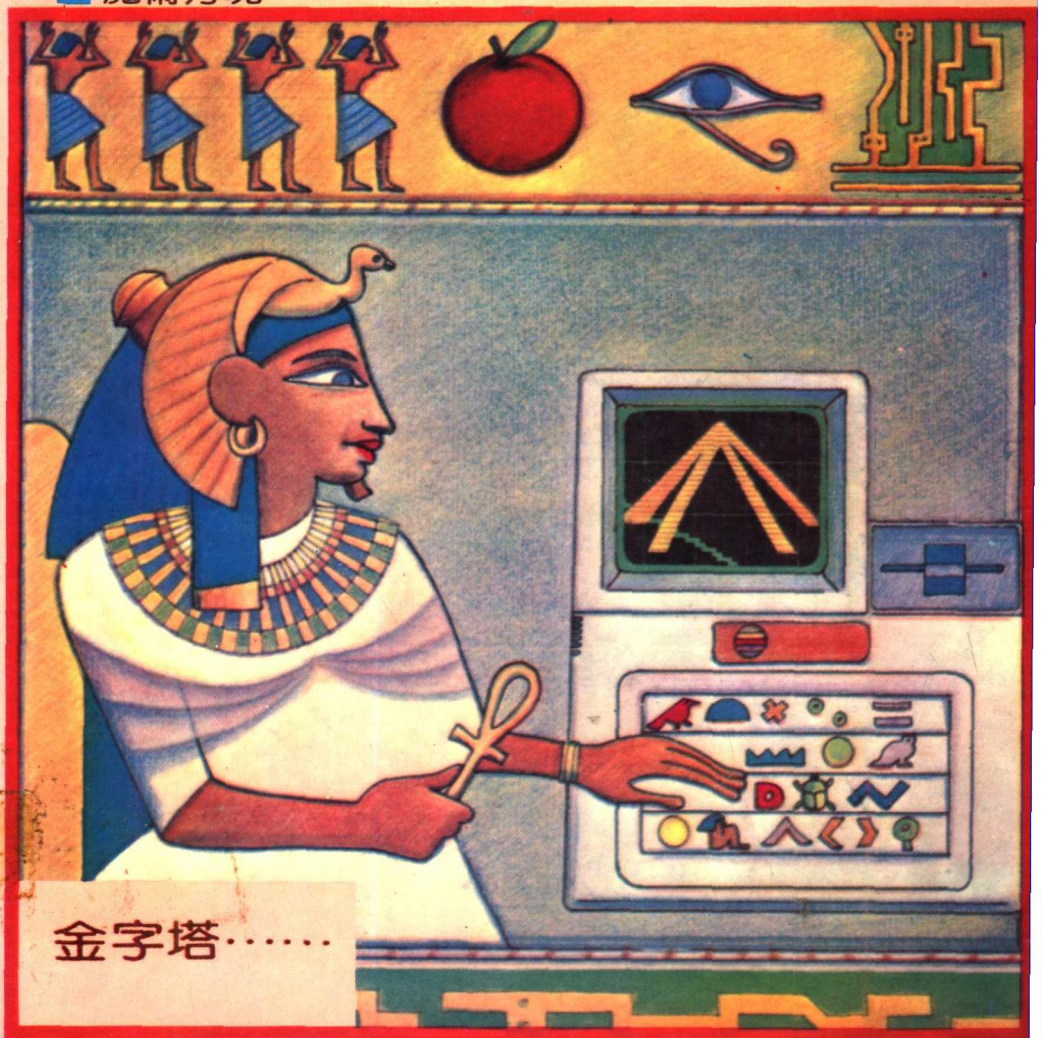
APPLE II 繪圖遊戲

宗世麟 著

從APPLE II 到 圖形遊戲製作

GRAPHIC GAME

- 海空大戰
- 太空迷陣
- 打 磚 塊
- 繪 圖 板
- 魔術方塊



商 業 出 版 社

編著者：宗世麟
出版社：商業出版社
營地街一〇八號二樓
印刷廠：聯合印刷廠
賣青草地街九十六號
定價：港幣三十五元正

PRINTED IN MACAU

序　言

目前家用電腦風行台灣，許多家庭均擁有一部微電腦，其功能之繁多及精巧，更激起了許多人對微電腦的學習興趣。本書即根據這種需要，並以目前使用最廣的APPLE Ⅱ爲主，希望能藉著大家所熟悉的娛樂遊戲程式，逐步探討，使讀者能了解APPLE Ⅱ微電腦的結構、遊戲程式的製作原理與程式設計的觀念及技巧。

由於本書的目的並不是一本遊戲程式集，故範圍從軟體的設計到硬體的結構均包含在內，並爲了使讀者能徹底了解遊戲程式的製作方法吸收前人的長處及技巧，書中對每一部份均作了十分詳盡的說明。

全書主要分爲三部份，第一部份對所有必具的知識均加以討論，甚至對於APPLESOFT亦以系統的眼光來說明，使讀者不僅會使用BASIC語言，更能了解 BASIC 語言的內部結構。第二部份則討論一些在製作遊戲程式時，常用的一些工具（即程式），如音樂的產生、及高、低解像圖形的製作程式。以幫助讀者，使能輕易的製作出屬於自己的遊戲程式。第三部份，搜集了10個遊戲程式，涵蓋了大部份 BASIC 遊戲程式的種類，許多都是市面上包含在錄音帶或磁碟片上的遊戲程式，書中排列方按難易程度，由淺入深排列。

因爲本書所列出的各類遊戲程式並非很容易，因此作者對各

程式均說明的十分詳細，使只要略通 BASIC 程式設計的人均能輕易的看懂這些程式。

最後，作者要感謝鍾健堯同學提供資料及鼎力相助使本書能如期出版。

宗世麟

于 1983 年 4 月 20 日

目　錄

第一部份　基本概念

第二部份　幾個應用上的觀念

第三部份　遊戲程式

第一章　APPLE II之一般概念

隨著資訊工業的進步，電腦已成爲人類生活中不可或缺的要件，尤其目前個人用電腦（PERSONAL COMPUTER）已深入每個家庭中。最初它的功用也許只是娛樂或是一些簡單的應用，但經過一段時間後，必然會興起自己編寫程式的興趣，至此使用者可能會發現，各家用電腦廠商所提供的程式語言均與其機器本身的系統及硬體結構有密切關聯。因此若想在程式編寫上發揮機器的最大能力，則首先必須對所使用的系統內部結構有深刻的認識才行。

基於此點，本書首先在本章中，對目前使用最普遍的APPLE II電腦的內部結構做一番描述。但因本書是針對APPLE的遊戲程式（game），故所介紹者均爲與此應用性程式有相關性的結構，至於其他較低階的硬體結構則不在本書討論範圍，由於本書所討論的game 程式並非簡單的應用性程式，相信本章所討論的結構已足夠應付讀者的需要。

第一節　APPLE II記憶區的配置

APPLE II 以6505 微處理機做爲中央處理單元，故其輸入／輸出爲memory-map I／O 的型式。即所有輸入／輸出埠被視爲記憶區，其所佔的位址也屬於記憶區的一部份，故可將其記

憶體分爲三大部份：讀／寫記憶區（RAM），僅讀記憶區（ROM），及輸入／出埠（I/O port）。此外APPLE將256個位元組劃分爲一頁（page），所以完整的64K記憶體共可分爲256頁，並按次序分別賦予0到255的頁數編號。由第0頁至第191頁共48K位元組（10^4 = 1024位元組稱爲1K位元組），屬於讀／寫記憶區；第192頁至207頁共4K位元組（ROM）的位置，則供輸入／出埠及各週邊裝置使用，最後的12K位元組，自第208頁至第255頁爲僅讀記憶區，儲存一些常用的系統程式，提供使用者使用但不能更改其內容。在APPLE II PLUS 這部份包含了—APPLESOFT（BASIC 解譯程式）及系統監督程式（MONITOR）。（請參照圖1-1）

系統記憶體結構圖

頁數 10進位	16進位	
0	$00	RAM（48K）
1	$01	
2	$02	
⋮	⋮	
190	$BE	
191	$BF	
192	$C0	I/O（2K）
193	$C1	
⋮	⋮	
198	$C6	
199	$C7	
200	$C8	I/O ROM（2K）
201	$C9	
⋮	⋮	

206	$CE	
207	$CF	
208	$D0	ROM（12K）
209	$D1	
⋮	⋮	
⋮	⋮	
254	$FE	
255	$FF	

圖 1-1　記憶體配置圖

由於讀／寫記憶區不但可以讀出資料又可存入資料，故爲使用者存放程式的區域，而僅讀記憶區，僅能讀出系統預先存好的程式，故其用途便有了限制。但在APPLE Ⅱ中，使用者可藉著插入一片語言卡（language card），（具有額外的16k RAM）可取代原有的僅讀記憶區，使讀／寫記憶區擴展爲64k位元組，供其他方面的使用。如我們想執行PASCAL所編寫的程式，則記憶區中除了要有使用者所編寫的PASCAL語言應用程式外，還必須存入PASCAL的編譯程式（compiler，用來將PASCAL語言轉換成機器所能執行的較低階語言），此時原來僅讀記憶區所存之BASIC解譯程式，對我們而言便毫無用處，便可利用此語言卡，取代此僅讀記憶區。

雖然APPLE Ⅱ具有48k位元組的讀／寫記憶區，但使用者也不是可全部隨意使用。第0頁（$0000－$00FF）在APPLE Ⅱ中爲十分特殊的記憶位置。在64k記憶體的機器，若想利用與記憶位址有關的組合語言指令（Assembly instruction）表示出某一位址，須使用16位元（$2^{16}=64\times1024=64k$）即兩個位元組；但對APPLE Ⅱ而言，若想表出第0頁的位址，只須使用8個位元（$2^8=256=1$ page）便可（詳細情形，請參

考書後所附組合語言指令表所列出的各指令所佔的位元組數），故監督程式及BASIC解譯器均用此頁儲存其所使用到的資料，以減少所佔的記憶空間及加快執行速度。

第一頁（$0100－$01FF）的256個位址爲系統堆疊區（stack），做爲6502一些特殊指令所使用的暫存區。如PHA，PHP PLA、PLP均爲單一位元組指令，僅表出動作（operation）而沒有指出動作執行（operand）之位址，即代表其動作執行位址爲系統已知，無須再指明。而這些“已知位置”便是在此堆疊區的特定位置；此外在程式中常用到的CALL、GOSUB及RETURN指令亦利用到此頁記憶區。如在一BASIC程式中：

```
 20  GOSUB 100
100  GOSUB 500
110  X = X + 1
500
                    ┐
                    } 一副程式
700  RETURN         ┘
```

GOSUB 500 後，程式便跳至500去執行，最後到700的RETURN時，系統是如何知道應回去執行110的x = x + 1 而不是到20行的下一個指令呢？這便是因系統在由100跳到500去執行時，便已先將此GOSUB 500 的下一個指令的起始位址存入此堆疊區中，RETURN 時只須再將此位址取出（POP）便可按正常的次序執行。諸如此類的例子還有許多，因不在此書討論範圍，故不再詳述。

第二頁（$0200－$02FF）爲鍵盤輸入緩衝區，將由鍵盤送過來的ASCⅡ電碼存入此緩衝區中，直到“RETURN”爲止；例如我們在鍵盤上鍵入＊400.500 RETURN時，系統會由4、0、……逐字“收集”，而第二頁就用來暫時存放這些“收集”的字元，直到使用者鍵入“RETURN”通知系統命令已輸入完畢後，系統才加以處理，並準備再從$0200起收集下一道命令。但因一頁僅具有256個位址，故每一行程式連RETURN在內，不能超過256個字數（空格也算）。因此若將此鍵盤輸入緩衝區擴展成爲512個位址，則每一行程式的字數最多便可達到512個字了，但因實際上很少有這麼長的命令，且每行程式太長了便不易閱讀，故通常均將此鍵盤輸入緩衝區定爲256個位址。

第三頁（$0300－$03FF）的256個位址中，監督程式僅使用最後的16個位元組，故大部份的位址均可由使用者自由使用。通常game程式便常利用此部份儲存一些不太長的機器語言程式，讀者可在往後的程式中發現這一點。

第四頁至第七頁（$0400－$07FF）共1K位元組的位址，爲供字幕或低解像圖面第1頁所使用；而在第8頁至第11頁（$800－$BFF）的1K位元組位址，則供字幕或低解像圖面的第2頁所使用，但通常第8頁爲使用者使用區域的起始頁數，即我們很少利用此區域顯示低解像圖形或文字幕。

雖然自$800起至記憶體結束位址，均可由使用者存入程式或資料，但若程式中使用到高解像畫面，則記憶體自$2000至$3FFF共8K位址便須由高解像畫面的第一頁所使用，而$4000到$5FFF共8K位址便由高解像畫面的第2頁所使用。綜合以上所述可歸納如表1-1。

表 1-1 APPLE Ⅱ記憶區之配置

頁 數 (DEC)	位址 10 進位	位址 16 進位	用途
0	0－255	$0 －$0FF	系統程式使用（MONITOR，BASIC…）
1	256－511	$100 －$1FF	系統堆疊區
2	512－767	$200 －$2FF	鍵盤輸入緩衝區
3	768－1023	$300 －$3FF	自由使用及監督程式使用
4 5 6 7	1024－2047	$400 －$7FF	文字幕及低解像圖面第 1 頁
8 9 10 11	2048－3071	$800 －$BFF	文字幕及低解像圖面第 2 頁
12 至 31	3072－8191	$C00 －$1FFF	自由使用區
32 至 63	8192－16383	$2000－$3FFF	高解像畫面第 1 頁
64 至 95	16384－24575	$4000－$5FFF	高解像畫面第 2 頁
96 至 191	24576－49151	$6000－$BFFF	
192	49152－49279	$C000－$C07F	特殊內部 I/O 位址
	49280－49407	$C080－$C0FF	週邊插座使用
193 至 199	49408－51199	$C100－$C7FF	週邊 ROM
200 至 207	51200－53247	$C800－$CFFF	週邊擴展用 ROM
208 至 247	53248－63487	$D000－$F7FF	BASIC 解譯程式 ROM
248 至 255	63488－65535	$F800－$FFFF	監督程式 ROM

第二節　螢幕與第一頁文字幕記憶區

一部電腦可能不具有畫面結構，但文字幕卻是各電腦所必須的，用來與使用者相互溝通。因此在APPLE Ⅱ開機之後，監督程式會立刻設定螢幕爲文字幕形式，並輸出一閃爍的空白字形在螢幕上，告訴使用者下一個輸入文、數字的顯現位置，此即一般所稱的游標（ cursor ）。故當使用者輸入命令至電腦時，實際上電腦要做下列幾件事：首先測試鍵盤是否有輸入？若有則依所輸入的字鍵，將其存入文字幕中適當位置（或採取適當動作），而顯示在螢幕上，並令游標向右移動一個位置，當然還要將此文、數字或特殊符號以ASC Ⅱ的電碼形式存入第 2 頁（ $200－$2FF），以逐字搜集命令，供系統執行。

可能說到這裡有些讀者還不了解文字幕是什麼？簡單的說文字幕，就是位於系統讀／寫記憶區的一塊位址，電腦硬體能自動將其內容顯示在螢光幕上，故我們按其意義稱它爲“幕”。由於具有此文字幕儲存了使用者的輸入字鍵，所以電腦能很簡單的在此位址上從事一些修改先前輸入字鍵或跳行的工作（如當使用者按下“→”，“←”或RETURN等鍵時）。但在正式討論文字幕的記憶區結構時，我們想先對APPLE Ⅱ 的輸入／出字形編碼做一番說明。

事實上；一部電腦內部所能儲存的只是以0,1構成的一些二進位型式的數字資料，因此爲了要能將文字幕記憶區的資料以字形、符號的方式印在螢幕上，便須對這些字形、符號定出一套對應的編碼，如此才能使電腦硬體在讀到 193時能在螢幕上顯現“A”，讀到 163時顯示出“＃”。目前較爲大家所熟知的是一套ASCⅡ編碼，但APPLE Ⅱ所使用的編碼方式與ASC Ⅱ 編碼略

有不同。在監督程式中，APPLE 是以負的ASCⅡ碼（即ASCⅡ碼加上128）方式存入。這種方式使得儲存此輸入碼的記憶位址的最高位元（bit 7），若有輸入則變爲1，否則便爲0。當然APPLE的監督程式在取出此輸入碼後，必須將其最高位元再變爲0。控制這些動作主要有二個位址：一爲$C000（－16384）即爲APPLE 的鍵盤按鍵字碼的存入位址（keyboard data input），若其內含值超過127（即bit 7爲1）則代表有鍵被按了；當然將此值減去128即是標準的ASCⅡ 值。此外還有一個位址爲$C010（－16368）爲鍵盤旗標（clear keyboard strobe），用來將鍵盤觸發（strobe）訊號歸零。即當鍵盤上有鍵被按了，－16384 位址上的第7位元被設定爲1，必須再經由LDA $C010的指命（或POKE－16368，0），將其第7位元歸零，以便繼續讀入其他按鍵。因爲APPLE 的監督程式，在測試是否有按鍵輸入時，僅測試$C000位址的第7位元是否爲1，而不管其內容。故使用者應特別注意，若程式中由－16384（$C000）位址讀取資料後，應記得將鍵盤觸發訊號歸零（POKE－16368，0或LDA $C010），才能接受到下次新的按鍵輸入字碼。

我們知道APPLE 每個位址可存8個位元資料，2^8 爲256故每個位址所能表示的碼數共有256 種，但螢幕上可印出的字形只有64個（見表1-2）。所以APPLE 將低於128的碼分成兩部份，分別對應於反白字形及閃爍字形各64 個；而將128至159的碼編當做特殊控制碼；160～223的碼供64個正常字形之用；及224至255的碼保留給小寫字母使用（見表1-2）。但因APPLEⅡ的字形產生器無法產生小寫字母，故一般的APPLEⅡ的這些小寫字母只是重覆64個正常字形而已。此外讀者

應注意表1-2是螢幕字形碼，在由鍵盤輸入的字碼中沒有反白及閃爍的字形，只有正常字形，故由鍵盤輸入時，其輸入鍵碼必大於127，不會有小於128的情形。且讀者應清楚一項事實，並不是每一個螢幕字形碼皆能顯現在螢幕上，如CONTROL-G僅能使喇叭發聲，而不會在螢幕上產生動作。

表1-2

	Inverse				Flashing				Normal (Control)								(Lowercase)	
Decimal	0		16		32		48		64	80	96	112	128	144	160	176	192	208	224	240
Hex	$00		$10		$20		$30		$40	$50	$60	$70	$80	$90	$A0	$B0	$C0	$D0	$E0	$F0
0 $0	@	P		0	@	P		0	@	P		0		0	@	P		0		0
1 $1	A	Q	!	1	A	Q	!	1	A	Q	!	1	!	1	A	Q	!	1	!	1
2 $2	B	R	"	2	B	R	"	2	B	R	"	2	"	2	B	R	"	2	"	2
3 $3	C	S	#	3	C	S	#	3	C	S	#	3	#	3	C	S	#	3	#	3
4 $4	D	T	$	4	D	T	$	4	D	T	$	4	$	4	D	T	$	4	$	4
5 $5	E	U	%	5	E	U	%	5	E	U	%	5	%	5	E	U	%	5	%	5
6 $6	F	V	&	6	F	V	&	6	F	V	&	6	&	6	F	V	&	6	&	6
7 $7	G	W	'	7	G	W	'	7	G	W	'	7	'	7	G	W	'	7	'	7
8 $8	H	X	(	8	H	X	(	8	H	X	(	8	(	8	H	X	(	8	(	8
9 $9	I	Y	)	9	I	Y	)	9	I	Y	)	9	)	9	I	Y	)	9	)	9
10 $A	J	Z	*	:	J	Z	*	:	J	Z	*	:	*	:	J	Z	*	:	*	:
11 $B	K	[	+	;	K	[	+	;	K	[	+	;	+	;	K	[	+	;	+	;
12 $C	L	\	,	<	L	\	,	<	L	\	,	<	,	<	L	\	,	<	,	<
13 $D	M	]	-	=	M	]	-	=	M	]	-	=	-	=	M	]	-	=	-	=
14 $E	N	^	.	>	N	^	.	>	N	^	.	>	.	>	N	^	.	>	.	>
15 $F	O	_	/	?	O	_	/	?	O	_	/	?	/	?	O	_	/	?	/	?

表2-0

討論了APPLE的輸入及螢幕字形碼後，我們便可接著研究文字螢幕與記憶區的對應關係。在上一節中我們提過APPLE II的文字幕與低解像圖面共同使用$400至$7FF共1K的位置。而文字幕與低解像圖面結構在與記憶區的對應關係上大致是相同的；且在編寫game程式中有時無法很容易的用PLOT等一些繪圖指令將複雜的圖形畫出或控制圖形的移動，此時若能明白螢幕與記憶位址的對應關係，便可很容易的解決此問題。

在文字幕中，APPLE II能顯示24行文字且每行能印出

40個字形，即總共能印出 $24 \times 40 = 960$ 個字。而每個字用去一個儲存位址，共用了960個位址，但文字幕的記憶區共佔1024個位置，還有 $1024 - 960 = 64$ 個位址呢？我們發現它是以8個位址爲一組共分8組，每組位址提供給一個週邊插座(sot)使用，此外螢幕上的次序是否正好與記憶區的順序相配合呢？事實上不是。我們可做個小小的試驗，讀者可試試下列指令：

```
 0  CALL -936
10  FOR I=0 TO 255
20  POKE 1024+I , I
30  NEXT I
40  VTAB 20 : END
```

在上面程式中，將螢幕字形表中的各字形按文字幕記憶區的位址順序分別存入，以便由執行中發現一些有趣的現象。我們可發現：第一、螢幕上24列文字並非依位址順序排列，而是將其分爲三部份（1～8列、9～16列、17～24列）。顯現時每一部份依次序輪流各印一行後再重覆，即第1列印過後，印第9列、後再印第17列，然後再印第2列……。第二、有一部份字形沒有顯示在螢幕上（如8、9、：、；、=、？）。因此儲存這些字形碼的位址可能在960個顯現位址之外，即此960個顯現位址並非連續，而是將64個週邊插座所用的位址分散在其間。事實上文字幕的位址結構如表1-3。由表1-3(a)我們看到當第17列顯示完後，以下120～127等8個字形碼（8、9、：、；、<、=、>、？）被存入$478至$47F，而此位址爲供I/O之用，而不是用來做螢幕顯示之用的，這解釋了爲什麼有些字形會不見了的原因。

表1-3 文字幕位址分配

Table No. 1 - In Memory Order Sequence			Table No 2 - In Screen Line Sequence		
DEC:	HEX:	LINE No.	DEC:	HEX:	LINE No.
1024-1063	400-427	1	1024-1063	400-427	1
1064-1103	428-44F	9	1152-1191	480-4A7	2
1104-1143	450-477	17	1280-1319	500-527	3
1144-1151	478-47F	*	1408-1447	580-5A7	4
1152-1191	480-4A7	2	1536-1575	600-627	5
1192-1231	4A8-4CF	10	1664-1703	680-6A7	6
1232-1271	4D0-4F7	18	1792-1831	700-727	7
1272-1279	4F8-4FF	*	1920-1959	780-7A7	8
1280-1319	500-527	3	1064-1103	428-44F	9
1320-1359	528-54F	11	1192-1231	4A8-4CF	10
1360-1399	550-577	19	1320-1359	528-54F	11
1400-1407	578-57F	*	1448-1487	5A8-5CF	12
1408-1447	580-5A7	4	1576-1615	628-64F	13
1448-1487	5A8-5CF	12	1704-1743	6A8-6CF	14
1488-1527	5D0-5F7	20	1832-1871	728-74F	15
1528-1534	5F8-5FF	*	1960-1999	7A8-7CF	16
1536-1575	600-627	5	1104-1143	450-477	17
1576-1615	628-64F	13	1232-1271	4D0-4F7	18
1616-1655	650-677	21	1360-1399	550-577	19
1656-1663	678-67F	*	1488-1527	5D0-5F7	20
1664-1703	680-6A7	6	1616-1655	650-677	21
1704-1743	6A8-6CF	14	1744-1783	6D0-6F7	22
1744-1783	6D0-6F7	22	1872-1911	750-777	23
1784-1791	6F8-6FF	*	2000-2039	7D0-7F7	24
1792-1831	700-727	7	1144-1151	478-47F	*
1832-1871	728-74F	15	1272-1279	4F8-4FF	*
1872-1911	750-777	23	1400-1407	578-57F	*
1912-1919	778-77F	*	1528-1535	5F8-5FF	*
1920-1959	780-7A7	8	1656-1663	678-67F	*
1960-1999	7A8-7CF	16	1784-1791	6F8-6FF	*
2000-2039	7D0-7F7	24	1912-1919	778-77F	*
2040-2047	7F8-7FF	*	2040-2047	7F8-7FF	*

明白了文字幕與記憶位址的對應關係後，我們知道並不是我們直覺上所認爲的1024個位址中的前960個位址對應到螢幕上