MS3100 软件源程序

```
#include "Show.h"

/**********全局变量声明******/
float Speed_Twinkle_Time,Time_Twinkle_Time;//速度闪烁时间和时间闪烁时间

/**********局部变量声明******/
uint8_t Tab[] = {0xFA,0x0A,0xD6,0x9E,0x2E,0xBC,0xFC,0x1A,0xFE,0xBE};//0·9
uint8_t Speed_ShowFlag,Time_ShowFlag;//速度、时间显示的标志位 0:常亮 1：熄灭
uint8_t SpeedIcn_ShowFlag,TimeIcn_ShowFlag;//速度图标闪烁和时间图标闪烁

/*
**********************************************************************
 * 函数原型：static void Icn_Twinkle(float dT)
 * 功      能：图标闪烁
 * 输      入: dT:执行周期
 * 参      数：float dT
 * 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)
            {
                if(Speed.CwIcn)
                {
                    if(SpeedIcn_ShowFlag > 0)
                        SpeedIcn_ShowFlag --;//速度图标闪烁;
                    if(SpeedIcn_ShowFlag < 1)
                        SpeedIcn_ShowFlag = 3;
                }
                else
                {
                    SpeedIcn_ShowFlag ++;//速度图标闪烁;
                    if(SpeedIcn_ShowFlag > 3)
                        SpeedIcn_ShowFlag = 1;
                }
            }
            else
                SpeedIcn_ShowFlag = 0;

            if(Time.Rel)
                TimeIcn_ShowFlag = ~TimeIcn_ShowFlag;//时间图标闪烁;
            else
            {
```

```
                    TimeIcn_ShowFlag = 0;
                }
                T = 0;
            }
        }
        else
        {
            SpeedIcn_ShowFlag = 0;//不显示速度图标
            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 Speed_T,Time_T;
    if(Speed_Twinkle_Time && !EC11A[1].EC11A_Knob)//速度闪烁和没有操作按键时
    {
        Speed_T += dT;
        if(Speed_T >= 0.5f)
        {
            Speed_Twinkle_Time -= 0.5;//闪烁计时
            Speed_ShowFlag = ~Speed_ShowFlag;//速度闪烁
            if(Speed_Twinkle_Time == 0)//如果闪烁结束
            {
                SpeedSet_Flag = 1;//进入速度设定
                Beep_Time = 0.1;//蜂鸣器响 0.1S
            }
            Speed_T = 0;
        }
    }
    else
    {
        Speed_ShowFlag = 0;//常亮
        Speed_T = 0;
    }

    if(Time_Twinkle_Time && !EC11A[0].EC11A_Knob)//时间闪烁和没有操作按键时
    {
        Time_T += dT;
        if(Time_T >= 0.5f)
```

```
        {
            Time_Twinkle_Time -= 0.5;//闪烁计时
            Time_ShowFlag = ~Time_ShowFlag;//时间闪烁
            if(Time_Twinkle_Time == 0)//如果闪烁结束
            {
                TimeSet_Flag = 1;//进入时间设定
                Beep_Time = 0.1;//蜂鸣器响 0.1S
            }
            Time_T = 0;
        }
    }
    else
    {
        Time_ShowFlag = 0;//常亮
        Time_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_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 seg10,seg11,seg12,seg13,seg14,seg15,seg16,seg17,seg18,seg19;
    seg10=0;seg11=0;seg12=0;seg13=0;seg14=0;seg15=0;seg16=0;seg17=0;seg18=0;seg19=0;
    uint8_t Speed_QU,Speed_BU,Speed_SU,Speed_GU;//实际速度的计算位数取值
    uint8_t Speed_QD,Speed_BD,Speed_SD,Speed_GD;//设定速度的计算位数取值
    uint16_t Val;//用于百十个取出来的数字
    if(Speed_ShowFlag == 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
            Speed_QD = Tab[Val];
        }
        else
        {
            Speed_QD = Tab[0];//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
            Speed_BD = Tab[Val];
        }
        else
        {
            Speed_BD = Tab[0];//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
            Speed_SD = Tab[Val];
        }
        else
        {
            Speed_SD = Tab[0];//显示 0
        }
        Val=dis_set_speed%10;//取出个位
        Speed_GD = Tab[Val];
    }
    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**********/
    if(dis_rel_speed > 999)//大于 999 时
    {
        Val=dis_rel_speed/1000;//取出千位
        Speed_QU = Tab[Val];
    }
    else
    {
        Speed_QU = Tab[0];//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
    Speed_BU = Tab[Val];
}
else
{
    Speed_BU = Tab[0];//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
    Speed_SU = Tab[Val];
}
else
{
    Speed_SU = Tab[0];//显示 0
}
Val=dis_rel_speed%10;//取出个位
Speed_GU = Tab[Val];


/***************rpm******************/
seg15 &= 0x7F;seg15 |= 0x80;//rpm

/**************转速图标***************/
switch(SpeedIcn_ShowFlag)
{
    case 0: seg10 &= 0xFE;seg10 |= 0x01;//S5
            seg10 &= 0xFD;seg10 |= 0x02;//S4
            seg11 &= 0xFD;seg11 |= 0x02;//S3
        break;
    case 1: seg10 &= 0xFE;seg10 |= 0x00;//S5
            seg10 &= 0xFD;seg10 |= 0x02;//S4
            seg11 &= 0xFD;seg11 |= 0x02;//S3
        break;
    case 2: seg10 &= 0xFE;seg10 |= 0x01;//S5
            seg10 &= 0xFD;seg10 |= 0x02;//S4
            seg11 &= 0xFD;seg11 |= 0x00;//S3
        break;
    case 3: seg10 &= 0xFE;seg10 |= 0x01;//S5
            seg10 &= 0xFD;seg10 |= 0x00;//S4
            seg11 &= 0xFD;seg11 |= 0x02;//S3
        break;
    default:
        break;
```

```
    }

    /*************数据拆分***************/
    seg19 &= 0xF0;seg19 |= Speed_QU>>4;
    seg18 &= 0xF1;seg18 |= Speed_QU & 0x0E;
    seg19 &= 0x0F;seg19 |= Speed_QD & 0xF0;
    seg18 &= 0x8F;seg18 |= (Speed_QD & 0x0F) << 3;

    seg12 &= 0xF0;seg12 |= Speed_BU>>4;
    seg13 &= 0xF1;seg13 |= Speed_BU & 0x0E;
    seg12 &= 0x0F;seg12 |= Speed_BD & 0xF0;
    seg13 &= 0x8F;seg13 |= (Speed_BD & 0x0F) << 3;

    seg14 &= 0xF0;seg14 |= Speed_SU>>4;
    seg15 &= 0xF1;seg15 |= Speed_SU & 0x0E;
    seg14 &= 0x0F;seg14 |= Speed_SD & 0xF0;
    seg15 &= 0x8F;seg15 |= (Speed_SD & 0x0F) << 3;

    seg16 &= 0xF0;seg16 |= Speed_GU>>4;
    seg17 &= 0xF1;seg17 |= Speed_GU & 0x0E;
    seg16 &= 0x0F;seg16 |= Speed_GD & 0xF0;
    seg17 &= 0x8F;seg17 |= (Speed_GD & 0x0F) << 3;

    /*************发送数据***************/
    Write_Addr_Dat_N(18, seg10,1);//SEG16
    Write_Addr_Dat_N(20, seg11,1);//SEG15
    Write_Addr_Dat_N(22, seg12,1);//SEG16
    Write_Addr_Dat_N(24, seg13,1);//SEG15
    Write_Addr_Dat_N(26, seg14,1);//SEG14
    Write_Addr_Dat_N(28, seg15,1);//SEG13
    Write_Addr_Dat_N(30, seg16,1);//SEG12
    Write_Addr_Dat_N(32, seg17,1);//SEG11
    Write_Addr_Dat_N(34, seg18,1);//SEG10
    Write_Addr_Dat_N(36, seg19,1);//SEG9
}

/*
*********************************************************************
 * 函数原型：void Display_Temp(int32_t dis_set_temp,int32_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：int32_t dis_set_temp,int32_t dis_rel_temp
*********************************************************************
*/
void Display_Time(int32_t dis_set_time,int32_t dis_rel_time)
{
    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 Time_QU,Time_BU,Time_SU,Time_GU;//实际温度的计算位数取值
    uint8_t Time_QD,Time_BD,Time_SD,Time_GD;//设定温度的计算位数取值
```

```
uint8_t SH,H,SM,M;//时间的单位取值

/***********设定温度计算**********/
if(Time_ShowFlag == 0)
{
    if(dis_set_time > 0)
    {
        SH = dis_set_time/3600/10;//计算十位单位的小时数
        H = dis_set_time/3600%10;//计算个位单位的小时数
        SM = dis_set_time%3600/60/10;//计算十分位单位的分钟数
        M = dis_set_time%3600/60%10;//计算个分位单位的分钟数
        Time_QD = Tab[SH];
        Time_BD = Tab[H];
        Time_SD = Tab[SM];
        Time_GD = Tab[M];
        /***********实际时间冒号**********/
        seg4 &= 0x7F;seg4 |= 0x80;
    }
    else
    {
        Time_QD = 0x04;//显示"-"
        Time_BD = 0x04;//显示"-"
        Time_SD = 0x04;//显示"-"
        Time_GD = 0x04;//显示"-"
        /***********实际时间冒号**********/
        seg4 &= 0x7F;seg4 |= 0x00;
    }
}
else
{
    Time_QD = 0x00;//显示"-"
    Time_BD = 0x00;//显示"-"
    Time_SD = 0x00;//显示"-"
    Time_GD = 0x00;//显示"-"
    seg4 &= 0x7F;seg4 |= 0x00;
}

/***********实际温度计算**********/
if(dis_set_time > 0)
{
    SH = dis_rel_time/3600/10;//计算十位单位的小时数
    H = dis_rel_time/3600%10;//计算个位单位的小时数
    SM = dis_rel_time%3600/60/10;//计算十分位单位的分钟数
    M = dis_rel_time%3600/60%10;//计算个分位单位的分钟数
    Time_QU = Tab[SH];
    Time_BU = Tab[H];
    Time_SU = Tab[SM];
    Time_GU = Tab[M];

    /***********实际时间冒号**********/
```

```
        if(!TimeIcn_ShowFlag)
        {
            seg4 &= 0xFE;seg4 |= 0x01;
        }
        else
        {
            seg4 &= 0xFE;seg4 |= 0x00;
        }
    }
    else
    {
        Time_QU = 0x04;//显示"-"
        Time_BU = 0x04;//显示"-"
        Time_SU = 0x04;//显示"-"
        Time_GU = 0x04;//显示"-"
        /***********实际时间冒号**********/
        seg4 &= 0xFE;seg4 |= 0x00;
    }

    /**************数据拆分***************/
    seg1 &= 0xF0;seg1 |= Time_QU>>4;
    seg2 &= 0xF1;seg2 |= Time_QU & 0x0E;
    seg1 &= 0x0F;seg1 |= Time_QD & 0xF0;
    seg2 &= 0x8F;seg2 |= (Time_QD & 0x0F) << 3;

    seg3 &= 0xF0;seg3 |= Time_BU>>4;
    seg4 &= 0xF1;seg4 |= Time_BU & 0x0E;
    seg3 &= 0x0F;seg3 |= Time_BD & 0xF0;
    seg4 &= 0x8F;seg4 |= (Time_BD & 0x0F) << 3;

    seg5 &= 0xF0;seg5 |= Time_SU>>4;
    seg6 &= 0xF1;seg6 |= Time_SU & 0x0E;
    seg5 &= 0x0F;seg5 |= Time_SD & 0xF0;
    seg6 &= 0x8F;seg6 |= (Time_SD & 0x0F) << 3;

    seg7 &= 0xF0;seg7 |= Time_GU>>4;
    seg8 &= 0xF1;seg8 |= Time_GU & 0x0E;
    seg7 &= 0x0F;seg7 |= Time_GD & 0xF0;
    seg8 &= 0x8F;seg8 |= (Time_GD & 0x0F) << 3;

    /**************发送数据***************/
    Write_Addr_Dat_N(0, seg1,1);//SEG27
    Write_Addr_Dat_N(2, seg2,1);//SEG26
    Write_Addr_Dat_N(4, seg3,1);//SEG25
    Write_Addr_Dat_N(6, seg4,1);//SEG24
    Write_Addr_Dat_N(8, seg5,1);//SEG23
    Write_Addr_Dat_N(10, seg6,1);//SEG22
    Write_Addr_Dat_N(12, seg7,1);//SEG21
    Write_Addr_Dat_N(14, seg8,1);//SEG20
}
```

```
/*
**********************************************************************
 * 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)//启动的情况下
    {
        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
        {
            Speed.New = Speed.Rel;//记录当前速度
            if(Speed.New > Speed.Display_Rel)//当前速度大于显示速度
            {
                if(Speed.Display_Rel < Speed.New)
                    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(sys.Motor_Stop == 0)
            {
                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
        {
            Speed.New = Speed.Rel;//记录当前速度

            if(Speed.New < Speed.Display_Rel)//当前速度小于上一次速度
            {
```

```
                    if(Speed.Display_Rel > Speed.New)
                        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(sys.Motor_Stop == 0)
                {
                    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
        {
            Speed.Display_Rel = 0;//实际速度显示为零
            Speed.New =0;//现在的速度清零
            Speed.ADDMode = 0;//清除显示处理
        }
}

/*
**********************************************************************
 * 函数原型：  void Show_Display(void)
 * 功      能：  显示屏幕内容
**********************************************************************
*/
void Show_Display(void)
{
    Speed.Display_Set = Speed.Set;//显示设定转速
    Time.Display_Rel = Time.Rel + 59;//显示控制时间
    Time.Display_Set = Time.Set;//显示设定时间

    Display_Speed(Speed.Display_Set, Speed.Display_Rel);//显示转速
    Display_Time(Time.Display_Set, Time.Display_Rel);//显示时间
}
#include "Param.h"

/**********宏定义**********/
struct _Save_Param_ Param;//原始数据
```

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

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

    Param.Speed = 100;//转速 100
    Param.Time = 0;//时间常动

    Param.Flash_Check_End   = FLASH_CHECK_END;
}

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

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

    //板子从未初始化
    if(Param.Flash_Check_Start != FLASH_CHECK_START || Param.Flash_Check_End != FLASH_CHECK_END)
    {
        Param_Reset();
        Speed.Set = Param.Speed;//将 Flash 中的速度赋值
        Time.Set = Param.Time;//将 Flash 中的时间赋值
        SpeedSet_Flag=TimeSet_Flag=1;//进入设置
        Save_Param_En = 1;
    }
```

```
    else
    {
        Speed.Set = Param.Speed;//将 Flash 中的速度赋值
        Time.Set = Param.Time;//将 Flash 中的时间赋值
        SpeedSet_Flag=TimeSet_Flag=1;//进入设置
    }

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

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

    if(Save_Param_En)
    {
        time += dT;

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

/*
**********************************************************************
 * 函数原型：void AltPID_Calculation(float dT, float Expect, float Freedback, _PID_Arg_ *
PID_Arg, _PID_Val_ * PID_Val, float Error_Lim, float Integral_Lim)
 * 功    能：微分先行 PID 计算
 * 输    入：dT：周期（单位：秒）
             Expect：期望值（设定值）
             Freedback：反馈值
             _PID_Arg_ * PID_Arg：PID 参数结构体
             _PID_Val_ * PID_Val：PID 数据结构体
```

```
                Error_Lim：误差限幅
                Integral_Lim：积分误差限幅
 * 参      数：float dT, float Expect, float Freedback, _PID_Arg_ * PID_Arg, _PID_Val_ * PID_Val, float Error_Lim, float Integral_Lim
**********************************************************************
*/
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, 0, 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
}

/*
**********************************************************************
 * 函数原型：void IncPID_Calculation(float dT,float Expect,float Freedback,_PID_Arg_ * PID_Arg,_PID_Val_ * PID_Val,float Integral_Lim)
 * 功      能：增量式 PID 计算
 * 输      入：dT：周期（单位：秒）
               Expect：期望值（设定值）
               Freedback：反馈值
               _PID_Arg_ * PID_Arg：PID 参数结构体
               _PID_Val_ * PID_Val：PID 数据结构体
 * 参      数：float dT,float Expect,float Freedback,_PID_Arg_ * PID_Arg,_PID_Val_ * PID_Val
**********************************************************************
*/
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 SpeedSet_Flag,TimeSet_Flag;//速度设置，时间设置

/*
*********************************************************************
 * 函数原型：void Check_Set(float dT)
 * 功      能：检测设置
*********************************************************************
*/
void Check_Set(float dT)
{
    if(SpeedSet_Flag)//速度设置
    {
        if(Speed.Ctrl != Speed.Set)//控制速度和设定速度不同时
        {
            Speed.Ctrl = Speed.Set;
            Param.Speed = Speed.Set;
            if(Speed.ADDMode != 0)//速度显示模式不等于零时
                Speed.ADDMode = 0;//速度显示模式清零
            Save_Param_En = 1;//保存
        }
        SpeedSet_Flag = 0;//设置标志位清零
    }

    if(TimeSet_Flag)//时间设置
    {
        if(Time.Rel != Time.Set)//实际时间和设定时间不同时
        {
            Time.Rel = Time.Set;
            Param.Time = Time.Set;
            Save_Param_En = 1;//保存
        }
        TimeSet_Flag = 0;//设置标志位清零
    }
}
#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();
        }
    }
}
#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;

/*
**************************************************************
 * 函数原型：void Loop_Check(void)
 * 功      能：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();
}

static void Loop_2ms(float dT)//2ms 执行一次
{
    Deal_Speed(dT);//速度显示处理
}
```

```
static void Loop_10ms(float dT)//10ms 执行一次
{
    EC11AKey_Scan(dT);//EC11A 按键扫描
    Check_Set(dT);//检测设置
}

static void Loop_50ms(float dT)//50ms 执行一次
{
    Motor_Ctrl(dT);//电机控制
    Check_Speed(dT);//检测速度是否停止
    Buzzer_Status(dT);//蜂鸣器的状态检测
}

static void Loop_100ms(float dT)//100ms 执行一次
{
    Cheak_TimeDown(dT);//时间倒计时检测
    Twinkle(dT);//闪烁函数
    Check_MotorStop(dT);//检测电机是否停止
}

static void Loop_200ms(float dT)//200ms 执行一次
{

}

static void Loop_500ms(float dT)//500ms 执行一次
{
    Check_Press(dT);//检测按键按下状态
    Param_Save_Overtime(dT);//保存标志位置 1，0.5s 后保存
}

static void Loop_1S(float dT)//1S 执行一次
{
    EC11A_Speed(dT);//EC11A 旋钮速度计算
}

/*
**********************************************************************
 * 函数原型：void Sys_Loop(void)
 * 功      能：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;
        }
}

#include "Ctrl_Motor.h"

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

/*
*********************************************************************
 * 函数原型：void Motor_PID(void)
 * 功      能：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;
}

/*
*********************************************************************
 * 函数原型：void Motor_Ctrl(float dT)
 * 功      能：电机控制
*********************************************************************
*/
void Motor_Ctrl(float dT)
```

```
{
    if(sys.Run_Status)//启动
    {
        if(Speed.Ctrl)
        {
            if(sys.Motor_Stop)
            {
                if(Speed_Val.Out)
                    Speed_Val.Out -= 1;//降速太慢*2
                if(Speed_Val.Out < 0)
                    Speed_Val.Out = 0;
                PWM = Speed_Val.Out;//pid 输出
            }
            else
            {

    AltPID_Calculation(dT,Speed.Ctrl,Speed.Rel,&Speed_Arg,&Speed_Val,400,400);//电机 PID
控制
                PWM = Speed_Val.Out;//pid 输出
            }
        }
    }
    else
    {
        Speed_Val.Out = 0;
        PWM = Speed_Val.Out;//pwm 不输出
    }
}

/*
*****************************************************************
 * 函数原型：void Check_MotorStop(float dT)
 * 功      能：检测电机是否停止，停止后开盖
*****************************************************************
*/
float CW_Time;
void Check_MotorStop(float dT)
{
    static float T;
    if(sys.Motor_Stop)
    {
        if(Speed.Cw)//进入改变转向
        {
            if(sys.Run_Status)
            {
                if(Speed.Rel <= 100)
                {
                    T += dT;
                    CW_Time = 2.0f;
                    if(T > CW_Time)
```

```
                    {
                        if(Speed.CwIcn)
                        {
                            HAL_GPIO_WritePin(CW_GPIO_Port,          CW_Pin,
GPIO_PIN_RESET);//逆时针
                            Speed.ADDMode = 0;
                            Speed.CwIcn = 0;
                        }
                        else
                        {
                            HAL_GPIO_WritePin(CW_GPIO_Port,          CW_Pin,
GPIO_PIN_SET);//顺时针
                            Speed.ADDMode = 0;
                            Speed.CwIcn = 1;
                        }
                        Beep_Time = 0.1f;
                        sys.Motor_Stop = 0;//电机停止清零
                        Speed.Cw = 0;//改变转向清零
                        Speed_Val.Integral = 27;//电机起步的 PWM
                        T = 0;
                    }
                }
            }
            else
            {
                if(Speed.CwIcn)
                {
                    HAL_GPIO_WritePin(CW_GPIO_Port, CW_Pin, GPIO_PIN_RESET);//
逆时针
                    Speed.ADDMode = 0;
                    Speed.CwIcn = 0;
                }
                else
                {
                    HAL_GPIO_WritePin(CW_GPIO_Port, CW_Pin, GPIO_PIN_SET);//顺
时针
                    Speed.ADDMode = 0;
                    Speed.CwIcn = 1;
                }
                sys.Motor_Stop = 0;//电机停止清零
                Speed.Cw = 0;//改变转向清零
            }
        }
        else
        {
            if(Speed.Rel <= 100)
            {
                T += dT;
                if(T > 1.0f)
                {
```

```
                        TimeSet_Flag=1;//设置变量
                        sys.Run_Status = 0;//关闭
                        sys.Motor_Stop = 0;//电机已经停止
                        T = 0;
                    }
                }
            }
        }
}
#include "Ctrl_DownTime.h"

/*
**********************************************************************
 * 函数原型：void Cheak_TimeDown(float dT)
 * 功      能：时间倒计时检测
 * 输      入: dT:执行周期
 * 参      数：float dT
**********************************************************************
*/
void Cheak_TimeDown(float dT)
{
    static float T;
    if(sys.Run_Status && sys.Motor_Stop == 0)//启动系统
    {
        if(Time.Rel > 0)
        {
            T += dT;
            if(T >= 1.0f)//1S
            {
                if(Time.Rel)
                    Time.Rel--;//控制时间--
                if(Time.Rel == 0)
                {
                    Speed.ADDMode = 2;//进入降速显示
                    Speed.Cw = 0;//转向清零
                    sys.Motor_Stop = 1;//检测电机
                    SpeedSet_Flag=TimeSet_Flag=1;//进入设置
                    Speed_Twinkle_Time = Time_Twinkle_Time = 0;//关闭闪烁
                    Beep_Flash = 5;//响 5 下
                }
                 T = 0;//周期清零
            }
        }
    }
}
#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, 75);//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_EC11A.h"
```

```
/**********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 = EC1A_Pin;//EC11A 旋钮中断引脚
    EC11A[0].EC11A_Pin = EC1B_Pin;//EC11A 旋钮输入引脚
    EC11A[0].EC11A_GPIO = EC1B_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 = EC2A_Pin;//EC11A 旋钮中断引脚
    EC11A[1].EC11A_Pin = EC2B_Pin;//EC11A 旋钮输入引脚
    EC11A[1].EC11A_GPIO = EC2B_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(EC11A[0].EC11A_Knob)//旋钮 0 旋转
        EC11A[0].EC11A_Knob -= dT;//倒计时

    if(EC11A[1].EC11A_Knob)//旋钮 1 旋转
        EC11A[1].EC11A_Knob -= 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(!Speed.Cw)//改变转向的话
                {
                    Speed.ADDMode = 2;//进去减速显示
                    sys.Motor_Stop = 1;//电机停止
```

```
                    Speed.Cw = 1;//进入改变转向
                    SpeedSet_Flag=TimeSet_Flag=1;//进入设置
                    Speed_Twinkle_Time = Time_Twinkle_Time = 0;//关闭闪烁
                }
                Beep_Time = 0.1;//蜂鸣器响 0.1S
            }
            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)
                {
                    sys.Run_Status = 1;//启动
                    Speed.ADDMode = 0;//进入判断速度显示处理
                    Speed_Val.Integral = 60;//电机起步的 PWM
                    SpeedSet_Flag=TimeSet_Flag=1;//进入设置
                    Speed_Twinkle_Time = Time_Twinkle_Time = 0;//关闭闪烁
                }
                else
                {
                    Speed.ADDMode = 2;//进入降速显示
                    Speed.Cw = 0;//转向清零
```

```
                    sys.Motor_Stop = 1;//检测电机
                    SpeedSet_Flag=TimeSet_Flag=1;//进入设置
                    Speed_Twinkle_Time = Time_Twinkle_Time = 0;//关闭闪烁
                }
                Beep_Time = 0.1;//蜂鸣器响 0.1S
            }
            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
                sys.SetMode_Option = 2;//设置时间
                if(HAL_GPIO_ReadPin(EC11A[0].EC11A_GPIO,EC11A[0].EC11A_Pin) ==
0)//加
                {
                    if(sys.SetMode_Option == 2)
```

```
                    {
                        if(EC11A[0].EC11A_Speed < EC11A[0].EC11A_Fast)//如果慢慢旋转
                            Time.Set += 60;
                        else
                            Time.Set += 600;
                        if(Time.Set > Time_MAX)
                            Time.Set = Time_MAX;
                        Key_Status = 1;//设置时 2S 不闪烁
                        Time_Twinkle_Time = 2.0f;//时间闪烁 2S
                    }
                    break;
                }
                else if(HAL_GPIO_ReadPin(EC11A[0].EC11A_GPIO,EC11A[0].EC11A_Pin) == 1)//减
                {
                    if(sys.SetMode_Option == 2)
                    {
                        if(EC11A[0].EC11A_Speed < EC11A[0].EC11A_Fast)//如果慢慢旋转
                            Time.Set -= 60;
                        else
                            Time.Set -= 600;
                        if(Time.Set <= Time_MIN)
                            Time.Set = Time_MIN;
                        Key_Status = 1;//设置时 2S 不闪烁
                        Time_Twinkle_Time = 2.0f;//时间闪烁 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
```

```
                sys.SetMode_Option = 1;//设置速度
                if(HAL_GPIO_ReadPin(EC11A[1].EC11A_GPIO,EC11A[1].EC11A_Pin) ==
0)//加
                {
                    /*加*/
                    if(sys.SetMode_Option == 1)
                    {
                        if(EC11A[1].EC11A_Speed  <  EC11A[1].EC11A_Fast)//如果慢慢
旋转
                        {
                            Speed.Set += 5;
                            if(Speed.Set == 5)//从零转开始为 100 转，判断是 5 后
                                Speed.Set = 100;//设定转速为 100 开始
                        }
                        else
                        {
                            Speed.Set += 50;
                            if(Speed.Set == 50)//从零转开始最低为 50 转，判断是 50 后
                                Speed.Set = 100;//设定转速为 100 开始
                        }
                        if(Speed.Set > Speed_MAX)
                            Speed.Set = Speed_MAX;
                        Key_Status = 1;//设置时 2S 不闪烁
                        Speed_Twinkle_Time = 2.0f;//速度闪烁 2S
                    }
                    break;
                }
                else if(HAL_GPIO_ReadPin(EC11A[1].EC11A_GPIO,EC11A[1].EC11A_Pin)
== 1)//减
                {
                    /*减*/
                    if(sys.SetMode_Option == 1)
                    {
                        if(EC11A[1].EC11A_Speed  <  EC11A[1].EC11A_Fast)//如果慢慢
旋转
                        {
                            Speed.Set -= 5;
                        }
                        else
                        {
                            Speed.Set -= 50;
                        }
                        if(Speed.Set < Speed_MIN)
                            Speed.Set = 50;
                        Key_Status = 1;//设置时 2S 不闪烁
                        Speed_Twinkle_Time = 2.0f;//速度闪烁 2S
                    }
                    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++;
    }
}
#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;
}
```