C语言代码



MLX90614 和单片机 STC12C5604AD 的 SMBus, PWM 通信

1 目的

该文档的主要目的是描述怎样实现单片机和红外温度计 MLX90614 的 SMBus 通信,并简略介绍利用单片机对 MLX90614 的 PWM 输出脉冲进行测量并计算相关温度。代码是基于 STC12C5604AD 单片机的。实例给出了由 C语言实现 MLX90614 读取物体温度,写入数据到 MLX90614、以及对 PWM 脉冲的测量和计算等功能。

2 C 代码

2.1 由 MLX90614 读取温度部分

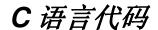
实例给出了应用 C 语言来实现从 MLX90614 中读取物体温度的程序。为了使程序操作和运行简单起见,整个项目被分为几个子项目。在该文档中整个 C 程序分为主文件 SMBus.c 《用来实现对物体温度的读取、改变 SMBus 地址、改变发射率、以及改变 PWMCTRL 配置参数功能); 子文件 SMBus_CM.c (具体描述 SMBus 通信的起始状态,终止状态,发送和接受数据); 子文件 SMBus_OP.c (包含了利用 SMBus 通信由 MLX90614 读取数据,对 MLX90614 写入数据和 PEC 校验码计算的程序); 子文件 Delay.c (调用延迟函数); 子文件 dec2hex.c (将十进制转换为十六进制); 子文件 CalTem.c (根据十六进制数值计算温度读数); 子文件 digitalLED.c (在数字 LEDs 上显示温度读数)。在主文件的头文件中引用相应的子文件.h 文件,以将所有功能综合、链接起来。

| 数);子文件 dec2hex.c(将 | 十进制转换为十六进制) 在数字 LEDs 上显示温度 | ;子文件 CalTem.c(根据十六进制数值计算温度 读数)。在主文件的头文件中引用相应的子文件. | 度读 |
|---|--|--|-------|
| // | | | // |
| //主文件 SMBus.c | | | |
| // | \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ | | |
| 的读取,对 MLX90614 写入数值并在数字 LEDs 上显示温// | TC12C5604AD 编写的 (女据的功能。实例给出了i 度的程序,但是此为可说 | C语言程序,可通过 SMBus 实现对 MLX90614 温 读取物体温度的程序,以及可根据十六进制数值计 选项,用户可根据自己的应用另行选择其他方式。 | - 算温度 |
| #include <stc12c5410ad.h "caltem.h"="" "dec2hex.h"="" "delay.h"<="" "digitalled.h"="" "intrins.h"="" "smbus_cm.h"="" "smbus_op.h"="" "stdio.h"="" "string.h"="" #include="" th=""><th>//引用 SMBus_CM.h : //引用 SMBus_OP.h : //引用 digitalLED.h 文 //引用 CalTem.h 文件 //引用 dec2hex.h 文件</th><th>文件(包含起始状态,终止状态,发送,接收字节等文件(包含起始状态,终止状态,发送,接收字节等文件(包含由 MLX90614 中读取数据和写入数据)件以在数字 LEDs 上显示温度读数来根据十六进制数值计算温度 中,并进制转换为十六进制</th><th>等)</th></stc12c5410ad.h> | //引用 SMBus_CM.h : //引用 SMBus_OP.h : //引用 digitalLED.h 文 //引用 CalTem.h 文件 //引用 dec2hex.h 文件 | 文件(包含起始状态,终止状态,发送,接收字节等文件(包含起始状态,终止状态,发送,接收字节等文件(包含由 MLX90614 中读取数据和写入数据)件以在数字 LEDs 上显示温度读数来根据十六进制数值计算温度 中,并进制转换为十六进制 | 等) |
| ### ********************************* | | | |
| #define _SDA_OUTPUT P1N #define _SDA_INPUT P1M0= #define _SCL_IO P1M0=0x0 sbit SDA = P1^4; | M0=0x10; P1M1=0x10; =0x10; P1M1=0x00; | | ,, |



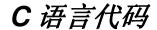


```
//指定 SCL 线给 P13
sbit SCL = P1<sup>3</sup>;
//-----
主函数功能
作用: 读取物体温度
void main()
  unsigned char
                slaveaddress;
  unsigned long int DATA;
  unsigned int
                *mahm:
                            //引用宏定义-设置 SCL 为开漏式 I/O
  _SCL_IO;
  _SDA_OUTPUT;
                            //引用宏定义-设置 SDA 为开漏式输出
  SCL=0:
                            //SMBus 请求时间、将 PWM 模式转换为 SMBus 模式(至少为 2ms)
  Delay(1200);
  SCL=1;
  while(1)
    slaveaddress=MEM READ(0x00,0x2E);
    //读取存于 MLX90614 EEPROM "0Eh"地址中的 SMBus 地址
    DATA=MEM READ(slaveaddress,0x07);
   //基于上述地址由 MLX90614 的内存 07h 中读取物体温度
    mahm=CALTEMP(DATA);
    //基于所得的十六进制温度格式计算实际温度
    show(mahm,5);
    //在数字 LEDs 上显示计算所得温度
  }
子文件 SMBus CM.c
该文档包含了 SMBus 通信的起始状态,终止状态,发送和接受字节等
头文件
#include <STC12C5410AD.H>
#include "SMBus_CM.h"
                                         //头文件中引用本身
#include "intrins.h"
#include "Delay.h"
宏定义 I/O 端口和 SMBus 信号的方向
//------//
#define _SDA_OUTPUT P1M0=0x10; P1M1=0x10; //设置 SDA 为开漏输出 #define _SDA_INPUT P1M0=0x10; P1M1=0x00; //设置 SDA 为高阻输入
#define _SCL_IO P1M0=0x08; P1M1=0x08;
                                         //设置 SCL 为开漏式 I/O 端口
sbit SDA = P1<sup>4</sup>;
                                        //指定 SDA 线给 P14
sbit SCL = P1<sup>3</sup>:
                                        //指定 SCL 线给 P13
```





```
函数名: start_bit
功能:在 SMBus 总线上产生起始状态
注解:参考"系统管理总线说明书-版本 2.0"
void start_bit()
 SDA OUTPUT;
                                //设置 SDA 为输出
 SDA=1;
                                //设置 SDA 线为高电平
 _nop_();_nop_();
 SCL=1;
                                //设置 SCL 线为高电平
                                //在终止和起始状态之间产生总线空闲时间(Tbuf=4.7us 最小值)
 Delay(5);
 SDA=0;
                                //设置 SDA 线为低电守
 Delay(5);
 //(重复)开始状态后的保持时间,在该时间后,产生第一个时钟信
                                //Thd:sta=4us 最小值
 SCL=0:
                                //设置 SCL 线为低电平
 _nop_();_nop_();
函数名: stop_bit
功能: 在 SMBus 总线上产生终止状态
注解:参考"系统管理总线说明书-版本 2.0"
void stop_bit()
 SDA OUTPUT;
                                   //设置 SDA 为输出
SCL=0;
                                   //设置 SCL 线为低电平
Delay(5);
SDA=0;
                                   //设置 SDA 线为低电平
 Delay(5);
 SCL=1;
                                  //设置 SCL 线为高电平
Delay(5);
                                  //终止状态建立时间(Tsu:sto=4.0us 最小值)
 SDA=1;
                                  //设置 SDA 线为高电平
函数名: send bit
功能:在 SMBus 总线上发送一位数据
void send_bit(unsigned char bit_out)
 _SDA_OUTPUT;
                              //设置 SDA 为开漏输出以在总线上传送数据
                              //核对字节的位
if(bit out==0)
                              //如果 bit_out=1,设置 SDA 线为高电平
     SDA=0;
 else
     SDA=1;
                              //如果 bit_out=0,设置 SDA 线为低电平
 _nop_();
 _nop_();
                              //Tsu:dat=250ns 最小值
 _nop_();
 SCL=1;
                              //设置 SCL 线为高电平
```





```
//时钟脉冲高电平脉宽(10.6us)
Delay(4);
SCL=0;
                                //设置 SCL 线为低电平
Delay(4);
                               //时钟脉冲低电平脉宽
函数名: receive bit
功能:在SMBus总线上接收一位数据
unsigned char receive bit()
unsigned char bit in;
_SDA_INPUT;
                                      //设置 SDA 为高阻输入
                                      //设置 SCL 缓为高电平
SCL=1;
Delay(2);
                                      //从总线上读取一位, 赋给 bit in
if(SDA==1)
   bit in=1;
else
   bit in=0;
Delay(2);
SCL=0;
                                      沙设置 SCL 线为低电平
Delay(4);
                                       //返回 bit in 值
return bit in;
函数名: slave ack
功能: 由受控器件 MLX90614 中读取确认位
返回值: unsigned char ack
1 - ACK
0 - NACK
unsigned char slave_ack()
 unsigned char ack;
 ack=0;
 _SDA_INPUT;
                                       //设置 SDA 为高阻输入
 SCL=1:
                                       //设置 SCL 线为高电平
 Delay(2);
 if(SDA==1)
                                       //从总线上读取一位, 赋给 ack
    ack=0;
 else
    ack=1;
 Delay(2);
                                       //设置 SCL 线为低电平
 SCL=0;
 Delay(4);
 return ack;
发送一个字节
函数名: TX_byte
功能:在 SMBus 总线上发送一个字节
参数: unsigned char TX buffer (将要在总线上发送的字节)
注解: 先发送字节的高位
```



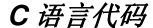


```
void TX_byte(unsigned char TX_buffer)
 unsigned char Bit_counter;
 unsigned char bit_out;
 for(Bit counter=8;Bit counter;Bit counter--)
   if(TX buffer&0x80)
              bit_out=1;
                          //如果 TX_buffer 的当前位是 1,设置 bit_out 为 1
           else
             bit_out=0;
                            //否则,设置 bit_out 为 0
                              //发送 SMBus 总线上的当前位
  send bit(bit out);
  TX buffer<<=1;
                              //核对下一位
接收一个字节
函数名: RX byte
功能:在 SMBus 总线上接收一个字节
参数: unsigned char ack_nack (确认位)
0-主控器件发送 ACK
1 - 主控器件发送 NACK
返回值: unsigned char RX buffer (总线接收的字
注解: 先接收字节的高位
//-----
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)
                                    //由 SDA 线读取一位
                                    //如果位为 "1", 赋 "1"给 RX_buffer
                 RX buffer<<=1;
                 RX buffer|=0x01;
                                    //如果位为 "0", 赋 "0"给 RX buffer
           else
            {
                 RX buffer<<=1;
                 RX buffer&=0xfe:
  }
      send bit(ack nack);
                                    //发送确认位
      return RX buffer;
子文件 SMBus OP.c
//------//
该文档包含了 SMBus 通信时从 MLX90614 读取数据,写入数据和 PEC 校验码计算的程序
```





```
头文件
#include <STC12C5410AD.H>
#include "SMBus_CM.h"
                             //引用 SMBus_CM.h 文件
#include "intrins.h"
#include "SMBus_OP.h"
                             //头文件中引用本身
#include "Delay.h"
sbit SDA = P1<sup>4</sup>:
                             //指定 MLX90614 的 SDA 线给单片机 P14 引脚
                            //指定 MLX90614 的 SCL 线给单片机 P13 引脚
sbit SCL = P1<sup>3</sup>;
计算 PEC 包裹校验码
函数名: PEC cal
功能: 根据接收的字节计算 PEC 码
参数: unsigned char pec[], int n
返回值: pec[0] - 该字节包含计算所得 crc 数值
注解: 参考"系统管理总线说明书-版本 2.0"和应用指南"MCLI和 MLX90614 的 SMBus 通信"
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;
                                     //载入 CRC 数值 0x00000000107
     crc[4]=0;
     crc[3]=0;
     crc[2]=0;
     crc[1]=0x01;
     crc[0]=0x07;
     Bitposition=47:
                                     //设置 Bitposition 的最大值为 47
     shift=0;
     //在传送的字节中找出第一个"1"
     i=5;
                                     //设置最高标志位 (包裹字节标志)
     j=0;
                                     //字节位标志, 从最低位开始
     while((pec[i]&(0x80>>j))==0 && (i>0))
      Bitposition--;
      if(j < 7)
          j++;
      else
         j=0x00:
         i--;
     }//while 语句结束,并找出 Bitposition 中为"1"的最高位位置
     shift=Bitposition-8;
                                      //得到 CRC 数值将要左移/右移的数值 "shift"
                                      //对 CRC 数据左移 "shift" 位
     while(shift)
```





```
for(i=5;i<0xFF;i--)
          if((crc[i-1]&0x80) && (i>0))
                                    //核对字节的最高位的下一位是否为"1"
                                    //是 - 当前字节 + 1
                                    //否 - 当前字节 + 0
             temp=1;
                                    //实现字节之间移动"1"
          }
          else
             temp=0;
          }
          crc[i]<<=1;
          crc[i]+=temp;
         shift--:
     //pec 和 crc 之间进行异或计算
     for(i=0;i<=5;i++)
         pec[i]^=crc[i];
  }while(Bitposition>8);
                                    #返回计算所得的 crc 数值
      return pec[0];
由 MLX90614 RAM/EEPROM 读取的数据
函数名: MEM READ
功能: 给定受控地址和命令时由 MLX90614 读取数据
参数: unsigned char slave addR (受控地址)
    unsigned char cmdR (命令)
返回值: unsigned long int Data
unsigned long int MEM READ(unsigned char slave addR, unsigned char cmdR)
       unsigned char DataL;
      unsigned char DataH;
                                               //由 MLX90614 读取的数据包
      unsigned char PEC;
      unsigned long int Data;
                                               //由 MLX90614 返回的寄存器数值
      unsigned char Pecreg;
                                               //存储计算所得 PEC 字节
      unsigned char arr[6];
                                               //存储已发送字节的缓冲器
      unsigned char ack nack;
      unsigned char SLA;
      SLA=(slave addR<<1);
 begin:
      start_bit();
                                               //发送起始位
       TX byte(SLA);
                                               //发送受控器件地址,写命令
      if(slave_ack()==0)
         stop_bit();
         goto begin;
                                               //发送命令
       TX byte(cmdR);
       if(slave_ack()==0)
```



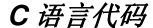


```
stop_bit();
         goto begin;
       start_bit();
                                               //发送重复起始位
       TX byte(SLA+1);
                                               //发送受控器件地址,读命令
       if(slave ack()==0)
         stop bit();
         goto begin;
       DataL=RX_byte(0);
                                               //
       DataH=RX byte(0);
       PEC=RX_byte(ack_nack);
                                               从读取 MLX906) 4 的 PEC 码
       if(ack nack==1)
                                               //主控器件发送 ack 或是 nack
       //取决于 pec 计算,如果 PEC 是不正确的,发送 nack 并返回到 goto begin
         stop_bit();
         goto begin;
                                                //发送终止位
       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);
                                               //调用计算 CRC 的函数
       if(PEC==Pecreg)
              ack_nack=0;
       else
        {
              ack nack=1;
       Data=(DataH*256)+DataL;
       return Data:
MLX90614 EEPROM 中写入数据
函数名: EEPROM_WRITE
功能: 根据命令写入相关数据到给定受控器件地址的 MLX90614
参数: unsigned char slave addW (受控器件地址)
unsigned char cmdW (命令)
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;
                                              //存储计算所得 PEC 字节
  unsigned char SLA;
  unsigned char arr[6];
                                              //存储将要发送字节的缓冲器
```



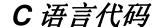


```
SLA=(slave_addW<<1);
  arr[5]=0;
  arr[4]=SLA;
  arr[3]=cmdW;
  arr[2]=DataL;
  arr[1]=DataH;
  arr[0]=0;
  Pecreg=PEC_cal(arr,6);
 begin:
                                                    //发送起始心
  start_bit();
  TX_byte(SLA);
  if(slave_ack()==0)
    stop_bit();
    goto begin;
  TX_byte(cmdW);
  if(slave_ack()==0)
    stop_bit();
    goto begin;
  TX_byte(DataL);
                                                    //发送数据低位字节
  if(slave_ack()==0)
    stop_bit();
    goto begin;
  TX_byte(DataH);
                                                    //发送数据高位字节
  if(slave_ack()==0)
    stop_bit();
    goto begin;
  TX_byte(Pecreg);
                                                    //发送 PEC 码
  if(slave_ack()==0)
    stop_bit();
    goto begin;
                                                    //发送终止位
  stop_bit();
  Delay(200);
                                                    //等候 5ms
子文件 Delay.c
延迟函数
```



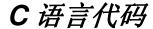


```
#include <STC12C5410AD.H>
#include "intrins.h"
#include "Delay.h"
函数名: Delay
功能: 延迟一定的时间
参数: N表示延迟时间为 MCU 机器周期的 N倍
注解: 机器周期是时钟周期(1/Fosc)的 12 倍,对于 STC12C5604AD, Fosc=11.0592MHz
void Delay(unsigned int N)
  unsigned int i;
 for(i=0;i<N;i++)
  nop ();
子文件 dec2hex.c
十进制转换为十六进制
//-----
                                   //头文件中引用自身
#include "dec2hex.h"
#include "stdio.h"
函数名: dec2hex
功能: 将十进制代码转换为十六进制代码
参数: float e (发射率数值)
返回值: unsigned int c
       c 是数组 c[4]的首地址
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;
```

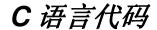




```
while(a!=0)
    j=a%16;
                                   //求模运算
    c[i++]=b[j];
                                   //求余运算
      a=a/16;
      m++;
  for(i=0;i< m/2;i++)
                                   //
      {
                                   //将数组 c[4]倒序
        t=c[i];
        c[i]=c[m-i-1];
        c[m-i-1]=t;
      return c;
                                   //返回数组 c[4]的首地址
子文件 CalTem.c
根据十六进制数值计算温度
#include "CalTem.h"
                                   //头文件中引用自身
函数名: CALTEMP
功能: 计算温度
参数: unsigned long int TEMP (由 MLX90614 中读到的数据)
返回值: unsigned int mah
        mah 是数组 mah[5]的首地址
注解: 将十六进制代码转换为温度数据的公式为 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;
```



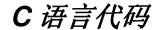


```
mah[0]=A8;
       return mah;
//-----//
子文件 digitalLED.c
根据计算所得数据在数字式 LEDs 上显示
#include <STC12C5410AD.H>
#include "digitalLED.h" //头文件引用自身
/定义 I/O 端口去控制数字式 LEDs
                        //指定 a0 线给 P1入
sbit a0 = P1^7:
sbit a1 = P1^5;
                       //指定 a1 线给 P15
sbit a2 = P1^6;
                       //指定 a2 线给 P16
//-----
函数名: show
功能: 在数字式 LEDs 上显示温度读数
参数: unsigned int mahh[],int number
注解:显示存储在数组 mahh 5 的数据
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
                                                         //3
                                      0xF2,
                                                         //4
                                      0x66,
                                      0xB6,
                                                         //5
                                                         //6
                                      0xBE,
                                                         //7
                                      0xE0,
                                      0xFE,
                                                         //8
                                      0xF6,
                                                         //9
               };
     P2=0;
                                //关显示
     a0=1;
     a1=1;
     //a2,a1,a0 控制显示位置,可由 0-7 之间变化
     //控制显示位置, 当其值为 0-7 时, 对应数码管位 1-8 位
     while(1)
     {
           if(++yanshicon>200)
                                //每 200 个扫描周期执行一次
                 vanshicon=0:
             if(++weicon>7) weicon=0; //显示位置轮流,需要严格限制范围为 0-7
```





```
P2=0;
                        //关显示
      switch(weicon)
                           //第1位显示内容
             case 0:
              P2=0;
                           //关显示
              a0=0;
              a1=0;
              a2=0;
             break;
                           //第2位显示内容
             case 1:
                           //关显示
              P2=0;
              a0=1:
              a1=0;
              a2=0;
             break;
                           //第3位显示内容
             case 2:
                           示显关\\
              P2=0;
              a0=0;
              a1=1;
              a2=0;
             break;
                                  //第 4 位显示内容
             case 3:
              P2=LED tab[mahh[4]]; //显示数组 mahh[4]的数值
              a0=1;
              a1=1;
              a2=0;
             break;
                                  //第5位显示内容
             case 4:
              P2=LED_tab[mahh[3]]; //显示数组 mahh[3]的数值
               a0=0;
               a1=0:
              a2=1;
             break;
             case 5:
                                  //第6位显示内容
              P2=LED_tab[mahh[2]]; //显示数组 mahh[2]的数值
              a0=1;
              a1=0;
              a2=1;
             break;
                                  //第7位显示内容
             case 6:
              P2=LED tab[mahh[1]]; //显示数组 mahh[1]的数值
              a0=0;
              a1=1;
              a2=1;
             break;
                                  //第8位显示内容
             case 7:
              P2=LED_tab[mahh[0]]; //显示数组 mahh[0]的数值
              a1=1;
              a2=1;
             break:
             default:
             break;
      }
}
```





Microelectronic Integrated Systems MLX90614 和单片机 STC12C5604AD 的 SMBus, PWM 通信

| PWM 主文件 | ICTRL 配置) SMBus.c 中同样可以实现改变% | 部分(改变发射率、改变 SMBus 地址、改变 | |
|-------------------|---|---|-----|
| 率,SM | | 入或减去需要或不需要的子头文件即可。 | |
| // | | | // |
| 功能: | 改变发射率 | | .// |
| ,, | | | |
| void ma | | | 11 |
| { | unsigned char slaveaddress; unsigned int *Emv; unsigned int EmvLO; unsigned int EmvHI; float Emissivity=0.5; Emv=dec2hex(Emissivity); EmvLO=(*(Emv+2)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvEmvHI=(*(Emv+0)<<4)+*(EmvEmvEmvEmvEmvEmvEmvEmvEmvEmvEmvEmvEmvE | | t |
| | _SCL_IO; _SDA_OUTPUT; SCL=0; Delay(1200); | // //SMBus 请求时间,将 PWM 模式转换为 SMBus 模式(至少为 2) | ms) |
| | SCL=1; | // | |
| | | 00,0x2E); | |
| // | | | // |
| ,, | | | // |
| | 改变 SMBus 地址 | | // |
| // | | | // |
| void ma { | ain() | | |
| ur ur | nsigned char slaveaddress; nsigned int DataLO; nsigned int DataHI; | | |



C语言代码

Microelectronic Integrated Systems MLX90614 和单片机 STC12C5604AD 的 SMBus, PWM 通信

| DataLO=0x5A; DataHI=0x00; _SCL_IO; _SDA_OUTPUT; | //载入受控器件地址低字节(用户需自己选择) //载入受控器件地址高字节 |
|---|--|
| SCL=0; Delay(1200); SCL=1; | // //SMBus 请求时间,将 PWM 模式转换为 SMBus 模式(至少为 2ms) // |
| slaveaddress=MEM_READ(0x | (00,0x2E); //得到存于 EEPROM "0Eh"的旧的受控器件地址 |
| EEPROM_WRITE(slaveaddres | ss,0x2E,0x00,0x00); |
| EEPROM_WRITE(slaveaddress) } // | // // |
| // | _ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ |
| // | |
| PWMCTRH = 0x02; //PWMCTRL 高字节 (0x02 代表 I _SCL_IO; _SDA_OUTPUT; | 先定单个 PWM 模式,使能 PWM,并且 SDA 引脚设置为 Push-Pull 推挽式) PWM 周期为 1.024ms*1,并且无 PWM 重复) |
| SCL=0; Delay(1200); SCL=1; | // //SMBus 请求时间,将 PWM 模式转换为 SMBus 模式(至少为 2ms) // |
| slaveaddress=MEM_READ(0x0 EEPROM_WRITE(slaveaddres | |
| EEPROM_WRITE(slaveaddres //写入新的 PWMCTRL 数值到 E //需要重启以激活 } | ss,0x22,PWMCTRLO,PWMCTRHI); EPROM "02h" |
| <i> </i> | // |
| 功能:改变 PWMCTRL 配置, | // 配置为 SMBus 输出格式 // |
| // | // |





```
void main()
   unsigned char slaveaddress;
   unsigned char PWMCTRLO;
   unsigned char PWMCTRHI;
   PWMCTRLO=0x01:
//PWMCTRL 低字节(0x01 代表选定单个 PWM 模式,未使能 PWM,并且 SDA 引脚设置为 Open-Drain 开漏式)
   PWMCTRHI=0x02;
   //PWMCTRL 高字节(0x02 代表 PWM 周期为 1.024ms*1, 并且无 PWM 重复)
   SCL IO;
   _SDA_OUTPUT;
     SCL=1;
     Delay(10);
     SCL=0;
     Delay(1200);
                   //SMBus 请求时间,将PWM模式转换为 SMBus 模式(至少为 2ms)
     SCL=1;
    //切换为 SMBus 模式后, 改变 PWMCTRL 配置(SMBus), 再关断/重启电源, 器件将工作在 SMBus 模式
    slaveaddress=MEM_READ(0x00,0x2E);
    EEPROM WRITE(slaveaddress,0x22,0x00,0x00);
                                  //写入数据 0x0000 到 EEPROM "02h"
    EEPROM WRITE(slaveaddress,0x22,PWMCTRLO,PWMCTRHI);
    //写入新的 PWMC/RL数值到 EEPROM "02h"
    //需要重启以激活
}
2.3 子文件对应的.h 文件介绍
//-----//
//-----//
SMBus CM.h
头文件保护
#ifndef SMBUS CM H
#define SMBUS CM H
||-----||
||-----||
函数声明
//-----//
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
```





Microelectronic Integrated Systems MLX90614 和单片机 STC12C5604AD 的 SMBus, PWM 通信

| // SMBus_OP.h | // |
|---|----------------------|
| 头文件保护 | // |
| / #ifndef SMBUS_OP_H #define SMBUS_OP_H / | // |
| / 函数声明 / | |
| unsigned char PEC_cal(unsigned char pecunsigned long int MEM READ(unsigned char | [],int n); |
| #endif / | |
| // | |
| Delay.h | |
| // 函数声明 | |
| void Delay(unsigned int N); | |
| | // |
| dec2hex.h | // |
| 函数声明 | // |
| unsigned int *dec2hex(float e); | // |
| ′/ | // |
| | // |
| CalTem.h // | // |
| 函数声明 / | // |
| extern unsigned int *CALTEMP(unsigned lo | ong int TEMP); // |
| '/ | // |
| digitalLED.h | // |
| 函数声明 | // |
| extern void show(unsigned int mahh[],int nu | umber); |
| '/ | // |
| / | // |





3 根据 PWM 输出模式计算温度读数

MLX90614 可以设置为 PWM 输出模式,实例给出用 STC MCU 定时器功能实现对温度计 PWM 格式的读出,并计算和显示相关温度。(注意,本例中 MLX90614 的输出方式为单个 PWM 格式,PWM 的周期为 1.024ms,显示为物体温度读数。) 关于 PWM 输出方式的介绍,具体请参考应用指南-由 MLX90614 读取 PWM 格式数据并利用 PIC 18 MCU 实现温度计算。

| 并利用 PIC 18 MCU 实现温度计算 | |
|--|---|
| // | |
| // | |
| 主文件 PWM.c | |
| // | |
| // | |
| // 头文件 | |
| 天义竹 // | |
| #include <stc12c5410ad.h> #include "stdio.h" #include "math.h" #include "intrins.h"</stc12c5410ad.h> | |
| #include "string.h" #include "PWM_display.h" | 四山用 RWM_display.h (在数字式 LEDs 上显示温度读数) |
| // | |
| sbit P3_3=P3^3; | ///////////////////////////////////// |
| //功能:根据 PWM 脉冲宽度和周期 | (占空比)来计算需要显示的温度数据 // |
| int Calculate(unsigned int 11pwm,u { | |
| <pre>int DC; long int T; int T1; int K; int T1max=120; int T1min=-20; K=2*(T1max-T1min);</pre> | //物体温度的最大值,存于 EEPROM 00h 地址,用户可自行设置 //物体温度的最小值,存于 EEPROM 01h 地址,用户可自行设置 |
| DC=t1pwm*100000/T1pwm; T=(DC-0.125*100000)*K+T1min | |
| // 根据公式 $Tout = [2(DC - 0.12)]$ T1=T/1000; return T1; | $(T_{\text{max}} - T_{\text{min}})$]+ T_{min} 计算单个 PWM 模式下物体温度,DC 为占空比 //为实现分辨率为 0.01℃,结果需除以 1000 |
| } | <i>II</i> |
| | 端口 1、定时器 0 和 1 实现对 PWM 脉冲宽度和周期的测量 |
| 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;
                       //设置定时器 0 (GATE=1) 和定时器 (GATE=0) 都为定时方式 1
 TH0=0x00:
                       //TH0,TL0 清零
 TL0=0x00;
                       //开启定时器 0 中断允许, 允许定时器 0 中断
 ET0=1;
 ET1=1;
                       //开启定时器 1 中断允许、允许定时器 ) 中断
 EA=1:
                       //开启全局中断允许, 允许所有中断
 TL1=0x00:
                       //TH1,TL1 清零
 TH1=0x00;
 EX1=0;
                       //关 INT1 中断
                       //等待 INT1 引脚低电平
while(P3_3==1)
while(P3_3==0)
                       //等待 INT 引脚高电平
{;}
TR0=1;
                       #开启定时器 0 开始计数
TR1=1;
                       以开启定时器 1 开始计数
                       #等待 INT1 引脚低电平
while(P3 3==1)
{;}
TR0=0;
                       //停止定时器0计数
C=TL0;
                       //T0 低字节计数值送到 C
D=TH0;
                       //T0 高字节计数值送到 D
while(P3_3==0)
                       //等待 INT1 引脚高电平
{;}
TR1=0;
                       //停止定时器 1 计数
A=TL1;
                       //T1 低字节计数值送到 A
B=TH1:
                       //T1 高字节计数值送到 B
Data2=(D << 8) + C;
Data1=(B < < 8) + A;
Data=Calculate(Data2,Data1);
                       //调用函数,根据两个计数器数值计算温度
display(Data);
                       //显示温度数据
}
                 -----//
子文件 PWM display.c
头文件
#include <STC12C5410AD.H>
#include "Intrins.h"
#include "PWM_display.h"
```



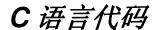


```
I/O 端口的宏定义
sbit a0 = P1^7;
                               //指定 P17 给 a0 线
sbit a1 = P1^5;
                               //指定 P15 给 a1 线
                               //指定 P16 给 a2 线
sbit a2 = P1^6;
unsigned char yanshicon;
unsigned char weicon;
unsigned char code LED_tab[10]={
                                                 0xFC,
                                                                           //0
                                                 0x60,
                                                                           //1
                                                 OXDA.
                                                                           //2
                                                 0xF2,
                                                                           //3
                                                 0x66.
                                                                           //4
                                                 0xB6,
                                                                           //5
                                                                           //6
                                                 0xBE,
                                                                           //7
                                                 0xE0,
                                                 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;
   }
                              //关显示
P2=0;
a0=1;
a1=1;
a2=1;
//a2,a1,a0 控制显示位置、当其值为 0-7 时,对应数码管位 1-8 位
while(1)
      if(++yanshicon>200)
                              //每 200 个扫描周期执行一次
             yanshicon=0;
        if(++weicon>7) weicon=0; //显示位置轮流, 需要严格限制范围为 0-7
             P2=0;
                              //关显示
             switch(weicon)
                                 //第1位显示内容
                   case 0:
                                 //关闭显示
                     P2=0;
                     a0=0;
                     a1=0;
                     a2=0;
                   break;
                                 //第2位显示内容
                   case 1:
                     P2=0;
                                //关闭显示
                     a0=1;
                     a1=0;
                     a2=0;
                   break;
                   case 2:
                                 //第3位显示内容
                     P2=0;
                                //关闭显示
                     a0=0;
                     a1=1;
                     a2=0;
```





```
break;
                           case 3:
                                             //第4位显示内容
                            P2=LED_tab[A4]; //显示 A4
                            a0=1;
                            a1=1;
                            a2=0;
                           break;
                                             //第5位显示内容
                          case 4:
                            P2=LED_tab[A5];
                                            //显示 A5
                            a0=0;
                            a1=0;
                            a2=1;
                           break;
                           case 5:
                                             //第6位显示内
                                             //显示 A6
                            P2=LED_tab[A6];
                            a0=1;
                            a1=0;
                            a2=1;
                           break;
                                              #第7位显示内容
                           case 6:
                            P2=LED_tab[A7];
                                             ₩显示 A7
                            a0=0;
                            a1=1;
                            a2=1;
                           break;
                                             //第8位显示内容
                           case 7:
                            P2=LED tab[A8];
                                            //显示 A8
                            a0=1;
                            a1=1;
                            a2=1;
                           break;
                           default:
                           break;
      }
PWM display.h
函数声明
void display(int Data);
```