DB4100 软件源程序

```
#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_HT162x.h"

/*
*********************************************************************
```

```
 * 函数原型：static void LCD_Delay(void)
 * 功      能：LCD_us 延时
 * 调      用：内部调用
**********************************************************************
*/
static void LCD_Delay(void)
{
    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)
{
    LCD_Delay();
    Clr_162x_Wr;//RW = 0;
    LCD_Delay();
    Set_162x_Dat;//DA = 1;
    Set_162x_Wr;//RW = 1;
    LCD_Delay();
    Clr_162x_Wr;//RW = 0;
    LCD_Delay();
    Clr_162x_Dat;//DA = 0;
    LCD_Delay();
    Set_162x_Wr;//RW = 1;
    LCD_Delay();
    Clr_162x_Wr;//RW = 0;
    LCD_Delay();
    if (0 == MODE)
    {
        Clr_162x_Dat;//DA = 0;
    }
    else
    {
        Set_162x_Dat;//DA = 1;
    }
    LCD_Delay();
    Set_162x_Wr;//RW = 1;
    LCD_Delay();
}

/*
```

```
*****************************************************************
 * 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_162x_Wr;
        if ((Cbyte >> (7 - i)) & 0x01)
        {
            Set_162x_Dat;
        }
        else
        {
            Clr_162x_Dat;
        }
        LCD_Delay();
        Set_162x_Wr;
        LCD_Delay();
    }
    Clr_162x_Wr;
    LCD_Delay();
    Clr_162x_Dat;
    Set_162x_Wr;
    LCD_Delay();
}

/*
*****************************************************************
 * 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_162x_Wr;
        if ((Abyte >> (6 - i)) & 0x01)
```

```
        {
            Set_162x_Dat;
        }
        else
        {
            Clr_162x_Dat;
        }
        LCD_Delay();
        Set_162x_Wr;
        LCD_Delay();
    }
}

/*
*********************************************************************
 * 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_162x_Wr;
        if ((Dbyte >> (7 - i)) & 0x01)
        {
            Set_162x_Dat;
        }
        else
        {
            Clr_162x_Dat;
        }
        LCD_Delay();
        Set_162x_Wr;
        LCD_Delay();
    }
}

/*
*********************************************************************
 * 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_162x_Wr;
		if ((Dbyte >> (3 - i)) & 0x01)
		{
			Set_162x_Dat;
		}
		else
		{
			Clr_162x_Dat;
		}
		LCD_Delay();
		Set_162x_Wr;
		LCD_Delay();
	}
}

/*
**************************************************************************
 * 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_162x_Cs;
	Set_162x_Wr;
	Set_162x_Dat;
	LCD_Delay();
	Clr_162x_Cs;//CS = 0;
	LCD_Delay();
	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_162x_Cs;//CS = 1;

	HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
	__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 50);//背光 pwm
```

```
    Lcd_All();
    Temp_Init();
    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_162x_Cs;//CS = 0;
    LCD_Delay();
    Write_Mode(1);
    Write_Address(_addr);
    for (i = 0; i < n; i++)
    {
        Write_Data_8bit(_dat);
    }
    Set_162x_Cs;//CS = 1;
}
```

```
#include "Drv_NTC.h"

/**********局部变量**********/
const uint16_t R100K_TAB[] = //R25=100K B25=3950K -25-150
{
//      160,//-10
//    170,//-12
      180,//-10
      189,//-9
      200,//-8
      210,//-7
      221,//-6
      233,//-5
      245,//-4
      258,//-3
      271,//-2
      284,//-1
      298,//0
      313,//1
      328,//2
      344,//3
      361,//4
      372,//5
      396,//6
      416,//7
      440,//8
      462,//9
      482,//10
      508,//11
      528,//12
      558,//13
      576,//14
      600,//15
      632,//16
      654,//17
      682,//18
      710,//19
      734,//20
      758,//21
      794,//22
      826,//23
      860,//24
      892,//25
      917,//26
      950,//27
      982,//28
      1014,//29
      1044,//30
      1075,//31
      1107,//32
```

```
1146,//33
1182,//34
1260,//35
1294,//36
1324,//37
1372,//38
1393,//39
1415,//40
1465,//41
1497,//42
1556,//43
1598,//44
1621,//45
1652,//46
1720,//47
1740,//48
1760,//49
1804,//50
1842,//51
1882,//52
1920,//53
1958,//54
1994,//55
2045,//56
2083,//57
2122,//58
2142,//59
2193,//60
2245,//61
2290,//62
2330,//63
2359,//64
2400,//65
2418,//66
2460,//67
2502,//68
2539,//69
2553,//70
2580,//71
2614,//72
2648,//73
2681,//74
2710,//75
2736,//76
2765,//77
2788,//78
2812,//79
2845,//80
2870,//81
2894,//82
```

```
    2934,//83
    2963,//84
    2989,//85
    3022,//86
    3042,//87
    3064,//88
    3092,//89
    3112,//90
    3132,//91
    3154,//92
    3176,//93
    3192,//94
    3202,//95
    3222,//96
    3240,//97
    3269,//97
    3292,//98
    3315,//99
    3338,//100
    3354,//102
    3374,//103
    3384,//104
    3392,//105
    3409,//106
    3426,//107
    3442,//108
    3458,//109
    3473,//110
    3489,//111
    3503,//112
    3518,//113
    3532,//114
    3545,//115
    3559,//116
    3572,//117
    3585,//118
    3597,//119
    3609,//120
    3621,//121
    3632,//122
    3643,//123
    3654,//124
    3665,//125
    3675,//126
    3685,//127
    3695,//128
    3705,//129
    3714,//130
};
```

```
/**********全局变量**********/
uint16_t ADC_Val;//adc 的值

/*
*********************************************************************
 * 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;

    HAL_ADC_Start(&hadc);//开始读取 adc 的值
    ADCvalue_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 += 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：uint16_t 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 Get_Ntc_Temp(uint16_t value_adc)
{
    uint8_t R100k_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 < R100K_TAB[0])
    {
        return 3;
    }

    else if(value_adc > R100K_TAB[R100k_Tab_Size - 1])
    {
        return 4;
    }

    index_l = 0;
    index_r = R100k_Tab_Size - 1;
    for(; index_r - index_l > 1;)
    {
        if((value_adc <= R100K_TAB[index_r]) && (value_adc > R100K_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(R100K_TAB[index_l] == value_adc)
    {
        temp = (((int)index_l) - 10) * 10; //rate *10
    }
    else if(R100K_TAB[index_r] == value_adc)
    {
        temp = (((int)index_r) - 10) * 10; //rate *10
    }
    else
    {
        if(R100K_TAB[index_r] - R100K_TAB[index_l] == 0)
        {
            temp = (((int)index_l) - 10) * 10; //rate *10
        }
        else
        {
            temp = (((int)index_l) - 10) * 10 + ((value_adc - R100K_TAB[index_l] ) * 100 + 5)
```

```
/ 10 / (R100K_TAB[index_r] - R100K_TAB[index_l]);
            }
        }

        return temp;
}

/*
*********************************************************************
 * 函数原型：uint16_t Get_ADCVal(int16_t temp)
 * 功      能：计算当前温度的 adc 值
 * 输      入：temp：当前温度
 * 输      出：ADC 值
 * 参      数：uint16_t temp
*********************************************************************
*/
uint16_t Get_ADCVal(int16_t temp)
{
        int16_t adc,adc1;
        float val2;
        int16_t val3,val1;

        val3 = (temp/10)+10;
        val2 = (float)(temp%10)/10;

        val1 = ((temp+10)/10)+10;

        adc = R100K_TAB[val3];
        adc1 = R100K_TAB[val1];
        return adc+((adc1-adc)*val2);
}

/*
*********************************************************************
 * 函数原型：void Read_Temp(float dT)
 * 功      能：读取温度-10ms
*********************************************************************
*/
void Read_Temp(float dT)
{
        static float T;
        T += dT;

        ADC_Val = Filter_ADC();//滤波获取 adc 的滑动平均值

        if(T >= 1.0f)//1S
        {
            Temp.Rel = Get_Ntc_Temp(ADC_Val);//计算温度
            T = 0;
        }
```

```
}


/*
**********************************************************************
 * 函数原型：void Temp_Init(void)
 * 功      能：开机显示温度
**********************************************************************
*/
void Temp_Init(void)
{
      for(uint8_t i =0;i<=100;i++)
      {
            ADC_Val = Filter_ADC();
            if(i == 100)//1S
            {
                  Temp.Rel = Get_Ntc_Temp(ADC_Val);//计算温度
                  Temp.Display_Rel = Temp.Rel;//用来显示
            }
            HAL_Delay(10);//没有延时开机读不出温度
      }
}
#include "Drv_KEY.h"

/**********全局变量**********/
float Key_Status;//在操作按键时

/**********局部变量声明******/
float Key_Cnt1,Key_Cnt2,Key_Cnt3,Key_Cnt4,Key_Cnt5;//按下时间
uint8_t Key_Flag1,Key_Flag2,Key_Flag3,Key_Flag4,Key_Flag5;//按键按下标志
uint8_t LongPress1,LongPress2,LongPress3,LongPress4,LongPress5;//按键长按标志

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

/*
**********************************************************************
 * 函数原型：void Key_Scan(float dT)
 * 功      能：按键扫描
**********************************************************************
*/
```

```
void Key_Scan(float dT)
{
    /*************************************MENU                                    键
*************************************/
    if(!Key1)//按下按键
    {
        if(sys.Run_Status == 1)
            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)//按键被按下
    {
        if(Key1)//抬起按键
        {
            if(Key_Cnt1 > 0.05f && Key_Cnt1 < 1.5)//按键时间大于 0.05S 小于 1.5S 是单击
            {
                sys.SetMode_Option++;
                if(PMode.Show_Circle)
                {
                    if(sys.SetMode_Option > 3)
                    {
                        sys.SetMode_Option = 0;
                    }
                    Twinkle_Time = 6.0f;
                    Key_Status = 0;
                    Beep_Time = 0.1;
                }
                else
                {
                    if(sys.SetMode_Option > 2)
                    {
                        sys.SetMode_Option = 0;
                    }
                    Twinkle_Time = 6.0f;
                    Key_Status = 0;
                    Beep_Time = 0.1;
                }
            }
            Key_Flag1 = 0;//按键事件结束，等待下一次按下
            LongPress1 = 0;//长按标志清零
            Key_Cnt1 = 0;//按钮计数清零
        }
        if(Key_Cnt1 > 1.5 && Key_Cnt1 < 3.0)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress1 == 0)//如果没有一直一直长按着
            {
```

```
                LongPress1 = 1;//长按标志置一
            }
        }
    }

    /************************************ 加 键
****************************************/
    if(!Key2)//按下按键
    {
        if(sys.Run_Status == 1)
            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(Key2)//抬起按键
        {
            if(Key_Cnt2 < 1.5)//小于 1.5S 是单击
            {
                if(sys.SetMode_Option == 1)
                {
                    Temp.Set++;
                    if(Temp.Set > Temp_MAX)
                    {
                        Temp.Set = Temp_MAX;
                    }
                }
                else if(sys.SetMode_Option == 2)
                {
                    if(Time.Set < 3600)
                        Time.Set += 5;
                    else
                        Time.Set += 60;
                    if(Time.Set > Time_MAX)
                    {
                        Time.Set = Time_MAX;
                    }
                }
                else if(sys.SetMode_Option == 3)
                {
                    PMode.Option ++;
                    if(PMode.Option > 9)
                    {
                        PMode.Option = 1;
                    }
                    Param_Read();
                }
                Key_Status = 2.0f;
```

```
            Twinkle_Time = 6.0f;
        }
        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)
        {
            Temp.Set += 1;
            if(Temp.Set > Temp_MAX)
            {
                Temp.Set = Temp_MAX;
            }
            if(Temp.Set % 10 == 0)
            {
                Key_Cnt2 = 1.8;//按钮计数清零
            }
            else
            {
                Key_Cnt2 = 1.88f;//按钮计数清零
            }
        }
        else if(sys.SetMode_Option == 2)
        {
            if(Time.Set < 3600)
            {
                Time.Set += 5;
                if(Time.Set>3600)
                    Time.Set = 3600;
                if(Time.Set % 60 == 0)
                {
                    Key_Cnt2 = 1.84f;//按钮计数清零
                }
                else
                {
                    Key_Cnt2 = 1.88f;//按钮计数清零
                }
            }
            else
            {
                Time.Set += 300;
                if(Time.Set % 600 == 0)
                {
                    Key_Cnt2 = 1.84f;//按钮计数清零
                }
                else
                {
                    Key_Cnt2 = 1.88f;//按钮计数清零
                }
```

```
                }
                if(Time.Set > Time_MAX)
                {
                    Time.Set = Time_MAX;
                }
            }
            else if(sys.SetMode_Option == 3)
            {
                PMode.Option ++;
                if(PMode.Option > 9)
                {
                    PMode.Option = 1;
                }
                Key_Cnt2 = 1.66;//按钮计数清零
            }
            Key_Status = 2.0f;
            Twinkle_Time = 6.0f;
            Key_Flag2 = 0;//按键事件结束，等待下一次按下
        }
    }
    else
        Key_Cnt2 = 0;//按钮计数清零

    /************************************ 减 键
************************************/
    if(!Key3)//按下按键
    {
        if(sys.Run_Status == 1)
            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(Key3)//抬起按键
        {
            if(Key_Cnt3 < 1.5)//按键时间大于 0.05S 小于 1.5S 是单击
            {
                if(sys.SetMode_Option == 1)
                {
                    Temp.Set--;
                    if(Temp.Set < Temp_MIN)
                    {
                        Temp.Set = Temp_MIN;
                    }
                }
                else if(sys.SetMode_Option == 2)
                {
                    if(Time.Set < 3600)
                        Time.Set -= 5;
```

```
                else
                {
                    Time.Set -= 60;
                    if(Time.Set < 3600)
                        Time.Set = 3595;
                }
                if(Time.Set < Time_MIN)
                {
                    Time.Set = Time_MIN;
                }
            }
            else if(sys.SetMode_Option == 3)
            {
                PMode.Option --;
                if(PMode.Option < 1)
                {
                    PMode.Option = 9;
                }
                Param_Read();
            }
            Key_Status = 2.0f;
            Twinkle_Time = 6.0f;
        }
        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
    }
    else 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)
        {
            Temp.Set -= 1;
            if(Temp.Set < Temp_MIN)
            {
                Temp.Set = Temp_MIN;
            }
            if(Temp.Set % 10 == 0)
            {
                Key_Cnt3 = 1.8;//按钮计数清零
            }
            else
            {
                Key_Cnt3 = 1.88f;//按钮计数清零
            }
        }
        else if(sys.SetMode_Option == 2)
        {
            if(Time.Set < 3600)
            {
                Time.Set -= 5;
                if(Time.Set % 60 == 0)
```

```
                    {
                        Key_Cnt3 = 1.84f;//按钮计数清零
                    }
                    else
                    {
                        Key_Cnt3 = 1.88f;//按钮计数清零
                    }
                }
                else
                {
                    Time.Set -= 300;
                    if(Time.Set < 3600)
                        Time.Set = 3540;
                    if(Time.Set % 600 == 0)
                    {
                        Key_Cnt3 = 1.84f;//按钮计数清零
                    }
                    else
                    {
                        Key_Cnt3 = 1.88f;//按钮计数清零
                    }
                }
                if(Time.Set < Time_MIN)
                {
                    Time.Set = Time_MIN;
                }
            }
            else if(sys.SetMode_Option == 3)
            {
                PMode.Option --;
                if(PMode.Option < 1)
                {
                    PMode.Option = 9;
                }
                Key_Cnt3 = 1.66;//按钮计数清零
            }
            Key_Status = 2.0f;
            Twinkle_Time = 6.0f;
            Key_Flag3 = 0;//按键事件结束，等待下一次按下
        }
    }
    else
        Key_Cnt3 = 0;//按钮计数清零

    /*************************************Start 键
**************************************/
    if(!Key4)//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)//按键被按下
    {
        if(Key4)//抬起按键
        {
            if(Key_Cnt4 > 0.05f && Key_Cnt4 < 1.5)//按键时间大于 0.05S 小于 1.5S 是单击
            {
                if(sys.Run_Status != 1)
                {
                    sys.Run_Status = 1;//系统启动
                    sys.SetMode_Option = 0;//清除设置
                    Temp_Val.Integral = 40;//开始就 5 分之一功率加热
                    SetOK_Flag = 1;//速度设置，时间设置
                    Twinkle_Time = Key_Status =0;//闪烁时间清零
                    Temp.Last_Set = Temp.Set;//记录设定值
                    Temp.Last_Mode = 0;//记录关闭时的显示模式
                    Beep_Time = 0.1;//蜂鸣器响 0.1S
                }
                else
                {
                    sys.Run_Status = 2;//系统关闭
                    Temp.Last_Mode = Temp.ADDMode;//记录关闭时的显示模式
                    Temp.ADD_Wait_Count = 0;//清楚缓慢计时数
                    SetOK_Flag = 1;//速度设置，时间设置
                    Beep_Time = 0.1;//蜂鸣器响 0.1S
                    Temp.ADDMode = 0;//加热显示模式清零
                }
            }
            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 < 3.0)//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
            }
        }
    }
    /*************************************P                                    键
*****************************************/
    if(!Key5)//按下按键
    {
        if(sys.Run_Status == 1)
            return;
```

```
            if(LongPress5 == 0)//没有长按过
            {
                Key_Cnt5 += dT;//按下时间++
                Key_Flag5 = 1;//按键按下标志置一
            }
        }
        if(Key_Flag5)//按键被按下
        {
            if(Key5)//抬起按键
            {
                if(Key_Cnt5 > 0.05f && Key_Cnt5 < 1.5)//按键时间大于 0.05S 小于 1.5S 是单击
                {
                    if(PMode.Show_Circle == 0)
                    {
                        PMode.Show_Circle = 1;//显示外框和 P-几模式
                        PMode.Option = 1;//记忆模式 1
                        Param_Read();//读取参数
                        SetOK_Flag = 1;//设置参数
                        sys.SetMode_Option = 3;//进入设置 P 值模式
                        Twinkle_Time = 6;//闪烁时间 6S
                    }
                    else
                    {
                        PMode.Show_Circle = 0;//不显示外框和 P-几模式
                        PMode.Option = 0;//记忆模式 0
                        Param_Read();//读取参数
                        sys.SetMode_Option = 0;//模式清零
                        SetOK_Flag = 1;//设置参数
                    }
                    Beep_Time = 0.1;//蜂鸣器响 0.1S
                }
                Key_Flag5 = 0;//按键事件结束，等待下一次按下
                LongPress5 = 0;//长按标志清零
                Key_Cnt5 = 0;//按钮计数清零
            }
            if(Key_Cnt5 > 1.5 && Key_Cnt5 < 3.0)//按键时间大于 1.5S 小于 3S 表示长按
            {
                if(LongPress5 == 0)//如果没有一直一直长按着
                {

                    LongPress5 = 1;//长按标志置一
                }
            }
        }
}
#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_HEAT.h"

/*
*****************************************************************
 * 函数原型：void HEAT_Init(void)
 * 功　　能：初始化加热
*****************************************************************
*/
//55-75
void HEAT_Init(void)
{
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);//开启 tim3 通道二
}
#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,Time_ShowFlag,Pmode_Flag;//温度、时间显示的标志位 0:常亮 1：熄灭
```

```
uint8_t TempIcn_ShowFlag,TimeIcn_ShowFlag;//加热图标闪烁和时间图标闪烁

/*
*********************************************************************
 * 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 == 1)
    {
        T += dT;
        if(T >= 0.5f)
        {
            if(Temp.Set)
                TempIcn_ShowFlag = ~TempIcn_ShowFlag;//温度图标闪烁;
            if(Time.Rel)
                TimeIcn_ShowFlag = ~TimeIcn_ShowFlag;//开盖图标闪烁;
            T = 0;
        }
    }
    else
    {
        TempIcn_ShowFlag = 0;//不显示温度图标
        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
    {
        Temp_ShowFlag = 0;//常亮
        Time_ShowFlag = 0;//常亮
        Pmode_Flag = 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
                Time_ShowFlag = 0;//时间常亮
                Pmode_Flag = 0;//常亮
            }
            else if(sys.SetMode_Option == 2)//设置时间
            {
                Temp_ShowFlag = 0;//温度常亮
                Time_ShowFlag = ~Time_ShowFlag;//时间闪烁
                Pmode_Flag = 0;//常亮
            }
            else if(sys.SetMode_Option == 3)//设置 PMode
            {
                Temp_ShowFlag = 0;//温度常亮
                Time_ShowFlag = 0;//时间闪烁
                Pmode_Flag = ~Pmode_Flag;//常亮
            }
            if(Twinkle_Time == 0)//如果闪烁结束
            {
                sys.SetMode_Option = 0;//模式选择清零
            }
            T = 0;
        }
    }
    else
    {
        Temp_ShowFlag = 0;//常亮
        Time_ShowFlag = 0;//常亮
        Pmode_Flag = 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,seg9;
    seg1=0;seg2=0;seg3=0;seg4=0;seg5=0;seg6=0;seg7=0;seg8=0;seg9=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 > 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 = 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;//℃

    /**************加热图标显示************/
    if(sys.Run_Status == 1 && TempIcn_ShowFlag == 0)
    {
        seg9 &= 0xFB;seg9 |= 0x04;
    }
    else
    {
        seg9 &= 0xFB;seg9 |= 0x00;
    }

    /**************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
    Write_Addr_Dat_N(16, seg9,1);//SEG19
}
```

```
/*
**********************************************************************
 * 函数原型：void Display_Time(int32_t dis_set_time,int32_t dis_rel_time)
 * 功      能：显示时间
 * 输      入: dis_set_time  设定转速   dis_rel_time  实际转速
 * 参      数：int32_t dis_set_time,int32_t dis_rel_time
**********************************************************************
*/
void Display_Time(int32_t dis_set_time,int32_t dis_rel_time)
{
    uint8_t seg10,seg11,seg12,seg13,seg14,seg15,seg16,seg17,seg18,seg19;
    seg10=0;seg11=0;seg12=0;seg13=0;seg14=0;seg15=0;seg16=0;seg17=0;seg18=0;seg19=0;
    uint8_t Time_QU,Time_BU,Time_SU,Time_GU;//实际时间的计算位数取值
    uint8_t Time_QD,Time_BD,Time_SD,Time_GD;//设定时间的计算位数取值
    uint8_t SH,H,SM,M;//时间的单位取值

    /***********设定时间计算**********/
    if(Time_ShowFlag == 0)
    {
        if(dis_set_time)
        {
            if(Time.Set < 3600)
            {
                SH=dis_set_time%3600/60/10;//计算十位单位的分钟数
                H=dis_set_time%3600/60%10;//计算个位单位的分钟数
                SM=dis_set_time%60/10;//计算十分位单位的秒钟数
                M=dis_set_time%60%10;//计算十分位单位的秒钟数
            }
            else
            {
                SH=dis_set_time/3600/10;//计算十位单位的小时数
                H=dis_set_time/3600%10;//计算个位单位的小时数
                SM=dis_set_time%3600/60/10;//计算十分位单位的分钟数
                M=dis_set_time%3600/60%10;//计算个分位单位的分钟数
            }
            Time_QD = Tab[SH];
            Time_BD = Tab[H];
            Time_SD = Tab[SM];
            Time_GD = Tab[M];
        }
        else
        {
            Time_QD = 0x04;
            Time_BD = 0x04;
            Time_SD = 0x04;
            Time_GD = 0x04;
        }
    }
    else
    {
```

```
        Time_QD = 0x00;//不显示设定速度
        Time_BD = 0x00;//不显示设定速度
        Time_SD = 0x00;//不显示设定速度
        Time_GD = 0x00;//不显示设定速度
    }

    /***********实际时间计算**********/
    if(dis_rel_time)
    {
        if(Time.Set < 3600)
        {
            SH=dis_rel_time%3600/60/10;//计算十位单位的分钟数
            H=dis_rel_time%3600/60%10;//计算个位单位的分钟数
            SM=dis_rel_time%60/10;//计算十分位单位的秒钟数
            M=dis_rel_time%60%10;//计算十分位单位的秒钟数
        }
        else
        {
            SH=dis_rel_time/3600/10;//计算十位单位的小时数
            H=dis_rel_time/3600%10;//计算个位单位的小时数
            SM=dis_rel_time%3600/60/10;//计算十分位单位的分钟数
            M=dis_rel_time%3600/60%10;//计算个分位单位的分钟数
        }
        Time_QU = Tab[SH];
        Time_BU = Tab[H];
        Time_SU = Tab[SM];
        Time_GU = Tab[M];
    }
    else
    {
        Time_QU = 0x04;
        Time_BU = 0x04;
        Time_SU = 0x04;
        Time_GU = 0x04;
    }

    /***************rpm******************/
//  seg15 &= 0x7F;seg15 |= 0x80;//rpm
    if(Time.Set < 3600)
    {
        seg17 &= 0xFE;seg17 |= 0x01;//sec
    }
    else
    {
        seg17 &= 0x7F;seg17 |= 0x80;//min
    }

    /***************时间冒号显示**************/
    seg13 &= 0x7F;seg13 |= 0x80;//底下的时间冒号
    seg13 &= 0xFE;seg13 |= 0x01;//上面的时间冒号
```

```
    /***************时间图标显示***************/
    if(TimeIcn_ShowFlag == 0)
    {
        seg10 &= 0xF8;seg10 |= 0x07;
        seg11 &= 0xFD;seg11 |= 0x02;
    }
    else
    {
        seg10 &= 0xF8;seg10 |= 0x00;
        seg11 &= 0xFD;seg11 |= 0x00;
    }

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

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

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

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

    /**************发送数据***************/
    Write_Addr_Dat_N(18, seg10,1);//SEG18
    Write_Addr_Dat_N(20, seg11,1);//SEG17
    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_Pmode(uint8_t dis_option)
```

```
 * 功      能：显示 P 模式
 * 输      入: dis_option：当前的记忆模式
 * 参      数：uint8_t dis_option
*****************************************************************
*/
void Display_Pmode(uint8_t dis_option)
{
    uint8_t seg20,seg21,seg22,seg23,seg24,seg25,seg26,seg27;
    seg20=0;seg21=0;seg22=0;seg23=0;seg24=0;seg25=0;seg26=0;seg27=0;
    uint8_t Val;//计算位数取值

    if(PMode.Show_Circle)
    {

        /**********P 模式外框**********/
        seg20 &= 0x87;seg20 |= 0x78;
        seg21 &= 0x7F;seg21 |= 0x80;
        seg23 &= 0x7F;seg23 |= 0x80;
        seg24 &= 0xF7;seg24 |= 0x08;
        seg25 &= 0x7F;seg25 |= 0x80;
        seg27 &= 0x0F;seg27 |= 0xF0;

        if(Pmode_Flag == 0)
        {
            seg23 &= 0xDF;seg23 |= 0x20;//"-"

            seg25 &= 0xCF;seg25 |= 0x30;//"P"
            seg26 &= 0x8F;seg26 |= 0x70;

            Val = Tab[dis_option];
            seg22 &= 0x0F;seg22 |= Val & 0xF0;
            seg21 &= 0x8F;seg21 |= (Val & 0x0F) << 3;
        }
        else
        {
            seg23 &= 0xDF;seg23 |= 0x00;//"-"

            seg25 &= 0xCF;seg25 |= 0x00;//"P"
            seg26 &= 0x8F;seg26 |= 0x00;

            Val = Tab[dis_option];
            seg22 &= 0x0F;seg22 |= Val & 0x00;
            seg21 &= 0x8F;seg21 |= (Val & 0x00) << 3;
        }
    }
    else
    {
        /**********P 模式外框**********/
        seg20 &= 0x87;seg20 |= 0x00;
        seg21 &= 0x7F;seg21 |= 0x00;
```

```
            seg23 &= 0x7F;seg23 |= 0x00;
            seg24 &= 0xF7;seg24 |= 0x00;
            seg25 &= 0x7F;seg25 |= 0x00;
            seg27 &= 0x0F;seg27 |= 0x00;


            seg23 &= 0xDF;seg23 |= 0x00;//"-"

            seg25 &= 0xCF;seg25 |= 0x00;//"P"
            seg26 &= 0x8F;seg26 |= 0x00;

            Val = 0x00;
            seg22 &= 0x0F;seg22 |= Val & 0xF0;
            seg21 &= 0x8F;seg21 |= (Val & 0x0F) << 3;
      }
      /**************发送数据***************/
      Write_Addr_Dat_N(38, seg20,1);//SEG20
      Write_Addr_Dat_N(40, seg21,1);//SEG21
      Write_Addr_Dat_N(42, seg22,1);//SEG22
      Write_Addr_Dat_N(44, seg23,1);//SEG23
      Write_Addr_Dat_N(46, seg24,1);//SEG24
      Write_Addr_Dat_N(48, seg25,1);//SEG25
      Write_Addr_Dat_N(50, seg26,1);//SEG26
      Write_Addr_Dat_N(52, seg27,1);//SEG27
}

/*
*********************************************************************
 * 函数原型：void ADD_Show(float dT)
 * 功      能：显示上升时间
 * 输      入：dT:执行周期
 * 参      数：float dT
*********************************************************************
*/
void ADD_Show(float dT)
{
      static float T;
      if(Temp.ADD_Wait_Count)//加入慢速上升标志位大于一并且升温状态在慢速上升时
      {
            T += dT;
            if(T >= 1.0f)//1S
            {
                  Temp.ADD_Wait_Count--;//慢速上声标志位--
                  if(Temp.ADD_Wait_Count == 0)//慢速上升标志位等于 0 时
                        Temp.ADDMode = 3;//进入稳定温度模式
                  T = 0;
            }
      }
}
```

```
/*
*********************************************************************
 * 函数原型：void Deal_Temp(float dT)
 * 功      能：温度显示处理
*********************************************************************
*/
void Deal_Temp(float dT)
{
    static float T;

    if(sys.Run_Status == 0)//没启动的情况下
    {
        Temp.ADDMode = 0;
        Temp.Display_Rel = Temp.Rel;
    }
    else if(sys.Run_Status == 1)//启动的情况下
    {
        if(Temp.ADDMode == 0)//判断数据处理显示
        {
            if(Temp.Ctrl > Temp.Display_Rel)//设定温度大于显示温度
            {
                Temp.ADDMode = 1;//进入加热模式下
                Temp.New = 0;//将之前的记入值清零
             }
            else
            {
                Temp.ADDMode = 2;//进入降温模式下
                Temp.New = 0;//将之前的记入值清零
             }
        }
        if(Temp.ADDMode == 1)//在加热模式下
        {
             if(Temp.Rel > Temp.Display_Rel && Temp.Display_Rel <= Temp.Ctrl)
             {
                Temp.Display_Rel ++;
                T = 0;
             }
             if(Temp.Display_Rel >= Temp.Set - 20)//实际温度大于等于设定温度-2℃
             {
                Temp.ADD_Wait_Count = ((Temp.Set-Temp.Display_Rel)*10);//200S 的缓慢
升温显示
                Temp.ADDMode = 3;
                return;
             }
        }
        else if(Temp.ADDMode == 2)//降温模式下
        {
            if(Temp.Rel < Temp.Display_Rel && Temp.Display_Rel >= Temp.Ctrl)//当前温度
小于上一次温度
            {
```

```
                    Temp.Display_Rel--;//显示当前温度
                    T = 0;
                }
                if(Temp.Display_Rel <= Temp.Set + 20)//实际温度小于等于设定温度-2℃
                {
                    Temp.ADD_Wait_Count  =  (-(Temp.Set-Temp.Display_Rel)*10);//200S 的缓慢升温显示
                    Temp_Val.Integral = 0;//积分清零
                    Temp.ADDMode = 3;
                    return;
                }
            }
            else if(Temp.ADDMode == 3)//等待降温后开始升温
            {
                if(Temp.Display_Rel < Temp.Set)
                    Temp.Display_Rel=(Temp.Set-20)+(20-(Temp.ADD_Wait_Count)*2/20);// 缓慢显示数值
                else if(Temp.Display_Rel > Temp.Set)
                    Temp.Display_Rel=(Temp.Set+20)-(20-(Temp.ADD_Wait_Count)*2/20);// 缓慢显示数值
                else
                    Temp.ADDMode = 4;
            }
            else if(Temp.ADDMode == 4)//温度稳定模式下
            {
                Temp.Display_Rel = Temp.Ctrl;//显示当前显示温度
            }
        }
        else if(sys.Run_Status == 2)//关闭的情况下
        {
            if(Temp.Display_Rel < Temp.Rel)
            {
                T += dT;
                if(T >= 10.0f)
                {
                    Temp.Display_Rel -= 1;
                    if(Temp.Display_Rel< 370)
                        sys.Run_Status = 0;
                    T = 0;
                }
            }
            else
            {
                T = 0;
                sys.Run_Status = 0;
            }
        }
}

/*
```

```
*****************************************************************
 * 函数原型：  void Show_Display(void)
 * 功      能：  显示屏幕内容
*****************************************************************
*/
void Show_Display(void)
{
    Temp.Display_Set = Temp.Set;//显示设定温度

    Time.Display_Rel = Time.Rel;//显示控制时间
    Time.Display_Set = Time.Set;//显示设定时间

    Display_Pmode(PMode.Option);

    if(Temp.Display_Rel >= 1000)
        Temp.Display_Rel = 1000;

    Display_Temp(Temp.Display_Set,Temp.Display_Rel);//显示温度
    Display_Time(Time.Display_Set,Time.Display_Rel);//显示时间
}
#include "Param.h"

/**********宏定义************/
struct _Save_Param_ Param;//原始数据

/**********全局变量声明******/
uint8_t Save_Param_En;

/*
*****************************************************************
 * 函数原型：void Param_Reset(void)
 * 功      能：初始化硬件中的参数
*****************************************************************
*/
void Param_Reset(void)
{
    Param.Flash_Check_Start = FLASH_CHECK_START;

    for(uint8_t i=0;i <= 9;i++)
    {
        Param.P_Param[i][0] = 370;//温度
        Param.P_Param[i][1] = 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 = Param.P_Param[PMode.Option][0];//将 Flash 中的温度赋值
        Time.Set = Param.P_Param[PMode.Option][1];//将 Flash 中的时间赋值
        SetOK_Flag = 1;
        Save_Param_En = 1;
    }
    else
    {
        Temp.Set = Param.P_Param[PMode.Option][0];//将 Flash 中的温度赋值
        Time.Set = Param.P_Param[PMode.Option][1];//将 Flash 中的时间赋值
        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();
            Save_Param_En = 0;
        }
    }
    else
        time = 0;
}
#include "SetVal.h"

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

/*
*********************************************************************
 * 函数原型：void Check_Set(float dT)
 * 功    能：检测设置
*********************************************************************
*/
void Check_Set(float dT)
{
    if(Key_Status)
    {
        SetOK_Flag = 1;//检测到设置，等待退出设置模式
    }
    if(SetOK_Flag)
    {
        if(Temp.Ctrl != Temp.Set)
        {
            Temp.Ctrl = Temp.Set;
            Param.P_Param[PMode.Option][0] = Temp.Set;
        }
        if(Time.Rel != Time.Set)
        {
            Time.Rel = Time.Set;
            Param.P_Param[PMode.Option][1] = Time.Set;
        }
        Save_Param_En = 1;//保存
        SetOK_Flag = 0;
    }
}
#include "PID.h"
```

```
/*
**********************************************************************
 * 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, 0, 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
}

/*
**********************************************************************
 * 函数原型：void IncPID_Calculation(float dT,float Expect,float Freedback,_PID_Arg_ * PID_Arg,_PID_Val_ * PID_Val,float Integral_Lim)
 * 功      能：增量式 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
 * 参      数：float dT,float Expect,float Freedback,_PID_Arg_ * PID_Arg,_PID_Val_ * PID_Val
**********************************************************************
*/
void IncPID_Calculation(float dT,float Expect,float Freedback,_PID_Arg_ * PID_Arg,_PID_Val_ * PID_Val,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 - PID_Val->Error_Last);//比例 = 比例系数*（当前误差-上一次误差）
    PID_Val->Integral       = PID_Arg->Ki * PID_Val->Error * 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->Differential = PID_Arg->Kd * (PID_Val->Error - 2.0f*PID_Val->Error_Last + PID_Val->Error_Previous) * safe_div(1.0f,dT,0);//微分 = 微分系数 *（当前误差-2*上一次误差+上上次误差）*频率

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

    PID_Val->Error_Previous = PID_Val->Error_Last;//将上一次误差赋值给上上次误差
    PID_Val->Error_Last = PID_Val->Error;//将当前误差赋值给上一次误差
}
```