

組合語言編輯程式活用系列(二)

APPLE II 組合語言編輯程式

TOOL KIT

活用法

徐寶龍 編著

美國暢銷書「THIRD WAVE」作者托佛勒教授以「THIRD WAVE」稱呼資訊時代，並以「波前(Wave front)」代表鼓動資訊風潮一些前進的事物。

無疑的，在台灣電腦與管理專業書籍的出版是「THIRD WAVE」新文明的「波前」之一，她將負起引導台灣社會邁向資訊時代的重大責任。

然而坊間電腦書籍，內容不盡理想，浪費讀者寶貴的時間與金錢。本公司有鑑於此，乃邀請專家學者有系統的撰寫一系列電腦專業書籍。以最高品質，服務讀者。同時達到「波前出版，必屬好書」的口碑。

波前電腦管理圖書有限公司



波前電腦管理圖書有限公司

序

承蒙各位親愛讀者的愛護與波前圖書公司的支持，本書得以再次出版。在本書初版後八個月內，不斷地收到各階層讀者的指正，於此衷心感謝大家的關心與照顧。

使用本書的各項功能都必須配合 TOOL KIT 磁片，這個磁片與一般的 DOS 磁片相同，而其上存有 EDASM 等可以執行各項功能的程式。請注意不要刪除下列各個檔案：EDASM, EDASM.OBJ；ASSM, EDITOR, ASMIDSTAMP, RLOAD, RBOOT。其中 ASMIDSTAMP 不可被鎖定。有了 RLOAD 及 RBOOT 兩個程式即可使用重新定址 (Relocation) 的檔案。

在本書再次出版之際，深感家庭與社會給予我很大的支持與鼓勵，所以要把本書再次獻給社會大眾與養育恩重的父母。

徐寶龍

1984 仲夏於台北

目錄

第 1 章	概論	1
1 - 1	系統說明	1
1 - 2	編輯程式	2
1 - 3	組合語言譯碼程式	3
1 - 4	使用本系統前的準備工作	3
1 - 5	系統的啟動	4
第 2 章	編輯程式	8
2 - 1	前言	8
2 - 2	如何使用編輯程式	9
2 - 3	命令模式的功能	10
2 - 4	參數的語法	15
2 - 5	編輯命令	17
2 - 6	處理磁片和磁帶的命令	32
2 - 7	操作命令 (Operating Command)	37
第 3 章	組合語言譯碼程式 (Assembler)	46
3 - 1	前言	46
3 - 2	如何使用組合語言譯碼程式	49
3 - 3	控制譯碼組合程序命令	51
3 - 4	原始程式的格式	53
3 - 5	組合語言譯碼程式的假指令	58
3 - 6	控制列表的假指令	63
3 - 7	定義資料的假指令	66
3 - 8	有條件性譯碼組合之假指令	68
3 - 9	符號表的顯示	70
附錄 A	編輯程式／組合語言譯碼程式使用的記憶體及用途	74
附錄 B	編輯程式顯示的 DOS 錯誤	77
附錄 C	再定位載入程式 (RELOCATING LOADER)	80
附錄 D	組合語言譯碼程式的 DOS 錯誤碼 (DOS ERROR CODES) “OOPS ”	83
附錄 E	目的檔的格式	85
附錄 F	標記符號表 (Symbol table) 的格式	88
附錄 G	BASIC 程式的編輯	90

第 1 章

概論

1-1系統說明

Apple II 組合語言譯碼程式／編校程式 系統 (Assembler / Editor System) 提供編輯 (Editing) 和將 6502 組合語言程式予以譯碼組合 (Assembling) 的功能。此一系統由三個程式所構成：編輯程式 (Editor)、組合語言譯碼程式 (Assembler) 和命令釋義程式 (Command Interpreter)。使用者可經由命令釋義程式來呼叫編輯程式或組合語言譯碼程式。系統啟動後，命令釋義程式一直在記憶體中。螢幕上任一行的開頭有出現提示字元 (Prompt Character) ' : ' 則表示編輯程式也在記憶體中。當使用者透過 ASM 命令，要求系統將使用者的 6502 組合語言程式予以譯碼組合時，系統就會將組合語言譯碼程式從磁碟機讀入，取代在記憶體中的編輯程式，而在編輯緩衝區中經過編輯的檔案也將被清洗掉。一旦譯碼組合的程序完成後

，系統將再度把編輯程式從磁碟機中讀入，取代在記憶中的組合語言譯碼程式。

此系統的使用，需要具有 32K 記憶體之 Apple II 或 Apple II plus 和至少一部磁碟機。如果要經過譯碼組合的組合語言程式相當大時，則需 48K 的記憶體和另外一部來讀取原始檔磁片的磁碟機。

組合語言譯碼程式／編輯程式 系統存於 Applesoft Tool Kit 磁片，此磁片包含 16-Sector DOS，此系統 DOS 檔的結構跟 Applesoft 和 Integer BASIC 具有共通性 (Compatibility)，但無法與 Apple PASCAL 系統共通，Apple PASCAL 有自己的編輯程式和組合語言譯碼程式。

系統可分為三種模式 (Mode)：

- 1 命令模式 (Command Mode)
- 2 輸入模式 (Input Mode)
- 3 編輯模式 (Editing Mode)

當系統在命令模式時，則表示命令釋義程式準備接受使用者的命令來執行相關的動作。在輸入模式時，則表示使用者可以經由編輯程式在編輯緩衝區裏建立原始程式。在編輯模式時，使用者可經由編輯程式來編輯或修改在編輯緩衝區裏的原始程式。

1-2 編輯程式

編輯程式是讓使用者用來建立能符合組合語言譯碼程式所要求的格式之原始程式 (Source Program)，或使用編輯命令來編輯修改原始程式。此一編輯程式亦可用來建立 EXEC 檔和用來

編輯或建立 BASIC 程式。有一點須注意的是，它並不是為了提供文字編輯功能而設計的。它能夠處理磁帶的檔案，且經由 DOS 亦可處理磁碟的檔案。

1-3 組合語言譯碼程式

組合語言譯碼程式能提供譯碼組合 (Assembly) 的功能，也就是說，它能夠將使用者所指定的文字檔 (text file) 中之原始程式轉換為 6502 機器碼 (Machine Code)。它從磁碟中讀取使用者所指定的原始程式，然後將譯碼組合後的結果寫入磁碟中。它僅保留 Symbol table 在記憶體裏面，如果須要話，RLD (Relocation dictionary) 也可保留。因此，此一組合語言譯碼程式不會因記憶空間的關係，而使得譯碼組合的規模受到太大的限制。所以它能夠處理相當大的 6502 組合語言程式。

這個組合語言譯碼程式執行著 Two-Pass 的譯碼組合程序。它亦提供有條件譯碼組合 (Conditional Assembly) 的功能給使用者。組合語言譯碼程式平常並不在記憶體裏，只有在使用者使用 ASM 的命令來要求系統將使用者的原始程式轉換為 6502 機器碼時，系統才會將它從磁碟機中讀入，一旦工作完畢，它將不存在記憶體裏。

1-4 使用本系統前的準備工作

在未熟悉本系統之前，而想嘗試使用編輯程式或組合語言譯碼程式時，最好先將 Applesoft Tool Kit 磁片複製一份，以

防止因操作不當而將 Applesoft Tool Kit 磁片破壞，致使工作無法完成。如果你只有一部磁碟機，那你可利用 Apple Utility 磁片中的檔案複製程式 FID (File Duplicator) 來做拷貝的工作。

這本使用手冊是假設此系統的使用者熟悉 6502 組合語言和 Apple DOS 而寫成的，如果你對於 Apple DOS 不熟悉的話，最好先閱 DOS 手冊，然後再使用本系統。

1-5 系統的啓動

將 Applesoft Tool Kit 磁片放入磁碟中，然後啓動 (boot) 磁碟機，啓動後，螢幕會出現：



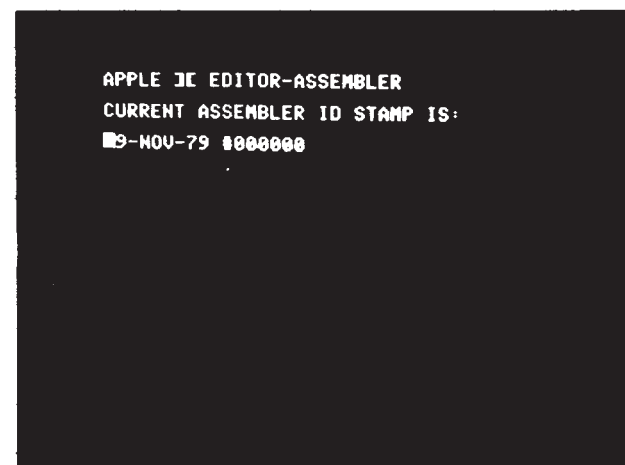
然後從鍵盤按入

RUN EDASM

然後按 RETURN 鍵。如果你的 Apple 只有 Integer BASIC 的話，則須按入

RUN INTEDASM

然後按 RETURN 鍵。螢幕就會顯示出組合語言譯碼程式的 ID Stamp



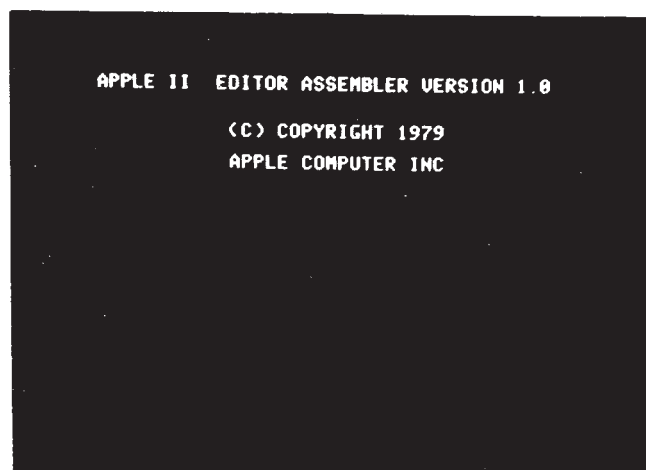
此 ID Stamp 可依使用者的意思予以更改。直接按 RETURN 鍵，則原先的 ID Stamp 沒有更改。如果使用者希望更改 ID Stamp，則可利用 ← 和 → 鍵來移動游標至欲更改字元的位置，然後按入新的字元以代替原先的字元，一旦將欲更改的字元全部更改完畢後，按 RETURN 鍵，系統會將新的 ID Stamp 存入磁片，代替舊的 ID Stamp，此時螢幕顯示的 ID Stamp 為新的 ID Stamp。(此一 ID Stamp 包含空格，一共可容納 17 個字元，如果更改後的 ID Stamp 超過 17 個字元，則系統僅將前面的 17 個字元存入磁片中，以後顯示出的 ID Stamp 也只有這 17

個字元)例如,使用者希望新的 ID Stamp 為

29-NOV-82 #0 0 0 0 0 0

利用→鍵移動游標至原來的 ID Stamp 中字元 7 的位置,然後按入 82,再按 RETURN 鍵,則新的 ID Stamp 將存入磁片,而且螢幕顯示出新的 ID Stamp。

系統將新的 ID Stamp 存入磁片後,螢幕會顯示出



螢幕底的左邊有一個字元“:”,此一字元稱為提示字元(Prompt Character),閃動的底綫(underline)為游標。系統顯示在螢幕上任一行的第一個字元為提示字元“:”,則表示命令釋義器準備接受命令。當使用者按入命令和 RETURN 後,命令釋義程式將執行相關動作。

注意: Applesoft Tool Kit 磁片須放在 Driver 1,如此,命令釋義程式才有辦法讀到組合語言譯碼程式或編輯程式。

如果 DOS 已經在記憶體中,而且你準備使用組合語言譯碼

程式或編輯程式時,只要將 Applesoft Tool Kit 磁片放入 Driver 1,然後按 RUN EDASM 或 RUN INTEDASM 以及 RETURN 鍵,這樣即可進入本系統,而不需要重新啟動(reboot)磁片中的 DOS。如果不想改變 ID Stamp,則按入

BRUN EDASM.OBJ

然後按 RETURN 即可。

第 2 章

編輯程式

2-1前言

系統在螢幕上任何一行 (line) 的開頭顯示出提示字元 “ : ”，則表示命令釋義程式準備接受命令，也就是說系統處於命令模式。而這時，編輯程式亦在記憶體中，隨時可以進行編輯工作。如果按入的命令前面、後面或命令字元串中間有空格存在，命令釋義程式仍能接受，因為命令釋義程式會將這些空格予以忽略。這一章所討論的各種命令皆以字元串表示，字元串中包含有大寫及小寫的字元。字元串是用來表達命令的意義。但命令的輸入並不須要將這整個字元串按入，而只須按入大寫字元的部分，小寫字元可以省略。例如，COpy 此命令是要求系統將某一段程式複製一份，而這命令的輸入只須從鍵盤按入 CO，而 py 可以省略不按。

編輯程式並沒有將行號 (line number) 存於編輯檔中。

它定義在檔中兩個 RETURN 間的內容為一行 (line)，而且在每一行的前面加上行號，然後將其顯示於螢光幕上，加上行號主要目的是使編輯工作方便進行。當被編輯檔中有某些的行被刪除，則接在被刪除行後面的行之行號將自動遞減，如果有某些行插入被編輯檔中，則接在插入行後的行之行號會自動地增加。

編輯程式經由 Tab 命令，提供了 16 個可調整的表格定位設定 (TAB Setting)，以使得 text 容易閱讀。編輯程式的輸出裝置為 40 Column 的螢幕顯示，而命令的輸入是從鍵盤輸入，亦可使用外加的終端裝置，但輸出程式會將每 40 字列為一行。欲編輯的檔 (Editing File) 存於編輯緩衝區，此一緩衝區的大小跟所使用的 Apple 有所不同。擁有 48K RAM 的 Apple，其緩衝區的大小為 29K, 32K RAM 的 Apple，其緩衝區的大小為 13K。以 48K 的 Apple 而言，其緩衝區大概可容納 700 行至 1400 行的原始程式。典型的 6502 組合語言，包括註解，一行大約有 40 個字，而在一個程式裏，有些指令很短，而有些沒有註解。

本系統的編輯程式並不是為了用來做文字處理 (Word Processing) 而設計的。其設計的主要目的是用來發展程式。它亦可用來編輯 Applesoft 或 Integer BASIC 程式。讀者可參考附錄 G。

2-2如何使用編輯程式

前面已經提過，編輯程式有兩種功能，一為讓使用者在緩衝區建立原始程式，另一為讓使用者來編輯修正在編輯緩衝區中的欲編輯檔或原始程式。

使用者可以利用 2-5 節中的 Add 命令在編輯緩衝區建立新的原始程式。Add 命令的執行，使系統處於輸入模式此時使用者可建立新的原始程式。一旦原始程式建立完畢後，按入 CTRL-D 或 CTRL-Q，可以使系統回至命令模式。這時候，使用者可以使用 2-6 節中的 SAVE 或 TSAVE 命令，將在編輯緩衝區中的原始程式存入磁片或磁帶。如果新建立的原始程式有需要編輯修正時，使用者可以利用 2-5 節中的各種編輯命令來做編輯的工作。

如果使用者所要編輯修正的原始程式並不在編輯緩衝區內，而是存在磁片或磁帶中，在這種情況時，使用者可以利用 2-6 節中的 LOAD 或 TLOAD 命令，將存於磁片或磁帶中的原始程式讀入編輯緩衝區內，然後使用 2-5 節的各種編輯命令來做編輯修正的工作。

2-3 命令模式的功能

使用者可以使用 Help 命令來要求系統將各種命令及各個命令的語法 (Syntax) 顯示在螢幕上。從鍵盤按入

?

然後按 RETURN，螢幕上會顯示出各種命令及其語法：

```

?
ASM SOURCE FILE,<OBJECT FILE>,<L>,<L>
APPEND <LINE#> SOURCE FILE
A <LINE#>
CAT <LINE#>
CD <LINE#> - <LINE#> TO <LINE#>
CP <LINE#> <BEGIN#> <-END#> OLDSTR.NEWSTR.
DE <LINE#> <BEGIN#> <-END#>
END <BEGIN#> <-END#>
FILE <BEGIN#> <-END#> .STRING.
HT <BEGIN#> <-END#> .STRING.
LE <LINE#>
LOAD SOURCE FILE
LO <LINE#>

```

全部的命令需要二次螢幕顯示，再按任何鍵，就可將其餘的命令顯示在螢幕上：

```

E <BEGIN#> <-END#> .STRING.
FILE <BEGIN#> <-END#> .STRING.
HT <BEGIN#> <-END#> .STRING.
LE <LINE#>
LOAD SOURCE FILE
LO <LINE#>
MON <BEGIN#> <-END#>
NEW <BEGIN#> <-END#>
PR <BEGIN#> <-END#>
RE <BEGIN#> <-END#>
S <BEGIN#> <-END#>
SAVE <BEGIN#> <-END#> <SOURCE FILE>
TLOAD <BEGIN#> <-END#>
TSAVE <BEGIN#> <-END#>
TIME <BEGIN#> <-END#> .STRING.
LE <LINE#>

```

顯示在螢幕上的命令字元串中，用黑底白字顯示的字元為必需字

元，亦即，要求系統執行此命令，須要將必須字元按入。以黑字白底顯示的字元為可以省略的字元。例如，DELETE 命令，輸入此命令只須按入必須字元 D，DELETE 可以省略。螢幕僅顯示各種命令後面的參數之一般型式。有關各命令參數之更詳細語法，讀者可查閱下面各節中各個命令的說明。

命令模式有許多功能使得系統的使用更為方便，這些功能敘述如下：

(1) 允許數個命令同時輸入

這一功能允許使用者在一行中按入數個命令。命令釋義程式會將這些命令依照順序儲存起來，每當一個命令執行完畢後，命令釋義程式會取出下一個命令來執行。在按入命令時，每個命令之間需用命令標點 (Command Delimeter) 來分開。此一標點通常設定為冒號 “:”，使用者可以改變命令標點為另外一種符號。任何命令語法錯誤將會使命令釋義程式清除儲存著的其他命令。然而，Find, Change, Edit 命令如發生找不到字元串的情形，系統並不將這種情況視為命令的錯誤。在儲存著的命令中，ASM 將是最後被執行的命令，因為，一旦組合語言譯碼程式被讀入時，任何跟在 ASM 後面的命令將會被清除。

Add, Insert, COpY, Delete 和 Replace 命令會使得被編輯程式某些行的行號受到影響而改變。因此，使用者須注意，當會改變行號的命令執行後，而儲存著尚未執行的命令後面之行號參數是否仍為原來使用者欲處理那行之行號，例如：

```
1  LDA  # $ 34
2  STA  POA
3  STA  POB
```

```
4  LDA  POC
5  STA  BUFFER
6  STA  REG
```

我們希望將第 3 行刪除，然後將第 1 行 COpY 至第 6 行前面，數個命令同時輸入為

```
: D 3 : CO 1 TO 5
```

因為刪除第 3 行的命令先執行，這時原先的第 6 行變為第 5 行，因此 COpY 的參數行號須為 5 而不是 6。如果先執行 COpY 的命令後，再執行刪除的命令，則此問題便可以解決。

這時，命令輸入應為

```
: CO 1 TO 6 : D 3
```

(2) 命令標點的設定 (Command Delimeter Set)

這一功能允許使用者改變命令標點的符號。在使用者沒有改變命令標點之符號時，機定 (default) 的命令標點符號為 “:”，因為冒號一般在 6502 組合語言程式並沒有使用到。Find, Change 和 Edit 命令所尋找的字元串不可含有與命令標點一樣的符號字元。否則，命令釋義程式會將此符號字元視為命令標點，因此執行的結果會有錯誤產生。所以命令標點設定的功能，使這種錯誤得以避免。

更改命令標點的方法為：按 ESC 鍵，然後按冒號 “:”，再按新的命令標點字元，新的標點就會顯示在螢幕上。如果按入新的命令標點字元為空格，則喇叭會嗶一聲，系統希望你重新按入另一個命令標點字元。如果按入新的命令標點字元為 RETURN 鍵時，則系統會將目前使用的命令標點字元顯示在螢幕上，然後回到命令模式。任何字元皆可設定為命令標點字元。

(3)再執行上個List 命令的動作一次

這一功能允許使用者只按一個字元，就能再執行一次上個List 命令，而不須將上一個List 命令及參數重覆按入。按CT RT-R，然後按RETURN，即可再執行上個List 命令。在多個命令輸入時，在命令標點之後的CTRL-R，也會被當成一個命令被儲存起來。

(4)直接使用DOS 命令

這一功能允許使用者在命令模式時，仍然能夠直接使用DOS 命令。編輯程式也因而具有了DOS 的檔案管理功能。能夠直接使用的DOS 命令有：RENAME，LOCK，UNLOCK，MON，NOMON。在DOS 的命令前面加一句點“.”，就可以在命令模式時直接使用DOS 的命令。例如，

:. RENAME XX, YY

注意：直接使用DOS 命令的功能，提供給熟練的使用者相當大的便利。如果對於DOS 的命令不太瞭解，應避免使用其他的DOS 命令，因為用錯其他的DOS 命令所得到的結果，通常是無法預期的，而且很可能破壞了本系統。

(5)能夠將各種命令及其語法顯示在螢幕上

這一功能在本節的開頭即有提到。這一功能給予使用者相當大的便利。因為使用者隨時可以在螢幕上參考各種命令及其語法。按入?後再按入命令的字元串時，則系統會將該命令及其語法顯示在螢幕上。如果按入?後再按入的命令字元串為不存在的命令，則系統認為你不知道有那些命令可以使用，因此系統就將此命令視同只有按?，然後將各種命令及語法顯示在螢幕上。

2-4參數的語法

在這裏定義一些名詞，而這些名詞是作為各命令的參數名稱。有些命令不需要使用參數，有些命令必須使用參數。有的命令後面的參數可以省略。有些命令後的參數有許多個，而這些參數有些需要，有些可以依情況取捨，這些參數須按照順序輸入。

參數如果為字元串，而使用參數時，須用兩個相同標點(Delimiter)將字元串標示出來，例如，'ME'。ME為參數所使用的字元串，單引號為標點。使用的標點不可為逗點，連字號()，數元(0~9)，空格或RETURN。建議使用的標點為，單引號，雙引號，句點，問號，驚嘆號和分號。如果使用括弧當作標點，則用來標示字元串的兩個標點須同為左括弧或右括弧。例如，<ME<或>ME>或(ME(或)ME)皆為符合語法。

參數名稱	說明
Rnumber	任一命令後面有此參數時，必須指定十進位的正整數給這參數，這數值的範圍依命令的而定。
Onumber	任一命令後有此參數，可依情況予以取捨(optional)。此參數所使用的數值須為十進位正整數，數值的範圍依命令而定。
Rlinenum	任一命令後有此參數，必須指定一個十進位的正整數給此參數。這個參數代表行號(line number)，其範圍從1至65519。
Olinenum	這個參數代表行號，亦為可依情況取捨的參數

Range	。其值為十進位正整數，範圍從 1 至 65519。 這參數由一個或二個 Olinenum 所構成，因此，這個參數也是可依情況予以取捨。如果是由二個 Olinenum 構成，兩個 Olinenum 之間需用連字號連接，亦即 Olinenum-Olinenum。此時這參數所代表的範圍從第一個 Olinenum 至第二個 Olinenum。如果第二個 Olinenum 小於第一個的值，並且小於 100，則第二個 Olinenum 的值表示此一範圍的行數，包括第一個 Olinenum 這一行。這表示 100-99 和 100-199 是一樣的意思。如果 Range 這參數僅由一個 Olinenum 構成，則此參數所代表的範圍僅為 Olinenum 這一行。如果這參數的型式為 Olinenum —，則此參數所代表的範圍從 Olinenum 這一行至最後一行。
Rangelist	這是一組 Range 參數，各個 Range 之間使用逗點分開。這一參數並沒有限制 Range 的排列須按照大小。
Dstring	此一字元參數是由兩個相同的標點和字元串構成。字元串在兩個相同標點之間。例如，. M E . 或 “MY”。
Chgstring	此一參數的型式為：標點後面跟著字元串，此字元串後面再跟著同樣的標點，這個標點後面再跟著字元串和同樣的標點。例如，. ME . MY . 或 “ME” MY”。

Rfilename 此參數表示檔名，由 1 至 30 個字元構成，但不可包含逗點。任一命令後面有此參數時，則在使用此命令時須指定檔名給予此參數。

Ofilename 此參數結構跟 Rfilename 一樣，但是 Ofilename 可依情況取捨。

Oobjfilename 此參數跟 Ofilename 一樣，但此參數用來表示由組合語言譯碼程式所產生的 Binary 或 Relocatable object file 的檔名。

Dev Ctlstring 這是用來啟動做為組合語言譯碼程式的輸出裝置之週邊卡。例如：CTL - I 120 N 或 CTL - AF。

2-5 編輯命令

使用者可以使用本節所討論的各種命令來編輯修正在編輯緩衝區中的原始程式。這些命令是用含有大寫及小寫字元的字元串來表示，命令的輸入只須將大寫字元按入，而小寫字元可以省略。在命令字元串後面的參數，如果在方括弧 [] 裏面，則表示此參數可依情況而予以取捨。如果沒有方括弧，則表示在命令輸入時，此參數也須要輸入。

ADD

Add [Olinenum]

Add 命令的用途有兩種：

(1) 在編輯緩衝區內的欲編輯檔中加入一行或一段程式。如果 Olinenum 沒有省略，新加入的部分便加至行號為 Olinenum 的

那一行下面。例如，我們希望在程式 2-1 中的第 2 行下面新加入

1 LDY \$ 20	1 LDY \$ 20
2 LDX #\$ 00	2 LDX #\$ 00
3 CLC	3 STX \$ 20
4 LDA \$ 21 , X	4 CLC
5 ADC \$ 51 , X	5 LDA \$ 21 , X
6 STA \$ 21 , X	6 ADC \$ 51 , X
	7 STA \$ 21 , X

程式 2-1

程式 2-2

一行或一段程式。從鍵盤按入 A2 RETURN 後，則編輯程式會在螢幕上顯示出 3 和游標，這表示，編輯程式準備接受新加入一行的輸入。當使用者新加入的一行輸入完畢，按 RETURN 後，編輯程式會在螢幕上顯示下一行的行號 4 及游標，表示它繼續接受新的輸入。因此使用者可以依上面的程序加入一行或數行的程式。系統執行 Add 命令時，系統是處於輸入模式 (Input Mode)，使用者新加入的資料全部輸入完畢後，在編輯程式將行號及游標顯示在螢幕上時，按 CTRL-D 或 CTRLQ 和 RETURN，可使系統結束輸入模式而回到命令模式。在程式 2-1 的第 2 行加入一行指令，其結果如程式 2-2 所示。原先在第 2 行以下各行之行號均已改變。如果 Add 命令後面的 Olinenum 參數省略，只按 A 及 RETURN，編輯程式會計算在編輯緩衝區中程式最後一行的行號，然後將下一行的行號及游標顯示在螢幕上，準備接受新加入的資料。因此新加入的資料是加在編輯緩衝區中程式的最後面。例如，我們希望在程式 2-3 最後面加一段程式。按 A 及 RETURN 後，螢幕出現行號 7 及游標，然後按下新加入的資料

，全部資料輸入完畢後，在螢幕上出現行號及游標時，按 CTRL-D 或 CTRLQ 及 RETURN。在程式 2-3 最後面加一段程式，其結果如程式 2-4 所示。

1 DELUEL LDY #\$ 00
2 LDA (\$ 30) , Y
3 TAX
4 LDA \$ 2 F
5 NEXTEL INY
6 CMP (\$ 30) , Y

程式 2-3

1 DELUEL LDY #\$ 00
2 LDA (\$ 30) , Y
3 TAX
4 LDA \$ 2 F
5 NEXTEL INY
6 CMP (\$ 30) , Y
7 BEQ END
8 DEX
9 BNE NEXTEL
10 END RTS

程式 2-4

(2) 在編輯緩衝區中建立新的程式或文字檔 (text file)。如果編輯緩衝區沒有任何程式或檔案存在，按 A 和 RETURN 後，編輯程式在螢幕上顯示出第 1 行之行號及游標，這時系統處於輸入模式，使用者可以從鍵盤輸入程式。程式輸入完畢後，按 C

TRL-D 或 CTRL-Q 和 RETURN 使系統回至命令模式，系統在螢幕上顯示出提示字元“：”，準備接受命令。

當系統在輸入模式時，一般的 Apple 輸入編輯功能也可使用。如果你的 Apple 有 Autostart ROM，系統在輸入模式時，使用者亦可使用 ESC-I, J, K, M 來移動游標，做輸入編輯的工作。系統在輸入模式時，允許使用者輸入控制字元 (Control Character) 到檔案中。如果使用者在建立組合語言程式時，應注意不要將控制字元混入程式中，因為控制字元在組合語言是不合語法的。

COPY

COPY Rlinenum 1 [- Olinenum 2] TO Rlinenum 3
這個命令是用來將欲編輯檔 (在編輯緩衝區中的程式或 text file) 中某一段複製一份，然後放在行號為 Rlinenum 3 這一行之前。如果 Olinenum 2 省略，參數型式為：Rlinenum 1 TO Rlinenum 3，則被複製的這一段僅包括 Rlinenum 1 這一行。如果 COPY 命令有 Olinenum 2，其參數列的型式為：Rlinenum 1 - Olinenum 2 TO Rlinenum 3，則被複製的一段為 Rlinenum 1 至 Olinenum 2。在各種編輯命令裏，只有這一命令須要在參數列裏加上 TO 的字元。COPY 的命令輸入只須按 CO 即可，py 可以省略。

在編輯命令裏並沒有 MOVE 這個命令，亦即，編輯命令裏沒有能夠直接將被編輯檔中的某一段從某一位置移至另一位置的命令。使用 COPY 和 Delete 命令就可達到 Move 的功能。首先使用 COPY 將欲移至新位置的一段複製至新位置，然後用 Delete 命令將在原來位置的該段刪除。COPY 命令執行後，在原來位置

的該段中各行之行號可能改變，因此，使用 Delete 時，應注意在原來位置的該段各行之行號是否已經改變。圖 2-1 和圖 2-2 為

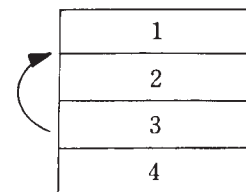


圖 2-1

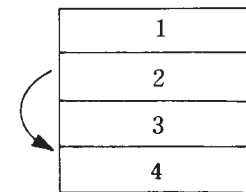


圖 2-2

Move 的兩種情況。以圖 2-1 而言，經過 COPY 的命令執行後，在原來位置的第 3 段中各行號均已增加，改變後的行號為原先的行號加上第 3 段所包括的行數。以圖 2-2 而言，經過 COPY 命令執行後，在原來位置的第 2 段中各行之行號沒有改變。或許讀者還記得，在命令模式時，系統允許數個命令同時輸入的功能。因此，同時將 COPY 和 Delete 命令輸入，就相當於 MOVE 的命令。例如，我們欲將程式 2-5 的第 7 行至第 8 行這一段移至第 4 行

```

1 MPADD LDY $ 20
2          LDX # 00
3          CLC
4 NXTBY  LDA $ 21, X
5          ADC $ 51, X
6          STA $ 21, X
7          INX
8          DEY
9          BNE NXTBY
10         RTS

```

程式 2-5

```

1 MPADD LDY $ 20
2      LDX # 00
3      CLC
4      INX
5      DEY
6 NXTBY LDA $ 21 , X
7      ADC $ 51 , X
8      STA $ 21 , X
9      BNE NXTBY
10     RTS

```

程式 2-6

上面（這為圖 2-1 之情況），按入

: CO 7 - 8 TO 4 : D 9 - 10

然後按 RETURN，程式 2-6 為 Move 後的結果。注意 Delete 命令後面的行號為原先的行號加上第 7 行至第 8 行這一段的行數。

如果我們欲將程式 2-7 第 1 行至第 6 行 Move 至第 10 行上面（這為圖 2-2 之情況），按入

: CO 1 - 6 TO 10 : D 1 - 6

然後按 RETURN，程式 2-8 為 Move 後的結果。

```

1 NXTBY LDA $ 21 , X
2      SBC $ 51 , X
3      STA $ 21 , X
4      INX
5      DEY

```

```

6      BNE NXTBY
7 MP SUB LDY $ 20
8      LDX # 00
9      SEC
10     RTS

```

程式 2-7

```

1 MP SUB LDY $ 20
2      LDX # 00
3      SEC
4 NXTBY LDA $ 21 , X
5      SBC $ 51 , X
6      STA $ 21 , X
7      INX
8      DEY
9      BNE NXTBY
10     RTS

```

程式 2-8

CHANGE

Change [Rangelist] Chgstring

Change 命令是用來將被編輯檔中的某一字元串用另一字元串來取代。Change 命令後面如有 Rangelist，編輯程式僅在 Rangelist 所指定的範圍尋找欲被代替的字元串，一旦找到後，就用新的字元串予以取代，直到這範圍內所有的欲被代替字元串均被取代。如果 Rangelist 省略，編輯程式就會尋找整個檔，一旦找到欲被代替字元，就用新的字元予以取代，直到整個檔中所有的

欲被代替字元均被取代。

Chgstring 的語法型式為

`< delim > String < delim > [ String ] [ < delim > ]`

其中 `< delim >` 表示標點 (`delimiter`)，`String` 表示字元串。第一個字元串為欲被代替的字元串，第 2 個字元串為用來取代的字元串。第一個及第二個標點和第一個字元串均不可省略。第 3 個標點可以省略。如果將第 2 個字元串省略，編輯程式就將在指定範圍或整個檔中的欲被代替字元串予以刪除。標點的使用字元，請參閱 2-4。

當按入 **Change** 命令後，螢幕會顯示

ALL OR SOME ? (A / S)

如果使用者按 **S** 鍵或 ASCII 值大於 **S** 的 ASCII 值之任何字元鍵，編輯程式會將含有欲被代替字元串的一行顯示在螢幕上，顯示在螢幕上的這一行，為已經用新的字元串取代原先的字元串，但是在被編輯檔中的並沒有更改。這時使用者如按空格鍵或除了 **ESC** 或控制字元的任何鍵，編輯程式就會用在螢幕已更改的一行取代被編輯檔中原先沒有更改的，然後顯示出下一個包含欲被代替字元串的一行。如果使用者按 **ESC** 或除了 **CTRL - C** 的控制字元，編輯程式僅在螢幕上顯示出下一個包含欲被代替字元串的一行 (螢幕上這一行原來的字元串已被取代，但在被編輯檔的仍然沒有改變)，而沒有用螢幕上已更改的一行來取代被編輯檔中原先沒有更改的。使用者可以重覆上述步驟，直到指定範圍或整個被編輯檔結束，然後系統回到命令模式。如果使用者在螢幕顯示 **ALL OR SOME (A / L)** 後，按 **A** 或 ASCII 值較 **S** 的 ASCII 值小之字元，編輯程式就將指定範圍或整個檔中全部的欲被代替

字元串用新的字元串來代替，並將更改後的行，顯示在螢幕上，然後回到命令模式。

編輯程式並沒有再檢查更改後的一行中是否仍含有欲被更改字元串。例如有一行，其中含有連續 10 個 *****，而你希望將 ****** 更改為 *****，當 **Change** 命令執行後，這一行只含有連續 5 個 *****，而編輯程式不會因這一行仍含有欲被更改字元串，******，而更度予以更改。

DELETE

Delete Rangelist

Delete 命令是用來將 **Rangelist** 所指定的範圍刪除。如果 **Rangelist** 包含數個 **Range**，則這些 **Range** 的排列必須為反順序。亦即，行號高的 **Range** 在前，行號低的 **Range** 在後。如果沒有這樣排列，當行號小的 **Range** 先被刪除後，在這一 **Range** 所指定範圍後面的各行之行號均變小了。因此，後面較高行號 **Range** 所指定的範圍已不是原先使用者欲刪除的範圍。**Delete** 命令後面的 **Rangelist** 不可省略，如果沒有 **Rangelist** 參數，則系統會在螢幕顯示

ERR : SYNTAX

有關 **Rangelist** 的語法請參閱 2-4 節。但有一點不同，**Delete** 命令的 **Rangelist** 中之 **Range** 的語法型式必須為 **Rlinenum 1 - Rlinenum 2**，如 10-20 或 100-150，而 **Rlinenum -**，例如 10 - 或 100 -，為不合語法的型式。

EDIT

`Edit [ Rangelist ] [ Dstring ]`

在所有的編輯命令中，**Edit** 命令為功能最多的一個。**Edit** 命令

使系統進入編輯模式 (Edit Mode)，它提供使用者完整的一次一個字元之編輯功能。編輯程式從指定範圍裏，一次取一行來編輯修正，使用者僅能用←鍵和→鍵來移動游標向左和向右。

Edit 命令使系統進入編輯模式，Rangelist 和 Dstring 用來指定編輯範圍和對象。如果 Rangelist 和 Dstring 皆省略，則被編輯檔的第一行至最後一行皆是編輯的對象。如只有 Rangelist，則在 Rangelist 所指定的範圍中每一行都是編輯對象。在 Edit 命令後面只有 Dstring 參數，在被編輯檔中，任一行只要有 Dstring，就是編輯對象。如果在 Edit 命令後面，Rangelist 及 Dstring 都沒有省略，在 Rangelist 所指定範圍中，任一行只要有 Dstring，就是編輯對象。

系統在編輯模式時，使用者可以使游標往左移動，往右移動，將一行中的某些字元用別的字元取代，插入字元，刪除字元。螢幕會立即顯示更改後的結果。在編輯模式時，所有的編輯功能皆以控制字元 (Control Character) 來指示編輯程式執行。如果使用者按入不被接受的控制字元，則 Apple 的喇叭會嗶一聲。經過編輯後的一行之長度並不受原來的長度限制，經過編輯後的長度可較原來的長度為長或短。按 RETURN 後，顯示在螢幕上已經過編校的一行將會取代在編輯緩衝區中未經編校的同一行。

使用 Edit 命令使系統進入編輯模式，使用者所指定範圍中的第一行將顯示在螢幕上，而且游標出現在此行第一個字的位置。使用者可用←和→鍵，使游標移至欲編輯字元的位置，然後可進行下列編輯功能：

更改字元

更改一行中的某字元時，按←或→鍵，使游標移至欲更改字元的

位置，然後按入新的字元。按入的字元將會取代在螢幕上與游標同一位置的字元，游標並移至下一字元的位置。如果連續按入新的字元，則一行中的字元將會一字一字地被新按入的字元取代。

插入字元

欲在某一字元前面插入新的字元時，按←或→鍵，使游標移至此字元的位置，然後按 CTRL - I，再按欲插入的字元。按入的字元會出現在游標的位置，而原先與游標同一位置的字元和游標右邊各字元將隨游標往右移一個字元的位置。如果連續按入字元，則這些字元將會由左至右地依序出現在游標的前面。←或→鍵，或者除了 CTRL - V 以外的控制字元均可使插入字元的動作結束，以便進行另外的編輯功能。

刪除字元

欲刪除一行中的某字元時，按←或→鍵使游標移至欲刪除字元的位置，按 CTRL - D，則欲刪除的字元被刪除掉，而游標右邊各字元將向左移一個字元的位置，游標的位置不變，因此，重覆按 CTRL - D 或同時按 CTRL - D 和 REPT 鍵，可連續刪除字元。

重新顯示

使用者在螢幕上編輯一行完畢後，按 RETURN，編輯程式就使用螢幕上經過編輯的一行來取代編輯緩衝區中未經編輯的一行。因此，在使用者按 RETURN 前，編輯緩衝區中欲編輯的這一行並未改變。如果使用者在編校某一行發生錯誤時，而希望重新編校此行，按 CTRL - R，則編輯緩衝區中未經編校的同一行會重新顯示在螢幕上。此時，使用者可重新在螢幕上編校此行。

尋找字元

如欲尋找游標右邊字元下次出現的位置，按 CTRL - F，如果找

到了，游標便跳至此字元的位置，如果沒有，則游標停留在原來位置。

控制字元 (Control Character) 有時很令人困擾：當你不需要它時，它出現了；當你最需要它時，它卻消失了。編輯程式提供兩種可以處理控制字元的功能：顯示 (popout) 及逐字處理 (Verbatim)。

POPOUT 以閃爍的方式顯示出一行中的控制字元。通常，在螢幕上是看不到控制字元。此功能使你能夠查驗控制字元在一行中的位置。在編輯模式時，當你於一行中移動游標向前或向後，且移至控制字元的位置，控制字元會以閃爍的方式顯示出來，控制字元以後的字元會移向右邊。如果你無法確定此字元是否為控制字元 (因為，游標也是以閃爍的方式出現)，僅須移開游標，如果此字元消失，則此字元為控制字元。

逐字處理 (Verbatim)，CTRL-V，的功能使你能以替代 (Replacement) 或插入 (Insertion) 的方法，將控制字元放置入你的檔中 (通常編輯模式不接受控制字元喇叭會有嗶一聲)。移動游標至欲被代替的字元之位置，然後按 CTRL-V，再按代替的控制字元。如果你希望更換更多的字元，則在每次按入新的字元之前，均須按 CTRL-V。

在游標前面加入控制字元。按 CTRL-I，然後按 CTRL-V 欲加入的控制字元。如果你希望加入更多的字元，則在每次按入新的字元之前，均須按 CTRL-V。按 CTRL-I 後，系統便處於插入模式 (insert mode)，按←或→鍵以移動游標，便可使系統離開此模式。

如果你已經完成所有的更改，或者決定不再進行任何更改，

可經由三種方法使系統離開編輯模式。第一種方法為，可在一行中的任何地方按 CTRL-X。這方法可完全停止編輯模式，甚至指定的編輯範圍並未全部經過編輯，然後系統回到命令模式，並使目前在螢幕上編輯的這一行保持原狀，縱然這一行在螢幕已經編輯過，但是在檔中的此行並未改變。

第二種方法為，按 RETURN，則目前在螢幕上編輯的這一行取代了檔中未編輯的此行，然後系統離開編輯模式。在檔中的此行，在使用者按 RETURN 之前，並不會改變。當使用者按 RETURN 後，在螢幕上編輯的這一整行取代檔中的此行，與游標在此行中的位置無關。下一個方法使一行中，游標右邊的被刪除，其餘的部分取代了檔中的此行。

第三種方法為，按 CTRL-T，則目前在螢幕上編輯的這一行中游標右邊的部分被刪除，此行剩下的部分取代了檔中未編輯的此行。

一旦編輯模式經由 RETURN 或 CTRL-T 結束，而指定的編輯範圍中，仍有其他未經編輯處理的行，則此行會顯示出來，游標出現在此行第一個字元的位置。如果指定的編輯範圍已全部經過編輯處理，則系統自動結束編輯模式，並回到命令模式。編輯模式之控制字元之綜合整理如下：

編輯模式控制字元	功 能 說 明
RETURN	以目前在螢幕上編輯的一行來取代檔中未經編輯的此行。
CTRL-X	完全停止編輯模式，目前在螢幕編輯的一行並沒有取代檔中未經編輯的此行。
CTRL-T	刪除目前在螢幕上編輯的這一行中游標右邊的

部分，並以未刪除部分來取代檔中未經編輯的此行。

- CTRL - R 將目前在螢幕上編輯的一行從螢幕上消除，並將在檔中未經編輯的此行重新顯示在螢幕上，以讓使用者能夠重新編輯此行。
- CTRL - D 將與游標同位置的字元刪除，被編輯的此行之長度將減少一個字元的長度。
- CTRL - I 在游標的前面加入字元，被編輯的此行之長度將增加一個字元的長度。
- 鍵 使游標往右移動。
- ←鍵 使游標往左移動。
- CTRL - V 逐字處理控制字元。
- CTRL - F 尋找游標右一字元下次出現的位置。

FIND

Find [Rangelist] [Dstring]

Find 命令是用來找尋在 Rangelist 所指定的範圍中含有 Dstring 的各行，並將其顯示出來。如果沒有指定 Rangelist，則 Find 命令尋找的範圍為整個欲編輯檔。如果沒有指定 Dstring，則 Rangelist 所指定的範圍均會顯示出來。如果 Rangelist 和 Dstring 皆沒有指定，整個欲編輯檔均會顯示出來。

INSERT

Insert Rlinenum

Insert 命令使系統進入輸入模式，Add 命令也可使系統進入輸入模式，並將使用者新鍵入的各行放在 Rlinenum 所指定的那行之前。例如，按 I1，則新鍵入的各行將依序放在第1行之前，

亦即原先的第1行將不再是第1行。欲結束輸入模式，按 CTRL - D 或 CTRL - Q。在 Insert 命令執行後，原先行號為 Rline - num 及以下各行之行號均會增加，因此，在執行下一個編輯命令之前，應確定各行新的行號。

LIST

List [Rangelist]

List 命令會將 Rangelist 所指定的範圍顯示在螢幕上，且各行的行號顯示在該行的前面。如果沒有指定 Rangelist，則整個檔將顯示出來。按 CTRL - C 可使顯示停止，系統回到命令模式。按空格鍵可使顯示暫時停止，再按除 CTRL - C 以外的任何鍵可使顯示繼續開始。如果再按的是空格鍵，List 命令會顯示下一行，並且再度暫時停止顯示。因此，空格鍵提供單步 (Single - Step) 顯示的功能。

指定的範圍顯示完畢後，系統回至命令模式。如果 Rangelist 指定許多個範圍，則 List 命令顯示一個範圍完畢後，會先顯示一行空格，然後再顯示下一個範圍。

PRINT

Print [Rangelist]

Print 命令的功能和 List 命令的功能一樣，但是經由 Print 命令顯示出來的各行，並沒有行號顯示於該行之前面。

REPLACE

Replace Range

實際上，Replace 命令的功能就是 Delete 和 Add 命令功能的結合。編輯程式先將 Range 所指定的範圍刪除，然後使系統進入輸入模式。此時，使用者可以經由鍵盤輸入新的各行。編輯程式

3.2 TOOL KIT 活用法

將使用者新輸入的各行，依序地放在 Range 所指定的範圍內。一旦此範圍的最後一行輸入完畢，系統就回到命令模式。Replace 命令後面的 Range 參數必須指定範圍的開始和結束。例如，10-20。僅有一個行號或 linenum 皆不合語法。例如，10，10—

2-6處理磁片和磁帶的命令

本系統有三個命令，可以讓使用者從磁碟中讀取文字檔，或將文字檔存入磁碟中。系統進行磁片輸入和輸出的方法與 DOS 手冊上的敘述之 BASIC 程式進行磁片輸入和輸出的方法一樣。在磁片上的文字檔之讀出與寫入是順序地 (Sequentially) 進行。本系統有另外二個命令，可以讓使用者從磁帶中讀取文字檔，或將文字檔存入磁帶中。編輯程式顯示於每行的前面之行號並不屬於文字檔的內容，且不會存入磁片或磁帶中。

編輯程式之磁碟機命令 (disk command) 是用來處理順序文字檔 (Sequential text file) 而不是隨機存取文字檔 (Random-access text file)。因此，編輯程式可以用來建立和修改 EXEC 檔，亦可檢視 (examine) 和修改由 BASIC 程式所寫成的順序資料檔 (Sequential data file)。

本節中的三個磁碟機命令使用了兩個參數：週邊連接插座的號碼 (slot number) 和磁碟機的號碼。磁碟機命令並沒有提供這兩個參數。當本系統啟動時，此兩個參數分別被設定為啟動週邊連接插座 (boot slot) 和 1 號磁碟機。這兩個參數可經由 SLot 和 DRive 命令來更改。SLot 和 DRive 命令將會在下一

節討論到。

使用任一磁碟命令，經常出現的 DOS 錯誤訊息有：DISK FULL ERROR, I/O ERROR, NO BUFFERS AVAILABLE 或 SYNTAX ERROR。本手冊假設你對 DOS 和這些錯誤的訊息相當熟悉。附錄 B 編輯程式顯示的 DOS 錯誤，討論了本系統最常出現的 DOS 錯誤，引起這些錯誤的可能情況，和錯誤發生後的補救辦法。其他的 DOS 錯誤訊息，DOS 手冊有詳細的討論。

LOAD

LOAD Rfile name

LOAD 命令是用來從 SLot 和 DRive 命令所指定的磁碟機中讀取 Rfilename 所指定的檔 (關於 SLot 和 DRive 命令請參閱 2-7)。編輯程式會將此檔讀至編輯緩衝區。當讀取完畢時，Apple 的喇叭會嗶一聲，螢幕上顯示 END OF DATA，然後系統回至命令模式。此時，使用者可用編輯命令來處理在編輯緩衝區中的檔。

如果在磁碟中並沒有 Rfilename 所指定的檔，而磁片沒有 Write protected，則 Rfilename 所指定的檔名會新建立在 Catalog 中。因而建立了一個空白檔 (null file) 在磁片上，且沒有任何資料被讀至編輯緩衝區。一般而言，我們並不希望建立空白檔在磁片中，所以發生以上所述的情況時，可以使用 DOS 的 .DELETE 命令將此空白檔的檔名從 Catalog 中刪除。由 LOAD 命令讀至編輯緩衝區的文字檔經過編輯處理後，使用者必須使用 SAVE 命令將經過編校處理的文字檔存回磁片，此時 SAVE 命令可以不用指定檔名。

SAVE

SAVE [Rangelist] [Gfilename]

SAVE命令是用來將編輯緩衝區中的整個文字檔或其中一部分存入SLOT和DRIVE命令所指定的磁碟中。如果磁片是write protected，則發生DOS的錯誤，然後系統回至命令模式。經編校過的文字檔所佔用的空間，可能比原來的檔所佔用的空間為大，亦可能較原來的檔所佔用的空間為小。因此，使用SAVE命令時，系統會將磁片中原來的檔刪除，然後再將經過編校後的文字檔存入磁片。

使用本系統時，最好使用DOS的LOCK和UNLOCK的命令來保護磁片上其他的文字檔，UNLOCK欲編校的文字檔，而LOCK其他的檔，則SAVE命令不會因人為的疏忽，而將與被編輯檔檔名相似的檔刪除。因為有些檔的檔名，互相之間僅有一個或兩個字母不同而已。

注意：不可使用DOS的LOAD命令，因為這個命令的意義為“從磁片中讀取BASIC程式”。如使用LOAD命令，DOS會將Rfilename所指定的文字檔當成BASIC程式來讀取，則讀進來的資料會將在記憶體中的編輯程式覆蓋掉。亦不可使用DOS的SAVE命令，因為這個命令的意義為“將目前BASIC程式的內容存至磁片中”。由於沒有這樣的程式，DOS用來決定程式大小的指標(pointer)便存著沒有用的數據，因此，這個命令執行後的結果將是無法預料的。

如果SAVE命令後面的參數為Rangelist，亦即同時指定許多個Range。編輯程式只將第一個Range所指定範圍的資料存入磁碟中，其他的Range則忽略。

在使用SAVE命令之前，曾經用過LOAD或其他的SAVE命令，且有指定的檔名與目前SAVE命令所指定的檔名相同，則目前的SAVE命令可以將檔名省略。SAVE或LOAD命令後面跟有檔名，此檔名將被保存起來，此檔名便成為機定檔名(default filename)。如果往後的SAVE命令後面沒有檔名，則此機定檔名便成為經SAVE命令儲存之資料的檔名。在任何時候，你可以使用FILE命令來找出機定的檔名。如果前幾個的SAVE和LOAD命令沒有指定檔名，而後來使用的SAVE命令亦沒有指定檔名，則會出現DOS SYNTAX ERROR，然後系統回至命令模式。

發生DOS錯誤以及SAVE的動作停止，或許已有部分的資料已存入磁片，此時可用DELETE命令將此不完整的檔刪除。而且前在編輯緩衝區中的被編輯檔仍然存在，縱然有DOS的錯誤發生。因此，使用SAVE命令，有DOS的錯誤發生時，最好使用另外一片磁片，再試一次SAVE命令。

使用者如果鍵入或編輯一個相當大的程式時，大約每20分鐘的鍵入或編輯後，就應做一次SAVE的動作。因為僅是一秒鐘的電源中斷就足以使數小時的輸入和編輯之辛勞付諸流水。不要一直忙碌地輸入或編輯而忘了做SAVE的動作。

APPEND

APPEND [Olinenum] Rfilename

APPEND命令一般是用來讀取Rfilename所指定的檔，並將其加於目前在編輯緩衝區中的被編輯檔之後面。如果編輯緩衝區中沒有任何檔存在，APPEND命令仍有作用，此時APPEND命令的作用與SAVE命令的作用一樣。

如果 APPEND 命令的後面有參數 Olinenum 時，則 APPEND 命令從磁碟讀取的檔是附於被編輯檔由 Olinenum 指定的那行以後。

TLOAD

TLOAD [Olinenum]

TLOAD (Tape LOAD) 命令是用來讀取經由 TSAVE 命令存入磁帶中的磁帶檔。使用錄放音機讀取資料時，經常會發生一些問題，因此使用者須能夠解決這些問題，和調整錄放音機以使資料能夠順利地讀取。當你認為需要任何幫助時，可參閱 Apple II BASIC programming Manual 或其他 Apple 手冊。此命令讓使用者能夠使用較便宜的磁帶來儲存資料。

TLOAD 命令後面有 Olinenum 參數時，則 TLOAD 命令讀取的磁帶文字檔是加於被編輯檔由 Olinenum 指定的那行之前。這是一個唯一不需經由鍵盤輸入而能直接將一段程式加於被編輯檔中間的命令。APPEND 命令後面跟著 COPY 和 Delete 命令也可達成從磁碟讀取資料而加於被編輯檔中間的功能。

TSAVE

TSAVE [Rangelist]

TSAVE 命令是用來將被編輯檔以本系統的磁帶檔格式存入磁帶來。如果此命令後面沒有 Rangelist 參數，在編輯緩衝區中的整個被編輯檔將被存於磁帶中。如有 Rangelist 參數，Rangelist 所指定的範圍將存於磁帶中。編輯緩衝區中如沒有任何資料，TSAVE 命令仍將 header 存於磁帶中，然後系統回至命令模式。由於本系統磁帶檔案的格式與 BASIC 程式的磁帶格式不一樣，因此讀取兩者的方法也不一樣。經由 TSAVE 命令存入磁

帶中的資料其格式如下：

(為時 10 秒的 header)

(16 位元檔的長度)

(為時 10 秒的 header)

(檔的資料)

雖然使用本系統時，一定要有磁碟機，因此使用卡式錄音機來存取資料並不是常用的方法，但是舊的程式或文字檔可以利用較便宜的磁帶來儲存。

2-7 操作命令 (Operatating Command)

本節所討論的各種命令中，有些命令可以用來改變某些命令的參數，有些命令可讓使用者取得某些有用的資料。

SLOT

SLOT Rnumber

SLOT 命令是用來設定磁碟機控制卡所在的週邊連接插座的號碼，系統便將此數值存於記憶體。往後有關磁碟機操作的命令皆以此週邊連接插座為主。當系統啟動時，此號碼自動設定為系統磁片所在磁碟機之週邊連接插座的號碼。SLOT 命令一般很少使用，系統提供此命令給予具有一片以上磁碟機控制卡的使用者使用。參數 Rnumber 必須為有效的週邊連接插座號碼，1 ~ 7。系統並沒有查驗 Rnumber 是否正確，或者指定的週邊連接插座上是否有磁碟機控制卡。如果 Rnumber 不正確，或者指定的週邊連接插座上沒有磁碟機控制卡，則下次使用有關磁碟操作的命令時，系統會在螢幕顯示 I/O ERROR。

D R I V E

DRive 1

或

DRive 2

在週邊連接插座上的磁碟機控制卡可接兩部磁碟機，DRive 命令就是用來設定磁碟機的號碼，1 或 2。SLOT 和 DRive 這兩個命令所設定的參數是用來指定磁碟機。往後系統有關磁碟機的操作均在此磁碟機進行。SLOT 命令和 DRive 命令所設定的數值是存於記憶體中，而不是存在磁片上。所以，一旦系統重新啟動時，系統自動將 DRive 命令所設定的數值定為 1，將 SLOT 命令所設定的數值定為系統磁片所在磁碟機之週邊連接插座的號碼。

C A T A L O G

CATalog

這個命令僅是一個執行 DOS CATALOG 命令比較簡單的方法。如果使用直接 DOS 命令 (Direct DOS Command) 則為 .CATALOG, Dd, Ss

其中，Dd 是磁碟機號碼，Ss 是週邊連接插座的號碼。此 CATalog 命令和直接 DOS 命令的差別有兩點：(a) CATalog 命令可用縮寫，亦即 CAT。(b) CATalog 命令的後面不可指定磁碟機號碼和週邊連接插座的號碼。

F I L E

FILE

FILE 命令是來將目前 SAVE 的檔名和由 LENgth 命令所提供的資料顯示在螢幕上。如果沒有任何檔經由 SAVE 命令和 LOAD 命令指定，則僅有 LENgth 所提供的資料顯示出來。

H I M E M =

HImem = Rnumber

此命令是用來設定編輯緩衝區的上限位址 (highest available address) HIMEM，亦即被編輯程式在記憶體中能使用的最高位址。在系統啟動時，編輯緩衝區的上限位址設定為可能的最大位址。如果使用者欲縮小編輯緩衝區，以取得此一記憶體作為他用時，可使用此命令來設定編輯緩衝區的上限位址，以使編輯緩衝區縮小。如果在編輯緩衝區內已有資料，此時使用 HIMEM 命令將會有點冒險。編輯緩衝區上限位址設定太小，將使得編輯緩衝區不足容納欲編輯的文字檔，因此，欲編輯的文字檔將有一部分會損失掉。

參數 Rnumber 表示用來設定編輯緩衝區上限位址的十進位數值。系統並沒有查核 Rnumber 指定的數值，如果此數值設定不恰當，將會使系統的操作迅速地破壞掉。一般而言，此命令很少使用，而且編輯緩衝區上限位址設定不當而使得編輯緩衝區無法容納輸入的欲編輯檔，此時就會發生“OUT OF MEMORY”的錯誤。

擁有 48K 記憶體的 Apple，系統設定編輯緩衝區上限位址為 38400，36K 的上限位址為 26112，32K 的上限位址為 22016。從上述的值減去編輯緩衝區欲縮減的數值，則所得的結果即為已縮減的編輯緩衝區之上限位址。如果編輯緩衝區的下限位址 LOMEM (lowest memory address) 沒有改變，其機定值 (default value) 為 8192，將此值加上 LENgth 命令所顯示的數值，所得的結果即為在編輯緩衝區中欲編輯的文字檔所使用之最高位址。

LOMEM =

LOmem = Rnumber

此命令是用來設定編輯緩衝區的下限位址 (lowest memory address) LOMEM。編輯程式使用了緊鄰此位址之下的記憶體。LOMEM在系統啟動時自動被設定為 8192，此值即為機定值 (default value)。如果使用者設定LOMEM之值小於 8192，將會使編輯程式遭到破壞。

LENGTH

LENgth

LENgth命令是用來顯示在編輯緩衝區中的欲編輯文字檔使用記憶體空間的數量，以 byte 和字元的數量表示，以及剩下多少記憶體空間尚未使用。這兩個數值的和為編輯緩衝區之大小，亦即編輯程式所能使用的記憶體空間。

MON

MON

MON命令提供一個至監督程式 (monitor) 的簡單途徑，亦可很容易地回到命令模式。除非使用者欲做一些較特殊的事情，一般而言，此命令並不常用。按 CTRL - Y 可使系統從監督程式回到命令模式。至監督程式後，使用者欲做任何事情皆可，但是本系統所使用的記憶體區域則不可破壞。有關本系統所使用記憶體區域，請參閱附錄編輯程式使用之記憶體及用途。MON 命令為熟練的 Apple 使用者及程式規劃者之工具，初學者應避免使用。從監督程式至 BASIC 的兩種方法之任一種皆可能破壞本系統所使用的記憶體空間。按 “ 3D0G ” 亦可從監督程式回到命令模式。

NEW

NEW

NEW 命令能使編輯程式清除編輯緩衝區中的資料。實際上，編輯緩衝區的內容並沒有破壞。此命令僅是設定在編輯緩衝區的文字檔之結束指標 (end-of-text pointer) 等於開始指標 (beginning-of-text pointer)。開始指標之值即為編輯緩衝區的下限位址LOMEM。因此，有可能經由監督程式的命令，使被NEW 命令清除的文字檔再回到編輯緩衝區。這需要直接研究編輯緩衝區及其指標，以決定結束指標原來的值。

PR#

PR# p [, Dev Ctlstrg]

此命令是用來通知編輯程式，將 assembly listing 輸出至其他裝置，不同於Apple 的螢幕顯示器。通常此裝置為接至列表機介面卡的列表機，亦可為 80 行的終端機或其他類似的輸出裝置。p 表示輸出裝置所連接的週邊連接插座之號碼，其值為 0 ~ 7 的正整數。如果 p 為 0，則表示輸出裝置為Apple 的螢幕顯示器。

Assembly listing 僅傳至一種輸出裝置。但使用者可以設計使傳送至選定的輸出裝置之資料回傳 (echo) 至Apple 的螢幕顯示器。Apple 列表機介面卡無法將接收到的資料立即回傳至Apple 的螢幕顯示器。

“Dev Ctlstrg” (Device Control String) 表示須傳送至列表機的控制字元串，以啟動列表機，使其執行 Assembly listing 的列表工作。此字元串的字元數目不可超過 32 個。此字元串分為三部分。第一部分為Logical page 的長度，第二部分為Physical page 的長度。這兩部分可以依使用者的意

願加以取捨。Logical page 的長度為印在一頁上的行數，以 L 及兩個數元 (two-digit) 的數值表示。Physical page 的長度為一個表格的開始 (top-of-form) 至另一個表格的開始之行數，以 P 及兩個數元的數值表示。Logical page 的機定長度為 60。如果 physical page 的長度沒有指定，且組合語言程式中有使用 SBTL (請參閱 3-6)，則每一頁後就會進行 form feed。

第三部分則不可省略。以 Apple 列表機介面卡而言，此部分為 CTRL - I 80N 的字元串 (CTRL - I 為一個控制字元，而不是 6 個字元的字元串)，或類似的字元串。此字元串關掉 Apple 的螢幕顯示器，設定列表機每行所印的字數，以適應列表機所使用的紙張。

這裏有個例子，這例子說明了 PR# 命令的用法。從鍵盤按入

```
PR# 1, L54 P66 CTRL - I 80 N
```

則螢幕顯示

```
PR# 1, L54 P66 80 N
```

此命令會將 Assembly listing 傳送至連接在 1 號週邊連接插座的列表機。此命令使列表機在長度為 66 行的一頁上印 54 行，每一行 80 個字元。

此命令的參數是存在記憶體中，而不是磁片。因此，如果需要的話，每次譯碼組合後，此命令須再使用，以使下一次的譯碼組合之 Assembly listing 傳送至指定的輸出裝置。

TRUNCATE

TRunc ON

或

TRunc OFF

此命令提供了截割 (truncation) 螢幕顯示的功能。在系統啟動後，此一功能沒有作用，但可經由 TR ON 的命令使此截割的功能有作用。將顯示在螢幕上的欲編輯檔之各行截短，以配合 Apple 每行 40 字的螢幕顯示，此為截割功能的目的。此命令並沒有改變欲編輯檔的內容，只是改變經由 List 及 Print 命令顯示出來的資料。

每一行的截割點是在一行中第一次出現分號的位置，且此分號前面須有一個空格，以 6502 組合語言而言，如果一行中含有註解，則註解部分從此分號開始。如果不顯示出一行中的註解部分，將使顯示在螢幕的原始程式更容易閱讀，因為，註解部分的顯示經常會延伸至第二行。在沒有列表機，且所有的輸出輸料皆傳送至螢幕的情況，截割的功能將使螢幕的顯示較整齊且容易閱讀。系統在編輯模式時，截割的功能自動失效。

TABS

Tabs [Tablist] [Dstring]

Tabs 命令能讓使用者使用空格鍵 (或其他鍵) 如同打字機的表格定位鍵 (TAB Key) 一般，從鍵盤輸入原始程式時，能夠將其格式化。系統啟動後，表格定位自動設定為 6502 組合語言的標準格式。

編輯程式顯示在螢幕的各行中，僅在前面的 40 個字，表格定位的功能才有作用。編輯程式將忽略任何超過第 40 個字的表格定位設定。如果運算元中有空格，則表格定位功能將會把空格後面的部分視為註解，因而將此部分移至註解欄。所以運算元 (

operand) 中不可含有空格。

Tablist 表示一列 (a list) 的絕對位置 (absolute position)。這列最多可擁有 16 個絕對位置。所謂絕對位置是指表格定位字元 (tab character) 之後的字元位於顯示在螢光幕中的位置，在 Tablist 的絕對位置必須以增加的方式排列。例如，

T 15 , 25

此命令的意義為：編輯程式在顯示文字檔時，在一行中遇到第一個表格定位字元時，將第一個表格定位字元後的字元從 Column 15 開始顯示，遇到第二個表格定位字元時，將表格定位字元後的字元從 Column 25 開始顯示。如果 Tabs 命令的後面沒有 Tablist 參數，則表格定位的功能不作用。Dstring 參數是用來設定表格定位字元，此字元可為除了 RETURN 外的任何字元。空格字元為機定的表格定位字元，因為，空格字元是組合語言原始程式中最適當的表格定位字元。編輯程式允許使用任何字元做為表格定位字元，但是組合語言譯碼程式只允許空格字元做為表格定位字元。

在顯示各行時，一旦遇到表格定位字元，表格定位的功能就有作用。這表示說，在編輯緩衝區中的欲編輯檔之內容能夠控制顯示時的表格定位功能。

W H E R E

Where Rlinenum

這個命令是處理在編輯緩衝區中的欲編輯檔的工具。此命令可提供欲編輯的文字檔中任一行在記憶體中的十六進制位址。亦可使用此命令來找出編輯緩衝區的下限位址 LOMEM。使用此命令來找出 LOMEM 的方法等，從鍵盤輸入

W 1

然後按 RETURN。則螢幕會顯示 “= \$ 2000”，此數值相當於十進制的 8192。

E N D

E N D

此命令是用來離開本系統，然後回到 BASIC 語言。END 命令提供一個離開本系統的簡便方法。如果使用者，不經意地使用此命令離開本系統。使用者亦可由 Integer Basic 回至本系，方法為，從鍵盤按入

C A L L 3075

然後按 RETURN。但是，使用者在使用 SAVE 命令來儲存編輯緩衝區中的欲編輯檔之前，必須確定在編輯緩衝區中的資料並沒有改變。

第 3 章

組合語言譯碼程式 (Assembler)

3-1前言

此一組合語言譯碼程式 (Assembler) 是為便利程式設計者在 Apple 上編寫組合語言程式而設計的。它提供許多個組合語言譯碼指令 (Assembler directive) 或稱為假指令 (Pseudo instruction)。原始程式先由程式設計者利用編輯程式 (Editor) 建立，並且依照本章所敘述的格式編寫。由於，此組合語言譯碼程式是從磁碟中讀取原始程式，然後進行譯碼組合 (Assembling)，所以，經由編輯程式建立的原始程式在進行譯碼組合之前，須先儲存至磁片上。

ASM 命令是用來呼叫組合語言譯碼程式。組合語言譯碼程式將原始程式中的指令轉換為 6502 機器碼，然後產生輸出目的檔 (output object file) 儲存在磁片上。此組合語言譯碼程式的譯碼組合程序為 Two-Pass。Pass-One 時，它從磁碟讀

取原始程式，建立符號表 (Symbol table)，並指定數值給程式設計者所定的符號標記 (label)，決定所有指令的長度，在運算元 (Operand) 這一欄中，任何尚未定義的符號標記，組合語言譯碼程式皆將其數值大小定為 16 bits。因此，數值大小為 8 bits 的符號標記，程式設計者必須在此符號標記出現在運算元這欄之前，指定數值給此符號標記。在 Pass-two 時，組合語言譯碼程式再度從磁碟讀取原始程式，然後產生完整的機械碼。它參考符號表，以便填上指令中，位址的部分。在 pass-two 時，組合語言譯碼程式一次寫一段 (256 Bytes) 目的程式 (object program) 於輸出目的檔。這個檔的型式為二進制檔 (Binary file) 或可再定位檔 (Relocatable file)。

在 pass-one 和 pass-two 時，如果組合語言譯碼程式遇到錯誤，它會將錯誤顯示出來。它在 pass-two 時，產生組合語言表列 (Assembly listing)，然後將此表列顯示在 Apple 的顯示幕，或使用使用者所指定的其他輸出裝置。

使用者可經由 REL 假指令，要求組合語言譯碼程式產生可再定位目的檔 (Relocatable object file)。此種型式的 DOS 檔提供將程式再定位時所需的資料，以使程式在記憶中任何位址皆可執行。

將程式再定位的工作是由再定位載入程式 (Relocating Loader) 來執行，而且不需重覆譯碼組合，即可達成程式再定位的功能。這些在程式再定位時所需的資料構成 RLD (Relocation Dictionary)，組合語言譯碼程式將 RLD 放在目的檔的後面。

組合語言譯碼程式的呼叫是由 ASM 命令來執行。關於 AS

M 命令的功能及語法，請參閱 3-2。編輯程式和組合語言譯碼程式並不同時在記憶體裏，當系統在命令模式時，僅命令釋義程式和編輯程式在記憶體裏。因此，在進行譯碼組合 (assembling) 之前，須先用 BLOAD 命令將組合語言譯碼程式從磁碟讀至記憶體中。這個動作在使用者按入 ASM 命令後，命令釋義程式會自動執行。

並非所有的錯誤在系統讀入組合語言譯碼程式和執行譯碼組合之前就能夠被系統發明。例如，使用者指定了一個檔名，而在磁碟中並沒有這個檔，當 DOS 嘗試打開這個檔的時候發現並無此檔，則螢幕上會出現

OOPS DOS ERROR CODE = ??

然後系統回至命令模式。當發生有關 DOS 的 I/O 錯誤或在譯碼組合過程時記憶體不夠使用，組合語言譯碼程式會顯示出上面的錯誤訊息。?? 表示有關的錯誤之 ERROR CODE。附錄 D 說明了各種 ERROR CODE 所代表的錯誤。組合語言譯碼程式會將有關原始程式的錯誤顯示在輸出裝置，此一輸出裝置可為螢幕或印表機，但不會將錯誤訊息同時傳至螢幕及列表機。

本系統的組合語言譯碼程式從磁碟讀入原始程式，然後進行譯碼組合，再將譯碼組合後的目的碼 (object code) 存入磁碟。此組合語言譯碼程式並非從記憶體中讀取原始程式，亦非將輸出目的檔存入記憶體。

如果在編輯緩衝區中存有已經編輯過的文字檔或是新建立的文字檔，在使用 ASM 命令要求組合語言譯碼程式進行譯碼組合之前，使用者必須先用 SAVE 命令將編輯緩衝區中的文字檔存至磁碟。否則，系統從磁碟將組合語言譯碼程式讀至記憶體，將

會破壞編輯緩衝區中的文字檔。

3-2 如何使用組合語言譯碼程式

ASM 命令是用來要求系統將組合語言譯碼程式從磁片讀至記憶體中，然後由組合語言譯碼程式從磁片中讀取存於使用者所指定文字檔 (Text File) 中的原始程式，並將其譯碼組合 (Assembling)。系統從磁片讀取組合語言譯碼程式，並將其放在與編輯緩衝區相同位址的記憶體中。因此，使用者必須在使用 ASM 命令之前，先將在編輯緩衝區中經過編校過的程式存至磁片，否則 ASM 命令執行後，在編輯緩衝區中的內容將會被破壞。ASM 命令的語法 (Syntax) 如下：

ASM Rfilename [, Objfilename, Ss, Dd]

ASM 命令是用來指示系統將組合語言譯碼程式從磁片中讀至記憶體中，系統並傳送由 Slot 和 Drive 命令所設定兩個參數給組合語言譯碼程式 (關於 Slot 和 Drive 命令，請參閱 2-7)，然後由組合語言譯碼程式根據 Slot 和 Drive 命令所指定的磁碟中讀取原程式。在這種情況下，ASM 命令中，Ss 和 Dd 便可省略。如果使用者欲另外指定磁碟機時，則 ASM 命令中的 Ss 和 Dd 參數就是用來指示組合語言譯碼程式從那部磁碟中讀取原始程式。Ss 的型式為 S1 ~ S7 或字母 S 省略，為 1 ~ 7。Dd 的型式為 D1 或 D2，或者字母 D 省略，為 1 或 2。

Rfilename 為不可省略的參數，亦即，使用 ASM 命令時，必須指定一個檔名。組合語言譯碼程式便根據 Rfilename 所指定的檔中去讀取原始程式。此一原始程式必須是由 6502 組合語言

所編寫的，並且須符合本組合語言譯碼程式所要求的格式。使用者先使用編輯程式在編輯緩衝區中建立一原始程式，然後用 SAVE 命令將此一原程式存入磁片成爲一文字檔。或者利用 LOAD 命令將存於磁片中某一文字檔的原始程式讀至編輯緩衝區，然後加以編校修正，再使用 SAVE 命令將經編校修正後的原始程式存入磁片。這時才使用 ASM 命令，進行譯碼組合。

Objfilename 是用來指定組合語言譯碼程式的輸出檔名。本系統的組合語言譯碼程式將原始程式譯碼組合後得到的 6502 機器碼直接存入磁片，而不是存於記憶體中，因此須設定一個檔用來存譯碼組合後的結果，此檔之檔名即爲 Objfilename 所指定的檔名。如果使用者將 Objfilename 參數省略，組合語言譯碼程式便自動設定存譯碼組合後的結果之檔名爲 Rfilename.OBJ。例如，存原始程式的文字檔之檔名爲 TITLE，使用 ASM 命令後，其中 Objfilename 參數省略，則組合語言譯碼程式自動將輸出檔名定爲 TITLE.OBJ。

在 ASM 命令後面的參數，須按照順序排列。使用者如將 ASM 命令後面某一參數省略，但此參數後面另一參數並沒有省略，則省略參數前面的逗點不可省略。例如，

```
:ASM PROGRAM,,,D2
```

其中 Objfilename 和 Ss 參數省略，而這兩個參數後面仍有 D2 這個參數，因此，Objfilename 和 Ss 參數前面的逗點並沒有省略。下面列舉三個較典型的 ASM 命令型式。讀者可從這三個典型的 ASM 命令型式中，瞭解 ASM 命令的使用方法。

```
:ASM MYFILE
```

此命令要求系統將組合語言譯碼程式從磁碟中讀入記憶體，然後

從同一磁碟讀取原始程式，此原始程式存於檔名爲 MYFILE 的文字檔。譯碼組合後的輸出檔爲 MYFILE.OBJ。組合語言譯碼程式將輸出檔直接存入磁片。

```
:ASM MYFILE,MYPROGRAM
```

此一命令除了使用者指定輸出檔名爲 MYPROGRAM 外，其餘與上一個 ASM 命令相同。

```
:ASM BIGFILE,BIG.OBJ,,,D2
```

此命令要求系統將組合語言譯碼程式從原先 Slot 和 Drive 命令所指定的磁碟中讀入至記憶體，然後由組合語言譯碼程式從與目前使用磁碟機同一槽號 (Slot Number) 的第 2 部磁碟機中讀取原始程式，並將其譯碼組合，並將結果存於檔名爲 BIG.OBJ 的檔。

組合語言譯碼程式在執行譯碼組合之前，會在螢幕上顯示 “PRESS ANY KEY TO CONTINUE”，此時，使用者可以將系統磁片取出，然後將存有欲譯碼組合的原始程式之磁片放入磁碟中，再按任一字鍵，除了 RESET 鍵以外，以通知組合語言譯碼程式開始譯碼組合。一旦譯碼組合完畢後，螢幕上會出現相同的字幕。此時，使用者應將系統磁片放入磁碟機，然後按任一字鍵。系統就從磁碟中，將編輯程式讀入至記憶體，然後進入命令模式。

3-3 控制譯碼組合程序的命令

本系統的組合語言譯碼程式提供了三個命令，這三個命令可以在組合語言譯碼程式進行譯碼組合時控制其動作。

中止譯碼組合——CTRL - C 可以讓組合語言譯碼程式停止譯碼組合。組合語言譯碼程式在進行譯碼組合時，會隨時察看鍵盤，一旦有CTRL - C 按入時，它便中止譯碼組合，並且將打開的檔予以關閉，並將DOS緩衝區開放使用，然後系統回至命令模式。組合語言譯碼程式中止譯碼組合後並沒有將輸出檔刪除。系統回至命令模式時，使用者可以用 .DELETE 命令來刪除輸出檔。暫停或單步列表 (Single-Step Listing) ——組合語言譯碼程式在 pass two 時進行列表，如果使用者在其列表時按入空格鍵，則組合語言譯碼便停止列表，直到使用者再按任一字鍵，才繼續列表。使用者可以使用空格鍵來進行單步列表。亦即，按一次空格鍵，顯示一行於螢幕上。使用空格鍵使組合語言譯碼程式停止列表，亦會使其譯碼組合的工作停止，而且磁碟機的馬達亦保持旋轉不停。

顯示程式中一部分於輸出裝置——此組合語言譯碼程式有一個假指令 (directive 或 pseudo op)，LST ON 或 LST OFF，可放在原始程式中，用來指示組合語言譯碼程式做列表的工作。組合語言譯碼程式執行列表時，如在原始程式中遇到 LST OFF 假指令時，它便停止顯示程式，直到遇 LST ON 假指令時，再繼續顯示程式於輸出裝置。CTRL - N 的功能與 LST OFF 假指令相同，CTRL - O 的功能與 LST ON 的功能相同。pass two 時，從鍵盤按入 CTRL - N 可以停止將程式顯示於輸出裝置，直到組合語言譯碼程式在原始程式中遇到 LST ON 假指令，或使用者按入 CTRL - O，才繼續將程式顯示於輸出裝置。

3-4 原始程式的格式

用來存原始程式的檔為DOS文字檔，亦即為組合語言譯碼程式的輸入檔。此檔是由許多個邏輯記錄 (logical record) 所構成。一個字元串 (Character String) 為一個邏輯記錄，此字元串最後的一個字元必為回機字元 (carriage return)。所有字元的ASCII值之MSB必須為1。每一個邏輯記錄共分為4欄：標記 (label)，運算碼 (opcode)，運算元 (operand) 和註釋 (comment)。每一欄通常是由一個空格加以分開。一個邏輯記錄應有一個運算碼，除非這個記錄純粹為註釋。一個邏輯記錄必須有運算碼，而標記，運算元，註釋則視情況取捨。僅有回機字元的記錄為不合語法。組合語言譯碼程式使用空格字元作為其表格定位字元 (tab character)，因此每一欄間應以一個空格分開，兩個空格則表示前面的一欄省略。各欄的格式如下：

標記欄——其格式如下：

label

或

label :

標記欄可依情況而取捨，任一邏輯記錄均可加上標記，除了純粹為註釋的記錄。標記為一個由1至250個字元組成的字元串，這些字元僅可為字母 (letter)，數元 (digit) 和句點。標記的第一個字元須為A - Z的字母。一個標記的字元數目實際上限制為16。冒號或空格表示標記的結束 (或是除了字母，數元，

句號和 RETURN 以外)，此一字元並不被視為標記中的字元。

標記必須從邏輯記錄的第一個字元位置開始。在原始程式中，任一標記僅能在標記欄出現一次，亦即任一標記僅能定義一次，否則，組合語言譯碼程式將顯示出“DUPLICATE SYMBOL”。在邏輯記錄中的標記欄如有標記時，組合語言譯碼程式會計算出此記錄中運算碼的位址，然後將此位址值指定給此標記。

如果標記出現在運算元這一欄，組合語言譯碼程式在譯碼組合時，將用已經指定給此標記的值取代在運算元這一欄出現的標記。絕對值 (absolute value，亦即其數值大小為 16 個位元) 的標記可在程式中任何位置定義其值。零頁 (Zero Page，亦即其值大小為 8 個位元) 的標記則必須在它出現於運算元這欄之前予以定義。如此，組合語言譯碼程式在 pass one 時，才能將指令的 Byte 數計算出來。如果零頁標記在它出現於運算元這一欄之前沒有定義，則組合語言譯碼將會視其為絕對值標記。

運算碼欄——邏輯記錄中第 2 欄為運算碼這一欄。此欄與前面的標記欄之分隔為一個空格字元。如果標記欄省略，則運算碼這一欄前面至少須要有一個空格字元。運算碼為一由 2 個或 3 個字元組成的字元串，此字元串稱為簡字碼 (Mnemonic Code)。

運算碼這欄中的運算碼可為簡字碼或組合語言譯碼程式的假指令。在本章的後面附有 6502 簡字碼和假指令。組合語言譯碼程式的假指令是用來指揮譯碼組合程序的執行。

運算元欄——有些操作碼需要運算元，而某些操作碼並不需要運算元。因此，一個邏輯記錄中，運算元這一欄是否需要則須根據運算碼而定。運算元一般為一個算術式子 (expression)，這

個算術式子是由常數 (constant)，標記和算術運算子 (arithmetic operator) 構成，其運算規則為由左至右，而非先乘除後加減。運算元亦可為常數或標記。讀者如對 6502 的指令及定址模式 (addressing mode) 不熟悉，在編寫原始程式之前，請先參閱有關 6502 組合語言的書籍。因為本組合語言譯碼程式是為 6502 組合語言設計的。

上面已經提過運算元可為常數，標記或由常數，標記及算術運算子所構成的算術式子。在 pass two，所有標記的值均已指定後，組合語言譯碼程式會計算出運算元的數值。在 pass one 時，組合語言譯碼程式必須決定每個指令敘述的 Byte 數，因此，它僅根據算術式子的第一項來決定算術式子之值為 16 位元或是 8 位元。如果第一項的值為 16 位元，則組合語言譯碼程式亦將算術式子之值設定為 16 位元。如第一項的值為 8 位元，則算術式子之值設定為 8 位元。標記，常數，算術式子三種運算元的型式說明如下：

標記

以 6502 組合語言而言，標記的種類可分為兩種，分別是其值為 8 位元和 16 位元的標記。第一種的標記亦稱為零頁標記 (Zero page label)。出現在標記欄的標記，其值將被定為程式計數器 (program counter) 之值。如果標記出現在 E Q U 假指令前面的標記欄，組合語言譯碼程式就會將 E Q U 假指令後面運算元這欄的計算結果指定給標記欄的標記。

組合語言譯碼程式在計算運算元這欄結果時，如遇到標記，則以原先指定給標記的值取代在運算元這欄中的標記，然後將運算元這欄結果計算出來。

常數

此系統的組合語言譯碼程式允許使用在運算元這欄的常數有四種：字元串常數 (String Constant)、十進位常數、十六進位常數和八進位常數。常數是用來表示實際的資料，如 ASCII 字元，數據表 (table of data)，位址補償值 (address offset)，硬體裝置的位址，或 6502 微處理機設計上所設定的固定位址。常數的值為固定不變，而標記的值會因為程式的修改和程式重新定址於記憶體的其他地方而改變。

(1) 十進制常數。此常數表示以 10 為底的數目，其值為 0 到 65535 的正整數。一個數目的前面沒有任何符號，組合語言譯碼程式便將其視為十進制常數。

(2) 十六進制常數。此常數表示以 16 為底的數目，用來表示 16 進制常數的字元，除了 0 ~ 9 的數元外，字元 A ~ F 分別表示 10 ~ 15 之數值。一個數目的前面有 \$ 的符號，則表示這個數目為 16 進制。此種常數的數值為 \$0000 ~ \$FFFF，相當於十進制的 0 ~ 65535。

(3) 八進制常數。此常數表示以 8 為底的數目。0 ~ 7 為用來表示 8 進制常數的數元。數目的前面有 @ 符號，表示此數目為八進制常數。此種常數的數值從 @0 ~ @177777，相當於十進制的 0 ~ 65535。

(4) 字元串常數。此種常數是用來表示字元串中各字元的 ASCII 值，表示方法是在字元串的兩邊各加上單引號，如 'MYFILE'。字元串的長度不可超過一個邏輯記錄的界限。如果字元串常數作為立即模式 (immediate mode) 指令的運算元時，則字元串後面的引號可以省略，因為此種字元串其長度僅可為一個字元。一

個字元串中各字元的數值相當於此字元的 ASCII 值加上 MSB。MSB 為 0 或 1 是由假指令 MSB 所定。如果字元串出現在 DCI 假指令的運算元這一欄，則各字元的 MSB 皆為 0，但是最後一個字元的 MSB 為 1。一般而言，6502 組合語言譯碼程式不允許使用單字元保留字 (one character reserved word) A, X, Y, P 和 S 做為標記。但是本系統的組合語言譯碼程式允許這些保留字做為標記，但不允許在運算元這一欄有保留字 A 做為標記。雖然如此，使用者也應避免使用這些保留字做為標記，以使程式與標準的 6502 語法具有共通性 (compactibility)。

算術式子

本系統的組合語言譯碼程式允許在算術式子中使用的算術運算子有四種：+，-，* 和 /，分別表示加減乘除。算術式子是由運算元和上述的算術運算子構成的，其中運算元為標記或常數。算術式子的計算是由左至右。在算術式子的開頭和結束都不可以為算術運算子。算術式子的語法型式如下：

Expression := Spnd [AOP Spnd [AOP Spnd]
.....]

其中 Spnd 表示運算元，可為標記或常數，AOP 表示算術運算子。如果組合語言譯碼程式遇到不合上面語法的算術式子時，在 pass two，就會顯示出 BAD EXPRESSION 的字幕。在算術式子中不可有空格，否則組合語言譯碼程式將視空格後面的部分為註釋而不予以處理。

進行算術式子計算時，組合語言譯碼程式並沒有檢查結果是否溢位 (overflow)，它僅保留 16 位元的計算結果。

註釋欄——此欄是用來解釋指令的功能和作用，以及說明程式的

動作。註釋這欄可以視情況而予以取捨。由於註釋僅是用來使程式清晰容易瞭解，因此組合語言譯碼程式在譯碼組合時將註釋欄忽略，僅在程式列表時，將註釋顯示出來。使用在註釋欄的字元並沒有限制，這一欄與其他欄是用一個空格和分號“；”來分開。星號*和分號出現在邏輯記錄的第一個字元位置時，表示此邏輯記錄純粹為註釋。

3-5組合語言譯碼程式的假指令

假指令 (directive 或 pseudo op) 是用來指示組合語言譯碼程式進行譯碼組合時的動作執行，以及用來定義標記。假指令的使用和一般的運算碼使用方法相同。某些假指令需要標記，因為這些假指令是用來設定標記的數值或其特殊的意義。任一個假指令的前面可有標記，後面亦可加上註釋。本系統的組合語言譯碼程式擁有的假指令如下：

ORG —— 此假指令的型式為

ORG expression

其中 expression 表示算術式子。此一假指令是用來設定經過譯碼組合後輸出目的碼之起始位址，算術式子之數值即為起始位址。如果在算術式子中有標記存在，則這些標記之數值須在此之前指定。在原始程式中有ORG假指令，組合語言譯碼程式才會產生輸出目的檔。如果在原始程式中沒有ORG假指令，組合語言譯碼程式僅將程式列表在輸出裝置，但不產生輸出目的檔。ORG假指令有兩種型式：絕對 (absolute) 和相對的 (relative)。

相對的ORG假指令的算術式子中含有可再定位標記 (relocatable label)，它僅更新 (update) 目前目的檔 (object file) 中的位址。下面列舉幾個例子：

ORG * + 5

ORG SYM + 10

ORG * - 2

這些相對的ORG假指令，使組合語言譯碼程式將此假指令以後的原始程式經過譯碼組合後產生之目的碼放在新的位址。新的位址可為目前位址的前面位址或後面位址。

絕對的ORG假指令的算術式子為絕對算術式子 (absolute expression)。亦即，此算術式子為一常數。絕對ORG假指令是用來設定輸出目的檔的起始位址，輸出目的檔為二進制檔 (Binary File)，可用BLOAD 命令來將此檔讀入記憶體中，或用BRUN 命令來執行此檔中的程式。在譯碼組合的過程中，組合語言譯碼程式每遇到一個絕對ORG假指令，它就產生一個新的輸出目的檔，然後把目前的輸出目的檔予以關閉。新的目的檔名為上一個目的檔之檔名最後一個字元加1。通常在一個譯碼組合的過程中僅能有一個絕對ORG假指令。

OBJ —— 本系統的組合語言譯碼程式的上一版本，擁有OBJ假指令，是用來指定存輸出目的檔的記憶體位址。目前這一個組合語言譯碼程式將輸出目的檔直接寫入磁碟，因此用不到OBJ假指令。但是仍將此假指令保留，以供往後本系統的功能擴充。

EQU —— 此假指令的型式為

label EQU expression

其中 label 為標記，expression 表示算術式子。EQU假指令

是用來指定一數值給予標記。亦即將 `expression` 的結果指定給標記。

MSB —— 此假指令的型式為

`MSB ON`

或

`MSB OFF`

此假指令是用來指定經由組合語言譯碼程式產生的 `ASCII` 字元之 `MSB` 為 0 或 1。但是，`DCI` 假指令的運算元這欄中的字元串之 `MSB` 值並不受 `MSB` 假指令影響。在組合語言程式中，使用此命令的次數並沒有限定。因此，使用者可依其需要的情況，來使用此命令。組合語言譯碼程式機定 (default) `MSB ON`。因為，在正常的顯示時，Apple II 接受的 `ASCII` 字元之 `MSB` 為 1。

DSECT —— 此假指令的型式為

`DSECT`

此假指令是用來界定某一區域的記憶體，此區域的記憶體可做為數據表 (data table) 或命令控制區段 (command control block)。但此假指令實際上並沒有產生任何輸出目的碼。`DSECT` 命令是用來標示出程式中某一區段的敘述 (statement) 之開始，這些敘述是用來定義標記的值。這些標記的值為某些區段記憶體之位址。`DSECT` 命令最常用來定義資料項 (data items) 的標記，及指標 (pointer) 等等。這些經常為 6502 零頁區的記憶體。

`DSECT` 命令會使 `ORG` 設定為位址 0，亦使目的碼輸出暫時停止。`DSECT` 所標定的區域內可有任何的組合語言譯碼程式假指

令，包括 `ORG` 假指令。`DSECT` 是由 `Dummy SECTION` 而來的，這是因為 `DSECT` 所標定的區域中之假指令並沒有產生輸出。如果在此標定的區域內包含有 `ORG` 假指令，則此 `ORG` 假指令是用來控制指定給出現在此區域之標記的位址，此 `ORG` 並不是一般用來改變目前位址或程式計數器之值的 `ORG` 命令。組合語言譯碼程式遇到 `DSECT` 假指令時，會將目前程式計數器之值儲存起來。組合語言譯碼程式不允許多重 (nest) 的 `DSECT`，亦即，在 `DSECT` 標定的區域內含有其他的 `DSECT` 假指令。

DEND —— 此假指令的型式為

`DEND`

此假指令是用來標示 `DSECT` 所標示區段的結束，將儲存起來的程式計數器之值重新存回程式計數器，並且使組合語言譯碼程式繼續產生目的碼輸出。

如果在原始程式中只有 `DSECT`，而沒有 `DEND` 假指令來標示結束，則 `DSECT` 假指令以下的程式將不會有目的碼產生，但是在譯碼組合之列表顯示時，這部分仍會顯示出來。輸出檔中有部分的目的碼離奇地消失之原因，就是因為遺漏了 `DEND` 假指令。

REL —— 此假指令的型式為

`REL`

此假指令是用來指示組合語言譯碼程式建立一個 `RLD` (Relocation Dictionary)，以提供重新定址載入程式 (relocating loader program) 使用。`RLD` 僅在譯碼組合程式進行的時候產生。如果在原始程式中有 `REL` 假指令時，`RLD` 將被寫入輸出目的檔，此目的檔在 `DOS Catalog` 中，以字元 `R` 標示此檔的

型式。可再定位檔 (relocatable file) 的格式在附錄 E 有敘述。RLOAD 程式的說明請參閱附錄 C。DOS 的 BLOAD 和 BRUN 命令不可使用於可再定位檔。在原始程式中，REL 假指令必須放在相當前面。亦即，在使用 REL 假指令之前，不可有任何標記出現。

EXTRN——此假指令的型式為

EXTRN label

此假指令是用來通知組合語言譯碼程式，label 為外界定義 (external define) 的標記，而不是在本程式中定義的。組合語言譯碼程式視這種標記之值的長度為 two-byte，而不是零頁標記 (Zero-page Symbol)。如果一個指令之運算元部分有外界定義的標記，組合語言譯碼程式在譯碼組合時，會將此指令的地址部分填零。在原始程式中，如果有使用 REL 假指令，並且有外界定義的標記，組合語言譯碼程式會在 RL D 的後面加上 E S D (External Symbol Dictionary)。連結載入程式 (Linking Loader Program) 將會使用 E S D 來解決外界定義標記的問題。EXTRN 假指令不可使用於 DSECT 所標定的區域內。

ENTRY——此假指令的型式為

ENTRY label

此假指令與 EXTRN 假指令是互補的。ENTRY 假指令是用來定義程式中可能出現於其他程式中指令的運算元這一欄之標記。一個程式中可用許多個 ENTRY 假指令，以定義程式的各個進入點 (entry point)。這些標記會在 E S D 中標示出示。這些標記跟著其他必需的資料一起存在 E S D 中。

在 ENTRY 假指令後面的 label 就是定義為進入點的標記。此標

記的值可以在程式中任何位置定義，此標記亦可為 ENTRY 假指令標記欄中的標記。如果是這樣的話，此時程式計數器的值，亦即目前位址，為一個進入點。ENTRY 假指令不可使用於 DSECT 假指令所標定的區域中。

如果程式中沒有使用 REL 假指令，ENTRY 和 EXTRN 假指令仍然可以使用於程式中，甚至沒有連結載入程式。當產生自身修改 (Self modifying) 碼時，使用這些假指令可提供一個能夠在程式執行時才定義標記的方法。ENTRY 可使一個未定義的標記一直保持未定義，而在譯碼組合時仍不產生錯誤。

CHN——此假指令的型式為

CHN Sourcefilename [, slot expression [, drive expression]]

CHN 假指令是用來將一個相當大的原始程式之各部分連接起來。Slot 和 drive expression 可依使用者需要加以取捨。原始檔名 Sourcefile name 不可省略。CHN 必須為原始程式中最後一個指令，因為，在 CHN 後面所有的指令均會被組合語言譯碼程式忽略。

3-6 控制列表的假指令

本節中的假指令是用來控制組合語言譯碼程式列表的動作。使用者可利用這些假指令來設定輸出表列 (output listing) 的型式和格式，以使輸出表列更具可讀性。

PAGE——此假指令的型式為

PAGE

當組合語言譯碼程式在譯碼組合時，如遇到 PAGE 假指令，它就傳送一個 FF (Form - Feed, 其 ASCII 值為 \$0C) 的 ASCII 字元至輸出裝置，輸出裝置便執行跳頁的動作。組合語言譯碼程式亦同時傳送一行空格至螢光幕。組合語言譯碼程式在列表時，並沒有將 PAGE 假指令顯示出來，這一行的行號亦沒有顯示出來。因此，在表列中，如果有行號沒有顯示出來，則表示此行爲 PAGE 假指令。

LST —— 此假指令之型式爲

LST ON

或

LST OFF

使用者可用 LST 假指令來指示組合語言譯碼程式在列表時，只顯示部分的原始程式，或整個原始程式皆不顯示出來。用 LST 假指令來停止原始程式的列表，可使譯碼組合的速度加快。當列表的工作是在列表機 (priter) 上執行時，一旦使用 LST 假指令使列表工作停止，則譯碼組合的速度將更顯著地加快。

REP —— 此假指令之型式爲

REP expression

其中 expression 爲算術式子。組合語言譯碼程式在譯碼組合時，如遇到 REP 假指令，它就在表列中原始程式的部分，顯示一連串相同的字元，從一行的第一個字元之位置開始，字元的數目由 expression 的數值決定。這個假指令通常是用來副程式的開頭產生一行星號，以標出程式的名稱和註解。這個假指令以及算術式子所佔的空間遠比一行星號所佔的空間小。因此，REP 假指令可以減少原始程式在文字檔中所佔的位置。REP 假指令

所顯示的字元機定 (default) 爲星號 “*”。CHR 假指令可用來改變 REP 所顯示的字元。算術式子的有效數值爲 8 個位元，亦即從十進制的 0 ~ 255。如果算術式子計算出來的結果超過 8 個位元，則僅保持右邊的 8 個位元有效。

CHR —— 此假指令之型式爲

CHR ?

CHR 假指令是用來改變 REP 假指令顯示的字元。? 表示程式設計者希望 REP 假指令顯示的字元。

SKP —— 此假指令之型式爲

SKP expression

組合語言譯碼程式在譯碼組合時，一旦遇到 SKP 假指令，它就會在輸出表列加上數行空白，加入空白行的數目由算術式子的結果決定。如果列表的工作是在列表機或其他輸出裝置上執行時，譯碼組合語言就會將回機字元 (Carriage return Character) 之 ASCII 傳送至輸出裝置，以通知其執行跳行的動作。如果輸出裝置須要 LF (Line Feed) 字元來控制跳行的動作，則此裝置在接收到回機字元的 ASCII 時，它必須能夠提供本身 LF 字元，以進行跳行的動作。

SBTL —— 此假指令之型式爲

SBTL Dstring

此假指令是用來列表程式時，一頁的開頭加上一行標題。這一標題由 Dstring 所定義。Dstring 爲字元串，字元串的前後各有一單引號。如 'MYPROGRAM'。

程式設計者可以用 SBTL 假指令，使輸出表列中特殊的部分容易辨識。顯示出來的標題後面跟著組合語言譯碼程式的 ID Stamp

，這是一個日期和六個數字的資料。ID Stamp 存於檔名為 ASM IDSTAMP 的二進制檔。起動本系統時，ID Stamp 會顯示在螢光幕上。ID Stamp 對於系統並無多大意義，亦即，有沒有 ASMIDSTAMP 的二進制檔均不影響本系統的執行，因此，此檔可依使用者的意願加以取捨。

3-7 定義資料的假指令

本節中的假指令使用於組合語言程式中，用來預置 (allocate) 或設定資料區。這些假指令可用來建立位址表 (address table) 和一些在組合語言中常用的資料結構。

ASC —— 此假指令的型式為

ASC Dstring

此假指令置定一連串的文字資料於輸出目的檔。亦即，將這一連串字元的 ASCII 值存於輸出目的檔。Dstring 是用來指定這一連串字元。如果 ASC 假指令前面有標記，組合語言譯碼程式就會將目前程式計數器之值指定給此標記。這個數值就是 ASC 假指令所設定的字元串中第一個字元在記憶體中的位址。MSB 假指令可用來控制這些字元 ASCII 值之 MSB 為 0 或 1。關於 Dstring 參數，請參閱 2—4

DCI —— 此假指令之型式為

DCI Dstring

此假指令的功能和 ASC 假指令的功能類似。但是，使用 DCI 假指令所設定的字元串 ASCII 值之 MSB 為 0 或 1，無法由 MSB 假指令控制。DCI 假指令所設定的字元串中，各字元的 ASCII 值之 M

SB 為 0，但是最後一個字元的 ASCII 之 MSB 為 1。

DFB —— 此假指令之型式為

DFB erpr[,expr……]

其中 expr 表示算術式子。此假指令是用來設定數據資料。組合語言譯碼程式在譯碼組合時，如遇到 DFB 假指令，它就會將算術式子的結果計算出來。如果 DFB 假指令前面有標記，組合語言譯碼程式就會將目前程式計數器之值定給此標記。此值即為 DFB 假指令後面第一個算術式子結果存於記憶體之位址。算術式子之結果有效的最大值為 255。此假指令後面的算術式子如有許多個，則每個算術式子之間用逗點分開，而且僅能使用逗點來分開。一個 DFB 假指令後面的算術式子的個數，最好不超過 5 至 10 個。如果許多個算術式子，可以使用多個 DFB 假指令來定義。

DW —— 此命令的型式為

DW expression

此假指令是用來設定 16 位元的數值。此 16 位元數值中，低的 8 個位元先存，然後再存高的 8 個位元。亦即，儲存的時候，低的 8 個位元在前，高的 8 個位元在後。因此，這個 16 位元的數值就可當作 6502 組合語言間接定此模式 (indirect addressing mode) 的指標 (pointer)。expression 的結果為一個 16 位元的數值。如果在 DW 假指令前面有標記，組合語言譯碼程式就會將目前程式計數器之值指定給此指標，此數值就是 DW 假指令所定義的 16 位元數值中低的 8 個位元存在記憶體中的位址。

DDB —— 此假指令的型式為

DDB expression

此假指令的功能與 DW 假指令的功能相類似。但是 DDB 假指令所設定的 16 位元數值中，高的 8 個位元先存，然後再存低的 8 個位元。

DS ——此假指令的型式為

DS expression

此假指令是用來預置一塊記憶區，但是沒有設定任何資料存於此區域。此區域的大小由 expression 的結果來決定。其結果為一個 16 位元的數值。expression 中不可含有尚未設定數值的標記。由 DS 假指令所預置的記憶區包括在輸出目的檔內。如果此區域很大，則輸出目的檔也相對地變成相當大。通常 expression 為一個數值相當小的常數，以用來預置一塊小的數據資料儲存區。因為，大的緩衝區和工作區 (Work area) 不應包含於輸出目的檔。

如果 DS 假指令的前面有標記，組合語言譯碼程式會將目前的程式計數器之值定給此標記。這一個數值就是 DS 假指令預置記憶區的位址。

3-8 有條件性譯碼組合之假指令

本系統的組合語言譯碼程式提供了有條件的譯碼組合 (Conditional Assembly) 之功能。此功能讓程式設計者有條件性地選擇原始程式中某部分是否要經過譯碼組合。使用於有條件性譯碼組合之假指令有三個：DO, ELSE 和 FIN。DO 和 FIN 這兩個假指令必須成對使用，分別表示原始程式中，有條件性譯碼組合部分的開始和結束。ELSE 僅能使用於這一個有條件性譯

碼組合部分的裏面。有條件性譯碼組合的控制由 DO 假指令來決定。ELSE 假指令是用來倒轉 (invert) 由 DO 假指令所決定的狀態。這些假指令前面可以加標記，後面也可以加上註解。

DO ——此假指令的型式為

DO expression

DO 此假指令具有兩個功能：其中之一為，通知組合語言譯碼程式有條件性譯碼組合部分的開始。另一為，計算出 expression 的結果，以決定此部分是否須要經過譯碼組合。

DO 假指令後面的 expression 是在 pass one 時計算結果，因此，expression 中不可含有未指定數值的標記。如果 expression 的結果不為零，組合語言譯碼程式將繼續譯碼組合。如果 expression 的結果等於零，組合語言譯碼程式將停止譯碼組合。DO 假指令以下的原始程式，直到 FIN 假指令以後，才再開始譯碼組合。

ELSE ——此假指令的型式為

ELSE .

此假指令僅能用於由 DO 假指令和 FIN 假指令所界定的程式裏面。ELSE 假指令是用來倒轉由 DO 假指令決定的狀態。如果 DO 假指令使組合語言譯碼程式停止譯碼組合，到了 ELSE 假指令以後，組合語言譯碼程式將繼續譯碼組合。如果 DO 假指令沒有使組合語言譯碼程式停止譯碼組合，到了 ELSE 假指令以後，組合語言譯碼程式將停止譯碼組合。以上兩種狀態中任一種狀態，直到 FIN 假指令後，組合語言譯碼程式便恢復正常的譯碼組合程序。

FIN ——此假指令的型式為

FIN

此假指令是用來表示有條件性譯碼組合的部分之結束。在 FIN 假指令後，組合語言譯碼程式便回到正常的譯碼程序。在原始程式中，不可僅有 FIN 或 DO 假指令，這兩個假指令必須成對地使用。DO, ELSE, FIN 三個假指令之第一個字母順序和各假指令在原始程式中出現的順序是一樣的。其中 ELSE 假指令可依情況取捨。

3-9 符號表的顯示

組合語言譯碼程式在 Listing 時，會顯示出 2 個符號表 (symbol table)，第一個是以各符號名稱之第一個字元順序排列的符號表，第二個是各符號的位址值順序排列的符號表。如果使用 CTRL-C 來中止譯碼組合程序；或是，使用 LST OFF 或 CTRL-N 來停止組合語言譯碼程式 Listing 的工作，則符號表亦不會顯示出來。在程式的開頭使用 LST OFF 假指令，組合語言譯碼程式將不做 Listing 的工作，如果在此程式的結束使用 LST ON 假指令，則僅有符號表顯示出來。

符號表的顯示會自動調整其寬度，以配合 Apple 每行 40 字的顯示幕或列表機。符號標記 (symbol label) 的各種特性可由其位址值顯示出來的格式及顯示在位址值前面的標幟 (flag) 字元表示出來。下面這個例子說明了符號表顯示出來的意義。

SYMBOL TABLE	SORTED BY SYMBOL	27-JUL-79	#00090	PG 11
6D ADPTR	6F ADTBLND	2D3E ALPHAS	24 CH	
N2C00 SYMDUMP	0072 SYTYPE	2D6E SWAP	?2DF4 SWIPEIT	
2E5F TABLND	?2E84 TESTLBL	? 0A TXTBEG	77 VAL	
*2E99 XXXXX	X 76 YYYAV			

如果標記符號之位址值前面有兩個空格，則表示此符號標記為零頁標記 (Zero-page label)。在位址值之前的 ? 字元表示該符號標記已定義，但沒有在程式中其他指令的運算元欄出現。* 字元表示，此符號標記曾出現在程式中其他指令的運算元欄，但是其位址值尚未定義。X 表示此符號標記為 EXTRN label。N 表示此符號標記為 ENTRY label。

ADDRESSING MODE SUMMARY

Note that all required syntax may be preceded by an optional label.

Addressing Mode	Required Syntax
Implied (no address)	opc
Accumulator	opc A
Immediate	opc #expression
Low 8 bits of address	opc #>expression
High 8 bits of address	opc #<expression
Zero page	opc zpg-expression
Indexed X	opc zpg-expression,X
Indexed Y	opc zpg-expression,Y
Absolute	opc abs-expression
Indexed X	opc abs-expression,X
Indexed Y	opc abs-expression,Y
Indirect,Indexed X	opc (zpg-expression,X)
Indirect,Indexed Y	opc (zpg-expression),Y
Absolute Indirect	JMP (zpg-expression)

ASSEMBLER DIRECTIVE SUMMARY

ORG	ORiGin
OBJ	OBJect
EQU	EQUate
MSB	MSB
DSECT	DSECTiOn
DEND	DsectEND
REL	RELocatable
EXTRN	EXTeRNaL
ENTRY	ENTRY
DO	DO if
ELSE	ELSE
FIN	FINish
PAGE	PAGE Eject
LST	LISting
REP	REPeat
CHR	CHaRacter
SKP	SKiP
SBTL	SuBTitLe
ASC	ASCii
DCI	DCI
DFB	DeFine Byte
DW	DeFine Word
DDB	DeFine Double Byte
DS	DeFine Storage

OPERATION CODE SUMMARY

ADC	A + M + C → A	JMP	Jump to New Location
AND	A and M → A	JSR	Jump to Subroutine
ASL	C ← [7..0] ← 0	LDA	M → A
BCC	Branch on C = 0	LDX	M → X
BCS	Branch on C = 1	LDY	M → Y
BEQ	Branch on Z = 1	LSR	0 → [7..0] → C
BIT	A and M, M7 → N, M6 → V	NOP	No Operation (PC=PC+1)
BGE	Branch on C = 1	PHA	A → Ms S-1 → S
BLT	Branch on C = 0	PHP	P → Ms S-1 → S
BMI	Branch on N = 1	PLA	S+1 → S Ms → A
BNE	Branch on Z = 0	PLP	S+1 → S Ms → P
BPL	Branch on N = 0		
BRK	Force Break		
BVC	Branch on V = 0		
BVS	Branch on V = 1		
CLC	0 → C	ROL	← [7..0] ← C
CLD	0 → D	ROR	→ C → [7..0] →
CLI	0 → I	RTI	Return from Interrupt
CLV	0 → V	RTS	Return from Subroutine
CMP	A - M → P	SBC	A - M - C → A
CPX	X - M → P	SEC	1 → C
CPY	Y - M → P	SED	1 → D
		SEI	1 → I
		STA	A → M
		STX	X → M
		STY	Y → M
DEC	M - 1 → M	TAX	A → X
DEX	X - 1 → X	TAY	A → Y
DEY	Y - 1 → Y	TSX	S → X
EOR	A xor M → A	TXA	X → A
INC	M + 1 → M	TXS	X → S
INX	X + 1 → X	TYA	Y → A
INY	Y + 1 → Y		

附錄 A

編輯程式/組合語言譯碼 程式使用的記憶體及用途

編輯程式是由兩種語言混合寫成的：6502 組合語言及虛擬的 16 位元機器 (pseudo 16-bit machine) 之語言 Sweet 16。組合語言譯碼程式本身並沒有產生 Sweet 16 的機器碼。此 Sweet 16 的釋譯程式 (interpreter) 包含於本系統的 Editor module (在磁片的 Catalog 中標示為 EDITOR) 中。它是由 Apple II 的 Integer Basic 修改而得的，以使其速度能加快。命令釋義程式 (Command Interpreter) 是由 6502 組合語言寫成的。其主要功能是分析命令，並將參數置於參數表列 (parameter list)，和呼叫編輯程式。它分析 ASM 命令，將參數置於參數列，然後呼叫組合語言譯碼程式。

下面列出各記憶區在命令釋義程式，編輯程式，組合語言譯碼程式中的用途。

記 憶 位 址	在命令釋義程式及 編輯程式中的用途	在組合語言譯碼程 式中的用途
\$0000	SWEET 16 的	在譯碼組合程序時

}	暫存器，變數，編	的指標及變數
\$00FF	編輯標幟	
\$0100		
}	6502 硬體堆疊 (	6502 硬體堆疊 (
\$01FF	stack)	stack)
\$0200	編輯程式的輸入緩	
}	衝區，以供命令及	
\$02FF	本文 (text) 輸入	不 用
	時使用。	
\$0300	編輯程式之數字的	輸入參數及 ASM
}	輸入堆疊，固定標	IDSTAMP 檔
\$03CF	幟，ASMID	
	STAMP 檔	
\$03DO		DOS 界面 JUM
}	不 用	PS 及 SUBRS
\$03FF		
\$0400	編輯程式的螢幕記	組合語言譯碼程式
}	憶區	的螢幕記憶區
\$07FF		
\$0800	編輯程式的文字串	組合語言譯碼程式
}	參數堆疊	的參數表列及輸出
\$08FF		螢幕記憶區
\$0900	編輯程式命令堆疊	組合語言譯碼程式
}		的輸出螢幕記憶區
\$09FF	保存區	
\$0A00	編輯程式 LIST	組合語言譯碼程式
}		的輸出螢幕記憶區
\$0AFF	命令保存區	

\$OB00		組合語言譯碼程式
}	不 用	
\$OBFF		的輸出螢幕記憶區
\$OC00	EDASM Module	EDASM Module
}		
\$11FF	(6502)	(6502)
\$1200	EDASM Module	ASSM Module
}		
\$1D80	(6502 & Sweet	(6502)
}	16)	:
\$2000	SWEET 16 ma-	:
}	chine(6502 碼)	:
\$3180	編輯緩衝區下限位址	
}	址 (LOMEM)	
		標記表格 (Sym
		bol table) 的開
		始 (first symbol)
		(last Symbol)
		(last RLD en-
		try)
		:
		(first RLD en-
		try)
HIMEM=	編輯緩衝區上限位	
	址 (HIMEM)	
\$9600		
}	DOS 48k	DOS 48k
\$BFFF		

附錄 B

編輯程式顯示的DOS錯誤

命令釋義程式及編輯程式DOS之間界面的方法和BASIC 程式與DOS之間界面的方法一樣。當系統啟動後，系統改變DOS以使DOS視命令釋義程式為其主機語言 (host language)。這使得系統能夠在DOS錯誤發時，掌握DOS錯誤狀況，並將錯誤訊息傳至編輯程式。經由命令×3DOG重新啟動 (reinitalizing) DOS時，系統會重新進入命令釋義程式。同樣地，如果你的Apple II有Autostart ROM，按RESET鍵，會使系統經由DOS進入命令釋義程式。

所有的錯誤訊息均由DOS將其顯示在Apple螢光幕上。當使用LOAD命令來讀取磁碟中的文字檔，如果DOS讀取文字檔完畢，它會在螢幕上顯示“END OF DATA”

下面的表並沒將所有的錯誤均列出來，但是此表將最常見的錯誤列出來。

DOS 錯誤訊息	促使錯誤發生的狀況
WRITE PROTECTED	使用SAVE命令，欲將資料存入有Write-protect的磁片，或是，使

	用 ASM 命令於有 Write-protect 的系統磁片。這使得 ASMI DST-AMP 檔無法更新其資料。
END OF DATA	使用 LOAD 命令讀取文字檔完畢。
FILE NOT FOUND	如果在系統磁片中沒有 ASMI DS-TAMP 檔，使用 ASM 命令時會立即發生此一錯誤。實際上，系統即使沒有 ASMI DSTAMP 檔亦能執行。因此，在這種情況下，此訊息可以忽略不管。如果你欲使用直接 DOS 命令來處理文字檔，例如 LOCK 和 UNLOCK，而磁片並沒有此文字檔時，亦會發生此錯誤訊息。
I/O ERROR	此錯誤一般不常發生。使用 SAVE, LOAD, CATalog 命令時，有可能發生此種錯誤。使用 SAVE 命令時，發生此種錯誤，則寫入的檔之資料可能有錯誤。最好換另外一張磁片，然後再使用 SAVE 命令。如果使用 LOAD 命令時，發生此錯誤，則文字檔只有其中一部分被讀取，且此檔可能已損失。如果使用 CATalog 命令時，發生此錯誤，你

DISK FULL	的磁片的所有資料均已損失。 使用 SAVE 命令最常見此訊息：如果發生此訊息，另外再拿一張磁片來重新儲存上一個文字檔。
FILE LOCKED	如果你習慣使用 LOCK 命令來鎖住你的檔，而又使用 SAVE 命令欲將資料存入已鎖住的檔，此時就會發生 FILE LOCKED 的錯誤。使用其他的檔名或使用 UNLOCK 命令便可解決此問題。除了目前在編輯緩衝區中的欲編輯檔以外，用 LOCK 命令將其他的檔鎖住，可避免錯用檔名，而將其他的檔改掉使用語法錯誤的直接 DOS 命令，會發生此錯誤訊息。
SYNTAX ERROR	如果 MAXFILES 設定後使用此系統，且你用 DOS 命令欲同時處理兩個檔時，此錯誤便發生。
NO BUFFERS AVAILABLE	如果你使用 LOAD 命令來讀取非文字檔之其他型式的檔，或是使用 SAVE 命令來儲存文字檔，而此檔之檔名與其他型式的檔之檔名相同時，此錯誤便發生。
FILE TYPE MISMATCH	使用直接 DOS 命令與系統發生衝突時，便發生此錯誤。
PROGRAM TOO LARGE	

附錄 C

再定位載入程式(RELOCATING LOADER)

組合語言譯碼程式可用來產生可再定位目的檔 (relocatable object file)。可再定位目的檔經由再定位載入程式載入時，可以重新指定此檔之位址。再定位載入程式並不是連結載入程式 (linking loader)。在本系統磁片中的兩個程式，RBOOT 和 RLOAD，構成了 ROM 或 Language Card Applesoft II 的再定位載入程式。

這兩個程式提供了從 Applesoft 程式載入 (load) 1 個以上的組合語言程式的方法，而且，每當一個程式被載入之後，便可得到該程式的載入位址。程式載入的開始位址就在編輯緩衝區上限位址 HI MEM 之下，因此，新的 HI MEM 之值等於原先之值減去被載入程式之長度加 1 (length+1)。此種簡易的記憶體管理方法要求在使用再定位載入程式之前，不可有文字串預置位址 (allocate)，因為文字串預置的位址可能就是被載入程式在記憶體中的位址。

除了上述使用再定位載入程式之前，不可使用字元串變數的

限制之外，程式設計者不可在使用 RBOOT 和 RLOAD 將程式載入時使用 DIMENSION 或預置新的數字或字元串變數。因為 RBOOT 和 RLOAD 佔用有關 Applesoft 變數表的記憶體區。

RBOOT 程式，類似 Apple Soft 的 CHAIN 程式，從磁片被讀至記憶體，從 \$208 至 \$3CF，可經由 CALL 520 來執行此程式。通常的 DOS BLOAD 命令應由 CHR\$(4) 來代替，記住，不可有字元串！RBOOT 程式是用來載入及再定位 RLOAD 於記憶中，且佔用 Apple Soft 變數表結束的上面之空間至少有 1 頁。並且設定 Apple Soft USR 函數跳躍位址為 RLOAD 的進入點 (entry point)。

程式設計者可經由 USR(0) 來使用 RLOAD 程式將可再定位目的檔載入。RLOAD 會傳回載入點的位址，或是在載入的過程中所遭遇到的問題。

RBOOT 並不需要任何參數，它假設 RLOAD 程式所在的磁片與 RBOOT 所在的磁片相同。RLOAD 程式需要三個參數，這三個參數須用引號將此三個參數標示出來，且用逗號與 USR 函數隔開。下面的例子說明了 RBOOT 和 RLOAD 的用法：

```
10 ADRS = 0 : REM PRE ALLOCATE VARIABLE TABLE
20 PRINT CHR$(4) ; "BLOAD RBOOT" : CALL 520
30 ADRS = USR(0), "MYMODULE, S6, D1"
```

其中 S6 及 D1 參數可視情況取捨，但檔名 MYMODULE 不可省略，且此檔的型式必須為可再定位檔，不可為二進制檔。如果沒有指定檔名，則會顯示 'FILE NOT FOUND' 而不是 'SY-

NTAX ERROR'。

USR函數所得到的結果之數值為有正負符號之實數。經由CALL命令使用此數值可以進入被載入的程式。將許多個JUMP指令放在程式的開頭，可以造成多進入點(entry point)的效果。如CALL ADRS, ADRS+3, AORS+6, ADRS+9等。這允許被載入程式之內容增加或縮減而不會破壞BASIC與被載入程式間的介面。

程式設計者可以載入數個程式，只要所有被載入程式和RLOAD所佔記憶空間總數不超過可供利用的記憶空間。RLOAD大約佔1.5K的記憶空間。RLOAD要求至少有一個檔緩衝區(file buffer)可供其利用，否則會出現NO BUFFERS AVAILABLE的錯誤。RLOAD並沒有使HIMEM恢復為原來的值，因此，在程式結束時，程式設計者應將HIMEM恢復為原來的值。

使用RLOAD來測試程式時應注意，每次執行時，RLOAD載入程式於記憶體中的位址都不一樣，所以連續地測試程式的執行將會使記憶體中存有許多個相同的程式，而且使得HIMEM持續地減小。先用FP命令，然後再從磁碟取出程式來執行便可解決此問題。

附錄 D 組合語言 譯碼程式的 DOS 錯誤碼 (DOS ERROR CODES) “OOPS”

在整個譯碼組合程序中，組合語言譯碼程式一直使用著DOS。在整個過程中，DOS可能無法達成組合語言譯碼程式某些的要求。這些錯誤並非是一般性的，在譯碼組合程序中，這些錯誤具有相當嚴重的破壞性。當發生這些錯誤時，組合語言譯碼程式便停止譯碼組合，並顯示出

OOPS! DOS ERROR! COOE=×

且喇叭會連續發出三次嗶聲。

當錯誤發生時，組合語言譯碼程式將停止譯碼組合程序，並嘗試關閉已打開的檔。如果無法關閉已打開的檔，則會有其他的錯誤發生，此時，組合語言譯碼程式會放棄關閉已打開的檔，然後系統回到命令模式。

OOPS 錯誤碼

錯誤碼的意義及原因

04

WRITE PROTECTED DISKETTE

組合語言譯碼程式欲將輸出目的檔存入之磁片為Write-protect 磁片。此錯誤在pass2時才會發

生。

06 FILE NOT FOUND

A S M命令所指定的檔名之檔並不在磁片中，或
C H N命令所指定的檔名之檔並不在磁片中，此
時便發生此錯誤。

08 I/O ERROR

此錯誤通常是讀取資料時的錯誤或由於不良磁片
所造成寫入資料時的錯誤。如果在寫入資料時發
生錯誤，則輸出目的檔亦會建立於 Catalog 中，
且 Catalog 顯示此檔僅佔 1 個 Sector 的空間。

09 DISK FULL ERROR

如果磁片上可用的空間不夠輸出目的檔的使用，
則會發生此錯誤。

0C(12) NO BUFFERS AVAILABLE

此種錯誤不太可能發生，除非進入系統時，MA-
XFILES設定為 1。正常的譯碼組合程序要求至
少有二個緩衝區可供使用。如果 A S M發生錯誤
，組合語言譯碼程式停止譯碼組合程序，且無法
關掉所有已打開的檔。則此錯誤有可能發生。

0A(10) FILE LOCKED

如果目的檔被鎖住，則發生此錯誤。

附錄 E

目的檔的格式

組合語言譯碼程式可產生兩種型式的 D O S 目的檔：二進制
記憶體對映檔 (Binary memory-image file) 和可再定位二進
碼檔 (Relocatable binary code file)。組合語言譯碼程式所
產生的二進制檔與 D O S 所產生二進制檔的格式是相同的。除非
在組合語言程式的開頭有使用 R E L 假指令，否則組合語言譯碼
程式皆產生二進制檔。

在 BASIC / DOS 的情況，可以使用 BLOAD 來讀取或使用
BRUN 來執行二進制檔中的程式。二進制檔的第一個 byte 就必
須是操作碼 (opcode)，如果這個二進制檔是要經由 BRUN 命令
執行的話。並且在檔的開頭不可為資料區。

可再定位二進制檔是 D O S 的擴充：下面的表說明此種型式
的檔之格式。在此表中，符號⇒可讀為“表示”(indicate)

Sector	byte	byte 的內容
1	0	R A M的開始位址，low byte
	1	R A M的開始位址，high byte

	2	RAM image 的長度, low byte
	3	RAM image 的長度, high byte
	4	Code image 的長度, low byte
	5	Code image 的長度, high byte
1 ~ N	6 ~ C1 + 6	二進制碼 image, 其長度是由上面的 byte 4 和 byte 5 表示。
	C1 + 7	RLD (Relocation Dictionary) 的開始 RLD 是由 N 個 4 byte entry 所構成。 N 為變數。
	1	RLD 標幟 byte, 包含下列所述的 4 個標幟位元。
\$ 80 位元		可再定位範圍 (relocatable field) 的大小, Clear $\Rightarrow$ 1 byte, Set $\Rightarrow$ 2 byte
\$ 40 位元		16 位元數值的高位 / 低位 (Upper / Lower) 8 位元, Clear $\Rightarrow$ 低 8 位元, Set $\Rightarrow$ 高 8 位元。
\$ 20 位元		正常 / 倒轉 (Normal / reversed) 2 byte 範圍 Clear $\Rightarrow$ low-hi, Set $\Rightarrow$ hi-low
\$ 10 位元		Field 為 EXTRN 16 位元 reference Clear $\Rightarrow$ not EXTRN Set $\Rightarrow$ EXTRN
\$ 01 位元		“NOT END OF RLD” 標幟位元, RLD 的 entry 一直為 Set, Clear 表示 RLD 結束
	2	Field offset in code, low byte
	3	Field offset in code, high byte

4	含有 Upper 8 位元的 8 位元範圍之 16 位元 數值中的低 8 位元。Zero 如果 RLD byte one 的 \$ 40 位元 Clear。如果 \$ 10 位元為 Set, 則此值為 ESD 符號值。
N * 4 + 1	二進制 00 表示 RLD 的結束。
N * 4 + 2	ESD (External Symbol Dictionary) 之開始。
1 ~ S1	EXTRN / ENTRY 符號名長度 S1 bytes。 除了最後的 byte, 其餘的 byte 之 \$ 80 位 元 Set。
S1 + 1	符號型式標幟 byte, 定義何種型式的符號 ENTRY 或 EXTRN。
\$ 10 位元	Set $\Rightarrow$ EXTRN 型式的符號
\$ 08 位元	Set $\Rightarrow$ ENTRY 型式的符號
S1 + 2	RLD entry 之 EXTRN / ENTRY 符號 值。
S1 + 3	ENTRY 型式符號 offset 的高 byte。
END mark	二進制 Zero byte 表示 ESD entry 的結 束。

附錄 F

標記符號表(Symbol table)的程式

標記符號表是在譯碼組合程式之 pass 1 時產生。標記符號表中的 entry 為可變長度 (Variable length)，以標幟位元 (flag bit) 來標示可變長度字元串的結束。

其基本格式為 (符號名) (標幟 byte) (Lowvalue) (High Value)

- | | |
|---------|--|
| 符號名 | 由 1 至 n 個字元所組成，其所有的字元之 \$80 位元為 1，但最後一個字元之 \$80 位元為 0。 |
| 標幟 byte | 包含用來定義此符號的特性，符號的值，和如何用此符號值來產生指令等的位元。 |
| \$80 位元 | Forward Reference bit Set .
這表示此符號標記出現在運算元欄，但是尚未定義其值。如果符號標記之值已經定義，則此位元為 reset。如果在 pass1 結束時，符號標記之值仍未定義，則在 pass2 時會發生 “ NO SUCH LABEL ” 的錯誤。 |

- | | |
|---------|---|
| \$40 位元 | Unreferenced Symbol bit set
此符號標記之值已定義，但是此符號標記不會在程式中其他指令的運算元欄出現。 |
| \$20 位元 | Relative Symbol bit set
此符號標記為相對的 (relative)，而不是絕對位址 (absolute address)。 |
| \$10 位元 | EXTRN Symbol bit set
此符號標記是經由 EXTRN 假指令定義為 external Symbol。此符號標記於 ESD 中，以避免此標記符號被視為未定義的標記符號，縱然沒有指定此標記符號之值。 |
| \$08 位元 | ENTRY Symbol bit set
此符號標記是經由 ENTRY 假指令定義為程式中的 entry point。 |
| \$04 位元 | MACRO name bit set
此符號標記為 MACRO 檔的檔名。 |
| \$02 位元 | NO Such Label Error bit set
一個符號標記可造成多次的 NO SUCH LABEL 的錯誤。此位元可以防止在 pass2 時，將同一錯誤重覆地顯示出來。 |
| \$01 位元 | Absolute Address bit set
表示符號標記其值為 16 位元之數值。 |

附錄 G

BASIC 程式的編輯

使用編輯程式來建立 BASIC 程式或編校在編輯緩衝區中的 BASIC 程式。DOS 手冊中 “Capturing Programs in a Text File” 有解釋如何從 BASIC 建立 text file。使用 LOAD 命令將文字檔讀入至編輯緩衝區後，此時可使用編輯程式來處理文字檔。Find 命令可以用來尋找行號所指定的行或含有指定變數名的行。Change 命令可以用來將文字檔中所有的指定變數名為另外新的變數名，或改變 GOSUB 的行號。

當使用 Change 命令來改變某一行的行號時應注意，在其他的行中有 reference 到此行號並沒有隨著改變。在 BASIC 程式應盡量避免改變各行的行號。在一行中加入字元，改變字元均可由編輯程式命令完成。

BASIC 程式中每一行在編輯程式顯示時均有二個行號，第一個行號為此行在編輯緩衝區中的行號，第二個行號為 BASIC 行號。不可使用此 BASIC 行號做為編輯程式命令的參數。改變在編輯緩衝區中的順序，但沒有改變 BASIC 行號，並沒有改變

其在 BASIC 的順序。當你完成文字檔編輯工作，在經由 END 命令回到 BASIC 之前，須使用 SAVE 命令將編輯緩衝區中的文字檔存回磁片。回到 BASIC 之後，你必須使用類似下面的 DOS 命令

EXEC filename

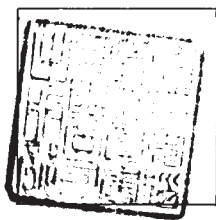
此命令會將 filename 所指定的文字檔從磁片讀至 BASIC，如同從鍵盤輸入一般。

波前新書介紹

- APPLE II 資料管理系統
施威銘 凌雲志 著 定價 220 元
- 蘋果世界中的 CP/M BASIC
江宗綬 編著 定價 150 元
- 專訪蘋果 BASIC 編譯器(2)繪圖篇
楊國雄 編著 定價 220 元
- 蘋果的數學
(1)國中上篇 (2)國中中篇 (3)國中下篇
劉又中 王家昌 黃彥男合著
楊國雄校訂 中冊定價 130 元
上下二冊每本定價 150 元
- 作業系統簡介
高金城 編著 定價 110 元
- APPLE II 磁片保護策略
蘇哲能 編著 定價 180 元
- VISICALC 與企業決策
林鳳翔 編著 定價 120 元
- 青少年學電腦
張振芳 編著 定價 140 元
- 微電腦自動控制基本原理
周淳朴 編著 定價 140 元
- APPLE II PASCAL 入門與實例
廖明顯 編著 定價 140 元
- 小朋友來學電腦
胡恩賜老師 著 定價 80 元
- PFS 個人檔案系統與實例設計
環台電腦編譯 定價 80 元
- BASIC 結構化程式設計
閔國田譯 定價 200 元

- 專訪蘋果 BASIC 編譯器(1)語言系統篇
楊國雄 編著 定價 240 元
- Z-80 微電腦學
林益正 編著 定價 230 元
- 從 TOOL KIT 看 APPLE 系統
彭之瑞 著 定價 210 元
- CP/M 與 PACKAGE 大全
吳國寶 編著 定價 220 元
- CP/M MBASIC 入門與實例
李華敏 彭淇湘 編著 定價 170 元
- APPLE II 高級 BASIC 結構化設計
鐘健堯 著 定價 155 元
- APPLE II 磁片 COPY 大全第 2 冊
唐兆立 游允毅 編著 定價 160 元
- APPLE II 硬體應用專案設計
曾振賢 編著 定價 160 元
- APPLE II DOS 高級技巧
宗世麟等編著 定價 215 元
- APPLE II BASIC 繪圖
吳國寶 著 定價 170 元
- CP/M 作業系統入門
李華敏 著 定價 180 元
- APPLE II PASCAL 實用大全
吳國寶 編著 定價 210 元
- APPLE II BASIC 程式設計入門
胡文美 著 定價 140 元
- APPLE II 組合語言演習
章豪 編譯 定價 130 元
- APPLE II 機械人與無接點程式控制
吳占鰲著 定價 230 元

- APPLE II 組合語言
(1) LISA 活用法
(增訂版·增加大量實例)
李天佑編著 定價 150 元
- (2) TOOL KIT 活用法
徐寶龍編著 定價 70 元
- (3) CP/M 活用法
杜文宗等編著 定價 120 元
- APPLE II 世界程式競賽選輯
林和賢著 定價 150 元
- 蘋果的智慧 黃松榮著 定價 150 元
- APPLE II 硬體介面實驗初步
徐寶龍等編譯 定價 160 元
- APPLE II 組合語言精華
李天佑著 定價 230 元
- 磁碟機·印表機活用法
胡文美著 定價 120 元
- APPLE II 軟硬體改良設計
彭之瑞等編著 定價 220 元
- APPLE II 硬體線路分析
李士彬等著 定價 290 元
- APPLE II 程式師手冊
宗世麟編譯 定價 190 元
- APPLE II 微電腦徹底研究系列
(一)入門篇 (二)進階篇 (三)磁碟系統篇
(四)動畫技巧篇·施威銘著定價 220 元
- APPLE II 磁片 COPY 大全
梁振宇等編著 定價 160 元
- APPLE II 圖形遊戲系統設計
林和賢著 定價 180 元
- 從 APPLE II 到圖形遊戲製作
宗世麟著 定價 180 元
- APPLE II 自動控制實務設計
吳占鰲編著 定價 250 元



著作權、版權所有，翻印必究

法律顧問：邱晁泉律師

APPLE II 組合語言 TOOL KIT活用法

編著者：徐寶龍

發行人：凌雲志

發行所：波前電腦管理圖書有限公司

地址：台北市泰順街2巷30號

電話：3927101～2

郵撥：056141-5 帳戶：劉麗雪

定價：70元

印刷者：名揚印刷公司

著作權執照字號：台內著字第 號

中華民國 74 年 4 月 出版

鄭重推薦本書相關作品

從 TOOL KIT 看 APPLE 系統

彭之瑞、楊國雄編著 定價 210 元

- 本書為您解說 RBOOT RLOAD APA、ANIMATRIX 等程式。
- 告訴您！RBOOT、RLOAD 的原理，R 類檔的由來，以及許多學電腦不可或缺的觀念。
- 讓你知道 APA 可以提供什麼額外的功能，如何自己設計一個 APA。
- 教您使用 HRCG 及 ANIMATRIX，並進而利用 HRCG 與 ANIMATRIX 來編寫有趣的電玩程式。
- 以 HRCG 的原理，設計出一個 LRCG，告訴您如何靈活的控制螢幕上的畫面。

本書特點

- 您是否擁有 TOOL KIT 磁片，那您將更需要擁有本書
- 本書將導引您如何使用 TOOL KIT 中的 Assembler / Editor 程式，使您編寫 6502 組合語言的工作更為便利
- 如果您已看完原文手冊，但仍不會使用 Assembler / Editor，請您試一試本書，您將有意想不到的收穫
- 本書共分三章，結構分明，您並不需花費多少時間，即可熟練地使用 Assembler / Editor 系統

● 波前出版 ●  ● 必屬好書 ●

TOOL KIT

活用法



波前電腦管理圖書有限公司