嵌入式系統

單元6：uC/OS-II為例之RTOS

授課教師：雲林科技大學 張慶龍 老師

單元學習目標與大綱

- uC/OS-II簡介
- uC/OS-II排程
- uC/OS-II軟體架構
- uC/OS-II Task同步-Semaphore
- uC/OS-II 訊息傳遞-Mailbox
- uC/OS-II 多訊息傳遞-Message Queue
- uC/OS-II 多事件觸發-Flags
- uC/OS-II 記憶體管理
- uC/OS-II 時間管理

Part 1
uC/OS-II簡介

uC/OS-II (1)

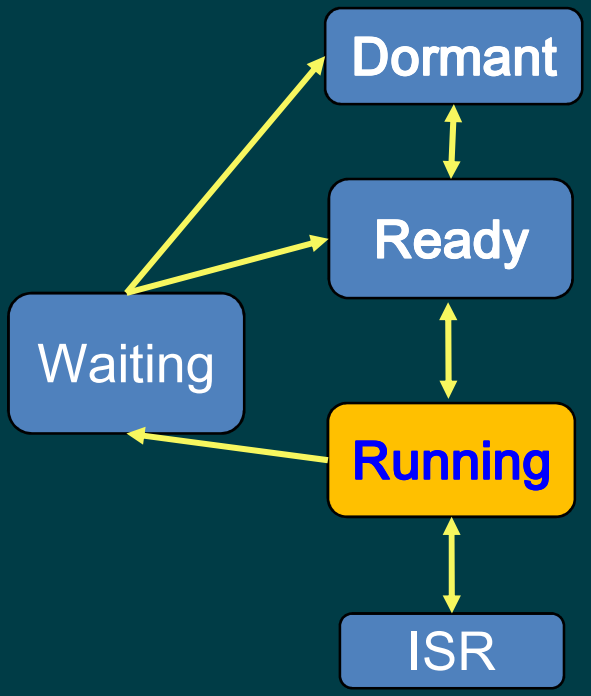
- Real-time kernel
- Priority-based preemptive 排程機制
  - 目前在即行的Task即為ready-list內最高優先權的Task
- 基本上有64個priority number
  - Number愈小，優先權愈大
- 每個task都要指定一個priority number
  - 不同task，不可設定相同的priority number
  - Priority number 0, 1, 2, 3為系統保留的號碼，不要使用

uC/OS-II (2)

- 兩個system tasks: Idle Task, Statistics Task
  - 對應的priority number 為 63與62
- Priority number即為task之ID
- 相關資訊可參加書籍 MicroC/OS-II: The Real-Time Kernel (A complete portable, ROMable scalable preemptive RTOS), Jean J. LaBrosse, CMP Books

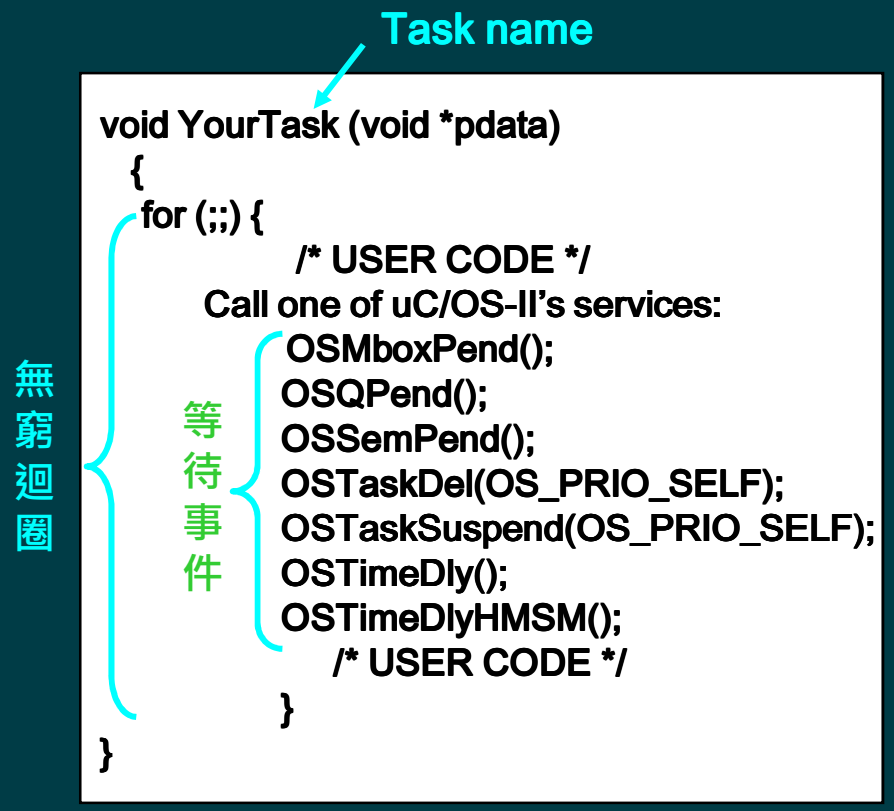
uC/OS-II Task States

- 5種可能的狀態
  - Dormant state: 被create但OS還不知其存在
  - Ready state: 在ready-list等待被執行
  - Running state:目前正在CPU執行
  - Waiting state: 在waiting-list等待事件發生
  - ISR state: 在執行時，被中斷插斷



uC/OS-II Task

- 為一無窮迴圈的函數
- 建立Task
  - OSTaskCreate()
  - OSTaskCreateExt()
- 刪除Task
  - OSTaskDel()
- 暫停Task
  - OSTaskSuspend()
- 恢復Task
  - OSTaskResume()



uC/OS-II Critical Section保護

- 利用disable interrupts方式進入critical sections，離開時再reenable interrupts
- uC/OS-II 定義兩個macros可對中斷disable/enable
  - OS_ENTER_CRITICAL( )
  - OS_EXIT_CRITICAL( )
- 三種方式實現macros，主要是採用方法二
  - #define OS_ENTER_CRITICAL () \
    asm(“ PUSH PSW”) \
    asm(“DI”)
  - #define OS_EXIT_CRITICAL () \
    asm(“POP PSW”)

Part 2
uC/OS-II排程

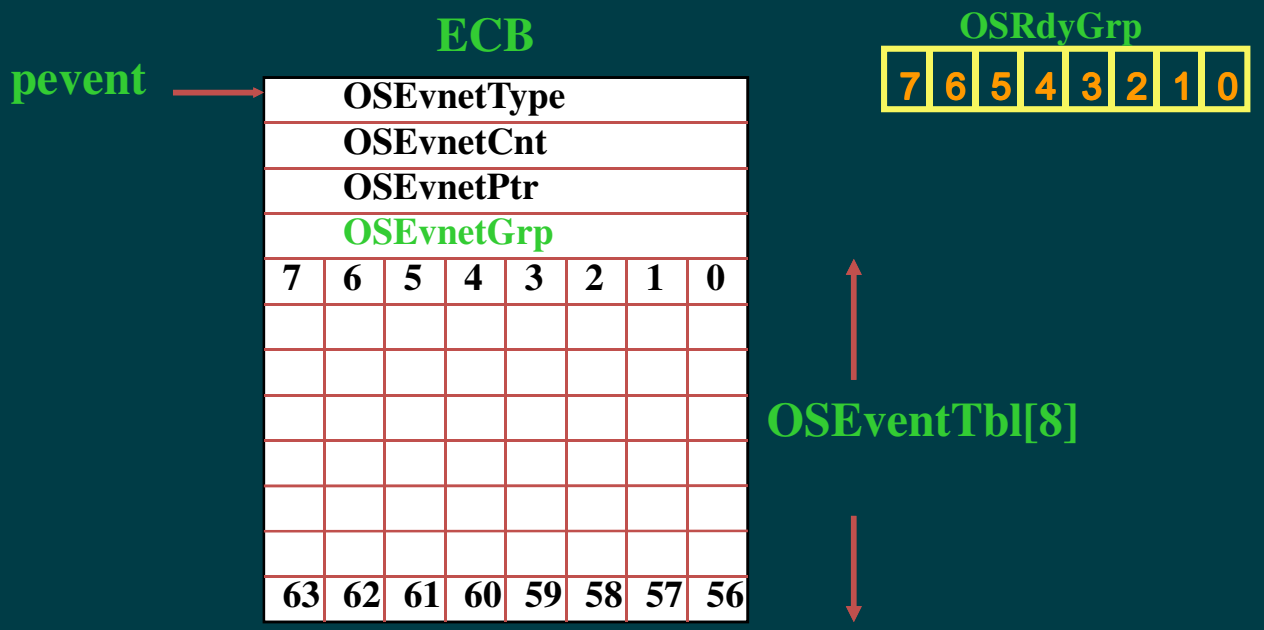
uC/OS-II排程技巧

- 每個 semaphore, mailbox, and queue 都有自己的ECB (Event Control Block)
  - 當有Task等待semaphore, mailbox,或 queue, 即將此等待Task ID放在對應的ECB
  - 當等待事件滿足時，即藉由ECB找出所有等待Task中，最高優先權的Task，將其由waiting state改為ready state

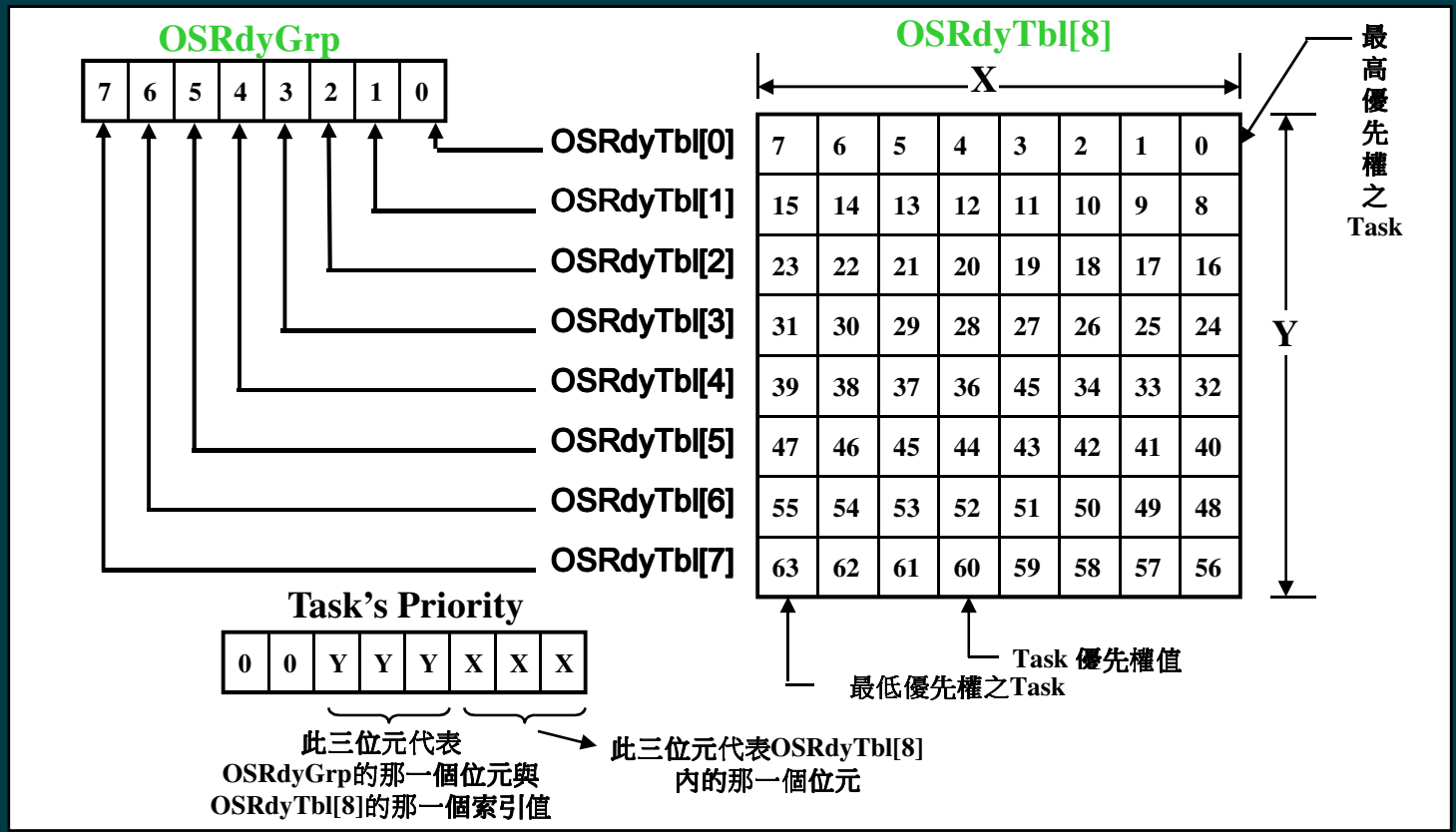
ECB

```
typedef struct {
   INT8U   OSEventType;              /* Event type                    */
   INT8U   OSEventGrp;               /* Group for wait list           */
   INT16U  OSEventCnt;               /* Count (when event is a semaphore) */
   void   *OSEventPtr;               /* Ptr to message or queue structure */
   INT8U   OSEventTbl[OS_EVENT_TBL_SIZE]; /* Wait list for event to occur */
   } OS_EVENT;
```

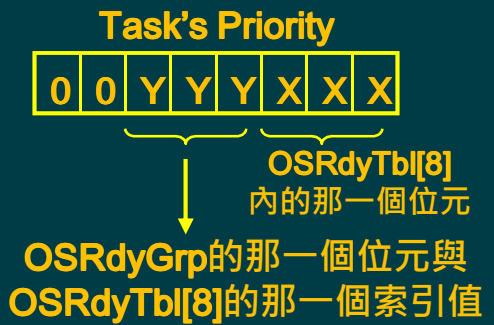
ECB主要是由EventGrp與EventTbl組成



EventGrp與EventTbl的關係



將Task加入Waiting List與移出



將Task加入Event的waiting list

```
pevent->OSEventGrp  |= OSMapTbl[prio >> 3];
pevent->OSEventTbl[prio >> 3]  |= OSMapTbl[prio & 0x07];
```

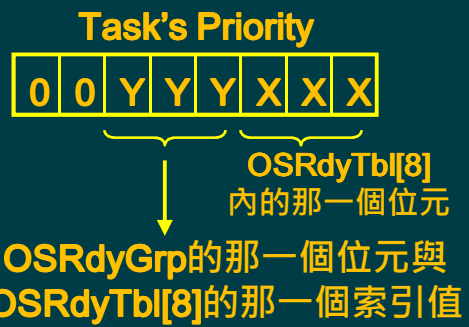
將Task由Event的waiting list移除

```
if ((pevent->OSEventTbl[prio >> 3] &= ~OSMapTbl[prio & 0x07]) == 0)
    { pevent->OSEventGrp &= ~OSMapTbl[prio >> 3]; }
```

OSMapTbl[ ]

Index	Bit Mask(Binary)
0	00000001
1	00000010
2	00000100
3	00001000
4	00010000
5	00100000
6	01000000
7	10000000

找出最高優先權的Task




```
y    = OSUnMapTbl[pevent->OSEventGrp];
x    = OSUnMapTbl[pevent->OSEventTbl[y]];
prio = (y << 3) + x;
```

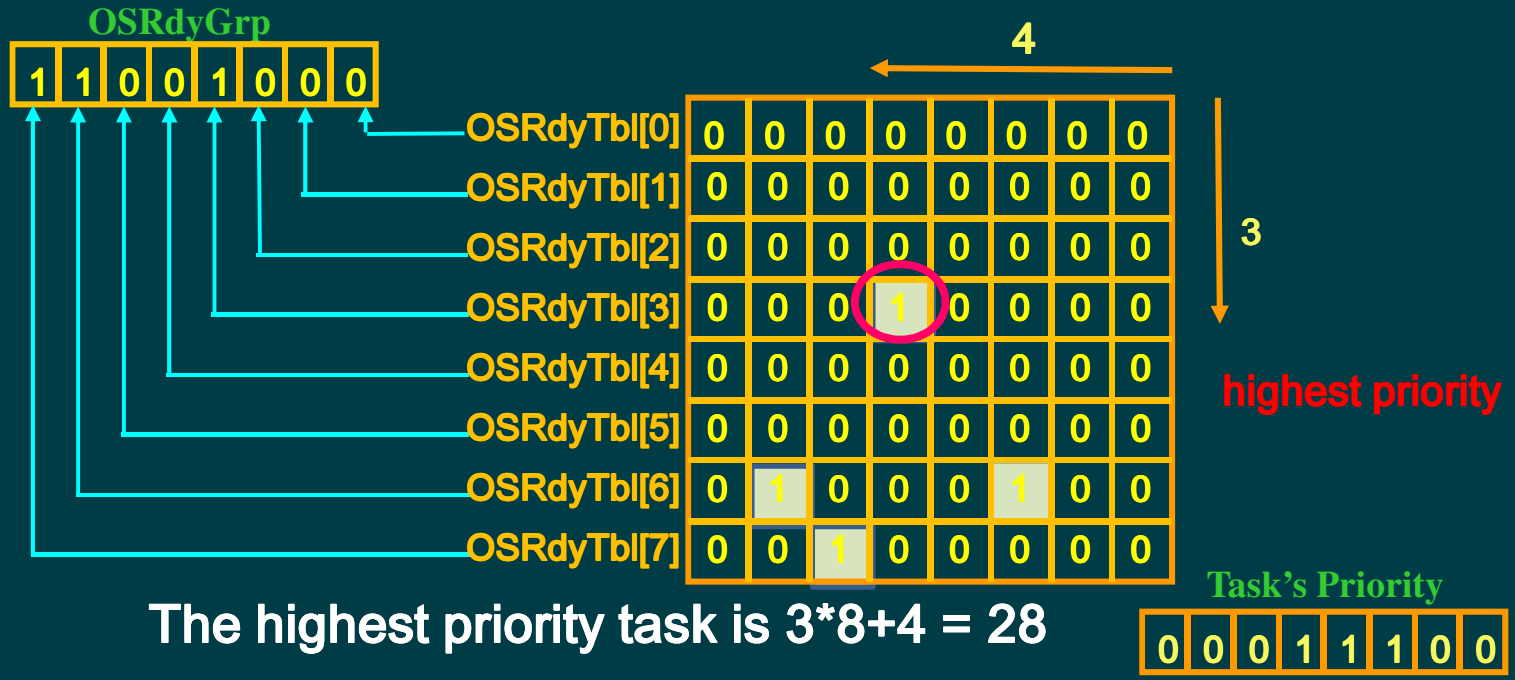
OSUnMapTbl[ ]

```
INT8U const OSUnMapTbl[] = {
    0, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    6, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    7, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    6, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0};
```

Example: 由ECB wait list挑出最高優先權Task

OSEventGrp = 11001000=> OSUnMapTbl[OSEventGrp] = 3

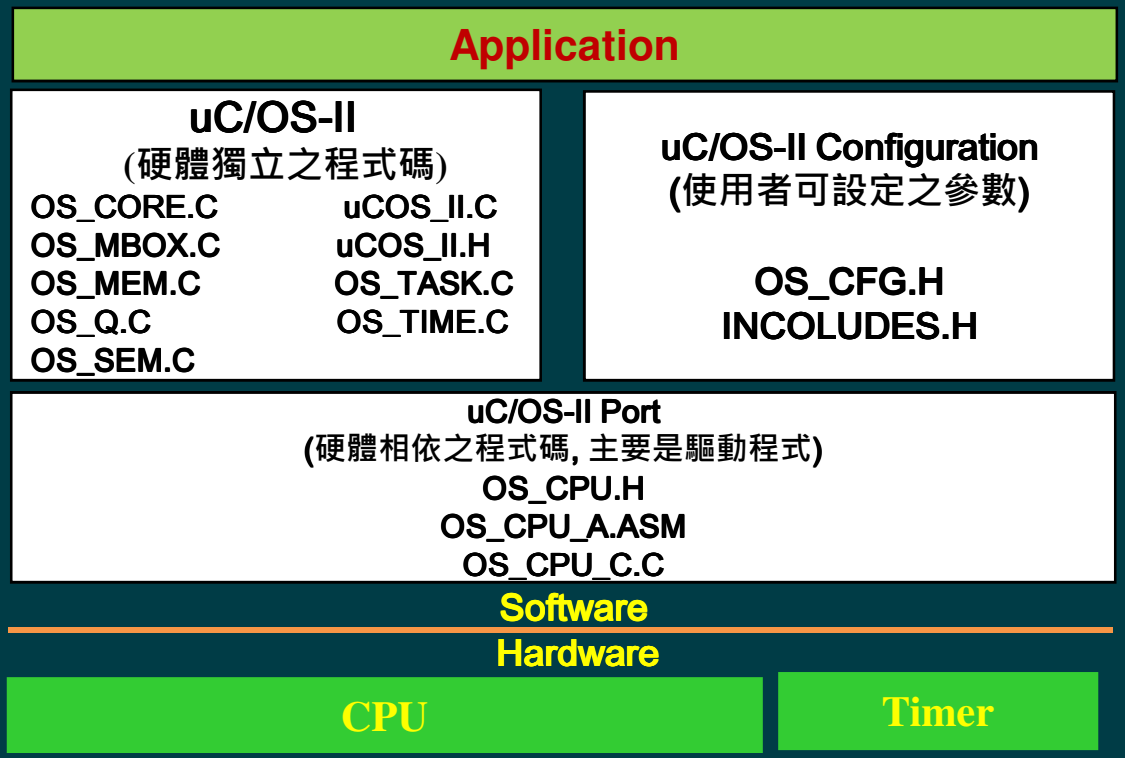
OSEventTbl[3] = 00010000 ➔ OSUnMapTbl[OSEventTbl[3]] = 4



The highest priority task is 3*8+4 = 28

Part 3
uC/OS-II軟體架構

uC/OS-II軟/硬體架構



撰寫使用者程式

- 宣告task的stack
- 撰寫各別task之內容
- 撰寫main()程式碼(程式進入點)以對OS初始化與啟動
  - 對MCU, I/O周邊與OS做初始化
  - 啟動timer, 提供tick clock
  - 創建系統所需之semaphore, mail box, …
  - 創建task
  - 啟動OS

uC/OS-II RTOS核心啟動

```
OS_STK   TaskStartStk[TASK_STK_SIZE],
void main (void)
{
   OSInit(); //create Idle task and statistical task
   RandomSem = OSSemCreate(1);
   OSTaskCreate(TaskStart  ,   //Task name
            (void *)0,       //passing parameter
            (void *)&TaskStartStk[0], //Task stack
            0);  //priority number
   OSStart();  //啟動作業系統
}
```

```
void TaskStart (void *p_arg)
{
  /* Initial System Tick */
#if (OS_TASK_STAT_EN > 0)
        OSStatInit
#endif
/*Create your application tasks  */
for(;;) {
   /* Code for TaskStart() goes here! */
}
```

撰寫Task內容

- 宣告Task所需之stack
  - 需為全域陣列變數，其變數類型為 OS_STK
- 所創建的task可為
  - 無窮迴圈的函數: 每一次迴圈必需呼叫下列函數以放棄CPU
    - OSSemPend(), OSMboxPend(), OSQPend(), OSTaskSuspend() (為事件觸發/同步Task)
    - OSTimeDly(), OSTimeDlyHMSM() (為周期性Task)
  - 只執行一次的Task:函數的最後需執行OSTaskDel()，讓自己不再被scheduler排程

一般Task Structure

- 正常Task程式架構為下列兩種

```
void YourTask (void *pdata)
        {
          for (;;) {
                    /* USER CODE */
            Call one of uC/OS-II's services:
              OSMboxPend();
              OSQPend();
              OSSemPend();
              OSTaskDel(OS_PRIO_SELF);
              OSTaskSuspend(OS_PRIO_SELF);
              OSTimeDly();
              OSTimeDlyHMSM();
            }
        }
```

等待事件Task

```
void YourTask (void *pdata)
{

        /* USER CODE */
        OSTaskDel(OS_PRIO_SELF);

}
```

只執行一次的Task

創建Task

- 可利用下列兩個函數來創建task:
  1. OSTaskCreate()
  2. OSTaskCreateExt()
- OSTaskCreate( ) 需要 4 參數:

```
INT8U OSTaskCreate(void (*task)(void *tname),
                   void *pdata,
                   OS_STK *ptos,
                   INT8U prio);
```

  1. ‘tname’ 為指標變數，指向要創建之task程式位址
  2. ‘pdata’為指標變數，當task第一次執行時，指向提供給創建task之參數
  3. ‘ptos’為指標變數，指向創建之stack啟始位址
  3. ‘prio’ 所要創建task之優先權

Task創建範例

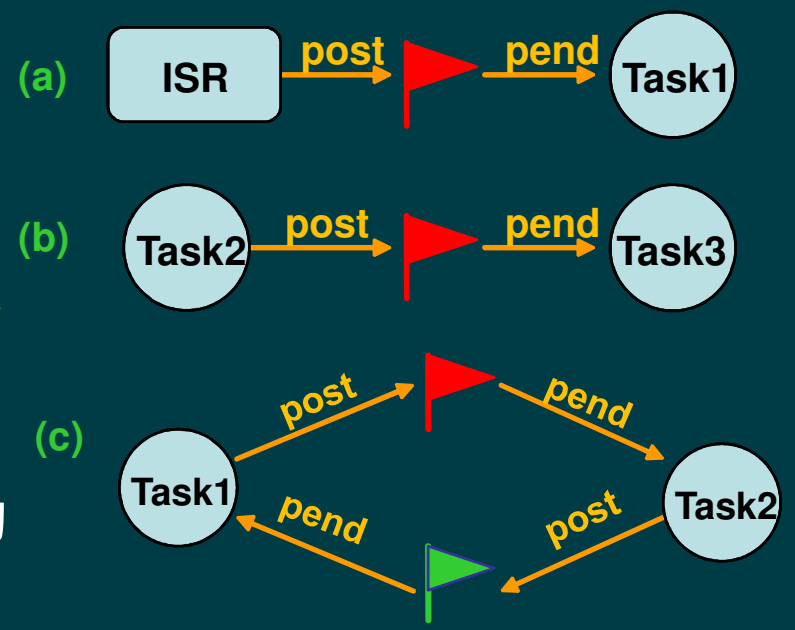
```
OS_STK Task1Stk[1024];
void main (void)
{
INT8U err;
…
OSInit(); /* Initialize µC/OS-II */
…
OSTaskCreate(Task1,
        (void *)0,
        &Task1Stk[1023],
        25);
        …
OSStart(); /* Start Multitasking */
}
```

```
void Task1 (void *p_arg)
{
(void)p_arg; /* Prevent compiler warning */
for (;;) {
                … /* Task code */
                …
}
}
```

Part 4
Task同步-Semaphore

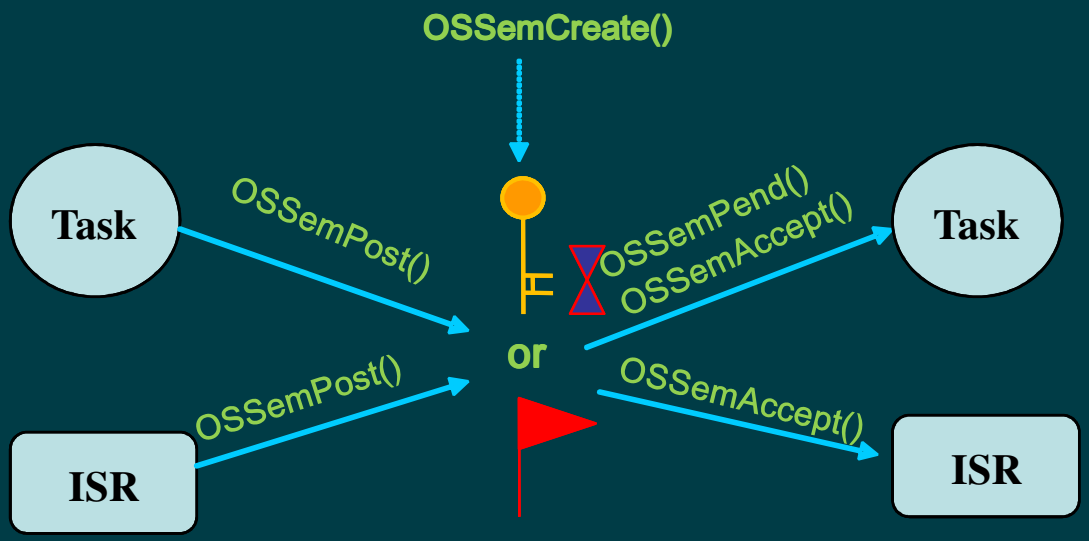
Task間同步-Semaphore

- 單一事件觸發的同步
  - ISR程式觸發Task
  - Task觸發Task
- 不可由ISR觸發ISR與Task觸發ISR
  - 會造成ISR程式執行時間不確定➔影響系統即時性能力



uC/OS-II -Semaphore

Semaphore相關之函數
OSSemCreate()
OSSemPost()
OSSemPend()
OSSemAccept()



OSSemCreate()函數範例

```
OS_EVENT *DispSem;

void main (void)
{
…
OSInit(); /* Initialize µC/OS-II */
...
DispSem = OSSemCreate(1); /* Create Display Semaphore */
...
OSStart(); /* Start Multitasking */
}
```

OSSemPost()函數範例

```
OS_EVENT *DispSem;
void TaskX (void *p_arg)
{   INT8U err;
    (void)p_arg;
     for (;;) {
          …
          err = OSSemPost(DispSem);
          switch (err) {
                        case OS_ERR_NONE:
                              /* Semaphore signaled */
                              break;
                        case OS_ERR_SEM_OVF:
                              /* Semaphore has overflowed */
                              break;
                        }
          }
}
```

OSSemPend()函數範例

```
OS_EVENT *DispSem;
void DispTask (void *p_arg)
{
        INT8U err;
        (void)p_arg;
        for (;;) {
                ...
                OSSemPend(DispSem, 0, &err);
                . /* The only way this task continues is if _ */
                . /* _ the semaphore is signaled! */
        }
}
```

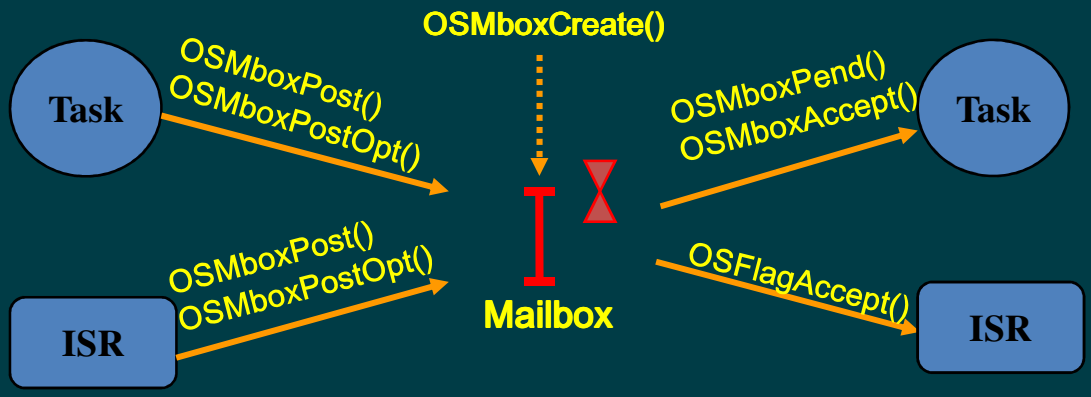
OSSemAccept()函數範例

```
OS_EVENT *DispSem;
void DispTask (void *p_arg)
{
        INT8U err;
        (void)p_arg;
        for (;;) {
                ...
                value = OSSemAccept(DispSem);
                if (value > 0) {
                        ... /* Resource available, process */
                }
}
```

Part 5
uC/OS-II 訊息傳遞-Mailbox

Mailbox相關之函數

Mailbox相關之函數	OS_CFG.H需設定之參數
OSMboxCreate()	
OSMboxPost()	OS_Mbox_Post_EN = 1
OSMboxPostOpt()	OS_Mbox_Post_OPT_EN = 1
OSMboxPend()	
OSMboxAccept()	OS_Mbox_ACCEPT_EN = 1



OSMboxCreate()函數範例

```
OS_EVENT *CommMbox;
void main (void)
{
        ...
        OSInit(); /* Initialize µC/OS-II */
        …
        CommMbox = OSMboxCreate((void *)0); /* Create COMM mailbox */
        OSStart(); /* Start Multitasking */
}
```

OSMboxPost()函數範例

```
OS_EVENT *CommMbox;
INT8U CommRxBuf[100];

void CommTaskRx (void *p_arg)
 {
        INT8U err;
        (void)p_arg;
        for (;;) {
                ...
                err = OSMboxPost(CommMbox, (void *)&CommRxBuf[0]);
                ...
               }
}
```

OSMboxPostOpt()函數範例

```
OS_EVENT *CommMbox;
INT8U CommRxBuf[100];
void CommRxTask (void *p_arg)
{
        INT8U err;
        (void)p_arg;
        for (;;) {
                ...
                err = OSMboxPostOpt(CommMbox,
                                    (void *)&CommRxBuf[0],
                                    OS_POST_OPT_BROADCAST);
                ...
              }
}
```

OSMboxPend()函數範例

```
OS_EVENT *CommMbox;
void CommTask(void *p_arg)
{           INT8U err;
            void *pmsg;
            (void)p_arg;
            for (;;) {      ...
                            pmsg = OSMboxPend(CommMbox, 10, &err);
                            if (err == OS_ERR_NONE) {
                                                ...
                                                . /* Code for received message */
                            } else {            ...
                                                . /* Code for message not received within timeout */
                                        }
                             ...
                           }
}
```

OSMboxAccept()函數範例

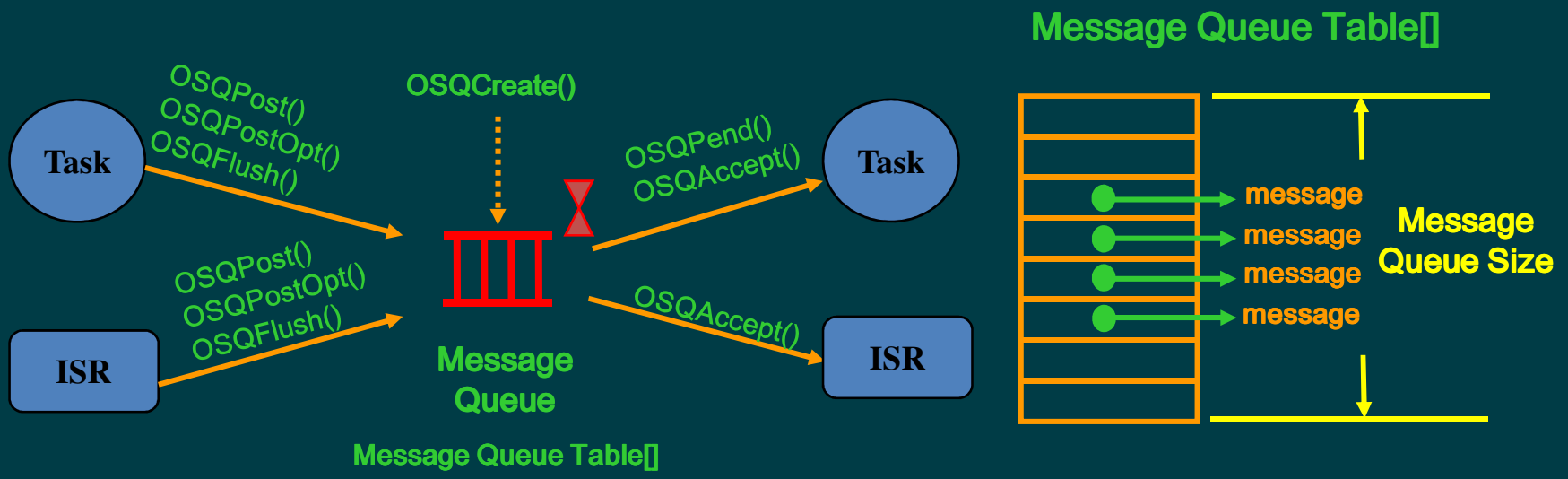
```
OS_EVENT *CommMbox;
void Task (void *p_arg)
{
        void *pmsg;
        (void)p_arg;
        for (;;) {
                pmsg = OSMboxAccept(CommMbox); /* Check mailbox for a message */
                if (pmsg != (void *)0) {
                        … /* Message received, process */
                } else {
                        … /* Message not received, do .. */
                        … /* .. something else */
                }
                ...
        }
}
```

Part 6
Message Queue

Message Queue相關之函數

相關之函數	OS_CFG.H需設定之參數
OSQCreate()	
OSQPost()	OS_Q_Post_EN = 1
OSQPostOpt()	OS_Q_Post_OPT_EN = 1
OSQPend()	
OSQAccept()	OS_Q_ACCEPT_EN = 1
OSQFlush()	OS_Q_FLUSH_EN = 1

Message Queue



OSQCreate()函數範例

```
OS_EVENT *CommQ;
void *CommMsg[10];
void main (void)
{
        OSInit(); /* Initialize µC/OS-II */
        …
        CommQ = OSQCreate(&CommMsg[0], 10); /* Create COMM Q */
        …
        OSStart(); /* Start Multitasking */
}
```

Queue Head

Queue Length

OSQPost()函數範例

```
OS_EVENT *CommQ;
INT8U CommRxBuf[100];
void CommTaskRx (void *p_arg)
{           INT8U err;
            (void)p_arg;
            for (;;) {      ...
                            err = OSQPost(CommQ, (void *)&CommRxBuf[0]);
                            switch (err) {          case OS_ERR_NONE:
                                                            /* Message was deposited into queue */
                                                            break;
                                                    case OS_ERR_Q_FULL:
                                                            /* Queue is full */
                                                            break;
                                                    …
                                                    }
                            …
                            }
}
```

Queue name

Message address

OSQPostOpt()函數範例

```
OS_EVENT *CommQ;
INT8U CommRxBuf[100];

void CommRxTask (void *p_arg)
{
        INT8U err;
        (void)p_arg;
        for (;;) {
                ...
                err = OSQPostOpt(CommQ,
                                 (void *)&CommRxBuf[0],
                                 OS_POST_OPT_BROADCAST);
                ...
                }
}
```

Queue name

Message address

廣播此訊息給所有等待的Tasks

OSQPend()函數範例

```
OS_EVENT *CommQ;

void CommTask(void *p_arg)
{
        INT8U err;
        void *pmsg;
        (void)p_arg;
        for (;;) {      ...
                        pmsg = OSQPend(CommQ, 100, &err);
                        if (err == OS_ERR_NONE) {
                                ...
                                        . /* Code for received message within 100 ticks*/
                        } else {
                                ...
                                        . /* Code for message not received, must have timeout */
                                }
                        ...
                        }
}
```

Queue name

最多等待100個ticks

OSQAccept()函數範例

```
OS_EVENT *CommQ;

void Task (void *p_arg)
{               void *pmsg;
                (void)p_arg;
                for (;;) {
                                pmsg = OSQAccept(CommQ); /* Check message queue for a message */
                                if (pmsg != (void *)0) {
                                                … /* Message received, process */
                                        } else {
                                                … /* Message not received, do .. */
                                                … /* .. something else */
                                          }
                                ...
                        }
}
```

Queue name

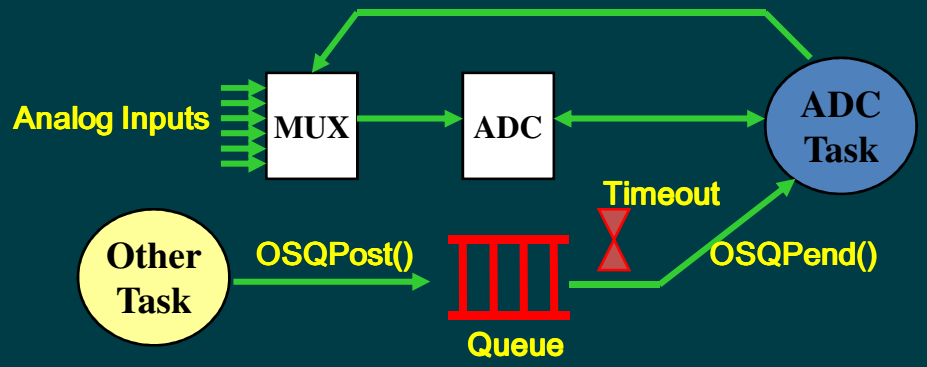
OSQFlush()函數範例

```
OS_EVENT *CommQ;

void main (void)
{
        INT8U err;
        OSInit(); /* Initialize µC/OS-II */
        ...
        err = OSQFlush(CommQ);
        …
        OSStart(); /* Start Multitasking */
}
```

Message Queue於ADC環境之應用

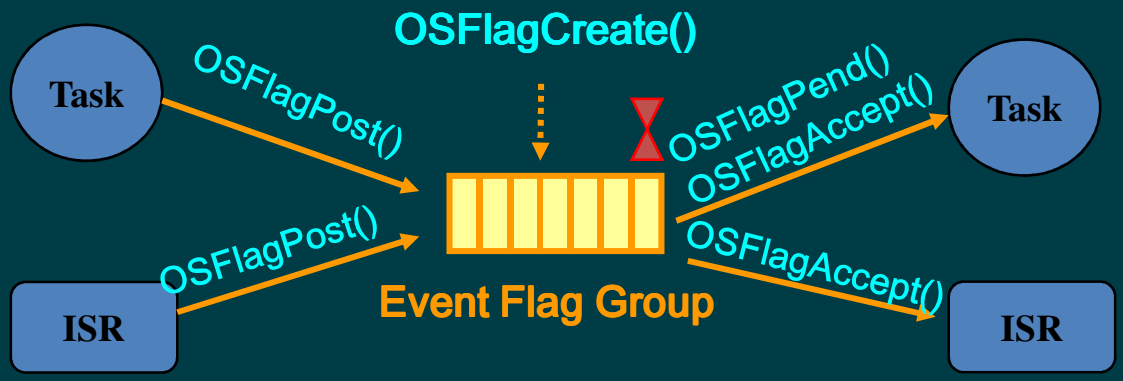
- 利用Message Queue提供週期性的ADC轉換與即時轉換需求



Part 7
uC/OS-II多事件觸發-Flags

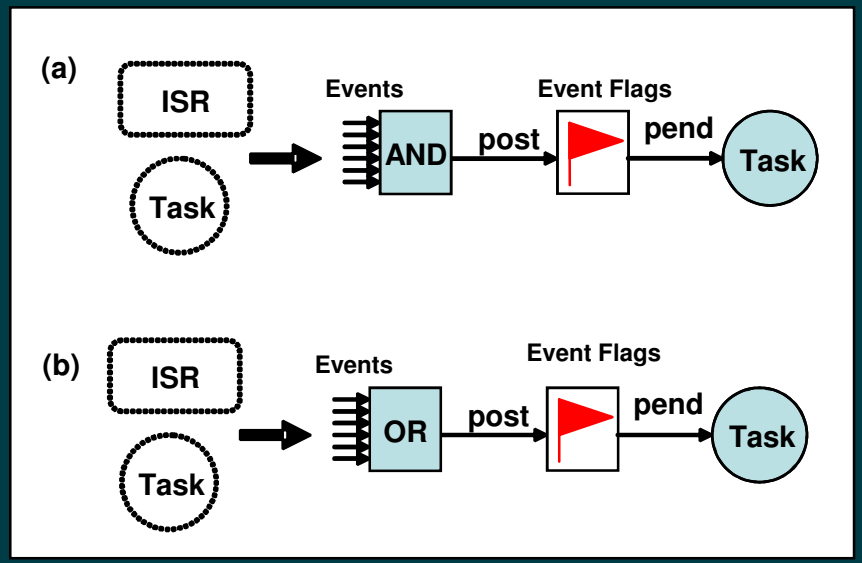
Event Flags相關之函數

Event Flags相關之函數	OS_CFG.H需設定之參數
OSFlagCreate()	
OSFlagPost()	
OSFlagPend()	
OSFlagAccept()	OS_Flag_ACCEPT_EN = 1



Task間同步-Event Falgs

- 多事件觸發同步
  - 多事件皆發生時觸發Task執行 (AND事件)
  - 多事件中任一事件發生觸發Task執行(OR事件)



OSFlagCreate()

```
OS_FLAG_GRP *OSFlagCreate(OS_FLAGS flags,
                          INT8U *perr);
```

- 參數:
  - flags: 為設定所產生的Event Flags初始值。
  - perr: 為一指標，指向錯誤碼變數，其可能之變數值為:
    - OS_ERR_NONE: 代表成功產生Event Flags。
- 回傳值:
  - OSFlagCreate()函數回傳值為一個指向Event Flags結構的指標，若已不允許產生新的Event Flags時，其回傳值為「NULL」

OSFlagCreate()函數範例

```
OS_FLAG_GRP *EngineStatus;
void main (void)
{
        INT8U err;
        ...
        OSInit(); /* Initialize µC/OS-II */
        …
        /* Create a flag group containing the engine‟s status */
        EngineStatus = OSFlagCreate(0x00, &err);
        …
        OSStart(); /* Start Multitasking */
}
```

OSFlagPost()

```
OS_FLAGS OSFlagPost(OS_FLAG_GRP *pgrp,
                    OS_FLAGS flags,
                    INT8U opt,
                    INT8U *perr);
```

- 參數:
  - pgrp: 指向Event Flags結構的指標
  - flags: 描述要將Event Flags的那幾個位元設為1或0(由opt參數決定)，若opt參數為OS_FLAG_SET，則將flags參數所指定的位元設為1；若opt為OS_FLAG_CLR，則設為0。舉例來說，若想設定Event Flags的第0, 4, 5位元，則flag需設為0x31 (00110001)
  - opt: 描述flags所指定的位元要設為1 (OS_FLAG_SET) or 0 (OS_FLAG_CLR).
  - perr: 指向執行結果之錯誤碼變數
- 回傳值: 回傳設定後的Event Flags值

OSFlagPost()函數範例

```
#define ENGINE_OIL_PRES_OK   0x01
#define ENGINE_OIL_TEMP_OK   0x02
#define ENGINE_START   0x04
OS_FLAG_GRP *EngineStatusFlags;
void TaskX (void *p_arg)
{           INT8U err;
            (void)p_arg;
            for (;;) {
                        ...
                        err = OSFlagPost(EngineStatusFlags,
                                                ENGINE_START,
                                                OS_FLAG_SET,
                                                &err);
                        ...
            }
}
```

OSFlagPend()

```
OS_FLAGS OSFlagPend(OS_FLAG_GRP *pgrp,
                    OS_FLAGS flags,
                    INT8U wait_type,
                    INT32U timeout,
                    INT8U *perr);
```

- pgrp: 為指向某Event Flags結構的指標。
- flags: 設定所要等待Event Flags的那些事件位元。
- timeout：設定要等待Event Flags事件之時間，若等待時間Timeout會返回，但會回傳一個錯誤碼，當此值設為「0」時，表無條件等待直至事件位元組發生。
- perr：表執行此函數是否正確，若其值為OS_ERR_NONE，表正確執行此函數
- 回傳值: 為0代表有問題，若非0值，其值代表所要等待Event Flags之值

OSFlagPend()

```
OS_FLAGS OSFlagPend(OS_FLAG_GRP *pgrp,
                    OS_FLAGS flags,
                    INT8U wait_type,
                    INT32U timeout,
                    INT8U *perr);
```

- wait_type: 決定是要等待Event Flags之位元事件為1或為0，其設定參數為:
  - OS_FLAG_WAIT_CLR_ALL: Event Flags的所有事件位元需皆為0
  - OS_FLAG_WAIT_CLR_ANY:依flags參數所設定的事件位元皆為0
  - OS_FLAG_WAIT_SET_ALL: Event Flags的所有事件位元需皆為1
  - OS_FLAG_WAIT_SET_ANY:依flags參數所設定的事件位元皆為1
- 若要將所發生的事件位元清為0，則此wait_type參數可設為
  - OS_FLAG_WAIT_SET_ANY + OS_FLAG_CONSUME

OSFlagPend()函數範例

```
#define ENGINE_OIL_PRES_OK   0x01
#define ENGINE_OIL_TEMP_OK   0x02
#define ENGINE_START   0x04
OS_FLAG_GRP *EngineStatus;
void Task (void *p_arg)
{
        INT8U err;
        OS_FLAGS value;
        (void)p_arg;
        for (;;) {    value = OSFlagPend(EngineStatus,
                        ENGINE_OIL_PRES_OK + ENGINE_OIL_TEMP_OK,
                        OS_FLAG_WAIT_SET_ALL + OS_FLAG_CONSUME,
                        10,
                        &err);
```

OSFlagPend()函數範例

```
        switch (err) {
                case OS_ERR_NONE:
                        /* Desired flags are available */
                                break;
                case OS_ERR_TIMEOUT:
                        /* The desired flags were NOT available before .. */
                        /* .. 10 ticks occurred */
                                break;
                        }
        ...
        }
}
```

OSFlagAccept()

```
OS_FLAGS OSFlagAccept(OS_FLAG_GRP *pgrp,
                      OS_FLAGS flags,
                      INT8U wait_type,
                      INT8U *perr);
```

- 參數: 其參數值與OSFlagPend()函數除了timeout參數外，其餘皆相同。

OSFlagAccept()函數範例

```
#define ENGINE_OIL_PRES_OK   0x01
#define ENGINE_OIL_TEMP_OK   0x02
#define ENGINE_START   0x04
OS_FLAG_GRP *EngineStatus;

void Task (void *p_arg)
{           INT8U err;
            OS_FLAGS value;
            (void)p_arg;
            for (;;) {     value = OSFlagAccept(EngineStatus,
                                   ENGINE_OIL_PRES_OK + ENGINE_OIL_TEMP_OK,
                                   OS_FLAG_WAIT_SET_ALL,
                                   &err);
```

OSFlagAccept()函數範例

```
                switch (err) {
                        case OS_ERR_NONE:
                                /* Desired flags are available */
                                 break;
                        case OS_ERR_FLAG_NOT_RDY:
                                /* The desired flags are NOT available */
                                break;
                }
        ...
}
```

Part 8
uC/OS-II記憶體管理

記憶體管理相關之函數

相關之函數	OS_CFG.H需設定之參數
OSMemCreate()	
OSMemGet()	
OSMemPut()	

- 是採用Partition方式的記憶體管理
  - 每個Partition內為多個相同大小的記憶體區塊
  - 利用OSMemCreate()函數設定對所宣告的記憶Partition管理

OSMemCreate ()函數範例

```
OS_MEM *CommMem;
INT32U CommBuf[16][32];

void main (void)
{
        INT8U err;
        OSInit(); /* Initialize µC/OS-II */
        ...
        CommMem = OSMemCreate(&CommBuf[0][0], 16, 32 * sizeof(INT32U), &err);
        ...
        OSStart(); /* Start Multitasking */
}
```

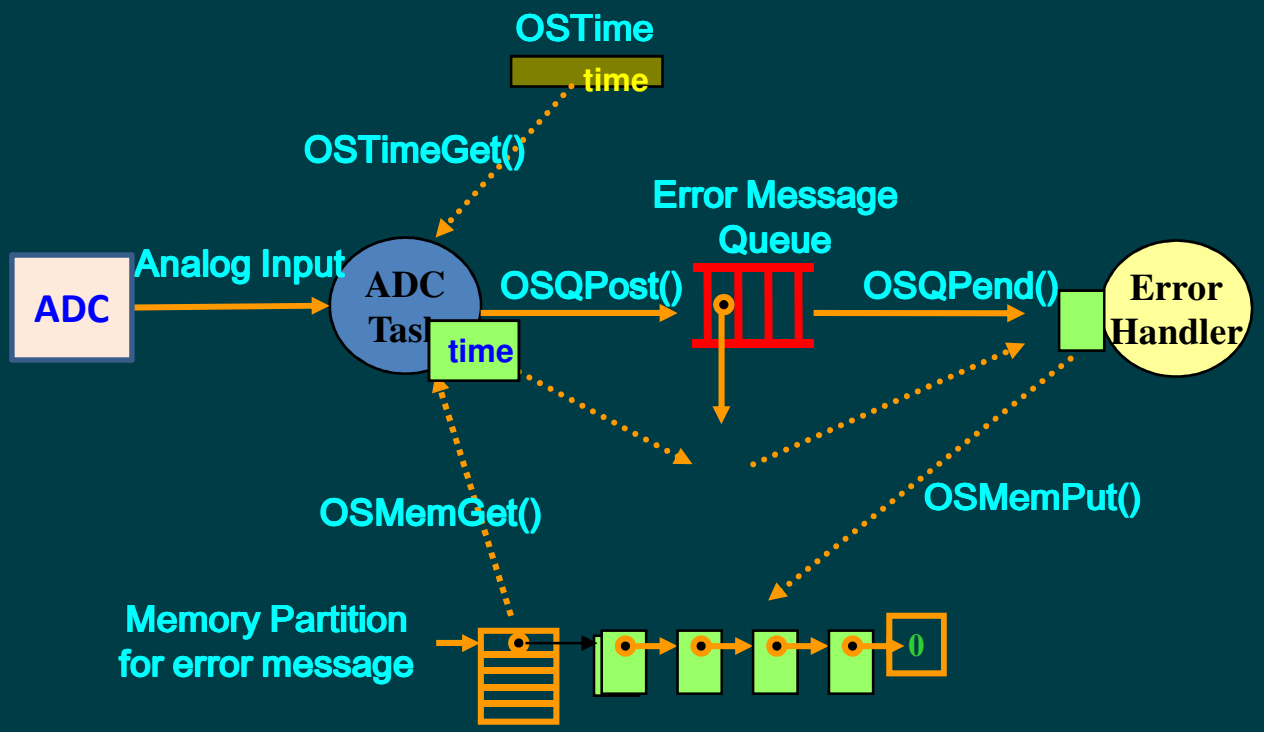
OSMemGet ()函數範例

```
OS_MEM *CommMem;
void Task (void *p_arg)
{       INT8U *pmsg;
        (void)p_arg;
        for (;;) {
                pmsg = OSMemGet(CommMem, &err);
                if (pmsg != (INT8U *)0) {
                ... /* Memory block allocated, use it. */
                }
                ...
            }
}
```

OSMemPut ()函數範例

```
OS_MEM *CommMem;
INT8U *CommMsg;
void Task (void *p_arg)
{
        INT8U err;
        (void)p_arg;
        for (;;) {
                err = OSMemPut(CommMem, (void *)CommMsg);
                if (err == OS_ERR_NONE) {
                  .... /* Memory block released */
                }
                …
              }
}
```

Memory Partition應用

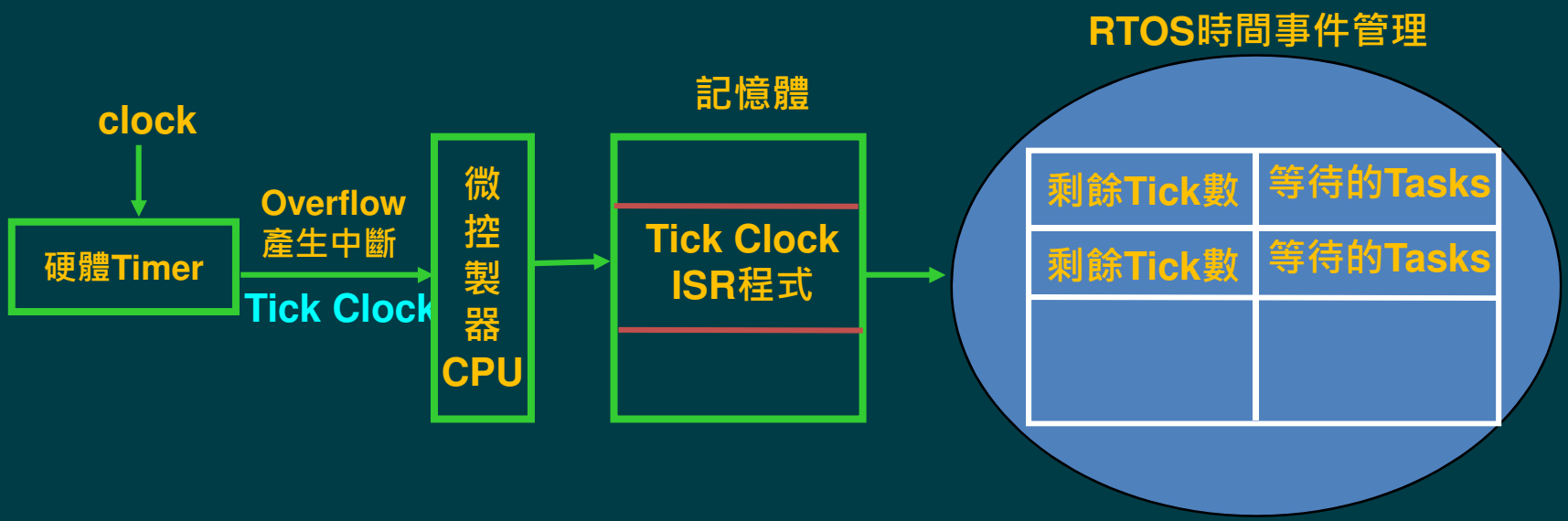


Part 9
uC/OS-II時間管理

RTOS為何需要Timer?

- Round-Robin排程器，每個Task執行一固定slice time即context switch，換下一個Task執行
- Task要周期性的polling周邊裝置之狀況
- Task在設定時間內，等待某事件發生，若時間已到，事件尚未發生，則Task將不繼續等待

RTOS時間事件管理


clock
硬體Timer
Overflow
產生中斷
Tick Clock
微控製器CPU
記憶體
Tick Clock
ISR程式
RTOS時間事件管理
剩餘Tick數
等待的Tasks
剩餘Tick數
等待的Tasks

時間管理

- Tick Clock為RTOS一個重要的時間參考依據
  - 為一週期性的中斷(Tick)

時間管理相關之函數

相關之函數	OS_CFG.H需設定之參數
OSTimeDly()	
OSTimeDlyHMSM()	OS_Q_Post_EN = 1
OSTimeGet()	OS_Q_Post_OPT_EN = 1
OSTimeSet()	

OSTimeDly ()函數範例

```
void TaskX (void *p_arg)
{
for (;;) {
        …
        OSTimeDly(10);  /* Delay task for 10 clock ticks */
        …
        }
}
```

OSTimeDlyHMSM ()函數範例

```
void TaskX (void *p_arg)
{
for (;;) {
        …
        OSTimeDlyHMSM(0, 0, 1, 0); /* Delay task for 1 second */
        …
        }
}
```

OSTimeGet ()函數範例

```
void TaskX (void *p_arg)
{
INT32U  clk;
for (;;) {
        …
        clk = OSTimeGet();
        …               /* Get current value of system clock */
        }
}
```

OSTimeSet ()函數範例

```
void TaskX (void *p_arg)
{
for (;;) {
        …
        OSTimeSet(0L);    /* Reset the system clock */
        …
        }
}
```