# 嵌入式系统实验报告



|  |  |
| --- | --- |
| 实验名称： | Lab7 STM32 EXTI 中断 |
| 姓 名： | 江姝潼 |
| 学 号： | 2019211653 |
| 学 院(系)： | 计算机学院 |
| 专 业： | 网络工程 |
| 指导教师： | 戴志涛、刘健培 |

2021年 12 月 15 日

# 实验目的

本实验主要是练习 STM32 中断的处理，一共有三种 I/O 操作技术，分别为轮询、中断和DMA操作，本次实验用中断的方式实现。

# 实验环境

* ST-Link 仿真器
* KeiluVision5 MDK集成开发软件
* PC机Window10 (64bit)

# 实验要求

1.实现按键中断方式控制 LED 的功能。

2.完成步骤 3.2，将程序源码和程序输出截图贴在作业答卷里。

# 背景知识

* 在 I/O 接口操作技术中，CPU 控制 I/O 接口的方法主要有：

一、直接程序控制：数据的输入输出是在 CPU 的执行程序的控制下完成的。

分两种：

（1）无条件传送：外设时刻准备就绪，无条件随时接受/传送 CPU 执行程

序发来的数据；

（2）程序查询方式：通过 CPU 执行程序查询外设状态，确定是否传输数据；

二、中断方式：CPU 根据接口是否发出中断请求来决定是否接受/传输数据；中断处理方式的特点是 CPU 不再被动等待，而是可以执行其他程序，一旦外设为数据交换准备就绪，可以向 CPU 提出服务请求，CPU 如果响应该请求，便暂时停止当前程序的执行，转去执行与该请求对应的服务程序，完成后，再继续执行原来被中断的程序。

三、直接存储器存取（DMA）：数据在内存和 I/O 设备中直接成块传送，不

需要经过 CPU 的控制干涉。

* STM32 的“中断”机制

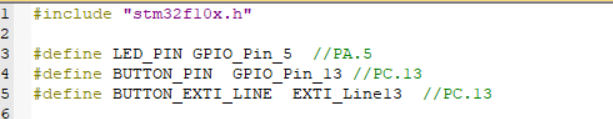
STM32 之中、Cortex-m4 核之外的“中断”，即为 interrupt/IRQ。STM32 通 过 IRQ Channel 向 NVIC 请求处理 IRQ，而 NVIC 处理包括 IRQ 在内的各

种 exception，调用其“中断处理程序”ISR。

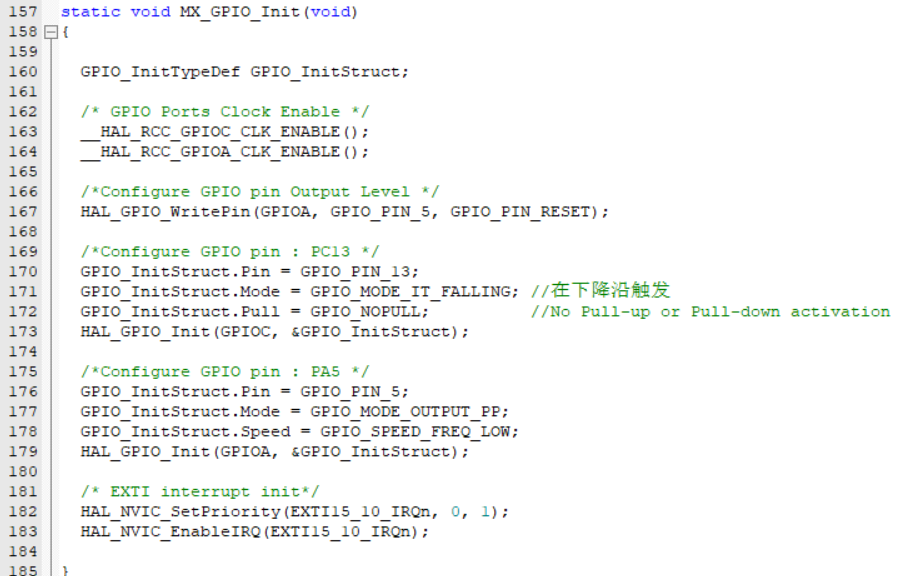
GPIO通过EXTI—与 NVIC 接口。GPIO 与 EXTI 之间的接口称为 EXTI line；而 EXTI 与 NVIC 之间则为 IRQ Channel。此外，EXTI line 上除了支持 interrupt 之外，还支持 event。event 被触发之后，并不传递给 NVIC 去处理（像 IRQ 那样），而是发送一个脉冲给电源管理模块，可以用来实现唤醒功能。

# 实验步骤

* 和lab6-3前述步骤相同，使用 STM32 标准外设库的方式，添加库代码，并定义好引脚的初始值：



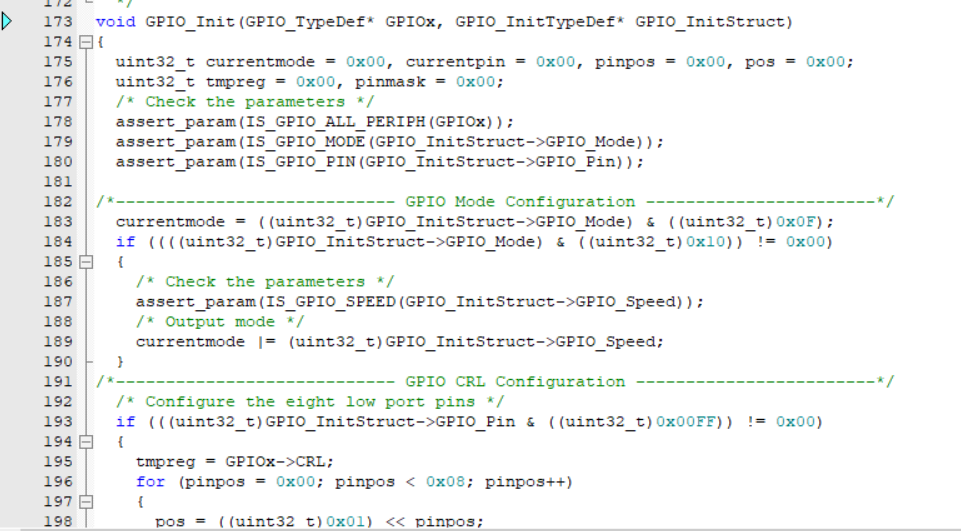
* 下面进行各引脚的初始化，可以参照lab7-1自动生成的MX\_GPIO\_Init()里的流程，发现可以分为GPIOC, GPIOA, EXTI三部分的初始化。



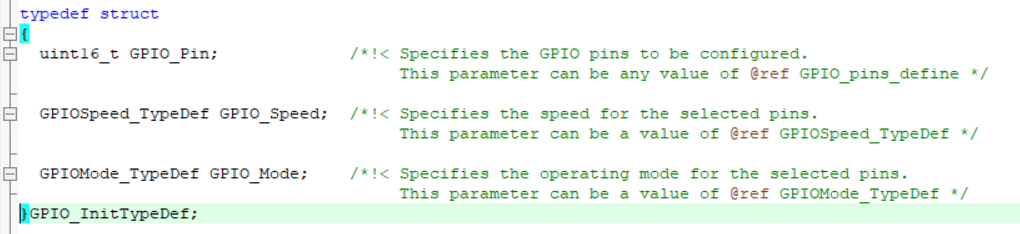
下面先进行GPIOC.13的初始化，放在Button\_Init()函数中：

首先配置GPIOC.13的引脚，先用RCC\_APB2PeriphClockCmd函数初始化GPIOC的时钟，再用GPIO\_Init函数完成初始化。

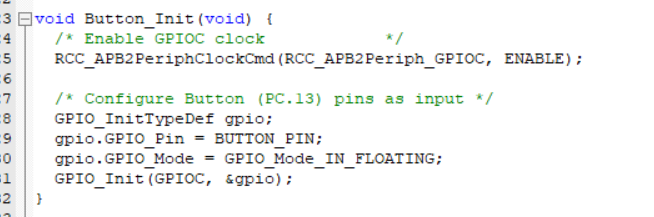
查找GPIO\_Init函数的定义，发现其功能是完成GPIO的初始化，需要传入GPIOx, GPIO\_InitStruct两个参数。



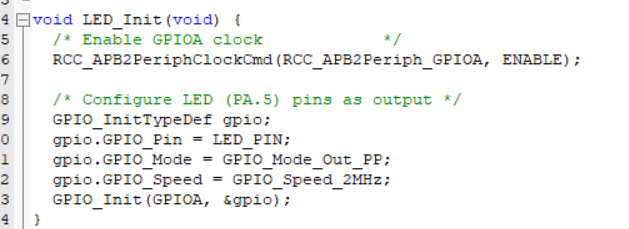
GPIOx参数即为GPIOC，GPIO\_InitStruct的类型为GPIO\_InitTypeDef，转去查看其定义：



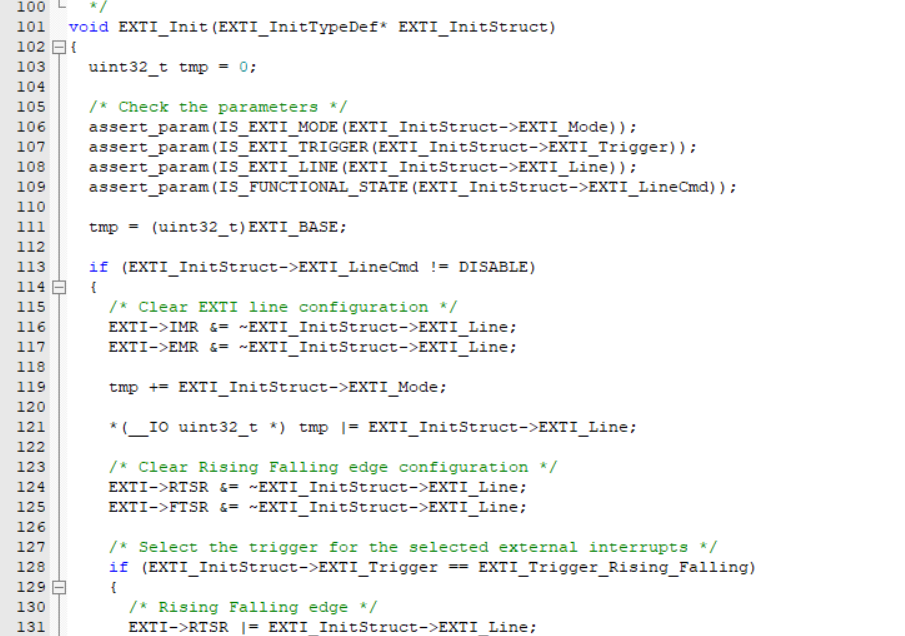
可以看到这是一个结构体，里面包括了引脚号、频率、模式这三部分信息，将其置为我们需要的模式，可以写出代码如下：



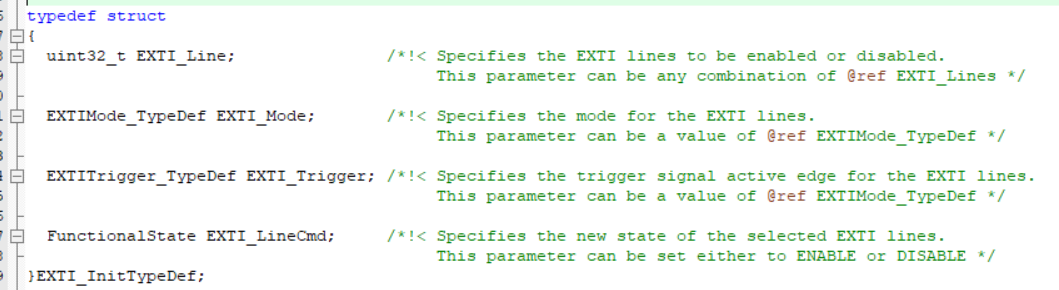
同样的方式初始化GPIOA.5端口，同样需要完成使能时钟、配置端口的两部分工作，写出的代码如下：



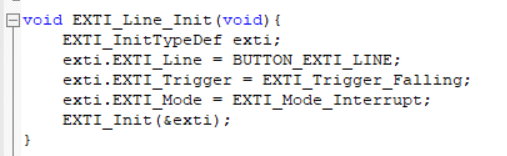
下面进行EXTI的初始化，搜索stm32f10x\_exti.c文件，可以找到一个EXTI\_Init()函数，阅读函数内容可知，该函数可以根据传入的EXTI\_InitStruct配置EXTI。



转到EXTI\_InitStruct变量的类型定义EXTI\_InitTypeDef：



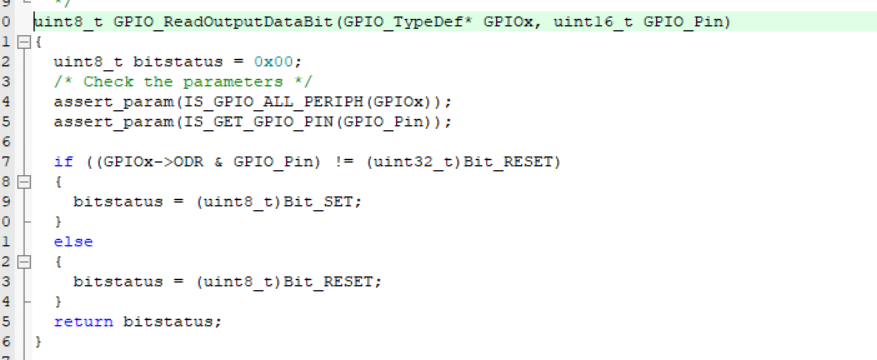
里面包括EXTI Line号、模式、触发条件和LineCmd，其中LineCmd是可选参数。本次实验要求是在按钮的下降沿触发，所以EXTI\_Trigger选择EXTI\_Trigger\_Falling，EXTI\_Line即为指定的EXTI\_Line13，模式为中断模式，再调用EXTI\_Init()函数，写出代码如下：



* 下面根据实现需求，写中断处理函数：

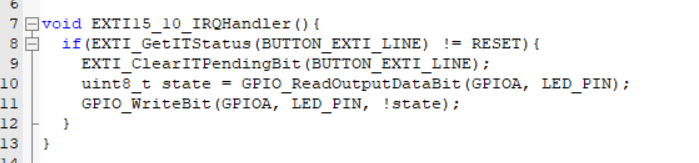
中断的原理很简单，如果检测到了中断（即处于下降沿），执行中断的相关操作，在本实验中是置LED灯的状态为上个状态的相反状态。其中一个需要注意的操作是在执行中断时，需要将该条EXTI\_Line的PendingBit清零，便以后续的操作。

下面使用GPIO\_ReadOutputDataBit函数来读取当前GPIO口的状态，查找函数定义，可以看到有GPIOx号和uint16\_t GPIO\_Pin引脚两个参数，置成我们需要的值就可以。

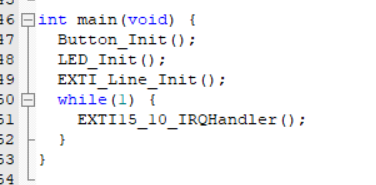


再使用GPIO\_WriteBit写回返回的值，由于需要发生LED灯的跳变，写回的状态为原来状态的相反状态。

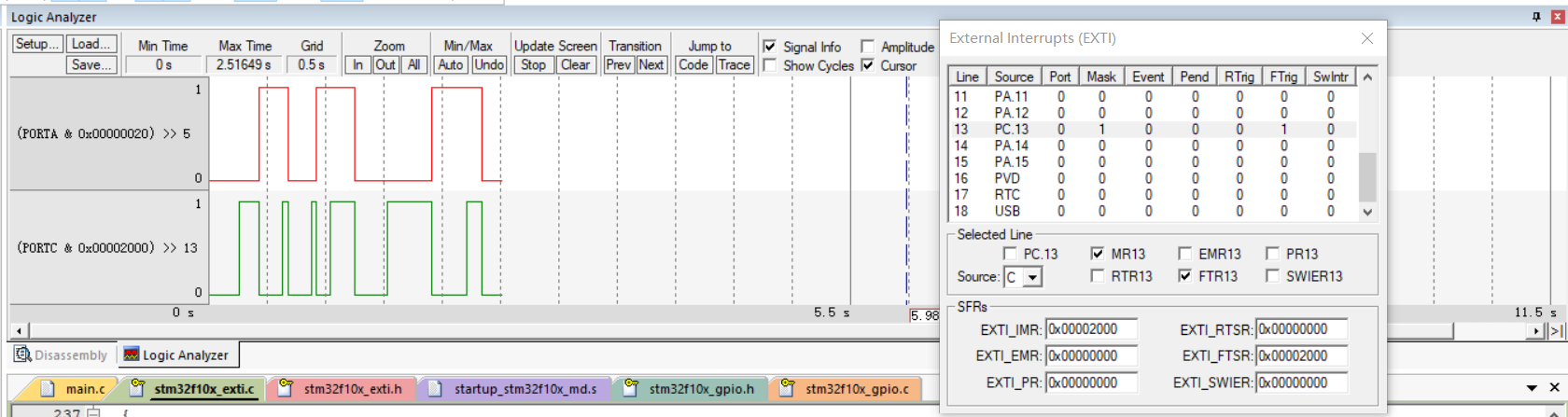
写出的中断执行函数如下：

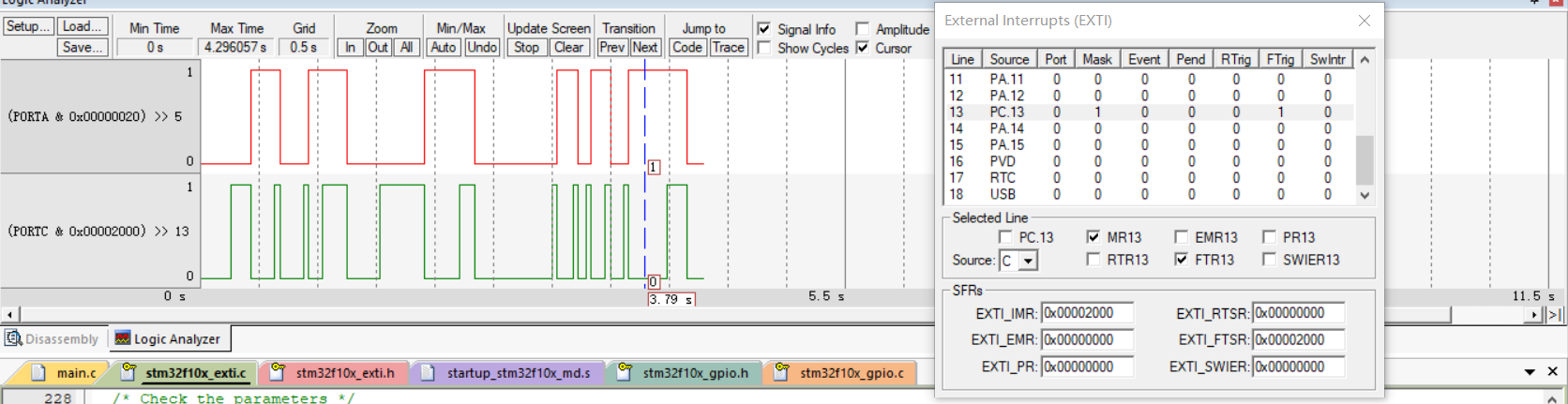


之后将该函数加在mian函数的while(1)里即可，检测是否产生了中断。



* 编译执行，效果如下：





可以看出，PORTA的输出状态在PORTC的状态为下降沿时会发生跳变。

# 实验总结

最后嵌入式课程的最后一个小实验中，我独立的完成了中断函数代码的编写，这让我感到非常的有成就感。因为在学期刚一开始的时候，我从连环境都不会配、代码完全看不懂的小白，到后面已经能够自己查询API去调用函数，同时理解它内部的硬件构造原理，这个进步的过程让我感受到了我一直在收获知识，提升自己。

同时也非常感谢老师们的付出，刘老师详细的作业指导书帮助我了解了各个实验的背景知识，并非常详细地写了各个步骤来指导我完成。但同时指导书也并没有直接告诉我答案，而是用引导的方式，一步步指导我去思考，体会STM32的运行流程。这个实验给我这种感受尤其深刻，在lab7-1中用CubeMX生成的代码简单易懂，并展示了中断的执行流程，为后面自己编写代码明确了思路。同时lab6-3的API调用方式的代码更为我自己写代码提供了一个参考，告诉我应该如何查询调用API、又如何传入参数，引导我迁移知识，完成代码的编写。

在实验的过程中也并不是一帆风顺的，过程中遇到了很多奇怪但又不知道如何解决的问题，很多次询问同学也无果，只能多次发送邮件询问老师。每次刘老师都非常及时地回复，而且一针见血地指出我所存在的问题，这让我非常感动，感谢老师们的辛勤付出。

最后我感觉这些嵌入式实验让我从自己在动手的过程中深入了解了嵌入式系统的各个原理，整体我的收获是非常大的，这些实验让我们从仿真的角度去了解了芯片的内部构造和运行的原理，让我在硬件方面多了很多知识积累。