12.1 事件标志组

有时候一个任务可能需要和多个事件同步，这个时候就需要使用事件标志组。事件标志组与任务之间有两种同步机制：“或”同步和“与”同步，当任何一个事件发生，任务都被同步的同步机制是“或”同步；需要所有的事件都发生任务才会被同步的同步机制是“与”同步，这两种同步机制如图 12.1.1 所示。

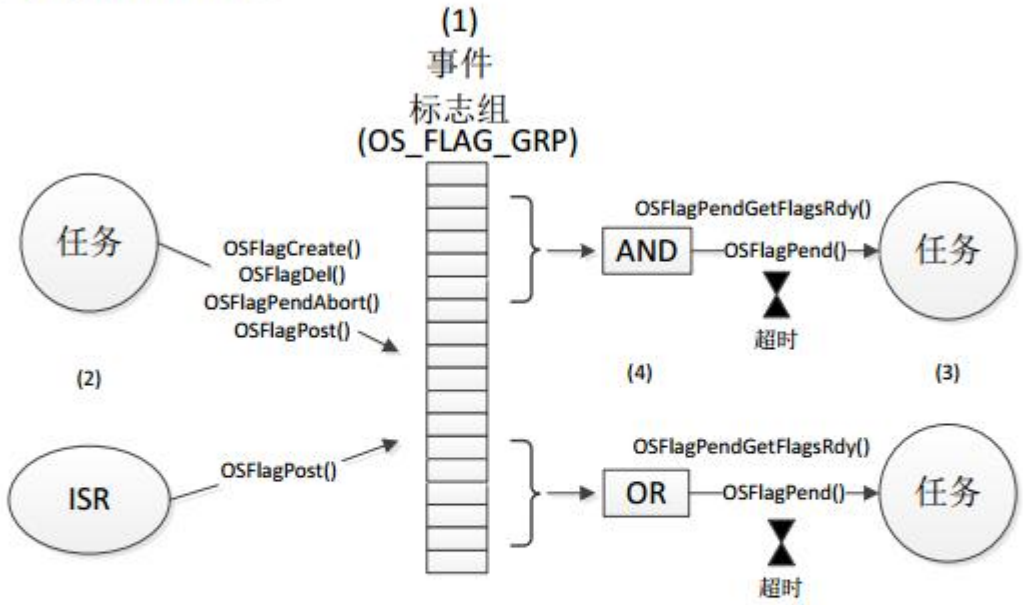

图 12.1.1 事件标志组

(1) 在 UCOSIII 中事件标志组是 OS_FLAG_GRP，在 os.h 文件中有定义，事件标志组中也包含了一串任务，这些任务都在等待着事件标志组中的部分(或全部)事件标志被置 1 或被清零，在使用之前，必须创建事件标志组。

(2) 任务和 ISR(中断服务程序)都可以发布事件标志，但是，只有任务可以创建、删除事件标志组以及取消其他任务对事件标志组的等待。

(3) 任务可以通过调用函数 OSFlagPend()等待事件标志组中的任意个事件标志，调用函数 OSFlagPend()的时候可以设置一个超时时间，如果过了超时时间请求的事件还没有被发布，那么任务就会重新进入就绪态。

(4) 我们可以设置同步机制为“或”同步还是“与”同步。

UCOSIII 中关于事件标志组的 API 函数如表 12.1.1 所示，一般情况下我们只使用 OSFlagCreate()、OSFlagPend()和 OSFlagPost()这三个函数。

函数	描述
OSFlagCreate()	创建事件标志组
OSFlagDel()	删除事件标志组
OSFlagPend()	等待事件标志组
OSFlagPendAbort()	取消等待事件标志组
OSFlagPendGetFlagsRdy()	获取使任务就绪的事件标志
OSFlagPost()	向事件标志组发布标志

表 12.1.1 事件标志组 API 函数

12.2 事件标志组相关函数

12.2.1 创建事件标志组

在使用事件标志组之前，需要调用函数 OSFlagCreate()创建一个事件标志组，OSFlagCreate()函数原型如下。

```
void    OSFlagCreate ( OS_FLAG_GRP     *p_grp,
                       CPU_CHAR        *p_name,
                       OS_FLAGS        flags,
                       OS_ERR          *p_err)
```

p_grp： 指向事件标志组，事件标志组的存储空间需要应用程序进行实际分配，我们可以按照下面的例子来定义一个事件标志组。

```
OS_FLAG_GRP    EventFlag;
```

p_name： 事件标志组的名字。

flags： 定义事件标志组的初始值。

p_err： 用来保存调用此函数后返回的错误码。

12.2.2 等待事件标志组

等待一个事件标志组需要调用函数 OSFlagPend()，函数原型如下。

```
OS_FLAGS   OSFlagPend ( OS_FLAG_GRP    *p_grp,
                        OS_FLAGS       flags,
                        OS_TICK        timeout,
                        OS_OPT         opt,
                        CPU_TS         *p_ts,
                        OS_ERR         *p_err)
```

OSFlagPend()允许将事件标志组里事件标志的“与或”组合状态设置成任务的等待条件。任务等待的条件可以是标志组里任意一个标志置位或清零，也可以是所有事件标志都置位或清零。如果任务等待的事件标志组不满足设置的条件，那么该任务被置位挂起状态，直到等待的事件标志组满足条件、指定的超时时间到、事件标志被删除或另一个任务终止了该任务的挂起状态。

p_grp： 指向事件标志组。

opt： 决定任务等待的条件是所有标志置位、所有标志清零、任意一个标志置位还是任意一个标志清零，具体的定义如下。

OS_OPT_PEND_FLAG_CLR_ALL	等待事件标志组所有的位清零
OS_OPT_PEND_FLAG_CLR_ANY	等待事件标志组中任意一个标志清零
OS_OPT_PEND_FLAG_SET_ALL	等待事件标志组中所有的位置位
OS_OPT_PEND_FLAG_SET_ANY	等待事件标志组中任意一个标志置位

调用上面四个选项的时候还可以搭配下面三个选项。

OS_OPT_PEND_FLAG_CONSUME	用来设置是否继续保留该事件标志的状态。
OS_OPT_PEND_NON_BLOCKING	标志组不满足条件时不挂起任务。
OS_OPT_PEND_BLOCKING	标志组不满足条件时挂起任务。

这里应该注意选项 OS_OPT_PEND_FLAG_CONSUME 的使用方法，如果我们希望任务等待事件标志组的任意一个标志置位，并在满足条件后将对应的标志清零那么就可以搭配使用选项 OS_OPT_PEND_FLAG_CONSUME。

p_ts： 指向一个时间戳，记录了发送、终止和删除事件标志组的时刻，如果为这个指针赋值 NULL，则函数的调用者将不会收到时间戳。

p_err： 用来保存调用此函数后返回的错误码。

12.2.3 向事件标志组发布标志

调用函数 OSFlagPost()可以对事件标志组进行置位或清零，函数原型如下。

```
OS_FLAGS    OSFlagPost (  OS_FLAG_GRP    *p_grp,
                          OS_FLAGS       flags,
                          OS_OPT         opt,
                          OS_ERR         *p_err)
```

一般情况下，需要进行置位或者清零的标志由一个掩码确定（参数 flags）。OSFlagPost()修改完事件标志后，将检查并使那些等待条件已经满足的任务进入就绪态。该函数可以对已经置位或清零的标志进行重复置位和清零操作。

p_grp： 指向事件标志组。

flags： 决定对哪些位清零和置位，当 opt 参数为 OS_OPT_POST_FLAG_SET 的时，参数 flags 中置位的位就会在事件标志组中对应的位也将被置位。当 opt 为 OS_OPT_POST_FLAG_CLR 的时候参数 flags 中置位的位在事件标志组中对应的位将被清零。

opt： 决定对标志位的操作，有两种选项。

OS_OPT_POST_FLAG_SET 对标志位进行置位操作

OS_OPT_POST_FLAG_CLR 对标志位进行清零操作

p_err： 保存调用此函数后返回的错误码。