Mian.c

```
#include "stm32f10x.h"
#include "delay.h"
#include "di_pb10.h"
#include "exti10.h"
#include "usart1.h"
#include "timer3.h"
#include "relay.h"
#include "stdio.h"
uint16_t a[11][2]={{0,7351},{10,7224},{20,7100},{30,6976},{40,6853},
{50,6728},{60,6600},{70,6468},{80,6330},{90,6186},{100,6033}};
uint16_t humidity=0;
uint16_t functionxy(uint16_t x1,uint16_t y1,uint16_t x2,uint16_t y2,uint16_t x);
/*----------全局变量----------*/
extern uint8_t relay_flag;
extern u32 Freq_InputCapture;
/*----------主程序---------*/
int main(void)
{
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); //设置 NVIC 中断分组 2
    TIM3_Cap_Init(0XFFFE,0); //以 1Mhz 的频率计数
    di_Configure();
    EXTI10_Configure();
    RELAY_Configure();
    USART1_Configure();//USART 初 始 化
    delay_ms(100);// 延时以显示打印字符
    while(1)
    {
        delay_ms(1000);
        printf("频率:%d Hz\r\n",Freq_InputCapture); //打印总的高点平时间
        if((Freq_InputCapture <= 7351)&&(Freq_InputCapture > 7224))
        {
            humidity=functionxy(7351,0,7224,10,Freq_InputCapture);
        }
        else if((Freq_InputCapture <= 7224)&&(Freq_InputCapture > 7100))
        {
            humidity=functionxy(7224,10,7100,20,Freq_InputCapture);
        }
        else if((Freq_InputCapture <= 7100)&&(Freq_InputCapture > 6976))
        {
            humidity=functionxy(7100,20,6976,30,Freq_InputCapture);
        }
        else if((Freq_InputCapture <= 6976)&&(Freq_InputCapture > 6853))
        {
```

```
                humidity=functionxy(6976,30,6853,40,Freq_InputCapture);
            }
            else if((Freq_InputCapture <= 6853)&&(Freq_InputCapture > 6728))
            {
                humidity=functionxy(6853,40,6728,50,Freq_InputCapture);
            }
            else if((Freq_InputCapture <= 6728)&&(Freq_InputCapture > 6600))
            {
                humidity=functionxy(6728,50,6600,60,Freq_InputCapture);
            }
            else if((Freq_InputCapture <= 6600)&&(Freq_InputCapture > 6468))
            {
                humidity=functionxy(6600,60,6468,70,Freq_InputCapture);
            }
            else if((Freq_InputCapture <= 6468)&&(Freq_InputCapture > 6330))
            {
                humidity=functionxy(6468,70,6330,80,Freq_InputCapture);
            }
            else if((Freq_InputCapture <= 6330)&&(Freq_InputCapture > 6186))
            {
                humidity=functionxy(6330,80,6186,90,Freq_InputCapture);
            }
            else if((Freq_InputCapture <= 6186)&&(Freq_InputCapture >= 6033))
            {
                humidity=functionxy(6186,90,6033,100,Freq_InputCapture);
            }
            printf("湿度:%d  %RH\r\n",humidity);
            if(relay_flag==0)
                printf("冰箱门关\r\n");
            else if(relay_flag==1)
                printf("冰箱门开\r\n");
            if(humidity >=80)
            {
                RELAY1_On();
            }
            if(humidity <=65)
            {
                RELAY1_Off();
            }
        }
}

uint16_t functionxy(uint16_t x1,uint16_t y1,uint16_t x2,uint16_t y2,uint16_t x)
{
```

```
    uint16_t y;
    y=((y2-y1)/(x2-x1))*(x-x2)+y2;
    return y;
}
```

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
```

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"
```

```
uint8_t relay_flag=0;
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_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority =1;
    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
    {
        if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_10)==1)//无磁铁靠近，冰箱门未关闭
        {
            RELAY2_On();
            relay_flag=1;
        }
        else if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_10)==0)//磁铁 S 极靠近
        {
            RELAY2_Off();
            relay_flag=0;
        }
        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
```

Timer3.c

```
#include "timer3.h"
u32 Freq_InputCapture = 0;
void TIM3_Cap_Init(u16 arr,u16 psc)
{
    TIM_ICInitTypeDef TIM3_ICInitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    NVIC_InitTypeDef NVIC_InitStructure;
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE); //①使能 TIM3 时钟
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA|RCC_APB2Periph_AFIO, ENABLE); //①
使能 GPIOA 时钟
    //初始化 GPIOA.6 ①
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6; //PA6 设置
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD; //PA6 输入
    GPIO_Init(GPIOA, &GPIO_InitStructure); //初始化 GPIOA.6
    GPIO_ResetBits(GPIOA,GPIO_Pin_6); //PA6 下拉
    //②初始化 TIM3 参数
    TIM_TimeBaseStructure.TIM_Period = arr; //设定计数器自动重装值
    TIM_TimeBaseStructure.TIM_Prescaler =psc; //预分频器
    TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1; // TDTS = Tck_tim
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up; //TIM 向上计数模式
    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure); //初始化 TIM3
    //③初始化 TIM3 输入捕获通道 1
    TIM3_ICInitStructure.TIM_Channel = TIM_Channel_1; //选择输入端 IC1 映射到 TI1 上
    TIM3_ICInitStructure.TIM_ICPolarity = TIM_ICPolarity_Rising; //上升沿捕获
    TIM3_ICInitStructure.TIM_ICSelection = TIM_ICSelection_DirectTI; //映射到 TI1 上
    TIM3_ICInitStructure.TIM_ICPrescaler = TIM_ICPSC_DIV1; //配置输入分频,不分频
    TIM3_ICInitStructure.TIM_ICFilter = 0x0000; //IC1F=0000 配置输入滤波器 不滤波
    TIM_ICInit(TIM3, &TIM3_ICInitStructure); //初始化 TIM3 输入捕获通道 1
    //⑤初始化 NVIC 中断优先级分组
    NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn; //TIM3 中断
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 2; //先占优先级 2 级
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; //从优先级 0 级
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE; //IRQ 通道被使能
    NVIC_Init(&NVIC_InitStructure); //初始化 NVIC
    TIM_ITConfig( TIM3,TIM_IT_CC1,ENABLE);//④允许更新中断捕获中断
    TIM_Cmd(TIM3,ENABLE ); //⑥使能定时器 3
}

//⑤定时器 3 中断服务程序
void TIM3_IRQHandler(void)
{
    static uint16_t capture_value1 = 0, capture_value2 = 0;
```

```
    static uint16_t flag_capture = 0;    //标志，为 0 表示当前为第一个上升沿，为 1 表示是第二个上升沿
    static uint16_t capture =0;    //捕获差值
    if(TIM_GetITStatus (TIM3,TIM_IT_CC1)!= RESET)
    {
        if(flag_capture ==0)
        {
            flag_capture = 1;
            capture_value1 = TIM_GetCapture1(TIM3);//捕获数值 1
        }
        else if(flag_capture ==1)
        {
            flag_capture = 0;
            capture_value2 = TIM_GetCapture1(TIM3);//捕获数值 2
            /*计算捕获差值 capture */
            if(capture_value2 > capture_value1)
                capture = (capture_value2 - capture_value1);
            else if(capture_value2 < capture_value1)
                capture = ((0xFFFF -capture_value1) + capture_value2);
            else
                capture = 0;
            /*计算频率  */
            Freq_InputCapture =     SystemCoreClock / capture;
        }
        TIM_ClearITPendingBit(TIM3, TIM_IT_CC1|TIM_IT_Update); //清除中断标志位
    }
}
```

Timer3.h

```
#ifndef __TIMER3_H
#define __TIMER3_H
#include "stm32f10x.h"
#define SystemCoreClock 72000000
void TIM3_Cap_Init(u16 arr,u16 psc);
#endif
```