VIC da zero +64

di Tommaso Pantuso

Salviamo lo schermo

Nell'articolo di questo mese metteremo insieme tutte le nozioni apprese nei due numeri precedenti, riguardanti la memoria video e il buffer della tastiera, e cercheremo di trovare un metodo per codificare in maniera intelligibile a tutti, ciò che appare sul video, ad esempio i disegni che utilizzano i caratteri grafici Commodore che non sempre è facile interpretare quando, integrati in un programma, servono a produrre una schermata complessa. Inoltre avremo modo in seguito di applicare quanto sappiamo sul video per ottenere la memorizzazione su disco di intere schermate.

Un breve riepilogo

Abbiamo visto che esiste in memoria Ram, sia del Vic che del 64, una zona che riproduce, con un opportuno codice, tutto quanto compare sullo schermo. In pratica, se ad esempio sul video è possibile posizionare 1000 caratteri (40 colonne × 25 righe), in memoria troveremo una zona di 1000 byte, ciascuno dei quali corrispondente ad una determinata locazione dello schermo, contenenti dei numeri che è abbastanza facile mettere in corrispondenza con quanto effettivamente vediamo rappresentato. Ad esempio alla lettera A corrisponde il numero 1, alla B il numero 2 e così via. Il processo è naturalmente invertibile, nel senso che se noi andiamo a scrivere nella memoria video (ad esempio con una Poke) un numero che rappresenta il codice di un carattere, il carattere rappresentato comparirà sullo schermo in una certa posizione facilmente individuabile. In effetti abbiamo osservato che, per vedere effettivamente qualcosa, dobbiamo assegnare al carattere in questione un colore, cosa semplicemente ottenibile memorizzando un opportuno codice in un'altra zona Ram, la memoria di colore. In definitiva è come avere un sandwich a tre strati: il primo rappresenta lo schermo TV, il secondo la memoria video e il terzo quella di colore; per ottenere un carattere sul primo strato bisogna definire gli opportuni parametri sugli altri due.

Abbiamo anche visto l'importanza che assume quella zona di memoria chiamata Buffer di tastiera che, usata opportunamente, permette alla macchina, in un certo senso, di autoprogrammarsi in quanto è possibile, durante lo svolgimento del programma, aggiungere delle nuove linee senza arrestare l'elaborazione.

Vediamo allora ciò che possiamo produrre con tutte le nozioni accumulate negli ultimi articoli.

Schermo e "DATA"

Se possiamo scrivere in memoria video ed ottenere un'immagine sullo schermo, possiamo allora evitare di utilizzare l'istruzione Print. Ciò torna utile, come accennavamo prima, quando si vogliono comporre dei disegni la cui codifica, in un listato, sia facilmente interpretabile e quindi riproducibile. Cosa c'è allora meglio dei numeri per ottenere quanto vogliamo? Spieghiamoci meglio prendendo come punto di riferimento — senza perdere di generalità — il C 64, ritenendo abbastanza immediate le modifiche da effettuare sul Vic che in definitiva sono legate solo al formato dello

```
0 REM --- QUESTO PROGRAMMA INTRODUCE AUTOMATICAMENTE -
1 REM ---   DELLE LINEE DI DATA CHE CODIFICANO IL    -
2 REM --- CONTENUTO DI UNA SCHERMATA DEL C 64        -
3 POKE52,156:POKE56,156:POKE650,128
5 PRINT"□";
6 POKE53280,0:POKE53281,0:PRINT"■";
10 GETA$:IFA$=""THENGOSUB100:GOTO10
30 IFA$="■"THEN GOSUB 200          :REM TASTO F1
40 IFA$=CHR$(34)THEN10             :REM VIRGOLETTE
50 IFA$="■"THENGOSUB500            :REM TASTO F3
60 IFA$="■"THENPRINT"□":GOTO 1000:REM TASTO F5
90 PRINTA$;:GOTO10
98 END
99 REM ------------------------------------------------
100 PRINT"■ ■";
110 FORI=1TO50 :NEXT
120 PRINT"■■ ";
130 REM FORI=1TO100:NEXT
140 PRINT"■■";
150 RETURN
151 REM ------------------------------------------------
200 FORI=0TO959
210 POKE39936+I,PEEK(1024+I)
230 NEXT
240 RETURN
241 REM ------------------------------------------------
500 FORI=55296TO56295
510 POKEI,1:NEXT
520 FORI=0TO959
530 POKE1024+I,PEEK(39936+I)
540 NEXT
550 RETURN
551 REM ------------------------------------------------
1000 REM ------- AUTODATA ---------
1010 REM ---------------------------
1020 POKE252,0
1030 REM PRINT"□";
1040 A=39936+16*PEEK(252)
1050 PRINTA;"D♠ ";
1060 FORS=0TO15
1070 A$=A$+RIGHT$(STR$(PEEK(A+S)),3)+","
1080 NEXT:PRINTA$+"■■ ";
1090 POKE252,PEEK(252)+1
1100 POKE631,145:POKE632,145:POKE633,13
1110 POKE634,71:POKE635,207:POKE636,49
1120 POKE637,49:POKE638,52:POKE639,48
1130 POKE640,13:POKE198,10:END
1140 IF39936+16*PEEK(252)<39936+16*60THEN1030
1150 GOTO10
```

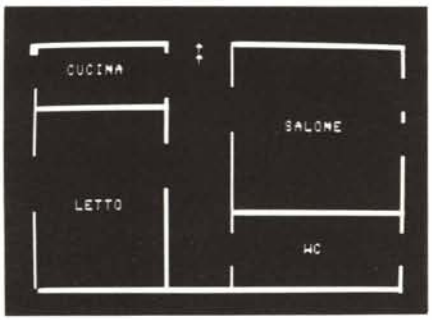
schermo ed alla diversa mappatura dello schermo in memoria.

Supponiamo di voler scrivere nella prima locazione dello schermo quella più in alto a sinistra, il simbolo delle picche (quello delle carte). L'istruzione da dare in un eventuale programma sarebbe:

```
PRINT"<Shift+A>"
```

Ciò naturalmente vale anche se si vuol scrivere qualunque altro carattere o stringa di caratteri grafici ma, mentre alcuni dei caratteri che compaiono nei listati sono facilmente individuabili sulla tastiera, altri lo sono un po' meno e quando essi compaino in numero massiccio a volte non si riesce ad ottenere esattamente l'effetto desiderato dopo aver ricopiato un listato su cui sono presenti delle schermate grafiche introduttive o altre cose del genere.

Dato che ormai siamo pratici di memoria di schermo, viene abbastanza logico pensare che lo stesso effetto potrebbe essere ottenuto andando a scrivere nella opportuna locazione il codice delle picche



(65) ed abilitando convenientemente il colore. In altre parole, con

```
POKE 1024,65
```

(supponendo per comodità già abilitato il colore) vedremo comparire in alto a sinistra il carattere desiderato senza aver fatto nessun riferimento esplicito al simbolo grafico. Vi ricordiamo che da 1024 inizia la memoria di schermo per il 64.

Se poi la stringa è più complessa, le cose non cambiano affatto in quanto basterà scrivere in tutte le locazioni interessate il codice del carattere desiderato. Ad esempio:

```
POKE1024,83:POKE1025,90
POKE1026,88:POKE1027,65
```

farà comparire, nelle prime quattro locazioni dello schermo, rispettivamente i caratteri cuori, quadri, fiori e picche, (supponendo sempre di aver abilitato il colore).

Naturalmente risulta abbastanza scomodo andare a ricavare dalle tabelle i codici di ciascun carattere che compone il disegno che vogliamo ottenere e scriverlo con delle Poke nelle precise locazioni di memoria, ma non è esattamente questo che noi vi proponiamo. Se potessimo ottenere, partendo da un'immagine dello schermo in memoria, dei codici introdotti automaticamente in un certo numero di linee Data, potremmo poi facilmente da questi ricostruire lo schermo andando a "pokare" i dati nelle locazioni di schermo interessate. Ci spieghiamo meglio.

Considerando l'esempio precedente, per quanto abbiamo detto, partendo dai simboli cuori, quadri, fiori e picche scritti nelle prime quattro locazioni dello schermo, dovremmo poter ottenere automaticamente nel programma, partendo dalla stringa in questione, una linea:

```
DATA 83,90,88,65
```

attraverso la quale potremmo ricostruire le prime quattro locazioni di schermo con un segmento del tipo:

```
10 FOR I=0TO3:READ A:POKE1024+I,A
20 DATA 83,90,88,65
```

Naturalmente il processo è riconducibile a tutto lo schermo nel senso che se codifichiamo mille numeri con delle linee Data, cambiando il ciclo For... Next, avremo accesso a tutte le locazioni immagine.

Questo processo, per essere comodo dovrebbe essere automatizzato a partire da una schermata nel senso che, una volta composto il disegno sullo schermo, da questo dovrebbero essere ricavate delle linee Data e introdotte automaticamente nel programma. È proprio l'implementazione di questo procedimento che vogliamo spiegarvi nelle righe che seguono.

AutoDATA

Il processo sintetizzato lo chiameremo d'ora in poi AutoDATA. Esso è ottenibile con una certa disinvoltura impiegando tutte le nozioni apprese sul comportamento del Buffer di tastiera in certe situazioni. Per meglio comprendere il procedimento d'insieme, facciamo rifeimento alle linee che vanno da 1000 a 1150 del solito programma dimostrativo riportato nel listato 1 con il quale è possibile comporre un disegno o delle scritte sullo schermo e poi trasformare il contenuto della sua immagine in memoria in linee di dati. Ma cominciamo a commentare l'ultima parte, racchiusa tra le linee indicate e che rappresenta il cuore del programma, considerandola un programma a sé.

Vogliamo inserire in un certo numero di

```
39936 DATA  32,120,120,120,120,120,120,120,120,120,120,120,120,120, 97, 32
39952 DATA  32, 30, 32, 32,118,120,120,120,120,120,120,120,120,120,120,120
39968 DATA 120,120,120,120,120,120, 97, 32, 32, 32, 32, 32, 32, 32, 32, 32
39984 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 30, 32, 32,118, 32, 32
40000 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 97, 32
40016 DATA  32, 32, 32, 32, 3, 21, 3, 9, 14, 1, 32, 32, 32, 32, 32, 32
40032 DATA  32, 32, 32, 32,118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40048 DATA  32, 32, 32, 32, 32, 32, 97, 32, 32, 32, 32, 32, 32, 32, 32, 32
40064 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,118, 32, 32, 32
40080 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40096 DATA 118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40112 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40128 DATA  32, 32, 32, 32, 32, 32, 32, 32,118,121,121,121,121,121,121,121
40144 DATA 121,121,121,121,121,121, 97, 32, 32, 32, 32, 32, 32, 32, 32, 32
40160 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40176 DATA 118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 97, 32
40192 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40208 DATA  32, 32, 32, 32, 32, 32, 97, 32,118, 32, 32, 32, 32, 32, 32, 32
40224 DATA  32, 32, 32, 32, 32, 32, 97, 32, 32, 32, 32, 32, 32, 32, 32, 32
40240 DATA  32, 32, 19, 1, 12, 15, 14, 5, 32, 32, 32, 32, 32, 32, 32, 32
40256 DATA 118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 97, 32
40272 DATA  32, 32, 32, 32,118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40288 DATA  32, 32, 32, 32, 32, 32, 32, 32,118, 32, 32, 32, 32, 32, 32, 32
40304 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,118, 32, 32, 32
40320 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40336 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40352 DATA  32, 32, 32, 32,118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40368 DATA  32, 32, 32, 32, 32, 32, 97, 32, 32, 32, 32, 32, 32, 32, 32, 32
40384 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,118, 32, 32, 32
40400 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 97, 32
40416 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40432 DATA  32, 32, 32, 32,118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40448 DATA  32, 32, 32, 32, 32, 32, 97, 32, 32, 32, 32, 32, 32, 32, 32, 32
40464 DATA  32, 32, 32, 32, 32, 32, 97, 32, 32, 32, 32, 32,118, 32, 32, 32
40480 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 97, 32
40496 DATA  32, 32, 32, 32, 32, 12, 5, 20, 20, 15, 32, 32, 32, 32, 97, 32
40512 DATA  32, 32, 32, 32,118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40528 DATA  32, 32, 32, 32, 32, 32, 32, 97, 32,118, 32, 32, 32, 32, 32, 32
40544 DATA  32, 32, 32, 32, 32, 32, 97, 32, 32, 32, 32, 32, 32,118,120,120,120
40560 DATA 120,120,120,120,120,120,120,120,120,120,120,120,120,120, 97, 32
40576 DATA 118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 97, 32
40592 DATA  32, 32, 32, 32,118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40608 DATA  32, 32, 32, 32, 32, 32, 97, 32,118, 32, 32, 32, 32, 32, 32, 32
40624 DATA  32, 32, 32, 32, 32, 32, 32, 97, 32, 32, 32, 32, 32, 32, 32, 32
40640 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40656 DATA 118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 97, 32
40672 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 23, 3, 32, 32
40688 DATA  32, 32, 32, 32, 32, 32, 32, 32,118, 32, 32, 32, 32, 32, 32, 32
40704 DATA  32, 32, 32, 32, 32, 32, 97, 32, 32, 32, 32, 32, 32, 32, 32, 32
40720 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40736 DATA 118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 97, 32
40752 DATA  32, 32, 32, 32,118, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40768 DATA  32, 32, 32, 32, 32, 32, 97, 32,118, 32, 32, 32, 32, 32, 32, 32
40784 DATA  32, 32, 32, 32, 32, 32, 97, 32, 32, 32, 32, 32,118, 32, 32, 32
40800 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 97, 32
40816 DATA  32,120,120,120,120,120,120,120,120,120,120,120,120,120,120,120
40832 DATA 120,120,120,120,120,120,120,120,120,120,120,120,120,120,120,120
40848 DATA 120,120,120,120,120,120,126, 32, 32, 32, 32, 32, 32, 32, 32, 32
40864 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
40880 DATA  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
```

1 Data che codificano il disegno della foto in alto a sinistra.

linee Data il contenuto del pezzo di memoria che va dalla locazione 39936 in su per 960 byte: naturalmente potrebbe trattarsi di qualsiasi altra zona di memoria. La prima cosa importante che viene fatta (1020) è quella di azzerare una locazione di Ram non utilizzata dal sistema operativo per memorizzarvi una variabile che ci farà comodo nel corso dell'elaborazione. Nella linea 1040 viene definita una variabile

```
A=39936+16*PEEK(252)
```

che verrà poi incrementata di 16 unità per volta incrementando di un'unità il contenuto della locazione 252 (1090). In altre parole, quando PEEK (252) = 0 (contenuto della locazione 252), allora A = 39936; quando PEEK (252) = 1 segue che A vale 39952 e così via (se ancora non capite non preoccupatevi perché tra breve metteremo insieme tutti gli elementi). Con la linea 1050 viene scritta sullo schermo la stringa di due caratteri che compare tra virgolette e che rappresenta l'abbreviazione della parola chiave "DATA" mentre dalla linea 1100 a 1130 che sintetizziamo nella seguente tabella:

Loc.	Cod.	Carattere
631	145	curs. su
632	145	curs. su
633	13	Return
634	71	G
635	207	Shift + 0
636	49	1
637	49	1
638	52	4
639	48	0
640	13	Return

La sequenza rappresentata, come è facile osservare, oltre a spostare il cursore due posizioni verso l'alto genera anche un GOTO 1140. Ciò viene fatto per una ragione che abbiamo discusso la volta scorsa ma su cui vogliamo ritornare per maggiore chiarezza. Dato che il Buffer della tastiera viene svuotato solo dopo un "END" (lo trovate in linea 1130 dopo POKE 198,10 che indica il numero di caratteri che devono essere scaricati sullo schermo), questa necessità produrrebbe un arresto del programma e noi vogliamo che ciò si verifichi al fine di rendere automatica tutta la procedura. Introduciamo allora l'invio alla linea 1140 che fa ripartire il programma da tale posizione.

Capite ora l'importanza di aver conservato il valore del contatore del numero di linea Data in una locazione Ram con una Poke e non come una variabile nel modo consueto: con un Goto, tale variabile verrebbe perduta.

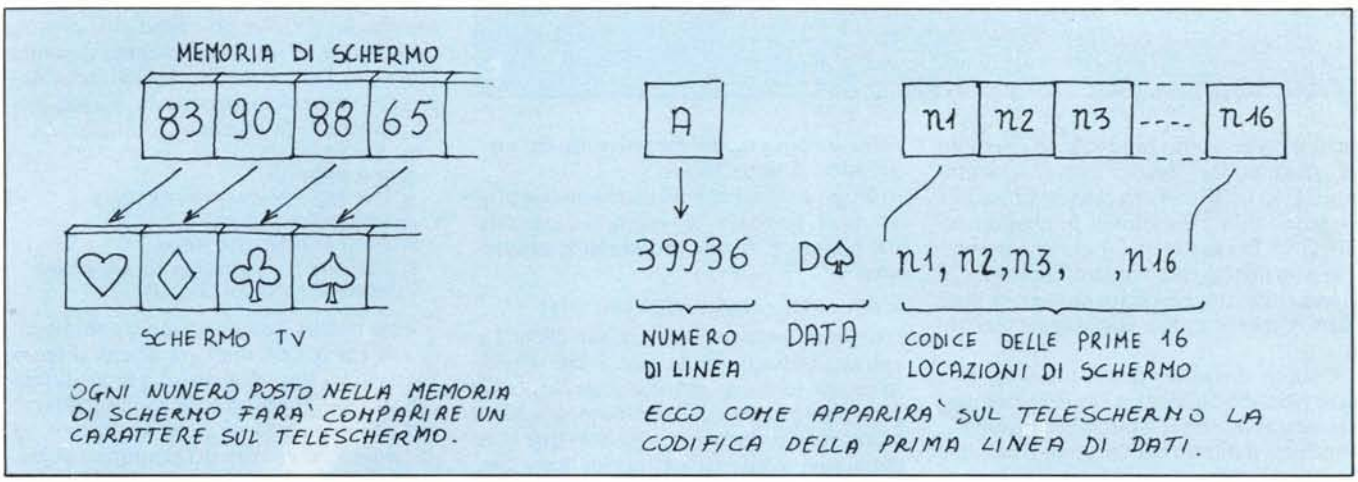
Tutto il procedimento viene ripetuto per un numero di volte tale da coprire tutta l'area interessata.

Vogliamo farvi notare che un tale sistema di conversione potrebbe essere utilizzato anche per introdurre in linee Data un programma in Lm per poi utilizzarlo in un caricatore Basic.

Analizzata la parte più importante del programma passiamo ad illustrare le parti principali di ciò che rimane.

Per prima cosa viene protetto da intrusioni del Basic 1K di memoria posto nella parte più alta di quella disponibile per il programma (linea 3). La linea 6 cambia il colore dello schermo e del carattere; la 10 mette il programma in attesa oltre ad inviare, ad intervalli regolari, ad una subroutine (100-150) che simula il lampeggio del cursore, cosa che sfrutteremo per orientarci sullo schermo nel corso della composizione del disegno. Vediamo ora le opzioni delle linee 30, 50 e 60.

Premendo il tasto f1 il contenuto dello schermo viene copiato nella zona precedentemente protetta che comincia dalla locazione 39936 (156*256). Questo perché durante il processo di autoDATA, per quanto detto precedentemente, il contenuto della memoria schermo vera e propria andrebbe perduto. Premendo f3 è invece possibile ricostruire sul monitor il contenuto dello schermo dopo la precedente operazione di copiatura. Come è immediato constatare, il contenuto della zona protetta è scaricato in memoria video e viene



1060 alla 1080 viene composta una stringa, A$, contenente i numeri situati nelle prime 16 locazioni da codificare, separati da una virgola.

In base a questi primi elementi, vediamo come si comporta il programma fino alla linea 1090.

Azzerata la locazione 252 e cancellato lo schermo viene calcolato il primo valore di A; poi vengono scritte sullo schermo: A, D + <il simbolo di picche>, A$ e viene incrementata la locazione 252. In definitiva a questo punto sullo schermo vedremo scritto:

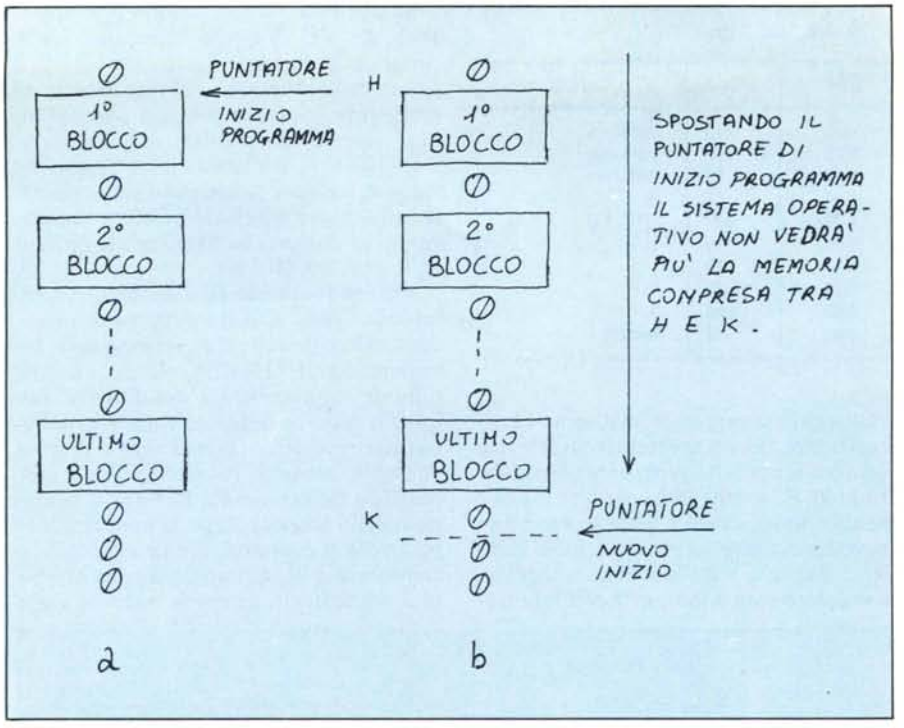
```
39936 DATA n1, n2, ... n16
```

dove n1... n16 sono i contenuti delle prime 16 locazioni interessate. Specifichiamo che la parola chiave DATA verrà scritta in forma abbreviata (D+picche) ma noi la riportiamo per intero per esigenze tipografiche.

A questo punto ci vuole una opportuna sequenza scritta nel Buffer di tastiera che (come già sapete) porti il cursore sulla linea scritta e generi un Return affinché essa sia introdotta nel programma. Ciò è ottenuto con la sequenza contenuta nelle linee da riempita di "1" la memoria del colore. Vi facciamo osservare che abbiamo preferito non implementare la codifica dell'ultima linea video che non potrebbe essere riprodotta fedelmente sia per la presenza su di essa del cursore sia per la necessità di lasciare l'ultima locazione di tale riga in bianco onde evitare lo scroll verso l'alto a causa dell'"a capo" generato impegnando l'ultima locazione del video. Infine il tasto f5 avvia il procedimento di codifica che già conosciamo.

Come isolare le linee di dati

Una volta ottenute le linee di programma con i Data dovremo in qualche modo isolarle ed introdurle in un contesto che le renda utilizzabili. La prima cosa che ci viene da pensare è — partendo dal programma già elaborato contenente le linee aggiuntive — di cancellare le linee che precedono la 39936 (la prima linea di dati) scrivendo semplicemente i numeri delle righe che vogliamo cancellare seguiti da Return. Questo primo procedimento non ci lascia comunque molto soddisfatti: certo



tutto sarebbe molto più facile se avessimo un comando di "Delete" come, ad esempio, il C 16 o il Plus 4, ma purtroppo un tale comando non c'è e quindi dobbiamo arrangiarci. Pensando un po' ci viene in mente che un metodo per rendere "invisibili" al sistema operativo un certo numero di linee esiste ed è abbastanza semplice da mettere in atto.

Oramai dovreste sapere tutto (se non altro perché ne abbiamo ampiamente parlato in questa stessa rubrica) su alcuni dei principali puntatori del sistema e su come viene formattato un programma in memoria. Riferendoci — sempre per fissare le idee — al 64, l'area utilizzata per i programmi inizia dalla locazione 2049 (in effetti dalla 2048 che contiene il flag 0) ed è identificabile leggendo il valore di un puntatore a due byte contenuto nelle locazioni 43 e 44 nelle quali, all'accensione, troveremo i numeri 1 e 8. Da una rapida verifica ricaviamo infatti che:

1 + 8•256 = 2049.

Il sistema comincia il suo lavoro sul programma a partire dal punto indicato dal contenuto di queste due locazioni. In memoria, ogni linea rappresenta un blocco a sé il quale comprende dei byte che permettono al sistema operativo la concatenazione dei vari blocchi. Ciascuno di essi inoltre è separato dal successivo per mezzo di uno "0" ed infine il programma termina ponendo la serie "000" in calce all'ultimo dei blocchi in questione. La fine del programma viene puntata dal contenuto delle locazioni 45 e 46.

Se noi, avendo un programma in memoria, manipoliamo opportunamente i puntatori descritti, potremo ingannare facilmente il sistema: sfrutteremo questo fatto per i nostri scopi. Cominciamo le nostre osservazioni e modifiche partendo dal programma di autoDATA.

Dopo aver caricato in memoria tale programma, proviamo a leggere il contenuto dei puntatori di fine programma; otterremo:

```
PEEK (45) = 240 e PEEK (46) = 11
```

cioè viene puntata la locazione 3056. La prima osservazione da fare è che il programma termina praticamente se comprendiamo lo zero di fine blocco, alla locazione 3053 (compresa) e gli altri due byte aggiuntivi si riferiscono ai due zero che, insieme a quello precedente, realizzano la sequenza "000" cui accennavamo poc'anzi. Appuntiamoci i valori 240, 11 e 3053 e andiamo avanti. Disegnamo ciò che vogliamo sullo schermo e avviamo la sequenza di autoDATA: per quanto detto le linee aggiuntive verranno aggiunte a partire dalla locazione 3053 ed i due zero verranno spostati alle fine della codifica dell'ultimo blocco di tali linee. Naturalmente tutto ciò avviene in modo a noi del tutto trasparente. A questo punto, se vogliamo far ignorare al sistema tutto ciò che precede le linee con i Data, basterà comunicargli che il programma non inizia più dalla locazione 2049 ma dalla 3053 e questo è semplicemente ottenibile con:

```
POKE43,238:POKE44,11
```

Se ora date il List, vedrete che non compariranno più le linee precedenti la 39936 e quindi avremo isolato tutte quelle con i Data. Per mettere in funzione autonomamente ciò che avete ricavato, provate a questo punto ad aggiungere il seguente segmento:

```
10 FORI = 55296 TO 56295
20 POKE I,1: NEXT
30 FORI = 0TO959:READ A
40 POKE1024 + I,A:NEXT
50 GOTO 50
```

e a dare il Run: vedrete il vostro disegno ricomparire sullo schermo.

Un semplice APPEND

È evidente che potreste impiegare le linee appena isolate in un altro programma e sarebbe scomodo doverle ribattere tutte in macchina. Se avessimo un comando di Merge o di Append potremmo ottenere lo scopo semplicemente ma, dato che non ne disponiamo, dovremo anche questa volta inventarci qualche cosa. Di seguito vi proponiamo una sequenza che permette di appendere un programma in coda ad un altro: l'unica condizione perché ciò possa verificarsi è che i due programmi non devono avere linee etichettate nello stesso modo ed inoltre il massimo numero di linea del primo programma deve essere inferiore al minimo numero del secondo. Nel nostro caso questa condizione si verifica abbastanza naturalmente essendo le linee numerate da 39936 in poi; negli altri casi, se siamo noi a scrivere i programmi da unire l'uno con l'altro, faremo attenzione a numerarli in maniera opportuna. La sequenza proposta è la seguente:

```
1) A = PEEK(43)
2) B = PEEK(44)
3) C = 256•PEEK(46) + PEEK(45)-2
4) POKE43,C AND 255
5) POKE44,INT(C/256):NEW
6) Caricare il programma da appendere
7) POKE43,A:POKE44,B:CLR
```

Essa non ha bisogno di molti commenti: A e B per il C64 sono 1 e 8; con il terzo passaggio viene ricavato il punto da cui cominciare l'Append mentre con il quarto ed il quinto tale punto viene scomposto nei due valori da inserire nel puntatore di inizio programma ed azzerato quello di fine programma. Ciò permette al programma che sarà caricato di non andarsi a sovrapporre a quello già in memoria. Dopo il caricamento viene riabbassato il puntatore 43/44 scoprendo così entrambi i programmi. Il procedimento descritto può essere ripetuto più volte per appendere programmi fino a riempire tutta l'area di memoria disponibile.

Per il Vic

Tutti i discorsi fatti finora per il C 64 restano concettualmente gli stessi per il Vic 20. Naturalmente bisogna tener presente alcune differenze tra l'una e l'altra macchina tra cui le più rilevanti sono il diverso formato di schermo e la mappa della memoria che varia a seconda dell'espansione posseduta. Naturalmente, chi ci ha sempre seguiti troverà facilmente le modifiche da effettuare, ma anche gli altri sicuramente non troveranno molte difficoltà.

Sperando di aver stimolato in qualche modo la vostra fantasia vi diamo appuntamento al prossimo numero sempre ricco di interessanti informazioni sul Vic e sul C 64.

MC

IL BITTEGONE

00123 ROMA VIA U. COMANDINI 49 TEL 06/6133025-2920559 TX.621166 FEPAG II

IMPORTATORI DIRETTI DALL' ESTREMO ORIENTE

di Felice Pagnani

SUSY 5 — E' IBM PC COMPATIBILE

SISTEMI

ECCEZIONALE

IL SISTEMA PER CHI INIZIA
128K 1 DRIVE 360K, USCITA MONOCROMATICA GRAFICA 720X390 USCITA STAMPANTE E STAMPANTE GRAFICA, VIDEO 25MHZ E BASCULAMENTO. DOS 2.0 TASTIERA, ALIM. 130W, 8 SLOTS DI ESPANSIONE 3.575.000
SUSY 5/HD Come sopra ma con Hard Disk 10 Mbytes, 6.000.000
SUSY5 BASE, 128K, grafica monocromatica, uscita parall. monitor un floppy 2.990.000

INTERFACCE PER SUSY 5

ESPANS. MEMORIA 64/512	408.000
GRAFICA COLORE	460.000
PORTA PARALLELA	102.000
MULTIFUNZIONE. 64/384	562.000
MULTIFUNZIONE 64/256	408.000
KIT RAM 64K	81.000
MONOCROMATICA P.PARALLELA GRAFICA 720X390	538.000
SCHEDA MONOCROMATICA	408.000

MEMORIE DI MASSA ROTANTI

KIT 10MB SLIM PC/XT	2.700.000
KIT 20MB SOLO PER XT	3.800.000
KIT 30MB SOLO PER XT	4.800.000
KIT 10M + 5M RIMOVIB.	5.700.000
completo di cavi	1.600.000
10 MBYTES IN BOX	3.150.000
20 MBYTES IN BOX	4.200.000
30 MBYTES IN BOX	5.200.000

COLLEGAMENTI LOCALI & REMOTI

DATANETS coppia di schede DATANET per rete locale con cavo e software	2.550.000
DATANET1 scheda DATANET per rete locale di un sistema aggiuntivo	1.300.000
PCOX scheda di emulazione di terminali 3278/3279 tramite control unit 3274 e cavo coassiale	2.600.000
PC5251 scheda per il collegamento via modem con sistemi IBM 34, 36 o 38	1.400.000
IRMA scheda per l'emulazione di terminali 3278 tramite control unit 3274 e cavo coassiale	$ 1.700
ACCOPPIATORE ACUSTICO	300.000
MODEM 300 BAUD	260.000

MONITORS & ACCESSORI

MONITOR MONOC.ALTA RIS. 25MHZ	
BASCULANTE 18,4KHZ/H'	330.000
MONITOR COLORE RGBI	889.000
MONITOR COLORE BARCO	2.100.000
CABINET E ACCESSORI	277.800
TASTIERA SLIM	254.000
PCMOUSE ottico completo di interfaccia RS232, alimentatore e tavoletta	640.000
PCPAINT come sopra con l'aggiunta del SW PCPAINT	740.000
STAMPANTI:	
80CHR/120 CPS	740.000
80CHR/180 CPS	990.000
136CHR/180 CPS	1.420.000
margherita	1.250.000
margherita	720.000
margherita	2.750.000

CADD PER IBM

Sistema grafico per IBM adatto per applicazioni in architettura, meccanica ed elettronica comprende:
MONITOR 0.31 dot pich colore VIDEO CONTROLLER 512 KB RAM 1024X1024X4 pixels, NEC 7220800 nsec velocita' di plotting due livelli di disegno, campo di indirizzamento 32.000 x 24.000 pixels. Il controller video e' direttamente connesso al BUS del PC IBM mediante un cavo ed una scheda di adattamento.
TABLET risoluzione: 0.125 mm, dim. 305x305 mm. RS232.
PLOTTER 40 cm/sec DIN A3/A4 Completo di SW 26.000.000

COMPUTER LINE

SUSY 2 — E' APPLE 2 COMPATIBILE

SISTEMI

48K pad numerico	774.000
64K pad num. e t.ffunz.	832.000
64K tastiera separata, 2Mbytes su 2 minifl. tastiera separata e contenitore tipo IBM, monitor ergonomico 25Mhz. DOS 3.3 Pascal 1.1 e CP/M 2.2. stampante grafica	3.300.000

****ECCEZIONALE*****
UN SISTEMA PER CHI INIZIA
SUSY 2 48K, UN FLOPPY SLIM UN MONITOR 12" 25Mhz 1.400.000

SCHEDE MADRI SUSY 2

48K zoccoli	381.000
64K " "	465.000

INTERFACCE PER SUSY 2

DISK DRIVE CARD	78.000
DISK DRIVE DOUBLE/FACE	121.800
PRINT INT. EPSON CARD	73.500
PARALLEL PRINTER CARD	72.200
UNIVERSAL PRINTER CARD	134.400
PRINT CABLE	41.600
LANGUAGE CARD	102.300
16K RAM CARD	102.300
INTEGER CARD	102.300
Z80 CP/M CARD	77.400
80X24 VIDEO CARD	129.500
80X24 VIDEO W/SWITCH	169.700
RS232 CARD	109.100
COMMUNICATION CARD	102.100
7710 ASYNCHRONOUS CARD	200.200
FORTH CARD	89.400
PARALL.CARD W/32KRAM	296.700
BUFFER CABLE 2/PCS	58.800
6522 CONTROL CARD	84.800
IEE488 CARD	225.800
SPEECH CARD W/SW	88.100
128K RAM SATURN W/SW	505.300
6809 CARD W/FLEX MANU.	358.800
MUSIC SYSTEM W/SW	128.900
WILD CARD	88.700
PAL CARD W/MODULATORE	110.000
AD/DA CARD W/SW	337.500
EPROM WRT (2716-32-64)	127.300
CLOCK CARD W/SW	124.400
OLIVETTI PRAXIS CARD	234.300
IBM CARD 8088 W/SW	625.200
RGB CARD W/CABLE	131.800
UNIVERSAL PARALL.PRINT	131.300
8748/49 MPU PROGRAMMER	429.800
JHON BELL A/D CARD	162.300
TELEFAX CARD	556.800
APPLI C. Z80,64K W/SW	649.900
IC TEST CARD W/SW	255.900
INTER.DRIVE 2 MBYTES	385.000

ACCESSORI PER SUSY 2

RF MODULATOR	17.800
JOYSTICK	23.500
DESK TOP JOYSTICK	33.900
JOYSTICK AUTOCENTERING	47.100
JOYSTICK AUTO QUIK FIRE	55.400
91 FUNCTION EXT.APPLE	133.200
FAN	29.500
COOLING FAN W/CABLE	75.100
LIGHT PEN HI RES.	393.400
TAVOLETTA GRAFICA	123.200
DRIVE 5" SINGLE HEAD	400.800
DRIVE 5" DOUBLE HEAD	748.400
DRIVE 5" 640KBYTES	500.600
DOUB.DRIVE MULTITECH	1.042.900
TAST. SUSY 2 MULTITECH	235.700
TASTIERA CON PAD NUM.	190.900
MONITOR COLORE RGB 14"	560.000
MONITOR B/N,VERDE,OCRA	215.000
BASCULAMENTO EUROVIDEO	35.000

PRO-DOS (COMPATIBILE)

Il PRO-DOS compatibile con tutti i compatibili. Riconosce l'ambiente in cui si trova e ci si adatta automaticamente:
Il disketto: 35.000
Un altro modo per rendere compatibile con il PRO-DOS il proprio compatibile: EPROM PRO DOS UNIVERSALE si inserisce al posto della ROM F8 e non e' piu' necessario modificare i dischetti. 2KEPROM 38.500
4KEPROM 42.500

Tutti i sistemi SUSY 2 sia da 48 sia da 64k RAM vengono gia' forniti compatibili con la EPROM di PRO-DOS.

CABINETS PER SISTEMI
Monitor ergonomico con spazio per scheda e alim. 100.000
Computer in due pezzi 75.000

PRINTER AD AGHI
STAMPANTI A MARGHERITA

SP100 100 CPS GRAFICA	642.000
SP120 120 CPS GRAFICA	730.000
MACCHINA DA SCRIVERE E STAMPANTE A MARGHERITA	700.000

ECCEZIONALE

XY PLOTTER
* rullo da 114 e 210 mm.
* Numerous intelligent funct.
* 4 Colour graphics.
* OHP Film Drawings.
* Usable as printer.
* Support graphics and special graphic symbols 760.000

MATERIALI DI CONSUMO

DISCHETTI 5" ACCUTRACK	37.000
DATA LIFE SF/DD BOX	40.000
RHONE POULENK SF/DD	40.000
RHONE POULENK DF/DD	60.000
CARTA 2000 FOGLI 80 C	40.000
CARTA 1000 FOGLI 80 C	20.000
CONTENITORE DISCHI	44.000
dischetti da 8" sf/sd e df/dd	

IVA

Tutti i prezzi sono IVA esclusa, pagamento in contanti, spedizioni in tutta Italia contrassegno. GARANZIA 3 MESI.

COMUNICAZIONE

Nonostante i massimi storici che il dollaro conquista noi abbiamo aumentato del minimo. I prezzi che pagate sono quelli indicati, quindi niente sorprese all'atto della fatturazione per il mese di MARZO.

LA PROPOSTA DEL MESE

CONTINUA CON ENORME SUCCESSO LA PROPOSTA DEL MESE CHE PARTITA A SETTEMBRE NON ACCENNA AD ESAURIRSI SI TRATTA DI Una scatola con 10 dischetti con tutti i migliori giochi del mondo piu' un joystick analogico autocentering. In tutto 10 megabytes di tutto divertimento 90.000
Solo la scatola 60.000

nostra esclusiva Progettazione e produzione

SUSY SUPER GRAPHIC

SUSY SUPER-GRAPHIC

Trasforma un SUSY][o compatibile, un APPLE][E in un potentissimo sistema grafico. I piu' alti livelli della grafica per impieghi professionali prima irraggiungibili per l'alto costo ora sono alla portata dei piu'.
Risoluzione 1Mega pixels (1024 x 1024 b/n o 512x512 4 piani colore). Generazione di disegni da hardware: vettori, cerchi, archi e rettangoli. possibilita' di PAN, SCROLL, ZOOM (fino a 16 volte). Uscita RGB. 128 KRAM a bordo, processore NEC7220 (16bit). Software fornito: interprete e PAINT consente l'uso di una tavoletta grafica digitale o del joystick. 1.750.000
Schedino PIG-BACK SSG per avere un uscita RGB lineare, videocomposito e una tavolozza di 4096 colori 275.000
Software opzionale:
PRIMITIVE consente l'uscita dal PAINT su basic con possibilita' di aggancio di set di caratteri e figure, generare delle funzioni. 175.000
RAM DISK consente di usare la scheda anche come disco virtuale sotto DOS 3.3 e sotto CP/M 100.000

SCHEDE IN STD-BUS Z80

ADATTE PER CONTROLLI INDUSTRIALI IN AMBIENTI AD ELEVATO STRESS - FUNZIONAMENTO 24 ORE SU 24 - ELEVATISSIMA AFFIDABILITA' - ADATTE ANCHE PER GESTIONALI CON IMPIEGO GRAVOSO.

CPU-I/O 64KRAM 2 seriali 1 parallela zocc.EPROM	750.000
CPU-I/O cs.senza RAM	390.000
FC2 Floppy Contr.DD	515.000
DR1 RAM 64K	470.000
DR2 RAM 256K	880.000
SPP1 4 p.seriali	407.000
BW1 8 zoccoli Byte W.	242.000
PPP1 4 p.parallele	319.000
AD1 adapter Winchest.	96.000
accessori:	
BOX 4 posti scheda	180.000
BOX 6 con terminazioni	220.000
BOX 8 con terminazioni	270.000
PAL/20 PAL PROGRAMMER	900.000
PPG/128 EPROM PROG	700.000
VDB 033 scheda video 80X24 RS232	320.000

SOFTWARE DI SUPPORTO ALLE SCHEDE: ADATTAMENTO CP/M, MP/M MULTIUSERS, BASIC RESIDENTE.

UN TERMINALE IN UFFICIO UNO A CASA E I DATI.....IN TASCA
Un nuovo concetto di portatilita':

SISTEMA 10 POCKET Z80 4MHZ, 2 porte seriali, 1 parallela, 64kRam, 10MBytes Winchester, 700KBytes minifloppy. CP/M. Piu' piccolo di un beauty-case a lire 4.500.000

GESTIONALE FP10M MODULARE ESPANDIBILE

10MBytes + 1 Floppy 5", 256K RAM, 2 PORTE SERIALI, 1 PARALLELA, 1 TERMINALE VIDEO, PREZZO 6.200.000
OGNI TERMINALE IN PIU 900.000

GESTIONALE FP10 SBC, 10MBYTES + 700 KBYTES minifloppy. 64k RAM. Interfacce 2 seriali, 1 parallela. 1 video 5.500.000