大型程式專案管理技巧

欲速則不達定律

WICE 2.5 組譯器具有非常完整強悍的專案管理工具，能有效管理大型程式，如果是在多人協力開法軟體的情況下，更能發揮組織管理整合的優點，如果您認爲寫程式只要使用一個檔案就可以開始了，何必將檔案劃分的那麼細膩，自找麻煩，那您可能有所不知。如果您過去寫程式的習慣是一個檔案寫到底的方式，一定會發現程式愈來愈大，就愈難管理了，以 EM78862 來說，如果全部的程式都塞滿的話，光是有效程式碼就會超過一萬六千行，這還不包括註解及巨集定義，所以程式看起來當然很「壯觀」，相對地要維護這個程式可就不容易了，往往千頭萬緒不知從何找起。WICE 2.5 提供了相當好用的虛指令(pseudo instruction)，如果您經常寫大程式但是不會使用虛指令，那您可能只學了一半的功夫，大部分的時間都在做苦力，因爲組譯器其實可以讓您的工作效率提高幾十倍，而關鍵在於有沒有專案管理的觀念。

何謂專案管理?

就是透過一個專案計劃將程式分門別類的切分成數個檔案，由於檔案切割後每個檔案大小明顯變小，很容易掌握，同時使用組譯器所提供的虛指令將各個模組連結在一起，各模組使用共同的環境變數及同樣的巨集指令，以避免矛盾現象發生。專案管理很適合使用在多人共同開發軟體的環境下，能夠輕易地整合多人協同開發的程式，減少犯錯的機率，並且易於維護。

實例演練

現在筆者將以一個實際的例子說明如何將一個大程式一分爲五，如圖一所示，這是一組人準備要開發一個簡單的電子記事本程式，由主程式 main.dt 呼叫位於其他模組內的程式。除了主程式以外，另外有三個檔案，每個檔案內各有兩個副程式可供其他模組呼叫，此例中，分配給每個檔案程式記憶體 1KB 的空間，除此之外每個獨立檔案都必須透過 VAR.H 當作溝通路徑。在此種架構下，每個獨立的檔案就可由不同的人分工處理，每個人所使用到的變數名稱位址都必須向 VAR.H 取得，在這裡 VAR.H 就好像檔案總管一樣，每個人要使用的巨集指令、記憶體空間都得透過 VAR.H 取得,所以不會有所謂變數位址衝突問題。

現在問題是假設 MAIN.DT、CALC.DT、NOTE.DT、LCD.DT 都各由一個人負責完成(分別是甲、乙、丙、丁四人)，每個人的檔案中都有一些副程式模組要能讓別人使用，這四個人之間應該如何溝通才行？我們以 CALC.DT 程式爲例子說明，某乙在 CALC.DT 裡面寫了兩個副程式：SUB_C1 及 SUB_C2，這必須能讓甲、丙、丁

在他們的程式當中能夠呼叫得到，所以某乙必須使用假指令 PUBLIC 宣告他有兩個副程式模組 SUB_C1 及 SUB_C2 可供外部呼叫，同理，如果甲、丙、丁都有副程式要供別人呼叫的話，都得使用 PUBLIC 指令讓外界了解到他有哪些模組可以提供給別人使用。

相反地，如果我們要跟別人借東西也要先打聲招呼，才不致產生誤解，所以要用別人的副程式模組之前要用 EXTERN 假指令宣告，比方說甲要用乙的副程式 SUB_C1 及 SUB_C2，他應該在自己的程式中寫下：

```
EXTERN   SUB_C1
EXTERN   SUB_C2
```

在此還要補充說明一點，在同一個檔案中不可以同時對一個副程式模組宣告爲 PUBLIC 又同時宣告爲 EXTERN，這樣會造成組譯時的矛盾現象，自然不被組譯器所接受。

如果甲還要用丙的副程式 SUB_N1 及 SUB_N2，也必須寫下丙的副程式名稱。同理，其他的人要互相呼叫模組都必須遵守這個規則；如果副程式愈來愈多的時候，每個人勢必會在自己的程式中有愈來愈多的 EXTETRN 指令，且不斷重複，這樣子所產生的問題是讓人眼花撩亂，產生不少垃圾程式，所以我們要使用另外一個技巧避開這些重複性的宣告。

仔細觀察每個人都必須 INCLUDE 變數檔 VAR.H，那何不在這上面動腦筋呢？所以筆者讓每個檔案模組都掛上一個號碼牌，以 MAIN.DT 檔爲例子掛上：

```
H_MAIN   EQU    100
```

EQU 也是一個假指令，它和「==」的意義是相同的，都是指定一個常數或是位址，在這裡我們隨意爲 MAIN.DT 找一個名稱代替叫做 H_MAIN，後面的 100 是隨意指定的，只是用以區別名稱的差異。我們分別爲每個獨立檔案都掛上一個號碼牌，這個號碼牌我們將它放在 INCLUDE 指令之前，以便組譯器在組譯 VAR.H 時能夠識別，接著在 VAR.H 檔案中加入以下幾行程式：

```
IFNDEF H_MAIN
  EXTERN  SUB_M1, SUB_M2
ENDIF

IFNDEF H_LCD
  EXTERN  SUB_L1, SUB_L2
ENDIF
```

```
IFNDEF H_CALC
  EXTERN  SUB_C1，SUB_C2
ENDIF

IFNDEF H_NOTE
  EXTERN  SUB_N1
  EXTERN  SUB_N2
ENDIF
```

假指令 EXTERN 的用法可以用逗號區隔兩個副程式名稱，亦可分做兩行寫如上述最後一個寫法一樣，在這裡的重點應該說 IFNDEF 的用法，讀者亦可回顧條件式組譯的用法，它的意思是「若程式中未定義.....就組譯 IFNDEF 後面的運算式子」。如此以 MAIN.DT 來看，因爲 MAIN.DT 當中已經定義了 H_MAIN 這個常數，所以第一個 IFNDEF 將不會成立，也就是說當組譯器在組譯 MAIN.DT 時可以避免同時對 SUB_M1、SUB_M2 宣告爲 PUBLIC 又宣告爲 EXTERN 的窘境，同時後面三個 IFNDEF 都會成立(因爲在 MAIN.DT 中只看得到 H_MAIN 的宣告)，所以也達到引用其他副程式模組宣告的目的。

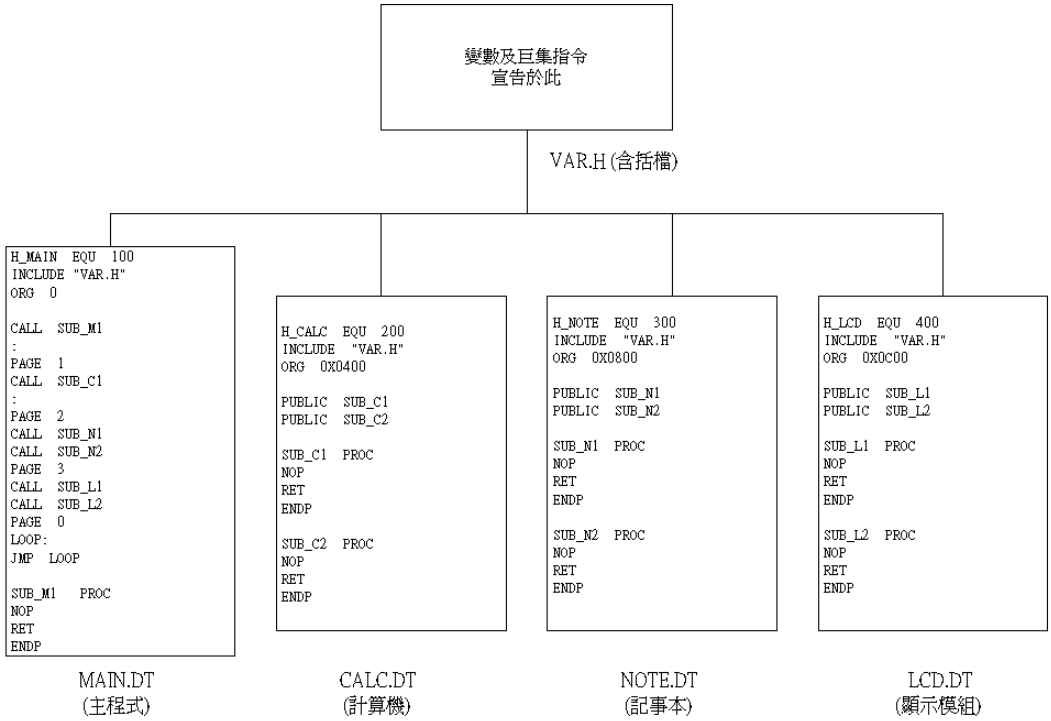
這樣解釋也許還有讀者不明白，爲何組譯器不會同時見到所有檔案的常數宣告，反而能用 IFNDEF 避開矛盾現象？這也算是使用專案管理的另一個好處，因爲四個程式其實組譯器是分成四次組譯的，每次組譯之後都會產生一個目的檔(*.BBJ)，最後在將全部的 BBJ 檔透過 LINKER 連結起來產生最後的燒錄檔.CDS，所以對組譯器而言可以減輕負擔，對程式設計者而言如果只修改程式的一小部份，組譯器僅針對被修改過的那個模組重新組譯，不必全部重來一次，可以節省不少時間，也能縮小除錯的範圍，組譯及連結的過程原理如圖 4-3 所示。

由於組譯器每次只組譯一個單獨的檔案之故，所以位址都是由 0 開始計算，當 MAIN.DT 被組譯完成後，它已經佔用到程式記憶體 0 開始的位址，等到第二次組譯 CALC.DT 時，組譯器並不知道程式記憶體 0 開始的位址已經被佔用了，又會將 CALC.DT 的程式從 0 開始放置，造成組譯時發生位址衝突的現象，所以有這種情況發生時會得到一個錯誤訊息：

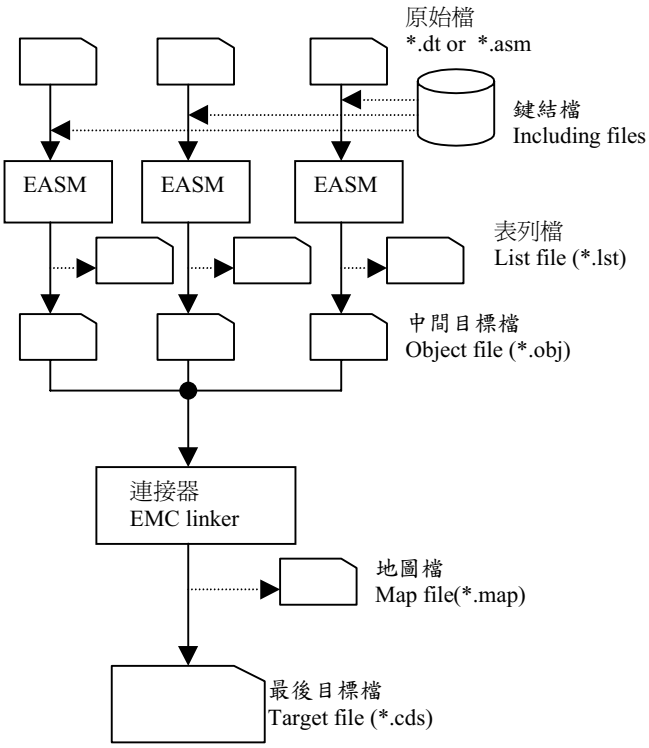
```
error L006:ROM address 0 is conflict.
```

所以我們應該爲每個獨立的程式分配一個唯一的空間，這個空間僅分配給這個模組使用，別的程式不得佔用，程式中可以在開始的地方用 ORG 指令爲每一個獨立模組指定一個安全的位址空間。

程式一是一個大型程式的切割範例，主程式呼叫了位於不同檔案中的副程式，範例中的副程式沒有做任何事情，執行一個 NOP 指令後就返回了上一層，在 VAR.H 中我們也示範了其他不同假指令的用法實例供讀者參考。



圖一 大程式分割架構


原始檔
*.dt or *.asm
鍵結檔
Including files
EASM
EASM
EASM
表列檔
List file (*.lst)
中間目標檔
Object file (*.obj)
連接器
EMC linker
地圖檔
Map file(*.map)
最後目標檔
Target file (*.cds)

圖二 組譯流程圖

程式範例

VAR.H

```
; ############################
; #           VAR.H          #
; ############################

; ## Page address ##
  PAGE_0   ==  0000H
  PAGE_1   ==  0400H
  PAGE_2   ==  0800H
  PAGE_3   ==  0C00H
  PAGE_4   ==  1000H
  PAGE_5   ==  1400H
  PAGE_6   ==  1800H
  PAGE_7   ==  1C00H
  PAGE_8   ==  2000H
  PAGE_9   ==  2400H
  PAGE_10  ==  2800H
  PAGE_11  ==  2C00H
  PAGE_12  ==  3000H
  PAGE_13  ==  3400H
  PAGE_14  ==  3800H
  PAGE_15  ==  3C00H

;## Conditional Assembly ##
IFNDEF H_MAIN
  EXTERN  SUB_M1, SUB_M2
ENDIF

IFNDEF H_LCD
  EXTERN  SUB_L1, SUB_L2
ENDIF

IFNDEF H_CALC
  EXTERN  SUB_C1
  EXTERN  SUB_C2
ENDIF

IFNDEF H_NOTE
  EXTERN  SUB_N1
  EXTERN  SUB_N2
ENDIF

; ## Macros ##
BANK  MACRO   NUM
IF NUM==0
   BC   0X04,6
   BC   0X04,7
ELSEIF NUM==1
   BS   0X04,6
   BC   0X04,7
ELSEIF NUM==2
   BC   0X04,6
   BS   0X04,7
```

```
ELSEIF NUM==3
   BS   0X04,6
   BS   0X04,7
ELSE
   MESSAGE "ERROR: BANK OVERFLOW"
ENDIF
ENDM

;## Constant ##
DEF_YEAR    VAR    101        ; 0~199:(1900~2099)
DEF_MONTH   VAR     02
DEF_DAY     VAR     12

;## Special Register [0X00~0X0F] ##
R0      ==    0X00
TCC     ==    0X01
STATUS  ==    0X03
; {
    C   EQU       0
    DC  EQU       1
    Z   EQU       2
; }

R4      ==    0X04
R5_ROM  ==    0X05

;## User data area [0X10~0X1F] ##
ISK1    ==    0X10
ISK2    ==    ISK1+1
ISK3    ==    ISK2+1
XX      ==    0X13
YY      ==    0X14
ZZ      ==    0X15

; ## Bank 0~3 ##
YEAR    ==    0X20
MONTH   ==    0X21
DAY     ==    0X22
```

MAIN.DT

```
; ###################################
; # Filename: MAIN.DT               #
; # Date: 2/12/2001                 #
; # File version: 1.0               #
; # Author: Jenwen-Tan              #
; # Company: ELAN corp.             #
; # Note:                           #
; ###################################

    H_MAIN      EQU     PAGE_0
    INCLUDE     "C:\S1\VAR.H"

; ## Public Procedure ##
PUBLIC      SUB_M1
PUBLIC      SUB_M2

; ## Program begin ##
    ORG     PAGE_0
    JMP     INIT

; ## Interrupt vector ##
    ORG     8
INTERRUPT   PROC
;{
    MOV     ISK1,A                  ; Save A
    SWAP    ISK1
    SWAPA   STATUS                  ; Save R3 PSW
    MOV     ISK2,A
    SWAPA   R5_ROM
    MOV     ISK3,A                  ; Save R5 ROM page
 ;  PAGE    0

ENDINT:
    SWAPA   ISK3
    MOV     R5_ROM,A                ; Restore R5 ROM page
    SWAPA   ISK2
    MOV     STATUS,A                ; Restore R3 status
    SWAPA   ISK1                    ; Restore ACC
    RETI
;}
    ENDP

; ## Main program ##
; main program code goes here
INIT:

HARDWARE:
    ; { initial register }
SOFTWARE:
    ; { initial variable }
    MOV     A,@DEF_YEAR
    MOV     YEAR,A
    MOV     A,@DEF_MONTH
    MOV     MONTH,A
    MOV     A,@DEF_DAY
```

```
    MOV     DAY,A

START:
    ; { main loop }
    CALL    SUB_M1
    CALL    SUB_M2
    PAGE    3
    CALL    SUB_L1
    CALL    SUB_L2
    PAGE    1
    CALL    SUB_C1
    CALL    SUB_C2
    PAGE    2
    CALL    SUB_N1
    CALL    SUB_N2
    PAGE    0
    JMP     START

; ## Sub-routine ##
SUB_M1      PROC
    NOP
    RET
    ENDP

SUB_M2      PROC
    NOP
    RET
    ENDP
```

CALC.DT

```
;############################
;#        CALC.DT           #
;############################

    H_CALC      EQU     200
    INCLUDE     "C:\S1\VAR.H"
    ORG         PAGE_1

PUBLIC  SUB_C1
PUBLIC  SUB_C2

SUB_C1  PROC
    NOP
    RET
    ENDP

SUB_C2  PROC
    NOP
    RET
    ENDP
```

NOTE.DT

```
;############################
;#        NOTE.DT           #
;############################

    H_NOTE      EQU     300
    INCLUDE     "C:\S1\VAR.H"
    ORG         PAGE_2

PUBLIC  SUB_N1
PUBLIC  SUB_N2

SUB_N1  PROC
    NOP
    RET
    ENDP

SUB_N2  PROC
    NOP
    RET
    ENDP
```

LCD.DT

```
;#############################
;#        LCD.DT             #
;#############################

    H_LCD       EQU     400
    INCLUDE     "C:\S1\VAR.H"
    ORG         PAGE_3

PUBLIC  SUB_L1
PUBLIC  SUB_L2

SUB_L1  PROC
    NOP
    RET
    ENDP

SUB_L2  PROC
    NOP
    RET
    ENDP
```

程式 一