



**数字集成电路设计实验报告**

|  |  |
| --- | --- |
| 实验题目： | 实验四 中断机制 |
|  |  |
| 学生学号： | 35320212200396 |
| 学生姓名： | 黄腾熙 |
| 院 系： | 电子科学与技术学院 |
| 专 业： | 集成电路设计与集成系统 |

# 实验四 中断机制

# 一、实验原理

本实验通过设计和实现基于中断机制的AHB定时器和UART外设，探索实时任务处理的原理。硬件部分包括定时器中断信号的触发与清除逻辑、UART接收FIFO非空中断的设计，并将中断信号连接至处理器。软件部分通过汇编语言编写中断服务例程，实现定时器中断驱动的计数器功能及UART中断驱动的串口通信。调试环节结合Keil MDK和硬件信号分析工具，验证了中断机制的设计及其在实时任务中的应用。

# 二、实验步骤

1、程序改成用C语言实现

2、使用中断机制实现实验1的开关控制LED灯，并且每拨动一次开关，在PC串口终端打出一个任意的可显字符

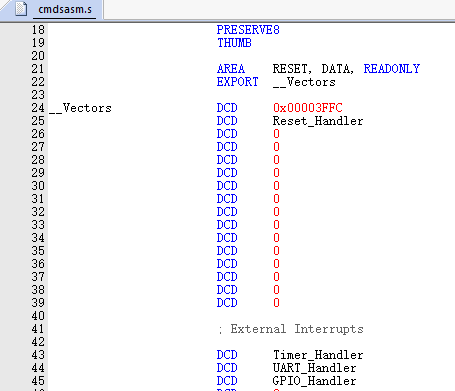
3、在题目2的基础上，输出字符为所拨动开关的序号（从0开始）

4、在题目2的基础上，使用睡眠指令实现低功耗。通过把CPU的SLEEPING和SLEEPDEEP信号接到LED灯上，观察并描述这两个信号在拨动开关时的变化（由于中断处理时间很短，要观察到灯的变化可以在ISR中加入延迟，或者直接用ILA抓取实时波形），并且与实验1做对比

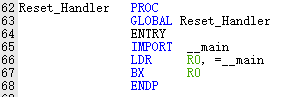
# 三、实验结果分析

**（一）程序改成用C语言实现**

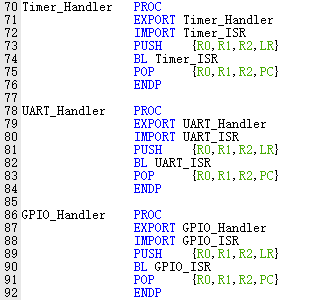
1、首先在汇编程序中添加定时器、串口中断



同时默认跳转到main函数

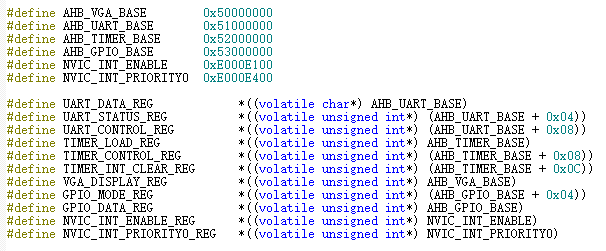


并且增加各模块的ISR函数入口

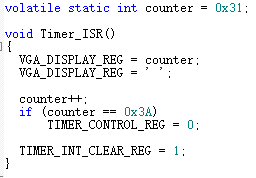


其中PUSH即为在进入ISR之前的保存操作，POP即为ISR执行结束后的恢复操作

2、在main.c文件中增加对应的宏定义



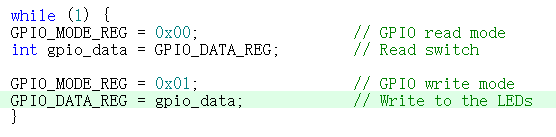
3、编写对应的ISR程序，其中定时器ISR即为在屏幕上显示递增数字，并用空格隔开



串口ISR即为回环函数

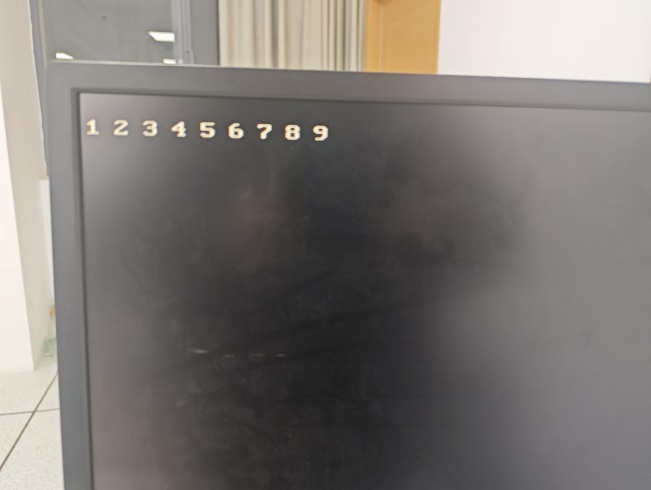


在主循环中不断检测开关状态并输出到对应的LED上

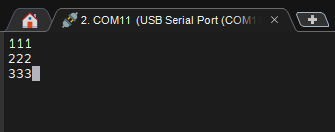


4、对程序进行测试

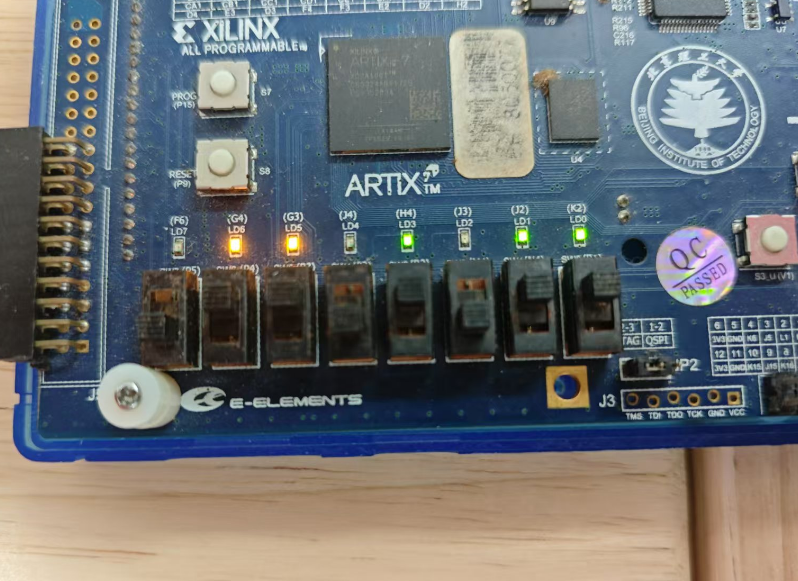
显示效果如图所示，显示效果正常



串口回环效果正常



LED显示效果如下，可见开关和LED状态均对应，说明C语言main函数功能正常



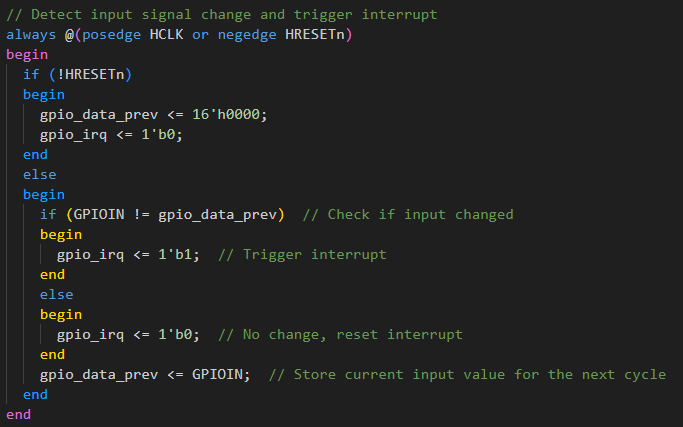
**（二）使用中断机制实现实验1的开关控制LED灯，并且每拨动一次开关，在PC串口终端打出一个任意的可显字符**

**（三）在题目2的基础上，输出字符为所拨动开关的序号（从0开始）**

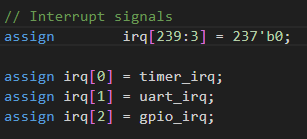
**因为要求2和要求3有较多相似，在此合并分析**

1、首先分析进入中断的条件为GPIO有输入，即开关状态变化

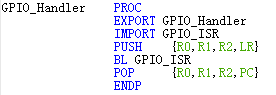
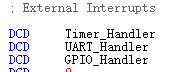
因此修改Verilog代码，增加以下功能：时刻保存GPIO当前状态，并与输入状态进行比较，如果输入变化，就拉高中断信号



之后将中断信号连接到软核的IRQ接口

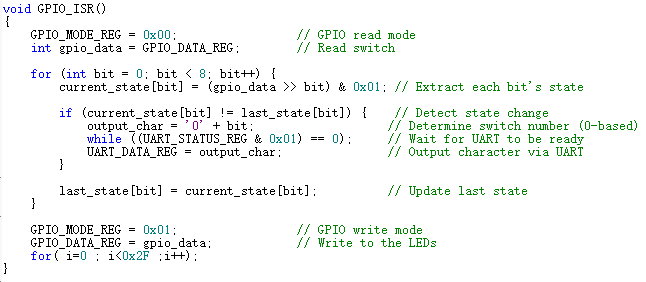


2、随后在汇编文件中增加GPIO中断

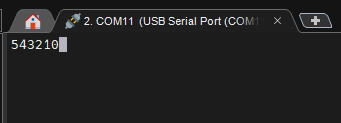


在C程序中增加GPIO\_ISR函数，ISR功能为将LED状态与开关状态对应。

这里实现的方法为：将读取到的状态存在一个长度为8的字符数组中，并与存储的上一个状态比较，一旦状态发生变化就向串口输出对应的字符



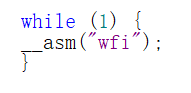
3、GPIO的ISR执行效果如下，当按顺序拨动开关时，串口能正常输出字符



以上即实现了拨动开关输出字符的要求

**（三）在题目2的基础上，使用睡眠指令实现低功耗。通过把CPU的SLEEPING和SLEEPDEEP信号接到LED灯上，观察并描述这两个信号在拨动开关时的变化（由于中断处理时间很短，要观察到灯的变化可以在ISR中加入延迟，或者直接用ILA抓取实时波形），并且与实验1做对比**

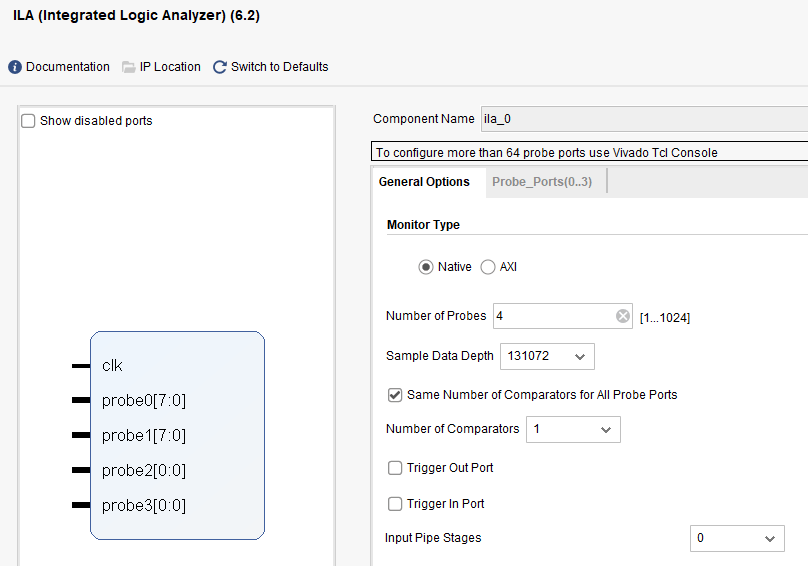
1、首先在main函数中执行wfi指令，使核心一直处于休眠状态



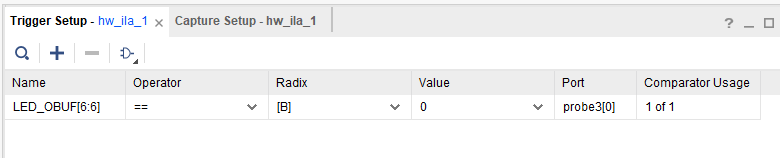
同时，将LED6作为SLEEP的状态显示灯



2、配置ILA用于捕获中断



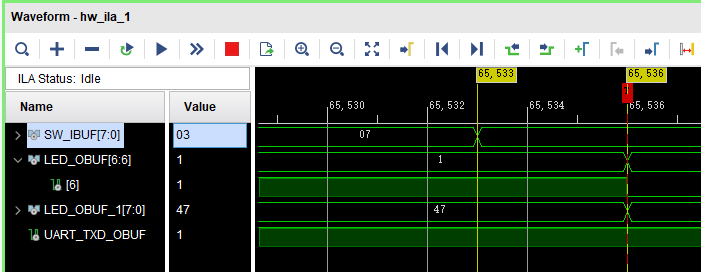
3、将LED6（即SLEEP状态指示管脚）设为触发条件，当LED6熄灭时（退出SLEEP状态），即能捕获所需的信号波形



4、以下为捕获的波形

其中SW状态由0x07变为0x03，此刻SW2被关闭

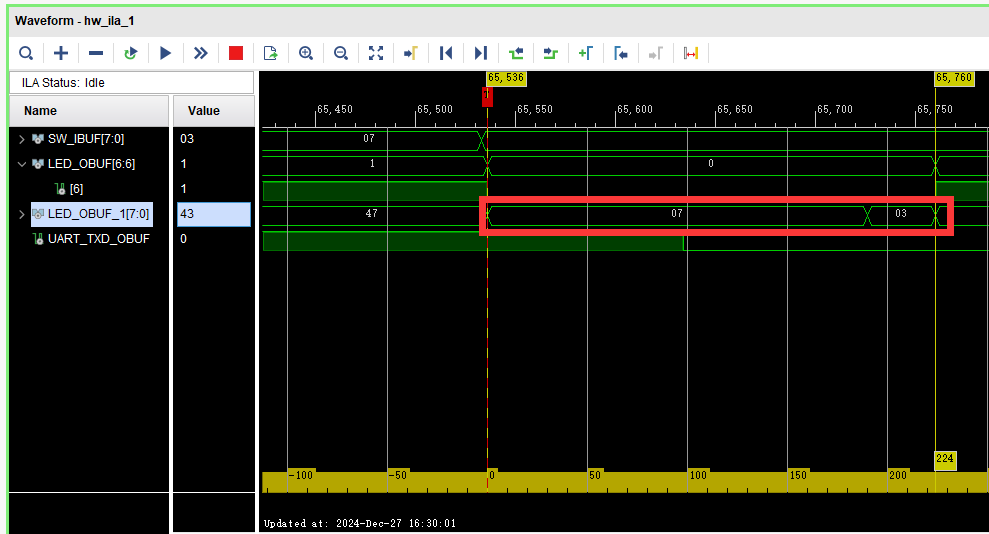
SW关闭3个周期后，LED6被拉低，此时退出睡眠状态，开始执行中断程序



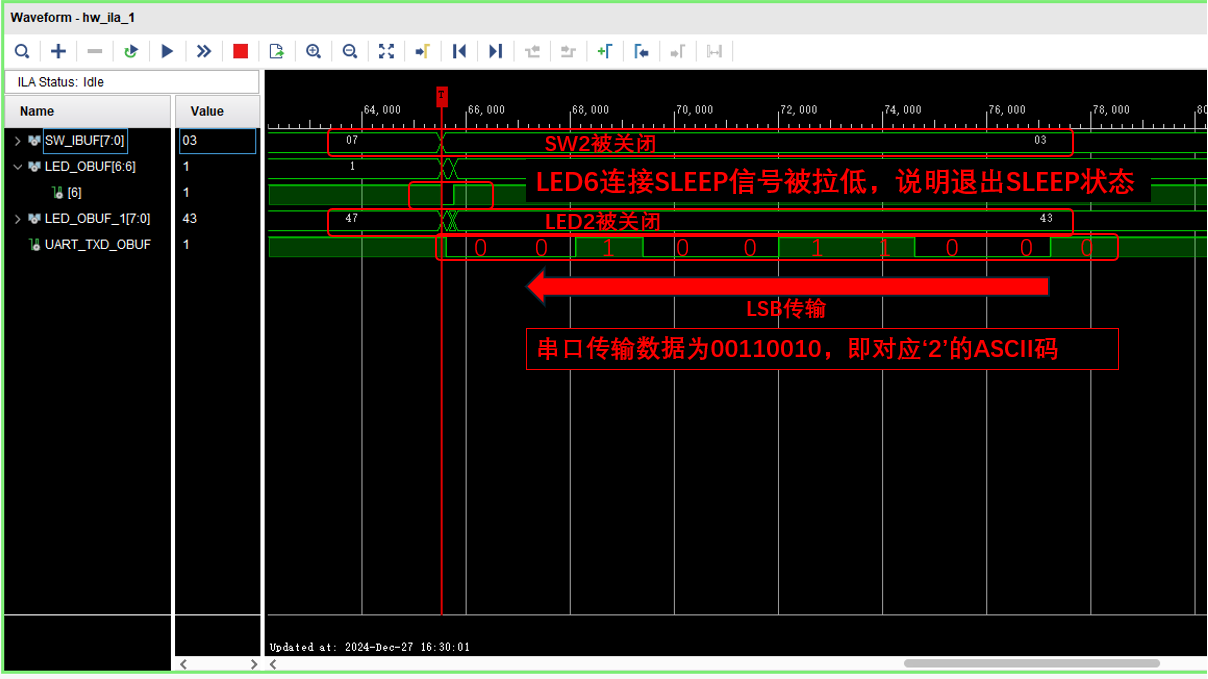
在224个周期后，LED输出由0x07变为0x03，此时GPIO中断执行完毕。

同时，UART输出拉低，开始发送串口数据

随后，系统重新进入睡眠状态



以下为拨动开关后的完整波形，可以发现正常退出SLEEP状态并执行中断，并恢复SLEEP状态，同时后续也完整发送了串口数据



同时，发送串口数据并用ILA抓取波形，可以发现正常退出SLEEP并执行UART中断，并回送字符‘a’

