RM3100 软件源程序

```
#include "Drv_HT1623.h"

/*
*********************************************************************
 * 函数原型：static void delay(uint16_t time)
 * 功      能：us 延时
 * 输      入: time ：时间
 * 参      数：uint16_t time
 * 调      用：内部调用
*********************************************************************
*/
static void delay(uint16_t time)
{
    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)
{
    delay(10);
    Clr_1625_Wr;//RW = 0;
    delay(10);
    Set_1625_Dat;//DA = 1;
    Set_1625_Wr;//RW = 1;
    delay(10);
    Clr_1625_Wr;//RW = 0;
    delay(10);
    Clr_1625_Dat;
    delay(10);//DA = 0;
    Set_1625_Wr;//RW = 1;
    delay(10);
    Clr_1625_Wr;//RW = 0;
    delay(10);
    if(0 == MODE)
    {
        Clr_1625_Dat;//DA = 0;
    }
    else
    {
        Set_1625_Dat;//DA = 1;
```

```
    }
    delay(10);
    Set_1625_Wr;//RW = 1;
    delay(10);
}

/*
*********************************************************************
 * 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_1625_Wr;
        if((Cbyte >> (7 - i)) & 0x01)
        {
            Set_1625_Dat;
        }
        else
        {
            Clr_1625_Dat;
        }
        delay(10);
        Set_1625_Wr;
        delay(10);
    }
    Clr_1625_Wr;
    delay(10);
    Clr_1625_Dat;
    Set_1625_Wr;
    delay(10);
}

/*
*********************************************************************
 * 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_1625_Wr;
        if((Abyte >> (6 - i)) & 0x01)
        {
            Set_1625_Dat;
        }
        else
        {
            Clr_1625_Dat;
        }
        delay(10);
        Set_1625_Wr;
        delay(10);
    }
}

/*
*********************************************************************
 * 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_1625_Wr;
        if((Dbyte >> (7 - i)) & 0x01)
        {
            Set_1625_Dat;
        }
        else
        {
            Clr_1625_Dat;
        }
        delay(10);
        Set_1625_Wr;
        delay(10);
    }
}

/*
```

```
**********************************************************************
 * 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_1625_Wr;
        if((Dbyte >> (3 - i)) & 0x01)
        {
            Set_1625_Dat;
        }
        else
        {
            Clr_1625_Dat;
        }
        delay(10);
        Set_1625_Wr;
        delay(10);
    }
}

/*
**********************************************************************
 * 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_1625_Cs;
    Set_1625_Wr;
    Set_1625_Dat;
    delay(500);
    Clr_1625_Cs;//CS = 0;
    delay(10);
    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_1625_Cs;//CS = 1;

    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2, 95);//不输出 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_1625_Cs;//CS = 0;
    delay(10);
    Write_Mode(1);
```

```
    Write_Address(_addr);
    for (i = 0; i < n; i++)
    {
        Write_Data_8bit(_dat);
    }
    Set_1625_Cs;//CS = 1;
}
#include "Drv_Motor.h"

/*
**********************************************************************
 * 函数原型：void Motor_Init(void)
 * 功      能：电机初始化
**********************************************************************
*/
void Motor_Init(void)
{
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);//开启 tim1 通道一
}
#include "Drv_Beep.h"

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

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

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

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

    HAL_FLASH_Unlock();//解锁 Flash

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

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

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

```
    for(uint16_t i=0; i<len; i=i+2)
    {
        uint16_t temp;//临时存储数值
        if(i+1 <= len-1)
            temp = (uint16_t)(addr[i+1]<<8) + addr[i];
        else
            temp = 0xff00 + addr[i];
        //对 Flash 进行烧写，FLASH_TYPEPROGRAM_HALFWORD 声明操作的 Flash 地址的 16 位的，此外还有 32 位跟 64 位的操作，自行翻查 HAL 库的定义即可
        FlashStatus = HAL_FLASH_Program(FLASH_TYPEPROGRAM_HALFWORD, PARAMFLASH_BASE_ADDRESS+i, temp);
        if (FlashStatus != HAL_OK)
            return 0;
    }
    HAL_FLASH_Lock();//锁住 Flash
    return 1;
}

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

/**********全局变量声明******/
```

```
uint8_t Key_Status;//按键按下标志

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

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

/*
**********************************************************************
 * 函数原型：void Key_Scan(float dT)
 * 功      能：矩阵按键扫描
**********************************************************************
*/
void Key_Scan(float dT)
{
    /**************************************Start                                    键
****************************************/
    if(KEY3 == 0)//按下按键
    {
        if(LongPress1 == 0)//没有长按过
        {
            Key_Cnt1 += dT;//按下时间++
            Key_Flag1 = 1;//按键按下标志置一
        }
    }
    if(Key_Flag1 == 1)//按键被按下
    {
        if(KEY3 == 1)//抬起按键
        {
            if(Key_Cnt1 < 1.5)//小于 1.5S 是单击
            {
                if(sys.Run_Status == 0)//系统没有启动的话
                {
                    Speed_Val.SumError = 0x7AC4;//启动电机系数
//                  Speed_Val.SumError = 0x6F9B;//启动电机系数
                    SetOK_Flag = 1;//设定数值标志
                    Ctrl_Time = Set_Time;//把设定时间赋值给控制时间
                    Param.Time = Set_Time;//时间
```

```
                    sys.Run_Status = 1;//启动系统
                    sys.SetMode_Option = 0;//设置选项清零
                    Speed_ADDMode = 0;//数据处理重新开始
                }
                else//系统启动的话
                {
                    sys.Motor_Stop = 1;//检测电机
                    Speed_ADDMode = 4;//进入减速模式下
                }
                Beep_Time = 0.1;//蜂鸣器响 0.1S
            }
            Key_Flag1 = 0;//按键事件结束，等待下一次按下
            LongPress1 = 0;//长按标志清零
            Key_Cnt1 = 0;//按钮计数清零
        }
        if(Key_Cnt1 > 1.5 && Key_Cnt1 < 3.0)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress1 == 0)/*长按*///如果没有一直一直长按着
            {

                LongPress1 = 1;//长按标志置一
            }
        }
    }
    /************************************ 加 键
****************************************/
    if(KEY2== 0)//按下按键
    {
        Key_Cnt2 += dT;//按下时间++
        Key_Flag2 = 1;//按键按下标志置一
    }
    if(Key_Flag2 == 1)//按键被按下
    {
        if(KEY2 == 1)//抬起按键
        {
            if(Key_Cnt2 < 1.5)//小于 1.5S 是单击
            {
                if(sys.SetMode_Option == 1)//在设置速度时
                {
                    Set_Speed +=1;//速度++
                    if(Set_Speed > 80)//速度大于 80 时
                        Set_Speed = 80;//速度最大为 80 就上不去了
                }
                if(sys.SetMode_Option == 2)//在设置时间时
                {
                    Set_Time += 60;//设置时间加 1 分钟
                    if(Set_Time>86399)//时间大于 23 小时 59 分
                        Set_Time = 86399;//时间最大为 23 小时 59 分
                }
                Key_Status = 1;//按键一秒，表示按下时有 1s 的等待，不闪烁
```

```
                Twinkle_Time = 6;//一共闪烁 6S
            }
            Key_Flag2 = 0;//按键事件结束，等待下一次按下
            Key_Cnt2 = 0;//按钮计数清零
        }
        if(Key_Cnt2 > 1.9 && Key_Cnt2 < 2.1)//按键时间大于 1.9S 小于 2.1S 表示长按
        {
            if(sys.SetMode_Option == 1)//在设置速度时
            {
                Set_Speed +=10;//速度加 10
                if(Set_Speed > 80)//速度大于 80 时
                    Set_Speed = 80;//速度最大为 80 就上不去了
            }
            if(sys.SetMode_Option == 2)//在设置时间时
            {
                Set_Time += 600;//设置时间加 10 分钟
                if(Set_Time>86399)//时间大于 23 小时 59 分
                    Set_Time = 86399;//时间最大为 23 小时 59 分
            }
            Key_Status = 1;//按键一秒，表示按下时有 1s 的等待，不闪烁
            Twinkle_Time = 6;//一共闪烁 6S
            Key_Flag2 = 0;//按键事件结束，等待下一次按下
            Key_Cnt2 = 1.5;//按钮计数
        }
    }

    /************************************ 减 键
************************************/
    if(KEY4 == 0)//按下按键
    {
        Key_Cnt3 += dT;//按下时间++
        Key_Flag3 = 1;//按键按下标志置一
    }
    if(Key_Flag3 == 1)//按键被按下
    {
        if(KEY4 == 1)//抬起按键
        {
            if(Key_Cnt3 < 1.5)/*单击*///小于 1.5S 是单击
            {
                if(sys.SetMode_Option == 1)//在设置速度时
                {
                    Set_Speed -=1;//速度--
                    if(Set_Speed < 20)//速度小于 20
                        Set_Speed = 20;//速度最小为 20 就下不去了
                }
                if(sys.SetMode_Option == 2)//在设置时间时
                {
                    Set_Time -= 60;//每次时间减一分钟
                    if(Set_Time < 60)//时间小于一秒时
                        Set_Time = 0;//设置时间为 0
```

```
                }
                Key_Status = 1;//按键一秒，表示按下时有 1s 的等待，不闪烁
                Twinkle_Time = 6;//一共闪烁 6S
            }
            Key_Flag3 = 0;//按键事件结束，等待下一次按下
            Key_Cnt3 = 0;//按钮计数清零
        }
        if(Key_Cnt3 > 1.9 && Key_Cnt3 < 2.1)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(sys.SetMode_Option == 1)//在设置速度时
            {
                Set_Speed -=10;//速度减 10
                if(Set_Speed < 20)//速度小于 20
                    Set_Speed = 20;//速度最小为 20 就下不去了
            }
            if(sys.SetMode_Option == 2)//在设置时间时
            {
                Set_Time -= 600;//每次时间减十分钟
                if(Set_Time < 60)//时间小于一秒时
                    Set_Time = 0;//设置时间为 0
            }
            Key_Flag3 = 0;//按键事件结束，等待下一次按下
            Key_Cnt3 = 1.5;//按钮计数
            Key_Status = 1;//按键一秒，表示按下时有 1s 的等待，不闪烁
            Twinkle_Time = 6;//一共闪烁 6S
        }
    }
    /**************************************MENU 键
*****************************************/
    if(KEY1 == 0)//按下按键
    {
        if(sys.Motor_Stop)
            return;
        if(LongPress4 == 0)//没有长按过
        {
            Key_Cnt4 += dT;//按下时间++
            Key_Flag4 = 1;//按键按下标志置一
        }
    }
    if(Key_Flag4 == 1)//按键被按下
    {
        if(KEY1 == 1)//抬起按键
        {
            if(Key_Cnt4 < 1.5)//小于 1.5S 是单击
            {
                sys.SetMode_Option++;//设置选项++
                if(sys.Run_Status == 0)//没启动的情况下
                {
                    if(sys.SetMode_Option > 2)//设置选项大于 2
                    {
```

```
                        sys.SetMode_Option = 0;//设置选项清零
                    }
                }
                else//启动的情况下
                {
                    if(sys.SetMode_Option > 1)//设置选项大于 1
                    {
                        sys.SetMode_Option = 0;//设置选项清零
                    }
                }
                Twinkle_Time = 6;//一共闪烁 6S
                Beep_Time = 0.1;//蜂鸣器响 0.1S
            }
            Key_Flag4 = 0;//按键事件结束，等待下一次按下
            LongPress4 = 0;//长按标志清零
            Key_Cnt4 = 0;//按钮计数清零
        }
        if(Key_Cnt4 > 1.5 && Key_Cnt4 < 3)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress4 == 0)//如果没有一直一直长按着
            {
                LongPress4 = 1;//长按标志置一
            }
        }
    }
}
#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(&htim3, TIM_CHANNEL_1);//Motor 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_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
            Speed_Cnt = 0;//计数清零
        }
}

/*
**********************************************************************
 * 函数原型：void TIM3CaptureChannel3Callback(void)
 * 功      能：Tim3 通道 3 的输入捕获回调函数
**********************************************************************
*/
uint32_t Capture, Capture1, Capture2;//捕获和计算的值
uint32_t rel;//实际计算后的速度
void TIM3CaptureChannel1Callback(void)
{
    Capture1 = __HAL_TIM_GET_COMPARE(&htim3, TIM_CHANNEL_1);//输入捕获到的数值
    if(Capture1 > Capture2)//当前的数值大于之前的数时
        Capture = Capture1 - Capture2;//算出捕获到的数
    else//加入当前数值小于之前的数值时，表示溢出了
        Capture = Capture1 + (0xFFFF - Capture2);//用当前数值加上 65535-之前数值
    if(Capture < 100)//过滤
        return;
    #if(Type == 0)
        rel = 60000000 / (Capture * 9)/56;//(60000000us/(捕获到的值*一圈 9 个脉冲))/56 的减速比,RM3100
    #elif(Type == 1)
        rel = (60000000 / (Capture * 9))/30;//(60000000us/(捕获到的值*一圈 9 个脉冲))/30 的减速比,RM4100
    #endif
    Capture2 = Capture1;//当前的值赋值给之前的值，做记录
//  if((rel - Rel_Speed < 50 && rel - Rel_Speed > 0)  || (Rel_Speed - rel < 50 && Rel_Speed - rel > 0))
        Rel_Speed = rel;
    Speed_Cnt = 0;//速度计数清零，用于判断速度是否为 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 == TIM3)//进入 TIM3 的中断
    {
        if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_1)//进入 TIM3 通道一的中断
        {
            TIM3CaptureChannel1Callback();//进入计算速度的函数
```

```
            }
        }
}
#include "Show.h"

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

/**********局部变量声明******/
uint8_t Time_ShowFlag,Speed_ShowFlag;//时间、速度显示的标志位 0:常亮 1：熄灭
uint8_t TimeIcn_ShowFlag,SpeedIcn_ShowFlag;//时间图标闪烁,速度单位图标闪烁
uint8_t Tab[] = {0x77,0x24,0x5D,0x6D,0x2E,0x6B,0x7B,0x25,0x7F,0x6F};//0·9

/*
**********************************************************************
 * 函数原型：  void Check_ShowFlag(float dT)
 * 功      能：  闪烁检测
 * 输      入:  dT:执行周期
 * 参      数：  float dT
**********************************************************************
*/
void Check_ShowFlag(float dT)
{
    static float T,T1;
    if(sys.Run_Status == 1)//系统启动后
    {
        T1 += dT;//闪烁计时
        if(T1 >= 0.5f)//大于 0.5S 后
        {
            if(Set_Time > 0)//只有设定时间大于 0 才闪烁
                TimeIcn_ShowFlag = ~TimeIcn_ShowFlag;//运行闪烁
            else
                TimeIcn_ShowFlag = 0;//时间冒号图标常亮
            SpeedIcn_ShowFlag = ~SpeedIcn_ShowFlag;//速度单位运行闪烁
            T1 = 0;//计时清零
        }
    }
    else//没启动的情况下
    {
        SpeedIcn_ShowFlag = 0;//速度单位图标常亮
        TimeIcn_ShowFlag = 0;//时间冒号图标常亮
        T1 = 0;//计时清零
    }
    if(sys.SetMode_Option == 0 || Key_Status)//如果没在设置选项中或者按键的情况下，则都点亮，不闪烁
    {
        Speed_ShowFlag = 0;//常亮
        Time_ShowFlag = 0;//常亮
        T = 0;//计时清零
        return;
```

```
    }
    if(Twinkle_Time && Key_Status==0)//闪烁和没有操作旋钮时
    {
        T += dT;//闪烁计时
        if(T >= 0.5f)//大于 0.5S 后
        {
            Twinkle_Time -= 0.5;//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)//设置时间
            {
                Speed_ShowFlag = 0;//速度常亮
                Time_ShowFlag = ~Time_ShowFlag;//时间闪烁
            }
            if(Twinkle_Time == 0)//如果闪烁结束
            {
                sys.SetMode_Option = 0;//模式选择清零
            }
            T = 0;//计时清零
        }
    }
}

/*
*****************************************************************
 * 函数原型：  void Display(int16_t speed,int32_t time)
 * 功      能：  显示速度和时间
*****************************************************************
*/
void Display(int16_t speed,int32_t 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;
    uint16_t Val;//用于百十个取出来的数字
    uint8_t SH,H,SM,M;//时间的单位取值

    /***********设定转速计算**********/
    if(Speed_ShowFlag == 0)//设置时闪烁
    {
        /***********speed 千位***********/
        if(speed > 999)//大于 999 时
        {
            Val = speed/1000;//取出千位
            seg1&=0x80;seg1|=Tab[Val];//数字
        }
        else
        {
```

```
            seg1&=0x80;seg1|=0x00;//不显示
        }

        /***********speed 百位***********/
        if(speed > 99)//大于 99 时
        {
            Val=speed/100;//取出百位
            if(speed > 999)//大于 999 时
                Val=Val%10;//取出百位
            seg2&=0x80;seg2|=Tab[Val];//数字
        }
        else
        {
            seg2&=0x80;seg2|=0x00;//不显示
        }
        /***********speed 十位***********/
        if(speed > 9)//大于 9 时
        {
            Val=speed/10;//取出十位
            if(speed > 99)//大于 99 时
                Val=Val%10;//取出十位
            seg3&=0x80;seg3|=Tab[Val];//数字
        }
        else
        {
            seg3&=0x80;seg3|=0x00;//不显示
        }

        /***********speed 个位***********/
        Val=speed%10;//取出个位
        seg4&=0x80;seg4|=Tab[Val];//数字
    }
    else//不显示
    {
        seg1&=0x80;seg1|=0x00;//不显示
        seg2&=0x80;seg2|=0x00;//不显示
        seg3&=0x80;seg3|=0x00;//不显示
        seg4&=0x80;seg4|=0x00;//不显示
    }

    /***********时间计算**********/
    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(Time_ShowFlag == 0)//时间设置时闪烁
    {
        if(Set_Time > 0)//设置时间大于零时显示时间
        {
```

```
            /***********set_time 十小时位***********/
            seg5&=0x80;seg5|=Tab[SH];//数字

            /***********set_time 小时位***********/
            seg6&=0x80;seg6|=Tab[H];//数字

            /***********set_time 十分位***********/
            seg7&=0x80;seg7|=Tab[SM%10];//数字

            /***********set_time 分位***********/
            seg8&=0x80;seg8|=Tab[M%10];//数字
        }
        else//设置时间小于等于 0 时显示“-- --”
        {
            seg5&=0x80;seg5|=0x08;//"-"
            seg6&=0x80;seg6|=0x08;//"-"
            seg7&=0x80;seg7|=0x08;//"-"
            seg8&=0x80;seg8|=0x08;//"-"
        }
    }
    else//时间闪烁
    {
        seg5&=0x80;seg5|=0x00;//不显示
        seg6&=0x80;seg6|=0x00;//不显示
        seg7&=0x80;seg7|=0x00;//不显示
        seg8&=0x80;seg8|=0x00;//不显示
    }
    /*************xg&rpm*********************/
    if(SpeedIcn_ShowFlag == 0)//速度单位闪烁
    {
        seg4&=0x7F;seg4|=0x80;//显示 rpm
    }
    else
    {
        seg4&=0x7F;seg4|=0x00;//不显示 rpm
    }
    /*************时间冒号图标***************/
    if(TimeIcn_ShowFlag == 0)//时间冒号闪烁
    {
        seg6&=0x7F;seg6|=0x80;//显示时间冒号
    }
    else
    {
        seg6&=0x7F;seg6|=0x00;//不显示时间冒号
    }
    /*************时间单位图标***************/
    seg8&=0x7F;seg8|=0x80;//显示 min

    Write_Addr_Dat_N(0,seg1,1);//SEG9
    Write_Addr_Dat_N(2,seg2,1);//SEG10
```

```
    Write_Addr_Dat_N(4,seg3,1);//SEG11
    Write_Addr_Dat_N(6,seg4,1);//SEG12
    Write_Addr_Dat_N(8,seg5,1);//SEG13
    Write_Addr_Dat_N(10,seg6,1);//SEG14
    Write_Addr_Dat_N(12,seg7,1);//SEG15
    Write_Addr_Dat_N(14,seg8,1);//SEG16
}

/*
*****************************************************************
 * 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 = Speed;
            if(Ctrl_Speed > Rel_Speed)//控制速度大于实际速度
            {
                Speed_ADDMode = 1;//进入加速模式下
            }
            else if(Ctrl_Speed <= Display_Speed)//控制速度小于显示速度
            {
                Speed_ADDMode = 2;//进入减速模式下
            }
        }
        if(Speed_ADDMode==1)//在进入加速模式下
        {
            if(Display_Speed< Rel_Speed)
                Display_Speed++;
            if(Display_Speed >= Ctrl_Speed)//实际速度大于等于控制速度
            {
                Speed_ADDMode = 3;//进入稳定模式
                return;
            }
        }
        else if(Speed_ADDMode == 2)//速度下降模式下
        {
            if(Display_Speed>Rel_Speed)
                Display_Speed--;
            if(Rel_Speed <= Ctrl_Speed)//实际速度小于等于控制速度
            {
                Speed_ADDMode = 3;//稳定模式
                return;
            }
```

```
        }
        else if(Speed_ADDMode == 3)//速度稳定模式下
        {
            Display_Speed = Ctrl_Speed;//显示控制速度
        }
        else if(Speed_ADDMode == 4)//速度下降停止模式下
        {
            if(Display_Speed>Rel_Speed)
                Display_Speed--;
            if(Rel_Speed <= 0)//实际速度小于等于 0
            {
                Speed_ADDMode = 3;//稳定模式
                return;
            }
        }
    }
}

/*
**************************************************************************
 * 函数原型：  void Show_Display(void)
 * 功      能：  显示屏幕内容
**************************************************************************
*/
void Show_Display(void)
{
    if(sys.Run_Status == 0)//不启动
    {
        Display_Speed   = Set_Speed;//显示设定速度
        Display_Time    = Set_Time;//显示设定时间
    }
    else//启动后
    {
        if(sys.SetMode_Option == 1)//在设置速度模式下
        {
            Display_Speed   = Set_Speed;//显示设定速度
            Display_Time = Ctrl_Time+59;//显示实际时间
        }
        else if(sys.SetMode_Option == 2)//在设置时间模式下
        {
            Deal_Speed();//速度显示处理
            Display_Time = Set_Time;//显示设定时间
        }
        else//在不设置模式下
        {
            Deal_Speed();//速度显示处理
            Display_Time = Ctrl_Time+59;//显示实际时间
        }
    }
    Display(Display_Speed,Display_Time);//显示屏幕
```

```
}
#include "SetVal.h"

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

/*
*********************************************************************
 * 函数原型：void Check_Set(float dT)
 * 功      能：检测设置
*********************************************************************
*/
void Check_Set(float dT)
{
    if(Key_Status != 0)//按下加减按键
    {
        SetOK_Flag = 1;//检测到波动旋钮，等待退出设置模式
    }
    if(SetOK_Flag == 1)//检测到按下加减按键后
    {
        if(sys.SetMode_Option == 0)//在设定好后
        {
            if(Ctrl_Speed != Set_Speed)//判断控制速度和设定速度是不是不一样
            {
                Ctrl_Speed = Set_Speed;//把设定速度赋值给控制速度
                if(Speed_ADDMode  !=  0)//假如工位只有在启动并且设置了速度的情况下
不等于 0，不在未处理模式下
                    Speed_ADDMode = 0;//进入未处理，判断加速还是减速
                Param.Speed = Set_Speed;//转速
                Speed = Rel_Speed;
                if(sys.Run_Status)
                    Beep_Time = 0.1;
            }
            if(sys.Run_Status == 0)//在没有启动的情况下可以设置时间
            {
                if(Ctrl_Time != Set_Time)//实际时间不等于设定时间
                {
                    Ctrl_Time = Set_Time;//把设定时间赋值给控制时间
                    Param.Time = Set_Time;//时间
                }
            }
            SetOK_Flag = 0;//表示已经设置好了，将设置标志位清零
            Save_Param_En = 1;//保存
        }
    }
}
#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;//保存到 Flash 的标志位

/*
*********************************************************************
 * 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.Time    = 120;//时间
    Param.Speed = 20;//转速
    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();//初始化硬件中的参数
        Set_Time = Param.Time;//时间
        Set_Speed = Param.Speed;//转速
        Save_Param_En = 1;//保存标志位置一
        SetOK_Flag = 1;//设置标志位置一
    }
    else
```

```
    {
        Set_Time = Param.Time;//时间
        Set_Speed = Param.Speed;//转速
        SetOK_Flag = 1;//设置标志位置一
    }

    //保存参数
    if(Save_Param_En)
    {
        Param_Save();//保存到 Flash
        Save_Param_En = 0;//保存标志位置零
    }
}

/*
**********************************************************************
 * 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 T;

    if(Save_Param_En)//加入保存标志位置一
    {
        T += dT;//时间加上 dT

        if(T >= 0.5f)//大于 0.5S 时
        {
            Param_Save();//保存到 Flash
            Save_Param_En = 0;//保存标志位置零
        }
    }
    else
        T = 0;//时间清零
}
#include "PID.h"

/**********结构体************/
PID_val_t Speed_Val;//pid 数据结构
PID_arg_t Speed_Arg;//pid 数据系数

/*
**********************************************************************
 * 函数原型：  void PID_Init(void)
 * 功      能：  pid 系数初始化
**********************************************************************
*/
void PID_Init(void)
```

```
{
    Speed_Arg.Kp=0.0018;
    Speed_Arg.Ki=0.0035;
    Speed_Arg.Kd=0.002;
}

/*
**********************************************************************
 * 函数原型：  void PID_Speed(
              uint16_t Expect,    //期望值（设定值）
              uint16_t Feedback, //反馈值（实际值）
              PID_arg_t *pid_arg,//PID 参数结构体
              PID_val_t *pid_val)//PID 数据结构体
 * 功      能：  PID 控制
 * 输      入:  Expect,    //期望值（设定值）
               Feedback, //反馈值（实际值）
               PID_arg_t *pid_arg,//PID 参数结构体
               PID_arg_t *pid_arg,//PID 参数结构体
**********************************************************************
*/
void PID_Speed(
              uint16_t Expect,    //期望值（设定值）
              uint16_t Feedback, //反馈值（实际值）
              PID_arg_t *pid_arg,//PID 参数结构体
              PID_val_t *pid_val)//PID 数据结构体
{
    pid_val->Error = Expect - Feedback;//当前误差
    if(pid_val->Error > 40)//限幅误差值，这样可以降低上升速度
    {
        pid_val->Error = 40;
    }
    if(pid_val->Error < -90)//限幅误差值，这样可以降低下降速度
    {
        pid_val->Error = -90;
    }
    pid_val->SumError = pid_val->Error + pid_val->SumError;//误差和
    pid_val->D_Error = pid_val->Error - pid_val->LastError;//误差偏差
    pid_val->LastError = pid_val->Error;//保存上一次误差
    pid_val->Out                                                    =
pid_arg->Kp*pid_val->Error+pid_arg->Ki*pid_val->SumError+pid_arg->Kd*pid_val->D_Error;
    if(pid_val->Out<1)
        pid_val->Out=1;
    if(pid_val->Out>1&&pid_val->Out<200)
        pid_val->Out=pid_val->Out;
}
#include "Ctrl_Scheduler.h"

uint16_t   T_cnt_10ms=0,
           T_cnt_50ms=0,
           T_cnt_100ms=0,
```

```
        T_cnt_500ms=0;

void Loop_Check(void)
{
    T_cnt_10ms++;
    T_cnt_50ms++;
    T_cnt_100ms++;
    T_cnt_500ms++;

    Sys_Loop();
}

static void Loop_10ms(void)//10ms 执行一次
{
    Key_Scan(0.01f);//矩阵按键扫描
    Check_Set(0.01f);//检测设置
}

static void Loop_50ms(void)//50ms 执行一次
{
    Check_MotorStop(0.05f);//检测电机是否停止
    Motor_Ctrl(0.05f);//控制速度
    Check_Speed(0.05f);//速度静止检测
}

static void Loop_100ms(void)//100ms 执行一次
{
    Buzzer_Status(0.1f);//蜂鸣器的状态检测
    Cheak_TimeDown(0.1f);//时间倒计时检测
    Check_ShowFlag(0.1f);//闪烁函数
    Param_Save_Overtime(0.1f);//保存标志位置
}

static void Loop_500ms(void)//500ms 执行一次
{
    Check_Press(0.5f);//检测按键按下状态
}

void Sys_Loop(void)
{
    if(T_cnt_10ms >= 10) {
        Loop_10ms();
        T_cnt_10ms = 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_500ms >= 500) {
            Loop_500ms();
            T_cnt_500ms = 0;
        }
}
#include "Ctrl_Motor.h"

/*
**********************************************************************
 * 函数原型：void Motor_Ctrl(float dT)
 * 功      能：电机控制
**********************************************************************
*/
void Motor_Ctrl(float dT)
{
    if(sys.Run_Status == 1)//启动
    {
        if(Ctrl_Speed && ((DownTime_Over == 0)||(Ctrl_Time)))//速度大于 0 和定时器没有
结束
        {
            if(sys.Motor_Stop)//在停止减速模式下
            {
                if(Speed_Val.Out)
                {
                    Speed_Val.Out -= (dT*6);
                }
                PWM = Speed_Val.Out;//PID 输出
            }
            else
            {
                PID_Speed(Ctrl_Speed,Rel_Speed,&Speed_Arg,&Speed_Val);//电机 PID 控
制
//                  if(Speed_Val.Out < 100)
//                      Speed_Val.Out = 100;
                PWM = Speed_Val.Out;//PID 输出
            }
        }
        else
        {
            sys.Motor_Stop = 1;//检测电机
            if(sys.Motor_Stop)//在停止减速模式下
            {
                if(Speed_Val.Out)
                {
                    Speed_Val.Out -= (dT*6);
                }
                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
        }
}

/*
**********************************************************************
 * 函数原型：void Check_MotorStop(float dT)
 * 功      能：检测电机是否停止
**********************************************************************
*/
void Check_MotorStop(float dT)
{
    if(sys.Motor_Stop)//在停止减速模式下
    {
        if(Rel_Speed <= 10)//速度为 10
        {
            SetOK_Flag = 1;//设置参数置一
            sys.Run_Status = 0;//关闭
            sys.Motor_Stop = 0;//电机已经停止
        }
    }
}
#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(Set_Time == 0)//设置时间为 0 的话，不进入计时
        return;
    if(sys.Run_Status && sys.Motor_Stop == 0)//启动系统
    {
        T += dT;//时间加 dT
        if(T >= 1)//1S
        {
            if(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)//控制监视大于 0
                        Ctrl_Time--;//控制时间--
                    else
                    {
                        DownTime_Over= 1;//time1 倒计时结束
                        Speed_ADDMode = 4;//进入减速模式下
                        Beep_Flash = 3;//蜂鸣器响 3 下
                    }
                }
                T = 0;//周期清零
            }
        }
        else
        {
            T = 0;//周期清零
            DownTime_Over = 0;//将倒计时结束的标志位清零
        }
}

#include "System_Init.h"

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

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

    /**********LCD 初始化**************/
    Lcd_Init();

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

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

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

    /**********开机蜂鸣器响************/
    Beep_Time = 0.1;
```

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

#include "stm32f0xx_hal.h"

#define BYTE0(dwTemp)       (*(char *)(&dwTemp))
#define BYTE1(dwTemp)       (*((char *)(&dwTemp) + 1))
#define BYTE2(dwTemp)       (*((char *)(&dwTemp) + 2))
#define BYTE3(dwTemp)       (*((char *)(&dwTemp) + 3))

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 DownTime_Status;//倒计时状态
    uint8_t SetMode_Option;//设置时当前设置的选项
    uint8_t Motor_Stop;//电机停止
}_sys_;
extern _sys_ sys;//系统初始化检测

extern int32_t Speed;//临时速度
extern uint32_t Ctrl_Speed;//控制速度（期望值）
extern int32_t Set_Speed;//设置速度
extern uint32_t Speed_Cnt;//检测转速
extern uint32_t Rel_Speed;//实际速度
extern uint32_t Speed_New;//用于速度显示处理更新
extern uint32_t Speed_Last;//用于速度显示处理存储
extern uint32_t Speed_ADDMode;//用于判断速度时上升还是下降
extern uint32_t Display_Speed;//显示速度

extern int32_t Ctrl_Time;//控制时间（期望值）
extern int32_t Set_Time;//设置时间
extern int32_t Display_Time;//显示控制时间
extern uint8_t DownTime_Over;//时间倒计时结束

#endif
```