DB3100 源程序

```
#include "Ntc.h"

/**********局部变量**********/
const uint16_t R10K_TAB[] = //R25=10K?3% B25/50=4100K?3% 10K??//-20-105
{
    122,//-20
    129,//-19
    137,//-18
    145,//-17
    153,//-16
    161,//-15
    170,//-14
    180,//-13
    189,//-12
    200,//-11
    210,//-10
    221,//-9
    233,//-8
    245,//-7
    258,//-6
    271,//-5
    284,//-4
    298,//-3
    313,//-2
    328,//-1
    344,//0
    361,//1
    378,//2
    395,//3
    413,//4
    432,//5
    452,//6
    472,//7
    493,//8
    514,//9
    536,//10
    559,//11
    582,//12
    606,//13
    631,//14
    656,//15
    682,//16
    709,//17
    736,//18
    764,//19
    793,//20
    822,//21
    852,//22
    882,//23
```

```
914,//24
945,//25
977,//26
1010,//27
1044,//28
1077,//29
1112,//30
1147,//31
1182,//32
1218,//33
1254,//34
1290,//35
1327,//36
1364,//37
1402,//38
1440,//39
1478,//40
1516,//41
1554,//42
1593,//43
1632,//44
1670,//45
1709,//46
1748,//47
1787,//48
1826,//49
1865,//50
1904,//51
1943,//52
1981,//53
2020,//54
2058,//55
2096,//56
2134,//57
2171,//58
2209,//59
2246,//60
2282,//61
2319,//62
2355,//63
2390,//64
2426,//65
2460,//66
2495,//67
2529,//68
2562,//69
2595,//70
2628,//71
2660,//72
2692,//73
```

```
2723,//74
2754,//75
2784,//76
2813,//77
2843,//78
2871,//79
2899,//80
2927,//81
2954,//82
2981,//83
3007,//84
3032,//85
3057,//86
3082,//87
3106,//88
3129,//89
3152,//90
3175,//91
3197,//92
3218,//93
3240,//94
3260,//95
3280,//96
3300,//97
3319,//98
3338,//99
3356,//100
3374,//101
3392,//102
3409,//103
3426,//104
3442,//105
3458,//106
3473,//107
3489,//108
3503,//109
3518,//110
3532,//111
3545,//112
3559,//113
3572,//114
3585,//115
3597,//116
3609,//117
3621,//118
3632,//119
3643,//120
3654,//121
3665,//122
3675,//123
```

```
    3685,//124
    3695,//125
    3705,//126
    3714,//127
    3723,//128
    3732,//129
    3741,//130
    3749,//131
    3757,//132
    3765,//133
    3773,//134
    3781,//135
    3788,//136
    3795,//137
    3802,//138
    3809,//139
    3816,//140
    3822,//141
    3829,//142
    3835,//143
    3841,//144
    3847,//145
    3852,//146
    3858,//147
    3863,//148
    3869,//149
    3874,//150
};

/**********全局变量**********/
int rel_temp;//实际温度
int set_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
}
#define Type_Mode   1//0:常规模块  1：高模块   2：平底
/*
*********************************************************************
 * 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 = 171;
    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) - 18) * 10; //rate *10
    }
    else if(R10K_TAB[index_r] == value_adc)
    {
        temp = (((int)index_r) - 18) * 10; //rate *10
    }
    else
    {
        if(R10K_TAB[index_r] - R10K_TAB[index_l] == 0)
        {

            temp = (((int)index_l) - 18) * 10; //rate *10
        }
        else
        {
            temp = (((int)index_l) - 18) * 10 + ((value_adc - R10K_TAB[index_l] ) * 100 + 5) / 10 / (R10K_TAB[index_r] - R10K_TAB[index_l]);
        }
    }

/*******************温度补偿**********************/
    #if(Type_Mode == 0)//常规模块
    if(set_temp <= 800)//设定温度小于 80℃
        rel_temp = temp-5;//实际温度等于测得温度
    else if((set_temp > 800)&&(set_temp <= 1000))//设定温度在 80 到 100℃之间
        rel_temp = temp + 20;//温度补偿 1℃
    #elif(Type_Mode == 1)//高模块
    if(set_temp <= 400)//设定温度小于 80℃
        rel_temp = temp-20;//实际温度等于测得温度
    else if((set_temp > 400)&&(set_temp <= 750))//设定温度在 80 到 100℃之间
        rel_temp = temp-80;//实际温度等于测得温度
    else if((set_temp > 750)&&(set_temp <= 1000))//设定温度在 80 到 100℃之间
        rel_temp = temp - 100;//温度补偿 1℃
    #elif(Type_Mode == 2)//平底模块
```

```
    if(set_temp <= 400)//设定温度小于 80℃
        rel_temp = temp-10;//实际温度等于测得温度
    else if((set_temp > 400)&&(set_temp <= 750))//设定温度在 80 到 100℃之间
        rel_temp = temp-23;//实际温度等于测得温度
    else if((set_temp > 750)&&(set_temp <= 1000))//设定温度在 80 到 100℃之间
        rel_temp = temp - 100;//温度补偿 1℃
    #endif
    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 "main.h"
#include "adc.h"
#include "tim.h"
#include "gpio.h"
/**********结构体*************/
_sys_ sys;//系统
void SystemClock_Config(void);
int main(void)
{

    sys.Init_ok = 0;
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_TIM1_Init();
  MX_ADC_Init();
  MX_TIM3_Init();
  /* USER CODE BEGIN 2 */
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);//开启 tim1 通道 2 的 pwm
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);//开启 tim3 通道 2 的 pwm
    VLED = 160;//设置 pwm 值为 160，控制背光电压
    HEAT = 0;//加热模块 pwm 0—400
    lcd_init();//lcd 初始话
```

```
    HAL_Delay (5);
    Sys_Init();//系统参数初始化
    /************系统初始化成功**************/
    sys.Init_ok = 1;
  while (1)
  {
    LCD_Display();//显示界面

  }
}

void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};


  RCC_OscInitStruct.OscillatorType                                    =
RCC_OSCILLATORTYPE_HSI14|RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSI14State = RCC_HSI14_ON;
  RCC_OscInitStruct.HSI14CalibrationValue = 16;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL4;
  RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                |RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
}
void Error_Handler(void)
{
  __disable_irq();
  while (1)
  {
  }
}

void assert_failed(uint8_t *file, uint32_t line)
```

```
{

}
#endif
#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 ADD_Wait_Count;//升温显示缓慢上升
uint8_t SetTime_State;//未设定时间显示“----”

/**********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 ADD_Show(uint16_t dT)
 * 功      能：  显示上升时间
 * 输      入:  dT:执行周期
 * 参      数：  uint16_t dT
*********************************************************************
*/
void ADD_Show(uint16_t dT)
{
      static uint16_t T;
```

```
    if(ADD_Wait_Count && ADD_Mode == 9)//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 == 1000)//1S
        {
            ADD_Wait_Count--;//慢速上声标志位--
            if(ADD_Wait_Count == 0)//慢速上升标志位等于 0 时
                ADD_Mode = 2;//进入稳定温度模式
            T = 0;
        }

    }
}

/*
**********************************************************************
 * 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
    }
}

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

    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
    if(SetTime_State)//未设定时间显示“----”
    {
        Tab[3]= 0x08;
        Tab[2]= 0x08;
        Tab[1]= 0x08;
        Tab[0]= 0x08;
    }

//      Tab[0]=Tab[0]|0x80;//制冷图标

    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)//未设定时间显示“----”
    {
        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
    }

    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);
}

/*
```

```
*********************************************************************
 * 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
    {
        if(Dis_Rel_Temp <= rel_temp)//如果显示温度小于实际温度
        {
            Dis_Rel_Temp = Dis_Rel_Temp;//显示当前显示的温度
        }
        else//显示温度大于或者等于实际温度时
        {
            ADD_Mode = 8;//显示温度跟着时间--
        }
    }
    else//启动的情况下
    {
        if(ADD_Mode == 0)//判断数据处理显示
        {//加热降温前，显示温度必须和实际温度相同
            if(Dis_Rel_Temp != rel_temp)//显示温度不等于实际温度的话
            {
                if(Dis_Rel_Temp < rel_temp)//显示温度小于实际温度的话
                {
                    Temp_Control = 0;//关闭温度控制
                    if(Dis_Rel_Temp <= set_temp && (rel_temp > set_temp - 20 ))//设定温
度大于等于显示温度
                        ADD_Mode = 4;//进入显示温度随时间++
                    else
                        ADD_Mode = 7;//等待实际降温下来
                }
                else//显示温度大于等于实际温度的话
                {
                    if(Dis_Rel_Temp <= set_temp)//显示温度小于等于设定温度
                    {
                        Temp_Control = 1;//开始加热
                        ADD_Mode = 6;//等待实际温度加热上来
                    }
                    else
                    {
                        Temp_Control = 1;//开启加热
                        ADD_Mode = 5;//进入温度--
                    }
                }
            }
            else//显示温度等于实际温度的话
            {
```

```
                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
                }
                else
                {
                    Temp_Control = 1;//停止加热
                    ADD_Mode = 3;//进入降温模式下
                }
            }
            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 - 20)//实际温度大于等于设定温度-2℃
            {
                ADD_Mode = 9;//进入最后的缓慢升温模式
                ADD_Wait_Count = -(Dis_Rel_Temp-set_temp)*10;//200S 的缓慢升温显示
            }
        }
        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
        }
        else if(ADD_Mode == 3)//降温模式下
        {
            if( Dis_Rel_Temp < rel_temp)
            {
                Dis_Rel_Temp = Dis_Rel_Temp;
            }
            else
            {
                Dis_Rel_Temp = rel_temp;//显示温度等于实际温度
                if(set_temp >= (rel_temp-3))//在快降到显示温度时
                {
                    Temp_Control = 1;//开启加热
                    ADD_Mode = 2;//温度稳定模式
                }
```

```
                }
            }
            else if(ADD_Mode == 6)//等待降温
            {
                Dis_Rel_Temp = Dis_Rel_Temp;//显示温度等于显示温度
                if(Dis_Rel_Temp <= rel_temp)//显示温度小于等于实际温度时
                {
                    Dis_Rel_Temp = rel_temp;//显示温度等于实际温度
                    ADD_Mode = 0;//重新判断
                }
            }
            else if(ADD_Mode == 7)//等待降温后开始升温
            {
                Dis_Rel_Temp = Dis_Rel_Temp;//显示温度等于显示温度
                if(Dis_Rel_Temp >= rel_temp)//显示温度小于等于实际温度时
                {
                    Dis_Rel_Temp = rel_temp;//显示温度等于实际温度
                    Temp_Control = 1;//开启加热
                    ADD_Mode = 0;//重新判断
                }
            }
            else if(ADD_Mode == 9)//等待降温后开始升温
            {
                Dis_Rel_Temp=(set_temp-20)+(20-(ADD_Wait_Count)*2/20);//缓慢显示数值
                if(rel_temp >= set_temp && Dis_Rel_Temp == set_temp)
                   ADD_Mode = 2;
            }
        }
}

/*
**********************************************************************
 * 函数原型：  void Time_Contole_TempDown(uint16_t dT)
 * 功      能：  时间控制温度下降
 * 输      入:  dT：运行周期
 * 参      数：  uint16_t dT
**********************************************************************
*/
void Time_Contole_TempDown(uint16_t dT)
{
    static uint16_t T;//记入周期
    if(ADD_Mode == 4)//时间控制显示温度上降模式
    {
        T += dT;//运行时间记入
        if(T == 500)//1S
        {
            Dis_Rel_Temp ++;
            if(Dis_Rel_Temp >= rel_temp || Dis_Rel_Temp == set_temp -20)//假如显示温度大
于等于实际温度时
            {//等实际温度降到显示温度数值的时候
```

```
                Dis_Rel_Temp = rel_temp;//显示实际温度
                Temp_Control = 1;//开启加热
                if(rel_temp >= set_temp && Dis_Rel_Temp == set_temp)
                  ADD_Mode = 2;//重新判断加热
              }
               T = 0;//周期清零
            }
        }
        else if(ADD_Mode == 5)//开启时时间控制显示温度下降模式
        {
             T += dT;//运行时间记入
             if(T == 500)//1S
             {
                  Dis_Rel_Temp --;//显示温度--
                  if(Dis_Rel_Temp <= set_temp)
                  {
                       Dis_Rel_Temp = set_temp;
                  }
                  if(Dis_Rel_Temp <= rel_temp)//假如显示温度小于等于实际温度时
                  {//时间下降到实际温度时
                       Dis_Rel_Temp = rel_temp;//显示实际温度
                       Temp_Control = 1;//开始加热
                       ADD_Mode = 0;//重新判断加热
                  }
                  T = 0;//周期清零
             }
        }
        else if(ADD_Mode == 8)//关闭后控制显示温度下降模式
        {
             T += dT;//运行时间记入
             if(T == 500)//0.5S
             {
                  if(Dis_Rel_Temp <= rel_temp)//假如显示温度小于等于实际温度时
                  {//时间下降到实际温度时
                       Dis_Rel_Temp = rel_temp;//显示实际温度
                  }
                  else
                  {
                       Dis_Rel_Temp --;//显示温度--
                  }
                  T = 0;//周期清零
             }
        }
}

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

/**********全局变量**********/
uint16_t run_mode = 1;//运行模式

/**********局部变量**********/
uint16_t cur=300;//连续按加快加减速度
uint16_t Scan_Status=0;//快速加减标志
uint8_t KEY1_Pin_ON=0;//长按标志
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,KEY3_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--;//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
                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
        }
        if(Select_Option == 3)//在设置模式选项
        {
            save_buf [0] = set_temp;//每次按下将当前温度记录
            save_buf [1] = set_time;//每次按下将当前时间记录
            flash_write_buf(flash_addr + 0x1000 * (run_mode - 1), save_buf, 2);//写入到 Flash
            run_mode--;//P 记忆位置--
            if(run_mode < 1)//小于 1 时返回到第九个位置
            {
                run_mode = 9;
            }
            set_temp = (*((uint16_t *) (flash_addr + 0x1000 * (run_mode - 1))));//读取温度
            if(set_temp > 1000)//如果设定温度大于 100℃
            {
                flash_write_buf(flash_addr + 0x1000 * (run_mode - 1), init_buf, 2);//将初始
数组写入到 Flash 37℃  20：00
                set_temp = (*((uint16_t *) (flash_addr + 0x1000 * (run_mode - 1))));//将温度
写入 Flash
            }
            set_time = (*((uint16_t *) (flash_addr + 0x1000 * (run_mode - 1) + 4)));//将时间
写入 Flash
            rel_time = set_time;//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
    }

/*****************************加*************************************/
    if((Key_Scan(GPIOB,KEY2_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++;//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
        }
        if(Select_Option == 3)//在设置模式选项
        {
            save_buf [0] = set_temp;//每次按下将当前温度记录
            save_buf [1] = set_time;//每次按下将当前时间记录
            flash_write_buf(flash_addr + 0x1000 * (run_mode - 1), save_buf, 2);//写入到 Flash
            run_mode++;//P 记忆位置++
            if(run_mode > 9)//大于 9 时返回到第一个位置
            {
                run_mode = 1;
            }
            set_temp = (*((uint16_t *) (flash_addr + 0x1000 * (run_mode - 1))));//读取温度
            if(set_temp > 1000)//如果设定温度大于 100℃
            {
                flash_write_buf(flash_addr + 0x1000 * (run_mode - 1), init_buf, 2);//将初始
数组写入到 Flash 37℃ 20：00
                set_temp = (*((uint16_t *) (flash_addr + 0x1000 * (run_mode - 1))));//将温度
写入 Flash
            }
             set_time = (*((uint16_t *) (flash_addr + 0x1000 * (run_mode - 1) + 4)));//将时间
写入 Flash
            rel_time = set_time;//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
    }

/*****************************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
            if(Set_Mode_Enable == 1)//在 P 模式下
            {
                Sove_Flag = 1;//检测标志保存
            }
        }
        else if(Select_Option >= 4)//从 P 模式出来，Select_Option = 4
            Select_Option = 1;//让他直接进入设定温度模式
        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
    }

/*******************************P 键************************************/
    if((Key_Scan(GPIOB,KEY5_Pin) == KEY_ON))//P 键
    {
        if(Run_Status > 0) //运行中不能设置
            return;
        KEY1_Pin_ON++;
        if(KEY1_Pin_ON == 4)//长按（这里让他初始化数值）
        {
            Set_Mode_Enable = 0;//不显示 P 模式
            run_mode_flag = 0;//不显示 P 模式框
            set_time=1200;//退出 P 模式设置设定时间为 20min
            rel_time=1200;//退出 P 模式设置实际时间为 20min
            set_temp=370;//退出 P 模式设置设定温度 37℃
            Select_Option = 0;//不闪烁设置
        }
        else if(KEY1_Pin_ON == 1)//按一下
        {
            Set_Mode_Enable = 1;//显示 p 模式的框
            Select_Option = 3;//进入设定 p 的位置
            set_temp = (*((uint16_t *) (flash_addr + 0x1000 * (run_mode - 1))));//读取温度
            if(set_temp > 1000)//如果设定温度大于 100℃
            {
                flash_write_buf(flash_addr + 0x1000 * (run_mode - 1), init_buf, 2);//将初始
数组写入到 Flash 37℃  20：00
                set_temp = (*((uint16_t *) (flash_addr + 0x1000 * (run_mode - 1))));//将温度
写入 Flash
            }
            set_time = (*((uint16_t *) (flash_addr + 0x1000 * (run_mode - 1) + 4)));//将时间
写入 Flash
            rel_time = set_time;//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
            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
        {
            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
            if(Set_Mode_Enable == 1)//在 P 模式下
            {
                Sove_Flag = 1;//检测标志保存
            }
        }
        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
            if(Set_Mode_Enable == 1)//只有在 P 模式下读取
            {
                set_temp = (*((uint16_t *) (flash_addr + 0x1000 * (run_mode - 1))));
                set_time = (*((uint16_t *) (flash_addr + 0x1000 * (run_mode - 1) + 4)));
            }
            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 "User.h"

/*
**********************************************************************
版本更新
2021-12.29：实际时间显示“-- --”
2022-01.05：常规和高模块整合加平底
**********************************************************************
```

```
*/


/**********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)
{
    set_time=1200;//开机设置设定时间为 20min
    rel_time=1200;//开机设置实际时间为 20min
    set_temp=370;//开机设置设定温度 37℃
    set_mode_p = 0;//开机进入 P 模式在记忆模式下（本代码不操作此寄存器，单加热）
    mode_run_p1=1;//梯度模式 p1 的值（本代码不操作此寄存器，单加热）
    mode_run_p2=1;//梯度模式 P2 的值（本代码不操作此寄存器，单加热）
    Run_Status = 0;//未启动
    circle_dis = 13;//梯度模式下外圈转动显示（本代码不操作此寄存器，单加热）
    HAL_ADC_Start(&hadc);//开始读取 adc 的值
    HAL_Delay(10);//没有延时开机读不出温度
    res = func_get_ntc_temp(HAL_ADC_GetValue(&hadc));//将 adc 的值储存
    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
    Dis_Rel_Temp = rel_temp;
    PID_Init();//PID 系数初始化
}
```