

APPLE II

彻底研究(之二)



APPLE II

微電腦徹底研究

(二)

The Apple II monitor, Model: William H. Green (Vice-President of Apple Computer Inc.)
Hardware engineer: Martin Morris
Software engineer: Steve Jobs
(Apple II (based on the Apple II+ group)
CPU: MOS 6502 (8-bit)
RAM: 4K (expandable to 16K)

Monitor express (based on the Apple II+ group)
CPU: MOS 6502 (8-bit)
RAM: 4K (expandable to 16K)
Keyboard: 50 keys (based on the Apple II+ group)
Monitor express (based on the Apple II+ group)

北方电脑公司信息资料部

序 言

在推出尚未用過的硬體時，上面會印出幾句話：「這是第一款外接顯示器，它的小巧與易用性，將帶來多樣化的應用。」這句話其實是出自於 RICHARD·MURKIN，他當時是 Apple 公司的工程師，我們可以說他是最早開始研究 Apple II 硬體的人。這段文字以直白、大膽的語句開頭，失去單獨寫作的感覺。

本章主要講述如何使用 Apple II 的文字模式，它將開啟你對個人電腦的新視野。了解顯示器的基本運作原理之後，專研視覺必順，玉成辦不勝感激。謹希望不會令讀者失望。至於內容方面，大概是目前市面上最簡單易懂的一冊，毫不誇張！

】書本影響來自於香港

這本書是配合 Apple II 徹底研究(一)而寫的，如果讀過第一冊之後再閱讀本書，那麼我希望至少能夠提供您幾點幫助：一對 Apple II 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 章除了為引起讀者興趣之外，也指出了 COUT、READ 等副程式的 importance，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 II 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，也可以做為程式編寫的參考。

總之，進入第二冊之後，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 祁盛餘

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真理，即或是真理，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：

「在半計數(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 跳轉至 Clock 1，
計數比重為 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，真的會令你「顫動」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「顫動」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」第一冊之後，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。

溫度

目 錄

第0章 30·3FFR

0 — 0	30·3FFR	1
0 — 1	COUT	3
0 — 2	READ	5
0 — 3	保護與解法	7

第1章 COUT徹底研究

1 — 0	COUT 字元輸出副程式	9
1 — 1	COUT 的應用	12
1 — 2	COUT 1	16
1 — 3	VIDOUT 的零頁位址	19
1 — 4	VIDOUT 的基本副程式單元	21
1 — 5	VIDOUT 總整理	30
1 — 6	CR 副程式	32

第2章 蘋果與教授

2 — 0	蘋果與教授	38
2 — 1	高解像畫面提要	39
2 — 2	用 PRINT 指令在高解像畫面印字	48

2 — 3 在高解像畫面造出低解像圖形	59
---------------------------	----

第3章 創新BASIC指令

3 — 0 “ & ” 與 TXTPTTR	68
3 — 1 SWAP —— APPLE Ⅲ ?	69
3 — 2 GOTO A 與 GOSUB A	73
3 — 3 SOUND 、 BELL 及其他	78
3 — 4 創新BASIC 指令	80

第4章 WRITE與READ徹底研究

4 — 0 硬體線路：\$C020 與 \$C060	91
4 — 1 串列訊號的處理	95
4 — 2 常用的卡帶讀寫方法	97
4 — 3 APPLE 的 WRITE	103
4 — 4 HEADR	107
4 — 5 APPLE 的 READ	115

附錄A 30.300R (金甲蟲)

A — 0 30.300R	123
A — 1 程式檢討	124
A — 2 保護方式解說	126
A — 3 還有一層保護	128

附錄B 30.1200R (大進擊 II)

B — 0 30.1200 R	131
-----------------------	-----

B—1	程式檢討.....	132
B—2	保護方式解說.....	134
B—3	盲目拷貝行不通.....	135

附錄C 200.300R (戰略作戰)

C—0	200.300R	137
C—1	程式檢討.....	138
C—2	保護方式解說.....	143

附錄D 200.300R (攻陷地球)

D—0	200.300R	145
D—1	保護方式解說.....	148

附錄E 200.2FFR (世界末日)

E—0	200.2FFR	151
E—1	程式檢討.....	153
E—2	保護方式解說.....	155

附錄F 3種BASIC保護法

BASIC I	POKE 2049,0	157
BASIC II	POKE 214,128	158
BASIC III	POKE 82,213	158

附錄G 6502 指令表

30. 3FFR

O

傘兵部隊



0-0 30.3FFR

在第一冊當中，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為“傘兵部隊”的GAME讀到\$1030～\$13FF位址區間來，要記得在上一冊書裡我們已經知道，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									
1208-	FF									
1210-	FF									
1218-	FF									
1220-	FF									
1228-	FF									
1230-	FF									
1238-	FF									
1240-	FF									
1248-	FF									
1250-	FF									
1258-	FF									
1260-	FF	FF	FF	FB	FF	FF	FF	BF		
1268-	FF	FE	E9	F9	BA	AB	FF	FF		
1270-	FF									
1278-	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 頁是最為複雜的一個頁區，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(→第0章)

據中！ * 合規 ASCII 用法本章並未述及，果請

*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

當程式存入完畢後，程式會自動執行，這是因為 (\$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，這是因為 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，為什麼呢？因為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。

*1030.10AO

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
10AO-	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為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？為什麼這個指位器是指向 \$003C 而不是 \$0030 呢？這是因為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 · BFFF 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，譬如第一冊介紹的 200 · 1200R 那種具有兩段錄音的程式，可以用 800 · 1800R · 1801 · BFFF 來讀入程式，而以 800 · 1800W · 1801 · BFFF W 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 · BFFF 來讀入，然後 1030 · BFFF W 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。

本章完結。6-0

600 001 時未見李顯譽，單獨處一室中，且翻看舊書。收拾
整理 3.1996 年春分至清明時，李的多由李昇昇來 3.1996
年春分至清明時，丁人再進來。而後 3.1996.03.1996.03
中傳真，臨時容尚未接通，只回信。3.1996.03.1996.03
神山派祖師道慈 3.1996.03.1996.03.1996.03.1996.03
文一大師掛像原系面壁坐忘於香爐石中，當增其外觀古，香衣顏

色綠如臘，不老矣。大師前，和對一高僧，劉一先生識書。該
客請大眾以書於寫 3.1996.03.1996.03.1996.03.1996.03
中見題，提出書稿，並簽名。以下收入編本 3.1996.03.1996.03
書稿全稿一函誠寄。聖同，請勿取用。可以印製此函為最宜。
3.1996.03.1996.03.1996.03.1996.03.1996.03.1996.03
1996.03.1996.03.1996.03.1996.03.1996.03.1996.03
景純。已錄入 3.1996.03.1996.03.1996.03.1996.03.1996.03
書稿，並給 3.1996.03.1996.03.1996.03.1996.03.1996.03
東詩。已錄入 3.1996.03.1996.03.1996.03.1996.03.1996.03
此为试读, 需要完整PDF请访问: www.ertongbook.com

此为试读, 需要完整PDF请访问: www.ertongbook.com