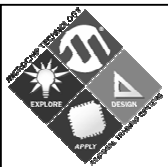




## 201ASP PIC16Fxxx 中階系列 PIC® 周邊配置與組合語言撰寫



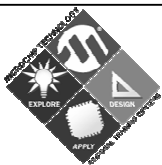
### 目標

- 學習完本課程後，將能够：
  - 理解基本PIC®周邊及相關的暫存器
  - 具有初始化中階系列微控器周邊的 **“動手實驗”** 經驗
  - 能實現本課程中未涉及到的周邊
  - 了解中斷和查詢的技巧
  - 從頭開始撰寫自己的應用程式



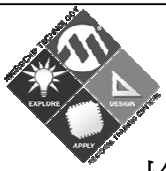
## 本課程可獲得最大收益

- **理想情况** 應熟悉以下内容：
  - 組合語言撰寫
  - 中階系列微控器的基本指令集
  - 資料暫存器和程式記憶器的架構
  - MPLAB® IDE 開發環境的使用
  - Microchip ICD2 除錯器



## 201ASP 課程安排

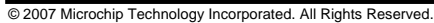
- **簡要地** 回顧中階系列微控器的架構、指令集和 MCHP 工具
- 中階PIC®微控器的中斷
  - 基本中斷練習
- 周邊討論：
  - 輸入 / 輸出腳位功能
  - 計時器
    - Timer0
    - Timer1
      - Timer1練習
    - Timer2
      - Timer2練習



## 201ASP 課程安排

- 增強型脈波量測器 /比較/ PWM 模組 (ECCP)
  - PWM 和計時比較練習
- 類比電壓比較器
- 類比轉換器 (ADC)
  - ADC練習
- 增強型通用步/非同步串列通訊 (EUSART)
  - EUSART 通訊練習
- I<sup>2</sup>C 和 SPI 通訊介面
  - I2C 通訊練習
- 其它：
  - 多中斷練習，Internal EEPROM 練習，LCD 驅動練習
- 總結

# 中階系列微控器 基本架構 和 開發工具



201ASP

Slide 7



- **最大 8K Word :**
  - **(8K x 14-bit) = 14 KB** 的程式定址空間
- **重置向量位於 0x0000**
  - 發生任何重置動作時，**PC** 將歸零到此地址
- **中斷向量位於 0x0004**
  - 發生任何中斷時，**PC** 將跳到此地址

© 2007 Microchip Technology Incorporated. All Rights Reserved.

201ASP

Slide 8



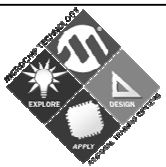
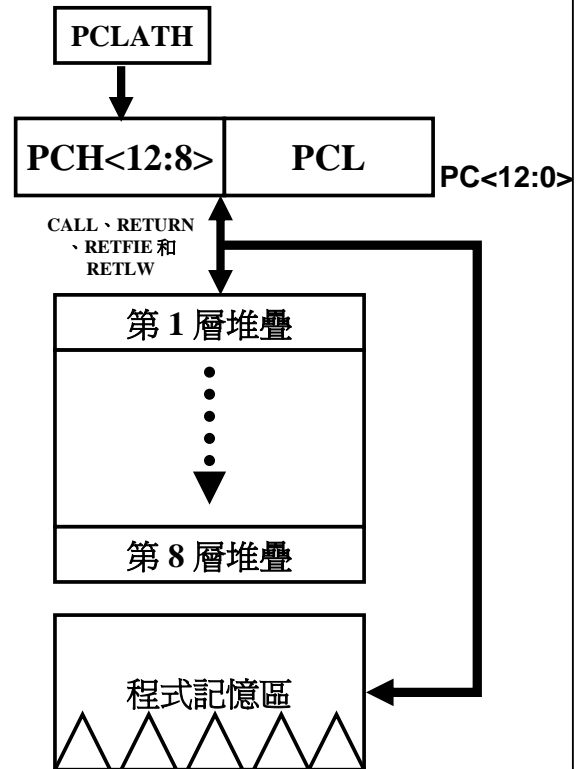
## 程式計數器 (PC) 和堆疊

### ● 13-bit 程式計數器 (PC)

- 直接操作的程式空間 - 11 bit
- 最高的兩個位元為程式頁選擇位元:
  - 通過PCLATH<4:3>兩位元更新
  - 指定程式記憶的切換頁(Page)

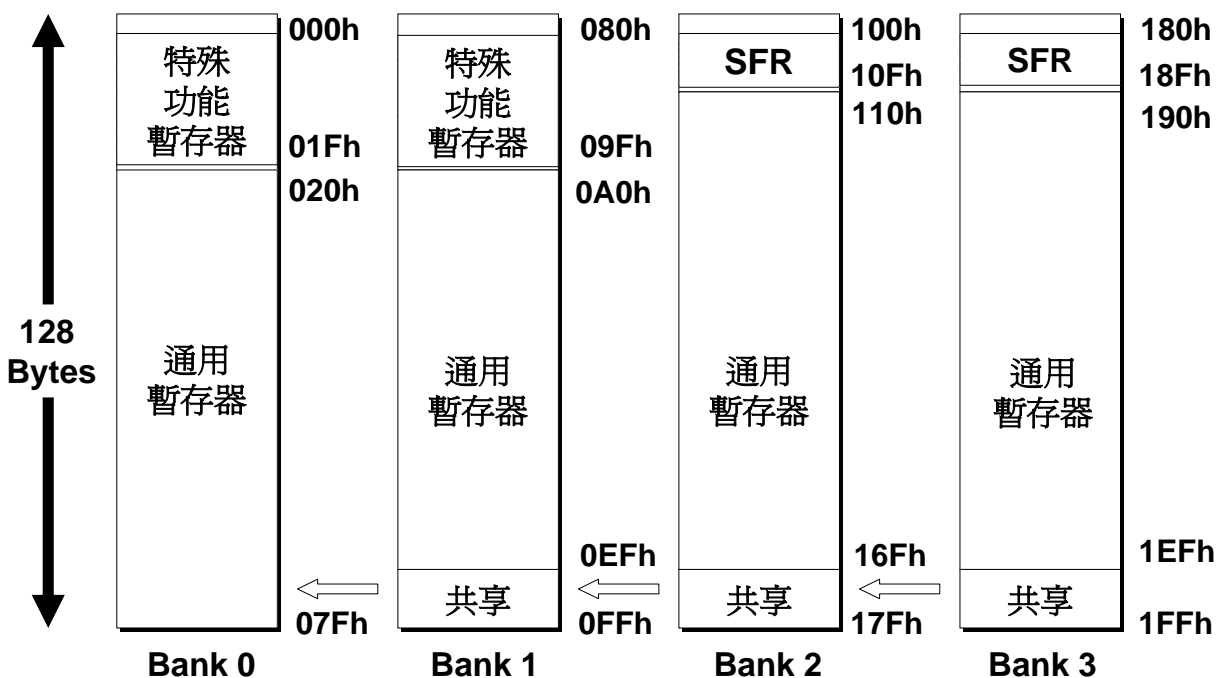
### ● 8 層深堆疊

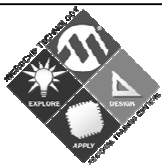
- 暫存 PC 的內容
  - 推入堆疊
    - 呼叫副程式 / 中斷
  - 彈出堆疊
    - RETURN、RETFIE 和 RETLW



## Data Memory Map

以 PIC16F887 做說明，實際 RAM 大小隨元件有所變化。



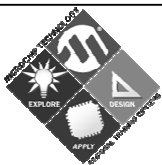


## 特殊功能暫存器 (SFR)

- 暫存器採一般 **RAM** 的概念
- 存取方式與存取其它通用暫存器一樣簡單
- 某些暫存器可跨越所有 **Data Bank** (**PCLATH**, **STATUS**, **INTCON**等)

|               |     |               |     |
|---------------|-----|---------------|-----|
| <b>PORTB</b>  | 06h | <b>TRISB</b>  | 86h |
| <b>PORTC</b>  | 07h | <b>TRISC</b>  | 87h |
| <b>PORTD</b>  | 08h | <b>TRISD</b>  | 88h |
| <b>PORTE</b>  | 09h | <b>TRISE</b>  | 89h |
| <b>PCLATH</b> | 0Ah | <b>PCLATH</b> | 8Ah |
| <b>INTCON</b> | 0Bh | <b>INTCON</b> | 8Bh |
| <b>PIR1</b>   | 0Ch | <b>PIE1</b>   | 8Ch |
| <b>PIR2</b>   | 0Dh | <b>PIE2</b>   | 8Dh |

**Bank0**                      **Bank1**



## STATUS 暫存器

- 包含：
  - ALU的算術運算狀態
  - 重置狀態
  - RAM 的 **BANK** 選擇

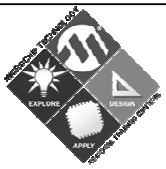


| <b>RP1</b> | <b>RP0</b> |              |
|------------|------------|--------------|
| <b>0</b>   | <b>0</b>   | <b>BANK0</b> |
| <b>0</b>   | <b>1</b>   | <b>BANK1</b> |
| <b>1</b>   | <b>0</b>   | <b>BANK2</b> |
| <b>1</b>   | <b>1</b>   | <b>BANK3</b> |

暫存器暫存區選擇：  
(用於間接定址-FSR)

**1 = Bank 2和Bank 3**

**0 = Bank 0和Bank 1**



# 指令集總覽表

- 35 個 Single Word 指令
- 除程式分歧指令外，所有指令都是單週期指令
- 可歸內下列 3 種類型操作：

## 位元組操作類型指令

|        |      |                              |
|--------|------|------------------------------|
| ADDWF  | f, d | Add W and f                  |
| ANDWF  | f, d | AND W with f                 |
| CLRF   | f    | Clear f                      |
| CLRW   | -    | Clear W                      |
| COMF   | f, d | Complement f                 |
| DECf   | f, d | Decrement f                  |
| DECFSZ | f, d | Decrement f, Skip if 0       |
| INCF   | f, d | Increment f                  |
| INCFSZ | f, d | Increment f, Skip if 0       |
| IORWF  | f, d | Inclusive OR W with f        |
| MOVF   | f, d | Move f                       |
| MOVWF  | f    | Move W to f                  |
| NOP    | -    | No Operation                 |
| RLF    | f, d | Rotate Left f through Carry  |
| RRF    | f, d | Rotate Right f through Carry |
| SUBWF  | f, d | Subtract W from f            |
| SWAPF  | f, d | Swap nibbles in f            |
| XORWF  | f, d | Exclusive OR W with f        |

## 位元操作類型指令

|       |      |                           |
|-------|------|---------------------------|
| BCF   | f, b | Bit Clear f               |
| BSF   | f, b | Bit Set f                 |
| BTFSC | f, b | Bit Test f, Skip if Clear |
| BTFSS | f, b | Bit Test f, Skip if Set   |

## 立即數和控制操作類型指令

|        |   |                             |
|--------|---|-----------------------------|
| ADDLW  | k | Add literal and w           |
| ANDLW  | k | AND literal with w          |
| CALL   | k | Call Subroutine             |
| CLRWDt | - | Clear Watchdog Timer        |
| GOTO   | k | Go to address               |
| IORLW  | k | Inclusive OR literal with w |
| MOVLW  | k | Move literal to w           |
| RETFIE | - | Return from interrupt       |
| RETLW  | k | Return with literal in w    |
| RETURN | - | Return from Subroutine      |
| SLEEP  | - | Go into Standby mode        |
| SUBLW  | k | Subtract w from literal     |
| XORLW  | k | Exclusive OR literal with w |

全部在一張投影片中!!!

HANDS-ON

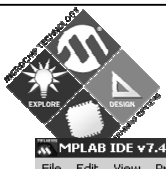
Training

PIC<sup>®</sup> 開發工具

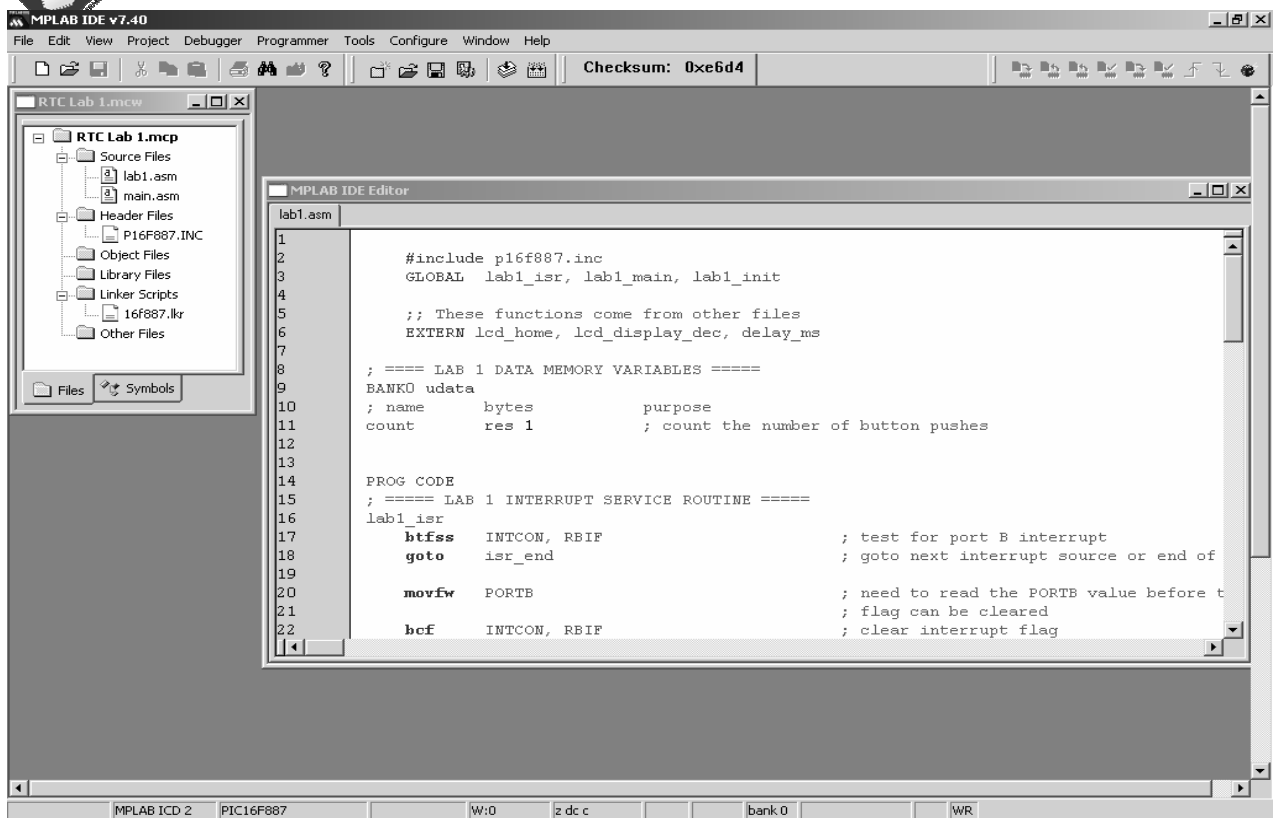


## PIC®：開發工具 MPLAB® IDE

- **MPLAB® IDE**：整合式開發環境
- 它整合了多種 **Microchip** 和 **Third-Party** 工具
  - 程式編輯器
  - 交叉編譯器
  - 組合語言編譯器
  - 軟體模擬器 (MPLAB SIM)、線上除錯器(ICD) 和線上模擬器(ICE)
  - 燒錄器



## PIC®：開發工具 MPLAB® IDE



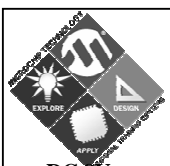
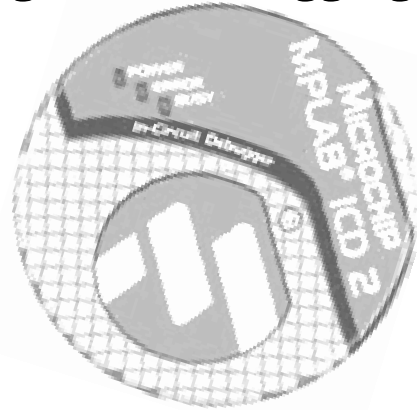




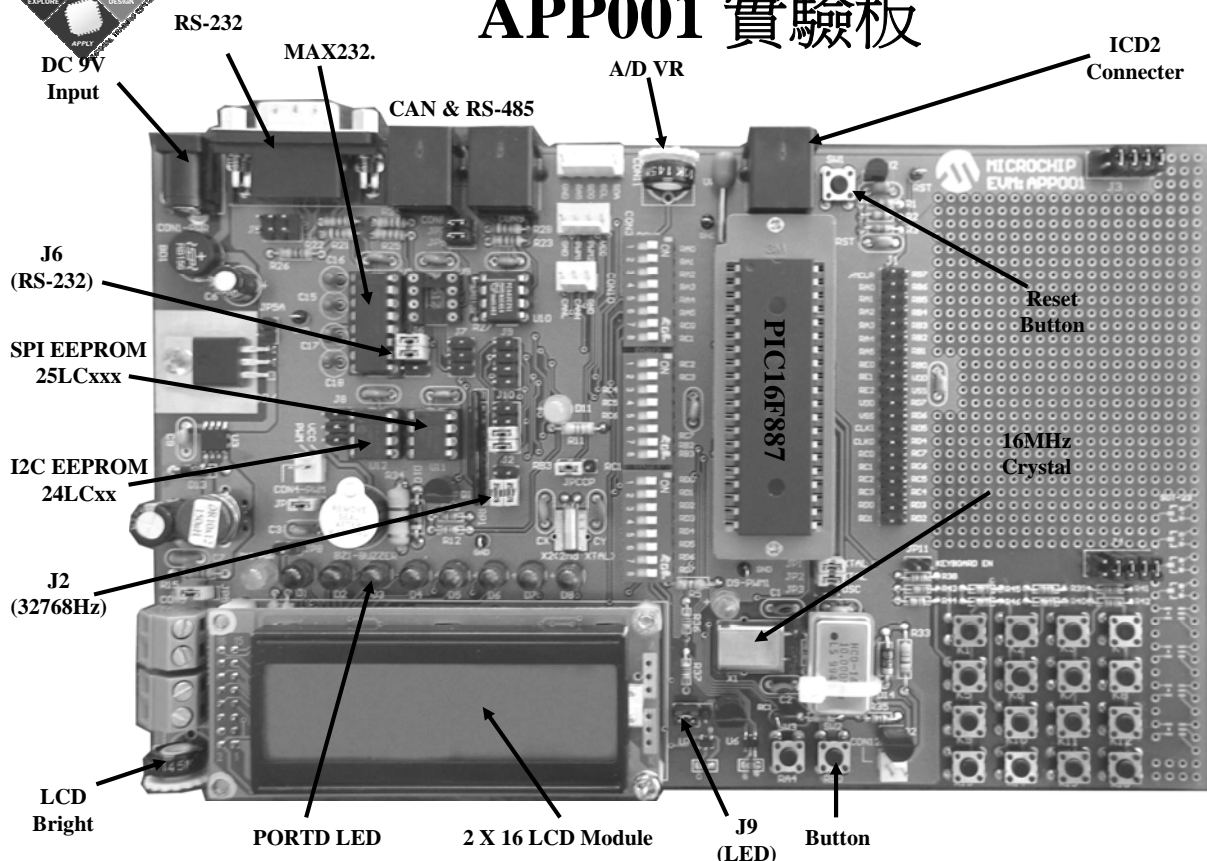
## ICD 2 (線上除錯器)

- **MPLAB® ICD 2** 是一款低價位的即時除錯器和燒錄器。提供以下功能：

- 讀/寫目標板 PIC 微控器的**Program**和**EEDATA**
- 即時背景除錯 (**Real time background debugging**)
- 清除、燒錄程式並比對
- 燒錄 **Configuration Bits**

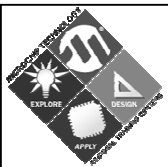


## APP001 實驗板



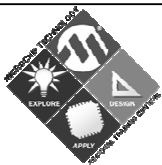


## 中斷



## 查詢和中斷

- 發生特殊事件時，我們總是希望處理器能執行任務
- 有兩種方法用於檢查事件是否發生：
  - 查詢 (Polling)：
    - 在程式執行的各個時間點連續對事件進行檢查
  - 中斷(Interrupt)：
    - 僅在事件發生時暫時“中斷”正常的程式執行



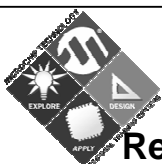
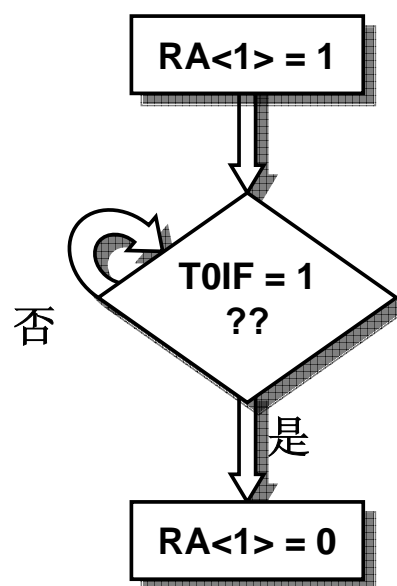
## 查詢

```
bsf    PORTA,1    ;Set bit 1 of
                  ;PORTA

btfss  INTCON, T0IF ;Check Timer0
                  ;interrupt flag in
                  ;INTCON and
                  ;skip the next
                  ;instruction if it
                  ;is set

goto   $-1        ;Go back to
                  ;previous
                  ;instruction

bcf    PORTA,1    ;Clear bit 0 of
                  ;PORTA
```



## 中斷

Reset

```
code 000h
goto Start
```

```
=====
int_vector code 004h
```

retfie

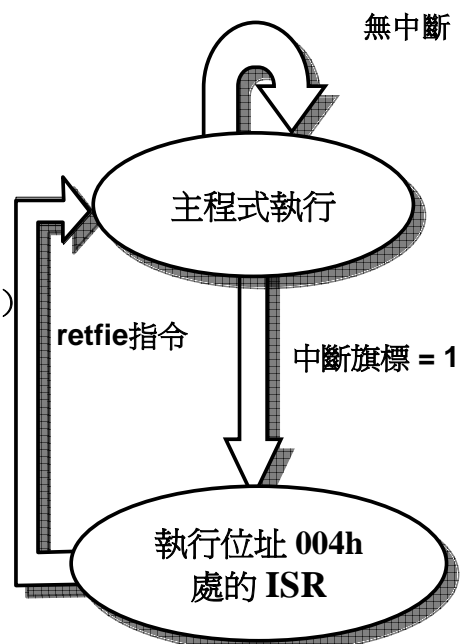
```
-----
;return from
;interrupt
```

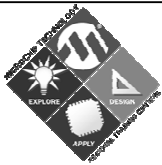
```
=====
main_prog code
```

```
Start ;start label for main code
```

```
end
```

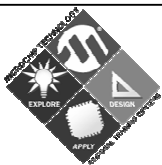
```
-----
} 主程式程式
```





# 打開中斷功能

- 必須告知處理器程式中將使用中斷
  - 使用具有中斷致能位元的幾個暫存器：
    - 中斷控制暫存器 (**INTCON**)
    - 周邊中斷致能暫存器 1 (**PIE1**)
    - 周邊中斷致能暫存器 2 (**PIE2**)



## INTCON (中斷控制暫存器)

### 中斷控制位元

GIE : 中斷致能總控制位元

\*必須設1以開啓PIC®微控器中的任何中斷

PEIE : 周邊中斷致能位元

\*必須設1以使用周邊中斷

TOIE : Timer0 溢位中斷致能位元

INTE : RB0/INT 外部中斷致能位元

RBIE : PORTB 電位改變中斷致能位元



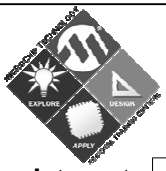
\*即使未設定其中斷致能位元，  
事件發生時也會被設為 1 !

### 中斷旗標位元

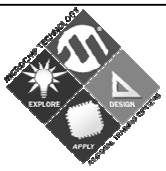
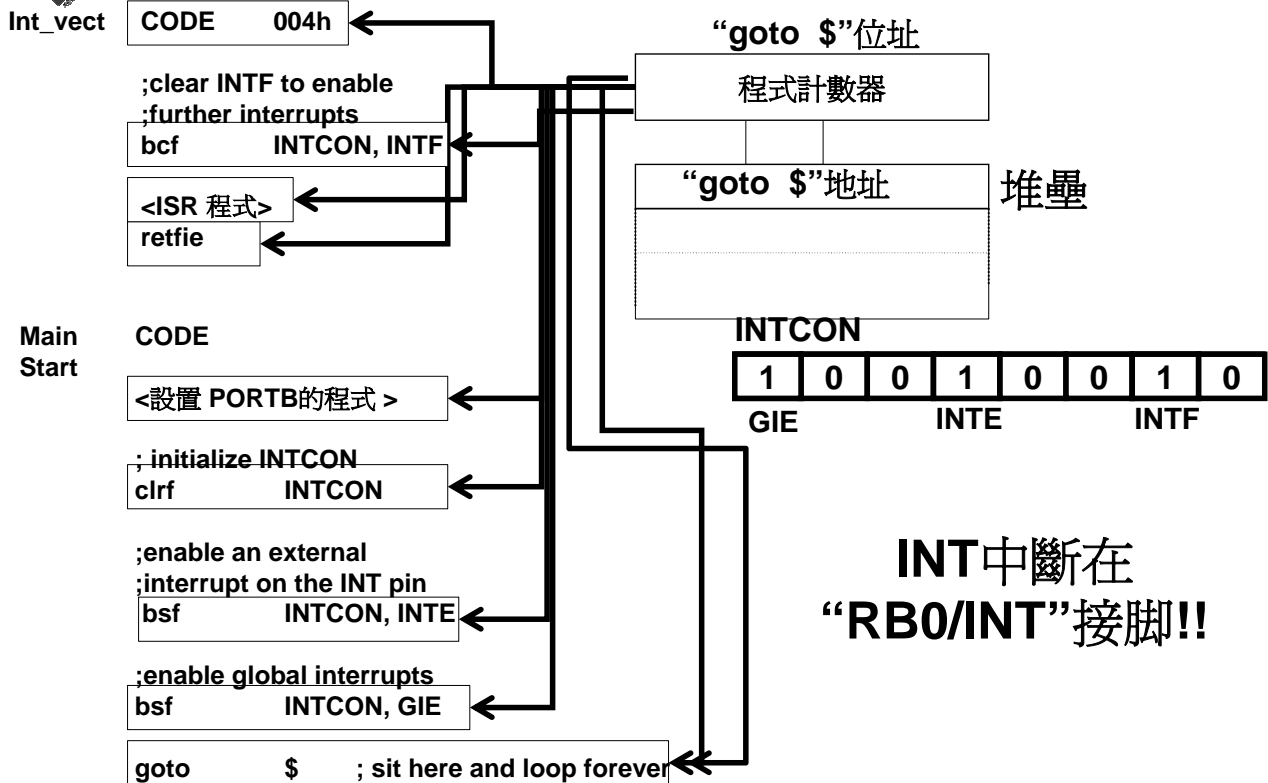
TOIF : Timer0 溢位中斷旗標位元

INTF : RB0/INT 外部中斷旗標位元

RBIF : PORTB 電位改變中斷旗標位元



## INT 中斷動作說明

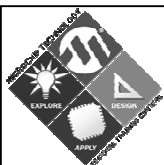


## 周邊中斷

- “致能” 特定周邊中斷的兩個暫存器：
  - 周邊中斷致能暫存器 1 (PIE1)
  - 周邊中斷致能暫存器 2 (PIE2)
- 包含特定周邊中斷“旗標”的兩個暫存器：
  - 周邊中斷請求暫存器 1 (PIR1)
  - 周邊中斷請求暫存器 2 (PIR2)

\*即使未設定其中斷致能位元，  
事件發生時相對應的旗標也會被設為 1

!



## PIE1和PIR1暫存器\*

### PIE1暫存器（中斷致能控制）

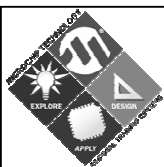
|  |      |      |      |       |        |        |        |
|--|------|------|------|-------|--------|--------|--------|
|  | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE |
|--|------|------|------|-------|--------|--------|--------|

### PIR1暫存器（中斷旗標）

|  |      |      |      |       |        |        |        |
|--|------|------|------|-------|--------|--------|--------|
|  | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF |
|--|------|------|------|-------|--------|--------|--------|

| 致能位元   | 旗標位元   | 條件                         |
|--------|--------|----------------------------|
| ADIE   | ADIF   | ADC 轉換完成                   |
| RCIE   | RCIF   | EUSART 接收緩衝器接收到資料          |
| TXIE   | TXIF   | EUSART 發送緩衝器傳送完畢           |
| SSPIE  | SSPIF  | I <sup>2</sup> C™ 或 SPI 中斷 |
| CCP1IE | CCP1IF | Timer1 暫存器脈波量測器或比較匹配       |
| TMR2IE | TMR2IF | Timer2 值和 PR2 周期值匹配        |
| TMR1IE | TMR1IF | Timer1 暫存器溢位               |

\*請查看資料手冊



## PIE2和PIR2暫存器

### PIE2暫存器（中斷致能控制）

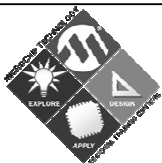
|        |      |      |      |       |         |  |        |
|--------|------|------|------|-------|---------|--|--------|
| OSCFIE | C2IE | C1IE | EEIE | BCLIE | ULPWUIE |  | CCP2IE |
|--------|------|------|------|-------|---------|--|--------|

### PIR暫存器（中斷旗標）

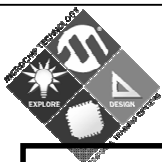
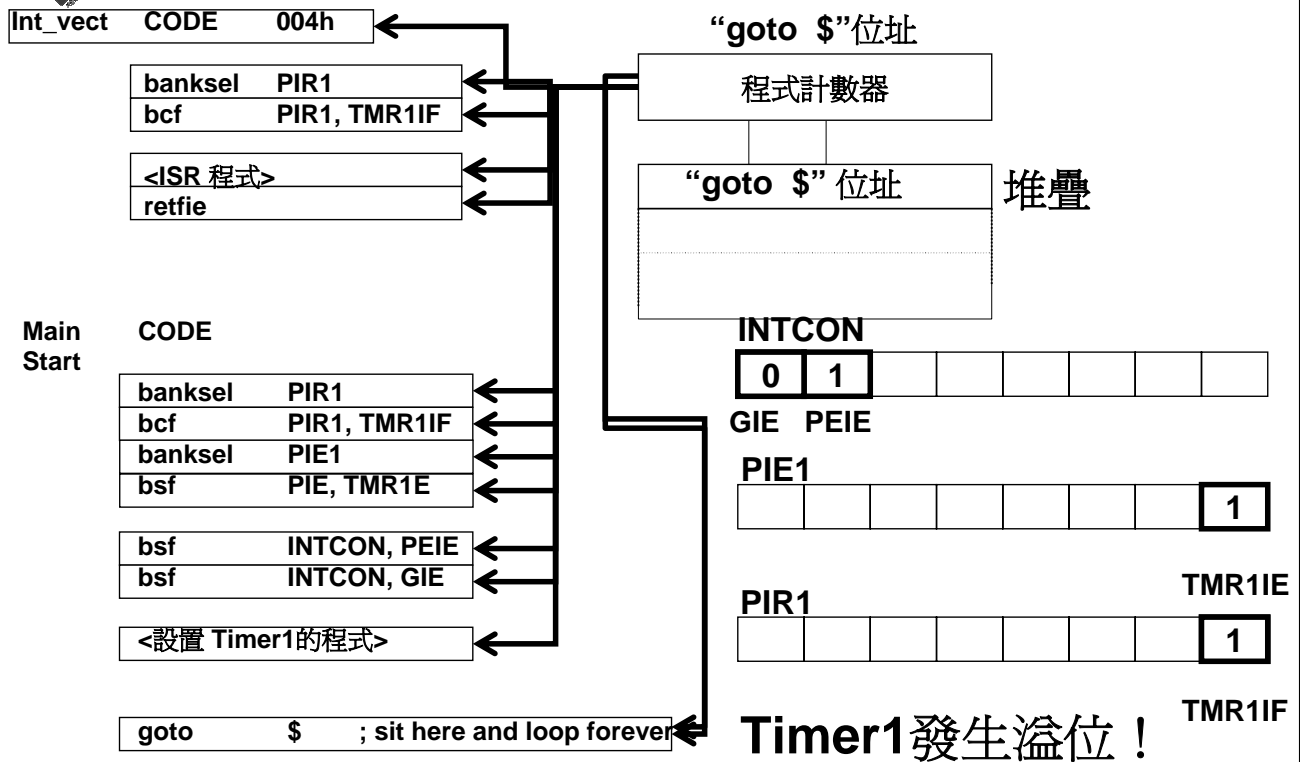
|        |      |      |      |       |         |  |        |
|--------|------|------|------|-------|---------|--|--------|
| OSCFIF | C2IF | C1IF | EEIF | BCLIF | ULPWUIF |  | CCP2IF |
|--------|------|------|------|-------|---------|--|--------|

| 致能位元    | 旗標位元    | 條件                                  |
|---------|---------|-------------------------------------|
| OSCFIE  | OSCFIF  | 系統振盪器發生故障                           |
| C2IE    | C2IF    | 比較器 2 輸出發生變化                        |
| C1IE    | C1IF    | 比較器 1 輸出發生變化                        |
| EEIE    | EEIF    | EEDATA 寫入操作已完成                      |
| BCLIE   | BCLIF   | 在 MSSP I <sup>2</sup> C™ 模式下發生了總線衝突 |
| ULPWUIE | ULPWUIF | 出現了喚醒條件                             |
| CCP2IE  | CCP2IF  | 發生了timer1 脈波量測器或比較匹配                |

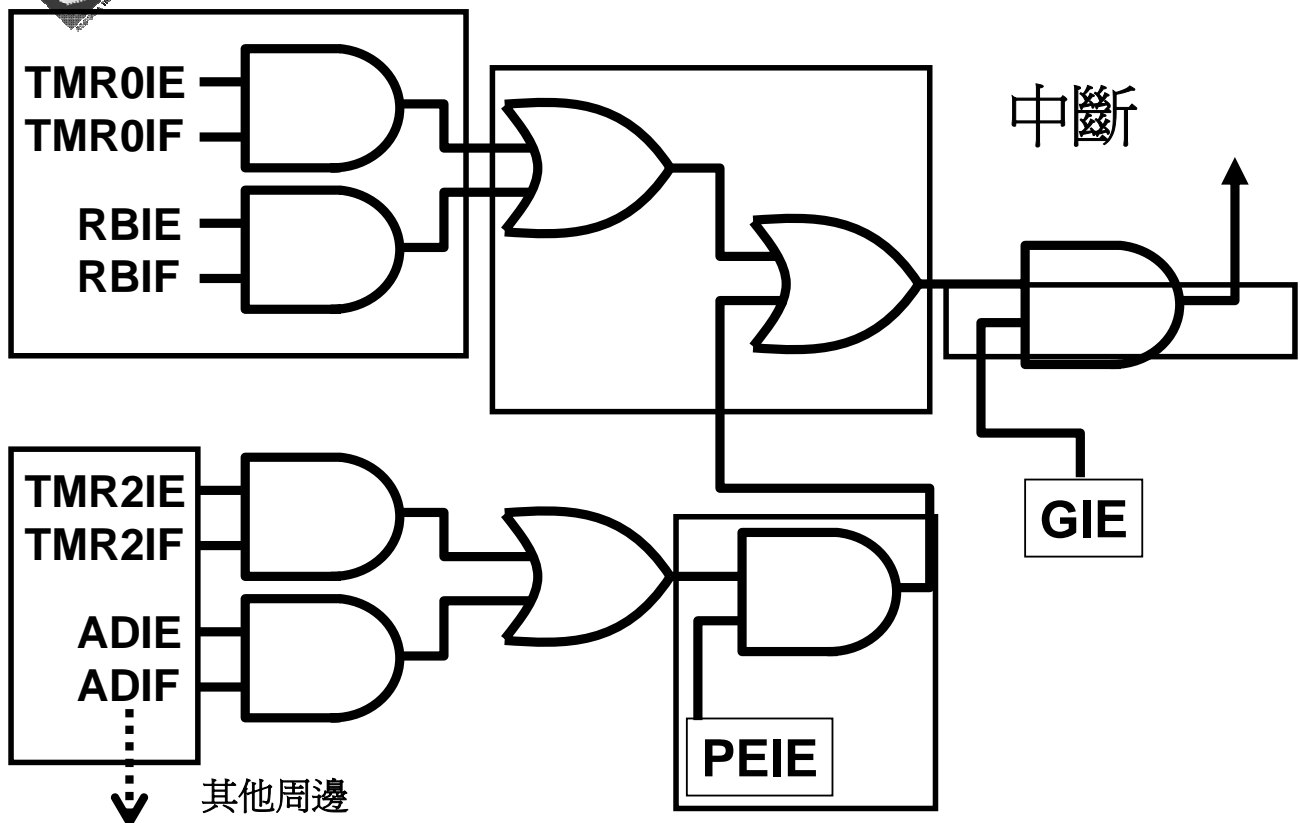
\*請查看資料手冊

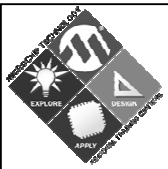


## 致能周邊中斷



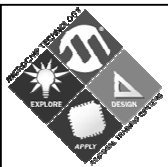
## 中斷邏輯





## 中斷響應時間

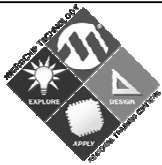
- 中斷響應時間：
  - 從中斷事件發生到指令開始執行位址 **0004h** 時所需的時間
  - 同步的中斷（通常為內部中斷，與 **Fosc** 同步）
    - 響應延遲時間為 **3Tcy**
  - 非同步的中斷（通常為外部中斷）
    - 響應延遲時間為 **3 到 3.75Tcy**



## 背景儲存(Context Saving)

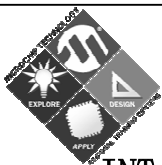
- 在中斷期間：
  - 僅保存**PC** 值（推入堆疊）
  - 在 **ISR** 中修改的暫存器值將被永久地被改變
- 需要保存的關鍵暫存器為：
  - 工作暫存器 ( **W Reg.** )
  - **Status** 暫存器
  - **PCLATH** 暫存器
  - 使用著定義的暫存器





## 中斷優先等級

- 中階 **PIC<sup>®</sup>** 微控器的所有的中斷都具有相同的優先等級
- 使用著必須執行以下操作：
  - 確定中斷來源
  - 決定中斷的處理的順序



## 中斷優先等級範例

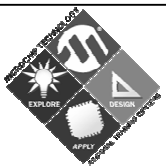
```
INT_VECTOR CODE      0x004      ; interrupt vector location
;
    movwf    temp_w      ; save WREG
    movf     STATUS, w
    movwf    temp_status  ; save STATUS register

    btfsc    INTCON, RBIF  ; PORTB change?
    call     PORTB_ISR
    btfsc    PIR1, TMR2IF  ; Timer 2 interrupt ?
    call     Timer2_ISR
    btfsc    PIR2, TMR1IF  ; Timer 1 interrupt ?
    call     Timer2_ISR

Restore_context:
    movf     temp_status, w
    movwf    STATUS      ; restore STATUS reg.
    movf     temp_w, w    ; restore WREG
    retfie              ; return from interrupt
```

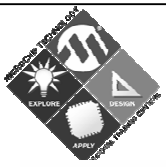


## 基本中斷練習 (Lab 1)



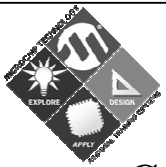
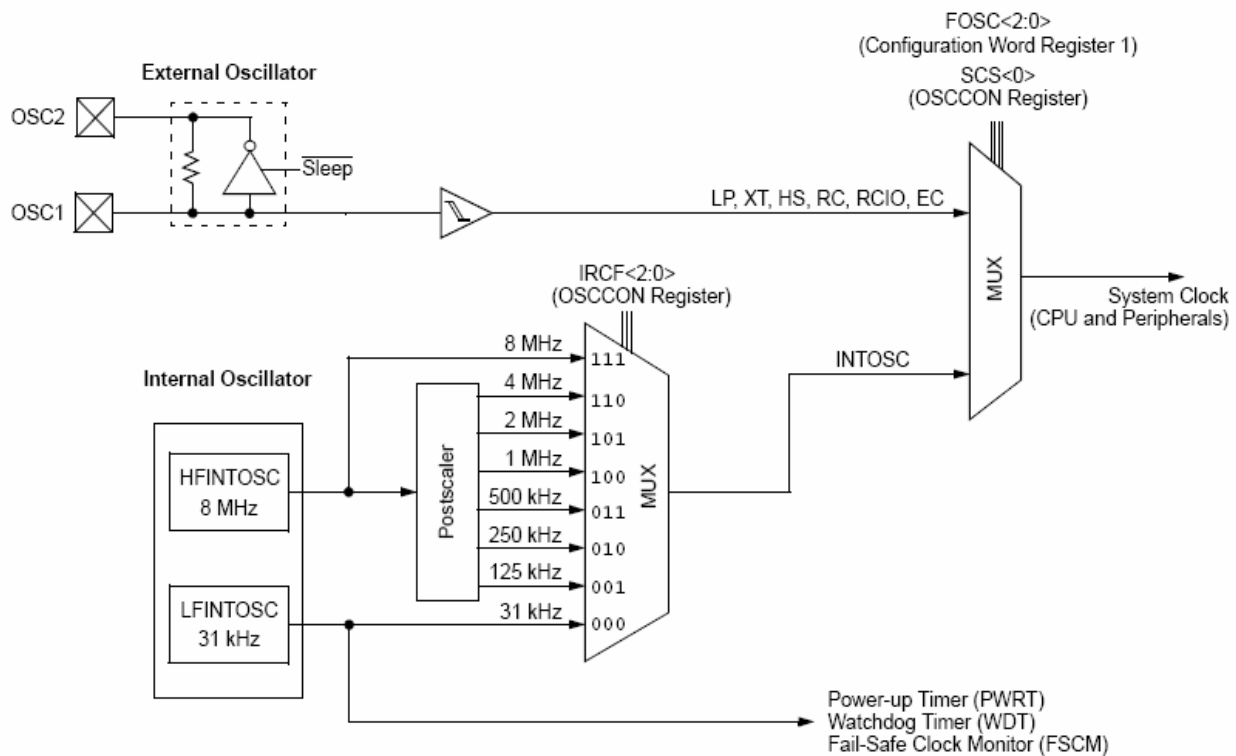
## 基本中斷練習

- 本練習的目標是：
  - ◆ 學習如何設定和致能中階PIC®系列微控器的內部中斷
  - ◆ 更加熟悉 MPLAB IDE、APP001實驗板 和 ICD2
    - 組譯程式項目
    - 使用 ICD 2 設定中斷點



## PIC16F887 震盪器方塊圖

本課程所使用得震盪方式採 **Internal RC @ 4MHz**



## 練習一的 Configuration 選項

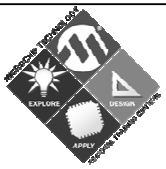
- **`_CONFIG _CONFIG1, _INTRC_OSC_NOCLKOUT & _WDT_OFF & _PWRTE_OFF & _MCLRE_ON & _CP_OFF & _CPD_OFF & _BOR_ON & _IESO_OFF & _FCMEN_OFF & _LVP_OFF & _DEBUG_ON;`**
- **`_CONFIG _CONFIG2, _WRT_OFF & _BOR40V;`**

**`_CONFIG` 設定 Configuration Words 的 Macro 指令**

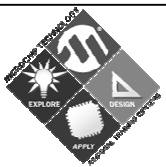
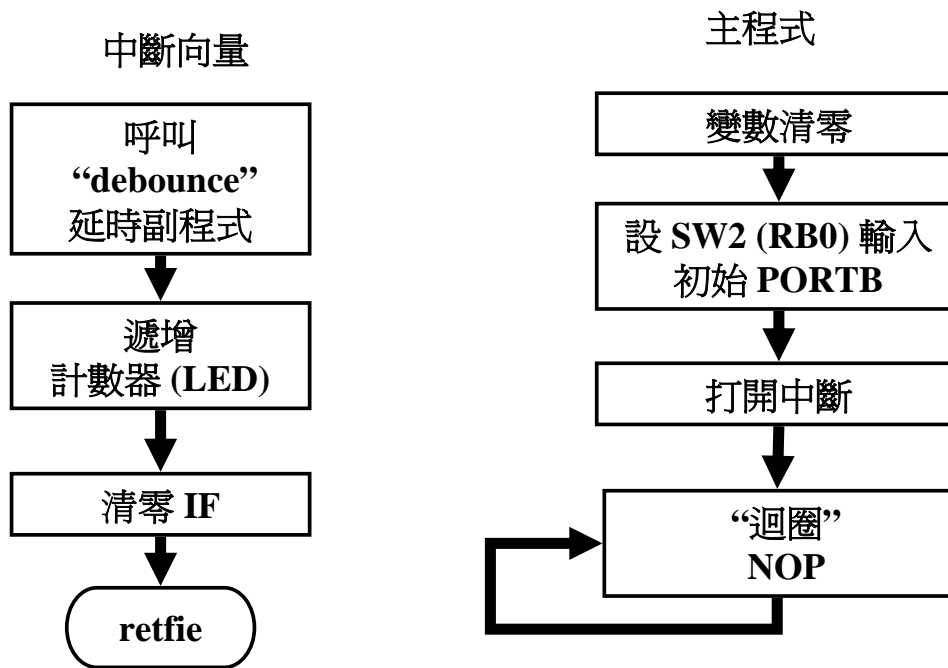
**`_CONFIG1` : 位址 0x2007 的 Configuration Word**

**`_CONFIG2` : 位址 0x2008 的 Configuration Word**

其餘的參數選項須參考 **p16f887.inc** 裡的定義與 **Data Sheet** 的說明。



## 練習 1 - 基本中斷概述



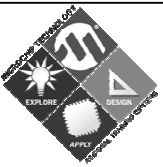
## 練習 1 細節（續）

- 程式位於 C:\RTC\201\_ASP\Lab1-INT
- SW2 連接在”RB0/INT” PORTB接腳
- 使用MPLAB IDE 修改程式並完成組譯動作，利用 ICD 2 將程式碼燒錄 PIC16F887 後，執行程式當按下SW2後觀察 LEDs 的輸出變化。



## 需要了解的內容

- INTCON暫存器各個位元的功能
- 一個名為“**debounce**”的副程式，該副程式延遲處理並防止SW2機械抖動，進而產生多個中斷（稍後的練習中將詳細介紹）
- 如何在MPLAB中設置斷點和使用“*Watch Window*”

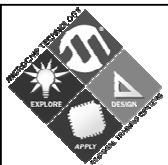


## 練習一解答

```
- bcf  INTCON,INTF
- bsf  INTCON,INTE ; ### Enable INT0 Interrupt
- bsf  INTCON,GIE  ; ### Enable Global Interrupt
- ;
- Loop
-   nop
-   goto Loop
- :
- :
- Int_Service_Routine
-   call Debounce ; Delay until switch stops bouncing
-   incf PORTD,F  ; increments number of time button
-                  ; has been pushed
-   bcf  INTCON,INTF ; ### clears the INT0 Interrupt Flag
-   retfie
```

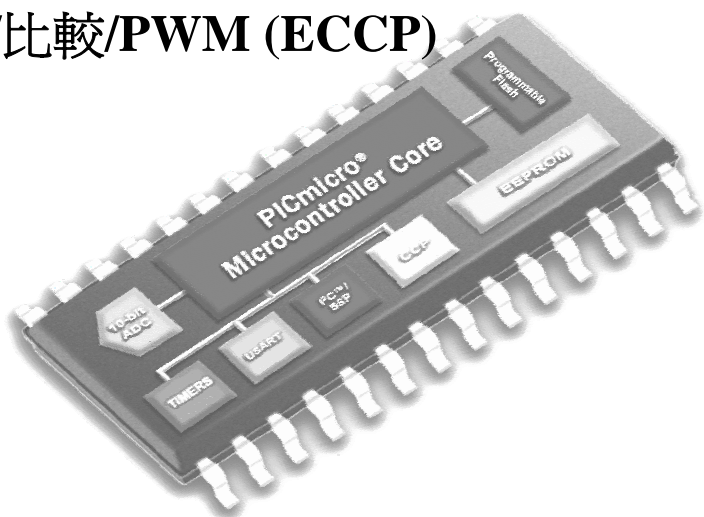


## 周邊



### 中階系列周邊

- 一般輸出、入 I/O 接腳
- 計時器 (0、1、2)
- 增強型脈波量測器/比較/PWM (ECCP)
- 類比電壓比較器
- 類比轉換器
- EUSART
- I<sup>2</sup>C和SPI 通訊面





## 輸出、入 I/O 概述

- 最多 35 個雙向 I/O
  - 其中一些接腳與周邊功能共用
- 高驅動能力
  - 25 mA
- 直接，單指令周期的位元操作
- 多數 I/O 具有 ESD 保護二極體
- 啓動時，如該 I/O 接腳與類比輸入共用，則內定爲類比輸入功能（高阻態）



## PORTx 和 TRISx 暫存器

- 每個 PORT（A、B、C、D 或 E）都具有相對應的資料方向暫存器 TRISx

### PORTB 暫存器

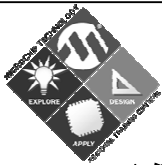
|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 |
|-----|-----|-----|-----|-----|-----|-----|-----|

### PORTB 三態控制暫存器 TRISB

|        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|
| TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 |
|--------|--------|--------|--------|--------|--------|--------|--------|

1 = 相應的 PORTB 接腳爲輸入

0 = 相應的 PORTB 接腳爲輸出



## ANSEL 和 ANSELH 暫存器

- 兩個用來將 I/O 設定為數位功能的暫存器
  - 啟動時與類比周邊共用的 I/O，內定為類比輸入

類比輸入選擇暫存器 (ANSEL)

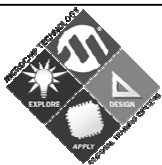
|      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
| ANS7 | ANS6 | ANS5 | ANS4 | ANS3 | ANS2 | ANS1 | ANS0 |
|------|------|------|------|------|------|------|------|

類比輸入選擇暫存器的高位元組 (ANSELH)

|  |  |       |       |       |       |      |      |
|--|--|-------|-------|-------|-------|------|------|
|  |  | ANS13 | ANS12 | ANS11 | ANS10 | ANS9 | ANS8 |
|--|--|-------|-------|-------|-------|------|------|

1 = 接腳被設置為類比輸入

0 = 數位 I/O



## 初始設定數位 I/O 腳

- 範例：
  - 初始設定 PORTB，設置 RB<7:4> 為輸入，RB<3:0> 為輸出

```
;-----configure PORTB for digital -----
banksel    PORTB                ;Go to bank containing PORTB
                                ;register
clrf       PORTB                ;Clear PORTB
banksel    ANSELH               ;Go to bank containing ANSELH
                                ;register
clrf       ANSELH               ;Set as all digital
clrf       ANSEL

;-----Set up direction of each PORTB pin-----
banksel    TRISB                ;Go to bank containing TRISB
                                ;register
movlw      b'11110000'          ;Value to set TRISB<7:4> high
                                ;and TRISB<3:0> low move into
                                ;W register
movwf     TRISB                ;Move value in W into TRISB
```





## PORTB 位準改變的中斷

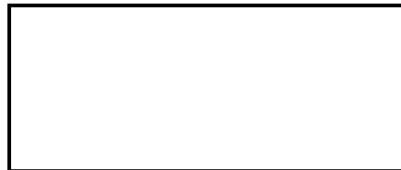
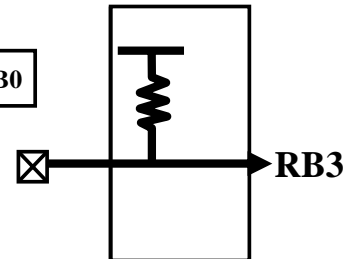
- 在PIC16F887中所有PORTB接腳都具有位準改變中斷和弱提升電阻的選項

弱提升電阻PORTB 暫存器 (WPUB)

|       |       |       |       |   |       |       |       |
|-------|-------|-------|-------|---|-------|-------|-------|
| WPUB7 | WPUB6 | WPUB5 | WPUB4 | 1 | WPUB2 | WPUB1 | WPUB0 |
|-------|-------|-------|-------|---|-------|-------|-------|

1 = 開啟提升電阻

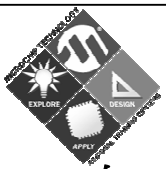
0 = 關閉提升電阻



中斷控制暫存器 (INTCON)

|     |      |      |      |      |      |      |      |
|-----|------|------|------|------|------|------|------|
| GIE | PEIE | TOIE | INTE | RBIE | TOIF | INTF | RBIF |
|-----|------|------|------|------|------|------|------|

\*必須先讀/寫 PORTB 一次，然後才能用軟體清除 RBIF



## PORTB 位準改變的中斷

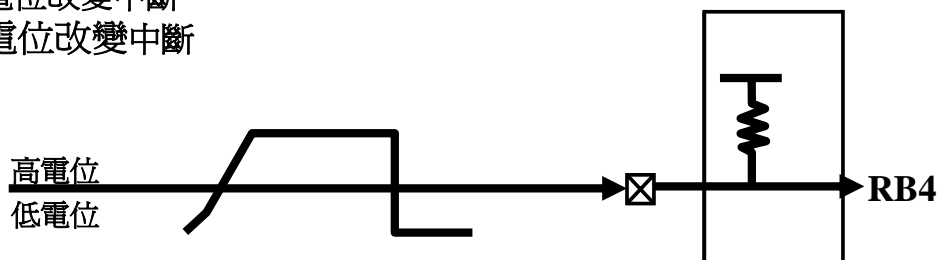
- 在PIC16F887中所有PORTB接腳都具有位準改變中斷和弱提升電阻的選項

電位改變中斷PORTB 暫存器 (IOCB)

|       |       |       |   |       |       |       |       |
|-------|-------|-------|---|-------|-------|-------|-------|
| IOCB7 | IOCB6 | IOCB5 | 1 | IOCB3 | IOCB2 | IOCB1 | IOCB0 |
|-------|-------|-------|---|-------|-------|-------|-------|

1 = 致能電位改變中斷

0 = 禁止電位改變中斷



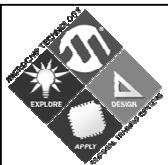
中斷控制暫存器 (INTCON)

|     |      |      |      |      |      |      |      |
|-----|------|------|------|------|------|------|------|
| GIE | PEIE | TOIE | INTE | RBIE | TOIF | INTF | RBIF |
|-----|------|------|------|------|------|------|------|

\*必須先讀/寫 PORTB 一次，然後才能用軟體清除 RBIF

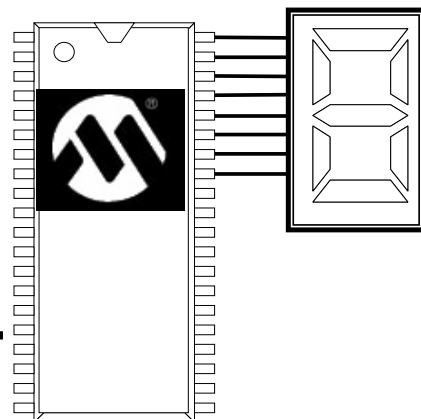


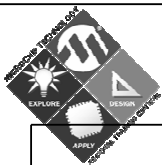
# 計時器



## 計時器

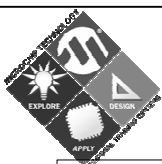
- 計時器可用於多種功能：
  - 時序參考以產生事件中斷
  - 計算事件數
  - 波形產生等
- PIC16F887有 3 個計時器
  - Timer0
  - Timer1
  - Timer2



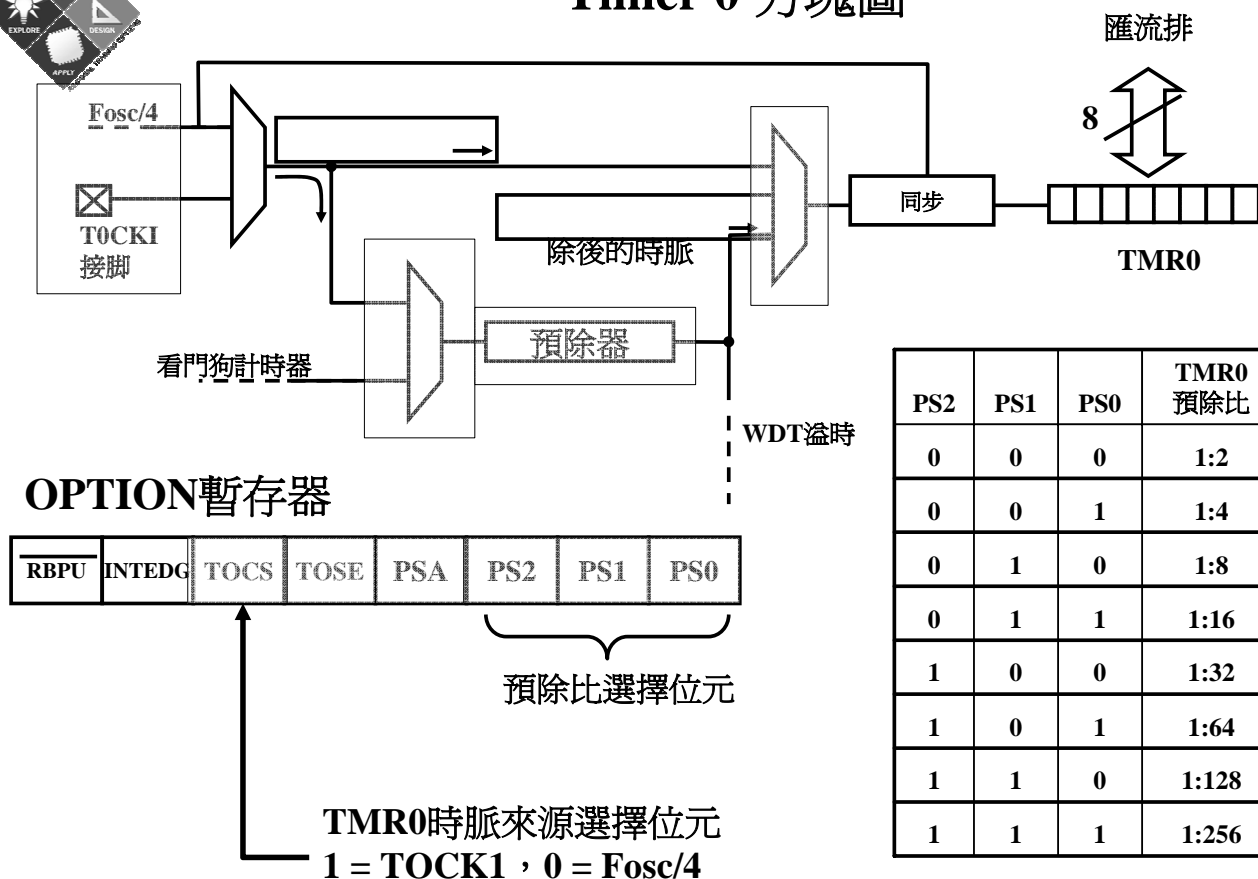


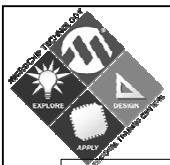
## PIC16F887 計時器比較

|                   | TIMER0                          | TIMER1                             | TIMER2                                    |
|-------------------|---------------------------------|------------------------------------|---|
| 暫存器大小             | 8位 (TMR0)                       | 16位<br>(TMR1H:TMR1L)               | 8位 (TMR2)                                 |
| 時脈源<br>(內部)       | Fosc/4                          | Fosc/4                             | Fosc/4                                    |
| 時脈源<br>(外部)       | T0CKI 接腳                        | T1CKI接腳或Timer 1<br>振盪器 (T1OSC)     | 無   |
| 可用的時脈除頻<br>(解析度)  | 8位預除器<br>(1:2→1:256)            | 3位預除器 (÷1、<br>÷2、÷4或÷8)            | 預除器<br>(1:1、1:4或1:8)<br>後除器<br>(1:1→1:16) |
| 發生中斷事件<br>中斷旗標位置  | 溢出時<br>FFh→00h<br>(T0IF在INTCON) | 溢出時<br>FFFFh→0000h<br>(TMR1F在PIR1) | TMR2與PR2匹配<br>(TMR2F在PIR2)                |
| 將PIC微控器從休<br>眠中喚醒 | 否                               | 是<br>(需使用外部震盪輸入)                   | 否   |

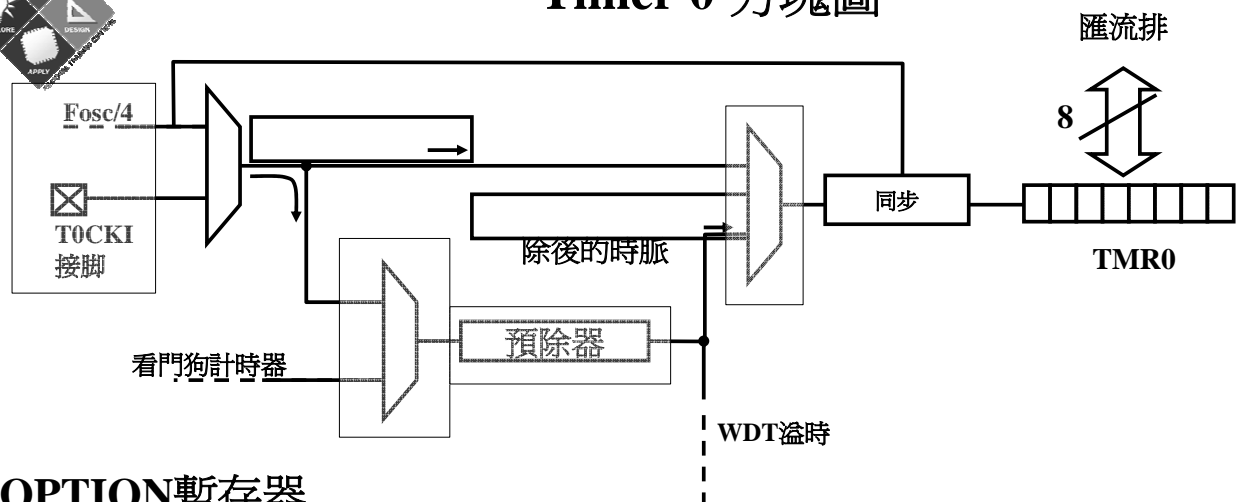


## Timer 0 方塊圖





## Timer 0 方塊圖

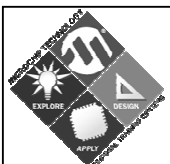


### OPTION暫存器

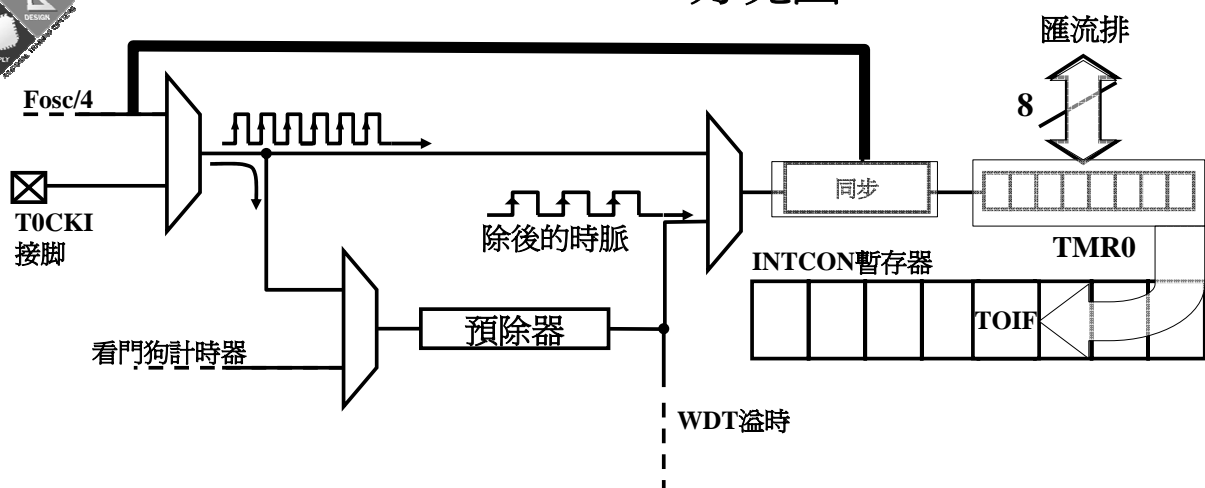
| RBPV | INTEDG | TOCS | TOSE | PSA | PS2 | PS1 | PS0 |
|------|--------|------|------|-----|-----|-----|-----|
|------|--------|------|------|-----|-----|-----|-----|

預除器分配位元  
 1= 預除器分配給WDT  
 0= 預除器分配給Timer 0

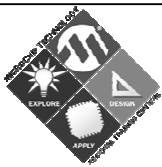
觸發緣選擇位元  
 1 =在上升緣遞增TMR0  
 0 =在下降緣遞增TMR0



## Timer 0 方塊圖



- 若  $T0CK1$  用作時脈源，可以先送給預除器除頻，然後再與內部時脈同步
- 可通過匯流排直接讀寫 Timer 0
  - 寫入操作將禁止計時器遞增（時間為2個  $T_{CY}$ ）
- $INTCON<T0IF>$  由Timer 0 歸零時會設為“1”
- $FFh \rightarrow 00h$



## Timer0 初始化（内部时脉来源）

;Make sure the TMR0 register is clear

```
banksel    TMR0
clrf       TMR0
```

; Clear T0IF

```
bcf        INTCON,T0IF
```

;Setup the following in the OPTION\_REG

;Timer0 increment from internal clock

;with a prescaler of 1:16

```
banksel    OPTION_REG
movlw      b'00000011'
movwf      OPTION_REG
```

;The TMR0 interrupt is disabled, do polling  
;on the T0IF overflow bit

```
btfss      INTCON, T0IF
goto       $-1
```

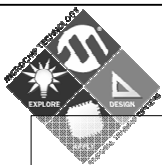
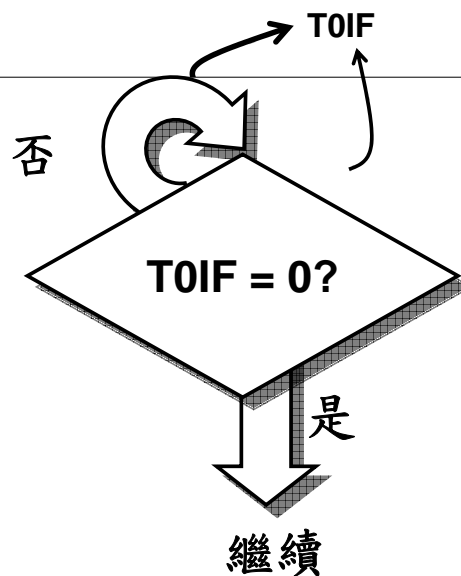
<continue>

TMR0

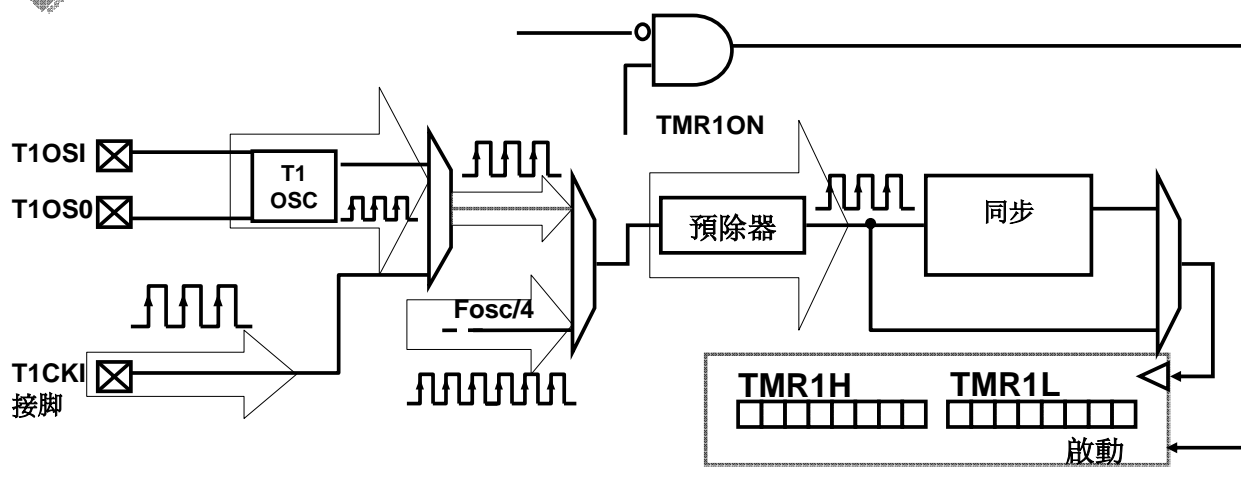
|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

INTCON

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|



## Timer1 方塊圖



Timer1控制暫存器 (T1CON)

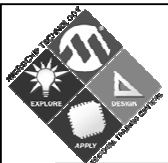
|        |        |         |         |         |        |        |        |
|--------|--------|---------|---------|---------|--------|--------|--------|
| T1GINV | TMR1GE | T1CKPS1 | T1CKPS0 | T1OSCEN | T1SYNC | TMR1CS | TMR1ON |
|--------|--------|---------|---------|---------|--------|--------|--------|

Timer1啟動位元  
1 = 啟動Timer1

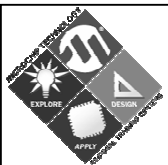
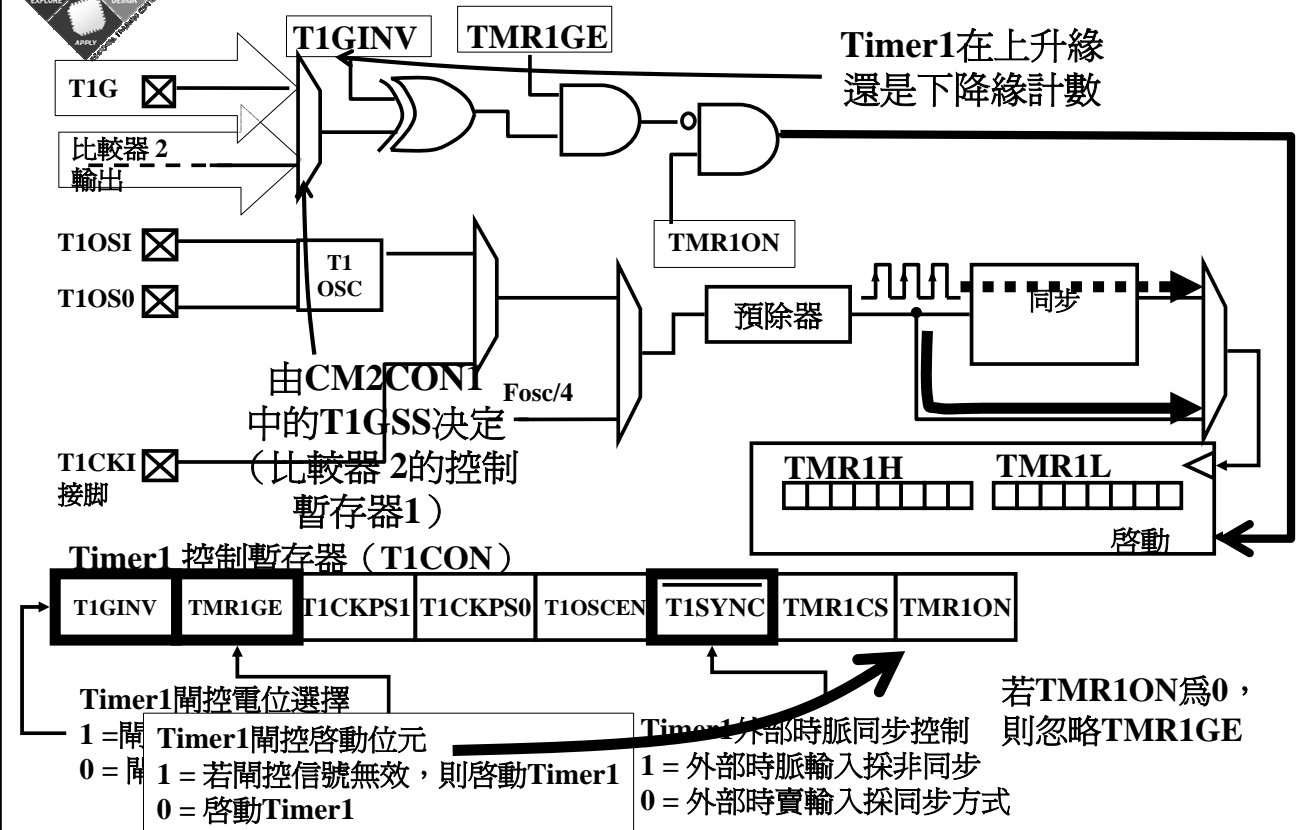
| T1CKPS1 | T1CKPS0 | 除頻比率 |
|---------|---------|------|
| 1       | 1       | 1:8  |
| 1       | 0       | 1:4  |
| 0       | 1       | 1:2  |
| 0       | 0       | 1:1  |

外部低頻震盪設定位元  
1 = 啟動外部震盪器  
0 = 關閉外部震盪器

時脈來源選擇位元  
1 = 外部 (T1CKI的上升沿緣)  
0 = 內部F<sub>OSC</sub>/4



## Timer1 方塊圖



## Timer1 中斷

### Main Code

#### Start

;Start by clearing the Timer1 interrupt flag

banksel PIR1

bcf PIR1, TMR1IF

### PIR1



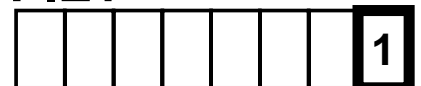
TMR1IF

;Enable Timer1 interrupt

banksel PIE1

bsf PIE1, TMR1IE

### PIE1



TMR1IE

;Enable Global and Peripheral Interrupts

bsf INTCON, GIE

bsf INTCON, PEIE

### INTCON



GIE PEIE

;



## Timer1 初始化（内部時脈源）

Make sure the TMR1 registers are clear

```
banksel    TMR1H
clrf       TMR1H
clrf       TMR1L
```

Make sure the TMR1IF flag in PIR1

```
; is cleared
banksel    PIR1
bcf        PIR1,TMRIF
```

Setup T1CON register for internal clock with  
1:8 prescaler, Timer1 is stopped and T1 osc  
is disabled

```
movlw      b'00110000'
movwf      T1CON
```

Start Timer1 incrementing

```
bsf        T1CON,TMR1ON
```

The TMR1 interrupt is disabled, do polling  
on the TMR1IF overflow bit

```
btfss      PIR1,TMR1IF
goto       $+1
```

TMR1H

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

TMR1L

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

INTCON

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

PIR1

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

PIE1

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

T1CON

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

TMR1IF

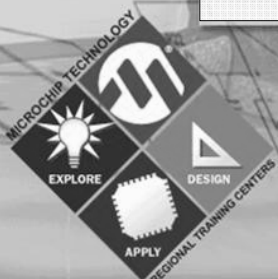
TMR1IE

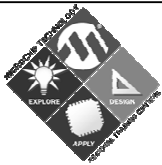


# HANDS-ON

# Training

## Timer 1 練習 (Lab 2)



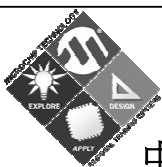


# Timer 1 練習

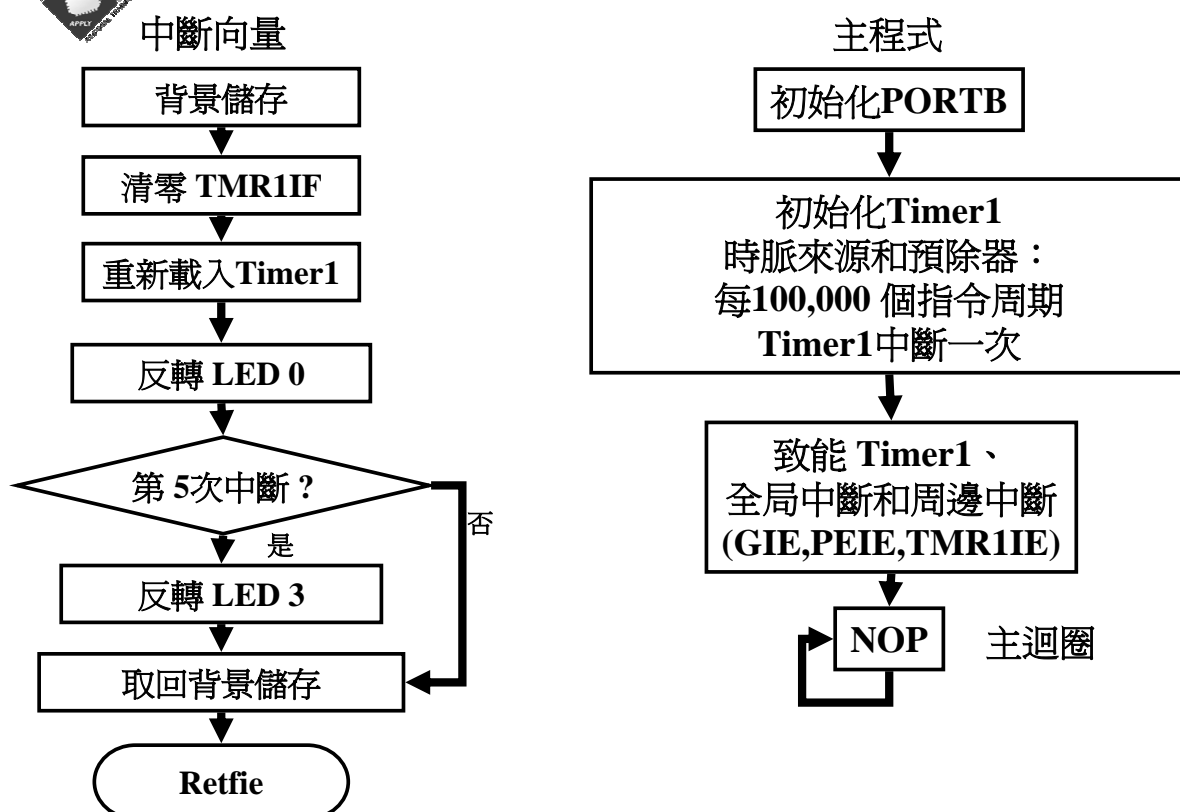
- 本練習目標是熟悉 Timer1 的工作原理

以及

- 獲得周邊中斷致能的經驗



## 練習二流程

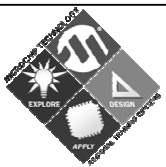






## 練習二細節

- 本練習的程式位於  
C:\RTC\201\_ASP\Lab2-TMR1
- 在 lab2.asm 中完成以下任務
  - 將 Timer 1 時脈來源設置為  $F_{osc}/4$  ( $F_{cy}$ )
  - 將 Timer 1 預除比率設為 2
  - Timer 1 載入 0x3CB0 (65,356 – 50,000)
  - 啟動 Timer 1
  - 致能 Timer 1 中斷 (GIE、PFIE & TMR1IE)



## 需要了解的內容

- INTCON、T1CON、TMR1H、TMR1L 和 PIE1 暫存器的操作
- 設定值 0x3CB0 和預除比為 2，Timer1 將每 100,000 個指令周期溢位一次
- 提供了反轉 LED 的中斷向量程式

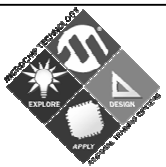


## 練習二 解答

```
*****
;
; Set code to Select clock source, Set pre-scaler to 2, load hex 3CB0 into Timer1
; and turn on Timer1
;
*****

    movlw      (.65536-.50000) / .256      ; initialize TMR1L and TMR1H
    movwf     TMR1H
    movlw     (.65536-.50000) % .256
    movwf     TMR1L
    bsf       T1CON,T1CKPS0                ; set pre-scaler to 2
    bcf       T1CON,TMR1CS                 ; set Clock source to Fosc/4
    bsf       T1CON,TMR1ON                 ; turn Timer1 on

;
; *****
; Enable Timer1 interrupts, Peripheral Interrupts and Global Interrupts
; *****
;
;
    bsf       STATUS,RP0                   ; go to bank1
    bsf       PIE1,TMR1IE
    bsf       INTCON,GIE
    bsf       INTCON,PEIE
    bcf       STATUS,RP0                   ; return to bank0
```



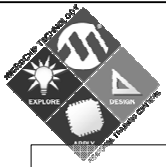
## 練習二問與答

問： Timer 1 在進入中斷這段時間內是否繼續工作？

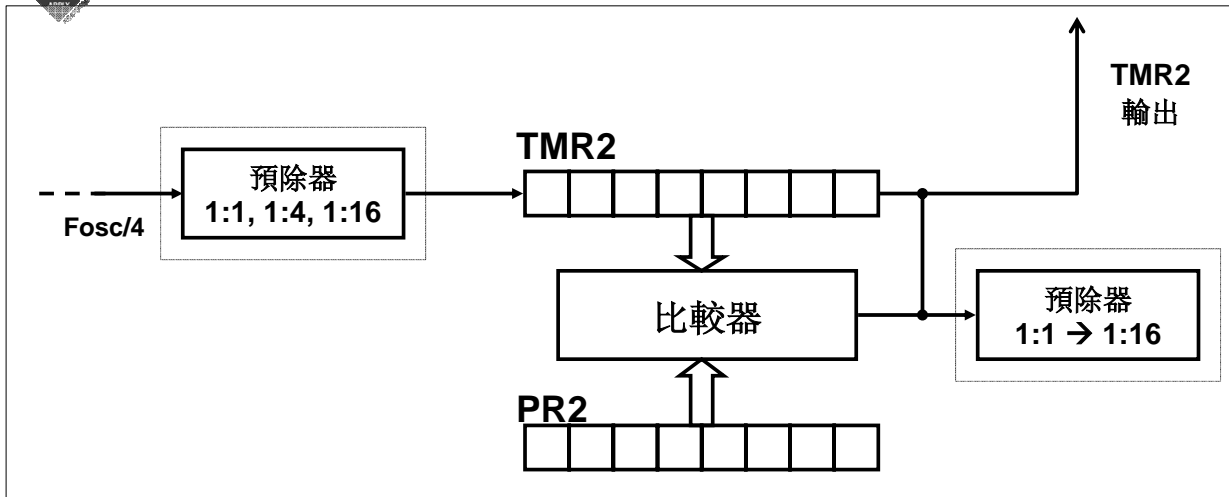
答： 是

問： 這將對要被重新裝載到TMR1L和TMR1H的值產生什麼影響？

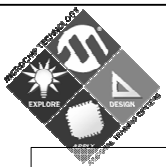
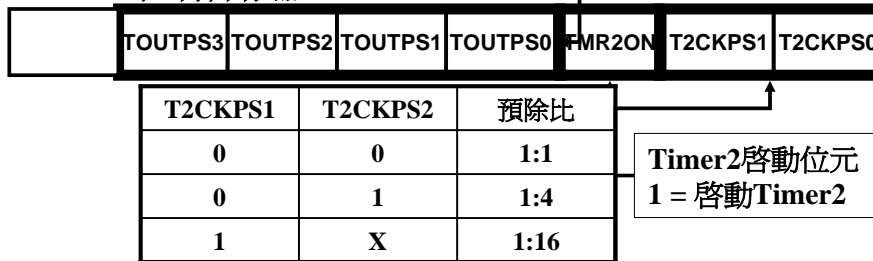
答： 影響很大 – 要實現高精度，應當考慮重新載入 Timer1 的中斷響應的延遲時間



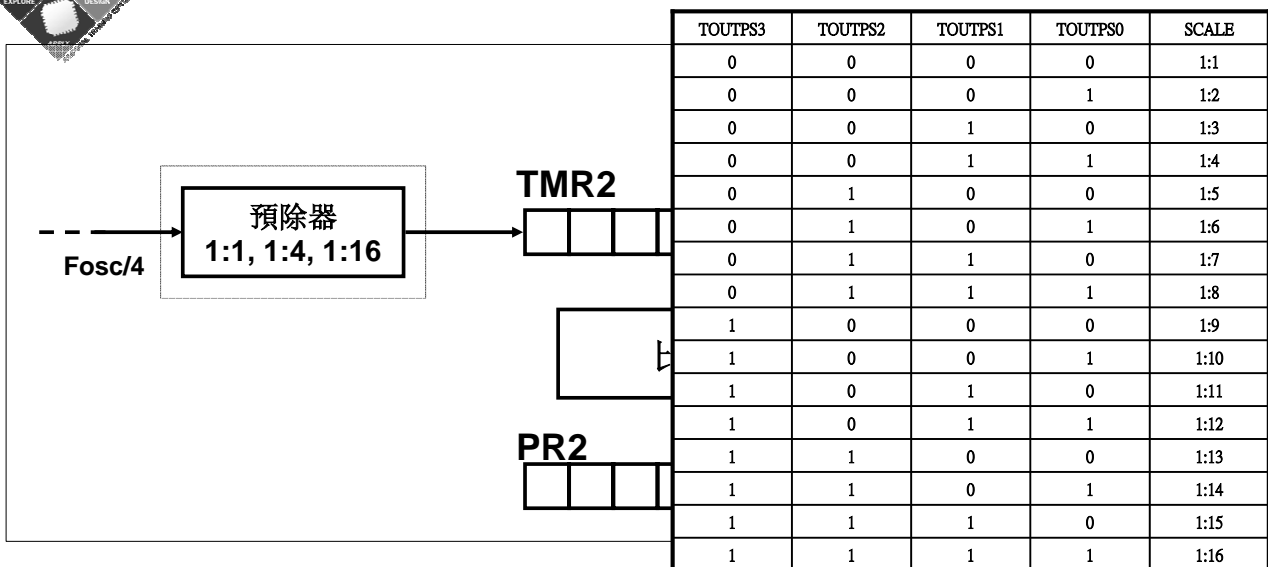
## Timer2 方塊圖



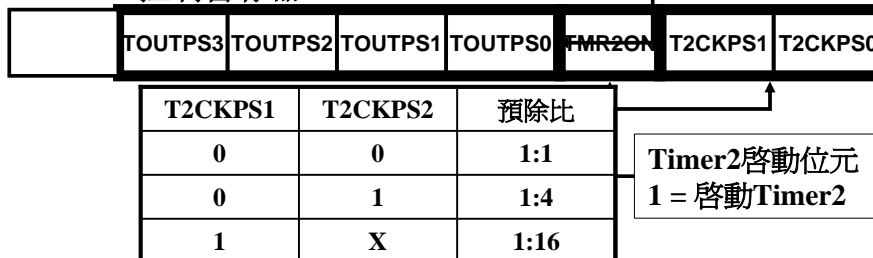
### Timer2 控制暫存器 (T2CON)

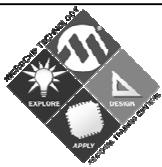


## Timer2 方塊圖

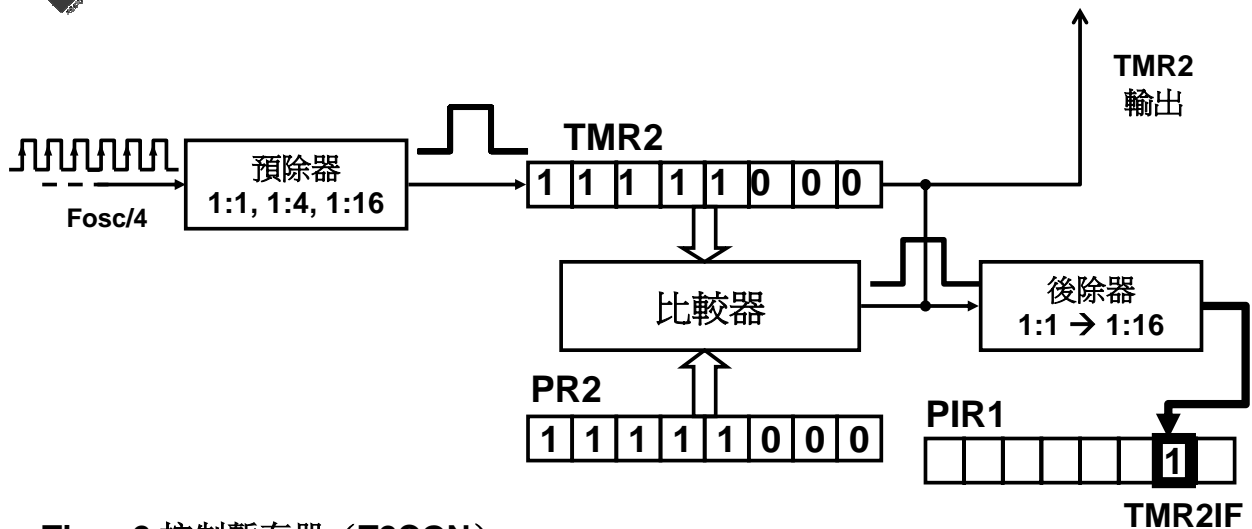


### Timer2 控制暫存器 (T2CON)





## Timer2 方塊圖



### Timer2 控制暫存器 (T2CON)

|  |         |         |         |         |        |         |         |
|--|---------|---------|---------|---------|--------|---------|---------|
|  | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
|--|---------|---------|---------|---------|--------|---------|---------|



## Timer2 初始化

```

;Disable TMR1 interrupts in the PIE1
;register that holds the TMR2IE bit.
;Make sure the TMR2IF flag in PIR1 is cleared

```

```

banksel      PIE1
bcf          PIE1, TMR2IE
banksel      PIR1
bcf          PIR1, TMR2IF

```

```

;Setup T2CON register for:

```

```

; Postscaler = 1:15
; Prescaler = 1:16

```

```

;Timer2 is off

```

```

movlw       b'01110010'
movwf       T2CON

```

```

;Make sure the TMR2 register is clear

```

```

banksel      TMR2
clrf         TMR2

```

```

;Load the PR2 register with a predetermined

```

```

;value

```

```

banksel      PR2
movlw       b'10000000'
movwf       PR2

```

```

;Start Timer2 incrementing

```

```

banksel      T2CON
bsf          T2CON, TMR2ON

```

```

;The TMR2 interrupt is disabled, do polling

```

```

;on the TMR2IF flag

```

```

btfss       PIR1, TMR2IF
goto        $-1

```

### TMR2

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

### PIE1

|  |  |  |  |  |  |   |  |
|--|--|--|--|--|--|---|--|
|  |  |  |  |  |  | 0 |  |
|--|--|--|--|--|--|---|--|

### TMR2IE

### PIR1

|  |  |  |  |  |  |   |  |
|--|--|--|--|--|--|---|--|
|  |  |  |  |  |  | 0 |  |
|--|--|--|--|--|--|---|--|

### TMR2IF

### T2CON

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

TOUTPS<3:0>

T2CKPS<1:0>

後除比  
設置為1:15

Timer2  
關閉

預除比  
設置為1:16

### PR2

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|



## Timer2 初始化

| TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | 除比   |
|---------|---------|---------|---------|------|
| 0       | 0       | 0       | 0       | 1:1  |
| 0       | 0       | 0       | 1       | 1:2  |
| 0       | 0       | 1       | 0       | 1:3  |
| 0       | 0       | 1       | 1       | 1:4  |
| 0       | 1       | 0       | 0       | 1:5  |
| 0       | 1       | 0       | 1       | 1:6  |
| 0       | 1       | 1       | 0       | 1:7  |
| 0       | 1       | 1       | 1       | 1:8  |
| 1       | 0       | 0       | 0       | 1:9  |
| 1       | 0       | 0       | 1       | 1:10 |
| 1       | 0       | 1       | 0       | 1:11 |
| 1       | 0       | 1       | 1       | 1:12 |
| 1       | 1       | 0       | 0       | 1:13 |
| 1       | 1       | 0       | 1       | 1:14 |
| 1       | 1       | 1       | 0       | 1:15 |
| 1       | 1       | 1       | 1       | 1:16 |

| T2CKPS1 | T2CKPS2 | 後除比  |
|---------|---------|------|
| 0       | 0       | 1:1  |
| 0       | 1       | 1:4  |
| 1       | X       | 1:16 |

TMR2

**TMR2 = PR2**

PIE1

0

TMR2IE

PIR1

1

TMR2IF

T2CON

0 1 1 1 0 1 1 0

TOUTPS<3:0>

T2CKPS<1:0>

後除比  
設置為1:15

Timer2  
關閉

預除比  
設置為1:16

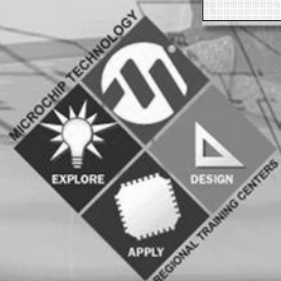
PR2

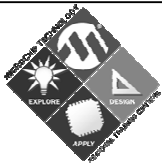
1 0 0 0 0 0 0 0

HANDS-ON

Training

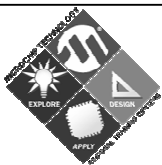
## Timer 2 練習 (Lab 3)



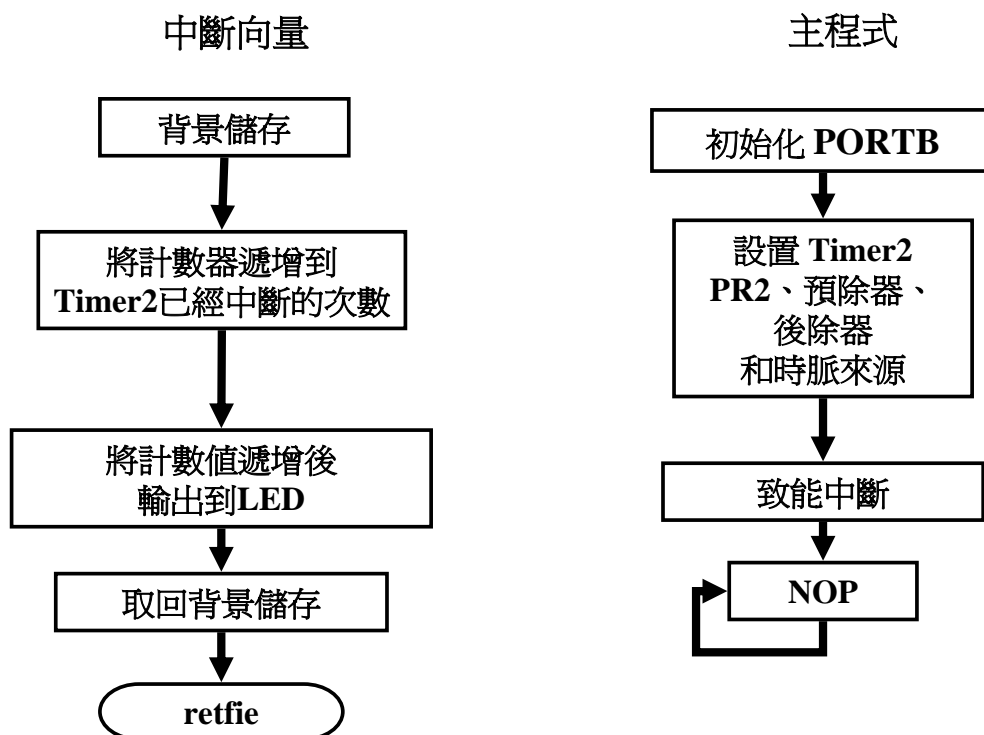


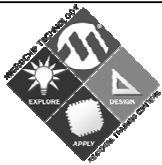
## Timer 2 練習

- 練習 3 的目標是熟悉以下設定：
  - Timer2 時脈來源
  - 預除值
  - 後除值
  - 啓動 Timer2
  - 讓 Timer2 產生中斷所需要設定的中斷控制位元



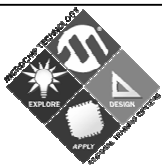
## 練習三流程





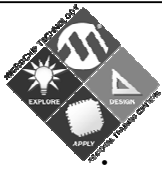
## 練習三細節

- 本練習程式位於  
C:\RTC\201\_ASP\Lab3-TMR2
- 完成程式的以下部分
  - 將 Timer 2 預除比設置為 4
  - 將 Timer 2 後除比設置為 10
  - 啓動 Timer 2
  - 設定 GIE、PEIE 和 Timer 2 中斷致能位元



## 需要了解的內容

- 本練習需要的特殊功能暫存器（SFR）為：  
INTCON、PIE1、PR2 和 T2CON
- 將 PR2 設置為 250、預除比設置為 4，並將後除比設置為 10，使用 4Mhz 振盪器時（ $F_{osc}/4 = 1 \text{ Mhz}$ ），Timer 2 每 10 ms 中斷一次

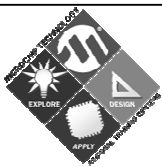


## Timer 2 練習的解答

```
*****
;
; configure Timer2 Prescaler of 4, PR2 of 250 and a postscaler of 13
; and turn timer2 on.
;
*****

    banksel        0                ; set bank to 0
    movlw          0x68              ; set 13 as postscaler
    movwf          T2CON
    bsf            T2CON,T2CKPS0     ; set prescaler to 4
    bsf            T2CON,TMR2ON      ; turn on Timer2

;
; *****
; Enable Timer2 interrupts, Peripheral Interrupts and Global Interrupts
; *****
;
    bsf            STATUS,RP0        ; go to bank1
    bsf            PIE1,TMR2IE
    bsf            INTCON,GIE
    bsf            INTCON,PEIE
    bcf            STATUS,RP0        ; return to bank0
```



## 練習三問與答

問： Timer2 是否與 Timer1 一樣在中斷響應延時期間繼續運行？

答： 是的！

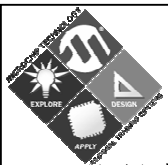
問：要確保精確的中斷周期，是否需必須考慮自由運行的 Timer2 ？

答： 不用，因為中斷是在匹配而非溢位時產生的





## 增強型 脈波量測器/比較/PWM模組 (ECCP)



### ECCP 概述

- 脈波量測器功能
  - 計算事件持續時間
- 比較
  - 經過了確定的時間後，觸發特殊事件
- 脈衝寬調變器（PWM）
  - 以定義的頻率建立可設定脈衝寬度穩定的方波輸出
  - 為各種橋式連接提供增強型功能

\* 模組會使用 Timer 1 或 Timer 2



## ECCP 概述

- 脈波量測器功能
  - 計算事件持續時間
- 比較

| ECCP模式 | 計時器資源   |
|--------|---------|
| 脈波量測器  | Timer 1 |
| 比較     | Timer 1 |
| PWM    | Timer 2 |

\* 模組會使用 Timer 1 或 Timer 2



## ECCP 控制暫存器

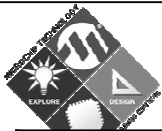
增強型CCP1控制暫存器 (CCP1CON)

|      |      |       |       |        |        |        |        |
|------|------|-------|-------|--------|--------|--------|--------|
| P1M1 | P1M0 | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 |
|------|------|-------|-------|--------|--------|--------|--------|

| 位元         | 功能  |
|------------|---|
| CCP1M<3:0> | ECCP 模式選擇位元 <sup>(i)</sup><br>將模組配置為各種模式，包括觸發特殊事件和邊緣檢測                      |
| P1M<1:0>   | 輸出設定位元 <sup>(ii)</sup><br>脈波量測器/比較模式 = 未使用“00”<br>增強型PWM = 提供半橋式或全橋式輸出的極性控制 |
| DC1B<1:0>  | PWM 脈衝寬度的 2 個LSB ( 8 個MSB 位於CCPR1L中)<br>*不使用者在脈波量測器或比較模式中                   |

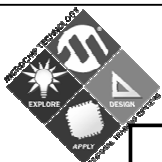
除了以下兩點不同外，CCP2CON與CCP1CON類似：

- 在模式選擇位中，比較模式將“反轉”輸出接腳準位
- 增強型PWM 設定功能（輸出設定位元）

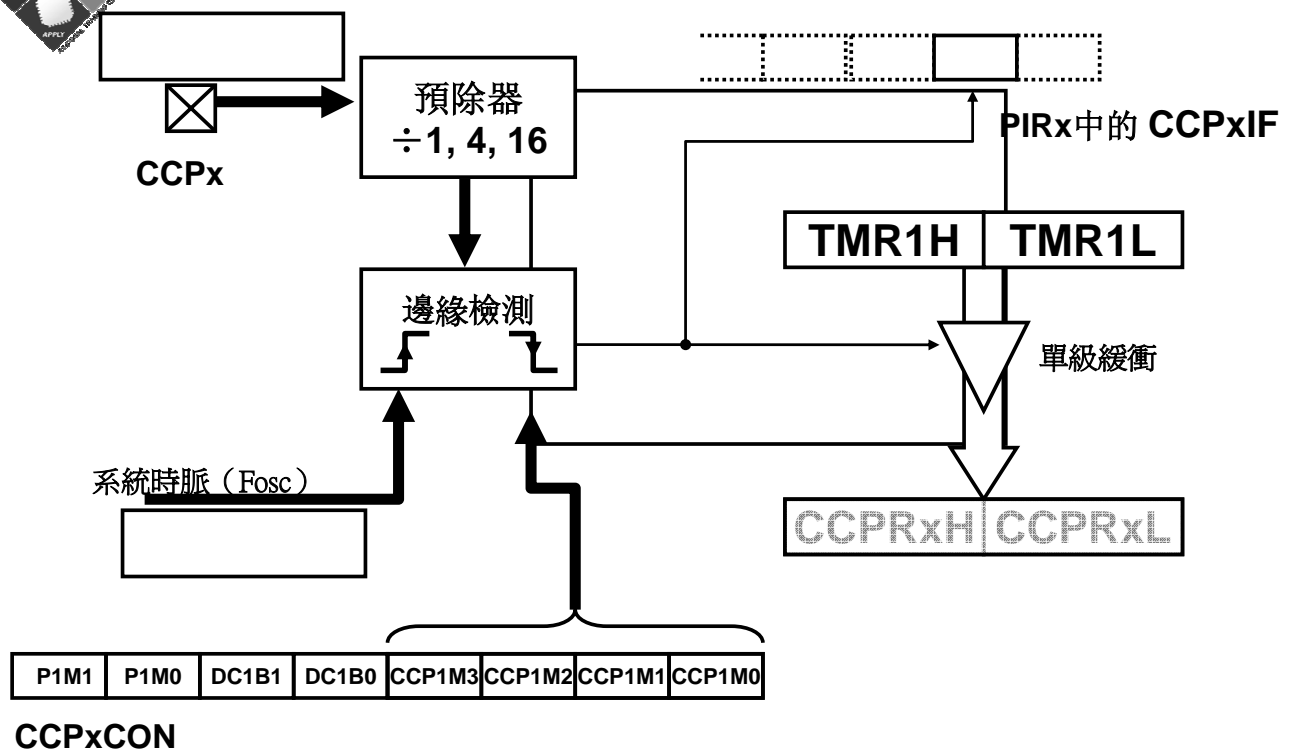


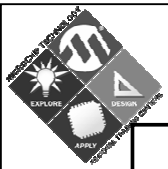
# ECCP 控制暫存器

| CCPxM3 | CCPxM2 | CCPxM1 | CCPxM0 | 選擇ECCP模式                         |
|--------|--------|--------|--------|----------------------------------|
| 0      | 0      | 0      | 0      | 脈波量測器/比較/PWM 關閉 (重置ECCP模組)       |
| 0      | 0      | 0      | 1      | 未用 (保留)                          |
| 0      | 0      | 1      | 0      | 比較模式，匹配時反轉輸出                     |
| 0      | 0      | 1      | 1      | 未用 (保留)                          |
| 0      | 1      | 0      | 0      | 脈波量測器模式，每個下降緣脈波量測器一次             |
| 0      | 1      | 0      | 1      | 脈波量測器模式，每個上升緣脈波量測器一次             |
| 0      | 1      | 1      | 0      | 脈波量測器模式，每4個上升緣脈波量測器一次            |
| 0      | 1      | 1      | 1      | 脈波量測器模式，每16個上升緣脈波量測器一次           |
| 1      | 0      | 0      | 0      | 比較模式，匹配時置 1 輸出                   |
| 1      | 0      | 0      | 1      | 比較模式，匹配時清零輸出                     |
| 1      | 0      | 1      | 0      | 比較模式，匹配時產生中斷                     |
| 1      | 0      | 1      | 1      | 比較模式，觸發特殊事件                      |
| 1      | 1      | 0      | 0      | PWM模式；P1A和P1C高電位有效；P1B和P1D 高電位有效 |
| 1      | 1      | 0      | 1      | PWM模式；P1A和P1C高電位有效；P1B和P1D低電位有效  |
| 1      | 1      | 1      | 0      | PWM模式；P1A和P1C低電位有效；P1B和P1D高電位有效  |
| 1      | 1      | 1      | 1      | PWM模式；P1A和P1C低電位有效；P1B和P1D低電位有效  |

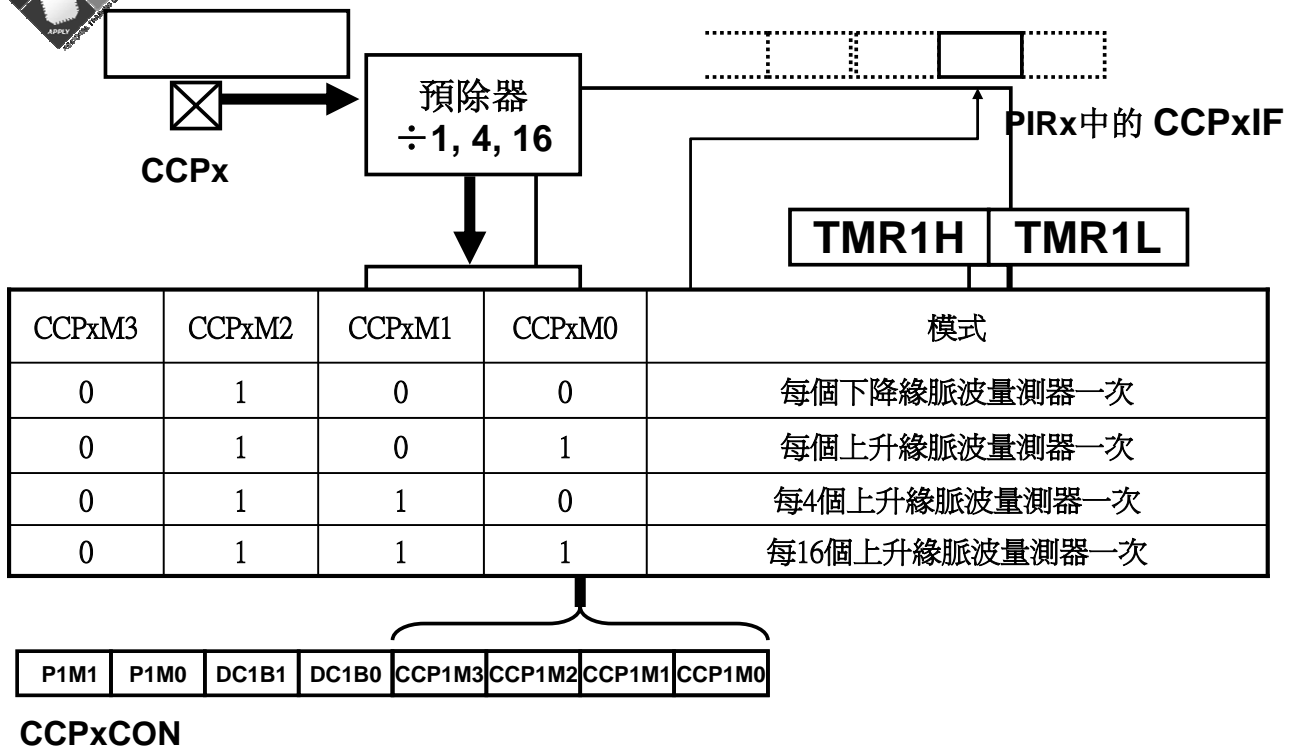


## 脈波量測器模式





## 脈波量測器模式



## 脈波量測器模式的初始化

```

;Turn off CCP module
banksel    CCP1CON
clrf       CCP1CON
;Make sure Timer1 is off
bcf        T1CON, TMR1ON
;Clear Timer1 result registers
clrf       TMR1H
clrf       TMR1L
;Disable all interrupts for CCP
bcf        PIR1, CCP1IF
banksel    PIE1
bcf        PIE1, CCP1IE
;Set CCP1 pin for input
bsf        TRISC, 2
;Initialize Capture for every 4th rising edge
banksel    CCP1CON
movlw      b'00000110'
movwf      CCP1CON
;Start Timer1 incrementing
bsf        T1CON, TMR1ON
;Test the CCP1IF flag for capture
btfss      PIR1, CCP1IF
goto       $-1
    
```

**TMR1H**

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

**TMR1L**

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

**CCPR1H**

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

**CCPR1L**

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

**PIR1**

|  |  |  |  |  |   |  |  |
|--|--|--|--|--|---|--|--|
|  |  |  |  |  | 0 |  |  |
|--|--|--|--|--|---|--|--|

**CCP1IF**

**CCP1CON**

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

**T1CON**

|  |  |  |  |  |  |  |   |
|--|--|--|--|--|--|--|---|
|  |  |  |  |  |  |  | 1 |
|--|--|--|--|--|--|--|---|

**TMR1ON**



## 脈波量測器模式的初始化

```
;Turn off CCP module
banksel    CCP1CON
clrf       CCP1CON
;Make sure Timer1 is off
bcf        T1CON, TMR1ON
;Clear Timer1 result registers
clrf       TMR1H
clrf       TMR1L
;Disable all interrupts for CCP
bcf        PIR1, CCP1IF
banksel    PIE1
bcf        PIE1, CCP1IE
;Set CCP1 pin for input
bsf        TRISC, 2
;Initialize Capture for every 4th rising edge
banksel    CCP1CON
movlw      b'00000110'
movwf      CCP1CON
;Start Timer1 incrementing
bsf        T1CON, TMR1ON
;Test the CCP1IF flag for capture
btfss      PIR1, CCP1IF
goto       $-1
```

當前Timer1值  
檢測到第4個上升沿!!

當前Timer1值

當前Timer1值

TMR1H  
TMR1L

CCP1IF

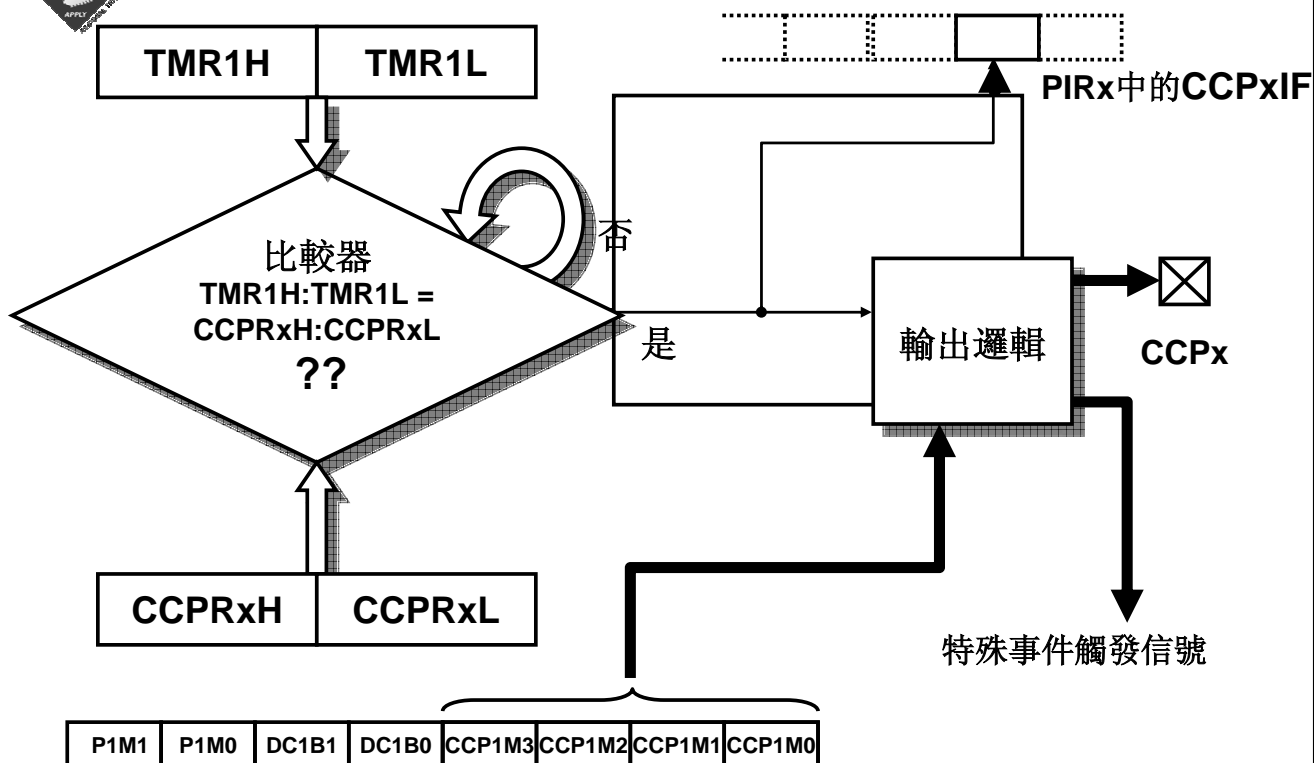
CCP1CON  
0 0 0 0 0 1 1 0

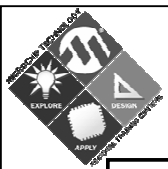
T1CON

TMR1ON

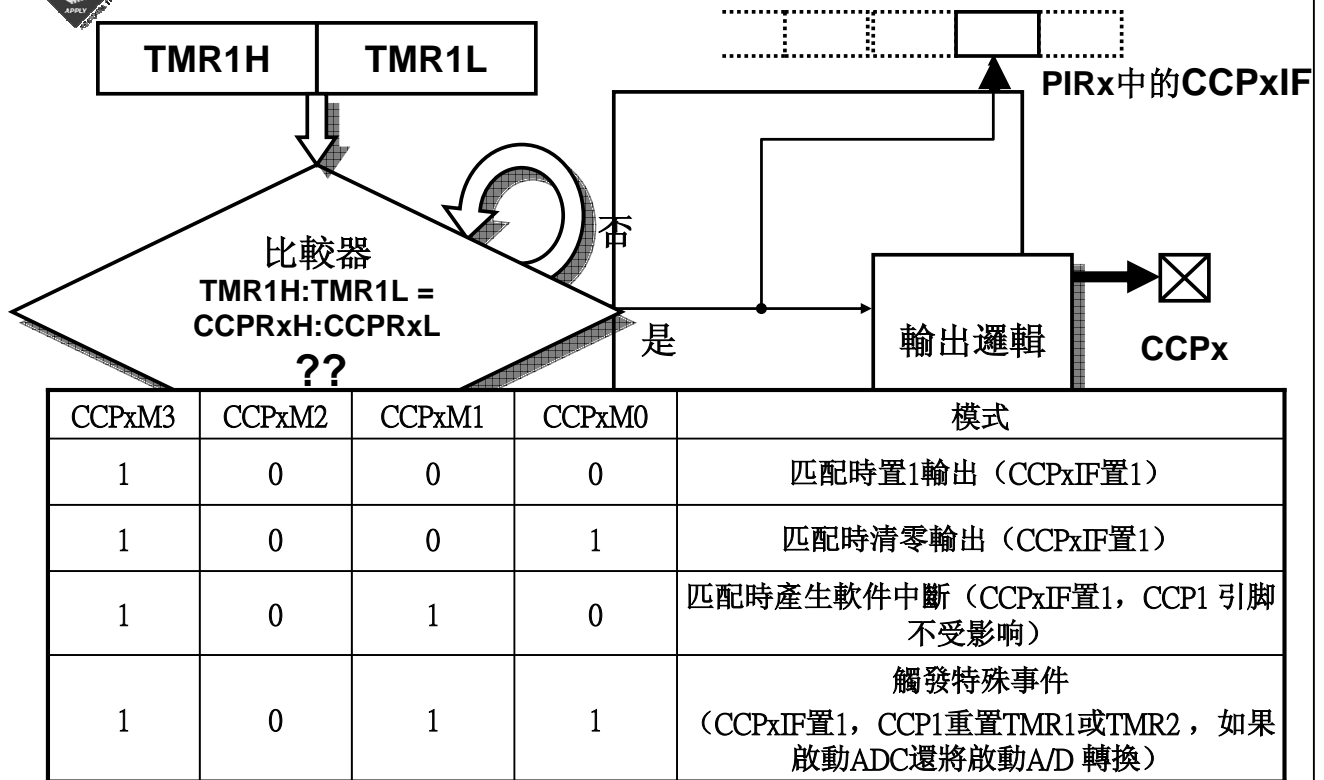


## 比較模式





## 比較模式



## 比較模式初始化

```

; Turn off the CCP module
banksel CCP1CON
clrf CCP1CON
; Turn off Timer1
bcf T1CON, TMR1ON
; Clear Timer1 result registers
clrf TMR1H
clrf TMR1L
; Disable CCP1 interrupt and make sure its flag is clear
banksel PIE1
bcf PIE1, CCP1IE
banksel PIR1
bcf PIR1, CCP1IF
; Make CCP1 pin output
banksel TRISC
bcf TRISC, 2
; Initialize Compare to set output on match
banksel CCP1CON
movlw b'00001000'
movwf CCP1CON
; Load half of Timer1 full scale value into CCP1H:CCP1L
banksel CCP1H
movlw b'10000000'
movwf CCP1H
clrf CCP1L
; Start Timer1 incrementing
bsf T1CON, TMR1ON
; Test CCP1IF for Timer1 match with CCPx
btfss PIR1, CCP1IF
goto $-1
    
```

TMR1H

0 0 0 0 0 0 0 0

TMR1L

0 0 0 0 0 0 0 0

CCPR1H

1 0 0 0 0 0 0 0

CCPR1L

0 0 0 0 0 0 0 0

T1CON

1

TMR1開啓

CCP1CON

0 0 0 0 1 0 0 0

PIR1

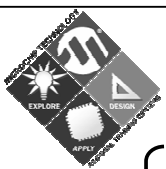
1

CCP1IF

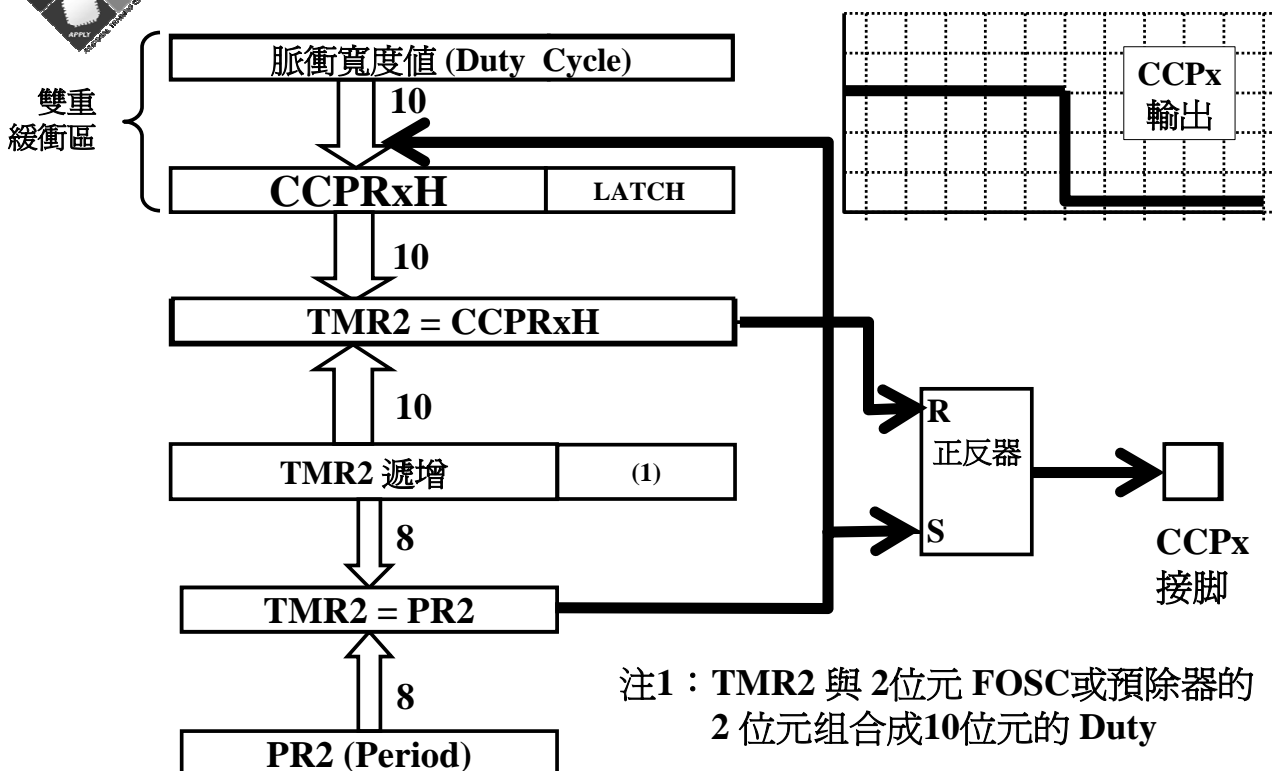


## PWM模式

- 在CCPx 接脚上產生脈衝寬調變器（ PWM ）輸出
- 由以下暫存器設定脈衝寬度、周期和分辨率
  - PR2
  - T2CON
  - CCPRxL
  - CCPxCON



## PWM 方塊圖





# PWM 操作的設定

```
;Turn off CCPx pin by setting TRISC bit HIGH
bankset    TRISC
bsf         TRISC, 2
```

```
;Clear Timer2
bankset     TMR2
clrf        TMR2
```

```
;Set up Period and Duty Cycle using an 8MHz oscillator
movlw       b'01111111' ;
movwf       PR2          ;Load a 64uS Period Value
movlw       b'00011111' ;
movwf       CCP1L         ;Load Duty Cycle Value
                        ; (25%) of PWM period
```

```
;Configure ECCP module for single PWM
; with P1A active HIGH and
;LSB's of Duty Cycle are '10'
movlw       b'00101100' ;
movwf       CCP1CON      ;ECCP module is configured
                        ;for PWM and Duty Cycle
                        ;LSB's loaded
```

```
;Turn CCPx pin back on
bankset     TRISC
bcf         TRISC, 2      ;Make CCP1 output
```

```
;Timer2 starts when TMR2ON is set beginning PWM
movlw       b'00000100' ;incrementing
movwf       T2CON         ;Prescaler and Postscaler
                        ;are both 1:1
```

TMR2

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

PR2

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

CCPR1L

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

CCP1CON

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

脈衝寬度  
LSB  
DC1B<1:0>

PWM模式  
CCP1M<3:0>

T2CON

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

後預除比位元  
TOUTPS<3:0>

前除比位元  
T2CKPS<1:0>

TMR2ON

## HANDS-ON

## Training

### 脈衝寬調變器練習 (PWM) Lab4





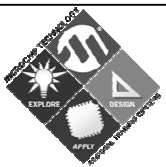


## PWM 練習四

- 本練習的目標是熟悉 ECCP 模組的設定和 PWM 模式下的操作

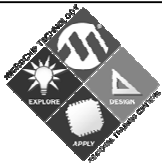
以及

- 了解有關 Timer2 設定的更多信息

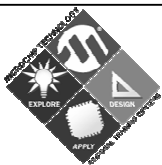
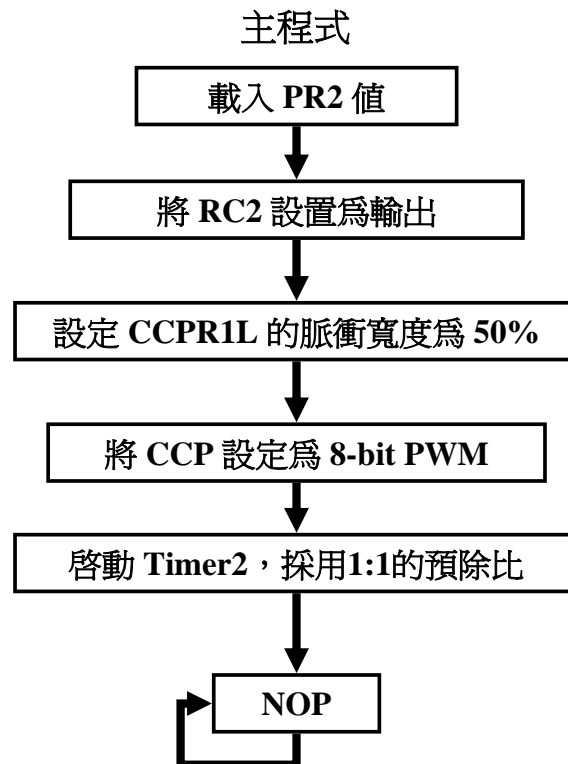


## PWM 練習四概述

- PWM 波形從 CCP1 接腳（RC2）輸出，使得 APP001 板上的蜂鳴器發出一個聲音。
- 完成練習後，周期為  $256/(F_{osc}/4)$ 、脈衝寬度為 50% 的信號將驅動此蜂鳴器(3.8KHz)



## PWM 練習四流程



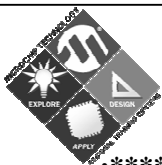
## 練習四細節

- 本練習程式位於  
C:\RTC\201\_ASP\Lab4-PWM
- 完成以下部分
  - 將 PORTC 的RC2 (CCP1) 設為輸出
  - 設置 CCP工作在 PWM 模式
  - 清零 DCB1和 DCB0 (8-bit PWM)
  - 將 Timer2 的前預除比值設置為 1:1



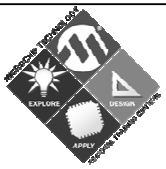
## 需要了解的內容

- 提供週期 PR2 (Timer2) 和設置50%脈衝寬度的程式。可在程式中找到這些值
- 在PIC16F887中，CCP1接腳為RC2
- 完成本練習所需的暫存器為：TRISC、T2CON 和 CCP1CON



## 練習四解答

```
*****  
; Set CCPx as an output  
*****  
; bcf TRISC,2 ; set CCP1 pin as output pin  
;  
; set duty cycle for 50%  
;  
; bcf STATUS,RP0 ; go to bank 0  
; movlw 0x80  
; movwf CCP1L ; set duty cycle  
*****  
; Put CCP1 module in PWM mode.  
; Configure CCP to clear DCB1 and DCB0 ( 8-bit PWM)  
*****  
; movlw 0x0C  
; movwf CCP1CON  
*****  
; Configure Timer2 Pre and post scale of 1:1  
; and turn Timer2 on  
*****  
; bsf T2CON,TMR2ON ; turn on Timer2
```



## 練習四問與答

問：爲什麼我們不使用PWM的中斷呢？

答：PWM可與PIC MCU同時運行，而不會降低處理器的速度

# HANDS-ON Training

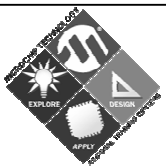
## 計時比較器練習 (Lab5)

MICROCHIP TECHNOLOGY  
EXPLORE DESIGN APPLY  
REGIONAL TRAINING CENTERS

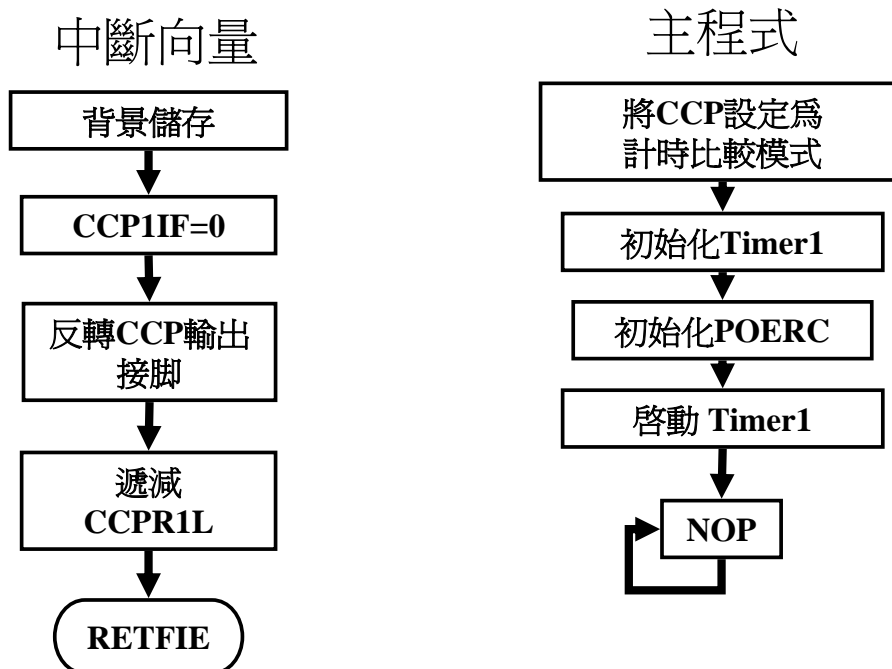


## 計時比較器練習

- 本練習的目標是獲取以下經驗：
  - 設定 ECCP 工作於計時比較模式
  - 設定特殊事件旗標以重置 Timer1
  - 設定 ECCP 以在 Timer1 溢出時產生中斷
  - 使用中斷向量來修改中斷間的時間間隔



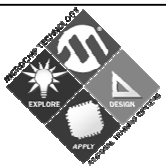
## 練習五流程





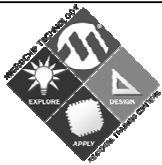
## CCP 練習五細節

- 本練習程式位於  
C:\RTC\201\_ASP\Lab5-CCP
- 完成以下部分：
  - 將CCP設定為計時比較模式，比較匹配時將設定特殊事件旗標和 CCP1IF
  - 設定 Timer1，將其時脈來源設置為  $F_{osc}/4$ ，預除比為 1:8
  - 設定 CCP 中斷相關的中斷設定



## 需要了解的內容

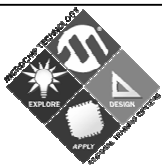
- 完成本練習需要的暫存器為：INTCON、T1CON和CCP1CON
- 提供了中斷向量
- 值 CCPR1L 將從 0 計數到 0xFF 後繼續遞減



## CCP 練習五解答

```
;
; Set CCP1CON to Output Compare mode with Special Event Trigger
; to clear the Timer 1 register pair on a match
;*****
    movlw          0x0B
    movwf          CCP1CON          ; set value in CCP1CON
;
; Configure Timer 1 for Fosc/4 operation. 8:1 Pre-scaler
;
;*****
    movlw          0x30
    movwf          T1CON
;

;
; Enable Timer 1 interrupts, Peripheral Interrupts and Global Interrupts
;
;*****
    bsf             PIE1,CCP1IE
    bsf             INTCON,GIE
    bsf             INTCON,PEIE
```



## 練習五問與答

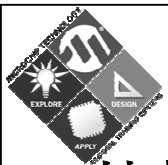
問：PWM工作不需要中斷。那麼在計時比較模式下是否需要中斷？

答：不必

- 我們在本練習中確實使用了中斷，那是因為我們在計時比較模式下的任務不在該問題所討論的範圍內

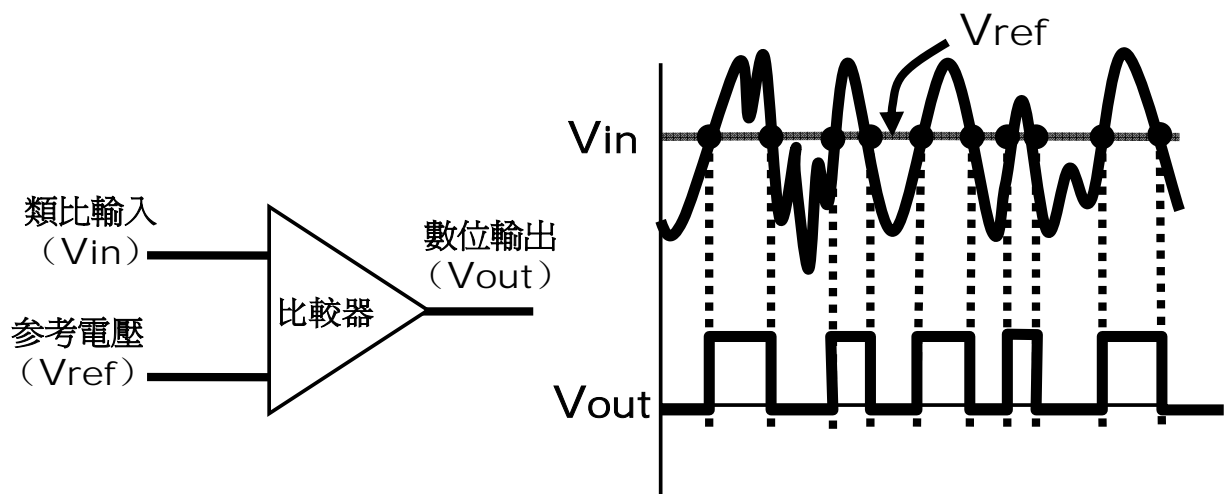


# 電壓比較器



## 電壓比較器概述

- 比較器模組：
  - 將類比輸入電壓與參考電壓做比較，並輸出數位電位結果
  - **PIC16F887**有**2**個比較器（**C1**和**C2**）

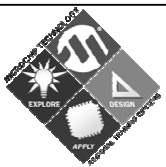




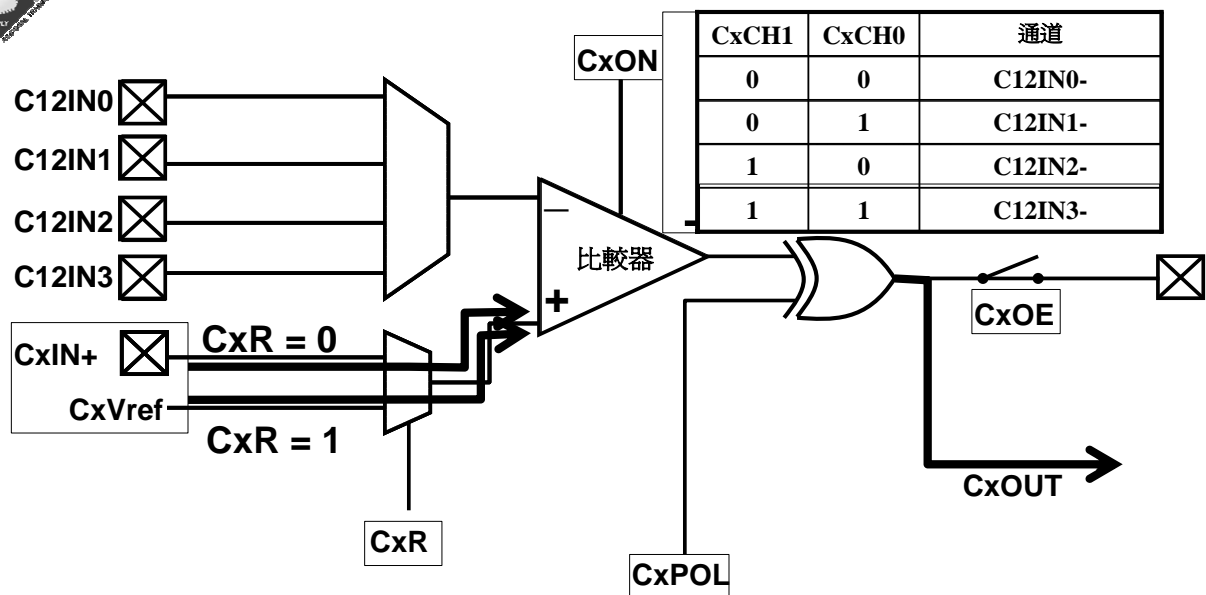


## 比較器模組暫存器

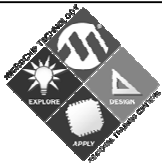
- 每個比較器（C1和C2）都有自己的控制暫存器
  - CM1CON0 和 CM2CON0
  - 比較器 2 還有一個 CM2CON1，用於與 Timer1 的連接
  - CMxCON0 暫存器控制以下操作
    - 啟動
    - 輸入選擇
    - 參考電壓選擇
    - 輸出選擇
    - 輸出極性



## 比較器簡化後方塊圖



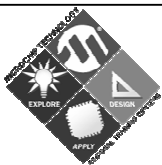
|      |       |      |       |     |     |       |       |
|------|-------|------|-------|-----|-----|-------|-------|
| CxON | CxOUT | CxOE | CxPOL | --- | CxR | CxCH1 | CxCH2 |
|------|-------|------|-------|-----|-----|-------|-------|



## 比較器模組暫存器 CM2CON1

|        |        |        |        |     |     |       |        |
|--------|--------|--------|--------|-----|-----|-------|--------|
| MC1OUT | MC2OUT | C1RSEL | C2RSEL | --- | --- | T1GSS | C2SYNC |
|--------|--------|--------|--------|-----|-----|-------|--------|

| 位元         | 功能   |
|------------|--|
| MC1OUT     | C1OUT位元的鏡像拷貝   |
| MC2OUT     | C2OUT位元的鏡像拷貝   |
| } 同時讀兩個比較器 |  |
| C1RSEL     | 1 = CVREF 連接到比較器C1的 C1VREF 輸入端<br>0 = 0.6V絕對參考電壓連接到 C1VREF |
| C2RSEL     | 1 = CVREF連接到比較器C2的 C2VREF 輸入端<br>0 = 0.6V絕對參考電壓連接到 C2VREF  |
| T1GSS      | 0 = 在比較器輸出時，Timer1 遞增； 1 = 計時器閘控信號隨着外部接腳上的輸入頻率遞增           |
| C2SYNC     | 1 = 比較器 2 輸出與 Timer1 時脈的下降緣同步<br>0 = 比較器輸出採非同步方式           |



## 比較器中斷

- 比較器中斷旗標 **PIR2** 暫存器中的 (**C1IF / C2IF**)
  - 相關比較器的輸出產生任何變化時設為 1
  - **PIE2**中的**C1IE/C2IE**以及**INTCON**中的**PEIE**和 **GIE** 必須設為 1
  - 中斷旗標必須用軟體方式清零，以致能後面的比較器中斷
    - 在清除**C1IF**或**C2IF**旗標之前必須先讀取比較器以清除旗標。



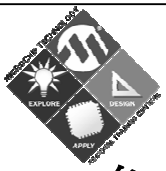
## 比較器和休眠模式

- 如果在執行**SLEEP**指令之前啟動了比較器，比較器將在休眠模式下繼續工作
- 比較器輸出電位的變化將喚醒 **PIC** 微控制器
- **C1IE/C2IE (PIE2)** 和 **PEIE (INTCON)** 必須設 1
  - 喚醒時執行**SLEEP**指令後的下一條指令或 **ISR**（如果有致能中斷）
  - 致能 **GIE (INTCON)** 轉而執行 **ISR**
- 任何重置將關閉比較器，使得所有暫存器回到內定的初始狀態



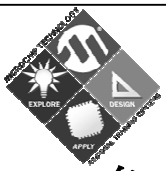
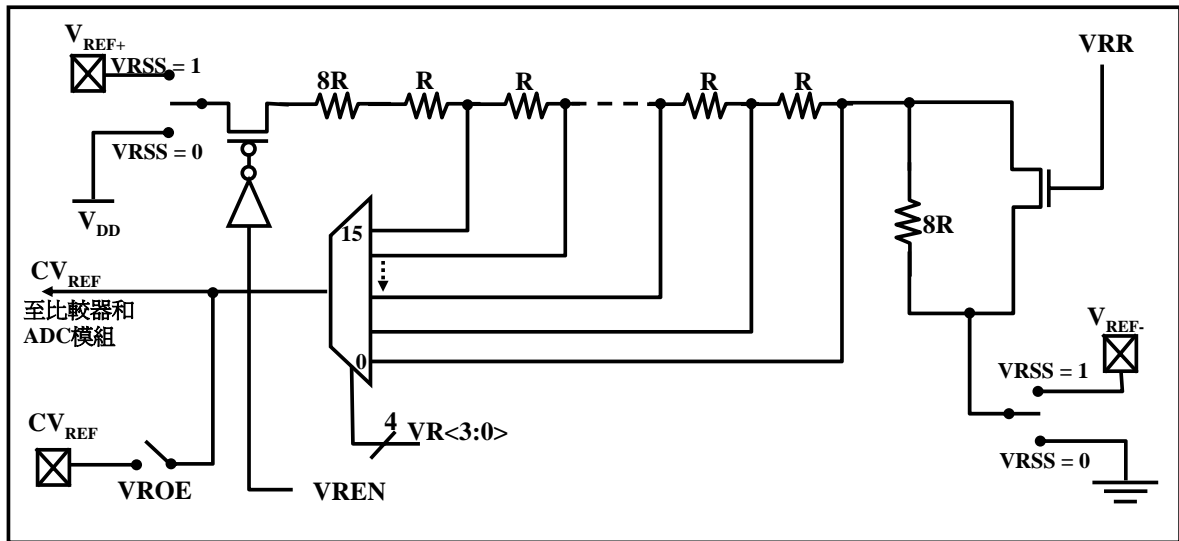
## 比較器參考電壓

- 參考電壓模組：
  - 獨立於比較器操作
  - 提供兩個具有**16** 種電壓選擇的電壓範圍
  - 輸出電壓被限制到 **V<sub>SS</sub>**
  - 提供與**V<sub>DD</sub>** 成比例的電壓
  - 固定參考電壓輸出 (**0.6V**)



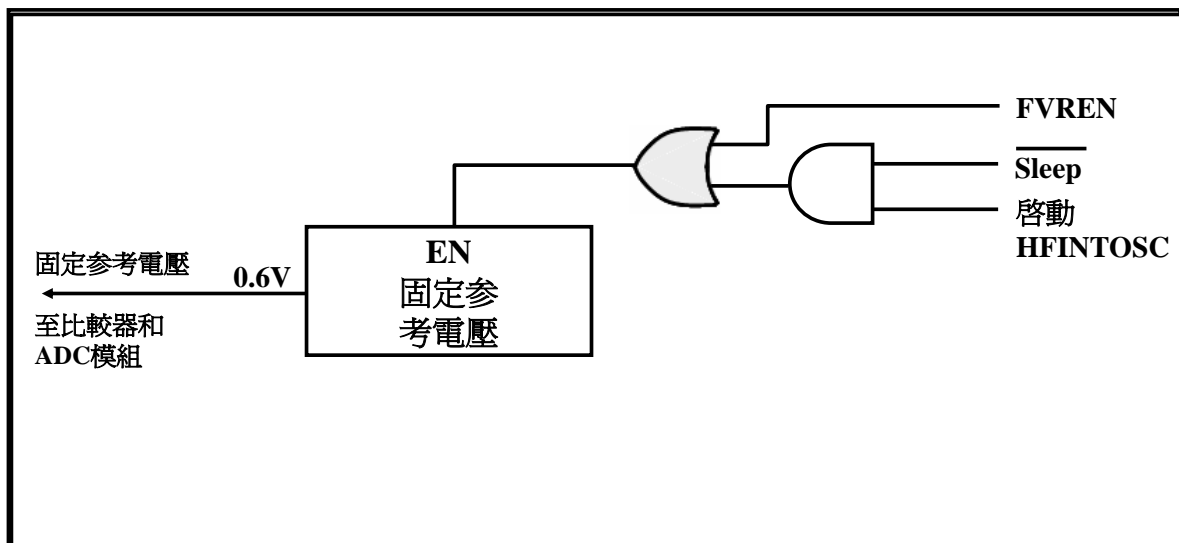
## 比較器參考電壓

- 參考電壓模組：



## 比較器參考電壓

- 參考電壓模組：

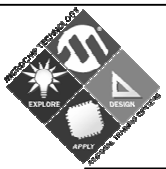




## 參考電壓控制暫存器 (VRCON)

| VREN | VROE | VRR | VRSS | VR3 | VR2 | VR1 | VR0 |
|------|------|-----|------|-----|-----|-----|-----|
|------|------|-----|------|-----|-----|-----|-----|

| 位元      | 功能   |
|---------|--|
| VREN    | 比較器1 參考電壓啟動位，1 = 啟動  |
| VROE    | 比較器 2 參考電壓啟動位元<br>1 = CVref 電壓也是 RA2/AN2/V <sub>REF</sub> /C2IN+ 接腳的輸出<br>0 = CVref 電壓與 RA2/AN2/V <sub>REF</sub> /C2IN+ 接腳斷開                                 |
| VRR     | CVref範圍選擇位元<br>1 = 低電壓範圍<br>0 = 高電壓範圍  |
| VRSS    | 範圍選擇位元<br>1 = 比較器參考電壓源，CV <sub>src</sub> = (V <sub>ref+</sub> ) – (V <sub>ref-</sub> )<br>0 = 比較器參考電壓源，CV <sub>src</sub> = V <sub>dd</sub> - V <sub>ss</sub> |
| VR<3:0> | CVref 值選擇位元  |



## 參考電壓控制暫存器 (VRCON)

| VREN | VROE | VRR | VRSS | VR3 | VR2 | VR1 | VR0 |
|------|------|-----|------|-----|-----|-----|-----|
|------|------|-----|------|-----|-----|-----|-----|

若 **VRR = 1** 或選擇了低電壓範圍：  

$$CV_{ref} = (VR<3:0>/24) \times V_{dd}$$

或

若 **VRR = 0** 或選擇了高電壓範圍：  

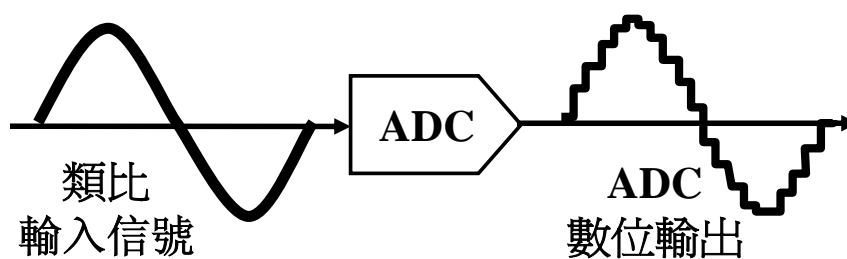
$$CV_{ref} = V_{dd}/4 + (VR<3:0>/32) \times V_{dd}$$



# 類比轉換器 (ADC)

## ADC 概述

- 類比轉換器模組
  - 將類比輸入信號轉換為8或10位二進制值
  - 可選的内部或外部參考電壓
  - 轉換完成後可以產生中斷
  - 中斷可用於將PIC微控器從休眠模式喚醒





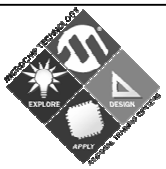
## ADC 暫存器

- ADC實現兩個控制暫存器
  - ADCON0和ADCON1

### ADCON0

| ADCS1 | ADCS2 | CHS3 | CHS2 | CHS1 | CHS0 | GO/DONE | ADON |
|-------|-------|------|------|------|------|---------|------|
|-------|-------|------|------|------|------|---------|------|

| 位元      | 功能  |
|---------|---|
| ADCSx位  | A/D 轉換時脈選擇位元<br>00 = $F_{osc}/2$ , 01 = $F_{osc}/8$ , 10 = $F_{osc}/32$ ,<br>11 = FRC (內部 RC 振盪器) |
| CHSx位   | 類比通道選擇位元  |
| GO/DONE | 1 = A/D 轉換正在進行<br>0 = A/D 轉換完成  |
| ADON    | 啟動 ADC 模組   |



## ADC暫存器

### ADCON1

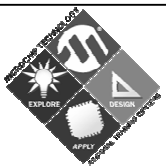
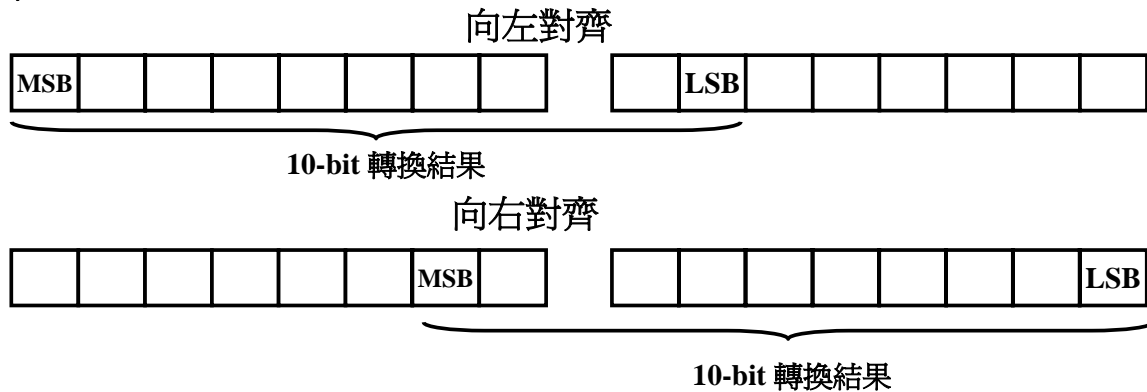
| ADFM | --- | VCFG1 | VCFG0 | --- | --- | --- | --- |
|------|-----|-------|-------|-----|-----|-----|-----|
|------|-----|-------|-------|-----|-----|-----|-----|

| 位元    | 功能                                     |
|-------|--|
| ADFM  | 轉換結果暫存器對齊方式位<br>1 = 向右對齊 , 0 = 向左對齊    |
| VCFG1 | 負參考電壓<br>1 = 來自Vref-接腳的外部電壓源 , 0 = Vss |
| VCFG0 | 正參考電壓<br>1 = 來自Vref+接腳的外部電壓源 , 0 = Vdd |

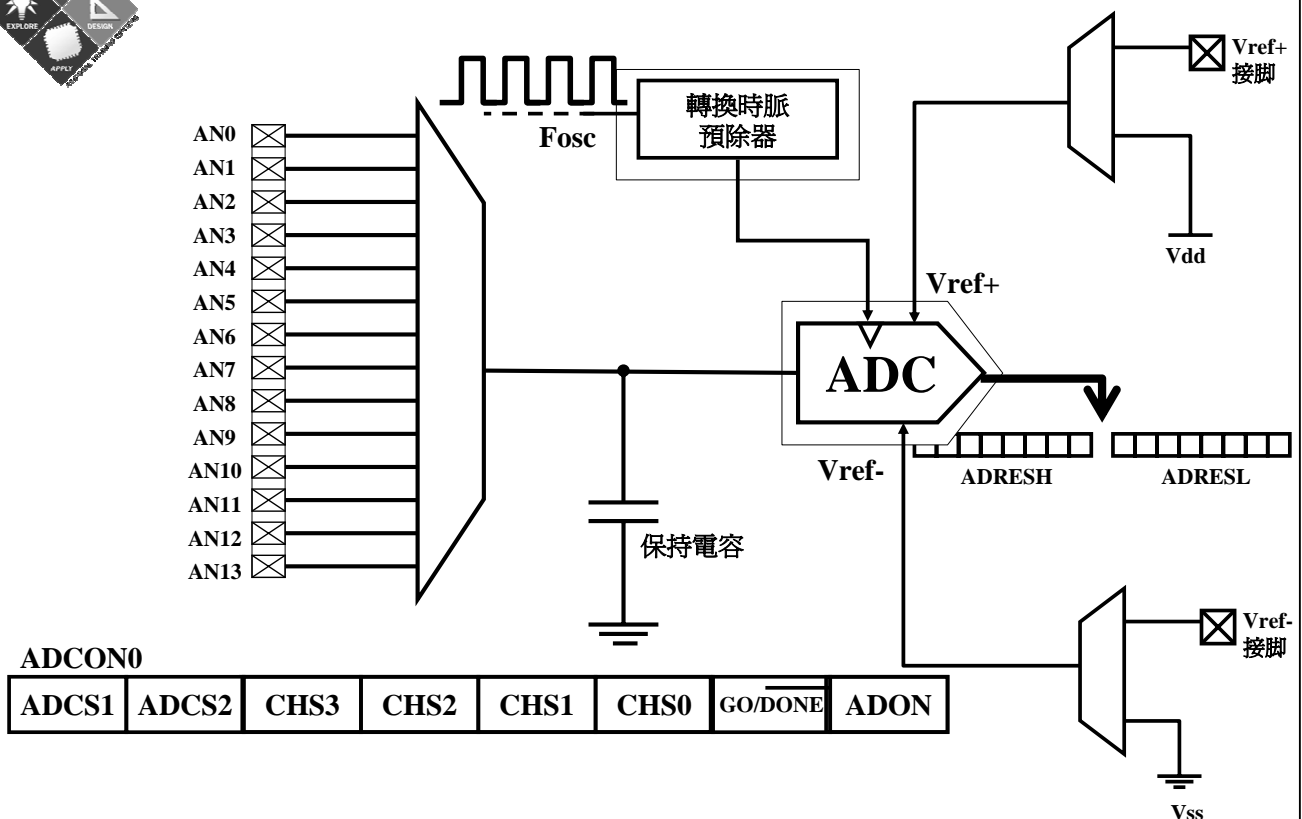


## ADC 暫存器

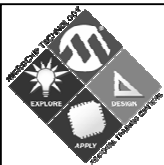
- 轉換完成後，ADC 轉換結果被放到個結果暫存器 ADRESH 和 ADRESL 中
- 10-bit ADC 轉換結果可以向左對齊也可向右對齊



## A/D模組方塊圖







# A/D模組方塊圖

| CHS3 | CHS2 | CHS1 | CHS0 | 通道   |
|------|------|------|------|------|
| 0    | 0    | 0    | 0    | AN0  |
| 0    | 0    | 0    | 1    | AN1  |
| 0    | 0    | 1    | 0    | AN2  |
| 0    | 0    | 1    | 1    | AN3  |
| 0    | 1    | 0    | 0    | AN4  |
| 0    | 1    | 0    | 1    | AN5  |
| 0    | 1    | 1    | 0    | AN6  |
| 0    | 1    | 1    | 1    | AN7  |
| 1    | 0    | 0    | 0    | AN8  |
| 1    | 0    | 0    | 1    | AN9  |
| 1    | 0    | 1    | 0    | AN10 |
| 1    | 0    | 1    | 1    | AN11 |
| 1    | 1    | 0    | 0    | AN12 |
| 1    | 1    | 0    | 1    | AN13 |

| ADCS1 | ADCS2 | 轉換時脈                        |
|-------|-------|-----------------------------|
| 0     | 0     | Fosc/2                      |
| 0     | 1     | Fosc/8                      |
| 1     | 0     | Fosc/32                     |
| 1     | 1     | F <sub>RC</sub> (專用內部RC振盪器) |

ADC

Vref-

ADRESH

ADRESL

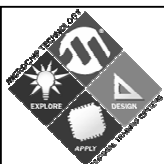
Vref+  
接脚

Vref-  
接脚

Vss

ADCON0

| ADCS1 | ADCS2 | CHS3 | CHS2 | CHS1 | CHS0 | GO/DONE | ADON |
|-------|-------|------|------|------|------|---------|------|
|-------|-------|------|------|------|------|---------|------|



# A/D模組方塊圖

AN0  
AN1  
AN2  
AN3  
AN4  
AN5  
AN6  
AN7  
AN8  
AN9  
AN10  
AN11  
AN12  
AN13

Fosc

轉換時脈  
預除器

VCFG0 = 0

Vref+

Vref+  
接脚

Vdd

ADC

保持電容

ADRESH

ADRESL

VCFG1 = 0

Vref-  
接脚

Vss

ADCON1

| 0 | VCFG1 | VCFG0 |
|---|-------|-------|
|---|-------|-------|



## ADC 時序注意事項

- 選擇了一個類比轉換通道時，必須花費一定的時間使保持電容充電
- 所有 **10-bit** 轉換的完成需要 **11** 個周期
- 用戶必須根據系統的頻率選擇適當的 ADC 的工作時脈 (**T<sub>ad</sub>**)

**HANDS-ON**

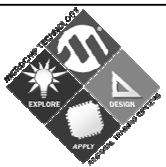
**Training**

**類比轉換練習  
(Lab 6)**



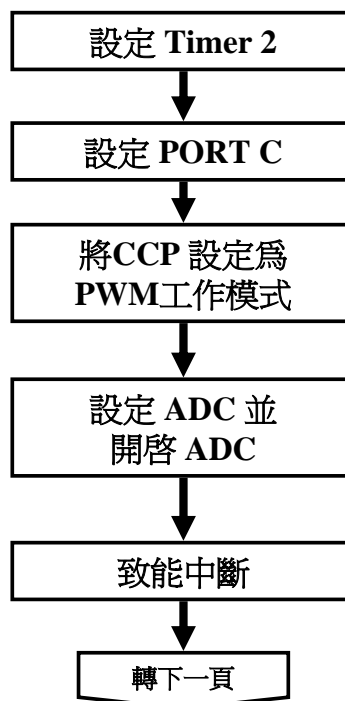
## 類比轉換器練習

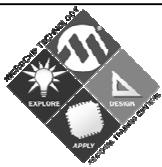
- 本練習將使您熟悉以下操作：
  - 設定ADC模組
  - 從“主”程式而非中斷向量對周邊進行操作
  - 使用從一個周邊（ADC）讀到的值來驅動另一個周邊（PWM模式下的ECCP）



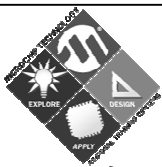
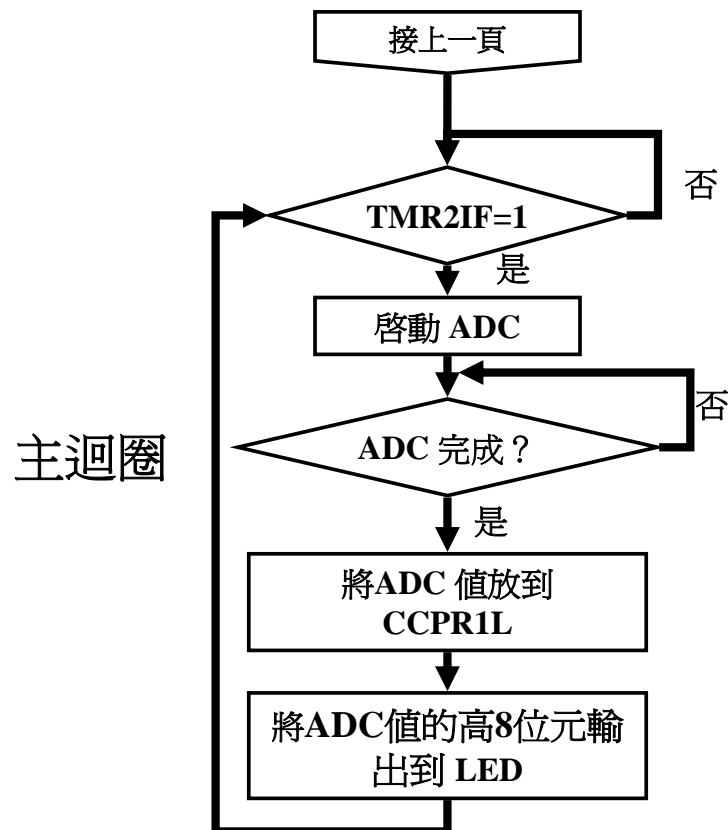
## ADC 練習六概述

### 主程式





## ADC 練習六概述（續）



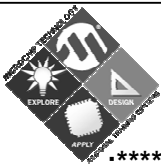
## 練習六細節

- 完成項目C:\RTC\201\_ASP\Lab6-ADC 中程式的以下部分
  - 設定ADC以向左對齊的值返回到 ADRESH
  - 將  $T_{ad}$  設置為  $T_{osc} * 8$
  - 開啓 ADC 模組
  - 完成此程式以啓動 ADC，並在主控制循環中等待轉換結束



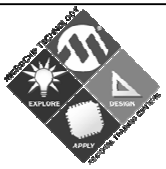
## 需要了解的內容

- 本練習不使用中斷方式執行ADC轉換，而是使用查詢方法。
- 將ADC轉換的結果值寫入CCPR1L 將更改蜂鳴器的脈衝寬度
- 使用ADCON1和ADCON0 特殊功能暫存器完成本練習



## ADC 練習六解答

```
*****
;
;  Configure ADC , Channel 0, left justified, Tad=8 * Tosc, turn on ADC
;  *****
;
    clrf      ADCON0          ; ensure default Channel is set to channel 0
    bsf      ADCON0,ADCS1     ; set Tad = 8 Tosc
    bsf      ADCON0,ADON      ; turn on ADC unit
    bsf      STATUS,RP0       ; go to bank 1
    movlw    0x0E             ; Left Justify and set configuration
    movwf    ADCON1
;
;  Enable Timer 2 interrupts, Peripheral Interrupts and Global Interrupts
;
    bsf      PIE1,TMR2IE
    bsf      INTCON,GIE
    bsf      INTCON,PEIE
    bcf      STATUS,RP0       ; return to bank 0
;
loop
;
;  *****
;  ; add three lines of code to start the ADC conversion and wait for the conversion
;  ; to complete
;  *****
;
    bsf      ADCON0,GO         ; start A-to-D conversion on channel 0
    btfsc    ADCON0,GO         ; Is the conversion done?
    goto     $-1               ; No: Check again
```



## ADC 練習六問與答

問：不在主程式中等待 TMR2IF 被設為 1，能否從中斷副程式中啟動 ADC 轉換？

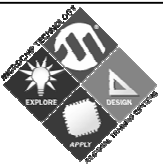
答：可以

**增強型  
泛用同步非同步串列收發器  
(EUSART)**



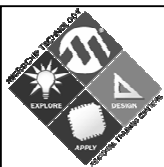
## EUSART 概述

- 串列式 I/O 通信周邊
  - 有時也稱為“串列通信介面”或 SCI
- 主要功能：
  - 同步或非同步模式
  - 可以接收或發送
    - 全雙工非同步發送和接收 (UART)
    - 半雙工同步主模式和從模式
- 最常使用
  - RS-232 與 PC 串列端通信
    - 需要用 RS-232 電位轉換的驅動器 (Max232)
- 增加 LIN Bus 介面
- 非同步接收自動速率偵測功能



## EUSART 暫存器

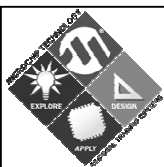
- EUSART 所使用的暫存器：
  - 速率產生暫存器：
    - SPBRG 和 SPBRGH
  - 發送狀態和控制暫存器 (TXSTA)
  - 接收狀態和控制暫存器 (RCSTA)
  - 接收和發送資料暫存器
    - 發送資料暫存器 (TXREG)
    - 接收資料暫存器 (RCREG)



## TXSTA暫存器

| CSRC | TX9 | TXEN | SYNC | SENB | BRGH | TRMT | TX9D |
|------|-----|------|------|------|------|------|------|
|------|-----|------|------|------|------|------|------|

| 位元   | 功能  |
|------|---|
| CSRC | 時鐘源選擇位<br>1 = 主模式（由內部 BRG產生時鐘信號）<br>0 = 從模式（由外部時鐘源產生時鐘信號） |
| TX9  | 第9位發送啟動位  |
| TXEN | 1 = 發送啟動  |
| SYNC | EUSART模式，1 = 同步模式，0 = 非同模式                                |
| SENB | 1 = 發送同步間隔字符位<br>0 = 同步間隔字符發送完成                           |
| BRGH | 速率選擇位，1 = 高速，0 = 低速                                       |
| TRMT | 1 = 發送移位暫存器（TSR）為空，0 = TSR滿<br>指示最後一位何時移出                 |
| TX9D | 發送資料的第9位元   |

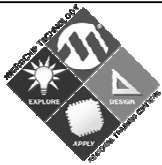


## RCSTA 暫存器

| SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
|------|-----|------|------|-------|------|------|------|
|------|-----|------|------|-------|------|------|------|

| 位元    | 功能  |
|-------|---|
| SPEN  | 串列訊界面啟動位元<br>1 = 啟動串列介面（將RX/DT和TX/CK接腳設定為串列介面接腳）<br>0 = 關閉串列介面（保持在重置狀態） |
| RX9   | 1 = 啟動 9-bit 資料接收，0 = 8-bit 資料  |
| SREN  | 同步模式 – 主模式，1 = 啟動，0 = 禁止單位元組接收  |
| CREN  | 連續接收啟動位元  |
| ADDEN | 1 = 啟動 9-bit Address 檢測（致能中斷並在RSR<9> =1 時裝載入接收緩衝器）                      |
| FERR  | 1 = 發生Frame 錯誤（未檢測到 Stop Bit）   |
| OERR  | 1 = 發生溢位錯誤（接收資料發生重疊的現象）   |
| RX9D  | 接收資料的第 9-bit  |





## 速率控制暫存器 (BAUDCTL)

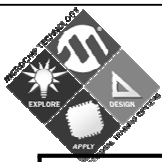
|        |       |      |      |       |      |     |       |
|--------|-------|------|------|-------|------|-----|-------|
| ABDOVF | RCIDL | ---- | SCKP | BRG16 | ---- | WUE | ABDEN |
|--------|-------|------|------|-------|------|-----|-------|

| 位元     | 功能  |
|--------|---|
| ABDOVF | 自動速率檢測溢位 (僅用於非同步模式)<br>1 = 自動速率計時器發生溢位  |
| RCIDL  | 接收器空閒旗標, 1 = 接收器空閒中, 0 = 接收到了起始位, 接收器正在接收   |
| SCKP   | 同步時序極性位元<br>非同步模式: 1 = 將取1's 補數後的資料發送到RB7/TX/CK接腳<br>同步模式: 1 = 在時脈的上升緣傳輸資料<br>0 = 在時脈的下降緣傳輸資料 |
| BRG16  | 16-bit 速率發生器<br>1 = 選擇16-bit BRG, 0 = 選擇 8-bit BRG  |
| WUE    | 喚醒啟動位元 (僅用於非同步模式)   |
| ABDEN  | 自動速率檢測啟動位元, 1 = 啟動<br>在休眠模式下, 當第 9 位元被設為 1 時, 進行檢測  |

© 2007 Microchip Technology Incorporated. All Rights Reserved.

201ASP

Slide 145



## 速率公式

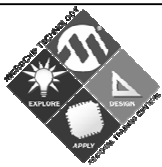
| 設定位元            |                    |                 | BRG/<br>EUSART<br>模式 | 速率公式                 |
|-----------------|--------------------|-----------------|----------------------|----------------------|
| SYNC<br>(TXSTA) | BRG16<br>(BAUDCTL) | BRGH<br>(TXSTA) |                      |                      |
| 0               | 0                  | 0               | 8-bit /非同            | $F_{osc}/[64 (n+1)]$ |
| 0               | 0                  | 1               | 8-bit /非同            | $F_{osc}/[16 (n+1)]$ |
| 0               | 1                  | 0               | 16-bit /非同           |                      |
| 0               | 1                  | 1               | 16-bit /同步           | $F_{osc}/[4 (n+1)]$  |
| 1               | 0                  | X               | 8-bit /同步            |                      |
| 1               | 1                  | X               | 16-bit /同步           |                      |

\*n = SPBRGH:SPBRG 暫存器對的值

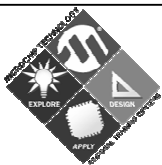
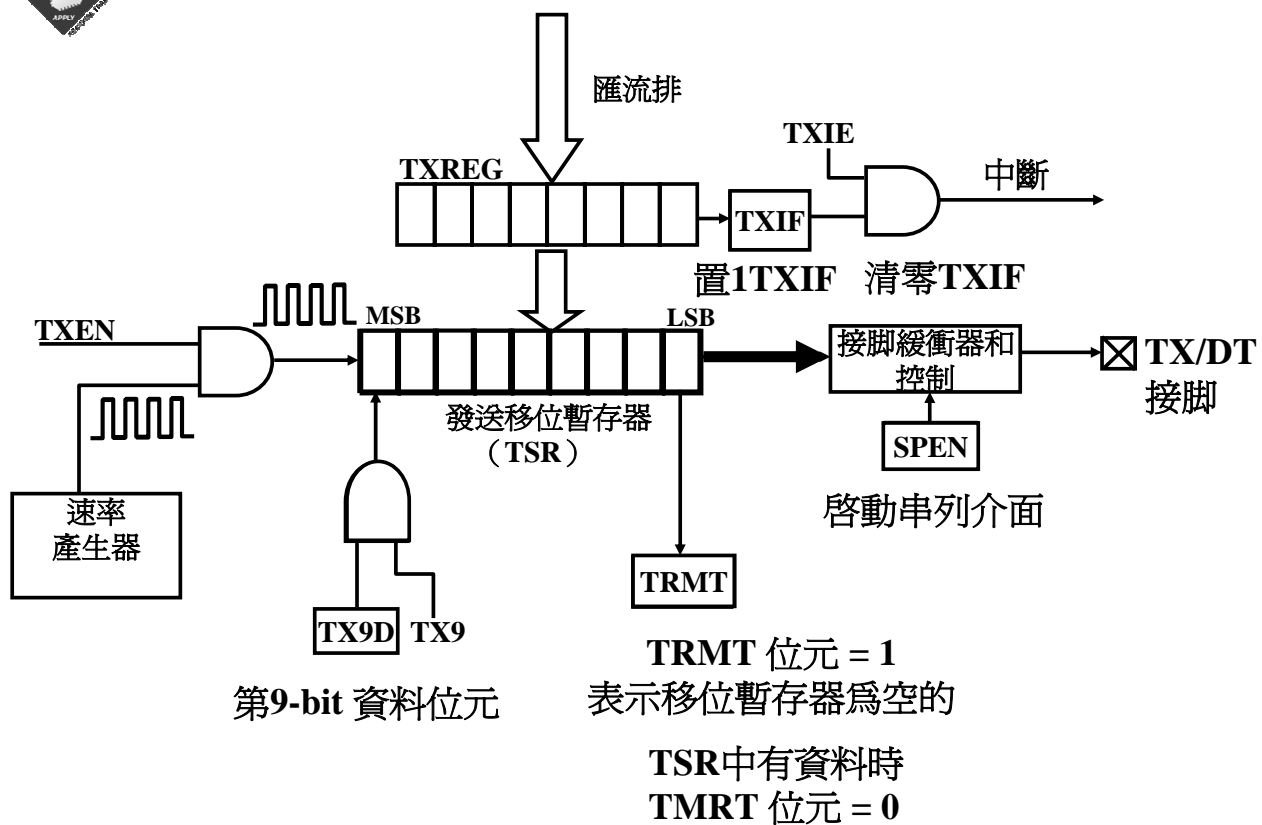
© 2007 Microchip Technology Incorporated. All Rights Reserved.

201ASP

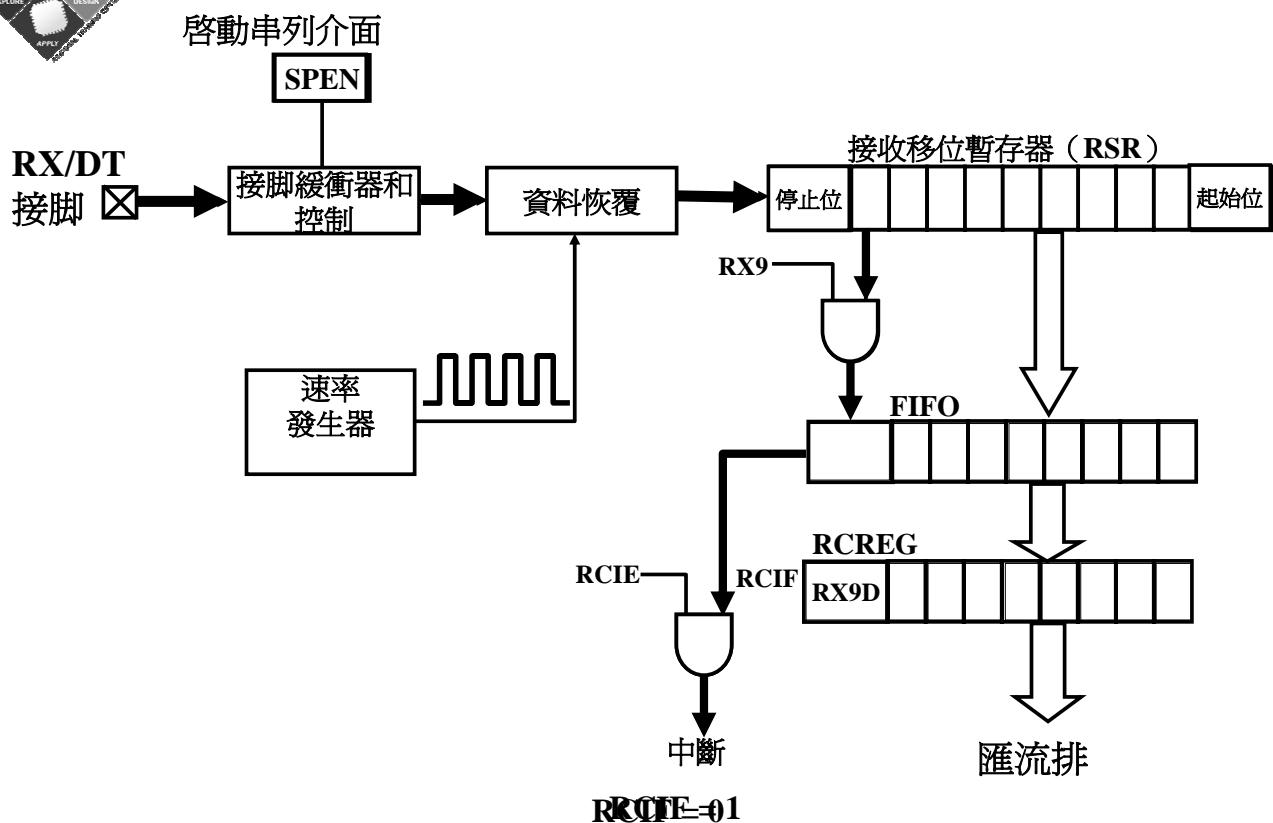
Slide 146



## 發送端方塊圖



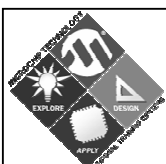
## 接收端方塊圖





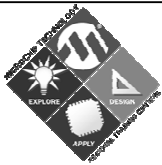
# **EUSART 接收與發送**

## **(Lab7)**



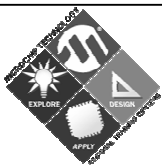
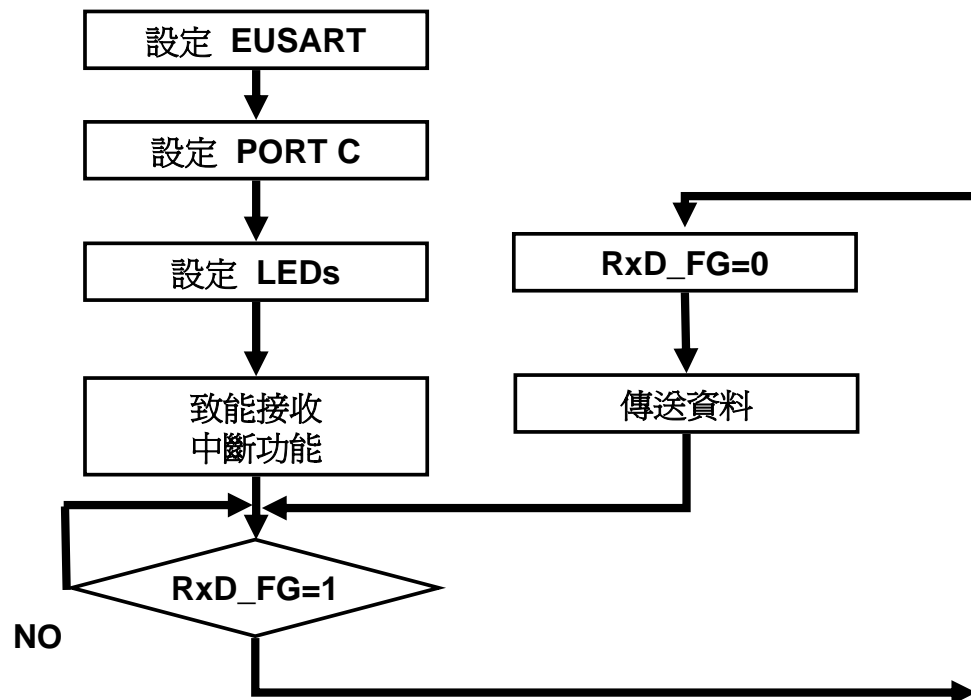
## **EUSART 實驗七**

- 本實驗可以让你了解：
  - 設定 EUSART 模組工作於 9600,N,8,1
  - 利用接收中斷服務副程式的方式，接收終端機透過 RS-232 所傳送的資料
  - 主程式在主迴圈等待”接收旗號”被設置後傳送資料到 EUSART



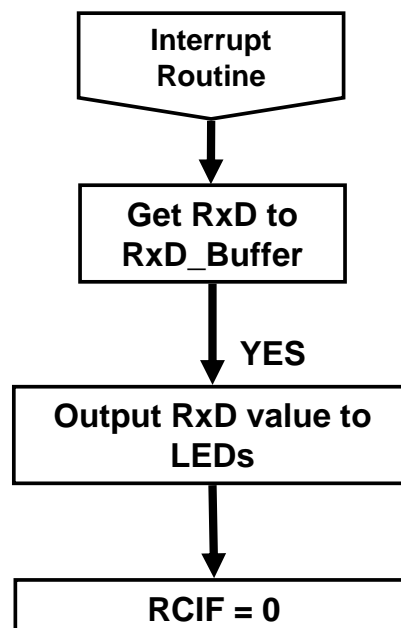
## USART 實驗七 主程式流程

### 主程式



## USART 實驗七 中斷程式流程

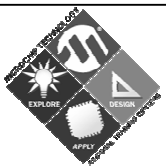
### 接收中斷副程式





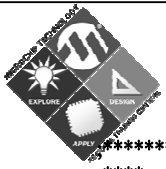
## Lab7 說明

- 本實驗的程式方在以下的目錄：  
C:\RTC\201\_ASP\Lab7-EUSART
  - 設定 USART 在非同步模式
    - 9600 bps, None Parity Check, 8-bit Data, 1 Stop bit
    - $4\text{MHz}/16/(25+1)=9615\text{ bps}$  , SPBRG = 25
  - Set PORTC for TxD & RxD
  - Set PORTD for LEDs output
  - Turn on USART module
  - RS-232 資料接收採中斷方式，接收後再交由主程式傳送到 RS-232



## 必須了解的是

- 接收採用中斷方式來達成即時接收，以取代 polling 的方法
- 中斷裡使用設定 real-time flag 的方式通知主程式處理事件
- 設定標準的 RS-232 通訊協定
- 使用 PC 標準的串列通訊介面 COMx 配合超級終端機 ( Hyper-Terminal ) 的軟體來驗證



## USART 實驗七解答

```
*****
;****
;***** Initial USART as 9600,N,8,1 *****
;*****
Init_USART
    banksel    BAUDCTL                ; Bank 3
    movlw     b'00000000'            ; disable Auto-Baud Detect, TxD is RC6, BRG = 8-bit
    movwf     BAUDCTL

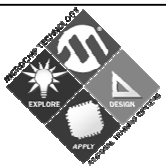
    banksel    TXSTA                  ; ### Bank 1
    movlw     b'00100100'            ; ### 8-bit data mode , ASYNC
    movwf     TXSTA                  ; ### High Speed mode, Enable TxD
;
    movlw     .25                     ; ### Set baud rate at 9600 with High Speed mode
    movwf     SPBRG                  ; ### System Clock are 4MHz using internal RC

    bcf       PIE1,TXIE              ; ### Disable TxD interrupt
    bsf       PIE1,RCIE              ; ### Enable RxD interrupt

    bsf       TRISC,7                ; ### set input for RC7, RxD receiving pin
    bcf       TRISC,6                ; ### set output for RC6, TxD pin

    banksel    0
    movlw     b'10010000'            ; ### Enable Serial Port, 8-bit receive
    movwf     RCSTA                  ; ### Continuous Receive, Disable Address Detection
;
    bcf       PIR1,TXIF              ; Clear TxD interrupt flag;
    bcf       PIR1,RCIF              ; Clear RxD interrupt flag

;
    bsf       INTCON,PEIE
    bsf       INTCON,GIE
```



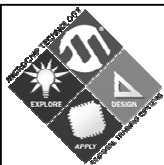
## USART 實驗七問與答

問：在本實驗中傳送資料在主程式下完成，可否利用中斷副程式發送資料呢？

答：可以

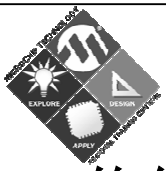


## 主同步串列介面 (**MSSP**) 模組



### 概述

- MSSP 模組有以下兩種工作模式：
  - 串列周邊介面 (SPI)
  - I<sup>2</sup>C<sup>TM</sup>
    - 完全主(Master)模式
    - 從(Slave)模式 (帶有廣播地址呼叫)
- I<sup>2</sup>C 介面通過硬體可以支援以下模式：
  - 主模式 (Master Mode)
  - 多主機模式 (Multi-Master Mode)
  - 從模式 (Slave Mode)



## I<sup>2</sup>C 訊號條件

### ● 條件:

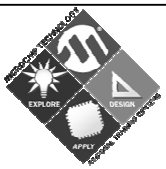
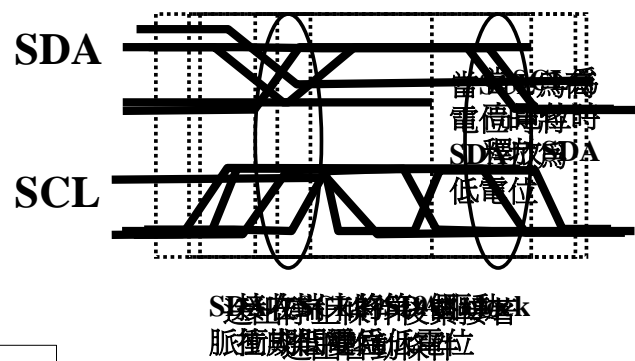
- 啟動 (S)

- 停止 (P)

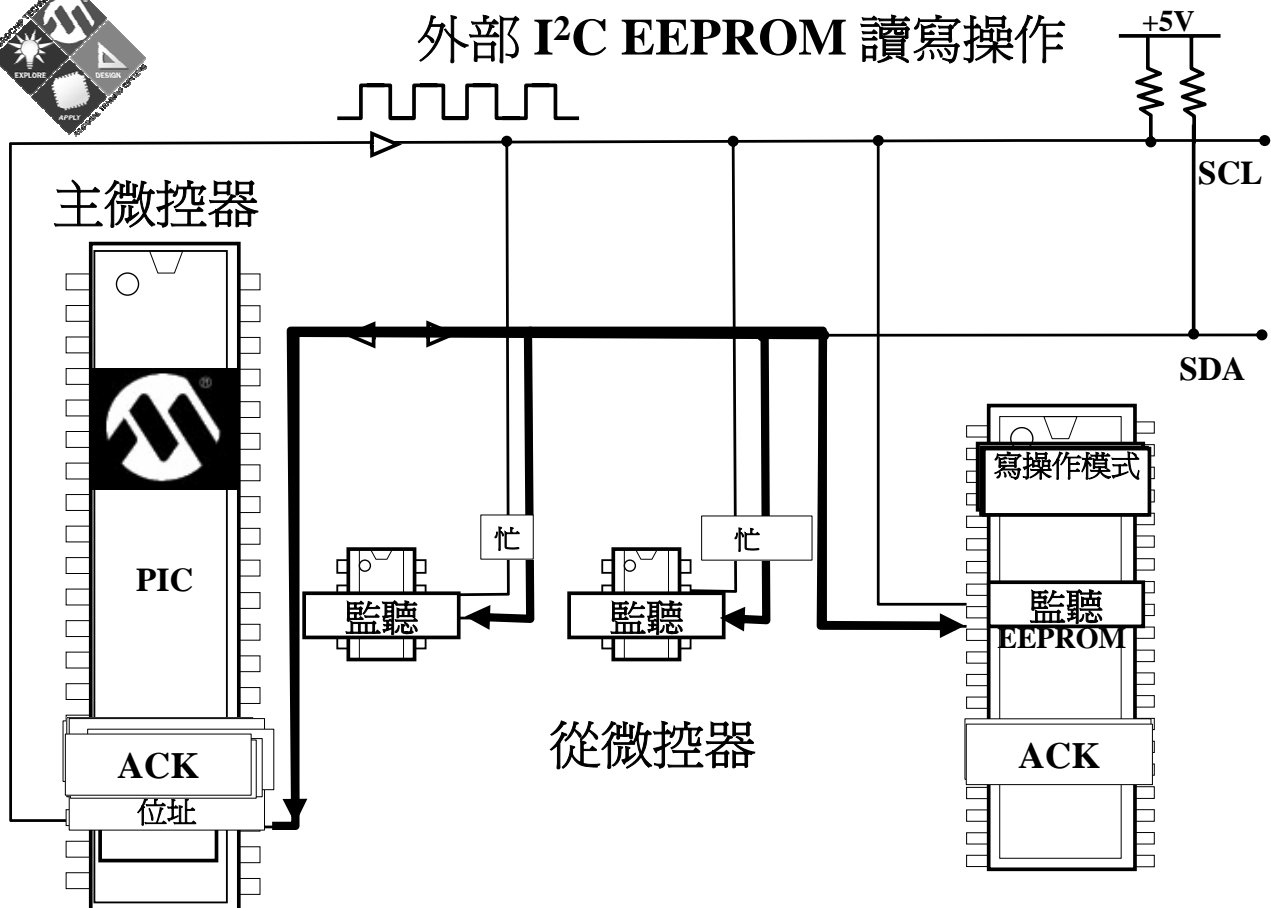
- 應答 (A)

- 重覆啟動 (R)

- 否定或無應答 (N)



## 外部 I<sup>2</sup>C EEPROM 讀寫操作







## MSSP 控制暫存器 (一)

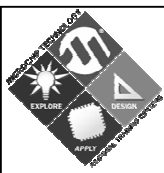
- 有3 個相關的控制暫存器
- MSSP 狀態暫存器 (SSPSTAT)

|     |     |              |   |   |              |    |    |
|-----|-----|--------------|---|---|--------------|----|----|
| SMP | CKE | D/ $\bar{A}$ | P | S | R/ $\bar{W}$ | UA | BF |
|-----|-----|--------------|---|---|--------------|----|----|

☐ 控制位元

☐ 檢測位元 (旗標)

| 位元           | 功能                      |
|--------------|-------------------------|
| SMP          | 斜率控制位元                  |
| CKE          | I <sup>2</sup> C 模式下不使用 |
| D/ $\bar{A}$ | Rx/Tx的最後一個位元組是資料還是地址    |
| P            | 檢測到停止條件                 |
| S            | 檢測到啟動條件                 |
| R/ $\bar{W}$ | 從微控制器：讀/寫或主微控制器 = 正在發送  |
| UA           | 地址需要更新                  |
| BF           | SSPBUF暫存器滿              |



## MSSP 控制暫存器(二)

- MSSP控制暫存器1 (SSPCON)

|      |       |       |     |       |       |       |       |
|------|-------|-------|-----|-------|-------|-------|-------|
| WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 |
|------|-------|-------|-----|-------|-------|-------|-------|

☐ 控制位元

☐ 錯誤檢測位元 (FLAGS)

| 位元    | 功能  |
|-------|---|
| WCOL  | 檢測到寫入衝突                                     |
| SSPOV | 接收緩衝器 SSPBUF 發生資料覆蓋                         |
| SSPEN | 啟動 MSSP 模組                                  |
| CKP   | 在 Slave Mode : = 0, SCK 拉 Low ; = 1 時，致能SCK |
| SSPM3 | 工作模式選擇位元                                    |
| SSPM2 |   |
| SSPM1 |   |
| SSPM0 |   |



## MSSP 控制暫存器(二)

| SSPM3 | SSPM2 | SSPM1 | SSPM0 | 模式                                      |
|-------|-------|-------|-------|---|
| 0     | 0     | 0     | 0     | SPI主模式，時鐘 = FOSC/4                      |
| 0     | 0     | 0     | 1     | SPI主模式，時鐘 = FOSC/16                     |
| 0     | 0     | 1     | 0     | SPI主模式，時鐘 = FOSC/64                     |
| 0     | 0     | 1     | 1     | SPI主模式，時鐘 = TMR2輸出/2                    |
| 0     | 1     | 0     | 0     | SPI從模式，時鐘 = SCK接腳，啟動SS接腳控制              |
| 0     | 1     | 0     | 1     | SPI從模式，時鐘 = SCK接腳，禁止SS接腳控制，SS可用作 I/O 接腳 |
| 0     | 1     | 1     | 0     | I2C從模式，7-bit Address Mode               |
| 0     | 1     | 1     | 1     | I2C從模式，10-bit Address Mode              |
| 1     | 0     | 0     | 0     | I2C主模式，時鐘 = FOSC / (4 * (SSPADD+1))     |
| 1     | 0     | 0     | 1     | 保留                                      |
| 1     | 0     | 1     | 0     | 保留                                      |
| 1     | 0     | 1     | 1     | I2C 固件控制的主模式（從微控制器空閒）                   |
| 1     | 1     | 0     | 0     | 保留                                      |
| 1     | 1     | 0     | 1     | 保留                                      |
| 1     | 1     | 1     | 0     | I2C從模式，7-bit Address，並致能起始位元和停止位元中斷     |
| 1     | 1     | 1     | 1     | I2C 從模式，10-bit Address，並致能起始位元和停止位元中斷   |



## MSSP 控制暫存器 (三)

### 3. MSSP 控制暫存器 2 (SSPCON2)

| GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN |
|------|---------|-------|-------|------|-----|------|-----|
|------|---------|-------|-------|------|-----|------|-----|

☐ 控制位元

☐ 檢測位元（旗標）

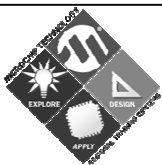
| 位元      | 功能                        |
|---------|---------------------------|
| GCEN    | 接收到共用位址時(0x00)產生中斷（從模式）   |
| ACKSTAT | 0 = 接收到來自從微控制器的應答信號（發送模式） |
| ACKDT   | 0 = ACK, 1 = NACK（接收模式）   |
| ACKEN   | 發出 ACK/NACK 條件（發送ACKDT位）  |
| RCEN    | 啟動接收模式                    |
| PEN     | 發出停止條件                    |
| RSEN    | 發出重複啟動條件                  |
| SEN     | 發出啟動條件                    |



## 與 I<sup>2</sup>C 相關的暫存器

### 4. MSSP 接收/發送緩衝器 (SSPBUF)

- 保存要發送的資料或MSSP模組接收到的資料
- 緩衝器滿時，SSPSTAT暫存器中的BF(緩衝器已滿)位原會設為1
- 在發送/接收資料期間，忽略任何寫SSPBUF暫存器的操作，並且SSPCON暫存器中的寫入衝突檢測位WCOL設為1



## 與 I<sup>2</sup>C 相關的暫存器

### 4. I<sup>2</sup>C Slave Address (SSPADD) :

- **Slave Mode :**
  - 設定 PIC 微控器的 I<sup>2</sup>C Slave Address
  - 與接收到的位址值進行比較
- **Master Mode :**
  - 用於計算 I<sup>2</sup>C 系統的時鐘速率 (速率)

$$\text{速率} = \frac{F_{osc}}{4 \times (SPADD + 1)}$$

\*注：F<sub>osc</sub> 是振盪器頻率，而非指令週期 T<sub>CY</sub>



## MSSP 中斷

- 發生以下事件時，**PIR1**暫存器中的 **SSPIF** 中斷旗標會設 1
  - 啓動條件
  - 停止條件
  - 資料傳輸位元組已發送/已接收
  - 發送應答
  - 重覆啓動條件

當設定了 **PIE1<SSPIE>** 位元以及 **INTCON** 中的 **GIE** 和 **PEIE** 位元時，才會產生 **SSP** 的中斷。

**HANDS-ON Training**

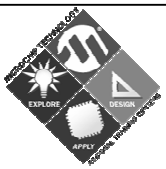
**MSSP Module**  
**I2C 實驗**  
**(Lab8)**

MICROCHIP TECHNOLOGY  
EXPLORE DESIGN APPLY  
REGIONAL TRAINING CENTERS



## I<sup>2</sup>C 實驗八

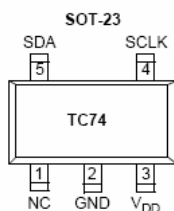
- 本實驗可以讓你了解：
  - 設定 MSSP 模組工作於：
    - 7-bit I<sup>2</sup>C Master mode
    - 100KHz bus rate
  - 了解 I<sup>2</sup>C 的命令程序，一步接一步說明
  - 讀取 I<sup>2</sup>C 界面的溫度感應器 TC74 的溫度值後顯示在 LEDs



## TC74 命令格式

### Read Temperature I2C command from the TC74-A7

| s | TC74 Addr | Wr | Ack | Read Temp. Command | Ack | R S | TC74 Addr | Rd | Ack | Temp. Data | NAck | P |
|---|-----------|----|-----|--------------------|-----|-----|-----------|----|-----|------------|------|---|
|   | 1001111   | 1  | 0   | 00000000           | 0   |     | 1001111   | 0  | 0   | 00010011   | 1    |   |



TC74 Package

| Actual Temperature | Registered Temperature | Binary Hex |
|--------------------|------------------------|------------|
| +130.00°C          | +127°C                 | 0111 1111  |
| +127.00°C          | +127°C                 | 0111 1111  |
| +126.50°C          | +126°C                 | 0111 1110  |
| +25.25°C           | +25°C                  | 0001 1001  |
| +0.50°C            | 0°C                    | 0000 0000  |
| +0.25°C            | 0°C                    | 0000 0000  |
| 0.00°C             | 0°C                    | 0000 0000  |
| -0.25°C            | -1°C                   | 1111 1111  |
| -0.50°C            | -1°C                   | 1111 1111  |
| -0.75°C            | -1°C                   | 1111 1111  |
| -1.00°C            | -1°C                   | 1111 1111  |
| -25.00°C           | -25°C                  | 1110 0111  |

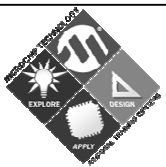


## 實驗八說明

- 本實驗的程式方在以下的目錄：

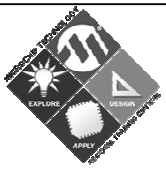
C:\RTC\201\_ASP\Lab8-I2C

- 設定 MSSP 模組工作於 7-bit I<sup>2</sup>C Master Mode
- 設定 SCL & SDA 為輸入腳位
- 發送 I<sup>2</sup>C 讀取溫度的命令到 TC74-A7 以取的現在的溫度的值
- 將溫度值顯示在 LEDs



## 需要了解的是

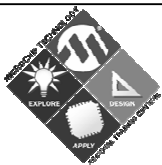
- 讀取溫度的命令類似於 I<sup>2</sup>C EEPROM ( 24LCxx) 的 Random Read Command
- 利用 SSPIF 來檢查每一動作的完成
- 是那一元件會送出 Ack/NAck 的訊息，誰該偵測此一訊號，如果沒有去偵測會怎樣？
- Bus Collision interrupt bit (BCLIF) 的處理



# USART 實驗八解答

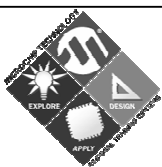
```
Init_I2C_Master
    BANKSEL TRISC          ; Initial PortC, bit 3 & 4 as Input
    bsf      SCL           ; RC3 = SCL , RC4 = SDA
    nop
    bsf      SDA
;
    BANKSEL 0
    movlw   b'00101000' ; ### I2C Master Mode, Clock Rate: FOSC/(4*SSPADD+1)
    movwf   SSPCON        ; ###
;
    banksel SSPADD
    movlw   .9             ; ### This gives 100KHz I2C clock @ 4MHz
    movwf   SSPADD        ; ### (4MHz/4) / (9+1)= 100KHz
;
    movlw   b'10000000'    ; ### Disable slew rate control,
    movwf   SSPSTAT        ; ### and clear status bits
;
    movlw   b'00000000'    ; Set SCL, SDA into Ready status
    movwf   SSPCON2
;
    return
```

**附錄 A：**  
**多中斷練習**  
**(LAB9)**



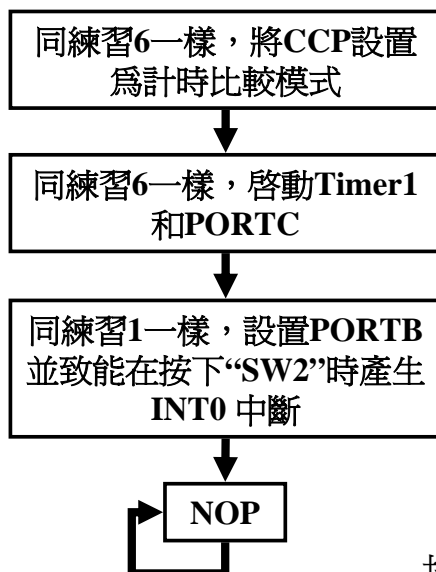
## 多中斷練習

- 本練習包括：
  - 處理 2 個（或更多個）同時發生的中斷
  - 確定中斷源
  - 決定那一個中斷請求會先被執行



## 練習概述

### 主程式

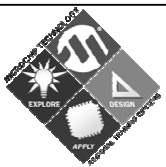
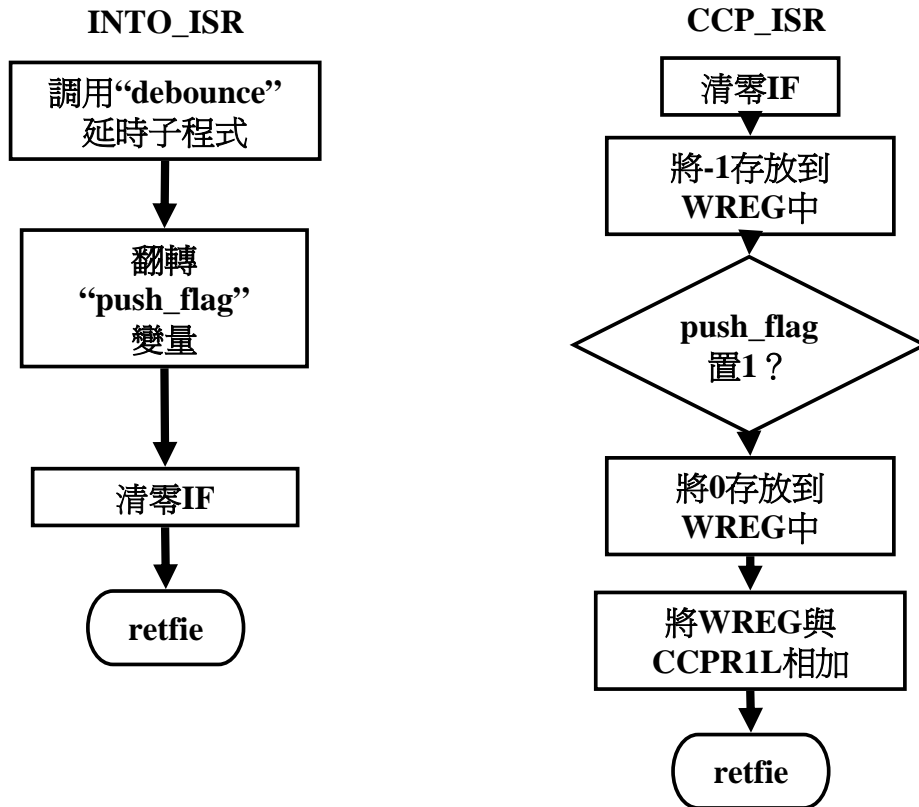


接下一張投影片

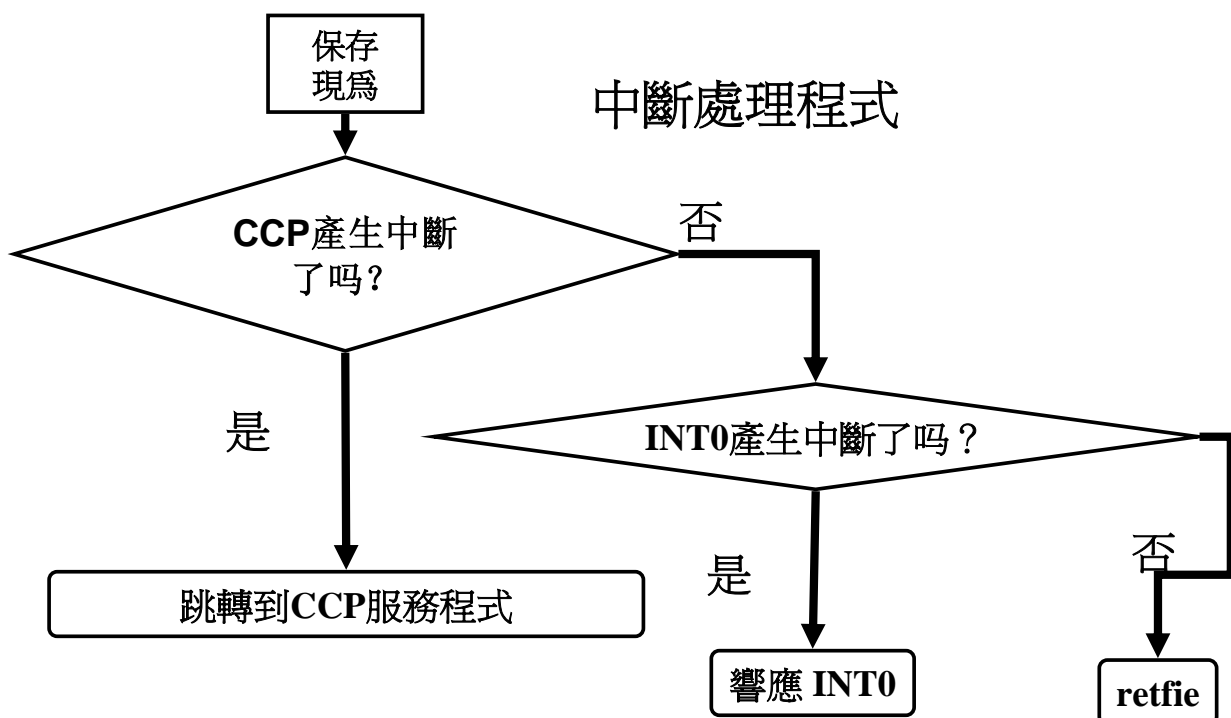




## 練習概述



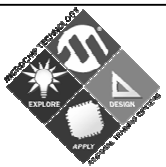
## 練習概述（續）





## 練習細節

- 本練習位於：
  - C:\RTC\201\_ASP\Lab8-MXINT
- 提供了兩個中斷服務程式 (INT0\_ISR 和 CCP\_ISR)
- 完成程式的以下部分
  - 中斷發生時，確定中斷來源並將控制權轉到相對應的中斷服務程式 (ISR)
  - 設置 SFR 以致能產生 INT0 和 CCP1 中斷



## 本練習中需要了解的內容

- 本練習使用了 INTCON 和 PIR 特殊功能暫存器

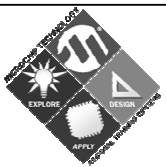


## 練習九解決方案

中斷向量解析部分

```
NT_VECTOR CODE    0x004    ; interrupt vector location
;
;   Save Wreg, STATUS, and PCLATH during Interrupt Service
;
    call    save_regs;

    btfsc   INTCON,INTF    ; test for INT0 interrupt
    goto    INT0_ISR
    btfsc   PIR1,CCP1IF    ; test for CCP Interrupt request
    goto    CCP_ISR
;
```



## 練習九問與答

問：為什麼在按下 SW2 時會非常安靜？

答：因為在中斷期間呼叫了“debounce”副程式，並清除了 GIE 位元，在這段時間內不致能產生反轉蜂鳴器的 CCP1 中斷。所以蜂鳴器沒有發出聲音

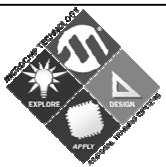


## 練習九問與答（續）

問：如何取消靜音，並使蜂鳴器繼續運行？

答：

1. 在主程式中捕獲 SW2 按下事件，並在GIE 設為 1 時呼叫”debounce”副程式
2. 使用計時器完成此延時
3. 在 INT0 中斷服務程式中重新致能中斷



## 練習九問與答（續）

問：三個方法中那個最好？

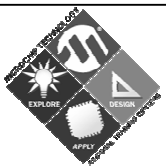
答：依具體情況而定！！

- 每個方法都有各自的優點和缺點



## 附錄 B. 實驗十 存取內建的 **EEPROM**

- 程式放在：..\201\_ASP\Lab10-EEPROM\Lab10-EEPROM.mcp
- 實驗十的內容
  - 直接定義 EEPROM 的資料在程式裡
  - 如何接資料寫入 EEPROM
  - 如何從 EEPROM 裡讀取資料
- 本實驗使用 MPLAB SIM 軟體模擬來完成
- 打開 EEPROM 及 RAM 的是窗觀察結果

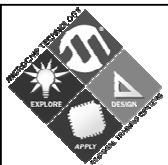


## 附錄 C. 實驗十一 LCD Display Module

- 提供標準的 **HD44780** 的 **LCD** 驅動副程式給 **APP001** 實驗版的 **LCD** 模組使用。
- 使用 **global & extern** 虛指令宣告，可讓其它程式呼叫。



## 201ASP 課程總結



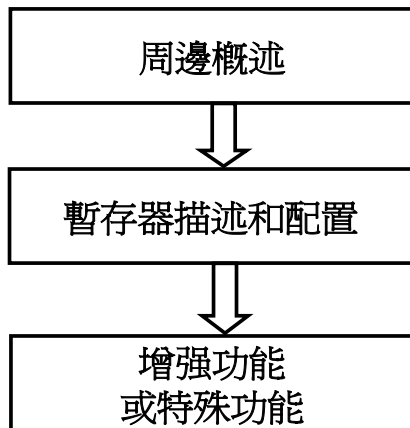
## 201ASP 課程總結

- 我們討論了中階系列的下列周邊
  - I/O 接面
  - 中斷架構和處理
  - 計時器 (Timer0、Timer1 和 Timer2)
  - ECCP 模組 (計時比較、輸入脈波量測器和PWM)
  - 比較器和類比轉換器
    - 參考電壓
  - EUSART – 串列通訊介面
  - 使用 MSSP 模組的 I<sup>2</sup>C 和 SPI



## 結語

- 本次討論遵循標準 **MCHP** 資料手冊流程：



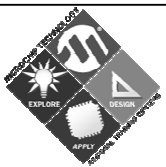
使用這部分內容：

- 開發邏輯流程圖或偽程式  
(避免覆雜而混亂的燒錄!!)

其他提示：

- 詳細程式注解
- 為用戶定義的暫存器選擇  
描述性的名稱

\*封裝和電氣規範在資料手冊的頁尾



## 資源

- 造訪 [www.microchip.com](http://www.microchip.com) 網站並點擊相對應的連結以了解以下訊息：
  - 每周 7 天 24 小時全天候技術支持
  - 應用筆記
  - 網上研討會
  - 程式範例
  - 資料手冊
  - 以及更多詳細訊息！

# HANDS-ON

# Training

謝謝!!

請不要忘記填寫評估表

