# 嵌入式系统实验报告



|  |  |
| --- | --- |
| 实验名称： | GPIO—按键与 LED |
| 姓 名： | 王小龙 |
| 学 号： | 2020211502 |
| 学 院(系)： | 计算机学院 |
| 专 业： | 网络工程 |
| 指导教师： | 刘健培、戴志涛 |

2022年 11月 30 日

# 实验目的

⚫ 通过 FSM4 实验板了解实验的软硬件环境，熟悉 MDK 开发环境的使用。

⚫ 学习查阅文档和数据手册，获取需要的信息。

⚫ 学会使用 C 语言控制 IO 寄存器完成设备控制功能。

⚫ 学会 STM32 GPIO 的基本操作方式（初始化、读、写）。

⚫ 理解开关消抖的原理与机制。

⚫ 理解同时操作多个 IO 的方式。

⚫ 掌握基本的驱动软件编写与调试方式。

# 实验环境

⚫ FS-STM32F407开发平台

⚫ ST-Link 仿真器

⚫ RealView MDK5.23集成开发软件

⚫ PC机Window10 (64bit)

⚫ 串口调试工具

# 实验要求

⚫ 使用实验板上的 4 个按键（K3-K6）控制 4 个 LED 灯（D6-D9）的状态

变化。

⚫ 无按键操作时，D6-D9 以跑马灯（1 个灯光在 4 个 LED 上依次滚动亮起，

收尾相接）的方式顺时针闪烁。滚动周期 1 秒。

⚫ K3——短按，暂停键。第一次短按，暂停跑马灯，第二次短按，继续跑

马灯。

⚫ K4——双击，前进键。双击 K4，跑马灯顺时针闪烁。

⚫ K5——双击，后退键。双击 K5，跑马灯逆时针闪烁。

⚫ K6——长按，呼吸灯。长按（超过 2 秒），4 个 LED 灯模拟呼吸灯亮灭

（即依次逐渐亮-灭-亮），松开后，呼吸灯停止。

# 实验原理

1.STM32 GPIO 的配置

LED 灯的亮灭、蜂鸣器的鸣响、按键电平的读入都需要使用 STM32 芯片的

I/O 引脚。

STM32 芯片上，I/O 引脚可以被软件设置成各种不同的功能，如输入或输出，

所以被称为 GPIO (General-purpose I/O)。而 GPIO 引脚又被分为 GPIOA、

GPIOB…GPIOG 不同的组，每组端口分为 0~15，共 16 个不同的引脚。

程序控制 GPIO，包括初始化和主逻辑 2 部分。

STM32 GPIO 的初始化需要 2 步：

1. 开启 GPIO 时钟

2. 设置好 GPIO 的配置寄存器

如果是复用 GPIO 管脚的外设，则一般需要 4 步：

1. 开启复用管脚的 GPIO 时钟

2. 设置好复用管脚的 GPIO 配置寄存器

3. 开启外设时钟

4. 设置好外设的配置寄存器

主逻辑部分主要是完成程序功能，如本实验的摩斯码编码输出。

一般主逻辑可分为 2 部分：

1. 与底层外设无关的逻辑部分。如摩斯码的编解码。

2. 需要通过外设与外部交互的驱动部分。如摩斯码的滴答到 LED/蜂鸣器的映射。

2. 按键扫描与消抖

由于机械触点的弹性作用，按键开关在闭合时不会马上稳定的接通，而是有

一段时间的抖动，在断开时也不会立即断开。

抖动时间由按键的机械特性所决定，一般为 5ms~20ms。如果不作处理这个

抖动会给系统带来一些不稳定的因素，甚至是错误的结果，所以在做按键检测时

都要加一个消抖的过程。

软件消抖有如下多种方案

（1）CPU 延时重采样。

延时重采样的意思是，当第一次检测到键值变化（如 1->0）时，再延时一段时间（如 20ms），再次采样，确认是否仍是'0'；若是'0'则认为此时键值为'0'，否则，重新执行检测过程。延时可以采用轮询，也可以采用定时器。

（2）持续采样

持续采样的原理是，当检测到按键处于某电平（如'0'）时，在之后的 N 个时钟周期内连续检测此按键的电平，如果一直不变，则读出此按键的电平值（如'0'）。

（3）使用外部中断

按键按动时会产生高低电平的变化，可以将按键管脚设置为边缘触发中断（上边缘、下边缘 or 双边）。但按键抖动可能触发多次中断，所以处理中断时还需要以上的延迟或者重采样思想来消抖。

（4）使用 ADC 转换器

当按键接通时，通过 ADC 转换后对按键的电压值比较判断，这是一种滤波法。

（5）软件数字滤波法

滤波算法有很多种，比如中位值滤波法、算术平均滤波法、加权递推平均滤波法、消抖滤波法等，通过对历史数据与新数据的特性进行判别，得到当前的稳定值。不同滤波算法与不同的特性

下面介绍递推平均滤波法：

每隔固定时间（例如 1ms 或它的倍数），读取按钮的数字状态并将其保存到缓冲区数组中。例如设置缓冲区长度为 4 个字节，在任何时间查看该 4 个字节长的缓冲区，它将反映该按钮的最后 4 个状态的历史记录。每次采样都添加一个新读数，并删除最旧的读数，然后通过将数组中的所有读数相加并除以 4 来

计算平均值。

因为按键的输出是 0-1 数字值，也可以改为对最后 4 个状态求和，如果总

和等于 4，我们可以将按钮视为 HIGH。否则，它是一个 LOW。

这实际上是通用 FIR（有限脉冲响应）数字滤波器的“特例”，所有系数都

相同且等于（0.25 或通常为 1/n，其中 n 是缓冲区的长度）。这会在系统的输出上产生 LPF（低通滤波器）效果，这正是我们想要的，只是为了摆脱高频弹跳影响。

3. 呼吸灯与 PWM 控制原理

呼吸灯，就是指灯光设备的亮度随着时间由暗到亮逐渐增强，再由亮到暗逐

渐衰减，很有节奏感地一起一伏，就像是在呼吸一样，呼吸的特性是一种类似图 指数曲线 中的指数曲线过程，吸气是指数上升过程，呼气是指数下降过程，成年人吸气呼气整个过程持续约 3 秒。

要控制 LED 灯达到呼吸灯的效果，实际上就是要控制 LED 灯的亮度拟合呼吸特性曲线。若控制脉冲的占空比在 3 秒的时间周期内按呼吸特性曲线变化，那么就可以实现呼吸灯的效果了。

这种使用脉冲占空比拟合不同波形的方式称为 PWM(脉冲宽度调制)控制技

术——通过对一系列脉冲的宽度进行调制，来等效地获得所需要波形（含形状和

幅值）。 PWM 控制的基本原理为：冲量相等而开头不同的窄脉冲加在具有惯性的环节上时，其效果基本相同。其中冲量指窄脉冲的面积；效果相同指环节输出响应波形基本相同。

在本章的实验中，实现使用计算法得到的呼吸曲线 PWM 波和 SPWM 波。

在本实验中，PWM 波的生成使用软件延迟模拟，即：根据占空比，先输出高电

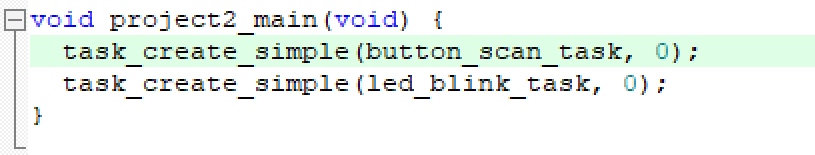
平，然后程序延迟一段时间（task\_sleep 函数），再输出低电平，延迟一段时间。

# 实验步骤

1.创建工程并导入参考代码。

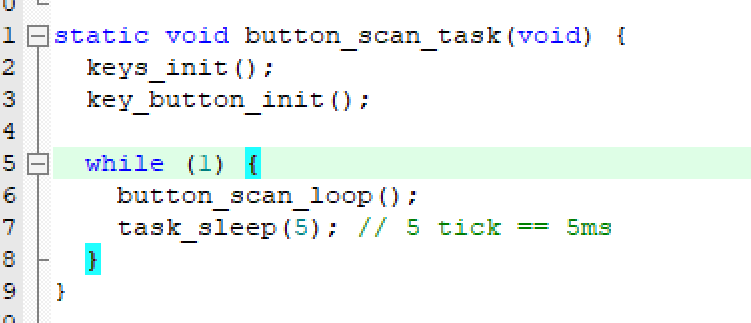
设置按键 K3-K6的 GPIO 口。

void project2\_main(void)里面执行task\_create\_simple(button\_scan\_task, 0)

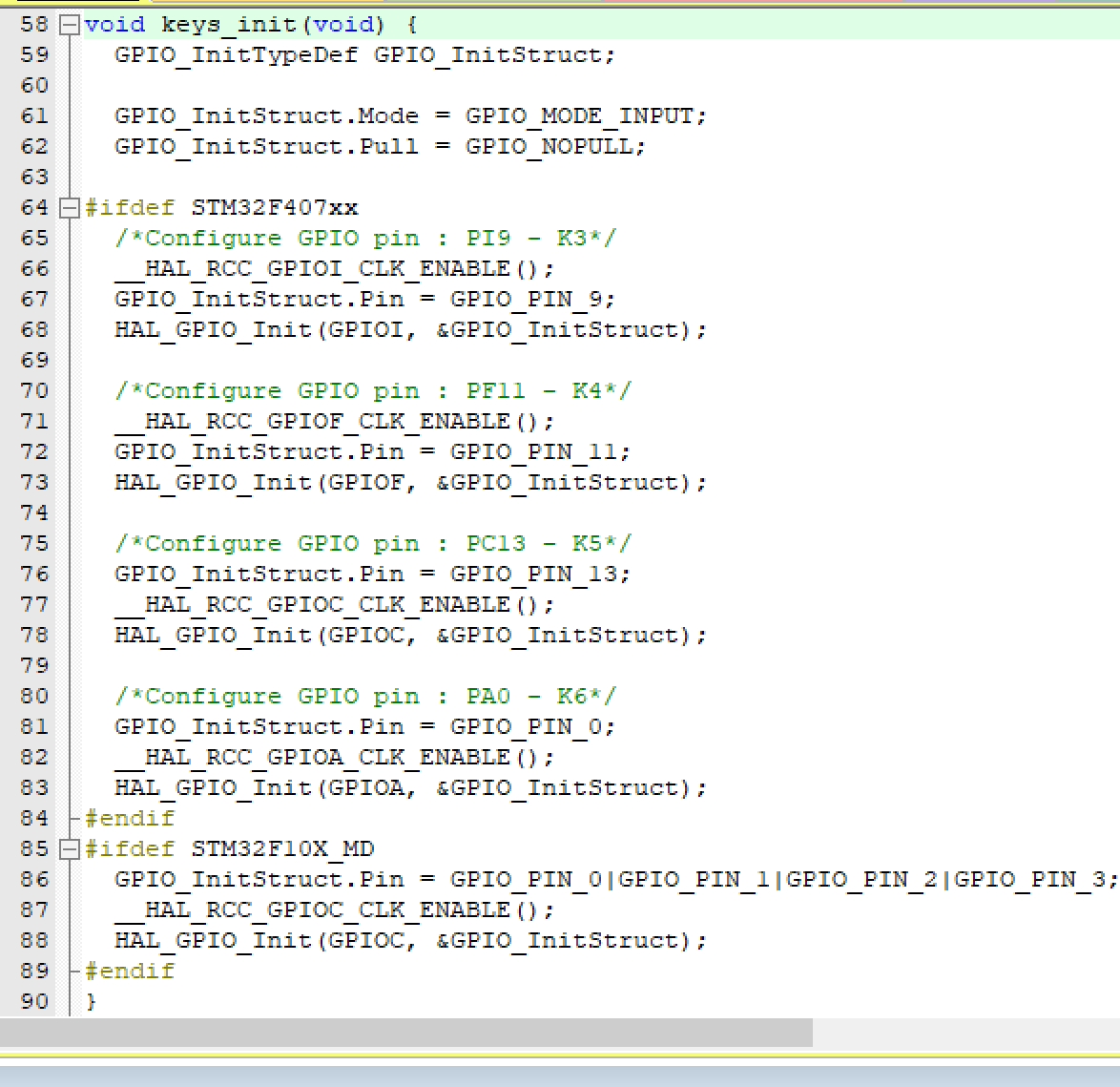


查看static void button\_scan\_task(void)函数，可以看到该函数调用了keys\_init();

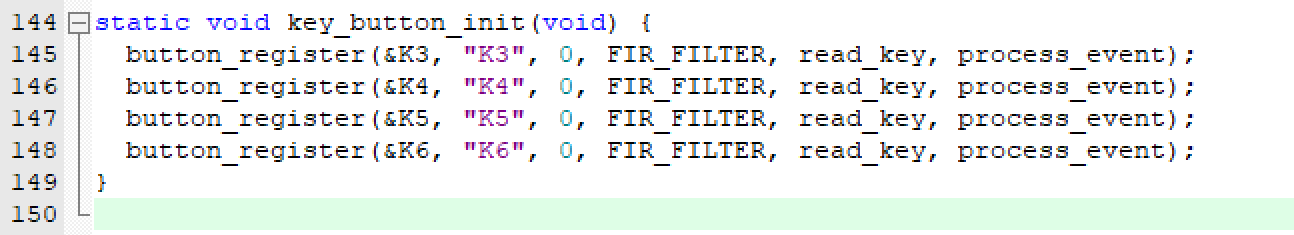
key\_button\_init();两个函数：



keys\_init()函数绑定了GPIO引脚，初始化了按键K3、K4、K5、K6：

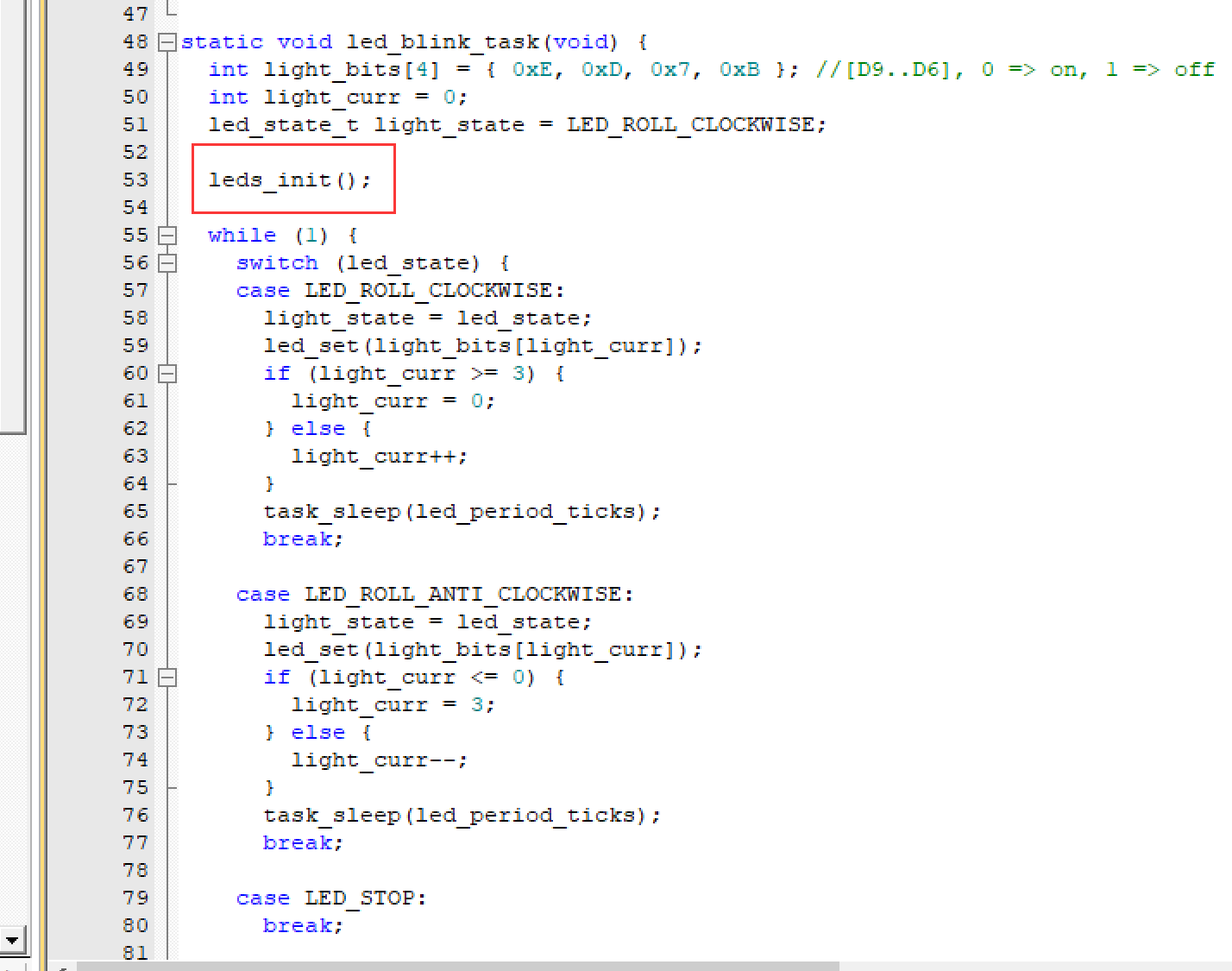


key\_button\_init()函数对读入的按键状态确定处理方式：



2.设置LED6-9的 GPIO 口

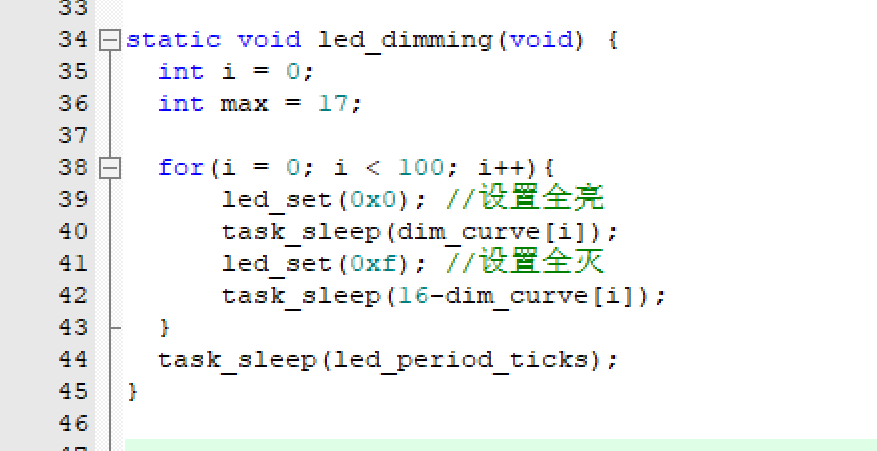
查看main函数里创建的第二个任务static void led\_blink\_task(void)函数，可以发现里面包括包括初始化、状态处理两部分：



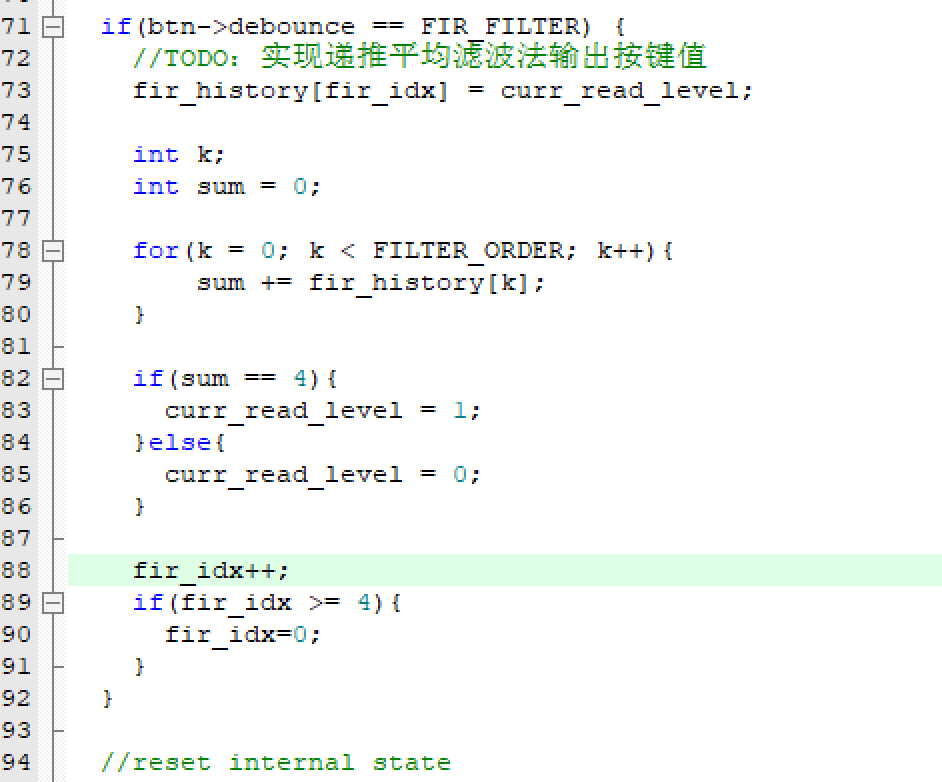
leds\_init()函数绑定了LED对应的引脚；

执行完初始化函数后，static void led\_blink\_task(void)函数开始进入循环，根据状态执行对应操作，当状态为LED\_DIMMING时，调用led\_dimming函数

编写调用led\_dimming函数如下：



3.在button\_scan.c/scan\_button函数里实现使用递推平均滤波法消抖的代码如下：



4.验证程序

在 sscom 中输入 prj 2 命令，将调用 project2\_main 函数，启动按键采样与 LED

控制任务。

配置 Logic Analyzer，用于观察 LED 灯的 IO 输出（0 – 亮，1 - 灭）。

模拟 K3 操作。

取消勾选 PORTC0，模拟按键 K3 按下操作，一小段时间后勾选 PORTC0，

模拟按键松开操作，可以识别一次短按。

观察 Logic Analyzer，分析 D6-D9（PORTC.6- PORTC.9）是否处于跑马灯（每

次亮一个灯，亮灯依次循环）或者暂停模式。

模拟长按 K6。验证输出。

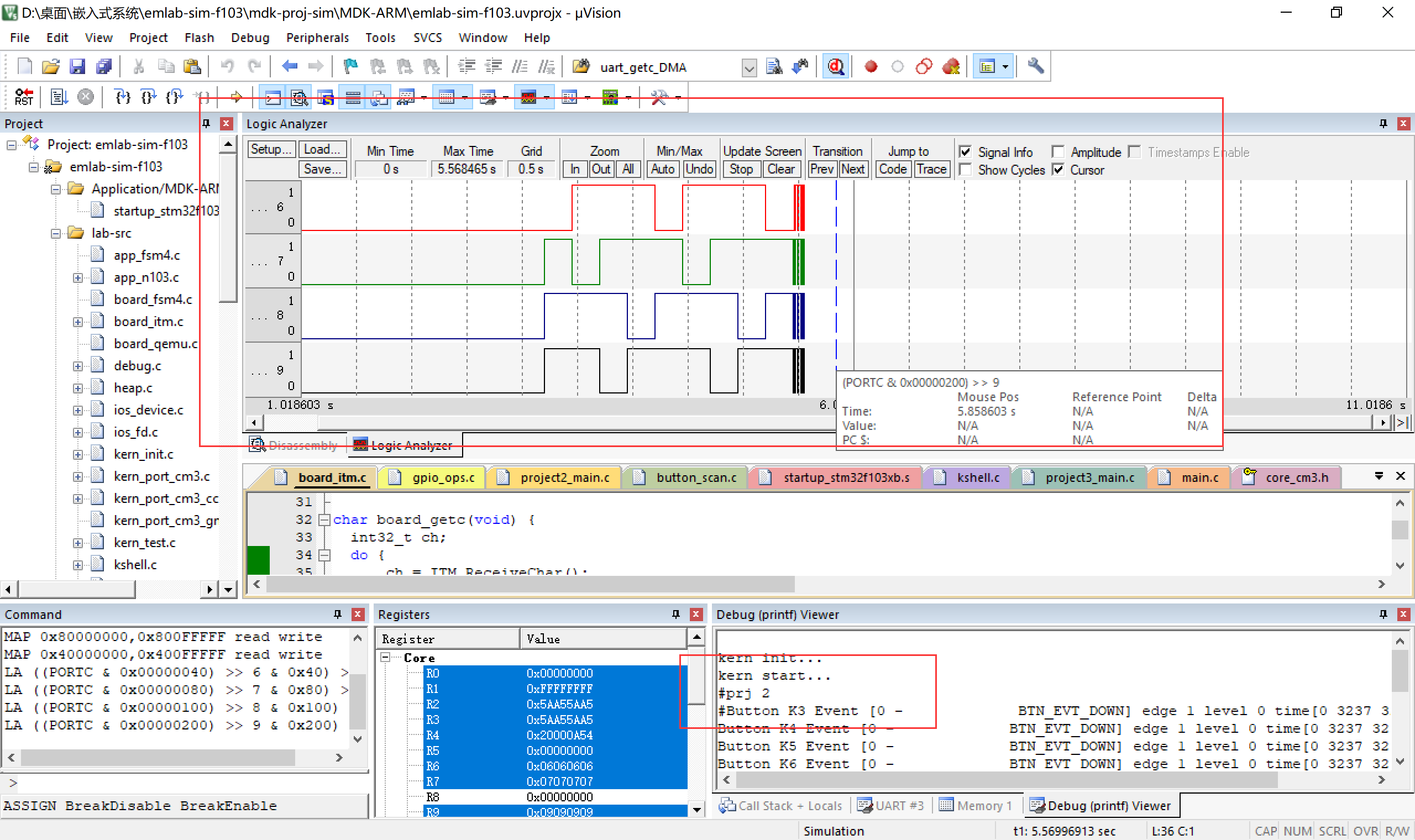
取消勾选 PORTC3，模拟按键 K6 按下操作，一段时间后识别到长按（时间

比实验板略久）。

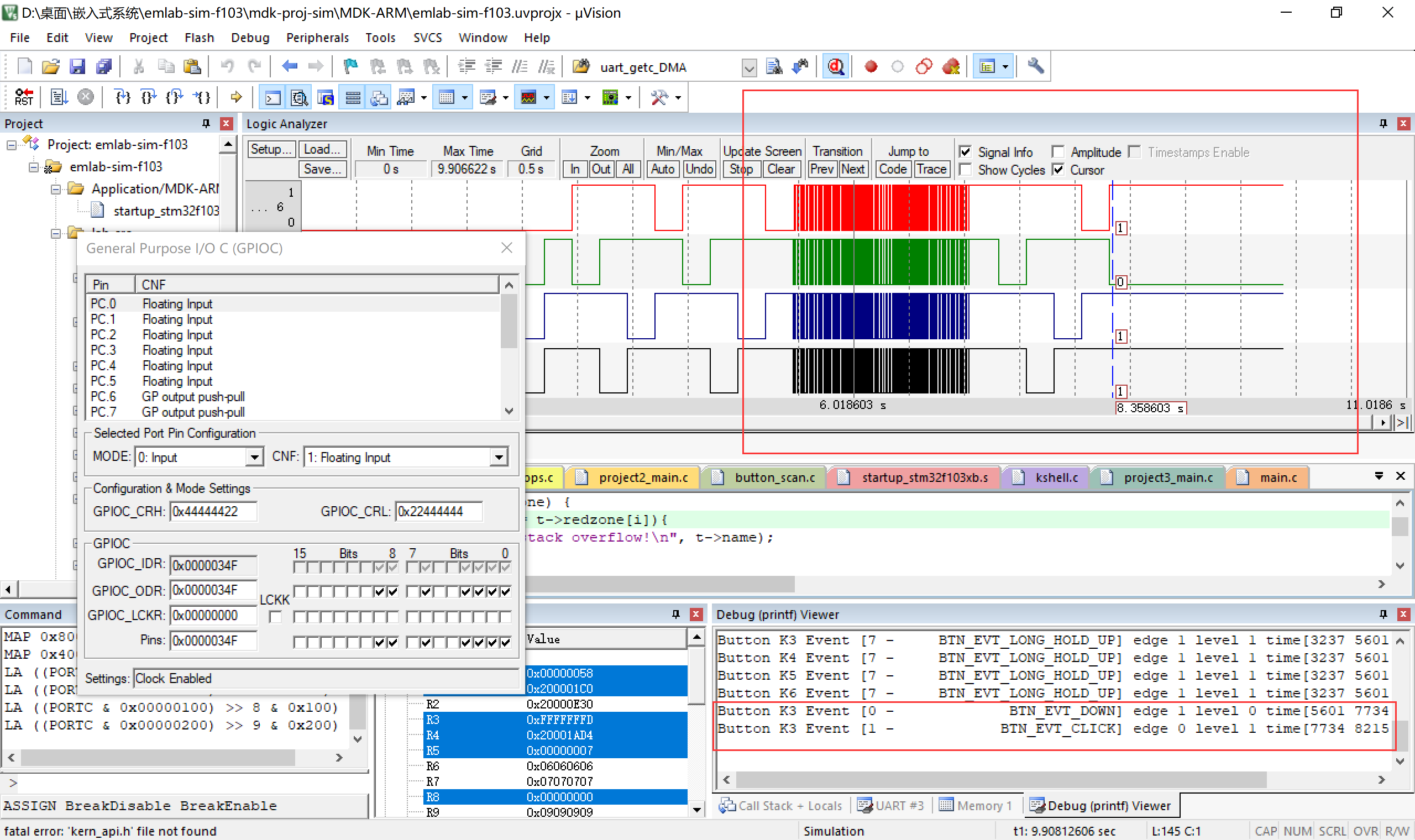
观察 Logic Analyzer，分析 D6-D9（PORTC.6- PORTC.9）是否处于呼吸灯模

模式（观察波形占空比的变化）。

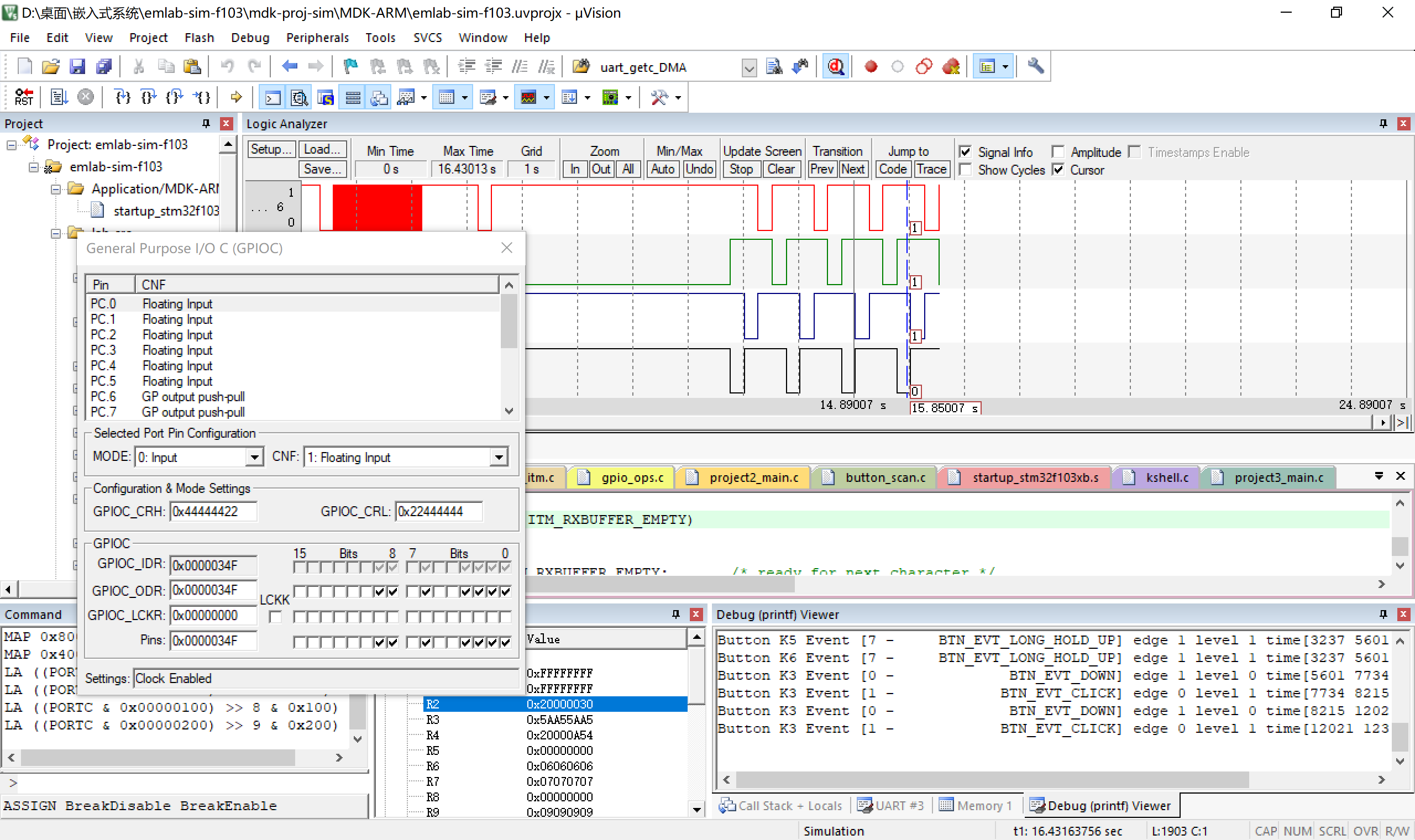
输入prj 2后，可以看到波形输出：



首先验证K3——短按，暂停键。

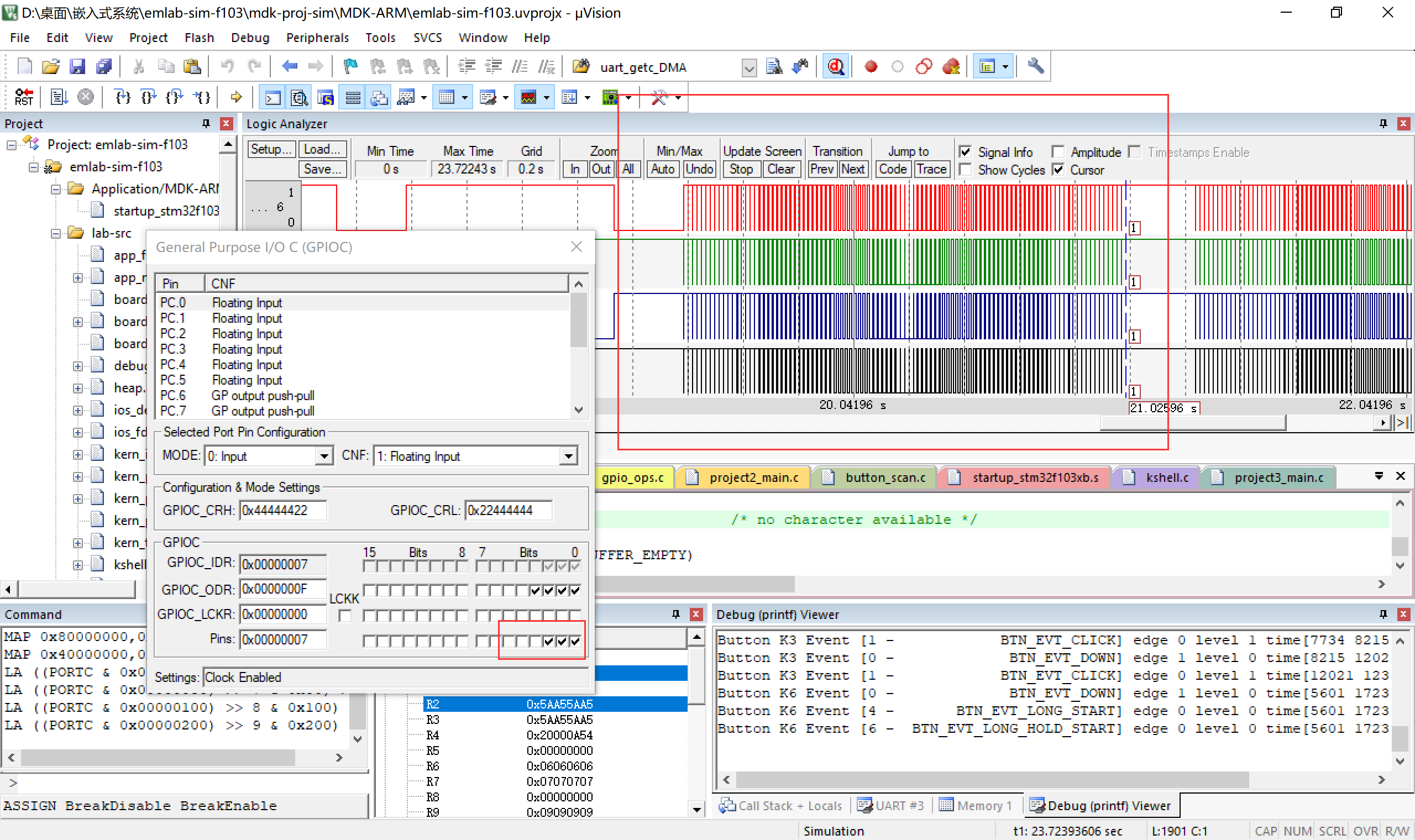


由上图可知，模拟短按一次K3后，波形被暂停输出，再短按一次K3后，波形又恢复，如下图，所以K3成功



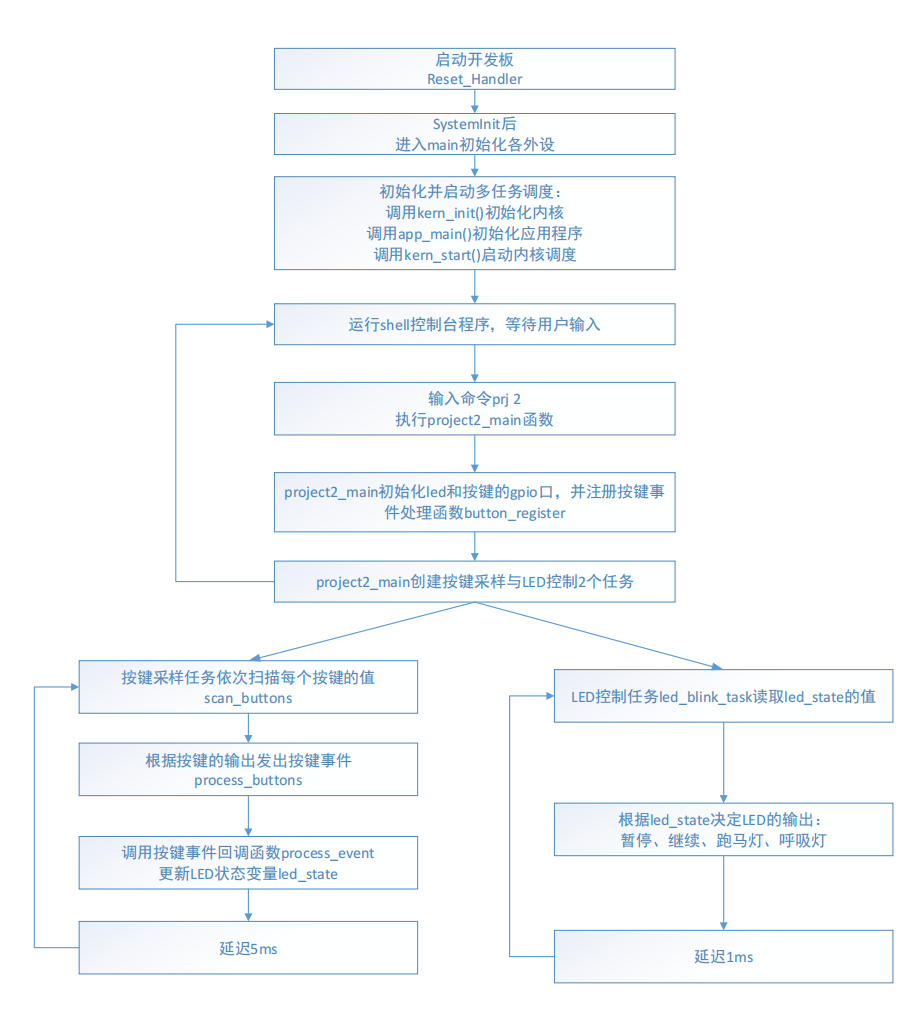
下面模拟K6——长按，呼吸灯。

可以看到波形正确，成功：

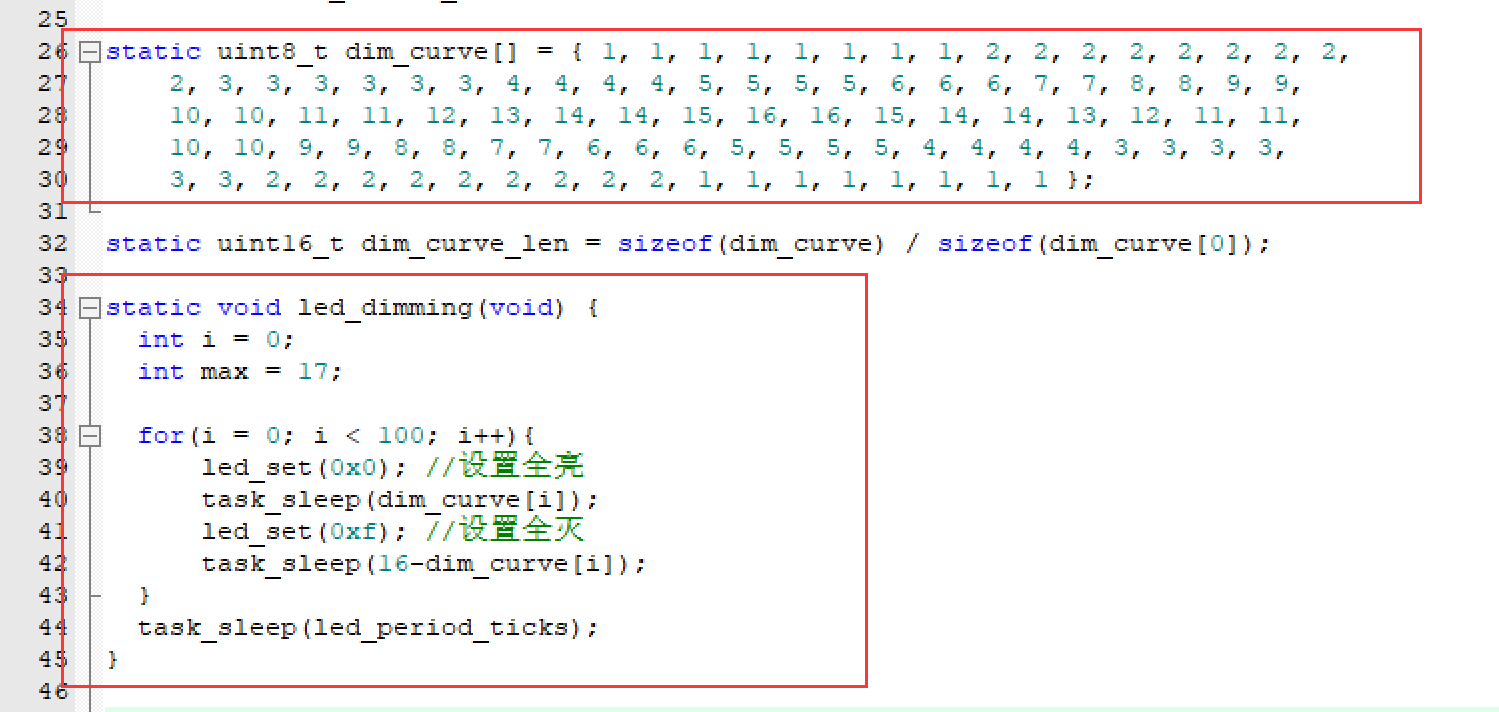


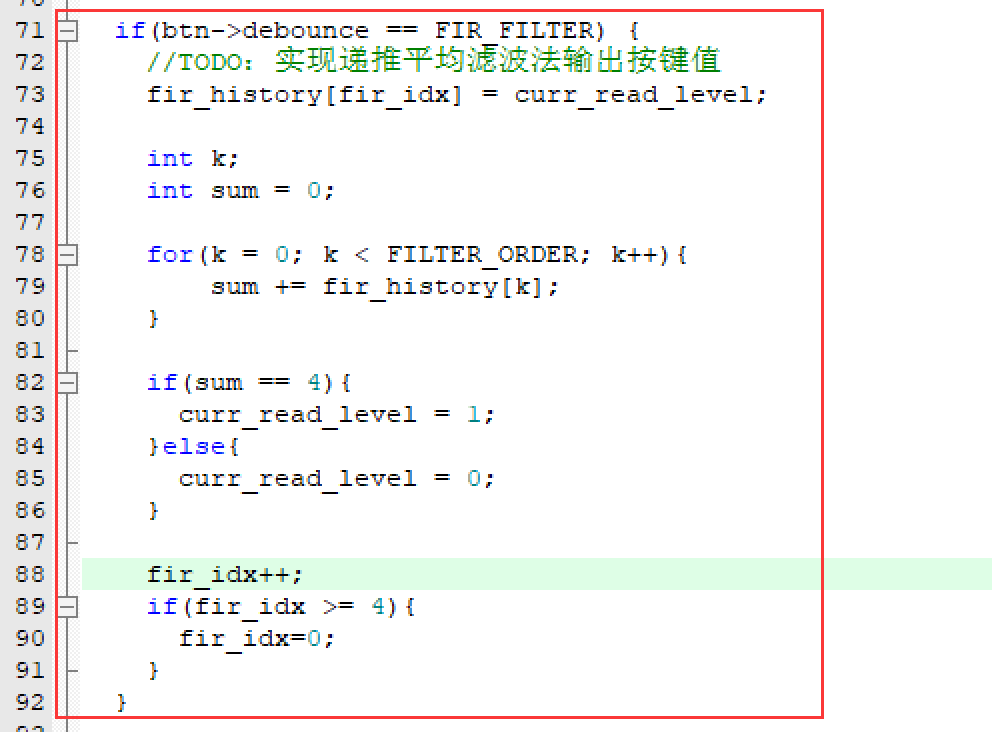
# 实验方案与实现

## 软件结构



## 源代码





# 实验结果与分析

1.请描述从按下按键到 LED 灯改变状态的程序运行流程

每隔5ms扫描一次按键button的状态，若识别到摁下了按键，则会根据按键号对应地修改存储LED灯的状态变量led\_state。每隔1ms，LED控制任务会读取led\_state的值，根据值的内容改变LED灯的状态。

2.本实验通过轮询方式（不间断扫描）检查按键状态，因此会一直占用 CPU

资源，这对低功耗应用场景不友好。如果要降低正常工作模式下的功耗，该如何

处理？

可以选择不使用轮询的方式，而是采用外部中断的方法，按键按动时会产生高低电平的变化，可以将按键管脚设置为边缘触发中断。

# 实验总结

通过本次实验，我对GPIO外设端口的有关知识有了一定的了解，学会了相关的操作，同时了解了一些有关按键消抖的方法，收获颇多。