FTHR-G0001详细按键触发中断点灯

任务：通过按键触发FTHR-G0001下降沿中断，在中断服务函数中控制LED亮灭。硬件原理图如下。



LED原理图



按键原理图

代码详细解：

#include "exti\_interrupt.h"

#include "main.h" //首先调用必要的函数包

void GPIO\_init(void)

{

 GPIO\_InitTypeDef GPIO\_InitStruct; // 定义GPIO初始化结构体变量，用于存储GPIO配置参数

 RCC\_AHBPeriphClockCmd(RCC\_AHBENR\_GPIOA, ENABLE); // 使能GPIOA外设的时钟

 GPIO\_StructInit(&GPIO\_InitStruct); // 用默认值填充GPIO初始化结构体

 GPIO\_InitStruct.GPIO\_Pin = GPIO\_Pin\_11; // 配置GPIO引脚：选择GPIOA的第11号引脚

 GPIO\_InitStruct.GPIO\_Speed = GPIO\_Speed\_50MHz;

 GPIO\_InitStruct.GPIO\_Mode = GPIO\_Mode\_Out\_PP; // 配置IO口工作模式，推挽输出可以输出高、低电平

 GPIO\_Init(GPIOA, &GPIO\_InitStruct);//从这里开始，就可以设置PA11输出高低电平了

 GPIO\_WriteBit(GPIOA, GPIO\_Pin\_11, Bit\_SET);

}

void EXTI2\_Configure()

{

 EXTI\_InitTypeDef EXTI\_InitStruct;

 GPIO\_InitTypeDef GPIO\_InitStruct;

 NVIC\_InitTypeDef NVIC\_InitStruct;

 RCC\_AHBPeriphClockCmd(RCC\_AHBENR\_GPIOA, ENABLE);

 GPIO\_StructInit(&GPIO\_InitStruct);

 GPIO\_InitStruct.GPIO\_Pin = GPIO\_Pin\_2;

 GPIO\_InitStruct.GPIO\_Mode = GPIO\_Mode\_IPU;//注意这里官方历程是GPIO\_Mode\_IPD,默认下拉电阻，导致无法检测按键传来的下降沿

 GPIO\_Init(GPIOA, &GPIO\_InitStruct);

 SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOA, EXTI\_PinSource2);

 EXTI\_StructInit(&EXTI\_InitStruct);

 EXTI\_InitStruct.EXTI\_Line = EXTI\_Line2;

 EXTI\_InitStruct.EXTI\_Mode = EXTI\_Mode\_Interrupt;

 EXTI\_InitStruct.EXTI\_Trigger = EXTI\_Trigger\_Rising;

 EXTI\_InitStruct.EXTI\_LineCmd = ENABLE;

 EXTI\_Init(&EXTI\_InitStruct);

 NVIC\_InitStruct.NVIC\_IRQChannel = EXTI2\_3\_IRQn;

 NVIC\_InitStruct.NVIC\_IRQChannelPriority = 0x01;

 NVIC\_InitStruct.NVIC\_IRQChannelCmd = ENABLE;

 NVIC\_Init(&NVIC\_InitStruct);

}

int main(void)

{

 GPIO\_init();

 EXTI2\_Configure();

 while (1)

 {

 }

}

/\*

中断服务函数

\*/

void EXTI2\_3\_IRQHandler(void)

{

 if (RESET != EXTI\_GetITStatus(EXTI\_Line2))

 {

 GPIOA->ODR ^= GPIO\_Pin\_11; //ODR是GPIO的输出数据寄存器，直接对他改写，就是操控IO口

 EXTI\_ClearITPendingBit(EXTI\_Line2);

 }

}