VM4100_3D 源程序

```
#include "Show.h"

/**********全局变量声明******/
uint32_t Twinkle_Time;//闪烁时间

/**********局部变量声明******/
uint8_t    LCD_ADD[] = {0x5f, 0x06, 0x3d, 0x2f, 0x66, 0x6b, 0x7b, 0x0e, 0x7f, 0x6f};
uint8_t Speed_ShowFlag,Time_ShowFlag,Run_ShowFlag;//速度、时间、运行显示的标志位 0:常亮 1：熄灭

/*
*********************************************************************
 * 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 == 1)
        Run_ShowFlag = ~Run_ShowFlag;//运行闪烁
    else
        Run_ShowFlag = 0;
    if(SetMode_Option == 0 || Key_Status)//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
        return;
    }
    if(Twinkle_Time && Key_Status==0)//闪烁和没有操作旋钮时
    {
        Twinkle_Time -= dT;//闪烁计时
        if(SetMode_Option == 1)//设置速度
        {
            Speed_ShowFlag = ~Speed_ShowFlag;//速度闪烁
            Time_ShowFlag = 0;//时间常亮
        }
        else if(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
        {
            SetMode_Option = 0;//模式选择清零
        }
    }
}
```

```
/*
*********************************************************************
 * 函数原型：  void DiaPlay_Speed(uint8_t speed)
 * 功      能：  显示速度
 * 输      入:  speed ：速度
 * 参      数：  uint8_t speed
*********************************************************************
*/
void DiaPlay_Speed(uint8_t speed)
{
    uint8_t Val;//用于十个取出来的数字
    uint8_t seg2,seg4;
    seg2 = 0; seg4 = 0;
    Val = speed / 10;//取出十位
    seg2 = LCD_ADD[Val];
    Val = speed % 10;//取出个位数
    seg4 = LCD_ADD[Val];

    if(Speed_ShowFlag > 1)//闪烁速度
    {
        seg2 = 0x00;seg4 = 0x00;
    }

    Write_Addr_Dat_N(0x02, seg2, 1);
    Write_Addr_Dat_N(0x04, seg4, 1);
}

/*
*********************************************************************
 * 函数原型：  void DisPlay_Time(uint16_t time)
 * 功      能：  显示时间
 * 输      入:  time ：时间
 * 参      数：  uint16_t time
*********************************************************************
*/
void DisPlay_Time(uint16_t time)
{
    uint8_t T1,T2;//用于显示时间
    uint8_t seg6,seg8,seg10;
    seg6 = 0; seg8 = 0; seg10 = 0;

    if(time > 59)//大于 59 秒时
        Time_Unit = 1;//显示分钟图标
    else
        Time_Unit = 0;//显示秒钟图标

    if(Time_Unit == 1)//单位是分钟时，最大 5940
    {
        T1 = time/60/10;//计算十位的分钟数
        T2 = time/60%10;//计算个位的分钟数
```

```
        seg8&=0xF6;seg8|=0x08;//显示"min"图标
    }
    else
    {
        T1 = time/10;//计算十位的秒钟数
        T2 = time%10;//计算个位的秒钟数
        seg8&=0xF6;seg8|=0x01;//显示"sec"图标
    }
    switch(T2)
    {
        case 0:seg8&=0x8F;seg10&=0x0F;seg8|=0x50;seg10|=0xF0;//数字 0
            break;
        case 1:seg8&=0x8F;seg10&=0x0F;seg8|=0x00;seg10|=0x60;//数字 1
            break;
        case 2:seg8&=0x8F;seg10&=0x0F;seg8|=0x30;seg10|=0xD0;//数字 2
            break;
        case 3:seg8&=0x8F;seg10&=0x0F;seg8|=0x20;seg10|=0xF0;//数字 3
            break;
        case 4:seg8&=0x8F;seg10&=0x0F;seg8|=0x60;seg10|=0x60;//数字 4
            break;
        case 5:seg8&=0x8F;seg10&=0x0F;seg8|=0x60;seg10|=0xB0;//数字 5
            break;
        case 6:seg8&=0x8F;seg10&=0x0F;seg8|=0x70;seg10|=0xB0;//数字 6
            break;
        case 7:seg8&=0x8F;seg10&=0x0F;seg8|=0x00;seg10|=0xE0;//数字 7
            break;
        case 8:seg8&=0x8F;seg10&=0x0F;seg8|=0x70;seg10|=0xF0;//数字 8
            break;
        case 9:seg8&=0x8F;seg10&=0x0F;seg8|=0x60;seg10|=0xF0;//数字 9
            break;
        default:
            break;
    }
    seg6 = LCD_ADD[T1];

    if(Time_State == 0)//显示"--"
    {
        seg6 &= 0X80; seg6 |= 0X20; seg8 &= 0X00; seg8 |= 0X20; seg10 &= 0X00; seg10 |=
0X00;
    }

    if(Time_ShowFlag > 1)//闪烁时间
    {
        seg6 &= 0X80; seg6 |= 0X00; seg8 &= 0X09; seg8 |= 0X00; seg10 &= 0X00; seg10 |=
0X00;
    }
    if(Run_ShowFlag == 0)//运行闪烁
    {
        seg6&=0x7F; seg6|=0x80;//显示"rpm
    }
```

```
    else
    {
        seg6&=0x7F; seg6|=0x00;//不显示"rpm
    }
    Write_Addr_Dat_N(0x06, seg6, 1);
    Write_Addr_Dat_N(0x08, seg8, 1);
    Write_Addr_Dat_N(0x0A, seg10, 1);
}
/*
*********************************************************************
 * 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
        {
            if(Ctrl_Speed >= Display_Speed)//控制速度大于实际速度
            {
                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
            }
            else if(Ctrl_Speed < Display_Speed)//控制速度小于实际速度
            {
                Speed_New=0;//现在的速度清零
                Speed_Last = Display_Speed;//之前的速度等于当前显示速度
                Speed_ADDMode = 2;//进入减速模式下
            }
        }
        if(Speed_ADDMode==1)//在进入加速模式下
        {
            if(Rel_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;
            }
            Speed_New = Rel_Speed;//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 == 2)//速度下降模式下
        {
            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;
            }
            Speed_New = Rel_Speed;//记录当前速度
            if(Speed_New < Speed_Last)//当前速度小于上一次速度
                Display_Speed = Speed_New;//显示当前速度
            else//当前速度大于上一次速度
            {
                Display_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_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 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(SetMode_Option == 1)//在设置速度模式下
        {
            Display_Speed   = Set_Speed;//显示设定速度
            Display_Time = Rel_Time;//显示实际时间
        }
        else if(SetMode_Option == 2)//在设置时间模式下
        {
```

```
                Deal_Speed();
                Display_Time = Set_Time; //显示设定时间
            }
            else//在不设置模式下
            {
                Deal_Speed();
                Display_Time = Rel_Time;//显示实际时间
            }
        }
        DiaPlay_Speed(Display_Speed);
        DisPlay_Time(Display_Time);
}
#include "SetVal.h"

/**********全局变量声明******/
uint8_t SetOK_Flag;//检测是否波动旋钮和设置标志位

/*
*********************************************************************
 * 函数原型：  void Check_Set(void)
 * 功      能：  检测设置
*********************************************************************
*/
void Check_Set(void)
{
    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(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
            }
            if(Rel_Time != Set_Time)//实际时间不等于设定时间
            {
                Rel_Time = Set_Time;//把设定时间赋值给控制时间
            }
            SetOK_Flag = 0;
        }
    }
}
#include "Speed.h"
```

```
/**********局部变量声明******/
uint32_t P_Status;//捕获周期计数状态   1 开启  0 关闭
uint16_t TIM1CH1_CAPTURE_STA=0;//捕获溢出的周期数
uint32_t TIM1CH1_CAPTURE_VAL;//捕获未溢出的计数值
uint8_t CAPTURE_First=0;//捕获第一个高电平
uint8_t CAPTURE_Status=0;//捕获状态
uint16_t Speed_Flag;//速度调 0 标志位

/*
**********************************************************************
 * 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_3);//motor 输入捕获
    HAL_TIM_Base_Start_IT(&htim1);//开启定时器 1 的中断
}

/*
**********************************************************************
 * 函数原型：  void Check_Speed(void)
 * 功      能：  检测速度是否停止
**********************************************************************
*/
void Check_Speed(void)
{
    if(Speed_Flag)
        Speed_Flag--;
    if(Speed_Flag==0)
        Rel_Speed=0;
}

/*
**********************************************************************
 * 函数原型：  void Check_Status(void)
 * 功      能：  检测捕获状态
**********************************************************************
*/
void Check_Status(void)
{
    if(CAPTURE_Status)//捕获结束
    {
        __HAL_TIM_ENABLE(&htim1);//重新开始捕获
        CAPTURE_Status=0;//开始捕获
        TIM1CH1_CAPTURE_STA=0;//溢出时间清零
    }
}
```

```
/*
**********************************************************************
 * 函数原型：  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
 * 功      能：  定时器中断
**********************************************************************
*/
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance == TIM1)
    {
        if(P_Status)//捕获周期计数
        {
            TIM1CH1_CAPTURE_STA++;//溢出加 1
        }
    }
}

/*
**********************************************************************
 * 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(CAPTURE_Status==0)
    {
         Speed_Flag=2;//每次进入都赋值 2，如果 2S 后不进入表示速度为 0
        if(CAPTURE_First)
        {
            CAPTURE_Status=1;//停止捕获计时
            CAPTURE_First=0;//清除捕获第一个上升沿标志

    TIM1CH1_CAPTURE_VAL=HAL_TIM_ReadCapturedValue(&htim1,TIM_CHANNEL_3);
//获取当前捕获计数值
            long long temp=0;
            temp=TIM1CH1_CAPTURE_STA;//溢出的次数（用于计算进入多少个周期）
            temp*=100;//一个周期溢出的时间（us）计算=定时器周期（1/（48000000/12）
*200）=0.0001S=100us
            temp+=TIM1CH1_CAPTURE_VAL;//一个周期所需的 us 数(用溢出的时间加上
没有溢出的时间）得出一个周期用了多少 us
            temp=60000000/temp/9/56;//一分钟有 60000000us，一分钟内有多少个脉冲（周
期）。9 是一圈有几个脉冲得出一分钟转了多少圈。56 是减速比
            Rel_Speed=temp;
            P_Status=0;//捕获周期计数关闭
            __HAL_TIM_SET_COUNTER(&htim1,0);//定时器寄存器清零
            __HAL_TIM_DISABLE(&htim1);//不进入定时器中断（不溢出计数）
        }
        else
        {
```

```
                TIM1CH1_CAPTURE_STA=0;//清除周期计数
                TIM1CH1_CAPTURE_VAL=0;//清除捕获寄存器
                CAPTURE_First=1;//已捕获第一个上升沿
                CAPTURE_Status=0;//捕获计时
                P_Status=1;//捕获周期计数开始
            }
        }
}
#include "Ctrl_Scheduler.h"

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

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

    Sys_Loop();
}

static void Loop_2ms(void)//2ms 执行一次
{
    Check_Set();//检测设置
}

static void Loop_10ms(void)//10ms 执行一次
{
    Key_Handle();
    Motor_Ctrl();//电机控制
}

static void Loop_50ms(void)//50ms 执行一次
{

}

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

```
    Buzzer_Status(0.1);
}

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_Key();//检测按键是否还在按
    Check_Speed();//检测速度是否停止
}

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_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;

    /**********背光源亮度控制**********/
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_2,30);//Set_Speed);//pwm
0—400

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

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

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

    /**********电调初始化**********/
    Encoder_Init();

    /**********数值初始化**********/
    Motor1 = 0;
    Set_Speed = 20;
    Ctrl_Speed = Set_Speed;
    Beep_Time = 0.1;//蜂鸣器响 0.1S

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

```
extern _sys_ sys;//系统初始化检测

extern int16_t Speed;//临时速度
extern int16_t Set_Speed;//设定速度
extern int16_t Rel_Speed;//实际速度
extern int16_t Ctrl_Speed;//控制速度
extern int16_t Display_Speed;//显示速度
extern int16_t Speed_Cnt;//没进入输入捕获的时间
extern uint8_t Speed_ADDMode;//用于判断速度时上升还是下降
extern int16_t Speed_New;//用于速度显示处理更新
extern int16_t Speed_Last;//用于速度显示处理存储

extern int16_t Time;//临时时间
extern int16_t Set_Time;//设定时间
extern int16_t Rel_Time;//实际时间
extern int16_t Display_Time;//显示时间
extern uint8_t Time_Unit;//时间分钟秒钟切换
extern uint8_t Time_State;//时间的状态
extern uint8_t DownTime_Over;//时间倒计时结束



#endif
#include "Drv_KEY.h"

/**********全局变量**********/
uint8_t Key_Status;//在操作按键时
uint8_t SetMode_Option;//选择设置模式

/**********局部变量**********/
uint16_t cur=300;//连续按加快加减速度
uint16_t Scan_Status=0;//快速加减标志
uint8_t KEY1_Pin_ON=0;//长按标志

/*
*****************************************************************
 * 函数原型：  static uint8_t Key_Scan(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
 * 功    能：  按键扫描
 * 输    入:  *GPIOx：gipo 管脚  GPIO_Pin：引脚
 * 输    出：  KEY_ON/KEY_OFF
 * 参    数：  GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin
 * 调    用：  内部调用
*****************************************************************
*/
static uint8_t Key_Scan(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
{
    if(HAL_GPIO_ReadPin (GPIOx, GPIO_Pin) == 0)//按键按下
    {
        uint32_t cur_time = HAL_GetTick();//相当于延时 8ms
        static uint32_t start_time = 0;
```

```
            if(cur_time - start_time < cur)
                return KEY_OFF;
            if(HAL_GPIO_ReadPin (GPIOx, GPIO_Pin) == 0)
            {
                Scan_Status++;
                if(Scan_Status > 3)//一直按着的时间
                    cur = 2;
                start_time = cur_time;
                return KEY_ON;
            }
        }
        else//松开按键后
        {
            if((HAL_GPIO_ReadPin (GPIOB, KEY2_Pin) == 1) && (HAL_GPIO_ReadPin
(GPIOB, KEY3_Pin) == 1 ) && (HAL_GPIO_ReadPin (GPIOB, KEY1_Pin) == 1 ) )
            {
                Scan_Status = 0;
                cur = 300;
                return KEY_OFF;
            }
        }
        return KEY_OFF;
}

/*
**********************************************************************
 * 函数原型：  void Key_Handle(void)
 * 功      能：  按键功能
**********************************************************************
*/
void Key_Handle(void)
{
/*******************************减*****************************************/
        if((Key_Scan(GPIOB,KEY3_Pin) == KEY_ON))//减
        {
            if(SetMode_Option == 1)//在设置速度时
            {
                Set_Speed -=1;
                if(Set_Speed < 10)
                    Set_Speed = 10;
            }
            if(SetMode_Option == 2)//在设置时间时
            {
                if(Set_Time <= 60)
                {
                    Set_Time -= 1;
                }
                else
                {
                    Set_Time -= 60;
```

```
            }
            if(Set_Time < 1)
            {
                Set_Time = 0;
                Time_State = 0;
            }
        }
        Key_Status = 1;
        Twinkle_Time = 6000;
    }

/*****************************加*****************************************/
    if((Key_Scan(GPIOB,KEY2_Pin) == KEY_ON))//加
    {
        if(SetMode_Option == 1)//在设置速度时
        {
            Set_Speed +=1;
            if(Set_Speed > 80)
                Set_Speed = 80;
        }
        if(SetMode_Option == 2)//在设置时间时
        {
            if(Time_State == 0)
            {
                Set_Time += 60;
                Time_State = 1;
            }
            else if(Time_Unit == 0)
            {
                Set_Time += 1;
                Time_State = 1;
            }
            else
            {
                Set_Time += 60;
            }
            if(Set_Time>5940)
            {
                Set_Time = 5940;
            }
        }
        Key_Status = 1;
        Twinkle_Time = 6000;
    }

/*****************************菜单键*****************************************/
    if((Key_Scan(GPIOB,KEY1_Pin) == KEY_ON))//菜单键
    {
        SetMode_Option++;
        if(SetMode_Option > 2)
```

```
        {
            SetMode_Option = 0;

        }
        Twinkle_Time = 6000;
        Beep_Time = 0.1;//蜂鸣器响 0.1S
    }

/*******************************开始/停止*******************************/
    if((Key_Scan(GPIOB,KEY4_Pin) == KEY_ON))//开始/停止
    {
        if(sys.Run_Status ==0)
        {
            sys.Run_Status = 1;
            Speed_ADDMode = 0;//屏幕显示模式
            Speed_Val.SumError = 0x23fff;//启动的脉冲，能到 10rpm
            Speed_New=0;//现在的速度清零
            Speed_Last = 0;//之前的速度等于当前显示速度
        }
        else
        {
            sys.Run_Status = 0;
        }
        SetMode_Option = 0;
        Beep_Time = 0.1;//蜂鸣器响 0.1S
    }
}

/*
*********************************************************************
 * 函数原型：void Check_Key(void)
 * 功      能：检测按键状态-1s
*********************************************************************
*/
void Check_Key(void)
{
    if(Key_Status)
        Key_Status--;
}
/**
```