



---

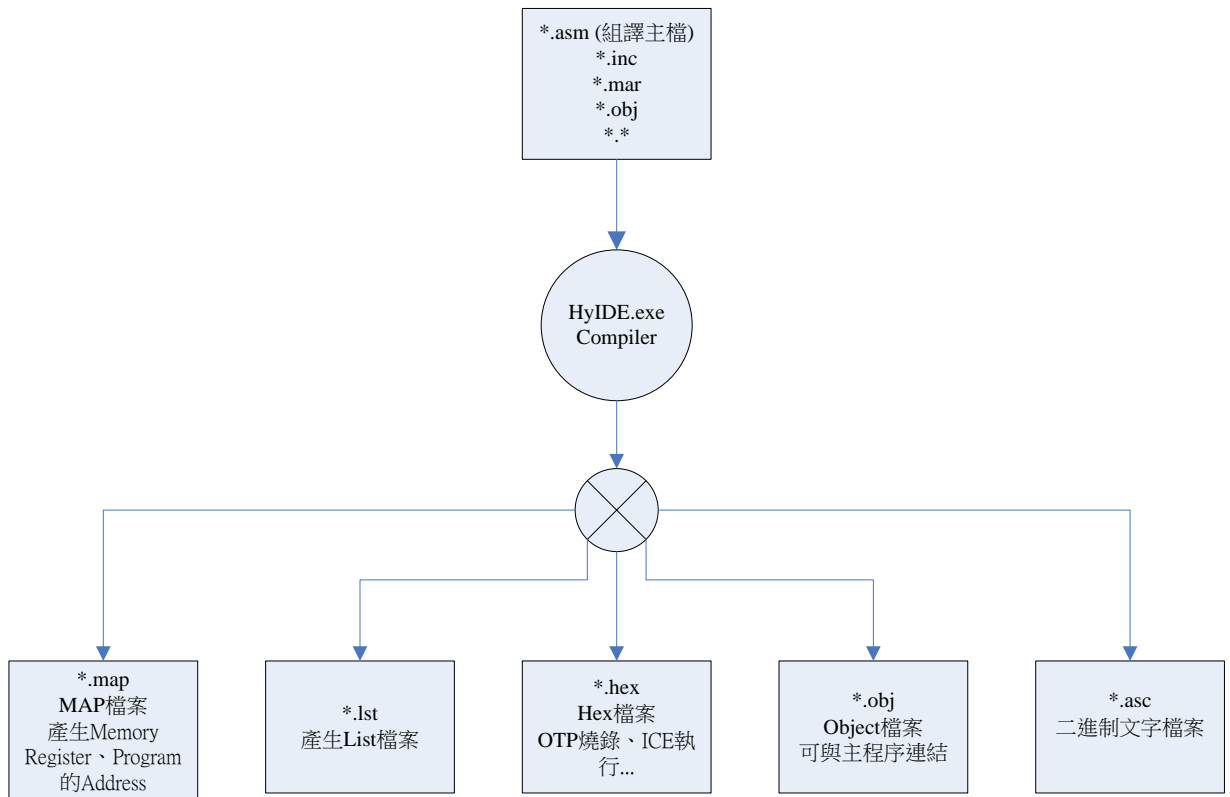
# HY-MCU Compiler 說明

## 目 錄

1. COMPILER .....	3
2. COMPILER 文件檔案限制.....	4
3. 虛擬指令 (PSEUDO).....	5
4. OBJECT CODE .....	11
5. 組譯 與 組譯&&除錯的差異.....	12
6. 錯誤訊息 .....	12

## 1. Compiler

檔案產生流程



### 1.1 組譯主檔

程式執行主檔，Compiler 主要的檔案名稱，副檔名除 obj、hex、asc、lst、map、msk、pro、err 以及 ifo 外，皆可被 Compiler 接受。

### 1.2 Object 檔

目標連結檔案，副檔名為\*.obj，可供主程序引用函式。

### 1.3 MAP 檔

參數的產生檔案，副檔名為\*.map，參數包括，所有 EQU、DS 定義 Memory 或 Register 的 Address，Object 檔案內的 Label 的 Address。

### 1.4 List 檔

產生 Compiler 後的所有資訊，程序的行數，Address，機器碼，主程序。

### 1.5 Hex 檔

產生 Hex 檔案，供 ICE 程序模擬，OTP 燒錄。

### 1.6 Asc 檔

產生機器碼的二進制文字檔案。

## 2. Compiler 文件檔案限制

1. 文件檔案最大為 128k byte，如果超過最大限制，Compiler 並不會有警告發生，但文件會被毀壞，所以使用者須注意文件檔案最大限制。
2. 當文件檔案超過 128k 時，需要用 Include 的方式來分開檔案，但是每一個文件檔案的限制也是 128K，使用者須注意。
3. 文字不論大小寫，只要字組相同，接辨識為相同的文字字組。
4. Include XXXX.obj 檔時，必須放置文件最後方。
5. Call .obj 內部的函數時，於該行程序下方，需補上 nopf 函數 這行指令

例如：

```
CALL HY17P52WR3
NOPF    HY17P52WR3
```

6. 檔案名稱長度不可超過 128 字元，包括 Path。

例如：

```
C:\Example\ASM\example.asm ← 此字串長度不可超過 128 字元
```

7. 編輯程式每一行的文字長度限制 160 字元。

例如：

```
MVF    TEMPABLE , F, BANK ← 此字串長度不可超過 160 字(包括控格...)
```

8. Label、Memory、Register 的定義文字長度最大 32 字元

例如：

```
Label1:          ← 此字串長度不可超過 32 字元
ECK    equ    80h ← 'ECK' 不可超過 32 字元
```

9. Label、Memory、Register 最多可定義 8192 個(Total)

10. Macro 最大數目為 4096 個

11. Macro 定義最大的參數為 32 個。

例如：

```
Exm Macro A,B,C..... ← A,B,C 的參數不可超過 32 個
```

12. Macro 內引用另外的 Macro 最大層數不可超過 32 層

例如：

```
ANB Macro CC
    Exm
ENDM
ANB1 Macro CC
    ANB
ENDM
ANB2 Macro CC
    ANB1
ENDM
.....
```

```

ANBn Macro CC
      ANBn-1      ← 不可超過 32 層
ENDM
    
```

### 3. 虛擬指令 (Pseudo)

定義程式、記憶體或特殊寄存器的 Address 時，需用到虛擬指令來定義，但此指令並不佔記憶體或程序的空间(除了定義程式的虛擬指令'DB'與'DW'例外)。

Compiler 共有下列虛擬指令：

INCLUDE	開啟另一個文件
ORG	定義程序的 Address
END	程序結束
MACRO	定義模組
ENDM	模組定義結束
EQU	定義立即數
DB	定義一個 Byte 的程序
DW	定義一個 Word 的程序
MEMAR	宣告 SRAM 的 Address
DS	宣告 SRAM 的的長度
EXTENR	引用外部參數
GLOBAL	提供參數給外部程式使用
SET	保留
LOCAL	保留
UP	Label or address / 0x10000
HIGH	Label or address / 0x100
LOW	Label or address mod 0x100
DEFINE	賦予數值給 IF 內事件的參數
IF	條件與句，判斷是否組譯 ENDIF 之前的語句
ELSE	否則，配合 IF 的用語
ENDIF	結束 IF

#### 1. INCLUD

開啟主程序內所包括的檔案，檔案內容可以為 Marco 檔案、Object 檔案、定義檔案以及 Assemble 檔案。

如果 INCLUDE 的是 Object 或 Assemble 檔案，Compiler 會按照順序，將指令安排在 ROM 的

Address 內。

例如：

**SYSINI.asm** 的內容為

```
CLRF 080h
CLRF 081h
```

.....

**MAIN.asm** 的內容為

```
ORG 0000h
RJ START
INCLUDE SYSINI.asm
ORG 0100h
```

START:

....

Compiler **MAIN.asm** 後

Address	Program
0000h	RJ 100h
0001h	CLRF 080h
0002h	CLRF 081h

.....

0100h

....

如果是 **Marco** 或定義檔案，是不會被安排在 ROM 的 Address 中，除非在程序有呼叫到 **Marco**，才會將 **Marco** 內的程序安排在相對應的 ROM Address。

例如：

**Def.mar** 的內容為

```
W EQU 0
F EQU 1
ACCE EQU 0
BANK EQU 1
ADDWORD MARCO A, B
LBSR B
MVF B, W, BANK
LBSR A
ADDF A, F, BANK
LBSR B
MVF B, W, BANK
LBSR A
ADDC A, F, BANK
```

ENDM

**MAIN.asm** 的內容為

```
INCLUDE Def.mar
BUFA EQU 080h
BUFB EQU 102h
ORG 0000h
GOTO START
```

....

```
ORG 0100h
```

START:

```
MVL 012h
MVF BUFA, F, ACCE
```

```
MVL      034h
MVF      BUFA+1, F, ACCE
MVL      056h
LBSR     BUFB
MVF      BUFB, F, BANK
MVL      078h
MVF      BUFB+1, F, ACCE
ADDWORD  A, B
....
```

Compiler **MAIN.asm** 後

```
Address      Program
0000h        GOTO   START
.....
0100h        MVL    12h
0101h        MVF    80h, 1, 0
0102h        MVL    34h
0103h        MVF    081h, 1, 0
0104h        MVL    56h
0105h        LBSR   1
0106h        MVF    02h, 1, 0
0107h        MVL    78h
0108h        MVF    03h, 1, 0
0109h       LBSR   1
010Ah       MVF    02h, 0, 1
010Bh       LBSR   0
010Ch       ADDF   80h, 1, 1
010Dh       LBSR   1
010Eh       MVF    03h, 0, 1
010Fh       LBSR   0
0110h       ADDC   81h, 1, 1
....
```

## 2. ORG

指定程式記憶體的位址。

例如：

```
ORG      0000h
RJ       Begin
....
ORG      0100h
Begin:
MVL      10h
....
```

Compiler 後

```
0000h    RJ    101h
....
0100h    MVL   10h
```

## 3. END

程式結束。

當程式中如果有 END 的虛擬指令，則 Compiler 就會立即中止，不再對 END 以下的程式組譯。

## 4. MACRO

巨集指令。

可將指令巨集起來，方便程式引用。

例如：

```
MOVFW  MACRO  A, B
MVF  A, 0, B
ENDM
```

```
MOVWF  MACRO  A, B
MVF  A, 1, B
ENDM
ORG  0000h
MOVFW  80h,0
MOVWF  10h,1
```

#### Compiler 後

```
0000h  MVF  80h, 0, 0
0001h  MVF  10h, 1, 1
```

## 5. ENDM

巨集指令結束。

宣告 MACRO 後，需有 ENDM 做結束。

## 6. EQU

定義立即數。

可利用 EQU 來宣告 SRAM 的位址。

例如：

```
COUNT  EQU  80h
```

COUNT 的位址等於 80h

## 7. DB

定義 1 個 Byte 的程序。

DB 需要雙數個參數，如果是單數，則會在前面補上 00h，來合成一個 WORD。

例如：

```
ORG  100h
DB  012h
DB  053h,046h
```

#### Compiler 後

```
0100h  0012h
0101h  05346h
```

## 8. DW

定義 1 個 WORD 的程序。

例如：

```
ORG  100h
DW  01234h
```

#### Compiler 後

```
0100h  01234h
```

## 9. MEMAR

宣告 SRAM 的 Address。



Compiler 會由 MEMAR 後面的參數來定義 SRAM 的起始位址。

## 10. DS

宣告 SRAM 的的長度。

例如：

```
MEMAR    080h      ← SRAM 由 080h 開始編排
TEMP     DS      16  ← 定義 TEMP 有 16 byte (由 080h 到 08Fh)
COUNT   DS       2  ← 定義 COUNT 為 2 個 Byte · 位址
          090h,091h
BUF      DS       1   ← 定義 BUF 為 1 個 Byte · 位址為 092h
```

## 11. EXTERN

引用外部參數。

在建立 Object Code 時，如果要引用外部程序所定義的參數，不須在本程序中定義，可藉由 EXTERN 的宣告來完成。

例如：

```
          EXTERN    TEMP1, TEMP2
          GLOBAL    ADDBLK

          LEN      EQU    3
ADDBLK:
          MVF      TEMP2, 0, 0
          ADDF     TEMP1, 1, 0
          .....
```

## 12. GLOBAL

提供參數給外部程式使用。

在建立 Object Code 時，可以將本程序中的參數或子程序釋出，提供給外部的程序引用。

## 13. UP

取數值的最高位元 → 數值 / 0x10000

例如：

```
MVL     UP Label1 → UP Label1 = 0x1000 / 0x10000 = 0x00
....
ORG     01000h
Label1:
```

## 14. HIGH

取數值的高位元 → 數值 / 0x100

例如：

```
MVL     HIGH Label1 → HIGH Label1 = 0x1000 / 0x100 = 0x10
....
ORG     01000h
Label1:
```

## 15. LOW

取數值的低位元 → 數值 MOD 0x100

例如：

```
MVL     LOW Label1 → LOW Label1 = 0x10F2 MOD 0x100 = 0xF2
....
ORG     010F2h
```

Label1:

## 16. DEFINE 、IF、ELSE and ENDIF

DEFINE：賦予數值給 IF 事件的參數。

條件語句 IF

用法：

```
IF 事件
    內容 1
ELSE
    內容 2
ENDIF
```

說明：

- DEFINE 的變數與 EQU 定義的變數名稱可以相同(兩者無關聯)
- 當事件成立執行內容 1，否則執行內容 2
- 事件成立可以為"TRUE" OR "1"
- 事件可以為 DEFINE 中的變數
- 可以在 MACRO 內用

限制：不允許 IF 巢路(NEST)

例如：

```
Define    FSR0 = 0
Define    FSR1 = 1
Define    H08B = 0
```

```
MOVLf    MACRO  RAMLabel , FSR
    MVL    RAMLabel
    IF    FSR = 0
        MVF    FSR0L, F, ACCE
    ELSE
        MVF    FSR1L, F, ACCE
    ENDIF
ENDM
```

```
ORG      0
JMP      BEGIN
```

```
.....
.....
BEGIN:
    MOVLf    TEMP, FSR0
    .....
    .....
FSR0L    EQU    010h
FSR0H    EQU    012h
W        EQU    0
```

```

F          EQU    1
ACCE      EQU    0
BANK      EQU    1
TEMP      EQU    080h
TEMP1     EQU    081h
    
```

組譯後

```

          ORG    0
          JMP    BEGIN
          .....

          .....
BEGIN:
          MVL    TEMP
          MVF    FSR0L, F, ACCE
          .....
          .....
    
```

## 4. Object Code

- 程式碼以二進制形式存放。
- 透過虛擬指令 `Global` 產生函式給主程序引用。
- 透過虛擬指令 `Extern` 來引用外部的參數或程序
- 主程序利用 `Include` 的方式來將 `Object Code` 包含進來。
- 透過 `MAP` 檔案可以知道哪些函式可以被引用呼叫。

---

### Memory , Register and Const

---

Name	Location	Address
	----	-----
	INDF0	Const 0x0000
	POINC0	Const 0x0001
	PODEC0	Const 0x0002

---

### Local Program Address

---

Name	Location	Address
	----	-----
	RESET	Program 0x0000
	MAINLOOP	Program 0x000C
	MAINLOOP_DATA	Program 0x0025
	MAINLOOPAA	Program 0x0100
	TEST	Program 0x0265

---

### Extern Program Address

```

=====
Name      Location  Address
-----
          CLEAR   Program  0x0102
          MULTIPLY Program  0x0106
          DIVIDE45 Program  0x0128
          DIVIDE   Program  0x0145

HTOB      Program  0x0186
          BTOH    Program  0x01B9
          SQRT    Program  0x01F0
          TT1     Program  0x025F
          TEMM    Program  0x025C
    
```

## 5. 組譯 與 組譯&&除錯的差異

### 組譯

1. 如果有宣告外部參數(EXTERN)，而在程式段有引用到外部參數，組譯皆能成功。
2. 組譯不會判斷 RAM 或 Program 是否有超出晶片規格範圍，因此選擇任一晶片皆可組譯。

### 組譯&&除錯

1. 在程式段有引用到外部參數(EXTERN)，皆無法組譯，並顯示錯誤訊息。
2. 組譯會檢查 RAM 或 Program 是否有超出晶片規格範圍，只要超過 Program Size 或引用的 RAM 不存在皆會顯示錯誤訊息。

## 6. 錯誤訊息

1. ERROR[Code 0]: Unknown error.  
指令名稱錯誤。  
指令名稱錯誤或是沒有在 Macro 中定義名稱。
2. ERROR[Code 1]: Couldn't open source file  
找不到 Source 檔案。
3. ERROR[Code 2]: Couldn't open file  
Include 檔案名稱錯誤，或找不到檔案名稱。
4. ERROR[Code 3]: Temp file creation error  
暫存檔案開啟錯誤。  
重複開啟檔案。
5. ERROR[Code 4]: Duplicate label or redefining symbol  
重複定義 Symbol 或 Label
6. ERROR[Code 5]: Undefined Symbol

未定義 Symbol 或 Label

7. ERROR[Code 6]: Out of memory...  
Register、Memory or Label 定義超過 8192 個，或 Macro 宣告超過 4096...
8. ERROR[Code 8]: Illegal Symbol  
Register、Memory or Label 定義名稱與指令名稱相同
9. ERROR[Code 9]: Illegal argument  
LBSR 參數大於 16
10. ERROR[Code 10]: Overwriting previous address contents.  
Address 位置安排重疊  
例如：

```
ORG 0
Nop
Nop
Nop
Nop
JMP Start
ORG 0004H
```
11. ERROR[Code 11]: Addresses above .....  
程式超過 ROM Size
12. ERROR[Code 13]: Missing argument...  
指令後面的參數不足  
例如：

```
MVF 080h
```
13. ERROR[Code 14]: Too many arguments...  
指令後面太多參數  
例如：

```
CLRF 080h, 1, 0
```
14. ERROR[Code 15]: Undeclared variable  
未宣告的變數
15. ERROR[Code 16]: Illegal Macro...  
不合法的 Macro
- ~~16. ERROR[Code 17]: Illegal Condition  
無效的狀態~~
- ~~17. ERROR[Code 18]: Out of Range...  
超過 Range~~
18. ERROR[Code 19]: Macro nested too deep...  
Macro 內再引用的 Marco 巢路超過 32 層
19. ERROR[Code 20]: Circular file reference...  
重複引用檔案，或是 INCLUDE 後面沒有檔案名稱
20. ERROR[Code 21]: Circular macro reference...  
重複定義 Macor
- ~~21. ERROR[Code 22]: Use the software illegally~~

## 非法使用軟件

22. ERROR[Code 23]: Over Range

超過 RJ 或 RCALL 範圍

23. ERROR[Code 24]: illegally Memory or Register

定義到不存在的暫存器或記憶體

24. ERROR[Code 25]: illegally Bit

定義到不存在的 bit

例如

INTF1 - ADCIF - - TMAIF WDTIF E1IF E0IF

BSF INTF1, 7, 0

25. ERROR[Code 26]: Memory Over

定義超出記憶體範圍

## 7. 修訂記錄

以下描述本文件差異較大的地方，而標點符號與字形的改變不在此描述範圍。

---

版本	頁次	變更摘要
V01	All	新增
V02	04	新增 Include .obj