HS2200_5C 软件源程序

```
#include "Drv_HT162x.h"

/*
*********************************************************************
 * 函数原型：static void LCD_Delay(void)
 * 功      能：LCD_us 延时
 * 调      用：内部调用
*********************************************************************
*/
static void LCD_Delay(void)
{
    unsigned char a;
    for(a = 100; a > 0; a--);
}

/*
*********************************************************************
 * 函数原型：static void Write_Mode(unsigned char MODE)
 * 功      能：写入模式,数据 or 命令
 * 输      入: MODE ： 数据 or 命令
 * 参      数：unsigned char MODE
 * 调      用：内部调用
*********************************************************************
*/
static void Write_Mode(unsigned char MODE)
{
    LCD_Delay();
    Clr_162x_Wr;//RW = 0;
    LCD_Delay();
    Set_162x_Dat;//DA = 1;
    Set_162x_Wr;//RW = 1;
    LCD_Delay();
    Clr_162x_Wr;//RW = 0;
    LCD_Delay();
    Clr_162x_Dat;//DA = 0;
    LCD_Delay();
    Set_162x_Wr;//RW = 1;
    LCD_Delay();
    Clr_162x_Wr;//RW = 0;
    LCD_Delay();
    if (0 == MODE)
    {
        Clr_162x_Dat;//DA = 0;
    }
    else
    {
        Set_162x_Dat;//DA = 1;
    }
    LCD_Delay();
    Set_162x_Wr;//RW = 1;
```

```
    LCD_Delay();
}

/*
*********************************************************************
 * 函数原型：static void Write_Command(unsigned char Cbyte)
 * 功    能：LCD 命令写入函数
 * 输    入: Cbyte：控制命令字
 * 参    数：unsigned char Cbyte
 * 调    用：内部调用
*********************************************************************
*/
static void Write_Command(unsigned char Cbyte)
{
    unsigned char i = 0;
    for (i = 0; i < 8; i++)
    {
        Clr_162x_Wr;
        if ((Cbyte >> (7 - i)) & 0x01)
        {
            Set_162x_Dat;
        }
        else
        {
            Clr_162x_Dat;
        }
        LCD_Delay();
        Set_162x_Wr;
        LCD_Delay();
    }
    Clr_162x_Wr;
    LCD_Delay();
    Clr_162x_Dat;
    Set_162x_Wr;
    LCD_Delay();
}

/*
*********************************************************************
 * 函数原型：static void Write_Address(unsigned char Abyte)
 * 功    能：LCD 地址写入函数
 * 输    入: Abyte：地址
 * 参    数：unsigned char Abyte
 * 调    用：内部调用
*********************************************************************
*/
static void Write_Address(unsigned char Abyte)
{
    unsigned char i = 0;
    Abyte = Abyte << 1;
```

```
    for (i = 0; i < 6; i++)
    {
        Clr_162x_Wr;
        if ((Abyte >> (6 - i)) & 0x01)
        {
            Set_162x_Dat;
        }
        else
        {
            Clr_162x_Dat;
        }
        LCD_Delay();
        Set_162x_Wr;
        LCD_Delay();
    }
}

/*
*********************************************************************
 * 函数原型：static void Write_Data_8bit(unsigned char Dbyte)
 * 功      能：LCD 8bit 数据写入函数
 * 输      入: Dbyte：数据
 * 参      数：unsigned char Dbyte
 * 调      用：内部调用
*********************************************************************
*/
static void Write_Data_8bit(unsigned char Dbyte)
{
    int i = 0;
    for (i = 0; i < 8; i++)
    {
        Clr_162x_Wr;
        if ((Dbyte >> (7 - i)) & 0x01)
        {
            Set_162x_Dat;
        }
        else
        {
            Clr_162x_Dat;
        }
        LCD_Delay();
        Set_162x_Wr;
        LCD_Delay();
    }
}

/*
*********************************************************************
 * 函数原型：void Write_Data_4bit(unsigned char Dbyte)
 * 功      能：LCD 4bit 数据写入函数
```

```
 * 输      入: Dbyte：数据
 * 参      数：unsigned char Dbyte
 * 调      用：内部调用
**********************************************************************
*/
void Write_Data_4bit(unsigned char Dbyte)
{
    int i = 0;
    for (i = 0; i < 4; i++)
    {
        Clr_162x_Wr;
        if ((Dbyte >> (3 - i)) & 0x01)
        {
            Set_162x_Dat;
        }
        else
        {
            Clr_162x_Dat;
        }
        LCD_Delay();
        Set_162x_Wr;
        LCD_Delay();
    }
}

/*
**********************************************************************
 * 函数原型：void Lcd_Init(void)
 * 功      能：LCD 初始化，对 lcd 自身做初始化设置
**********************************************************************
*/
void Lcd_Init(void)
{
    Set_162x_Cs;
    Set_162x_Wr;
    Set_162x_Dat;
    LCD_Delay();
    Clr_162x_Cs;//CS = 0;
    LCD_Delay();
    Write_Mode(0);//命令模式
    Write_Command(0x01);//Enable System
    Write_Command(0x03);//Enable Bias
    Write_Command(0x04);//Disable Timer
    Write_Command(0x05);//Disable WDT
    Write_Command(0x08);//Tone OFF
    Write_Command(0x18);//on-chip RC 震荡
    Write_Command(0x29);//1/4Duty 1/3Bias
    Write_Command(0x80);//Disable IRQ
    Write_Command(0x40);//Tone Frequency 4kHZ
    Write_Command(0xE3);//Normal Mode
```

```
    Set_162x_Cs;//CS = 1;

    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 180);//背光 pwm

    Lcd_All();
    HAL_Delay(1000);
    Lcd_Clr();
}

/*
**********************************************************************
 * 函数原型：void Lcd_Clr(void)
 * 功      能：LCD 清屏函数
**********************************************************************
*/
void Lcd_Clr(void)
{
    Write_Addr_Dat_N(0x0, 0x00, 60);
}

/*
**********************************************************************
 * 函数原型：void Lcd_All(void)
 * 功      能：LCD 全显示函数
**********************************************************************
*/
void Lcd_All(void)
{
    Write_Addr_Dat_N(0x0, 0xFF, 60);
}

/*
**********************************************************************
 * 函数原型：void Write_Addr_Dat_N(unsigned char _addr, unsigned char _dat, unsigned char
n)
 * 功      能：屏幕显示
 * 输      入: _addr：地址   char _dat：数据  n：个数
 * 参      数：unsigned char _addr, unsigned char _dat, unsigned char n
**********************************************************************
*/
void Write_Addr_Dat_N(unsigned char _addr, unsigned char _dat, unsigned char n)
{
    unsigned char i = 0;
    Clr_162x_Cs;//CS = 0;
    LCD_Delay();
    Write_Mode(1);
    Write_Address(_addr);
    for (i = 0; i < n; i++)
    {
```

```
            Write_Data_8bit(_dat);
        }
        Set_162x_Cs;//CS = 1;
}
#include "Drv_KEY.h"

#if (Key_Type == 0)

/**********全局变量声明******/
float Key_Status;//按键按下标志

/**********局部变量声明******/
float Key_Cnt1,Key_Cnt2,Key_Cnt3,Key_Cnt4;//按下时间
uint8_t Key_Flag1,Key_Flag2,Key_Flag3,Key_Flag4;//按键按下标志
uint8_t LongPress1,LongPress2,LongPress3,LongPress4;//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)
{
    /*************************************MENU                                键
*****************************************/
    if(KEY1 == KEY_DOWN)//按下按键
    {
        if(sys.Run_Status)
            return;
        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 == KEY_UP)//抬起按键
        {
            if(Key_Cnt1 > 0.1 && Key_Cnt1 < 1.5)//小于 1.5S 是单击
            {
                sys.SetMode_Option++;
                if(sys.SetMode_Option > 3)
                    sys.SetMode_Option = 0;
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 6;//闪烁时间 6S
            }
            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)//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
            {
                LongPress1 = 1;//长按标志置一
            }
        }
    }
    /************************************ 加 键
************************************/
    if(KEY2== KEY_DOWN)//按下按键
    {
        if(sys.Run_Status)
            return;
        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 == KEY_UP)//抬起按键
        {
            if(Key_Cnt2 < 1.4)//小于 1.5S 是单击
            {
                if(sys.SetMode_Option == 1)//设置温度
                {
                    Temp.Set += 10;//温度加 1 度
                    if(Temp.Set > Temp_MAX)//假如温度大于 Temp_MAX 度时
                        Temp.Set = Temp_MAX;//温度等于 Temp_MAX 度
                }
                else if(sys.SetMode_Option == 2)//设置速度
                {
                    Speed.Set += 10;//转速加 10 转
                    if(Speed.Set == 10)//从零转开始最低为 50 转，判断是 10 后
                        Speed.Set = 50;//设定转速为 50 开始
                    if(Speed.Set > Speed_MAX)//假如转速大于 Speed_MAX
                        Speed.Set = Speed_MAX;//转速等于 Speed_MAX
```

```
                }
                else if(sys.SetMode_Option == 3)//设置时间
                {
                    Time.Set += 60;//时间加 1 分钟
                    if(Time.Set > Time_MAX)//假如时间大于 Time_MAX 时
                        Time.Set = Time_MAX;//时间等于 Time_MAX
                }
                Key_Status = 2;//设置时 2S 不闪烁
                Twinkle_Time = 6;//闪烁时间 6S
            }
            Key_Flag2 = 0;//按键事件结束，等待下一次按下
            Key_Cnt2 = 0;//按钮计数清零
        }
        if(Key_Cnt2 > 1.9 && Key_Cnt2 < 2.1)//按键时间大于 1.9S 小于 2.1S 表示长按
        {
            if(sys.SetMode_Option == 1)//设置温度
            {
                Temp.Set += 100;//温度加 10 度
                if(Temp.Set > Temp_MAX)//假如温度大于 Temp_MAX 度时
                    Temp.Set = Temp_MAX;//温度等于 Temp_MAX 度
            }
            else if(sys.SetMode_Option == 2)//设置速度
            {
                Speed.Set += 100;//转速加 100 转
                if(Speed.Set > Speed_MAX)//假如转速大于 Speed_MAX
                    Speed.Set = Speed_MAX;//转速等于 Speed_MAX
            }
            else if(sys.SetMode_Option == 3)//设置时间
            {
                Time.Set += 600;//时间加 10 分钟
                if(Time.Set > Time_MAX)//假如时间大于 Time_MAX 时
                    Time.Set = Time_MAX;//时间等于 Time_MAX
            }
            Key_Status = 2;//设置时 2S 不闪烁
            Twinkle_Time = 6;//闪烁时间 6S
            Key_Flag2 = 0;//按键事件结束，等待下一次按下
            Key_Cnt2 = 1.4;//按钮计数从 1.4s 开始
        }
    }

    /************************************ 减 键
************************************/
    if(KEY3 == KEY_DOWN)//按下按键
    {
        if(sys.Run_Status)
            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 == KEY_UP)//抬起按键
    {
        if(Key_Cnt3 < 1.4)/*单击*///小于 1.5S 是单击
        {
            if(sys.SetMode_Option == 1)//设置温度
            {
                Temp.Set -= 10;//温度减 1 度
                if(Temp.Set < 0)//假如温度小于 0 度时
                    Temp.Set = 0;//温度等于 0 度
            }
            else if(sys.SetMode_Option == 2)//设置速度
            {
                Speed.Set -= 10;//转速减 10 转
                if(Speed.Set < 50)//假如转速小于 50 时
                    Speed.Set = 0;//转速等于 0
            }
            else if(sys.SetMode_Option == 3)//设置时间
            {
                Time.Set -= 60;//时间减 1 分钟
                if(Time.Set < 0)//假如时间小于 0 时
                    Time.Set = 0;//时间等于 0
            }
            Key_Status = 2;//设置时 2S 不闪烁
            Twinkle_Time = 6;//闪烁时间 6S
        }
        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
    }
    if(Key_Cnt3 > 1.9 && Key_Cnt3 < 2.1)//按键时间大于 1.9S 小于 2.1S 表示长按
    {
        if(sys.SetMode_Option == 1)//设置温度
        {
            Temp.Set -= 100;//温度减 10 度
            if(Temp.Set < 0)//假如温度小于 0 度时
                Temp.Set = 0;//温度等于 0 度
        }
        else if(sys.SetMode_Option == 2)//设置速度
        {
            Speed.Set -= 100;//转速减 100 转
            if(Speed.Set < 50)//假如转速小于 50 时
                Speed.Set = 0;//转速等于 0
        }
        else if(sys.SetMode_Option == 3)//设置时间
        {
            Time.Set -= 600;//时间减 10 分钟
            if(Time.Set < 0)//假如时间小于 0 时
                Time.Set = 0;//时间等于 0
        }
        Key_Status = 2;//设置时 2S 不闪烁
```

```
            Twinkle_Time = 6;//闪烁时间 6S
            Key_Flag3 = 0;//按键事件结束，等待下一次按下
            Key_Cnt3 = 1.4;//按钮计数从 1.5s 开始
        }
    }

    /*************************************Start 键
*****************************************/
    if(KEY4== KEY_DOWN)//按下按键
    {
        if(LongPress4 == 0)//没有长按过
        {
            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 == KEY_UP)//抬起按键
        {
            if(Key_Cnt4 > 0.1 && Key_Cnt4 < 1.5)//小于 1.5S 是单击
            {
                if(sys.Run_Status == 0 && (Speed.Set || Temp.Set))//系统没启动的话
                {
                    sys.Run_Status = 1;//启动系统
                    Speed_Val.Integral = 43;//电器起步
                    sys.SetMode_Option = 0;//设定模式设置为 0
                    Temp_Val.Integral = 0;//加热的积分清零
                }
                else//系统启动的话
                {
                    Speed.ADDMode = 1;//进入减速模式
                    Speed.Ctrl = 0;//将控制速度设置为 0
                }
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 0;//闪烁时间 6S
                sys.SetMode_Option = 0;
            }
            Key_Flag4 = 0;//按键事件结束，等待下一次按下
            LongPress4 = 0;//长按标志清零
            Key_Cnt4 = 0;//按钮计数清零
        }
        if(Key_Cnt4 > 1.5 && Key_Cnt4 < 3)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress4 == 0)//如果没有一直一直长按着
            {
                LongPress4 = 1;//长按标志置一
            }
        }
    }
```

```
}
#endif
#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 < 0.2)//如果小于 0.2s 时
        {
            Beep_ON;//蜂鸣器响
        }
        else if(BT >= 0.2 && BT < 0.3)//在 0.2 和 0.3s 之间时
        {
            Beep_OFF;//关闭蜂鸣器
        }
        else if(BT >= 0.3)//大于等于 0.2s 时
        {
            Beep_Flash--;//次数--
            BT = 0;//周期清零
        }
    }
}
#include "Drv_EC11A.h"
#if (Key_Type == 1)
/**********结构体**********/
```

```
_EC11A_ EC11A[2];//旋钮参数

/**********全局变量声明******/
float Key_Status;//按键按下标志

/*
*********************************************************************
 * 函数原型：void EC11A_Init(void)
 * 功　　能：EC11A 初始化定时器
*********************************************************************
*/
void EC11A_Init(void)
{
    /**********EC11A_1**********/
    EC11A[0].EXTI_Pin = KEY1A_Pin;//EC11A 旋钮中断引脚
    EC11A[0].EC11A_Pin = KEY1B_Pin;//EC11A 旋钮输入引脚
    EC11A[0].EC11A_GPIO = KEY1B_GPIO_Port;//EC11A 旋钮输入 GPIO 端口

    EC11A[0].Key_Pin = KEY1_Pin;//EC11A 按键输入引脚
    EC11A[0].Key_GPIO = KEY1_GPIO_Port;//EC11A 按键输入 GPIO 端口

    EC11A[0].Tim = &EC11A_Tim_1;//定时器选择
    EC11A[0].EC11A_Fast = EC11A_FastSpeed;//判断旋转速度阈值

    /**********EC11A_2**********/
    EC11A[1].EXTI_Pin = KEY2A_Pin;//EC11A 旋钮中断引脚
    EC11A[1].EC11A_Pin = KEY2B_Pin;//EC11A 旋钮输入引脚
    EC11A[1].EC11A_GPIO = KEY2B_GPIO_Port;//EC11A 旋钮输入 GPIO 端口

    EC11A[1].Key_Pin = KEY2_Pin;//EC11A 按键输入引脚
    EC11A[1].Key_GPIO = KEY2_GPIO_Port;//EC11A 按键输入 GPIO 端口

    EC11A[1].Tim = &EC11A_Tim_2;//定时器选择
    EC11A[1].EC11A_Fast = EC11A_FastSpeed;//判断旋转速度阈值
}

/*
*********************************************************************
 * 函数原型：void EC11A_Speed(float dT)
 * 功　　能：EC11A 旋钮速度计算
*********************************************************************
*/
void EC11A_Speed(float dT)
{
    /**********EC11A_1**********/
    EC11A[0].EC11A_Speed = EC11A[0].EC11A_Cnt*60/20;//一秒检测一次。转一圈 20 个反
馈，一分钟的速度
    EC11A[0].EC11A_Cnt = 0;//将检测到的计数清零

    /**********EC11A_2**********/
```

```
    EC11A[1].EC11A_Speed = EC11A[1].EC11A_Cnt*60/20;//一秒检测一次。转一圈 20 个反馈，一分钟的速度
    EC11A[1].EC11A_Cnt = 0;//将检测到的计数清零
}

/*
**********************************************************************
 * 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 EC11AKey_Scan(float dT)
 * 功      能：EC11A 按键扫描
**********************************************************************
*/
void EC11AKey_Scan(float dT)
{
    /**********EC11A_1**********/
    if(HAL_GPIO_ReadPin(EC11A[0].Key_GPIO,EC11A[0].Key_Pin) == KEY_DOWN)//按下按键
    {
        if(EC11A[0].LongPress == 0)//没有长按过
        {
            EC11A[0].Key_Cnt += dT;//按下时间++
            EC11A[0].Key_Flag = 1;//按键按下标志置一
        }
    }
    if(EC11A[0].Key_Flag == 1)//按键被按下
    {
        if(HAL_GPIO_ReadPin(EC11A[0].Key_GPIO,EC11A[0].Key_Pin)  ==  KEY_UP)//抬起按键
        {
            if(EC11A[0].Key_Cnt > 0.1 && EC11A[0].Key_Cnt < 1.5)//小于 1.5S 是单击
            {
                if(sys.Run_Status == 0 && (Speed.Set || Temp.Set))//系统没启动的话
                {
                    sys.Run_Status = 1;//启动系统
                    Speed_Val.Integral = 43;//电器起步
                    sys.SetMode_Option = 0;//设定模式设置为 0
                    Temp_Val.Integral = 0;//加热的积分清零
                    Temp.Old = Temp.Rel;
                }
```

```
                else//系统启动的话
                {
                    Speed.ADDMode = 1;//进入减速模式
                    Speed.Ctrl = 0;//将控制速度设置为 0
                }
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 0;//闪烁时间 6S
                sys.SetMode_Option = 0;
            }
            EC11A[0].Key_Flag = 0;//按键事件结束，等待下一次按下
            EC11A[0].LongPress = 0;//长按标志清零
            EC11A[0].Key_Cnt = 0;//按钮计数清零
        }
        if(EC11A[0].Key_Cnt > 1.5 && EC11A[0].Key_Cnt < 3)//按键时间大于 1.5S 小于 3S
表示长按
        {
            if(EC11A[0].LongPress == 0)//如果没有一直一直长按着
            {

                EC11A[0].LongPress = 1;//长按标志置一
            }
        }
    }

    /**********EC11A_2**********/
    if(HAL_GPIO_ReadPin(EC11A[1].Key_GPIO,EC11A[1].Key_Pin) == KEY_DOWN)//按下
按键
    {
        if(EC11A[1].LongPress == 0)//没有长按过
        {
            EC11A[1].Key_Cnt += dT;//按下时间++
            EC11A[1].Key_Flag = 1;//按键按下标志置一
        }
    }
    if(EC11A[1].Key_Flag == 1)//按键被按下
    {
        if(HAL_GPIO_ReadPin(EC11A[1].Key_GPIO,EC11A[1].Key_Pin)  ==  KEY_UP)//抬
起按键
        {
            if(EC11A[1].Key_Cnt > 0.1 && EC11A[1].Key_Cnt < 1.5)//小于 1.5S 是单击
            {
                if(sys.Run_Status == 0 && (Speed.Set || Temp.Set))//系统没启动的话
                {
                    sys.Run_Status = 1;//启动系统
                    Speed_Val.Integral = 43;//电器起步
                    sys.SetMode_Option = 0;//设定模式设置为 0
                    Temp_Val.Integral = 0;//加热的积分清零
                    Temp.Old = Temp.Rel;
                }
                else//系统启动的话
```

```
                {
                    Speed.ADDMode = 1;//进入减速模式
                    Speed.Ctrl = 0;//将控制速度设置为 0
                }
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 0;//闪烁时间 6S
                sys.SetMode_Option = 0;
            }
            EC11A[1].Key_Flag = 0;//按键事件结束，等待下一次按下
            EC11A[1].LongPress = 0;//长按标志清零
            EC11A[1].Key_Cnt = 0;//按钮计数清零
        }
        if(EC11A[1].Key_Cnt > 1.5 && EC11A[1].Key_Cnt < 3)//按键时间大于 1.5S 小于 3S
表示长按
        {
            if(EC11A[1].LongPress == 0)//如果没有一直一直长按着
            {

                EC11A[1].LongPress = 1;//长按标志置一
            }
        }
    }
}

/*
**********************************************************************
 * 函数原型：void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
 * 功    能：外部中断
**********************************************************************
*/
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    UNUSED(GPIO_Pin);

    /**********EC11A_1**********/
    if(GPIO_Pin == EC11A[0].EXTI_Pin)//A 上升沿触发外部中断
    {
        HAL_TIM_Base_Start_IT(EC11A[0].Tim);//开始定时器
        while(EC11A[0].TIM_Cnt <= 2)//定时器一个周期 1ms，计时 2ms 内看看 A 有没有电
跳变
        {
            if(GPIO_Pin == EC11A[0].EXTI_Pin)//在 2ms 内，检测到电平变化
            {
                HAL_TIM_Base_Stop_IT(EC11A[0].Tim);//停止定时器
                EC11A[0].TIM_Cnt = 0;//清除 TIM 计数
                EC11A[0].EC11A_Cnt++;//旋钮计数
                EC11A[0].EC11A_Knob = 2;//在旋转旋钮时
                if(sys.Run_Status)
                    return;
                sys.SetMode_Option = 1;//设置温度
```

```
                    if(HAL_GPIO_ReadPin(EC11A[0].EC11A_GPIO,EC11A[0].EC11A_Pin) ==
0)//加
                    {
                        if(sys.SetMode_Option == 1)
                        {
                            if(EC11A[0].EC11A_Speed < EC11A[0].EC11A_Fast)//如果慢慢
旋转
                                Temp.Set += 10;
                            else
                                Temp.Set += 30;
                            if(Temp.Set > Temp_MAX)
                                Temp.Set = Temp_MAX;
                            Key_Status = 1;//设置时 2S 不闪烁
                            Twinkle_Time = 2;//闪烁时间 6S
                        }
                        break;
                    }
                    else if(HAL_GPIO_ReadPin(EC11A[0].EC11A_GPIO,EC11A[0].EC11A_Pin)
== 1)//减
                    {
                        if(sys.SetMode_Option == 1)
                        {
                            if(EC11A[0].EC11A_Speed < EC11A[0].EC11A_Fast)//如果慢慢
旋转
                                Temp.Set -= 10;
                            else
                                Temp.Set -= 30;
                            if(Temp.Set <= 0)
                                Temp.Set = 0;
                            Key_Status = 1;//设置时 2S 不闪烁
                            Twinkle_Time = 2;//闪烁时间 2S
                        }
                        break;
                    }
                    break;
                }
            }
            HAL_TIM_Base_Stop_IT(EC11A[0].Tim);//停止定时器
            EC11A[0].TIM_Cnt = 0;//清除 TIM 计数
        }

        /**********EC11A_2**********/
        if(GPIO_Pin == EC11A[1].EXTI_Pin)//A 上升沿触发外部中断
        {
            HAL_TIM_Base_Start_IT(EC11A[1].Tim);//开始定时器
            while(EC11A[1].TIM_Cnt <= 2)//定时器一个周期 1ms，计时 2ms 内看看 A 有没有电
跳变
            {
                if(GPIO_Pin == EC11A[1].EXTI_Pin)//在 2ms 内，检测到电平变化
                {
```

```
                HAL_TIM_Base_Stop_IT(EC11A[1].Tim);//停止定时器
                EC11A[1].TIM_Cnt = 0;//清除 TIM 计数
                EC11A[1].EC11A_Cnt++;//旋钮计数
                EC11A[1].EC11A_Knob = 2;//在旋转旋钮时
                if(sys.Run_Status)
                    return;
                sys.SetMode_Option = 2;//设置速度
                if(HAL_GPIO_ReadPin(EC11A[1].EC11A_GPIO,EC11A[1].EC11A_Pin) ==
0)//加
                {
                    /*加*/
                    if(sys.SetMode_Option == 2)
                    {
                        if(EC11A[1].EC11A_Speed < EC11A[1].EC11A_Fast)//如果慢慢
旋转
                        {
                            Speed.Set += 10;
                            if(Speed.Set == 10)//从零转开始最低为 50 转，判断是 10 后
                                Speed.Set = 200;//设定转速为 200 开始
                        }
                        else
                        {
                            Speed.Set += 30;
                            if(Speed.Set == 30)//从零转开始最低为 50 转，判断是 10 后
                                Speed.Set = 200;//设定转速为 200 开始
                        }
                        if(Speed.Set > Speed_MAX)
                            Speed.Set = Speed_MAX;
                        Key_Status = 1;//设置时 2S 不闪烁
                        Twinkle_Time = 2;//闪烁时间 6S
                    }
                    break;
                }
                else if(HAL_GPIO_ReadPin(EC11A[1].EC11A_GPIO,EC11A[1].EC11A_Pin)
== 1)//减
                {
                    /*减*/
                    if(sys.SetMode_Option == 2)
                    {
                        if(EC11A[1].EC11A_Speed < EC11A[1].EC11A_Fast)//如果慢慢
旋转
                        {
                            Speed.Set -= 10;
                        }
                        else
                        {
                            Speed.Set -= 30;
                        }
                        if(Speed.Set < 200)
                            Speed.Set = 0;
```

```
                                Key_Status = 1;//设置时 2S 不闪烁
                                Twinkle_Time = 2;//闪烁时间 6S
                            }
                            break;
                        }
                        break;
                    }
                }
                HAL_TIM_Base_Stop_IT(EC11A[1].Tim);//停止定时器
                EC11A[1].TIM_Cnt = 0;//清除 TIM 计数
            }
        }

/*
**********************************************************************
 * 函数原型：void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
 * 功      能：定时器计数中断
**********************************************************************
*/
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance == EC11A_Tim_1.Instance)
    {
        EC11A[0].TIM_Cnt++;
    }

    if(htim->Instance == EC11A_Tim_2.Instance)
    {
        EC11A[1].TIM_Cnt++;
    }
}
#endif

#include "Drv_PT1000.h"
/*
-22.1-300 摄氏度
*/
const float Temp_map[1521]=
{
/*-22*/913.733,  913.340,  912.946,  912.553,  912.159,  911.766,  911.372,
    910.979,  910.585,  910.192,
/*-21*/917.666,  917.273,  916.879,  916.486,  916.093,  915.700,
    915.306,  914.913,  914.520,  914.126,
/*-20*/921.599,  921.206,  920.812,  920.419,  920.026,  919.633,
    919.239,  918.846,  918.453,  918.059,
/*-19*/925.531,  925.138,  924.745,  924.351,  923.958,  923.565,
    923.172,  922.779,  922.385,  921.992,
/*-18*/929.460,  929.067,  928.674,  928.281,  927.888,  927.496,
    927.103,  926.710,  926.317,  925.924,
/*-17*/933.390,  932.997,  932.604,  932.211,  931.818,  931.425,  931.032,
```

```
        930.639,        930.246,        929.853,
/*-16*/937.317,        936.924,        936.532,        936.139,        935.746,        935.354,
        934.961,        934.568,        934.175,        933.783,
/*-15*/941.244,        940.851,        940.459,        940.066,        939.673,        939.281,
        938.888,        938.495,        938.102,        937.710,
/*-14*/945.170,        944.777,        944.385,        943.992,        943.600,        943.207,
        942.814,        942.422,        942.029,        941.637,
/*-13*/949.094,        948.702,        948.309,        947.917,        947.524,        947.132,
        946.740,        946.347,        945.955,        945.562,
/*-12*/953.016,        952.624,        952.232,        951.839,        951.447,        951.055,  950.663,
        950.271,  949.878,  949.486,
/*-11*/956.938,        956.546,        956.154,        955.761,        955.369,        954.977,  954.585,
        954.193,  953.800,  953.408,
/*-10*/960.859,        960.467,        960.075,        959.683,        959.291,        958.899,  958.506,
        958.114,  957.722,  957.330,
/*-9*/964.779,         964.387,        963.995,        963.603,        963.211,  962.819,  962.427,
        962.035,  961.643,  961.251,
/*-8*/968.697,         968.305,        967.913,        967.522,        967.130,        966.738,  966.346,
        965.954,  965.563,  965.171,
/*-7*/972.614,         972.222,        971.831,        971.439,        971.047,        970.656,  970.264,
        969.872,  969.480,  969.089,
/*-6*/976.529,         976.138,        975.746,        975.355,        974.963,        974.572,  974.180,
        973.789,  973.397,  973.006,
/*-5*/980.444,         980.053,        979.662,        979.270,        978.879,        978.487,  978.096,
        977.704,  977.313,  976.921,
/*-4*/984.358,         983.967,        983.575,        983.184,        982.793,        982.401,  982.010,
        981.618,  981.227,  980.835,
/*-3*/988.270,         987.879,        987.488,        987.096,        986.705,        986.314,  985.923,
        985.532,  985.140,  984.749,
/*-2*/992.181,991.790,  991.399,  991.008,  990.617,  990.226,  989.834,  989.443,  989.052,
        988.661,
/*-1*/996.091,995.700,  995.309,  994.918,  994.527,  994.136,        993.745,        993.354,
        992.963,        992.572,
/*0*/1000.000,         1000.391,       1000.782,       1001.172,       1001.563,       1001.954,
        1002.345,       1002.736,       1003.126,       1003.517,
/*1*/1003.908,         1004.298,       1004.689,       1005.080,       1005.470,       1005.861,
        1006.252,       1006.642,       1007.033,       1007.424,
/*2*/1007.814,         1008.205,       1008.595,       1008.986,       1009.377,       1009.767,
        1010.158,       1010.548,       1010.939,       1011.329,
/*3*/1011.720,         1012.110,       1012.501,       1012.891,       1013.282,       1013.672,
        1014.062,       1014.453,       1014.843,       1015.234,
1015.624,       1016.014,       1016.405,       1016.795,       1017.185,       1017.576,
        1017.966,       1018.356,       1018.747,       1019.137,
1019.527,       1019.917,       1020.308,       1020.698,       1021.088,       1021.478,
        1021.868,       1022.259,       1022.649,       1023.039,
1023.429,       1023.819,       1024.209,       1024.599,       1024.989,       1025.380,
        1025.770,       1026.160,       1026.550,       1026.940,
1027.330,       1027.720,       1028.110,       1028.500,       1028.890,       1029.280,
        1029.670,       1030.060,       1030.450,       1030.840,
1031.229,       1031.619,       1032.009,       1032.399,       1032.789,       1033.179,
```

1033.569, 1033.958, 1034.348, 1034.738,
1035.128, 1035.518, 1035.907, 1036.297, 1036.687, 1037.077,
1037.466, 1037.856, 1038.246, 1038.636,
1039.025, 1039.415, 1039.805, 1040.194, 1040.584, 1040.973,
1041.363, 1041.753, 1042.142, 1042.532,
1042.921, 1043.311, 1043.701, 1044.090, 1044.480, 1044.869,
1045.259, 1045.648, 1046.038, 1046.427,
1046.816, 1047.206, 1047.595, 1047.985, 1048.374, 1048.764,
1049.153, 1049.542, 1049.932, 1050.321,
1050.710, 1051.099, 1051.489, 1051.878, 1052.268, 1052.657,
1053.046, 1053.435, 1053.825, 1054.214,
1054.603, 1054.992, 1055.381, 1055.771, 1056.160, 1056.549,
1056.938, 1057.327, 1057.716, 1058.105,
1058.495, 1058.884, 1059.273, 1059.662, 1060.051, 1060.440,
1060.829, 1061.218, 1061.607, 1061.996,
1062.385, 1062.774, 1063.163, 1063.552, 1063.941, 1064.330,
1064.719, 1065.108, 1065.496, 1065.885,
1066.274, 1066.663, 1067.052, 1067.441, 1067.830, 1068.218,
1068.607, 1068.996, 1069.385, 1069.774,
1070.162, 1070.551, 1070.940, 1071.328, 1071.717, 1072.106,
1072.495, 1072.883, 1073.272, 1073.661,
1074.049, 1074.438, 1074.826, 1075.215, 1075.604, 1075.992,
1076.381, 1076.769, 1077.158, 1077.546,
1077.935, 1078.324, 1078.712, 1079.101, 1079.489, 1079.877,
1080.266, 1080.654, 1081.043, 1081.431,
1081.820, 1082.208, 1082.596, 1082.985, 1083.373, 1083.762,
1084.150, 1084.538, 1084.926, 1085.315,
1085.703, 1086.091, 1086.480, 1086.868, 1087.256, 1087.644,
1088.033, 1088.421, 1088.809, 1089.197,
1089.585, 1089.974, 1090.362, 1090.750, 1091.138, 1091.526,
1091.914, 1092.302, 1092.690, 1093.078,
1093.467, 1093.855, 1094.243, 1094.631, 1095.019, 1095.407,
1095.795, 1096.183, 1096.571, 1096.959,
1097.347, 1097.734, 1098.122, 1098.510, 1098.898, 1099.286,
1099.674, 1100.062, 1100.450, 1100.838,
1101.225, 1101.613, 1102.001, 1102.389, 1102.777, 1103.164,
1103.552, 1103.940, 1104.328, 1104.715,
1105.103, 1105.491, 1105.879, 1106.266, 1106.654, 1107.042,
1107.429, 1107.817, 1108.204, 1108.592,
1108.980, 1109.367, 1109.755, 1110.142, 1110.530, 1110.917,
1111.305, 1111.693, 1112.080, 1112.468,
1112.855, 1113.242, 1113.630, 1114.017, 1114.405, 1114.792,
1115.180, 1115.567, 1115.954, 1116.342,
1116.729, 1117.117, 1117.504, 1117.891, 1118.279, 1118.666,
1119.053, 1119.441, 1119.828, 1120.215,
1120.602, 1120.990, 1121.377, 1121.764, 1122.151, 1122.538,
1122.926, 1123.313, 1123.700, 1124.087,
1124.474, 1124.861, 1125.248, 1125.636, 1126.023, 1126.410,
1126.797, 1127.184, 1127.571, 1127.958,
1128.345, 1128.732, 1129.119, 1130.127, 1129.893, 1130.280,

1130.667, 1131.054, 1131.441, 1131.828,
1132.215, 1132.602, 1132.988, 1133.375, 1133.762, 1134.149,
1134.536, 1134.923, 1135.309, 1135.696,
1136.083, 1136.470, 1136.857, 1137.243, 1137.630, 1138.017,
1138.404, 1138.790, 1139.177, 1139.564,
1139.950, 1140.337, 1140.724, 1141.110, 1141.497, 1141.884,
1142.270, 1142.657, 1143.043, 1143.430,
1143.817, 1144.203, 1144.590, 1144.976, 1145.363, 1145.749,
1146.136, 1146.522, 1146.909, 1147.295,
1147.681, 1148.068, 1148.454, 1148.841, 1149.227, 1149.614,
1150.000, 1150.386, 1150.773, 1151.159,
1151.545, 1151.932, 1152.318, 1152.704, 1153.091, 1153.477,
1153.863, 1154.249, 1154.636, 1155.022,
1155.408, 1155.794, 1156.180, 1156.567, 1156.953, 1157.339,
1157.725, 1158.111, 1158.497, 1158.883,
1159.270, 1159.656, 1160.042, 1160.428, 1160.814, 1161.200,
1161.586, 1161.972, 1162.358, 1162.744,
1163.130, 1163.516, 1163.902, 1164.288, 1164.674, 1165.060,
1165.446, 1165.831, 1166.217, 1166.603,
1166.989, 1167.375, 1167.761, 1168.147, 1168.532, 1168.918,
1169.304, 1169.690, 1170.076, 1170.461,
1170.847, 1171.233, 1171.619, 1172.004, 1172.390, 1172.776,
1173.161, 1173.547, 1173.933, 1174.318,
1174.704, 1175.090, 1175.475, 1175.861, 1176.247, 1176.632,
1177.018, 1177.403, 1177.789, 1178.174,
1178.560, 1178.945, 1179.331, 1179.716, 1180.102, 1180.487,
1180.873, 1181.258,1181.644, 1182.029,
1182.414, 1182.800, 1183.185, 1183.571, 1183.956, 1184.341,
1184.727, 1185.112,1185.597, 1185.883,
1186.268, 1186.653, 1187.038, 1187.424, 1187.809, 1188.194,
1188.579, 1188.965,1189.350, 1189.735,
1190.120, 1190.505, 1190.890, 1191.276, 1191.661, 1192.046,
1192.431, 1192.816, 1193.201, 1193.586,
1193.971, 1194.356, 1194.741, 1195.126, 1195.511, 1195.896,
1196.281, 1196.666, 1197.051, 1197.436,
1197.821, 1198.206, 1198.591, 1198.976, 1199.361, 1199.746,
1200.131, 1200.516, 1200.900, 1201.285,
1201.670, 1202.055, 1202.440, 1202.824, 1203.209, 1203.594,
1203.979, 1204.364, 1204.748, 1205.133,
1205.518, 1205.902, 1206.287, 1206.672, 1207.056, 1207.441,
1207.826, 1208.210, 1208.595, 1208.980,
1209.364, 1209.749, 1210.133, 1210.518, 1210.902, 1211.287,
1211.672, 1212.056, 1212.441, 1212.825,
1213.210, 1213.594, 1213.978, 1214.363, 1214.747, 1215.120,
1215.516, 1215.901, 1216.285, 1216.669,
1217.054, 1217.438, 1217.822, 1218.207, 1218.591, 1218.975,
1219.360, 1219.744, 1220.128, 1220.513,
1220.897, 1221.281, 1221.665, 1222.049, 1222.434, 1222.818,
1223.202, 1223.586, 1223.970, 1224.355,
1224.739, 1225.123, 1225.507, 1225.891, 1226.275, 1226.659,

1227.043, 1227.427, 1227.811, 1228.195,
1228.579, 1228.963, 1229.347, 1229.731, 1230.115, 1230.499,
1230.883, 1231.267, 1231.651, 1232.035,
1232.419, 1232.803, 1233.187, 1233.571, 1233.955, 1234.338,
1234.722, 1235.106, 1235.490, 1235.874,
1236.257, 1236.641, 1237.025, 1237.409, 1237.792, 1238.176,
1238.560, 1238.944, 1239.327, 1239.711,
1240.095, 1240.478, 1240.862, 1241.246, 1241.629, 1242.030,
1242.396, 1242.780, 1243.164, 1243.547,
1243.931, 1244.314, 1244.698, 1245.081, 1245.465, 1245.848,
1246.232, 1246.615, 1246.999, 1247.382,
1247.766, 1248.149, 1248.533, 1248.916, 1249.299, 1249.683,
1250.066, 1250.450, 1250.833, 1251.216,
1251.600, 1251.983, 1252.366, 1252.749, 1253.133, 1253.516,
1253.899, 1254.283, 1254.666, 1255.049,
1255.432, 1255.815, 1256.199, 1256.582, 1256.965, 1257.348,
1257.731, 1258.114, 1258.497, 1258.881,
1259.264, 1259.647, 1260.030, 1260.413, 1260.796, 1261.179,
1261.562, 1261.945, 1262.328, 1262.711,
1263.094, 1263.477, 1263.860, 1264.243, 1264.626, 1265.009,
1265.392, 1265.775, 1266.157, 1266.540,
1266.923, 1267.306, 1267.689, 1268.072, 1268.455, 1268.837,
1269.220, 1269.603, 1269.986, 1270.368,
1270.751, 1271.134, 1271.517, 1271.899, 1272.282, 1272.665,
1273.048, 1273.430, 1273.813, 1274.195,
1274.578, 1274.691, 1274.803, 1274.916, 1275.029, 1275.141,
1275.254, 1275.366, 1275.479, 1275.591,
1278.404, 1278.786, 1279.169, 1279.551, 1279.934, 1280.316,
1280.699, 1281.081, 1281.464, 1281.846,
1282.228, 1282.611, 1282.993, 1283.376, 1283.758, 1284.140,
1284.523, 1284.905, 1285.287, 1285.670,
1286.052, 1286.434, 1286.816, 1287.199, 1287.581, 1287.963,
1288.345, 1288.728, 1289.110, 1289.492,
1289.874, 1290.256, 1290.638, 1291.021, 1291.403, 1291.785,
1292.167, 1292.549, 1292.931, 1293.313,
1293.695, 1294.077, 1294.459, 1294.841, 1295.223, 1295.605,
1295.987, 1296.369, 1296.751, 1297.133,
1297.515, 1297.897, 1298.279, 1298.661, 1299.043, 1299.425,
1299.807, 1300.188, 1300.570, 1300.952,
1301.334, 1301.716, 1302.098, 1302.479, 1302.861, 1303.243,
1303.625, 1304.006, 1304.388, 1304.770,
1305.152, 1305.533, 1305.915, 1306.297, 1306.678, 1307.060,
1307.442, 1307.823, 1308.205, 1308.586,
1308.968, 1309.350, 1309.731, 1310.113, 1310.494, 1310.876,
1311.270, 1311.639,1312.020, 1312.402,
1312.783, 1313.165, 1313.546, 1313.928, 1314.309, 1314.691,
1315.072, 1315.453, 1315.835, 1316.216,
1316.597, 1316.979, 1317.360, 1317.742, 1318.123, 1318.504,
1318.885, 1319.267, 1319.648, 1320.029,
1320.411,1320.792, 1321.173, 1321.554, 1321.935, 1322.316, 1322.697,

1323.079, 1323.460, 1323.841,
1324.222, 1324.603, 1324.985, 1325.366, 1325.747, 1326.128,
1326.509, 1326.890, 1327.271, 1327.652,
1328.033, 1328.414, 1328.795, 1329.176, 1329.557, 1329.938,
1330.319, 1330.700, 1331.081, 1331.462,
1331.843, 1332.224, 1332.604, 1332.985, 1333.366, 1333.747,
1334.128, 1334.509, 1334.889, 1335.270,
1335.651, 1336.032, 1336.413, 1336.793, 1337.174, 1337.555,
1337.935, 1338.316, 1338.697, 1339.078,
1339.458, 1335.839, 1332.220, 1328.600, 1324.981, 1321.361,
1317.742, 1314.123, 1310.503, 1306.884,
1343.264, 1343.645, 1344.025, 1344.406, 1344.786, 1345.167,
1345.570, 1345.928, 1346.308, 1346.689,
1347.069, 1347.450, 1347.830, 1348.211, 1348.591, 1348.971,
1349.352, 1349.732, 1350.112, 1350.493,
1350.873, 1351.253, 1351.634, 1352.014, 1352.394, 1352.774,
1353.155, 1353.535, 1353.915, 1354.295,
1354.676, 1355.056, 1355.436, 1355.816, 1356.196, 1356.577,
1356.957, 1357.337, 1357.717, 1358.097,
1358.477, 1358.857, 1359.237, 1359.617, 1359.997, 1360.377,
1360.757, 1361.137, 1361.517, 1361.897,
1362.277, 1362.657, 1363.037, 1363.417, 1363.797, 1364.177,
1364.557, 1364.937, 1365.317, 1365.697,
1366.077, 1366.456, 1366.836, 1367.216, 1367.596, 1367.976,
1368.355, 1368.735, 1369.115, 1369.495,
1369.875, 1370.254, 1370.634, 1371.014, 1371.393, 1371.773,
1372.153, 1372.532, 1372.912, 1373.292,
1373.671, 1374.051, 1374.431, 1374.810, 1375.190, 1375.569,
1375.949, 1376.329, 1376.708, 1377.088,
1377.467, 1377.847, 1378.226, 1378.606, 1378.985, 1379.365,
1379.744, 1380.123, 1380.503, 1380.882,
1381.262, 1381.641, 1382.020, 1382.400, 1382.779, 1383.158,
1383.538, 1383.917, 1384.296, 1384.676, //99
/*
100 度以后分辨率为 1℃
*/
/*100*/ 1385.055, 1388.847, 1392.638, 1396.428, 1400.217, 1404.005,
1407.791, 1411.576, 1415.360, 1419.143,
/*110*/ 1422.925, 1426.706, 1430.485, 1434.264, 1438.041, 1441.817,
1445.592, 1449.366, 1453.138, 1456.910,
/*120*/ 1460.680, 1464.449, 1468.217, 1471.984, 1475.750, 1479.514,
1483.277, 1487.040, 1490.801, 1494.561,
/*130*/ 1498.319, 1502.077, 1505.833, 1509.589, 1513.343, 1517.096,
1520.847, 1524.598, 1528.381, 1532.139,
/*140*/ 1535.843, 1539.589, 1543.334, 1547.078, 1550.820, 1554.562,
1558.302, 1562.041, 1565.779, 1569.516,
/*150*/ 1573.251, 1576.986, 1580.719, 1584.451, 1588.182, 1591.912,
1595.641, 1599.368, 1603.094, 1606.820,
/*160*/ 1610.544, 1614.267, 1617.989, 1621.709, 1625.429, 1629.147,
1632.864, 1636.580, 1640.295, 1644.009,

```
/*170*/  1647.721,  1651.433,  1655.143,  1658.852,  1662.560,  1666.267,
       1669.972,  1673.677,  1677.380,  1681.082,
/*180*/  1684.783,  1688.483,  1692.181,  1695.879,  1699.575,  1703.271,
       1706.965,  1710.658,  1714.349,  1718.040,
/*190*/  1721.729,  1725.418,  1729.105,  1732.791,  1736.475,  1740.159,
       1743.842,  1747.523,  1751.203,  1754.882,
/*200*/  1758.560,  1762.237,  1765.912,  1769.587,  1773.260,  1776.932,
       1780.603,  1784.273,  1787.941,  1791.610,

/*210*/1795.275,  1798.940,  1802.604,  1806.267,  1809.929,  1813.590,
       1817.249,  1820.907,  1824.564,  1828.220,
/*220*/1831.875,  1835.529,  1839.181,  1842.832,  1846.483,  1850.132,
       1853.779,  1857.426,  1861.072,  1864.716,
/*230*/1868.359,  1872.001,  1875.642,  1879.282,  1882.921,  1886.558,
       1890.194,  1893.830,  1897.463,  1901.096,
/*240*/1904.728,  1908.359,  1911.988,  1915.616,  1919.243,  1922.869,
       1926.494,  1930.117,  1933.740,  1937.361,
/*250*/1940.981,  1944.600,  1948.218,  1951.835,  1955.450,  1959.065,
       1962.678,  1966.290,  1969.901,  1973.510,
/*260*/1977.119,  1980.726,  1984.333,  1987.938,  1991.542,  1995.145,
       1998.746,  2002.347,  2005.946,  2009.544,
/*270*/2013.141,  2016.737,  2020.332,  2023.925,  2027.518,  2031.109,
       2034.699,  2038.288,  2041.876,  2045.463,
/*280*/2049.048,  2052.632,  2056.215,  2059.798,  2063.378,  2066.958,
       2070.537,  2074.114,  2077.690,  2081.265,
/*290*/2084.839,  2088.412,  2091.984,  2095.554,  2099.123,  2102.692,
       2106.259,  2109.824,  2113.389,  2116.953,
/*300*/ 2120.515,  2124.08,  2127.64,  2131.20,  2134.75,  2138.31,  2141.87,  2145.42,  2148.97,
       2152.52,
/*310*/ 2156.08,  2159.62,  2163.17,  2166.72,  2170.27,  2173.81,  2177.36,  2180.90,  2184.44,
       2187.98,
/*320*/ 2191.52,  2195.06,  2198.60,  2202.13,  2205.67,  2209.20,  2212.73,  2216.26,  2219.79,
       2223.32,
/*330*/ 2226.85,  2230.38,  2233.90,  2237.43,  2240.95,  2244.47,  2247.99,  2251.51,  2255.03,
       2258.55,
/*340*/ 2262.06,  2265.58,  2269.09,  2272.60,  2276.12,  2279.63,  2283.14,  2286.64,  2290.15,
       2293.66,
/*350*/ 2297.16
};

/**********全局变量**********/
#define AD_LEN 2//DMA 获取长度
uint16_t ADC_Val[AD_LEN];//adc 的值  0:台面温度 ad 值。  1：外部探头 ad 值
uint32_t ADC1_Val,ADC2_Val;//adc 的值
#define OP_Value   6.8//放大系数
#define Vref_3V3   3.30//3.3V 电压
#define K1              0.2327367//电阻基准系数
#define Vref          2.494//参考电压
#define IIR(x,y) (x)=((x) * 9 + (y)) / 10//滤波
```

```
/*
**********************************************************************
 * 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 100//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];

	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];

	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：void AFE_Sample_Handler(void)
 * 功      能：计算阻值
**********************************************************************
*/
float temp_correct,temp_correct1;//温度系数
float ADC_Val_Avg[2];//0 为台面温度   1 为探头温度
float AD_T1=0.0;//ADC 计算后的电压值
float AD_T2=0.0;//ADC 计算后的电压值
float PT_VALUE_1_TEMP;//外部探头阻值
float PT_VALUE_2_TEMP;//台面探头阻值
void AFE_Sample_Handler(void)
{
//    temp_correct=1.0104f;//外部


    temp_correct = 1.00167f;

    temp_correct1 = 1.0f;

    ADC_Val_Avg[0] = Filter_ADC();
    ADC_Val_Avg[1] = Filter_ADC1();
    AD_T1=((float)ADC_Val_Avg[1]*Vref_3V3/4096)/OP_Value/Vref+K1;//计算电压值
    PT_VALUE_1_TEMP=3000*AD_T1/(1-AD_T1)*temp_correct;//计算电阻值
    AD_T2=((float)ADC_Val_Avg[0]*Vref_3V3/4096)/OP_Value/Vref+K1;//计算电压值
    PT_VALUE_2_TEMP=3000*AD_T2/(1-AD_T2)*temp_correct1;//计算电阻值
}

/*
**********************************************************************
 * 函数原型：int AFE_GetTemperature(float tmp)
 * 功      能：查表
**********************************************************************
*/
int AFE_GetTemperature(float tmp)
{
    int temp;
    if(tmp<1000)//小于 0 摄氏度
    {
```

```
        for(int j = 0; j < 220; j++)
        {
            if(tmp < Temp_map[j])
            {
                temp = j-220;
                break;
            }
        }
    }

    else if((tmp<Temp_map[1219])&&(tmp>=1000))//小于 99.9 摄氏度
    {
        for(int j = 220;j < 1220; j++)
        {
            if (tmp < Temp_map[j])
            {
                temp = j-220;
                break;
            }
        }
    }

    else
    {
        for(int j = 1220; j < 1501; j++)
        {
            if(tmp < Temp_map[j])
            {
                temp = (j-1220)*10+1000;
                break;
            }
        }
    }
    return temp;
}

/*
*********************************************************************************
 * 函数原型：float Get_ADCVal(int16_t temp)
 * 功      能：查表读 ADC 值
*********************************************************************************
*/
float Get_ADCVal(int16_t temp)
{
    float adc_Val;
    if(temp < 0)//小于 0 摄氏度
    {
        adc_Val   =   Temp_map[220 + temp];
    }
    else if(temp >= 0 && temp < 1000)
```

```
    {
        adc_Val = Temp_map[220 + temp];
    }
    else if(temp >= 1000)
    {
        adc_Val = Temp_map[1220 + (temp-1000)/10];
    }
    return adc_Val;
}

/*
**********************************************************************
 * 函数原型：void ADCDMA_Init(void)
 * 功　　能：ADC 和 DMA 的初始化
**********************************************************************
*/
void ADCDMA_Init(void)
{
    HAL_ADCEx_Calibration_Start(&hadc);
    HAL_TIM_Base_Start_IT(&htim15);//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
    for(uint8_t i = 0; i < 200; i++)
    {
        AFE_Sample_Handler();//计算阻值
        Read_Temp(0.6f);
    }
}

/*
**********************************************************************
 * 函数原型：void Read_Temp(float dT)
 * 功　　能：读取温度-10ms
**********************************************************************
*/
void Read_Temp(float dT)
{
    static float T;
    Temp.Outside = AFE_GetTemperature(PT_VALUE_1_TEMP);//外部温度
    Temp.Mesa = AFE_GetTemperature(PT_VALUE_2_TEMP);//台面温度
    T += dT;
    AFE_Sample_Handler();//计算阻值

    if(T >= 1.0f)
    {
        if(PT_VALUE_1_TEMP < 2200)//假如插入外部探头
            Temp.Rel = Temp.Outside;//真实温度显示外部探头测的温度
        else//假如没有插入外部探头
            Temp.Rel = Temp.Mesa;//真实温度显示台面温度
        T = 0;
    }
```

```
}

#include "Drv_Motor.h"

/*
*********************************************************************
 * 函数原型：void Motor_Init(void)
 * 功      能：电机初始化
*********************************************************************
*/
void Motor_Init(void)
{
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_4);//开启 tim3 通道四
}
#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 "Show.h"

/**********全局变量声明******/
```

```
float Twinkle_Time;//闪烁时间

/**********局部变量声明******/
uint8_t Tab[] = {0x77,0x24,0x5D,0x6D,0x2E,0x6B,0x7B,0x25,0x7F,0x6F};//0·9
uint8_t Tab1[] = {0x77,0x12,0x5D,0x5B,0x3A,0x6B,0x6f,0x52,0x7F,0x7B};//0·9
uint8_t Temp_ShowFlag,Speed_ShowFlag,Time_ShowFlag;//温度、速度、时间显示的标志位 0:常亮 1：熄灭
uint8_t  TempIcn_ShowFlag,TimeIcn_ShowFlag,SpeedIcn_ShowFlag;//加热图标闪烁和时间图标闪烁和速度图标闪烁

/*
*********************************************************************
 * 函数原型：static void Icn_Twinkle(float dT)
 * 功      能：图标闪烁
 * 调      用：内部调用
*********************************************************************
*/
static void Icn_Twinkle(float dT)
{
    static float T;
    if(sys.Run_Status)
    {
        T += dT;
        if(T >= 0.5f)
        {
            if(Speed.Set)
            {
                SpeedIcn_ShowFlag ++;//速度图标闪烁;
                if(SpeedIcn_ShowFlag > 3)
                    SpeedIcn_ShowFlag = 1;
            }
            if(Temp.Set)
                TempIcn_ShowFlag = ~TempIcn_ShowFlag;//温度图标闪烁;
            if(Time.Rel && (!Temp.Ctrl || TempIcn_ShowFlag != TimeIcn_ShowFlag) && (Temp.Set == 0 || Temp.ADDMode == 3))
                TimeIcn_ShowFlag = ~TimeIcn_ShowFlag;//时间图标闪烁;
            T = 0;
        }
    }
    else
    {
        SpeedIcn_ShowFlag = 0;//不显示速度图标
        TempIcn_ShowFlag = 0;//不显示温度图标
        TimeIcn_ShowFlag = 0;//不显示时间图标
    }
}

/*
*********************************************************************
 * 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.SetMode_Option == 0)//如果没在设置选项中，则都点亮，不闪烁
    {
        Speed_ShowFlag = 0;//常亮
        Temp_ShowFlag = 0;//常亮
        Time_ShowFlag = 0;//常亮
        Twinkle_Time = 0;//闪烁计时清零
        return;
    }
    if(Twinkle_Time && Key_Status==0)//闪烁和没有操作按键时
    {
        T += dT;
        if(T >= 0.5f)
        {
            Twinkle_Time -= 0.5;//闪烁计时
            if(sys.SetMode_Option == 1)//设置温度
            {
                Temp_ShowFlag = ~Temp_ShowFlag;//温度闪烁
                Speed_ShowFlag = 0;//速度常亮
                Time_ShowFlag = 0;//时间常亮
            }
            else if(sys.SetMode_Option == 2)//设置速度
            {
                Temp_ShowFlag = 0;//温度常亮
                Speed_ShowFlag = ~Speed_ShowFlag;//速度闪烁
                Time_ShowFlag = 0;//时间常亮
            }
            else if(sys.SetMode_Option == 3)//设置时间
            {
                Temp_ShowFlag = 0;//温度常亮
                Speed_ShowFlag = 0;//速度常亮
                Time_ShowFlag = ~Time_ShowFlag;//时间闪烁
            }
            if(Twinkle_Time == 0)//如果闪烁结束
            {
                sys.SetMode_Option = 0;//模式选择清零
            }
            T = 0;
        }
    }
    else
    {
```

```
            Speed_ShowFlag = 0;//常亮
            Temp_ShowFlag = 0;//常亮
            Time_ShowFlag = 0;//常亮
            T = 0;
        }
}

/*
**********************************************************************
 * 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
    Icn_Twinkle(dT);//图标闪烁
}

/*
**********************************************************************
 * 函数原型：void Display_Temp(int16_t dis_set_temp,int16_t dis_rel_temp)
 * 功      能：显示温度
 * 输      入: dis_set_temp  设定温度    dis_rel_temp  实际温度
 * 参      数：int16_t dis_set_temp,int16_t dis_rel_temp
**********************************************************************
*/
void Display_Temp(int16_t dis_set_temp,int16_t dis_rel_temp)
{
    uint8_t seg1,seg2,seg3,seg4,seg5,seg6,seg7,seg8;
    seg1=0;seg2=0;seg3=0;seg4=0;seg5=0;seg6=0;seg7=0;seg8=0;
    uint16_t Val;//用于百十个取出来的数字

    /***********设定温度计算**********/
    if(Temp_ShowFlag == 0)
    {
        if(dis_set_temp > 0)
        {
            if(dis_set_temp > 999)//大于 999 时
            {
                Val=dis_set_temp/1000;//取出千位
                seg1 = Tab1[Val];
            }
            else
            {
                Val = 0x00;//不显示
            }
            if(dis_set_temp > 99)//大于 99 时
            {
                Val=dis_set_temp/100;//取出百位
```

```
                if(dis_set_temp > 999)//大于 999 时
                    Val=Val%10;//取出百位
                seg2 = Tab1[Val];
            }
            else
            {
                seg2 = 0x00;//不显示
            }
            if(dis_set_temp > 9)//大于 9 时
            {
                Val=dis_set_temp/10;//取出十位
                if(dis_set_temp > 99)//大于 99 时
                    Val=Val%10;//取出十位
                seg3 = Tab1[Val];
            }
            else
            {
                seg3 = Tab1[0];//不显示 0
            }
            Val=dis_set_temp%10;//取出个位
            seg4 = Tab1[Val];
            seg3 &= 0x7F;seg3 |= 0x80;//设定温度的小数点
        }
        else
        {
            seg1 = 0x08;//显示"-"
            seg2 = 0x08;//显示"-"
            seg3 = 0x08;//显示"-"
            seg4 = 0x08;//显示"-"
            seg3 &= 0x7F;seg3 |= 0x00;//不显示设定温度的小数点
        }
    }
    else
    {
        seg1 = 0x00;//不显示设定温度
        seg2 = 0x00;//不显示设定温度
        seg3 = 0x00;//不显示设定温度
        seg4 = 0x00;//不显示设定温度
        seg3 &= 0x7F;seg3 |= 0x00;//不显示设定温度的小数点
    }

    /***********实际温度计算**********/
    if(dis_rel_temp > 999)//大于 999 时
    {
        Val=dis_rel_temp/1000;//取出千位
        seg5 = Tab[Val];
    }
    else
    {
        seg5 = 0x00;//不显示
```

```
    }
    if(dis_rel_temp > 99)//大于 99 时
    {
        Val=dis_rel_temp/100;//取出百位
        if(dis_rel_temp > 999)//大于 999 时
            Val=Val%10;//取出百位
        seg6 = Tab[Val];
    }
    else
    {
        seg6 = 0x00;//不显示
    }
    if(dis_rel_temp > 9)//大于 9 时
    {
        Val=dis_rel_temp/10;//取出十位
        if(dis_rel_temp > 99)//大于 99 时
            Val=Val%10;//取出十位
        seg7 = Tab[Val];
    }
    else
    {
        seg7 = Tab[0];//不显示 0
    }
    Val=dis_rel_temp%10;//取出个位
    seg8 = Tab[Val];

    /**************温度小数点的图标********/
    seg7 &= 0x7F;seg7 |= 0x80;//实际温度的小数点

    /*****************℃*******************/
    seg4 &= 0x7F;seg4 |= 0x80;//℃

    /**************加热图标****************/
//  if(sys.Run_Status && Temp.Ctrl_Temp)
    if(TempIcn_ShowFlag == 0)
    {
        seg6 &= 0x7F;seg6 |= 0x80;//加热图标底部
        seg8 &= 0x7F;seg8 |= 0x80;//加热图标上半部分
    }
    else
    {
        seg6 &= 0x7F;seg6 |= 0x00;//不显示加热图标底部
        seg8 &= 0x7F;seg8 |= 0x00;//不显示加热图标上半部分
    }

    /**************发送数据****************/
    Write_Addr_Dat_N(38, seg1,1);
    Write_Addr_Dat_N(36, seg2,1);
    Write_Addr_Dat_N(34, seg3,1);
    Write_Addr_Dat_N(32, seg4,1);
```

```
    Write_Addr_Dat_N(0, seg5,1);
    Write_Addr_Dat_N(2, seg6,1);
    Write_Addr_Dat_N(4, seg7,1);
    Write_Addr_Dat_N(6, seg8,1);
}

/*
**********************************************************************
 * 函数原型：void Display_Speed(int16_t dis_set_speed,int16_t dis_rel_speed)
 * 功    能：显示转速
 * 输    入: dis_set_speed 设定转速   dis_rel_speed 实际转速
 * 参    数：int16_t dis_set_speed,int16_t dis_rel_speed
**********************************************************************
*/
void Display_Speed(int16_t dis_set_speed,int16_t dis_rel_speed)
{
    uint8_t seg1,seg2,seg3,seg4,seg5,seg6,seg7,seg8;
    seg1=0;seg2=0;seg3=0;seg4=0;seg5=0;seg6=0;seg7=0;seg8=0;
    uint16_t Val;//用于百十个取出来的数字
    if(Speed_ShowFlag == 0)
    {
        if(dis_set_speed > 0)
        {
            /***********设定转速计算**********/
            if(dis_set_speed > 999)//大于 999 时
            {
                Val=dis_set_speed/1000;//取出千位
                seg1 = Tab1[Val];
            }
            else
            {
                seg1 = 0x00;//显示 0
            }
            if(dis_set_speed > 99)//大于 99 时
            {
                Val=dis_set_speed/100;//取出百位
                if(dis_set_speed > 999)//大于 999 时
                    Val=Val%10;//取出百位
                seg2 = Tab1[Val];
            }
            else
            {
                seg2 = 0x00;//显示 0
            }
            if(dis_set_speed > 9)//大于 9 时
            {
                Val=dis_set_speed/10;//取出十位
                if(dis_set_speed > 99)//大于 99 时
                    Val=Val%10;//取出十位
                seg3 = Tab1[Val];
```

```
            }
            else
            {
                seg3 = 0x00;//显示 0
            }
            Val=dis_set_speed%10;//取出个位
            seg4 = Tab1[Val];
        }
        else
        {
            seg1 = 0x08;//显示"-"
            seg2 = 0x08;//显示"-"
            seg3 = 0x08;//显示"-"
            seg4 = 0x08;//显示"-"
        }
    }
    else
    {
        seg1 = 0x00;//不显示设定速度
        seg2 = 0x00;//不显示设定速度
        seg3 = 0x00;//不显示设定速度
        seg4 = 0x00;//不显示设定速度
    }

    /***********实际转速计算**********/
    if(dis_rel_speed > 999)//大于 999 时
    {
        Val=dis_rel_speed/1000;//取出千位
        seg5 = Tab[Val];
    }
    else
    {
        seg5 = 0x00;//显示 0
    }
    if(dis_rel_speed > 99)//大于 99 时
    {
        Val=dis_rel_speed/100;//取出百位
        if(dis_rel_speed > 999)//大于 999 时
            Val=Val%10;//取出百位
        seg6 = Tab[Val];
    }
    else
    {
        seg6 = 0x00;//显示 0
    }
    if(dis_rel_speed > 9)//大于 9 时
    {
        Val=dis_rel_speed/10;//取出十位
        if(dis_rel_speed > 99)//大于 99 时
            Val=Val%10;//取出十位
```

```
        seg7 = Tab[Val];
    }
    else
    {
        seg7 = 0x00;//显示 0
    }
    Val=dis_rel_speed%10;//取出个位
    seg8 = Tab[Val];


    /***************rpm******************/
    seg4 &= 0x7F;seg4 |= 0x80;//rpm

    /*************转速图标***************/
    switch(SpeedIcn_ShowFlag)
    {
        case 0: seg5 &= 0x7F;seg5 |= 0x80;//转速 S6
                seg6 &= 0x7F;seg6 |= 0x80;//转速 S5
                seg7 &= 0x7F;seg7 |= 0x80;//转速 S7
            break;
        case 1: seg5 &= 0x7F;seg5 |= 0x80;//转速 S6
                seg6 &= 0x7F;seg6 |= 0x80;//转速 S5
                seg7 &= 0x7F;seg7 |= 0x00;//转速 S7
            break;
        case 2: seg5 &= 0x7F;seg5 |= 0x80;//转速 S6
                seg6 &= 0x7F;seg6 |= 0x00;//转速 S5
                seg7 &= 0x7F;seg7 |= 0x80;//转速 S7
            break;
        case 3: seg5 &= 0x7F;seg5 |= 0x00;//转速 S6
                seg6 &= 0x7F;seg6 |= 0x80;//转速 S5
                seg7 &= 0x7F;seg7 |= 0x80;//转速 S7
            break;
        default:
            break;
    }

    /*************发送数据***************/
    Write_Addr_Dat_N(22, seg1,1);
    Write_Addr_Dat_N(20, seg2,1);
    Write_Addr_Dat_N(18, seg3,1);
    Write_Addr_Dat_N(16, seg4,1);
    Write_Addr_Dat_N(8, seg5,1);
    Write_Addr_Dat_N(10, seg6,1);
    Write_Addr_Dat_N(12, seg7,1);
    Write_Addr_Dat_N(14, seg8,1);
}

/*
*************************************************************
 * 函数原型：void Display_Time(int32_t dis_time)
```

```
 * 功      能：显示时间
 * 输      入: dis_time 时间
 * 参      数：int32_t dis_time
*****************************************************************
*/
void Display_Time(int32_t dis_time)
{
    uint8_t seg1,seg2,seg3,seg4;
    seg1=0;seg2=0;seg3=0;seg4=0;
    uint8_t SH,H,SM,M;//时间的单位取值

    if(Time.Set || sys.SetMode_Option == 3)//设定时间大于 0，在设定时间和闪烁下
    {
        if(!Time_ShowFlag)
        {
            if(Time.Set)//假如设定时间大于 0
            {
                /**************时间计算**************/
                SH=dis_time/3600/10;//计算十位单位的小时数
                H=dis_time/3600%10;//计算个位单位的小时数
                SM=dis_time%3600/60/10;//计算十分位单位的分钟数
                M=dis_time%3600/60%10;//计算个分位单位的分钟数
                seg1 = Tab1[SH];
                seg2 = Tab1[H];
                seg3 = Tab1[SM];
                seg4 = Tab1[M];
            }
            else
            {
                seg1 = 0x08;//显示"-"
                seg2 = 0x08;//显示"-"
                seg3 = 0x08;//显示"-"
                seg4 = 0x08;//显示"-"
            }
        }
        else//设定时间等于 0
        {
            seg1 = 0x00;//不显示时间
            seg2 = 0x00;//不显示时间
            seg3 = 0x00;//不显示时间
            seg4 = 0x00;//不显示时间
        }
    }
    else//设定时间等于 0
    {
        seg1 = 0x00;//不显示时间
        seg2 = 0x00;//不显示时间
        seg3 = 0x00;//不显示时间
        seg4 = 0x00;//不显示时间
    }
```

```
    if(Time.Set > 0 || sys.SetMode_Option == 3)
    {
        /************时间冒号图标************/
        seg2 &= 0x7F;seg2 |= 0x80;//时间冒号

        /************时间单位图标***********/
        seg4 &= 0x7F;seg4 |= 0x80;//min

        /************时间图标**************/
        if(TimeIcn_ShowFlag == 0)
        {
            seg1 &= 0x7F;seg1 |= 0x80;//时间图标
        }
        else
        {
            seg1 &= 0x7F;seg1 |= 0x00;//不显示时间图标
        }
    }
    else
    {
        seg2 &= 0x7F;seg2 |= 0x00;//不显示时间冒号
        seg4 &= 0x7F;seg4 |= 0x00;//不显示 min
        seg1 &= 0x7F;seg1 |= 0x00;//不显示显示时间图标
    }

    /*************外部探头的图标*********/
    if(PT_VALUE_1_TEMP < 2200)//假如插入外部探头
    {
        seg3 &= 0x7F;seg3 |= 0x80;//外部探头的图标
    }
    else
    {
        seg3 &= 0x7F;seg3 |= 0x00;//不显示外部探头的图标
    }

    /*************发送数据**************/
    Write_Addr_Dat_N(30, seg1,1);
    Write_Addr_Dat_N(28, seg2,1);
    Write_Addr_Dat_N(26, seg3,1);
    Write_Addr_Dat_N(24, seg4,1);
}

/*
*********************************************************************
 * 函数原型：void Deal_Speed(float dT)
 * 功      能：速度显示处理
*********************************************************************
*/
void Deal_Speed(float dT)
```

```
{
    if(sys.Run_Status)
    {
        if(Speed.ADDMode==0)//在进入加速模式下
        {
            if(Speed.Display_Rel >= Speed.Ctrl)//当前的速度大于等于控制速度
            {
                Speed.ADDMode = 2;//进入稳定模式
                return;
            }
            Speed.New = Speed.Rel;//记录当前速度
            if(Speed.New > Speed.Last)//当前速度大于上一次速度
                Speed.Display_Rel = Speed.New;//显示当前速度
            else//当前速度小于上一次速度
            {
                Speed.Display_Rel = Speed.Last;//显示上一次速度，不让速度小于当前速
度。呈现攀升速度的现象
                Speed.New = Speed.Last;//将上一次速度赋值给当前速度
            }
            Speed.Last = Speed.New;//将当前速度保存
        }
        else if(Speed.ADDMode==1)//在进入减速模式下
        {
            if(Speed.Display_Rel <= Speed.Ctrl)//当前的速度大于等于控制速度
            {
                sys.Run_Status = 0;//关闭系统
                SetOK_Flag = 1;//设置标志置一
                return;
            }
            Speed.New = Speed.Rel;//记录当前速度
            if(Speed.New < Speed.Last)//当前速度小于上一次速度
                Speed.Display_Rel = Speed.New;//显示当前速度
            else//当前速度大于上一次速度
            {
                Speed.Display_Rel = Speed.Last;//显示上一次速度，不让速度大于当前速
度。呈现下降速度的现象
                Speed.New = Speed.Last;//将上一次速度赋值给当前速度
            }
            Speed.Last = Speed.New;//将当前速度保存
        }
        else if(Speed.ADDMode == 2)//速度稳定模式下
        {
            Speed.Display_Rel = Speed.Ctrl;//显示控制速度
        }
    }
    else
    {
        Speed.New =0;//现在的速度清零
        Speed.Last = 0;//之前的速度清零
        Speed.ADDMode = 0;//清除显示处理
```

```
    }
}

/*
*********************************************************************************
 * 函数原型：void Deal_Temp(float dT)
 * 功      能：温度显示处理
*********************************************************************************
*/

void Deal_Temp(float dT)
{
    static float T;
    uint8_t val;
    if(sys.Run_Status && Temp.Ctrl)
    {
        if(PT_VALUE_1_TEMP < 2200)//假如插入外部探头
            val = 30;
        else//假如没有插入外部探头
            val = 40;
        if(ABS(Temp.Ctrl - Temp.Rel) < val)
        {
            if(Temp.ADDMode==0)//判断模式
            {
                if(Temp.Ctrl > Temp.Rel)
                {
                    Temp.ADDMode = 1;//进入升温模式
                    Temp_Arg.Kd = 30;
                    Temp.Last = Temp.Rel;
                }
                else if(Temp.Ctrl < Temp.Rel)
                {
                    Temp.ADDMode = 2;//进入降温模式
                    Temp_Arg.Kd = 0;
                    Temp.Last = Temp.Rel;//记录当前温度
                }
                else
                {
                    Temp.ADDMode = 3;//进入稳定模式
                }
            }

            else if(Temp.ADDMode==1)//在进入升温模式下
            {
                Temp.New = Temp.Rel;//记录当前温度
                if(Temp.New > Temp.Last)//当前温度大于上一次温度
                    Temp.Display_Rel = Temp.New;//显示当前温度
                else//当前温度小于上一次温度
                {
                    if(Temp.Ctrl == 1000)
```

```
                {
                    if(Temp.Display_Rel == Temp.Last)
                    {
                        T += dT;
                        if(T > 10)
                        {
                            Temp.Display_Rel += 1;
                            Temp.Last = Temp.Display_Rel;
                            T = 0;
                        }
                    }
                    else
                    {
                        T = 0;
                    }
                }
                    Temp.Display_Rel = Temp.Last;//显示上一次温度，不让温度小于当前温度。呈现攀升速度的现象
                    Temp.New = Temp.Last;//将上一次温度赋值给当前温度
                }
                Temp.Last = Temp.New;//将当前温度保存
                if(Temp.Display_Rel >= Temp.Ctrl)//当前的温度大于等于控制温度
                {
                    Temp.ADDMode = 3;//进入稳定模式
                    Temp_Val.Integral = 0;//温度控制积分清零
                    Temp_Arg.Kd = 10;//pid 参数调整
                }
            }

            else if(Temp.ADDMode==2)//在进入降温模式下
            {
                Temp.New = Temp.Rel;//记录当前温度
                if(Temp.New < Temp.Last)//当前温度小于上一次温度
                    Temp.Display_Rel = Temp.New;//显示当前温度
                else//当前温度大于上一次温度
                {
                    Temp.Display_Rel = Temp.Last;//显示上一次温度，不让温度小于当前温度。呈现攀升速度的现象
                    Temp.New = Temp.Last;//将上一次温度赋值给当前温度
                }
                Temp.Last = Temp.New;//将当前温度保存
                if(Temp.Display_Rel <= Temp.Ctrl)//当前的温度小于等于控制温度
                {
                    Temp.ADDMode = 3;//进入稳定模式
                    Temp_Val.Integral = 0;//温度控制积分清零
                    Temp_Arg.Kd = 10;//pid 参数调整
                }
            }

            else if(Temp.ADDMode == 3)//温度稳定模式下
```

```
                {
                    Temp.Display_Rel = Temp.Ctrl;//显示控制温度
                }

            }
            else
            {
                Temp.ADDMode = 0;//进入稳定模式
                Temp.Display_Rel = Temp.Rel;//显示实际温度
            }
        }
        else
        {
            Temp.Display_Rel = Temp.Rel;//显示实际温度
            Temp.New =0;//现在的速度清零
            Temp.Last = 0;//之前的速度清零
            Temp.ADDMode = 0;//清除显示处理
            T = 0;
        }
}

/*
**********************************************************************
 * 函数原型：  void Show_Display(void)
 * 功      能：  显示屏幕内容
**********************************************************************
*/
void Show_Display(void)
{
    Temp.Display_Set = Temp.Set;//显示设定温度

    Speed.Display_Set = Speed.Set;//显示设定转速

    if(sys.Run_Status)
        Time.Display = Time.Rel + 59;//显示控制时间
    else
        Time.Display = Time.Set;//显示设定时间

    Display_Temp(Temp.Display_Set,Temp.Display_Rel);//显示温度
    Display_Speed(Speed.Display_Set,Speed.Display_Rel);//显示转速
    Display_Time(Time.Display);//显示时间
}
#include "Speed.h"

/*
**********************************************************************
 * 函数原型：void Encoder_Init(void)
 * 功      能：编码器初始化
**********************************************************************
*/
```

```
void Encoder_Init(void)
{
    HAL_TIM_Base_Start_IT(&htim1);
    HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_1);//开启 time1 通道 1 输入捕获
}

/*
**********************************************************************
 * 函数原型：void Check_Speed(float dT)
 * 功    能：检测速度是否停止-0.05s
**********************************************************************
*/
void Check_Speed(float dT)
{
    Speed.Stop_Cnt += dT;//每 50ms 进入
    if(Speed.Stop_Cnt >= 1.0)//0.5s 发现没出发输入捕获
    {
        Speed.Rel = 0;//将速度清零
        Speed.Stop_Cnt = 0;//计数清零
    }
}

/*
**********************************************************************
 * 函数原型：void TIM1CaptureChannel1Callback(void)
 * 功    能：Tim1 通道 1 的输入捕获回调函数
**********************************************************************
*/
uint32_t L1_capture,L1_capture1,L1_capture2;
float rel1;
void TIM1CaptureChannel1Callback(void)
{
    L1_capture1=__HAL_TIM_GET_COMPARE(&htim1, TIM_CHANNEL_1);//获取 Tim1 通
道 1 的输入捕获
    if(L1_capture1>L1_capture2)
        L1_capture=L1_capture1-L1_capture2;
    else
        L1_capture=L1_capture1+(0xFFFF-L1_capture2);
    if(L1_capture<100)
        return;
    rel1=10000.0f/(float)L1_capture;//计算速度
    L1_capture2=L1_capture1;
    Speed.Rel=rel1*60/2;//将速度赋值给 L1 的实际速度
    Speed.Stop_Cnt = 0;
}

/*
**********************************************************************
 * 函数原型：  void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
 * 功    能：  TIM_IC 回调函数
```

```
**********************************************************
*/
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
	if(htim->Instance == TIM1)
	{
		if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_1)
		{
			TIM1CaptureChannel1Callback();
		}
	}
}
```