HS3100 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(&htim1, TIM_CHANNEL_2);
	__HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2, 20);//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_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)
{
        /***********************************MENU                                    键
***************************************/
        if(HAL_GPIO_ReadPin(KEY_MENU_GPIO_Port,KEY_MENU_Pin) == 0)//按下按键
        {
                if(sys.Run_Status)
                        return;
                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(HAL_GPIO_ReadPin(KEY_MENU_GPIO_Port,KEY_MENU_Pin) == 1)//抬起按键
        {
            if(Key_Cnt1 > 0.1 && Key_Cnt1 < 1.5)//小于 1.5S 是单击
            {
                sys.SetMode_Option++;
                if(sys.SetMode_Option > 3)
                    sys.SetMode_Option = 0;
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 6;//闪烁时间 6S
            }
            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)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress1 == 0)//如果没有一直一直长按着
            {
                LongPress1 = 1;//长按标志置一
            }
        }
    }
    /***************************************                 加                  键
***************************************/
    if(HAL_GPIO_ReadPin(KEY_PLUS_GPIO_Port,KEY_PLUS_Pin )== 0)//按下按键
    {
        if(sys.Run_Status)
            return;
        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(KEY_PLUS_GPIO_Port,KEY_PLUS_Pin) == 1)//抬起按键
        {
            if(Key_Cnt2 < 1.4)//小于 1.5S 是单击
            {
                if(sys.SetMode_Option == 1)//设置温度
                {
                    Temp.Set_Temp += 10;//温度加 1 度
                    if(Temp.Set_Temp > Temp_MAX)//假如温度大于 Temp_MAX 度时
                        Temp.Set_Temp = Temp_MAX;//温度等于 Temp_MAX 度
                }
                else if(sys.SetMode_Option == 2)//设置速度
                {
                    Speed.Set_Speed += 10;//转速加 10 转
                    if(Speed.Set_Speed == 10)//从零转开始最低为 50 转，判断是 10 后
                        Speed.Set_Speed = 50;//设定转速为 50 开始
                    if(Speed.Set_Speed > Speed_MAX)//假如转速大于 Speed_MAX
                        Speed.Set_Speed = Speed_MAX;//转速等于 Speed_MAX
                }
                else if(sys.SetMode_Option == 3)//设置时间
                {
                    Time.Set_Time += 60;//时间加 1 分钟
                    if(Time.Set_Time > Time_MAX)//假如时间大于 Time_MAX 时
                        Time.Set_Time = Time_MAX;//时间等于 Time_MAX
                }
                Key_Status = 2;//设置时 2S 不闪烁
                Twinkle_Time = 6;//闪烁时间 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)//设置温度
            {
                Temp.Set_Temp += 100;//温度加 10 度
                if(Temp.Set_Temp > Temp_MAX)//假如温度大于 Temp_MAX 度时
                    Temp.Set_Temp = Temp_MAX;//温度等于 Temp_MAX 度
            }
            else if(sys.SetMode_Option == 2)//设置速度
            {
                Speed.Set_Speed += 100;//转速加 100 转
                if(Speed.Set_Speed > Speed_MAX)//假如转速大于 Speed_MAX
                    Speed.Set_Speed = Speed_MAX;//转速等于 Speed_MAX
            }
            else if(sys.SetMode_Option == 3)//设置时间
            {
                Time.Set_Time += 600;//时间加 10 分钟
                if(Time.Set_Time > Time_MAX)//假如时间大于 Time_MAX 时
                    Time.Set_Time = Time_MAX;//时间等于 Time_MAX
            }
            Key_Status = 2;//设置时 2S 不闪烁
            Twinkle_Time = 6;//闪烁时间 6S
            Key_Flag2 = 0;//按键事件结束，等待下一次按下
            Key_Cnt2 = 1.4;//按钮计数从 1.4s 开始
        }
    }

    /**************************************          减          键
***************************************/
    if(HAL_GPIO_ReadPin(KEY_DEC_GPIO_Port,KEY_DEC_Pin) == 0)//按下按键
    {
        if(sys.Run_Status)
            return;
        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(KEY_DEC_GPIO_Port,KEY_DEC_Pin) == 1)//抬起按键
    {
        if(Key_Cnt3 < 1.4)/*单击*///小于 1.5S 是单击
        {
            if(sys.SetMode_Option == 1)//设置温度
            {
                Temp.Set_Temp -= 10;//温度减 1 度
                if(Temp.Set_Temp < 0)//假如温度小于 0 度时
                    Temp.Set_Temp = 0;//温度等于 0 度
            }
            else if(sys.SetMode_Option == 2)//设置速度
            {
                Speed.Set_Speed -= 10;//转速减 10 转
                if(Speed.Set_Speed < 50)//假如转速小于 50 时
                    Speed.Set_Speed = 0;//转速等于 0
            }
            else if(sys.SetMode_Option == 3)//设置时间
            {
                Time.Set_Time -= 60;//时间减 1 分钟
                if(Time.Set_Time < 0)//假如时间小于 0 时
                    Time.Set_Time = 0;//时间等于 0
            }
            Key_Status = 2;//设置时 2S 不闪烁
            Twinkle_Time = 6;//闪烁时间 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 1.9S 小于 2.1S 表示长按
    {
        if(sys.SetMode_Option == 1)//设置温度
        {
            Temp.Set_Temp -= 100;//温度减 10 度
            if(Temp.Set_Temp < 0)//假如温度小于 0 度时
                Temp.Set_Temp = 0;//温度等于 0 度
        }
        else if(sys.SetMode_Option == 2)//设置速度
        {
            Speed.Set_Speed -= 100;//转速减 100 转
            if(Speed.Set_Speed < 50)//假如转速小于 50 时
                Speed.Set_Speed = 0;//转速等于 0
        }
        else if(sys.SetMode_Option == 3)//设置时间
        {
            Time.Set_Time -= 600;//时间减 10 分钟
            if(Time.Set_Time < 0)//假如时间小于 0 时
                Time.Set_Time = 0;//时间等于 0
                }
                Key_Status = 2;//设置时 2S 不闪烁
                Twinkle_Time = 6;//闪烁时间 6S
                Key_Flag3 = 0;//按键事件结束，等待下一次按下
                Key_Cnt3 = 1.4;//按钮计数从 1.5s 开始
            }
        }

        /***************************************Start                                    键
****************************************/
        if(HAL_GPIO_ReadPin(KEY_STATE_GPIO_Port,KEY_STATE_Pin )== 0)//按下按键
        {
            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(HAL_GPIO_ReadPin(KEY_STATE_GPIO_Port,KEY_STATE_Pin) == 1)//抬起按键
            {
                if(Key_Cnt4 > 0.1 && Key_Cnt4 < 1.5)//小于 1.5S 是单击
                {
                    if(sys.Run_Status == 0 && (Speed.Set_Speed || Temp.Set_Temp))//系统没启
动的话
                    {
                        sys.Run_Status = 1;//启动系统
                        Speed_Val.SumError = 0x12E58;//给 pid 的积分和来起步电机
                        sys.SetMode_Option = 0;
                    }
                    else//系统启动的话
                    {
                        Speed.Speed_ADDMode = 1;//进入减速模式
                        Speed.Ctrl_Speed = 0;//将控制速度设置为 0
                    }
                    Beep_Time = 0.1;//蜂鸣器响 0.1S
                    Twinkle_Time = 0;//闪烁时间 6S
                    sys.SetMode_Option = 0;
                }
                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 "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_PT1000.h"
/*
-22.1-300 摄氏度
*/
const float Temp_map[1421]=
{
/*-22*/913.733,   913.340,   912.946,   912.553,   912.159,   911.766,  911.372,
        910.979,   910.585,   910.192,
/*-21*/917.666,   917.273,   916.879,   916.486,   916.093,   915.700,
        915.306,   914.913,   914.520,   914.126,
/*-20*/921.599,   921.206,   920.812,   920.419,   920.026,   919.633,
        919.239,   918.846,   918.453,   918.059,
/*-19*/925.531,   925.138,   924.745,   924.351,   923.958,   923.565,
        923.172,   922.779,   922.385,   921.992,
/*-18*/929.460,   929.067,   928.674,   928.281,   927.888,   927.496,
        927.103,   926.710,   926.317,   925.924,
/*-17*/933.390,   932.997,   932.604,   932.211,  931.818,   931.425,   931.032,
        930.639,   930.246,   929.853,
/*-16*/937.317,   936.924,   936.532,   936.139,   935.746,   935.354,
          934.961,      934.568,      934.175,      933.783,
/*-15*/941.244,   940.851,      940.459,      940.066,      939.673,      939.281,
          938.888,      938.495,      938.102,      937.710,
/*-14*/945.170,   944.777,      944.385,      943.992,      943.600,      943.207,
          942.814,      942.422,      942.029,      941.637,
/*-13*/949.094,   948.702,      948.309,      947.917,      947.524,      947.132,
          946.740,      946.347,      945.955,      945.562,
/*-12*/953.016,   952.624,      952.232,      951.839,      951.447,      951.055,  950.663,
          950.271,  949.878,  949.486,
/*-11*/956.938,   956.546,      956.154,      955.761,      955.369,      954.977,  954.585,
          954.193,  953.800,  953.408,
/*-10*/960.859,   960.467,      960.075,      959.683,      959.291,      958.899,  958.506,
          958.114,  957.722,  957.330,
/*-9*/964.779,    964.387,      963.995,      963.603,      963.211,  962.819,  962.427,
          962.035,  961.643,  961.251,
/*-8*/968.697,    968.305,      967.913,      967.522,      967.130,      966.738,  966.346,
          965.954,  965.563,  965.171,
/*-7*/972.614,    972.222,      971.831,      971.439,      971.047,      970.656,  970.264,
          969.872,  969.480,  969.089,
/*-6*/976.529,    976.138,      975.746,      975.355,      974.963,      974.572,  974.180,
          973.789,  973.397,  973.006,
/*-5*/980.444,    980.053,      979.662,      979.270,      978.879,      978.487,  978.096,
          977.704,  977.313,  976.921,
/*-4*/984.358,    983.967,      983.575,      983.184,      982.793,      982.401,  982.010,
          981.618,  981.227,  980.835,
/*-3*/988.270,    987.879,      987.488,      987.096,      986.705,      986.314,  985.923,
          985.532,  985.140,  984.749,
/*-2*/992.181,991.790,  991.399,  991.008,  990.617,  990.226,  989.834,  989.443,  989.052,
          988.661,
/*-1*/996.091,995.700,  995.309,  994.918,  994.527,  994.136,      993.745,      993.354,
          992.963,      992.572,
/*0*/1000.000,      1000.391,     1000.782,     1001.172,     1001.563,     1001.954,
          1002.345,     1002.736,     1003.126,     1003.517,
/*1*/1003.908,      1004.298,     1004.689,     1005.080,     1005.470,     1005.861,
          1006.252,     1006.642,     1007.033,     1007.424,
/*2*/1007.814,      1008.205,     1008.595,     1008.986,     1009.377,     1009.767,
          1010.158,     1010.548,     1010.939,     1011.329,
/*3*/1011.720,      1012.110,     1012.501,     1012.891,     1013.282,     1013.672,
          1014.062,     1014.453,     1014.843,     1015.234,
1015.624,     1016.014,     1016.405,     1016.795,     1017.185,     1017.576,
          1017.966,     1018.356,     1018.747,     1019.137,
1019.527,     1019.917,     1020.308,     1020.698,     1021.088,     1021.478,
          1021.868,     1022.259,     1022.649,     1023.039,
1023.429,     1023.819,     1024.209,     1024.599,     1024.989,     1025.380,
          1025.770,     1026.160,     1026.550,     1026.940,
1027.330,     1027.720,     1028.110,     1028.500,     1028.890,     1029.280,
          1029.670,     1030.060,     1030.450,     1030.840,
1031.229,     1031.619,     1032.009,     1032.399,     1032.789,     1033.179,
          1033.569,     1033.958,     1034.348,     1034.738,
1035.128,     1035.518,     1035.907,     1036.297,     1036.687,     1037.077,
    1037.466,   1037.856,   1038.246,   1038.636,
1039.025,   1039.415,   1039.805,   1040.194,   1040.584,   1040.973,
    1041.363,   1041.753,   1042.142,   1042.532,
1042.921,   1043.311,   1043.701,   1044.090,   1044.480,   1044.869,
    1045.259,   1045.648,   1046.038,   1046.427,
1046.816,   1047.206,   1047.595,   1047.985,   1048.374,   1048.764,
    1049.153,   1049.542,   1049.932,   1050.321,
1050.710,   1051.099,   1051.489,   1051.878,   1052.268,   1052.657,
    1053.046,   1053.435,   1053.825,   1054.214,
1054.603,   1054.992,   1055.381,   1055.771,   1056.160,   1056.549,
    1056.938,   1057.327,   1057.716,   1058.105,
1058.495,   1058.884,   1059.273,   1059.662,   1060.051,   1060.440,
    1060.829,   1061.218,   1061.607,   1061.996,
1062.385,   1062.774,   1063.163,   1063.552,   1063.941,   1064.330,
    1064.719,   1065.108,   1065.496,   1065.885,
1066.274,   1066.663,   1067.052,   1067.441,   1067.830,   1068.218,
    1068.607,   1068.996,   1069.385,   1069.774,
1070.162,   1070.551,   1070.940,   1071.328,   1071.717,   1072.106,
    1072.495,   1072.883,   1073.272,   1073.661,
1074.049,   1074.438,   1074.826,   1075.215,   1075.604,   1075.992,
    1076.381,   1076.769,   1077.158,   1077.546,
1077.935,   1078.324,   1078.712,   1079.101,   1079.489,   1079.877,
    1080.266,   1080.654,   1081.043,   1081.431,
1081.820,   1082.208,   1082.596,   1082.985,   1083.373,   1083.762,
    1084.150,   1084.538,   1084.926,   1085.315,
1085.703,   1086.091,   1086.480,   1086.868,   1087.256,   1087.644,
    1088.033,   1088.421,   1088.809,   1089.197,
1089.585,   1089.974,   1090.362,   1090.750,   1091.138,   1091.526,
    1091.914,   1092.302,   1092.690,   1093.078,
1093.467,   1093.855,   1094.243,   1094.631,   1095.019,   1095.407,
    1095.795,   1096.183,   1096.571,   1096.959,
1097.347,   1097.734,   1098.122,   1098.510,   1098.898,   1099.286,
    1099.674,   1100.062,   1100.450,   1100.838,
1101.225,   1101.613,   1102.001,   1102.389,   1102.777,   1103.164,
    1103.552,   1103.940,   1104.328,   1104.715,
1105.103,   1105.491,   1105.879,   1106.266,   1106.654,   1107.042,
    1107.429,   1107.817,   1108.204,   1108.592,
1108.980,   1109.367,   1109.755,   1110.142,   1110.530,   1110.917,
    1111.305,   1111.693,   1112.080,   1112.468,
1112.855,   1113.242,   1113.630,   1114.017,   1114.405,   1114.792,
    1115.180,   1115.567,   1115.954,   1116.342,
1116.729,   1117.117,   1117.504,   1117.891,   1118.279,   1118.666,
    1119.053,   1119.441,   1119.828,   1120.215,
1120.602,   1120.990,   1121.377,   1121.764,   1122.151,   1122.538,
    1122.926,   1123.313,   1123.700,   1124.087,
1124.474,   1124.861,   1125.248,   1125.636,   1126.023,   1126.410,
    1126.797,   1127.184,   1127.571,   1127.958,
1128.345,   1128.732,   1129.119,   1130.127,   1129.893,   1130.280,
    1130.667,   1131.054,   1131.441,   1131.828,
1132.215,   1132.602,   1132.988,   1133.375,   1133.762,   1134.149,
        1134.536,    1134.923,    1135.309,    1135.696,
1136.083,    1136.470,    1136.857,    1137.243,    1137.630,    1138.017,
        1138.404,    1138.790,    1139.177,    1139.564,
1139.950,    1140.337,    1140.724,    1141.110,    1141.497,    1141.884,
        1142.270,    1142.657,    1143.043,    1143.430,
1143.817,    1144.203,    1144.590,    1144.976,    1145.363,    1145.749,
        1146.136,    1146.522,    1146.909,    1147.295,
1147.681,    1148.068,    1148.454,    1148.841,    1149.227,    1149.614,
        1150.000,    1150.386,    1150.773,    1151.159,
1151.545,    1151.932,    1152.318,    1152.704,    1153.091,    1153.477,
        1153.863,    1154.249,    1154.636,    1155.022,
1155.408,    1155.794,    1156.180,    1156.567,    1156.953,    1157.339,
        1157.725,    1158.111,    1158.497,    1158.883,
1159.270,    1159.656,    1160.042,    1160.428,    1160.814,    1161.200,
        1161.586,    1161.972,    1162.358,    1162.744,
1163.130,    1163.516,    1163.902,    1164.288,    1164.674,    1165.060,
        1165.446,    1165.831,    1166.217,    1166.603,
1166.989,    1167.375,    1167.761,    1168.147,    1168.532,    1168.918,
        1169.304,    1169.690,    1170.076,    1170.461,
1170.847,    1171.233,    1171.619,    1172.004,    1172.390,    1172.776,
        1173.161,    1173.547,    1173.933,    1174.318,
1174.704,    1175.090,    1175.475,    1175.861,    1176.247,    1176.632,
        1177.018,    1177.403,    1177.789,    1178.174,
1178.560,    1178.945,    1179.331,    1179.716,    1180.102,    1180.487,
        1180.873,    1181.258,1181.644,    1182.029,
1182.414,    1182.800,    1183.185,    1183.571,    1183.956,    1184.341,
        1184.727,    1185.112,1185.597,    1185.883,
1186.268,    1186.653,    1187.038,    1187.424,    1187.809,    1188.194,
        1188.579,    1188.965,1189.350,    1189.735,
1190.120,    1190.505,    1190.890,    1191.276,    1191.661,    1192.046,
        1192.431,    1192.816,    1193.201,    1193.586,
1193.971,    1194.356,    1194.741,    1195.126,    1195.511,    1195.896,
        1196.281,    1196.666,    1197.051,    1197.436,
1197.821,    1198.206,    1198.591,    1198.976,    1199.361,    1199.746,
        1200.131,    1200.516,    1200.900,    1201.285,
1201.670,    1202.055,    1202.440,    1202.824,    1203.209,    1203.594,
        1203.979,    1204.364,    1204.748,    1205.133,
1205.518,    1205.902,    1206.287,    1206.672,    1207.056,    1207.441,
        1207.826,    1208.210,    1208.595,    1208.980,
1209.364,    1209.749,    1210.133,    1210.518,    1210.902,    1211.287,
        1211.672,    1212.056,    1212.441,    1212.825,
1213.210,    1213.594,    1213.978,    1214.363,    1214.747,    1215.120,
        1215.516,    1215.901,    1216.285,    1216.669,
1217.054,    1217.438,    1217.822,    1218.207,    1218.591,    1218.975,
        1219.360,    1219.744,    1220.128,    1220.513,
1220.897,    1221.281,    1221.665,    1222.049,    1222.434,    1222.818,
        1223.202,    1223.586,    1223.970,    1224.355,
1224.739,    1225.123,    1225.507,    1225.891,    1226.275,    1226.659,
        1227.043,    1227.427,    1227.811,    1228.195,
1228.579,    1228.963,    1229.347,    1229.731,    1230.115,    1230.499,
```

1230.883, 1231.267, 1231.651, 1232.035,
1232.419, 1232.803, 1233.187, 1233.571, 1233.955, 1234.338,
1234.722, 1235.106, 1235.490, 1235.874,
1236.257, 1236.641, 1237.025, 1237.409, 1237.792, 1238.176,
1238.560, 1238.944, 1239.327, 1239.711,
1240.095, 1240.478, 1240.862, 1241.246, 1241.629, 1242.030,
1242.396, 1242.780, 1243.164, 1243.547,
1243.931, 1244.314, 1244.698, 1245.081, 1245.465, 1245.848,
1246.232, 1246.615, 1246.999, 1247.382,
1247.766, 1248.149, 1248.533, 1248.916, 1249.299, 1249.683,
1250.066, 1250.450, 1250.833, 1251.216,
1251.600, 1251.983, 1252.366, 1252.749, 1253.133, 1253.516,
1253.899, 1254.283, 1254.666, 1255.049,
1255.432, 1255.815, 1256.199, 1256.582, 1256.965, 1257.348,
1257.731, 1258.114, 1258.497, 1258.881,
1259.264, 1259.647, 1260.030, 1260.413, 1260.796, 1261.179,
1261.562, 1261.945, 1262.328, 1262.711,
1263.094, 1263.477, 1263.860, 1264.243, 1264.626, 1265.009,
1265.392, 1265.775, 1266.157, 1266.540,
1266.923, 1267.306, 1267.689, 1268.072, 1268.455, 1268.837,
1269.220, 1269.603, 1269.986, 1270.368,
1270.751, 1271.134, 1271.517, 1271.899, 1272.282, 1272.665,
1273.048, 1273.430, 1273.813, 1274.195,
1274.578, 1274.691, 1274.803, 1274.916, 1275.029, 1275.141,
1275.254, 1275.366, 1275.479, 1275.591,
1278.404, 1278.786, 1279.169, 1279.551, 1279.934, 1280.316,
1280.699, 1281.081, 1281.464, 1281.846,
1282.228, 1282.611, 1282.993, 1283.376, 1283.758, 1284.140,
1284.523, 1284.905, 1285.287, 1285.670,
1286.052, 1286.434, 1286.816, 1287.199, 1287.581, 1287.963,
1288.345, 1288.728, 1289.110, 1289.492,
1289.874, 1290.256, 1290.638, 1291.021, 1291.403, 1291.785,
1292.167, 1292.549, 1292.931, 1293.313,
1293.695, 1294.077, 1294.459, 1294.841, 1295.223, 1295.605,
1295.987, 1296.369, 1296.751, 1297.133,
1297.515, 1297.897, 1298.279, 1298.661, 1299.043, 1299.425,
1299.807, 1300.188, 1300.570, 1300.952,
1301.334, 1301.716, 1302.098, 1302.479, 1302.861, 1303.243,
1303.625, 1304.006, 1304.388, 1304.770,
1305.152, 1305.533, 1305.915, 1306.297, 1306.678, 1307.060,
1307.442, 1307.823, 1308.205, 1308.586,
1308.968, 1309.350, 1309.731, 1310.113, 1310.494, 1310.876,
1311.270, 1311.639,1312.020, 1312.402,
1312.783, 1313.165, 1313.546, 1313.928, 1314.309, 1314.691,
1315.072, 1315.453, 1315.835, 1316.216,
1316.597, 1316.979, 1317.360, 1317.742, 1318.123, 1318.504,
1318.885, 1319.267, 1319.648, 1320.029,
1320.411,1320.792, 1321.173, 1321.554, 1321.935, 1322.316, 1322.697,
1323.079, 1323.460, 1323.841,
1324.222, 1324.603, 1324.985, 1325.366, 1325.747, 1326.128,

```
        1326.509,       1326.890,       1327.271,       1327.652,
1328.033,       1328.414,       1328.795,       1329.176,       1329.557,       1329.938,
        1330.319,       1330.700,       1331.081,       1331.462,
1331.843,       1332.224,       1332.604,       1332.985,       1333.366,       1333.747,
        1334.128,       1334.509,       1334.889,       1335.270,
1335.651,       1336.032,       1336.413,       1336.793,       1337.174,       1337.555,
        1337.935,       1338.316,       1338.697,       1339.078,
1339.458,       1335.839,       1332.220,       1328.600,       1324.981,       1321.361,
        1317.742,       1314.123,       1310.503,       1306.884,
1343.264,       1343.645,       1344.025,       1344.406,       1344.786,       1345.167,
        1345.570,       1345.928,       1346.308,       1346.689,
1347.069,       1347.450,       1347.830,       1348.211,       1348.591,       1348.971,
        1349.352,       1349.732,       1350.112,       1350.493,
1350.873,       1351.253,       1351.634,       1352.014,       1352.394,       1352.774,
        1353.155,       1353.535,       1353.915,       1354.295,
1354.676,       1355.056,       1355.436,       1355.816,       1356.196,       1356.577,
        1356.957,       1357.337,       1357.717,       1358.097,
1358.477,       1358.857,       1359.237,       1359.617,       1359.997,       1360.377,
        1360.757,       1361.137,       1361.517,       1361.897,
1362.277,       1362.657,       1363.037,       1363.417,       1363.797,       1364.177,
        1364.557,       1364.937,       1365.317,       1365.697,
1366.077,       1366.456,       1366.836,       1367.216,       1367.596,       1367.976,
        1368.355,       1368.735,       1369.115,       1369.495,
1369.875,       1370.254,       1370.634,       1371.014,       1371.393,       1371.773,
        1372.153,       1372.532,       1372.912,       1373.292,
1373.671,       1374.051,       1374.431,       1374.810,       1375.190,       1375.569,
        1375.949,       1376.329,       1376.708,       1377.088,
1377.467,       1377.847,       1378.226,       1378.606,       1378.985,       1379.365,
        1379.744,       1380.123,       1380.503,       1380.882,
        1381.262,       1381.641,       1382.020,       1382.400,       1382.779,       1383.158,
        1383.538,       1383.917,       1384.296,       1384.676, //99
/*
100 度以后分辨率为 1℃
*/
        1385.055,       1388.847,       1392.638,       1396.428,       1400.217,       1404.005,
        1407.791,       1411.576,       1415.360,       1419.143,
        1422.925,       1426.706,               1430.485,       1434.264,       1438.041,       1441.817,
        1445.592,       1449.366,       1453.138,       1456.910,
        1460.680,       1464.449,       1468.217,       1471.984,       1475.750,               1479.514,
        1483.277,       1487.040,       1490.801,       1494.561,
        1498.319,       1502.077,       1505.833,       1509.589,       1513.343,       1517.096,
        1520.847,       1524.598,       1528.381,       1532.139,
        1535.843,       1539.589,       1543.334,       1547.078,       1550.820,       1554.562,
        1558.302,       1562.041,       1565.779,       1569.516,
        1573.251,       1576.986,       1580.719,       1584.451,       1588.182,       1591.912,
        1595.641,       1599.368,       1603.094,       1606.820,
        1610.544,       1614.267,       1617.989,       1621.709,       1625.429,               1629.147,
        1632.864,       1636.580,       1640.295,       1644.009,
        1647.721,       1651.433,       1655.143,       1658.852,       1662.560,               1666.267,
        1669.972,       1673.677,       1677.380,       1681.082,
        1684.783,    1688.483,    1692.181,    1695.879,    1699.575,    1703.271,
        1706.965,    1710.658,    1714.349,    1718.040,
        1721.729,    1725.418,    1729.105,    1732.791,    1736.475,    1740.159,
        1743.842,    1747.523,    1751.203,    1754.882,
        1758.560,    1762.237,    1765.912,    1769.587,    1773.260,    1776.932,
        1780.603,    1784.273,    1787.941,    1791.610,
        1795.275,    1798.940,    1802.604,    1806.267,    1809.929,    1813.590,
        1817.249,    1820.907,    1824.564,    1828.220,
        1831.875,    1835.529,    1839.181,    1842.832,    1846.483,    1850.132,
        1853.779,    1857.426,    1861.072,    1864.716,
        1868.359,    1872.001,    1875.642,    1879.282,    1882.921,    1886.558,
        1890.194,    1893.830,    1897.463,    1901.096,
        1904.728,    1908.359,    1911.988,    1915.616,    1919.243,    1922.869,
        1926.494,    1930.117,    1933.740,    1937.361,
        1940.981,    1944.600,    1948.218,    1951.835,    1955.450,    1959.065,
        1962.678,    1966.290,    1969.901,    1973.510,
        1977.119,    1980.726,        1984.333,    1987.938,    1991.542,    1995.145,
        1998.746,    2002.347,        2005.946,    2009.544,
        2013.141,    2016.737,    2020.332,    2023.925,    2027.518,    2031.109,
        2034.699,    2038.288,    2041.876,    2045.463,
        2049.048,    2052.632,    2056.215,    2059.798,    2063.378,    2066.958,
        2070.537,    2074.114,    2077.690,    2081.265,
        2084.839,    2088.412,    2091.984,    2095.554,    2099.123,    2102.692,
        2106.259,    2109.824,    2113.389,    2116.953,
        2120.515
};

/**********全局变量**********/
#define AD_LEN 2//DMA 获取长度
uint16_t ADC_Val[AD_LEN];//adc 的值  0:台面温度 ad 值。  1：外部探头 ad 值
uint32_t ADC1_Val,ADC2_Val;//adc 的值
#define OP_Value   15.15//放大系数
#define Vref_3V3   3.30//3.3V 电压
#define K1                 0.0834097//电阻基准系数
#define Vref           2.494//参考电压
#define IIR(x,y) (x)=((x) * 9 + (y)) / 10//滤波

/*
*****************************************************************
  *  函数原型：   int Filter_ADC(void)
  *  功      能：   滑动平均值滤波
  *  输      出：   滤波后的值
*****************************************************************
*/
#define N 100//采集 100 次
int ADCvalue_Buf[N];//用于储存采集到的 adc 值
int i = 0;
int Filter_ADC(void)
{
        char count;
```

```
        long sum = 0;

        ADCvalue_Buf[i++] =ADC_Val[0];

        if (i == N)//加入读了 100 组就从新开始
        {
                i = 0;
        }
        for (count = 0; count < N; count++)
        {
                sum += ADCvalue_Buf[count];//100 组相加
        }
        if(ADCvalue_Buf[N-1] == 0)//如果没有读到 100 组就用第一次读到的数
                return ADCvalue_Buf[0];
        else//读到 100 组后
                return (int)(sum / N);//输出平均值
}

/*
******************************************************************
  * 函数原型：  int Filter_ADC1(void)
  * 功      能：  滑动平均值滤波
  * 输      出：  滤波后的值
******************************************************************
*/
int ADCvalue_Buf1[N];//用于储存采集到的 adc 值
int j = 0;
int Filter_ADC1(void)
{
        char count;
        long sum = 0;

        ADCvalue_Buf1[j++] =ADC_Val[1];

        if (j == N)//加入读了 100 组就从新开始
        {
                j = 0;
        }
        for (count = 0; count < N; count++)
        {
                sum += ADCvalue_Buf1[count];//100 组相加
        }
        if(ADCvalue_Buf1[N-1] == 0)//如果没有读到 100 组就用第一次读到的数
                return ADCvalue_Buf1[0];
        else//读到 100 组后
                return (int)(sum / N);//输出平均值
}
/*
******************************************************************
  * 函数原型：void AFE_Sample_Handler(void)
```

```
 * 功      能：计算阻值
*****************************************************************
*/
float temp_correct,temp_correct1;//温度系数
float ADC_Val_Avg[2];//0 为台面温度    1 为探头温度
float AD_T1=0.0;//ADC 计算后的电压值
float AD_T2=0.0;//ADC 计算后的电压值
float PT_VALUE_1_TEMP;//外部探头阻值
float PT_VALUE_2_TEMP;//台面探头阻值
void AFE_Sample_Handler(void)
{
    temp_correct=1.0104f;//外部
//  temp_correct1 = 0.9926f;
    temp_correct1 = 1.0f;

    ADC_Val_Avg[0] = Filter_ADC();
    ADC_Val_Avg[1] = Filter_ADC1();
    AD_T1=((float)ADC_Val_Avg[1]*Vref_3V3/4096)/OP_Value/Vref+K1;//计算电压值
    PT_VALUE_1_TEMP=10000*AD_T1/(1-AD_T1)*temp_correct;//计算电阻值
    AD_T2=((float)ADC_Val_Avg[0]*Vref_3V3/4096)/OP_Value/Vref+K1;//计算电压值
    PT_VALUE_2_TEMP=10000*AD_T2/(1-AD_T2)*temp_correct1;//计算电阻值
}

/*
*****************************************************************
 * 函数原型：int AFE_GetTemperature(float tmp)
 * 功      能：查表
*****************************************************************
*/
int AFE_GetTemperature(float tmp)
{
    int temp;
    if(tmp<1000)//小于 0 摄氏度
    {
        for(int j = 0; j < 220; j++)
        {
            if(tmp < Temp_map[j])
            {
                temp = j-220;
                break;
            }
        }
    }

    else if((tmp<Temp_map[1219])&&(tmp>=1000))//小于 99.9 摄氏度
    {
        for(int j = 220;j < 1220; j++)
        {
            if (tmp < Temp_map[j])
            {
```

```
                        temp = j-220;
                        break;
                }
            }
        }

        else
        {
            for(int j = 1220; j < 1421; j++)
            {
                if(tmp < Temp_map[j])
                {
                        temp = (j-1220)*10+1000;
                        break;
                }
            }
        }
        return temp;
}

/*
*******************************************************************
 * 函数原型：float Get_ADCVal(int16_t temp)
 * 功      能：查表读 ADC 值
*******************************************************************
*/
float Get_ADCVal(int16_t temp)
{
        float adc_Val;
        if(temp < 0)//小于 0 摄氏度
        {
            adc_Val   =   Temp_map[220 + temp];
        }
        else if(temp >= 0 && temp < 1000)
        {
            adc_Val = Temp_map[220 + temp];
        }
        else if(temp >= 1000)
        {
            adc_Val = Temp_map[1220 + (temp-1000)/10];
        }
        return adc_Val;
}

/*
*******************************************************************
 * 函数原型：void ADCDMA_Init(void)
 * 功      能：ADC 和 DMA 的初始化
*******************************************************************
*/
```

```
void ADCDMA_Init(void)
{
	HAL_ADCEx_Calibration_Start(&hadc);
	HAL_TIM_Base_Start_IT(&htim3);//开启 TIM3 的定时，用于刷新
	HAL_ADC_Start_DMA(&hadc, (uint32_t *)ADC_Val, AD_LEN);//用 DMA 获取 adc 值
	for(uint8_t i = 0; i < 200; i++)
	{
		AFE_Sample_Handler();//计算阻值
		Read_Temp(0.6f);
	}
}

/*
*****************************************************************
 *  函数原型：void Read_Temp(float dT)
 *  功	能：读取温度-10ms
*****************************************************************
*/
void Read_Temp(float dT)
{
	static float T;
	Temp.Outside_Temp = AFE_GetTemperature(PT_VALUE_1_TEMP);//外部温度
	Temp.Mesa_Temp = AFE_GetTemperature(PT_VALUE_2_TEMP);//台面温度
	T += dT;
	AFE_Sample_Handler();//计算阻值

	if(T >= 1.0f)
	{
		if(PT_VALUE_1_TEMP < 2000)//假如插入外部探头
			Temp.Rel_Temp = Temp.Outside_Temp;//真实温度显示外部探头测的温度
		else//假如没有插入外部探头
			Temp.Rel_Temp = Temp.Mesa_Temp;//真实温度显示台面温度
		T = 0;
	}
}
#include "Show.h"

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

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

/*
*****************************************************************
 *  函数原型：static void Icn_Twinkle(float dT)
 *  功	能：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)
                {
                        if(Speed.Set_Speed)
                                SpeedIcn_ShowFlag = ~SpeedIcn_ShowFlag;//开盖图标闪烁;
                        if(Time.Ctrl_Time)
                                TimeIcn_ShowFlag = ~TimeIcn_ShowFlag;//开盖图标闪烁;
                        T = 0;
                }
        }
        else
        {
                SpeedIcn_ShowFlag = 0;//显示时间图标
                TimeIcn_ShowFlag = 0;//显示时间图标
        }
}

/*
*****************************************************************
 * 函数原型：static void Check_ShowFlag(float dT)
 * 功      能：闪烁检测
 * 输      入: dT:执行周期
 * 参      数：float dT
 * 调      用：内部调用
*****************************************************************
*/
static void Check_ShowFlag(float dT)
{
        static float 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
                Temp_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)//设置温度
                {
                    Temp_ShowFlag = ~Temp_ShowFlag;//温度闪烁
                    Speed_ShowFlag = 0;//速度常亮
                    Time_ShowFlag = 0;//时间常亮
                }
                else if(sys.SetMode_Option == 2)//设置速度
                {
                    Temp_ShowFlag = 0;//温度常亮
                    Speed_ShowFlag = ~Speed_ShowFlag;//速度闪烁
                    Time_ShowFlag = 0;//时间常亮
                }
                else if(sys.SetMode_Option == 3)//设置时间
                {
                    Temp_ShowFlag = 0;//温度常亮
                    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;
            }
        }
        else
        {
            Speed_ShowFlag = 0;//常亮
            Temp_ShowFlag = 0;//常亮
            Time_ShowFlag = 0;//常亮
            T = 0;
        }
}

/*
*****************************************************************
  * 函数原型：void Twinkle(float dT)
  * 功      能：闪烁函数
*****************************************************************
*/
void Twinkle(float dT)
{
    Check_ShowFlag(dT);//闪烁检测
    Icn_Twinkle(dT);//图标闪烁
}

/*
*****************************************************************
  * 函数原型：void Display_Temp(int16_t dis_set_temp,int16_t dis_rel_temp)
```

```
 * 功      能：显示温度
 * 输      入: dis_set_temp  设定温度    dis_rel_temp  实际温度
 * 参      数：int16_t dis_set_temp,int16_t dis_rel_temp
*****************************************************************
*/
void Display_Temp(int16_t dis_set_temp,int16_t dis_rel_temp)
{
        uint8_t seg1,seg2,seg3,seg4,seg5,seg6,seg7,seg8;
        seg1=0;seg2=0;seg3=0;seg4=0;seg5=0;seg6=0;seg7=0;seg8=0;
        uint8_t Temp_QU,Temp_BU,Temp_SU,Temp_GU;//实际温度的计算位数取值
        uint8_t Temp_QD,Temp_BD,Temp_SD,Temp_GD;//设定温度的计算位数取值
        uint16_t Val;//用于百十个取出来的数字

        /***********设定温度计算**********/
        if(Temp_ShowFlag == 0)
        {
                if(dis_set_temp > 0)
                {
                        if(dis_set_temp > 999)//大于 999 时
                        {
                                Val=dis_set_temp/1000;//取出千位
                                Temp_QD = Tab[Val];
                        }
                        else
                        {
                                Temp_QD = 0x00;//不显示
                        }
                        if(dis_set_temp > 99)//大于 99 时
                        {
                                Val=dis_set_temp/100;//取出百位
                                if(dis_set_temp > 999)//大于 999 时
                                        Val=Val%10;//取出百位
                                Temp_BD = Tab[Val];
                        }
                        else
                        {
                                Temp_BD = 0x00;//不显示
                        }
                        if(dis_set_temp > 9)//大于 9 时
                        {
                                Val=dis_set_temp/10;//取出十位
                                if(dis_set_temp > 99)//大于 99 时
                                        Val=Val%10;//取出十位
                                Temp_SD = Tab[Val];
                        }
                        else
                        {
                                Temp_SD = Tab[0];//不显示 0
                        }
                        Val=dis_set_temp%10;//取出个位
```

```
            Temp_GD = Tab[Val];
            seg6 &= 0x7F;seg6 |= 0x80;//设定温度的小数点
        }
        else
        {
            Temp_QD = 0x04;//显示"-"
            Temp_BD = 0x04;//显示"-"
            Temp_SD = 0x04;//显示"-"
            Temp_GD = 0x04;//显示"-"
            seg6 &= 0x7F;seg6 |= 0x00;//不显示设定温度的小数点
        }
    }
    else
    {
        Temp_QD = 0x00;//不显示设定温度
        Temp_BD = 0x00;//不显示设定温度
        Temp_SD = 0x00;//不显示设定温度
        Temp_GD = 0x00;//不显示设定温度
        seg6 &= 0x7F;seg6 |= 0x00;//不显示设定温度的小数点
    }

    /***********实际温度计算**********/
    if(dis_rel_temp > 999)//大于 999 时
    {
        Val=dis_rel_temp/1000;//取出千位
        Temp_QU = Tab[Val];
    }
    else
    {
        Temp_QU = 0x00;//不显示
    }
    if(dis_rel_temp > 99)//大于 99 时
    {
        Val=dis_rel_temp/100;//取出百位
        if(dis_rel_temp > 999)//大于 999 时
            Val=Val%10;//取出百位
        Temp_BU = Tab[Val];
    }
    else
    {
        Temp_BU = 0x00;//不显示
    }
    if(dis_rel_temp > 9)//大于 9 时
    {
        Val=dis_rel_temp/10;//取出十位
        if(dis_rel_temp > 99)//大于 99 时
            Val=Val%10;//取出十位
        Temp_SU = Tab[Val];
    }
    else
```

```
{
    Temp_SU = Tab[0];//不显示 0
}
Val=dis_rel_temp%10;//取出个位
Temp_GU = Tab[Val];

/*************温度小数点的图标*******/
seg6 &= 0xFE;seg6 |= 0x01;//实际温度的小数点

/****************℃*****************/
seg8 &= 0x7F;seg8 |= 0x80;//℃

/*************外部探头的图标*********/
if(PT_VALUE_1_TEMP < 2000)//假如插入外部探头
{
    seg8 &= 0xFE;seg8 |= 0x01;//外部探头的图标
}
else
{
    seg8 &= 0xFE;seg8 |= 0x00;//不显示外部探头的图标
}

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

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

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

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

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

```
	Write_Addr_Dat_N(12, seg7,1);//SEG21
	Write_Addr_Dat_N(14, seg8,1);//SEG20
}

/*
*****************************************************************
 * 函数原型：void Display_Speed(int16_t dis_set_speed,int16_t dis_rel_speed)
 * 功		能：显示转速
 * 输		入: dis_set_speed  设定转速	dis_rel_speed  实际转速
 * 参		数：int16_t dis_set_speed,int16_t dis_rel_speed
*****************************************************************
*/
void Display_Speed(int16_t dis_set_speed,int16_t dis_rel_speed)
{
	uint8_t seg12,seg13,seg14,seg15,seg16,seg17,seg18,seg19;
	seg12=0;seg13=0;seg14=0;seg15=0;seg16=0;seg17=0;seg18=0;seg19=0;
	uint8_t Speed_QU,Speed_BU,Speed_SU,Speed_GU;//实际速度的计算位数取值
	uint8_t Speed_QD,Speed_BD,Speed_SD,Speed_GD;//设定速度的计算位数取值
	uint16_t Val;//用于百十个取出来的数字
	if(Speed_ShowFlag == 0)
	{
		/***********设定转速计算**********/
		if(dis_set_speed > 999)//大于 999 时
		{
			Val=dis_set_speed/1000;//取出千位
			Speed_QD = Tab[Val];
		}
		else
		{
			Speed_QD = Tab[0];//显示 0
		}
		if(dis_set_speed > 99)//大于 99 时
		{
			Val=dis_set_speed/100;//取出百位
			if(dis_set_speed > 999)//大于 999 时
				Val=Val%10;//取出百位
			Speed_BD = Tab[Val];
		}
		else
		{
			Speed_BD = Tab[0];//显示 0
		}
		if(dis_set_speed > 9)//大于 9 时
		{
			Val=dis_set_speed/10;//取出十位
			if(dis_set_speed > 99)//大于 99 时
				Val=Val%10;//取出十位
			Speed_SD = Tab[Val];
		}
		else
```

```
        {
                Speed_SD = Tab[0];//显示 0
        }
        Val=dis_set_speed%10;//取出个位
        Speed_GD = Tab[Val];
}
else
{
        Speed_QD = 0x00;//不显示设定速度
        Speed_BD = 0x00;//不显示设定速度
        Speed_SD = 0x00;//不显示设定速度
        Speed_GD = 0x00;//不显示设定速度
}

/***********实际转速计算**********/
if(dis_rel_speed > 999)//大于 999 时
{
        Val=dis_rel_speed/1000;//取出千位
        Speed_QU = Tab[Val];
}
else
{
        Speed_QU = Tab[0];//显示 0
}
if(dis_rel_speed > 99)//大于 99 时
{
        Val=dis_rel_speed/100;//取出百位
        if(dis_rel_speed > 999)//大于 999 时
                Val=Val%10;//取出百位
        Speed_BU = Tab[Val];
}
else
{
        Speed_BU = Tab[0];//显示 0
}
if(dis_rel_speed > 9)//大于 9 时
{
        Val=dis_rel_speed/10;//取出十位
        if(dis_rel_speed > 99)//大于 99 时
                Val=Val%10;//取出十位
        Speed_SU = Tab[Val];
}
else
{
        Speed_SU = Tab[0];//显示 0
}
Val=dis_rel_speed%10;//取出个位
Speed_GU = Tab[Val];
```

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

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

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

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

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

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

/*
******************************************************************
 * 函数原型：void Display_Time(int32_t dis_time)
 * 功	能：显示时间
 * 输	入: dis_time  时间
 * 参	数：int32_t dis_time
******************************************************************
*/
void Display_Time(int32_t dis_time)
{
	uint8_t seg9,seg10,seg11,seg20,seg21,seg22,seg23,seg24,seg25,seg26,seg27;
	seg9=0;seg10=0;seg11=0;seg20=0;seg21=0;seg22=0;seg23=0;seg24=0;seg25=0;seg26=0;seg
27=0;
	uint8_t Time_Q,Time_B,Time_S,Time_G;//时间的计算位数取值
	uint8_t SH,H,SM,M;//时间的单位取值
```

```
if(Time.Set_Time || sys.SetMode_Option == 3)//设定时间大于 0，在设定时间和闪烁下
{
    if(!Time_ShowFlag)
    {
        if(Time.Set_Time)//假如设定时间大于 0
        {
            /**************时间计算*************/
            SH=dis_time/3600/10;//计算十位单位的小时数
            H=dis_time/3600%10;//计算个位单位的小时数
            SM=dis_time%3600/60/10;//计算十分位单位的分钟数
            M=dis_time%3600/60%10;//计算个分位单位的分钟数
            Time_Q = Tab[SH];
            Time_B = Tab[H];
            Time_S = Tab[SM];
            Time_G = Tab[M];
        }
        else
        {
            Time_Q = 0x04;//显示"-"
            Time_B = 0x04;//显示"-"
            Time_S = 0x04;//显示"-"
            Time_G = 0x04;//显示"-"
        }
    }
    else//设定时间等于 0
    {
        Time_Q = 0x00;//不显示时间
        Time_B = 0x00;//不显示时间
        Time_S = 0x00;//不显示时间
        Time_G = 0x00;//不显示时间
    }
}
else//设定时间等于 0
{
    Time_Q = 0x00;//不显示时间
    Time_B = 0x00;//不显示时间
    Time_S = 0x00;//不显示时间
    Time_G = 0x00;//不显示时间
}

if(Time.Set_Time > 0 || sys.SetMode_Option == 3)
{
    /************时间冒号图标************/
    seg9 &= 0xFE;seg9 |= 0x01;//时间冒号

    /*************时间单位图标***********/
    seg11 &= 0xF7;seg11 |= 0x08;//min

    /*************时间图标***************/
```

```
        if(TimeIcn_ShowFlag == 0)
        {
            seg20 &= 0x7F;seg20 |= 0x80;//时间图标
        }
        else
        {
            seg20 &= 0x7F;seg20 |= 0x00;//不显示时间图标
        }
    }
    else
    {
        seg9    &= 0xFE;seg9    |= 0x00;//不显示时间冒号
        seg11 &= 0xF7;seg11 |= 0x00;//不显示 min
        seg20 &= 0x7F;seg20 |= 0x00;//不显示显示时间图标
    }

    /*************加热图标***************/
    if(sys.Run_Status && Temp.Ctrl_Temp)
    {
        seg9 &= 0xFB;seg9 |= 0x04;//加热图标
    }
    else
    {
        seg9 &= 0xFB;seg9 |= 0x00;//不显示加热图标
    }

    /*************转速图标***************/
    if(SpeedIcn_ShowFlag == 0)
    {
        seg10 &= 0xFE;seg10 |= 0x01;//逆转指针
        seg11 &= 0xFD;seg11 |= 0x02;//转速图标
    }
    else
    {
        seg10 &= 0xFE;seg10 |= 0x00;//不显示逆转指针
        seg11 &= 0xFD;seg11 |= 0x00;//不显示转速图标
    }

    /*************数据拆分***************/
    seg21 &= 0xF0;seg21 |= Time_G>>4;
    seg20 &= 0xF1;seg20 |= Time_G>>1 & 0x07;

    seg23 &= 0xF0;seg23 |= Time_S>>4;
    seg22 &= 0xF1;seg22 |= Time_S>>1 & 0x07;

    seg25 &= 0xF0;seg25 |= Time_B>>4;
    seg24 &= 0xF1;seg24 |= Time_B>>1 & 0x07;

    seg27 &= 0xF0;seg27 |= Time_Q>>4;
    seg26 &= 0xF1;seg26 |= Time_Q>>1 & 0x07;
    /*************发送数据***************/
    Write_Addr_Dat_N(16, seg9, 1);//SEG19
    Write_Addr_Dat_N(18, seg10,1);//SEG18
    Write_Addr_Dat_N(20, seg11,1);//SEG17
    Write_Addr_Dat_N(38, seg20,1);//SEG8
    Write_Addr_Dat_N(40, seg21,1);//SEG7
    Write_Addr_Dat_N(42, seg22,1);//SEG6
    Write_Addr_Dat_N(44, seg23,1);//SEG5
    Write_Addr_Dat_N(46, seg24,1);//SEG4
    Write_Addr_Dat_N(48, seg25,1);//SEG3
    Write_Addr_Dat_N(50, seg26,1);//SEG2
    Write_Addr_Dat_N(52, seg27,1);//SEG1
}

/*
*****************************************************************
 * 函数原型：void Deal_Speed(void)
 * 功    能：速度显示处理
*****************************************************************
*/
void Deal_Speed(void)
{
    if(sys.Run_Status)
    {
        if(Speed.Speed_ADDMode==0)//在进入加速模式下
        {
            if(Speed.Display_RelSpeed >= Speed.Ctrl_Speed)//当前的速度大于等于控制速
度
            {
                Speed.Speed_ADDMode = 2;//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_RelSpeed = Speed.Speed_New;//显示当前速度
            else//当前速度小于上一次速度
            {
                Speed.Display_RelSpeed = 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==1)//在进入减速模式下
        {
            if(Speed.Display_RelSpeed <= Speed.Ctrl_Speed)//当前的速度大于等于控制速
度
            {
                sys.Run_Status = 0;//关闭系统
                SetOK_Flag = 1;//设置标志置一
                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_RelSpeed = Speed.Speed_New;//显示当前速度
            else//当前速度大于上一次速度
            {
                Speed.Display_RelSpeed = 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 == 2)//速度稳定模式下
        {
            Speed.Display_RelSpeed = Speed.Ctrl_Speed;//显示控制速度
        }
    }
    else
    {
        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 = 0;//清除显示处理
    }
}

/*
*****************************************************************
 * 函数原型：void Deal_Temp(void)
 * 功      能：温度显示处理
*****************************************************************
*/
void Deal_Temp(void)
{
    if(sys.Run_Status && Temp.Ctrl_Temp)
    {
        if(ABS(Temp.Ctrl_Temp - Temp.Rel_Temp) < 40)
        {
            if(Temp.Temp_ADDMode==0)//判断模式
            {
                if(Temp.Ctrl_Temp > Temp.Rel_Temp)
                {
                    Temp.Temp_ADDMode = 1;//进入升温模式
                    Temp.Temp_Last = Temp.Rel_Temp;
                }
                else if(Temp.Ctrl_Temp < Temp.Rel_Temp)
                {
                    Temp.Temp_ADDMode = 2;//进入降温模式
                    Temp.Temp_Last = Temp.Rel_Temp;//记录当前温度
                }
                else
                {
                    Temp.Temp_ADDMode = 3;//进入稳定模式
                }
            }
            else if(Temp.Temp_ADDMode==1)//在进入升温模式下
            {
                Temp.Temp_New = Temp.Rel_Temp;//记录当前温度
                if(Temp.Temp_New > Temp.Temp_Last)//当前温度大于上一次温度
                    Temp.Display_RelTemp = Temp.Temp_New;//显示当前温度
                else//当前温度小于上一次温度
                {
                    Temp.Display_RelTemp = Temp.Temp_Last;//显示上一次温度，不让温度小于当前温度。呈现攀升速度的现象
                    Temp.Temp_New = Temp.Temp_Last;//将上一次温度赋值给当前温度
                }
                Temp.Temp_Last = Temp.Temp_New;//将当前温度保存
                if(Temp.Display_RelTemp  >=  Temp.Ctrl_Temp)//当前的温度大于等于控制温度
                {
                    Temp.Temp_ADDMode = 3;//进入稳定模式
                }
            }
            else if(Temp.Temp_ADDMode==2)//在进入降温模式下
            {
                Temp.Temp_New = Temp.Rel_Temp;//记录当前温度
                if(Temp.Temp_New < Temp.Temp_Last)//当前温度小于上一次温度
                    Temp.Display_RelTemp = Temp.Temp_New;//显示当前温度
                else//当前温度大于上一次温度
                {
                    Temp.Display_RelTemp = Temp.Temp_Last;//显示上一次温度，不让温度小于当前温度。呈现攀升速度的现象
                    Temp.Temp_New = Temp.Temp_Last;//将上一次温度赋值给当前温度
                }
                Temp.Temp_Last = Temp.Temp_New;//将当前温度保存
                if(Temp.Display_RelTemp  <=  Temp.Ctrl_Temp)//当前的温度小于等于控制温度
                {
                    Temp.Temp_ADDMode = 3;//进入稳定模式
                }
            }
            else if(Temp.Temp_ADDMode == 3)//温度稳定模式下
            {
                Temp.Display_RelTemp = Temp.Ctrl_Temp;//显示控制温度
            }
        }
        else
        {
            Temp.Temp_ADDMode = 0;//进入稳定模式
```

```
            Temp.Display_RelTemp = Temp.Rel_Temp;//显示实际温度
        }
    }
    else
    {
        Temp.Display_RelTemp = Temp.Rel_Temp;//显示实际温度
        Temp.Temp_New =0;//现在的速度清零
        Temp.Temp_Last = 0;//之前的速度清零
        Temp.Temp_ADDMode = 0;//清除显示处理
    }
}

/*
*****************************************************************
 * 函数原型： void Show_Display(void)
 * 功    能： 显示屏幕内容
*****************************************************************
*/
void Show_Display(void)
{
    Temp.Display_SetTemp = Temp.Set_Temp;//显示设定温度
    Deal_Temp();//温度显示处理

    Speed.Display_SetSpeed = Speed.Set_Speed;//显示设定转速
    Deal_Speed();//速度显示处理

    if(sys.Run_Status)
        Time.Display_Time = Time.Ctrl_Time+59;//显示控制时间
    else
        Time.Display_Time = Time.Set_Time;//显示设定时间

    Display_Temp(Temp.Display_SetTemp,Temp.Display_RelTemp);//显示温度
    Display_Speed(Speed.Display_SetSpeed,Speed.Display_RelSpeed);//显示转速
    Display_Time(Time.Display_Time);//显示时间
}
#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.P_Param[0] = 370;//温度
    Param.P_Param[1] = 100;//转速
    Param.P_Param[2] = 0;//时间

    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();
        Temp.Set_Temp = Param.P_Param[0];//温度
        Speed.Set_Speed = Param.P_Param[1];//转速
        Time.Set_Time = Param.P_Param[2];//时间
        SetOK_Flag = 1;
        Save_Param_En = 1;
    }
    else
    {
        Temp.Set_Temp = Param.P_Param[0];//温度
        Speed.Set_Speed = Param.P_Param[1];//转速
        Time.Set_Time = Param.P_Param[2];//时间
        SetOK_Flag = 1;
    }
	//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();//保存硬件中的参数
			Save_Param_En = 0;//关闭保存标志位
		}
	}
	else
		time = 0;
}
#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(float dT)
  * 功　　能：检测设置
******************************************************************
*/
void Check_Set(float dT)
{
	if(Key_Status != 0)
	{
		SetOK_Flag = 1;//检测到设置标志后
	}
	if(SetOK_Flag == 1)
	{
```

```
            if(sys.SetMode_Option == 0)//在设定好后
            {
                  if(Temp.Ctrl_Temp != Temp.Set_Temp)//控制温度不等于设定温度的话
                  {
                        Temp.Ctrl_Temp = Temp.Set_Temp;//控制温度等于设定温度
                        Param.P_Param[0] = Temp.Ctrl_Temp;//将控制温度保存到 Flash
                  }

                  if(Speed.Ctrl_Speed != Speed.Set_Speed)//控制速度不等于设定速度的话
                  {
                        Speed.Ctrl_Speed = Speed.Set_Speed;//控制速度等于设定速度
                        Param.P_Param[1] = Speed.Ctrl_Speed;//将控制速度保存到 Flash
                  }

                  if(Time.Ctrl_Time != Time.Set_Time)//控制时间不等于设定时间的话
                  {
                        Time.Ctrl_Time = Time.Set_Time;//控制时间等于设定时间
                        Param.P_Param[2] = Time.Ctrl_Time;//将控制时间保存到 Flash
                  }

                  SetOK_Flag = 0;//设定好后清零标志位
                  Save_Param_En = 1;//保存 Flash 标志位置一
            }
      }
}
#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
_PID_Arg_ Temp_Arg;
_PID_Val_ Temp_Val;



/*
*****************************************************************
 *  函数原型：  void PID_Init(void)
 *  功      能：  pid 系数初始化
*****************************************************************
*/
void PID_Init(void)
{
      Speed_Arg.Kp=0.08;
      Speed_Arg.Ki=0.000646;
      Speed_Arg.Kd=0.00043;

      Temp_Arg.Kp = 1000*0.001f;
      Temp_Arg.Ki = 20*0.001f;
      Temp_Arg.Kd = 8000*0.001f;//控台面
}
```

```
/*
*****************************************************************
 * 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
                    PID_val_t *pid_val)//PID 数据结构体

{
       pid_val->Error = Expect - Feedback;//当前误差
       if(pid_val->Error > 100)
              pid_val->Error = 100;
       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<0)
              pid_val->Out=0;
       if(pid_val->Out>0&&pid_val->Out<2500)
              pid_val->Out=pid_val->Out;
}


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

```
PID_Val, float Error_Lim, float Integral_Lim
*****************************************************************
*/
void AltPID_Calculation(float dT, float Expect, float Freedback, _PID_Arg_ * PID_Arg, _PID_Val_ * PID_Val, float Error_Lim, float Integral_Lim)
{
	PID_Val->Error = Expect - Freedback;//误差 = 期望值-反馈值

	PID_Val->Proportion	= PID_Arg->Kp * PID_Val->Error;//比例 = 比例系数*误差
	PID_Val->Fb_Differential = -PID_Arg->Kd * ((Freedback - PID_Val->Freedback_Old) * safe_div(1.0f, dT, 0));//微分 = -（微分系数） * （当前反馈值-上一次反馈值） *频率
	PID_Val->Integral += PID_Arg->Ki * LIMIT(PID_Val->Error, -Error_Lim, Error_Lim) * dT;//积分 = 积分系数*误差*周期
	PID_Val->Integral = LIMIT(PID_Val->Integral, -Integral_Lim, Integral_Lim);//积分限幅

	PID_Val->Out = PID_Val->Proportion + PID_Val->Integral + PID_Val->Fb_Differential;//PID 输出

	PID_Val->Freedback_Old = Freedback;//将当前反馈值赋值给上一次反馈值
}
#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 TIM1CH3_CAPTURE_STA=0;//捕获溢出的周期数
uint32_t TIM1CH3_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(float dT)
 * 功      能：检测速度是否停止
*****************************************************************
*/
void Check_Speed(float dT)
{
	if(Speed_Flag)
		Speed_Flag -= dT;
	if(Speed_Flag==0)
		Speed.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
		TIM1CH3_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
		{
			TIM1CH3_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

      TIM1CH3_CAPTURE_VAL=HAL_TIM_ReadCapturedValue(&htim1,TIM_CHANNEL_3);
//获取当前捕获计数值
                  long long temp=0;
                  temp=TIM1CH3_CAPTURE_STA;//溢出的次数（用于计算进入多少个周期）
                  temp*=50;//一个周期溢出的时间（us）计算=定时器周期（1/（48000000/6）*400）
=0.00005S=50us
                  temp+=TIM1CH3_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/2;//一分钟有 60000000us，一分钟内有多少个脉冲(周期)。
2 是一圈有几个脉冲得出一分钟转了多少圈
                  Speed.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
            {
                  TIM1CH3_CAPTURE_STA=0;//清除周期计数
                  TIM1CH3_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_500ms=0,
              T_cnt_1S=0;

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

      Sys_Loop();
}
```

```
static void Loop_2ms(float dT)//2ms 执行一次
{

}

static void Loop_10ms(float dT)//10ms 执行一次
{
        Key_Scan(dT);//按键扫描
        Check_Set(dT);//检测设置
        Motor_Ctrl(dT);//电机控制
        Read_Temp(dT);//读取温度-10ms
}

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

}

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

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

}

static void Loop_1S(float dT)//1S 执行一次
{
        Temp_Control(dT);//温度加热控制
        Check_Speed(dT);//检测速度是否停止
}

void Sys_Loop(void)
{
        if(T_cnt_2ms >= 2) {
                Loop_2ms(0.002f);
                T_cnt_2ms = 0;
        }
        if(T_cnt_10ms >= 10) {
                Loop_10ms(0.01f);
                T_cnt_10ms = 0;
        }
        if(T_cnt_50ms >= 50) {
```

```
		Loop_50ms(0.05f);
		T_cnt_50ms = 0;
	}
	if(T_cnt_100ms >= 100) {
		Loop_100ms(0.1f);
		T_cnt_100ms = 0;
	}
	if(T_cnt_500ms >= 500) {
		Loop_500ms(0.5f);
		T_cnt_500ms = 0;
	}
	if(T_cnt_1S >= 1000) {
		Loop_1S(1.0f);
		T_cnt_1S = 0;
	}
}
#include "Ctrl_ControlTemp.h"

/**********局部变量声明******/
uint8_t Out_Enable;//积分运算的开关
float adc_val,ctrl_val;
uint8_t water_type,step;//判断是油还是水 0：水，1 油
uint8_t Temp_type;//温度类型

/*
*****************************************************************
 *  函数原型：static void Mesa_Ctrl(float dT, int32_t Ctrl_temp)
 *  功      能：台面温度控制
*****************************************************************
*/
static void Mesa_Ctrl(float dT, int32_t Ctrl_temp)
{
	if(Ctrl_temp > 2800) Ctrl_temp = 2800;//最高加热 280 度
	adc_val = Get_ADCVal(Ctrl_temp)*10;//查表读 ADC 值
	ctrl_val = PT_VALUE_2_TEMP*10;//实际的 adc 值
	if(Ctrl_temp > 2300 && Temp.Mesa_Temp > 2000)//如果控制温度大于 230 度，台面温度
超过 200 度
		Out_Enable = 1;//打开积分计算
	else if((adc_val - ctrl_val > -200 && adc_val - ctrl_val < 200))
		Out_Enable = 1;//打开积分计算
	else
		Out_Enable = 0;//关闭积分计算
	AltPID_Calculation(dT, adc_val, ctrl_val, &Temp_Arg, &Temp_Val, 100, Out_Enable *
1000);
	HEAT_ON();//加热模块开启
	if(Temp_Val.Out < 0)
		Temp_Val.Out = 0;
	HEAT =(int)Temp_Val.Out;
}
```

```
/*
*****************************************************************
  * 函数原型：void Temp_Control(float dT)
  * 功      能：温度加热控制
*****************************************************************
*/
void Temp_Control(float dT)
{
      if(sys.Run_Status && PT_VALUE_1_TEMP >= 2000)//启动系统控制台面
      {
            if(Temp_type == 1)//如果是外部探头切换过来
            {
                  Temp.Temp_ADDMode = 0;
                  Temp_type = 0;
            }
            if(Temp.Ctrl_Temp)
            {
                  Mesa_Ctrl(dT, Temp.Ctrl_Temp);
            }
            else
            {
                  water_type = 0;
                  HEAT_OFF();//加热模块关闭
                  HEAT = 0;//pwm 不输出
                  step = 0;
            }
      }
      else if(sys.Run_Status && PT_VALUE_1_TEMP < 2000)//启动系统控制水温
      {
            if(Temp.Ctrl_Temp)
            {
                  if(Temp_type == 0)//如果是台面探头切换过来
                  {
                        Temp.Temp_ADDMode = 0;
                        Temp_type = 1;
                  }
                  if(Temp.Ctrl_Temp < 1000)//控制温度小于 100℃
                  {
                        if(Temp.Ctrl_Temp - Temp.Rel_Temp > 100)
                        {
                              Mesa_Ctrl(dT, Temp.Ctrl_Temp*2.8f);
                        }
                        else  if(Temp.Ctrl_Temp  -  Temp.Rel_Temp  <  100  &&  Temp.Ctrl_Temp  -
Temp.Rel_Temp >= 50)
                        {
                              Mesa_Ctrl(dT, Temp.Ctrl_Temp*2.5f);
                        }
                        else  if(Temp.Ctrl_Temp  -  Temp.Rel_Temp  <  50  &&  Temp.Ctrl_Temp  -
Temp.Rel_Temp >= 10)
                        {
```

```
                    Mesa_Ctrl(dT, Temp.Ctrl_Temp*2.2f);
                }
                else  if(Temp.Ctrl_Temp  -  Temp.Rel_Temp  <  10  &&  Temp.Ctrl_Temp  -
Temp.Rel_Temp > 0)
                {
                    Mesa_Ctrl(dT, Temp.Ctrl_Temp*2.1f);
                }
                else
                {
                    Temp_Val.Out = 0;
                    HEAT =(int)Temp_Val.Out;
                }
            }
            else if(Temp.Ctrl_Temp == 1000)//控制温度等于 100℃
            {
                if(!water_type)
                {
                    if(step == 0)
                    {
                        Mesa_Ctrl(dT, 2800);
                        if(Temp.Rel_Temp >= 1000)
                        {
                            step = 1;
                        }
                    }
                    else if(step == 1)
                    {
                        Mesa_Ctrl(dT, 2200);
                    }
                    if(Temp.Rel_Temp >= 1040)
                    {
                        water_type = 1;//判断为油
                    }
                }
                else
                {
                    if(Temp.Ctrl_Temp - Temp.Rel_Temp > 100)
                    {
                        Mesa_Ctrl(dT, Temp.Ctrl_Temp*2.8f);
                    }
                    else if(Temp.Ctrl_Temp - Temp.Rel_Temp < 100 && Temp.Ctrl_Temp -
Temp.Rel_Temp >= 50)
                    {
                        Mesa_Ctrl(dT, Temp.Ctrl_Temp*2.5f);
                    }
                    else  if(Temp.Ctrl_Temp  -  Temp.Rel_Temp  <  50  &&  Temp.Ctrl_Temp  -
Temp.Rel_Temp >= 10)
                    {
                        Mesa_Ctrl(dT, Temp.Ctrl_Temp*2.2f);
                    }
                    else if(Temp.Ctrl_Temp - Temp.Rel_Temp < 10 && Temp.Ctrl_Temp -
Temp.Rel_Temp > 0)
                    {
                        Mesa_Ctrl(dT, Temp.Ctrl_Temp*2.1f);
                    }
                    else
                    {
                        Temp_Val.Out = 0;
                        HEAT =(int)Temp_Val.Out;
                    }
                }
            }
            else if(Temp.Ctrl_Temp > 1000)//控制温度大于 100℃
            {
                if(Temp.Ctrl_Temp - Temp.Rel_Temp > 100)
                {
                    Mesa_Ctrl(dT, Temp.Ctrl_Temp * 2.2f);
                }
                else if(Temp.Ctrl_Temp - Temp.Rel_Temp < 100 && Temp.Ctrl_Temp -
Temp.Rel_Temp >= 50)
                {
                    Mesa_Ctrl(dT, Temp.Ctrl_Temp * 2.0f);
                }
                else if(Temp.Ctrl_Temp - Temp.Rel_Temp < 50 && Temp.Ctrl_Temp -
Temp.Rel_Temp > 0)
                {
                    Mesa_Ctrl(dT, Temp.Ctrl_Temp * 1.8f);
                }
                else
                {
                    Temp_Val.Out = 0;
                    HEAT =(int)Temp_Val.Out;
                }
            }
        }
        else
        {
            water_type = 0;
            HEAT_OFF();//加热模块关闭
            HEAT = 0;//pwm 不输出
            step = 0;
        }
    }
    else
    {
        water_type = 0;
        HEAT_OFF();//加热模块关闭
        HEAT = 0;//pwm 不输出
        step = 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)//启动
	{
		if(Speed.Set_Speed > 0)
		{
			MO_ON();//电机启动
			PID_Speed(Speed.Ctrl_Speed,Speed.Rel_Speed,&Speed_Arg,&Speed_Val);// 电机
PID 控制
			PWM = Speed_Val.Out;//pid 输出
		}
		else
		{
			MO_OFF();//电机关闭
			Speed_Val.SumError = 0;//清除积分合
			PWM = 0;//pwm 不输出
		}
	}
	else
	{
		MO_OFF();//电机关闭
		Speed_Val.SumError = 0;//清除积分合
		PWM = 0;//pwm 不输出
	}
}
#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)//启动系统
	{
		if(Time.Ctrl_Time > 0 && Speed.Speed_ADDMode != 1)
```

```
            {
                T += dT;
                if(T >= 1.0f)//1S
                {
                    if(Time.Ctrl_Time)
                        Time.Ctrl_Time--;//控制时间--
                    if(Time.Ctrl_Time == 0)
                    {
                        Speed.Speed_ADDMode = 1;//进入减速模式
                        Speed.Ctrl_Speed = 0;//将控制速度设置为 0
                        Beep_Flash = 5;//响 5 下
                    }
                    T = 0;//周期清零
                }
            }
        }
}
#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)
{
        /**********系统初始化成功**********/
        sys.Init_ok = 0;

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

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

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

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

        /*********ADC&DMA 初始化********/
        ADCDMA_Init();

        /**********加热初始化**********/
        HEAT_Init();

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

```
    /**********蜂鸣器开机响********/
    Beep_Time = 0.1;

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