Mian.c

```
#include "stm32f10x.h"
#include "delay.h"
#include "usart1.h"
#include "adc1.h"
#include "relay.h"
#include "stdio.h"
/*----------全局变量----------*/
extern __IO uint16_t ADCConvertedValue1;//AD 值
extern __IO uint16_t ADCConvertedValue2;//AD 值
extern __IO uint32_t DMA_ADC_OK;//AD 采集完成标志
#define Max_Voltage1 1.0 //上限
#define Max_Voltage2 2.0 //上限
#define Min_Voltage1 0.7 //下限
#define Min_Voltage2 1.7 //下限
/*----------主程序---------*/
int main(void)
{
    float voltage1,voltage2;//电压值
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
    RELAY_Configure();
    USART1_Configure();//USART 初 始 化
    ADC_Configure();//ADC 初始化
    delay_ms(100);// 延时以显示打印字符
    while(1)
    {
        delay_ms(1000);//延时 1s
        voltage1 =(float)ADCConvertedValue1*(3.3/4096);//AD 值转换为电压值
        voltage2 =(float)ADCConvertedValue2*(3.3/4096);//AD 值转换为电压值
        printf("热电偶:%.2fV\r\n",voltage1);//打印电压值，两位小数
        printf("气体:%.2fV\r\n",voltage2);//打印电压值，两位小数
        if(voltage1 > Max_Voltage1)
        {
            RELAY1_On();
        }
        else if(voltage1 < Min_Voltage1)
        {
            RELAY1_Off();
        }
        if(voltage2 > Max_Voltage2)
        {
            RELAY1_Off();
            RELAY2_On();
        }
```

```
            else if(voltage2 < Min_Voltage2)
            {
                RELAY2_Off();
            }
            ADC_SoftwareStartInjectedConvCmd(ADC1, ENABLE);
        }
}
```

Usart1.c

```
#include "usart1.h"
#include "stdio.h"
uint8_t USART1_RX_Buffer[USART_RX_MAX] = { 0 }; //定义 1.USART1 接收缓存
uint8_t USART1_RX_Index = 0; //定义 2.USART1 接收数组下标
uint8_t USART1_RX_OverFlag = 0; //定义 3.USART1 接收完成标志位
/**
  * @简介：将 C 库中 printf 重定向到 USART
  * @参数: ch-待发送字符，f-指定文件
  * @返回值：ch
  */
int fputc(int ch, FILE *f)
{
    USART_SendData(USART1, (u8) ch);
    while(!(USART_GetFlagStatus(USART1, USART_FLAG_TXE) == SET))
    {
    }
    return ch;
}

void USART1_Configure(void)
{
    /* 定义 GPIO 初始化结构体 */
    GPIO_InitTypeDef GPIO_InitStructure;
    /* 定义 USART 初始化结构体 */
    USART_InitTypeDef USART_InitStructure;
    NVIC_InitTypeDef NVIC_InitStructure;
    /* 打开 GPIOA、AFIO 和 USART1 时钟 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_AFIO |
RCC_APB2Periph_USART1, ENABLE);
    /* 配置 PA9(USART_Tx)为复用推挽输出，IO 速度 50MHz */
```

```
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    /* 完成配置 */
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    /* 配置 PA10(USART1_Rx)为浮空输入 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    /* 完成配置 */
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    /* 配置 USART 波特率、数据位、停止位、奇偶校验、硬件流控制和模式 */
    USART_InitStructure.USART_BaudRate = 9600;//波特率 115200
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;//8 数据位
    USART_InitStructure.USART_StopBits = USART_StopBits_1;//1 停止位
    USART_InitStructure.USART_Parity = USART_Parity_No;//无奇偶校验
    USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;//
无硬件流控制
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;//接收和发送模
式
    /* 完成配置 */
    USART_Init(USART1, &USART_InitStructure);
    /* 使能 USART1 */
    USART_Cmd(USART1, ENABLE);
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE); //开启接收 RXNE 中断
    USART_ITConfig(USART1, USART_IT_IDLE, ENABLE); //开启接收 IDLE 中断
    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn; //USART1 中断通道
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; //抢占优先级 1
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1; //子优先级 3
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE; //IRQ 通道使能
    NVIC_Init(&NVIC_InitStructure); //配置生效
}

void USART1_IRQHandler(void)
{
    uint8_t Res;
    /* 如果发生了接收中断 */
    if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
    {
        //Res = USART1->DR; //寄存器方式读取数据
        Res = USART_ReceiveData(USART1); //库函数方式读取接收到的 1 个字节
        if(USART1_RX_Index >= USART_RX_MAX)
        USART1_RX_Index = 0; //防止下标越界
        USART1_RX_Buffer[USART1_RX_Index++] = Res;
        /* 清除接收中断标志位(注:也可以省略，读 DR 自动清除) */
```

```
        USART_ClearFlag(USART1, USART_FLAG_RXNE);
    }
    if(USART_GetITStatus(USART1, USART_IT_IDLE) != RESET)
    {
        USART1_RX_OverFlag = 1; //接收完成标志位置 1
        USART_ClearFlag(USART1, USART_FLAG_IDLE);
        USART_ITConfig(USART1, USART_IT_IDLE, DISABLE); //关闭接收 IDLE 中断
    }
}
```

Usart1.h

```
#ifndef __USART1_H
#define __USART1_H
#include "stm32f10x.h"
#define USART_RX_MAX 255 //定义最大接收字节数为 255
extern uint8_t USART1_RX_Buffer[USART_RX_MAX]; //定义 1.USART1 接收缓存
extern uint8_t USART1_RX_Index; //定义 2.USART1 接收数组下标
extern uint8_t USART1_RX_OverFlag; //定义 3.USART1 接收完成标志位
void USART1_Configure(void);
#endif
```

Dma1.c

```
#include "dma1.h"
#define ADC1_DR_Address ((uint32_t)0x40012400+0x4C)
__IO uint16_t ADCConvertedValue;
__IO uint32_t DMA_ADC_OK;

void DMA_ADC_Configure(void)
{
    /* 定 义 DMA 初始化结构体*/
    DMA_InitTypeDef DMA_InitStructure;
    /*定义 NVIC 初始化结构体 */
    NVIC_InitTypeDef NVIC_InitStructure;
    /* 打 开 DMAl 时 钟 */
```

```
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1,ENABLE);
    NVIC_InitStructure.NVIC_IRQChannel = DMA1_Channel1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    /*FLASH    外设基地址为 SRC_Const_Buffer */
    DMA_InitStructure.DMA_PeripheralBaseAddr =ADC1_DR_Address;
    /*RAM      基地址为 DST_Buffer */
    DMA_InitStructure.DMA_MemoryBaseAddr =(uint32_t)&ADCConvertedValue;
    /*      传输方向，外设为源地址，即 FLASH 到 RAM */
    DMA_InitStructure.DMA_DIR =DMA_DIR_PeripheralSRC;
    /*缓冲区大小，即一次传输的数据量，范围为 0~65536,此处为 32 */
    DMA_InitStructure.DMA_BufferSize =1;
    /* 外 设 和 RAM 地址自增 1 */
    DMA_InitStructure.DMA_PeripheralInc =DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryInc =DMA_MemoryInc_Disable;
    /*外设和 RAM 数据宽度，数据为 32 位(字的长度),因此宽度为字*/
    DMA_InitStructure.DMA_PeripheralDataSize =DMA_PeripheralDataSize_HalfWord;
    DMA_InitStructure.DMA_MemoryDataSize =DMA_MemoryDataSize_HalfWord;
    /*传输模式， 一次传输*/
    DMA_InitStructure.DMA_Mode =DMA_Mode_Circular;
    /* 优 先 级 高*/
    DMA_InitStructure.DMA_Priority =DMA_Priority_High;
    /*内存到内存传输*/
    DMA_InitStructure.DMA_M2M =DMA_M2M_Disable;
    /*完 成 配 置*/
    DMA_Init(DMA1_Channel1,&DMA_InitStructure);
    /*允许传输完成中断*/
    DMA_ITConfig(DMA1_Channel1,DMA_IT_TC,ENABLE);
    /* 开 启 DMA */
    DMA_Cmd(DMA1_Channel1,ENABLE);
    /*允许 DMA1 通道 1 中断 */
}

void DMA1_Channel1_IRQHandler(void)
{
    /* 检测 DMA1 Channel1 传输完成中断 */
    if(DMA_GetITStatus(DMA1_IT_TC1))
    {
        /* 传输完成标志置 1 */
        DMA_ADC_OK=1;
        /* 清除中断 */
        DMA_ClearITPendingBit(DMA1_IT_GL1);
```

```
    }
}
```

Dma1.h

```
#ifndef __DMA1_H
#define __DMA1_H
#include "stm32f10x.h"
void DMA_ADC_Configure(void);
#endif
```

Adc1.c

```
#include "adc1.h"
__IO uint16_t ADCConvertedValue1;
__IO uint16_t ADCConvertedValue2;
void ADC_Configure(void)
{
    /* 定义 GPIO 和 ADC 初始化结构体 */
    GPIO_InitTypeDef GPIO_InitStructure;
    ADC_InitTypeDef ADC_InitStructure;
    NVIC_InitTypeDef NVIC_InitStructure;
    /* 使能时钟，并配置 PB0、PB1 为模拟输入 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB | RCC_APB2Periph_ADC1, ENABLE);

    NVIC_InitStructure.NVIC_IRQChannel = ADC1_2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
```

```
    RCC_ADCCLKConfig(RCC_PCLK2_Div6);
    /* 设置 ADC 工作模式：独立、扫描、连续、不使用外部触发、数据右对齐、1 个转换 */
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;//独立
    ADC_InitStructure.ADC_ScanConvMode = ENABLE;//扫描
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;//连续
    //ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;//无外部触发
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;//数据右对齐
    /* 完成配置 */
    ADC_Init(ADC1, &ADC_InitStructure);
    /* 注入通道转换序列长度为 2 */
    ADC_InjectedSequencerLengthConfig(ADC1, 2);
    /* PB0 设置为注入通道序列 1 */
    ADC_InjectedChannelConfig(ADC1, ADC_Channel_8, 1, ADC_SampleTime_55Cycles5);
    /* PB1 设置为注入通道序列 2 */
    ADC_InjectedChannelConfig(ADC1, ADC_Channel_9, 2, ADC_SampleTime_55Cycles5);
    /* 注入通道无外部触发 */
    ADC_ExternalTrigInjectedConvConfig(ADC1, ADC_ExternalTrigInjecConv_None);
    /* 使能注入通道中断 */
    ADC_ITConfig(ADC1, ADC_IT_JEOC, ENABLE);
    /* 使能 ADC1 */
    ADC_Cmd(ADC1, ENABLE);
    /* 复位 ADC1 的校准寄存器 */
    ADC_ResetCalibration(ADC1);
    /*等待 ADC 校准寄存器复位完成*/
    while(ADC_GetResetCalibrationStatus(ADC1));
    /* 开始校准 ADC */
    ADC_StartCalibration(ADC1);
    /* 等待校准完成*/
    while(ADC_GetCalibrationStatus(ADC1));
    /* 软件方式触发 ADC 注入通道转换 */
    ADC_SoftwareStartInjectedConvCmd(ADC1, ENABLE);
}

void ADC1_2_IRQHandler(void)
{
    if(ADC_GetITStatus(ADC1,   ADC_IT_JEOC)!=RESET)
    {
        ADCConvertedValue1=ADC_GetInjectedConversionValue(ADC1,
ADC_InjectedChannel_1);
        ADCConvertedValue2=ADC_GetInjectedConversionValue(ADC1,
ADC_InjectedChannel_2);
    }
    ADC_ClearITPendingBit(ADC1, ADC_IT_JEOC);
}
```

Adc1.h

```
#ifndef __ADC1_H
#define __ADC1_H
#include "stm32f10x.h"
void ADC_Configure(void);
#endif
```

Relay.c

```
#include "relay.h"
void RELAY_Configure(void)
{
	GPIO_InitTypeDef GPIO_InitStructure;
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD|RCC_APB2Periph_GPIOC,ENABLE);
	GPIO_InitStructure.GPIO_Pin =GPIO_Pin_2;
	GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
	GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;
	GPIO_Init(GPIOD,&GPIO_InitStructure);
	GPIO_InitStructure.GPIO_Pin =GPIO_Pin_12;
	GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
	GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;
	GPIO_Init(GPIOC,&GPIO_InitStructure);
	GPIO_ResetBits(GPIOD,GPIO_Pin_2);
	GPIO_ResetBits(GPIOC,GPIO_Pin_12);
}

void RELAY1_On(void)
{
	GPIO_SetBits(GPIOC,GPIO_Pin_12);
}

void RELAY1_Off(void)
{
	GPIO_ResetBits(GPIOC,GPIO_Pin_12);
}

void RELAY2_On(void)
```

```
{
	GPIO_SetBits(GPIOD,GPIO_Pin_2);
}

void RELAY2_Off(void)
{
	GPIO_ResetBits(GPIOD,GPIO_Pin_2);
}
```

Relay.h

```
#ifndef _RELAY_H
#define _RELAY_H
#include "stm32f10x.h"
#include "delay.h"
void RELAY_Configure(void);//LED 引脚初始化
void RELAY1_On(void);
void RELAY1_Off(void);
void RELAY2_On(void);
void RELAY2_Off(void);
#endif
```

Exti10.c

```
#include "exti10.h"
#include "delay.h"
#include "relay.h"
void EXTI10_Configure(void)
{
	NVIC_InitTypeDef NVIC_InitStructure;
	EXTI_InitTypeDef EXTI_InitStructure;
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB | RCC_APB2Periph_AFIO,ENABLE);
	GPIO_EXTILineConfig(GPIO_PortSourceGPIOB,GPIO_PinSource10);
	EXTI_InitStructure.EXTI_Line =EXTI_Line10;
	EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
	EXTI_InitStructure.EXTI_Trigger =EXTI_Trigger_Rising_Falling;
	EXTI_InitStructure.EXTI_LineCmd	=ENABLE;
	EXTI_Init(&EXTI_InitStructure);
```

```
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
    NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority =2;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority    =0;
    NVIC_InitStructure.NVIC_IRQChannelCmd    =ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}

void EXTI15_10_IRQHandler(void)
{
    if(EXTI_GetITStatus(EXTI_Line10) != RESET) //确保是否产生了 EXTI Line 中断
    {
        delay_ms(10);
        if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_10)==1)//按键按下
        {
            RELAY1_On();
        }
        else if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_10)==0)
        {
            RELAY1_Off();
        }
        EXTI_ClearITPendingBit(EXTI_Line10);        //清除中断标志位
    }
}
```

Exti10.h

```
#ifndef __EXTI10_H
#define __EXTI10_H
#include "stm32f10x.h"
void EXTI10_Configure(void);
#endif
```

Di_pb10.c

```
/*----------头文件包含--------------*/
#include "di_pb10.h"
/**＊@简介：按键初始化 */
```

```
void di_Configure(void)
{
/*定义 GPIO 初始化结构体*/
	GPIO_InitTypeDef GPIO_InitStructure;
	/*打开 GPIOB 时钟*/
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
	/*配置 PB10 输入*/
	GPIO_InitStructure.GPIO_Pin=GPIO_Pin_10;
	GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IN_FLOATING;
	/*完成配置*/
	GPIO_Init(GPIOB,&GPIO_InitStructure);
}
```

Di_pb10.h

```
/*****宏定义防止重复包含*****/
#ifndef	_DI_PB10_H
#define	_DI_PB10_H
/*******头文件包含*******/
#include "stm32f10x.h"
/******函数声明*********/
void di_Configure(void);
#endif
```