1 Purpose

The purpose of this document is to describe how to implement SMBus communication with MLX90614 and STC MCU, also briefly introduces how to measure PWM pulse by using MLX90614 then calculates the temperature. C code is based on STC12C5604AD. The example below gives C programme for reading temperature from MLX90614, writing data into MLX90614 EEPROM to adjust Emissivity and SMBus address, measuring PWM pulse and calculating temperature.

2 C code

2.1 Read Temperature from MLX90614

C programme for reading temperature from MLX90614 is given as an example. For the sake of making programme simple and easily operated, the entire C file is divided into several sub C files. For instance, in this document, the whole C file including:

Main programme **SMBus.c** (Reading temperature from MLX90614, Adjust SMBus address, Emissivity, PWM configuration etc); Sub programme **SMBus_CM.c** (Describing SMBus communication: start_bit, stop_bit, transmit and receive bytes on SMBus); Sub programme **SMBus_OP.c** (Containing reading temperature from MLX90614, writing data into MLX90614 and PEC calculation programme); Sub programme **Delay.c** (Delay a certain period); Sub programme **dec2hex.c** (Convert decimal data to hex data); Sub programme **CalTem.c** (Calculate Temperature based on hex data); Sub programme **digitalLEDs.c** (Show temperature on digital LEDs). These sub programmes are included in head file in order to combine, link all of the programmes.


```
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
```

Main file SMBus.c

```
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
IR Thermometer --- Read Temperature from MLX90614
Describe: This file is C programme for reading temperature from MLX90614 or writing data into MLX90614,
which is based on STC12C5604AD MCU. In this example, reading temperature, calculating temperature
based on hex data and showing temperature on digital LEDs are given, but the last two functions are optional,
customer can make their own choice for the corresponding application.
//-----------------------------------------------------------------------------------------------------------------------//
Head files
//-----------------------------------------------------------------------------------------------------------------------//
#include <STC12C5410AD.H>
#include "stdio.h"
#include "intrins.h"
#include "string.h"
#include "SMBus_CM.h"          //Include SMBus_CM.h (Contain Start, Stop, Send, Receive etc.)
#include "SMBus_OP.h"
//Include SMBus_OP.h (Contain Read data from MLX90614 and Write data // into MLX90614)
#include "digitalLED.h"        //Include digitalLED.h (For showing Temperature on digital LEDs)
#include "CalTem.h"            //Include CalTem.h (For calculating Temperature based on Hex data)
#include "dec2hex.h"           //Include dec2hex.h (For convertering dec to hex)
#include "Delay.h"             //Include Delay
//-----------------------------------------------------------------------------------------------------------------------
Define I/O port and the direction for SMBus signals
//-----------------------------------------------------------------------------------------------------------------------//
#define _SDA_OUTPUT P1M0=0x10; P1M1=0x10;      //Set SDA as Open-drain Output
#define _SDA_INPUT P1M0=0x10; P1M1=0x00;       //Set SDA as Input
```

```
#define _SCL_IO P1M0=0x08; P1M1=0x08;                //Set SCL as Open-drain I/O


sbit SDA = P1^4;                                     //Assign P14 as SDA line
sbit SCL = P1^3;                                     //Assign P13 as SCL line
//-----------------------------------------------------------------------------------------------------------------------
Main function
Function: read object temperature
//-----------------------------------------------------------------------------------------------------------------------
void main()
{
    unsigned char           slaveaddress;
    unsigned long int       DATA;
    unsigned int            *mahm;

      _SCL_IO;
      _SDA_OUTPUT;

      SCL=0;                        //
      Delay(1200);                  //SMBus request，Switch PWM mode to SMBus mode(at least 2ms)
      SCL=1;                        //
      while(1)
      {
            slaveaddress=MEM_READ(0x00,0x2E);
            //Get the slave address which stored in EEPROM "0Eh"
            DATA=MEM_READ(slaveaddress,0x07);
            //Read Object Temperature from MLX90614 RAM 07h
            mahm=CALTEMP(DATA);
            //Calculate the Temperature based on Hex code
             show(mahm,5);
            //Show the Temperature on Digital LEDs
          }
 }
//-----------------------------------------------------------------------------------------------------------------------
```



```
//-----------------------------------------------------------------------------------------------------------------------
```

Sub file SMBus_CM.c

```
//-----------------------------------------------------------------------------------------------------------------------

//-----------------------------------------------------------------------------------------------------------------------
The document contains start_bit, stop_bit, tranmit and receive bytes for SMBus communicaton
//-----------------------------------------------------------------------------------------------------------------------
Head files
//-----------------------------------------------------------------------------------------------------------------------
#include <STC12C5410AD.H>
#include "SMBus_CM.h"
#include "intrins.h"
#include "Delay.h"
//-----------------------------------------------------------------------------------------------------------------------
Define I/O port and the direction for SMBus signals
//-----------------------------------------------------------------------------------------------------------------------
#define _SDA_OUTPUT P1M0=0x10; P1M1=0x10;       //Set SDA as Open-drain Output
#define _SDA_INPUT P1M0=0x10; P1M1=0x00;        //Set SDA as Input
#define _SCL_IO P1M0=0x08; P1M1=0x08;           //Set SCL as Open-drain I/O
sbit SDA = P1^4;                                //Assign P14 as SDA line
sbit SCL = P1^3;                                //Assign P13 as SCL line
```

```
//-------------------------------------------------------------------------------------------------------------------//
Name: start_bit
Function: Generates start condition on SMBus
Comments: Refer to "System Management BUS specification Version 2.0
//-------------------------------------------------------------------------------------------------------------------//
void start_bit()
{
  _SDA_OUTPUT;                          //Set SDA as output
  SDA=1;                                //Set SDA line
  _nop_();_nop_();
  SCL=1;                                //Set SCL line
  Delay(5);       //Generate bus free time between stop and start condition (Tbuf=4.7us min)
  SDA=0;                                //Clear SDA line
  Delay(5);       //Hold time after (Repeated) start condition, after this time, generate the first clock
                                        //Thd:sta=4us min
  SCL=0;                                //Clear SCL line
  _nop_();_nop_();
}
//-------------------------------------------------------------------------------------------------------------------//
Name: stop_bit
Function: Generates stop condition on SMBus
Comments: Refer to "System Management BUS specification Version 2.0
//-------------------------------------------------------------------------------------------------------------------//
void stop_bit()
{
 _SDA_OUTPUT;                           //Set SDA as output
 SCL=0;                                 //Clear SCL line
 Delay(5);
 SDA=0;                                 //Clear SDA line
 Delay(5);
 SCL=1;                                 //Set SCL line
 Delay(5);                              //Stop condition setup time(Tsu:sto=4.0us min)
 SDA=1;                                 //Set SDA line
}
//-------------------------------------------------------------------------------------------------------------------//
Name: send_bit
Function: sends a bit on SMBus
//-------------------------------------------------------------------------------------------------------------------//
void send_bit(unsigned char bit_out)
{

 _SDA_OUTPUT;                           //Set SDA as output to transmit data on SMBus
 if(bit_out==0)                         //Check bit
                                        //Set SDA if bit_out=1
        SDA=0;
 else
        SDA=1;                          //Clear SDA if bit_out=0

 _nop_();                               //
 _nop_();                               //Tsu:dat=250ns minimum
 _nop_();                               //
 SCL=1;                                 //Set SCL line
 Delay(4);                              //High Level of Clock Pulse (10.6us)
 SCL=0;                                 //Clear SCL line
 Delay(4);
}
```

```
//-----------------------------------------------------------------------------------------------------------------------//
Name: receive_bit
Function: receives a bit on SMBus
//-----------------------------------------------------------------------------------------------------------------------//
unsigned char receive_bit()
{
  unsigned char bit_in;
  _SDA_INPUT;                          //Set SDA as input
  SCL=1;                               //Set SCL line
  Delay(2);
  if(SDA==1)                           //Read bit, save it in bit_in
      bit_in=1;
  else
      bit_in=0;
  Delay(2);
  SCL=0;                               //Clear SCL line
  Delay(4);
  return bit_in;
}
//-----------------------------------------------------------------------------------------------------------------------//
Name: slave_ack
Function: Get acknowledgment bit from slave device
Return:  unsigned char ack
          1 - ACK
          0 – NACK
//-----------------------------------------------------------------------------------------------------------------------//
unsigned char slave_ack()
{
   unsigned char ack;
   ack=0;
   _SDA_INPUT;                                 //Set SDA as input
   SCL=1;                                      //Set SCL line
   Delay(2);

   if(SDA==1)                                  //Read bit, save it in ack
       ack=0;
   else
       ack=1;
   Delay(2);
   SCL=0;                                      //Clear SCL line
   Delay(4);
   return ack;
}
//-----------------------------------------------------------------------------------------------------------------------//
TRANSMIT Byte
Name: TX_byte
Function: Sends a byte on SMBus
Parameters: unsigned char TX_buffer (the byte which will be send on the SMBus)
Comments: Sends MSbit first
//-----------------------------------------------------------------------------------------------------------------------//
void TX_byte(unsigned char TX_buffer)
{
   unsigned char Bit_counter;
   unsigned char bit_out;
   for(Bit_counter=8;Bit_counter;Bit_counter--)
   {
```

```
    if(TX_buffer&0x80)
            bit_out=1;                          //If the current bit of TX_buffer is 1, set bit_out
    else
             bit_out=0;                         //Otherwise clear bit_out
          send_bit(bit_out);                    //Send the current bit on SMBus
          TX_buffer<<=1;                        //Get next bit to check
}
//--------------------------------------------------------------------------------------------------------------------//
RECEIVE Byte
Name: RX_byte
Function: Receives a byte on SMBus
Parameters: unsigned char ack_nack (acknowledgment bit)
      0 - Master device sends ACK
      1 - Master device sends NACK
Return:  unsigned char RX_buffer (Received byte on the SMBus)
Comments: MSbit received first
//--------------------------------------------------------------------------------------------------------------------//
unsigned char RX_byte(unsigned char ack_nack)
{
   unsigned char RX_buffer;
   unsigned char Bit_counter;
   for(Bit_counter=8;Bit_counter;Bit_counter--)
   {
        if(receive_bit()==1)                    //Read a bit from the SDA line
                  {
                       RX_buffer<<=1;           //If the bit is HIGH save 1 in RX_buffer
                       RX_buffer|=0x01;
                  }
        else                                    //If the bit is LOW save 0 in RX_buffer
                  {
                       RX_buffer<<=1;
                       RX_buffer&=0xfe;
                            }
         }

         send_bit(ack_nack);                    //Sends acknowledgment bit
         return RX_buffer;
}
//--------------------------------------------------------------------------------------------------------------------//

//--------------------------------------------------------------------------------------------------------------------//
```

Sub file SMBus_OP.c

```
//--------------------------------------------------------------------------------------------------------------------//

//--------------------------------------------------------------------------------------------------------------------//
This document contains C programmes for reading data from MLX90614, writing data into MLX90614
PEC calculation
//--------------------------------------------------------------------------------------------------------------------//
Head files
//--------------------------------------------------------------------------------------------------------------------//
#include <STC12C5410AD.H>
#include "SMBus_CM.h"              //Including SMBus_CM.h
#include "intrins.h"
#include "SMBus_OP.h"
#include "Delay.h"
//--------------------------------------------------------------------------------------------------------------------//
sbit SDA = P1^4;                   //Assign P14 as SDA line
```

```
sbit SCL = P1^3;                    //Assign P13 as SCL line

//-------------------------------------------------------------------------------------------------------------------//
CALCULATE THE PEC PACKET
Name: PEC_cal
Function: Calculate the PEC of received bytes
Parameters: unsigned char pec[], int n
Return: pec[0] - This byte contains calculated crc value
Comments:     Refer to "System Management BUS specification Version 2.0" and " AN "SMBus communication with MLX90614"
//-------------------------------------------------------------------------------------------------------------------//
unsigned char PEC_cal(unsigned char pec[],int n)
{
    unsigned char crc[6];
    unsigned char Bitposition=47;
    unsigned char shift;
    unsigned char i;
    unsigned char j;
    unsigned char temp;
 do{
        crc[5]=0;                           //Load CRC value 0x000000000107
        crc[4]=0;
        crc[3]=0;
        crc[2]=0;
        crc[1]=0x01;
        crc[0]=0x07;
        Bitposition=47;                     //Set maximum bit position at 47
        shift=0;

        //Find first 1 in the transmitted bytes

        i=5;                                //Set highest index (package byte index)
        j=0;                                //Byte bit index, from lowest
        while((pec[i]&(0x80>>j))==0 && (i>0))
       {
          Bitposition--;
          if(j<7)
              {
                j++;
             }
          else
              {
               j=0x00;
               i--;
             }
         }//End of while, and the position of highest "1" bit in Bitposition is calculated

        shift=Bitposition-8;                //Get shift value for CRC value
                                            //Shift CRC value left with "shift" bits
        while(shift)
           {
           for(i=5;i<0xFF;i--)
                {
                if((crc[i-1]&0x80) && (i>0))     //Check if the MSB of the byte lower is "1"
                     {                           //Yes - current byte + 1
                     temp=1;                     //No - current byte + 0
                    }                            //So that "1" can shift between bytes
```

```
                else
                    {
                    temp=0;
                    }
                crc[i]<<=1;
                crc[i]+=temp;
              }
              shift--;
           }
         //Exclusive OR between pec and crc

         for(i=0;i<=5;i++)
           {
              pec[i]^=crc[i];
            }
                    }while(Bitposition>8);
     return pec[0];
}
//------------------------------------------------------------------------------------------------------------------------//
READ DATA FROM RAM/EEPROM
Name: MEM_READ
Function: Read the data from MLX90614 with given slave address and command
Parameters: unsigned char slave_addR (slave address)
            unsigned char cmdR (command)
Return: unsigned long int Data
//------------------------------------------------------------------------------------------------------------------------//
unsigned long int MEM_READ(unsigned char slave_addR, unsigned char cmdR)
{
          unsigned char DataL;                    //
          unsigned char DataH;                    //Data packets from MLX90614
          unsigned char PEC;                      //
          unsigned long int Data;                 //Register value returned from MLX90614
          unsigned char Pecreg;                   //Calculated PEC byte storage
          unsigned char arr[6];                   //Buffer for the sent bytes
          unsigned char ack_nack;
          unsigned char SLA;

          SLA=(slave_addR<<1);
  begin:
          start_bit();                            //Send start bit
          TX_byte(SLA);                           //Send slave address, write
          if(slave_ack()==0)
          {
             stop_bit();
             goto begin;
          }                                       //Send command
          TX_byte(cmdR);
          if(slave_ack()==0)
          {
             stop_bit();
            goto begin;
          }
                                                  //Send Repeated start bit
          start_bit();                            //Send slave address, read
          TX_byte(SLA+1);
          if(slave_ack()==0)
          {
```

```
            stop_bit();
            goto begin;
          }

          DataL=RX_byte(0);                                    //
                                                               //Read two bytes data
          DataH=RX_byte(0);                                    //

          PEC=RX_byte(ack_nack);                               //Read PEC from MLX90614
          if(ack_nack==1)                                      //Master sends ack or nack

         //This depends on the pec calculation, if the PEC is not correct, send nack and goto begin
          {
            stop_bit();
            goto begin;
          }                                                    //Send stop bit
          stop_bit();

         arr[5]=(SLA);
         arr[4]=cmdR;
         arr[3]=(SLA+1);
         arr[2]=DataL;
         arr[1]=DataH;
         arr[0]=0;

          Pecreg=PEC_cal(arr,6);                               //Calculate CRC
          if(PEC==Pecreg)
            {
              ack_nack=0;
             }
          else
            {
              ack_nack=1;
             }
           Data=(DataH*256)+DataL;
           return Data;
}
//-------------------------------------------------------------------------------------------------------------------------------//
WRITE DATA INTO MLX90614 EEPROM
Name: EEPROM_WRITE
Function: Write the data into MLX90614 with given slave address, command and corresponding data
Parameters: unsigned char slave_addW (slave address)
        unsigned char cmdW     (command)
        unsigned char DataL
        unsigned char DataH
//-------------------------------------------------------------------------------------------------------------------------------//
void EEPROM_WRITE(unsigned char slave_addW,unsigned char cmdW,unsigned char DataL,unsigned char
DataH)
{
    unsigned char Pecreg;                                  //Calculated PEC byte storage
    unsigned char SLA;
    unsigned char arr[6];                                  //Buffer for the transmitted bytes

    SLA=(slave_addW<<1);

    arr[5]=0;
    arr[4]=SLA;
```

```
    arr[3]=cmdW;
    arr[2]=DataL;
    arr[1]=DataH;
    arr[0]=0;
    Pecreg=PEC_cal(arr,6);

  begin:
    start_bit();                                    //Send start bit
    TX_byte(SLA);                                   //Send slave address, write
    if(slave_ack()==0)
    {
          stop_bit();
          goto begin;
    }
    TX_byte(cmdW);                                  //Send command
    if(slave_ack()==0)
    {
          stop_bit();
          goto begin;
    }
    TX_byte(DataL);                                 //Send Low Data byte
    if(slave_ack()==0)
    {
          stop_bit();
          goto begin;
    }
    TX_byte(DataH);                                 //Send High Data byte
    if(slave_ack()==0)
    {
          stop_bit();
          goto begin;
    }
    TX_byte(Pecreg);                                //Send PEC
    if(slave_ack()==0)
    {
          stop_bit();
          goto begin;
    }
    stop_bit();                                     //Send stop bit
    Delay(200);                                     //Wait 5ms

}
//-----------------------------------------------------------------------------------------------------------------------//
```



```
//-----------------------------------------------------------------------------------------------------------------------//
```

Sub file Delay.c

```
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
Delay function
//-----------------------------------------------------------------------------------------------------------------------//
#include <STC12C5410AD.H>
#include "intrins.h"
#include "Delay.h"
//-----------------------------------------------------------------------------------------------------------------------//
```

Name: Delay
Function: Delay for a certain time

```
Parameters: N, means delay time is about N times of MCU machine cycle
Comments: Machine cycle is equal to 12 times of clock cycle (1/Fosc), For STC12C5604AD,
Fosc=11.0592MHz
//-----------------------------------------------------------------------------------------------------------//
void Delay(unsigned int N)
{
   unsigned int i;
   for(i=0;i<N;i++)
   _nop_();
}
//-----------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------//
```

Sub file dec2hex.c

```
//-----------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------//
Convert decimal to hex
//-----------------------------------------------------------------------------------------------------------//
#include "dec2hex.h"
#include "stdio.h"
//-----------------------------------------------------------------------------------------------------------//
Name: dec2hex
Function: Converter dec code to hex code
Parameters: float e (New emissivity)
Return:  unsigned int c
         c is the head address of c[4]
//-----------------------------------------------------------------------------------------------------------//
unsigned int *dec2hex(float e)
{
        long int a=(e*65535)+0.5;
        int j,m=0,i=0;
        unsigned int b[16];
        unsigned int c[4],t;
        b[15]=0x0f;
        b[14]=0x0e;
        b[13]=0x0d;
        b[12]=0x0c;
        b[11]=0x0b;
        b[10]=0x0a;
        b[9]=0x09;
        b[8]=0x08;
        b[7]=0x07;
        b[6]=0x06;
        b[5]=0x05;
        b[4]=0x04;
        b[3]=0x03;
        b[2]=0x02;
        b[1]=0x01;
        b[0]=0;

        while(a!=0)
        {
          j=a%16;                          //Modulus operator
          c[i++]=b[j];
          a=a/16;                          //Remainder operator
          m++;
```

```
        }
        for(i=0;i<m/2;i++)
        {                                   //
            t=c[i];                         //Reverse order of array c[4]
            c[i]=c[m-i-1];                  //
            c[m-i-1]=t;
        }

        return c;                           //Return the head address of c[4]
}
//-----------------------------------------------------------------------------------------------------------------//
```

Sub file CalTem.c

```
//-----------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------//
Calculate Temperature based on hex data
//-----------------------------------------------------------------------------------------------------------------//
#include "CalTem.h"
//-----------------------------------------------------------------------------------------------------------------//
Name: CALTEMP
Function: Calculate Temperature
Parameters: unsigned long int TEMP(Data read from MLX90614)
Return:  unsigned int mah
         mah is the head address of mah[5]
Comments: The equation for conversion Hex code to Temperature data is T= (Data)*0.02-273.15
//-----------------------------------------------------------------------------------------------------------------//
unsigned int *CALTEMP(unsigned long int TEMP)
{
    unsigned long int T;
    unsigned int a,b;
    unsigned int A4,A5,A6,A7,A8;
    unsigned int mah[5];
    T=TEMP*2;
    if(T>=27315)
        {
          T=T-27315;
          a=T/100;
          b=T-a*100;
          if(a>=100)
             {
               A4=a/100;
               a=a%100;
               A5=a/10;
               a=a%10;
               A6=a;
             }
          else if(a>=10)
             {
               A4=0;
               A5=a/10;
               a=a%10;
               A6=a;
             }
          else
             {
```

```
                A4=0;
                A5=0;
                A6=a;
               }
            if(b>=10)
               {
                A7=b/10;
                b=b%10;
                A8=b;
               }
             else
               {
                A7=0;
                A8=b;
               }
          }
       else
          {
            T=27315-T;
            a=T/100;
            b=T-a*100;
            A4=9;
            if(a>=10)
               {
                 A5=a/10;
                 a=a%10;
                 A6=a;
               }
             else
               {
                 A5=0;
                 A6=a;
               }
             if(b>=10)
               {
                 A7=b/10;
                 b=b%10;
                 A8=b;
               }
             else
               {
                 A7=0;
                 A8=b;
               }
          }

          mah[4]=A4;
          mah[3]=A5;
          mah[2]=A6;
          mah[1]=A7;
          mah[0]=A8;
          return mah;
}
```

```
//-----------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------//
```

Sub file digitalLED.c

```
//-----------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------//
Show temperature on digital LEDs
//-----------------------------------------------------------------------------------------------------------//
#include <STC12C5410AD.H>
#include "digitalLED.h"
//-----------------------------------------------------------------------------------------------------------//
Define I/O port to control digital LEDs
//-----------------------------------------------------------------------------------------------------------//
sbit a0 = P1^7;                 //Assign P17 as a0 line
sbit a1 = P1^5;                 //Assign P15 as a1 line
sbit a2 = P1^6;                 //Assign P16 as a2 line
//-----------------------------------------------------------------------------------------------------------//
Name: show
Function: show the temperature on digital LEDs
Parameters: unsigned int mahh[],int number
Comments: show the numbers which stored in mahh[5]
//-----------------------------------------------------------------------------------------------------------//
void show(unsigned int mahh[],int number)
{
unsigned char yanshicon;
unsigned char weicon;
unsigned char code LED_tab[10]={
                                                0xFC,           //0
                                                0x60,           //1
                                                0xDA,           //2
                                                0xF2,           //3
                                                0x66,           //4
                                                0xB6,           //5
                                                0xBE,           //6
                                                0xE0,           //7
                                                0xFE,           //8
                                                0xF6,           //9

                        };
        P2=0;                                   //Turn-off display
        a0=1;
        a1=1;
        a2=1;
        //a2,a1,a0 control the display position, when it is changed from 0-7
        //the corresponding digital LEDs is from 1-8

        while(1)
        {
                if(++yanshicon>200)             //Execute once every 200 scan cycle
                {
                        yanshicon=0;
                   if(++weicon>7) weicon=0;     //Rotate the display position within the range from 0-7

                        P2=0;                   //Turn-off display
                        switch(weicon)
                        {
```

```
                    case 0:          //The content of first display position
                      P2=0;          //Turn-off display
                      a0=0;
                      a1=0;
                      a2=0;
                    break;
                    case 1:          //The content of second display position
                      P2=0;          //Turn-off display
                      a0=1;
                      a1=0;
                      a2=0;
                    break;
                    case 2:          //The content of third display position
                      P2=0;          //Turn-off display
                      a0=0;
                      a1=1;
                      a2=0;
                    break;
                    case 3:                    //The content of fourth display position
                      P2=LED_tab[mahh[4]];     //Display mahh[4]
                      a0=1;
                      a1=1;
                      a2=0;
                    break;
                    case 4:                    //The content of fifth display position
                      P2=LED_tab[mahh[3]];     //Display mahh[3]
                      a0=0;
                      a1=0;
                      a2=1;
                    break;
                    case 5:                    //The content of sixth display position
                      P2=LED_tab[mahh[2]];     //Display mahh[2]
                      a0=1;
                      a1=0;
                      a2=1;
                    break;
                    case 6:                    //The content of seventh display position
                      P2=LED_tab[mahh[1]];     //Display mahh[1]
                      a0=0;
                      a1=1;
                      a2=1;
                    break;
                    case 7:                    //The content of eighth display position
                      P2=LED_tab[mahh[0]];     //Display mahh[0]
                      a0=1;
                      a1=1;
                      a2=1;
                    break;
                    default:

                    break;
                }
            }
        }
}

//-------------------------------------------------------------------------------------------------------------------//
```

2.2 Write data into MLX90614 EEPROM (Adjust Emissivity, SMBus address、PWMCTRL Configuration)

Main file is also used for adjusting Emissivity, SMBus address, PWMCTRL configuration etc. The corresponding C programmes are given below. In order to implement these functions, main function (Read temperature from MLX90614) which in main file should be replaced by new function (Change Emissivity, SMBus address).

```
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
Function：Adjust Emissivity
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
void main()
{
        unsigned char slaveaddress;
        unsigned int *Emv;                  //Define a pointer to point integral variable
        unsigned int EmvLO;
        unsigned int EmvHI;
        float Emissivity=0.5;               //Given a new Emissivity
        Emv=dec2hex(Emissivity);
       //Call subroutine (Converter dec code to hex code) return array head address

       EmvLO=(*(Emv+2)<<4)+*(Emv+3);        //Load New emissivity Low byte
       EmvHI=(*(Emv+0)<<4)+*(Emv+1);        //Load New emissivity High byte

       _SCL_IO;
       _SDA_OUTPUT;

       SCL=0;                               //
       Delay(1200);                         //SMBus request，Switch PWM mode to SMBus mode(at least 2ms)
       SCL=1;                               //
       slaveaddress=MEM_READ(0x00,0x2E);
       //Get the slave address which stored in EEPROM "0Eh"

        EEPROM_WRITE(slaveaddress,0x24,0x00,0x00);
        //Write 0x0000 into EEPROM "04h"

        EEPROM_WRITE(slaveaddress,0x24,EmvLO,EmvHI);
        //Write New emissivity into EEPROM "04h"
        //Need repower to active it
}
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
Function：Adjust SMBus address
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
void main()
{
      unsigned char slaveaddress;
      unsigned int DataLO;
      unsigned int DataHI;
      DataLO=0x5A;                          //Load New slave address Low byte
```

```
        DataHI=0x00;                        //Load New slave address High byte

        _SCL_IO;
        _SDA_OUTPUT;

        SCL=0;                              //
        Delay(1200);                        //SMBus request，Switch PWM mode to SMBus mode(at least 2ms)
        SCL=1;                              //

       slaveaddress=MEM_READ(0x00,0x2E);
        //Get the slave address which stored in EEPROM "0Eh"

        EEPROM_WRITE(slaveaddress,0x2E,0x00,0x00);
        //Write 0x0000 into EEPROM "0Eh"

        EEPROM_WRITE(slaveaddress,0x2E,DataLO,DataHI);
        //Write New slave address into EEPROM "0Eh"
        //Need repower to active it
}
//-------------------------------------------------------------------------------------------------------------------//

//-------------------------------------------------------------------------------------------------------------------//
```

Function：Change PWMCTRL configuration, set as PWM output mode

```
//-------------------------------------------------------------------------------------------------------------------//

//-------------------------------------------------------------------------------------------------------------------//
 void main()
{
        unsigned char slaveaddress;
        unsigned char PWMCTRLO;
        unsigned char PWMCTRHI;
        PWMCTRLO=0x07;
 //Load New PWMCTRL Low byte (0x07 select single PWM mode,enable PWM,and SDA pin as Push-Pull)
        PWMCTRHI=0x02;
        //Load New PWMCTRL High byte(0x02 means PWM period is 1.024ms*1 and no PWM repetition)

        _SCL_IO;
        _SDA_OUTPUT;

        SCL=0;              //
        Delay(1200);        //SMBus request time for switching PWM mode to SMBus mode (At least 2ms)
        SCL=1;              //

        slaveaddress=MEM_READ(0x00,0x2E);
        //Get the slave address which stored in EEPROM "0Eh"

        EEPROM_WRITE(slaveaddress,0x22,0x00,0x00);
        //Write 0x0000 into EEPROM "02h"

       EEPROM_WRITE(slaveaddress,0x22,PWMCTRLO,PWMCTRHI);
       //Write New PWMCTRL into EEPROM "02h"
       //Need repower to active
}
```

```
//-----------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------//
Function：Change PWMCTRL configuration, set as SMBus output mode
//-----------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------//
void main()
{
        unsigned char slaveaddress;
        unsigned char PWMCTRLO;
        unsigned char PWMCTRHI;
        PWMCTRLO=0x01;
 //Load New PWMCTRL Low byte(0x01 select single PWM mode,disable PWM,and SDA pin as Open-Drain)
        PWMCTRHI=0x02;
        //Load New PWMCTRL High byte(0x02 means PWM period is 1.024ms*1 and no PWM repetition)

        _SCL_IO;
        _SDA_OUTPUT;

        SCL=1;
        Delay(10);

       SCL=0;               //
       Delay(1200);         //SMBus request time for switching PWM mode to SMBus mode (At least 2ms)
       SCL=1;               //

      //After switching SMBus mode, change PWMCTRL configuration (SMBus), then after power off/on, the
device will work in SMBus mode

        slaveaddress=MEM_READ(0x00,0x2E);
       EEPROM_WRITE(slaveaddress,0x22,0x00,0x00);        //Write 0x0000 into EEPROM "02h"

       EEPROM_WRITE(slaveaddress,0x22,PWMCTRLO,PWMCTRHI);
       //Write New PWMCTRL into EEPROM "02h"
       //Need repower to active
}
```

2.3 The corresponding .h files

```
//-----------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------//
SMBus_CM.h
//-----------------------------------------------------------------------------------------------------------------//
Head file protection
//-----------------------------------------------------------------------------------------------------------------//
#ifndef SMBUS_CM_H
#define SMBUS_CM_H
//-----------------------------------------------------------------------------------------------------------------//
Function declaration
//-----------------------------------------------------------------------------------------------------------------//
void start_bit();
void stop_bit();
void send_bit(unsigned char bit_out);
```

```
unsigned char receive_bit();
unsigned char slave_ack();
void TX_byte(unsigned char TX_buffer);
unsigned char RX_byte(unsigned char ack_nack);

#endif
//---------------------------------------------------------------------------------------------------------//
```

```
//---------------------------------------------------------------------------------------------------------//
```

SMBus_OP.h

```
//---------------------------------------------------------------------------------------------------------//
```

Head file protection

```
//---------------------------------------------------------------------------------------------------------//
#ifndef SMBUS_OP_H
#define SMBUS_OP_H
//---------------------------------------------------------------------------------------------------------//
```

Function declaration

```
//---------------------------------------------------------------------------------------------------------//
unsigned char PEC_cal(unsigned char pec[],int n);
unsigned long int MEM_READ(unsigned char slave_addR,unsigned char cmdR);
void EEPROM_WRITE(unsigned char slave_addW,unsigned char cmdW,unsigned char DataL,unsigned char
DataH);

#endif
//---------------------------------------------------------------------------------------------------------//
```

```
//---------------------------------------------------------------------------------------------------------//
```

Delay.h

```
//---------------------------------------------------------------------------------------------------------//
```

Function declaration

```
//---------------------------------------------------------------------------------------------------------//
void Delay(unsigned int N);
//---------------------------------------------------------------------------------------------------------//

//---------------------------------------------------------------------------------------------------------//
```

dec2hex.h

```
//---------------------------------------------------------------------------------------------------------//
```

Function declaration

```
//---------------------------------------------------------------------------------------------------------//
unsigned int *dec2hex(float e);

//---------------------------------------------------------------------------------------------------------//

//---------------------------------------------------------------------------------------------------------//
```

CalTem.h

```
//---------------------------------------------------------------------------------------------------------//
```

Function declaration

```
//---------------------------------------------------------------------------------------------------------//
extern unsigned int *CALTEMP(unsigned long int TEMP);
//---------------------------------------------------------------------------------------------------------//

//---------------------------------------------------------------------------------------------------------//
```

digitalLED.h

```
//---------------------------------------------------------------------------------------------------------//
```

Function declaration

```
//---------------------------------------------------------------------------------------------------------//
```

```
extern void show(unsigned int mahh[],int number);
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
```

3 Calculate Temperature based on PWM mode

MLX90614 can be set as PWM output mode, in the example, STC MCU timer is utilized for measuring PWM output. C code includes PWM measurement, calculation and showing temperature on digital LEDs. (Note: MLX90614 output is set as single PWM mode, PWM period is 1.024ms, and the result is object temperature.) Please refer to application note – Read PWM from MLX90614 and calculate temperature with PIC 18 MCU.

```
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
```

Main file PWM.c

```
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
```

Head files

```
//-----------------------------------------------------------------------------------------------------------------------//
#include <STC12C5410AD.H>
#include "stdio.h"
#include "math.h"
#include "intrins.h"
#include "string.h"
#include "PWM_display.h"          //Including PWM_display.h (showing temperature on digital LEDs)
//-----------------------------------------------------------------------------------------------------------------------//
```

Macro definition of I/O ports

```
//-----------------------------------------------------------------------------------------------------------------------//
sbit P3_3=P3^3;
//utilize MCU external interrupt source 1（INT1）to measure PWM pulse on SDA pin of MLX90614
//-----------------------------------------------------------------------------------------------------------------------//
//-----------------------------------------------------------------------------------------------------------------------//
```

Function: Calculate Temperature according PWM pulse width and period（Duty cycle）

```
//-----------------------------------------------------------------------------------------------------------------------//
int Calculate(unsigned int t1pwm, unsigned int T1pwm)
{

  int DC;
  long int T;
  int T1;
  int K;
  int T1max=120;                          //The maximum object temperature, saved in EEPROM 00h
  int T1min=-20;                          //The minimum object temperature, saved in EEPROM 01h
  K=2*(T1max-T1min);

  DC=t1pwm*100000/T1pwm;
  //Calculate Duty cycle, times 100,000 shifts the fixed point 5 position to the right

 T=(DC-0.125*100000)*K+T1min*100000;
 // Tout = [2(DC - 0.125)(Tmax - Tmin)] + Tmin  is used to calculate object temperature, DC is duty cycle
  T1=T/1000;
  //To truncate the resolution to 0.01℃, a division by 1000 is done on the results
  return T1;
}
```

```
//-----------------------------------------------------------------------------------------------------------------------//
```

Name：main

Function：Use STC MCU external interrupt source port 1、Timer 0 and 1 to measure PWM pulse width and period

```
//-----------------------------------------------------------------------------------------------------------------------//
void main(void)
{
  unsigned int A;
  unsigned int B;
  unsigned int C;
  unsigned int D;
  unsigned int Data1;
  unsigned int Data2;
  int Data;

  TMOD=0x19;                    //Set timer 0（GATE=1）and Timer 1（ GATE=0）work in mode 1
  TH0=0x00;                     //Clear TH0, TL0
  TL0=0x00;
  ET0=1;                        //Open Timer 0 interrupt enable
  ET1=1;                        //Open Timer 1 interrupt enable
  EA=1;                         //Open overall interrupt enable
  TL1=0x00;                     //Clear TH1, TL1
  TH1=0x00;
  EX1=0;                        //Turn-off INT1 interrupt

while(P3_3==1)                  //Wait INT1 low pulse
{;}
while(P3_3==0)                  //Wait INT1 high pulse
{;}
TR0=1;                          //Open timer 0
TR1=1;                          //Open timer 1
while(P3_3==1)                  //Wait INT1 low pulse
{;}
TR0=0;                          //Clear timer 0
C=TL0;                          //T0 Low byte sends to C
D=TH0;                          //T0 High byte sends to D
while(P3_3==0)                  //Wait INT1 high pulse
{;}
TR1=0;                          //Clear timer 1
A=TL1;                          //T1 Low byte sends to A
B=TH1;                          //T1 High byte sends to B
Data2=(D<<8)+C;
Data1=(B<<8)+A;

Data=Calculate(Data2,Data1);    //Calculate temperature based on counter value
display(Data);                  //Show temperature
}

//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
```

Sub file PWM_display.c

```
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
```

Head file

```
//-----------------------------------------------------------------------------------------------------------------------//
#include <STC12C5410AD.H>
#include "Intrins.h"
#include "PWM_display.h"
//-----------------------------------------------------------------------------------------------------------------------//
```

Macro definition of I/O ports

```
//-----------------------------------------------------------------------------------------------------------------------//
sbit a0 = P1^7;                 //Assign P17 as a0 line
sbit a1 = P1^5;                 //Assign P15 as a1 line
sbit a2 = P1^6;                 //Assign P16 as a2 line
//-----------------------------------------------------------------------------------------------------------------------//
unsigned char yanshicon;
unsigned char weicon;

unsigned char code LED_tab[10]={

                                                0xFC,           //0
                                                0x60,           //1
                                                0xDA,           //2
                                                0xF2,           //3
                                                0x66,           //4
                                                0xB6,           //5
                                                0xBE,           //6
                                                0xE0,           //7
                                                0xFE,           //8
                                                0xF6,           //9

};

void display(int Data)
{
    int A4,A5,A6,A7,A8;
    if(Data>=10000)
            {
              A4=Data/10000;
              Data=Data%10000;
              A5=Data/1000;
              Data=Data%1000;
              A6=Data/100;
              Data=Data%100;
              A7=Data/10;
              Data=Data%10;
              A8=Data;
            }
        else if(Data>=1000)
            {
              A4=0;
              A5=Data/1000;
              Data=Data%1000;
              A6=Data/100;
              Data=Data%100;
              A7=Data/10;
              Data=Data%10;
              A8=Data;
            }
```

```
    else if(Data>=100)

       {
         A4=0;
         A5=0;
         A6=Data/100;
         Data=Data%100;
         A7=Data/10;
         Data=Data%10;
         A8=Data;
       }
    else if(Data>=10)
       {
         A4=0;
         A5=0;
         A6=0;
         A7=Data/10;
         Data=Data%10;
         A8=Data;
       }
     else
       {
         A4=0;
         A5=0;
         A6=0;
         A7=0;
         A8=Data;
       }

  P2=0;                                  //Turn-off display
  a0=1;
  a1=1;
  a2=1;
 //a2,a1,a0 control the display position, when it is changed from 0-7
 //the corresponding digital LEDs is from 1-8

while(1)
  {
          if(++yanshicon>200)            //Execute once every 200 scan cycle
          {
                  yanshicon=0;
             if(++weicon>7) weicon=0;    //Rotate the display position within the range from 0-7
                  P2=0;                  //Turn-off display
                  switch(weicon)
                  {
                          case 0:        //The content of first display position
                            P2=0;        //Turn-off display
                            a0=0;
                            a1=0;
                            a2=0;
                          break;
                          case 1:        //The content of second display position
                            P2=0;        //Turn-off display
                            a0=1;
                            a1=0;
                            a2=0;
                          break;
```

```
                    case 2:             //The content of third display position
                      P2=0;             //Turn-off display
                      a0=0;
                      a1=1;
                      a2=0;
                    break;
                    case 3:                 //The content of fourth display position
                      P2=LED_tab[A4];       //Display A4
                      a0=1;
                      a1=1;
                      a2=0;
                    break;
                    case 4:                 //The content of fifth display position
                      P2=LED_tab[A5];       //Display A5
                      a0=0;
                      a1=0;
                      a2=1;
                    break;
                    case 5:                 //The content of sixth display position
                      P2=LED_tab[A6];       //Display A6
                      a0=1;
                      a1=0;
                      a2=1;
                    break;
                    case 6:                 //The content of seventh display position
                      P2=LED_tab[A7];       //Display A7
                      a0=0;
                      a1=1;
                      a2=1;
                    break;
                    case 7:                 //The content of eighth display position
                      P2=LED_tab[A8];       //Display A8
                      a0=1;
                      a1=1;
                      a2=1;
                    break;
                    default:
                    break;
                }
            }
        }

}
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
```

PWM_display.h

```
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------//
```

Function declaration

```
//-----------------------------------------------------------------------------------------------------------------------//
void display(int Data);
//-----------------------------------------------------------------------------------------------------------------------//

//-----------------------------------------------------------------------------------------------------------------------/
```