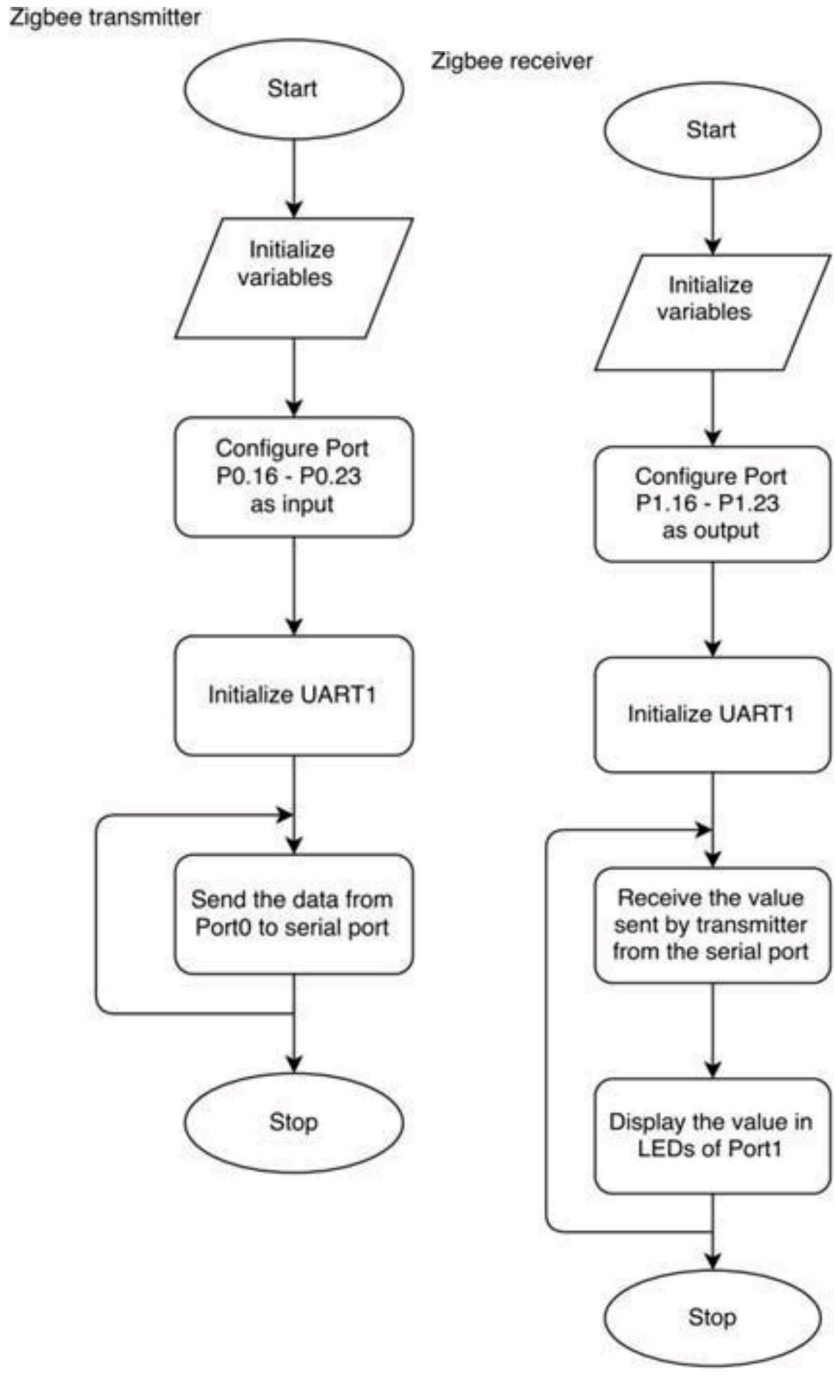
Thus the ADC and DAC module is interfaced with ARM and the temperature is measured.

Expt no 9: IMPLEMENTING ZIGBEE PROTOCOL WITH ARM

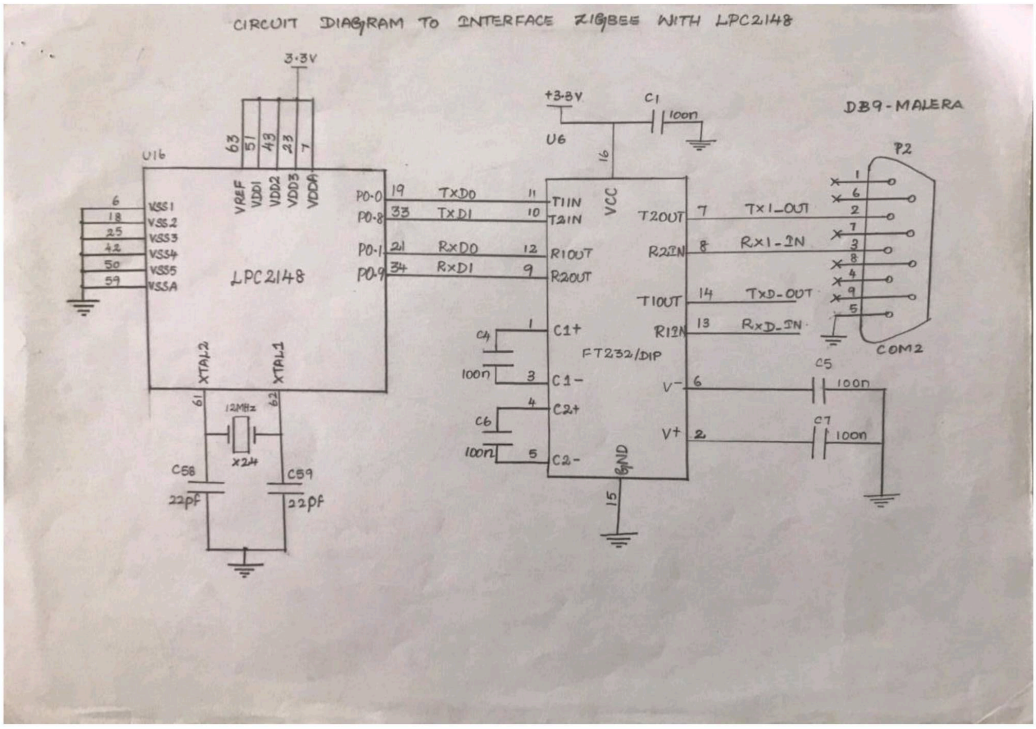
Aim: To implement Zigbee protocol with ARM to transmit and receive data

Apparatus Required: LPC2148 ARM evaluation kit, Zigbee module, KeiluVision, Flash Magic

Flowchart



Interfacing Diagram



Program

Transmitter Program

```
#include <LPC21xx.H>

#define PCLK            30000000

void UART1_Init(void);
void UART1_Txmt(unsigned char);
void DelayMs(unsigned int);

unsigned int SW_RD;

int main(void)
{
        VPBDIV = 0x02;                  //Divide Pclk by two===>clk@12Mhz==30Mhz
        UART1_Init();
```

```
        while(1)
        {
                        SW_RD    = ((IOPIN0 & 0xFF0000)>>16 & 0xFF);
                        UART1_Txmt(SW_RD);
                        DelayMs(100);
        }
}

void UART1_Init(void)           /* Initialize Serial Interface       */
{
 PINSEL0 = 0x00050000;      /* Enable RxD1 and TxD1              */
 U1LCR          = 0x00000083;      /* 8 bits, no Parity, 1 Stop bit     */
 U1DLL          = 0x000000C3;      /* 9600 Baud Rate @ 30MHz VPB Clock */
 U1LCR          = 0x00000003;      /* DLAB = 0                          */
}

void UART1_Txmt(unsigned char Chr)                  /* Write character to Serial Port    */
{
 while (!(U1LSR & 0x20));
 U1THR = Chr;
}

void DelayMs(unsigned int Ms)
        //Delay Definition
{
        int delay_cnst;
        while(Ms>0)
        {
                Ms--;
                for(delay_cnst = 0;delay_cnst<1200;delay_cnst++);
        }
}
```

Receiver Program:

```
#include <LPC21xx.H>

#define PCLK            30000000

void UART1_Init(void);
unsigned char UART1_Recv(void);
void DelayMs(unsigned int);

unsigned char Rec_Val;

int main(void)
```

```
{
        VPBDIV = 0x02;                  //Divide Pclk by two===>clk@12Mhz==30Mhz
        PINSEL2         = 0x00000000;//Configure P1.16 - P1.31 as GPIO
        IO1DIR =        0x00FF0000;
         UART1_Init();
        while(1)
        {
                        Rec_Val = UART1_Recv();
                        IO1PIN  = (Rec_Val<<16);
        }
}

void UART1_Init(void)                           /* Initialize Serial Interface       */
{
  PINSEL0 = 0x00050000;       /* Enable RxD1 and TxD1              */
  U1LCR          = 0x00000083;       /* 8 bits, no Parity, 1 Stop bit     */
  U1DLL          = 0x000000C3;       /* 9600 Baud Rate @ 30MHz VPB Clock  */
  U1LCR          = 0x00000003;       /* DLAB = 0                          */
}

unsigned char UART1_Recv(void)       /* Receive character from Serial Port    */
{
  while (!(U1LSR & 0x01));
  return (U1RBR);
}

void DelayMs(unsigned int Ms)
        //Delay Definition
{
        int delay_cnst;
        while(Ms>0)
        {
                Ms--;
                for(delay_cnst = 0;delay_cnst<1200;delay_cnst++);
        }
}
```

RESULT:

Thus the Zigbee module is interfaced with the ARM controller to transmit and receive data.