**目录**

[1. 总体描述 3](#_Toc59226195)

[1.1 信号量 3](#_Toc59226196)

[1.2 互斥锁 3](#_Toc59226197)

[2. 要点 4](#_Toc59226198)

[3. 代码文件结构 4](#_Toc59226199)

[4. 函数列表、功能及函数关系描述 4](#_Toc59226200)

[4.1函数列表和功能 4](#_Toc59226201)

[4.1.1 os\_sem.c 4](#_Toc59226202)

[4.1.2 os\_mutex.c 5](#_Toc59226203)

[4.2函数关系描述 6](#_Toc59226204)

[5. 主要数据结构 6](#_Toc59226205)

[5.1 OS\_EVENT 6](#_Toc59226206)

[5.2 OS\_SEM\_DATA 7](#_Toc59226207)

[5.3 OS\_MUTEX\_DATA 7](#_Toc59226208)

[6. 全局常量、变量描述 7](#_Toc59226209)

[6.1与是否启用某函数相关的常量 7](#_Toc59226210)

[6.2与事件类型相关的常量 8](#_Toc59226211)

[6.3与错误标志相关的常量 8](#_Toc59226212)

[6.3与任务状态相关的常量 9](#_Toc59226213)

[6.4其他常量 9](#_Toc59226214)

[6.4全局变量 9](#_Toc59226215)

[7. 各函数功能和流程图 10](#_Toc59226216)

[7.1 OSSemAccept() 10](#_Toc59226217)

[7.1.1 概述 10](#_Toc59226218)

[7.1.2 流程图 10](#_Toc59226219)

[7.2 OSSemCreate() 11](#_Toc59226220)

[7.2.1 概述 11](#_Toc59226221)

[7.2.2 流程图 12](#_Toc59226222)

[7.3 OSSemDel() 12](#_Toc59226223)

[7.3.1 概述 12](#_Toc59226224)

[7.3.2 流程图 14](#_Toc59226225)

[7.4 OSSemPend() 15](#_Toc59226226)

[7.4.1 概述 15](#_Toc59226227)

[7.4.2 流程图 16](#_Toc59226228)

[7.5 OSSemPendAbort() 17](#_Toc59226229)

[7.6 OSSemPost() 19](#_Toc59226230)

[7.6.1概述 19](#_Toc59226231)

[7.6.2流程图 20](#_Toc59226232)

[7.7 OSSemQuery() 20](#_Toc59226233)

[7.8 OSSemSet() 23](#_Toc59226234)

[7.8.1 概述 23](#_Toc59226235)

[7.8.2 流程图 24](#_Toc59226236)

[7.9 OSMutexAccept() 25](#_Toc59226237)

[7.9.1概述 25](#_Toc59226238)

[7.9.2流程图 26](#_Toc59226239)

[7.10 OSMutexCreate() 27](#_Toc59226240)

[7.11 OSMutexDel() 29](#_Toc59226241)

[7.12 OSMutexPend() 29](#_Toc59226242)

[7.12.1概述 29](#_Toc59226243)

[7.12.2流程图 30](#_Toc59226244)

[7.13 OSMutexPost() 32](#_Toc59226245)

[7.13.1概述 32](#_Toc59226246)

[7.13.2流程图 33](#_Toc59226247)

[7.14 OSMutexQuery() 35](#_Toc59226248)

[7.14.1概述 35](#_Toc59226249)

[7.14.2流程图 36](#_Toc59226250)

[7.15 OSMutex\_RdyAtPrio() 37](#_Toc59226251)

[8. 分工与贡献率表格 38](#_Toc59226252)

[8.1函数分工 38](#_Toc59226253)

[8.2其他分工 39](#_Toc59226254)

[8.3贡献率表格 39](#_Toc59226255)

[9. 附录：加上注释的源代码 39](#_Toc59226256)

**uC/OSⅡ源码报告—信号量与互斥锁**

# 总体描述

## 1.1 信号量

信号量（Semaphore）是一种实现任务间通信的机制，可以实现任务之间同步或临界资源的互斥访问，常用于协助一组相互竞争的任务来访问临界资源。在多任务系统中，各任务之间需要同步或互斥实现临界资源的保护，信号量功能可以为用户提供这方面的支持。

抽象的来讲，信号量是一个非负整数，所有获取它的任务都会将该整数减一（获取它当然是为了使用资源），当该整数值为零时，所有试图获取它的任务都将处于阻塞状态。通常一个信号量的计数值用于对应有效的资源数，表示剩下可被占用的临界资源数，其值的含义分两种情况：

1. 0：表示没有积累下来的释放信号量操作，且有可能有在此信号量上阻塞的任务。
2. 正值，表示有一个或多个释放信号量操作。

## 1.2 互斥锁

互斥量又称互斥信号量（本质也是一种信号量，不具备传递数据功能），是一种特殊的二值信号量，它和信号量不同的是，它支持互斥量所有权、递归访问以及防止优先级翻转的特性，用于实现对临界资源的独占式处理。任意时刻互斥量的状态只有两种，开锁或闭锁。当互斥量被任务持有时，该互斥量处于闭锁状态，这个任务获得互斥量的所有权。当该任务释放这个互斥量时，该互斥量处于开锁状态，任务失去该互斥量的所有权。

互斥锁与二值信号量有许多共同点，二者都可以保护临界资源，保证多任务的访问不发生冲突。但是信号量会导致一个潜在问题：优先级反转。而uC/OS提供的互斥锁可以通过优先级继承机制降低优先级反转产生的影响。

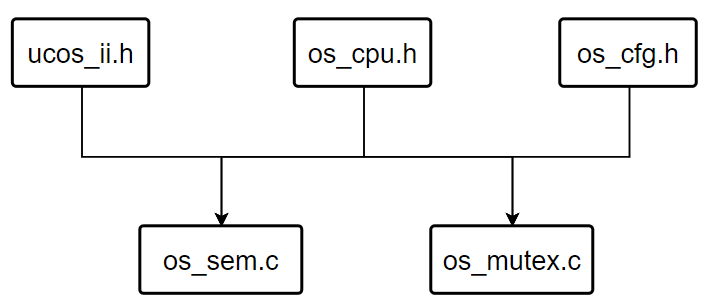
# 要点

* 优先级反转问题：

当一个低优先级task占有了某个资源，此时一个高优先级task抢占了CPU。这个高优先级task也需要用这个资源，但是资源已被低优先级task占有。高优先级task不得不等待低优先级task先运行完毕。

* 优先级继承机制：
  + 当一个低优先级task占有了某个资源，此时一个高优先级task抢占了CPU。这个高优先级task也需要用这个资源。高优先级task提升低优先级task的优先级，让它能获得CPU尽快运行及时释放资源。资源被释放后低优先级task的优先级恢复到原来的优先级。优先级继承机制被用来缓解优先级反转问题。
  + Task在申请互斥锁时，uC/OSⅡ可以使用优先级继承机制（在创建互斥锁时决定是否启用）来缓解优先级反转问题。

# 代码文件结构



* **ucos\_ii.h:** 头文件集合，其他.c文件几乎都会包括该头文件。定义ucos\_ii系统中使用到的各种类型、变量等。
* **os\_cpu.h:** 与处理器相关的头文件
* **os\_cfg.h:** 与系统配置相关的文件。启用/关闭系统的功能，如果要对系统进行裁剪和配置则需要修改这个文件。
* **os\_sem.c:** 信号量管理以及相关函数实现的文件。
* **os\_mutex.c:** 互斥信号量管理相关函数的实现文件。

# 函数列表、功能及函数关系描述

## 4.1 os\_sem.c

文件代码行数：736

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名 | 组别 | 功能描述 | 代码行数 |
| OSSemAccept() | 1 | 这个函数检查信号量，看看是否有资源可用，或者是否发生了事件。与OSSemPend()不同，如果资源不可用或事件未发生，OSSemAccept()不会挂起调用任务。 | 32 |
| OSSemCreate() | 1 | 创建一个信号量。 | 43 |
| OSSemDel() | 2 | 这个函数删除一个信号量，并准备信号量上所有挂起的任务。 | 112 |
| OSSemPend() | 3 | 等待一个信号量。 | 87 |
| OSSemPendAbort() | 4 | 这个函数中止并准备当前等待信号量的所有任务。这个函数应该用于故障中止信号量上的等待，而不是通过OSSemPost()正常地发送信号量。 | 68 |
| OSSemPost() | 3 | 这个函数发出信号。 | 39 |
| OSSemQuery() | 4 | This function obtains information about a semaphore.  查询信号量的信息。 | 43 |
| OSSemSet() | 1 | 这个函数将信号量计数设置为参数指定的值。  通常，这个值为0。 | 56 |

## 4.2 os\_mutex.c

文件代码行数：919

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名 | 组别 | 功能描述 | 代码行数 |
| OSMutexAccept() | 6 | 此函数检查互斥信号量以查看是否有一个互斥信号量资源可用。与OSMutexPend()不同，如果资源不可用或事件未发生，则OSMutexAccept()不会暂停调用任务。 | 68 |
| OSMutexCreate() | 7 | 这个函数创建一个互斥信号量。 | 83 |
| OSMutexDel() | 7 | 这个函数删除一个互斥信号量，并准备它上挂起的所有任务。 | 145 |
| OSMutexPend() | 6 | 此函数等待一个互斥信号量。首先检查互斥信号量以查看是否有一个互斥信号量资源可用。如果资源不可用，考虑持有该信号量的task是否需要进行优先级提升，如果需要，则进行优先级提升。最后进行任务调度，让出CPU，当前任务从运行态变成阻塞态，让占有互斥锁的task有机会先运行。当这个任务重新回到运行态时，检查状态，判断互斥信号量是否申请成功。不管是否成功，结束该次申请事件，并将相关变量重置。 | 183 |
| OSMutexPost() | 5 | 这个函数发出互斥信号量 | 75 |
| OSMutexQuery() | 5 | 这个函数获取互斥量的信息 | 61 |
| OSMutex\_RdyAtPrio() | 7 | 这个函数使任务在指定的优先级就绪 | 29 |

# 主要数据结构

## 5.1 OS\_EVENT

事件控制块的基本数据结构OS\_EVENT，所有的通信信号都被看成是事件(event), uC/OS-II通过事件控制块(ECB)来管理每一个具体事件。结构体成员如下：

|  |  |  |
| --- | --- | --- |
| 类型 | 标识符 | 含义 |
| INT8U | OSEventType | 事件的类型，所有可能的事件类型有OS\_EVENT\_TYPE\_UNUSED，无事件；  OS\_EVENT\_TYPE\_MBOX，消息邮箱事件；  OS\_EVENT\_TYPE\_Q，消息队列事件；  OS\_EVENT\_TYPE\_SEM，信号量事件，  OS\_EVENT\_TYPE\_MUTEX，互斥锁事件；  OS\_EVENT\_TYPE\_FLAG，标志事件（默认被禁用） |
| Void\* | OSEventPtr | 对于互斥锁来说，它指向占有互斥锁的task |
| INT16U | OSEventCnt | 对于信号量来说，是计数值；  对于互斥锁来说，高8位存储了提升的优先级，低8位存储占有互斥锁task的优先级 |
| OS\_PRIO | OSEventGrp | 记录有哪些task在等待当前事件 |
| OS\_PRIO | OSEventTbl[OS\_EVENT\_TBL\_SIZE] | 记录有哪些task在等待当前事件 |
| INT8U\* | OSEventName | 事件名字(默认被禁用) |

## 5.2 OS\_SEM\_DATA

记录信号量信息的数据结构OS\_SEM\_DATA，用于存储信号量查询的结果。结构体成员如下：

|  |  |  |
| --- | --- | --- |
| 类型 | 标识符 | 含义 |
| INT16U | OSCnt | 信号量计数 |
| OS\_PRIO | OSEventTbl[OS\_EVENT\_TBL\_SIZE] | 事件标志，主要存放等待相应信号的任务标志 |
| OS\_PRIO | OSEventGrp | 等待事件发生的任务对应的组 |

## 5.3 OS\_MUTEX\_DATA

记录互斥型信号量信息的数据结构OS\_MUTEX\_DATA，用于存储信号量查询的结果。结构体成员如下：

|  |  |  |
| --- | --- | --- |
| 类型 | 标识符 | 含义 |
| OS\_PRIO | OSEventTbl[OS\_EVENT\_TBL\_SIZE] | 等待事件发生的任务列表 |
| OS\_PRIO | OSEventGrp | 互斥量值（OS\_FALSE表示互斥量已使用，OS\_TRUE表示互斥量可使用） |
| BOOLEAN | OSValue | 等待事件发生的任务对应的组 |
| INT8U | OSOwnerPrio | Mutex所有者的任务优先级，如果没有所有者，则为0xFF |
| INT8U | OSMutexPCP | 优先级上限优先级或0xFF |

# 全局常量、变量描述

## 6.1与是否启用某函数相关的常量

|  |  |  |
| --- | --- | --- |
| 类型 | 标识符 | 作用 |
| 全局常量 | OS\_SEM\_PEND\_ABORT\_EN | 是否启用OSSemPendAbort()函数 |
| 全局常量 | OS\_SEM\_QUERY\_EN | 是否启用OSSemQuery()函数 |
| 全局常量 | OS\_MUTEX\_ACCEPT\_EN | 是否启用OSMutexAccept()函数 |
| 全局常量 | OS\_SEM\_DEL\_EN | 是否启用OSSemDel ()函数 |

## 6.2与事件类型相关的常量

|  |  |  |
| --- | --- | --- |
| 类型 | 标识符 | 作用 |
| 全局常量 | OS\_EVENT\_TYPE\_SEM | 值为3，信号量类型事件。用于判断事件控制块是否是由OSSemCreate() 创建的 |
| 全局常量 | OS\_EVENT\_TYPE\_MUTEX | 值为4，互斥类型事件 |

## 6.3与错误标志相关的常量

|  |  |  |
| --- | --- | --- |
| 类型 | 标识符 | 作用 |
| 全局常量 | OS\_NO\_ERR | 未发生错误 |
| 全局常量 | OS\_ERR\_PEVENT\_NULL | 传入的事件指针pevent是空指针 |
| 全局常量 | OS\_ERR\_PEND\_ISR | 在中断服务程序中调用OSSemPend()函数错误。虽然uCOS规定了不允许在中断中调用OSSemPend()，但ucos仍然包含了检测这种情况的功能； |
| 全局常量 | OS\_ERR\_EVENT\_TYPE | 传入的事件指针pevent不是指向信号量的指针。 |
| 全局常量 | OS\_ERR\_PEND\_LOCKED | 调度器挂起锁定错误 |
| 全局常量 | OS\_ERR\_PEND\_ABORT | 信号量获取被终止 |
| 全局常量 | OS\_ERR\_TIMEOUT | 超时 |
| 全局常量 | OS\_ERR\_INVALID\_OPT | 没有将opt参数定义为2种合法的参数之一 |
| 全局常量 | OS\_ERR\_TASK\_WAITING | 有一个或一个以上的任务在等待信号量; |
| 全局常量 | OS\_ERR\_PRIO\_INVALID | 错误（非法）的优先级 |
| 全局常量 | OS\_ERR\_CREATE\_ISR | 创建的事件处于中断程序里 |
| 全局常量 | OS\_ERR\_PRIO\_EXIST | 优先级对应的指针非空（已存在） |

## 6.3与任务状态相关的常量

|  |  |  |
| --- | --- | --- |
| 类型 | 标识符 | 作用 |
| 全局常量 | OS\_STAT\_SEM | 任务等待信号量状态 |
| 全局常量 | OS\_STAT\_PEND\_OK | 待处理状态正常--未待处理或待完成 |
| 全局常量 | OS\_STAT\_PEND\_ABORT | 获取信号量被终止 |
| 全局常量 | OS\_STAT\_PEND\_TO | 等待超时 |
| 全局常量 | OS\_TCB\_RESERVED | 在创建任务的时候, 用来占用该优先级，但是尚未初始化 |

## 6.4其他常量

|  |  |  |
| --- | --- | --- |
| 类型 | 标识符 | 作用 |
| 全局常量 | OS\_ARG\_CHK\_EN | 启用（1）或禁用（0）参数检查 |
| 全局常量 | OS\_SAFETY\_CRITICAL | 是否启用安全检查 |
| 全局常量 | OS\_CRITICAL\_METHOD | 系统的临界区方法 |
| 全局常量 | OS\_PRIO\_MUTEX\_CEIL\_DIS | 是否启用优先级继承机制 |
| 全局常量 | OS\_MUTEX\_KEEP\_LOWER\_8 | 值为(INT16U)0x00FFu做与运算获取低8位 |
| 全局常量 | OS\_MUTEX\_KEEP\_UPPER\_8 | 值为(INT16U)0xFF00u做与运算获取高8位 |
| 全局常量 | OS\_MUTEX\_AVAILABLE | 互斥锁可用 |
| 全局常量 | OS\_PRIO\_MUTEX\_CEIL\_DIS | 禁用互斥优先级上限提升 |
| 全局常量 | OS\_EVENT\_TBL\_SIZE | 事件表窗口大小 |
| 全局常量 | OS\_RDY\_TBL\_SIZE | 就绪表窗口大小 |
| 全局常量 | OS\_MUTEX\_KEEP\_LOWER\_8 | 互斥量低八位 |
| 全局常量 | OS\_MUTEX\_KEEP\_UPPER\_8 | 互斥量高八位 |
| 全局常量 | OS\_MUTEX\_AVAILABLE | 互斥量可使用 |

## 6.4全局变量

|  |  |  |
| --- | --- | --- |
| 类型 | 标识符 | 作用 |
| 全局变量 | OSIntNesting | 中断嵌套级别 |
| 全局常量 | OSLockNesting | 调度器是否加锁 |
| 全局变量 | OSTCBCur | 指向当前运行的任务控制块的指针 |
| 全局变量 | OSTCBPrioTbl[OS\_LOWEST\_PRIO + 1u] | 任务控制块优先级表，可以通过任务优先级号快速找到当前任务在任务控制块中的首地址 |

# 各函数功能和流程图

## 7.1 OSSemAccept()

### 7.1.1 概述

* 函数头：

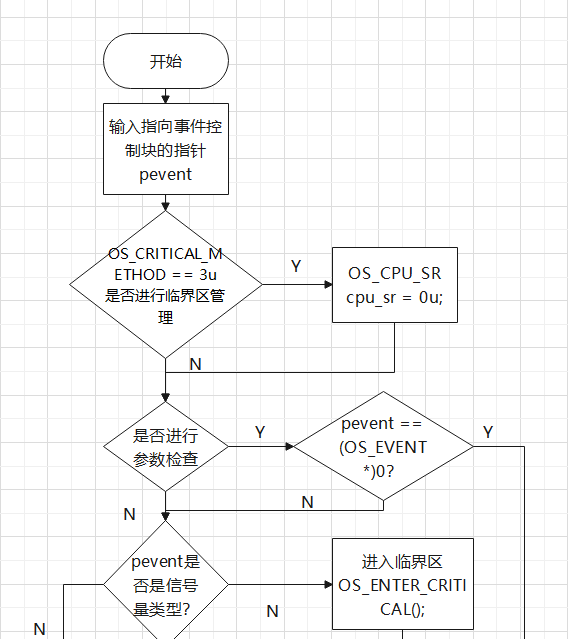
INT16U OSSemAccept（OS\_EVENT\*pevent）；

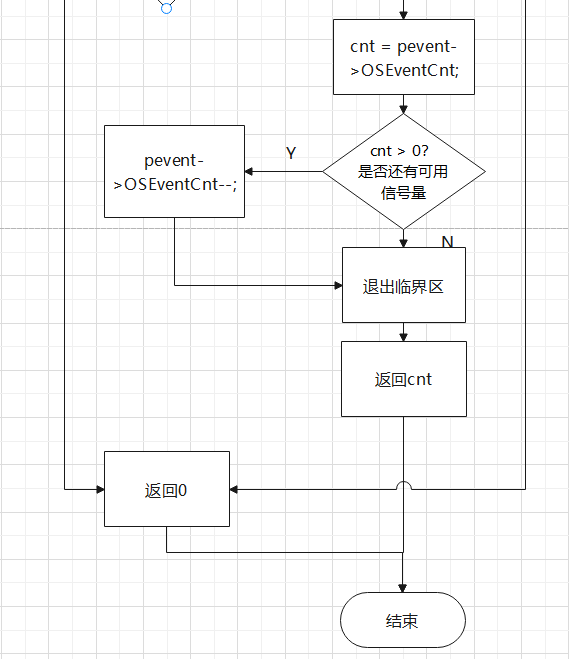
* 函数功能：

根据输入的信号量，检查信号量对应的资源是否可用或对应的事件是否发生，进行申请使用，但不会挂起事件。如果资源不可用/事件未发生/pevent是空指针则返回0，否则返回值大于0.

* 参数：
  + pevent: 一个指向事件控制块的指针.

### 7.1.2 流程图





## 7.2 OSSemCreate()

### 7.2.1 概述

* 函数头：

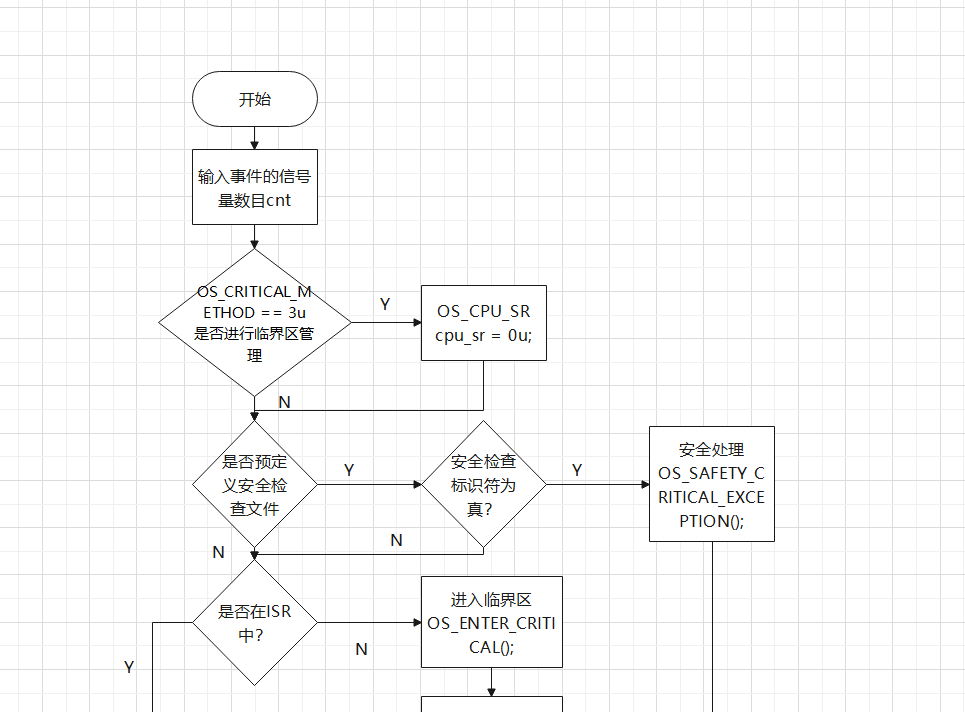
OS\_EVENT  \*OSSemCreate (INT16U cnt)

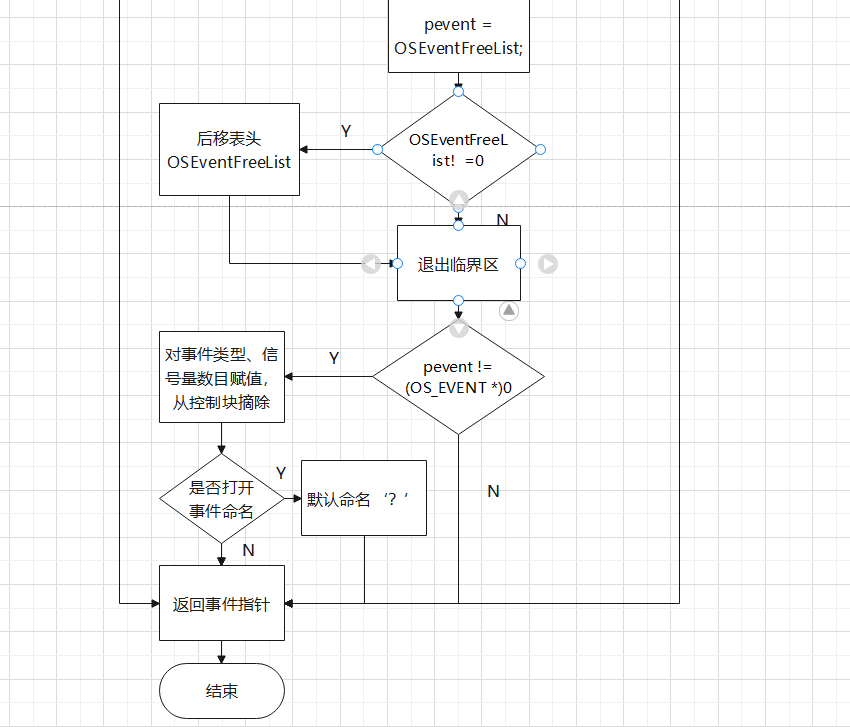
* 函数功能：

根据输入的信号量数目上限cnt创建一个无人等待、无人使用的事件，返回该事件的指针。如果该函数在ISR（中断程序）中被调用，或事件控制块链表表头为空，则返回空，创建失败。如果创建成功，从事件控制块摘除该事件，默认事件名为“？”（如果命名开启）。

* 参数：
  + cnt: 创建的事件的信号量数目.

### 7.2.2 流程图





## 7.3 OSSemDel()

### 7.3.1 概述

* 函数头：

void OS\_EVENT\* OSSemDel(OS\_EVENT\* pevent,INT8U opt,INT8U\* perr)

* 函数功能：

用于删除一个信号量。使用本函数有风险，因为多任务中的其它任务可能还想使用这个信号量，必须特别小心。一般而言，在删除信号量之前，应该先处理所有可能会用到这个信号量的任务。

* 参数：
  + pevent: 指向与所需信号量关联的事件控制块的指针

如果信号量已被删除，返回空指针;

若信号量没有删除，则返回pevent(信号量指针)。

* + opt: 该选项定义信号量的删除条件，可能有以下两种取值

OS\_DEL\_NO\_PEND 只能在已经没有任何任务在等待该信号量时，才能删除该信号量

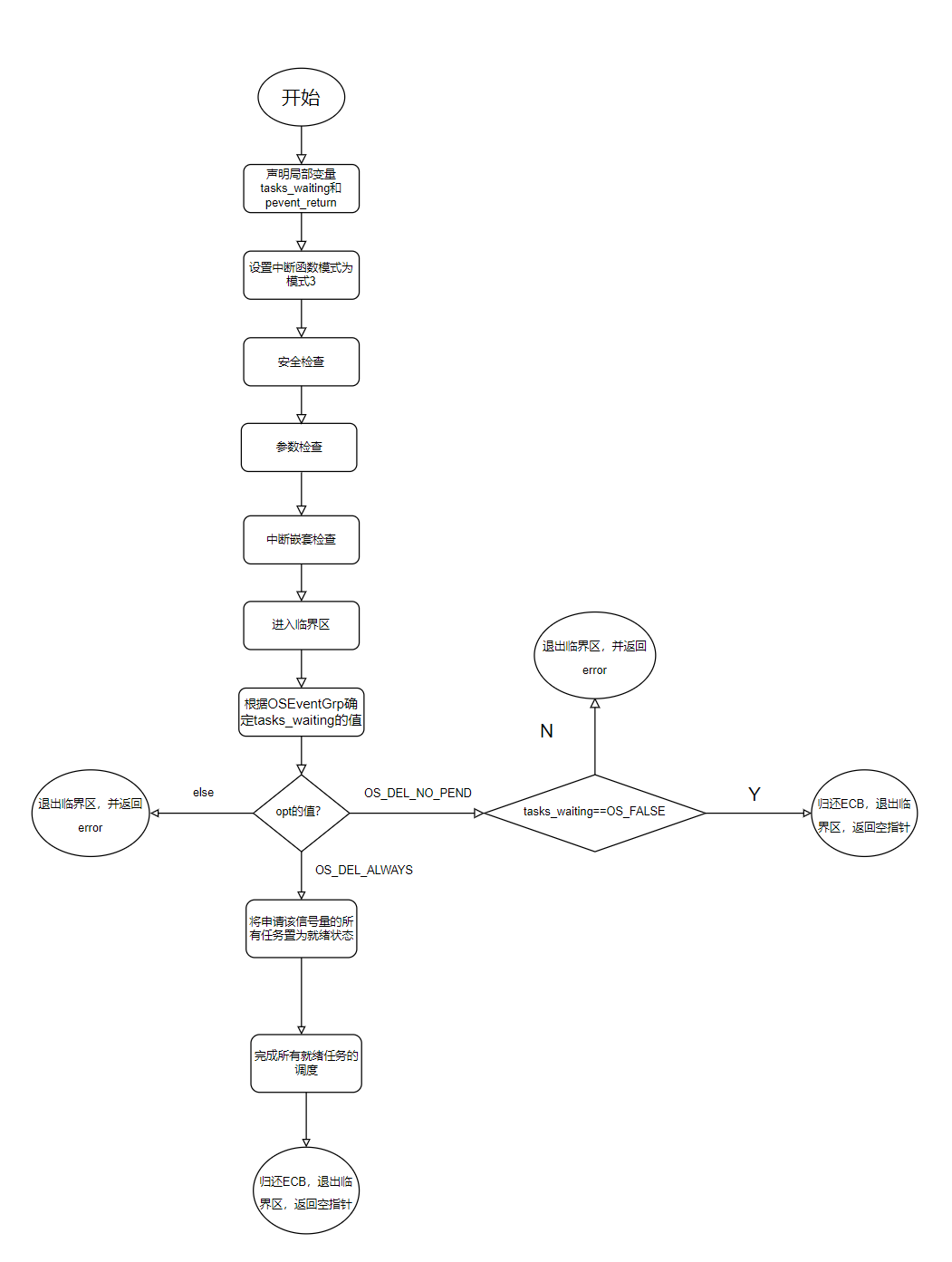
OS\_DEL\_ALWAYS 不管有没有任务在等待该信号量，立即删除该信号量，在这种情况下，所有等待该信号量的任务都立即进入就绪状态

* + perr: 指向错误消息存放位置的指针。
* OS\_ERR\_NONE：调用成功，信号量已被删除;
* OS\_ERR\_DEL\_ISR：试图在中断服务子程序中删除信号量;
* OS\_ERR\_INVALID\_OPT：没有将opt参数定义为2种合法的参数之一;
* OS\_ERR\_TASK\_WAITING：有一个或一个以上的任务在等待信号量;
* OS\_ERR\_EVENT\_TYPE ：pevent不是指向信号量的指针;
* OS\_ERR\_PEVENT\_NULL：pevent指针为空
* 注意事项：

1)使用此函数必须特别小心，因为多任务中的其它任务可能还想使用这个信号量;

2)当挂起的任务进入就绪状态时，中断是关闭的,这意味着中断延迟时间与等待信号量的任务数有关。

### 7.3.2 流程图



* **函数流程分析：**

对OSSemDel函数的流程分析只需抓住它的**三个参数**(OS\_EVENT\* pevent,INT8U opt,INT8U\* perr)即可，明白什么情况下会传出什么样的参数，就了解了这一函数的全貌，在我的理解里，这个函数又可以分成**三个部分**去理解。

* **第一部分 函数的预处理部分**

这部分包括局部变量的声明，中断函数模式的设置，安全检查；只有在安全检查部分存在函数出口，没通过安全检查就直接返回空指针。

* **第二部分 函数的检查部分**

这部分包括参数检查、中断嵌套检查；没通过参数检查存在两种可能，一是pevent指针为空，此时返回OS\_ERR\_EVENT\_TYPE，二是pevent指向的事件类型不是信号量，此时返回OS\_ERR\_EVENT\_TYPE，而没通过中断嵌套检查，就返回OS\_ERR\_DEL\_ISR。

* **第三部分 函数的判断处理部分**

这部分从函数进入临界区开始，到函数的结束为止。这部分做的第一件事是根据OSEventGrp确定tasks\_waiting的值，即有无等待该信号量的任务。随后根据opt的值，可以分成三种情况，一是没有将opt参数定义为2种合法的参数之一，此时返回OS\_ERR\_INVALID\_OPT；二是opt为OS\_DEL\_NO\_PEND，然后根据tasks\_waiting的值又可以分成两种情况，tasks\_waiting为FLASE，则正常删除该信号量，归还ECB，退出临界区，并返回空指针和OS\_ERR\_NONE，如果tasks\_waiting为TRUE，则返回OS\_ERR\_TASK\_WAITING；三是opt为OS\_DEL\_ALWAYS，就直接将申请该信号量的所有任务置为就绪状态并依次完成调度，最后也是正常删除该信号量，归还ECB，退出临界区，并返回空指针和OS\_ERR\_NONE。

## 7.4 OSSemPend()

### 7.4.1 概述

* 函数头：

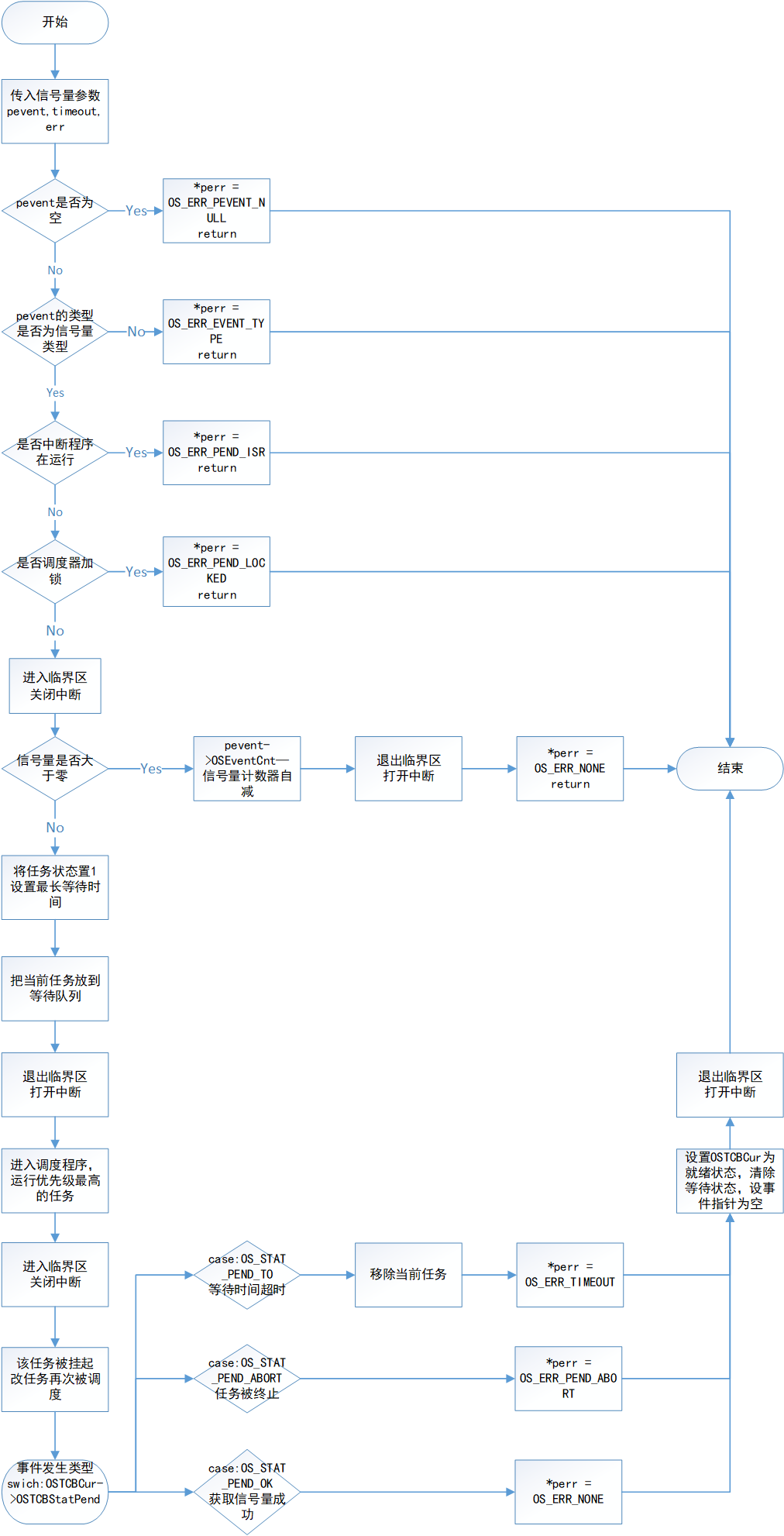
void OSSemPend (OS\_EVENT \*pevent, INT32U timeout, INT8U \*perr);

* 函数功能：

任务等待信号量函数。任务试图获取共享资源使用权、任务需要与其它任务或中断同步及任务需要等待特定事件的发生的场合。若任务调用该函数，且信号量的值>0，那么递减信号量的值；若任务调用该函数，且信号量的值＝0，那么将任务加入该信号量的等待队列中。

* 参数：
  + pevent: 指向与所需信号量关联的事件控制块的指针.
  + timeout: 任务请求信号量等待的最长时间。
  + perr: 指向错误消息存放位置的指针。

### 7.4.2 流程图



#### 7.4.3 改进建议

建议在OS\_PEVENT中增添一个num字段用于记录信号量初始大小，以标记该资源最多能被多少个进程使用，而不是判断<65535,另外也可以用此字段标记二值信号量和计数信号量。

## 7.5 OSSemPendAbort()

#### 7.5.1 概述

* 函数头：

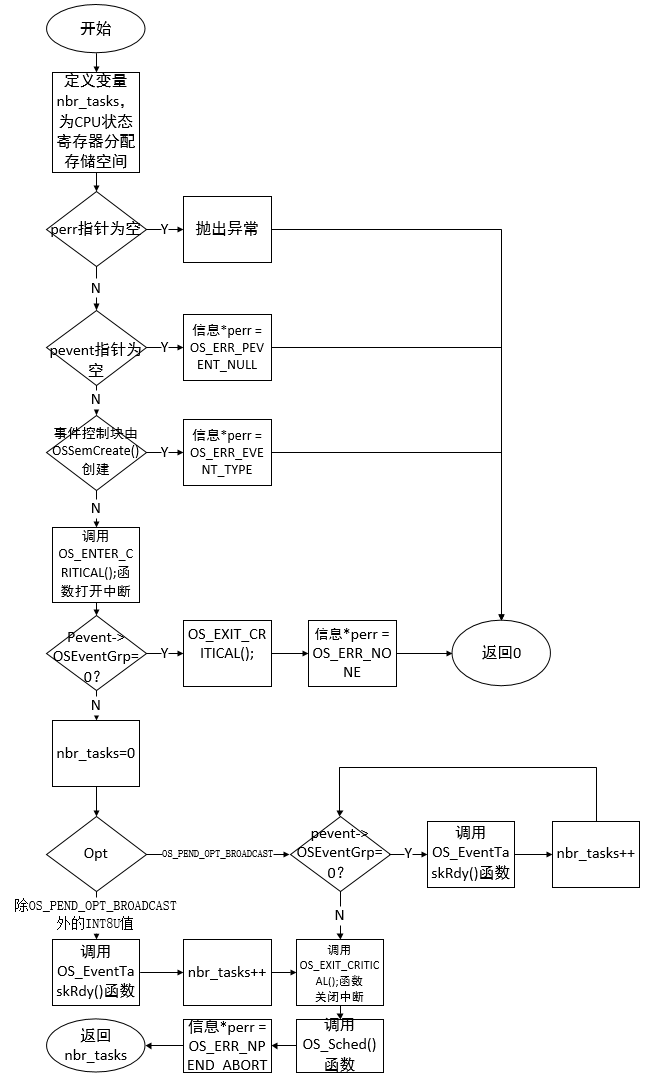
INT8U OSSemPendAbort (OS\_EVENT \*pevent, INT8U opt, INT8U \*perr);

* 函数功能：

中止任务对信号量的等待，使任务进入ready状态。一般在极为特殊的情况下（如要删除一个任务，而这个任务当前在等待信号量时），才使用该函数。

* 参数：
  + pevent: 是指向与所需信号量关联的事件控制块的指针。
  + opt: 决定了中断的类型，可取如下参数：
    - OS\_PEND\_OPT\_NONE：对正在等待信号量的单个任务(最高优先级的任务HPT)，中断等待。
    - OS\_PEND\_OPT\_BROADCAST：对正在等待信号量的所有任务，中断等待。
  + perr: 是指向错误消息存放位置的指针。可能的错误消息是：
    - OS\_ERR\_NONE：没有正在等待信号量的任务。
    - OS\_ERR\_PEND\_ABORT：至少已准备好一个等待信号量的任务，并通知其已中止的等待。 检查返回值以等待其信号量中止的任务数。
    - OS\_ERR\_EVENT\_TYPE：如果没有传递指向信号量的指针。（UCOSII中一共有4种同步方式。(信号量、互斥信号量、消息邮箱、消息队列）
    - OS\_ERR\_PEVENT\_NULL：如果'pevent'是空指针。
* 返回值nbr\_tasks：如果没有任务正在等待信号量或发生错误，则返回值为0；如果现在有一个或多个任务正在等待信号量，已准备好并被通知中断， 则返回值> 0，为中断等待的任务数

#### 7.5.2 流程图



#### 7.5.3 改进建议

由上图可知，当若pevent->OSEventGrp =0时，即没有等待信号量的任务，无需中断等待，而图中额外执行了OS\_ENTER\_CRITICAL和OS\_EXIT\_CRITICAL函数分别用例打开和关闭中断，这些冗余的操作无疑增加了CPU状态寄存器开销，因此可以将打开中断的函数OS\_ENTER\_CRITICAL放在判断pevent->OSEventGrp =0之后，能够减少开销，提高执行效率。

## 7.6 OSSemPost()

### 7.6.1概述

* 函数头：

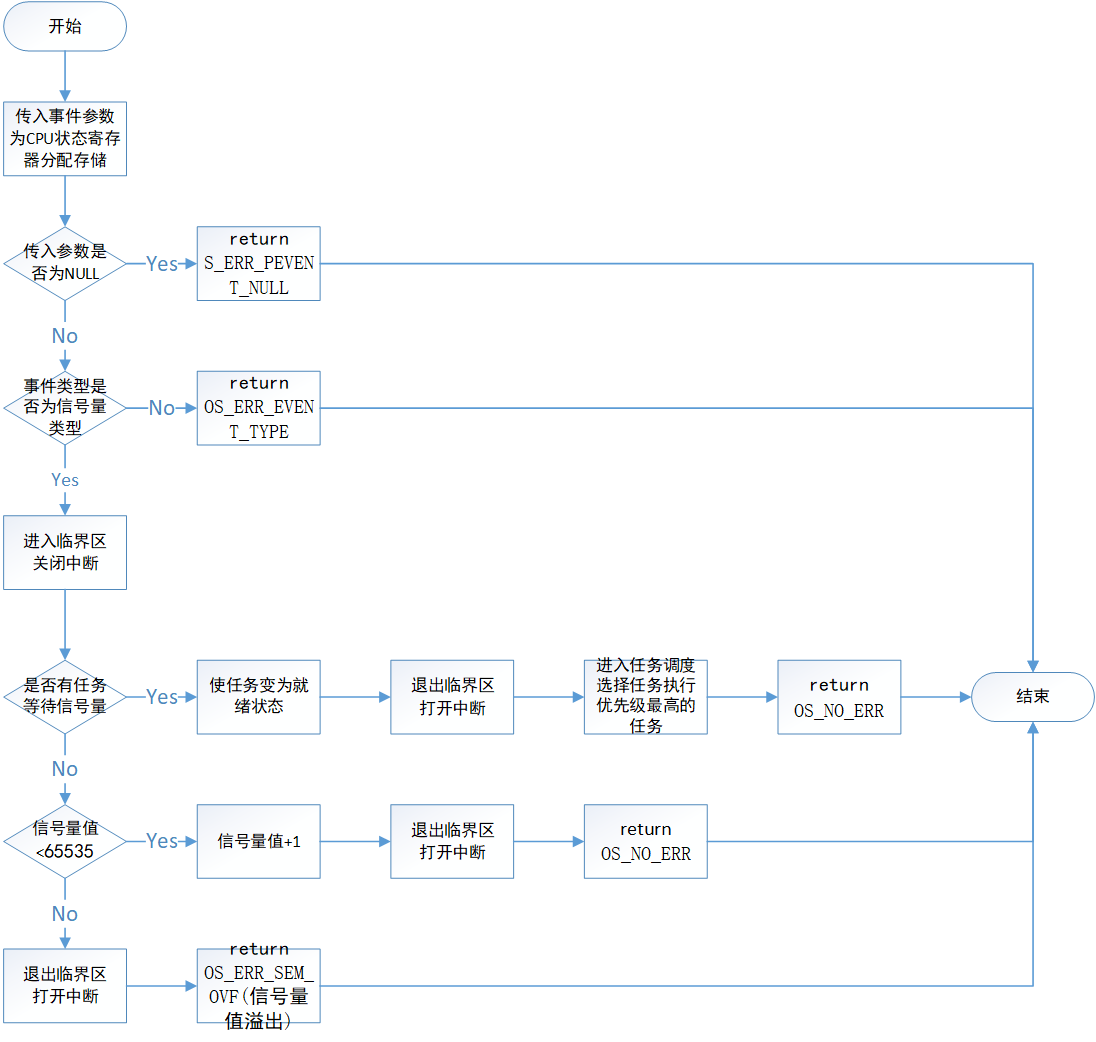
INT8U OSSemPost (OS\_EVENT \*pevent)

* 函数功能：

任务释放信号量函数。置位指定的信号量。如果指定的可用信号量未满，该函数则递增该信号量并返回一个正确提示。如果有任务在等待信号量，那么该任务进入就绪态，然后调度一个最高优先级任务将得到该信号量并进入任务调度。

* 参数：
  + pevent: 指向与所需信号量关联的事件控制块的指针。

### 7.6.2流程图



## 7.7 OSSemQuery()

#### 7.7.1 概述

* 函数头：

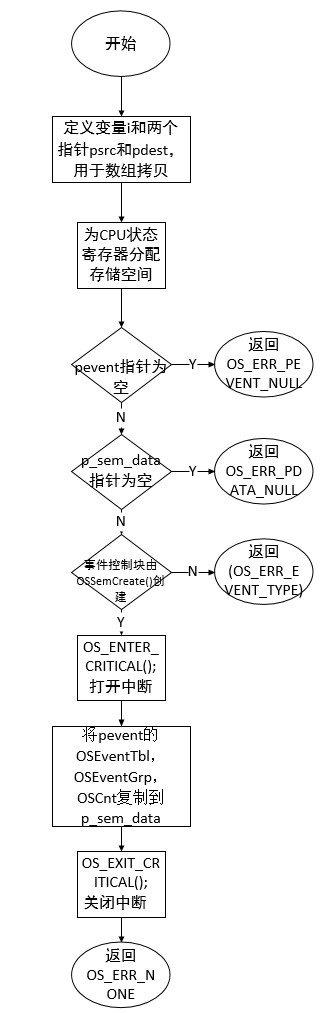
INT8U OSSemQuery (OS\_EVENT \*pevent, OS\_SEM\_DATA \*p\_sem\_data);

* 函数功能：

查询信号量，用于获取有关信号量的信息,将指定信号量的内容拷贝到OS\_SEM\_DATA的数据结构中。

* 参数：
  + pevent: 是指向与所需信号量关联的事件控制块的指针。
  + p\_sem\_data：是指向将包含有关信号量的信息的结构的指针。
* 返回值：
  + OS\_ERR\_NONE：调用成功，消息已发送
  + OS\_ERR\_EVENT \_TYPE：如果您试图从非信号量获取数据。
  + OS\_ERR\_PEVENT\_NULL：如果“pevent”是空指针。
  + OS\_ERR\_PDATA\_NULL：如果“p\_sem\_data”是空指针

#### 7.7.2 流程图



## 7.8 OSSemSet()

### 7.8.1 概述

* 函数头：

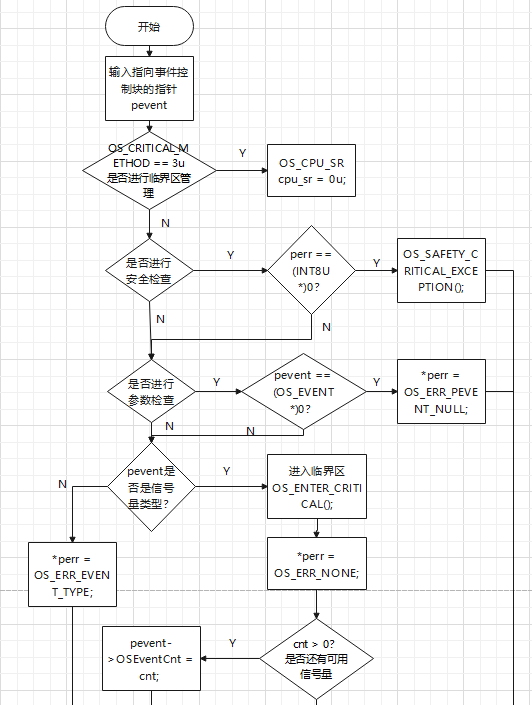
Void OSSemSet (OS\_EVENT \*pevent,INT16U cnt,INT8U \*perr);

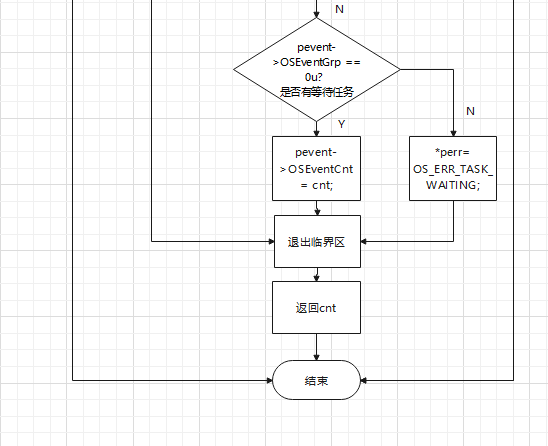
* 函数功能：

此函数用于将信号量计数设置为指定为参数的值。将输入的cnt值设置为pevent指向的事件控制块的信号量数目，如果函数执行中发生错误，向perr中写入错误信息。改好输一般用于重置事件信号资源。

* 参数：
  + pevent: 指向与所需信号量关联的事件控制块的指针。
  + cnt:要设定的信号量的数。
  + perr:指向函数返回的错误。
    - OS\_ERR\_NONE：函数正确执行，无错误。
    - OS\_ERR\_EVENT\_TYPE：指针不是指向信号量类型。
    - OS\_ERR\_PEVENT\_NULL：pevent为空指针。
    - OS\_ERR\_TASK\_WAITING：任务在等待信号量。

### 7.8.2 流程图





## 7.9 OSMutexAccept()

### 7.9.1概述

* 函数头：

BOOLEAN OSMutexAccept (OS\_EVENT \*pevent, INT8U \*perr);

* 函数功能：

这个函数检查互斥信号量，看是否有资源可用。与OSMutexPend()不同，如果资源不可用或者事件没有发生，OSMutexAccept()不会挂起调用任务。这个函数会检查是否启用了优先级继承机制。

* 参数：
  + pevent: 指向与所需信号量关联的事件控制块的指针。
  + perr: 指向错误消息存放位置的指针。可能的错误有：
    - OS\_ERR\_PEVENT\_NULL：事件指针pevent为空。
    - OS\_ERR\_EVENT\_TYPE：事件指针pevent指向的不是互斥锁类型。
    - OS\_ERR\_PEND\_ISR：不能在中断处理程序中请求获取互斥锁。
    - OS\_ERR\_PCP\_LOWER：可提升的优先级比占有互斥锁task的优先级还低。注：发生此错误时，仍可获取到互斥锁。
    - OS\_ERR\_NONE：未发生错误。
* 返回值：
  + OS\_TRUE: 成功获取互斥锁。
  + OS\_FALSE：获取互斥锁失败。

### 7.9.2流程图



## OSMutexCreate()

#### 7.10.1 概述

* 函数头：

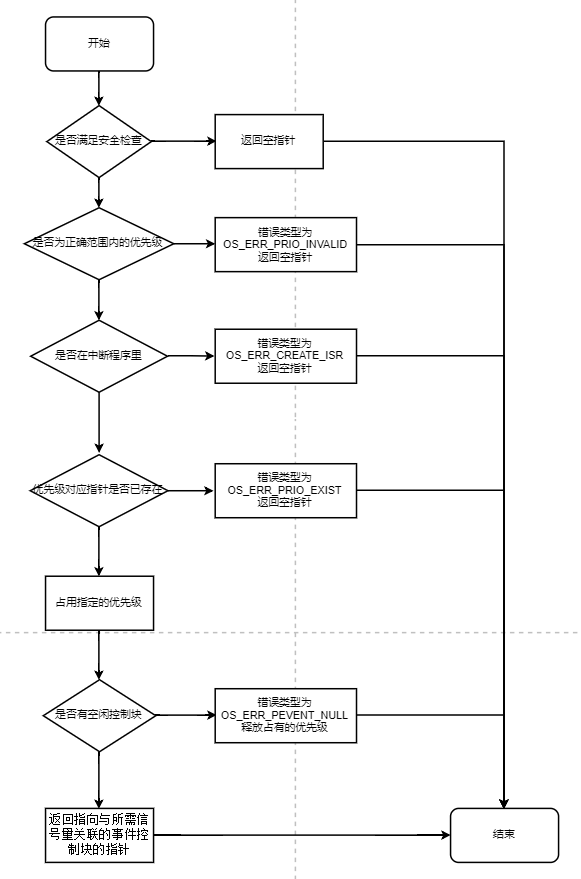
OS\_EVENT \*OSMutexCreate (INT8U prio,INT8U \*perr)

* 函数功能：

该函数的目的是初始化事件控制块为互斥信号量，其中，当prio输入为OS\_PRIO\_MUTEX\_CEIL\_DIS，任务将不进行优先级提升，此时和单一的信号量的作用是相同的。

* 参数：
  + prio: 获得互斥型信号量的任务的优先级。换句话说，当有任务已经获得了信号量，但是更高优先级任务尝试获得该信号量时，占有信号量的任务将其优先级提高到高优先级。此时任务的优先级比所有竞争mutex的任务优先级都高。
  + perr: 指向错误消息存放位置的指针。可能的错误有：
  + pevent：指向与所需信号量关联的事件控制块的指针。
    - OS\_ERR\_PRIO\_INVALID：错误（非法）的优先级。
    - OS\_ERR\_CREATE\_ISR：事件处于中断程序里
    - OS\_ERR\_PRIO\_EXIST：优先级对应的指针非空（已存在）。
    - OS\_ERR\_PEVENT\_NULL：无空闲控制块。
    - OS\_ERR\_NONE：无错误；
    - OS\_ERR\_PEVENT\_NULL：没有可用的事件控制块
* 返回值：
  + !=0 返回一个指针 ,该指针指向分配给mutex的事件控制块。
  + ==0 有错误，返回空指针。
* 注释：
  + OSEventCnt的低八位用来表示拥有互斥量任务的优先级，如果没有任务拥有该互斥量，低八位为0xff
  + OSEventCnt的高八位存放的优先级是用来缓解优先级反转

#### 7.10.2流程图



## 7.11 OSMutexDel()

7.11.1概述

* 函数头：

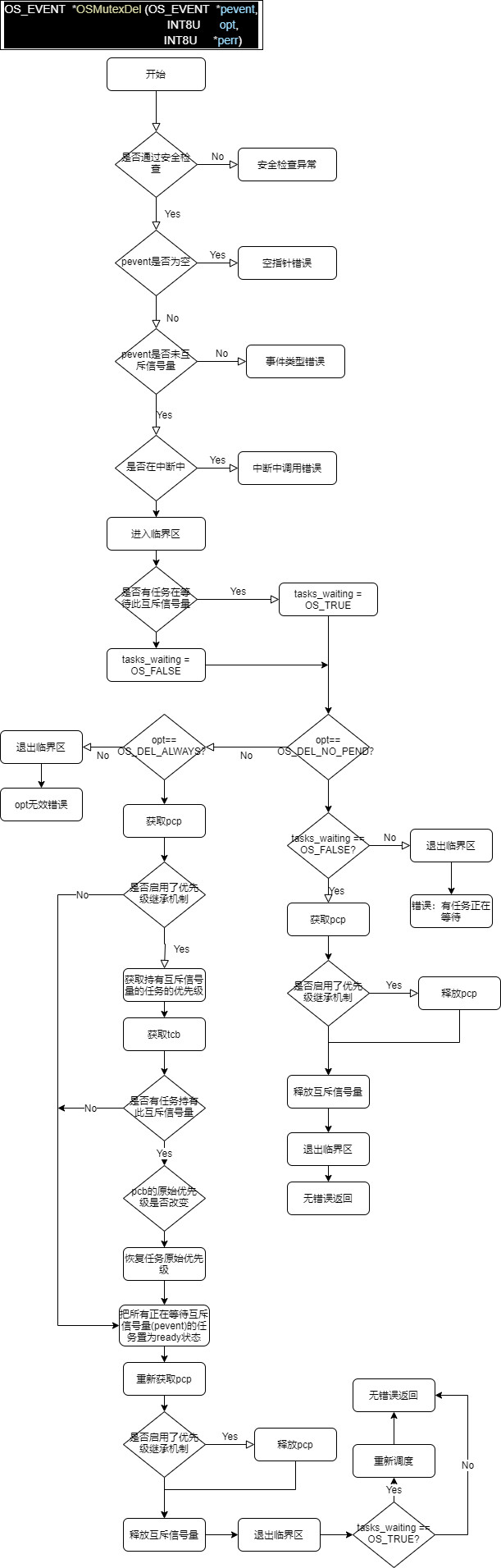
OS\_EVENT \*OSMutexDel (OS\_EVENT \*pevent, INT8Uopt, INT8U \*perr)

* 函数功能：

此函数用于删除一个互斥信号量，有两种模式，第一种是OS\_DEL\_NO\_PEND，只在没有任务等待此互斥信号量的时候删除它，第二种是OS\_DEL\_ALWAYS，即使有任务在等待也要删除，它会把正在等待的任务置为就绪状态

* 参数：
  + pevent: 指向与所需信号量关联的事件控制块的指针。
  + opt: 删除的模式，OS\_DEL\_NO\_PEND或OS\_DEL\_ALWAYS。
  + perr: 指向错误消息存放位置的指针。可能的错误有：
    - OS\_ERR\_NONE：无错误。
    - OS\_ERR\_DEL\_ISR：在中断中调用此函数。
    - OS\_ERR\_PEND\_ISR：不能在中断处理程序中请求获取互斥锁。
    - OS\_ERR\_INVALID\_OPT：opt参数不是OS\_DEL\_NO\_PEND或OS\_DEL\_ALWAYS。
    - OS\_ERR\_TASK\_WAITING：有任务正在等待此互斥信号量。
    - OS\_ERR\_EVENT\_TYPE：pevent类型不是互斥信号量。
    - OS\_ERR\_PEVENT\_NULL：pevent是空指针。

### 7.11.2流程图



## 7.12 OSMutexPend()

### 7.12.1概述

* 函数头：

void OSMutexPend (OS\_EVENT \*pevent, INT32U timeout, INT8U \*perr);

* 函数功能：

这个函数等待一个互斥锁。如果互斥锁可用，调用者会立即拿到互斥锁，否则调用者会进入阻塞状态。在发现互斥锁被其他task占用时，它会检查是否可以使用优先级继承机制，如果可以调用者会提升占有互斥锁task的优先级，然后进入阻塞状态。

* 参数：
  + pevent: 指向与所需信号量关联的事件控制块的指针。
  + timeout: 获取互斥锁的最长等待时间。
  + perr: 指向错误消息存放位置的指针。可能的错误有：
    - OS\_ERR\_PEVENT\_NULL：事件指针pevent为空。
    - OS\_ERR\_EVENT\_TYPE：事件指针pevent指向的不是互斥锁类型。
    - OS\_ERR\_PEND\_ISR：不能在中断处理程序中请求获取互斥锁。
    - OS\_ERR\_PEND\_LOCKED：不能在调度器加锁时请求互斥锁，否则可能出现死锁。
    - OS\_ERR\_PCP\_LOWER：可提升的优先级比占有互斥锁task的优先级还低。注：发生此错误时，仍可获取到互斥锁。
    - OS\_ERR\_NONE：未发生错误。
    - OS\_ERR\_PEND\_ABORT：获取互斥锁被终止。
    - OS\_ERR\_TIMEOUT：获取互斥锁超时。

### 7.12.2流程图



## 7.13 OSMutexPost()

### 7.13.1概述

* 函数头：

OSMutexPost(OS\_EVENT \*pevent)

* 函数功能：

使得任务自己释放信号量。如果发现等待队列中有任务，从中选出优先级最高的任务置为就绪态，并让他占用互斥量；如果等待队列中没有任务，将调用函数的任务释放互斥量之后就退出。

* 参数：
  + pevent：指向与互斥量的事件控制块的指针。
* 返回值：
  + OS\_ERR\_NONE：调用成功，返回无错误；
  + OS\_ERR\_EVENT\_TYPE：pevent指针指向的类型不是mutex；
  + OS\_ERR\_PEVENT\_NULL：pevent为空指针；
  + OS\_ERR\_POST\_ISR：从中断服务子程序中调用；
  + OS\_ERR\_NOT\_MUTEX\_OWNER：发出信号的任务没有占用互斥量；
  + OS\_ERR\_PIP\_LOWER：拥有互斥量的任务的优先级比互斥量优先级上限高。

### 7.13.2流程图



## 7.14 OSMutexQuery()

### 7.14.1概述

* 函数头：

OSMutexQuery (OS\_EVENT \*pevent,

OS\_MUTEX\_DATA \*p\_mutex\_data)

* 函数功能：

用于获取关于互斥量的信息，并将互斥型信号量的信息储存在OS\_MUTEX\_DATA类型的p\_mutex\_data中。

* 参数：
  + pevent: 指向与互斥量的事件控制块的指针。
  + p\_mutex\_data：指向互斥量信息储存地址的指针。
* 返回值：
  + OS\_ERR\_NONE：调用成功，互斥量消息也成功存储。
  + OS\_ERR\_QUERY\_ISR：从中断服务子程序中调用该函数；
  + OS\_ERR\_PEVENT\_NULL：pevent为空指针；
  + OS\_ERR\_PDATA\_NULL：p\_mutex\_data为空指针；
  + OS\_ERR\_EVENT\_TYPE：pevent指针指向的类型不是mutex；

### 7.14.2流程图



## 7.15 OSMutex\_RdyAtPrio()

#### 7.15.1 概述

* 函数头：

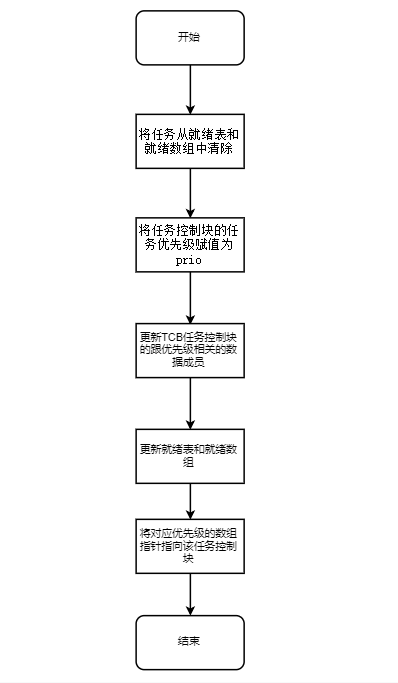
OSMutex\_RdyAtPrio (OS\_TCB \*ptcb, INT8U prio)

* 函数功能：

将任务还原为其原始优先级，该功能是将之前为了防止优先级反转而提高的任务的优先级恢复。

* 参数：
  + prio: 获得互斥型信号量的任务的优先级。换句话说，当有任务已经获得了信号量，但是更高优先级任务尝试获得该信号量时，占有信号量的任务将其优先级提高到高优先级。此时任务的优先级比所有竞争mutex的任务优先级都高。
  + ptcb：指向要准备就绪的任务的OS\_TCB的指针。
* 返回值：无

#### 7.15.2流程图



# 分工与贡献率表格

## 8.1函数分工

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

平均每人代码行数约为125行。

## 8.2其他分工

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

## 8.3贡献率表格

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |