uCOSIII

Semaphore

(\*) Semaphore可以由pending的task來post,也可以經由其他task或ISR來post

(\*) If the semaphore counter ( .Ctr of OS\_SEM) is greater than zero, the counter is decremented and returns, which indicates that the signal occurred. This is OSSemPend()

the outcome that the caller expects

If the semaphore counter is zero, this indicates that the signal has not occurred and the calling task might need to wait for the semaphore to be released

1.可以將OSSemPend()視為其他library的lock()函式; 但有些不一樣

OSSemPend()執行時, 就算sem不被其他task所佔, 還是必須等ISR或另外的Task post, pending的task才會回到ready state??

Binary semaphore, countable semaphore?? P209參考

(\*) Counting Semaphores:在Create Sem時, 將其counter value 設定為n; (n>1)

(\*) binary Semaphore: 在Create Sem時,將其counter value 設定為1

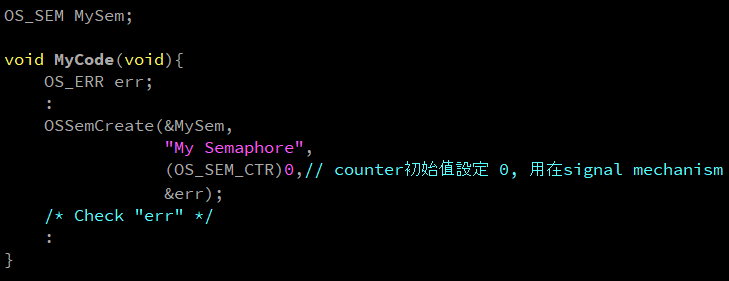
(\*)A semaphore is more often used as a signaling mechanism and therefore, the semaphore counter is initialized to zero.

1. semaphore更常被使用來 作為signal的機制, 所以此情況下,semaphore在create時,他的counter value會設為”0”. P220

(\*) uC/OS-III keeps track of how many times the semaphore is signaled with this counter and this field is typically initialized to zero by OSSemCreate(). P251

(\*)The value “N” next to the flag indicates that the semaphore can accumulate events or credits. An ISR (or a task) can post (or signal) multiple times to a semaphore and the semaphore will remember how many times it was posted. It is possible to initialize the semaphore with a value other than zero, indicating that the semaphore initially contains that number of events

1. 當creaete Sem時將counter value設定為”0”, 則kernel會追蹤semaphore被signal(post)的次數.



2.Semaphore本身有一個counter在其中, 表示event發生的數目

3. N 表示semaphore 可以累計event,或credit

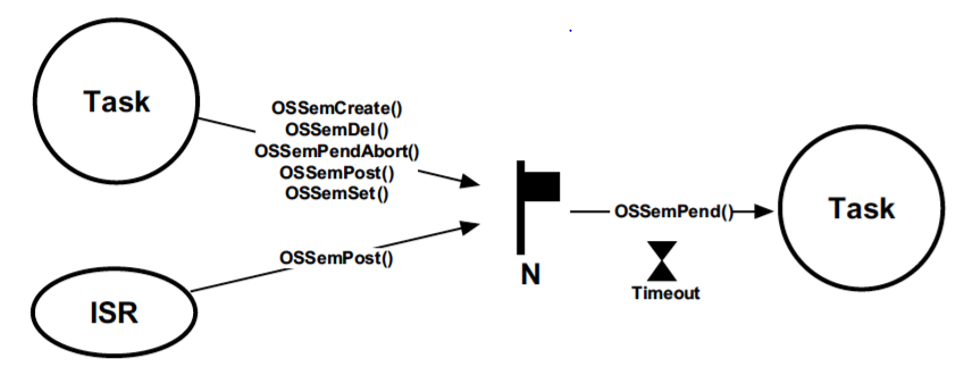
累計: task pend的次數, 且semaphore尚未被post出來

Credit: semaphore 被post的次數, 且尚未被消耗, ISR或task可以post出一個semaphore很多次, 此semaphore會記錄被post的次數

3.可以初始化semaphore為非0值; countable semaphore (n>1)或 binary semaphore(n=1)

(\*) This timeout indicates that the task is willing to wait for the semaphore to be signaled(or posted to) within a certain amount of time.

1.在pending semaphore的呼叫時,可設定pending的timeout值; 表示task過了此timeout,會跳出pending狀態, 將狀態遷移到ready.



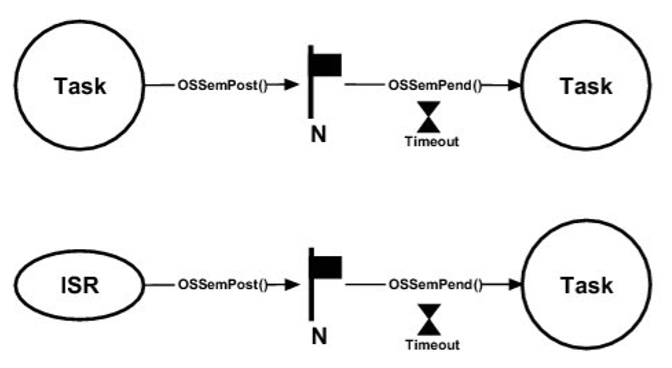
(\*)When used for synchronization, a semaphore keeps track of how many times it was signaled using a counter

當用semaphore來記錄被post多少次, 需要用一個內部counter來計數OS\_SEM\_CTR

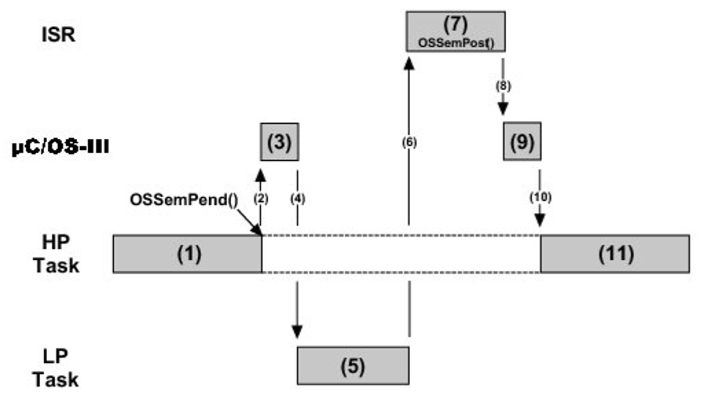
(\*) a task can be synchronized with an ISR (or another task) by using a semaphore

可以用ISR或task來post一個signal來同步另外一個task;此種方式稱為Unilateral Rendez-Vous

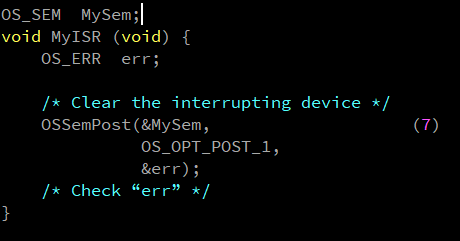
此方式下pending的task不用主動將的semaphore post出去,post的動作由ISR或其他task來完成.

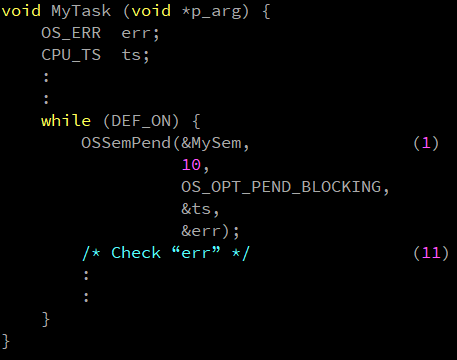


時序圖:



範例程式:

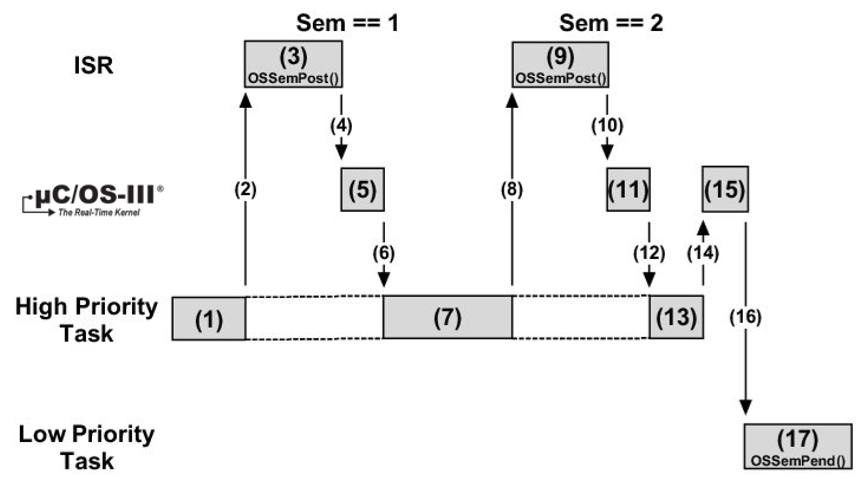




(\*)Credit Tracking

As previously mentioned, a semaphore “remembers” how many times it was signaled (or posted to). In other words, if the ISR occurs multiple times before the task waiting for the event becomes the highest-priority task, the semaphore will keep count of the number of times it was signaled. When the task becomes the highest priority ready-to-run task, it will execute without blocking as many times as there were ISRs signaled

1.Semaphore會記錄自己被post幾次; low Priority被 h priority preemt, 且過程中ISR發生多次且post多次sem; 當重新被某個task pend, 則此task執行次數會跟post一樣的次數, 而且是在不會被blocking的狀況之下執行

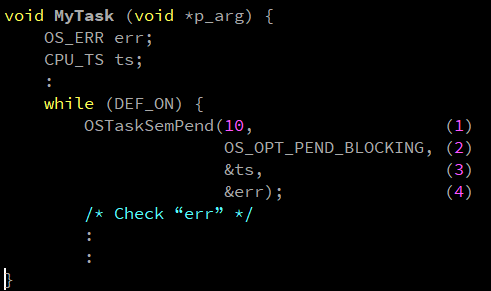


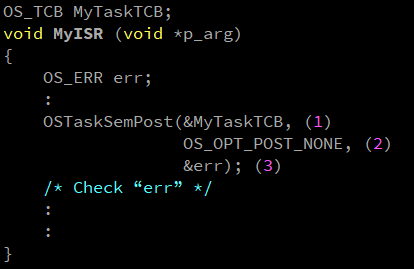
(\*)Broadcasting is a common technique used to synchronize multiple tasks and have them start executing at the same time.

1.Post semaphore,可以用broadcasting的方式做post, 此模式可提讓pend的task 同時一次被喚醒, 同時從pending state, 轉換到ready state.

(\*)Task Semaphore:

1. Task本身就有內建semaphore, built-in semaphore

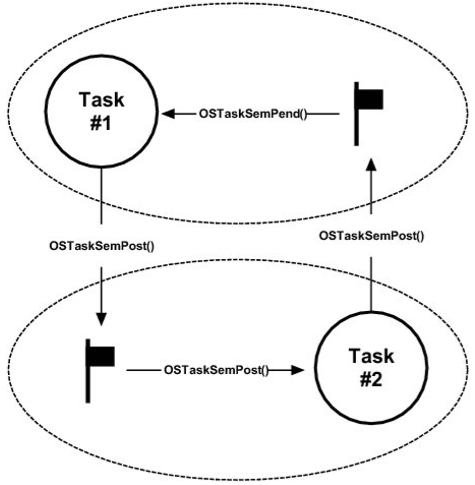




(\*)Bilateral Rendez-Vous - Task Synchronization

Two tasks can synchronize their activities by using two task semaphores

1.兩個task 利用兩個不同的semahore來互相同步



範例程式:

