6.1 任务创建和删除实验

6.1.1 OSTaskCreate()函数

UCOSIII 是多任务系统，那么肯定要能创建任务，创建任务就是将任务控制块、任务堆栈、任务代码等联系在一起，并且初始化任务控制块的相应字段。在 UCOSIII 中我们通过函数 OSTaskCreate()来创建任务，OSTaskCreate()函数原型如下(在 os_task.c 中有定义)。调用 OSTaskCreat()创建一个任务以后，刚创建的任务就会进入就绪态，注意！不能在中断服务程序中调用 OSTaskCreat()函数来创建任务。

```
void    OSTaskCreate (OS_TCB                *p_tcb,
                      CPU_CHAR              *p_name,
                      OS_TASK_PTR           p_task,
                      void                  *p_arg,
                      OS_PRIO               prio,
                      CPU_STK               *p_stk_base,
                      CPU_STK_SIZE          stk_limit,
                      CPU_STK_SIZE          stk_size,
                      OS_MSG_QTY            q_size,
                      OS_TICK               time_quanta,
                      void                  *p_ext,
                      OS_OPT                opt,
                      OS_ERR                *p_err)
```

***p_tcb：** 指向任务的任务控制块 OS_TCB。

***p_name：** 指向任务的名字，我们可以给每个任务取一个名字

p_task： 执行任务代码，也就是任务函数名字

***p_arg：** 传递给任务的参数

prio： 任务优先级，数值越低优先级越高，用户不能使用系统任务使用的那些优先 级！

***p_stk_base：** 指向任务堆栈的基地址。

stk_limit： 任务堆栈的堆栈深度，用来检测和确保堆栈不溢出。

stk_size： 任务堆栈大小

q_size： UCOSIII 中每个任务都有一个可选的内部消息队列，我们要定义宏 OS_CFG_TASK_Q_EN>0，这是才会使用这个内部消息队列。

time_quanta： 在使能时间片轮转调度时用来设置任务的时间片长度，默认值为时钟节拍除以 10。

***p_ext：** 指向用户补充的存储区。

opt： 包含任务的特定选项，有如下选项可以设置。

OS_OPT_TASK_NONE	表示没有任何选项
OS_OPT_TASK_STK_CHK	指定是否允许检测该任务的堆栈
OS_OPT_TASK_STK_CLR	指定是否清除该任务的堆栈
OS_OPT_TASK_SAVE_FP	指定是否存储浮点寄存器，CPU 需要有浮点运算硬件并且有专用代码保存浮点寄存器。

***p_err：** 用来保存调用该函数后返回的错误码。

6.1.2 OSTaskDel()函数

OSTaskDel()函数用来删除任务，当一个任务不需要运行的话，我们就可以将其删除掉，删除任务不是说删除任务代码，而是 UCOSIII 不再管理这个任务，在有些应用中我们只需要某个任务只运行一次，运行完成后就将其删除掉，比如外设初始化任务，OSTaskDel()函数原型如下：

```
void    OSTaskDel (OS_TCB    *p_tcb,
                   OS_ERR    *p_err)
```

***p_tcb：** 指向要删除的任务 TCB，也可以传递一个 NULL 指针来删除调用 OSTaskDel()函数的任务自身。

***p_err：** 指向一个变量用来保存调用 OSTaskDel()函数后返回的错误码。

虽然 UCOSIII 允许用户在系统运行的时候来删除任务，但是应该尽量的避免这样的操作，如果多个任务使用同一个共享资源，这个时候任务 A 正在使用这个共享资源，如果删除了任务 A，这个资源并没有得到释放，那么其他任务就得不到这个共享资源的使用权，会出现各种奇怪的结果。

我们调用 OSTaskDel()删除一个任务后，这个任务的任务堆栈、OS_TCB 所占用的内存并没有释放掉，因此我们可以利用他们用于其他的任务，当然我们也可以使用内存管理的方法给任务堆栈和 OS_TCB 分配内存，这样当我们删除掉某个任务后我们就可以使用内存释放函数将这个任务的任务堆栈和 OS_TCB 所占用的内存空间释放掉。

6.2 任务挂起和恢复实验

6.2.1 OSTaskSuspend()函数

有时候有些任务因为某些原因需要暂停运行，但是以后还要运行，因此我们就不能删除掉任务，这里我们可以使用 OSTaskSuspend()函数挂起这个任务，以后再恢复运行，函数 OSTaskSuspend()的原型如下：

```
void    OSTaskSuspend (OS_TCB    *p_tcb,OS_ERR    *p_err)
```

***p_tcb ：** 指向需要挂起的任务的 OS_TCB，可以通过指向一个 NULL 指针将调用该函数的任务挂起。

***p_err：** 指向一个变量，用来保存该函数的错误码。

我们可以多次调用 OSTaskSuspend ()函数来挂起一个任务，因此我们需要调用同样次数的 OSTaskResume()函数才可以恢复被挂起的任务，这一点非常重要。

6.2.2 OSTaskResume()函数

OSTaskResume()函数用来恢复被 OSTaskSuspend()函数挂起的任务，OSTaskResume()函数是唯一能恢复被挂起任务的函数。如果被挂起的任务还在等待别的内核对象，比如事件标志组、信号量、互斥信号量、消息队列等，即使使用 OSTaskResume()函数恢复了被挂起的任务，该任务也不一定能立即运行，该任务还是要等相应的内核对象，只有等到内核对象后才可以继续运行，OSTaskResume()函数原型如下：

```
void    OSTaskResume (OS_TCB    *p_tcb,OS_ERR    *p_err)
```

***p_tcb ：** 指向需要解挂的任务的 OS_TCB，指向一个 NULL 指针是无效的，因为该任务正在运行，不需要解挂。

***p_err：** 指向一个变量，用来保存该函数的错误码。