

HANDS-ON

Training

102 ASP PIC18F 基礎課程

架構，指令集，組合語言，基本週邊





歡迎參加 102 ASP 教育訓練課程

■ 使用軟體工具

- **MPLAB IDE v8.xx (或更新版本)**
- **MPASM, MPLINK, MPLIB**

■ 使用硬體工具

- **MPLAB ICD2**
- **Microchip APP001 Workshop Board (PIC18F4520 inside)**

■ 參考書籍 (可以從網站上下載)

- **MPASM User's Guide with MPLINK and MPLIB (DS33014J)**
- **MPLAB IDE User's Guide**
- **PIC18F4520 Data Sheet (DS39631A)**



102 ASP 課程內容

■ PIC18F 系列

- **PIC18F MCU** 的架構
- **PIC18F MCU** 指令集
- **PIC18F MCU** 主要周邊功能
- 開發工具介紹與使用
- 撰寫 **PIC18** 系列組合語言
- **MPLAB-IDE** 的基本設定及**ICD2** 的實際除錯

- 練習一：I/O 的操作
- 練習二：使用 Timer1 與 中斷
- 練習三：使用 A/D 轉換器
- 練習四：使用 USART
- 練習五：Table Read



PIC18F452

- 最早的 PIC18Fxxx 元件，目前 Silicon Revision C1
- 32KBytes Flash, 256B EEPROM, 1.5KB RAM
- Timer : Timer0, 1, 2, 3, Watch-Dog Timer
- 3 CCP modules
- I²C, SPI , USART, PSP
- 10-bit ADC with 8 Channel
- PBOR, PLVD, ICD, Self Programming
- 40-pin PDIP(44ML, 44PT) with 34 I/O pin
- 40MHz @ 5V

新建議元件 PIC18F4520



新改良 PIC18Fxxxx (四碼)

- 基本架構仍是採用 PIC18F452 核心
- **Extended Instruction Sets for C18 Compiler Efficiency**
- **Support nano-Watt Technology**
 - 內建 31KHz ~ 8MHz 自由切換的 RC 震盪器
 - 快速 RC 啟動模式
 - 雙速啟動模式
 - 系統時脈監視 / 故障切換功能
 - **Low-Power Watch-Dog Timer**
 - **Ultra Low-Power Wake-Up**
 - 省電模式 : Idle , Sleep



MICROCHIP

PIC18Fxxxx

Enhanced PICmicro MCU
的架構及主要特色



PIC18 系列 MCU 的主要特色

■ PIC18Fxxxx

- 提供較多的指令 (77個基本指令)
- 更大的程式與資料記憶體的定址能力
 - **Program Memory : Up to 2M Bytes**
 - **Data Memory : Up to 4K**
- 程式記憶體使用線性排列方式存取
- 提供 **Access Bank** 設計觀念
- 內建 **PLL 4 倍頻線路**，用 **10Mhz** 就得到 **10 MIPS** 的執行效能 (**100 ns** 的指令執行時間)
- 有 **8 * 8** 的硬體乘法器
- 全系列提供 **10 Bits** 的 **A/D 轉換器** 及 **ICD** 除錯功能



PIC18F4520 增加的功能

■ FLASH Program Memory

- 除了程式記憶體為 FLASH 外，CPU 可自我燒錄程式
- 10 萬次的寫入壽命

■ ICD Capability

- 支援 ICD 的介面
- 高燒錄效率，PIC18F4520 燒錄時間約 3 秒 (32KB 程式)
- 每個 CPU 都可視為一個 ICE Chip

■ EEPROM

- PIC18F4520 都有內建 256 Bytes 的 EEPROM
- 100 萬次的寫入壽命



PIC18F4520 nanoWatt 技術

■ 什麼是 **nano-Watt** 技術

- **Sleep Mode** 下消耗功率低於 **1uW** 以下
- **PIC18F4520 Sleep Mode** 下低於 **0.1uA**

■ 內建的電源管理

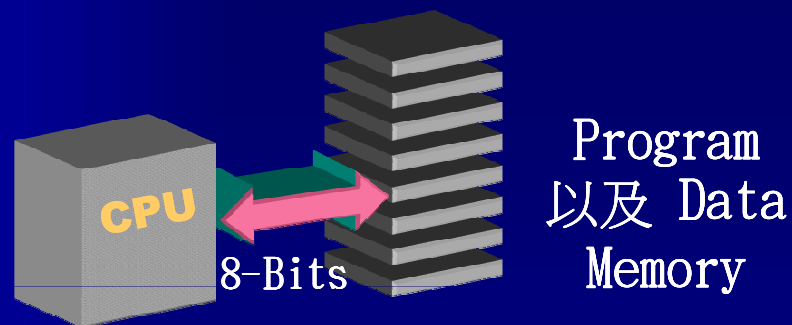
- 內建 **31KHz ~ 8MHz** 自由切換的 **RC** 震盪器
- 快速 **RC** 啟動模式
- 雙速啟動模式
- 系統時脈監視 / 故障切換功能
- **Low-Power Watch-Dog Timer**
- **Ultra Low-Power Timer1**
- 硬體設定或軟體控制的 **Watch-Dog Timer**
- 省電模式 : **Idle (5.8uA), Sleep (0.1uA)**



PICmicro 的架構比較

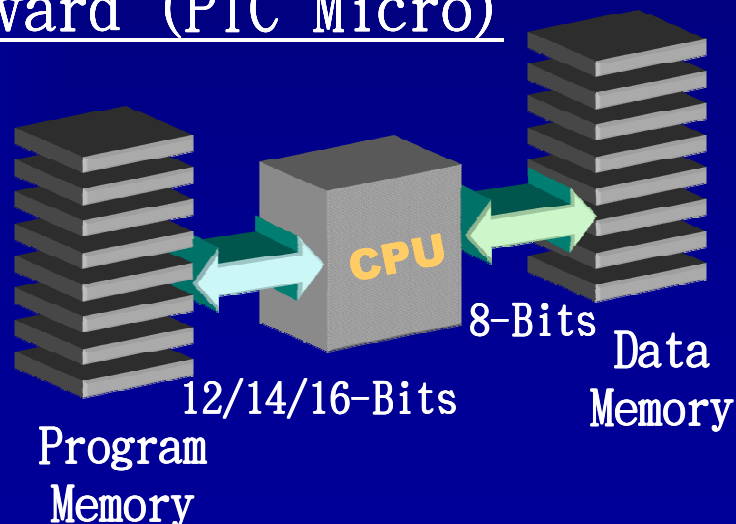
PIC 使用 Harvard Architecture

Von Neumann (一般MCU)



- 經由相同的匯流排來存取指令與資料
- 指令與資料無法有效率的同時被處理
- MCU 的操作效率受到此結構影響而變差

Harvard (PIC Micro)



- 使用兩個匯流排來存取程式碼和資料
- 增加處理資料的效能
- 使得MCU可以具有不同寬度的程式記憶體與資料記憶體 (8 Bit 寬的 Data Memory , 12/14/16 Bit 寬的 Program Memory)



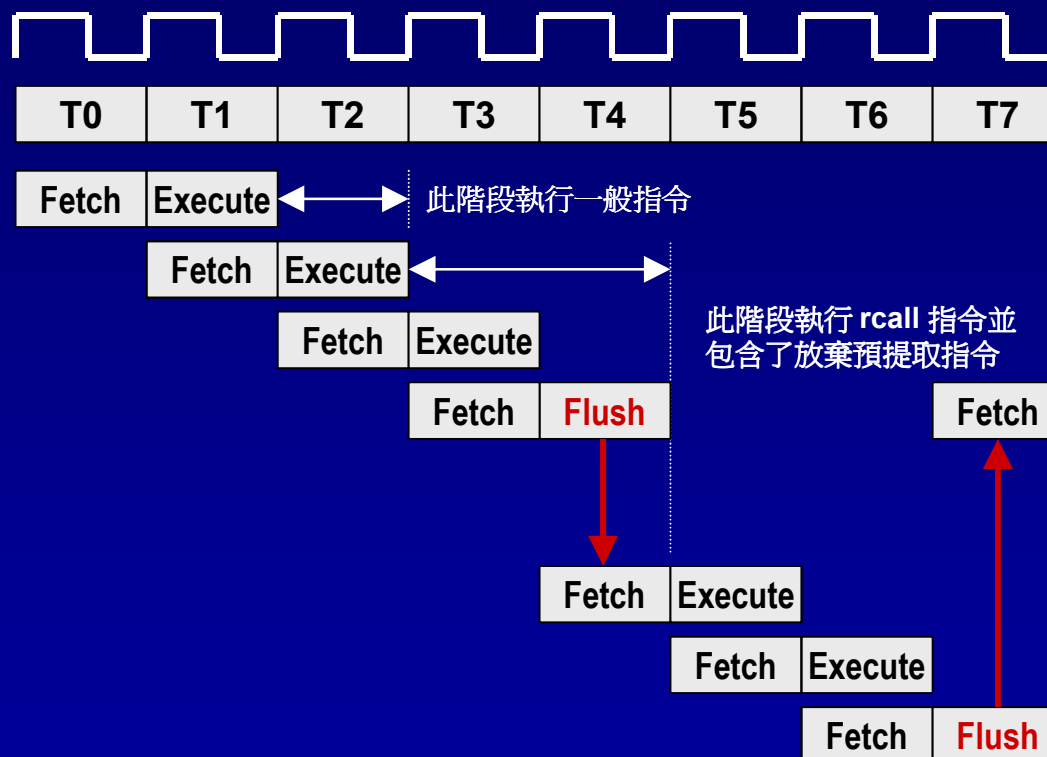
Instruction Pipelining (管線式的指令執行)

- 指令的提取與執行是同時進行的 (因為內部的 **Bus** 是分開的)
- 程式分支的有關指令 (例如: **GOTO, CALL** 或 **Write to PC**) 則需要兩個指令周期

指令週期

程式範例

1	MAIN	movlw	0x05
2		movwf	REG1
3		rcall	SUB1
4		addwf	REG2
⋮			
51	SUB1	movf	PORTB,w
52		return	
53	SUB2	movf	PORTC,w
54		return	





Instruction Pipelining

預提取指令

movlw 0x05

執行指令

—

指令週期

程式範例

1	MAIN	movlw	0x05
2		movwf	REG1
3		rcall	SUB1
4		addwf	REG2
⋮			
51	SUB1	movf	PORTB,w
52		return	
53	SUB2	movf	PORTC,w
54		return	

T0

Fetch



Instruction Pipelining

預提取指令

movwf REG1

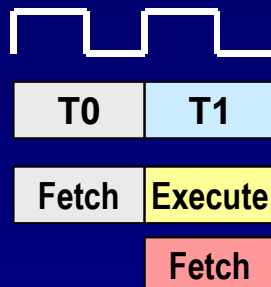
執行指令

movlw 0x05

指令週期

程式範例

1	MAIN	movlw	0x05
2		movwf	REG1
3		rcall	SUB1
4		addwf	REG2
⋮			
51	SUB1	movf	PORTB,w
52		return	
53	SUB2	movf	PORTC,w
54		return	





Instruction Pipelining

預提取指令

rcall SUB1

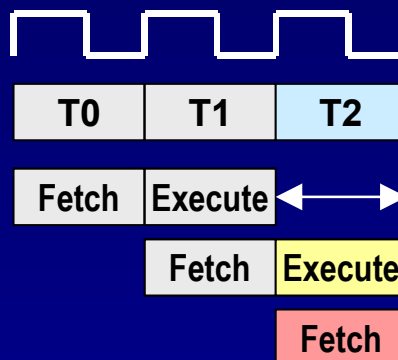
執行指令

movwf REG1

指令週期

程式範例

1	MAIN	movlw	0x05
2		movwf	REG1
3		rcall	SUB1
4		addwf	REG2
⋮			
51	SUB1	movf	PORTB,w
52		return	
53	SUB2	movf	PORTC,w
54		return	



此階段執行一般指令



Instruction Pipelining

預提取指令

addwf REG2

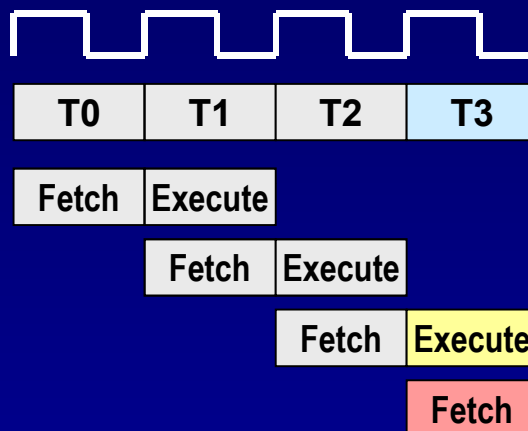
執行指令

rcall SUB1

指令週期

程式範例

1	MAIN	movlw	0x05
2		movwf	REG1
3		rcall	SUB1
4		addwf	REG2
⋮			
51	SUB1	movf	PORTB,w
52		return	
53	SUB2	movf	PORTC,w
54		return	





Instruction Pipelining

預提取指令

movf PORTB,w

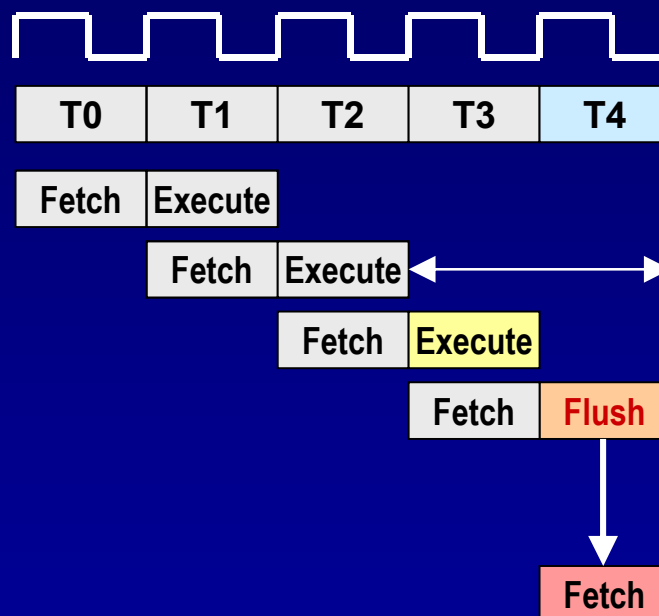
執行指令

rcall SUB1

指令週期

程式範例

1	MAIN	movlw	0x05
2		movwf	REG1
3		rcall	SUB1
4		addwf	REG2
⋮			
51	SUB1	movf	PORTB,w
52		return	
53	SUB2	movf	PORTC,w
54		return	



放棄 **rcall** 指令，重新提取下一個指令



Instruction Pipelining

預提取指令

return

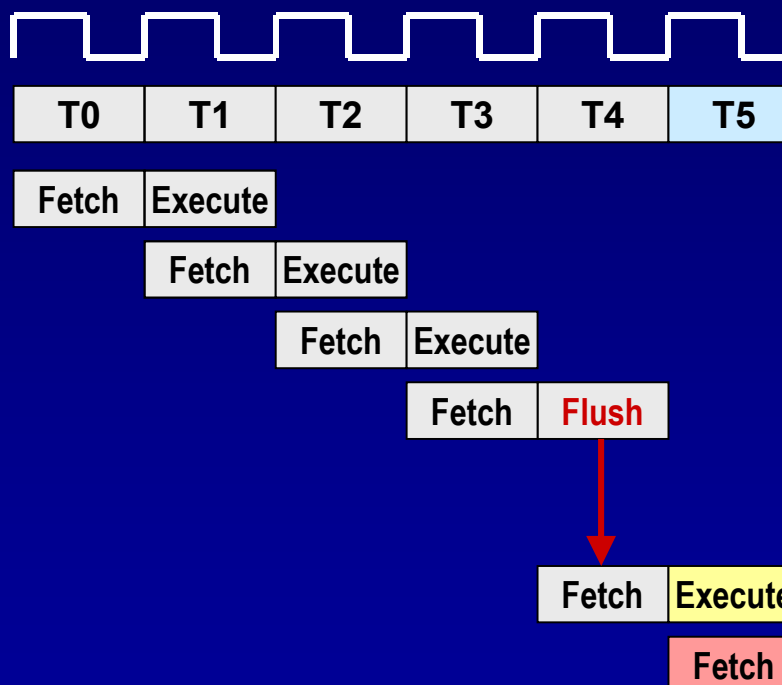
執行指令

movf PORTB,w

指令週期

程式範例

1	MAIN	movlw	0x05
2		movwf	REG1
3		rcall	SUB1
4		addwf	REG2
⋮			
51	SUB1	movf	PORTB,w
52		return	
53	SUB2	movf	PORTC,w
54		return	





Instruction Pipelining

預提取指令

movf PORTC, w

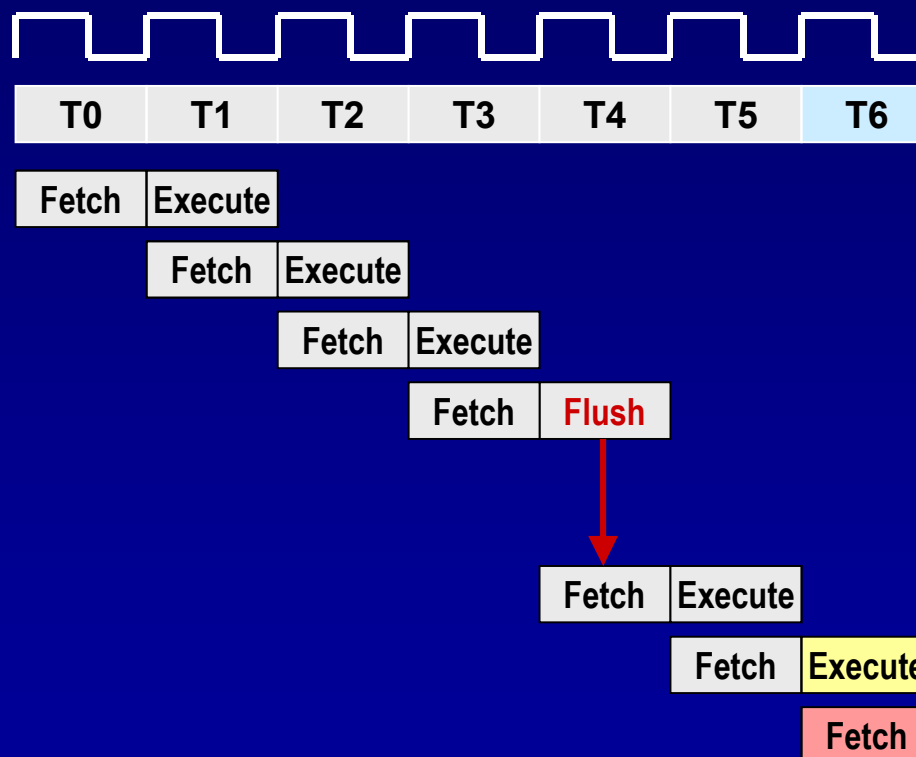
執行指令

return

指令週期

程式範例

1	MAIN	movlw	0x05
2		movwf	REG1
3		rcall	SUB1
4		addwf	REG2
⋮			
51	SUB1	movf	PORTB, w
52		return	
53	SUB2	movf	PORTC, w
54		return	





Instruction Pipelining

預提取指令

addwf REG2

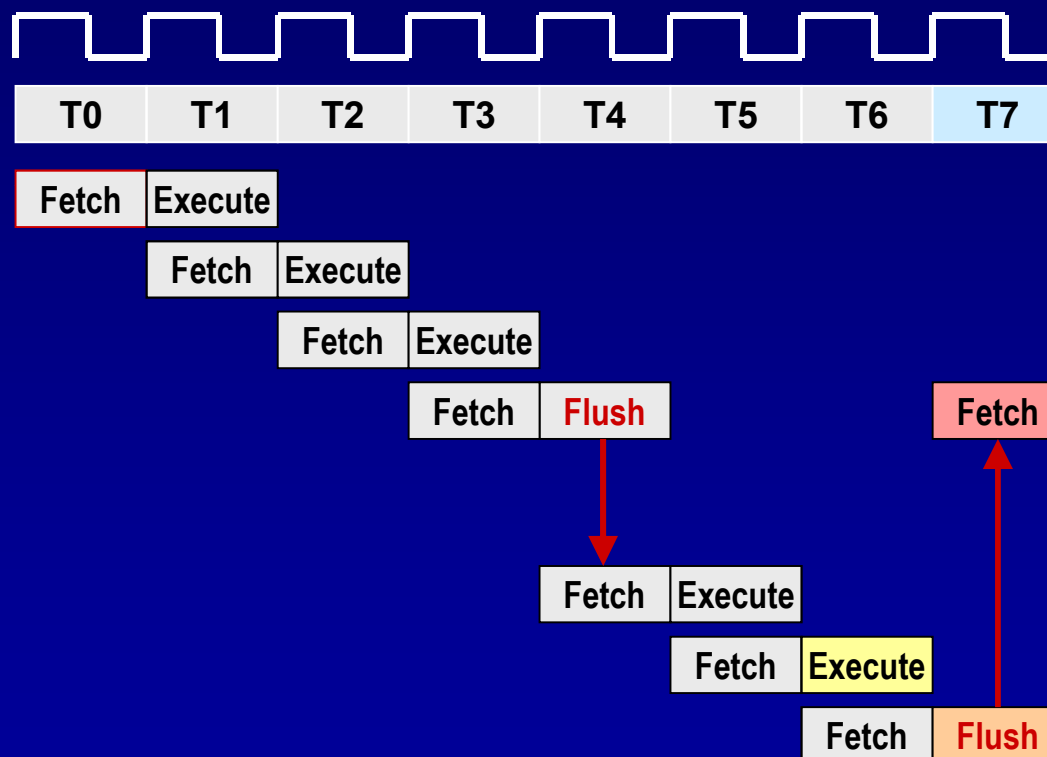
執行指令

return

指令週期

程式範例

1	MAIN	movlw	0x05
2		movwf	REG1
3		rcall	SUB1
4		addwf	REG2
⋮			
51	SUB1	movf	PORTB,w
52		return	
53	SUB2	movf	PORTC,w
54		return	





PICmicro[®] MCU 的架構

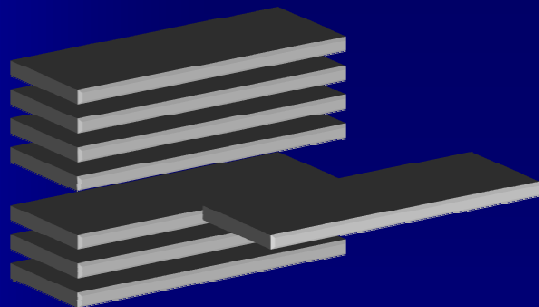
長指令寬度的編碼方式 (Long Word Instruction)

- ◆ PIC 指令寬度有分成三種 12, 14 or 16-bits，這種指令寬度稱為長指令 (Long Word Instruction)。
- ◆ PIC 的資料匯流排(Data Bus)寬度是 8 bits，所以 PIC 是一個 8-bit MCU。
- ◆ 若於 PIC18Fxxxx 具備 2K x 16 words 的程式記憶體，可完成的工作約相當於其他有4K 記憶體的 MCU (80C51).
- ◆ 單一周期的指令運作使 MCU 的處理能力提高許多



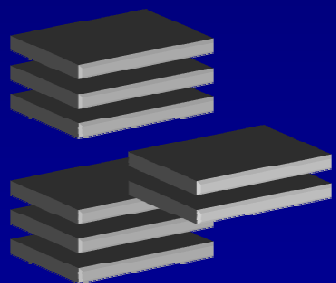
PICmicro[®] MCU 的架構

長指令寬度的編碼方式
(Long Word Instruction)



PIC MCU

movlw #imm<8>



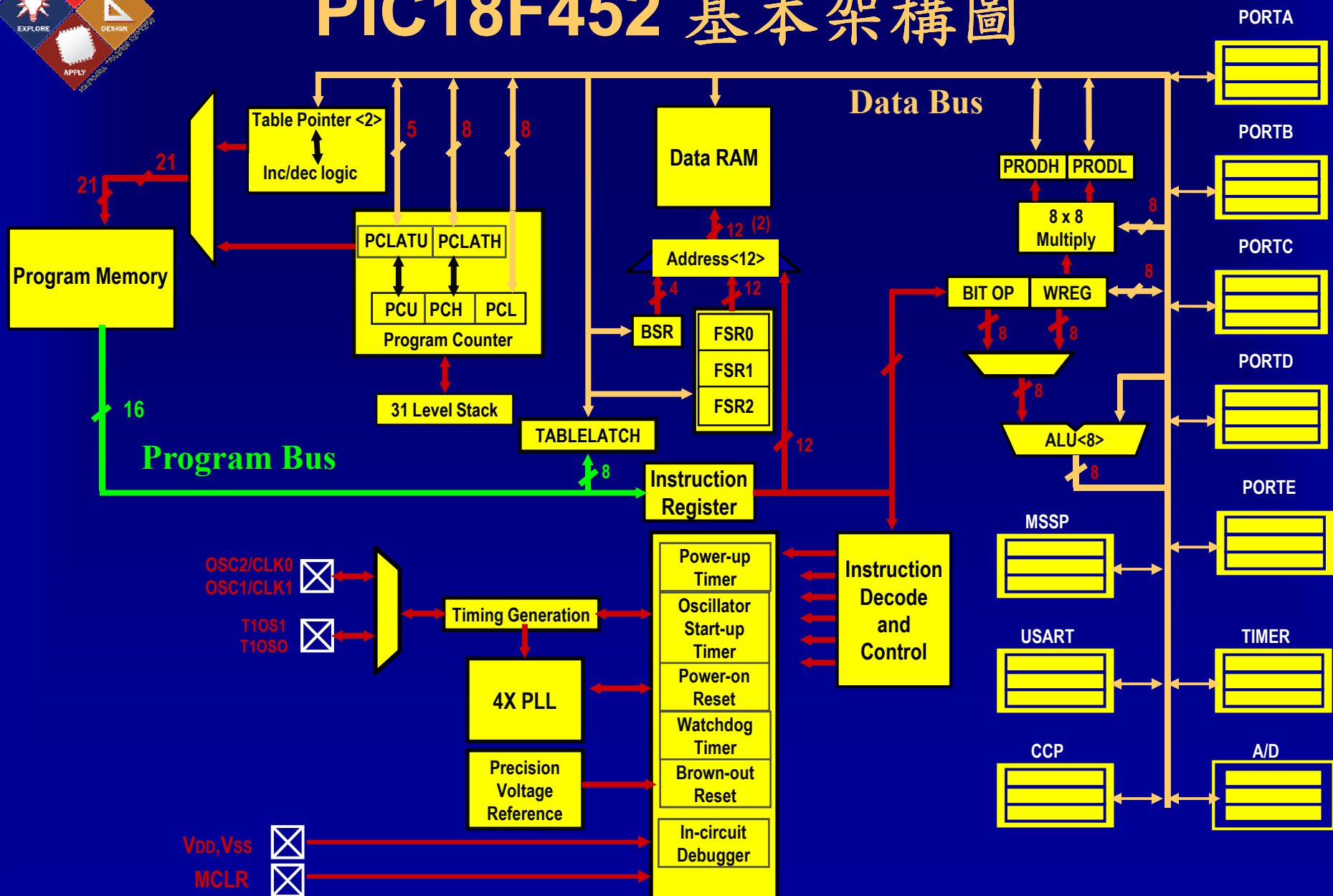
MC68HC05

ldaa #imm<8>



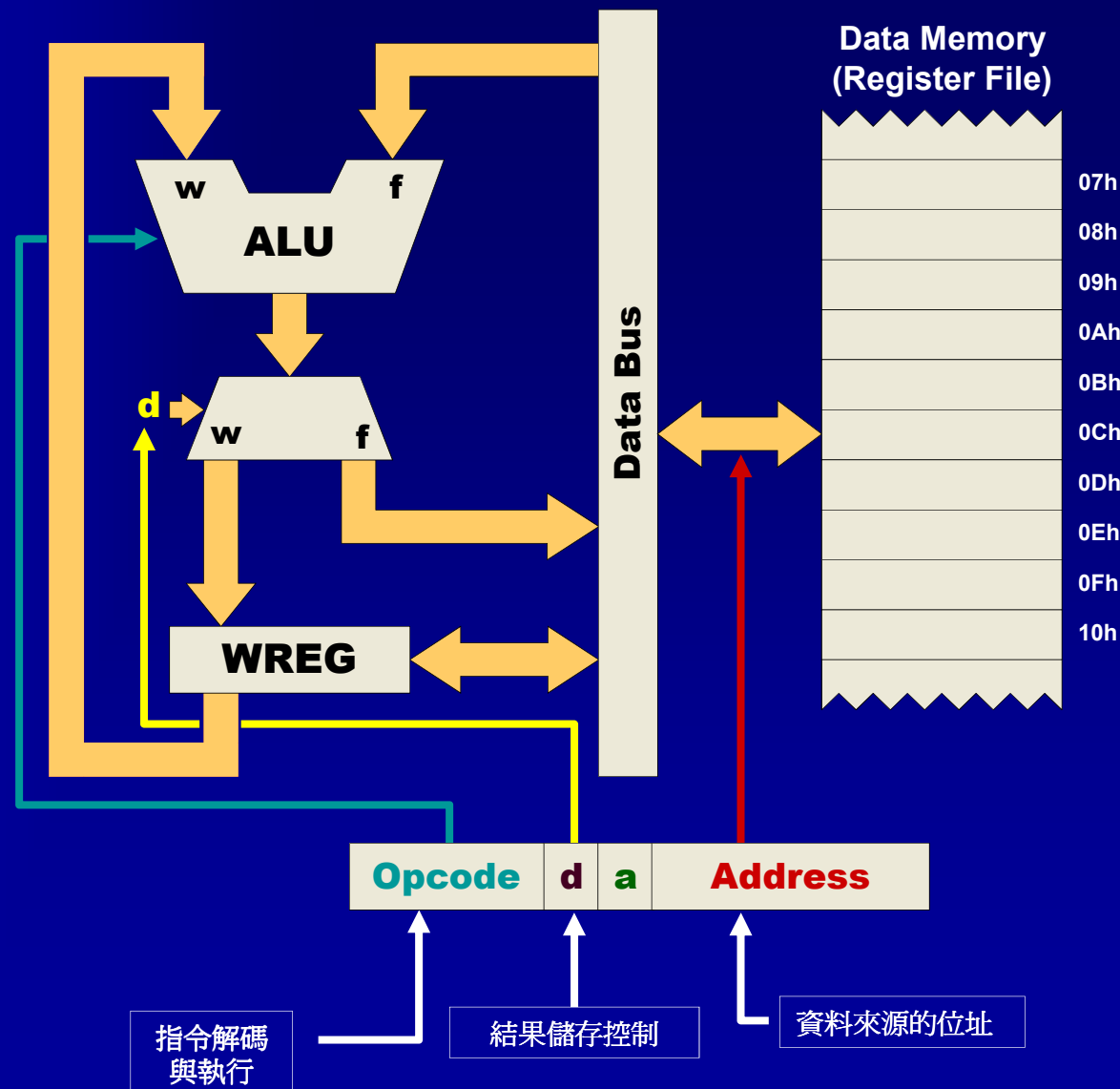


PIC18F452 基本架構圖





引用 Register File 的觀念

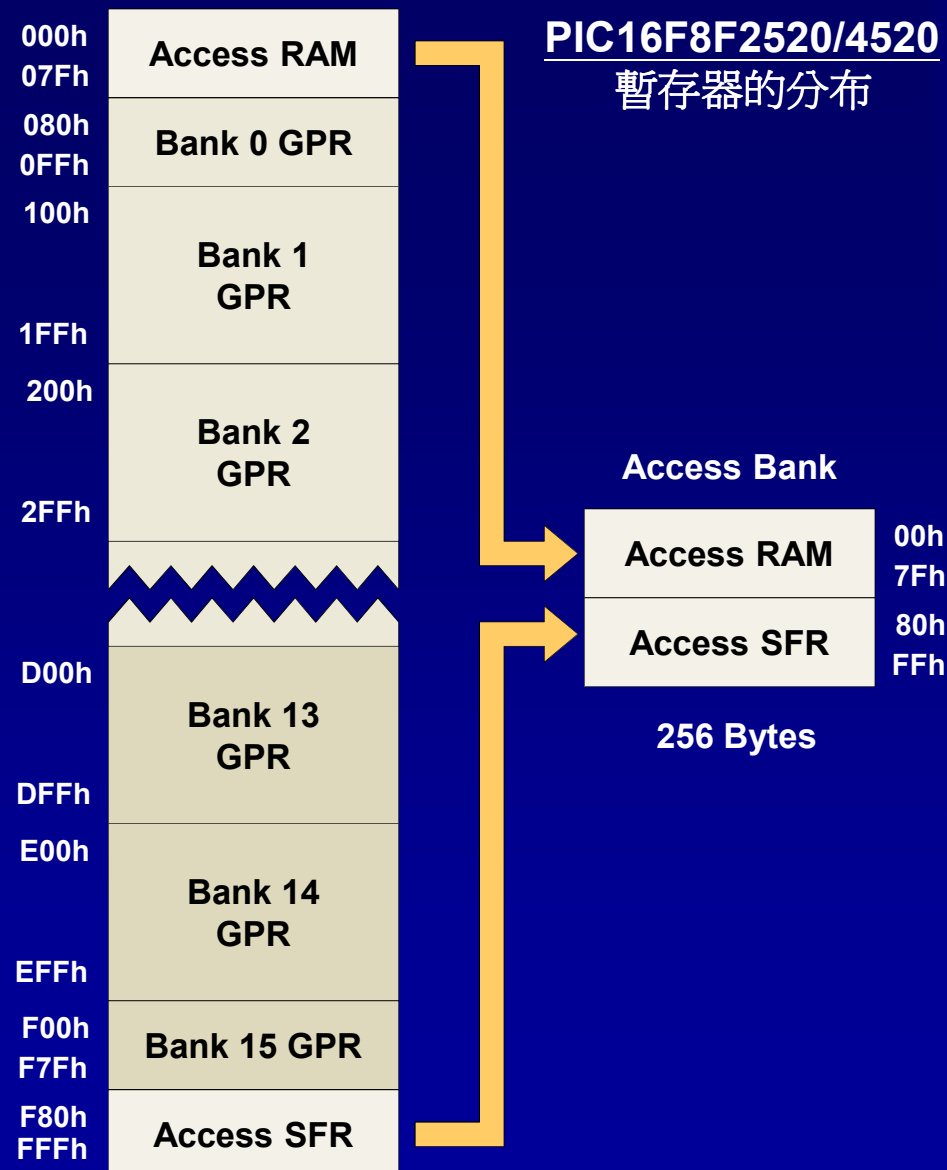


- **SFR's** 位於 **Bank 15** 所有指令皆可對任一暫存器操作，無須切換 **Bank**
- **Long word** 的指令讓各暫存器可被直接定址
 - **MOVFF src , dest**
- 周邊暫存器及狀態暫存器皆為 **Register files** 的一部份



Data Memory 架構

- 有16 banks 的 Data Memory，每個 Bank 由 256 Bytes 組成 (Max. 4K Bytes)
- Special Function Registers (SFRs) 被安排在 Bank 15
- 使用 BSR<3:0> 來選擇適當的 BANK 或使用 BANKSEL 來做切換 BANK 的工作
- MOVFF 指令可免去切換 BANK 的工作，但需 2 words 的指令空間來完成
- 使用 Access Bank 的暫存器，無須做 Bank 的切換

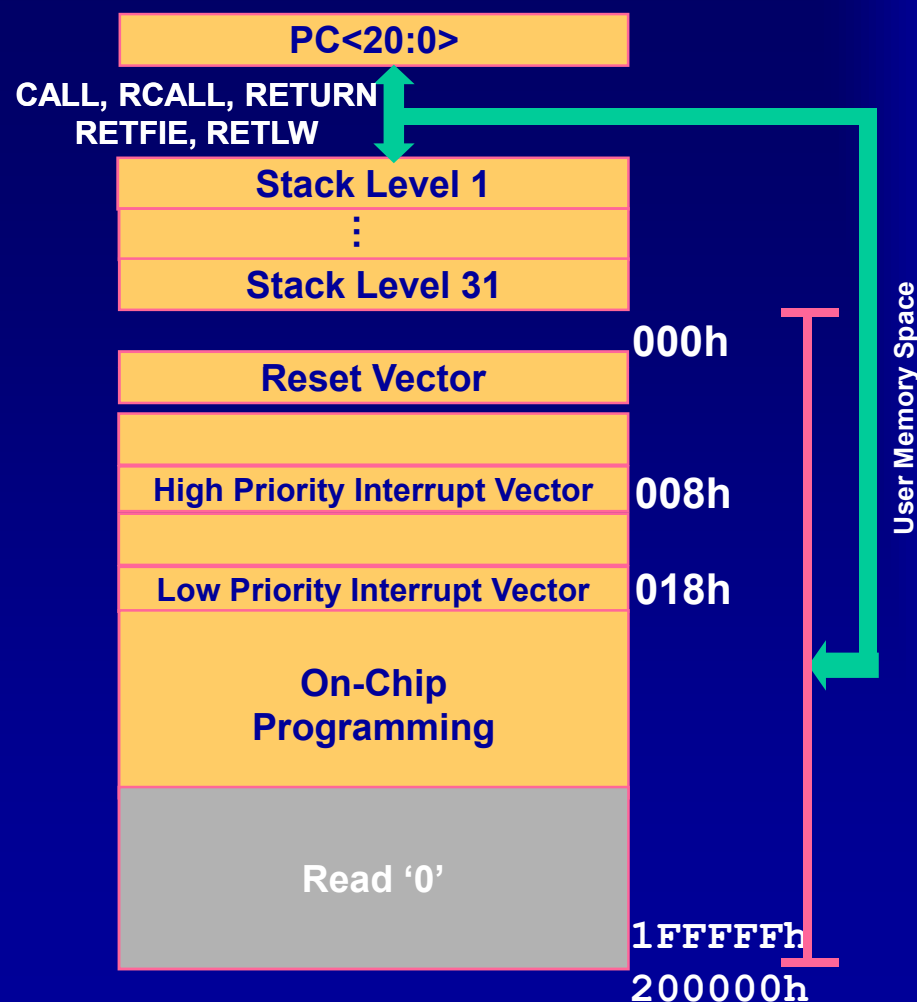




PIC18 系列的架構

Program Memory

- 最大可達 **2M Bytes (21 bits)** 的程式空間
- **Reset Vector** 位於程式位址 **0x0000**
- 高優先權中斷向量的位址位於 **0008h**
- 低優先權中斷向量的位址位於 **0018h**
- **31 層** 獨立的位址堆疊區





PIC18 系列的架構

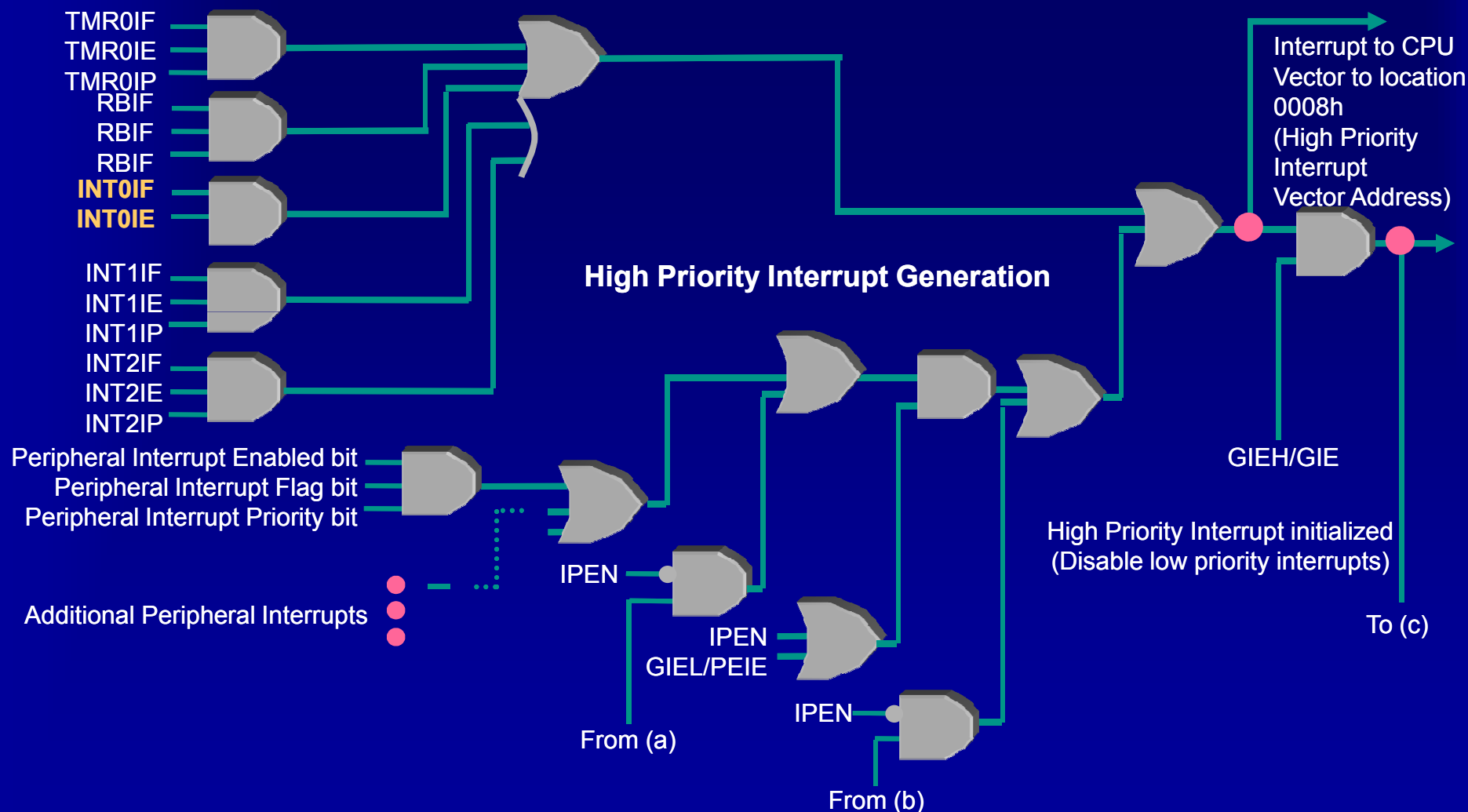
Interrupt Overview

- 豐富的內部與外部中斷來源
 - 幾乎所有周邊皆有中斷 CPU 的能力
 - 有些 I/O 的變化也可構成中斷條件 (PORTB)
- 每個周邊中斷可以被設定成具有 高/低優先權 (Hi/Low Priority)
 - INT0 永遠為高優先權外部中斷
- 具相同優先權設定的中斷則可由軟體來決定其優先順序
- 有整體及各別的中斷致能控制位元
- 每個中斷都有其獨立的中斷旗標，可用軟體詢問的方式
- 大部份的中斷可以用來喚醒處於 SLEEP 狀態的 PICmicro
- 中斷延遲固定為 3 個 cycle
 - 容易以軟體來判斷中斷的經過時間



PIC18 系列的架構

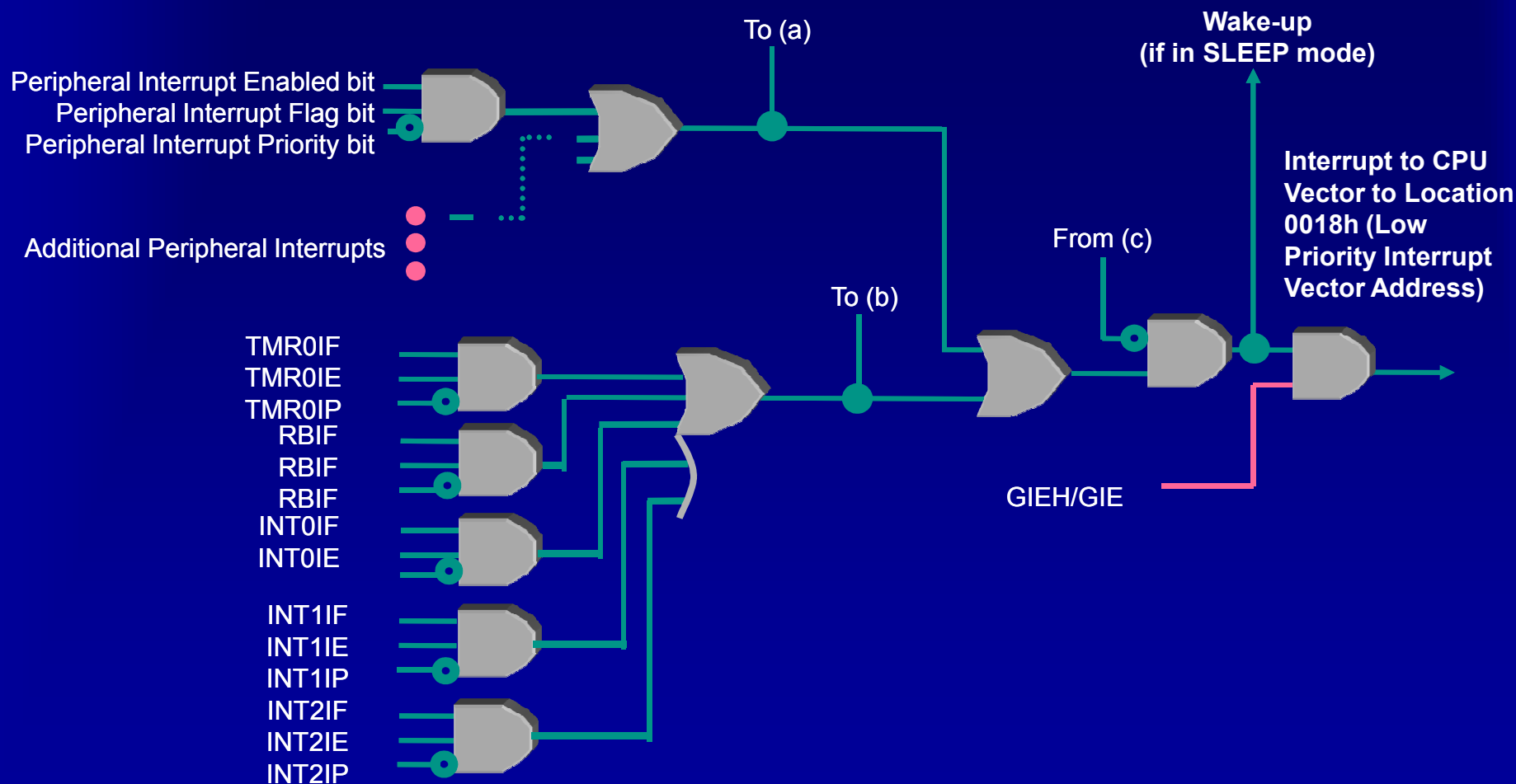
高優先全中斷邏輯架構





PIC18 系列的架構

低優先全中斷邏輯架構

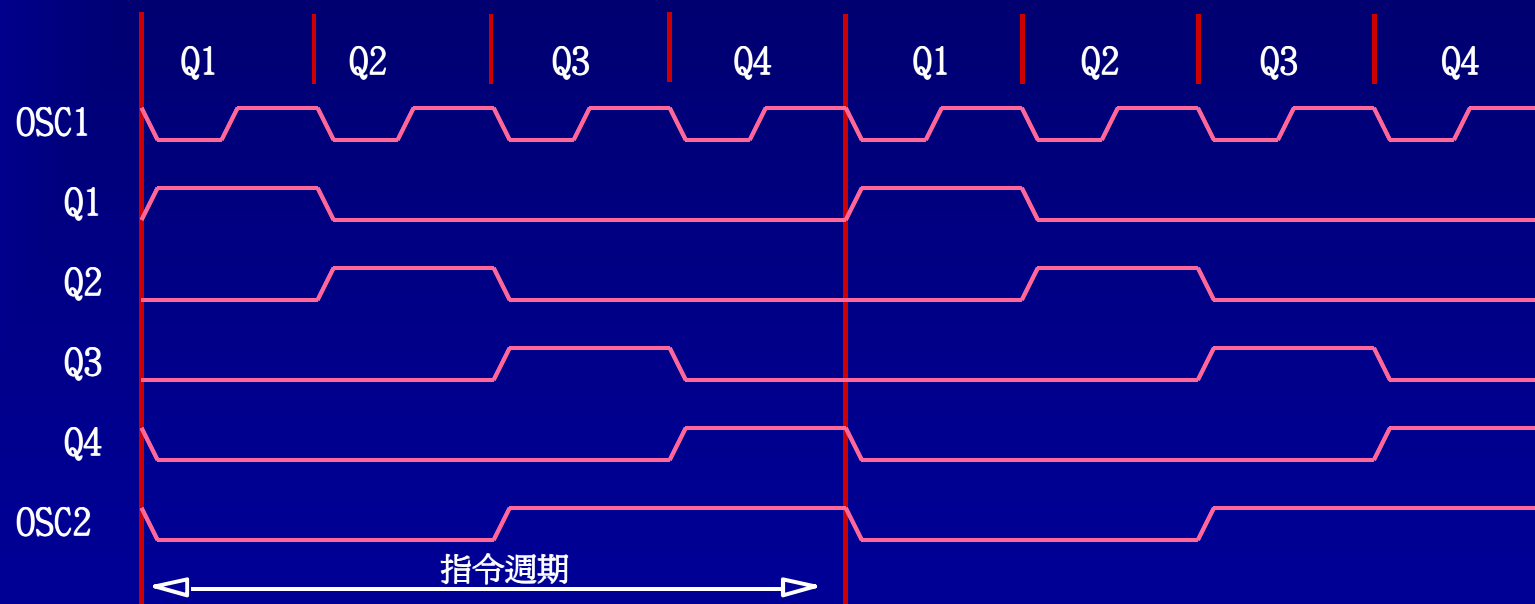




PIC18 系列的架構

Clocking Scheme

- 一個指令周期等於輸入 CPU 頻率的 1/4
- 若 CPU 操作於 40 Mhz 時，其指令周期為 100 ns
- F_{osc} 振盪頻率 / 4 = T_{osc} 指令週期



一般性的動作：Q1 = 解碼 Q2 = 讀取 Q3 = 執行 Q4 = 寫入



MICROCHIP

PIC18Fxxxx

定址模式



18Fxxx 定址模式

■ 四種基本的定址模式

- 立即定址模式
- 直接定址模式
- 間接定址模式
- 相對定址模式



PIC18 定址模式 立即數的操作

- **8-bit** 的常數值 (**literal value**) 被包含於程式碼之中
- 使用於 **literal instructions** , 例如
 - `movlw, addlw, retlw, etc.`

16-bit Instruction for Literal Instructions

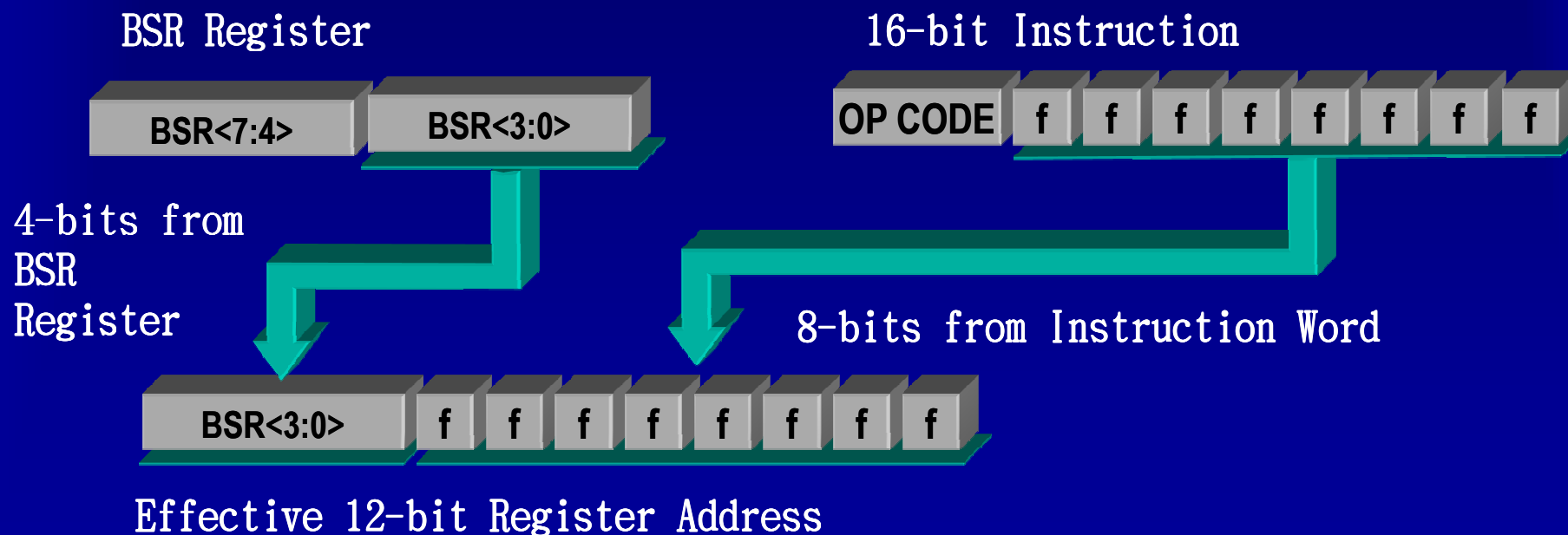




PIC18 定址模式

Data Memory: 直接定址 (Banked)

- Data Memory 總共有效位址為 12 Bits (4K Bytes)
- 8-bit 的直接位址由指令中得到
- 其他高位的 4-bits 由 BSR<0..3> 獲得





直接定址 – 範例



```
MyReg equ 0x120
```

⋮

```
movlw b'11110000'
```

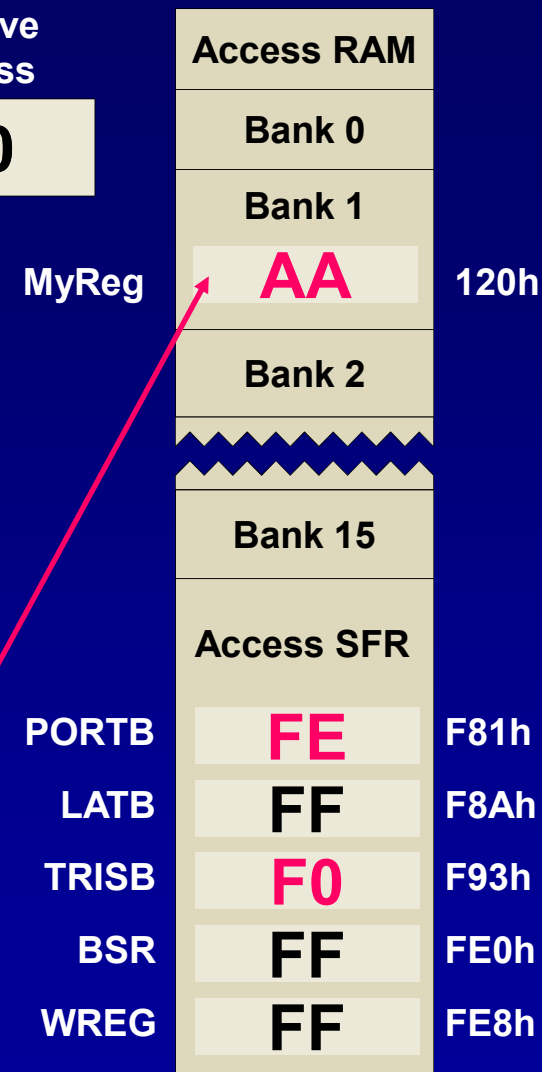
```
movwf TRISB, a
```

```
bsf PORTB, 0, a
```

```
movlb 0x01 ; 選擇 BANK 1
```

```
movlw 0xAA
```

```
movwf MyReg ; @120h
```



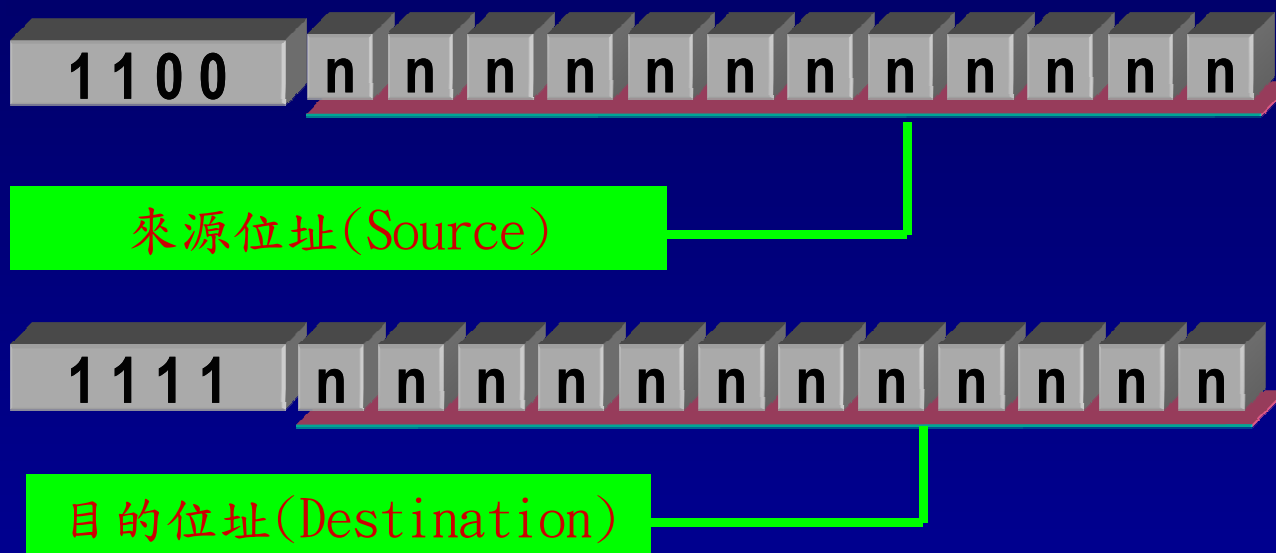


PIC18 定址模式

Data Memory：直接定址 (Non Banked)

■ MOVFF src, dest (RAM 對 RAM 的資料搬移)

- 2 Word 的指令, 直接指定兩個 4K 的位址



善用 Access BANK 或選擇 BANKED 的 Single Word 指令
可增加執行效能及佔用較少的程式記憶體



PIC18 定址模式

Data Memory: 間接定址 (Indirect addressing)

- 三個 12-bit 寬的間接定址暫存器
 - FSR0, FSR1, FSR2
- FSRn (File Select Register) 的內容可做以下調整:
 - Pre-Incremented : (PREINCx) 先被加一再存取其內容
 - Post-Incremented : (POSTINCx) 存取完 FSRn 所指內容後將 FSRn 加 1
 - Post-Decrement : (POSTDECx) 存取完 FSRn 所指內容後將 FSRn 減 1
 - Offset by WREG value (signed) : (PLUAWx) FSRn 不變，以 W 為偏移值相加後以產生新的索引指標
 - 保持不變 : (INDFx)

12-bit FSRn Register





PIC18 定址模式

Data Memory: 間接定址 (Indirect Addressing)

- 範例：清除由位址 **0x000 to 0x5FF** 的資料內容
 - 間接位址被寫入 **FSRn** 中
 - 每當 **INDFn** 被當成運算元(operand)，實際上被操作的位址乃是由 **FSRn** 內容所指的位址
 - 類似 8051 的 **@R0**、**@R1** 定址法，但功能更齊全

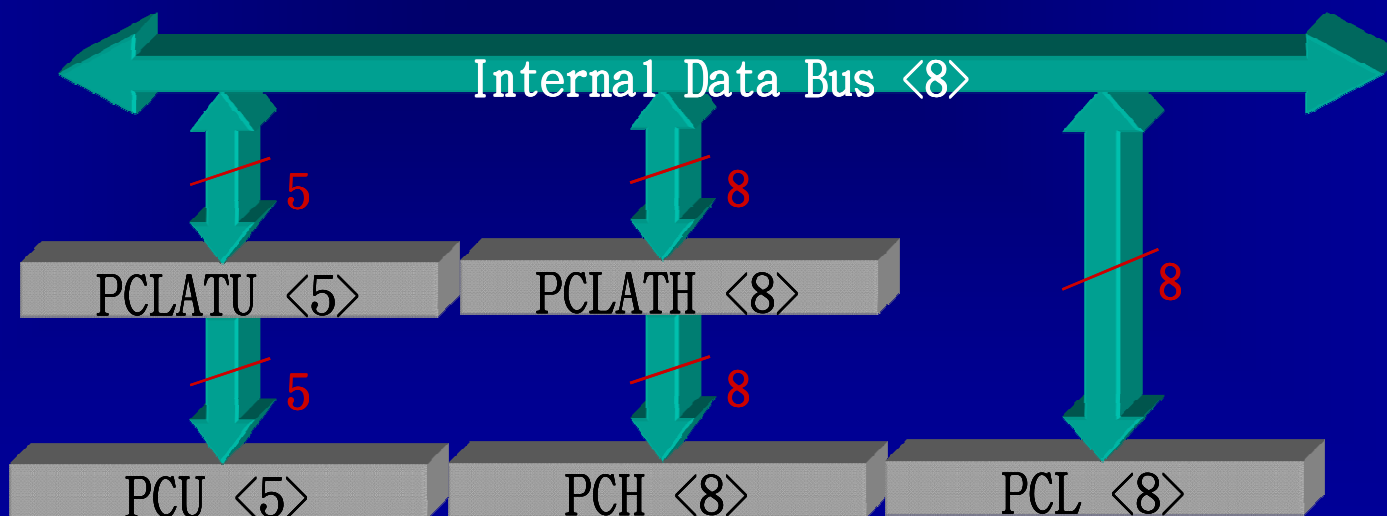
```
_Loop      movlw    0x05           ; Value to compare
           lfsr     1,0x000       ; Clear address from 0
           clrf     POSTINC1      ; Clear, increment FSR1
           cpfsgt   FSR1H         ; FSR1H > 5 ? ( 0x600 )
           goto     _Loop        ; NO, loop
           :                   ; YES, next instruction
```



PIC18 定址模式

相對定址使用方法

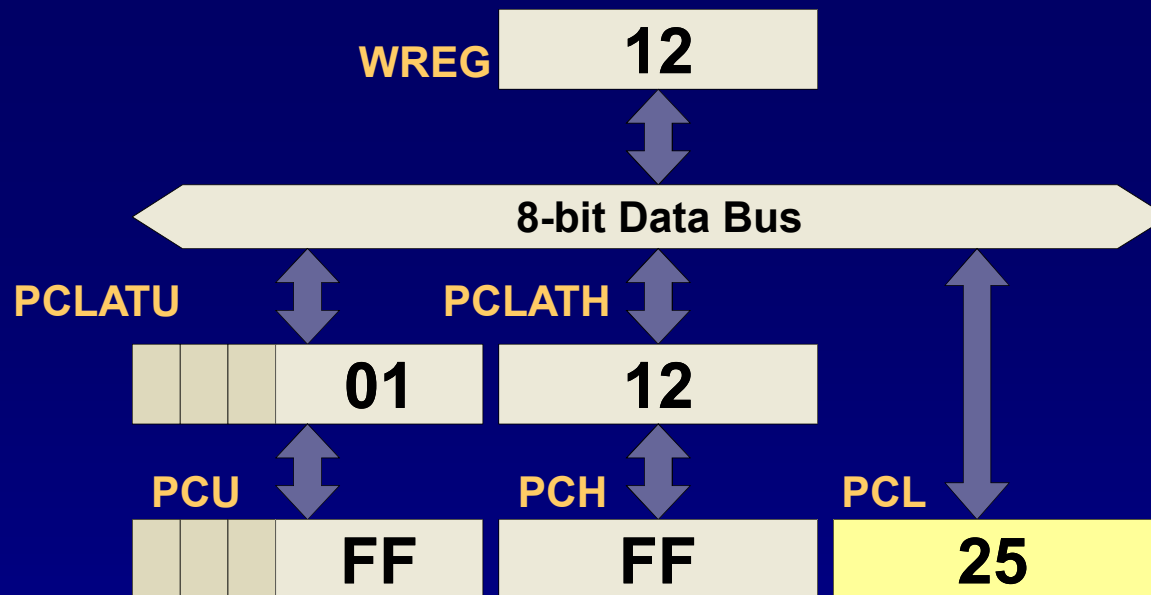
- PIC18 系列的程式計數器為 21 Bits，無法以 8 Bit 的 Data Bus 直接將其修改，故以栓鎖(Latch Buffer)的架構來輔助以達成修改的工作
- 先將欲執行位址的高位元部份寫入 PCLATU 和 PCLATH
- 最後將低位元部份寫入 PCL，此時存於此三個暫存器的 21-bit 內容將被一次寫入 Program Counter





相對定址說明

利用此方式就可以
達到 **GOTO** 的位址
計算，直接修改
program counter



To write to PC:

- ① Write upper byte to PCLATU
- ② Write high byte to PCLATH
- ③ Write low byte to PCL

PC_OFFSET =

25

```
movlw UPPER 0x011250
movwf PCLATU
movlw HIGH 0x011250
movwf PCLATH
movff PC_OFFSET, PCL
```



PC 相對定址：計算 GOTO 位址

- 計算 **GOTO** 位址，使用索引查表方式建立跳躍表格

JTS

```
org      0x002180
movlw    UPPER_JUMP_TABLE
movwf    PCLATU
movlw    HIGH_JUMP_TABLE
movwf    PCLATH
movlw    0x04
mulwf    USER_INPUT
movf     PRODL W
addwf    PCL, f
goto     DoUserInput0
goto     DoUserInput1
goto     DoUserInput2
goto     DoUserInput3
goto     DoUserInput4
```

@ 0x002190

JUMP_TABLE

USER_INPUT

02

WREG

08

PCLATU PCLATH

00

21

FF

FF

FF

PCU

PCH

PCL

計算跳躍的偏移量位址
每一個 **goto** 指令佔 4 bytes:
所以要乘四後才可以得到正確的偏移位址



PIC18 Return Address Stack

Top of Stack Registers

TOSU TOSH TOSL

0 0 1 2 6 0

23

0

STKPTR Register

STKOVF STKUNF SP4:SP0

0 0 - 0 0 1 0 0

7

6

5

4

0

Program Counter

PCU:PCH:PCL

0 0 1 2 6 0

Events that push PC+2 onto the stack:

PUSH

CALL

RCALL

Interrupt

Events that pop the top of the stack:

POP

RETURN

RETLW

RETFIE

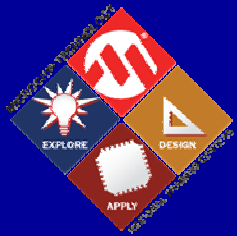
- ◆ PCLATU 及 PCLATH 不受任何的 Return 指令影響
- ◆ 每層的堆疊寬度為 21-bit (整個 4M 位址)
- ◆ 只有 STKPTR 所指到的最上層的堆疊資料 (TOSU, TOSH, TOSL) 允許被程式讀取或修改

Stack

Stack Level

0	31
0	30
0	09
0	08
0	07
0	06
0	05
001260	04
XXXXXX	03
XXXXXX	02
XXXXXX	01

堆疊方向



MICROCHIP

PIC18FXXX

指令集



PIC18系列 MCU 指令集

Instruction Set Overview

◆ 77 個指令

- 73 個是 Single Word 指令
- 4 個 two-word 指令：MOVFF，LFSR，CALL，GOTO

◆ 55% 的指令只需 **single cycle** 即可完

- 4 個 two word 的指令須耗費 2 周期
- 10 個 branches，calls 會依情況耗費一或兩個周期
- 8 個對 Table 操做的指令為 2 周期
- 3 個 return 的指令需耗費 2 周期
- 10 個條件式的 SKIP 指令依狀況需耗費 1 or 2 周期



PIC18系列 MCU 指令集

Instruction Set Overview

- ◆ 具有記憶體直接搬移的指令
 - Data memory 對 Data memory 的直接操作
 - Data memory 對 Program memory 的讀取或寫入
- ◆ Single cycle 的 8 x 8 硬體乘法器 (100 ns)
- ◆ 完整的位元操作指令
 - 單一執行周期的 bit set, clear 或 toggle 操作
 - 直接可對所有暫存器做位元操作 (包括 I/O Port)



PIC18系列 MCU 指令集

Summary (16-bit core)

Byte-Oriented Operations

ADDWF	f, d, a	Add WREG and f
ADDWFC	f, d, a	Add WREG and f
ANDWF	f, d, a	AND WREG and f
CLRF	f, a	Clear f
COMF	f, d, a	Complement f
CPFSEQ	f, a	Compare f with WREG, skip =
CPFSGT	f, a	Compare f with WREG, skip >
CPFSLT	f, a	Compare f with WREG, skip <
DECF	f, d, a	Decrement f
DECFSZ	f, d, a	Decrement f, skip if zero
DCFSNZ	f, d, a	Decrement f, skip if not zero
INCF	f, d, a	Increment f
INCFSZ	f, d, a	Increment f, skip if zero
INFSNZ	f, d, a	Increment f, skip if not zero
IORWF	f, d, a	Inclusive OR WREG and f
MOVF	f, d, a	Move f
MOVFF	fs, fd	Move fs (source) to fd (destination)
MOVWF	f, a	Move WREG to f

Byte -Oriented Operations

MULWF	f, a	Multiple WREG with f
NEGF	f, a	Negate f
RLCF	f, d, a	Rotate left f through carry
RLNCF	f, d, a	Rotate left f, no carry
RRCF	f, d, a	Rotate right f through carry
RRNCF	f, d, a	Rotate right f, no carry
SETF	f, a	Set f
SUBFWB	f, d, a	Subtract f from WREG with borrow
SUBWF	f, d, a	Subtract WREG from f
SUBWFB	f, d, a	Subtract WREG from f with borrow
SWAPF	f, d, a	Swap nibbles of f
TSTFSZ	f, a	Test f, skip if zero
XORWF	f, d, a	Exclusive OR WREG and f

f = File Register, d = destination (1=f, 0=WREG), a = access bank (0=access bank, 1=BSR)



PIC18系列 MCU 指令集

Summary (16-bit core)

Bit -Oriented Operations

BCF	f, b, a	Bit clear f
BSF	f, b, a	Bit set f
BTG	f, b, a	Bit toggle f
BTFSC	f, b, a	Bit test f, skip if clear
BTFSS	f, b, a	Bit test f, skip if set

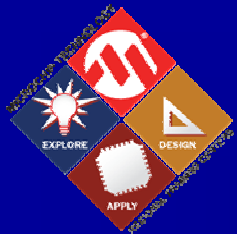
Data Memory <--> Program Memory Operations

TBLRD*	-	Table Read
TBLRD*+	-	Table Read with post-inc.
TBLRD*-	-	Table Read with post-dec.
TBLRD+*	-	Table Read with pre-inc.
TBLWT*	-	Table Write
TBLWT*+	-	Table Write with post-inc.
TBLWT*-	-	Table Write with post-dec.
TBLWT+*	-	Table Write with pre-inc.

Literal Operations

ADDLW	k	Add literal with WREG
ANDLW	k	AND literal with WREG
IORLW	k	Inclusive OR literal with WREG
LFSR	f, k	Move literal to FSRn
MOVLB	k	Move literal to BSR
MOVLW	k	Move literal to WREG
MULLW	k	Multiply literal to WREG
RETLW	k	Return, place literal in WREG
SUBLW	k	Subtract WREG from literal
XORLW	k	Exclusive OR literal with WREG

f = File Register, k = literal value (8-bit), b = bit address (0 - 7)



PIC18系列 MCU 指令集

Summary (16-bit core)

Control Operations

BC	n	Branch if Carry (n is 8-bit)
BN	n	Branch if Negative (n is 8-bit)
BNC	n	Branch if No Carry (n is 8-bit)
BNN	n	Branch if Not Negative (n is 8-bit)
BNOV	n	Branch if No Overflow (n is 8-bit)
BNZ	n	Branch if Not Zero (n is 8-bit)
BOV	n	Branch if Overflow (n is 8-bit)
BRA	n	Branch (n is 11-bit)
BZ	n	Branch if Zero (n is 8-bit)
CALL	n, s	Call subroutine (n is 20-bit)
CLRWDT	-	Clear Watchdog Timer

Control Operations

DAW	-	Decimal Adjust WREG
GOTO	n	Go to address (n is 20-bit)
NOP	-	No Operation
POP	-	Pop top of return stack (TOS)
PUSH	-	Push top of return stack (TOS)
RCALL	n	Relative call (n is 11-bit)
RESET	-	Software device reset
RETFIE	s	Return from interrupt enable
RETLW	k	Return, place literal in WREG
RETURN	s	Return from subroutine
SLEEP	-	Go into standby mode

n = absolute address or address offset, *k* = literal value (8-bit), *s* = fast restore



PIC18系列 MCU 指令集

以位元組為操作的對象

Byte-Oriented Operations

ADDWF	f, d, a
ADDWFC	f, d, a
ANDWF	f, d, a
CLRF	f, a
COMF	f, d, a
CPFSEQ	f, a
CPFSGT	f, a
CPFSLT	f, a
DECF	f, d, a
DECFSZ	f, d, a
DCFSNZ	f, d, a
INCF	f, d, a
INCFSZ	f, d, a
INFSNZ	f, d, a
IORWF	f, d, a
MOVF	f, d, a
MOVFF	fs, fd
MOVWF	f, a

16-bit Instruction for Byte Oriented Operations



d = Destination Bit

d = 0 運算結果置於 W

d = 1 運算結果置於 F
(Default d=1)

a = Access Bit

a = 0 指定置於 Access Bank 的暫存器

(Default a=0, 或不寫)

a = 1 實際的位址由 BSR 與 8-bit 的暫存器位址組成

f = 8-bit 暫存器位址

Example:

ADDWF 0, 0

ADDWF f, d, a



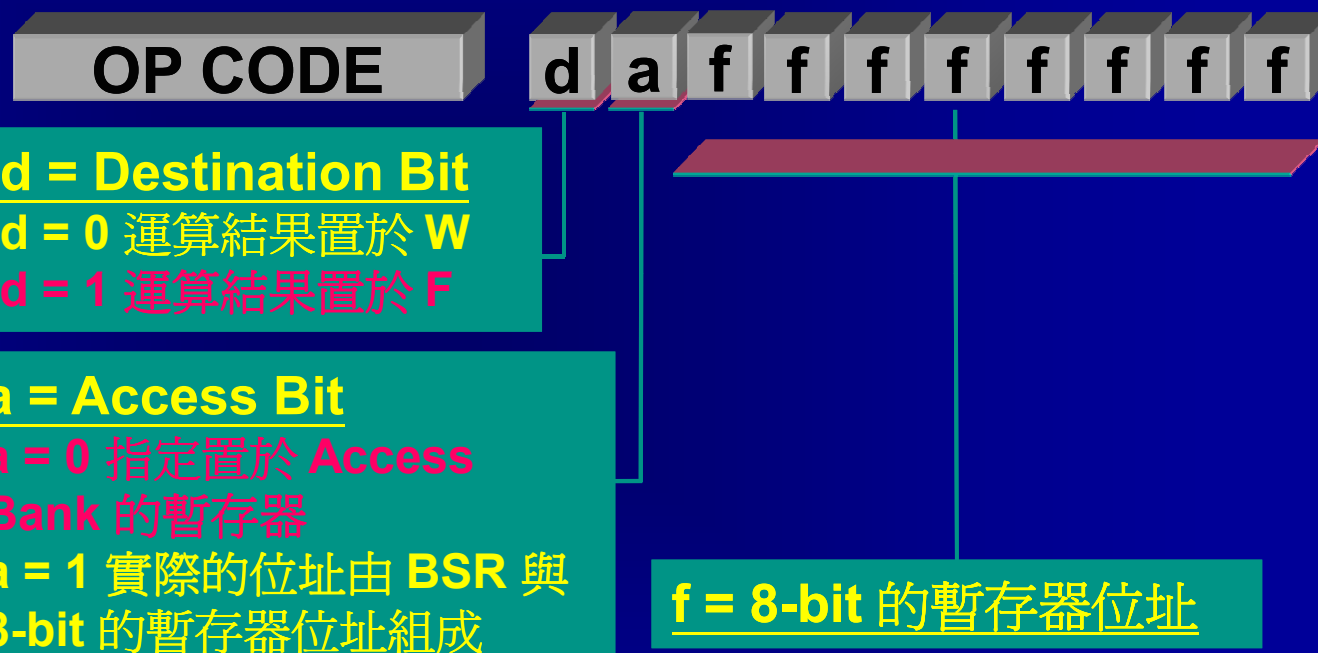
PIC18系列 MCU 指令集

以位元組為操作的對象

Byte-Oriented Operations

MULWF	f, a
NEGF	f, a
RLCF	f, d, a
RLNCF	f, d, a
RRCF	f, d, a
RRNCF	f, d, a
SETF	f, a
SUBFWB	f, d, a
SUBWF	f, d, a
SUBWFB	f, d, a
SWAPF	f, d, a
TSTFSZ	f, a
XORWF	f, d, a

16-bit Instruction for Byte Oriented Operations



Example:

SUBWF VAR1, 0, 0
SUBWF AD_VALUE, 0, a



PIC18系列 MCU 指令集

以位元組為操作的對象

Bit-Oriented Operations

BCF	f, b, a
BSF	f, b, a
BTG	f, b, a
BTFSC	f, b, a
BTFSS	f, b, a

16-bit Instruction for Bit Oriented Operations



b = 3-Bit , 用來指定
欲操作位元的位址
(Bit Number)

a = Access Bit
a = 0 指定置於 **Access Bank** 的暫存器
a = 1 實際的位址由 **BSR**
與 **8-bit** 的暫存器位址組成

f = 8-bit 的暫存器位址

Example:

BTFSC

BTFSC

STATUS, C, 0

FLAGS, BIT_0, a



PIC18系列 MCU 指令集

Literal Operations (立即數操作)

Literal Operations

ADDLW	k
ANDLW	k
IORLW	k
LFSR	f, k
MOVLB	k
MOVLW	k
MULLW	k
RETLW	k
SUBLW	k
XORLW	k

Example:
MOVLW 0x5A
MOVLW CONST

用來操作 FSR 的 16-bit, 2 Word 指令 “LFSR”



f = 2-bit FSR 的選擇位元



k = 用來指定 FSRn 的 12-bit 立即值

其他操作常數運算元的 16-bit 指令



k = 8-bit Immediate Value



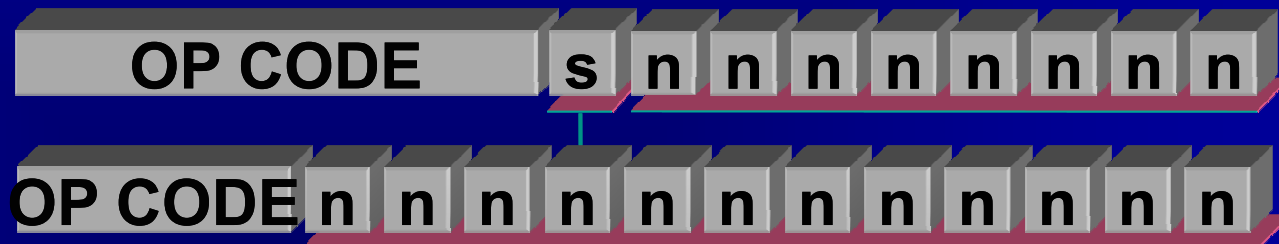
PIC18系列 MCU 指令集

Control Operations

Control Operations

BC	n
BN	n
BNC	n
BNN	n
BNOV	n
BNZ	n
BOV	n
BRA	n
BZ	n
CALL	n, s
GOTO	n
RCALL	n

16-bit Instruction for CALL and GOTO



s = 1-bit 選擇是否使用 fast Save/Restore 的功能

k = 20-bit 位址的立即值 A0=0

PC<20:1>

16-bit Instruction for RCALL and BRA



k = 11-bit 位址的立即值



PIC18系列 MCU 指令集

Control Operations

Control Operations

CLRWDT	-
DAW	-
NOP	-
POP	-
PUSH	-
RESET	-
SLEEP	-
RETURN	-
RETLW	-
<i>k</i>	
RETURN	<i>s</i>
RETFIE	<i>s</i>

16-bit Instruction for RETLW



k = 欲置於 W 暫存器後返回的8-bit 資料

16-bit Instruction for RETURN



s = 1-bit, 選擇是否使用 Fast Save/Restore 的功能



PIC18系列 MCU 指令集

Data Memory <> Program Memory

Table Pointer Operations

TBLRD*	-
TBLRD*+	-
TBLRD*-	-
TBLRD+*	-
TBLWT*	-
TBLWT*+	-
TBLWT*-	-
TBLWT+*	-

16-bit Instruction for TBLRD* or TBLWT*



$nn = 00$ * (指標不變)

$nn = 01$ *+ (查表後指標加一)

$nn = 10$ *- (查表後指標減一)

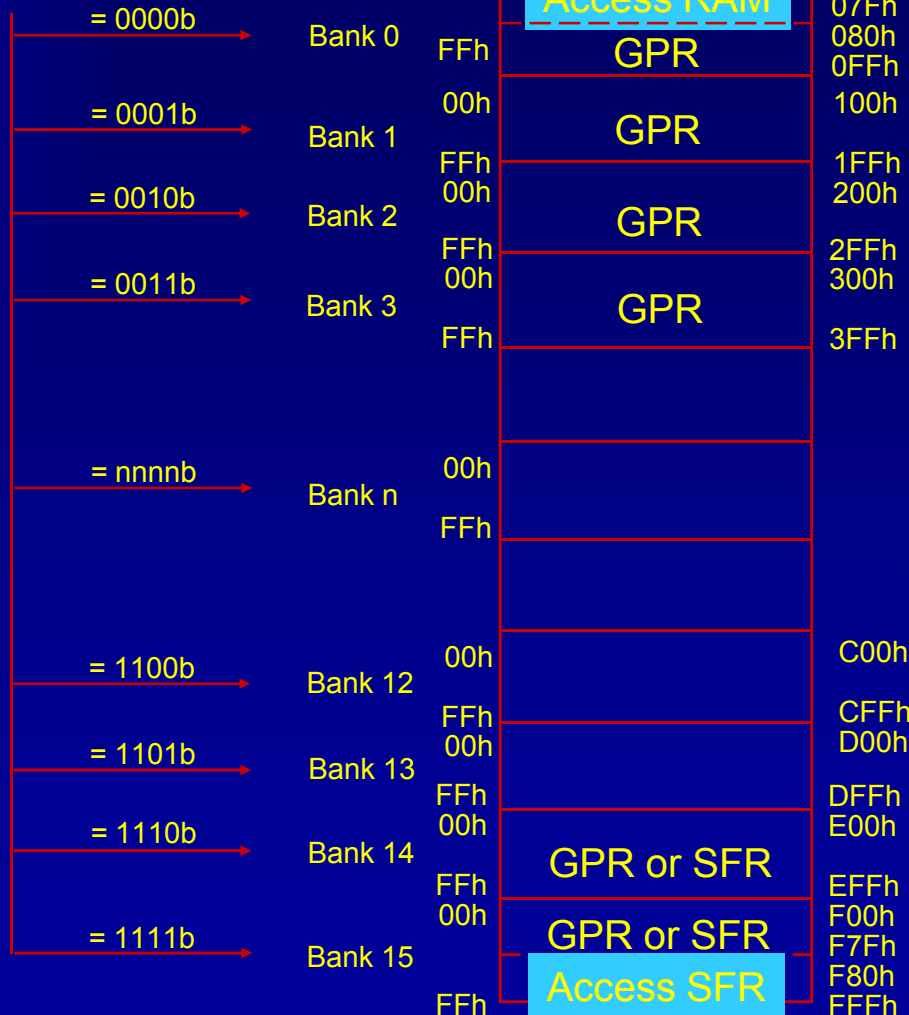
$nn = 11$ +* (先將指標加一再查表)



PIC18FXXX 加強的部份

Access BANK 的資料般移

BSR<3:0>



MOV FREG, W, 0(1)

當指令中的選項位元 **a = 0**,
BSR 的內容將不被使用而以
Virtual RAM bank 來取代。
Access Bank前半部 由 **Bank 0**
 的部份一般暫存器組成, 而後半
 部則由特殊功能暫存器 (**SFRs**)
 組合而成 (from Bank 15)

Access Bank

Access RAM
 Access SFR

00h
 7Fh
 80h
 FFh

Access RAM 與 Access SFR
 的實際大小視不同的編號而
 有所差異

當指令中的選項位元 **a = 1**,
BSR 被用來配合指令中指定的位址
 一起形成 **12 bits** 的實際位址



PIC18FXXX 加強的部份

- 相當有彈性的資料指標
 - 3 個資料指標, 各有五種操作模式
- 更方便且有效率的 **Table Read/Write** 功能
 - 以 **Byte** 為操作單位
 - 多種的指標更新選項
- 用簡單的指令集來支援複雜的架構
 - 新的 **MOVFF**, **BRA**, 以及條件式分支指令



PIC18FXXX 加強的部份

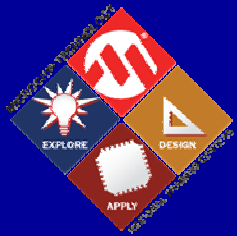
資料指標 (Data Pointer)

■ 總共有三個獨立的指標

- FSR0, FSR1, FSR2
- 指標為 12 Bits, 可定址 4K 的資料空間
- 指標有兩個暫存器組成 (例: FSR0L, FSR0H)

■ 五種指標的操作方式 (使用不同的虛擬暫存器)

- INDF_n : 直接操作指標所指的內容
- POSTINC_n : 操作指標所指內容後將指標加一
- POSTDEC_n : 操作指標所指內容後將指標減一
- PREINC_n : 將指標先加一後再對指標所指位址操作
- PLUSW_n : 將 WREG 的內容作為 Offset 值與 FSR_n 相加後以其結果為指標來存取記憶體



MICROCHIP

PIC18xxxx

特殊功能



PIC18 振盪器選擇

XT	Standard frequency crystal oscillator	100kHz - 4MHz
HS	High frequency crystal oscillator	DC - 40MHz
HS+PLL	High frequency crystal with 4x PLL	4MHz - 10MHz
LP	Low frequency crystal oscillator	5kHz - 200kHz
RC	External RC oscillator	DC - 4MHz
RCIO	External RC oscillator, OSC2=RA6	DC - 4MHz
INTRC	Internal RC oscillator	Various
EC	External Clock, $OSC2=f_{osc}/4$	DC - 40MHz
ECIO	External Clock, OSC2=RA6	DC - 40MHz

■ 提供多樣的振盪器選擇方式，讓設計更加彈性：

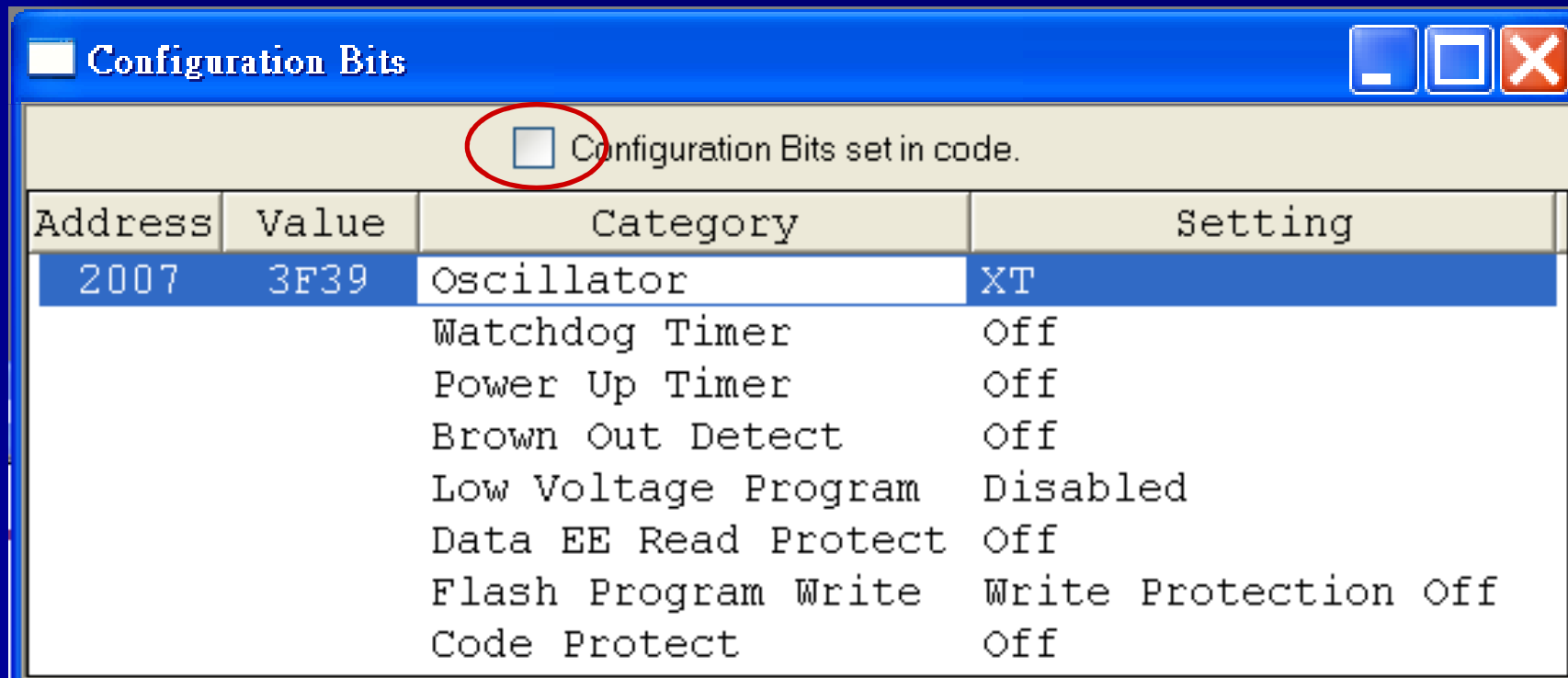
- LP 振盪模式提供了低功耗功能
- RC 或 INTRC 振盪模式提供了低成本的效益
- XT 提供一般使用的工作頻率
- HS 提供需高速處理所需的工作頻率

■ 速度範圍參考資料手冊所建議的範圍



設定 Configuration Word 的方式

- 直接在 MPLAB IDE 下設定
 - 先確定 MPLAB IDE 使用的元件名稱
 - MPLAB IDE 目錄下 : **Configure → Configuration Bits**
 - 取消 **Configuration Bits Set in code**





程式設定 Configuration Word

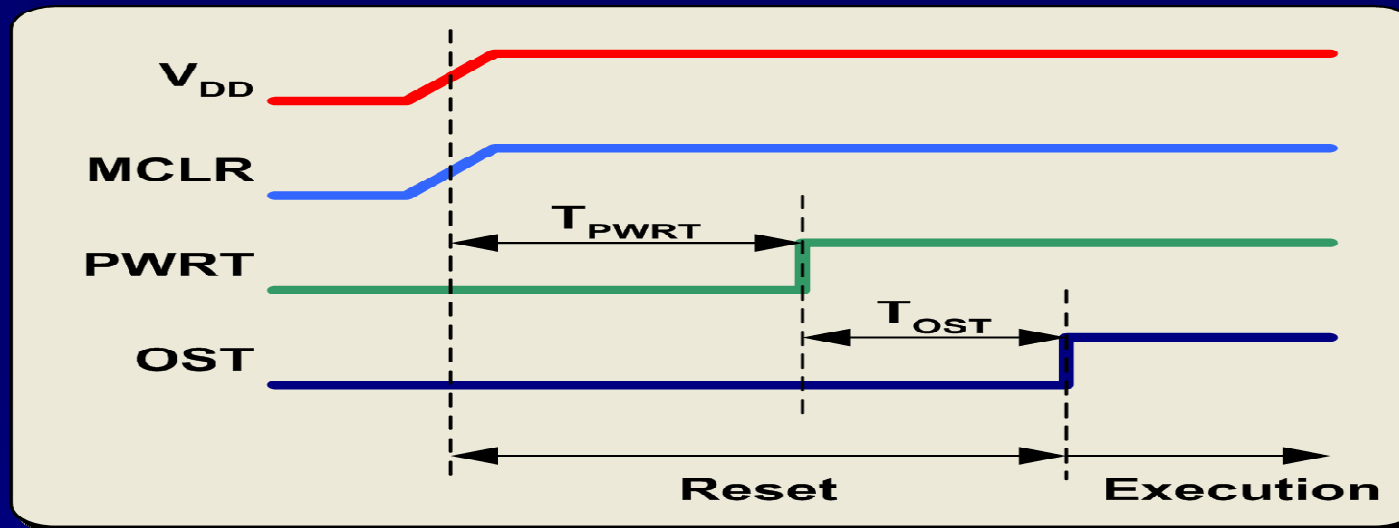
- 一般會建議利用程式內建 **Config.** 方式來設定
 - 編譯過後，**HEX** 檔會有設定值的存在
 - 避免燒錄時的人為設定上的選擇錯誤
- 舊版 MPASM 使用 **__config** 方式，新版已經改為 **CONFIG** 的語法
 - 參考 C:\Program Files\Microchip\MPASM Suite\Template\Code 裡的 18f4520tmp.asm 裡的範例
 - 相關設定的定義字參考 **inc** 檔

```
list          p=18f4520          ; list directive to define processor
#include      <p18f4520.inc>      ; processor specific variable definitions

CONFIG      OSC = LP , PWRT = ON , WDT = OFF
```



什麼是 POR, OST, PWRT



■ POR: 電源開機重置

- MCLR 腳接到 VDD，當 VDD 上升時會產生重置動作 (Reset)

■ PWRT: 電源上電延遲計時器

- 偵測到電源開機後(POR)，自動延遲72mS以確保MCU可以在工作電壓下執行

■ OST: 振盪器穩定計時器

- HS, XT, LP 三種振盪模式的啟動穩定延遲，自動計數 1024 的 Cycles 後送出振盪信號給 MCU。(不適用在 RC 模式) 只在 POR 啟動後及喚醒睡眠模式下會做此動作確保頻率的穩定



睡眠模式 (Sleep Mode)

- 指微控制器可以執行 **Sleep** 指令進入休眠狀態節省消耗功率
 - 系統主振盪時脈停止運作
 - 處理器進入靜態的狀態 – 資料保持不變
 - **Watch-Dog Timer** 因使用特定的 **RC** 振盪可以持續工作不受影響
 - 處理器進入最低耗電模式 – 通常是漏電流 (**0.1 - 2.0 μ A typical**)

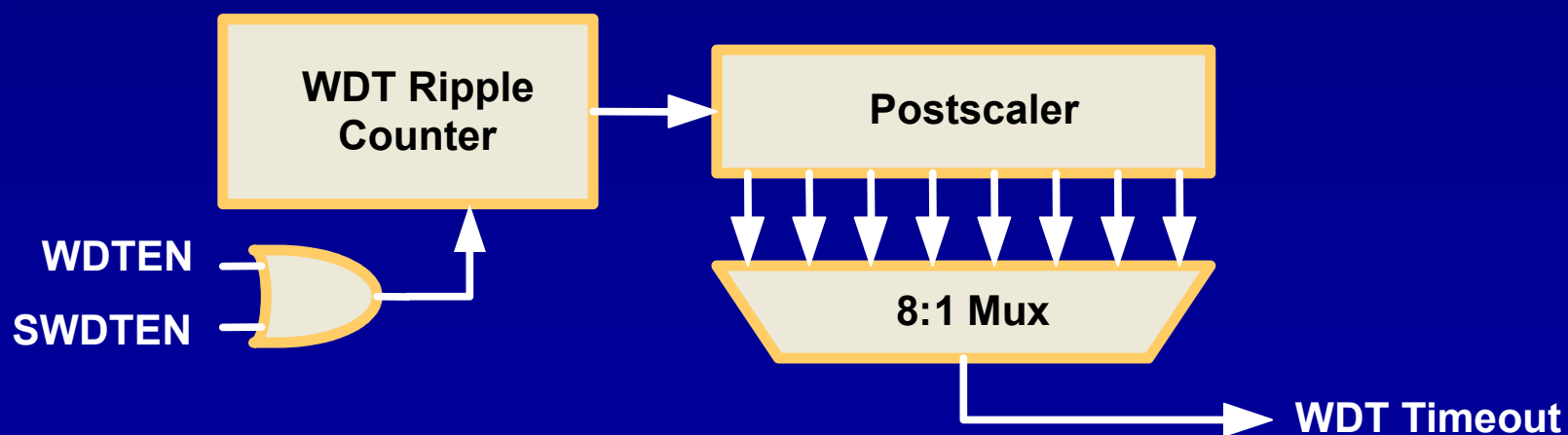
一些可以喚醒睡眠模式的外部觸發事件

MCLR	Master Clear Pin Asserted (pulled low)
WDT	Watchdog Timer Timeout
INT	INT Pin Interrupt
TMR1	Timer 1 Interrupt (or also TMR3 on PIC18)
ADC	A/D Conversion Complete Interrupt
CMP	Comparator Output Change Interrupt
CCP	Input Capture Event
PORTB	PORTB Interrupt on Change
SSP	Synchronous Serial Port (I²C Mode) Start / Stop Bit Detect Interrupt
PSP	Parallel Slave Port Read or Write



計時監視器 (Watchdog Timer)

- 以硬體方式監測軟體的故障並恢復運作
- 使用內建的獨立運作的 **RC** 振盪器
- **WDT** 的時間計數器可以用 **CLRWDT** 指令歸零
- 使用 **Config** 硬體啟動 **WDT** 後無法用軟體方式關閉
- **WDT** 發生計時溢位時會重置微處理控制器 (**Reset MCU**)
- 可程式控制計時溢位間：**18ms to 3.0s**
- 可以在 **SLEEP Mode** 下透過溢位來喚醒 **CPU**



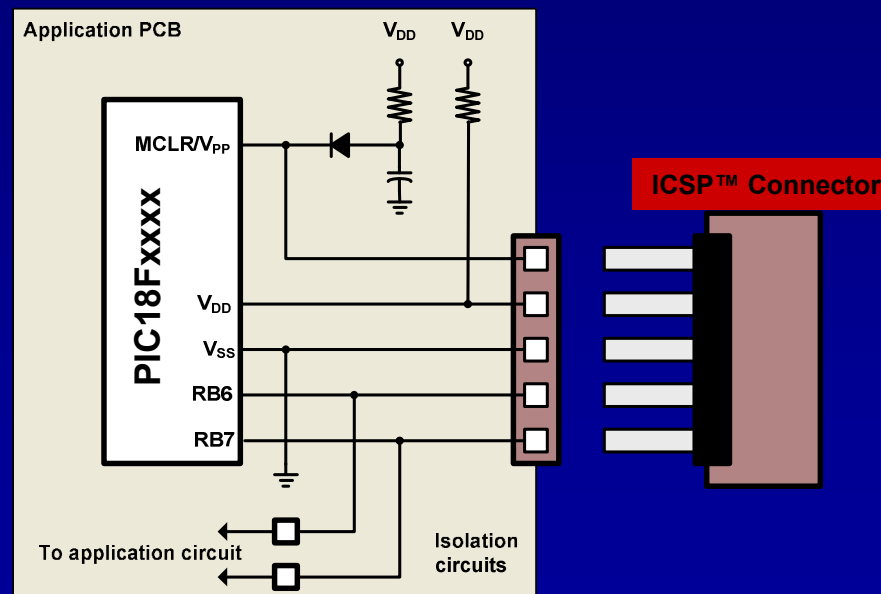


串列燒錄功能

In-Circuit Serial Programming™

- 燒錄時只用到兩支腳位 (請勿接電阻到地)
 - RB6 → PGC
 - RB7 → PGD
- V_{pp} 會有 13V 的觸發電壓輸入，要使用 10K ohm 電阻接 V_{DD}，最好不要再接電容到地以免影響 V_{pp} 上升時間造成燒錄錯誤。
- 這種燒錄介面可以使用在 Microchip 所有的開發工具裏
 - ICD2, ICD3, Real ICE
 - PICkit2, PM3

Pin	Function
V _{PP}	Programming Voltage = 13V
V _{DD}	Supply Voltage
V _{SS}	Ground
RB6	Clock Input
RB7	Data I/O & Command Input





掉電偵測功能

BOR –Brown Out Reset

- 當工作電壓突然掉到所設定的低電壓門檻時，**PIC** 將會進入重置模式
- 避免一些奇怪及不可預期的運作（電源不穩時所造成的誤動作）
- 功能像一般具延遲功能的 **Reset** 電路，且偵測的電壓門檻是可以設定的以配合各種工作電壓的需求
- **BOR** 功能一但被啓動就無法關閉，如果要考慮極低的耗電時可能須關閉 **BOR** 功能
- 使用 **BOR** 功能可以防止系統開機不良的情形
 - 短時間內 **On , Off , On , Off , On** 時所產生的當機



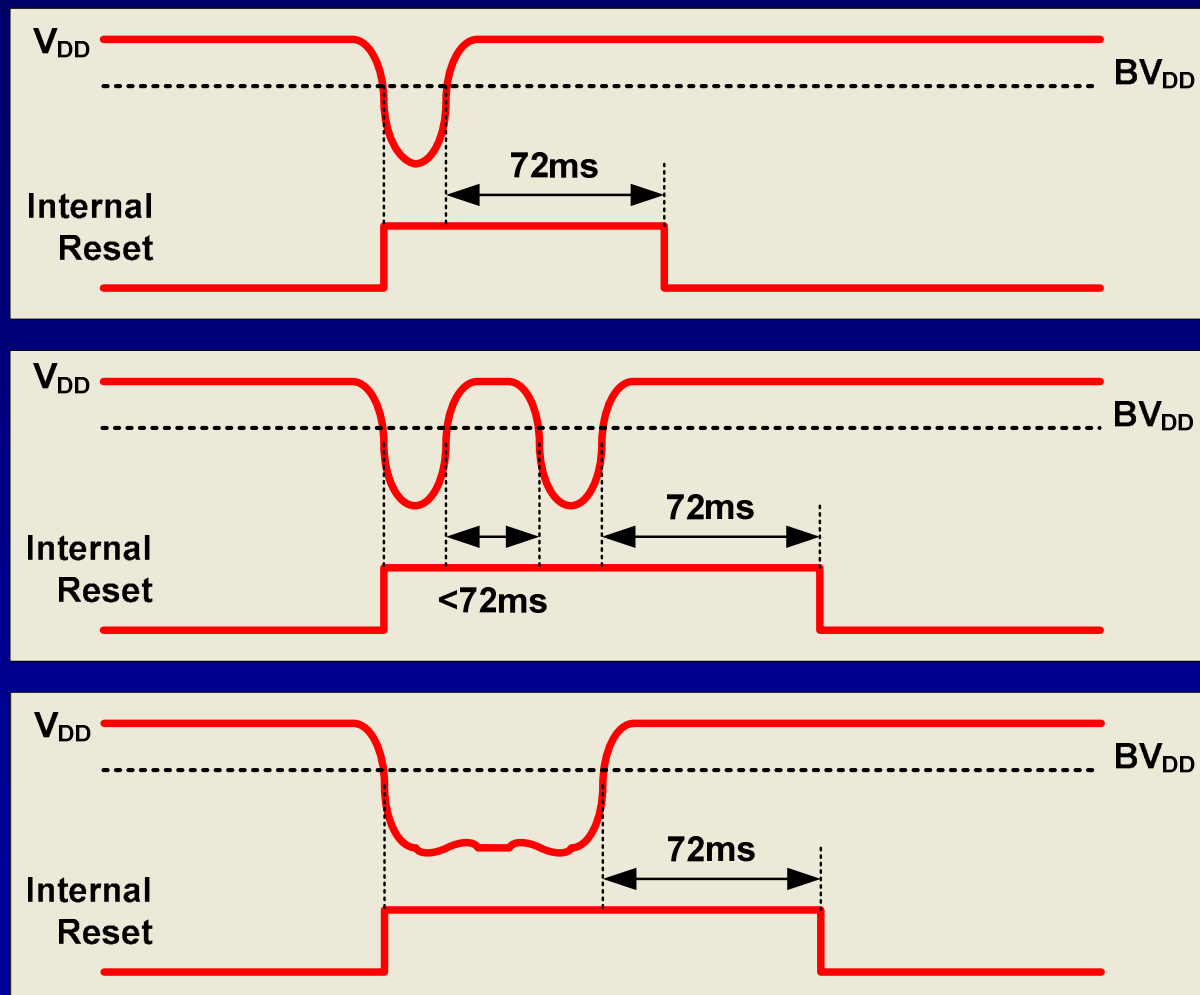
PBOR – 可程式化掉電偵測功能

- 直接在 **Configuration** 設定 (燒錄前設定)
 - 此功能無法用軟體方式開啓或關閉
- 可以設定的電壓門檻 **BV_{DD}** :
 - **2.5V** – 最低電壓門檻 **V_{DD}** , – 適用在 **3.0V** 工作電壓
 - **2.7V** – 適用在 **3.3V** 工作電壓
 - **4.2V**
 - **4.5V** – 適用在 **5V** 工作電壓
- 如果以上的電壓門檻值無法滿足的話，請使用外加的 **Reset** 元件
 - **Microchip's supervisor : MCP1xx, MCP8xx/TCM8xx, or TC12xx**



可程式化掉電偵測功能

- 當 V_{DD} 電壓升高到臨界電壓點以上時，能持續保持在重置(Reset)狀態，並維持 72mS 的延遲計時

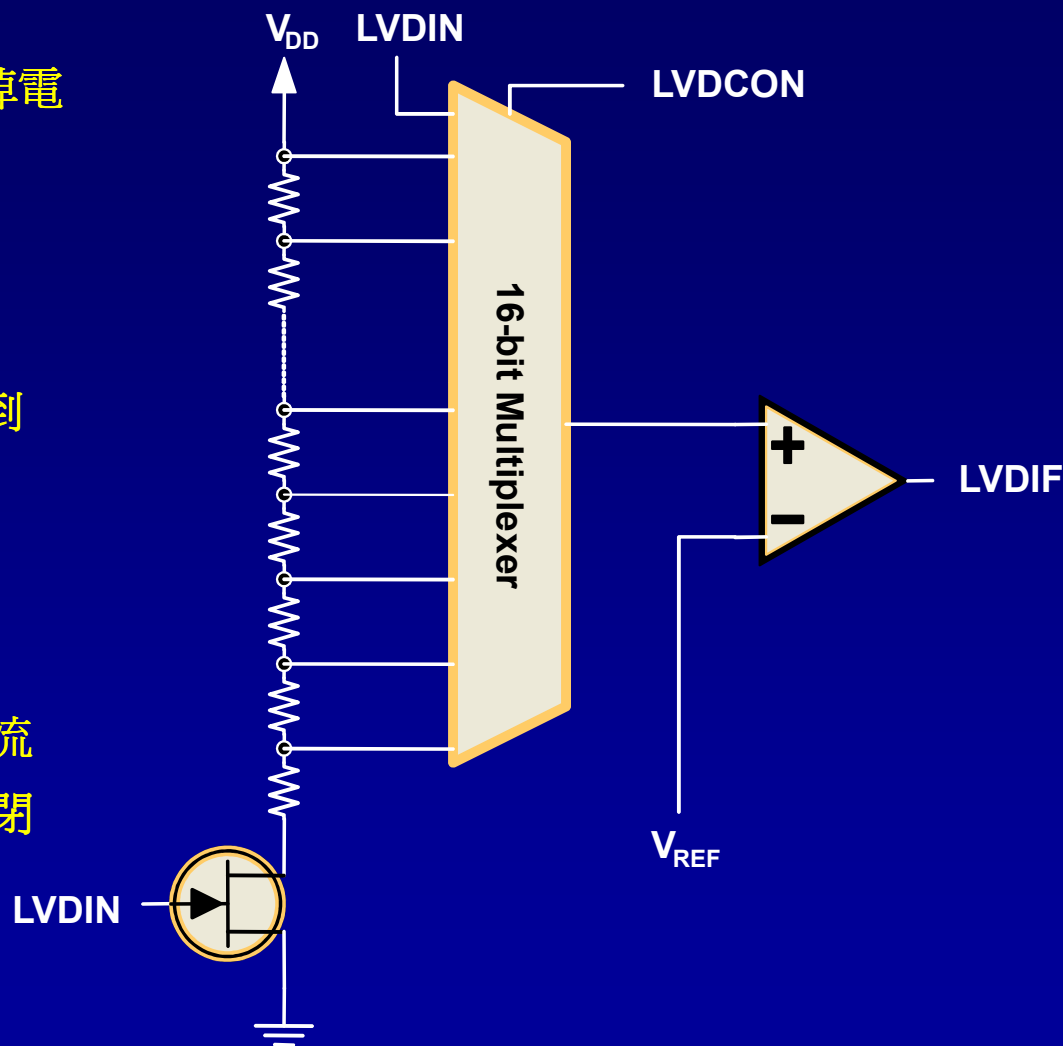




可程式化低電壓偵測功能

PLVD – Programmable Low Voltage Detect

- 可以設定比 **BOR** 更早偵測出掉電狀態，進行重要資料的保存 (EEPROM)
- 16 種可以選擇電壓偵測位準：
 - 1.8V 到 4.5V 每一階為 0.1 到 0.2V 的步進電壓值
 - 可使用外部類比電壓輸入
- 內建參考電壓元 V_{REF}
- 此功能為類比架構消耗較多電流，如考慮功耗可用軟體方式關閉此模組。





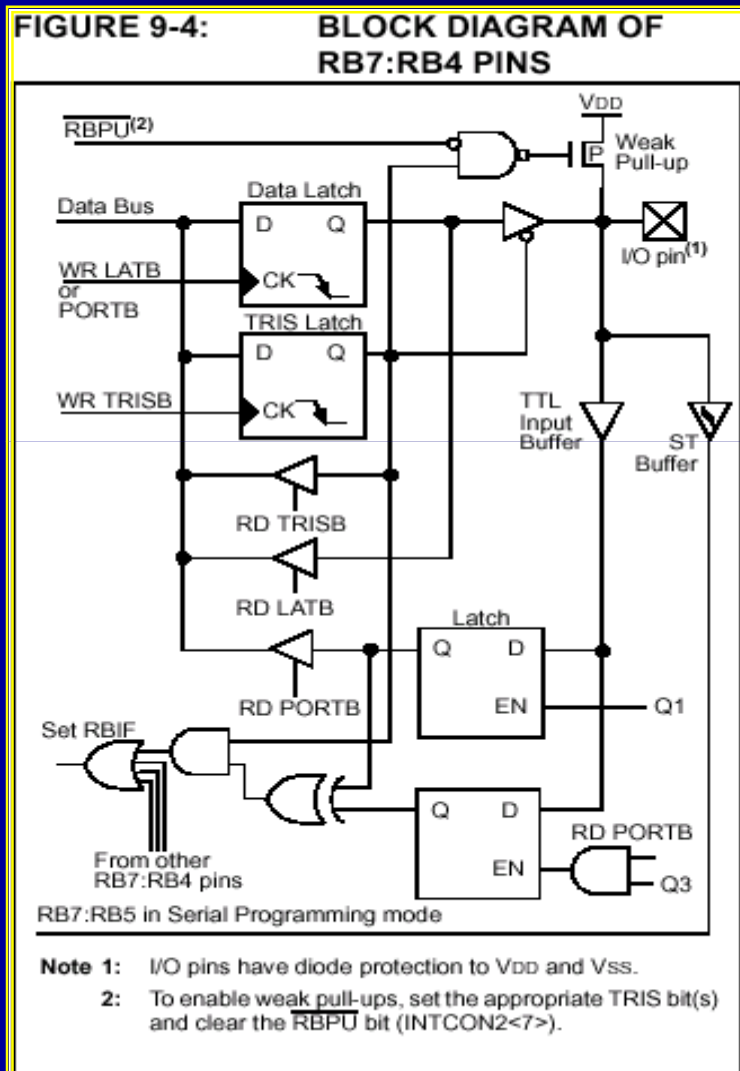
MICROCHIP

PIC18Fxxxx

I/O 介紹與運用



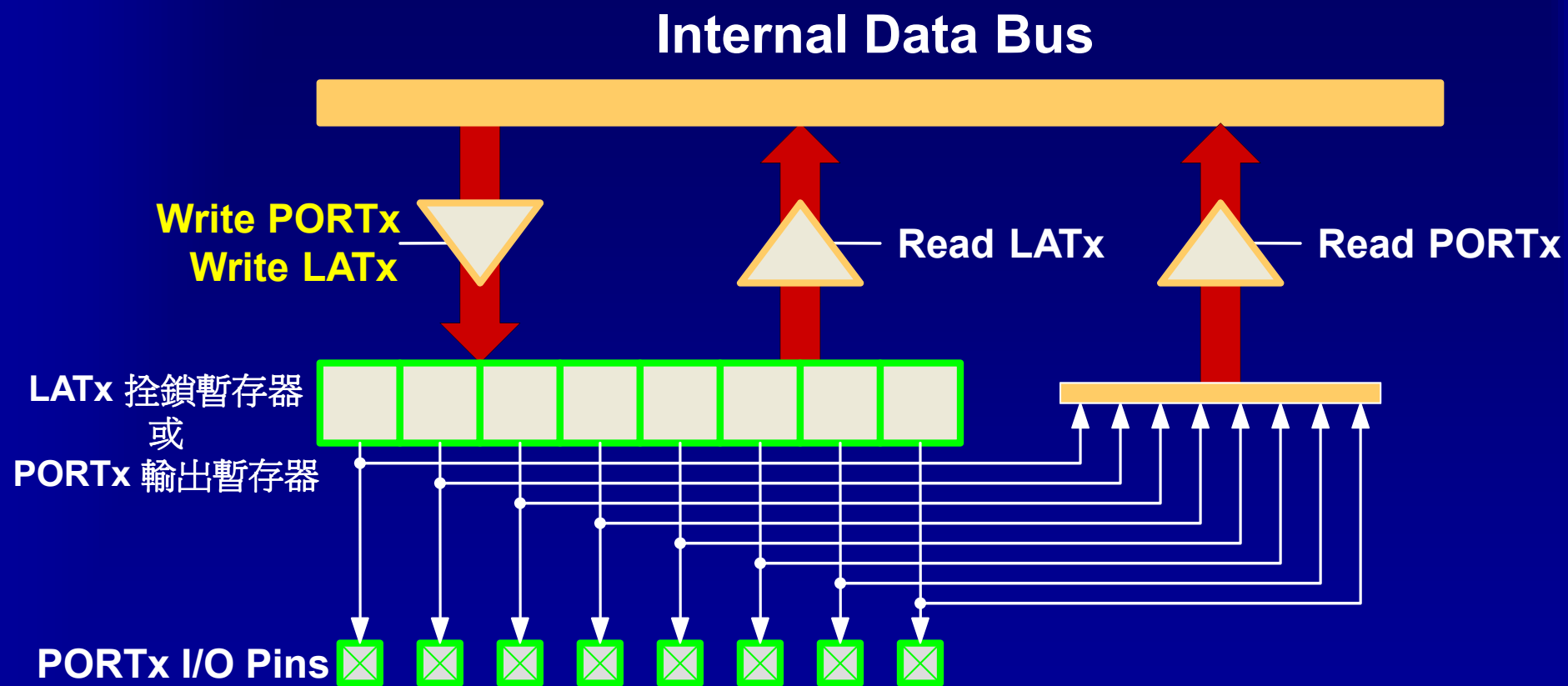
PIC18 系列的周邊：I/O Ports



- 輸入/輸出腳，為一般 MCU 最基本的工作
- PIC18 系列的 I/O 多了一個 LATx，可有效解決所謂 Read-Modify-Write 指令動作帶來的困擾
- 每個 I/O Port 皆有三個獨立的控制暫存器
 - PORTx
 - LATx
 - TRISx



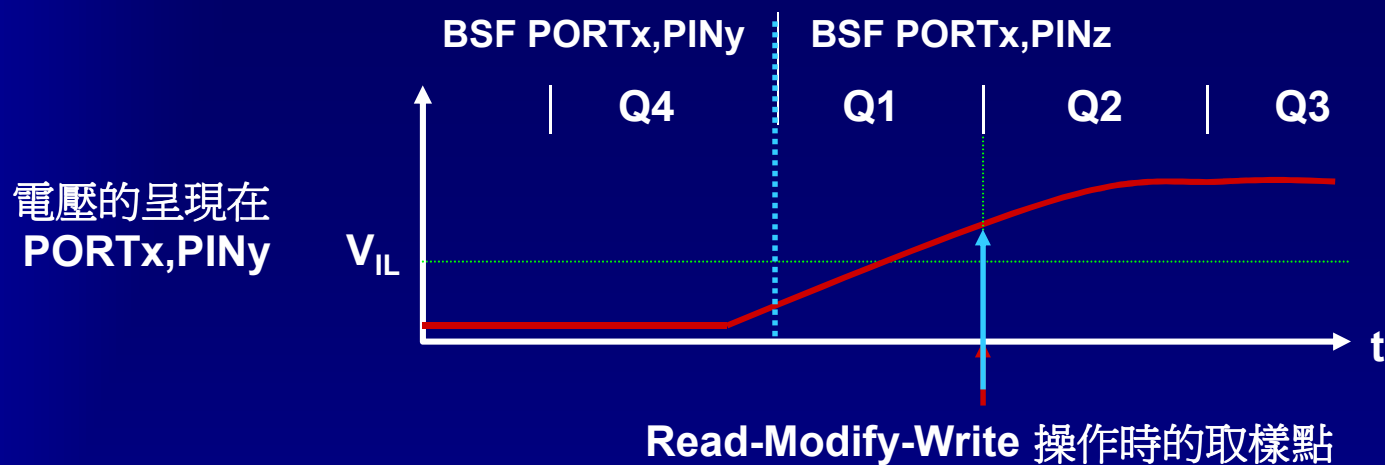
I/O 的輸出入暫存器



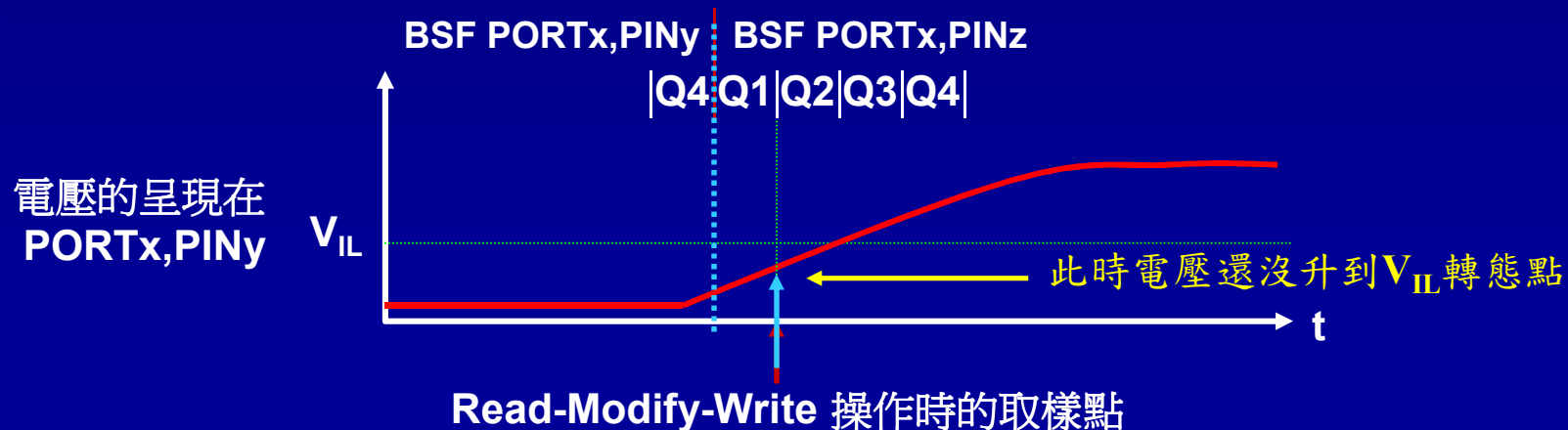


為什麼加入 LATx 拴鎖暫存器

1. 如果工作在較低頻率 或 較小的負載電容環境



2. 如果工作在較高頻率 或 較大的負載電容環境





如何解決 Read-Modify-Write

■ 如果是對相同的 **PORT_x** 連續操作

- 解決之道是在連續兩個 BSF/BCF 指令間加入適當個 NOP 指令

- BSF PORTA,5
- NOP
- BSF PORTA,0

- 使用 C18 則可加入 Nop() 的巨集

■ 建議直接使用 **LAT_x** 的來取代 **PORT_x** 的操作

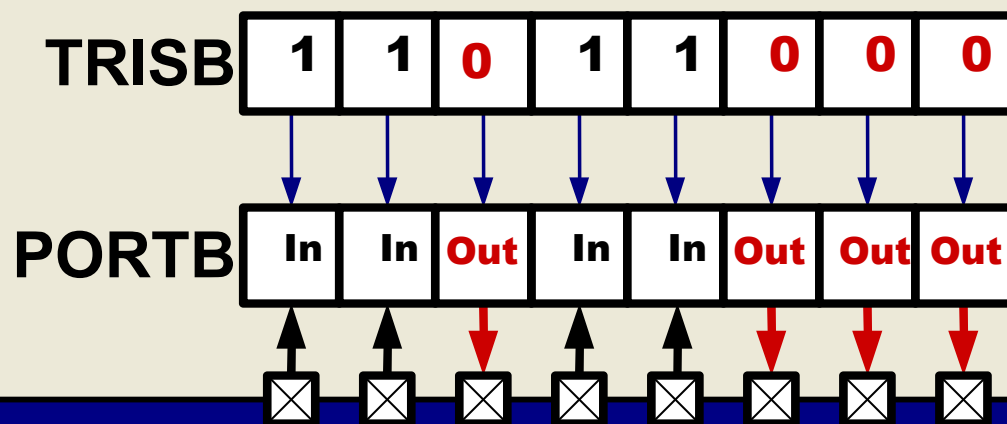
- 這樣直接就對拴鎖暫存器操作，而不是對外面的 PORT 操作也就沒有因電容所造成的延遲

- BSF LATA,5
- BSF LATA,0



I/O 輸出 / 輸入控制

I/O Pin 方向控制



- **TRIS_x** 的每一獨立位元可單獨去控制相對應 **PORT_x** 裡的位元的資料方向 (**Reset** 的初始定義為輸入功能)
- **1 = 輸入, 0 = 輸出**



類比 或 數位 腳位？

- 有些接腳是類比輸入與數位共用同一隻接腳（內定為類比輸入）
- 類比輸入有 **ADC** 輸入與電壓比較器輸入
- **ADCON1** 暫存器用來設定腳位是類比輸入還是數位 **I/O**

Bit 7				Bit 0			
-	-	X	X	1	1	1	1
VCFG1		VCFG0		PCFG3		PCFG2	

設定最低 4 bits 為 '1' 時，關閉所有 **ADC** 輸入腳功能



PIC18 系列的周邊：I/O Ports

- 最多可擁有 68 個標準輸入/輸出接腳 (PIC18F8720)
- 某些接腳具有可設定為：周邊功能或一般接腳
- 接腳具有25mA的高驅動電流 (sink & source)
 - 可直接驅動 LEDs 及耗電流較高的負載
- 每一接腳可直接做位元運算，且只需一個指令時間
- 每一接腳均具有下列功能：
 - 程式設定其為輸入或輸出腳
 - 資料門鎖 (可直接進行 read-modify-writes 的功能)
 - 輸入資料直接讀取輸入的腳位
- 每一接腳有獨立的 ESD 保護二極體

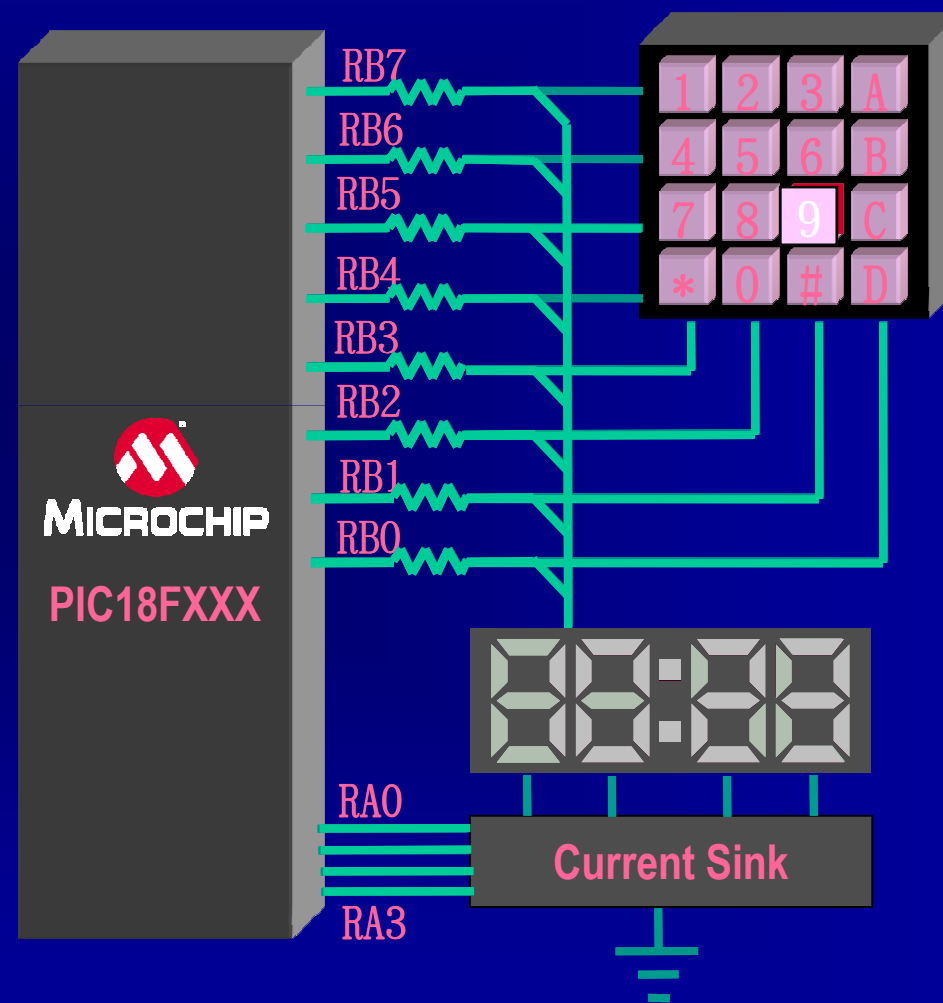


PIC18 系列的周邊：I/O Ports

典型的應用範例

■ 顯示與鍵盤

- 利用最少的外加零件及軟體來完成：點亮 4 個七節顯示器並掃描一個 4 X 4 的矩陣鍵盤。
- RB0 - RB7 為顯示字形的輸出，RA0 - RA3 控制欲被點亮的七節顯示器位置。
- 每隔 5mS 點亮一個七節顯示器，共 20mS 完成一周。
- 在點亮七節顯示器一周後，改變 RA0~RA3 為輸入關閉七節顯示器 100uS 以掃描鍵盤。
- RB0 - RB3 為輸出 pins，RB4 - RB7 為輸入 pins（內含 pull-up 電阻）

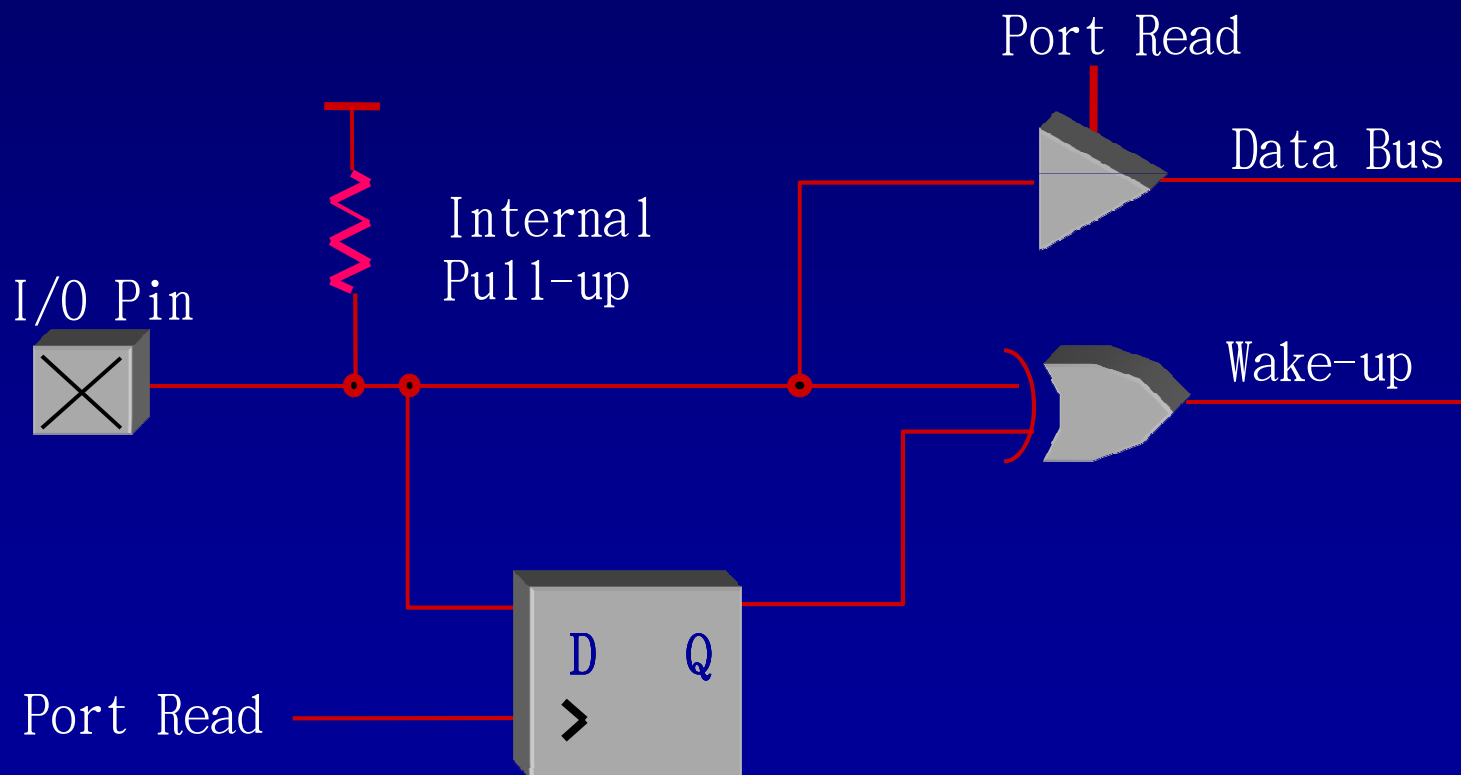




PIC18 系列的周邊 I/O Ports

PORTB 的特殊功能：Interrupt on Change

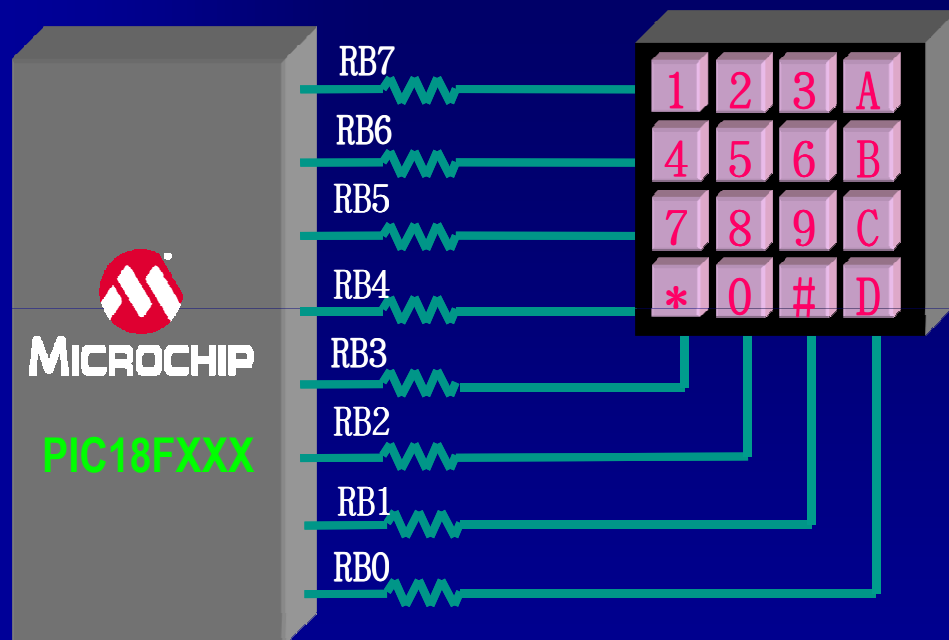
- **PORTB 的 RB4 至 RB7 可因為偵測到接腳電位的改變來產生中斷，並喚醒PICmicro**





PIC18 系列的周邊：I/O Ports

PORTB 的特殊功能：Interrupt on Change



* Optional resistors for ESD protection

- RB4 - RB7 內有 pull-up 電阻，設定為輸入腳
- RB0-RB3 為輸出腳並同時輸出低電位
- 對 PORTB 做一次假讀取以記錄 RB4-RB7 目前的電位並進入睡眠模式
- 當任一鍵被按下時，RB4 - RB7 會有一電位的改變進而產生中斷喚醒MCU
- 對鍵盤進行掃描動作讀取按鍵值



正確的語法表達 (一)

數值、數字(字元)表示法

■ 十進制表示(Decimal): $D'<\text{十進制數目}> \text{ ' \& .<十進制數目> }$

	MOVLW	D'100'	; 載入常數 100 ₍₁₀₎ to W Reg.
	MOVLW	.100	; 載入常數 100 ₍₁₀₎ to W Reg.
Const1	EQU	D'200'	; Const1 = 200 (十進制)

■ 十六進制表示(Hexadecimal): $H'<\text{十六進制數目}> \text{ ' \& 0x<十六進制數目> \& <十六進制數目> h }$

	MOVLW	H'3F'	; 載入常數 3F ₍₁₆₎ to W Reg.
	MOVLW	0x3F	; 載入常數 3F ₍₁₆₎ to W Reg.
	MOVLW	0FEh	; 載入常數 FE ₍₁₆₎ to W Reg.
Const1	EQU	H'5A'	; Const1 = 5A (十六進制)



正確的語法表達 (二)

數值、數字(字元)表示法

■ 二進制表示(Binary): **B'**<二進制數目> ‘

MOVLW B'11110000' ; 載入常數 0xF0 to W Reg.

Const1 EQU B'01010101' ; Const1 = 01010101 (二進制)

■ 字元(ASCII): **A'**<字元> ‘&’ <字元> ‘

MOVLW A'R' ; 載入字元 “R” to W Reg.

MOVLW 'c' ; 載入字元 “c” to W Reg.

Const1 EQU 'a' ; Const1 = 小寫的字元 A



必需要使用的虛擬指令

- **LIST** - 目錄控制 (Listing Control)
 - `list p=PIC18F4520`
- **#INCLUDE** - 加入一原始檔、定義檔或敘述檔
 - `#include <18f4520.inc>`
- **EQU** - 宣告常數、變數 (不可重新定位)
 - `memory equ 0x3f`
 - `count equ .100`
 - `io_set equ B'11000011'`
- **ORG** - 設定程式組譯的起始位址
 - `org 0x00 ; 組譯位址從 “00h”開始`
 - `org 0x30 ; 組譯位址從 “30h”開始`
- **END** - 程式結束



虛擬指令的使用範例

```
list          p=18F4520          ; 定義使用的 MCU 為 PIC18F4520
#include       <p18F4520.inc>      ; 使用 18F4520 的標準定義檔

;***** 定義變數 ‘常數、參數區’ *****
T_DELAY      EQU          D'100'      ; 設定常數值
dly_count    EQU          H'00        ; 設定變數位址
;*****

ORG          0x000          ; 設定程式執行位址從 “0000” 開始
clrf         PCLATU
clrf         PCLATH
goto        main

;

ORG          0x0008          ; 高優先權中斷向量進入位址
;***** 中斷處理副程式區 *****
retfie       s              ; 中斷返回

;

main         movlw         T_DELAY      ; 主程式開始
            movwf         dly_count
;** remaining code goes here
END          ; 程式結束
```




練習一：I/O 的操作

- 本練習的程式 **Ex1.asm** 放在
 - **..\102_ASP_TW\exer\Ex1.asm**
- 建立一個屬於這個練習的 **Project** 名稱定為 **Ex1.mcp**
- 修改這個練習，使程式能依按鍵的選擇來控制 **LED**
 - JP9 的 2 & 3 腳 (左側) 需用 **Jumper** 短路，**LED** 接地
 - 設定 **RB0** , **RA4** 為輸入
 - 設定 **PORTD** 為輸出 (驅動 **LED**)
 - 按 **SW2 (RB0)** 時，點亮 **LED0 – LED3**
 - 按 **SW3 (RA4)** 時，點亮 **LED4 – LED7**



MICROCHIP

Timer & CCP Module

介紹



- [illegible]

102 ASP-TW



PIC18 系列的周邊

Timer1 and Timer3

- **16-bit** 模式的計數器或計時器
- 由兩個可讀/寫的 **8-bit** 計數器串聯而成
- 預除器有四種選擇： $\div 1$ ， $\div 2$ ， $\div 4$ ，**or** $\div 8$
- 三種功能：計時器，同步模式計數器，非同步模式計數器（睡眠模式下使用非同步時序喚醒）
- 專用石英振盪電路可作為外部計數時序或第二系統時序（**System Clock**）選擇
- 當計數器或計時器產生溢位時 **FFFFh to 0000h**，即產生中斷





PIC18 系列的周邊

Timer1 and Timer3 as a Real-Time Clock

- 使用 32.768 kHz 的石英振盪器作為 Timer1 或 Timer3 的計時訊號
 - 系統的主振盪器可使用低價的 **RC** 振盪電路。
 - 睡眠模式下，**Timer1 (Timer3)** 繼續遞增。
 - 當 **Timer1 (Timer3)** 產生溢位時，中斷會喚醒 **MCU**，主振盪器 (**RC**) 立即起振。
 - 可利用 **32.768KHz** 的值來校正 **RC** 振盪。



PIC18 系列的周邊 Timer2

- **8-bit** 模式的計時器，有預除器及後除器之功能
- **PWM** 輸出模式下，基本的頻率來源
- **TMR2** 為一可讀、寫具有自動載入功能的計時器
- **TMR2** 會自動加一並與設定的值相比；若相等則送出訊號至後除器或產生中斷，並自動將自己清除為零，重新計時
- **MSSP (SPI™)** 傳送速率的設定

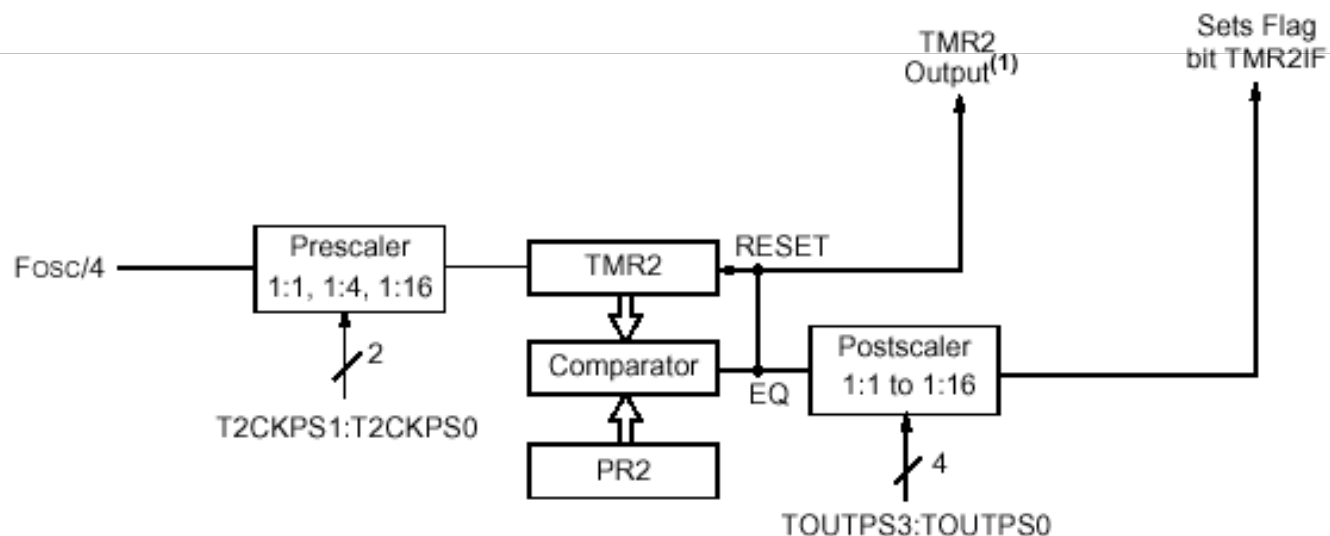


PIC18 系列的周邊 Timer2

■ Timer2 的方塊圖

➤ PR2 的內容為 Timer2 的週期 (0 → 255)

FIGURE 12-1: TIMER2 BLOCK DIAGRAM



Note 1: TMR2 register output can be software selected by the SSP Module as a baud clock.



PIC18 系列的周邊

Capture / Compare / PWM (CCP) Module

- **CCP** 模組可規劃為：脈波量測器、計時比較模式、脈衝寬度調變器
- 當起動脈波量測或計時比較功能時，使用 **TMR1** 或 **TMR3 (16-Bit Timer)** 當計時的基準。
- 在脈衝寬度調變輸出模式下，啓動**TMR2**當脈衝寬度(**Duty**)和週期(**Period**)的計時基準。



PIC18 系列的周邊

Capture / Compare / PWM (CCP)

Module (*continued*)

■ 定義：

- Input Capture：計算輸入脈衝的時間、寬度及頻率；必須使用**16-bit**的 **Timer1** 為計時基準。
- Output Compare：設定一 **16-bit** 的計時比較值與 **16-bit** 的 **Timer1** 相比，若相等則依其設定來改變輸出。
- PWM (Pulse Width Modulation)：頻率固定的方波，藉由改變其脈衝寬度(**Duty**)來進行數值的調變



PIC18 系列的周邊

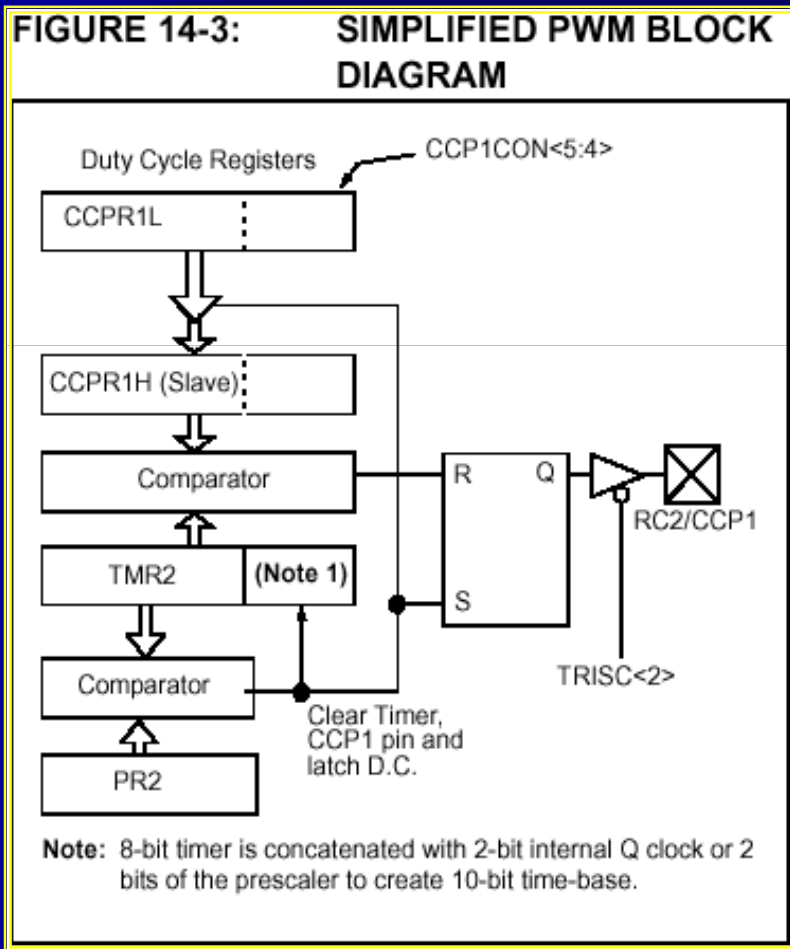
CCP Module: **PWM** Mode

- 最高可達 **10-bit** 的解析度 (當工作頻率為 **40 Mhz**)
 - 輸出頻率 39.06 kHz 時，解析度為 10-bit
 - ✓ $40\text{MHz} / 1024 = 39.06\text{KHz}$
 - 輸出頻率 156.25 kHz 時，解析度為 8-bit
 - 輸出頻率 312.5 kHz 時，解析度為 7-bit
- 脈衝寬度的最高解析度為 **25 ns**
 - (**@ 40 MHz, high-resolution mode**)

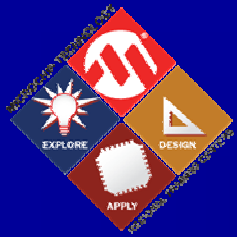


PIC18 系列的周邊

CCP Module: PWM Mode



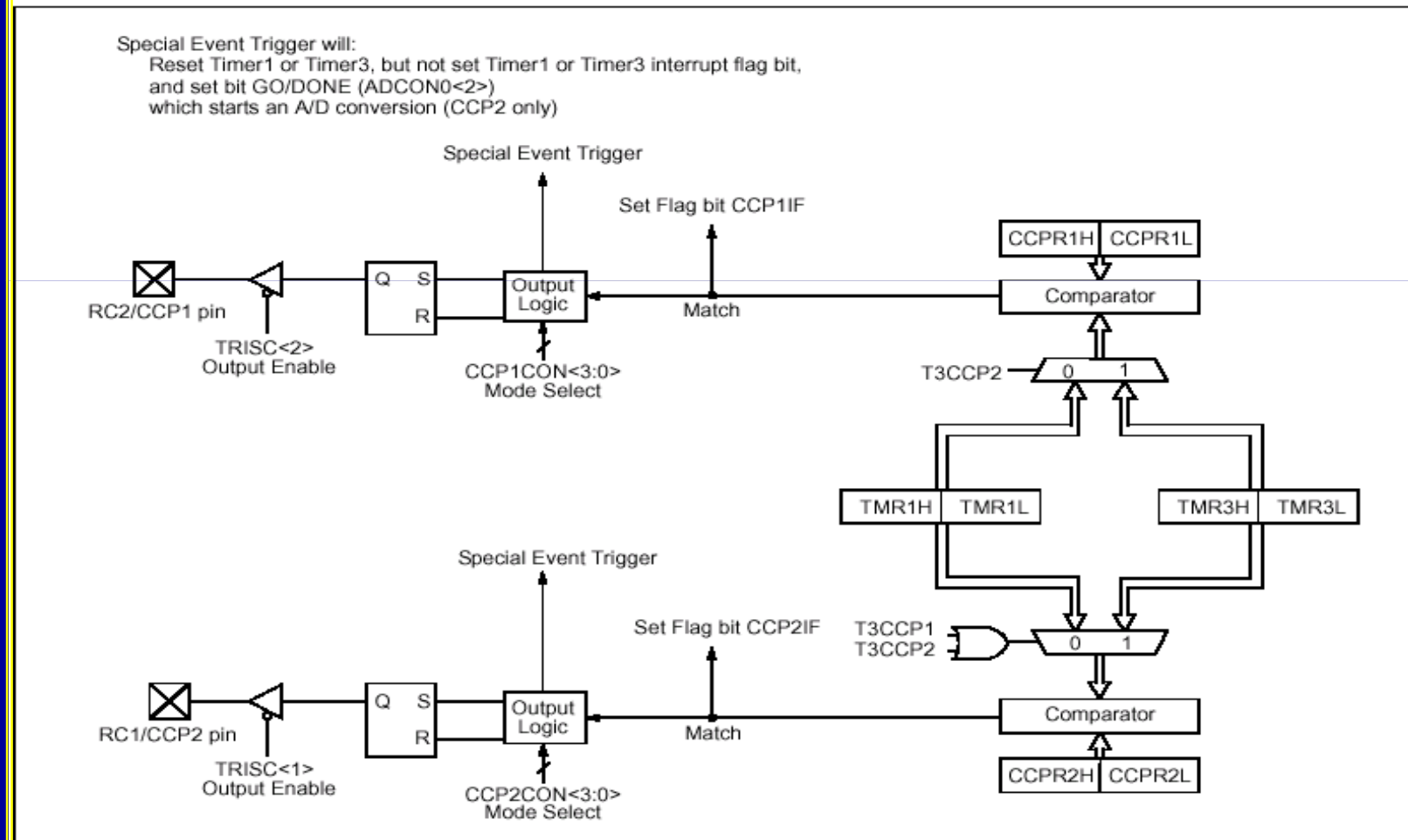
- Period 由 PR2 設定
- Duty Cycle 由 CCPR1H 決定
- PWM 為 10-bit，較低的兩個位元置於 CCP1CON 的 bit <5:4>
- 每一新的週期開始時，CCPR1L 的設定值就會重新被載入且正反器被設為 1 輸出
- 當 TMR2 = Duty Cycle (CCPR1H)，則正反器的輸出會被清除為零



PIC18 系列的周邊

CCP Module: Output Compare Mode

FIGURE 14-2: COMPARE MODE OPERATION BLOCK DIAGRAM

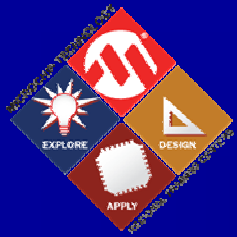




PIC18 系列的周邊

CCP Module: Output Compare Mode

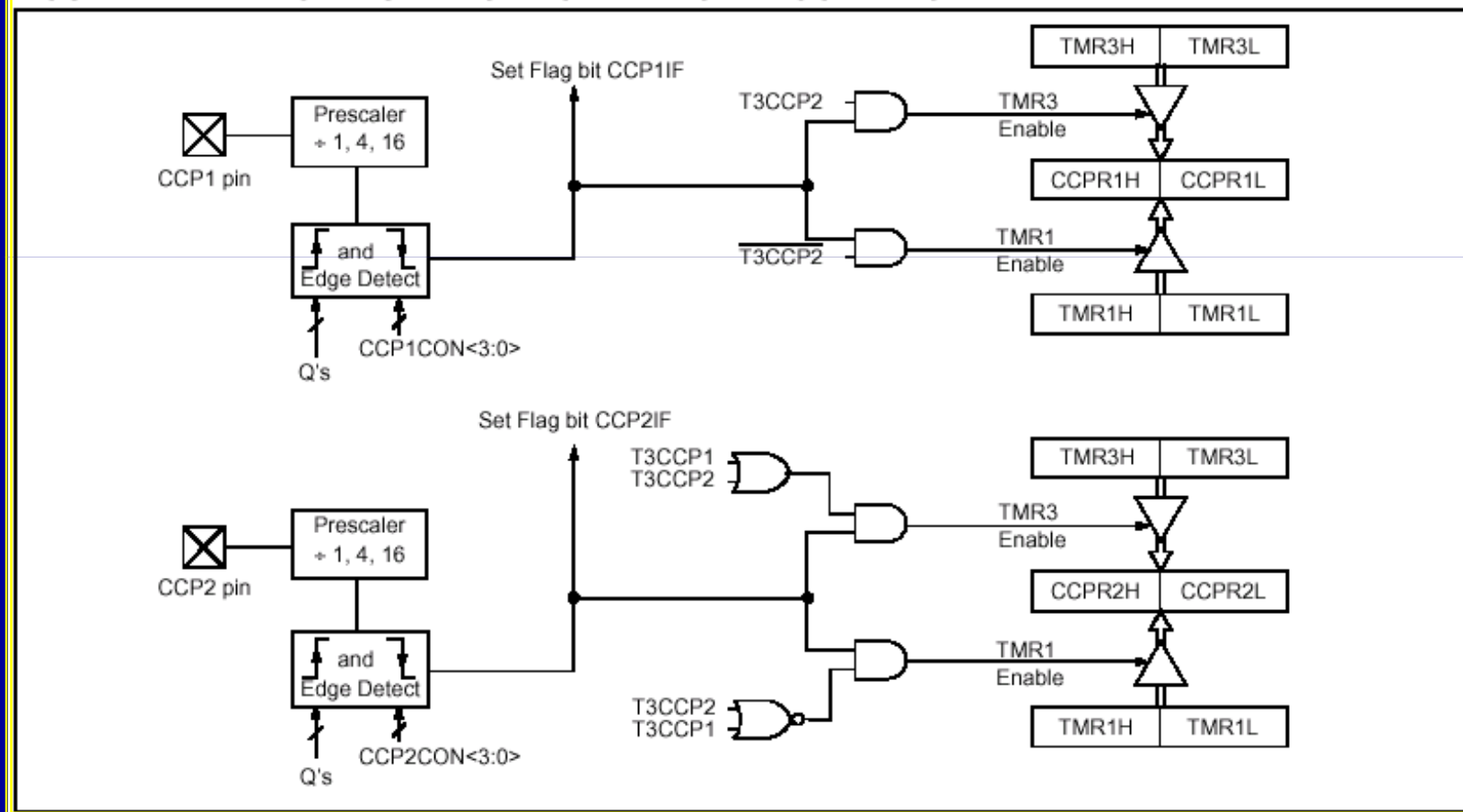
- 當16-bit CCPRx 暫存器的內容值等於計時器 TMR1或TMR3時，CCPx 腳可經由事先的設定有以下之輸出變化：
 - High 輸出
 - Low 輸出
 - 輸出不改變
- 比較值相等時，也可產生中斷
- 亦可設定比較值相等時，自動清除 TMR1 或啟動 A/D 轉換器



PIC18 系列的周邊

CCP Module: Input Capture Mode

FIGURE 14-1: CAPTURE MODE OPERATION BLOCK DIAGRAM





PIC18 系列的周邊

CCP Module: Input Capture Mode

- 當下列事件發生在輸入腳CCPx時，TMR1或是 TMR3 目前的計數值(16-Bit)會直接被複製並儲存在CCPRxH & CCPRxL 暫存器
 - 每一上緣信號 (上緣或下緣觸發可設定)
 - 每一下緣信號
 - 每四個上緣或下緣信號
 - 每十六個上緣或下緣信號
- 事件的觸發也可產生中斷



18F452 中斷處理

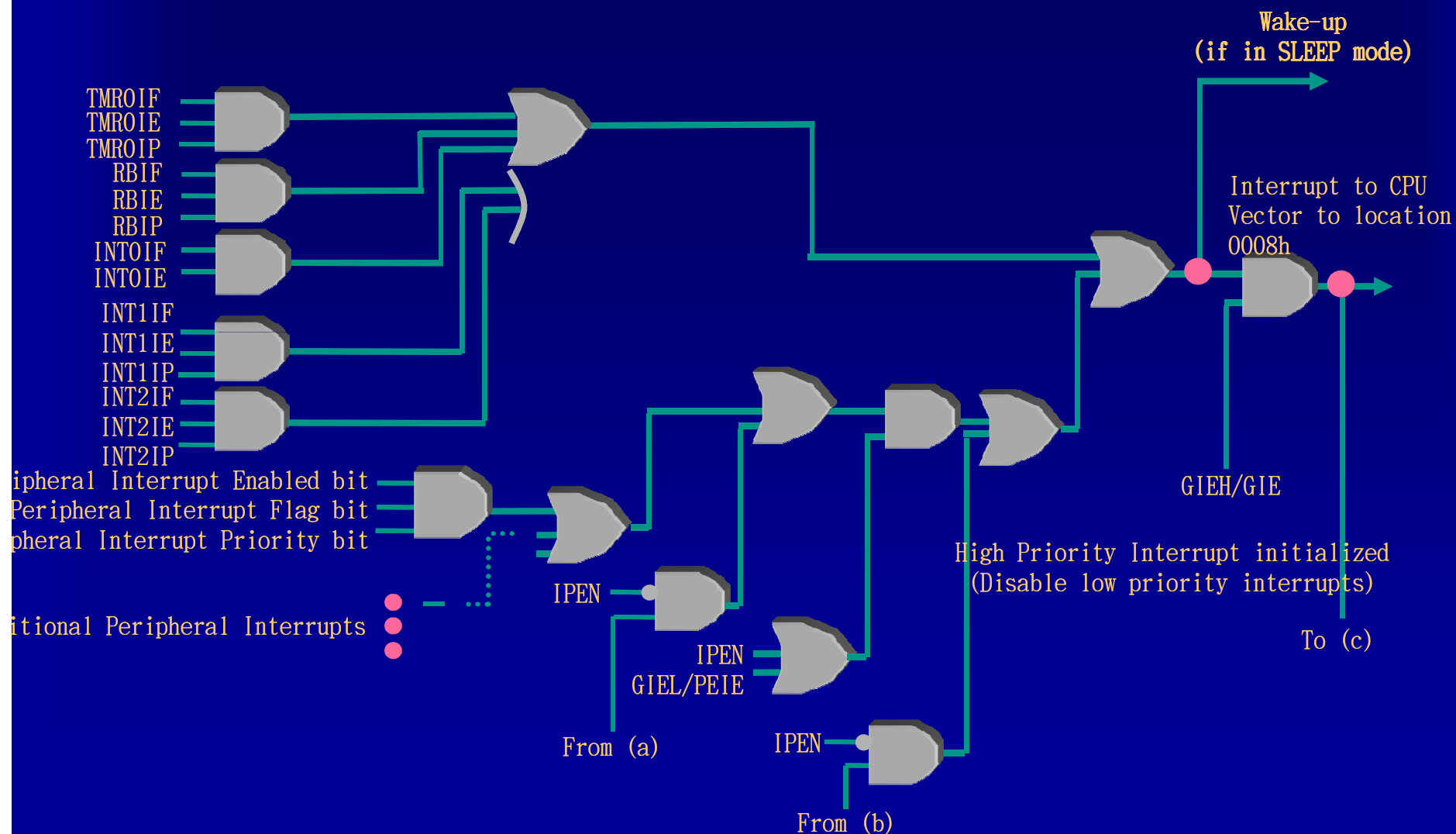
■ 18F452有兩個中斷向量點

- 高優先權 ==> 中斷向量位址 **0x0008**
- 低優先權 ==> 中斷向量位址 **0x0018**
- 每個中斷源均可選擇其中斷優先權（二選一）
- 每個中斷源均有獨立的中斷旗標(**Flag**)
- 中斷旗標的清除 ==> 自行用軟體清除
- 每個中斷源均可 **Enable** 或 **Disable**

■ 當然PIC18系列也可設定與PIC16Fxxx系列的中斷相容 （關掉優先權的設定）



PIC18F452 中斷邏輯架構 (High Priority Level)





Shadow 暫存器

- 18F452有“Shadow Register”的設計，能提高中斷程式對事件的反應速度
- 高優先權中斷
 - W，BSR，STATUS 的存入/取出使用 Shadow Register
 - 程式的返回：retfie FAST
- 低優先權中斷
 - W，BSR，STATUS 的存入/取出則必需透過軟體方式
 - 程式的返回：retfie 0



Timer1 的中斷設定

- 有關 T1CON 的控制位元，請參考 Page 105 (PIC18F452)
- RCON 的 <IPEN> 位元：啓動高、低權位的中斷功能
- GIEH：允許高優先權中斷
- GIEL：允許低優先權中斷
- TMR1IE：打開 Timer1 的中斷設定
- TMR1IP = 1，設定 Timer1 爲高優先權中斷

TABLE 11-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
INTCON	GIE/GIEH	PEIE/GIEI	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP ⁽¹⁾	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

Note 1: The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.



練習二：使用 Timer1 與 中斷

■ 使用 Timer1 外部的 32.768KHz 並啟動 Timer1 的中斷：

- 練習程式：C:\W400 workshop\Exercise\Ex2.asm
- 設定 PIC18F452 的工作模式後，進入主程式的永久迴圈
- 將 Timer1 設定為 500 ms 中斷 CPU 一次 (使用 $F_{osc} / 1$)
 - ✓ $0.5\text{Sec} = \text{Timer Value} * (1 / 32768\text{Hz})$
 - ✓ $\text{Timer Value} = ??$
- Timer1 每次中斷後，必須將 LED (PORTD) 的輸出值加一

注意！ 參考電路圖的設定

JP2 的設定是 3,5 及 4,6 短路； JP9 的設定是 1,2 短路



MICROCHIP

A/D 轉換器 & 電壓比較器

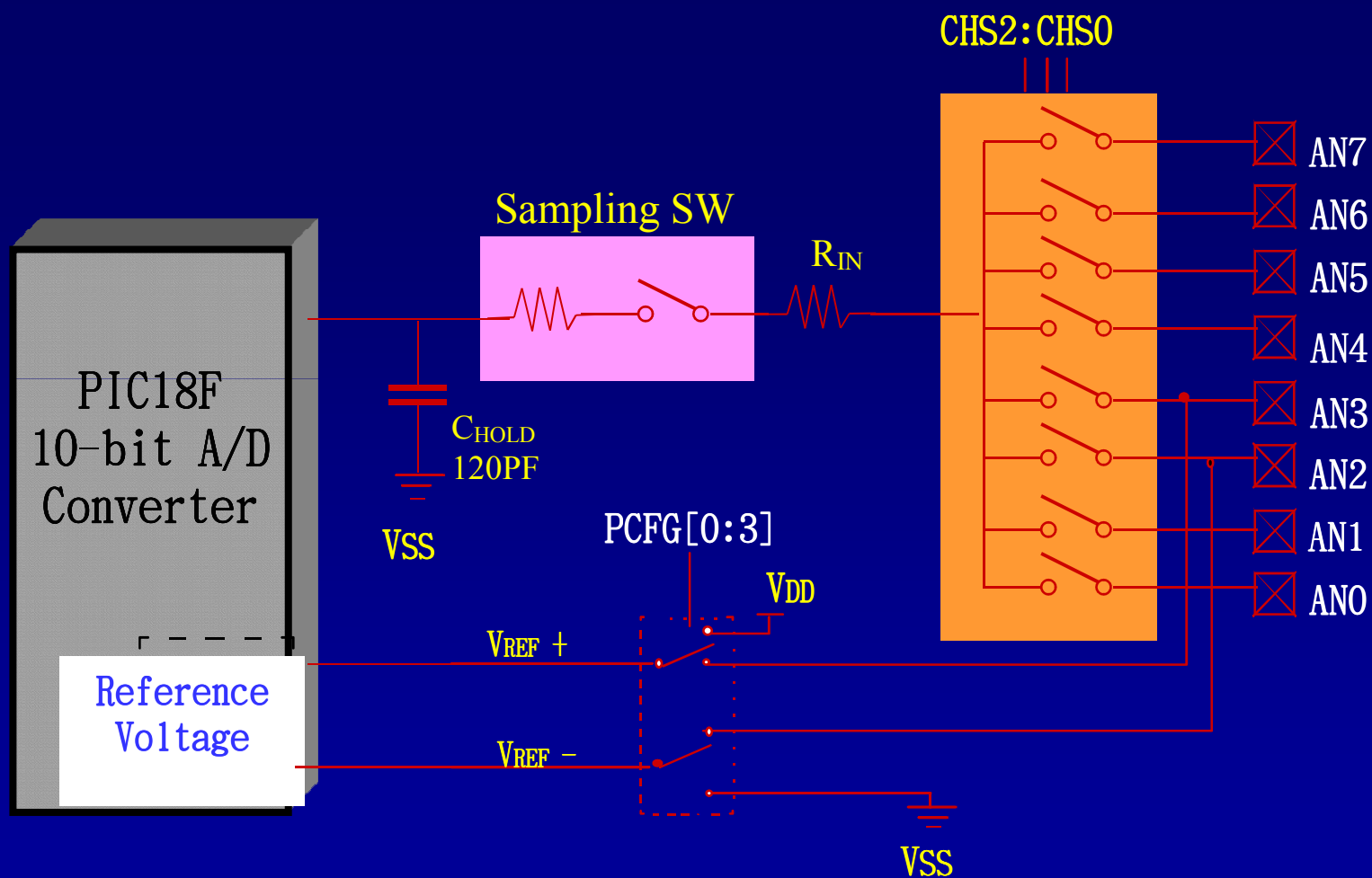


10-bit A/D 轉換器

- 8組類比轉換多工輸入選擇，10 bits 解析度
- 類比輸入取樣時間：20 μ S (輸入阻抗<10K)
- 類比輸入轉換時間：19.2 μ S (12 T_{AD})
- 10-bit 解析度時，只有一位元的誤差
- 允許使用外部參考電壓：VREF+ & VREF-
- 轉換的結果允許自動向左、向右對齊修正
- 完整的轉換時間共須 39.2 μ s
 - 如輸入腳位固定，其轉換時間只需：29.2 μ s



10-bit A/D 方塊圖





See Data Book

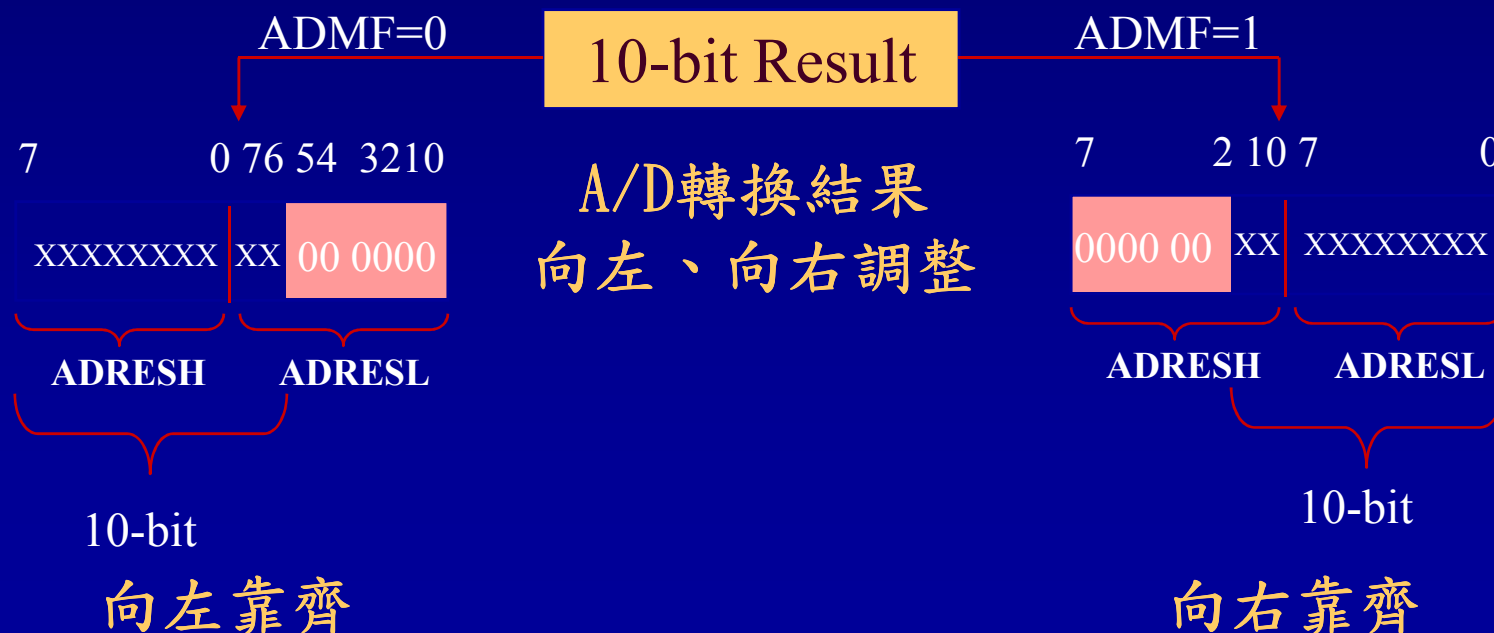
A/D 控制暫存器

ADCON0 Register

ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	---	ADON
bit7							bit0

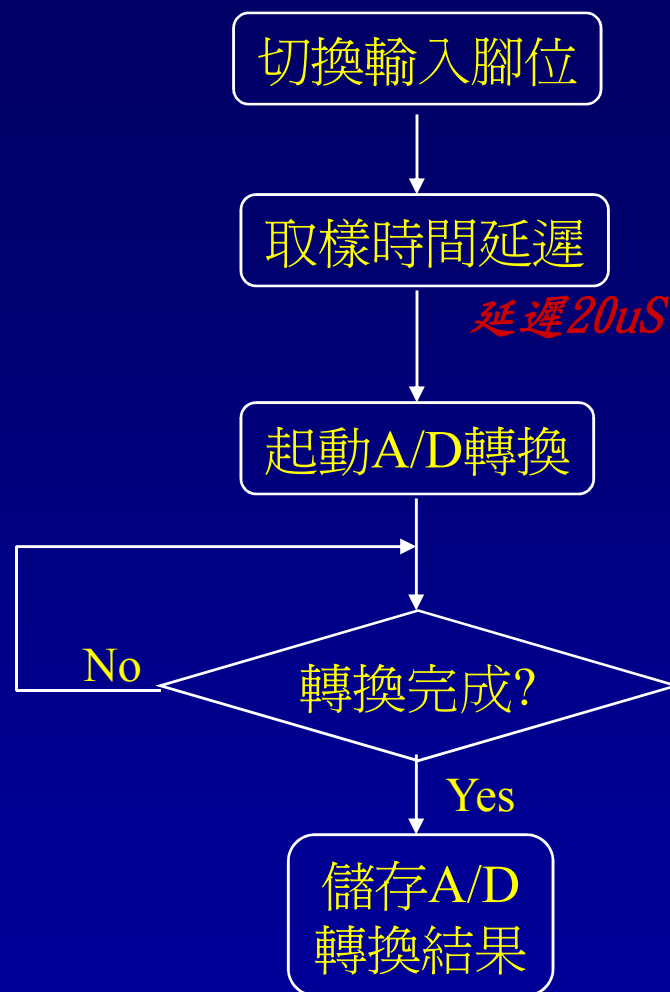
ADCON1 Register

ADFM	ADCS2	----	----	PCFG3	PCFG2	PCFG1	PCFG1
------	-------	------	------	-------	-------	-------	-------





A/D 轉換基本流程

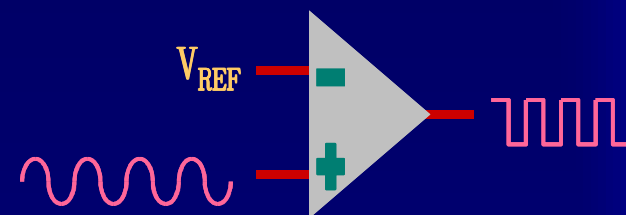




PIC18 系列的周邊

Analog Comparator Module

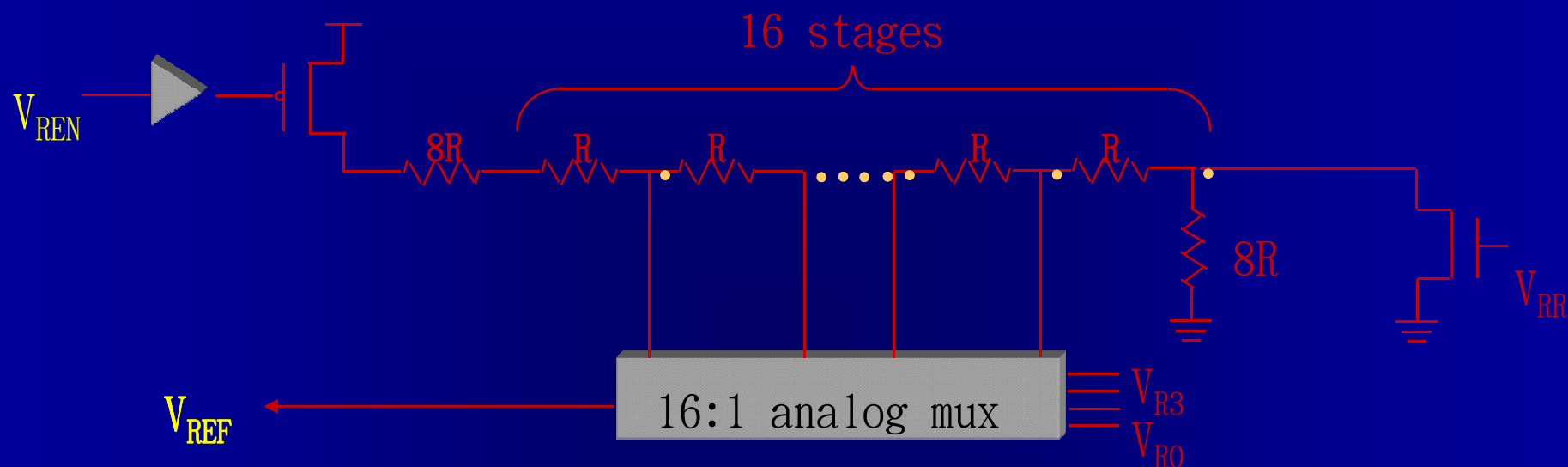
- 兩組類比的比較器
- 內建可程式規劃的參考電壓源
- 8 種工作模式的選擇
- 允許在睡眠模式下工作，比較器的輸出發生改變時可產生中斷來喚醒 MCU
- 比較器的輸出、輸入腳位可規劃成一般 I/O 腳使用





PIC18 系列的周邊

Internal V_{REF} : Block Diagram



- V_{REF} 有 32 段的電壓輸出設定
- 如果關閉參考電壓源，則 V_{REF} 會自動關閉
- 可做為簡易 D/A 轉換器
- 可將參考電壓 (V_{REF}) 可直接輸出於接腳上



練習三：使用 A/D 轉換器

■ 使用 A/D 轉換器將 VR1 的電壓透過 AN0 的輸入腳進行 A/D 的電壓轉換：

- 練習程式：C:\W400 workshop\Exercise\Ex3.asm
- 有關 A/D 的設定，請參考 ADCON0 及 ADCON1 兩個暫存器的設定說明
 - PIC18Fxx2 Data Sheet (Page 179 & 180)
- A/D 轉換的結果，向左調整後只取其最高的 8-bits (最低的兩個位元不予理會)，並將每次轉換後的結果顯示在 LED



MICROCHIP

MSSP & USART



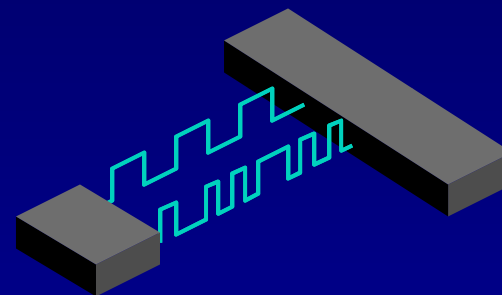
同步串列通訊(一)

Master Synchronous Serial Port (MSSP)

可工作於SPI或 I²C 兩種模式之一

■ SPI (Serial Peripheral Interface) 模式

- 可程式化速率設定
- 最高速率 (@ 40 MHz) :
 - Master 10.0 Mbps
 - Slave 4.29 Mbps
- 可程式化設定接收、發送的時脈(clock)動作極性
- 支援兩種傳輸模式 Microwire™ 和 Motorola's SPI





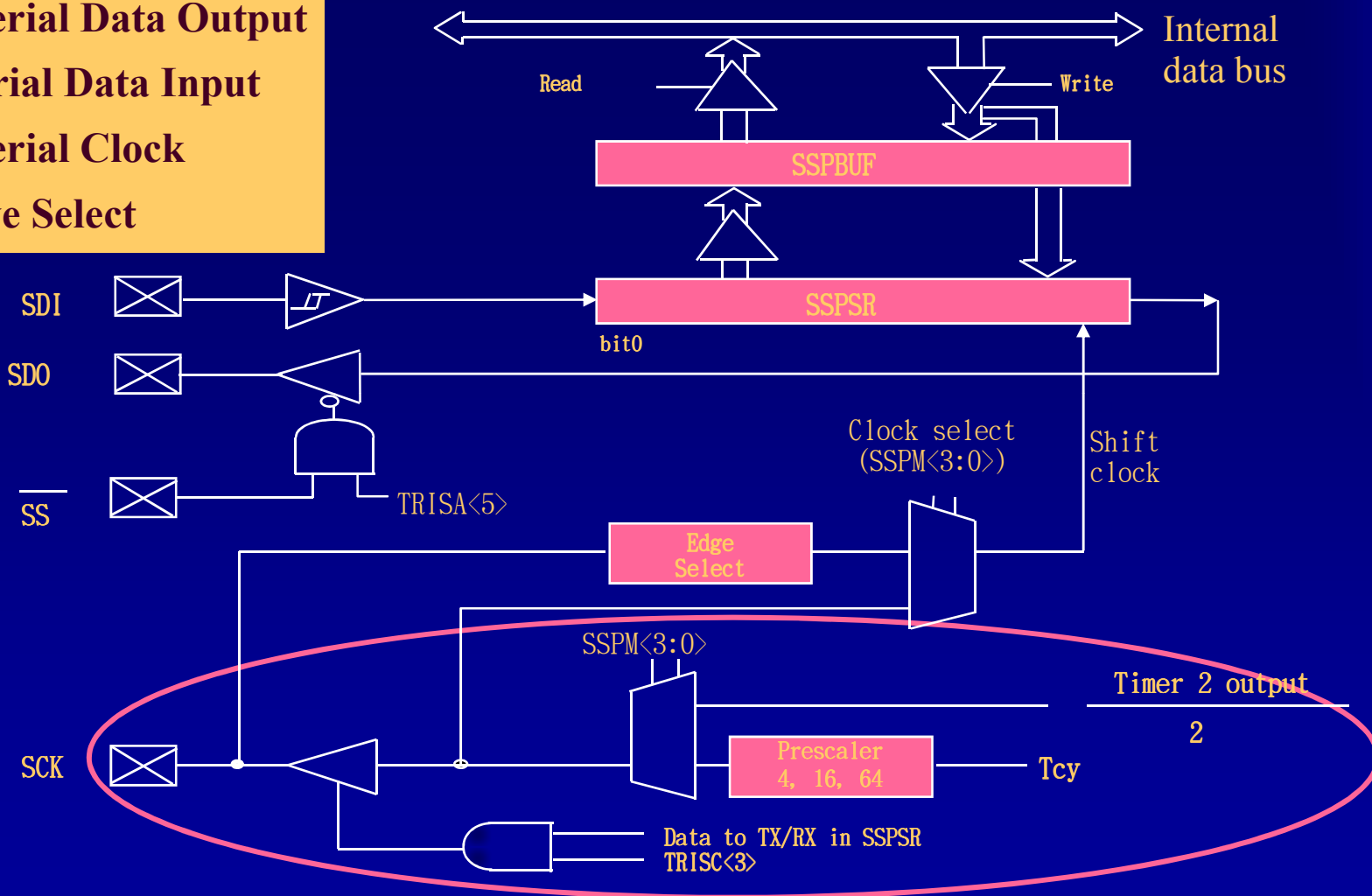
SPI 方塊圖

SDO: Serial Data Output

SDI: Serial Data Input

SCK: Serial Clock

\overline{SS} : Slave Select



Master Mode: SCK is the clock output pin



同步串列通訊(二)

Master Synchronous Serial Port (MSSP)

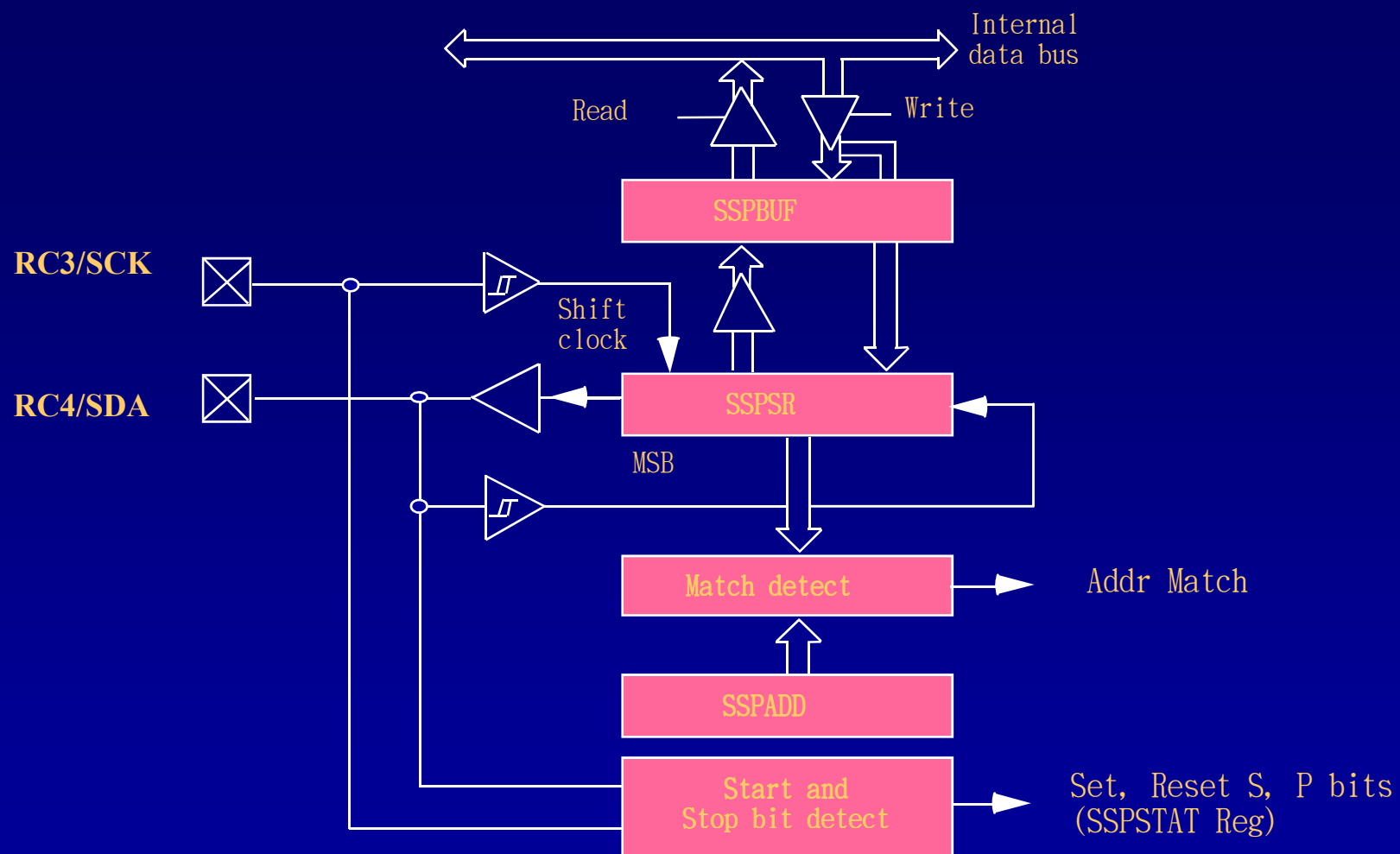
■ I²C (Inter-Integrated Circuit) 模式

- 全硬體化設計，支援下列三種I²C的操作模式
 - **Master Mode**
 - **Multi-master Mode**
 - **Slave Mode**
- 支援 **7-bit** 或 **10-bit** 位址模式
- 傳送及接收速度:**100kHz**， **400kHz**， **1 MHz**
- 支援通用位址 (**General call**) 模式 “**address=0x00**”



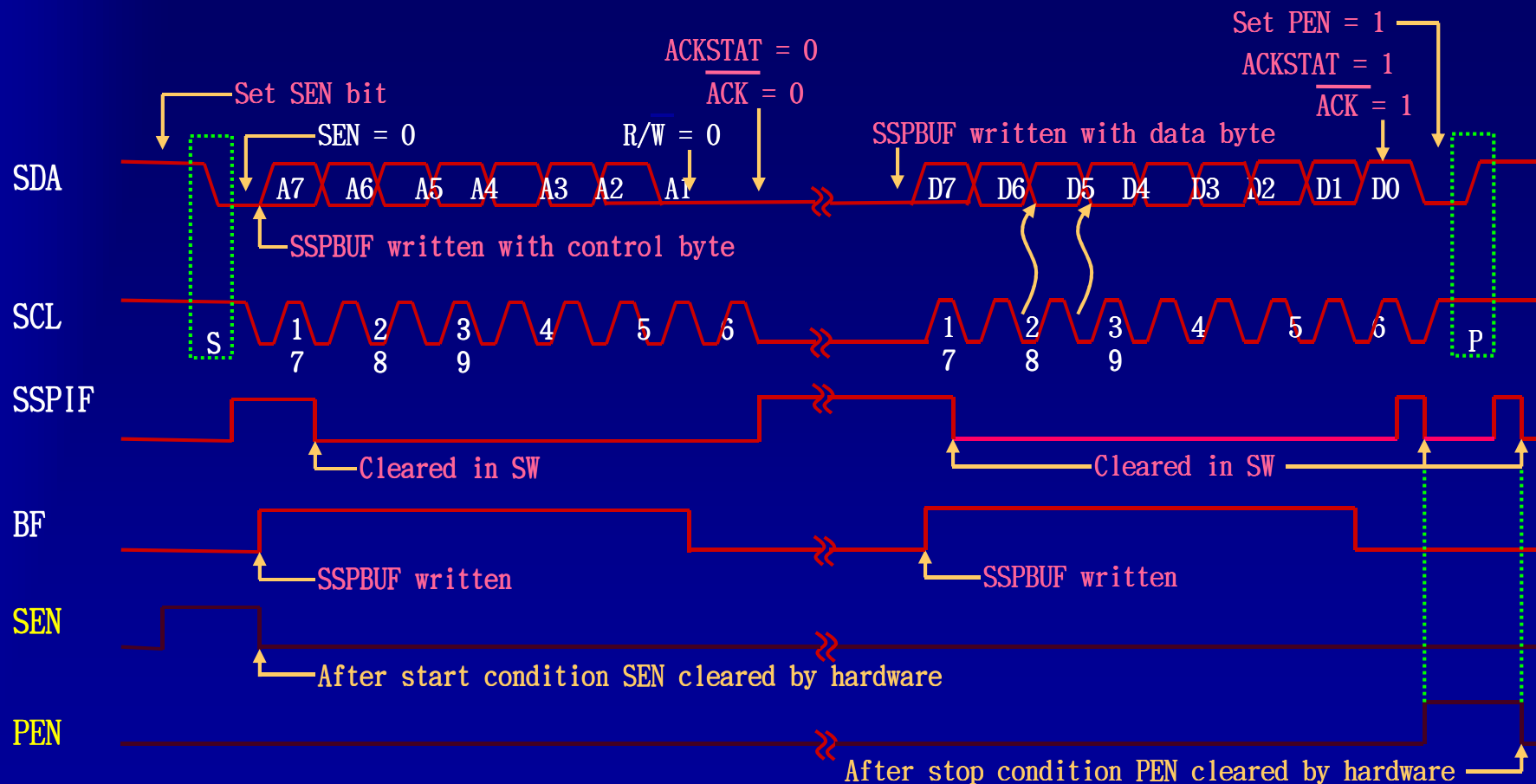


I²C Slave 模式方塊圖





I²C Master 模式下 發送資料的時序

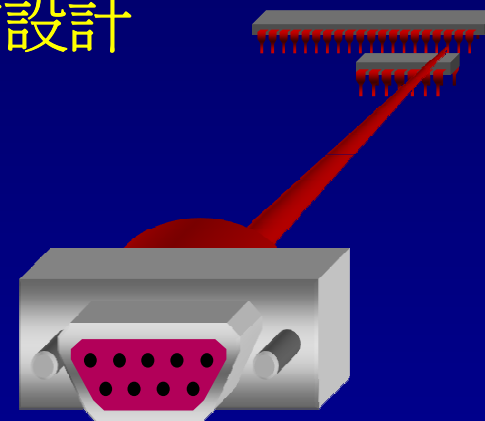




PIC18 系列的周邊

Addressable USART (AUSART)

- 支援全雙工非同步串列傳輸，半雙工同步串列傳輸
- 8-bit 或 9-bit 的資料格式
- 串列接收與發送採用雙資料緩衝器的設計
- 獨立的收、發中斷產生
- 資料的接收、傳送以 **b0** 為第一位元
- 獨立的速率產生器
- 在 20MHz 時的最高速率：
 - 同步串列傳輸模式：5 Mbaud
 - 非同步串列傳輸模式：312.5 Kbps (低速模式)
1.25 Mbps (高速模式)
- 支援 9-bit 位址或資料的判斷模式

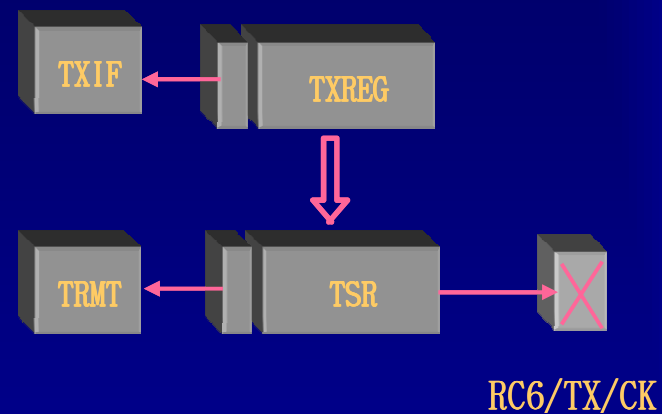






TXIF & TRMT 的動作

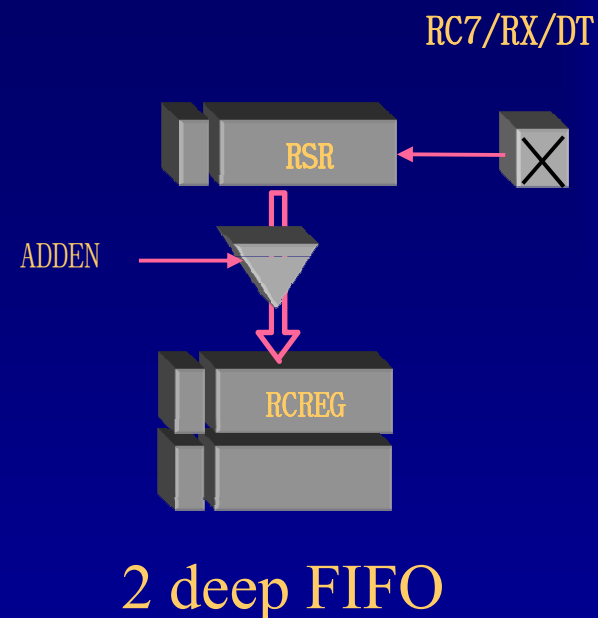
- 假如TXREG的資料被載到TSR，TXREG會空出;則TXIF = 1
- 假如TSR的資料串列傳送完畢;則TRMT = 1
- 假設TXREG剛載入資料時TRMT 為空的(TRMT=1)，則這筆資料會立即被送到TSR，串列傳送會動作，同時TXIF = 1
- TXIF是可單獨使用，即使USART 的TX中斷是關閉的(TXIE=0)
- 由以上動作可知偵測發送狀態TXIF會比TRMT來的快





RCIF的動作

- **RSR** 為一個移位器，接收**8-bits**的串列資料及 **start & stop bit**.
- 接收到的串列資料會被載入到 **RCREG FIFO** 並將設定 **RCIF**
- 假設上一筆資料在 **FIFO** 中未被拿走此時又有一筆串列資料接收近來，則這筆新的資料會被存在第二級的 **FIFO** 中
- 若兩級 **FIFO** 均有資料，接收中斷產生將第一級資料提走，中斷處理完畢後，第二級 **FIFO** 的資料會立即產生中斷





PIC18 系列的周邊

RS-485 的功能說明

- **RS-485** 適用於高雜訊、長距離通訊，具有多個接收並聯的功能：
 - 採用差動式雙絞線傳送，標準**UART**的傳輸格式
- 半雙工模式只需一組對絞線（多點通訊）
- 全雙工模式需兩組對絞線（點對點通訊）
- 利用第九個位元(**b8**)作為資料或命令的指引
 - 當**b8**為1時，表示該筆資料為命令，啟動位址判斷；只有被呼叫的接收器才會被允許接收下一筆的資料
 - **b8=0**時，該筆資料為一般的資料



PIC18 系列的周邊

RS-485 的功能說明

- **ADDEN** 位元通知 **USART** 檢查串列資料的第九位元是否為零，若為零則忽略此次的資料：
 - 此次接收無效，勿須傳送至接收緩衝器中
 - 不產生中斷

- **注意！** 發送器的程式必須有效地控制第九位元



USART 速率的計算

■ PIC18F452 可設定 UART 的傳輸速率

- $BRGH = 0$, 低速模式, $\text{Baud Rate} = F_{\text{osc}} / [64 (x + 1)]$
- $BRGH = 1$, 高速模式, $\text{Baud Rate} = F_{\text{osc}} / [16 (x + 1)]$

■ 以 16MHz 的工作頻率為例：欲求出 9600 bps 的速率

- $\text{Baud Rate} = F_{\text{osc}} / [64 (x + 1)]$
- $9600 = 16\text{MHz} / [64 (x + 1)]$
- $x = [(16\text{MHz} / 9600) / 64] - 1$
- $x = 25.025$
- $\text{SPBRG} = 25$



USART 相關的暫存器

- 傳送資料時可不必使用中斷傳送
- **BRGH = 0**
- 傳送格式設為：
9600，N，8，1

TABLE 16-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP ⁽¹⁾	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

TABLE 16-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP ⁽¹⁾	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

- 接收最好使用中斷方式接收
- **CREN = 1**
- **ADDEN = 0**
- 接收格式設為：
9600，N，8，1



練習四：使用 USART

- 練習程式：C:\W400 workshop\Exercise\Ex4.asm
- 設定 USART 的工作模式為：9600，N，8，1
- 電腦的超級終端機設定使用 COM1 或 COM2，通訊協定為：
 - 9600，N，8，1，Handshake：None
- 傳送資料：不採用中斷方式
- 接收資料：採用中斷即時接收方式
- APP001 電路設定需將 J6 的 1&2 及 3&4 短路
- 連接 15-Pin 的 RS-232 Cable 到 PC 端

- 程式會每隔 1 秒 (Timer1) 將 A/D 的轉換結果 (10-bit) 轉成 ASCII 碼後，傳送到超級終端機上顯示 (每次傳送自動加入換行字元)
- PC 鍵盤按 “P” 鍵，18F452 停止傳送 A/D 值，按 “C” 鍵則繼續傳送



MICROCHIP

PIC18FXXX

**Directive Instructions
Internal Data EEPROM
Table Read / Write**



常用的虛擬指令 -- #DEFINE

#DEFINE *<name>* *<string>*

- 通常用來定義文字、標記、常數
- 在此定義中，*<string>* 所描述的文字用 *<name>* 為助憶文字，在組合語言中可以用 *<name>* 來代替 *<string>* 以增加程式的閱讀性

例如:

```
#DEFINE  S_CLK      PORTC, 3
#DEFINE  S_DATA     PORTC, 2
#DEFINE  S_FLAG     0x30, 7
```

```
list p=16f74
#include  <p16f74.inc>
;
val_ff   equ      0xf8
val_250  equ      .80
count    equ      0x20
;
#define   m_flag   count, 6
#define   clk      PORTC, 0
#define   sda      PORTC, 1
:
:
bsf      clk
btfss    sda
goto     sda_low
call     sda_hi
```



常用的虛擬指令 -- CBLOCK & ENDC

- **CBLOCK** 宣告變數的區域段，並設定該區域段在暫存器的起始位址
- **CBLOCK** 所定義的第一個變數 (暫存器) 位址，即是該起始位址，下一個變數位址會自動加一。
- **ENDC** 是結束此變數區域段的宣告。

		00016	cblock h'20'
		00017	count, count_ms, led_count
		00018	led_index
		00019	led_buff
		00020	endc
		:	
0042	3008	00081	movlw 0x8
0043	00A2	00082	movwf led_count
0044	0DA4	00083	rlf led_buff, f
0045	0C86	00084	rrf PORTB, f
0046	2054	00085	call delay_250ms
0047	0BA2	00086	decfsz led_count, f
0048	2844	00087	goto loop_right
0049	283C	00088	goto loop_index



常用的虛擬指令 -- MACRO & ENDM

- **MACRO**一般稱之為巨集函數，是用來制定自己的專用的指令，並可傳遞參數給巨集函數內的指令

User_instru **MACRO** **var_in1, var_in2, var_out**

使用者定義的指令名稱

參數及變數傳遞

- 在巨集函數內，如需執行 **GOTO** 指令可用 “\$” 符號做為跳躍的基準位址做加(往下跳)或減(往上跳)的短程跳躍。

swap_led

MACRO

```
movlw    h'8'  
movwf    led_count  
rrf       led_right,f  
rlf       led_lift,f  
decfsz    led_count  
goto     $-6  
ENDM
```

- **ENDM** 是用來結束本巨集函數。
- 有 **MACRO** 的宣告就必須有 **ENDM** 的存在。



MACRO & ENDM 的使用範例

```
portc_bfr equ 0x50
w_buffer equ 0x51
s_buffer equ 0x52
;
PUSH MACRO
    movwf w_buffer
    swapf w_buffer,F
    swapf STATUS,W
    movwf s_buffer
ENDM

;
POP MACRO
    swapf s_buffer,W
    movwf STATUS
    swapf w_buffer,W
ENDM

;
```

```
MOV MACRO regd,regs
    movf regs,w
    movwf regd
ENDM

;*****
; Program start
;*****
;
    org 0x0
    goto start
;
    org 0x04 ; Interrupt
              vector
;
PUSH
MOV portc_bfr,PORTC
:
POP
retfie
```



常用的虛擬指令 -- DB

標記 **DB** <字元>, <字串>

- 產生 **8-bit** 的固定資料存放在程式記憶體
- 此固定資料會以 **Intel's HEX Code** 型態與程式碼結合
- 因 **PIC18Fxxxx** 是**16-bit** 的程式碼型態，若資料長度為奇數則該字元組的偶數位元組 (較高的**Byte**) 會自動補零
- 固定資料型態可為： **Hex code** 、 **ASCII** 字元、字串

```
0001DE 5B1B 4A32 0000    String_0 db    0x1b, '[', '2', 'J', 0x00    ; Clear Terminal Screen
0001E4 2020 2020 2020    String_1 db    "                    Microchip Technology Taiwan ; Workshop
                         400", 0x0a, 0x0d, 0x00
                         2020 4D20 6369
                         6F72 6863 7069
                         5420 6365 6E68
                         6C6F 676F 2079
                         6154 7769 6E61
                         3B20 5720 726F
                         736B 6F68 2070
                         3034 0A30 000D
```



內部資料 EEPROM

PIC18F452

- **256-Bytes EEPROM**
- 具有一百萬次的寫入或清除能力
 - **Flash Program Memory** 具有十萬次的寫入或清除能力
- 資料持久年限：超過 **40** 年
- 具備 **EEPROM** 寫入完成的中斷功能
- **EEPROM** 具有全電壓範圍的燒錄
 - **PIC18LF452 : 2.0V ~5.5V**
- **EEPROM** 寫入時間：**4mS**



EEPROM 相關暫存器

PIC18F452

- **EEDATA (0xFA8)**
 - 用以存放 **EEPROM** 寫入或讀取的資料
- **EEADR (0xFA9)**
 - 用以存放 **EEPROM** 的欲存取的地址 (0x00 ~ 0xFF)
- **EECON1 (0xFA6)**
 - **EEPROM** 讀 / 寫控制暫存器
- **EECON2 (0xFA7)**
 - 特殊的物理上不存在的暫存器
 - 讀出的結果全為 '0'
 - 專用于 **EEPROM** 的寫入順序控制的操作



EECON1 寄存器

EEPGD	CFGS	—	FREE	WRERR	WREN	WR
bit7						bit0

- Bit7 : EEGPD 1 = 存取 Flash Program
 0 = 存取 EEPROM
- Bit6 : CFGS 1 = 存取 Configuration
 0 = 存取 Flash Program 或 EEPROM
- Bit4 : FREE Flash memory Row Erase Enable
- Bit3 : WRERR Write operation Error Flag
- Bit2 : WREN Write Enable
- Bit1 : WR Write Control Bit
- Bit0 : RD Read Control Bit



讀取 EEPROM 的資料

PIC18F452

- 把欲讀取 **EEPROM** 的位址寫到 **EEADR** 暫存器
- 清除 **EEPGD** 位元，**EECON1<7>**
- 清除 **CFGSS** 位元，**EECON1<6>**
- 設定 **RD** 位元，**EECON1<0>**
- 等過了兩個指令周期的時間後，資料會在 **EEDATA** 暫存器中就緒



讀取 EEPROM 資料的程式

READ_EEPROM:

```
    movff    Data_EE_Addr, EEADR
;
    bcf      EECON1, EEPGD
    bcf      EECON1, CFGS
    bsf      EECON1, RD
    movf     EEDATA, W
;
    return
```

EEPROM 讀取副程式

將欲讀取的位址存入 **EEADR**

清除 **EEPGD** 位元

清除 **CFGS** 位元

啟動讀取 **EEPROM** 的動作

讀取動作完成，將資料存到 **W Reg.**

副程式返回



寫入資料到 EEPROM

PIC18F452

- 把欲寫入 **EEPROM** 的位址寫到 **EEADR** 暫存器
- 把欲寫入 **EEPROM** 的資料寫到 **EEDATA** 暫存器
- 清除 **EEPGD** 位元，**EECON1<7>**
- 清除 **CFGFS** 位元，**EECON1<6>**
- 設定 **WREN** 位元，**EECON1<2>**
- 關閉所有的中斷
- 寫 **55h** 到 **EECON2** 暫存器
- 寫 **AAh** 到 **EECON2** 暫存器
- 設定 **WR** (**EECON1<1>**)，以啟動寫入的動作
- 允許中斷
- 清除 **WREN** 位元，**EECON1<1>**



寫資料到 EEPROM PIC18F452

EEPROM 寫入副程式

設定欲寫入 **EEPROM** 的位址與資料

清除 **EEPGD** 位元

清除 **CFGS** 位元

設定 **WREN** 位元，啟動寫入功能

(關閉所有的中斷，以避免程序錯亂)

啟動寫入順序步驟

開始寫入 **EEPROM**

(允許中斷功能)

測試是否寫入完成？

寫入完成，關閉 **EEPROM** 寫入功能

Write_EE_Data:

```
movff Data_EE_Addr, EEADR
movff Data_EE_Data, EEDATA
;
BCF EECON1, EEPGD
BCF EECON1, CFGS
BSF EECON1, WREN
;
;
BCF INTCON, GIE
MOVLW 0X55
MOVWF EECON2
MOVLW 0XAA
MOVWF EECON2
BSF EECON1, WR
BSF INTCON, GIE
;
LOOP BTFSS PIR2, EEIF
      GOTO LOOP
;
BCF EECON1, WREN
BCF PIR2, EEIF
RETURN
```



讀取程式記憶體 PIC18F452

- **Table Read** 指令讀取 **Program Memory** 中的資料至 **Data Memory** 中
 - 程式查表, PIC16Cxx 系列並無此功能
 - PIC16F87x, 16F7x, 16C78x 則有 **Table Latch** 讀取功能
- **Program memory** 以 **Byte** (位元組) 為單位, 以 **22 bit** 的 **Table pointer** 來指定其位址 (**TABPTRU:TABPTRH:TABPTRL**)
- 被讀入的資料將置於 **TABLAT** 暫存器中
- **PIC18Fxxxx** 提供一共四種型態的 **Table Read** 指令



PIC18F452 的 Table Read 指令

■ 四種 TABLE Read 的操作方式

- TBLRD* : 直接讀取指標位址所指的內容
- TBLRD*+ : 讀取指標位址所指的內容後將指標加一
- TBLRD*- : 讀取指標位址所指的內容後將指標減一
- TBLRD+* : 將指標先加一後再讀取指標位址所指的內容



程式記憶體의讀取範例

PIC18F452

```
LFSR      FSR0, RAMBUFADDR      ; 設定 22-bit 的程式位址
MOVLW     UPPER (PROGMEMADDR)   ;
MOVWF     TBLPTRU                ;
MOVLW     HIGH (PROGMEMADDR)    ;
MOVWF     TBLPTRH                ;
MOVLW     LOW (PROGMEMADDR)     ;
MOVWF     TBLPTRL

;
TBLRD*+   ; 讀取目前位址的程式資料，並將指標加一
;
MOVFF     TABLAT, POSTINC0       ; 將資料取出存到 FSR0 所指到的 RAM
; 的位址後，FSR0加一只到下一位址
```



寫入程式記憶體

PIC18F452

- **Table Write** 指令允許資料由 **Data Memory** 寫入 **Program Memory**
- **Program memory** 以 **Byte** (位元組) 為單位，以 **22 bit** 的 **Table pointer** 來指定其位址
 - **TABPTRU : TABPTRH : TABPTLL**
- **TABLAT** 暫存器存放欲寫入 **Program Memory** 的資料
- **PIC18Fxxxx** 提供四種型態的 **Table Write** 指令



PIC18Fxxx 的 Table Write 指令

■ 四種 TABLE Write 的操作方式

- TBLWT* : 直接寫入資料到指標所指的位址
- TBLWT*+ : 寫入資料到指標所指的位址後將指標加一
- TBLWT*- : 寫入資料到指標所指的位址後將指標減一
- TBLWT+* : 將指標先加一後再將資料寫入指標所指位址



程式記憶體의寫入範例

PIC18F452

```
: ; 將欲被用來結束 Long Write 程
: ; 序的中斷來源設定好
LFSR    FSR0, RAMBUFADDR ; 將要寫入的位址寫入 22 Bits
MOVLW   UPPER (PROGMEMADDR) ; 的 Table Pointer 中
MOVWF   TBLPTRU ; TBLPTRU, TBLPTRH, TBLPTRL
MOVLW   HIGH (PROGMEMADDR) ;
MOVWF   TBLPTRH ;
MOVLW   LOW (PROGMEMADDR) ;
MOVWF   TBLPTRL ;
MOVFF   POSTINC0, TABLAT ; 將要寫入程式記憶體的值寫入
; TABLAT 暫存器
TBLWT*+ ; 寫入至 holding register
MOVFF   POSTINC0, TABLAT ; 載入第二個 byte
TBLWT*+ ; 寫入至 MSB (odd address)
; 起始 long write 的程序
```




練習五：Table Read

- 練習程式：C:\W400 workshop\Exercise\Ex5.asm
- 將存在 **ROM** 的字串資料用**Table Read** 的方式讀出後，透過 **RS-232** 傳送到超級終端機顯示
- 通訊協定：**9600 , N , 8 , 1 , None Handshake**

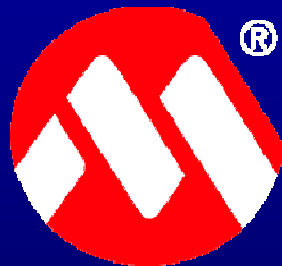
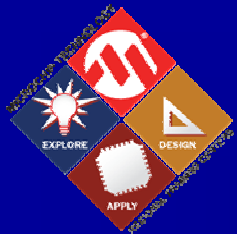
- 程式會每隔 1 秒 (Timer1) 先傳送字串 “The A/D Results are : “ 到終端機顯示後，再將 A/D 的轉換結果 (10-bit) 轉成 ASCII 碼後，傳送到終端機上顯示 (每次傳送自動加入換行字元)
- PC 鍵盤按 “P” 鍵後，停止傳送 A/D 值，並將字串 “The A/D is stopped conversion !” 傳到終端機顯示
- PC 鍵盤按 “C” 鍵則繼續傳送 A/D 值，並將字串 “The A/D is running !” 傳到終端機顯示



MICROCHIP

附錄 A

開發工具介紹



MICROCHIP

開發工具介紹

APP001 多功能實驗版

MPLAB IDE v8.xx

MPLAB ICD2, ICE2000

MPLAB Real ICE



APP001 v1.0 實驗板





APP001 實驗板功能

- 練習 PIC16F877A 內部周邊的使用方法及典型的應用方式
- 40 Pin 的 PIC16Fxxx與 PIC18Fxxx 腳位相容
- 直接與 MPLAB-ICD2 連接, 不須經由 Header Board 轉接
- 可練習的主要外接電路有:
 - LCD Module, 按鍵處理, I²C EEPROM, I²C溫度存取
 - SPI, RS-232, RS-485, CAN Controller
 - AD 轉換, PWM 與 I/O 控制
- 101ASP 課程使用的主要資源
 - PORTD 用以控制 8 個獨立的 LED
 - RA0 : 使用於讀取可變電阻的電位值 (AN0)
 - RA4 : 使用於按鍵控制 (SW3)



MPLAB-IDE 功能介紹

- 高整合度的微處理器軟體 / 硬體研發及偵錯平台
- 採用專案管理模式 (**Project**)
- 多視窗原始程式編輯、修改
- 直接組譯 / 編譯原始程式
- 軟體模擬
- 具有輸入模擬功能
- 支援原始程式偵錯
- 支援 **MPASM**、**MPLINK**
- 支援 **C compiler**

- 硬體除錯工具：

- **ICE2000**
- **Real ICE**
- **MPLAB ICD2**
- **MPLAB ICD2 LE**
- **PICKit2 Debugger**

- 程式燒錄工具：

- **PM-III**
- **PICSTART Plus**
- **MPLAB ICD2**
- **PICKIT2 Debugger**



MPLAB IDE - 整合式的發展環境

MPLAB®

Integrated Development Environment

內含多功能
程式編輯器

單一系統專案
管理模式

原始檔案程式
偵錯功能

語言工具

**MPASM™
Assembler**

**MPLINK™
MPLIB™**

**MPLAB C18
MPLAB C30
MPLAB C32**

軟體模擬

**MPLAB SIM
Software
Simulator**

模擬器及除錯器

**MPLAB
REAL ICE**

**MPLAB ICE
2000 & 4000**

**PICKit 2
MPLAB ICD 2**

燒錄器

**PICKit 1
PICSTART®
Plus**

MPLAB PM3

協力廠商
支援工具

**Compilers
IAR, Hi-Tech,
CCS,
ME Labs,
Green Hills**

**Real-time Operating
Systems
CMX, Vector,
Realogy, Express
Logic**

**MATLAB
Live Devices, CMX,
Momentum Data
Systems**

Uniquely supporting 8, 16 and 32 bit MCUS within one integrated development Environment!



MPLAB - ICE2000

連接 Host 到 Pod 的 Cable

***Emulator Pod**

***Processor Module**

Flex Circuit Cable

***Device Adapter**

***SOIC Transition Socket**

* 每個元件可分開採購



MPLAB – ICE2000

- 支援所有 **PICmicro[®]** MCU 系列的模擬
- 全速的模擬支援
 - **Up to 25 MHz (PIC18CXXX)**
 - 將有支援 **40 Mhz** 以上的版本
- 支援低操作電壓環境的模擬
 - 最低至 **2.5 V**
- 與 PC 使用 **Parallel printer port** 的界面
- **Software programmable clock** (在MPLAB-IDE 中可直接設定所需頻率)



MPLAB – ICE2000

- 改良甚多的 **Trace** 能力
 - 可將中斷點設置於 **internal registers/RAM**
 - 可以 **Trace internal registers/RAM**
 - **32K * 128 bits Trace Buffer** 以及 **Logic analyzer**
 - 可記錄執行階段的時間標記 (**Time Stamp**)
- **Four level conditional break/trace/trigger**
- 可精確量測兩事件間的時間間隔
(**Time between intervals**)



MPLAB ICD 2

- 支援 **FS-USB** 介面
- 支援 **PIC16F** 裡具有 **ICD** 功能之 **MCU** 的除錯
- 支援 **PIC18F**，**PIC24**，**dsPIC30**，**dsPIC33** & **PIC32**
- 可設定中斷點，單步執行以及進行變數的觀察
- **32K Bytes (PIC18F4520)** 的程式可於 **3 秒**內燒錄完成





Real ICE 基本配備

ICE 主機

USB 插座

Logic Probe

ICSP Cable

USB Cable

ICSP 連接板

MPLAB[®] REAL ICE[™] In-Circuit Emulator



傳統 ICE 技術上的瓶頸

- ICE 專用晶片價格昂貴
 - 量少，難以大量生產
 - 需與量產元件同步上市，困難度高
- 信號傳輸上時序與速度的誤差
 - CPU/記憶體 超過 20MHz 時的存取限制
 - ICE 實際的線路距離與元件之間的規格差異
- ICE 價格
 - 速度越快，價格越高
 - 價格昂貴，需要高速的外接硬體邏輯電路、FPGA & SRAM



REAL ICE 功能

■ “基本” 功能

- 完全整合在 **MPLAB IDE** 的環境下
- 開發環境下可作為燒錄器
- 中斷功能
- 監看變數視窗功能
- 程式執行控制

- Run
- Halt
- Single step

- Step into
- Step over
- Reset



REAL ICE 進階功能

■ Advanced Features

- **Breakpoints** (軟體與硬體中斷設定)
- **Stopwatch** (程式執行計時碼表)
- **Real time watch** (即時變數觀測)
- **Trace** (追蹤)
 - **Program execution** (追蹤程式執行)
 - **Variable values** (追蹤變數值)



REAL ICE 功能一覽表

	PIC10F PIC12F PIC16F	PIC18F	dsPIC30F	PIC24F	PIC24H dsPIC33F
Run, Halt	√	√	√	√	√
Single Step	√	√	√	√	√
Hardware Breakpoints	1	1-3	1-4	1-6	1-6
Peripheral Freeze on Halt	√	√	√	√	√
Break on Data Fetch or Write		(√)	√	√	√
Break on Stack Error		√	-	-	-
Stopwatch		(√)	√	√	√
Pass Counter		√	√	√	√
Break address or data match		√	√	√	√
WDT Overflow		(√)	√	√	√
Real Time Watch		(√)	√	√	√
Software Breakpoints		1K	1K	1K	1K
Trace Data and Program Flow		√	√	√	√



APP001 實驗板功能

- 練習 **PIC18F452** 內部周邊的使用方法及典型的應用方式
- **40 Pin** 的 **PIC16Fxxx**與 **PIC18Fxxx** 腳位相容
- 直接與 **MPLAB-ICD2** 連接
- 可練習的主要外接電路有：
 - **LCD Module** , 按鍵處理 , **I²C EEPROM** , **I²C**溫度存取
 - **SPI** , **RS-232** , **RS-485** , **CAN Controller**
 - **AD 轉換** , **PWM** 與 **I/O** 控制
- **WORKSHOP 400** 使用的主要資源
 - **PORTD** 用以控制 **8** 個獨立的 **LED**
 - **RA0** : 使用於讀取可變電阻的電位值 (**AN0**)
 - **RA4 & RB0** : 使用於按鍵控制 (**SW3 & SW2**)



MICROCHIP

使用 MPLAB IDE v8.xx



安裝 MPLAB IDE V8.xx

■ 安裝 MPLAB IDE v8.xx

- 之前已有安裝舊版的 MPLAB IDE，需先移除
- 別忘了，還要安裝 ICD2 的 USB 驅動軟體
 - C:\Program Files\Microchip\MPLAB IDE\ICD2\Drivers\clnicd2.htm 裡的說明
- 如果有使用 ICE2000 還要再安裝驅動程式
 - C:\Program Files\Microchip\MPLAB IDE\ICE 2000\Drivers\DDINSTXP.HTM 裡的說明

MPLAB IDE 畫面

工具圖示區

工作項目

C 的原始程式

設定中斷點

編譯輸出

暫存器顯示

變數觀察視窗

狀態顯示列



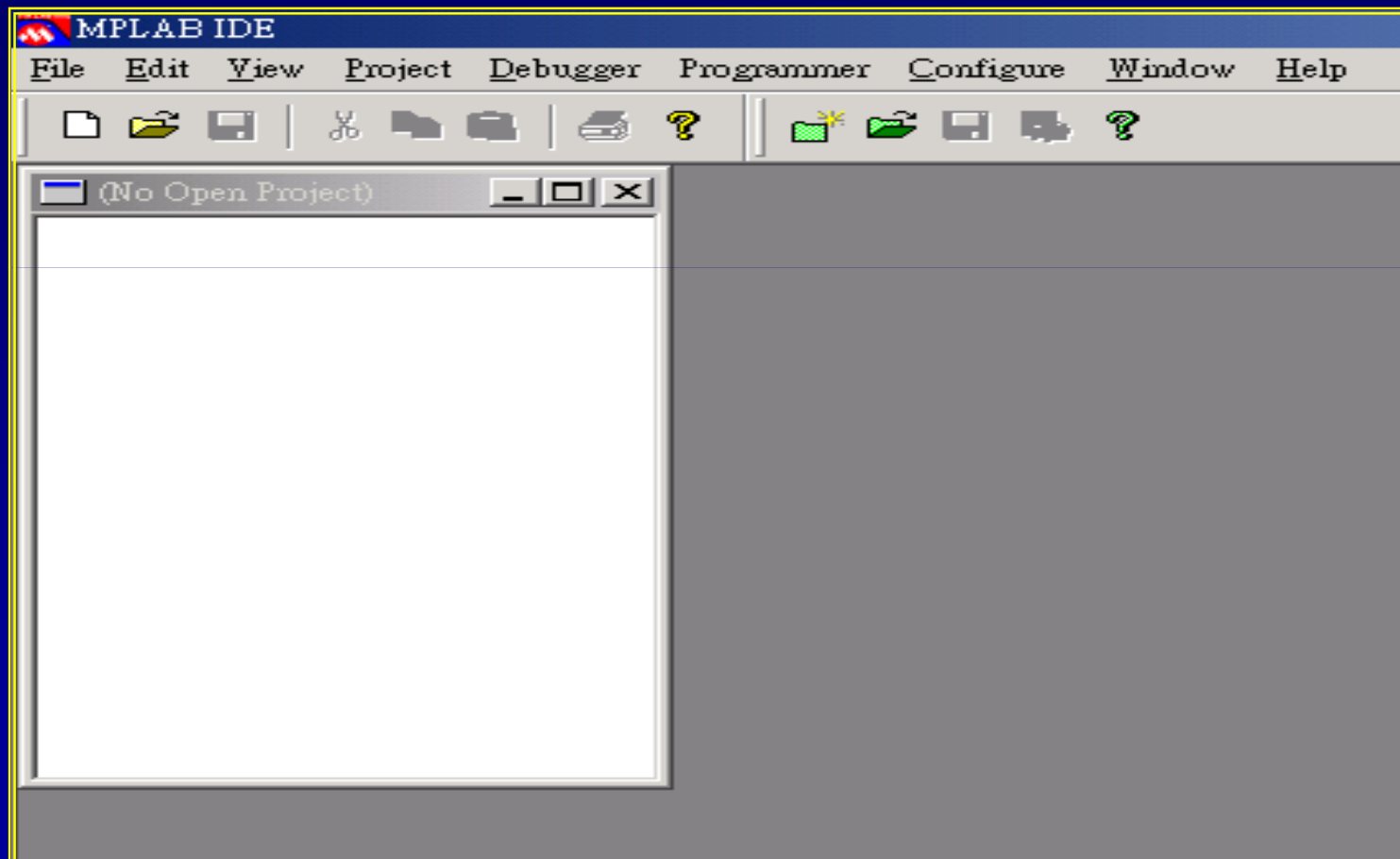
專案管理 (Project Management)

- **MPLAB IDE** 是採專案管理方式來完成軟體研發與設計，所以在使用 **MPLAB IDE** 時，就必需建立一個 **Project**。
- **Project** 的附加檔案名稱是 **xx.mcp**，最好與原始檔案放在同一個目錄以方便管理。
- 一個 **Project** 可記錄眾多資訊：
 - 視窗位置、大小、個數
 - 相關檔案的名稱、位置
 - 相關偵錯訊息的設定值
 - 組譯器、編譯器的選擇與設定
- 以 **Project** 觀念來保持一個完整的軟體設計，以便日後程式的維護、修改。



練習：建立新的專案 (Project)

- 執行 MPLAB IDE ，將顯示一初始的空白桌面



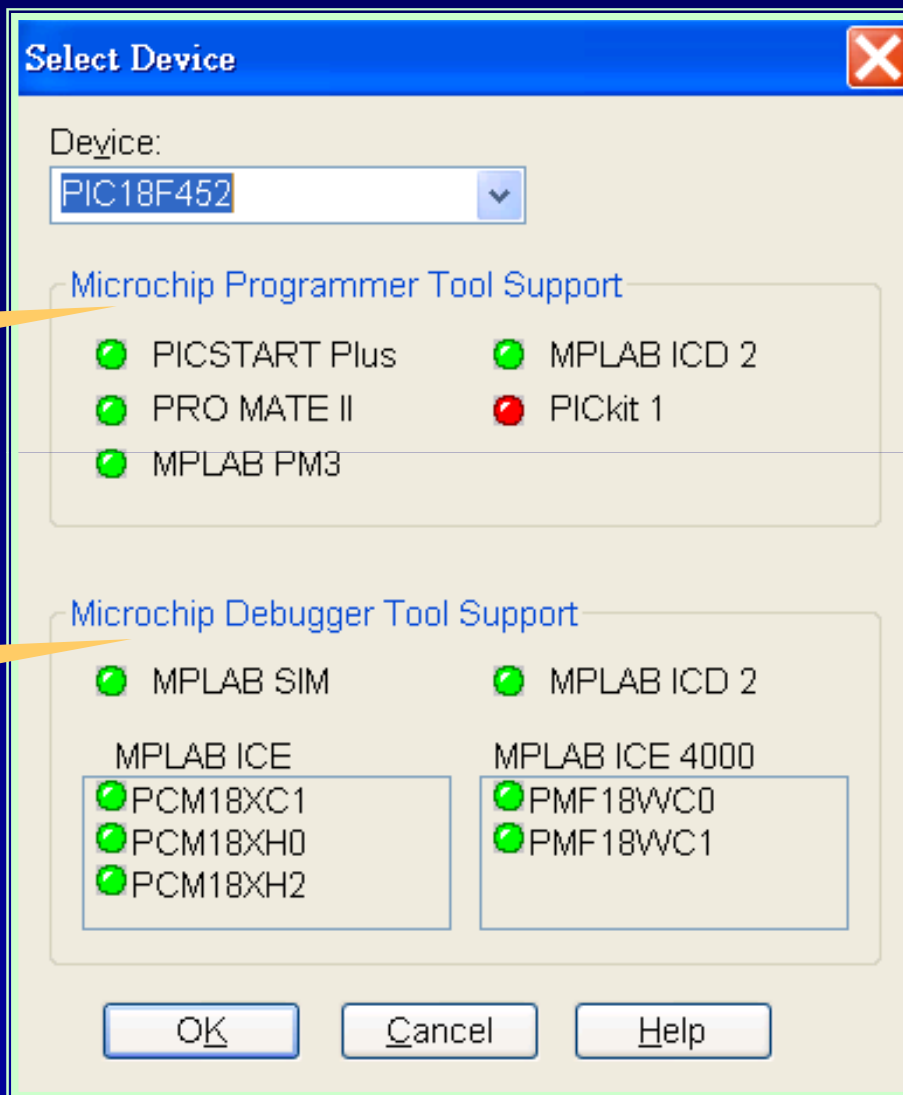


練習：建立新的專案

- 使用 “Configure → Select Device” 來指定 MCU 為 PIC18F452

目前所選擇的元件
所支援的燒錄器

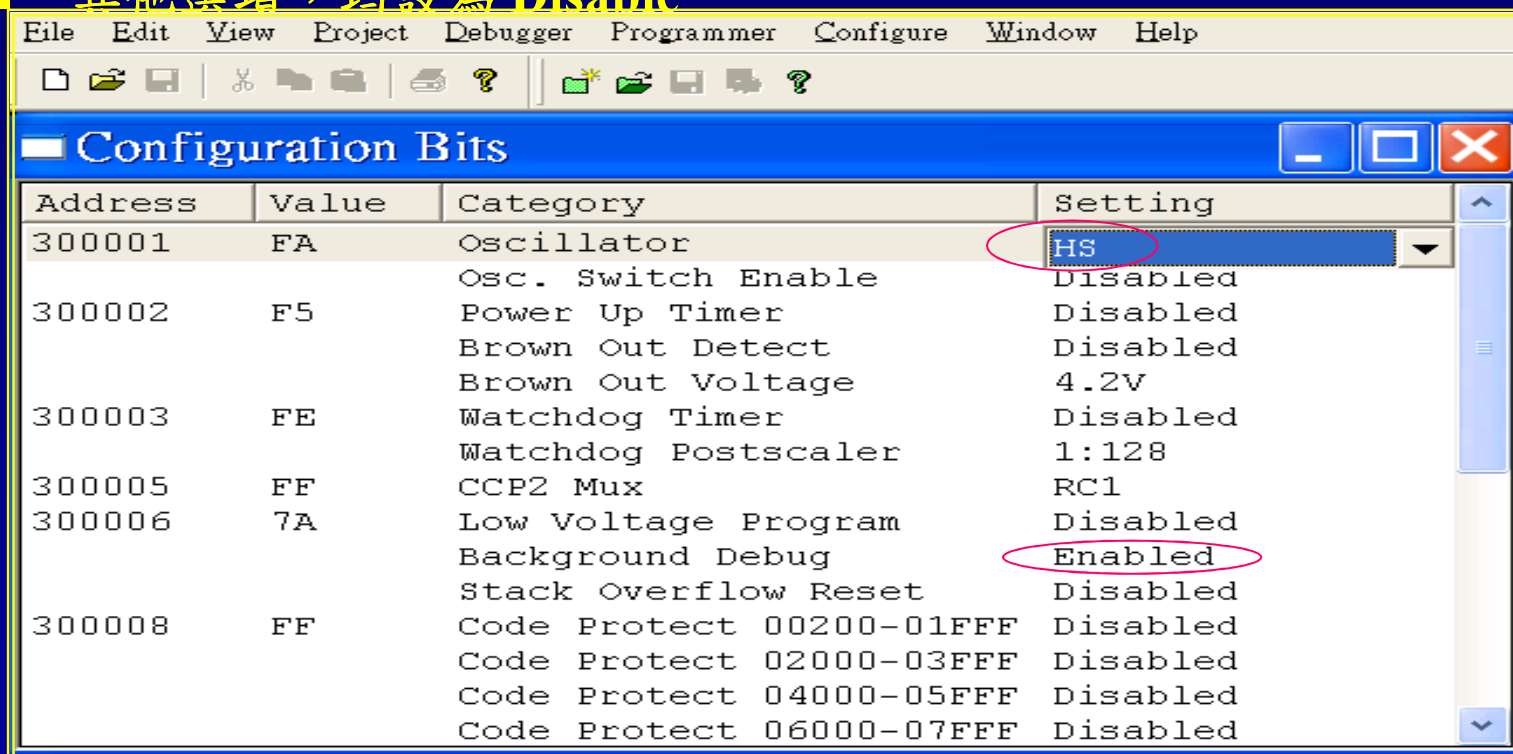
目前所選擇的元件
所支援的發展工具





練習：建立新的專案

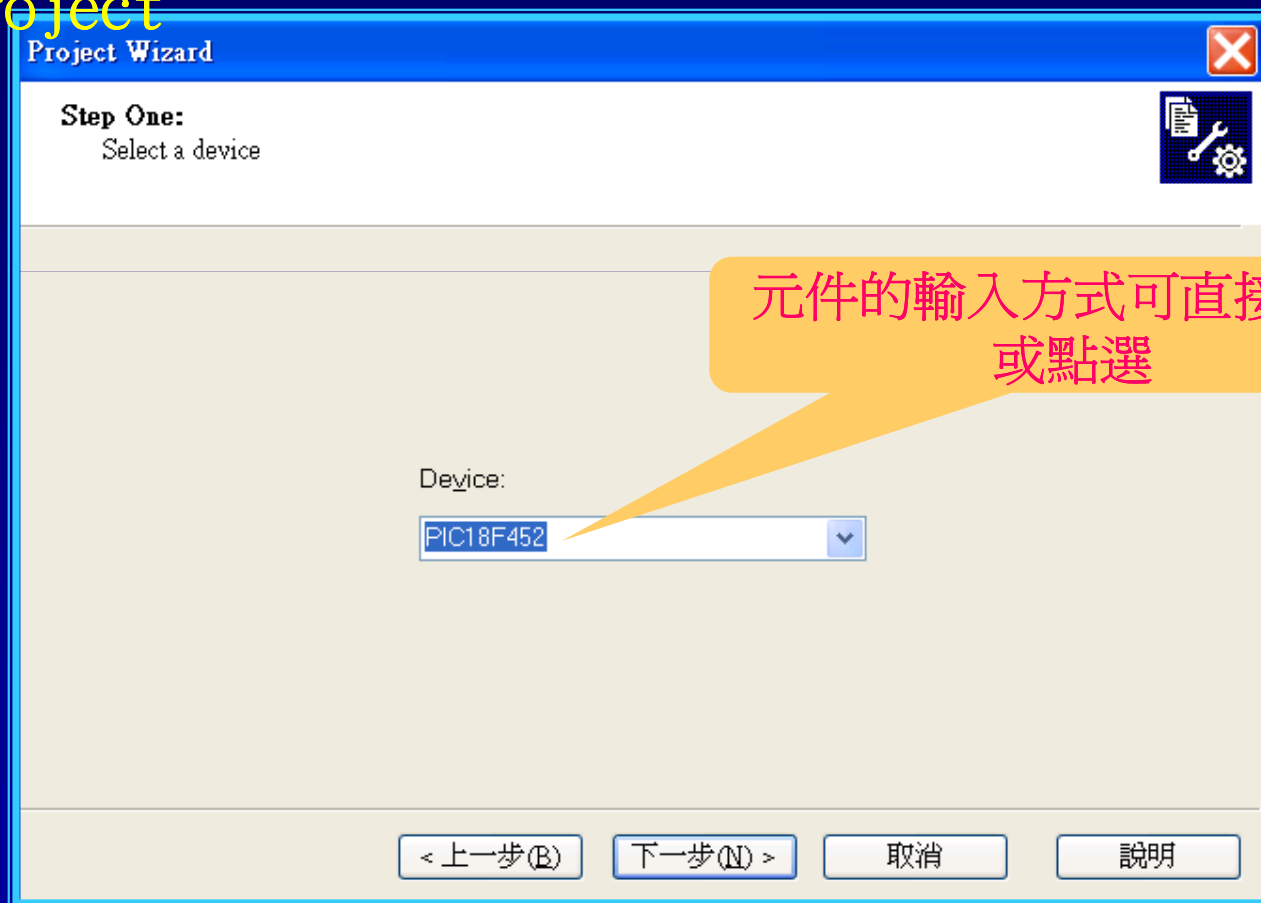
- 使用 “Configure → Configuration Bits” 來設定 IC 運作模式
 - 除 Background Debug 設定為 Enabled 以便使用 MPLAB ICD 2
 - 其他選項，均設為 Disable





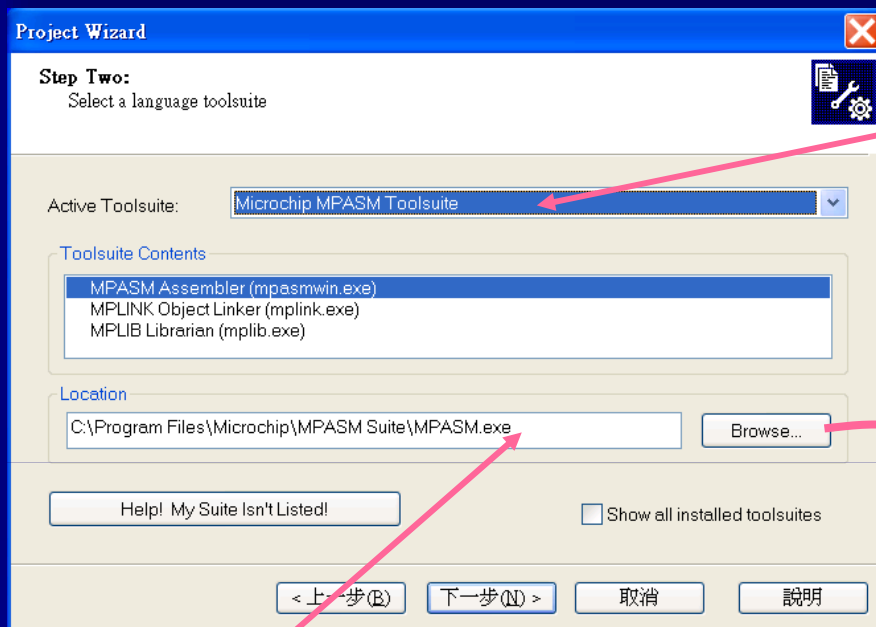
練習：建立新的專案

- 使用 “Project → Project Wizard 來建立一新的 Project





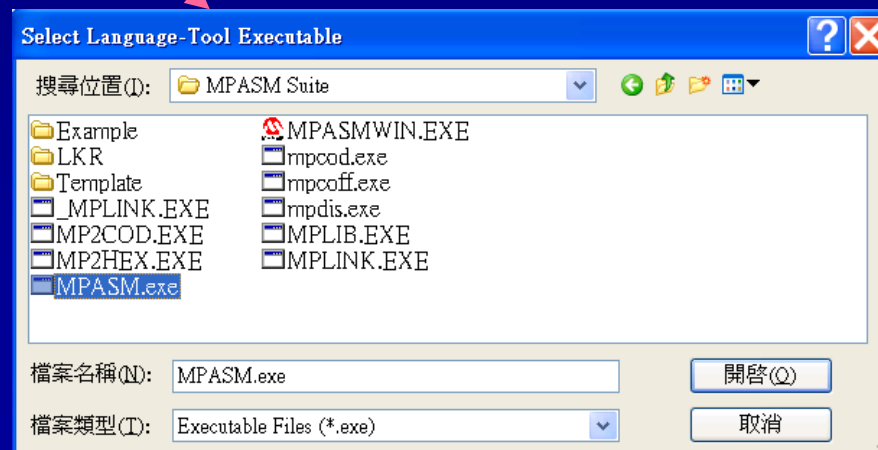
練習：建立新的專案



■ 設定語言工具

■ 也可以在 “Project → Set Language Toolsuite” 來設定使用 MPASM 來組譯

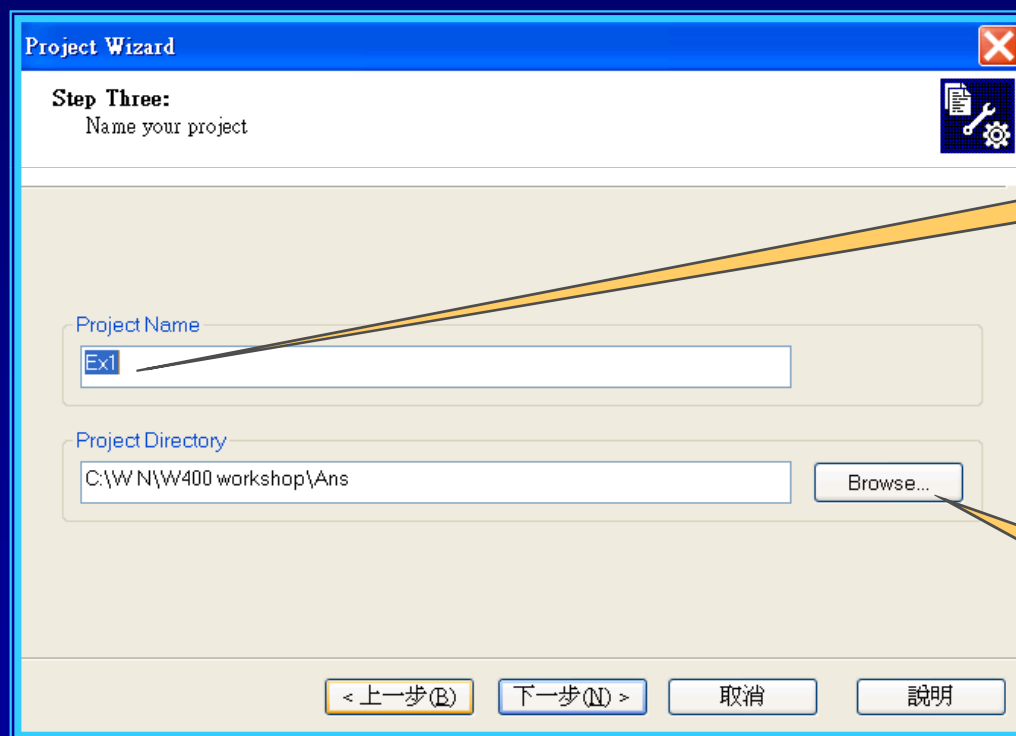
- 設定語言工具執行路徑
- 也可以用 “Project → Set Language Tool locations” 來確定相關程式所執行的位置





練習：建立新的專案

- 建立 Project 裡的 source code 及工作目錄



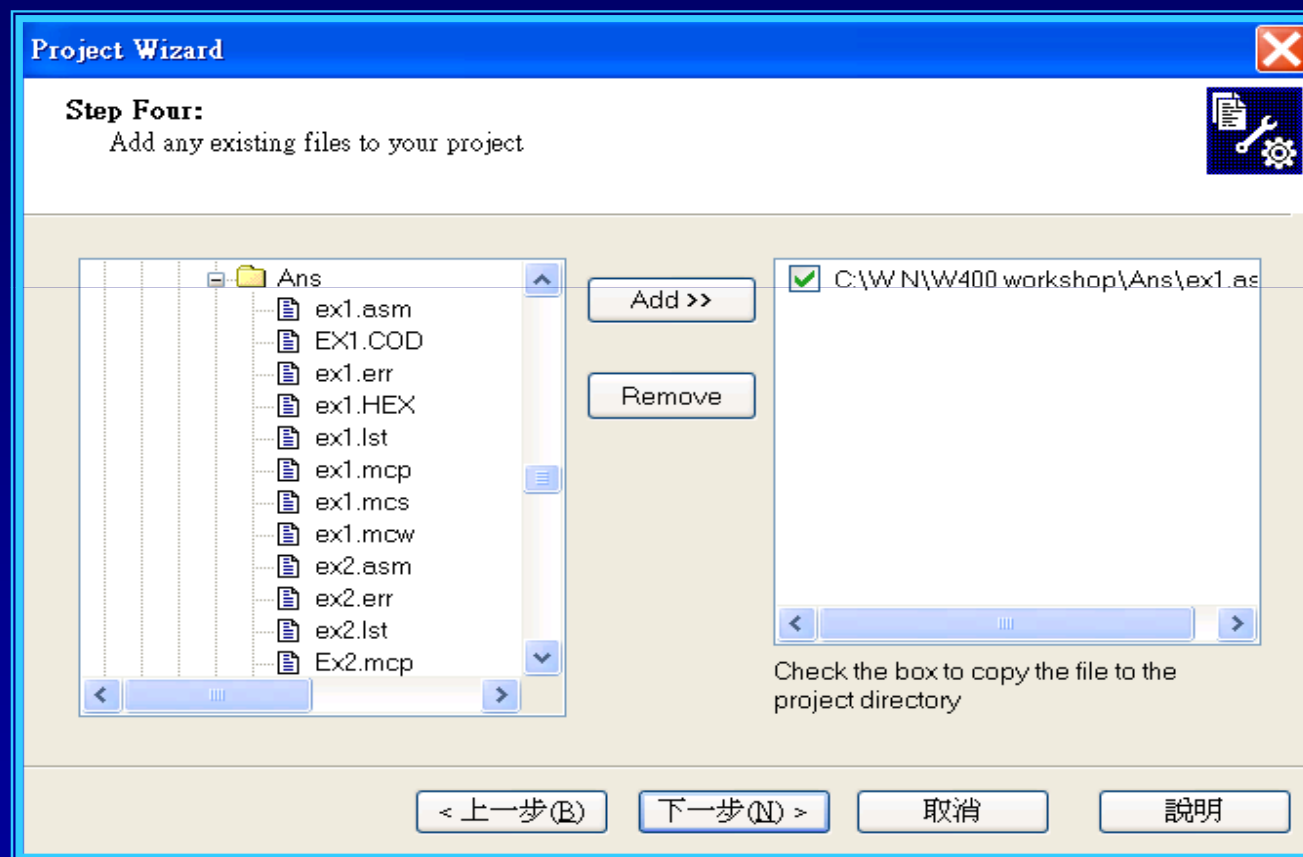
輸入專案名稱 Ex1.mcp

用 Browse 來指定目錄



練習：建立新的專案

■ 加入 source code 到 Project 的工作目錄





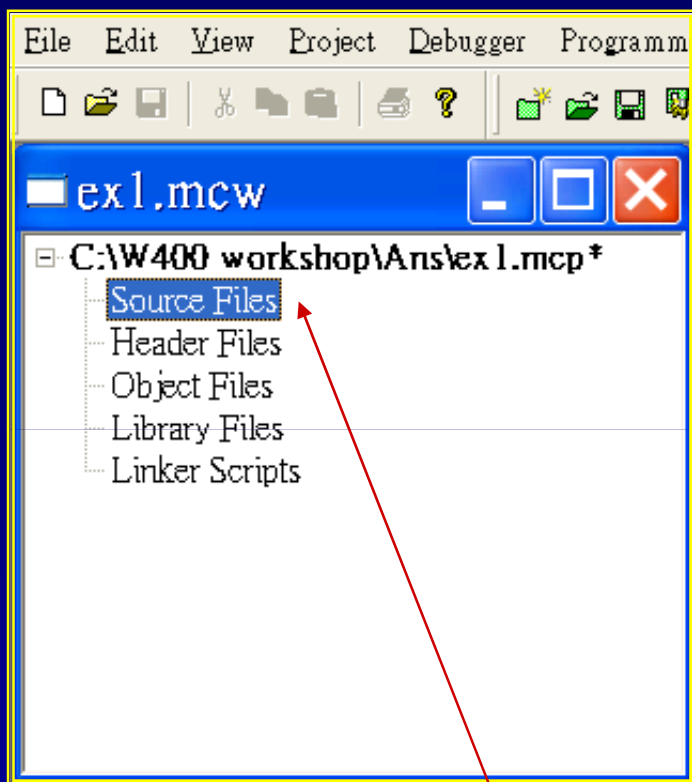
練習：建立新的專案

- **Project** 建立完成，請在確認設定是否正確

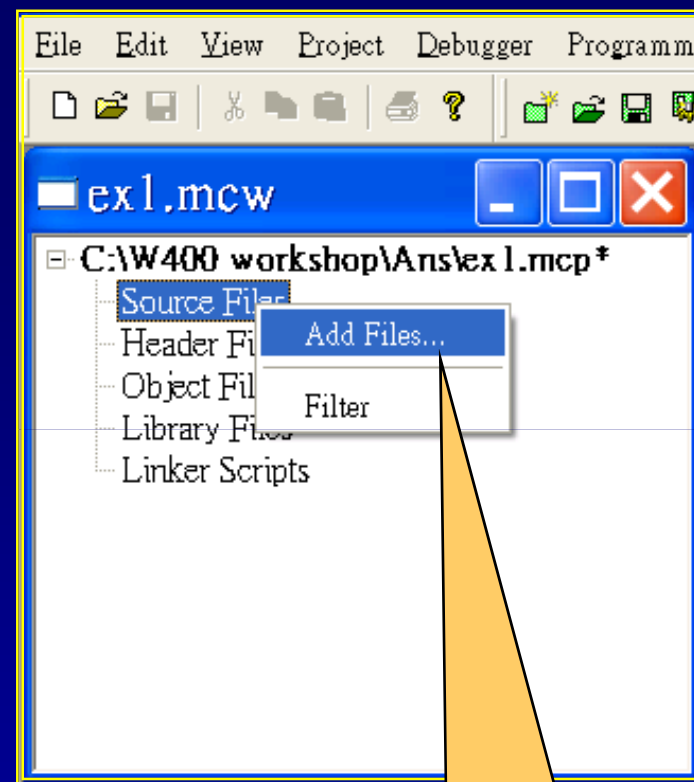




練習：加入原始檔案方式



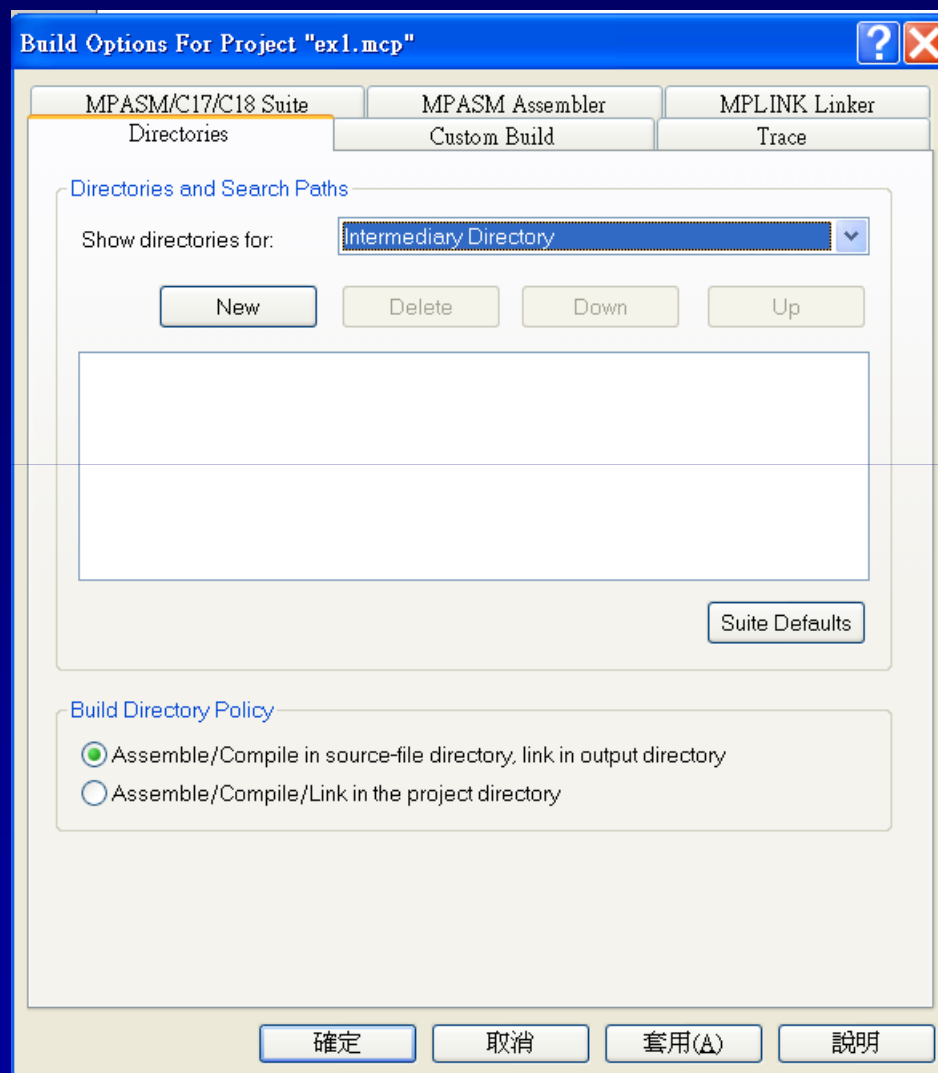
將滑鼠指到
Source Files
並按下滑鼠的右鍵



MPLAB IDE
將要求您加入檔案



練習：建立資料庫來源位置



- 使用 “Project → Build Options → Project” 啓動輸入視窗
- 該視窗是輸入資料庫、含入檔及連結描述檔的路徑位置，一般是使用 C 語言及多檔案的組合語言才需要輸入
- 使用單一組合語言者無需輸入路徑資料



練習：組譯程式

- 使用 Project → Build All 或 Ctrl+F10 來組譯程式

MPLAB IDE

File Edit View Project Debugger Programmer Configure Window Help

ex1.mcp C:\W400 workshop\Ans\ex1.asm

Source Files
ex1.asm
Header Files
Object Files
Library Files
Linker Scripts

Output

Build Find in Files MPLAB ICD 2

Deleting intermediary files... done.
Executing: "C:\Program Files\MPLAB IDE\MC
Loaded C:\W400 workshop\Ans\ex1.COD
BUILD SUCCEEDED

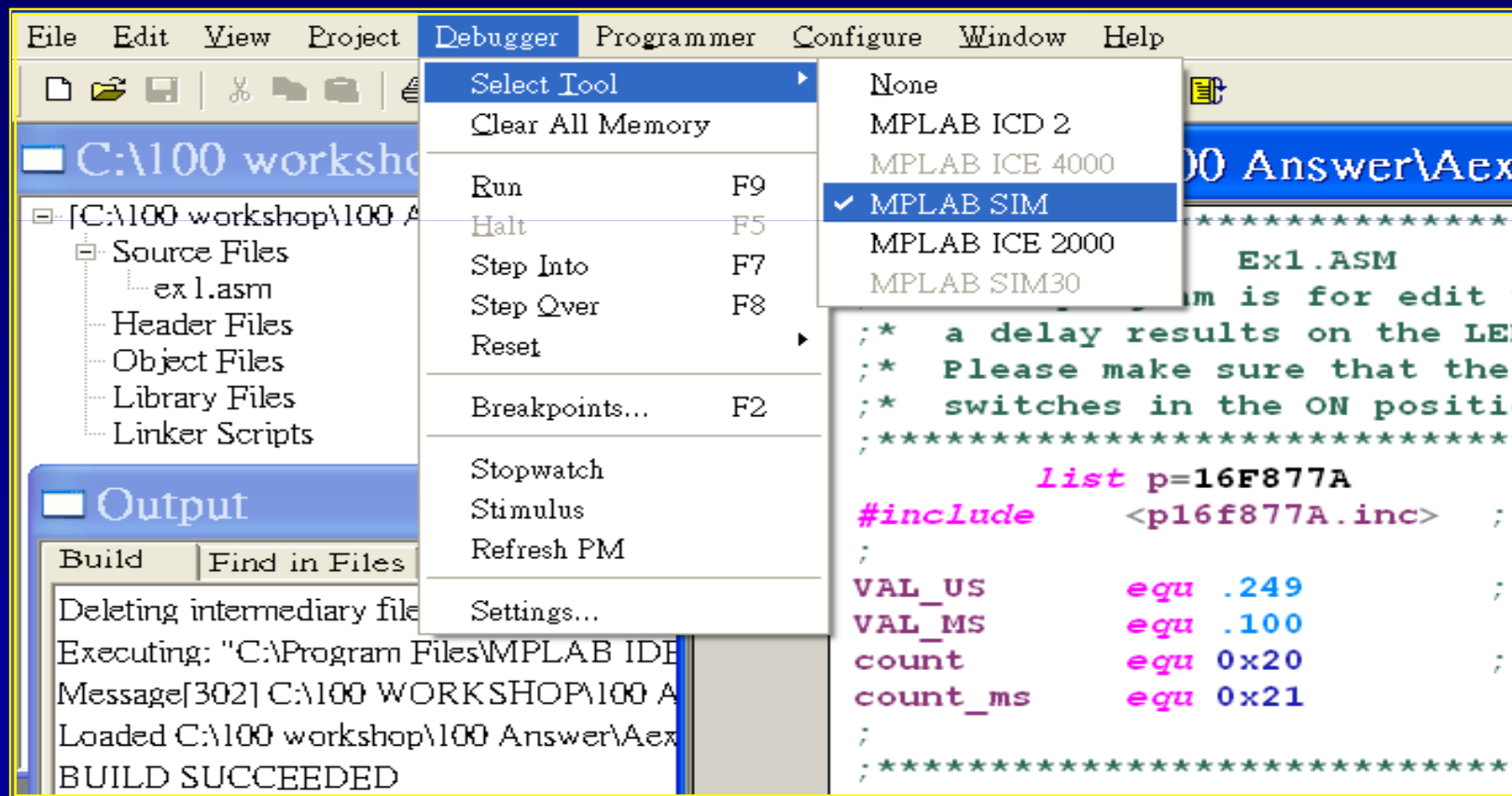
組譯成功

```
*****  
; The ex1.asm is exercise one for the I/O control  
; Press the SW3(RA4) the PORTD will show b11110000 on LED  
; Press the SW2(RB0) the PORTD will show b00001111 on LED  
*****  
  
LIST      P=18F452                ;directive to define process  
#include  <P18F452.INC>          ;processor specific variable def  
  
*****  
;  
#define   SW2      PORTB,0  
#define   SW3      PORTA,4  
  
*****  
;Reset vector  
; This code will start executing when a reset occurs.  
  
ORG       0x0000  
  
goto      Main                ;go to start of main code  
  
*****  
;Start of main program
```




啟動基本 MPLAB-SIM

- 點選 “Debugger → Select Tool → MPLAB SIM” 以啟動內建的軟體模擬程式





MPLAB SIM 的基本除錯功能

The screenshot shows the MPLAB IDE interface. The code editor displays the following code:

```
55 {  
56     ADCON0bits.GO=1;  
57     while(ADCON0bits.GO);  
58  
59     AD_Temp=ReadADC();  
60     AD_Result.AD_10bit=AD_T  
61  
62     SetDCPWM1(AD_Result.AD_  
63     AD_Result.AD_10bit>>2;  
64     PORTD=AD_Result.AD[0];  
65  
66
```

Callouts and their corresponding functions:

- Make Project**: Points to the 'Build' icon in the toolbar.
- 執行** (Execute): Points to the 'Run' icon in the toolbar.
- 單步執行** (Step Over): Points to the 'Step Over' icon in the toolbar.
- Reset**: Points to the 'Reset' icon in the toolbar.
- Build All**: Points to the 'Build All' icon in the toolbar.
- 停止** (Stop): Points to the 'Stop' icon in the toolbar.
- Step Over**: Points to the 'Step Over' icon in the toolbar.

The Stopwatch window is open, showing the following data:

Instruction Cycles	Time (mSecs)	Processor Frequency (MHz)
119516	119.516	4

Buttons in the Stopwatch window: Zero, 關閉 (Close), 說明 (Help).

利用mouse的右鍵
來設定中斷點

利用Stopwatch視窗
來量測程式執行
所需的時間



Watch Window 的重要性

- 變數在 RAM 的位置是誰安排的
- 你知道怎樣才能看到變數的內容
- MPLAB IDE 有專屬的變數觀察視窗

變數位置

變數名稱

變數內容

Watch		
Add SFR	PIR1	Add Symbol DS_Zero_FLG
Address	Symbol Name	Value
0098	AD_Temp	0004
009A	DS_Zero_FLG	00100000
0FE8	WREG	D2
0F81	PORTB	3F
0FC4	ADRESH	00
0F9E	PIR1	00000000
Watch 1 Watch 2 Watch 3 Watch 4		



編譯程式與除錯

- 利用 **Build All (Ctrl + F10)** 來編譯程式
 - 如有錯誤，請參考錯誤訊息並加以修正
 - 在錯誤訊息處按滑鼠兩次即能輕易切換至錯誤行
- 利用**Watch Window**來觀察變數
- 利用**Mouse**的右鍵來設定中斷點
 - 熟悉**Reset**，**Run**，**Halt**功能
 - **Step**，**Step Over**功能
 - **RAM**，**SFR** 視窗在 組合 語言下就不是那麼重要了？
- 其它的視窗可在“**view**” 下開啓



MICROCHIP

使用 MPLAB ICD2 除錯

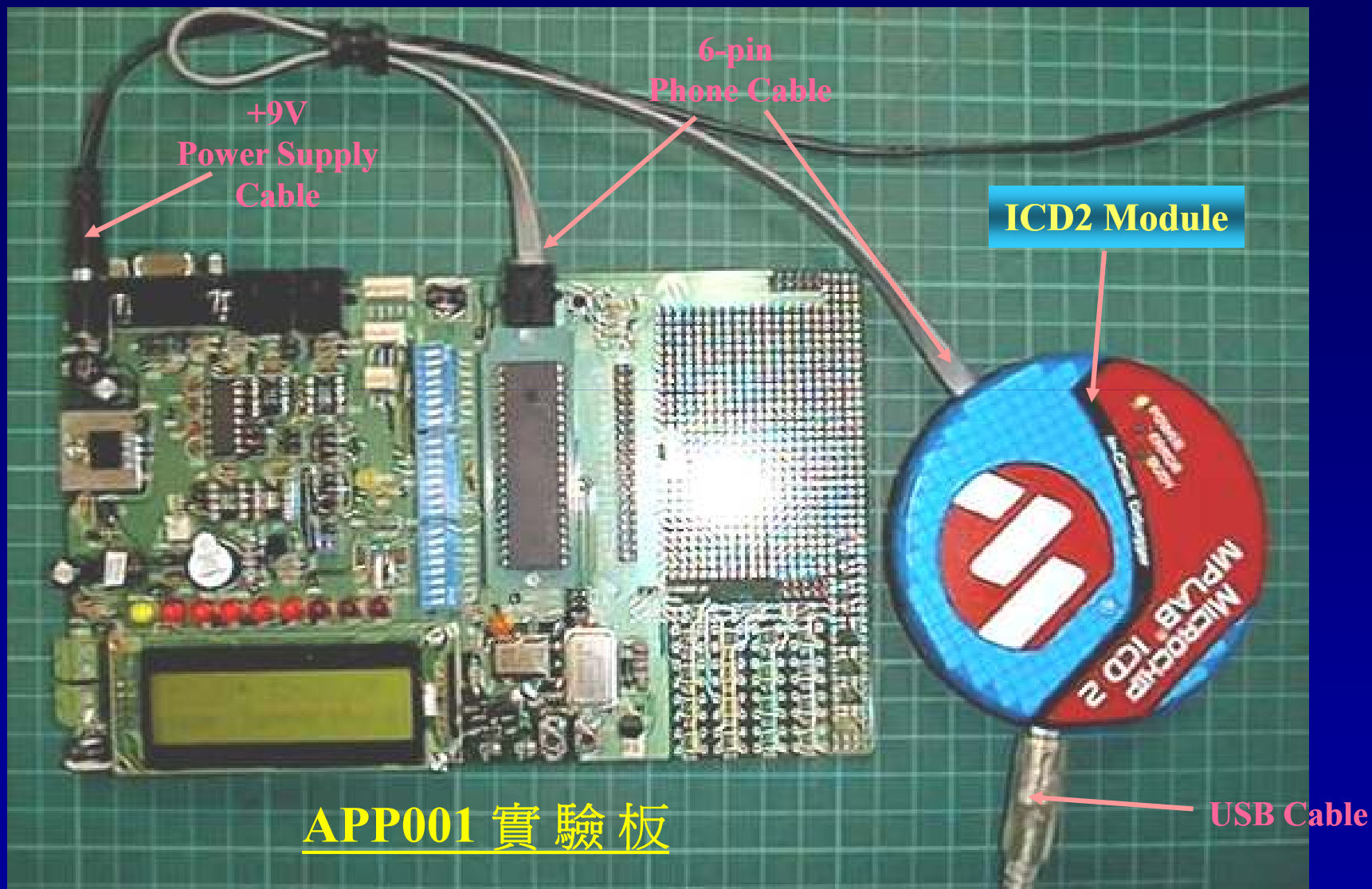


MPLAB-ICD2 功能

- 全速執行
- 單步執行
- 單/多點硬體中斷
- 變數觀察，原始程式除錯等級
- 快速載入程式到模擬元件
- 可當模擬元件的燒錄工具
- 工作電壓：**2.5V to 5.5V**
- 頻率範圍：**32KHz to 40MHz**
- **RS-232 或 USB** 介面
- 價格便宜



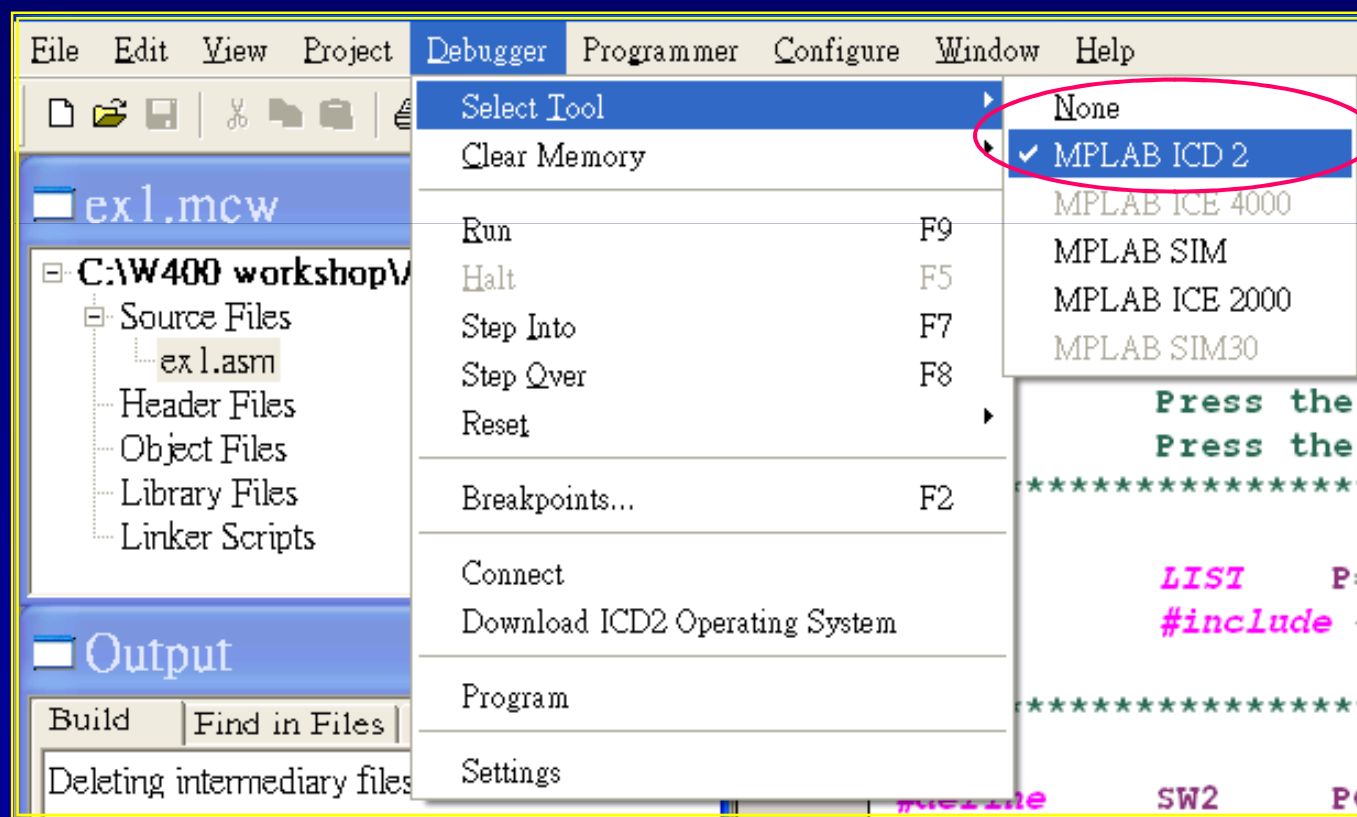
MPLAB-ICD2 配線圖





啟動 ICD2

- 點選 “**Debugger → Select Tool → MPLAB ICD2**” 以啟動內建的軟體模擬程式





MPLAB-ICD2 注意事項

PIC18F452 除錯時

- ICD2 會佔用 18F452 最後的程式空間 (除錯程式使用)
- 任何修改程式的動作，需重新執行燒錄動作
- 只能設定一個中斷除錯點
- 程式記憶體及 EEPROM 的內容值，在除錯的過程中不會自動更新，如需更新需以讀取 Device 的方式更新
- ICD2 不支援堆疊視窗功能
- ICD2 不支援 SLEEP 功能
- ICD2 在除錯的過程中不能啟動 Watch-Dog Timer
- RB6 & RB7 保留給 ICD 做除錯用
- MCLR pin 會出現 13V 的電壓 (thru a 1kohm resistor)



使用 MPLAB-ICD2

- 有關使用 ICD2 的詳細步驟，請上網參閱：
 - **MPLAB® ICD 2 In-Circuit Debugger User's Guide**

- 使用 ICD2 時，務必重新載入目前所使用 MPLAB IDE 版本的 ICD2 Firmware 以確保 ICD2 能與 MPLAB IDE 相容
 - 先點選 “**Debugger → Select Tool → MPLAB ICD2**”
 - 再點選 “**Debugger → Download ICD2 Operating System**”