任务4-3**多线程运行机制实验**

任务描述：本任务我们进一进研究多线程的运行机制。要求实现功能如下：创建两个线程，线程名称分别“LED”和“BEEP”，两个线程的任务是连续5次打印本线程的名字后退出线程。

设计本任务的目的是：观察LED线程和BEEP线程在操作系统中如何同时运行。

一、编写代码

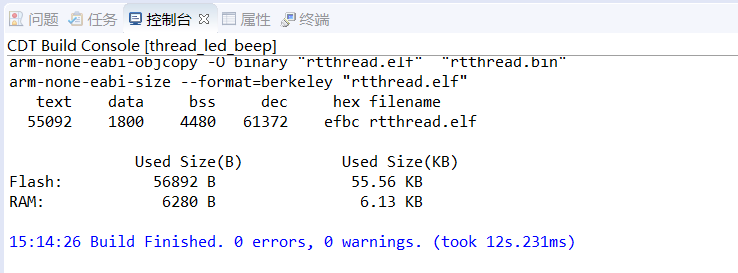
复制“操作指导\4-3”目录下的main.c文件到项目applications目录下

三、编译下载

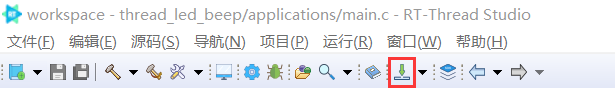
1、编译，点击下图构建按钮重新编译



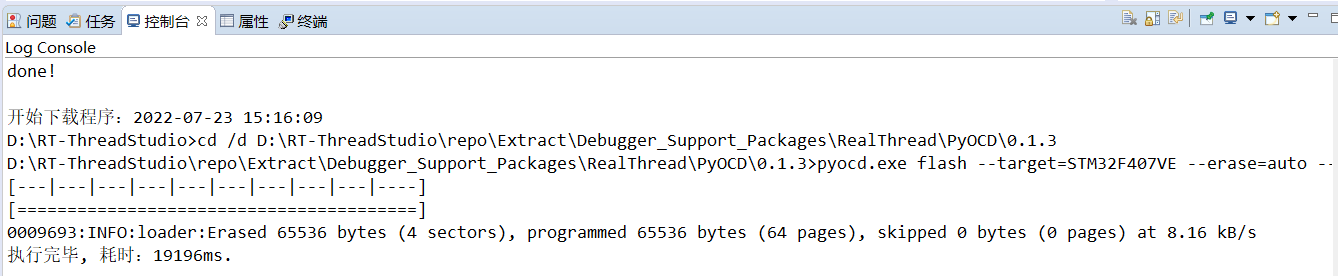
2、编译完成后，如下所示：



3、下载，点击如下图中的下载按钮进行程序下载



下载完成结果如下：



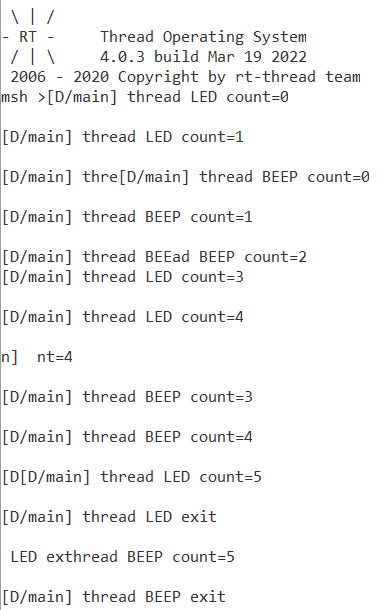
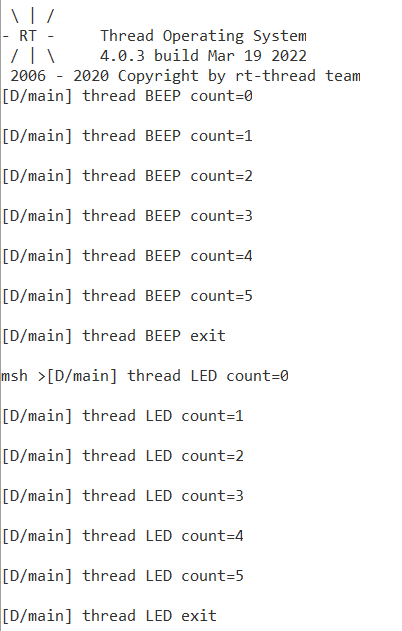
**四、程序测试**

(1) 先使用终端连接开发板，然后再按开发板的重启按键重启系统。观察终端打印的调试信息如图4-13(a)所示，发现两个线程轮流输出信息，可以间接说明两个线程是轮流执行的。

(2) 把上述代码的BEEP线程优先级改为19，如下所示：

#define THREAD\_PRIORITY\_BEEP 19

修改后重新构建后下载运行程序，按照（1）的方法，观察终端打印的调试信息如图4-13 (b)所示，可以看到，即使LED线程先于BEEP被创建，由于BEEP线程的优先极高于LED线程，所以BEEP线程被执行，而且要等到BEEP线程执行完成后LED线程才能执行。

（a） (b)

图4-13 优先级对比

（3）打开预处理宏定义“SCHEDULER\_HOOK”，把LED和BEEP两个线程优先级都设置为20，如下所示：

#define THREAD\_PRIORITY\_LED 20

#define THREAD\_PRIORITY\_BEEP 20

#define SCHEDULER\_HOOK

修改后重新构建后下载运行程序，观之终端打印的调试信息如图4-14 (a)所示， 可以看到，系统先运行main线程，再运行tshell线程，这是因为系统中main线程优先级默认为10，比tshell默认优先级20高（数值越小优先级越高），所以系统先运行main线程。

tshell运行后LED线程和BEEP线程接着轮流运行，由于这3个线程的优先级别都是20，所以，它们在属于同一个优先级队列中，并且按启动先后顺序排列（注意，是启动顺序，即rt\_thread\_startup()函数的执行顺序，而不是创建顺序），从而在调度顺序上也是按照启动先后顺序进行调度。

LED线程和BEEP线程退出后，进入tidle0线程运行，tidle0优先级在系统中最低，当所有高优先级的线程退出或者睡眠时，会进和tidle0线程运行。

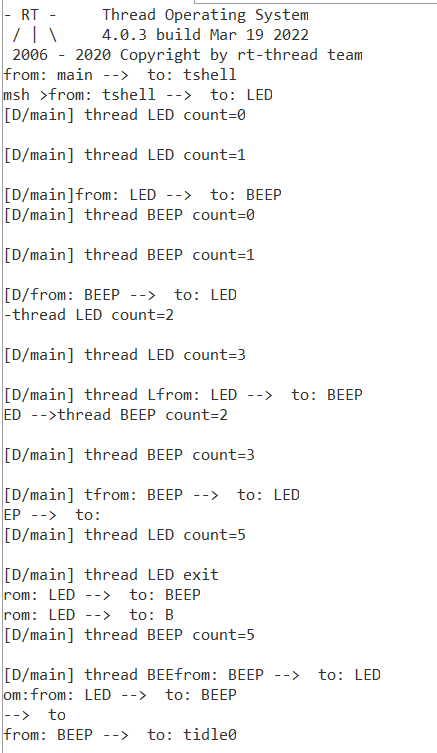
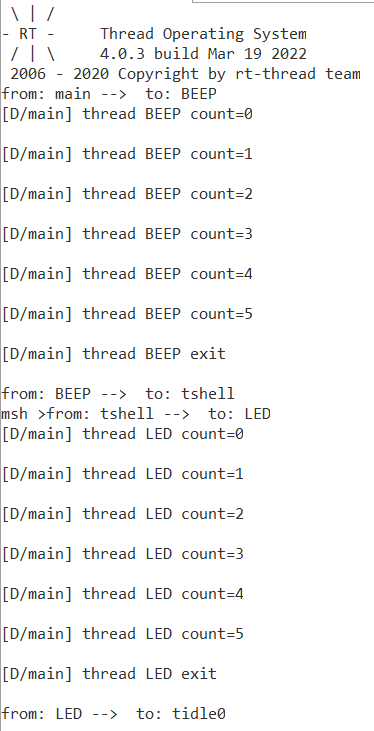
（4）打开预处理宏定义“SCHEDULER\_HOOK”，把LED和BEEP两个线程优先级分别设置为20和19，如下所示：

#define THREAD\_PRIORITY\_LED 20

#define THREAD\_PRIORITY\_BEEP 19

#define SCHEDULER\_HOOK

修改后重新构建后下载运行程序，观之终端打印的调试信息如图4-14(b)所示， 可以看到，系统线程的运行顺序为：main线程->BEEP线程->tshell线程->LED线程，读者可自行分析原因。

1. (b)

图4-14 调度钩子

结论：在操作系统中，所有线程各自独立运行，所有线程看起来是同时工作，但在只有一个CPU核的情况下，在同一时刻只能有一个线程在CPU上运行，操作系统为每个线程分配一定的运行时间片，当线程的运行时间耗尽，操作系统会调度下一个线程到CPU运行。由于时间片很小，使我们觉得线程是在同时运行的。