OS3100 软件源程序

```
#include "main.h"
#include "tim.h"
#include "gpio.h"
void SystemClock_Config(void);
int main(void)
{
    HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_TIM1_Init();
  MX_TIM3_Init();
    System_Init();
  while (1)
  {


  }

}

void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL6;
  RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                |RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
```

```
}
void Error_Handler(void)
{
  __disable_irq();
  while (1)
  {
  }

}

void assert_failed(uint8_t *file, uint32_t line)
{

}
#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_IC_Start_IT(&htim1, TIM_CHANNEL_2);//motor 输入捕获
}

/*
*********************************************************************
 * 函数原型：  void Check_Speed(void)
 * 功      能：  检测速度是否停止
*********************************************************************
*/
void Check_Speed(void)
{
    Speed_Cnt++;//每 50ms 进入
     if(Speed_Cnt>=10)//0.5s 发现没出发输入捕获
          Rel_Speed = 0;//将速度清零
}

/*
*********************************************************************
 * 函数原型：  void TIM1CaptureChannel2Callback(void)
 * 功      能：  Tim1 通道 2 的输入捕获回调函数
*********************************************************************
*/
uint32_t capture, capture1, capture2;
uint32_t rel;
void TIM1CaptureChannel2Callback()
{
```

```
    capture1 = __HAL_TIM_GET_COMPARE(&htim1, TIM_CHANNEL_2);
    if(capture1 > capture2)
        capture = capture1 - capture2;
    else
        capture = capture1 + (0xFFFF - capture2);
    if(capture < 100)
        return;
    rel = 60000000 / (capture * 6);
    capture2 = capture1;
    Rel_Speed = rel;
    Speed_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_2)
        {
            TIM1CaptureChannel2Callback();
        }
    }

}
#include "Ctrl_DownTime.h"

/*
**********************************************************************
 * 函数原型：  void Cheak_TimeDown(uint16_t dT)
 * 功      能：  时间倒计时检测
 * 输      入:  dT:执行周期
 * 参      数：  uint16_t dT
**********************************************************************
*/
void Cheak_TimeDown(uint16_t dT)
{
    static uint16_t T;
    if(DownTime_Over==1)//工位倒计时结束
    {
        DownTime_Over = 0;//将倒计时结束的标志位清零
        Ctrl_Time = Set_Time;//将设置时间重新赋值给控制时间
        Beep_Flash = 5;//蜂鸣器响 5 下
        sys.Run_Status = 0;//关闭
```

```
    }
    if(sys.Run_Status)//启动系统
    {
        T += dT;
        if(T == 1000)//1S
        {
            if(Time_State==0 && DownTime_Over == 0 && Ctrl_Speed)//如果实际时间显
示和倒计时没有结束的标志还在
            {
                if(Ctrl_Time)
                    Ctrl_Time--;//控制时间--
                else
                {
                    DownTime_Over= 1;//time1 倒计时结束
                }
            }
            T = 0;//周期清零
        }
    }
    else
    {
        DownTime_Over = 0;//将倒计时结束的标志位清零
        Ctrl_Time = Set_Time;//将设置时间重新赋值给控制时间
    }
}
#include "Ctrl_Motor.h"

/*
**********************************************************************
 * 函数原型：  void Motor_Check(float dT)
 * 功      能：  电机停止检测
**********************************************************************
*/
void Motor_Check(float dT)
{
    static float T1;
    if(Rel_Speed == 0)//设定速度并且实际速度等于 0
    {
        T1 += dT;
        if(T1 >= 0.5)
        {
            Speed_Val.SumError=0x1F00;//启动电机系数
        }
    }
    else
    {
        T1 = 0;
    }
}
```

```
/*
**********************************************************************
 * 函数原型：  void Motor_Ctrl(void)
 * 功      能：  电机控制
**********************************************************************
*/
void Motor_Ctrl(void)
{
    static float Tp;
    if(sys.Run_Status == 1)//启动
    {
        Tp++;
        Motor_Check(0.05);
        if(Tp>=11)
        {
            /**********Speed_L1**********/
            if(Ctrl_Speed && ((DownTime_Over == 0)||(Ctrl_Time)))//速度大于 0 和定时器
没有结束
            {

    HAL_GPIO_WritePin(BREAKEZ_GPIO_Port,BREAKEZ_Pin,GPIO_PIN_SET);// 高 电 平
不刹车，低电平刹车
                PID_Speed(Ctrl_Speed*4,Rel_Speed,&Speed_Arg,&Speed_Val);// 电 机  PID
控制
                PWM = Speed_Val.Out;//pid 输出
            }
            else
            {
                PWM = 0;//pwm 不输出
                Speed_Val.SumError  =  0;//防止关闭再打开时速度一下子就冲到之前的速
度
            }
            Tp = 12;
        }
    }
    else
    {
        HAL_GPIO_WritePin(BREAKEZ_GPIO_Port,BREAKEZ_Pin,GPIO_PIN_RESET);//
高电平不刹车，低电平刹车
        PWM = 0;//pwm 不输出
        Speed_Val.SumError = 0;//防止关闭再打开时速度一下子就冲到之前的速度
        Tp = 0;
    }
}

/*
**********************************************************************
 * 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_1);//开启 tim3 通道一
    HAL_GPIO_WritePin(BREAKEZ_GPIO_Port,BREAKEZ_Pin,GPIO_PIN_SET);// 高 电 平 不刹车，低电平刹车
    HAL_GPIO_WritePin(DIR_GPIO_Port,DIR_Pin,GPIO_PIN_SET);//高电平正转，低电平反转
}
#include "Ctrl_Scheduler.h"

uint16_t   T_cnt_2ms=0,
           T_cnt_10ms=0,
           T_cnt_20ms=0,
           T_cnt_50ms=0,
           T_cnt_100ms=0,
           T_cnt_200ms=0,
           T_cnt_500ms=0,
           T_cnt_1S=0;

void Loop_Check(void)
{
    T_cnt_2ms++;
    T_cnt_10ms++;
    T_cnt_20ms++;
    T_cnt_50ms++;
    T_cnt_100ms++;
    T_cnt_200ms++;
    T_cnt_500ms++;
    T_cnt_1S++;

    Sys_Loop();
}

static void Loop_2ms(void)//2ms 执行一次
{
    Display_Show();//显示屏幕
    EC11A_FlagCheak(2);//旋钮检测延时
}

static void Loop_10ms(void)//10ms 执行一次
{
     Check_KeyState();//按键检测
}

static void Loop_20ms(void)//20ms 执行一次
{


}
```

```
static void Loop_50ms(void)//50ms 执行一次
{
    Buzzer_Status(0.05);//蜂鸣器检测
    Check_Speed();//检测速度是否停止
    Motor_Ctrl();//电机控制
}

static void Loop_100ms(void)//100ms 执行一次
{
    Cheak_TimeDown(100);//时间倒计时检测
}

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

}

static void Loop_500ms(void)//500ms 执行一次
{
    Check_ShowFlag(500);//屏幕闪烁检测
}

static void Loop_1S(void)//1S 执行一次
{
    Check_Knob();//旋钮旋动检测
}

void Sys_Loop(void)
{
    if(T_cnt_2ms >= 2) {
        Loop_2ms();
        T_cnt_2ms = 0;
    }
    if(T_cnt_10ms >= 10) {
        Loop_10ms();
        T_cnt_10ms = 0;
    }
    if(T_cnt_20ms >= 20) {
        Loop_20ms();
        T_cnt_20ms = 0;
    }
    if(T_cnt_50ms >= 50) {
        Loop_50ms();
        T_cnt_50ms = 0;
    }
    if(T_cnt_100ms >= 100) {
        Loop_100ms();
        T_cnt_100ms = 0;
    }
```

```
    if(T_cnt_200ms >= 200) {
        Loop_200ms();
        T_cnt_200ms = 0;
    }
    if(T_cnt_500ms >= 500) {
        Loop_500ms();
        T_cnt_500ms = 0;
    }
    if(T_cnt_1S >= 1000) {
        Loop_1S();
        T_cnt_1S = 0;
    }
}
#include "System_Init.h"

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

    /**********电机初始化**********/
    Motor_Init();

    /**********编码器初始化**********/
    Encoder_Init();

    /**********系统参数初始化**********/
    PID_Init();//pid 系数初始化
    Set_Speed = 200;//开机设定速度位为 50 转
    Speed = Set_Speed;//将设定速度存储到临时速度
    Ctrl_Speed = Set_Speed;//将设定速度赋值给控制速度
    Beep_Time = 0.1;//蜂鸣器响 0.1S
    Time_State = 1;//时间开机显示“----”

    /**********系统初始化成功**********/
    sys.Init_ok = 1;
}
#include "Structs.h"

_sys_ sys;//系统初始化检测

uint16_t Speed;//临时速度
uint16_t Rel_Speed;//实际速度
int16_t Set_Speed;//设定速度
```

```
uint16_t Ctrl_Speed;//控制速度
int16_t Display_Speed;//显示速度
uint8_t Speed_Cnt;//速度清零计数
uint8_t Speed_ADDMode;//速度显示模式
uint16_t Speed_New;//现在的速度
uint16_t Speed_Last;//上一次的速度

int32_t Time;//临时时间
int32_t Set_Time;//设定时间
int32_t Ctrl_Time;//控制时间
uint8_t Time_State;//时间的状态
int32_t Display_Time;//显示时间
uint8_t DownTime_Over;//倒计时结束
#ifndef __Structs_H__
#define __Structs_H__

#include "stm32f0xx_hal.h"

typedef struct
{
    uint8_t Init_ok;//系统初始化是否完成，完成为 1
    uint8_t Run_Status;//系统状态
    uint8_t SetMode_Option;//选择设置模式
}_sys_;
extern _sys_ sys;//系统初始化检测

extern uint16_t Speed;//临时速度
extern uint16_t Rel_Speed;//实际速度
extern int16_t Set_Speed;//设定速度
extern uint16_t Ctrl_Speed;//控制速度
extern int16_t Display_Speed;//显示速度
extern uint8_t Speed_Cnt;//速度清零计数
extern uint8_t Speed_ADDMode;//速度显示模式
extern uint16_t Speed_New;//现在的速度
extern uint16_t Speed_Last;//上一次的速度

extern int32_t Time;//临时时间
extern int32_t Set_Time;//设定时间
extern int32_t Ctrl_Time;//控制时间
extern uint8_t Time_State;//时间的状态
extern int32_t Display_Time;//显示时间
extern uint8_t DownTime_Over;//倒计时结束
#endif
#include "Drv_EC11A.h"

/**********全局变量声明******/
uint8_t EC11A_Knob;//在旋动旋钮时

/**********局部变量声明******/
uint8_t EC11A_Flag;//进入中断延时标志
```

```
uint8_t Key1_Press;//按下按钮
uint16_t KEY1_Count;//记录 KEY1 按下的时间

/*
**********************************************************************
 * 函数原型：void EC11A_FlagCheak(uint16_t dT)
 * 功      能：检测延时检测延时-2ms
 * 输      入: dT ： 周期
 * 参      数：uint16_t dT
**********************************************************************
*/
void EC11A_FlagCheak(uint16_t dT)
{
    static uint16_t T;
    T += dT;//周期加加
    if(T % 4 == 0)//计时 4ms
    {
        EC11A_Flag = 1;//进入中断
        T = 0;//计时清零
    }
}

/*
**********************************************************************
 * 函数原型：void Check_Knob(void)
 * 功      能：检测旋钮状态-500ms
**********************************************************************
*/
void Check_Knob(void)
{
    if(EC11A_Knob)//旋钮被转动
        EC11A_Knob--;//1S 倒计时
}

/*
**********************************************************************
 * 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)
{
    if(EC11A_Flag == 1)//进入中断
    {
        /***********左旋转**********/
        if(GPIO_Pin == KEY1B_Pin)//左边旋钮触发
        {

if((HAL_GPIO_ReadPin(KEY1A_GPIO_Port,KEY1A_Pin)==1)&&(HAL_GPIO_ReadPin(KEY
1B_GPIO_Port,KEY1B_Pin)==0))//如果向左旋转
```

```
            {
                if(sys.SetMode_Option == 1)
                {
                    Set_Speed -= 10;
                    if(Set_Speed <= 50)
                        Set_Speed = 50;
                    Speed = Set_Speed;
                    Ctrl_Speed = Set_Speed;
                }
                if(sys.SetMode_Option == 2)
                {
                    Set_Time -= 60;
                    if(Set_Time <= 0)
                    {
                        Time_State = 1;//显示“----”
                        Set_Time = 0;
                    }
                    Time = Set_Time;
                    Ctrl_Time = Set_Time;
                }
                EC11A_Knob = 1;//检测是不是在旋动旋钮
                Twinkle_Time = 6000;//闪烁显示 6S
            }
            else
if((HAL_GPIO_ReadPin(KEY1A_GPIO_Port,KEY1A_Pin)==0)&&(HAL_GPIO_ReadPin(KEY
1B_GPIO_Port,KEY1B_Pin)==0))//如果向右旋转
            {
                if(sys.SetMode_Option == 1)
                {
                    Set_Speed += 10;
                    if(Set_Speed >= 1200)
                        Set_Speed = 1200;
                    Speed = Set_Speed;
                    Ctrl_Speed = Set_Speed;
                }
                if(sys.SetMode_Option == 2)
                {
                    Set_Time += 60;
                    Time_State = 0;//不显示“----”
                    if(Set_Time >= 86400)
                        Set_Time = 86400;
                    Time = Set_Time;
                    Ctrl_Time = Set_Time;
                }
                EC11A_Knob = 1;//检测是不是在旋动旋钮
                Twinkle_Time = 6000;//闪烁显示 6S
            }
        }
        EC11A_Flag = 0;//关闭中断
    }
```

```
    /**********左边按钮中断**********/
    if(GPIO_Pin ==KEY1_Pin)
    {
        Key1_Press = 1;//按下标志被置一
    }
}

/*
*********************************************************************
 * 函数原型：void Check_KeyState(void)
 * 功      能：按键检测
*********************************************************************
*/
void Check_KeyState(void)
{
    /**********KEY1**********/
    if(Key1_Press == 1)//按钮被按下
    {
        if(HAL_GPIO_ReadPin(KEY1_GPIO_Port,KEY1_Pin)==0)//如果 KEY1 按下
            KEY1_Count++;//按下时间++
        if(HAL_GPIO_ReadPin(KEY1_GPIO_Port,KEY1_Pin)==1)//如果 KEY1 抬起
        {
            if(KEY1_Count < 200)//短按
            {
                if(sys.Run_Status == 0)
                {
                    sys.SetMode_Option++;//设置模式++
                    if(sys.SetMode_Option == 3)
                    {
                        sys.SetMode_Option = 0;
                    }
                }
                else//启动下单击按键，直接停止
                {
                    sys.Run_Status = 0;//关闭运行
                }
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 6000;//闪烁显示 6S
                KEY1_Count = 0;//按钮计数清零
                Key1_Press = 0;//按钮状态为抬起
            }
            else//等于 200 时再清零
            {
                KEY1_Count = 0;//按钮计数清零
                Key1_Press = 0;//按钮状态为抬起
            }

        }
        if(KEY1_Count > 200 && KEY1_Count < 400)//长按
```

```
        {
            if(sys.Run_Status == 0)
            {
                sys.Run_Status = 1;
                sys.SetMode_Option = 0;
                Speed_ADDMode = 0;
            }
            else
            {
                sys.Run_Status = 0;
            }
            Beep_Time = 0.1;//蜂鸣器响 0.1S
            KEY1_Count = 400;//按钮计数等于 200，这样就不会在抬起再进入单击了
        }
    }
}
#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 < 0.1)//如果小于 0.1s 时
        {
            Beep_ON;//蜂鸣器响
        }
```

```
            else if(BT >= 0.1 && BT < 0.2)//在 0.1 和 0.2s 之间时
            {
                Beep_OFF;//关闭蜂鸣器
            }
            else if(BT >= 0.2)//大于等于 0.2s 时
            {
                Beep_Flash--;//次数--
                BT = 0;//周期清零
            }
        }
}
#include "Drv_LedDisplay.h"

/**********全局变量声明******/
uint16_t Twinkle_Time;//闪烁时间
uint8_t IconRun_Time;//跑圈时间

/**********局部变量声明******/
uint8_t DIG,UC,UC10;//DIG,UC 的数据
uint8_t SPEED_Tab[] = {0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE0, 0xFE, 0xF6};//显
示 0-9
uint8_t Icon_Run[] = {0xB4, 0xD8, 0x6C};//圆圈跑起来
uint8_t t;//显示时间
uint8_t Speed_ShowFlag = 0;//速度闪烁
uint8_t Time_ShowFlag = 0;//时间闪烁
uint8_t TimeIcon_ShowFalg = 0;//时间冒号闪烁
uint8_t Icon_ShowFlag = 0;//运行闪烁

/*
*********************************************************************
 * 函数原型：  void Check_ShowFlag(uint16_t dT)
 * 功      能：  闪烁检测
 * 输      入:  dT:执行周期
 * 参      数：  uint16_t dT
*********************************************************************
*/
void Check_ShowFlag(uint16_t dT)
{
    if(sys.Run_Status)//运行时
    {
        TimeIcon_ShowFalg = ~TimeIcon_ShowFalg;
        Icon_ShowFlag = ~Icon_ShowFlag;
    }
    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
        Time_ShowFlag = 0;//常亮
        Twinkle_Time = 0;//闪烁计时清零
        return;
    }
```

```
    if(Twinkle_Time && EC11A_Knob==0)//闪烁和没有操作旋钮时
    {
        Twinkle_Time -= dT;//闪烁计时
        if(sys.SetMode_Option == 1)//设置速度
        {
            Speed_ShowFlag = ~Speed_ShowFlag;//速度闪烁
            Time_ShowFlag = 0;//时间常亮
        }
        else if(sys.SetMode_Option == 2)//设置温度
        {
            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
        }
    }
}

/*
**********************************************************************
 * 函数原型：  void UCdata_Display(uint8_t uc)
 * 功    能：  UC 数据判断控制引脚
**********************************************************************
*/
void UCdata_Display(uint8_t uc)
{
     for(uint8_t i=0; i<7; i++)
    {
        if((uc<<i) & 0x80)
        {
            switch(i)
            {
                case 0: HAL_GPIO_WritePin(UC_A_GPIO_Port, UC_A_Pin,
GPIO_PIN_SET);break;
                case 1: HAL_GPIO_WritePin(UC_B_GPIO_Port, UC_B_Pin,
GPIO_PIN_SET);break;
                case 2: HAL_GPIO_WritePin(UC_C_GPIO_Port, UC_C_Pin,
GPIO_PIN_SET);break;
                case 3: HAL_GPIO_WritePin(UC_D_GPIO_Port, UC_D_Pin,
GPIO_PIN_SET);break;
                case 4: HAL_GPIO_WritePin(UC_E_GPIO_Port, UC_E_Pin,
GPIO_PIN_SET);break;
                case 5: HAL_GPIO_WritePin(UC_F_GPIO_Port, UC_F_Pin,
GPIO_PIN_SET);break;
                case 6: HAL_GPIO_WritePin(UC_G_GPIO_Port, UC_G_Pin,
GPIO_PIN_SET);break;
            }
        }
```

```
            else
            {
                switch(i)
                {
                    case    0:    HAL_GPIO_WritePin(UC_A_GPIO_Port,    UC_A_Pin,
GPIO_PIN_RESET);break;
                    case    1:    HAL_GPIO_WritePin(UC_B_GPIO_Port,    UC_B_Pin,
GPIO_PIN_RESET);break;
                    case    2:    HAL_GPIO_WritePin(UC_C_GPIO_Port,    UC_C_Pin,
GPIO_PIN_RESET);break;
                    case    3:    HAL_GPIO_WritePin(UC_D_GPIO_Port,    UC_D_Pin,
GPIO_PIN_RESET);break;
                    case    4:    HAL_GPIO_WritePin(UC_E_GPIO_Port,    UC_E_Pin,
GPIO_PIN_RESET);break;
                    case    5:    HAL_GPIO_WritePin(UC_F_GPIO_Port,    UC_F_Pin,
GPIO_PIN_RESET);break;
                    case    6:    HAL_GPIO_WritePin(UC_G_GPIO_Port,    UC_G_Pin,
GPIO_PIN_RESET);break;
                }
            }
        }
}

/*
**********************************************************************
 * 函数原型：  DIGdata_Display(uint8_t DIG)
 * 功      能：  DIG 数据判断控制引脚
**********************************************************************
*/
void DIGdata_Display(uint8_t DIG)
{
       for(uint8_t i=0; i<8; i++)
      {
            if((DIG<<i) & 0x80)
            {
                switch(i)
                {
                    case    0:    HAL_GPIO_WritePin(DIG1_GPIO_Port,    DIG1_Pin,
GPIO_PIN_RESET);break;
                    case    1:    HAL_GPIO_WritePin(DIG2_GPIO_Port,    DIG2_Pin,
GPIO_PIN_RESET);break;
                    case    2:    HAL_GPIO_WritePin(DIG3_GPIO_Port,    DIG3_Pin,
GPIO_PIN_RESET);break;
                    case    3:    HAL_GPIO_WritePin(DIG4_GPIO_Port,    DIG4_Pin,
GPIO_PIN_RESET);break;
                    case    4:    HAL_GPIO_WritePin(DIG5_GPIO_Port,    DIG5_Pin,
GPIO_PIN_RESET);break;
                    case    5:    HAL_GPIO_WritePin(DIG6_GPIO_Port,    DIG6_Pin,
GPIO_PIN_RESET);break;
                    case    6:    HAL_GPIO_WritePin(DIG7_GPIO_Port,    DIG7_Pin,
```

```
GPIO_PIN_RESET);break;
                    case    7:    HAL_GPIO_WritePin(DIG8_GPIO_Port,    DIG8_Pin,
GPIO_PIN_RESET);break;
                }
            }
            else
            {
                switch(i)
                {
                    case    0:    HAL_GPIO_WritePin(DIG1_GPIO_Port,    DIG1_Pin,
GPIO_PIN_SET);break;
                    case    1:    HAL_GPIO_WritePin(DIG2_GPIO_Port,    DIG2_Pin,
GPIO_PIN_SET);break;
                    case    2:    HAL_GPIO_WritePin(DIG3_GPIO_Port,    DIG3_Pin,
GPIO_PIN_SET);break;
                    case    3:    HAL_GPIO_WritePin(DIG4_GPIO_Port,    DIG4_Pin,
GPIO_PIN_SET);break;
                    case    4:    HAL_GPIO_WritePin(DIG5_GPIO_Port,    DIG5_Pin,
GPIO_PIN_SET);break;
                    case    5:    HAL_GPIO_WritePin(DIG6_GPIO_Port,    DIG6_Pin,
GPIO_PIN_SET);break;
                    case    6:    HAL_GPIO_WritePin(DIG7_GPIO_Port,    DIG7_Pin,
GPIO_PIN_SET);break;
                    case    7:    HAL_GPIO_WritePin(DIG8_GPIO_Port,    DIG8_Pin,
GPIO_PIN_SET);break;
                }
            }
        }
}

/*
**********************************************************************
 * 函数原型：  void DIGdata_Set(void)
 * 功      能：  1-8DIG 引脚全部拉高
**********************************************************************
*/
void DIGdata_Set(void)
{
    HAL_GPIO_WritePin(DIG1_GPIO_Port, DIG1_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(DIG2_GPIO_Port, DIG2_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(DIG3_GPIO_Port, DIG3_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(DIG4_GPIO_Port, DIG4_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(DIG5_GPIO_Port, DIG5_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(DIG6_GPIO_Port, DIG6_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(DIG7_GPIO_Port, DIG7_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(DIG8_GPIO_Port, DIG8_Pin, GPIO_PIN_SET);
}

/*
**********************************************************************
```

```
 * 函数原型：  void Display_SpeedShow(uint16_t speed)
 * 功      能：  显示速度
 * 输      入:  speed：要显示的速度
 * 参      数：  uint16_t speed
**********************************************************************
*/
void Display_SpeedShow(uint16_t speed)
{
    uint8_t Val;//用于百十个取出来的数字
    if(t == 1)
    {
        /***********L1 千位***********/
        if(speed > 999)//大于 999 时
        {
            UC = SPEED_Tab[1];//显示 1
            UCdata_Display(UC);
            if(Speed_ShowFlag >= 1 && EC11A_Knob == 0)//闪烁速度
                DIGdata_Display(0x00);
            else
                DIGdata_Display(0x80);
        }
        else//小于 999 时
        {
            DIGdata_Display(0x00);
        }
    }
    else if(t == 2)
    {
        /***********L1 百位***********/
        if(speed > 99)//大于 99 时
        {
            Val=speed/100;//取出百位的数字
            if(speed > 999)//加入大于 999 时
                Val=Val%10;//取出百位的数字
            switch(Val)
            {
                case 0:UC = SPEED_Tab[0];//数字 0
                    break;
                case 1:UC = SPEED_Tab[1];//数字 1
                    break;
                case 2:UC = SPEED_Tab[2];//数字 2
                    break;
                case 3:UC = SPEED_Tab[3];//数字 3
                    break;
                case 4:UC = SPEED_Tab[4];//数字 4
                    break;
                case 5:UC = SPEED_Tab[5];//数字 5
                    break;
                case 6:UC = SPEED_Tab[6];//数字 6
                    break;
```

```
                case 7:UC = SPEED_Tab[7];//数字 7
                    break;
                case 8:UC = SPEED_Tab[8];//数字 8
                    break;
                case 9:UC = SPEED_Tab[9];//数字 9
                    break;
                default:
                    break;
            }
            UCdata_Display(UC);
            if(Speed_ShowFlag >= 1 && EC11A_Knob == 0)
                DIGdata_Display(0x00);
            else
                DIGdata_Display(0x40);
        }
        else
        {
            DIGdata_Display(0x00);//不显示
        }

    }
    else if(t == 3)
    {
        /***********L1 十位***********/
        if(speed > 9)//大于 9 时
        {
            Val=speed/10;//取出十位的数字
            if(speed > 99)//大于 99 时
                Val=Val%10;//取出十位的数字
            switch(Val)
            {
                case 0:UC = SPEED_Tab[0];//数字 0
                    break;
                case 1:UC = SPEED_Tab[1];//数字 1
                    break;
                case 2:UC = SPEED_Tab[2];//数字 2
                    break;
                case 3:UC = SPEED_Tab[3];//数字 3
                    break;
                case 4:UC = SPEED_Tab[4];//数字 4
                    break;
                case 5:UC = SPEED_Tab[5];//数字 5
                    break;
                case 6:UC = SPEED_Tab[6];//数字 6
                    break;
                case 7:UC = SPEED_Tab[7];//数字 7
                    break;
                case 8:UC = SPEED_Tab[8];//数字 8
                    break;
                case 9:UC = SPEED_Tab[9];//数字 9
```

```
                    break;
                default:
                    break;
            }
            UCdata_Display(UC);
            if(Speed_ShowFlag >= 1 && EC11A_Knob == 0)
                DIGdata_Display(0x00);
            else
                DIGdata_Display(0x20);
        }
        else
        {
            DIGdata_Display(0x00);//不显示
        }
    }
    else if(t == 4)
    {
        /***********L1 个位***********/
        UC = SPEED_Tab[0];//数字 0
        UCdata_Display(UC);
        if(Speed_ShowFlag >= 1 && EC11A_Knob == 0)
            DIGdata_Display(0x00);
        else
            DIGdata_Display(0x10);
    }
}

/*
**********************************************************************
 * 函数原型：  void Display_TimeShow(int32_t time)
 * 功      能：  显示时间
 * 输      入:  time：要显示的时间
 * 参      数：  int32_t time
**********************************************************************
*/
void Display_TimeShow(int32_t time)
{
    uint8_t SH,H,SM,M;//用于百十个取出来的数字
    SH=time/3600/10;//计算十位单位的小时数
    H=time/3600%10;//计算个位单位的小时数
    SM=time%3600/60/10;//计算十分位单位的分钟数
    M=time%3600/60%10;//计算个分位单位的分钟数
    if(t == 5)
    {
        /***********L1 千位***********/
        switch(SH)
        {
            case 0:UC = SPEED_Tab[0];//数字 0
                break;
            case 1:UC = SPEED_Tab[1];//数字 1
```

```
                break;
            case 2:UC = SPEED_Tab[2];//数字 2
                break;
            case 3:UC = SPEED_Tab[3];//数字 3
                break;
            case 4:UC = SPEED_Tab[4];//数字 4
                break;
            case 5:UC = SPEED_Tab[5];//数字 5
                break;
            case 6:UC = SPEED_Tab[6];//数字 6
                break;
            case 7:UC = SPEED_Tab[7];//数字 7
                break;
            case 8:UC = SPEED_Tab[8];//数字 8
                break;
            case 9:UC = SPEED_Tab[9];//数字 9
                break;
            default:
                break;
        }
        if(Time_State == 1)
            UCdata_Display(0x02);
        else
            UCdata_Display(UC);
        if(Time_ShowFlag >= 1 && EC11A_Knob == 0)//时间闪烁
            DIGdata_Display(0x00);
        else
            DIGdata_Display(0x08);
    }
    else if(t == 6)
    {
        /***********L1 百位***********/
        switch(H)
        {
            case 0:UC = SPEED_Tab[0];//数字 0
                break;
            case 1:UC = SPEED_Tab[1];//数字 1
                break;
            case 2:UC = SPEED_Tab[2];//数字 2
                break;
            case 3:UC = SPEED_Tab[3];//数字 3
                break;
            case 4:UC = SPEED_Tab[4];//数字 4
                break;
            case 5:UC = SPEED_Tab[5];//数字 5
                break;
            case 6:UC = SPEED_Tab[6];//数字 6
                break;
            case 7:UC = SPEED_Tab[7];//数字 7
                break;
```

```
            case 8:UC = SPEED_Tab[8];//数字 8
                break;
            case 9:UC = SPEED_Tab[9];//数字 9
                break;
            default:
                break;
        }
        if(Time_State == 1)
            UCdata_Display(0x02);
        else
            UCdata_Display(UC);
        if(Time_ShowFlag >= 1 && EC11A_Knob == 0)
            DIGdata_Display(0x00);
        else
            DIGdata_Display(0x04);
    }
    else if(t == 7)
    {
        /***********L1 十位***********/
        switch(SM)
        {
            case 0:UC = SPEED_Tab[0];//数字 0
                break;
            case 1:UC = SPEED_Tab[1];//数字 1
                break;
            case 2:UC = SPEED_Tab[2];//数字 2
                break;
            case 3:UC = SPEED_Tab[3];//数字 3
                break;
            case 4:UC = SPEED_Tab[4];//数字 4
                break;
            case 5:UC = SPEED_Tab[5];//数字 5
                break;
            case 6:UC = SPEED_Tab[6];//数字 6
                break;
            case 7:UC = SPEED_Tab[7];//数字 7
                break;
            case 8:UC = SPEED_Tab[8];//数字 8
                break;
            case 9:UC = SPEED_Tab[9];//数字 9
                break;
            default:
                break;
        }
        if(Time_State == 1)
            UCdata_Display(0x02);
        else
            UCdata_Display(UC);
        if(Time_ShowFlag >= 1 && EC11A_Knob == 0)
            DIGdata_Display(0x00);
```

```
            else
                DIGdata_Display(0x02);
        }
        else if(t == 8)
        {
            /***********L1 个位***********/
            switch(M)
            {
                case 0:UC = SPEED_Tab[0];//数字 0
                    break;
                case 1:UC = SPEED_Tab[1];//数字 1
                    break;
                case 2:UC = SPEED_Tab[2];//数字 2
                    break;
                case 3:UC = SPEED_Tab[3];//数字 3
                    break;
                case 4:UC = SPEED_Tab[4];//数字 4
                    break;
                case 5:UC = SPEED_Tab[5];//数字 5
                    break;
                case 6:UC = SPEED_Tab[6];//数字 6
                    break;
                case 7:UC = SPEED_Tab[7];//数字 7
                    break;
                case 8:UC = SPEED_Tab[8];//数字 8
                    break;
                case 9:UC = SPEED_Tab[9];//数字 9
                    break;
                default:
                    break;
            }
            if(Time_State == 1)
                UCdata_Display(0x02);
            else
                UCdata_Display(UC);
            if(Time_ShowFlag >= 1 && EC11A_Knob == 0)
                DIGdata_Display(0x00);
            else
                DIGdata_Display(0x01);
        }
}

/*
*********************************************************************
 * 函数原型：  void Display_Icon(void)
 * 功      能：  显示图标
*********************************************************************
*/
void Display_Icon(void)
{
```

```
    if(t == 9)
    {
        UC = 0x18;//时间冒号图标
        UCdata_Display(UC);
        DIGdata_Set();
        if(TimeIcon_ShowFalg >= 1 && sys.Run_Status == 1 && Time_State == 0)//闪烁
        {
            DIG9_OFF;
        }
        else
        {
            DIG9_ON;
        }
        DIG10_OFF;
    }
    else if(t == 10)
    {
        static uint8_t cnt;
        IconRun_Time++;
        if(IconRun_Time >= 10)//200ms 跑圈
        {
            cnt ++;
            if(cnt == 4)
                cnt = 1;
            switch(cnt)
            {
                case 1:UC10 = Icon_Run[0];break;
                case 2:UC10 = Icon_Run[1];break;
                case 3:UC10 = Icon_Run[2];break;
            }
            IconRun_Time = 0;
        }
        if(sys.Run_Status == 0)
            UC10 = 0xFC;
        UC = UC10;
        UCdata_Display(UC);
        DIGdata_Set();
        DIG9_OFF;
        DIG10_ON;
    }
    if(t == 11)//刷新
    {
        DIGdata_Set();
        DIG9_OFF;
        DIG10_OFF;
        t = 0;
    }
}

/*
```

```
*********************************************************************
 * 函数原型：  void Deal_Speed(void)
 * 功    能：  速度显示处理
*********************************************************************
*/
void Deal_Speed(void)
{
    /**********SpeedL1_ADD_Mode**********/
    if(sys.Run_Status == 1)//启动的情况下
    {
        if(Speed_ADDMode == 0)//在电机控制中，速度未处理
        {
            Display_Speed = 0;
            Speed_New =0;//现在的速度清零
            Speed_Last = 0;//之前的速度清零
            Speed_ADDMode = 1;//进入加速模式下
        }
        if(Speed_ADDMode==1)//在进入加速模式下
        {
            if((Rel_Speed/4) >= Ctrl_Speed)//实际速度大于等于控制速度
            {
                Speed_ADDMode = 3;//进入稳定模式
                return;
            }
            Speed_New = (Rel_Speed/4);//记录当前速度
            if(Speed_New > Speed_Last)//当前速度大于上一次速度
                Display_Speed = Speed_New;//显示当前速度
            else//当前速度小于上一次速度
            {
                Display_Speed = 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 == 3)//速度稳定模式下
        {
            Display_Speed = Ctrl_Speed;//显示控制速度
        }
    }
}

/*
*********************************************************************
 * 函数原型：  void Display_Show(void)
 * 功    能：  显示屏幕
*********************************************************************
*/
void Display_Show(void)
{
```

```
    t++;
    if(sys.Run_Status == 0)
    {
        Display_Speed = Set_Speed;
        Display_Time = Set_Time;
    }
    else
    {
        Deal_Speed();
        Display_Time = Ctrl_Time + 59;//显示时间加 1 分钟

    }
    Display_SpeedShow(Display_Speed);
    Display_TimeShow(Display_Time);
    Display_Icon();
}
```