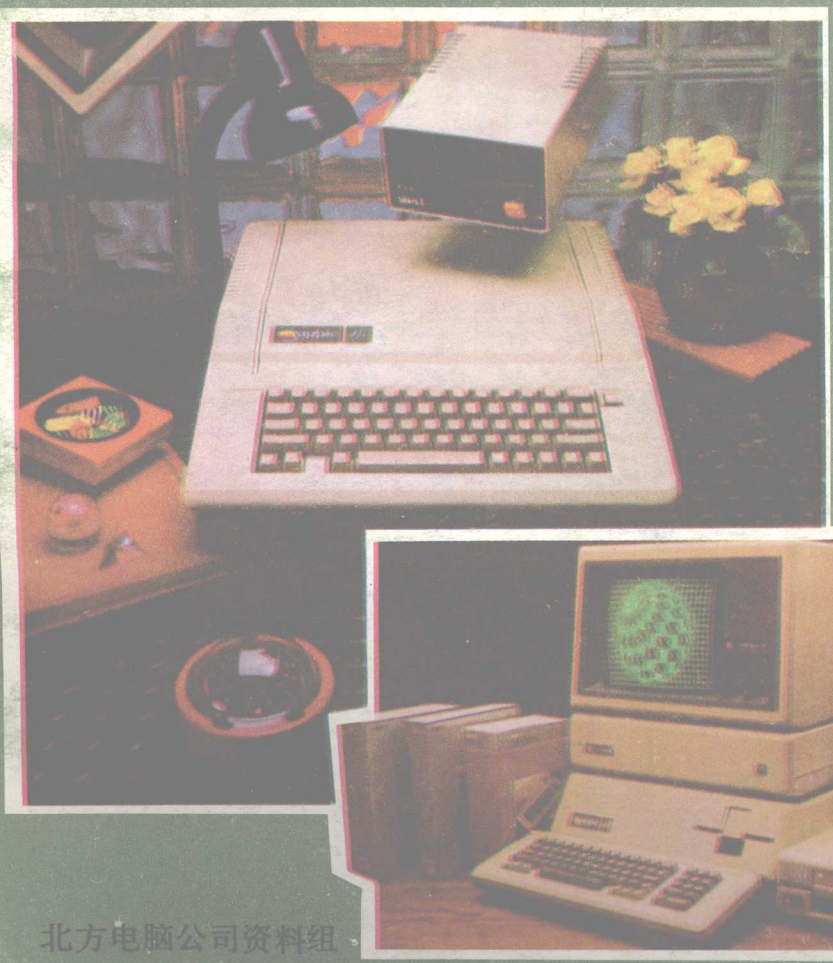
APPLE II

彻底研究(之二)



北方电脑公司资料组

APPLE Ⅱ
微電腦徹底研究

(二)

北方电脑公司信息资料部

序言

這本書是配合APPLE Ⅱ徹底研究㈠而寫的，如果讀過第一册之後再閱讀本書，那麼我希望至少能夠提供您幾點幫助：一對APPLE的瞭解更加深入。二增進編寫及解讀機械語言程式的能力。三彈性運用電腦內藏的系統功能。四徹底明白事物原理後的樂趣。

第0章除了爲引起讀者興趣之外，也指出了COUT、READ等副程式的重要性，如果對GAME等程式保護方法有興趣的話，可以直接閱讀附錄A至F，連同第0章本書總共列了9種磁帶程式保護的方式。第一章是COUT副程式的詳細探討，唯有完全明白COUT的運作原理，我們對APPLE的畫面才能完全掌握。第二章便是模仿第一章APPLE TEXT螢幕的處理技巧來處理高解像畫面，我們發現可以很方便的混合文字、低解像方塊、高解像

圖形於同一畫面上。第三章是說明如何運用&向量來造出新的BASIC指令，同時也嘗試造出一個小小型的BASIC解譯程式，當然這個解譯程式不獨可以由&來使用，利用CALL或COUT讀者也可以自行造出同樣的效果吧！第四章是將READ、WRITE等串行資料處理副程式徹底的加以瞭解，同時也介紹了其他幾種微電腦的讀寫處理方式，除了相互比較之外，也可以做爲程式編寫的參考。

總之，進入第二册之後，我們發現APPLE可以做更多更有用的事，也更徹底的瞭解電腦內部的運作情形了。個人能力有限，內容之表達、校訂多有不完善處，如蒙高明不嫌指正，則必感激不盡。將個人的樂趣分享每一個人，該是本書的目的吧！請大家愉快的來讀這本書！

作者　施威銘

唐凌 序

假如有一天，你用你自己的腦筋，寫了一本書，送到出版社去，或則只是費力地寫好一篇文章，投到某專業性雜誌去發表，你可能遭到一項極大的困難——出版商或雜誌的編輯會向你索取「原文」。

出版商或雜誌編輯之希望文有出處的心理，是不難理解的，一是沾光，一是如有謬誤可一推了之。

事實上，這樣的心態，普通地存在於現居台灣的每一個人心裡，而不只是出版界。

做事有個依據是對的，然而任何依據本身不可能是眞理，即或是眞理，在被引用的過程也可能造成扭曲。

下面就是一段隨處可見而扭曲得可笑的例子：

「在半計数（Counter halves）之間，外來的顫動（Jumper）必須供給。

「若經由Clock 1進入，而且Q 8跳躍至Clock 1，計数比重為1－2－4－5，Q 1是位元比重最大的輸出而且在輸出端有一個對稱的方波。……

這是一段對 7490 動作的描述，如果你對 7490 毫無概念，那麼看了這一段，眞的會令你「顚動」不已。

不幸的事情是，對這樣荒唐的謬誤，出版商給予容忍，讀者也同樣給予容忍。於是成千上萬的學子們，只好一天天地「顚動」下去。

另外一種情況也是很可悲的，也就是不知從什麼時候開始，大學生都喜歡抱厚厚的洋裝書，使得一些類如三用表用法、OTL、OCL設計或 CMOS 玩具製作之類的小書，有興趣的人只好偷偷地看，總覺得這些小書，好像很不入流一樣。

我與施威銘先生只是初識，當我看完「APPLE Ⅱ微電腦徹底研究」第一册之後，覺得與我主編的「音響技術月刊」頗有其臭相類之處，那就是作者躍然紙上，隨時與你共處。欲這樣，顯然是需要一些勇氣的，一些面對可能的疏失的勇氣，一些面對抱洋裝書的人的勇氣。我未嘗聽說有人照著食譜做菜，把菜燒焦了、燒爛了、糊了，會找作者理論一番的，但如果你寫技術的書、電腦的書，隨時你都得迎接焦了、爛了、糊了的挑戰。

總之，這不是一篇「序」，而是一點感想，我由衷地希望，每一位讀者都能珍惜作者施威銘先生在這本書中「掏」給你的東西。

目　錄

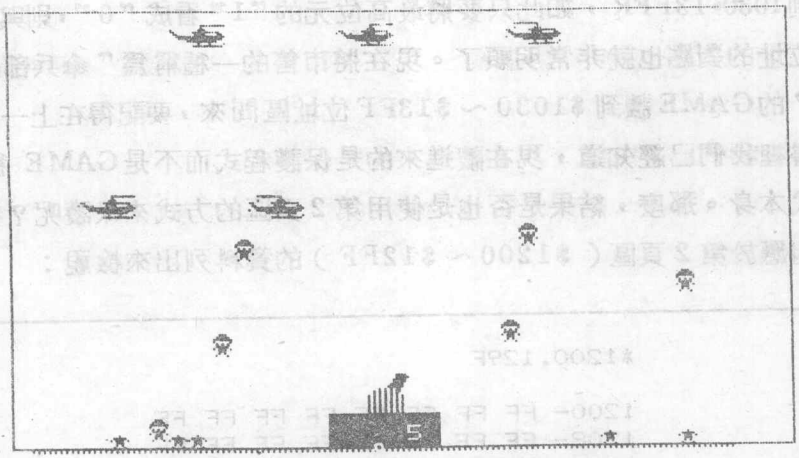
第〇章　30·3FFR

第1章　COUT徹底研究

第2章　蘋果與教授

30. 3FFR 0

—— 傘兵部隊 ——



0—0 30·3FFR

在第一册當中，我們說明了第 2 頁區的功能，因此譬如 200·1200 R這一類的程式保護，相信大家已清楚的瞭解其運作方式而運用自如了。至於其它的保護方法呢？此處讓我們來看 30·3FFR這種程式……

將程式以 30·3FFR讀進 APPLE裡頭，結果與第2頁區的功能一樣，會自動執行程式，這樣的功能必然是如 200·1200

一般，使用了APPLE的特殊系統結構才能達成，而APPLE的特殊系統使用位址是如附錄所列出的第0頁、第1頁、第2頁、及部份的第3頁位址，而30•3FFR竟然涵蓋了大部份的這些敏感位址，因此解決的第一步是仿照200•1200的方式，先將程式存到自由使用的記憶位址（避開上述的敏感位址），然後將讀進來的資料列出，再慢慢解讀！譬如，我們可以將30•3FFR存到1030•13FFR，如此只要將最高位元的"1"看成"0"，則與原位址的對應也就非常明顯了。現在將市售的一種稱爲"傘兵部隊"的GAME讀到$1030～$13FF位址區間來，要記得在上一册書裡我們已經知道，現在讀進來的是保護程式而不是GAME程式本身。那麼，結果是否也是使用第2頁區的方式來保護呢？將對應於第2頁區（$1200～$12FF）的資料列出來檢視：

```
*1200.129F

1200- FF FF FF FF FF FF FF FF
1208- FF FF FF FF FF FF FF FF
1210- FF FF FF FF FF FF FF FF
1218- FF FF FF FF FF FF FF FF
1220- FF FF FF FF FF FF FF FF
1228- FF FF FF FF FF FF FF FF
1230- FF FF FF FF FF FF FF FF
1238- FF FF FF FF FF FF FF FF
1240- FF FF FF FF FF FF FF FF
1248- FF FF FF FF FF FF FF FF
1250- FF FF FF FF FF FF FF FF
1258- FF FF FF FF FF FF FF FF
1260- FF FF FF FB FF FF FF BF
1268- FF FE E9 F9 BA AB FF FF
1270- FF FF FF FF FF FF FF FF
1278- FF FF FF FF FF FF FF FF
```

結果，我們並未發現存在任何有用的ASCII 指令＊！也就是說第2頁區內並未存有保護程式。其它位址呢？第3頁只不過是一些中斷或重置指位器，而第一頁是堆疊暫存區，因此最具可能性的該是0頁位址區了！可是0頁是最爲複雜的一個頁區，如何找出問題的所在呢？

0-1 COUT

APPLE的零頁位址有一對指位器（CSWL、H）=（$36，$37）與程式的自動執行有關，這對指位器是由監督程式內的COUT副程式來使用，每次卡帶程式讀入完畢後，便是以JMP COUT來令喇叭發出聲音並且使游標重現。而COUT所去執行的位址，則又由（CSWL、H）來指向，因此，只要GAME程式進入位址存入（$36，$37），那麼一旦程式讀回完畢COUT被呼叫時，便會跳往程式進入點，GAME便自動執行了。我們可以先寫下一小段程式如下：

＊見徹底研究㈠第0章

```
*300:20 DD FB 4C 00 03

*300L

0300-   20 DD FB    JSR   $FBDD
0303-   4C 00 03    JMP   $0300
0306-   FF          ???
```

鍵入 300G 看看，這是連續使喇叭發聲的程式，RESET之後，將 $0300 存入（ $36 ， $37 ），並且連同主程式存入到卡帶上：

```
*36:00 03 N 30.3FFW
```

當程式存入完畢後，程式會自動執行，這是因爲（ $36 ， $37 ）已經指向 $0300 ，當程式存完之後游標重現時，COUT 被呼叫，於是程式便執行了。RESET之後，再將程式讀回：

```
*30.3FFR
```

結果如何？是否與GAME的程式一樣效果呢？如果將主程式裡的喇叭發聲副程式BELL 1 = $FBDD改成BELL = $FF3A ，結果發現程式不能動作，這是因爲BELL副程式使用到COUT ，而COUT改被改變了方向，功能亦已改變，所以便達不到預期的結果。

如果是欲將存入卡帶的資料先在$1030至13FF的位址區間安排完畢，再存入卡帶，結果應該也可以動作吧？

```
*1036:00 03
*1300:20 DD FB 4C 00 03
*1030.13FFW
*30.3FFR
```

當由卡帶讀回程式時，發現系統似乎失去控制了，爲什麼呢？因爲零頁位址內有許多由系統來使用的指位器、暫存器*、旗號指示器等等，現在除了（$36，$37）是存入指定的數值外，其他位址則是存入$1038至$10FF的一些亂數，當然系統會當掉了。譬如（$3C，$3D）及（$3E，$3F）這兩對指位器……。

0-2 READ

再回到"傘兵部隊"的GAME來，難道這麼複雜的程式竟然可以存放在$30至$3FF不到1K的區間嗎？那麼依照上一節的結論來看看（$36，$37）所指的程式進入點在何處吧！

---

* 是指暫時儲存資料用的記憶體，不是CPU內部的暫存器。

```
*1030.10A0

1030- 00 00 FF AA 08 28 00 1D
1038- 81 9E 28 90 3C 00 00 90
1040- 30 00 00 98 2D 98 AE D8
1048- 00 B7 00 00 00 00 3E 45
1050- 69 FF 55 52 00 0D 33 08
1058- 00 00 00 00 00 00 38 DE
1060- 00 00 00 00 00 00 00 01
1068- 08 44 08 44 08 44 08 00
1070- 96 FF 95 00 96 0A FF 00
1078- 00 28 08 00 00 00 08 00
1080- 00 00 00 00 00 00 00 FF
1088- 00 00 00 00 00 00 00 03
1090- 4C 46 00 00 00 00 00 00
1098- 00 00 00 00 00 88 FF FF
10A0- FF
```

現在觀察$1036，$1037這兩個位址，意外的，程式進入點竟然是$1D00！那麼程式是如何存入的呢？

（$3C，$3D）及（$3E，$3F）這兩對指位器是用來做爲卡帶讀入的起始及終止位址指示用的。由上表　可以看出（$3E，$3F）現在所存的是$00與$90，原來程式讀入到（$3E，$3F）時便將$9000存入這對指位器裡，如此一來程式就不止存到$3FF而已，必須存到$9000才算完整。現在問題就很明顯了，程式事實上是存在於$30至$9000之間，而進入點是$1D00。但是存放程式讀入起始位址的（$3C，$3D）呢？爲什麼這個指位器是指向$003C而不是$0030呢？這是因爲如果程式剛好讀入到（$3C，$3D）位址時，而把$0030存入其中的話，這等於是將（$3C，$3D）又指向$0030，如此APPLE會將接下來的資料又重新由$0030開始來存放了，因此必須存入

$003C才可以使接下來的資料依序讀入。

現在已經知道程式的起始、終止位址以及進入點了，那麼如何來解開這種保護呢？

0—3 保護與解法

如果只是爲了拷貝，那麼方法很簡單，避開零頁然後以1030 A000 R來將原來由$30至$9000的程式讀至APPLE裡頭，接着以1030．A000W便可以複製程式了，當然執行時還是要以30．3FFR來讀入才行。但如果是要觀察程式內容的話，那就必須將有關RESET指位器$3F2、$3F3、$3F4等位址加以修改才行，否則於操作當中可能會使程式破壞而須重新再讀入一次。

這一種連主程一齊只有一段的卡帶程式，甚至不必加以解讀其位址與進入點也一樣可以拷貝，只要避開0頁然後以最大的容量830．BFFFR來讀入便可以了，雖然最後可能出現ERR字樣，但是程式仍然是可以正確讀進來的。同理，譬如第一册介紹的200．1200R那種具有兩段錄音的程式，可以用800．1800R 1801．BFFFR來讀入程式，而以800．1800W 1801．BFFFW來複製，當然複製程式仍然是要以200．1200R來執行。於是拿來一捲稱爲"大進擊"的卡帶以1030．BFFFR讀入，然後1030．BFFFW複製了一捲卡帶，最後以30．3FFR執行，結果……？

卡帶的保護方式很多，除非徹底瞭解其原理，否則可說是解不勝解，對上述那種不經解讀程式而直接拷貝的方法，如果程式

是三段的話？四段的話？甚或是以稍異於APPLE的讀寫副程式來存取磁帶資料的話？盲目的拷貝是行不通的，對於這些問題有興趣的讀者，請直接閱讀本書附錄A至附錄F的8種保護法及例子。