LC2100_40 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;
        //Delay_us(10);
        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(&htim14, TIM_CHANNEL_1);
     __HAL_TIM_SET_COMPARE(&htim14, TIM_CHANNEL_1, 50);//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, 50);
}

/*
*********************************************************************
 * 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_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_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,Key_Cnt5,Key_Cnt6;//按下时间
uint8_t Key_Flag1,Key_Flag2,Key_Flag3,Key_Flag4,Key_Flag5,Key_Flag6;//按键按下标志
uint8_t LongPress1,LongPress2,LongPress3,LongPress4,LongPress5,LongPress6;//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)
{
    ROW1_L;
    ROW2_H;
    ROW3_H;
    /*************************************          减          键
**************************************/
    if(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin) == 0)//按下按键
    {
        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(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin) == 1)//抬起按键
        {
            if(Key_Cnt1 < 1.5)//小于 1.5S 是单击
```

```
        {
            if(sys.SetMode_Option == 1)//设置速度
            {
                if(Speed.Speed_Unit)
                {
                    Speed.Set_Speed -= 100;//离心率减 100
                    if(Speed.Set_Speed < Xg_Min)//离心率小于于 Xg_Min 时
                        Speed.Set_Speed = Xg_Min;//离心率等于 Xg_Min
                }
                else
                {
                    Speed.Set_Speed -= 100;//速度减 100
                    if(Speed.Set_Speed < Speed_Min)//速度小于 Speed_Min 时
                        Speed.Set_Speed = Speed_Min;//速度等于 Speed_Min
                }
            }
            else if(sys.SetMode_Option == 2)//设置时间
            {
                if(Time.Time_Unit == 0)
                    Time.Set_Time -= 10;//时间减 10S
                else
                    Time.Set_Time -= 60;//时间减 60S
                if(Time.Set_Time < Time_Min)//设置时间小于 Time_Min
                    Time.Set_Time = Time_Min;//设置时间等于 Time_Min
            }
            Key_Status = 2;
            Twinkle_Time = 6;//闪烁时间 6S
        }
        Key_Flag1 = 0;//按键事件结束，等待下一次按下
        Key_Cnt1 = 0;//按钮计数清零
    }
    if(Key_Cnt1 > 1.9 && Key_Cnt1 < 2.1)//按键时间大于 1.9S 小于 2.1S 表示长按
    {
        if(sys.SetMode_Option == 1)//设置速度
        {
            if(Speed.Speed_Unit)
            {
                Speed.Set_Speed -= 100;//离心率减 100
                if(Speed.Set_Speed < Xg_Min)//离心率小于于 Xg_Min 时
                    Speed.Set_Speed = Xg_Min;//离心率等于 Xg_Min
            }
            else
            {
                Speed.Set_Speed -= 1000;//速度减 1000
                if(Speed.Set_Speed < Speed_Min)//速度小于 Speed_Min 时
                    Speed.Set_Speed = Speed_Min;//速度等于 Speed_Min
            }
        }
        else if(sys.SetMode_Option == 2)//设置时间
        {
```

```
                if(Time.Time_Unit == 0)
                    Time.Set_Time -= 60;//时间减 60S
                else
                    Time.Set_Time -= 600;//时间减 600S
                if(Time.Set_Time < Time_Min)//设置时间小于 Time_Min
                    Time.Set_Time = Time_Min;//设置时间等于 Time_Min
            }
            Key_Status = 2;
            Twinkle_Time = 6;//闪烁时间 6S
            Key_Flag1 = 0;//按键事件结束，等待下一次按下
            Key_Cnt1 = 1.5;//按钮计数清零
        }
    }
    ROW1_H;
    ROW2_L;
    ROW3_H;

    /*************************************MENU 键
**************************************/
    if(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin )== 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(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin) == 1)//抬起按键
        {
            if(Key_Cnt2 < 1.5)//小于 1.5S 是单击
            {
                sys.SetMode_Option++;//设置模式++
                if(sys.SetMode_Option > 2)//退出设置
                    sys.SetMode_Option = 0;//清零
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 6;//闪烁时间 6S
            }
            Key_Flag2 = 0;//按键事件结束，等待下一次按下
            LongPress2 = 0;//长按标志清零
            Key_Cnt2 = 0;//按钮计数清零
        }
        if(Key_Cnt2 > 1.5 && Key_Cnt2 < 3)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress2 == 0)//如果没有一直一直长按着
            {
                if(Speed.Speed_Unit)//假如在离心力的模式下
                {
                    Speed.Speed_Unit = 0;//显示速度单位
                    Speed.Set_Speed = Param.P_Param[1];//设定速度
                    SetOK_Flag = 1;//设置
                }
```

```
                else
                {
                    Speed.Speed_Unit = 1;//显示离心力单位
                    Speed.Set_Speed = Param.P_Param[2];//设定离心力
                    SetOK_Flag = 1;//设置
                }
                sys.SetMode_Option = 0;
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 6;//闪烁时间 6S
                LongPress2 = 1;//长按标志置一
            }
        }
    }

    /*************************************Start 键
*************************************/
    if(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin) == 0)//按下按键
    {
        if(LongPress3 == 0)//没有长按过
        {
            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(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin) == 1)//抬起按键
        {
            if(Key_Cnt3 < 1.5)/*单击*///小于 1.5S 是单击
            {
                if(HAL_GPIO_ReadPin(UC_IN_GPIO_Port,UC_IN_Pin)==  1)//电磁锁 1 闭
合时
                {
                    if(sys.Run_Status == 0)
                    {

                        Speed_Val.SumError = 0x42FE;//启动电 机系数
                        SetOK_Flag = 1;//设定值
                        sys.Run_Status = 1;
                        sys.SetMode_Option = 0;
                        Speed.Speed_ADDMode = 0;
                        Beep_Time = 0.1;//蜂鸣器响 0.1S
                    }
                    else
                    {
                        sys.Motor_Stop = 1;//检测电机
                        Speed.Speed_ADDMode = 2;//进入减速模式下
                        Beep_Time = 0.1;//蜂鸣器响 0.1S
                    }
                }
```

```
                else
                {
                    Beep_Flash = 7;//蜂鸣器响 7 下
                    sys.Lock_On = 1;//开盖图标闪烁标志位
                }
            }
            Key_Flag3 = 0;//按键事件结束，等待下一次按下
            LongPress3 = 0;//长按标志清零
            Key_Cnt3 = 0;//按钮计数清零
        }
        if(Key_Cnt3 > 1.5 && Key_Cnt3 < 3)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress3 == 0)/*长按*///如果没有一直一直长按着
            {

                LongPress3 = 1;//长按标志置一
            }
        }
    }
    ROW1_H;
    ROW2_H;
    ROW3_L;
    /************************************ 加 键
**********************************/
    if(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin )== 0)//按下按键
    {
        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(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin) == 1)//抬起按键
        {
            if(Key_Cnt4 < 1.5)//小于 1.5S 是单击
            {
                if(sys.SetMode_Option == 1)//设置速度
                {
                    if(Speed.Speed_Unit)
                    {
                        Speed.Set_Speed += 100;//离心率加 100
                        if(Speed.Set_Speed > Xg_MAX)//离心率大于 2100 时
                            Speed.Set_Speed = Xg_MAX;//离心率等于 2100
                    }
                    else
                    {
                        Speed.Set_Speed += 100;//速度加 100
                        if(Speed.Set_Speed > Speed_MAX)//速度大于 2500 时
                            Speed.Set_Speed = Speed_MAX;//速度等于 2500
                    }
                }
```

```
                else if(sys.SetMode_Option == 2)//设置时间
                {
                    if(Time.Time_Unit == 0)
                        Time.Set_Time += 10;//时间加 10S
                    else
                        Time.Set_Time += 60;//时间加 60S
                    if(Time.Set_Time > Time_MAX)//设置时间大于 59 分 50 秒时
                        Time.Set_Time = Time_MAX;//设置时间等于 59 分 50 秒时
                }
            }
            Key_Status = 2;
            Twinkle_Time = 6;//闪烁时间 6S
            Key_Flag4 = 0;//按键事件结束，等待下一次按下
            Key_Cnt4 = 0;//按钮计数清零
        }
        if(Key_Cnt4 > 1.9 && Key_Cnt4 < 2.1)//按键时间大于 1.9S 小于 2.1S 表示长按
        {
            if(sys.SetMode_Option == 1)//设置速度
            {
                if(Speed.Speed_Unit)
                {
                    Speed.Set_Speed += 100;//离心率加 100
                    if(Speed.Set_Speed > Xg_MAX)//离心率大于 2100 时
                        Speed.Set_Speed = Xg_MAX;//离心率等于 2100
                }
                else
                {
                    Speed.Set_Speed += 1000;//速度加 100
                    if(Speed.Set_Speed > Speed_MAX)//速度大于 2500 时
                        Speed.Set_Speed = Speed_MAX;//速度等于 2500
                }
            }
            else if(sys.SetMode_Option == 2)//设置时间
            {
                if(Time.Time_Unit == 0)
                    Time.Set_Time += 60;//时间加 10S
                else
                    Time.Set_Time += 600;//时间加 60S
                if(Time.Set_Time > Time_MAX)//设置时间大于 59 分 50 秒时
                    Time.Set_Time = Time_MAX;//设置时间等于 59 分 50 秒时
            }
            Key_Status = 2;
            Twinkle_Time = 6;//闪烁时间 6S
            Key_Flag4 = 0;//按键事件结束，等待下一次按下
            Key_Cnt4 = 1.5;//按钮计数清零
        }
    }
    /*************************************OPEN                                    键
*************************************/
    if(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin) == 0)//按下按键
```

```
    {
        Key_Cnt5 += dT;//按下时间++
        Key_Flag5 = 1;//按键按下标志置一
    }
    if(Key_Flag5 == 1)//按键被按下
    {
        if(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin) == 1)//抬起按键
        {
            if(Key_Cnt5 < 1.5)//小于 1.5S 是单击
            {
                if(HAL_GPIO_ReadPin(UC_IN_GPIO_Port,UC_IN_Pin)==  1)//电磁锁 1 闭
合时
                {
                    Lock_Status = 1;//打开电磁锁 1
                    Beep_Time = 0.1;//蜂鸣器响 0.1S
                    sys.Run_Status = 0;//关闭系统
                }

            }
            Key_Flag5 = 0;//按键事件结束，等待下一次按下
            Key_Cnt5 = 0;//按钮计数清零
        }
        if(Key_Cnt5 > 1.9 && Key_Cnt5 < 2.1)//按键时间大于 1.9S 小于 2.1S 表示长按
        {

            Key_Flag5 = 0;//按键事件结束，等待下一次按下
            Key_Cnt5 = 1.5;//按钮计数清零
        }
    }
}
#include "Drv_Motor.h"

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

/*******************用法*******************/
//Flash_Write((uint8_t *)(&Param),sizeof(Param));
//Flash_Read((uint8_t *)(&Param),sizeof(Param));
/*
*********************************************************************
 * 函数原型：uint8_t Flash_Write(uint8_t *addr, uint16_t len)
```

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

    HAL_FLASH_Unlock();//解锁 Flash

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

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

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

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

/*
**********************************************************************
 * 函数原型：uint8_t Flash_Read(uint8_t *addr, uint16_t len)
 * 功      能：读取 Flash
```

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

/**********全局变量声明******/
uint8_t Lock_Status;//电磁铁的状态
uint8_t Lid_State;//显示图标

/*
**********************************************************************
 * 函数原型：void Ctrl_Lock(float dT)
 * 功      能：电磁铁控制
**********************************************************************
*/
void Ctrl_Lock(float dT)
{
    if(Lock_Status == 1)
    {
        Lock_ON;//打开电磁锁 1
        Lock_Status = 0;
    }
    else
    {
        Lock_OFF;//关闭电磁锁 1
    }
    if((HAL_GPIO_ReadPin(UC_IN_GPIO_Port,UC_IN_Pin)== 1))//电磁锁闭合时
    {
```

```
            Lid_State = 0;//关闭盖子，显示图标
        }
        else
        {
            Lid_State = 1;//打开盖子，显示图标
        }
}
#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  Lock_ShowFlag,TimeIcn_ShowFlag,SpeedIcn_ShowFlag;//开盖图标和时间图标闪烁和速度单位图标闪烁
uint8_t Tab[] = {0x77,0x24,0x5D,0x6D,0x2E,0x6B,0x7B,0x25,0x7F,0x6F};//0·9

/*
**************************************************************************
 * 函数原型：static void Check_ShowFlag(float dT)
 * 功      能：闪烁检测
 * 输      入: dT:执行周期
 * 参      数：float dT
 * 调      用：内部调用
**************************************************************************
*/
static void Check_ShowFlag(float dT)
{
    static float T;
    if(sys.SetMode_Option == 0)//如果没在设置选项中，则都点亮，不闪烁
    {
        Speed_ShowFlag = 0;//常亮
        Time_ShowFlag = 0;//常亮
        Twinkle_Time = 0;//闪烁计时清零
        return;
    }
    if(Twinkle_Time && Key_Status==0)//闪烁和没有操作按键时
    {
        T += dT;
        if(T >= 0.5f)
        {
            Twinkle_Time -= 0.5;//闪烁计时
            if(sys.SetMode_Option == 1)//设置时速度
            {
                Speed_ShowFlag = ~Speed_ShowFlag;//速度常亮
                Time_ShowFlag = 0;//时间常亮
            }
            else if(sys.SetMode_Option == 2)//设置时间
            {
```

```
                    Time_ShowFlag = ~Time_ShowFlag;//时间闪烁
                    Speed_ShowFlag = 0;//速度常亮
                }
                if(Twinkle_Time == 0)//如果闪烁结束
                {
                    sys.SetMode_Option = 0;//模式选择清零
                }
                T = 0;
            }
        }
        else
        {
            Speed_ShowFlag = 0;//常亮
            Time_ShowFlag = 0;//常亮
            T = 0;
        }
}

/*
**********************************************************************
 * 函数原型：static void Time_Twinkle(float dT)
 * 功      能：时间图标闪烁
 * 调      用：内部调用
**********************************************************************
*/
static void Icn_Twinkle(float dT)
{
    static float T;
    if(sys.Run_Status)
    {
        T += dT;
        if(T >= 0.5f)
        {
            TimeIcn_ShowFlag = ~TimeIcn_ShowFlag;//时间图标闪烁;
            SpeedIcn_ShowFlag = ~SpeedIcn_ShowFlag;//速度图标闪烁;
            T = 0;
        }
    }
    else
    {
        TimeIcn_ShowFlag = 0;//显示时间图标
        SpeedIcn_ShowFlag = 0;//显示速度图标;
    }
}

/*
**********************************************************************
 * 函数原型：void Check_Lock(float dT)
 * 功      能：开盖图标闪烁检测
**********************************************************************
```

```
*/
void Check_Lock(float dT)
{
    static float T;
    if(sys.Lock_On == 0)
    {
        Lock_ShowFlag = 0;
        return;
    }
    T += dT;
    if(T >= 0.5f)
    {
        Lock_ShowFlag = ~Lock_ShowFlag;//开盖图标闪烁;
        T = 0;
    }
    if(HAL_GPIO_ReadPin(UC_IN_GPIO_Port,UC_IN_Pin)==  1)//电磁锁 1 和电磁锁 2 都闭合时
        sys.Lock_On = 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
    Check_Lock(dT);//开盖图标闪烁
}

/*
*********************************************************************
 * 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**********/
    if(time >= 60)//如果设定时间大于 1 分钟时
        Time.Time_Unit=1;//单位变成分
```

```
else
    Time.Time_Unit=0;//不然就是秒

SH=time%3600/60/10;//计算十位单位的分钟数
H=time%3600/60%10;//计算个位单位的分钟数
SM=time%60/10;//计算十分位单位的秒钟数
M=time%60%10;//计算十分位单位的秒钟数

if(Time_ShowFlag == 0)//时间设置时闪烁
{
    if(Time.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
{
    if(Speed.Speed_Unit)
    {
        seg5&=0x7F;seg5|=0x80;//显示 xg
    }
    else
    {
        seg4&=0x7F;seg4|=0x80;//显示 rpm
    }
```

```
    }
    else
    {
        seg4&=0x7F;seg4|=0x00;//不显示 rpm
        seg5&=0x7F;seg5|=0x00;//不显示 xg
    }
    /**************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
        seg7&=0x7F;seg7|=0x80;//显示 sec
    }
    else
    {
        seg6&=0x7F;seg6|=0x00;//不显示时间冒号
        seg7&=0x7F;seg7|=0x00;//显示 sec
    }
    /**************时间单位图标***************/
//  if(Time.Time_Unit)//时间单位为分钟时
//  {
//      seg8&=0x7F;seg8|=0x80;//显示 min
//  }
//  else//时间单位为秒时
//  {
        seg7&=0x7F;seg7|=0x80;//显示 sec
//  }

    /**************开盖关盖图标***************/
    if(Lid_State == 0)
    {
        seg2&=0x7F;seg2|=0x80;//显示底部方块
        seg3&=0x7F;seg3|=0x80;//显示关盖
    }
    else
    {
        if(Lock_ShowFlag == 0)//盖子底部
        {
            seg1&=0x7F;seg1|=0x80;//显示开盖
            seg2&=0x7F;seg2|=0x80;//显示盖子底部
        }
        else
        {
            seg1&=0x7F;seg1|=0x00;//不显示开盖
            seg2&=0x7F;seg2|=0x00;//不显示盖子底部
        }
    }

    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)
{
    if(sys.Run_Status == 1)//启动的情况下
    {
        if(Speed.Speed_ADDMode == 0)//在电机控制中，速度未处理
        {
            Speed.Display_Speed = 0;
            Speed.Speed_New =0;//现在的速度清零
            Speed.Speed_Last = 0;//之前的速度清零
            Speed.Speed_ADDMode = 1;//进入加速模式下
        }
        else if(Speed.Speed_ADDMode==1)//在进入加速模式下
        {
            if(Speed.Rel_Speed - Speed.Speed_Last < 400)
            {
                if(Speed.Rel_Speed >= Speed.Ctrl_Speed)//实际速度大于等于控制速度
                {
                    Speed.Speed_ADDMode = 3;//进入稳定模式
                    return;
                }
                Speed.Speed_New = Speed.Rel_Speed;//记录当前速度
                if(Speed.Speed_New > Speed.Speed_Last)//当前速度大于上一次速度
                    Speed.Display_Speed = Speed.Speed_New;//显示当前速度
                else//当前速度小于上一次速度
                {
                    Speed.Display_Speed = 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.Speed_New  =  Speed.Speed_Last;//将上一次速度赋值给当前速
度
                }
                Speed.Speed_Last = Speed.Speed_New;//将当前速度保存
            }
        }
        else if(Speed.Speed_ADDMode==2)//在进入加速模式下
        {
            if(Speed.Speed_Last - Speed.Rel_Speed < 400)
            {
```

```
                    if(Speed.Rel_Speed <= 0)//实际速度小于等于控制速度
                    {
                        Speed.Display_Speed = 0;//进入稳定模式
                        return;
                    }
                    Speed.Speed_New = Speed.Rel_Speed;//记录当前速度
                    if(Speed.Speed_New < Speed.Speed_Last)//当前速度小于上一次速度
                        Speed.Display_Speed = Speed.Speed_New;//显示当前速度
                    else//当前速度大于上一次速度
                    {
                        Speed.Display_Speed = Speed.Speed_Last;//显示上一次速度，不让速度大于当前速度。呈现下降速度的现象
                        Speed.Speed_New = Speed.Speed_Last;//将上一次速度赋值给当前速度
                    }
                    Speed.Speed_Last = Speed.Speed_New;//将当前速度保存
                }
            }
            else if(Speed.Speed_ADDMode == 3)//速度稳定模式下
            {
                Speed.Display_Speed = Speed.Ctrl_Speed;//显示控制速度
            }
        }
        else
        {
            Speed.Display_Speed = 0;
        }
}

/*
*****************************************************************
 * 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)
    {
        Speed.Display_Speed = Speed.Set_Speed;
        Time.Display_Time = Time.Set_Time;
    }
    else
    {
        Deal_Speed();//速度显示处理
        Time.Display_Time = Time.Ctrl_Time;
    }
    Display(Speed.Display_Speed,Time.Display_Time);//显示速度和时间
}
```

```
#include "Speed.h"

/**********局部变量声明******/
uint32_t high,cycle;//高电平时间（us），和周期时间（us）
float frq;//周期频率值
__IO uint32_t TIM1_TIMEOUT_COUNT = 0;//定时器 1 定时溢出计数
uint32_t TIM1_CAPTURE_BUF[3]    = {0, 0, 0};//分别存储上升沿计数、下降沿计数、下个上升沿计数
__IO uint8_t TIM1_CAPTURE_STA = 0;//状态标记

/*
**********************************************************************
 * 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_1);//motor1  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.Speed_Cnt++;//每 50ms 进入
    if(Speed.Speed_Cnt >= 10)//0.5s 发现没出发输入捕获
    {
        Speed.Rel_Speed = 0;//将速度清零
        Speed.Speed_Cnt = 0;//计数清零
    }
}

/*
**********************************************************************
 * 函数原型：void TIM1_SetCapturePolarity(uint32_t TIM_ICPolarity)
 * 功      能：设置 TIM1 输入捕获极性
 * 输      入: TIM_INPUTCHANNELPOLARITY_RISING   ：上升沿捕获
 *             TIM_INPUTCHANNELPOLARITY_FALLING ：下降沿捕获
 *             TIM_INPUTCHANNELPOLARITY_BOTHEDGE：上升沿和下降沿都捕获
**********************************************************************
*/
void TIM1_SetCapturePolarity(uint32_t TIM_ICPolarity)
{
    htim1.Instance->CCER &= ~(TIM_CCER_CC1P | TIM_CCER_CC1NP);
    htim1.Instance->CCER |= (TIM_ICPolarity & (TIM_CCER_CC1P | TIM_CCER_CC1NP));
```

```
}

/*
**********************************************************************
 * 函数原型：void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
 * 功　　能：定时器 1 时间溢出回调函数
**********************************************************************
*/
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == htim1.Instance)
    {
        TIM1_TIMEOUT_COUNT++;//溢出次数计数
    }
}

/*
**********************************************************************
 * 函数原型：void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
 * 功　　能：输入捕获回调函数
**********************************************************************
*/
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == htim1.Instance)
    {
        switch (TIM1_CAPTURE_STA)
        {
        case 1:
            {
                TIM1_CAPTURE_BUF[0]    =    HAL_TIM_ReadCapturedValue(htim,
TIM_CHANNEL_1) + TIM1_TIMEOUT_COUNT * 50;//因为是在高电平段，所以 25 的计算
方式为=（1/(48000000/分频数 12)*计数值 200）/上升沿的原因所以除以 2

TIM1_SetCapturePolarity(TIM_INPUTCHANNELPOLARITY_FALLING);// 设置为下降沿触
发
                TIM1_CAPTURE_STA++;//下一步骤
                break;
            }
        case 2:
            {
                TIM1_CAPTURE_BUF[1]    =    HAL_TIM_ReadCapturedValue(htim,
TIM_CHANNEL_1) + TIM1_TIMEOUT_COUNT * 50;//因为是在低电平段，所以 25 的计算
方式为=（1/(48000000/分频数 12)*计数值 200）/下降沿的原因所以除以 2
                TIM1_SetCapturePolarity(TIM_INPUTCHANNELPOLARITY_RISING);//
设置为上升沿触发
                TIM1_CAPTURE_STA++;//下一步骤
                break;
            }
        case 3:
```

```
            {
                TIM1_CAPTURE_BUF[2] = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1) + TIM1_TIMEOUT_COUNT * 50;//因为是在高电平段，所以 25 的计算方式为=（1/(48000000/分频数 12)*计数值 200）/上升沿的原因所以除以 2
                HAL_TIM_IC_Stop_IT(htim, TIM_CHANNEL_1);//停止捕获
                HAL_TIM_Base_Stop_IT(&htim1);//停止定时器更新中断
                TIM1_CAPTURE_STA++;//下一步骤
                break;
            }
        default:
            break;
        }
    }
}

/*
**********************************************************************
 * 函数原型： void TIM1_Poll(void)
 * 功    能： TIM1 轮训状态切换
**********************************************************************
*/
uint32_t rel;
void TIM1_Poll(void)
{
    switch (TIM1_CAPTURE_STA)
    {
    case 0:
        {
            TIM1_TIMEOUT_COUNT = 0;//溢出清零
            __HAL_TIM_SET_COUNTER(&htim1, 0);//清除定时器 2 现有计数
            memset(TIM1_CAPTURE_BUF, 0, sizeof(TIM1_CAPTURE_BUF));//清除捕获计数
            TIM1_SetCapturePolarity(TIM_INPUTCHANNELPOLARITY_RISING);// 设置为上升沿触发
            HAL_TIM_Base_Start_IT(&htim1);//启动定时器更新中断
            HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_1);//启动捕获中断
            TIM1_CAPTURE_STA++;//下一步骤
            break;
        }
    case 4:
        {
            high  = TIM1_CAPTURE_BUF[1] - TIM1_CAPTURE_BUF[0];//高电平持续时间
            cycle = TIM1_CAPTURE_BUF[2] - TIM1_CAPTURE_BUF[0];//周期
            frq = 1.0 / (((float)cycle) / 1000000.0);//频率计算，用 1S/（周期/1000000.0) ;（周期/1000000.0)为转化单位为 S
            TIM1_CAPTURE_STA = 0;//重新测电平
            if(frq > 80) return;
            rel = 60 * frq;//用一分钟/高电平时间/电机的一圈脉冲
            if((rel - Speed.Rel_Speed < 500 && rel - Speed.Rel_Speed > 0) ||
```

```
(Speed.Rel_Speed - rel < 500 && Speed.Rel_Speed - rel > 0))
                    Speed.Rel_Speed = rel;
                Speed.Speed_Cnt =0;//速度计数清零，用于判断速度是否为 0
                break;
            }
        default:
            break;
        }
}
#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;

/*
*********************************************************************
 * 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.P_Param[0] = 300;//时间
    Param.P_Param[1] = 4000;//转速
    Param.P_Param[2] = 100;//离心率

    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();
        Time.Set_Time = Param.P_Param[0];//时间
        if(Speed.Speed_Unit)
            Speed.Set_Speed = Param.P_Param[2];//离心率
        else
            Speed.Set_Speed = Param.P_Param[1];//转速
        SetOK_Flag = 1;//设置
        Save_Param_En = 1;//保存
    }
    else
    {
        Time.Set_Time = Param.P_Param[0];//时间
        if(Speed.Speed_Unit)
            Speed.Set_Speed = Param.P_Param[2];//离心率
        else
            Speed.Set_Speed = Param.P_Param[1];//转速
        SetOK_Flag = 1;//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"

/**********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=0.0014;
    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(sys.Motor_Stop == 0)
```

```
    {
        if(pid_val->Error > 100)
        {
            pid_val->Error = 100;
        }
    }
    else
    {
        if(pid_val->Error < -200)
        {
            pid_val->Error = -200;
        }
    }
    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<50)
        pid_val->Out=50;
    if(pid_val->Out>50&&pid_val->Out<400)
        pid_val->Out=pid_val->Out;
}
#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)
    {
        SetOK_Flag = 1;//检测到波动旋钮，等待退出设置模式
    }
    if(SetOK_Flag == 1)
    {
        if(sys.SetMode_Option == 0)//在设定好后
        {
            if(Speed.Ctrl_Speed != Speed.Set_Speed)//判断控制速度和设定速度是不是不一
样
            {
                Speed.Ctrl_Speed = Speed.Set_Speed;//把设定速度赋值给控制速度
                if(Speed.Speed_Unit == 0)
                    Param.P_Param[1] = Speed.Set_Speed;//转速
```

```
                    else
                        Param.P_Param[2] = Speed.Set_Speed;//离心率
                }
                if(Time.Ctrl_Time != Time.Set_Time)//实际时间不等于设定时间
                {
                    Time.Ctrl_Time = Time.Set_Time;//把设定时间赋值给控制时间
                    Param.P_Param[0] = Time.Set_Time;//时间
                }
                Save_Param_En = 1;//保存
                SetOK_Flag = 0;
            }
        }
}
#include "MyMath.h"
/*
*****************************************************************
 * 函数原型：float My_Sqrt(float number)
 * 功      能：快速计算开根号的倒数
 * 输      入：被开方数，长整型
 * 输      出：开方结果，整型
 * 参      数：float number
*****************************************************************
*/
float My_Sqrt(float number)
{
    long i;
    float x, y;
    const float f = 1.5F;
    x = number * 0.5F;
    y = number;
    i = * ( long * ) &y;
    i = 0x5f3759df - ( i >> 1 );

    y = * ( float * ) &i;
    y = y * ( f - ( x * y * y ) );
    y = y * ( f - ( x * y * y ) );
    return number * y;
}
#include "Ctrl_Scheduler.h"

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

void Loop_Check(void)
{
    T_cnt_10ms++;
```

```
    T_cnt_20ms++;
    T_cnt_50ms++;
    T_cnt_100ms++;
    T_cnt_200ms++;
    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_20ms(void)//20ms 执行一次
{
    Ctrl_Lock(0.02f);//电磁铁控制
}

static void Loop_50ms(void)//50ms 执行一次
{
    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
    Twinkle(0.1f);//闪烁函数
    Param_Save_Overtime(0.1f);//保存标志位置
}

static void Loop_200ms(void)//200ms 执行一次
{
    Check_MotorStop(0.2f);//检测电机是否停止，停止后开盖
}

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_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;
    }
}
#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((HAL_GPIO_ReadPin(UC_IN_GPIO_Port,UC_IN_Pin)== 1))//电磁锁闭合时
        {
            if(Speed.Ctrl_Speed && ((Time.DownTime_Over == 0)||(Time.Ctrl_Time)))//速度
大于 0 和定时器没有结束
            {
                if(sys.Motor_Stop)
                {
                    PID_Speed(0,Speed.Rel_Speed,&Speed_Arg,&Speed_Val);//电机 PID 控
制
                    PWM = Speed_Val.Out;//pid 输出
                }
                else
                {
                    if(Speed.Speed_Unit)
                        PID_Speed(1000                                          *
My_Sqrt(Speed.Ctrl_Speed/(11.18*96)),Speed.Rel_Speed,&Speed_Arg,&Speed_Val);//电机  PID
控制离心率
```

```
                    else

    PID_Speed(Speed.Ctrl_Speed,Speed.Rel_Speed,&Speed_Arg,&Speed_Val);//电机 PID 控制转速
                        PWM = Speed_Val.Out;//pid 输出
                    }
                }
                else
                {
                    sys.Motor_Stop = 1;//检测电机
                    if(sys.Motor_Stop)
                    {
                        PID_Speed(0,Speed.Rel_Speed,&Speed_Arg,&Speed_Val);//电机 PID 控制
                        PWM = Speed_Val.Out;//pid 输出
                    }
                }
            }
            else
            {
                sys.Run_Status = 0;//不启动
            }
        }
        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，停止后开盖
**********************************************************************
*/
void Check_MotorStop(float dT)
{
    static float T;
    if(sys.Motor_Stop)
    {
        if(Speed.Rel_Speed == 0)
        {
            T += dT;
            if(T>2)
            {
                Lock_Status = 1;//电磁锁打开
                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
```

```
                    T = 0;
                }
            }
            else
            {
                T = 0;
            }
        }
        else
        {
            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
    {
        T += dT;
        if(T >= 1)//1S
        {
            if(Time.DownTime_Over == 0 && Speed.Ctrl_Speed)//如果实际时间显示和倒计
时没有结束的标志还在
            {
                if(Time.Ctrl_Time)
                    Time.Ctrl_Time--;//控制时间--
                else
                {
                    Time.DownTime_Over= 1;//time1 倒计时结束
                    Speed.Speed_ADDMode = 2;//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
        Time.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();//56 不转，57 转

    /**********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();

    /**********蜂鸣器响 0.1S************/
    Beep_Time = 0.1;

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