嵌入式系統

單元二：嵌入式系統之程式架構

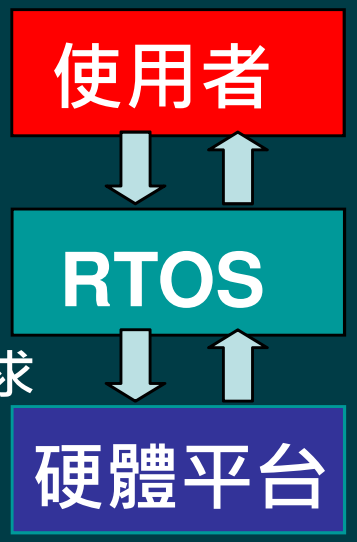
授課教師：雲林科技大學 張慶龍 老師

單元學習目標與大綱

- 單工程式-循環執行
- 單工程式-foreground/background
- 多工程式-nesting/non-nesting中斷
- 多工程式-程式狀態

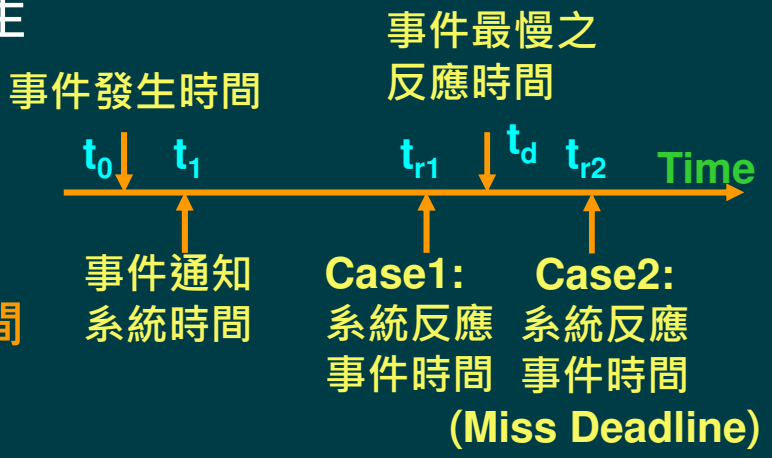
RTOS之嵌入式系統架構

- 硬體平台
  - Ex: Cortex M3 開發板
  - 提供計算處理、Timer/ADC等周邊、通訊、I/O
- 即時性作業系統 (Real-Time Operation System)
  - Ex: uC/OS-II RTOS
  - 提供 硬體資源管理、多工程式排程、滿足即時性要求
- 使用者程式
  - 系統所需之相關多工程式



基本名詞

- 事件(Event)
  - 周邊元件產生: 如網路介面收到一個封包之事件
  - 外部感測器所產生: 如溫度超過某一臨界值之事件
  - CPU可藉由polling或中斷瞭解事件的發生
- Deadline
  - 一事件所允許之最慢反應時間
- Miss Deadline
  - 系統對事件之反應時間超過Deadline時間



即時性系統之分類

- 即時性系統依操作反應時間與Deadline關係，可分為
  - **Hard Real-time**
    - Miss Deadline時，會造成系統嚴重危害，如飛機偵測到飛彈時，若Miss Deadline，將被飛彈打到，造成嚴重後果
  - **Soft Real-time**
    - Miss Deadline時，會讓人覺得系統效能不好，但並不影響系統運作的正確性，如手機對按鍵的反應時間

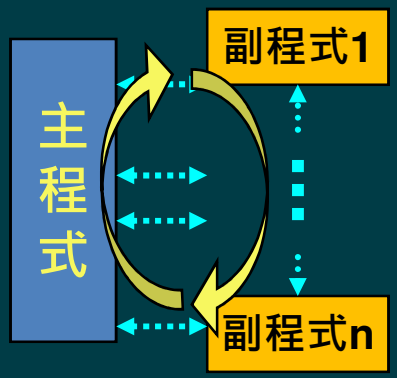
RTOS即時性作業系統

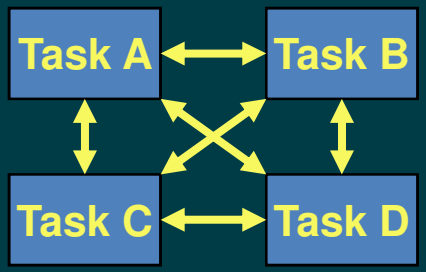
- 大部分的嵌入式系統皆為real-time系統
- 大部分的real-time系統同時包含有hard real-time事件與soft real-time事件
- 即時性作業系統為了符合不同的deadline要求
  - 多工程式架構(Multi-task)
  - Event driven
  - 優先權排程機制 (priority-based scheduling)
    - 高real-time要求的工作(task)設定較高執行優先權

Part 1
單工程式-循環執行

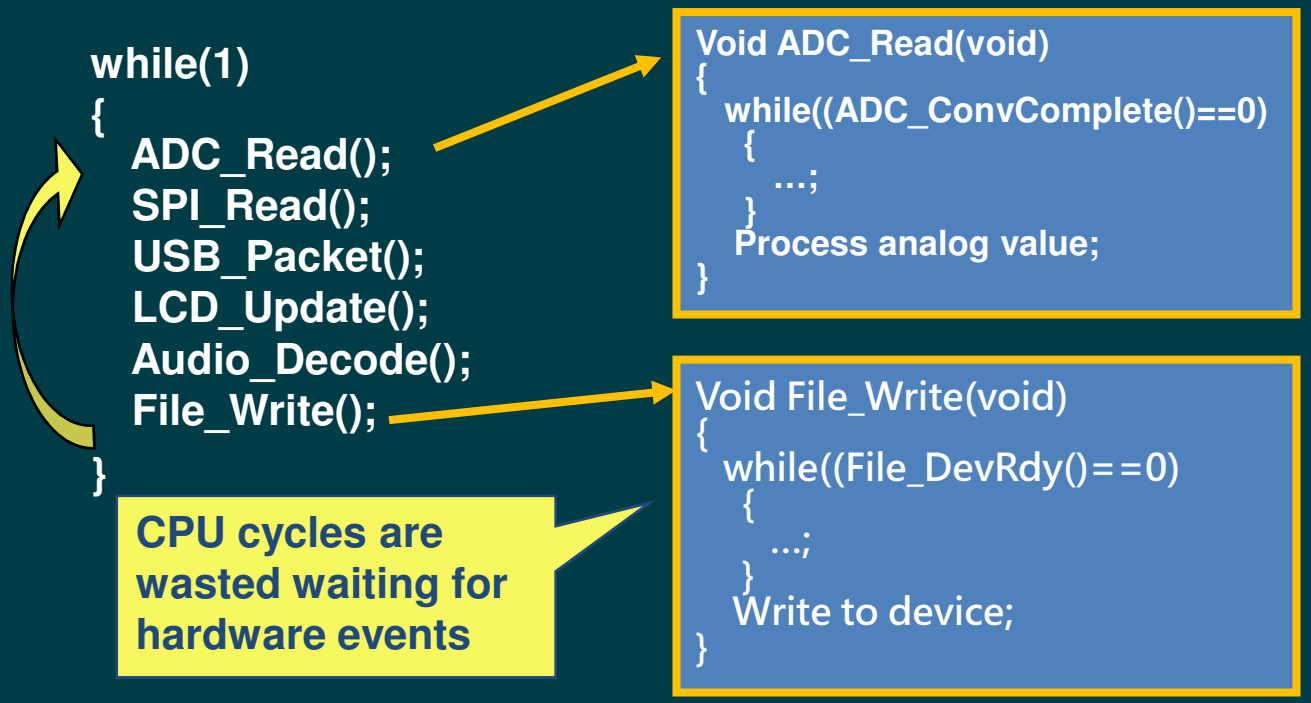
嵌入式系統之程式架構

- 單工程式架構(single task programming)
  - Cyclic Execution(循環執行)
  - Cyclic Execution + Interrupt ISR(循環執行+中斷)
    - Non-nesting Interrupt(無巢狀中斷)
    - Nesting Interrupt(允許巢狀中斷)
  - 執行順序事先決定➔即時反應效果較差
- 多工程式架構(Multi-tasking programming)
  - 事件觸發執行
  - 執行順序是依實際事件發生的順序決定
    - ➔具有較佳即時反應



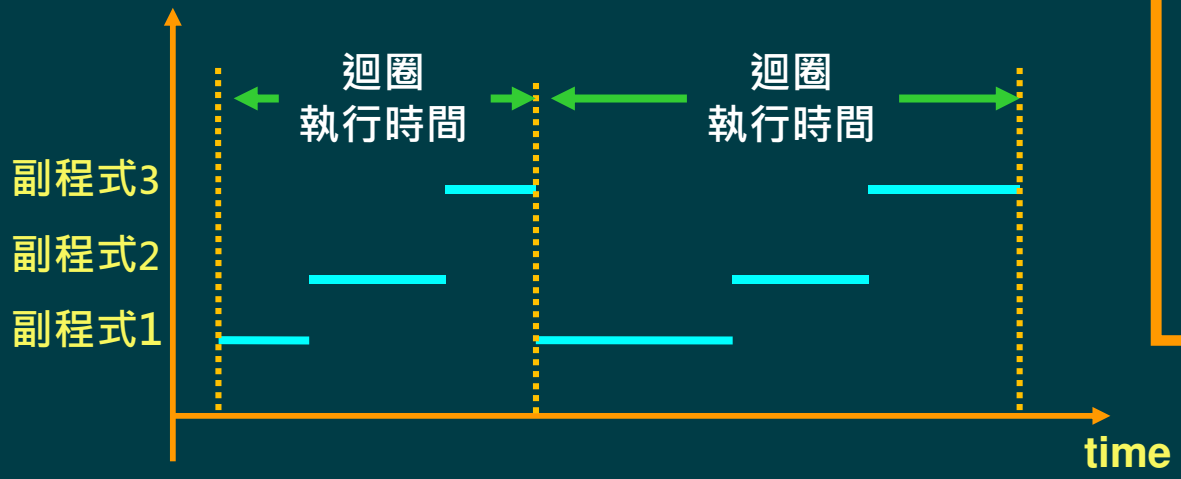


單工程式架構- 循環執行(Cyclic Execution)



循環執行(Cyclic Execution)

- 程式執行順序是寫程式的人事先決定
- 一次迴圈之執行時間並不確定




```
main( )
{

副程式1( ... );
副程式2( ... );
副程式3( ... );

}
```

以counter控制循環架構下副程式執行率

```
Int main(void)
{
  unsigned short I;
     i=0;
     while (1)
     {
       ADC_Read();
       if((i%2)==0)
            SPI_Read();
       USB_Packet()
       LCD_Update();
       if((i%5)==0
          Audio_Decode();
       File_Write();
       i++;
     }
}
```

每個副程式可以有不同的執行速率

個別副程式的執行速率是依迴圈執行間而定

Question?

- 上一張投影片中
  1. 主程式迴圈執行前3次之函數之呼叫順序為何?

     ADC_Read()->SPI_Read()->USB_Packet()->LCD_Update()->Audio_Decoder()->
     File_Write()->ADC_Read()->USB_Packet()->LCD_Update->File_Write()->
     ADC_Read()->SPI_Read()->USB_Packet()->LCD_Update()->File_Write()->
  2. 若主程式迴圈最長執行時間為2秒，則ADC最差之轉換頻率為何?

     0.5 Hz

Part 2
單工程式-foreground/background

foreground/background架構

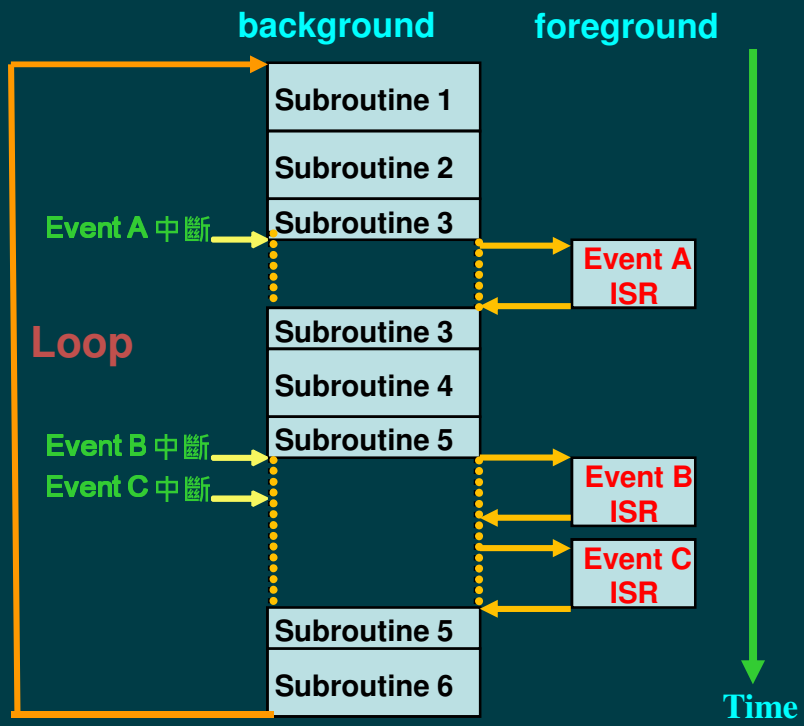
- Cyclic execution + Interrupt ISR程式所組成
  - Interrupt 之ISR程式稱為 foreground程式
    - 只要一發生中斷，即會去執行其ISR程式
    - 即時性要求較高的事件，需放於foreground程式
  - Cyclic execution loop稱為background程式
    - 沒有發生中斷時，CPU一直執行此loop程式
    - Cyclic execution loop的事件反應時間最長為loop最長執行時間
    - 執行順序事先已決定
  - 系統負擔較小

foreground/background架構

- Non-nesting interrupt

```
main( )
{
subroutine1( ... );
subroutine2( ... );
subroutine3( ... );
}
```

Event A ISR程式	Event B ISR程式	Event C ISR程式



優先權: C > B > A

Event C需等Event B執行完後才能執行

foreground/background架構

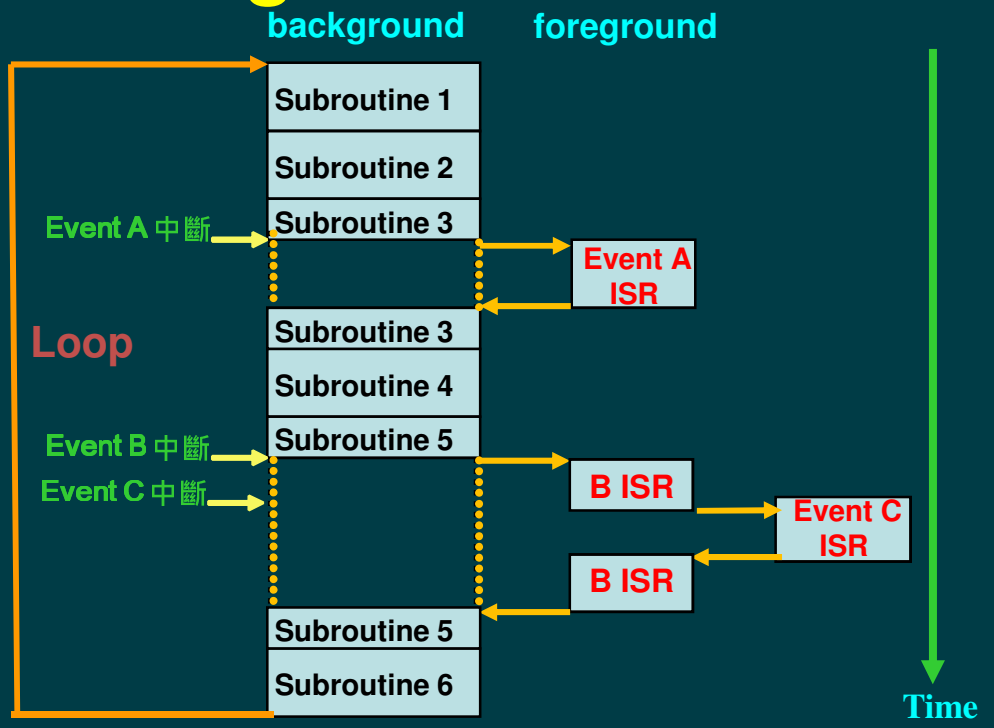
- Nesting interrupt

```
main( )
{
subroutine1( ... );
subroutine2( ... );
subroutine3( ... );
}
```

Event A ISR程式

Event B ISR程式

Event C ISR程式



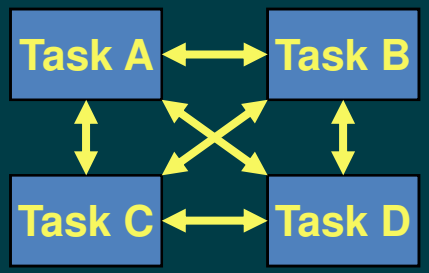
優先權: C > B > A

Event C可立即得到處理➔有較好的即時性

Part 3
多工程式之程式狀態

多工程式環境架構

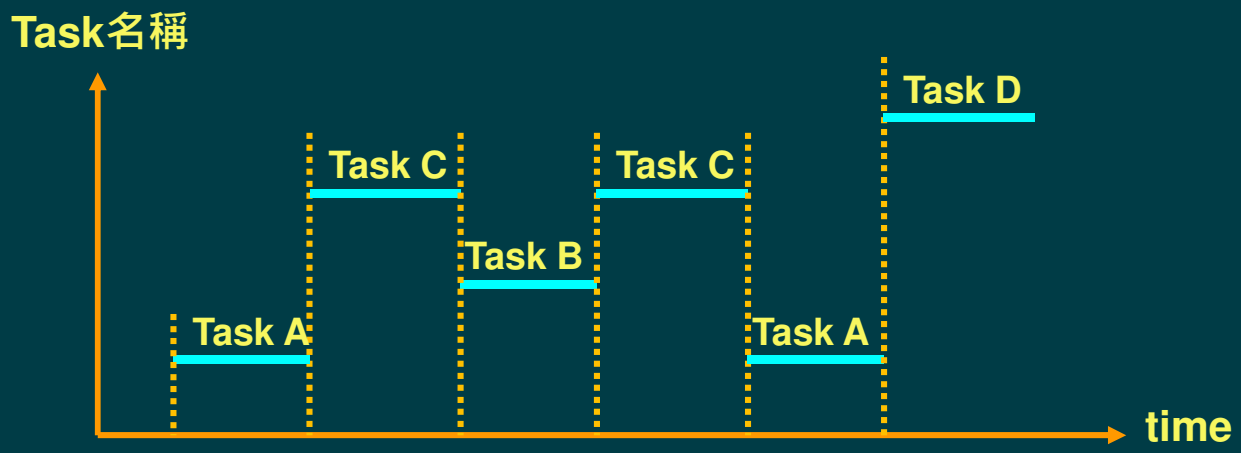
- 一個嵌入式系統功能是由多個Task分工而成
  - Ex: Task A、Task B、Task C與Task D
- Task A、Task B、Task C與Task D為互相獨立之程式(program)
  - 有自己的變數、自己的堆疊(stack)
  - 彼此會互相溝通/同步執行
- 每個Task為無窮迴圈
- Task的執行與否是依相對應的事件(Event)是否發生而決定
  - Task的執行順序是依事件發生的順序決定，而不是程式撰寫者事先決定

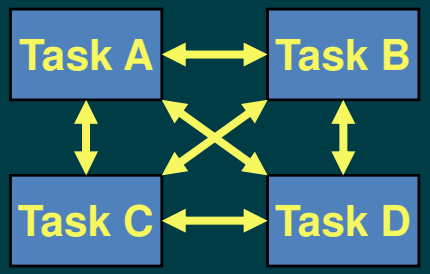



```
task()
{
  while(1)
  {
    等待事件發生;
    執行服務該事件所需之工作
        內容
  }
}
```

等待事件發生才會執行

多工程式執行順序





依事件的發生與Task優先權決定執行的順序

Task可能的狀態

- **Ready state**
  - 可以執行，等待排程器排入CPU執行
- **Running state**
  - 正在CPU內執行
- **Waiting state**
  - 等待事件發生後才可執行

Example


一開始,產生
此4個Tasks
Task4
Task3
Task2
Task1
Ready
state
Ready Queue
Task4
Task3
Task2
Task1
Task4
Task3
Task2
Task4
Task3
Task2
Task1
Task4
Task3
Task2
Task1
Task4
Task3
Task4
Time
MCU
Running state
Task1
Task2
Task1
Task2
Task3
Task4
等待事件
Task1
事件產生
等待事件
等待事件
等待事件
Waiting
List
Task1
Task1
Task2
Task1
Task3
Task2
Task1