MICROCHIP

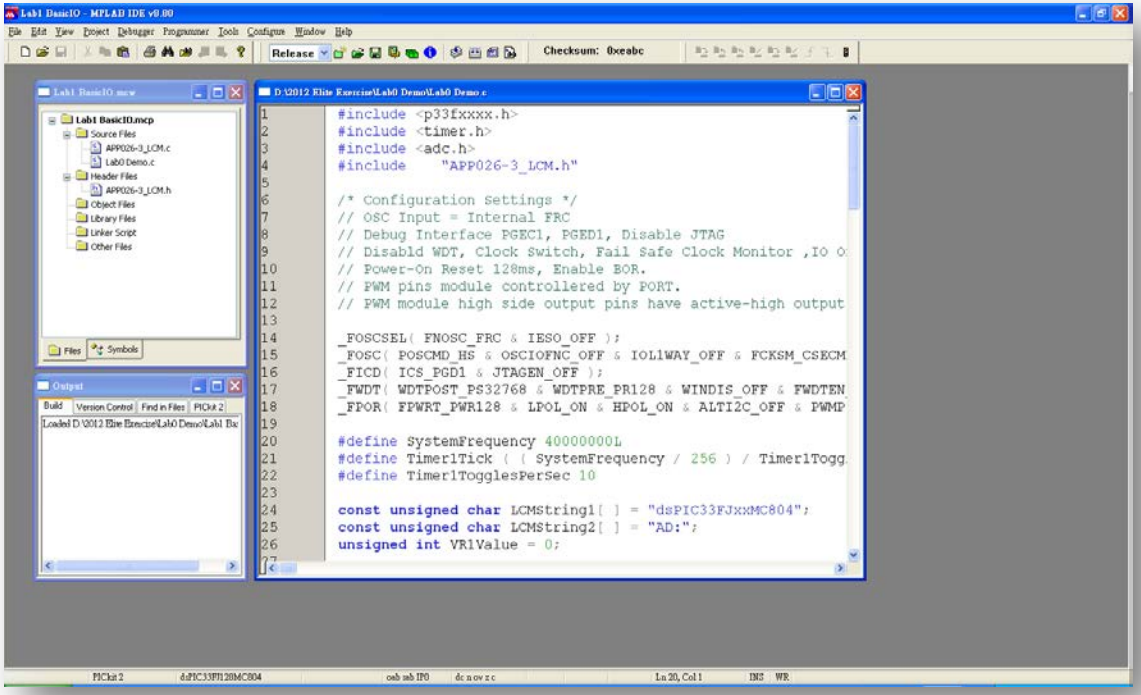
*Regional Training Centers*

# Section 2
# MPLAB IDE, MPLAB C30 and Development Tools Introduction

MICROCHIP
Regional Training Centers

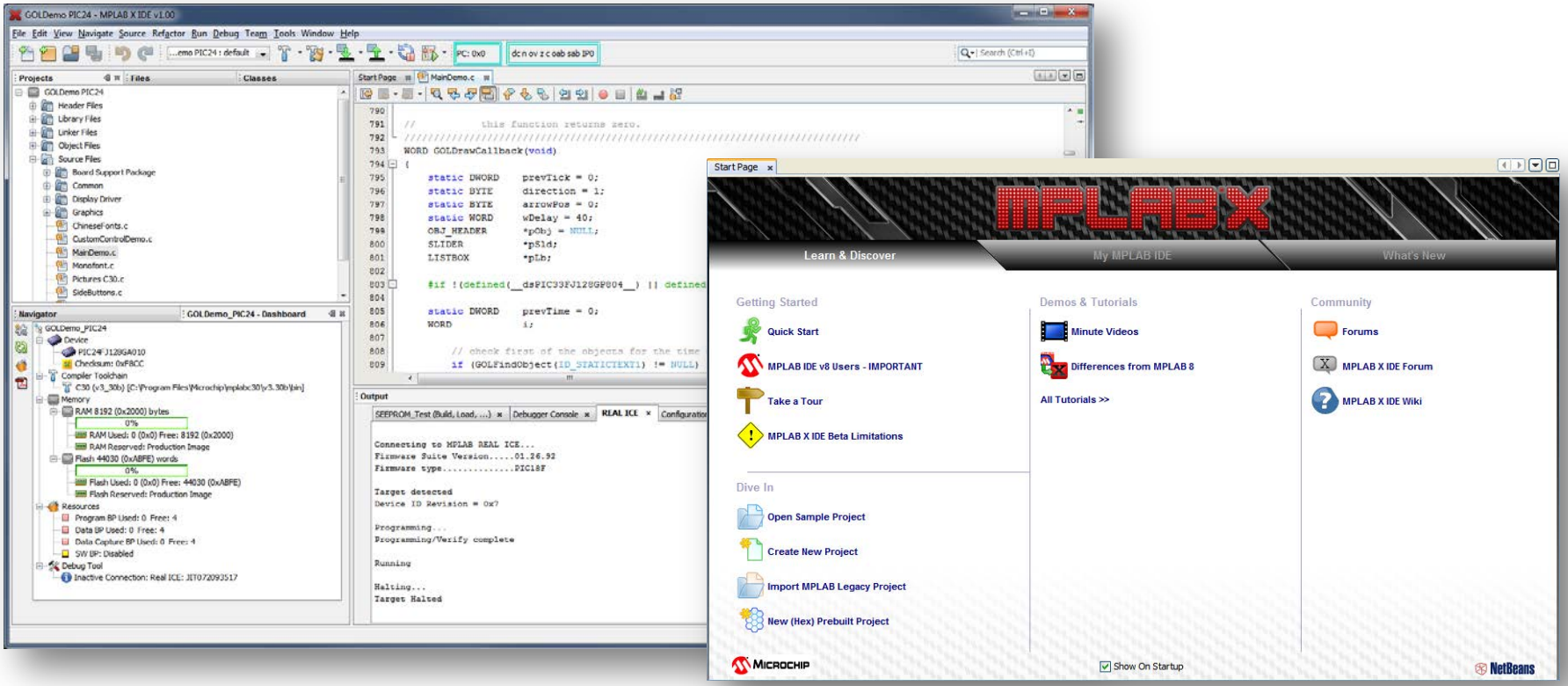
# MPLAB IDE

- Microchip提供的整合式開發環境, 支援全系列8-Bits, 16-Bits及32-Bits的MCU。所有的MCU都可透過相同的環境開發。
- 可整合各式的組譯器/編譯器(MPLAB C, Hi-TECH C, etc.),開發工具(PICkit3, ICD3, Real ICE, etc..) 。



# MPLAB X IDE

- 新一代的整合式開發環境, 可支援全系列MCU, 擁有需多個便利的功能與特徵。Java Base, 可以跨平台。
- 不支援ICD2, ICE2000/4000, ProMATE II, PICStart Plus.



MICROCHIP
*Regional Training Centers*

# Debug/Program Tools

Features/Speed/Trace

**PICkit™ 3**
Full Speed USB HID,
Run, Halt, SS,
Simple Breakpoints,
Program, Read
All of Microchip's Flash PIC®
MCU and dsPIC DSCs

**USD $69.0**


**MPLAB® ICD 3**
High Speed USB 2.0,
Run, Halt, SS
Software Breakpoints,
Complex
Breakpoints,
Program, Read,
All of Microchip's Flash PIC®
MCUs and dsPIC® DSCs

**USD $200.0**

**MPLAB REAL ICE™ Emulator**
High Speed USB2.0
Run, Halt, SS
Software Breakpoints,
Complex
Breakpoints/Triggers,
Stopwatch,
Program, Read,
Real-Time Watch,
Log, Trace,
Logic Probes,
Performance Pak
All of Microchip's Flash PIC
MCUs and dsPIC DSCs

**USD $500.0**

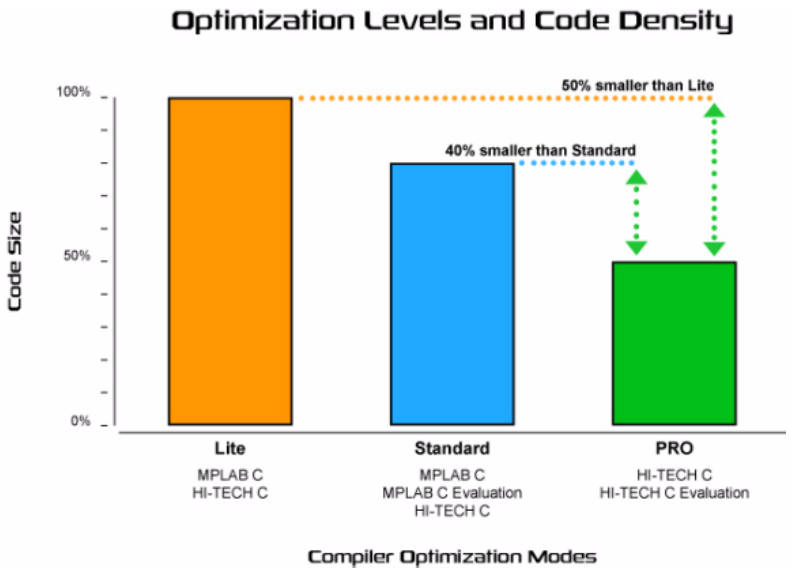
MICROCHIP
Regional Training Centers

# MPLAB C30

- 支援Microchip 16-Bits MCU的C Compiler 。架構於GCC Compiler, 相容於ANSI 1989規範。
- 支援以下16-Bits MCU系列:
  PIC24F(16 MIPS),PIC24H(40 MIPS),PIC24E(60 MIPS),
  dsPIC33F(40 MIPS),dsPIC30F(30 MIPS),dsPIC33E(60 MIPS) 。
- MPLAB C30可整合於 MPLAB IDE中。
- 提供標準C Functions(printf, scanf, etc..),週邊函式庫,數學運算函數等。
- 可至Microchip網站, 免費取得。
  http://www.microchip.com/c30

# MPLAB C30 版本

- MPLAB C30提供三種不同版本的Compiler:
  正式版(Standard), 精簡版(Lite), 評估版(Evaluation) 。
- 正式版(Standard)
  付費版本,擁有完整編譯功能及最佳化(Optimizations)的功能,可跟精簡版相比,最高可以減少20%的程式碼空間(Code Size)。
- 精簡版(Lite)
  免費版本,擁有完整編譯功能, 與正式版唯一的差異,就是沒有完整的最佳化功能,只擁有Level 1最佳化功能。
- 評估版(Evaluation)
  安裝後60天內,具有標準版的所有功能,60天後自動變成精簡版。

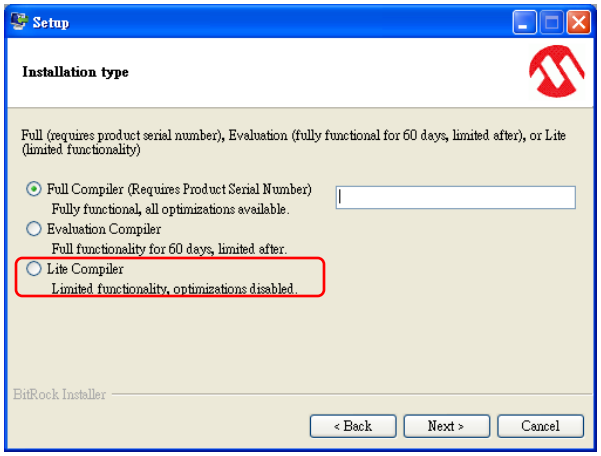


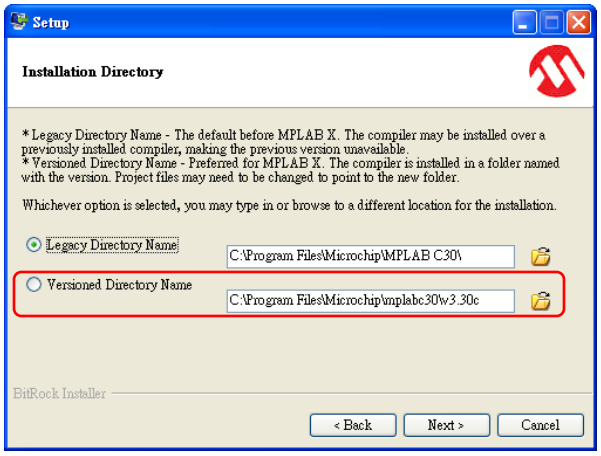
# MPLAB C30 版本

- **MPLAB C30正式版(Standard)分為三種套件**
- **MPLAB C30 for PIC24 and dsPIC DSCs**
  適用於PIC24系列與dsPIC DSCs系列
  (SW006012) USD $895.0
- **MPLAB C30 for dsPIC DSCs only**
  (SW006013) $495.0
  適用於dsPIC DSCs系列
- **MPLAB C30 for PIC24 only**
  (SW006014) USD $495.0
  適用於PIC24系列

# MPLAB C30 的安裝

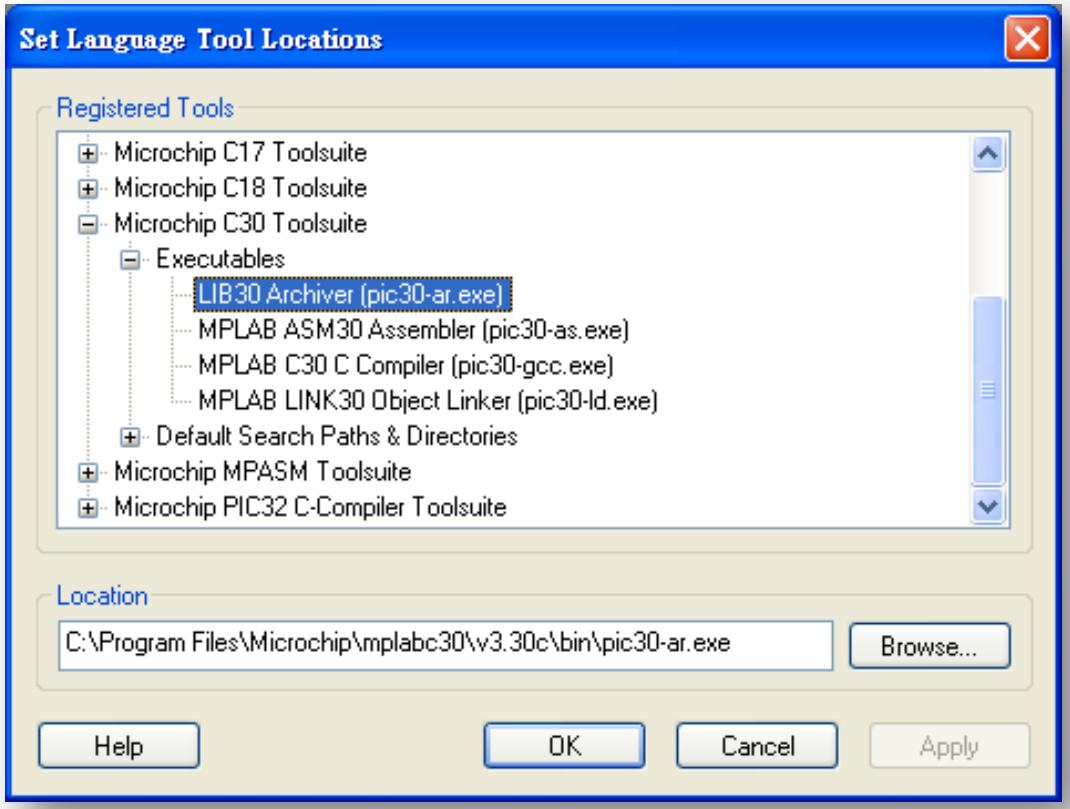
- 先自Microchip網站取得MPLAB C30的安裝檔。
- 目前版本安裝流程,已將正式版(Standard), 精簡版(Lite), 評估版(Evaluation)整合在同一安裝檔中。版本選擇在安裝過程中指定即可。
- 安裝時,可選擇安裝路徑,建議選擇安裝在"依版本並命名"的路徑下(Versioned Directory Name),
  (C:\Program Files\Microchip\MPLABC30\vx.x\) 。
- 若有舊版本安裝於
  C:\pic30_tools\,
  C:\Program Files\
  Microchip\
  MPLAB C30\,
  建議將其移除。



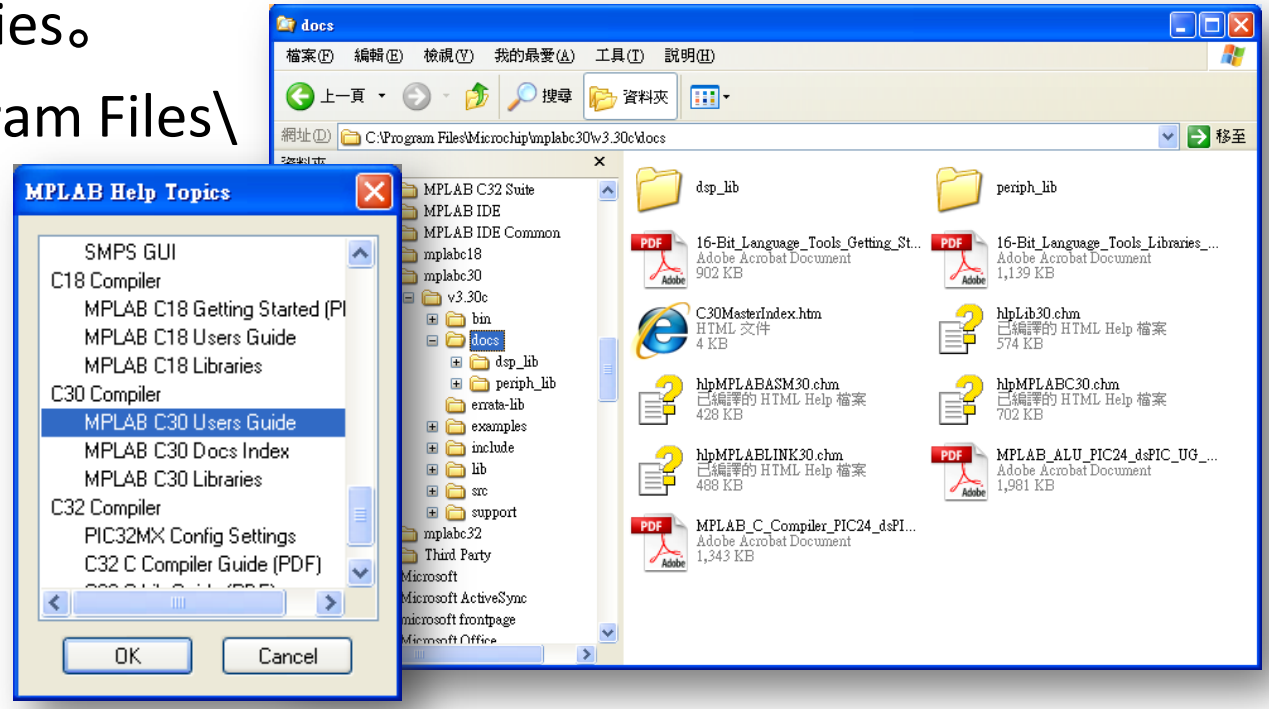
# MPLAB C30 的安裝

- 安裝成功後可MPLAB IDE中的功能表選擇Project\Set Language Tool Location,來檢查Microchip C30 Toolsuite的安裝情形與檔案路徑。



# MPLAB C30 的相關說明文件

- MPLAB C30的相關文件,可以在MPLAB C30中的功能表選擇Help\Topics,再由Topics的列表中尋找MPLAB C30的相關文件。
- 目前列表有
  MPLAB C30 Users Guide, MPLAB C30 Docs Index, MPLAB C30 Libraries。
- 也可以在C:\Program Files\Microchip\mplabc30\vx.x\docs\裡找到許多相關的文件。



MICROCHIP
Regional Training Centers

# MPLAB C30's Data Type

- 資料型別(Data Type)是用來定義資料存放時所佔記憶體的大小及其被處理的方式,MPLAB C30支援以下幾種資料型別(Data Type):

整數(Integer)資料型別

| Type | Bits | Min | Max |
|---|---|---|---|
| char, signed char | 8 | -128 | 127 |
| unsigned char | 8 | 0 | 255 |
| short, signed short | 16 | -32768 | 32767 |
| unsigned short | 16 | 0 | 65535 |
| int, signed int | 16 | -32768 | 32767 |
| unsigned int | 16 | 0 | 65535 |
| long, signed long | 32 | $-2^{31}$ | $2^{31}-1$ |
| unsigned long | 32 | 0 | $2^{32}-1$ |
| long long**, signed long long** | 64 | $-2^{63}$ | $2^{63}-1$ |
| unsigned long long** | 64 | 0 | $2^{64}-1$ |

** ANSI-89 extension

浮點數(Float)資料型別

| Type | Bits | E Min | E Max | N Min | N Max |
|---|---|---|---|---|---|
| float | 32 | -126 | 127 | $2^{-126}$ | $2^{128}$ |
| double* | 32 | -126 | 127 | $2^{-126}$ | $2^{128}$ |
| long double | 64 | -1022 | 1023 | $2^{-1022}$ | $2^{1024}$ |

E = Exponent
N = Normalized (approximate)
* double is equivalent to long double if -fno-short-double is used.

MICROCHIP
Regional Training Centers

# 變數(Variable)

- 變數(Variable)
  變數可任意的讀取或寫入,因為必須可以讀寫,因此變數必須存放於資料記憶體(Data Memory)中。
- 變數要先宣告後才能使用。
  Ex:
  int X,Y ;
  unsigned char ASCII_Char ;
  long MyID = 0x1234 ;  // 宣告並指定初值
- 如果變數會用於中斷服務程式中或者不希望被最佳化處理則需加上volatile修飾語
  Volatile int  VarForISR;

MICROCHIP
Regional Training Centers

# 常數(Constant)

- 常數(Constant)
  常數只能讀取但無法改變(寫入),因為只需要讀不需要寫,因此常數可以存放於資料記憶體(Data memory)或程式記憶體(Program Memory)中。
- 常數跟變數相同,要先宣告後才能使用。
  Ex:
  const int X = 0 ,Y = 1;
  const  long MyID = 0x1234 ; // 宣告並指定初值
- MPLAB C30的預設,如果宣告常數時,會將該資料放至在程式記憶體(Program Memory)空間,並自動透過PSV來存取。
- 如果想改為存放在資料記憶體(Data memory),可在MPLAB IDE中,功能表Project\Build Options\Project中的MPLAB C30選項選擇constants-in-memory memory model。
- 由於PSV的每個Page為32K Bytes,因此宣告的常數大小不能超過32K Bytes。

MICROCHIP
Regional Training Centers

# SFR 的使用

- 每個特殊功能暫存器(Special Function Register)都有特定的名稱與位址。MPLAB C30為了使用SFR,必需使用MCU的標頭檔(Header File)來定義其名稱與結構,並利用連結檔(Linker Script File, *.gld)來定義其位址。
- 要使用到SFR時,必須先含入(include)對應MCU的標頭檔(Header File)。例如:

  #include <p33fj128mc804.h>

  #include <p24fj256gb106.h>
- 標頭檔(Header File)的檔案命名方式並非所有使用者都很熟悉,因此建議可直接含入(include)通用標頭檔(Generic Header File),由MPLAB C30自己來尋找正確的標頭檔(Header File)。

  #include <p33fxxxx.h> // for dsPIC33

  #include <p24fxxxx.h> // for PIC24

MICROCHIP
Regional Training Centers

# SFR 的存取

- 透過MCU標頭檔(Header File)與連結檔(Linker Script File, *.gld)的定義,因此可以直接在C中,存取SFR。例如:

  TRISD= 0xFFFE;

  LATD = 0x0001;

- MCU標頭檔(Header File)中,另外定義了各個SFR的結構型態,因此也可以以結構的方式SFR內的特定成員(*SFRNAME*bits.*BITNAME*)。例如:|

  TRISDbits.TRISD0 = 0;

  LATDbits.LATD0 = 1;

- TRISD及TRISDbits都參考到同一SFR的位址(透過Linker Script指定)。因此操作TRISD或TRISDbits對得到相同的結果。
- 詳細的結構型態與連結檔的定義,建議直接打開標頭檔(Header File)與連結檔(Linker Script File, *.gld)來觀察看看。

# MCU標頭檔 (Header File)

- p24fj256gb106.h MCU標頭檔(Head File)的內容片段

```
extern volatile unsigned int  LATD __attribute__((__sfr__));

typedef struct tagLATDBITS
{
   unsigned LATD0:1;
   unsigned LATD1:1;
   unsigned LATD2:1;
   unsigned LATD3:1;
   unsigned LATD4:1;
   unsigned LATD5:1;
   unsigned LATD5:1;
   unsigned LATA7:1;
   unsigned LATA8:1;
   unsigned LATA9:1;
   unsigned LATA10:1;
   unsigned LATA11:1;
}LATDBITS;
extern volatile LATDBITS LATDbits __attribute__((__sfr__));
```

| File Name | Addr | Bit 15(1) | Bit 14(1) | Bit 13(1) | Bit 12(1) | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | All Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRISD | 02D8 | | | | | TRISD11 | TRISD10 | TRISD9 | TRISD8 | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | FFFF |
| PORTD | 02DA | | | | | RD11 | RD10 | RD9 | RD8 | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | xxxx |
| LATD | 02DC | | | | | LATD11 | LATD10 | LATD9 | LATD8 | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 | xxxx |
| ODCD | 02DE | | | | | ODD11 | ODD10 | ODD9 | ODD8 | ODD7 | ODD6 | ODD5 | ODD4 | ODD3 | ODD2 | ODD1 | ODD0 | 0000 |

# 連結檔 (Linker Script File)

- p24fj256gb106.gld 連結檔(Linker Script File)的內容片段:

```
 TRISD       = 0x2D8;
_TRISD       = 0x2D8;
_TRISDbits   = 0x2D8;
 PORTD       = 0x2DA;
_PORTD       = 0x2DA;
_PORTDbits   = 0x2DA;
 LATD        = 0x2DC;
_LATD        = 0x2DC;
_LATDbits    = 0x2DC;
```

- 連結檔(Linker Script File)中除了定義SFR的位址外,也定義了資料與程式記憶體的大小,位置,重置向量(Reset Vector),中斷向量表的位址等。
  (中斷架構章節會詳細說明)

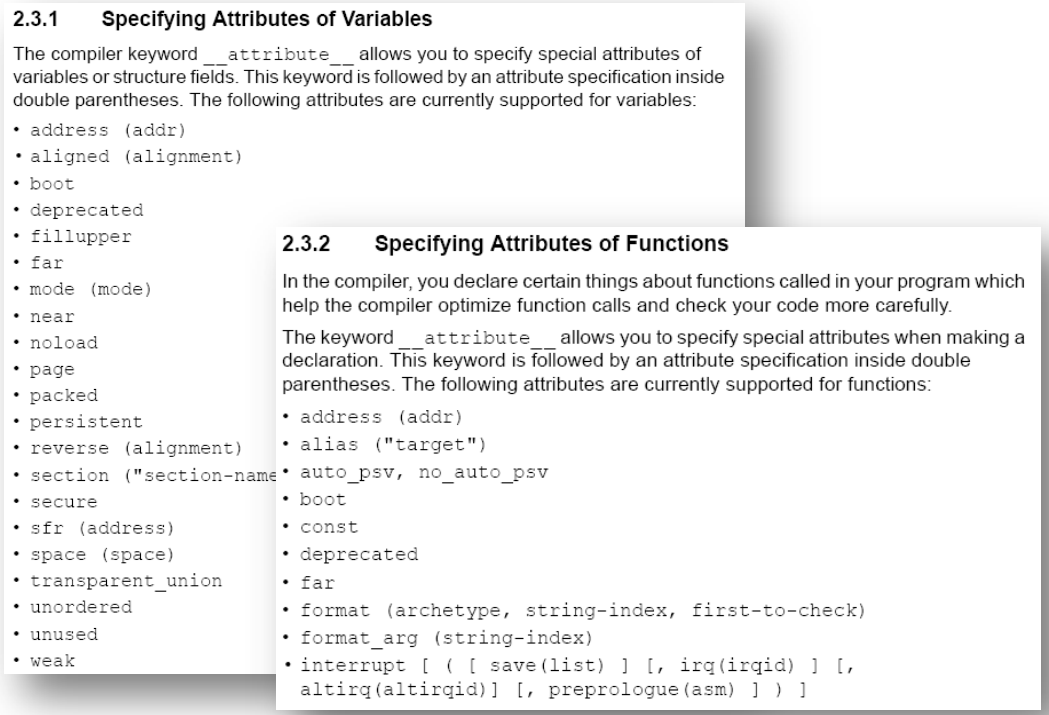
```
MEMORY
{
   data(a!xr):ORIGIN = 0x800,LENGTH = 0x4000
   reset:ORIGIN = 0x0,LENGTH = 0x4
   ivt:ORIGIN = 0x4,LENGTH = 0xFC
   _reserved:ORIGIN = 0x100,LENGTH = 0x04
   aivt:ORIGIN = 0x104,LENGTH = 0xFC
   program(xr):ORIGIN = 0x200,LENGTH = 0x2A9F6
   …
}

.ivt __IVT_BASE :
{
   LONG( DEFINED(__ReservedTrap0) ?
   ABSOLUTE(__ReservedTrap0):
   ABSOLUTE(__DefaultInterrupt));
   ….
}
```

MICROCHIP
Regional Training Centers

# 屬性(Attribute)

- Attribute是MPLAB C30的保留字。Attribute可用來指定變數或函數的特定屬性, 如:
  指定放置在特定區域, 特定位址。
  指定放置變數時對齊或保留的情形。
  中斷服務函式的指定。
  ...
- Attribute的詳細用法可以參考MPLAB C30的User Guide。



2.3.1 Specifying Attributes of Variables

The compiler keyword __attribute__ allows you to specify special attributes of variables or structure fields. This keyword is followed by an attribute specification inside double parentheses. The following attributes are currently supported for variables:

- address (addr)
- aligned (alignment)
- boot
- deprecated
- fillupper
- far
- mode (mode)
- near
- noload
- page
- packed
- persistent
- reverse (alignment)
- section ("section-name
- secure
- sfr (address)
- space (space)
- transparent_union
- unordered
- unused
- weak

2.3.2 Specifying Attributes of Functions

In the compiler, you declare certain things about functions called in your program which help the compiler optimize function calls and check your code more carefully.

The keyword __attribute__ allows you to specify special attributes when making a declaration. This keyword is followed by an attribute specification inside double parentheses. The following attributes are currently supported for functions:

- address (addr)
- alias ("target")
- auto_psv, no_auto_psv
- boot
- const
- deprecated
- far
- format (archetype, string-index, first-to-check)
- format_arg (string-index)
- interrupt [ ( [ save(list) ] [, irq(irqid) ] [, altirq(altirqid)] [, preprologue(asm) ] ) ]

MICROCHIP
Regional Training Centers

# 變數的屬性(Attribute)設定

- **__attribute__ ( ( space ( *area* ) ) )**
  *auto_psv*:Compiler managed PSV
  *psv*:User managed PSV
  *xmemory*:X Data memory
  *ymemory*:Y Data memory
  *eedata*:EEPROM memory
  *dma*:DMA memory
  *eds*:Extended Data memory
- **__attribute__ ( ( *parameter* ) )**
  aligned( ):Start align boundary
  reverse( ):End align boundary
  near:Near data (First 8K Bytes)
  far:Far Data(64K Bytes)
  address( ):Start address
  persistent:Uninitialized on warm reset

MICROCHIP
Regional Training Centers

# 變數的屬性(Attribute)設定

- Example:
  安排在EEPROM區域
  int a __attribute__( ( *space( eedata )* ) );
  安排在前面8K Bytes的區域
  int b __attribute__( *near* );
  安排在資料記憶體0x1000的位置
  int c __attribute__( ( *address( 0x1000 )* ) );
  對齊可被32整除的記憶體位置
  int d __attribute__( *( aligned( 32 )* ) );
- 有關 attribute 的更多說明及範例,建議參考 MPLAB C30 Users Guide第2章的說明。

MICROCHIP
Regional Training Centers

# 函式的屬性(Attribute)設定

- __attribute__ ( ( *address( )* ) )
  __attribute__( ( *interrupt , auto_psv* ) )
- Example:
  安排在程式記憶體0x1000的位置
  void function( void ) __attribute__( ( *address( 0x1000 )* ) )
  {
  }
  指定function為ADC1的中斷服務函式
  void _ADC1Interrupt( ( void ) __attribute__( ( *interrupt , auto_psv* ) )
  {
  }