

致 讀 者

「APPLE 界面卡電路原理與設計」這本書，是**大眾電子**編輯部同仁，專門為讀者企劃編輯的微電腦硬體技術叢書之四。

由於很多學硬體的朋友常來詢問有關於界面電路的技術問題，總覺得界面電路的技術牽連非常廣泛，要一一瞭解須耗費很多時間，即使原理能夠瞭解，但是還是無法動手去製作一個符合自己須要的界面電路，有鑑於此，故本書是從有關的界面資料中挑選了近二十篇在原理的解說與製作方面均實用的文章彙編而成，以供愛好電腦硬體界面的朋友參考，並從其中學到設計的技術。

本書包括了各種界面的原理說明，硬體電路設計及軟體程式的設計，每一篇文章都可以成為一個「專題製作」，最適合大專同學的專題製作及作為愛好電腦硬體的讀者之設計範例。

儘管本書在編輯過程中經過審慎的處理，但錯漏之處仍難避免，尚請先進惠予指正，以供本社改進參考，僅此致謝！

梁 超 謹誌於**大眾電子**編輯部

民國 75 年 6 月

製作一個 GRAFEX-32 界面卡

使你的APPLE能顯示640×400點

Apple II 的 280 × 192 點繪圖解析度已不能再說是“藝術的結晶”了。新的 Apple Macintosh 和 Lisa 電腦，其顯示圖像時，可高達 32K 位元組的記憶容量，超過 Apple II 顯示影像時所用的記憶容量四倍以上。

最近由於 64K 動態RAM晶片和VLSI繪圖控制器以合理的價格推出市面，故祇要使用一片簡單的插入式線路板，即可輕易的將 Apple II 的繪圖解析度予以擴充。GRAFEX-32 界面板使用了 4 個 16K × 4 位元的RAM，配合 7220 繪圖控制器來把 32K 位元組的記憶容量當作 640 × 400 點解析度的位元圖像。這片板子再插入一個 64K × 4 位元RAM晶片就可以擴充成 128K 位元組，也可以在系統內裝 3 片同樣的板子，來提供彩色繪圖顯示所需要用的紅、藍、綠信號。

我們設計此線路板時，是採用未來可擴充電路的簡易方式，而不是採用一些沒人用過的複雜設計，雖然基本電路只包括

20個晶片，但它在標準的 Apple 監視器上會顯示 256000 點單色解析度。由於這種基準已超出 Macintosh 電腦的繪圖解析能力，所以我們不打算再增加基本電路板的複雜性，需要更高解析度與彩色影像的使用者，可利用改良硬體的方式達到目的。此種設計非常吻合 Apple II 的原有設計觀念。

APPLE II/IIe

Apple II / IIe 的成功要歸功於擴充槽的擴充能力。在這部電腦上市的六、七年間，一些硬體廠商接受了要將那些擴充槽填滿的挑戰，因而發展了各式各樣的界面板，因而也擴充了機器的功能，目前市面上所出售界面卡，令人驚異的是那些卡片應用的範圍幾乎是無所不包。

擴充 Apple II 繪圖解析度也是屬於界面電路中的一種應用。雖然有少數卡片是採用將影像重疊並配合亮度而顯示在螢幕上，但並不能增加螢幕上的解析度。關於

此種缺點，有些可能要歸因於IBM PC推出時分散了其他硬體廠商的注意力，當時市面上已有的VLSI繪圖晶片和便宜的記憶體，已使得擴充Apple II繪圖能力變成是一件很簡單的事。

在Apple II這方面除了可用增加螢幕解析度之外，還可獲得來自新繪圖技術方面的好處。在Apple II中是將顯示記憶體的內容映射至6502微處理器的某一段記憶位址內，這是採用一種名叫“記憶體映射的方式”的技術，這種技術是一種很好的選擇，因為它能让6502使用所有記憶參考指令和定址模式，向螢幕記憶體做一切的存取動作，就好像它是在做普通系統RAM定址一樣。事實上，假如沒有使用高解析度螢幕時，則分配給它使用的記憶空間就可以讓程式和資料儲存使用。Apple II的設計之所以要限制繪圖解析度，至少是有兩個理由：第一是受到家用彩色電視頻寬的限制，因為Apple II使用者有些是用彩色電視機來當作他們的顯示器。第二個原因是因為Apple的6502微處理機所能定址的記憶空間有其限制。6502這片晶片有16條位址線，可定址到65536個位元組的記憶空間，所以記憶映射視訊設計若佔用了一半的記憶空間是一件不切合實際的事。

記憶映射視訊設計並不是只限制在低解析度繪圖使用而已，此點可由新的Apple Macintosh和Lisa等機器加以證明。這兩種電腦中所用的68000微處理機是由24條位址線所組成的匯流排，可直接定址到系統記憶體16M位元組以上的記憶空間。雖然顯示記憶體的內容映射到主記憶體的繪圖RAM可使用較多的記憶位址，但並不表示儲存程式或資料所能用的記憶空間也會比較多。

改善Apple II的繪圖解析度而不需將記憶體全部用完的方法，是將繪圖用

RAM和系統用RAM分開。6502不能直接對顯示RAM做存取動作，但有一些新晶片最適於用來作大容量的記憶體映射方式。然而6502實際上是藉著不把系統RAM分出一部份給繪圖使用的方式，而增加程式可用的記憶空間。

7220 GDC

7220繪圖顯示控制器(簡稱爲GDC)是NEC公司所設計，用來處理需要以光域掃描方式在CRT上做圖形、線和字元繪圖這類工作。7220和以前只能用來做顯示更新及視訊同步這類工作的CRT控制晶片(例如MOTOROLA 6845晶片)不太一樣。

7220有一套指令集，可使它能在顯示記憶器中做讀取、修改和寫入資料等動作。

7220會對送到系統匯流排和顯示記憶器之間的指令有所反應，在不需處理機介入的狀況下作繪圖工作。由於GDC能處理相當多的像素繪圖和修改工作，因此相對的微處理機與顯示記憶器通道所需要的頻帶寬度也就非常窄。在作向量或作幾何圖形繪圖時，由微處理機送來的大部份資料，都一定是以命令和參數的形式直接送到GDC，而GDC在將命令及參數都譯成像素後，把那些像素放在高頻寬GDC/顯示記憶器通道上傳送出去。此時匯流排的頻帶寬度會加寬，而傳送速率也會調整到最快的速度來處理這些工作。例如，工作時脈頻率爲5MHz的7220可以用每條像素800ns的速率來畫一個圖。這個速度和所繪的圖形種類無關。並且比6502這類通用微處理機處理同樣工作的速度快多了。

此項設計之所以選擇7220是因為它的8位元資料匯流排可以和Apple的8位元6502直接界接，且7220的16位元資料匯流排和18位元位址匯流排，可使它能在不受6502之16位元定址的限制下，配合大

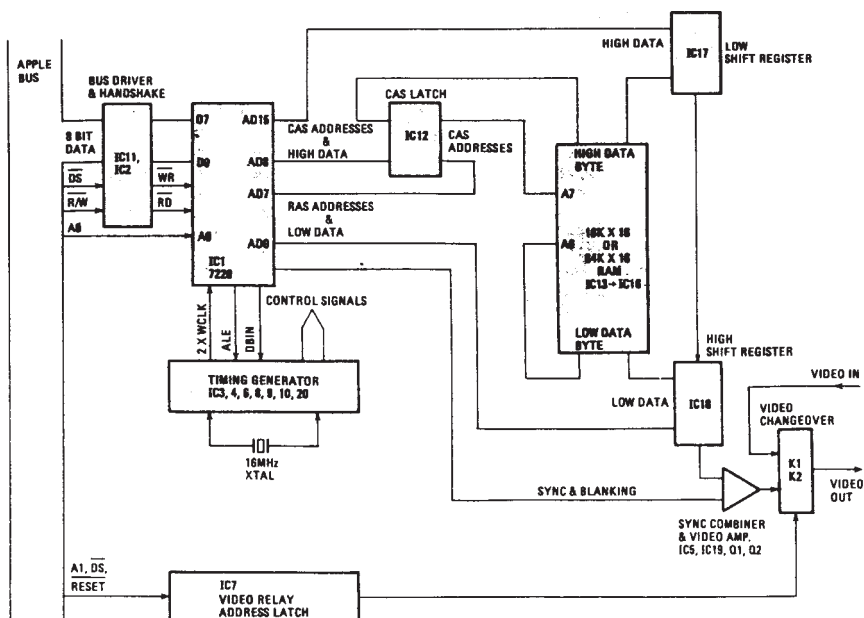


圖 1 GRAFEX-32 的方塊圖應配合詳細電路圖使用以便容易了解操作原理

的位元映射顯示記憶體一起工作。它也處理動態RAM更新與視訊同步產生的工作，這部份是裝在一片40根接腳陶瓷的DIP式包裝內，而且是在3μM NMOS上製造的，它含有13000個以上的等效電晶體。

顯示記憶體

7220要求它的顯示記憶器的結構是以16位元為一個字的方式組成，最常用的是64K動態RAM。例如4164，其結構是65536個Bit，所以需要16個Bit才能組成系統所要的16位元一個字。

另有一片64K RAM晶片是由每4個位元一組所組成的16384個位置。這個部份適用於4416，可由德州儀器、Inmos和Fujitsu等公司購得。7220所需要的16位

元資料匯流排可由4個64K RAM組成，如此可減少功率的耗損與印刷电路板的複雜性。若4416使用單獨的啟動接腳時，則所需要的晶片數量可以減到最少，至於那根啟動接腳可以用三態激勵的方式來啟動。採用上述這種方式可以使7220和顯示記憶體之間能夠非常緊密的接合，更可簡化PC板的佈線。

德儀公司最近發表了4464，NEC公司也發表了41254的動態RAM晶片，都是由65536個4位元所組成。這兩種零件都是裝在和4416一樣的一個18根接腳DIP包裝內，這種18支接腳包裝的方式可使系統將來便於改用較大記憶容量而接腳數量相同的晶片。以Grafex線路為例，就是設計成能夠向上述兩種晶片所提供的增額記憶體作定址的方式，並將Grafex線路由

32K 位元組擴充成 128K 位元組，而擴充的方式只是換 4 片顯示 RAM 晶片而已。

電路工作原理

Grafex 電路依據圖 1 的方塊圖，以及圖 2 的詳細電路圖上可以了解，它可區分成五個部份。這五個部份分別為：

(a) 微處理機匯流排界面部份，(b) 7220 繪圖顯示控制部份，(c) 顯示記憶器部份，(d) 視訊輸出電路部份，以及 (e) 時脈產生器部份。

IC1 (UPD 7220D) 是經由 8 條雙向資料線可以與 6502 主處理機作交談。IC11 是 74LS245 用來提供作緩衝，而 IC2 是 74LS00，用來將 Apple 的讀／寫信號與 DS 信號轉換成 7220 所用的 8080 式的讀／寫信號。Grafex 電路以 16 條位址線中的 4 條位址線來指定裝在 Apple II 上的周邊插槽。這 4 條位址線是用來選擇 7220 的命令與參數暫存器，並用以控制視訊轉換繼電器 IC7。這 4 條位址線及其功能列於表一之內。Apple 匯流排上的重置線是與此視訊繼電器連接的，以便 Apple 開機時其視訊信號會產生一虛設信號。

7220 是透過 16 條經過多工處理的位址／資料線來和顯示記憶器交談的。這 16 條線在方塊圖上分別標示為 AD0～AD15。IC12，CAS 門鎖是一個 74LS374，用來門鎖位址線的高階 8 位元，然後循序將它們送入顯示記憶器的 RAS／CAS 8 條位址線內。這是一種以資料與位址用多工處理的標準方式所作的緊密耦合設計。7220 所載的低階位址／資料通道分別是行 (column) 的位址與列 (row) 的位址。最後是顯示記憶器資料的低階位元組。顯示記憶器的高階位元組是經由單獨的一條通道直接和 7220 的 AD8～AD15 接腳連接。

顯示記憶器 IC13～IC16 四顆 RAM，其中每一顆 IC 都帶有一個輸出啟動信號端子，以及經過多工處理的 8 位元位址流排與一組 16 位元資料輸入／輸出線。

IC13～IC16 四顆 RAM 的 8 條列位址線和其他動態 RAM 陣列一樣，會先被選通 (strobe) 到晶片內的門鎖上，接著 8 個行位址位元才會在位址線上顯示，最後才一起選通到記憶器內。存取時間符合規格之後，資料會在讀取週期讀出，在寫入週期寫入。4416 與 4464 等零件具有一條單獨的輸出啟動接腳，而輸出的啟動信號可使資料能由選擇的位址上讀出來，但除非必要，否則這些資料不會顯示在輸出接腳上。“G”接腳可允許輸出緩衝有一快速 (40 ns) 動作，以便於稍後的讀取週期需要時將資料供應給外部電路。8 條行／列位址線也可依照此種方式，來控制往返於 7220 與視訊移位暫存器之間的低階 8 Bit 資料，而不需由外部再加一個三態緩衝器。此種耦接方式可用於印刷線路板佈線上，並可用來減少電路的內部接線方式，改進其可靠度。

由顯示記憶器讀取的 16 位元資料，會顯示在 IC17，IC18 這兩個 8 位元的並列輸入／串列輸出移位暫存器上。這兩個 74LS166 移位暫存器會將資料轉變成串列式的視訊，其速率為 16 MHz／Dot-Clock Rate。7220 送來的遮沒信號、垂直同步信號、水平同步信號，是採用 74LS174 正反器將資料送入信號線上。它會採用和移位暫存器相同方法將信號載入。

視訊位元串會由 IC4 配合 7220 的遮沒信號一起輸入，然後和 IC5 這個互斥或閘所提供的複合同步信號混合。視訊放大器由 Q1 與 Q2 組成，提供 1V_{p-p} 標準合成視訊到 75 歐姆電阻上。此合成視訊信號是

接到視訊轉換繼電器 K1、K2 上。這兩個繼電器所作的選擇，若不是以 Grafex 視訊信號作為視訊顯示器的信號源，就是以外部輸入信號為信號源。當 Grafex 電路板裝在 Apple 內部時，外部輸入信號通常是接到 Apple 的視訊輸出接頭，而轉換繼電器輸出則是接到視訊顯示器上。

反之，兩個顯示器（一為普通顯示器，一為高解析度顯示器）可同時用來顯示 Grafex 和 Apple 的視訊，視訊轉換繼電器是以軟體啟動的，對外部輸入信號而言，是在重置時產生一個虛設信號，以便使 Apple 系統能在開機後正常操作。除非 Grafex 電路板被特別定址，否則依照此種

方式，使用者根本就不需要理會 Grafex 板是否存在。

時脈產生器是由一個 16MHz 石英振盪器和 IC3, IC4, IC6, IC8, IC9, IC10 及 IC20 所組成的，用來提供系統所要用的各種時序脈衝與控制信號。所有的時脈都是同時由 16MHz 時脈信號分頻而來的。時脈產生器有兩種操作模式，看 7220 是在執行顯示週期或讀取週期，修改週期或寫入週期 (RMW 週期) 而定。此兩種操作型態是以 7220 的 DBIN 來區分的。7220 輪流使用一個 2MHz 標名為 2Xwclk 的信號來作為主時脈信號。此時脈信號就如其名稱所隱含的意思一樣，是以顯示字元速

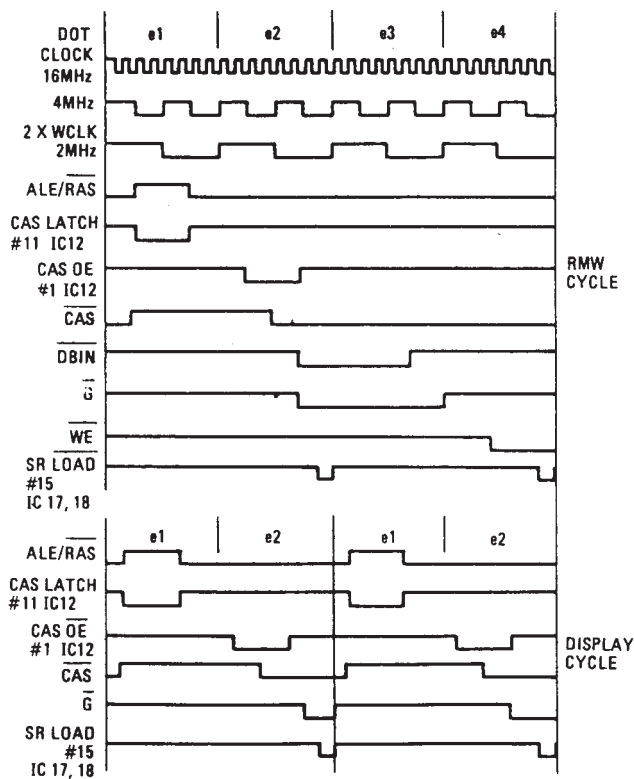


圖 3 時序圖，說明時序步驟與順序，欲知詳情請看全文

率的兩倍速率來工作的。且 7220 的所有內部時脈都是由這個信號分頻而來的，因為 16 個像素需要一個 16 位元的顯示記憶體。本設計所用的 2Xwclk 為 $2 \times 16 \times 65\text{ns} = 500\text{ns}$ 或 2MHz。

7220 顯示週期需要 2 個 2Xwclk 週期，而一個 RMW 週期需要 4 個 2Xwclk 週期。在 RMW 週期中，所發生的狀態依序說明如下。圖 3 中，4 個 2Xwclk 的時序各以 e1, e2, e3, e4 表示之，現在分別說明如下：

e1 — 7220 開始將顯示記憶體位址輸出到 16 條位址線上，ALE 信號變成低電位以表示此時的位址是適當可用的位址。這個信號（也叫 RAS 信號）會將低階 8 個位址位元選通到顯示記憶體內，並將高階 8 個位址位元門在 CAS 門鎖 IC12 內。

e2 — 7220 以三態緩衝方式激勵位址線，且於 DBIN 信號變成低電位時，表示目前正在執行 RMW 週期。CAS 門鎖的 OE 接腳會變成低電位，將 8 個行位址位元送到顯示記憶體，接著 CAS 線會變成低電位，將這個位址內的資料選通到顯示

RAM 晶片內。記憶器的 G 接腳會變成低電位以便讀取 7220 所要的資料。

e3 — 7220 由記憶體內讀取 16 位元資料並執行修改工作。

e4 — 記憶器的 G 接腳會變成高電位，以便以三態緩衝方式激勵 RAM 資料線。7220 會把它修改過的資料送到位址線上，而記憶器上的 WE 接腳變為低電位，以便將修改過的資料再寫回顯示記憶體內。然後修改過的資料會被載入視訊移位暫存器內。

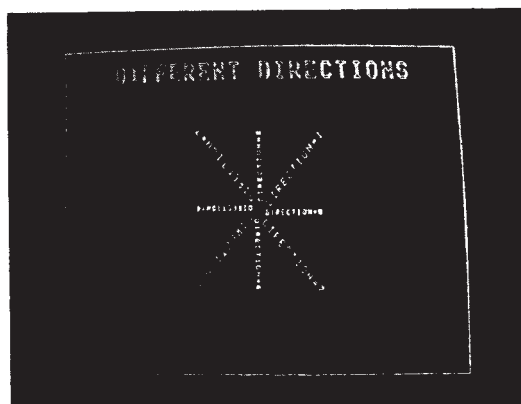
組成顯示週期的兩個 2Xwclk 時序內容說明如下（e1 和 e2 是用來表示時序圖上所標示的兩個 2Xwclk）。

e1 — 7220 開始將顯示記憶體位址輸出到 16 條位址線上，ALE 信號變成低電位時，表示此時的位址是適當可用的位址。這個信號會把低階 8 個位址位元選通到顯示記憶體內並把高階 8 個位址位元門在 CAS 門鎖 IC12 上。

e2 — 7220 以三態緩衝方式激勵位址線，且 DBIN 信號會保持在高電位以表示目前正在執行顯示週期。CAS 門鎖上的



主文 80 行 $\times$ 50 列的旋轉效果，字元集是由一個 8×8 的方塊中的一個 5×7 的點矩陣組成的。這個字元集包括整個 128 字的 ASC II 碼。



八個繪圖方向，這個影像是以將英文字串 DIFFERENT DIRECTION 定成放大兩倍再向 X 和 Y 起點位置移一個像素，操作模式定為反相的方式產生的

OE 接腳會變成低電位以便把 8 個行位址位元送到顯示記憶體，接著 CAS 線會變成低電位，以便將那個位址內的資料選通到顯示 RAM 晶片內。此時顯示記憶器的 G 接腳會變成低電位以便所讀取的資料能送到視訊移位暫存器，使所讀取的資料載入視訊移位暫存器內。

注意：視訊移位暫存器不管 7220 是在執行顯示週期或 RMW 週期，它每兩個 2Xwclk 時脈間隔會載入一批新資料。

在 RMW 週期時，載入移位暫存器的資料無法和適當的螢幕資料對應，若 7220 在主動視訊時序內存取螢幕 RAM 的資料，則會使螢幕上產生可見的瞬間雜訊。7220 有一種特殊功能，可使 RMW 週期在垂直同步 (Vsync) 與水平同步 (Hsync) (即遮沒) 的時間間隔內進行更改 VRAM 的工作，以防止螢幕顯示發生干擾。雖然視訊移位暫存器的資料在 RMW 操作時仍會被載入，但 RMW 這幾個週期是被限制在螢幕遮沒時間的 30 % 左右，所以不會看到錯誤資料。

當兩片或多片板子被裝入系統之內時，它們是以插入擴充接頭 P1, P2 的方法串在一起。這兩片板子也可由使用者指定其中任一片作為主板，擔任傳送 16MHz (Dot-Clock) 和 7220 視訊同步信號到指定的僕板上。P3 上的跳線只有在該片板子是主板時才會裝上去。由軟體啟動時，指定為僕板的 7220 會將它的時脈和主板的時脈同步，以便所有 7220 都可以同時工作。在一個系統內如果有三片 Grafex 板子時，它們就以上述方式在工作，每片板子可驅動紅藍綠彩色監示器其中的一隻電子槍。那些監示器的視訊上附載的混合同步信號，能夠接受綠色輸入信號的同步信號。RGB 監示器有外部同部輸入，可由 Grafex 板的擴充接頭來驅動。圖 4 和圖 5 是你自

己動手製作 Grafex 時所需要的資料。

如何規劃 7220

開機後，7220 必須由一串命令與參數來設定起始值，以便將它們組合成所有的顯示方式。通常這些命令與參數都是儲存在一個表內，這個表內有一個能提供作為參考的設定起始常式，當設定起始值後再把命令與參數送到 7220。

在主處理機 6502 與 7220 之間的資料流通通道是 7220 內部的一種先進先出緩衝器，命令與參數是由主處理機載入這個緩衝器內，由 GDC 的命令處理機從另一端移走。程式師在設計程式時必須小心謹慎，以免在 GDC 將緩衝器資料取走之前將資料送入緩衝器，即送入資料的速度太快，取出資料的速度太慢，而會造成溢位。基於上述原因，GDC 用一個狀態暫存器中的幾個位元來指示先進先出緩衝器究竟是滿了還是空的，以及主處理機何時可以讀取資料。參考表一，可知狀態暫存器的值會被映射至 Apple 的擴充槽內，且其狀態可隨時讀取出來。此暫存器的其他位元，會顯示垂直同步與水平同步視訊時序計數器的

表 一

位 址	讀 取	寫 入	視訊繼電器
第 0 台	狀 態 暫 存 器	參 數	Apple 視 訊 顯 示
第 1 台	先進先出	指 令	
第 2 台	狀 態 暫 存 器	參 數	Grafex 視 訊 顯 示
第 3 台	先進先出	指 令	

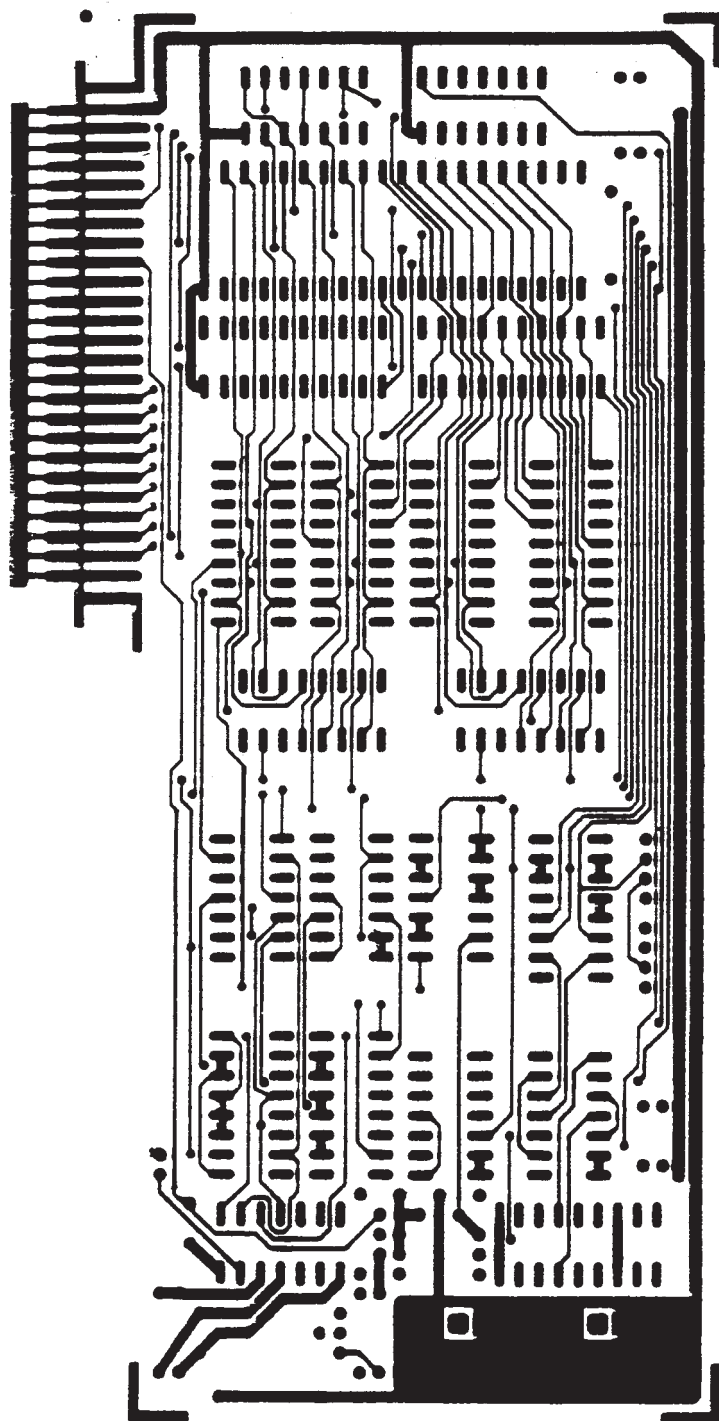
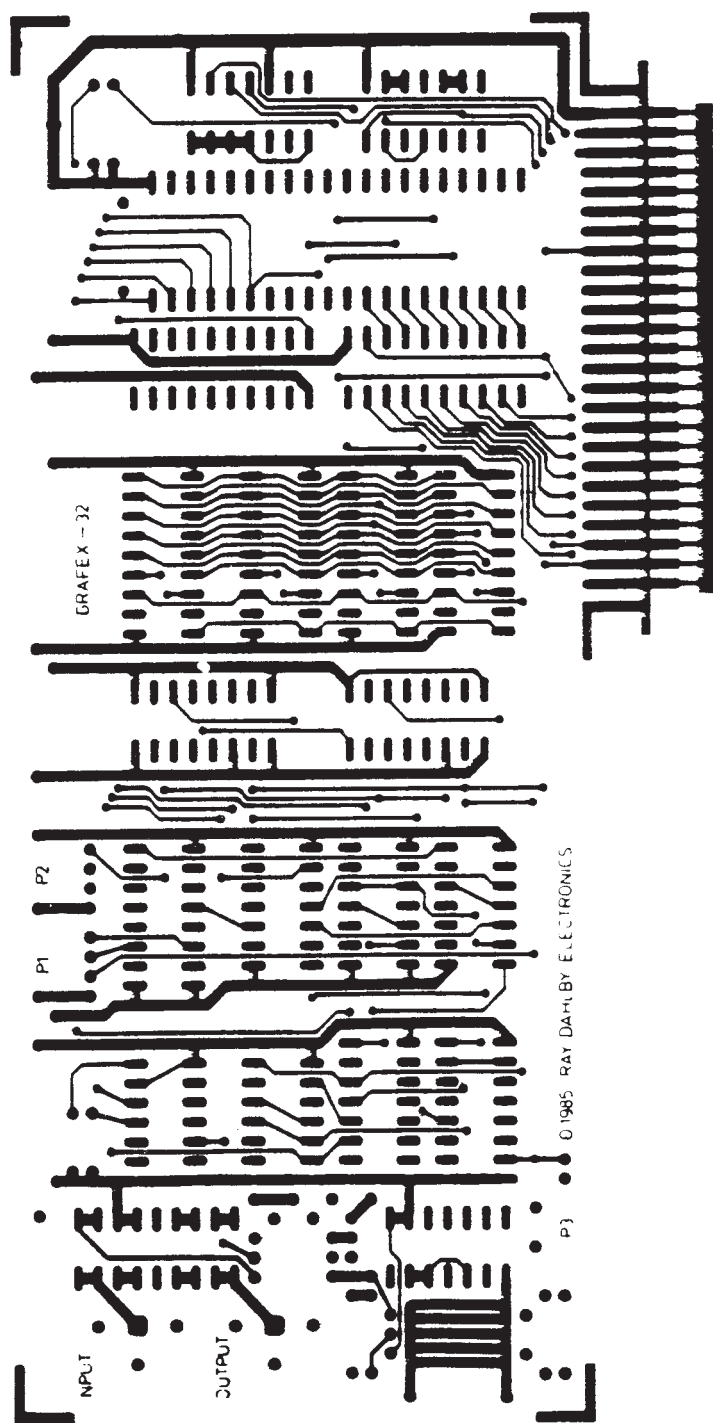


圖 4 印刷電路板



狀態，以便緩和螢幕捲動（Scrolling）的速度，以及其他需要對螢幕畫面速率做軟體同步的效果。

開機後第一個發出的命令是重置命令，這個命令是由先進先出前面的特殊硬體來執行的，以便確定GDC的內部暫存器先進先出緩衝器，以及命令處理機在設定啓始值命令與參數之前是重置在等待（即不工作）狀態。通常最好是在輸出每個位元組資料之前先檢查狀態暫存器是否有先進先出緩衝器已滿的狀態。開機時，狀態暫存器內的旗號並無意義，所以在讀取狀態暫存器資料或將其他命令或參數載入GDC之內時，必須先發出重置命令。

列表一所示是一個典型的設定啓始值的程式。此程式使GDC能產生 640 × 400 點組成的視訊，其視訊時序參數如表二所列。當GDC被設定成主控制器時，動態RAM會自動更新而成為致能狀態。而所

選的清晰操作模式會使GDC只有在遮沒時間內才能作更改VRAM的工作。且整個螢幕是於螢幕窗設定在記憶體頂端時被定義在記憶體映射繪圖範圍內。當這個程式執行的時候，7220開始由顯示記憶體內的開機隨機資料中將視訊輸出到系統CRT監示器上。

GRAFEX-32 電源與接地接腳

	+5volts	ground
IC1	40	20
IC2	14	7
IC3	14	7
IC4	14	7
IC5	14	7
IC6	14	7
IC7	14	7
IC8	14	7
IC9	14	7
IC10	16	8
IC11	20	10
IC12	20	10
IC13	9	18
IC14	9	18
IC15	9	18
IC16	9	18
IC17	16	8
IC18	16	8
IC19	16	8
IC20	16	8

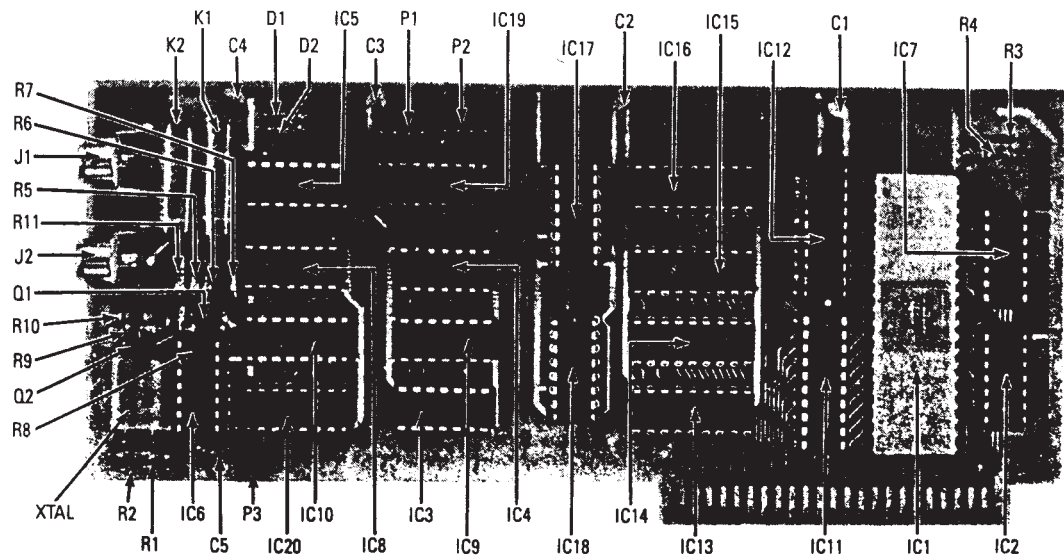


圖 5 零件位置如上圖所示，板子背面沒有插任何零件

表 二
視 訊 參 數

Active Line =	40μs
HFP =	7μs
HBP =	12μs
HSYNC =	6μs
Total Line Time	64μs
Active lines per video field =	200 lines
VFP =	30 lines
VBP =	16 lines
Vsync =	16 lines
Total lines per field =	262 lines
Video field rate = 1/64μs × 262 lines) = 59.637 Hz	

列表二所示為螢幕清除程式。這個程式會把顯示記憶體整個 32768 個位元組的資料，在 16 ms 之內或少於一個螢幕畫面所需要的時間之內清除為零。將列表二第 48 行改成 CPX # \$04，就可以將這個常式

修改成用來清除顯示記憶器的 128K 位元組內的資料。

列表三所示的最後一個程式，其命令與參數以 100 個垂直像素 × 100 個水平像素在螢幕中央畫一個長方形。

列表一

1L

```

1  IO      EQU  $FE
2          ORG  $0300
3          LDA  #$F2      ; SLOT7
4          STA  IO
5          LDA  #$C0
6          STA  IO+1
7  INIT    LDX  #00
8  LOOP    LDA  TABLE1,X
9          LDY  TABLE2,X
10         STA  (IO),Y
11         INX
12         CPX  #30
13         BNE  LOOP
14         RTS
15 TABLE1 DFB  $00,$1F,$26,$04,$1A
             , $0B,$1E,$C8,$40,$6F,$47,$28,$70,
             $00,$00,$00,$19,$4B,$00,$C0,$00,$
             46,$00,$7B,$FF,$FF,$23,$4C,$10,$6
             B
16 TABLE2 DFB  1,0,0,0,0,0,0,0,0,1
             , 1,0,1,0,0,0,0,1,0,0,0,1,0,1,0,0,
             , 1,1,0,1

```

列表二

!L

```

1 IO      EQU  $FE
2          ORG  $0300
3          LDA  #$F2      ;SLOT7
4          STA  IO
5          LDA  #$C0
6          STA  IO+1
7          JSR  FIFO      ;MAKE SURE FIFO IS EMPTY
8          LDY  #01
9          LDA  #$49      ;CURSOR COMMAND
10         STA  (IO),Y
11         DEY
12         LDA  #00
13         STA  (IO),Y
14         STA  (IO),Y
15         INY
16         LDA  #$4A      ;MASK COMMAND
17         STA  (IO),Y
18         DEY
19         LDA  #$FF
20         STA  (IO),Y
21         STA  (IO),Y
22         INY
23         LDA  #$0F      ;SYNC COMMAND
24         STA  (IO),Y
25         DEY
26         LDA  #$0F      ;FAST MODE
27         STA  (IO),Y
28         LDX  #$00
29 LOOP    LDY  #01
30         LDA  #$4C      ;FIGS COMMAND
31         STA  (IO),Y
32         DEY
33         LDA  #$02
34         STA  (IO),Y
35         LDA  #$FF
36         STA  (IO),Y
37         LDA  #$3F
38         STA  (IO),Y
39         INY
40         LDA  #$22      ;WDAT MODE
41         STA  (IO),Y
42         DEY
43         LDA  #$FF
44         STA  (IO),Y
45         STA  (IO),Y
46         JSR  FIFO      ;MAKE SURE FIFO IS EMPTY
47         INX
48         CPX  #$01      ;CHANGE TO #$04 FOR 128K
49         BNE  LOOP
50         RTS
51 FIFO    LDY  #00
52 WAIT    LDA  (IO),Y
53         AND  #$04
54         BEQ  WAIT
55         RTS

```

列表三

1	IO	EQU	\$FE	
2		ORG	\$300	
3		LDA	#\$F2	;SLOT7
4		STA	IO	
5		LDA	#\$C0	
6		STA	IO+1	
7		JSR	FIFO	;MAKE SURE FIFO IS EMPTY
8		LDY	#01	
9		LDA	#\$23	;WDAT MODE
10		STA	(IO),Y	
11		LDA	#\$7B	;PATTERN RAM
12		STA	(IO),Y	
13		DEY		
14		LDA	#\$FF	
15		STA	(IO),Y	
16		STA	(IO),Y	
17		INY		
18		LDA	#\$49	;CURSOR COMMAND
19		STA	(IO),Y	
20		DEY		
21		LDA	#\$81	
22		STA	(IO),Y	
23		LDA	#\$17	
24		STA	(IO),Y	
25		LDA	#\$00	
26		STA	(IO),Y	
27		INY		
28		LDA	#\$4C	;FIGS COMMAND
29		STA	(IO),Y	
30		DEY		
31		LDA	#\$40	
32		STA	(IO),Y	
33		LDA	#\$03	
34		STA	(IO),Y	
35		LDA	#\$00	
36		STA	(IO),Y	
37		LDA	#\$63	
38		STA	(IO),Y	
39		LDA	#\$00	
40		STA	(IO),Y	
41		LDA	#\$63	
42		STA	(IO),Y	
43		LDA	#\$00	
44		STA	(IO),Y	
45		LDA	#\$FF	
46		STA	(IO),Y	
47		LDA	#\$3F	
48		STA	(IO),Y	
49		LDA	#\$63	
50		STA	(IO),Y	
51		LDA	#\$00	
52		STA	(IO),Y	
53		INY		
54		LDA	#\$6C	;FIGD COMMAND
55		STA	(IO),Y	
56		RTS		
57	FIFO	LDY	#00	
58	WAIT	LDA	(IO),Y	
59		AND	#\$04	
60		BEQ	WAIT	
61		RTS		

製作一個能與6502同時工作的

Z80H界面卡

Z80H卡的功能

在 6502 系統和 6800 系列的個人電腦中，都配備有 Z80 界面卡，在這兩系列的電腦中，由於都是採用使主機板的 CPU 停止工作而改成為 Z80 來工作的方式，所以主機板的 CPU 會變成單純的字符端末機。

現在之所以要製作能使 6502 和 Z80 可以同時工作的 Z80H 界面卡，是因為如果使用 Z80H 的話，就可以使用 CP/M，而如果用 Z80H 能與 6502 同時工作的話，則是可以同時控制 CP/M 系統中所用的兩面倍密度，倍磁軌的小型磁碟機，和印表機緩衝器等。

圖 1 是本界面卡的方塊圖，表 1 是功能的一覽表。

由於使用本界面卡便能同時工作，故可以執行多種功能。此外，在本文所討論的系統中，CP/M 的各檔案是儲存在 5 英寸 1MB 容量的磁碟內。（但是，ROM-DISK 除外），除了可使用上述的磁碟（Disk）之外，也可以使用 Apple 原有的軟

性磁碟。此外，除了表 1 所舉之功能之外，尚有其他功能。

方塊圖（圖 1）的同時工作時序電路，是本界面卡的特點，容後再詳細敘述。

為了使 DRAM 是在 8MHz 動作之故，最慢的存取時間也要有 100ns。此外，ROM 有 32KB，但這是為了要使 CP/M 能置於界面卡上的緣故，所以一旦此界面卡開始動作，便能迅速顯示 CP/M 的開啓訊息（open message）。而這種富於上述多項功能的界面卡，當然在設計考量上，是和以往的 Z80 界面卡不同。

但是讀者在製作時，仍需注意到下列各點：

(1) 原則上，必須有可執行和 Apple 不同的 CP/M (V2.2) 的系統，以及 ROM 的燒錄。

(2) 在 Apple 的插槽 3 (slot 3) 上，要有 80 行卡 (80 column card)。

(3) 在本文中所使用的 (support) Z80 FDD 是兩面倍密、倍磁軌型（若是其他的 FDD 則必須修改 BIOS 或電路）。

〔表 1〕 能夠和 6502 同時工作的 Z80H 界面卡的功能表

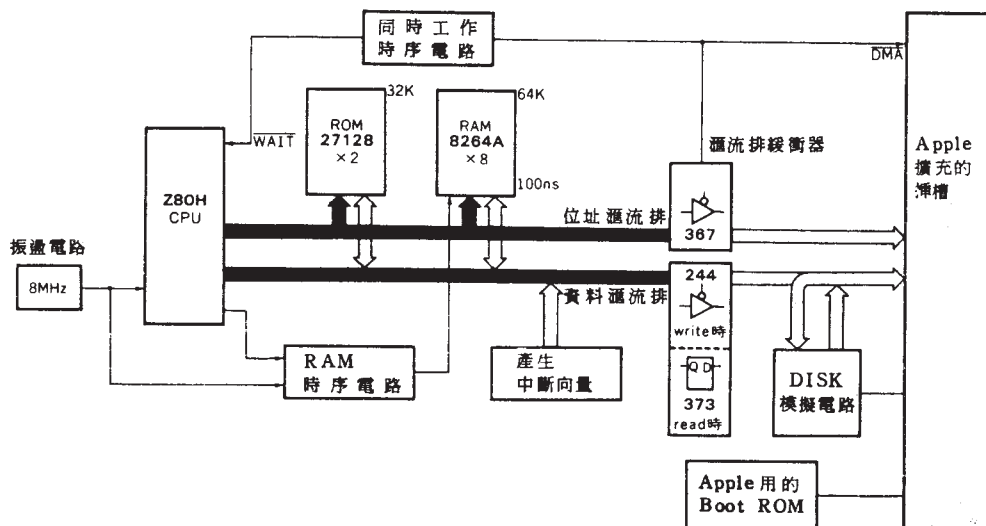
	採用 8MHz 時脈 (clock) 的高頻來執行工作
○	可使 6502 和 Z80H 同時工作
	使用 CP/M V2.2
	所用 RAM 為 64K
○	將 CP/M 的載入程式燒錄到 ROM 內使執行速度加快
◎	Apple 的印表機緩衝器 (printer buffer)
◎	Apple 的 5 英吋 • 雙面 • 倍密度 • 倍磁軌 • FDD 控制系統
△ ◎	Apple 的 DISK II 控制系統 (可利用 DISK II)
△ ◎	Z80 的 DISK II (Simulate)
△ ◎	供其他的 Apple 以緩衝器來做接地 (buffer ground) 處理

○：本界面卡專有的特點

◎：在 Z80H 和 6502 同時工作的條件下才有的功能

△：本專題中沒有軟體支援 (support)

〔圖 1〕 是 Z80H CPU 界面卡的方塊圖



(4) 本 Z80 卡不能太大。如果不考量界面卡的空間的話，則 Apple 的上蓋就無法關上（假如只是實驗性機器的話，則不必考慮上蓋能不能關上的問題）。

(5) 由於銲接的地方很多，所以接線時必須很謹慎。

(6) 採用以 8MHz 為工作頻率的 Z80H 時，由於工作時序不夠長，所以最好是改 6MHz 使 Z80H 版本（Version）的工作時序加長。

(7) 由於有關 CP/M，以及有關 Z80 原理及應用的解說，本社另有數本專書討論，所以本文中將它省略掉（至於 Z80H 的 CPU 探討，請參閱本社所出版之 Z80 微電腦系統原理與設計一書）。

硬體電路的設計

Z80H和6502同時執行的時序

圖 2 (a) 是 Z80H CPU 界面卡的電路圖。FDC 部份是應用於別的界面卡上（後述）。同時工作可以採用在 Z80H 的 I/O 空間插入 Apple 的 64K RAM，並以 Apple 的模式 2 (MODE 2) 中斷方式來執行工作。

假如 Z80 把 Apple 的 RAM 空間當做 I/O 空間的話，則 Z80 只能存取 I/O 空間的資料，而且不能和 Apple 競爭匯流排使用權。還有，Z80 不存取 I/O 記憶空間的資料時，Apple 的匯流排主權是由 6502 來控制，故 6502 此時可以動作。

圖 2 (a) 中的同時工作時序電路，就是採用上述方式的應用電路。這一部份是本界面卡的特點，所以請務必詳細閱讀。

圖 3 是同時工作的時序圖。1A_b LS74 是以 ϕ_3 的降緣來門鎖 (latch) Apple 的 ϕ_1 週期。1A_b 的 Q 輸出經過

ϕ_0 之相位位移變成 1C_b 之 D 輸入，最初的狀態雖是“H”，但是一旦 IORQ 為“L”時（Z80 存取 I/O 空間的資料時），則會降為“L”。同時送到 Z80 的 $\overline{\text{WAIT}}$ 信號，也會降為“L”，Z80 會進入等待週期 (WAIT CYCLE)。

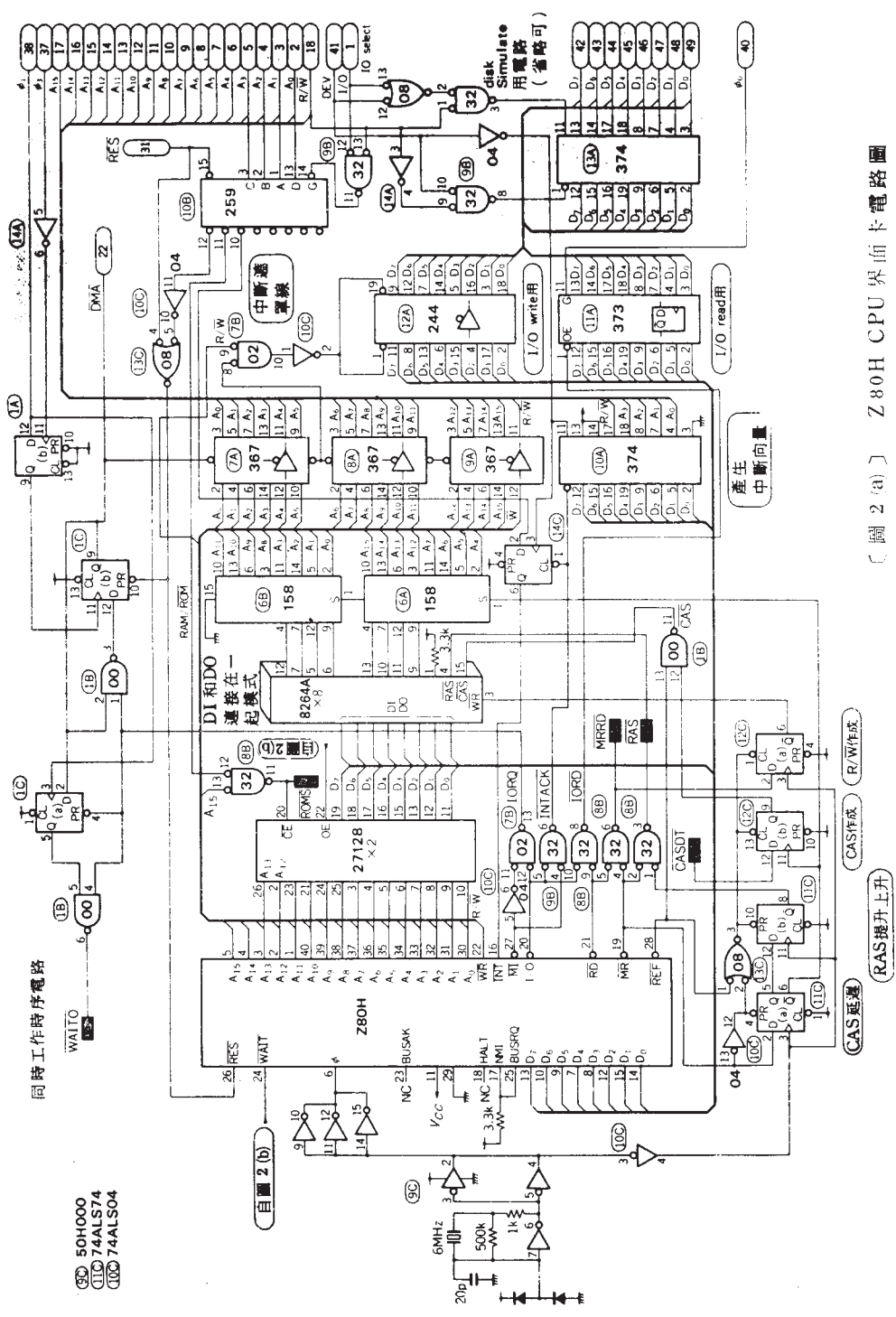
在此種狀態時，若 1A_b 的 Q 輸出上升為“H”的話，則 1C_b 的 Q 輸出的 DMA 則降為“L”。 $\overline{\text{DMA}}$ 信號也會被送到 Apple 的匯流排線上，6502 放掉匯流排的控制權，而此時 Z80 的位址緩衝器 (7A, 8A, 9A) 也受此 DMA 信號控制而得到控制權。在持續的 ϕ_0 週期中，Z80 要完全作有對 Apple 的 RAM 存取資料的控制權。

其次， ϕ_1 上升緣 (edge) 時，則 1C_a 的 Q 輸出會降為“L”， $\overline{\text{WAIT}}$ 信號也會被解除。一旦， $\overline{\text{WAIT}}$ 信號解除後，會延遲大約 200ns 左右，而使 IOQR 信號結束工作（IORQ 降為“L”）。讀取週期時可以讀入門鎖在 11A 的資料，這個時候，1C_a 的 Q 輸出就可以再次成“H”。

其後，1A_b 的 Q 輸出上升的脈衝 (pulse) 切掉了 $\overline{\text{DMA}}$ ，則 $\overline{\text{DMA}}$ 就會被解除，Apple BUS 也會再度被 6502 佔用。因此，如果要執行這個動作，則必須以 2.5 MHz 頻率以上的時脈信號 (clock) 使 Z80 動作。

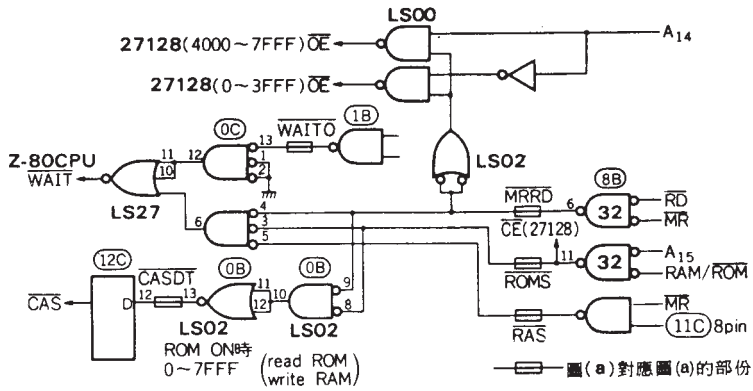
此外，儘管 Z80 可連續向 I/O 做存取資料的工作，但是還是必須每隔 1 個時脈週期 (Apple 的)，才能作 I/O 存取的動作。換言之，Apple 因 $\overline{\text{DMA}}$ 而停止工作的比例至少有 1/2。雖然 Z80 可以一面顯示 80 行，一面執行之，而在 6502 上並不能執行存取磁碟 (disk) 資料的工作，但是至少也不會發生資料遺失的狀況。

由6502向Z80的中斷

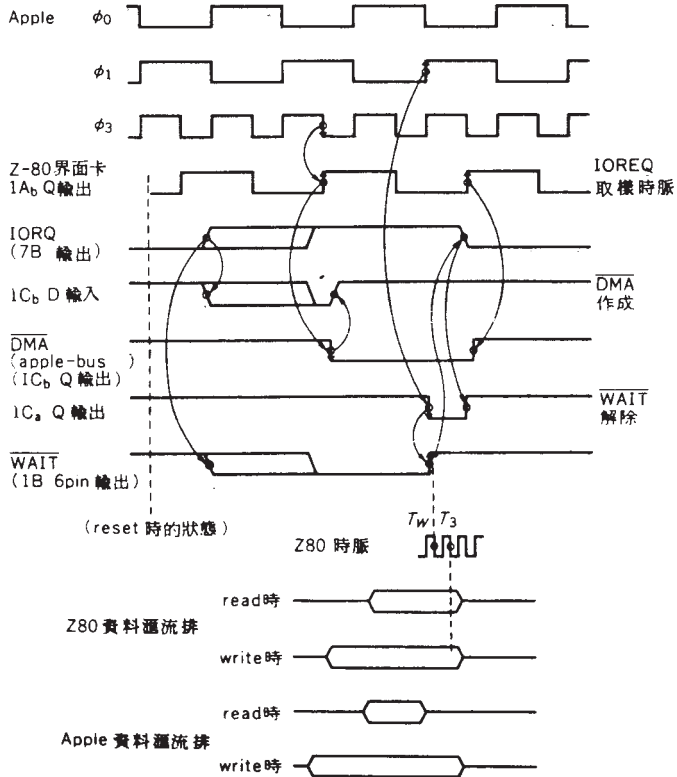


〔圖 2 (a)〕 Z80H CPU 界面卡電路圖

〔圖 2 (b)〕 改良電路圖



〔圖 3〕 同時工作的時序圖



在前面我們已經敘述過 Z80 如何存取 Apple 的 RAM，現在所要介紹的是，如何自 6502 呼叫 Z80 的方法。

圖 4 是通往 Z80 的中斷電路，在此電路中把 LS259 之 Q_5 輸出當做中斷之遮蓋信號 (MASK) 使用。如果寫到 Apple 上之 $CN \times BH$ 位址的話，則便是設定在可以中斷的狀態。禁止中斷時，則寫到 $CN \times AH$ 位址內。重置時是禁止中斷的。

Z80 的中斷是採用 MODE 2。6502 是在 $\overline{DEVSEL}$ ($C080H \sim C0FH$) 信號上升後，開始向 Z80 中斷的。

接收到中斷信號後，Z80 自資料匯流排讀入中斷向量。而這個向量是以 IC 編號為 $10A$ 之 LS374 來門鎖 6502 的位址 $A_0 \sim A_3$ 。本界面卡各中斷向量的功能，如表 2 所示。中斷向量共有 32 種，但是目前只

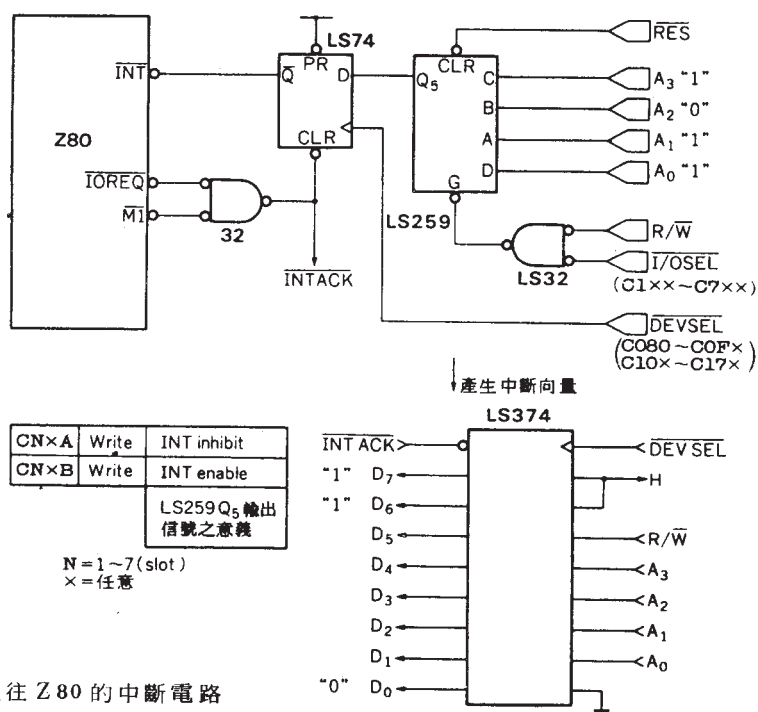
使用 3 種。

INT 信號在接收到中斷向量的同時就會被清除。

RAM 電路

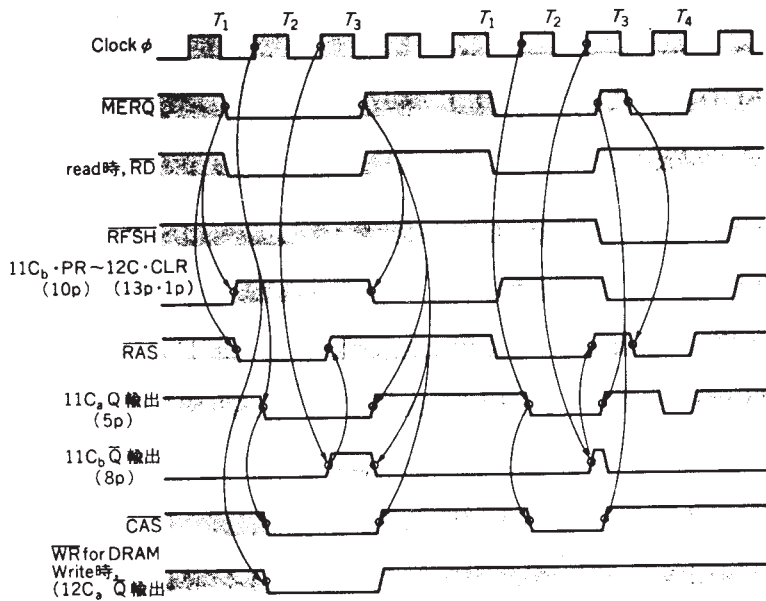
64K DRAM 的輸入端子和輸出端子是連接在一起，以 Early Light Mode 來使用。

爲了要使 $\overline{RAS}$, $\overline{CAS}$ 信號工作，故把 $11C_a$ 當做 CAS 延遲下降用， $11C_b$ 當做 $\overline{RAS}$ 上升信號來使用， $12C_b$ 當做 $\overline{CAS}$ 信號來使用， $12C_a$ 當做讀／寫之 $\overline{WR}$ 信號來使用。詳細的時序，請參閱圖 5。如果要以 8MHz 來作為 Z80 工作頻率的話，則必須使用存取時間在 100ns 之內的 DRAM。如果 100ns 的 DRAM 很難購得的話，則把 Z80H 之時脈改為 6MHz 之後，就可以使

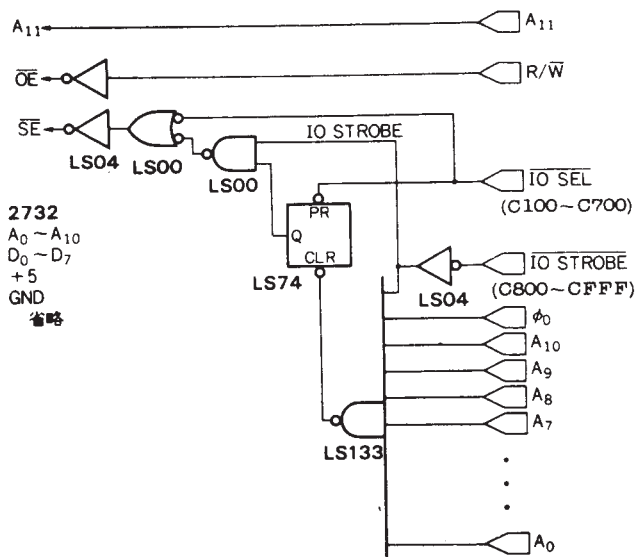


〔圖 4〕 通往 Z80 的中斷電路

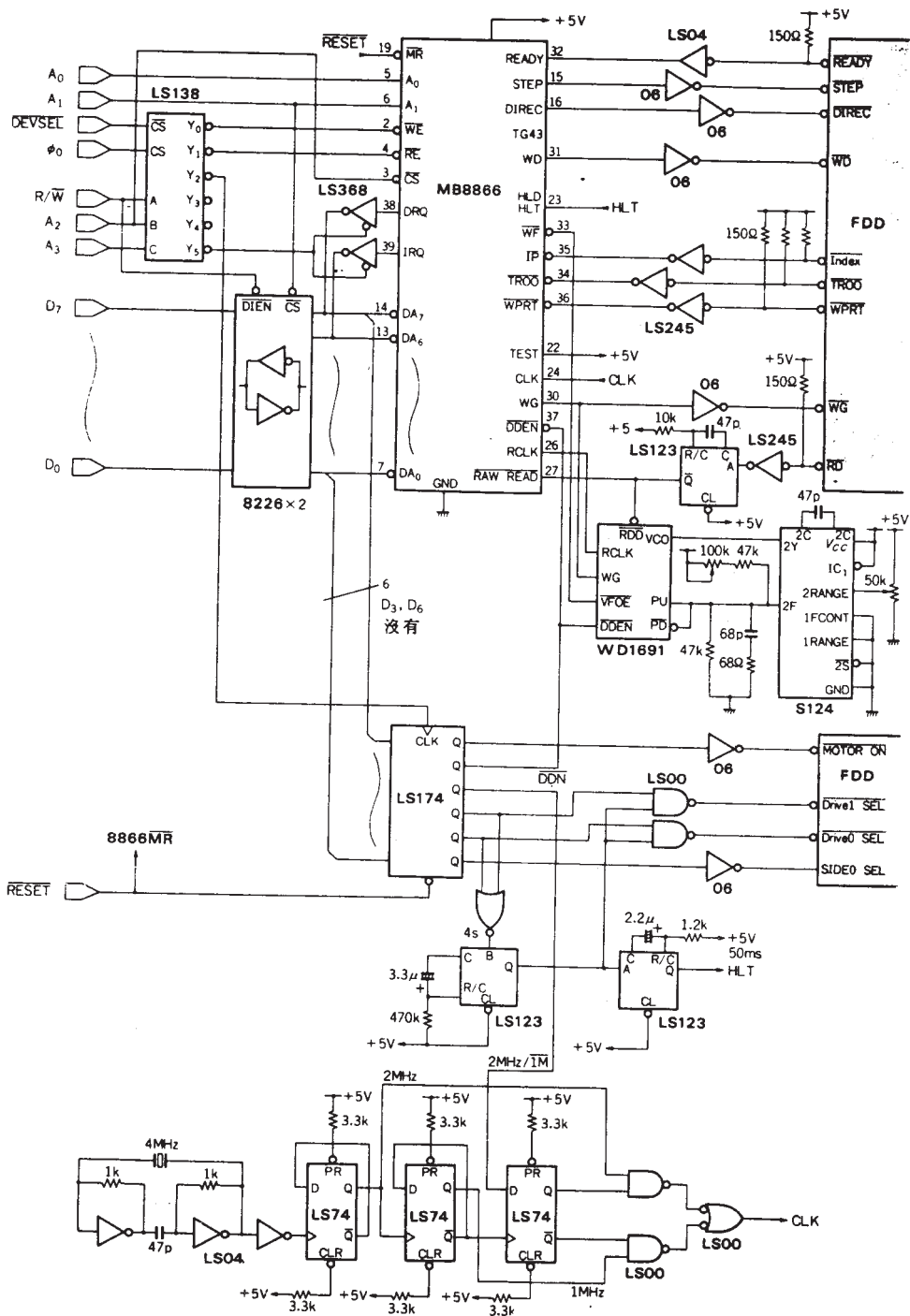
〔圖 5〕 RAM 周邊的時序圖



〔圖 6〕 Apple 用 ROM 的電路圖



〔圖 7〕 軟式磁碟控制電路



A ₀ ~A ₃ ×	產生向量		功 能 分 配	現行使用狀態
	Read時	Write時		
0	E0	C0	界面卡中斷 DISK Ready中斷(命令致能)	○(只有Write)
1	E2	C2		×(只有Write)
2	E4	C4	DISK Ready中斷(磁碟命令終了)	○(只有Write)
3	⋮	⋮	} 未定義	} ×
⋮	⋮	⋮		
F	FE	DE		

使用=○ 未使用=×

用120ns之DRAM。此外，由於在DRAM中，有些東西的實力超過規格以上，所以，你可以在這些零件中，發揮出業餘者的實力。

APPLE端的ROM電路

這個Z80H界面卡是從Apple那邊鍵入PR#N指令(N=2~7)，再按RET鍵來啟動的。

因此，①靴帶式載入程式(bootstrap program)、及②FDD控制程式(FDD control program)是放在1個2732的EPROM內。

圖6是這個ROM的配線圖。在2732的前面2KB中，由於是以反覆執行靴帶式載入程式8次的方式將256B放進去，因此，無論把Z80界面卡放在那一個插槽，都能夠啟動。而在C800H~CFFFH這個範圍內，載入FDD的控制程式。而靴帶式載入程式與FDD控制程式的交換工作只要用A₁₁就可以完成了。有關這個電路，請參照Apple參考手冊。

軟式磁碟機控制電路

這個部份是由另外一個界面卡來執行的，這個界面卡是和Z80 CPU界面卡不

同。這個界面卡是固定放在插槽7(slot 7)，圖7的電路是FDD的控制電路(有關於磁碟機詳細的動作情形，請參考本社出版之“Z80微電腦原理與設計”一書)。在FDD上使用8866是因為它是通用零件之故，將來也會有所改良。一般而言，磁碟機的馬達都是一直在ON的狀態，當然啦！不喜歡這樣做的人，也可以適當地變更6502的FDD控制程式。

軟體的設計

前面我們介紹過硬體的設計，下面我你將為各位讀者介紹軟體方面的設計。

本系統中備有下列的程式：

- 列表1(LIST 1) Apple用靴帶式載入程式。
- 列表2(LIST 2) Apple上的FDD控制／印表機緩衝程式(print buffer program)。
- 列表3(LIST 3) Z80用BIOS程式。
- 列表4(LIST 4) Z80用(CP/M)監督程式(monitor program)。
- 列表5(LIST 5) Z80 ROM使用的SUBMIT程式。
- 列表6(LIST 6) 監督程式IPL(

Monitor IPL)

- 列表 7 (LIST 7) 1K 磁區格式規劃傾印程式 (Sector format dump list)。

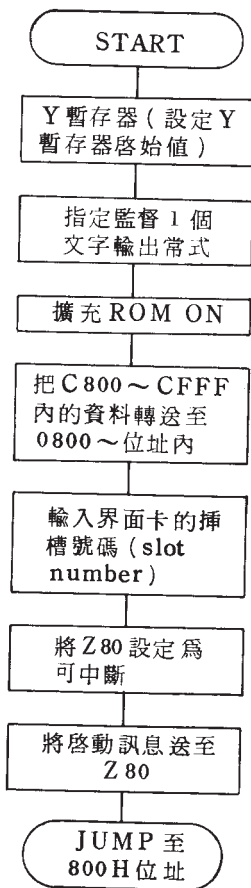
APPLE用靴帶式載入程式

圖 8 是引導程式流程圖，這個圖是假設 Z80 Card 是接在第 2 插槽，而由 C 200H 這個位址開始。

此外，這個程式的註解部份，是放在列表 2 的 ORG 100H 這個位址（有些部份會有所不同）。

[圖 8]

Apple 用靴帶式載入程式之流程圖



FDD控制及印表機緩衝程式

這個部份雖然是放在 C800~CFFFH 區的 ROM 中，但是，卻是經由前面的靴帶式載入程式，轉送到 800H 記憶區以下之記憶區內。

Apple 上之特別區中，尤其需要注意的是 300H~302H 區之文字輸出表 (table) (302H 是印表機用 1 文字 buffer)，及 0~1FH 區之磁碟控制表。尤其是 0~1FH，如果在寫入時不小心的話，FDD 的速度會變成很快，而磁碟內的內容也會有消失之虞。

有關 1 個文字鍵入到 Z80 的程序，請參閱圖 9。

Z80H用的設定初始化載入程式

設定中斷向量的 Jump 目標，把 CP/M 轉送至 E 000H 區以下，執行設定 MODE 2 中斷等的啓始值處理。

這個程式是放在 Z80 界面卡上之 27128 的 0H~100H 區。圖 10 是 ROM 的位址分配圖 (address map)。有關 100H 以下的部份，容後再述，我們先由 CP/M 開始看起。

BIOS常式(CP/M用)

首先我們將 BIOS 的特點綜合理列示如下：

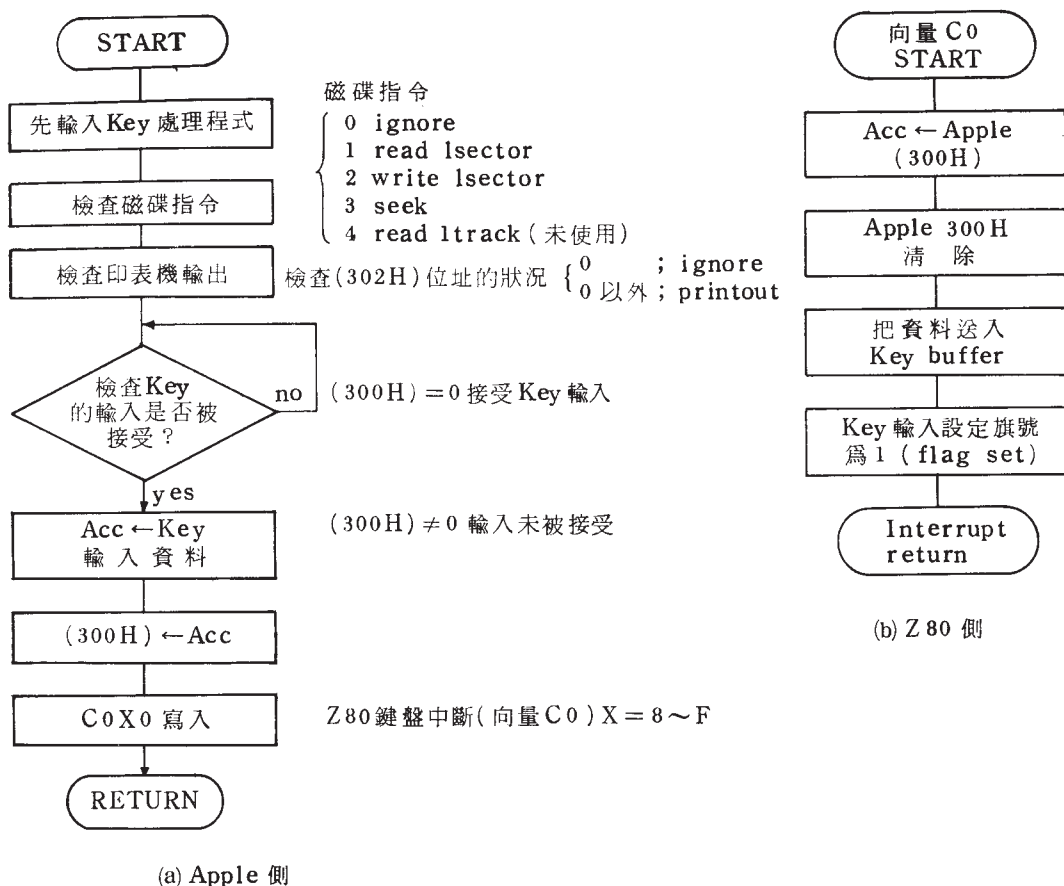
① CP/M 的 Z80 界面卡，是把 Z80 上之 2000H~7FFFH 記憶區，當做 ROM DISK 的記憶區來使用的。

② 以 Z80 直接操作 Apple 的 80 行界面卡 (顯示迅速)。

③ 擁有先執行 Key 輸入處理的功能。

④ 把軟式磁碟一個磁區的記憶量為

〔圖 9〕 一個文字輸入常式的流程圖



256 位元組 (Byte) 者，和 1K Byte 者，分別以不同的磁碟機 (drive A, B) 來命名。但是，實際上的 drive 則為相同 (使用 1K Byte/Sector 的話，可以縮短資料轉送時間)。

首先，雖然是①的 ROM DISK，但是，一個磁軌中的 8K Byte = 8 個磁區，這是設定於 1K Byte 的目錄表 (列表 3 之 DPBO 的資料表)。block 0 (2000H ~ 23FFH) 是目錄表記憶區。4000H ~ 7FFH 是放在第 2 個 27128 ROM 內，相當於

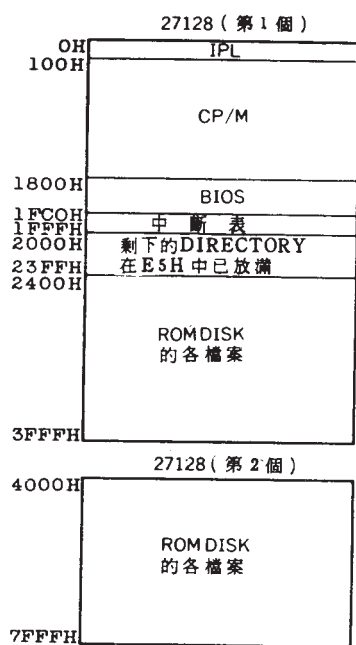
第 8 ~ 17 區。用來儲存任何一個 CP/M 檔案。

在第 7 個 block 中載入原始監督程式 (然後再轉送到 C000H ~ C3FFH 位址)，在程式開始執行工作時，監督程式是放在列表 4 內 (LIST 4)。

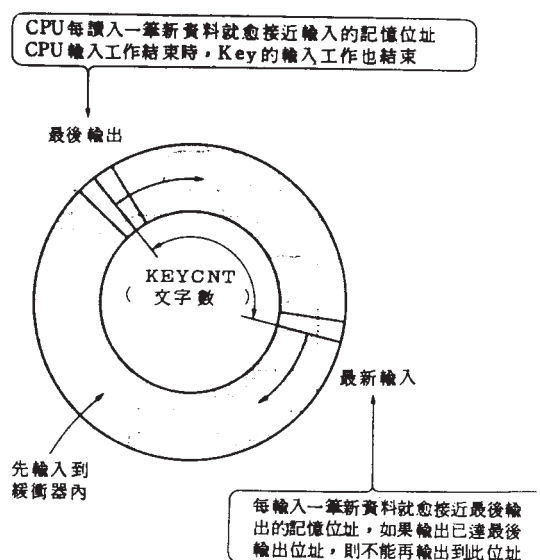
第②項的顯示常式，是由 Z80 存取 80 行卡的資料。請參考 Z80 之 I/O 空間之存取方法。

第③項的 Key 輸入的優先處理方式，相當於 LIST 的 CONIN, INCHR, IRQ-

〔圖 10〕 Z80 用 ROM 位址圖



〔圖 11〕 先執行 Key 輸入處理的環形緩衝器



〔表 3〕 邏輯、磁碟編號的分配

邏輯 磁碟編號	分 配
0 (A:)	ROM 磁碟 (27128)
1 (B:)	RAM 磁碟 (本系統中未使用)
2 (C:)	1K sector drive 1 (YD480)
3 (D:)	1K sector drive 2 (YD280)
4 (E:)	256 sector drive 1 (YD480)
5 (F:)	256 sector drive 2 (YD280)

WO, IRQ 01 的常式。這就是 RING BUFFER。請參考圖 11。

第④項的處理以 SECTTRAN 以下的流程常式執行。但是, EMADDR~WRITE 間是採用 ROM DISK 的讀/寫常式。

如果是 256 Byte/Sector 的磁碟和 1K Byte/Sector 的磁碟的話, 則由於 skew table 不同, 所以要計算所使用的記憶位址。而 sector skow, 是由 PWO-PER 程式來計算其記憶位址。

此外, 由於 CPU 界面卡的縮小, 所以, 如果省略 ROM DISK 的話, 則邏輯磁碟編號 (表 3) 能變換如 0 → 2, 1 → 3, ……般, 並能對照。

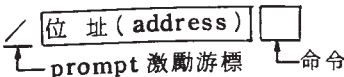
監督控制命令

列表 4 的監督控制命令是 CP/M 的一個檔案。如果是 board ON CP/M 的話, 則最好把它當做 ROM 中的檔案。假如將它當作檔案 (file) 的話, 則把列表 6 (檔案名 "MONIPL") 放置在第 100 個磁區起的磁區內, 而列表 4 (LIST 4) (檔案名 "MONITOR") 則放置於 200H 磁區名 "MONITOR" 則放置於 200H 磁區內。在下述的監督控制命令中, 不需要列

表 6 的 IPL。

監督控制命令的使用方法如下：

命令的一般輸入格式為



命令中可包括文數字，但第一個字必須是英文字母，不能為數字。表 4 是歸納

整理後的命令表，命令是以按 **RETURN** 鍵來結束。

依照 **/ ADDRESS SPACE** 的格式，可以向任意的記憶位址作傾印 (dump) 的工作。有些監督控制命令即使是在沒有使用 CP/M 的情況下，也可以使用，有關這些容後再述。

〔表 4〕 Z80 用監督控制表

命 令	意 義	說 明〔前置位址（必要=R，不必要=N）〕	
O	Set ROM	把 Z80 前面 32K 設定於 ROM 位址內，把後 32K 設定於 RAM 位址內	N
A	Set RAM	把 Z80 用位址的 64K 設定於 RAM 位址內	N
H	Help	顯示命令目錄	
M	Modify memory	緊接著命令之後，以 (SPACE) 命令向較高位址前進將位址的號碼增量。以 / (SLASH) 命令向較低位址減量	R
G	Go to	跳躍 (JUMP) 至前置位址（只以 G 命令回到 CP/M）	R
L	Load from Apple	把 Apple 記憶體內的資料 (4100H ~) 移至 Z80 上的記憶體 (8100H) 內	N
X	I/O dump	顯示 I/O 空間 (Apple 記憶體) 的內容	R
SPACE	Dump	顯示傾列記憶內容	R

獨立型的監督控制命令

這種獨立型 (stand-alone) 的監督控制命令，是製作 Z80 界面卡時，專門用來檢查界面卡工作狀態，這個監督控制命令可以在沒有 CP/M 的情況下工作。

如圖 12 所示，這種方式在實際的製作上幫助很大。

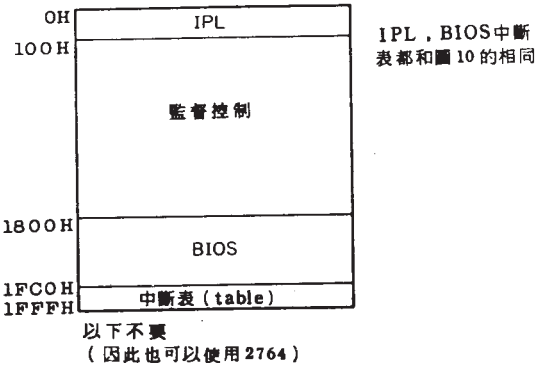
CP/M 的 ROM 移植法

如果我們第一次所接觸到，則必須是擁有 CP/M 能使用，而且附有能燒錄 ROM 的電腦機器的環境中為前提。

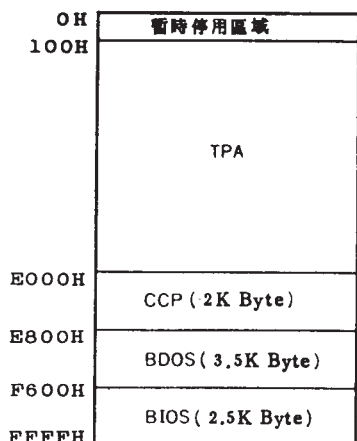
如圖 10 所示，將各程式放在 BIOS 以外的位置，所使用的是標準的格式。

〔圖 12〕

獨立型監督控制命令的 ROM 位址表



程式的記憶



列表 5 是將各程式放在 ROM 內的 SUBMIT 命令。把移動過的 CP/M 移至 CPM63.COM 檔案 (File) 中。由於本界面卡的 BIOS，稍微超過 1.5K Byte，所以變為「63K CP/M」，CP/M 的移動，請參考圖 10 和圖 13，用瞬變命令 (transient command) MOVCPM 等命令來執行。這部份請參考 CP/M 原理一書。

在這個例子中，100H～1FFH，
20C0H～20FFH區是CP/M的一部份。

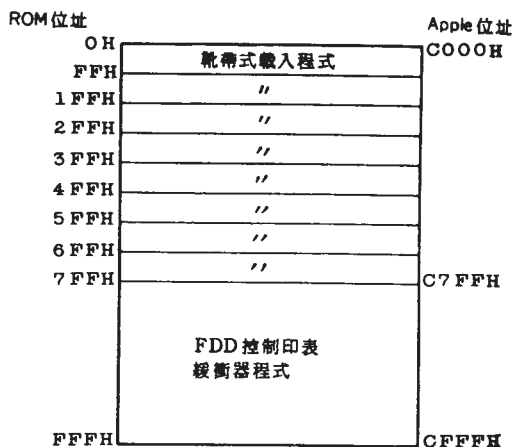
此外，最後的APLOAD檔案，是把以前所放進去的程式，移至ROM WRITE用的區域（area）的程式。這一部份，請各位讀者配合自己的系統來做。

APPLE用的/ROM燒錄方式

其配置方法如圖 14 所示。

靴帶式載入程式容量是在 256 位元組以下，所以，反覆執行 8 次的話（Apple 共有 8 個 slot），則在任一個 slot 上的 Z80H 界面卡也會啟動。

Apple 用 ROM (2732) 分配圖



後記

由於Z80H的8MHz工作頻率相當快，所以，一旦執行TYPE命令的話，畫面的滑動十分快速。但是，另一方面，倍軌，小型磁碟的存取也會有略慢之感。爲了要改良這個缺點，最好是由BIOS中叫出放在磁碟控制程式內的磁軌讀入常式，將這個讀入常式稍爲修改，以配合本系統的工作。

此外，由於是以 Apple 來控制磁碟機 (DISK DRIVE)，所以只要變更 FDC 程式，就能使用 Apple-DOS 常式之 Apple-DISK。而 Z80 方面，也可以裝 6502 的模擬器程式 (Simulator)。使用這個模擬程式，對 Apple 的程式解析會有所幫助。

另外，有一點是讀者們必須考慮到的，在電路上必須考慮到 Apple 的磁碟模擬器程式。看了上述的介紹，各位讀者難道不想向軟體挑戰嗎？

列表 1 APPLE 用靴帶式載入程式

*C200LLLLL

C200-	A0 00	LDY	#\$00	C249-	0A	ASL	
C202-	EA	NOP		C24A-	29 70	AND	#\$70
C203-	A5 36	LDA	\$36	C24C-	8D F4 0C	STA	\$0CF4
C205-	D0 0A	BNE	\$C211	C24F-	A5 19	LDA	\$19
C207-	A9 F0	LDA	#\$F0	C251-	85 01	STA	\$01
C209-	85 36	STA	\$36	C253-	A9 00	LDA	#\$00
C20B-	A9 FD	LDA	#\$FD	C255-	85 00	STA	\$00
C20D-	85 37	STA	\$37	C257-	A0 0F	LDY	#\$0F
C20F-	D0 08	BNE	\$C219	C259-	91 00	STA	(\$00).Y
C211-	A9 1B	LDA	#\$1B	C25B-	88	DEY	
C213-	85 38	STA	\$38	C25C-	A2 00	LDX	#\$00
C215-	A9 FD	LDA	#\$FD	C25E-	48	PHA	
C217-	85 39	STA	\$39	C25F-	68	PLA	
C219-	8D FF CF	STA	\$CFFF	C260-	91 00	STA	(\$00).Y
C21C-	AD FF CF	LDA	\$CFFF	C262-	48	PHA	
C21F-	A9 C8	LDA	#\$C8	C263-	68	PLA	
C221-	85 01	STA	\$01	C264-	E8	INX	
C223-	A9 08	LDA	#\$08	C265-	D0 FB	BNE	\$C262
C225-	85 03	STA	\$03	C267-	A0 0B	LDY	#\$0B
C227-	A9 00	LDA	#\$00	C269-	91 00	STA	(\$00).Y
C229-	85 00	STA	\$00	C26B-	A2 00	LDX	#\$00
C22B-	85 02	STA	\$02	C26D-	EA	NOP	
C22D-	A2 08	LDX	#\$08	C26E-	EA	NOP	
C22F-	B1 00	LDA	(\$00).Y	C26F-	AC F4 0C	LDY	\$0CF4
C231-	91 02	STA	(\$02).Y	C272-	BD 06 08	LDA	\$0806.X
C233-	C8	INY		C275-	29 FF	AND	#\$FF
C234-	D0 F9	BNE	\$C22F	C277-	F0 0E	BEQ	\$C287
C236-	E6 01	INC	\$01	C279-	8D 00 03	STA	\$0300
C238-	E6 03	INC	\$03	C27C-	99 80 C0	STA	\$C080.Y
C23A-	CA	DEX		C27F-	AD 00 03	LDA	\$0300
C23B-	D0 F2	BNE	\$C22F	C282-	D0 FB	BNE	\$C27F
C23D-	20 58 FF	JSR	\$\$\$F58	C284-	E8	INX	
C240-	BA	TSX		C285-	D0 EB	BNE	\$C272
C241-	BD 00 01	LDA	\$0100.X	C287-	4C 00 08	JMP	\$0800
C244-	85 19	STA	\$19	C28A-	FF	???	
C246-	0A	ASL		C28B-	FF	???	
C247-	0A	ASL		C28C-	FF	???	
C248-	0A	ASL		C28D-	FF	???	

列表 2 APPLE 上的 FDD 控制 / 印表機緩衝程式

```

;
;FILE 'RW280' READ / WRITE ROUTINES
;YD280:      ACC.IY=COMMAND TABLE ADDRESS
;            CY=ERROR.ERCODE=EROOR CODE
= 0001      PSLOT      EQU      1      :PRINTER SLOT
= 0800      START      EQU      800H
= 0019      SLOTAD     EQU      19H
;
= 0010      SKCOM1     EQU      10H      :command 0..none
;          :          1..read
;          :          2..write
;          :          3..seek
;          :          4..track read
= 0004      MLTRK      EQU      4      ;          5..initialize

```

```

:
= 0011 SKDSK EQU 11H
= 0004 MAXDSK EQU 4
= 0002 DSK1 EQU 2 :drive 2..YD480..1K/sect
= 0003 DSK2 EQU 3 : 3..YD280..1K/sect
= 0004 DSK3 EQU 4 : 4..YD480..256/sec
= 0005 DSK4 EQU 5 : 5..YD280..256/sec
= 0006 DSK5 EQU 6 : 6..APDSK1
= 0007 DSK6 EQU 7 : 7..APDSK2..slot5
:
= 0012 SKTRK EQU 12H :track
= 0013 SKSEC EQU 13H :sector
= 0014 SKADR EQU 14H :address
= 0016 SKERR EQU 16H :ERROR REGISTER
= 0018 ONBLK EQU 18H :CURRENT BLOCK NUMBER
:
= 0002 INPOINT EQU 2
= 0004 PRPOINT EQU 4
= 0006 PRSIZE EQU 6
:
= 0008 ONCOM EQU 8 :COMMAND REG.
= 0009 ONDSK EQU 9 :DRIVE No.
= 000A ONTRK EQU 10 :TRACK REG..BIT7=SIDE.TRACK=0..80
= 000B ONSEC EQU 11 :SECTOR REG..SECTOR=1..16
= 000C ONADR EQU 12 :ADDRESS REG.
= 000E TIME60 EQU 14
:
= 0000 TEMP0 EQU 0
:* YD280 DRIVE ROUTINE
:* FDC...MB8876
:* NO. 7 SLOT
= C0F0 FDCB: EQU 0C0F0H
:* FDC COMAND & STATUS
:* FDC+1 TRACK
:* FDC+2 SECTOR
:* FDC+3 DATA
= C0F4 FDLB: EQU FDCB+4
:* B7 1:MOTOR ON
:* B5 1:DOUBLE.0:SINGL DENSITY
:* B4 1: 1 MHZ.0: 2MHZ
:* B2 0:DRIVE 1 SELECT
:* B1 0:DRIVE 0 SELECT
:* B0 SIDE SELECT...1:SIDE 0
= C0F8 FDFB: EQU FDCB+8
:* B7 DATA REQUEST
:* B6 IRQ
:* B5 BUSY FLAG
:IDATA: EQU START+400H
:* INITIALIZE DATA
:
= 00EF FREQ2: EQU 0EFH :2MHZ D0 S1 0FCH
= 00FD DRIVE0: EQU 0FDH :DRIVE 0 D1 S0 0FBH
= 00FB DRIVE1: EQU 0FBH :DRIVE 1 D1 S1 0FAH
= 0001 SIDE0: EQU 1
= 00FE SIDE1: EQU 0FEH :SIDE 1
= 0082 RDDATA: EQU 082H :READ DATA WITH SIDE TEST
= 0092 TREAD EQU 092H
= 00A2 WRDATA: EQU 0A2H :WRITE DATA WITH SIDE TEST
= 0008 HOMEB: EQU 8
= 0018 SEEK: EQU 18H
= 00D8 FDINT: EQU 0D8H :FORTH INTERRUPT COMMAND
= 00D0 CLINT: EQU 0D0H :CLEAR FORTH INTERRUPT

```

```

= 001E      BREAKY      EQU      01EH
:
= 4000      PRTOP       EQU      4000H      :PRINTER BUFFER TOP ADDRESS
= 8000      PRBTM       EQU      8000H
= 4000      BFSIZE      EQU      PRBTM-PRTOP
:
= C082      FDACK       EQU      0C082H      :FD ACKNOWRIDGE
= C081      FDSOH       EQU      0C081H      :COMMAND READ
= C083      INTIME      EQU      0C083H
:
= C000      KEY EQU      0C000H
= FDED      OUTC        EQU      0FDEDH
= C090      MPDATA      EQU      0C080H+10H*PSLOT
= C1C0      ACK EQU      0C0C0H+100H*PSLOT
= C1C1      PBUSY       EQU      0C0C1H+100H*PSLOT
= 0020      MAXCHR      EQU      32
:
: Z80 CONTROL PROGRAM
:
= 0300      INREG       EQU      300H
= 0302      PRDATA      EQU      302H
= 000B      IRQON       EQU      0BH
= 000F      ZRESET      EQU      0FH
= C080      INPUT       EQU      0C080H
= FC58      CLEAR       EQU      0FC58H
= 0036      CSW EQU      36H
= 0033      KSW EQU      38H
= FDF0      COUT1       EQU      0FDF0H
= FD1B      KEYIN       EQU      0FD1BH
:
0000 = 0100      ORG      100H      :START ROUTINE
0100 EA          NOP
0101 EA          NOP
0102 EA          NOP
0103 A000        LDY      #0          :IY INITIALIZE
0105 A536        LDA      CSW
0107 D00A ^0113  BNE      VECTEX
0109 A9F0        LDA      #COUT1 AND 0FFH
010B 8536        STA      CSW
010D A9FD        LDA      #COUT1/256
010F 8537        STA      CSW+1
0111 D008 ^011B  BNE      VEX1
0113             VECTEX
0113 A91B        LDA      #KEYIN AND 0FFH
0115 8538        STA      KSW
0117 A9FD        LDA      #KEYIN/256
0119 8539        STA      KSW+1
011B             VEX1
011B 8DFFCF      STA      0CFFFH      :ROM OFF
011E ADFFCF      LDA      0CFFFH
0121 A9C8        LDA      #0C8H      :ROMADDR
0123 8501        STA      TEMP0+1
0125 A908        LDA      #START/256
0127 8503        STA      TEMP0+3
0129 A900        LDA      #0
012B 8500        STA      TEMP0
012D 8502        STA      TEMP0+2
012F A208        LDX      #8
0131             SLOOP1
0131 B100        LDA      [TEMP0],Y
0133 9102        STA      [TEMP0+2],Y

```

0135	C8	INY		
0136	D0F9 ^0131	BNE	SLOOP1	
0138	E601	INC	TEMP0+1	
013A	E603	INC	TEMP0+3	
013C	CA	DEX		
013D	D0F2 ^0131	BNE	SLOOP1	
	:			
013F	2058FF	JSR	0FF58H	;TAKE SLOT NO.
0142	BA	TSX		
0143	BD0001	LDA	100H.X	
0146	8519	STA	SLOTAD	;SLOT ADDRESS
0148	0A	ASL	A	
0149	0A	ASL	A	
014A	0A	ASL	A	
014B	0A	ASL	A	
014C	2570	AND	70H	
014E	8DF10C	STA	ZSLOT	
0151	A519	LDA	SLOTAD	
0153	8501	STA	TEMP0+1	
0155	A900	LDA	#0	
0157	8500	STA	TEMP0	
0159	A00F	LDY	#ZRESET	
015B	9100	STA	[TEMP0].Y	
015D	88	DEY		
015E	A200	LDX	#0	
0160	48	PHA		
0161	68	PLA		
0162	9100	STA	[TEMP0].Y	;Z80 RESET
	:			
0164		SLOOP0		
0164	48	PHA		
0165	68	PLA		
0166	E8	INX		
0167	D0FB ^0164	BNE	SLOOP0	
0169	A00B	LDY	#IRQON	
016B	9100	STA	[TEMP0].Y	
016D	A200	LDX	#0	
016F	EA	NOP		
0170	EA	NOP		
0171	ACF10C	LDY	ZSLOT	
0174		COLDLP		
0174	BD0308	LDA	GOZ80.X	;MESSAGE BUFFER
0177	29FF	AND	#0FFH	
0179	F00E ^0189	BEQ	STARTJ	
017B	8D0003	STA	INREG	
017E	9980C0	STA	INPUT.Y	
0181		LP1		
0181	AD0003	LDA	INREG	
0184	D0FB ^0181	BNE	LP1	
0186	E8	INX		
0187	D0EB ^0174	BNE	COLDLP	
0189	4C0008	STARTJ	JMP	START
	:			
018C	FFFFFFFF	DW	-1,-1,-1,-1,-1,-1,-1,-1,-1,-1	
	:			
01A0	= 0800	ORG	START	
	:			
0800	4C0B08	JMP	STARTB	
	:			
0803	2053544152	GOZ80	DB	START!'.0
	:			

080B	A900	STARTB	LDA	#0
080D	8510	STA	SKCOM	
080F	8516	STA	SKERR	
0811	8DAF0C	STA	KEYCNT	
0814	8DB00C	STA	KEYDSP	
0817	8D0003	STA	INREG	
081A	20E808	JSR	INITZ	
081D		ST2		
081D	20900B	JSR	INKEYZ	
0820	A510	ST1 LDA	SKCOM	
0822	F003 ^0827	BEQ	ST3	
0824	209808	JSR	FDJOB	
0827		ST3:		
0827	206108	JSR	PRCHECK ;PRINTER CHECK	
082A	C60E	DEC	TIME60 ;TIMER CHECK	
082C	D0EF ^081D	BNE	ST2	
082E	AEF10C	LDX	ZSLOT	
0831	9D83C0	STA	INTIME.X ;TIMER IRQ	
0834	4C1D08	JMP	ST2	
		:		
0837	A507	INPR	LDA	PRSIZE+1
0839	C940	CMP	#BFSIZE/256	
083B	B023 ^0860	BCS	INPR2	
083D	A504	LDA	PRPOINT	
083F	6506	ADC	PRSIZE	
0841	8502	STA	INPOINT	
0843	A505	LDA	PRPOINT+1	
0845	6507	ADC	PRSIZE+1	
0847	1003 ^084C	BPL	INPR1	
0849	38	SEC		
084A	E940	SBC	#BFSIZE/256	
084C		INPR1		
084C	8503	STA	INPOINT+1	
084E	A000	LDY	#0	
0850	AD0203	LDA	PRDATA	
0853	9102	STA	[INPOINT].Y	
0855	A900	LDA	#0	
0857	8D0203	STA	PRDATA	
085A	E606	INC	PRSIZE	
085C	D002 ^0860	BNE	INPR2	
085E	E607	INC	PRSIZE+1	
0860		INPR2		
0860	60	RTS		
		:		
0861	AD0203	PRCHECK	LDA	PRDATA
0864	F003 ^0869	BEQ	PRINT	
0866	203708	JSR	INPR	
0869	A506	PRINT:	LDA	PRSIZE ;PRINTER BUFFER EMPTY?
086B	0507	ORA	PRSIZE+1	
086D	F028 ^0897	BEQ	RETURN ;NO DATA	
086F	ADC1C1	LDA	PBUSY ;PRINTER BUSY	
0872	3023 ^0897	BMI	RETURN	
0876	B104	LDA	[PRPOINT].Y	
0878	C90C	CMP	#0CH ;CS	
087A	F003 ^087F	BEQ	PRINT00	
087C	8D90C0	STA	MPDATA ;MP80 DATA REGISTER	
087F	A506	PRINT00	LDA	PRSIZE
0881	D002 ^0885	BNE	PRINT1	
0883	C607	DEC	PRSIZE+1	
0885	C606	PRINT1	DEC	PRSIZE

0887	E604	INC	PRPOINT	
0889	D00C ^0897	BNE	RETURN	
088B	E605	INC	PRPOINT+1	
088D	A505	LDA	PRPOINT+1	
088F	C980	CMP	#PRBTM/256	
0891	D004 ^0897	BNE	RETURN	
0893	A940	LDA	#PRTOP/256	
0895	8505	STA	PRPOINT+1	
0897	60	RETURN	RTS	
		;		
0898	20A408	FDJOB	JSR	YD280
089B	A900	LDA	#0	
089D	AEF10C	LDX	ZSLOT	
08A0	9D82C0	STA	FDACK.X	
08A3	60	RTS		
		;		
08A4	A205	YD280:	LDX	#5
08A6	B510	YD001	LDA	SKCOM.X
08A8	9508	STA	ONCOM.X	
08AA	CA	DEX		
08AB	10E9 ^08A6	BPL	YD001	
08AD	A900	LDA	#0	
08AF	8510	STA	SKCOM	
08B1	8DA40C	STA	ERCONT	
08B4	8516	STA	SKERR	
08B6	8518	STA	ONBLK	:NO SECTER ACCESS YET
08B8	AEF10C	LDX	ZSLOT	
08BB	9D81C0	STA	FDSOH.X	
08BE	A509	LDA	ONDSK	:DRIVE TEST
08C0	C902	CMP	#DSK1	
08C2	901F ^08E3	BCC	DSKERR	:less then drivel
08C4	C906	CMP	#DSK5	
08C6	B01B ^08E3	BCS	DSKERR	
08C8	38	SEC		
08C9	E902	SBC	#DSK1	
08CB	8509	STA	ONDSK	
08CD	A608	LDX	ONCOM	
08CF	1003 ^08D4	BPL	YD002	
08D1	20000C	JSR	DISPS	
08D4	CA	YD002	DEX	
08D5	F03C ^0913	BEQ	READZ	
08D7	CA	DEX		
08D8	F030 ^090A	BEQ	WRITEZ	
08DA	CA	DEX		
08DB	F030 ^090D	BEQ	SEEKZ	
08DD	CA	DEX		
08DE	F030 ^0910	BEQ	READSZ	:TRACK READ
08E0	CA	DEX		
08E1	F005 ^08E8	BEQ	INITZ	
08E3	A9FF	DSKERR	LDA	#0FFH
		;		
08E5	4C2209	JMP	FDCRET	
08E8	20470B	INITZ:	JSR	BUSYON
08EB	A9FF	LDA	#0FFH	
08ED	A203	LDX	#MAXDSK-1	:MAX DRIVES
08EF	9DA00C	INITZ1	STA	OLDTRK.X
08F2	CA	DEX		
08F3	10FA ^08EF	BPL	INITZ1	
08F5	8DA90C	STA	OLDDSK	
08F8	A900	LDA	#0	
08FA	8504	STA	PRPOINT	

08FC	8506	STA	PRSIZE	
08FE	8507	STA	PRSIZE+1	
0900	8D0203	STA	PRDATA	
0903	A940	LDA	#PRTOP/256	
0905	8505	STA	PRPOINT+1	
0907	4C1F0A	JMP	NOERR	:STA DOESN'T CHANGE FLAG
;				
090A	4C8C09	WRITEZ:	JMP	WRITE
090D	4C580A	SEEKZ:	JMP	SEEKB
0910	4C3909	READSZ	JMP	READS
;				
0913		READZ:	:YD280	READ
0913	A90A	LDA	#10	:ERROR COUNT
0915	8DA40C	STA	ERCONT	
0918	203E0A	RDB2:	JSR	SETUPB :FDD SETUP
091B	B018 ^0935	BCS	RDYERR	
091D	20EB09	JSR	RDSUB	:1 SECTOR READ SUB
0920	D00C ^092E	BNE	RDERR	
0922		FDCRET		
0922	8516	STA	SKERR	:command or drive error
0924	A90A	LDA	#10	
0926	38	SEC		
0927	EDA40C	SBC	ERCONT	
092A	8DA40C	STA	ERCONT	
092D	60	RTS		
;				
092E		RDERR:		
092E	CEA40C	DEC	ERCONT	:DEC ERROR COUNT
0931	D0E5 ^0918	BNE	RDB2	
0933	F0ED ^0922	BEQ	FDCRET	
;				
0935	A980	RDYERR:	LDA	#80H :NOT READY ERROR
0937	D0E9 ^0922	BNE	FDCRET	
;				
0939		READS		
0939	A901	LDA	#1	
093B	850B	STA	ONSEC	:SECTOR CLEAR
093D	ADAC0C	LDA	SECMAX	:SECTOR MAX
0940	8DAD0C	STA	SECCNT	:SECTOR COUNT
0943	A90A	LDA	#10	
0945	8DA40C	STA	ERCONT	:CLEAR ERROR COUNTER
0948		RDS161:		
0948	203E0A	JSR	SETUPB	:FDD SETUP
094B	B0E8 ^0935	BCS	RDYERR	
094D		RDS16:		
094D	A60B	LDX	ONSEC	
094F	8EF2C0	STX	FDCB+2	:SECTOR REGISTER
0952	A50C	LDA	ONADR	
0954	8500	STA	TEMP0	
0956	A50D	LDA	ONADR+1	
0958	8501	STA	TEMP0+1	
095A	ADAE0C	LDA	FDLDT	
095D	8DF4C0	STA	FDLB	
0960	20EB09	JSR	RDSUB	:READ 1 SECTOR
0963	D01F ^0984	BNE	RDSEERR	:ANY ERROR
0965	E60B	INC	ONSEC	
0967	ADAB0C	LDA	SECSIZ	:SECTOR SIZE
096A	AA	TAX		
096B	18	CLC		
096C	6518	ADC	ONBLK	
096E	8518	STA	ONBLK	

0970	8A	TXA		
0971	650D	ADC	ONADR+1	:LOAD ADDRESS HIGH BIT
0973	850D	STA	ONADR+1	:NEXT ADDRESS SET
0975	A90A	LDA	#10	
0977	8DA40C	STA	ERCONT	:CLEAR ERROR COUNTER
097A	CEAD0C	DEC	SECCNT	
097D	D0CE ^094D	BNE	RDS16	:NEXT SECTOR
097F	A900	LDA	#0	
0981	4C2209	JMP	FDCRET	
;				
0984		RDSERR		
0984	CEA40C	DEC	ERCONT	:DEC ERROR COUNT
0987	D0BF ^0948	BNE	RDS161	
0989	4C2209	JPRET	JMP	FDCRET
;				
098C		WRITE:		:YD280 WRITE ROUTINE
098C	A90A	LDA	#10	:ERROR COUNT
098E	8DA40C	STA	ERCONT	
0991	203E0A	JSR	SETUPB	
0994	B09F ^0935	BCS	RDYERR	:NOT READY ERROR
0996	ADF0C0	LDA	FDCB	
0999	2940	AND	#40H	:TEST WRITE PROTECT
099B	D0EC ^0989	BNE	JPRET	:PROTECT ERROR
099D	A9A2	LDA	#WRDATA	:WRITE COMMAND
099F	0DAA0C	ORA	SIDEREGR	:SIDE DATA
09A2	8DF0C0	STA	FDCB	
09A5	A000	LDY	#0	
09A7	F003 ^09AC	BEQ	LOOPW1	
09A9	8DF3C0	WRB1:	STA	FDCB+3
09AC	B100	LOOPW1	LDA	[TEMP0].Y
09AE	C8	INY		:ONLY 256 BYTE
09AF	D002 ^09B3	BNE	LOOPW	
09B1	E601	INC	TEMP0+1	
09B3	2CF8C0	LOOPW:	BIT	FDFB :FDC TEST
09B6	30F1 ^09A9	BMI	WRB1	
09B8	50F9 ^09B3	BVC	LOOPW	
09BA	ADF0C0	LDA	FDCB	
09BD	297C	AND	#7CH	
09BF	D009 ^09CA	BNE	WERB	
09C1	20C60A	JSR	WAIT1	:WAIT 1ms
09C4	20C60A	JSR	WAIT1	
09C7	4C1F0A	JMP	NOERR	
;				
09CA		WERB:		
09CA	CEA40C	DEC	ERCONT	
09CD	F014 ^09E3	BEQ	ERRORW	:ER MESSAGE OUT
09CF	2950	AND	#50H	
09D1	F0BE ^0991	BEQ	WRB2	:SOFT ERROR
;				
09D3	2940	ERW2:	AND	#40H
09D5	D00C ^09E3	BNE	ERRORW	
09D7	20C60A	JSR	WAIT1	
09DA	20C60A	JSR	WAIT1	
09DD	20D40A	JSR	SKHOME	
09E0	4C9109	JMP	WRB2	
09E3		ERRORW		
09E3	ADF0C0	LDA	FDCB	
09E6	297C	AND	#7CH	
09E8	8516	STA	SKERR	
09EA	60	RTS		
;				

09EB		RDSUB	
09EB	ADAE0C	LDA	FDLDT
09EE	8DF4C0	STA	FDLB
09F1	A982	LDA	#RDDATA :READ
09F3	AEAB0C	LDX	SECSIZ
09F6	0DAA0C	ORA	SIDEREGL :SIDE1?
09F9	8DF0C0	STA	FDCB
09FC	A000	LDY	#0
09FE	F00B ^0A0B	BEQ	LOOPR
0A00	ADF3C0	RDB1:	LDA FDCB+3 :4:DATA
0A03	9100	STA	[TEMP0],Y :6
0A05	C8	INY	:2:ONLY 256 BYTE
0A06	D003 ^0A0B	BNE	LOOPR :3..2
0A08	E601	INC	TEMP0+1 :5
0A0A	CA	DEX	:2:SECTOR COUNT
0A0B	2CF8C0	LOOPR:	BIT FDFB :4:FDC TEST
0A0E	30F0 ^0A00	BMI	RDB1 :3..2
0A10	50F9 ^0A0B	BVC	LOOPR :3
		:BYTE COUNT TEST	
0A12	CA	DEX	
0A13	F003 ^0A18	BEQ	RDB3 :OVERRUN ERROR?
0A15	E8	INX	
0A16	D01E ^0A36	BNE	DSKER1 :DISK ERROR IX must be 0 or 1
0A18	ADF0C0	RDB3	LDA FDCB
0A1B	291C	AND	#01CH :ERROR TEST
0A1D	D003 ^0A22	BNE	RDZER
0A1F	A900	NOERR	LDA #0 :NO ERROR RETURN
0A21	60	RTS	:ZY=1 NOERROR , ZY=0 ERROR
		:	
0A22	48	RDZER	PHA
0A23	2910	AND	#10H :SEEK ERROR
0A25	F00B ^0A32	BEQ	RDERR2 :SOFT ERROR
0A27	20D40A	NOFIND:	JSR SKHOME :SEEK HOME
0A2A	CEA40C	DEC	ERCONT
0A2D	D003 ^0A32	BNE	RDERR2
0A2F	EEA40C	INC	ERCONT
0A32		RDERR2	
0A32	68	PLA	
0A33	291C	AND	#1CH :CLEAR ZY
0A35	60	RTS	
		:	
0A36		DSKER1	
0A36	A901	LDA	#1
0A38	8DA40C	STA	ERCONT :SECTOR SIZE ERROR
0A3B	A910	LDA	#10H :SEEK ERROR
0A3D	60	RTS	
		:	
0A3E	20580A	SETUPB:	JSR SEEKB :SEEK WITHOUT WAIT
0A41	A60B	LDX	ONSEC :SECTOR DATA SET
0A43	8EF2C0	STX	FDCB+2 :SECTOR REGISTER
0A46	A50C	LDA	ONADR
0A48	8500	STA	TEMP0
0A4A	A50D	LDA	ONADR+1
0A4C	8501	STA	TEMP0+1
0A4E	205C0B	JSR	FDRDY :FD DRIVE READY?
0A51	ADAE0C	LDA	FDLDT
0A54	8DF4C0	STA	FDLB
0A57	60	RTS	
		:	
0A58		SEEKB:	
0A58	A509	LDA	ONDSK :NEXT DISK NO.

0A5A	CDA90C	CMP	OLDDSK	:DISK COMPAIR
0A5D	F003 ^0A62	BEQ	SEEKB1	:NO CHANGE
0A5F	20EE0A	JSR	XCHGFD	:EXCHANGE FD DRIVE & SET FDLDT FD
0A62	20470B	SEEKB1:	JSR	BUSYON on..2 MHz
0A65	A50A	LDA	ONTRK	:TRACK TEST
0A67	4A	LSR	A	:SECTER BIT
0A68	A8	TAY		:TRACK NO.
0A69	ADAE0C	LDA	FDLDT	
0A6C	9006 ^0A74	BCC	SEEKB2	:SIDE0?
0A6E	29FE	AND	#SIDE1	:SIDE 1 FDLDT
0A70	A208	LDX	#8	:SIDE1
0A72	D004 ^0A78	BNE	SEEKB3	
0A74	0901	SEEKB2:	ORA	#SIDE0
0A76	A200	LDX	#0	
0A78	8DAE0C	SEEKB3	STA	FDLDT
0A7B	29EF	AND	#FREQ2	:2MHz
0A7D	8DF4C0	STA	FDLB	
0A80	CCF1C0	CPY	FDCB+1	:TRACK REGISTER
0A83	F021 ^0AA6	BEQ	SEEKB4	:SAME TRACK
0A85	8EAA0C	STX	SIDEREG	:SIDE SET
0A88	98	TYA		:TRACK 0 TEST
0A89	F049 ^0AD4	BEQ	SKHOME	:SEEK HOME
0A8B	8DF3C0	STA	FDCB+3	:DATA REGISTER...TRACK SET
0A8E	A918	LDA	#SEEK	:SEEK COMMAND
0A90	8DF0C0	STA	FDCB	:SEEK
0A93	20520B	JSR	BUSY	:WAIT FOR READY
0A96	ADAE0C	LDA	FDLDT	
0A99	8DF4C0	STA	FDLB	
0A9C	20B80A	JSR	WAIT18	:WAIT 18 msec.
0A9F	ADAE0C	LDA	FDLDT	
0AA2	8DF4C0	STA	FDLB	
0AA5	60	RTS		
		:		
0AA6	ADAE0C	SEEKB4:	LDA	FDLDT :FDSET DATA
0AA9	8DF4C0	STA	FDLB	:FD ON 1MHz
0AAC	ECAA0C	CPX	SIDEREG	:SIDE COMPAIR
0AAF	F006 ^0AB7	BEQ	SEEKB5	
0AB1	8EAA0C	STX	SIDEREG	
0AB4	20C60A	JSR	WAIT1	:SIDE CHANGE WAIT
0AB7	60	SEEKB5:	RTS	
		:		
0AB8	A912	WAIT18:	LDA	#18
0ABA	8DA50C	STA	SECOND	
0ABD		WAITA:		:WAIT ACC SECOND
0ABD	20C60A	JSR	WAIT1	
0AC0	CEA50C	DEC	SECOND	
0AC3	D0F8 ^0ABD	BNE	WAITA	
0AC5	60	RTS		
		:		
0AC6		WAIT1:		
0AC6	A93E	LDA	#62	
0AC8	8DA60C	STA	TIMER1	:992 micro sec.
0ACB		WAIT3:	:4	
0ACB	20AD0B	JSR	INKEY	:INKEY SUPORT
0ACE		WAIT2:		
0ACE	CEA60C	DEC	TIMER1	:6
0AD1	D0F8 ^0ACB	BNE	WAIT3	:3 LOOP=16*63
0AD3	60	RTS		
0AD4		SKHOME:		
0AD4	ADAE0C	LDA	FDLDT	
0AD7	29EF	AND	#FREQ2	

0AD9	8DF4C0	STA	FDLB
0ADC	A908	LDA	#HOMEB
0ADE	8DF0C0	STA	FDCB
0AE1	20520B	JSR	BUSY
0AE4	20B80A	JSR	WAIT18
0AE7	ADAE0C	LDA	FDLDT
0AEA	8DF4C0	STA	FDLB
0AED	60	RTS	
;			
0AEE	20520B	XCHGFD:	JSR BUSY
0AF1	A9FF	LDA	#0FFH ;FD OFF
0AF3	8DF4C0	STA	FDLB
0AF6	8DAE0C	STA	FDLDT ;FDL DATA REG
0AF9	AEA90C	LDX	OLDDSK
0AFC	3006 ^0B04	BMI	XFD2
0AFE	ADF1C0	LDA	FDCB+1 ;OLD TRACK
0B01	9DA00C	STA	OLDTRK,X
0B04		XFD2:	;SEEK FREQ
0B04	20200B	JSR	GETFD ;GET FD PARAMETER
0B07	ADAE0C	LDA	FDLDT
0B0A	29EF	AND	#FREQ2
0B0C	8DF4C0	STA	FDLB ;2MHz parameter
0B0F	A609	LDX	ONDSK
0B11	8EA90C	STX	OLDDSK
0B14	BDA00C	LDA	OLDTRK,X
0B17	8DF1C0	STA	FDCB+1 ;CURRENT TRACK
0B1A	1003 ^0B1F	BPL	XFD1
0B1C	20D40A	JSR	SKHOME
0B1F		XFD1:	
0B1F	60	RTS	
;			
0B20	A904	GETFD:	LDA #4 ;1K/SEC
0B22	A205	LDX	#5 ;5SECTOR/TRACK
0B24	A409	LDY	ONDSK ;
0B26	F00E ^0B36	BEQ	GETFD0 ;DRIVE 0
0B28	88	DEY	
0B29	F007 ^0B32	BEQ	GETFD1 ;DRIVE1 1K/SEC
0B2B	A901	LDA	#1 ;256BYTES/SEC
0B2D	A210	LDX	#16 ;16 SECTOR/TRACK
0B2F	88	DEY	
0B30	F004 ^0B36	BEQ	GETFD0 ;DRIVE0
0B32		GETFD1:	
0B32	A0FB	LDY	#DRIVE1
0B34	D002 ^0B38	BNE	GETFD2
0B36	A0FD	GETFD0:	LDY #DRIVE0
0B38	8DAB0C	GETFD2:	STA SECSIZ ;SECTOR SIZE
0B3B	8EAC0C	STX	SECMAX
0B3E	A202	LDX	#2 ;SIDE 0 DATA
0B40	8CAE0C	STY	FDLDT ;
0B43	8EAA0C	STX	SIDEREG
0B46	60	RTS	
;			
0B47		BUSYON:	
0B47	ADF0C0	LDA	FDCB
0B4A	4A	LSR	A
0B4B	900E ^0B5B	BCC	BUSY3
0B4D	A9D8	LDA	#FDINT
0B4F	8DF0C0	STA	FDCB
;			
0B52		BUSY:	;KEY INPUT?
0B52	20AD0B	JSR	INKEY
0B55	ADF0C0	BUSY2	LDA FDCB ;FLAG
0B58	4A	LSR	A ;BUSY BIT -> CY

0B59	B0F7 ^0B52	BCS	BUSY	
0B5B		BUSY3:		
0B5B	60	RTS		
		:		
0B5C	A07C	FDRDY:	LDY	#7CH :ABOUT 2 SECOND
0B5E	8CA70C	STY	TIME	
0B61	8CA80C	STY	TIME+1	
0B64	20AD0B	FDRY1:	JSR	INKEY
0B67	ADF0C0	FDRY2	LDA	FDCB
0B6A	1022 ^0B8E	BPL	RDY00	:DISK READY
0B6C	C8	INY		
0B6D	D0F5 ^0B64	BNE	FDRY1	
0B6F	EEA70C	INC	TIME	
0B72	D0F0 ^0B64	BNE	FDRY1	
0B74	EEA80C	INC	TIME+1	
0B77	10EB ^0B64	BPL	FDRY1	
0B79	ADAE0C	LDA	FDLDT	:FDL DATA REG
0B7C	8DF4C0	STA	FDLB	:FDD WAKE UP
0B7F	A9D8	LDA	#0D8H	:INTERRUPT
0B81	8DF0C0	STA	FDCB	
0B84	20520B	JSR	BUSY	:TEST BUSY
0B87	A9D0	LDA	#CLINT	:INTERUPT CLEAR
0B89	8DF0C0	STA	FDCB	
0B8C	38	SEC		
0B8D	60	RTS		
0B8E	18	RDY00:	CLC	:ERROR RET
0B8F	60	RTS		
		:		
0B90		INKEYZ		
0B90	2C00C0	BIT	KEY	
0B93	1040 ^0BD5	BPL	INKEY2	
0B95	AD00C0	LDA	KEY	
0B98	2C10C0	BIT	KEY+16	
0B9B	297F	AND	#7FH	
0B9D	C91E	CMP	#BREAKY	
0B9F	D019 ^0BBA	BNE	INKEY1	
		:		
0BA1	00	BRK		
0BA2	EA	NOP		
0BA3	EA	NOP		
0BA4	EA	NOP		
0BA5	EA	NOP		
0BA6	AA	TAX		
0BA7	68	PLA		
0BA8	68	PLA		
0BA9	8A	TXA		
0BAA	4CBA0B	JMP	INKEY1	
		:		
0BAD	2C00C0	INKEY	BIT	KEY
0BB0	1023 ^0BD5	BPL	INKEY2	
0BB2	AD00C0	LDA	KEY	
0BB5	2C10C0	BIT	KEY+16	
0BB8	297F	AND	#7FH	
0BBA		INKEY1		
0BBA	ACB00C	LDY	KEYDSP	
0BBD	99B10C	STA	KEYBUF.Y	
0BC0	AEAF0C	LDX	KEYCNT	
0BC3	E8	INX		
0BC4	E020	CPX	#MAXCHR	
0BC6	B00D ^0BD5	BCS	INKEY2	:BUFFER FULL
0BC8	C8	INY		

0BC9	C020	CPY	#MAXCHR
0BCB	9002 ^0BCF	BCC	INKEY3
0BCD	A000	LDY	#0
0BCF		INKEY3	
0BCF	8CB00C	STY	KEYDSP
0BD2	8EAF0C	STX	KEYCNT
0BD5		INKEY2	
0BD5	A0AF0C	LDX	KEYCNT
0BD8	F025 ^0BFF	BEQ	INKEY4
0BDA	AD0003	LDA	INREG
0BDD	D020 ^0BFF	BNE	INKEY4
0BDF	ADB00C	LDA	KEYDSP
0BE2	38	SEC	
0BE3	EDAF0C	SBC	KEYCNT
0BE6	B002 ^0BEA	BCS	INKEY5
0BE8	6920	ADC	#MAXCHR
0BEA		INKEY5	
0BEA	CA	DEX	
0BEB	8EAF0C	STX	KEYCNT
0BEE	AA	TAX	
0BEF	BDB10C	LDA	KEYBUF.X
0BF2	C91E	CMP	#BREAKY
0BF4	F009 ^0BFF	BEQ	INKEY4
0BF6	8D0003	STA	INREG
0BF9	AEF10C	LDX	ZSLOT
0BFC	9D80C0	STA	INPUT.X
0BFF		INKEY4	
0BFF	60	RTS	
0C00		DISPS	
0C00	A200	LDX	#0
0C02	206F0C	JSR	DISPS1
0C05	A509	LDA	ONDSK
0C07	18	CLC	
0C08	69B0	ADC	#0B0H
0C0A	9DD007	STA	7D0H.X
0C0D	E8	INX	
0C0E	206F0C	JSR	DISPS1
0C11	A50A	LDA	ONTRK
0C13	C964	CMP	#100
0C15	900B ^0C22	BCC	D100
0C17	38	SEC	
0C18	E964	SBC	#100
0C1A	48	PHA	
0C1B	A9B1	LDA	#0B1H
0C1D	9DD007	STA	7D0H.X
0C20	68	PLA	
0C21	E8	INX	
0C22		D100	
0C22	A0B0	LDY	#0B0H
0C24	38	D101	SEC
0C25	E90A	SBC	#10
0C27	C8	INY	
0C28	B0FA ^0C24	BCS	D101
0C2A	69BA	ADC	#0B0H+10
0C2C	88	DEY	
0C2D	48	PHA	
0C2E	98	TYA	
0C2F	9DD007	STA	7D0H.X
0C32	E8	INX	
0C33	68	PLA	
0C34	9DD007	STA	7D0H.X
0C37	E8	INX	

n	ACK	C1C0	2#30						
	BFSIZE	4000	2#21	4/40	4/49				
	BREAKY	001E	2#17	12/ 5	12/53				
	BUSY	0B52	9/51	10/31	10/37	11#27	11/31	11/51	
n	BUSY2	0B55	11#29						
	BUSY3	0B5B	11/23	11#32					
	BUSYON	0B47	6/13	9/29	11#20				
n	CLEAR	FC58	2#41						
	CLINT	00D0	2#16	11/52					
	COLDLP	0174	3#58	4/ 8					
	COUT1	FDF0	2#44	2/54	2/56				
	CSW	0036	2#42	2/52	2/55	2/57			
	D100	0C22	13/12	13#20					
	D101	0C24	13#22	13/25					
	DISP2	0C7C	14/ 6	14#11					
	DISPS	0C00	5/58	12#61					
	DISPS1	0C6F	13/ 3	13/ 9	13/35	13/50	14# 5	14/10	
	DRIVE0	00FD	2# 6	11/12					
	DRIVE1	00FB	2# 7	11/10					
	DSK1	0002	1#18	5/49	5/54				
n	DSK2	0003	1#19						
n	DSK3	0004	1#20						
n	DSK4	0005	1#21						
	DSK5	0006	1#22	5/51					
n	DSK6	0007	1#23						
	DSKER1	0A36	8/47	9# 6					
	DSKERR	08E3	5/50	5/52	6# 9				
	DSP10	0C4C	13/38	13#46					
	DSP11	0C62	13/54	13#57					
	DSP12	0C64	13/56	13#58					
n	ENDPRO	0CF2	14#32						
	ERCONT	0CA4	5/43	6/35	6/44	6/45	6/49	7/ 2	7/27
			7/34	7/40	8/ 7	8/58	8/60	9/ 8	14#16
	ERRORW	09E3	8/ 8	8/13	8#19				
n	ERW2	09D3	8#12						
	FDACK	C082	2#23	5/33					
	FDCB	C0F0	1#46	1/51	1/58	7/ 8	7/43	7/48	7/52
			7/59	8/20	8/31	8/34	8/48	9/14	9/43
			9/48	9/50	10/30	10/43	10/53	11/21	11/25
			11/29	11/39	11/50	11/53			
	FDCRET	0922	6/11	6#40	6/51	6/54	7/31	7/36	
	FDFB	C0F8	1#58	7/56	8/40				
	FDINT	00D8	2#15	11/24					
	FDJOB	0898	4/30	5#30					
	FDLB	C0F4	1#51	7/14	8/27	9/21	9/42	9/53	9/50
			9/60	10/28	10/34	10/39	10/49	11/48	
	FDLDT	0CAE	7/13	8/26	9/20	9/33	9/40	9/52	9/55
			9/59	10/26	10/33	10/40	10/47	11/16	11/47
			14#25						
	FDRDY	0B5C	9/19	11#35					
	FDRY1	0B64	11#38	11/42	11/44	11/46			
n	FDRY2	0B67	11#39						
	FDSOH	C081	2#24	5/47					
	FREQ2	00EF	2# 5	9/41	10/27	10/48			
	GETFD	0B20	10/46	10#59					
	GETFD0	0B36	11/ 2	11/ 8	11#12				
	GETFD1	0B32	11/ 4	11# 9					
	GETFD2	0B38	11/11	11#13					

GOZ80	0803	3/59	4#17						
HOMEB	0008	2#13	10/29						
INITZ	08E8	4/25	6/ 8	6#13					
INITZ1	08EF	6#16	6/18						
INKEY	0BAD	10/19	11/28	11/38	12#19				
INKEY1	0BBA	12/ 6	12/17	12#24					
INKEY2	0BD5	11/61	12/20	12/30	12#38				
INKEY3	0BCF	12/33	12#35						
INKEY4	0BFF	12/40	12/42	12/54	12#58				
INKEY5	0BEA	12/46	12#48						
INKEYZ	0B90	4/27	11#59						
INPOIN	0002	1#31	4/44	4/51	4/54				
INPR	0837	4#39	5/ 5						
INPR1	084C	4/47	4#50						
INPR2	0860	4/41	4/58	4#60					
INPUT	C080	2#40	4/ 3	12/57					
INREG	0300	2#36	4/ 2	4/ 5	4/24	12/41	12/55		
INTIME	C083	2#25	4/36						
IRQON	000B	2#38	3/52						
JPRET	0989	7#36	7/45						
KEY	C000	2#27	11/60	12/ 2	12/ 3	12/19	12/21	12/22	
KEYBUF	0CB1	12/26	12/52	14#29					
KEYCNT	0CAF	4/22	12/27	12/37	12/39	12/45	12/50	14#27	
KEYDSP	0CB0	4/23	12/25	12/36	12/43	14#28			
KEYIN	FD1B	2#45	2/60	3/ 2					
KSW	0038	2#43	2/61	3/ 3					
LOOPR	0A0B	8/33	8/37	8#40	8/42				
LOOPW	09B3	7/54	7#56	7/58					
LOOPW1	09AC	7/50	7#52						
LP1	0181	4# 4	4/ 6						
MAXCHR	0020	2#32	12/29	12/32	12/47				
MAXDSK	0004	1#17	6/15						
MES1	0C7D	14/ 5	14#13						
n MLTTRK	0004	1#14							
MPDATA	C090	2#29	5/15						
NOERR	0A1F	6/27	8/ 4	8#51					
n NOFIND	0A27	8#57							
OLDDSK	0CA9	6/19	9/26	10/41	10/51	14#20			
OLDTRK	0CA0	6/16	10/44	10/52	14#15				
ONADR	000C	1#39	7/ 9	7/11	7/24	7/25	9/15	9/17	
ONBLK	0018	1#29	5/45	7/21	7/22				
ONCOM	0008	1#35	5/38	5/56	13/51	13/59	13/61		
ONDSK	0009	1#36	5/48	5/55	9/25	10/50	10/61	13/ 4	
ONSEC	000B	1#38	6/58	7/ 7	7/17	9/13	13/36		
ONTRK	000A	1#37	9/30	13/10					
n OUTC	FDED	2#28							
PBUSY	C1C1	2#31	5/ 9						
PRBTM	8000	2#20	2/21	5/24					
PRCHEC	0861	4/32	5# 3						
PRDATA	0302	2#37	4/53	4/56	5/ 3	6/24			
PRINT	0869	5/ 4	5# 6						
PRINT0	087F	5/14	5#16						
PRINT1	0885	5/17	5#19						
PRPOIN	0004	1#32	4/42	4/45	5/12	5/20	5/22	5/23	
		5/27	6/21	6/26					
PRSIZE	0006	1#33	4/39	4/43	4/46	4/57	4/59	5/ 6	
		5/ 7	5/16	5/18	5/19	6/22	6/23		
PRTOP	4000	2#19	2/21	5/26	6/25				
PSLOT	0001	1# 5	2/29	2/30	2/31				
RDB1	0A00	8#34	8/41						
RDB2	0918	6#36	6/50						

RDB3	0A18	8/45	8#48						
RDDATA	0082	2#10	8/28						
RDERR	092E	6/39	6#48						
RDERR2	0A32	8/56	8/59	8#61					
RDS16	094D	7# 6	7/29						
RDS161	0948	7# 3	7/35						
RDSERR	0984	7/16	7#33						
RDSUB	09EB	6/38	7/15	8#25					
RDY00	0B8E	11/40	11#56						
RDYERR	0935	6/37	6#53	7/ 5	7/42				
RDZER	0A22	8/50	8#54						
READS	0939	6/31	6#56						
READSZ	0910	6/ 6	6#31						
READZ	0913	5/60	6#33						
RETURN	0897	5/ 8	5/10	5/21	5/25	5#28			
SECCNT	0CAD	6/60	7/28	14#24					
SECMAX	0CAC	6/59	11/14	14#23					
SECOND	0CA5	10/ 8	10/11	14#17					
SECS1Z	0CAB	7/18	8/29	11/13	14#22				
SEEK	0018	2#14	9/49						
SEEKB	0A58	6/30	9/12	9#24					
SEEKB1	0A62	9/27	9#29						
SEEKB2	0A74	9/34	9#38						
SEEKB3	0A78	9/37	9#40						
SEEKB4	0AA6	9/44	9#59						
SEEKB5	0AB7	10/ 2	10# 5						
SEEKZ	090D	6/ 4	6#30						
SETUPB	0A3E	6/36	7/ 4	7/41	9#12				
SIDE0	0001	2# 8	9/38						
SIDE1	00FE	2# 9	9/35						
SIDERE	0CAA	7/47	8/30	9/45	9/61	10/ 3	11/17	14#21	
n SKADR	0014	1#27							
SKCOM	0010	1# 9	4/20	4/28	5/37	5/42			
n SKDSK	0011	1#16							
SKERR	0016	1#28	4/21	5/44	6/41	8/22			
SKHOME	0AD4	8/16	8/57	9/47	10#25	10/55			
n SKSEC	0013	1#26							
n SKTRK	0012	1#25							
SLOOP0	0164	3#47	3/51						
SLOOP1	0131	3#15	3/19	3/23					
SLOTAD	0019	1# 7	3/28	3/35					
n ST1	0820	4#28							
ST2	081D	4#26	4/34	4/37					
ST3	0827	4/29	4#31						
START	0800	1# 6	3/ 9	4/ 9	4/13				
STARTB	080B	4/15	4#19						
STARTJ	0189	3/61	4# 9						
TEMPO	0000	1#42	3/ 8	3/10	3/12	3/13	3/16	3/17	
		3/20	3/21	3/36	3/38	3/40	3/45	3/53	
		7/10	7/12	7/52	7/55	8/35	8/38	9/16	
		9/18							
TIME	0CA7	11/36	11/37	11/43	11/45	14#19			
TIME60	000E	1#40	4/33						
TIMER1	0CA6	10/17	10/21	14#18					
n TREAD	0092	2#11							
VECTEX	0113	2/53	2#59						
VEX1	011B	2/58	3# 4						
WAIT1	0AC6	8/ 2	8/ 3	8/14	8/15	10/ 4	10/10	10#15	
WAIT18	0AB8	9/54	10# 7	10/32					
n WAIT2	0ACE	10#20							
WAIT3	0ACB	10#18	10/22						
WAITA	0ABD	10# 9	10/12						

WERB	09CA	7/61	8# 6						
WRB1	09A9	7#51	7/57						
WRB2	0991	7#41	8/10	8/17					
WRDATA	00A2	2#12	7/46						
WRITE	098C	6/29	7#38						
WRITEZ	090A	6/ 2	6#29						
XCHGFD	0AEE	9/28	10#37						
XFD1	0B1F	10/54	10#56						
XFD2	0B04	10/42	10#45						
YD001	08A6	5#37	5/40						
YD002	08D4	5/57	5#59						
YD280	08A4	5/30	5#36						
ZRESET	000F	2#39	3/39						
ZSLOT	0CF1	3/34	3/57	4/35	5/32	5/46	12/56	14#30	

列表 3 Z 80 用的 BIOS 程式

```

:
: APPLE ][ BIOS PROGRAM
: V 2.10 AUG. 1 '84
:
= 0000 TEST EQU 0
= 003F MSIZE EQU 63 :RAM SIZE (Kilo Byte)
= 0019 ZSLOT EQU 19H :Z80 BOAD SLOT
= 0003 VSLOT EQU 3 :80 CHR VRAM SLOT
:
MESDB MACRO
DB '63K CP/M 2.2 ON APPLE ][ REV. 2.10'
DB ' (84/8/1)'
DB CR.LF.07.0
ENDM

:
= 0006 DSKMAX EQU 6 : 2 DRIVE SYSTEM
:
= FFC0 SPTOP2 EQU 0FFC0H
= FFA0 SPTOP EQU 0FFA0H
= FC00 RAM$TOP EQU MSIZE * 1024
= F600 BIOS EQU RAM$TOP - 600H
= E800 BDOS EQU BIOS - 0E00H
= E000 CCP EQU BDOS - 0800H
:
= 0003 IOBYTE EQU 3
= 0004 DRIVE EQU 0004H
:
= 000D CR EQU 0DH
= 000A LF EQU 0AH
= 001C CHANGY EQU 1CH :FS
= 0020 MAXCHR EQU 32
:
= 0300 INREG EQU 300H
= 0302 PROUT EQU 302H
= 000C ROM EQU 00CH
= 000D RAM EQU 00DH
= 1800 DBUF EQU 1800H
= 2C00 DBUF2 EQU 2C00H
:
= C0B0 CRTCO EQU VSLOT*10H+0C080H :SLOT 3
= C0B1 CRTCI EQU CRTCO+1 :DATA REGISTER

```

```

      = C0B0      BANK0      EQU      CRTCO
      = C0B4      BANK1      EQU      CRTCO+4
      = C0B8      BANK2      EQU      CRTCO+8
      = C0BC      BANK3      EQU      CRTCO+12
      = C300      VRAMON     EQU      VSLOT*100H+0C000H
      = CC00      VRAM       EQU      0CC00H
      = CE00      VRAME      EQU      0CE00H
      = C030      BELREG     EQU      0C030H
      = 000E      CURH       EQU      14
      = 000F      CURL       EQU      15
      = 000C      STADH      EQU      12
      = 000D      STADL      EQU      13
      :
      = 0000      WRALL      EQU      0
      = 0001      WRDIR      EQU      1
      = 0002      WRUAL      EQU      2
      :
      = 0010      FDOP       EQU      10H      :FD OPERATION FLAG
      = 0011      FDDSK      EQU      11H
      :
      = 0012      FDTRK      EQU      12H
      = 0013      FDSEK      EQU      13H
      = 0014      FDADR      EQU      14H
      = 0016      FDFLG      EQU      16H
      = 0017      FDRTY      EQU      17H      :RETRY COUNT
      :
      = 0800      BLKSIZ     EQU      2048
      :
0000 = F600      :      ORG      BIOS
      :
      :      JUMP VECTOR FOR INDIVIDUAL ROUTINE
      :
F600 C3F2F6      JMP      BOOT      :RESET ENTRY ADDRESS
F603 C3E4F6      WBOOT: JMP      WBOOT
F606 C33AF7      JMP      CONST
F609 C343F7      INCHR: JMP      CONIN
F60C C36EF7      JMP      CONOUT
F60F C32EF7      JMP      LPTOUT
      :
F612 79          PUNCH:      MOV      A.C
F613 C9          RET
F614 00          NOP
F615 3E1A        READER:     MVI      A.1AH      :^Z FILE END MARK OF CPM
F617 C9          RET
F618 C3F3F7      JMP      HOME
F61B C3C1F7      SELFDD     JMP      SELDSK      :::
F61E C3F5F7      JMP      SETTRK
F621 C3FBF7      JMP      SETSEC
F624 C301F8      JMP      SETDMA
F627 C37CF8      JMP      READ
F62A C38DF8      JMP      WRITE
F62D C32CF7      JMP      LSTST      :::
F630 C308F8      JMP      SECTRAN      :::
      :
F633             DPBASE
F633             DPBAS0:     :ROM DISK ON CARD
F633 00000000    DW      0.0      :SWKEW ?
F637 00000000    DW      0.0
F63B D9FD93F6    DW      DIRBUF.DPB0
F63F 8CFC8AFC    DW      CSV0.ALVO
      :

```

F643		DPBAS1:	:RAM DISK ON APPLE ... NOT INPLEMENTED
F643	00000000	DW	0.0
F647	00000000	DW	0.0
F64B	D9FDA2F6	DW	DIRBUF.DPB1
F64F	9DFC94FC	DW	CSV1.ALV1
;			
F653		DPBAS2:	:YD480 1K/SECTOR
F653	00000000	DW	0.0
F657	00000000	DW	0.0
F65B	D9FDC0F6	DW	DIRBUF.DPB2
F65F	D7FCA5FC	DW	CSV2.ALV2
;			
F663		DPBAS3:	:YD280 1K/SECTOR
F663	00000000	DW	0.0
F667	00000000	DW	0.0
F66B	D9FDC0F6	DW	DIRBUF.DPB2
F66F	29FDF7FC	DW	CSV3.ALV3
F673		DPBAS4:	:YD480 256/SECTOR
F673	00000000	DW	0.0 -
F677	00000000	DW	0.0
F67B	D9FDB1F6	DW	DIRBUF.DPB4
F67F	71FD49FD	DW	CSV4.ALV4
;			
F683		DPBAS5:	:YD280 256/SECTOR
F683	00000000	DW	0.0
F687	00000000	DW	0.0
F68B	D9FDB1F6	DW	DIRBUF.DPB4
F68F	B9FD91FD	DW	CSV5.ALV5
;			
F693		DPB0:	:27128 ROM X 2
F693	4000	SPT DW	64 : SEC PER TRACK
F695	03	BSH DB	3 : BLOCK SHIFT
F696	07	BLM DB	7 : BLOCK MASK
F697	00	EXM DB	0 : EXTENT MASK
F698	1700	DSM DW	23 : DISK SIZE-1
F69A	1F00	DRM DW	31 : DIRECTORY MAX
F69C	80	ALO DB	080H : ALLOC0
F69D	00	ALI DB	0 : ALLOC1
F69E	0000	CKS DW	0 : CHECK SIZE
F6A0	0100	OFF DW	1 : OFFSET
;			
F6A2		DPB1:	:16 RAM SIZE ?
F6A2	4000	DW	64 : SEC PER TRACK
F6A4	03	DB	3 : BLOCK SHIFT
F6A5	07	DB	7 : BLOCK MASK
F6A6	00	DB	0 : EXTENT MASK
F6A7	1000	DW	16 : DISK SIZE-1
F6A9	1F00	DW	31 : DIRECTORY MAX
F6AB	80	DB	080H : ALLOC0
F6AC	00	DB	0 : ALLOC1
F6AD	0000	DW	0 : CHECK SIZE
F6AF	0000	DW	0 : OFFSET
;			
F6B1		DPB4:	
F6B1	2000	DW	32 : SEC PER TRACK
F6B3	04	DB	4 : BLOCK SHIFT
F6B4	0F	DB	15 : BLOCK MASK
F6B5	00	DB	0 : EXTENT MASK
F6B6	3B01	DW	315 : DISK SIZE-1
F6B8	7F00	DW	127 : DIRECTORY MAX
F6BA	C0	DB	0C0H : ALLOC0

```

F6BB 00          DB      0          : ALLOC1
F6BC 2000        DW      32         : CHECK SIZE
F6BE 0200        DW      2          : OFFSET
;
F6C0            DPB2:
F6C0 2800        DW      40         :1K BYTE/SEC
F6C2 04          DB      4
F6C3 0F          DB      15
F6C4 00          DB      0
F6C5 8F01        DW      399        :800K BYTE/DRIVE
F6C7 7F00        DW      127
F6C9 C0          DR      0C0H
F6CA 00          DB      0
F6CB 2000        DW      32
F6CD 0000        DW      0          :ALL DATA AREA
F6CF            SKEW1: :256/SECTOR SKEW
F6CF 01060B1005  DB      1.6.11.16.5.10.15.4
F6D7 090E03080D DB      9.14.3.8.13.2.7.12
;
F6DF            SKEW0: :1K/SECTOR SKEW
F6DF 0103050204 DB      1.3.5.2.4
;
      = 0000      IF      TEST
      -           WBMES    DB      'WARM BOOT'.0DH.0AH.00
;               ENDIF
;
F6E4            WBOOT:
F6E4 31A0FF      LXI      SP,SPTOP
;
      = 0000      IF      TEST
      -           LXI      H,WBMES
      -           CALL     WHL
;               ENDIF
;
F6E7 011900      LXI      BC,ZSLOT
F6EA ED78        $      IN.C  A
F6EC 47          MOV      B,A
F6ED 0E0C        MVI      C,ROM
F6EF ED79        $      OT.C  A      :ROM MODE
F6F1 CF          RST      1          :CALL WBOOT2
;
F6F2            BOOT:
F6F2 011900      LXI      BC,ZSLOT
F6F5 ED78        $      IN.C  A      :RAM CHANGE
F6F7 47          MOV      B,A
F6F8 0E0D        MVI      C,RAM
F6FA ED79        $      OT.C  A      :CHANGE INTO RAM FROM ROM
;               :::::::::::::::::::::::BOOT:
F6FC FB          EI
F6FD 018000      LXI      B,80H
F700 CD01F8      CALL     SETDMA
F703 3A1DFA      LDA      ERFLAG
F706 A7          ANA      A
F707 280B ^F714$ JRZ      BOOT1
F709 3E00        MVI      A,0
F70B 3212FA      STA      SEKDSK
F70E 320400      STA      DRIVE
F711 3207FA      STA      DSKA      :NOMAL ROTATION
F714            BOOT1
F714 3A0EFA      LDA      DSKA
F717 E601        ANI      1
F719 280A ^F725$ JRZ      BOOT2

```

F71B	3A0EFA	LDA	DSKA	
F71E	E680	AND	80H	
F720	EE80	XOR	80H	:BIT CHANGE
F722	320EFA	STA	DSKA	
F725		BOOT2		
F725	3A0400	LDA	DRIVE	
F728	4F	MOV	C.A	
		:		
	= 0000	IF	TEST	
-		ADI	30H	
-		CALL	OUTCHR	
-		LXI	H.GOMES	
-		CALL	WHL	
-		MOV	A.C	
		ENDIF		
		:		
F729	C300E0	JMP	CCP	
		:		
	= 0000	IF	TEST	
-		GOMES	DB	' GO CPM'.0AH.0DH.0
		ENDIF		
		:		
F72C		LSTST:		
F72C	AF	XRA	A	
F72D	C9	RET		
		:		
F72E		LPTOUT		
F72E	59	MOV	E.C	
F72F	010203	LXI	B.PROUT	
F732	ED78	\$ LPTOT1:	IN.C	A
F734	A7	AND	A	
F735	20FB ^F732\$	JRNZ	LPTOT1	
F737	ED59	\$ OT.C	E	
F739	C9	RET		
		:		
F73A		CONST		
F73A	FB	EI		
F73B	3A68FC	LDA	KEYCNT	
F73E	A7	AND	A	
F73F	C8	RZ		
F740	3EFF	MVI	A.0FFH	
F742	C9	RET		
		:		
F743		CONIN:		
F743	FB	EI		
F744	3A68FC	LDA	KEYCNT	
F747	A7	AND	A	
F748	28F9 ^F743\$	JRZ	CONIN	
F74A	F3	DI		
F74B	E5	PUSH	HL	
F74C	C5	PUSH	BC	
F74D	3A68FC	LDA	KEYCNT	
F750	47	MOV	B.A	
F751	3A69FC	LDA	KEYDSP	
F754	90	SUB	B	
F755	3002 ^F759\$	JRNC	INCHR1	
F757	C620	ADI	MAXCHR	
F759		INCHR1		
F759	C66A	ADI	LOW RCHREG	
F75B	6F	MOV	L.A	
F75C	3E00	MVI	A.0	

```

F75E CEFC          ACI      RCHREG/256
F760 67            MOV      H.A
F761 46            MOV      B.M
F762 3A68FC        LDA      KEYCNT
F765 3D            DCR      A
F766 3268FC        STA      KEYCNT
F769 78            MOV      A.B
F76A FB            EI
F76B C1            POP      BC
F76C E1            POP      HL
F76D C9            RET
F76E 79            CONOUT    MOV      A.C
:
: Z80 OUTPUT ROUTINE for APPLE 80 character board
: '84 JAN.
F76F OUTCHR
F76F F5            PUSH     AF
F770 C5            PUSH     BC
F771 E5            PUSH     HL
: ST0             SP,SPREG
: LXI             SP,SPTOP2
: LXI             BC,ZSLOT
: IN.C            A          :ROM CHANGE
: MOV             B.A
: MVI             C.ROM
: OT.C            A
F772 CD27FA        CALL     OUTCH
: LXI             BC,ZSLOT
: IN.C            A          :RAM CHANGE
: MOV             B.A
: MVI             C.RAM
: OT.C            A
: LD              SP,SPREG
F775 E1            POP      HL
F776 C1            POP      BC
F777 F1            POP      AF
F778 C9            RET
:
F779 IRQW0:        :INCHR IRQ ROUTINE
F779 F5            PUSH     AF
F77A C5            PUSH     BC
F77B 010003        LXI      BC,INREG
F77E ED78          $ IN.C    A
F780 FE1C          CPI      CHANGY :CONTROL ¥
F782 200A ^F78E$  JRNZ     IRQW01
F784 3A0EFA        LDA      DSKA
F787 F601          OR       1
F789 320EFA        STA      DSKA
F78C 182B ^F7B9$  JR        IRQW02
F78E IRQW01
F78E A7            AND      A
F78F 2828 ^F7B9$  JRZ      IRQW02 :IGNORE NUL CODE
F791 E5            PUSH     HL
F792 D5            PUSH     DE
F793 57            MOV      D.A
F794 3A68FC        LDA      KEYCNT :KEY COUNT REG
F797 3C            INR      A
F798 FE20          CPI      MAXCHR
F79A 301B ^F7B7$  JRNZ     IRQW03 :OVERFLOW
F79C 3268FC        STA      KEYCNT
F79F 3A69FC        LDA      KEYDSP :KEY DISPLACEMENT REGISTER

```

F7A2	C66A	ADI	LOW RCHREG
F7A4	6F	MOV	L.A
F7A5	3E00	MVI	A.0
F7A7	CEFC	ACI	RCHREG/256
F7A9	67	MOV	H.A
F7AA	72	MOV	M.D
F7AB	3A69FC	LDA	KEYDSP
F7AE	3C	INR	A
F7AF	FE20	CPI	MAXCHR
F7B1	3801 ^F7B4\$	JRC	IRQW04 :RING BUFFER
F7B3	AF	XRA	A :MVI A.0
F7B4		IRQW04	
F7B4	3269FC	STA	KEYDSP
F7B7		IRQW03	
F7B7	D1	POP	D
F7B8	E1	POP	H
F7B9		IRQW02	
F7B9	97	SUB	A
F7BA	ED79	\$ OT.C	A
F7BC	C1	POP	BC
F7BD	F1	POP	AF
F7BE	FB	IRQRET	E1
F7BF	ED4D	\$ RETI	
		;	
	= 0000	IF	TEST
-		DSKMES	DB 'SELDISK :'.0
		ENDIF	
		;	
F7C1		SELDISK	
		;	
	= 0000	IF	TEST
-		PUSH	B
-		LXI	H.DSKMES
-		CALL	WHL
-		POP	B
-		MOV	A.C
-		ADI	30H
-		CALL	OUTCHR
		ENDIF	
		;	
F7C1	210000	LXI	H.0
F7C4	79	MOV	A.C
F7C5	FE06	CPI	DSKMAX
F7C7	D0	RNC	:DRIVE ERROR
F7C8	320400	STA	DRIVE
F7CB	3A0EFA	LDA	DSKA :NORMAL ROTATION?
F7CE	A7	AND	A
F7CF	F2DDF7	JP	SELD1 :MSB = 0 THEN NORMAL
F7D2	79	MOV	A.C
F7D3	C602	ADI	2 :DISK SHIFT
F7D5	4F	MOV	C.A
F7D6	FE06	CPI	DSKMAX
F7D8	3803 ^F7DD\$	JRC	SELD1
F7DA	D606	SUI	DSKMAX
F7DC	4F	MOV	C.A
F7DD		SELD1	
F7DD	79	MOV	A.C
F7DE	3212FA	STA	SEKDSK
F7E1	F5	PUSH	PSW
F7E2	CD0CF8	CALL	PRE :MAX SECTOR SET
F7E5	F1	POP	PSW
F7E6	2600	MVI	H.0

F7E8	87	ADD	A	
F7E9	87	ADD	A	
F7EA	87	ADD	A	
F7EB	87	ADD	A	: *16
F7EC	6F	MOV	L.A	
F7ED	1133F6	LXI	D.DPBASE	
F7F0	19	DAD	D	
F7F1	97	SUB	A	
F7F2	C9	RET		
;				
	= 0000	IF	TEST	
-		TRKMES	DB	SETTRK '.0
		ENDIF		
;				
F7F3		HOME	:SEEK HOME	
F7F3	0E00	MVI	C.0	
;				
F7F5		SETTRK:		
	= 0000	IF	TEST	
-		LXI	H.TRKMES	
-		CALL	WHL	
-		MOV	A.C	
-		ADI	30H	
-		CALL	OUTCHR	
		ENDIF		
;				
		SET TRACK IN ORDER TO BC		
F7F5	79	MOV	A.C	
F7F6	3213FA	STA	SEKTRK	
F7F9	97	SUB	A	
F7FA	C9	RET		
;				
	= 0000	IF	TEST	
-		SECMES	DB	SETSEC '.0
		ENDIF		
;				
F7FB		SETSEC:		
	= 0000	IF	TEST	
-		LXI	H.SECMES	
-		CALL	WHL	
-		MOV	A.C	
-		ADI	30H	
-		CALL	OUTCHR	
		ENDIF		
;				
		SET sector in order to C		
F7FB	79	MOV	A.C	
F7FC	3214FA	STA	SEKSEC	
F7FF	97	SUB	A	
F800	C9	RET		
;				
F801		SETDMA:		
;				
		Set DMA address in order to BC		
F801	60	MOV	H.B	
F802	69	MOV	L.C	
F803	2223FA	SHLD	DMAADR	
F806	97	SUB	A	
F807	C9	RET		
;				
F808		SECTAN:		
;				
F808	60	MOV	H.B	
F809	69	MOV	L.C	
F80A	23	INX	H	

F80B	C9		RET	
;				
F80C	3E00		PRE:	MVI A.0
F80E	321DFA		STA	ERFLAG
F811	3A12FA		LDA	SEKDSK
F814	FE02		CPI	2
F816	D8		RC	
F817	FE04		CPI	4
F819	3E29		MVI	A.41
F81B	3802 ^F81F\$		JRC	PRE1
F81D	3E21		MVI	A.33
F81F			PRE1:	
F81F	3222FA		STA	MAXSEC
F822	A7		ANA	A
F823	C9		RET	:CLEAR CY
;				
F824			MEMADDR:	
F824	3A13FA		LDA	SEKTRK
F827			MEMADR	
F827	A7		ANA	A
F828	0600		MVI	B.0
F82A			MEMAD1:	
F82A	1F		RAR	
F82B	CB18	\$	RR	B
F82D	1F		RAR	
F82E	CB18	\$	RR	B
F830	1F		RAR	
F831	CB18	\$	RR	B
F833	60		MOV	H.B
F834	47		MOV	B.A
F835	2E00		MVI	L.0
F837	3A14FA		LDA	SEKSEC
F83A	3D		DCR	A
F83B	A7		ANA	A
F83C	1F		RAR	
F83D	CB1D	\$	RR	L
F83F	84		ADD	H
F840	67		MOV	H.A
F841	C9		RET	
;				
F842			MEMWRT:	
F842	FE00		CPI	0
F844	2003 ^F849\$		JRNZ	DSKERR
F846	CD24F8		CALL	MEMADDR
F849			DSKERR:	
F849	3EFF		MVI	A.-1
F84B	A7		ANA	A
F84C	C9		RET	
;				
F84D			MEMRD:	:MEMORY READ
F84D	CD24F8		CALL	MEMADDR
F850	3A12FA		LDA	SEKDSK
F853	FE01		CPI	1
F855	2821 ^F878\$		JRZ	:RAM DISK
;				
F857	011900		LXI	BC.ZSLOT
F85A	ED78	\$	IN.C	A
F85C	47		MOV	B.A
F85D	0E0C		MVI	C.ROM
F85F	F3		DI	
F860	ED79	\$	OT.C	A
F862	ED5B23FA	\$	LD	DE.DMAADR

F866	018000		LXI	B,128	
F869	EDB0	\$	LDIR		
F86B	011900		LXI	BC,ZSLOT	
F86F	ED78	\$	IN.C	A	:RAM CHANGE
F870	47		MOV	B.A	
F871	0E0D		MVI	C.RAM	
F873	ED79	\$	OT.C	A	
F875	FB		EI		
F876	97		SUB	A	
F877	C9		RET		
F878			RAMRD:		
F878	3EFF		MVI	A,-1	
F87A	A7		ANA	A	
F87B	C9		RET		
		:			
		:			
	= 0000		IF	TEST	
-			RDMS	DB	' READ '.0
			ENDIF		
		:			
F87C			READ		
	= 0000		IF	TEST	
-			LXI	H,RDMES	
-			CALL	WHL	
			ENDIF		
			:READ:		
F87C	CD0CF8		CALL	PRE	
F87F	DA4DF8		JC	MEMRD	:ROM OR RAM READ
F882	AF		XRA	A	
F883	3220FA		STA	WRTYPE	
F886	3C		INR	A	
F887	321FFA		STA	READOP	
F88A	C3E6F8		JMP	ALLOC	
		:			
	= 0000		IF	TEST	
-			WRMES	DB	' WRITE '.0
			ENDIF		
		:			
F88D			WRITE		
	= 0000		IF	TEST	
-			LXI	H,WRMES	
-			CALL	WHL	
			ENDIF		
			:WRITE:		
F88D	CD0CF8		CALL	PRE	
F890	DA42F8		JC	MEMWRT	:ROM OR RAM WRITE
F893	97		SUB	A	
F894	321FFA		STA	READOP	
F897	79		MOV	A.C	
F898	3220FA		STA	WRTYPE	
F89B	FE02		CPI	WRUAL	
F89D	C2B7F8		JNZ	CHKUNA	
		:			
F8A0	3E10		MVI	A,BLKSIZE/128	
F8A2	3218FA		STA	UNACNT	
F8A5	3A12FA		LDA	SEKDSK	
F8A8	3219FA		STA	UNADSK	
F8AB	3A13FA		LDA	SEKTRK	
F8AE	321AFA		STA	UNATRK	
F8B1	3A14FA		LDA	SEKSEC	
F8B4	321BFA		STA	UNASEC	
F8B7			CHKUNA:		

F8B7	3A18FA	LDA	UNACNT	
F8BA	A7	AND	A	
F8BB	CAE6F8	JZ	ALLOC	
;				
F8BE	3D	DCR	A	
F8BF	3218FA	STA	UNACNT	
F8C2	1112FA	LXI	D.SEKDSK	
F8C5	2119FA	LXI	H.UNADSK	
F8C8	CDA6F9	CALL	COMP	
F8CB	C2E6F8	JNZ	ALLOC	
. ;				
F8CE	2B	DCX	H	
F8CF	34	INR	M	
F8D0	3A22FA	LDA	MAXSEC	
F8D3	BE	CMP	M	
F8D4	3009 ^F8DF\$	JRNC	NOOVF	
;				
F8D6	3601	MVI	M.1	:FIRST. SECTOR
F8D8	3A1AFA	LDA	UNATRK	
F8DB	3C	INR	A	
F8DC	321AFA	STA	UNATRK	
F8DF		NOOVF:		
F8DF	AF	XRA	A	
F8E0	321EFA	STA	RSFLAG	
F8E3	C3EEF8	JMP	RWOPER	
;				
F8E6		ALLOC:		
F8E6	97	SUB	A	
F8E7	3218FA	STA	UNACNT	
F8EA	3C	INR	A	
F8EB	321EFA	STA	RSFLAG	
F8EE		RWOPER:		
; SECTOR TRANSRATE ROUTINE				
;				
F8EE	3A12FA	LDA	SEKDSK	
F8F1	FE04	CPI	4	
F8F3	3A14FA	LDA	SEKSEC	
F8F6	3D	DCR	A	
F8F7	380A ^F903\$	JRC	SET1K	:1K SECTOR
; 256 SECTOR				
F8F9	F5	PUSH	PSW	
F8FA	E601	ANI	1	
F8FC	47	MOV	B.A	
F8FD	F1	POP	PSW	
F8FE	21CFF6	LXI	H.SKEW1	
F901	180C ^F90F\$	JR	RWOP1	
;				
F903		SET1K		: 1K SECTOR
F903	F5	PUSH	PSW	
F904	E607	ANI	7	
F906	47	MOV	B.A	
F907	F1	POP	PSW	
F908	21DFF6	LXI	H.SKEW0	
F90B	CB3F	\$ SRL	A	
F90D	CB3F	\$ SRL	A	
F90F		RWOP1:		
F90F	CB3F	\$ SRL	A	
F911	E60F	ANI	0FH	
F913	85	ADD	L	
F914	6F	MOV	L.A	
F915	3E00	MVI	A.0	
F917	8C	ADC	H	

F918	67		MOV	H.A
F919	7E		MOV	A.M
F91A	3211FA		STA	SEKHST
F91D	78		MOV	A.B
F91E	3225FA		STA	OFFSET
;				
F921	210FFA		LXI	H.HSTACT
F924	7E		MOV	A.M
F925	3601		MVI	M.1
F927	B7		ORA	A
F928	CA41F9		JZ	FILHST
;				
F92B	1111FA		LXI	D.SEKHST
F92E	2115FA		LXI	H.HSTSEC
F931	CDA6F9		CALL	COMP
F934	CA61F9		JZ	MATCH
F937		NOMATCH:		
F937	3A10FA		LDA	HSTWRT
F93A	B7		ORA	A
F93B	2804 ^F941\$		JRZ	FILHST
F93D	CDB0F9		CALL	WRITEHST
F940	C0		RNZ	;WRITE ERROR
;				
F941		FILHST:		
F941	3A12FA		LDA	SEKDSK
F944	3216FA		STA	HSTDSK
F947	3A13FA		LDA	SEKTRK
F94A	3217FA		STA	HSTTRK
F94D	3A11FA		LDA	SEKHST
F950	3215FA		STA	HSTSEC
F953	3A1EFA		LDA	RSFLAG
F956	B7		ORA	A
F957	C4BCF9		CNZ	READHST
F95A	F5		PUSH	AF
F95B	AF		XRA	A
F95C	3210FA		STA	HSTWRT
F95F	F1		POP	AF
F960	C0		RNZ	;READ ERROR
F961		MATCH:		
F961	3A25FA		LDA	OFFSET
F964	2E00		MVI	L.0
F966	CB3F	\$	SRL	A
F968	CB1D	\$	RR	L
F96A	67		MOV	H.A
F96B	010018		LXI	B.DBUF
F96E	09		DAD	B
F96F	44		MOV	B.H
F970	4D		MOV	C.L
;				
F971	2A23FA		LHLD	DMAADR
F974	1E80		MVI	E.128
F976	3A1FFA		LDA	READOP
F979	B7		ORA	A
F97A	C29AF9		JNZ	RMOVE
; WRITE MOVE				
F97D	3C		INR	A
F97E	3210FA		STA	HSTWRT
F981	7E		WMOVE:	MOV A.M
F982	ED79	\$	OT.C	A
F984	23		INX	H

F985	03		INX	B	
F986	1D		DCR	E	
F987	20F8 ^F981\$		JRNZ	WMOVE	
F989	3A20FA		LDA	WRTYPE	
F98C	FE01		CPI	WRDIR	
F98E	3A1DFA		LDA	ERFLAG	
F991	C0		RNZ		
;					
F992	B7		ORA	A	
F993	C0		RNZ		
F994	3210FA		STA	HSTWRT	
F997	C3B0F9		JMP	WRITEHST	
;					
F99A			RMOVE:		
F99A	ED78	\$	IN,C	A	
F99C	77		MOV	M.A	
F99D	23		INX	H	
F99E	03		INX	B	
F99F	1D		DCR	E	
F9A0	20F8 ^F99A\$		JRNZ	RMOVE	
F9A2	3A1DFA		LDA	ERFLAG	
F9A5	C9		RET		
;					
F9A6			COMP:		
F9A6	0603		MVI	B.3	
F9A8	1A		COMP1:	LDAX	D
F9A9	BE		CMP	M	
F9AA	C0		RNZ		
F9AB	13		INX	D	
F9AC	23		INX	H	
F9AD	10F9 ^F9A8\$		DJNZ	COMP1	
F9AF	C9		RET		
F9B0			WRITEHST:		
F9B0	3E02		MVI	A.2	:SET READ MODE
F9B2	CDCDF9		CALL	RWOPHT	
F9B5	CDC1F9		CALL	RDHST1	
F9B8	3210FA		STA	HSTWRT	
F9BB	C9		RET		
;					
F92C			READHST:		
F9BC	2501		MVI	A,1	:WRITE MODE
F9BE	CDCDF9		CALL	RWOPHT	
F9C1	FB		RDHST1:	EI	
F9C2	3A21FA		LDA	FDCSTR	
F9C5	A7		ANA	A	
F9C6	28F9 ^F9C1\$		JRZ	RDHST1	
F9C8			RDHST2:		
F9C8	3A1DFA		LDA	ERFLAG	
F9CB	B7		ORA	A	
F9CC	C9		RET		
;					
F9CD			RWOPHT:		
F9CD	F5		PUSH	AF	
F9CE	97		SUB	A	
F9CF	3221FA		STA	FDCSTR	
F9D2	011500		LXI	B.FDADR+1	
F9D5	210018		LXI	H.DBUF	
F9D8	ED61	\$	OT.C	H	
F9DA	0B		DCX	B	
F9DB	ED69	\$	OT.C	L	

F9DD	0B		DCX	B	
F9DE	3A15FA		LDA	HSTSEC	
F9E1	ED79	\$	OT.C	A	
F9E3	0B		DCX	B	
F9E4	3A17FA		LDA	HSTTRK	
F9E7	ED79	\$	OT.C	A	
F9E9	0B		DCX	B	
F9EA	3A16FA		LDA	HSTDSK	
F9ED	ED79	\$	OT.C	A	
F9EF	0B		DCX	B	
F9F0	F1		POP	PSW	
F9F1	ED79	\$	OT.C	A	
F9F3	C9		RET		
;					
F9F4			IRQW2	;FD READY	IRQ
F9F4	F5		PUSH	PSW	
F9F5	C5		PUSH	B	
F9F6	011600		LXI	B,FDFLG	
F9F9	ED78	\$	IN.C	A	
F9FB	321DFA		STA	ERFLAG	
F9FE	03		INX	B	
F9FF	ED78	\$	IN.C	A	
FA01	321CFA		STA	RETRY	
FA04	3E01		MVI	A,1	
FA06	3221FA		STA	FDCSTR	
FA09	C1		POP	B	
FA0A	F1		POP	PSW	
FA0B	FB		EI		
FA0C	ED4D	\$	RETI		
;					
FA0E	00		DSKA	DB	0
;					
FA0F	00		HSTACT:	DB	0
FA10	00		HSTWRT:	DB	0
;					
FA11	01		SEKHST:	DB	1
FA12	00		SEKDSK:	DB	0
FA13	00		SEKTRK:	DB	0
FA14	01		SEKSEC:	DB	1
;					
FA15	01		HSTSEC:	DB	1
FA16	00		HSTDSK:	DB	0
FA17	00		HSTTRK:	DB	0
;					
FA18	00		UNACNT:	DB	0
FA19	= 0001		UNADSK:	DS	1
FA1A	= 0001		UNATRK:	DS	1
FA1B	= 0001		UNASEC:	DS	1
;					
FA1C	00		RETRY:	DB	0
FA1D	00		ERFLAG:	DB	0
FA1E	00		RSFLAG:	DB	0
FA1F	01		READOP:	DB	1
FA20	00		WRTYPE:	DB	0
FA21	00		FDCSTR	DB	0
FA22	21		MAXSEC:	DB	21H
FA23	8000		DMAADR:	DW	80H
FA25	0000		OFFSET	DW	0
;					
FA27			OUTCH:		

FA27	4F		MOV	C.A
FA28	3A67FC		LDA	APSFLG
FA2B	A7		ANA	A
FA2C	C2B8FB		JNZ	APSX
FA2F	79		MOV	A.C
FA30	0100C3		LXI	BC.VRAMON
FA33	ED48	\$	IN.C	C
FA35	ED4B63FC	\$	LD	BC.CHBANK
FA39	ED48	\$	IN.C	C
FA3B	ED4B61FC	\$	LD	BC.CHADDR
FA3F	FE20		CPI	20H
FA41	DA50FB		JC	CONT
FA44	FE80		CPI	80H
FA46	3806 ^FA4E\$		JRC	OUTCH1
FA48	FEA0		CPI	0A0H
FA4A	3002 ^FA4E\$		JRNC	OUTCH1
FA4C	E67F		AND	7FH
FA4E	ED79	\$	OUTCH1	OT.C A
FA50			APF:	
FA50	2A65FC		LHLD	ADDR
FA53	23		INX	HL
FA54	2265FC		SHLD	ADDR
FA57	0C		INR	C :NEXT ADDRESS
FA58	2014 ^FA6E\$		JRNZ	APF1 :IF BANKEND THEN
FA5A	04		INR	B :..BANK END CHECK
FA5B	78		MOV	A.B :..BANK BC >0CC00H,<0CE00H
FA5C	FECE		CPI	VRAME/256 :0CEH
FA5E	200E ^FA6E\$		JRNZ	APF1 :BANK END
FA60	06CC		MVI	B.VRAM/256 :0CCH
FA62	3A63FC		LDA	CHBANK :OLD BANK
FA65	C604		ADI	4 :NEXT BANK
FA67	E60F		ANI	0FH :
FA69	C6B0		ADI	VSLOT*10H+80H
FA6B	3263FC		STA	CHBANK
FA6E	ED4361FC	\$	APF1:	STO BC.CHADDR
FA72	3A5EFC		LDA	APX
FA75	3C		INR	A
FA76	325EFC		STA	APX
FA79	FE50		CPI	80 :TEST END OF LINE?
FA7B	DA94FA		JC	SETCUR
:				
FA7E	97		CRLF1:	SUB A
FA7F	325EFC		STA	APX
FA82	2A5FFC		APD LHLD	APY
FA85	7C		MOV	A.H
FA86	FE07		CPI	7
FA88	284B ^FAD5\$		JRZ	SCROL
FA8A	015000		LXI	B.80
FA8D	09		DAD	B
FA8E	225FFC		SHLD	APY
FA91	CDAafa		SETADR	CALL GETAD
FA94			SETCUR:	
FA94	2A65FC		LHLD	ADDR
FA97	01B0C0		LXI	BC.BANK0
FA9A	3E0F		MVI	A.CURL
FA9C	ED79	\$	OT.C	A
FA9E	03		INX	BC
FA9F	ED69	\$	OT.C	L
FAA1	0B		DCX	BC
FAA2	3E0E		MVI	A.CURH

FAA4	ED79	\$	OT.C	A	
FAA6	03		INX	BC	
FAA7	ED61	\$	OT.C	H	
FAA9	C9		RET		
;					
FAAA	F5		GETAD:	PUSH	AF
FAAB	2A5FFC		LHLD	APY	
FAAE	3A5EFC		LDA	APX	
FAB1	4F		MOV	C.A	
FAB2	0600		MVI	B,0	
FAB4	09		DAD	B	
FAB5	ED4B5CFC	\$	LD	BC.VTOP	
FAB9	09		DAD	B	
FABA	2265FC		SHLD	ADDR	
FABD	7C		MOV	A.H	
FABE	E606		ANI	06H	
FAC0	87		ADD	A	
FAC1	C6B0		ADI	VSLOT*10H+080H	
FAC3	4F		MOV	C.A	
FAC4	06C0		MVI	B,0C0H	
FAC6	ED4363FC	\$	STO	BC.CHBANK	
FACA	7C		MOV	A.H	
FACB	E601		ANI	1	
FACD	C6CC		ADI	0CCH	
FACF	67		MOV	H.A	
FAD0	2261FC		SHLD	CHADDR	
FAD3	F1		POP	AF	
FAD4	C9		RET		
;					
FAD5			SCROL:		
FAD5	2A5CFC		LHLD	VTOP	
FAD8	015000		LXI	B.80	
FADB	09		DAD	B	
FADC	225CFC		SHLD	VTOP	
FADF	01B0C0		LXI	BC.BANK0	
FAE2	3E0C		MVI	A,STADH	
FAE4	ED79	\$	OT.C	A	
FAE6	03		INX	BC	
FAE7	ED61	\$	OT.C	H	
FAE9	0B		DCX	BC	
FAEA	3E0D		MVI	A,STADL	
FAEC	ED79	\$	OT.C	A	
FAEE	03		INX	BC	
FAEF	ED69	\$	OT.C	L	
FAF1	215EFC		LXI	HL,APX	
FAF4	7E		MOV	A.[HL]	
FAF5	A7		AND	A	
FAF6	2810 ^FB08\$		JRZ	SCR1	
FAF8	F5		PUSH	AF	
FAF9	3600		MVI	[HL].0	
FAFB	CDAafa		CALL	GETAD	
FAFE	CD0BFB		CALL	EOL	
FB01	F1		POP	AF	
FB02	325EFC		STA	APX	
FB05	C391FA		JMP	SETADR	
;					
FB08	CD91FA		SCR1	CALL	SETADR
;					
FB0B	0100C3		EOL:	LXI	BC.VRAMON
FB0E	ED78	\$	IN.C	A	

FB10	ED4B63FC	\$	LD	BC,CHBANK	
FB14	ED78	\$	IN.C	A	
FB16	ED4B61FC	\$	LD	BC,CHADDR	
FB1A	3A5EFC		LDA	APX	
FB1D	D650		SUI	80	
FB1F	6F		MOV	L,A	
FB20	3E20		MVI	A,'	
FB22	3802 ^FB26\$		JRC	EOL1	
FB24	2EB0		MVI	L,-80	
FB26	ED79	\$	EOL1:	OT.C	A
FB28	0C		INR	C	
FB29	201A ^FB45\$		JRNZ	EOL2	
FB2B	04		INR	B	
FB2C	78		MOV	A,B	
FB2D	FECE		CPI	VRAME/256	
FB2F	2012 ^FB43\$		JRNZ	EOL3	
FB31	06CC		MVI	B.VRAM/256	
FB33	C5		PUSH	BC	
FB34	ED4B63FC	\$	LD	BC,CHBANK	
FB38	79		MOV	A,C	
FB39	C604		ADI	4	
FB3B	E60F		AND	0FH	
FB3D	C6B0		ADI	V SLOT*10H+80H	
FB3F	4F		MOV	C,A	
FB40	ED78	\$	IN.C	A	
FB42	C1		POP	BC	
FB43	3E20		EOL3	MVI	A,20H
FB45	2C		EOL2:	INR	L
FB46	20DE ^FB26\$		JRNZ	EOL1	
FB48	C9		RET		
;					
FB49	97		APR:	SUB	A
FB4A	325EFC		STA	APX	
FB4D	C391FA		JMP	SETADR	
;					
FB50	FE1D		CONT:	CPI	01DH
FB52	CA0BFB		JZ	EOL	
FB55	FE1E		CPI	1EH	
FB57	CAB2FB		JZ	APS	
FB5A	D607		SUI	7	
FB5C	CAA2FB		JZ	BEL	
FB5F	3D		DCR	A	
FB60	CA79FB		JZ	APB	
FB63	3D		DCR	A	
FB64	CA50FA		JZ	APF	:9
FB67	3D		DCR	A	
FB68	CA82FA		JZ	APD	
FB6B	3D		DCR	A	
FB6C	CA88FB		JZ	APU	
FB6F	3D		DCR	A	
FB70	3D		DCR	A	
FB71	CA49FB		JZ	APR	:13
FB74	3D		DCR	A	
FB75	CAFEFB		JZ	CLEAR	:14 CLR=CS
FB78	C9		RET		
;					
FB79	3A5EFC		APB LDA	APX	
FB7C	3D		DCR	A	
FB7D	325EFC		STA	APX	
FB80	F291FA		JP	SETADR	

FB83	3E4F	MVI	A.79	
FB85	325EFC	STA	APX	
FB88	2A5FFC	APU:	LHLD	APY
FB8B	7D	MOV	A.L	
FB8C	B4	OR	H	
FB8D	2009 ^FB98\$	JRNZ	APU1	
FB8F	213007	LXI	HL,80*23	
FB92	225FFC	SHLD	APY	
FB95	C391FA	JMP	SETADR	
FB98		APU1		
FB98	01B0FF	LXI	BC,-80	
FB9B	09	DAD	B	
FB9C	225FFC	SHLD	APY	
FB9F	C391FA	JMP	SETADR	
;				
FBA2	0130C0	BEL:	LXI	BC,BELREG
FBA5	26C8	MVI	H.200	
FBA7	2E32	BEL1	MVI	L.50
FBA9	2D	BEL2	DCR	L
FBAA	20FD ^FBA9\$	JRNZ	BEL2	
FBAC	ED78 \$	IN,C	A	
FBAE	25	DCR	H	
FBAF	20F6 ^FBA7\$	JRNZ	BEL1	
FBB1	C9	RET		
;				
FBB2	3E01	APS:	MVI	A.1
FBB4	3267FC	STA	APSFLG	
FBB7	C9	RET		
;				
FBB8	FE01	APSX	CPI	1
FBBA	79	MOV	A.C	
FBBD	200F ^FBCC\$	JRNZ	APS2	
FBBD	D620	APS1	SUI	20H
FBBF	D8	RC		
FBC0	FE50	CPI	80	
FBC2	D0	RNC		
FBC3	325EFC	STA	APX	
FBC6	3E02	MVI	A.2	
FBC8	3267FC	STA	APSFLG	
FBCB	C9	RET		
;				
FBCC	D620	APS2	SUI	20H
FBCE	D8	RC		
FBCF	FE18	CPI	24	
FBD1	D0	RNC		
FBD2	6F	MOV	L.A	
FBD3	2600	MVI	H.0	
FBD5	29	DAD	H	
FBD6	29	DAD	H	*4
FBD7	29	DAD	H	*8
FBD8	29	DAD	H	*16
FBD9	4D	MOV	C.L	
FBD A	44	MOV	B.H	
FBD B	29	DAD	H	*32
FBD C	29	DAD	H	*64
FBD D	09	DAD	B	*80
FBDE	225FFC	SHLD	APY	
FBE1	97	SUB	A	
FBE2	3267FC	STA	APSFLG	
FBE5	C391FA	JMP	SETADR	

FBE8	CD6FF7	:	WHL1:	CALL	OUTCHR
FBEB	7E	:	WHL:	MOV	A.[HL]
FBEC	23		INX	HL	
FBED	A7		AND	A	
FBEE	20F8 ^FBE8\$		JRNZ	WHL1	
FBF0	C9		RET		
;					
FBF1	F5	:	CRLF:	PUSH	AF
FBF2	3E0D		MVI	A.00DH	
FBF4	CD6FF7		CALL	OUTCHR	
FBF7	3E0A		MVI	A.0AH-	
FBF9	CD6FF7		CALL	OUTCHR	
FBFC	F1		POP	AF	
FBFD	C9		RET		
;					
FBFE		:	CLEAR:		
FBFE	11B0C0		LXI	DE,BANK0	
FC01	0100C3		LXI	BC,VRAM0	
FC04	ED78	\$	IN,C	A	
FC06	4B		CLR0:	MOV	C.E
FC07	42		MOV	B,D	
FC08	ED78	\$	IN,C	A	
FC0A	0100CC		LXI	BC,VRAM	
FC0D	3E20		CLR2:	MVI	A,20H
FC0F	ED79	\$	CLR1:	OT.C	A
FC11	0C		INR	C	
FC12	20FB ^FC0F\$		JRNZ	CLR1	
FC14	04		INR	B	
FC15	78		MOV	A,B	
FC16	FECE		CPI	VRAME/256	
FC18	20F3 ^FC0D\$		JRNZ	CLR2	
FC1A	7B		MOV	A,E	
FC1B	C604		ADI	4	
FC1D	5F		MOV	E,A	
FC1E	E60F		ANI	0FH	
FC20	20E4 ^FC06\$		JRNZ	CLR0	
;					
FC22	01B0C0		INIT	LXI	BC.CRTC0
FC25	215BFC		LXI	HL.INTBL+15	
FC28	1E0F		MVI	E,15	
FC2A	ED59	\$	INIT1	OT,C	E
FC2C	03		INX	BC	
FC2D	7E		MOV	A,M	
FC2E	ED79	\$	OT,C	A	
FC30	0B		DCX	BC	
FC31	2B		DCX	HL	
FC32	1D		DCR	E	
FC33	F22AFC		JP	INIT1	
FC36	210000		LXI	H,0	
FC39	225CFC		SHLD	VTOP	
FC3C	225FFC		SHLD	APY	
FC3F	97		SUB	A	
FC40	3267FC		STA	APSFLG	
FC43	325EFC		STA	APX	
FC46	CDAafa		CALL	GETAD	
FC49	C394FA		JMP	SETCUR	
;					
FC4C	7B5061291B	:	INTBL	DB	7BH.50H.61H.29H.1BH.8.18H.1AH.
;					
0.8.0C0H,8.0.0.0.0					

FC5C	0000	VTOP	DW	0	
FC5E	00	APX DB	0		
FC5F	0000	APY DW	0		
FC61	00CC	CHADDR	DW	0CC00H	
FC63	B0C0	CHBANK	DW	0C080H+VSLOT*10H	
FC65	0000	ADDR	DW	0	
FC67	00	APSFLG	DB	0	
		:			
FC68	00	KEYCNT	DB	0	
FC69	00	KEYDSP	DB	0	
FC6A	= 0020	RCHREG	DS	MAXCHR	
		:			
FC8A	= 0002	ALV0:	DS	2	:DSM/8+1
FC8C	= 0008	CSV0:	DS	8	:(DRM+1)/4
FC94	= 0009	ALV1	DS	9	
FC9D	= 0008	CSV1	DS	8	
FCA5	= 0032	ALV2	DS	50	:400 BLOCKS
FCD7	= 0020	CSV2	DS	32	:128 ENTRY
FCF7	= 0032	ALV3	DS	50	
FD29	= 0020	CSV3	DS	32	
FD49	= 0028	ALV4	DS	40	:316 BLOCKS
FD71	= 0020	CSV4	DS	32	:128 DIR
FD91	= 0028	ALV5	DS	40	
FDB9	= 0020	CSV5	DS	32	
		:			
FDD9	= 0080	DIRBUF:	DS	128	
FE59		ENDBIOS			
FE59	FFFFFFFF	DW	-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1		
		:			
FE77	= FFC0	ORG	0FFC0H		
FFC0	79F7BEF7F4	IRQTB	DW	IRQW0.IRQRET.IRQW2.IRQRET.IRQRET.IRQRET.IRQRET.IRQRET	
FFD0	BEF7BEF7BE	DW	IRQRET.IRQRET.IRQRET.IRQRET.IRQRET.IRQRET.IRQRET.IRQRET		
FFE0	BEF7BEF7BE	DW	IRQRET.IRQRET.IRQRET.IRQRET.IRQRET.IRQRET.IRQRET.IRQRET		
FFF0	BEF7BEF7BE	DW	IRQRET.IRQRET.IRQRET.IRQRET.IRQRET.IRQRET.IRQRET.IRQRET		
		:			
0000	= 0000	ORG	0	:5400H	
		:			
0000		RESET:			
0000	31A0FF	LXI	SP,SPTOP		
0003	0600	MVI	B,0		
0005	1804 ^000B\$	JR	LOOP1		
0007	= 0008	ORG	8H	:RST 1	
0008	C35E00	JMP	WBOOT2		
000B	00	LOOP1:	NOP		
000C	10FD ^000B\$	DJNZ	LOOP1		
000E		LOOP2:			
000E	F5	PUSH	AF		
000F	F1	POP	AF		
0010	10FC ^000E\$	DJNZ	LOOP2		
0012	014000	LXI	B,40H		
0015	21C01F	LXI	H,01FC0H		
0018	11C0FF	LXI	D,SPTOP2		
001B	EDB0	\$	LDIR	:IRQ ENTRY TABLE SET	
001D	01A01F	LXI	B,SPTOP-CCP		
0020	210001	LXI	H,DTOP		
0023	1100E0	LXI	D,CCP		
0026	EDB0	\$	LDIR	:CPM COPY	
0028	97	SUB	A		
0029	3268FC	STA	KEYCNT		
002C	3269FC	STA	KEYDSP		

```

002F 3EFF          MVI    A,SPTOP2/256    :INTERRUPT TABLE
0031 ED47          $      MOV    I,A
0033 ED5E          $      IM2
0035 FB            EI
0036 218600        ENTRY1: LXI    H,KEYWORD
0039 CD09F6        ENTRY2: CALL   INCHR
003C BE            CMP    M
003D 23            INX    H
003E 20F6 ^0036$   JRNZ   ENTRY1
0040 7E            MOV    A,M
0041 A7            AND    A
0042 20F5 ^0039$   JRNZ   ENTRY2
;
0044 CDFEFB        CALL   CLEAR
0047 218D00        LXI    H,MESSAGE
004A CDEBFB        CALL   WHL              :MONITOR
004D AF            XRA    A              :DRIVE 0 SELECT
004E 320EFA        STA    DSKA
0051 320300        STA    IOBYTE
0054 4F            MOV    C,A
0055 CD1BF6        CALL   SELFDD
0058 CD6900        CALL   GOCPM1          :CP/M BOOT
005B C300F6        JMP    BIOS
;
005E 010016        WBOOT2  LXI    B,BIOS-CCP ;WARM BOOT SUBROUTINE
0061 210001        LXI    H,DTOP
0064 1100E0        LXI    D,CCP
0067 EDB0          $      LDIR
0069              GOCPM1
0069 F3            DI
006A 3EFF          MVI    A,SPTOP2/256    :INTERRUPT TABLE
006C ED47          $      MOV    I,A
006E ED5E          $      IM2
0070 FB            EI
0071 3EC3          MVI    A,0C3H          :JMP OPCODE
0073 320000        STA    0
0076 320500        STA    5
0079 2103F6        LXI    H,WBOOTE
007C 220100        SHLD   1
007F 2106E8        LXI    H,BDOS+6
0082 220600        SHLD   6
0085 C9            RET
;
0086              KEYWORD:
0086 5354415254     DB      'START!'.0
;
008D              MESSAGE:
008D              MESDB
;
;
00BC = 0100        ORG    100H
0100              DTOP:
;
0100              END

```

no ERRORS. 226 Labels, 9795h bytes not used. Program LWA = 0100h.

	ADDR	FC65	15/20	15/22	15/53	16/14	20# 6		
n	AL0	F69C	3#20						
n	AL1	F69D	3#21						
	ALLOC	F8E6	10/29	10/61	11/ 8	11#25			
	ALV0	FC8A	2/41	20#13					
	ALV1	FC94	2/47	20#15					
	ALV2	FCA5	2/53	20#17					
	ALV3	FCF7	2/59	20#19					
	ALV4	FD49	3/ 5	20#21					
	ALV5	FD91	3/11	20#23					
	APB	FB79	17/43	17#57					
	APD	FA82	15#44	17/47					
	APF	FA5Q	15#19	17/45					
	APF1	FA6E	15/24	15/28	15#35				
	APR	FB49	17#32	17/52					
	APS	FBB2	17/39	18#26					
n	APS1	FBBD	18#33						
	APS2	FBCC	18/32	18#42					
	APSFLG	FC67	15/ 2	18/27	18/39	18/59	19/54	20# 7	
	APSX	FBB8	15/ 4	18#30					
	APU	FB88	17/49	18# 3					
	APU1	FB98	18/ 6	18#10					
	APX	FC5E	15/36	15/38	15/43	16/ 8	16/45	16/54	17/ 4
			17/33	17/57	17/59	18/ 2	18/37	19/55	20# 2
	APY	FC5F	15/44	15/50	16/ 7	18/ 3	18/ 8	18/13	18/57
			19/52	20# 3					
	BANK0	C0B0	1#42	15/54	16/35	19/18			
n	BANK1	C0B4	1#43						
n	BANK2	C0B8	1#44						
n	BANK3	C0BC	1#45						
	BDOS	E800	1#22	1/23	21/40				
	BEL	FBA2	17/41	18#16					
	BEL1	FBA7	18#18	18/23					
	BEL2	FBA9	18#19	18/20					
	BELREG	C030	1#49	18/16					
	BIOS	F600	1#21	1/22	2/ 9	21/23	21/25		
	BLKSIZ	0800	2# 7	10/50					
n	BLM	F696	3#16						
	BOOT	F6F2	2/14	4#27					
	BOOT1	F714	4/39	4#44					
	BOOT2	F725	4/47	4#52					
n	BSH	F695	3#15						
	CCP	E000	1#23	5/ 4	20/54	20/56	21/25	21/27	
	CHADDR	FC61	15/10	15/35	16/26	17/ 3	20# 4		
	CHANGY	001C	1#30	6/33					
	CHBANK	FC63	15/ 8	15/30	15/34	16/21	16/61	17/19	20# 5
	CHKUNA	F8B7	10/48	10#58					
n	CKS	F69E	3#22						
	CLEAR	FBFE	17/54	19#17	21/14				
	CLR0	FC06	19#21	19/37					
	CLR1	FC0F	19#26	19/28					
	CLR2	FC0D	19#25	19/32					
	COMP	F9A6	11/ 7	12/15	13#24				
	COMP1	F9A8	13#26	13/31					
	CONIN	F743	2/17	5#31	5/35				
	CONOUT	F76E	2/18	5#61					
	CONST	F73A	2/16	5#23					
	CONT	FB50	15/12	17#36					
	CR	000D	1#28	21/49					
n	CRLF	FBF1	19# 9						

n	CRLF1	FA7E	15#42						
	CRTC0	C0B0	1#40	1/41	1/42	1/43	1/44	1/45	19/39
n	CRTC1	C0B1	1#41						
	CSV0	FC8C	2/41	20#14					
	CSV1	FC9D	2/47	20#16					
	CSV2	FCD7	2/53	20#18					
	CSV3	FD29	2/59	20#20					
	CSV4	FD71	3/ 5	20#22					
	CSV5	FDB9	3/11	20#24					
	CURH	000E	1#50	15/60					
	CURL	000F	1#51	15/55					
	DBUF	1800	1#37	12/45	13/57				
n	DBUF2	2C00	1#38						
	DIRBUF	FDD9	2/40	2/46	2/52	2/58	3/ 4	3/10	20#26
	DMAADR	FA23	8/46	9/56	12/50	14#57			
	DPB0	F693	2/40	3#13					
	DPB1	F6A2	2/46	3#25					
	DPB2	F6C0	2/52	2/58	3#49				
	DPB4	F6B1	3/ 4	3/10	3#37				
n	DPBAS0	F633	2#37						
n	DPBAS1	F643	2#43						
n	DPBAS2	F653	2#49						
n	DPBAS3	F663	2#55						
n	DPBAS4	F673	2#61						
n	DPBAS5	F683	3# 7						
	DPBASE	F633	2#36	7/58					
	DRIVE .	0004	1#26	4/42	4/53	7/35			
n	DRM	F69A	3#19						
	DSKA	FA0E	1/43	4/45	4/48	4/51	6/35	6/37	7/36
			14#31	21/18					
	DSKERR	F849	9/37	9#39					
	DSKMAX	0006	1#16	7/33	7/42	7/44			
n	DSM	F698	3#18						
	DTOP	0100	20/55	21/26	21#52				
n	ENDBIO	FE59	20#27						
	ENTRY1	0036	21# 5	21/ 9					
	ENTRY2	0039	21# 6	21/12					
	EOL	FB0B	16/52	16#59	17/37				
	EOL1	FB26	17/ 8	17#10	17/29				
	EOL2	FB45	17/12	17#28					
	EOL3	FB43	17/16	17#27					
	ERFLAG	FA1D	4/37	8/58	13/ 6	13/21	13/48	14/20	14#51
n	EXM	F697	3#17						
	FDADR	0014	2# 3	13/56					
	FDCSTR	FA21	13/44	13/55	14/25	14#55			
n	FDDSK	0011	1#60						
	FDFLG	0016	2# 4	14/18					
n	FDOP	0010	1#59						
n	FDRTY	0017	2# 5						
n	FDSEK	0013	2# 2						
n	FDTRK	0012	1#61						
	FILHST	F941	12/11	12/20	12#24				
	GETAD	FAAA	15/51	16# 6	16/51	19/56			
	GOCPM1	0069	21/22	21#29					
	HOME	F7F3	2/26	8# 7					
	HSTACT	FA0F	12/ 7	14#33					
	HSTDSK	FA16	12/26	14/ 8	14#42				
	HSTSEC	FA15	12/14	12/30	14/ 2	14#41			
	HSTTRK	FA17	12/28	14/ 5	14#43				
	HSTWRT	FA10	12/18	12/36	12/57	13/11	13/37	14#34	

	INCHR	F609	2#17	21/ 6					
	INCHR1	F759	5/43	5#45					
n	INIT	FC22	19#39						
	INIT1	FC2A	19#42	19/49					
	INREG	0300	1#33	6/31					
	INTBL	FC4C	19/40	19#59					
	IOBYTE	0003	1#25	21/19					
	IRQRET	F7BE	7#12	20/31	20/31	20/31	20/31	20/31	20/31
			20/32	20/32	20/32	20/32	20/32	20/32	20/32
			20/32	20/33	20/33	20/33	20/33	20/33	20/33
			20/33	20/33	20/34	20/34	20/34	20/34	20/34
			20/34	20/34					
n	IRQTBL	FFC0	20#31						
	IRQW0	F779	6#28	20/31					
	IRQW01	F78E	6/34	6#39					
	IRQW02	F7B9	6/38	6/41	7# 7				
	IRQW03	F7B7	6/48	7# 4					
	IRQW04	F7B4	6/60	7# 2					
	IRQW2	F9F4	14#15	20/31					
	KEYCNT	FC68	5/25	5/33	5/39	5/52	5/54	6/45	6/49
			20# 9	20/59					
	KEYDSP	FC69	5/41	6/50	6/57	7/ 3	20#10	20/60	
	KEYWOR	0086	21/ 5	21#44					
	LF	000A	1#29	21/49					
	LOOP1	000B	20/41	20#44	20/45				
	LOOP2	000E	20#46	20/49					
	LPTOT1	F732	5#17	5/19					
	LPTOUT	F72E	2/19	5#14					
	LSTST	F72C	2/33	5#10					
	MATCH	F961	12/16	12#39					
	MAXCHR	0020	1#31	5/44	6/47	6/59	20/11		
	MAXSEC	FA22	9/ 7	11/12	14#56				
n	MEMAD1	F82A	9#16						
	MEMADD	F824	9#11	9/38	9/45				
n	MEMADR	F827	9#13						
	MEMRD	F84D	9#44	10/24					
	MEMWRT	F842	9#35	10/42					
	MESDB	mac	1#10	21/49					
	MESSAG	008D	21/15	21#47					
	MSIZE	003F	1# 6	1/20					
n	NOMATC	F937	12#17						
	NOOVF	F8DF	11/14	11#20					
n	OFF	F6A0	3#23						
	OFFSET	FA25	12/ 5	12/40	14#58				
	OUTCH	FA27	6/16	14#60					
	OUTCH1	FA4E	15/14	15/16	15#18				
	OUTCHR	F76F	6# 5	19/ 2	19/11	19/13			
	PRE	F80C	7/50	8#57	10/23	10/41			
	PRE1	F81F	9/ 4	9# 6					
	PROUT	0302	1#34	5/16					
n	PUNCH	F612	2#21						
	RAM	000D	1#36	4/31	10/ 2				
	RAM\$TO	FC00	1#20	1/21					
	RAMRD	F878	9/48	10# 7					
	RCHREG	FC6A	5/46	5/49	6/51	6/54	20#11		
	RDHST1	F9C1	13/36	13#43	13/46				
n	RDHST2	F9C8	13#47						
	READ	F87C	2/31	10#17					
n	READER	F615	2#24						
	READHS	F9BC	12/33	13#40					

	READOP	FA1F	10/28	10/44	12/52	14#53			
n	RESET	0000	20#38						
	RETRY	FA1C	14/23	14#50					
	RMOVE	F99A	12/54	13#14	13/20				
	ROM	000C	1#35	4/23	9/53				
	RSFLAG	FA1E	11/22	11/29	12/31	14#52			
	RWOP1	F90F	11/44	11#54					
	RWOPER	F8EE	11/23	11#30					
	RWOPHT	F9CD	13/35	13/42	13#52				
	SCR1	FB08	16/48	16#57					
	SCROL	FAD5	15/47	16#30					
	SECTRA	F808	2/34	8#50					
	SEKDSK	FA12	4/41	7/48	8/59	9/46	10/52	11/ 5	11/33
			12/25	14#37					
	SEKHST	FA11	12/ 3	12/13	12/29	14#36			
	SEKSEC	FA14	8/38	9/26	10/56	11/35	14#39		
	SEKTRK	FA13	8/20	9/12	10/54	12/27	14#38		
	SELD1	F7DD	7/38	7/43	7#46				
	SELSK	F7C1	2/27	7#19					
	SELFDD	F61B	2#27	21/21					
	SET1K	F903	11/37	11#46					
	SETADR	FA91	15#51	16/55	16/57	17/34	17/60	18/ 9	18/14
			18/60						
	SETCUR	FA94	15/40	15#52	19/57				
	SETDMA	F801	2/30	4/36	8#42				
	SETSEC	F7FB	2/29	8#28					
	SETTRK	F7F5	2/28	8#10					
	SKEW0	F6DF	4# 5	11/51					
	SKEW1	F6CF	3#61	11/43					
n	SPT	F693	3#14						
	SPTOP	FFA0	1#19	4/13	20/39	20/54			
	SPTOP2	FFC0	1#18	20/52	20/61	21/31			
	STADH	000C	1#52	16/36					
	STADL	000D	1#53	16/41					
	TEST	0000	1# 5	4/ 8	4/15	4/56	5/ 6	7/15	7/21
			8/ 3	8/11	8/24	8/29	10/13	10/18	10/31
			10/36						
	UNACNT	FA18	10/51	10/59	11/ 4	11/27	14#45		
	UNADSK	FA19	10/53	11/ 6	14#46				
	UNASEC	FA1B	10/57	14#48					
	UNATRK	FA1A	10/55	11/17	11/19	14#47			
	VRAM	CC00	1#47	15/29	17/17	19/24			
	VRAME	CE00	1#48	15/27	17/15	19/31			
	VRAMON	C300	1#46	15/ 6	16/59	19/19			
	VSLOT	0003	1# 8	1/40	1/46	15/33	16/18	17/23	20/ 5
	VTOP	FC5C	16/12	16/31	16/34	19/51	19#61		
	WBOOT	F6E4	2/15	4#12					
	WBOOT2	005E	20/43	21#25					
	WBOOTE	F603	2#15	21/38					
	WHL	FBEB	19# 3	21/16					
	WHL1	FBE8	19# 2	19/ 6					
	WMOVE	F981	12#58	13/ 3					
n	WRALL	0000	1#55						
	WRDIR	0001	1#56	13/ 5					
	WRITE	F88D	2/32	10#35					
	WRITEH	F9B0	12/21	13/12	13#33				
	WRTYPE	FA20	10/26	10/46	13/ 4	14#54			
	WRUAL	0002	1#57	10/47					
	ZSLOT	0019	1# 7	4/20	4/28	9/50	9/59		

列表 4 Z80 用 (CP/M) 的監督程式

```

:APPLE Z80 BOARD CONTROL PROGRAM
:
= 003F      MSIZE      EQU      63      :RAMSIZE (Kbvt)
= FC00      RAM$TOP EQU      MSIZE * 1024
= F600      BIOS      EQU      RAM$TOP - 600H
= E800      BDOS      EQU      BIOS - 0E00H
= E000      CCP EQU      BDOS - 0800H
:
= E000      START     EQU      CCP      :0E000H
= 000C      ROM EQU      0CH
= 000D      RAM EQU      0DH
= 0019      ZSLOT     EQU      19H      :Z80 SLOT ADDRESS.....OCXH
:
0000 = E000      ORG      START
:
ENTRY:
E000      2A0100      LHLD      1      :WBOOT ADDRESS
E003      EB          XCHG
E004      210300      LXI      H.3      :STATUS
E007      19          DAD
E008      2298E1      SHLD     INTEST+1
E00B      210600      LXI      H.6
E00E      19          DAD
E00F      229DE1      SHLD     INCHR+1
E012      210900      LXI      H.9
E015      19          DAD
E016      22AAE1      SHLD     OUTCHR+1
E019      011900      LXI      B,ZSLOT
E01C      ED60        $      IN.C      H
E01E      2E00        MVI      L.0
E020      22EFE2      SHLD     ZMEMO
:
E023      3E0E        MONIT:    MVI      A.14      :CLEAR
E025      CDA2E1      CALL     OUTCA
E028      CDBBE1      CALL     CRLF
E02B      CD74E0      CALL     HELP
E02E      212EE0      REENT:    LXI      HL,REENT
E031      E5          PUSH     HL
E032      3E2F        LOOPM:    MVI      A.02FH
E034      CDA2E1      CALL     OUTCA
E037      CDC8E1      CALL     INHEX
E03A      FE0D        CMP      00DH
E03C      CA32E0      JZ       LOOPM
E03F      F5          PUSH     AF
E040      CDBBE1      CALL     CRLF
E043      F1          POP      AF
E044      FE20        CMP      ' '
E046      CA9DE2      JZ       DUMP
E049      FE4D        CMP      'M'
E04B      CAAAE2      JZ       MODIFY
E04E      FE47        CMP      'G'
E050      CAA7E2      JZ       JPHL
E053      FE52        CMP      'R'
E055      CA94E2      JZ       RAMON
E058      FE4F        CMP      'O'
E05A      CA8BE2      JZ       ROMON
E05D      FE4C        CMP      'L'
E05F      CAC7E2      JZ       LOAD
E062      FE48        CMP      'H'
E064      CA74E0      JZ       HELP
E067      FE49        CMP      'I'
E069      CA75E1      JZ       IOSET
E06C      FE58        CMP      'X'
E06E      CA30E1      JZ       IODUMP
:
E071      C332E0      JMP      LOOPM
:
E074      CDB5E1      HELP:    CALL     WMES
E077      302D392C41  DB      '0-9.A-F HEXDECIMAL'.0DH.0AH
E08B      20203A204D  DB      ' ' : MEMORY DUMP'.0DH.0AH
E09C      47203A2047  DB      'G : GOTO'.0DH.0AH
E0A6      48203A2048  DB      'H : HELP MENU DISPLAY'.0DH.0AH
E0BD      49203A2049  DB      'I : IO ACCESS'.0DH.0AH
E0CC      4C203A204C  DB      'L : LOAD FROM APPLE 4100H TO Z80 8100H'.0DH
E0F4      4D203A204D  DB      'M : MEMORY MODIFY'.0DH.0AH      .0AH
E107      4F203A2053  DB      'O : SET ROM'.0DH.0AH

```

```

E114 52203A2053 DB 'R : SET RAM'.ODH.OAH
E121 5B203A2049 DB 'X : IO DUMP'.ODH.OAH
E12E 00 DB 0
E12F C9 RET

;
E130 ;IODUMP:
E130 CD3CE1 CALL IODMP1
E133 CD9CE1 CALL INCHR
E136 FE0D CMP 00DH
E138 C230E1 JNZ IODUMP
E13B C9 RET

;
E13C E5 ;IODMP1
E13D CDEAE1 CALL PUSH HL
E140 CD06E2 IODMP2 CALL PR4HL
E143 CD06E2 IODMP3 CALL OUTSP
E146 44 MOV B,H
E147 4D MOV C,L
E148 ED78 $ IN.C A
E14A CDEFE1 CALL PR2HEX
E14D 23 INX HL
E14E 7D MOV A,L
E14F E60F AND 00FH
E151 CA5AE1 JZ IODMP4
E154 E603 AND 003H
E156 2BE8 ^E140$ JRZ IODMP2
E158 1BE9 ^E143$ JR IODMP3
E15A ;IODMP4
E15A CD06E2 CALL OUTSP
E15D CD06E2 CALL OUTSP
E160 C1 POP BC
E161 1E10 MVI E,16
E163 ED78 $ IODMP7 IN.C A
E165 FE20 CPI 20H
E167 3002 ^E16B$ JRNC IODMP5
E169 3E2E MVI A,'.'
E16B CDA2E1 IODMP5 CALL OUTCA
E16E 03 INX BC
E16F 1D DCR E
E170 20F1 ^E163$ JRNZ IODMP7
E172 C3BBE1 JMP CRLF

;
E175 ;IOSET:
E175 CD65E2 CALL IODATA
E178 4F MOV C,A
E179 7B MOV A,B
E17A C5 PUSH B
E17B A7 AND A
E17C 2B04 ^E182$ JRZ IOS1
E17E 44 MOV B,H
E17F 4D MOV C,L
E180 ED59 $ DT.C E
E182 CDBBE1 IOS1: CALL CRLF
E185 C1 POP B
E186 79 MOV A,C
E187 FE0D CMP 00DH
E189 C8 RZ
E18A FE2F CMP
E18C 2001 ^E18F$ JRNZ IOS2
E18E 2B HL
E18F FE20 IOS2: CMP
E191 2001 ^E194$ JRNZ IOS3
E193 23 INX HL
E194 C375E1 IOS3: JMP IOSET

;
E197 ;INTEST:
E197 CD06F6 CALL BIOS+6
E19A A7 AND A
E19B C9 RET

;
E19C ;INCHR:
E19C C309F6 JMP BIOS+9

;
E19F ;INECOH:
E1A2 CD9CE1 CALL INCHR
E1A3 C5 OUTCA: PUSH B
E1A4 4F MOV C,A
E1A7 CDA9E1 CALL OUTCHR
E1A8 C1 POP B
E1A8 C9 RET

;
E1A9 C30CF6 ;OUTCHR: JMP BIOS+12

```

E1AC	CDA2E1	WHL1:	CALL	OUTCA
E1AF	7E	WHL:	MOV	A, [HL]
E1B0	23		HL	
E1B1	A7		AND	A
E1B2	20F8 ^E1AC\$		JRNZ	WHL1
E1B4	C9		RET	
E1B5	E3	WMES:	XTHL	
E1B6	CDAFE1		CALL	WHL
E1B9	E3		XTHL	
E1BA	C9		RET	
E1BB	F5	CRLF:	PUSH	AF
E1BC	3E0D		MVI	A, 00DH
E1BE	CDA2E1		CALL	OUTCA
E1C1	3E0A		MVI	A, 0AH
E1C3	CDA2E1		CALL	OUTCA
E1C6	F1		POP	AF
E1C7	C9		RET	
E1C8	210000	INHEX:	LXI	HL, 00000H
E1CB	0600		MVI	B, 0
E1CD	CD9FE1	INHEX1:	CALL	INECOH
E1D0	FE30		CMP	'0'
E1D2	D8		RC	
E1D3	FE47		CMP	'G'
E1D5	D0		RNC	
E1D6	FE3A		CMP	'9'+1
E1D8	3805 ^E1DF\$		JRC	INHEX2
E1DA	FE41		CMP	'A'
E1DC	D8		RC	
E1DD	D607		SUB	007H
E1DF	D630	INHEX2:	SUB	'0'
E1E1	29		DAD	HL
E1E2	29		DAD	HL
E1E3	29		DAD	HL
E1E4	29		DAD	HL
E1E5	85		ADD	L
E1E6	6F		MOV	L, A
E1E7	04		INR	B
E1E8	18E3 ^E1CD\$		JR	INHEX1
E1EA	7C	PR4HL:	MOV	A, H
E1EB	CDEFE1		CALL	PR2HEX
E1EE	7D		MOV	A, L
E1EF	F5	PR2HEX:	PUSH	AF
E1F0	0F		RRC	
E1F1	0F		RRC	
E1F2	0F		RRC	
E1F3	0F		RRC	
E1F4	CDF8E1		CALL	PR1HEX
E1F7	F1		POP	AF
E1F8	E60F	PR1HEX:	AND	00FH
E1FA	FE0A		CMP	00AH
E1FC	3802 ^E200\$		JRC	PR1HX1
E1FE	C607		ADI	007H
E200	C630	PR1HX1:	ADI	030H
E202	CDA2E1		CALL	OUTCA
E205	C9		RET	
E206	F5	OUTSP:	PUSH	AF
E207	3E20		MVI	A, 020H
E209	CDA2E1		CALL	OUTCA
E20C	F1		POP	AF
E20D	C9		RET	
E20E	CD06E2	OUTSPB:	CALL	OUTSP
E211	10FB ^E20E\$		DJNZ	
E213	C9		RET	
E214	E5	DUMP1L:	PUSH	HL
E215	CDEAE1		CALL	PR4HL
E218	CD06E2	DMP1L1:	CALL	OUTSP
E21B	CD06E2	DMP1L2:	CALL	OUTSP
E21E	7E		MOV	A, [HL]
E21F	CDEFE1		CALL	PR2HEX
E222	23		INX	HL
E223	7D		MOV	A, L
E224	E60F		AND	00FH
E226	2806 ^E22E\$		JRZ	DMP1L3
E228	E603		AND	003H

E22A	28EC ^E218\$	JRZ	DMP1L1	
E22C	18ED ^E218\$	JR	DMP1L2	
E22E		DMP1L3		
E231	CD06E2	CALL	OUTSP	
E234	E3	CALL	OUTSP	
E235	0610	MVI	B,16	
E237	7E	DMP1L6	MOV	A,M
E238	FE20	CMP	20H	
E23A	3002 ^E23E\$	JRNC	DMP1L4	
E23C	3E2E	MVI	A,' '	
E23E	CDA2E1	DMP1L4	CALL	OUTCA
E241	23	INX	HL	
E242	10F3 ^E237\$	DJNZ	DMP1L6	
E244	E1	POP	HL	
E245	C3BBE1	JMP	CRLF	
E248	CDEAE1	PRIO:	CALL	PR4HL
E24B	CD06E2	CALL	OUTSP	
E24E	CD06E2	CALL	OUTSP	
E251	44	MOV	B,H	
E252	4D	MOV	C,L	
E253	ED78	\$ IN,C	A	
E255	C3EFE1	JMP	PR2HEX	
E258	CDEAE1	PRDATA:	CALL	PR4HL
E25B	CD06E2	CALL	OUTSP	
E25E	CD06E2	CALL	OUTSP	
E261	7E	MOV	A,[HL]	
E262	C3EFE1	JMP	PR2HEX	
E265	CD48E2	INDATA	CALL	PRIO
E268	1803 ^E26D\$	JR	IND1	
E26A	CD58E2	INDATA:	CALL	PRDATA
E26D	CD06E2	IND1:	CALL	OUTSP
E270	3E2D	MVI	A,' - '	
E272	CDA2E1	CALL	OUTCA	
E275	CD06E2	CALL	OUTSP	
E27B	0600	MVI	B,000H	
E27A	E5	PUSH	H	
E27B	CDC8E1	CALL	INHEX	
E27E	EB	XCHG		
E27F	E1	POP	H	
E280	FE0D	CMP	00DH	
E282	C8	RZ		
E283	FE20	CMP		
E285	C8	RZ		
E286	FE2F	CMP	' / '	
E288	20E0 ^E26A\$	JRNZ	INDATA	
E28A	C9	RET		
E28B	2AEFE2	RAMON:	LHLD	ZMEMO
E28E	44	MOV	B,H	
E28F	0E0C	MVI	C,ROM	
E291	ED79	\$ OT,C	A	
E293	C9	RET		
E294	2AEFE2	RAMON:	LHLD	ZMEMO
E297	44	MOV	B,H	
E298	0E0D	MVI	C,RAM	
E29A	ED79	\$ OT,C	A	
E29C	C9	RET		
E29D	CD14E2	DUMP:	CALL	DUMP1L
E2A0	CD9CE1	CALL	INCHR	
E2A3	FE0D	CMP	00DH	
E2A5	C29DE2	JNZ	DUMP	
E2A8	C9	RET		
E2A9	E9	JPHL:	PCHL	
E2AA	CD6AE2	MODIFY:	CALL	INDATA
E2AD	4F	MOV	C,A	
E2AE	78	MOV	A,B	
E2AF	A7	AND	A	
E2B0	2801 ^E2B3\$	JRZ	MOD1	
E2B2	73	MOV	[HL],E	
E2B3	CDBBE1	MOD1:	CALL	CRLF
E2B6	79	MOV	A,C	
E2B7	FE0D	CMP	00DH	

```

E2B9 C8 RZ
E2BA FE2F CMP
E2BC 2001 ^E2BF$ JRNZ
E2BE 2B DCX
E2BF FE20 MOD2:
E2C1 2001 ^E2C4$ JRNZ
E2C3 23 INX
E2C4 C3AAE2 MOD3:
;
E2C7 2100B1 ; LOAD: LXI H,8100H
E2CA 010041 LXI B,4100H
E2CD 110020 LXI D,2000H
E2D0 ;
E2D0 ED7B $ IN,C
E2D2 77 MOV A,M
E2D3 23 INX H
E2D4 03 INX B
E2D5 1D DCR E
E2D6 20F8 ^E2D0$ JRNZ
E2D8 15 DCR D
E2D9 20F5 ^E2D0$ JRNZ
E2DB C9 RET
;
E2DC 0600 TIME MVI B,0
E2DE CDE5E2 TIME1 CALL TEST
E2E1 04 INR B
E2E2 20FA ^E2DE$ JRNZ
E2E4 C9 RET
;
E2E5 3E20 TEST: MVI A,20H
E2E7 CD2EE0 TEST1 CALL REENT
E2EA 3C INR A
E2EB F2E7E2 JF TEST1
E2EE C9 RET
;
E2EF 00C4 ; ZMEMD: DW 0C400H
E2F1 ; END

```

no ERRORS. 69 Labels, 886h bytes not used. Program LWA = E2F1h.

D>XDIR

Directory drive - D

Act165	com	12K	Bios16	asm	18K	Ldcpm	sub	2K	Rw280	asm	12K
Act80	com	14K	Bios16	bak	18K	Lpt80	prn	30K	Rw280	bak	12K
Apload	asm	2K	Bios2	asm	16K	Monipl	asm	2K	Rw280	ban	12K
Apload	bak	2K	Bios2	hex	6K	Monipl	bak	2K	Rw280	hex	4K
Apload	hex	2K	Bios2	prn	54K	Monipl	hex	2K	Rw280	prn	40K

列表 5 Z80 ROM使用的 SUBMIT 程式

```

D>TYPE LDCPM.CSUB
XSUB
ZSID CPM63.COM
M980.2800.200
IBIOS2.HEX
R2200
M2200.22FF.100
M21C0.21FF.20C0
F2100.24FF.E5
F2500.43FF.FF
IAPLOAD.HEX
R

```



```

>PR#0
>LOAD FORMAT1K
>LIST
10 REM YD280 FORMAT PROGRAM
15 D$=""
20 REM 2 SIDES,DOUBLE DENSITY,80TR
  ACK/SIDE,5 SECTOR/TRACK, 1 K BYT
  E/SECTOR
23 PRINT D$;"BLOAD .FORMAT.COM,A$600
  0"
25 PRINT D$;"BLOAD .FORMAT.DTA,A$640
  0"

```

```

30 PRINT "INSERT DISKETT & PRESS C/
  R KEY THEN START....CNTL-C & C/R
  THEN STOP"
40 INPUT A$
50 PRINT "FORMAT START"
80 CALL 6*4096
90 GOTO 30

```

```

>BLOAD .FORMAT.COM,A$6000
>CALL-151

```

*

```

>PR#0
>CALL-151

```

*6000.61BF

```

6000- 4C 09 60 4C 88 60 4C E6
6008- 60 A9 FD 8D F4 C0 A9 00
6010- 8D F0 C0 A9 00 85 00 F0
6018- 05 A9 50 8D F0 C0 2C F8
6020- C0 50 FB A9 00 85 04 A9
6028- FD D0 06 A9 01 85 04 A9
6030- FC 8D F4 C0 A9 04 20 A8
6038- FC A9 00 85 02 A9 64 85
6040- 03 A0 00 AD F0 C0 30 FB
6048- A9 F0 8D F0 C0 D0 08 8D
6050- F3 C0 C8 D0 02 E6 03 B1
6058- 02 C9 50 30 08 F0 04 A5
6060- 04 10 02 A5 00 2C F8 C0
6068- 30 E5 50 F9 AD F0 C0 29
6070- E4 D0 12 A5 04 F0 84 E6
6078- 00 A5 00 C9 50 90 3A AD
6080- F0 C0 85 00 50 00 EA EA
6088- A9 0A 8D BC 61 20 50 61
6090- 20 9A 61 80 3E A9 82 A6
6098- 10 10 02 09 08 8D F0 C0
60A0- A5 12 85 02 A5 13 85 07
60A8- A0 00 F0 0A AD F3 C0 91
60B0- 02 C8 D0 02 E6 03 2C F8
60B8- C0 30 F1 50 F9 AD F0 C0
60C0- 29 1C D0 05 18 AD F0 C0
60C8- 50 AA 29 10 D0 0A CE BC
60D0- 61 D0 BA AD F0 C0 38 60
60D8- A5 10 29 80 20 78 61 DE
60E0- BC 61 D0 A9 38 60 A9 0A
60E8- 8D BC 61 20 50 61 AD F0
60F0- C0 29 40 D0 DE 20 9A 61
60F8- 80 D9 A9 A2 A6 10 10 02
6100- 09 08 8D F0 C0 A5 12 85
6108- 02 A5 13 85 03 A0 00 F0
6110- 0A B1 02 8D F3 C0 C8 D0
6118- 02 E6 03 2C F8 C0 30 F1
6120- 50 F9 AD F0 C0 29 7E D0
6128- 05 AD F0 C0 18 60 AA 29
6130- 40 D0 0A 8A 29 10 D0 0A
6138- CE BC 61 D0 AE AD F0 C0
6140- 38 30 A5 10 29 80 20 78
6148- 61 CE BC 61 D0 9D F0 ED
6150- AD F0 C0 4A 80 FA A5 10
6158- 29 7F CD F1 C0 F0 08 A5
6160- 10 20 78 61 18 90 E9 A5
6168- 11 8D F2 C0 A9 FD A6 10
6170- 10 02 A9 FC 8D F0 C0 60
6178- A8 10 04 A0 EC D0 02 A0
6180- ED 8C F4 C0 29 7F F0 0C

```

```

6188- 8D F3 C0 A9 18 8D F0 C0
6190- AD F0 C0 60 A9 08 8D F0
6198- C0 60 A0 00 8C BD 61 8C
61A0- BE 61 AD F0 C0 10 13 C8
61A8- D0 F8 EE BD 61 D0 F3 EE
61B0- BE 61 10 EE 20 3A FF 38
61B8- B0 E0 18 60 01 00 00 FF

```

*

```

>BLOAD .FORMAT.DTA,A$6400
>CALL-151

```

*6400.7DFF

```

6400- 4E 4E 4E 4E 4E 4E 4E 4E
6408- 4E 4E 4E 4E 4E 4E 4E 4E
6410- 4E 4E 4E 4E 4E 4E 4E 4E
6418- 4E 4E 4E 4E 4E 4E 4E 4E
6420- 4E 4E 4E 4E 4E 4E 4E 4E
6428- 4E 4E 4E 4E 4E 4E 4E 4E
6430- 4E 4E 4E 4E 4E 4E 4E 4E
6438- 4E 4E 4E 4E 4E 4E 4E 4E
6440- 4E 4E 4E 4E 4E 4E 4E 4E
6448- 4E 4E 4E 4E 4E 4E 4E 4E
6450- 00 00 00 00 00 00 00 00
6458- 00 00 00 00 F6 F6 F6 FC
6460- 4E 4E 4E 4E 4E 4E 4E 4E
6468- 4E 4E 4E 4E 4E 4E 4E 4E
6470- 4E 4E 4E 4E 4E 4E 4E 4E
6478- 4E 4E 4E 4E 4E 4E 4E 4E
6480- 4E 4E 4E 4E 4E 4E 4E 4E
6488- 4E 4E 4E 4E 4E 4E 4E 4E
6490- 4E 4E 00 00 00 00 00 00
6498- 00 00 00 00 00 00 F5 F5
64A0- F5 FE 50 60 01 03 F7 4E
64A8- 4E 4E 4E 4E 4E 4E 4E 4E
64B0- 4E 4E 4E 4E 4E 4E 4E 4E
64B8- 4E 4E 4E 4E 4E 00 00 00
64C0- 00 00 00 00 00 00 00 00
64C8- 00 F5 F5 F5 F5 F5 F5 F5
64D0- F5 F5 F5 F5 F5 F5 F5 F5
64D8- F5 F5 F5 F5 F5 F5 F5 F5
64E0- F5 F5 F5 F5 F5 F5 F5 F5
64E8- F5 F5 F5 F5 F5 F5 F5 F5
64F0- F5 F5 F5 F5 F5 F5 F5 F5
64F8- F5 F5 F5 F5 F5 F5 F5 F5
6500- F5 F5 F5 F5 F5 F5 F5 F5
6508- F5 F5 F5 F5 F5 F5 F5 F5
6510- F5 F5 F5 F5 F5 F5 F5 F5
6518- F5 F5 F5 F5 F5 F5 F5 F5
6520- F5 F5 F5 F5 F5 F5 F5 F5
6528- F5 F5 F5 F5 F5 F5 F5 F5
6530- F5 F5 F5 F5 F5 F5 F5 F5
6538- F5 F5 F5 F5 F5 F5 F5 F5
6540- F5 F5 F5 F5 F5 F5 F5 F5
6548- F5 F5 F5 F5 F5 F5 F5 F5
6550- F5 F5 F5 F5 F5 F5 F5 F5

```

67D8-
67E0-
67E8-
67F0-
67F8-
6800-
6808-
6810-
6818-
6820-
6828-
6830-
6838-
6840-
6848-
6850-
6858-
6860-
6868-
6870-
6878-
6880-
6888-
6890-
6898-
68A0-
68A8-
68B0-
68B8-
68C0-
68C8-
68D0-
68D8-
68E0-
68E8-
68F0-
68F8-
6900-
6908-
6910-
6918-
6920-
6928-
6930-
6938-
6940-
6948-
6950-
6958-
6960-
6968-
6970-
6978-
6980-
6988-
6990-
6998-
69A0-
69A8-
69B0-
69B8-
69C0-
69C8-
69D0-
69D8-
69E0-
69E8-
69F0-
69F8-
6A00-
6A08-
6A10-
6A18-
6A20-
6A28-
6A30-
6A38-
6A40-
6A48-
6A50-

6558-
6560-
6568-
6570-
6578-
6580-
6588-
6590-
6598-
65A0-
65A8-
65B0-
65B8-
65C0-
65C8-
65D0-
65D8-
65E0-
65E8-
65F0-
65F8-
6600-
6608-
6610-
6618-
6620-
6628-
6630-
6638-
6640-
6648-
6650-
6658-
6660-
6668-
6670-
6678-
6680-
6688-
6690-
6698-
66A0-
66A8-
66B0-
66B8-
66C0-
66C8-
66D0-
66D8-
66E0-
66E8-
66F0-
66F8-
6700-
6708-
6710-
6718-
6720-
6728-
6730-
6738-
6740-
6748-
6750-
6758-
6760-
6768-
6770-
6778-
6780-
6788-
6790-
6798-
67A0-
67A8-
67B0-
67B8-
67C0-
67C8-
67D0-

6F58-
6F60-
6F68-
6F70-
6F78-
6F80-
6F88-
6F90-
6F98-
6FA0-
6FAB-
6FB0-
6FB8-
6FC0-
6FC8-
6FD0-
6FDB-
6FE0-
6FE8-
6FF0-
6FF8-
7000-
7008-
7010-
7018-
7020-
7028-
7030-
7038-
7040-
7048-
7050-
7058-
7060-
7068-
7070-
7078-
7080-
7088-
7090-
7098-
70A0-
70AB-
70B0-
70B8-
70C0-
70CB-
70D0-
70DB-
70E0-
70EB-
70F0-
70FB-
7100-
7108-
7110-
7118-
7120-
7128-
7130-
7138-
7140-
7148-
7150-
7158-
7160-
7168-
7170-
7178-
7180-
7188-
7190-
7198-
71A0-
71AB-
71B0-
71BB-
71C0-
71CB-
71D0-

71DB-
71E0-
71EB-
71F0-
71FB-
7200-
7208-
7210-
7218-
7220-
7228-
7230-
7238-
7240-
7248-
7250-
7258-
7260-
7268-
7270-
7278-
7280-
7288-
7290-
7298-
72A0-
72AB-
72B0-
72BB-
72C0-
72CB-
72D0-
72DB-
72E0-
72EB-
72F0-
72FB-
7300-
7308-
7310-
7318-
7320-
7328-
7330-
7338-
7340-
7348-
7350-
7358-
7360-
7368-
7370-
7378-
7380-
7388-
7390-
7398-
73A0-
73AB-
73B0-
73BB-
73C0-
73CB-
73D0-
73DB-
73E0-
73EB-
73F0-
73FB-
7400-
7408-
7410-
7418-
7420-
7428-
7430-
7438-
7440-
7448-
7450-

7458-
7460-
7468-
7470-
7478-
7480-
7488-
7490-
7498-
74A0-
74A8-
74B0-
74B8-
74C0-
74C8-
74D0-
74D8-
74E0-
74E8-
74F0-
74F8-
7500-
7508-
7510-
7518-
7520-
7528-
7530-
7538-
7540-
7548-
7550-
7558-
7560-
7568-
7570-
7578-
7580-
7588-
7590-
7598-
75A0-
75A8-
75B0-
75B8-
75C0-
75C8-
75D0-
75D8-
75E0-
75E8-
75F0-
75F8-
7600-
7608-
7610-
7618-
7620-
7628-
7630-
7638-
7640-
7648-
7650-
7658-
7660-
7668-
7670-
7678-
7680-
7688-
7690-
7698-
76A0-
76A8-
76B0-
76B8-
76C0-
76C8-
76D0-

76D8-
76E0-
76E8-
76F0-
76F8-
7700-
7708-
7710-
7718-
7720-
7728-
7730-
7738-
7740-
7748-
7750-
7758-
7760-
7768-
7770-
7778-
7780-
7788-
7790-
7798-
77A0-
77A8-
77B0-
77B8-
77C0-
77C8-
77D0-
77D8-
77E0-
77E8-
77F0-
77F8-
7800-
7808-
7810-
7818-
7820-
7828-
7830-
7838-
7840-
7848-
7850-
7858-
7860-
7868-
7870-
7878-
7880-
7888-
7890-
7898-
78A0-
78A8-
78B0-
78B8-
78C0-
78C8-
78D0-
78D8-
78E0-
78E8-
78F0-
78F8-
7900-
7908-
7910-
7918-
7920-
7928-
7930-
7938-
7940-
7948-
7950-

周邊插槽 轉接器製作

APPLE II 微電腦之所以受到大家的歡迎，除了具有豐富的軟體供應之外，另外一個原因就是具有 8 個周邊插槽(peripheral slot)，可以作各種界面之擴充，這些周邊插槽之編號分別為 slot # 0~7。

雖然 APPLE 具有上述優點，但仍然有一點很不方便的地方，主要就是 8 個周邊插槽是安排在主機板上的，所以使用時主機的蓋子要隨時打開着，而且周邊界面板對周邊插槽的插進拔出，常常會造成主機板之插座鬆脫或對主機板造成震動而產生不良之影響。由上面所提到的這些問題來看，可以知道若要利用 APPLE 之周邊插槽來作一些測量或設計電路時，將會很不方便。所以本文將介紹一個 APPLE 周邊插槽轉接器(slot Repeater)以使你能夠將 APPLE 主機封蓋起來，仍能在 APPLE 主機外面來對 APPLE 作界面電路之連接或進行資料之存取。

爲了使周邊插槽轉接器電路板能正常工作，所以必須利用排線(cable)將 APPLE 之 Slot # 7 之信號連接至周邊轉接器電路板上。此處我們建議在 Slot # 7 之連接可利用 50 PIN 邊連接器(edge connector)，然後利用排線將信號連接至轉接器界面板上。

有一點要注意的是：Apple 匯流排(bus)上的信號，並不需要全部連接到轉接器界面電路板上，例如，在 Apple 匯流排上的電源線，並沒有被連接到轉接器界面電路上，這樣作的主要原因，是避免轉接器界面電路消耗太大的電流，所以在轉接器界面電路上另外保留有一些接腳，可用來連接外加電源。另外，轉接器上還需要適當的解碼電路，使得在轉接器界面電路上取得之位址信號，可以和 Apple 主機上的位址分配相符合。

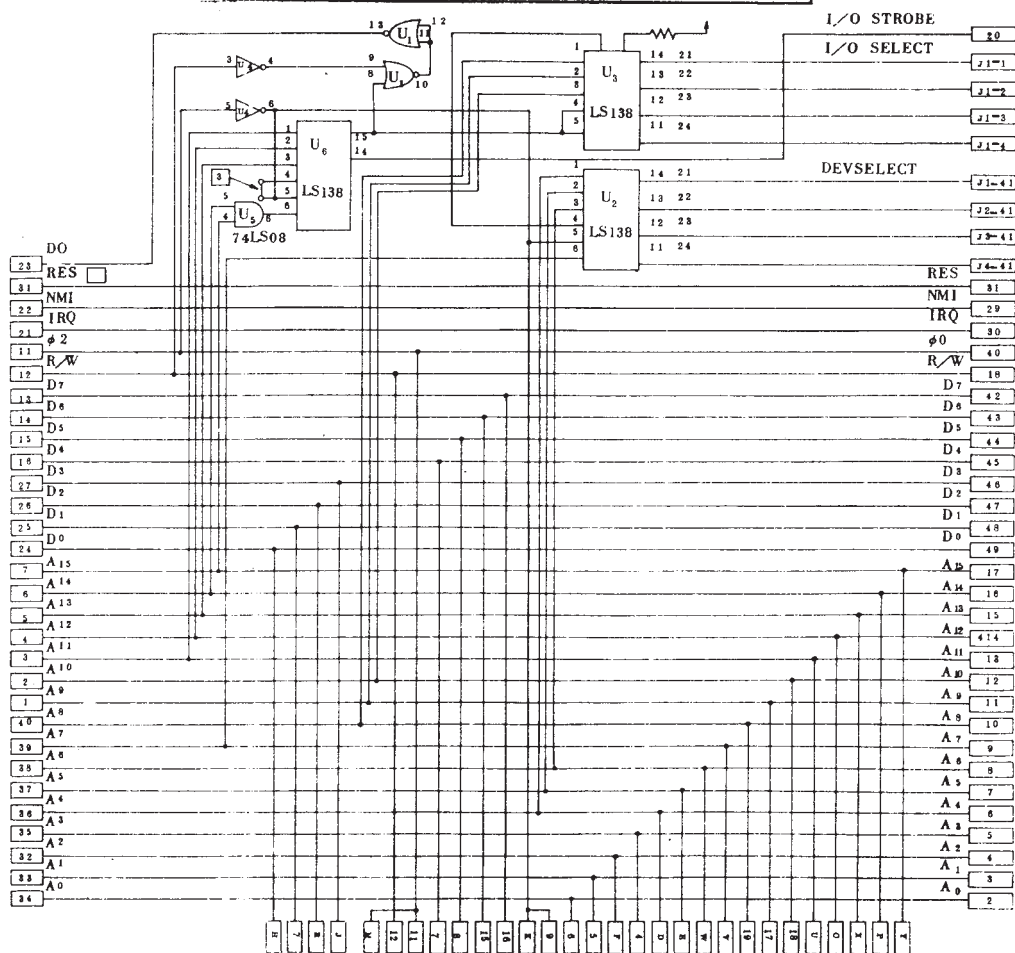
由圖 2 可以看出，位址信號 $A_{11} \sim A_{15}$ 是經由 74LS138 (3~8 線解碼器) 加以解碼，然

後產生輸入／出選擇 (I/O select) 信號，以及輸入／出激發 (I/O strobe) 信號。由圖上可以看出 74LS138(U6) 之 Z0 (ACTIVE LOW) 輸出信號，可以使第二個 74LS138(U3) 致能。74LS138(U3) 之主要作用就是將輸入端之位址信號 (A₈~A₁₀) 加以解碼，以產生 Slot # 1~4 的 I/O 選擇信號，例如 C100~C1FF 的位址信號經解碼後，即會選擇到第一個周邊連接器。同理由圖上可以看到 74LS138(U3) 之 Z0 輸出信號，可

以使第三個 74LS138(U2) 致能。這個 74LS138(U2) 可以將輸入端之位址信號 (A₄~A₇) 加以解碼，以產生裝置選擇 (device select) 信號，例如 C090~C09F 的位址信號經解碼後，即可產生 Slot # 1 之裝置選擇信號。

上面已說明了插槽轉接器 (Slot repeater) 界面板上之位址解碼情形，為了清楚起見，特別再將各位址範圍之解碼情形列表如下：

Slot	I/O SELECT	DEVICE SELECT	I/O-STROBE
2	C200 ~ C2FF	C0A0 ~ C0AF	C800 ~ CFFF
3	C300 ~ C3FF	C0B0 ~ C0BF	C800 ~ CFFF
4	C400 ~ C4FF	C0C0 ~ C0CF	C800 ~ CFFF
5	C500 ~ C5FF	C0D0 ~ C0DF	C800 ~ CFFF



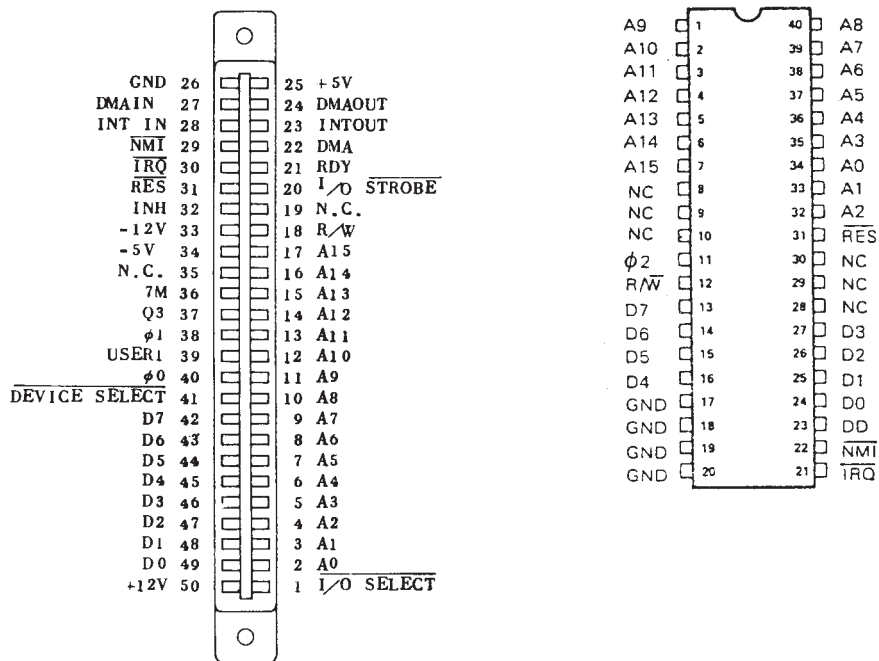


圖 1 插槽轉接器之連接器接腳情形

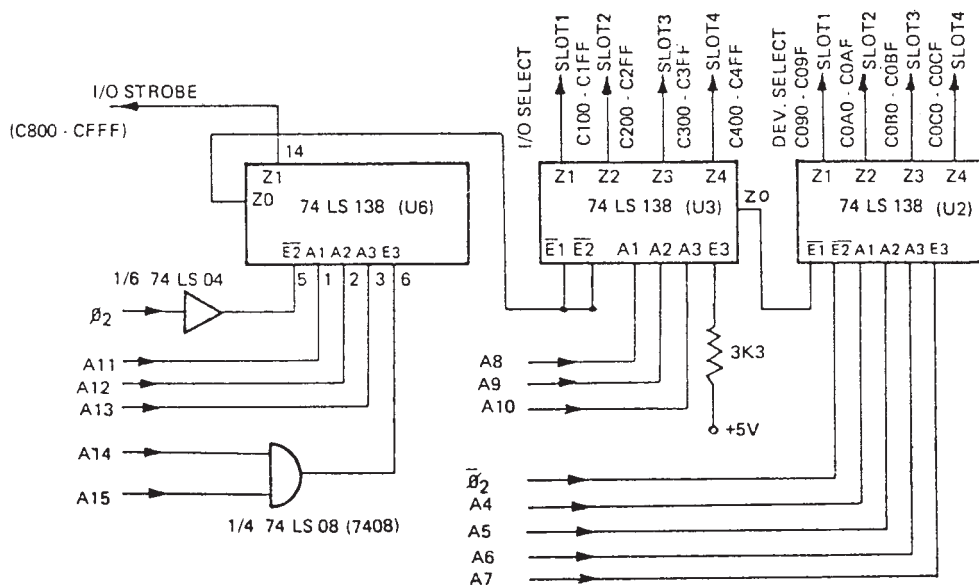


圖 4 是周邊插槽轉接器之整個電路接線情形。

電路組裝

首先將轉接器界面板上之各個 IC 插座及連接器組裝到電路板上。接著將電源接腳連接到電源供應器上去。在完成上述步驟後，接下來就是將電容 C 1，電阻 R 1 焊接到電路板上，然後將排線連接的 50 PIN（腳）連接器插接到 Apple 主機板的 Slot # 7。最後一個步驟就是裝上 IC 且應注意 IC 之方向不要裝反了，以免燒壞。插槽轉接器界面板上之零件位置安排情形如圖 5 所示。 ◆

圖 8 零件表

數量	名稱
3	14PIN 雙排腳插座
3	16PIN 雙排腳插座
1	40PIN 雙排腳插座
1	10 μ F/35V 鉭質電容
3	SN74LS138
1	SN74LS08
1	SN74LS02
1	SN74LS04
4	50PIN 邊連接器
1	3K/0.25W 電阻

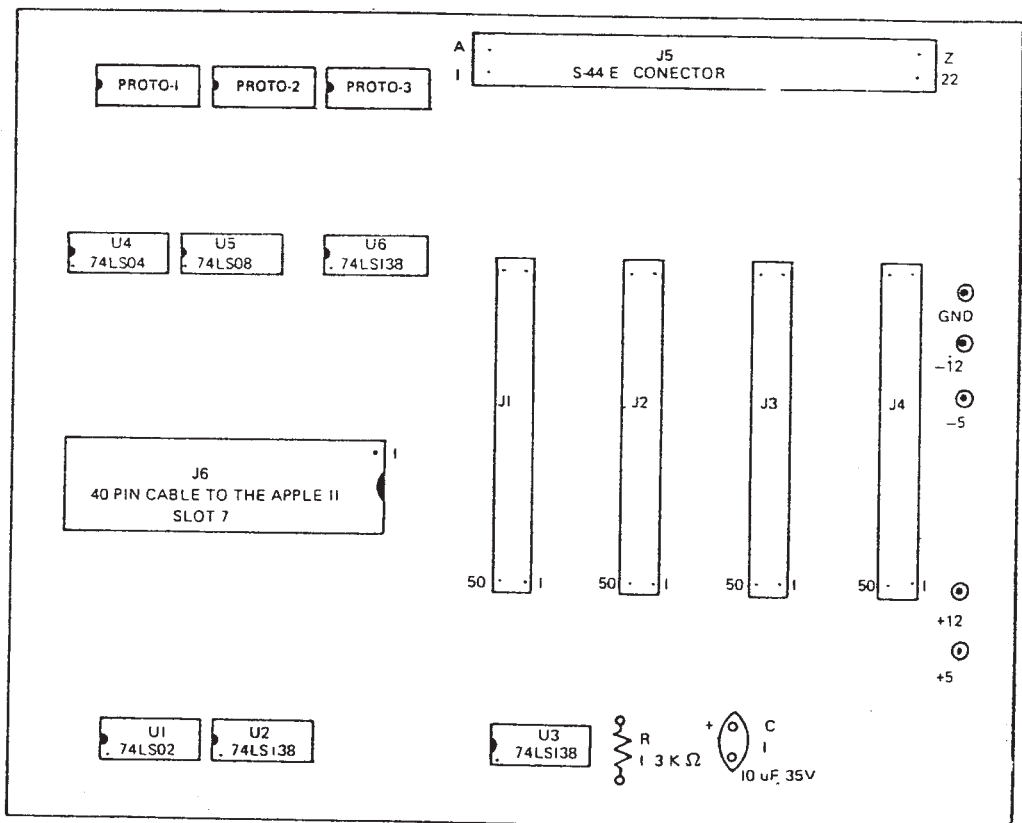


圖 5 零件佈置圖

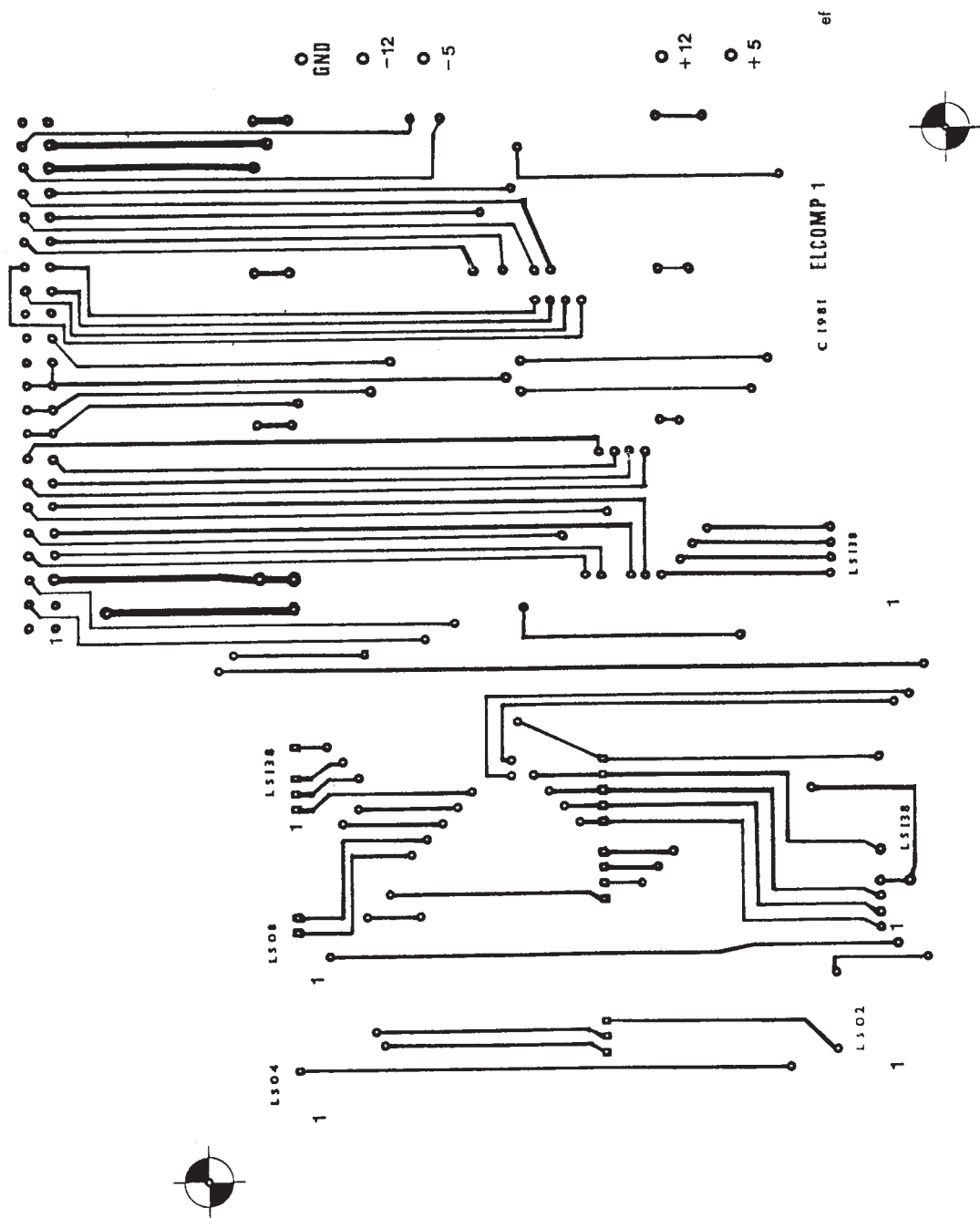


圖 6a 插槽轉接器界面印刷電路板之零件面(正面)電路圖

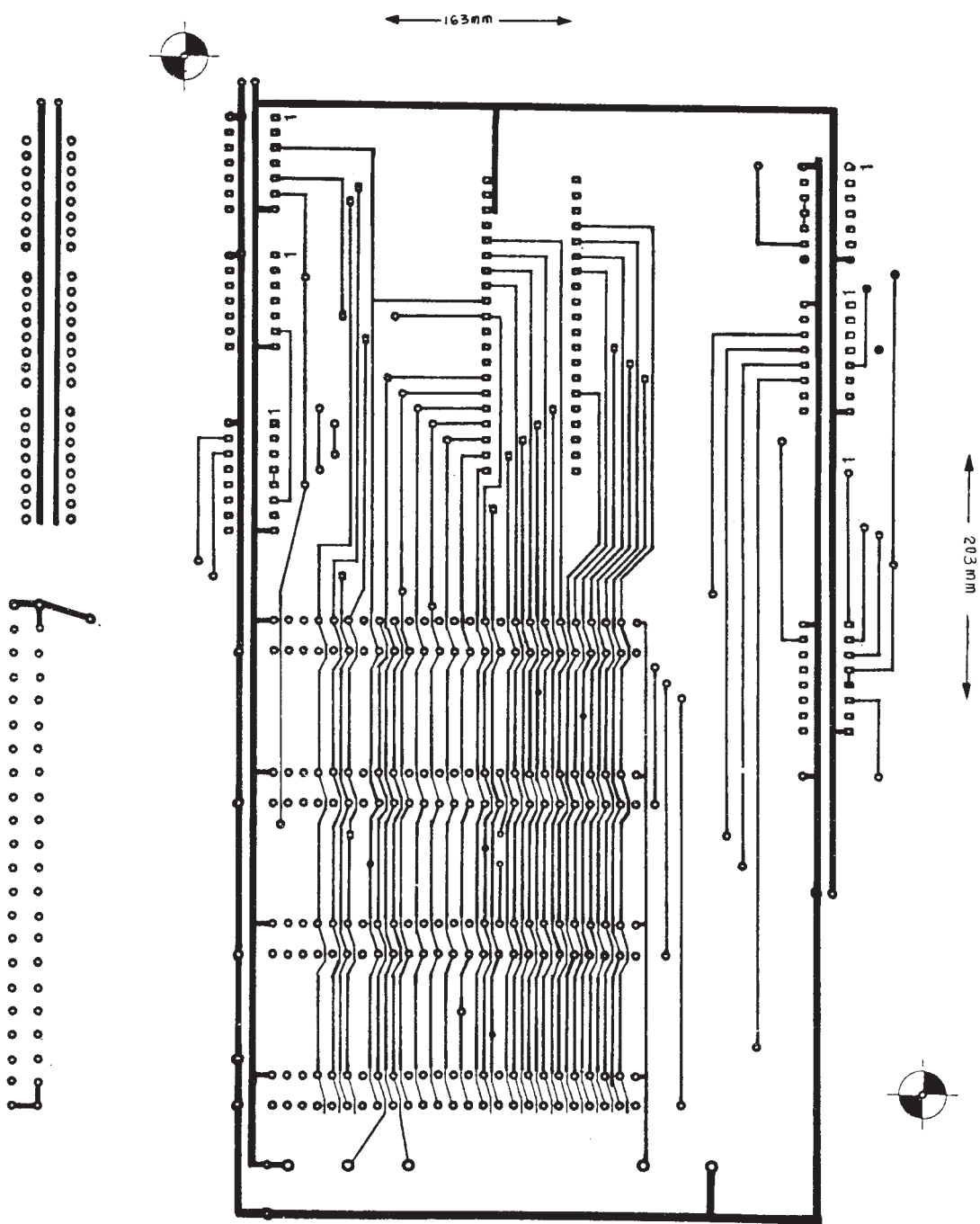


圖 6b 界面板底面（反面）之電路圖

6522多用途

I/O界面板製作

APPLE 用來當做控制電腦也很方便，它有很多的周邊界面，這次我們來研究 65 系列的一顆 I/O 專用的 IC-6522 VIA，6522 是一顆握手式 (hand shake) 雙埠 (共 16 Bit) 的並聯輸出入 IC，它和 8255 及 6821 等功能相類似。

一般周邊裝置與電腦間用 6522 來介接，可做資料的並聯輸出入控制，或單用某幾個 Bit 來做控制，可控制各種電機、儀表、家電……等。

下面將來討論一個利用 6522 的多用途 I/O 界面卡原理與製作，以及應用的問題。

實際問題的考慮

要製作 I/O 界面板之前，首先會遭遇到第一個問題就是 6522 I/O 晶片在速度上要較 6502 CPU 慢。因此，為了使 6502

CPU 和 6522 I/O 晶片在速度上能互相同步，必須加入一個延時程式。6522 I/O 界面板本身含有 1K RAM 的記憶容量，但在同一時間內僅有 $\frac{1}{4}$ K 的容量可被用到，雖然僅有 256 個位元組 (bytes) 的記憶容量，可是對於一個小的機械語言監督程式而言，這個容量是足夠的，且將機械程式存在這個範圍，可避免被主機上 BASIC 程式蓋掉的情形。在 I/O 板上的 1K RAM 是由兩個 $1K \times 4$ 的 2114 static (靜態) RAM 所組成，它們主要是用來儲存你的機械語言程式，但是在同一時間內只有 $\frac{1}{4}$ K Bytes 的 RAM 會被用到。至於是那 $\frac{1}{4}$ K 的 RAM 被用到，則是以 I/O 界面板所在的周邊連接器之插槽編號 (Slot #) 來決定。例如，假設你是要將一個小的機械語言常式存放在 I/O 界面板的 RAM 中，而 I/O 界面板是插於周邊連接器的 Slot #4，則你可

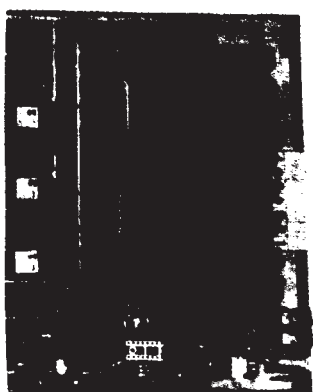


圖 1 APPLE 周邊連接器之插槽 (Slot)

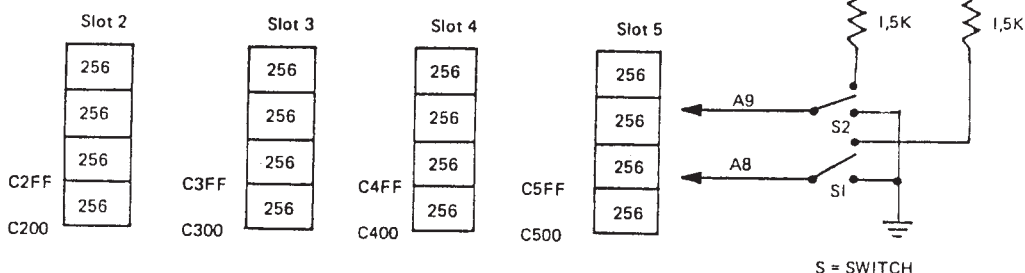


圖 2 I/O 界面板上 1K RAM 在每個插槽上的位址分配情形以及其對應的開關選擇情形

可利用開關的設定，來選用在這 1K RAM 中四個不同的機械語言常式（每個機械語言常式最多為 256 bytes）。因此利用開關的設定，即可以在同一記憶區選擇四個不同的常式。

圖 3 所示是 6522 I/O 界面板的方塊圖，在圖形上 I/O 界面板左方空白的區域，即是讓你安置自己電路的地方。這個 6522 I/O 界面板之主要優點是，它可以利用各種不同的程式語言來規劃及控制，而這些程式可能是機械語言，也可能是由高階語言的 PEEK 和 POKE 命令寫成，至於詳細的應用將在後面說明。

在圖 3 中所示之 6522 VIA 晶片，主要

將機械語言常式寫入由 \$C400 開始之 RAM 記憶位址。並且因為在 I/O 界面板上有兩個開關 (S1, S2)，所以可用來選擇 1K RAM 中的任何一個 1/4 K 範圍。而這 1/4 K 範圍主要還是由 I/O 界面板所在的插槽編號來決定。例如，假設 I/O 界面板是插在 Slot #5，則此時所分配之 RAM 記憶範圍是 \$C500 - \$C5FF。

假如你的應用需要四個不同的機械語言常式。則你可以將這些常式寫入位址 \$C500 - \$C5FF (設 I/O 界面板插於 Slot #5)，然後利用 I/O 界面板上的兩個開關 (S1, S2) 來選擇四個不同的記憶方塊。如此一來，你即

	S1	S2
1. 1/4 k	off	off
2. 1/4 k	on	off
3. 1/4 k	off	on
4. 1/4 k	on	on

含有兩個埠 PORT A 和 PORT B，以及 8 位元的雙向資料匯流排，此外它也含有兩個計時器，1 個 8 位元移位暫存器，和 4 條握手連絡 (hand-shaking) 線。這些握手連絡線主要是用以讓 6502 能和外界裝置互相連絡。而這些裝置通常都具有感知備妥 (READY) 或未備妥 (NOT READY) 狀態之能力。

在圖 5 所示是 6522 I/O 界面板及 RAM 插在各周邊插槽所對應之記憶位址。因為

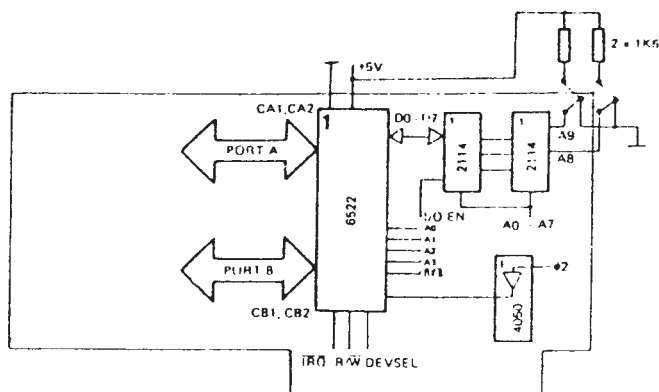


圖 3 6522 I/O 界面板之方塊圖



圖 4 6522 I/O 界面板之實際圖

board in slot	6522			
	Hex		Decimal	
2	from C0A0	to C0AF	from -16224	to -16209
3	C0B0	C0BF	-16208	-16193
4	C0C0	C0CF	-16192	-16177
5	C0D0	C0DF	-16176	-16161
board in slot	RAM			
	Hex		Decimal	
2	C200	C2FF	-15862	-15617
3	C300	C3FF	-15616	-15361
4	C400	C4FF	-15360	-15105
5	C500	C5FF	-15104	-14849

圖 5 記憶位址分配表

6522 是記憶對映 (memory mapped) 的方式，所以圖 5 中所示之位址，即是你可以用來傳輸和控制 6522 I/O 界面板的實際記憶位址。

6522 I/O 界面板之程式規劃

6522 中的埠 (PORT) A 和 B 可以作為輸入／輸出埠，至於究竟要作為輸入或輸出，則是由程式存入資料方向暫存器 DDRA 和 DDRB 的內含值來決定的。例如，假設 DDRA 或 DDRB 中的某個位元被設定為 1，則埠 A 或埠 B 中相對於此位元的資料線將被設定為輸入。反之，假如 DDRA 或 DDRB 中的某個位元被設定為 0

，則埠 A 或埠 B 中相對於此位元的資料線將被設定為輸出。此外，若將 255 (\$FF) 載入

DDRA 中，則表示所有連接埠 A 的資料線都被設定為輸出。要將資料載入埠 A 或埠 B，可利用機械語言先將欲載入之值，載入累積器 A 中，然後再存入埠 A 或埠 B 所對應之記憶位址，即可完成上述埠 A 或埠 B 之載入工作。假若是在 BASIC 程式中，則可利用 POKE 指令直接將資料值存入埠 A 或埠 B 所對應之記憶位址。

一旦資料方向暫存器 (DDRA 或 DDRB) 中的位元值被設定後，則它們將一直維持在所設定之狀態，直到電腦被重定為止，而重定動作可由按鍵重定 (RESET)，重新開機或將新的數值載入資料方向暫存器來完成。在電腦發生重定 (RESET) 或開機動作 (POWER UP) 時，則所有埠資料線都會被設定為 0，也就是說此時所有的埠資料線都是用作輸入。且這些埠資料線一直維持在輸入狀態，除非利用程式來改變其狀態。

利用 6522 作一個顯示指示器

上面已將 6522 埠 A 和埠 B 的控制作了介紹。為了能對 6522 I/O 界面板能有更進一步的認識。下面將利用 6522 I/O 界面板來作一個顯示指示器的應用。此應用主要是利用程式之規劃來控制 6522 VIA 晶片，使 8 個 LED 產生各種顯示變化。在此應用中，每個

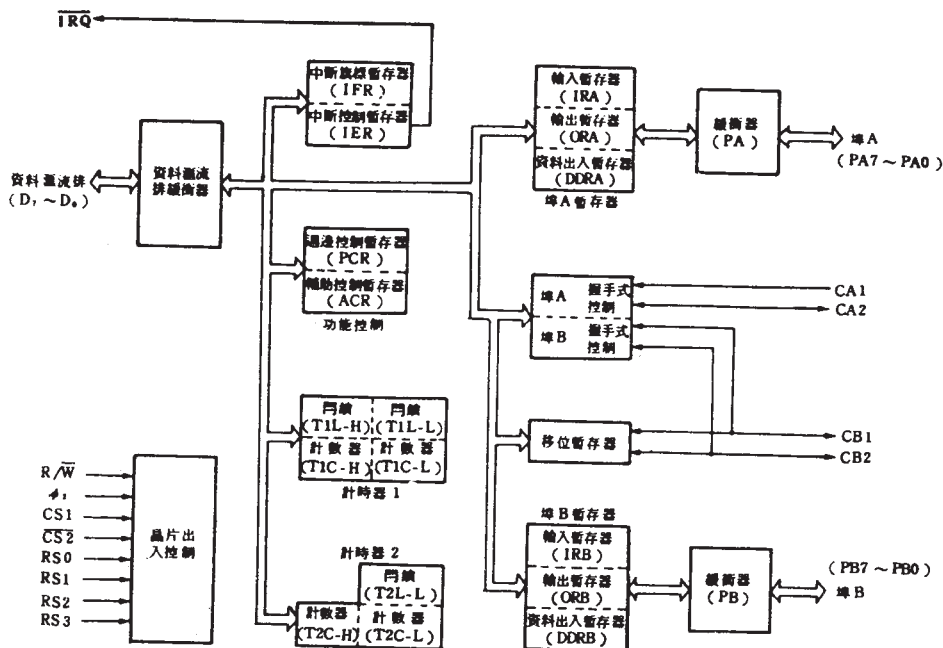


圖 6 6522 內部方塊圖

Register Design.	Description		SLOT 2		SLOT 3		SLOT 4		SLOT 6	
	Write	Read	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
ORB/IRB	Output Register "B"	Input Register "B"	C0A0	-16224	C0B0	-16208	C0C0	-16192	C0D0	-16176
ORA/IRA	Output Register "A"	Input Register "A"	C0A1	-16223	C0B1	-16207	C0C1	-16191	C0D1	-16175
DDRB	Data Direction Register "B"		C0A2	-16222	C0B2	-16206	C0C2	-16190	C0D2	-16174
DDRA	Data Direction Register "A"		C0A3	-16221	C0B3	-16205	C0C3	-16189	C0D3	-16173
TIC-L	T1 Low-Order Latches	T1 Low-Order Counter	C0A4	-16220	C0B4	-16204	C0C4	-16188	C0D4	-16172
TIC-H	T1 High-Order Counter		C0A5	-16219	C0B5	-16203	C0C5	-16187	C0D5	-16171
TIL-L	T1 Low-Order Latches		C0A6	-16218	C0B6	-16202	C0C6	-16186	C0D6	-16170
TIL-H	T1 High-Order Latches		C0A7	-16217	C0B7	-16201	C0C7	-16185	C0D7	-16169
T2C-L	T2 Low-Order Latches	T2 Low-Order Counter	C0A8	-16216	C0B8	-16200	C0C8	-16184	C0D8	-16168
T2C-H	T2 High-Order Counter		C0A9	-16215	C0B9	-16199	C0C9	-16183	C0D9	-16167
SR	Shift Register		C0AA	-16214	C0BA	-16198	C0CA	-16182	C0DA	-16166
ACR	Auxiliary Control Register		C0AB	-16213	C0BB	-16197	C0CB	-16181	C0DB	-16165
PCR	Peripheral Control Register		C0AC	-16212	C0BC	-16196	C0CC	-16180	C0DC	-16164
IFR	Interrupt Flag Register		C0AD	-16211	C0BD	-16195	C0CD	-16179	C0DD	-16163
IER	Interrupt Enable Register		C0AE	-16210	C0BE	-16194	C0CE	-16178	C0DE	-16162
ORA/IRA	Same as RegA Except No "Handshake"		C0AF	-16209	C0BF	-16193	C0CF	-16177	C0DF	-16161

圖 7 6522 內部暫存器在各個周邊連接器上所對應之記憶位址

LED 都應加上限流電阻。

如圖 8 所示，將各 LED 之陽極連接至 6522 的雙向資料匯流排去，且各 LED 之陰極則經由 220Ω 的電阻接地。

程式說明

圖 9 所示之示範程式，是假設 6522 I/O 界面板是在周邊連接器上之 Slot #4。在行 30 是設定資料方向暫存器 (DDRA) 的記憶位址，以及埠 A (TA) 之記憶位址。行 40 的 POKE 指述則是將埠 A 的所有資料線設定為輸

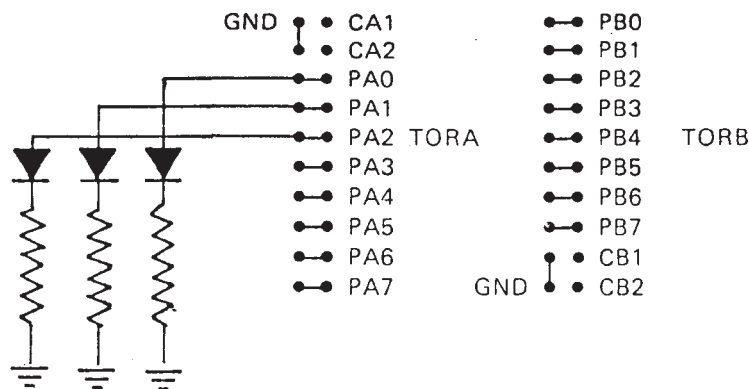


圖 8 顯示如何將 LED 連接至埠資料線的情形

圖 9 用以產生棒狀圖形之 BASIC 程式

```

10 REM BARGRAPH 1
20 REM BOAARD IN SLOT 4
30 DDRA = - 16189:TA = - 16191
40 POKE DDRA,255
50 A = 1
60 POKE TA,A
70 GOSUB 200
80 A = A * 2
90 IF A = 256 THEN A = 1
100 GOTO 60
200 REM TIME DELAY
210 FOR I = 1 TO 50
220 NEXT I: RETURN

```

出狀態。行 50 先設定 A 值為 1。行 60 將 A 值存入埠 A。行 70 是呼叫行 200 之時間延遲程式，如此一來才能看到 LED 的變化情形。

在上述程式迴路中，迴路每執行一次則變數 A 即向左移動一個位元，以便熄滅前一個點亮的 LED 且點亮下一個 LED……依此類推。利用上述方法即可將連接至埠 A 的 8 個 LED 依照順序點亮且每次僅會點亮一個。而行 90 則是重新設定變數 A 之值，以重新執行另一週期的顯示。

另一個棒狀圖形示範

下面將要介紹這個程式，主要是教你如何

利用 6522 I/O 界面板來產生真正的棒狀圖形顯示。也就是說在 8 個 LED 中的某一較高次位元之 LED 被點亮時，則其它較低次位元之 LED 也會被同時點亮。

程式說明

在圖 10 之程式中，行 10~行 40 和圖 9 一樣，而其它行號說明如下，在行 50 變數 A 和 B 均被設定為 1。在行 60 變數 B 被輸出至埠 A。而行 70 的 GOSUB 200 則是跳到一個延時副常式去。行 80 將 A 值乘以 2 主要是將 LED 之顯示指向下一個 LED。而 $B = B + A$ 之主要作用即是當連接到較高次位元之 LEI 被點亮時，則較低次位元之 LED 亦被同時點

圖 10 用以產生真正棒狀圖形之 BASIC 程

```

10 REM BARGRAPH 2
20 REM BOARD IN SLOT 4
30 DDRA = - 16189:TA = - 16191
40 POKE DDRA,255
50 B = 1:A = 1
60 POKE TA,B
70 GOSUB 200
80 A = A * 2:B = B + A
90 IF A = 256 THEN 50
100 GOTO 60
200 REM TIME DELAY
210 FOR I = 1 TO 50
220 NEXT I: RETURN

```

亮，如此即可造成真正的棒狀顯示。行 90 是當所有 LED 均被點亮時，再重新啓始另一個新的顯示週期。

6522 內部計時器之規劃

6522 內部計時器含有兩個 8 位元門鎖器，以及 16 位元計數器。

兩個 8 位元門鎖器是以 TIL/L 及 TIL/H 表示。至於 16 位元計時器也被分成兩個 8 位元之部分，且以 TIC/L 及 TIC/H 表示。對於較低次的計數器 TIC/L 而言，當你利用程式對它進行讀出或寫入時，將會得到不同的功能。例如，若對 TIC/L 進行寫入之動作，就好像是寫入 TIL/L 一樣，這動作如同對記憶位址執行寫入一樣。另外，若對 TIC/L 進行讀出動作，則將可以取得計時器之低位元組。而且若對 TIC/H 進行寫入動作，則可以啓動計數器之計數。在此操作期間 TIL/L 的內含值將被傳送至 TIC/L。每當 PB6 (PORT B 之 b6) 收到一個脈衝時，計數器 TIC/L 之內含值即被減 1，並且每當計數器之內含值減 1 時，6522 會自動檢查計數值是否已倒數至零。

在上面所討論之計時型態中，假如計數器之值已倒數至零，則可能會有兩種情形發生；主要是視計數器早先設定的作業型態來決定。即此時可能會有中斷產生或埠 B 的位元會被設定。而且在計數器倒數至零時，TIL/L 和 TIL/H 之內含值將被送至計數器去，以重新啓始另一計數週期。

上面所提到的作業型態，是由輔助控制暫存器 (ACR) 的 b6 和 b7 之設定狀態來決定

。下面表格即是顯示 ACR 之 b6 和 b7 所設定的作業型態的情形。

計時器之作業型態

如圖 11 所示，當輔助控制暫存器 (ACR) 之 b6 和 b7 均被設定為 1 時，則計時器將處於連續計數的作業型態，且每當計時暫存器的低 8 位元值為零時，則埠 B 的 b7 之狀態將會反相。如此一來，埠 B 的 b7 即可當作一個方波產生器。而所產生之方波週期，則是由輸入計時器之值來決定。例如，假設有一個 2 被存入計時器中，則將會產生一個 $2\mu S$ 的正向脈衝，其後跟隨著一個 $2\mu S$ 的負向脈衝，也就是說會產生一個全周期為 $4\mu S$ 的方波。而此方波的周期恰為存入計時器中之值的兩倍。爲了解這項應用，下面將列舉一個程式，它可以利用 6522 I/O 界面板來產生方波，而由這個程式所產生之方波周期為 100ms。

方波產生程式

在圖 12 程式中的行 12~15 是利用假指令 (pseudo-instruction) 來指定在程式中所用到之各標題 (label) 所對應之位址值。在行 18 是利用 LDA #C0 來設定作業型態 (ACR7 ACR6=11)。行 20~22 是將示範程式所用之值載入計時器中。由程式中可以知道計時器將被載入 \$C47F (50,303) 之值。而行 23 即啓始計時器之動作，有一點要附帶一提的是，此處之所以不將 50,000 存入計時器中，而却將 50,303 之值存入計時器中。主要原因就是 APPLE II 微電腦的時鐘脈波頻率並不正好等於 1MHz。實際上 Apple

ACR7	ACR6	Mode
0	0	Oneshot, only Interrupt, no Signal at PB7
0	1	Running Interrupts, no Signal at PB7
1	0	Oneshot, Interrupt, negative Pulse at PB7
1	1	Free running, square wave at PB7

圖 11 計時器的作業型態

圖 12 方波產生程式

```

0800          1          DCM "PR#1"
0800          2          ;
0800          3          ;
0800          4          ;*****
0800          5          ;*
0800          6          ;* SQUAREWAVE GENERATOR WITH *
0800          7          ;* A PERIOD OF 100.0 MS *
0800          8          ;*
0800          9          ;*****
0800         10          ;
0800         11          ;
0800         12          ACR      EQU $C0CB
0800         13          T1CL     EQU $C0C4
0800         14          T1CH     EQU $C0C5
0800         15          MONITO   EQU $FF59
0800         16          ;
0800         17          ;
0800 A9C0      18          LDA     #$C0          ;SET OPERATION MODE
0802 8DCBC0    19          STA     ACR
0805 A94E      20          LDA     #$4E          ;LOAD LO BYTE
0807 8DC4C0    21          STA     T1CL
080A A9C4      22          LDA     #$C4          ;LOAD HI BYTE
080C 8DC5C0    23          STA     T1CH          ;AND START TIMER
080F 4C59FF    24          JMP     MONITO
0812          25          ;
0812          26          ;

```

圖 13 單穩方波產生程式

```

0800          1          DCM "PR#1"
0800          2          ;
0800          3          ;
0800          4          ;*****
0800          5          ;*
0800          6          ;* MONOFLOP/ONESHOT *
0800          7          ;*
0800          8          ;*****
0800          9          ;
0800         10          ;
0800         11          ACR      EQU $C0CB
0800         12          T1CL     EQU $C0C4
0800         13          T1CH     EQU $C0C5
0800         14          IFR      EQU $C0CD
0800         15          ;
0800 A980      16          MONOFL  LDA     #$80          ; SET OPERATIONMODE
0802 8DCBC0    17          STA     ACR
0805 A94E      18          LDA     #$4E          ; LOAD LO BYTE
0807 8DC4C0    19          STA     T1CL
080A A9C4      20          LDA     #$C4          ; LOAD HI BYTE
080C 8DC5C0    21          STA     T1CH          ; START TIMER IFR6 SET TO 1
080F ADCDC0    22          M      LDA     IFR
0812 2940      23          AND     #$40
0814 F0F9      24          BEQ     M
0816 60        25          RTS
0817          26          ;
0817          27          ;
28          END

```

的速度要比廣告上所宣稱的還要快一些。

正如前面所提到的，當 6522 中的計時器被啟動後，它將會一直執行計數動作（不需 CPU 之幫忙），且在這個期間內無論有什麼輸入，均不會影響計數器之動作。而計數之動作僅有在電腦被中斷 (RESET)，開機 (POWER UP) 或改變暫存器之內含值時，才會改變。也就是說在上面三種情形都會終止計時器之連續計數 (free running) 作業型態。假如你希望改變程式操作時之計數頻率，則只要將新的數值載入門鎖器 TIL/L 和 TIL/H 中即可。在你將新值載入暫存器後，則原方波之產生動作將會停止，且當計時器倒數至零後，則新載入之值將會被接受，而產生新頻率的方波。

6522 計時器的另一個應用例

在這項應用中，我們將利用 6522 計時器來作為單擊或單穩的方波產生器。為了要完成這項工作，必須將 ACR 之值由 \$C0 更改成 \$80，以改變作業型態。用以產生單擊或單穩方波的程式如圖 13 所示。

利用計時器作為計數器

6522 VIA 晶片中的計時器，可以用來計數在埠 B 的 b6 (第 7 位元) 所出現的負向脈衝的個數。而輔助控制暫存器 (ACR) 的 b5 (

第 6 位元)，則可以決定計時器是作為單擊的方波產生器或連續的脈衝計數器。當 ACR 的 b5 被設定為 1 時，則計時器是作為連續的脈衝計數器，若 ACR 的 b5 被設定為 0 時，則計時器是作為單擊的脈波產生器。為了清楚起見，下面將舉一個程式來說明。

在下面將列舉的示範程式，是利用 6522 中的 1 個計時器來產生脈衝，而由另一個計時器來加以計數。在這項應用中是將埠 B 的腳 6 接至腳 7，且 T1 計時器當作一個連續的脈衝產生器，T2 當作計數器，如此一來即可組成一個理想的計時器，以測量各種不同程式或常式的執行時間。

現在就來說明一下，這個將計時器作為碼錶的示範程式，此示範程式主要是由兩部分組成：1 個簡短的 BASIC 程式以及機械語言常式。機械語言常式主要是用以設定計時器之作業型態，而且啟動計時器之動作（起始位址為 \$C04C）。另外由位址 \$C426 開始的機械語言常式，主要是將程式之執行時間（以 1/100 sec 為單位），存入位址 \$C4FE 和 \$C4FF 中。而 BASIC 程式之主要功能，即是在對這些位址進行資料的讀取，以計算出某一程式之執行時間。此處所提到的機械語言常式，是存於 6522 I/O 界面上的 RAM 中（假設界面板插於 Slot #4）。因此，可避免機

圖 14 BASIC “執行時間”計時器

作為計時器功能的機械語言副常式

```
1  REM  RUNTIME TEST
10  START = - 15348:FIN = - 15322:LO = - 15106
15  .I = - 15105
20  DS = CHR$(4)
25  PRINT DS;"BLOAD ETIME"
30  CALL START
40  GOSUB 1000
50  CALL FIN:GOSUB 990:END
990 PRINT "EXECUTION TIME=";
992 H% = PEEK (HI):L% = PEEK (LO)
994 PRINT (H% * 256 + L%) / 100;" SECONDS"
999 RETURN
1000 REM  PROGRAM UNDER TEST
1010 Q = 2.5:B = 1.2:C = 3.4
1020 E = 1 / Q
1030 FOR I = 1 TO 100
1040 A = (B + C) * Q
1050 NEXT I
1060 RETURN
```

CALL-151

*C400LL

```

C400- 20 0C C4 JSR $C40C
C403- 4C 59 FF JMP $FF59
C406- 20 26 C4 JSR $C426 ; RESET (WARM START)
C409- 4C 59 FF JMP $FF59
C40C- A9 E0 LDA #E0
C40E- 8D CB C0 STA $C0CB ; 設定 ACR7. ACR6 ACR5 ACR4=1110
C411- A9 01 LDA #01
C413- 8D C8 C0 STA $C0C8 ; 設定 T2C-L
C416- A9 00 LDA #00 ACR5=1 選取脈
C418- 8D C9 C0 STA $C0C9 ; 設定 T2C-H 且開始計數 衝計數作業型態
C41B- A9 EC LDA #EC 9284
C41D- 8D C4 C0 STA $C0C4 ; 設定 T1C-L
C420- A9 13 LDA #13 ACR7 ACR6=11
C422- 8D C5 C0 STA $C0C5 ; 設定 T1C-H 且開始計數 9288
C425- 60 RTS
C426- AD C8 C0 LDA $C0C8
C429- 8D FE C4 STA $C4FE ; 取入 1/100 sec 為單位之計數值
C42C- AD C9 C0 LDA $C0C9
C42F- 8D FF C4 STA $C4FF
C432- 38 SEC
C433- A9 00 LDA #00
C435- ED FE C4 SBC $C4FE
C438- 8D FE C4 STA $C4FE
C43B- A9 00 LDA #00
C43D- ED FF C4 SBC $C4FF
C440- 8D FF C4 STA $C4FF
C443- 60 RTS

```

C400.C443

```

C400- 20 0C C4 4C 59 FF 20 26
C408- C4 4C 59 FF A9 E0 8D CB
C410- C0 A9 01 8D C8 C0 A9 00
C418- 8D C9 C0 A9 EC 8D C4 C0
C420- A9 13 8D C5 C0 60 AD C8
C428- C0 8D FE C4 AD C9 C0 8D
C430- FF C4 38 A9 00 ED FE C4
C438- 8D FE C4 A9 00 ED FF C4
C440- 8D FF C4 60
*
```

成語言被 BASIC 語言蓋掉的情形發生。行 110 是呼叫要測量執行時間的副常式。當程式執行由副程式回返時，即呼叫中斷常式以中斷計時器之動作。然後程式即跳到計算程式執行時間的常式去。行 994 即用以計算且顯示程式之執行時間。

6522 內部移位暫存器之程式規劃

6522 內部移位暫存器，可用作串聯的輸入／輸出埠。利用此暫存器，可以接收由 CPU 送出的並行資料，且轉換成串行資料以送到外

部裝置去。或是你可以利用此暫存器將外界輸入的串聯資料，轉換成並行的資料然後送給 CPU。有一點要特別提到的是，若要使用移位暫存器作上述功用，則可以使用一個外部時脈或 CPU 本身之時脈，或是你也可以利用 6522 內部計時器來設計屬於自己的時脈。此處所提到之移位暫存器之作業型態，主要是由 ACR 暫存器的 b4b3b2 三位元之狀態來設定。下表所示就是設定這 3 位元所對應的各種作業型態。

ACR4	ACR3	ACR2	Mode
0	0	0	Shift Register Disabled
0	0	1	Shift in under control of Timer 2
0	1	0	Shift in at System Clock Rate
0	1	1	Shift in under control of external input pulses

圖 15 移位暫存器作業型態之設定

上面提到 6522 的移位暫存器可用來作為串聯的輸入／輸出埠。而在 6522 上主要即是利用 CB2 接腳來完成這些傳輸工作。即經由這個 CB2 接腳，串聯的輸入／輸出埠可以對移位暫存器進行讀／寫的動作。假如你計劃使用外部的時脈來控制串聯的輸入／輸出動作，則必須將外部時鐘信號送到 CB1 去。反之，若使用內部時脈，則 CB1 將當作一個激發 (strobe) 信號，以便和 CB2 的進出資料保持同步。

由上述說明可以知道，CB1 可用作一個同步脈衝，至於是輸出一個同步脈衝，或接受外部時鐘脈衝的輸入，則是由輔助控制暫存器中

的 b4、b3、b2 來決定。假如你是使用計時器來作為內部時脈，則只能有一個 8 位元的計時器和移位暫存器相連接，且最低的移位頻率大約是 0.5ms。

一個可變工作周期的方波產生器

對於上面所提到的移位暫存器而言，若改變輔助控制暫存器中 b4、b3、b2 中的位元組合形式，則移位暫存器將會改變所產生方波的工作周期。且若改變計數器的 T2L/L 之閂鎖值，則允許你改變方波產生器所產生的時鐘脈波頻率。如圖 16 所示程式，即可完成這項功能。

圖 16

```

PR#1
0800          1          DCM "PR#1"
0800          2          ;
0800          3          ;
0800          4          ;*****
0800          5          ;*
0800          6          ;* VARIABLE DUTY CYCLE
0800          7          ;* SQUAREWAVE GENERATOR
0800          8          ;*
0800          9          ;*****
0800         10          ;
0800         11          ;
0800         12          ;
0800         13          ACR      EQU %C0CB
0800         14          T2LL     EQU %C0C8
0800         15          SR       EQU %C0CA
0800         16          MONITO   EQU %FF59
0800         17          ;
0800 A9FF      18          LDA    #$FF
0802 8DC8C0    19          STA    T2LL
0805 A910      20          LDA    #$10
0807 8DCBC0    21          STA    ACR
080A A90F      22          LDA    #$0F
080C 8DCAC0    23          STA    SR
080F 4C59FF    24          JMP    MONITO
0812          25          ;
0812          26          ;
0812          27          END

```

;SET TIMER 2 FOR SLOWEST
;FREQUENCY
;SET OPERATION MODE

;4 TIMES ZERO AND 4 TIMES
;ONE TO THE SR

前面曾提到 6522 I/O 界面板，可利用高階語言或低階語言來控制，而在上面已列舉了許多組合語言的應用例。爲了對 I/O 界面板的應用能有更多的了解。下面將再列舉出一個以 FORTH 語言寫成的程式，此程式可用來控制埠 A 的 8 個輸出腳。此程式之所以用 FORTH 語言來寫，主要是因爲以 FORTH 語言來寫控制程式要較機械語言容易，且在速度上要較 BASIC 快。爲了配合程式的控制，在硬體上需先完成下列工作：將 8 個 LED'S 接至 6522 晶片的埠 A（或許你也可以用別的面來取代 LED'S 以作別的應用）。有一點要注意的是在埠 A 所連接的裝置，不可以由電腦吸取太大的電流。在此程式中假設連接到埠 A 的 8 個 LED'S 編號爲 1~8，且分別由埠 A 的位元 b0~b7 來控制。這個 FORTH 語言寫成的程式，可以控制某編號的 LED 之點亮或熄滅。即操作時只要鍵入所要控制之 LED 編號，然後鍵入 ON 或 OFF 即可。

圖 17 以 FORTH 語言寫成的 LED 驅動程式

```
: START HEX 00 FF C0C3 11 ;
: AN C0C1 11 ;
: NR 1 0 2 UNDER SWAP DO 2• LOOP 2/ ;
: NEW 2 UNDER OR DUP ;
: ON NR NEW AN ;
: NEC 2 UNDER SWAP COMPLEMENT AND DUP ;
: OFF NR NEC AN ;
```

```
START
2 ON
3 ON
2 OFF
```

程式說明

圖 17 所示就是以 FORTH 語言寫成的程式。在程式的第一行定義了 START 這個指令之功能，是設定埠 A 之資料方向暫存器（置於位址 \$C0C3）之內含值爲 \$FF（255），即將埠 A 設定爲輸出，且將零置於堆疊器（STACK）頂端。在第二行是將 AN 值存入位址 \$C0C1（埠 A）中。在第三行即定義 NR 爲所要點亮或熄滅之 LED 編號。在呼叫 NR 之

前，這個編號是置於堆疊器之頂端，在進入 DO 迴路後，堆疊器的頂端是 1，且 N 是迴路的上級。在程式中是將堆疊器頂端的 1 乘以 2（作左移），且左移 N 次以指出所要點亮或熄滅之 LED。

圖 18 所示爲整個 6522 I/O 界面板的完整線路圖。在圖形的右上方可以看到兩個編號爲 U 2 和 3 的 2114 RAM，它們是由 APPLE II 周邊連接器上的輸入／輸出選擇線（I/O SEL）來選擇，而 6522 則是由 APPLE 周邊連接器上的裝置選擇線（DEV SEL）來加以選擇。至於 6522 的選擇線 RS0~RS3，分別連接到位址匯流排上的 A₀~A₃。而編號爲 U 4 的 IC 正如前面所提到的，它主要是利用系統 ϕ 2 時脈來提供所需的時間延遲。由圖形左方可看出，6522 之輸出線是被分別連接到兩個不同的連接器去。

6522 I/O 界面板之組裝

爲了清楚起見，將 6522 I/O 界面板上的零件安置情形顯示如下。另外在界面板上的兩條跳線也在圖中以虛線表示出來。

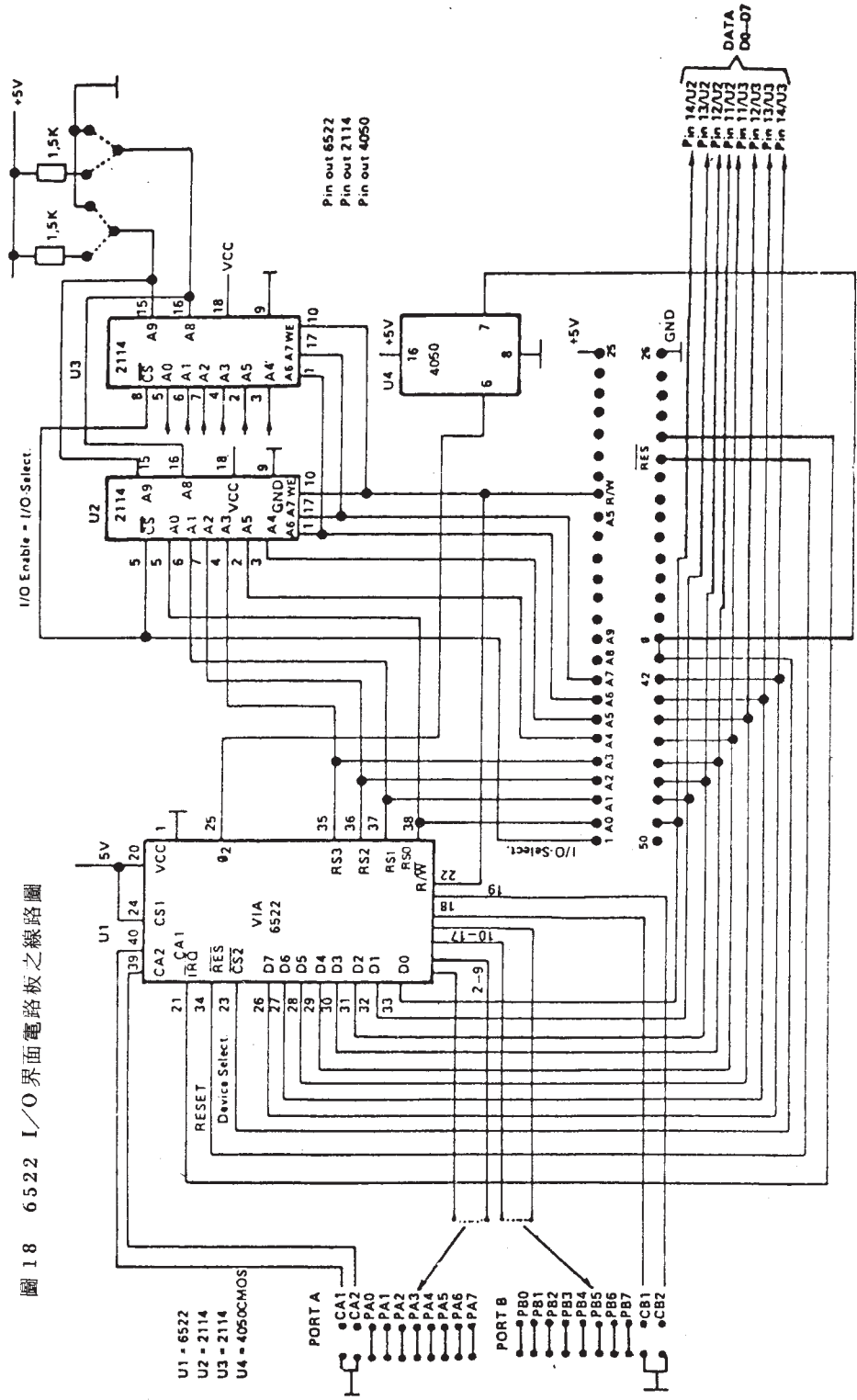
下面所要介紹給各位的是一個非常有實用性的技術——用 Apple 來控制前級馬達——希望本文能給一般對於 Apple 應用方面有興趣的讀者有所助益。

以 6522 作介面電路控制步級馬達

步級馬達，正如它的名字所含義的意思，表示馬達的工作是在一個時間裏作一級一級的轉動。同時它每一級轉動的大小是可以由我們來設定的。但是爲了要達到步級馬達的最大效率，每一級的轉動範圍，最好是在 0°~90° 之間較爲適當。由於步級馬達具有很高的精確性，且容易和電腦配合使用。所以大多數的電腦週邊設備所使用的馬達，都採用步級馬達。譬如磁碟機、印表機，以及機器人……等。

本文中，所使用的馬達是 Airpex 公司出品的 K82416-P2 12V 的步級馬達，（當然你也可以使用其他的 12V 步級馬達）。馬達每一

圖 18 6522 I/O 界面電路板之線路圖



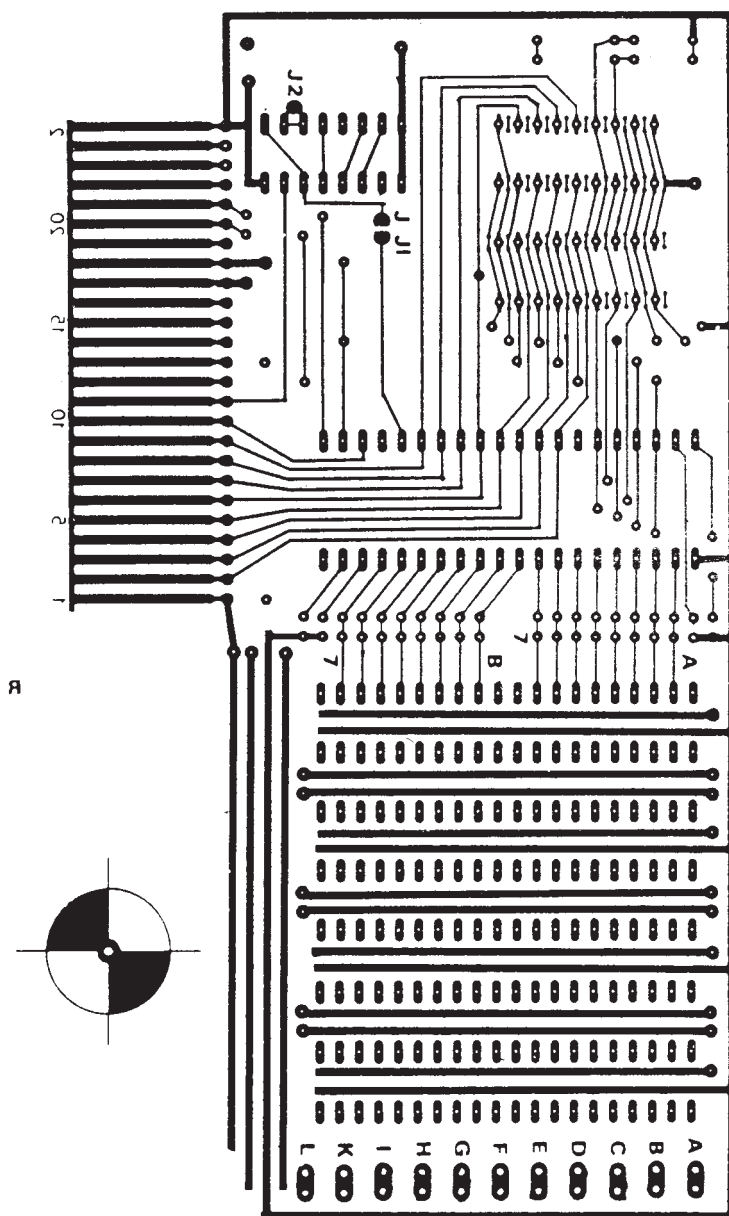
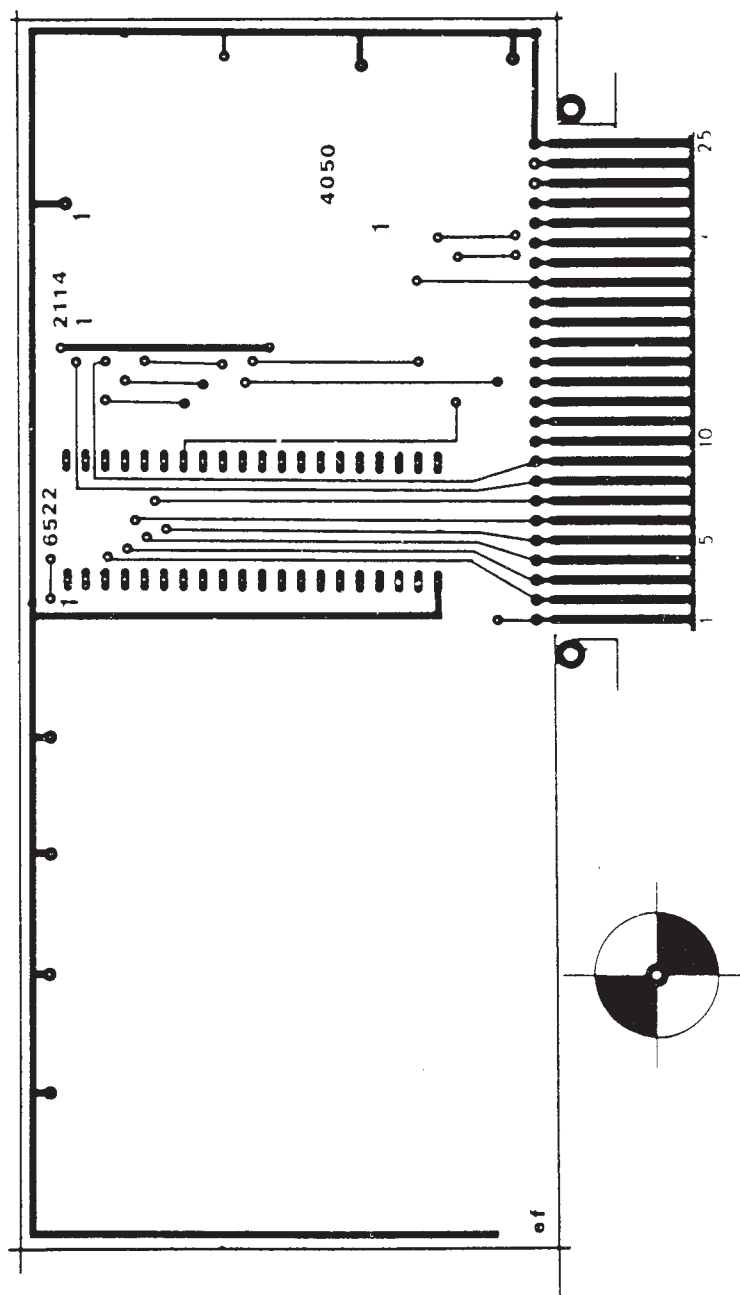


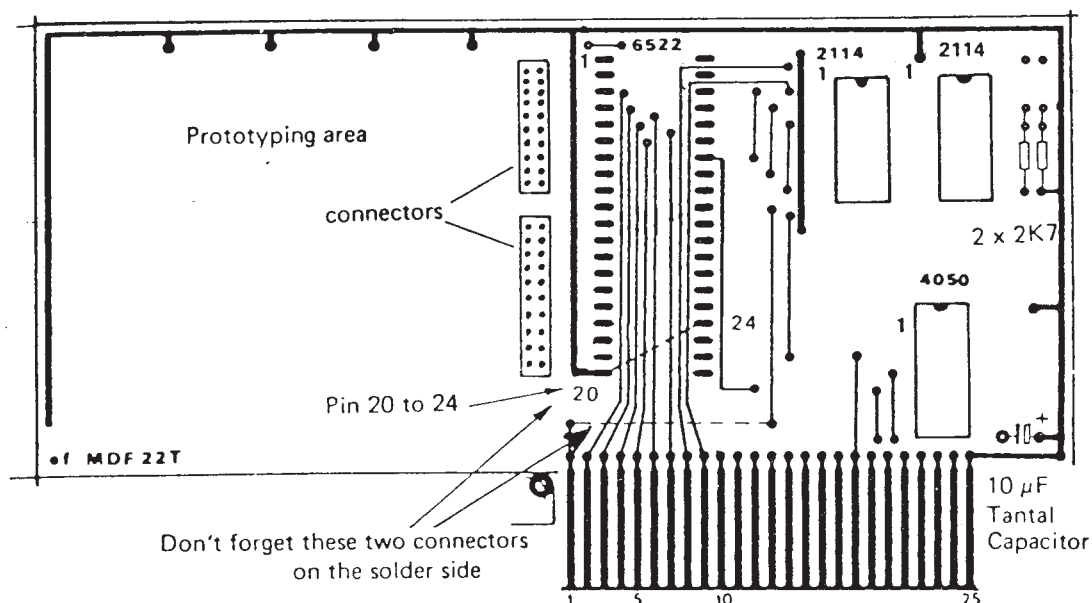
圖 19 I/O 界面板之印刷電路板 ①底面(背面)



② 零件面（正面）

圖 20 6522 I/O 界面板之零件佈線圖

2 pol DIL
switches



Qty	Description
1	Capacitor tantal 10 μ F/35V
1	DIP switch, 2 poles / 3 poles
2	Connectors with 20 pin each, for port A and B connectors
1	40 pin socket DIL
2	18 pin socket DIL
1	16 pin socket DIL
1	6522 VIA (Rockwell)
1	4050 Motorola
2	2114 L RAM chips Synelec or Rockwell
1	6522 / I / Board

圖 21 6522 I/O 界面板之零件表

約是 \$42 (有齒輪比), \$38 (無齒輪比)。

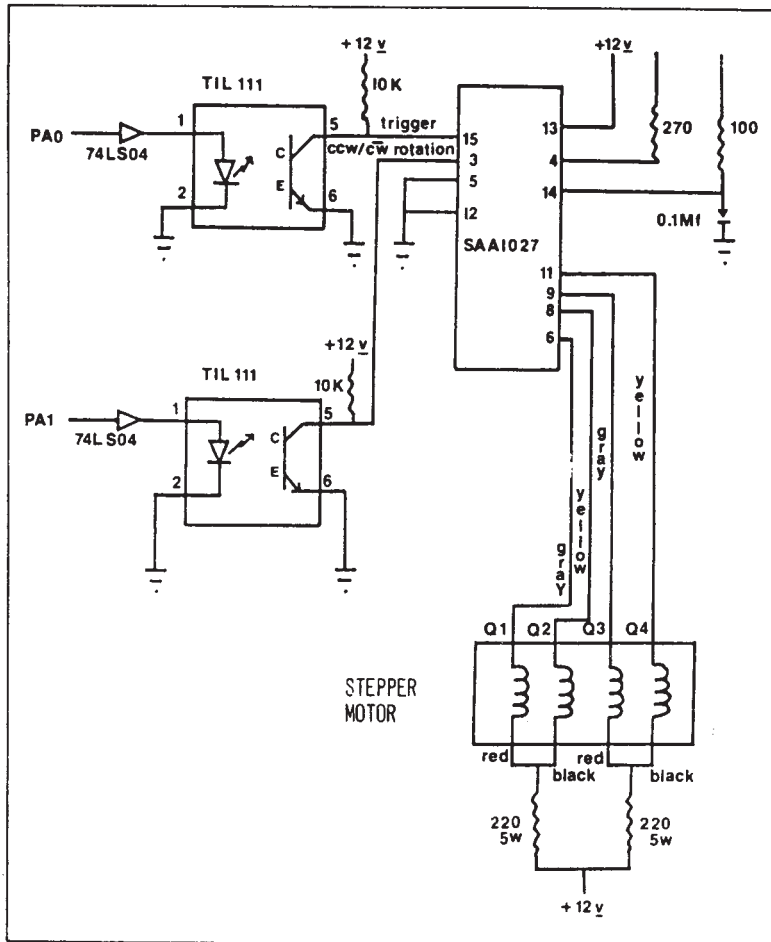
Apple II 它使用一個 6522 介面卡來推動步級馬達, 6522 它提供二個 "Ports", 一個輸入 ports 和輸出 ports。但在這裏我們祇要一個 port 就夠用了。

當然 6522 並不一定絕對需要, 但是, 在 Apple 中, 因為它能提供簡單的方法來連接資料匯流排 (DATA BUS) 和位址匯流排 (ADDRESS BUS), 基於這個理由, 所以我們用 6522 來做為介面板, 以推動步級馬達。6522 的參考價格約為 \$65。

6522 不管是作為 A/D, D/A, 或 STEPPER (步級馬達) 等介面工作, 你可以把 6522 板子插在任何 1~7 的插座上。

如果我們使用 SAA1027 這個馬達驅動器, 那麼步級馬達和電腦的介面工作就變得更為

級的大小, 被調整為 1.5 度, 同時還有一個 5:1 的這輪比。它比一般沒有齒輪比的馬達更能提供較大的輸出馬力。而這個馬達的價格大



簡單了。SAA1027 這個 IC 是用來驅動 12V 的步級馬達。使用這個晶片來驅動步級馬達，我們需要從 Apple 拉出二條控制信號。第一個輸出訊號是由 6522 的其中一脚拉出，用來控制馬達轉動的方向，當這個輸出訊號是“低狀態”（LOW）時，馬達就作順時針轉動。相反的，當訊號是“高狀態”（High）時，馬達就作逆時針轉動。第 2 個輸出訊號，它必順經常的隨著程式作“ON”“OFF”的改變，因為當我們改變這個訊號的 High、LOW，將會使馬達重新選擇轉動的方向。

以 TIL 擔任光電耦合

整個電路如圖所示，其中兩個 TIL111 的光電耦合器（opto-isolators）是用來隔離和保護電路不受外部電路的影響，因為任何介面的應用，總是需要將電腦和外部電路隔離，以避免因外部電路的損壞，而造成過大的電流脈衝流進 Apple。尤其在馬達介面應用上，這一點更顯得格外的重要。因為馬達所產生的巨大脈衝電壓所造成的危險性，將會影響到 Apple。

從 6522 介面板輸出的電流因為不足以推動光耦合器，所以我們可以使用 2 個 7SLS04 IC，來增加電流量，以便推動光耦合器。另外從圖中我們看到光耦合器和一個 10K Ω 的電阻加在一起。這個 10K Ω 的提升電阻，是用來將電腦輸出的電壓提升為 12V，以提供給 SAA1027 使用。另外在電路上還有一些電阻，電容元件，它們都是配合 drive SAA1027 和馬達使用的，至於值的大小也都是根據 drive SAA1027 和馬達所設定的特定值。

至於控制馬達的電腦程式：表(一)所列的程式是說明我們可以由鍵盤上的兩個按鍵“C”、“W”來選擇馬達旋轉的方向。當你按“C”鍵時，馬達就作順時針轉動，按“W”時馬達就做逆時針轉動。

表(二)所列的程式：你可以按“C”或“W”使步級馬達每級做 90° 的轉動。當然你要每一級作 90° 轉動，便要觸發 60 次。（因為正常情況每級是 1.5 度）

表(一)程式的說明

- 20 → 49411 是 6522 PORT A 的資料方向暫存器的記憶位置。填 0 代表輸入，填 1 代表輸出。255 則是設定它為輸出。
- 30 → 等待輸入，按“C”表示作順時針轉動。按“W”表示逆時針轉動。
- 40 → 若按 C 鍵，它將改變 PORT A 的資料。記憶位置 49409 將 POKE 2 和 0 這兩個 bit 資料將使 PA₀ 為 LOW，因為馬達信號是連接到 PA₀，所以馬達會順時針轉動。同時會使 PA₁ “ON”，然後再“OFF”，並回到 30 行。
- 50 → 按“W”鍵，使馬達作逆時針轉動。同時使 PA₁ 作“ON”然後再“OFF”。並回到 30 行。

有關於 6522 的應用，讀者可以參考 50 期。

```

10 HOME: LIST
20 POKE 49411,255
30 GET A$
40 IF A$ = "C" THEN POKE 49409,2:POKE 49409,0:GOTO 30
50 IF A$ = "W" THEN POKE 49409,3:POKE 49409,1:GOTO 30
60 REM PA1 USED FOR TRIGGER
70 REM PA0 USED FOR DIRECTION
80 REM PRESS W FOR COUNTERCLOCKWISE ROT
90 REM PRESS C FOR CLOCKWISE ROT
95 REM PRESS W OR C WHILE HOLDING DOWN REPEAT KEEPS MOTOR TURNING

```

表一 設定馬達轉動方向

```

10 HOME: LIST
20 POKE 49411,255
30 GET A$
40 IF A$ = "C" THEN FOR I = 1 TO 60: POKE 49409,2:POKE 49409,0:NEXT I:
GOTO 30
50 IF A$ = "W" THEN FOR I = 1 TO 60: POKE 49409,3:POKE 49409,1:NEXT I:
GOTO 30

```

表二 設定步級為 90 度

A/D D/A 界面卡 製作與應用

8 bit 的 D/A 和 A/D 轉換器

本文將要介紹一個數位至類比(D/A)，以及類比至數位(A/D)轉換器之應用。首先要介紹的是由Ferranti所推出的數位至類比轉換器ZN428E IC。假如你了解如何利用APPLE II個人用電腦，來作為家庭或工業上的資料存取、狀態感知等等控制系統，則首先必須能夠將一些特定的數值轉換成一個電壓值(即D/A轉換)。例如，假設你要利用程式的轉換產生一些特定的電壓水平，則必須先產生一個數位化的數值，然後將數位化數值轉換成電壓值。因此為了數位至類比轉換後，能取得適當的電壓值，所以數位化數值的產生是必須的。為了能達到上述所說數位至類比轉換的目的，可以利用本文介紹的ZN428E D/A轉換器來完成，但實際的轉換則是藉著電腦中的軟體(程式)來完成。

電路構成

圖1所示是一個8位元數位至類比和類比至數位轉換器的完整圖形，在圖中的6522 VIA(Versatile Interface Adapter)是作為電腦和轉換器間的一個界面，一般稱為多用途界面轉換器。由圖上可以看到，ZN428E

晶片的資料輸入線，連接到6522的輸出埠A輸出埠A之線A0，連接到ZN428E數位至類比轉換器之資料線中的最小位元(Bit 0)，而A7則是連接到數位至類比轉換器之最大位元(bit 7)。ZN428E是利用6522 VIA的腳PB0來致能(enable)。當PB0=0時，數位至類比轉換器可以接收由6522輸出埠A所送過來的所有資料。當PB0=1時，則所有的輸入都會被ZN428E立即鎖住，而且必須一直維持在那個狀態，直到PB0=0為止。也就是最後送入轉換器的資料，將被儲存在ZN428E轉換器中。至於ZN428E的輸出電壓範圍是由運算放大器(1/4 TL074)來設定(調整)。ZN428E之內部參考電壓(VRF)是使用2.5V。圖2的方塊圖，是顯示如何調整輸出電壓之範圍。

圖1是顯示一個電路，其輸出電壓變化是0和+5V或0和-5V之間。但這個電壓並不是一個交變電壓，即它不是正的就是負的，下列的公式就是用來計算單極性輸出電壓(VFS)的數學式：

$$VFS = (1 + R1/R2) * VRF$$

由上面這個公式所產生的電壓變化，是位於0V和極大值(VFS)之間。由圖2中得知R1和R2所產生的並聯等效阻抗，應該近似

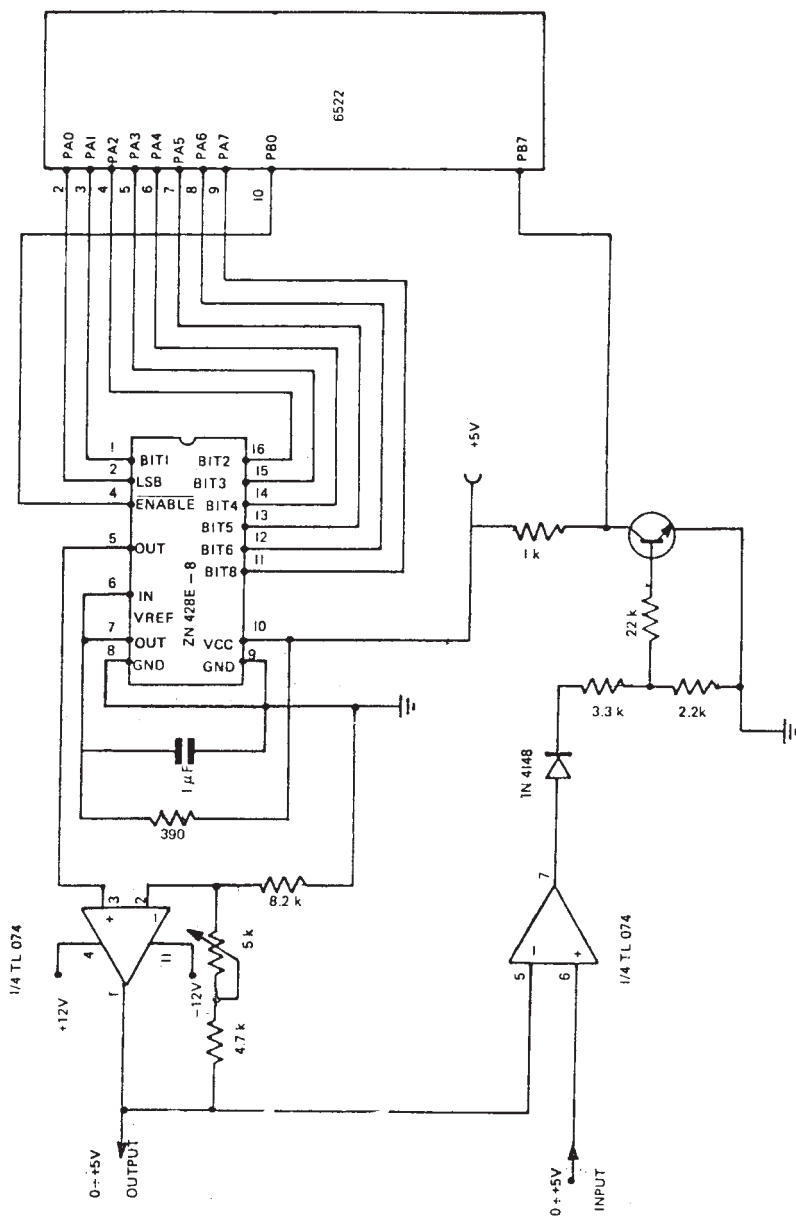


圖 1 8 位元 A/D 和 D/A 轉換電路

於轉換網路的內部阻抗。這個電阻應該近似於 4000 歐姆。對於一個輸出電壓是在 0 和 +5 V 之間且參考電壓 $V_{REF} = 2.5V$ 而言， $R_1 = R_2 = 8000$ 歐姆。

在圖 1 中 $R_2 = 8200\Omega$ 且 R_1 是 4700 Ω 和 5000 Ω 電位器之串聯值。利用這個結構可產生的最大輸出電壓是 +5 V。爲了要達到這個目的，可利用下列程式來完成。

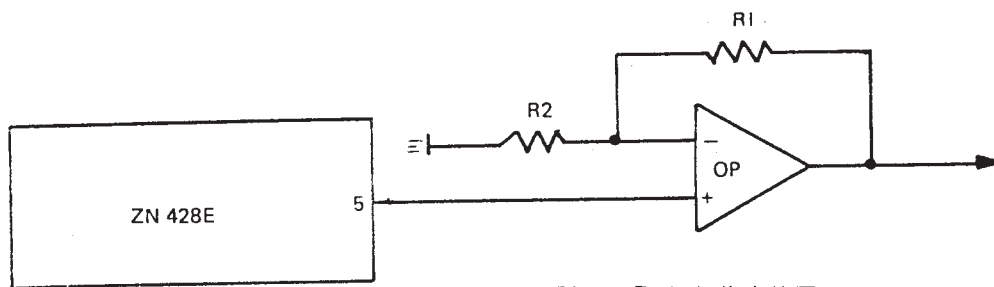


圖 2 D/A 轉換方塊圖

圖 3 轉換器輸出電壓調整之程式

```

10 REM *****
20 REM *      CONVERTER ADJUST      *
30 REM *****
100 REM PROGRAMMING THE PORTS
110 REM PORTA SET TO OUTPUT
120 POKE - 16189,255
130 REM PORTB SET TO OUTPUT
140 POKE - 16190,01
200 REM OUTPUT OF NUMBERS
210 INPUT " NUMBER=";Z
220 POKE - 16191,Z
230 PRINT "MORE (Y/N)";: GET W$
240 IF W$ < > "N" THEN 210
250 END

```

輸入電壓之大小

當 6522 VIA 界面電路板被插入 APPLE II 的界面連接器 Slot # 4 時，6522 之埠 A 和埠 B 的位址是 \$C0C1 和 \$C0C0。埠 A 的等效十進位位址是 -16192，埠 B 的等效十進位位址是 -16191，而資料傳送方向之暫存器 DDRB 和 DDRA 的位址是 \$C0C2（十進位是 -16190）和 \$C0C3（十進位是 -16189）。在執行圖 3 之小程序時，電腦會要使用者輸入一個數值。假如鍵入 255，則表示將轉換器之輸出電壓設定為最大值。然後可使用 5KΩ 電位器來將 +5V 電壓調整成 4.98V

。為了能完成這個準確的電壓調整，最好是採用數字電壓表。因為 +5V 等於 255 之數值。而電路最大僅能產生 \$FF(255)。因此就必須依上述方法將 +5V 電壓調整為 4.98V（即 +5V - 20mV）。而這裏所提到的 20mV 正好對等於最小位元之值。假如不是如上面所說輸入 255，而是輸入 0，則輸出電壓也一定是 0。另外也可嘗試輸入其它的數字，然後觀察一下運算放大器之腳 8 的輸出電壓。若輸入的數字是 128，則輸出電壓應該是 2.5V。

D/A 轉換之應用示範

有了上述觀念後，下面將利用 3 個以 6502

機械語言寫成的程式，來說明數位／類比（D／A）轉換器在APPLE II電腦上的工作情形。



圖4 鋸齒波產生器

圖5 用以產生鋸齒波（SAWTOOTH）之程式

```

0800          1          DCM "PR#1"
0800          2          ;
0800          3          ;
0800          4          ;*****
0800          5          ;*
0800          6          ;*   SAWTOOTH
0800          7          ;*
0800          8          ;*****
0800          9          ;
0800         10          ;
0800         11  DDRA    EQU $C0C3
0800         12  DDRB    EQU $C0C2
0800         13  TORA    EQU $C0C1
0800         14  TORB    EQU $C0C0
0800         15          ;
0800 A9FF      16          LDA #$FF
0802 8DC3C0    17          STA DDRA
0805 A901      18          LDA #$01
0807 8DC2C0    19          STA DDRB
080A A200      20          LDX #$00
080C 8EC1C0    21  M      STX TORA
080F E8        22          INX
0810 18        23          CLC
0811 90F9      24          BCC M
0813          25
          26          END

```



圖6 三角波產生器

程式說明

上面列出了三個功能不同的程式；下面就分別的來說明一下這些程式：

圖5所示之鋸齒波程式，是利用將索引暫存器X增值且將暫存器之內含值存在6522的輸出埠A。程式在一開始時就將輸出埠A和輸出埠B的PB0設定為輸出。而這些工作是利用將\$FF載入累積器且存入DDRA，以及將\$01載入累積器且送至DDR B來完成。

圖 7 用以產生三角波 (TRIANGLE) 之程式

```

0800          1          DCM "PR#1"
0800          2          ;
0800          3          ;
0800          4          ;*****
0800          5          ;*
0800          6          ;*      TRIANGLE      *
0800          7          ;*
0800          8          ;*****
0800          9          ;
0800         10          ;
0800         11  DDRA     EQU  $C0C3
0800         12  DDRB     EQU  $C0C2
0800         13  TORA     EQU  $C0C1
0800         14  TORB     EQU  $C0C0
0800         15          ;
0800 A9FF         16          LDA  #$FF
0802 8DC3C0       17          STA  DDRA
0805 A901         18          LDA  #$01
0807 8DC2C0       19          STA  DDRB
080A A200         20          LDX  #$00
080C 8EC1C0       21          STX  TORA
080F EEC1C0       22  M1     INC  TORA
0812 D0FB         23          BNE  M1
0814 CEC1C0       24  M2     DEC  TORA
0817 D0FB         25          BNE  M2
0819 F0F4         26          BEQ  M1
081B             27          ;
                28          END

```

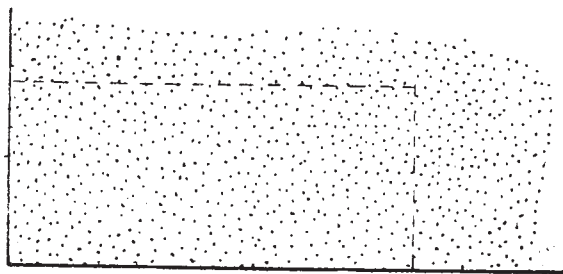


圖 8 二進制雜訊產生器

圖 7 所示之三角波產生器程式，也是同上面一樣的方法，來設定輸出埠 A 和 B 的值。然後再將一個零存入輸出埠 A。三角波的產生即是將輸出埠 A 的內含值增值來完成，直到輸出

埠 A 之值為零為止。然後輸出埠 A 又被減量，直到其值為零。重覆的執行上述程式迴路，即可產生三角波。

圖 9 用以產生二進制雜訊 (BINARY NOISE) 之程式

```

0800          1          DCM "PR#1"
0800          2          ;
0800          3          ;
0800          4          ;*****
0800          5          ;*
0800          6          ;*  BINARY NOISE
0800          7          ;*
0800          8          ;*****
0800          9          ;
0800         10          ;
0800         11  DDRA     EQU  $C0C3
0800         12  DDRB     EQU  $C0C2
0800         13  TORA     EQU  $C0C1
0800         14  TORB     EQU  $C0C0
0800         15          ;
0800         16  ZAHL     EPZ  $10
0800 A9FF         17          LDA  #$FF
0802 8DC3C0       18          STA  DDRA
0805 A901         19          LDA  #$01
0807 8DC2C0       20          STA  DDRB
080A 201308       21  M      JSR  RANDO
080D 8DC1C0       22          STA  TORA
0810 18           23          CLC
0811 90F7         24          BCC  M
0813             25          ;
0813 38           26  RANDO  SEC
0814 8511         27          STA  ZAHL+1
0816 6514         28          ADC  ZAHL+4
0818 6515         29          ADC  ZAHL+5
081A 8510         30          STA  ZAHL
081C A204         31          LDX  #$04
081E B510         32  Z1     LDA  ZAHL,X
0820 9511         33          STA  ZAHL+1,X
0822 CA           34          DEX
0823 10F9         35          BPL  Z1
0825 60           36          RTS
0826             37          ;
                   38          END

```

圖 9 所示之二進制雜訊產生程式，是利用一個稱為 RANDO 的副程式，來產生 0 至 255 的隨機數字。程式使用由標題 ZAHL 至 ZAHL + 5 所定義的記憶位置，作移位且加上某些特定的數字。這些數字再被傳送到 6522 的輸出埠 A，然後送至數位至類比轉換器的輸入端。

由上面的說明可以知道，只要預先設定好一些特定的數字，然後經由程式的控制，配合 ZN428E 的轉換，可產生各種不同的函數圖形。

A/D 轉換原理及適用範圍

直到現在，我們僅討論到 ZN428E 數位至類比轉換器在 D/A 方面的應用。而這個高功能的晶片，也允許你利用它來組成一個類比至數位轉換器，但其轉換工作必須要配合 AP-PLE II 的特殊軟體（程式）來完成。有一點要提到的是：電腦僅能工作於某些特定的電壓

，即僅能處理二進制的 0（“L”）和 1（“H”）。但是在我們周遭的信號，例如，溫度、壓力、光、聲音強度以及由換能器所產生的各種信號，都是類比形式的電壓或電流信號。要想讓電腦來處理這些信號，則必須將這些類比形式的電壓信號轉換成數位形式。

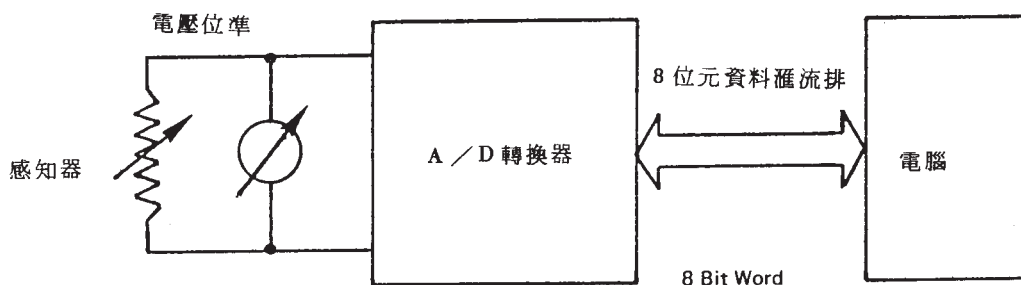


圖 10 A/D 轉換器的方塊圖

有許多種方法可以將一個待測電壓轉換成數位化的數值。首先，要介紹一個積分形式的類比至數位轉換器。仔細看圖 11 可以了解，當

類比開關 S 被一個脈波合上時，則積分器開始在運算放大器 OP1 的輸出端產生一個斜坡（RAMP FUNCTION）波形。

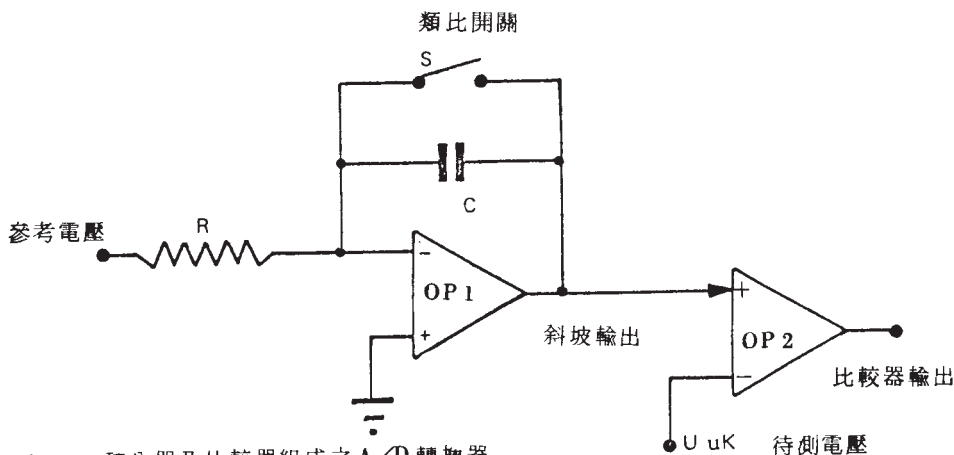
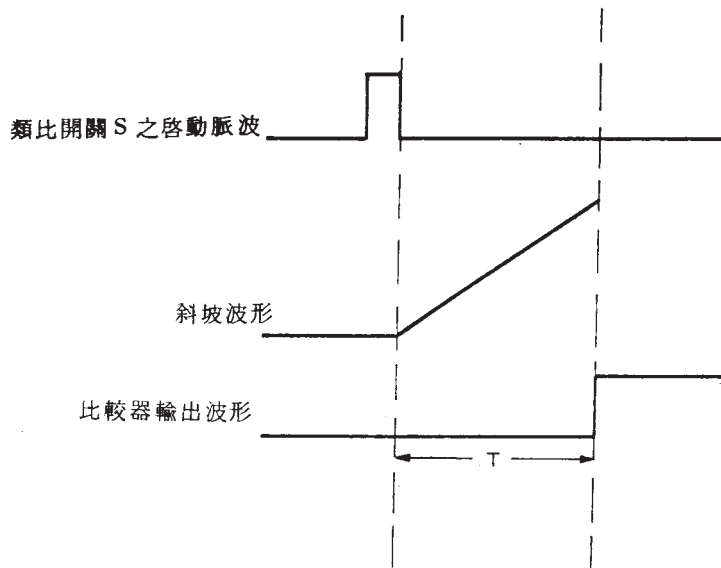


圖 11 積分器及比較器組成之 A/D 轉換器

研究圖 11 可以知道，由積分器所產生之電壓被送至比較器之非反相輸入端，然後與一個待測電壓（以 U_{uk} 表示）相比較。當積分器之輸出電壓達到待測電壓值時，則比較器的

輸出由 L 變成 H。至於啟動脈波（start pulse）開始至比較器之輸出發生轉態（由 L 至 H）之時間，可以利用數位計數器來測量。



利用計數器測量時間 T

圖 12 利用斜坡 (RAMP) 函數所產生的數位轉換

上圖所示之電路可用作單斜率 (single slope)，雙斜率或三斜率 (TRIPLE SLOPE) 之轉換器。另一個將電壓轉換成數位的方

法，是使用一個數位階梯函數 (Digital step function)。

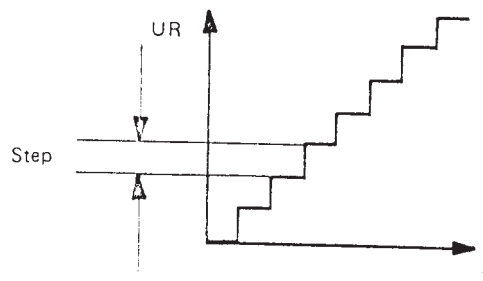


圖 13 階梯波

圖 13 所示之階梯波形，是用來和待測電壓相比較。當兩者相等時，則計數停止，且此時計數器計數之階梯脈波的數目正好等於待測電壓之值。這是一種非常慢速的轉換。第三種

轉換方法是連續趨近 (successive approximation) 方法，也是本文在應用時將要採用的方法。詳細情形待後面再討論。

A/D 轉換各種參數說明

目前在市面上有許多 A/D 轉換器。只要加上一些電阻以及 555 計時電路，即可組成一個很好的線路。一般而言，這些轉換器並不很精確，而且大都數都是配合搖桿（joysticks）使用，另外也可以應用到溫度測量和控制方面。但是常常聽到有人在討論 A/D 轉換器時，總會提到解離度（resolution）、準確性（accuracy）、線性（linearity）、安定時間（setting time）和鐘脈率（clock rate）等名詞。現在就來說明一下這些類比至數位轉換器的重要特性，以便對於 A/D 轉換器的適用範圍能有所了解。

解離度

所謂解離度（Resolution）是表示要使 A/D 轉換器的輸出產生增值，所需的輸入電壓變化量。一個具有 N 個開關的轉換器，可以將一個全刻度電壓解離成 2^N 個刻度。

至於輸入信號則是由一系列的數位化階梯波摹擬來完成。解離度可以按全刻度或二進制位元來表示。例如：一個解離度為 12 位元的 A/D 轉換器，可將一個全刻度電壓分解成 2^{12}

個刻度。也就是說一個刻度等於全刻度的 $1/4096$ 或等於全刻度的 0.0245% 。例如，一個具有 10V 全刻度的轉換器，可以分辨出 2.45 mV 的輸入變化。假如將上述規格和一個 8 位元的 A/D 轉換器相比較時，則將發現 8 位元的 A/D 轉換器僅具有 $1/256$ (0.3906%) 的解離度。因此對於一個 10V 的全刻度，它僅提供 39mV 的解離度。此處所提到的解離度是一個設計參數，對於準確度和線性而言，它實在是一個很重要的參數。

準確度

準確度（Accuracy）是描述實際輸入電壓和全刻度二進制輸出電壓間的一個差異。這其間含有量化誤差（quantizing error）和其他誤差在內。一個 12 位元的 A/D 轉換器是以 ± 1 LSB 為準確度。這相當於 0.0245% ，或兩倍於最小可能量化誤差 (0.0122%)。

量化誤差

所謂量化誤差（Quantizing error），對於一個理想的 A/D 轉換器而言，它是 A/D 轉換器上一個直接性轉換函數的極大變異。

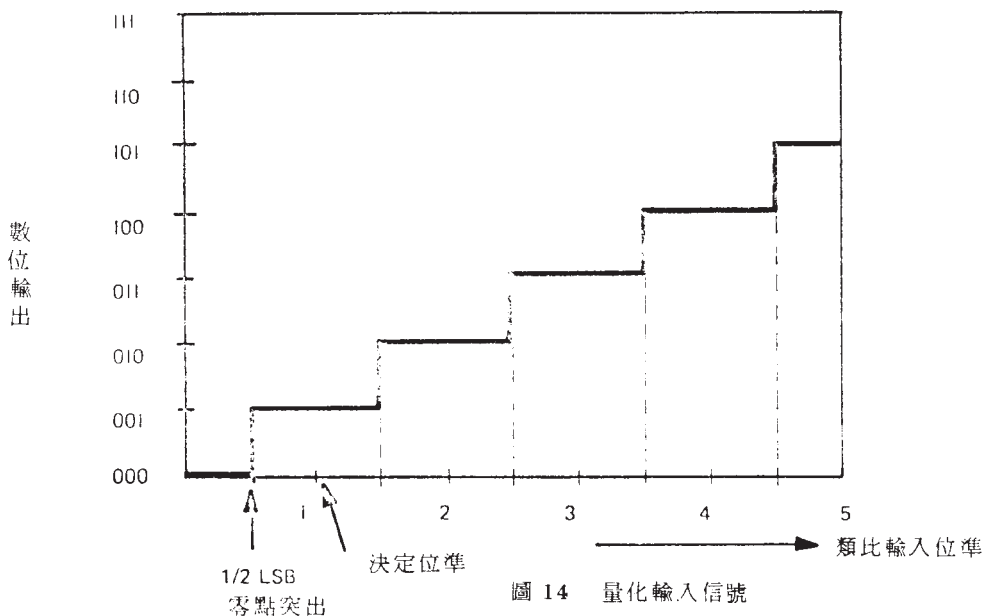


圖 14 量化輸入信號

如上圖所示，A/D轉換器將類比輸入量化成有限的數位碼輸出。

轉換／鐘脈率

所謂轉換率 (Conversion rate) 是 A/D 轉換器能連續完成資料轉換的速率。這個項目主要會受到各組成元件之傳輸延遲 (propagation delay) 的影響。由這裏的說明可以知道，轉換率可以定義為每秒鐘轉換的數目，或完成一個轉換所需要的時間 (單位 μs)。鐘脈率 (Clock rate) 是表示 A/D 轉換器計數器被推動之最小或最大脈波率。

由於在這裏我們是利用 D/A 轉換器來完成 A/D 轉換之功能，因此必須藉助電腦的軟體來完成這個功能。這個程式所使用的是一種稱為連續趨近 (successive approximation) 的方法。一個輸入到 ZN428E 的待測電壓是和全刻度電壓之一半進行比較。這個全刻度電壓，在本文所舉的例子是 +5V。假如輸入電壓是高於全刻度電壓的一半時，則電腦會繼續另一個比較週期，即將輸入電壓和 $\frac{3}{4}$ 全刻度電壓之值相比較。同理重複此步驟，即可取得待測輸入電壓之近似值。

如上所述，在經過 8 個比較後，則表示轉換完成。輸入電壓現在可以按 $\pm 20mV$ 的精密度被分辨出來。

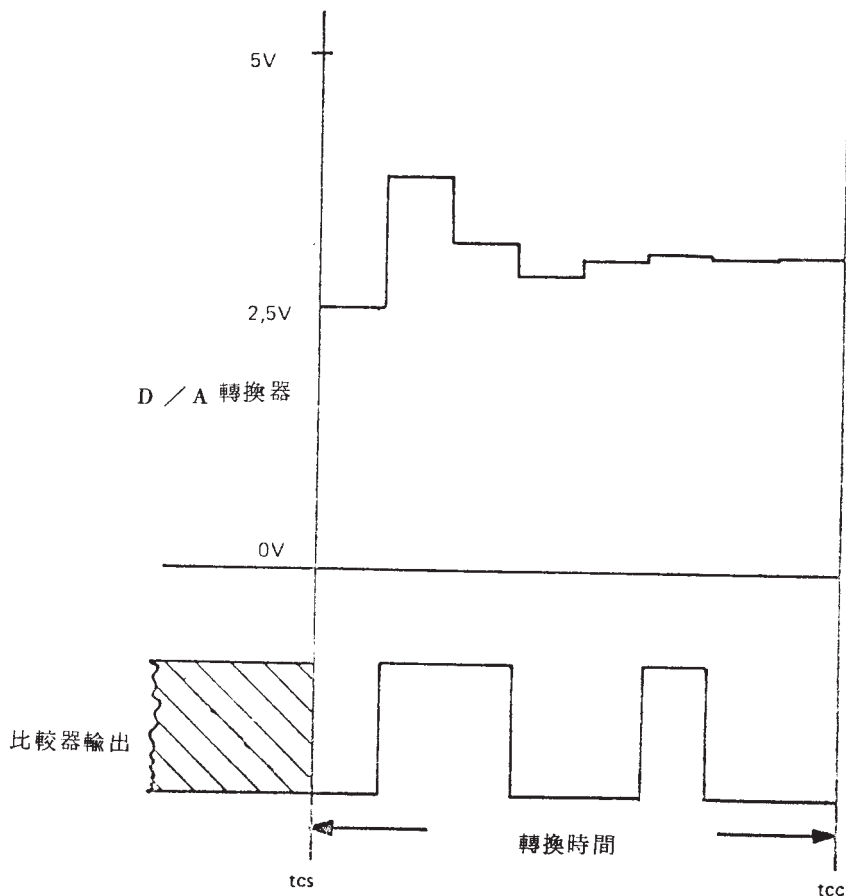


圖 15 利用連續趨近方法來完成 A/D 轉換器

由圖 15 可以看到，利用 D/A 轉換器以及比較器來完成 A/D 轉換之情形。在圖形的上半部是 D/A 轉換器的輸出；而圖形的下半部則是比較器的輸出。仔細觀察該圖可以知道轉換是由 tcs 開始，而在這時間之前，比較器的狀態則是未知的。在轉換一開始，輸入電壓是首先和 2.5V 進行比較，且在比較器接受輸入電壓之前，比較器之輸出是處於低狀態（L）。接著輸入電壓是和 3.75V（2.5 + 1.25）比較。當比較器輸出為高狀態（H），則表示輸入電壓小於 3.75V：所以這個電壓仍不會被比較器接受。接著第三個比較是將輸入電壓和 3.125V（2.5 + 0.625V）相比較，若輸入電壓小於 3.125V，則這個電壓也不會被接受，且比較器的輸出電壓仍然是 1。依照上面的方法，下一個比較電壓是 2.8125V（即 2.5 + 0.3125），此時比較器接受這個電壓，而且在輸出端產生一個零的輸出。對於電腦而言，它會將這個接受的電壓值標識為

1。所以由上面所討論的四種情形得知，數位數字的最後 4 個位元值恰為 1001。設轉換繼續下去，且接受下一個（第 5 個）電壓值，拒絕下下一個（第 6 個）電壓值及接受後兩個（第 7、8 個）電壓值。即整個的數位化數字最後變成 10011011 = \$9B（十六進位）。這相當於 1 個 3.099V 的電壓。但由於量化誤差的原因，所以輸入電壓位準是近似於 3.099 ± 20mV。由圖中可以看到，轉換是在 tcc 完成。

假如你要測量一個 10V 的輸入電壓。則必須使用一個電壓分壓器電路。但是誤差將會是雙倍（± 40mV）。而比較器 C 2（圖 1）的輸出信號將會是 +12V ~ -12V。為了要將此比較器輸出連接到 6522 晶片的 PB 7 輸入腳，則必須將比較器之輸出轉換成能和 TTL 匹配的電壓位準。要完成 A/D 轉換所需之程式如下所示。

圖 16 用以完成連續趨近功能之程式

```

0800          1          DCM "PR#1"
0800          2          ;
0800          3          ;*****
0800          4          ;*
0800          5          ;* ANALOG-DIGITAL-CONVER- *
0800          6          ;* SION BY SUCCESSIVE *
0800          7          ;* APPROXIMATION WITH A *
0800          8          ;* 8-BIT DA-CONVERTER *
0800          9          ;*
0800         10          ;*****
0800         11          ;
0800         12          DDRA EQU $C0C3
0800         13          DDRB EQU $C0C2
0800         14          TORA EQU $C0C1
0800         15          TORB EQU $C0C0
0800         16          VALUE EQU $C4FF
0800         17          Z EPZ $10
0800         18          PRTBYT EQU $FDDA
0800         19          ;
0800 2000C4      20          JSR INIT
0803 200BC4      21          JSR CONVER
0806 ADFFC4      22          LDA VALUE
0809 20DAFD      23          JSR PRTBYT

```

```

080C 00      24          BRK
C400         25          ORG $C400
C400         26          ;
C400         27          ;SET THE 6522 PORTS
C400         28          ;
C400 A901    29  INIT    LDA #$01
C402 8DC2C0  30          STA DDRB
C405 A9FF    31          LDA #$FF
C407 8DC3C0  32          STA DDRA
C40A 60      33          RTS
C40E         34          ;
C40B         35          ; CONVERT
C40B         36          ;
C40B A980    37  CONVER  LDA #$80
C40D 8510    38          STA Z
C40F A97F    39          LDA #$7F
C411 8DC1C0  40  W0      STA TORA
C414 EA      41          NOP
C415 EA      42          NOP
C416         43          ;ONLY NECESSARY BECAUSE
C416         44          ;OF SLOW COMPARATOR
C416 EA      45          NOP
C417 EA      46          NOP
C418 ACC0C0  47          LDY TORB
C41E 1002    48          BPL W1
C41D 0510    49          ORA Z
C41F 4610    50  W1      LSR Z
C421 B004    51          BCS FIN
C423 4510    52          EOR Z
C425 90EA    53          BCC W0
C427 8DFFC4  54  FIN     STA VALUE
C42A 60      55          RTS
C42B         56          ;
C42B         57          ;

```

在行 29 至行 33 的指令是用來設定資料方向暫存器 (data direction registers)。轉換程式由行 37 開始。此處是將記憶位置區的 bit 7 之初值設定為 1(H)。第一次是與 \$7F 作比較。假如輸入電壓較高，則在行 48 不會有 BPL 發生。然後行 52 的 OR 指令，會將累積器的 bit 7 設定為 1(H)。在 Z 將右移動一個位元後，其值為 \$40。利用 EOR 指令，則累積器的 bit 6 將會被清除。其內含值現在變成 \$BF，且被存入輸出埠 A。在能夠利

用 LDY 指令讀出輸出埠 B 的指令之前，轉換器必須允許被調整。由於 ZN428E 非常的快速，所以在 800 μ s 後就可以再讀取新的類比輸入信號。但就另一方面來說，以 TL074 所構成的比較器在速度上慢很多。為了解決這個問題，所以必須在程式中插入四個無運算 (NOP) 指令。當 LSR Z 指令將標識位元 (marked bit) 移入進位位元時，轉換即告結束。

圖 17 繪圖程式

```

10 REM *****
12 REM      * PLOTTING A CURVE      *
14 REM      * ON THE APPLESCREEN    *
18 REM *****
50 D$ = CHR$ (04)
60 PRINT D$;"BLOAD ADWC400.B"
100 INIT = - 15360:WA = - 15349
110 VA = - 15105
120 CALL INIT
200 HGR : COLOR= 15
210 X = 0
220 CALL WA
230 W = PEEK (VA)
240 P = 160 - W / 2
250 HPLOT X,P
260 X = X + 1
270 IF X < 280 THEN 220
280 END

```

圖 17 的 BASIC 程式，將轉換後的電壓值送到APPLE 螢幕上。因為有 255 種不同的電壓，但在螢幕上的垂直方向僅有 160 個畫素 (pixels) 可資利用，所以可將各個電壓值在顯示之前先分成兩個部分，也就是說將僅使用

127 個畫素變化 (pixels range) 來顯示所有的電壓值。圖形的零點是被置於距螢幕頂端 160 個畫素的下方。在每一次測量之後，X 值將被加 1。假如想要降低測量率，則可以在程式之行 270 插入一個延遲迴路 (delay loop)

圖 18 ADW C4φφ.B 程式

```

C400-   A9 01      LDA    #$01
C402-   8D C2 C0   STA    $C0C2
C405-   A9 FF      LDA    #$FF
C407-   8D C3 C0   STA    $C0C3
C40A-   60         RTS
C40B-   A9 80      LDA    #$80
C40D-   85 10      STA    $10
C40F-   A9 7F      LDA    #$7F
C411-   8D C1 C0   STA    $C0C1
C414-   20 2A C4   JSR    $C42A
C417-   AC C0 C0   LDY    $C0C0
C41A-   10 02      BPL    $C41E
C41C-   05 10      ORA    $10
C41E-   46 10      LSR    $10
C420-   B0 04      BCS    $C426
C422-   45 10      EOR    $10
C424-   90 EB      BCC    $C411
C426-   8D FF C4   STA    $C4FF
C429-   50         RTS
C42A-   A2 10      LDX    #$10

```

C42C-	CA	DEX	
C42D-	D0 FD	BNE	\$C42C
C42F-	60	RTS	

C400.C42F

C400-	A9	01	8D	C	C0	A9	FF	8D
C408-	C3	C0	60	A9	80	85	10	A9
C410-	7F	8D	C1	C0	20	2A	C4	AC
C418-	C0	C0	10	02	05	10	46	10
C420-	B0	04	45	10	90	EB	8D	FF
C428-	C4	60	A2	10	CA	D0	FD	60

*

圖18所示之轉換程式ADW C400.B 是被存入 6255 I/O board 的RAM 上。如此一來可避免 BASIC 程式發生錯誤時程式被毀壞。假如將 6255 VIA 界面電路板插入周邊插槽 slot #4, 則RAM記憶區的程式起始位址是 \$C4 $\phi\phi$ 。INIT 副程式是用來設定資料方向暫存器。WA 是轉換程式。轉換值將被存入記憶位置 \$C4FF (等效十進位位址是 -151 -15105, 參見圖 17 的程式列表)。在這些位置的數值將被傳送至 BASIC 程式。程式

ADW C4 $\phi\phi$ -B 中的指令 STA \$C ϕ C1 和指令 LDY \$C ϕ C ϕ 間, 被插入一個延時程式, 以解決比較器的時間問題。假如在電路中是使用快速的比較器, 如 LM393, 則對於電壓的比較而言, 可以省去這個副程式。然後在執行指令 \$C ϕ C1 後, 便可以立即取得比較的結果。假如採用如圖19所示之電路, 則必須將記憶位置 \$C41A 之跳躍指令改成 BMI \$C41E, 此時其轉換時間是近似於 220 μ s。

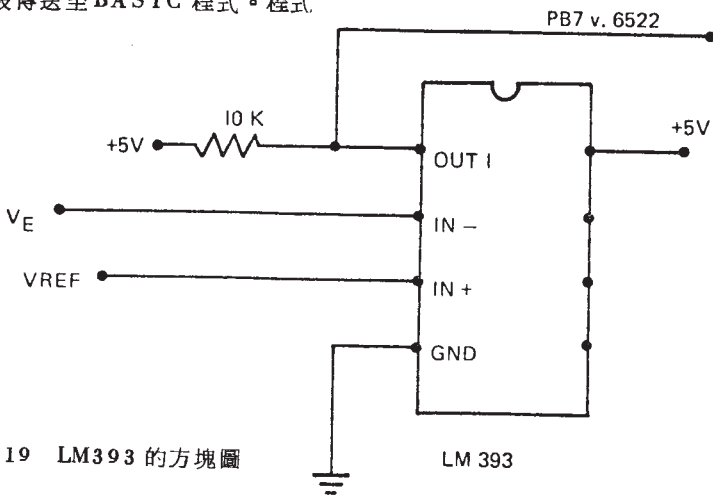


圖 19 LM393 的方塊圖

對每一項精確的類比至數位轉換而言, 在轉換期間輸入電壓的變化應該不要超過最小位元大小的一半。在本文所舉的例子中, 就是說在轉換期間輸入電壓的變化不能超過 10mV。由這裏我們可以計算出, 所允許的快速電壓變化是 45.5V/sec。於一個具有 2.5V 振幅

的信號而言, 我們僅能取得 3 週期/sec 的上限頻率。因此所能測量也是較此頻率為低的情形。

D/A A/D 加上 6522 I/O 界面組裝

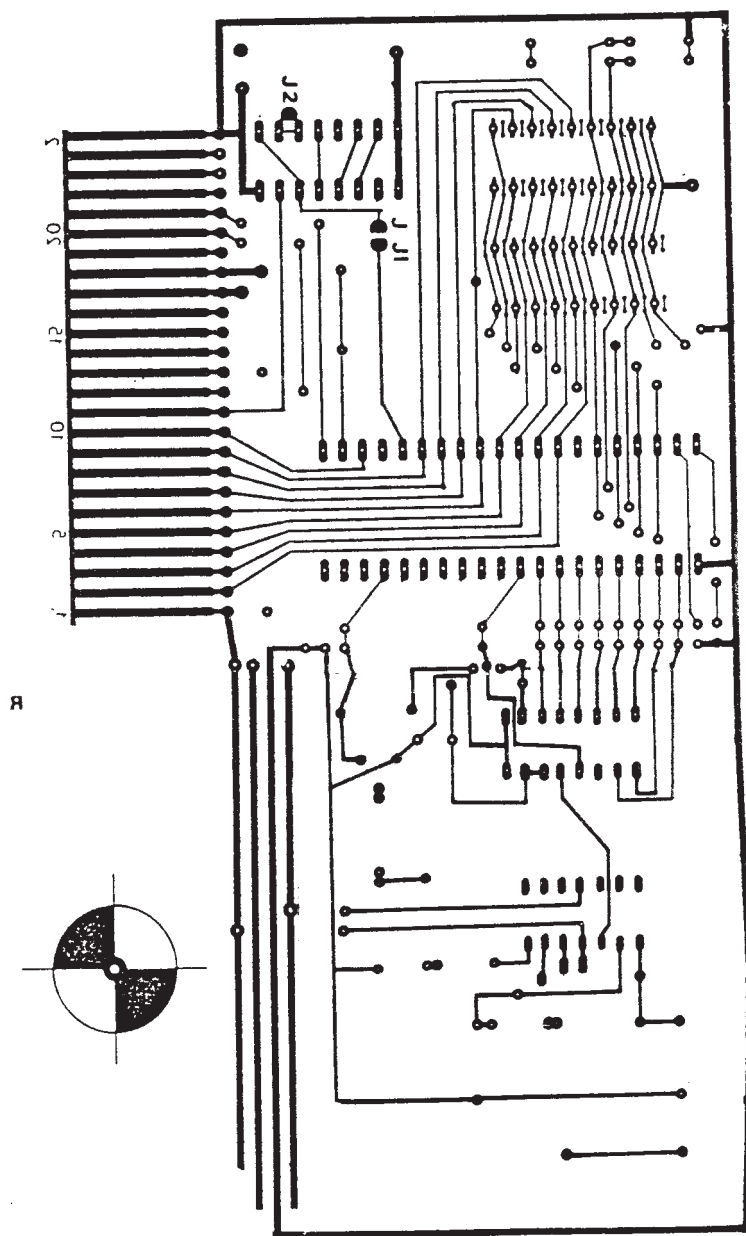
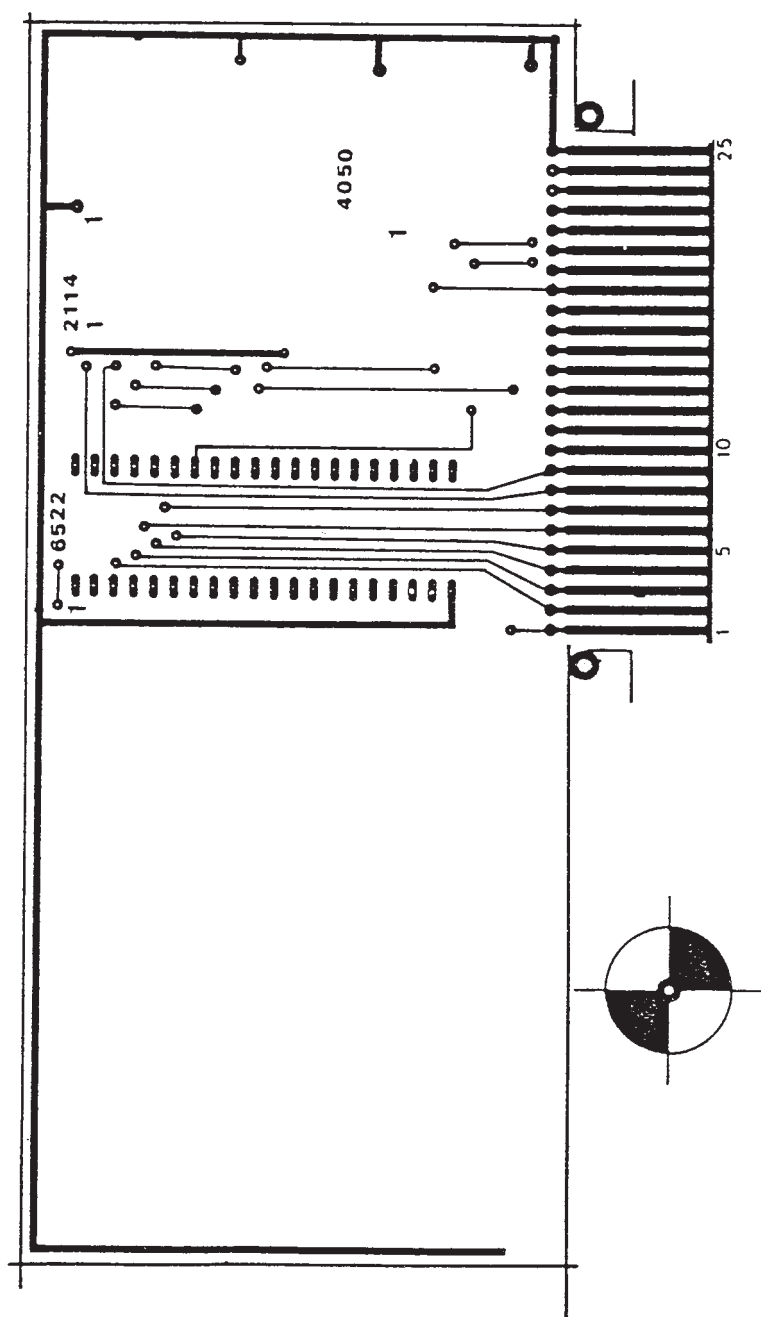


圖 20 A/D、D/A 界面板之印刷電路板①底面（背面）



②零件面（正面）

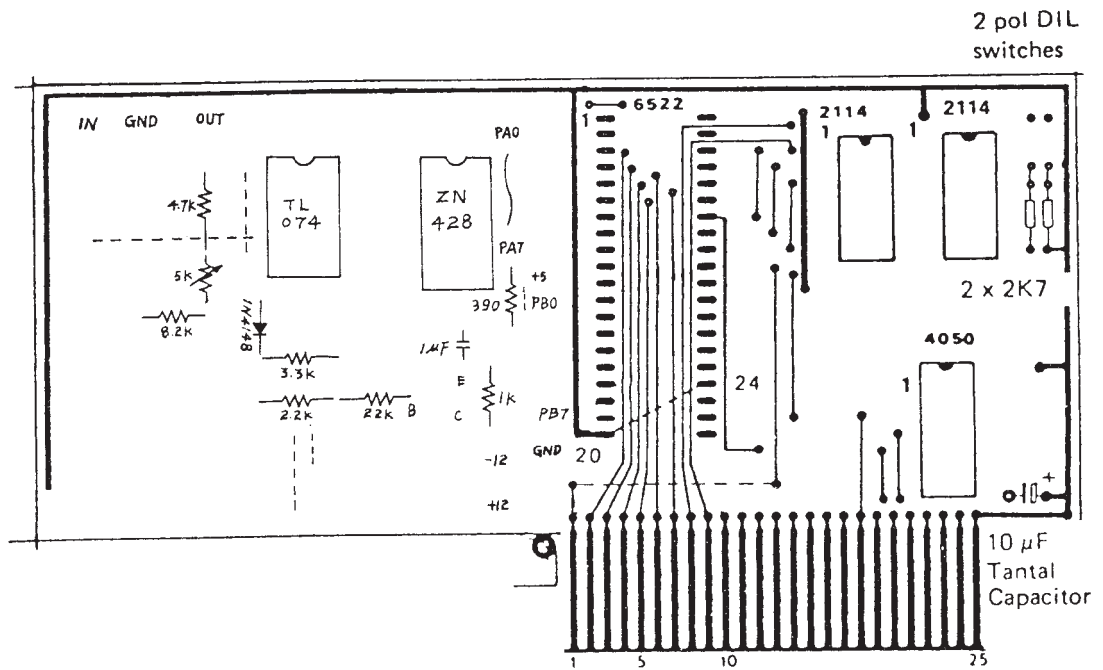


圖 21 A/D、D/A 界面板零件佈置圖

附圖 單獨A/D、D/A 基板圖

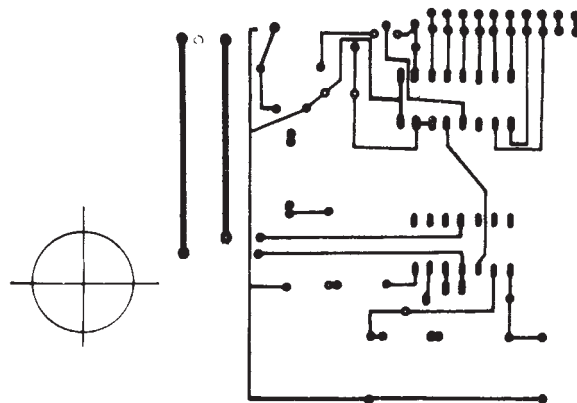
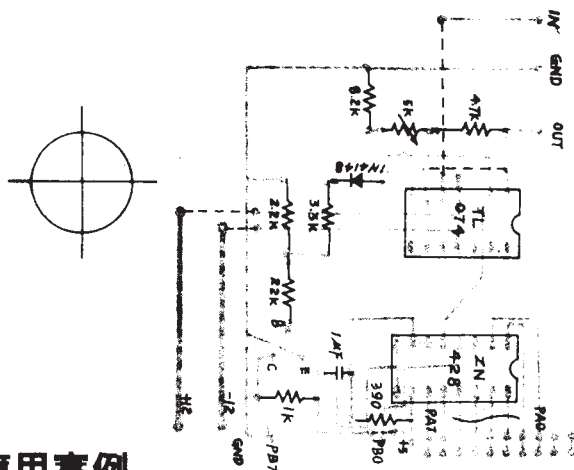


圖 20 為 A/D、D/A 界面板之印刷電路板銅箔面（雙面），圖 21 為 A/D、D/A PCB 之零件佈置圖，附圖是單獨製作一片 A/D、D/A 板，要用時再用連線接到 6522 板之插座

上。

下面給各位介紹 A/D、D/A 應用的實例，希望它的應用對您有所幫助。



A/D，D/A應用實例

使用兩個D/A轉換器

在許多應用中，若在同一電腦中使用兩個 D/A 轉換器，則應用起來將會非常的方便。這些應用可能包含將轉換的結果在 X/Y 描圖器上或 X/Y 儲存式示波器上顯示出來。此時所能看到的僅有波形的變化情形。或許也可以利用這些方式去控制許多馬達和機械臂以產生更複雜的波形，這些應用可參見圖 22 的圖形說明，在圖中直流馬達是由兩個放大器 A 1 和 A 2 驅動。而這些放大器的輸入電壓是由 D/A 轉換器 DAC1 和 DAC2 所提供。

由圖 22 的結構知，其速度和時間之關係如圖 23 所示。

此處所介紹的系統，可以很容易地擴展為數位控制系統。利用一個 A/D 轉換器，可以測量光強度、溫度、壓力等等。電腦會計算出系統所必須的反應，然後利用上面所提到的電路（圖 22）來產生動作。對於這項應用，我們使用兩個 ZN425E D/A 轉換器（裝在 6522 I/O board 上）。

在圖 24 中，U 2 的資料線是被連接到輸出埠 A，而 U 4 的資料線則是連接到 6522 的

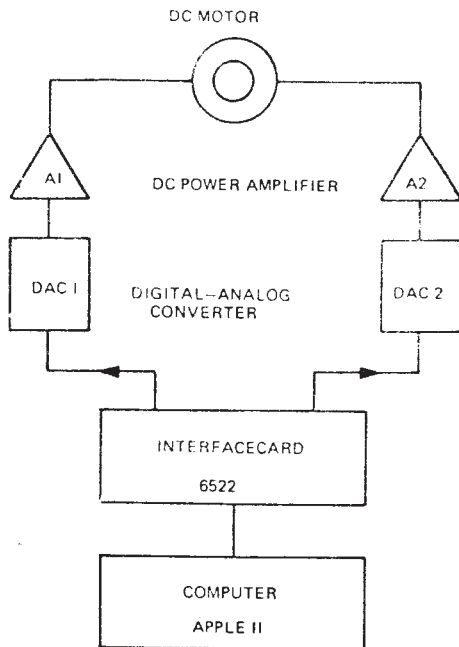


圖 22 直流馬達控制

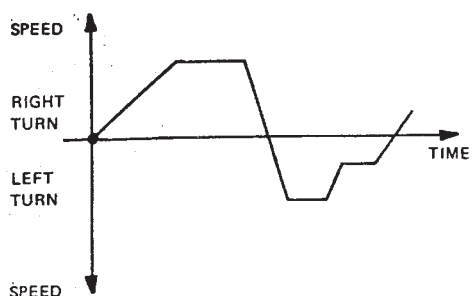


圖 23 速率／時間函數

輸出埠B。圖中的兩個運算放大器U3和U5，是用來測量ZN425E的輸出電壓(V_{OUT})和參考電壓(V_{REF})的差異。運算放大器U3和U5之輸出(腳6)電壓之變化是在 $+2.75V \sim -2.75V$ 。而 $+2.75V$ 的電壓值是對應於\$FF的輸入。 $-2.75V$ 是對應於\$00。當輸入為\$80(十進制128)時，輸出電壓是0V。在下列說明中，我們將考慮三個程式：一個是以BASIC寫成，另兩個是以機械語言寫成。在BASIC程式(圖25)中，我們將計算一個圓且使用兩個D/A轉換器以求出的數值畫在一個示波器的螢幕上。

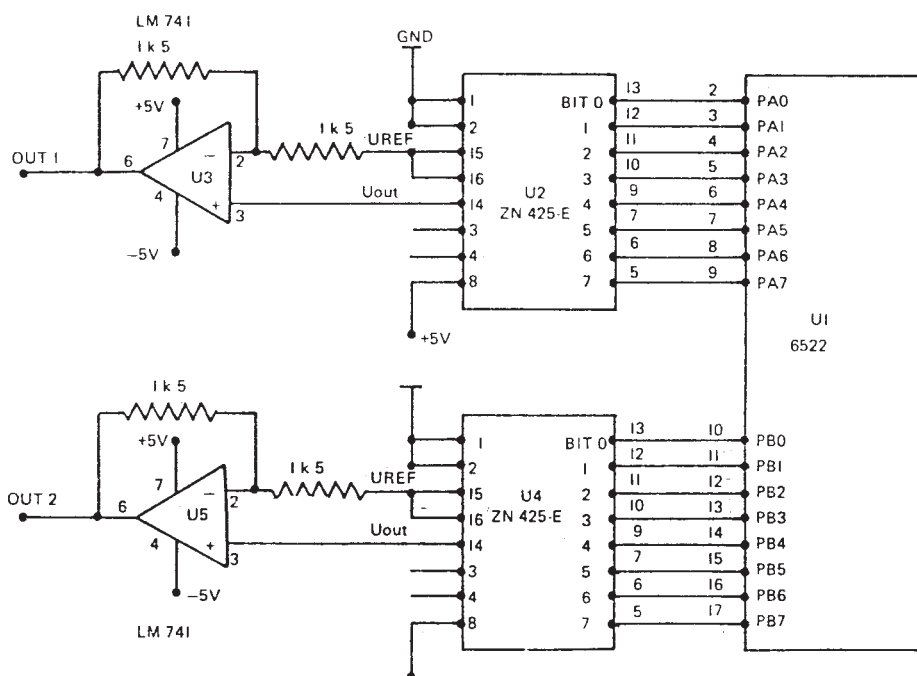


圖 24 將DAC ZN425E 連接到6522之情形

圖 25 圓周繪圖程式

```

1  REM *****
2  REM * PLOTTING A CIRCLE *
3  REM *****
10 POKE - 16190,255: POKE - 16189,255
20 TA = - 16191:TB = - 16192
100 PI = 3.14159
105 F = 2 * PI / 360
110 FOR T = 0 TO 360 STEP 5

```

```

120 X = SIN (T * F)
122 X = X * 127 + 128
125 Y = COS (T * F)
127 Y = Y * 127 + 128
130 POKE TB,X
135 POKE TA,Y
140 NEXT : GOTO 100

```

在圖 25 程式中的行 10和行 20，主要是用來設定資料方向暫存器，且將 T A 的值設定為輸出埠 A 的位址，T B 的值設定為輸出埠 B 的位址。至於其他的程式行，主要是用來計算圓周的值，此圓周中心是 X=128 和 Y=128。這也是 X 和 Y 座標的零電壓點。在行 130 和 135 主要是將計算的 X 和 Y 座標值存入 (POKE)輸出埠 A 和 B。而埠 B 的輸出電壓是連接到示波器的 X 輸入端。埠 A 的輸出電壓則是連接到示波器的 Y 輸入端。此時若注意示波器上螢幕的變化，則會看到光束慢速的環繞著兩軸旋轉。這個現象主要是由於兩個 POKE 指令所造成的時間延遲。或許你已注意到 BASIC 程式在速

度上並不太快。但無論如何，假如你使用 X / Y 描圖器來取代示波器，則這個速度將是正確的繪圖速度。在程式中亦可從事下列修改，例如，可以將程式行 110 改成 FOR T=0 TO 360 STEP 2。則可以獲得較圓的圓周。假如你將行 120 改成 X=SIN(2 *T * F)，則可以取得頻率比為 2：1 李沙育圖形。

下面就來討論一下，另兩個以機械語言寫成的程式。這些程式由於是以機械語言寫成，所以在執行上速度要快很多。圖 26 所示的機械語言，可以在示波器的螢幕上繪出一個方形。

圖 26 方形繪圖程式

```

0800          1          DCM "PR#1"
0800          2          ;
0800          3          ;
0800          4          ;*****
0800          5          ;*
0800          6          ;*      SQUARE
0800          7          ;*
0800          8          ;*****
0800          9          ;
0800         10          ;
0800         11 DDRA      EQU $C0C3
0800         12 DDRB      EQU $C0C2
0800         13 TORA      EQU $C0C1
0800         14 TORB      EQU $C0C0
0800         15          ;
0800 200608 16          JSR INIT
0803 4C0F08 17          JMP SQUARE
0806         18          ;
0806 A9FF   19 INIT      LDA #$FF
0808 8DC3C0 20          STA DDRA
080B 8DC2C0 21          STA DDRB
080E 60     22          RTS
080F         23          ;

```

```

080F          24 ;
080F A000     25 SQUARE LDY #$00
0811 A200     26         LDX #$00
0813 8EC1C0   27         STX TORA
0816 8CC0C0   28         STY TORB
0819 E8       29 S1      INX
081A 8EC1C0   30         STX TORA
081D E0FF     31         CPX #$FF
081F D0F8     32         BNE S1
0821 C8       33 S2      INY
0822 8CC0C0   34         STY TORB
0825 C0FF     35         CPY #$FF
0827 D0F8     36         BNE S2
0829 CA       37 S3      DEX
082A 8EC1C0   38         STX TORA
082D D0FA     39         BNE S3
082F 88       40 S4      DEY
0830 8CC0C0   41         STY TORB
0833 D0FA     42         BNE S4
0835 F0D8     43         BEQ SQUARE
0837          44 ;
                45
END

```

在上圖所示的程式，在執行時是首先設定資料方向暫存器以及副程式的初值。然後程式即會在螢幕上之左下角繪出一個方形。藉著將數值存在輸出埠A，則將會繪出方形的左邊。在增值達到\$FF後，則Y暫存器開始增值，且將數值存在輸出埠B，將可以使方形的右邊在螢幕上繪出來。至於方形的其它兩邊，則是首先將X暫存器，然後將Y暫存器減值，同時將這些數值對應的存入輸出埠A和B，即可在螢幕上繪出方形的另兩邊。由螢幕上可以看出機械語言程式的執行速度要較BASIC快的多

。正因為如此，所以不會再看到光束環繞的情形，此時所看到的將是直接產生一個方形的圖形。

最後要討論的機械語言程式是圖 27 所示的隨機繪圖(RANDOM WALK)程式，此處使用了前面提到過的RANDO副程式來產生隨機變數。這些數字被連續的輸出到埠A和B中。此時若注意著螢幕，則將會發現許多點被安排成方形，且向右移動，在上面所討論的這些例子，主要是說明要如何來將兩個D/A轉換器連接至電腦去，且用程式來控制其動作。

圖 27 隨機繪圖程式

```

0800          1          DCM "PR#1"
0800          2 ;
0800          3 ;
0800          4 ;*****
0800          5 ;*
0800          6 ;*  RANDOM WALK
0800          7 ;*
0800          8 ;*****
0800          9 ;
0800         10 ,
0800         11 DDRA   EQU $C0C3
0800         12 DDRB   EQU $C0C2

```

```

0800          13  TORA  EQU $C0C1
0800          14  TORB  EQU $C0C0
0800          15  ;
0800          16  ZAHL   EPZ $10
0800          17  ;
0800 A9FE      18      LDA #$FF
0802 8DC3C0    19      STA DDRA
0805 8DC2C0    20      STA DDRB
0808 201708    21  M    JSR RANDO
080B 8DC1C0    22      STA TORA
080E 201708    23      JSR RANDO
0811 8DC0C0    24      STA TORB
0814 18        25      CLC
0815 90F1      26      BCC M
0817          27  ;
0817 38        28  RANDO SEC
0818 8511      29      STA ZAHL+1
081A 6514      30      ADC ZAHL+4
081C 6515      31      ADC ZAHL+5
081E 8510      32      STA ZAHL
0820 A204      33      LDX #$04
0822 B510      34  Z1   LDA ZAHL,X
0824 9511      35      STA ZAHL+1,X
0826 CA        36      DEX
0827 10F9      37      BPL Z1
0829 60        38      RTS
082A          39  ;
          40      END

```

利用ADC1210來作A/D轉換

下面所要描述的是利用一個12位元(bit)的A/D轉換器，來作工業上的應用。這個A/D轉換器主要是用來測量一個慢速變化的電壓，由於是每秒測量一次，所以具有很大的準確性。正如前面所提到的，使用這個12位元的A/D轉換器將可以獲得比8位元A/D轉換器更好的解離度(resolution)和準確性(accuracy)。

整個應用電路的完整圖形如圖 28 所示。

如上圖所示，ADC 1210 的輸出是被連接到6522的各埠(proto)，在ADC1210的輸出中的較小位元是連接到埠A，且剩餘的4個較大位元是連接到埠B的PB0~PB3。至於PB4則是用來產生開始轉換脈衝(SC)，同時PB5是用以讀取轉換完成(CC)信號。在1210內部的A/D轉換形式，和前面提

到的8位元A/D轉換之方法一樣。ADC1210也可以利用連續趨近(successive approximation)方法，將類比信號轉換成數位形式，但此處是利用硬體來完成這個工作。ADC 1210 是使用一個外部時鐘(CLOCK)，它的頻率必須限制在60和70KHz之間。所以，我們將1MHz的時鐘脈波以除頻電路(4024)來加以除頻。因此能提供1210的輸入頻率為67.5KHz。1210的腳1至腳12輸出電位是在邏輯0(L)時是V₊，在邏輯1(H)時是0。且有一點要注意的是6522輸入端其電壓位準，不能超過TTL的電壓位準。因此電源供應電壓V₊是等於內部參考電壓+5.12V。這個電壓可由+12V的電源或其它可調式電源來獲得。而實際電壓則是由5000Ω電位器(P1)來調整。

由圖 28 可以知道，在A/D轉換器的腳19，可以有一個單極性的輸入電壓，此電壓變

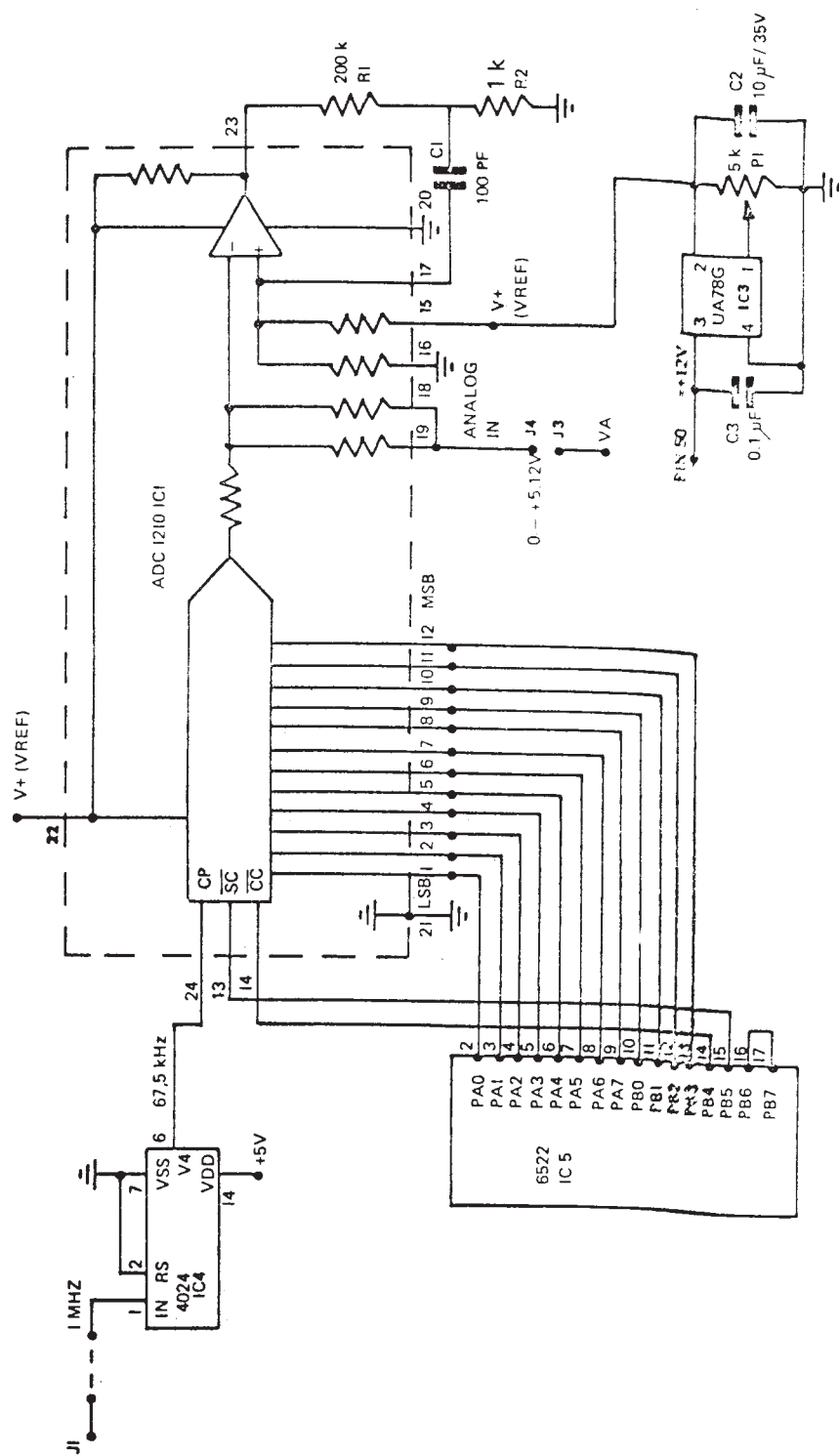


圖 28 ADC 1210 的電路連接情形

化是由 0~+5.10 V。有一點要注意的是，在多數的應用例子裏，感知器 (sensor) 將無法提供這個輸入電壓。若是要放大低電壓信號，

則可以利用電路板上的四個運算放大器 (OP)，來組成一個儀表放大器 (instrumentational amplifier)。

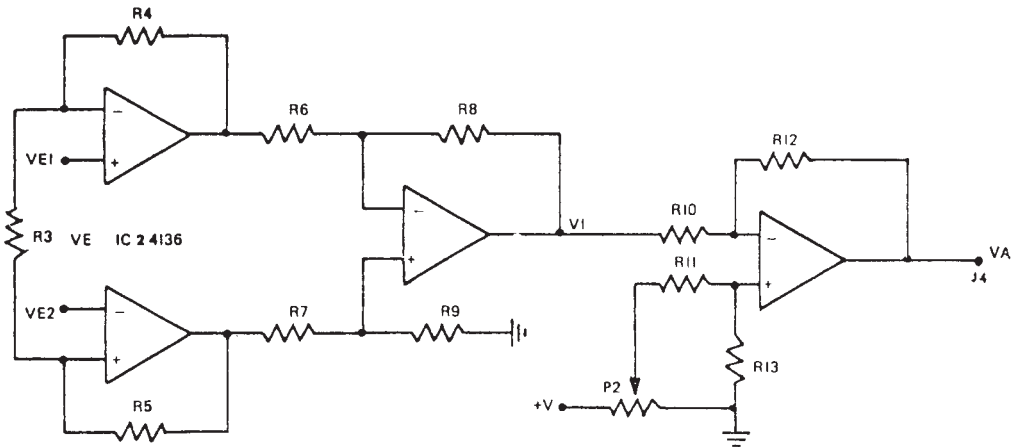


圖 29 儀表放大之電路組成

假如在上圖中，使 $R4=R5$ ，且 $R6=R7=R8=R9$ ，則增益因素是 $V=1+2 \times R4/R3$ ， $R4$ 和 $R6$ 值建議使用 $100K\Omega$ 之數值。若採用不同的輸入，則電壓 $V1=(1+2 \times R4/R3) \times (VE2-VE1)$ 。假如選擇 $R4=100K\Omega$ ， $R3=2K\Omega$ ，則電壓增益為 100。由圖上知第一級的放大器也是一個差動放大器。當 $R10=R11=R12$ 和 $R13$ 且其值為 $100,000\Omega$ 時，則電壓增益是 1。電位器 P 2 是用來調整圖 29 中輸出電壓 (VA) 之位準，即調整 A/D 轉換器輸入端之位準變化。

最後，由圖上可以看出 6522 VIA 的腳 PB6 和 PB7 是連接在一起。在腳 PB7 我們將利用計時器 1 產生一個週期為 0.1 sec 的方波。而計時器 2 則是作為一個計數器 (counter)。這個計數器是由程式設定為 10 且每十分之一秒減 1。當它的值為 0 時，即又是進行一個新測量時間。

現在就來討論一下程式，這個程式可分成兩部分。一部分是圖 30 所示的 BASIC 程式。另一個則是圖 31 所示的機械語言程式。

圖 30 A/D 轉換的 BASIC 輸入程式

```

10 REM *****
12 REM * ANALOG INPUT WITH THE *
14 REM * ADC 1210 . *
16 REM * RAGE 0 - 5.12 VOLTS *
18 REM * 1 MEASUREMENT/SECOND *
20 REM * STARTING WITH BUTTON 1*
30 REM *****
100 MSB = - 15105:LSB = - 15106
105 FIN = - 15107
106 DIM MW(500)
107 I = 0

```

```

110 INIT = - 15360:START = - 15248
      :MEASURE = - 15223:OFF = - 15200
112 CALL INIT
115 PRINT "START MEASUREMENT BY KEYPRESS"
120 CALL START: GOSUB 1000:
130 CALL MEASURE: GOSUB 1000
140 IF I > 3 THEN CALL OFF: GOSUB 1500
145 PRINT A
150 GOTO 130
160 END
1000 A = PEEK (MSB) * 256 + PEEK (LSB):
1010 A = A * 1.2942E - 03
1020 MW(I) = A:I = I + 1
1030 RETURN
1500 IF PEEK (FIN) > 127 THEN RETURN
1505 HGR : HCOLOR= 3
1506 HPLOT 1,159 TO 250,159
1507 Y = 25
1510 FOR K = 0 TO I - 1
1520 HPLOT K,159 - Y * MW(K)
1530 PRINT K;"      ";A
1540 NEXT K
1550 END

```

圖 31 機械語言形式

}CALL-151

*C400LLLLL

C400-	A9 20	LDA	#\$20	INIT
C402-	8D C2 C0	STA	\$C0C2	
C405-	8D C0 C0	STA	\$C0C0	
C408-	A9 E0	LDA	#\$E0	
C40A-	8D CB C0	STA	\$C0CB	
C40D-	A9 0A	LDA	#\$0A	
C40F-	8D C8 C0	STA	\$C0C8	
C412-	A9 00	LDA	#\$00	
C414-	8D C9 C0	STA	\$C0C9	
C417-	60	RTS		
C418-	EA	NOP		
C419-	EA	NOP		
C41A-	AD C0 C0	LDA	\$C0C0	
C41D-	29 10	AND	#\$10	
C41F-	D0 F9	BNE	\$C41A	
C421-	AD C0 C0	LDA	\$C0C0	
C424-	29 0F	AND	#\$0F	
C426-	49 0F	EOR	#\$0F	
C428-	8D FF C4	STA	\$C4FF	
C42B-	AD C1 C0	LDA	\$C0C1	
C42E-	49 FF	EOR	#\$FF	
C430-	8D FE C4	STA	\$C4FE	
C433-	60	RTS		
C434-	20 00 C4	JSR	\$C400	
C437-	20 70 C4	JSR	\$C470	

C43A-	AD FF C4	LDA	\$C4FF	
C43D-	20 DA FD	JSR	\$FDDA	
C440-	AD FE C4	LDA	\$C4FE	
C443-	20 DA FD	JSR	\$FDDA	
C446-	20 62 FC	JSR	\$FC62	Not Used
C449-	20 89 C4	JSR	\$C489	
C44C-	18	CLC		
C44D-	90 EB	BCC	\$C43A	
C44F-	20 62 FC	JSR	\$FC62	
C452-	CA	DEX		
C453-	D0 EB	BNE	\$C440	
C455-	4C 59 FF	JMP	\$FF59	
C458-	F7	???		
C459-	F7	???		
C45A-	F7	???		
C45B-	FF	???		
C45C-	7D FB FA	ADC	\$FAFB,X	
C45F-	FF	???		
C460-	A9 00	LDA	#\$00	
C462-	8D C0 C0	STA	\$C0C0	
C465-	A0 05	LDY	#\$05	
C467-	88	DEY		
C468-	D0 FD	BNE	\$C467	
C46A-	A9 20	LDA	#\$20	
C46C-	8D C0 C0	STA	\$C0C0	
C46F-	60	RTS		
C470-	AD 62 C0	LDA	\$C062	INIT
C473-	30 FB	BMI	\$C470	
C475-	A9 4E	LDA	#\$4E	
C477-	8D C4 C0	STA	\$C0C4	
C47A-	A9 C7	LDA	#\$C7	
C47C-	8D C5 C0	STA	\$C0C5	
C47F-	20 60 C4	JSR	\$C460	
C482-	20 1A C4	JSR	\$C41A	
C485-	60	RTS		
C486-	EA	NOP		
C487-	EA	NOP		
C488-	EA	NOP		
C489-	AD C8 C0	LDA	\$C0C8	MEASURE
C48C-	D0 FB	BNE	\$C489	
C48E-	A9 0A	LDA	#\$0A	
C490-	8D C8 C0	STA	\$C0C8	
C493-	A9 00	LDA	#\$00	
C495-	8D C9 C0	STA	\$C0C9	
C498-	20 60 C4	JSR	\$C460	
C49B-	20 1A C4	JSR	\$C41A	
C49E-	60	RTS		
C49F-	EA	NOP		
C4A0-	AD 62 C0	LDA	\$C062	OFF
C4A3-	8D FD C4	STA	\$C4FD	
C4A6-	60	RTS		
C4A7-	00	BRK		
C4A8-	8D CB C0	STA	\$C0CB	
C4AB-	60	RTS		

```

C4AC-   FF          ???
C4AD-   BF          ???
C4AE-   FB          ???
C4AF-   BF          ???
C4B0-   DD A4 FF    CMP

```

]CALL-151

*C400.C433

```

C400- A9 20 8D C2 C0 8D C0 C0
C408- A9 E0 8D CB C0 A9 0A 8D
C410- C8 C0 A9 00 8D C9 C0 60
C418- EA EA AD C0 C0 29 10 D0
C420- F9 AD C0 C0 29 0F 49 0F
C428- 8D FF C4 AD C1 C0 49 FF
C430- 8D FE C4 60

```

*

C460.C4A6

```

C460- A9 00 8D C0 C0 A0 05 88
C468- D0 FD A9 20 8D C0 C0 60
C470- AD 62 C0 30 FB A9 4E 8D
C478- C4 C0 A9 C7 8D C5 C0 20
C480- 60 C4 20 1A C4 60 EA EA
C488- EA AD C8 C0 D0 FB A9 0A
C490- 8D C8 C0 A9 00 8D C9 C0
C498- 20 60 C4 20 1A C4 60 EA
C4A0- AD 62 C0 8D FD C4 60

```

*

ADC1210 的資料是利用記憶位址 \$C4-FF(MSB) 及 \$C4FE(LSB) 來傳送至 BASIC 程式。另外一個記憶位址, \$C4FD 則是用來當作停止測量的一個旗號(flag)。

要開始測量, 是利用 Apple 主機板上 game I/O 的按鈕 P0 來控制。即按下按鈕時, 則計時器被設定且開始進行第一個測量。測量值是存在陣列 MW(I) 中, 也就是陣列將用以儲存所有的測量值, 在這個陣列之中, 測量值是一個接一個依序的儲存下去。若要求出所測量到的正確電壓值, 則必須將測量值乘以一個標度因素(scale factor): $A = V_{ref} / 4096$ ($= 0.00125$, 這是對 $V_{ref} = 5.12V$ 而言)。

在 BASIC 程式中儲存和計算這些測量值, 大約需 4/10 至 6/10 秒的時間。程式剩餘的時間是副程式 MEASURE 消耗掉。

爲了上面的應用, 輸入/輸出界面電路板 (I/O board) 被設計成印刷電路板的形式, 如圖 32 和 33 就是這個电路板的零件位置圖和零件表。

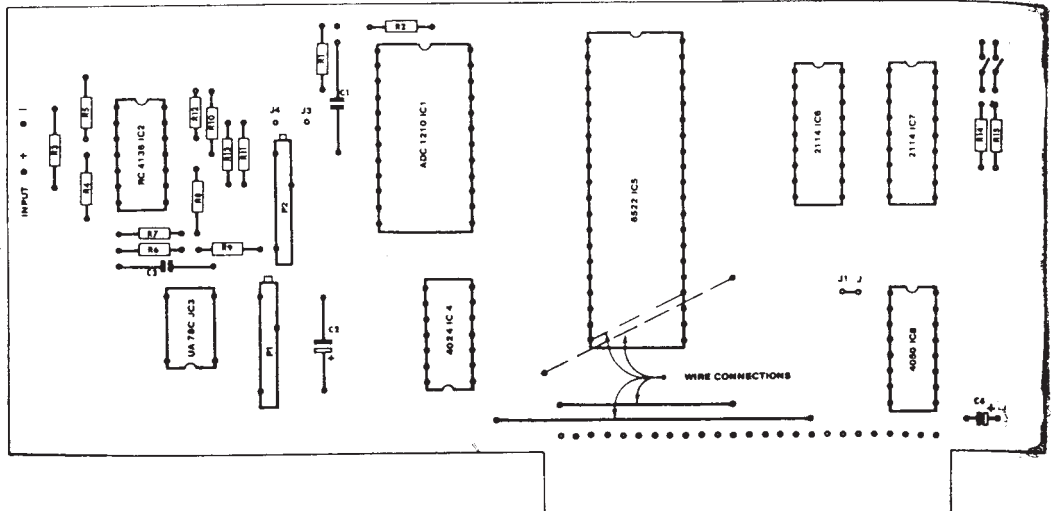
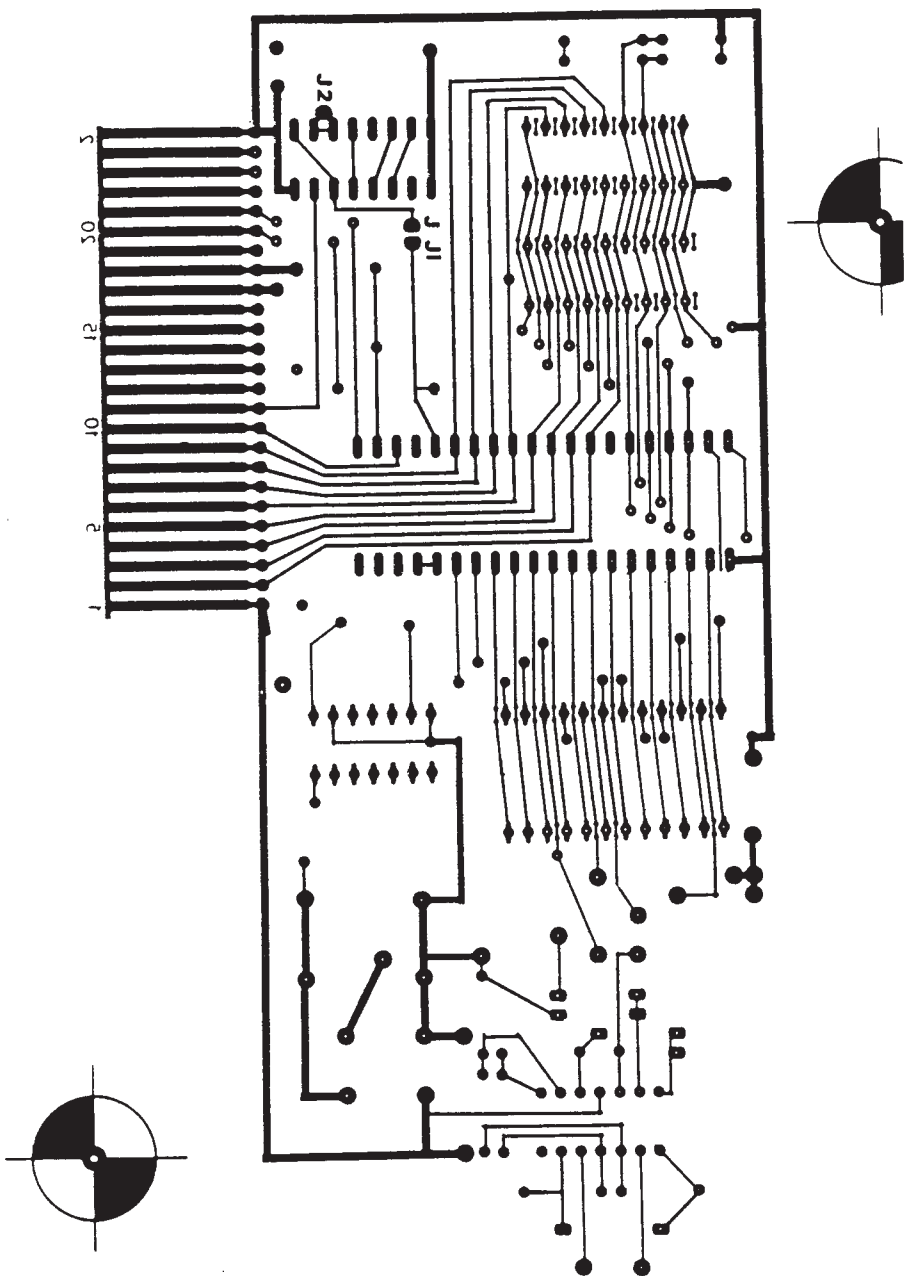
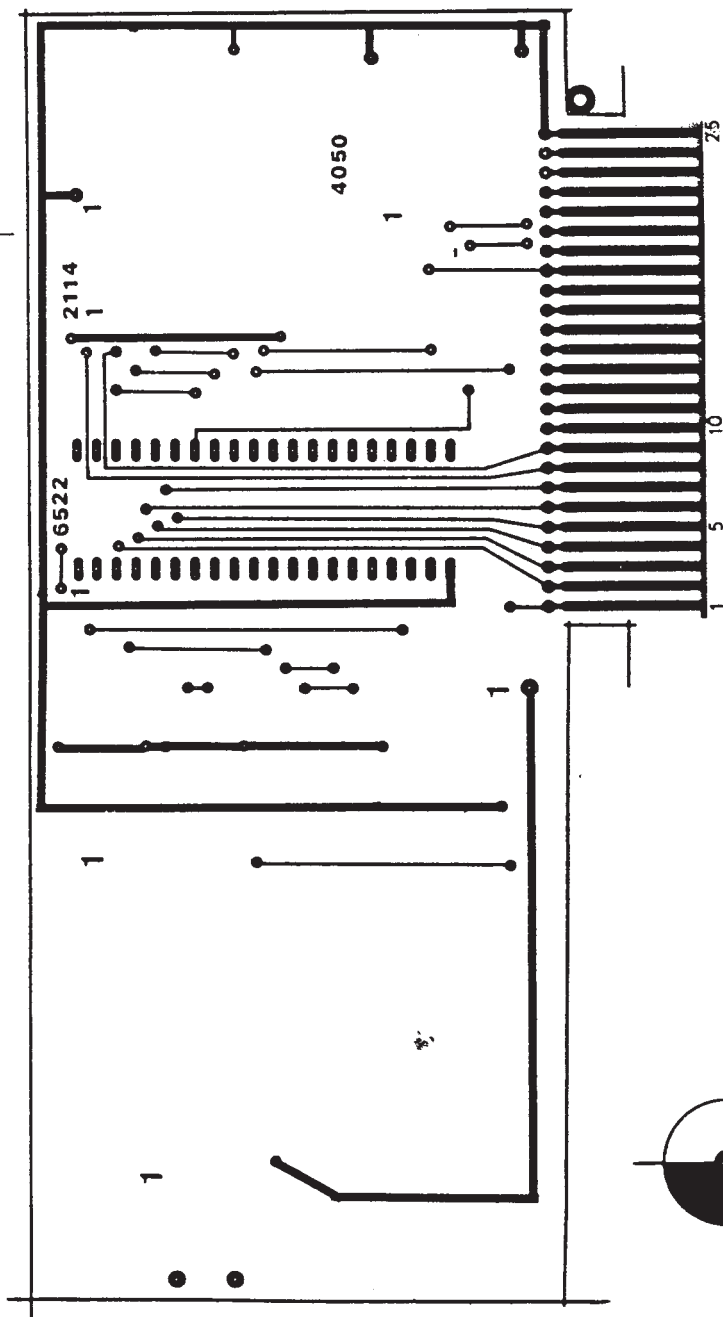


圖 32 ADC 1210 I/O 界面板零件位置圖





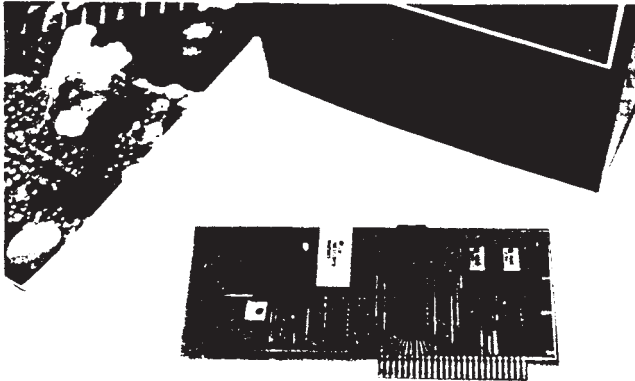
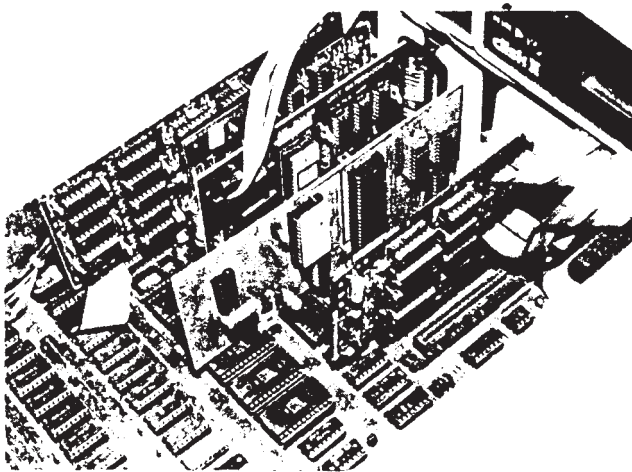


圖 33 零件表

R1	200k
R2	1k
R3 – R13	(See Text)
R14, R15	4.7k
C1	100 pF Ceramic
C2	10 μ F/35 V
C3	0.1 μ F
C4	10 μ F/35 V Tantal
P1, P2	5k Trimmer
IC1	ADC 1210 N.S.
IC2	RC 4136 T.I.
IC3	UA 78G Fairchild
IC4	4024
IC5	6522
IC6, IC7	2114
IC8	4050 Motorola
S	2 Pole Dual In-line Switch



EPROM/RAM界面板

本製作可先將RAM插入EPROM之插座位址，並將程式（機械語言）在RAM中測試無誤後，再將RAM取出且插入EPROM，然後將程式燒入。當然此處所用之RAM和EPROM必須具有共通性（Compatible）。

目前在台灣擁有APPLE II的人已不算少數，但利用它來作硬體實驗或用於控制方面的人却不多，主要原因是利用電腦來作硬體控制，大多必須配合程式來作，且為了使用方便，這些組合語言程式必須錄在EPROM中，但是正如大家所知道的，APPLE II在主機板上並沒有保留可以安置EPROM的空間，因此對於使用者而言，非常之不方便。基於這個原因，

在本文中我們將告訴你如何製作一個特殊的介面板，它可允許你將介面插入空的周邊插槽中，以讓你的APPLE II能用在控制方面。

本文所介紹這個介面板，可允許你插入4個EPROM或是和EPROM具共通性（Compatible）接觸的RAM。介面板的完整電路如下圖所示。

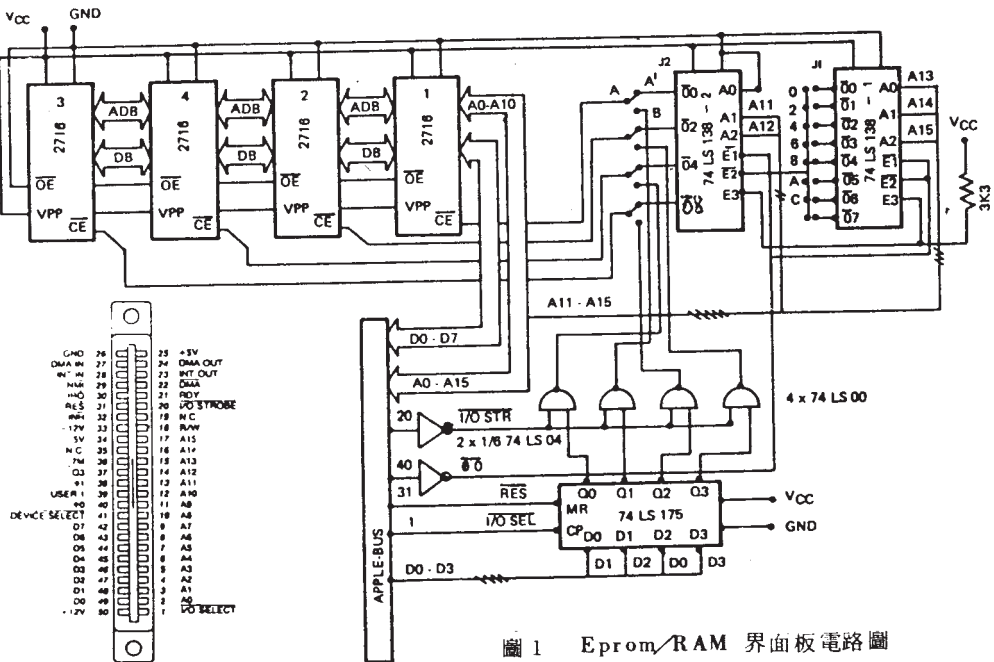


圖1 EPROM/RAM 界面板電路圖

記憶擴充觀念

BYTE WIDE 是由 Mostek 公司所推出的一種觀念，它允許你在介面板上使用相同的 24 腳插座，以便擴充 Eprom 或 RAM 記憶區的容量。但有一點要附帶一提的是，此處所提到的 RAM 是接腳和 Eprom 具共通性的 RAM，這些 RAM 有 1K 或 2K bytes 的形式可供選擇。

APPLE II 電腦在主機上總共具有 8 個保留的周邊擴充介面插座 (Interface slot)，其編號是 slot 0 至 7，slot 0 是作為記憶擴充連接用，例如，LANGUAGE CARD 或 ROM CARD。slot 1 是連接 printer card 用，slot 2 至 5 是作為使用者擴充用。slot 6 是用來連接 disk card (可接兩台磁碟機)

。slot 7 是使用在歐洲型的 PAL 或 SECAM 電視系統介面連接用。由這裏的說明，你可以明白的看出，在 APPLE II 上有 4 個空的介面插座可允許你自由使用。但是有一個很小的限制：即對每一個介面插座而言，你最多僅能選用 2K bytes 的記憶空間，且每個插座的定址範圍都限制在 \$C800-\$CFFF。也就是說在同一時間內，你僅能定址 2K 的 Eprom / RAM (接腳必須和 Eprom 具共通性)。雖然如此，但本文將介紹的 Eprom / RAM 介面板卻允許你使用 4 個具 2K bytes 的 Eprom 或 RAM。在這個製作中，我們特別發展出一個記憶庫切換電路 (bank-switching circuit)，以在同一時間內能定址到 4 個 Eprom 之一，且這個 Eprom 的位址範圍是在 \$C800-\$CFFF。

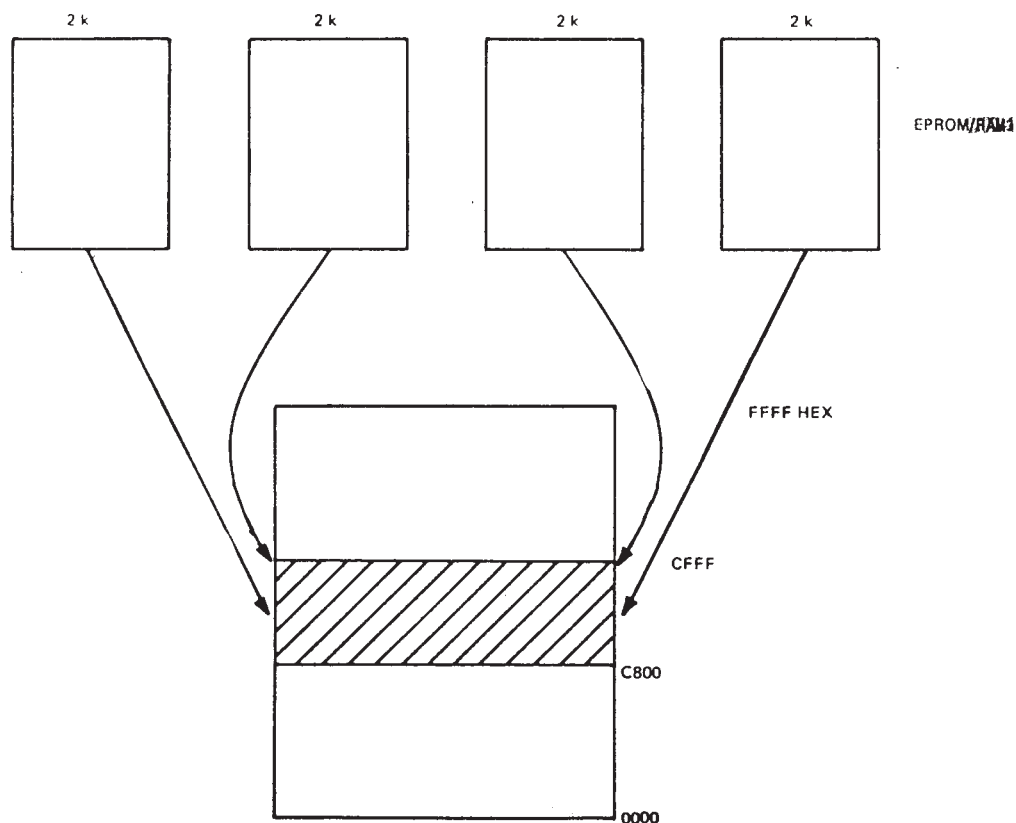


圖 2 記憶庫之位址分配情形

由上面的記憶圖可以看出 Eprom 的位址分配情形。且由圖上知道每個介面板僅能定址一個 2K byte 的 Eprom，位址是在 \$C800-\$CFFF。假如你使用 4 個 Eprom/RAM 介面板，且將它們插在 slot 2 至 5 的介面插座上，則可以使你的 APPLE 增加 16 顆 Eprom 的記憶容量，但是正如前面所提的，在同一時間內你僅能定址到一個 2K byte (\$C800-\$CFFF) 的 Eprom。前面曾提到介面上也可

使用和 Eprom 接腳具共通性的 2K RAM (4802)，另外也有一種價格較便宜的 1K RAM (4801) 可用。假如你是選用 RAM 4801，則你必須決定是要使用 \$C800-\$CFFF 的那一半記憶區。正如此處所提到的，假如你在介面上除了使用 Eprom 外，還要使用 RAM，則在線路上必須連接一條跳線，如下圖所示：

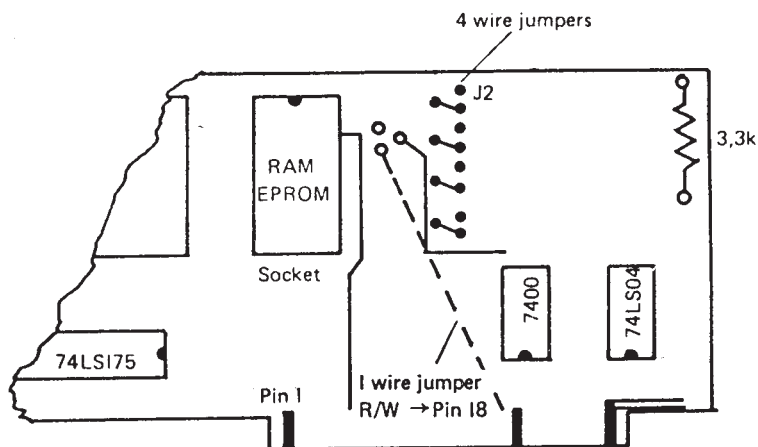


圖 3 零件佈線圖之跳線連接情形

圖 3 告訴你如何將零件安置在印刷電路板，而跳線是由周邊介面連接器的腳 18，連接到 Eprom/RAM 插座的腳 21。這條跳線主要是用以提供 RAM 記憶器讀／寫 (R/W) 信號，以使 CPU 對 RAM 進行讀／寫動作。對於 Eprom 而言，因為它僅能讀出而不能寫入，所以連接此跳線並不會影響 Eprom 操作。由這裏的說明可以知道，你可以在插座上使用 RAM 或 Eprom，而不用擔心跳線的連接，是否會影響到電路的操作。為了 Eprom/RAM 介面板能在 APPLE II 上正常操作，在介面板上 J2 的地方，還必須連接 4 條跳線 (參見圖 3)。

Eprom/RAM 介面板有另一個特點，就是它允許你在 RAM 中測試程式，然後才利用 Eprom 燒錄器將程式燒入 Eprom 中。

記憶切換電路及程式規劃

正如前面所說的，無論你在介面板上裝幾個 Eprom，在同一時間內你僅能定址 \$C800-\$CFFF 的 2K bytes 記憶空間，所以在這個介面製作中，發展出一個記憶庫切換電路，以便 CPU 能對 4 個 Eprom 中的一個，進行讀寫動作。即經由軟體的控制，你可以利用 74LS175 (四組 D 正反器) IC 來選取 4 個 Eprom 之一。例如，假設 Eprom/RAM 介面板是插在 slot 2，則你可以利用下述指令，選取介面板上的 Eprom 1：

```
LDA #02 及 STA $C200
```

假如你要將 Eprom/RAM 介面板插入 slot 4，且想選取介面板第三個插座上的

SURVEY over the most common
24 PIN MEMORY EPROMS & RAMS

DEVICE	TYPE	MANUF	SIZE	PIN 18	PIN 19	PIN 20	PIN 21
4801	RAM	MOSTEK	1Kx8	CS*	NC	OE*	WE*
4118	RAM	MOSTEK	1Kx8	CS*	L*	OE*	WE*
4008	RAM	TI	1Kx8	CS*	AR	OE*	WE*
2716	EPROM	INTEL	2Kx8	CS*/ PROG	+12V	A10	-5V
2516	EPROM	TI	2Kx8	CS*/ PROG	A10	OE*	Vpp
3636	PROM	INTEL	2Kx8	CS3	CS2	CS1*/ PROG	A10
4802/ 4016	RAM	MOST/TI	2Kx8	CS*	A10	OE*	WE*
58725	RAM	MITSU- BISHI	2Kx8	CS	I0	OE	WE

CS*/S* = Chip Select (Low)
 OE* = Output ENable (Low)
 WE* = Write Enable (Low)
 PD = Power Down
 PROG/(PE) = Program Enable
 Vpp = 25V (Program Voltage)
 L* = LATCH (LOW)

Bytewide EPROM•RAM•ROM

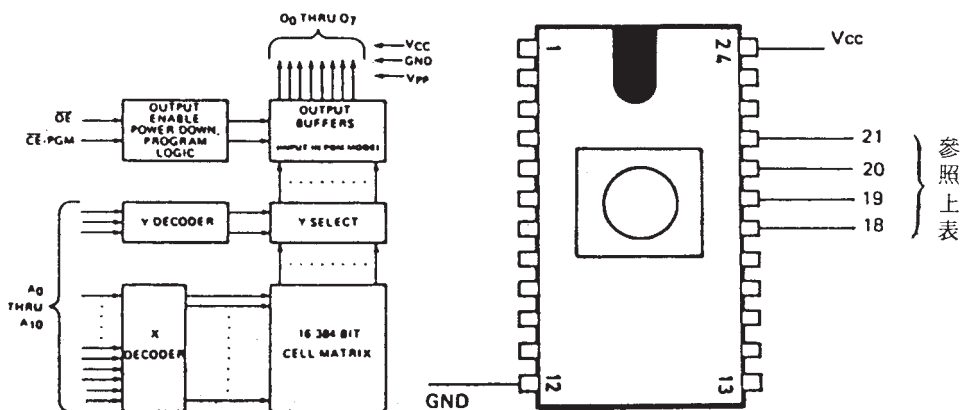
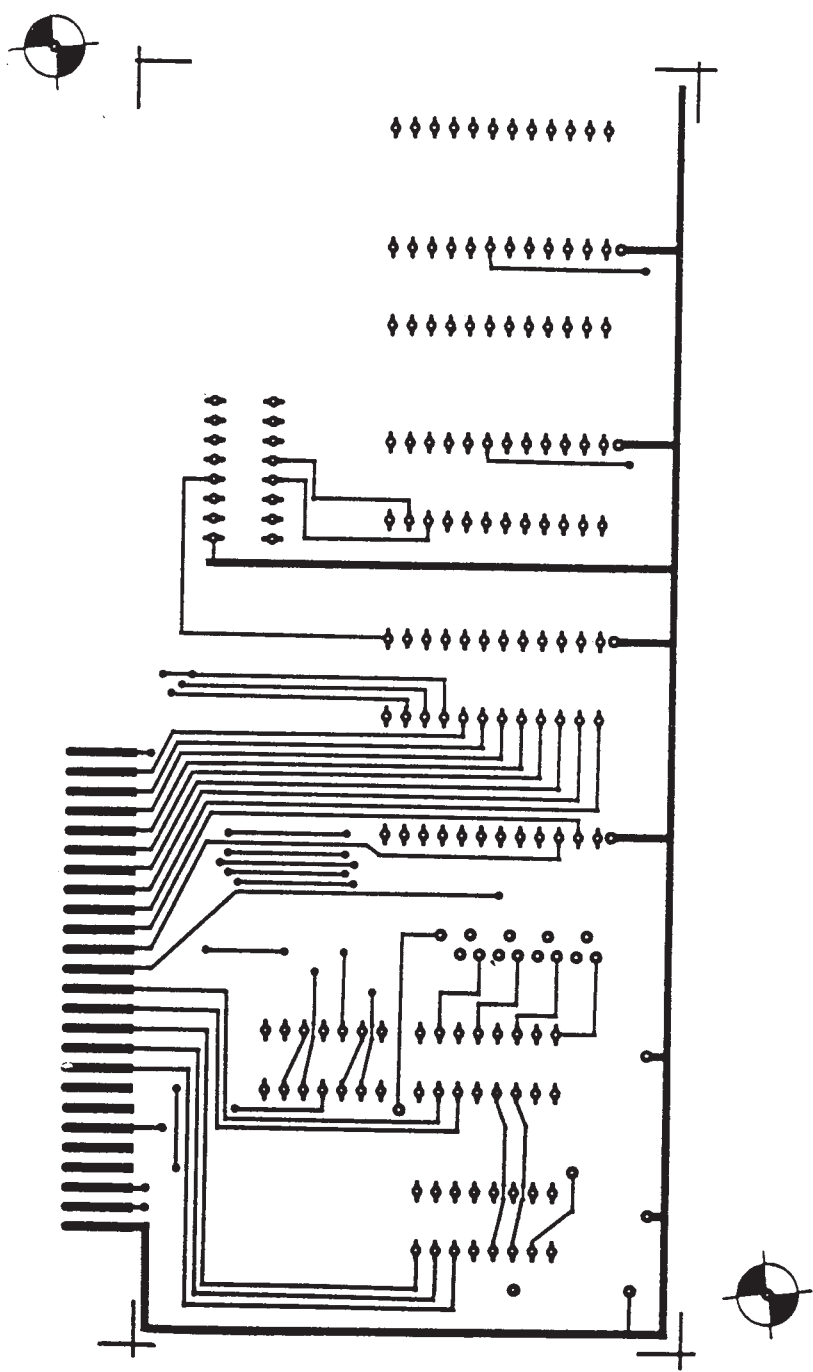
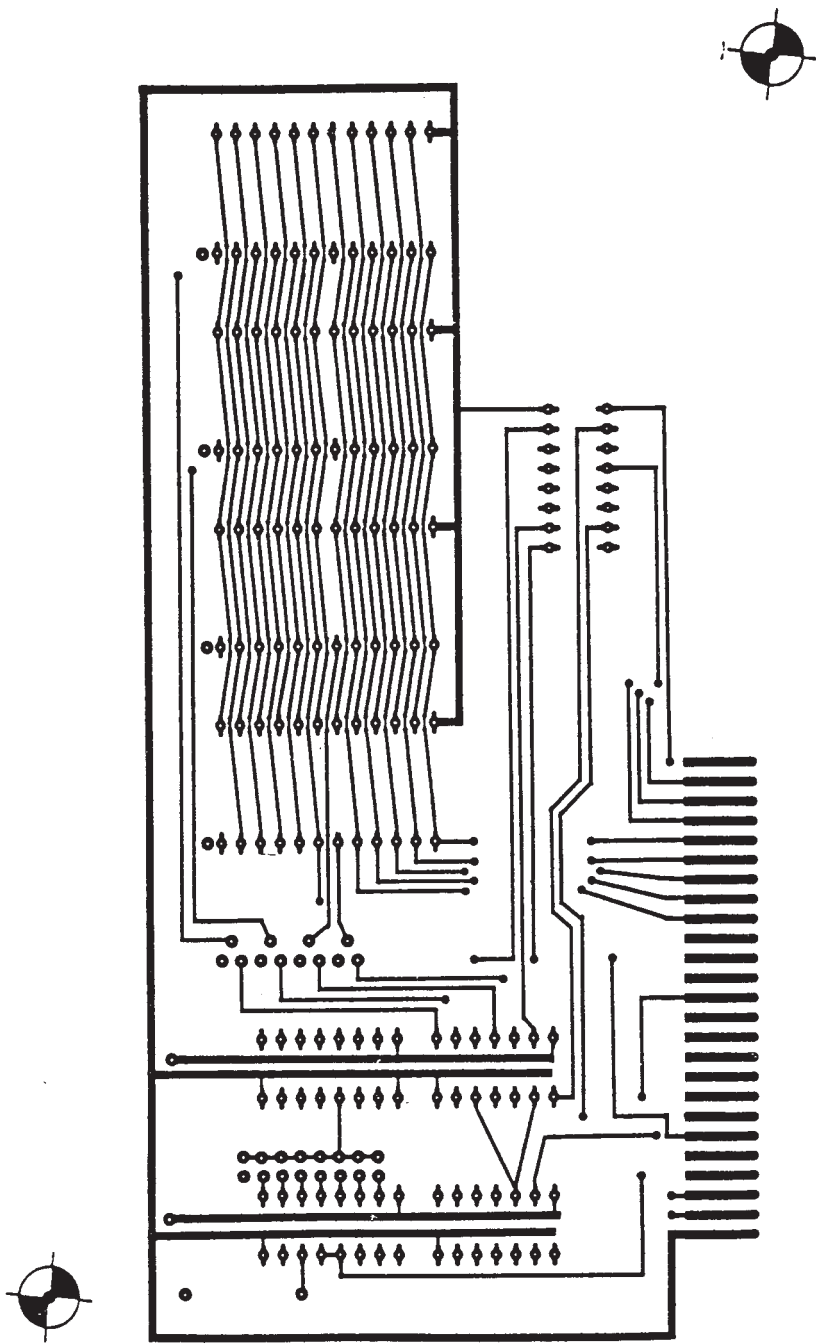


圖 4 Eprom 和具共通性接腳的RAM之特殊
接腳對照表



印刷線路板圖一零件面（頂面）



背面

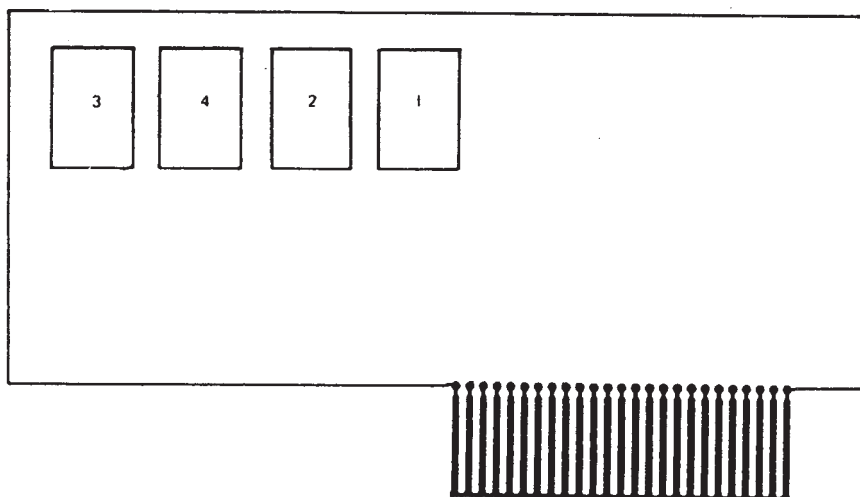


圖 5 介面板上插座之編號情形

Eprom，則可鍵入下列指令：

LDA #\$04 及 STA \$C400

爲了方便起見，特將各介面板上 Eprom

之選取指令，列表如下：

圖 6 Eprom選取指令表

SLOT 2	SLOT 3	SLOT 4	SLOT 5	SELECT
LDA #01 STA \$C200	LDA #01 STA \$C300	LDA #01 STA \$C400	LDA #01 STA \$C500	EPROM 1
LDA #02 STA \$C200	LDA #02 STA \$C300	LDA #02 STA \$C400	LDA #02 STA \$500	EPROM 2
LDA #04 STA \$C200	LDA #04 STA \$C300	LDA #04 STA \$C400	LDA #04 STA \$C500	EPROM 3
LDA #08 STA \$C200	LDA #08 STA \$C300	LDA #08 STA \$C400	LDA #08 STA \$C500	EPROM 4

由上表中可以很容易的找到要選取介面板上 Eprom/RAM 所需的指令。因爲可以允許有 4 片介面板在周邊介面插槽上，所以爲了避免同時存取兩個介面板，使資料匯流排發生問題的狀況，必須在存取另一個介面板之前，以指令 LDA #\$00 及 STA \$C200 將原先使用之介面板關掉，然後再利用指令進行另一介面板之存取。另外，若按下 RESET 鍵，也可以脫離介面板的工作模式，而回到 BASIC 中。

注意：假如你需要超過 2 K 的記憶擴充容量，則你可以寫一個管理程式，來使用 4 個介面板上 32K 的記憶容量。即利用這個管理程式，你可以先啓動某個介面板，然後再利用指令選取介面板上的某類 Eprom (\$C800-\$CFF F)，然後在工作完成後，程式之控制又轉回到管理程式中，而管理程式即可再對介面板上另一類 Eprom 進行資料之存取，依此方式來控制，你即可擁有額外 32K 之記憶擴充容量

界面板電路組裝

首先將所有的 I C 插座銲接到介面板上，然後連接圖 7 所示的 J 2 跳線以及 E prom/R A RAM 插座腳 21 和周邊介面連接器腳 18 間的跳線。在一切安裝無誤後，將 I C 按圖 3 所示方向插到介面板上，確記 I C 接腳方向不要插反，否則可能使 I C 燒毀。 ◆

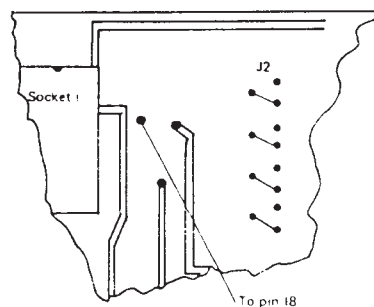


圖 7 介面板之跳線情形

製作一個 EPROM 燒錄器

EPROM 程式燒錄器

假如你是學電腦硬體 (hardware)，而且自己也寫了一些控制硬體 (電路) 的程式，那你一定會希望能將這些控制副程式，放到一個安全的地方，以使這些程式不致因電源中斷而喪失，或是因為外界程式的執行而破壞，要達到上述這些理想，就是要利用 EPROM WRITER (程式燒錄器) 來將程式燒入 EPROM 中。

另外，假如你希望進行硬體的發展或特殊應用的控制系統，那你也可以利用你的 APPLE 電腦，以及 EPROM 燒錄電路來創造你自己的微處理器以進行特殊的應用。

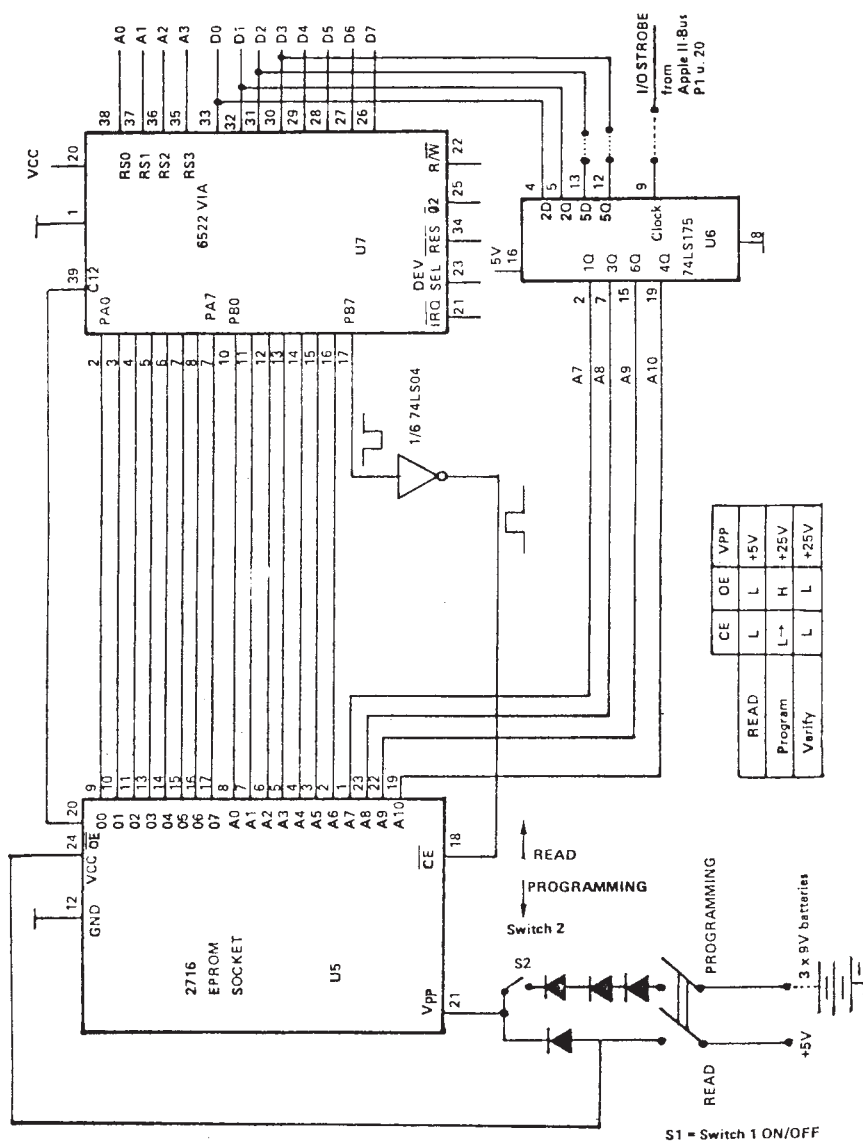
在本文中我們將告訴你，如何利用簡單的電路來組成 EPROM 燒錄器電路，以讓你的 APPLE 電腦在加上這個燒錄器介面電路後，能夠應用在硬體發展和系統控制應用方面。此處將使用一個 6522 I/O 介面板的電路，來製作 EPROM 燒錄器。在這個製作中，由於燒錄程式的執行是利用 APPLE 主機板上的 RAM，因此你並不需要額外的 RAM。另外，由於這個 EPROM 燒錄電路的結構和以往所介紹的 6522 I/O 板的結構有很大的不同，所以此處將另外仍介紹一個全新的電路，且完整的印刷線路也將在文章後面一併刊出，以作為你製作的參考。

正如 APPLE 其它的介面板一樣，本文所介紹的 EPROM WRITER 介面板，也可以插在 Slot 上，但由於在程式中是設定在 Slot #4 至 7 中的任一個 Slot，假如你欲將介面板插在別的 Slot 上，則必須對程式進行某些修改。

由於在將程式燒入 EPROM 時，必須有 +25V 的電壓加至 EPROM 的 V_{pp} 接腳上，因此在這個製作中，你必須提供 25V 的電壓供應，而這個 25V 的電壓供應，可以串聯 3 個 9V 電池或建立一個 25V 的直流電源供應電路。假如你是串聯 3 個 9V 電池則將得到 27V 的電壓供應，但因為 EPROM 在燒錄時僅需要 $25 \pm 0.5V$ 之電壓，因此必須再將 27V 電壓串聯 3 個或 4 個二極體，為了確定得到 $25V \pm 0.5V$ 之電壓，可以使用電表來測量，以決定所要串聯之二極體的數目，而為了讓你連接方便，所以我們在線路板上保留了 5 個二極體的位置，以供你自由取捨。

本文所介紹的 EPROM 燒錄電路，僅能燒錄具有 +5V 單電源供應的 EPROM 2716。EPROM 的燒錄工作主要是利用 6522 多用途周邊介面調適器 (VIA)，以及具有四個 D 正反器的 SN74LS175 IC 來完成。

為了能夠將資料 (程式) 輸入 EPROM 中



首先，必須由CPU取得這些資料，然後配合6522及74LS175所送出的位址、信號將資料寫入EPROM中。本文用來控制燒錄工作的軟體（程式），是假設燒錄器界面板是插在Slot #4，且經由軟體的控制，使得6522能

狗在適當的時間，將欲燒錄之資料經由 6522 之 PORT A 燒入 EPROM 中。為了能將資料正確的燒入 EPROM 的每個位址中，因此必須利用 6522 PORT B 較低次的 7 個位元 (b_0 — b_6) 以及 74LS175 的 4 個輸出來送出位址。

信號，以便能正確的將資料燒入具有 2 K 容量的 EPROM 中。在後面我們將會更詳細的討論 PORT B 的第 8 個位元 (b_7) 的工作情形。現在可以先將 b_7 視為一個位元，其主要工作是在燒錄期間，提供一個高狀態 (H) 的脈衝，經反相提供給 2716 的 CS 腳一個低狀態 (L) 的脈衝，以便能正確的將資料燒入 EPROM 2716 中。

正如上面所說的：由於 PORT B 僅能利用 7 個位元 (b_0-b_6) 來送出位址信號給 EPROM，因此剩餘的 4 個位元位址信號是由 74LS175 IC 來提供。參見圖 5 的 EOUT 副常式，先將較低次 8 個位元的位址信號存入 PORT B，以及 LACL 的記憶器位址中。而較高次的位址信號則被存入 LACH 之記憶器位址中。然後 LACL 暫存器被左旋一次，如此一來，LACL 的第 8 個位元 (b_7)，即被移入進位位元 C 中。然後將 LACH 左旋一次，則前述進位位元即被移入 LACH 暫存器之最低位元位置。接著下一指令 (STA STR) 即是用以產生一激發脈衝 (strube pulse)，以將位址信號的其餘 4 個高次位元送入 74LS175 門鎖器 IC 中。

現在我們已將位址信號的較低次 7 個位元 (A_0-A_6) 存入 PORT B (b_0-b_6) 中，且將較高次的 4 個位元 (A_7-A_{10}) 存入 74LS175 中，接著就是取得要燒錄的資料，而資料是利用儲存指令傳入 6522 的 PORT A 中。至此為止，位址信號 (A_0-A_{10}) 已可送到 EPROM 的位址輸入端，且資料也可由 6522 的 PORT A 取得。為了能將資料燒入 EPROM 中，必須送出一個具有特定週期 (近似於 50ms) 的脈衝，而此脈衝即是經由 PORT B 的 b_7 位元來送出。有一點要注意的是，在整個燒錄過程中，加至 EPROM 第 21 腳 (V_{pp}) 之電壓必須維持在 25V，以確保程式資料能穩定的燒入 EPROM 中。

為了在需要時能取得適當的電壓供應，在電路上使用了兩個開關 S1 和 S2。參見電路圖 1 下方的兩個開關。底下的開關 S2 是用以提供 +25V 的燒入電壓，而上面的開關 S1 則是

擔任保護，以便在你將 25V 的電壓加入 EPROM 時，不會將 +5V 的電源供應電壓移去。由圖上可以看到，①當兩個開關均向下撥時，將不產生任何作用。②當上面開關向上撥時，它可以提供 EPROM +5V 的電源供應電壓，以允許你進行讀或寫的動作。至於是讀或寫則由底下的開關來決定。(a)當底下開關向下撥時，代表此時僅允許你對 EPROM 進行讀的動作。(b)當底下開關向上撥時，即可提供 +25V 的電壓給 EPROM，也就是此時允許你對 EPROM 進行寫入動作。上面的開關是雙刀雙擲 (double-pole double-throw) 的形式，而底下的開關則是單刀雙擲 (single-pole double-throw) 的形式。

在圖 1 中所示的 CE 腳是晶片致能腳。為了能將資料由 EPROM 中讀出，則晶片選擇 (CE) 腳和輸出致能 (OE) 腳之電位必須為 "L"，當然此時還必須提供 +5V 的電源供應電壓給 EPROM。另外，在進行程式燒錄時，燒錄所需的 25V 供應電壓，是使 S1 和 S2 兩開關撥向上的位置來獲得。

EPROM 燒錄器的用法

在你按照本文介紹的方法 (後面將有詳細說明)，將 EPROM 燒錄器組裝完成後，你應該檢查接線、銲接是否正常，各 IC 是否有正確安裝，以確保電路能正常工作。在一切檢查無誤後，你可以將 EPROM 燒錄器置於 APPLE 電腦的 Slot #4。下一步驟就是確保兩個開關是撥向下的位置 (即關的位置)。在此情況下你即可以將 EPROM 插入 EPROM 插座中，而不必去理會 APPLE 電源是開或關，因為介面板的開關關掉時，已使 EPROM 插座和電腦的其餘部分隔離。在插入 EPROM 時必須確定 EPROM 的腳 1 是在上方，即和線路板上有標明 1 的位置一致 (EPROM IC 的缺口是朝向 6522)。在一切安置好後，假如 S2 (底下開關) 是撥向下且 S1 (上方開關) 是撥向上，則你可以將 EPROM 中的資料，讀入電腦的 RAM 記憶器中。假如是插入新的 EPROM，則程式

將會檢查 EPROM 是否完全空白……而這些工作都是程式在執行時自動來作，且程式也會告訴你其所檢查的 EPROM 是否完全空白，或是 EPROM 資料的燒錄是否成功等等。

軟體的執行

首先你必須鍵入 CALL-151，然後按下 CR 鍵以進入監督程式（*）。然後鍵入 800G CR 以執行程式，在程式執行後，你將會在螢幕上看到一些操作指示。即在你鍵入出現在螢幕上 3 個字中任一字的頭一字母後，將啟動程式執行對應的動作。假如你鍵入一個“R”字，則插於插座上之 EPROM 的整個資料內容，將被讀入 \$4000-\$47FF 的記憶位置中。假如你僅想讀取 EPROM 的某一部分位置的資料，則在開始時應該鍵入 803G 來執行程式。假如你決定如此作，則首先必須改變許多記憶位置的參數（圖 2 所示），以便指定程式所要讀取 EPROM 的記憶內含中之起始和終結位址。

下面所示之表格就是顯示你希望讀取

EPROM 之起始和結束位址，以及寫入 EPROM 之起始和結束位址之參數設定情形。

此處列出這個位址參數表之主要目的，就是允許你自己選擇設定，欲讀出 EPROM 的起始和結束位址，或寫入 EPROM 的起始和結束位址。若你沒有改變這些位址參數，則在程式中所設定的位址參數，是由 \$0000-\$07FF 的變化範圍。由於這些位址信號是經由 6522 VIA 和 74LS175 送出所以你必须藉著程式的執行使 6522 致能，以便能對 EPROM 進行正確的讀取動作。

例，假如你想讀取 EPROM 位址 \$05-\$15 中的資料，則必須將參數修改如下：SAEPL = \$05 且 SAEPH = \$00（表示欲讀取之 EPROM 起始位址），EAEPL = \$15 且 EAPH = \$00（表示欲讀取之 EPROM 結束位址）。

現在你必须決定由 EPROM 讀出來的資料要置於何處。假如你要他們存在 \$2005-\$2015 的 RAM 記憶區，則你必须設定 SAPL = \$05 且 SAPH = \$20。此處並不需要設定結束位址（EAPL 和 EAPH），因為程式之讀取動

10	SAEPL	Starting Address EPROM Low
11	SAEPH	Starting Address EPROM High
12	EAEPL	Ending Address EPROM Low
13	EAPH	Ending Address EPROM High
14	SAPL	Starting Address Program Low
15	SAPH	Starting Address Program High
16	EAPL	Ending Address Program Low
17	EAPH	Ending Address Program High
Default Values:		
	10 = 00	14 = 00
	11 = 00	15 = 40
	12 = FF	16 = FF
	13 = 07	17 = 47

圖 2 位址參數表

作在 \$0015 時即告結束。在將所有位址參數設定好後，你即可鍵入 * 803G **CR** 來執行程式。

測試 EPROM

爲了確保能完整的燒錄 EPROM，所以在燒錄之前最好能檢查測試一下，所欲燒錄之 EPROM 是否有完全空白或清除。也就是說，要先確定 EPROM 中的所有記憶位址之內含值爲 \$FF，才能開始進行程式的燒錄。而上面所說的檢查工作則是由程式來完成，假如程式發現 EPROM 中某個位置的內含值不是 \$FF，則它將會在螢幕上發出 EPROM NOT ERASED（EPROM 沒有清除）的訊息。假如你僅希望測試 EPROM 中的某一部份，則你必須依照前面所討論的方法，修改圖 2 中的位址參數，然後鍵入 * 803G **CR** 以執行程式。若程式發現 EPROM 已完全清除（每個位置之內含值都是 \$FF），則它會在螢幕上顯示出 EPROM ERASED（EPROM 已清除）的訊息。

EPROM 燒錄器電路板的組裝

假如你在程式執行後，由螢幕上所顯示之操作指示，鍵入 B 字母，則程式將會開始進行燒錄的程序。有一點要注意的是：在鍵入 B 以進行程式燒錄之前，你必須確定開關 S1 和 S2 是向上撥（ON）的位置，且由於將一個位元組（byte）的資料燒入 EPROM 的某個位置中，就需要 50ms 的時間，因此燒錄整個 EPROM 將需要大約 100sec 的時間。在每個位元組被燒入 EPROM 之後，程式將會自動檢查資料是否有正確燒入。假如程式發現在記憶器中的某位元組資料，與 EPROM 中讀出的資料不符合，則它會在螢幕上顯示 EPROM NOT PROGRAMMED（沒有燒錄成功）的訊息。假如一切進行正常，EPROM 中的每個位元組和記憶器中的每個位元組均相符合，則在燒錄完成後，螢幕上將會顯示 EPROM PROGRAMMED（燒錄完成）的訊息。正如前面所述鍵入 * 800G **CR**，將可燒錄整個 2K bytes 的 EPROM。

假如你僅要燒錄 EPROM 之某一部份，則可以和前面讀取部份 EPROM 的步驟一樣，只要修改圖 2 所示的位址參數即可。例子：假如你想將記憶位址 \$40GF 至 \$4137 中的資料，燒入 EPROM 以 \$187 爲起始位址的記憶器中。爲了達到這個目的，位址參數應該修改如下：SAPL = \$GF, SAPH = \$40, EAPL = \$37, EAPH = \$41, SAEPL = \$87 及 SAE-PH = \$01。此處並不需要設定結束位址（EAEPL 和 EAEPH）之值，因爲程式會自動在位址 \$4137 停止燒錄程序。

爲了使你更容易的利用 EPROM 燒錄器來進行程式的燒錄，特將本文所介紹 EPROM 燒錄器之燒錄操作程序總結如下：

1. 將 APPLE 電源關掉，然後將 EPROM 燒錄器插入 Slot 4。
2. 打開 APPLE 電源。
3. 將 EPROM 燒錄器之控制程式讀入電腦中，起始位址爲 \$800。
4. 檢查在 EPROM 燒錄器介面電路上的兩個開關 S1 和 S2 要撥到向下（OFF）的位置。
5. 將 EPROM 插入 EPROM 燒錄器之插座上，檢查 EPROM 腳 1 的位置和電路板上腳 1 的位置必須相符合。
6. 將欲拷貝至 EPROM 的程式讀入電腦中。
7. 將燒錄器上面的開關 S1 撥到向上（ON）的位置。
8. 鍵入 * 800G 以燒錄整個 2K 的 EPROM，或是修改圖 2 之位址參數然後鍵入 * 803G 以燒錄部分的 EPROM。
9. 正式燒錄之前，一定要得測試（test）EPROM，以便判別 EPROM 是否完全清除（空白）且準備程式燒入。
10. 將燒錄器下面的開關 S2 撥到向上（ON）的位置。
11. 鍵入 * 800G 或 * 803G 以開始程式燒錄。
12. 在完成燒錄工作後，先將開關 S1 和 S2 向下撥（OFF），然後取出 EPROM。

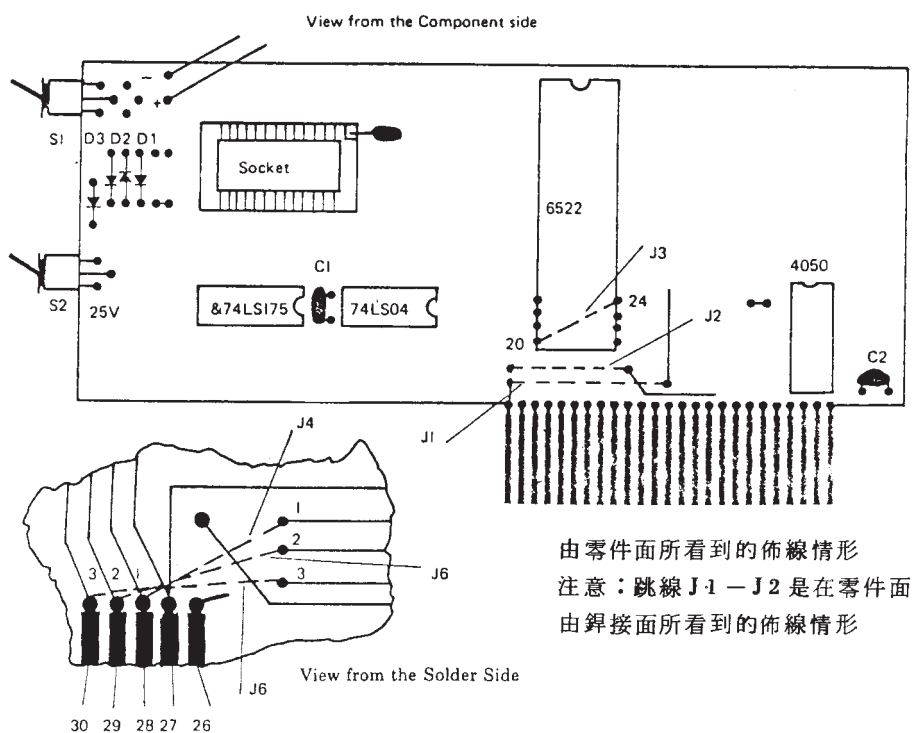


圖 3 零件佈線圖

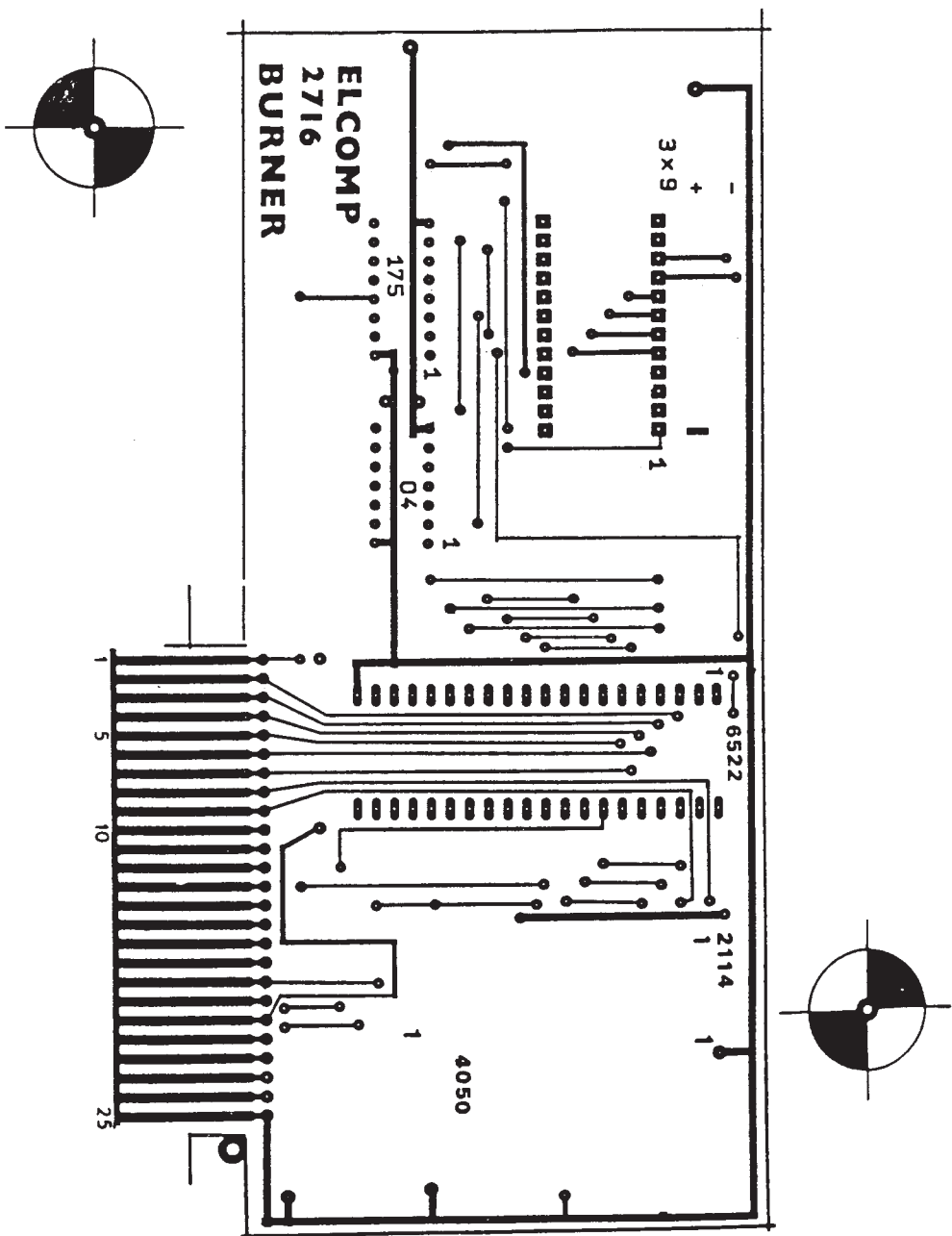
圖 4 EPROM 燒錄器之零件表

1	Capacitor tantal 10 μ F/35V
1	Capacitor 100 nF
1	DPDT-Switch
1	SPDT-Switch
1	Diode 2N 4148
1	14 pin socket DIL
1	16 pin socket DIL
1	18 pin socket DIL
1	40 pin socket DIL
1	24 pin socket TEXTOL
1	6522 (Rockwell)
1	4050 (Motorola)
1	74LS154
1	74LS04
1	PC-Board EPROM-BURNER
3	Diodes 2N 4148 see text

首先依照圖 3 線路所示，將各元件由低而高（指高度）一一銲裝上去。最後將 EPROM 24 PIN 插座銲接上去，且圖 3 所示之跳線也要記得銲接。

此外還需提供 25V 之電源供應，或是採用 3 個 9 V 電池串聯來提供。若是以 3 個 9 V 電池串聯來提供 25V 之 EPROM 燒入電壓，則必需如圖 3 線路所示，在左方串上 3 或 4 個二極體，以取得近似 25V 之電壓。

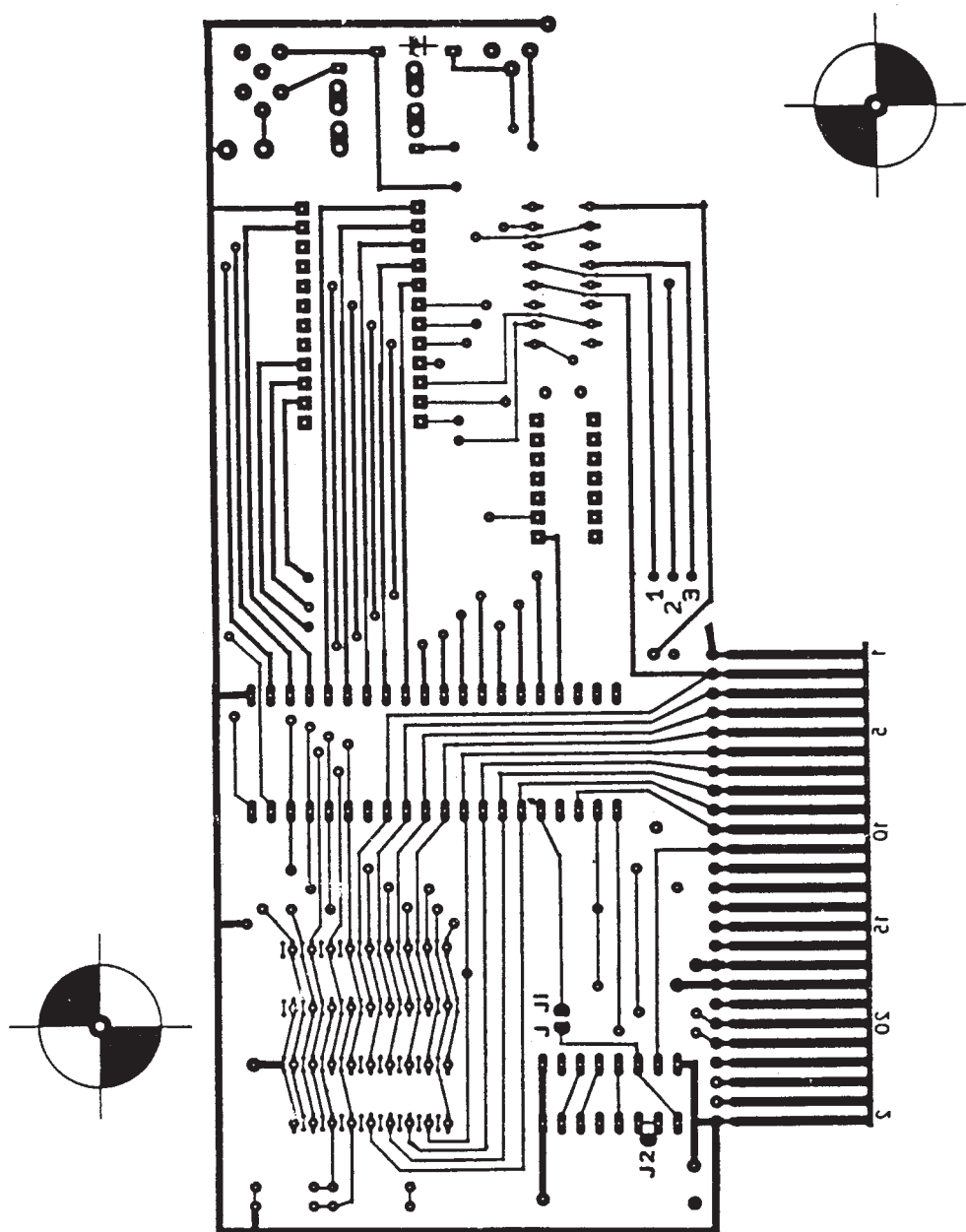
在將圖 3 線路上各元件安裝妥當後，接著就是安裝 S1 和 S2 兩個開關，但必須要檢查在上面的開關（S1）是雙刀雙擲的形式，而下面的開關（S2）是單刀雙擲的形式。S1 和 S2 開關在線路板上的接線情形，參見圖 3 所示。在 S1 和 S2 銲接完成後，就是將 25V 的電源，或



(Top)

上面

圖 6 印刷線路板圖



(Bottom)

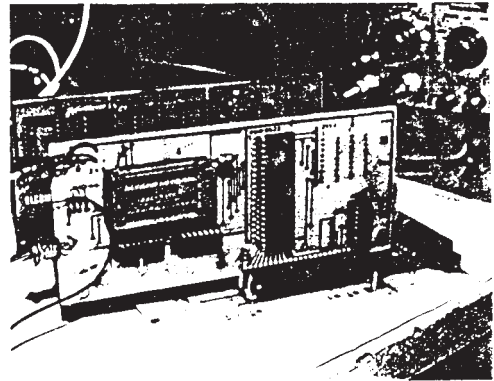
底面

圖 6 印刷線路板圖

3 個 9 V 電池串聯所提供的燒入電壓，連接至線路板上標有 + 和 - (參見圖 3 S1 上方) 的接點。另外在線路板正面，有二條跳線 J1 和 J2 要記得連接。其餘的跳線 J3, 4, 5 和 6 則是背面 (銲接面) 來連接。在連接這些跳線時，尤其要注意不要接錯。最後就是銲上電容 C1 和 C2 。

在上面組裝工作完成後，檢查所有接線、銲接點及元件的安置若沒有問題，即可將各 IC 插入插座上，小心 IC 方向不要插反。現在你已經擁有適用於你的 APPLE II 的 EPROM 燒錄器。接著就是看你如何充分有效的發揮它的功能了。

前面已將 EPROM 燒錄器之硬體線路，及元件組裝情形，作了詳細說明，但是除了上述



硬體 (電路) 結構外，還需有下述軟體 (程式) 才能進行程式的燒錄。現在將控制 EPROM 燒錄工作的程式列表說明如下：

圖 5 EPROM WRITER 程式

0800	1	DCM "PR#1"	；定義程式中各標題欄 (LABLE
C0C0	2	ORG \$C0C0	Field) 之等效記憶位址。
C0C0	3	TORB EQU *	
C0C0	4	TORA EQU *+!1	
C0C0	5	DDRB EQU *+!2	
C0C0	6	DDRA EQU *+!3	
C0C0	7	TlCL EQU *+!4	
C0C0	8	TlCH EQU *+!5	
C0C0	9	ACR EQU *+!11	
C0C0	10	PCR EQU *+!12	
C0C0	11	IFR EQU *+!13	
C0C0	12	;	
C0C0	13	;	
C0C0	14	STR EQU \$C800	
C0C0	15	COUT EQU \$FDED	
C0C0	16	RDCHAR EQU \$FD35	
C0C0	17	HOME EQU \$FC58	
C0C0	18	MONITO EQU \$FF59	
C0C0	19	;	
C0C0	20	SAEPL EPZ \$!0	；定義在零頁記憶位址之標題欄。
C0C0	21	SAEPH EPZ SAEPL+!1	
C0C0	22	EAEPH EPZ SAEPL+!2	
C0C0	23	EAEPH EPZ SAEPL+!3	
C0C0	24	SAPL EPZ SAEPL+!4	
C0C0	25	SAPH EPZ SAEPL+!5	
C0C0	26	EAPL EPZ SAEPL+!6	
C0C0	27	EAPH EPZ SAEPL+!7	

C0C0	28	LAL	EPZ	SAEPL+!8	
C0C0	29	LAH	EPZ	SAEPL+!9	
C0C0	30	LACL	EPZ	SAEPL+!10	
C0C0	31	LACH	EPZ	SAEPL+!11	
C0C0	32	HFZ	EPZ	SAEPL+!12	
C0C0	33				
C0C0	34				
0800	35		ORG	\$800	; 以\$0800為程式之起始記憶
0800 20C208	36	CSTART	JSR	DEFAU	位址
0803 201509	37	WSTART	JSR	INIT	; 冷啟動
0806 A246	38		LDX	#70	; 熱啟動
0808 203B08	39		JSR	TXTOUT	; 字元顯示常式
080B 2035FD	40		JSR	RDCHAR	
080E 8D4D08	41		STA	SAVEC	
0811 20EDFD	42		JSR	COUT	; 字元輸出程式
0814 AD4D08	43		LDA	SAVEC	; 判別是鍵入 R 或 B 或 T ?
0817 C9D2	44		CMP	"R"	
0819 D006	45		BNE	L1	
081B 205B09	46		JSR	LESEN	; 若是鍵入 R 則跳至 EPROM
081E 4C59FF	47		JMP	MONITO	資料讀取常式
0821 C9C2	48	L1	CMP	"B"	
0823 D006	49		BNE	L2	
0825 20CB09	50		JSR	PROGRA	; 若是鍵入 B 則跳至 EPROM
0828 4C59FF	51		JMP	MONITO	燒錄常式
082B C9D4	52	L2	CMP	"T"	
082D D0D4	53		BNE	WSTART	
082F 207709	54		JSR	PRUEFE	; 若是鍵入 T 則跳至 EPROM
0832 A265	55		LDX	#101	資料檢查常式
0834 203B08	56		JSR	TXTOUT	
0837 4C59FF	57		JMP	MONITO	
083A 00	58		BRK		
083B	59				
083B	60				
083B 8D4B08	61	TXTOUT	STA	SAVEA	; 字元顯示常式
083E BD4E08	62	TXTL	LDA	TEXT,X	
0841 F007	63		BEQ	FIN	
0843 20EDFD	64		JSR	COUT	
0846 E8	65		INX		
0847 18	66		CLC		
0848 90F4	67		BCC	TXTL	
084A 60	68	FIN	RTS		
084B 0000	69	SAVEA	HEX	0000	
084D 00	70	SAVEC	HEX	00	
084E	71				
084E	72				; 螢幕顯示字之等效 ASCII
084E 8D8D	73	TEXT	HEX	8D8D	、字元儲存區
0850 C5D0D2	74		ASC	"EPROM NOT EREASED	"
0853 CFCDA0					
0856 CECFD4					
0859 A0C5D2					
085C C5C1D3					
085F C5C4A0					
0862 A0A0A0					; \$00表示ASCII資料之
0865 008D	75		HEX	008D	結束，\$8D表示 CR
0867 C5D0D2	76		ASC	"EPROM NOT PROGRAMMED	"
086A CFCDA0					

086D	CECFD4		
0870	A0D0D2		
0873	CFC7D2		
0876	C1CDCD		
0879	C5C4A0		
087C	A0A0A0		
087F	008D	77	HEX 008D
0881	C5D0D2	78	ASC "EPROM PROGRAMMED "
0884	CFCDA0		
0887	D0D2CF		
088A	C7D2C1		
088D	CDCDC5		
0890	C4A0A0		
0893	008D	79	HEX 008D
0895	C2A9D5	80	ASC "B)URNING T)ESTING R)EADING "
0898	D2CEC9		
089B	CEC7A0		
089E	D4A9C5		
08A1	D3D4C9		
08A4	CEC7A0		
08A7	D2A9C5		
08AA	C1C4C9		
08AD	CEC7A0		
08B0	A0A0		
08B2	00	81	HEX 00
08B3	8D	82	HEX 8D
08B4	C5D0D2	83	ASC "EPROM EREASED"
08B7	CFCDA0		
08BA	C5D2C5		
08BD	C1D3C5		
08C0	C4		
08C1	00	84	HEX 00
08C2		85	;
08C2		86	;
08C2		87	;
08C2	A900	88	DEFAU LDA #\$00 ; 冷開機之位址參數之初值設定常式
08C4	8510	89	STA SAEPL
08C6	8511	90	STA SAEPL
08C8	8514	91	STA SAPL
08CA	A9FF	92	LDA #\$FF
08CC	8516	93	STA EAPL ; 設定EPROM之起始記憶位址 SAEPL
08CE	8512	94	STA EAEPL ; SAEPL 之值
08D0	A907	95	LDA #\$07 ; 設定EPROM之結束記憶位址 EAEPL
08D2	8513	96	STA EAEPL ; EAEPL 之值
08D4	A940	97	LDA #\$40 ; 設定RAM之起始記憶位址 SAPH
08D6	8515	98	STA SAPH ; SAPL 之值
08D8	A947	99	LDA #\$47 ; 設定RAM之結束記憶位址 EAPH
08DA	8517	100	STA EAPH ; EAPL 之值
08DC	60	101	RTS
08DD		102	;
08DD		103	;
08DD	A518	104	EOUT LDA LAL EPROM 定址常式
08DF	8DC0C0	105	STA TORB
08E2	851A	106	STA LACL
08E4	A519	107	LDA LAH ; 由PORT B 之 b ₀ -b ₆ 及 74LS175
08E6	851B	108	STA LACH 送出位址信號 , 以定址 EPROM 。

08E8	261A	109		ROL	LACL	
08EA	261B	110		ROL	LACH	
08EC	A51B	111		LDA	LACH	
08EE	8D00C8	112		STA	STR	
08F1	18	113		CLC		
08F2	60	114		RTS		
08F3		115				
08F3		116				
08F3	E618	117	NEXT	INC	LAL	；判別 EPROM 是否讀取或燒錄完畢之位址計算副常式
08F5	D002	118		BNE	N1	
08F7	E619	119		INC	LAH	
08F9	E610	120	N1	INC	SAEPL	
08FB	D002	121		BNE	N2	
08FD	E611	122		INC	SAEPH	
08FF	A511	123	N2	LDA	SAEPH	
0901	C513	124		CMP	EAEPL	
0903	900C	125		BCC	N3	
0905	F002	126		BEQ	N4	
0907	B00B	127		BCS	N5	
0909	A510	128	N4	LDA	SAEPL	
090B	C512	129		CMP	EAEPL	
090D	F002	130		BEQ	N3	
090F	B003	131		BCS	N5	
0911	20DD08	132	N3	JSR	EOUT	
0914	60	133	N5	RTS		
0915		134				
0915		135				
0915	2058FC	136	INIT	JSR	HOME	；初值化常式
0918	A900	137		LDA	#\$00	；設 PROT A 為輸入
091A	8DC3C0	138		STA	DDRA	
091D	AA	139		TAX		
091E	A8	140		TAY		
091F	A97F	141		LDA	#\$7F	；設 PORT B 之 b_0 - b_6 為輸出
0921	8DC2C0	142		STA	DDRB	
0924	A980	143		LDA	#\$80	；將 PORT B 之 b_7 設定為單穩觸發型態
0926	8DCBC0	144		STA	ACR	；PB7 MONOFLOP
0929	60	145		RTS		
092A		146				
092A		147				
092A	A510	148	START	LDA	SAEPL	；開始 EPROM 資料之讀取或燒錄
092C	8518	149		STA	LAL	
092E	851A	150		STA	LACL	
0930	3DC0C0	151		STA	TORB	
0933	A511	152		LDA	SAEPH	；將欲讀取或燒錄資料之 EPROM 位址信號，經由 PORT B 及 74LS175 送出
0935	8519	153		STA	LAH	
0937	851B	154		STA	LACH	
0939	261A	155		ROL	LACL	
093B	261B	156		ROL	LACH	
093D	A51B	157		LDA	LACH	
093F	8D00C8	158		STA	STR	
0942	60	159		RTS		
0943		160				
0943		161				
0943	C901	162	ERROR	CMP	#\$01	；錯誤處理常式
0945	D008	163		BNE	E1	
0947	A200	164		LDX	#\$00	

0949	203B08	165		JSR	TXTOUT	
094C	4C59FF	166		JMP	MONITO	
094F	C902	167	E1	CMP	#\$02	
0951	D005	168		BNE	E2	
0953	A218	169		LDX	#24	
0955	203B08	170		JSR	TXTOUT	
0958	4C59FF	171	E2	JMP	MONITO	
095B		172				
095B	201509	173	LESEN	JSR	INIT	; EPROM 資料讀取常式
095E	A90C	174		LDA	#\$0C	; OE="L" 使 EPROM 輸出致能
0960	8DCCC0	175		STA	PCR	
0963	202A09	176		JSR	START	
0966	ADC1C0	177	LES1	LDA	TORA	
0969	9114	178		STA	(SAPL),Y	
096B	E614	179		INC	SAPL	
096D	D002	180		BNE	LES2	
096F	E615	181		INC	SAPH	
0971	20F308	182	LES2	JSR	NEXT	
0974	90F0	183		BCC	LES1	
0976	60	184		RTS		
0977		185				
0977	201509	186	PRUEFE	JSR	INIT	; EPROM 資料檢查常式
097A	A90C	187		LDA	#\$0C	; OE="L" 使 EPROM 輸出致能
097C	8DCCC0	188		STA	PCR	
097F	202A09	189		JSR	START	
0982	ADC1C0	190	P1	LDA	TORA	
0985	C9FF	191		CMP	#\$FF	
0987	F005	192		BEQ	P2	
0989	A901	193		LDA	#\$01	
098B	4C4309	194		JMP	ERROR	
098E	20F308	195	P2	JSR	NEXT	
0991	90EF	196		BCC	P1	
0993	60	197		RTS		
0994		198				
0994	A90E	199	MONOFL	LDA	#\$0E	; EPROM 燒入脈衝供應常式
0996	8DCCC0	200		STA	PCR	; OE="H" 使 EPROM 寫入致能
0999	A950	201		LDA	#\$50	
099B	8DC4C0	202		STA	TlCL	
099E	A9C3	203		LDA	#\$C3	
09A0	8DC5C0	204		STA	TlCH	
09A3	ADCDC0	205	MO1	LDA	IFR	
09A6	2940	206		AND	#\$40	
09A8	F0F9	207		BEQ	MO1	
09AA	A90C	208		LDA	#\$0C	; OE="L" 使 EPROM 寫入失效
09AC	8DCCC0	209		STA	PCR	
09AF	60	210		RTS		
09B0		211				
09B0		212				
09B0	A200	213	CHANGE	LDX	#\$00	; EPROM 和 RAM 之位址參數置換常式
09B2	A004	214		LDY	#\$04	
09B4	B510	215	CA1	LDA	\$0010,X	
09B6	851C	216		STA	HFZ	
09B8	B91000	217		LDA	\$0010,Y	
09BB	9510	218		STA	\$0010,X	
09BD	A51C	219		LDA	HFZ	
09BF	991000	220		STA	\$0010,Y	

```

09C2 C8      221      INY
09C3 E8      222      INX
09C4 E004    223      CPX #$04
09C6 D0EC    224      BNE CA1
09C8 A000    225      LDY #$00
09CA 60      226      RTS
09CB         227      ;
09CB         228      ;
09CB 202A09  229      PROGRA JSR START      ; 開始 EPROM 程式燒錄
09CE 20B009  230      JSR CHANGE
09D1 A9FF    231      PR1   LDA #$FF      ; 設定 PORT A 為輸出
09D3 8DC3C0  232      STA DDRA
09D6 B110    233      LDA (SAEPL),Y
09D8 8DC1C0  234      STA TORA
09DB AA      235      TAX
09DC 209409  236      JSR MONOFL      ; 提供 燒入 脈衝
09DF A900    237      LDA #$00      ; 設定 PORT A 為輸入
09E1 8DC3C0  238      STA DDRA
09E4 8A      239      TXA
09E5 CDC1C0  240      CMP TORA
09E8 F00B    241      BEQ PR3
09EA A902    242      LDA #$02      ; 燒錄之資料和送入 EPROM 之
                                ; 資料不符，即燒錄錯誤
09EC 4C4309  243      JMP ERROR
09EF E610    244      PR2   INC SAEPL
09F1 D002    245      BNE PR3
09F3 E611    246      INC SAEPL
09F5 20F308  247      PR3   JSR NEXT
09F8 90D7    248      BCC PR1
09FA A232    249      LDX #50
09FC 4C3B08  250      JMP TXTOUT
09FF 60      251      RTS
0A00         252      ;
                                253      END

```

```

*****
*                                     *
*  SYMBOL TABLE -- V 1.5  *
*                                     *
*****

```

LABEL. LOC. LABEL. LOC. LABEL. LOC.

** ZERO PAGE VARIABLES:

SAEPL	0010	SAEPH	0011	EAEPL	0012	EAEPH	0013	SAPL	0014	SAPH	0015
EAPL	0016	EAPH	0017	LAL	0018	LAH	0019	LACL	001A	LACH	001B
HFZ	001C										

** ABSOLUTE VARIABLES/LABELS

TORB	C0C0	TORA	C0C1	DDRB	C0C2	DDRA	C0C3	T1CL	C0C4		
T1CH	C0C5	ACR	C0CB	PCR	C0CC	IFR	C0CD	STR	C800	COUT	FEDE
RDCHAR	FD35	HOME	FC58	MONITO	FF59	CSTART	0800	WSTART	0803	L1	0821
L2	082B	TXTOUT	083B	TXT1	083E	FIN	084A	SAVEA	084B	SAVEC	084D
TEXT	084E	DEFAU	08C2	EOUT	08DD	NEXT	08F3	N1	08F9	N2	08FE
N4	0909	N3	0911	N5	0914	INIT	0915	START	092A	ERROR	0943
E1	094F	E2	0958	LESEN	095B	LES1	0966	LES2	0971	PRUEFE	0977
P1	0982	P2	098E	MONOFL	0994	M01	09A3	CHANGE	09B0	CA1	09B4
PROGRA	09CB	PR1	09D1	PR2	09EF	PR3	09F5				

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:0212

EPROM燒錄器 電路分析

- ※適用於 APPLE II 微電腦
- ※能燒錄 2516 , 2716 , 2732 , 2764
- ※完全由軟體控制，不需任何切換開關

在此，我們來共同研究 EPROM WRITER (EPROM 燒錄器)，雖然以前在許多雜誌上都已研討過，而且市面上也有出售，但如果您使用過的話，就知道有許多不便之處，它需要動手來搬動切換開關，若稍不注意，可能會把您可愛的 EPROM 毀於瞬間，而且最近即將大量被採用的 2764 無法燒錄，有見於此，我們將 APPLE 專用 EPROM 燒錄器，重新設計、改良，將那些手動切換開關交由電腦去做，而且在燒錄 2764 時能夠勝任愉快，甚至將來 27128 普遍使用時，也能依其基本設計擴充完成。

我們先從 APPLE 的介面着手起，因為我們需要從 EPROM 讀取資料，或由 APPLE 送出資料給 EPROM，因此需要一顆介面元件 (Peripheral Interface Adapter PIA)，我們選用 MC 6821，有關 MC 6821 詳細資料的取得並不困難，因此我們只討論一些簡單的控制線路及控制程式。

由表一可知道 RS1 及 RS0 是這部份的主角，這部份的工作是否能完成，就要看這兩條的控制線路是否使用得當了。再配合圖一的線路來討論，比較容易了解其輸出或讀入的控制

情形。若將圖一的線路接在 APPLE 主機板的第一個 Solt 上，可以以下例的程式來完成輸出或讀入的工作。

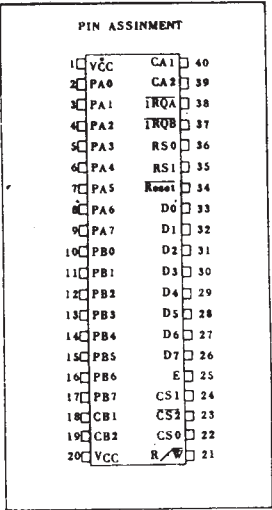
輸出：

```
LDA #00      ; CRA2 = 0
STA $C091     ; 選用方向暫存器 A
LDA #FF
STA $C090     ; 把 Port A 的方向全部定
               為輸出
LDA #04      ; CRA2 = 1
STA $C091     ; 選用週邊暫存器 A
LDA #0A      ; 資料為 $0A
STA $C090     ; 將資料輸出到 Port A 上
```

讀入：

```
LDA #00
STA $C091     ; 選用方向暫存器 A
STA $C090     ; 把 Port A 的方向全部定
               為輸入
LDA #04      ; CRA2 = 1
STA $C091     ; 選用週邊暫存器 A
LDA $C090     ; 將資料讀入
```

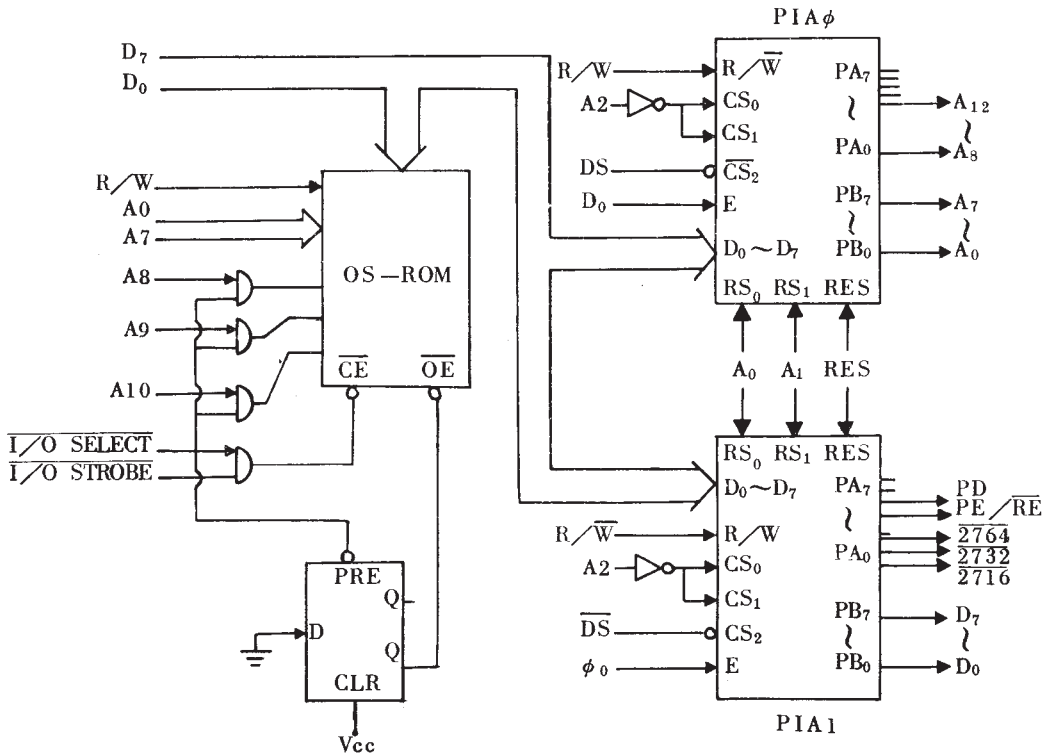
在圖二的線路中使用二顆 MC 6821，稍後您將會了解若使用 8255 三個 Port 的控制線是



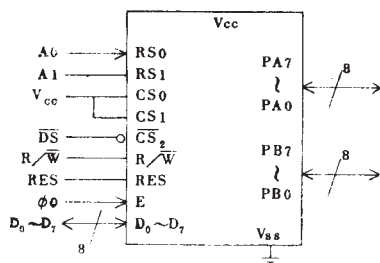
RS1	RS0	CRA2	CRB2	所 選 位 置
○	○	1	×	週邊暫存器 A
○	○	○	×	方向控制暫存器 A
○	1	×	×	控制暫存器 A
1	○	×	1	週邊暫存器 B
1	○	×	○	方向暫存器 B
1	1	×	×	控制暫存器 B

* CRA2 (CRB2) 是控制暫存器 A (A B) 的第二個 bit 。

表一 MC 6821



圖二 OS-ROM 及 MC 6821 控制線路



圖一 簡單MC 6821 控制線路

不夠的，因地址線需 13 條（A0—A12），資料線需八條，及一些控制線，即已超過 8255 所能供應的 24 條控制線，而且爲了以後設計擴充 27128 控制線路及軟體的方便，使用兩顆 MC 6821，以一勞永逸。

接着討論的是，此 EPROM 燒錄器的首腦，即儲控制程式的 OS—ROM（見圖二）。此 OS—ROM 的控制線路由四個 AND GATE 及一個 D—FF 完成，由這控制線路將每一個 Slot 留給介面卡用的 256 byte 及 2K 的 I/O ROM 解碼到一顆 OS—ROM 上，在分析此線路前，先有個前題必須注意，就是此 EPROM 燒錄器，除了 Slot 0 不能使用之外，在其他的任何一個 Slot 都能使用，因此必須有個軟體開關來啟動 OS—ROM，這個軟體開關就是圖二中的 D—FF，唯有使用者動用到該 Slot 256 byte 的地址（\$Cn00—CnFF，n 是 Slot 的號數）才能啟動執行 OS—ROM 内的控制程式。至於 A8，A9，A10，與 $\overline{\text{IOSELECT}}$ AND 的目的，是將 \$Cn00—\$CnFF 的地址解碼到 OS—ROM 上 \$0000—\$00FF 的位置。因此，使用者在 APPLE 開機後，而未使用 EPROM 燒錄器之前，並不因此燒錄器的存在而影響整個系統的工作，唯有在 APPLE 執行 PR #n（n 是 Slot 的號數）的命令後才開始工作。當然，這需要軟體配合才能完成這啟動的工作。因此，在控制程式的開始，需要一段啟動程式。APPLE 在執行 PR #n 的命令後，將 \$Cn00 存入一對 Memory 的位置。即 Cn 存入 \$37 的地址，00 存入 \$36 的地址。經過下

例啟動程式處理後，整個控制程式就能認知此燒錄器所在 Slot 的位置：

啓始程式：

```
LDA $37
ASL
ASL
ASL
ASL      ; 找出 Slot 數的參數 $no
STA $00   ; 將參數存入地址 $00 備用

LDA #F0
STA $36

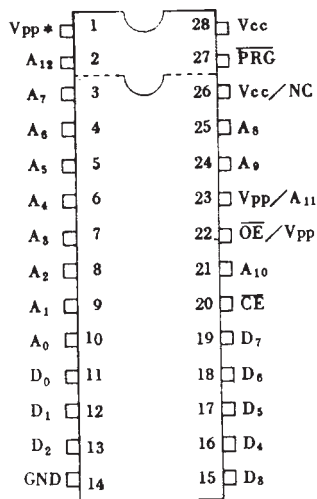
LDA #FD   ; 將監督程式所使用的參數歸還
STA $37   ; 給監督程式
JMP MAIN PROGRAM
```

在以後控制程式中若要使用 I/O 的位置時，不論是 Slot 1~7 中的任何一個 Slot，只需利用地址 \$00 所存的參數即可完成。下例就是一個 I/O 指令的例子：

```
LDX $00   ; 取參數
STA $C080,X; 輸出一個資料
```

接下來我們研究一下 APPLE 留給使用者使用者 2K I/O ROM，即 \$C800—\$CFFF 的位置，在圖二中 OS—ROM 的控制線路會發現 \$C800—\$C8FF 與 \$Cn00—\$CnFF 所解碼的位置是同一個位置，請勿認爲這裡有些矛盾，這樣反而使我們在設計控制程式時方便許多，因我們不需在意那一部份的控制程式要寫在 \$Cn00—\$CnFF 或寫在 \$C800—\$CFFF 的位置，只要從 \$C800 的位置一路寫下來就對了。到此爲止，設計硬體的工作可算是告一段落，建議您先將此部份線路測試一番，可能的話將上述所測的啟動程式燒入 OS—ROM（2716）裡，再配合 I/O 程式，執行幾次不同的情況，測試此部份線路是否正常，若有問題，請勿繼續往下做。

下一步驟，就是設計切換及燒錄的控制線路。在設計線路之前，先將 2716，2732，2764 的接腳情形及其工作函數分析，規劃清楚。因爲這三種 EPROM 爲 INTEL 同一系列產品



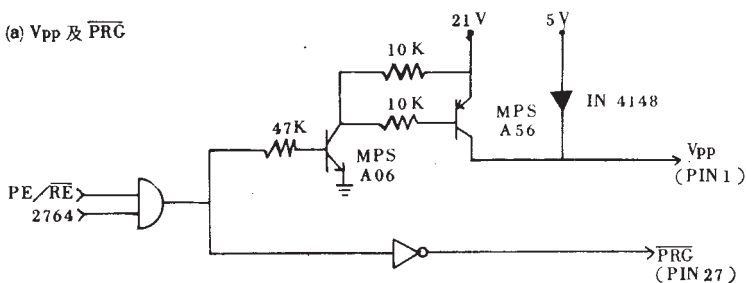
SOCKET

	V_{pp}^* (1)	$\overline{CE}$ (20)	$\overline{OE}/V_{pp}$ (22)	V_{pp}/A_{11} (23)	$\overline{PGM}$ (27)
2716: READ		V_L	V_L	+ 5	
Program			V_H	+25	
2732: READ			V_L	A_{11}	
Program			+ 25	A_{11}	
2764: READ	+ 5		V_L	A_{11}	V_H
Program	+21		V_L	A_{11}	V_L

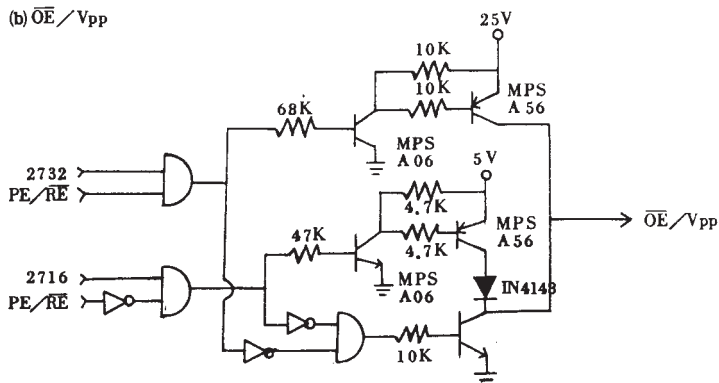
* $t = 50 \text{ ms}$

表二 2716, 2732, 2764 接脚及工作函数

(a) V_{pp} 及 $\overline{PRG}$



(b) $\overline{OE}/V_{pp}$



圖三 V_{PP} , $\overline{PRG}$, V_{PP}/A_{11} , 及 $\overline{OE}/V_{PP}$ 的切換線路

The circuit diagram shows a 2716 decoder with inputs $\overline{PE}/\overline{RE}$ and A_{11} . The output V_{PP}/A_{11} is connected to a 5V supply. The circuit includes two MPS A56 transistors, two MPS A06 transistors, and an IN 4148 diode. The 5V supply is connected to the base of the top MPS A56 transistor, which is also connected to the output V_{PP}/A_{11} . The base of the bottom MPS A56 transistor is connected to the output V_{PP}/A_{11} through a 10K resistor. The circuit also includes resistors of 10K, 68K, 47K, 4.7K, and 10K, and a diode IN 4148.

控 制 線 E _{PROM}	2716	PE/ $\overline{\text{RE}}$	PD	$\overline{\text{CE}}$	狀 態 說 明
2 7 1 6	1	0	0	0 *	燒錄前之預備狀態
	1	1	1	1	燒錄狀態
	1	0	1	0	讀取資料
2 7 3 2 or 2 7 6 4	0	0	0	1 *	燒錄前之預備狀態
	0	1	1	0	燒錄狀態
	0	0	1	0	讀取資料

有了表二的規劃表，即可分配控制線，參考圖二將 PIA0 的 PA4—PA0 及 PB7—PB0 分配給地址線 A12—A0，將 PIA1 的 PB7—PB0 分配給資料線 D7—D0 使用，剩下五個未被分配的接腳 V_{pp} ， $\overline{CE}$ ， $\overline{OE}/V_{pp}$ ， V_{pp}/A_{11} 及 $\overline{PGM}$ ，則需設計一個切換線路來控制了，首先考慮將切換三種 EPROM 的三條控制線交由 PA2，PA1，及 PA0 來控制。控制情形是一若 EPROM 是 2716，則 (PA2, PA1, PA0) = (0 0 1)。若是 2732 則 (PA2, PA1, PA0) = (0 1 0)。若是 2764，則 (PA2, PA1, PA0) = (1 0 0)，另外需一條控制線來控制 EPROM 為讀 (READ) 或燒錄 (PROGRAMMING) 的情況，以 $\overline{PE}/\overline{RE}$ 代表這條控制線，我們把 PIA1 的 PA4 分配給

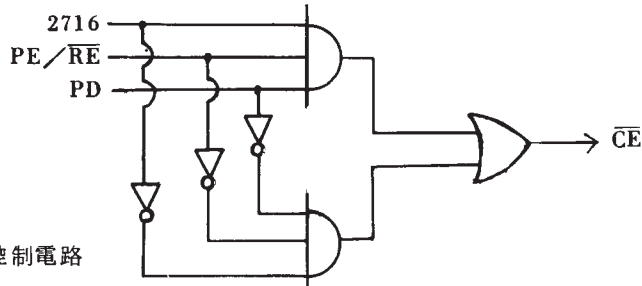
最後還有一支接腳沒有解決—CE (PIN 20)，這支接腳是在切換及燒錄控制線路中較不易解決的一支接腳，在 2716，2732，2764 使用手冊中，強調說明此接腳在燒錄時，必須是 Pulse，若是 DC 訊號則無法燒錄，更

表四 $\overline{CE}$ 之 Karnaugh Map

PE 2716 PD	00	01	11	10
0	1	0	0	—
1	0	0	1	—

$$* PE = PE / \overline{RE}$$

$$\overline{CE} = 2716 \cdot \overline{PE} \cdot \overline{PD} + 2716 \cdot PE \cdot PD$$



圖四 CE 之控制電路

不妙的是 2716 與 2732 (or 2764) Pulse 的方向不同，如前面表二中所示。因此，必須多加一條控制線來控制 Pulse 的方向，以 PD (Pulse Direction) 代表這條控制線。在設計此接腳控制線路時，先將其值表列出，如表三所示：

在數位電路中一相當有名的定理 — Karnaugh Method，在此不妨學以致用一番，參考表三，將 Karnaugh Map 畫出，如表四所示：

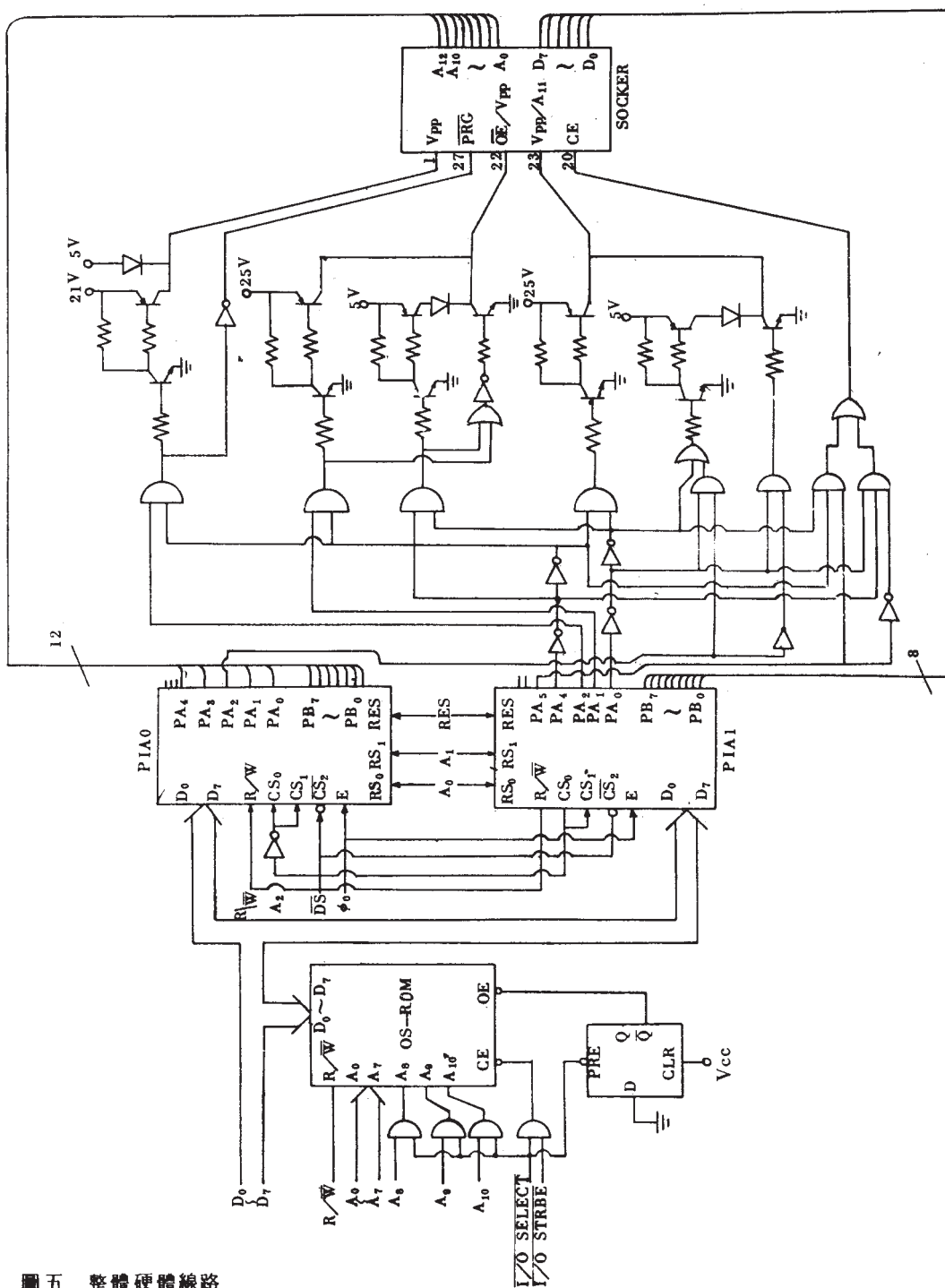
$\overline{OE}$ 求出後，即可將 $\overline{OE}$ 之控制電路設計出，如圖四所示，在軟體控制方面，只要注意在送出 50ms 之 Pulse 前，將 ($PE/\overline{RE}$, PD) 設定為 (0 , 0) 即可，根本可以不去理會被燒錄的 EPROM 是那一種。如果您對 Karnaugh Method 不甚了解，另外可以用軟體程式解決，就是將 PD 控制線直接接到 $\overline{CE}$ 的接腳，如果您這樣設計的話，就必須在軟體方面花一番腦筋了。

至此，EPROM 燒錄器的硬體電路全部設計完成，再將上述討論的各個控制線路一一連接組合起來，如圖五所示，其整體的工作情形即可一目了然。

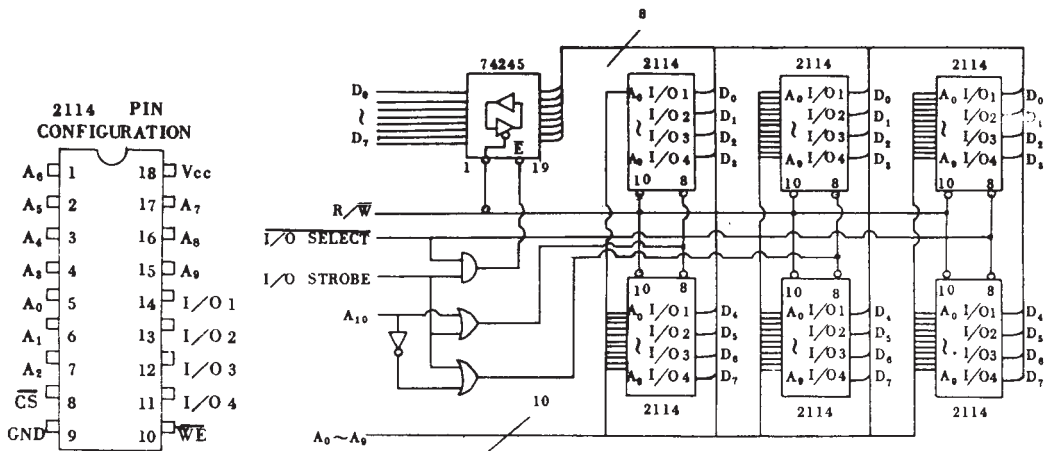
完成硬體設計後，接著便是控制程式的設計，據統計，一位高級軟體工程師，平均每天大約以 2 ~ 3 個指令的進度進行。因此，在設計此一不算短的控制程式時，心裡先有個準備，就是這段控制程式不可能在短短的幾天內可以完成，不過，正因如此，這正是訓練自己設計機械語言的最好機會。

在動手寫控制程式前，先考慮一件事，Apple 留給我們使用的 I/O 2K ROM 的位置，並沒有任何 Memory 可存控制程式，因此只好在別的 Memory 位置寫好控制程式，然後一個指令一個指令對照其相對位置燒錄在 OS-ROM 內，如果您想這樣做的話，像如此長達四、五百個指令的程式，將有抓不完，殺不盡的“虫”，建議您不如花費一些小錢換取您寶貴的時間，買六顆 ROM-2114，製作一片介面卡，我們姑且稱之為 I/O-ROM 卡，如圖 6 所示，做好後插入 Apple 中的任何一個 Slot，您就能愉快的使用該 Slot 的 256 byte 及 2K 的 I/O ROM 的位置。

另一項建議，就是一您的控制程式不妨用 LISA 系統來設計，而且 LISA 的使用方法相當簡便，只要買本 LISA 2, 5 user's gu-



圖五 接體硬體線路



圖六 I/O RAM 卡

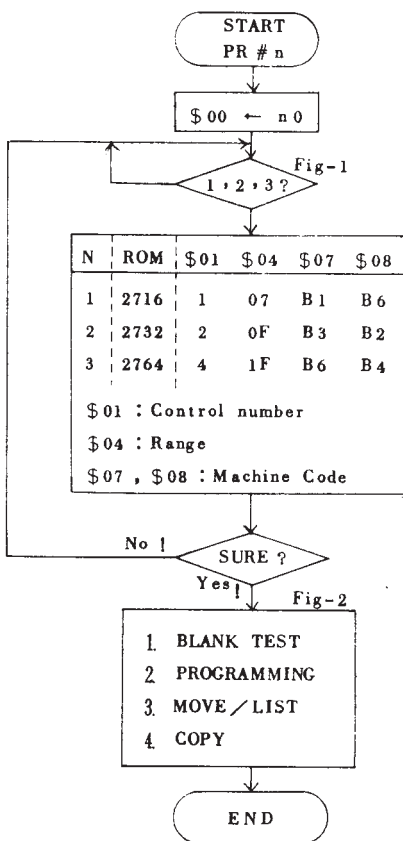


FIG-1

EPROM WRITER

(1)-2716
(2)-2732
(3)-2764
WHAT KIND IS IT ?

FIG-2

27XX EPROM WRITER

(1) BLANK TEST
(2) PROGRAMMING
(3) MOVE / LIST
(4) COPY
WHAT CAN I DO FOR YOU ?

流程一 控制程式之流程圖

FIG-3

```

27XX    BLANK TEST

        BEGIN POINT : _ _ _ _

        READY ?
  
```

FIG-4

```

27XX    PROGRAMMING

        PROGRAM ADDRESS
          FROM _ _ _ _ TO _ _ _ _

        ROM
          BEGIN POINT : _ _ _ _
  
```

FIG-5

```

27XX    MOVE / LIST

        ROM LOCATION
          FROM _ _ _ _ TO _ _ _ _

        MOVE TO MEMORY
          BEGIN POINT: _ _ _ _
  
```

ide，照著圖上所寫的操作，不出幾小時，您就能操縱自如。一切準備就緒後，就開始動手寫控制程式吧！首先要知道的是我們將付與這 EPROM 燒錄器那些功能，一般而言，一個燒錄器必須具備三種基本功能：(一)空白的測試 (BLANK)，(二)燒錄程式或資料 (PROGRAMMING)，(三)顯示 EPROM 的內容 (Data Move/List)。若要加強其複製 (COPY) 的功能，只需將上述三種基本功能串接起來即可。在流程一 1 所示，即為控制式的大略流程。

在起始程式後第一個遭遇的問題，就是處理螢幕顯示文字的問題，要顯示字幕必先將被顯示的字碼 (ASCII Code) 找出。當然，這些字碼不需要一個字一個字的去查，去記，那是相當落伍的方法，我們既然有台 Apple，就應該把這份煩人的工作交由 Apple 處理。如程式一 1 所做的工作就是將字碼查出的工作，在執行這段程式時，無論螢幕出現什麼字，就會將該字的字碼存入一特定的 Memory，存好字碼後，再執行程式一 2，即可將在程式一 1

時所顯示的字幕完全一模一樣的再顯現在螢幕上。如此一來，解決螢幕顯示的問題就變得簡單而快速。在整個控制程式中，可把處理螢幕顯示的程式一 2 當作一個子程式 (SUBROUTINE) 來用。凡是需要改變字幕時，只需叫 (CALL) 這個子程式即可完成。

程式一 1：查存字符的 ASCII code (以 256 字為限)

```

JSR $FC58
DATAIN      ;清除螢幕
JSR $FD35
            ;由KEYBOARD 輸入字符
LDX #0
STA $CD00,X
            ;字碼從$CD00 的位置開始存起
CMP #93
            ;字碼存完時，按CTRL-S
BEQ END
JSR $FDED
  
```

```

; 將該字顯示在螢幕上
INX
JMP DATAIN
; 輸入下一個字符

```

END END

程式－2：字幕顯示（以 256 字為限）

```

JSR $FC58
; 清除螢幕
LDY #0
GETDATA LDA $CD00, Y
; 取字碼
BEQ END
JSR $FDED
; 將字符顯示出
INY
JMP GETDATA
END       END

```

在流程－1 的 Fig－1 所示，在燒錄 EPROM 之前，必須先確定燒錄的 EPROM 是那一種，即配合硬體切換線路將特殊控制碼存在 \$01 的位置，若 EPROM 是 2716 則存入 #\$01，若 EPROM 是 2732 則存入 #\$02，若 EPROM 是 2764 則存在 #\$04，其燒錄的控制程式如程式－3 所示：

程式－3：燒錄控制程式

```

燒錄資料：LDX $00
; 取 Slot 之參數
LDA $01
; 決定 EPROM 的種類
CLC
ADC #10
; 定燒錄狀態的控制碼
STA $C084, X
; 燒錄開始
LDA #8C
JSR $FCA8
; 延遲 50msec
LDA #00
STA $C084, X
; 燒錄完成

```

讀取資料：LDX \$00

LDA \$01

CLC

ADC #20

; 定讀取資料狀態的控制碼

STA \$C084, X

; 將控制碼輸出

LDA \$C086, X

; 將資料讀入

至此，已經具備了啟動程式，螢幕處理程式、燒錄、讀取以及切換控制程式，剩下的工作，就是將整個過程連貫起來，下面所述是三種基本功能簡單的工作情形及要求：

(一)空白測試（BLANK TEST）：

將 EPROM 的資料讀入與 #FF 比，如果不相等，就跳到一個顯示錯誤的子程式。為加強其聲動的效果，可加入聲音及畫面的效果。

(二)燒錄資料（PROGRAMMING）：

先將地址及資料給定好之後，再送出燒錄控制碼及延遲一段 50m Sec 的時間，燒錄完成後，再將燒入的資料讀入與原資料比較，若不相等，則跳到錯誤的子程式。

(三)搬移（顯示）EPROM 內的資料（MOVE /LIST）：

將 EPROM 的資料讀入存到使用者所指定的位置。

好了，剩下的工作，都是別人幫不上忙的，只有靠您自己了。本來世界上有許多事情除了自己外，沒有人能把它做得更好，不是嗎？在此討論的 EPROM 燒錄器，其過程只能提供做為參考，因為您自己能把它做得更好，也許您的過程更符合電腦時代的要求。

控制程式之記憶地址安排

\$00：存放此 EPROM 燒錄器所在 Slot 之參數－\$N0，（N 為 Slot 之號數），此參數將在執行 I/O 程式時被應用到。

\$01：存放硬體線路切換控制碼，在控制程式中用以認知使用的 EPROM 的種類是 2716 或 2732 或 2764。

\$04：所存之值用以辨認該 EPROM 的最大範圍。2716 其範圍 \$0000～\$07FF，在 \$04 的位置存 \$07。2732 一範圍 \$0000～\$0FFF，存 \$0F。2764 一範圍 \$0000～\$1FFF，存 \$1F。

\$07，\$08：存放 Machine Code，用以顯示在 Fig 1～Fig 5 頂端所示的 27XX（2716，2732，or 2764），以便能使 User 查覺是否操作無誤。
（位址 \$01，\$04，\$07，\$08 所存之值已例在流程一 1 中）

\$24，\$25：此對記憶位置與 Subroutine \$FC22 配合執行，可任意調整 Cursor 的位置，此位置即為下一個 Character

將被顯示的位置。

\$A0，\$A1：此對記憶位置所存的地址是處理螢幕顯示字幕之字碼（ASCII Code）所在的位置，在顯示字幕常式（DISFIG Subroutine）中將被用到。

\$A7，\$A8：為工作常式指標（Service Routine Vector）。

\$60，\$61 \$62，\$63 \$A2，\$A3：在顯幕顯示中有“FROM TO”的字樣，是由 User 下定工作的範圍，此六個記憶位址即用來運算 User 給予的範圍，而且將運算出來的範圍存在（\$A2，\$A3）的位置中，以便讓控制程式認知其工作是否完成。

OS-ROM 機械語言程式列表

0800	1	DCM 'PR#1'	
0800	2 *		
0800	3 *	EPROM WRITER	
0800	4 *		
0800	5 ;		
C800	6	ORG \$C800	
C800 A5 37	7	LDA \$37	;GET NUMBER OF SLOT
C802 0A	8	ASL	
C803 0A	9	ASL	
C804 0A	10	ASL	
C805 0A	11	ASL	
C806 85 00	12	STA \$0	;STORE NUMBER \$NO
C808 A9 F0	13	LDA 0F0	;BACK TO MONITER
C80A 85 36	14	STA \$36	
C80C A9 FD	15	LDA 0FD	
C80E 85 37	16	STA \$37	
C810 4C 4C CA	17	JMP START	
C813 20 47 C9	18	DOBLANK JSR ROMREAD	;BLANK ROUTINE
C816 C9 FF	19	CMP OFF	
C818 F0 03	20	BEQ BLCONT	
C81A 4C 78 C9	21	JMP ERROR	
C81D 4C 12 CA	22	BLCONT JMP BCONT	;BACK TO LOOP
C820 A4 00	23	DOPROG LDY \$0	;PROGRAMMING ROUTINE
C822 A9 00	24	LDA 0	;CONTROL BUFFER
C824 99 87 C0	25	STA \$C087,Y	
C827 A9 FF	26	LDA OFF	;OUT CONDITION
C829 99 86 C0	27	STA \$C086,Y	
C82C A9 04	28	LDA 04	
C82E 99 87 C0	29	STA \$C087,Y	
C831 B1 62	30	LDA (\$62),Y	;GET DATA
C833 99 86 C0	31	STA \$C086,Y	;OUTPUT DATA
C836 85 64	32	STA \$64	
C838 A9 00	33	LDA 0	
C83A 99 84 C0	34	STA \$C084,Y	
C83D A5 01	35	LDA \$1	
C83F 18	36	CLC	
C840 69 10	37	ADC 10	;WRITE CONTION
C842 99 84 C0	38	STA \$C084,Y	
C845 A9 8C	39	LDA 8C	;DELAY 50 MS

C847	20	A8	FC	40		JSR %FCAS	
C84A	20	47	C9	41		JSR ROMREAD	
C84D	C5	64		42		CMP %64	;CHECK DATA
C84F	D0	03		43		BNE TRY	;IF ERR TRY AGAIN
C851	4C	05	CA	44		JMP BCONT0	;BACK TO COUNT LOOP
C854	20	DD	FB	45	TRY	JSR %FBDD	;WRITE ANOTHER 3 TIMES
C857	A5	65		46		LDA %65	
C859	C9	03		47		CMP 03	
C85B	F0	05		48		BEQ BLANKERR	
C85D	E6	65		49		INC %65	
C85F	4C	20	C8	50		JMP DOPROG	
C862	4C	78	C9	51	BLANKERR	JMP ERROR	
C865	A5	63		52	DOMOVE	LDA %63	;MOVE ROUTINE
C867	99	80	C0	53		STA %C080,Y	;GIVE ROM LOCATION
C86A	A5	62		54		LDA %62	
C86C	99	82	C0	55		STA %C082,Y	
C86F	20	47	C9	56		JSR ROMREAD	
C872	A2	00		57		LDX 0	
C874	81	60		58		STA (%60,X)	;STORE DATA IN %1000
C876	4C	05	CA	59		JMP BCONT0	
C879	20	35	FD	60	TKEY	JSR %FD35	;TEST KEY IN
C87C	85	1E		61		STA %1E	
C87E	A2	C1		62		LDX 0C1	
C880	86	1A		63	CH1	STX %1A	;CHECK A-F
C882	C5	1A		64		CMP %1A	
C884	F0	24		65		BEQ ADD9	
C886	E6	1A		66		INC %1A	
C888	A6	1A		67		LDX %1A	
C88A	E0	C7		68		CPX 0C7	
C88C	F0	03		69		BEQ N9	
C88E	4C	80	C8	70		JMP CH1	
C891	A2	B0		71	N9	LDX 0B0	;CHECK 0-9
C893	86	1A		72	N91	STX %1A	
C895	C5	1A		73		CMP %1A	
C897	F0	14		74		BEQ MUL1	
C899	E6	1A		75		INC %1A	
C89B	A6	1A		76		LDX %1A	
C89D	E0	BA		77		CPX 0BA	
C89F	F0	03		78		BEQ ERR1	
C8A1	4C	93	C8	79		JMP N91	
C8A4	20	DD	FB	80	ERR1	JSR %FBDD	;KEY IN ERR
C8A7	4C	79	C8	81		JMP TKEY	
C8AA	18			82	ADD9	CLC	;TRANSFER M-CODE TO 0-F
C8AB	69	09		83		ADC 09	
C8AD	29	0F		84	MUL1	AND 0F	
C8AF	60			85		RTS	
C8B0	20	79	C8	86	DIS0	JSR TKEY	;H-LEVEL OF BYTE
C8B3	0A			87		ASL	
C8B4	0A			88		ASL	
C8B5	0A			89		ASL	
C8B6	0A			90		ASL	
C8B7	A0	00		91		LDY 00	
C8B9	91	1B		92		STA (%1B),Y	;STORE AT (%1B)
C8BB	A5	1E		93		LDA %1E	
C8BD	20	ED	FD	94		JSR %FDED	
C8C0	20	79	C8	95	DISNEXT	JSR TKEY	;L-LEVEL OF BYTE
C8C3	A0	00		96		LDY 0	
C8C5	18			97		CLC	
C8C6	71	1B		98		ADC (%1B),Y	
C8C8	91	1B		99		STA (%1B),Y	
C8CA	A5	1E		100		LDA %1E	
C8CC	20	ED	FD	101		JSR %FDED	
C8CF	60			102		RTS	
C8D0	85	24		103	BPOINT	STA %24	
C8D2	86	25		104		STX %25	
C8D4	20	22	FC	105		JSR %FC22	;DEFIND CURSOR LOCATION

C8D7	A2	00	106	LDX 0	
C8D9	A9	61	107	LDA 61	;STORE H-BEGIN-POINT
C8DB	85	1B	108	STA #1B	
C8DD	86	1C	109	STX #1C	
C8DF	20	B0 C8	110	JSR DISO	
C8E2	C6	1B	111	DEC #1B	;#60 STORE L-BEGIN-POINT
C8E4	20	B0 C8	112	JSR DISO	
C8E7	60		113	RTS	
C8E8	85	A1	114	DISFIG STA #A1	; (AX) GIVE DATA LOCAION
C8EA	86	A0	115	STX #A0	
C8EC	A0	00	116	DISFIG1 LDY 0	
C8EE	B1	A0	117	DISFIG2 LDA (#A0),Y	
C8F0	C9	C0	118	CMP #*Q*	;REFERENCE DATA OF STOP
C8F2	F0	13	119	BEQ FIGRTS	
C8F4	20	ED FD	120	JSR #FDED	
C8F7	18		121	CLC	
C8F8	A5	A0	122	LDA #A0	
C8FA	69	01	123	ADC 1	
C8FC	85	A0	124	STA #A0	
C8FE	A5	A1	125	LDA #A1	
C900	69	00	126	ADC 00	
C902	85	A1	127	STA #A1	
C904	4C	EC C8	128	JMP DISFIG1	
C907	60		129	FIGRTS RTS	
C908	A9	06	130	BFIG LDA 6	
C90A	85	24	131	STA #24	
C90C	A9	0B	132	LDA 0B	
C90E	85	25	133	STA #25	
C910	20	22 FC	134	JSR #FC22	;DEFIND CURSOR POSITION
C913	A9	CF	135	LDA OCF	
C915	A2	15	136	LDX 15	
C917	20	E8 C8	137	JSR DISFIG	;DIS 'BEGIN POINT'
C91A	60		138	RTS	
C91B	A9	0F	139	BACKF LDA 0F	
C91D	A2	06	140	LDX 06	
C91F	20	D0 C8	141	JSR BPOINT	;DIS 'FROM_TO_'
C922	A5	61	142	LDA #61	
C924	85	63	143	STA #63	
C926	A5	60	144	LDA #60	
C928	85	62	145	STA #62	
C92A	A9	17	146	LDA 17	
C92C	A2	06	147	LDX 6	
C92E	20	D0 C8	148	JSR BPOINT	
C931	38		149	SEC	
C932	A5	60	150	LDA #60	
C934	E5	62	151	SBC #62	
C936	85	A2	152	STA #A2	
C938	A5	61	153	LDA #61	
C93A	E5	63	154	SBC #63	
C93C	85	A3	155	STA #A3	; (A3.A2) STORE RANGE
C93E	60		156	RTS	
C93F	A9	13	157	BACKB LDA 13	
C941	A2	0B	158	LDX 0B	
C943	20	D0 C8	159	JSR BPOINT	
C946	60		160	RTS	
C947	A5	01	161	ROMREAD LDA #01	;READ ROM
C949	18		162	CLC	
C94A	69	20	163	ADC 20	
C94C	A4	00	164	LDY #00	
C94E	99	84 C0	165	STA #C084,Y	
C951	A9	00	166	LDA 0	
C953	99	87 C0	167	STA #C087,Y	
C956	99	86 C0	168	STA #C086,Y	
C959	A9	04	169	LDA 04	
C95B	99	87 C0	170	STA #C087,Y	
C95E	B9	86 C0	171	LDA #C086,Y	

C961	60	172		RTS	
C962	A9 06	173	READY	LDA 6	DIS 'READY?'
C964	85 24	174		STA #24	
C966	A9 0E	175		LDA 0E	
C968	85 25	176		STA #25	
C96A	20 22 FC	177		JSR #FC22	
C96D	A9 CF	178		LDA OCF	
C96F	A2 28	179		LDX 28	
C971	20 E8 C8	180		JSR DISFIG	
C974	20 35 FD	181		JSR #FD35	
C977	60	182		RTS	
C978	20 DD FB	183	ERROR	JSR #FBDD	DIS 'ERROR'
C97B	20 DD FB	184		JSR #FBDD	
C97E	A9 06	185		LDA 6	
C980	85 24	186		STA #24	
C982	A9 0E	187		LDA 0E	
C984	85 25	188		STA #25	
C986	20 22 FC	189		JSR #FC22	
C989	A9 CF	190		LDA OCF	
C98B	A2 31	191		LDX 31	
C98D	20 E8 C8	192		JSR DISFIG	
C990	60	193		RTS	
C991	A9 06	194	AGAIN	LDA 6	DIS 'AGAIN'
C993	85 24	195		STA #24	
C995	A9 11	196		LDA 11	
C997	85 25	197		STA #25	
C999	20 22 FC	198		JSR #FC22	
C99C	A9 CF	199		LDA OCF	
C99E	A2 3E	200		LDX 3E	
C9A0	20 E8 C8	201		JSR DISFIG	
C9A3	20 35 FD	202		JSR #FD35	
C9A6	60	203		RTS	
C9A7	A5 07	204	ROMN	LDA #7	DIS ROM NUMBER
C9A9	8D 83 04	205		STA #483	
C9AC	A5 08	206		LDA #8	
C9AE	8D 84 04	207		STA #484	
C9B1	60	208		RTS	
C9B2	85 A8	209	CONT	STA #A8	COUNTER LOOP
C9B4	86 A7	210		STX #A7	BE USED IN SERVICE SUBROUTIN
E					
C9B6	A4 00	211		LDY #0	
C9B8	A9 00	212		LDA 0	
C9BA	99 81 C0	213		STA #C081,Y	
C9BD	99 83 C0	214		STA #C083,Y	
C9C0	99 85 C0	215		STA #C085,Y	
C9C3	A9 FF	216		LDA OFF	
C9C5	99 80 C0	217		STA #C080,Y	
C9C8	99 82 C0	218		STA #C082,Y	
C9CB	99 84 C0	219		STA #C084,Y	
C9CE	A9 04	220		LDA 04	
C9D0	99 81 C0	221		STA #C081,Y	
C9D3	99 83 C0	222		STA #C083,Y	
C9D6	99 85 C0	223		STA #C085,Y	
C9D9	A9 0E	224		LDA 0E	
C9DB	85 22	225		STA #22	
C9DD	20 58 FC	226		JSR #FC58	CLEAR CONT SCREEN
C9E0	A9 00	227		LDA 0	
C9E2	85 22	228		STA #22	
C9E4	A9 13	229	CONT1	LDA 13	
C9E6	85 24	230		STA #24	
C9E8	A9 0E	231		LDA 0E	
C9EA	85 25	232		STA #25	
C9EC	20 22 FC	233		JSR #FC22	
C9EF	A5 61	234		LDA #61	
C9F1	A6 60	235		LDX #60	
C9F3	20 41 F9	236		JSR #F941	COUNTER OUT
C9F6	A4 00	237		LDY #0	
C9F8	A5 61	238		LDA #61	
C9FA	99 80 C0	239		STA #C080,Y	

C9FD	A5	60	240		LDA #60	
C9FF	99	82	C0	241	STA #C082,Y	
CA02	6C	A7	00	242	JMP (#A7)	
CA05	A5	62	243	BCONTO	LDA #62	
CA07	18		244		CLC	
CA08	69	01	245		ADC 1	
CA0A	85	62	246		STA #62	
CA0C	A5	63	247		LDA #63	
CA0E	69	00	248		ADC 0	
CA10	85	63	249		STA #63	
CA12	A5	A3	250	BCONT	LDA #A3	
CA14	C9	00	251		CMP 0	
CA16	D0	09	252		BNE CONTL00P	
CA18	A6	A2	253		LDX #A2	
CA1A	E0	00	254		CPX 00	
CA1C	D0	03	255		BNE CONTL00P	
CA1E	4C	3E	CA	256	JMP OK	
CA21	A5	A2	257	CONTL00P	LDA #A2	
CA23	38		258		SEC	
CA24	E9	01	259		SBC 1	
CA26	85	A2	260		STA #A2	
CA28	A5	A3	261		LDA #A3	! RANGE-1
CA2A	E9	00	262		SBC 0	
CA2C	85	A3	263		STA #A3	
CA2E	A5	60	264		LDA #60	
CA30	18		265		CLC	
CA31	69	01	266		ADC 1	
CA33	85	60	267		STA #60	! ADDRESS+1
CA35	A5	61	268		LDA #61	
CA37	69	00	269		ADC 0	
CA39	85	61	270		STA #61	
CA3B	4C	E4	C9	271	JMP CONT1	
CA3E	A9	CF		272	OK	LDA "0
CA40	8D	63	04	273		STA #463
CA43	A9	CB		274		LDA "K
CA45	8D	64	04	275		STA #464
CA48	20	DD	FB	276		JSR #FBDD
CA4B	60		277		RTS	
CA4C	20	58	FC	278	START	JSR #FC58
CA4F	A9	00	279			LDA 0
CA51	85	65	280			STA #65
CA53	A9	CD	281			LDA 0CD
CA55	A2	00	282			LDX 00
CA57	20	E8	C8	283		JSR DISFIG
CA5A	20	35	FD	284	NN	JSR #FD35
CA5D	C9	B1	285			CMP #"1"
CA5F	F0	0E	286			BEQ N1
CA61	C9	B2	287			CMP #"2"
CA63	F0	1D	288			BEQ N2
CA65	C9	B3	289			CMP #"3"
CA67	F0	2C	290			BEQ N3
CA69	20	DD	FB	291		JSR #FBDD
CA6C	4C	5A	CA	292		JMP NN
CA6F	A9	00	293	N1		LDA 00
CA71	85	01	294			STA #01
CA73	A9	07	295			LDA 7
CA75	85	04	296			STA #4
CA77	A9	B1	297			LDA 0B1
CA79	85	07	298			STA #7
CA7B	A9	B6	299			LDA 0B6
CA7D	85	08	300			STA #8
CA7F	4C	A5	CA	301		JMP SURE
CAB2	A9	01	302	N2		LDA 1
CAB4	85	01	303			STA #1
CAB6	A9	0F	304			LDA 0F
CAB8	85	04	305			STA #4

CABA	A9	B3	306		LDA	OB3	
CAB0	85	07	307		STA	#7	
CABE	A9	B2	308		LDA	OB2	
CA90	85	08	309		STA	#8	
CA92	4C	A5	310		JMP	SURE	
CA95	A9	02	311	N3	LDA	2	12764
CA97	85	01	312		STA	#1	
CA99	A9	1F	313		LDA	1F	
CA9B	85	04	314		STA	#4	
CA9D	A9	B6	315		LDA	OB6	
CA9F	85	07	316		STA	#7	
CAA1	A9	B4	317		LDA	OB4	
CAA3	85	08	318		STA	#8	
CAA5	A9	CD	319	SURE	LDA	0CD	
CAA7	A2	52	320		LDX	52	
CAA9	20	E8	321		JSR	DISFIG	DIS FIG 'SURE'
CAAC	A5	07	322		LDA	#7	
CAAE	8D	2C	323		STA	#72C	DIS ROM NUMBER
CAB1	A5	08	324		LDA	#8	
CAB3	8D	2D	325		STA	#72D	
CAB6	20	35	326	SUREIN	JSR	#FD35	
CAB9	C9	D9	327		CMP	"Y"	
CABB	F0	0A	328		BEQ	FIG2	
CABD	C9	CE	329		CMP	"N"	
CABF	F0	8B	330		BEQ	START	
CAC1	20	DD	331		JSR	#FBDD	
CAC4	4C	B6	332		JMP	SUREIN	
CAC7	20	58	333	FIG2	JSR	#FC58	DIS FIG2
CACA	A9	CD	334		LDA	0CD	
CACC	A2	64	335		LDX	64	
CACE	20	E8	336		JSR	DISFIG	
CAD1	20	A7	337		JSR	ROMN	
CAD4	20	35	338	NIN	JSR	#FD35	
CAD7	C9	B1	339		CMP	"1"	
CAD9	F0	18	340		BEQ	BLANK	
CADB	C9	B2	341		CMP	"2"	
CADD	F0	60	342		BEQ	PROG	
CADF	C9	B3	343		CMP	"3"	
CAE1	D0	03	344		BNE	NCOPY	
CAE3	4C	89	345		JMP	MOVE	
CAE6	C9	B4	346	NCOPY	CMP	"4"	
CAE8	D0	03	347		BNE	NINERR	
CAEA	4C	CE	348		JMP	COPY	
CAED	20	DD	349	NINERR	JSR	#FBDD	KIY IN ERR
CAFO	20	D4	350		JSR	NIN	
CAF3	20	58	351	BLANK	JSR	#FC58	BLANK TEST
CAF6	A9	CD	352		LDA	0CD	
CAF8	A2	E8	353		LDX	0E8	
CAFA	20	E8	354		JSR	DISFIG	DIS FIG3
CAFD	20	A7	355		JSR	ROMN	
CB00	20	08	356		JSR	BFIG	
CB03	20	3F	357		JSR	BACKB	
CB06	38		358		SEC		
CB07	A9	FF	359		LDA	OFF	
CB09	E5	60	360		SBC	#60	
CB0B	85	A2	361		STA	#A2	
CB0D	A5	04	362		LDA	#4	FETCH RANGE
CB0F	E5	61	363		SBC	#61	
CB11	85	A3	364		STA	#A3	
CB13	20	62	365	BLANKIN	JSR	READY	
CB16	C9	D9	366		CMP	"Y"	
CB18	F0	0A	367		BEQ	BLANKGO	
CB1A	C9	CE	368		CMP	"N"	
CB1C	F0	D5	369		BEQ	BLANK	
CB1E	20	DD	370		JSR	#FBDD	
CB21	4C	13	371		JMP	BLANKIN	

CB24	A9	C8	372	BLANKGO	LDA	OC8			
CB26	A2	13	373		LDX	13			
CB28	20	B2	C9	374		JSR	CONT		
CB2B	20	91	C9	375	INBL	JSR	AGAIN		
CB2E	C9	D9	376		CMP	"Y			
CB30	F0	C1	377		BEQ	BLANK			
CB32	C9	CE	378		CMP	"N			
CB34	D0	03	379		BNE	BERR			
CB36	4C	4C	CA	380		JMP	START		
CB39	20	DD	FB	381	BERR	JSR	*FBDD		
CB3C	4C	2B	CB	382		JMP	INBL		
CB3F	20	58	FC	383	PROG	JSR	*FC58		
CB42	A9	23	384		LDA	23			
CB44	85	A7	385		STA	*A7			
CB46	A9	C8	386		LDA	OC8			
CB48	85	A8	387		STA	*A8			
CB4A	A9	CD	388		LDA	0CD			
CB4C	A2	FD	389		LDX	0FD			
CB4E	20	E8	CB	390		JSR	DISFIG		
CB51	20	A7	C9	391		JSR	ROMN		
CB54	20	08	C9	392		JSR	BFIG		
CB57	20	1B	C9	393		JSR	BACKF		
CB5A	20	3F	C9	394		JSR	BACKB		
CB5D	20	62	C9	395	PROGIN	JSR	READY		
CB60	C9	D9	396		CMP	"Y			
CB62	F0	0A	397		BEQ	PROGO			
CB64	C9	CE	398		CMP	"N			
CB66	F0	D7	399		BEQ	PROG			
CB68	20	DD	FB	400		JSR	*FBDD		
CB6B	4C	5D	CB	401		JMP	PROGIN		
CB6E	A9	C8	402	PROGO	LDA	OC8			
CB70	A2	20	403		LDX	20			
CB72	20	B2	C9	404		JSR	CONT		
CB75	20	91	C9	405	PROIN	JSR	AGAIN		
CB78	C9	D9	406		CMP	"Y			
CB7A	F0	C3	407		BEQ	PROG			
CB7C	C9	CE	408		CMP	"N			
CB7E	D0	03	409		BNE	PROINERR			
CB80	4C	4C	CA	410		JMP	START		
CB83	20	DD	FB	411	PROINERR	JSR	*FBDD		
CB86	4C	75	CB	412		JMP	PROIN		
CB89	20	58	FC	413	MOVE	JSR	*FC58		
CB8C	A9	CE	414		LDA	0CE			
CB8E	A2	57	415		LDX	57			
CB90	20	E8	CB	416		JSR	DISFIG		
CB93	20	A7	C9	417		JSR	ROMN		
CB96	20	08	C9	418		JSR	BFIG		
CB99	20	1B	C9	419		JSR	BACKF		
CB9C	20	3F	C9	420		JSR	BACKB		
CB9F	20	62	C9	421	MOVEIN	JSR	READY		
CBA2	C9	D9	422		CMP	"Y			
CBA4	F0	0A	423		BEQ	MOVEGO			
CBA6	C9	CE	424		CMP	"N			
CBA8	F0	DF	425		BEQ	MOVE			
CBAA	20	DD	FB	426		JSR	*FBDD		
CBAD	4C	9F	CB	427		JMP	MOVEIN		
CBBO	A9	C8	428	MOVEGO	LDA	OC8			
CBB2	A2	64	429		LDX	64			
CBB4	20	B2	C9	430		JSR	CONT		
CBB7	20	91	C9	431	MOVEINO	JSR	AGAIN		
CBBA	C9	D9	432		CMP	"Y			
CBBC	F0	CB	433		BEQ	MOVE			
CBBE	C9	CE	434		CMP	"N			
CBCO	D0	03	435		BNE	MOERR			
CBC2	4C	4C	CA	436		JMP	START		
CBC5	20	DD	FB	437	MOERR	JSR	*FBDD		

CBC8	4C	B7	CB	438	JMP	MOVEINO	
CBC8	4C	B2	C9	439	JMP	CONT	
CBCE	20	58	FC	440	COPY	JSR	*FC58 ; COPY
CBD1	A9	CE		441		LDA	OCE
CB03	A2	B2		442		LDX	0B2
CB05	20	E8	CB	443		JSR	DISFIG ; PUT MAST ROM
CB08	20	62	C9	444	COPYIN	JSR	READY
CBDB	C9	D9		445		CMP	*Y
CBDD	F0	1C		446		BEQ	COPYGO
CBDF	C9	CE		447		CMP	*N
CRE1	D0	03		448		BNE	COPYERR
CRE3	4C	4C	CA	449		JMP	START
CBE6	20	DD	FB	450	COPYERR	JSR	*FBDD
CBE9	4C	CE	CB	451		JMP	COPY
CBEC	A5	04		452	CORANGE	LDA	*4
CBEE	85	A3		453		STA	*A3
CBFO	A9	FF		454		LDA	OFF
CBF2	85	A2		455		STA	*A2 ; GIVE RANGE
CBF4	A9	00		456		LDA	0
CBF6	85	62		457		STA	*62
CBF8	85	60		458		STA	*60
CBFA	60			459		RTS	
CBFB	20	EC	CB	460	COPYGO	JSR	CORANGE
CBFE	85	63		461		STA	*63
CC00	A9	10		462		LDA	10
CC02	85	61		463		STA	*61 ; MOVE TO *1000
CC04	A9	C8		464		LDA	0C8
CC06	A2	64		465		LDX	64
CC08	A2	64		466		LDX	64
CC0A	20	B2	C9	467		JSR	CONT ; DO MOVE
CC0D	20	DD	FB	468		JSR	*FBDD
CC10	20	58	FC	469	COPY1	JSR	*FC58
CC13	A9	CE		470		LDA	OCE
CC15	A2	E4		471		LDX	0E4
CC17	20	E8	CB	472		JSR	DISFIG ; PUT BLANK ROM
CC1A	20	62	C9	473	COPY1IN	JSR	READY
CC1D	C9	D9		474		CMP	*Y
CC1F	F0	0D		475		BEQ	COPY1GO
CC21	C9	CE		476		CMP	*N
CC23	D0	03		477		BNE	COPY1ERR
CC25	4C	CE	CB	478		JMP	COPY
CC28	20	DD	FB	479	COPY1ERR	JSR	*FBDD
CC2B	4C	1A	CC	480		JMP	COPY1IN
CC2E	20	EC	CB	481	COPY1GO	JSR	CORANGE
CC31	85	61		482		STA	*61
CC33	A9	10		483		LDA	10
CC35	85	63		484		STA	*63
CC37	A9	C8		485		LDA	0C8
CC39	A2	20		486		LDX	20
CC3B	20	B2	C9	487		JSR	CONT ; DO PROG
CC3E	20	91	C9	488	COPYIN2	JSR	AGAIN
CC41	C9	D9		489		CMP	*Y
CC43	D0	03		490		BNE	NOCOPY
CC45	4C	CE	CB	491		JMP	COPY
CC48	C9	CE		492	NOCOPY	CMP	*N
CC4A	D0	03		493		BNE	COPYERR2
CC4C	4C	4C	CA	494		JMP	START
CC4F	20	DD	FB	495	COPYERR2	JSR	*FBDD
CC52	4C	3E	CC	496		JMP	COPYIN2
CC55				497		END	

***** END OF ASSEMBLY

使APPLE說話的

界面板製作

在前面幾個APPLE II 專題製作中，我們曾介紹如何利用 6522 VIA 一多用途周邊界面調適器，配合程式及簡單的硬體，來作 6522 I/O CARD，EPROM WRITER CARD，APPLE II 周邊界面轉接器，EPROM/RAM CARD……等各種應用。在你看了上述技術與理論並重的製作後，是否想嘗試一些較輕鬆且有趣味性的製作呢？

在本文的硬體製作中，我們將介紹一個由 G I 公司所推出的可程式控制聲音產生器（Programmable Sound Generator，簡稱 PSG）AY-3-8912 的工作原理，然後告訴你如何利用 6522 VIA 配合程式的控制，

使 PSG 晶片模擬各種聲音，如槍聲、爆炸聲、鋼琴聲……等。

PSG(Programmable Sound Generator)——可程式控制聲音產生器，能將三個由程式控制產生的方波和一個噪音產生器(Noise Generator)所產生的信號混合，且利用 D/A converter 使三個不同的頻率，分別從不同的頻道輸出。而每個頻道輸出可個別的連接到放大器。另外在 PSG 晶片中還有一個包線產生器(envelope generator)可用來控制輸出信號的包線(envelope)。

在 PSG 晶片，即AY-3-8912 IC 中，具有16個可接受程式控制的暫存器，因此只要

		BIT							
REGISTER		B7	B6	B5	B4	B3	B2	B1	B0
R0	Channel A Tone Period	8-BIT Fine Tune A							
R1						4-BIT Coarse Tune A			
R2	Channel B Tone Period	8-BIT Fine Tune B							
R3						4-BIT Coarse Tune B			
R4	Channel C Tone Period	8-BIT Fine Tune C							
R5						4-BIT Coarse Tune C			
R6	Noise Period					5-BIT Period Control			
R7	Enable	IN/OUT		Noise			Tone		
		IOB	IOA	C	B	A	C	B	A
R10	Channel A Amplitude				M	L3	L2	L1	L0
R11	Channel B Amplitude				M	L3	L2	L1	L0
R12	Channel C Amplitude				M	L3	L2	L1	L0
R13	Envelope Period	8-BIT Fine Tune E							
R14		8-BIT Coarse Tune E							
R15	Envelope Shape/Cycle					CONT	ATT	ALT	HOLD
R16	I/O Port A Data Store	8-BIT PARALLEL I/O on Port A							
R17	I/O Port B Data Store	8-BIT PARALLEL I/O Port B							

圖 1 PSG 晶片中各暫存器之功能

利用一些簡單的程式控制，即可完成上述功能。爲了能有效的控制AY-3-8912的動作，現在就先簡單的說明其工作原理。

AY-3-8912 主要是利用除頻的方式來產生單一的聲音，首先將 I C 輸入端的時脈信號除以16，然後這個信號在輸入 I C 後會再被一個 COUNTER 除以12。而這個12位元的 channel A 信號將會被放入暫存器 R 0（存放較低次的8個位元），而剩的較高次4個位元則放入暫存器 R 1。根據上面的說明知道，對一個已知的時鐘脈波，可以利用下述公式算出其聲頻周期（tone period）tp：

$$tp = f_{clock} / (f * 16)$$

f = 希望產生的頻率

f_{clock} = 加到 I C 上的時鐘脈波頻率
單位均是 Hz

例如：f = 440Hz，f_{clock} = 1,000,000Hz
tp = 1,000,000 / 440 * 16 = 142.04Hz

假如你將142轉換成12 bit的二進位數字，即相當於\$8E（\$表示十六進位）。然後將\$8E存入暫存器R0，且將\$00存入暫存器R1，則你將可以得到一個440Hz的頻率。但若真的如此作，則由於你將上述計算中求得的142.04四捨五入，所以你得到的頻率將會是440.14Hz，而不是你所想要的440 Hz頻率。下圖所示就是在不同時鐘脈波頻率下，由計算所得的頻率和實際產生的頻率：

Frequency	1 MHz	1.78977 MHz
1046.496 (C6)	1041.666	1045.428
7040.00 (A8)	6944.444	6991.299

圖2 在不同的時脈頻率下，由PSG晶片產生的頻率及由計算所得頻率之差別

爲了方便起見，列舉下列程式可以用來求出要求產生頻率F（由鍵盤輸入），必需存入暫存器中的十六進位數值，而且同時顯示實際能產生的頻率值：

圖3

LIST

```

10 REM CALCULATING THE CONTENTS OF THE REGISTERS
20 REM FOR THE PSG AY-3-8912
30 REM CLOCKFREQUENCY 1MHZ (FC)
40 REM OUTPUT OF THE 12-BIT VALUES IN HEX
50 REM DESIRED AND TRUE FREQUENCY IS PRINTED
100 INPUT "F=" ;F
110 FC = 1000000 ; 輸入希望產生的頻率 F。
120 TP = FC / (16 * F) ; 設時脈頻率爲 1MHz。
130 MSD = INT (TP / 256)
140 TP = TP - MSD * 256
150 NSD = INT (TP / 16)
160 LSD = INT (TP - NSD * 16 + 0.5)
165 FI = FC / ((MSD * 256 + NSD * 16 + LSD) * 16)
170 GOSUB 200
180 END
200 IF MSD > 9 THEN MSD = MSD + 7
210 MSD = MSD + 48:A$ = CHR$ (MSD)
220 IF NSD > 9 THEN NSD = NSD + 7
230 NSD = NSD + 48:B$ = CHR$ (NSD)
240 IF LSD > 9 THEN LSD = LSD + 7
250 LSD = LSD + 48:C$ = CHR$ (LSD)
260 PRINT F;" ";A$;B$;C$;" ";FI ; 顯示出應存入暫存器的十六進位數字。
270 RETURN ; 及 PSG 實際產生的頻率。

```

下圖顯示如何利用 3.579MHz 石英晶體 (crystal) 來產生時鐘脈波頻率，然後再利用 CMOS 4013 IC 來將時脈加以除頻。對大多

數應用而言，利用這種方式來產生時脈頻率，可能要較直接利用 APPLE 電腦系統的 1MHz 時鐘脈波要有彈性多了。

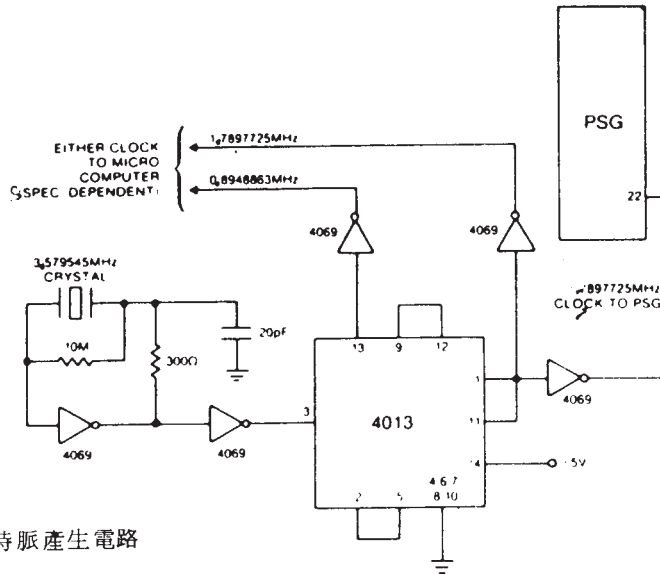


圖 4 時脈產生電路

AY-3-8912 內部暫存器的工作情形

參見圖 1，暫存器 R0-R5 可以利用程式的規劃來控制 A，B，C 三個頻道所產生的聲音周期 (tone period)。暫存器 R6 是用來控制噪音產生器 (Noise Generator)，由圖上可以看到，在控制時可使用此暫存器中 5 個較低次的位元。例如，若要產生較低的噪音頻率，只要將 \$1F 放入 R6 暫存器中。反之若要產生最高的噪音頻率，則只要將 \$01 放入 R6 暫存器中，正如前面曾提到的，輸入的時脈頻率將會首先被除以 16，然後被 R6 暫存器中的較低 5 個位元除頻，噪音周期 (Noise Period) 可以下列公式計算：

$$NP = f_{clock} / (16 * f_n)$$

f_{clock} : 輸入時脈頻率

NP: 噪音周期

f_n : 希望產生的噪音頻率

經由上述公式知，使用一個 1MHz 的時脈頻率，將可以產生一個介於 2MHz-75MHz 間的噪音，暫存器 7 是用以控制各個頻道的聲音

和噪音輸出。

下圖所示是暫存器 R7 用以控制各類道作聲音或噪音輸出的位元配置情形。

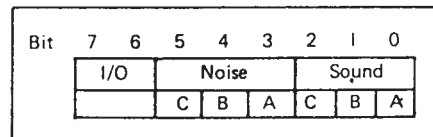


圖 5 PSG 晶片中暫存器 R7 的位元配置

如圖五所示，當暫存器 R7 中的某個位元被設定為 0 時，則對應的 channel 即被打開。

例如：(1) 若圖 5 之 R7 暫存器中存入 \$3E = 0011 1110，則可由 channel A 產生聲音 (SOUND)。

(2) 若存入 \$2A = 0010 1010，則可由 channel B 產生噪音 (Noise) 且由 channel A 和 C 產生聲音 (SOUND)。

在圖 5，R7 暫存器中的最高兩個位元，

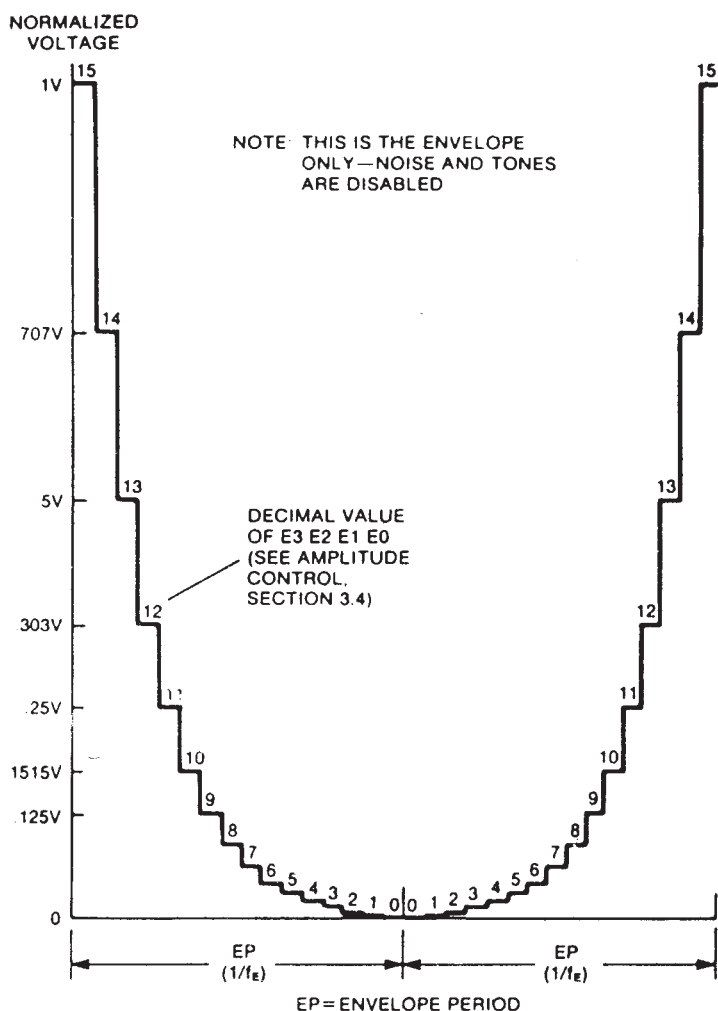


圖 6 包線周期

可以控制 PSG 晶片的輸入／輸出埠（I/O PORT）來作為資料的傳輸。假如你僅使用 PSG 晶片的聲音產生功能，則並不需使用這兩個位元。暫存器 R8, R9, R10 是分別負責 channel A, B, C 的聲音輸出值，這些暫存器的前面 4 個位元，可以將其對應頻道的音調（即輸出值），設定成 16 種電壓位準之一。但必須注意的是，這項設定並不是線性的，而是對數型態的。

假如上面所提到的 R8, R9 及 R10 暫存器

的位元 5 (b₅) 被設定為邏輯 1，則暫存器所對應的那個頻道 (channel) 的振幅是由包線產生器 (envelope generator) 所控制，且包線發生器可以利用暫存器 R11, R12 和 R13 的程式規劃來加以控制。暫存器 R11 和 R10 組成一個 16 位元的計數器 (COUNTER) 以控制包線的周期。時鐘脈波頻率首先被除以 256，然後由暫存器 R11 和 R12 的內容來加以除頻。

由上面的說明知道，利用一個 1MHz 的時鐘脈波頻率，將可以產生 0.06 Hz 至 4000 Hz

的包線周期。若要計算包線周期 (ENVELOPE PERIOD)，可以使用下列公式：

$$EP = f_{clock} / (256 * f_e)$$

f_e = 包線頻率

EP = 包線周期

在這個應用中，EP 的 16 位元二進制是被寫入 R11 和 R12 中，若要求 EP 的值，則只要將圖 3 程式行 120 改成 $EP = FC / (256 * F)$ 即可。

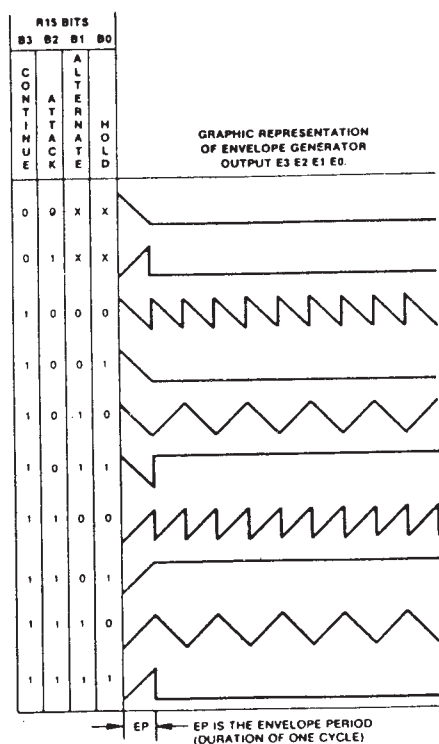
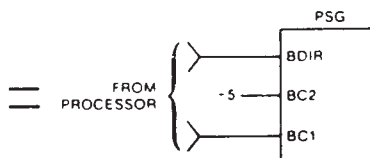


圖 7 包線

BDIR	BC2	BC1	PSG FUNCTION
0	1	0	INACTIVE.
0	1	1	READ FROM PSG.
1	1	0	WRITE TO PSG.
1	1	1	LATCH ADDRESS.



ANALOG CHANNEL A, B, C (outputs): pins 4, 3, 38 (AY-3-8910)
pins 5, 4, 1 (AY-3-8912)

圖 8 PSG 功能控制

如上圖所示的第 2 個波形，是在暫存器 R13 中存入 \$04，而產生一個隨 EP 周期而增加音調的聲音。由圖上知，在 EP 周期的尾端，音調會突然消失。

利用程式控制 PSG 晶片

可程式控制的聲音產生器 - PSG 晶片的 BDIR 及 BC2，是用來控制 PSG 晶片內部暫存器的動作。另一條控制線 BC2 是連到 +5V。有一個很特別的地方，就是 PSG 晶片的資料線或控制線，也可以利用 6522 VIA 來加以控制。

在本文應用中，我們是使用 6502 微處理器的系統時脈 $\phi 2$ 來作為 PSG 晶片的時脈輸入。

資料線 DA0-DA7 是連接到 6522 的埠 A (PORT A)。控制線 BC1 和 BDIR 則裝到 PORT B 之 PB0 及 PB1 去。為了能將資料送入適當的暫存器中，所以必須利用控制線 BDIR 和 BC1 來控制資料線 (DA0-DA7) 接受位址或資料。(見圖 8)

在下列程式中，暫存器的位址信號是被存在 X 暫存器中，而暫存器的資料值則是存在 6502 CPU 的累積器(A)中，然後送到 OUT 的副常式去。(見圖 9)

在此程式執行之同時，PSG 晶片並沒有致能 (Not Enable)。當 6522 VIA 送出 PSG 晶片暫存器的位址信號時，BDIR 及 BC1 將有一段短時間是處於 "H" 的狀態；當資料輸出時，僅有 BDIR 是處於 "H" 的狀態。

PR#1

0800	1	DCM "PR#1"	
0800	2	;	
C0C0	3	ORG \$C0C0	
C0C0	4	TORB EQU *	
C0C0	5	TORA EQU *+!1	
C0C0	6	DDRB EQU *+!2	
C0C0	7	DDRA EQU *+!3	
C0C0	8	;	
0800	9	ORG \$800	; 將暫存器內含值存入 Y 中。
0800 A8	10	OUT TAY	; <A> --> YREG
0801 A9FF	11	LDA #\$FF	; 設定 PORT A 和 B 為輸出。
0803 8DC3C0	12	STA DDRA	
0806 8DC2C0	13	STA DDRB	
0809 8EC1C0	14	STX TORA	; 送出暫存器之位址。
080C A903	15	LDA #\$03	; 設定 PSG 為位址門鎖。
080E 8DC0C0	16	STA TORB	
0811 A900	17	LDA #\$00	; PSG 置於交談型態。
0813 8DC0C0	18	STA TORB	
0816 98	19	TYA	; 將暫存器內含值存入埠 A。
0817 8DC1C0	20	STA TORA	
081A A902	21	LDA #\$02	; 寫入 PSG。
081C 8DC0C0	22	STA TORB	
081F A900	23	LDA #\$00	; PSG 恢復交談型態。
0821 8DC0C0	24	STA TORB	
0824 60	25	RTS	
0825	26	;	
0825	27	TAB EQU \$1000	
0825	28	;	
0825 A200	29	LOAD LDX #\$00	
0827 BD0010	30	M LDA TAB,X	
082A 200008	31	JSR OUT	
082D E8	32	INX	
082E E010	33	CPX #16	
0830 D0F5	34	BNE M	
0832 60	35	RTS	

圖 9 OUT 副常式

除了上面所提到的方法可以用來控制 PSG 晶片外，另外還有一種方法，是將暫存器的內

含值預先放入一個表格中，然後使用一個程式來將這些數值讀入 PSG 晶片的各暫存器中。

截至目前為止，我們所列出的程式祇能影響到 PSG 晶片中的暫存器。若要產生聲音 (SOUND) 或噪音 (NOISE)，你還需加上其它程式，才能完成你的工作。這些程式將包含延遲常式 (delay routines)，以及檢查程序 (checking procedures)。圖 11 中的 WAIT 常式，主要即是用來作為一個延遲迴路。

圖 11 WAIT 常式

0833	36	;	
0833 38	37	WAIT	SEC.
0834 48	38	W2	PHA
0835 E901	39	W3	SBC #\$01
0837 D0FC	40		BNE W3
0839 68	41		PLA
083A E901	42		SBC #\$01
083C D0F6	43		BNE W2
083E 60	44		RTS
083F	45	;	
083F	46	;	

下面所示程式是用以控制 channel A 產生一個最高音調聲音 A 的程式。

```

083F          47 ;
083F A98E      48 LDA #$8E ; 產生 440 Hz 的聲音輸出。
0841 A200      49 LDX #$00
0843 200008    50 JSR OUT
0846 A93E      51 LDA #$3E ; 僅在 CHANNEL A 產生聲音。
0848 A207      52 LDX #7
084A 200008    53 JSR OUT
084D A90F      54 LDA #$0F ; 設定最高音調。
084F A208      55 LDX #8
0851 200008    56 JSR OUT
0854 00        57 BRK
0855          58 ;

```

圖 10 在 CHANNEL A 產生最高音調的聲音 A

圖10所示之程式，主要是利用 channel A，輸出一個接近 1 秒鐘長且頻率為 440 Hz 的聲音，而在下面的警報聲 (SIREN) 程式，將可以產生一個頻率接近 187 Hz 且延續約 1 秒鐘的聲音（假設時鐘頻率是 1MHz）。

```

0855          59 ;
0855          60 ;
0855 A93E      61 SIREN LDA #$3E ; 僅由 CHANNEL A 產生。
0857 A207      62 LDX #7
0859 200008    63 JSR OUT
085C A90F      64 LDA #$0F ; 設定最高音調。
085E A208      65 LDX #8
0860 200008    66 JSR OUT
0863 A98E      67 S LDA #$8E ; 440 HZ
0865 A200      68 LDX #$00
0867 200008    69 JSR OUT
086A A900      70 LDA #$00
086C A201      71 LDX #01
086E 200008    72 JSR OUT
0871 A9FF      73 LDA #$FF
0873 203308    74 JSR WAIT ; 延遲 350 MS
0876 A901      75 LDA #$01 ; 187 HZ
0878 A201      76 LDX #$01
087A 200008    77 JSR OUT
087D A94E      78 LDA #$4E
087F A200      79 LDX #$00
0881 200008    80 JSR OUT
0884 A9FF      81 LDA #$FF
0886 203308    82 JSR WAIT
0889 18        83 CLC
088A 90D7      84 BCC S
088C          85 ;

```

圖 12 SIREN 程式

用程式來產生鎗聲

若要模擬鎗聲，則你僅需使用噪音產生器來控制包線。我們在記憶體中預先存放一個表格的資料，且假如按鍵被壓下，則預先存放在

表格中的資料即被送入 PSG 晶片中。假如你將位址 \$1006 的內容（噪音頻率）更改為 \$00（最高噪音周期），且將位址 \$100C 的內容更改成 \$40（envelope 近似於 2 秒鐘），則你將可以模擬一個爆炸聲。

088C	86	;		
088C	87	KEY	EQU	\$FD35
088C	88	;		
088C	202508	89	SHOT	JSR LOAD
088F	2035FD	90		JSR KEY
0892	18	91		CLC
0893	90F7	92		BCC SHOT
0895		93	;	
0895		94	;	
1000		95	ORG	\$1000
1000	000000	96	HEX	00000000000000
1003	000000			; 沒有聲音。
1006	0F	97	HEX	0F
1007	07	98	HEX	07
1008	101010	99	HEX	101010
100B	0010	100	HEX	0010
100D	00	101	HEX	00
	102		END	

; 中等的噪音頻率。
; 所有頻道設定為噪音輸出。
; 音量設定為最大。
; 包線周期為 0.6s。
; 僅有一個周期。

圖 13 GUNSHOT 程式

下面所示是上面提到的各示範程式之十六進位起始位址：

\$083F ... SOUND 程式

\$0855 ... SIREN 程式

\$088C ... GUNSHOT 程式

```

0800- A8 A9 FF 8D C3 C0 8D C2
0808- C0 8E C1 C0 A9 03 8D C0
0810- C0 A9 00 8D C0 C0 98 8D
0818- C1 C0 A9 02 8D C0 C0 A9
0820- 00 8D C0 C0 60 A2 00 BD
0828- 00 10 20 00 08 E8 E0 10
0830- D0 F5 60 38 48 E9 01 D0
0838- FC 68 E9 01 D0 F6 60 A9
0840- 8E A2 00 20 00 08 A9 3E
0848- A2 07 20 00 08 A9 0F A2
0850- 08 20 00 08 00 A9 3E A2
0858- 07 20 00 08 A9 0F A2 08
0860- 20 00 08 A9 8E A2 00 20
0868- 00 08 A9 00 A2 01 20 00
0870- 08 A9 FF 20 33 08 A9 01
0878- A2 01 20 00 08 A9 4E A2
    
```

```

0880- 00 20 00 08 A9 FF 20 33
0888- 08 18 90 D7 20 25 08 20
0890- 35 FD 18 90 F7 90
*
    
```

11000.100D

```

1000- 00 00 00 00 00 00 0F 07
1008- 10 10 10 00 10 00
*
    
```

以程式來產生鋼琴聲

剛才已列舉了各種聲音產生方法，但却都是以副程式的方法，個別的介紹，為了讓你有個整體的觀念，現在就列舉一個可產生鋼琴聲的 PIANO 程式。這個程式的表格是被存放在 \$1010 至 \$1017 處。而程式是由位址 \$0900 開始，且在程式中也使用了 OUT, LOAD 和 KEY 等幾個副常式。

圖 14 PIANO 程式

0800	1		DCM "PR#1"	
0800	2	;		
C0C0	3		ORG \$C0C0	
C0C0	4	TORB	EQU *	
C0C0	5	TORA	EQU *+!1	
C0C0	6	DDRB	EQU *+!2	
C0C0	7	DDRA	EQU *+!3	
C0C0	8	;		
C0C0	9	KEY	EQU \$FD35	
C0C0	10	;		
0800	11		ORG \$800	
0800 A8	12	OUT	TAY	; 將暫存器內含值由累積執暫存到 Y。
0801 A9FF	13		LDA #\$FF	; 設定埠 A 和 B 為輸出。
0803 8DC3C0	14		STA DDRA	
0806 8DC2C0	15		STA DDRB	
0809 8EC1C0	16		STX TORA	; 送出暫存器之位址選擇信號。
080C A903	17		LDA #\$03	; 設定 PSG 為位址門鎖型態 (BDIR
080E 8DC0C0	18		STA TORB	和 BC1 = 1)。
0811 A900	19		LDA #\$00	; 恢復 PSG 為交談型態 (BDIR 和
0813 8DC0C0	20		STA TORA	BC1 = 0)。
0816 98	21		TYA	; 將暫存器內含值存入埠 A。
0817 8DC1C0	22		STA TORA	
081A A902	23		LDA #\$02	; 設定 PSG 為寫入型態。
081C 8DC0C0	24		STA TORB	
081F A900	25		LDA #\$00	; 恢復 PSG 為交談型態 (BDIR 和
0821 8DC0C0	26		STA TORB	BC1 = 0)。
0824 60	27		RTS	
0825	28	;		
0825	29	;		
0825 A200	30	LOAD	LDX #\$00	
0827 BD5B08	31	M	LDA TAB,X	
082A 200008	32		JSR OUT	
082D E8	33		INX	
082E E010	34		CPX #16	
0830 D0F5	35		BNE M	
0832 60	36		RTS	
0833	37	;		
0833 38	38	WAIT	SEC	
0834 48	39	W2	PHA	
0835 E901	40	W3	SBC #\$01	
0837 D0FC	41		BNE W3	
0839 68	42		PLA	
083A E901	43		SBC #\$01	
083C D0F6	44		BNE W2	
083E 60	45		RTS	
083F	46	;		
083F	47	;		
083F 2035FD	48	PIANO	JSR KEY	
0842 290F	49		AND #\$0F	
0844 AA	50		TAX	
0845 CA	51		DEX	

```

0846 BD6B08    52          LDA FTAB,X
0849 8D5B08    53          STA TAB
084C AA        54          TAX
084D CA        55          DEX
084E 8E5D08    56          STX TAB+2
0851 4A        57          LSR
0852 8D5F08    58          STA TAB+4
0855 202508    59          JSR LOAD
0858 4C3F08    60          JMP PIANO
085B          61          ;
085B          62          ;
085B 000000    63  TAB      HEX 00000000000000; 這 6 個保留 Byte 是由程式來存
085E 000000                                放資料以決定 CHANNEL A,B,
0861 0038      64          HEX 0038                                C 之輸出聲音。
0863 101010    65          HEX 101010                                ; 將所有頻道設定為聲音輸出型態。
0866 000A00    66          HEX 000A00                                ; 設定為最大音量輸出。
0869 0000      67          HEX 0000                                ; 包線衰減 0.8s。
086B EFD5BE    68  FTAB     HEX EFD5BEB39F8E7F75 ; 頻率表。
086E B39F8E
0871 7F75

                69  FIN      END

```

```

0800- A8 A9 FF 8D C3 C0 8D C2
0808- C0 8E C1 C0 A9 03 8D C0
0810- C0 A9 00 8D C0 C0 98 8D
0818- C1 C0 A9 02 8D C0 C0 A9
0820- 00 8D C0 C0 60 A2 00 BD
0828- 5B 08 20 00 08 E8 E0 10
0830- D0 F5 60 38 48 E9 01 D0
0838- FC 68 E9 01 D0 F6 60 20
0840- 35 FD 29 0F AA CA BD 6B
0848- 08 8D 5B 08 AA CA 8E 5D
0850- 08 4A 8D 5F 08 20 25 08
0858- 4C 3F 08 00 00 00 00 00
0860- 00 00 38 10 10 10 00 0A
0868- 00 00 00 00 EF D5 BE B3 9F
0870- 8E 7F 75

```

AY-3-8912 的聲音示範

下面要介紹的這個程式，主要是告訴你如何利用 BASIC 程式控制由 GI 所推出的 PSG 晶片 (AY-3-8912) 中的各個暫存器。在這個程式中，暫存器 R0-R13 的內含值是被存放在資料指述 (DATA STATEMENTS) 中。若你注意的觀察一下這個程式，你將會發現此程式中含有機械語言常式 (用 POKE 指述存入電腦中)，它主要是用來提供脈波，以便能將資料送到 PSG 晶片去。要注意的是，此處之所以

採用機械語言常式來作這個工作，主要是因為在設計程式時，發現若利用 BASIC 的 POKE 命令來產生脈波，則由於速度太慢，導致 AY-3-8912 IC 產生不可預期的結果。

程式說明：(見圖 15)

行 100-150 存入 (POKEING) 機械語言常式。

行 200 設定資料方向暫存器。

行 210-330 等待迴路及讀取資料。

行 1000-1040 利用機械語言常式將 D(A) 資料存入暫存器中。

行 2000-2040 產生不同聲音的資料。

聲音產生界面的組裝

爲了要完成聲音產生界面的組裝，首先必須組裝在前面所介紹過的 APPLE 實用專題製作中，所組裝過的 6522 VIA 界面板，然後利用界面板上左方的空間，來組裝本文所介紹的聲生電路。

在組裝時先將 AY-3-8912 IC 安裝到界面板上，但必須使 IC 的資料線 (DA0-DA7) 和 6522 VIA 晶片的 PA0-PA7 接腳相連接

圖 15 BASIC 聲音示範

```
100 POKE 687,169: POKE 688,3
110 POKE 689,141: POKE 690,192: POKE 691,192
120 POKE 692,169: POKE 693,0
130 POKE 694,141: POKE 695,192: POKE 696,192
140 POKE 697,96
150 POKE - 16190,255: POKE - 16189,255
200 DIM D(14)
210 HOME : HTAB (3): VTAB (5)
220 PRINT "SOUND DEMO"
230 FOR X = 1 TO 3000: NEXT
240 READ G$
250 HTAB (3): PRINT G$
260 GOSUB 500
270 FOR X = 1 TO 5000: NEXT
280 IF G$ = "SUEF" THEN FOR X = 1 TO 10000: NEXT
290 Y = Y + 1: IF Y < 5 THEN 320
300 A = 7:D(A) = 255: GOSUB 1000
310 Y = 0: RESTORE : GOTO 210
320 A = 7:D(A) = 255: GOSUB 1000
330 GOTO 240
500 FOR A = 0 TO 13
510 READ D(A)
520 GOSUB 1000
530 NEXT A
540 RETURN
1000 POKE - 16192,0: POKE - 16191,A
1010 POKE 688,3: CALL 687
1020 POKE - 16192,0: POKE - 16191,D(A)
1030 POKE 688,2: CALL 687
1040 RETURN
2000 DATA "PIANO",200,0,201,0,100,0,0,248,16,16,16,0,20,8
2010 DATA "EXPLOSION",0,0,0,0,0,0,31,7,16,16,16,0,20,0
2020 DATA "GUNSHOT",0,0,0,0,0,0,15,7,16,16,16,0,16,0
2030 DATA "LOCOMOTIVE",0,0,0,0,0,0,15,199,16,16,16,180,2,12
2040 DATA "SURF",0,0,0,0,0,0,31,199,16,16,16,16,255,60,14
```

(參見圖 16)，接著將 AY-3-8912 IC 連接到 6522 VIA 的 PB0-PB3 (4 條線) 切開。AY-3-8912 IC 的其它接腳如下所述：腳 20 接到 6522 的腳 10；腳 19 接到 +5V；腳 18 接到 6522 腳 1；腳 17 接到 +5V；腳 16 接到 6522 的腳 34；腳 15 接到 6522 腳 25；腳 6 接地且腳 3 接到 +5V。

參見圖 16 AY-3-8912 晶片的腳 1，4 和 5 是同一點輸出。你可以視實際需要將它們接到 P C 板上的銲接點位置。然後由銲接點接 1 個 1K Ω 電阻至地。然後由此點連接 10K Ω

電阻且串聯 1 個 100 μ F 電容器，以連接到你的音頻放大器去。在 6522 VIA 晶片，要將腳 24 及 20 用跳線連接，另外在 P C 板的零件面，必須有跳線，以便將 +5V 電源由銲接面連接到零件面來。

在本文中，我們配合以前製作過的 6522 VIA 界面板，加上可程式控制的 PSG 晶片，6522 VIA 的程式控制而組合成一個語音界面板。雖然文中亦舉了許多示範例，但或許還不能滿足你的好奇心；現在不妨發揮你的想像力及創造力，好好的寫一段程式，然後加上

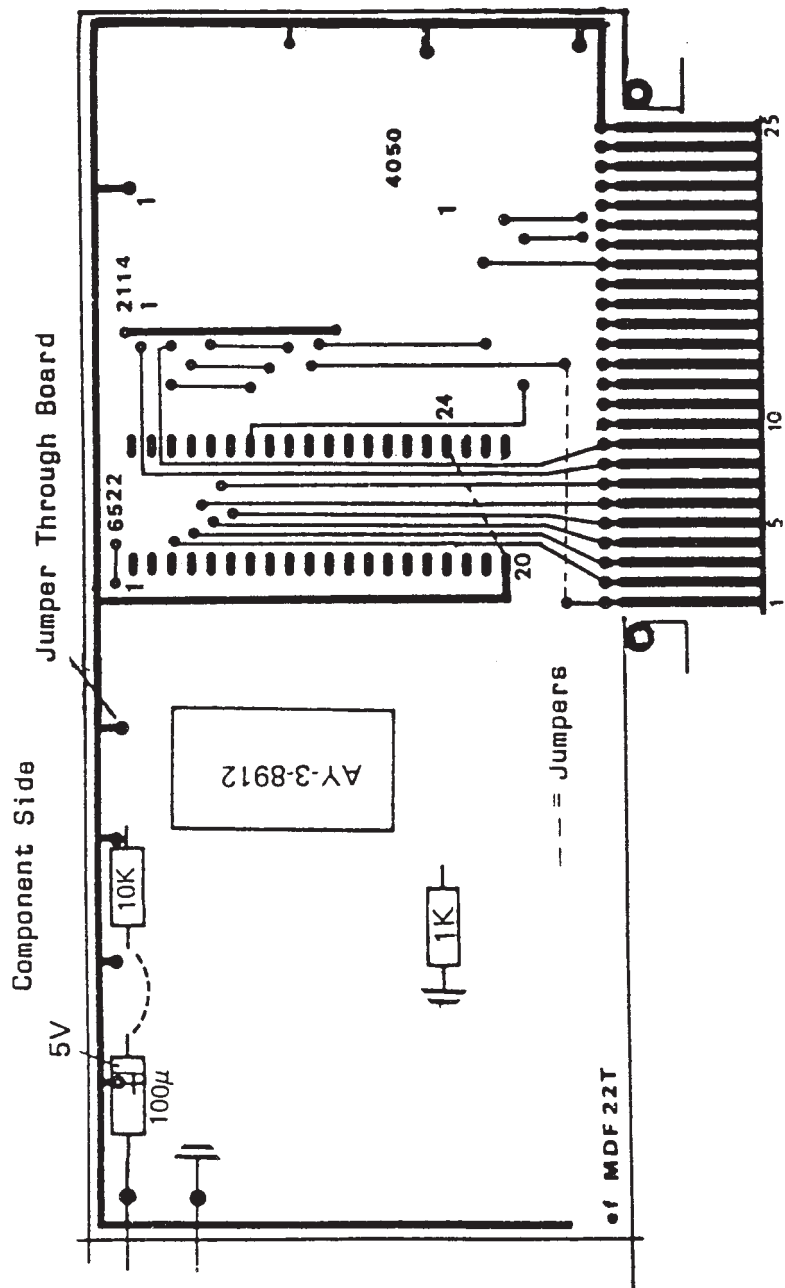


圖 17 聲音產生界面板佈線圖

本文介紹的語音介面板，使你的APPLE 開口說話吧！

編輯部後記：有關於通用之AY-3-8912

IC，市面上有售，讀者若製作而無法購得此

IC，可以打電話到通用器材公司微電子業務

部詢問。電話：9146234。◆

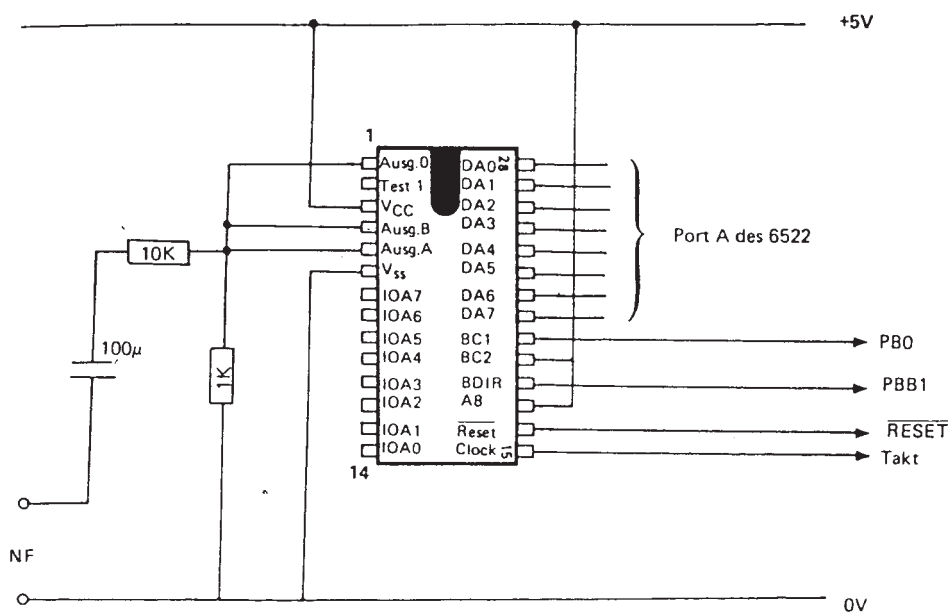


圖 16 聲音產生界面板上 PSG 晶片之接線情形

使APPLE II 能成為軟體和硬體測試器的介面

ICE界面卡原理與應用

加上 ICE (In-Circuit Emulation 電路內模擬) 介面卡以後，Apple II 便可以變成一套測試系統，用來測試 6502 微處理系統（或與 6502 通用的系統）的硬體和軟體。

模擬器一般都是與昂貴的微處理器發展用系統連結。然而，只要利用一個簡單的線路，

就可以使 Apple II 具有模擬器的功能，而使得 Apple II 可以用來測試 6502 系統（或與 6502 通用的系統）的硬體和軟體。

Apple II 的電路內模擬器（In-Circuit Emulation），是整個做在一片介面板上，然後利用一條 40 線的 Cable 連接到欲測試的微

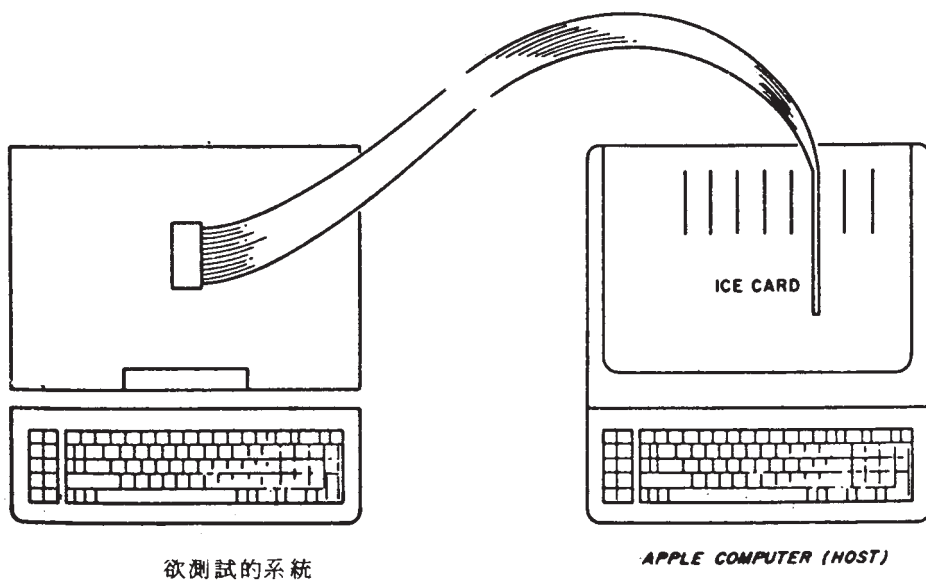


圖1 利用一條40線的 cable 把欲測試的系統連接到 ICE 介面卡上

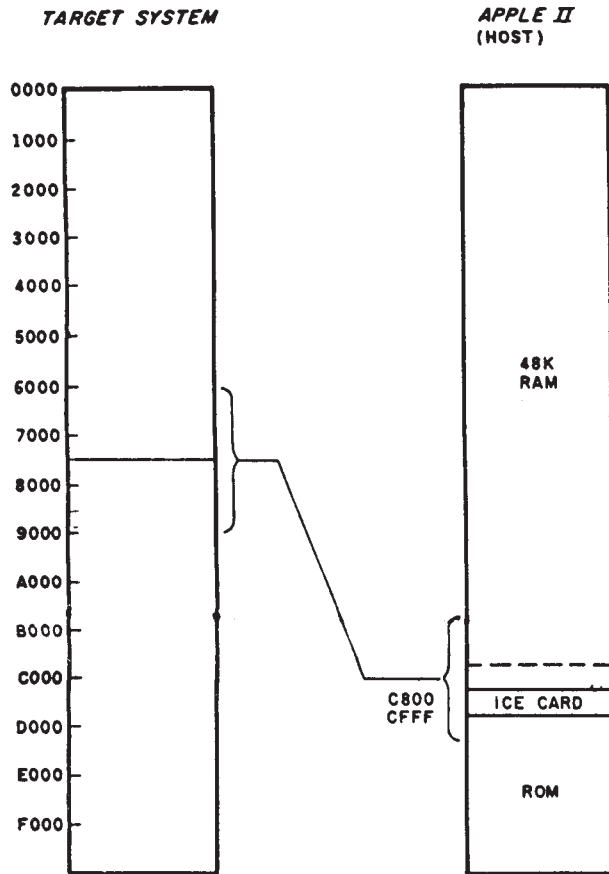


圖 2 欲測試系統上的任何 2K-byte 的記憶空間，都可以對應到平常為空白的由 C800 至 CFFF 的 Apple 記憶區

電腦上（如圖 1）。

利用 ICE 介面卡，Apple 可以把欲測試系統內的任意 2K byte 的位址空間，重新定位到 Apple 本身內平常都是空白記憶區的 C800 到 CFFF（參考圖 2）。至於所要測試的受測系統記憶區，使用者可以用軟體程式來選擇，也就是說，只要使用者在 Apple 上寫一個測試程式，那就可以測試到整個受測系統內的全部記憶區。

測試程式可以用高階語言來寫（如 Basic），或是用機器語言來寫，同時也可以針對所要測試的部份（如系統內的 bus，RAM，ROM 和 I/O 元件等等）來設計這個測試程式。

用 Basic 寫成的測試程式，雖然只是做個簡單的測試，但是仍然需要一段很長的時間，所以較有效率的方法是用機器語言來寫測試程式，然後再用 Basic 來寫全盤的測試方法，測試的順序和當錯誤發生時，建議使用者該如何

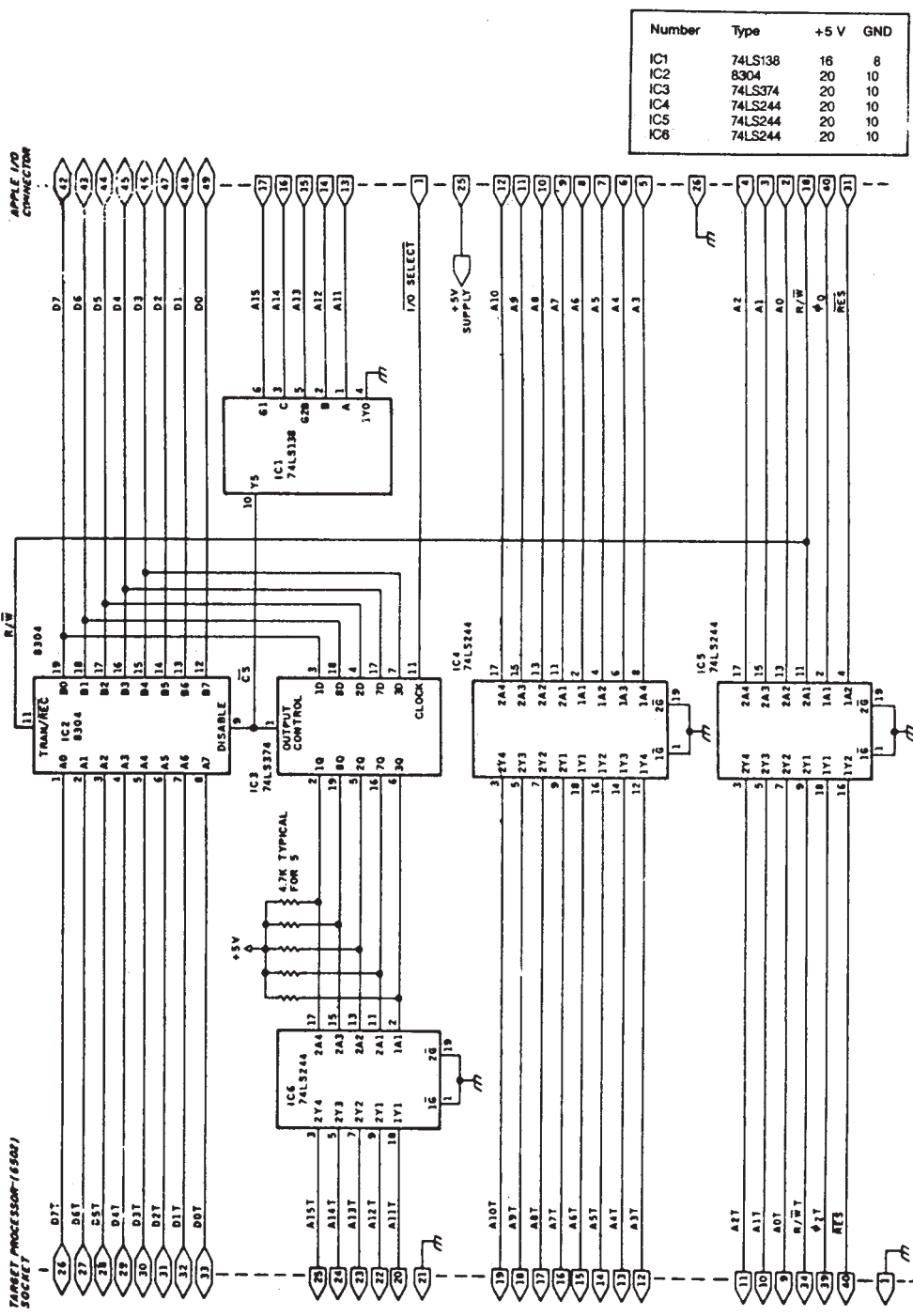


圖 3 ICE 介面卡的線路方塊圖

來處理等等的這些程式。

本文將會介紹如何來測試系統的 bus，RAM 和 ROM，最後並以一個實例，介紹如何利用 ICE 來測試 AIM-65 電腦。

ICE 硬體

圖 3 是 ICE 介面卡的線路方塊圖。Apple II 的位址線 A0 ~ A10 及控制線 $R/\overline{W}$ ， ϕ_0 和 $\overline{RES}$ ，直接經由 IC4 及 IC5 這兩個驅動器連接到受測系統的 A0_T ~ A10_T 及控制線 $R/\overline{W}_T$ ， ϕ_{2T} 和 $\overline{RES}$ ，然而受測系統的位址線 A11 ~ A15，却不是直接由 Apple II 連接過來，而是連接到記憶區選擇器 IC3，所以只要我們事先把控制資料設定到 IC3 上，那我們就可以選定要測試的受測系統記憶區。假設 ICE 卡是插在插座 5 上，則利用下面這段簡短的程式，就可以把受測系統的 A11 ~ A15 設定為 0：

LDA # \$00 : 把 A11 ~ A15 全部設定為 0

STA \$C500 : 使插座 5 的 I/O $\overline{SELECT}$ 線輸出為 LOW

這時候，只要 Apple 本身對位址 C800 至 CFFF 有任何的讀寫動作，都會使 IC1 的輸出為 LOW，而使 IC3 動作，也就是說我們可以測試受測系統內 0000 ~ 07FF 的記憶區。

IC1 是位址解碼器，用來控制 IC2 和 IC

3 的動作。IC1 看起來似乎是不需要用到，因為當位址在 C800 至 CFFF 之間時， $\overline{I/O STROBE}$ 同樣都是位於低電位狀態。但是 $\overline{I/O STROBE}$ 時脈 (timing) 的低電位狀態在時脈週期 (timing cycle) 內出現得太遲了，無法使受測系統內的記憶體或 I/O 元件動作，所以我們仍然需要用到 IC1 (參考圖 3) 的第 10 腳送出的 $\overline{CS}$ 信號，其時脈週期如圖 4 所示。

測試用的軟體

測試用的軟體可以分成兩大類：(1) 用來測試受測系統內不同功能區 (如系統的匯流排 (bus)、RAM、ROM 和 I/O 元件) 的功能測試程式，(2) 在測試的過程中，指引使用者、呼叫 (CALL) 功能測試程式 (也就是指出錯誤的地方和提供使用者適當的補救方法一例如，“換 IC28” 等等) 的全面測試程式。

下面我們就來描述一下幾種不同的功能測試。

位址匯流排和資料匯流排的反相測試

在進入複雜的系統 IC 測試之前，我們先測試一下系統的匯流排。利用反相測試，使用者可以使位址和資料匯流排的排線信號，“1

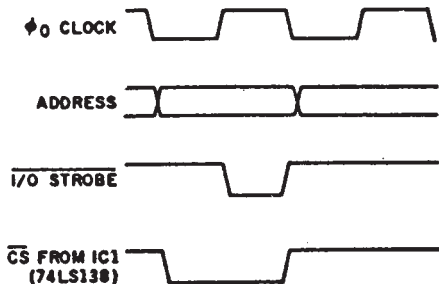


圖 4 Apple 插座上的 I/O STROBE 信號和由 IC1 接出來的 CS 信號間的相對時脈圖。很明顯的 I/O STROBE 信號出現得太遲了無法用來選擇受測系統的位址

”，“0”這樣來回變換著。列表1是這個測試所用的程式，首先要選擇受測系統的位址為10101……的型式，然後Apple再對位址\$C AAA做一個讀的動作，這樣便可以使受測系統的位址匯流排排線信號，成為“H”，“L”，“H”，“L”的型式。接著，選擇受測系統的位址為01010……的型式，然後Apple再對位址\$CD55做一個讀的動作，這樣便可以使受測系統的位址匯流排排線信號，成為“L”，“H”，“L”，“H”的型式，這個程序一直重複256次，然後再以類似的方法，測資料匯流排。操作人員可以利用示波器，或是邏輯探針，來觀察是否每一條信號線都是可

驅動的（drivable），以及是否每條信號線都由處理器插座連接到各個IC上。

ICE也可以用來測試線與線之間是否有發生短路的情形，只要在每條信號線上加入一特有的頻率或型式，然後利用頻率表或示波器來觀察，便可以檢查出各信號線間是否有短路的情形存在。

列表一的匯流排反相測試，雖然重複了256次，但是所花的時間，仍然很短，測試者甚至還未檢查好一個線路接點，程式就結束了，下面這段簡短的程式，就是要告訴使用者如何來用列表一這個測試程式。

180 PRINT "BUS TESTING —

列表1 位址匯流排和資料匯流排的反相測試

SOURCE FILE: APPTOG.			
0000:	1	;	
0000:	2	;	ADDRESS AND DATA BUS TEST
0000:	3	;	TOGGLE ADDRESS BUS AAAA-5555
0000:	4	;	256 TIMES
0000:	5	;	TOGGLE DATA BUS AA-55
0000:	6	;	256 TIMES
0000:	7	;	
0000:	8	;	
CS00:	9	SELECT	BQU \$CS00 ; 2 K 的記憶區選擇
NEXT OBJECT FILE NAME IS APPTOG.OBJ0			
2100:	10		
2100:A2 00	11	ORG	\$2100
2102:	12	LDX	#00 ; 將計數器設定為 0
2102:	12	;	EXERCISE ADDRESS BUS
2102:A9 AA	13	LDA	#5AA ; 選擇位址 10101×××
2104:8D 00 CS	14	STA	SELECT
2107:AD AA CA	15	LDA	\$CAAA ; 使受測系統的位址線信號為 AAAA
210A:A9 55	16	LDA	#555 ; 選擇位址 01010×××
210C:8D 00 CS	17	STA	SELECT
210F:AD 55 CD	18	LDA	\$CD55 ; 使受測系統的位址信號為 5555
2112:CA	19	DEX	
2113:D0 ED	20	BNE	ABUS ; 重複 256 次
2115:	21	;	EXERCISE DATA BUS
2115:A9 00	22	DBUS	LDA #500 ; 選擇位址 00000×××
2117:8D 00 CS	23	STA	SELECT
2117:A9 55	24	LDA	#555 ; 使受測系統的資料線信號為 55
211C:8D 00 CS	25	STA	\$C900 ; 使受測系統的位址線信號為 0100
211F:A9 AA	26	LDA	#5AA ; 使受測系統的資料線信號為 AA
2121:8D 00 CS	27	STA	\$C900 ; 使受測系統的位址線信號為 0100
2124:CA	28	DEX	
2125:D0 EE	29	BNE	DBUS ; 重複 256 次
2127:80	30	RTS	ABUS ; 反相測試完畢
*** SUCCESSFUL ASSEMBLY: NO ERRORS			
2102 ABUS	2115 DBUS	CS00 SELECT	2102 ABUS
2115 DBUS	CS00 SELECT		

```

190 PRINT "(PRESS SPACE FOR
    NEXT TEXT)"
200 CALL BTST
210 IF PEEK(-16384) <= 127
    THEN 200

```

且列表一的測試程式會一直不斷的執行，直到按下空白鍵為止。

測試 RAM 的基本方法就是：把測試資料寫入記憶體內，然後再由記憶體內讀回資料，這樣便可以檢查出，是否對 RAM 的讀寫動作都沒有問題。有很多不同的漏試資料型式都可以

```

SOURCE FILE: APPRAM
0000: 1 ; .....
0000: 2 ;PROGRAM TO CHECKERBOARD TEST RAM
0000: 3 ; (C800-CFFF)
0000: 4 ; .....
0000: 5 ;
0000: 6 COUT EQU $FDED ; 輸出一個字到螢幕上
0000: 7 CROUT EQU $FD8E ; 游標回頭
0000: 8 PRHEX EQU $FDE3 ; 輸出 16 進位數字
0000: 9 POINT EQU 08 ; 位置指示器
0000: 10 ;
----- NEXT OBJECT FILE NAME IS APPRAM.OBJO
2090: 11 ORG $2090
2090: 12
2090:A9 00 13 LDA #00 ; 指到 C800
2092:85 08 14 STA POINT
2094:A8 15 TAY
2095:A9 C8 16 LDA #$C8
2097:85 09 17 STA POINT+1
2099:A9 55 18 START LDA #$55 ; 先用 55 測試
209B:91 08 19 STA (POINT),Y ; 存
209D:D1 08 20 CMP (POINT),Y ; 讀回並比較
209F:F0 03 21 BEQ OK
20A1:4C BC 20 22 JMP ERROR ; 顯示錯誤並結束程式
20A4:A9 AA 23 OK LDA #$AA ; 換用 AA 測試
20A6:91 08 24 STA (POINT),Y ; 存
20A8:D1 08 25 CMP (POINT),Y ; 讀回並比較
20AA:F0 03 26 BEQ OKI
20AC:4C BC 20 27 JMP ERROR ; 顯示錯誤並結束程式
20AF:E8 08 28 OKI INC POINT ; 下一個位置
20B1:D0 E6 29 BNE START
20B3:B5 09 30 INC POINT+1
20B5:A5 09 31 LDA POINT+1
20B7:C9 D0 32 CMP #5D0
20B9:D0 DE 33 BNE START ; 判斷記憶體區是否已結束 (CFFF)
20BB:60 34 RTS ; 測試完成
20BC: 35 ;
20BC: 36 ;ERROR DISPLAY ROUTINE
20BC: 37 ;
20BC:A2 00 38 ERROR LDX #00 ; 訊息的指示器
20BE:BD D5 20 39 NEXTI LDA MESS,X
20C1:20 ED FD 40 JSR COUT ; 把訊息顯示到螢幕上
20C4:E8 41 INX ; 下一個字
20C5:E0 0F 42 CPX #50F ; 判斷訊息是否已輸出完畢
20C7:D0 F5 43 BNE NEXTI
20C9:A5 09 44 LDA 09 ; 發生錯誤的位址
20CB:38 45 SEC
20CC:E9 C8 46 SBC #$C8
20CE:20 E3 FD 47 JSR PRHEX
20D1:20 8E FD 48 JSR CROUT ; 游標回頭
20D4:60 49 RTS ; 結束
20D8: 50 ;ERROR MESSAGE
20D8:A0 C5 D2 51 MESS ASC
20D8:D2 CF D2
20D8:A0 CF CE
20DE:A0 D0 C1
20E1:C7 C5 A0

```

利用，每種型式都對某種特別的記憶體錯誤的測試，特別有效。一種較為普遍的測試資料型式就是棋盤式（checkboard）的測試方式（參考圖 5）。

	00	01	10	11
00	0	1	0	1
01	1	0	1	0
10	x	x	x	x
11	x	x	x	x

圖 5：棋盤測試的 RAM。將間隔的位元（bit）都設定為“1”和“0”，並且加以測試。這種型式一直維持到要測試下一列（row）的各位元為止。

列表 2 是這個 RAM 測試的程式。先選定一個記憶位置，然後將 55（16 進位）存到這個選定的記憶位置，並加以比較。如果比較時發現了錯誤，那麼這個測試便結束了，並且會在 Apple 螢幕上顯示出一個 RAM 錯誤的訊息。如果比較無誤，那就將測試資料 55 改成 AA，然後再測試一次。

在呼叫（call）主測試程式中的 RAM 測試程式之前，必須先利用記憶區選擇器來選定要測試的記憶區。例如，下面這段程式可以用來測試受測系統中的 RAM，由 0800 至 0FFF。

```

250 PRINT "RAM TESTING 0800 ~
      0FFF
260 POKE SELECT,08:CALL RAM-
      TEST
  
```

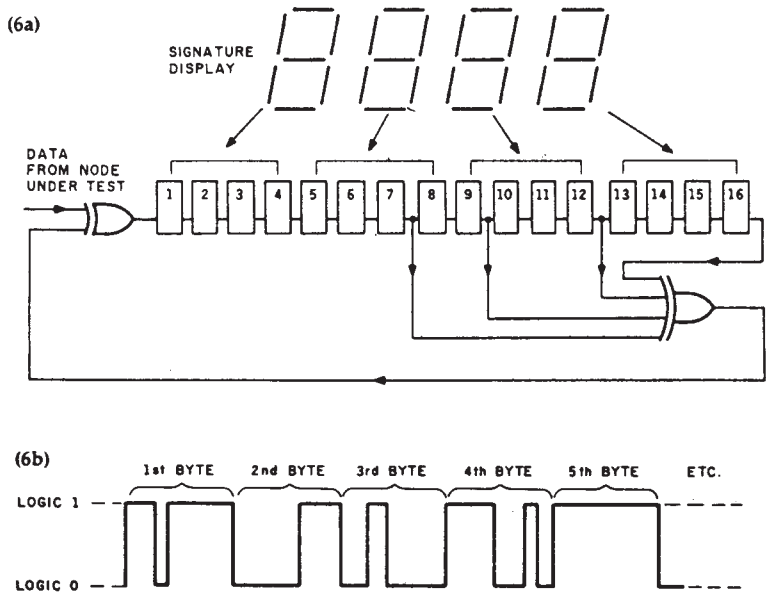


圖 6 a 利用一個具有回授的移位暫存器，對 ROM 做出一個循環費餘核對
 圖 6 b ROM 的資料一個 bit；一個 bit 的輸入移位暫存器內

測試ROM的方法，一般是利用ROM內的資料，做出一個核對和（checksum），然而，如果有許多個可以互相抵消對「核對和」的影響的錯誤發生，那用「核對和」的方法，就偵測不出來了。有一種較有效的測試技巧，就是對ROM內的資料，做一個「循環贅餘核對」（CRC—Cyclic Redundancy Check），這種技巧最先用於數據通訊（data communication），但最近已應用於信號分析方面了。

CRC可以用硬體或軟體的方法來實現，圖6是利用具有回授（feedback）的16位元線性的移位暫存器，所做成的典型CRC線路。資料依照順序一個位元、一個位元的移入移位暫存器內，當資料全部移入移位暫存器以後，最後遺留在移位暫存器內的資料，就做成了一個4位的「循環贅餘核對」。

列表3是CRC的軟體常式，在這個設計裏面，要測試的ROM的位元組（byte）資料（位元0至位元7），依照順序加到FEEDBACK這個副程式內，然後利用暫存器內的位元15，11，8，6資料及外界進來的資料，做成一個和。當16384（2K×8）個位元（bit）的資料，全部都移進這個回授的程序以後，留在記憶位置SIGH和SIGL內的資料，就成了最後的信號。

爲了要能夠檢查不只2K的ROM，所以在此使用了三個常式。NSIG（new signature 新的信號）常式可以重置（reset）暫存器組SIGH，SIGL爲零，然後把2K的ROM所形成的信號儲存在SIGH及SIGL內。CSIG（continue signature 繼續信號）這個常式，不會重置SIGH及SIGL爲零，所以可以用來繼續測試大於2K的ROM。DISPLAY常式可以用16進制型式顯示出暫存器組SIGH，SIGL的內容。

下面這段程式是說明如何使用上述三個常式，來做出一個由受測系統內，位址B000至BFFF這4K的ROM，所形成的信號：

```

320 REM B0 HEX IS 176 DECIMAL
MAL
330 POKE SELECT,176
340 CALL NSIG:REM FIRST 2K
BYTES
350 REM B8 HEX IS 184 DECIMAL
360 POKE SELECT,184
370 CALL CSIG:REM CONTINUE
WITH NEXT 2 BYTES
380 CALL DISPLAY:REM DISPLAY
FINAL SIGNATURE

```

實現測試程式

這裏所描述的Apple ICE 介面卡，可以廣泛的應用於6500系列的微電腦，只要這些電腦的設計執行速度是1MHz，並且所有的裝置電路部分，都是利用微處理器的 ϕ_2 計時脈衝來控制的。AIM-65是一個理想的受測系統，它包含有4K的RAM，20K的ROM和應用範圍廣泛的輸入／輸出元件——兩個6522 VIA（versatile interface adapter 多用途的介面裝置），一個6520 PIA（peripheral interface adapter 週邊介面裝置）和一個6532 RIOT（RAM input/output timer）。圖7是AIM-65的方塊圖和記憶分配圖。表4是AIM-65的測試程式，這個程式先測試系統的匯流排，排著測試4K的RAM，最後再測試ROM。這個測試程式同時也測試6522 VIA，測試VIA時，必須將PA0與PB0，PA1與PB1用硬配線（hard-wired）連接起來。測試VIA的常式首先將A當成輸入埠，B當成輸出埠來使用，然後由A埠輸入測試資料，再由B埠讀出輸出資料，接著比較輸入與輸出的資料。將A改成輸出埠，B改成輸入埠，重複上述測試。利用上述這些技巧，使用者可以自己寫一個更詳細的測試程式，用來測試其餘的輸入／輸出設備，例如顯示幕、印表機和鍵盤等等。

列表 3 POM的測試程式

```

SOURCE FILE: APPSIG
0000: 1 .....
0000: 2 ; PROGRAM TO EVALUATE SIGNATURE
0000: 3 ; OF 2KBYTE BLOCK (C800-CFFF)
0000: 4 ; EACH BYTE IS SERIALIZED BIT0-BIT7
0000: 5 .....
0000: 6 ;
0000: 7 ;
0000: 8 ;
0000: 9
----- NEXT OBJECT FILE NAME IS APPSIG.OBJO
2000: 10 COUNT ORG $2000
1900: 11 SIGL EQU $1900 ; SUM的儲存位置
1901: 12 SIGH EQU $1901 ; 信號低位元組
1902: 13 POINT EQU $1902 ; 信號高位元組
0008: 14 TEMP EQU $0008 ; 位元組計數
J903: 15 PRBYTE EQU SIGH+1 ; 暫存位置
FDDA: 16 CROUT EQU $FDDA ; 印出一個 16 進位元組
FD8E: 17 ; 游標回頭
2000: 18 START LDA #00 ; 把移位暫存器設為 0
2002:8D 01 19 STA SIGL
2005:8D 02 19 STA SIGH
2008:A9 00 21 WSTART LDA #00 ; WARM START
200A:85 08 22 STA POINT
200C:A8 23 TAY
200D:A9 C8 24 LDA #C8
200F:85 09 25 STA POINT+1
2011:B1 08 26 NBYTE LDA (POINT),Y
2013:8D 03 19 27 STA TEMP
2016:A2 08 28 LDX #08 ; 8 個位元
2018:AD 03 19 29 NBIT LDA TEMP
201B:29 01 30 AND #01 ; 位元 0 進入 count
2010:8D 00 19 31 STA COUNT
2020:20 36 20 32 ISR FEEDBACK ; 回授
2023:6E 03 19 33 ROR TEMP ; 下一個位元
2026:CA 34 DEX
2027:D0 EF 35 BNE NBIT
2029:E6 08 36 INC POINT ; 下一個位元組
202B:D0 E4 37 BNE NBYTE
2020:E6 09 38 INC POINT+1
202F:A5 09 39 LDA POINT+1
2031:C9 D0 40 CMP #D0
2033:D0 DC 41 BNE NBYTE ; 判斷記憶區是否已結束 (CFFF)
2035:60 42 RTS
2036: 43 ;
2036: 44 ;
2036: 45 ; FEEDBACK ALGORITHM—SUMS BITS
2036: 46 ; 15,11,8 AND 6 WITH INCOMING BIT
2036: 47 ; ON ENTRY 'COUNT' CONTAINS INPUT BIT
2036: 48 ;
2036:AD 02 19 49 FEEDBACK LDA SIGH
2039:10 03 50 BPL NEX1 ; 檢查位元 15
203B:EE 00 19 51 INC COUNT
203E:6A 52 NEX1 ROR A
203F:90 03 53 BCC NEX2 ; 檢查位元 8
2041:EE 00 19 54 INC COUNT
2044:6A 55 NEX2 ROR A
2045:6A 56 ROR A
2046:6A 57 ROR A
2047:90 03 58 BCC NEX3 ; 檢查位元 11
2049:EE 00 19 59 INC COUNT
204C:AD 01 19 60 NEX3 LDA SIGL
204F:2A 61 ROL A
2050:2A 62 ROL A
2051:90 03 63 BCC NEX4 ; 檢查位元 6
2053:EE 00 19 64 INC COUNT
2056:6E 00 19 65 NEX4 ROR COUNT ; 把和存入 Carry
2059:2E 01 19 66 ROL SIGL ; 把 Carry 移入低位元組位元 0
205C:2E 02 19 67 ROL SIGH ; Carry 移入高位元組位元 0
205F:60 68 RTS
2060:AD 02 19 69 DISPLAY LDA SIGH
2063:20 DA FD 70 ISR PRBYTE ; 顯示在螢幕上
2066:AD 01 19 71 LDA SIGL
2069:20 DA FD 72 ISR PRBYTE
206C:20 8E FD 73 ISR CROUT ; 游標回頭
206F:60 74 RTS

```

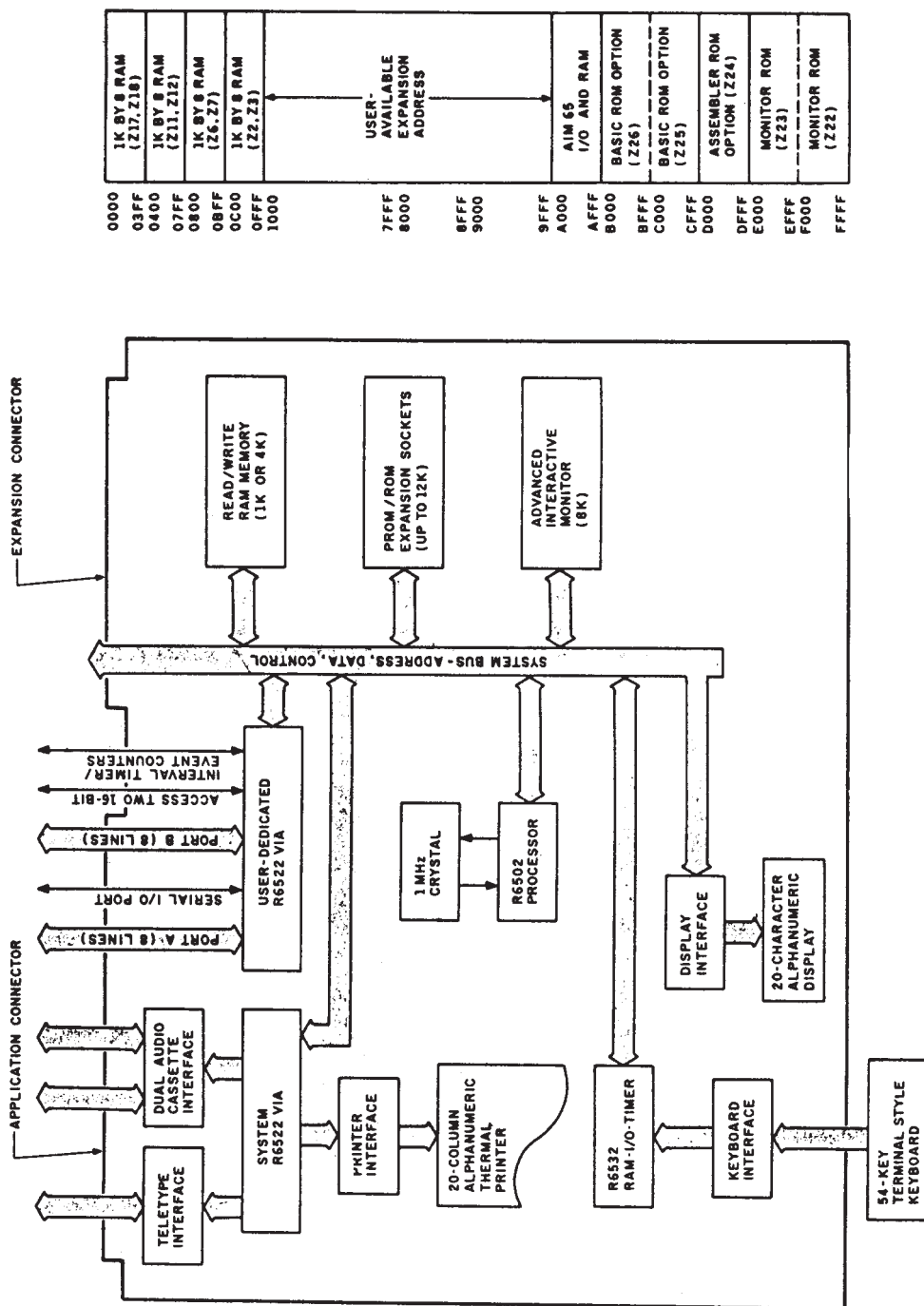


圖 7 AIM-65 方塊圖和記憶分配圖

列表四 AIM-65 測試程式

```

50      REM AIM65 TEST ROUTINE
60      HOME
70      REM DEFINE SYSTEM ADDRESSES
80      SELECT = - 15100
90      DISPLAY = 8298
100     NSIG = 8192
110     CSIG = 8200
120     BTEST = 8448
130     RAMTEST = 8336
140     PRINT " "
150     PRINT "LOADING MACHINE CODE TESTS"
160     PRINT "BLOAD APPTTESTS"
170     PRINT " "
180     PRINT "BUS TESTING-PROBE TARGET SYSTEM BUSES"
190     PRINT "(PRESS SPACE FOR NEXT TEST)"
200     CALL BTEST
210     IF PEEK (- 16384) < = 127 THEN 200
220     PRINT " "
230     PRINT "RAM TESTING 0000-07FF"
240     POKE SELECT,0: CALL RAMTEST
250     PRINT "RAM TESTING 0800-0FFF"
260     POKE SELECT,08: CALL RAMTEST
270     PRINT " ": PRINT " RAM TESTS COMPLETE "
280     PRINT " "
290     PRINT "ROM SIGNATURES BLOCKS B,C,D,E,F "
300     PRINT " "
310     FOR N = 176 TO 240 STEP 16
320     POKE SELECT,N: CALL NSIG
330     POKE SELECT,(N + 8): CALL CSIG
340     CALL DISPLAY
350     NEXT N
360     PRINT " ": PRINT " ROM SIGNATURES COMPLETE"
370     PRINT " "
380     PRINT " VIA TEST"
390     POKE SELECT,160: REM SELECT BLOCK AXXX
400     APRT = 51201:BPRT = 51200
410     ADIR = 51203:BDIR = 51202
420     POKE ADIR,0: POKE BDIR,255
430     REM A INPUT - B OUTPUT
440     FOR N = 0 TO 255
450     POKE BPRT,N
460     IF PEEK(APRT) < > N THEN PRINT "VIA ERROR"
470     NEXT N
480     POKE BDIR,0: POKE ADIR,255
490     REM BINPUT - A OUTPUT
500     FOR N = 0 TO 255
510     POKE APRT,N
520     IF PEEK (BPRT) < > N THEN PRINT "VIA ERROR"
530     NEXT N
540     PRINT " ": PRINT " TEST COMPLETE"
550     END

```

使您的APPLE 能夠擔任數位影像攝影機的——

影像攝影界面

-Micro D-Cam

原理與製作

隨著電腦科技的進步，各種精密的硬體設計及豐富的軟體不斷大量供應，使得電腦能快速而有效地完成許多人力所無法完成的工作。因此，近年來電腦已被廣泛的應用在家庭財務管理、學校教育、商業資料處理及工業自動控制方面。

這些年來微電腦在輸出裝置方面，如高速的影像顯示裝置，點矩陣印表機及語音合成等技術的發展日新月異，使得微電腦對於資料處理之結果，能夠更快速而有效的傳給使用者。但是不幸的是在微電腦輸入的技術方面，除了許多特殊應用之外，仍然是保持傳統式的鍵盤輸入。

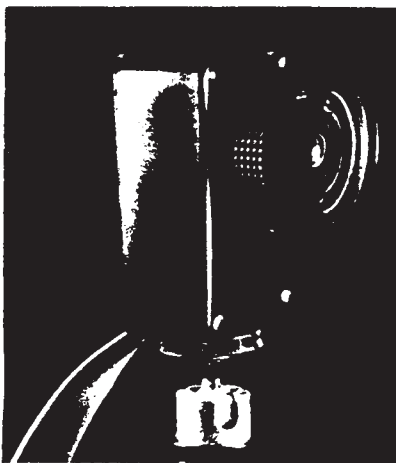
大家都知道，鍵盤輸入的資料，如果是屬於複雜的圖形或影像資料，這將會使輸入工作變得相當麻煩。基於這個原因，假如電腦能具有多用途的感知輸入裝置，則使用者可以不必再用鍵盤花費很長的時間，去輸入複雜的圖形或影像資料，取而代之的是可以利用電腦化攝

影機—computerized camera，將欲輸入之圖形或影像進行快速照像，以產生數位化影像資料，讓電腦能直接加以處理。

電腦攝影機

電腦影像攝影機雖然並不是一種新的產品，但是直到現在，這種產品的價格似乎仍嫌太貴。目前所使用的電腦影像輸入裝置，大都是採用傳統式的黑白照相機來作為影像感知器（image sensor）且照相機的影像輸出必須轉換成數位信號（“0”和“1”），以允許電腦作適當的處理，但是要完成此項工作却是相當困難的，因為影像輸出是採用Vidicon-type pick up tube，它將是一個高頻的類比信號分成每秒鐘30個圖框（frames of picture）來傳輸或掃描（北美以外的大多數電視系統是25圖框）。

大多數高級的電視攝影機介面是採用“



照片 1 圖中所示排線 (cable) 是接到 Apple 周邊連接器的介面卡

frame grabbing ”的方式，將類比信號轉換成電腦所能處理的數位信號，在此種處理方式之下，是在 $1/30$ 秒的圖框掃描期間，取樣 1 個圖框 (frame)，數位化且加以儲存。在這種高級的光感知系統，1 個快速的 A/D (analog to digital) 轉換器以即時 (real time) 的取樣率 (超過 5MHz)，將類比信號加以數位化，且將脈碼調變 (PCM) 資料存在快速的半導體記憶一緩衝器中，因為這種裝置是如此之快，因此對所照像的目標物之動作，能很靈敏的反應出來。

對於較低級的電視攝影機介面而言，設計者假設照相機的目標會保持靜止，以允許慢速的電路能有足夠的時間處理影像。當影像管保持固定不動時，所有的圖框信號是相同的，所以一般是採用一種順序行取樣技術 (sequential line-sampling techniques)，在這種裝置中，通常含有一個慢速的 A/D 轉換器 (每秒取樣 100,000 至 1,000,000 次)，此 A/D 轉換器連續周期或取樣窗 (sampling window) 動作的行和畫素位置計數器在圖框期間稍後的時間觸發。利用此種方式，在取樣窗之間，支援電路即具有足夠的時間，可將數

位化信號加以儲存且準備下一同步動作。由上面的說明可以知道：介面是利用許多圖框所擷取到的片段來組成單 1 的影像。

固態矩陣

上面所提到的 Vidicon 型攝影機，存在一個問題；因為它是一種類比裝置，所以其信號需經過轉換，才能應用於數位裝置中，因此若能有一種電腦影像攝影機本身就是數位式的，則可以省去 A/D 轉換，應用起來將更為方便。

但是，要想產生 1 個 256×128 或 128×128 的矩陣畫面，就必須設計 1 個光的、機械式的或電子式的矩陣，以將整個矩陣移到影像平面上有將影像移到矩陣上，以避免周期性地将圖形元素值記錄或儲存。

利用RAM担任光感知元件

過去曾有使用 type-1103 1K 位元 (bit) 的動態 RAM 作為光感知元件，以組成 64×64 圖素解像度的攝影機被發表，既然早期的 RAM 晶片可以作為感知另件，那是否意味著目前更新的 RAM 產品也能用來作為這種應用，以產生更高的解像度？即是否能使用 64K 位元 (bit) 的動態 RAM 來提供 256×256 矩陣圖像？

雖然上述想法並沒有錯，64K 位元的動態 RAM 確實是可以作為光感知元件，但問題是由於它們是設計來作為記憶裝置，因此並不是極佳的矩陣。會造成此項結果的主要原因是，記憶晶片的定址並不是非常線性地，即被定址到的某 1 個位元可能在矩陣的左上方，而另 1 個位元則可能在右下方……

在經過上述考慮之後，本想放棄所有的希望，後來偶而發現 1 個非傳統的動態 RAM 製造商，已經製造出一種能符合光感知水準的 64K 位元動態 RAM 裝置。Micron Technology 公司推出的動態 RAM 晶片，其內部是分成兩個記憶段，每個記憶段含有 250×128 個記憶單元 (cells)，參見圖 7。由圖 7 的記憶組

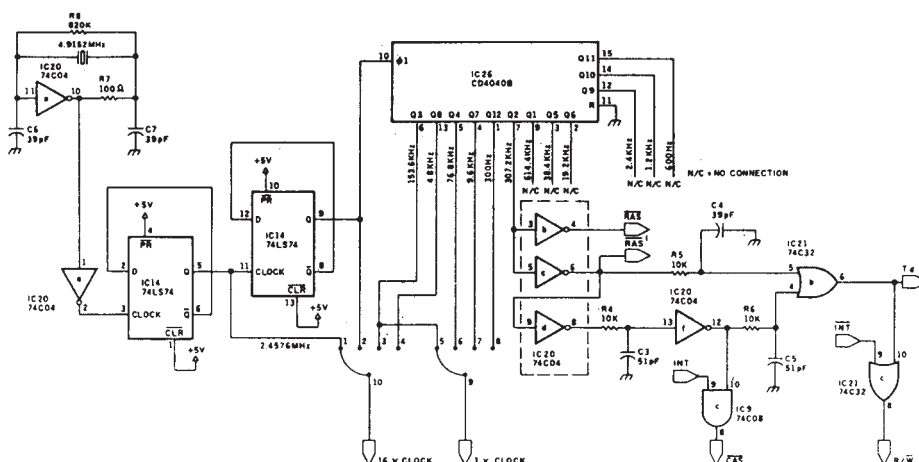
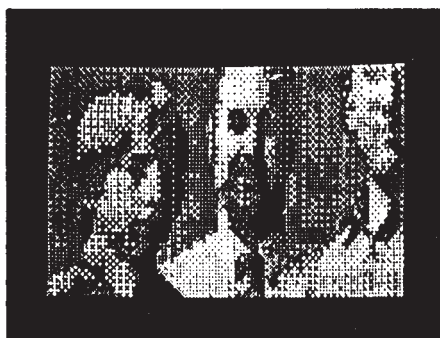


圖 1 時序產生器 (timing generator) 電路，電路中含有一個 CMOS 振盪器電路以產生基本的時鐘 (CLOCK) 振盪頻率。這個基本的時脈信號再經由除頻以提供各種不同的資料速率輸出同時控制 IC 32。資料速率時脈 (data-rate-clock) 信號是用以控制中斷產生器及傳送電路的動作順序。



照片 2a 此照片所示是利用 Apple II Plus 微電腦的高解像圖形常式將攝影機輸出再生的情形



照片 2b 以不同的白點密度表示不同的灰度大小 (gray levels) 的 Apple II Plus 顯示情形。此處主要是利用不同長度的曝光來形成灰度影像 (gray-scale image)

成結構知道，此種晶片可以很容易用來作為光感知元件。一種被特別測試過的 64 K 位元動態記憶裝置 IS32 Optic RAM 的包裝圖如照片 4 所示。

對於每個畫素花費而言，Micron Technology IS32 Optic RAM 要較早期的影像感知晶片一如 CCD 要節省 1000 倍。Optic RA

M 的分光靈敏度和一般的以矽為主所製造的光感知介質相同，但是它的 bit-for-bit uniformity 却不如 CCD 好。但是無論如何，Optic RAM 是唯一能使你的電腦具有直接處理影像信號的能力。

你可以自製一台微電腦攝影機

在本文中我們將告訴你如何使用 IS 32 RAM 晶片，來組裝 1 台相當廉價的數位影像攝影機，此攝影機我們將其稱為 Micro D-Cam（照片 1）。此數位攝影機的 256×128 畫素解像度（pixel resolution），可以適合許多的應用，如圖形、文字的辨別、機械人及安全系統等。（注意：Micro D-Cam 的輸出是數位信號，它不能直接用來驅動顯像裝置）。Micro D-Cam 可以配合 Apple II（II-Plus 及 IIe，參見照片 3）及 IBM PC 微電腦使用。總之，Micro D-Cam 具有串聯介面且僅需 5 條線的連接，因此它也可以連接到任何具有 RS-232C 串聯介面的微電腦去使用。

首先來瞭解 IS 32 RAM

Micro Technology 公司推出的 IS 32 RAM，是一種全數位化的影像感知裝置，其特性如表一所示。

1. 在 5.504×1.088 mm 的矩陣中含有 128×256 個單元（element）
2. 單元大小：8 microns $\times$ 9 microns
3. 垂直的中心至中心間隙：21.5 microns
4. 水平間隙：8.5 microns
5. 左、右矩陣間間隙：150 microns

表 1 由 Micro Technology 公司推出 IS32 Optic RAM 之特性規格，IS 32 RAM 是 1 個 64K bit 的記憶晶片，它是專為作為數位影像感知器而發展出來的

IS 32 RAM 含有 65,536（64K）的光感知記憶單元（memory cells），這些矩陣單元被分成兩個矩形陣列，每個陣列含 32,768 記憶單元，即 128 列 $\times$ 256 列的矩陣。且在記憶中的兩個矩陣，是以大約 25 個記憶單元寬的非感知晶帶 “dead zone” 加以分開。為了避免在影像中產生間隙或使用複雜的光系統來消除此中間區，因此在兩個矩陣中僅有 1 個被用來作為影像感知器。在記憶矩陣中的每個記憶單元都可以隨意的存取，只要控制電路能對所要選取的單元之行位址，及列位址進行適當的定址即可。

由曝光的動作瞭解攝影機工作原理

影像攝影機的工作原理是當影像攝影機攝取目標物時，焦距會將目標物反射回來的光，經由影像管傳送至 32768 個矩陣單元之 1。當 1 個個別的單元被光子撞擊到時，此矩陣單元中的電容器，即被啟動且啓先充電至 1 個定電壓，接著向 0 伏特電壓開始放電，電容器的放電速率和曝光期間的光強度成正比。

在曝光周期結束後，電路即經由記憶定址來讀取此單元。在記憶單元被存取期間，IS 32 RAM 中的感知放大器（sense Amplifier），會讀取電容器的電壓值，並且和一個固定的臨界電壓相比較，假如此矩陣單元的電壓是在臨界電壓之上，則此畫面單元（picture element）被視為黑色；假如矩陣單元的電壓值在臨界電壓之下，則此畫面單元被視為白色。IS 32 Optic RAM 中的資料輸出 Dout 接腳之電位即依據方才所討論的狀況來決定 “0” 或 “1”。

所有的動態記憶都需要記憶更新（refreshing）的動作，否則存在記憶單元中的資料，可能會因放電而消失掉。因此存在記憶單元中的資料必須被感知且每隔一段時間，將其所代表的信號位準重新寫回去一次。如此才能確保資料正確。此處所提到的記憶更新動作，通常發生在電腦從某一記憶單元讀取 1 位元時，但是更常發生在某一些外界電路定期地對記憶位

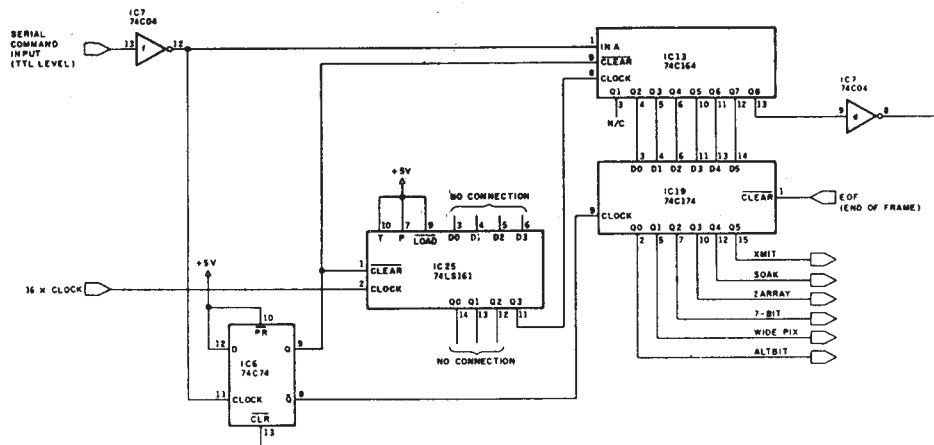
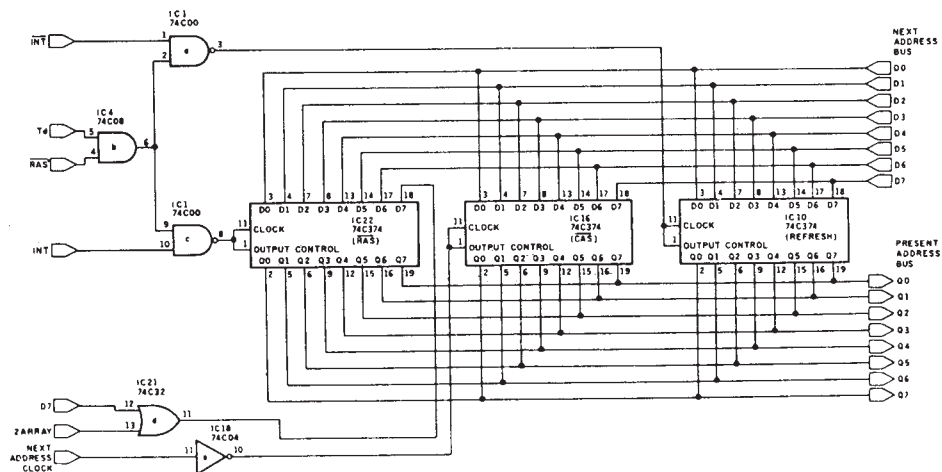


圖 2 如圖所示是許多串行的命令線用來將命令由電腦送至攝影機。資料是以串進的方式輸入命令接收器部份且由移位暫存器將這些資料組合成1個解碼的字組 (word) 存入門鎖中。

圖 3 此部份電路主要是用以門鎖列位址 (row address)，行位址 (column address) 及更新指標 (refresh pointer) 以作為 Optic RAM 定址用。位址暫存器 IC22 和 IC16 是用以儲存列及行位址。相對地，第 3 個暫存器 IC10 是作為更新暫存器



址進行週期性地傳送，以達到更新的目的。(許多的動作記憶晶片，包含 IS32 Optic RAM，均有能力在同一時間內更新整列的記憶單元)。IS32 作為一般記憶裝置時，正如上面所提到的必須有記憶更新的動作，但是在作為感光裝置時，則稍有不同。因為當 IS32 RAM 不是被更新，便是處於光感知 (light sen-

sitive) 狀態；因此要將 IS32 使用於攝影機的主要關鍵，是要小心地控制其光感知動作，並且以特殊方法完成其更新動作

在影像感知周期開始時，Micro D-Cam 電路會定址到所有正在作用的記憶單元，將它們填入表示 "1" 的正電壓值。接著在收到一個 SOAK 命令後，即開始曝光，這就好像打開

快門的動作一樣，在經過適當的曝光周期後，控制電路即發生更新的命令，它可以凍結記憶單元的狀態（或畫素）。然後控制電路啟動中斷狀態，在此期間1記憶單元的值即被讀取且傳送。中斷周期一直持續著，直到矩陣中所有位元都被重新傳送為止。（中斷狀態不能一直維持著，因為IS32在中斷期間無法被更新）。經由上述動作後，Micro D-Cam又使所有的記憶單元被重新設定為1。然後又重新開始一影像感知周期。

一個白色的畫素（邏輯“0”）的輸出，表示記憶單元的電容曝光的光強度足夠使電容器經由臨界點放電。1個黑色的畫素（邏輯“1”），表示光強度不夠時電容器經由臨界點放電。

攝影機能掃描影像改變的快慢，主要是依光強度而定。若要記憶單元能被快速的掃描或讀取，則需要更大的光強度。Micro D-Cam能以最快的速度15圖框／秒（frames/sec）來掃描影像。

與傳統的照像軟片相比較

若將數位影像感知器（digital image sensor）的動作，和傳統照相機的黑白軟片相比較，將更有助於了解。就像軟片一樣，IS32在單一的平面上含有許多光感和單元。影片被聚焦在平面上，使用者可以調整焦距或曝光長度。焦距是調整光照射到光感知介質的孔徑大小，曝光長度（相當照相機的快門速度）是利用驅動電子的掃描功能在IS32 Optic RAM中來加以調整。

Optic RAM的記憶單元是以二進制的形式來反應光的變化，因此僅能表示黑或白，即表示在曝光時某一額定數量光的存在與否，無論如何，照相軟片中的光感知單元具有各種不同的感知度，可感知各種不同的光強度，然而IS32對於某一時定狀況，幾乎僅能反應出相同的強度，爲了要克服此項限制，灰色的不同陰影可以藉著對同一感光影像的多重掃描，然後將各感光單元所得結果加以平均，對於各掃

描給予不同的臨界電壓或改變掃描率。

改變臨界電壓且將掃描影像及光強度保持固定，則Optic RAM中的區域將會被立即感光，不同的影像將會產生不同的輸出位準，臨界電壓位準2.1V，可以經由IS32的腳1（類比臨界）加以調整（1.5V至3V），但是Micron Technology公司建議灰度（gray-scale）能力，最好是藉改變掃描率而不是臨界電壓來完成。上述變化將會在影像是灰色（中間亮度）的地方之畫素表現出來，因此光撞擊單元電容量是接近臨界電壓。當然，在影像較黑的區域將會產生較多的邏輯“1”輸出，較亮的區域將會產生較多的邏輯“0”輸出。將這些輸出對掃描數求平均值，則適當的灰色陰影將可以在合成影像中被表現出來。

雖然Micro D-Cam並不含有機械式快門，但是它的電子電路可以很容易控制，以產生同等作用，只要傳送適當的控制命令給控制電路，即可完成此項工作。Optic RAM是根據其本身電壓的大小，來感知光的變化，利用這種方式可允許精確地連續控制Micro D-Cam的曝光值。

以6850擔任資料轉換

要將Micro D-Cam裝到電腦去是很容易的。單元控制電路提供所有必要的時脈信號，且能執行由電腦所送出的命令。Micro D-Cam能自動地控制Optic RAM的順序掃描，因此影像感知中的各記憶單元能被正確的存取，且將適當的影像信號傳給電腦以作為顯示或處理之用。

Micro D-Cam是使用C-mount lens影像管（此型式影像管常被用於16-millimeter電影攝影機或小型電視攝影機），具有可變的焦距。本電路所使用的影像管其攝物距離可以最少為18 inches（45 cm），在此種距離Micro D-Cam能分辨正在讀取的文字大小。爲了能以更大的倍數來觀看目標物，可以將1個調節器插入影像管及它安裝處之間，以調整影像管的焦距（參見照片2）。

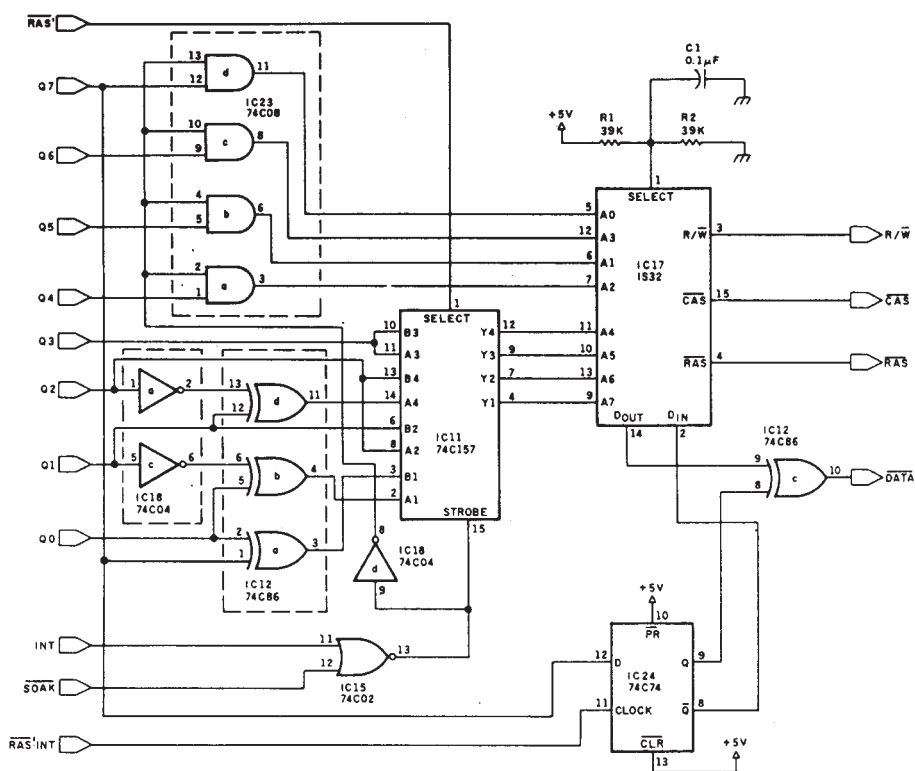


圖 4 此電路是由兩個反相器，3個互斥或閘，及一個多工器組成。位址重組（address-decrambling）電路是用以將RAM之位址重組。記憶截取（soak）電路是藉著更新周期（refresh cycle）的奪取，使得Optic RAM具光感知功能

電腦和Micro D-Cam之間是以1個TTL位準的串聯介面來鏈接。外加的資料速率時鐘（Data Rate CLOCK）信號允許電腦和Micro D-Cam保持同步，攝影機可以工作在自己的速度。

在電腦和攝影機間有5條線相連接，包含傳送，接收，地線，外加時鐘信號及+5V的電源供應。一個通用目的的6850 ACIA（非同步傳輸介面調整器）緩衝器晶片，用以完成串聯至並聯及並聯至串聯的資料轉換，在本文中所討論的是將Micro D-Cam連接至Apple II PLUS微電腦。

由時序電路談到更新及中斷狀態

時序產生電路（圖1），負責產生Micro D-Cam動作時所需的時序信號，此時序產生

電路含有1個CMOS振盪電路，用來產生1個基本的時脈頻率。然後此基本的時脈頻率即被加以除頻，以產生各種可能的資料輸出頻率且控制IS32。資料速率時脈（Data Rate CLOCK）信號，主要是負責控制中斷產生器及傳送電路的動作順序。

請看圖1，振盪電路發出1個4.9152MHz信號，經由74LS04（IC20a）反相器IC加以緩衝。接著此時脈信號經由1個D正反器加以除頻，然後所得的信號即可以和外界的資料速率選擇跳線相連接。IC26則可將頻率以2的乘冪加以除頻，以產生各種不同的輸出，這些不同的頻率輸出可連接到其他不同的資料速率選擇跳線。跳線5至8是用以選擇傳輸及中斷產生電路（圖5）的資料速率。另外跳線1至4是用在接收電路中。IC26腳7的輸

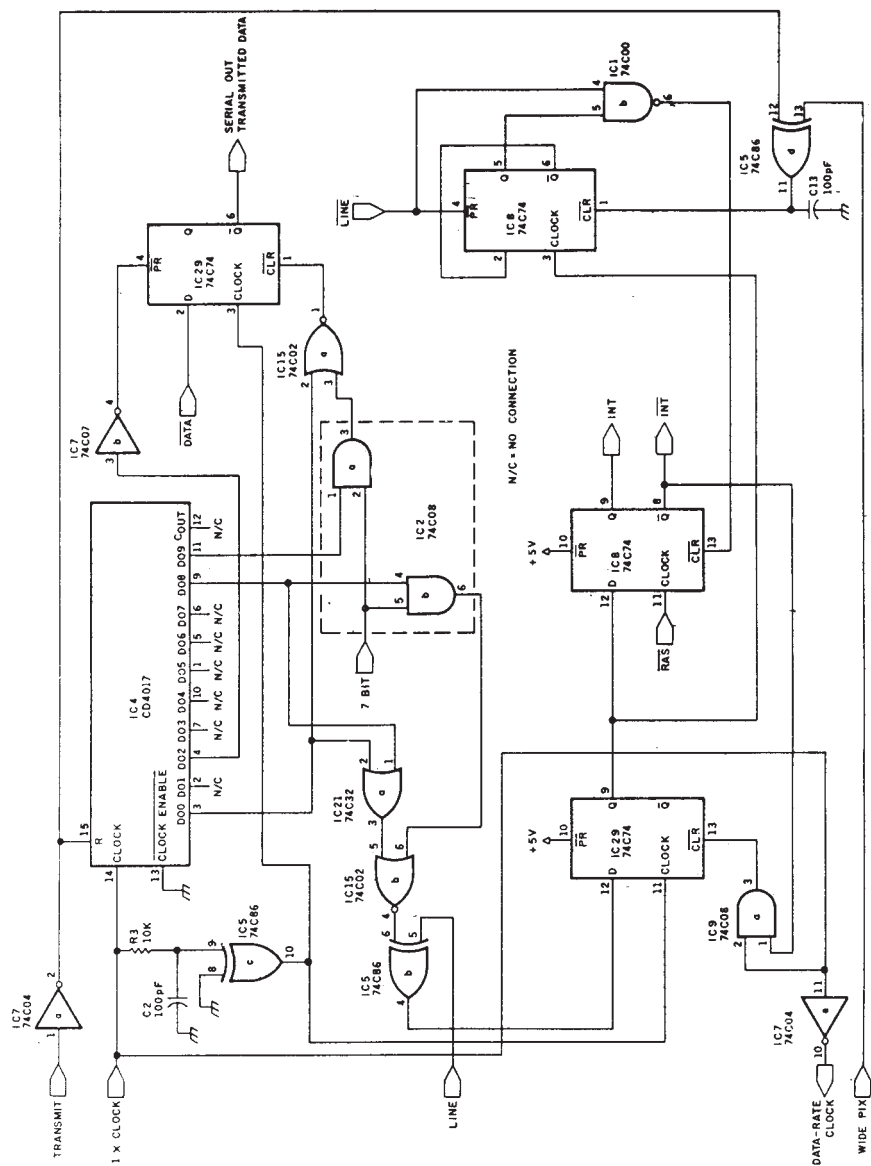


圖 5 此電路主要是用來將畫素資料 (pixel data) 傳給電腦且產生中斷狀態以允許 IS32 RAM 中的畫素資料能被讀出

出是用以驅動 Optic RAM 時序電路，以產生列位址激發 ($\overline{\text{RAS}}$) 信號，行位址激發 ($\overline{\text{CAS}}$) 信號，以及讀／寫 ($\text{R}/\overline{\text{W}}$) 信號。

當攝影機在和 Optic RAM 進行資料之傳輸時，Micro D-Cam 是處於中斷狀態，且 $\overline{\text{CAS}}$ 和 $\text{R}/\overline{\text{W}}$ 信號被加至 Optic RAM 上，當攝影機不在進行資料之傳輸時，中斷狀態即被

停止，且 $\overline{\text{CAS}}$ 及 $\text{R}/\overline{\text{W}}$ 信號失效；此時 INT (高態作用) 是 “L”，且 INT 是 “H”，因此用以驅動 CAS 信號的 AND gate 輸出仍維持 “H”，且用以驅動 $\text{R}/\overline{\text{W}}$ 信號的 OR gate 輸出仍維持 “L”。

在中斷周期期間，INT 變成 “H” 且 $\overline{\text{INT}}$ 為 “L”， $\overline{\text{CAS}}$ 及 $\text{R}/\overline{\text{W}}$ 信號被致能。 $\overline{\text{RAS}}$ 的

“H”狀態經過 2 個反相器（IC20d & f）組成的延遲線，以及一個 R/C 網路，然後此信號再和 $\overline{INT}$ 信號經由 AND gate（IC9C）加以組合，使 $\overline{CAS}$ 為“H”。當此動作發生時，行位址（Column Address）即被鎖入 Optic RAM 中，此時 $R/\overline{W}$ 信號仍維持“H”，所以對應位址之記憶單元的值即被讀取。在經過下一個延遲周期後， $R/\overline{W}$ 變為“L”，使 1 位元的信號被重新寫入此記憶單元中，並使此記憶單元能繼續保持光的變化。當 $\overline{RAS}$ 為“L”時，中斷周期結束且 $\overline{CAS}$ 和 $R/\overline{W}$ 失效。

命令—接收電路對資料的處理情形

串聯的命令信號線將電腦的命令傳給攝影機，這個資料以每次 1 位元的方式進入至命令接收部份（圖 2）然後再以下述的方式加以組合：第 1 個到達的位元是表示起始位元，隨後跟著 8 個資料位元，最後是結束位元。起始位元會使輸入移位暫存器致能，且啟動移位暫存器的時脈（最初設定為“L”）。當此時脈為“H”，起始位元（通常為“H”）就被鎖入移位暫存器 8 個位元中的第 1 個位元位置。當時脈又為“L”時，則第 1 個位元送達移位暫存器的輸入。且當時鐘脈波的上升緣（L→H）被檢知到時，移位暫存器將高態（“H”）起始位元由位置 1 移入位置 2，且將移位暫存器輸入端的第 1 個位元移入位置 1。依據此種方式，資料位元被連續地送至移位暫存器的輸入，且在時鐘脈波的上升緣檢知到時，在輸入端的一個位元即被移入移位暫存器中。

當起始位元最後到達 8 時，表示攝影機已收到整個命令位元組。因此在移位暫存器中的前 6 個資料位元即被傳至 1 個門鎖（1 位元組記憶）一稱為命令暫存器中。接著時脈失效，且移位暫存器被清除，此時 6 個攝影機命令位元留在命令暫存器中，而且現在接收器已準備好要接收下一個命令。

關於位址暫存器的工作情形

參見圖 3，位址暫存器電路將列位址，行

位址及記憶更新指標（edfresh pointer）加以門鎖，以作為 Optic RAM 定址用。位址暫存器 IC22 及 IC16 存有相對應的列位址及行位址，第 3 個暫存器 IC10 則是作為更新暫存器。圖 3 中的前兩個暫存器（IC22，IC16），僅在攝影機由 Optic RAM 提取或傳送 1 個位元資料時才被啟動（這裏所提到的提取動作—fetch operation 是中斷周期，正如我們前面所提到的，它是在 INT 為“H”時起始）。中斷周期是發生在 $\overline{RAS}$ 信號的下降緣，且在下一個 $\overline{RAS}$ 信號的下降緣結束。當攝影機在中斷周期沒有被提取時，更新暫存器仍保持動作著，即除了在中斷及曝光期間，更新暫存器會連續的將列位址的值由 0 增加到 255，且這些值也被送到 IS32 RAM 去，以完成更新（refreshing）的動作。此處所提到的 3 個位址暫存器都具有 3 態的輸出（也就是說在某一特定期間，它們的輸出可以假設為高阻抗狀態，即和滙流排是處於斷路狀態，且每 1 次僅有 1 個暫存器作用）。

如圖 3 所示，被選到的暫存器，將其資料送到一個稱為現位址滙流排（present address bus）的滙流排上。然後現位址信號經由 decramble 及 soak 電路（後面將討論到）送至 Optic RAM，以用來選擇 1 個列或行。另位現位址滙流排也連接到位址電路去，且 1 個 0，1 或 2 的值（由軟體選擇而定）也被加到現位址值上。產生的值即被送到位址滙流排（next address bus）去，下位址滙流排也被連接到各位址暫存器的輸入。且當下位址滙流排的值被門鎖入位址選擇暫存器後，接著暫存器即失效。

矩陣選擇電路僅選擇 1 個或 2 個 IS32 的單元矩陣來使用。假如 $\overline{2ARRAY}$ 是“H”，則 OR gate（IC21，腳 11）的輸出總維持“H”，且列位址暫存器的值（IC22），絕對不會比 128 小，所以僅有第 2 個矩陣（列 128 到 255）被定址及傳送。假如 $\overline{2ARRAY}$ 是“L”，無論如何 OR gate 將會出現暫態，且下位址滙流排上的信號線 D7 之值被送至 IC22，

由上面說明可以知道，所有由 0 至 255 的位址將被選到，且在兩個矩陣（第 1，第 2）的值將被傳送。

位址重組 & Array Soak

當存取 1 個記憶單元時，Optic RAM 的內部電路位截取列位址及行位址值，但因為記憶單元之位址是在光感知動作時才發出來，如圖 4 所示，位址重組（address descramble）電位將位址值重新組成 1 個新位址，而 Optic RAM 即用此新位址存取所要之畫素。

參見圖 4 所示，位址重組（descrambling）電路，含有 2 個反相器，3 個互斥或閘（EOR gates）及 1 個多工器（IC11）。此電路的主要工作是將位址暫存器的值轉換新成的位址值，而 Optic RAM 即將此位址值加以解碼，以存取所要的畫素。如圖 4 所示，反相器和互斥或閘是對列位址及行位址進行重組的工作，多工器則在適當的時間選取重組過的列位址或行位址，且將位址傳送到 Optic RAM 去。

多工器使用 $\overline{\text{RAS}}$ 信號決定要選取之位址。假如 $\text{RAS} = "H"$ ，即多工器的輸入選擇 SELECT = "H"（IC11，腳 1）則重組的列位址（B 輸入）被選到，若 $\overline{\text{RAS}} = "H"$ 則重組的行位址（A 輸入）被選到。

SOAK 電路的主要目的是防止在曝光期間，更新位址送到 Optic RAM 去（記住，Optic RAM 僅有在它不是記憶更新—refresh 時，才是光感知 light-sensitive）。在 $\text{INT} = "L"$ 及 $\text{SOAK} = "L"$ 期間，NOR gate（IC15d）的輸出是 "H"，這使得多工器的輸入致能為 "H"，且多工器的輸出為 "L"。NOR gate（IC15d）的 "H" 輸出，也使得反相器（IC18d）輸出 "L"，如此又使 4 個 AND gate 接至現位址匯流排（present address bus）及 IC32 的 4 個低次位址輸入。因此，Optic RAM 位址輸入仍維持 "L"，且記憶更新功能僅有在 address0 完成，當 SOAK 為 "H" 時，多工器及 AND gate 輸出被致能且更新位址送達 Optic RAM，因此，整個記憶器被更新，此時它不具感知能力。

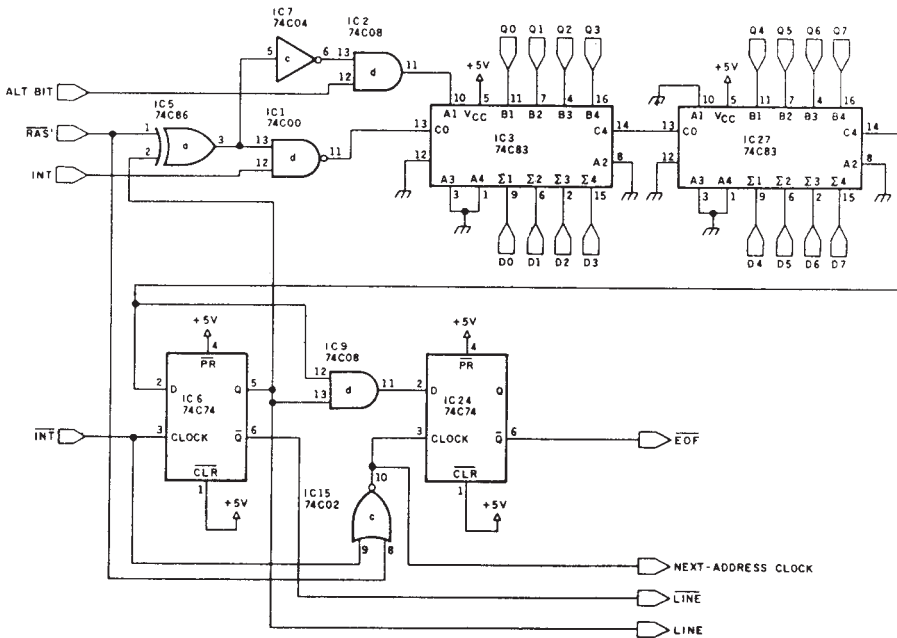


圖 6 是 1 個位址電路使得 Micro D-Cam 能記錄列、行或更新暫存器的值。

傳送器及中斷產生電路

傳送器及中斷產生電路如圖 5 所示，將畫素資料串聯的傳送到電腦去，且在資料間插入適當的起始及結束位元，且發出 $\overline{\text{INT}}$ 及 $\overline{\text{INT}}$ 信號，以便由 IS 32 提取畫素信號。

如圖 5 所示，漣波計數器 (ripple counter) IC4 是電路的心臟，當 Micro D-Cam 收到命令要傳送資料時，IC4 即被致能。IC4 在傳送起始及結束位元時，會使中斷產生器禁能 (inhibit)，以防止對 Optic RAM 進行存取，且當 IC4 開始傳送資料時，使得中斷電路致能。傳送器的頻率是由前面曾提到的資料速率時脈 (Data Rate CLOCK) 決定。在每個時脈周期期間，僅有 1 個起始及結束位元被傳送。

中斷產生器是由漣波計數器 IC4 及資料速率時脈 (Data Rate CLOCK) 所致能，但是中斷周期本身是由 $\overline{\text{RAS}}$ 所同步，因為中斷周期的目的是提取單 1 畫素 (pixel) 以作為傳送，因此每個時脈周期僅有 1 個畫素被傳送。資料速率時脈的上升緣會使中斷電路致能。下一個 $\overline{\text{RAS}}$ 信號的下降緣會啓始中斷周期，導致 Optic RAM 的一個畫素被讀取。 $\overline{\text{INT}}$ 信號回送到中斷電路，使中斷電路重定 (reseting)。

當 $\overline{\text{RAS}}$ 信號再次成為 "L" 時，中斷周期結束。且資料速率時脈的下一個下降緣 (H → L) 即再使中斷電路致能 (除非有一個起始位元或結束位元正被傳送)。由上面的說明可以知道，在每個資料速率時脈周期期間，僅有一個畫素被傳送。

WIDEPIX 電路，是用來補助補償 Optic RAM 與電腦圖形螢幕的寬與高之比。Optic RAM 的寬與高比是 2.5 : 1，大多數的螢幕顯示則是 4 : 3，且畫素 (pixel) 並不是方形的，假如螢幕上顯示的影像資料之寬與高比近似於 1 : 1，則影像顯示將會擠壓的近乎水平。WIDEPIX 電路主要是用來補助此點之補償，使每個畫素被傳送兩次，使得影像寬度被增

加 2 倍。當 Micro D-Cam 在進行資料傳送時，WIDEPIX 電路被致能，且其命令線為 "H"，這使得正反器 (IC8) 的輸出，在每個資料速率時脈周期 (Data Rate CLOCK) 被反相一次。而正反器在交變的資料速率時脈周期，會使中斷周期致能，在中斷被致能的資料速率時脈期間，上一中斷周期的畫素又再次被傳送。

$\overline{\text{LINE}}$ 和 $\overline{\text{LINE}}$ 信號，是用以指示行位址暫存器已達到終點計數，即表示最後一行已被掃描過。當這兩個信號發生作用時，會使得現位元組 (current byte) 傳送期間的任何中斷被禁能，所以前一存取的資料位元值重覆著，直到此位元組的其他位元被填滿為止。這保證下一位元組將會在下一列時脈才開始傳送。當結束位元被傳送時，在互斥或閘 (IC5b) 的一個輸入的 $\overline{\text{LINE}}$ 信號，會產生 1 個中斷要求，而 1 個 $\overline{\text{LINE}}$ 信號送至 NAND gate (IC1b) 的輸入可確保中斷正反器被致能，這種虛擬 "dummy" 中斷是用來使列位址暫存器增量。在這個周期被存取的畫素是在傳送結束位元時被清除成空白。

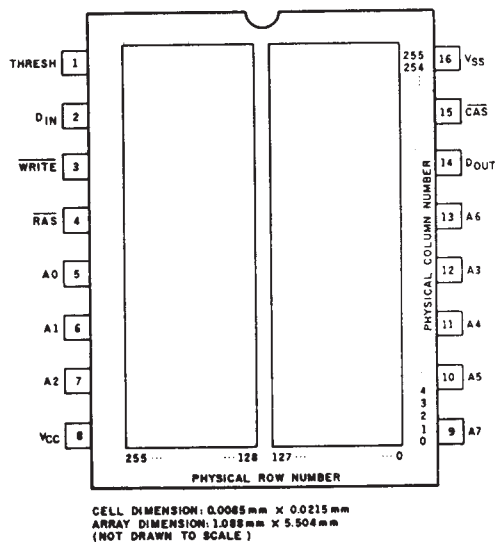
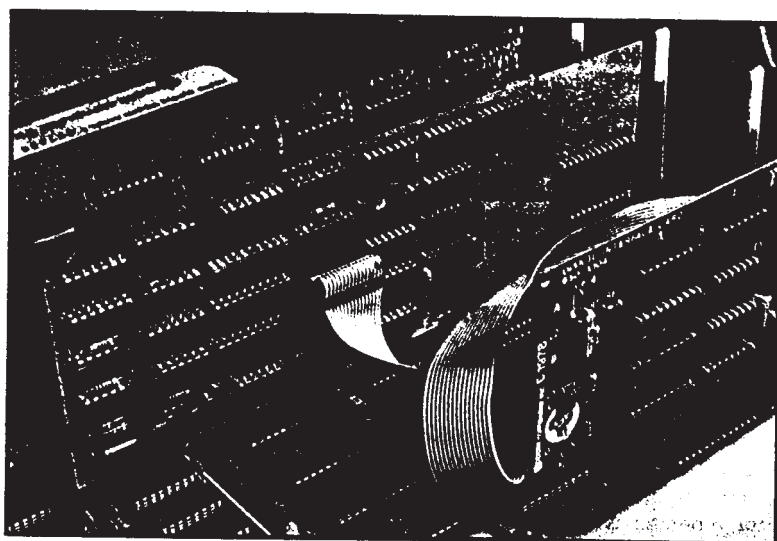
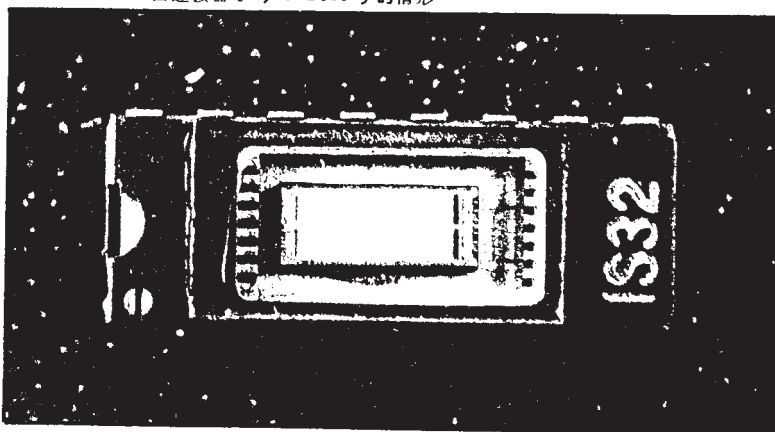


圖 7 如圖所示是 IS32 Optic RAM 的結構，按腳圖由圖中知其內部含有兩個矩陣，每個矩陣含有 128×256 記憶單元 (cell)



照片 3 此照片所示是微電腦數位影像攝影機 (Micro D-Cam image camera) 的控制电路安装在 Apple II Plus 微電腦輸入／輸出連接器 (I/O Slot) 的情形



照片 4 照片所示是由 Micro Technology 公司所推出的 IS32 Optic RAM, 64K bit 動態記憶裝置的包裝情形

加法器和圖框結束電路

加法器和圖框結束電路，參見圖 6 所示，將適當的增量加至列，行及更新暫存器且產生信號以指示 Optic RAM 的圖框結束。

當位址暫存器之 1 有將值送到現位址匯流排時，則加法器電路會收到這個值，且將它加上 0，1 或 2（依輸入控制 $\overline{RAS}$ ，LINE，ALTBIT 及 INT 之情況而定），且將所得之和

置於下位址匯流排上。當更新暫存器被啟動時，INT 信號會導致每個周期加 1，在中斷周期時，列和行位址暫存器被啟動，加法器將 0 加到列位址，1 加到行位址直到行結束。然後加法器再將列及行加 1，因此使得暫存器加 1，且行暫存器之值被重定為 0。

在中斷周期期間，ALTBIT 輸入僅將現位址匯流排加 1；因此列和行位址暫存器是總共被增加 2，而不是 1。

Number	Type	+5V	GND	Number	Type	+5V	GND
IC1	74C00	14	7	IC17	IS32	16	8
IC2	74C08	14	7	IC18	4069	14	7
IC3	74C83	5	12	IC19	74C174	16	8
IC4	4017	16	8	IC20	4069	14	7
IC5	74C86	14	7	IC21	74C32	14	7
IC6	74C74	14	7	IC22	74C374	20	10
IC7	74C04	14	7	IC23	74C08	14	7
IC8	74C74	14	7	IC24	74C74	14	7
IC9	74C08	14	7	IC25	74LS161	16	8
IC10	74C374	20	10	IC26	4040	16	8
IC11	74C157	16	8	IC27	74C83	5	12
IC12	74C86	14	7	IC28	74C00	14	7
IC13	74C164	14	7	IC29	74C74	14	7
IC14	74LS74	14	7	IC30	MC6850	12	1
IC15	74C02	14	7	IC31	74LS245	20	10
IC16	74C374	20	10				

表 2 是顯示各 IC 在 Micro D-Cam 電路中的電源接腳

Micro D-Cam 之控制及使用

用以控制 Micro D-Cam 的軟體程式都具有螢幕說明，因此操作起來相當容易。同時當攝影機在動作時，有許多及時命令 (Real time command) 可用來控制攝影機的動作，這些及時命令通常也都有顯示在螢幕上，以供使用者選擇。

當攝影機首次被打開時，即已開始影像收集程序，且此時使用者可以選擇由程式 (下一次討論) 所提供的螢幕菜單上所顯示之功能。假如一切正常的話，則由 Micro D-Cam 所攝取的影像，將會在電腦螢幕上顯示出來。假如電腦螢幕仍呈黑色，則曝光周期可能不夠。假如曝光周期超過，則螢幕可能呈現白色，解決此問題的方法是可以減少曝光時間或改善影像的焦距，當一切調整無誤時，你將可在螢幕上看到一個清楚的影像顯示。

附註：本文所介紹的 Micro D-cam 在國外有套件可買來自行組裝，亦有成品出售。出售公司及套件，成品價格如下所示：

The following items are available from:
The Micromint Inc.
561 Willow Ave.
Cedarhurst, NY 11596
(800) 645-3479 (for orders)
(516) 374-6793 (for information)

1. Complete Micro D-Cam unit including interface card, extension cable, IS32 Optic RAM, lens, remote housing, operators manual, and utility software. Specify Apple II (Plus or E), or IBM Personal Computer.
Assembled and tested \$295
2. Same as Item 1 except in kit form. Specify Apple II or IBM Personal Computer version.
Complete kit \$260
3. IS32 Optic RAM sold separately
IS32 each \$42
4. RS-232C-interfaced Micro D-Cam for general use. Call for price and delivery.

Please add \$4 shipping and insurance in continental United States, \$20 overseas. New York residents please include 7 percent sales tax.

Micro D-Cam的界接方式

在前面的文章中我們曾提到Micro D-Cam和電腦之間的信號傳送是採每次一位元(bit)的串聯傳送方式。若以最簡單的結構而言，Micro D-Cam需要4條信號線連接至電腦去，這些信號線包括+5V和地線，輸入及輸出串聯傳送信號線。對於沒有特別指定的結構而言，Micro D-Cam可以以RS-232C鏈接，操作在非同步的工作狀態下(資料傳送速率是

19,200 bps，位元/秒)，但在本文中，所談到的攝影機串聯界面，可以將Micro D-Cam直接和IBM PC及Apple II微電腦的匯流排相聯接(仍是串聯傳輸方式)。在這個界面中使用5條信號線，即將上面所提到的4條信號線，加上由驅動電路所提供的外加時鐘信號，使得Micro D-Cam可以工作在153,600 bps的資料傳送速率。此處所提到的界面電路的複雜度，是視電腦的匯流排結構及位址範圍而定。一個通用目的的硬體界面簡圖如圖8所示：

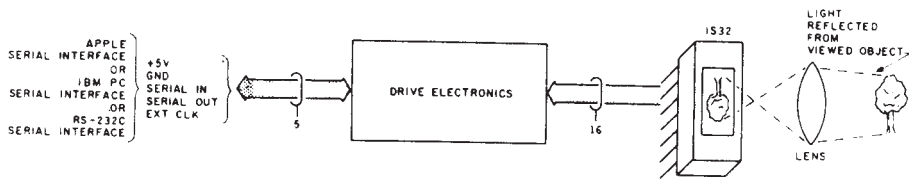


圖8 Micro D-Cam方塊圖

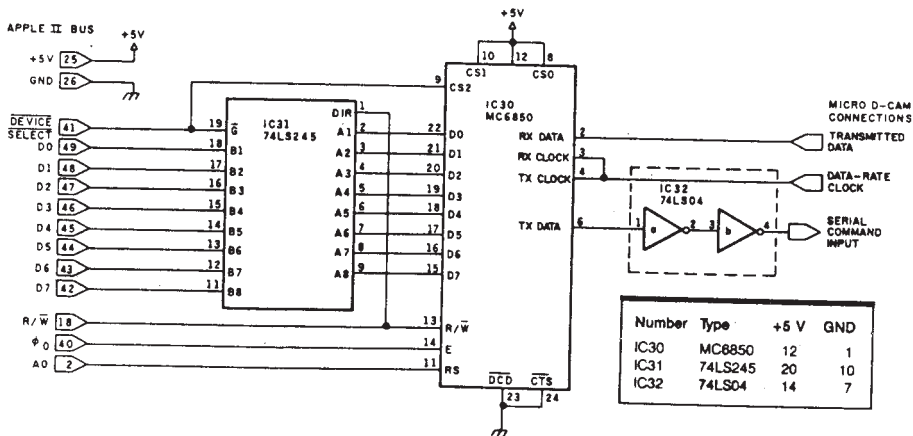


圖9 用以將Micro D-Cam連接到Apple II Plus或Apple IIe的界面電路，由Optic RAM送出的串聯資料，是被ACIA(MC6850)及74LS245 IC轉換成並聯的位元組，然後送至Apple的資料匯流排。由於攝影機控制电路的外加時鐘輸入，使得高速度的資料傳送(153,600 bps以上)成為可能。

圖9所示是用以連接Micro D-Cam至Apple II Plus微電腦的界面電路。此界面電路之所以如此簡化，主要應歸功於Apple II主

機板上已將各周邊連接器(I/O Slot)的位址預先解碼過，使得界面電路可以省却位址解碼電路。完整的串聯界面僅需兩個IC即可組

成。74LS245 匯流排傳送器接收器 (transceiver) 將 TTL 位準的串聯信號加以緩衝，然後送出。MC6850 ACIA (非同步串聯界面調適器) 的串聯資料傳送率，是由 Micro D-Cam 驅動電路的外加時鐘輸出所控制，對於以最快資料傳送率而言，時鐘頻率應設定為 153,600Hz。

圖10所示是 Micro D-Cam 連接至 IBM PC 的界面電路，由於 Intel 8088 微處理器較 Apple 的 6502 要複雜很多，且由於 PC 龐大的記憶空間，使得圖10的界面電路也複雜許多。Micro D-Cam 的兩個埠位址 (port address)，是由 3 個晶片 IC5，IC6 和 IC7 所解碼，這 3 個 IC 是 74LS136 開路集極互斥或閘

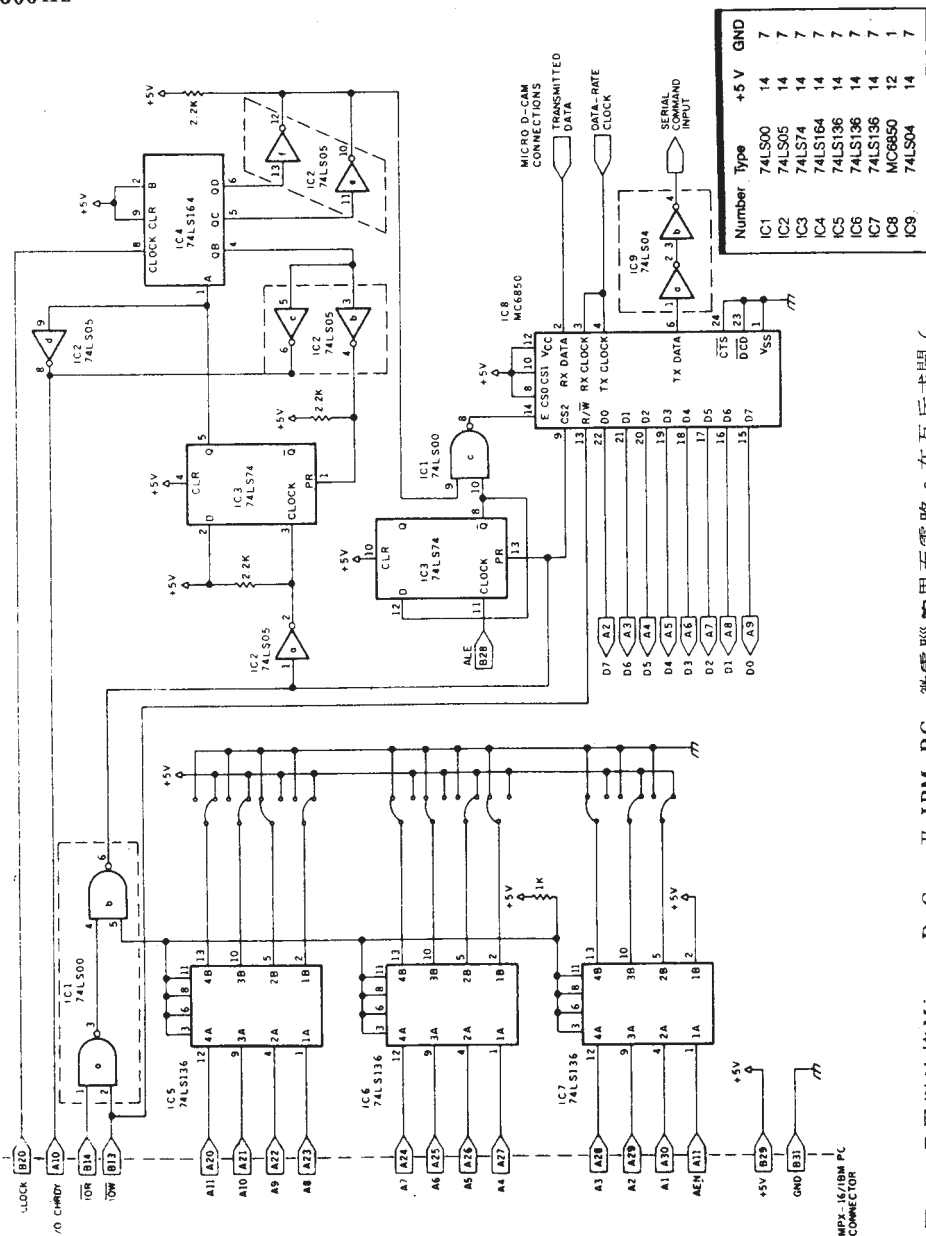


圖10 是用以連接 Micro D-Cam 及 IBM PC 微電腦的界面電路。在互斥或閘 (EOR gates) 輸入端的一些跳線，是用以決定界面的輸入/輸出匯流排位址。圖上所示是將位址設定為 xD26 及 xD27

(exclusive-OR)，連接成 " wired-OR " 的結構 (見圖 10)，連接至位址解碼器的 11 個輸入端的電壓，是用以決定界面板的位址，如圖 10 所示之電路連接，其解碼位置為 xD26 及 xD27 (x 之值可以是十六進位值 \$0 至 \$F 之任一值)。6850 ACIA (IC8) 在圖 10 中的工作，就和前面所提到的，另外 IC2 和 IC4 則是作為等待狀態 " wait-state " 產生器以定期地對匯流排進行存取。

資料和命令格式

6850 ACIA 內含 1 個資料暫存器 (data register) 及狀態暫存器 (status register)，你可以將數值寫入狀態暫存器以組成

動作參數 (如同位元 — parity，結束位元，起始位元，時鐘率等)。由此處的說明可以知道，在微電腦能存取 Micro D-Cam 之前，ACIA 必須預先設定各參數的初值，以便能正確的動作。本設計的控制軟體，是藉著寫入兩個位元組來完成，即兩個十六進位的 03 及 14 被連續寫入狀態暫存器。第 1 個位元組是用以完成 ACIA 的重定 (reset) 工作，第 2 個位元組則是用以指定串聯傳輸形式是 1 個起始位元，緊跟著 8 個資料位元，後加 1 個結束位元。

藉著控制程式對狀態暫存器的讀取，可以決定 ACIA 何時可以接收新資料，以及 ACIA 是否準備將資料送出。表 3 所示就是狀態暫存器中各位元被設定是所代表的狀態。

3 a	Status Bit	Meaning When Set to 1
	0	data has been received from the camera
	1	a command may be sent to the camera
	2	unused
	3	unused
	4	received data was improperly framed
	5	data received before previous byte read
3 b	Command Bit	Meaning When Cleared to 0
	7	none (always 1)
	6	none (always 1)
	5	alternating-bit mode (ALTBIT)
	4	wide-pixel mode (WIDEPix)
	3	7-bit data bytes (7BIT)
	2	transmit one frame instead of two (1ARRAY)
	1	refresh instead of soak (REFRESH)
	0	send the requested image (SEND)

表 3 專為 Apple II/Micro D-Cam 界面而設定的狀態暫存器中各元件所代表的意義。表 (3 a) 代表狀態位元，表 (3 b) 代表命令位元。

通常狀態暫存器在正常使用時，僅有位元 0 (bit 0) 被檢查，以判別攝影機是否有可用的資料。位元 4 和 5 則僅被用作除錯 (debugging)。當設計一個程式用以接收由攝影機而來的影像資料時，有一個很好的觀念，就是併入一個暫存功能，例如，當攝影機停止送出信號，或是程式失漏了某個位元組資料的接

收，則都可導致程式暫停執行。

在 Apple II Plus 及 IIe 中，十六進位位置 CO_nE 是用以存取狀態暫存器，CO_nF 則是用以存取資料暫存器，而這兩個位置主要是視界面板是置於那個週邊連接器 (SLOT) 而定。n 是十六進制的週邊連接器編號 (Slot no) 加 8 而得。例如，假設留面板是插在 Slot 3

； $3 + 8 = B$ （十六進制），因此 COBE 可用以存取狀態暫存器，COBF 則可用以存取資料暫存器。

命令功能

當攝影機在動作時，電腦會送出命令語（command words），以控制 Micro D-Cam 的工作模式。每個命令語均是由 8 個位元（bits）所組成，命令語中各位元所代表的意義如表 3 b 所示。

交變位元模式—ALTBIT Mode

當位元 5（bit 5）被清除為 0 時，Micro D-Cam 僅送出在 Optic RAM 中偶數列及行的圖素（pixels），這個模式通常可造成一個較在 NOALTBIT mode 要更清楚的影像。

加寬模式—WIDEPIX Mode

當位元 4（bit 4）被清除為 0 時，Micro D-Cam 會將矩陣中的每個圖素傳送兩次。由於在矩陣中的影像是長方形的形狀，所以將圖素採雙倍傳送“double-transmitting”的方式，則異影像在電腦螢幕上顯示出來時，將會有適當的寬對高之比。

7位元模式—7Bit Mode:

Apple II'e 微電腦完成高解像圖形的方法有些特殊。每個位元組的最高位元是被用作一群圖素的顏色位元（color bit）同時其他的 7 個位元則是以 1 或 0 來表示亮或不亮的圖素。在 7 Bit Mode 的工作模式上，Micro D-Cam 是傳送和 APPLE 高解像圖形具通用格式的資料，也就是說在每個位元組中有 7 位元的圖素資料。7 Bit Mode 可以將命令語的位元 3 清除為 0 來選擇。改善的 7 Bit Mode 是 8 Bit Mode，這個 8 Bit Mode 是藉著將命令語的位元 3 設定為 1 來選擇。8 Bit Mode 的工作模式，會使攝影機以正常的位元對映（bit-mapped）的格式將資料傳送出來，也就是說每個位元組中的 8 位元均是代表影像資料

，這種 8 Bit Mode 通常被廣泛的應用在 Apple 之外的許多微電腦中。

單|矩陣模式—| APPAY Mode

1 ARRAY mode，可以將命令語的位元 2（bit 2）清除為 0 來選擇。在選擇了這個工作模式後，則在光感知矩陣中僅有下半部份矩陣的影像資料被 Micro D-Cam 傳送出來。若命令語中的位元 2 被設定為 1 時，則可選擇雙矩陣模式—2ARRAY mode，在此種工作模式下，會使得 Micro D-Cam 將矩陣中所有的影像資料傳送出來。在 2ARRAY mode 的工作模式下，會在螢幕上產生分割（split-screen）效應，主要是因為光感知晶片中兩個矩陣間的空間所致。

更新模式—REFRESH Mode

就許多方面而言，Micro D-Cam 就像其他照相機一樣，它必須接受到適當的光，使得影像能適當的發展出來，但太多的光將會使影像過度曝光，而太少的光則又使得影像曝光不夠，此處所提到的曝光時間（Exposure time），主要是由微電腦中的控制程式，允許 Optic RAM 之記憶單元不被更新（refresh），即曝光的時間來決定。影像感知器的更新就和動態記憶裝置的更新程序完全一樣，即在記憶更新期間，現存於各記憶單元中的電荷值會被感知，且這個感知電壓值會和臨界電壓值相比較，然後一個新的 0 V 或 5 V 的電壓會被重新寫入記憶單元中。（Optic RAM 和一般動態記憶裝置的唯一不同是，當更新行動停止，即影像感知周期之初，所有記憶單元之初始值均必須為 +5 V）。假如影像感知器沒有被連續的更新，則聚集在各單元上的光，將會使得各記憶單元中的電壓值會漏失掉，且漏電的速率和光強度成正比。當影像感知器不在記憶更新時，我們可以說它是處於光的浸透“soaking”狀態，如此一來使得影像感知器有較長的浸透時間，也使 Micro D-Cam 在暗淡的光線下仍能看得很清楚。

命令字元	產生的控制動作
>	增加曝光時間
<	減少曝光時間
F	將現調整的曝光時間，固定為以後使用的曝光時間
L	由磁片中載入以前儲存的影像資料
N	將反相影像由 Epson 印表機印出
P	將螢幕影像由 Epson 印表機印出 (需 Graftrax option)
Q	停止且回到螢幕主菜單
R	切換曝光時間及光強度的顯示
S	將現顯示影像存入磁片中
T	使用現光強度位準且自動校正曝光

表 4 表上所示為 Micro D-Cam 控制程式所提供的命令選擇。參見表 5 可以看到由 GREY 16 程式所提供的選擇

當更新模式 (REFRESH mode) 被選擇時 (可清除命令語的位元 1 而得)，Micro D-Cam 會使得影像感知器被更新同時它也送出影像。當命令語的位元 1 被設定時，SOAK mode 的工作模式被要求。這導致攝影機在傳送影像時，同時保持浸透 - SOAK 狀態 (即仍然對光保持感知作用)。

影像傳送模式—SEND Mode

當命令語的位元 0 (bit 0) 被清除為 0 時，即進入影像傳送 (SEND) 模式，使得攝影機開始傳送影像信號，控制軟體。

雖然 Micromint 公司有推出適用於 Apple II 及 IBM PC 微電腦的控制軟體。這些控制軟體主要是控制 Micro D-Cam 之影像顯示。

在本文將提供兩個適用於 Apple II 的完整控制程式列表。1 個是用來作為 Micro D-Cam 的測試實驗用；另 1 個則是結構化的軟體常式，它主要是用以加強影像顯示且完成灰度 (gray scale ordered dithering) 的工作，除此之外，Micro D-Cam 軟體還含有許多具螢幕說明的公程式 (某些選擇如表 4 所示)

，所有在前面或下面文章中所看到的照片，都僅要使用我所提供的兩個程式即可顯現出來。

一個控制程式的實例

Micro D-Cam 示範程式 (列表 1)，主要是用以說明接收由攝影機而來的資料，且將影像顯示在 Apple 高解像螢幕上所需的簡單程式。這個程式實際上並不如它外表看起來那麼長或複雜，由圖 11 所示的流程圖即可看出這個程式的工作情形。軟體是由兩部份組成：一個短的 BASIC (列表 1a) 及一些機械語言常式 (列表 1b)。BASIC 程式將機械碼由磁片載入電腦中，且設定正確的 I/O-Slot 編號及曝光時間，然後呼叫機械語言常式以將影像顯示出來，在回到 BASIC 程式後，呼叫的程式會檢查你是否要結束這個程序。

在程式中最困難的部份是由機械語言常式所完成，它允許 Micro D-Cam 可以工作在 153, 600 bps 的速率。當機械語言常式被呼叫時，機械語言常式會先確定高解像螢幕正在顯示。然後對 ACIA 作初始化工作，且送一個命令以告訴攝影機進行浸透 (soak) 而不送出影像 (這個動作效應會導致 Optic RAM 被清除且開始曝光程序)，然後程式在曝光期間即進入等待狀態。

下一個步驟是讀取由攝影機而來的影像資料，且在螢幕上將影像顯示出使。為了要節省時間及記憶，軟體直接將影像送至高解像螢幕記憶區 (並不採用將影像讀入另一個記憶緩衝區，然後再送出去顯示的方法)，以簡化影像的處理程序。在交變圖素 (alternate-pixel) 模式，擴展圖素 (wide-pixel) 模式的工作模式下，都是採用 7 bit words 的資料格式。在影像被接收之前，有一些記憶指標被預先設定，以便能將影像資料送到螢幕的適當位置。一個命令會被送至攝影機以開始進行影像資料的傳輸，且一個程式迴路會讀入影像的各位元組資料且將它置於螢幕上。

雖然控制軟體知道對一個影像而言，應由攝影機接收多少位元組的資料，但是有一個問題可能會發生，主要是由位元組計數 (byte-

counting) 所造成 (位元組計數主要是用以決定何時停止資料之讀取) : 假如電腦漏失了某個位元組的資料, 則它可以使系統中止動作。爲了安全起見, 在影像讀取常式中也提供了一個暫停迴路常式。假如電腦執行了暫停程式, 則它會等待攝影機的修正動作, 它會使喇叭發出聲音且等待一個按鍵輸入, 然後再重新開始整個命令程序的執行。利用這種方法, 你可修正任何可能發生的問題。

因爲 Apple 的高解像螢幕顯示和記憶體的对映 (mapped), 並不是一種線性的方式, 所以在機械語言常式的結尾有一個長度表, 用以提供高解像螢幕上各連續列 (row) 的記憶起始位址, 機械語言即根據這些起始位址, 將由攝影機讀取的 40 位元組資料放置到螢幕上去。以下的連續列都是採用此種方式, 將資料在螢幕上顯示出來。

當影像在螢幕上顯示出來後, 一個命令會被送到攝影機去進行更新而不傳送影像信號, 這將使攝影機準備下一個曝光。最後, 機械語言在回到 BASIC 之前, 會檢查鍵盤且執行任何

命令輸入。

灰度-Gray Scale 的獲得

列表 2 附於文章最後的 GREY16 程式是一個能具有灰度 (gray-scale) 能力的示範程式。它具有一個工作模式可允許你快速的調整攝影機的焦距, 另外也可以讓最後影像的樣子能有個概念, 且可產生一個 15-intensity-level 的灰度影像在 Apple 上。 (有關上述程序的描述, 參見圖 12 的流程圖)。

使用 GREY16 程式, 你可以改變正被顯示影像的曝光時間, 或是你可以改變灰度影像的曝光上限及下限。一旦你獲得滿意地影像顯示後, 你可以將其存入磁片以便於日後使用或利用 Epson MX-80 印表機將其印出來 (使用螢幕傾印常式)。GREY16 程式可用的一些命令之目錄如表 5 所示。

當 Micro D-Cam 的電源被打開時, 你可以由 GREY16 程式所提供的菜單中選擇一項功能來啟動攝影機的動作。假如曝光時間不足, 螢幕將仍是黑色。假如曝光時間太過, 則螢幕

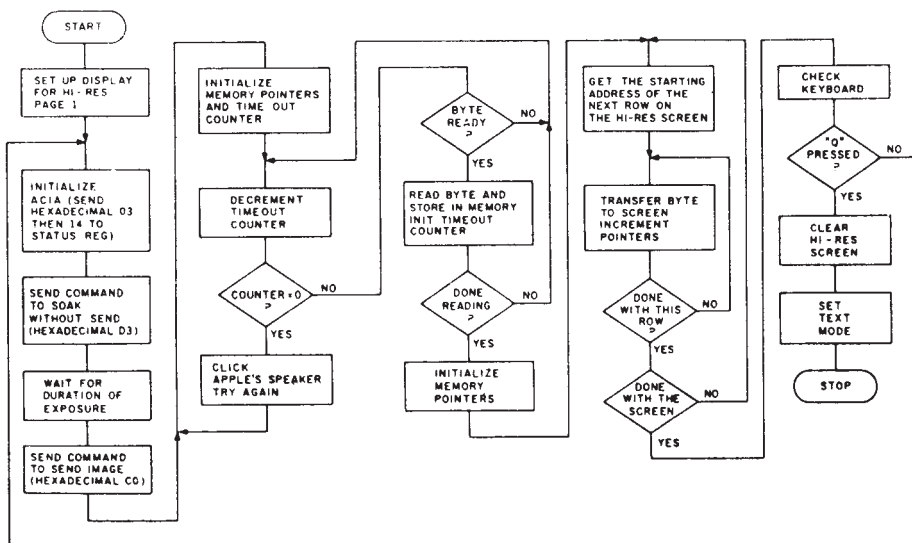


圖 11 適用於 Apple II 的 Micro D-Cam 示範程式的流程圖。這個示範程式含有一個 BASIC 主程式 (如列表 1a 所示), 以及一個 6502 機械語言常式 (如列表 1b 所示)

命令字元	產生的控制動作
N	將影像以正常大小顯示出來 (256 × 64)
F	將影像以全尺寸顯示出來 (256 × 128)
G	產生一個具有 15 種灰度位準且大小為 256 × 128 的影像 (這個程序大約需 30 sec 且會顯示一個曝光倒數計時由 F 至 0)。
E	改變現顯示影像的曝光時間，影像灰度 (gray scale) 的上限或影像灰度的下限
S	將現顯示影像存入磁片中 (可以有三種型態：normal、 full or gray)
Q	停止程式且回到 BASIC

表 5 列表 3 的 GREY 16 程式所提供給使用者的一些命令選擇

將會呈現完全白色。像上述這些狀況，可藉著調整曝光時間的長、短或改變攝影機的孔徑來補救。當上述一切程序調整無誤後，則應能看到一個清楚的影像顯示。

灰度 (gray-scale) 部份的程式示範，僅是稍稍對軟體做一些加強，以允許你產生一個具有十四中間位準亮度的影像且將影像顯示在 Apple 的高解像螢幕上。例如，照片 5 所示的汽車影像就是一個例子。

用以在 Apple II Plus 及 IIe 微電腦上顯示灰度 (gray-scale) 影像的技術被稱為 “ordered dithering”，在這種技術中，影像的濃淡特性是由多重的黑或白影像所組成。在這種技術處理程序中，Micro D-Cam 需對同一目標物取得 15 次曝光 (exposures)，每一次曝光都要較前一次要長一些。(這通常僅需要幾秒鐘)。在每一個曝光完成後，影像中的每個圖素 (pixel) 都要被檢查，假如圖素是表示亮 (顯示 1 的值，即亮度在曝光臨界之上)，則對應此圖素之記數位置之內含值會被加 1。在 15 次程序結束後，會產生一個表格 (由許多值組成)，表格中的每個值均表示其所對應圖素的光強度。例如，假如某一圖素的最

終值是 15，則此圖素應顯示為最亮；假如圖素的值是 8，則此圖素所顯示的亮度應介於黑白之間。

一旦上面所述的圖素強度表 (pixel-intensity table) 建立好之後，則一個 4 × 4 的矩陣 (dither matrix) 會被用來螢幕上每個位置的顯示值 (一個二進制圖素矩陣)，在本文所介紹的軟體，矩陣如下：

0	8	2	10
12	4	14	6
3	11	1	9
15	7	13	5

接著上面程序之後，在表格中的最後影像之大小值會被拿來和矩陣中的陣列元素值相比較。假如影像的圖素值要比陣列元素值大，則此陣列元素即被點亮。假如強度值是 0，則沒有任何陣列元素被點亮；假如強度值是 8，則陣列元素 0 至 8 被點亮；且假如強度值是 15，則所有的陣列元素都被點亮。以此種方法，即可產生 15 種亮度 (luminance) 位準但可能會有一些空間上的漏失。上面程序被重覆的執行，直到螢幕上的每個位置都分配一個值為止

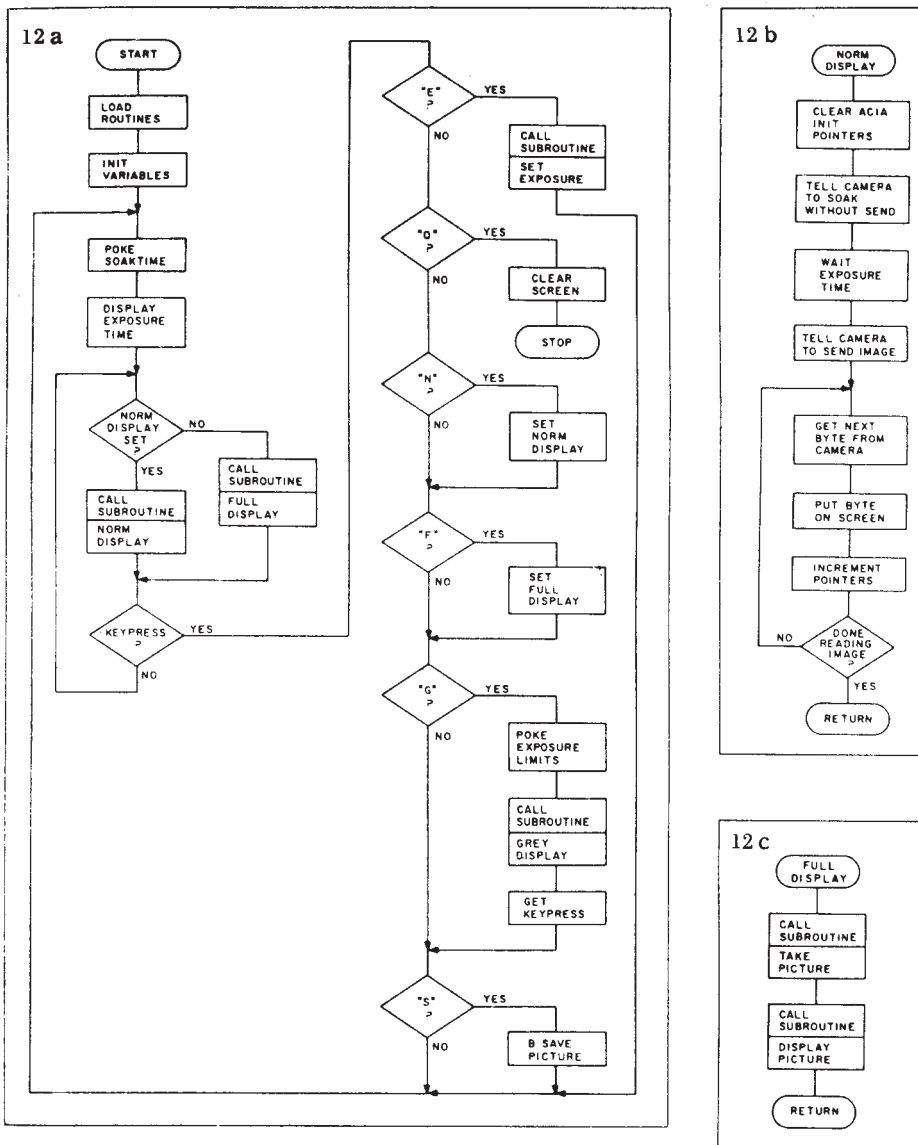
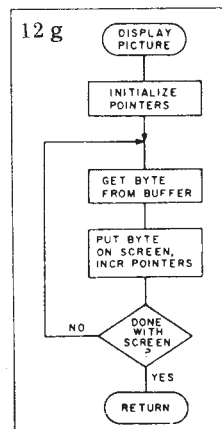
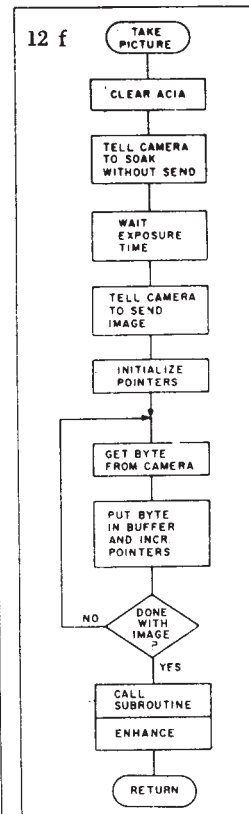
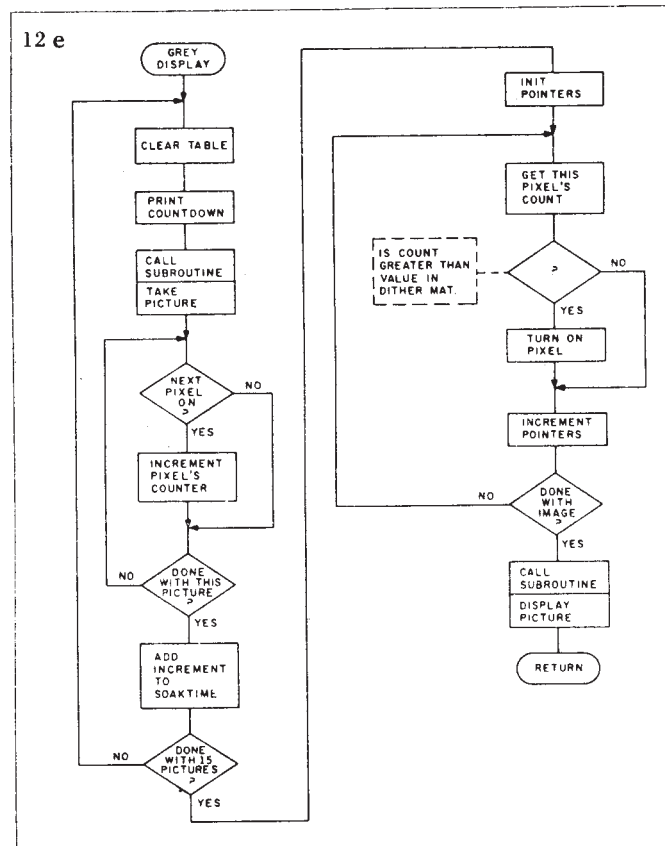
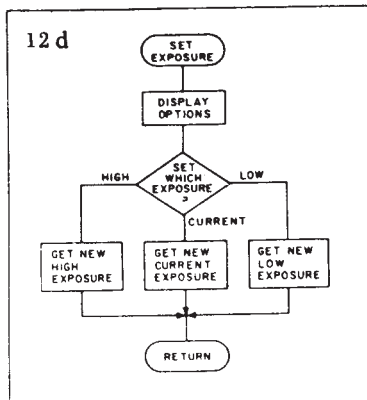
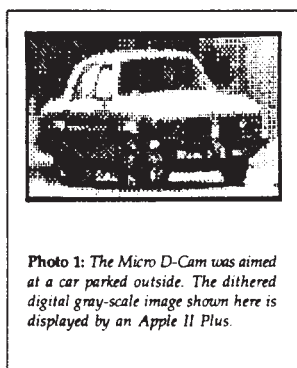
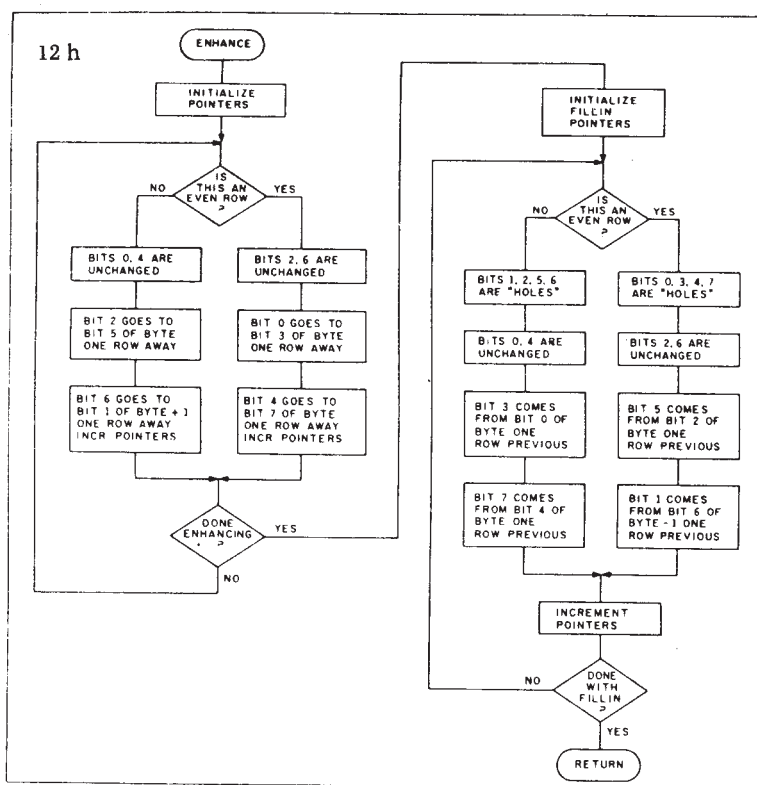


圖 12 適用於 Apple II 微電腦的 GREY 16 程式之流程圖。BASIC 程式如列表 2 a 所示，機械語言程式如列表 2b 所示。主常式 (12 a) 呼叫各個不同的副常式：正常顯示—NORM DISPLAY (12 b)，全顯示—FULL DISPLAY (12 c)，設定曝光時間—SET EXPOSURE (12 d)，灰度顯示—GREY DISPLAY (12 e)，照相—TAKE PICTURE (12 f)，顯示影像—DISPLAY PIC (12 g)，加強影像—ENHANCE (12 h)。

灰度顯示—GREY DISPLAY 常式由 15 exposures 取得感知圖素，且將它們轉換成矩陣中的小顯示圖素，以表現中間亮度。



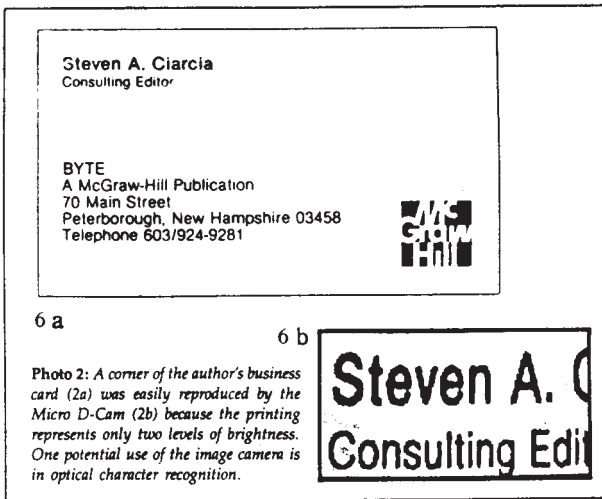


照片 5 由 Micro D-Cam 所攝得的汽車影像此照片是顯示在 Apple II 的高解像螢幕上

由上面的說明知道，假如定義不同大小的矩陣 (dithering matrices)，例如，一個 2×2 矩陣僅可以產生 5 種灰度位準，但却能有較佳的空間定義。對於 8×8 矩陣而言，雖可產生 64 種灰度位準，但却可能有較多空間定義的漏失。

GREY16 程式配合光感知器克服了許多限制。如照片 6 所示的名片 (白底黑字)，僅需使用 Micro D-Cam 就可以很容易的將影像再生。但是由於我們並不是生活在一個黑白的世界，所以對於三度空間的目標物而言，我們必須要有特殊的陰影襯托，才能在二度空間的螢幕顯示出立體效果來。

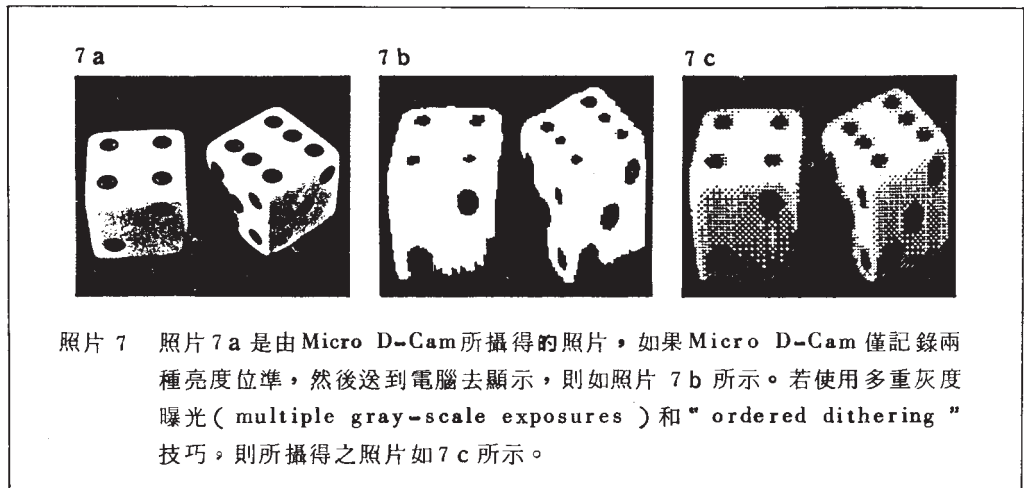
照片 7 所示，主要是用一系列骰子的照片，來說明使用 Micro D-Cam 攝得的三度空間實物的情形。照片 7a 是顯示目標物的顏色和



照片 6：

一張名片如照片(6 a)所示，其名片左上方一角，可以經由 Micro D-Cam 很容易的使影像再生，如照片(6 b)。

因為名片上僅有兩種亮度位準(黑，白)，所以影像攝影機僅需使用光文字識別，即可使影像再生。



亮度情形。假如我們使用 Micro D-Cam (不採用灰度能力—gray scale)，則將會攝得 7 b 所示之照片。

為了要得到更具有代表性的影像顯示，我們可選擇 GREY16 程式中的 G 命令，以得到照片 7 c 的影像顯示，假如影像是由具有顯示濃淡 (half-tones) 能力的電腦所再生，則將如照片 7 a 所示。

結語

從前面 Micro D-Cam 的硬體製作及工作原理，以至於本次的 Micro D-Cam 與 Apple

II，IIe 及 IBM PC 間的界面電路，和控制程式的討論，雖然除了在理論及製作上詳加說明外，在文章中也列舉了許多應用例，但可能仍有某些漏失。故若讀者諸君有什麼新的發現，也歡迎提出來大家一起討論，畢竟一項產品或發明也必須不斷的嘗試或修改，才能達到更完美的境地，您說是嗎……

Listing 1: An assembly-language routine (a) and a BASIC program (b) showing the technique most programmers use to interface two such programs. Note that BASIC USER calls are made directly to RTNO (line 140 of the assembly-language code) and RTN1 (line 340)

(a)

```

00010 ;*****
00020 ;*****
00030 ;** ASSEMBLY LANGUAGE SUBROUTINE **
00040 ;** TO BE CALLED FROM BASIC **
00050 ;*****
00060 ;*****
00070 ;
00080 ; ORG 0F000H ;START AT F000H
00090 ;
00100 ;*****
00110 ;** ROUTINE 0 **
00120 ;*****
00130 ;
00140 RTNO LD A,(DATA1) ;1ST ENTRY POINT
00150 LD B,05H ;IGNORE CODE
F000
F003 0605

```

(a)

```

00010 ;*****
00020 ;*****
00030 ;** ASSEMBLY LANGUAGE SUBROUTINE **
00040 ;** TO BE CALLED FROM BASIC **
00050 ;*****
00060 ;*****
00070 ;
00080 ; ORG 0F000H ;START AT F000H
00090 ;
00100 ;*****
00110 ;** ROUTINE 0 **
00120 ;*****
00130 ;
00140 RTNO LD A,(DATA1) ;1ST ENTRY POINT
00150 LD B,05H ;IGNORE CODE
F000
F003 0605

```

Listing 2: An assembly-language listing (a) and a BASIC program (b) illustrating the impact of the addition of one line of assembly code (line 240). The affected addresses in the BASIC program are underlined.

```

F03F 3244F0 00470 ;*****
F042 C9 00480 ;** ROUTINE 1 **
F043 00 00490 ;** ROUTINE 1 **
F044 00 00500 DATA3 DEFB 00H ;DATA FROM BASIC
0000 00510 DATA4 DEFB 00H ;ANSWER FOR BASIC
00520 ;
00530 ; END OF ROUTINES

```

(b)

```

10 REM ASSEMBLY PROGRAM ALREADY LOADED
20 REM BASIC MEMORY SIZE SET AT 0EFFFH
30 DEFUSR0 = &HF000: REM ENTRY POINT FOR ROUTINE 0
40 DEFUSR1 = &HF025: REM ENTRY POINT FOR ROUTINE 1

```

```

F01F E67F 00240 ;* ADD NEW CODE *
F021 3226F0 00250 LD (DATA2),A ;SAVE ANSWER
F024 C9 00260 RET ;RETURN TO BASIC
F025 00 00270 ;
F026 00 00280 DATA1 DEFB 00H ;DATA FROM BASIC
F026 00 00290 DATA2 DEFB 00H ;ANSWER FOR BASIC
00300 ;
00310 ;*****
00320 ;** ROUTINE 1 **
00330 ;*****
00340 ;
00350 RTN1 LD A,(DATA3) ;2ND ENTRY POINT
F027 3A45F0 00360 SLA A ;IGNORE CODE
F02A CB27

```

```

F041 3246F0 00480 LD (DATA4),A ;SAVE ANSWER
F044 C9 00490 RET ;RETURN TO BASIC
00500 ;
F045 00 00510 DATA3 DEFB 00H ;DATA FROM BASIC
F046 00 00520 DATA4 DEFB 00H ;ANSWER FOR BASIC
00530 ;
00540 ;END OF ROUTINES
00000 END

(b)
10 REM ASSEMBLY PROGRAM ALREADY LOADED
20 REM BASIC MEMORY SIZE SET AT 0EFFFF
30 DEFUSR0 = &HF000: REM ENTRY POINT FOR ROUTINE 0
40 DEFUSR1 = &HF027: REM ENTRY POINT FOR ROUTINE 1
.
.
.
100 POKE &HF025,B: REM SEND DATA TO ROUTINE 0
110 A=USR0(X): REM EXECUTE ROUTINE 0
120 C=PEEK(&HF026): REM GET THE ANSWER
.
.
.
200 POKE &HF045,D: REM SEND DATA TO ROUTINE 1
210 A=USR1(X): REM EXECUTE ROUTINE 1
220 E=PEEK(&HF046): REM GET THE ANSWER
.
.
.
300 END

```

```

00010 ;*****
00020 ;*****
00030 ;** ASSEMBLY LANGUAGE SUBROUTINE *****
00040 ;** TO BE CALLED FROM BASIC *****
00050 ;*****
00060 ;*****
00070 ;*****
00080 ;*****
00090 ;*****
00100 ;*****
00110 ;***** JUMP TABLE *****
00120 ;*****
00130 ;*****
00140 ENTRY0 JP RTN0 ;ROUTINE 0 ENTRY
00150 ENTRY1 JP RTN1 ;ROUTINE 1 ENTRY
00160 ENTRY2 JP 0000H ;SPARE
00170 ENTRY3 JP 0000H ;SPARE
00180 ENTRY4 JP 0000H ;SPARE
00190 ;*****
F000
F000 C313F0
F003 C336F0
F006 C30000
F009 C30000
F00C C30000

```

```

00200 ;*****DATA TABLE*****
00210 ;*
00220 ;*****DATA TABLE*****
00230 ;
FO0F 00 00240 DATA1 DEFB 00H ;ROUTINE 0 DATA
FO10 00 00250 DATA2 DEFB 00H ;ROUTINE 0 ANSWER
FO11 00 00260 DATA3 DEFB 00H ;ROUTINE 1 DATA
FO12 00 00270 DATA4 DEFB 00H ;ROUTINE 1 ANSWER
00280 ;
00290 ;*****ROUTINE 0*****
00300 ;*
00310 ;*****ROUTINE 0*****
00320 ;
FO13 3A0FF0 00330 RTN0 LD A,{DATA1} ;1ST ENTRY POINT
FO16 0605 00340 LD B,05H ;IGNORE CODE
;
;
00430 LD (DATA2),A ;SAVE ANSWER
00440 RET ;RETURN TO BASIC
00450 ;
00460 ;*****ROUTINE 1*****
00470 ;* ROUTINE 1
00480 ;*****ROUTINE 1*****
00490 ;
FO36 3A11F0 00500 RTN1 LD A,{DATA3} ;2ND ENTRY POINT
FO39 CB27 00510 SLA A ;IGNORE CODE
;
;
00630 LD (DATA4),A ;SAVE ANSWER
00640 RET ;RETURN TO BASIC
00650 ;
00660 ; END OF ROUTINES
(b)
10 REM ASSEMBLY PROGRAM ALREADY LOADED
20 REM BASIC MEMORY SIZE SET AT 0EPPFH
30 DEFUSR0 = $HF000: REM JUMP TABLE ENTRY FOR ROUTINE 0
40 DEFUSR1 = $HF003: REM JUMP TABLE ENTRY FOR ROUTINE 1
;
;
100 POKE $HF00F,B: REM SEND ROUTINE 0 DATA TO DATA TABLE
110 A=USR0(X): REM EXECUTE ROUTINE 0
120 C=PEEK($HF010): REM GET THE ANSWER FROM DATA TABLE
;
;
200 POKE $HF011,D: REM SEND ROUTINE 1 DATA TO DATA TABLE
210 A=USR1(X): REM EXECUTE ROUTINE 1
220 E=PEEK($HF012): REM GET THE ANSWER FROM DATA TABLE
;
;
300 END

```

```

F052 3212F0 00640 LD (DATA4),A ;SAVE ANSWER
F055 C9 00650 RET ;RETURN TO BASIC
00660 ;
00670 ;
0000 00670 END ;END OF ROUTINES

```

Listing 5: An assembly-language listing (a) and a BASIC program (b) that illustrate expansion of the number of USER calls available from BASIC to 255. The assembly-language listing shows the jump-table technique used in conjunction with the CLCINX routine (lines 340 to 450) to calculate the desired entry point.

```

(a) 00010 ;*****
00020 ;*****
00030 ;** ASSEMBLY LANGUAGE SUBROUTINE **
00040 ;** TO BE CALLED FROM BASIC **
00050 ;*****
00060 ;*****
00070 ;*****
F000 ;*****
00080 ;*****
00090 ;*****
01100 ;*****
01110 ;***** JUMP TABLE *****
01120 ;*****

```

```

F000 C314F0 00130 ; 00240 INDEX DEF8 00H ;NO. OF ROUTINE
F003 C326F0 00150 ENTRY1 JP RTN1 ;ROUTINE 1 DATA
F006 C348F0 00160 ENTRY2 JP RTN2 ;ROUTINE 1 ANSWER
F009 C30000 00170 ENTRY3 JP 0000H ;ROUTINE 2 DATA
F00C C30000 00180 ENTRY4 JP 0000H ;ROUTINE 2 ANSWER
F013 000000 00190 ; 00280 DATA4 DEF8 00H ;ROUTINE 2 ANSWER
F016 000000 00200 ; ***** ;*****
F019 000000 00210 ; * DATA TABLE ;*****
F022 000000 00220 ; ***** ;*****
F025 000000 00230 ; ***** ;*****
F028 000000 00240 INDEX DEF8 00H ;NO. OF ROUTINE
F031 00 00250 DATA1 DEF8 00H ;ROUTINE 1 DATA
F034 00 00260 DATA2 DEF8 00H ;ROUTINE 1 ANSWER
F037 00 00270 DATA3 DEF8 00H ;ROUTINE 2 DATA
F040 00 00280 DATA4 DEF8 00H ;ROUTINE 2 ANSWER
F043 00 00290 DATA5 DEF8 00H ;ROUTINE 2 ANSWER

```

```

00310 ;* CALCULATE ENTRY FROM INDEX *****
00320 ;*****
00330 ;*****

```

F017 B7	00350	OR A	;IS IT ZERO?
F018 C8	00360	RET Z	;IF YES, RETURN
F019 2600	00370	LD H,00H	;ZERO H REGISTER
F01B 6F	00380	LD L,A	;COPY INDEX TO L
F01C 1600	00390	LD D,00H	;ZERO D REGISTER
F01E 5F	00400	LD E,A	;COPY INDEX TO E

```

;ZERO D REGISTER
LD E,00H
F01C 1600 00390
F01E 5F 00400
;COPY INDEX TO E
LD E,A
F01F 19 00410
;DOUBLE INDEX
ADD HL,DE
F020 19 00420
;HL = 3 X (INDEX)
ADD HL,DE
F021 100F0 00430
;JUMP TABLE START
LD E,ENTRY0

```

Listing 5a continued:

```

00440      ADD HL,DE      ;HL = ENTRY POINT
00450      JP (HL)        ;GO TO JUMP TABLE
00460      ;*****
00470      ;***** ROUTINE 1 *****
00480      ;*****
00490      ;*****
00500      ;*****
00510 RTN1 LD A,(DATA1)  ;2ND ENTRY POINT
00520      LD B,05H        ;IGNORE CODE
      .
      .
      .
00610      LD (DATA2),A   ;SAVE ANSWER
00620      RET           ;RETURN TO BASIC
00630      ;*****
00640      ;***** ROUTINE 2 *****
00650      ;*****
00660      ;*****
00670      ;*****
00680 RTN2 LD A,(DATA3)  ;3RD ENTRY POINT
00690      SLA A          ;IGNORE CODE
      .
      .
      .
00810      LD (DATA4),A   ;SAVE ANSWER
00820      RET           ;RETURN TO BASIC
00830      ;*****
00840      END             ;END OF ROUTINES

(b)
10 REM ASSEMBLY PROGRAM ALREADY LOADED
20 REM BASIC MEMORY SIZE SET AT 00FFFFH
30 DEFUSR = &HF000: REM ENTRY POINT FOR INDEX ROUTINE
40 REM POKE NUMBER OF DESIRED ROUTINE AT 0F00FH
50 REM AND CALL USER0 TO EXECUTE
      .
      .
      .
100 POKE &HF010,B: REM SEND DATA TO ROUTINE 1
110 POKE &HF00F,1: A=USR(X): REM EXECUTE ROUTINE 1
120 C=PEEK(&HF011): REM GET THE ANSWER
      .
      .
      .
200 POKE &HF012,D: REM SEND DATA TO ROUTINE 2
210 POKE &HF00F,2: A=USR(X): REM EXECUTE ROUTINE 2
220 E=PEEK(&HF013): REM GET THE ANSWER
      .
      .
      .
300 END

```

Listing 6: A jump table of the BIOS for Digital Research's CP/M 2.2 operating system. The routines accessed from the jump table are customized for a specific hardware environment.

```

00010 ;*****
00020 ;***** EXAMPLE OF BIOS JUMP TABLE *****
00030 ;** FOR DIGITAL RESEARCH CP/M 2.2 **
00040 ;**
00050 ;*****
00060 ;*****
00070 ;
00080      ORG 4A00H      ;BASE OF BIOS FOR
00090      ; 20K SYSTEM
00100      ;
00110      JP BOOT       ;COLD START ENTRY
00120      JP WBOOT     ;WARM START
00130      JP CONST     ;CONSOLE IN STATUS
00140      JP CONIN     ;CONSOLE INPUT
00150      JP CONOUT     ;CONSOLE OUTPUT
00160      JP LIST     ;LIST DEVICE OUT
00170      JP PUNCH     ;PUNCH OUTPUT
00180      JP READER     ;READER INPUT
00190      JP HOME      ;DISK TO TRACK 0
00200      JP SELDSK    ;SELECT DISK
00210      JP SETTRK    ;SET TRACK NO.
00220      JP SETSEC    ;SET SECTOR NO.
00230      JP SETDMA    ;SET BUFFER ADDR
00240      JP READ      ;READ SECTOR
00250      JP WRITE     ;WRITE SECTOR
00260      JP LISTST    ;GET LIST STATUS
00270      JP SECTAN    ;TRANSLATE SECTOR
00280 ;*****
00290 ;***** BIOS ROUTINES GO HERE *****
00300 ;**
00310 ;*****
00320 ;
00330 ;
      .
      .
      .
00950      END          ;END OF BIOS

```

Listing 1a: The BASIC-language portion of the routines to test and demonstrate the Micro D-Cam.

```

10 REM MICRO DCAM DEMONSTRATION
15 REM PROGRAM
20 REM
25 REM COPYRIGHT (C) 1983 BY
30 REM CIRCUIT CELLAR, INC.
35 REM
40 HGR : TEXT : HOME
50 PRINT CHR$ (4)*BLOAD MICRO D
   CAM*
60 INPUT "ENTER CAMERA SLOT: ";S
   L
70 IF SL < 1 OR SL > 7 THEN GO
80 POKE 770,SL : 16: REM SLOT NU
   MBER
90 POKE 768,0: POKE 769,1: REM E
   XPOSURE TIME
100 POKE 771,0: REM UPPER 1/3 OF
   SCREEN
110 CALL 4096: REM UPDATE SCRIN
120 IF PEEK (773) = 0 THEN 130:
   REM CHECK FOR KEYPRESS
130 IF PEEK (772) = 209 THEN TEXT
   : HOME : END : REM CHECK FOR
   'D'
140 GOTO 110

```

Listing 1b: Micro D-Cam control subroutines, written in 6502 assembly language, called as a machine-language module by the BASIC routine of listing 1.

```

:ASH
1  ****
2  * MICRO DCAM SUPPORT ROUTINES
3  *
4  * COPYRIGHT (C) 1983
5  * BY CIRCUIT CELLAR, INC.
6  *
7  * WHEN CALLED, THIS ROUTINE READS
8  * AN IMAGE FROM THE CAMERA AND
9  * DISPLAYS IT ON THE APPLE'S HI-RES
10 * GRAPHICS SCREEN
11 ****

```

12	ORG \$1000	
13	KEYCLR	EQU \$C010
14	KEYHIT	EQU \$C000
15	BEEP	EQU \$C030
16	GMODE	EQU \$C050
17	TMODE	EQU \$C051
18	MIXED	EQU \$C053
19	PAGE1	EQU \$C054
20	HGR	EQU \$C057
21	STATUS	EQU \$C08E
22	DATA	EQU \$C08F
23		
24	SOAKTIME	EQU \$300
25	SLOTADR	EQU \$302
26	ROWSTART	EQU \$303
27	KEY	EQU \$304
28	KEYEXIT	EQU \$305
29		
30	RADR	EQU \$06
31	CTR	EQU \$08
32	YREG	EQU \$19
33		
1000:	20 B9 10 34	JSR SETGR
1003:	20 8F 10 35	JSR ACTACLR
1006:	A9 D3 36	LDA #9D3
1008:	20 C6 10 37	JSR SENDCMD
100B:	20 DA 10 38	JSR SOAK
100E:	A9 C0 39	LDA #4C0
1010:	20 C6 10 40	JSR SENDCMD
1013:	A2 00 41	LDX #0
1015:	A0 00 42	LDY #0
1017:	8D 08 11 43	LDA ROMPTR,X
101A:	18 44	CLC
101B:	6D 03 03 45	ADC ROWSTART
101E:	85 06 46	STA RADR
1020:	E8 47	INX
1021:	8D 08 11 48	LDA ROMPTR,X
1024:	85 07 49	STA RADR+1
1026:	E8 50	INX
1027:	84 19 51	BET
1029:	AC 02 03 52	LDY SLOTADR
102C:	B9 8E C0 53	LDA STATUS,Y

```

;SET UP FOR HIRES PAGE1
;FLUSH THE INPUT BUFFER
;SEND CMD TO SOAK W/O SEND

;SEND IMAGE W/O SOAK
;(ALT,WIDEPIX,7BIT-25X64)
;INITIALIZE THE ROW INDEX
;INIT COLUMN INDEX (Y)
;BUILD BASE ADDRS FOR CUR ROW

;0-SELECT UPPER 1/3 OF SCREEN
;#28-HID 1/3,#50-BOT 1/3
;RADR HAS ADDRS OF CUR ROW

;PNT X TO HIT ADDRS IN ROMPTR
;GET NEXT BYTE FROM CAMERA
;LOAD OFFSET TO CAMERA SLOT
;CHECK IF NEXT BYTE ARRIVED

```

Listing 1b continued:

102F: 4A	54	LSR		106E: 60	96	KEY1	RTS
1030: B0 21	55	BCC NORM3		108F: A9 03	98	ACTIACLR	LDA #3
1032: A9 00	56	LDA #0		1091: 84 19	99		STY YREG
1034: 85 08	57	STA CTR		1093: AC 02 03	100		LDY SLOADR
1036: A9 15	58	LDA #15		1096: 99 8E C0	101		STA STATUS,Y
1038: 85 09	59	STA CTR+1		1099: A9 14	102		LDA #14
103A: C6 08	60	DEC CTR		109B: 99 8E C0	103		STA STATUS,Y
103C: D0 0F	61	BNE NORM2		109E: A4 19	104		LDY YREG
103E: C6 09	62	DEC CTR+1		10A0: 60	105		RTS
1040: D0 08	63	BNE NORM2					
1042: AD 30 C0	64	LDA DEEP			106		
1045: AD 00 C0	65	LDA KEYHIT		10A1: A2 00	107	GRCLR	LDX #0
1048: 30 10	66	BMI MOONE		10A3: A0 00	108		LDY #0
104A: 4C 03 10	67	JMP NSTART		10A5: 84 06	109		STY RADR
104D: B9 8E C0	68	LDA STATUS,Y		10A7: A9 20	110		LDA #20
1050: 4A	69	LSR		10A9: 85 07	111		STA RADR+1
1051: 90 E7	70	BCC NORM1		10AB: 8A	112		TXA
1053: B9 8E C0	71	LDA DATA,Y		10AC: 91 06	113	CLR1	STA (RADR),Y
1056: A4 19	72	LDY YREG		10AE: C8	114		INY
1058: C0 28	73	CPY #40		10AF: D0 FB	115		BNE CLR1
105A: 80 02	74	BGE NORM4		10B1: E6 07	116		INC RADR+1
105C: 91 06	75	STA (RADR),Y		10B3: E8	117		INX
105E: C8	76	INY		10B4: E0 20	118		CPX #20
105F: C0 25	77	CPY #37		10B6: D0 F4	119		BNE CLR1
1061: D0 C4	78	BNE GET		10B8: 60	120		RTS
1063: E0 80	79	CPY #80			121		
1065: D0 AE	80	BNE NEWROM		10B9: AD 53 C0	122	SETBR	LDA MIXED
1067: A9 D1	81	LDA #D1		10BC: AD 57 C0	123		LDA HGR
1069: 20 C6 10	82	JSR SENDCMD		10BF: AD 54 C0	124		LDA PAGE1
106C: A9 20	83	LDA #20		10C2: AD 50 C0	125		LDA SRODE
106E: 8D 04 03	84	STA KEY		10C5: 60	126		RTS
1071: A9 00	85	LDA #0			127		
1073: 8D 05 03	86	STA KEYEXIT		10C6: 84 19	128	SENDCHD	STY YREG
1076: AD 00 C0	87	LDA KEYHIT		10C8: AC 02 03	129		LDY SLOADR
1079: 10 13	88	BPL KEY1		10CB: 48	130		PHA
107B: 2C 10 C0	89	BIT KEYCLR		10CC: B9 8E C0	131	SEND1	LDA STATUS,Y
107E: EE 05 03	90	INC KEYEXIT		10CF: 29 02	132		AND #2
1081: 8D 04 03	91	STA KEY		10D1: F0 F9	133		BEO SEND1
1084: C9 D1	92	CMP #9*		10D3: 68	134		PLA
1086: D0 06	93	BNE KEY1		10D4: 99 8F C0	135		STA DATA,Y
1088: 20 A1 10	94	JSR GRCLR		10D7: A4 19	136		LDY YREG
108B: AD 51 C0	95	LDA THODE		10D9: 60	137		RTS

; IF BYTE AVAILABLE BRANCH
 ; IF BYTE NOT YET AVAILABLE
 ; SET UP TIMEOUT COUNTER
 ; CHK FOR BYTE UNTIL TIMED OUT
 ; IF TIMED OUT, CLICK SPEAKER
 ; CHECK FOR KEYPRESS
 ; IF KEY HIT THEN RETURN
 ; OTHERWISE, TRY COMMAND AGAIN
 ; WHEN BYTE AVAILABLE GET IT
 ; RESTORE COL POINTER TO Y
 ; IF PAST 40TH BYTE IN CUR ROW,
 ; DON'T PUT ONTO HIRES SCREEN
 ; INCREMENT COLUMN POINTER
 ; END OF THE COLUMN?
 ; IF NOT, GET THE NEXT BYTE
 ; OTHERWISE
 ; IF NOT DONE, GOTO NEXTROW
 ; CLR 'EXIT CAUSED BY KEY' FLAG
 ; AND BLANK THE KEY VALUE
 ; CHECK -IF KEY WAS HIT
 ; CLEAR THE KEYBOARD STROBE
 ; SET 'EXIT CAUSED BY KEY' FLAG
 ; IF THE KEY WAS A 'Q'
 ; CLEAR THE GRAPHICS SCREEN
 ; RETURN TO TEXT MODE

Listing 1b continued:

138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199																																																																																																										
100A: AD 01 03 139	100B: 85 09 140	100C: 85 09 140	100D: 85 09 140	100E: 85 09 140	100F: 85 09 140	100G: 85 09 140	100H: 85 09 140	100I: 85 09 140	100J: 85 09 140	100K: 85 09 140	100L: 85 09 140	100M: 85 09 140	100N: 85 09 140	100O: 85 09 140	100P: 85 09 140	100Q: 85 09 140	100R: 85 09 140	100S: 85 09 140	100T: 85 09 140	100U: 85 09 140	100V: 85 09 140	100W: 85 09 140	100X: 85 09 140	100Y: 85 09 140	100Z: 85 09 140	101A: 85 09 140	101B: 85 09 140	101C: 85 09 140	101D: 85 09 140	101E: 85 09 140	101F: 85 09 140	101G: 85 09 140	101H: 85 09 140	101I: 85 09 140	101J: 85 09 140	101K: 85 09 140	101L: 85 09 140	101M: 85 09 140	101N: 85 09 140	101O: 85 09 140	101P: 85 09 140	101Q: 85 09 140	101R: 85 09 140	101S: 85 09 140	101T: 85 09 140	101U: 85 09 140	101V: 85 09 140	101W: 85 09 140	101X: 85 09 140	101Y: 85 09 140	101Z: 85 09 140	102A: 85 09 140	102B: 85 09 140	102C: 85 09 140	102D: 85 09 140	102E: 85 09 140	102F: 85 09 140	102G: 85 09 140	102H: 85 09 140	102I: 85 09 140	102J: 85 09 140	102K: 85 09 140	102L: 85 09 140	102M: 85 09 140	102N: 85 09 140	102O: 85 09 140	102P: 85 09 140	102Q: 85 09 140	102R: 85 09 140	102S: 85 09 140	102T: 85 09 140	102U: 85 09 140	102V: 85 09 140	102W: 85 09 140	102X: 85 09 140	102Y: 85 09 140	102Z: 85 09 140	103A: 85 09 140	103B: 85 09 140	103C: 85 09 140	103D: 85 09 140	103E: 85 09 140	103F: 85 09 140	103G: 85 09 140	103H: 85 09 140	103I: 85 09 140	103J: 85 09 140	103K: 85 09 140	103L: 85 09 140	103M: 85 09 140	103N: 85 09 140	103O: 85 09 140	103P: 85 09 140	103Q: 85 09 140	103R: 85 09 140	103S: 85 09 140	103T: 85 09 140	103U: 85 09 140	103V: 85 09 140	103W: 85 09 140	103X: 85 09 140	103Y: 85 09 140	103Z: 85 09 140	104A: 85 09 140	104B: 85 09 140	104C: 85 09 140	104D: 85 09 140	104E: 85 09 140	104F: 85 09 140	104G: 85 09 140	104H: 85 09 140	104I: 85 09 140	104J: 85 09 140	104K: 85 09 140	104L: 85 09 140	104M: 85 09 140	104N: 85 09 140	104O: 85 09 140	104P: 85 09 140	104Q: 85 09 140	104R: 85 09 140	104S: 85 09 140	104T: 85 09 140	104U: 85 09 140	104V: 85 09 140	104W: 85 09 140	104X: 85 09 140	104Y: 85 09 140	104Z: 85 09 140	105A: 85 09 140	105B: 85 09 140	105C: 85 09 140	105D: 85 09 140	105E: 85 09 140	105F: 85 09 140	105G: 85 09 140	105H: 85 09 140	105I: 85 09 140	105J: 85 09 140	105K: 85 09 140	105L: 85 09 140	105M: 85 09 140	105N: 85 09 140	105O: 85 09 140	105P: 85 09 140	105Q: 85 09 140	105R: 85 09 140	105S: 85 09 140	105T: 85 09 140	105U: 85 09 140	105V: 85 09 140	105W: 85 09 140	105X: 85 09 140	105Y: 85 09 140	105Z: 85 09 140	106A: 85 09 140	106B: 85 09 140	106C: 85 09 140	106D: 85 09 140	106E: 85 09 140	106F: 85 09 140	106G: 85 09 140	106H: 85 09 140	106I: 85 09 140	106J: 85 09 140	106K: 85 09 140	10

Listing 1b continued:

```

11A7: 3C
11A6: 28 21 28
11A8: 25 28 29
11AE: 28 2D 28
1181: 31 28 35
1184: 28 39 28
1187: 3D
1188: A8 21 A8
1189: 25 A8 29
118E: A8 2D A8
11C1: 31 A8 35
11C4: A8 39 A8
11C7: 3D
11C8: 28 22 28
11C9: 26 28 2A
11CE: 28 2E 28
11D1: 32 28 36
11D4: 28 3A 28
11D7: 3E
11D8: A8 22 A8
11D9: 26 A8 2A
11DE: A8 2E A8
11E1: 32 A8 36
11E4: A8 3A A8
11E7: 3E
11E8: 28 23 28
11EB: 27 28 28
11EE: 28 2F 28
11F1: 33 28 37
11F4: 28 3B 28
11F7: 3F
11F8: A8 23 A8
11FB: 27 A8 28
11FE: A8 2F A8
1201: 33 A8 37
1204: A8 3B A8
1207: 3F

--END ASSEMBLY--
ERRORS: 0

```

HEX A820A824A828A82CA830A834A838A83C

HEX 282128252829282D283128352839283D

HEX A821A825A829A82DA831A835A839A83D

HEX 28222826282A282E28322836283A283E

HEX A822A826A82AA82EA832A836A83AA83E

HEX 28232827282B282F28332837283B283F

HEX A823A827A82BA82FA833A837A83BA83F

520 BYTES

SYMBOL TABLE - ALPHABETICAL ORDER:

ACTACLR	= \$10BF	BEEP	= \$C030	CLR1	= \$10AC	CTR	= \$08
DATA	= \$C0BF	GET	= \$1027	GNODE	= \$C050	GRCLR	= \$10A1
HGR	= \$C057	KEY	= \$0304	KEY1	= \$10BE	KEYCLR	= \$C010
KEYEXIT	= \$0305	KEYHIT	= \$C000	MIXED	= \$C053	MSEC	= \$10FE
MSEC1	= \$1102	NDONE	= \$1067	NEWR0M	= \$1015	NORM1	= \$103A
NORM2	= \$104D	NORM3	= \$1053	NORM4	= \$105E	NSTART	= \$1003
PAGE1	= \$C054	RADR	= \$06	ROMPTR	= \$1108	ROWSTART	= \$0303
SEND1	= \$10CC	SENDMD	= \$10C6	SETGR	= \$10B9	SLOTADR	= \$0302
SOAK	= \$10DA	SOAK1	= \$10F2	SOAK2	= \$10FD	SOAKTIME	= \$0300
STATUS	= \$C08E	THODE	= \$C051	YREG	= \$19		

SYMBOL TABLE - NUMERICAL ORDER:

RADR	= \$06	CTR	= \$08	YREG	= \$19	SOAKTIME	= \$0300
SLOTADR	= \$0302	ROWSTART	= \$0303	KEY	= \$0304	KEYEXIT	= \$0305
NSTART	= \$1003	NEWR0M	= \$1015	GET	= \$1027	NORM1	= \$103A
NORM2	= \$104D	NORM3	= \$1053	NORM4	= \$105E	NDONE	= \$1067
KEY1	= \$10BE	ACTACLR	= \$10BF	GRCLR	= \$10A1	CLR1	= \$10AC
SETGR	= \$10B9	SENDMD	= \$10C6	SEND1	= \$10CC	SOAK	= \$10DA
SOAK1	= \$10F2	SOAK2	= \$10FD	MSEC	= \$10FE	MSEC1	= \$1102
ROMPTR	= \$1108	KEYHIT	= \$C000	KEYCLR	= \$C010	BEEP	= \$C030
GNODE	= \$C050	THODE	= \$C051	MIXED	= \$C053	PAGE1	= \$C054
HGR	= \$C057	STATUS	= \$C08E	DATA	= \$C08F		

Listing 2a: The BASIC portion of the GREY76 program that produces dithered gray-scale output on the Apple II's video screen from the Micro D-Cam's output.

```

10 REM GREY16BAS
12 REM
13 REM COPYRIGHT (C) 1983
15 REM BY CIRCUIT CELLAR, INC.
17 REM
18 REM *LOAD ROUTINES AND INITIALIZE VARIABLES
19 REM

```

Listing 2a continued.

```

20 HDR : TEXT : HOME : PRINT CHR$
  (4)*BL0AD GREY16-4BK*
40 NRM = 4096:FULL = 4099:GY = 41
  02
50 CAM = NRM
60 SOAK = 256:LO = 128:HI = 384
70 INC = (HI - LO) / 15
80 SL = 768:SH = 769:ST = 770:SP =
  771:KEY = 772:KP = 773:EI =
  774
90 HOME : INPUT "ENTER CAMERA S
  LOT: ";SN: IF SN < 1 OR SN >
  7 THEN 90
100 POKE ST,SN & 20: REM "SLOT N
  UMBER
110 POKE SP,0: REM "SCREEN POSIT
  ION
117 REM
118 REM "KEYBOARD PROCESSOR SEC
119 REM
120 POKE SH, INT (SOAK / 256): POKE
  SL,SOAK - INT (SOAK / 256) &
  256: REM "SOAK TIME
130 HOME : VTAB 21: PRINT TAB(
  11);"CURRENT EXP: "SOAK" MS
  "; PRINT : PRINT "LO EXP: "
  ;LO;" MS "; TAB( 24);"HI EX
  P: ";HI;" MS "
140 CALL CAM: REM "READ CAMERA A
  ND DISPLAY PICTURE IN CURREN
  T MODE
150 IF NOT ( PEEK (KP)) THEN 14
  0: REM "CHECK FOR KEYPRESS
160 KEYS = CHR$ ( PEEK (KEY) - 1
  28)
170 IF KEYS = "E" THEN 400: REM
  "CHANGED EXPOSURE
180 IF KEYS = "O" THEN TEXT : HOME
  : END : REM "QUIT PROGRAM
190 IF KEYS = "N" THEN CAM = NRM
  : IJR : REM "NORMAL SIZE

```

```

490 LO = NE: GOTO 540
500 IF CH$ < > "H" THEN 400
510 INPUT "ENTER NEW HI: ";NE$: IF
  NE$ = "" THEN 540
520 NE = VAL (NE$): IF NE < 1 OR
  NE > 8000 OR NE < LO THEN 51
  0
530 HI = NE
540 INC = (HI - LO) / 15
550 GOTO 120
597 REM
598 REM "SAVE PICTURE ON DISK
599 REM
600 HOME : VTAB 22: PRINT "ENTER
  A NAME FOR: INPUT "THE PIC
  TURE: ";NA$
610 IF LEN (NA$) < = 0 THEN RETURN
620 PRINT CHR$ (4)"BSAVE "NA$";
  AS$2000,LS$2000"
630 HOME : RETURN

```

Listing 2b: Assembly-language listing of the machine-code portion of the GREY16 program, called from BASIC.

```

1 #####
2 1
3 3 GREY16-4BK
4 1
5 3 COPYRIGHT (C) 1983
6 3 BY CIRCUIT CELLAR, INC.
7 1
8 #####
9 1
10 ORG $1000
11 KEYHIT EQU $C000
12 KEYCLR EQU $C010
13 BEEP EQU $C030
14 SMODE EQU $E050
15 TMODE EQU $C051

```

16	MIXED	EBU	\$C053	1009: 20 34 12 58	MORPIC	JSR	SET6R	;	SET GRAPHICS ON
17	PAGE1	EBU	\$C054	100C: 20 0A 12 59	INSTRT	JSR	ACTACLR	;	FLUSH THE INPUT BUFFER
18	HGR	EBU	\$C057	100F: A9 03 60		LDA	\$#03	;	SEND COMMAND TO SOAK W/O SEND
19	STATUS	EBU	\$C08E	1011: 20 41 12 61		JSR	SENDCHD	;	
20	DATA	EBU	\$C08F	1014: 20 55 12 62		JSR	SOAK	;	WAIT EXPOSURE TIME
21	PBLNWK	EBU	\$F948	1017: A9 C0 63		LDA	\$#C0	;	SEND IMAGE W/O SOAK
22	PRHEX	EBU	\$F9E3	1019: 20 41 12 64		JSR	SENDCHD	;	(ALT, WIDEPIX, 7BIT-256 X 64)
23	↓			101C: A2 00 65		LDX	#0	;	INITIALIZE THE ROW INDEX
24	TRUFRER	EBU	\$3000	101E: A0 00 66	MEMROW	LDY	#0	;	START NXT ROW, INIT COL INDEX
25	IBUFRER	EBU	\$4000	1020: 80 F4 13 67		LDA	ROMPTR,X	;	GET FIRST ROW ADDR
26	GRTABLE	EBU	\$5000	1023: 18 68		CLC		;	
27	↓			1024: 60 03 03 69		ADC	ROWSTART	;	0-SELECT UPPER 1/3 OF SCREEN
28	SOAKTIME	EBU	\$300	1027: 85 06 70		STA	RADR	;	\$28-MID 1/3, \$50-80T 1/3
29	SLOTADR	EBU	\$302	1029: E8 71		INX		;	RADR HAS ADDRS OF CUR ROW
30	ROWSTART	EBU	\$303	102A: 80 F4 13 72		LDA	ROMPTR,X	;	
31	KEY	EBU	\$304	102D: 85 07 73		STA	RADR+1	;	
32	KEYFLAG	EBU	\$305	102F: E8 74		INX		;	POINT X-REG TO NEXT ADDRESS
33	INCREMENT	EBU	\$306	1030: 84 18 75	GET	STY	YREG	;	GET NEXT BYTE FROM CAMERA
34	↓			1032: AC 02 03 76		LDY	SLOTADR	;	LOAD OFFSET TO CAMERA SLOT
35	RADR	EBU	\$06	1035: 89 8E C0 77		LDA	STATUS,Y	;	CHECK IF NXT BYTE HAS ARRIVED
36	DEST	EBU	\$08	1038: 4A 78		LSR		;	
37	DEST2	EBU	\$1C	1039: 80 21 79		BCS	C15	;	IF BYTE AVAILABLE THEN BRANCH
38	CTR	EBU	\$19	103B: A9 00 80		LDA	#0	;	IF BYTE NOT AVAILABLE THEN
39	YREG	EBU	\$1B	103D: 85 19 81		STA	CTR	;	SET UP TIMEOUT COUNTER
40	TMP	EBU	\$1E	103F: A9 15 82		LDA	\$#15	;	
41	COUNT	EBU	\$1F	1041: 85 1A 83		STA	CTR+1	;	
42	TABLE	EBU	\$EB	1043: C6 19 84	C0	DEC	CTR	;	CHECK FOR BYTE TILL TIMED OUT
43	IMAGE	EBU	\$ED	1045: 00 0F 85		BNE	C1	;	
44	KEEPCNT	EBU	\$F9	1047: C6 1A 86		DEC	CTR+1	;	
45	KEEFL6	EBU	\$FA	1049: D0 08 87		BNE	C1	;	
46	↓			104B: AD 30 C0 88		LDA	BEEP	;	IF TIMED OUT, CLICK SPEAKER
47	TEMP1	EBU	\$06	104E: A0 00 C0 89		LDA	KEYHIT	;	CHECK FOR KEYPRESS
48	TEMP2	EBU	\$07	1051: 30 1D 90		BMI	NDONE	;	IF KEY HIT, RETURN TO BASIC
49	TEMP3	EBU	\$1C	1053: 4C 0C 10 91		JMP	INSTRT	;	ELSE, RESTART CMD SEQUENCE
50	TEMP4	EBU	\$1D	1056: 89 8E C0 92	C1	LDA	STATUS,Y	;	
51	TEMP5	EBU	\$1B	1059: 4A 93		LSR		;	
52	↓			105A: 90 E7 94		BCC	C0	;	
1000: 4C 09 10 53		JMP	MORPIC	105C: 89 8F C0 95	C15	LDA	DATA,Y	;	WHEN BYTE AVAILABLE GET IT
1003: 4C 9B 10 54		JMP	FULLPIC	105F: A4 1B 96		LDY	YREG	;	RESTORE COL POINTER TO Y-REG
1006: 4C 15 11 55		JMP	GREY	1061: C0 28 97		CPY	#40	;	IF PAST 40TH BYTE IN CURRENT

Listing 2b continued:

1063: 80 02 98	BGE C3		10C1: 85 F9	139	STA	KEEPCNT
1065: 91 06 99	STA (RADR),Y	:ROM THEN DONT PUT ON SCREEN	10C3: A9 01	140	LDA	#01
1067: C8 100 C3	INY	:INCREMENT COLUMN POINTER	10C5: 85 FA	141	STA	KEEPLG
1068: C0 25 101	CPY #37	:CHECK FOR END OF COLUMN	10C7: A0 00	142	LDA	#0
1069: D0 C4 102	BNE GET	:IF NOT, GET THE NEXT BYTE	10C9: AE 02 03	143	LDA	SLOTADR
106C: E0 80 103	CPX #80	:CHECK FOR END OF IMAGE	10CC: A9 00	144	LDA	#0
106E: D0 AE 104	BNE NEWROM	:GOTO NEXT ROW	10CE: 85 19	145	STA	CTR
1070: A9 D1 105	LDA #D1		10D0: 80 BE C0	146	LDA	STATUS,X
1072: 20 41 12 106	JSR SENDCMD	:REFRESH W/O SENDING	10D3: 4A	147	LSR	
1075: A9 20 107	LDA #20		10D4: C6 19	148	DEC	CTR
1077: 80 04 03 108	STA KEY	:CLEAR 'KEY HIT' VALUE	10D6: D0 06	149	BNE	NOHANG
107A: A9 00 109	LDA #0		10D8: A0 30 C0	150	LDA	REEP
107C: 80 05 03 110	STA KEYFLAG	:CLEAR 'KEY HIT' FLAG	10DA: 4C A4 10	151	JMP	START
107E: A0 00 C0 111	LDA KEYHIT	:LOOK AT KEYBOARD	10DE: 90 EC	152	BCC	GSTAT
1082: 10 13 112	BPL D1	:BRANCH IF NO KEYPRESS	10E0: A5 FA	153	LDA	KEEPLG
1084: 2C 10 C0 113	BIT KEYCLR	:CLEAR KEYBOARD STROBE	10E2: F0 08	154	BEQ	IGNORE
1087: EE 05 03 114	INC KEYFLAG	:SET 'KEY HIT' FLAG	10E4: 80 BF C0	155	LDA	DATA,X
108A: 80 04 03 115	STA KEY	:SAVE KEYPRESS	10E7: 91 06	156	STA	(RADR),Y
108D: C9 D1 116	CMF #Q*	:CHECK IF 'Q' HIT	10E9: 4C EF 10	157	JMP	CONT
108F: D0 06 117	BNE D1	:IF NOT, RETURN	10EC: 80 BF C0	158	IGNORE	
1091: 20 1C 12 118	JSR GRCLR	:CLEAR GRAPHICS SCREEN	10EE: C6 F9	159	DEC	KEEPCNT
1094: A0 51 C0 119	LDA TMODE	:SET TEXT MODE	10F1: D0 0A	160	BNE	CONT1
1097: 60 120 D1	RTS	:RETURN TO BASIC	10F3: A9 20	161	LDA	#20
1098: 20 34 12 122	FULLPIC	:SET GRAPHICS ON	10F5: 85 F9	162	STA	KEEPCNT
109B: 20 A7 10 123	JSR TAKEPIC	:TAKE A PICTURE	10F7: A5 FA	163	LDA	KEEPLG
109E: 20 83 12 124	JSR MOVE	:MOVE TO CORRECT BUFFER	10F9: 49 01	164	EDR	#01
10A1: 20 A3 12 125	JSR DISPLAY	:TRANS IMAGE TO HIRS SCREEN	10FB: 85 FA	165	STA	KEEPLG
10A4: 4C 75 10 126	JMP DOVEY	:CHECK FOR KEYPRESS	10FD: A5 FA	166	LDA	KEEPLG
10A7: 20 0A 12 128	TAYEPTIC	:CLEAR ACTA	10FF: F0 C8	167	BEQ	GSTAT
10AA: A9 FB 129	START		1101: C8	168	INY	
10AC: 20 41 12 130	JSR SENDCMD	:TELL CAMERA TO SOAK W/O SEND	1102: D0 C8	169	BNE	GSTAT
10AF: 26 55 12 131	JSR SOAK	:WAIT EXPOSURE TIME	1104: E6 07	170	INC	RADR+1
10B2: A9 E8 132	LDA #E8	:TELL CAMERA TO SEND IMAGE	1106: A5 07	171	LDA	RADR+1
10B4: 20 41 12 133	JSR SENDCMD	:TELL CAMERA TO SEND IMAGE	1108: C9 50	172	CMF	#50
10B7: A9 00 134	LDA #0	:INDALT, WIDEXPIX, 8BIT-512X128)	110A: D0 C0	173	BNE	GSTAT
10B9: 85 06 135	STA RADR	:SET UP BUFFER ADDRESS	110C: A9 F9	174	LDA	#F9
10BB: A9 40 136	LDA #0		110E: 20 41 12 175	175	JSR	SENDCMD
10BD: 85 07 137	STA RADR+1		1111: 20 F2 12 176	176	JSR	ENHANCE
10BF: A9 20 138	LDA #20	:INIT KEEP COUNTER	1114: 60	177	RTS	
				178		

1115: A9 00	179	GREY	LDA #<RTABLE	;INIT TABLE POINTERS	1167: E6 EB	220	INC TABLE	;INCREMENT TABLE POINTER
1117: 85 EB	180		STA TABLE		1169: D0 02	221	BNE TBL1	
1119: A9 50	181		LDA #<RTABLE		1168: E6 EC	222	INC TABLE+1	
1118: 85 EC	182		STA TABLE+1		1160: D0	223	DEX TBL1	;DONE WITH THIS BYTE?
1110: A0 00	183		LDY #00		116E: D0 E3	224	BNE LOADTBL	;INCREMENT BUFFER POINTER
111F: 98	184	CLR1	TVA	;CLEAR GREYSCALE COUNTER TABLE	1170: E6 ED	225	INC IMAGE	
1120: 91 EB	185		STA (TABLE),Y		1172: D0 D9	226	BNE NXTBYTE	
1122: E6 EB	186		INC TABLE		1174: E6 EE	227	INC IMAGE+1	
1124: D0 F9	187		BNE CLR1		1176: A9 40	228	LDA #>BUFFER*1000	
1126: E6 EC	188		INC TABLE+1		1178: C5 EE	229	CMP IMAGE+1	;CHECK FOR END OF BUFFER
1128: A9 90	189		LDA #<RTABLE+44000		117A: D0 D1	230	BNE NXTBYTE	;IF NOT, GET NEXT BYTE
112A: C5 EC	190		CMP TABLE+1	;CHECK FOR END	117C: 18	231	CLC	
112C: D0 F1	191		BNE CLR1		117D: AD 00 03	232	LDA SOAKTIME	;INCREMENT EXPOSURE TIME FOR
112E: A9 0F	192		LDA #0F	;INIT EXPOSURE COUNTER	1180: D0 06 03	233	ADC INCRMENT	;NEXT EXPOSURE
1130: 85 IF	193		STA COUNT		1183: 8D 00 03	234	STA SOAKTIME	
1132: A5 1F	194	NEXTPIC	LDA COUNT		1186: AD 01 03	235	LDA SOAKTIME+1	
1134: 20 E3 F0 195	195		JSR PRHX	;DISPLAY COUNT FOR COUNTDOWN	1189: A9 00	236	ADC #00	
1137: 20 48 F9 196	196		JSR PRBLNK	;LEAVE THREE SPACES	118B: 8D 01 03	237	STA SOAKTIME+1	
113A: 20 A7 10 197	197		JSR TAYEPIC	;TAKE A PICTURE	118E: C6 1F	238	DEC COUNT	;DONE WITH 15 EXPOSURES?
1130: A9 00	198		LDA #<RTABLE	;INIT TABLE AND BUFFER PTRS	1190: D0 40	239	BNE NEXTPIC	;IF NOT, TAKE NEXT PICTURE
113F: 85 ED	199		STA INR6E			240		
1141: 85 EB	200		STA TABLE		1192: A9 00	241	LDA #<RTABLE	;INIT TABLE AND BUFFER PTRS
1143: A9 50	201		LDA #>TBUFFER		1194: 85 EB	242	STA TABLE	
1145: 85 EE	202		STA IMAGE+1		1196: 85 ED	243	STA IMAGE	
1147: A9 50	203		LDA #<RTABLE		1198: A9 50	244	LDA #>RTABLE	
1149: 85 EC	204		STA TABLE+1		119A: 85 EC	245	STA TABLE+1	
114B: A0 00	205		LDY #00		119C: A9 40	246	LDA #>TBUFFER	
114D: A2 04	206	NXTBYTE	LDX #04		119E: 85 EE	247	STA IMAGE+1	
114F: 81 ED	207		LDA (IMAGE),Y	;SET NEXT BYTE	11A0: A2 00	248	LDX #00	
1151: 85 1E	208		STA TMP		11A2: A9 20	249	NEXTROW LDA #20	
1153: 06 1E	209	LOADTBL	ASL TMP	;TEST EACH BIT IN THE BYTE	11A4: 85 1F	250	STA COUNT	;INIT COLUMN COUNTER
1155: A9 00	210		LDA #00		11A6: A0 00	251	THISROW LDY #00	
1157: 90 02	211		BCC ZERO		11A8: 84 1E	252	STY TMP	
1159: A9 01	212		LDA #01		11AA: A9 02	253	LDA #02	
115B: 06 1E	213	ZERO	ASL TMP		11AC: 85 19	254	STA CTR	;INIT BIT COUNTER
115D: 90 03	214		BCC ZERO1		11AE: 81 ER	255	THISBYTE LDA (TABLE),Y	;GET NEXT BYTE
115F: 16	215		CLC		11B0: 29 0F	256	AND #0F	;MASK OFF TOP NIBBLE
1160: A9 10	216		ADC #10		11B2: D0 E4 13	257	CMP VAL1,X	;COMP WITH OTHER MATRIX VAL
1162: 16	217	ZERO1	CLC		11B5: 26 1E	258	ROL TMP	;ROTATE CARRY BIT INTO BYTE
1163: 71 ER	218		ADC (TABLE),Y	;AND INCR APPROPRIATE CNTRS	11B7: 81 EB	259	LDA (TABLE),Y	;GET BYTE AGAIN
1165: 91 EB	219		STA (TABLE),Y	;TWO BIT-COUNTERS PER BYTE)	11B9: 29 F0	260	AND #F0	;MASK OFF LOWER NIBBLE
					11BB: D0 E8 13	261	CMP VAL2,X	;COMP WITH NEXT MATRIX VALUE

Listing 2b continued:

```

118E: 26 1E 262      ROL TMP      ;ROTATE CARRY BIT INTO BYTE
11C0: C9 263        INY          ;INCR TABLE INDEX
11C1: 81 EB 264      LDA (TABLE),Y ;GET NEXT BYTE
11C3: 2F 0F 265      AND #0F      ;MASK OFF UPPER NIBBLE
11C5: D0 EC 13 266   CMP VAL3,X  ;COMP WITH THIRD MATRIX VALUE
11C8: 26 1E 267      ROL TMP      ;ROTATE CARRY INTO BYTE
11CA: 81 EB 268      LDA (TABLE),Y ;GET NEXT BYTE AGAIN
11CC: 2F 0F 269      AND #0F      ;MASK OFF LOWER NIBBLE
11CE: D0 F0 13 270   CMP VAL4,X  ;COMP WITH FOURTH MATRIX VALUE
11D1: 26 1E 271      ROL TMP      ;ROTATE CARRY INTO BYTE
11D3: C8 272        INY          ;INCR TABLE INDEX
11D4: C6 19 273      DEC CTR      ;DECR BIT COUNTER
11D6: D0 D6 274      BNE THIS8YTE ;CONTINUE WITH THIS BYTE
11D8: 40 00 275      LDY #000
11DA: A5 1E 276      LDA TMP      ;GET BYTE FOR NXT COL IN IMAGE
11DC: 91 ED 277      STA (IMAGE),Y ;PUT INTO IMAGE BUFFER
11DE: 18 278        CLC
11DF: A9 04 279      LDA #04
11E1: 65 EB 280      ADC TABLE  ;INCR TABLE POINTER
11E3: 85 EB 281      STA TABLE
11E5: 90 02 282      BCC NEXT
11E7: E6 EC 283      INC TABLE+1 ;INCR BUFFER POINTER
11E9: E6 ED 284      INC IMAGE
11EB: D0 08 285      BNE NXT1
11ED: E6 EE 286      INC IMAGE+1
11EF: A5 EE 287      LDA IMAGE+1
11F1: C9 50 288      CMP #>1000 ;CHECK FOR END OF IMAGE
11F3: 80 0E 289      RGE DONE
11F5: C6 1F 290      DEC COUNT  ;IF NOT, DECR COL COUNTER
11F7: D0 AD 291      BNE THISROW  ;IF NOT END, STAY ON THIS ROW
11F9: EB 292        INX          ;INCR DITHER MATRIX INDEX
11FA: E0 04 293      CPIX #04
11FC: D0 02 294      BNE NXT2
11FE: A2 00 295      LDY #000
1200: 4C A2 11 296   JMP NEXTROW ;DO NEXT ROW
1203: 20 1C 12 297   DOWNE      ;CLEAR GRAPHICS SCREEN
1206: 20 A3 12 298   JSR DISPLAY ;SEND IMAGE TO SCREEN
1209: 60 299        RTS
120A: A9 03 300      LDA #3
120C: 84 1B 302      STY YREG

120E: AC 02 03 303   ;GET CAMERA ADDRESS
1211: 99 BE C0 304   STA STATUS,Y
1214: A9 14 305     LDA #14
1216: 99 BE C0 306   STA STATUS,Y
1219: A4 1B 307     LDY YREG
121B: 60 308        RTS
121C: A2 00 309     LDY #0
121E: A0 00 310     LDY #0
1220: 84 06 311     STY RADR
1222: A9 20 312     LDA #20
1224: 85 07 313     STA RADR+1
1226: 8A 314     TXA
1227: 91 06 315     STA (RADR),Y
1229: C8 316     INY
122A: D0 FB 317     BNE CLR2
122C: E6 07 318     INC RADR+1
122E: EB 319     INX
122F: E0 20 320     CPIX #20
1231: D0 F4 321     BNE CLR2
1233: 60 322     RTS
1234: AD 53 C0 323   SETSR
1237: AD 57 C0 324   SETSR
123A: AD 54 C0 325   LDA MIXED
123D: AD 50 C0 326   LDA HGR
123F: AD 50 C0 327   LDA PAGE1
1240: 60 328     LDA GMODE
1241: 84 1B 329     STY YREG
1243: AC 02 03 330   SENDCMD
1246: 48 331     PHA
1247: B9 BE C0 332   LDA STATUS,Y
124A: 2F 02 333   AND #2
124C: F0 F9 334   BEQ SEND1
124E: 68 335     PLA
124F: 99 8F C0 336   STA DATA,Y
1252: A4 1B 337   LDY YREG
1254: 60 338     RTS
1255: AD 01 03 339   SOAK
1258: 85 1A 340   STA CTR+1
125A: E6 1A 341   INC CTR+1

120E: AC 02 03 303   ;GET CAMERA ADDRESS
1211: 99 BE C0 304   STA STATUS,Y
1214: A9 14 305     LDA #14
1216: 99 BE C0 306   STA STATUS,Y
1219: A4 1B 307     LDY YREG
121B: 60 308        RTS
121C: A2 00 309     LDY #0
121E: A0 00 310     LDY #0
1220: 84 06 311     STY RADR
1222: A9 20 312     LDA #20
1224: 85 07 313     STA RADR+1
1226: 8A 314     TXA
1227: 91 06 315     STA (RADR),Y
1229: C8 316     INY
122A: D0 FB 317     BNE CLR2
122C: E6 07 318     INC RADR+1
122E: EB 319     INX
122F: E0 20 320     CPIX #20
1231: D0 F4 321     BNE CLR2
1233: 60 322     RTS
1234: AD 53 C0 323   SETSR
1237: AD 57 C0 324   SETSR
123A: AD 54 C0 325   LDA MIXED
123D: AD 50 C0 326   LDA HGR
123F: AD 50 C0 327   LDA PAGE1
1240: 60 328     LDA GMODE
1241: 84 1B 329     STY YREG
1243: AC 02 03 330   SENDCMD
1246: 48 331     PHA
1247: B9 BE C0 332   LDA STATUS,Y
124A: 2F 02 333   AND #2
124C: F0 F9 334   BEQ SEND1
124E: 68 335     PLA
124F: 99 8F C0 336   STA DATA,Y
1252: A4 1B 337   LDY YREG
1254: 60 338     RTS
1255: AD 01 03 339   SOAK
1258: 85 1A 340   STA CTR+1
125A: E6 1A 341   INC CTR+1

```

Listing 2b continued

125C: A0 00 03 345	LD A SOAKTIME	STA RADR	386	
125D: 85 19 346	STA CTR	LD A #1BUFFER		
125E: 85 19 347	INC CTR	STA RADR+1	388	
1261: E6 19 347	LD A SOAKTIME	LD A ROMPTR,X		;GET STARTING ADDR OF CUR ROW
1263: A0 00 03 348	LD A SOAKTIME	STA DEST	390	;PUT IN SCREEN POINTER
1266: 00 05 349	BNE SOAK1	INC INX	391	;INCR INDEX
1268: A0 01 03 350	LD A SOAKTIME+1	LD A ROMPTR,X	392	;GET UPPER HALF OF ADDRESS
126B: F0 08 351	BEO SOAK2	STA DEST+1	393	;PUT IN SCREEN POINTER
126C: 20 79 12 352	JSR MSEC	INC INX	394	;INCR INDEX
1270: C6 19 353	DEC CTR	LDY #20	395	;INIT COLUMN COUNTER
1272: D0 F9 354	BNE SOAK1	LD A (RADR),Y	396	;GET BYTE FROM BUFFER
1274: C6 1A 355	DEC CTR+1	DEY	397	;DECR COL COUNTER
1276: D0 F5 356	BNE SOAK1	STA (DEST),Y	398	;PUT BYTE ON SCREEN
1278: 60 357	SOAK2	CPY #0	399	;CHECK IF COUNTER IS ZERO
1279: 84 1B 359	MSEC	BNE MOV	400	;IF NOT, DO NEXT BYTE
127A: 84 1B 360	LDY #199	LDY #0	401	
127B: 88 361	DEY	PHP	402	;ROTATE EACH BYTE IN ROW SO
127C: D0 FD 362	BNE DLY1	STY TMP	403	;THE HIGH BIT IS CARRIED TO
127E: D0 FD 363	LDY YREG	LD A (DEST),Y	404	;THE NEXT BYTE
1282: 60 364	RTS	PLP	405	
1283: A0 00 366	MOVE	ROL	406	
1285: 84 ED 367	LDY #400	STA (DEST),Y	407	
1287: 84 08 368	STY DEST	PHP	408	
1288: 85 EE 370	STA IMAGE+1	INY	409	
1289: A9 30 369	LD A #1BUFFER	CPY #20	410	
128B: A9 40 371	LD A #1BUFFER	BNE SHFT	411	
128C: 85 09 372	STA DEST+1	PLP	412	
1291: 81 ED 373	LD A (IMAGE),Y	LDY TMP	413	
1293: 91 08 374	STA (DEST),Y	LD A (DEST),Y	414	;SHIFT THE LEADING BYTE SO THE
1295: C8 375	INY	LSR	415	;TOP BIT IS ZERO
1296: D0 F9 376	BNE LOOP	STA (DEST),Y	416	
1298: E6 09 378	INC DEST+1	INX	417	
129C: A9 50 379	LD A #1BUFFER+1000	CPY #20	418	;IF NOT DONE, ROTATE ALL OF
129E: C5 09 380	CMR DEST+1	BNE RESHFT	419	;ROW AFTER LEADING BYTE AGAIN
12A0: D0 EF 381	BNE LOOP	CLC	420	
12A2: 60 382	RTS	LD A RADR	421	
12A3: A2 00 384	DISPLAY	ADC #20	422	
12A5: A9 00 385	LD A #1BUFFER	STA RADR	423	
		BCC DISP1	424	
		INC RADR+1	425	
		CPY #0	426	;IF SCRN NOT DONE, DO NEXT ROW
		BNE NITROW	427	

Listing 2b continued:

```

12F1: 60      428      RTS
429      1
430      1
431      1 ENHANCE REARRANGES THE IMAGE IN I$BUFFER TO AN
432      1 IMAGE IN I$BUFFER THAT CORRESPONDS MORE
433      1 PRECISELY TO THE ACTUAL PICTURE BEING SEEN
434      1 BY THE CAMERA.
435      1 THE ALGORITHM REARRANGES EACH BYTE AS FOLLOWS:
436      1 FOR BYTES IN EVEN ROWS(STARTING WITH ROW 0) --
437      1 BITS 2 AND 6 ARE UNCHANGED
438      1 BIT 0 MOVES TO BIT 3 OF THE BYTE 1 NEXT ROW
439      1 BIT 4 MOVES TO BIT 7 OF THE BYTE 1 NEXT ROW
440      1
441      1 FOR BYTES IN ODD ROWS ---
442      1 BITS 0 AND 4 ARE UNCHANGED
443      1 BIT 2 MOVES TO BIT 5 OF THIS BYTE IN NEXT ROW
444      1 BIT 6 MOVES TO BIT 1 OF THIS BYTE+1 IN NEXT ROW
445      1
446      1
12F2: A9 DF      447      ENHANCE LDA #<I$BUFFER-$21 ;INIT BUFFER POINTERS
12F4: 85 06      448      STA RADR
12F6: 85 08      449      STA DEST
12F8: A9 3F      450      LDA #>I$BUFFER-$21
12FA: 85 07      451      STA RADR+1
12FC: A9 2F      452      LDA #>I$BUFFER-$21
12FE: 85 09      453      STA DEST+1
1300: A9 20      454      LDA #>20
1302: 85 1E      455      STA TMP
1304: A0 01      456      NEVEN
1306: 81 06      457      LDA (RADR),Y
1308: 0A         458      ASL
1309: 0A         459      ASL
130A: 0A         460      ASL
130B: 09 DF      461      ORA #0DF
130D: 85 1C      462      STA DEST2
130F: 88         463      DEY
1310: 81 06      464      LDA (RADR),Y
1312: 2A         465      ROL
1313: 2A         466      ROL
1314: 2A         467      ROL
1315: 09 FD      468      ORA #>FD
1317: 85 1D      469      STA DEST2+1

1319: A0 21      470      LDY #>21
131B: 81 06      471      LDA (RADR),Y
131D: 09 88      472      ORA #>88
131F: 29 FF      473      AND #>FF
1321: 25 1C      474      AND DEST2
1323: 25 1D      475      AND DEST2+1
1325: 91 08      476      STA (DEST),Y
1327: E6 06      477      INC RADR
1329: E6 08      478      INC DEST
132B: D0 04      479      BNE MODUB
132D: E6 07      480      INC RADR+1
132F: E6 09      481      INC DEST+1
1331: C6 1E      482      DEC TMP
1333: D0 CF      483      BNE NEVEN
1335: A9 20      484      LDA #>20
1337: 85 1E      485      STA TMP
1339: A0 01      486      MODUB
133B: 81 06      487      LDA (RADR),Y
133D: 0A         488      ASL
133E: 0A         489      ASL
133F: 0A         490      ASL
1340: 09 77      491      ORA #>77
1342: 85 1C      492      STA DEST2
1344: A0 21      493      LDY #>21
1346: 81 06      494      LDA (RADR),Y
1348: 09 EE      495      ORA #>EE
134A: 29 FF      496      AND #>FF
134C: 25 1C      497      AND DEST2
134E: 91 08      498      STA (DEST),Y
1350: E6 06      499      INC RADR
1352: E6 08      500      INC DEST
1354: D0 04      501      BNE MODUB2
1356: E6 07      502      INC RADR+1
1358: E6 09      503      INC DEST+1
135A: C6 1E      504      DEC TMP
135C: D0 D8      505      BNE MODD
135E: A9 20      506      LDA #>20
1360: 85 1E      507      STA TMP
1362: A5 08      508      LDA DEST
1364: C9 DF      509      CMP #<I$BUFFER+$FDF ;CHECK FOR END OF BUFFER
1366: D0 09      510      BNE NEVEN1
1368: A5 09      511      LDA DEST+1
136A: C9 3F      512      CMP #>I$BUFFER+$FDF

```

;PERFORM OPERATIONS AS DESCR
;ABOVE ON ODD ROWS :

;CHECK FOR END OF BUFFER

;I\$BUFFER+\$FDF

Listing 2nd continued

```

136C: D0 03 513      BNE NEVENJ
136E: 4C 74 13 514    JMP FILLIN
1371: 4C 04 13 515    NEVENJ JMP NEVEN      ;LOOP IF NOT DONE
516 1
517 #####
518 1 FILLIN COLORS HOLES IN THE IMAGE IN TBUFFER
519 1 EITHER BLACK OR WHITE, DEPENDING ON THE COLORS
520 1 OF THE SURROUNDING PIXELS.
521 1 THE ALGORITHM USED IS AS FOLLOWS:
522 1 FOR BYTES IN EVEN ROWS (STARTING WITH ROW 0) --
523 1 BITS 0,3,4,7 ARE 'HOLES'
524 1 BITS 2 AND 6 ARE UNCHANGED
525 1 BIT 5 COMES FROM BIT 2 OF THE BYTE 1 ROW PREV
526 1 BIT 1 COMES FROM BIT 6 OF THE BYTE 1 ROW
527 1 PREVIOUS LESS ONE BYTE
528 1
529 1 FOR BYTES IN ODD ROWS --
530 1 BITS 1,2,5,6 ARE 'HOLES'
531 1 BITS 0 AND 4 ARE UNCHANGED
532 1 BITS 3 AND 7 COME FROM BITS 0 AND 4 OF THE
533 1 BYTE 1 ROW PREVIOUS
534 #####
535 1
1374: A9 E0 536 FILLIN LDA #TBUFFER-#20 ;INIT BUFFER POINTER
1376: 85 08 537 STA DEST
1378: A9 2F 538 LDA #TBUFFER-#20
137A: 85 09 539 STA DEST+1
137C: 42 00 540 LDY #0
137E: A9 20 541 LDA ##20
1380: 85 1E 542 STA TMP
1382: A0 00 543 FILL1 LDY #0 ;FILLIN EACH ROW AS
1384: B1 08 544 LDA (DEST),Y ;DESCRIBED ABOVE
1386: 85 07 545 STA TEMP2
1388: A0 40 546 LDY #40
138A: B1 08 547 LDA (DEST),Y
138C: 85 1C 548 STA TEMP3
138E: 25 07 549 AND TEMP2
1390: 85 06 550 STA TEMP1
1392: A0 20 551 LDY #20
1394: B1 08 552 LDA (DEST),Y
1396: 4A 553 LSR
1397: 1D E0 13 554 ORA RMASK,X

139A: 05 06 555 ORA TEMP1
139C: 85 1D 556 STA TEMP4
139E: B1 08 557 LDA (DEST),Y
13A0: 0A 558 ASL
13A1: 1D E2 13 559 ORA LMASK,X
13A4: 05 06 560 ORA TEMP1
13A6: 85 18 561 STA TEMP5
13A8: A5 07 562 LDA TEMP2
13AA: 05 1C 563 ORA TEMP3
13AC: 1D DE 13 564 ORA CMASK,X
13AE: 31 08 565 AND (DEST),Y
13B1: 25 1D 566 AND TEMP4
13B3: 25 18 567 AND TEMP5
13B5: 91 08 568 STA (DEST),Y
13B7: E6 08 569 INC DEST
13B9: D0 08 570 BNE FILL2
13BB: E6 09 571 INC DEST+1
13BD: A9 40 572 LDA #TBUFFER+51000
13BF: C5 09 573 CMP DEST+1 ;CHECK FOR END OF BUFFER
13C1: F0 0F 574 BEQ FILL3
13C3: C6 1E 575 DEC TMP
13C5: D0 88 576 BNE FILL1
13C7: A9 20 577 LDA #20
13C9: 85 1E 578 STA TMP
13CB: 8A 579 TXA
13CD: A9 01 580 EOR #1
13CE: AA 581 TAX
13CF: 4C 82 13 582 JMP FILL1
13D2: A2 1F 583 FILL3 LDY #1F
13D4: D0 40 30 584 CLEAN LDA TBUFFER+440,X ;CLEAN UP THE FIRST ROW
13D7: 9D 00 30 585 STA TBUFFER,X
13DA: CA 586 DEX
13DB: 10 F7 587 BPL CLEAN
13DD: 60 588 RTS
13DE: 66 99 589 1
13E0: EE BB 590 CMASK DFB $66,$99
13E2: 77 DD 591 RMASK DFB $EE,$BB
13E4: 01 00 04 592 LMASK DFB $77,$DD
13E7: 0F 593 VAL1 DFB $01,$0D,$04,$0F
13E8: 90 50 C0 594 VAL2 DFB $90,$50,$C0,$80
13EB: 80 594 VAL2 DFB $90,$50,$C0,$80

```

Listing 2b continued:

13EC: 03 OF 02					
13EF1 0E	595 VAL3	DFB	603,60F,602,60E		
13F0: 80 70 60					
13F3: 60	596 VAL4	DFB	680,670,640,660		
	597 ↑				
	598 #####				
13F4: 00 20 00				1457: 27 00 28	
13F7: 24 00 28				145A: 00 2F 00	
13FA: 00 2C 00				145D: 33 00 37	
13FD: 30 00 34				1460: 00 3B 00	
1400: 00 3B 00				1463: 3F	605
1403: 3C				1464: 80 23 80	
1404: 80 20 80				1467: 27 80 28	
1407: 24 80 28				146A: 80 2F 80	
140A: 80 2C 80				146D: 33 80 37	
140D: 30 80 34				1470: 80 3B 80	
1410: 80 3B 80				1473: 3F	606
1413: 3C				1474: 2B 20 28	
1414: 00 21 00				1477: 24 28 28	
1417: 25 00 29				147A: 2B 2C 28	
141A: 00 20 00				147D: 30 2B 34	
141D: 31 00 35				1480: 2B 3B 28	
1420: 00 39 00				1483: 3C	607
1423: 3D				1484: 8B 20 AB	
1424: 80 21 80				1487: 24 AB 28	
1427: 25 80 29				148A: AB 2C AB	
142A: 80 2D 80				148D: 30 AB 34	
142D: 31 80 35				1490: AB 3B AB	
1430: 80 39 80				1493: 3C	608
1433: 3D				1494: 2B 21 28	
1434: 00 22 00				1497: 25 2B 29	
1437: 26 00 2A				149A: 2B 2D 28	
143A: 00 2E 00				149D: 31 2B 35	
143D: 32 00 36				14A0: 2B 39 28	
1440: 00 3A 00				14A3: 3D	609
1443: 3E				14A4: AB 21 AB	
1444: 80 22 80				14A7: 25 AB 29	
1447: 26 80 2A				14AA: AB 2D AB	
144A: 80 2E 80				14AD: 31 AB 35	
144D: 32 80 36				14B0: AB 39 AB	
1450: 80 3A 80				14B3: 3D	610
1453: 3E				14B4: 2B 22 28	
1454: 00 23 00				14B7: 26 2B 2A	
				14BA: 2B 2E 28	
				14BD: 32 2B 36	
				14C0: 2B 3A 2B	
				14C3: 3E	611
				14C4: AB 22 AB	
				14C7: 26 AB 2A	

HEX	00230027002B002F00330037003B003F	
HEX	80236027802B802F80338037803B803F	
HEX	28202824282B282C28302834283B283C	
HEX	A820A824A82BA82CA830A834A83BA83C	
HEX	282128252829282B2D2B312B352B392B3D	
HEX	AB21AB25AB29AB2DAB31AB35AB39AB3D	
HEX	28222826282A282E28322836283A283E	

Listing continued

Listing 2b continued:

```

14CA: A8 2E A8
14CB: 32 A8 36
14CD: A8 3A A8
14CE: 3E 612
14CF: 28 23 28
14D0: 27 28 28
14D1: 28 2F 28
14D2: 33 28 37
14D3: 28 3B 28
14D4: 3F 613
14E1: A8 23 A8
14E2: 27 A8 28
14E3: A8 2F A8
14E4: 33 A8 37
14E5: A8 3B A8
14F3: 3F 614

```

HEX A822A826A82AA82EA832A836A83A83E

HEX 2823282728282F28332837283B283F

HEX A823A827A82BA82FA833A837A83BA83F

--END ASSEMBLY--

ERRORS: 0

1268 BYTES

SYMBOL TABLE - ALPHABETICAL ORDER:

ACTACLR	= \$120A	BEEP	= \$C030	C0	= \$10A3	C1	= \$1056
C15	= \$105C	C3	= \$1067	CLEAN	= \$1304	CLR1	= \$111F
CLR2	= \$1227	CMASK	= \$130E	CONT	= \$10EF	CONT1	= \$10FD
COUNT	= \$1F	CTR	= \$19	D1	= \$1097	DATA	= \$C08F
DEST	= \$08	DEST2	= \$1C	DISP1	= \$1203	DISP1	= \$12F2
DLY1	= \$1270	DOKEY	= \$1075	DONE	= \$1203	ENHANCE	= \$12F2
FALL1	= \$1382	FILL2	= \$13C3	FILL3	= \$13D2	FILLM	= \$1374
FULLPIC	= \$1098	GET	= \$1030	GNODE	= \$C050	GRCLR	= \$121C
GREY	= \$1115	GRTABLE	= \$5000	GSTAT	= \$10CC	HGR	= \$C057
IBUFFER	= \$4090	IGNORE	= \$10EC	IMAGE	= \$E	INCREMENT	= \$0306
KEEPCNT	= \$F9	KEEPFLG	= \$FA	KEY	= \$0304	KEYCLR	= \$C010
KEYFLAG	= \$0305	KEYHIT	= \$C000	LMSK	= \$13E2	LOADTBL	= \$1153
LOOP	= \$1291	MIXED	= \$C053	MOV	= \$12B8	MOVE	= \$1283
MSEC	= \$1279	NONE	= \$1070	NEVEN	= \$1304	NEVENJ	= \$1371
MORP	= \$1339	NEXT	= \$11E9	NEXTPIC	= \$1132	NEXTROW	= \$11A2
MORPIC	= \$1009	NODUB	= \$1331	NODUB2	= \$135A	NOHANG	= \$10DE
		NSTART	= \$100C	NXT1	= \$11F5	NXT2	= \$1200

SYMBOL TABLE - NUMERICAL ORDER:

RADR	= \$06	TEMP1	= \$06	TEMP2	= \$07	DEST	= \$08
CTR	= \$19	YREG	= \$1B	TEMPS	= \$18	DEST2	= \$1C
TEMP3	= \$1C	TEMP4	= \$1D	TMP	= \$1E	COUNT	= \$1F
TABLE	= \$EB	IMAGE	= \$ED	KEEPCNT	= \$F9	KEEPFLG	= \$FA
SOAKTIME	= \$0300	SLOTADR	= \$0302	ROWSTART	= \$0303	KEY	= \$0304
KEYFLAG	= \$0305	INCREMENT	= \$0306	MORPIC	= \$1009	NSTART	= \$100C
NEWROW	= \$101E	GET	= \$1030	C0	= \$10A3	C1	= \$1056
C15	= \$105C	C3	= \$1067	NONE	= \$1070	DOKEY	= \$1075
D1	= \$1097	FULLPIC	= \$1098	TAKEPIC	= \$10A7	START	= \$10AA
GSTAT	= \$10CC	NOHANG	= \$10DE	IGNORE	= \$10EC	CONT	= \$10EF
CONT1	= \$10FD	GREY	= \$1115	CLR1	= \$111F	NEXTPIC	= \$1132
NXTBYTE	= \$1140	LOADTBL	= \$1153	ZERO	= \$1158	ZERO1	= \$1162
TBL1	= \$1160	NEXTROW	= \$11A2	THISROW	= \$11A6	THISBYTE	= \$11AE
NEXT	= \$11E9	NXT1	= \$11F5	NXT2	= \$1200	DONE	= \$1203
ACTACLR	= \$120A	GRCLR	= \$121C	CLR2	= \$1227	SETB	= \$1234
SEND CMD	= \$12A3	SEND1	= \$12A7	SOAK	= \$1255	SOAK1	= \$1260
SOAK2	= \$1278	MSEC	= \$1279	DLY1	= \$1270	MOVE	= \$1283
LOOP	= \$1291	DISPLAY	= \$12A3	NXTROW	= \$12AD	MOV	= \$12BB
RESHFT	= \$12C6	SHT	= \$12C9	DISP1	= \$12ED	ENHANCE	= \$12F2
NEVEN	= \$1304	NODUB	= \$1331	NOD	= \$1339	NODUB2	= \$135A
NEVENJ	= \$1371	FILLM	= \$1374	FILL1	= \$1382	FILL2	= \$13C3
FILL3	= \$13D2	CLEAN	= \$13D4	CWAK	= \$13DE	RMSK	= \$13E0
LMSK	= \$13E2	VAL1	= \$13E8	VAL2	= \$13EB	VAL3	= \$13EC
VAL4	= \$13F0	ROWPTR	= \$13F4	TBUFFER	= \$3000	IBUFFER	= \$4000
GRTABLE	= \$5000	KEYHIT	= \$C000	KEYCLR	= \$C010	BEEP	= \$C030
GNODE	= \$C050	THODE	= \$C051	MIXED	= \$C053	PAGE1	= \$C054
HGR	= \$C057	STATUS	= \$C0BE	DATA	= \$C0BF	PBLNK	= \$F948
PRHEX	= \$FDE3						

NXTBYTE	= \$1140	NXTROW	= \$12AD	PAGE1	= \$C054	PBLNK	= \$F948
PRHEX	= \$FDE3	RADR	= \$06	RESHFT	= \$12C6	RMSK	= \$13E0
ROWPTR	= \$13F4	ROWSTART	= \$0303	SEND1	= \$1247	SEND CMD	= \$1241
SETB	= \$1234	SHIFT	= \$12C9	SLOTADR	= \$0302	SOAK	= \$1255
SOAK1	= \$1260	SOAK2	= \$1278	SOAKTIME	= \$0300	START	= \$10AA
STATUS	= \$C0BE	TABLE	= \$EB	TAKEPIC	= \$10A7	TBL1	= \$1160
TBUFFER	= \$3000	TEMP1	= \$06	TEMP2	= \$07	TEMP3	= \$1C
TEMP4	= \$1D	TEMPS	= \$18	THISBYTE	= \$11AE	THISROW	= \$11A6
THODE	= \$C051	TMP	= \$1E	VAL1	= \$13E4	VAL2	= \$13EB
VAL3	= \$13EC	VAL4	= \$13F0	YREG	= \$1B	ZERO	= \$1158
ZERO1	= \$1162						



相信各位 APPLE 玩家們，在初學的時候，即在 BASIC 程式設計的技巧方面下了功夫學習 APPLE，日子久了漸漸的對電腦的操作有了概念之後，慢慢的，對於系統的操作程式感到興趣，進而想由高階走向低階的程式語言（ASSEMBLER），當然，由 BASIC 的程式設計進入組合語言的世界裡，您會因為不盡相同的設計觀念，及沒有 BASIC 的親切感，而感到茫然無從，啓先開始，會借助一些工具，例如舊的 APPLE MONITOR 裏面的 TRACE，以及 LISA 2.5 版背面的 TRACE 來達到你的學習目的，漸漸的，功力一天比一天的增強，進而設計出較大的程式，來幫你處理一些事物，不過爲了程式的體積，及閱讀的方便，即利用現有系統之副程式來使程式簡短些，一旦設計完成，想試 RUN 一次，結果不是出現一些奇怪的現象，就是當在裏面，跑不出來了，究竟較

大的程式，難免有幾隻小蟲在裡面作怪，而想要把它揪出來，必須依靠本身的經驗及耐心，始可達成，但是這對於入門不久，想一顯身手的新手而言，必然四處碰壁，無所適從，而失去了興趣與信心，而放棄了學習的機會，或許會利用前面所提到的 TRACE 程式，做追蹤的工作，在簡短程式效果還算不錯，如果遇到一個 LOOP，或是流程本身有問題，螢幕就好像捲不完的窗簾布似的，真累。而有些如 \$FD-ED 的，與本身的追蹤程式相衝突，只好停擺，不過 LISA 2.6 版即不受限制，而我在學習的階段，很喜歡觀看他人的設計技巧，於是將 FILE LOAD 進來，研究一番，啓始程式即做一些啓始設定，接著即有一些保護來防止他人閱讀程式，保護是可以解開，而有些的軟體，即將主程式及副程式或重要的 DATA 暫存與 MONITOR 所使用的區域相疊，所以只能大意的

了解，而不能細節技巧的研判。

在種種的障礙與阻撓下，要揣摩高等技巧的軟體，是愈來愈難了，勢必要借重硬體的支援方能達成。

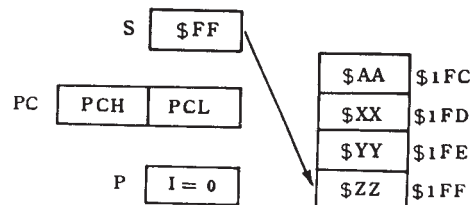
IRQ & NMI

在 6502 系統內，要利用介面信號的控制，來達成記憶的存取、修改，必須利用插斷 (INTERRUPT)，以及 DMA (Direct Memory Access) 的技巧來達成，後者即將 APPLE 之 6502 禁能以後，而由外介之位址巴士 (ADDRESS BUS)，直接由位址巴士控制，對 APPLE 之 64 K 的記憶體作直接存取，而不受 CPU 的控制，進而將任何一部分的資料、反組譯及更改向量，看來似乎是我們所要的功能，不過其缺點就是成本費非常高，它必須是一台語言學習機或是另一台 APPLE，而且其 BUS 線的連接還必須是三態緩衝，再加上一些閘控，始可達到目的，利用此一技術的系統並不多，在舊有系統裏較為常見，近年來大部分皆採用插斷的方法較經濟實用。

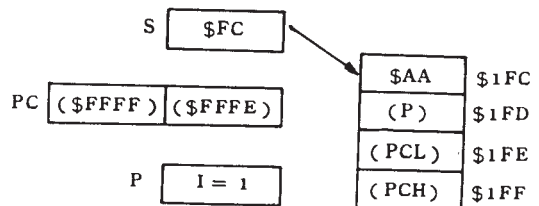
插斷是一外部產生之信號，該信號暫時延緩微處理機正在執行之程式，且令程式控制轉移至一專為服務 (Service) 發出插斷之設備而設計的副程式。週邊設備以插斷通知微處理機，其有資料要輸入給微處理機或必須由微處理機輸出資料。此一技巧去除了微處理機浪費寶貴時間，在沒有設備需要服務時，仍需不停地取樣 (Polling) 各週邊設備之狀態的必要。

而 6502 的插斷又分可罩蓋 IRQ (Interrupt ReQuest) 與不可罩蓋 NMI (Non-Maskable Interrupt)，6502 對 IRQ 之反應，若插斷禁能位元為零，而且某一外部設備使 IRQ 線動作 (將其拉至低電位)，則 6502 在執行完目前之指令後，

即會自動開始一個八週期之插斷系列。在此一系列裡，6502 將三位組之“回返”資訊推入堆疊器 (程式計數器之高、低位元組以及狀態暫存器之內含)。然後將專用的 IRQ 向量指示器 (\$FFFE 與 \$FFF) 取入程式計數器。同時 6502 亦將插斷禁能位元置定為 1，以暫時將爾後之插斷請求擋在門外。以下就是其對堆疊之影響。



(A) IRQ 前



(B) IRQ 後

6502 微處理機之另一插斷，不可罩蓋插斷 NMI，替在插斷請求被禁能期間無法等待之高優先設備或事件 (諸如停電) 提供了另一條門徑。與 IRQ 不同的是，NMI 所產生之插斷不可被漠視 (即被罩蓋或禁能)。一感應到 NMI 線為低電位時，6502 只要一完成正在執行之指令，即會立即開始處理不可罩蓋插斷，NMI 恒比可

罩蓋插斷 (IRQ) 具有較高優先。倘若有一插斷請求與一不可罩蓋插斷同時發生，則 6502 將毫無疑問地處理不可罩蓋插斷。除此之外 NMI 之動作與 IRQ 幾乎沒什麼差別了！在接受 NMI 插斷時，6502 同樣執行一個八週期之插斷系列，只不過，此向量指示器是來自 \$FFFA (低次位址位元組) 與 \$FFFB (含位址之高次位元組) 兩記憶位置，而非 \$FFFE 與 \$FFFF。事實上，上圖亦可以說明 6502 如何對 NMI 反應，除了 (B) 圖程式計數器之內含應改成 (\$FFFB) 與 (\$FFFA) 之外，其餘完全相同。

儘管以 NMI 的技術要比 DMA 要節省很多經費，及處理方式，不過以 DMA 的技術所達成的功能 (例如：MICE，以及發展系統) 即是 NMI 較難比擬其功效，如今我們所想要做的程式追蹤以 NMI 的方式，已是綽綽有餘的了。

第一代追蹤卡

當時玩 APPLE 的時候，購買了一片語言卡在那個時代中，大概只有當做整數卡來使用，就在這個時候，WILD CARD 強拷介面的出現，而對其所能產生的功能感到興趣，而久久未能取得此卡，而更進步的 REPLAY CARD 又相繼產生。而望著已半舊的語言卡，冷冷一笑，就把它當實驗而加以改裝。而加裝了一顆 74LS151 以及一個 DIP 開關以及數個二極體，二、三個主要控制開關，即把 BANK 的 RAM 分回了各介面的記憶體，以及強行讀取 ROM 或 RAM 的記憶與近來流行的 MASTER KEY 的 SWITCH 功能相類似。

初製成功，其功能相當令人滿意，起先因有跳訊產生，只好使用 IRQ 來做插斷工作，也足夠通吃大部分的軟體，不過好景不常，漸漸的愈來愈多軟體多了 SEI 的

指令，致使必須加裝一項單擊觸發的線路，直接使用 NMI 來插斷，設計時，555 以及 74123 皆在我的考慮範圍，不過終究還是採用友人的建議，利用 APPLE 本身的 KEY BOARD 的 STOB 信號來當單擊觸發的線路，效果也挺不錯的，只用到一條線上串個開關，各連接至 SLOT 上的 NMI 及 B10 的 7474D 型正反器的第 8 腳即可，要注意的就是用完後必須馬上切回，以免誤當鍵盤而再次觸發。

軟體操作大致與 REPLAY CARD 部分功能類似，所不同的是 REPLAY CARD 具有 COPY 的能力，而第一代追蹤卡僅具分析程式的功能。此追蹤卡以按鈕的方式來決定中斷，這對速度上 MHz 的 APPLE 來講，似乎有如在玩比賽果更不好猜的遊戲，以致在那歇腳，只得靠幸運之神的安排，而更不好玩的，就是有些軟體做完事情就跑到延遲的副程式去，尤其是 MONITOR 裏面延遲程式，一進入後即會破壞堆疊狀態，又偏偏命中率十分之高，以致分析程式，只有各憑本事了。

其實上述的難題只是初學者的一段小插曲罷了，真正的難題也就是一些你看不懂的指令，在反組譯以 ??? 來替代它，雖然它是在做一規律性的跳躍，其間也不知有否作一些運算，及改變暫存器，而這些難題必須經過實驗，始可通過層層的保護，另外因為我們的插斷服務常式是借用語言卡的部分空間，作暫存及回返中斷點的工作，不過目前的軟體，例如一些 GAME 及應用程式利用語言卡來暫存 DATA 的風氣是愈來愈盛，如不加以改進的話，很難在現階段來把它當一項工具了。

```

      <APPLE GROUP>   FOX SOFTWARE
      (C) COPY RIGHT 1984,3 BY W.L.W...
-----
      PAG 0  HIEI -LORES-TEXT  ALL-&TEXT
-----
MONIO                PC=FD24      SP=E3

RESTART              A=A0  X=00  Y=01

ERASE PAG BYTE              ONSTACK=9EB0 FD37

REFRESH PAG BYTE $3F2 RESET VECTOR=9DBF

FIND BYTE

CHANGE PAG                      NV*BDIZC
                                STATUS=00100011

HI-LO-TE DISPLAY

ALL-&TEXT INFO
-----

```

圖(一) SCREEN DUMP
第一代追蹤卡

第二代追蹤卡的設計

第一代的追蹤卡，也只不過是利用插斷而將程式暫停下來，而靠追蹤者對一些參數及 DATA 因不同時段的暫停來分析它們的變化，進而縮短程式分析的時間，然而憑追者本身對程式分析的能力，卡上的服務常式，足可應付大部分的軟體。

通常程式追蹤大概可分為中斷法及單部追蹤這兩種中斷法，亦即在程式流程的重要轉折點上安置個陷阱，每當 PC 路過於此，即被擋駕攔身，認為沒有問題，再將原值恢復，始可放行。

而單步追蹤的方式較為大眾所接受，即每執行一個指令，隨後將其暫存器之值，列印出來，程式的去留與變化皆一目了然，進而達到追蹤的目的。然而對於系統分析師而言，程式追蹤的過程，不僅是對暫存器的變化過程，有所了解。而且對於影響重要變數的指令群而言，亦是不可或缺的資料，甚至對某些資料自始至終的變化過程，亦是不可欠缺的，以致我們必將

這些功能皆合併在這塊卡上，才能真正達到程式追蹤的目的。

硬體的設計

執行單步追蹤時，必須每執行一個指令就得插斷一次，而 6502 的指令週期並非相同，必須利用 6502 的第七腳 (SY-NC) 在每一指令週期開始的時候產生一次，所以我們便可以利用這個特性使 6502 每執行一個指令，就有一次 NMI 的發生，然而每當插斷後，6502 參考插斷向量後，逕行跳至 \$3FB，而 \$3FB 亦即執行一個指令，致使又發生一次插斷，使得 PC 來回跑，程式也當在裏面，無法挽回，所以我們必須挪出一段記憶空間，當做安全區，未確保插斷後，不會繼續再做插斷的動作，而這段記憶空間，就選在 \$C800～\$CFFF 的位置，來儲存插斷後所執行的服務常式，以及一些記憶體來儲存插斷後的一些暫存器及 DATA，及必須修改 \$FFFA，\$FFFB 的向量指標，致於 NMI 的指標應永遠不變，不因語言卡的切換而有所更改，以致所使用的 2716，必須分成兩個部分，前面的 \$000～\$3FF 記憶體，定址在 \$C800～\$CBFF 的位址，而後面的記憶體 \$400～\$7FF，位址在 MONITOR \$FC00～\$FFFF，以求強行佔據向量指標，而這部分的記憶最好完全是 MONITOR 程式。

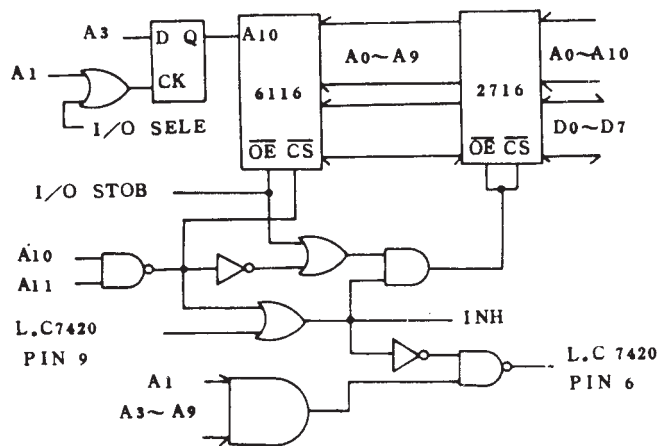
還有就是要做完全解碼，不管目前高位址是 RAM，是 ROM，皆必須是我們所加入的 ROM 致能，以期指標向量，永遠是我們的服務常式，而不因 RAM 卡的指標向量更改，而與我們的追蹤程式脫了節，不過本想設計成高位址解碼，有隱身的能力，以期瞞過一些 CHECK 常式，及一些刁鑽的保護程式，不過為了顧及卡上的空間安排，就把此功能作廢，如真需要如此專業

化的性能，可自行加個閘控。而關於 2 K 的 6116，則以一個 D 型正反器，切換它的二倍 4 頁的空間，即 \$CC00~\$CFFF 1 K 的空間，亦還有一個 BANK 的 RAM，以供服務常式作一些資料的暫存。

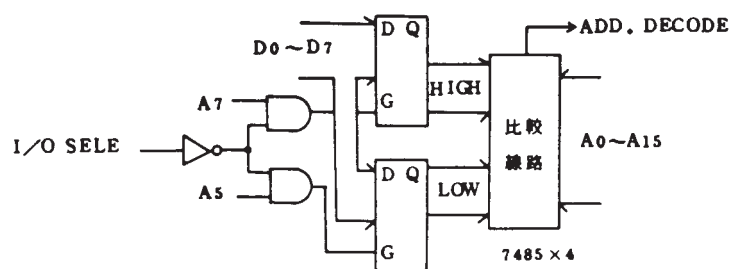
最後還要講的就是，可定址 64K 的任一位置的比較線路，此線路即以二個 74LS374 來閘鎖 DATA，而與位址巴士相比較，如果相同的話，即產生一插斷信號，就因為它的利用位址比較的方式，所以不管 6502 正在讀取操作及運算碼，或寫出與運算，皆有插斷的發生，也就是說如果我們鎖定了一個重要參數的位址，只要是讀取其值作為應用，或是改變其值及運算，皆能在此卡的監督之下無可遁形，堪稱此卡的主要功能。此功能甚至在無資料可循的情況下，找到一些變數的位址，就李小龍而言，如果我們將位址定在人數的畫

面位址上，程式讓它繼續執行，直到喪命的時候，必須更改畫面上的人數，此時插斷立刻產生，這時的 PC 值，必定是繪圖副程式，只要參考堆疊狀態往回倒追一下，便可獲知 \$902A，\$9029 為人數的暫存位址，而類似此種方法，亦可用在文字幕程式上，皆可達到異曲同工之妙。還有就是在研究程式讀寫的過程，將插斷點定在 RWTS 的入口處或出口處，再與單步追蹤合併，要了解其中的過程，就簡單多了。如何將此卡的優點發揮到極限，那只有看各位看官個人的功力如何了。

此卡要注意的就是其卡上有三個接點，分別外接至 6502 的 SYNC 及語言卡上的 74LS20，以期控制 RAM 的禁能，以及 ROM 6 的解碼。而卡上除了主開關以外，還有一個按鈕，以便隨時觀看程式。

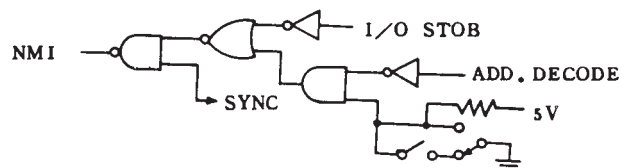


圖(二) 定址

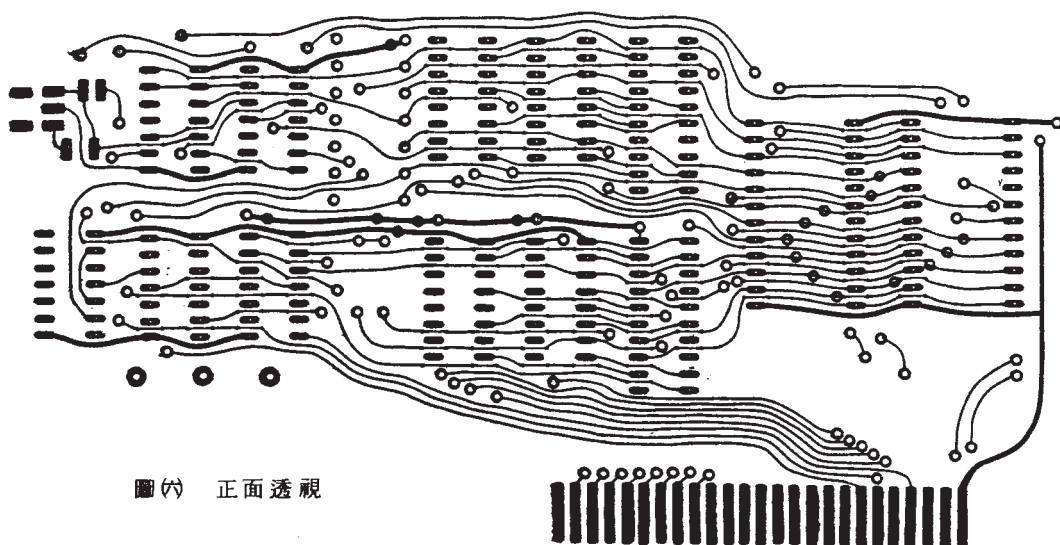


7437 × 2

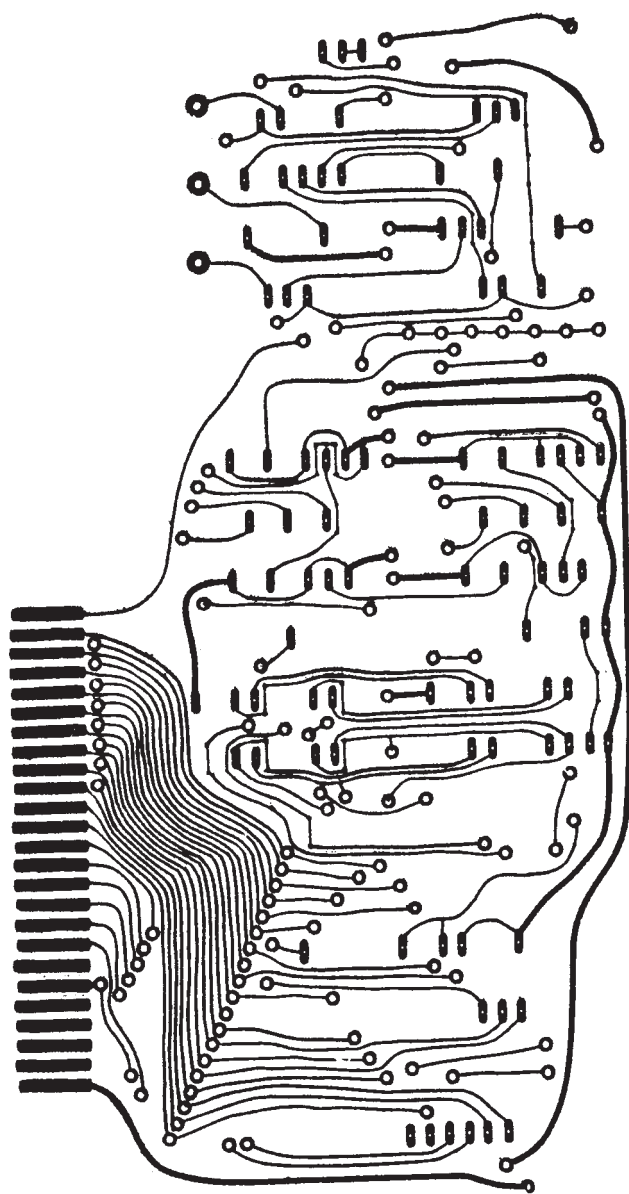
圖三 解碼插斷



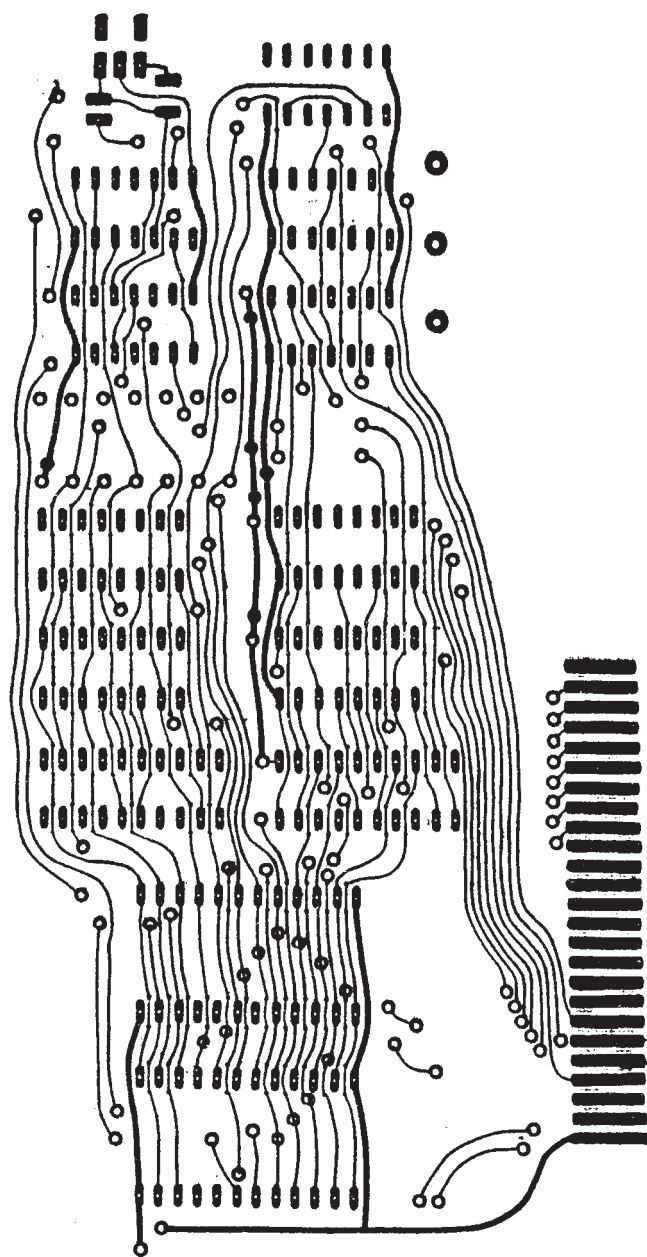
圖四 插斷線路



圖五 正面透視



圖(五) 正面線路



圖(七) 背面線路

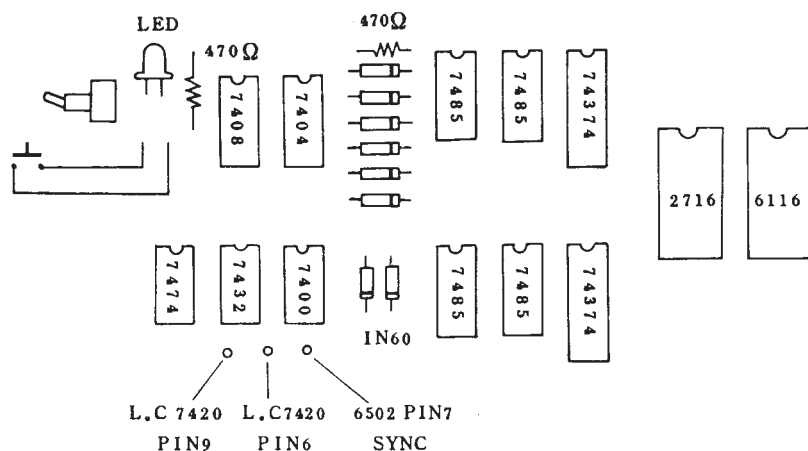


圖 (A) 零件圖

其中接至語言卡之 PIN6，必須仿效 REPLAY CARD 做法，將其 OPEN，然後接至追蹤卡上。

軟體部份

通常單步追蹤的方式，即每執行一次指令，就把 6502 的指令及暫存器列印一次，以致執行的速度即慢，且無效率，如果又遇上共同使用零頁位址及文字幕，那麼接下來的事情，也就面目全非了，以致 TRACE 程式，並不適合使用在大程式，況且觀看暫存器之值並非我們所必需的，或許我們應該利用這段時間，做我們想知道的事情及資料變化，以便節省時間，增快速度，以利分析。

燒在 ROM 的程式，就是幾個插斷模式所組成的 CHECK 常式，在每個 CHECK 常式裏，皆盡量的少用指令，以最簡單的方式完成我們所要做的事情，以增快追蹤程式的速度，然而這個用來 DEMO 的程式，單單文字幕的列印，就花了近三頁的記

憶空間，所剩的空間，只夠寫二個最簡單的 CHECK 常式，要想增加此卡的追蹤方式，只有讀者自行修改或是省掉部分的列印工作，以期有更多的功能，而最好的方法，就是依照自己的意願進行編寫。以下就是 DEMO 程式的使用方法

在平常追蹤的時候，螢幕並不顯示其間的追蹤過程，例如電玩的話，還是以畫面為顯示，只是速度上差異較大，一直到我們所需要的訊息資料出現後，即顯示 DEMO 程式的菜單，圖九中除了顯示暫存器及堆疊狀態，並有類似 COPY II PLUS 以空白棒選取各項命令，而各項命令中除了 SUB APPLE MONITOR 及 RET INT POINT 不須鍵入資料外，其餘命令皆需鍵入適當資料，例如軟體開關而言，其鍵入的資料以 BYTE 為單位，ANY KEY 為 PASS，回返鍵為結束每鍵入一個 BYTE，如尚需輸入，則以任何鍵為繼續輸入的依據，而如果是回返鍵，即為結束輸入，要注意的就是每輸入一項命令的資料後回

返插斷點，所執行的服務常式，則為該命令的 CHECK 常式，再其次就是各服務常式的功能的介紹。

插斷回返：即由主菜單返回插斷點，其過程，將文字幕及零頁等 DATA 恢復原值，並觸動一些語言卡及畫面的軟體開關及設定等工作。

位址解碼插斷：在前面對它的描寫應該相當明白，亦即以我們鍵入的 DATA 與位址線相比較來決定是否插斷。

程式單步計數：此功能較類似傳統的單步追蹤，所不同的，就是任意使用者止步或是任意值的抽樣追蹤。

尋找操作碼及運算碼：即將每次插斷之 PC 是否為我們所要尋找的操作碼及運算碼，此功能亦可去除尋找操作碼，直接比對運算碼，只要鍵入時操作碼為 \$00 即可。

監督程式：此功能即利用 APPLE 本身的 MONITOR，直接對追蹤的程式作反組譯，及修改記憶體內之 DATA。

軟體開關：在插斷回返的時候，必須恢復未插斷前的畫面及語言卡的讀寫狀態。

後記

也許一些讀者，會對於此卡的功能，到底可以利用到何種程度，一般說來，它只是在一大串程式碼當中，截取我們所需要的資料及程式流程，以其本身的處理能力，代替人力的分析以達到真正的自動追蹤，而筆者所試寫的程式，只有二、三種功能，並不適合大眾的需求，唯今的辦法，只有自行編寫才能有自己所預期的功能發揮無遺，不過四頁的位址，實在很難發揮讀者心目中的全能形象，而在硬體上保留了一個 D 型正反器，以便於 ROM 的切換，以增加編寫記憶。

到此為止，似乎破解保護的氣息非常濃厚，不過話又說回來，要分析程式之前，也必須做這些解碼的工作，而追蹤卡就省略這些繁複的破解程序直接觀察程式的執行情況，而唯一忌諱的就是磁碟讀寫 DATA 的時候，因嚴格的時序要求，無法觀察 NIBBLE 的讀寫狀況，以致無法偵測有問題的 NIBBLE，雖然可以以硬體斷點置於 RWTS 的出入口，以縮小偵測範圍，這也是美中不足的地方。

至於軟體編寫，必須要更新舊式追蹤的觀念，以更好既快的追蹤方式，來達到我們所需的目的是功能，目前的 APPLE 軟體，沒有一種不能被我們所追蹤的，如果你把它當破解磁片的工具，那未免太大材小用了，希望的就是將它當做一項除錯及分析程式的利器，以利巨大程式的偵錯不易，在此更期盼在未來的日子裏，第三代的追蹤卡能到達完全自動的境界，以便利國內軟體蓬勃發展。

```
AUTO TRACE CARD          NV*BDIZC
C                          10010001
S=20  A=50  X=91  Y=20  PC=2C92

STACK=FFFF FFFF FFFF FFFF FFFF FFFF FFFF
-----
$34C623
-----
RETURN INT POINT
ADD DECODE INT
STEP
FIND OP & OPERAND
MONITOR
SOFT SWITCH
```

圖(九) SCREEN DUMP
第二代追蹤卡

程式列印

```

1 *****
2 *
3 *      INTERRUPT      ROUTINE      *
4 *
5 *      FOX  SOFTWARE    *
6 *
7 *****
8 *
9 *      REGISTER
10 P      EQU $CC00
11 S      EQU P+1
12 A      EQU P+2
13 X      EQU P+3
14 Y      EQU P+4
15 PCH     EQU P+5
16 PCL     EQU P+6
17 COM     EQU P+7
18 VECL    EQU P+8
19 VECH    EQU P+9
20 CHECK   EQU P+10
21 LOOP    EQU P+11
22 AD1:
23         PHA
24         LSR
25         AND #$03
26         ORA #$04
27         STA $01
28         PLA
29         AND #$18
30         BCC AD2
31         ADC #$7F
32 AD2:
33         STA $00
34         ASL
35         ASL
36         ORA $00
37         STA $00
38         RTS
39 AD3:
40         STX $03
41         BNE AD5
42         PHA
43         LSR
44         LSR
45         LSR
46         LSR
47         JSR AD4
48         PLA
49 AD4:
50         AND #$0F
51         EOR #$80
52         CMP #$BA
53         BCC AD5
54         ADC #$06
55 AD5:

```

```

56         PHA
57         LDA $02
58         JSR AD1
59         LDY $03
60         PLA
61         STA ($00),Y
62         INC $03
63         RTS
64 AD6:
65         STY $02
66         STX $03
67 AD7:
68         PLA
69         STA $05
70         PLA
71         STA $06
72         BNE AD9
73 AD8:
74         LDY #$00
75         LDA ($05),Y
76         BEQ AD10
77         JSR AD5
78 AD9:
79         INC $05
80         BNE AD8
81         INC $06
82         BNE AD8
83 AD10:
84         LDA $06
85         PHA
86         LDA $05
87         PHA
88         RTS
89 AD11:
90         STY $02
91 AD12:
92         JSR AD13
93         INC $02
94         LDA $02
95         CMP #$17
96         BNE AD12
97         RTS
98 AD13:
99         LDA #$00
100        STA $03
101        LDA #$A0
102        JSR AD5
103        LDA $03
104        CMP #$28
105        BNE AD13+4
106        RTS
107 AD14:
108        LDY #$00
109        STY $04
110        JSR AD11
111        LDY #$00
112        LDX #$05

```

113	JSR AD6	170	JSR AD3+4
114	INV "AUTO TRACE CARD"	171	RTS
115	HEX 00	172 AD21:	
116	LDX ##20	173	LDY ##04
117	JSR AD6+2	174	LDX ##00
118	ASC "NV*BDIZC"	175	JSR AD6
119	HEX 00	176	ASC "STACK="
120	INC \$02	177	HEX 00
121	LDX ##20	178	LDX 5
122	STX \$03	179	INX
123	LDA P	180 AD22:	
124	PHA	181	LDA \$100,X
125 AD15:		182	PHA
126	PLA	183	INX
127	ASL	184	LDA \$100,X
128	PHA	185	INX
129	BCS AD16	186	STX \$01
130	LDA ##B0	187	JSR AD3+4
131	BNE AD17	188	PLA
132 AD16:		189	JSR AD3+4
133	LDA ##B1	190	INC \$03
134 AD17:		191	LDX \$07
135	JSR AD5	192	CPX ##FE
136	LDA \$03	193	BCS AD23
137	CMF ##28	194	LDA \$03
138	BNE AD15	195	CMF ##20
139	PLA	196	BCS AD23
140	INC \$02	197	BCC AD22
141	LDX ##01	198 AD23:	
142	STX \$03	199	LDY ##00
143	LDA ##D3	200	LDA ##AD
144	JSR AD16	201 AD24:	
145	LDA ##C1	202	STA \$680,Y
146	JSR AD18	203	STA \$780,Y
147	LDA ##D8	204	INX
148	JSR AD18	205	CPY ##28
149	LDA ##D9	206	BNE AD24
150	JSR AD18	207	HEX A9
151	LDX ##20	208	BYT AD26-1
152	JSR AD7-2	209	STA \$05
153	ASC "PC"	210	LDA ##07
154	HEX 00	211	STA \$02
155	JSR AD19	212 AD25:	
156	JSR AD20	213	HEX A9
157	BNE AD21	214	HBY AD25
158 AD18:		215	PHA
159	INC \$03	216	HEX A9
160	INC \$03	217	BYT AD25-1
161	INC \$03	218	PHA
162	JSR AD5	219	INC \$02
163 AD19:		220	INC \$05
164	LDA ##BD	221	LDX ##0B
165	JSR AD5	222	JMP (\$05)
166 AD20:		223 AD26:	
167	INC \$04	224	JSR AD7-2
168	LDY \$04	225	ASC "RETURN INT POINT"
169	LDA P,Y		

226	HEX 0060	283	AD35:
227	JSR AD7-2	284	LDA COM
228	ASC "ADD DECODE INT"	285	ASL
229	HEX 0060	286	ADC #\$08
230	JSR AD7-2	287	JSR AD1
231	ASC "STEP"	288	LDY #\$00
232	HEX 0060	289	AD36:
233	JSR AD7-2	290	LDA (\$00),Y
234	ASC "FIND OP & OPERAND"	291	AND #\$BF
235	HEX 0060	292	EOR #\$80
236	JSR AD7-2	293	STA (\$00),Y
237	ASC "MONITOR"	294	INY
238	HEX 0060	295	CPY #\$28
239	JSR AD7-2	296	BNE AD36
240	ASC "SOFT SWITCH"	297	RTS
241	HEX 0060	298	AD37:
242	PLA	299	HEX 008281830008
243	PLA	300	AD38:
244	AD27:	301	ADR COM1,COM2,COM3,COM4
245	LDA COM	302	AD39:
246	CMP #\$06	303	BIT \$C010
247	BCC AD28	304	LDA \$C000
248	LDA #\$00	305	BPL AD39+3
249	STA COM	306	EOR #\$E0
250	AD28:	307	CMP #\$0A
251	JSR AD35	308	BCC AD40
252	BIT \$C010	309	ADC #\$88
253	AD29:	310	CMP #\$FA
254	LDA \$C000	311	BCC AD40
255	BPL AD29	312	BCC AD39
256	LDX COM	313	AD40:
257	CMP #\$95	314	RTS
258	BEQ AD30	315	AD41:
259	CMP #\$88	316	LDX COM
260	BEQ AD31	317	LDA AD37,X
261	CMP #\$8D	318	PHA
262	BNE AD29-3	319	BPL AD42
263	BEQ AD41	320	LDA COM
264	AD30:	321	ASL
265	INX	322	TAY
266	BNE AD32	323	LDA AD38,Y
267	AD31:	324	STA VECL
268	DEX	325	INY
269	AD32:	326	LDA AD38,Y
270	BPL AD33	327	STA VECH
271	LDX #\$05	328	EOR #\$D5
272	AD33:	329	STA CHECK
273	CPX #\$06	330	AD42:
274	BNE AD34	331	PLA
275	LDX #\$00	332	BNE AD43
276	AD34:	333	LDA COM
277	STX \$04	334	BNE COM5
278	JSR AD35	335	JMP COM1
279	LDA \$04	336	COM5:
280	STA COM	337	JSR \$FB2F
281	JSR AD35+3	338	JSR \$FE93
282	BEQ AD29-3	339	JSR \$FE89

340	JMP \$FF69	397	RTS
341 AD43:		398 AD50:	
342	AND #\$7F	399	LDA COM
343	STA \$04	400	ASL
344	LDY #\$06	401	ASL
345	LDX #\$0F	402	ADC #\$10
346	JSR AD6	403	TAX
347	ASC "\$"	404	RTS
348	INV " "	405 AD51:	
349	HEX 00	406	LDA #\$06
350	DEC \$03	407	STA \$02
351 AD44:		408	JSR AD13
352	JSR AD50	409	JMP AD29-3
353	LDA #\$00	410 AD52:	
354	STA P,X	411	STX X
355 AD45:		412	STY Y
356	JSR AD47	413	PLA
357	PHA	414	STA P
358	JSR AD50	415	PLA
359	INC P,X	416	STA PCL
360	JSR AD49	417	PLA
361	PLA	418	STA PCH
362	STA P,X	419	TSX
363	BIT \$C010	420	STX S
364 AD46:		421	BIT \$C200
365	LDA \$C000	422	LDX #\$00
366	BPL AD46	423 AD53:	
367	CMP #\$8D	424	LDA \$00,X
368	BEQ AD51	425	STA \$CE00,X
369	DEC \$04	426	LDA \$200,X
370	BNE AD45	427	STA \$CF00,X
371	BEQ AD51	428	INX
372 AD47:		429	BNE AD53
373	JSR AD48	430	JSR AD54
374	ASL	431	BIT \$C200
375	ASL	432	JMP AD14
376	ASL	433 AD54:	
377	ASL	434	SEC
378	STA \$10	435	BIT \$C208
379	JSR AD48	436	LDX #\$04
380	AND #\$0F	437	LDY #\$00
381	ORA \$10	438	STY \$00
382	RTS	439	STX \$01
383 AD48:		440	STY \$02
384	JSR AD39	441	LDA #\$CC
385	PHA	442	STA \$03
386	JSR AD4	443	BCS AD56
387	LDA #\$20	444 AD55:	
388	JSR AD5	445	LDA (\$02),Y
389	DEC \$03	446	STA (\$00),Y
390	PLA	447	BCC AD57
391	RTS	448 AD56:	
392 AD49:		449	LDA (\$00),Y
393	JSR AD50	450	STA (\$02),Y
394	CLC	451 AD57:	
395	ADC P,X	452	INY
396	TAX	453	BNE AD55-2

454	INC \$01	511 COM4:	
455	INC \$03	512	STX X
456	DEX	513	STY Y
457	BNE AD55-2	514	TSX
458	RTS	515	LDA \$102,X
459 COM1:		516	STA \$FE
460	JSR AD54+1	517	LDA \$103,X
461 AD58:		518	STA \$FF
462	BIT \$C200	519	LDY P+\$1C
463	LDX P+\$24	520 AD63:	
464 AD59:		521	LDA P+\$1C,Y
465	LDA P+\$24,X	522	BNE AD64
466	TAY	523	CPY #\$01
467	STA \$C000,Y	524	BEQ AD65-3
468	DEX	525 AD64:	
469	BNE AD59	526	CMP (\$FE),Y
470	LDA P+\$19	527	BNE AD65
471	STA LOOP	528	DEY
472	SEC	529	BNE AD63
473	LDA P+\$15	530	JMP AD52+6
474	BMI AD60	531 AD65:	
475	CLC	532	LDA A
476 AD60:		533	LDX X
477	ROL	534	LDY Y
478	ROL	535	RTI
479	ROL	536 COM2:	
480	ROL	537	STA A
481	STA \$C242	538	LDA VECH
482	LDA P+\$16	539	EOR #\$D5
483	STA \$C222	540	CMP CHECK
484	LDX #\$00	541	BNE AD66
485 AD61:		542	JMP (VECL)
486	LDA \$CE00,X	543 AD66:	
487	STA \$00,X	544	JMP AD52
488	LDA \$CF00,X	545	END
489	STA \$200,X		
490	INX		
491	BNE AD61		
492	LDX S		
493	TXS		
494	STA \$C010		
495	LDX X		
496	LDY Y		
497	LDA PCH		
498	PHA		
499	LDA PCL		
500	PHA		
501	LDA P		
502	PHA		
503	RTI		
504 COM3:			
505	DEC LOOP		
506	BNE AD62		
507	JMP AD52		
508 AD62:			
509	LDA A		
510	RTI		

程式列印

```

0800- 48 4A 29 03 09 04 85 01
0808- 68 29 18 90 02 69 7F 85
0810- 00 0A 0A 05 00 85 00 60
0818- 86 03 D0 13 48 4A 4A 4A
0820- 4A 20 25 C8 68 29 0F 49
0828- B0 C9 BA 90 02 69 06 48
0830- A5 02 20 00 C8 A4 03 68
0838- 91 00 E6 03 60 84 02 86
0840- 03 68 85 05 68 85 06 D0
0848- 09 A0 00 B1 05 F0 0B 20
0850- 2F C8 E6 05 D0 F3 E6 06
0858- D0 EF A5 06 48 A5 05 48
0860- 60 84 02 20 6F C8 E6 02
0868- A5 02 C9 17 D0 F5 60 A9
0870- 00 85 03 A9 A0 20 2F C8
0878- A5 03 C9 28 D0 F5 60 A0
0880- 00 84 04 20 61 C8 A0 00

```

0888- A2 05 20 3D C8 01 15 14
 0890- 0F 20 14 12 01 03 05 20
 0898- 03 01 12 04 00 A2 20 20
 08A0- 3F C8 CE D6 AA C2 C4 C9
 08A8- DA C3 00 E6 02 A2 20 86
 08B0- 03 AD 00 90 48 68 0A 48
 08B8- B0 04 A9 B0 D0 02 A9 B1
 08C0- 20 2F C8 A5 03 C9 28 D0
 08C8- EC 68 E6 02 A2 01 86 03
 08D0- A9 D3 20 F4 C8 A9 C1 20
 08D8- F4 C8 A9 D8 20 F4 C8 A9
 08E0- D9 20 F4 C8 A2 20 20 3F
 08E8- C8 D0 C3 00 20 FD C8 20
 08F0- 02 C9 D0 19 E6 03 E6 03
 08F8- E6 03 20 2F C8 A9 BD 20
 0900- 2F C8 E6 04 A4 04 B9 00
 0908- 90 20 1C C8 60 A0 04 A2
 0910- 00 20 3D C8 D3 D4 C1 C3
 0918- CB BD 00 AE 01 90 E8 BD
 0920- 00 01 48 E8 BD 00 01 E8
 0928- 86 07 20 1C C8 68 20 1C
 0930- C8 E6 03 A6 07 E0 FE B0
 0938- 08 A5 03 C9 28 B0 02 90
 0940- DE A0 00 A9 AD 99 80 06
 0948- 99 80 07 C8 C0 28 D0 F5
 0950- A9 66 85 05 A9 07 85 02
 0958- A9 C9 48 A9 57 48 E6 02
 0960- E6 05 A2 08 6C 05 00 20
 0968- 3F C8 D2 C5 D4 D5 D2 CE
 0970- A0 C9 CE D4 A0 D0 CF C9
 0978- CE D4 00 60 20 3F C8 C1
 0980- C4 C4 A0 C4 C5 C3 CF C4
 0988- C5 A0 C9 CE D4 00 60 20
 0990- 3F C8 D3 D4 C5 D0 00 80
 0998- 20 3F C8 C6 C9 CE C4 A0
 09A0- CF D0 A0 A6 A0 CF D0 C5
 09A8- D2 C1 CE C4 00 60 20 3F
 09B0- C8 CD CF CE C9 D4 CF D2
 09B8- 00 60 20 3F C8 D3 CF C6
 09C0- D4 A0 D3 D7 C9 D4 C3 C8
 09C8- 00 60 68 68 AD 07 90 C9
 09D0- 06 90 05 A9 00 8D 07 90
 09D8- 20 11 CA 2C 10 C0 AD 00
 09E0- C0 10 FB AE 07 90 C9 95
 09E8- F0 0A C9 88 F0 09 C9 8D
 09F0- D0 E9 F0 5B E8 D0 01 CA
 09F8- 10 02 A2 05 E0 06 D0 02
 0A00- A2 00 86 04 20 11 CA A5
 0A08- 04 8D 07 90 20 14 CA F0
 0A10- CA AD 07 90 0A 69 08 20
 0A18- 00 C8 A0 00 B1 00 29 BF
 0A20- 49 80 91 00 C8 C0 28 D0
 0A28- F3 60 00 82 81 83 00 08
 0A30- 59 CB ED CB B0 CB BC CB
 0A38- 2C 10 C0 AD 00 C0 10 FB
 0A40- 49 B0 C9 0A 90 08 69 88

0A48- C9 FA B0 02 90 EA 60 AE
 0A50- 07 90 BD 2A CA 48 10 17
 0A58- AD 07 90 0A A8 B9 30 CA
 0A60- 8D 08 90 C8 B9 30 CA 8D
 0A68- 09 90 49 D5 8D 0A 90 68
 0A70- D0 14 AD 07 90 D0 03 4C
 0A78- 59 CB 20 2F FB 20 93 FE
 0A80- 20 89 FE 4C 69 FF 29 7F
 0A88- 85 04 A0 06 A2 0F 20 3D
 0A90- C8 A4 20 00 C6 03 20 EB
 0A98- CA A9 00 9D 00 90 20 C1
 0AA0- CA 48 20 EB CA FE 00 90
 0AA8- 20 E2 CA 68 9D 00 90 2C
 0AB0- 10 C0 AD 00 C0 10 FB C9
 0AB8- 8D F0 39 C6 04 D0 DF F0
 0AC0- 33 20 D2 CA 0A 0A 0A 0A
 0AC8- 85 10 20 D2 CA 29 0F 05
 0AD0- 10 60 20 38 CA 48 20 25
 0AD8- C8 A9 20 20 2F C8 C6 03
 0AE0- 68 60 20 EB CA 18 7D 00
 0AE8- 90 AA 60 AD 07 90 0A 0A
 0AF0- 69 10 AA 60 A9 06 85 02
 0AF8- 20 6F C8 4C DB C9 8E 03
 0B00- 90 8C 04 90 68 8D 00 90
 0B08- 68 8D 06 90 68 8D 05 90
 0B10- BA 8E 01 90 2C 00 C2 A2
 0B18- 00 B5 00 9D 00 CE BD 00
 0B20- 02 9D 00 CF E8 D0 F2 20
 0B28- 30 CB 2C 00 C2 4C 7F C8
 0B30- 38 2C 08 C2 A2 04 A0 00
 0B38- 84 00 86 01 84 02 A9 CC
 0B40- 85 03 B0 06 B1 02 91 00
 0B48- 90 04 B1 00 91 02 C8 D0
 0B50- F1 E6 01 E6 03 CA D0 EA
 0B58- 60 20 31 CB 2C 00 C2 AE
 0B60- 24 90 BD 24 90 A8 99 00
 0B68- C0 CA D0 F6 AD 19 90 8D
 0B70- 0B 90 38 AD 15 90 30 01
 0B78- 18 2A 2A 2A 2A 8D 42 C2
 0B80- AD 16 90 8D 22 C2 A2 00
 0B88- BD 00 CE 95 00 BD 00 CF
 0B90- 9D 00 02 E8 D0 F2 AE 01
 0B98- 90 9A 8D 10 C0 AE 03 90
 0BA0- AC 04 90 AD 05 90 48 AD
 0BA8- 06 90 48 AD 00 90 48 40
 0BB0- CE 0B 90 D0 03 4C FE CA
 0BB8- AD 02 90 40 8E 03 90 8C
 0BC0- 04 90 BA BD 02 01 85 FE
 0BC8- BD 03 01 85 FF AC 1C 90
 0BD0- B9 1C 90 D0 04 C0 01 F0
 0BD8- 07 D1 FE D0 06 88 D0 F0
 0BE0- 4C 04 CB AD 02 90 AE 03
 0BE8- 90 AC 04 90 40 8D 02 90
 0BF0- AD 09 90 49 D5 CD 0A 90
 0BF8- D0 03 6C 08 90 4C FE CA

爲你的APPLE增加一副眼睛——

監督卡

前言

1. 以 MASTER SWITCH + 破解過磁片的朋友，相信都曾經遇到過這個問題，那就是——破解是破解得出來，但是，並不知道是從何處開始執行，亦不知道剛載取下來的位址，而無法倒推到原始之進入點，所得到的，只是一堆無用的數據，無法讓你享受到破解程式的樂趣。

2. 曾經以組合語言來寫過程式的人，也曾經歷過這種現象，那就是——程式寫好了，執行之後，會無緣無故的當在裏面，而程式當在那個地方亦無法得知。可能又要花許多時間及精力來執行偵錯的工作，而此偵錯之工作是既費時，又費力的，以至於許多有志於學習組合語言的朋友，漸漸的失去了興趣。

3. 使用過電腦的朋友，可能在操作時，忽然間覺得程式似乎永無止境的一直在執行，而事實上你的電腦，可能因為某些因素而導致當機，而你尚未察覺，而以為是程式有了問題，於是導致你偵錯方向的錯誤。

監督卡之目的

1. 讓你可以馬上發現程式之可能進入點，或者是可以得知剛剛以 MASTER KEY + 所載取下來 CPU 控制權前，程式所正在執行之位址，讓你更容易推導到程式之真正進入點，讓你更容易的去解析別人的程式。

2. 可以偵察出你因程式錯誤，而造成無限迴圈之所在。一來可以偵錯，二來更能提高學習組合語言之操作及興趣，雖然

市面上有一些做此類工作之軟體，但是它必須佔去很多的記憶空間，使得你所能使用之空間非常之少，而本卡乃是以硬體之方法來解析，故無此困擾。

3. 可以偵察出你的機器是因為當機，或者程式之設計不良，而造成之無動作之現象。

原理

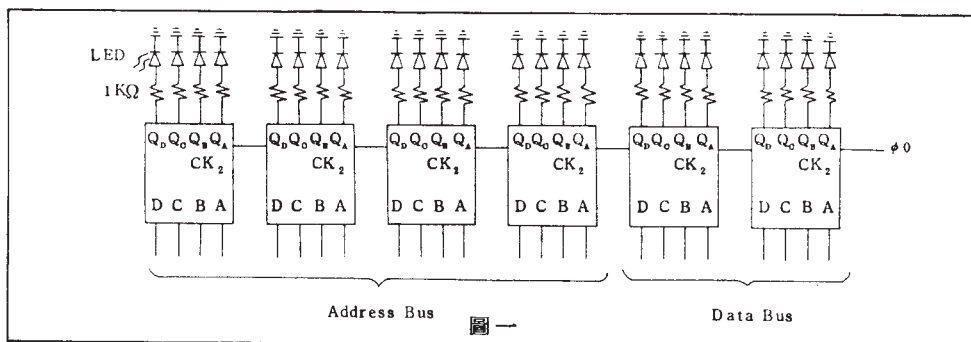
1. 將 APPLE 之 DATA BUS 以及 Address Bus 之資料攔截下來。

2. 將所攔截下來之數據予以解碼，而

以七段顯示器或 LED(須自行解碼)，而讀取其值，以了解其工作情形。

線路及其工作情形

1. 簡單型：如圖①之結構，以 6 顆 7495B 將 DATA BUS 及 Address Bus 之資料在 CLOCK 0 之降緣時 (CLOCK 0 之降緣時為 DATA BUS 及 Address Bus 資料處於最穩定狀態之時)，予以攔截其資料，送到上部之發光二極體 (LED)，使得相對之 LED 發亮，顯示出各 BUS 綫內之資料型態，然後依據表①之轉換表，予



圖一

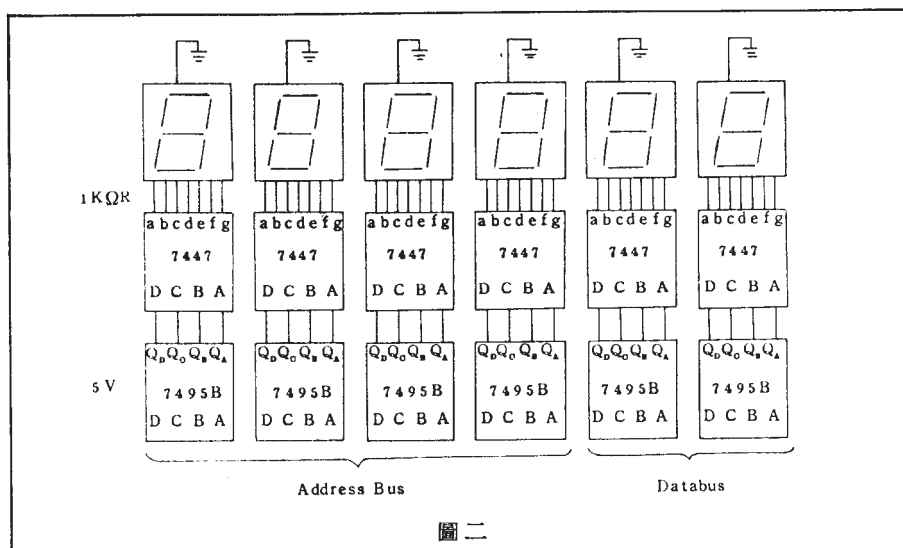
數字	燈 編 號				0 = 燈不亮 1 = 燈亮
	4	3	2	1	
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
A	1	0	1	0	
B	1	0	1	1	
C	1	1	0	0	
D	1	1	0	1	
E	1	1	1	0	
F	1	1	1	1	

表一

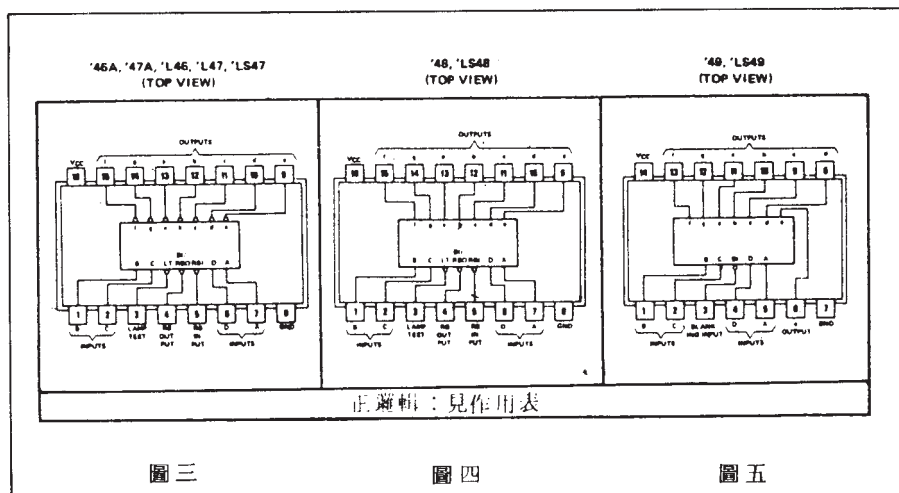
以解碼，即可得知其內涵值。

2. 簡單改良型：如圖②所示。仍然以 6 顆 7495B 將 DATA BUS 及 Address Bus 之資料予以攔截，再將此資料送到了 7447, 7448, 7449 (接腳圖如圖③④⑤所示)，予以解碼，再送到七段顯示器以顯示其值，再根據表②解碼成真正之十六進位值，而讀取其內涵值。

3. 直接型：如圖⑥所示，此綫路乃利用 APPLE 產生字型之原理，將十六進位 Q ~ F 之字型，直接燒錄於 EPROM 內，(七段顯示器之字型如表③，其各字型之資料列於表④，即 EPROM 所須儲存之資料)，其仍以 6 顆 7495B 將資料予以攔截，經過 EPROM 之解碼，而直接輸出字



圖二

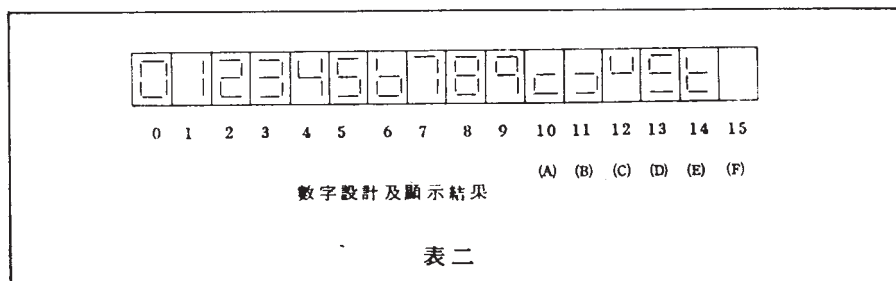


正邏輯：見作用表

圖三

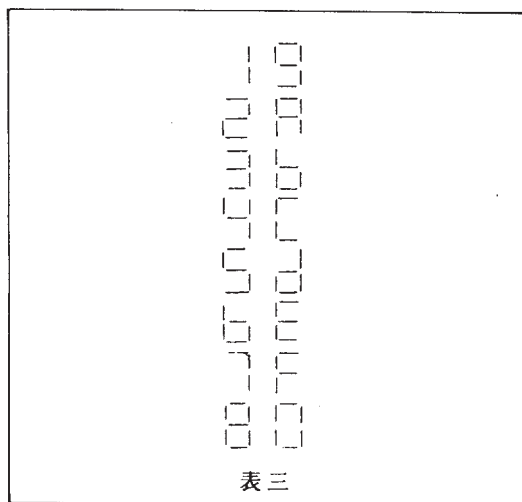
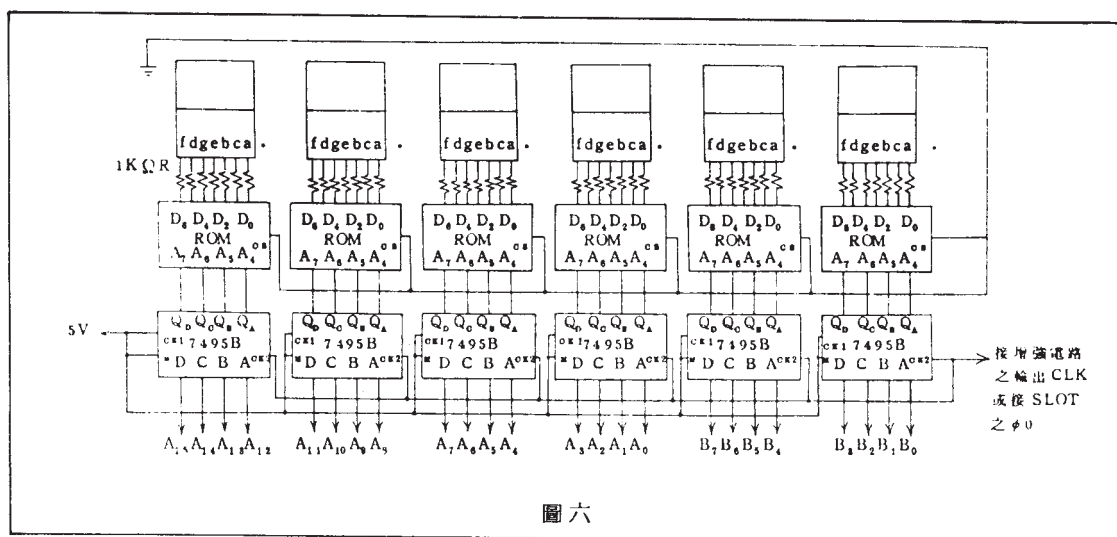
圖四

圖五



數字設計及顯示結果

表二



00:06〔0〕	80:7F〔8〕
10:06〔1〕	90:77〔9〕
20:3D〔2〕	A0:5F〔A〕
30:37〔3〕	B0:7A〔B〕
40:56〔4〕	C0:69〔C〕
50:73〔5〕	D0:3E〔D〕
60:7B〔6〕	E0:79〔E〕
70:07〔7〕	F0:59〔F〕

在ROM內各字型之位址及內容

表 四

型資料到七段顯示器，而讓你直接解讀。

各位讀者，可自行以萬用板直接連接，或製作印刷電路板（如圖⑦ 2708，圖⑧ 2716 之接腳須注意之）

注意事項：EPROM 字型資料，並非從位址之開頭開始，請注意表④所列之資料，並須注意你所用之 EPROM 之編號，而選取正確之印刷電路板及電路製作。

增強電路

目的：

使寫組合語言之朋友，能夠以單步執行指令之方式，一步一步地執行程式，而更能夠了解到程式之執行情形。

原理：

1. 利用CPU之RDY接脚 (如圖⑨所示)

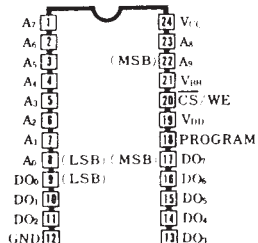
）使其具有類似Z80之WAIT 綫之功能。

2 CPU之SYNC（並非為SLOT 上之SYNC），在讀取指令週期時將為Hi 電

位，其它週期時將為Lo 電位，因此利用此腳及RDY接腳，再配合上一些附加電路，如此便可達到單步執行指令之功能。

8K nMOS UV-EPROM(1,024×8) 2708

◆接腳圖



◆電源

Vcc : + 5V	Pin24
Vss (GND)	Pin12
Vss : - 5V	Pin21
Vss : + 12V	Pin19
PROGRAM (GND)	Pin18

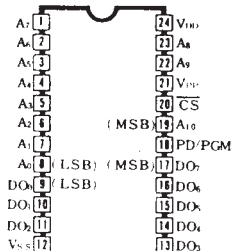
◆功能表

CS	功 能
H	非選擇
L	Read

圖七

16K nMOS UV-EPROM(2,048×8) i2716 TMS2716

◆接腳圖



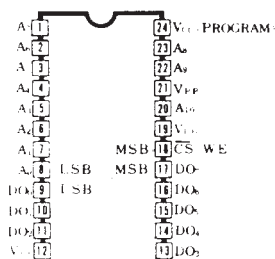
◆電源

Vss : + 5V	Pin24
Vss : + 5V	Pin21
Vss (GND)	Pin12

◆功能表

CS	功 能
H	非選擇
L	Read

◆接腳圖



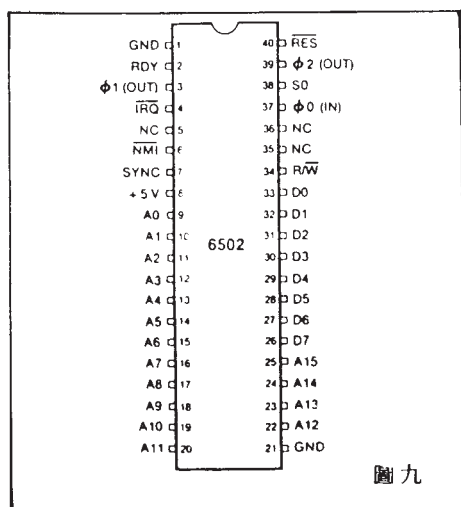
◆電源

Vcc : + 12V	Pin19
Vcc : + 5V	Pin24
Vss : + 5V	Pin21
Vss (GND)	Pin12

◆功能表

CS	功 能
H	非選擇
L	Read

圖八



圖九

3. 利用一比較電路，當你只想監督某一頁（PAGE）之程式執行，只須設定SW之ON 以及OFF，當程式執行到此頁時，便會自動轉換成單步執行指令之狀態，使你更能控制程式之流程

線路及其工作情形

其線路接綫圖如圖⑩所示，乃利用單穩態振盪電路、正反器，及NAND閘所組成，而使得CPU之工作受到此電路之控制。

而比較器之綫路如圖⑪所示，乃利用

SW1: 單步指令之執行
SW2: 模式選擇
SW3: RESET

D1, D2: 高速二極體
7476 VCC: 5
GND: 13
¼ 7400

5V
4.7K
SW1
N.C
N.0
4.7K
5V
D1
1K
1000P
2K
14
13
15
Rext
cext
C
¼ 74123
B
A
2
CPU SYNC
5V
10K
0.47µ
7
6
4.7K
D2
Rext
cext
B
¼ 74123
A
C
9
SW3
1K
RESET

J CLR Q
¼ 7476
CK
K PR Q̄
3
15
9
6
12
7
4
5
11
10
8
RDY
¼ 7400
連體模組
單步模組
7495之CLK
12
11
13
φ1

十一

2 個比較 IC 比較所決定之頁數，當比較為所指定之頁數時，便會使電路切換到單指令執行狀態，再依據單指令執行狀態之操作模式來操作。當程式執行到其它之 PAGE 時，則又恢復到正常之模式。

增加電路之操作方法

SW2→控制連續狀態 (ON) 或單步執行指令狀態 (OFF)。

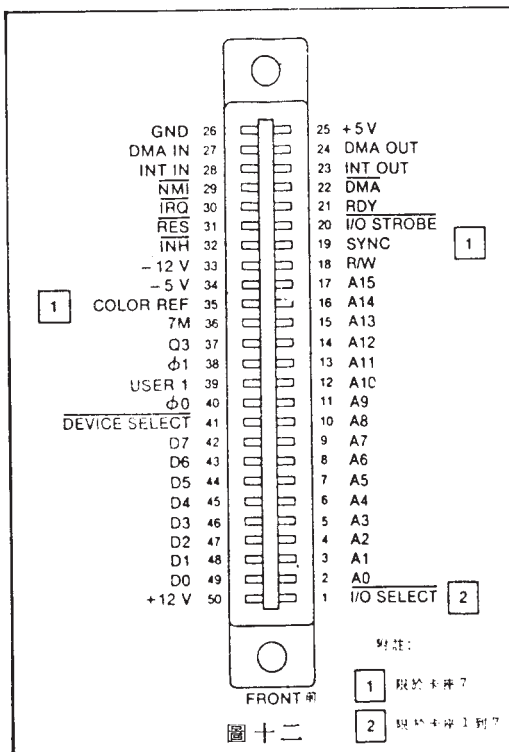
SW1→當 SW1 為 OFF 之狀態時，本 SW 才有作用，每按一次即可執行一個指令週期。

SW3→RESET 其功能與鍵盤上之 Reset 鍵相同。

P.S. : SW1 必須為能保持狀態之開關。

SW2 及 SW3 必須為能自動跳回原狀態之按鈕，各 IC 及

SLOT 之接腳如圖 12 ~ 13 所示。



接腳名稱 編號	信號名稱 [1]	說明 [2]	型式
1	I/O SEL	I	INPUT/OUTPUT SELECT. 7 條已被解碼的選擇線，分別接至卡座 1 ~ 7。在多 2 時間內存取 \$CNO0 - \$CNFF 時，此接腳會變為低電位 N = 卡座號碼。
2-17	AD0-AD15	B	16 位元的位址量流線。在 $\phi 1$ 時間內進入穩定態，並在整個 $\phi 2$ 時間內保持穩定。
18	R/W	B	READ/WRITE. 在 $\phi 1$ 時間內進入穩定態，並在整個 $\phi 2$ 時間內保持穩定。高電位代表讀；低電位代表寫。
19	SYNC	I	Video synchronization signal (C13-8). Connects to slot 7 only.
20	I/O STB	I	INPUT/OUTPUT STROBE. $\phi 2$ 時間內存取 \$C800 - \$CFFF 時，此接腳變為低電位。
21	RDY	B	READY. $\phi 1$ 時間內為低電位時可插入等待狀態，但 6502 只在讀取週期內注意 RDY。
22	DMA	O	DIRECT MEMORY ACCESS. 在 $\phi 1$ 時間之初變為低電位可禁止 6502 動作，並令位址資料，R/W 接腳進入高阻抗狀態。
23	INT OUT	O	INTERRUPT OUT. 串序連接至較低單位卡座。若此卡不具中斷能力，則直接接至 28 接腳。
24	DMA OUT	O	DIRECT MEMORY ACCESS OUT. 串序連接至較低單位卡座。若此卡不具 DMA 能力，則直接接至 27 接腳。
25	+5 V		+ 5 volts
26	GND		Ground
27	DMA IN	I	DIRECT MEMORY ACCESS IN. 串序連接自較高單位卡座，通常為高電位。
28	INT IN	I	INTERRUPT IN. 串序連接自較高單位卡座，通常為高電位。
29	NMI	O	NON-MASKABLE INTERRUPT. 低電位時，發覺 6502 強制中斷請求。
30	IRQ	O	INTERRUPT REQUEST. 低電位時，發覺 6502 中斷請求，但只有在 interrupt disable flag 沒有被設定時方起作用。
31	RESET	B	在寫輸出時，低電位可以 reset 6502 的週邊電路；在寫輸入時，可以因接 reset 鍵，開機或其他週邊電路的觸發，而變為低電位。
32	INH	O	INHIBIT. 低電位時可使 \$D000 - \$FFFF 間的 ROM 不起作用。
33	-12 V		- 12 volts
34	-5 V		- 5 volts
35	COLOR REF	I	COLOR REFERENCE. 連接至第 7 卡座。
36	7M	I	7 MHz 7.15809 MHz 時脈。
37	Q3	I	2.040968 MHz (平均) 時脈。
38	$\phi 1$	I	PHASE 1. 1.020484 MHz (平均) 系統時脈，用於單流線上，以代替 6502 $\phi 1$ 。
39	USER 1	O	低電位時可使 \$C000 - \$CFFF 間的記憶體不記作用。
40	$\phi 0$	I	PHASE 0. 1.020484 MHz (平均) 系統時脈，是 $\phi 1$ 的互補信號，用於單流線上，以代替 6502 $\phi 2$ 。
41	DEV SEL	I	DEVICE SELECT. 8 條已被解碼的選擇線，每個卡座一條，在多 2 時間內存取 \$C0X0 - \$C0XF 時，此接腳會變為低電位。X = N + 8, N = 卡座號碼。
42-49	D7-D0	B	8 位元雙向資料流線。
50	+12 V		+ 12 volts

附註 [1] 除非特別說明，否則信號出現於所有 8 個卡座。

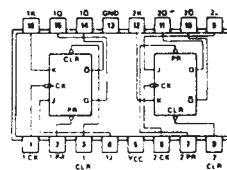
[2] I: Input
O: Output
B: Bidirectional
相對於週邊卡座而言。

'LS 76A.

函數表（右表）

輸入					O 輸出	
預置	清除	取鍵	I	K	O	O
L	M	X	X	X	M	L
M	L	X	X	X	L	M
L	L	X	X	X	M	M
M	M	∩	L	L	O ₀	O ₀
M	M	∩	M	L	M	L
M	M	∩	L	M	L	M
M	M	∩	M	M	TOGGLE	

輸入					輸出	
預置	清除	脈鐘	J	K	Q	Q̄
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H	H
H	H	↓	L	L	Q ₀	Q̄ ₀
H	H	↓	L	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	
H	H	↓	X	X	Q ₀	Q̄ ₀

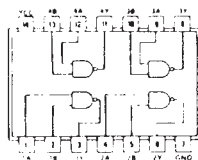


SN5476 (J, W) SN7476 (J, N)
SN54H76 (J, W) SN74H76 (J, N)
SN54LS76A (J, W) SN74LS76A (J, N)

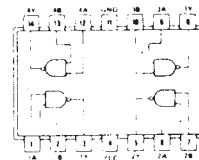
圖十六

00

正邏輯： $Y = AB$



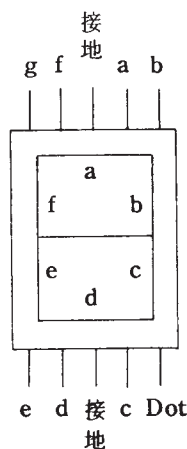
SN5400 (J)	SN7400 (J, N)
SN54H00 (J)	SN74H00 (J, N)
SN54L00 (J)	SN74L00 (J, N)
SN54LS00 (J, W)	SN74LS00 (J, N)
SN54S00 (J, W)	SN74S00 (J, N)



SN5400 (W)
SN54H00 (W)
SN54L00 (T)

圖十七

共陰之七段顯示器



如七段顯示器為共陽
則左圖之接地端改為
接 5V。ROM 內之
DATA 則全部變為 1
的補數。

例：0 字形之 DATA
為 6 F，變為 1
的補數則為 10。

$$\begin{array}{ccc} 110 & 1111 & \leftarrow 6F \\ \downarrow\downarrow\downarrow & \downarrow\downarrow\downarrow\downarrow & \\ 001 & 0000 & \leftarrow 10 \end{array}$$

圖十八

APPLE

標準界面電路

APPLE 有各式各樣的周邊裝置，並可利用 APPLE 主機界面板上的 V — F 型式 A / D 轉換器作為介面，同時以操縱桿 (paddle) 或數位化裝置 (digitizer) 使得與 GAME I / O 接連的 558 振盪器的頻率產生變化，且軟體讀取其頻率變化。

但是上面這種介面方式，並不能作為 sensor 或 trackball 與 APPLE 由介面連接。

這次所要製作的 APPLE 用標準 I / F 單元是作為外部與 APPLE 主機連接之用的標準界面電路，且只要將此界面電路板插入主機板上的輸入／輸出插孔 (I / O Slot) 中便可使用。

利用本 I / F 單元，可以使得 trackball 或 joystick 以及各式各樣的 sensor 與 APPLE 連接。當然也可使用以 APPLE GAME I / O 所作成的連接電路，不過就精密度或速度的觀點而點，還是以作成完全的 I / F 電路之效果較佳。

動作原理

APPLE 若要與外部連接，其中間需要資料的輸入／輸出埠才行。此處使用 68 系列的 PIA (MC6821 或其同等品) 作為這種資料輸入／輸出埠。

現在先將此電路略加說明。74LS645 為雙向性滙流排收發器，74LS04 內有 6 個反相器。而此處 74LS367 只作為緩衝器使用，因此控制線接地。

首先請參閱電路圖 (圖 1)。來自資料滙流排的信號被 74LS645 緩衝後，加入 6821。DEVICE SELECT (裝置選擇) 信號是 APPLE 主機 I / O slot 上所固有的，其位址是依據插入那個 I / O slot，來決定是 \$C080 ~ \$C0FF 中的那些位址。

由於是利用此介面電路單元所插入的 slot 來決定其位址，因此若將基板插入 I / O slot 0 時，便可由表 1 中查得其位址為 \$C080 ~ \$C08F。若想要將此位址加以變更時，只要將作為 A₂ 與 A₃ 緩衝用的 74LS367 的一部份，更換為 74LS04，而後分別連接至 PIA 的 CS 端即可 (如圖 2 所示)。這樣一來原來的電路圖，若要將由 \$C0X8 ~ \$C0XB (X 為任意 16 進數) 至 \$C0XC ~ \$C0XF 為止的位

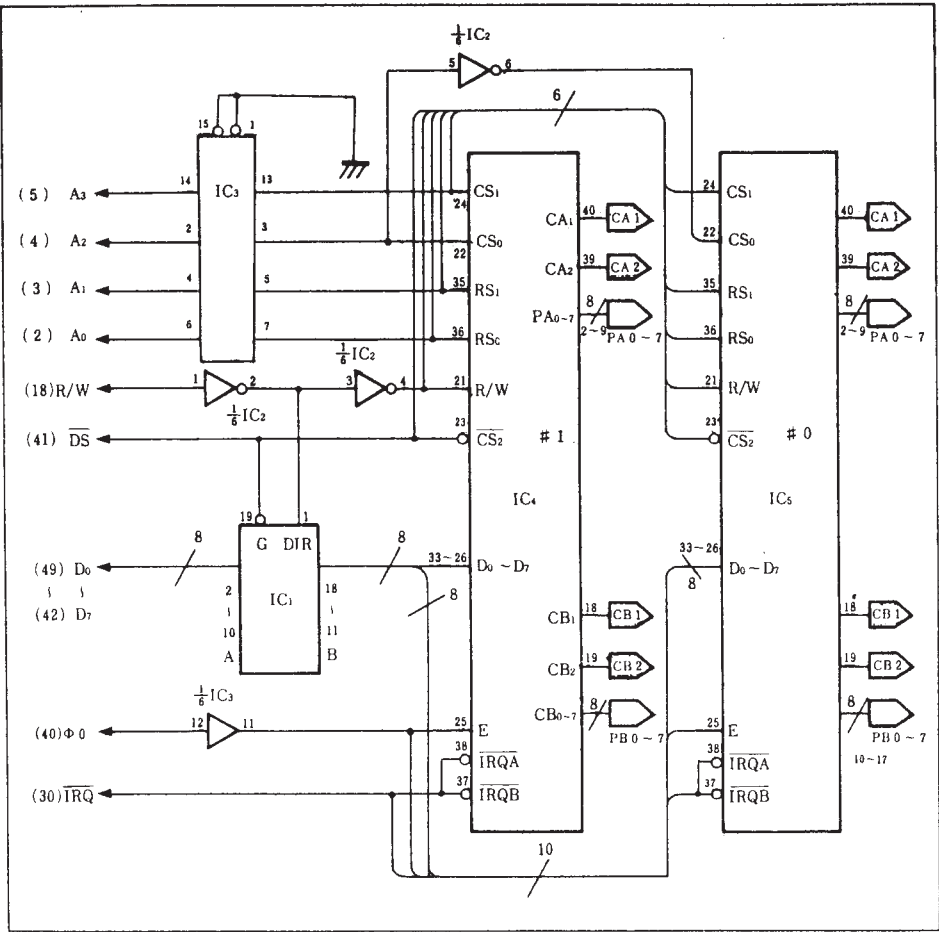


圖 1 電路圖

位址	插孔	編號
C08X	DEVICE SELECT	0
C09X	//	1
C0AX	//	2
C0BX	//	3
C0CX	//	4
C0DX	//	5
C0EX	//	6
C0FX	//	7

表 1 I/O 插孔與位址

X：任意的 16 進數

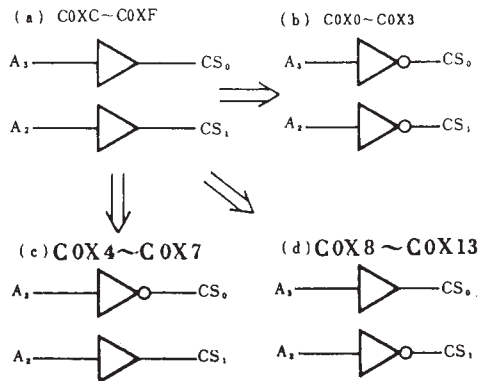


圖 2 位址的變更方法

加至 2 個 PIA 時，則只要依照圖 2 (a)與(c)的狀態加入 IC₄與 IC₅即可。

這裡所使用的 6821 是稱為周邊介面轉接器 (peripheral interface adapter , PIA) 的 68 系列 LSI，因為 65 系列的時序與 68 系列大致相同，因此我們在這裡加以使用。而為了能更有效的利用這個標準周邊界面電路，所以更應該去研究和了解 PIA 的控制和

使用方式。

PIA結構簡介

關於 PIA，在此作一簡單說明。首先請參考圖 3 的 PIA 內部方塊圖，由此圖可知 PIA 是由相同的 A，B 兩系統所構成的。此處僅就 A 系統加以說明。

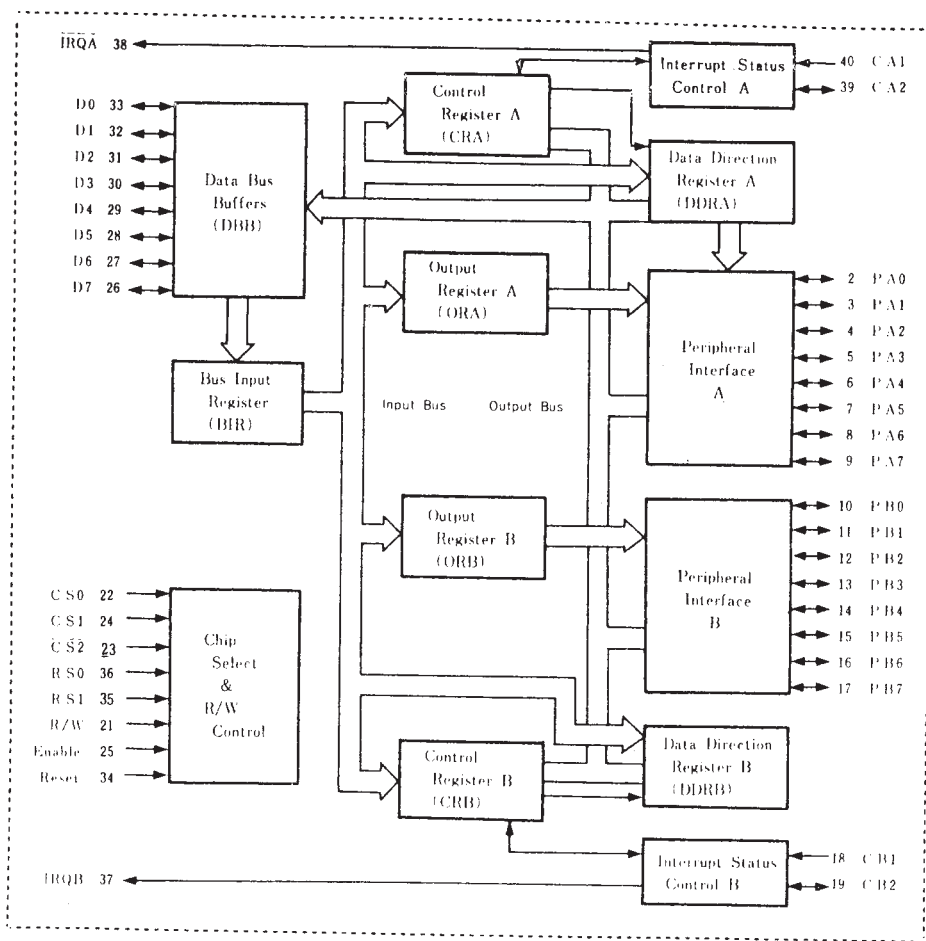


圖 3 方塊圖

A 系統佔據此方塊圖的上半部，其中 DBB 與 BIR 為 A、B 兩系統所共用，而 CRA、ORA、CA、DDRA 與周邊介面 A 為系統 A 所專用的部份。其中最重要的該屬 CRA、DDRA 與周邊介面 A 等三部份。

PIA 之所以稱為是通用的 (general purpose)，最主要的便是因為其功能係依其初期值 (initialize) 的設定而有所不同，而這些初期設定資料的寫入地方是 CRA。CRA 的各位元為如圖 4 般所示構成。

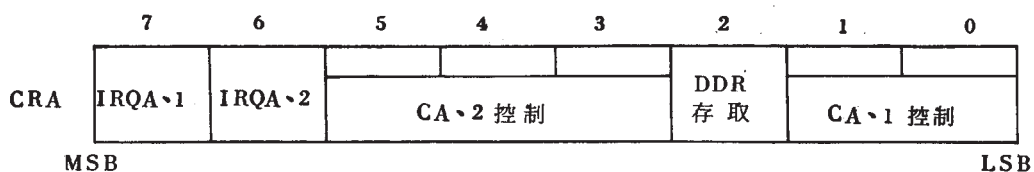


圖 4 控制字

此處請注意位元 2，位元 2 寫著 DDR 存取（DDR Access），當此位元為 1 時資料分派至周邊界面 A 暫存器，而當此位元為 0 時，資料則被分派至 DDRA，亦即這種狀態時資料被存取至 DDRA。

因此依據 DDRA 所寫入的資料為 1 或 0 時，便可對 A 暫存器各位元的方向加以決定。此時對應於 1 的位元方向為輸出，而 0 時則為輸入。例如 DDRA 中寫入 11111111 時，A 暫存器的全部位元被設定為輸出，而若寫成 00000000 時，A 暫存器的各個位元變成輸入狀態。

依據這種功能，A 暫存器便可被使用作輸入暫存器或者輸出暫存器，當然因為這些方向

都是將每個位元加以設定之故，因此也可以將偶數位元當作輸出，而奇數位元當作輸入而加以設定，此時 DDRA 中所寫入的資料為 01010101。依據這樣，A 暫存器的方向也可任意地加以設定。

另外，CRA 的位元 0、1 作為 CA1 的功能設定，而位元 3、4、5 作為 CA2 的功能設定。位元 6、7 為中斷旗號，這些旗號分別對應於 CA2、CA1 的輸入而被加以設定。當 CA2、CA1 設定為中斷許可時，便會有中斷發生。

R/W 信號作為控制 74LS645 的方向轉換之用。

APPLE 主機因為利用 I/O slot 將位址解碼信號線輸出，因此電路變得簡單而且易

零件名稱	型 號 （ 規 格 ）	數 量	備 考
通用 I/O 埠	PIA(6821)	2	
TTL	74LS645 74LS04 74LS367	1 1 11	74LS245 亦可
IC	40P 20P 14P 16P	2 1 1 1	依照製作方法自行選擇
旁路電容器	陶瓷電容器 0.1 μ F	數 個	
接 線 柱		適 量	配線時需要
排線連接器	FAP-34-03	2	
基 板	APPLE 用 萬用基板	1	

表 2 零件一覽表

做。

表 2 為零件一覽表。

本單元電路爲了要能與外部的 sensor 或 trackball 等以排線 (flat cable) 得以

連接起見，因此在這裡將稱爲接頭的排線連接器安裝在其電路板上，這樣只要將排線插入連接器中，介面電路便算大功告成了。零件的配置如圖 5 所示。

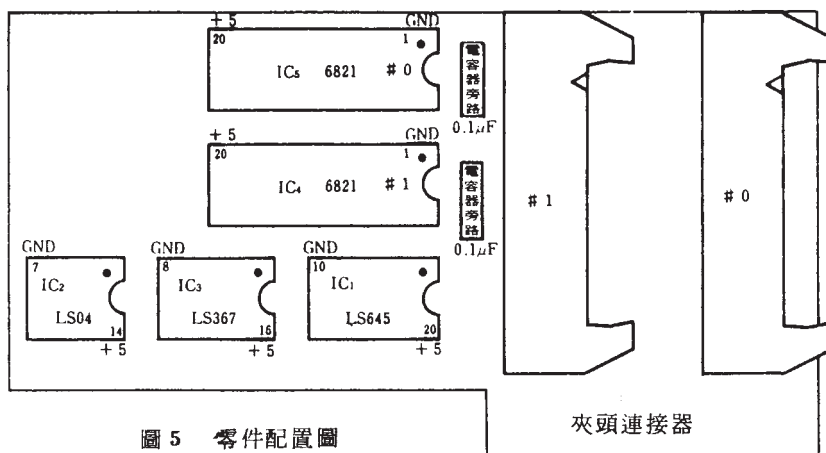


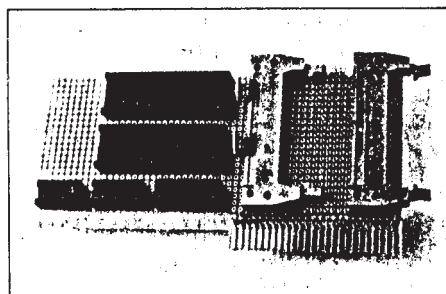
圖 5 零件配置圖

動手製作

本製作是使用萬用篩狀基板，因此是以跳線方式來連接電路，不過由於各個插座接腳相當接近之故，跳線時便需要對線與線是否接觸良好，以及各跳線是否會被接腳所割破而造成線間短路等情形加以留意。若在插座同一面連接跳線，由於插座接腳間過於接近，因此容易產生麻煩。解決之道便是將元件與配線分別在正反兩面進行，如此會減少許多不必要的問題發生。同時爲了避免將此基板插入 slot 中時，與相鄰 slot 中的其他介面卡形成面對面而增加兩卡間接觸的可能性，因此不僅要將元件與配線分成兩面，同時也請務必將元件配置面與其他介面卡一樣的安排在同一方向。

首先在電路板上必要的地方將旁路電容先行插好，然後再將其餘元件分別加以配置妥當 (照片一)。元件插好後，接著進行配線連接。此時可將電路圖複印成 2 份，其中一份作爲配線時之用，另一份則作爲配線檢查時使用。

配線時若能將電源線、位址匯流排、資料



照片 1 元件配置的情形

匯流排、控制線等分別以各種顏色加以區分，那麼，檢查時就會輕鬆愉快了。

標準連接器處若混入配線所用的線時，此時務必要好好考慮連接器各接腳張開時的裕度。

接至連接器的配線，請參考圖 6 的 I/O slot 信號表。

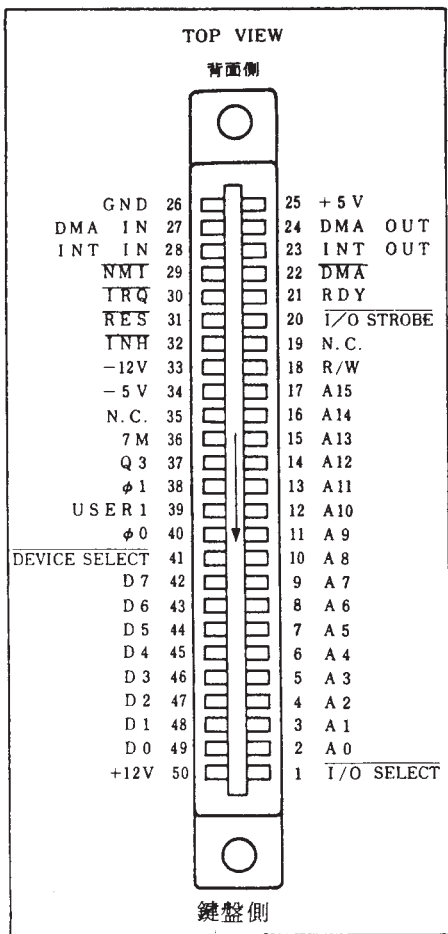
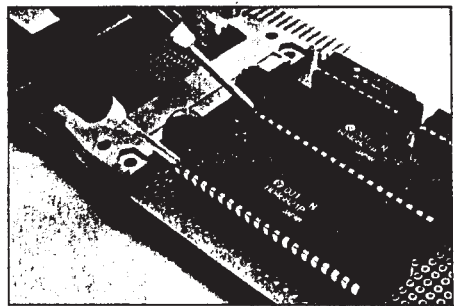


圖 6 APPLE II I/O 插孔信號表

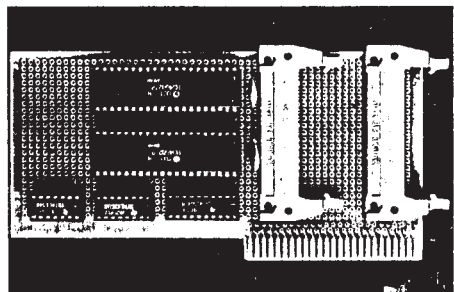
動作測試

首先對 IC 的插入狀態作導通檢查，並檢查配線是否有誤。檢查時可以前面所複印的電路圖加以對照，假如發現有所不同，立即加以修正，當完全無誤時，便表示配線已完全正確無誤。

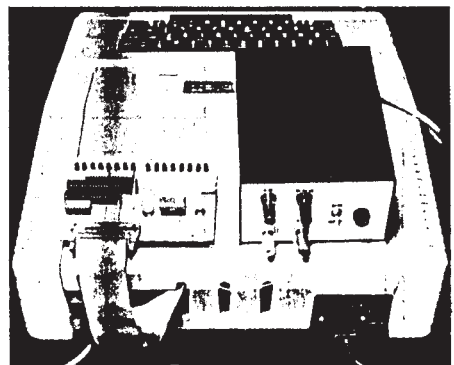
配線檢查完成後，將此界面電路板插入 I/O slot NO.4 的 slot 中。挿妥後請加上電源，此時可以 BASIC 語言執行看看，若有異常，則請關掉電源，並利用排線使與輸出入埠檢查器 (port checker) 連接。輸出入埠



照片 3 配線檢查



照片 4 已完成的標準 I/O 電路



照片 2 與 APPLE II 的接線

檢查器本身不具有電源 (照片 2)，因此當將輸出入埠檢查器的資料開關 OFF，則輸出入埠檢查器的電源與 APPLE 的電源同時加入，而 LED 全亮。此時若無法確定是否異常，可以將測試程式 (程式 1) 加以執行。

程式 1 測試程式(a)

```

10 REM *****
20 REM * TEST PROGRAM 1 FOR APPLE 2 *
30 REM *****
40 POKE -16129,0
50 POKE -16131,0
60 POKE -16130,255
70 POKE -16132,255
80 POKE -16129,4
90 POKE -16131,4
100 FOR A=0 TO 255
110 FOR B=0 TO 100
120 NEXT B
130 POKE -16130,A
140 POKE -16132,A
150 NEXT A
160 GOTO 100

```

程式 1 測試程式(b)

```

10 REM *****
20 REM * TEST PROGRAM 2 FOR APPLE 2 *
30 REM *****
40 POKE -16131,0
50 POKE -16132,0
60 POKE -16131,4
70 PRINT PEEK (-16132)
80 GOTO 70

```

(a)的程式是當電源ON時，將變成輸入型態的PIA的方向輸出，而由下位位元起，依序使LED一個接一個發光。

(b)為輸入的測試程式。首先請鍵入測試程式。其次將輸出入埠檢查器的資料輸入開關ON，以執行測試程式。此時會在畫面上將輸入開關的值以十進數輸出，若開關的值與畫面的值相同，則一切OK。

若測試程式不同便無法進行測試，因此請務必輸入相同的測試程式。

故障排除

這裡是當諸位的介面板一動也不動時才用得着的。當電路一動也不動時，可能有許多的地方或許出了問題。譬如說IC壞了，特別是IC因為發熱而燒壞，或者錯誤的配線而導致等等。

① IC變熱→IC是否反向插入，IC的電源接線是否有誤，IC由頂規圖來看是否接腳搞錯等都要加以檢查。74LS645依其結構可能會變熱，其他IC大體上還不致發熱。

② 加入電源時，APPLE不動作或者動作

異常。74LS645的data接腳相反。74LS-645的控制線配線錯誤。

③ 輸入、輸出資料不合→PIA與連接器間的配線有問題，因此PIA的位址不同。

問題大致只可歸納為上述三種，不過諸位不要因為故障小而加以忽視，若怠於檢查，等到釀成大問題而使系統遭到破壞時再踩腳就太遲了。

測試時將此介面板插入I/O slot的NO.4中檢查，不過若將其插入其他的slot中也無妨，只是因為此時的位址改變，因此當要將此介面板插入其他slot中檢查時，請將位址如表1所示加以改變。

GND	●	●	空腳
GND	●	●	空腳
GND	●	●	空腳
CA1(A $\overline{STB}$)	●	●	空腳
CA2(A $\overline{RDY}$)	●	●	PB7
PA0	●	●	PB6
PA1	●	●	PB5
PA2	●	●	PB4
PA3	●	●	PB3
PA4	●	●	PB2
PA5	●	●	PB1
PA6	●	●	PB0
PA7	●	●	CB2(BRDY)
空腳	●	●	CB1(B $\overline{STB}$)
空腳	●	●	GND
空腳	●	●	GND
空腳	●	●	GND

表 3 標準連接器信號表

輸出入埠檢查器

輸出入埠檢查器 (I/O PORT CHECKER) 是用以確認標準 I/F 單元電路的動作，此處我們用16個LED來作 I/O 狀態的顯示。

此檢查器經由排線與標準 I/F 單元電路連接，I/O 埠的狀態 (各位元為 "H" 或 "L") 以 LED 表示，同時資料開關的狀態也可以讀入 I/O 埠中。

表 3 為標準的連接器信號表，連接器由正面所看到的三角印與表中的三角印位置相對應。

各接腳的說明如下：

GND 全系統共同接地線。全部的信號線因為是以此線為標準以決定其電壓，因此此線一定要與主機的接地連接。

PA0~7 ... I/O 埠 LSI 的 A 組 8 條資料線。

PB0~7 ... I/O 埠 LSI 的 B 組 8 條資料線。

CA1 A 組所用的控制線，此處於開關資料輸入時使用。

這些線中，PA0~7 及 PB0~7 的 16 條資料線位準以 16 個 LED 來表示。亦即各接腳為 "H" 時 LED 發亮，而 "L" 時 LED 不亮。這

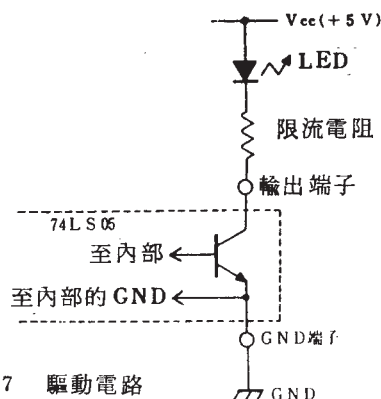


圖 7 驅動電路

些表示與埠的輸入、輸出狀態無關。輸出入埠於輸出狀態時的輸出資料或輸入狀態時的資料輸入開關若 ON 時，則表示該資料，而 OFF 則所表示的為不確定的資料。只是，資料開關只對 A 組的 8 位元有效，而無法有來自 B 組的輸入。

本檢查器電路可以分為 LED 驅動電路，資料開關及激勵開關三部份。

LED 驅動電路是作為使 LED 發光的電路，全部 16 組 LED 完全相同。分別是由開集極反相器 (NOT 電路)，電阻及 LED 所組成。當反相器輸入為 "H" 位準時，電晶體導通，電流由 $V_{cc} (+5V)$ 經由 LED，電阻

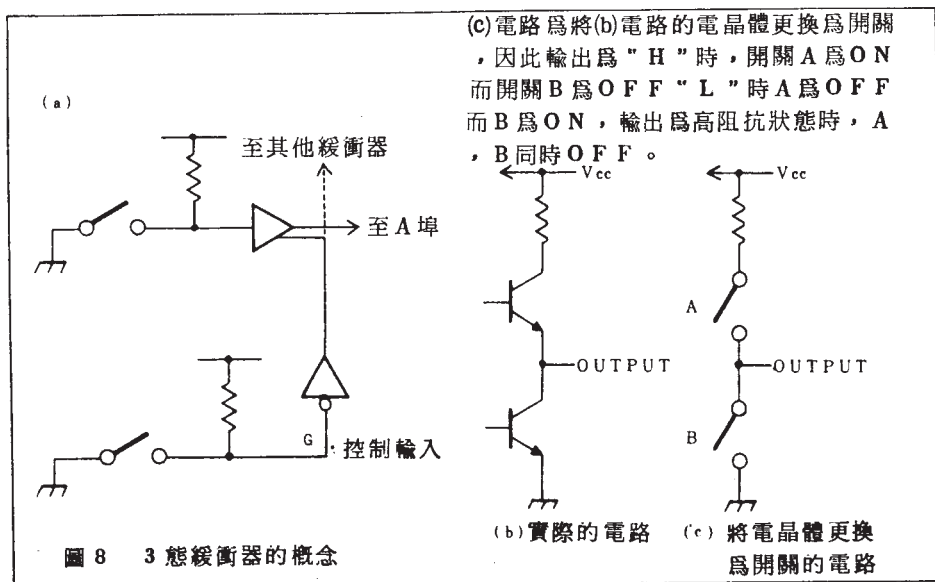


圖 8 3 態緩衝器的概念

流至接地，而使LED發亮；反之，反相器輸入為“L”時，無電流流動，而LED不發亮。如圖7所示即是LED驅動器的電路。

資料開關是與8個位元分別連接的開關，由1個控制開關與3態緩衝器所構成。這個方塊送資料給A埠，為了避免導致電流超過額定值，錯誤動作，異常發熱等足以損壞IC的情形，因此此處使用3態緩衝器IC。決定此IC的輸出為ON（“H”或“L”）或者OFF（高阻抗狀態）的便稱為控制用開關。當然決定資料的“H”或“L”的開關與此3態緩衝器的控制輸入相連接，如圖8所示。

激勵開關與輸出入埠LSI的激勵端子相連接，關於激勵端子的說明請參閱各輸出入埠用LSI的說明。

關於此輸出入埠檢查器使用的零件如表4所示，在此作一簡單說明：

TTL 為使用小功率動作的LS型，

74LS05（或74LS04亦可）內有6個NOT電路，LS00內有4個NAND閘，而LS244為8個3態緩衝器。

集合電阻為內有數個電阻，其中898-3內有8個獨立電阻，而898-1則內有15個電阻。

排線連接器為使用包裹線用的34腳元件。

電源連接器為5個接腳，其中包括Vcc（+5V），邏輯接地，類比電路用的電源（±15V）兩條，以及一支空腳。

激勵開關因為是與R-S正反器連接，因此使用轉換型的按鈕開關，資料輸入開關使用常用的跳擺（或稱撥鈕）開關，而資料開關則可使用鍵盤開關。

整個輸出入埠電路圖如圖9所示。而其零件配置如圖10所示。

與此I/F單元電路同時需要完成的事實上也只有輸出入埠與輸出入埠檢查器而已，並

元 件 名 稱	型 號 (規 格)	數 量	備 考
基 板	ICB 93W	1	篩 狀
LED		16	普通市售品即可
TTL	74LS05	3	
	74LS00	1	
	74LS244	1	
集 合 電 阻	898-3-R330	2	
	898-1-R4.7K	1	
I C 插 座	14P	4	依照製作方法自行決定
	包含DIP開關	4	
	20P	1	
排線連接器	FAP-34-03	1	依照製作方法自行決定
電源連接器	Morex L型5P	1組	5P以上
開 關	DIP開關（鍵盤）8P	1	
	按鈕開關（跳擺開關）	1	
		1	
接 線 柱		適量	有配線時需要
旁路電容器	0.01 μ F	數個	

表 4 輸出入埠檢查器零件一覽表

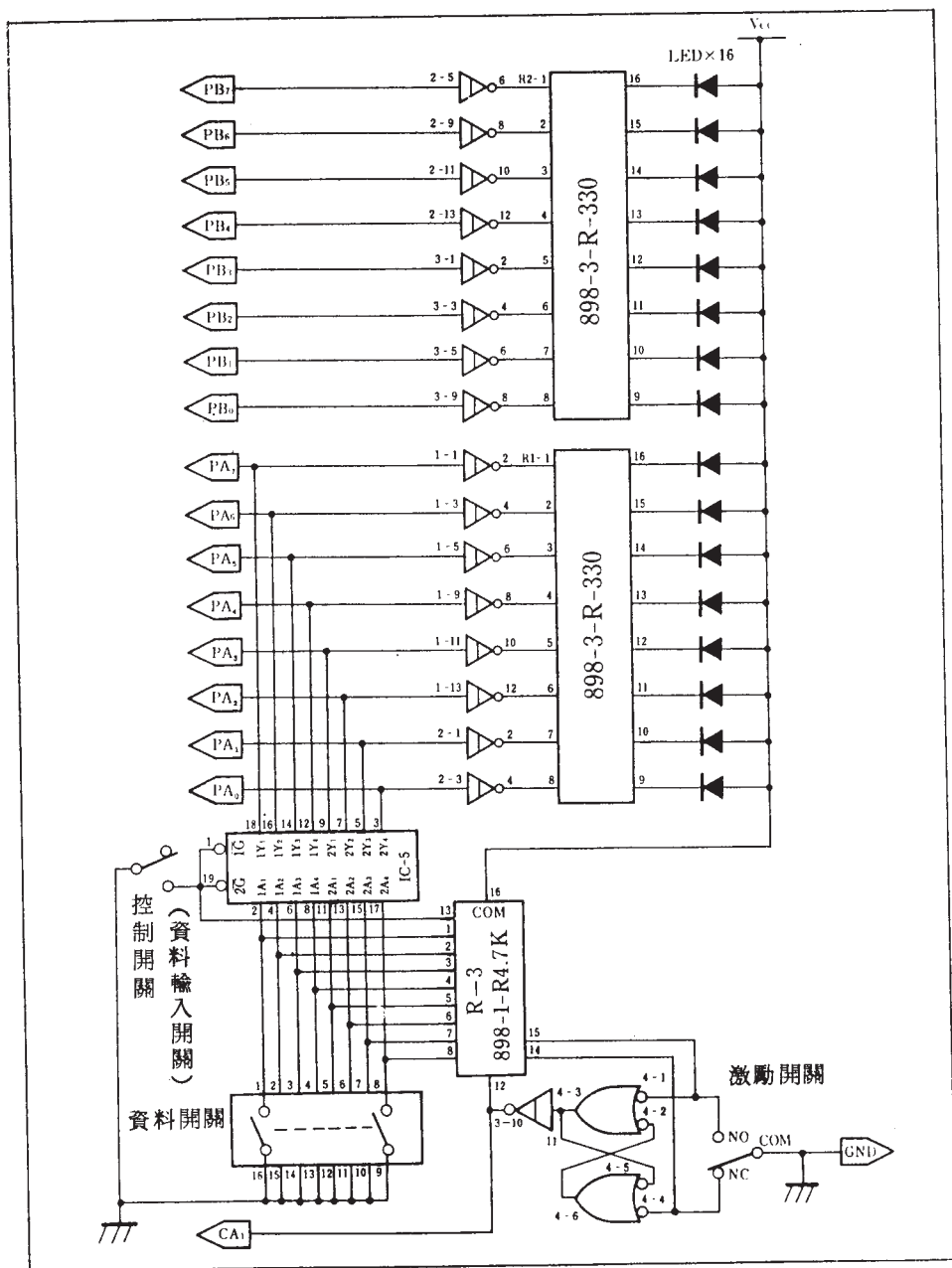


圖 9 電路圖

無多大意義。而唯有用以和 sensor 等外部電路連接使用時，才能真正發揮其價值。另外這

個基板也可以考慮作中斷使用。關於中斷的問題，請參考有關 P I A 的資料。

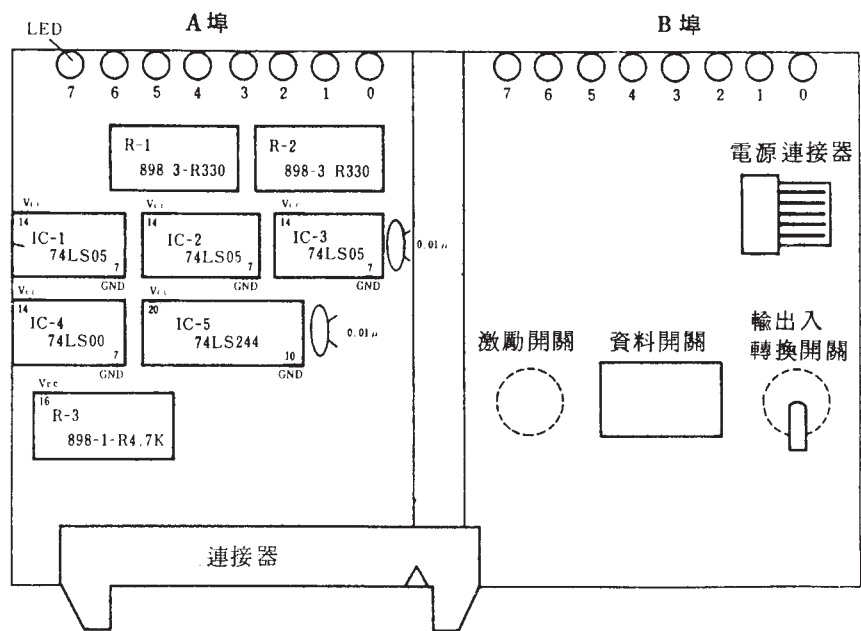
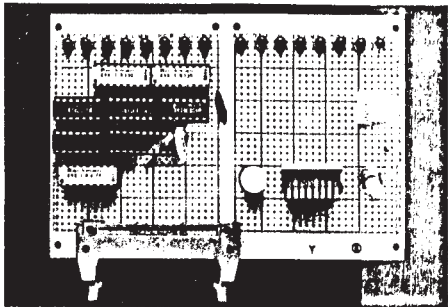


圖 10 零件配置圖



照片 5 完成的輸入埠檢查器

6809 CARD

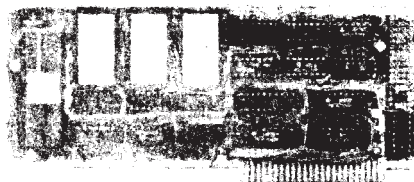
原理與製作

APPLE II 在設計當初有一偉大的傑作，便是留有 I/O 插孔，以供未來使用時的各種周邊擴充之用，本文便是要向各位介紹一種使用 6809 CPU 的卡。

6809 是美國 Motorola 公司所新近開發完成的微處理機 CPU，自推出以來，廣受使用者的注目，最近甚至有雜誌這麼說「在日本，若談論到 CPU，則不是指 Z80 便是 6809 莫屬了。」，由此可見，6809 之廣受重視，不過奇怪的是，6809 在國內却一直沒有受到應有的重視，因此本文特別介紹 APPLE II 6809 卡的原理與製作，希望能經由這篇文章，喚起您對 6809 的重視，另一方面，6809 與 6502 屬於同一系列，因此對於 APPLE 使用者而言，6809 卡應該是相當能夠接受的。

只要將此卡插在 APPLE II 的第 4 插孔，那麼 APPLE II 便搖身一變而為 6809 的微電腦了。不過此卡不是僅在將 APPLE II 轉換為 6809 機器，我們當然亦可由 6809 CPU 來充分的利用整個 APPLE II 的硬體設備，甚至自由地使用 APPLE II 所具有的所有軟體設施。

亦即，不僅可由 6809 來呼叫 6502 的副程式，反之，亦可自由地由 6502 來呼叫 6809 的副程式。



照片 1 6809 卡本機

這樣一來，由於此 6809 卡的功能，可以使得 APPLE II 系統的功能引伸至相當大的極限，而使以往 APPLE II 無能為力的事，甚至變為可能。

同時，6809 卡還具有可連結多種命令與 DOS 3.3 的強力監督器程式，因此 APPLE II 可被當作 6809 發展系統使用。

另外，亦可以 APPLE 的磁碟來執行 FLEX。

系統的概要說明

由於 6809 具有較為優越的能力，因此可以使得原本 6502 執行起來覺得困難異常的許多工作都變成簡單許多。因此若能以 6809 來

取代 6502 所能執行的某些工作，將會大幅地降低機器本身所能擔負的工作負荷。

例如，在 A/D 轉換器等的应用場合，以 6502 取入 8 位元資料 1024 個時，無論如何 1 個資料都需花費 $24\mu\text{S}$ ，而 16 位元一次取得則需花費 $36\mu\text{S}$ 左右。

6809 在 8 位元時為 $23\mu\text{S}$ ，並無多大差別，而在 16 位元時也需要 $26\mu\text{S}$ 來讀入。

然而，在浮動小數點運算等乘算指令或 16 位元運算指令時，具有使用者堆疊器 (user stack) 強力功能的 6809，就顯得比 6502 在處理能力及速度上高明多了。

這樣當在 APPLE II 中插入 6809 卡時，APPLE 便可以 6809 當為後補 CPU 般，而同時取得兩 CPU 相互具有的優點。

在作 6809 的系統開發時，對於磁碟的讀／寫副程式或印字機的輸出副程式等，並不需要另外作成 6809 專用的軟體，只要呼叫 APPLE II 的副程式便可以了。

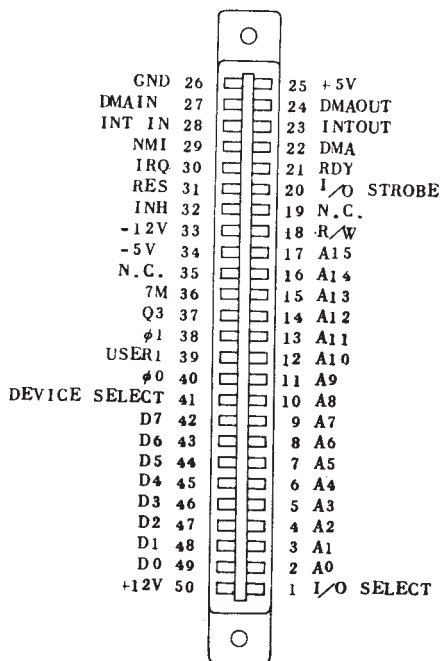


圖 1 I/O 插孔的連接器接腳圖

後面所要提到的監督程式，也如 DOS 3.3 的連結程式或 6502 的解組程式，錄音機的輸出入副程式一樣，照樣呼叫 APPLE 的監督程式便可。

因此，我們可以這樣說，插入 6809 卡後，就如同在 APPLE II 的主機板上，6502 與 6809 共存一般。

在 APPLE 上 RUN 6809 的秘密

APPLE II 共留有 8 個 50 接腳 I/O 插孔作為系統擴充之用。圖 1 所示為 I/O 插孔連接器的接腳圖，具有各種介面連接所需的位址，資料幹線以及控制信號接點。

6809 卡能夠在 APPLE 上執行，最主要的便是利用 $\overline{\text{DMA}}$ 信號。 $\overline{\text{DMA}}$ 是直接記憶體存取 (direct memory access) 的縮寫，是指資料能夠不經由 CPU 而直接與記憶體作存取的一種控制信號。當 APPLE 的 $\overline{\text{DMA}}$ 信號為 "L" 時，提供給 6502 的時脈信號停止，同時位址、資料幹線與 $\text{R}/\overline{\text{W}}$ 信號變成隔離 (H-Z) 狀態。

這樣 6502 便被切離系統，而成為 HALT (暫停) 狀態。如此，其他的外部裝置便可以自由地與記憶體作存取工作。此處所指的其他外部裝置在本文中指的自然是 6809 卡了。

若在 6809 卡中，此 $\overline{\text{DMA}}$ 信號被反轉而當作 HALT 信號時，由於 $\overline{\text{DMA}}$ 為 "H" 時，6809 變成 HALT 狀態，因此又由 6809 CPU 跳出而返回 6502 CPU。

因此，只要對此 $\overline{\text{DMA}}$ 信號妥善地加以控制，便可在 6502 與 6809 兩者間交互地執行了。

此外，因為 APPLE 是一具有擴張性的機器，因此具有 $\overline{\text{INH}}$ 信號。此信號與 APPLE 主機板上的 6 個 ROM 上的 CE (晶片選擇 chip-select) 端子相連接，只要此信號為 "L"，主機板上的 ROM 將全被封殺掉。

這便是 APPLE 具有 64K RAM 時，可以一個 ROM 卡上的開關來切換成 6K 或 10K BASIC 的秘密所在。LANGUAGE CARD 便

是利用此 $\overline{INH}$ 爲 "L" 來作卡上 RAM 的切換工作。此 6809 卡也可使用 LANGUAGE CARD。

以上所述乃是當初設計 APPLE 時的設計構想所在，希望您能充份地加以利用。

副程式呼叫的秘密

前面曾經提到，若能對 $\overline{DMA}$ 加以充份地控制，便可隨心所欲地作 6502 與 6809 的切換。然而又要如何由 6809 呼叫 6502 的副程式，或者由 6502 呼叫 6809 的副程式呢？

假如只是經由控制 $\overline{DMA}$ 來作 CPU 的切換，僅能由對方 CPU 變爲 $\overline{HALT}$ 後的下一指令開始執行而已，當然也就不可能作副程式的呼叫。

因此，無論如何於 CPU 切換時，若能使之同時跳接至某一特定的位址，便可以進行副程式的呼叫了。

此處是使用 NMI (Non Maskable interrupt 不可阻擋中斷) 使之跳至某一特定位址。也就是在以軟體方式進行 CPU 切換的同時，也使其發生硬體的 NMI 中斷。

圖 2 爲此時序圖的表示。CPU 切換的同時約 $2\mu S \sim$ 數 μS 之間，NMI 爲 "L"。被解除 $\overline{HALT}$ 狀態的 CPU，由次一 ϕ_0 起進入指令執行週，而由於 NMI 變成 "L"，所以也同樣進入 NMI 的中斷裡。

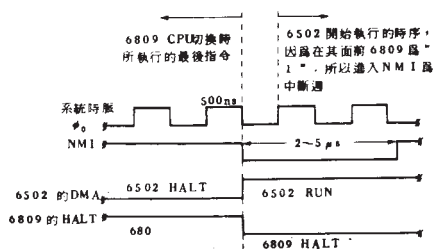


圖 2 CPU 切換時序爲 (6809 $\rightarrow$ 6502)

當進行副程式呼叫時，若能在呼叫端設定「對方的 NMI 向量位址」(NMI Vector Address) 與「本身的返回向量位址」，並且進行 CPU 的切換是最好不過的了。

當然，就對方而言，由副程式返回時，若不作 CPU 切換顯然是不行的，這些切換時所需要的程式被當作副程式準備在 6809 的監督程式內。

6502 的切換程式則是在 6809 被復置時，自動地被轉送至 RAM 區域 ($\$370 \sim \$3CF$)，因此無需對此多費心思，便可加以使用。

實際的切換程式都是在暫存器中進行授受的工作。這些工作爲了配合 6502 起見，在 6502 的 Acc.X.Y 和 6809 的 Acc.X 的下位 8 位元以及 Y 的下位 8 位元中，分別進行授受。

同時爲了確實地對 CPU 切換時的 NMI 加以判斷，因此兩者之間可以使用特別的關鍵字 (key word)，以提高動作的信賴性。

不過，此處不可不加以注意的是這種 CPU 切換動作的處理往往需要數十 μS 左右的時間。因此，若以提高處理速度爲目的來作 CPU 切換，便不能只進行單一資料的呼叫，最好是將某種程度的資料貯存後再進行呼叫工作。

位址配置區的變更

APPLE II 的記憶配置圖如圖 3 (a) 般，這個配置圖 6809 卡也能適用，不過爲了讓 6809 卡將來能跑 FLEX 起見，便有必要加以變更。由於 FLEX 由最低限度的 0 位址開始連續 16K 位元組有 DOS 存在，因此 RAM 便非安排在 $\$C000 \sim \$DFFF$ 的 8KB 不可。

因此爲了滿足這些條件，6809 卡的記憶配置圖變更爲如圖 3 (b) 所示。此配置圖若僅用在 6809 是一點也不會有問題的，不過若要呼叫 6502 時，對於共通區域的存取便非要加以留意不可了。

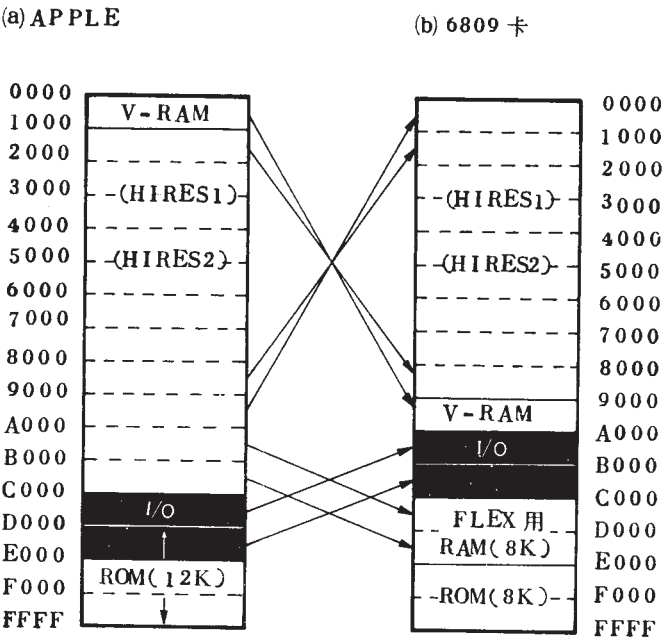
例如，6502 的 $\$0400 \sim \$07FF$ 爲 V-RAM 的內文 (TEXT) 第 1 頁，在 6809 則變更爲 $\$9400 \sim \$97FF$ 。而 I/O 位址也由

6502 的 \$CXXX 位址，變更爲 6809 的 \$AXXX。

這些變更只要習慣了便好了，若將來並不

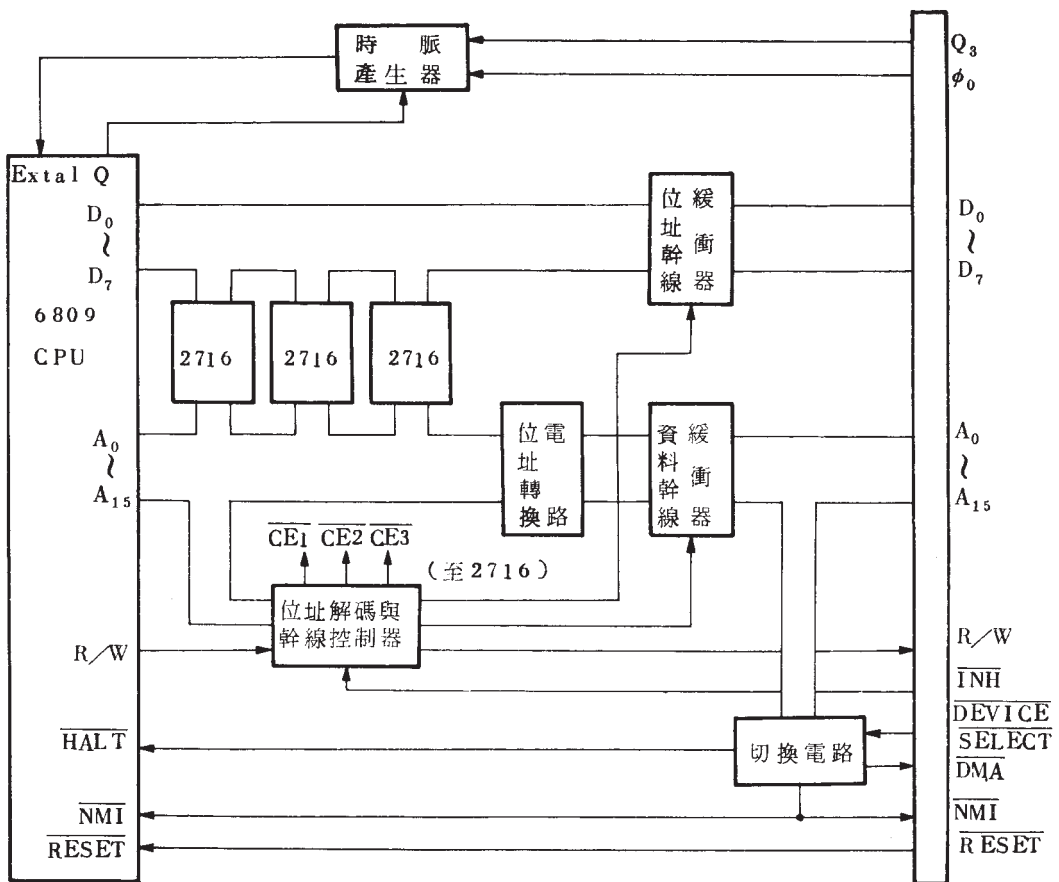
預定要跑 FLEX，那麼配置圖與 APPLE II 一樣是沒有關係的。這種變更在硬體上是很簡單的，但軟體上非要重新加以組合不可了。

圖 3 記憶配置圖的變更



位 址	用 途
0000~8FFF	使用者 RAM
9000~91FF	6502 零頁與堆疊器
93D0~93FF	6502 特殊區域
9200~93FF	6809 系統工作區
9400~97FF	TEXT 第 1 頁
9800~9BFF	TEXT 第 2 頁
9C00~9FFF	自由區域
A000~AFFF	I / O
B000~BFFF	沒使用
C000~DFFF	FLEX 用 RAM
E000~FFFF	監督程式 RAM

註: 2000~3FFF HIRES1
4000~5FFF HIRES2



電路說明

圖 4，圖 5 是表示此 6809 卡的方塊圖與全電路圖。其中最具特徵的電路是 6809 的時脈電路與 CPU 的切換部份。

前面我們提到過 DMA 為 "L" 時，時脈信號停止，而 CPU 變成 HALT 狀態，不過時脈信號的停止是僅指給與 6502 的時脈而言，時脈信號一樣是照舊加至整個系統。

因此，6809 在對 APPLE 作記憶存取等工作時，當然便不可忽視此時脈，因而必須對此時脈的同步問題加以考慮。

時脈的同步

APPLE 跑 6502 以外的 CPU，並非始

自今日，Microsoft 公司的 Z80 卡便是一例。不過這是利用加 WAIT 至 Z80，使其毫無理由地與 APPLE 系統可以取得同步，因此，多少總得犧牲某種程度的執行速度。

因此，我們不想用 WAIT 或時脈一起動作來取得與 APPLE 系統的同步。而是在 6809 執行時不損及速度的原則下來與 APPLE 的 ϕ_0 同步執行。

事實上，時脈的同步是此 6809 卡在開發時所遇到的最嚴重問題。

實際電路中，APPLE 系統的時脈 ϕ_0 是與 6809 的 E 相當，因此，無論是那邊的 CPU 要與外部作資料交換，都是在此信號 "H" 的時候進行。時脈的同步便是指此 ϕ_0 與 6809 的 E 取得一致而言。

圖 5 (a) 6809 卡的全電路圖 (CPU 周邊)

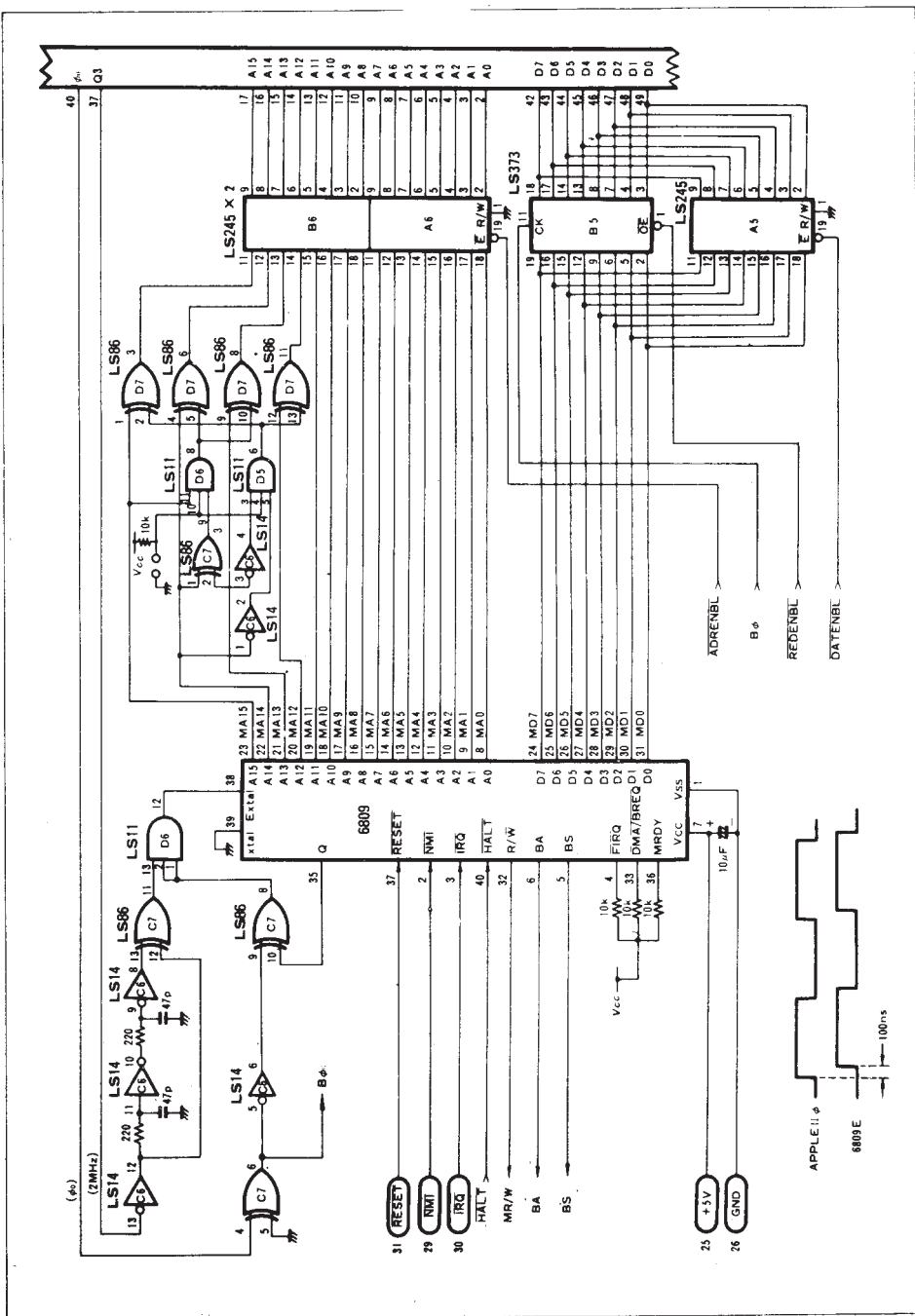
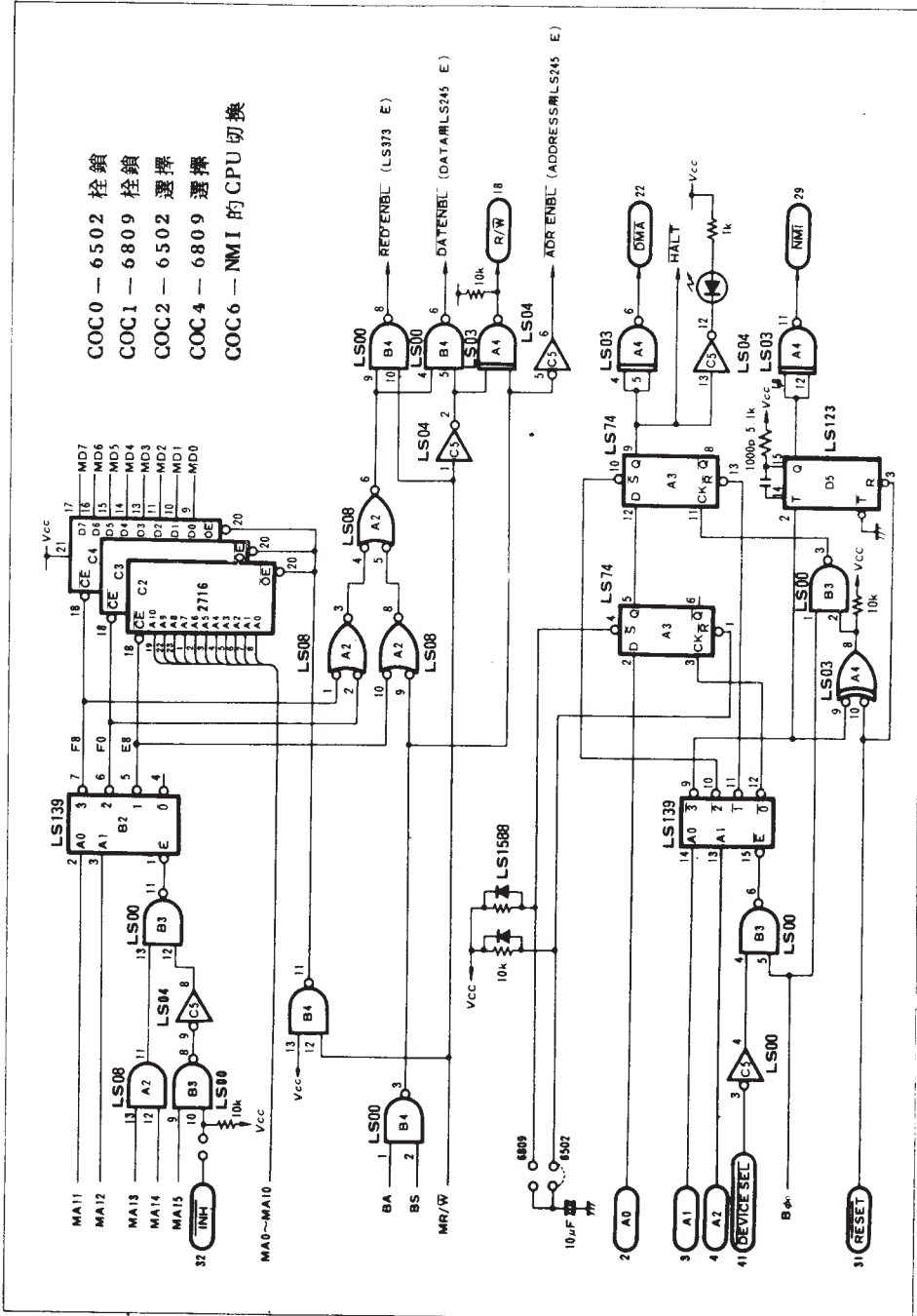


圖 5(b) 6809 卡的全電路圖 (位址解碼器, ROM, CPU 切換電路)



因為 6809 的時脈便非要是 4MHz 不可。然而在 APPLE 的 I/O 插孔不能得到此 4MHz 的頻率，因此將來自插孔的 Q_3 (2MHz) 倍頻以作成 4MHz，當作是 6809 的時脈。

不過，這樣並不能保證由 6809 所輸出的 E 與 APPLE 端的 ϕ 間的相位能正確配合，所以將由 6809 所輸出的 Q 加以回授，將其與 APPLE 的 ϕ 作相位比較來加以解決。

不過這裡還有一個問題，也就是 6809 的時脈產生器對於輸入的 4MHz 而言，輸出的 Q，E 大約有 100 μ S 的延遲存在 (照片 2)。此處由 6809 所輸出的時脈，為了與 6502 同步。而使用回授方式，所以 6809 的讀／寫動作完全與 APPLE 的 ϕ 同步。

因此，6809 在 E 下降時將資料讀入，而此時資料幹線上的資料全都被消滅掉。

如圖 6 所示，6809 所讀入的資料一旦經過栓鎖器，則於 ϕ 變為 "L" 後資料也會被保持住。而於寫入動作時，6809 於 Q 上升 225 以後，對輸出資料便可加以確定，因此不會構成任何問題。

這種情形，若能由我們在後面將要敘述的外部時脈輸入型 6809 E 着手，大致上便可加以簡化。

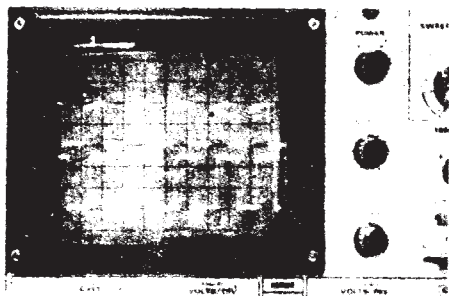
CPU 切換電路

這個電路便是前面曾經說明過的控制 $\overline{DMA}$ 以作為 CPU 切換的部份。電路是由 2 級的 D 型正反器所構成，利用分別對表 1 所示位址的存取來加以設定或復置。

因為使用這種方法，一旦 CPU 被切換，只有第一級的正反器狀態變化。即使再按

RESET 鍵，CPU 也不會再切換。也就是說，一旦進入 6809 的型態中，即使再加以 RESET，也無法切換成 6502，APPLE 完全變成 6809 機器。

因為第一級正反器附加有 CR (電阻電容電路) 方式的開機初始化 (POWER-ON initialized) 電路，因此開機後要使那邊的 CPU 先起動便可預先加以選擇。



照片 2

(上) APPLE II 時脈 (下) 6809 的 E

表 1 CPU 切換控制位址

位 址	動 作
\$C0C0 (\$A0C0)	將 6502 的切換狀態栓鎖。 第 1 級 F/F 被復置。
\$C0C1 (\$A0C1)	將 6809 的切換狀態栓鎖。 第 1 級 F/F 被設定
\$C0C2 (\$A0C2)	6502 的選擇。 與第 1 級 F/F 無關，第 2 級的 F/F 被復置。
\$C0C4 (\$A0C4)	6809 的選擇。與第 1 級的 F/F 無關，第 2 級的 F/F 被設定
\$C0C6 (\$A0C6)	CPU 的切換。第 1 級 F/F 的 狀態移至第 2 級，CPU 切換 同時發生 NMI 中斷。
RESET	CPU 切換。 第 1 級 F/F 的狀態移至第 2 級 CPU 切換。

※ () 內為由 6809 所見到的位址。

圖 5 的電路於開機後，6502 最先被起動，若將電容器反向連接，則會變成 6809 最先起動。

這種切換狀況的選擇，若以選擇開關來做是相當方便的。有關用軟體程式選擇的方法，留待後面再加說明。

■ 7 6809E卡電路圖

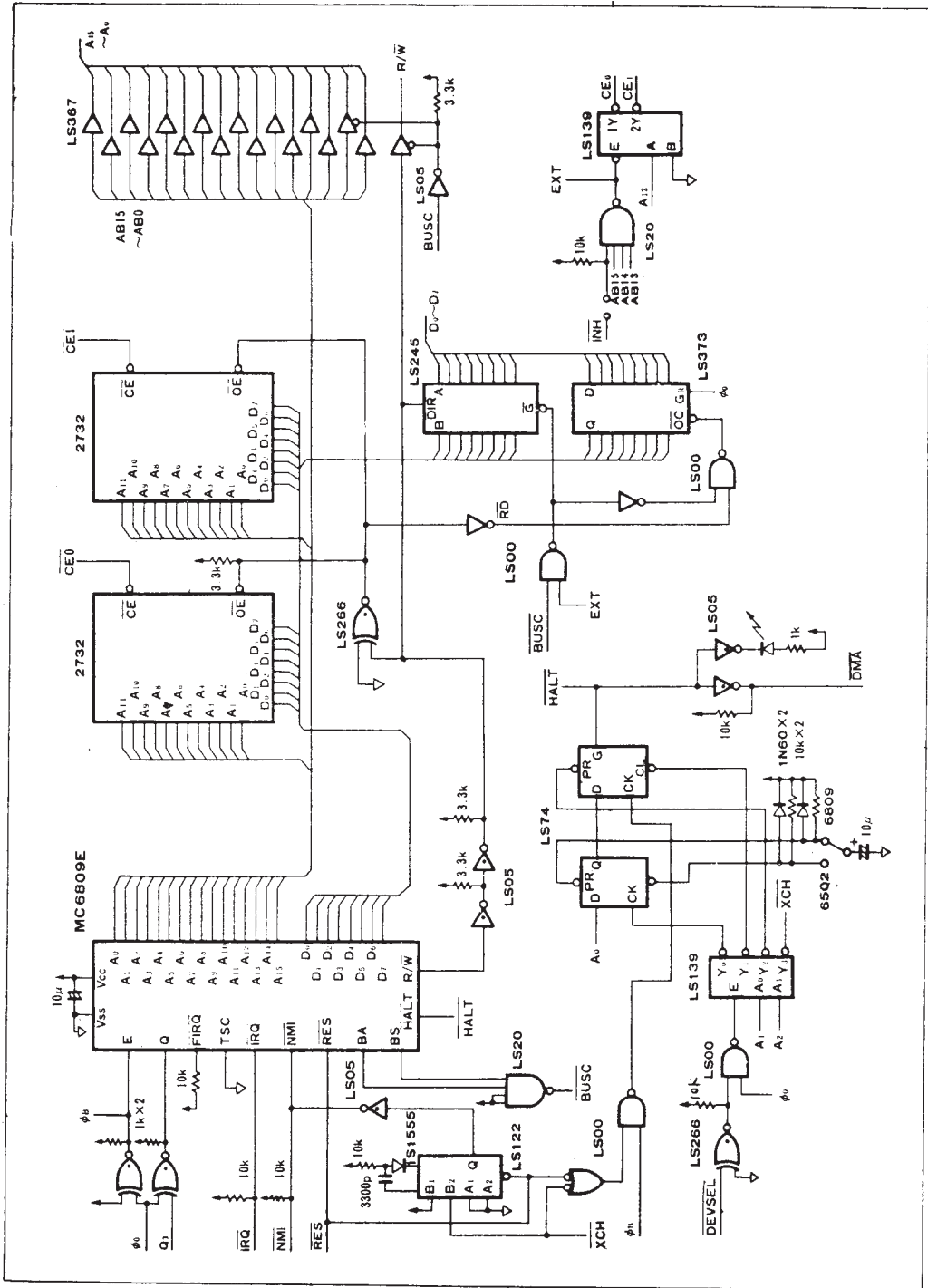
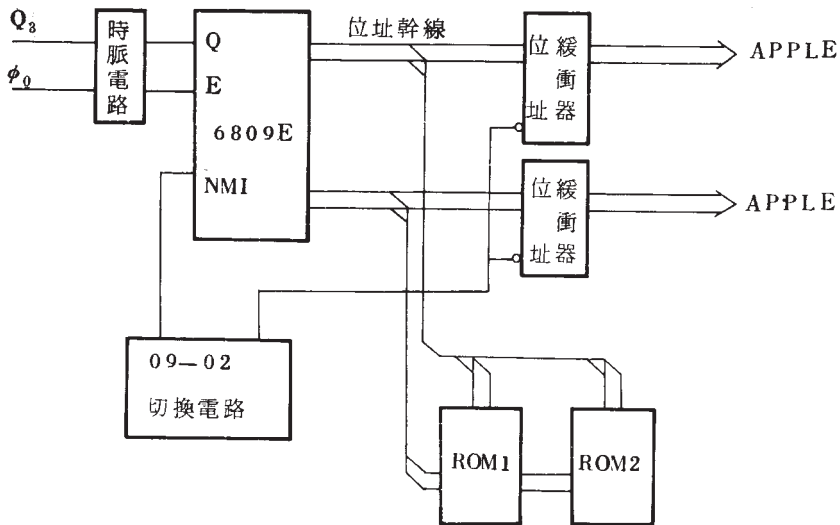


圖 8 6809 E卡方塊圖



作比 (duty) 不到 50%，6809 E 也可以充分動作。(6809 E 卡的方塊圖如圖 8 所示)

ROM 改用 2732 (使用 4 個 2716 亦可)。RAM 因為是使用 APPLE 主機上既有的，因此無需準備。

和以前電路一樣的，6502/6809 E 的切換是利用 NMI 為之，因此使用單擊多諧振盪器 (one-shot MTV) 為之。為了避免電路上所使用電容器對脈衝寬度有所影響，而產生動作上的問題起見，必須使用塑質或雲母等本性良好的電容器。

要對 APPLE 準備位址緩衝器，這在不使用內部 ROM 時，換言之，在作 APPLE 內的 RAM 存取時變成有效。資料幹線的緩衝器也同樣只在作 6809 卡外部存取時變成有效。

關於幹線的收發動作，有各種問題存在。不過使用 74LS245 大致上便得以解決。而 MC6809E 的 E 信號下降時，資料需要在資料幹線停留約 μS 以上，因此來到在 I/O 插孔上的 6809 卡時，便會有種種的延遲產生 (ϕ_0 信號的延遲特別大)，所以要以 6809 E 的 E 信號加以栓鎖。

關於這一點，即使使 E 信號更快地下降，也會有某種程度的遮蓋，因此為了簡單起見，可以將其栓鎖在幹線上作為準備。這種目的的資料幹線，使用 LS244 和 LS373 來栓鎖。

位址幹線使用 LS367。本來此處並不需要考慮 LS373，不過當考慮到 APPLE 內的各種延遲時，無論如何這種型態的系統是必要的。

關於 6502/6809 E 的切換可以前面的電路的相同方法加以考慮。使用單擊多諧振盪器，對 NMI 線送以一定寬度 (以 10K Ω 與 3300PF 決定) 的脈衝，而 NMI 係以回復軟體來進行切換。

LS74 被使用以決定開始時是由 6502 或 6809 E 何者開始起動。在第 4 插孔插入本卡時，利用 APPLE 的監督程式以 C0C1 RET RET 將 2 級的 LS74 清除，並對 6502 加上 DMA 請求，使 APPLE 變成停止狀態。

時脈產生部分

以來自外部的 ϕ_0 和 Q_3 信號作成 Q 和 E。因為開集極式 TTL 作為時脈產生電路並不理想

(假如原來的 Q 或 E 信號上升速率不良, 或者外部使用電晶體, 而希望改善上升速度時), 負載電阻要儘可能限制得小些。

在計算上可以小至 820Ω , 而作為 6809 E 的 Q, E 輸入端子輸入容量也只要 $1K\Omega$ 就相當足夠了。

由此產生器所得到的 E 信號, 在 6809 E 的控制下動作時, 中心的時序要能使在 E 下降時資料幹線上的資料是有效的。因此, APPLE 整機如前述以 ϕ 動作時, 無論如何也會有 ϕ 與 E 的相移產生, 當以 6809 的 E 信號將資料由資料幹線上取入時, 在 E 信號下降時, 全部的資料幹線會被封閉。

亦即, APPLE 內的 RAM 區域, 在資料幹線切換電路動作時, 可能會有所改變, 因而 6809 卡有錯誤動作產生。

因此, 6809 卡的資料幹線輸入, 不是以 6809 的 E 信號作為 APPLE 的時序, 而是以 ϕ 信號來作為栓鎖之用。這樣便能夠將 APPLE 與 6809 卡的相移加以吸收。

如此對於舊的 6809 卡而言, 6809 E 卡可以減少元件的數目, 並改善其不穩定。

6809 E 卡的監督程式

本監督程式的記憶容量為 2 K 位元組。(請看列表 1)

1. 動作概要

6809 卡被起動時, 要將 6502 的切換程式載入 $\$370 \sim \$3CF$, 成為命令等待狀態, 同時由於 $\$350 \sim \$36F$ 為作業區域, 所以請勿加以破壞。

不過在磁碟初次載入時 (initial load) 時, $\$350 \sim \$3CF$ 會有部份被破壞, 因此請加以注意。

CPU 的切換因為是使用 NMI 中斷, 所以需要數十 μS 左右的時間。因此, 進行過於頻繁的切換時, 效率將會降低。

各插孔的輸出入是在 APPLE 的 CIN, COUT 存入其插孔位址, 而該 CIN, COUT 是由 6809 以副程式呼叫的方式來加以實現。

DOS 副程式的執行使用 APPLE 的 COUT 副程式, 而 DOS 命令型態的切換, 可以 DOS 命令的送出使其實現。

2. CPU 的切換法

6502 及 6809 CPU 之切換步驟的流程如圖 9、圖 10 所示, 下面來看實際操作及一些命令的功能:

請將 6809 卡插入第 4 插孔。

①由 APPLE $\rightarrow$ 6809 卡的切換當 APPLE 在 BASIC 型態時, 以 `CALL-151 RET`

進入監督器型態。接著鍵入

`*C0C1 RET RET`

便切換至 6809 卡。

此後便可以使用表 2 的命令。

②利用程式作 APPLE $\rightarrow$ 6809 卡的切換由 APPLE 的 BASIC 型態, 執行

`POKE-16191,0`

`POKE-16186,0`

便切換至 6809 卡。此後使用表 2 的命令。

③由 APPLE 呼叫 6809 的副程式

每次 6809 卡起動後, 便返回 APPLE

的控制。此後, 在 $\$368, \369 位址分

別設定 6809 副程式的上位與下位位址

, 而在 $\$370(880_{10})$ 作副程式呼叫。

此時, A, X, Y 暫存器被移交給 6809

, 如果 6809 的副程式使得 A 暫存器與

X, Y 暫存器的下位 8 位元發生變化,

由 APPLE 接受。

④由 6809 呼叫 APPLE 的副程式

在 U 暫存器設定 6502 的副程式位址,

而 $\$FAEE$ 作為副程式呼叫之用。

此時, A 暫存器以及 X, Y 暫存器的下

位 8 位元被移交給 6502, 若因為 6502

的副程式而使得 A, X, Y 暫存器發生

變化, 由 6809 接受。

3. 命令的使用法

6809 卡被起動後, 以 # 表示在命令的等待狀態。

命令係以下列的型式表示。

命令 **參數** **RET**

命令名稱爲1個字元，參數以“，”分隔，空白可以忽略。

●特殊鍵

←退回一格。

→將通過的內容輸入。

CTRL S.....表示停止或再開始。

RET輸入結束。

命令的種類與功能請參考表2。一直到畫面末端爲止所顯示的內容，每1行作一次翻滾(Scroll)。

表2 命令一覽表

命令	參數	功能	意義
C	X, Y	由X至Y，每次輸出8位元組的Checksum。	Checksum
D	<X, Y>	由X至Y，以16進數與ASCII輸出。省略Y時，則由X輸出80位元組。	Dump
F	X, Y, Z	由X至Y，填入Z。	Fill
G	X	跳接至X。	Go
H	X {+} Y	進行並表示16進數加減算。	Hexadecimal
I	X	將第X插孔當作輸入並打開。	Input
J	X	將X作為副程式呼叫。	Jumpsub
M	X	將由X起的位址與內容加以表示及變更。以“，”區分時可連續地變更。不變更時可以“，”跳過。而後以RET表示欠一位址及其內容。此時16進數以外的輸入結束。	Memory Change
P	< X >	將第X插孔當作輸出並打開。	output
R	X, Y, Z	將由X至Y的內容移至Z起以後。	Relocate
T	< X >	將由X起的(不包含相對位址)3位元組命令變成追蹤(trace)型態。以後每當通過此位址時，在暫存器的內容被輸出後，該命令便被執行。X被省略時，追蹤型態停止，回到原來的命令。	Trace
V	X-Y-Z	將X至Y的內容與由Z起的内容互相比較，不同時則表示其個別的位址與內容。	Verify
X		由6809切換成6502，並跳接至X位址。X省略時，將APPLE的監督程式起動。	eXchange
&	DOS命令	接著&之後執行DOS 3.3。	

< >...表示可以省略。{ }...表示任意擇一。X, Y, Z...表示16進數。

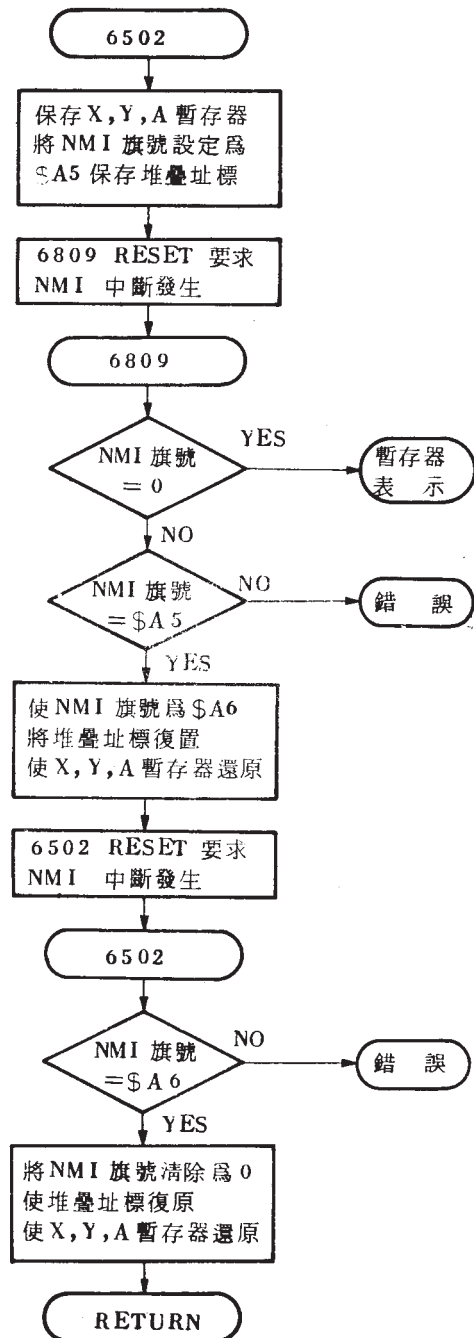


圖9 6502 → 6809 切換步驟

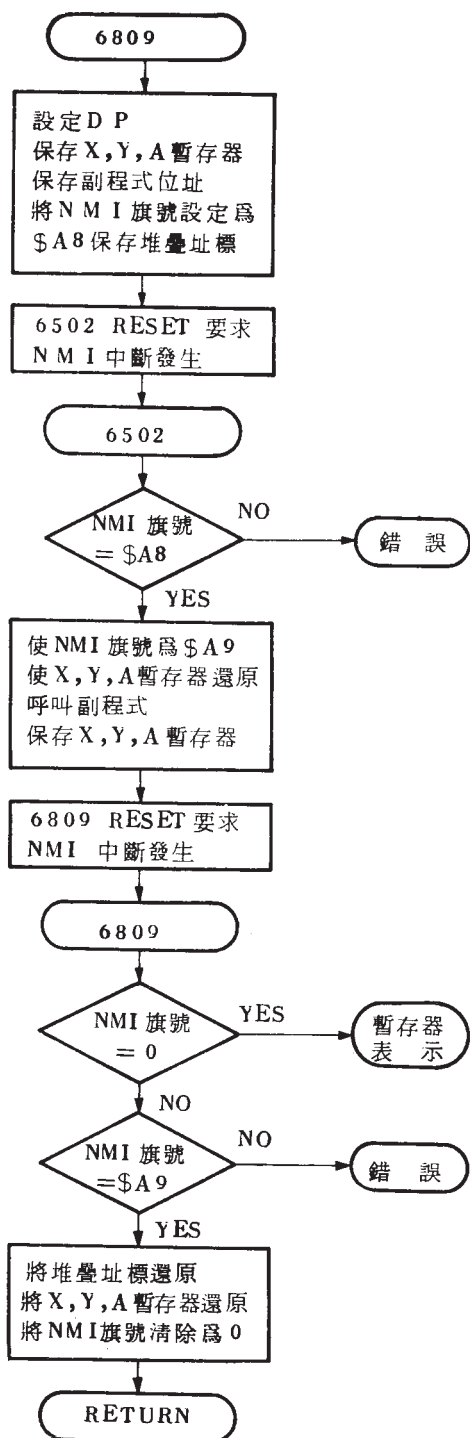


圖 10 6809 → 6502 切換步驟

4. 中斷

6809 卡的中斷功能與處理如下所示：

中 斷	位 址	功 能
RESET	\$F800	將 6809 起動，進入 6809 卡。
NMI	\$F802	進入切換程式。
FIRQ	\$3C6	堆疊址標，暫存器的表示。
IRQ	\$3C9	"
SWI	\$3CC	"
SWI2	\$3C3	"
SWI3	\$3C0	"

\$3C0～\$3CE 的位址，存入跳接指令與中斷的預備跳接位址。可將其改變，以作成自己所喜好的處理方式。

6809 卡監督程式的進入點配置與作業用變數表

6809 卡的監督程式進入點配置表 (monitor Entry Map) 如下表所示。

WORK AREA			
0200	CLI	EQU	\$200 ; INPUT BUFFER
0250	ERR02	EQU	\$250 ; ERROR RETURN AD FOR 6502
0352	OUTAD	EQU	\$352 ; OUTPUT ROUTINE AD
0354	INAD	EQU	\$354 ; INPUT ROUTINE AD
0356	ERR09	EQU	\$356 ; ERROR RETURN AD FOR 6809
0358	STDP	EQU	\$358 ; STACK POINTER SAVE
035A	WORK	EQU	\$35A ; WORK AREA
035C	RVECT	EQU	\$35C ; RESET VECTOR AD
035E	TWK	EQU	\$35E ; TRACE WORK
0364	PW3	EQU	\$364 ; POWER UP BYTE
0365	NMIFLG	EQU	\$365 ; NMI FLAG
0366	SAD02	EQU	\$366 ; SUBROUTINE AD FOR 6502
0368	SAD09	EQU	\$368 ; SUBROUTINE AD FOR 6809
036E	RAD09	EQU	\$36E ; RETURN AD FOR 6809

表 4 作業用變數表

表 3 監督程式進入點配置表

標名	位址	功 能	CC	A	B	DP	X	Y	U
START	F800	監督程式的起點							
NMI	F802	將 \$305 的內容寫入 \$A5, \$A9 以外，則跳至 \$305 內零的位址。							
NMIM	F81E	將 NMI 的訊息輸出，並且顯示暫存器內容，同時成爲待命狀態。							
FIRO	F824	輸出 FIRO 的訊息，以便與 NMIM 間。							
IRQ	F82A	輸出 IRQ 的訊息，以便與 NMIM 間。							
SWI	F830	在 \$358 中保存堆疊位址，輸出 SWI 的訊息，以便與 NMIM 間。							
SWI2	F83A	輸出 SWI2 的訊息，以便與 NMIM 間。							
SWI3	F840	輸出 SWI3 的訊息，以便與 NMIM 間。							
CAL02	F84E	將以 U 所表示的 \$502 的訊息，U 以位址 H，L 賦與。							
DOS	F84C	輸出由 X 所表示的位址起至 \$0D (CPU 爲止) 輸入的 DOS 命令。							
COUT02	F85D	將 A 的內容輸出至 \$502 的 COUT (\$FDD)。							
RDKY02	F86C	由 \$502 的 RDKY (\$5AE) 輸入 A。							
POPEN	F87A	將以 B 所表示的 (\$5C) 第 n 種孔將由 \$502 內輸出打開。							
IOPEN	F89E	將以 B 所表示的 (\$5C) 第 n 種孔將由 \$502 內輸入打開。							
HCKCV	FCF7	檢查 A 的內容是否爲 16 進位 ASCII 碼，若是 16 進位的數據，位的 2 進位，若否 16 進位，假定爲 16 進位。							
HEXCNV	FD1B	由 X 所表示的位址起，將 ASCII 碼的資料轉換爲 2 進位，直到 16 進位以外的資料進來為止。							
MSGOUT1	F03E	MSGOUT 前輸出 CR, LF。							
MSGOUT	FD40	將由 X 所表示位址起至 EOT (\$04) 到來為止的內容輸出畫面，而使 X 表示下一位址。							
OUT4HS	FD4F	進行 OUT4H，輸出空格。							
OUT4H	FD57	由 X 所表示位址起將 2 位元組以 16 進位 4 位數輸出，而使 X 表示下一位址。							
OUT2HS	FD53	進行 OUT2H，輸出空格。							
OUT2H	FD59	由 X 所表示位址起將 1 位元組以 16 進位 2 位數輸出，而使 X 表示下一位址。							
OUTXS	FD61	將 X 的內容以 16 進位 4 位數輸出且輸出空格。							
OUTDS	FD69	進行 OUTD，輸出空格。							
OUTD	FD6D	將 D 的內容以 16 進位 4 位數輸出。							
OUTAS	FD79	進行 OUTA，輸出空格。							
OUTA	FD83	將 A 的內容以 16 進位 2 位數輸出。							
BINHEX	FD93	將 A 的下位 4 位元轉換爲 16 進位 ASCII 碼。							
OUTSP	FDA0	輸出空格。							
CRLF	FDA8	輸出 CR, LF。							
GETIN	FD87	由鍵盤輸入資料至 CR (\$0D) 到來為止，並存在 X 所表示的輸出緩衝器中，可讀或可寫。							
OUTEE	FDE2	跳至 \$352 的內容位址，通常跳至 OUTA1。							
INEE	FDEA	跳至 \$354 的內容位址，通常跳至 INAI。							
INIT5	FDEE	將 V-RAM 設定在內又新 1 頁，且將圖的範圍設定爲最大 40,24。							
WINDOW	FD03	設定畫面輸出，將畫面上限存入 A，下限存入 B，而後開始。							
CLR CRT	FD0E	將畫面的範圍內清除。							
CLEAR	FE14	將目前畫面以後的畫面清除。							
INAI	FEA0	從鍵盤讀身，由鍵盤讀 1 個字元輸入 A。							
INKEY	FE2B	若有鍵盤按下則輸入 A，若無鍵盤下則設定爲位址 0。							
OUTA1	FECB	將 A 的內容以 1 個字元表示，CR 輸出時，若 CTRL (\$13) 被按下，則等到再有鍵盤按下時才輸出。							
BELL	FF60	將 1.3KHz 警報聲輸出 0.2 秒。							
BEL1	FF67	在 A 的範圍，鳴。時間的範圍。							
CAL09	30370	由 \$502 呼叫 \$800 的制程式，制程式位址從 H，L 的順序放入 \$308, \$309 中。							
NET00	03A5	由 \$800 呼叫 \$502 的制程式，最後制程式返回 \$800 進入點。							

(C = 保存, X = 讀取, I = 隨意讀或寫)

列表 1 監督程式

#DFB00,FFFF

```

F800 20 50 7D 03 65 26 04 6E : P) e& n
F808 9F 03 56 B6 03 65 B1 A9 : V e
F810 10 27 02 FE B1 A5 10 27 : '
F818 03 03 6E 9F 03 56 30 8D : n VO
F820 04 8A 20 20 30 8D 04 8A : O
F828 20 1A 30 8D 04 8B 20 14 : O
F830 10 FF 03 58 30 8D 04 87 : X0
F838 20 0A 30 8D 04 87 20 04 : O
F840 30 8D 04 8B 17 04 8C 17 : O
F848 04 F6 17 05 5B 17 04 16 : [
F850 20 60 10 CE 02 FF 17 04 : '
F858 7A 17 04 7B 30 8D 06 6D : z {O m
F860 BF 03 52 30 8D 06 39 BF : R0 9
F868 03 54 8E 65 FF BF 03 50 : T e P
F870 BE F8 00 BF 03 56 7D C0 : V)
F878 58 7D C0 5A 7D C0 5D 7D : X) Z) J)
F880 C0 5F 7D CF FF 7D C0 10 : _ ) }
F888 BE C0 81 B6 03 5C 8B A5 : ¥
F890 B1 03 64 26 04 6E 9F 03 : d& n
F898 5C 30 8D 00 15 BF 03 5C : ¥0 ¥
F8A0 B6 03 5C 8B A5 B7 03 64 : ¥ d
F8A8 17 05 3F 30 8D 00 40 17 : ?0 @
F8B0 04 8E 10 CE 02 FF 17 04 :
F8B8 EF B6 23 17 05 24 17 06 : # $
F8C0 9F 17 04 F0 17 04 E1 31 : 1
F8C8 BD 00 30 A6 B4 A1 A4 27 : O '
F8D0 11 31 23 A6 A4 AA A4 26 : 1# &
F8D8 F2 30 8D 00 1C 17 04 60 : O '
F8E0 20 D0 33 8C CD 34 40 30 : 3 430
F8E8 01 31 21 EC A4 6E AB 54 : 1! n T
F8F0 4F 4D 41 54 4F 2D 39 45 : OMATO-9E
F8F8 04 3F 04 47 00 30 44 00 : ? B OD
F900 45 4D 01 87 4A 00 34 54 : EM J 4T
F908 03 01 56 01 5B 52 01 1E : V XR
F910 5B 01 CD 26 02 38 00 02 : X & BP
F918 61 49 02 82 48 02 AC 43 : aI F H C
F920 00 97 46 01 2C 00 30 8C : F , O
F928 D0 16 04 14 17 03 BC 10 :
F930 BC 00 00 27 F1 32 62 6E : ' 2bn
F938 A4 17 03 AF 10 8C 00 00 :
F940 27 E4 6E A4 A6 B4 81 0D : ' n
F948 27 12 17 03 9E B1 2D 26 : ' ,&
F950 10 1F 23 17 03 95 B1 0D : #
F958 26 C0 20 0E FE 03 5A 20 : & Z
F960 06 B1 0D 26 C1 1F 23 31 : & #1
F968 C8 4F 10 BF 03 5A 1F 31 : O Z 1
F970 17 03 EE C6 08 34 10 A6 : 4,
F978 B4 17 03 FD 8C 03 5A 26 : 2&
F980 0C C2 09 53 5C 35 10 8D : S#5
F988 14 BF 03 5A 39 30 01 5A : Z90 Z
F990 26 E5 C6 08 35 10 8D 05 : & 5
F998 17 04 0D 20 D3 86 3A 17 : :
F9A0 04 40 A6 B0 81 20 24 04 : @ $
F9A8 B6 20 20 04 81 7F 22 F8 : \"
F9B0 17 04 2F 5A 26 EC 39 17 : /Z 9
F9B8 03 31 81 20 10 26 FF 66 : 1 , & f
F9C0 1F 23 17 03 26 81 0D 10 : # &
F9C8 26 FF 5B 10 BF 03 5A 1F : & [ Z
F9D0 31 17 03 D4 17 03 8A B6 : 1
F9D8 08 C6 00 EB 84 8C 03 5A :
F9E0 27 0B 30 01 4A 26 F4 8D : ' O J&
F9E8 02 20 E6 30 1F 86 2D 17 : O -
F9F0 03 F0 17 03 AB 17 03 69 : 1
F9F8 B6 3D 17 03 E5 17 03 A0 : =
FA00 1E 89 17 03 7E 30 01 39 : ~O 9
FA08 17 02 E0 B1 2C 10 26 01 : , &
FA10 F3 1F 23 17 02 D5 B1 2C : #
FA18 10 26 01 EB 10 BF 03 5A : & Z
FA20 17 02 CB B1 0D 10 26 01 : &

```

FA2B DB 1F 31 39 BD DA 8C 03 : 19
FA30 5A 22 D4 A6 80 A7 A4 A1 : Z"
FA3B A0 27 F3 31 3F 30 BD 00 : ' 1?0
FA40 0B 17 02 FA 1F 20 16 03 :
FA4B 24 3C 45 52 3E 3D 04 BD : \$<ER>=
FA50 B7 1F 20 1F 12 10 BC 03 :
FA5B 5A 22 AC E7 A4 E1 A0 26 : Z" &
FA60 DA 20 F2 8D A3 BC 03 5A : Z
FA6B 22 9D A6 84 A1 A4 27 13 : "
FA70 17 02 EE 17 03 03 1F 20 :
FA7B 17 02 EE A6 A4 17 02 F9 :
FA80 17 03 25 30 01 31 21 20 : %0 1!
FAB8 DC 17 02 5F 81 0D 20 26 : &
FA90 FE 94 1F 23 20 03 17 03 : #
FA9B 0F 1F 30 17 02 CB 1F 31 : 0 1
FAA0 17 02 B0 17 03 0E B6 02 :
FAAB 00 B1 0D 27 1D 17 02 6B : ' k
FAB0 B1 2C 27 06 B1 0D 27 0E : ' ,
FAB8 20 1C A1 1E 27 04 BD 0E : ' ,
FAC0 20 EB 33 41 20 E7 BD 06 : 3A
FACB 20 CC 33 41 20 CB 1F 20 : 3A
FAD0 E7 C4 E1 C0 26 01 39 1F : & 9
FADB 32 32 62 16 FF 5D 17 02 : 22b J
FAEQ 0A 10 BC 00 00 26 05 CE : &
FAEB FF 65 20 02 1F 23 34 4D : e #4M
FAF0 17 01 E0 8D 40 1F 30 1E : a 0
FAFB B9 FD 03 66 B6 4C B7 03 : f L
FB00 FB CC 7F 03 FD 03 FC B6 : \
FB0B AB B7 03 65 10 FF 03 5B : e x
FB10 20 1A 10 FE 03 5B 8D 2B : x (
FB18 7F 03 65 35 CD 7C 03 65 : \ e5 : e
FB20 10 CE 02 FF BD 1A AD 9F :
FB2B 03 68 BD 09 7D C0 C0 7D : h) >
FB30 C0 C6 16 FD 1D 97 45 1F : E
FB3B 10 D7 46 1F 20 D7 47 39 : F G9
FB40 4F D6 47 1F 02 D6 46 1F : 0 G F
FB4B 01 96 45 39 86 0D BD 0D : E9
FB50 86 04 BD 09 A6 80 BD 05 :
FB5B 81 0D 26 FB 39 34 72 B1 : & 94r
FB60 0A 27 07 BA 80 CE FD ED : '
FB6B 8D 84 35 F2 34 40 CE FD : 5 4Q
FB70 0C 17 FF 7A 84 7F 35 C0 : z \5
FB7B 8D 3A C1 C0 27 05 30 8C : : ' 0
FB80 DC 20 07 CC F0 FD 30 BD : 0
FB8B 03 43 BF 03 52 DD 36 CE : C R 6
FB90 AB 51 AE C4 8C A5 39 10 : Q 9
FB9B 27 FF 53 39 BD 16 C1 C0 : ' S9
FBA0 27 05 30 8C 07 20 07 CC : ' 0
FBA8 1B FD 30 8D 02 F2 BF 03 : 0
FBB0 54 DD 3B 39 17 01 64 1F : T 89 d
FBBB 20 C1 00 25 07 C1 08 24 : % \$
FBC0 03 CB C0 39 32 62 16 FD : 92b
FBCB 5D 17 01 4F 1F 23 B1 2B : J 0 # +
FBD0 27 0D B1 2D 10 26 FD 4E : ' - &. N
FBD8 BD 1C B3 03 5A 20 05 BD : Z
FBE0 15 F3 03 5A 34 06 B6 3D : Z4 =
FBE8 17 01 F7 35 06 17 01 93 : 5
BF0 F1 E8 17 01 BE 39 17 01 : 9
FBFB 22 B1 0D 26 07 10 BF 03 : " &
FC00 5A 1F 30 39 32 62 16 FD : Z 092b
FC0B 1D A6 84 B1 0D 27 27 17 :
FC10 00 D9 B1 0D 10 26 FD 0E : &
FC1B A6 A4 B7 03 5E AE 21 BF : ^ !
FC20 03 5F B6 39 B7 03 61 10 : - 9 a
FC2B BF 03 62 B6 BD A7 A0 30 : b 0
FC30 8D 00 12 AF A4 39 10 BE : 9
FC3B 03 62 B6 03 5E A7 A0 BE : b ^
FC40 03 5F AF A4 39 34 7F 30 : - 94 \0
FC4B BD 00 13 17 00 F2 FC 03 :
FC50 62 17 01 19 17 01 51 BD : b Q
FC5B 0D 35 7F 7E 03 5E 0D 0A : 5 \ ^
FC60 3C 54 52 3E 3D 04 30 BD : <TR>= 0
FC6B 00 1D 17 00 D3 1F 41 30 : A0
FC70 02 1F 10 17 00 F3 C6 04 :
FC7B 17 00 DB 5A 26 FA C6 04 : Z&

FC80 17 00 CC 5A 26 FA 39 53 : Z& 9S
FC8B 50 20 20 3A 43 43 20 41 : P : CC A
FC90 20 20 42 20 20 44 50 20 : B DP
FC9B 5B 20 20 20 59 20 20 : X Y
FCA0 20 20 55 20 20 20 50 : U P
FCAB 43 0D 0A 04 2A 4E 4D 49 : C #NMI
FCB0 2A 04 2A 46 49 52 51 2A : # \$FIRQ#
FCB8 04 2A 49 52 51 2A 04 2A : \$IRQ# \$
FCC0 53 57 49 2A 04 2A 53 57 : \$SWI# \$SW
FCCB 49 32 2A 04 2A 53 57 49 : I2# \$SWI
FCD0 33 2A 04 5F 1F 9B 39 34 : 3# - 94
FCD8 57 30 BD 02 B6 CE 03 70 : W0 - p
FCE0 C6 5F A6 B0 A7 C0 5A 26 : Z&
FCEB F9 35 D7 A6 B4 B1 20 27 : 5 '
FCF0 02 20 28 30 01 20 F4 BD : (0
FCFB 0D 25 0A B0 30 B1 09 23 : % 0 #
FD00 02 B0 07 1C FE 39 BD 0C : 9
FD0B 24 06 B1 47 24 03 B1 41 : \$ G\$ A
FD10 39 1A 01 39 B1 3A 24 F9 : 9 9 : \$
FD1B 81 30 39 10 BE 00 00 A6 : 09
FD20 80 B1 20 27 FA BD D0 25 : %
FD2B F1 34 02 1F 20 BD 06 EA : 4
FD30 E0 1F 02 20 EA 5B 49 5B : XIX
FD3B 49 58 49 58 49 39 BD 68 : IXIXI9 h
FD40 34 03 A6 B0 B1 04 26 02 : 4 &
FD4B 35 B3 17 00 95 20 F3 BD : 5
FD50 06 20 4D BD 04 20 49 BD : M I
FD5B 00 34 03 A6 B0 BD 24 35 : 4 \$5
FD60 83 34 17 EC 63 BD 02 35 : 4 c 5
FD6B 97 BD 02 20 33 34 07 A6 : 34
FD70 61 BD 10 A6 62 BD 0C 35 : a b 5
FD7B 87 BD 0B 34 03 86 20 BD : 4
FDB0 61 35 B3 34 03 44 44 44 : a5 4 DDD
FDBB 44 BD 0A BD 55 A6 61 BD : D U a
FD90 04 BD 4F 35 B3 B4 0F B8 : 05
FD9B 30 B1 3A 25 02 B8 07 39 : 0 : % 9
FDA0 34 03 B6 20 BD 3C 35 B3 : 4 <5
FDB8 34 33 B6 0D BD 34 B6 0A : 43 4
FDB0 BD 30 35 B3 BE 02 00 34 : 05 4
FDBB 14 C6 30 BD 29 BD 23 B1 : 0) #
FDC0 0B 26 09 C1 30 27 F4 5C : & 0' #
FDCB 30 1F 20 EF B1 15 26 05 : 0 &
FDD0 30 01 5A 20 E6 A7 B4 B1 : 0 Z
FDD8 0D 27 05 30 01 5A 26 DB : ' 0 Z&
FDE0 35 94 6E 9F 03 52 6E 9F : 5 n Rn
FDEB 03 54 BD 02 20 20 4F 5F : T 0_
FDF0 DD 24 7D C0 56 7D C0 54 : \$) V) T
FDFB 7D C0 51 0F 20 C6 2B D7 :) 0 (
FE00 21 C6 18 34 47 97 22 D7 : ! 4G "
FE0B 23 E0 61 5A 35 C7 BD 9B : # aZ5
FE10 34 07 BD 2E DC 24 34 06 : 4 . \$4
FE1B BD 17 96 20 97 24 D6 25 : \$ %
FE20 5C D1 23 24 06 D7 25 BD : # # \$ %
FE2B 0B 20 F3 35 06 DD 24 35 : 5 \$5
FE30 87 34 17 BD 17 D6 24 B6 : 4 \$
FE3B A0 A7 B0 5C D1 21 26 F9 : # !&
FE40 35 97 34 06 96 20 D6 22 : 5 4 "
FE4B DD 24 35 B6 34 07 96 25 : \$5 4 %
FE50 44 B4 03 B4 04 34 02 96 : D 4
FE5B 25 B4 18 24 02 B9 7F 34 : % \$ \4
FE60 02 4B 4B A4 E0 E6 E0 1E : HH
FE6B 89 DB 24 B8 00 1F 01 35 : \$ 5
FE70 87 96 20 D6 22 DD 24 BD : " \$
FE7B D3 1F 12 D6 25 5C D1 23 : % # #
FE80 24 15 D7 25 BD C6 D6 24 : \$ % \$
FE8B A6 B0 A7 A0 5C D1 21 26 : # !&
FE90 F7 D6 20 D7 24 20 E0 96 : \$
FE9B 20 D6 23 5A DD 24 20 91 : # Z \$
FEA0 34 1C 17 FE 2E BD A5 E6 : 4
FEAB B4 34 04 C4 3F CA 40 E7 : 4 ? @
FEB0 B4 BD 0B 24 FC 35 04 E7 : \$ 5
FEBB B4 35 9C 34 14 B6 C0 00 : 5 4
FEC0 2A 0B 7D C0 10 B4 7F 1A : \$) \
FECB 01 21 5F 35 94 34 3F 17 : ! 5 4?
FEDO FE 01 B1 20 24 04 BD 1E : \$

```

FED8 20 1A 17 FF 6F 8A 80 A7 : 0
FEE0 84 DC 24 4C 91 21 25 05 : $L !%
FEE8 96 20 5C D1 23 DD 24 25 : # # %%
FEF0 03 17 FF 7D 35 BF 81 0A : )5
FEF8 27 27 81 0D 27 0A 81 08 : ' ' '
FF00 27 30 81 15 27 41 20 58 : 'O 'A X
FF08 D6 20 B6 C0 00 2A 0F 81 : *
FF10 93 26 08 7D C0 10 B6 C0 : & )
FF18 00 2A FB 7D C0 10 D7 24 : * ) %
FF20 39 DC 24 5C D1 23 25 07 : 9 $% #%
FF28 34 02 17 FF 44 35 02 DD : 4 D5
FF30 24 39 D6 24 D1 20 23 03 : $9 $ #
FF38 0A 24 39 D6 25 D1 22 23 : $9 % " #
FF40 1E 0A 25 D6 21 20 D7 D6 : % !
FF48 24 5C D7 24 D1 21 25 0F : $% $ !%
FF50 D6 20 D7 24 D6 25 5C D1 : $ %%
FF58 23 25 02 D6 22 D7 25 39 : %% " %9
FF60 34 16 86 96 8E 01 2C 1F : 4
FF68 B9 7D C0 30 5A 26 FD 30 : ) 0Z& 0

```

```

FF70 1F 26 F4 35 96 00 00 00 : & 5
FF78 00 00 00 00 00 00 00 00 :
FF80 00 00 00 00 00 00 00 00 :
FF88 00 00 00 00 00 00 00 00 :
FF90 00 00 00 20 4A FF A9 A5 : J
FF98 BD 65 03 BA 8E CF 03 4C : e L
FFA0 A5 03 AD 65 03 C9 A6 D0 : e
FFAB 0C A9 00 8D 65 03 AE CF : e
FFB0 03 9A 4C 3F FF C9 AB F0 : L?
FFB8 03 6C 50 03 EE 65 03 20 : 1P e
FFC0 3F FF 20 AE 03 20 4A FF : ? J
FFC8 AD C1 C0 AD C6 C0 4C 59 : LY
FFD0 FF 6C 66 03 00 00 00 00 : 1f
FFD8 00 00 00 00 00 00 00 00 :
FFE0 00 00 00 7E F8 40 7E F8 : ~ @~
FFEB 3A 7E F8 24 7E F8 2A 7E : 1~ $~ %~
FFF0 FB 30 03 C0 03 C3 03 C6 : 0
FFF8 03 C9 03 CC FB 02 FB 00 :

```

8 位元組檢查和 (checksum) 表

*CF800,FFFF

F800 - F807 = ED	F9A0 - F9A7 = 33	FB58 - FB5F = 0C	FD10 - FD17 = 65	FEC8 - FECF = D4
F808 - F80F = 40	F9AB - F9AF = E4	FB60 - FB67 = FA	FD18 - FD1F = 2E	FED0 - FED7 = 73
F810 - F817 = 94	F9B0 - F9B7 = 06	FB68 - FB6F = 77	FD20 - FD27 = C4	FED8 - FEDF = 70
F818 - F81F = 59	F9B8 - F9BF = 7C	FB70 - FB77 = 94	FD28 - FD2F = E3	FEEO - FEEF = AC
F820 - F827 = 19	F9C0 - F9CF = 20	FB78 - FB7F = 30	FD30 - FD37 = 04	FEF8 - FEFF = 2C
F828 - F82F = BA	F9CB - F9CF = CB	FB80 - FB87 = 79	FD38 - FD3F = 89	FEF0 - FEF7 = 15
F830 - F837 = B2	F9D0 - F9DF = 49	FB88 - FB8F = 3B	FD40 - FD47 = 0A	FEF8 - FEFF = 96
F838 - F83F = 96	F9DD - F9DF = 56	FB90 - FB97 = E5	FD48 - FD4F = 04	FF00 - FF07 = CD
F840 - F847 = 07	F9E0 - F9EF = 54	FB98 - FB9F = D6	FD50 - FD57 = FA	FF08 - FF0F = 26
F848 - F84F = A2	F9EB - F9EF = 21	FBA0 - FBA7 = A2	FD58 - FD5F = 43	FF10 - FF17 = 87
F850 - F857 = 7A	F9F0 - F9FF = 3B	FBA8 - FBAF = 8B	FD60 - FD67 = E1	FF18 - FF1F = 6D
F858 - F85F = 40	F9FB - F9FF = 7C	FBB0 - FBB7 = 3D	FD68 - FD6F = 5A	FF20 - FF27 = B5
F860 - F867 = CF	FA00 - FA0F = A9	FBB8 - FBBF = FA	FD70 - FD77 = D4	FF28 - FF2F = A4
F868 - F86F = 5B	FA08 - FA0F = DD	FBC0 - FBC7 = 6E	FD78 - FD7F = 86	FF30 - FF37 = 6E
F870 - F877 = DB	FA10 - FA17 = D0	FBC8 - FBCF = B2	FD80 - FD87 = 1C	FF38 - FF3F = 78
F878 - F87F = 06	FA18 - FA1F = 4B	FBD0 - FBD7 = 63	FD88 - FD8F = 51	FF40 - FF47 = 11
F880 - F887 = B7	FA20 - FA27 = A6	FBD8 - FBDF = 6B	FD90 - FD97 = B6	FF48 - FF4F = A1
F888 - F88F = 41	FA28 - FA2F = 8A	FBE0 - FBE7 = 62	FD98 - FD9F = DD	FF50 - FF57 = 19
F890 - F897 = 52	FA30 - FA37 = 62	FBE8 - FBEF = F5	FDA0 - FDA7 = 5E	FF58 - FF5F = 77
F898 - F89F = 4C	FA38 - FA3F = E7	FBF0 - FBF7 = 9E	FDA8 - FDAF = 4B	FF60 - FF67 = 40
F8A0 - F8AF = 60	FA40 - FA47 = 73	FBF8 - FBFF = AF	FDB0 - FDB7 = 69	FF68 - FF6F = A3
F8AB - F8AF = 6F	FA48 - FA4F = 03	FC00 - FC07 = 89	FDB8 - FDBF = F1	FF70 - FF77 = 04
F8B0 - F8B7 = BC	FA50 - FA5F = 5A	FC08 - FC0F = 3A	FDC0 - FDC7 = 9F	FF78 - FF7F = 00
F8B8 - F8BF = F5	FA60 - FA67 = 35	FC10 - FC17 = A8	FDC8 - FDCF = 1F	FF80 - FF87 = 00
F8C0 - F8CF = D7	FA68 - FA6F = 68	FC18 - FC1F = F0	FDD0 - FDD7 = 3D	FF88 - FF8F = 00
F8C8 - F8CF = 53	FA70 - FA77 = 63	FC20 - FC27 = 4C	FDD8 - FDDF = C5	FF90 - FF97 = B7
F8D0 - F8DF = 23	FA78 - FA7F = 63	FC28 - FC2F = DE	FDE0 - FDE7 = 3B	FF98 - FF9F = 5B
F8D8 - F8DF = 46	FA80 - FA87 = E2	FC30 - FC37 = F9	FDE8 - FDEF = D4	FFA0 - FFA7 = FC
F8E0 - F8EF = 20	FA88 - FA8F = 1B	FC38 - FC3F = 81	FE00 - FED7 = 25	FFA8 - FFAF = 27
F8E8 - F8EF = 50	FA90 - FA97 = 11	FC48 - FC4F = A8	FE08 - FE0F = DF	FFB0 - FFB7 = 88
F8F0 - F8FF = 2B	FA98 - FA9F = 92	FC50 - FC57 = 89	FE10 - FE17 = 30	FFB8 - FFBF = 38
F8F8 - F8FF = 02	FAA0 - FAA7 = A9	FC58 - FC5F = B7	FE18 - FE1F = 10	FFC0 - FFC7 = 78
F900 - F90F = EC	FAAB - FAAF = 56	FC60 - FC67 = 1E	FE20 - FE27 = 03	FFC8 - FFCF = 06
F908 - F90F = 24	FAAC - FAAC = 51	FC68 - FC6F = 97	FE28 - FE2F = 8C	FFD0 - FFD7 = D4
F910 - F917 = DB	FAD0 - FADF = 51	FC70 - FC77 = 05	FE30 - FE37 = F6	FFD8 - FFD7 = 00
F918 - F91F = 67	FAE0 - FAE7 = 9F	FC78 - FC7F = 33	FE38 - FE3F = 34	FFE0 - FFE7 = 2C
F920 - F927 = C6	FAEB - FAEF = 49	FC80 - FC87 = E9	FE40 - FE47 = B4	FFE8 - FFEF = F2
F928 - F92F = E4	FAF0 - FAF7 = 32	FC88 - FC8F = R1	FE48 - FE4F = B2	FFF0 - FFF7 = 7A
F930 - F937 = A6	FAFB - FAFF = 7B	FC90 - FC97 = 76	FE50 - FE57 = 25	FFF8 - FFFF = 8D
F938 - F93F = 09	FAB0 - FAB7 = CB	FC98 - FC9F = 71	FE58 - FE5F = 23	
F940 - F947 = D5	FAB8 - FBAF = 31	FCA0 - FCA7 = 65	FE60 - FE67 = 00	
F948 - F94F = C4	FAB8 - FBAF = 31	FCA8 - FCAF = 6C	FE68 - FE6F = 68	
F950 - F957 = 8F	FAB0 - FBAF = 31	FCB0 - FCB7 = B4	FE70 - FE77 = C3	
F958 - F95F = 9B	FAB8 - FBAF = 31	FCC0 - FCC7 = F5	FE78 - FE7F = 4F	
F960 - F967 = EE	FAB8 - FBAF = 31	FCC8 - FCCF = C6	FE80 - FE87 = 82	
F968 - F96F = 93	FAB8 - FBAF = 31	FCD0 - FCD7 = E7	FE88 - FE8F = E1	
F970 - F977 = C0	FAB8 - FBAF = 31	FCD8 - FCD7 = OD	FE90 - FE97 = 7E	
F978 - F97F = DA	FAB8 - FBAF = 31	FCE0 - FCE7 = 32	FE98 - FE9F = 25	
F980 - F987 = 5B	FAB8 - FBAF = 31	FCE8 - FCEF = F7	FEA0 - FEA7 = AB	
F988 - F98F = F4	FAB8 - FBAF = 31	FCF0 - FCF7 = 1C	FEA8 - FEA7 = 80	
F990 - F997 = 80	FAB8 - FBAF = 31	FCF8 - FCF7 = 99	FEB0 - FEB7 = 59	
F998 - F99F = F2	FAB8 - FBAF = 31	FDD0 - FDD7 = 75	FEB8 - FEB7 = 13	
		FDOB - FDOF = DR	FEC0 - FEC7 = 9C	

音頻波形 的觀測與記憶

前言

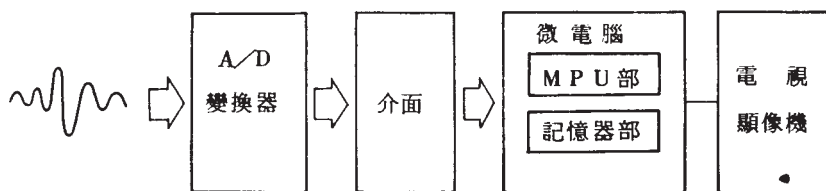
通常我們在作修理工作時，要觀測音頻波形總是會想到使用示波器，而且最好是一架高級示波器，它不但能夠顯示波形，也能夠將波形儲存（記憶）起來，這樣對修理的工作才會更方便，但是一部高級示波器的價格非常昂貴，如果你已經有一部 APPLE II 微電腦，請參考本文讓你的 APPLE II 擔任一部有儲存波形作用的示波器。

大家都知道，由於微電腦是屬於數位（digital）的領域，一般音頻信號則為類比（analog）之範疇，所以必須使用“類比／數

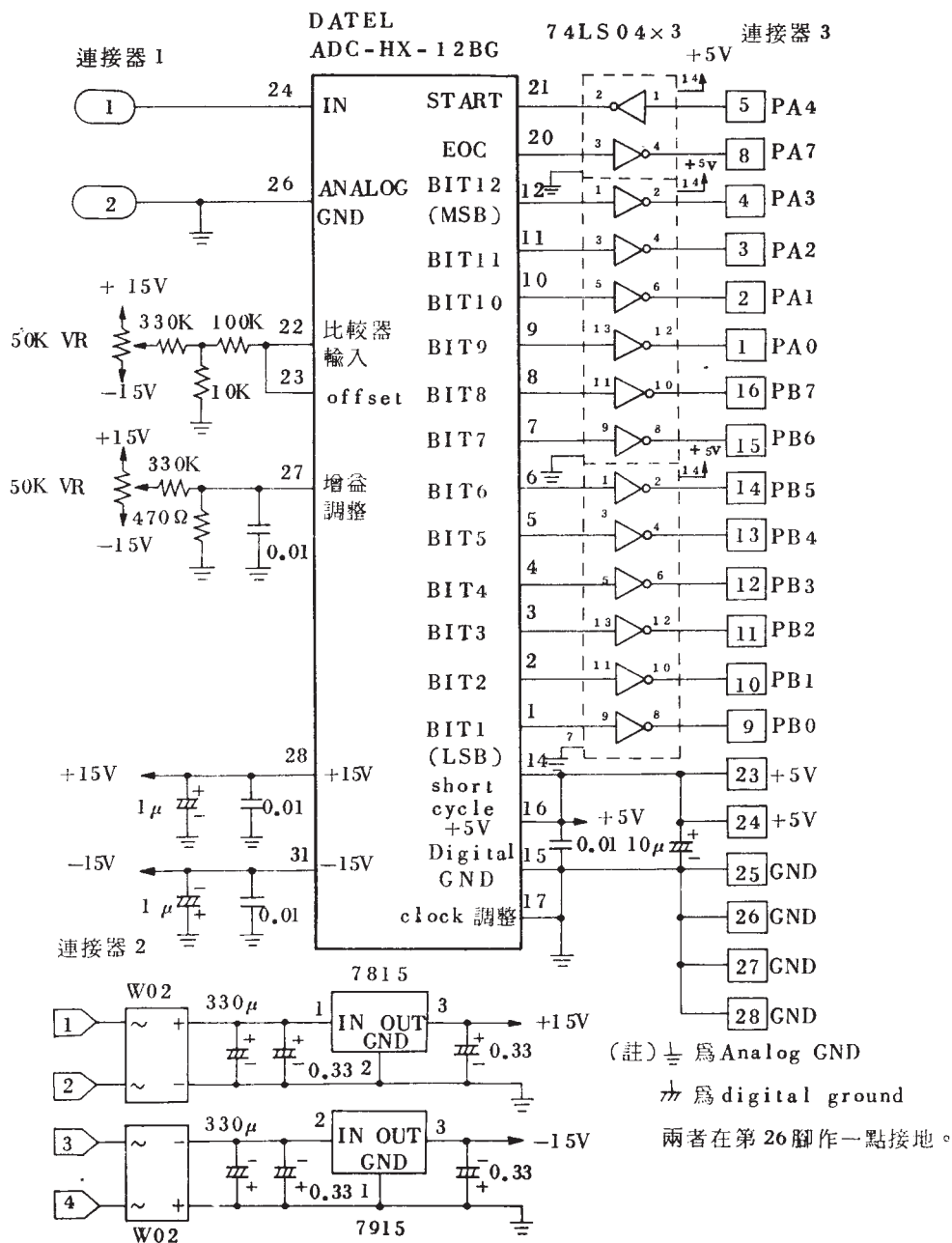
位轉換器”（Analog to digital converter……A/D converter）將類比值轉換為數位值後再行取入微電腦內。這就好比像數字三用表乃是先讀取類比值再以數字表示測量值是同樣道理的。

由 A/D 變換器所變換而來之資料（Data）被儲存（被記憶）於微電腦之記憶體內，這些資料和一般之資料同樣地可以在以後加以修正或整理，十分便利。

本文內將介紹如何把儲存於記憶體內之資料予以單純化或稍加整理而顯示於微電腦之電視顯像機上（第 1 圖）。



〔第 1 圖〕



〔第4圖〕 實際之電路例

硬體

A / D 變換部份

以音響之功率放大器的揚聲器端子間電壓為觀測對象，來設計 A / D 變換部份。一般而言，不論是那一家製造廠的 A / D 變換器 I C，其滿檔 (Full range) 電壓均為 $\pm 5V$ 或 $\pm 10V$ ，所以本文所介紹的 A / D 變換器之前面可以不加放大器，而直接可以在揚聲器端子上作測試。

所使用之 A / D 變換器為 DATEL 公司出品之 ADC-HX12 BGC，輸入電壓範圍有 $\pm 2.5V$ ， $\pm 5V$ ， $\pm 10V$ 3 種可供選擇，變換時間為 $20\mu S$ ，具有 12 bit 解析度之能力 (第 2 圖)。

所謂解析度 (resolution) 12 Bit 就是表示對滿檔 (Full range) 值可作 4096 (2^{12}) 個分割而變換成數位 (digital) 值，本文以 $\pm 5V$ 為滿檔值，所以 $+5V \sim -5V$ 共 $10V$ 作 4096 個分割 (被分割成 4096 段) 時每段為 $2.44mV$ ，意即小至此值之變化即可加以攫取。

所謂變換時間，就是將某一類比值 (analog value) 變換成數位值 (Digital value) 時所需之時間。作變換動作時需要時間，這就表示無法將類比資料 (analog data) 連續地變換成數位值，所以只能採用離散式之取樣 (Sampling) 法。

如欲辨認出一個頻率成分時，至少需要有

8 點之取樣點 (Sampling point)，所以 ADC-HX12 BGC，如果是在最快速度之情況下動作時，可掌握之變化率 (頻率) 如下：

$$20\mu S \times 8 = 160\mu S$$

$$\dots\dots \frac{1}{160\mu S} = 6,250\text{HZ} \text{ (如第 3 圖所示)}$$

實際之電路如第 4 圖所示，電路結構極為簡單，只需數個 C，R 及 A / D 變換器 (A / D converter) 即可構成之。

電阻採用金屬皮膜電阻，可變電阻採用溫度係數為數 100 ppm 者，只要注意幾點，那麼人人皆可安裝，且動作十分確實。

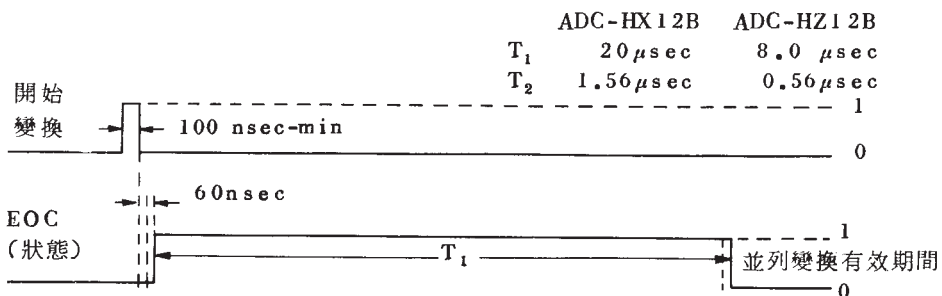
接收到 START 信號時，A / D 變換器開始進行 A / D 變換，正在進行變換工作之同時將 EOC 信號送往外部 (EOC 即 END OF CONVERSION 之縮寫，是結束 A / D 轉換之控制信號)。因此，微電腦首先送出 START 信號，令 A / D 變換器起動。接著，監視著 EOC 信號之狀態，如果確認已變換完成之後，則現在之類比值 (Analog value) 即可全部取入微電腦內 (第 5 圖)。

介面部份

介面部份就是用以將 A / D 變換部份和微電腦連結在一起的部份。

由於近年來 LSI 的發達，介面部份變得相當簡單，本文之介面即只需一個 LSI 即可構成。(第 6 圖)

LSI MC6821 乃是專為 MPU6800 系列，6500 系列而開發完成的可自由設定為輸入

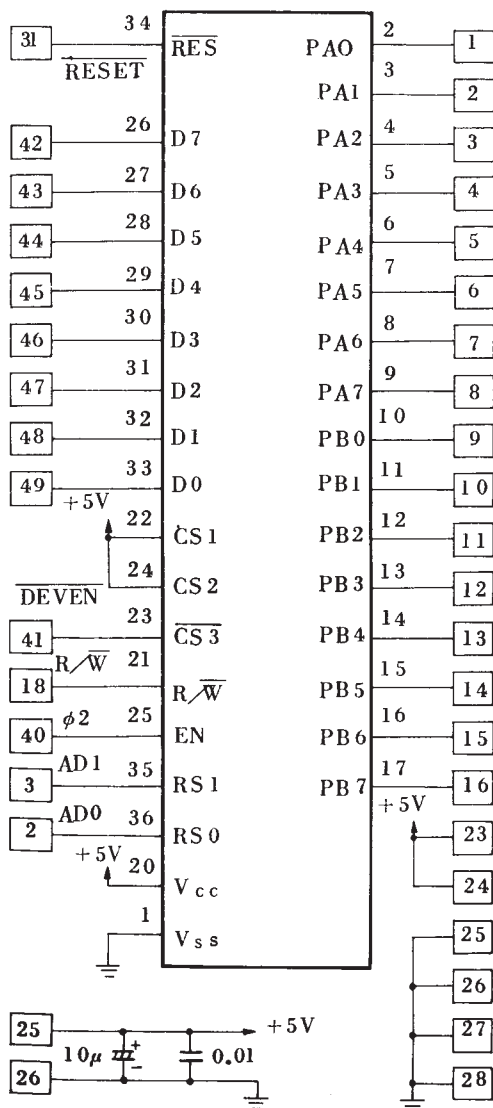


〔第 5 圖〕 時序 (Timing) 圖

或輸出的一種數位輸入輸出介面 IC，在本文內使用一個 MC6821 LSI，一支腳作輸出用（Start 信號用……PA4），13 支腳作為輸入用（資料線用 12 支腳，EOC 信號用 1 支腳，PBO~7，PA0~3，PA7）。

微電腦經由這個介面 IC 6821 將 start 信號送到 A/D 變換器內，並經由此一介面 IC

ageconnector MC6821 connector-4



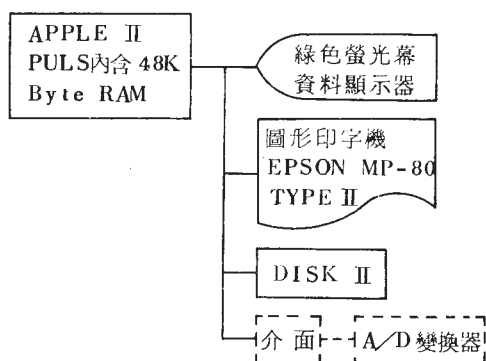
〔第 6 圖〕

接受來自 A/D 變換器之 EOC 信號及數位信號 (digital signal)。

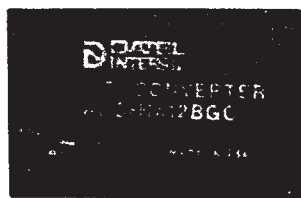
微電腦部份

微電腦使用的是個人用電腦 APPLE II PLUS。由於 APPLE II 具有高解像度之圖形 (graphic) 顯示機能，所以極為適合於本文所介紹之顯示音頻信號波形之用途。

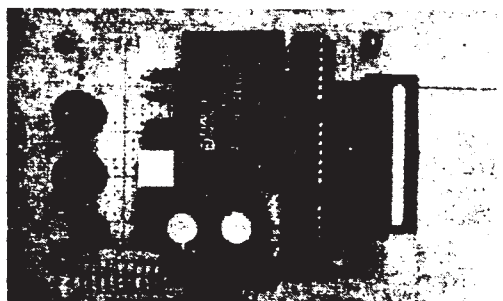
APPLE II 所使用之 MPU (微處理器) 為 6502，RAM 可擴充達 48K Byte。



〔第 7 圖〕



<照片 1> ADC-HX12BGC 之外觀



<照片 2> 本文之實驗所製作之 A/D 變換器

本文所介紹之系統由 APPLE II PLUS (48K Byte RAM) , 綠色螢幕資料顯示器 (green monitor) , 圖形印字機 EPSON MP-80 TYPE II 以及外部記憶裝置 DESK II 等所構成。(第7圖)

如果是僅止於執行本文所介紹之程式的場合只需 APPLE II PLUS (48K) 及資料顯示器即可。

軟體

本文之波形測試用程式，可分別為2部份，一個是由 A/D 變換器讀取資料之程式，另外一個是將所取入之資料顯示於資料顯示器上的程式。

JCALL-151

*1F00LLLLLL

```

1F00- A9 3F LDA ##3F
1F02- A9 1D STA $1D
1F04- A9 1C STA $1C
1F06- A9 00 LDA $00
1F08- BD A1 C0 STA $C0A1
1F0D- BD A3 C0 STA $C0A3
1F10- A9 10 LDA $10
1F12- BD A0 C0 STA $C0A0
1F15- A9 00 LDA $00
1F17- BD A2 C0 STA $C0A2
1F1A- A9 04 LDA $04
1F1C- BD A1 C0 STA $C0A1
1F1F- BD A3 C0 STA $C0A3
1F22- A9 10 LDA $10
1F24- BD A0 C0 STA $C0A0
1F27- D0 16 BNE $1F3F
1F29- BD A0 C0 LDA $C0A0
1F2C- F9 00 AND $F9
1F30- A9 00 LDA $00
1F32- BD A0 C0 LDA $C0A0
1F35- A9 0F AND $0F
1F37- 91 1C STA ($1C),Y
1F39- C8 INY
1F3D- BD A2 C0 LDA $C0A2
1F3F- 91 1C STA ($1C),Y
1F41- BD A0 C0 LDA $C0A0
1F44- A9 10 LDA $10
1F46- BD A0 C0 STA $C0A0
1F49- BD 02 LDA $02
1F4B- BD 1C ADC $1C
1F4D- BD 1C STA $1C
1F4F- BD 00 LDA $00
1F51- BD 1D ADC $1D
1F53- 95 STA $95
1F55- C9 CMP $95
1F57- D0 D7 BNE $1F30
1F59- RTS
1F5B- ???
1F5D- ???
1F5F- BRK
1F61- ???
1F63- BRK
1F65- ???

```

此部份乃是用以將介面之 PA4 定為輸出，PA7, PA3~0, PB7 ~ 0 定為輸入。

此部份用以將 A/D 變換器送來之資料讀入

資料之取入

資料之取入，採用如第8圖所示之組合語言所寫成之程式來執行。

這個程式的內容就是由 A/D 變換器每隔 $59\mu s$ 進行一次資料取樣 (Data Sampling)，然後逐一地將該資料儲存於 RAM 內。由於 RAM 之空間的限制，在進行到 11008 件取樣時就宣告終了，換句話說，本程式由起動點開始測試時間至 $649.472 ms$ ($= 59\mu s \times 11008$) 時終止。

1	起動後之資料值
2	$59\mu s$ 後之資料值
3	$118\mu s$ 後之資料值
4	$177\mu s$ 後之資料值
5	$236\mu s$ 後之資料值
6	$259\mu s$ 後之資料值
...	
11002	$649.059ms$ 後之資料值
11003	$649.118ms$ 後之資料值
11004	$649.177ms$ 後之資料值
11005	$649.236ms$ 後之資料值
11006	$649.295ms$ 後之資料值
11007	$649.354ms$ 後之資料值
11008	$649.413ms$ 後之資料值

[第8圖]

[第9圖]

測試後之資料如第 9 圖所示儲存於 R A M 上，儲存於 R A M 內之資料除非變更或電源切斷，否則它是保存於 R A M 之內不變的，因此，假設在第 9 圖內欲知道開始計測後 295 μ s 時之電壓的場合，只要查看第 6 個 R A M 的內容即可。

A / D 變換器可以作每隔 20 μ s 之取樣變換，而在此却只能每隔 59 μ s 作一次取樣，其

原因乃是由於 APPLE II 之微處理器 M P U 6502 的處理能力不足所致。因此，以 APPLE II 來作頻率成份分析時其最高頻率只能達 2KHz 左右，如下式所示。

$$59 \mu s \times 8 \text{ points} = 472 \mu s$$

$$\dots\dots \frac{1}{472 \mu s} = 2118 \text{Hz}$$

LIST

```

10 HIMEM: 7936: REM $1F00
20 D$ = ""
30 PRINT D$:"BLOAD GETAD"
40 DIM A(280)
50 PRINT D$:"PR#4"
60 PRINT " "
100 TEXT
110 HOME
120 PRINT "DATA GET ..... 1"
130 PRINT "DATA DISPLAY ..... 2"
140 PRINT "END ..... 3"
150 INPUT " "
160 IF A = 1 THEN GOTO 1000
170 IF A = 2 THEN GOTO 2000
180 END
1000 HOME
1005 TEXT
1010 INPUT "START OK : ";A$
1020 CALL 7936
2000 HOME
2010 PRINT "DISPLAY READY !!"
2020 PRINT
2030 INPUT "DISPLAY SIZE : ";A1
2035 IF A1 < 280 THEN PRINT "ERR. INPUT": GOTO 2030
2040 INPUT "SCALE : ";B1
2045 INPUT "METHOD : ";SW
2050 INPUT "START POSITION : ";P1
2100 VTAB 22
2110 PRINT "SIZE = ";A1,"SCALE = ";B1
2120 PRINT "METHOD= ";SW,"POSITION= ";P1
3000 A2 = INT (A1 / 280)
3010 P2 = (P1 - 1) * 2 + 16384
3020 HGR : HCOLOR= 3
3030 HPLOT 0,80 TO 279,80
3100 ON SW GOSUB 5000,7000,4000,6000
3120 GOTO 100
4000 FOR X = 0 TO 279
4010 C1 = 0: C2 = 0
4020 FOR J = P2 TO P2 + A2 * 2 - 1 STEP 2
4030 C1 = C1 + PEEK (J)
4040 C2 = C2 + PEEK (J + 1)
4050 NEXT J
4060 P2 = P2 + A2 * 2
4070 C = INT ((C1 * 256 + C2) / A2)
4080 C = C - 2048
4090 Y = INT (C * 160 / B1)
4095 A(X) = Y
4100 Y = 80 - Y
4110 GOSUB 9000
4120 NEXT X
4130 RETURN
5000 FOR X = 0 TO 279
5010 C = PEEK (P2) * 256 + PEEK (P2 + 1)
5020 C = C - 2048
5030 Y = INT (C * 160 / B1)
5035 A(X) = Y
5040 Y = 80 - Y
5050 P2 = P2 + A2 * 2
5060 GOSUB 9000
5070 NEXT X
5080 RETURN
6000 GOSUB 4000
6005 HGR : HPLOT 0,80 TO 279,80
6010 FOR X = 2 TO 277
6020 A(X) = INT ((A(X - 2) * 0.25 + A(X - 1) * .5
+ A(X) + A(X + 1) * 0.5 + A(X + 2) * .25) / 2.5)
6030 NEXT X
6040 FOR X = 2 TO 277
6050 Y = 80 - A(X)
6060 GOSUB 9000
6070 NEXT X
6080 RETURN
7000 GOSUB 5000
7010 GOTO 6005
9000 IF Y < 0 THEN Y = 0
9010 IF Y > 160 THEN Y = 160
9020 HPLOT X,160 TO X,Y
9030 RETURN

```

MENU 選擇部份

資料取入程式之起動

顯示MENU 選擇部份

作METHOD 3 之處理的部份

作METHOD 1 之處理的部份

作METHOD 4 之處理的部份

作METHOD 2 之處理的部份

如欲分析更高之頻率的成份時，那麼就需把 A/D 變換器所輸出之資料儲存到專用之硬體電路內而不再是儲存在 RAM 內。

以資料顯示器來顯示資料

將資料顯示於資料顯示器時採用第10圖所示之以 BASIC 語言所寫成的程式。

執行此一程式時，在螢光幕上出現如第11圖所示的文字。

此時由 APPLE II 之鍵盤輸入 1 時，令前面所述之資料取入程式起動，在 0.7 秒之內進入測試動作狀態。如果由鍵盤輸入 2 時，已儲存於 RAM 內之測試資料即將顯示出來，此時在螢光幕上顯示如第12圖所示之字樣。

此時 APPLE II 向操作者要求下達下述 4 項之指示。

DISPLAY SIZE:

此一項目就是要求指示在監視器 (monitor TV) 之水平方向全域之相對應 RAM 上的 11008 點應顯示多大之範圍。照片 - 3 所示為以 11000 點為全域之情形，而照片 4 則為以 4800 點為全域之場合的顯示波形，……照片 - 6 所示為以 300 點為全域之場合的顯示波形。由這些照片可以看出，這就如同示波器之時間軸變化。

```
DATA GET ..... 1
DATA DISPLAY ..... 2
END ..... 3
```

〔第11圖〕

```
DISPLAY READY !!
DISPLAY SIZE :
SCALE :
METHOD :
START POSITION :
```

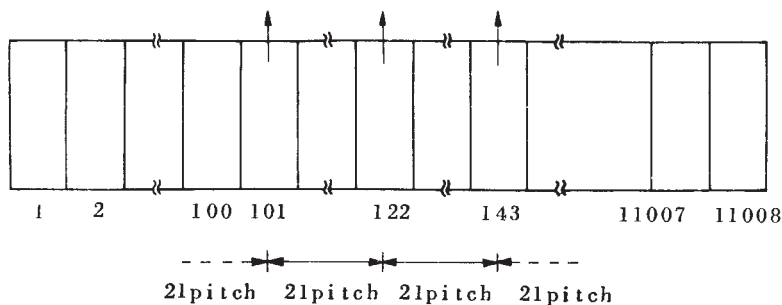
〔第12圖〕

監視器上電壓顯示範圍之決定

這個項目就是 APPLE II 向使用者要求指示在監視器 (monitor TV) 之垂直方向全域之電壓刻度範圍。例如，如果使用者下達指示值為 4096 時，則垂直方向全域為 $\pm 5V$ ，如果所下達之指示值為 2048 時，垂直方向全域為 $\pm 2.5V$ 。

顯示方法

雖說 APPLE II 具有高解像度之圖形顯示機能，但仍無法將多達 11000 點之資料全部顯示於畫面上。由於 APPLE II 的水平方向的解像度為 280 點，此一項目就是用以指示如何將 11000 點約簡為 280 點。



〔第13圖〕

METHOD:1

指定為 1 時，假設 DISPLAY SIZE 指定為 6000 時，則每隔 21 點 ($6000 \div 280$) 之間距由 RAM 取出資料而顯示於螢光幕上，意

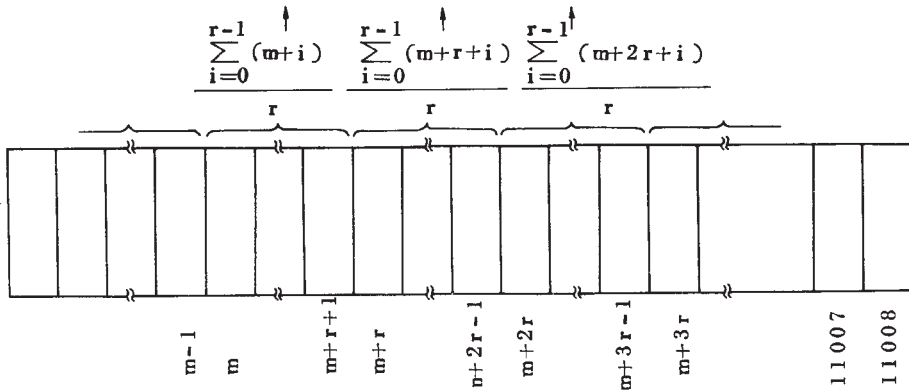
即取樣 (Sampling) 間隔變更。(第13圖)

METHOD:2

如果指定為 2 時，和 1 同樣地取出 280 點並依下式求加重移動平均值而顯示出來。

$$Y_i = \frac{Y_{i-2} \times 0.25 + Y_{i-1} \times 0.5 + Y_i + Y_{i+1} \times 0.5 + Y_{i+2} \times 0.25}{2.5}$$

像音頻信號這類複雜而未具特徵的資料，採取這種平均值到底具有何種意義呢？這是要稍微費一點時間才能下判斷的，不過介紹此法之目的在於讓讀者了解有這種用途存在。

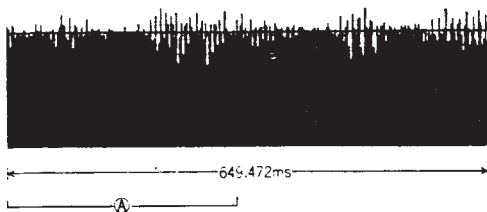


[第14圖] $r: pitch$ $(m): m$ 之點電壓

METHOD:3

指定為3時，和1同樣地首先決定間距（pitch），但是，並非顯示出每一間距之RAM的內容，而是計算每一間距之間的全部資料的單純平均之後再將該值顯示出來。（第14圖）雖說能將資料之全部顯示於畫面上最好不過，然而，假設SIZE為11000時，間距數多達39，此時如果將其顯示出來時實令人無法了解所顯示之內容。因此，至少應將資料之全體像限制在10個間距之內。

```
DISPLAY SIZE : 11000
SCALE       : 4000
METHOD      : 1
START POSITION : 1
```



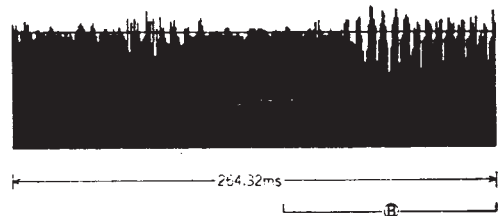
< 照片 3 >

METHOD:4

指定為4時，以METHOD 3所求得之280點再以METHOD 2相同之方法求加重移動平均值再行顯示出來。

1~4四種METHOD均為統計學之初步所使用的一種分析手段，利用此法來作音頻信號波形之分析可以說是以往夢想不到的，在第15圖所示之各顯示例中，如2-5就如同是將1-5之波形通過低頻濾波器（LPF）後一般，可以從中發覺十分有趣的地方。

```
DISPLAY SIZE : 4800
SCALE       : 4000
METHOD      : 1
START POSITION : 1
```

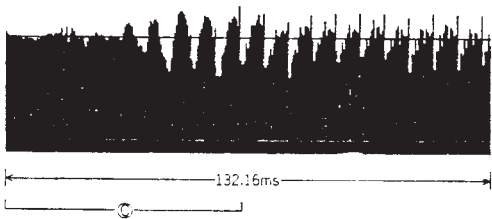


< 照片 4 >

1-1	2-1	3-1	4-1
1-2	2-2	3-2	4-2
1-3	2-3	3-3	4-3
1-4	2-4	3-4	4-4
1-5	2-5	3-5	4-5
1-6	2-6	3-6	4-6

[第15圖]

DISPLAY SIZE : 2400
 SCALE : 4000
 METHOD : 1
 START POSITION : 2700



<照片 5>

DISPLAY SIZE : 11000
 SCALE : 4000
 METHOD : 1
 START POSITION : 1



<照片 7> 人聲(女)

DISPLAY SIZE : 11000
 SCALE : 4000
 METHOD : 1
 START POSITION : 1

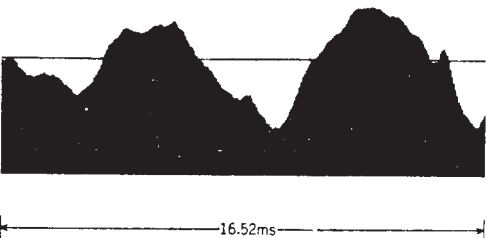


<照片 8> 人聲(男)

POSITION:

此項目就是指示從RAM之第幾個位置開始將資料顯示於螢光幕上。照片-3之㊸部份顯示成照片-4，照片-4之㊸的部份顯示成照片5……，此均為受POSITION之指示而發生的變化。這有點像示波器之TIME-BASS SWEEP。

DISPLAY SIZE : 300
 SCALE : 2000
 METHOD : 1
 START POSITION : 3300



<照片 6>

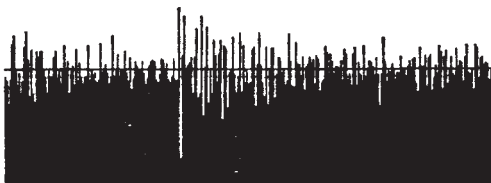
DISPLAY SIZE : 1200
 SCALE : 4000
 METHOD : 1
 START POSITION : 8000



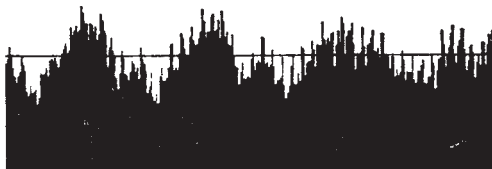
DISPLAY SIZE : 1200
 SCALE : 4000
 METHOD : 1
 START POSITION : 5000



DISPLAY SIZE : 11000
SCALE : 4000
METHOD : 1
START POSITION : 1

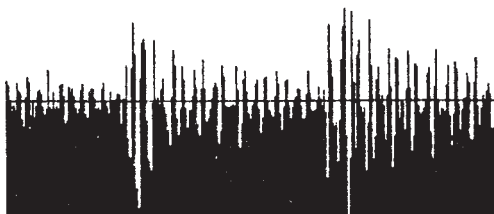


DISPLAY SIZE : 1200
SCALE : 4000
METHOD : 1
START POSITION : 1



< 照片 9 > 通俗音樂

DISPLAY SIZE : 11000
SCALE : 4000
METHOD : 1
START POSITION : 1

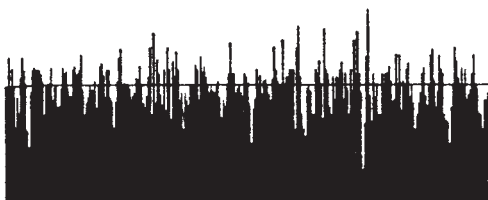


DISPLAY SIZE : 1200
SCALE : 4000
METHOD : 1
START POSITION : 2000

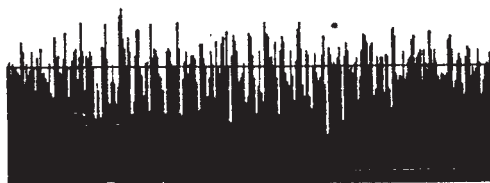


< 照片 10 > 鼓聲

DISPLAY SIZE : 11000
SCALE : 4000
METHOD : 1
START POSITION : 1



DISPLAY SIZE : 1200
SCALE : 4000
METHOD : 1
START POSITION : 1



各種音頻信號波形之測試例

照片 7 ~ 照片 11 所示為各種成音信號測試結果。

照片 - 7 所示為女性之聲音，照片 - 8 所示為男性之聲音，由此可以發現微電腦系統亦能覺查出女性的高頻成份較多。

照片 - 10 所示為鼓聲之波形，照片 - 11 所示為鑼鈸聲之波形，由波形上可以發現兩者之特徵不同。

DISPLAY SIZE : 300
SCALE : 4000
METHOD : 1
START POSITION : 1



< 照片 11 > 鑼鈸聲

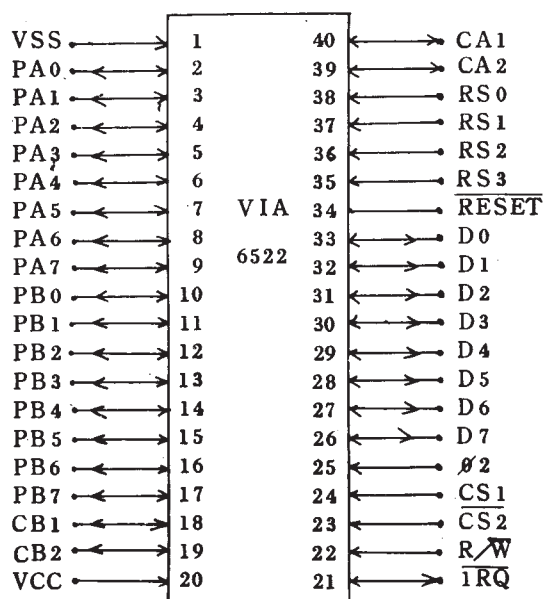
利用6522製作一個——

活動廣告看板

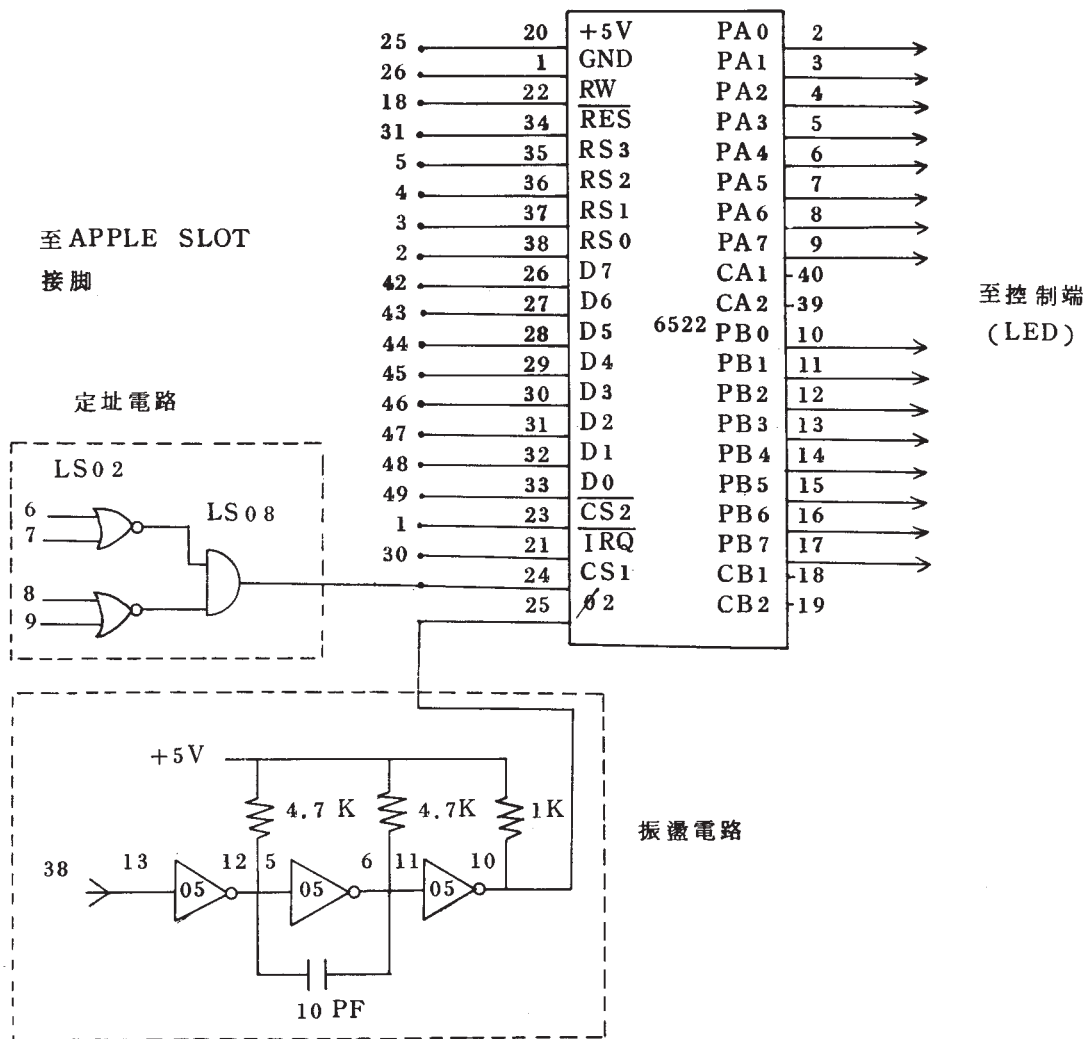
APPLE II 只能玩 GAME 嗎？答案當然是否定的。只要你能善加利用，也不難成為一台萬用的電腦，本文將介紹如何利用 APPLE II 製作活動廣告看板，所謂活動看板，就是由許多發光點（如燈炮、LED……）所組成的方形矩陣，我們可以事先在上面寫上一些廣告詞，藉著上移、下移、左移、右移，以便在有限的字幕上顯示出整句台詞。這種廣告看板相信在不久的將來會逐漸取代傳統式的看板。

6522和電腦的连接

6522 為一多功能界面轉接器（VIA），可算是 6500 系列界面晶片中功能最好的一個，具有 16 BIT 輸入／輸出埠，4 個握手式信號控制，二個 16 BIT 計數器，串並聯轉換……等功能。



6522 接脚圖



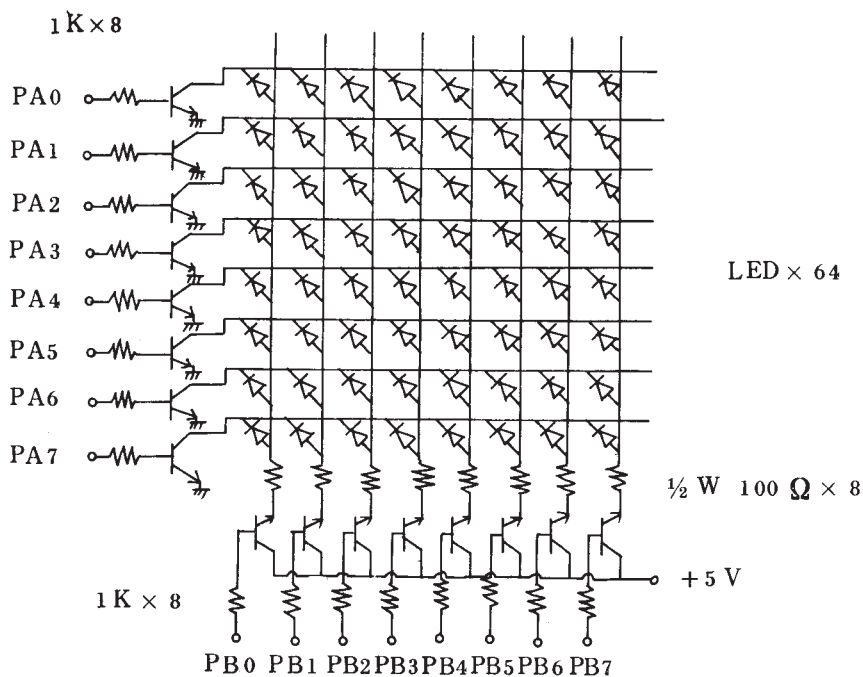
6522 與 I/O 槽連接圖

位 址	功 能	
\$C100	B 埠 輸入／輸出資料暫存器	
\$C101	A 埠 輸入／輸出資料暫存器	
\$C102	B 埠資料方向暫存器	0 = 輸入 1 = 輸出
\$C103	A 埠資料方向暫存器	
\$C104	(讀) 計數器 1 低位元，並清除中斷旗幟 (寫) 計數器 1 低位元	
\$C105	(讀) 計數器 1 高位元 (寫) 計數器 1 高位元，並清除中斷旗幟	
\$C106	(讀) 計數器 1 低門栓 (寫) 計數器 1 低門栓	
\$C107	(讀) 計數器 1 高門栓 (寫) 計數器 1 高門栓，並清除中斷旗幟	
\$C108	(讀) 計數器 2 低位元，並清除中斷旗幟 (寫) 計數器 2 低門栓	
\$C109	(讀) 計數器 2 高位元 (寫) 計數器 2 高位元，並清除中斷旗幟	
\$C10A	位移暫存器	
\$C10B	輔助控制暫存器	
\$C10C	週邊控制暫存器	
\$C10D	中斷旗幟暫存器	
\$C10E	中斷認可暫存器	
\$C10F	A 埠輸入／輸出暫存器	

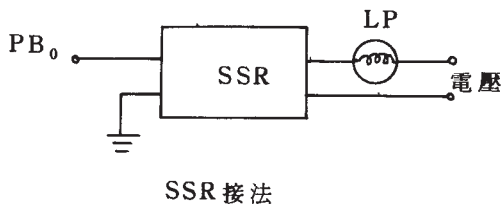
6522內16個暫存器位址、SLOT暫定在“1”

顯示板的製作

本製作以 8×8 的矩陣來顯示字幕，
爲了節省控制線以下面的方式設計：



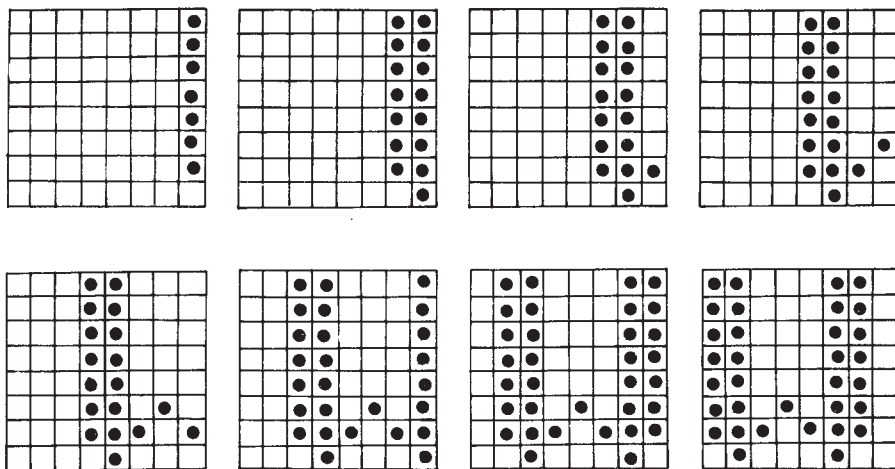
如果覺得LED不夠氣派，可以用燈炮來接，但如果換燈炮，驅動電路就不能只用電晶體，最好用SSR(固態繼電器)。



如何產生動態的畫面

畫面移動的原理和電影、卡通一樣，

是一張一張畫面顯示出來，因為人的視覺殘留而產生動態的畫面，例“W”由右而左分解如下：

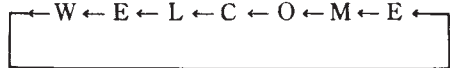


程式的設計

本線路矩陣長、寬分別以 PORT A 及 PORT B 控制，如我們以 PORT A 為資料輸出，PORT B 為選擇信號，畫面就由左→右或右→左，反之就是由上→下或下→上。程式(一)(二)(三)(四)分別為這四種狀態。

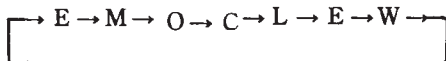
程式(一)

由右而左



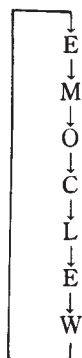
程式(二)

由左而右



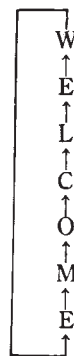
程式(三)

由上而下



程式(四)

由下而上



4 個程式結構大致一樣，下面簡介程式功能：

行 號	功 能
5-6	設定二 PORT 均為輸出
11-14	行、列的資料輸出
17-18	換行延遲時間
26-27	換頁延遲時間
35-49	顯示資料，讀者可修改成自己所希望的字形

※全部程式可用 LISA 去輸入，也可以直接輸入 MACHINE CODE。

左移程式

```
!A
**END OF PASS 1
```

```
**END OF PASS 2
```

```
0800          1  ;6522 I/O MOVEING DISPLAY
0800          2  ;L <---- R MOVE
FCAB          3  DELAY      EQU $FCAB
1000          4          ORG $1000
1000 A9 FF      5  START    LDA #$FF
1002 8D 02 C1    6          STA $C102
1005 8D 03 C1    7          STA $C103
1008 A9 01      8  A1       LDA #$1
100A 8D 10 23    9          STA $2310
100D A2 00     10         LDX #$0
100F BD 00 20   11  LOOP    LDA $2000,X
1012 8D 01 C1   12         STA $C101
1015 AD 10 23   13         LDA $2310
1018 8D 00 C1   14         STA $C100
101B 18         15         CLC
101C 2E 10 23   16         ROL $2310
101F A9 40      17         LDA #$40
1021 20 A8 FC   18         JSR DELAY
1024 E8         19         INX
1025 E0 08      20         CPX #$08
1027 D0 E6      21         BNE LOOP
1029 EE 10 10   22         INC $1010
102C AD 10 10   23         LDA $1010
102F C9 43      24         CMP #$43
1031 F0 09      25         BEQ QUIT
1033 A9 30      26         LDA #$30
1035 20 A8 FC   27         JSR DELAY
```

1038	4C 08 10	28		JMP A1
103B	60	29		RTS
103C	A9 00	30	QUIT	LDA #\$0
103E	8D 10 10	31		STA \$1010
1041	4C 00 10	32		JMP START
1044		33	;DATA	
2000		34		ORG \$2000
2000	00 00 00	35		HEX 0000000000000000
2003	00 00 00			
2006	00			
2007	7F FF 40	36		HEX 7FFF403040FF7F ;W
200A	30 40 FF			
200D	7F			
200E	00 00	37		HEX 0000
2010	FF FF 99	38		HEX FFFF99999981 ;E
2013	99 99 81			
2016	00 00	39		HEX 0000
2018	FF FF C0	40		HEX FFFFC0C0C0C0 ;L
201B	C0 C0 C0			
201E	00 00	41		HEX 0000
2020	3C 7E C3	42		HEX 3C7EC3C3C3C324 ;C
2023	C3 C3 C3			
2026	24			
2027	00 00	43		HEX 0000
2029	7E 7E 81	44		HEX 7E7E8181817E7E ;O
202C	81 81 7E			
202F	7E			
2030	00 00	45		HEX 0000
2032	FF FF 02	46		HEX FFFF020C02FFFF ;M
2035	0C 02 FF			
2038	FF			
2039	00 00	47		HEX 0000
203B	FF FF 99	48		HEX FFFF99999981 ;E
203E	99 99 81			
2041	00 00 00	49		HEX 00000000000000000000000000000000
2044	00 00 00			
2047	00 00 00			
204A	00 00 00			
204D	00 00			
204F		50		END

右移程式

0800		1	;6522 I/O MOVEING DISPLAY
0800		2	;L -----> R MOVE
FCA8		3	DELAY EQU \$FCA8
1000		4	ORG \$1000
1000	A9 FF	5	START LDA #\$FF
1002	8D 02 C1	6	STA \$C102
1005	8D 03 C1	7	STA \$C103
1008	A9 80	8	A1 LDA #\$80

100A	8D 10 23	9		STA \$2310
100D	A2 00	10		LDX #\$0
100F	BD 00 20	11	LOOP	LDA \$2000,X
1012	8D 01 C1	12		STA \$C101
1015	AD 10 23	13		LDA \$2310
1018	8D 00 C1	14		STA \$C100
101B	18	15		CLC
101C	6E 10 23	16		ROR \$2310
101F	A9 40	17		LDA #\$40
1021	20 A8 FC	18		JSR DELAY
1024	EB	19		INX
1025	E0 08	20		CPX #\$08
1027	D0 E6	21		BNE LOOP
1029	EE 10 10	22		INC \$1010
102C	AD 10 10	23		LDA \$1010
102F	C9 43	24		CMP #\$43
1031	F0 09	25		BEQ QUIT
1033	A9 30	26		LDA #\$30
1035	20 A8 FC	27		JSR DELAY
1038	4C 08 10	28		JMP A1
103B	60	29		RTS
103C	A9 00	30	QUIT	LDA #\$0
103E	8D 10 10	31		STA \$1010
1041	4C 00 10	32		JMP START
1044		33	; DATA	
2000		34		ORG \$2000
2000	00 00 00	35		HEX 0000000000000000
2003	00 00 00			
2006	00			
2007	7F FF 40	36		HEX 7FFF403040FF7F ;W
200A	30 40 FF			
200D	7F			
200E	00 00	37		HEX 0000
2010	81 99 99	38		HEX 81999999FFFF ;E
2013	99 FF FF			
2016	00 00	39		HEX 0000
2018	C0 C0 C0	40		HEX C0C0C0C0FFFF ;L
201B	C0 FF FF			
201E	00 00	41		HEX 0000
2020	24 C3 C3	42		HEX 24C3C3C3C37E3C ;C
2023	C3 C3 7E			
2026	3C			
2027	00 00	43		HEX 0000
2029	7E 7E 81	44		HEX 7E7E8181817E7E ;O
202C	81 81 7E			
202F	7E			
2030	00 00	45		HEX 0000
2032	FF FF 02	46		HEX FFFF020C02FFFF ;M
2035	0C 02 FF			
2038	FF			
2039	00 00	47		HEX 0000
203B	81 99 99	48		HEX 81999999FFFF ;E
203E	99 FF FF			

200F 0 00	37	HEX 0000
2011 7F 03 03	38	HEX 7F03033F3F03037F ;E
2014 3F 3F 03		
2017 03 7F		
2019 00 00	39	HEX 0000
201B 7F 7F 03	40	HEX 7F7F030303030303 ;L
201E 03 03 03		
2021 03 03		
2023 00 00	41	HEX 0000
2025 3C 3E 43	42	HEX 3C3E430303433E3C ;C
2028 03 03 43		
202B 3E 3C		
202D 00 00	43	HEX 0000
202F 1C 63 63	44	HEX 1C6363636363631C ;O
2032 63 63 63		
2035 63 1C		
2037 00 00	45	HEX 0000
2039 63 63 63	46	HEX 63636363636363 ;M
203C 63 6B 6B		
203F 77 63		
2041 00 00	47	HEX 0000
2043 7F 03 03	48	HEX 7F03033F3F03037F ;E
2046 3F 3F 03		
2049 03 7F		
204B 00 00 00	49	HEX 000000000000000000000000
204E 00 00 00		
2051 00 00 00		
2054 00 00 00		
2057 00 00		
2059	50	END

上移程式

0800	1	;6522 I/O MOVEING DISPLAY
0800	2	;UP MOVE
FCA8	3	DELAY EQU \$FCAB
1000	4	ORG \$1000
1000 A9 FF	5	START LDA #\$FF
1002 8D 02 C1	6	STA \$C102
1005 8D 03 C1	7	STA \$C103
1008 A9 01	8	A1 LDA #\$1
100A 8D 10 23	9	STA \$2310
100D A2 00	10	LDX #\$0
100F BD 00 20	11	LOOP LDA \$2000,X
1012 BD 00 C1	12	STA \$C100
1015 AD 10 23	13	LDA \$2310
1018 8D 01 C1	14	STA \$C101
101B 18	15	CLC
101C 2E 10 23	16	ROL \$2310
101F A9 40	17	LDA #\$40
1021 20 A8 FC	18	JSR DELAY

1024	EB	19	INX
1025	E0 08	20	CPX #\$0B
1027	D0 E6	21	BNE LOOP
1029	EE 10 10	22	INC \$1010
102C	AD 10 10	23	LDA \$1010
102F	C9 4F	24	CMP #\$4F
1031	F0 09	25	BEQ QUIT
1033	A9 30	26	LDA #\$30
1035	20 A8 FC	27	JSR DELAY
1038	4C 08 10	28	JMP A1
103B	60	29	RTS
103C	A9 00	30	QUIT LDA #\$0
103E	8D 10 10	31	STA \$1010
1041	4C 00 10	32	JMP START
1044		33	; DATA
2000		34	ORG \$2000
2000	00 00 00	35	HEX 0000000000000000
2003	00 00 00		
2006	00		
2007	63 63 63	36	HEX 636363636B6B7722 ;W
200A	63 6B 6B		
200D	77 22		
200F	00 00	37	HEX 0000
2011	7F 03 00	38	HEX 7F03033F3F03037F ;E
2014	3F 3F 03		
2017	03 7F		
2019	00 00	39	HEX 0000
201B	03 03 03	40	HEX 0303030303037F7F ;L
201E	03 03 03		
2021	7F 7F		
2023	00 00	41	HEX 0000
2025	3C 3E 43	42	HEX 3C3E430303433E3C ;C
2028	03 03 43		
202B	3E 3C		
202D	00 00	43	HEX 0000
202F	1C 63 63	44	HEX 1C63636363631C ;D
2032	63 63 63		
2035	63 1C		
2037	00 00	45	HEX 0000
2039	63 77 6B	46	HEX 63776B6B63636363 ;M
203C	6B 63 63		
203F	63 63		
2041	00 00	47	HEX 0000
2043	7F 03 03	48	HEX 7F03033F3F03037F ;E
2046	3F 3F 03		
2049	03 7F		
204B	00 00 00	49	HEX 00000000000000000000000000000000
204E	00 00 00		
2051	00 00 00		
2054	00 00 00		
2057	00 00		
2059		50	END

看了前面的說明，是不是很吸引人，建議你立刻去買一片萬用板，將所需的零件焊上，試試看效果如何。如果覺得 8×8 矩陣太小了，可多用幾個 6522 來控制，筆者曾用 4 個 6522 控制 16×48 點矩陣，效果還不錯可以試一試。下面將介紹一個規模不算小的廣告看板，希望對有興趣的讀者有更進一步的幫助。首先我必須聲明，讀者在製作這個看板時，先衡量一下自己的經濟能力（製作費：不包括電腦，大概在 NT\$5000 元以上），如能找幾個朋友合資來做最好不過。

控制界面的選擇

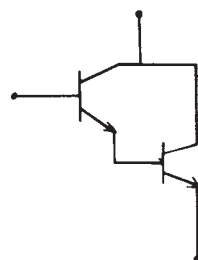
在所多 I/O 晶片，如 6821，8255，6522 ……等。都可以用在活動看板的製作上，這些晶片通常都具備了輸入／輸出的功能，而在本製作中只用到輸出的功能，其他的功能也就浪費掉了，如果讀者對電腦界面製作還熟悉，可以改用 74LS273 亦可，但本文介紹還是以 6522 為主。

螢幕規格選擇

這個看板的規格，定為 16×70 點，也就是由 1120 個 LED 所組成的矩陣，點數愈多，字體也愈漂亮，之所以定為 16×70 ，完全是為配合漢卡 16×16 字體，如果還要再擴大，可以這 16×70 為一基本單位拼湊起來，要多大有多大。

驅動電路

16×70 的點矩陣，共需要 $16 + 70 = 86$ 個驅動電路，筆者用二個電晶體接成達靈頓放大電路，效果還好，讀者如有更好的電路也可以，不過不可以用機械式的繼電器，不僅聲音太大，速度也不夠快



2SC2236 $\times 2$

達靈頓放大接法

電源

控制界面直接從電腦取出電源，顯示板因為比較費電，如用電腦中的電源，功率大的電源還可承受，最好獨立接電源，以免造成電源的傷害。外接電源原則上使用 5 V，最簡單的可以用 7805 系列穩壓 IC，功率愈大愈好。

其他

顯示板上的 LED 儘量排到整齊，以免影像移動時產生飄浮的現象。品質最好一致（亮度一致），這樣比較美觀。線路板最好用萬用板拉綫即可，如果洗印刷電路板，數量多還可以，一、二片就不太合算。控制界面和顯示板之間的連接，大約須 90 條，用排線即可。

輸入資料

在活動看板的製作中，如何輸入資料是個關鍵問題，要達到快又美觀，英文字體沒有什麼問題，最多連大小寫及阿拉伯數字，一些符號加起來也不過 100 個左右，可以事先畫好，要用再拿出來。中文字體就完全不同，不僅字數多，光是教育部公布常用字就有 3400 個字，用人力去劃十分浪費時間又不見得美觀，於是就有利

用漢卡的構想，一來輸入資料方便迅速，連英文、阿拉伯數字符號都有了。

以上先為製作前交代一番，接下該談主題了，先從 6522 界面卡說起。卡上共有 6 個 6522，一個 74LS05，74LS138 和幾個電阻電容，用一片 APPLE 的界面萬用板，足可容納得下，IC 最好用 IC 座，如果你有把握直接焊上也無妨。這片界面卡原理十分簡單，74LS05 振盪用，74LS138 定址用，使 6 個 6522 分別佔在：

- (1) \$CN00 ~ \$CN0F
- (2) \$CN10 ~ \$CN1F
- (3) \$CN20 ~ \$CN2F

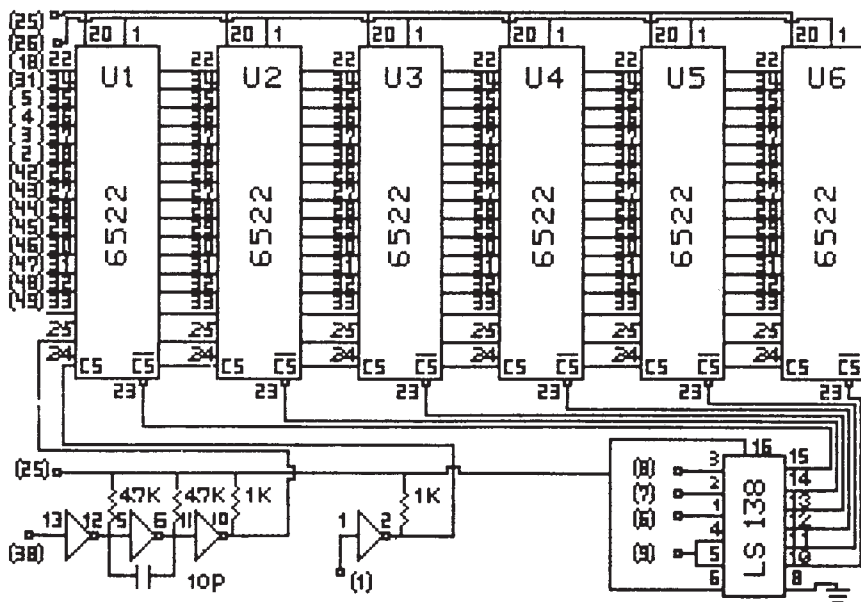
(4) \$CN30 ~ \$CN3F

(5) \$CN40 ~ \$CN4F

(6) \$CN50 ~ \$CN5F

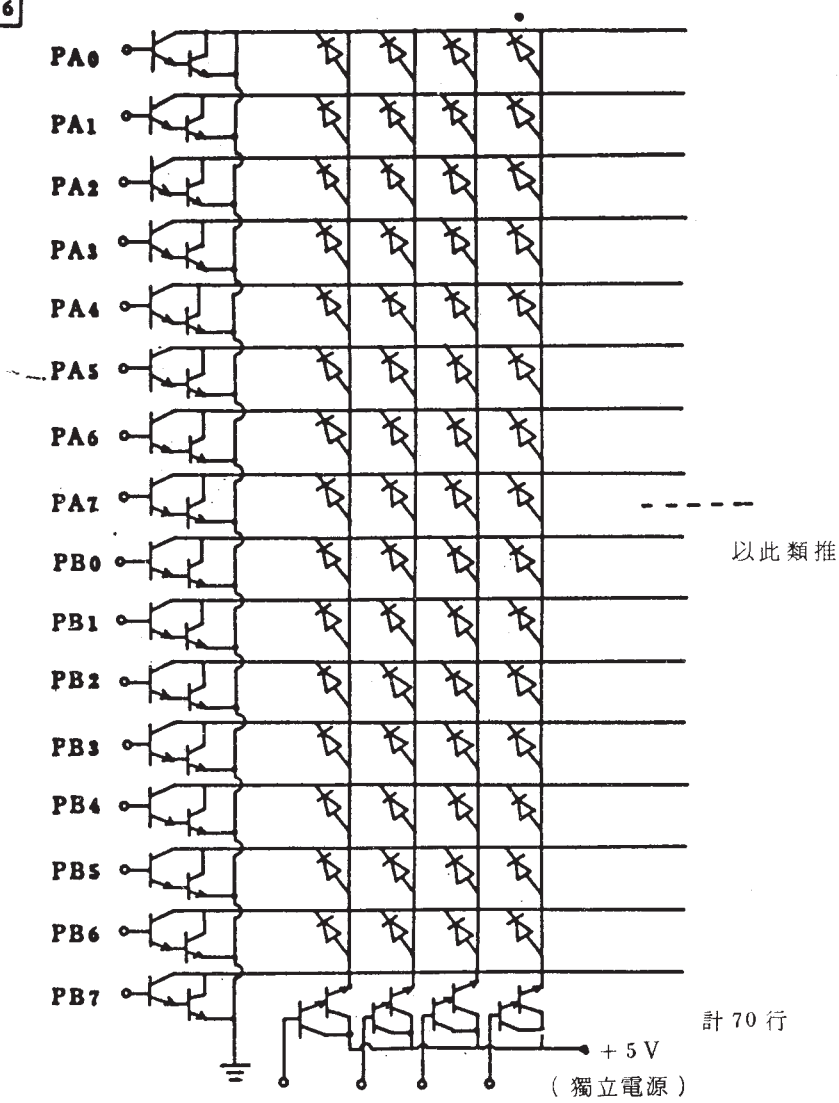
N：為 SLOT 號碼

由線路上看來不難發現，6 個 6522 除了 23 PIN (CS2)，PA0-7，PB0-7，CA1-2，CB1-2 外，其他接腳均接在一起，CS2 分別接至 74LS138 做定址用，CA1-2 CB1-2 空接不用，PA0-6 PB0-6 經過驅動電路接至顯示板 X 軸及 Y 軸，PA7 PB7 不接，為何不用呢？後面再說明。



註：有〔 〕記號表示為 SLOT 接腳
控制界面線路

U6

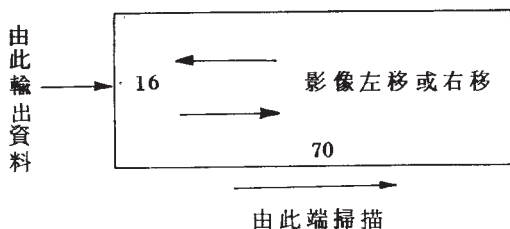


X 軸

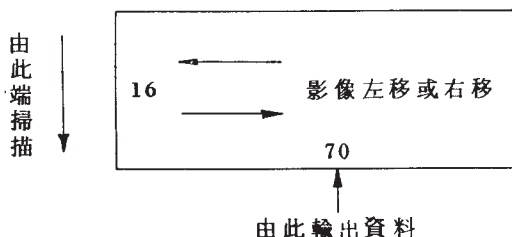
				
U1	U2	U3	U4	U5
PA0 ~ PA6 PB0 ~ PB6	PA0 ~ PA6 PB0 ~ PB6	PA0 ~ PA6 PB0 ~ PB6	PA0 ~ PA6 PB0 ~ PB6	PA0 ~ PA6 PB0 ~ PB6

* 顯示板接線圖 *

上述硬體製作均告完成後，按下該談軟體控制部份，要利用矩陣方式顯示出字形來，非得要用掃描方式，以人的視覺殘留來達到移動的效果，掃描的方式不外有二種：



這種方式來顯示程式比較簡單，但顯示速度慢下來的時候，有閃動的現象，而且顯示幕加大，亮度也會減弱。



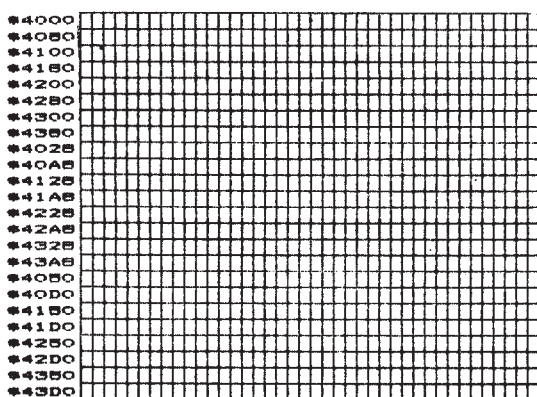
此種方式較第一種效果好多了，程式也比較複雜，顯示速度慢下來也不會有閃動的現象，而且不管顯示幕如何加大，也感覺不出亮度有減弱。

很明顯以上二種方式，後者比前者好，本文所介紹的程式是以後者的方式設計的，所以效果不錯。

如何利用漢卡輸入資料

我在還沒有利用漢卡輸入字型之前，在顯示幕上的資料都要先畫在方格紙上後

，修修改改，再轉換成顯示用的機械碼然後才輸入電腦，十分費時且易出錯。要是這看板是用來報告新聞，等到稿文全部輸入完後，恐怕都變成舊聞了。自從利用漢卡後，只要懂得“倉頡輸入法”，要有多快就有多快。其實也不是直接將漢卡的字搬到顯示幕上，而是先顯示到第二頁高解析螢幕上（HGR 2）再轉到顯示幕上，所以我們必須先對HGR 2結構徹底了解。



0	50000
1024	50400
2048	50800
3072	51200
4096	51600
5120	52000
6144	52400
7168	52800

BIT

0	1	2	3	4	5	6
---	---	---	---	---	---	---

高解析度格式

每一個小格子內

每一小方塊內有 7×8 點，X 軸共有 7×40 格 = 280 點，Y 軸有 $8 \times 24 = 192$ 點，前面曾經提過 6522 每一 PORT 內有 8bit I/O，只用了 7bit，就是要配合螢幕上 7bit 為一小單位，雖然浪費了一個輸出，若能使程式好寫也是值得。

螢幕資料與顯示板資料轉換

16

\$6000	\$6010	\$6021		
\$6001	\$6011			
\$6002	\$6012			
\$6003	\$6013			
\$6004	\$6014			
\$6005	\$6015			
\$6006	\$6016			
\$6007	\$6017			
\$6008	\$6018			
\$6009	\$6019			
\$600A	\$601A			
\$600B	\$601B			
\$600C	\$601C			
\$600D	\$601D			
\$600E	\$601E			
\$600F	\$601F	\$602F		

7

7

7

7

** 顯示板格式 **

資料從位址 \$6000 開始存放，最大到 \$8FFF 為止。

16

\$4000	\$4001	\$4002		
\$4400	\$4401			
\$4800	\$4801			
\$4C00	\$4C01			
\$5000	\$5001			
\$5400	\$5401			
\$5800	\$5801			
\$5C00	\$5C01			
\$4080	\$4081			
\$4480	\$4481			
\$4880	\$4881			
\$4C80	\$4C81			
\$5080	\$5081			
\$5480	\$5481			
\$5880	\$5881			
\$5C80	\$5C81	\$5C82		

7

7

7

7

** 螢幕格式 **

二種格式透過程式的轉換，不難將後者轉到前者。

我們了解了二種格式後，另外還有一個問題：漢卡顯示中文，行與行空了二小點，第一行與頂端也空了一點如下圖(→)，

但爲了配合格式，最好這些距離都不存在，因爲我對漢卡 0.S 不十分了解，也無參考資料，所以只有從螢幕上動手脚，在轉換前先將這些空格消掉如圖(←)。

** 活動廣告看板 ** 螢幕規格：16 X 70 點 利用 APPLE II 微電腦 以六個 VIA-6522 來控制 可無限制擴大 輸入方式：以漢卡 或 直接繪圖 輸入 顯示方式：左移,右移,上移,下移.....等 千變萬化,用途廣泛 對本文有任何問題 請洽 台南縣仁德鄉成功村 278 號 黃志凱	** 活動廣告看板 ** 螢幕規格：16 X 70 點 利用 APPLE II 微電腦 以六個 VIA-6522 來控制 可無限制擴大 輸入方式：以漢卡 或 直接繪圖 輸入 顯示方式：左移,右移,上移,下移.....等 千變萬化,用途廣泛 對本文有任何問題 請洽 台南縣仁德鄉成功村 278 號 黃志凱
---	---

程式說明

控制程式分成三部份：程式(一)、(二)、(三) 程式(一)是 BASIC 程式，(二)、(三)則是 ASSEMBLY LANGUAGE。程式(二)BS-AVE PROGRAM2，A\$300，L\$D0，程式(三)BSAVE PROGRAM，A\$1500，L\$AFF。

活動廣告看板

(1) 編輯	(2) 存檔
(3) 取檔	(4) 修飾
(5) 修改	(6) 轉換
(7) 目錄	(8) 開始顯示
(9) 重新轉換	(0) 長度設定

請選擇項目 0--9?_

- (1)編輯：清除現有的資料，進行一個畫面的編輯。
- (2)存檔：存取編輯後的檔案，存取完畢
- (3)取檔：按任一鍵回到主菜單。
- (4)修飾：畫漢卡不能產生的字型，也可作局部修改。

I
 ↑
 J ← → K 控制方向
 ↓
 M

[X]：不畫

[C]：畫

[CTRL-C]：回到主菜單

- (5)修改：更正編輯的畫面，以 **[ESC]** **[I]** **[M]** **[J]** **[K]** 移動游標，更正完畢後將游標移至句子的末端，按 **[CT-**

RL-C] 便可，注意：如不移至句末，游標以下的任何圖形會被清除。

- (6)轉換：將螢幕資料轉換成顯示板資料，最多可轉換 2 個畫面。

- (7)目錄：顯示磁片上的檔案。

- (8)開始顯示：以 **[→]** **[←]** 加速、減速，**[R]**：重頭顯示，**[S]** 暫停，**[ESC]** 回主菜單。

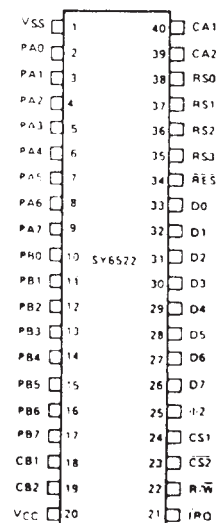
- (9)重新轉換：將轉換的指標重頭開始。

- (0)長度設定：設定顯示資料長度。

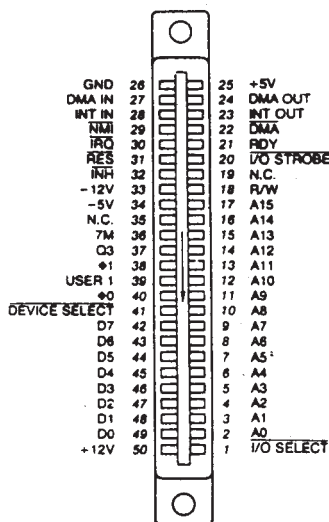
註：漢卡在 SLOT 3，I/O 界面在 SLOT 2。

這個程式是將字形，由右至左移動，若要其他動作，不需改變硬體部份，用軟體控制即可，這種看板可說是千變萬化，不能一一介紹，有興趣的讀者不妨自己寫寫不同的變化程式。

PIN CONFIGURATION



6522 的接腳



APPLE SOFT 接腳圖

本製作所需材料：

- | | |
|------------------|---------|
| (1) APPLE II 微電腦 | (48K) |
| (2) 磁碟機一台 | (5 ¼ 吋) |
| (3) 漢卡一片 | (一般漢卡) |
| (4) 6522 | × 6 |
| (5) 74LS 05 | × 1 |
| (6) 74LS 138 | × 1 |
| (7) 1K 電阻 | × 2 |
| (8) 4.7K 電阻 | × 2 |
| (9) 10P 電容 | × 1 |
| (10) LED | × 1120 |
| (11) 萬用板 (界面) | × 1 |
| (12) IC 萬用板 | 少許 |
| (13) 電晶體 2SC2236 | × 172 個 |
| (14) 排線 | 少許 |
| (15) 5V 電源 | × 1 |
| (16) 外殼 | × 1 |

以上概略地介紹 16 × 70 點廣告看板
的原理和實作，因成稿匆促，程式應可再
精簡些，為免佔篇幅，組合語言部分僅以

機械碼印出，看起來可能費力一點，請見
諒，最後祝製作愉快！

張

程式一

```

1 HIMEM: 5300: PRA 3: PRINT " : POKE 43683,3: POKE 43684,195: POKE 43685,48: POKE 43686,1
95: POKE 54,189: POKE 55,158: POKE 56,129: POKE 57,158: FOR C = 1 TO 15: GET A$: NEXT D =
117: E = 0: F = 3: G = 49152 + 256 * F: H = G + 21: I = G + 24: J = G + 36: K = G + 39: L = G + 4
2
2 M = 768: N = 5376: O = 6618: P = 7222: Q = 7338: R = 7349: S = 7399: PRINT CHR$ (13): BLOAD P
ROGRAM, A$1500: PRINT CHR$ (13): BLOAD PROGRAM, A$300
3 CALL H: PRINT " 活動廣告看板 " : PRINT " : PRINT " (1) 編輯 (2)
存檔: PRINT " (3) 取檔 (4) 修飾: PRINT " (5) 修改 (6) 轉換
4 PRINT " (7) 目錄 (8) 開始顯示: PRINT " (9) 重新轉換 (10) 長度設定: PRINT "
: PRINT " : CALL I: POKE 215,9: POKE 214,2: PRINT "請選擇項目 0-9?": GET A$: IF A$ = "1
" THEN GOSUB 15

```

```

5 IF A$ = "2" THEN GOSUB 16
6 IF A$ = "3" THEN GOSUB 18
7 IF A$ = "4" THEN GOSUB 19
8 IF A$ = "5" THEN GOSUB 55
9 IF A$ = "6" THEN GOSUB 44
10 IF A$ = "7" THEN GOSUB 43
11 IF A$ = "8" THEN GOSUB 52
12 IF A$ = "9" THEN GOSUB 56
13 IF A$ = "0" THEN GOSUB 63
14 GOTO 3
15 CALL H: CALL M: GOSUB 62: RETURN
16 CALL H: PRINT "存檔": PRINT " ": PRINT " ": PRINT " ": INPUT "請輸入檔案名稱?"; C$: PO
KE - 16300,0: POKE - 16297,0: POKE - 16302,0: POKE - 16304,0: PRINT CHR$(13): PRINT
CHR$(4); "SAVE "; C$; ",A$2000,L$1FFF": GET F$
17 POKE - 16299,0: POKE - 16297,0: POKE - 16302,0: POKE - 16304,0: RETURN
18 CALL H: PRINT "取檔": PRINT " ": PRINT " ": PRINT " ": INPUT "請輸入檔案名稱?"; C$: PO
KE - 16300,0: POKE - 16297,0: POKE - 16302,0: POKE - 16304,0: PRINT CHR$(13): PRINT
CHR$(4); "LOAD "; C$; ",A$2000": GET F$: POKE - 16299,0: POKE - 16297,0: POKE - 16302,
0: POKE - 16304,0: RETURN
19 POKE - 16300,0: POKE - 16297,0: POKE - 16302,0: POKE - 16304,0: POKE 229,0: POKE 2
30,32: T = 1: U = 1: HPLLOT T,U: HCOLOR= 3: GOTO 31
20 GET A$: IF A$ = "I" THEN U = U - 1
21 IF A$ = "M" THEN U = U + 1
22 IF A$ = "J" THEN T = T - 1
23 IF A$ = "K" THEN T = T + 1
24 IF A$ = "X" THEN 31
25 IF PEEK ( - 16304) = 3 THEN 42
26 IF T > 279 THEN T = 279
27 IF T < 0 THEN T = 0
28 IF U < 0 THEN U = 0
29 IF U > 191 THEN U = 191
30 HPLLOT T,U: GOTO 20
31 A = T: B = U: GET B$: IF B$ = "I" THEN U = U - 1

```

```

32 IF B$ = "M" THEN U = U + 1
33 IF B$ = "J" THEN T = T - 1
34 IF B$ = "K" THEN T = T + 1
35 IF B$ = "C" THEN 20
36 IF PEEK (- 16304) = 3 THEN 42
37 IF T > 279 THEN T = 279
38 IF T < 0 THEN T = 0
39 IF U < 0 THEN U = 0
40 IF U > 191 THEN U = 191
41 HCOLOR= 0: HPL0T A,B: HCOLOR= 3: HPL0T T,U: GOTO 31
42 POKE - 16288,0: POKE - 16302,0: RETURN
43 CALL H: PRINT CHR$(13): PRINT CHR$(4): "CATALOG": PRINT " ": PRINT " 按任意鍵
"; GET F$: RETURN
44 E = E + 1: IF E = 3 THEN 50
45 CALL H: POKE 215,4: PRINT "已經轉換了";E - 1;"筆 現轉換第";E;"筆": CALL I
46 POKE 214,0: POKE 215,6: PRINT "確定要轉換嗎?"; GET D$: CALL I: IF D$ = "Y" THEN 49
47 IF D$ = "N" THEN 51
48 GOTO 46
49 CALL R: CALL S: CALL D: GOSUB 62: RETURN
50 E = 2: CALL H: CALL I: POKE 215,4: POKE 214,0: PRINT "資料已滿 最多 2 筆"; GET D$:
CALL I: GOSUB 62: RETURN
51 GOSUB 62:E = E - 1: RETURN
52 CALL H: CALL I: POKE 215,4: PRINT " 按任意鍵開始顯示"; GET E$: CALL I: P
RINT " ": CALL I: POKE 215,7: PRINT "<ESC> 停止 (R) 從頭顯示"
53 POKE 215,8: PRINT "<-,> 減速,加速 (S) 暫停": CALL I: POKE 215,4: PRINT "
": CALL I: SPEED= 100: CALL I: POKE 215,4: PRINT " 資料
顯示中 ": CALL I: SPEED= 255: POKE 8,240: POKE 7,0: CALL N: GOSUB 62
54 POKE 1400 + F,0: RETURN
55 POKE 214,0: CALL R: CALL M: GOSUB 62: RETURN
56 CALL H: CALL I: POKE 215,2: POKE 214,0: PRINT "現有資料";E;"筆": CALL I
57 POKE 215,4: POKE 214,0: PRINT "確定重新開始嗎?"; GET G$: CALL I: IF G$ = "Y" THEN 61
58 IF G$ = "N" THEN 60
59 GOTO 57

```

60 RETURN

61 E = 0: CALL P: POKE 215,0: POKE 214,0: PRINT "資料指標重新開始": CALL I: FOR C = 1 TO 1

600: NEXT C: RETURN

62 POKE 40286,30: POKE 40287,176: CALL 40383: POKE 40286,60: POKE 40287,212: RETURN

63 CALL H: INPUT "請輸入長度? (0-255)";D: CALL I: RETURN

程式二

*300.3CF

```
0300: A9 00 85 D7 20 0C FD 8D
0308: 2A 03 C9 83 F0 13 A5 D7
0310: C9 09 D0 04 A9 08 85 D7
0318: AD 2A 03 20 ED FD 4C 04
0320: 03 20 18 C3 20 2A C3 4C
0328: AA 1C EA A9 FF 8D 02 C2
0330: 8D 03 C2 8D 12 C2 8D 13
0338: C2 8D 22 C2 8D 23 C2 8D
0340: 32 C2 8D 33 C2 8D 42 C2
0348: 8D 43 C2 8D 52 C2 8D 53
0350: C2 60 A9 00 8D 01 C2 8D
0358: 00 C2 8D 11 C2 8D 10 C2
0360: 8D 21 C2 8D 20 C2 8D 31
0368: C2 8D 30 C2 8D 41 C2 8D
0370: 40 C2 60 A9 60 8D A8 15
0378: 8D AE 15 8D B4 15 8D BA
0380: 15 8D C0 15 8D C6 15 8D
0388: CC 15 8D D2 15 8D D8 15
0390: 8D DE 15 8D E4 15 A9 00
0398: 3D A7 15 A9 10 8D AD 15
03A0: A9 20 8D B3 15 A9 30 8D
03A8: B9 15 A9 40 8D BF 15 A9
03B0: 50 8D C5 15 A9 60 8D CB
03B8: 15 A9 70 8D D1 15 A9 80
03C0: 8D D7 15 A9 90 8D DD 15
03C8: A9 A0 8D E3 15 60 1F 39
```

```
1580: 86 00 A5 01 C9 01 F0 0B
1588: C9 02 F0 0C A9 01 85 01
1590: A6 00 60 A2 51 4C 9A 15
1598: A2 50 A5 02 9D 00 C2 86
15A0: 03 A6 00 60 A2 00 BD 00
15A8: 60 9D 50 04 BD 10 60 9D
15B0: 60 04 BD 20 60 9D 70 04
15B8: 8D 30 60 9D 80 04 BD 40
15C0: 60 9D 90 04 BD 50 60 9D
15C8: A0 04 BD 60 60 9D B0 04
15D0: BD 70 60 9D C0 04 BD 80
15D8: 60 9D D0 04 BD 90 60 9D
15E0: E0 04 BD A0 60 9D F0 04
15E8: E8 E0 10 D0 B9 60 BD 50
15F0: 04 8D 01 C2 BD 60 04 BD
15F8: 00 C2 BD 70 04 BD 11 C2
1600: BD 80 04 8D 10 C2 BD 90
1608: 04 BD 21 C2 BD A0 04 BD
1610: 20 C2 BD B0 04 BD 31 C2
1618: BD C0 04 8D 30 C2 BD D0
1620: 04 8D 41 C2 BD E0 04 BD
1628: 40 C2 60 5E F0 04 90 22
1630: BD E0 04 85 09 46 09 90
1638: 04 A9 01 85 06 18 A9 40
1640: 65 09 85 09 A5 09 9D E0
1648: 04 A5 06 C9 01 F0 08 4C
1650: 7D 16 5E E0 04 90 26 A9
1658: 00 85 06 BD D0 04 85 09
1660: 46 09 90 04 A9 01 85 06
1668: 18 A9 40 65 09 85 09 A5
1670: 09 9D D0 04 A5 06 C9 01
1678: F0 08 4C A8 16 5E D0 04
1680: 90 26 A9 00 85 06 BD C0
1688: 04 85 09 46 09 90 04 A9
1690: 01 85 06 18 A9 40 65 09
1698: 85 09 A5 09 9D C0 04 A5
16A0: 06 C9 01 F0 08 4C D3 16
16A8: 5E C0 04 90 26 A9 00 85
16B0: 06 BD B0 04 85 09 46 09
16B8: 90 04 A9 01 85 06 18 A9
16C0: 40 65 09 85 09 A5 09 9D
16C8: B0 04 A5 06 C9 01 F0 08
16D0: 4C FE 16 5E B0 04 90 26
16D8: A9 00 85 06 BD A0 04 85
16E0: 09 46 09 90 04 A9 01 85
16E8: 06 18 A9 40 65 09 85 09
16F0: A5 09 9D A0 04 A5 06 C9
16F8: 01 F0 08 4C 29 17 5E A0
1700: 04 90 26 A9 00 85 06 BD
1708: 90 04 85 09 46 09 90 04
1710: A9 01 85 06 18 A9 40 65
1718: 09 85 09 A5 09 9D 90 04
1720: A5 06 C9 01 F0 08 4C 54
```

程式三

*1500.1EAF

```
1500: A9 06 85 E3 20 52 15 20
1508: 2B 03 20 77 15 2C 10 C0
1510: 20 73 03 20 A4 15 A0 00
1518: 20 64 18 AD 00 C0 C9 9B
1520: F0 2A C9 D2 F0 E1 C9 8B
1528: F0 45 C9 95 F0 2F A5 E3
1530: 8D C0 18 C8 C0 07 D0 E0
1538: 20 CD 17 AD A8 15 C9 90
1540: D0 07 AD A7 15 C9 00 F0
1548: BE 4C 13 15 2C 10 C0 4C
1550: 77 15 A5 07 8D 3F 15 A5
1558: 08 8D 46 15 60 C6 E3 A5
1560: E3 C9 00 D0 04 A9 01 85
1568: E3 2C 10 C0 4C 2E 15 E6
1570: E3 2C 10 C0 4C 2E 15 A9
1578: 00 8D 51 C2 8D 50 C2 60
```

1728: 17 5E 90 04 90 26 A9 00
 1730: 85 06 BD 80 04 85 09 46
 1738: 09 90 04 A9 01 85 06 18
 1740: A9 40 65 09 85 09 A5 09
 1748: 9D 80 04 A5 06 C9 01 F0
 1750: 08 4C 7F 17 5E 80 04 90
 1758: 26 A9 00 85 06 BD 70 04
 1760: 85 09 46 09 90 04 A9 01
 1768: 85 06 18 A9 40 65 09 85
 1770: 09 A5 09 9D 70 04 A5 06
 1778: C9 01 F0 08 4C AA 17 5E
 1780: 70 04 90 26 A9 00 85 06
 1788: BD 60 04 85 09 46 09 90
 1790: 04 A9 01 85 06 18 A9 40
 1798: 65 09 85 09 A5 09 9D 60
 17A0: 04 A5 06 C9 01 F0 08 4C
 17A8: C9 17 5E 60 04 90 1A A9
 17B0: 00 85 06 BD 50 04 85 09
 17B8: 46 09 18 A9 40 65 09 85
 17C0: 09 A5 09 9D 50 04 4C CC
 17C8: 17 5E 50 04 60 A0 00 EE
 17D0: A7 15 EE AD 15 EE B3 15
 17D8: EE B9 15 EE BF 15 EE C5
 17E0: 15 EE CB 15 EE D1 15 EE
 17E8: D7 15 EE DD 15 EE E3 15
 17F0: CB C0 10 D0 DA AD A7 15
 17F8: C9 00 D0 03 EE A8 15 AD
 1800: AD 15 C9 00 D0 03 EE AE
 1808: 15 AD B3 15 C9 00 D0 03
 1810: EE B4 15 AD B9 15 C9 00
 1818: D0 03 EE BA 15 AD BF 15
 1820: C9 00 D0 03 EE C0 15 AD
 1828: C5 15 C9 00 D0 03 EE C6
 1830: 15 AD CB 15 C9 00 D0 03
 1838: EE CC 15 AD D1 15 C9 00
 1840: D0 03 EE D2 15 AD D7 15
 1848: C9 00 D0 03 EE D8 15 AD
 1850: DD 15 C9 00 D0 03 EE DE
 1858: 15 AD E3 15 C9 00 D0 03
 1860: EE E4 15 60 20 77 15 85
 1868: 04 A9 01 85 02 85 01 A2
 1870: 00 20 C7 18 20 EE 15 20
 1878: 80 15 20 2B 16 A9 11 20
 1880: A8 FC 20 77 15 18 26 02
 1888: A5 02 C9 00 D0 1B 86 05
 1890: A9 01 85 02 A9 00 A6 03
 1898: BD 00 C2 A6 05 E6 01 A5
 18A0: 01 C9 03 D0 04 A9 01 85
 18A8: 01 20 0A 19 20 4D 19 E8
 18B0: E0 10 D0 BD AD 00 C0 C9
 18B8: D3 F0 02 E6 04 A5 04 C9
 18C0: 06 D0 A6 20 90 19 60 BD
 18C8: 50 04 9D 50 05 BD 60 04
 18D0: 9D 60 05 BD 70 04 9D 70
 18D8: 05 BD 80 04 9D 80 05 BD
 18E0: 90 04 9D 90 05 BD A0 04
 18E8: 9D A0 05 BD B0 04 9D B0
 18F0: 05 BD C0 04 9D C0 05 BD
 18F8: D0 04 9D D0 05 BD E0 04
 1900: 9D E0 05 BD F0 04 9D F0
 1908: 05 60 BD 50 04 9D 50 06
 1910: BD 60 04 9D 60 06 BD 70
 1918: 04 9D 70 06 BD 80 04 9D
 1920: 80 06 BD 90 04 9D 90 06
 1928: BD A0 04 9D A0 06 BD B0
 1930: 04 9D B0 06 BD C0 04 9D
 1938: C0 06 BD D0 04 9D D0 06
 1940: BD E0 04 9D E0 06 BD F0
 1948: 04 9D F0 06 60 BD 50 05
 1950: 9D 50 04 BD 60 05 9D 60
 1958: 04 BD 70 05 9D 70 04 BD
 1960: 80 05 9D 80 04 BD 90 05

1968: 9D 90 04 BD A0 05 9D A0
 1970: 04 BD B0 05 9D B0 04 BD
 1978: C0 05 9D C0 04 BD D0 05
 1980: 9D D0 04 BD E0 05 9D E0
 1988: 04 BD F0 05 9D F0 04 60
 1990: A2 00 BD 50 06 9D 50 04
 1998: BD 60 06 9D 60 04 BD 70
 19A0: 06 9D 70 04 BD 80 06 9D
 19A8: 80 04 BD 90 06 9D 90 04
 19B0: BD A0 06 9D A0 04 BD B0
 19B8: 06 9D B0 04 BD C0 06 9D
 19C0: C0 04 BD D0 06 9D D0 04
 19C8: BD E0 06 9D E0 04 BD F0
 19D0: 06 9D F0 04 E8 E0 10 D0
 19D8: B9 60 20 7A 1B 20 EF 19
 19E0: 20 91 1B 20 EF 19 20 AB
 19E8: 1B 20 EF 19 4C 00 C3 A9
 19F0: 00 85 00 A2 00 EA BD 00
 19F8: 40 BD 00 60 BD 00 44 BD
 1A00: 01 60 BD 00 48 BD 02 60
 1A08: BD 00 4C BD 00 60 BD 00
 1A10: 50 BD 04 60 BD 00 54 BD
 1A18: 05 60 BD 00 58 BD 06 60
 1A20: BD 00 5C BD 07 60 BD 80
 1A28: 40 BD 08 60 BD 80 44 BD
 1A30: 09 60 BD 80 48 BD 0A 60
 1A38: BD 80 4C BD 0B 60 BD 80
 1A40: 50 BD 0C 60 BD 80 54 BD
 1A48: 0D 60 BD 80 58 BD 0E 60
 1A50: BD 80 5C BD 0F 60 20 71
 1A58: 1A E8 E0 27 D0 97 20 49
 1A60: 1B E6 00 A5 00 C9 04 F0
 1A68: 03 4C F3 19 A9 00 85 00
 1A70: 60 A0 00 EE FA 19 EE 00
 1A78: 1A EE 06 1A EE 0C 1A EE
 1A80: 12 1A EE 1B 1A EE 1E 1A
 1A88: EE 24 1A EE 2A 1A EE 30
 1A90: 1A EE 36 1A EE 3C 1A EE
 1A98: 42 1A EE 4B 1A EE 4E 1A
 1AA0: EE 54 1A CB C0 10 D0 CB
 1AA8: AD FA 19 C9 00 D0 03 EE
 1AB0: FB 19 AD 00 1A C9 01 D0
 1AB8: 03 EE 01 1A AD 06 1A C9
 1AC0: 02 D0 03 EE 07 1A AD 0C
 1AC8: 1A C9 03 D0 03 EE 0D 1A
 1AD0: AD 12 1A C9 04 D0 03 EE
 1AD8: 13 1A AD 1B 1A C9 05 D0
 1AE0: 03 EE 19 1A AD 1E 1A C9
 1AE8: 06 D0 03 EE 1F 1A AD 24
 1AF0: 1A C9 07 D0 03 EE 25 1A
 1AF8: AD 2A 1A C9 08 D0 03 EE
 1B00: 2B 1A AD 30 1A C9 09 D0
 1B08: 03 EE 31 1A AD 36 1A C9
 1B10: 0A D0 03 EE 37 1A AD 3C
 1B18: 1A C9 0B D0 03 EE 3D 1A
 1B20: AD 42 1A C9 0C D0 03 EE
 1B28: 43 1A AD 4B 1A C9 0D D0
 1B30: 03 EE 49 1A AD 4E 1A C9
 1B38: 0E D0 03 EE 4F 1A AD 54
 1B40: 1A C9 0F D0 03 EE 55 1A
 1B48: 60 EE FB 19 EE FE 19 EE
 1B50: 04 1A EE 0A 1A EE 10 1A
 1B58: EE 16 1A EE 1C 1A EE 22
 1B60: 1A EE 2B 1A EE 2E 1A EE
 1B68: 34 1A EE 3A 1A EE 40 1A
 1B70: EE 46 1A EE 4C 1A EE 52
 1B78: 1A 60 A9 00 85 01 20 BF
 1B80: 1B A9 80 85 02 20 DA 1B
 1B88: 20 F5 1B A9 04 BD 66 1A
 1B90: 60 A9 2B 85 01 20 BF 1B
 1B98: A9 A8 85 02 20 DA 1B 20
 1BA0: F5 1B A9 04 BD 66 1A 60

```

1BA8: A9 50 85 01 20 BF 1B A9
1BB0: D0 85 02 20 DA 1B 20 F5
1BB8: 1B A9 01 8D 66 1A 60 A5
1BC0: 01 8D F7 19 8D FD 19 8D
1BC8: 03 1A 8D 09 1A 8D 0F 1A
1BD0: 8D 15 1A 8D 1B 1A 8D 21
1BD8: 1A 60 A5 02 8D 27 1A 8D
1BE0: 2D 1A 8D 33 1A 8D 39 1A
1BE8: 8D 3F 1A 8D 45 1A 8D 4B
1BF0: 1A 8D 51 1A 60 A9 40 8D
1BF8: F8 19 8D 28 1A A9 44 8D
1C00: FE 19 8D 2E 1A A9 48 8D
1C08: 04 1A 8D 34 1A A9 4C 8D
1C10: 0A 1A 8D 3A 1A A9 50 8D
1C18: 10 1A 8D 40 1A A9 54 8D
1C20: 16 1A 8D 46 1A A9 58 8D
1C28: 1C 1A 8D 4C 1A A9 5C 8D
1C30: 22 1A 8D 52 1A 60 A9 60
1C38: 8D FB 19 8D 01 1A 8D 07
1C40: 1A 8D 0D 1A 8D 13 1A 8D
1C48: 19 1A 8D 1F 1A 8D 25 1A
1C50: 8D 2B 1A 8D 31 1A 8D 37
1C58: 1A 8D 3D 1A 8D 43 1A 8D
1C60: 49 1A 8D 4F 1A 8D 55 1A
1C68: A2 00 8E FA 19 E8 8E 00
1C70: 1A E8 8E 06 1A E8 8E 0C
1C78: 1A E8 8E 12 1A E8 8E 18
1C80: 1A E8 8E 1E 1A E8 8E 24
1C88: 1A E8 8E 2A 1A E8 8E 30
1C90: 1A E8 8E 36 1A E8 8E 3C
1C98: 1A E8 8E 42 1A E8 8E 48
1CA0: 1A E8 8E 4E 1A E8 8E 54
1CA8: 1A 60 A9 40 85 00 A9 20
1CB0: 85 01 4C C0 1C A9 20 85
1CB8: 00 A9 40 85 01 4C C0 1C
1CC0: A9 00 85 04 85 02 A5 00
1CC8: 85 05 A5 01 85 03 A2 20
1CD0: A0 00 B1 04 91 02 C8 D0
1CD8: F9 E6 05 E6 03 CA D0 F2
1CE0: 60 A9 02 85 00 A2 10 8E
1CE8: 01 20 FA 1C A9 10 18 65
1CF0: 01 85 01 A5 01 C9 C0 D0
1CF8: F0 60 A5 00 85 02 A6 01
1D00: BD F0 1D 85 05 BD 30 1D
1D08: 85 06 E8 BD F0 1D 85 03
1D10: BD 30 1D 85 04 A0 27 B1
1D18: 03 91 05 88 10 F9 E0 BF
1D20: D0 DE A9 00 A0 27 91 03
1D28: 88 10 FB C6 02 D0 CF 60

```

```

1D30: 40 44 48 4C 50 54 58 5C
1D38: 40 44 48 4C 50 54 58 5C
1D40: 41 45 49 4D 51 55 59 5D
1D48: 41 45 49 4D 51 55 59 5D
1D50: 42 46 4A 4E 52 56 5A 5E
1D58: 42 46 4A 4E 52 56 5A 5E
1D60: 43 47 4B 4F 53 57 5B 5F
1D68: 43 47 4B 4F 53 57 5B 5F
1D70: 40 44 48 4C 50 54 58 5C
1D78: 40 44 48 4C 50 54 58 5C
1D80: 41 45 49 4D 51 55 59 5D
1D88: 41 45 49 4D 51 55 59 5D
1D90: 42 46 4A 4E 52 56 5A 5E
1D98: 42 46 4A 4E 52 56 5A 5E
1DA0: 43 47 4B 4F 53 57 5B 5F
1DA8: 43 47 4B 4F 53 57 5B 5F
1DB0: 40 44 48 4C 50 54 58 5C
1DB8: 40 44 48 4C 50 54 58 5C
1DC0: 41 45 49 4D 51 55 59 5D
1DC8: 41 45 49 4D 51 55 59 5D
1DD0: 42 46 4A 4E 52 56 5A 5E
1DD8: 42 46 4A 4E 52 56 5A 5E
1DE0: 43 47 4B 4F 53 57 5B 5F
1DE8: 43 47 4B 4F 53 57 5B 5F
1DF0: 00 00 00 00 00 00 00 00
1DF8: 80 80 80 80 80 80 80 80
1E00: 00 00 00 00 00 00 00 00
1E08: 80 80 80 80 80 80 80 80
1E10: 00 00 00 00 00 00 00 00
1E18: 80 80 80 80 80 80 80 80
1E20: 00 00 00 00 00 00 00 00
1E28: 80 80 80 80 80 80 80 80
1E30: 28 28 28 28 28 28 28 28
1E38: A8 A8 A8 A8 A8 A8 A8 A8
1E40: 28 28 28 28 28 28 28 28
1E48: A8 A8 A8 A8 A8 A8 A8 A8
1E50: 28 28 28 28 28 28 28 28
1E58: A8 A8 A8 A8 A8 A8 A8 A8
1E60: 28 28 28 28 28 28 28 28
1E68: A8 A8 A8 A8 A8 A8 A8 A8
1E70: 50 50 50 50 50 50 50 50
1E78: D0 D0 D0 D0 D0 D0 D0 D0
1E80: 50 50 50 50 50 50 50 50
1E88: D0 D0 D0 D0 D0 D0 D0 D0
1E90: 50 50 50 50 50 50 50 50
1E98: D0 D0 D0 D0 D0 D0 D0 D0
1EA0: 50 50 50 50 50 50 50 50
1EA8: D0 D0 D0 D0 D0 D0 D0 D0

```

GAME I/O的應用

遊戲輸入／輸出連接器 (Game I/O connector) 是 APPLE II 主機板上的--個週邊界面連接器。這個 Game I/O connector 是位於 APPLE II 主機板上鍵盤的右後方。它通常是用來連接遊戲搖桿 (Game paddles) 以作為遊戲控制用。除此之外，它還保留有四個輸出埠 (annunicator output) AN0 ~ AN3，可以作為其它控制用。

遊戲輸入／輸出連接器的接腳圖，可以由 APPLE II 參考手冊知其接腳形式如下：

+5v	1	16	NC
PB0	2	15	AN0
PB1	3	14	AN1
PB2	4	13	AN2
STROBE	5	12	AN3
GC0	6	11	GC3
GC2	7	10	GC1
Gnd	8	9	NC

接腳功能說明：PB=按鍵輸入

GC=遊戲搖桿輸入

AN=控制輸出

NC=不連接 (空腳)

GND=接地

+5V=電源

注意：在上圖中腳1是由主機板上鍵盤的後方看過去，遊戲輸入／輸出連接器上的右前方第一腳。

當遊戲搖桿連接到插座上時，僅有 GC0, GC1, PB0 和 PB1 這幾個接腳有用到 (包含 +5V 和 GND 接腳)。由圖上知，除了上述這些腳外，還有其它接腳尚未利用到，這些接腳含有：另外一個按鍵輸入，以及遊戲控制輸入，和四個控制輸出 (annunicator output)。

本文的目的，即在介紹如何應用遊戲輸入／輸出連接器上的四個控制輸出 (annunicator output) 來驅動外部裝置。

很顯然地，在 APPLE II 主機上的這四個控制輸出，可以允許使用者 (user) 利用這四個輸出去驅動一系列的燈泡，以作為顯示裝置。但由於這四個控制輸出的輸出信號就是標準的

74LS 系列之TTL 準位輸出，所以它們也可以用來驅動繼電器（relays），揚聲器和其它裝置。而在本文中就是要介紹如何利用這些輸出埠來推動架段式的LED 顯示裝置，以顯示0～9的數字。

電路之組裝

爲了要完成這個線路，必須具有下列裝備：一個共陽極（common anode）的七段式LED，其編號爲DL707；一個BCD to 7-segment 的解碼器／驅動器，其編號爲7447。另外，還需要7個330Ω 的電阻以及一個麵包板，一些接線和一個16腳的雙排腳跳線排組（DIP jumper cable），以作爲麵包板和遊戲輸入／輸出連接器間之連接用。

當你將上述零件準備好了之後，將跳線排組（jumper cable）的一端插入遊戲輸入／輸出連接器上，且將另一端連接至麵包板上。然後找出麵包板上跳線排組和遊戲輸入／輸出之接腳關係，以便按照圖1所示之線路圖，來組合線路。

接腳說明：

$R=330\Omega$

AN0 是遊戲輸入／輸出連接器的腳15

AN1 是遊戲輸入／輸出連接器的腳14

AN2 是遊戲輸入／輸出連接器的腳13

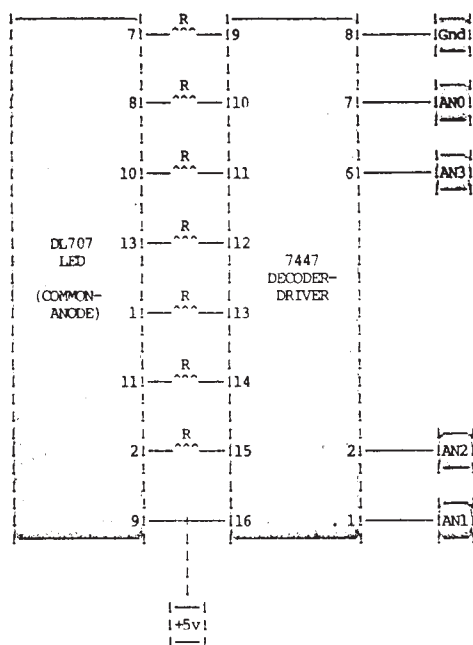
AN3 是遊戲輸入／輸出連接器的腳12

+5V 是遊戲輸入／輸出連接器的腳1

GND 是遊戲輸入／輸出連接器的腳8

電路工作情形

如圖1中所示的SN74LS47 解碼器／驅動器裝置，主要是用來將其輸入端的四個TTL 準位輸入，轉換成7段式的輸出，以驅動DL707（七段式LED 顯示裝置）。DL707所



顯示的數字是隨著7447的四個輸入而改變，因此其顯示變化共有 $2^4 = 16$ 種，即可顯示0～15之任一數字。另外，在圖1中之7個電阻，主要是作爲限流用，以避免在線路動作時，將七段式LED 燒壞。

輸出控制

在嘗試利用遊戲輸入／輸出（AN0～AN3）來控制DL707 的數字顯示之前，首先來討論一下，如何利用機械語言、APPLESOFT 或 INTEGER BASIC 語言，來控制AN0～AN3 這四個輸出的開或關。

對於控制輸出（annunciator output）AN0～AN3中的每一個輸出而言，在APPLE II 64K 的記憶空間中，均有指定到兩個記憶位址，這些特殊的記憶位址，即一般所稱的軟體開關（soft switch），因爲當6502

CPU 參照 (Reference) 到這些控制輸出 (annunciator output) 所分配到的第一個記憶位址時，將會使這些輸出處於關 (OFF) 的狀態。反之，當 6502 CPU 參照到這些控制輸出所分配到的第二個記憶位址時，將會使這些輸出處於開 (ON) 的狀態。當遊戲輸入／輸出 (ANO～AN3) 處於 OFF 狀態時，其接腳上的電壓是接近 0 V，即處於“L”狀態。反之，當其處於 ON 狀態時，其接腳上的電壓是接近 +5 V，即處於“H”狀態。

下表所示就是 6502 CPU 參照到的記憶位址和控制輸出 AN0～AN3 的開／關情形。

ADDRESS			
AN#	STATE	Decimal	Hex
0	off	-16296	\$C058
	on	-16295	\$C059
1	off	-16294	\$C05A
	on	-16293	\$C05B
2	off	-16292	\$C05C
	on	-16291	\$C05D
3	off	-16290	\$C05E
	on	-16289	\$C05F

一般而言，在 Integer 或 Applesoft BASIC 中，可以使用 $\text{POKE}-16296+2 \times N, 0$ 來使 AN#N (N=0, 1, 2, 3) 處於 OFF 的狀態。反之若利用 $\text{POKE}-16295+2 \times N, 0$ ，則可使 AN#N 處於開 (ON) 的狀態。為了寫程式方便且簡化起見，可以設定一個旗號 F 來加以控制。例如，F = 1 則可使 AN#N 處於開的狀態；若 F = 0 則可使 AN#N 處於關的狀態。因此可以將上面兩指述修改成 $\text{POKE} - 16296 + 2 \times N + F, 0$ 的形式，以控制 AN0～AN3 的狀態。

機械語言的控制

假如不用 BASIC 來作控制，也可使用下述組譯程式語言來作控制：

邏輯表

雖然現在我們已知道，如何利用程式來控制 AN0～AN3 的“ON”和“OFF”狀態，但為了實驗方便，我們還需要知道 AN0～AN3 這

```

LDA ANNUN          ; 取得控制輸出的編號 (AN#)
ASL                 ; 乘以 2
ORA STATE           ; 假如“ON”則加 1，假如“OFF”則加 0
TAX
STA $C058,X         ; 觸發輸出
RTS
STATE DS 1           ; IF “ON” 則 STATE = “1”，IF “OFF” 則 STATE = “0”
ANNUN DS 1           ; 控制輸出之編號 (0, 1, 2, 3)

```

四個輸出的狀態和 DL707 所顯示數字的關係。這些關係之邏輯表如下所示：

DIGIT	AN3	AN2	AN1	AN0
ZERO	0	0	0	0
ONE	0	0	0	1
TWO	0	0	1	0
THREE	0	0	1	1
FOUR	0	1	0	0
FIVE	0	1	0	1
SIX	0	1	1	0
SEVEN	0	1	1	1
EIGHT	1	0	0	0
NINE	1	0	0	1

在上表中“0”表示AN是“OFF”狀態，而“1”則表示AN是“ON”狀態，例如：十進制的7其對應的二進制為0111。

程式例

假設要DL707顯示出十進制的4，則參照前面邏輯表知，必須清除AN3,AN1和AN0

之輸出，且設定AN2之輸出，可利用下述BASIC之POKE指述，來完成這件工作：

POKE-16296,0 (清除AN0)

POKE-16294,0 (清除AN1)

POKE-16291,0 (清除AN2)

POKE-16290,0 (清除AN3)

爲了能對這個電路能有更深入的了解，所以在下面再舉出兩個示範程式。第一個程式是控制DL707顯示一個隨機的數字(random digit)，第二個程式則控制DL707顯示一個指定的數字(specific digit)。看完了上面的說明後，下面當然是連接電路且鍵入程式，以開始驗證一下遊戲輸入／輸出的功能。當然，也可以利用這些觀念，去作別的控制。

APPLE II 光筆的製作

利用軟體的設計，加上簡單的硬體，使你的 Apple
具有一支可以用來繪圖或輸入信號的光筆。

—費雲—

你是否看過有人拿著一支筆狀的東西，在螢幕上描了幾下，結果就能令電腦產生音樂、繪製圖形，或執行程式……等，在看了這些動作後，你是否會驚訝於電腦神奇呢？其實這種情形是 Apple 光筆（LIGHT PEN）的一種應用而已！

所謂光筆—LIGHT PEN，主要是允許使用者利用光筆直接由螢幕上將程式或命令輸入電腦，以省却用KEY BOARD鍵入的麻煩。

光筆具有很多吸引人的地方，至於光筆的工作原理也是一個值得探討的話題，所以在本文中我們將告訴你如何利用簡單的硬體，製作一支適用於 Apple 的光筆。在這個製作中爲了簡化硬體的設計，所以大多數的工作都是以軟體來完成。

光筆的工作原理

光筆—LIGHT PEN 的設計，主要是利用一個光感知器以感知從螢光幕發射出來的光，當光筆在螢光幕上某個位置，感知到螢光幕發射出來的光源時，它即會透過本身的硬體裝置，傳送一個電壓信號給電腦，反之，則不會。而另外要特別一提的是，電腦爲了要決定光筆在螢光幕上所處的位置，所以電腦是在某個已知的螢光幕位置繪出一個發光（白色）的方塊（其他方塊則爲黑色），然後再利用程式去檢查光筆的輸入，以判斷光筆是否在此位置上。假如光筆有輸入信號，即表示光筆在螢幕上的位置，就是此時電腦發射出光源的方塊之座標所在。反之若光筆沒有輸入信號，即表示光筆並不在發出光源的方塊座標位置上，電腦即將發光的方塊移到其他座標，以繼續上述步驟。上面所提的這些步驟會不斷地被執行，直到電

腦找到光筆的位置或直到整個螢幕都被搜尋完畢爲止。

光筆的用途

光筆的用途很多，例如，透過光筆在螢光幕上的移動，你可以控制繪圖機畫圖，控制機械臂作一些特殊的動作……但是由於這些工作都需要精密的硬體，配合強有力的軟體控制，所以一般簡單結構的光筆，並無法作這些工作。本文所介紹的光筆，主要可以用在兩方面：第一個是作爲功能選擇—menu selection，即使用者可以利用光筆來選擇螢幕上所列菜單中之某一項功能，這項功能，對於沒有經驗的電腦使用者而言，可以說是相當的方便，因爲透過光筆的這項功能，將可以使使用者輸入錯誤的機會減至最少。

光筆的第二個功能，可能是較重要而且是較常用到的功能，就是它允許使用者直接由螢幕上輸入圖形資料，即經由光筆的控制，使用者可以在螢幕上描繪自己喜歡的圖形，以將圖形資料存入電腦記憶體中。使用這種方法的主要優點是，它沒有利用程式設計圖形時，需計算座標的複雜程序。

光筆在APPLE II 上的操作

爲了能有效的使用光筆，所以首先必須了解 Apple II 的圖形顯示方式。Apple II 具有兩種圖形顯示型式：高解像度（high-res.）圖形顯示及低解像度（low-res.）圖形顯示。這兩種圖形顯示都是採用位元對映（bit-mapped）的方式，即這兩種顯示的影像資料都是存在電腦的某些特定的記憶位址中，低解像度圖形顯示主要是和文字顯示共用 \$400-\$7

FF 的 1K bytes 螢幕記憶區。利用這個記憶區，Apple II 可以產生 40×40 方塊的低解像度圖形，且螢幕的最下方還可以有四行的文字顯示。而高解像度圖形顯示則是占用 \$2000-\$3FFF 的 8K bytes 螢幕記憶區，利用這個螢幕記憶區，Apple 可以顯示 280×160 點矩陣的高解像度圖形顯示。在本文的製作中，當光筆在螢幕上移動時，程式是以高解像度螢幕來作為顯示，但是光筆在螢幕上移動時，信號的感知則是透過低解像度螢幕來完成。

每次電腦要找出光筆在螢幕上的位置時，它必須經由程式的控制，對整個螢幕（低解像度顯示）掃瞄一次，利用這個方法，則可以將光筆置於螢幕上的任何位置，而不會錯過電腦的掃瞄。

爲了要達到上述目的，有人曾經以 Apple soft BASIC 寫成的程式來完成整個螢幕的掃瞄工作，但是要利用這個 BASIC 程式來找出光筆在螢幕上的位置，却必須花掉 10 至 15 秒的時間，因此就速度上來說，實在是太慢了。

爲了要加速搜尋的工作，所以只有嘗試以 6502 組合語言來寫程式，但是一開始即發現組合語言程式之執行速度太快，主要原因是因爲螢幕顯示的更新速率爲 $1/60 \text{ sec}$ （16.7 ms）。而組合語言的執行速度却較這個時間還要短，所以可能在使用者的光筆還沒有來得及移動到螢幕的其它位置時，程式即已完成整個螢幕上光筆的搜尋工作，基於這個原因，電腦可能無法正確的檢知光筆的輸入。而解決這個問題的方法，就是在程式於螢幕上繪出方塊（低解像）及檢知光筆的位置之間，插入一段延遲常式，以允許繪出的方塊在螢幕上有足夠的顯示時間，俾能有效的檢知光筆的位置。

雖然上面所提到插入一段延遲程式的方法很不錯，但不幸的是，能夠符合工作需要的最短延遲程式大約需要 4ms 的時間，因而導致整個螢幕的掃瞄需要 6sec. 的時間。

爲了克服慢速度搜尋時間所造成的問題，因此對程式進行一些修改，也就是程式在搜尋光筆於螢幕的位置時，並不採用整個螢幕掃瞄

的方式，而是採用每次掃瞄螢幕低解像度顯示的 5×5 方塊（25 方塊）的螢幕區，而這種掃瞄總是以前一光筆所在的位置爲中心區，使用此種方法的主要優點是程式不再掃瞄整個低解像度螢幕的 1600（ 40×40 ）個方塊，而僅需 25 方塊的螢幕區，因此原來費時 6sec. 的掃瞄工作，現在僅需 100ms。但是由於這種較低的速率，而使螢幕上的掃瞄區會產生明顯的閃爍現象，可喜的是此種現象尚不致予人不愉快的感覺。

上面提到爲了加速螢幕的掃瞄時間，而採取了 25 方塊的局部螢幕掃瞄方式，這樣一來程式消耗在螢幕上的搜尋時間，雖然是縮短了，但相對的光筆在螢幕上移動的方式也受到了限制，也就是當電腦在搜尋螢幕上的 25 個方塊區時，你的光筆必須是在螢幕上滑動著，如此電腦才不會漏失掉光筆輸入的信號，由這一段的說明可以知道，假如電腦在 25 方塊螢幕掃瞄時，沒有檢知到光筆的存在，則電腦會假設沒有光筆在螢幕上，因此螢幕上將不會有任何變化。總之，在本文程式中所採用的光筆檢知方式，雖然速度上提高了很多，但相對的也造成使用者的光筆在螢幕上必須保持滑動形式的麻煩，也就是在使用光筆繪圖時，光筆必須緊貼著螢幕移動，否則 5×5 個方塊的螢幕掃瞄將無法感知到光筆的存在。

光筆的硬體組成

光筆主要是使用 TTL 位準的信號，經由 Apple 的 game I/O 連接器來將光筆在螢幕上的感知信號傳送給 Apple。爲了要使硬體簡化，光筆的信號是經由遊戲輸入／出（game I/O）的 PB2 輸入至系統中，由 Apple 線路圖上可以看出此輸入信號是直接連到 SN74LS 251 資料選擇器的輸入端，而這個信號的檢知，可以利用程式檢查位置 \$C063 的最高位元（ D_7 ）來完成。

在光筆的組成上，主要是以光電晶體當作基本的感知元件。由圖 1 上可以看到光電晶體的集極是經由 47k Ω 電阻連接到 +5V，射極

接地，而光的輸入即是當作光電晶體的基極驅動。但有一點要注意的是光電晶體本身並無法產生足夠的邏輯位準來驅動輸入，也就是說從螢幕上產生的光源，可能無法使光電晶體完全飽和，以產生一個0的邏輯位準。

一個完整的光筆電路如圖1所示，你可以根據需要將這個線路裝在任何筆形的包裝中。圖1的光電晶體的集極是經由47k Ω 電阻而提昇到+5V，採用這麼大的電阻之主要目的是希望光電晶體只要有極小的基極電流輸入，就能產生夠低的 V_{CE} 電壓以觸發比較器。另外由圖1可以看到比較器的反相輸入端經由100k

Ω 而連接到500k Ω 的電位器去，而此電位器的主要作用就是用來調整光筆感知電路的靈敏度。

在圖1的電路中，當光筆移到方塊的邊緣時，會造成比較器太快關掉（turn-off），而且此時由於程式控制方塊（光源）位置的移動，可能會造成電磁干擾。當方塊移動時，由於光電晶體已不再飽和，因此會使比較器關掉。為了避免光電晶體太快關掉，比較器的輸出被連接到一個單穩（monostable）多諧振盪器的輸入，然後以單穩振盪的輸出來當作實際的輸出。

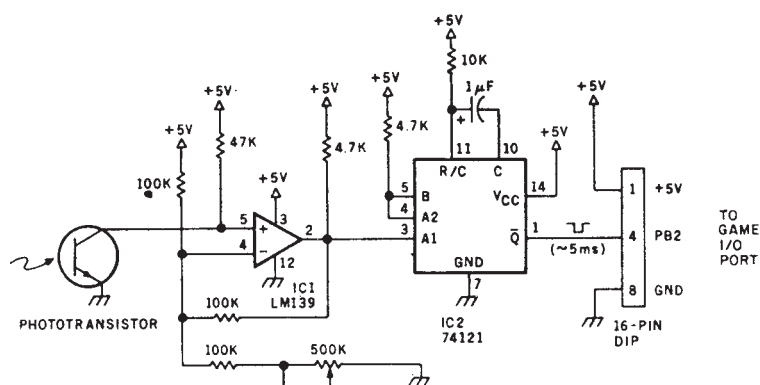


圖1

由上面的說明知道，在平常時圖1的電路是提供PB2一個“H”的輸入，但是當光電晶體在螢幕上檢知到光時，它將提供一個近似於5ms的“L”輸入給PB2。也就是電腦將偵測到一個負向脈衝。

光筆搜尋常式

程式列表1的光筆搜尋常式—PEN Subroutine 主要之功能是搜尋光筆在螢幕上所在的位置。為了縮短程式的執行時間，所以此常式是以6502組合語言寫成。由於Apple本身設計是以第3頁記憶區為機械語言常式之儲存位置，基於這個原因，所以本製作所用到的機械語言常式，也是安排在第3頁記憶區且允許使用者對這些位址加以呼叫（CALL）。

光筆搜尋常式在執行完畢後，會產生水平

和垂直座標且存在記憶體中，以接受程式的使用，此處所提到的水平和垂直座標，其值都是介於0至39之間；（0，0）是表示低解像螢幕左上角的座標而（39，39）則是表示低解像螢幕右下角的座標。由光筆搜尋常式所產生的垂直座標值是被存入記憶位置\$303中，水平座標值則存入\$304中。假如光筆搜尋常式被呼叫時光筆並沒有指在螢幕上，則常式所送出的座標值即是前一光筆所在位置的座標值，也就是說位置\$303和\$304的內含值將保持不變。

正如前面提到的，光筆搜尋常式是以前一光筆所在的位置為基準點，然後搜尋其周圍5×5個方塊的低解像螢幕區，以找出新的光筆位置。由此處的說明可以知道，對螢幕上某一點的光筆座標而言，一個5×5個方塊的低解

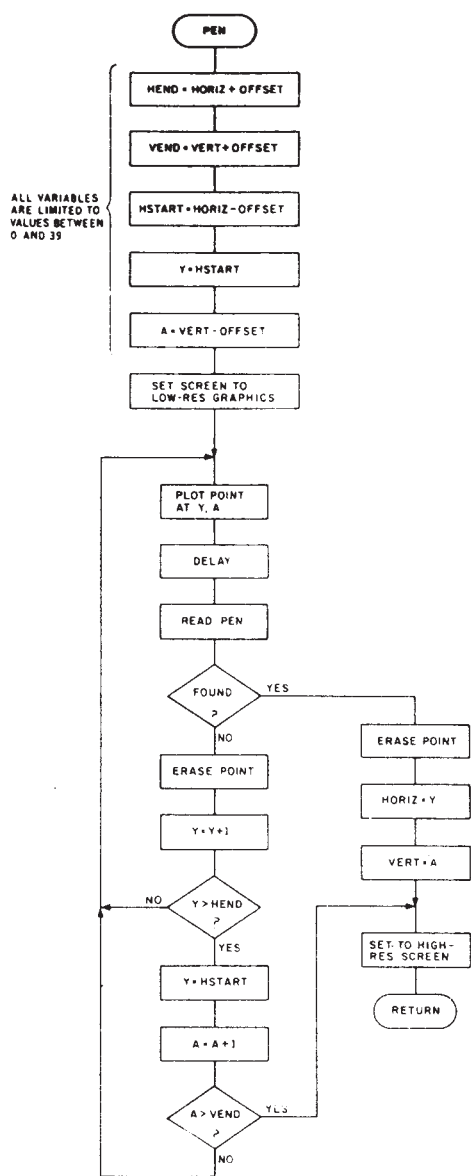


圖 2

像螢幕區，對於螢幕上某一點的光筆座標即相當於水平及垂直方向上 2 個方塊的位移 (OFF SET)。由圖 2 可以知道，光筆搜尋常式首先以前一光筆所在的座標位置 (HORIZ, VERT) 加上 2 個方塊的位移 (OFF SET)，以計算出搜尋常式的極限座標是否超過低解像螢幕區 0 至 39 的範圍。因為程式將極限座標限制在 0

至 39，使得光筆的搜尋不會超出螢幕的範圍。在程式進行時，現在光筆的水平座標值是存在 Y 暫存器中，且垂直座標則是存在累積器 A 中。

由圖 2 可以知道，機械語言搜尋常式在完成上述初始化工作後，即將螢幕設定為低解像顯示 (LDA \$C056)。接著程式即使為 Apple 監督程式中 PLOT 常式在 5 × 5 個方塊的搜尋繪出第一個方塊，然後在大約 4ms 的延遲後，常式即檢查光筆的輸入。假如光筆在螢幕上感知到繪出的方塊，則程式即會將此方塊抹除掉，且重新算出欲繪出方塊的新座標，然後回到呼叫的程式去，假如光筆並沒有感知到方塊，則程式即會將方塊抹除且右移到下一個水平位置，繼續重複此步驟直到第一行方塊的結束為止。接著移到下一行的搜尋方塊重覆上述步驟……此一程序一直重覆，直到光筆被感知到或是整個方塊均已搜尋完畢為止。在上述程序的過程中，將會在螢幕上看到一系列閃爍的方塊隨著光筆在螢幕上移動，當光筆不在螢幕上時，則方塊仍然保持靜止。

上面已將光筆搜尋常式 — PEN subroutine 的功能大致的介紹了一下，這個常式之主要優點是它具有通用性且可以配合應用程式來使用，以找出光筆在螢幕上的位置。但有一點要注意的是，因為 PEN subroutine 是使用低解像度螢幕顯示，僅保留螢幕下方四行作為文字顯示，而這一段螢幕記憶區也是文字顯示所使用的記憶區，因此若有程式其文字顯示超過底下四行的範圍，則程式必須加以修改。

繪圖程式

程式列表 2 的 BASIC 繪圖程式 — LINES program，使你可以利用光筆搜尋常式 (PEN subroutine) 在螢幕上繪出圖形。你藉著光筆移動閃爍的方塊，在螢幕上到處移動以繪出圖形。若有錯誤發生，則會有錯誤信息在螢幕下方的文字顯示區顯示出來。當你將閃爍的方塊 (游標) 移到螢幕上的某個位置時你可以按下空白鍵，以設定此點，則程式即會在設定點的位置顯示一個叉號 (×)，並且將此點座標

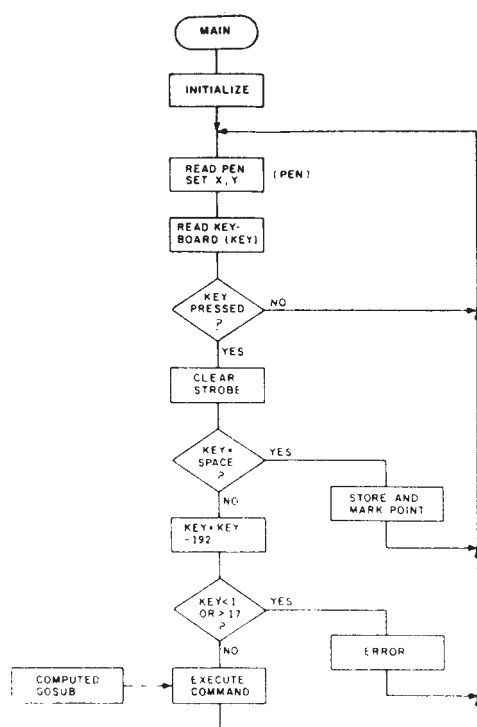


圖 3

流程圖是顯示 LINES program 的工作流程，由流程圖中可以看出光筆繪圖的過程

之數值存在某個陣列中，在利用上述方式設定許多座標點後，你可以按下命令鍵（L 或 P）以將這些設定點用線連接起來。

參見程式列表 2，LINES program 的第一個步驟是先設定各個變數的初值，這個步驟也包含將含有 PEN subroutine 的常式的檔案由磁片載入電腦中，然後程式即呼叫這個副常式以使用線將光筆所設定的各個點連接起來。接著呼叫 L 命令的副常式，且清除陣列（點計數 N 被清除為 0），以讓你開始定義一個新的圖形，假如你所設定的點不超過 3 個，則螢幕下方將會有錯誤信息顯示出來且命令也不被接受。

下面所示是程式列表 2 中各命令之意義：

- L 將螢幕上設定的各個點用線段連接起來

，這個命令和命令 P 一樣，但是有一點不同的即是命令 L 並不會將頭一點和最後一點連接起來，因此它允許你將點與點用線連接起來而不會造成密閉的圖形。但是有一點要注意的是，你必須至少設定兩個點以便連線，否則將會有錯誤信息會顯示出來。

- C 改變顏色。螢幕上將會有一個菜單顯示，它列出隨後連線可能具有的 7 種顏色，且這些顏色的設定，將不會影響先前繪出的線段之顏色。此處所提到的 7 種顏色包含 black, green, blue, white, black2, yellow 和 blue2 等顏色。但這些顏色也可能因你顯像裝置色相調整之不同而有所不同，所以你也可以自己略加調整。
- N 清除螢幕。使用這個命令可以將螢幕清除成黑色且將陣列之內含值重定為 0，也就是說你可以重新開始繪圖而不必再重新執行程式。
- Q 使用這個命令可以終止程式的執行。此外，它也提供了一個選擇，以允許你將高解像的圖形顯示存入磁片中。行號 1750 的 POP 指述是用以清除 STACK 的內含值，主要是因為 Q 命令的執行，使得副常式的 RETURN 指述無法執行之故。
- H 鍵入 H，可以在螢幕上列出所有的命令。

結語

本文所介紹的 Apple LIGHT PEN—光筆的製作，由於大多數的處理程序都是以軟體（程式）來完成，所以使得其硬體（電路）結構變得相當的簡單，且由於在程式中將光筆的搜尋區限制在 5×5 個方塊的區域，使得光筆位置的感知在時間上節省了很多。光筆的主要優點是它提供了一種更方便，有效的方法來將圖形資料輸入給 Apple，使得 Apple 強有力的繪圖能力，獲得更大的發揮。

程式列表 1 光筆搜尋常式之組合語言程式

```

SOURCE FILE: PEN SUBROUTINE
0000:      1 ;SUBROUTINE TO FIND LIGHTPEN          ; 在 5 x 5 個方塊 (BLOCK) 的
0000:      2 ;IN SMALL SQUARE                    ; 低解像螢幕區，搜尋光筆之位
----- NEXT OBJECT FILE NAME IS PEN SUBROUTINE.OBJO 置
0300:      3      ORG    $300                      ; 機械語言之起始位址
0300:4C 09 03      4 FINDPEN JMP    START
0303:      5 *****
0303:      6 *   DEFINE STORAGE                      ; 設定光筆搜尋常式所用到之特
0303:      7 *****                      ; 定記憶位置之初值
0303:00      8 VERT      DFB    0                    ; 現在光筆之垂直座標
0304:00      9 HORIZ    DFB    0                    ; 現在光筆之水平座標
0305:00     10 SAVEA     DFB    0                    ; 累積 A 用以儲存現在光筆所在位置之垂直座標
0306:00     11 HEND      DFB    0                    ; 5 x 5 個方塊搜尋區之水平終點座標
0307:00     12 VEND      DFB    0                    ; 5 x 5 個方塊搜尋區之垂直終點座標
0308:00     13 HSTART    DFB    0                    ; Y 暫存器用以儲存現在光筆所在位置之水平座標
0309:      14 ;
0309:      15 ;DEFINE EQUATES                      ; 定義各標顯之等效值
0309:      16 ;
0002:      17 OFFSET    EQU    2
000F:      18 WHITE     EQU    15
0000:      19 BLACK      EQU    0
FB00:      20 PLOT      EQU    $FB00
FB64:      21 SETCOL    EQU    $FB64
FCAB:      22 WAIT      EQU    $FCAB
0028:      23 DELAY     EQU    40
C063:      24 PEN       EQU    $C063
C056:      25 LORES     EQU    $C056
C057:      26 HIRES     EQU    $C057
C054:      27 PAGE1     EQU    $C054
C050:      28 GRAPHIC   EQU    $C050
FB36:      29 CLRSCR    EQU    $FB36
0309:      30 ;
0309:      31 *****
0309:      32 *   MAIN PROGRAM                      ; 光筆搜尋常式之主程式
0309:      33 *****
0309:      34 ;
0309:20 36 FB      35 START    JSR    CLRSCR
030C:AD 54 C0      36      LDA    PAGE1
030F:      37 ;
030F:      38 ;INIT VARIABLES WITH LIMIT CHECKS ; 設定低解像畫面為最大之光筆
030F:      39 ;                      ; 搜尋範圍。
030F:18      40      CLC
0310:AD 04 03      41      LDA    HORIZ
0313:69 02      42      ADC    #OFFSET
0315:C9 28      43      CMP    #40
0317:90 02      44      BCC    HOK
0319:A9 27      45      LDA    #39
031B:BD 06 03      46 HOK     STA    HEND
031E:18      47      CLC
031F:AD 03 03      48      LDA    VERT
0322:69 02      49      ADC    #OFFSET
0324:C9 28      50      CMP    #40
0326:90 02      51      BCC    VOK
0328:A9 27      52      LDA    #39
032A:BD 07 03      53 VOK     STA    VEND
032D:38      54      SEC
032E:AD 04 03      55      LDA    HORIZ
0331:E9 02      56      SBC    #OFFSET
0333:10 02      57      BPL    H2OK
0335:A9 00      58      LDA    #0
0337:BD 08 03      59 H2OK    STA    HSTART
033A:A8      60      TAY
033B:38      61      SEC
033C:AD 03 03      62      LDA    VERT
033F:E9 02      63      SBC    #OFFSET
0341:10 02      64      BPL    V2OK

```

0343:A9 00	65	LDA #0	
0345:BD 05 03	66	V20K STA SAVEA	
0348:	67	:	
0348:	68	;FIND LIGHTPEN	; 搜尋光筆位置
0348:	69	:	
0348:AD 56 C0	70	AGAIN LDA LORES	
034B:A9 0F	71	LDA #WHITE	
034D:20 64 FB	72	JSR SETCOL	
0350:AD 05 03	73	LDA SAVEA	
0353:20 00 FB	74	JSR PLOT	
0356:A9 28	75	LDA #DELAY	
0358:20 A8 FC	76	JSR WAIT	
035B:AD 63 C0	77	LDA PEN	
035E:10 29	78	BPL FOUND	
0360:A9 00	79	LDA #BLACK	
0362:20 64 FB	80	JSR SETCOL	
0365:AD 05 03	81	LDA SAVEA	
0368:20 00 FB	82	JSR PLOT	
036B:AD 57 C0	83	LDA HIRES	
036E:C8	84	INY	
036F:CC 06 03	85	CPY HEND	
0372:90 D4	86	BCC AGAIN	
0374:F0 D2	87	BEQ AGAIN	
0376:AC 08 03	88	LDY HSTART	
0379:EE 05 03	89	INC SAVEA	
037C:AD 05 03	90	LDA SAVEA	
037F:CD 07 03	91	CMP VEND	
0382:90 C4	92	BCC AGAIN	
0384:F0 C2	93	BEQ AGAIN	
0386:4C 9A 03	94	JMP DONE	
0389:A9 00	95	FOUND LDA #BLACK	
038B:20 64 FB	96	JSR SETCOL	
038E:AD 05 03	97	LDA SAVEA	
0391:BD 03 03	98	STA VERT	
0394:8C 04 03	99	STY HORIZ	
0397:20 00 FB	100	JSR PLOT	
039A:AD 57 C0	101	DONE LDA HIRES	
039D:60	102	RTS	

*** SUCCESSFUL ASSEMBLY: NO ERRORS

程式列表 2 控制光筆繪圖之 BASIC 主程式

ALIST

100 REM *****	
110 REM	; 用以控制光筆繪圖之 BASIC 主程
120 REM PROGRAM TO DRAW	式
140 REM	
150 REM *****	
160 REM	
170 PRINT CHR\$(4); "LOAD PENSUBROUTINE.BR	; 載入光筆搜尋常式
JO"	
180 NMAX = 50	; 最多可在螢幕上設定 50 個繪圖點
190 DIM PX(NMAX),PY(NMAX)	; 繪圖點之儲存陣列宣告
200 GOSUB 390	; 至程式 390 去執行初值設定常式
210 GOSUB 610	; 至程式行 610 去呼叫光筆搜尋常式且算出光筆在高解像畫面之對應座標點
220 KEY = PEEK (KB)	; 讀取按鍵輸入
230 IF KEY < 128 GOTO 210	
240 HOME	
250 T = PEEK (FS)	; 清除按鍵徵登
260 IF KEY = SPACE THEN GOSUB 710: GOTO 210	; 若按下空白鍵 (SPACE BAR) 則設定光筆所在位置為繪圖點
270 KEY = KEY - 192	
280 IF KEY < 1 OR KEY > 17 THEN GOSUB 2060	; 若輸入命令有誤則跳至程式行 2060 去執行錯誤處理
: GOTO 210	
290 ON KEY GOSUB 2060,2060,1400,860,2060,2060,2060,1840,2060,2060,1240,2060,1600,2060,1090,1690	; 根據輸入命令跳至對應之副常式去執行

```

300 T = PEEK (B)
310 GOTO 210
320 REM
330 REM *****
340 REM
350 REM INITIALIZE
360 REM
370 REM *****
380 REM
390 KB = 49152:KS = 49168
400 PEN = 768
410 HZ = PEN + 4:VERT = PEN + 3
420 CL = 3
430 SPACE = 160:G = 49232
440 N = 0
450 POKE HZ,20:POKE VERT,20
460 TEXT : HOME : NORMAL : SPEED= 255
470 VTAB 12:PRINT TAB(7):"*** WELCOME TO
    LINES ***"
480 PRINT :PRINT TAB(10):"DO YOU WANT A
    MENU ?";
490 GET Y$
500 IF Y$ = "Y" THEN GOSUB 1840
510 T = PEEK (KS)
520 HGR : HCOLOR= CL
530 RETURN
540 REM
550 REM *****
560 REM
570 REM FIND PEN. SET X,Y
580 REM
590 REM *****
600 REM
610 CALL PEN
620 X = 7 * PEEK (HZ):Y = 4 * PEEK (VERT)
630 RETURN
640 REM
650 REM *****
660 REM
670 REM "SPACE"---MARK POINT
680 REM
690 REM *****
700 REM
710 IF N < NMAX GOTO 750
720 GOSUB 2260
730 PRINT TAB(3):"TOO MANY POINTS."
740 RETURN
750 N = N + 1
760 PX(N) = X:PY(N) = Y
770 GOSUB 2160
780 RETURN
790 REM
800 REM *****
810 REM
820 REM "D"---DELETE POINT
830 REM
840 REM *****
850 REM
860 I = 1
870 IF PX(I) = X AND PY(I) = Y GOTO 930
880 I = I + 1
890 IF I <= N GOTO 870
900 GOSUB 2260
910 PRINT TAB(3):"NO POINT TO DELETE"
920 RETURN
930 IF I = N GOTO 970
940 FOR Z = I TO N - 1
950 PX(Z) = PX(Z + 1):PY(Z) = PY(Z + 1)
960 NEXT Z
970 N = N - 1
980 HCOLOR= 0

```

;清除按鍵激發

;初始化常式

; { KB =按鍵輸入
KS =清除按鍵激發
PEN =光筆搜尋常式之起始位置
HZ =光筆之水平座標儲存位置
VERT =光筆之垂直座標儲存位置
CL = 3 設定顏色為白色

;決定是否要顯示螢幕輔助菜單？

;若要則跳至程式行1840顯示螢幕輔助菜單

;呼叫光筆搜尋常式且求出光筆位置
在高解線畫面之對應座標

;按下空白鍵以設定繪圖點

;取消現在光筆所在位置之繪圖點

```

990 GOSUB 2160
1000 HCOLOR= CL
1010 RETURN
1020 REM
1030 REM *****
1040 REM
1050 REM "P"---MAKE A POLYGON ; 將所設定之各繪圖點以線連接起來
1060 REM
1070 REM *****
1080 REM
1090 IF N >= 3 GOTO 1130
1100 GOSUB 2260
1110 PRINT TAB(3); "NEED AT LEAST 3 POINTS
"
1120 RETURN
1130 HPLOT PX(N),PY(N) TO PX(1),PY(1)
1140 GOSUB 1240
1150 N = 0
1160 RETURN
1170 REM
1180 REM *****
1190 REM
1200 REM "L"---DRAW LINES ; 將第一個繪圖點和最後一個繪圖點
1210 REM ; 以線連接起來
1220 REM *****
1230 REM
1240 IF N >= 2 GOTO 1280
1250 GOSUB 2260
1260 PRINT TAB(3); "NEED AT LEAST 2 POINTS
"
1270 RETURN
1280 FOR I = 1 TO N - 1
1290 HPLOT PX(I),PY(I) TO PX(I + 1),PY(I +
1)
1300 NEXT I
1310 N = 0
1320 RETURN
1330 REM
1340 REM *****
1350 REM
1360 REM "C"---CHANGE COLOR ; 設定光筆繪圖之顏色
1370 REM
1380 REM *****
1390 REM
1400 TEXT : HOME
1410 VTAB (3): PRINT "THE POSSIBLE COLORS A
RE:"
1420 PRINT : PRINT " 0=BLACK"
1430 PRINT : PRINT " 1=GREEN"
1440 PRINT : PRINT " 2=BLUE"
1450 PRINT : PRINT " 3=WHITE"
1460 PRINT : PRINT " 4=BLACK"
1470 PRINT : PRINT " 5=YELLOW"
1480 PRINT : PRINT " 6=BLUE"
1490 PRINT : PRINT : INPUT "WHAT COLOR (0-6
) ?";CL
1500 IF CL < 0 OR CL > 6 GOTO 1490
1510 HCOLOR= CL
1520 RETURN
1530 REM
1540 REM *****
1550 REM
1560 REM "N"---NEW CLEAR SCRIN ; 清除螢幕顯示且將繪圖點清除為 0
1570 REM
1580 REM *****
1590 REM
1600 N = 0: HGR
1610 RETURN
1620 REM
1630 REM *****
1640 REM

```

```

1650 REM "Q"--QUIT
1660 REM
1670 REM *****
1680 REM
1690 VTAB (23): PRINT "DO YOU WANT TO SAVE
THIS? "
1700 GET Y$
1710 IF Y$ < > "Y" GOTO 1740
1720 INPUT "NAME OF FILE? ";Y$
1730 PRINT CHR$ (4);"BSAVE ";Y$;"_PIC,AB19
2,LB192"
1740 TEXT : HOME : NORMAL
1750 POP
1760 END
1770 REM
1780 REM *****
1790 REM
1800 REM "H"--HELP (MENU)
1810 REM
1820 REM *****
1830 REM
1840 TEXT : HOME : NORMAL
1850 PRINT : PRINT "HERE'S WHAT YOU CAN DO-
-"
1860 PRINT : PRINT : PRINT
1870 PRINT : PRINT TAB( 2);"SPACE--MARK CU
RRENT POINT"
1880 PRINT : PRINT TAB( 6);"D--DELETE CURR
ENT POINT"
1890 PRINT : PRINT TAB( 6);"P--CONNECT AS
A POLYGON"
1900 PRINT : PRINT TAB( 6);"L--CONNECT FIR
ST TO LAST POINTS"
1910 PRINT : PRINT TAB( 6);"C--CHANGE COLO
R"
1920 PRINT : PRINT TAB( 6);"N--NEW: CLEAR
SCREEN"
1930 PRINT : PRINT TAB( 6);"Q--QUIT"
1940 PRINT : PRINT TAB( 6);"H--HELP!!!"
1950 PRINT : PRINT TAB( 6);"PRESS
ANY KEY TO CONTINUE..."
1960 GET Y$
1970 T = PEEK (KS)
1980 RETURN
1990 REM
2000 REM *****
2010 REM
2020 REM COMMAND ERROR
2030 REM
2040 REM *****
2050 REM
2060 GOSUB 2260
2070 PRINT TAB( 3);"UNRECOGNIZED COMMAND"
2080 RETURN
2090 REM
2100 REM *****
2110 REM
2120 REM PLOT CROSS
2130 REM
2140 REM *****
2150 REM
2160 HPLLOT X - 1,Y - 1 TO X + 1,Y + 1
2170 HPLLOT X - 1,Y + 1 TO X + 1,Y - 1
2180 RETURN
2190 REM
2200 REM *****
2210 REM
2220 REM ERROR MESSAGE
2230 REM
2240 REM *****
2250 REM
2260 VTAB (24): FLASH : PRINT "ERROR!!!": NORMAL
: RETURN

```

； 終止程式之執行且決定是否將螢
幕顯示之高解像圖形儲存至磁片中

； 螢幕輔助菜單—光筆操作說明

； 按下空白鍵，設定現在光筆所在位
置為繪圖點

； 取消現在光筆所在之點

； 將所設定之各點繪圖點用線連接起
來

； 將頭一點和最後一點連接起來

； 改變顏色

； 清除螢幕

； 停止程式之執行

； 顯示光筆操作說明

； 按下任一鍵以繼續程式之執行

； 等待一個按鍵輸入

； 消除按鍵激發

； 命令錯誤處理常式

； 在設定之繪圖點上畫一個交叉 (x
) 符號

； 顯示錯誤信息

利用——

APPLE II

控制幻燈機電路設計

控制幻燈機電路設計

控制幻燈機電路設計

控制幻燈機電路設計

前言

利用微電腦做許多控制電路工作，例如控制步進馬達，紅綠燈等工作，在以前已有許多篇文章討論到，相信大家一定十分熟悉。現在我們介紹一種低成本介面控制電路（材料費低於100元），來作APPLE-II與幻燈機介面，使APPLE-II控制幻燈機前進一張幻燈片，後退一張幻燈片，以及能依需要選擇任一張編號之幻燈片等功能，使您的APPLE-II電腦功能得以做進一步發揮。

各項變因的考慮

在設計這項電路時，我們必須先考慮各項輸入，輸出條件，方能有效配合設計介面電路。

輸出條件選擇

因為是由APPLE-II控制幻燈機，所

以針對APPLE-II輸出信號考慮發現由APPLE-II game paddle 中第11～14脚，可以提供我們所需信號其條件如下：

(A) 輸出電壓是0 V與5 V

(B) 輸出電流是5mA～20mA

(C) 可以由BASIC 程式控制其輸出0 V或5 V，例如程式中POKE-16289，0 (PEEK (-16289))，會使APPLE-II game paddle 12脚輸出+5 V電壓，而POKE-16290，0 (PEEK (-16290))會使12脚輸出腳電壓降至0 V，同理POKE-16291,0與POKE-16292,0也會使APPLE-II game paddle 13脚產生5 V，0 V。

由以上條件，所以我們必須找尋一種開關能配合上述條件，且不影響幻燈機控制電源，經過比較後採用Reed Relay作為我們輸出信號開關，其特點如下：

(A) 可以用 5 V , 0 V 推動其工作，不工作。

(B) 電源輸入，輸出部份分開不互相干擾。

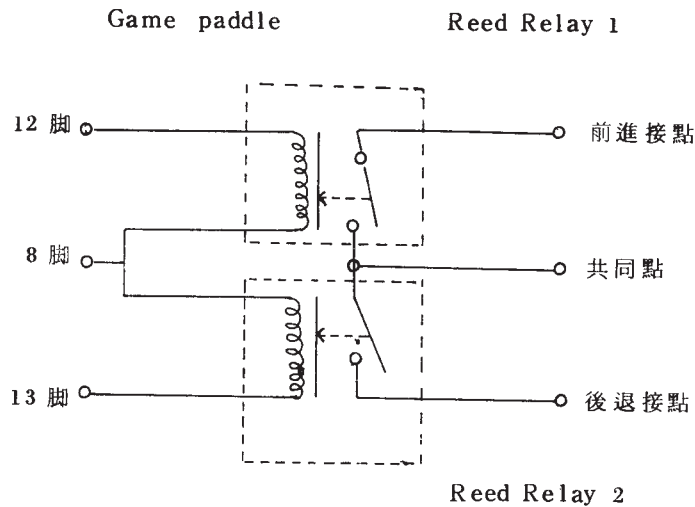
輸入信號條件分析

輸入為柯達幻燈機上，遙控開關之輸入點，其輸入信號是 5 V 之交流電，由共

同點接至前進點或接至後退點來決定其前進、後退，所以我們把 Reed Relay 輸出（被控制）部份，接至柯達幻燈機遙控開關上，以達控制目的。

硬體電路

經過剛才輸出入信號考慮後，線路設計如下：

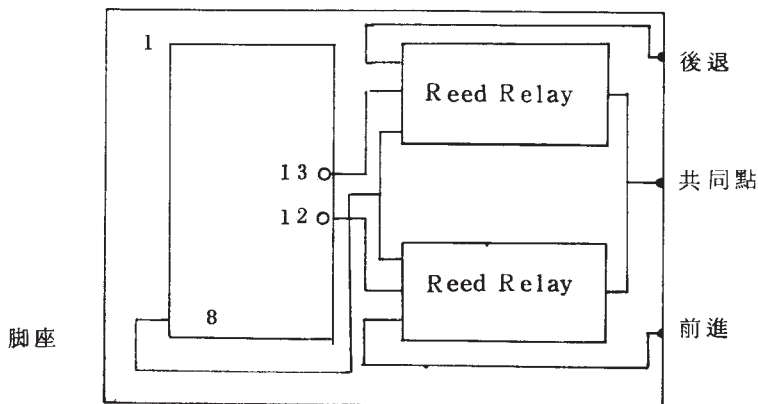


圖一

材料：2 個 Reed Relay 開關

1 個 IC 板、連線、腳座

線路接腳圖



圖二

軟體程式及工作原理

控制幻燈機前進一張，後退一張幻燈片及
依需要找尋一張編號之幻燈片等功能。

現在我們用下列三個副程式說明如何

1. 前進一張幻燈片之副程式

```
1000 G = PEEK ( - 16289): FOR I = 送出正 5 V 信號至 12 脚  
1 TO 500: NEXT I  
1010 G = PEEK ( - 16290): FOR I = 關閉 12 脚正 5 V 信號使其為 0  
1 TO 3000: NEXT I  
1020 RETURN
```

原理說明：

當行號 1000 被執行時 Game paddle 12 脚送出一個 + 5 V 電壓，經線路至 Reed Relay 1 時，使 Reed Relay 1 工作，這使得前進接點與共同點相通，幻燈片前進一張。經一小段時間後行號 1010

執行時，使 Game paddle 12 脚送出 0 V，使 Reed Relay 1 不工作，使前進點與共同點中斷恢復原始狀態，但已達成前進一張目的。

2. 退後一張幻燈片副程式

```
2000 G = PEEK ( - 16291): FOR I = ( 送出一個 + 5 V 信號至 13 脚 )  
1 TO 500: NEXT I  
2010 G = PEEK ( - 16292): FOR I = ( 關閉 13 脚信號使為 0 V )  
1 TO 3000: NEXT I  
2020 RETURN
```

原理說明：

後退一張的原理同於前進一張，是由 POKE-16291,0 提 + 5 V 至 13 脚推動 Reed Relay 使後退一張工作，POKE-16292,0 關閉之。

3. 對於如何依需要選擇任何一張幻燈

片其作法如下：

(A) 先用位址 850 (十進制) 作為幻燈片編號記錄位址。

(B) 再利用所需編號之值與記錄位址中，編號之值相減，所得之值為該前進幾張或後退幾張之值。

```
3000 QQ = PEEK (850)  
3010 LL = PP - QQ  
3020 IF LL < 0 THEN 3100
```

找出目前幻燈片編號
求由輸入值 (PP) 與目前幻燈
編號之差
若負值表需後退一張

3030 IF LL > 0 THEN 3050	若正值表需前進一張
3040 IF LL = 0 THEN 3200	若為 0 表示已達我們所要之張數
3050 G = PEEK (- 16289): FOR I =	
1 TO 500: NEXT I	前進副程式
3060 G = PEEK (- 16290): FOR I =	
1 TO 3000: NEXT I	前進，編號加一張
3070 POKE 850,00 + 1	
3080 GOTO 3000	
3100 G = PEEK (- 16291): FOR I =	
1 TO 500: NEXT I	後退副程式
3110 G = PEEK (- 16292): FOR I =	
1 TO 3000: NEXT I	後退編號減一張
3120 POKE 850,00 - 1	
3130 GOTO 3000	
3200 G = PEEK (- 16292):G = PEEK (消除 Relay 1,2 之電壓)	
(- 16290)	
3210 RETURN	

結論

本介面電路經過多次實驗後證明其功能十分實用，雖然是一個簡單，省錢電路

，但也提供 APPLE-II 與幻燈機間之控制，相信聰明讀者能由此簡單例子想出更複雜、更精密之 I/O 控制電路。

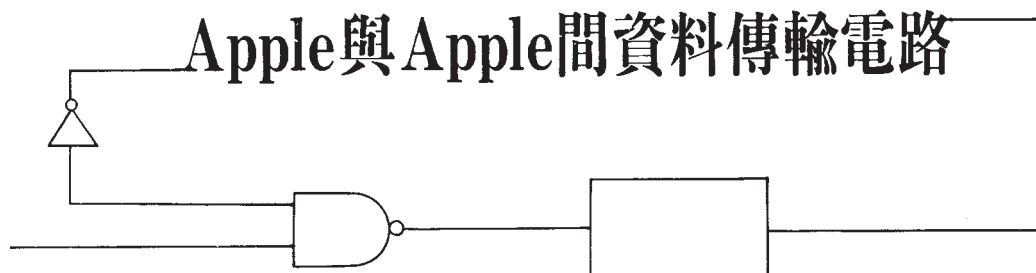
(上接 58 頁)

95C0- 08 AA BD 80 C0 A6 08 60	95E8- 1C 1C 1C 00 00 00 00 00
95C8- A2 13 CA D0 FD 38 E9 01	95F0- 00 00 00 00 00 00 00 00
95D0- D0 F6 60 01 30 28 24 20	95F8- 00 00 00 00 00 00 00 00
95D8- 1E 1D 1C 1C 1C 1C 1C 70	9600- D0
95E0- 2C 26 22 1F 1E 1D 1C 1C	

請建議您的親友：如果祇是爲了學習而想買電腦，請儘量到本刊「廉讓專欄」裡來選購，等到學會了，仍然可以到「廉

讓欄」再折價讓給他人學習，如此當可皆大歡喜！

利用game I/O埠界面設計一個——



Apple與Apple間資料傳輸電路

前言

Apple-II微電腦之所以廣受大家歡迎的原因，「電腦（動）遊戲」（game），也佔了不少原因，您應該想過，也早就知道，當在玩電腦（動）遊戲時，你搖桿按左移，你的太空船也左移，右移時，太空船也右移，你按發射鍵（Paddle上）時，你的太空船也配合發射雷射彈，這一些工作是Apple-II微電腦內部Paddle界面電路在配合你的工作。這一些工作是把外界線性輸入，按鍵（機械動作）轉成數位信號輸入，以便處理這些機械性（左移，右移）動作，使你在玩電動遊戲時，更加刺激有趣。

而Apple-II game I/O插座是否僅限於控制遊戲game呢？就我們所知道，在外面也有不少書籍及廠商，也利用這搖桿（game）I/O界面電路，做了不少工

業方面的控制程式、電路，如控制馬達、順序控制等，這在以前各期中也討論了許多，而今天我們是拿搖桿（game I/O）界面電路中輸入、出的功能，討論如何設計Apple-II微電腦與另一台Apple-II間資料傳輸電路。

設計要求

在討論這電路以前，我們先了解，所謂資料傳輸就是希望把一台電腦資料，透過外界電路，送到另一台電腦裏面。今天我們是討論簡單資料傳輸，目的是達成一個實驗性、學習性研究，故我們所採取電路都是簡單、易懂材料元件。

現在我們就討論Apple-II搖桿game I/O傳輸界面電路設計，基於剛才所說的「把某一台電腦（Apple-II）資料透過外部電路傳送到另一台電腦記憶中」，我們可以勾出電路設計要求如下圖：

甲 Apple-II 微電腦

乙 Apple-II 微電腦

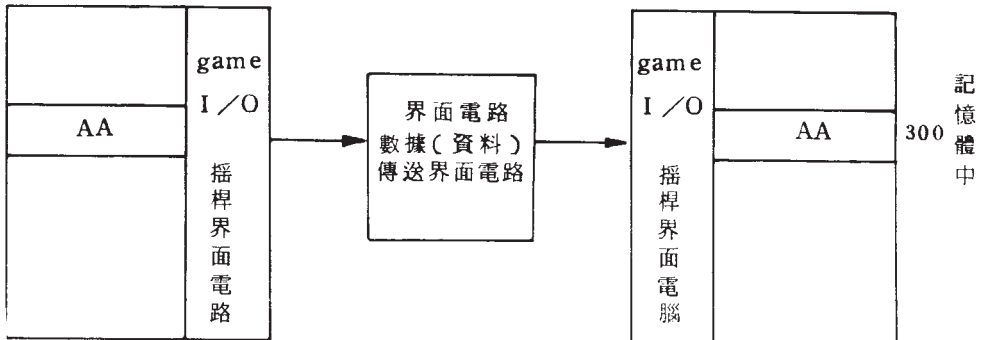


圖 1

我們再把圖 1 做更詳細說明如下：

這圖 1 說明把 Apple-II 微電腦 (甲) 其記憶地址 300 內容 AA (\$300-AA)，透過本身 game I/O 界面電路，傳送到資料界面電路，再送至另一 Apple-II 微電腦 (乙) game I/O 界面電路再傳至 Apple-II 微電腦 (乙) 記憶體位址 300 內，使乙電腦 Apple-II 記憶位址 300 內容變成 AA，這就達成我們的設計要求。

軟體界面電路設計

基於上述設計要求，我們討論

(1) 如何把 300 位址中內容 AA，傳送到達 game I/O 搖桿輸出部分。

首先考慮到 Apple-II game I/O 輸

出部分，是電鈴式輸出提供高低電壓 (H-5V L-0V)，這做法如下，使 game I/O 埠第 15 腳 (AN0) 輸出電鈴輸出 H, L，其做法只要存放位址到 \$C058 (POKE 49240, 0)，就可使 game I/O 埠第 15 腳輸出低電壓 (0V)，同理只要存放位址到 \$C059 (POKE 49241, 0) 就可使 game I/O 埠第 15 腳輸出高電壓 (5V)。我們如果把 300 位址中內容 (AA) 做左旋轉，把內容如 ASL 命令送出內含 (把第 8 Bit 內容送至 C 暫存器)，再判斷 C 暫存器 0 或 1，以此來提供觸動 \$C058 (POKE 49240, 0) 或觸動 \$C059 (POKE 49241, 0) 位址以達成輸出信號目的。

輸出設計草圖

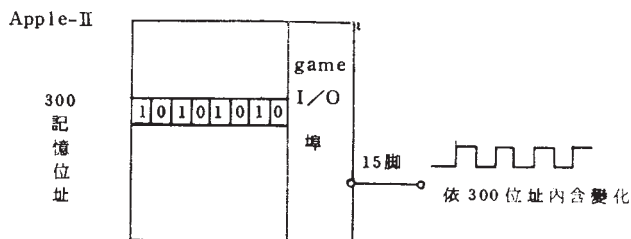
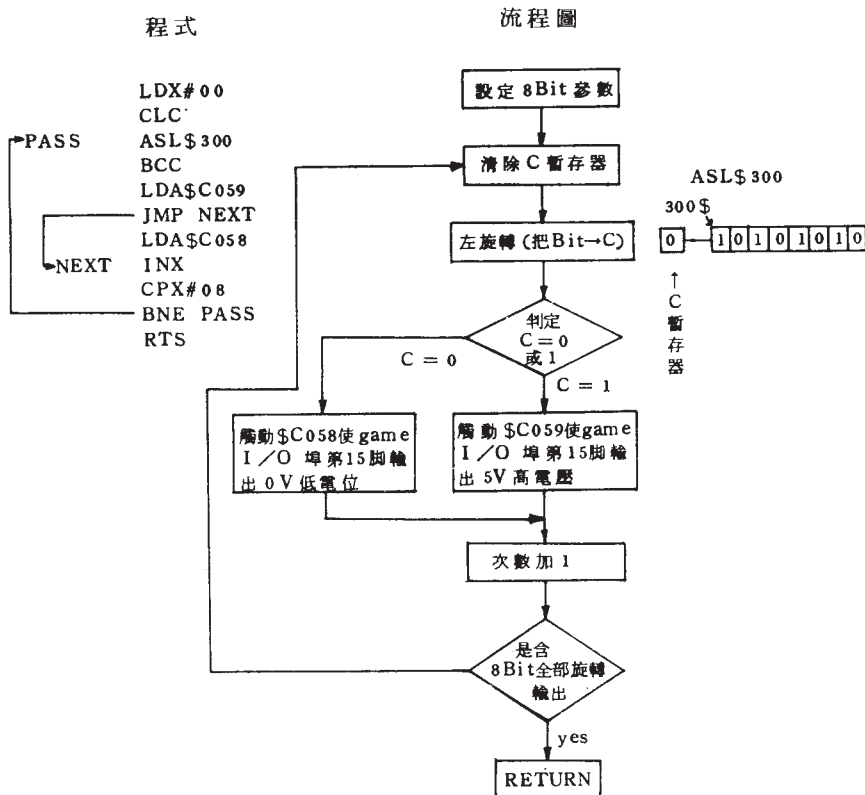


圖 2

其程式及流程圖如下：



(2)再下來我們討論分析如何設計一軟體程式 I/O 埠如何接受外界信號後，把信號轉化成地址 \$300 內容。

在 Apple-II game I/O埠有下列一個特性，可以接受外來開關轉換成信號，其電路如下：

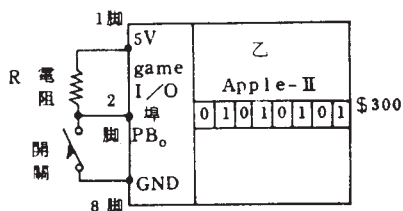


圖 3

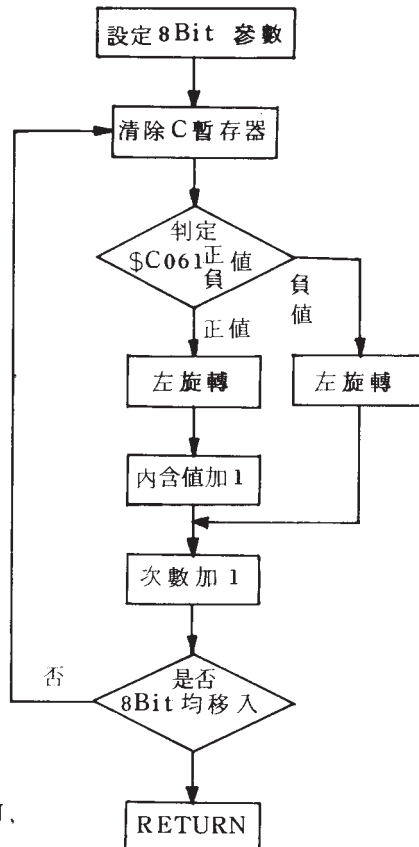
這一個電路十分簡單，也就是說我們可以以開關的 ON (開)，OFF (關)，影響 PB₀ 的電位 (0V 或 5V)，而使得 Apple-II 內含值 \$C061 位址上 Bit 7 發生 0, 1 值變化，換一句話說開關關上時 PB₀ 電位為 0V，迫使 \$C061 之內含值變成負值，當開關打開時 PB₀ 電位為 5V，也使得 \$C061 之內含值變成正值，如果我們在設計上使得輸出 (剛才) 信號 0V, 5V 影響 PB₀ 使得 PB₀ 發生變化，再用如下程式加以轉換記錄信號，轉成乙電腦 \$300 內含

程 式

```

LDX#00
CLC
LDA$C061
BMI
ASL$300
ADC#01
JMP
ASL$300
INX
CPX#08
BNE
    
```

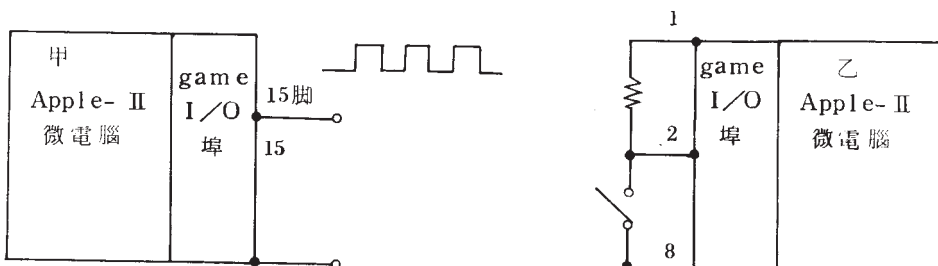
流程圖



如此就可以達到輸入目的（由開關 ON、OFF 輸入）。

談到這裏我們把上述輸出、輸入的草圖列出如下：

硬體界面電路設計



所以我們只要考慮由甲 Apple-II game I/O 埠微電腦 15 腳電壓影響，乙 Apple-II 微電腦 game I/O 埠 2 腳開關就可達成我們所要求資料傳送目的。如果乙

Apple-II 微電腦的開關可以用電晶體代替 1，再用甲電腦 15 腳正負電壓觸動就可達成我們設計其電路圖如下：

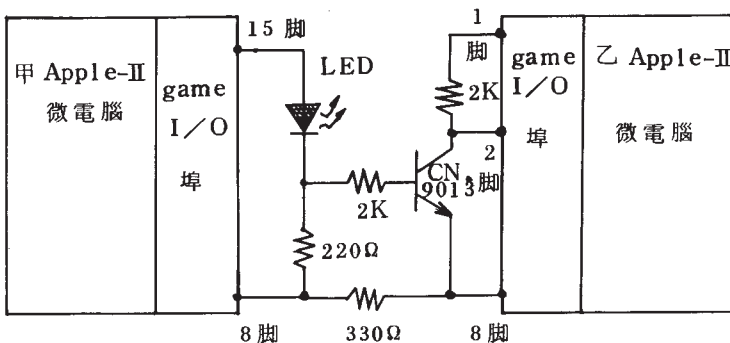


圖 4

這圖的工作原理如下，如剛才所說的由甲 Apple-II 微電腦 game I/O 埠已由內部把位址 \$300 內容轉成電壓 (0V, 5V) 送到 game I/O 埠 15 腳去影響電晶體 9013 使其開或關使得乙 game I/O 埠依甲 Apple-II 地址 \$300 內容變化，然後再存入乙 Apple-II 位址 \$300 內，如此就達到資料傳送的目的。

傳輸工作原理及動作介紹

現在談到了這裏，我想大家都很清楚了解傳送過程了，接下來我們談的問題是如何確保資料傳送完整，也就是說乙微電腦如何知道何時開始接受資料，甲微電腦如何了解乙微電腦已準備好了，可以發射

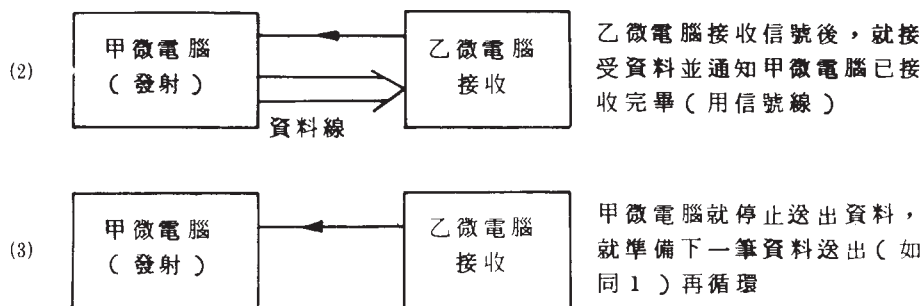
資料了。這我們先談一談所謂“握手式”輸入輸出來解決上述問題。

(1) 握手式傳輸

握手式傳輸是無時序 (unclocked) 傳輸方式，其工作方式是先由甲微電腦 (發射、輸出部份) 先把資料準備好了，甲微電腦在用通知信號告訴乙微電腦 (接收、輸入部份)，乙微電腦也把要接收部份準備好，再用信號線通知甲微電腦，此時甲微電腦就送資料，乙微電腦收資料，乙微電腦接收完畢後通知甲微電腦，甲微電腦也通知乙微電腦發射完畢，完成傳輸工作，由圖我們可以更清楚了解工作情況。



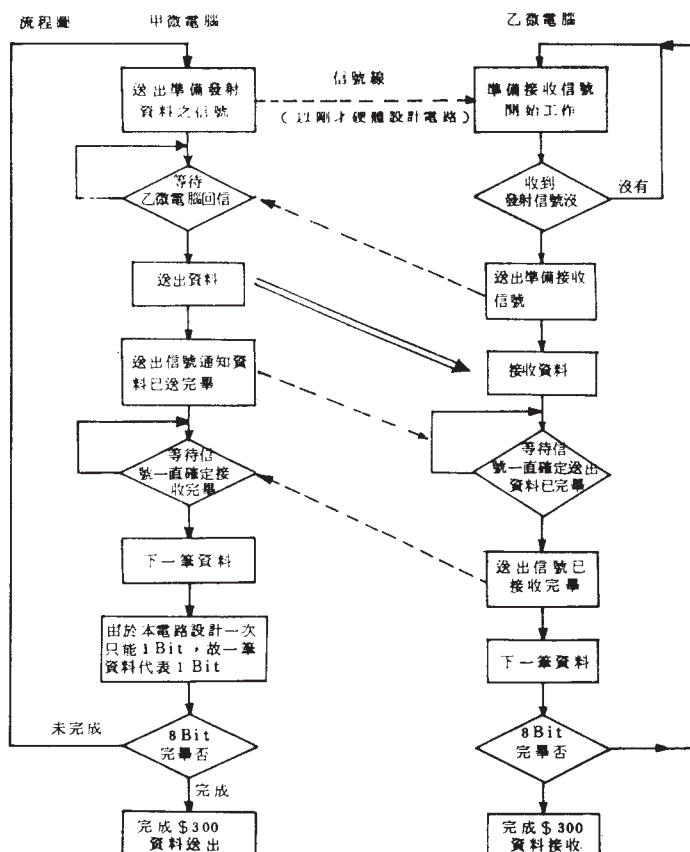
甲微電腦 (發射) 送出資料
同時發射信號線通知乙微電腦 (接收)



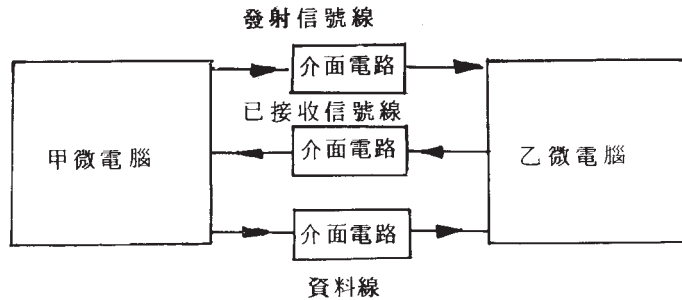
(2)原理

我們就設計我們的軟體程式，達成我

們先前要求 (把甲微電腦位址 \$ 300 內容 (AA) 傳到乙微電腦位址 \$ 300 內)，依剛才所談原理其流程圖如下。

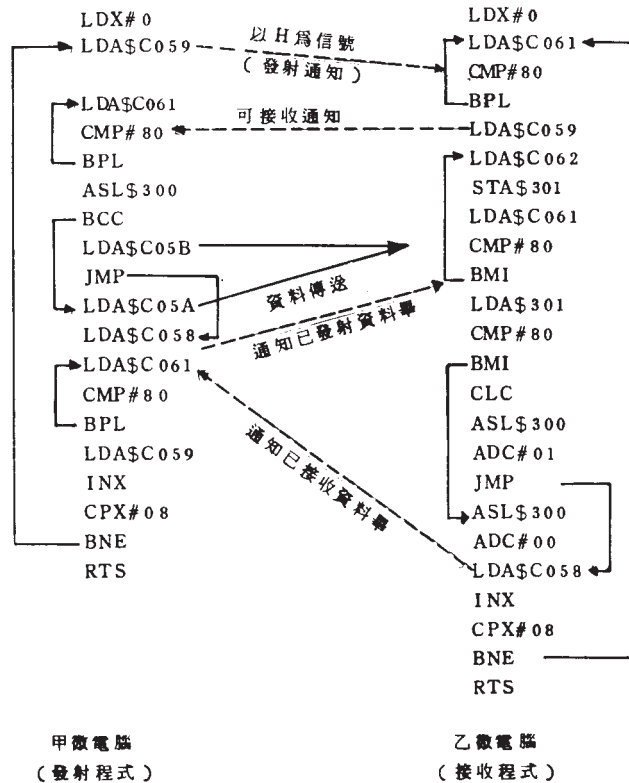


(3)硬體方塊



介面電路為剛才所探討軟體介面電路
 一共要三組以上（三組如圖作單向傳送資料，四組就可双向傳送）。

(4)程式設計



檢討與建議

經過剛才討論過後，我想大家很清楚了解到資料如何經過 game I/O 介面傳送，下面針對這研究後，做下列建議：

1. 繼續擴大程式，把本次研究的副程式做成連續一段落或一檔案轉移，這可以提高研究功能。

2. 硬體電路所用 9013 電晶體更改更高頻性電晶體更佳，但是本次研究也有檢討地方：

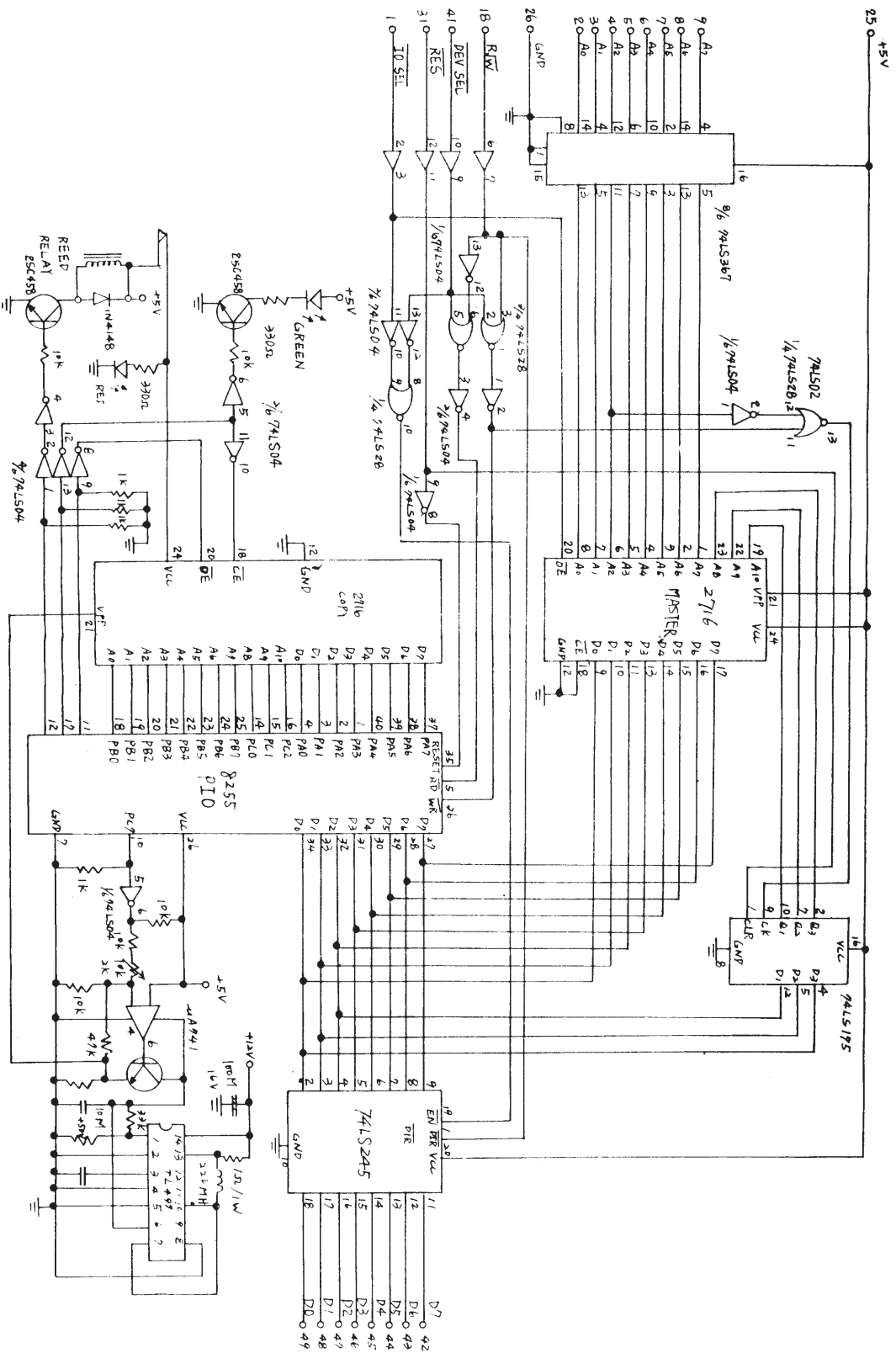
(1) 佔用 game I/O 介面，使在傳送時 game I/O 介面不能使用，減少微電

腦功能。

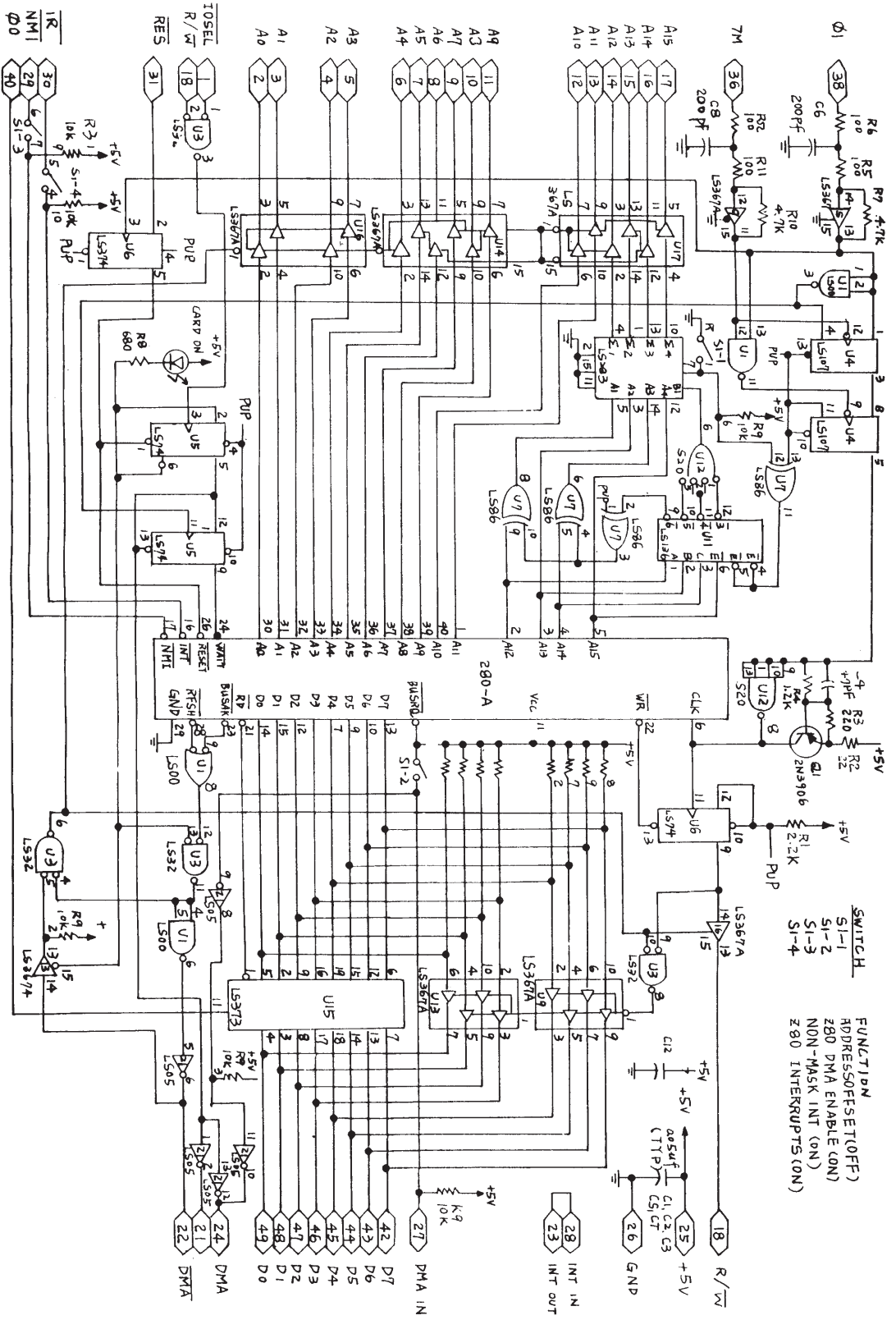
(2) 傳送速度太慢，經研究後，本研究傳送大約每秒 10 個～30 字元之間，不實用。

(3) 使用電晶體電路，如果速度提高時或距離加長時，傳送時會有錯誤，故對雜訊處理力弱。

總之，不管檢討如何，在開始我們就已界定本研究是提供一個簡單，易了解電路，程式來完成資料傳送功能，各個研究前輩如果有意見或有問題請郵寄嘉義師專合作社洪燕竹收，來給我一個指正機會或提供給我更好意見，謝謝！ ■



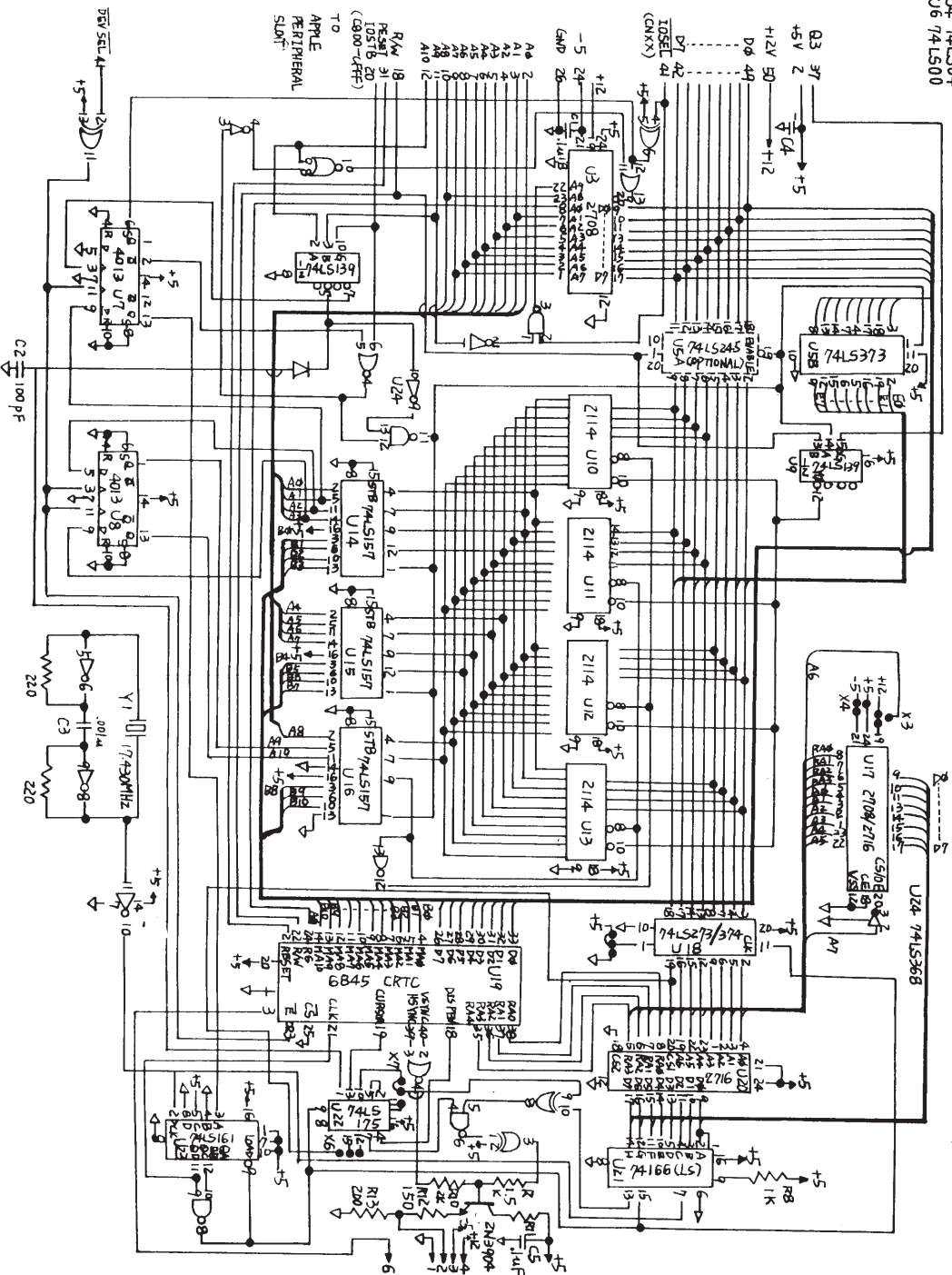
⑨ 2716 EPROM WRITER CARD



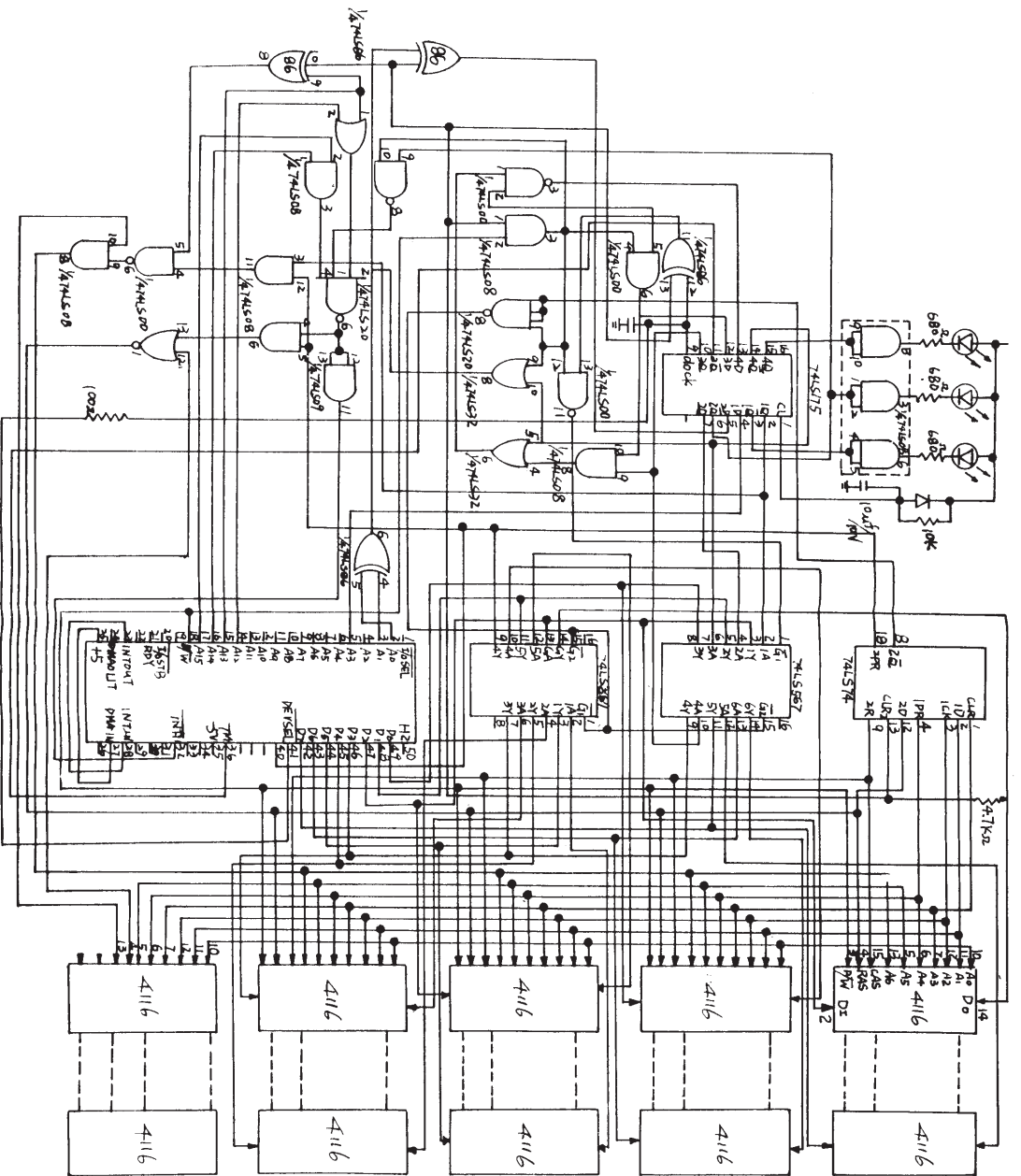
FUNCTION
 ADDRESSOFTWARE(0FF)
 Z80 DMA ENABLE (ON)
 NON-MASK INT (ON)
 Z80 INTERRUPTS (ON)

① Z80 SOFT CARD

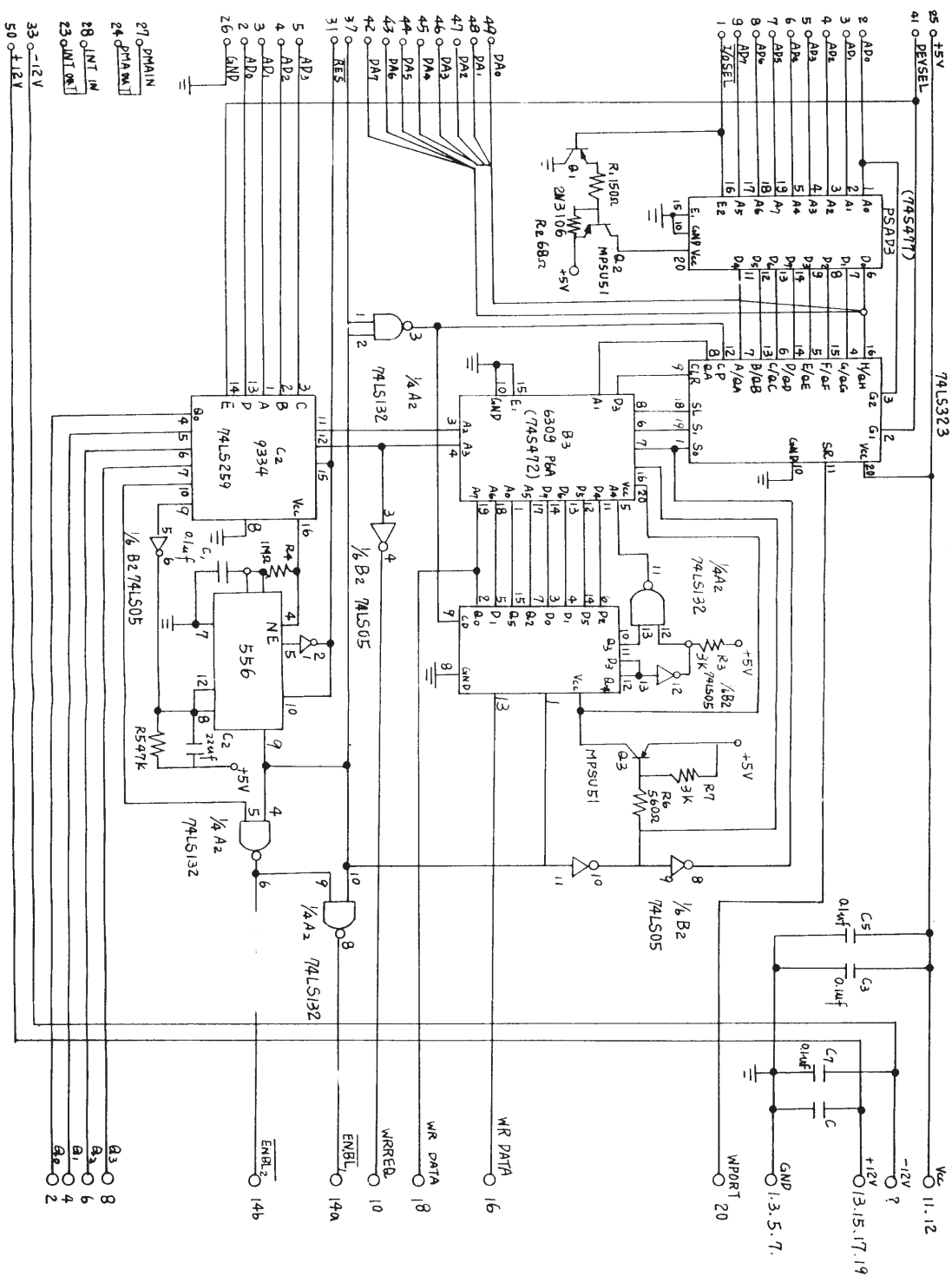
U1 74LS86
U2 74LS02
U4 74LS04
U6 74LS00



② 80 - COLUMN CARD

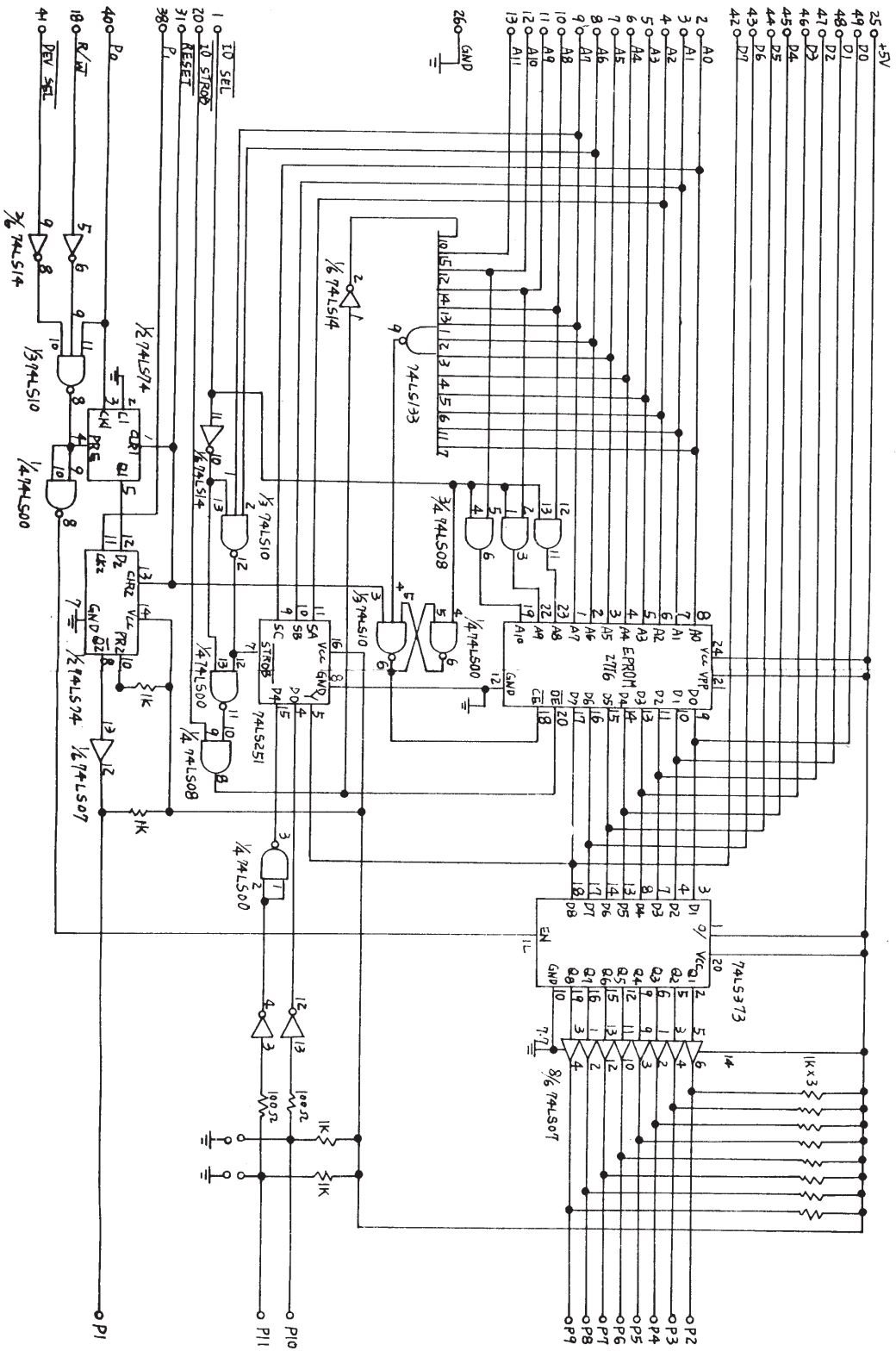


③ 16K RAM CARD

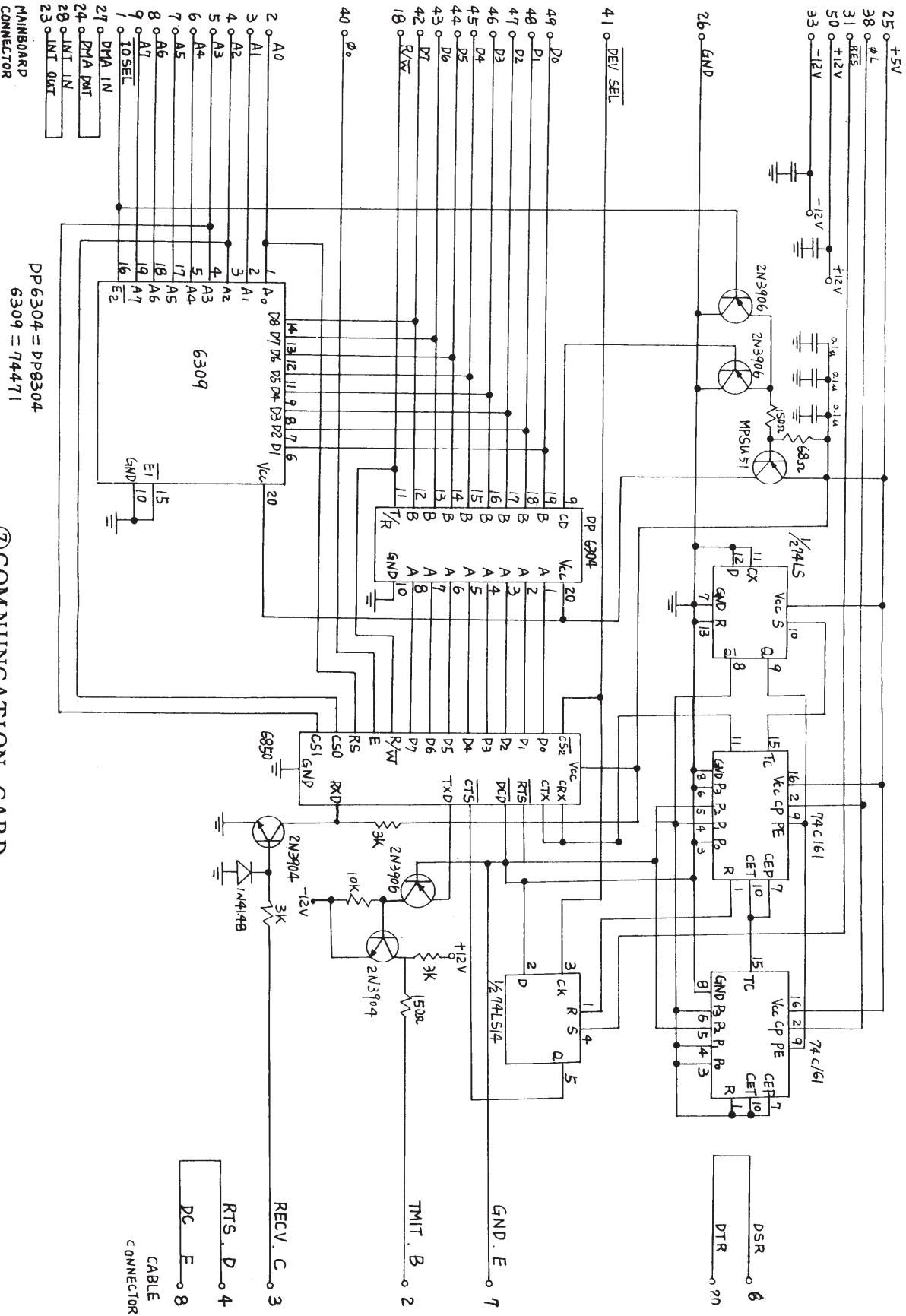




RES



⑧ PERIPHERALS CARD



⑦ COMMUNICATION CARD