**《按键检测》实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称**： **嵌入式系统设计** | **年级**：23级 | **上机实践成绩**： |
| **指导教师**：郭建 | **姓名**： **张建夫** |  |
| **上机实践名称**： **按键检测** | **学号：10235101477** | **上机实践日期**：2025/03/25 |
| **上机实践编号**： | **组号**： | **上机实践时间**： 14：50~16：30 |

1. **目的与要求**

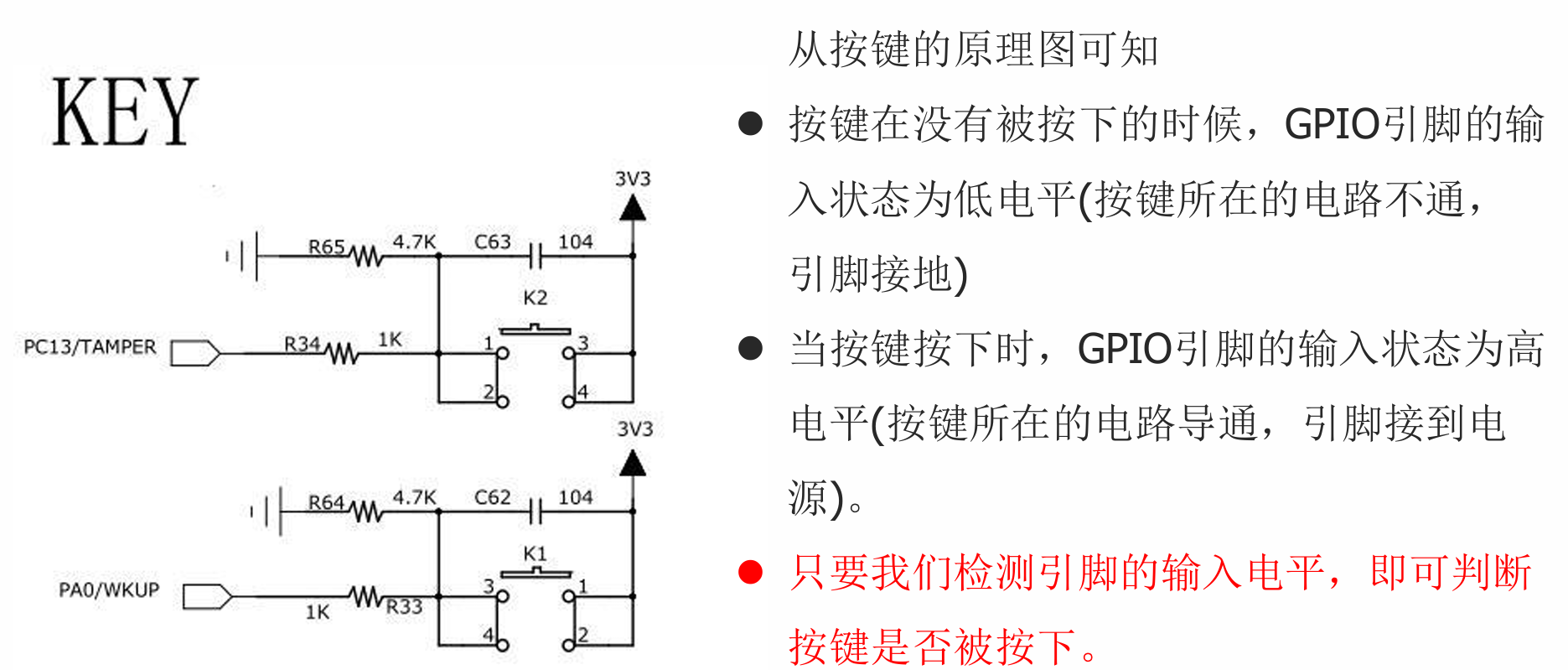
**掌握如何对按键进行检测，且控制不同按键点亮不同的LED灯**

1. **内容与实验原理**

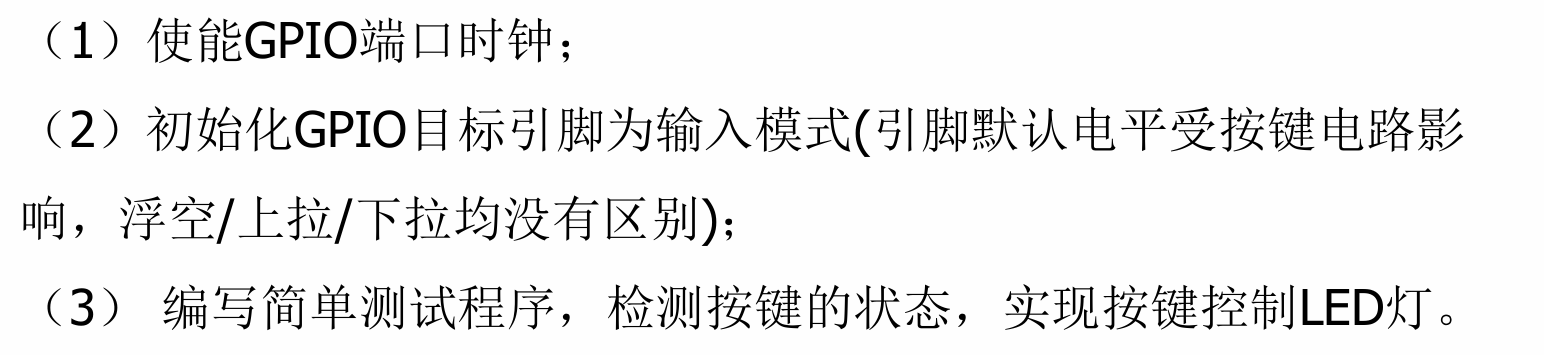
**1.实验内容：**学习如何操作按键，并能用轮询方式检测按键

**2.实验原理：**

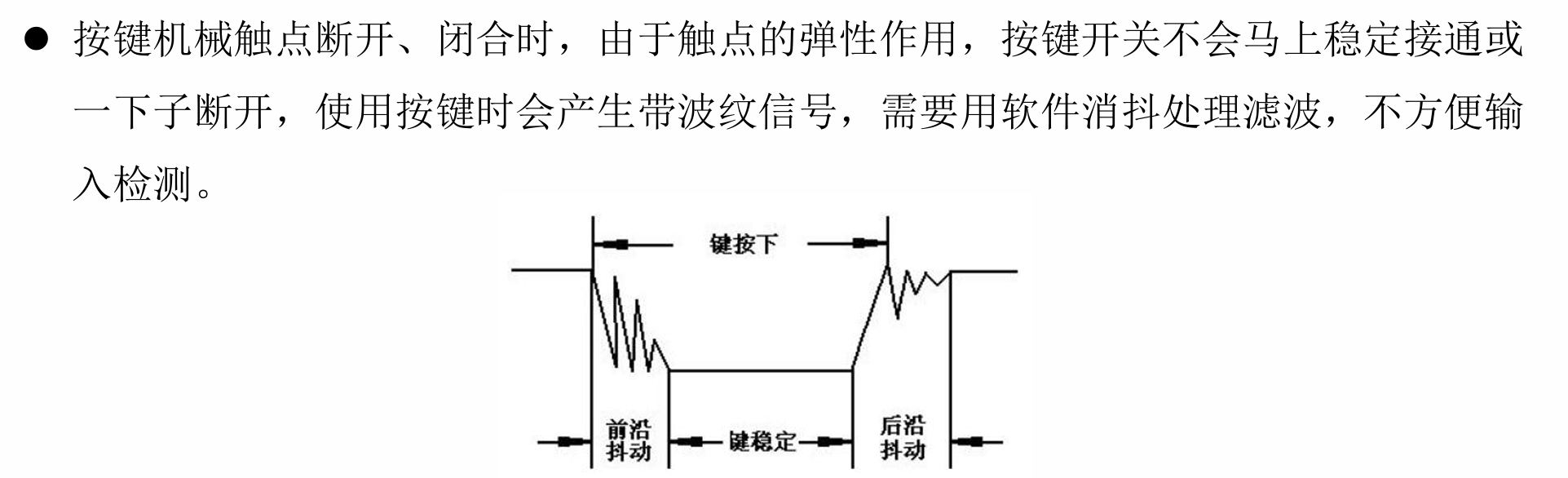
**（1）硬件设计：**

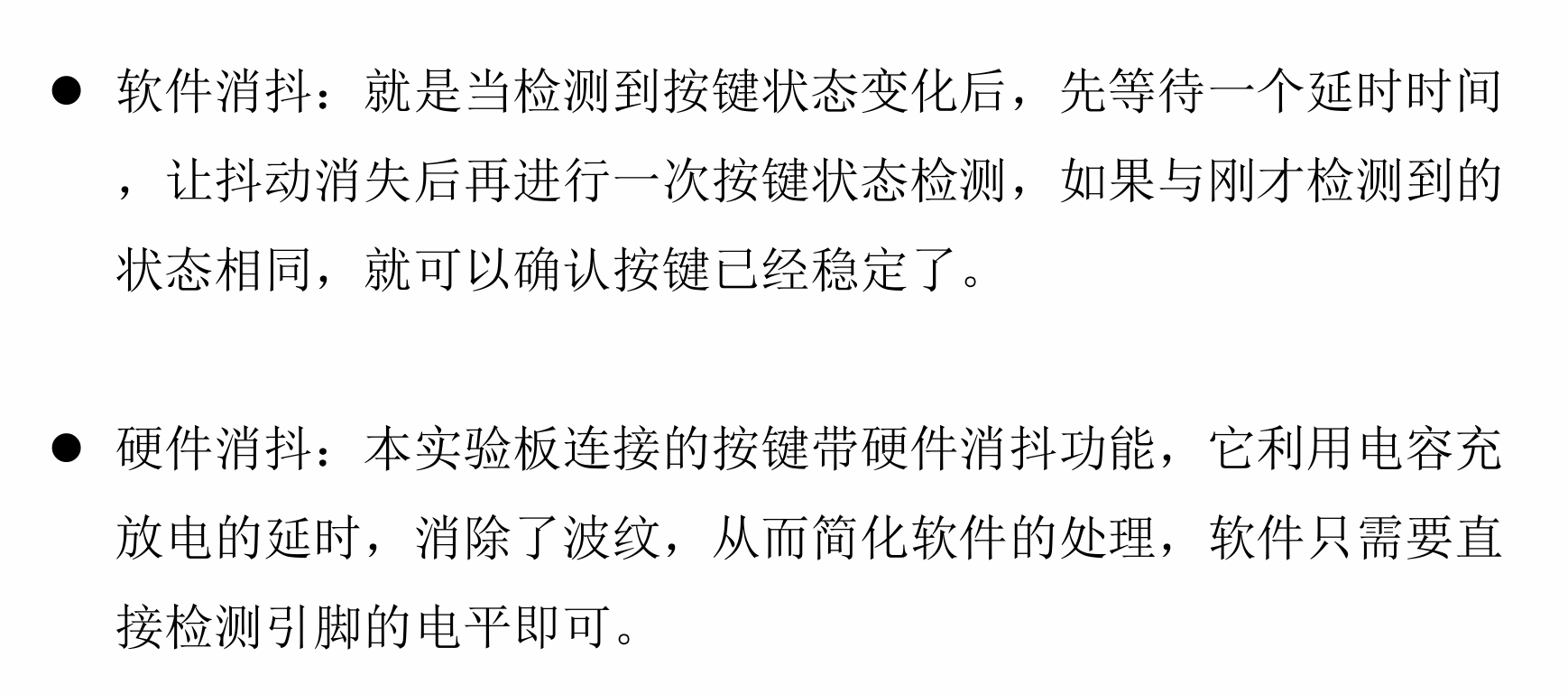
****

**（2）软件设计—编程要点：**

****

**（3）软件设计—消抖：**

****

****

1. **使用环境**

调用dxdiag工具：

Operating System: Windows 11 家庭中文版 64-bit (10.0, Build 22621) (22621.ni\_release.220506-1250)

Language: Chinese (Simplified) (Regional Setting: Chinese (Simplified))

System Manufacturer: HP

System Model: HP Pavilion Aero Laptop 13-be2xxx

BIOS: F.13 (type: UEFI)

Processor: AMD Ryzen 5 7535U with Radeon Graphics (12 CPUs), ~2.9GHz

Memory: 16384MB RAM

Available OS Memory: 15574MB RAM

Page File: 27604MB used, 5685MB available

Windows Dir: C:\WINDOWS

DirectX Version: DirectX 12

DX Setup Parameters: Not found

User DPI Setting: 144 DPI (150 percent)

System DPI Setting: 192 DPI (200 percent)

DWM DPI Scaling: UnKnown

Miracast: Available, with HDCP

Microsoft Graphics Hybrid: Not Supported

1. **主要实验内容和结果展示**

**前言：个人认为这次实验过于简单（给出的模板改两个字母就可以完成第二个任务了），于是我想是否能把这几次实验结合起来做些其他事情，于是便萌生了做一个智能红绿灯的想法，即该红绿灯能够通过按键控制，按下按键后，红绿灯会停在当前状态，即如果当前展现的是红灯，按下按键，红灯将暂时“卡住”，直到下次按下按键，红绿灯才重新交替闪烁。我在第3小节记录了这个过程。**

1. **示例实验：**

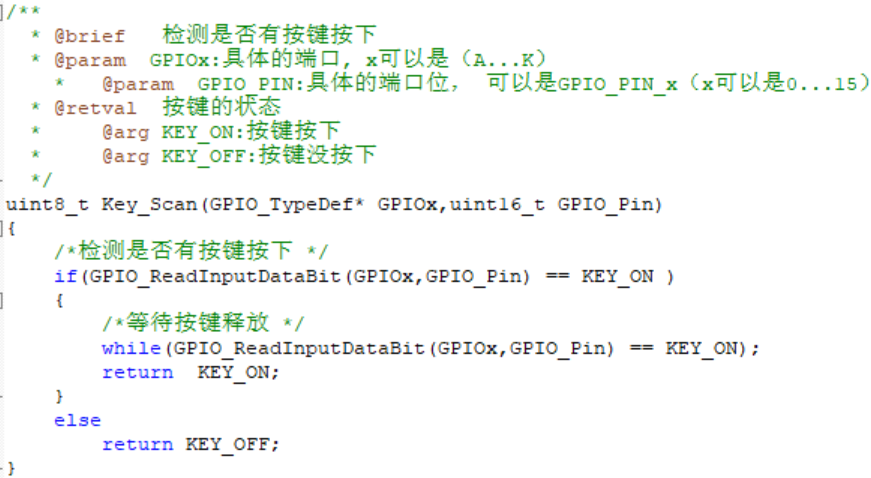
将示例工程打开并烧录，便可以实现key1控制红灯的亮灭，下面简单解释key1是如何控制红灯的亮灭。

首先要初始化key1按键：



按键的初始化涉及4个方面，第一个是初始化按键gpio口的时钟，这里初始化了两个引脚的时钟（key1和key2），使用了“|”表示。第二个初始化按键的引脚为key1的引脚，这样就能控制对应的引脚。第三个初始化引脚的输入/输出模式，即该引脚是负责输入还是输出。第四个初始化引脚的上下拉情况。最后将初始化结构体传入库函数实现按键的初始化。

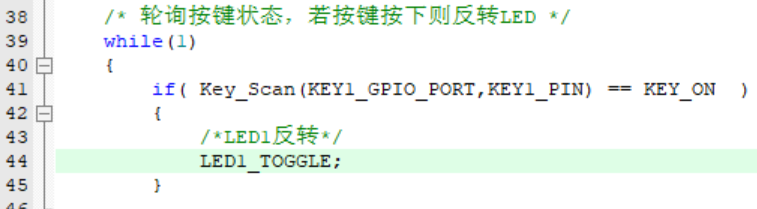
初始化按键后，我们还需要通过检测对应引脚状态来判断按键状态以此实现按键控制led灯的亮灭：



这里的key\_Scan函数会检测是否有按键按下，里面调用了关键函数GPIO\_ReadInputDataBit（一个库函数，读取当前对应引脚和端口的状态）来实际判断状态，同时这里用了忙等的方式使用户松开按键后才会返回结果。

需要注意的是，此处没有消抖处理是因为开发板的硬件已经做了消抖处理，无须额外的等待。

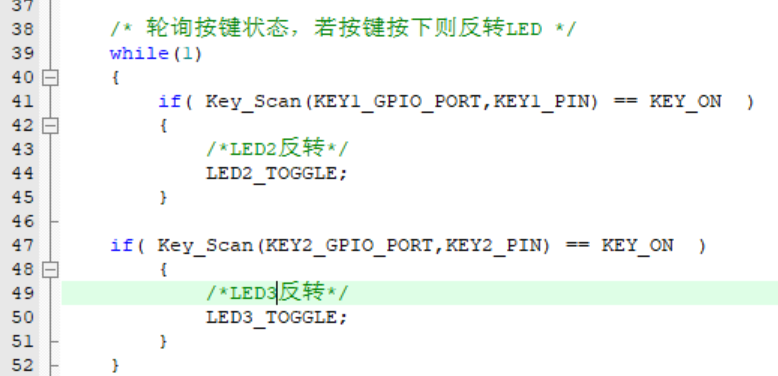
最后，使用上面编写的函数来实现按钮对红灯的控制：



其中LED1\_TOGGLE是提前定义好的宏，能够点亮红灯，这里就是不断轮询，查看用户是否按下按键，按下按键则反转红灯状态。

1. **“**key1按键控制绿灯的亮灭，key2按键控制蓝灯的亮灭”**：**

前面key1和key2的初始化代码无需变动，唯一要更改的部分就是main函数的轮询代码，将main函数的轮询改为两个灯并发轮询即可：

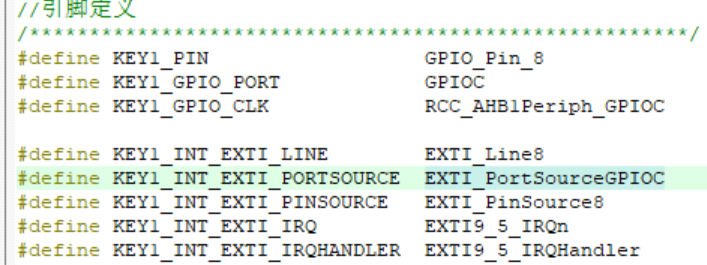


1. 探索“红绿灯”：

做这个项目的时候一波三折，刚开始想用已有的知识来做，但是一直没有想出来合适的解法，后面在网上查资料说要用中断，便转向使用中断实现，但是由于没有使用过中断，中间实现的过程一直报错，后面不了了之 ，直到一天后偶然又想了想，发现可以在不用中断的情况下实现，才成功完成。

**使用中断：**这个想法相当简单，主程序设定一个paused变量用来表示红绿灯是否停下，在按下按键后，产生外部中断，在中断处理函数中将paused变量取反（这样每次按下按键，状态就会切换），主函数中，在每个灯熄灭前，检查paused变量，如果paused变量置1了，就通过跳进空循环，不断检查下一次按键来实现冻结。

首先要更改key1的设置，使其能够相应中断：



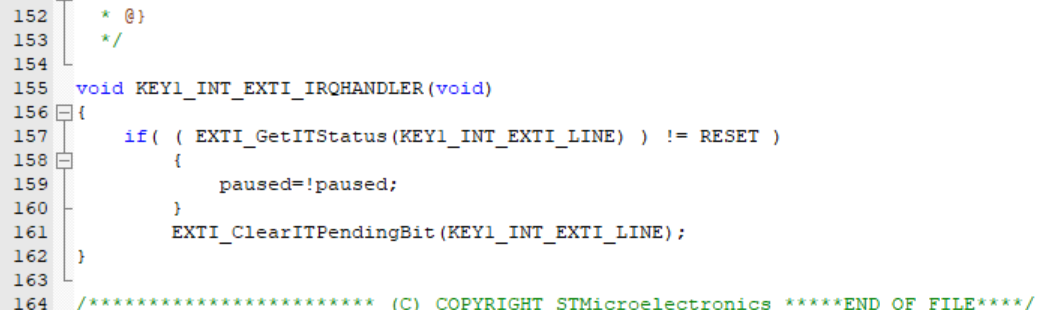
该处的代码是根据网上的代码照葫芦画瓢改的，设定了中断处理函数为中断线EXTI5~9共享的中断处理函数EXTI9\_5\_IRQHandler以及按键对应的中断线等初始化。

中断初始化函数如下：



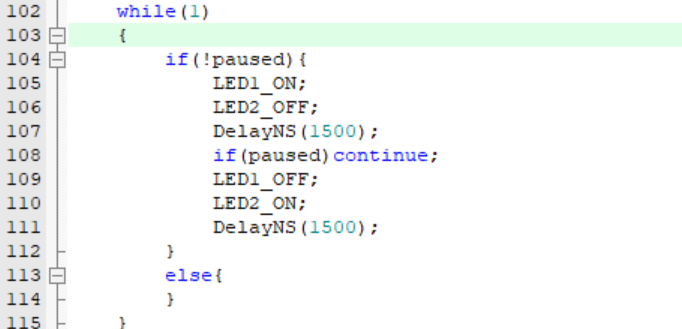
两个函数均为从网上拷贝后稍微修改得到，stm32f4系列的微处理器需要初始化中断向量表和中断引脚。

中断处理函数如下：



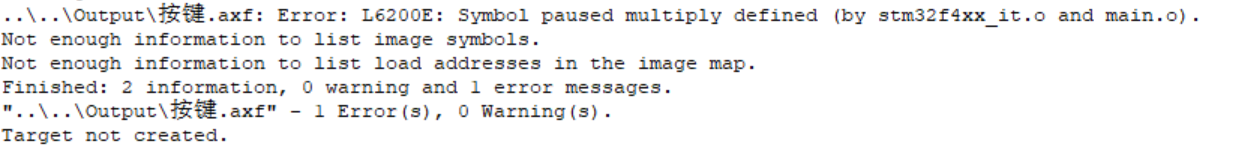
使用了两个库函数来实现主要逻辑，这里修改的是stm32f4xx\_it.c文件（之前是直接在main.c里加的，查了资料发现中断处理函数应该加在stm32f4xx\_it.c，并且要加上相应的头文件）

主函数如下：



这里的逻辑就是在灯的间隙判断paused变量的值，一但paused被置位，就会使整个循环空转，直到下次按下按键，才会恢复。（所有代码见附件）

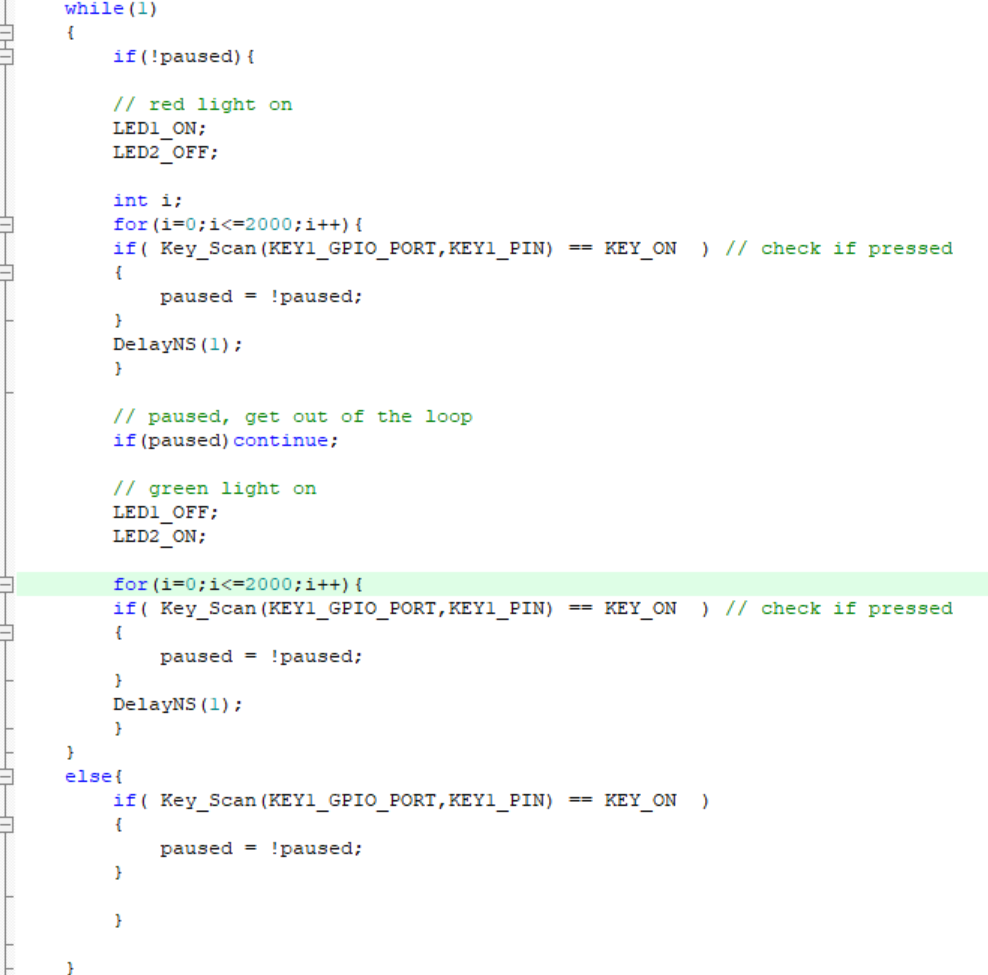
写好代码后编译，得到如下报错：



该报错的原因根据网上的资料是有源文件未被包含进项目，但是我查了几次，所有新加的源文件和头文件都应该纳入了本项目，但是报错仍旧，遂放弃。

**不用中断（在本实验的基础上加东西）：**此处的思路是，在每个灯亮的过程中不断检查按键的输入，如果按下则反转paused的值，并在灯灭前检查paused，paused为1则跳入空循环，空循环中不断检查按键是否按下，以随时返回灯亮的状态。

以下是主循环：



与中断的区别就是，中断的代码要简洁很多，也更为灵敏，而我这个代码实际上就是把中断处理函数里面的内容写了三次（红灯亮时，绿灯亮时，循环空转时），其他的代码和原实验里面的内容相同，此处不再赘述（所有代码和演示见附件）

**五、实验总结**

本次实验我收获颇多，通过加入按键这一个功能，使我使用开发板的能力大大增强，尤其是对多文件编写项目的能力和端口映射的重要性。此外，通过对自己想法的探索，发现自己对嵌入式编程的认识仍然不够深入，还需要不断地学习，同时也很好的激发了我对嵌入式开发探索的兴趣，后面有机会找一些有意思的嵌入式项目做一些实际的开发。