E200 软件源程序

```
#include "Drv_GN1640.h"

/*
*********************************************************************
 * 函数原型：static void GN1640_Start(void)
 * 功      能：开始传输
 * 调      用：内部调用
*********************************************************************
*/
static void GN1640_Start(void)
{
    GN1640_CLK_H;
    GN1640_DATA_H;
    Delay_us(10);
    GN1640_DATA_L;
    Delay_us(10);
    GN1640_CLK_L;
    Delay_us(10);
}

/*
*********************************************************************
 * 函数原型：static void GN1640_Stop(void)
 * 功      能：停止传输
 * 调      用：内部调用
*********************************************************************
*/
static void GN1640_Stop(void)
{
    GN1640_DATA_L;
    GN1640_CLK_H;
    Delay_us(10);
    GN1640_DATA_H;
    Delay_us(10);
}

/*
*********************************************************************
 * 函数原型：static void GN1640_Write_Byte(uint8_t date)
 * 功      能：写入一个字节
 * 输      入：date：字节
 * 参      数：uint8_t date
 * 调      用：内部调用
*********************************************************************
*/
static void GN1640_Write_Byte(uint8_t date)
{
    uint8_t i;
    uint8_t Temp;
```

```
    Temp=date;
    GN1640_CLK_L;
    GN1640_DATA_L;
    for(i=0;i<8;i++)
    {
        GN1640_CLK_L;
        Delay_us(2);
        if(Temp&0x01)
        {
            GN1640_DATA_H;
            Delay_us(10);
        }
        else
        {
            GN1640_DATA_L;
            Delay_us(10);
        }
        GN1640_CLK_H;
        Delay_us(1);
        Temp = Temp >> 1;
    }
    GN1640_CLK_L;
    GN1640_DATA_L;
}

/*
*********************************************************************
 * 函数原型：static void Write_Com(uint8_t date)
 * 功    能：发送命令字
 * 输    入：date：命令字节
 * 参    数：uint8_t date
 * 调    用：内部调用
*********************************************************************
*/
static void Write_Com(uint8_t date)
{
    GN1640_Start();
    GN1640_Write_Byte(date);
    GN1640_Stop();
}

/*
*********************************************************************
 * 函数原型：void GN1640_Init(void)
 * 功    能：GN1640 初始化
 * 调    用：内部调用
*********************************************************************
*/
void GN1640_Init(void)
{
```

```
    GN1640_DATA_H;
    GN1640_CLK_H;
    GN1640_Start();
    Write_Com(0x44);
    GN1640_Stop();
    GN1640_Start();
    Write_Com(0x89); //控制显示，开显示，0x88, 0x89, 0x8a, 0x8b, 0x8c, 0x8d, 0x8e, 0x8f 分别对应脉冲宽度为:
    GN1640_Stop();

    GN1640_ALL();//全屏函数
    HAL_Delay(900);
    GN1640_CLS();
}

/*
*********************************************************************
 * 函数原型：void GN1640_CLS(void)
 * 功    能：清屏函数
 * 调    用：内部调用
*********************************************************************
*/
void GN1640_CLS(void)
{
    unsigned char i;
    Write_Com(0x40);//连续地址模式
    GN1640_Start();
    GN1640_Write_Byte(0xC0);
    for(i=0;i<16;i++)
        GN1640_Write_Byte(0x00);
    GN1640_Stop();
}

/*
*********************************************************************
 * 函数原型：void GN1640_ALL(void)
 * 功    能：全屏函数
 * 调    用：内部调用
*********************************************************************
*/
void GN1640_ALL(void)
{
    unsigned char i;
    Write_Com(0x40);//连续地址模式
    GN1640_Start();
    GN1640_Write_Byte(0xC0);
    for(i=0;i<16;i++)
        GN1640_Write_Byte(0xFF);
    GN1640_Stop();
}
```

```
/*
*********************************************************************
 * 函数原型：void GN1640_Write_DATA(uint8_t add,uint8_t DATA)
 * 功      能：指定地址写入数据
 * 输      入：uint8_t add,uint8_t DATA
 * 参      数：add：地址，DATA：数据
 * 调      用：内部调用
*********************************************************************
*/
void GN1640_Write_DATA(uint8_t add,uint8_t DATA)
{
      Write_Com(0x44);
      GN1640_Start();
      GN1640_Write_Byte(0xc0|add);
      GN1640_Write_Byte(DATA);
      GN1640_Stop();
}
#include "Drv_Beep.h"

/**********全局变量**********/
float Beep_Time;//蜂鸣器响的时间
float Beep_Flash;//蜂鸣器响的次数

/*
*********************************************************************
 * 函数原型：   void Buzzer_Status(float dT)
 * 功      能：  蜂鸣器的状态检测
 * 输      入:  dT:执行周期
 * 参      数：  uint16_t dT
*********************************************************************
*/
void Buzzer_Status(float dT)
{
      static float BT;
      if(Beep_Time <= 0 && Beep_Flash <= 0)//蜂鸣器的时间小于等于 0 时
      {
            Beep_OFF;//关闭蜂鸣器
            return;
      }
      if(Beep_Time)
      {
            Beep_ON;//打开蜂鸣器
            Beep_Time -= dT;//蜂鸣器响的时间--
      }
      if(Beep_Flash)
      {
            BT =    BT + dT;//周期++
            if(BT < 0.3)//如果小于 0.2s 时
```

```
            {
                Beep_ON;//蜂鸣器响
            }
            else if(BT >= 0.3 && BT < 0.5)//在 0.3 和 0.5s 之间时
            {
                Beep_OFF;//关闭蜂鸣器
            }
            else if(BT >= 0.5)//大于等于 0.4s 时
            {
                Beep_Flash--;//次数--
                BT = 0;//周期清零
            }
        }
}
#include "Drv_Key.h"

/**********全局变量声明******/
uint8_t Key_Status;//按键按下标志
uint8_t LongPress;//检测上下长按

/**********局部变量声明******/
float Key_Cnt1,Key_Cnt2,Key_Cnt3,Key_Cnt4,Key_Cnt5,Key_Cnt6;//按下时间
uint8_t Key_Flag1,Key_Flag2,Key_Flag3,Key_Flag4,Key_Flag5,Key_Flag6;//按键按下标志
uint8_t LongPress1,LongPress2,LongPress3,LongPress4,LongPress5,LongPress6;//按键长按标志

/*
**********************************************************************
 * 函数原型：void Check_Press(float dT)
 * 功      能：检测按键按下状态-500ms
**********************************************************************
*/
void Check_Press(float dT)
{
    if(Key_Status)//按键按下
        Key_Status -= dT;//倒计时
}

/*
**********************************************************************
 * 函数原型：void Key_Scan(float dT)
 * 功      能：矩阵按键扫描
**********************************************************************
*/

void Key_Scan(float dT)
{
    /**************************************Temp_Set                                    键
**************************************/
    if(KEY1 == 0)//按下按键
    {
```

```
        if(LongPress1 == 0)//没有长按过
        {
            Key_Cnt1 += dT;//按下时间++
            Key_Flag1 = 1;//按键按下标志置一
        }
    }
    if(Key_Flag1 == 1)//按键被按下
    {
        if(KEY1 == 1)//抬起按键
        {
            if(Key_Cnt1 < 1.5)//小于 1.5S 是单击
            {
                if(sys.Calibration)//如果在校准界面时
                {
                    if(Heat_Temp.ADDMode)
                    {
                        sys.Calibration_Step++;//校准步骤加加
                        if(sys.Calibration_Step>1)//如果校准不走大于 1
                        {
                            sys.Calibration = 0;//关闭校准
                            sys.Calibration_Step = 0;//步骤清零
                            sys.TempOption_Mun = 0;//设置是位数清零
                            Calibration(sys.Calibration_Temp1,sys.Calibration_Temp2);//
温度校准计算
                            sys.Calibration_Temp1 = 400;
                            sys.Calibration_Temp2 = 600;
                            sys.SetMode_Option = 0;
                        }
                        Heat_Temp.ADDMode = 0;//温度显示处理为 0
                    }
                }
                else
                {
                    sys.SetMode_Option++;
                    if(sys.SetMode_Option > 2)
                        sys.SetMode_Option = 1;
                    sys.TempOption_Mun = 0;
                    Twinkle_Time = 5;//闪烁 5S
                }
            }
            Key_Flag1 = 0;//按键事件结束，等待下一次按下
            LongPress1 = 0;//长按标志清零
            Key_Cnt1 = 0;//按钮计数清零
        }
        if(Key_Cnt1 > 1.5 && Key_Cnt1 < 3.1)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress1 == 0)//如果没有一直一直长按着
            {
                /*校准温度*/
                if(sys.Calibration == 0)//没有在校准界面时
```

```
                {
                    sys.Calibration = 1;//进入校准界面
                    sys.Calibration_Step = 0;//校准步骤清零
                    sys.Calibration_Temp1 = 400;//设置温度一为 40 度
                    sys.Calibration_Temp2 = 600;//设置温度二为 60 度
                    sys.SetMode_Option = 4;//设置模式为四
                    sys.TempOption_Mun = 2;//温度位数为最后一位
                    Heat_Temp.ADDMode = 0;//温度显示处理为 0
                    Cool_Temp.ADDMode = 0;//温度显示处理为 0
                }
                LongPress1 = 1;//长按标志置一
            }
        }
    }
    /**************************************Time_Set                                    键
**************************************/
    if(KEY2 == 0)//按下按键
    {
        if(sys.Calibration)
            return;
        if(LongPress2 == 0)//没有长按过
        {
            Key_Cnt2 += dT;//按下时间++
            Key_Flag2 = 1;//按键按下标志置一
        }
    }
    if(Key_Flag2 == 1)//按键被按下
    {
        if(KEY2 == 1)//抬起按键
        {
            if(Key_Cnt2 < 1.5)//小于 1.5S 是单击
            {
                if(sys.SetMode_Option == 0)
                {
                    sys.SetMode_Option = 3;//设置时间
                    Twinkle_Time = 5;//闪烁 5S
                }
            }
            Key_Flag2 = 0;//按键事件结束，等待下一次按下
            LongPress2 = 0;//长按标志清零
            Key_Cnt2 = 0;//按钮计数清零
        }
        if(Key_Cnt2 > 1.5 && Key_Cnt2 < 3.1)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress2 == 0)//如果没有一直一直长按着
            {

                LongPress2 = 1;//长按标志置一
            }
        }
```

```
    }

    /************************************ 加 键
************************************/
    if(KEY3 == 0)//按下按键
    {
        if(!sys.SetMode_Option)
            return;
        Key_Cnt3 += dT;//按下时间++
        Key_Flag3 = 1;//按键按下标志置一
    }
    if(Key_Flag3 == 1)//按键被按下
    {
        if(KEY3 == 1)//抬起按键
        {
            if(Key_Cnt3 < 1.5)//小于 1.5S 是单击
            {
                if(sys.SetMode_Option == 1)//设置加热区域时
                {
                    if(sys.TempOption_Mun == 0)
                        Heat_Temp.Set_Temp += 10;
                    else if(sys.TempOption_Mun == 1)
                        Heat_Temp.Set_Temp += 100;
                    else if(sys.TempOption_Mun == 2)
                        Heat_Temp.Set_Temp += 1;
                    if(Heat_Temp.Set_Temp > 600)
                        Heat_Temp.Set_Temp = 600;
                    Twinkle_Time = 5;//闪烁 5S
                    Key_Status = 2.0f;//操作时常亮 1S
                }
                else if(sys.SetMode_Option == 2)//设置制冷区域时
                {
                    if(sys.TempOption_Mun == 0)
                        Cool_Temp.Set_Temp += 10;
                    else if(sys.TempOption_Mun == 1)
                        Cool_Temp.Set_Temp += 100;
                    else if(sys.TempOption_Mun == 2)
                        Cool_Temp.Set_Temp += 1;
                    if(Cool_Temp.Set_Temp > 500)
                        Cool_Temp.Set_Temp = 500;
                    Twinkle_Time = 5;//闪烁 5S
                    Key_Status = 2.0f;//操作时常亮 1S
                }
                else if(sys.SetMode_Option == 3)//设置时间时
                {
                    if(sys.TimeOption_Mun == 0)
                        Time.Set_Time += 60;
                    else if(sys.TimeOption_Mun == 1)
                        Time.Set_Time += 600;
                    else if(sys.TimeOption_Mun == 2)
```

```
                    Time.Set_Time += 1;
                else if(sys.TimeOption_Mun == 3)
                    Time.Set_Time += 10;
                if(Time.Set_Time > 5999)//如果大于 99 分 59 秒
                    Time.Set_Time = 5999;
                Twinkle_Time = 5;//闪烁 5S
                Key_Status = 2.0f;//操作时常亮 1S
            }
            else if(sys.SetMode_Option == 4)//设置校准温度
            {
                if(sys.Calibration_Step == 0)
                {
                    if(sys.TempOption_Mun == 2)
                        sys.Calibration_Temp1 += 1;
                    if(sys.Calibration_Temp1 > 800)
                        sys.Calibration_Temp1 = 800;
                }
                if(sys.Calibration_Step == 1)
                {
                    if(sys.TempOption_Mun == 2)
                        sys.Calibration_Temp2 += 1;
                    if(sys.Calibration_Temp2 > 800)
                        sys.Calibration_Temp2 = 800;
                }
                Key_Status = 2.0f;//操作时常亮 1S
            }
        }
        Key_Flag3 = 0;//按键事件结束，等待下一次按下
        Key_Cnt3 = 0;//按钮计数清零
        LongPress = 0;//长按清零
    }
    if(Key_Cnt3 > 2.0 && Key_Cnt3 < 2.5)//按键时间大于 2.0 小于 2.5 表示长按
    {
        if(sys.SetMode_Option > 0 && sys.SetMode_Option <= 2)//设置温度时
        {
            sys.TempOption_Mun++;//温度位数++
            if(sys.TempOption_Mun > 2)
                sys.TempOption_Mun = 0;
            LongPress = 1;//表示长按加减
            Twinkle_Time = 5;//闪烁 5S
        }
        if(sys.SetMode_Option == 3)//设置时间时
        {
            sys.TimeOption_Mun++;//时间位数++
            if(sys.TimeOption_Mun > 3)
                sys.TimeOption_Mun = 0;
            LongPress = 1;//表示长按加减
            Twinkle_Time = 5;//闪烁 5S
        }
        Key_Flag3 = 0;//按键事件结束，等待下一次按下
```

```
            Key_Cnt3 = 1.5;//按钮计数清零
        }
    }

    /************************************** 减 键
**************************************/
    if(KEY4 == 0)//按下按键
    {
        if(!sys.SetMode_Option)
            return;
        Key_Cnt4 += dT;//按下时间++
        Key_Flag4 = 1;//按键按下标志置一
    }
    if(Key_Flag4 == 1)//按键被按下
    {
        if(KEY4 == 1)//抬起按键
        {
            if(Key_Cnt4 < 1.5)//小于 1.5S 是单击
            {
                if(sys.SetMode_Option == 1)//设置加热区域时
                {
                    if(sys.TempOption_Mun == 0)
                        Heat_Temp.Set_Temp -= 10;
                    else if(sys.TempOption_Mun == 1)
                        Heat_Temp.Set_Temp -= 100;
                    else if(sys.TempOption_Mun == 2)
                        Heat_Temp.Set_Temp -= 1;
                    if(Heat_Temp.Set_Temp < 300)
                        Heat_Temp.Set_Temp = 300;
                    Twinkle_Time = 5;//闪烁 5S
                    Key_Status = 2.0f;//操作时常亮 1S
                }
                else if(sys.SetMode_Option == 2)//设置制冷区域时
                {
                    if(sys.TempOption_Mun == 0)
                        Cool_Temp.Set_Temp -= 10;
                    else if(sys.TempOption_Mun == 1)
                        Cool_Temp.Set_Temp -= 100;
                    else if(sys.TempOption_Mun == 2)
                        Cool_Temp.Set_Temp -= 1;
                    if(Cool_Temp.Set_Temp < 200)
                        Cool_Temp.Set_Temp = 200;
                    Twinkle_Time = 5;//闪烁 5S
                    Key_Status = 2.0f;//操作时常亮 1S
                }
                else if(sys.SetMode_Option == 3)//设置时间时
                {
                    if(sys.TimeOption_Mun == 0)
                        Time.Set_Time -= 60;
                    else if(sys.TimeOption_Mun == 1)
```

```
                    Time.Set_Time -= 600;
                else if(sys.TimeOption_Mun == 2)
                    Time.Set_Time -= 1;
                else if(sys.TimeOption_Mun == 3)
                    Time.Set_Time -= 10;
                if(Time.Set_Time < 1)//如果小于 1 秒
                    Time.Set_Time = 1;
                Twinkle_Time = 5;//闪烁 5S
                Key_Status = 2.0f;//操作时常亮 1S
            }
            else if(sys.SetMode_Option == 4)//设置校准温度
            {
                if(sys.Calibration_Step == 0)
                {
                    if(sys.TempOption_Mun == 2)
                        sys.Calibration_Temp1 -= 1;
                    if(sys.Calibration_Temp1 < 200)
                        sys.Calibration_Temp1 = 200;
                }
                if(sys.Calibration_Step == 1)
                {
                    if(sys.TempOption_Mun == 2)
                        sys.Calibration_Temp2 -= 1;
                    if(sys.Calibration_Temp2 < 200)
                        sys.Calibration_Temp2 = 200;
                }
                Key_Status = 2.0f;//操作时常亮 1S
            }
        }
        Key_Flag4 = 0;//按键事件结束，等待下一次按下
        Key_Cnt4 = 0;//按钮计数清零
        LongPress = 0;//长按清零
    }
    if(Key_Cnt4 > 2.0 && Key_Cnt4 < 2.5)//按键时间大于 2.0S 小于 2.5 表示长按
    {
        if(sys.SetMode_Option > 0 && sys.SetMode_Option <= 2)//设置温度时
        {
            if(sys.TempOption_Mun)
                sys.TempOption_Mun--;//温度位数--
            else
                sys.TempOption_Mun = 2;
            LongPress = 1;//表示长按加减
            Twinkle_Time = 5;//闪烁 5S
        }
        if(sys.SetMode_Option == 3)//设置时间时
        {
            if(sys.TimeOption_Mun)
                sys.TimeOption_Mun--;//温度位数--
            else
                sys.TimeOption_Mun = 3;
```

```
                LongPress = 1;//表示长按加减
                Twinkle_Time = 5;//闪烁 5S
            }
            Key_Flag4 = 0;//按键事件结束，等待下一次按下
            Key_Cnt4 = 1.5;//按钮计数清零
        }
    }

    /*************************************Count                                    键
*************************************/
    if(KEY5 == 0)//按下按键
    {
        if(Heat_Temp.ADDMode == 0 || Cool_Temp.ADDMode == 0 || sys.Calibration == 1)
            return;
        if(LongPress5 == 0)//没有长按过
        {
            Key_Cnt5 += dT;//按下时间++
            Key_Flag5 = 1;//按键按下标志置一
        }
    }
    if(Key_Flag5 == 1)//按键被按下
    {
        if(KEY5 == 1)//抬起按键
        {
            if(Key_Cnt5 < 1.5)/*单击*///小于 1.5S 是单击
            {
                if(Time.CountDown_Start)//如果倒计时也在启动
                {
                    Time.CountDown_Start = 0;//关闭倒计时
                    Time.Ctrl_Time = Time.Set_Time;//赋值
                    Time.TimeDisplay_Flag = 0;//显示正计时
                }
                else if(Time.Count_Start)//如果正计时启动
                {
                    Time.Count_Start = 0;//关闭正计时
                    Time.Count_Time = 0;//清除正计时时间
                    Time.TimeDisplay_Flag = 0;//显示倒计时
                }
                else//如果正计时没有启动
                {
                    Time.TimeDisplay_Flag = 1;//显示正计时
                    Time.Count_Start = 1;//打开正计时
                    Beep_Time = 0.2;//蜂鸣器响 0.2S
                }

            }
            Key_Flag5 = 0;//按键事件结束，等待下一次按下
            LongPress5 = 0;//长按标志清零
            Key_Cnt5 = 0;//按钮计数清零
        }
```

```
        if(Key_Cnt5 > 1.5 && Key_Cnt5 < 3)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress5 == 0)/*长按*///如果没有一直一直长按着
            {
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                LongPress5 = 1;//长按标志置一
            }
        }
    }

    /*************************************CountDown                                键
**************************************/
    if(KEY6 == 0)//按下按键
    {
        if(Heat_Temp.ADDMode == 0 || Cool_Temp.ADDMode == 0 || sys.Calibration == 1)
            return;
        if(LongPress6 == 0)//没有长按过
        {
            Key_Cnt6 += dT;//按下时间++
            Key_Flag6 = 1;//按键按下标志置一
        }
    }
    if(Key_Flag6 == 1)//按键被按下
    {
        if(KEY6 == 1)//抬起按键
        {
            if(Key_Cnt6 < 1.5)/*单击*///小于 1.5S 是单击
            {
                if(Time.Count_Start)//如果正在计时
                {
                    Time.Count_Start = 0;//关闭正计时
                    Time.Count_Time = 0;//清除正计时时间
                    Time.TimeDisplay_Flag = 0;//显示倒计时
                }
                else if(Time.CountDown_Start)//如果正在倒计时
                {
                    Time.TimeDisplay_Flag = 0;//显示倒计时
                    Time.CountDown_Start = 0;//关闭倒计时
                    Time.Ctrl_Time = Time.Set_Time;//赋值
                }
                else
                {
                    Time.TimeDisplay_Flag = 0;//显示倒计时
                    Time.CountDown_Start = 1;
                    Beep_Time = 0.2;//蜂鸣器响 0.2S
                }
            }
            Key_Flag6 = 0;//按键事件结束，等待下一次按下
            LongPress6 = 0;//长按标志清零
            Key_Cnt6 = 0;//按钮计数清零
```

```
        }
        if(Key_Cnt6 > 1.5 && Key_Cnt6 < 3)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress6 == 0)/*长按*///如果没有一直一直长按着
            {
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                LongPress6 = 1;//长按标志置一
            }
        }
    }
}
#include "Drv_Flash.h"

/*********************用法*********************/
//Flash_Write((uint8_t *)(&Param),sizeof(Param));
//Flash_Read((uint8_t *)(&Param),sizeof(Param));
/*
*********************************************************************
 * 函数原型：uint8_t Flash_Write(uint8_t *addr, uint16_t len)
 * 功    能：写入 Flash
 * 输    入：addr 需要写入结构体的地址，len 结构体长度
 * 输    出：写入是否成功
 * 参    数：uint8_t *addr, uint16_t len
*********************************************************************
*/
uint8_t Flash_Write(uint8_t *addr, uint16_t len)
{
    uint16_t   FlashStatus;//定义写入 Flash 状态
    FLASH_EraseInitTypeDef  My_Flash;// 声 明  FLASH_EraseInitTypeDef  结 构 体 为 My_Flash

    HAL_FLASH_Unlock();//解锁 Flash

    My_Flash.TypeErase = FLASH_TYPEERASE_PAGES;//标明 Flash 执行页面只做擦除操作
    My_Flash.PageAddress = PARAMFLASH_BASE_ADDRESS;//声明要擦除的地址
    My_Flash.NbPages = 1;//说明要擦除的页数，此参数必须是 Min_Data = 1 和 Max_Data =(最大页数-初始页的值)之间的值

    uint32_t PageError = 0;//设置 PageError,如果出现错误这个变量会被设置为出错的 FLASH 地址

    FlashStatus = HAL_FLASHEx_Erase(&My_Flash, &PageError);//调用擦除函数（擦除 Flash）
    if(FlashStatus != HAL_OK)
        return 0;

    for(uint16_t i=0; i<len; i=i+2)
    {
        uint16_t temp;//临时存储数值
```

```
            if(i+1 <= len-1)
                temp = (uint16_t)(addr[i+1]<<8) + addr[i];
            else
                temp = 0xff00 + addr[i];
            //对 Flash 进行烧写，FLASH_TYPEPROGRAM_HALFWORD 声明操作的 Flash 地址的 16 位的，此外还有 32 位跟 64 位的操作，自行翻查 HAL 库的定义即可
            FlashStatus = HAL_FLASH_Program(FLASH_TYPEPROGRAM_HALFWORD, PARAMFLASH_BASE_ADDRESS+i, temp);
            if (FlashStatus != HAL_OK)
                return 0;
        }
        HAL_FLASH_Lock();//锁住 Flash
        return 1;
}

/*
**********************************************************************
 * 函数原型：uint8_t Flash_Read(uint8_t *addr, uint16_t len)
 * 功    能：读取 Flash
 * 输    入：addr 需要写入结构体的地址，len 结构体长度
 * 输    出：读取是否成功
 * 参    数：uint8_t *addr, uint16_t len
**********************************************************************
*/
uint8_t Flash_Read(uint8_t *addr, uint16_t len)
{
    for(uint16_t i=0; i<len; i=i+2)
    {
        uint16_t temp;
        if(i+1 <= len-1)
        {
            temp = (*(__IO uint16_t*)(PARAMFLASH_BASE_ADDRESS+i));//*(__IO uint16_t *)是读取该地址的参数值,其值为 16 位数据,一次读取两个字节
            addr[i] = BYTE0(temp);
            addr[i+1] = BYTE1(temp);
        }
        else
        {
            temp = (*(__IO uint16_t*)(PARAMFLASH_BASE_ADDRESS+i));
            addr[i] = BYTE0(temp);
        }
    }
    return 1;
}
#include "Drv_NTC.h"

/**********局部变量**********/
const uint16_t R100K_TAB[] = //R25=100K B25=3950K -25-150
{
    122,//-17
```

```
129,//-16
137,//-15
145,//-14
153,//-13
161,//-12
170,//-11
180,//-10
189,//-9
200,//-8
210,//-7
221,//-6
233,//-5
245,//-4
258,//-3
271,//-2
284,//-1
298,//0
313,//1
328,//2
344,//3
361,//4
378,//5
395,//6
413,//7
432,//8
452,//9
472,//10
493,//11
514,//12
536,//13
559,//14
582,//15
606,//16
631,//17
656,//18
682,//19
709,//20
736,//21
764,//22
793,//23
822,//24
852,//25
882,//26
914,//27
945,//28
977,//29
1010,//30
1044,//31
1077,//32
1112,//33
```

```
1147,//34
1182,//35
1218,//36
1254,//37
1290,//38
1327,//39
1364,//40
1402,//41
1440,//42
1478,//43
1516,//44
1554,//45
1593,//46
1632,//47
1670,//48
1709,//49
1748,//50
1787,//51
1826,//52
1865,//53
1904,//54
1943,//55
1981,//56
2020,//57
2058,//58
2096,//59
2134,//60
2171,//61
2209,//62
2246,
2282,
2319,
2355,
2390,
2426,
2460,
2495,
2529,
2562,
2595,
2628,
2660,
2692,
2723,
2754,
2784,
2813,
2843,
2871,
2899,
```

```
    2927,
    2954,
    2981,
    3007,
    3032,
    3057,
    3082,
    3106,
    3129,
    3152,
    3175,
    3197,
    3218,
    3240,
    3260,
    3280,
    3300,
    3319,//109
    3338,//108
    3356,//107
    3374,//106
    3392,//105
    3409,//106
    3426,//107
    3442,//108
    3458,//109
    3473,//110
    3489,//111
    3503,//112
    3518,//113
    3532,//114
    3545,//115
    3559,//116
    3572,//117
    3585,//118
    3597,//119
    3609,//120
    3621,//121
    3632,//122
    3643,//123
    3654,//124
    3665,//125
    3675,//126
    3685,//127
    3695,//128
    3705,//129
    3714,//130
};

/**********全局变量**********/
```

```
#define AD_LEN 2//DMA 获取长度
uint16_t ADC_Val[AD_LEN];//adc 的值  0:温度 ad 值。  1：ad 值;
uint16_t ADC_Val1,ADC_Val2;
float    Calibration_Temp;

/*
*********************************************************************
 * 函数原型：int Filter_ADC(void)
 * 功      能：滑动平均值滤波
 * 输      出：滤波后的值
*********************************************************************
*/
#define N 50//采集 100 次
int ADCvalue_Buf[N];//用于储存采集到的 adc 值
int i = 0;
int Filter_ADC(void)
{
      char count;
      long sum = 0;

      ADCvalue_Buf[i++] = ADC_Val[0];//将 adc 的值储存

      if (i == N)//加入读了 100 组就从新开始
      {
            i = 0;
      }
      for (count = 0; count < N; count++)
      {
            sum += ADCvalue_Buf[count];//100 组相加
      }
      if(ADCvalue_Buf[N-1] == 0)//如果没有读到 100 组就用第一次读到的数
            return ADCvalue_Buf[0];
      else//读到 100 组后
            return (int)(sum / N);//输出平均值
}

/*
*********************************************************************
 * 函数原型：int Filter_ADC1(void)
 * 功      能：滑动平均值滤波
 * 输      出：滤波后的值
*********************************************************************
*/
int ADCvalue_Buf1[N];//用于储存采集到的 adc 值
int j = 0;
int Filter_ADC1(void)
{
      char count;
      long sum = 0;
```

```
    ADCvalue_Buf1[j++] = ADC_Val[1];//将 adc 的值储存

    if (j == N)//加入读了 100 组就从新开始
    {
        j = 0;
    }
    for (count = 0; count < N; count++)
    {
        sum += ADCvalue_Buf1[count];//100 组相加
    }
    if(ADCvalue_Buf1[N-1] == 0)//如果没有读到 100 组就用第一次读到的数
        return ADCvalue_Buf1[0];
    else//读到 100 组后
        return (int)(sum / N);//输出平均值
}

/*
**********************************************************************
 * 函数原型：uint16_t Get_Ntc_Temp(uint16_t value_adc)
 * 功    能：计算出 Ntc 的温度
 * 输    入: value_adc:adc 读到的值
 * 参    数：uint16_t value_adc
**********************************************************************
*/
#define SHORT_CIRCUIT_THRESHOLD 15
#define OPEN_CIRCUIT_THRESHOLD 4096
uint8_t index_l, index_r ;
uint16_t Get_Ntc_Temp(uint16_t value_adc)
{
    uint8_t R100k_Tab_Size = 148;
    int temp = 0;
    if(value_adc <= SHORT_CIRCUIT_THRESHOLD)
    {
        return 1;
    }
    else if(value_adc >= OPEN_CIRCUIT_THRESHOLD)
    {
        return 2;
    }
    else if(value_adc < R100K_TAB[0])
    {
        return 3;
    }

    else if(value_adc > R100K_TAB[R100k_Tab_Size - 1])
    {
        return 4;
    }

    index_l = 0;
```

```
    index_r = R100k_Tab_Size - 1;
    for(; index_r - index_l > 1;)
    {
        if((value_adc <= R100K_TAB[index_r]) && (value_adc > R100K_TAB[(index_l + index_r) % 2 == 0 ? (index_l + index_r) / 2 : (index_l + index_r) / 2 ]))
        {
            index_l = (index_l + index_r) % 2 == 0 ? (index_l + index_r) / 2 : (index_l + index_r) / 2 ;
        }
        else
        {
            index_r = (index_l + index_r) / 2;
        }
    }
    if(R100K_TAB[index_l] == value_adc)
    {
        temp = (((int)index_l) - 17) * 10; //rate *10
    }
    else if(R100K_TAB[index_r] == value_adc)
    {
        temp = (((int)index_r) - 17) * 10; //rate *10
    }
    else
    {
        if(R100K_TAB[index_r] - R100K_TAB[index_l] == 0)
        {
            temp = (((int)index_l) - 17) * 10; //rate *10
        }
        else
        {
            temp = (((int)index_l) - 17) * 10 + ((value_adc - R100K_TAB[index_l] ) * 100 + 5) / 10 / (R100K_TAB[index_r] - R100K_TAB[index_l]);
        }
    }

    /******************温度补偿**********************/
    return temp;
}

/*
*********************************************************************
 * 函数原型：uint16_t Get_ADCVal(uint16_t temp)
 * 功　　能：计算当前温度的 adc 值
 * 输　　入：temp：当前温度
 * 输　　出：ADC 值
 * 参　　数：uint16_t temp
*********************************************************************
*/
uint16_t Get_ADCVal(uint16_t temp)
{
```

```
    uint16_t adc,adc1;
    float val2;
    uint16_t val3,val1;

    val3 = (temp/10)+17;
    val2 = (float)(temp%10)/10;

    val1 = ((temp+10)/10)+17;

    adc = R100K_TAB[val3];
    adc1 = R100K_TAB[val1];
    return adc+((adc1-adc)*val2)+((adc1-adc)*0.1)-1;
}


/*
**********************************************************************
 * 函数原型：void Calibration(uint16_t temp,uint16_t temp1)
 * 功    能：温度校准计算
 * 输    入：temp：40 的温度，temp1：60 的温度
 * 参    数：uint16_t temp,uint16_t temp1
**********************************************************************
*/
void Calibration(uint16_t temp,uint16_t temp1)
{
    float Cal1,Cal2;
    uint16_t ADCv1,ADCv2;

    ADCv1 = Get_ADCVal(temp);
    Cal1 = 1364/(float)ADCv1;
    ADCv2 = Get_ADCVal(temp1);
    Cal2 = 2134/ADCv2;
    Calibration_Temp = (float)(Cal1 + Cal2)/2;
    Param.Calibration_Temp = Calibration_Temp;//温度校准系数
    Save_Param_En = 1;//保存

}

/*
**********************************************************************
 * 函数原型：void ADCDMA_Init(void)
 * 功    能：ADC 和 DMA 的初始化
**********************************************************************
*/
void ADCDMA_Init(void)
{
    HAL_TIM_Base_Start_IT(&htim3);//开启 TIM3 的定时，用于刷新
    HAL_ADC_Start_DMA(&hadc,(uint32_t *)ADC_Val, AD_LEN);//用 DMA 获取 adc 值
    HAL_ADCEx_Calibration_Start(&hadc);
    for(uint8_t i=0;i<10;i++)
```

```
    {
        Heat_Temp.Rel_Temp = Get_Ntc_Temp(ADC_Val[0]*Calibration_Temp);//计算温度
        Cool_Temp.Rel_Temp = Get_Ntc_Temp(ADC_Val[1]/Calibration_Temp);//计算温度
        HAL_Delay(10);
    }
}

/*
**********************************************************************
 * 函数原型：  void Read_Temp(float dT)
 * 功      能：  读取温度-10ms
**********************************************************************
*/
void Read_Temp(float dT)
{
    static float T;
    T += dT;
    ADC_Val1 = Filter_ADC();//滤波获取 adc 的滑动平均值
    ADC_Val2 = Filter_ADC1();
    if(T >= 2.0f)//1S
    {
        if(sys.Calibration)
        {
            Heat_Temp.Rel_Temp = Get_Ntc_Temp(ADC_Val1);//计算温度
        }
        else
        {
            Heat_Temp.Rel_Temp  =  Get_Ntc_Temp(ADC_Val1*Calibration_Temp);//计算温
度
        }
        Cool_Temp.Rel_Temp = Get_Ntc_Temp(ADC_Val2/Calibration_Temp);//计算温度
        T = 0;

    }
}
#include "SysTick.h"

/*
**********************************************************************
 * 函数原型：void Delay_us(__IO uint32_t delay)
 * 功      能：微妙延时函数
**********************************************************************
*/
#define CPU_FREQUENCY_MHZ     48//STM32 时钟主频
void Delay_us(__IO uint32_t delay)
{
    int last,curr,val;
    int temp;
    while(delay != 0)
    {
```

```
            temp = delay > 900 ? 900 : delay;
            last = SysTick->VAL;
            curr = last - CPU_FREQUENCY_MHZ * temp;
            if (curr >= 0)
            {
                do
                {
                    val = SysTick->VAL;
                }
                while ((val < last) && (val >= curr));
            }
            else
            {
                curr += CPU_FREQUENCY_MHZ * 1000;
                do
                {
                    val = SysTick->VAL;
                }
                while ((val <= last) || (val > curr));
            }
            delay -= temp;
        }
}
#include "Show.h"

/**********全局变量声明******/
float Twinkle_Time;//闪烁时间

/**********局部变量声明******/
uint8_t Tab[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};//0·9
uint8_t  HeatTemp_ShowFlag,CoolTemp_ShowFlag,Time_ShowFlag;//加热温度闪烁,制冷温度闪烁，时间闪烁
uint8_t HeatTemp_XSDFlag,CoolTemp_XSDFlag;//单加热小数点闪烁，制冷小数点闪烁

/*
*********************************************************************
 * 函数原型：static void Check_ShowFlag(float dT)
 * 功      能：闪烁检测
 * 输      入: dT:执行周期
 * 参      数：float dT
 * 调      用：内部调用
*********************************************************************
*/
static void Check_ShowFlag(float dT)
{
    static float T;
    if(sys.Calibration)
    {
        if(T == 0)
        {
```

```
            T += dT;
            if(sys.SetMode_Option == 4)//设置 CoolTemp 时
            {
                HeatTemp_ShowFlag = 0;//加热温度常亮
                CoolTemp_ShowFlag = !CoolTemp_ShowFlag;//制冷温度闪烁
                Time_ShowFlag = 0;//时间常亮
            }
        }
        else
        {
            T += dT;
            if(T >=0.5f)
                T = 0;
        }
    }
    else
    {
        if(LongPress == 1)
        {
            if(sys.SetMode_Option == 1)//设置 HeatTemp 时
            {
                HeatTemp_ShowFlag = 1;//加热温度常灭
                CoolTemp_ShowFlag = 0;//制冷温度常亮
                Time_ShowFlag = 0;//时间常亮
            }
            else if(sys.SetMode_Option == 2)//设置 CoolTemp 时
            {
                HeatTemp_ShowFlag = 0;//加热温度常亮
                CoolTemp_ShowFlag = 1;//制冷温度常灭
                Time_ShowFlag = 0;//时间常亮
            }
            else if(sys.SetMode_Option == 3)//设置 CoolTemp 时
            {
                HeatTemp_ShowFlag = 0;//加热温度常亮
                CoolTemp_ShowFlag = 0;//制冷温度常亮
                Time_ShowFlag = 1;//时间常灭
            }
            return;
        }
        if(Twinkle_Time && Key_Status==0)//闪烁和没有操作按键时
        {
            if(T == 0)
            {
                T += dT;
                Twinkle_Time -= 0.5;//闪烁计时
                if(Twinkle_Time == 0)//如果闪烁结束
                {
                    sys.TempOption_Mun = 0;//温度设置位数复位
                    sys.TimeOption_Mun = 0;//时间设置位数复位
                    sys.SetMode_Option = 0;//模式选择清零
```

```
                    }
                    if(sys.SetMode_Option == 1)//设置 HeatTemp 时
                    {
                        HeatTemp_ShowFlag = !HeatTemp_ShowFlag;//加热温度闪烁
                        CoolTemp_ShowFlag = 0;//制冷温度常亮
                        Time_ShowFlag = 0;//时间常亮
                    }
                    else if(sys.SetMode_Option == 2)//设置 CoolTemp 时
                    {
                        HeatTemp_ShowFlag = 0;//加热温度常亮
                        CoolTemp_ShowFlag = !CoolTemp_ShowFlag;//制冷温度闪烁
                        Time_ShowFlag = 0;//时间常亮
                    }
                    else if(sys.SetMode_Option == 3)//设置 Time 时
                    {
                        HeatTemp_ShowFlag = 0;//加热温度常亮
                        CoolTemp_ShowFlag = 0;//制冷温度常亮
                        Time_ShowFlag = !Time_ShowFlag;//时间闪烁
                    }
                }
                else
                {
                    T += dT;
                    if(T >=0.5f)
                        T = 0;
                }
            }
            else
            {
                HeatTemp_ShowFlag = 0;//加热温度常亮
                CoolTemp_ShowFlag = 0;//制冷温度常亮
                Time_ShowFlag = 0;//时间常亮
                T = 0;
            }
        }
    }

/*
**********************************************************************
 * 函数原型：static void Heating_Twinkle(float dT)
 * 功      能：小数点闪烁检测
 * 输      入: dT:执行周期
 * 参      数：float dT
 * 调      用：内部调用
**********************************************************************
*/
static void Heating_Twinkle(float dT)
{
    static float T,T1;
    if(Heat_Temp.ADDMode == 0)
```

```
    {
        T += dT;
        if(T >= 0.5f)
        {
            T = 0;
            HeatTemp_XSDFlag = !HeatTemp_XSDFlag;//单加热小数点闪烁
        }
    }
    else
    {
        T = 0;
        HeatTemp_XSDFlag = 0;//常亮
    }

    if(Cool_Temp.ADDMode == 0)
    {
        T1 += dT;
        if(T1 >= 0.5f)
        {
            T1 = 0;
            CoolTemp_XSDFlag = !CoolTemp_XSDFlag;//单加热小数点闪烁
        }
    }
    else
    {
        T1 = 0;
        CoolTemp_XSDFlag = 0;//常亮
    }
}

/*
**********************************************************************
 * 函数原型：void Twinkle(float dT)
 * 功      能：闪烁函数
**********************************************************************
*/
void Twinkle(float dT)
{
    Check_ShowFlag(dT);//闪烁检测
    Heating_Twinkle(dT);//小数点闪烁
}

/*
**********************************************************************
 * 函数原型：void Display_HeatTemp(int16_t temp)
 * 功      能：显示加热区域温度
 * 输      入: temp：加热区域的温度
 * 参      数：int16_t temp
**********************************************************************
*/
```

```
void Display_HeatTemp(int16_t temp)
{
    uint8_t seg1,seg2,seg3;
    seg1=0;seg2=0;seg3=0;
    uint16_t Val;//用于百十个取出来的数字

    if(temp >= 0)
    {
        /***********temp 百位***********/
        if(sys.TempOption_Mun == 1 && HeatTemp_ShowFlag)//假如在位数为百位并且温度在闪烁
            seg1 = 0x00;//不显示
        else
        {
            if(temp > 99)//大于 99 时
            {
                Val=temp/100;//取出百位
                if(temp > 999)//大于 999 时
                    Val=Val%10;//取出百位
                seg1 = Tab[Val];
            }
        }

        /***********temp 十位***********/
        if(sys.TempOption_Mun == 0 && HeatTemp_ShowFlag)//假如在位数为十位并且温度在闪烁
            seg2 = 0x00;//不显示
        else
        {
            if(temp > 9)//大于 9 时
            {
                Val=temp/10;//取出十位
                if(temp > 99)//大于 999 时
                    Val=Val%10;//取出十位
                seg2 = Tab[Val];
            }
        }

        /***********temp 小数位***********/
        if(sys.TempOption_Mun == 2 && HeatTemp_ShowFlag)//假如在位数为小数位并且温度在闪烁
            seg3 = 0x00;//不显示
        else
        {
            Val=temp%10;//取出个位
            seg3 = Tab[Val];
        }
    }
    else
    {
```

```
            /***********temp 负号***********/
            seg1 = 0x40;

            /***********temp 十位***********/
            if((-temp) > 9)//大于 9 时
            {
                Val=(-temp)/10;//取出十位
                if((-temp) > 99)//大于 999 时
                    Val=Val%10;//取出十位
                seg2 = Tab[Val];
            }

            /***********temp 小数位***********/
            Val=(-temp)%10;//取出个位
            seg3 = Tab[Val];
        }

        /***********temp 小数点*********/
        if((sys.SetMode_Option == 0 || sys.SetMode_Option == 3 || sys.SetMode_Option == 4) &&
HeatTemp_XSDFlag)//假如在加热的情况下，并且小数点闪烁
        {
            seg2&=0x7F;seg2|=0x00;//不显示
        }
        else
        {
            seg2&=0x7F;seg2|=0x80;
        }

        /************************************/
        GN1640_Write_DATA(0, seg1);
        GN1640_Write_DATA(1, seg2);
        GN1640_Write_DATA(2, seg3);
}

/*
*********************************************************************
 * 函数原型：void Display_CoolTemp(int16_t temp)
 * 功      能：显示制冷区域温度
 * 输      入: temp：制冷区域的温度
 * 参      数：int16_t temp
*********************************************************************
*/
void Display_CoolTemp(int16_t temp)
{
        uint8_t seg1,seg2,seg3;
        seg1=0;seg2=0;seg3=0;
        uint16_t Val;//用于百十个取出来的数字

        if(temp >= 0)
        {
```

```
        /***********temp 百位***********/
        if(sys.TempOption_Mun == 1 && CoolTemp_ShowFlag)//假如在位数为百位并且温度在闪烁
            seg1 = 0x00;//不显示
        else
        {
            if(temp > 99)//大于 99 时
            {
                Val=temp/100;//取出百位
                if(temp > 999)//大于 999 时
                    Val=Val%10;//取出百位
                seg1 = Tab[Val];
            }
        }

        /***********temp 十位***********/
        if(sys.TempOption_Mun == 0 && CoolTemp_ShowFlag)//假如在位数为十位并且温度在闪烁
            seg2 = 0x00;//不显示
        else
        {
            if(temp > 9)//大于 9 时
            {
                Val=temp/10;//取出十位
                if(temp > 99)//大于 999 时
                    Val=Val%10;//取出十位
                seg2 = Tab[Val];
            }
        }

        /***********temp 小数位***********/
        if(sys.TempOption_Mun == 2 && CoolTemp_ShowFlag)//假如在位数为小数位并且温度在闪烁
            seg3 = 0x00;//不显示
        else
        {
            Val=temp%10;//取出个位
            seg3 = Tab[Val];
        }
    }
    else
    {
        /***********temp 百位***********/
        seg1 = 0x40;

        /***********temp 十位***********/
        if((-temp) > 9)//大于 9 时
        {
            Val=(-temp)/10;//取出十位
            if((-temp) > 99)//大于 999 时
```

```
                Val=Val%10;//取出十位
            seg2 = Tab[Val];
        }

        /***********temp 小数位***********/
        Val=(-temp)%10;//取出个位
        seg3 = Tab[Val];
    }

    /***********temp 小数点*********/
    if((sys.SetMode_Option == 0 || sys.SetMode_Option == 3) && CoolTemp_XSDFlag)//假如在加热的情况下，并且小数点闪烁
    {
        seg2&=0x7F;seg2|=0x00;//不显示
    }
    else
    {
        seg2&=0x7F;seg2|=0x80;
    }

    /************************************/
    GN1640_Write_DATA(3, seg1);
    GN1640_Write_DATA(4, seg2);
    GN1640_Write_DATA(5, seg3);
}

/*
*********************************************************************
 * 函数原型：void Display_Time(uint8_t flag,int16_t time)
 * 功      能：显示时间
 * 输      入: flag：是否在校准界面  time：时间
 * 参      数：uint8_t flag,int16_t time
*********************************************************************
*/
void Display_Time(uint8_t flag,int16_t time)
{
    uint8_t seg1,seg2,seg3,seg4;
    seg1=0;seg2=0;seg3=0;seg4=0;
    uint8_t SM,M,SS,S;//时间的单位取值

    if(flag)
    {
        seg1 = 0x77;//A
        seg2 = 0x5E;//d
        seg3 = 0x0E;//j
        seg4 = 0x78;//t
    }
    else
    {
        /***********设定时间***********/
```

```
        SM=time/60/10;//计算十位单位的分钟数
        M=time/60%10;//计算个位单位的分钟数
        SS=time%60/10;//计算十分位单位的秒钟数
        S=time%60%10;//计算十分位单位的秒钟数

        /***********十分时间***********/
        if(sys.TimeOption_Mun == 1 && Time_ShowFlag)//假如在位数为十分并且时间在闪
烁
            seg1 = 0x00;//不显示
        else
        {
            seg1 = Tab[SM];
        }

        /***********分钟时间***********/
        if(sys.TimeOption_Mun == 0 && Time_ShowFlag)//假如在位数为分并且时间在闪烁
            seg2 = 0x00;//不显示
        else
        {
            seg2 = Tab[M];
        }

        /***********十秒时间***********/
        if(sys.TimeOption_Mun == 3 && Time_ShowFlag)//假如在位数为十秒分并且时间在
闪烁
            seg3 = 0x00;//不显示
        else
        {
            seg3 = Tab[SS];
        }
        /***********秒钟时间***********/
        if(sys.TimeOption_Mun == 2 && Time_ShowFlag)//假如在位数为秒并且时间在闪烁
            seg4 = 0x00;//不显示
        else
        {
            seg4 = Tab[S];
        }

        /***********time 冒号***********/
        seg2&=0x7F;seg2|=0x80;
    }

    /************************************/
    GN1640_Write_DATA(6, seg1);
    GN1640_Write_DATA(7, seg2);
    GN1640_Write_DATA(8, seg3);
    GN1640_Write_DATA(9, seg4);
}

/*
```

```
**************************************************************
 * 函数原型：void Deal_Temp(float dT)
 * 功      能：温度处理
**************************************************************
*/
void Deal_Temp(float dT)
{
    static float T,T1;
    if(sys.Calibration)
    {
        if(sys.Calibration_Step == 0)
        {
            if(Heat_Temp.Rel_Temp == 400)
            {
                if(!Heat_Temp.ADDMode)
                    T += dT;
                if(T >= 2.0f)
                {
                    T = 0;
                    Heat_Temp.ADDMode = 1;
                }
            }
            else if(400 - Heat_Temp.Rel_Temp > 5 || 400 - Heat_Temp.Rel_Temp < -5 )
            {
                if(Heat_Temp.ADDMode)
                    T += dT;
                if(T >= 2.0f)
                {
                    T = 0;
                    Heat_Temp.ADDMode = 0;
                }
            }
        }
        else
        {
            if(Heat_Temp.Rel_Temp == 600)
            {
                if(!Heat_Temp.ADDMode)
                    T += dT;
                if(T >= 2.0f)
                {
                    T = 0;
                    Heat_Temp.ADDMode = 1;
                }
            }
            else if(600 - Heat_Temp.Rel_Temp > 5 || 600 - Heat_Temp.Rel_Temp < -5 )
            {
                if(Heat_Temp.ADDMode)
                    T += dT;
                if(T >= 2.0f)
```

```
                    {
                        T = 0;
                        Heat_Temp.ADDMode = 0;
                    }
                }
            }
        }
        else
        {
            if(Heat_Temp.Rel_Temp == Heat_Temp.Ctrl_Temp)
            {
                if(!Heat_Temp.ADDMode)
                    T += dT;
                if(T >= 2.0f)
                {
                    T = 0;
                    Heat_Temp.ADDMode = 1;
                }
            }
            else if(Heat_Temp.Ctrl_Temp - Heat_Temp.Rel_Temp > 5 || Heat_Temp.Ctrl_Temp - Heat_Temp.Rel_Temp < -5 )
            {
                if(Heat_Temp.ADDMode)
                    T += dT;
                if(T >= 2.0f)
                {
                    T = 0;
                    Heat_Temp.ADDMode = 0;
                }
            }
        }

        if(Cool_Temp.Rel_Temp == Cool_Temp.Ctrl_Temp)
        {
            if(!Cool_Temp.ADDMode)
                T1 += dT;
            if(T1 >= 2.0f)
            {
                T1= 0;
                Cool_Temp.ADDMode = 1;
            }
        }
        else if(Cool_Temp.Ctrl_Temp - Cool_Temp.Rel_Temp > 5 || Cool_Temp.Ctrl_Temp - Cool_Temp.Rel_Temp < -5 )
        {
            if(Cool_Temp.ADDMode)
                T1 += dT;
            if(T1 >= 2.0f)
            {
                T1= 0;
```

```
                Cool_Temp.ADDMode = 0;
            }
        }

}

/*
*********************************************************************
 * 函数原型：  void Show_Display(void)
 * 功      能：  显示屏幕内容
*********************************************************************
*/
void Show_Display(void)
{
    static uint8_t flag = 0;

    if(sys.SetMode_Option >0 && sys.SetMode_Option < 3 && sys.Calibration==0)
    {
        Heat_Temp.Display_Temp = Heat_Temp.Set_Temp;
        Cool_Temp.Display_Temp = Cool_Temp.Set_Temp;
    }
    else
    {
        if(sys.Calibration)
        {
            if(Heat_Temp.ADDMode == 0)
                Heat_Temp.Display_Temp = Heat_Temp.Rel_Temp;
            else
            {
                if(sys.Calibration_Step == 0)
                    Heat_Temp.Display_Temp = 400;
                else
                    Heat_Temp.Display_Temp = 600;
            }

            if(sys.Calibration_Step == 0)
                Cool_Temp.Display_Temp = sys.Calibration_Temp1;
            else if(sys.Calibration_Step == 1)
                Cool_Temp.Display_Temp = sys.Calibration_Temp2;
        }
        else
        {
            if(Heat_Temp.ADDMode == 0)
                Heat_Temp.Display_Temp = Heat_Temp.Rel_Temp;
            else
                Heat_Temp.Display_Temp = Heat_Temp.Ctrl_Temp;

            if(Cool_Temp.ADDMode == 0)
                Cool_Temp.Display_Temp = Cool_Temp.Rel_Temp;
            else
```

```
                    Cool_Temp.Display_Temp = Cool_Temp.Ctrl_Temp;
            }
        }

        if(Time.TimeDisplay_Flag)//正计时
        {
            Time.Display_Time = Time.Count_Time;
        }
        else//倒计时
        {
            if(sys.SetMode_Option == 3)
                Time.Display_Time = Time.Set_Time;
            else
                Time.Display_Time = Time.Ctrl_Time;
            if(sys.Calibration)
                flag = 1;
            else
                flag = 0;
        }
        Display_HeatTemp(Heat_Temp.Display_Temp);
        Display_CoolTemp(Cool_Temp.Display_Temp);
        Display_Time(flag,Time.Display_Time);
}
#include "Param.h"

/**********结构体**********/
struct _Save_Param_ Param;//原始数据

/**********全局变量声明******/
uint8_t Save_Param_En;

/*
*********************************************************************
 * 函数原型：void Param_Reset(void)
 * 功      能：初始化硬件中的参数
*********************************************************************
*/
void Param_Reset(void)
{
        Param.Flash_Check_Start = FLASH_CHECK_START;

        Param.Heat_Temp = 450;//初始温度 45℃
        Param.Cool_Temp = 260;//初始温度 26℃
        Param.CountDown_Time = 1200;//20 分钟

        Param.Calibration_Temp = 1.0;//温度校准系数

        Param.Flash_Check_End    = FLASH_CHECK_END;
}
```

```
/*
*********************************************************************
 * 函数原型：  void Param_Save(void)
 * 功      能：  保存硬件中的参数
*********************************************************************
*/
void Param_Save(void)
{
    Flash_Write((uint8_t *)(&Param),sizeof(Param));
}

/*
*********************************************************************
 * 函数原型：void Param_Read(void)
 * 功      能：读取硬件中的参数，判断是否更新
*********************************************************************
*/
void Param_Read(void)
{
    Flash_Read((uint8_t *)(&Param),sizeof(Param));

    //板子从未初始化
    if(Param.Flash_Check_Start != FLASH_CHECK_START || Param.Flash_Check_End != FLASH_CHECK_END)
    {
        Param_Reset();
        Heat_Temp.Set_Temp = Param.Heat_Temp;//加热区域温度
        Cool_Temp.Set_Temp = Param.Cool_Temp;//制冷区域温度
        Time.Set_Time = Param.CountDown_Time;//倒计时时间
        Calibration_Temp = Param.Calibration_Temp;//温度校准系数
        SetOK_Flag = 1;
        Save_Param_En = 1;
    }
    else
    {
        Heat_Temp.Set_Temp = Param.Heat_Temp;//加热区域温度
        Cool_Temp.Set_Temp = Param.Cool_Temp;//制冷区域温度
        Time.Set_Time = Param.CountDown_Time;//倒计时时间
        Calibration_Temp = Param.Calibration_Temp;//温度校准系数
        SetOK_Flag = 1;
    }

    //保存参数
    if(Save_Param_En)
    {
        Save_Param_En = 0;
        Param_Save();
    }
}
```

```
/*
**********************************************************************
 * 函数原型：void Param_Save_Overtime(float dT)
 * 功      能：保存标志位置 1，0.5s 后保存
**********************************************************************
*/
void Param_Save_Overtime(float dT)
{
    static float time;

    if(Save_Param_En)
    {
        time += dT;

        if(time >= 0.5f)
        {
            Param_Save();
            Save_Param_En = 0;
        }
    }
    else
        time = 0;
}
#include "SetVal.h"

/**********全局变量声明******/
uint8_t SetOK_Flag;//检测是否按下按键

/*
**********************************************************************
 * 函数原型：void Check_Set(float dT)
 * 功      能：检测设置
**********************************************************************
*/
void Check_Set(float dT)
{
    if(Key_Status != 0)
    {
        SetOK_Flag = 1;//检测到波动旋钮，等待退出设置模式
    }
    if(SetOK_Flag == 1)
    {
        if(sys.SetMode_Option == 0)//在设定好后
        {
            if(Heat_Temp.Ctrl_Temp != Heat_Temp.Set_Temp)//判断控制温度和设定温度是
不是不一样
            {
                Heat_Temp.Ctrl_Temp  =  Heat_Temp.Set_Temp;//把设定温度赋值给控制温
度
                Param.Heat_Temp = Heat_Temp.Set_Temp;//赋值到保存参数
```

```
                    HeatTemp_Val.SumError = 0;//Pid 的积分和清零
                    Heat_Temp.ADDMode = 0;//温度处理清零
                }
                if(Cool_Temp.Ctrl_Temp != Cool_Temp.Set_Temp)//判断控制温度和设定温度是
不是不一样
                {
                    Cool_Temp.Ctrl_Temp  =  Cool_Temp.Set_Temp;//把设定温度赋值给控制温
度
                    Param.Cool_Temp = Cool_Temp.Set_Temp;//赋值到保存参数
                    CoolTemp_Val.SumError = 0;//Pid 的积分和清零
                    Cool_Temp.ADDMode = 0;//温度处理清零
                }
                if(Time.Ctrl_Time != Time.Set_Time)//实际时间不等于设定时间
                {
                    Time.Ctrl_Time = Time.Set_Time;//把设定时间赋值给控制时间
                    Param.CountDown_Time = Time.Set_Time;//赋值到保存参数
                }
                Save_Param_En = 1;//保存
                SetOK_Flag = 0;
            }
        }
}
#include "PID.h"

/**********结构体***********/
PID_val_t HeatTemp_Val;//pid 数据结构
PID_arg_t HeatTemp_Arg;//pid 数据系数
PID_val_t CoolTemp_Val;//pid 数据结构
PID_arg_t CoolTemp_Arg;//pid 数据系数

/*
*********************************************************************
 * 函数原型：  void PID_Init(void)
 * 功      能：  pid 系数初始化
*********************************************************************
*/
void PID_Init(void)
{
    HeatTemp_Arg.Kp=2.6;
    HeatTemp_Arg.Ki=25;
    HeatTemp_Arg.Kd=0;

    CoolTemp_Arg.Kp=45;
    CoolTemp_Arg.Ki=25;//25
    CoolTemp_Arg.Kd=0;
}

/*
*********************************************************************
 * 函数原型：void PID_Temp(
```

```
                int16_t Expect,     //期望值（设定值）
                int16_t Feedback, //反馈值（实际值）
                PID_arg_t *pid_arg,//PID 参数结构体
                PID_val_t *pid_val)//PID 数据结构体
 * 功      能：  PID 控制
 * 输      入:  Expect,     //期望值（设定值）
                 Feedback, //反馈值（实际值）
                 PID_arg_t *pid_arg,//PID 参数结构体
                 PID_arg_t *pid_arg,//PID 参数结构体
*****************************************************************
*/
void PID_Temp(
                int16_t Expect,     //期望值（设定值）
                int16_t Feedback, //反馈值（实际值）
                PID_arg_t *pid_arg,//PID 参数结构体
                PID_val_t *pid_val)//PID 数据结构体
{
    pid_val->Error = Expect - Feedback;//当前误差
    if(pid_val->Error > - 45 &&   pid_val->Error < 45)
        pid_val->SumError += pid_val->Error;
    else
        pid_val->SumError = 0;

    if(HeatTemp_Val.Error< 0 && HeatTemp_Val.SumError <0)
        HeatTemp_Val.SumError = 0;
    pid_val->D_Error = pid_val->LastError - pid_val->PrevError;//误差偏差
    pid_val->PrevError = pid_val->LastError;//保存上一次误差
    pid_val->LastError = pid_val->Error;//保存误差
    pid_val->Integral = pid_arg->Ki*pid_val->SumError/10000.0;

    if(pid_val->Integral >400)
    {
        pid_val->Integral = 400;
    }
    pid_val->Out                                                    =
pid_arg->Kp*pid_val->Error+pid_val->Integral+pid_arg->Kd*pid_val->D_Error;
    if(pid_val->Out> 999)
        pid_val->Out= 999;
    if(pid_val->Out< -999)
        pid_val->Out= -999;
}
#include "Ctrl_Scheduler.h"

uint16_t T_cnt_1ms=0,
          T_cnt_2ms=0,
          T_cnt_4ms=0,
          T_cnt_6ms=0,
          T_cnt_10ms=0,
          T_cnt_12ms=0,
          T_cnt_20ms=0,
```

```
            T_cnt_50ms=0,
            T_cnt_100ms=0,
            T_cnt_200ms=0,
            T_cnt_500ms=0,
            T_cnt_1S=0;

void Loop_Check(void)
{
    T_cnt_1ms++;
    T_cnt_2ms++;
    T_cnt_4ms++;
    T_cnt_6ms++;
    T_cnt_10ms++;
    T_cnt_12ms++;
    T_cnt_20ms++;
    T_cnt_50ms++;
    T_cnt_100ms++;
    T_cnt_200ms++;
    T_cnt_500ms++;
    T_cnt_1S++;

    Sys_Loop();
}

static void Loop_1ms(float dT)
{

}

static void Loop_2ms(float dT)
{

}

static void Loop_4ms(float dT)
{

}

static void Loop_6ms(float dT)
{

}

static void Loop_10ms(float dT)
{
    Key_Scan(dT);//按键检测
    Read_Temp(dT);//读取温度
    Check_Set(dT);//检测设置
}
```

```
static void Loop_12ms(float dT)
{

}

static void Loop_20ms(float dT)
{
    Deal_Temp(dT);//温度处理
}

static void Loop_50ms(float dT)
{
    Temp_Control(dT);//温度加热控制
}

static void Loop_100ms(float dT)
{
    Buzzer_Status(dT);//蜂鸣器的状态检测
    Cheak_TimeDown(dT);//时间倒计时检测
    Twinkle(dT);//闪烁函数
    Param_Save_Overtime(dT);//保存标志位置
}

static void Loop_200ms(float dT)
{

}

static void Loop_500ms(float dT)
{
    Check_Press(dT);//检测按键按下状态

}

static void Loop_1S(float dT)
{

}

void Sys_Loop(void)
{
    if(T_cnt_1ms >= 1) {
        Loop_1ms(0.001f);
        T_cnt_1ms = 0;
    }
    if(T_cnt_2ms >= 2) {
        Loop_2ms(0.002f);
        T_cnt_2ms = 0;
    }
```

```
    if(T_cnt_4ms >= 4) {
        Loop_4ms(0.004f);
        T_cnt_4ms = 0;
    }
    if(T_cnt_6ms >= 6) {
        Loop_6ms(0.006f);
        T_cnt_6ms = 0;
    }
    if(T_cnt_10ms >= 10) {
        Loop_10ms(0.01f);
        T_cnt_10ms = 0;
    }
    if(T_cnt_12ms >= 12) {
        Loop_12ms(0.012f);
        T_cnt_12ms = 0;
    }
    if(T_cnt_20ms >= 20) {
        Loop_20ms(0.02f);
        T_cnt_20ms = 0;
    }
    if(T_cnt_50ms >= 50) {
        Loop_50ms(0.05f);
        T_cnt_50ms = 0;
    }
    if(T_cnt_100ms >= 100) {
        Loop_100ms(0.1f);
        T_cnt_100ms = 0;
    }
    if(T_cnt_200ms >= 200) {
        Loop_200ms(0.2f);
        T_cnt_200ms = 0;
    }
    if(T_cnt_500ms >= 500) {
        Loop_500ms(0.5f);
        T_cnt_500ms = 0;
    }
    if(T_cnt_1S >= 1000) {
        Loop_1S(1.0f);
        T_cnt_1S = 0;
    }
}
#include "Ctrl_DownTime.h"

/*
**********************************************************************
 * 函数原型：void Cheak_TimeDown(float dT)
 * 功      能：时间计时检测
 * 输      入: dT:执行周期
 * 参      数：float dT
**********************************************************************
```

```
*/
void Cheak_TimeDown(float dT)
{
    static float T,T1;
    if(Time.CountDown_Start)//启动倒计时
    {
        T += dT;
        if(T >= 1)//1S
        {
            if(Time.Ctrl_Time)
                Time.Ctrl_Time--;//控制时间--
            else
            {
                Time.CountDown_Start= 0;//倒计时结束
                Time.Ctrl_Time = Time.Set_Time;
                Beep_Flash = 5;//蜂鸣器响 5 下
            }
            T = 0;//周期清零
        }
    }
    else
    {
        T = 0;//周期清零
    }

    if(Time.Count_Start)//启动正计时
    {
        T1 += dT;
        if(T1 >= 1)//1S
        {
            Time.Count_Time++;//正计时++
            if(Time.Count_Time >= 5999)
            {
                Time.Count_Start = 0;//关闭正计时
                Time.TimeDisplay_Flag = 0;//切换到倒计时
                Time.Count_Time = 0;//正计时清零
                Beep_Flash = 5;//蜂鸣器响 5 下
            }
            T1 = 0;//周期清零
        }
    }
    else
    {
        T1 = 0;//周期清零
    }
}
#include "Ctrl_ControlTemp.h"

/**********全局变量声明******/
int Temp_Out,UC_Temp_Out;//加热制冷的 pwm 和但加热的 pwm
```

```
int8_t val;

/*
**********************************************************************
 * 函数原型：void Temp_Iint(void)
 * 功　　能：温度初始化
**********************************************************************
*/
void Temp_Iint(void)
{
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);//开启 tim1 通道 2 的 PWM
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);//开启 tim3 通道 2 的 PWM
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);//开启 tim3 通道 2 的 PWM
}

/*
**********************************************************************
 * 函数原型：void CoolTemp_Mode(float dT,int pwm)
 * 功　　能：加热制冷切换
**********************************************************************
*/
void CoolTemp_Mode(float dT,int pwm)
{
    static float T;
    static uint8_t Mode,mode;//临时存储当前模式

    if(pwm >= 0)
    {
        if(pwm > 120)
            WIND_OFF;
        mode = 0;//加热
    }
    else if(pwm < 0)
    {
        if(pwm < -600)
            WIND_ON;
        mode = 1;//制冷
    }

    if(Mode != mode)//如果模式变换了
    {
        T += dT;//开始计时
        if(T <= 1.0f)//一秒没把加热和制冷都拉低不工作
        {
            COLD = 0;
            HEAT = 0;
        }
        else//一秒后将改的模式赋值
        {
            T = 0;
```

```
                Mode = mode;
            }
        }
        else//没有改变模式的情况下
        {
            if(mode == 0)
            {
                val = 1;
                HEAT = pwm;
            }
            else if(mode == 1)
            {
                val = -1;
                if(Cool_Temp.Ctrl_Temp <= 220)
                    COLD = (-pwm)+450;
                else if(Cool_Temp.Ctrl_Temp <= 240)
                    COLD = (-pwm)+400;
                else if(Cool_Temp.Ctrl_Temp <= 250)
                    COLD = (-pwm)+365;
                else if(Cool_Temp.Ctrl_Temp <= 280)
                    COLD = (-pwm)+300;
                else if(Cool_Temp.Ctrl_Temp <= 300)
                    COLD = (-pwm)+200;
                else if(Cool_Temp.Ctrl_Temp <= 310)
                    COLD = (-pwm)+180;
                else if(Cool_Temp.Ctrl_Temp <= 320)
                    COLD = (-pwm)+170;
                else if(Cool_Temp.Ctrl_Temp <= 330)
                    COLD = (-pwm)+160;
                else
                    COLD = (-pwm)+20;
            }
        }
}

/*
*********************************************************************
 * 函数原型：void Temp_Control(float dT)
 * 功      能：温度加热控制
*********************************************************************
*/
void Temp_Control(float dT)
{
    if((sys.SetMode_Option == 0 || sys.SetMode_Option == 3) && sys.Calibration == 0)
    {

    PID_Temp(Get_ADCVal(Heat_Temp.Ctrl_Temp)/Calibration_Temp,ADC_Val1,&HeatTemp
_Arg,&HeatTemp_Val);//PID 计算
        UC_Temp_Out = (int)HeatTemp_Val.Out;//PID 值赋值
```

```
    PID_Temp(Get_ADCVal(Cool_Temp.Ctrl_Temp)*Calibration_Temp,ADC_Val2,&CoolTem
p_Arg,&CoolTemp_Val);//PID 计算
        Temp_Out = (int)CoolTemp_Val.Out;//PID 值赋值
    }
    else if(sys.Calibration == 1)
    {
        WIND_OFF;//关闭风扇
        Temp_Out = 0;
        if(sys.Calibration_Step == 0)
        {

    PID_Temp(Get_ADCVal(400),ADC_Val1,&HeatTemp_Arg,&HeatTemp_Val);//PID 计算
            UC_Temp_Out = (int)HeatTemp_Val.Out;//PID 值赋值
        }
        else
        {

    PID_Temp(Get_ADCVal(600),ADC_Val1,&HeatTemp_Arg,&HeatTemp_Val);//PID 计算
            UC_Temp_Out = (int)HeatTemp_Val.Out;//PID 值赋值
        }
    }
    else
    {
        WIND_OFF;//关闭风扇
        UC_Temp_Out = 0;
        Temp_Out = 0;
    }

    CoolTemp_Mode(dT, Temp_Out);
    if(UC_Temp_Out < 0)
        UC_Temp_Out = 0;
    UC_HEAT = UC_Temp_Out;
}
#include "System_Init.h"

/*
*********************************************************************
 * 函数原型：void System_Init(void)
 * 功      能：系统功能初始化
*********************************************************************
*/
void System_Init(void)
{
    /**********系统初始化开始**********/
    sys.Init_ok = 0;

    /**********参数初始化**************/
    Param_Read();

    /*********ADC&DMA 初始化********/
```

```
    ADCDMA_Init();

    /**********GN1640 初始化***********/
    GN1640_Init();

    /**********温度初始化**************/
    Temp_Iint();

    /**********PID 初始化**********/
    PID_Init();

    /**********蜂鸣器响一下*************/
    Beep_Time = 0.2;//蜂鸣器响 0.1S

    /**********系统初始化成功**********/
    sys.Init_ok = 1;
}
```