第 12 章 *配置手册*

本章将介绍µC/OS-II 中的初始化配置项。由于µC/OS-II 向用户提供源代码，初始化配置项由一系列#define constant 语句构成，都在文件 OS_CFG.H 中。用户的工程文件组中都应该包含这个文件。

本节介绍每个用#define constant 定义的常量，介绍的顺序和它们在 OS_CFG.H 中出现的顺序是相同的。表 12.1 列出了常量控制的µC/OS-II 函数。“类型”为函数所属的类型，“置 1”表示当定义常量为 1 时可以打开相应的函数，“其他常量”为与这个函数有关的其他控制常量。

注意编译工程文件时要包含 OS_CFG.H，使定义的常量生效。

表 T12.1 µC/OS-II 函数和相关的常量（#define constant 定义）

表 T12.1 µC/OS-II 函数和相关常量

类型	*置1*	*其他常量*
杂相		
OSInit()	无	OS_MAX_EVENTS OS_Q_EN and OS_MAX_QS OS_MEM_EN OS_TASK_IDLE_STK_SIZE OS_TASK_STAT_EN OS_TASK_STAT_STK_SIZE
OSSchedLock()	无	无
OSSchedUnlock()	无	无
OSStart()	无	无
OSStatInit()	OS_TASK_STAT_EN && OS_TASK_CREATE_EXT_EN	OS_TICKS_PER_SEC
OSVersion()	无	无
中断处理		
OSIntEnter()	无	无
OSIntExit()	无	无
消息邮箱		
OSMboxAccept()	OS_MBOX_EN	无
OSMboxCreate()	OS_MBOX_EN	OS_MAX_EVENTS
OSMboxPend()	OS_MBOX_EN	无
OSMboxPost()	OS_MBOX_EN	无
OSMboxQuery()	OS_MBOX_EN	无
内存块管理		

OSMemCreate()	OS_MEM_EN	OS_MAX_MEM_PART
OSMemGet()	OS_MEM_EN	无
OSMemPut()	OS_MEM_EN	无
OSMemQuery()	OS_MEM_EN	无
消息队列		
OSQAccept()	OS_Q_EN	无
OSQCreate()	OS_Q_EN	OS_MAX_EVENTS OS_MAX_QS
OSQFlush()	OS_Q_EN	无
OSQPend()	OS_Q_EN	无
OSQPost()	OS_Q_EN	无
OSQPostFront()	OS_Q_EN	无
OSQQuery()	OS_Q_EN	无
信号量管理		
OSSemAccept()	OS_SEM_EN	无
OSSemCreate()	OS_SEM_EN	OS_MAX_EVENTS
OSSemPend()	OS_SEM_EN	无
OSSemPost()	OS_SEM_EN	无
OSSemQuery()	OS_SEM_EN	无
任务管理		
OSTaskChangePrio()	OS_TASK_CHANGE_PRIO_EN	OS_LOWEST_PRIO
OSTaskCreate()	OS_TASK_CREATE_EN	OS_MAX_TASKS OS_LOWEST_PRIO
OSTaskCreateExt()	OS_TASK_CREATE_EXT_EN	OS_MAX_TASKS OS_STK_GROWTH OS_LOWEST_PRIO
OSTaskDel()	OS_TASK_DEL_EN	OS_LOWEST_PRIO
OSTaskDelReq()	OS_TASK_DEL_EN	OS_LOWEST_PRIO
OSTaskResume()	OS_TASK_SUSPEND_EN	OS_LOWEST_PRIO
OSTaskStkChk()	OS_TASK_CREATE_EXT_EN	OS_LOWEST_PRIO
OSTaskSuspend()	OS_TASK_SUSPEND_EN	OS_LOWEST_PRIO
OSTaskQuery()		OS_LOWEST_PRIO
时钟管理		
OSTimeDly()	无	无
OSTimeDlyHMSM()	无	OS_TICKS_PER_SEC
OSTimeDlyResume()	无	OS_LOWEST_PRIO
OSTimeGet()	无	无
OSTimeSet()	无	无

OSTimeTick()	无	无
用户定义函数		
OSTaskCreateHook()	OS_CPU_HOOKS_EN	无
OSTaskDelHook()	OS_CPU_HOOKS_EN	无
OSTaskStatHook()	OS_CPU_HOOKS_EN	无
OSTaskSwHook()	OS_CPU_HOOKS_EN	无
OSTimeTickHook()	OS_CPU_HOOKS_EN	无

OS_MAX_EVENTS

OS_MAX_EVENTS 定义系统中最大的事件控制块的数量。系统中的每一个消息邮箱，消息队列，信号量都需要一个事件控制块。例如，系统中有 10 个消息邮箱，5 个消息队列，3 个信号量，则 OS_MAX_EVENTS 最小应该为 18。只要程序中用到了消息邮箱，消息队列或是信号量，

则 OS_MAX_EVENTS 最小应该设置为 2。

OS_MAX_MEM_PARTS

OS_MAX_MEM_PARTS 定义系统中最大的内存块数，内存块将由内存管理函数操作（定义在文件 OS_MEM.C 中）。如果要使用内存块，OS_MAX_MEM_PARTS 最小应该设置为 2，常量 OS_MEM_EN 也要同时置 1。

OS_MAX_QS

OS_MAX_QS 定义系统中最大的消息队列数。要使用消息队列，常量 OS_Q_EN 也要同时置 1。如果要使用消息队列，OS_MAX_ QS 最小应该设置为 2。

OS_MAX_TASKS

OS_MAX_MEM_TASKS 定义用户程序中最大的任务数。OS_MAX_MEM_TASKS 不能大于 62，这是由于μC/OS-II 保留了两个系统使用的任务。如果设定 OS_MAX_MEM_TASKS 刚好等于所需任务数，则建立新任务时要注意检查是否超过限定。而 OS_MAX_MEM_TASKS 设定的太大则会浪费内存。

OS_LOWEST_PRIO

OS_LOWEST_PRIO 设定系统中的任务最低优先级（最大优先级数）。设定 OS_LOWEST_PRIO 可以节省用于任务控制块的内存。μC/OS-II 中优先级数从 0（最高优先级）到 63（最低优先级）。设定 OS_LOWEST_PRIO 小于 63 意味着不会建立优先级数大于 OS_LOWEST_PRIO 的任务。μC/OS-II 中保留两个优先级系统自用：OS_LOWEST_PRIO 和 OS_LOWEST_PRIO-1。其中 OS_LOWEST_PRIO 留给系统的空闲任务（Idle task）(OSTaskIdle（）)。OS_LOWEST_PRIO-1 留给统计任务（OSTaskStat（））。用户任务的优先级可以从 0 到 OS_LOWEST_PRIO-2。OS_LOWEST_PRIO 和 OS_MAX_TASKS 之间没有什么关系。例如，可以设 OS_MAX_TASKS 为 10 而

OS_LOWEST_PRIO 为 32。此时系统最多可有 10 个任务，用户任务的优先级可以是 0 到 30。当然，OS_LOWEST_PRIO 设定的优先级也要够用，例如设 OS_MAX_TASKS 为 20，而 OS_LOWEST_PRIO 为 10，优先级就不够用了。

OS_TASK_IDLE_STK_SIZE

OS_TASK_IDLE_STK_SIZE 设置µC/OS-II 中空闲任务（Idle task）堆栈的容量。注意堆栈容量的单位不是字节，而是 OS_STK（µC/OS-II 中堆栈统一用 OS_STK 声明，根据不同的硬件环境，OS_STK 可为不同的长度----译者注）。空闲任务堆栈的容量取决于所使用的处理器，以及预期的最大中断嵌套数。虽然空闲任务几乎不做什么工作，但还是要预留足够的堆栈空间保存 CPU 寄存器的内容，以及可能出现的中断嵌套情况。

OS_TASK_STAT_EN

OS_TASK_STAT_EN 设定系统是否使用µC/OS-II 中的统计任务（statistic task）及其初始化函数。如果设为 1，则使用统计任务 OSTaskStat（）。统计任务每秒运行一次，计算当前系统 CPU 使用率，结果保存在 8 位变量 OSCPUUsage 中。每次运行，OSTaskStat（）都将调用 OSTaskStatHook（）函数，用户自定义的统计功能可以放在这个函数中。详细情况请参考 OS_CORE.C 文件。统计任务 OSTaskStat（）的优先级总是设为 OS_LOWEST_PRIO-1。
当 OS_TASK_STAT_EN 设为 0 的时候，全局变量 OSCPUUsage，OSIdleCtrMax，OSIdleCtrRun 和 OSStatRdy 都不声明，以节省内存空间。

OS_TASK_STAT_STK_SIZE

OS_TASK_STAT_STK_SIZE 设置µC/OS-II 中统计任务（statistic task）堆栈的容量。注意单位不是字节，而是 OS_STK（µC/OS-II 中堆栈统一用 OS_STK 声明，根据不同的硬件环境，OS_STK 可为不同的长度----译者注）。统计任务堆栈的容量取决于所使用的处理器类型，以及如下的操作：

- 进行 32 位算术运算所需的堆栈空间。
- 调用 OSTimeDly（）所需的堆栈空间。
- 调用 OSTaskStatHook（）所需的堆栈空间。
- 预计最大的中断嵌套数。

如果想在统计任务中进行堆栈检查，判断实际的堆栈使用，用户需要设 OS_TASK_CREATE_EXT_EN 为 1，并使用 OSTaskCreateExt（）函数建立任务。

OS_CPU_HOOKS_EN

此常量设定是否在文件 OS_CPU_C.C 中声明对外接口函数（hook function），设为 1 为声明。µC/OS-II 中提供了 5 个对外接口函数，可以在文件 OS_CPU_C.C 中声明，也可以在用户自己的

代码中声明：

- OSTaskCreateHook（）
- OSTaskDelHook（）
- OSTaskStatHook（）
- OSTaskSwHook（）
- OSTimeTickHook（）

OS_MBOX_EN

OS_MBOX_EN 控制是否使用μC/OS-II 中的消息邮箱函数及其相关数据结构，设为 1 为使用。如果不使用，则关闭此常量节省内存。

OS_MEM_EN

OS_MEM_EN 控制是否使用μC/OS-II 中的内存块管理函数及其相关数据结构，设为 1 为使用。如果不使用，则关闭此常量节省内存。

OS_Q_EN

OS_Q_EN 控制是否使用μC/OS-II 中的消息队列函数及其相关数据结构，设为 1 为使用。如果不使用，则关闭此常量节省内存。如果 OS_Q_EN 设为 0，则语句#define constant OS_MAX_QS 无效。

OS_SEM_EN

OS_SEM_EN 控制是否使用μC/OS-II 中的信号量管理函数及其相关数据结构，设为 1 为使用。如果不使用，则关闭此常量节省内存。

OS_TASK_CHANGE_PRIO_EN

此常量控制是否使用μC/OS-II 中的 OSTaskChangePrio（）函数，设为 1 为使用。如果在应用程序中不需要改变运行任务的优先级，则将此常量设为 0 节省内存。

OS_TASK_CREATE_EN

此常量控制是否使用μC/OS-II 中的 OSTaskCreate（）函数，设为 1 为使用。在μC/OS-II 中推荐用户使用 OSTaskCreateExt（）函数建立任务。如果不使用 OSTaskCreate（）函数，将

OS_TASK_CREATE_EN 设为 0 可以节省内存。注意 OS_TASK_CREATE_EN 和 OS_TASK_CREATE_EXT_EN 至少有一个要为 1，当然如果都使用也可以。

OS_TASK_CREATE_EXT_EN

此常量控制是否使用µC/OS-II 中的 OSTaskCreateExt（）函数，设为 1 为使用。该函数为扩展的，功能更全的任务建立函数。如果不使用该函数，将 OS_TASK_CREATE_EXT_EN 设为 0 可以节省内存。注意，如果要使用堆栈检查函数 OSTaskStkChk（），则必须用 OSTaskCreateExt（）建立任务。

OS_TASK_DEL_EN

此常量控制是否使用µC/OS-II 中的 OSTaskDel（）函数，设为 1 为使用。如果在应用程序中不使用删除任务函数，将 OS_TASK_DEL_EN 设为 0 可以节省内存。

OS_TASK_SUSPEND_EN

此常量控制是否使用µC/OS-II 中的 OSTaskSuspend（）和 OSTaskResume（）函数，设为 1 为使用。如果在应用程序中不使用任务挂起-唤醒函数，将 OS_TASK_SUSPEND_EN 设为 0 可以节省内存。

OS_TICKS_PER_SEC

此常量标识调用 OSTimeTick（）函数的频率。用户需要在自己的初始化程序中保证 OSTimeTick（）按所设定的频率调用（即系统硬件定时器中断发生的频率----译者注）。在函数 OSStatInit（），OSTaskStat（）和 OSTimeDlyHMSM（）中都会用到 OS_TICKS_PER_SEC。