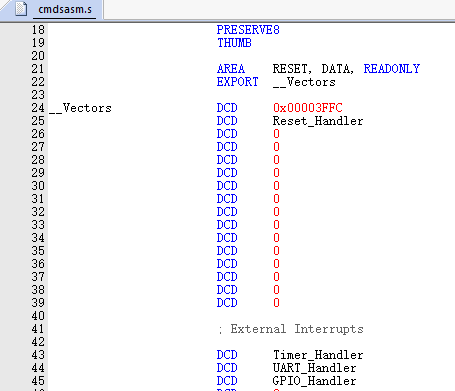
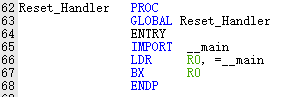
1、 程序改成用C语言实现。

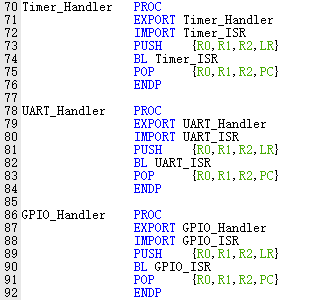
首先在汇编程序中添加定时器、串口中断



同时默认跳转到main函数

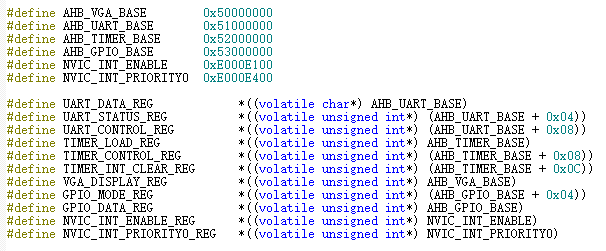


并且增加各模块的ISR函数入口

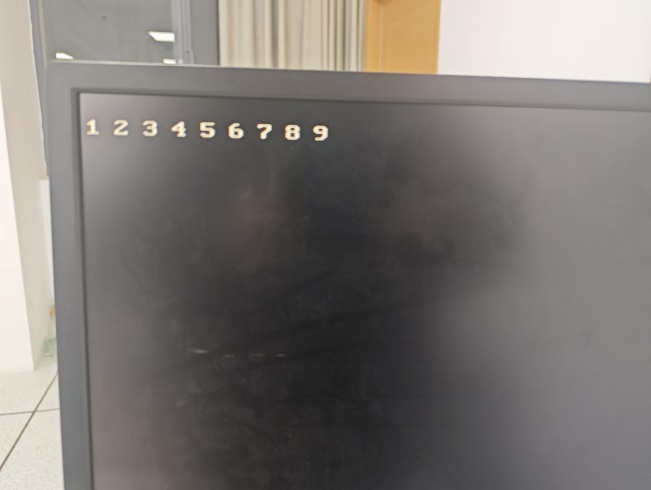
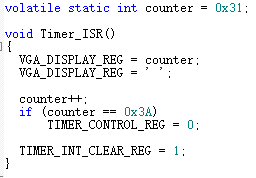


其中PUSH即为在进入ISR之前的保存操作，POP即为ISR执行结束后的恢复操作

随后在main.c文件中增加对应的宏定义



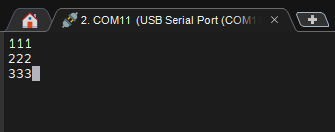
并且编写对应的ISR程序，其中定时器ISR即为在屏幕上显示递增数字，并用空格隔开，显示效果如图所示，显示效果正常



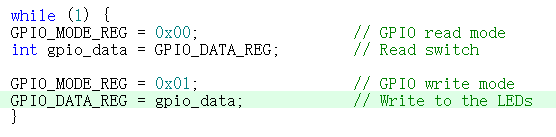
串口ISR即为回环函数



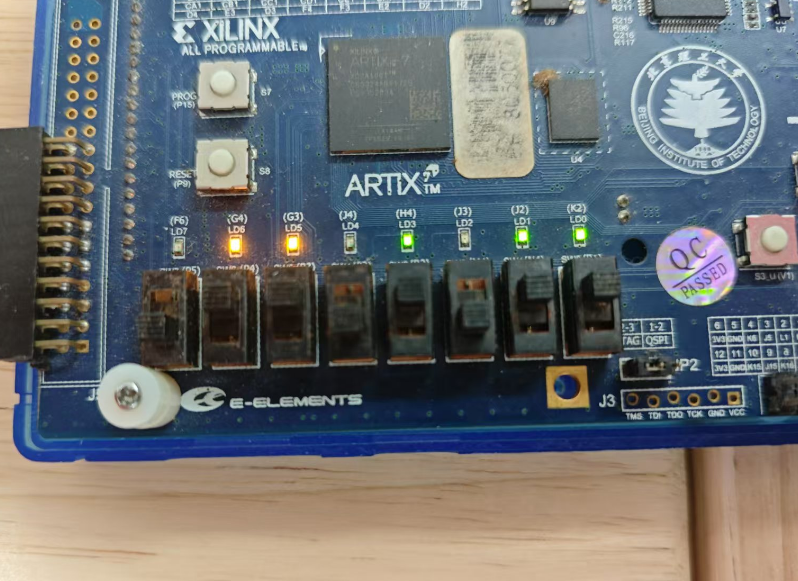
串口回环效果正常



在主循环中不断检测开关状态并输出到对应的LED上



显示效果如下，可见开关和LED状态均对应，说明C语言程序实现成功



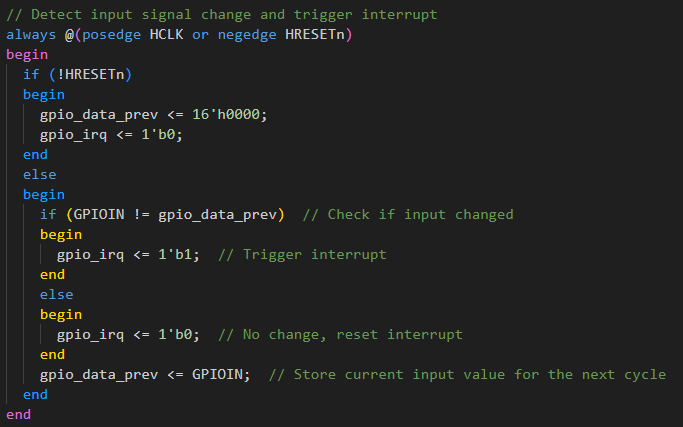
2、 使用中断机制实现实验1的开关控制LED灯，并且每拨动一次开关，在PC串口终端打出一个任意的可显字符。

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）。

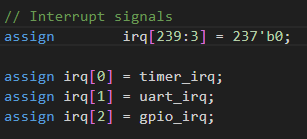
因为要求2和要求3有较多相似，在此合并分析

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，增加以下部分，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，就发送中断信号

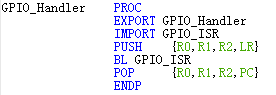
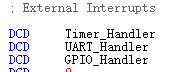


之后将中断信号连接到软核的IRQ接口



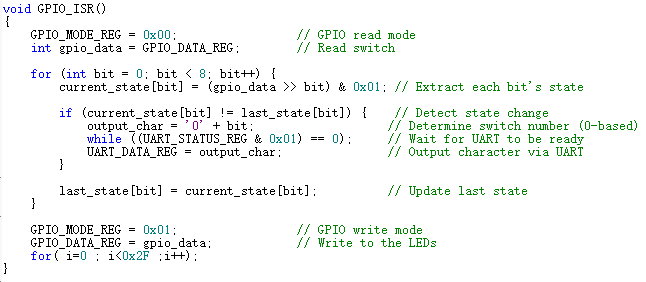
以上即为硬件部分的修改

随后在汇编文件中增加GPIO中断

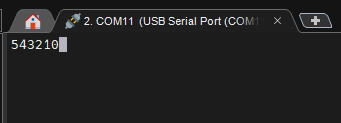


并且在C程序中增加GPIO ISR函数，GPIO的ISR功能为将开关状态反映到LED上

这里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



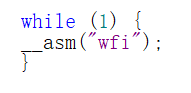
GPIO的ISR执行效果如下，当按顺序拨动开关时，串口能正常输出字符



以上即实现了拨动开关输出字符的要求

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。

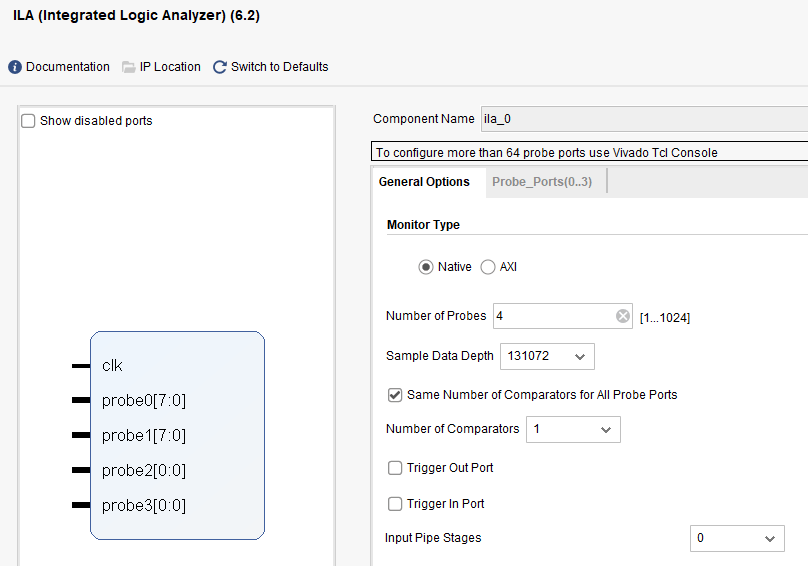
首先在main函数中执行wfi指令，使核心一直处于休眠状态



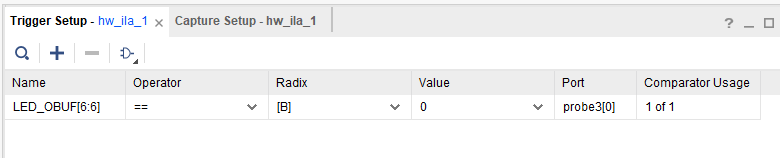
同时，将LED6作为SLEEP的状态显示灯



最后，配置ILA用于捕获中断



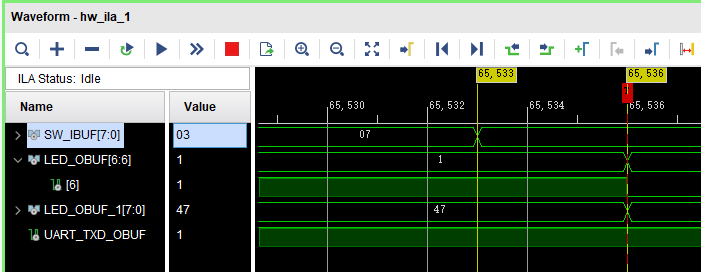
并且将LED6（即SLEEP状态指示管脚）设为触发条件，当LED6熄灭时（退出SLEEP状态），即能捕获所需的信号波形



以下为捕获的波形

其中SW状态由0x07变为0x03，说明此刻SW2被关闭

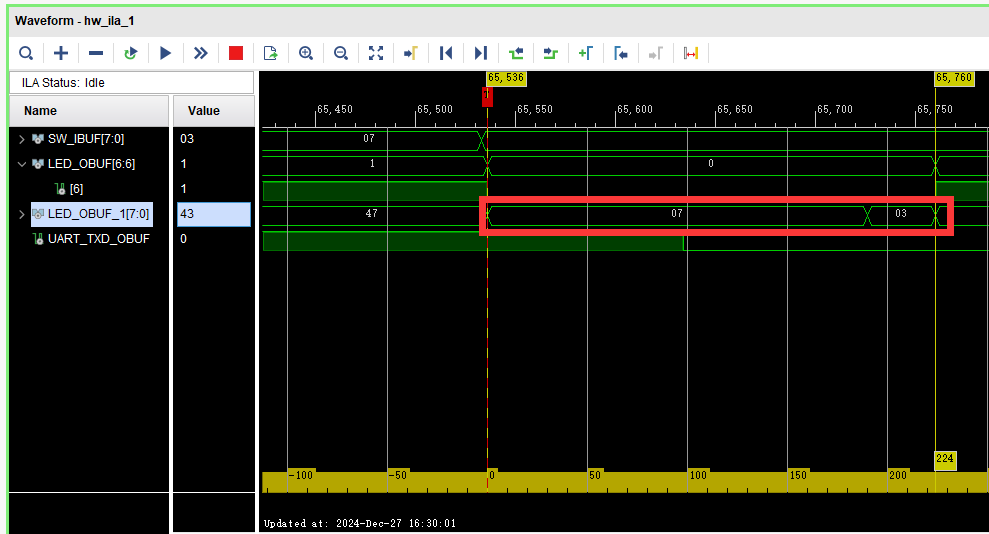
在3个周期后，LED6被拉低，此时退出睡眠状态，开始执行中断程序



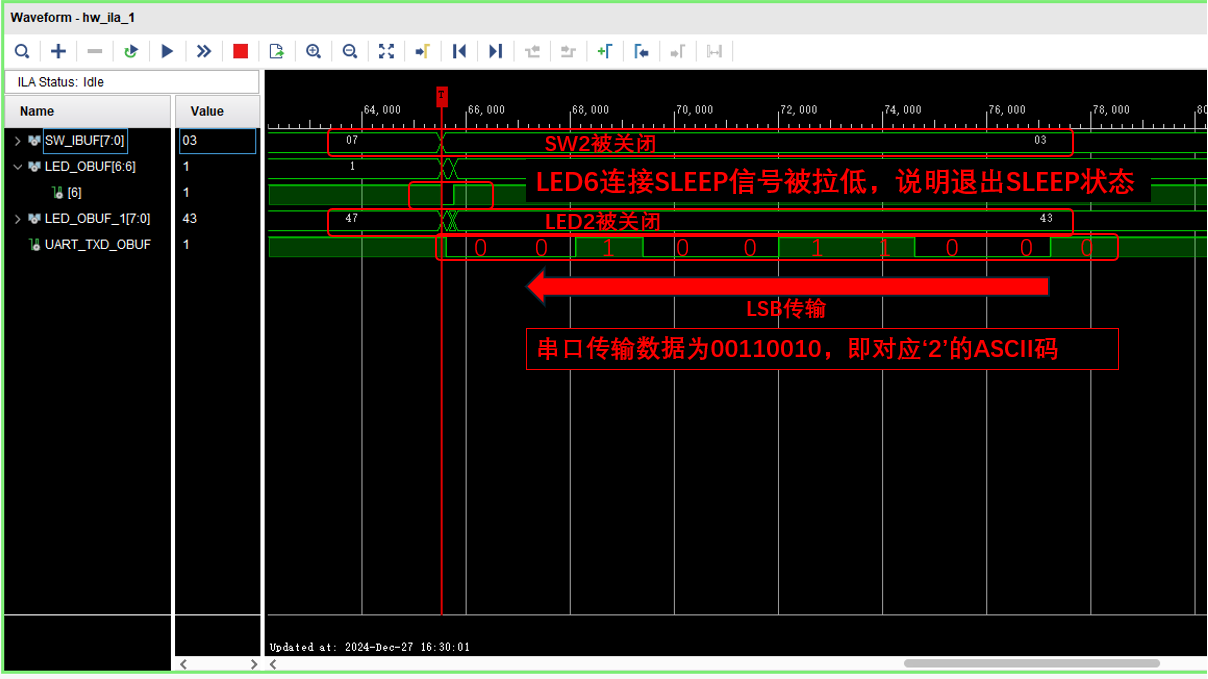
在224个周期后，LED输出由0x07变为0x03，此时GPIO中断执行完毕。

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’

