# DBS3100_L 软件源程序

```
#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; //  保存标志位

/**
 * @brief  初始化硬件中的参数
 *
 */
void Param_Reset(void)
{
    Param.Flash_Check_Start = FLASH_CHECK_START;

    for (uint8_t i = 0; i <= 9; i++)
    {
        Param.P_Param[i][0] = 370;    //  温度
        Param.P_Param[i][1] = 1500; //  速度
        Param.P_Param[i][2] = 5;      //  时间
    }

    Param.Flash_Check_End = FLASH_CHECK_END;
}

/**
 * @brief  保存硬件中的参数
 *
 */
void Param_Save(void)
{
    Flash_Write((uint8_t *)(&Param), sizeof(Param));
}

/**
 * @brief  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();
        Temp.Set = Param.P_Param[PMode.Option][0];     //  将 Flash 中的温度赋值
        Speed.Set = Param.P_Param[PMode.Option][1];     //  将 Flash 中的速度赋值
```

```
            Time.Set = Param.P_Param[PMode.Option][2];       // 将 Flash 中的时间赋值
            SetOK_Flag = 1;
            Save_Param_En = 1;
        }
        else
        {
            Temp.Set = Param.P_Param[PMode.Option][0];   // 将 Flash 中的温度赋值
            Speed.Set = Param.P_Param[PMode.Option][1]; // 将 Flash 中的速度赋值
            Time.Set = Param.P_Param[PMode.Option][2];   // 将 Flash 中的时间赋值
            SetOK_Flag = 1;
        }

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

/**
 * @brief 保存标志位置 1，0.5s 后保存
 *
 * @param dT 任务周期
 */
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 "PID.h"

/**
 * @brief 微分先行 PID 计算
 *
 * @param dT 周期（单位：秒）
 * @param Expect 期望值（设定值）
 * @param Freedback 反馈值
```

```
 * @param PID_Arg PID 参数结构体
 * @param PID_Val PID 数据结构体
 * @param Error_Lim  误差限幅
 * @param Integral_Lim  积分误差限幅
 */
void AltPID_Calculation(float dT, float Expect, float Freedback, _PID_Arg_ *PID_Arg, _PID_Val_ *PID_Val, float Error_Lim, float Integral_Lim)
{
    PID_Val->Error = Expect - Freedback; // 误差 = 期望值-反馈值

    PID_Val->Proportion = PID_Arg->Kp * PID_Val->Error; // 比例 = 比例系数*误差
    PID_Val->Fb_Differential = -PID_Arg->Kd * ((Freedback - PID_Val->Freedback_Old) * safe_div(1.0f, dT, 0)); // 微分 = -（微分系数） *（当前反馈值-上一次反馈值） *频率
    PID_Val->Integral += PID_Arg->Ki * LIMIT(PID_Val->Error, -Error_Lim, Error_Lim) * dT; // 积分 = 积分系数*误差*周期
    PID_Val->Integral = LIMIT(PID_Val->Integral, -Integral_Lim, Integral_Lim); // 积分限幅

    PID_Val->Out = PID_Val->Proportion + PID_Val->Integral + PID_Val->Fb_Differential; // PID 输出

    PID_Val->Freedback_Old = Freedback; // 将当前反馈值赋值给上一次反馈值
}

/**
 * @brief 增量式 PID 计算
 *
 * @param dT dT：周期（单位：秒）
 * @param Expect 期望值（设定值）
 * @param Freedback 反馈值
 * @param PID_Arg PID 参数结构体
 * @param PID_Val PID 数据结构体
 * @param Integral_Lim 积分误差限幅
 */
void IncPID_Calculation(float dT, float Expect, float Freedback, _PID_Arg_ *PID_Arg, _PID_Val_ *PID_Val, float Integral_Lim)
{
    PID_Val->Error = Expect - Freedback; // 误差 = 期望值-反馈值

    PID_Val->Proportion = PID_Arg->Kp * (PID_Val->Error - PID_Val->Error_Last); // 比例 = 比例系数*（当前误差-上一次误差）
    PID_Val->Integral = PID_Arg->Ki * PID_Val->Error * dT; // 积分 = 积分系数*误差*周期
    PID_Val->Integral = LIMIT(PID_Val->Integral, -Integral_Lim, Integral_Lim); // 积分限幅
    PID_Val->Differential = PID_Arg->Kd * (PID_Val->Error - 2.0f * PID_Val->Error_Last + PID_Val->Error_Previous) * safe_div(1.0f, dT, 0); // 微分 = 微分系数 *（当前误差-2*上一次误差+上上次误差）*频率
```

```
    PID_Val->Out += PID_Val->Proportion + PID_Val->Integral + PID_Val->Differential; // PID 输出

    PID_Val->Error_Previous = PID_Val->Error_Last; // 将上一次误差赋值给上上次误差
    PID_Val->Error_Last = PID_Val->Error;          // 将当前误差赋值给上一次误差
}
#include "SetVal.h"

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

/**
 * @brief 检测设置
 *
 * @param dT 任务周期
 */
void Check_Set(float dT)
{
    if (Key_Status)
    {
        SetOK_Flag = 1; // 检测到设置，等待退出设置模式
    }
    if (SetOK_Flag)
    {

        if (Temp.Ctrl != Temp.Set)
        {
            Temp.Ctrl = Temp.Set;
            Param.P_Param[PMode.Option][0] = Temp.Set;
        }
        if (Speed.Ctrl != Speed.Set)
        {
            Speed.Ctrl = Speed.Set;
            Param.P_Param[PMode.Option][1] = Speed.Set;
            if (Speed.ADDMode != 0)
                Speed.ADDMode = 0;
        }
        if (Time.Rel != Time.Set)
        {
            Time.Rel = Time.Set;
            Param.P_Param[PMode.Option][2] = Time.Set;
        }
        Save_Param_En = 1; // 保存
        SetOK_Flag = 0;
    }
}
#include "Show.h"

/**********全局变量**********/
float Twinkle_Time; // 闪烁时间
```

```
/**********局部变量声明*******/
uint8_t Tab[] = {0xF5, 0x05, 0xD3, 0x97, 0x27, 0xB6, 0xF6, 0x15, 0xF7, 0xB7}; // 0~9
uint8_t Tab1[] = {0x5F, 0x06, 0x6B, 0x2F, 0x36, 0x3D, 0x7D, 0x07, 0x7F, 0x3F, 0x20, 0x73}; // 0~9 10:- 11:P
uint8_t Tab2[] = {0x5F, 0x06, 0x3D, 0x2F, 0x66, 0x6B, 0x7B, 0x0E, 0x7F, 0x6F}; // 0~9
uint8_t Pmode_ShowFlag, Temp_ShowFlag, Speed_ShowFlag, Time_ShowFlag; // P 模式、时间、速度、温度显示的标志位，0：常亮，1：熄灭
uint8_t PModeP1_ShowFlag, PModeP2_ShowFlag; // P1，P2 闪烁
uint8_t Icn_Falg, TimeIcn_Flag, BaseIcn_Flag; // 温度、时间、底座图标显示的标志位

/**
 * @brief 闪烁检测
 *
 * @param dT 任务周期
 */
static void Check_ShowFlag(float dT)
{
    static float T;
    if (!sys.SetMode_Option) // 如果没有在设置选项中，则都点亮，不闪烁
    {
        Pmode_ShowFlag = 0;    // 常亮
        Temp_ShowFlag = 0;     // 常亮
        Speed_ShowFlag = 0;    // 常亮
        Time_ShowFlag = 0;     // 常亮
        PModeP1_ShowFlag = 0; // 常亮
        PModeP2_ShowFlag = 0; // 常亮
        Twinkle_Time = 0;      // 闪烁计时清零
        return;
    }
    if (Twinkle_Time && !Key_Status) // 闪烁和没有操作按键时
    {
        if (T == 0)
        {
            if (Twinkle_Time == 0)
            {
                sys.SetMode_Option = 0; // 模式选择清零
                Pmode_ShowFlag = 0;      // 常亮
                Temp_ShowFlag = 0;       // 常亮
                Speed_ShowFlag = 0;      // 常亮
                Time_ShowFlag = 0;       // 常亮
                PModeP1_ShowFlag = 0;    // 常亮
                PModeP2_ShowFlag = 0;    // 常亮
            }
            if (sys.SetMode_Option == 1)
            {
```

```
        Pmode_ShowFlag = ~Pmode_ShowFlag; // P 模式闪烁
        Temp_ShowFlag = 0;                  // 常亮
        Speed_ShowFlag = 0;                 // 常亮
        Time_ShowFlag = 0;                  // 常亮
        PModeP1_ShowFlag = 0;               // 常亮
        PModeP2_ShowFlag = 0;               // 常亮
    }
    else if (sys.SetMode_Option == 2)
    {
        Pmode_ShowFlag = 0;                 // 常亮
        Temp_ShowFlag = ~Temp_ShowFlag; // 温度闪烁
        Speed_ShowFlag = 0;                 // 常亮
        Time_ShowFlag = 0;                  // 常亮
        PModeP1_ShowFlag = 0;               // 常亮
        PModeP2_ShowFlag = 0;               // 常亮
    }
    else if (sys.SetMode_Option == 3)
    {
        Pmode_ShowFlag = 0;                 // 常亮
        Temp_ShowFlag = 0;                  // 常亮
        Speed_ShowFlag = ~Speed_ShowFlag; // 速度闪烁
        Time_ShowFlag = 0;                  // 常亮
        PModeP1_ShowFlag = 0;               // 常亮
        PModeP2_ShowFlag = 0;               // 常亮
    }
    else if (sys.SetMode_Option == 4)
    {
        Pmode_ShowFlag = 0;                 // 常亮
        Temp_ShowFlag = 0;                  // 常亮
        Speed_ShowFlag = 0;                 // 常亮
        Time_ShowFlag = ~Time_ShowFlag; // 时间闪烁
        PModeP1_ShowFlag = 0;               // 常亮
        PModeP2_ShowFlag = 0;               // 常亮
    }
    else if (sys.SetMode_Option == 5)
    {
        Pmode_ShowFlag = 0;                 // 常亮
        Temp_ShowFlag = 0;                  // 常亮
        Speed_ShowFlag = 0;                 // 常亮
        Time_ShowFlag = 0;                  // 常亮
        PModeP1_ShowFlag = ~PModeP1_ShowFlag; // P1 闪烁
        PModeP2_ShowFlag = 0;               // 常亮
    }
    else if (sys.SetMode_Option == 6)
    {
        Pmode_ShowFlag = 0;                 // 常亮
        Temp_ShowFlag = 0;                  // 常亮
        Speed_ShowFlag = 0;                 // 常亮
        Time_ShowFlag = 0;                  // 常亮
        PModeP1_ShowFlag = 0;               // 常亮
```

```
                PModeP2_ShowFlag = ~PModeP2_ShowFlag; // P2 闪烁
            }
        }
        T += dT;
        if (T >= 0.5f)
        {
            Twinkle_Time -= 0.5f;
            if (Twinkle_Time <= 0)
            {
                sys.SetMode_Option = 0;
                if (PMode.Status == 2)
                {
                    PMode.Option = PMode.P1;
                    Param_Read(); //  读取参数
                }
            }
            T = 0;
        }
    }
    else
    {
        Pmode_ShowFlag = 0;     //  常亮
        Temp_ShowFlag = 0;      //  常亮
        Speed_ShowFlag = 0;     //  常亮
        Time_ShowFlag = 0;      //  常亮
        PModeP1_ShowFlag = 0; //  常亮
        PModeP2_ShowFlag = 0; //  常亮
        T = 0;
    }
}

/**
 * @brief  外框动画
 *
 * @param dT  任务周期
 */
void Circle_Duty(float dT)
{
    static float T;
    if (PMode.Status)
    {
        if ((sys.Run_Status == 1) && PMode.Status == 2) //  启动，并且在梯度模式下
        {
            if (T == 0)
            {
                PMode.Circle_Step++;
                if (PMode.Circle_Step > 17)
                    PMode.Circle_Step = 6;
                switch (PMode.Circle_Step)
                {
```

```
case 0:
    PMode.Light_BIT = 0x0FFF; //  全部点亮
    break;
case 1:
    PMode.Light_BIT = 0x0001; // L1 点亮
    break;
case 2:
    PMode.Light_BIT = 0x0003; // L1，2 点亮
    break;
case 3:
    PMode.Light_BIT = 0x0007; // L1，2，3 点亮
    break;
case 4:
    PMode.Light_BIT = 0x000F; // L1，2，3，4 点亮
    break;
case 5:
    PMode.Light_BIT = 0x001F; // L1，2，3，4，5，点亮
    break;
case 6:
    PMode.Light_BIT = 0x003F; // L1，2，3，4，5，6 点亮
    break;
case 7:
    PMode.Light_BIT = 0x007E; // L2,3,4,5,6,7 点亮
    break;
case 8:
    PMode.Light_BIT = 0x00FC; // L3,4,5,6,7,8 点亮
    break;
case 9:
    PMode.Light_BIT = 0x01F8; // L4,5,6,7,8,9 点亮
    break;
case 10:
    PMode.Light_BIT = 0x03F0; // L5,6,7,8,9,10 点亮
    break;
case 11:
    PMode.Light_BIT = 0x07E0; // L6,7,8,9,10,11,点亮
    break;
case 12:
    PMode.Light_BIT = 0x0FC0; // L7,8,9,10,11,12 点亮
    break;
case 13:
    PMode.Light_BIT = 0x0F81; // L8,9,10,11,12,1 点亮
    break;
case 14:
    PMode.Light_BIT = 0x0F03; // L9,10,11,12,1,2 点亮
    break;
case 15:
    PMode.Light_BIT = 0x0E07; // L10,11,12,1，2，3 点亮
    break;
case 16:
    PMode.Light_BIT = 0x0C0F; // L11，12，1，2，3，4 点亮
```

```
                    break;
                case 17:
                    PMode.Light_BIT = 0x081F; // L12，1，2，3，4，5 点亮
                    break;
                default:
                    break;
                }
            }
            T += dT;
            if (T >= 0.5f)
            {
                T = 0;
            }
        }
        else
        {
            PMode.Light_BIT = 0x0FFF; //  全部点亮
            T = 0;
        }
    }
    else
    {
        PMode.Light_BIT = 0x0000; //  全部熄灭
        T = 0;
    }
}

/**
 * @brief  转速图标转动任务
 *
 * @param dT  任务周期
 */
void SpeedIcn_Duty(float dT)
{
    static float T;
    if ((sys.Run_Status == 1) && Speed.Ctrl) //  启动，并且在设定了转速的情况下
    {
        if (T == 0)
        {
            Speed.IcnStep++;
            if (Speed.IcnStep > 3)
                Speed.IcnStep = 1;
        }
        T += dT;
        if (T >= 0.5f)
        {
            T = 0;
        }
    }
    else
```

```
    {
        Speed.IcnStep = 0;
        T = 0;
    }
}

/**
 * @brief  图标闪烁
 *
 * @param dT  任务周期
 */
static void Icn_Twinkle(float dT)
{
    static float T;

    if (sys.Run_Status == 1)
    {
        T += dT;
        if (T >= 0.5f)
        {
            Icn_Falg = ~Icn_Falg;
            T = 0;
        }
    }
    else
    {
        Icn_Falg = 0;
    }

    if (Temp.Base_Err)
    {
        if (!HALL1 || !HALL2)
        {
            Temp.Base_Err = 0;
            T = 0;
            BaseIcn_Flag = 0;
        }
        else
        {
            T += dT;
            if (T >= 0.5f)
            {
                BaseIcn_Flag = ~BaseIcn_Flag;
                T = 0;
            }
        }
    }
}

/**
```

```
 * @brief  梯度模式显示转换
 *
 * @param dT  任务周期
 */
static void Check_PMode_Mode(float dT)
{
    static float T;
    if (PMode.Status == 2)
    {
        if (sys.Run_Status == 1)
        {
            T += dT;
            if (T > 2.0f)
            {
                PMode.Mode = 1;
                if (T >= 4)
                    T = 0;
            }
            else
            {
                PMode.Mode = 0;
            }
        }
        else
        {
            T = 0;
            PMode.Mode = 1;
        }
    }
}

/**
 * @brief  闪烁函数
 *
 * @param dT  任务周期
 */
void Twinkle(float dT)
{
    Check_ShowFlag(dT);
    Circle_Duty(dT);
    SpeedIcn_Duty(dT);
    Icn_Twinkle(dT);
    Check_PMode_Mode(dT);
}

/**
 * @brief  显示温度
 *
 * @param dis_set_temp  设定温度
 * @param dis_rel_temp  实际温度
```

```
 */
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;
    uint8_t Temp_QU, Temp_BU, Temp_SU, Temp_GU; //  实际温度的计算位数取值
    uint8_t Temp_QD, Temp_BD, Temp_SD, Temp_GD; //  设定温度的计算位数取值

    Temp_QD = 0;
    Temp_BD = 0;
    Temp_SD = 0;
    Temp_GD = 0;
    Temp_QU = 0;
    Temp_BU = 0;
    Temp_SU = 0;
    Temp_GU = 0;

    /**********设定温度计算**********/
    if (!Temp_ShowFlag)
    {
        if (dis_set_temp > 999) //  大于 999 时
        {
            Temp_QD = Tab1[dis_set_temp / 1000];
            Temp_BD = Tab1[dis_set_temp / 100 % 10];
            Temp_SD = Tab1[dis_set_temp / 10 % 10];
            Temp_GD = Tab1[dis_set_temp % 10];
        }
        else if (dis_set_temp > 99) //  大于 99 时
        {
            Temp_QD = 0x00;
            Temp_BD = Tab1[dis_set_temp / 100 % 10];
            Temp_SD = Tab1[dis_set_temp / 10 % 10];
            Temp_GD = Tab1[dis_set_temp % 10];
        }
        else if (dis_set_temp >= 0) //  大于 99 时
        {
            Temp_QD = 0x00;
            Temp_BD = 0x00;
            Temp_SD = Tab1[ABS(dis_set_temp) / 10 % 10];
            Temp_GD = Tab1[ABS(dis_set_temp) % 10];
        }
        else if (dis_set_temp > -100) //  大于 99 时
        {
```

```
            Temp_QD = 0x00;
            Temp_BD = Tab1[10];
            Temp_SD = Tab1[ABS(dis_set_temp) / 10 % 10];
            Temp_GD = Tab1[ABS(dis_set_temp) % 10];
        }
        else if (dis_set_temp > -1000) //  大于 99 时
        {
            Temp_QD = Tab1[10];
            Temp_BD = Tab1[ABS(dis_set_temp) / 100 % 10];
            Temp_SD = Tab1[ABS(dis_set_temp) / 10 % 10];
            Temp_GD = Tab1[ABS(dis_set_temp) % 10];
        }
        seg7 &= 0x7f;
        seg7 |= 0x80; //  设定温度小数点
    }
    else
    {
        Temp_QD = 0x00;
        Temp_BD = 0x00;
        Temp_SD = 0x00;
        Temp_GD = 0x00;
        seg7 &= 0x7f;
        seg7 |= 0x00; //  设定温度小数点
    }

    /**********实际温度计算**********/
    if (dis_rel_temp > 999) //  大于 999 时
    {
        Temp_QU = Tab[dis_rel_temp / 1000];
        Temp_BU = Tab[dis_rel_temp / 100 % 10];
        Temp_SU = Tab[dis_rel_temp / 10 % 10];
        Temp_GU = Tab[dis_rel_temp % 10];
    }
    else if (dis_rel_temp > 99) //  大于 99 时
    {
        Temp_QU = 0x00;
        Temp_BU = Tab[dis_rel_temp / 100 % 10];
        Temp_SU = Tab[dis_rel_temp / 10 % 10];
        Temp_GU = Tab[dis_rel_temp % 10];
    }
    else if (dis_rel_temp >= 0) //  大于 99 时
    {
        Temp_QU = 0x00;
        Temp_BU = 0x00;
        Temp_SU = Tab[ABS(dis_rel_temp) / 10 % 10];
        Temp_GU = Tab[ABS(dis_rel_temp) % 10];
    }
    else if (dis_rel_temp > -100) //  大于 99 时
    {
        Temp_QU = 0x00;
```

```
        Temp_BU = 0x02;
        Temp_SU = Tab[ABS(dis_rel_temp) / 10 % 10];
        Temp_GU = Tab[ABS(dis_rel_temp) % 10];
    }
    else if (dis_rel_temp > -1000) //  大于 99 时
    {
        Temp_QU = 0x02;
        Temp_BU = Tab[ABS(dis_rel_temp) / 100 % 10];
        Temp_SU = Tab[ABS(dis_rel_temp) / 10 % 10];
        Temp_GU = Tab[ABS(dis_rel_temp) % 10];
    }

    /**********外框的图标**********/
    if (PMode.Light_BIT & BIT5)
    {
        seg1 &= 0x7f;
        seg1 |= 0x80; // L6 点亮
    }
    else
    {
        seg1 &= 0x7f;
        seg1 |= 0x00; // L6 熄灭
    }
    if (PMode.Light_BIT & BIT6)
    {
        seg2 &= 0x7f;
        seg2 |= 0x80; // L7 点亮
    }
    else
    {
        seg2 &= 0x7f;
        seg2 |= 0x00; // L7 熄灭
    }
    if (PMode.Light_BIT & BIT7)
    {
        seg3 &= 0x7f;
        seg3 |= 0x80; // L8 点亮
    }
    else
    {
        seg3 &= 0x7f;
        seg3 |= 0x00; // L8 熄灭
    }
    if (PMode.Light_BIT & BIT8)
    {
        seg4 &= 0x7f;
        seg4 |= 0x80; // L9 点亮
    }
    else
    {
```

```
        seg4 &= 0x7f;
        seg4 |= 0x00; // L9 熄灭
    }
    if (PMode.Light_BIT & BIT9)
    {
        seg5 &= 0x7f;
        seg5 |= 0x80; // L10 点亮
    }
    else
    {
        seg5 &= 0x7f;
        seg5 |= 0x00; // L10 熄灭
    }

    /**********温度小数点的图标**********/
    seg6 &= 0xf7;
    seg6 |= 0x08; //  实际温度小数点

    /**********℃**********/
    seg8 &= 0x7F;
    seg8 |= 0x80; //  ℃

    /**********数据拆分**********/
    seg2 &= 0xF8;
    seg2 |= (Temp_QU & 0x07);
    seg1 &= 0xF0;
    seg1 |= (Temp_QU >> 4) & 0x0F;
    seg2 &= 0x87;
    seg2 |= ((Temp_QD << 3) & 0x70) | ((Temp_QD << 3) & 0x08);
    seg1 &= 0x8F;
    seg1 |= (Temp_QD & 0x70);

    seg4 &= 0xF8;
    seg4 |= (Temp_BU & 0x07);
    seg3 &= 0xF0;
    seg3 |= (Temp_BU >> 4) & 0x0F;
    seg4 &= 0x87;
    seg4 |= ((Temp_BD << 3) & 0x70) | ((Temp_BD << 3) & 0x08);
    seg3 &= 0x8F;
    seg3 |= (Temp_BD & 0x70);

    seg6 &= 0xF8;
    seg6 |= (Temp_SU & 0x07);
    seg5 &= 0xF0;
    seg5 |= (Temp_SU >> 4) & 0x0F;
    seg6 &= 0x0F;
    seg6 |= Temp_SD << 4;
    seg5 &= 0x8F;
    seg5 |= (Temp_SD & 0x70);
```

```
    seg8 &= 0xF8;
    seg8 |= (Temp_GU & 0x07);
    seg7 &= 0xF0;
    seg7 |= (Temp_GU >> 4) & 0x0F;
    seg8 &= 0x87;
    seg8 |= ((Temp_GD << 3) & 0x70) | ((Temp_GD << 3) & 0x08);
    seg7 &= 0x8F;
    seg7 |= (Temp_GD & 0x70);

    /**********发送数据**********/
    Write_Addr_Dat_N(0, seg1, 1);
    Write_Addr_Dat_N(2, seg2, 1);
    Write_Addr_Dat_N(4, seg3, 1);
    Write_Addr_Dat_N(6, 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);
}

/**
 * @brief 显示转速
 *
 * @param dis_set_speed 设定转速
 * @param dis_rel_speed 实际转速
 */
void Display_Speed(int16_t dis_set_speed, int16_t dis_rel_speed)
{
    uint8_t seg9, seg10, seg11, seg12, seg13, seg14, seg15, seg16, seg17;
    seg9 = 0;
    seg10 = 0;
    seg11 = 0;
    seg12 = 0;
    seg13 = 0;
    seg14 = 0;
    seg15 = 0;
    seg16 = 0;
    seg17 = 0;
    uint8_t Speed_QU, Speed_BU, Speed_SU, Speed_GU; // 实际转速的计算位数取值
    uint8_t Speed_QD, Speed_BD, Speed_SD, Speed_GD; // 设定转速的计算位数取值

    if (!Speed_ShowFlag)
    {
        /**********设定转速计算**********/
        Speed_BD = Tab1[dis_set_speed / 100 % 10];
        Speed_SD = Tab1[dis_set_speed / 10 % 10];
        Speed_QD = Tab1[dis_set_speed / 1000];
        Speed_GD = Tab1[dis_set_speed % 10];
    }
    else
```

```
{
    Speed_QD = 0x00; //  不显示设定速度
    Speed_BD = 0x00; //  不显示设定速度
    Speed_SD = 0x00; //  不显示设定速度
    Speed_GD = 0x00; //  不显示设定速度
}

/**********实际转速计算**********/
Speed_QU = Tab[dis_rel_speed / 1000];
Speed_BU = Tab[dis_rel_speed / 100 % 10];
Speed_SU = Tab[dis_rel_speed / 10 % 10];
Speed_GU = Tab[dis_rel_speed % 10];

/**********加热图标**********/
if (!Temp.Icon && (sys.Run_Status == 1) && Icn_Falg)
{
    seg9 &= 0xFE;
    seg9 |= 0x01;
}
else
{
    seg9 &= 0xFE;
    seg9 |= 0x00;
}

/**********制冷图标**********/
if (Temp.Icon && (sys.Run_Status == 1) && Icn_Falg)
{
    seg9 &= 0xFD;
    seg9 |= 0x02;
}
else
{
    seg9 &= 0xFD;
    seg9 |= 0x00;
}

/**********时间图标**********/
if (sys.Run_Status == 1)
{
    if (!Icn_Falg && Temp.ADDMode == 4)
    {
        seg9 &= 0xFB;
        seg9 |= 0x00;
    }
    else
    {
        seg9 &= 0xFB;
        seg9 |= 0x04;
    }
```

```
}
else
{
    seg9 &= 0xFB;
    seg9 |= 0x04;
}

/**********上盖图标**********/
if (ADC_Val2)
{
    seg9 &= 0xF7;
    seg9 |= 0x08;
}
else
{

    seg9 &= 0xF7;
    seg9 |= 0x00;
}

seg9 &= 0xEF;
seg9 |= 0x10;
seg9 &= 0xDF;
seg9 |= 0x20;//先全量


if (!HALL1 && !HALL2)//低模块
{
    Speed.MAX = 1500;
    seg9 &= 0xEF;
    seg9 |= 0x00;
}
else if (HALL1 && HALL2)//无模块
{
    seg9 &= 0xEF;
    seg9 |= 0x00;
    seg9 &= 0xDF;
    seg9 |= 0x00;
}
else//高
{
    Speed.MAX = 800;
    if(Speed.Ctrl > Speed.MAX)
    {
        Speed.Set = Speed.MAX;
        SetOK_Flag = 1;
    }
    for (uint8_t i = 0; i <= 9; i++)
    {
        Param.P_Param[i][1] = Speed.MAX; //  速度
```

```
        }
        Save_Param_En = 1;
    }

    /**********底部模块图标**********/
    if (!BaseIcn_Flag)
    {
        seg9 &= 0xBF;
        seg9 |= 0x40;
    }
    else
    {
        seg9 &= 0xBF;
        seg9 |= 0x00;
    }

    /**********L11 图标**********/
    if (PMode.Light_BIT & BIT10)
    {
        seg9 &= 0x7F;
        seg9 |= 0x80; //  点亮
    }
    else
    {
        seg9 &= 0x7F;
        seg9 |= 0x00; //  熄灭
    }

    /**********转速图标**********/
    if (Speed.IcnStep == 2)
    {
        seg10 &= 0x7F;
        seg10 |= 0x00; // S5 图标熄灭
    }
    else
    {
        seg10 &= 0x7F;
        seg10 |= 0x80; // S5 图标
    }
    if (Speed.IcnStep == 1)
    {
        seg11 &= 0x7F;
        seg11 |= 0x00; // S3 图标熄灭
    }
    else
    {
        seg11 &= 0x7F;
        seg11 |= 0x80; // S3 图标
    }
```

```
/**********时间单位图标**********/
if (Time.Set || sys.SetMode_Option == 4) //  设定时间大于 0，在设定时间下
{
    if (Time.Set < 3600)
    {
        seg12 &= 0x7F;
        seg12 |= 0x80; // sec 图标
    }
    else
    {
        seg15 &= 0x7F;
        seg15 |= 0x80; // min 图标
    }
}
else
{
    seg12 &= 0x7F;
    seg12 |= 0x00; //  不显示 sec 图标
    seg15 &= 0x7F;
    seg15 |= 0x00; //  不显示 min 图标
}

//
//    /**********时间冒号图标**********/
//    seg13 &= 0xFE;seg13 |= 0x01;//设定转速上的冒号
//    seg14 &= 0x7F;seg14 |= 0x80;//实际转速上的冒号
//
//    /**********时间单位图标**********/
//    seg16 &= 0x7F;seg16 |= 0x80;//设定转速 min 图标
//
/**********转速单位图标**********/
seg17 &= 0x7F;
seg17 |= 0x80; //  设定转速的 rpm 单位图标

/**********数据拆分**********/
seg11 &= 0xF8;
seg11 |= (Speed_QU & 0x07);
seg10 &= 0xF0;
seg10 |= (Speed_QU >> 4) & 0x0F;
seg11 &= 0x87;
seg11 |= ((Speed_QD << 3) & 0x70) | ((Speed_QD << 3) & 0x08);
seg10 &= 0x8F;
seg10 |= (Speed_QD & 0x70);

seg13 &= 0x1F;
seg13 |= ((Speed_BU << 1) & 0x0E);
seg12 &= 0xF0;
seg12 |= (Speed_BU >> 4) & 0x0F;
seg13 &= 0x0F;
seg13 |= Speed_BD << 4;
```

```
    seg12 &= 0x8F;
    seg12 |= (Speed_BD & 0x70);

    seg15 &= 0xF8;
    seg15 |= (Speed_SU & 0x07);
    seg14 &= 0xF0;
    seg14 |= (Speed_SU >> 4) & 0x0F;
    seg15 &= 0x87;
    seg15 |= ((Speed_SD << 3) & 0x70) | ((Speed_SD << 3) & 0x08);
    seg14 &= 0x8F;
    seg14 |= (Speed_SD & 0x70);

    seg17 &= 0xF8;
    seg17 |= (Speed_GU & 0x07);
    seg16 &= 0xF0;
    seg16 |= (Speed_GU >> 4) & 0x0F;
    seg17 &= 0x87;
    seg17 |= ((Speed_GD << 3) & 0x70) | ((Speed_GD << 3) & 0x08);
    seg16 &= 0x8F;
    seg16 |= (Speed_GD & 0x70);

    /**********数据发送**********/
    Write_Addr_Dat_N(16, seg9, 1);
    Write_Addr_Dat_N(18, seg10, 1);
    Write_Addr_Dat_N(20, seg11, 1);
    Write_Addr_Dat_N(22, seg12, 1);
    Write_Addr_Dat_N(24, seg13, 1);
    Write_Addr_Dat_N(26, seg14, 1);
    Write_Addr_Dat_N(28, seg15, 1);
    Write_Addr_Dat_N(30, seg16, 1);
    Write_Addr_Dat_N(32, seg17, 1);
}

/**
 * @brief 显示时间
 *
 * @param dis_time 显示的时间
 */
void Display_Time(int32_t dis_time)
{
    uint8_t seg18, seg19, seg20, seg21, seg22, seg23, seg24, seg25;
    seg18 = 0;
    seg19 = 0;
    seg20 = 0;
    seg21 = 0;
    seg22 = 0;
    seg23 = 0;
    seg24 = 0;
    seg25 = 0;
    uint8_t Time_Q, Time_B, Time_S, Time_G; // 时间的计算位数取值
```

```
uint8_t P_B, P_S, P_G;                          // P 模式框中的位数取值
uint8_t SH, H, SM, M;                           // 时间的单位取值
P_B = 0;
P_S = 0;
P_G = 0;

if (Time.Set || sys.SetMode_Option == 4) // 设定时间大于 0，在设定时间下
{
    if (!Time_ShowFlag)
    {
        if (Time.Set) // 假如设定时间大于 0
        {
            /**********时间计算**********/
            if (Time.Set < 3600)
            {
                SH = dis_time % 3600 / 60 / 10; // 计算十位为单位的分钟数
                H = dis_time % 3600 / 60 % 10;  // 计算个位为单位的分钟数
                SM = dis_time % 60 / 10;        // 计算十分位为单位的秒钟数
                M = dis_time % 60 % 10;         // 计算十分位为单位的秒钟数
            }
            else
            {
                SH = dis_time / 3600 / 10;      // 计算十位为单位的小时数
                H = dis_time / 3600 % 10;       // 计算个位为单位的小时数
                SM = dis_time % 3600 / 60 / 10; // 计算十分位为单位的分钟数
                M = dis_time % 3600 / 60 % 10;  // 计算个分位为单位的分钟数
            }
            /**********冒号图标**********/
            seg21 &= 0xFE;
            seg21 |= 0x01;
            Time_Q = Tab2[SH];
            Time_B = Tab2[H];
            Time_S = Tab2[SM];
            Time_G = Tab2[M];
        }
        else
        {
            Time_Q = 0x20; // 显示‘-’
            Time_B = 0x20; // 显示‘-’
            Time_S = 0x20; // 显示‘-’
            Time_G = 0x20; // 显示‘-’
        }
    }
    else
    {
        Time_Q = 0x00; // 不显示
        Time_B = 0x00; // 不显示
        Time_S = 0x00; // 不显示
        Time_G = 0x00; // 不显示
    }
```

```
}
else
{
    Time_Q = 0x00; //  不显示
    Time_B = 0x00; //  不显示
    Time_S = 0x00; //  不显示
    Time_G = 0x00; //  不显示
}

if (PMode.Status) //  进入 PMode
{
    if (!PMode.Mode) //  在 P 模式下
    {
        if (!Pmode_ShowFlag)
        {
            P_B = Tab1[11];                 // P
            P_S = Tab1[10];                 // -
            P_G = Tab1[PMode.Option]; // 2
        }
        else
        {
            P_B = 0x00; //  不显示
            P_S = 0x00; //  不显示
            P_G = 0x00; //  不显示
        }
    }
    else //  梯度模式下
    {
        if (!PModeP1_ShowFlag)
            P_B = Tab1[PMode.P1];
        else
            P_B = 0x00; //  不显示

        P_S = Tab1[10]; // -

        if (!PModeP2_ShowFlag)
            P_G = Tab1[PMode.P2];
        else
            P_G = 0x00;
    }
}
else //  不进入 P 模式不显示
{
    P_B = 0x00; //  不显示
    P_S = 0x00; //  不显示
    P_G = 0x00; //  不显示
}

/**********转速图标**********/
if (Speed.IcnStep == 3)
```

```
{
    seg19 &= 0xF7;
    seg19 |= 0x00; // S4 图标熄灭
}
else
{
    seg19 &= 0xF7;
    seg19 |= 0x08; // S4 图标
}
//    seg19 &= 0x7F;seg19 |= 0x80;//S6 时间 L 图标

/**********L12 图标**********/
if (PMode.Light_BIT & BIT11)
{
    seg19 &= 0x8F;
    seg19 |= 0x40; //  点亮
}
else
{
    seg19 &= 0x8F;
    seg19 |= 0x00; //  熄灭
}

/**********L1 图标**********/
if (PMode.Light_BIT & BIT0)
{
    seg20 &= 0xEF;
    seg20 |= 0x10; //  点亮
}
else
{
    seg20 &= 0xEF;
    seg20 |= 0x00; //  熄灭
}

/**********L2 图标**********/
if (PMode.Light_BIT & BIT1)
{
    seg22 &= 0xEF;
    seg22 |= 0x10; //  点亮
}
else
{
    seg22 &= 0xEF;
    seg22 |= 0x00; //  熄灭
}
/**********L3 图标**********/
if (PMode.Light_BIT & BIT2)
{
    seg23 &= 0xF7;
```

```
        seg23 |= 0x08; //  点亮
    }
    else
    {
        seg23 &= 0xF7;
        seg23 |= 0x00; //  熄灭
    }
    /**********L4 图标**********/
    if (PMode.Light_BIT & BIT3)
    {
        seg24 &= 0xEF;
        seg24 |= 0x10; //  点亮
    }
    else
    {
        seg24 &= 0xEF;
        seg24 |= 0x00; //  熄灭
    }

    /**********L5 图标**********/
    if (PMode.Light_BIT & BIT4)
    {
        seg25 &= 0xF7;
        seg25 |= 0x08; //  点亮
    }
    else
    {
        seg25 &= 0xF7;
        seg25 |= 0x00; //  熄灭
    }
    /**********数据拆分**********/
    seg18 &= 0xF0;
    seg18 |= (Time_G & 0x0F);
    seg19 &= 0xF8;
    seg19 |= ((Time_G >> 4) & 0x07);

    seg20 &= 0xF0;
    seg20 |= (Time_S & 0x0F);
    seg21 &= 0xF1;
    seg21 |= ((Time_S >> 3) & 0x0E);
    seg20 &= 0x1F;
    seg20 |= ((P_B << 1) & 0xE0);
    seg21 &= 0x0F;
    seg21 |= ((P_B << 4) & 0xF0);

    seg22 &= 0xF0;
    seg22 |= (Time_B & 0x0F);
    seg23 &= 0xF8;
    seg23 |= ((Time_B >> 4) & 0x07);
    seg22 &= 0x1F;
```

```
    seg22 |= ((P_S << 1) & 0xE0);
    seg23 &= 0x0F;
    seg23 |= ((P_S << 4) & 0xF0);

    seg24 &= 0xF0;
    seg24 |= (Time_Q & 0x0F);
    seg25 &= 0xF8;
    seg25 |= ((Time_Q >> 4) & 0x07);
    seg24 &= 0x1F;
    seg24 |= ((P_G << 1) & 0xE0);
    seg25 &= 0x0F;
    seg25 |= ((P_G << 4) & 0xF0);

    /**********数据发送**********/
    Write_Addr_Dat_N(34, seg18, 1);
    Write_Addr_Dat_N(36, seg19, 1);
    Write_Addr_Dat_N(38, seg20, 1);
    Write_Addr_Dat_N(40, seg21, 1);
    Write_Addr_Dat_N(42, seg22, 1);
    Write_Addr_Dat_N(44, seg23, 1);
    Write_Addr_Dat_N(46, seg24, 1);
    Write_Addr_Dat_N(48, seg25, 1);
}

/**
 * @brief 速度显示处理
 *
 * @param dT 任务周期
 */
void Deal_Speed(float dT)
{
    if (sys.Run_Status == 1)
    {
        if (Speed.ADDMode == 0) // 在电机控制中，速度未处理
        {
            if (Speed.Ctrl >= Speed.Display_Rel) // 控制速度大于实际速度
            {
                Speed.ADDMode = 1; // 进入加速模式下
            }
            else if (Speed.Ctrl < Speed.Display_Rel) // 控制速度小于实际速度
            {
                Speed.ADDMode = 2; // 进入减速模式下
            }
        }
        if (Speed.ADDMode == 1) // 在进入加速模式下
        {
            if (Speed.Rel > Speed.Display_Rel) // 当前速度大于显示速度
            {
                if (Speed.Display_Rel < Speed.Rel)
                    Speed.Display_Rel += 1; // 显示当前速度
```

```
                }
                else // 当前速度小于上一次速度
                {
                    Speed.Display_Rel = Speed.Display_Rel; // 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
                }
                if (Speed.Display_Rel >= Speed.Ctrl) // 实际速度大于等于控制速度
                {
                    Speed.ADDMode = 3; // 进入稳定模式
                    return;
                }
            }
            if (Speed.ADDMode == 2) // 速度下降模式下
            {
                if (Speed.Rel < Speed.Display_Rel) // 当前速度小于上一次速度
                {
                    if (Speed.Display_Rel > Speed.Rel)
                        Speed.Display_Rel -= 1; // 显示当前速度
                }
                else // 当前速度大于上一次速度
                {
                    Speed.Display_Rel = Speed.Display_Rel; // 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
                }

                if (Speed.Display_Rel <= Speed.Ctrl) // 实际速度小于等于控制速度
                {
                    Speed.ADDMode = 3; // 进入稳定模式
                    return;
                }
            }
            else if (Speed.ADDMode == 3) // 速度稳定模式下
            {
                Speed.Display_Rel = Speed.Ctrl; // 显示控制速度
            }
        }
        else
        {
            if (Speed.Display_Rel && Speed.Display_Rel > Speed.Rel)
                Speed.Display_Rel -= 1; // 显示当前速度
        }
}

/**
 * @brief 温度显示处理
 *
 * @param dT 任务周期
 */
void Deal_Temp(float dT)
{
```

```
static float T;
if (sys.Run_Status == 0)
{
    Temp.ADDMode = 0;
    Temp.Display_Rel = Temp.Rel;
}
else if (sys.Run_Status == 1)
{
    if (Temp.ADDMode == 0)
    {
        if (Temp.Set > Temp.Display_Rel)
        {
            Temp.ADDMode = 1; //  进入加热模式
        }
        else
        {
            Temp.ADDMode = 2; //  进入制冷模式
        }
    }
    else if (Temp.ADDMode == 1)
    {
        if (Temp.Rel > Temp.Display_Rel && Temp.Display_Rel <= Temp.Ctrl)
        {
            Temp.Display_Rel++;
            T = 0;
        }
        if (Temp.Display_Rel >= Temp.Set - 20)
        {
            Temp.ADDMode = 3;
            return;
        }
    }
    else if (Temp.ADDMode == 2)
    {
        if (Temp.Rel < Temp.Display_Rel && Temp.Display_Rel >= Temp.Ctrl)
        {
            Temp.Display_Rel--;
            T = 0;
        }
        if (Temp.Display_Rel <= Temp.Set + 20)
        {
            Temp.ADDMode = 3;
            return;
        }
    }
    else if (Temp.ADDMode == 3)
    {
        T += dT;
        if (Temp.Display_Rel < Temp.Set)
        {
```

```
                if (T >= 6.0f)
                {
                    Temp.Display_Rel += 1;
                    T = 0;
                }
            }
            else if (Temp.Display_Rel > Temp.Set)
            {
                if (T >= 10.0f)
                {
                    Temp.Display_Rel -= 1;
                    T = 0;
                }
            }
            else
            {
                Temp.ADDMode = 4;
                T = 0;
            }
        }
        else if (Temp.ADDMode == 4)
        {
            Temp.Display_Rel = Temp.Ctrl;
        }
    }
    else if (sys.Run_Status == 2)
    {
        if (Temp.Display_Rel < Temp.Rel)
        {
            T += dT;
            if (T >= 10.0f)
            {
                Temp.Display_Rel += 1;
                T = 0;
            }
        }
        else if (Temp.Display_Rel > Temp.Rel)
        {
            T += dT;
            if (T >= 10.0f)
            {
                Temp.Display_Rel -= 1;
                T = 0;
            }
        }
        else
        {
            sys.Run_Status = 0;
            T = 0;
        }
```

```
    }
}

/**
 * @brief 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 == 1)
    {
        Time.Display = Time.Rel;
    }
    else
    {
        Time.Display = Time.Set; // 显示设定时间
    }
    Display_Temp(Temp.Display_Set, Temp.Display_Rel);
    Display_Speed(Speed.Display_Set, Speed.Display_Rel);
    Display_Time(Time.Display);
}
#include "Speed.h"

/**
 * @brief 编码器初始化
 *
 */
void Encoder_Init(void)
{
    HAL_TIM_Base_Start_IT(&htim14);
    HAL_TIM_IC_Start_IT(&htim14, TIM_CHANNEL_1); // 开启 time14 通道 1 输入捕获
}

/**
 * @brief 检测速度是否停止-0.05s
 *
 * @param dT 任务周期
 */
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
    }
```

```
}

uint32_t capture, capture1, capture2;
float rel1;
/**
 * @brief Tim14 通道 1 的输入捕获回调函数
 *
 */
void TIM14CaptureChannel1Callback(void)
{
    capture1 = __HAL_TIM_GET_COMPARE(&htim14, TIM_CHANNEL_1); // 获取 Tim14 通道 1 的输入捕获
    if (capture1 > capture2)
        capture = capture1 - capture2;
    else
        capture = capture1 + (0xFFFF - capture2);
    rel1 = 10000.0f / (float)capture; // 计算速度
    capture2 = capture1;
    Speed.Rel = rel1 * 60 / 5; // 将速度赋值给 L1 的实际速度
    Speed.Stop_Cnt = 0;
}

/**
 * @brief TIM_IC 回调函数
 *
 * @param htim
 */
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == TIM14)
    {
        if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
        {
            TIM14CaptureChannel1Callback();
        }
    }
}
#include "Ctrl_ControlTemp.h"

/**********结构体**********/
_PID_Arg_ Temp_Arg;
_PID_Val_ Temp_Val;
_PID_Val_ HEATGAITemp_Val;

/**********局部变量**********/
uint16_t CtrlTempADC_Val;

/**
 * @brief  温度控制 PID 系数
 *
```

```
 */
void Temp_PID(void)
{
    Temp_Arg.Kp = 2;
    Temp_Arg.Ki = 0.04;
    Temp_Arg.Kd = 0 * 0.001f;
}

/**
 * @brief 温度控制
 *
 * @param dT 任务周期
 */
uint8_t Out_Enable;
void Temp_Control(float dT)
{
    if (sys.Run_Status == 1)
    {
        CtrlTempADC_Val = Get_ADCVal(Temp.Set);
        if((CtrlTempADC_Val - ADC_Val1 > -100 && CtrlTempADC_Val - ADC_Val1 < 100))
            Out_Enable = 1;
        else
            Out_Enable = 0;
        AltPID_Calculation(dT, CtrlTempADC_Val, ADC_Val1, &Temp_Arg, &Temp_Val, 100,
Out_Enable * 989); // 加热制冷控制

        if(Temp_Val.Out< -200)
            Temp_Val.Out = -200;

        HEAT_Duty(dT, (int)Temp_Val.Out);

//          if (ADC_Val2)
//          {
//                  AltPID_Calculation(dT, CtrlTempADC_Val, ADC_Val2, &Temp_Arg,
&HEATGAITemp_Val, 199, 199); // 热盖加热控制
//              HEATGAI = HEATGAITemp_Val.Out;
//          }
//          else
//          {
//              HEATGAI = 0;
//          }
    }
    else
    {
        Temp_Val.Out = 0;
        COLD = 0;
        HEAT = 0;
        HEATGAI = 0;
    }
}
```

```
#include "Ctrl_DownTime.h"

/**
 * @brief 时间倒计时检测
 *
 * @param dT 任务周期
 */
void Cheak_TimeDown(float dT)
{
    static float T;

    if (sys.Run_Status == 1) // 启动系统
    {

        if (Time.Rel > 0 && Temp.ADDMode == 4)
        {
            T += dT;
        }
        if (T >= 1.0f) // 1S
        {
            if (Time.Rel)
                Time.Rel--; // 控制时间--

            if (Time.Rel == 0)
            {
                if (PMode.Status == 2) // 梯度模式下
                {
                    PMode.Option++;
                    if (PMode.Option <= PMode.P2)
                    {
                        Param_Read(); // 读取参数
                        sys.SetMode_Option = 0;
                        Speed.ADDMode = 0;
                        Temp.ADDMode = 0;
                        if (Temp.Set >= Temp.Rel)
                        {
                            Temp.Icon = 0; // 加热图标
                            Temp.Mode = 0; // 加热
                            WIND_OFF;
                        }
                        else
                        {
                            Temp.Icon = 1; // 制冷图标
                            Temp.Mode = 1; // 制冷
                            WIND_ON;
                        }
                    }
                    else
                    {
                        PMode.Option = PMode.P1;
```

```
                                Param_Read(); //  读取参数
                                sys.Run_Status = 2;
                                sys.SetMode_Option = 0;
                                Speed.ADDMode = 2;
                                Temp.ADDMode = 0;
                                SetOK_Flag = 1;
                                WIND_OFF;
                                PMode.Circle_Step = 0;
                                Beep_Flash = 5; //  响 5 下
                            }
                        }
                        else
                        {
                            sys.Run_Status = 2;
                            sys.SetMode_Option = 0;
                            Speed.ADDMode = 2;
                            Temp.ADDMode = 0;
                            SetOK_Flag = 1;
                            WIND_OFF;
                            PMode.Circle_Step = 0;
                            Beep_Flash = 5; //  响 5 下
                        }
                        T = 0; //  周期清零
                    }
                    T = 0;
                }
            }
        }
}
#include "Ctrl_Motor.h"

/**********结构体*************/
_PID_Arg_ Speed_Arg;
_PID_Val_ Speed_Val;

/**
  * @brief  电机控制 PID 系数
  *
  */
void Motor_PID(void)
{
    Speed_Arg.Kp = 40 * 0.001f;
    Speed_Arg.Ki = 60 * 0.001f;
    Speed_Arg.Kd = 0 * 0.001f;
}

/**
  * @brief  电机控制
  *
  * @param dT  任务周期
  */
```

```
void Motor_Ctrl(float dT)
{
    if (sys.Run_Status == 1) // 启动
    {
        if (Speed.Ctrl) // 速度大于 0 和定时器没有结束
        {
            AltPID_Calculation(dT, Speed.Ctrl, Speed.Rel, &Speed_Arg, &Speed_Val, 999,
999); // 电机 PID 控制
            if (Speed_Val.Out < 0)
                Speed_Val.Out = 0;
            PWM = Speed_Val.Out; // PID 输出
        }
        else
        {
            PWM = 0; // PID 输出
        }
    }
    else
    {
        if (Speed_Val.Out)
            Speed_Val.Out -= 1;
        if (Speed_Val.Out < 0)
            Speed_Val.Out = 0;
        PWM = Speed_Val.Out; // PWM 输出
    }
}
#include "Ctrl_Scheduler.h"

/**********局部变量声明*******/
uint16_t T_cnt_2ms = 0,
          T_cnt_10ms = 0,
          T_cnt_50ms = 0,
          T_cnt_100ms = 0,
          T_cnt_200ms = 0,
          T_cnt_500ms = 0,
          T_cnt_1S = 0;

/**
 * @brief 时间检测
 *
 */
void Loop_Check(void)
{
    T_cnt_2ms++;
    T_cnt_10ms++;
    T_cnt_50ms++;
    T_cnt_100ms++;
    T_cnt_200ms++;
    T_cnt_500ms++;
    T_cnt_1S++;
```

```
    Sys_Loop();
}

/**
 * @brief 2ms 执行一次
 *
 * @param dT  任务周期
 */
static void Loop_2ms(float dT)
{
    Deal_Speed(dT);
}

/**
 * @brief 10ms 执行一次
 *
 * @param dT  任务周期
 */
static void Loop_10ms(float dT)
{
    Key_Scan(dT);
    Read_Temp(dT);
    Check_HALL(dT);
    Check_Base(dT);
}

/**
 * @brief 50ms 执行一次
 *
 * @param dT  任务周期
 */
static void Loop_50ms(float dT)
{
    Motor_Ctrl(dT);
    Buzzer_Status(dT);
    Check_Speed(dT);
    Check_Set(dT);
}

/**
 * @brief 100ms 执行一次
 *
 * @param dT  任务周期
 */
static void Loop_100ms(float dT)
{
    Cheak_TimeDown(dT);
    Twinkle(dT);
    Param_Save_Overtime(dT);
```

```
}

/**
 * @brief 200ms 执行一次
 *
 * @param dT  任务周期
 */
static void Loop_200ms(float dT)
{

}

/**
 * @brief 500ms 执行一次
 *
 * @param dT  任务周期
 */
static void Loop_500ms(float dT)
{
    Deal_Temp(dT);
    Check_Press(dT);
}

/**
 * @brief 1S 执行一次
 *
 * @param dT  任务周期
 */
static void Loop_1S(float dT)
{
    Temp_Control(dT);
}


/**
 * @brief  任务调度
 *
 */
void Sys_Loop(void)
{
    if (T_cnt_2ms >= 2)
    {
        Loop_2ms(0.002f);
        T_cnt_2ms = 0;
    }
    if (T_cnt_10ms >= 10)
    {
        Loop_10ms(0.01f);
        T_cnt_10ms = 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 >= 10)
    {
        Loop_200ms(0.01f);
        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;
    }
}
```