UC/OSⅡ 基本函数

```
*********************************************************************************************************
* 事件标志管理 (EVENT FLAGS MANAGEMENT)
 *
*OSFlagAccept()    检查事件标志组函数(标志组的指针、事件标志位、等待事件标志位的方式、错误码指针)
* OSFlagCreate()    建立一个事件标志组(初值、错误码)
* OSFlagDel()       删除一个事件标志组(指针、条件值、错误值)
*OSFlagPend()       等待事件标志组的事件标志位(事件组指针、需要检查的标志位、等待事件标志位的方式、
* 允许等待的时钟节拍、出错代码的时钟节拍)
* OSFlagPost()      置位或清0事件标志组中的标志位(指针、标志位、条件值、错误码)
* OSFlagQuery()     查询事件标志组的当前事件标志状态(事件标志组的指针、错误代码的指针)
*********************************************************************************************************
```

```
*********************************************************************************************************
* 消息邮箱管理 (MESSAGE MAILBOX MANAGEMENT)
*
* OSMboxAccept ()   查看消息邮箱(消息邮箱指针)
* OSMboxCreate ()   建立并初始化一个消息邮箱(msg 参数不为空含内容)
* OSMboxDel ()      删除消息邮箱(消息邮箱指针、删除条件、出错代码指针)
* OSMboxPend ()     等待一个消息邮箱函数(消息邮箱指针、允许等待的时钟节拍、代码错误指针)
* OSMboxPost ()     发送消息函数(消息邮箱指针、即将实际发送给任务的消息)
* OSMboxPostOpt ()  向邮箱发送一则消息(邮箱指针、消息、条件)
* OSMboxQuery ()    查询一个邮箱的当前状态(信号量指针、状态数据结构指针)
*********************************************************************************************************
```

```
*********************************************************************************************************
* 内存管理项 (MEMORY MANAGEMENT)

*OSMemCreate()   建立并初始化一块内存区(起始地址、需要的内存块数目、内存块大小、返回错误的指针)
* OSMemGet ()     从内存区分配一个内存块
* OSMemPut ()     释放一个内存块，内存块必须释放回原先申请的内存区
* OSMemQuery ()   得到内存区的信息
*********************************************************************************************************
```

```
*********************************************************************************************************
* 互斥型信号量项管理 (MUTUAL EXCLUSION SEMAPHORE MANAGEMENT)
*
* OSMutexAccept ()  无等待地获取互斥型信号量[任务不挂起](信号量指针、错误代码)
* OSMutexCreate ()  建立并初始化一个互斥型信号量(优先级继承优先级(PIP)、出错代码指针)
* OSMutexDel ()     删除互斥型信号量(信号指针、删除条件、错误指针)
* OSMutexPend ()    等待一个互斥型信号量(指针、等待超时时限、出错代码指针)
* OSMutexPost ()    释放一个互斥型信号量(互斥型信号量指针)
* OSMutexQuery ()   查询一个互斥型信号量的当前状态(互斥型信号量指针、状态数据结构指针)
*********************************************************************************************************
```

```
*********************************************************************************************
* 消息队列管理 (MESSAGE QUEUE MANAGEMENT)
*
* OSQAccept ()   检查消息队列中是否已经有需要的消息(消息队列的指针)
* OSQCreate ()   建立一个消息队列(消息内存区的基地址(指针数组)、消息内存区的大小)
* OSQDel ()      删除一个消息队列(消息队列指针、删除条件、错误指针)
* OSQFlush ()    清空消息队列(指向得到消息队列的指针)
* OSQPend ()     任务等待消息队列中的消息(消息队列指针、允许等待的时钟节拍、代码错误指针)
* OSQPost ()     向消息队列发送一则消息FIFO(消息队列指针、发送的消息)
* OSQPostFront () 向消息队列发送一则消息LIFO(消息队列指针、发送的消息)
* OSQPostOpt () 向消息队列发送一则消息LIFO(消息队列指针、发送的消息、发送条件)
* OSQQuery ()   查询一个消息队列的当前状态(信号量指针、状态数据结构指针)
*********************************************************************************************



/*********************************************************************************************
* 消息队列数据 (MESSAGE QUEUE DATA)
*********************************************************************************************
*/
队列控制块是一个用于维护消息队列信息的数据结构，它包含了以下的一些域。这里，仍然在各个变量前加入一个[.]来表示它们是数据结构中的一个域。
* 1).OSQPtr: 在空闲队列控制块中链接所有的队列控制块。一旦建立了消息队列，该域就不再有用了。
*2).OSQStart: 是指向消息队列的指针数组的起始地址的指针。用户应用程序在使用消息队列之前必须先定义该数组
* 3).OSQEnd: 是指向消息队列结束单元的下一个地址的指针。该指针使得消息队列构成一个循环的缓冲区。
*4).OSQIn: 是指向消息队列中插入下一条消息的位置的指针。当.OSQIn和.OSQEnd相等时，.OSQIn被调整指向消息队列的起始单元。
* 5).OSQOut: 是指向消息队列中下一个取出消息的位置的指针。当.OSQOut和.OSQEnd相等时，.OSQOut被调整指向消息队列的起始单元。
* 6).OSQSize: 是消息队列中总的单元数。该值是在建立消息队列时由用户应用程序决定的。在uC/OS-II中,该值最大可以是65,535。
* 7).OSQEntries: 是消息队列中当前的消息数量。当消息队列是空的时，该值为0。当消息队列满了以后，该值和 .OSQSize值一样。 在消息队列刚刚建立时，该值为0。
*********************************************************************************************
*/



/*********************************************************************************************
* 信号量管理 (SEMAPHORE MANAGEMENT)
*
* OSSemAccept()   无条件地等待请求一个信号量函数
* OSSemCreate()   建立并初始化一个信号量(输入一个信号量值)
* OSSemDel()      删除一个信号量(信号指针、删除条件、错误指针)
* OSSemPend ()    等待一个信号量函数(信号量指针、允许等待的时钟节拍、代码错误指针)
* OSSemPost ()   发出一个信号量函数(信号量指针)
* OSSemQuery ()   查询一个信号量的当前状态(信号量指针、状态数据结构指针)
---------------------------------------------------------------------------------------------
*/



/*
*********************************************************************************************
* 任务管理 (TASK MANAGEMENT)
*
* OSTaskChangePrio ()   改变一个任务的优先级(任务旧的优先级、任务新的优先级)
```

```
* OSTaskCreate ()        建立任务(任务代码指针、传递参数指针、分配任务堆栈栈顶指针、任务优先级)
*OSTaskCreateExt ()      建立扩展任务(任务代码指针/传递参数指针/分配任务堆栈栈顶指针/分配任务优先级
* //(未来的)优先级标识(与优先级相同)/分配任务堆栈栈底指针/指定堆栈的容量(检验用)
* //指向用户附加的数据域的指针/建立任务设定选项)
* OSTaskDel ()           删除任务(任务的优先级)
* OSTaskDelReq ()        请求一个任务删除其它任务或自身?(任务的优先级)
* OSTaskResume ()        唤醒一个用OSTaskSuspend()函数挂起的任务(任务的优先级)
* OSTaskStkChk ()        检查任务堆栈状态(任务优先级、检验堆栈数据结构)
* OSTaskSuspend ()       无条件挂起一个任务(任务优先级)
* OSTaskQuery ()         获取任务信息(任务指针、保存数据结构指针)
*********************************************************************************************
*/

/*
*********************************************************************************************
* 时钟管理项 (TIME MANAGEMENT)
*
* OSTimeDly ()          任务延时函数(时钟节拍数)
* OSTimeDlyHMSM ()      将一个任务延时若干时间(设定时、分、秒、毫秒)
* OSTimeDlyResume ()    唤醒一个用OSTimeDly()或OSTimeDlyHMSM()函数的任务(优先级)
* OSTimeGet ()          获取当前系统时钟数值
* OSTimeSet ()          设置当前系统时钟数值
*********************************************************************************************
*/


/*********************************************************************************************
* 混杂函数定义
*
* OSInit()          初始化UCOS-II函数
* OSIntEnter()      中断函数正在执行
* OSIntExit()       中断函数已经完成(脱离中断)
* OSSchedLock()     给调度器上锁
* OSSchedUnlock()   给调度器解锁
* OSStart()         启动多个任务
* OSStatInit()      统计任务初始化
* OSVersion()       获得版本号
*********************************************************************************************
/


/*********************************************************************************************
* 内部函数原型 INTERNAL FUNCTION PROTOTYPES
* 你在应用程序中不能使用它们 (Your application MUST NOT call these functions)
*
* OS_Dummy()            建立一个虚拟函数
* OS_EventTaskRdy()     使一个任务进入就绪态(OS_EVENT *pevent, void *msg, INT8U msk)
* OS_EventTaskWait()    使一个任务进入等待某事件发生状态(ECB指针)
* OS_EventTO()          由于超时而将任务置为就绪态(ECB指针)
* OS_EventWaitListInit()  事件控制块列表初始化(事件控制块指针)
* OS_FlagInit()         初始化事件标志结构
* OS_FlagUnlink()       把这个OS_FLAG_NODE从事件标志组的等待任务链表中删除(OS_FLAG_NODE *pnode)
* OS_MemInit()          初始化内存分区
* OS_QInit()            初始化事件队列结构
```

```
* OS_Sched()           任务调度函数
* OS_TaskIdle()       空闲任务函数(指向一个数据结构)
* OS_TaskStat()       统计任务(指向一个数据结构)
* OS_TCBInit()        初始化任务控制块TCB(优先级指针、栈顶指针、栈底指针、任务标志符、
 * 堆栈容量、扩展指针、选择项)
*********************************************************************************************
*/
```