WB3100 软件源程序

```
#include "Flash.h"

/*
*****************************************************************
 * 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 "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;
}

/*
*********************************************************************
 * 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 "KEY.h"

/**********全局变量**********/
uint16_t run_mode = 1;//运行模式
uint8_t WaterErr;//防干烧屏幕闪烁

/**********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
uint8_t Key_Status;//在操作按键时

/*
*********************************************************************
 * 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) == 1 )//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) == 1)
        {
            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) == 0) && (HAL_GPIO_ReadPin
(GPIOB, KEY3_Pin) == 0 ) && (HAL_GPIO_ReadPin (GPIOB, KEY1_Pin) == 0 ) )
        {
            if(HAL_GPIO_ReadPin (GPIOB, KEY5_Pin) == 0)
            {
                KEY1_Pin_ON = 0;//长按计数
            }
            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,KEY2_Pin) == KEY_ON))//减
    {
        if(Run_Status > 0) //运行中不能设置
            return;
        if(Select_Option == 1)//在设置温度选项
```

```
        {
            set_temp-=10;//温度--;
            if(set_temp < 0)//如果设定温度小于 0 时（单加热只能自动降温）
            {
                set_temp = 0;//将设定温度保持在 0
            }
        }
        if(Select_Option == 2)//在设置时间选项
        {
            if(time_status == 0)//在秒单位模式下
            {
                if(set_time)
                    set_time -= 5;//时间减 5s
                if(set_time < 5)//小于 5s 的设定值时
                {
                    time_Last = 1;//跳出倒计时
                    SetTime_State = 1;//设定时间显示“----”
                }
            }
            else//在分为单位的模式下
            {
                if(set_time == 86399)
                    set_time -= 119;//时间减 1 分钟
                else if(set_time == 3600)
                    set_time -= 5;//时间减 1 分钟
                else
                    set_time -= 60;//时间减 1 分钟
            }
//              rel_time = set_time;//调时间时，将设定时间赋值给实际倒计时时间
        }
        Twinkle_On = 6000;//闪烁倒计时，如果停止按键设置，6S 后停止闪烁
        Key_Status = 1;//按键操作时不闪烁，2s 后闪烁
        Beep_Time = 0.1;//蜂鸣器响 0.1S
    }

/*******************************加*****************************************/
    if((Key_Scan(GPIOB,KEY3_Pin) == KEY_ON))//加
    {
        if(Run_Status > 0) //运行中不能设置
            return;
        if(Select_Option == 1)//在设置温度选项
        {
            set_temp+=10;//温度++;
            if(set_temp > 1000)//最高设定温度在 100℃
                set_temp = 1000;
        }
        if(Select_Option == 2)//在设置时间选项
        {
            if(time_status == 0)//在秒单位模式下
            {
```

```
                    set_time += 5;//时间加 5s
                    time_Last = 0;//加入倒计时
                    SetTime_State = 0;//设定时间退出显示“----”
                }
                else//在分单位模式下
                    set_time += 60;//时间加 60s
                if(set_time > 86399)//最高可定时 23.99 小时
                    set_time = 86399;
//                  rel_time = set_time;//调时间时，将设定时间赋值给实际倒计时时间
            }
            Twinkle_On = 6000;//闪烁倒计时，如果停止按键设置，6S 后停止闪烁
            Key_Status = 1;//按键操作时不闪烁，2s 后闪烁
            Beep_Time = 0.1;//蜂鸣器响 0.1S
        }

/*****************************菜单键*************************************/
        if((Key_Scan(GPIOB,KEY1_Pin) == KEY_ON))//菜单键
        {
            if(Run_Status > 0) //运行中不能设置
                return;

                Select_Option++;//设置选项切换
                if(Select_Option > 2 && Select_Option <=3)//在温度和时间来换选择，后面的等
于 3 为了防止从 p 模式当出来，会没有马上选中温度
                {
                    Select_Option = 0;//不进行设置
                }
                Twinkle_On = 6000;//闪烁倒计时，如果停止按键设置，6S 后停止闪烁
                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(Run_Status == 0)//系统没启动
            {
                if(Protect == 1)//防干烧
                {
                    Beep_Flash = 30;
                    Run_Status = 0;
                    WaterErr = 1;
                    return;
                }
                Select_Option = 0;//设定选项清零
                Run_Status = 1; //系统启动
                Temp_Control = 1;//温度控制开始
                time_disable = 0;//关闭倒计时
                ADD_Mode = 0;//加热状态清零
            }
            else
```

```
            {
                Run_Status = 0;//关闭系统
                time_disable = 0;//关闭倒计时
                Temp_Control = 0;//关闭温度控制
                ADD_Mode = 0;//加热状态清零
                rel_time = set_time;//将设定时间赋值给实际倒计时时间
            }
            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--;
}
#include "Ntc.h"

/**********局部变量**********/
const uint16_t R10K_TAB[] = //R25=10K?3% B25/50=4100K?3% 10K??//-20-105
{
      342,//-20
      361,//-19
      381,//-18
      401,//-17
      423,//-16
      445,//-15
      468,//-14
      492,//-13
      517,//-12
      543,//-11
      570,//-10
      597,//-9
      626,//-8
      655,//-7
      685,//-6
      717,//-5
      749,//-4
      782,//-3
      815,//-2
      850,//-1
      886,//0
      922,//1
      959,//2
```

997,//3
1036,//4
1075,//5
1115,//6
1156,//7
1198,//8
1240,//9
1282,//10
1325,//11
1369,//12
1413,//13
1457,//14
1502,//15
1547,//16
1592,//17
1638,//18
1683,//19
1729,//20
1775,//21
1821,//22
1866,//23
1912,//24
1958,//25
1986,//26
2032,//27
2093,//28
2130,//29
2170,//30
2215,//31
2265,//32
2297,//33
2335,//34
2380,//35
2420,//36
2445,//37
2484,//38
2513,//39
2533,//40
2596,//41
2630,//42
2670,//43
2695,//44
2719,//45
2733,//46
2750,//47
2844,//48
2876,//49
2906,//50
2939,//51
2967,//52

```
2996,//53
3027,//54
3050,//55
3081,//56
3106,//57
3130,//58
3161,//59
3182,//60
3209,//61
3227,//62
3248,//63
3270,//64
3290,//65
3311,//66
3333,//67
3351,//68
3370,//69
3390,//70
3409,//71
3424,//72
3443,//73
3457,//74
3476,//75
3489,//76
3504,//77
3519,//78
3535,//79
3549,//80
3564,//81
3574,//82
3592,//83
3600,//84
3612,//85
3624,//86
3634,//87
3648,//88
3658,//89
3667,//90
3678,//91
3687,//92
3697,//93
3705,//94
3715,//95
3722,//96
3732,//97
3739,//98
3747,//99
3757,//100
3762,//101
3777,//102
```

```
    3786,//103
    3795,//104
    3803,//105
    3812,//106
    3820,//107
    3827,//108
    3835,//109
    3842,//110
    3850,//111
    3856,//112
    3862,//113
    3869,//114
    3875,//115
    3881,//116
    3887,//117
    3893,//118
    3898,//119
    3903,//120
};

/**********全局变量**********/
int rel_temp;//实际温度
int set_temp;//设定温度
int ctrl_temp;//设定温度
uint16_t val;//adc 的值
uint8_t res;//温度采样返回的状态

/*
*********************************************************************
 * 函数原型：  int filter(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 value_buf[N];//用于储存采集到的 adc 值
int i = 0;
int filter(void)
{
    char count;
    long sum = 0;

    HAL_ADC_Start(&hadc);//开始读取 adc 的值
    value_buf[i++] = HAL_ADC_GetValue(&hadc);//将 adc 的值储存
    if (i == N)//加入读了 100 组就从新开始
    {
        i = 0;
    }
    for (count = 0; count < N; count++)
    {
```

```
            sum += value_buf[count];//100 组相加
        }
        if(value_buf[99] == 0)//如果没有读到 100 组就用第一次读到的数
            return value_buf[0];
        else//读到 100 组后
            return (int)(sum / N);//输出平均值
}

/*
*****************************************************************
 * 函数原型：  uint16_t func_get_ntc_temp(uint16_t value_adc)
 * 功      能：  计算出 Ntc 的温度
 * 输      入:  value_adc:adc 读到的值
 * 参      数：  uint16_t value_adc
*****************************************************************
*/
#define SHORT_CIRCUIT_THRESHOLD 15
#define OPEN_CIRCUIT_THRESHOLD 4096
uint8_t index_l, index_r ;
uint16_t func_get_ntc_temp(uint16_t value_adc)
{
    uint8_t r10k_tab_size = 141;
    int temp = 0;
    if(value_adc <= SHORT_CIRCUIT_THRESHOLD)
    {
        return 1;
    }
    else if(value_adc >= OPEN_CIRCUIT_THRESHOLD)
    {
        return 2;
    }
    else if(value_adc < R10K_TAB[0])
    {
        return 3;
    }

    else if(value_adc > R10K_TAB[r10k_tab_size - 1])
    {
        return 4;
    }

    index_l = 0;
    index_r = r10k_tab_size - 1;
    for(; index_r - index_l > 1;)
    {
        if((value_adc <= R10K_TAB[index_r]) && (value_adc > R10K_TAB[(index_l +
index_r) % 2 == 0 ? (index_l + index_r) / 2 : (index_l + index_r) / 2 ]))
        {
            index_l = (index_l + index_r) % 2 == 0 ? (index_l + index_r) / 2 : (index_l +
index_r) / 2 ;
```

```
            }
            else
            {
                index_r = (index_l + index_r) / 2;
            }
        }
        if(R10K_TAB[index_l] == value_adc)
        {
            temp = (((int)index_l) - 21) * 10; //rate *10
        }
        else if(R10K_TAB[index_r] == value_adc)
        {
            temp = (((int)index_r) - 21) * 10; //rate *10
        }
        else
        {
            if(R10K_TAB[index_r] - R10K_TAB[index_l] == 0)
            {
                temp = (((int)index_l) - 21) * 10; //rate *10
            }
            else
            {
                temp = (((int)index_l) - 21
                ) * 10 + ((value_adc - R10K_TAB[index_l] ) * 100 + 5) / 10 / (R10K_TAB[index_r]
- R10K_TAB[index_l]);
            }
        }

/******************温度补偿**********************/
//      if(set_temp <= 750)//设定温度小于 75℃
            rel_temp = temp;//实际温度等于测得温度
//      else if((set_temp > 750)&&(set_temp <= 900))//设定温度在 75 到 100℃之间
//              rel_temp = temp + 50;//问题补偿 1℃
//       else if((set_temp > 900)&&(set_temp <= 1000))//设定温度在 75 到 100℃之间
//              rel_temp = temp + 100;//问题补偿 1℃
        return 0;
}

/*
*********************************************************************
 * 函数原型：  void Read_Temp(void)
 * 功      能：  读取温度-10ms
*********************************************************************
*/
void Read_Temp(void)
{
        static uint8_t Num;
        Num++;
        val = filter();//滤波获取 adc 的滑动平均值
        if(Num == 100)//1S
```

```
    {
        res = func_get_ntc_temp(val);//计算温度
        Num = 0;
    }
}
#include "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_FGS();
    FGS_Stop();
}

static void Loop_10ms(void)//10ms 执行一次
{
    Read_Temp();//读取温度
    Key_Handle();//按键检测
    Check_Set();//检测设置
}

static void Loop_50ms(void)//50ms 执行一次
{
    Buzzer_Status(0.05);
    Buzzer(0.05);
}

static void Loop_100ms(void)//100ms 执行一次
{
    Cheak_ShowFlag(100);//闪烁检测
```

```
    Cheak_TimeDown(100);//倒计时
    Param_Save_Overtime(0.1f);//保存标志位置
}

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

}

static void Loop_500ms(void)//500ms 执行一次
{
    temp_control();//温度控制
    Cheak_Protect();
}

static void Loop_1S(void)//1S 执行一次
{
    Check_Key();
}

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 "Show.h"

/**********全局变量**********/
uint32_t rel_time;//实时时间
uint32_t set_time;//设定时间
uint8_t time_status;//时间显示模式
uint8_t Set_Mode_Enable;//P 键进入设置模式 0：模式设置不予显示 1：模式设置显示
uint8_t run_mode_flag;//进入 P 时显示
uint8_t Select_Option;//设置时当前设置的选项
uint16_t Twinkle_Time;//闪烁的时间
uint16_t Twinkle_On;//闪烁倒计时
uint8_t circle_dis;//梯度模式下外圈转动显示（本代码不操作此寄存器，单加热）
uint8_t circle_dis_flag;//外圈开始转动（本代码不操作此寄存器，单加热）
uint8_t mode_flag_p1;//梯度模式下 P1 的闪烁（本代码不操作此寄存器，单加热）
uint8_t mode_flag_p2;//梯度模式下 P1 的闪烁（本代码不操作此寄存器，单加热）
uint8_t mode_run_p1;//梯度模式下 P1 的值（本代码不操作此寄存器，单加热）
uint8_t mode_run_p2;//梯度模式下 P1 的值（本代码不操作此寄存器，单加热）
uint8_t  set_mode_p;//P 模式下切换梯度模式还是就记忆模式 1：梯度模式 0：p 模式（本代码不操作此寄存器，单加热）
uint8_t SetTime_State;//未设定时间显示“----”
uint8_t Twinkle_Protect;//防干烧闪烁标志

/**********局部变量**********/
uint8_t FIRST_Tab[] = {0xee, 0x24, 0xba, 0xb6, 0x74, 0xd6, 0xde, 0xa4, 0xfe, 0xf6};
uint8_t LAST_Tab[] = {0x77, 0X24, 0x5d, 0x6d, 0x2e, 0x6b, 0x7b, 0x25, 0x7f, 0x6f};
uint8_t MID_Tab[] =    {0x77, 0x12, 0x5d, 0x5b, 0x3a, 0x6b, 0x6f, 0x52, 0x7f, 0x7b};
uint8_t Tab[4] = {0, 0, 0, 0};
int Dis_Rel_Temp;//显示实际温度
int Dis_Set_Temp;//显示温度
int Dis_Rel_Time;//实时时间
int Dis_Set_Time;//设定时间
uint8_t temp_flag;//选中设置温度时闪烁
uint8_t time_flag;//选中设置时间时闪烁
uint8_t mode_flag;//选中设置模式时闪烁

/*
*********************************************************************
 * 函数原型：  void Circle_Go(void)
 * 功      能：  跑梯度模式-单加热没有用到
*********************************************************************
*/
void Circle_Go(void)
{
//      run_mode_flag = 1;//不固定显示外框
//      circle_dis_flag = 1;//外框开始跑圈（本代码不操作此寄存器，单加热）
      if((circle_dis_flag ) && (Run_Status > 0))//跑梯度标志位置一，系统启动
      {
            circle_dis -= 1;//显示--
            if(circle_dis < 1)//小于 1 表示一圈跑完
```

```
            circle_dis = 12;//从头跑
        }
}
void Cheak_Protect(void)
{
        if(Protect == 0)
        {
            return;
        }
        if(WaterErr == 1)
        {
            Twinkle_Protect =    ~Twinkle_Protect;
        }
//       else
//       {
//            Twinkle_Protect = 0;
//       }
}
/*
**********************************************************************
 * 函数原型：   void Cheak_ShowFlag(uint16_t dT)
 * 功       能：  闪烁检测
 * 输       入:  dT:执行周期
 * 参       数：  uint16_t dT
**********************************************************************
*/
void Cheak_ShowFlag(uint16_t dT)
{
        if(Select_Option == 0 || Key_Status)//如果没在设置选项中，则都点亮，不闪烁
        {
            temp_flag = 0;//点亮
            time_flag = 0;//点亮
            mode_flag = 0;//点亮
            return;
        }
        Twinkle_Time += dT;//闪烁计时
        Twinkle_On -= dT;//无操作闪烁倒计时
        if(Select_Option == 1)//设置温度
        {
            if(Twinkle_Time % 500 == 0)//每 0.5S 转换状态
            {
                temp_flag = ~ temp_flag;
                time_flag = 0;
                mode_flag = 0;
            }
        }
        else if(Select_Option == 2)//设置时间
        {
            if(Twinkle_Time % 500 == 0)//每 0.5S 转换状态
            {
```

```
                time_flag = ~ time_flag;
                temp_flag = 0;
                mode_flag = 0;
            }
        }
        else if(Select_Option == 3)//设置模式
        {
            if(Twinkle_Time % 500 == 0)//每 0.5S 转换状态
            {
                temp_flag = 0;
                time_flag = 0;
                mode_flag = ~ mode_flag;
            }
        }
        else if(Select_Option == 4)//在梯度模式下闪烁 P1（本代码不操作此寄存器，单加热）
        {
            if(Twinkle_Time % 500 == 0)//每 0.5S 转换状态
            {
                mode_flag_p1 = ~mode_flag_p1;
                mode_flag_p2 = ~mode_flag_p2;
            }
        }
        else if(Select_Option == 5)//在梯度模式下闪烁 P2（本代码不操作此寄存器，单加热）
        {
            if(Twinkle_Time % 500 == 0)//每 0.5S 转换状态
            {
                mode_flag_p1 = ~mode_flag_p1;
                mode_flag_p2 = ~mode_flag_p2;
            }
        }
        if(Twinkle_Time == 10000)//闪烁 6 次，约 10S
        {
            Twinkle_Time = 0;
        }
        if(Twinkle_On == 0)//如果闪烁倒计时完了
        {
            Twinkle_Time = 0;//把闪烁计时清零
            Select_Option = 0;//选项闪烁结束
        }
}

/*
**********************************************************************
 * 函数原型：  void Dis_RelTemp(int dis_rel_temp)
 * 功      能：  显示实际温度
 * 输      入:  dis_rel_temp：实际温度
 * 参      数：  int dis_rel_temp
**********************************************************************
*/
void Dis_RelTemp(int dis_rel_temp)
```

```
{
    if(dis_rel_temp > 999)//千位
    {
        Tab[0] = LAST_Tab[dis_rel_temp / 1000];
        Tab[1] = LAST_Tab[dis_rel_temp / 100 % 10];
        Tab[2] = LAST_Tab[dis_rel_temp / 10 % 10];
        Tab[3] = LAST_Tab[dis_rel_temp % 10];
    }
    else if(dis_rel_temp > 99)//百位
    {
        Tab[0] = 0;
        Tab[1] = LAST_Tab[dis_rel_temp / 100];
        Tab[2] = LAST_Tab[dis_rel_temp / 10 % 10];
        Tab[3] = LAST_Tab[dis_rel_temp % 10];
    }
    else if(dis_rel_temp > 9)//十位
    {
        Tab[0] = 0;
        Tab[1] = 0;
        Tab[2] = LAST_Tab[dis_rel_temp / 10];
        Tab[3] = LAST_Tab[dis_rel_temp % 10];
    }
    else if(dis_rel_temp > -1)//个位
    {
        Tab[0] = 0;
        Tab[1] = 0;
        Tab[2] = LAST_Tab[dis_rel_temp / 10];
        Tab[3] = LAST_Tab[dis_rel_temp % 10];
    }
    else if(dis_rel_temp > -10)//负数
    {
        Tab[0] = 0;
        Tab[1] = 0x08;
        Tab[2] = LAST_Tab[0];
        Tab[3] = LAST_Tab[(-dis_rel_temp)];
    }
    else if(dis_rel_temp > -100)//负十位
    {
        Tab[0] = 0;
        Tab[1] = 0x08;
        Tab[2] = LAST_Tab[(-dis_rel_temp) / 10];
        Tab[3] = LAST_Tab[(-dis_rel_temp) % 10];
    }
    else//负百位
    {
        Tab[0] = 0x08;
        Tab[1] = LAST_Tab[1];
        Tab[2] = LAST_Tab[0];
        Tab[3] = LAST_Tab[0];
    }
```

```
    if(Run_Status == 1)//开始控制温度时
        Tab[3]=Tab[3]|0x80;//加热图标


    Tab[2] = Tab[2] | 0x80;//实际温度的小数点
    Tab[0] = Tab[0] | 0x80;//设置温度的℃符号

    if(WaterErr == 1)
    {
        if(Twinkle_Protect >= 1)
        {
            Tab[0] = 0xFF;
            Tab[1] = 0x7F;
            Tab[2] = 0xFF;
            Tab[3] = 0xFF;
        }
        else
        {
            Tab[0] = 0x00;
            Tab[1] = 0x00;
            Tab[2] = 0x00;
            Tab[3] = 0x00;
        }
    }
    write_addr_dat_n(0, Tab[0], 1);
    write_addr_dat_n(2, Tab[1], 1);
    write_addr_dat_n(4, Tab[2], 1);
    write_addr_dat_n(6, Tab[3], 1);
}

/*
*********************************************************************
 * 函数原型：  void Dis_SetTemp(int dis_set_temp)
 * 功      能：  显示设定温度
 * 输      入:  dis_set_temp：设定温度
 * 参      数：  int dis_set_temp
*********************************************************************
*/
void Dis_SetTemp(int dis_set_temp)
{
    if(dis_set_temp > 999)//千位
    {

        Tab[0] = FIRST_Tab[dis_set_temp / 1000];
        Tab[1] = FIRST_Tab[dis_set_temp / 100 % 10];
        Tab[2] = FIRST_Tab[dis_set_temp / 10 % 10];
        Tab[3] = FIRST_Tab[dis_set_temp % 10];
    }
    else if(dis_set_temp > 99)//百位
```

```
{

    Tab[0] = 0;
    Tab[1] = FIRST_Tab[dis_set_temp / 100];
    Tab[2] = FIRST_Tab[dis_set_temp / 10 % 10];
    Tab[3] = FIRST_Tab[dis_set_temp % 10];
}
else if(dis_set_temp > -1)//十位
{

    Tab[0] = 0;
    Tab[1] = 0;
    Tab[2] = FIRST_Tab[dis_set_temp / 10];
    Tab[3] = FIRST_Tab[dis_set_temp % 10];
}
else if(dis_set_temp > -10)//个位
{
    Tab[0] = 0;
    Tab[1] = 0x10;
    Tab[2] = FIRST_Tab[0];
    Tab[3] = FIRST_Tab[(-dis_set_temp)];
}
else if(dis_set_temp > -100)//负数
{
    Tab[0] = 0;
    Tab[1] = 0x10;
    Tab[2] = FIRST_Tab[(-dis_set_temp) / 10];
    Tab[3] = FIRST_Tab[(-dis_set_temp) % 10];
}
else//负百位
{
    Tab[0] = 0x10;
    Tab[1] = FIRST_Tab[1];
    Tab[2] = FIRST_Tab[0];
    Tab[3] = FIRST_Tab[0];
}

Tab[2] = Tab[2] | 0x01;//设置温度的小数点

if(temp_flag)//闪烁
{
    Tab[0] = 0;
    Tab[1] = 0;
    Tab[2] = 0;
    Tab[3] = 0;
}
if(WaterErr == 1)
{
    if(Twinkle_Protect >= 1)
    {
```

```
                Tab[0] = 0xFE;
                Tab[1] = 0xFE;
                Tab[2] = 0xFF;
                Tab[3] = 0xFF;
            }
            else
            {
                Tab[0] = 0x00;
                Tab[1] = 0x00;
                Tab[2] = 0x00;
                Tab[3] = 0x00;
            }
        }
        write_addr_dat_n(32, Tab[3], 1);
        write_addr_dat_n(34, Tab[2], 1);
        write_addr_dat_n(36, Tab[1], 1);
        write_addr_dat_n(38, Tab[0], 1);
}

/*
**********************************************************************
 * 函数原型：  void Dis_RelTime(int dis_rel_time)
 * 功      能：  显示实际时间
 * 输      入:  dis_rel_time：实际时间
 * 参      数：  int dis_rel_time
**********************************************************************
*/
void Dis_RelTime(int dis_rel_time)
{
    if(time_status ==0)//在秒显示状态下
    {
        if(dis_rel_time>59)//分钟以 60 进一单位
        {
            Tab[0]=LAST_Tab[dis_rel_time/60/10];
            Tab[1]=LAST_Tab[dis_rel_time/60%10];
            Tab[2]=LAST_Tab[dis_rel_time%60/10];
            Tab[3]=LAST_Tab[dis_rel_time%60%10];
        }
        else
        {
            Tab[0]=LAST_Tab[0];
            Tab[1]=LAST_Tab[0];
            Tab[2]=LAST_Tab[dis_rel_time%60/10];
            Tab[3]=LAST_Tab[dis_rel_time%60%10];
        }
    }
    else//在分显示状态下
    {
        Tab[0]=LAST_Tab[dis_rel_time/3600/10];
        Tab[1]=LAST_Tab[dis_rel_time/3600%10];
```

```
            Tab[2]=LAST_Tab[dis_rel_time%3600/60/10];
            Tab[3]=LAST_Tab[dis_rel_time%3600/60%10];
        }

        Tab[2]=Tab[2]|0x80;//实时时间冒号
//      Tab[0]=Tab[0]|0x80;//制冷图标
        if(WaterErr == 1)
        {
            if(Twinkle_Protect >= 1)
            {
                Tab[0] = 0x7F;
                Tab[1] = 0x7F;
                Tab[2] = 0xFF;
                Tab[3] = 0xFF;
            }
            else
            {
                Tab[0] = 0x00;
                Tab[1] = 0x00;
                Tab[2] = 0x00;
                Tab[3] = 0x00;
            }
        }
        write_addr_dat_n(8,Tab[0], 1);
        write_addr_dat_n(10,Tab[1], 1);
        write_addr_dat_n(12,Tab[2], 1);
        write_addr_dat_n(14,Tab[3], 1);
}

/*
*********************************************************************
 * 函数原型：  void Dis_SetTime(int dis_set_time)
 * 功      能：  显示设定时间
 * 输      入:  dis_set_time：设定时间
 * 参      数：  int dis_set_time
*********************************************************************
*/
void Dis_SetTime(int dis_set_time)
{
    if(dis_set_time > 3599)//如果设定时间大于 59.59 分钟时
    {
        time_status=1;//单位变成分
    }
    else
        time_status=0;//不然就是秒

    if(time_status ==0)//在秒显示状态下
    {
        if(dis_set_time>59)//分钟以 60 进一单位
        {
```

```
            Tab[0]=MID_Tab[dis_set_time/60/10];
            Tab[1]=MID_Tab[dis_set_time/60%10];
            Tab[2]=MID_Tab[dis_set_time%60/10];
            Tab[3]=MID_Tab[dis_set_time%60%10];
        }
        else
        {
            Tab[0]=MID_Tab[0];
            Tab[1]=MID_Tab[0];
            Tab[2]=MID_Tab[dis_set_time%60/10];
            Tab[3]=MID_Tab[dis_set_time%60%10];
        }
    }
    else
    {
        Tab[0]=MID_Tab[dis_set_time/3600/10];
        Tab[1]=MID_Tab[dis_set_time/3600%10];
        Tab[2]=MID_Tab[dis_set_time%3600/60/10];
        Tab[3]=MID_Tab[dis_set_time%3600/60%10];
    }
    Tab[1]=Tab[1]|0x80;//设定时间冒号
    if(SetTime_State)
    {
        Tab[3]= 0x08;
        Tab[2]= 0x08;
        Tab[1]= 0x08;
        Tab[0]= 0x08;
    }

    if(time_flag)//闪烁
    {
        Tab[0]=0;
        Tab[1]=0;
        Tab[2]=0;
        Tab[3]=0;
    }


    if(time_status)
        Tab[3]=Tab[3]|0x80;//分钟单位显示
    else
        Tab[2]=Tab[2]|0x80;//秒单位显示

    if(Set_Mode_Enable)//（本代码不操作此寄存器，单加热）
    {
        if((((circle_dis<10)&&(circle_dis>3))&&(circle_dis!=0))||(circle_dis==13))
        {
            Tab[0]=Tab[0]|0x80;//模式外圈显示
        }
        else if((circle_dis==3)||(circle_dis==0))
```

```
            Tab[0]=Tab[0]&0x7f;//模式外圈不显示
        if(run_mode_flag==0)
            Tab[0]=Tab[0]|0x80;//模式外圈显示
    }
    else
    {
        Tab[0]=Tab[0]&0x7f;//模式外圈不显示
    }
    if(WaterErr == 1)
    {
        if(Twinkle_Protect >= 1)
        {
            Tab[0] = 0x7F;
            Tab[1] = 0xFF;
            Tab[2] = 0xFF;
            Tab[3] = 0xFF;
        }
        else
        {
            Tab[0] = 0x00;
            Tab[1] = 0x00;
            Tab[2] = 0x00;
            Tab[3] = 0x00;
        }
    }
    write_addr_dat_n(16,Tab[3], 1);
    write_addr_dat_n(18,Tab[2], 1);
    write_addr_dat_n(20,Tab[1], 1);
    write_addr_dat_n(22,Tab[0], 1);
}

/*
*********************************************************************
 * 函数原型：  void Dis_RunMode(uint8_t E,uint8_t P,uint8_t P1,uint8_t P2)
 * 功    能：  显示运行模式
 * 输    入:  E：P 模式框显示   P：记忆和梯度选择 P1：梯度模式下 P1 值 P2：梯度模
式下 P2 值
 * 参    数：  uint8_t E,uint8_t P,uint8_t P1,uint8_t P2
*********************************************************************
*/
void Dis_RunMode(uint8_t E,uint8_t P,uint8_t P1,uint8_t P2)
{
    static uint8_t tab1=0;
    if(E)//进入 P 模式显示
    {
        if(circle_dis)//如果标准位大于一
        {
            switch(circle_dis)//用 switch 语句实现动画转圈
            {
                case 0:write_addr_dat_n(24,0x00, 1);
```

```
                break;
            case 1: tab1|=0X01;tab1&=0Xbf; write_addr_dat_n(24,tab1, 1);
                break;
            case 2: tab1|=0X02;tab1&=0X7f; write_addr_dat_n(24,tab1, 1);
                break;
            case 3: tab1|=0X04; write_addr_dat_n(24,tab1, 1);
                break;
            case 4: tab1|=0X08; write_addr_dat_n(24,tab1, 1);
                break;
            case 5: tab1|=0X10; write_addr_dat_n(24,tab1, 1);
                break;
            case 6: tab1|=0X20; write_addr_dat_n(24,tab1, 1);
                break;
            case 7: tab1|=0X40;tab1&=0XFE; write_addr_dat_n(24,tab1, 1);
                break;
            case 8: tab1|=0X80;tab1&=0XFC; write_addr_dat_n(24,tab1, 1);
                break;
            case 9: tab1&=0XFB; write_addr_dat_n(24,tab1, 1);
                break;
            case 10: tab1&=0XF7; write_addr_dat_n(24,tab1, 1);
                break;
            case 11: tab1&=0XEF; write_addr_dat_n(24,tab1, 1);
                break;
            case 12: tab1&=0XCF; write_addr_dat_n(24,tab1, 1);
                break;
            case 13: tab1&=0XCF; write_addr_dat_n(24,0xff, 1);
                break;
        }
    }

    if(run_mode_flag==0)//加入不在 p 模式下
        write_addr_dat_n(24,0xff, 1);//外框消失，方便转圈


    //模式选择
    if(P)//梯度模式下
    {
        Tab[2]=FIRST_Tab[P1];//模式一
        Tab[1]=0X10;//-
        Tab[0]=FIRST_Tab[P2];//模式二
    }
    else//记忆模式下
    {
        Tab[0]=FIRST_Tab[run_mode];//显示模式数
        Tab[1]=0X10;//-
        Tab[2]=0xf8;//显示字母 P
    }

    if(mode_flag)//闪烁显示
    {
```

```
            Tab[0]=0;
            Tab[1]=0;
            Tab[2]=0;
        }

        if(mode_flag_p1)//梯度模式下 P1 闪烁（本代码不操作此寄存器，单加热）
            Tab[2]=0;
        if(mode_flag_p2)//梯度模式下 P2 闪烁（本代码不操作此寄存器，单加热）
            Tab[0]=0;

        //模式外圈显示
        if((((circle_dis<11)&&(circle_dis>4))&&(circle_dis!=0))||(circle_dis==13))
            Tab[2]=Tab[2]|0x01;//模式外圈显示
        else if((circle_dis==4)||(circle_dis==0))
            Tab[2]=Tab[2]&0xfe;//模式外圈不显示
        if(run_mode_flag==0)
            Tab[2]=Tab[2]|0x01;//模式外圈显示
        if((((circle_dis<13)&&(circle_dis>6))&&(circle_dis!=0))||(circle_dis==13))
            Tab[0]=Tab[0]|0x01;//模式外圈显示
        else if((circle_dis==6)||(circle_dis==0))
            Tab[0]=Tab[0]&0xfe;//模式外圈不显示
        if(run_mode_flag==0)
            Tab[0]=Tab[0]|0x01;//模式外圈显示
        if((((circle_dis<12)&&(circle_dis>5))&&(circle_dis!=0))||(circle_dis==13))
            Tab[1]=Tab[1]|0x01;//模式外圈显示
        else if((circle_dis==5)||(circle_dis==0))
            Tab[1]=Tab[1]&0xfe;//模式外圈不显示
        if(run_mode_flag==0)
            Tab[1]=Tab[1]|0x01;//模式外圈显示
    }
    else//不显示
    {
        Tab[0]=0;
        Tab[1]=0;
        Tab[2]=0;
        write_addr_dat_n(24,0x00, 1);
    }

    write_addr_dat_n(26,Tab[0], 1);
    write_addr_dat_n(28,Tab[1], 1);
    write_addr_dat_n(30,Tab[2], 1);
}

int temp_c;
/*
*********************************************************************
 * 函数原型：  void Deal_Temp(void)
 * 功    能：  温度显示处理
*********************************************************************
*/
```

```
void Deal_Temp(void)
{
    static int Temp_New,Temp_Last;//现在温度、之前温度
    if(Run_Status == 0)//没启动的情况下
    {
        Dis_Rel_Temp = rel_temp;
    }
    else//启动的情况下
    {
        if(ADD_Mode == 0)//判断数据处理显示
        {
            if(set_temp > rel_temp)//设定温度大于显示温度
            {
                Temp_Control = 1;//开启加热
                ADD_Mode = 1;//进入加热模式下
            }
             Temp_New = 0;//将之前的记入值清零
             Temp_Last = 0;//将之前的记入值清零
        }
        if(ADD_Mode==1)//在加热模式下
        {
             Temp_New = rel_temp;//记录当前温度
             if(Temp_New > Temp_Last)//当前温度大于上一次温度
                 Dis_Rel_Temp = Temp_New;//显示当前温度
             else//当前温度小于上一次温度
             {
                 Dis_Rel_Temp = Temp_Last;//显示上一次温度，不让温度小于当前温度。
呈现攀升温度的现象
                 Temp_New = Temp_Last;//将上一次温度赋值给当前温度
             }
             Temp_Last = Temp_New;//将当前温度保存
             if(rel_temp >= set_temp)
                 ADD_Mode = 2;
        }
        else if(ADD_Mode == 2)//温度稳定模式下
        {
             Dis_Rel_Temp = set_temp;//显示当前显示温度
        }
    }
}

/*
*********************************************************************
 * 函数原型：void LCD_Display(void)
 * 功      能：屏幕显示
*********************************************************************
*/
void LCD_Display(void)
{
    Deal_Temp();
```

```
    /**********************显示实际温度**********************/
//      Dis_Rel_Temp = rel_temp;
    Dis_RelTemp(Dis_Rel_Temp);

    /**********************显示设定温度**********************/
    Dis_Set_Temp = set_temp;
    Dis_SetTemp(Dis_Set_Temp);

    /**********************显示实际时间**********************/
    Dis_Rel_Time = rel_time;
    Dis_RelTime(Dis_Rel_Time);

    /**********************显示设定时间**********************/
    Dis_Set_Time = set_time;
    Dis_SetTime(Dis_Set_Time);

    Dis_RunMode(Set_Mode_Enable,set_mode_p,mode_run_p1,mode_run_p2);//P 模式

}
#include "TempControl.h"

/**********全局变量**********/
uint8_t Temp_Control = 0;//温度控制标志位
uint8_t ADD_Mode;//温度加热状态
uint8_t Protect;//防干烧保护功能

/**********结构体**********/
PID_val_t Temp_Val;//pid 数据结构
PID_arg_t Temp_Arg;//pid 数据系数

void Check_FGS(void)
{
    if(HAL_GPIO_ReadPin(FGS_GPIO_Port,FGS_Pin) == 1)
    {
        Protect = 1;
    }
    else
    {
        Protect =0;
        WaterErr = 0;
    }
}

void FGS_Stop(void)
{
    if(Run_Status == 0)
        return;
    else
    {
```

```
            if(Protect == 1)
            {
                Run_Status = 0;
                Beep_Flash = 30;
                WaterErr = 1;
            }
        }
}


/*
*********************************************************************
 * 函数原型：  void PID_Init(void)
 * 功      能：  PID 系数初始化
*********************************************************************
*/
void PID_Init(void)
{
    Temp_Arg.Kp = 12;//比例系数
    Temp_Arg.Ki = 0;//积分系数
    Temp_Arg.Kd = 10; //微分系数
}

/*
*********************************************************************
 * 函数原型：  void PID_Calc(float NextPoint ,float SetPoint)
 * 功      能：  PID 算法
 * 输      入:  NextPoint；实际温度 , SetPoint：设定温度
 * 参      数：  float NextPoint ,float SetPoint
*********************************************************************
*/
void PID_Calc(float NextPoint ,float SetPoint)
{
    Temp_Val.Error = SetPoint-NextPoint;//计算误差
    //Temp_Val.SumError+=Temp_Val.Error;//计算误差和
    Temp_Val.D_Error =Temp_Val.LastError-Temp_Val.PrevError;//计算微分误差
    Temp_Val.PrevError =   Temp_Val.LastError;//保存这次误差
    Temp_Val.LastError =   Temp_Val.Error;//保存上一次误差
    Temp_Val.Integral = Temp_Arg.Ki*Temp_Val.SumError/10000.0;//计算积分值
    if(Temp_Val.Integral>1000)//积分限幅
    {
        Temp_Val.Integral=1000;
    }
    Temp_Val.Out                                                    =
Temp_Arg.Kp*Temp_Val.Error+Temp_Val.Integral+Temp_Arg.Kd*Temp_Val.D_Error;// 输 出
PID 的值
}
int Temp_Out;
/*
*********************************************************************
```

```
 * 函数原型：  void temp_control(void)
 * 功      能：  温度加热控制
*********************************************************************
*/
void temp_control(void)
{
    if(Run_Status < 1 && Temp_Control == 0 )//如果温度没有启动控制
    {
        HEAT = 0;//加热不工作
        return;
    }
    if(Protect)
    {
        Run_Status = 0;//系统关闭
        Beep_Flash = 30;
        WaterErr = 1;
        HEAT = 0;
        time_disable = 1;//关闭倒计时标志位
        ADD_Mode = 0;//加热模式置 0
        Temp_Control = 0;//温度不控制
        Beep_Flash = 5;
        rel_time = set_time;//将设定时间赋值给实际倒计时时间
    }
    PID_Calc(rel_temp,ctrl_temp);//pid 计算
    Temp_Out = (int)Temp_Val.Out;//pid 值赋值
    if(Temp_Out > 90 && Temp_Out<1000)     Temp_Out = 1000;//pid 输出限幅
    if(Temp_Out > 25 && Temp_Out < 90)      Temp_Out =380;//pid 输出限幅
    if(Temp_Out <=25)Temp_Out = 0;

    HEAT = Temp_Out;//控制加热模块
}
#include "TimeDown.h"

/**********全局变量**********/
uint8_t time_disable;//关闭计时
uint8_t time_Last;//一直计时

/*
*********************************************************************
 * 函数原型：  void Sys_Init(void)
 * 功      能：  时间倒计时检测
 * 输      入:  dT:执行周期
 * 参      数：  uint16_t dT
*********************************************************************
*/
void Cheak_TimeDown(uint16_t dT)
{
    static uint16_t T;
    if(ADD_Mode != 2)//系统启动和开始倒计时
        return;
```

```
    if(T == 1000)//1S
    {
        if(time_Last == 1)
            return;
        if(rel_time)//实际时间大于 0
        {
            rel_time--;//每一秒减一
        }
        else//倒计时结束
        {
            Run_Status = 0;//系统关闭
            time_disable = 1;//关闭倒计时标志位
            ADD_Mode = 0;//加热模式置 0
            Temp_Control = 0;//温度不控制
            Beep_Flash = 5;
            rel_time = set_time;//将设定时间赋值给实际倒计时时间
        }
    }
    else if(T > 1000)//大于 1S 后
    {
        T = dT;//因为已经对比++;
    }
    T += dT;//周期是 100ms
}
/**********全局变量**********/
uint8_t Run_Status;//系统状态

/*
*********************************************************************
 * 函数原型：  void Sys_Init(void)
 * 功    能：  系统参数初始化
*********************************************************************
*/
void Sys_Init(void)
{
    Run_Status = 0;//未启动
    lcd_init();//lcd 初始话
    HAL_Delay(5);
    for(uint8_t i =0;i<=100;i++)
    {
        val = filter();//滤波获取 adc 的滑动平均值
        if(i == 100)//1S
        {
            res = func_get_ntc_temp(val);//计算温度
        }
        HAL_Delay(10);//没有延时开机读不出温度
    }
    lcd_clr();//屏幕熄灭
    Dis_Rel_Temp = rel_temp;
    Dis_RelTemp(rel_temp);//显示温度
```

```
    Beep_OFF;//关闭蜂鸣器
    Beep_Time = 0.1;//蜂鸣器响 0.1S
    Protect = 1;
    Param_Read();
    if(set_time == 0)
    {
        time_Last = 1;//跳出倒计时
        SetTime_State = 1;//设定时间显示“----”
    }
    PID_Init();//PID 系数初始化
}
#include "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)
{
    if(Beep_Time <= 0 && Beep_Flash <= 0)
    {
        Beep_OFF;
        return;
    }
    Beep_ON;
    Beep_Time -= dT;
}

void Buzzer(float dT)
{
    static float BT;
    if(Beep_Flash <= 0 && Beep_Time <= 0)//蜂鸣器的时间小于等于 0 时
    {
        Beep_OFF;//关闭蜂鸣器
        return;
    }
    if(Beep_Flash)
    {
        BT =   BT + dT;//周期++
        if(BT < 0.1)//如果小于 0.1s 时
        {
            Beep_ON;//蜂鸣器响
```

```
            }
            else if(BT >= 0.2 && BT < 0.3)//在 0.1 和 0.2s 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 "SetVal.h"

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

/*
*********************************************************************
 * 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(Select_Option == 0)//在设定好后
        {
            if(Set_Mode_Enable == 1)
            {
                Param.P_Param[1] = set_time;
                Param.P_Param[0] = set_temp;
                Save_Param_En = 1;
            }
            if(ctrl_temp != set_temp)//判断控制速度和设定速度是不是不一样
            {
                ctrl_temp = set_temp;//把设定速度赋值给控制速度
                if(ADD_Mode  !=  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
                ADD_Mode = 0;//进入未处理，判断加速还是减速
                Param.P_Param[0] = set_temp;
                Save_Param_En = 1;
            }
            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
                    Param.P_Param[1] = set_time;
                    Save_Param_En = 1;
                }
                if(set_time > 0 )
                {
                    SetTime_State = 0;//设定时间退出显示“----”
                    time_Last = 0;
                }
                else
                {
                    SetTime_State = 1;//设定时间显示“----”
                    time_Last = 1;
                }
                SetOK_Flag = 0;
            }
        }
}
```