# 第一周测试习题

用户名：\_\_\_\_\_139+sharemon\_\_\_\_\_\_ 联系邮箱：\_\_\_sharemon@163.com\_\_\_\_\_\_\_\_

（用户名格式： 学号+电子发烧友用户名）

**温馨提示：**

亲爱的同学，你好！

为了帮助你测试自己对课程内容的掌握程度，特地编写了此试题。在做题之前，请摒弃学生时代的那种答题方式，不要为了分数而答题，而是以自我检测为目标。答题时如遇难题可随时翻阅课件资料，编写答案时可自由发挥、多写多画图，以便更好地检验自己的不足！

希望能够通过视频课程以及试题检测，更好地帮你掌握这些技术知识。如果通过课程学习带来的收益能够帮助你提升技术水平， 甚至获得工作提升，那将是我莫大的荣幸！

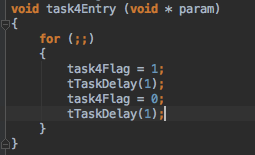
@01课堂 李述铜

**注意事项：**

1. 所有试题均基于《自己动手从0到1写嵌入式操作》课程中所用的软硬件完成，除非特别指出，否则试题内容的各项基础条件均与课程中所指一致
2. 试题类型包含选择题、简单题、程序设计题。选择题和简答题请直接在此文档中填写；程序设计题请在文档中说明设计原理，随同工程一起提交
3. 本试题非100分制，而是答对获得相应的分数，答错不计分，最后一起累计总分
4. 请在截止时间前提交试题，逾期不批改，也不计入相应分数
5. 完成试题后，提交至course@ilishutong.com和[qianshanshan@elecfans.com](mailto:qianshanshan@elecfans.com)，文档和工程分别作为两个附件。

# 选择题

1、单选题。如下所所示为某任务的代码：





请问：tTaskInit()中设置了初始运行参数值0x44444444，之后是通过何种方式将param设置成0x44444444传递给任务？（3分）

A：将参数值写入任务堆栈，在运行需要时从堆栈直接取值

B：将参数值写入任务堆栈，运行前堆栈弹出到R0寄存器

C：将参数值写入任务堆栈，内核将值赋值变量param

D：将参数值写入tTask4结构，内核运行前赋值给param

请填入答案：( C ) （2分）

# 简答题

1、请结合switch.c中PendSVC\_Handler()中的如下三行代码，说明各汇编语句的具体功能含义（5分），以及为什么仅通过下面的语句便可向currentTask->stack写入新值（2分）？

LDR R1, =currentTask

LDR R1, [R1]

STR R0, [R1]

LDR R1, =currentTask ; R1 = &currentTask, 将currentTask的地址存入R1寄存器

LDR R1, [R1] ; R1 = \*R1, 将currentTask的值存进R1寄存器

STR R0, [R1] ; \*R1 = R0, 将R0中的值传到currentTask指向的Task\_t结构体

有上面的注释可以看到，最后currentTask指向的Task\_t结构体被赋值。而Task\_t结构体的第一个变量就是stack，所以stack跟结构体地址相同，Task\_t结构体被赋值就是stack被赋值。所以通过这几条语句可以向currentTask->stack写入新址

2、对于学习RTOS而言，最核心的是理解任务切换的原理。请用文字并结合示意图详解如下问题：

* 为什么在创建任务时，要对任务的堆栈预先填入一些初始化数值？（3分）为什么要按指定的次序依次填入？（2分）为什么要填入这些数值，而不是其它？（2分）

1. 因为第一次任务切换的时候，内核会自动从堆栈中POP出一些运行参数，所以需要在初始化的时候预先填入一些值，以防任务切换的时候发生异常。
2. 因为任务切换的时候，内核压入堆栈的运行参数的顺序是固定的，所以需要按照规定的顺序依次填入。
3. 因为任务切换的时候，内核压入堆栈的每个运行参数是定义好的，所以需要在初始化的时候跟定义好的这些参数匹配一致。

* 系统启动后，通过具体怎样一个过程切换到第一个任务？（5分）
* 系统启动
* 初始化任务
* 触发PendSV异常进行任务调度（把PSP设成0以表示这是第一次调度）
* 取出第一个任务的运行参数（人为保存的部分）
* 将堆栈指针PSP指向第一个任务的堆栈
* 结束任务调度函数 (这时内核会自动取出内核保存的那些运行参数)
* 切换到第一个任务
* 在任务之间进行切换时，通过具体怎样一个过程切换到另一个任务？（5分）为什么当下次再次切换回某任务时，该任务能够从之前的位置继续往下运行，好像什么都没有发生一样？（2分）

任务切换流程：

* 任务1运行
* 任务1请求任务切换
* 触发PendSV异常， 进入PendSV异常处理函数（这时内衣已经自动保存好了当前任务的一部分运行参数）
* 保存当前任务（任务1）的其他运行参数
* 取出任务2的运行参数
* 将堆栈指针指向任务2的堆栈
* 结束任务调度 （这时系统会自动取出剩余的运行参数）
* 切换到任务2

之所以任务切换之后，任务还能像没事人似的继续运行，是因为在任务被切换出去之前程序保存了任务的运行参数，在任务切换回来之前程序又恢复了任务所需的运行参数。

1. 请说明当系统中创建了多个相同优先级任务后，rtos中调度这些同优先级的任务时，遵循什么样的策略将CPU分配给各个任务？（2分）为什么要采用这种策略？（3分）

优先级相同的任务会遵循时间片轮转策略进行任务切换，即每个任务各自占用CPU一段时间来运行。采用这种策略可以保证每个任务都能够得到一定的运行时间。

4、任务需要占用资源，比如cpu、堆栈。一般来讲，任务数少，占用的资源也相对尽少，整个系统在任务管理上的开销也减少。既然如此，为什么还要创建一个似乎什么也不干的空闲任务？（3分）此外，我们能否让空闲任务执行tTaskDelay()系统延时操作？（3分）

空闲任务的存在是为了让CPU在所有任务都处于等待状态下的时候有事情可以做，所以不能让空闲任务执行tTaskDelay()。

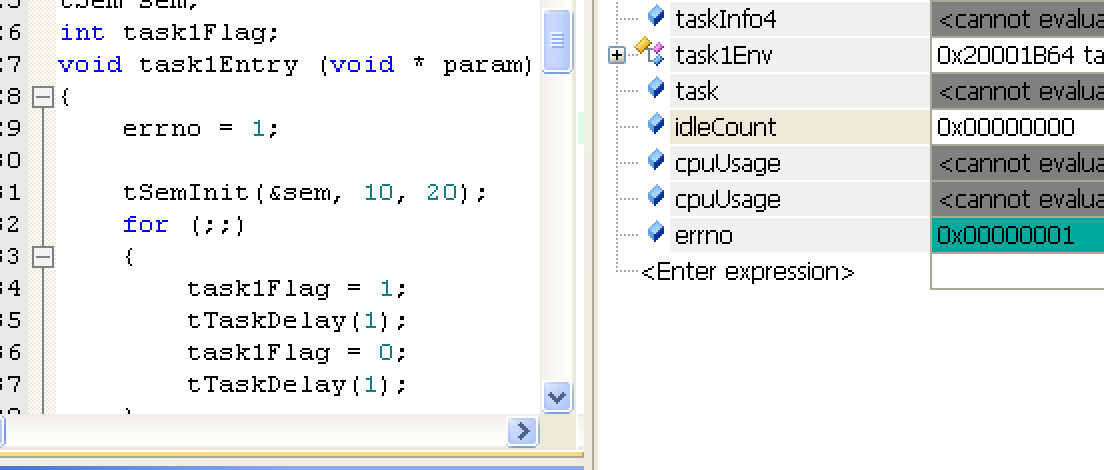
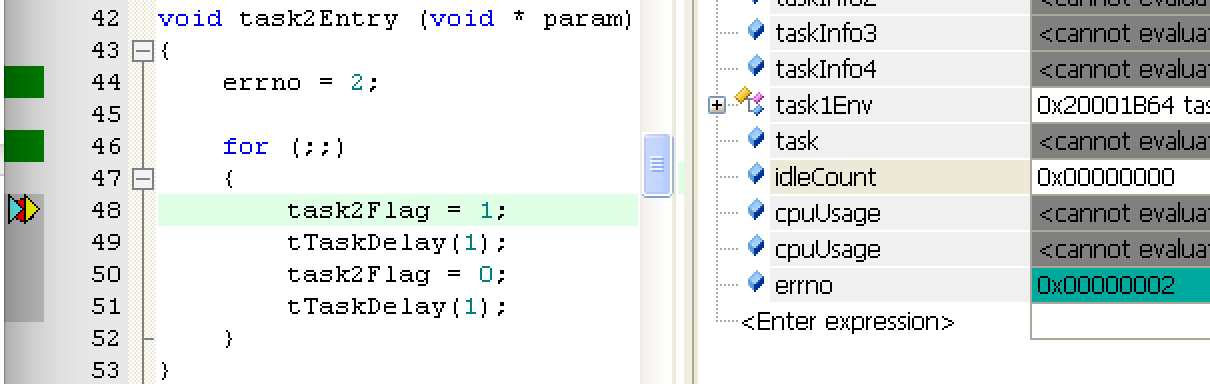
# 程序设计

1、在目前的OS实现中，通过将全部CPU内核寄存器的值在任务堆栈中保存和恢复，便实现了任务运行状态的保存和恢复。在视频课程中，我们实际上提到了任务可能还有一些其它的状态数据需要保存，这些状态数据也可以保存到堆栈中。例如，我们希望为每个任务配备一个errno变量，对于任务来说，它可以通过类似直接如下代码实现对errno的访问：

int readErrnor = errno；

errno = writeErrno;

为了实现这样的功能，我们可以将errno视为一种任务的其它状态数据。这样需要事先定义一个全局变量int errno，然后在每个任务运行前，将任务堆栈中的errno值恢复到errno变量，当任务暂停运行时，将errno值保存到堆栈中。

下面请在代码中实现这样的机制，并简要说明实现原理。最终的效果应当为： 每个任务都可以对errno进行赋值，例如task1先运行对errno写1；其后task2运行对errno写2。当再次运行至task1时，在task1中观察到的errno仍为1；再次运行至task2时，在task2中观察到的errno仍为2。（15分）

全部代码参见工程文件。

实现原理： 在任务切换的时候，保存完R11-R4，在当前任务堆栈的下一个地址保存此时的errno值。同时在下一个任务堆栈中相应的位置取出下一个任务的errno值付给errno变量。

主要代码：

\_\_asm void PendSV\_Handler(void)

{

IMPORT curTask

IMPORT nxtTask

IMPORT errno

MRS R0, PSP

CBZ R0, PendSVHandler\_firstRun

**LDR R1, =errno** ; R1 = &errno

**LDR R1, [R1]** ; R1 = errno

**STMDB R0!, {R1, R4-R11}** ; save R1(errno) after R4 to stack

LDR R1, =curTask

LDR R1, [R1]

STR R0, [R1]

PendSVHandler\_firstRun

LDR R0, =curTask ; R0 = &curTask

LDR R1, =nxtTask ; R1 = &nxtTask

LDR R1, [R1] ; R1 = nxtTask

STR R1, [R0] ; \*(&curTask) = nxtTask

LDR R1, [R1] ; R1 = \*nxtTask (nxtTask.stackPtr)

**LDMIA R1!, {R2, R4-R11}** ; Load R2 (errno), R4-R11 from stack

MSR PSP, R1 ; PSP = R1

**LDR R1, =errno** ; R1 = &errno

**STR R2, [R1]** ; errno = R2, restore errno

ORR LR, LR, #0x04 ; set code running on user level to match PSP

BX LR

}