歡迎參加 W400 教育訓練課程

- 使用軟體工具
  - **MPLAB IDE v7.xx (或更新版本)**
  - **MPASM , MPLINK , MPLIB**
- 使用硬體工具
  - **MPLAB ICD2**
  - **Microchip APP001 Workshop Board (PIC18F452 inside)**
- 參考書籍 (可以從網站上下載)
  - **MPASM User's Guide with MPLINK and MPLIB (DS33014J)**
  - **MPLAB IDE v6.10 中文使用手冊**
  - **PIC18Fxx2 Data Sheet (DS39564A)**

PICmicro Workshop 400
課程內容

- PIC18F系列
  - PIC18Fxxxx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

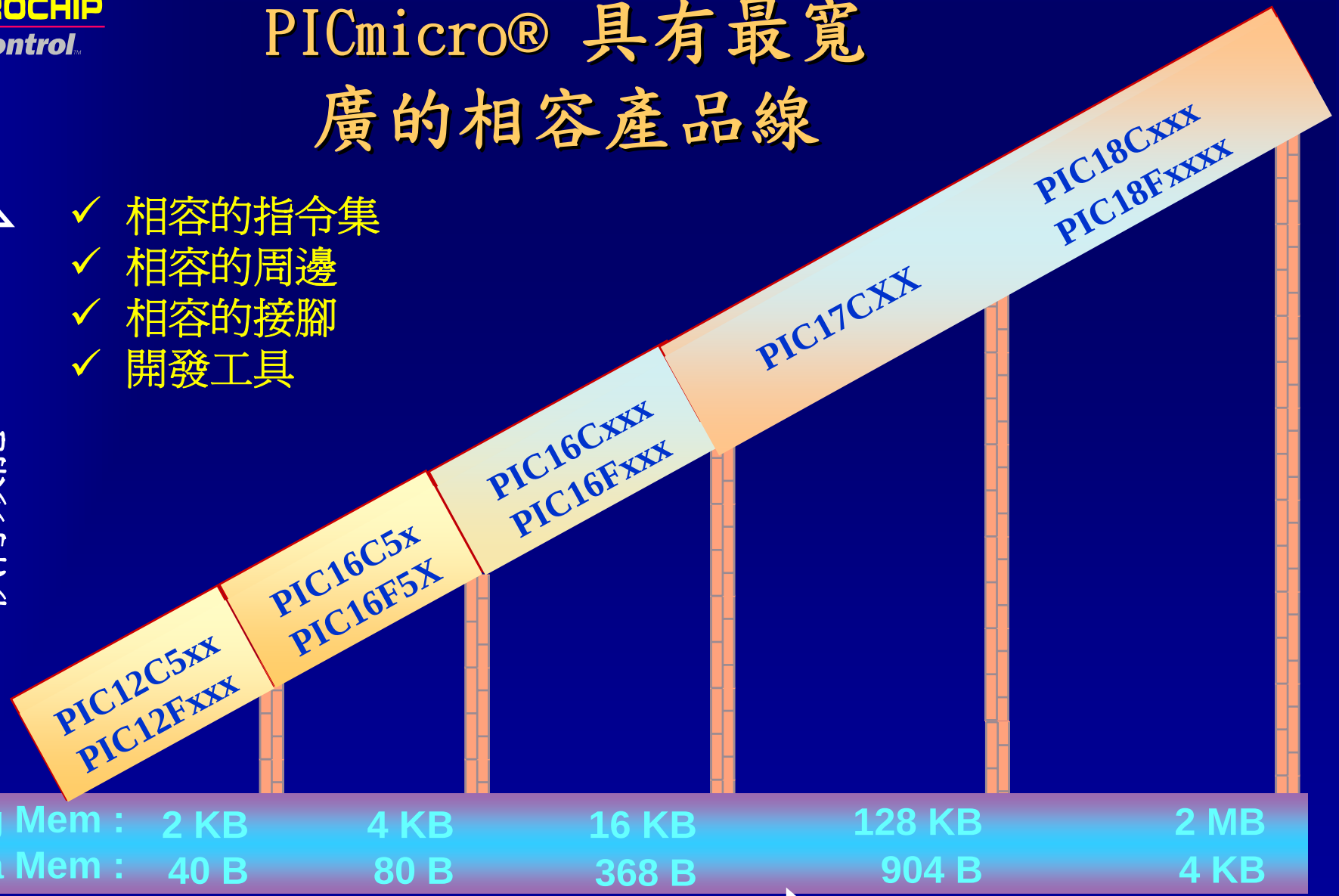
- 練習一 ： I/O 的操作
- 練習二 ： 使用 Timer1 與 中斷
- 練習三 ： 使用 A/D 轉換器
- 練習四 ： 使用 USART
- 練習五 ： Table Read

PIC18Fxxxx介紹

PICmicro® 具有最寬廣的相容產品線

- ✓ 相容的指令集
- ✓ 相容的周邊
- ✓ 相容的接腳
- ✓ 開發工具

執行效能



記憶體空間

PIC18F452

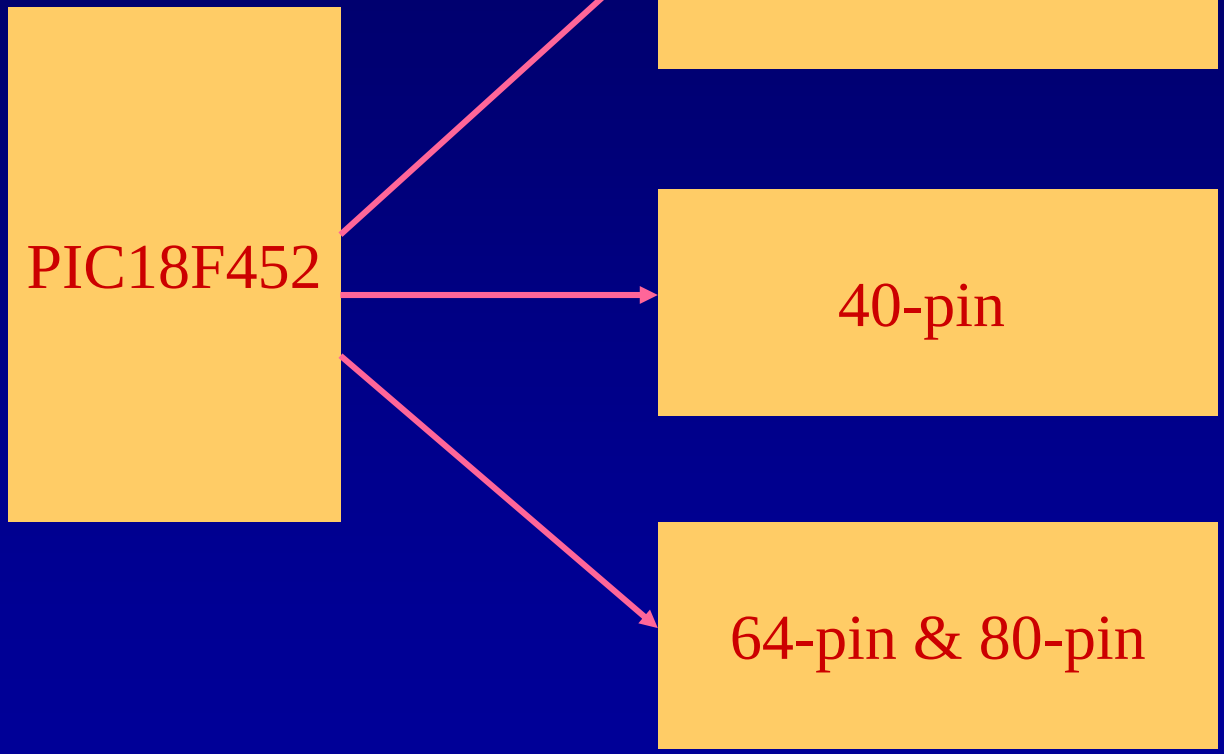
- 最早的 PIC18Fxxx 元件，目前 Silicon Revision C1
- 32KBytes Flash, 256B EEPROM, 1.5KB RAM
- Timer : Timer0, 1, 2, 3, Watch-Dog Timer
- 2 CCP modules
- I²C, SPI , USART, PSP
- 10-bit ADC with 8 Channel
- PBOR, PLVD, ICD, Self Programming
- 40-pin PDIP(44ML, 44PT) with 34 I/O pin
- 40MHz @ 5V

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** 啓動模式
  - 雙速啓動模式
  - 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**

PIC18Fxxxx 腳位的延伸



PIC18F1220, PIC18F1320
PIC18F2220, PIC18F2320
PIC18F2410, PIC18F2420
PIC18F2680, PIC18Fxxxx
(目前統計 29 個元件)

PIC18F4520, PIC18F4320
PIC18F4431, PIC18F4455
PIC18F4480, PIC18F4525
PIC18F4620, PIC18Fxxxx
(目前統計 24個元件)

PIC18F6310, PIC18F6680
PIC18F6520, PIC18F6585
PIC18F8410, PIC18F8622
PIC18F8722, PIC18Fxxxx
(目前統計 26 個元件)

PIC18Fxxxx + FS USB +CAN

PIC18F452

+

FS USB

PIC18F2455
PIC18F2550
PIC18F4455
PIC18F4550

CAN
or
ECAN

ECAN
PIC18F2480
PIC18F2580
PIC18F2585
PIC18F2680
PIC18F4480
PIC18F4580
PIC18F4585
PIC18F4680
PIC18F6585

ECAN
PIC18F8585
PIC18F8680

CAN
PIC18F248
PIC18F258
PIC18F448
PIC18F458

PIC18Fxxxx + Motor +LCD

PIC18F452

LCD

PIC18F6390
PIC18F6490
PIC18F8390
PIC18F8490

Motor Control

PIC18F1230
PIC18F1330
PIC18F2331
PIC18F2431
PIC18F4331
PIC18F4431

PIC18FxxJxx J系列

- 全系列爲低電壓 2.0V – 3.6V @ 10MIPs

High-Performance 8-Bit PICmicro® Microcontroller Family (16-bit Instruction Set) (continued)

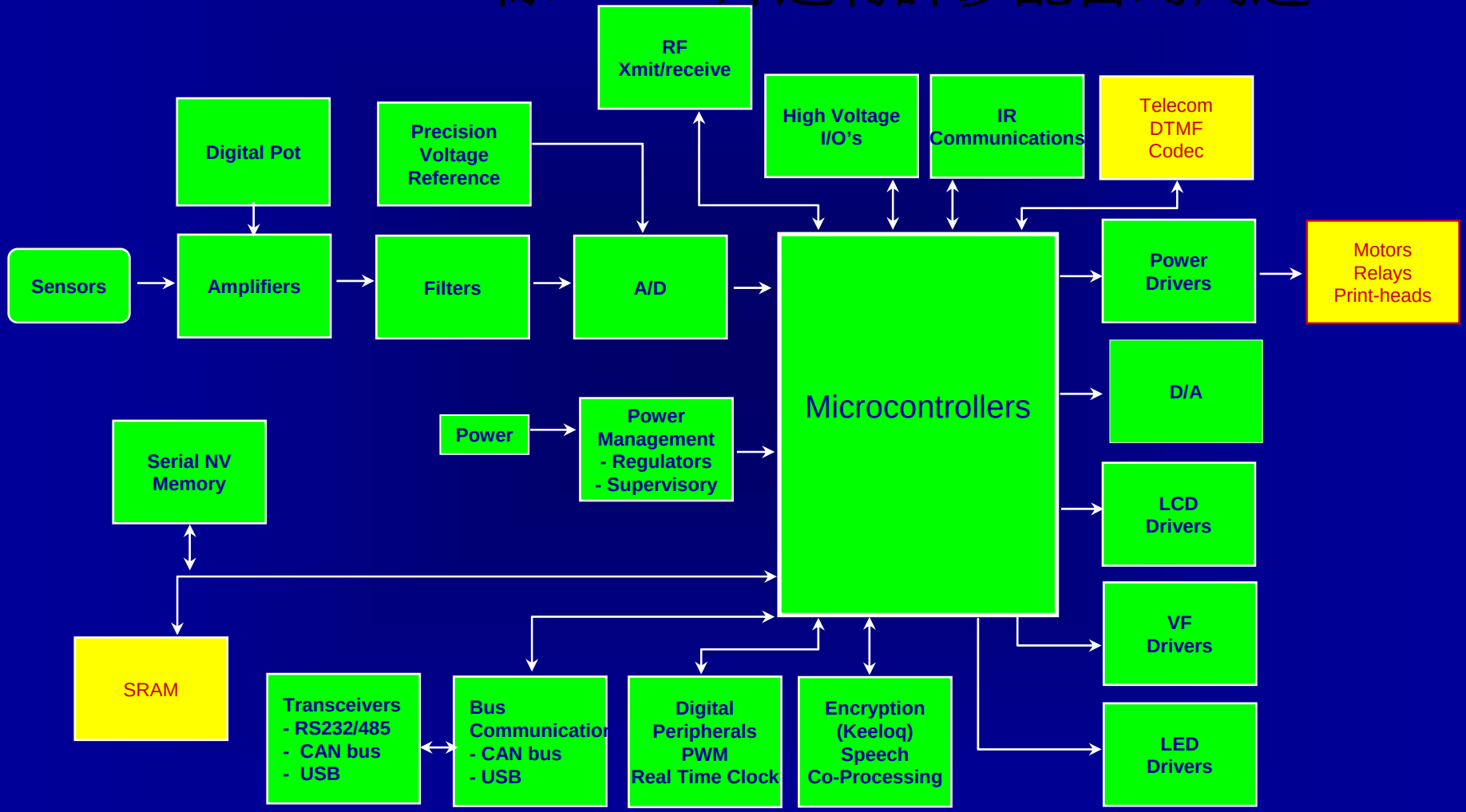
Product	Program Memory Bytes & Type (Words)	EEPROM Data Memory Bytes	RAM Bytes	I/O Pins	Packages	Analog		Digital		Max. Speed MHz	IntOSC	BOR/ PBOR/ PLVD	ICD # of Breakpoints	CCP/ECCP	nW	Other Features
						ADC Ch	Comp.	Timers/WDT	Serial I/O							
PIC18F24J10	16,384 StdFl (8,192)	—	1024	21	28SP, 28SO, 28SS, 28ML	10x10-bit 100 ksps	2	2-16 bit, 1-8 bit, 1-WDT	EUSART, MI²C/SPI	40	32 kHz	BOR	3	2/0	✓	
PIC18F25J10	32,768 StdFl (16,384)	—	1024	21	28SP, 28SO, 28SS, 28ML	10x10-bit 100 ksps	2	2-16 bit, 1-8 bit, 1-WDT	EUSART, MI²C/SPI	40	32 kHz	BOR	3	2/0	✓	
PIC18F44J10	16,384 StdFl (8,192)	—	1024	32	40P, 44ML, 44PT	13x10-bit 100 ksps	2	2-16 bit, 1-8 bit, 1-WDT	EUSART, 2xMI²C/SPI	40	32 kHz	BOR	3	1/1	✓	PSP
PIC18F45J10	32,768 StdFl (16,384)	—	1024	32	40P, 44ML, 44PT	13x10-bit 100 ksps	2	2-16 bit, 1-8 bit, 1-WDT	EUSART, 2xMI²C/SPI	40	32 kHz	BOR	3	1/1	✓	PSP
PIC18F65J10	32,768 StdFl (16,384)	—	2048	50	64PT	11x10-bit 100 ksps	2	3-16 bit, 2-8 bit, 1-WDT	2xEUSART, 2xMI²C/SPI	40	32 kHz	BOR	3	2/3	✓	PSP
PIC18F65J15	49,152 StdFl (24,576)	—	2048	50	64PT	11x10-bit 100 ksps	2	3-16 bit, 2-8 bit, 1-WDT	2xEUSART, 2xMI²C/SPI	40	32 kHz	BOR	3	2/3	✓	PSP
PIC18F66J10	65,536 StdFl (32,768)	—	2048	50	64PT	11x10-bit 100 ksps	2	3-16 bit, 2-8 bit, 1-WDT	2xEUSART, 2xMI²C/SPI	40	32 kHz	BOR	3	2/3	✓	PSP
PIC18F66J15	98,304 StdFl (49,152)	—	3936	50	64PT	11x10-bit 100 ksps	2	3-16 bit, 2-8 bit, 1-WDT	2xEUSART, 2xMI²C/SPI	40	32 kHz	BOR	3	2/3	✓	PSP
PIC18F67J10	131,072 StdFL (65,536)	—	3936	50	64PT	11x10-bit 100 ksps	2	3-16 bit, 2-8 bit, 1-WDT	2xEUSART, 2xMI²C/SPI	40	32 kHz	BOR	3	2/3	✓	PSP
PIC18F85J10	32,768 StdFL (16,384)	—	2048	66	80PT	15x10-bit 100 ksps	2	3-16 bit, 2-8 bit, 1-WDT	2xEUSART, 2xMI²C/SPI	40	32 kHz	BOR	3	2/3	✓	PSP, EMA
PIC18F85J15	49,152 StdFL (24,576)	—	2048	66	80PT	15x10-bit 100 ksps	2	3-16 bit, 2-8 bit, 1-WDT	2xEUSART, 2xMI²C/SPI	40	32 kHz	BOR	3	2/3	✓	PSP, EMA
PIC18F86J10	65,536 StdFL (32,768)	—	2048	66	80PT	15x10-bit 100 ksps	2	3-16 bit, 2-8 bit, 1-WDT	2x EUSART, 2x MI²C/SPI	40	32 kHz	BOR	3	2/3	✓	PSP, EMA
PIC18F86J15	98,304 StdFL (49,152)	—	3936	66	80PT	15x10-bit 100 ksps	2	3-16 bit, 2-8 bit, 1-WDT	2x EUSART, 2x MI²C/SPI	40	32 kHz	BOR	3	2/3	✓	PSP, EMA
PIC18F87J10	131,072 StdFl (65,536)	—	3936	66	80PT	15x10-bit 100 ksps	2	3-16 bit, 2-8 bit, 1-WDT	2x EUSART, 2x MI²C/SPI	40	32 kHz	BOR	3	2/3	✓	PSP, EMA

MICROCHIP

PIC18FXXX

Enhanced PICmicro MCU
的架構及主要特色

典型的控制系統 –
除 CPU 外還有許多配合的周邊



您需要什麼樣的處理器？

- 執行效能要符合系統需求
- 需整合適當的周邊以便簡化設計及降低系統成本
- 具備高度的穩定性以減少產品的售後服務費用
- 容易取得的資源
  - 技術支援
  - 適量的市場現貨流通量
  - 開發工具
  - 樣品的提供
- 有延伸性的產品以便功能提昇或進行 **“Cost Down”**
- 合理且穩定的市場價格！

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

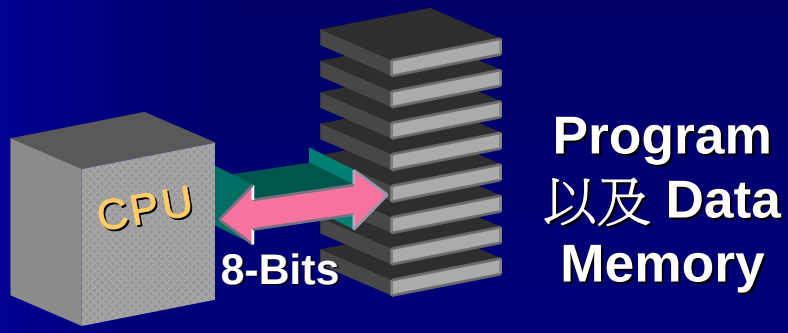
PIC18F 增加的功能

- **FLASH Program Memory**
  - 除了程式記憶體爲 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，PIC18F452 燒錄時間約 3 秒 (32KB 程式)
  - 每個 CPU 都可視爲一個 ICE Chip
- **EEPROM**
  - PIC18FXX2 都有內建 256 Bytes 的 EEPROM
  - 100 萬次的寫入壽命

PICmicro 的架構比較

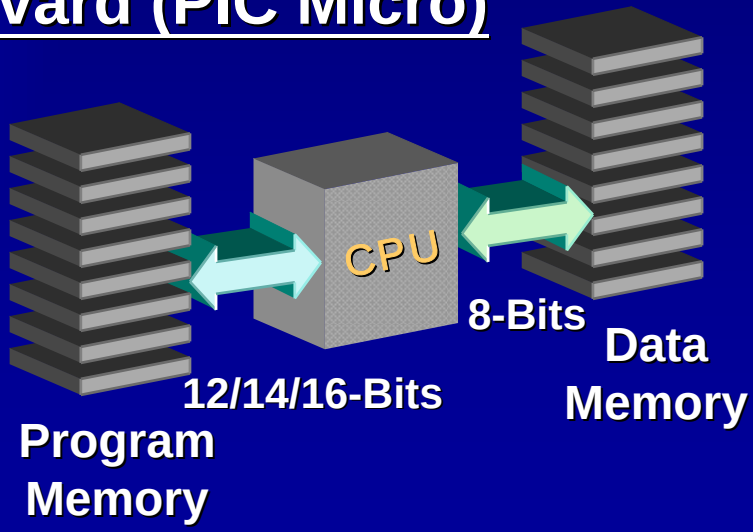
PIC 使用 Harvard Architecture

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

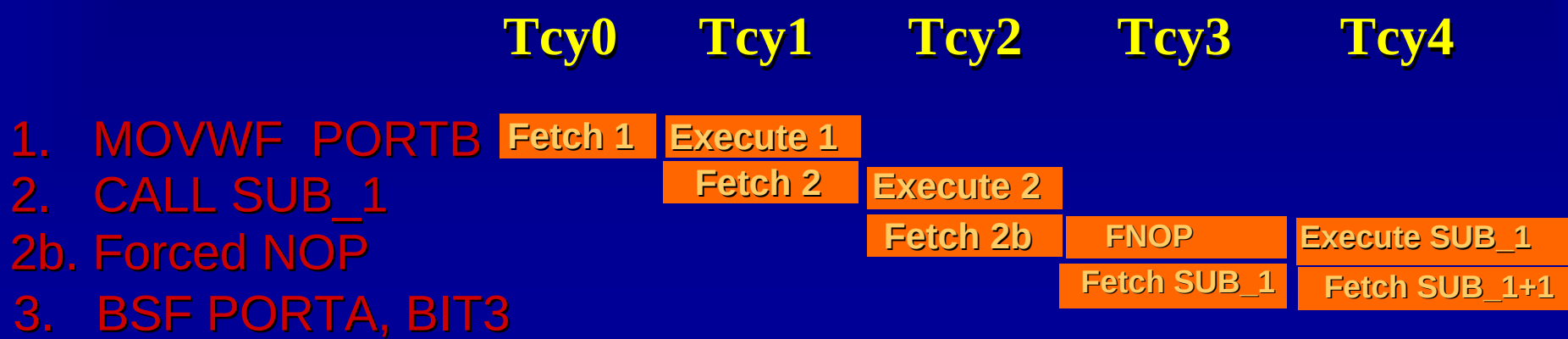
Harvard (PIC Micro)



- 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)

PICmicro® MCU 的架構
Pipelining (管線式的指令執行)

- 對大部份的 MCU 而言， 指令的提取與執行是順續發生的，每一個指令周期只能有一個動作發生
- PICmicro 使用 Harvard 結構且使用管狀式 (Pipeline) 的運作模式
- Pipeline 讓指令的提取與執行可同時進行.
- 指令的執行時間只需一個周期
- 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



PICmicro® MCU 的架構

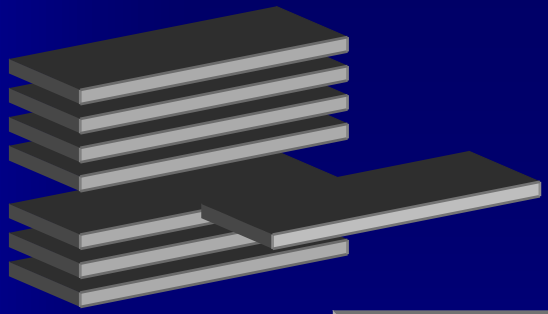
長指令寬度的編碼方式（Long Word Instruction）

- PICmicro 指令寬度爲 12, 14 or 16-bits，稱爲一個
- Instruction Word
- PICmicro 的資料匯流排(Data Bus)寬度是 8 bits
- Harvard architecture: 指令可於單一周期完成
- 若於 PIC18Fxxx 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.
- 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

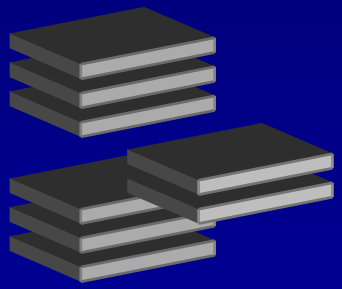
（Long Word Instruction）



PICmicro MCU

```
movlw     #imm<8>
```

0000 1110	k k k k k k k k

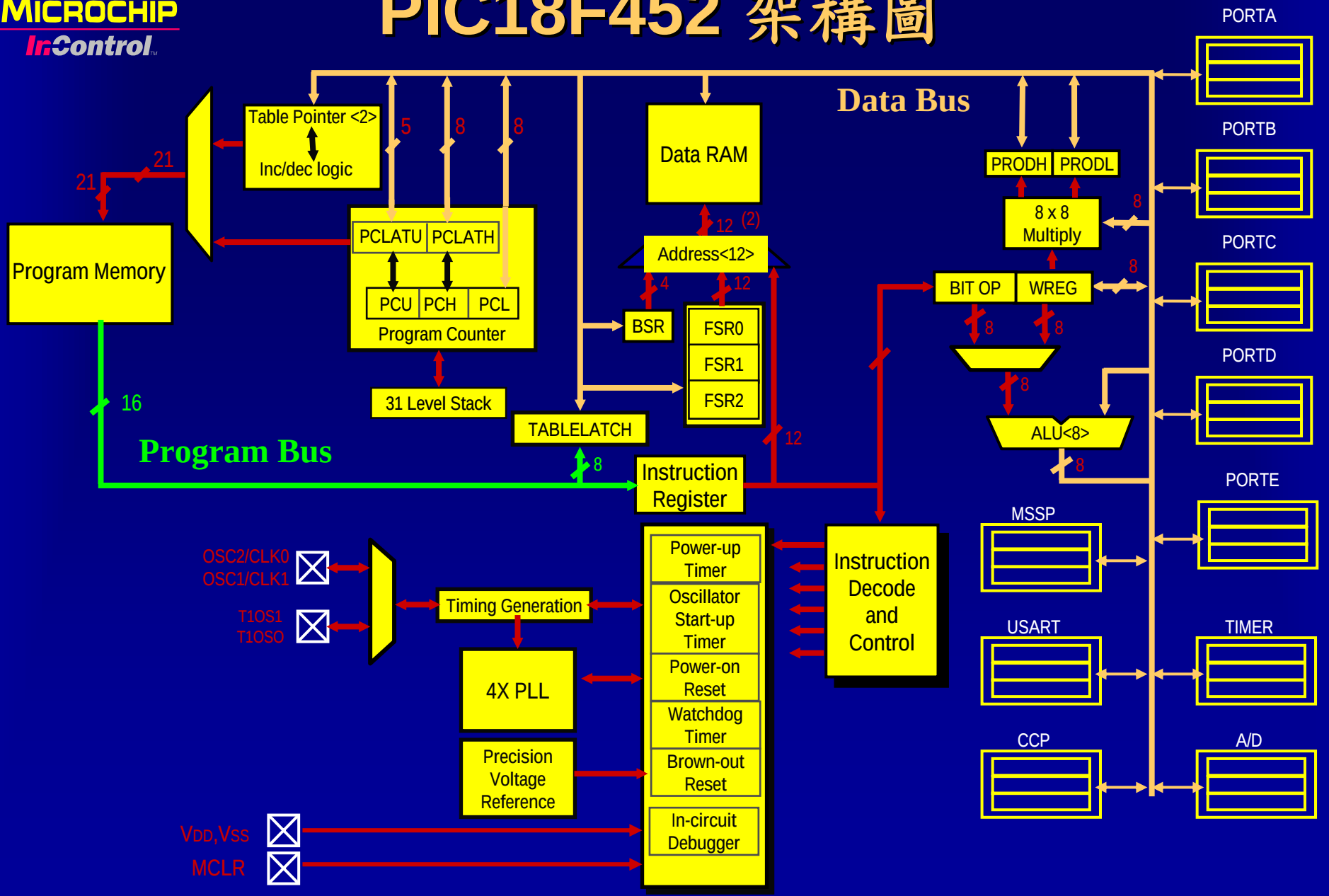


MC68HC05

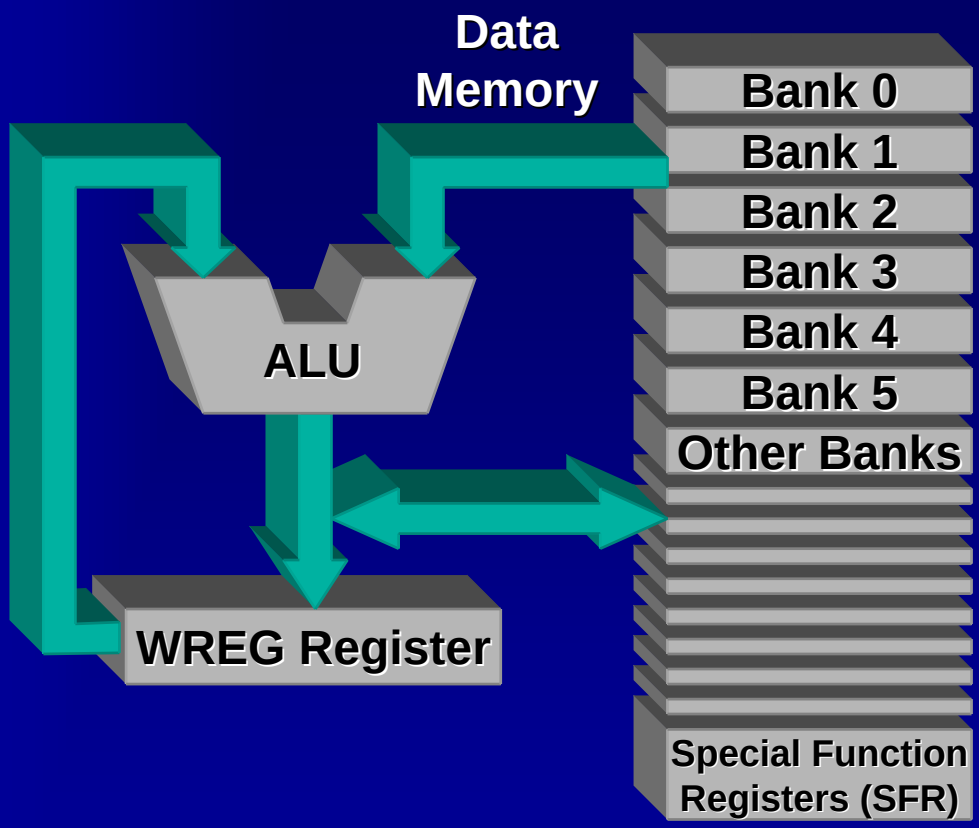
```
ldaa      #imm<8>
```

1000	0110
k k k k k k k k	

PIC18F452 架構圖



PICmicro® PIC18 系列的架構
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爲 Register files 的一部份

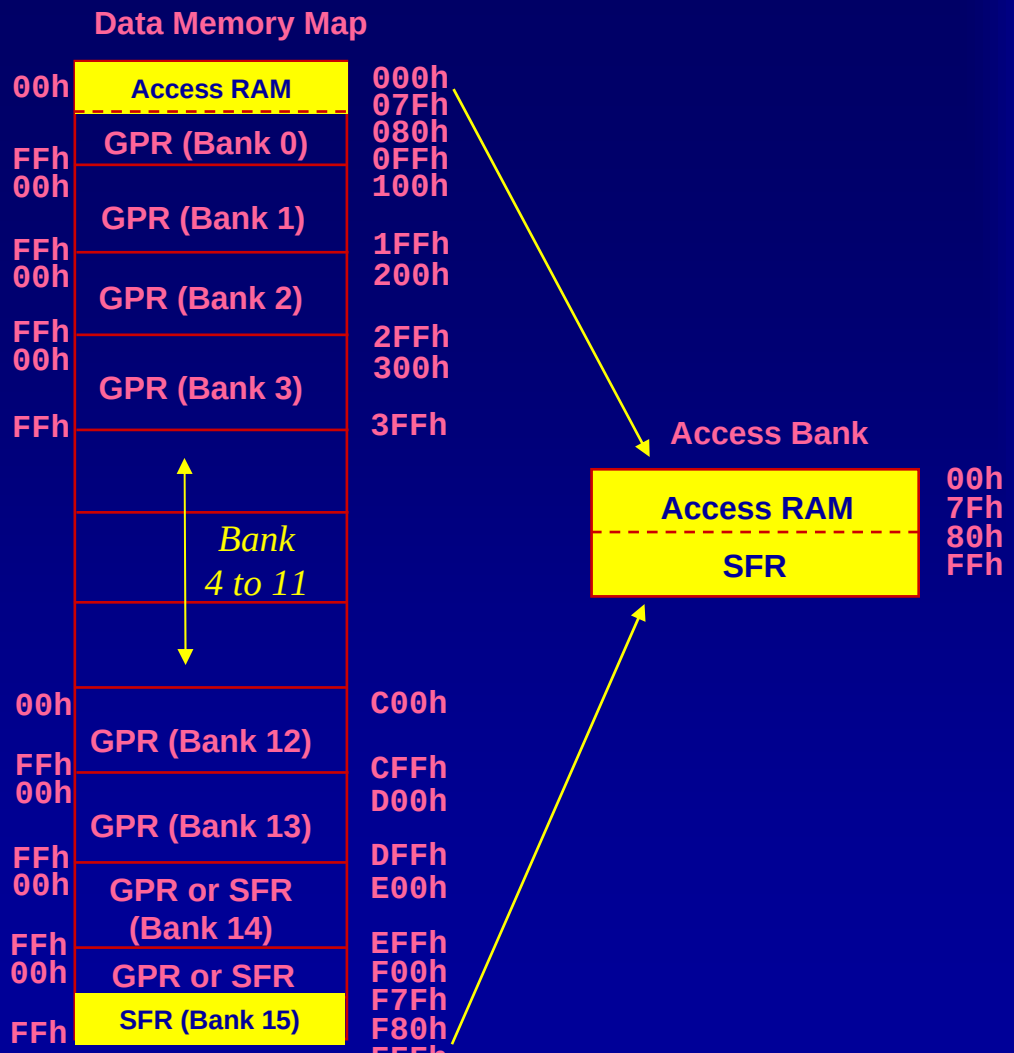
16-bit Instruction Format Example:

Op code <8>	Direct data addr <8>

PICmicro® PIC18 系列的架構

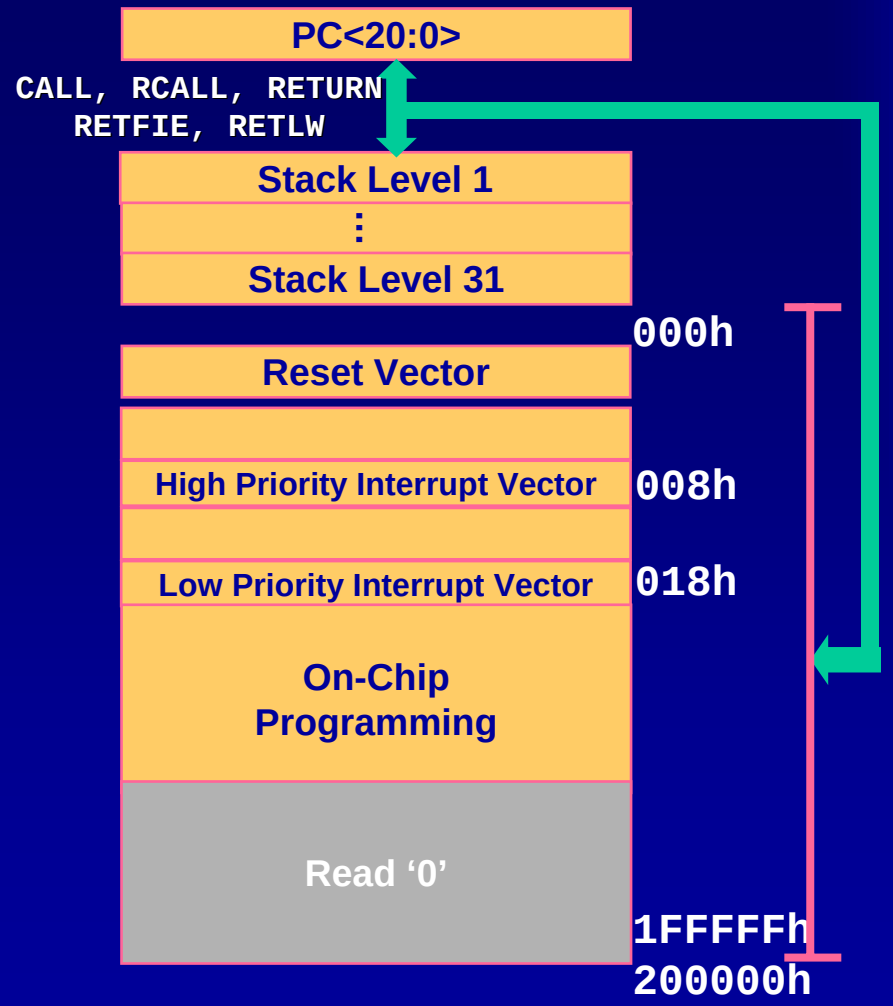
Data Memory

- 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



PICmicro® PIC18 系列的架構
Program Memory

- 最大可達 2M Bytes (21 bits) 的程式區間
- Reset Vector 位於位址 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

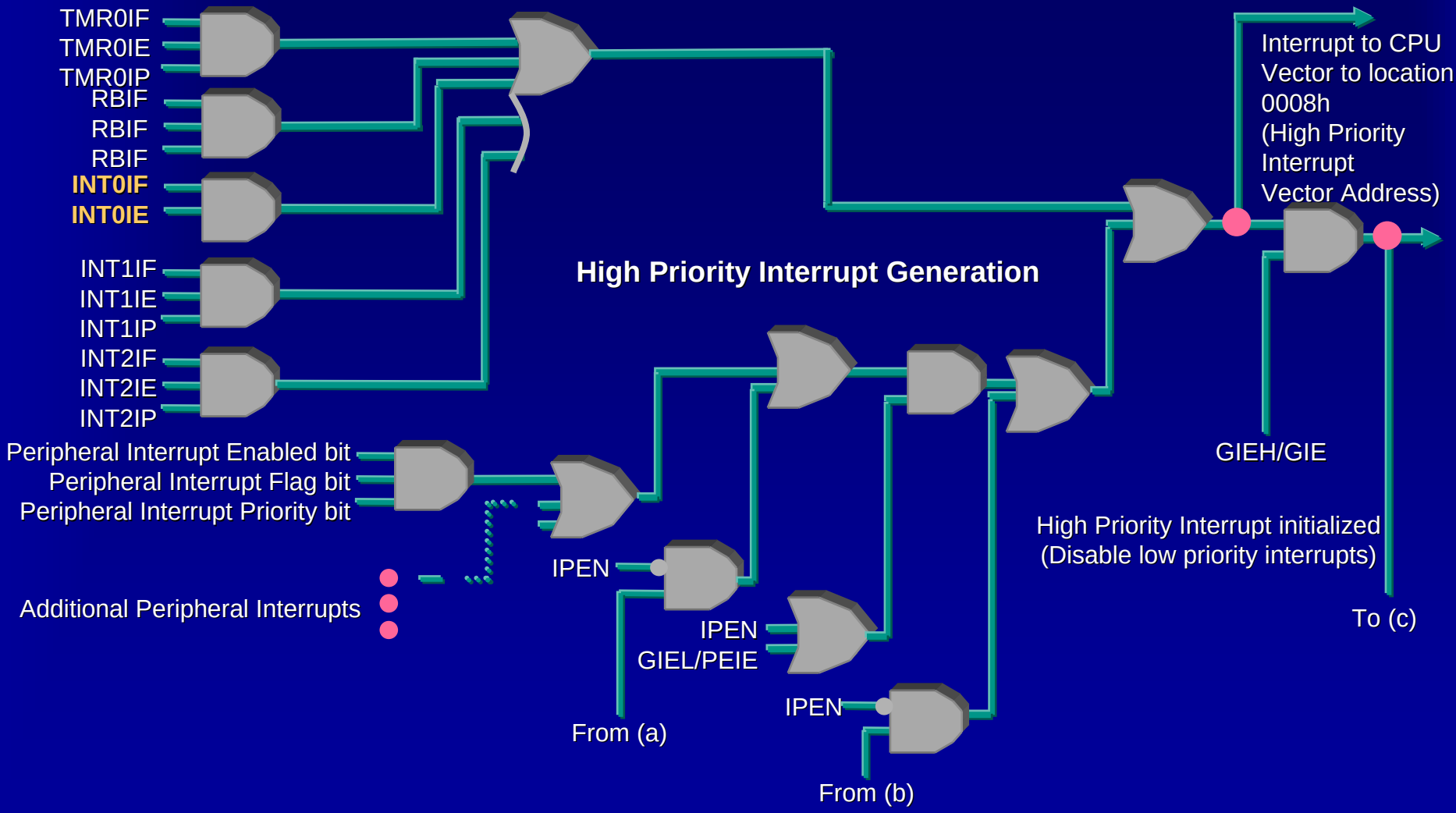


PICmicro® PIC18 系列的架構
Interrupt Overview

- 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 永遠爲高優先權外部中斷
- 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
- 中斷延遲固定爲 3 個 cycle
  - 容易以軟體來判斷中斷的經過時間

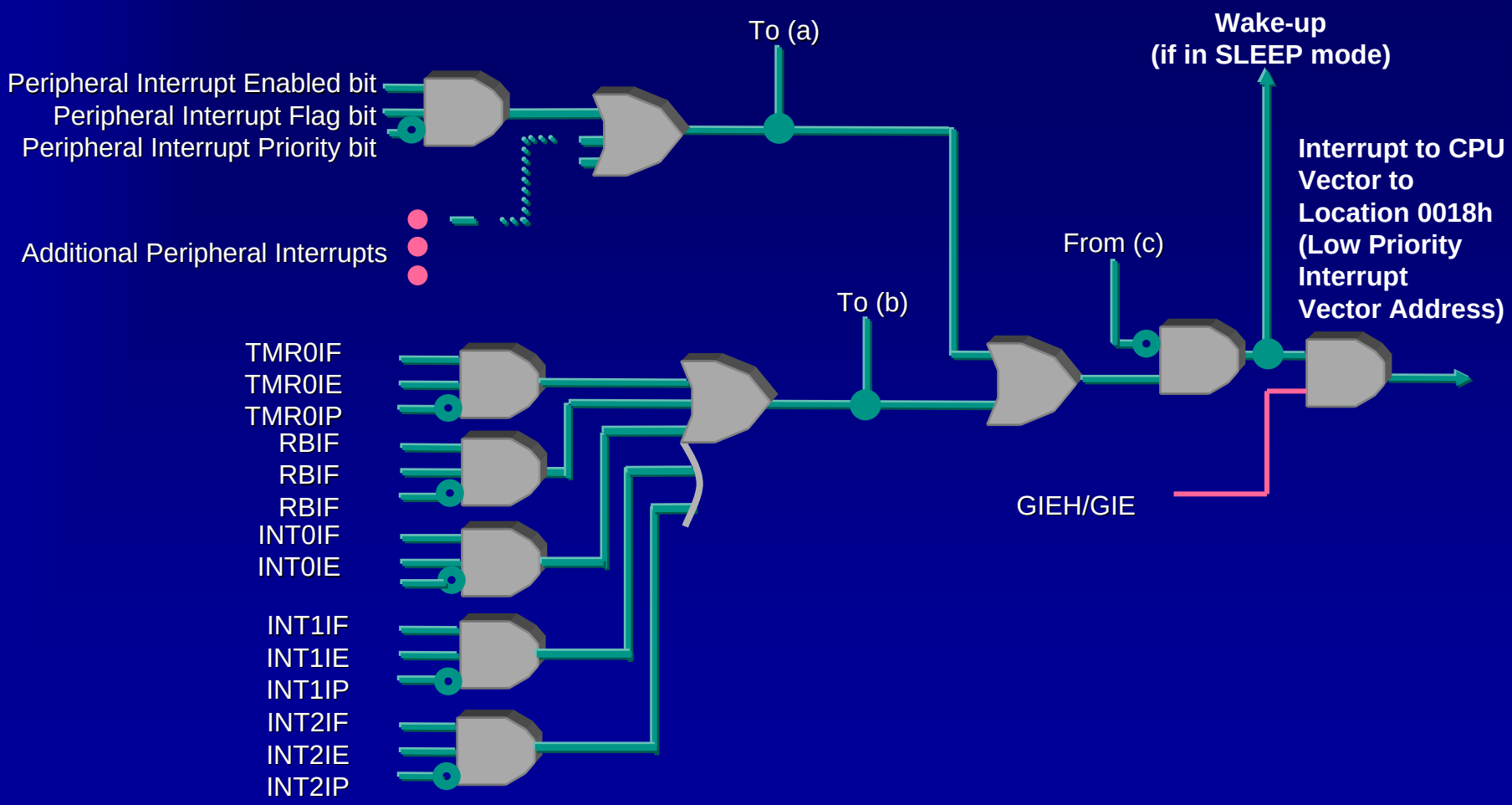
PICmicro® PIC18 系列的架構

Interrupt Logic (High Priority Level)



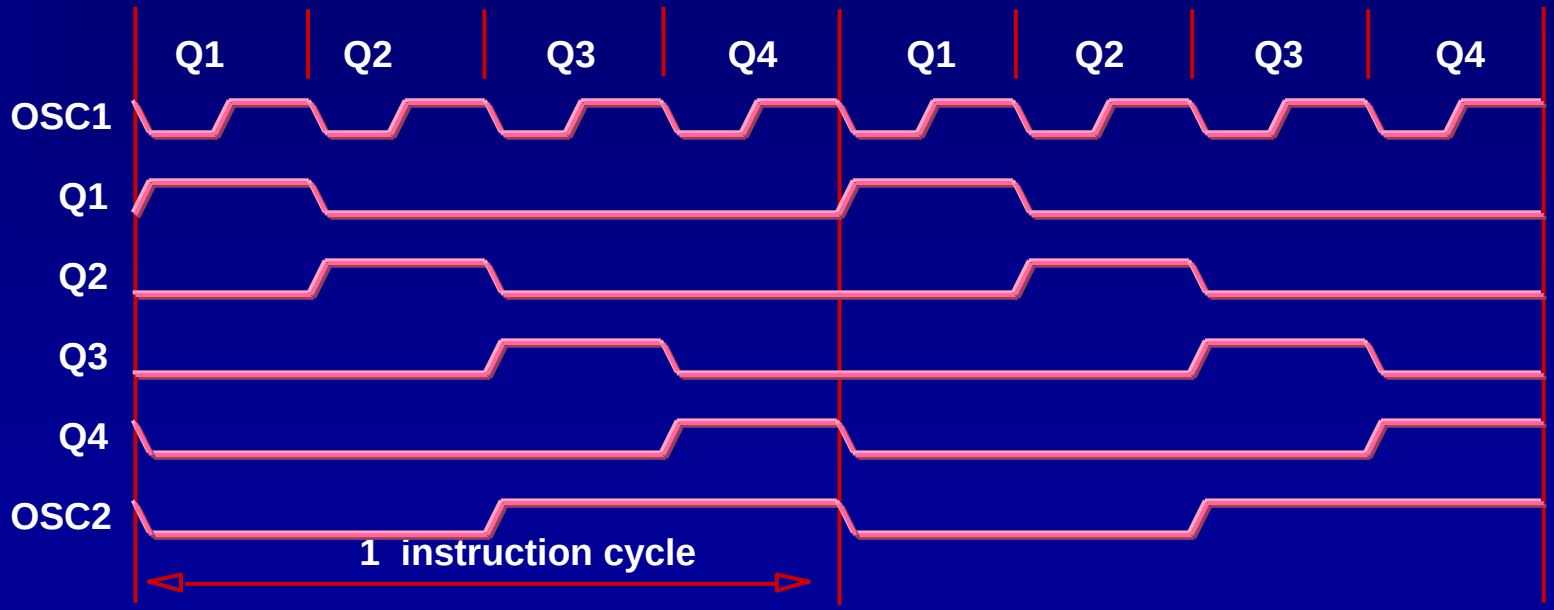
PICmicro® PIC18 系列的架構

Interrupt Logic (Low Priority Level)



PICmicro® 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，其指令周期爲 100 ns



PIC18FXXX

指令與定址模式

18Fxxxx 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

PICmicro® PIC18 系列的架構

立即值的操作

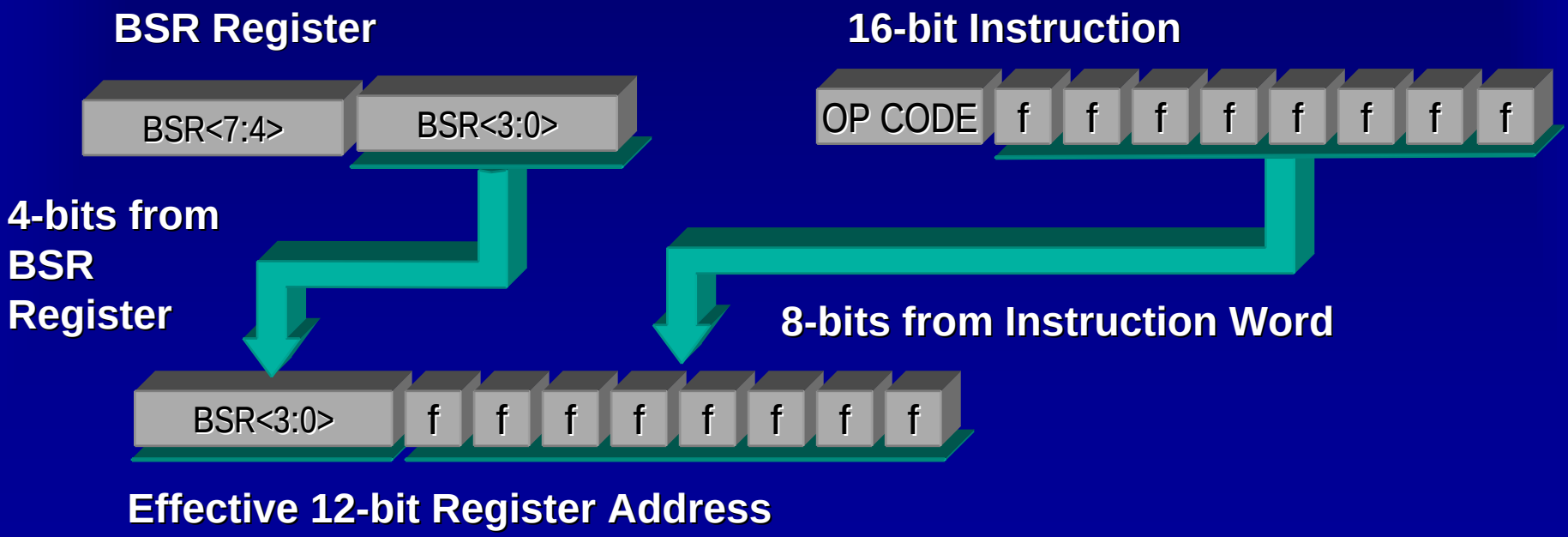
- **8-bit 的常數值 (literal value)** 被包含於程式碼之中
- 使用於 **literal instructions** , 例如
  - movlw, addlw, retlw, etc.

16-bit Instruction for Literal Instructions

PICmicro® PIC18 系列的架構

Data Memory: 直接定址 (Banked)

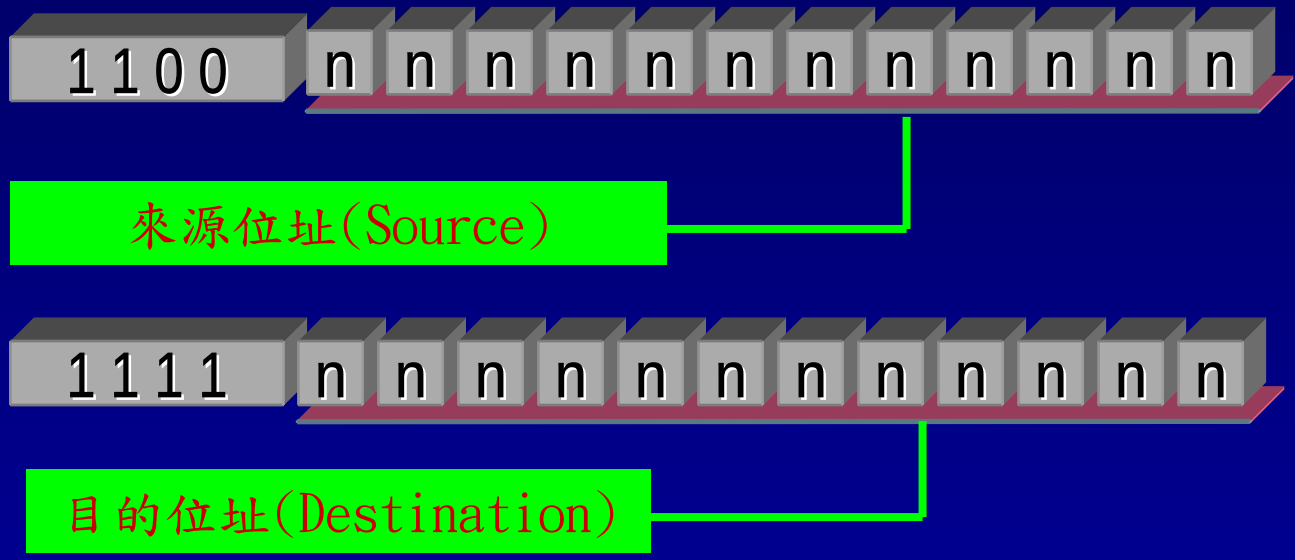
- Data Memory 總共有效位址爲 12 Bits (4K Bytes)
- 8-bit 的直接位址由指令中得到
- 其他高位的 4-bits 由 BSR<0..3> 獲得



PICmicro® PIC18 系列的架構

Data Memory：直接定址 (Non Banked)

- **MOVFF src, dest (記憶體對計憶體的資料搬移)**
  - 2 Word 的指令, 直接指定兩個 4K 的位址

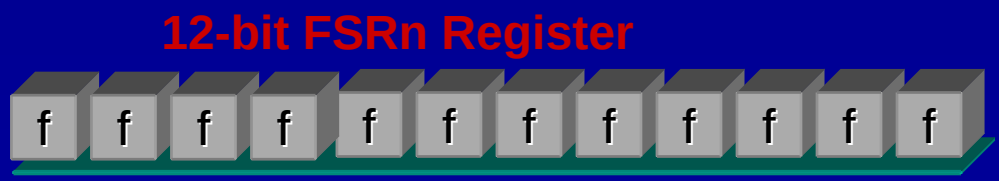


善用 Access BANK 或選擇 BANKED 的 Single Word 指令
可增加執行效能及佔用較少的程式記憶體

PICmicro® PIC18 系列的架構

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-Decremented : (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 爲偏移值相加後以產生新的索引指標
  - 保持不變 : (INDFx)



PICmicro® PIC18 系列的架構

Data Memory: 間接定址 (Indirect Addressing)

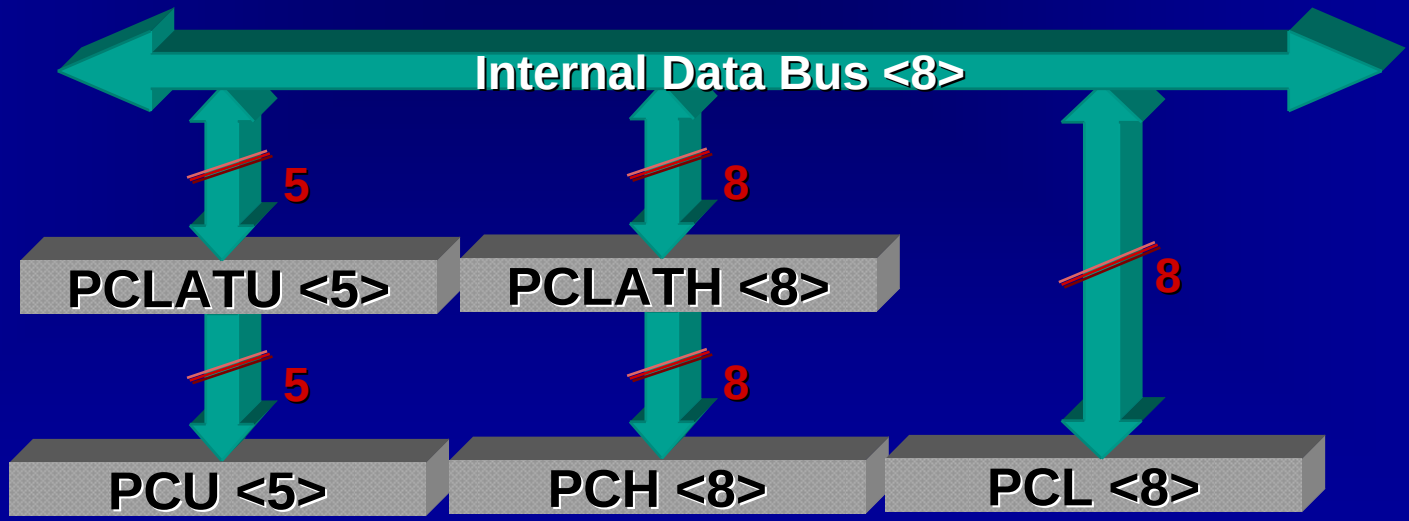
- 範例：清除由位址 0x000 to 0x5FF 的資料內容
  - 間接位址被寫入 FSRn 中
  - 每當 INDFn 被當成運算元(operand)， 實際上被操作的位址乃是由 FSRn 內容所指的位址
  - 類似 8051 的 @R0、@R1 定址法， 但功能更齊全

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

PICmicro® PIC18 系列的架構

PC 的相對定址使用方法

- 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



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 操做的指令爲 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單位元操作 (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 (1's complement)
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

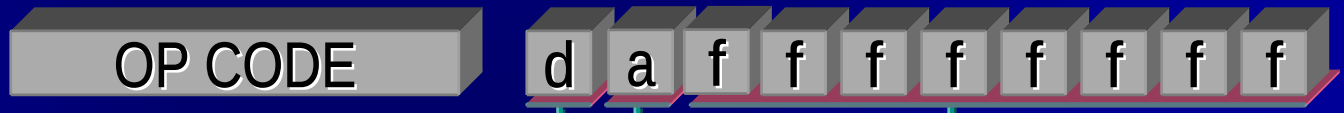
n = absolute address or address offset, k = literal value (8-bit), s = fast restore

PIC18系列 MCU 指令集
Byte-Oriented Operations

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

若 a=0 則 BSR=0x0 (f <= 0x07f) 或 BSR=0xf (f >= 0xf80)

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

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

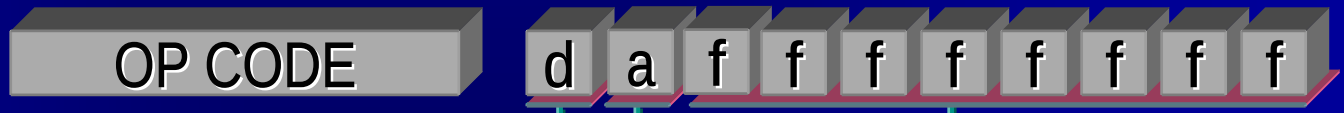
Example:
ADDWF 0, 0
ADDWF f, d, a

PIC18系列 MCU 指令集
Byte-Oriented Operations (cont.)

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

若 a=0 則 BSR=0x0 (f <= 0x07f) 或 BSR=0xf (f >= 0xf80)

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

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

PIC18系列 MCU 指令集
Bit-Oriented Operations (cont.)

Bit-Oriented Operations

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

若 a=0 則 BSR=0x0 (f <= 0x07f) 或 BSR=0xf (f >= 0xf80)

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 STATUS, C, 0
BTFSC 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

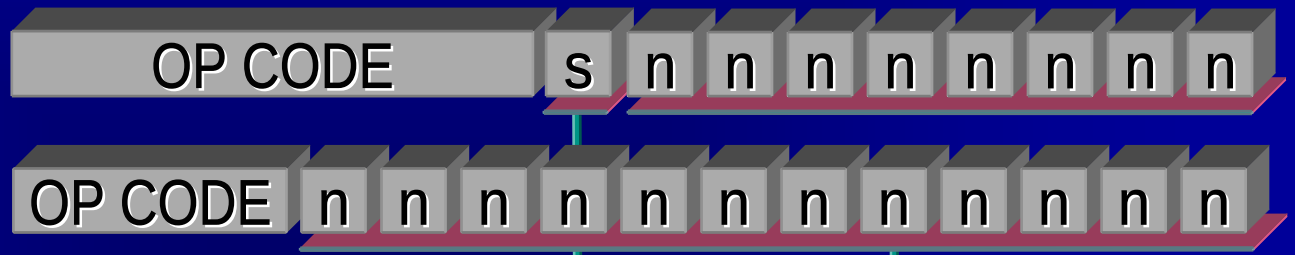
MOVLW CONST

用來操作 FSR 的16-bit， 2 Word 指令 “LFSR”

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
RETFIE	s
RETLW	k
RETURN	s

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 (cont.)

Control Operations	
CLRWDT	-
DAW	-
NOP	-
POP	-
PUSH	-
RESET	-
SLEEP	-

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)

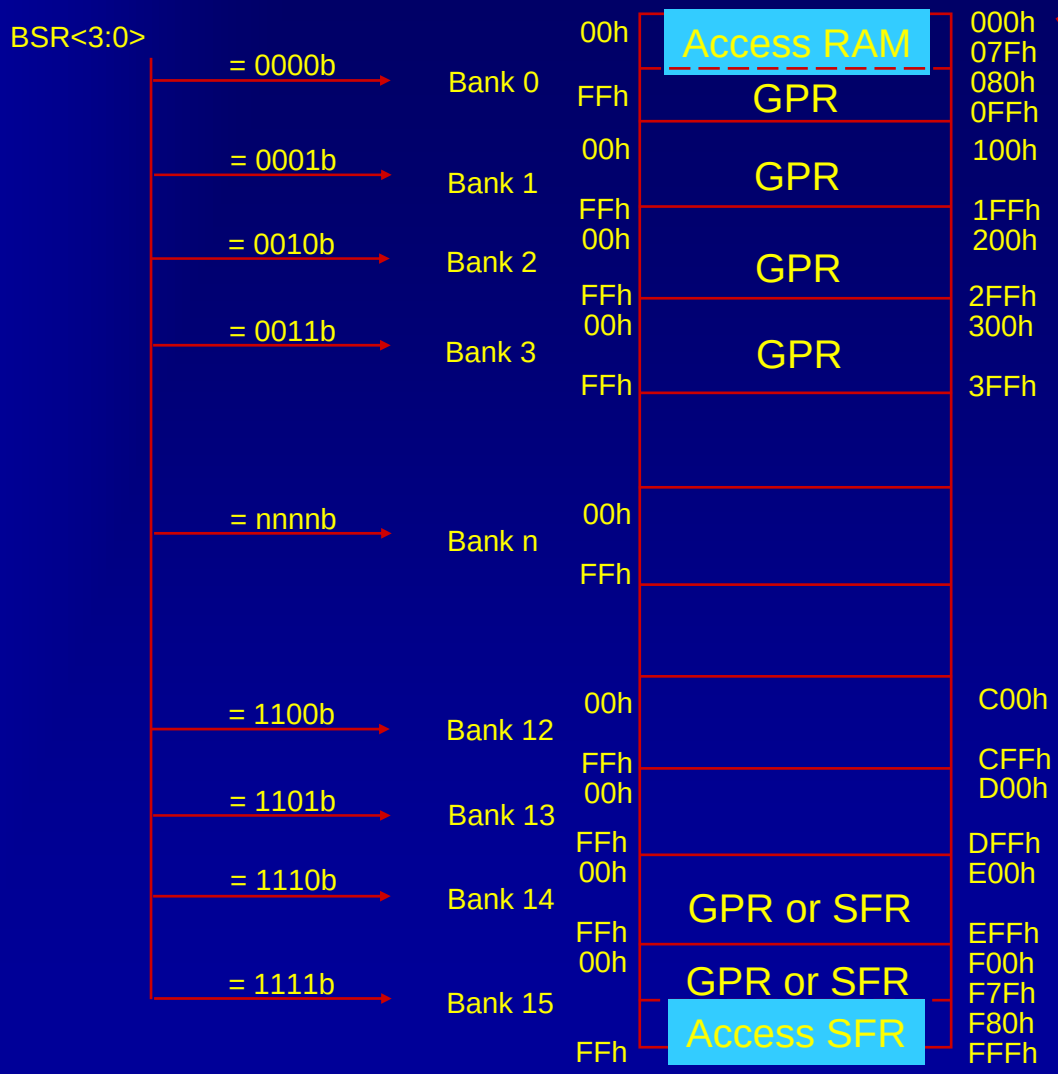
nn = 01 *+ (查表後指標加一)

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



Movf 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	00h – 7Fh
Access SFR	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 爲操作單位
  - 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

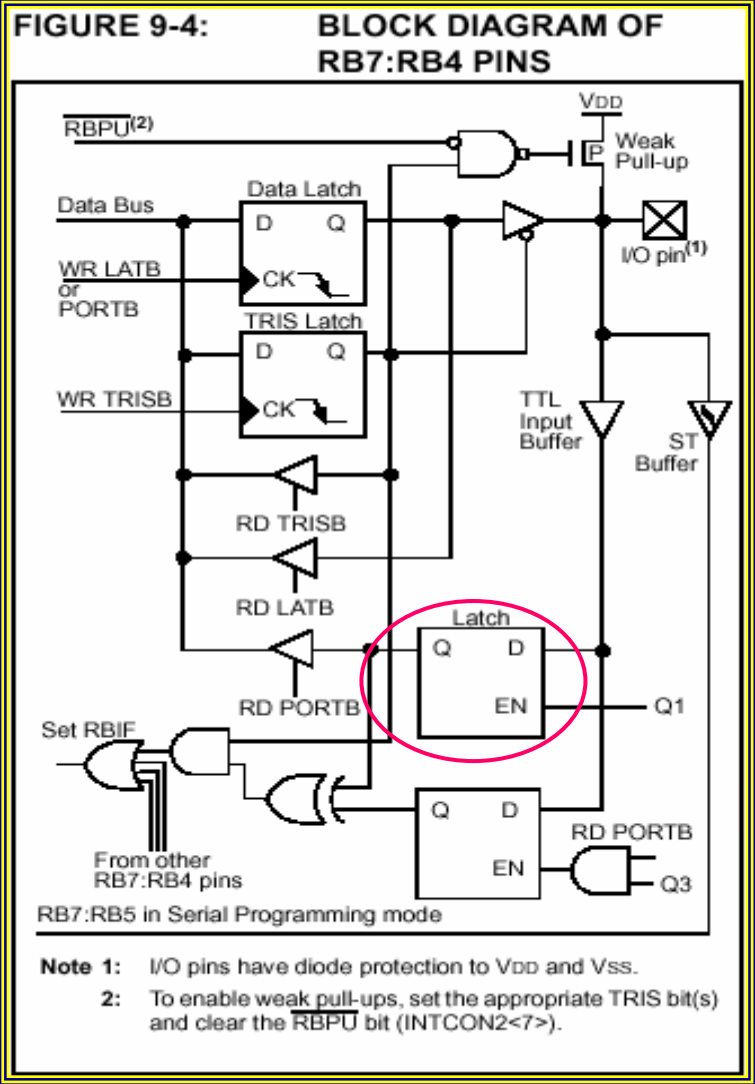
PIC18FXXX 加強的部份
資料指標(Data Pointer)

- 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)
  - INDFn : 直接操作指標所指的內容
  - POSTINCn : 操作指標所指內容後將指標加一
  - POSTDECn : 操作指標所指內容後將指標減一
  - PREINCn : 將指標先加一後再對指標所指位址操作
  - PLUSWn : 將 WREG 的內容作為 Offset 值與 FSRn 相加後以其結果為指標來存取記憶體

PIC18FXXX

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 Data Latch，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

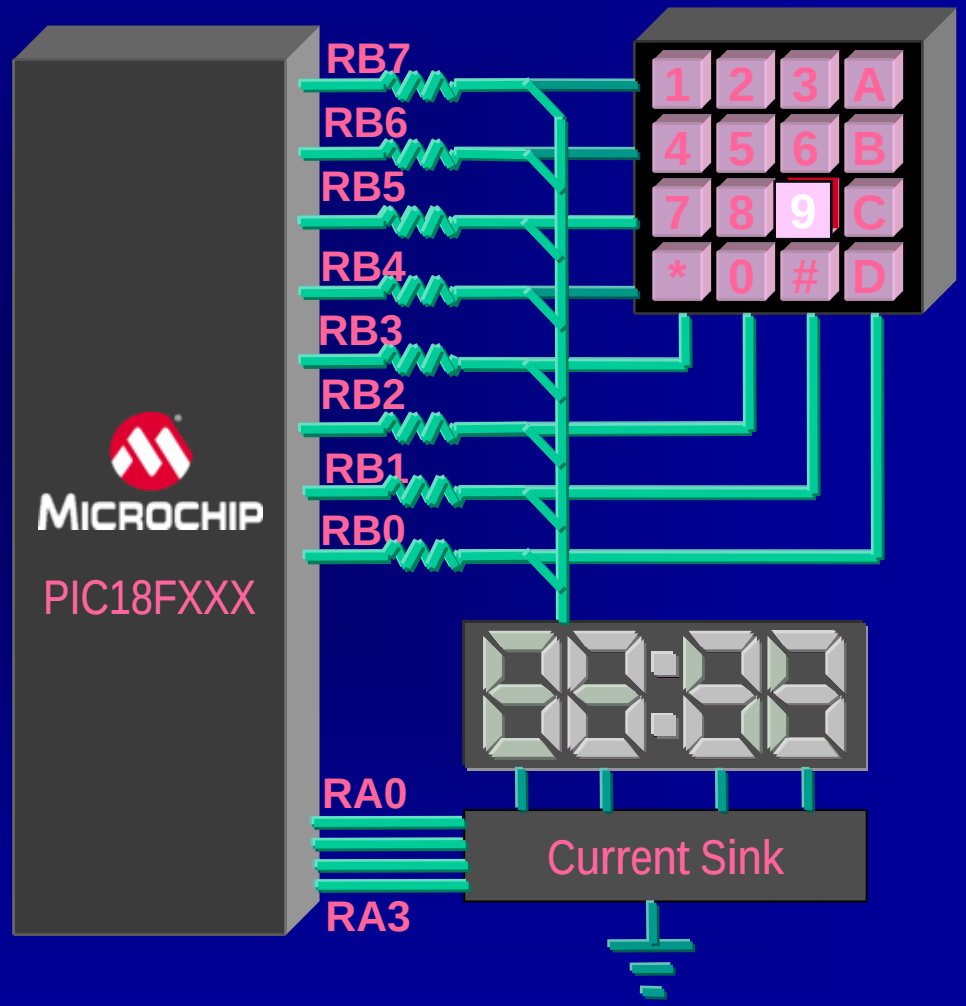
PIC18 系列的周邊 : I/O Ports

- 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
  - 資料閂鎖（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

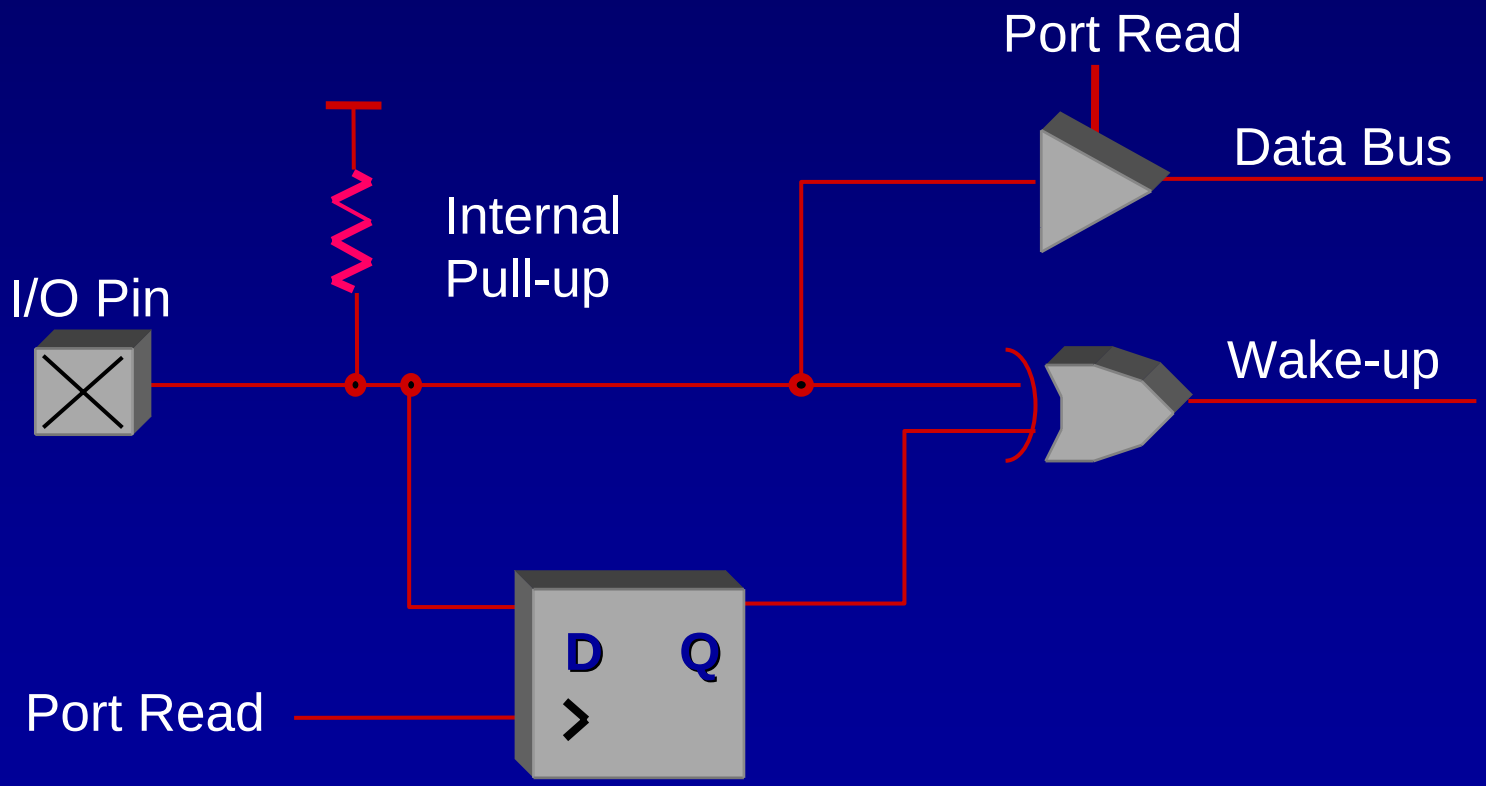
- 利用最少的外加零件及軟體來完成：點亮 4 個七節顯示器並掃瞄一個 4 X 4 的矩陣鍵盤。
- RB0 - RB7 為顯示字形的輸出，RA0 - RA3 控制欲被點亮的七節顯示器位置。
- 每隔 5mS 點亮一個七節顯示器，共 20mS 完成一周。
- 在點亮七節顯示器一周後，改變 RA0~RA3 為輸入關閉七節顯示器 100uS 以掃瞄鍵盤。
- RB0 - RB3 為輸出 pins，RB4 - RB7 為輸入 pins（ 內含 pull-up電阻 ）



PIC18 系列的周邊 I/O Ports

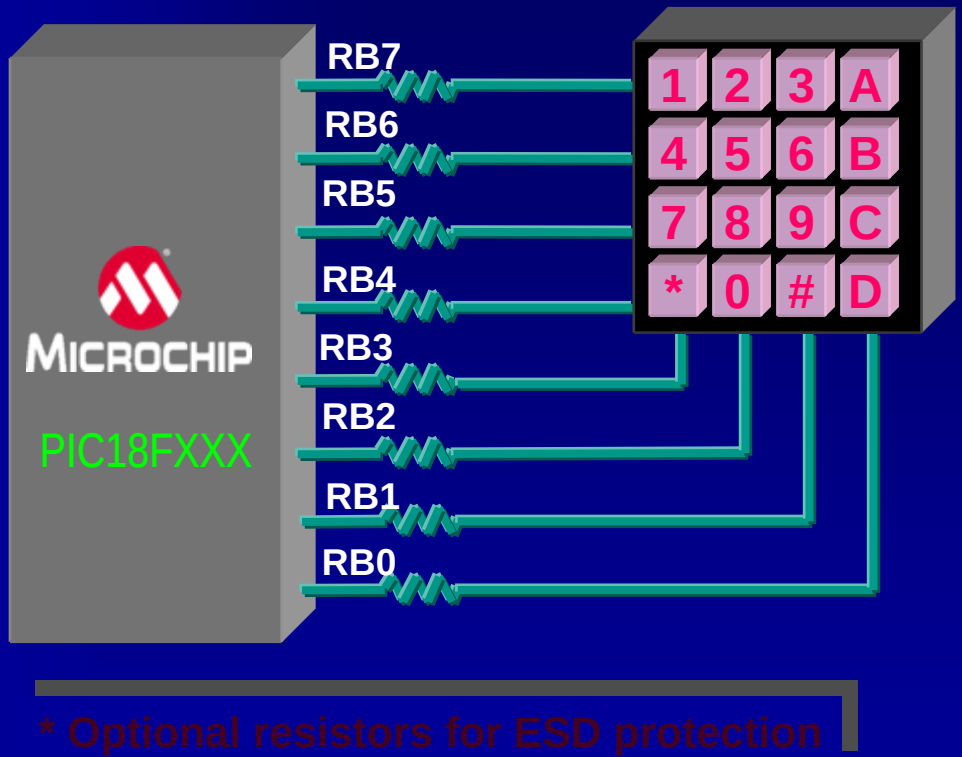
PORTB 的特殊功能：Interrupt on Change

- **PORTB 的 RB4 至 RB7 可因爲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



- **RB4 - RB7** 內有 **pull-up** 電阻，設定爲輸入腳
- **RB0-RB3** 爲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**
- 對鍵盤進行掃瞄動作讀取按鍵值

MICROCHIP

開發工具介紹

APP001多功能實驗版

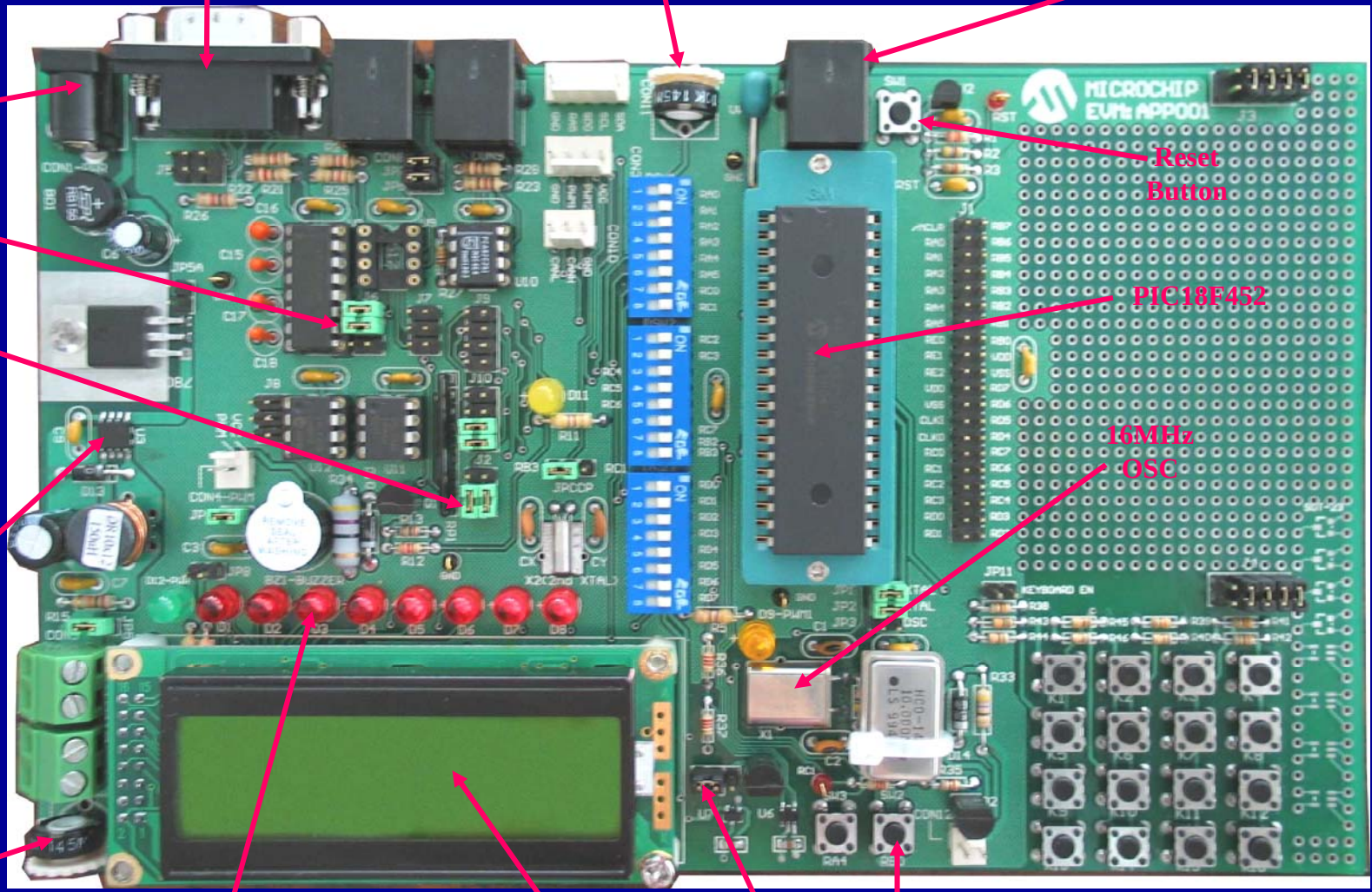
MPLAB IDE v7.xx

MPLAB ICE2000

MPLAB ICD2

MPASM & MPLINK

APP001 實驗板


RS-232
A/D VR
ICD2
Connecter
DC 9V
Input
Reset
Button
J6
(RS-232)
PIC18F452
J2
(32768Hz)
16MHz
OSC
+5V
S.P.S.
LCD
亮度調整
PORTD LED
2 X 16 LCD Module
J9
(LED)
Button

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，I^2C EEPROM， I^2C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)

MPLAB-IDE 功能介紹 (v7.xx)

- 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
- 支援 VDI

- 硬體除錯工具：
  - MPLAB-ICE 4000
  - MPLAB-ICE 2000
  - MPLAB-ICD2
- 程式燒錄工具：
  - PRO MATE - II
  - PICSTART Plus
  - MPLAB-ICD2
  - PICKIT 1
  - PICKIT 2

MPLAB - ICE2000

連接 Host 到 Pod 的Cable

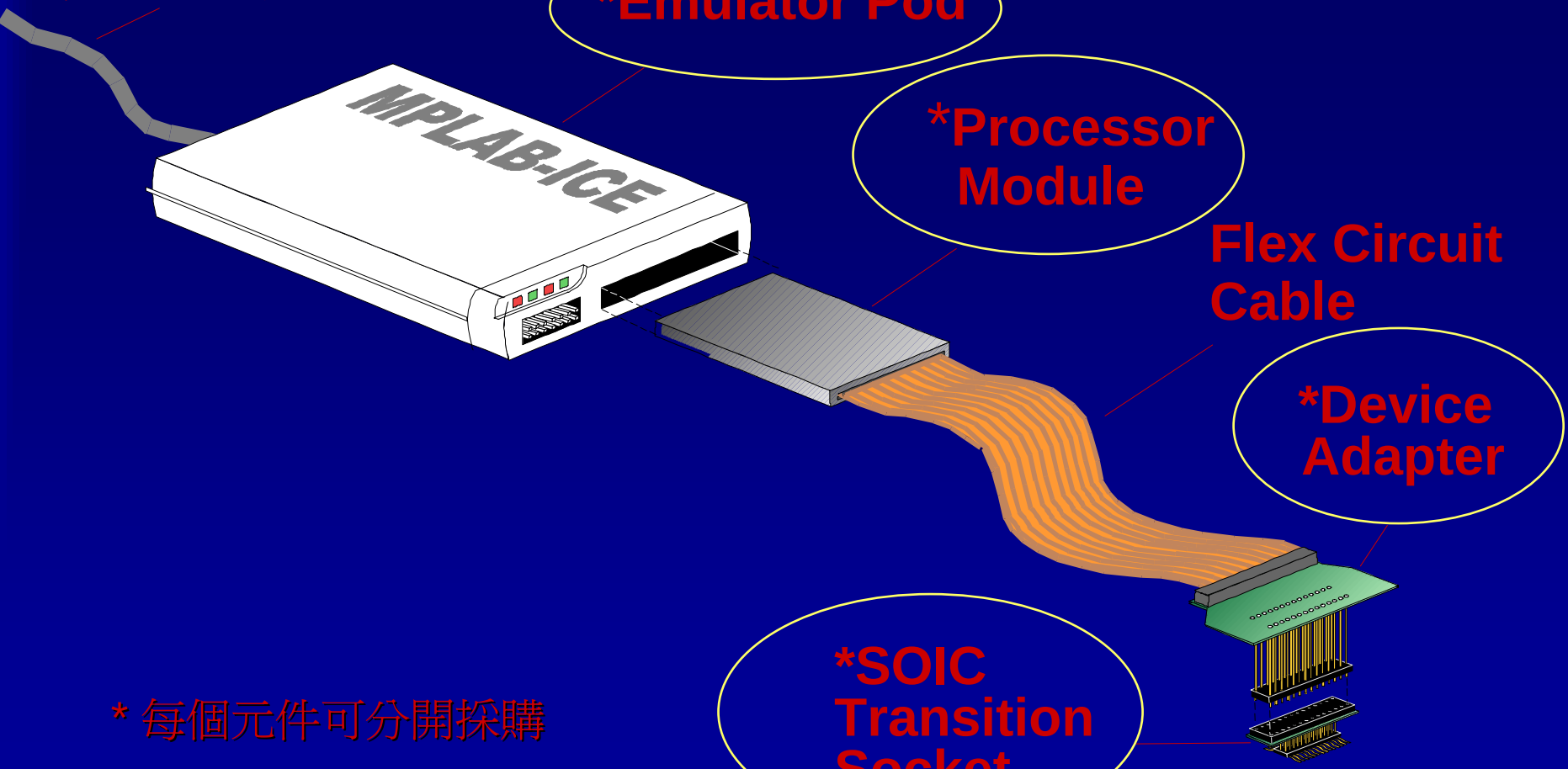
*Emulator Pod

*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 25/33 MHz (PIC18FXXX)**
  - 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

- 與 PC 通訊介面支援 USB 及 RS - 232
- 支援 PIC16F 系列具有 ICD除錯功能的 MCU
- 支援所有 PIC18F 、PIC24F 、dsPIC30F 及 dsPIC33F 系列
- 可設定單個或數個(最多4點)中斷點， 可單步執行以及進行變數的觀察
- 32K Bytes (PIC18F452) 的程式可於 3 秒內燒錄完成

MPLAB IDE 硬體支援

硬體設備	MPLAB v7.xx	MPLAB v5.xx
MPLAB-ICE4000	Yes	No
MPLAB-ICE2000	Yes	Yes
PICMASTER	No	Yes
ICEPIC	No	Yes
MPLAB-ICD	No	Yes
MPLAB-ICD2	Yes	Yes
PROMATE	Yes	Yes
PROMATE-II	Yes	Yes
PICSTART Plus	Yes	Yes

MPLAB™ v7.xx
整合式的發展環境

內含多功能程式編輯器

原始檔案程式偵錯功能

單一系統專案管理模式

語言工具	軟體模擬	硬體模擬器	程式燒錄器
MPASM 編譯器	MPLAB-SIM 軟體模擬器	MPLAB-ICE 2000 4000	PICSTART® Plus
MPLINK 連結器	dsPIC 軟體模擬器	PICMASTER (V5.xx)	PRO MATE®
MPLAB Cxx dsPIC C 編譯器		MPLAB ICD2	

MPASM 的功能

- 單一組合語言的架構
  - 檢查程式，並找出錯誤的語法及指令
  - 安排常數值、程式及各變數的位址
  - 將組合語言直接翻譯成 **HEX** 格式的機械碼

- 多組合語言的架構
  - 檢查程式，並找出錯誤的語法及指令
  - 安排常數值
  - 保留變數位址、程式相對及絕對位址給**MPLINK**安排
  - 將組合語言翻譯成可重新定位址的 **OBJ** 檔案
  - **OBJ** 檔需經 **MPLINK** 連結與安排位址後才會產生 **HEX** 格式的機械碼

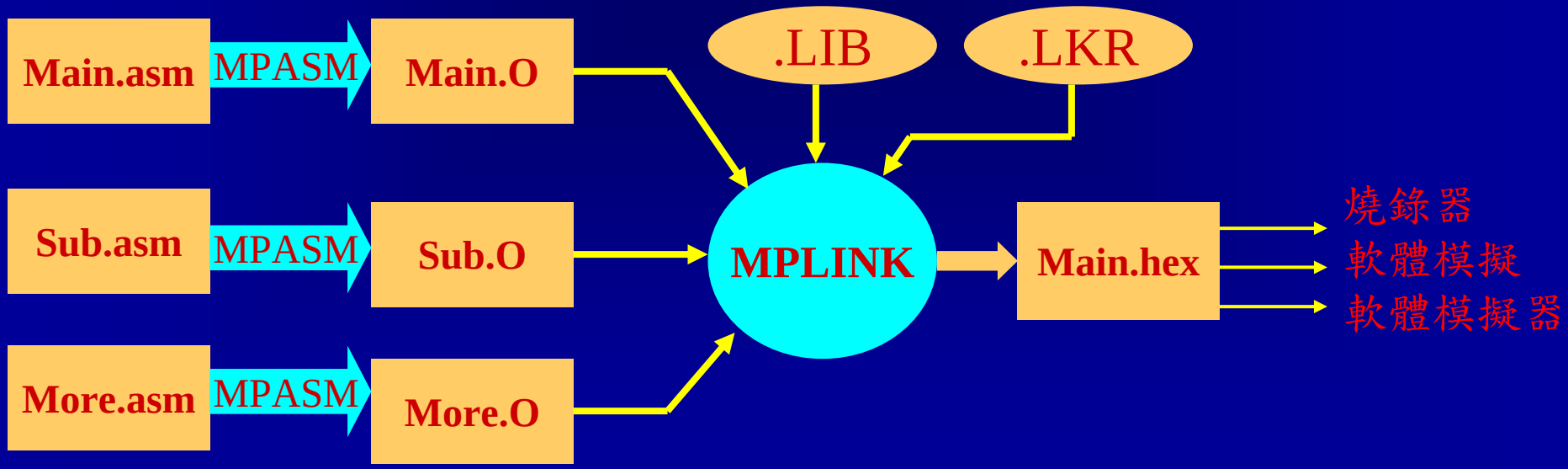
MPLINK 的功能

- 什麼是MPLINK?
  - MPLINK 是將組合語言的組譯(Assembler) 或 C- Compiler 所產生的 obj 檔加以連結並排定各程式及變數位址後，輸出一個可執行的 hex 檔
- MPLINK 能作做什麼？
  - 安排實際位址給程式 (CODE) 及資料 (RAM)
  - 產生可執行擋 (HEX)
  - 安排堆疊位址及深度給 MPLAB C18
  - 提供 COD 檔以利程式偵錯
  - 讓程式更容易模組化
  - 連結資料庫 (Library)

MPASM & MPLINK
組譯流程

- 單一原始組合語言檔 (.ASM FILE → .HEX FILE)

- 多原始組合語言檔 (.ASM FILE → .OBJ →.HEX FILE)



使用 MPLAB IDE v7.xx

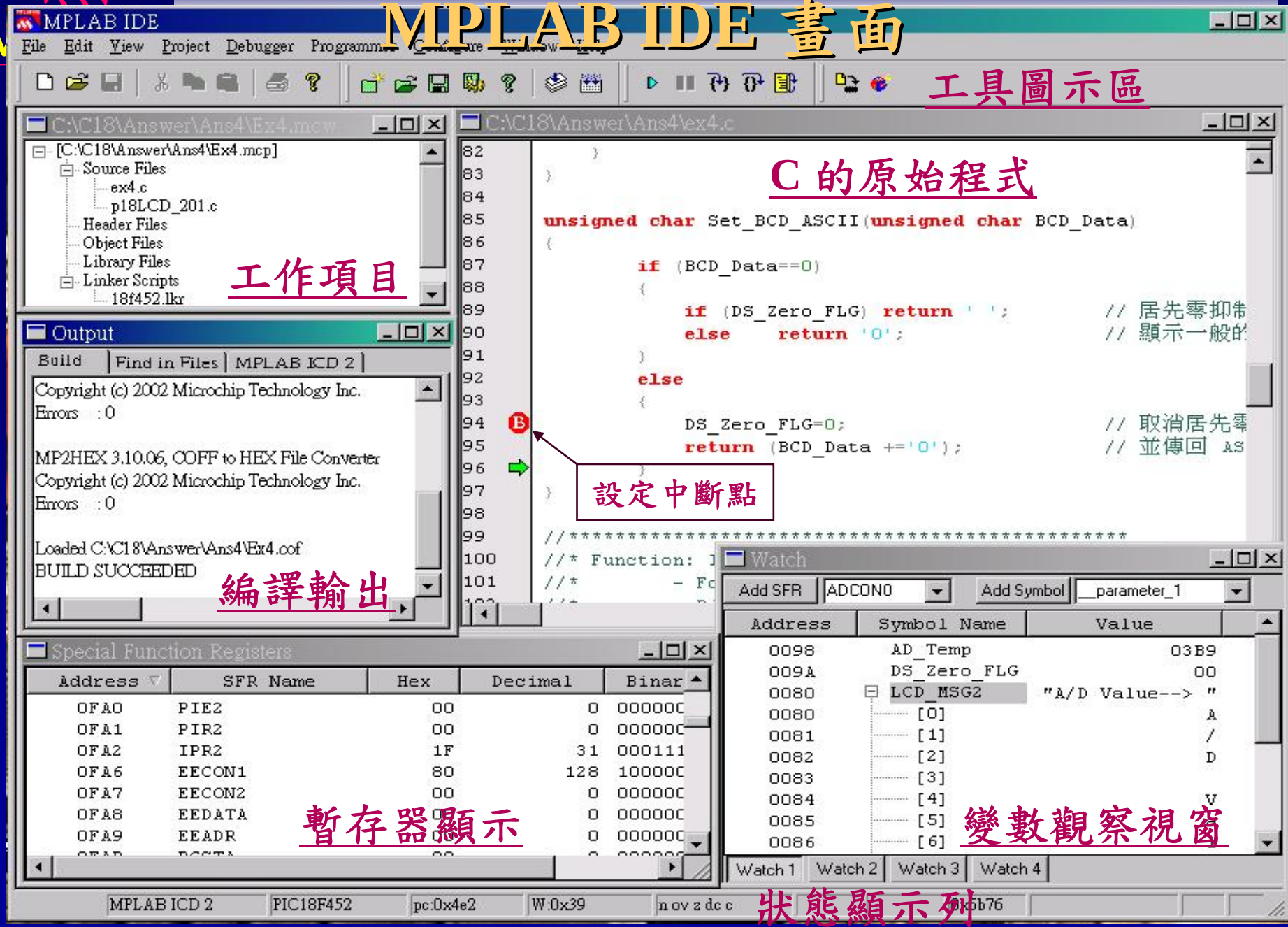
如何使用 MPLAB IDE V7.xx

- 詳細使用說明請詳細閱讀：
  - **MPLAB v6.10 中文使用手冊**
    - **www.microchip.com.tw**
- 安裝 **ICD2 USB** 驅動程式的說明檔案：
  - **C:\Program Files\Microchip\MPLAB IDE\ICD2\Drivers**
    - **Ezicd2.htm**
- 組譯工具 **(MPASM)** 與 **C compiler** 不知如何使用：
  - **MPLAB v6.10 中文使用手冊**
  - **MPLAB IDE Quick Start Guide**
    - **http://ww1.microchip.com/downloads/en/DeviceDoc/51281E.pdf**

安裝 MPLAB IDE V7.xx

- 安裝 MPLAB IDE v7.xx
  - MPLAB IDE v7.xx 安裝後總檔案大小超過 100M
  - 請參閱 MPLAB IDE v6.10 使用手冊 ，第 2-2 章的說明
  - 之前已有安裝舊版的 MPLAB IDE v6.xx ，需先移除
  - 別忘了，還要安裝 ICD2 的 USB 驅動軟體
  - 如果有使用 ICE2000 還要再安裝驅動程式

MPLAB IDE 畫面


MPLAB IDE
File Edit View Project Debugger Programmer Configure Window Help
工具圖示區
C:\C18\Answer\Ans4\Ex4.mcw
[C:\C18\Answer\Ans4\Ex4.mcp]
Source Files
ex4.c
p18LCD_201.c
Header Files
Object Files
Library Files
Linker Scripts
18f452.lkr
工作項目
C:\C18\Answer\Ans4\ex4.c
C 的原始程式
unsigned char Set_BCD_ASCII(unsigned char BCD_Data)
if (BCD_Data==0)
if (DS_Zero_FLG) return ' ';
else return '0';
else
DS_Zero_FLG=0;
return (BCD_Data +='0');
設定中斷點
Output
Build
Find in Files
MPLAB ICD 2
Copyright (c) 2002 Microchip Technology Inc.
Errors : 0
MP2HEX 3.10.06, COFF to HEX File Converter
Copyright (c) 2002 Microchip Technology Inc.
Errors : 0
Loaded C:\C18\Answer\Ans4\Ex4.cof
BUILD SUCCEEDED
編譯輸出
Special Function Registers
Address SFR Name Hex Decimal Binar
OFA0 PIE2 00 0 00000C
OFA1 PIR2 00 0 00000C
OFA2 IPR2 1F 31 000111
OFA6 EECON1 80 128 10000C
OFA7 EECON2 00 0 00000C
OFA8 EEDATA 0 0 00000C
OFA9 EEADR 0 0 00000C
暫存器顯示
Watch
Add SFR ADCON0
Add Symbol __parameter_1
Address Symbol Name Value
0098 AD_Temp 03B9
009A DS_Zero_FLG 00
0080 LCD_MSG2 "A/D Value--> "
0080 [0] A
0081 [1] /
0082 [2] D
0083 [3]
0084 [4] V
0085 [5]
0086 [6]
變數觀察視窗
Watch 1
Watch 2
Watch 3
Watch 4
MPLAB ICD 2
PIC18F452
pc:0x4e2
W:0x39
n ov z dc c
狀態顯示列

了解 MPLAB IDE V7.xx
相關檔案位置

- C:\Program Files\Microchip
  - ..\MPLA IDE\Core (次目錄)
    - MPLAB IDE 執行檔位置
  - ..\MPLA IDE\ICE2000\Drivers (次目錄)
    - ICD2000的 驅動程式
  - ..\MPLA IDE\ICD2\Drivers (次目錄)
    - ICD2 USB 的驅動程式
  - ..\MPLAB C30 (次目錄)
    - dsPIC30F C30 Compiler
  - ..\MPLAB ASM30 Suite (次目錄)
    - MPLAB IDE 組譯工具， inc 檔案，lkr檔案 及 範本程式

啟動 MPLAB IDE v7.xx

- 有關 **MPLAB IDE v6.xx** 的基本使用方式，請參考 **MPLAB IDE v6.10** 中文使用手冊第三章的說明。

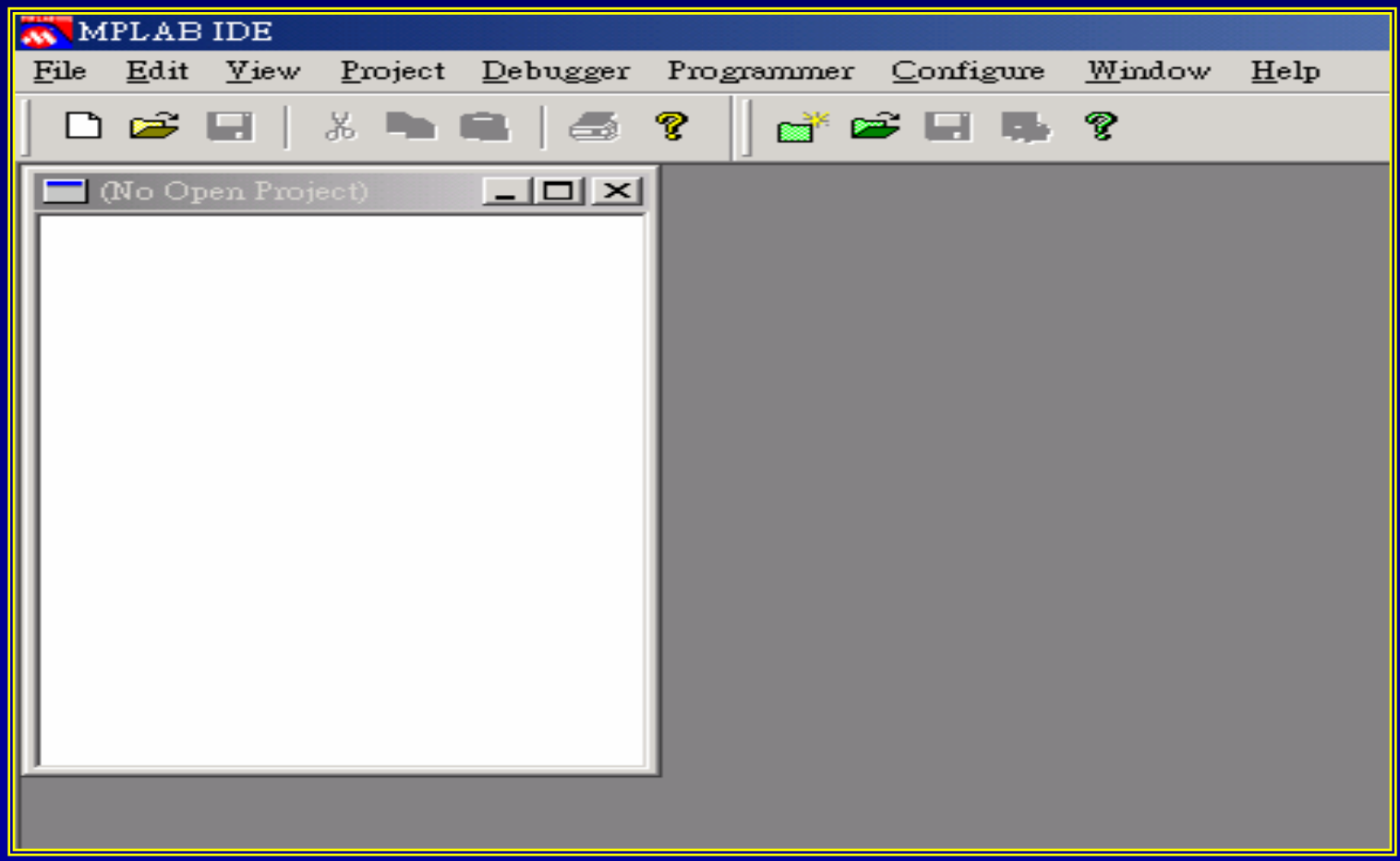
- 使用 **MPLAB IDE** 爲發展工具時，必須先建立一個專用的 **Project** ，建立 **Project** 有一定的程序不可以混淆。

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

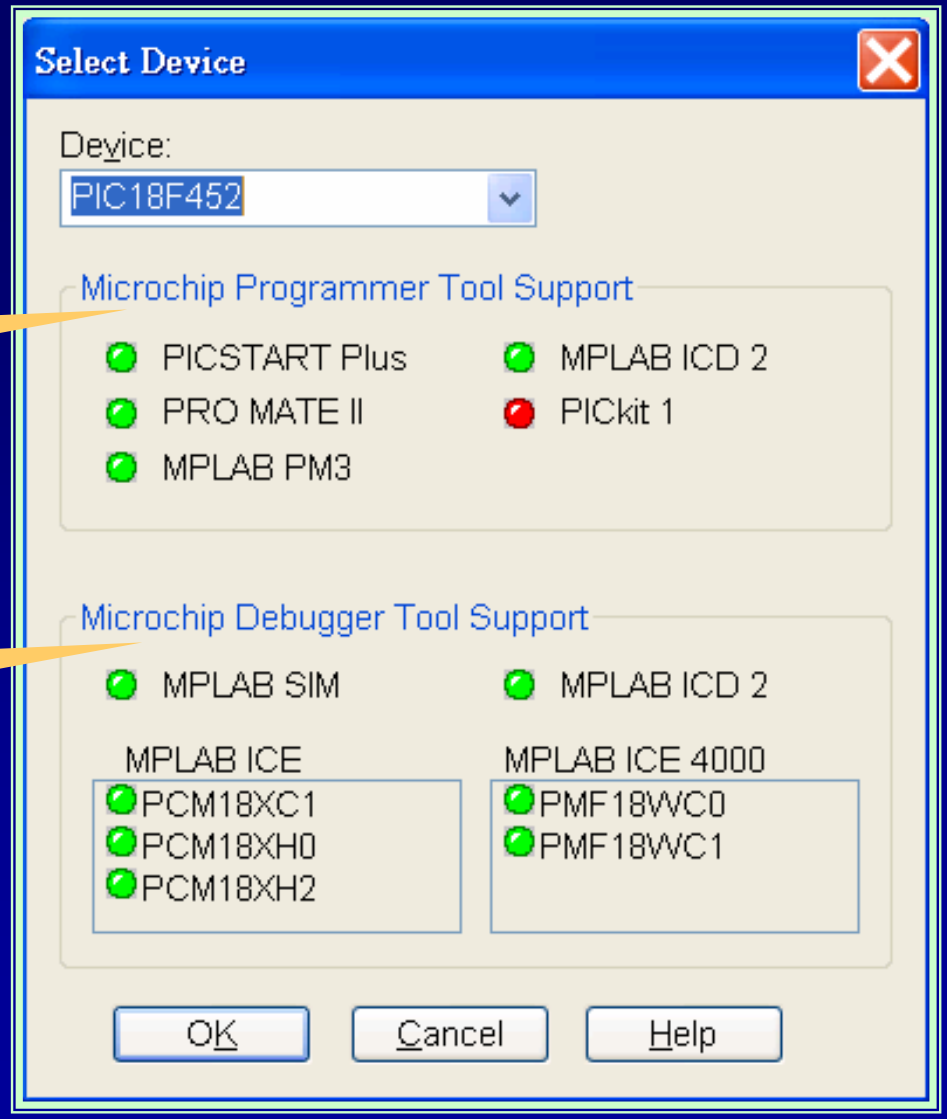


練習：建立新的專案

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

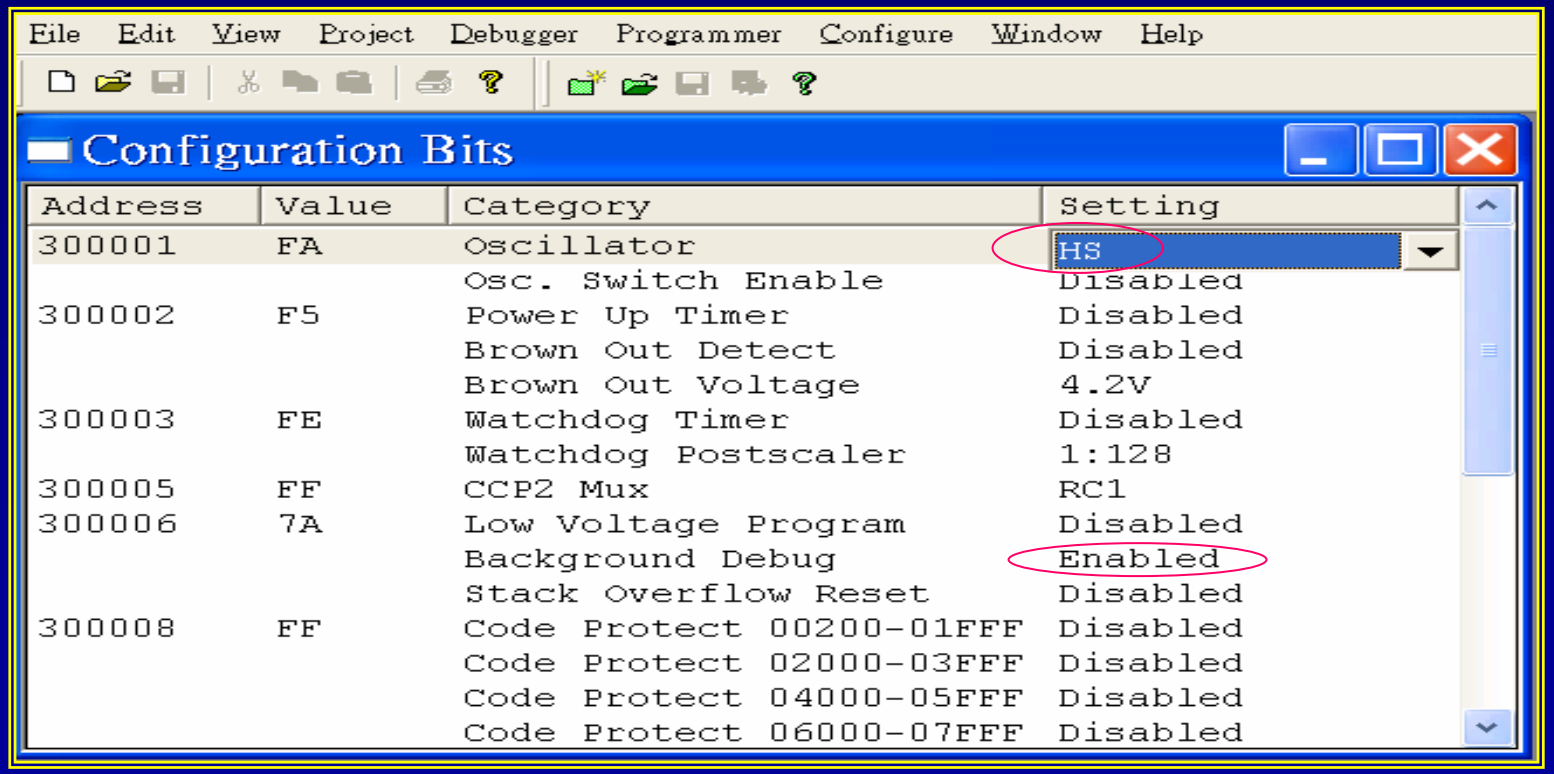
目前所選擇的元件所支援的燒錄器

目前所選擇的元件所支援的發展工具



練習：建立新的專案

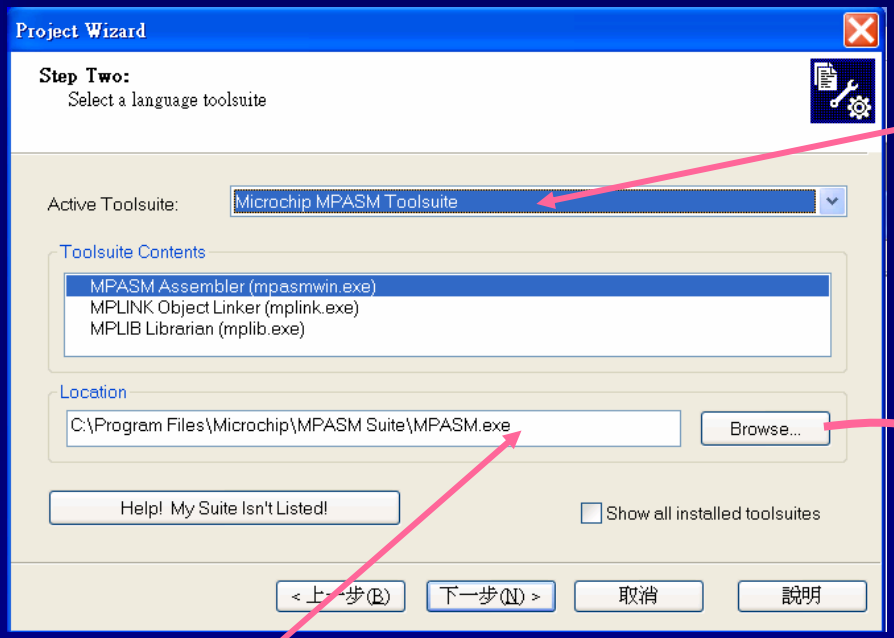
- 使用 “Configure → Configuration Bits” 來設定 IC 運作模式
  - 除 Background Debug 設定為 Enabled 以便使用 MPLAB ICD 2
  - 其他選項，均設為 Disable



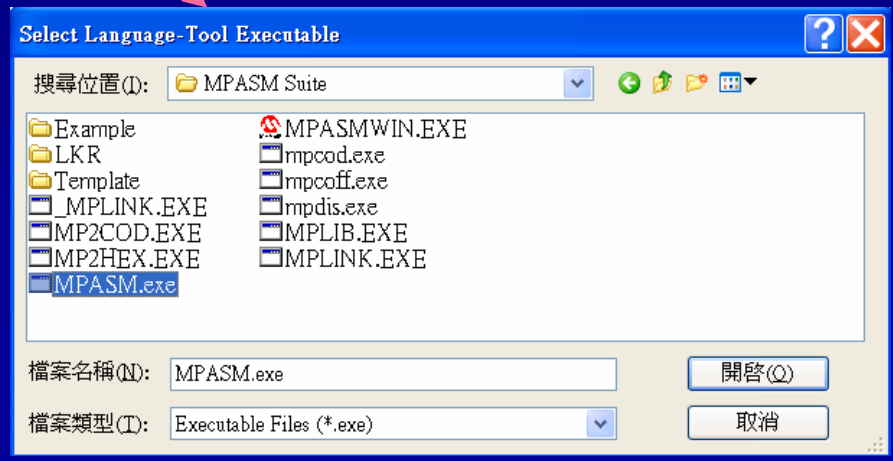
練習：建立新的專案

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

練習：建立新的專案



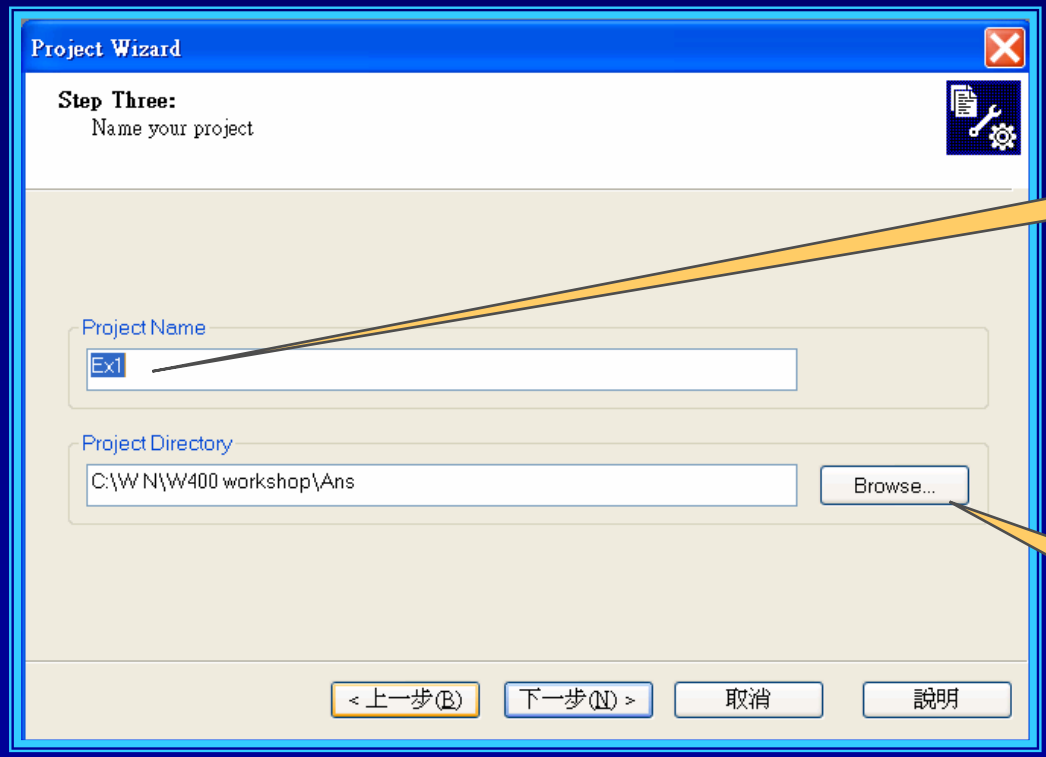
- 設定語言工具
- 也可以在 **“Project → Set Language Toolsuite”** 來設定使用 **MPASM** 來組譯



- 設頂語言工具執行路徑
- 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

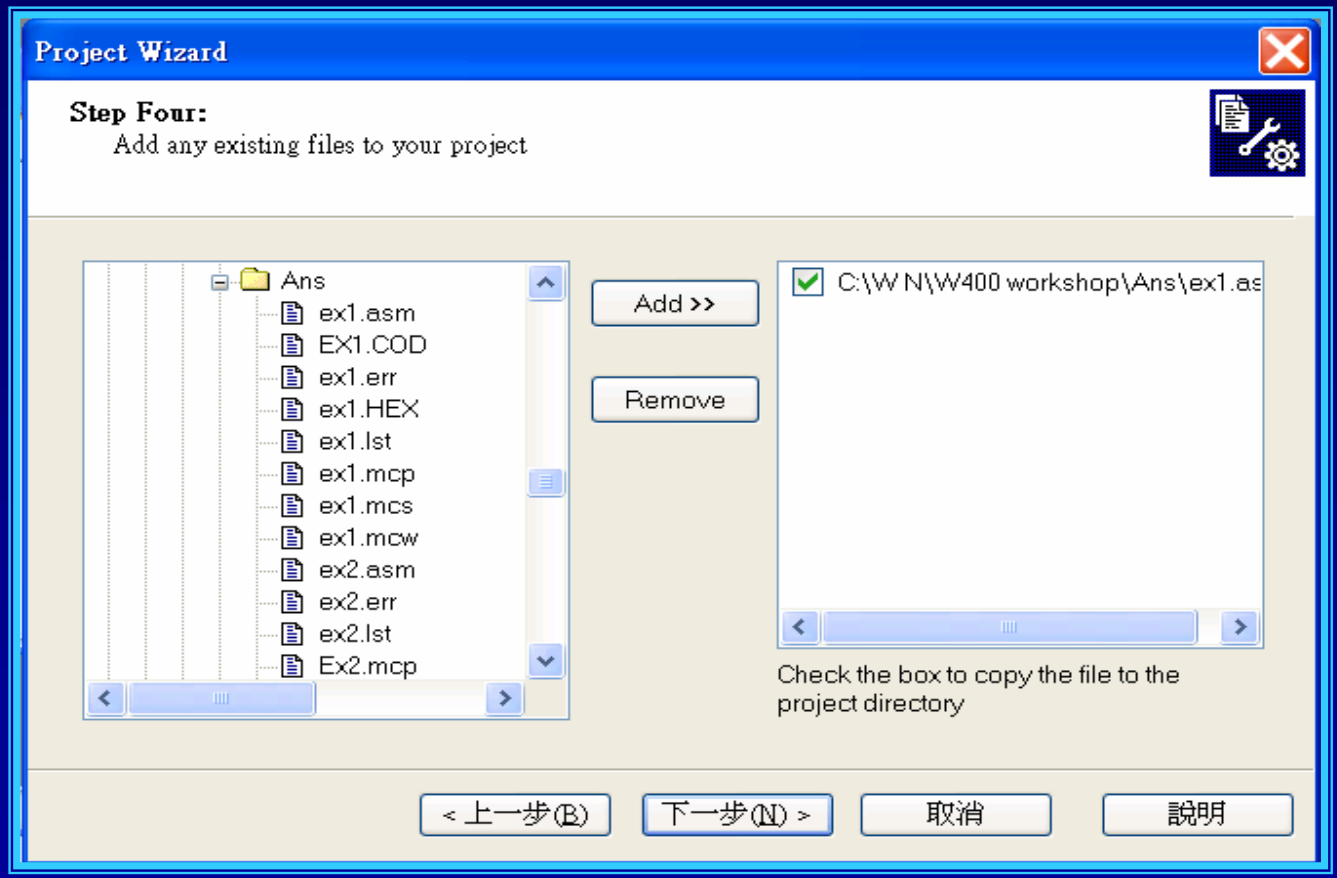


輸入專案名稱 **Ex1.mcp**

用 **Browse** 來指定目錄

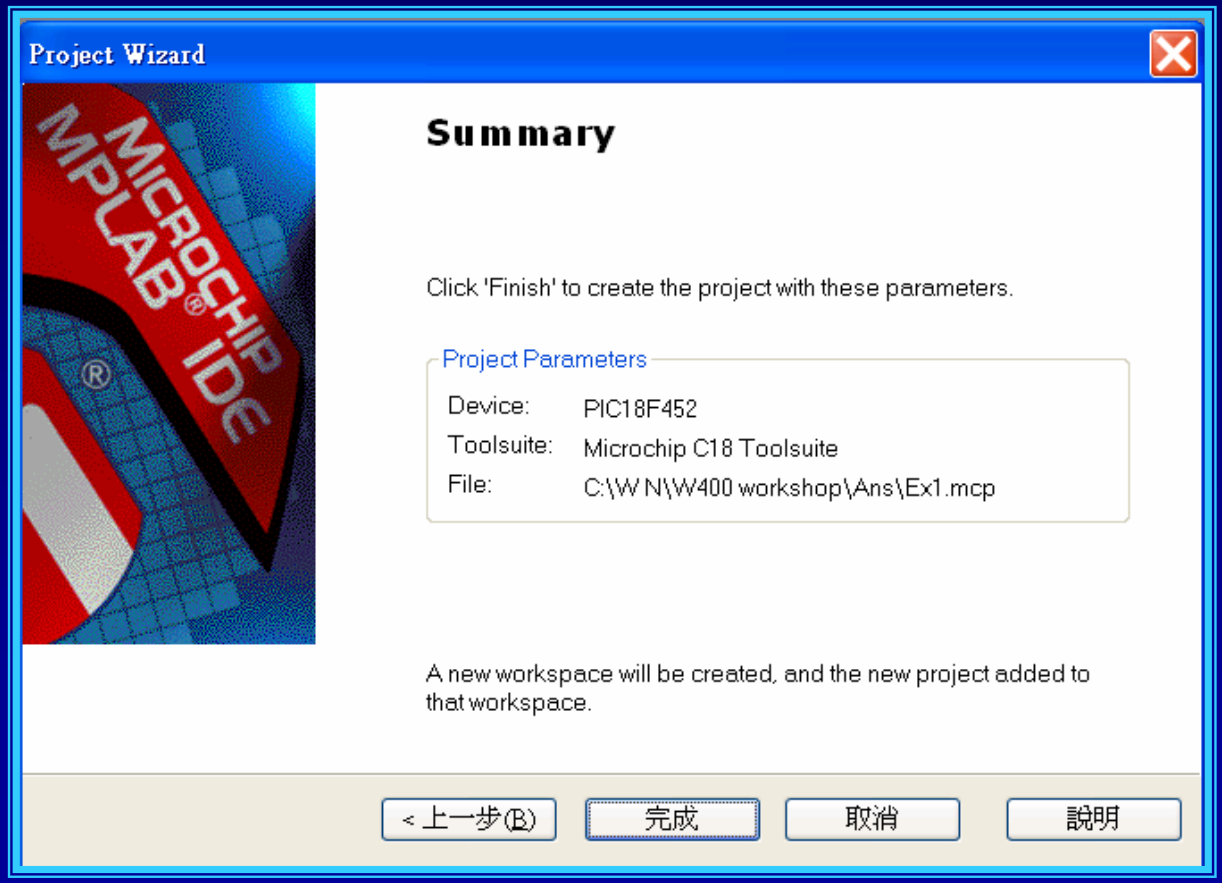
練習：建立新的專案

- 加入 **source code** 到 **Project** 的工作目錄

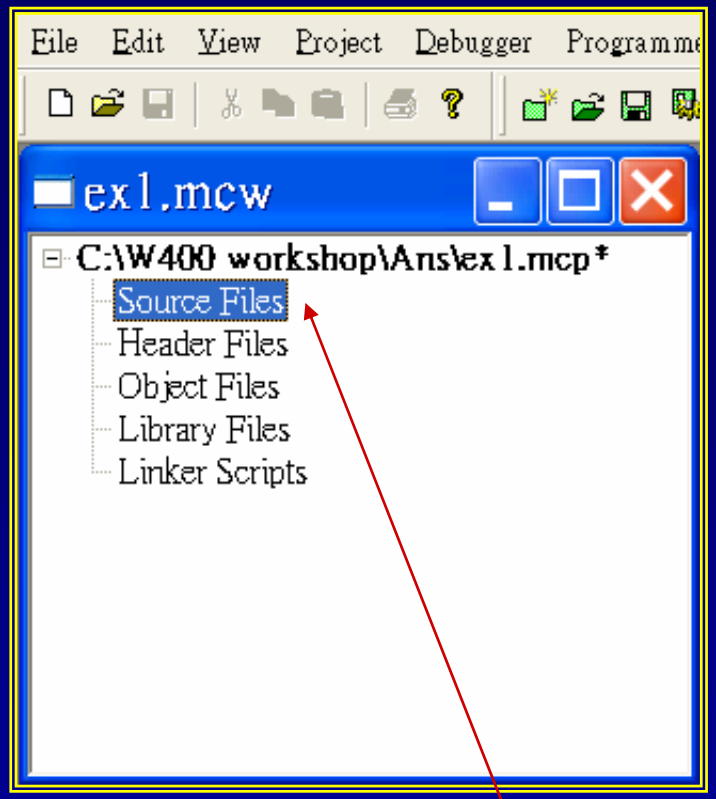


練習：建立新的專案

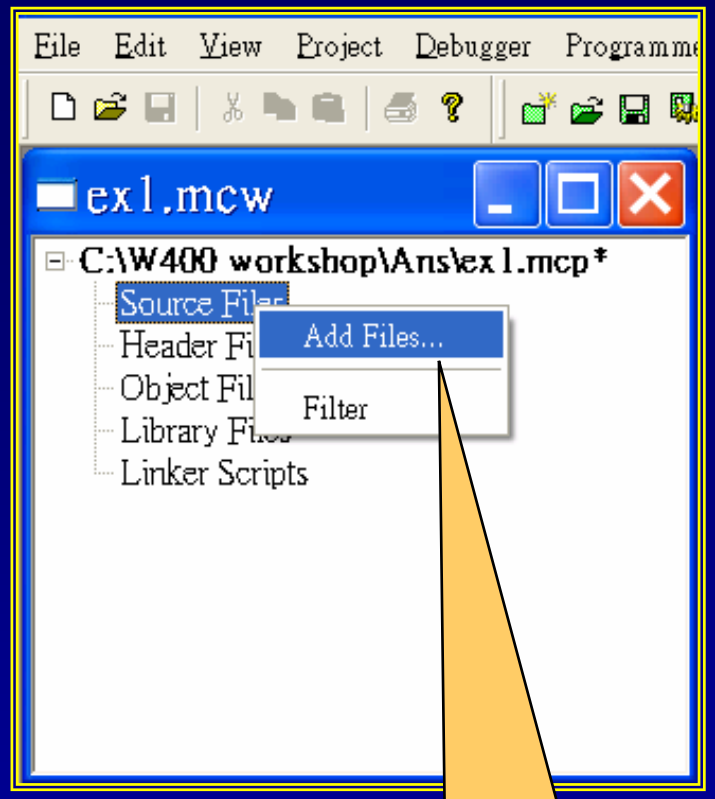
- **Project** 建立完成，請在確認設定是否正確



練習：加入原始檔案方式

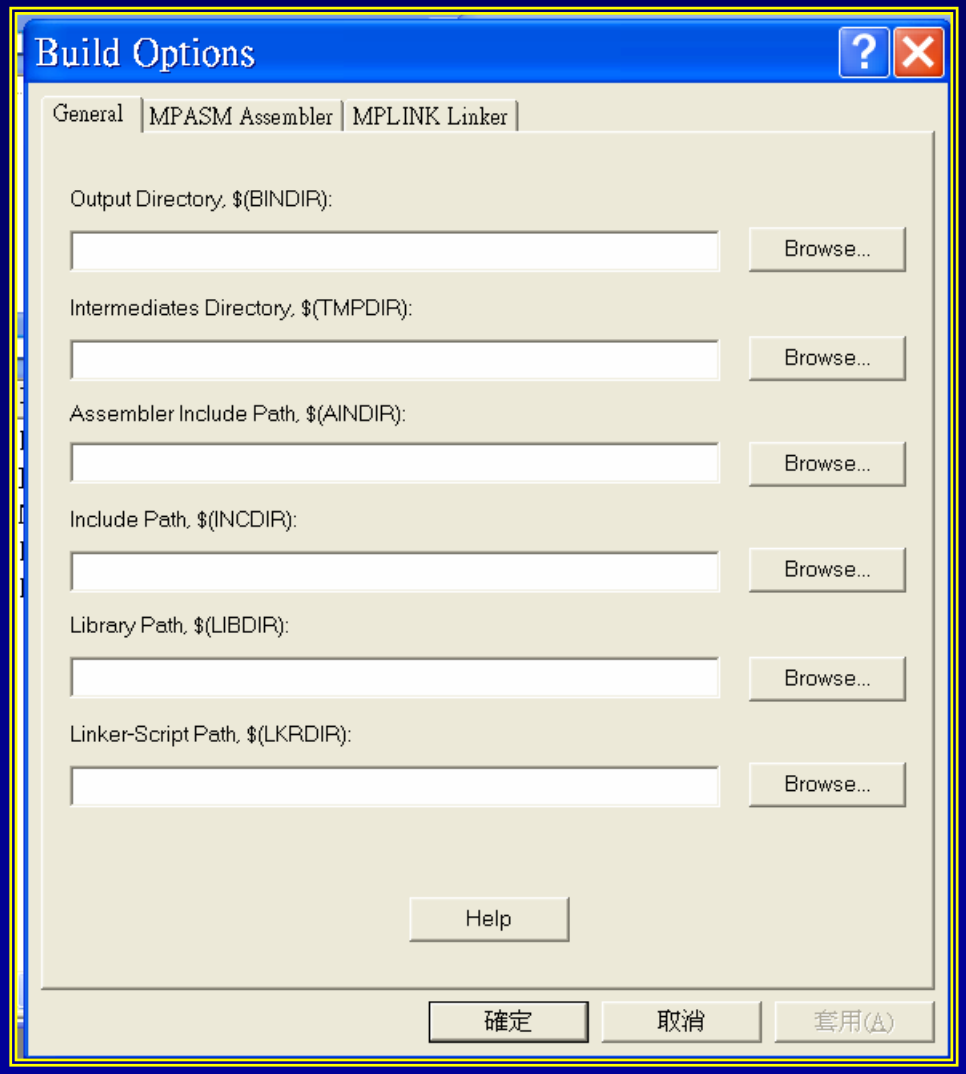


將滑鼠指到
Source Files
並按下滑鼠的右鍵



MPLAB IDE
將要求您加入檔案

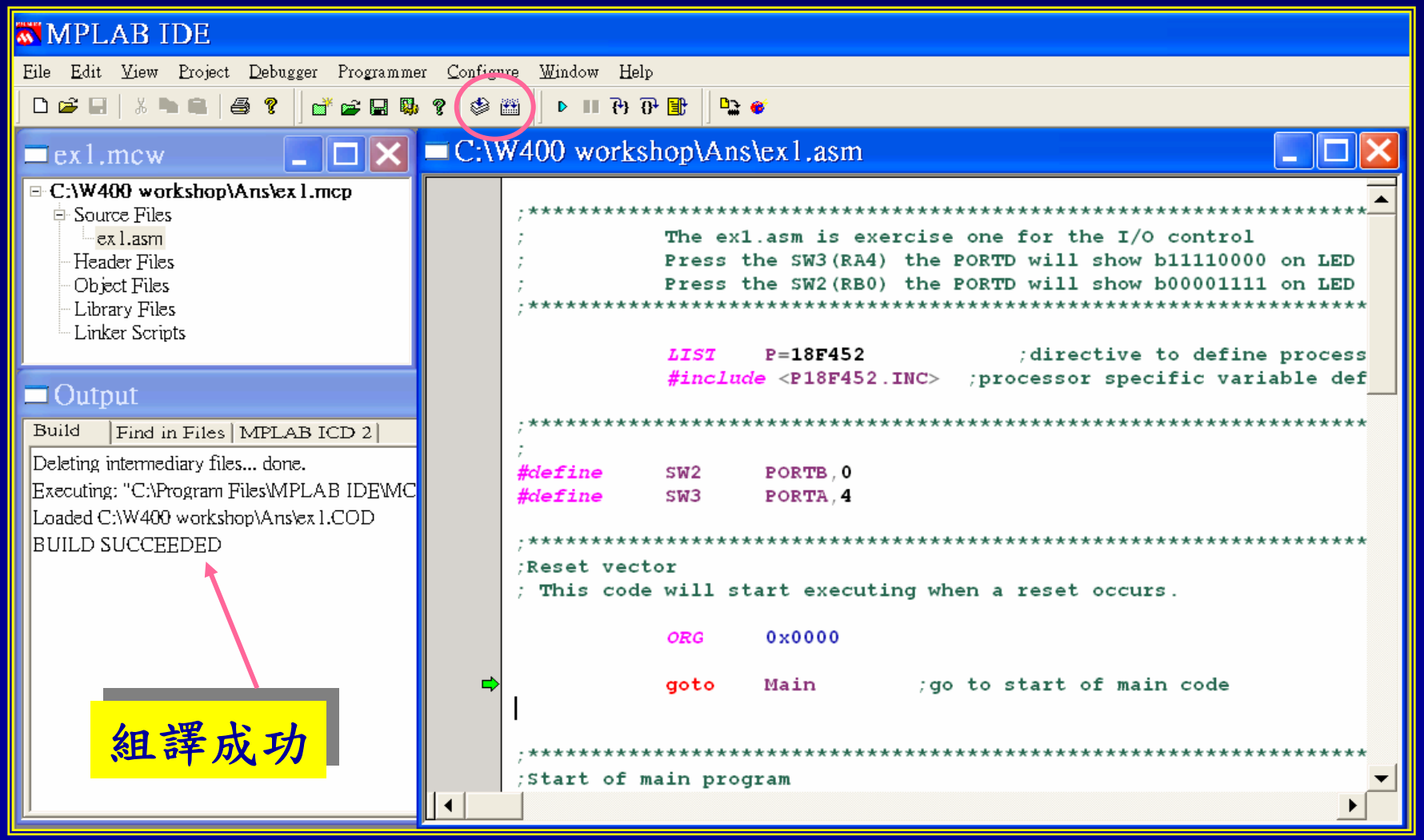
練習：建立資料庫來源位置



- 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

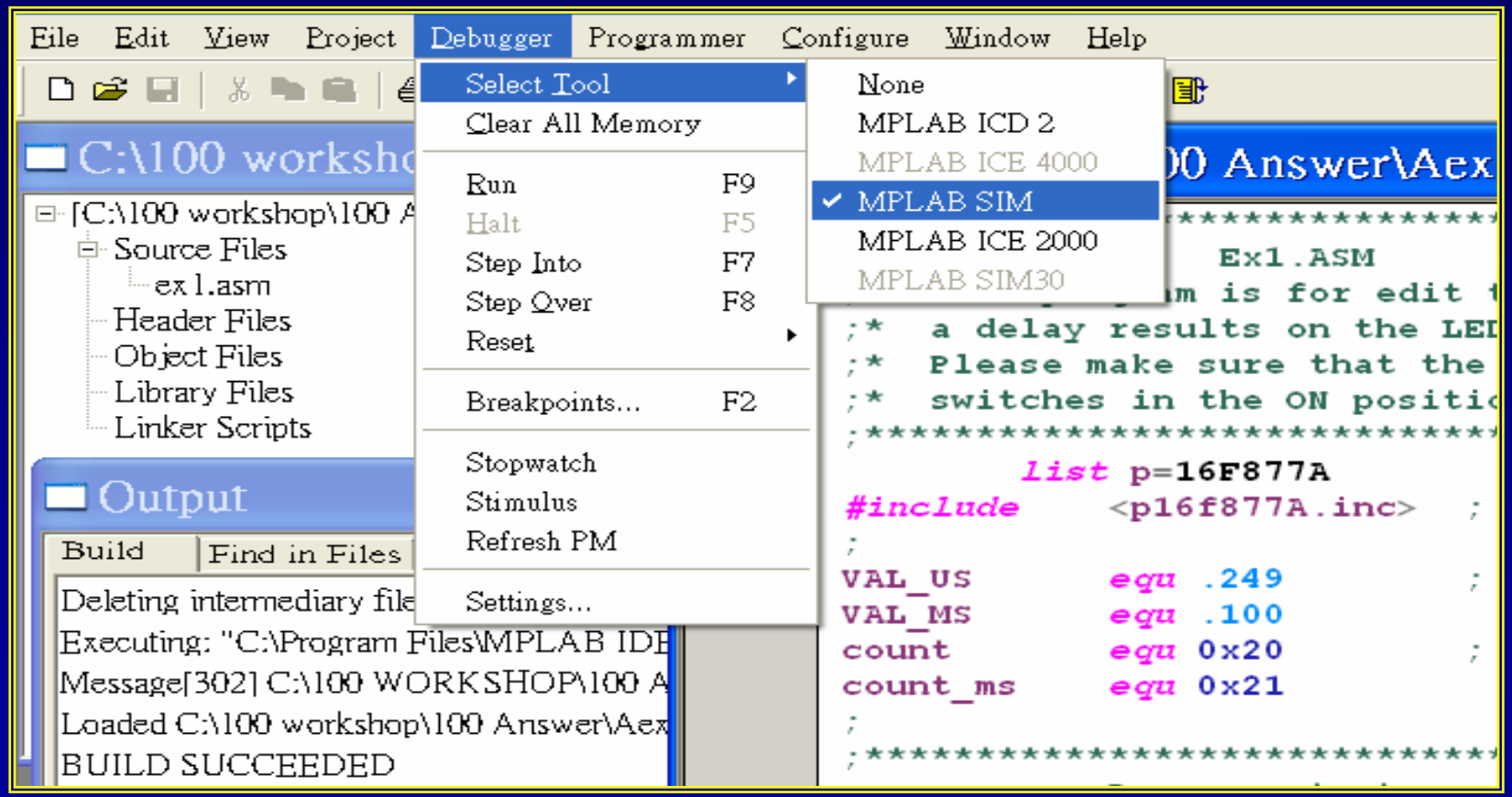
練習：組譯程式

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

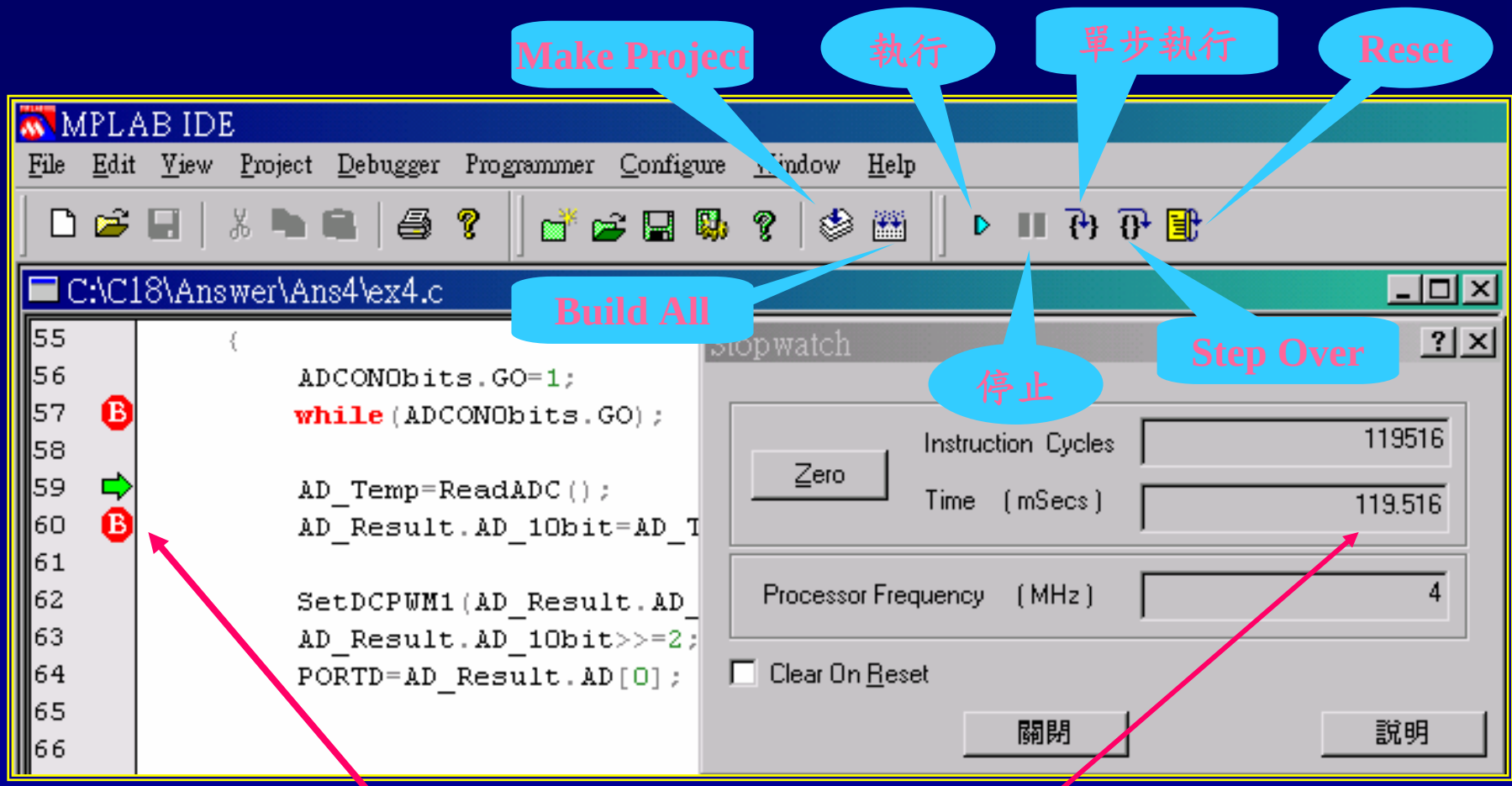


啟動基本 MPLAB-SIM

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



MPLAB SIM 的基本除錯功能

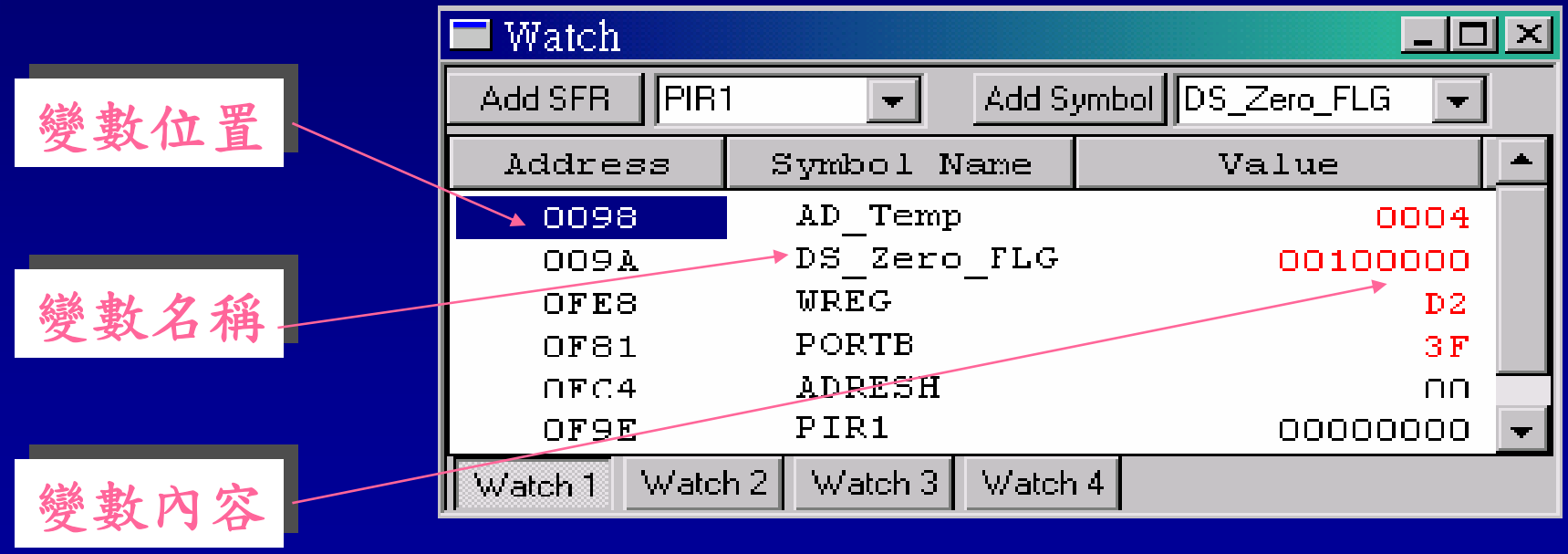


利用mouse的右鍵來設定中斷點

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



編譯程式與除錯

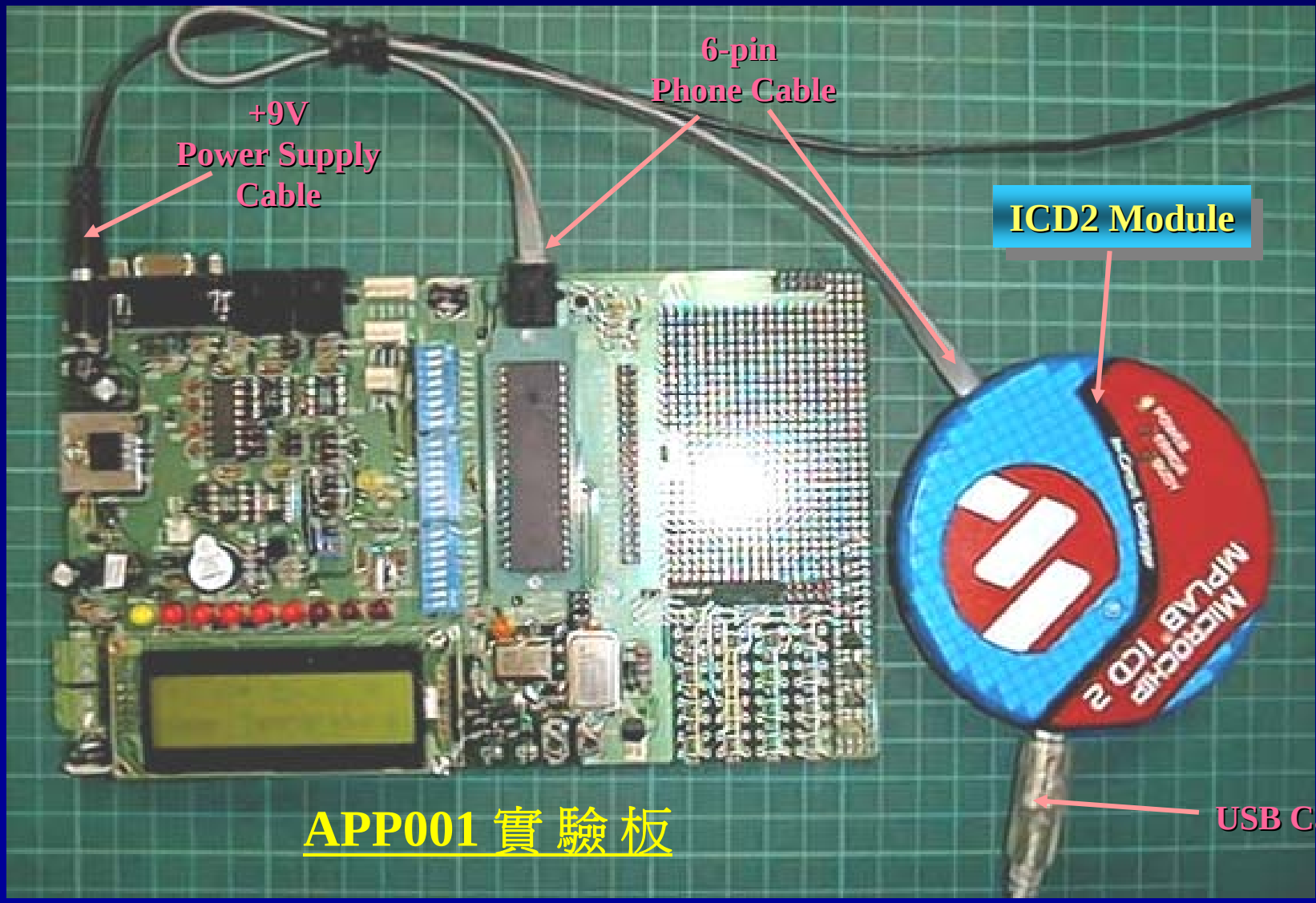
- 利用 **Build All (Ctrl + F10)** 來編譯程式
  - 如有錯誤，請參考錯誤訊息並加以修正
  - 在錯誤訊息處按滑鼠兩次即能輕易切換至錯誤行
- 利用**Watch Window**來觀察變數
- 利用**Mouse**的右鍵來設定中斷點
  - 熟悉**Reset**，**Run**，**Halt**功能
  - **Step**，**Step Over**功能
  - **RAM**，**SFR** 視窗在 組合 語言下就不是那麼重要了**?**
  - 其它的視窗 **?**

使用 MPLAB ICD2 除錯

MPLAB-ICD2 功能

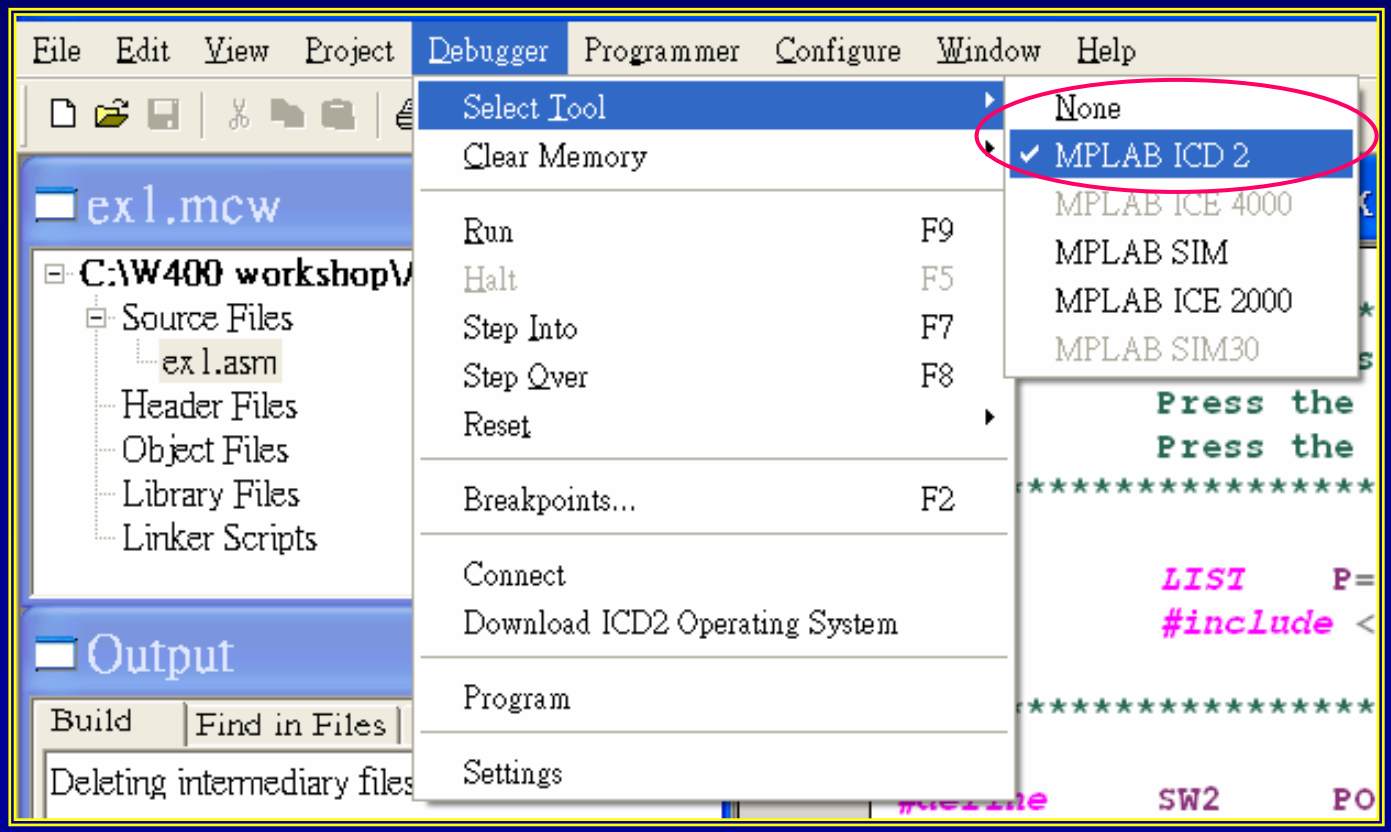
- 全速執行
- 單步執行
- 單/多點硬體中斷
- 變數觀察，原始程式除錯等級
- 快速載入程式到模擬元件
- 可當模擬元件的燒錄工具
- 工作電壓 ： **2.5V to 5.5V**
- 頻率範圍 ； **32KHz to 40MHz**
- **RS-232** 或 **USB** 介面
- 價格便宜

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啓動 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，請參閱：
  - **MPLAB IDE v6.10** 中文使用手冊，第五章

- 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”**

正確的語法表達 (一)
數值、數字(字元)表示法

- 十進制表示(Decimal): D'<十進制數目>' & . <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'<十六進制數目>' & 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=PIC18F452
- #INCLUDE - 加入一原始檔、定義檔或敘述檔
  - #include <18f452.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=18F452          ; 定義使用的 MCU 爲 PIC18F452
                #include    <p18F452.inc>     ; 使用 18F452 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                        ; 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

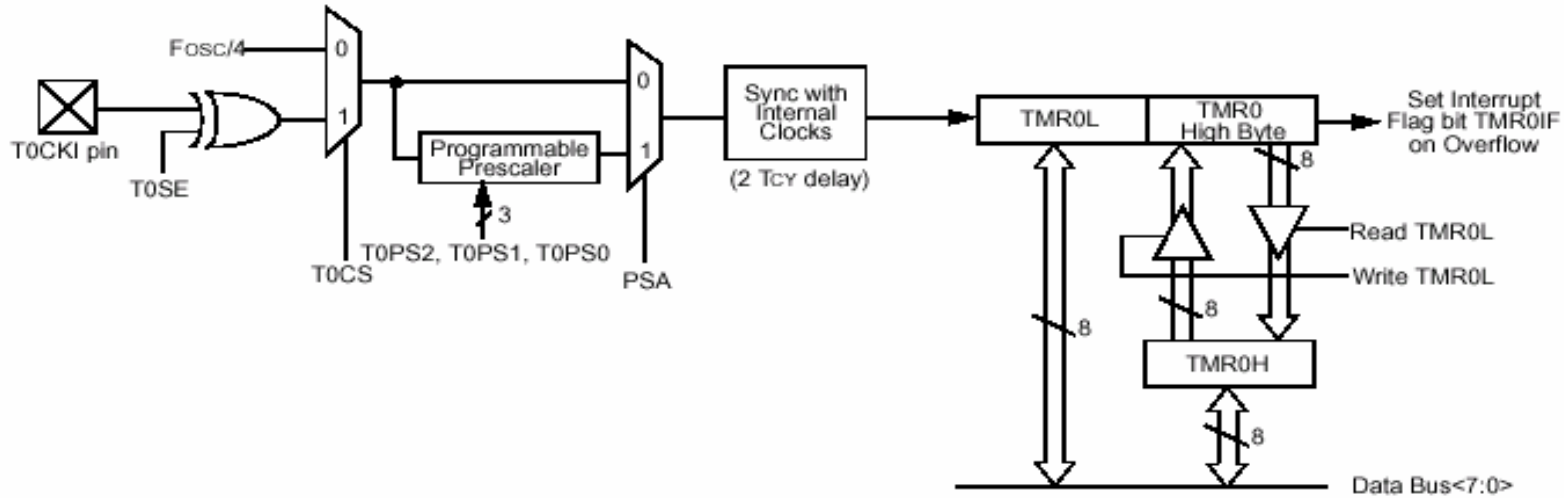
- 本練習的程式 **Ex1.asm** 放在
  - **C:\W400 workshop\Exercise\Ex1.asm**
- 建立一個屬於這個練習的 **Project**，**Project** 名稱自定

- 修改這個練習，使程式能依按鍵的選擇來控制 **LED**
  - **JP9** 的 **1 & 2** 腳需用 **Jumper** 短路
  - 設定 **RB0 , RA4** 爲輸入
  - 設定 **PORTD** 爲輸出 **(** 驅動 **LED)**
  - 按 **SW2 (RB0)** 時，點亮 **LED0 – LED3**
  - 按 **SW3 (RA4)** 時，點亮 **LED4 – LED7**

Timer & CCP Module
介紹

PIC18 系列的周邊 : Timer0

- Timer0 可設定爲 8-Bits 或 16-Bits 模式
- 16-bit mode 時， TMR0H 會在讀寫 TMR0L 時才真正的被讀出或寫入 Timer0
  - 可準確的由 8-bit 的 Data Bus 來讀取 16-bit 的 Timer 值
- 計時器產生溢位時 FFh to 00h (FFFFh to 0000h)，即產生中斷

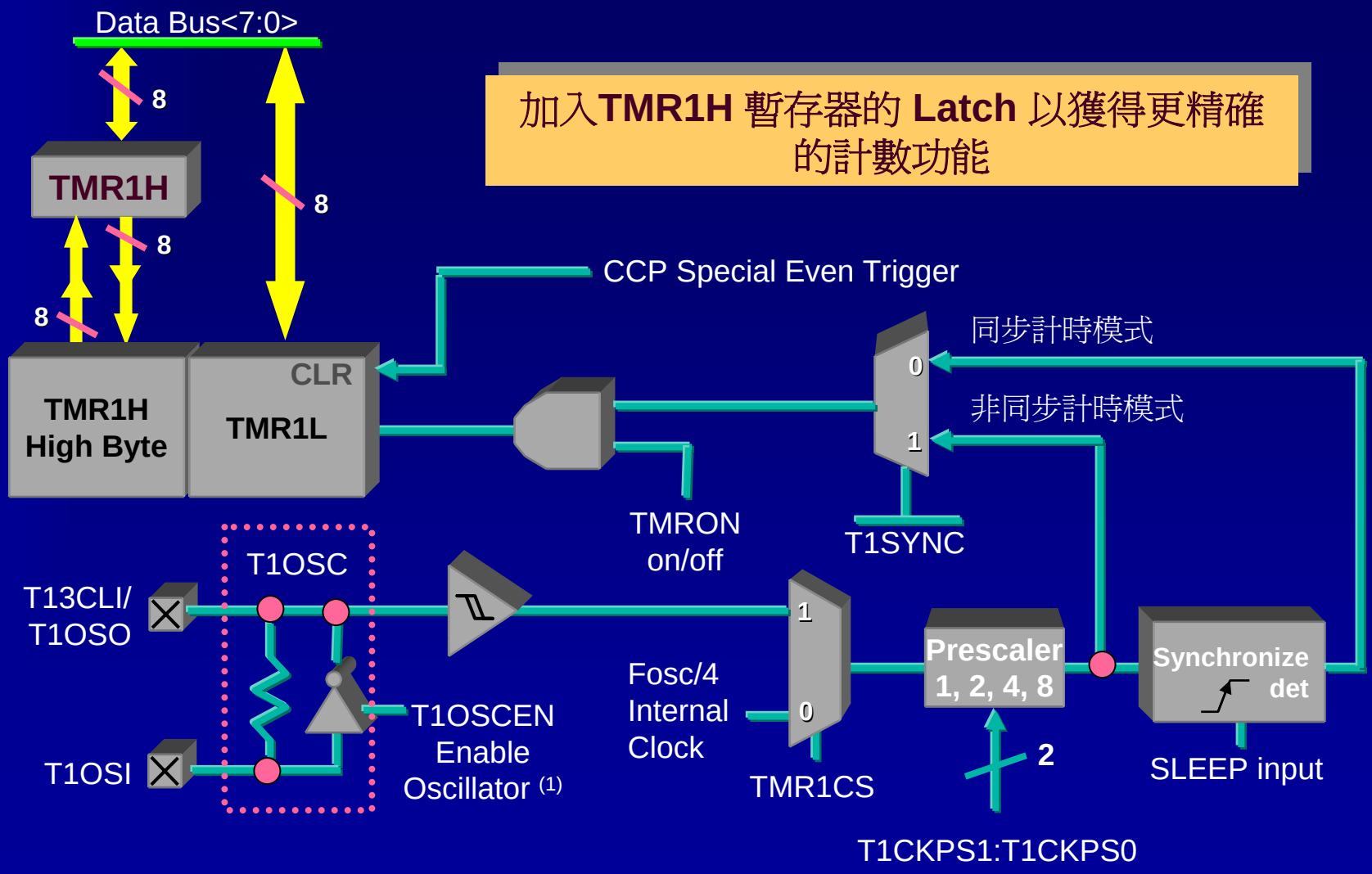


Note: Upon RESET, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

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 ：÷1， ÷2，÷4，or ÷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爲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 *(continued)*



加入**TMR1H** 暫存器的 **Latch** 以獲得更精確的計數功能

PIC18 系列的周邊
Timer1 and Timer3 as a Real-Time Clock

- 使用 32.768 kHz 的石英振盪器作爲 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。

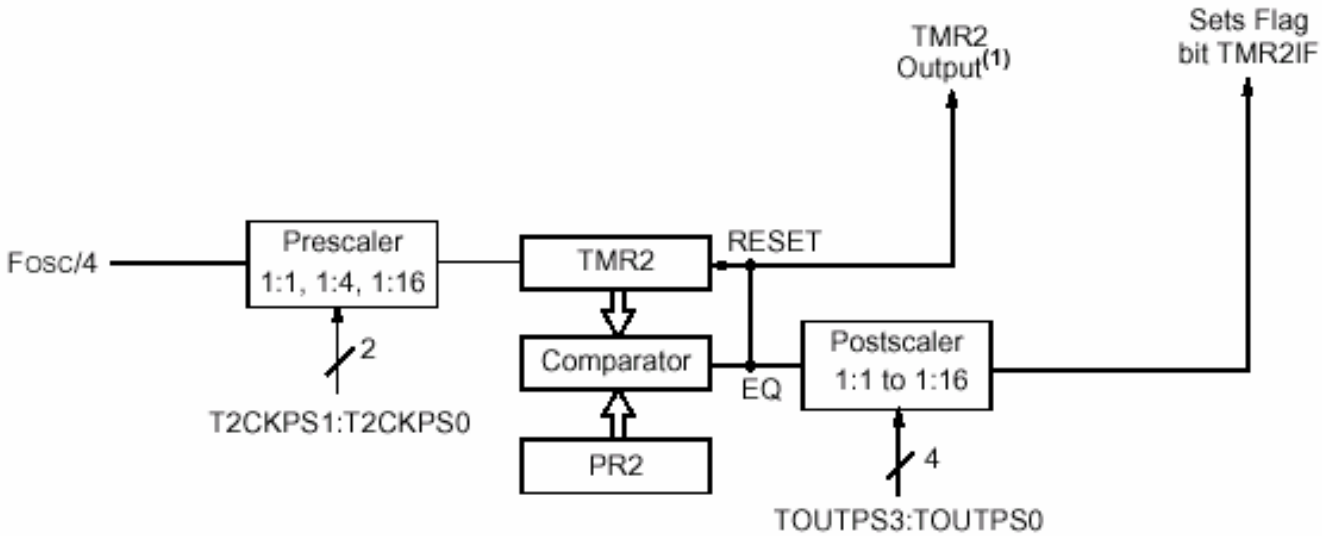
PIC18 系列的周邊
Timer2

- 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 爲一可讀、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，並自動將自己清除爲零，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 模組可規劃爲：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)*

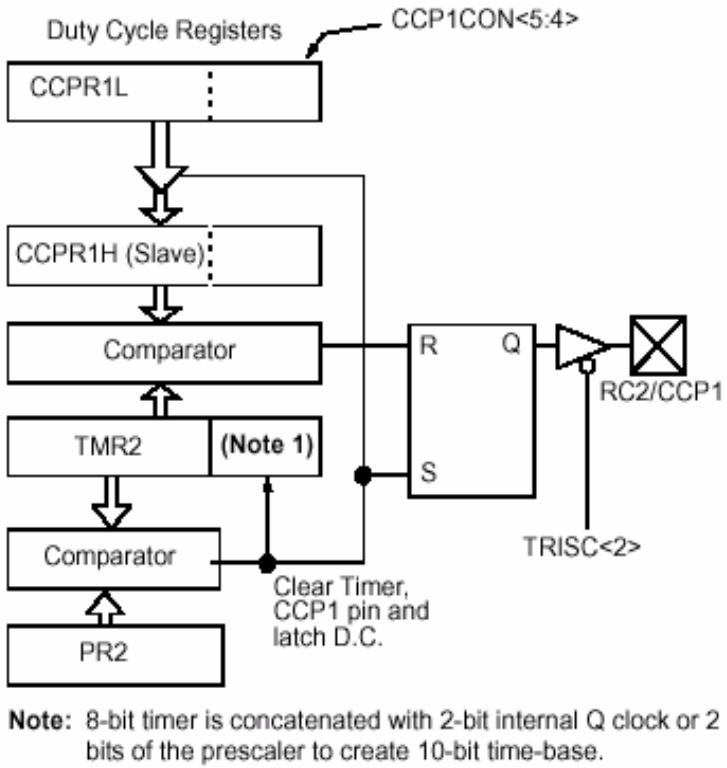
- 定義：
  - **Input Capture**：計算輸入脈衝的時間、寬度及頻率；必須使用**16-bit** 的 **Timer1** 爲計時基準。
  - **Output Compare**：設定一 **16-bit** 的計時比較值與 **16-bit** 的 **Timer1** 相比，若相等則依其設定來改變輸出。
  - **PWM (Pulse Width Modulation)**：頻率固定的方波，藉由改變其脈衝寬度**(Duty)**來進行數值的調變

PIC18 系列的周邊
CCP Module: PWM Mode

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

PIC18 系列的周邊
CCP Module: PWM Mode

FIGURE 14-3: SIMPLIFIED PWM BLOCK DIAGRAM



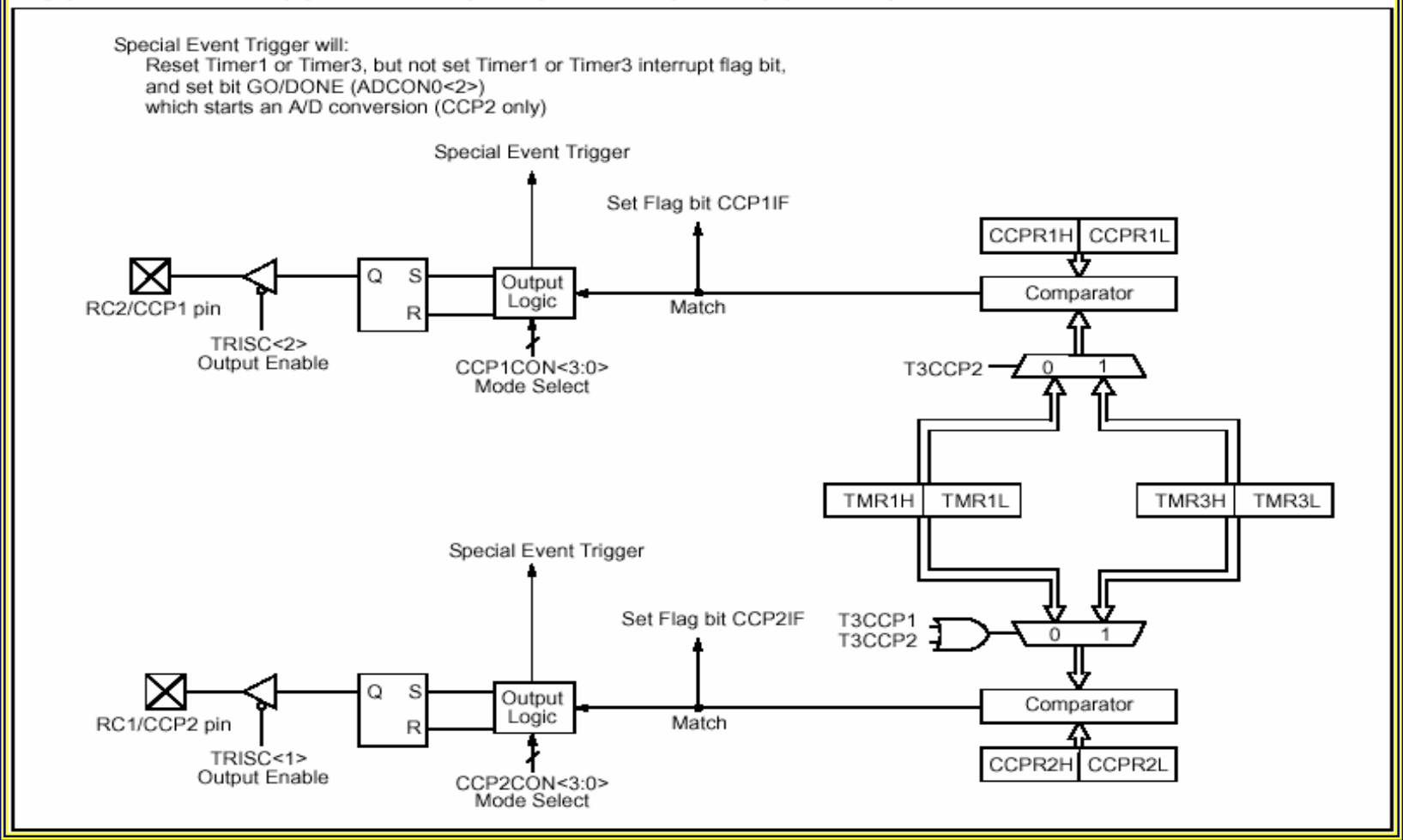
Note: 8-bit timer is concatenated with 2-bit internal Q clock or 2 bits of the prescaler to create 10-bit time-base.

- 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



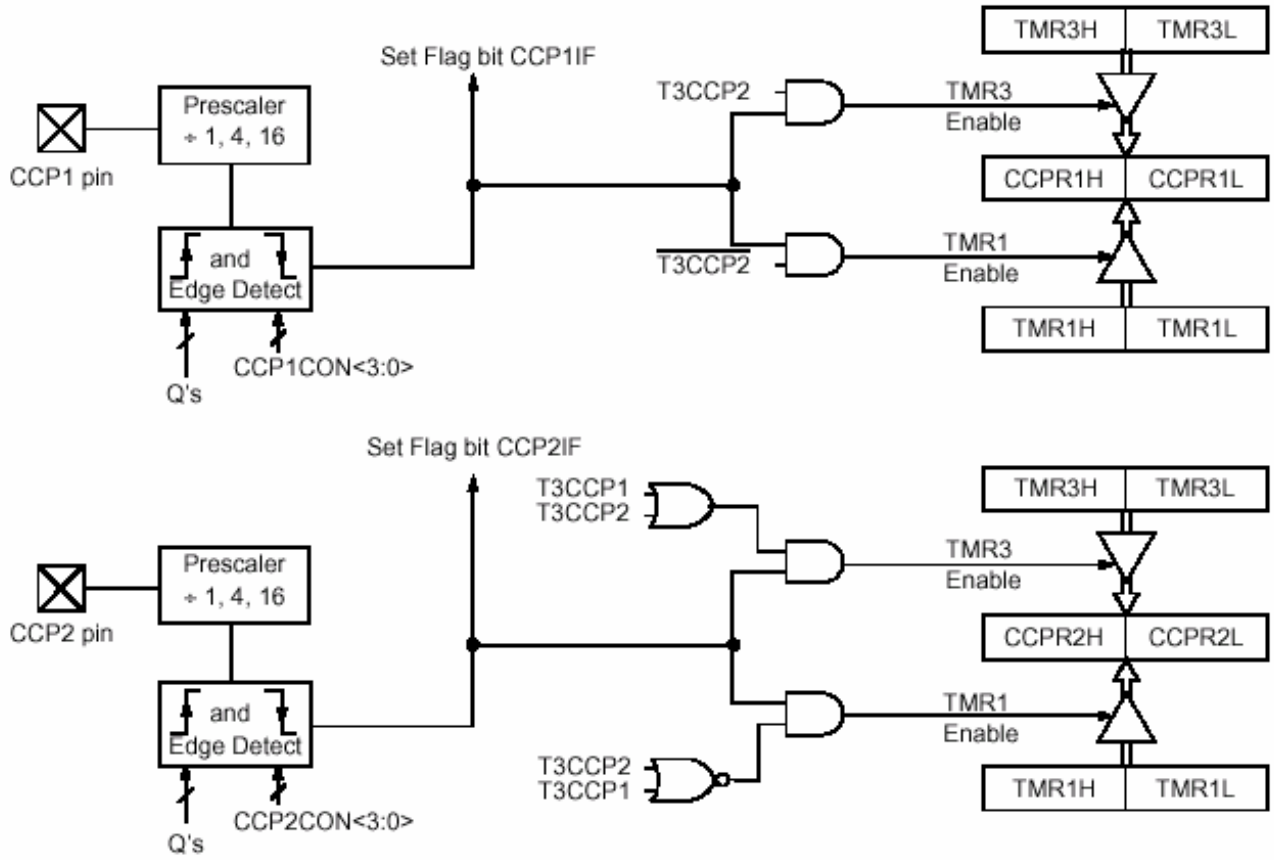
PIC18 系列的周邊
CCP Module: Output Compare Mode

- 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



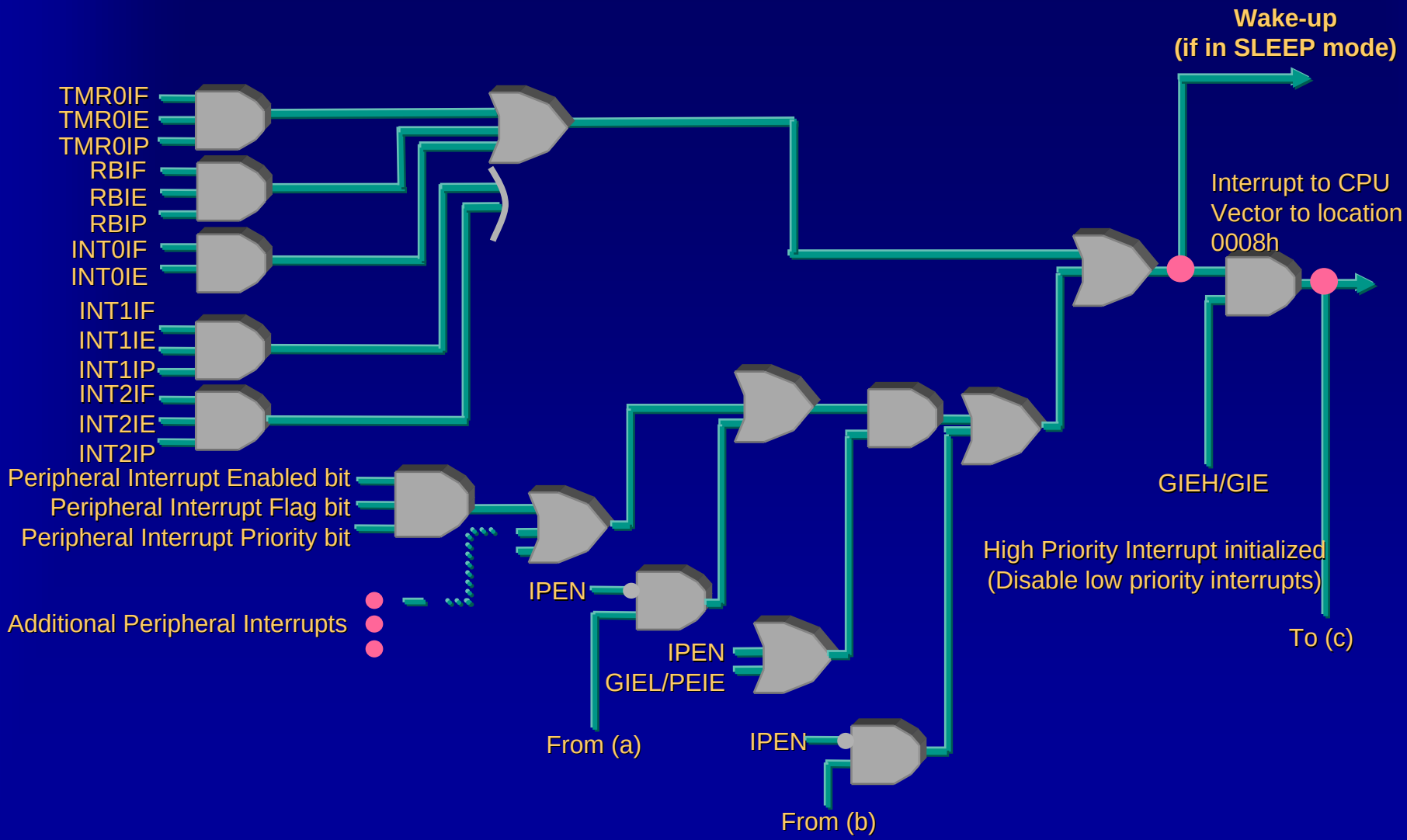
PIC18 系列的周邊
CCP Module: Input Capture Mode

- 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)
- ROCN 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/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
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** 並啓動 **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 設定爲 500 ms 中斷 CPU 一次 (使用Fosc / 1)
    - ✓ 0.5Sec = Timer Value * (1 / 32768Hz)
    - ✓ 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

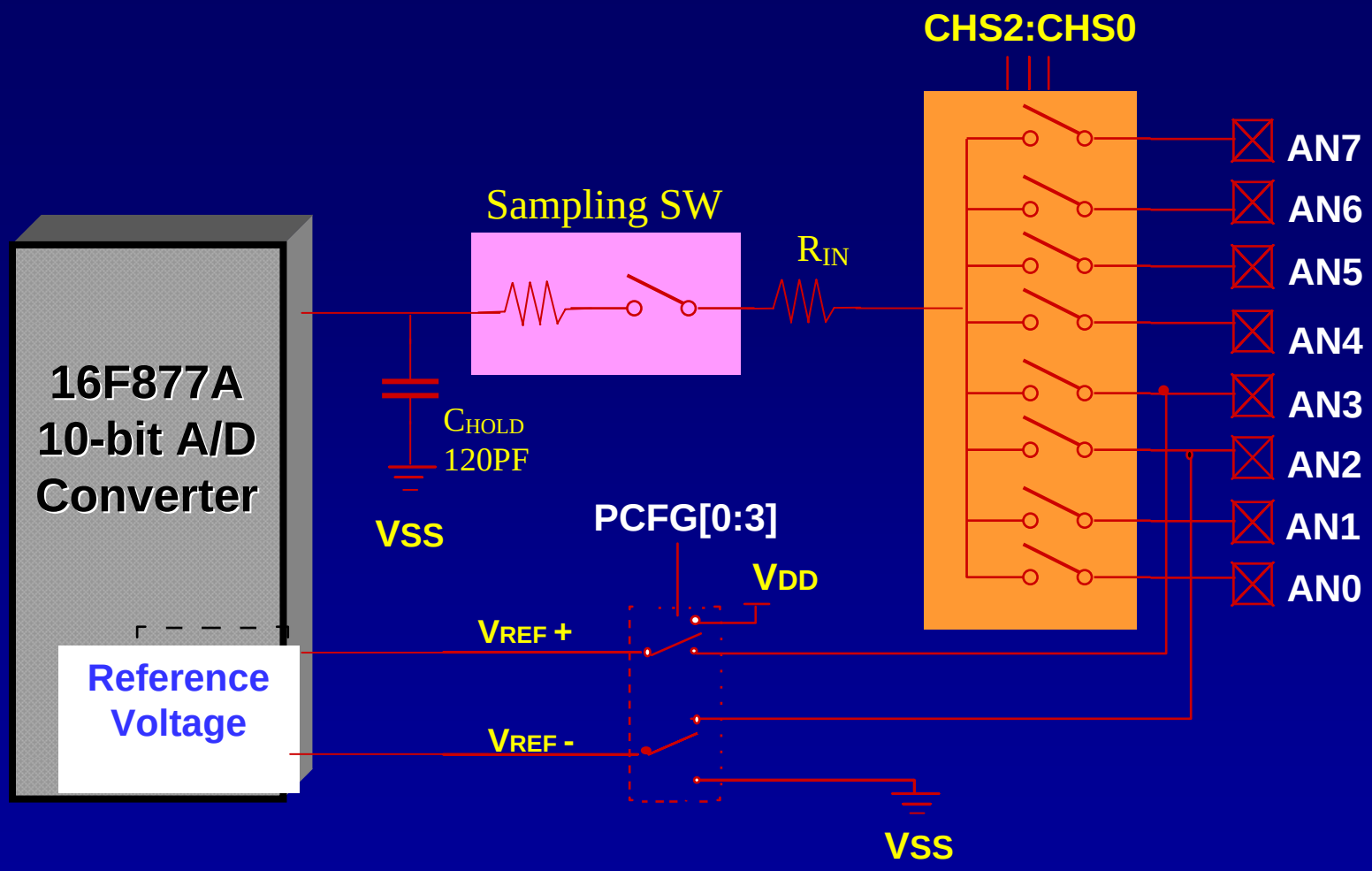
JP2 的設定是 3,5 及 4,6 短路； JP9 的設定是 1,2 短路

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 µS (輸入阻抗<10K)
- 類比輸入轉換時間：19.2 µS (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 µs
  - 如輸入腳位固定，其轉換時間只需：29.2 µs

10-bit A/D 方塊圖


CHS2:CHS0
AN7
AN6
AN5
AN4
AN3
AN2
AN1
AN0
Sampling SW
RIN
16F877A
10-bit A/D
Converter
CHOLD
120PF
VSS
PCFG[0:3]
VDD
VREF +
VREF -
Reference
Voltage
VSS

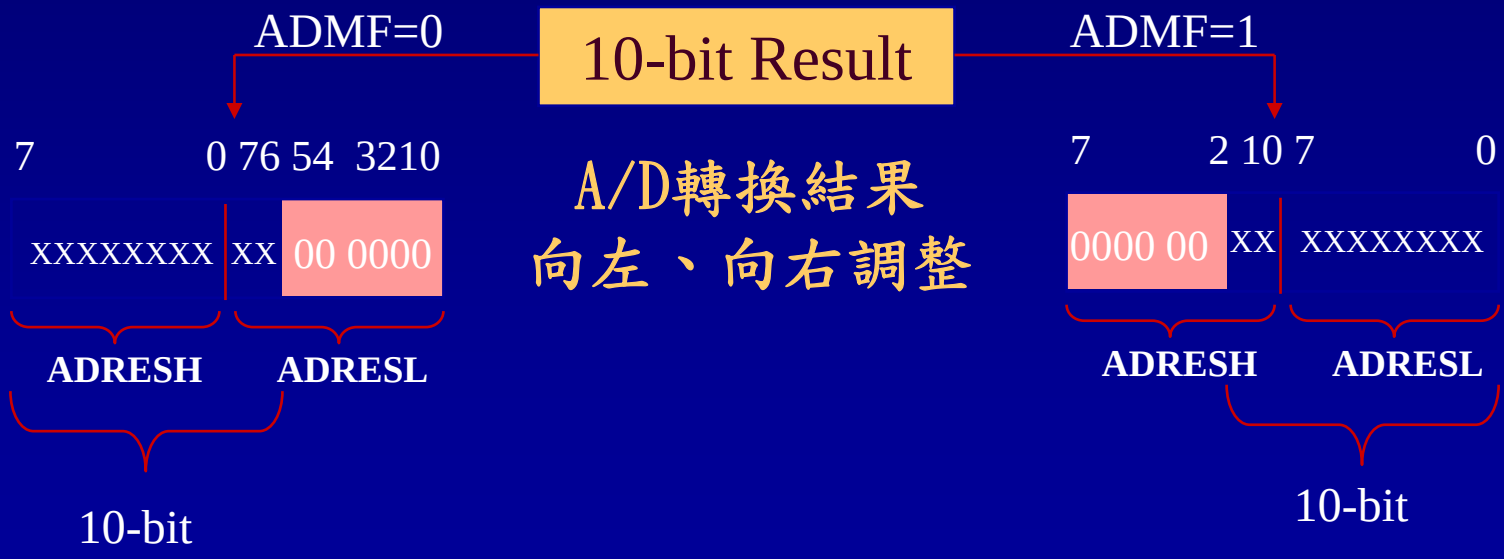
A/D 控制暫存器

ADCON0 Register

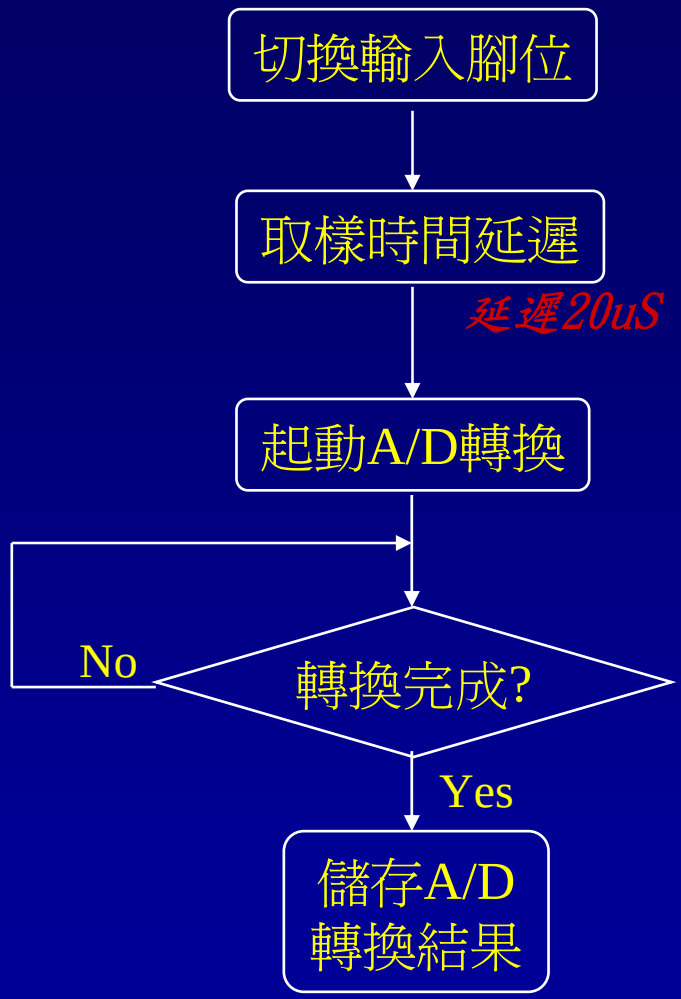
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	---	ADON
bit7							bit0

ADCON1 Register

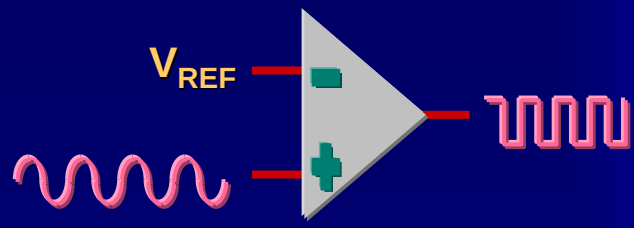
ADFM	ADCS2	-----	-----	PCFG3	PCFG2	PCFG1	PCFG1



A/D 轉換基本流程



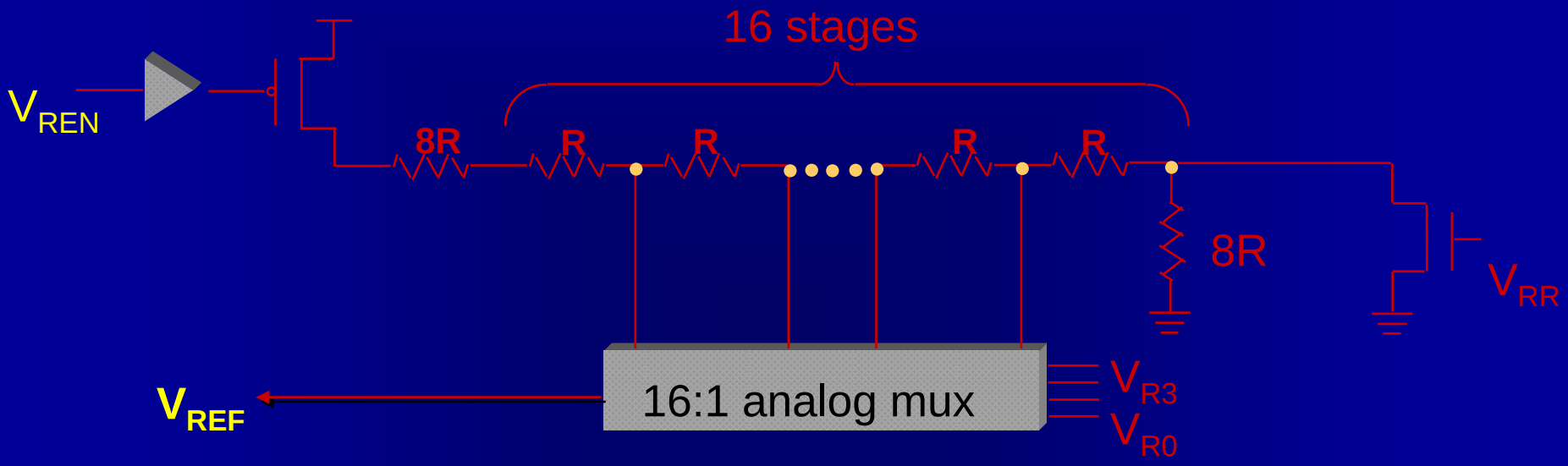
PIC18 系列的周邊
Analog Comparator Module



- 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 VREF: Block Diagram



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

練習三 ：使用 A/D 轉換器

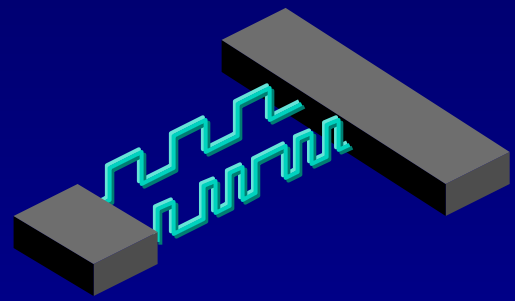
- 使用 **A/D** 轉換器將 **VR1** 的電壓透過 **AN0** 的輸入腳進行 **A/D** 的電壓轉換：
  - 練習程式 : C:\W400 workshop\Exercise\Ex3.asm
  - 有關 A/D 的設定，請參考 ADCON0 及 ADCON1 兩個暫存器的設定說明
    - PIC18Fxx2 Data Sheet (Page 179 & 180)
  - A/D 轉換的結果，向左調整後只取其最高的 8-bits (最低的兩個位元不予理會)，並將每次轉換後的結果顯示在 LED

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



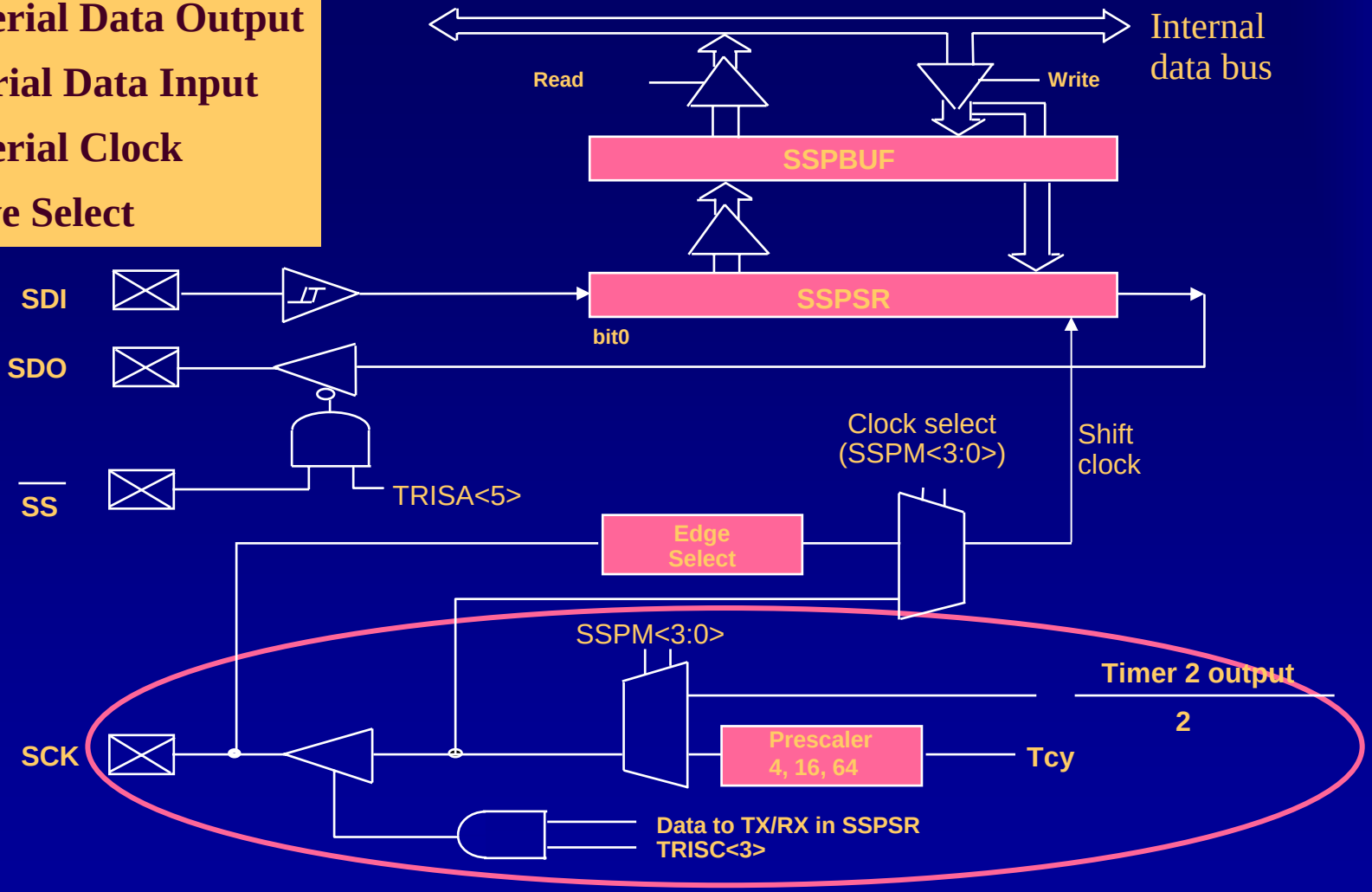
SPI 方塊圖

SDO: Serial Data Output

SDI: Serial Data Input

SCK: Serial Clock

$\overline{SS}$: Slave Select

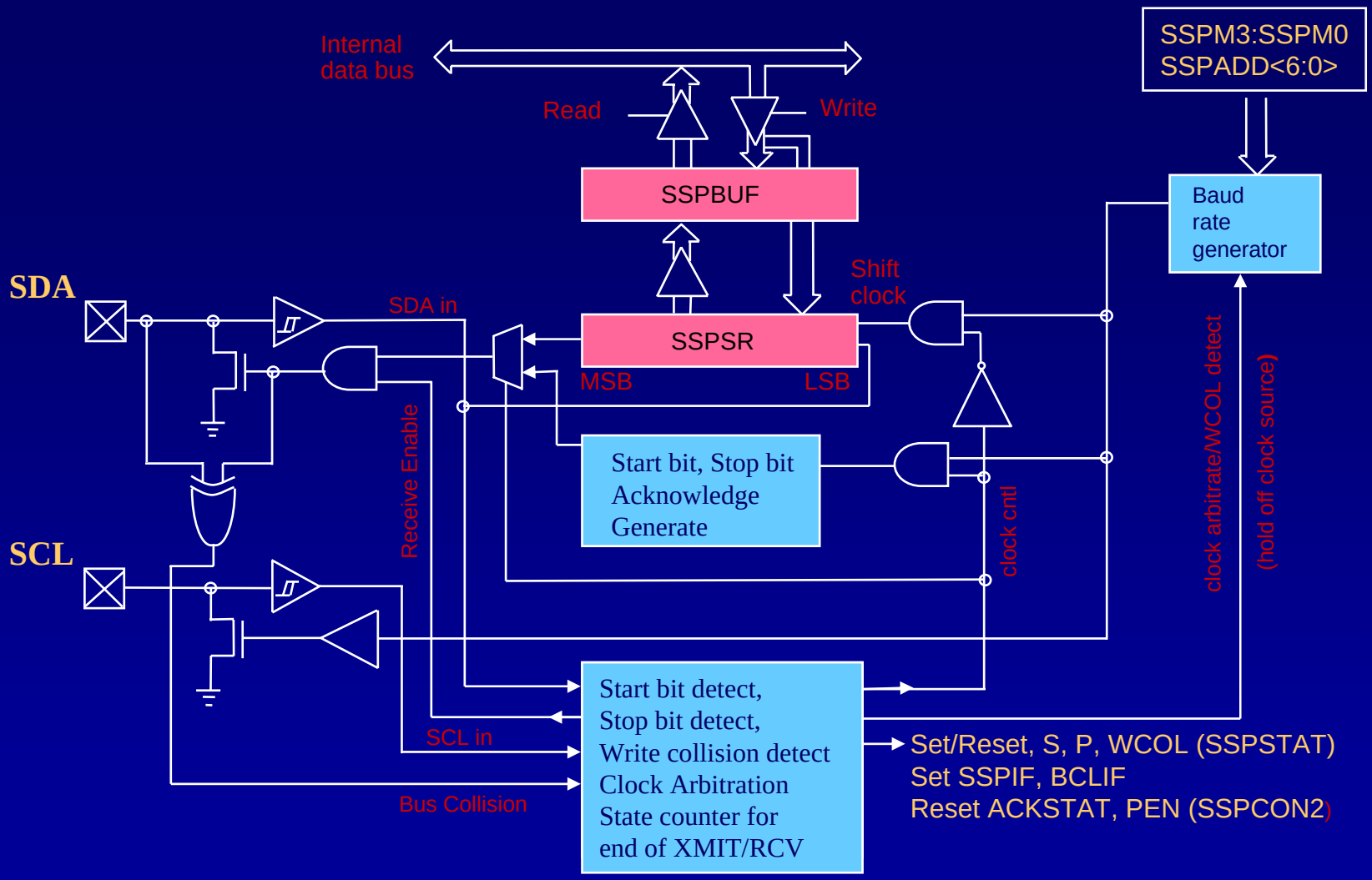


Master Mode: SCK is the clock output pin

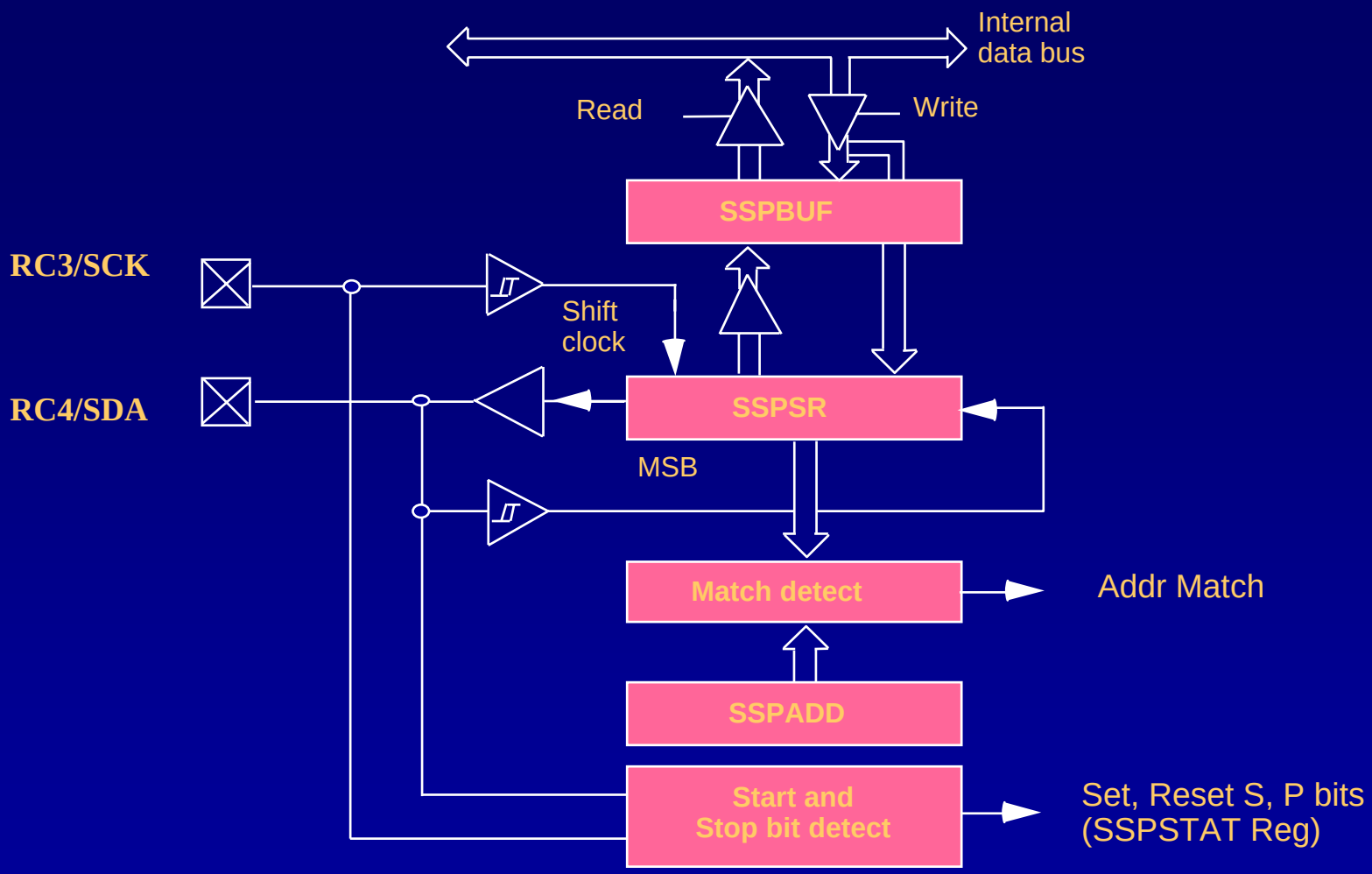
同步串列通訊(二)
Master Synchronous Serial Port (MSSP)

- **I^2C (Inter-Integrated Circuit) 模式**
  - 全硬體化設計，支援下列三種I^2C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 Master 模式方塊圖

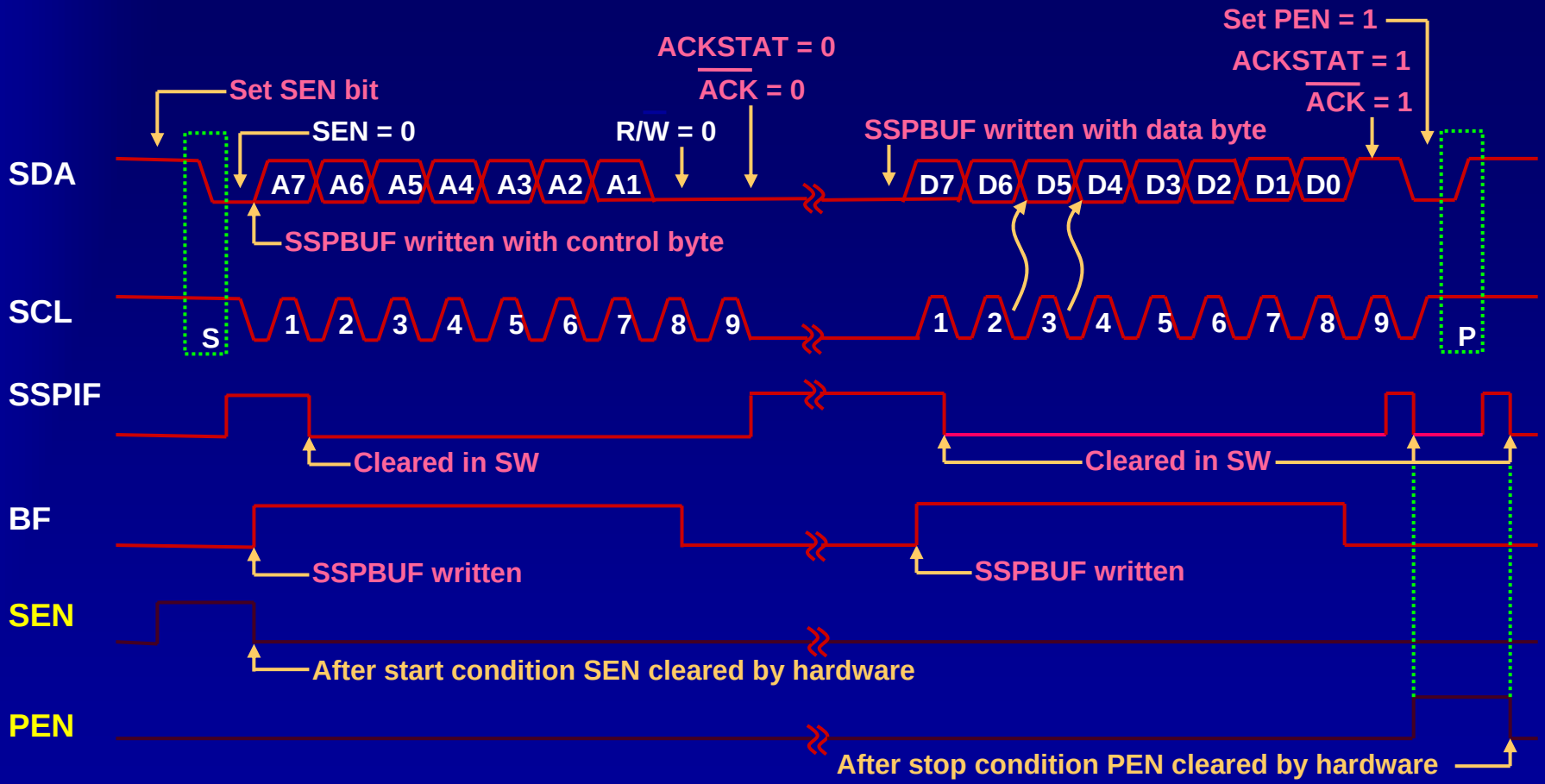


I²C Slave 模式方塊圖



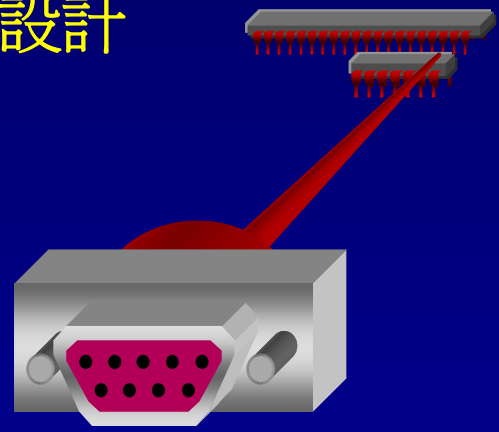
I²C Master 模式下

發送資料的時序

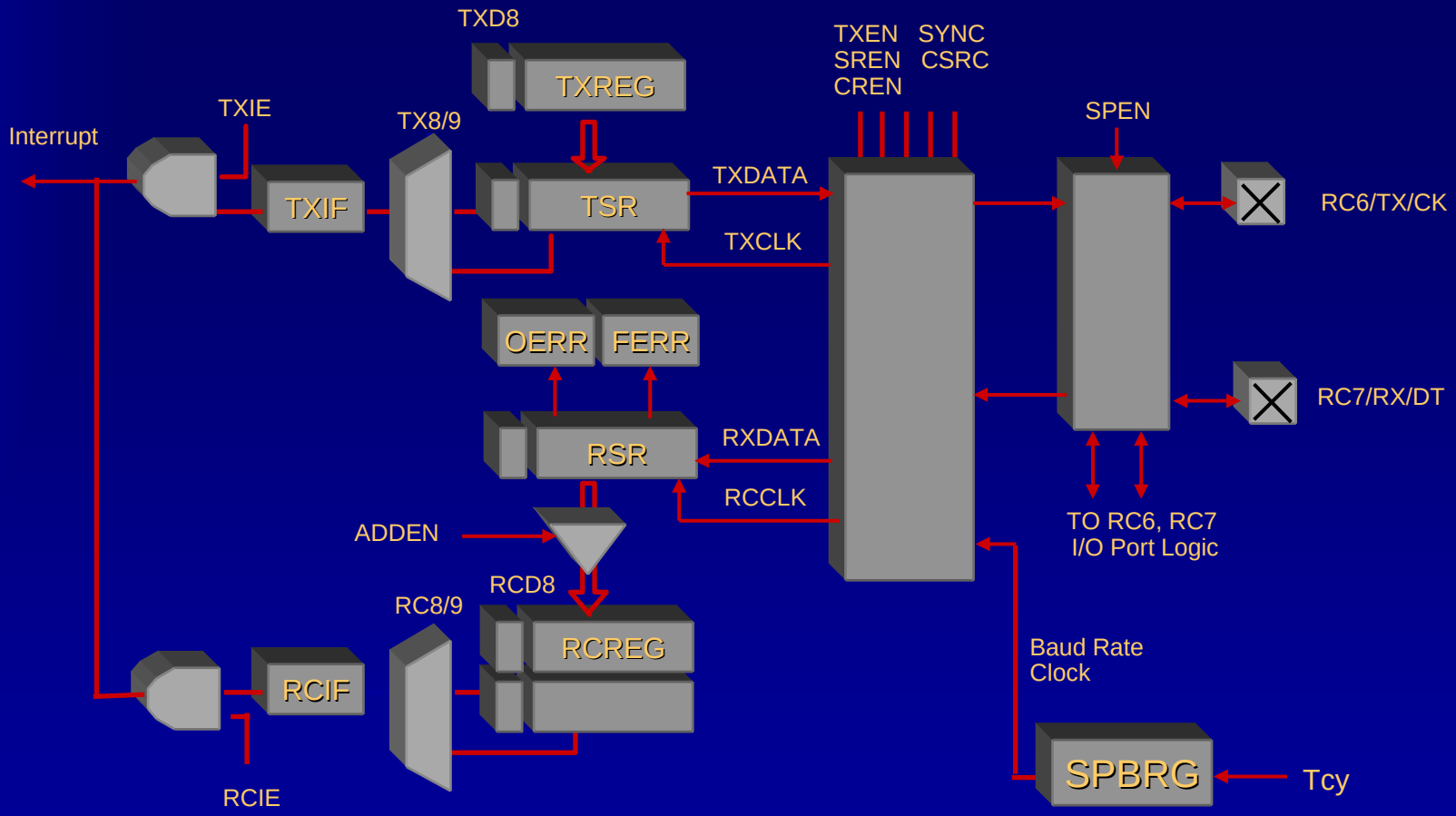


PIC18 系列的周邊
Addressable USART (AUSART)

- 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** 爲第一位元
- 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

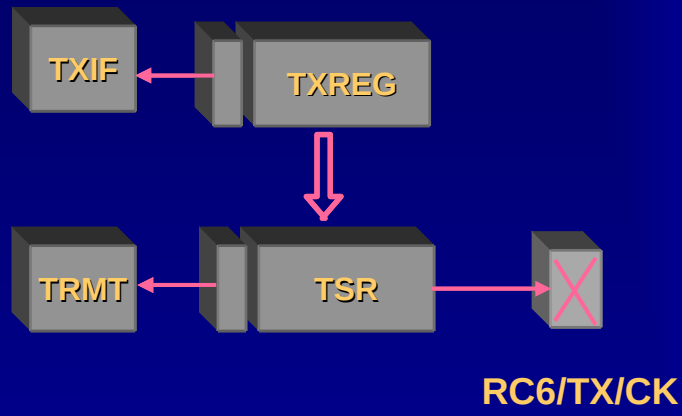


PIC18 系列的周邊
USART 的結構方塊圖



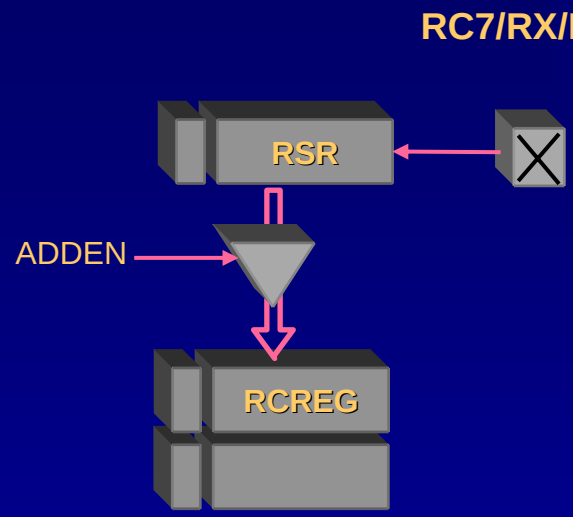
TXIF & TRMT 的動作

- 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TMRT = 1
- 假設TXREG剛載入資料時TMRT 爲空的(TMRT=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TMRT來的快



RCIF的動作

- RSR 爲一個移位器，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



2 deep FIFO

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)作爲資料或命令的指引
  - 當b8爲1時，表示該筆資料爲命令，啓動位址判斷；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，該筆資料爲一般的資料

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爲零，若爲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 , Baud Rate = Fosc / [64 (x + 1)]
  - BRGH = 1 , 高速模式 , Baud Rate = Fosc / [16 (x + 1)]

- 以 16MHz 的工作頻率爲例：欲求出 9600 bps 的速率
  - Baud Rate = Fosc / [64 (x + 1)]
  - 9600 = 16MHz / [64 (x + 1)]
  - x = [(16MHz / 9600) / 64) – 1
  - x = 25.025
  - 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**
- 傳送格式設爲：**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**
- 接收格式設爲：**9600，N，8，1**

練習四 ：使用 USART

- 練習程式 : C:\W400 workshop\Exercise\Ex4.asm
- 設定 **USART** 的工作模式爲：**9600，N，8，1**
- 電腦的超級終端機設定使用 **COM1** 或 **COM2** ，通訊協定爲：
  - **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

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> 爲助憶文字，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'
  00000020              00017                count, count_ms, led_count
  00000023              00018                led_index
  00000024              00019                led_buff
                        00020                endc
                         :
0042   3008             00081                movlw   0x8
0043   00A2             00082                movwf   led_count
0044   0DA4             00083 loop_right     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一般稱之爲巨集函數，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 “$”符號做爲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，若資料長度爲奇數則該字元組的偶數位元組(較高的**Byte**)會自動補零
- 固定資料型態可爲：　　**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
  - 讀出的結果全爲 **‘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	RD

bit7 bit0

Bit7 : EEPGD 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 : EREN 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 CFGS 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:                                  EEPROM 讀取副程式

    movff   Data_EE_Addr,EEADR      將欲讀取的位址存入 EEADR
;
    bcf     EECON1,EEPGD            清除 EEPGD 位元
    bcf     EECON1,CFGS             清除 CFGS 位元
    bsf     EECON1,RD               啓動讀取 EEPROM 的動作
    movf    EEDATA,W                讀取動作完成，將資料存到 W Reg.
;
    return                          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 CFGS 位元， EECON1<6>
- 設定 WREN 位元， EECON1<1>
- 關閉所有的中斷
- 寫 55h 到 EECON2 暫存器
- 寫 AAh 到 EECON2 暫存器
- 設定 WREN 位元 (EECON1<1>)，以啓動寫入的動作
- 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，啓動寫入功能

(關閉所有的中斷，以避免程序錯亂 **)**

啓動寫入順序步驟

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) 爲單位，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

程式記憶體的讀取範例
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) 爲單位，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

程式記憶體的寫入範例
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                     ; TABLPTRU,TABLPTRH,TABLPTRL
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**