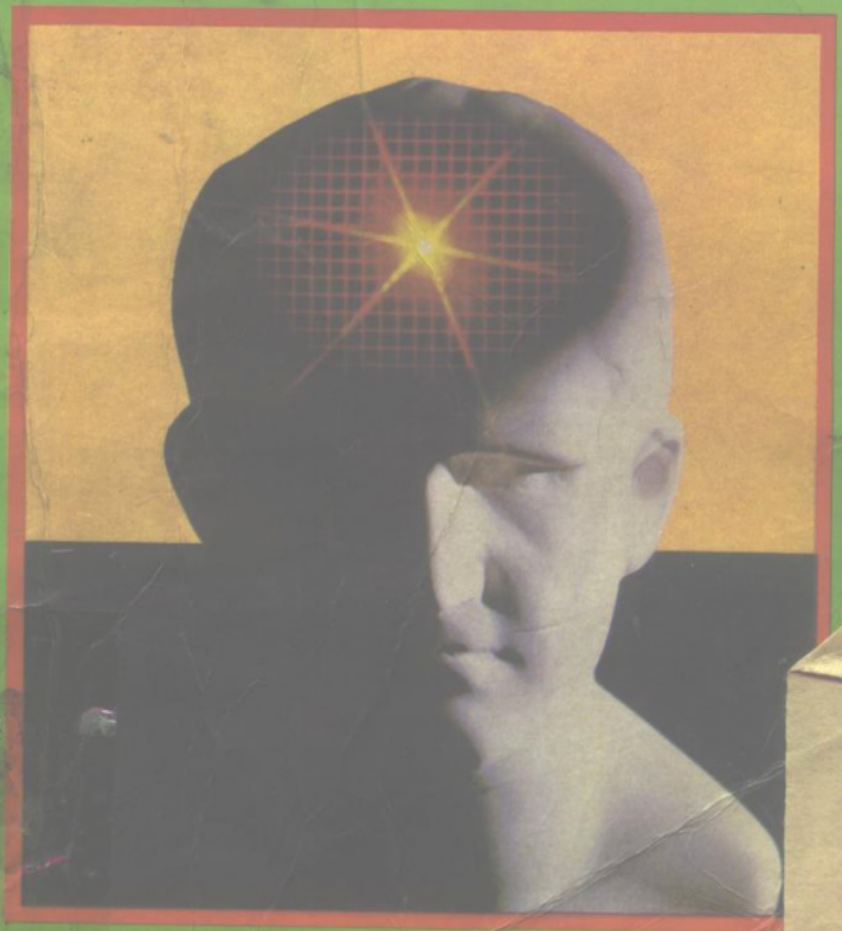
APPLE II

DOS高級技巧

宗世麟・唐兆立
黃松榮 編著



通大出版社

APPLE II DOS高級技巧

APPLE Ⅱ DOS 高級技巧

譯　者：宗世麟・唐兆立・黃松榮
出版者：通 大 出 版 社
地　址：香港洛克道160號三樓
印刷者：和達印刷廠
地　址：香港柴灣工廠大厦11樓H座

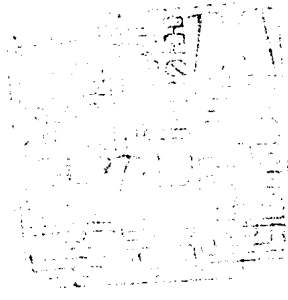
定　價：H.K.$42.00元正

序

對APPLE Ⅱ的愛好者而言，相信這是一本您期待已久的書。它告訴您一些在別的書上找不到的資料，甚至它還可能幫助您解決一些以前無法解決的疑惑。書中討論的範圍很廣，從 DOS 的原理到介面卡的應用均包括在內，足以應付您不同的需要。

本書主要根據“ CALL A.P.P.L.E - ALL ABOUT DOS ”爲藍本重新編寫而成。書中除了將原書中的部份錯誤加以訂正外，並對於整本書的內容做了一番新的安排：將一些不必要的文章刪除，加入較合乎實際需要的體裁，並按文章的主題加以分類重新排定章節，以一個全新的面貌呈現在您的面前。

最後，希望您喜歡這本書！



編者序

1984 年 4 月 13 日

目錄

總論

APPLE II 採用標準磁碟系統

和Apple 主機內 8 個 bit 並行傳送資料的方法不同的，Apple 與磁碟機間的資料傳輸是以一次一個 bit 的方式進行的。此種傳輸方法稱爲 bit serial transmission。這是因爲 Apple 所用的磁碟機中只有一個磁頭，只能一次一 bit 地讀取資料。許多下面將談到的問題都是由於使用這種記錄方式而產生的。但是在更進一步討論那些問題之前，先讓我們來看看一片標準磁碟的整體結構（軟，硬兩方面）。

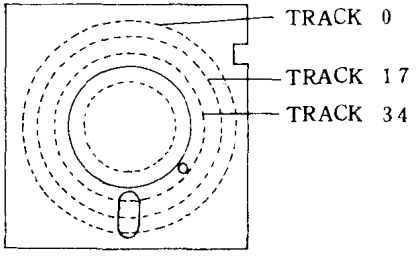
磁碟的硬體結構

在外觀上磁碟是一由彈性的空心圓碟（通常是一種極強韌的膠體）上下塗以磁性物質而構成。這圓碟被放進一紙質的保護封套之內，以防止在取用磁碟時不慎對記憶材料造成傷害。該紙套上開有一長孔，這是讓磁碟機上的磁頭讀寫資料的所在。中央的大圓孔則是讓磁碟機能將磁碟夾住之用。有些細心的讀者必定

注意到紙套上中央部位另有一較小的圓孔，如果你小心的將紙套內的圓碟轉動，你將發現在磁碟上也有小洞在相同的半徑位置上。這些孔洞是用來爲磁碟定位用的。如衆所週知，對於一個完美無瑕的球體你是無法分辨其方向的，除非能在上面作下一些記號，你才能說它現在是向東向西等。這就是那些孔的用途。

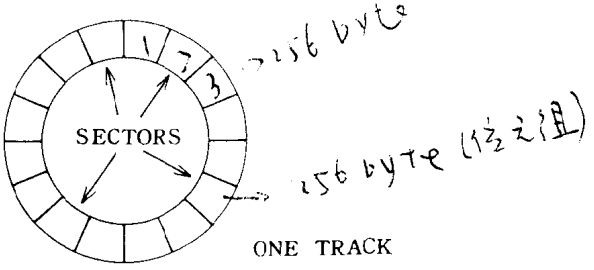
標準DOS系統下的磁碟記憶空間區分

可想見的到，當磁頭置於不同位置時可讀到的資料便會有所不同。這磁頭在某一特定位置上所能讀到的全部資料就稱作一條 track（磁軌）。在 APPLE 的 DOS 3.3 磁碟系統中，每片磁碟可被劃分爲 35 條同心圓的環帶記憶空間。這數目是被磁頭的隙縫寬度，控制馬達的誤差度及一些其他的技術因素所限制住的。



但是對於一般的使用情況而言，僅將一片磁碟區分爲35個空間還是太少了。例如，有些檔案所需要的記憶空間事實上只相當1／10條磁軌的記憶容量，那用整條磁軌去記錄這個檔案顯然是浪費了。於是 APPLE 的DOS 3.3 就把每條 track 再細分爲16個更小的記憶單位，叫 sector 。 APPLE DOS 對磁碟上記憶空間的使用就是以 sector 爲最小單位。在標準 DOS 3.2 或

DOS 3.3中這相當於 256 byte（位元組）的實際資料。



爲了識別的方便起，每個sector都會有一個address field（位址資料區），然後跟在後面的才是眞正的data field。

磁碟用來識別資料的方法

1.自我同步位元組（SELF - SYNG BYTE）

前面曾提到，磁碟的資料是以bit serial 的方式輸出的。在實際應用上，我們必須將這一連串的bit（位元）轉換爲獨立的byte(位元組）才行。但是我們要如何識別那一個bit是那一個byte 裏的資料呢？如下例。

1 1 0 1 0 1 1 0 0 1 1……

你要說它是“1101 0110”或是“…11 0101 1001…”呢？自我同步位元組（self-sync byte ）就是用來解決這個問題的。爲配合self-sync的作用，硬體上也採用了一個工作原則，既：每一個byte 的開頭第一個bit必需爲1。由這個工作原則的配合，我們對byte 的區分動作可能爲如下之一種：

5 AUTOSYNC BYTES

```
11111111001111111100111111110011111111001111111100
11111111001111111100111111110011111111001111111100
11111111001111111100111111110011111111001111111100
11111111001111111100111111110011111111001111111100
11111111001111111100111111110011111111001111111100
11111111001111111100111111110011111111001111111100
11111111001111111100111111110011111111001111111100
11111111001111111100111111110011111111001111111100
11111111001111111100111111110011111111001111111100
11111111001111111100111111110011111111001111111100
```

在這個例子中我們所見到的self-sync位元組是由10個bit組成的FF（16進位表示法，HEX）：

SELF-SYNC BYTE：FF （HEX）

1	1	1	1	1	1	1	1	0	0

在例子中各位可看出五個這樣的數碼之後可保證資料的識別正確，也就是說〝同步〞上了。這正是Apple DOS 3.3中所用的self-sync。或許有些讀者會想，爲什麼只用兩個零呢？用3個零不是更好，那樣同步會更快，更有效。的確有，那種同步數

碼爲FE(HEX)，有相當多的商用磁碟上是用這個同步記號。但是這種同步位元組是9個，而非10個，位元組所組成的byte:

SELF-SYNC BYTE : FE （HEX）

1	1	1	1	1	1	1	0	0

這個原因是硬體上的另一種限制：大多數的磁碟機及其驅動器（ disk driver ）無法穩定的連續處理兩個以上的零。

至此，我們已提出了一套可行的方法來識別磁碟上的位元串是如何被轉換爲位元組的。但是，在提出這個方法的同時，我們也對位元組可具有的型態加上了兩種限制：

⑴每個byte開頭的第一個bit一定要是〝1〞。

⑵不可以有兩個以上連續的〝0〞出現於任何地方，否則可能造成磁碟失去同步的現象。

由於這兩種限制的出現，我們不難發覺有許多形態的數碼將不被允許出現於資料之中，譬如

8F ： 0100 1111……頭一bit爲零。

E2 ： 1110 0010……連續叁個零。

但是，計算機的資料是不可能有這些限制的，我們沒有理由說8F或E2不會出現在計算過程中。於是乎，我們就必須想辦法將計算機中的數字重新編一次（ encode; 編碼），使編得之碼符合磁碟機的要求。然後再將這新編的碼輸入磁碟之中。這也就是說，在磁碟中所記錄的並不是直接的原始資料，而是一經過轉換之後的〝影像〞（ image ）。在讀出這〝影像〞之後，一過一道〝解碼〞（ decode ）的工作程序，而後才能得到

入的資料。

2.資料的編碼與解碼（ ENCODING AND DECODING ）

能符合前面所述的要求的編碼方法很多，我們這裏談兩種，它們都是爲Apple所採用的方法。

第一種方法，不論編碼或解碼都很方便而且很穩定可靠，但是却相當浪費空間。它的作法如下：

＜法一＞：既然硬體要求頭一bit必需爲1，而又不能有連續兩個以上的零出現，那如果我們將一個byte 的資料（D_7至D_0）記爲如下的兩個byte

$D_7D_6D_5D_4D_3D_2D_1D_0$ —— 編碼前

⇨ $1\ D_7\ 1\ D_5\ 1\ D_3\ 1\ D_1$ ⎫
$1\ D_6\ 1\ D_4\ 1\ D_2\ 1\ D_0$ ⎭ 編碼後

那不是一切問題都解決了嗎！

這個方法的編碼是利用移位（ SHIFT ）和邏輯「或」指令（ logic OR ）來完成的（讀者可自行想想看）。其解碼的方法更是簡單，只需將第一byte 的資料向左旋轉一個bit的位置，再和第二byte 的資料〝AND〞一次便可以了。

$D_7\ 1\ D_5\ 1\ D_3\ 1\ D_1\ 1$ （ROTATE left 1 bit）
AND) $1\ D_6\ 1\ D_4\ 1\ D_2\ 1\ D_0$

$D_7D_6D_5D_4D_3D_2D_1D_0$

這樣的方法稱之爲EVEN-ODD ENCODING(奇偶編碼法)。

雖然這個方法擁有方便，可靠兩種好處，但是它對記憶空間

的使用效率却實在是太低了。一個byte 的資料原本應該可以代表$2^8 = 256$ 種不同的訊息，去掉那些因硬體的限制而不使用的成分之後該還可含有大約70種的訊息。但在＜法一＞裏，每一個byte(8 bit)所表示的訊息却只有$2^4 = 16$ 種。針對這個缺點，我們再介紹第二種方法

＜法二＞：假如，我們將資料作如下的轉換：

" 6 and 2 "

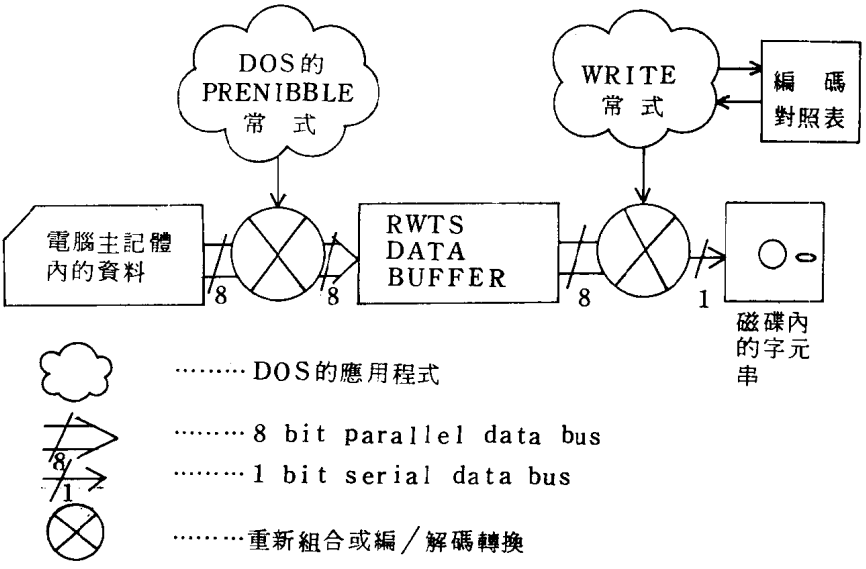
WRITE TRANSLATE TABLE

00 ⇨ 96	10 ⇨ B44	20 ⇨ D6	30 ⇨ ED
01 ⇨ 97	11 ⇨ B5	21 ⇨ D7	31 ⇨ EE
02 ⇨ 9A	12 ⇨ B6	22 ⇨ D9	32 ⇨ EF
03 ⇨ 9B	13 ⇨ B7	23 ⇨ DA	33 ⇨ F2
04 ⇨ 9D	14 ⇨ B9	24 ⇨ DB	34 ⇨ F3
05 ⇨ 9E	15 ⇨ BA	25 ⇨ DC	35 ⇨ F4
06 ⇨ 9F	16 ⇨ BB	26 ⇨ DD	36 ⇨ F5
07 ⇨ A6	17 ⇨ BC	27 ⇨ DE	37 ⇨ F6
08 ⇨ A7	18 ⇨ BD	28 ⇨ DF	38 ⇨ F7
09 ⇨ AB	19 ⇨ BE	29 ⇨ E5	39 ⇨ F9
0A ⇨ AC	1A ⇨ BF	2A ⇨ E6	3A ⇨ FA
0B ⇨ AD	1B ⇨ CB	2B ⇨ E7	3B ⇨ FB
0C ⇨ AE	1C ⇨ CD	2C ⇨ E9	3C ⇨ FC
0D ⇨ AF	1D ⇨ CE	2D ⇨ EA	3D ⇨ FD
0E ⇨ B2	1E ⇨ CF	2E ⇨ EB	3E ⇨ FE
0F ⇨ B3	1F ⇨ D3	2F ⇨ EC	3F ⇨ FF

AA
D5 Reserved Bytes

則我們可以8個bit 完全符合要求的數碼表示出6個bit 的資料。於是，對於記憶空間的使用率就提高到了75%的效率。雖然如此，但是這種編碼的方法却是比較煩複的。它需要較多的編碼及解碼。這是由ＤＯＳ中一個名爲prenibble 的常

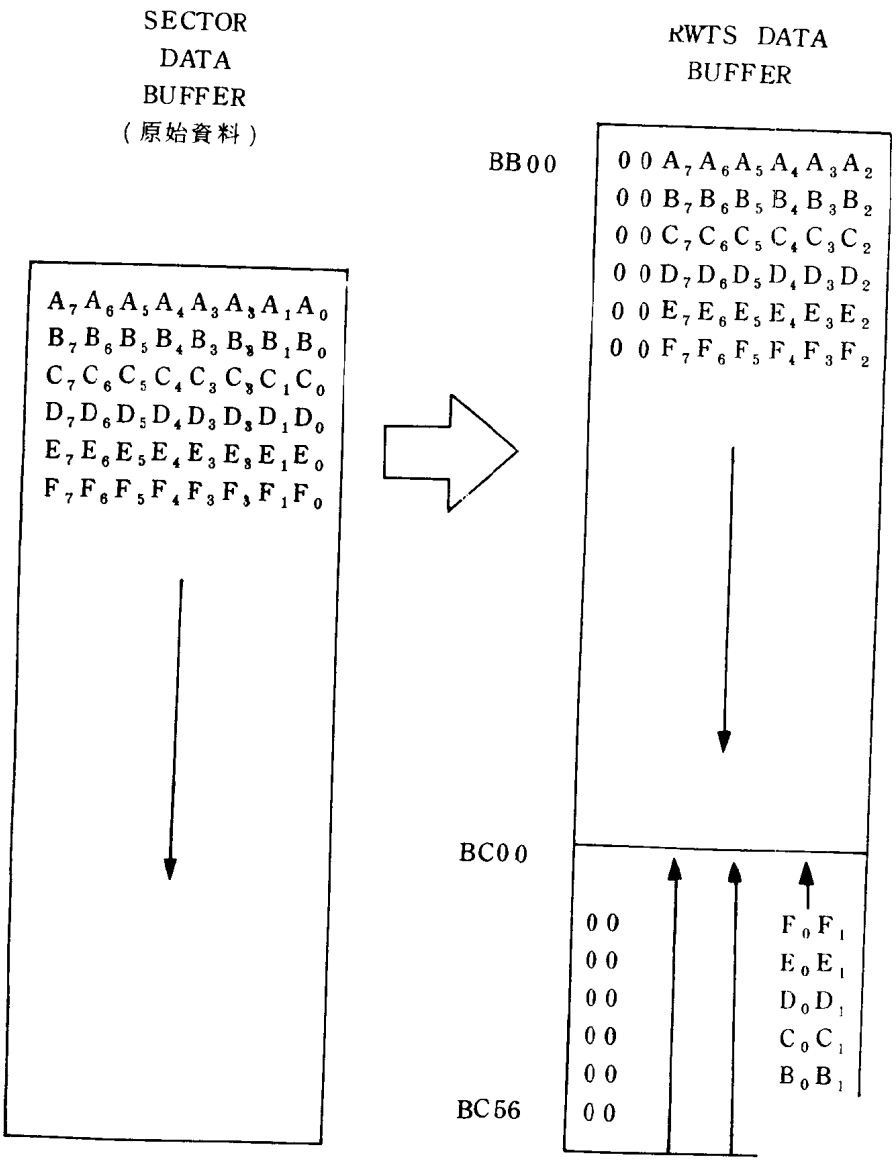
tine) 來完成。由於這個程式用了許多的步驟來重新組合資料，在此我們僅將兩者之間的對應關係圖示如下*：



這個圖顯示的是編碼及寫入的程序。解碼的過程大致相同，不另舉例。圖中RWTS DATA BUFFER中資料與原始資料間之關係可表示如下圖：

*詳請參考：APPLE徹底研究㈢，磁碟系統篇，第6章，波前，1984

" 6 and 2 PRENIBBILIZING



RWTS DATA BUFFR 中的資料再依照前述的編碼對照表轉換後（由WRITE常式完成）就可以成爲磁碟可接受之數碼而寫入磁碟之中了。

以上我們所介紹的兩種方法就是DOS 3.3中所用的資料記錄法。第一種方法由於解碼方便而且穩定，被用在記錄sector address上（下節將解釋何謂sector address），因爲sector的辨認必需迅速，而且由於資料數量很少，不會佔用太多的空間。第二種方法因爲對磁碟記憶空間的使用率高被用在一般資料的記錄上。至於DOS 3.2，由於推出時間較早，硬體的限制更大。它甚至不容許連續兩個的零出現，這使得在第二類的方法中我們只能作到每八個bit代表五個bit資料的地步。（對第一類方法則無影響）。也就是說它對磁碟記憶空間的最高使用率只有5／8約60%，而非DOS 3.3的75%。這便是爲何DOS 3.2的每一條track上只有13個sector，而DOS 3.3却有16個sector的原因了。

下面我們將來看一下Apple的磁碟系統又是如何的把一條track劃分出一個一個的sector，而系統又是藉由何種方法來區分出它所要找的sector的。

SECTOR的劃分

1.位址資料區（ADDRESS FIELD）

前面提到過，在標準的DOS裏一條track又再被細分爲16個sector。爲了識別方便起見，我們在每個sector內附了一個"地址"區域（sector address field）。這個區域內包含的

資料有四種:DISK VOLUME ;TRACK ADDRESS;SECTOR ADDRESS 及一個用來做核驗的CHECKSUM。這Sector address 的功用有如住家的門牌，可使電腦曉得它現在正處理的是那個 sector 的資料。

2.前導記憶與終結記憶(PROLOGUE & EPILOGUE)

但是問題來了，我們要怎麼分辨那些數據是sector address，而那些數據又是眞正的data field 呢？解決這個問題的方法是：我們用一些特殊的字符串附於各區域的前後，用這些特別符號來把資料隔開。這就是所謂的「前導記號」(PROLOGUE) 與「終結記號」(EPILOGUE)。在APPLE DOS 中所用的這類記號有：

DOS 版本	地址(Address)		資料(Data)	
	前導記號	終結記號	前導記號	終結記號
DOS 3.3	D5 AA 96	DE AA EB	D5 AA AD	DE AA EB

讀者可驗證一下這些數碼是否合於磁碟系統的兩個限制，同時在編碼對照表中這些字符也被保留下來而未被使用。由於這些特殊字符的使用，我們可以很確實的辨認出那些數碼是帶著位址訊息，而那些又帶著是資料訊息的。

關於在這兩個區域(address field 和data field)中所採用的資料記錄式，討論如下：