VIC da zero

di Tommaso Pantuso

Alla ricerca dei dati perduti

Rinfreschiamo le idee

Nello scorso numero della rivista abbiamo discusso su come recuperare i dati nella memoria del computer, quando questo si blocca per una qualsiasi ragione e non ci restituisce più il controllo della tastiera, per mezzo di un pulsante collegato alla macchina. Vogliamo in questo articolo chiudere l'argomento fornendovi una semplice routine che effettui automaticamente (o quasi) questo compito, dopo la pressione del pulsante citato. È bene però, prima di continuare, riassumere i risultati raggiunti la volta scorsa.

Quando noi inseriamo un programma in memoria, esso viene formattato dal sistema operativo secondo una struttura che schematizziamo nella figura 1. Per prima cosa all'accensione della macchina, grazie ad apposite routine, il sistema si accorge in quale configurazione di memoria si trova ed, in base a questa, stabilisce il punto da cui deve cominciare a memorizzare il programma (ricordiamo infatti che nel VIC 20 l'inizio del Basic varia in funzione delle espansioni possedute). Il punto d'inizio del programma viene indicato dal contenuto di due locazioni di memoria, per il VIC le locazioni decimali di pagina zero 43 e 44. Ad esempio in configurazione base tali registri contengono rispettivamente i numeri 1 e 16, quindi il programma inizia all'indirizzo:

$$1 + 16 \times 256 = 4097.$$

Esaminiamo più in dettaglio la figura 1 per meglio comprenderne la struttura. Essa descrive il modo in cui un programma in memoria viene diviso in vari blocchi e come questi vengono legati dalla indicazione fornita da un puntatore posto all'inizio di ciascun blocco. In altre parole, quando il sistema va ad eseguire un programma, controlla, dopo aver trovato un carattere di start (uno 0 nel nostro caso), il contenuto delle due locazioni di memoria che seguono tale carattere e grazie ad esso, al termine dell'esecuzione della prima linea, potrà identificare in memoria il punto da cui dovrà cominciare ad eseguire la seconda linea e così via. Se ad esempio il secondo blocco di programma inizia dalla locazione 4110, nel puntatore in testa al primo blocco troveremo i valori 14 e 16 che codificano appunto il salto a tale indirizzo (stiamo fornendo valori decimali) mediante la solita formula:

$$14 + 16 \times 256 = 4110.$$

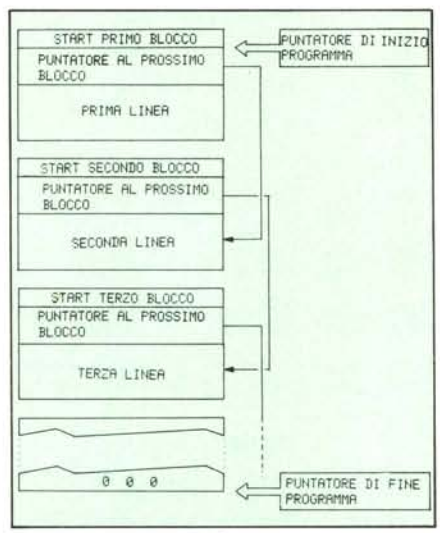

Figura 1 - Struttura della formattazione di un programma Basic in memoria. Uno '0' indica l'inizio di ciascun blocco di programma mentre la fine del programma è evidenziata dalla presenza di tre '0' consecutivi dopo l'ultimo blocco. Quando noi effettuiamo un reset, nel primo puntatore vengono memorizzati due zeri ed il puntatore di fine programma viene spostato di seguito ad essi: in tali condizioni il sistema "penserà" che in memoria non esiste alcun programma sia perchè non avrà possibilità di leggerlo mancando il primo link di "concatenazione" sia perché non "vedrà" per esso della memoria a disposizione. Sostituendo però nelle locazioni modificate i vecchi valori, preventivamente conservati in una apposita zona, il sistema sarà in grado di rioperare correttamente.

Un altro puntatore di cui ci interessa conoscere la funzione è quello contenuto nelle locazioni 45 e 46 che indica il punto in cui finisce il programma già in memoria e quindi il punto da cui il sistema può cominciare ad inserire una eventuale nuova linea. Il valore di tale puntatore varia naturalmente man mano che un programma viene caricato in memoria e la fine di un programma è indicata da tre "0" consecutivi.

Quando noi effettuiamo un reset hardware collegando a massa tramite l'apposito pulsante la linea 3 della user port oppure la linea X posta sulla porta di espansione della memoria, il sistema provvede a ristabilire le condizioni iniziali del puntatore che indica la fine del programma ed a memorizzare uno 0 nella seconda e terza locazione in testa al primo blocco le quali, come sappiamo, contengono l'indicazione che permette al sistema di collegare tale blocco ai successivi. Inoltre essendo resettato il puntatore di fine programma al valore posseduto all'accensione, la macchina interpreterà questo fatto come un'indicazione dell'assenza di programmi in memoria.

Ripeschiamo il programma

Se fossimo a conoscenza del contenuto delle locazioni che vengono modificate con il reset sarebbe facile recuperare il programma *nascosto* in memoria andando a sostituire i vecchi valori nelle locazioni che indicano la fine del programma e nel *link* che concatena il primo blocco ai successivi. Purtroppo se il sistema si blocca per una qualunque causa non ci dà il preavviso e non potendo in tal caso utilizzare la tastiera non avremo la possibilità di leggere tali contenuti.

A questo punto la struttura stessa del VIC 20 ci offre lo spunto per la soluzione dei nostri problemi.

Come ormai tutti saprete (se non altro perchè ne abbiamo parlato tanto in questa rubrica) sessanta volte al secondo il sistema attiva un insieme di routine dette di *manipolazione dell'interrupt* che permettono alla macchina di svolgere correttamente le proprie funzioni. Noi possiamo aggiungere alle routine di manipolazione una routine di nostra creazione la quale verrà così eseguita sessanta volte al secondo. Vi ricordiamo che una volta eseguita la manipolazione, della durata di pochi microsecondi, il sistema continua ad eseguire un eventuale programma lasciato in sospeso

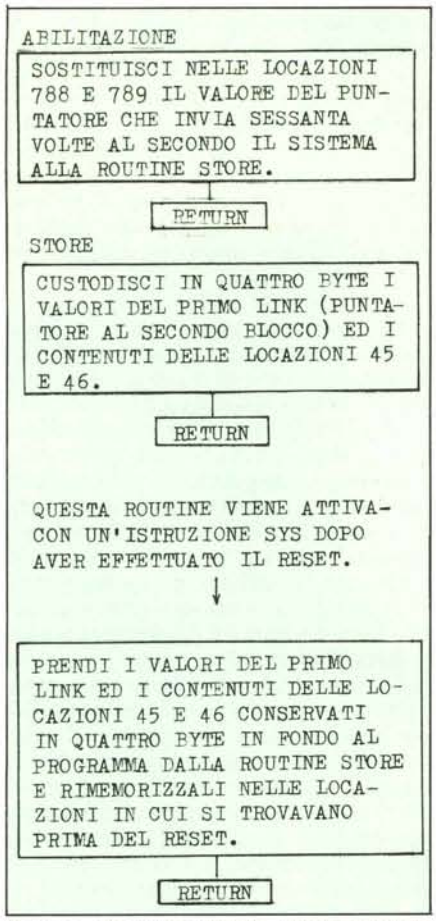

Figura 2 - Descrizione a blocchi delle funzioni svolte dalla routine OLD. La prima cosa da fare è mettere in grado il sistema di andare ad eseguire STORE seguendo il corso dell'interrupt. I valori conservati possono essere prelevati dopo il reset con un invio alla routine EXC.

al momento del salto verso le routine preposte all'elaborazione dell'interrupt. L'indirizzo d'inizio di tali routine è puntato dal contenuto di due locazioni della RAM, le 788 e 789, e può essere modificato affinchè indirizzi alla routine di nostra concezione la quale terminerà con un salto alle routine di manipolazione.

In altre parole, prima verrà eseguita la nostra routine, poi la normale manipolazione dell'interrupt ed infine il sistema riprenderà le sue normali funzioni. Questo sessanta volte al secondo circa.

La routine che vi proponiamo implementa l'algoritmo schematizzato in figura 2. In pratica, man mano che un programma viene introdotto in memoria sono conservati in una zona protetta dalla RAM i contenuti delle locazioni che vengono aggiornate ad ogni modifica che si effettua durante la composizione del programma, o quando questo viene caricato da un supporto magnetico. Dato che si usa la tecnica dell'interrupt appena menzionata, la zona contenente i valori che abbiamo scelto di proteggere verrà aggiornata sessanta volte al secondo. La routine, che battezzeremo OLD, è suddivisa in tre blocchi.

Il primo è preposto all'abilitazione dell'esecuzione della ROUTINE STORE (secondo blocco) sessanta volte al secondo. Tale segmento (STORE) provvede a prelevare il contenuto del puntatore alla fine del programma e del *Link* al secondo blocco ed a memorizzarlo nelle locazioni protette della RAM (fig. 3).

Supponendo di aver a che fare con il VIC in configurazione base, cioè senza alcuna espansione, proteggeremo dalla sovrapposizione di altri programmi 100 byte in fondo alla memoria. Tale zona andrà quindi da 7580 a 7680 ed in essa scriveremo il nostro programma in linguaggio macchina. Useremo le locazioni che vanno da 7656 a 7659 per conservarvi i puntatori da proteggere; esse verranno, ribadiamo, aggiornate ogni sessantesimo di secondo.

L'ultimo blocco, ROUTINE EXC, nel VIC base inizia dalla locazione 7633 e provvede, dopo il reset e se abilitato con una SYS 7633, ad effettuare il passaggio inverso cioè a rimemorizzare nelle locazioni modificate dalla routine di reset del sistema i vecchi contenuti riportandoli nelle condizioni in cui si trovavano prima del blocco. In figura 3 riportiamo il listato del programma OLD in LM mentre in figura 4 è rappresentato il dump esadecimale dello stesso.

```
                ;ABILITAZIONE INTERRUPT
                ;
SEI             ;PONI IL FLAG I
LDA  #$B8       ;CARICA A CON IL BYTE BASSO
                ;DELL'INDIRIZZO RUOTINE STORE
STA  $0314      ;MEMORIZZALO NEL REGISTRO $0314
LDA  #$1D       ;CARICA A CON IL BYTE ALTO
                ;DELL'INDIRIZZO RUOTINE STORE
STA  $0315      ;MEMORIZZALO NEL REGISTRO $0315
LDA  #$9C       ;CARICA A CON IL NUMERO $9C
STA  $33        ;MEMORIZZALO NEL REGISTRO $33
STA  $37        ;MEMORIZZALO NEL REGISTRO $37
LDA  #$1D       ;CARICA A CON IL NUMERO $1D
STA  $34        ;MEMORIZZALO NEL REGISTRO $34
STA  $38        ;MEMORIZZALO NEL REGISTRO $38
NOP             ;NESSUNA OPERAZIONE
NOP             ;NESSUNA OPERAZIONE
NOP             ;NESSUNA OPERAZIONE
CLI             ;DISATTIVA IL FLAG I
RTS             ;RITORNA AL PRG.PRINCIPALE
                ;
                ;RUOTINE STORE
                ;
LDA  ($1001)    ;CARICA A CON IL CONTENUTO
                ;DEL REGISTRO $1001
STA  $1DE8      ;MEMORIZZALO NEL REGISTRO $1DE8
LDA  ($1002)    ;CARICA A CON IL CONTENUTO
                ;DEL REGISTRO $1002
STA  $1DE9      ;MEMORIZZALO NEL REGISTRO $1DE9
LDA  ($2D)      ;CARICA A CON IL CONTENUTO
                ;DEL REGISTRO $2D
STA  $1DEA      ;MEMORIZZALO NEL REGISTRO $1DEA
LDA  ($2E)      ;CARICA A CON IL CONTENUTO
                ;DEL REGISTRO $2E
STA  $1DEB      ;MEMORIZZALO NEL REGISTRO $1DEB
JMP  $EABF      ;SALTA ALLA NORMALE ROUTINE DI
                ;INTERRUPT
                ;
                ;ROUTINE EXC
                ;
LDA  ($1DE8)    ;CARICA A CON IL CONTENUTO
                ;DEL REGISTRO $1DE8
STA  $1001      ;MEMORIZZALO NEL REGISTRO $1001
LDA  ($1DE9)    ;CARICA A CON IL CONTENUTO
                ;DEL REGISTRO $1DE9
STA  $1002      ;MEMORIZZALO NEL REGISTRO $1002
LDA  ($1DEA)    ;CARICA A CON IL CONTENUTO
                ;DEL REGISTRO $1DEA
STA  $2D        ;MEMORIZZALO NEL REGISTRO $2D
LDA  ($1DEB)    ;CARICA A CON IL CONTENUTO
                ;DEL REGISTRO $1DEB
STA  $2E        ;MEMORIZZALO NEL REGISTRO $2E
RTS             ;RETURN
```

Figura 3 - Listato assembler della routine proposta per il VIC in configurazione base. Nello stesso articolo forniamo i programmi in Basic che caricano in memoria tale routine in tutte le configurazioni di memoria.

```
Dump della routine OLD - VIC base

.
1D9C 78 A9 B8 8D 14 03 A9 1D 8D 15
1DA6 03 A9 9C 85 33 85 37 A9 1D 85
1DB0 34 85 38 EA EA EA 58 60 AD 01
1DBA 10 8D E8 1D AD 02 10 8D E9 1D
1DCA A5 2D 8D EA 1D A5 2E 8D EB 1D
1DCE 4C BF EA AD E8 10 8D 01 10 AD
1DD8 E9 1D 8D 02 10 AD EA 1D 85 2D
1DE2 AD EB 1D 85 2E 60 00 00 00 00
```

Figura 4 - Dump esadecimale della routine OLD.

Facciamo un esempio

A questo punto non resta altro da fare che verificare in pratica quanto appreso finora. Le operazioni che effettueremo saranno riferite al VIC in configurazione base riservandoci di estendere più avanti nell'articolo i risultati alla macchina posta in qualunque configurazione di memoria.

La prima cosa da fare è procurarsi un pulsante del tipo *normalmente aperto* (cioè quelli che chiudono il circuito se premuti) e collegarne un capo al terminale numero 3 ed un altro a quello numero uno di un connettore 12+12 da inserire nella user port oppure tra i terminali X e Z di un connettore 22+22 per la porta di espansione della memoria. Fatto ciò possiamo controllarne la funzionalità semplicemente premendolo: se il collegamento è stato eseguito a dovere sullo schermo dovrà comparire la scritta impressa ogni volta che viene accesa la macchina e cioè

CBM BASIC V2
3583 BYTE FREE.

A questo punto non resta altro da fare che inserire in memoria la routine OLD. Questa operazione può essere effettuata o utilizzando un assemblatore oppure (cosa che tutto sommato riteniamo più pratica) per mezzo delle poche linee riportate nel listato 1. Le linee 50 e 60 di quest'ultimo servono, come spiegato dettagliatamente la volta scorsa, ad *abbassare* i puntatori di fine memoria e di inizio stringhe, selezionando un'area che risulterà protetta dalla eventuale sovrapposizione di un programma in Basic.

La routine completa viene memorizzata a partire dalla locazione decimale 7580, inizio della zona protetta e per una lunghezza di 80 byte dalle istruzioni della linea 80. Alla fine della memorizzazione tramite una SYS 7580 essa viene resa operativa e da questo momento in poi potremo inserire qualunque programma, da nastro o disco oppure da tastiera. Provate ora, cioè dopo aver inserito in macchina un programma, ad effettuare una qualunque operazione che blocchi il sistema, ad esempio

```
10 REM ------------------------------------
20 REM ------- OLD 8K - (SYS 7633) -------
30 REM ---------- (C) TP  1984 ----------
40 REM ------------------------------------
50 POKE51,156:POKE52,29
60 POKE55,156:POKE56,29
70 FORI=7580TO7659
80 READA:POKEI,A:NEXT
90 DATA 120,169,184,141,20,3,169,29,141,21
100 DATA 3,169,156,133,51,133,55,169,29,133
110 DATA 52,133,56,234,234,234,88,96,173,1
120 DATA 16,141,232,29,173,2,16,141,233,29
130 DATA 165,45,141,234,29,165,46,141,235
140 DATA 29,76,191,234,173,232,29,141,1,16
150 DATA 173,233,29,141,2,16,173,234,29,133
160 DATA 45,173,235,29,133,46,96,0,0,0,0
170 CLR:SYS7580:NEW
```

Listato 1 Caricatore BASIC della routine OLD in un VIC in configurazione base.

SYS 1: il cursore scomparirà e non sarete più in grado di utilizzare la tastiera.

Non ci resta allora altro da fare che premere il pulsante di reset e, quando ritorniamo in possesso del cursore, digitare SYS 7633. Quest'ultima operazione permetterà il recupero totale del programma introdotto per la prova.

A questo punto precisiamo che l'operazione di reset ha effetto anche sul puntatore che dirottava il sistema verso la nostra routine, quindi la manipolazione dell'interrupt segue il suo corso originario e non vengono più conservati i contenuti delle locazioni che ci interessano nella zona protetta. Per ripristinare questa funzione dovremo, dopo ogni operazione di reset, effettuare una SYS 7580 (quella della linea

	LOCATION DEC	LOCATION HEX	VIC	VIC + 3K	VIC + 8K	VIC + 16K
1K	0 1023	0000 03FF	MICROSOFT BASIC RAM	MICROSOFT BASIC RAM	MICROSOFT BASIC RAM	MICROSOFT BASIC RAM
3K	1024 4095	0400 01FF	NON-EXISTENT (3K EXPANSION RAM)	USER BASIC PROGRAM AREA	NON-EXISTENT (3K EXPANSION)	NON-EXISTENT (3K EXPANSION)
3.5K	4096 7679	1000 1DFF	USER BASIC PROGRAM AREA		SCREEN RAM 4607 4608	SCREEN RAM 4607 4608
.5K	7680 8191	1E00 1FFF	SCREEN RAM (VIDEO MATRIX)	SCREEN RAM (VIDEO MATRIX)	USER BASIC PROGRAM AREA	USER BASIC PROGRAM AREA
8K	8192 16383	2000 3FFF	NON-EXISTENT (8K EXPANSION ROM/RAM)	NON-EXISTENT		
8K	16384 24575	4000 5FFF	NON-EXISTENT (8K EXPANSION ROM/RAM)	NON-EXISTENT	NON-EXISTENT	
8K	24576 32767	6000 7FFF	NON-EXISTENT (8K EXPANSION ROM/RAM)	NON-EXISTENT	NON-EXISTENT	NON-EXISTENT
4K	32768 36863	8000 8FFF	CHAR ROM (CHARACTER MATRIX)	CHAR ROM (CHARACTER MATRIX)	CHAR ROM	CHAR ROM
	36864 36879	9000 900F	VIC(6561)CHIP	VIC(6561)CHIP	VIC(6561)CHIP	VIC(6561)CHIP
	36880 37135	9010 910F	(VIC CHIP?)	?	?	?
	37136 37151	9110 911F	VIA(6522)CHIPS I/O	VIA(6522)CHIPS I/O	VIA(6522)CHIPS I/O	VIA(6522)CHIPS I/O
	37152 37167	9120 912F	I/O	I/O	I/O	I/O
1K	37888 38399	9400 95FF	Free Nybbles	Free Nybbles	COLOR RAM	COLOR RAM
	38400 38911	9600 97FF	COLOR RAM	COLOR RAM	Free Nybbles	Free Nybbles
2K	38912 40595	9800 9FFF	EXPANSION? I/O NON-EXISTENT	EXPANSION? I/O NON-EXISTENT	EXPANSION? I/O NON-EXISTENT	EXPANSION? I/O NON-EXISTENT
8K	40960 49151	AC00 BFFF	EXPANSION ROM NON-EXISTENT	EXPANSION ROM NON-EXISTENT	EXPANSION ROM NON-EXISTENT	EXPANSION ROM NON-EXISTENT
8K	49152 57343	C000 DFFF	BASIC ROM	BASIC ROM	BASIC ROM	BASIC ROM
8K	57344 65535	E000 FFFF	KERNAL ROM	KERNAL ROM	KERNAL ROM	KERNAL ROM

Figura 5 Mappa completa della memoria del VIC in qualunque configurazione fino a 16 K di espansione (figura tratta da Mastering the VIC 20).

Listato 2

```
10 REM ------------------------------------
20 REM ------- OLD 3K - (SYS 7633) -------
30 REM ---------- (C) TP  1984 -----------
40 REM ------------------------------------
50 POKE51,156:POKE52,29
60 POKE55,156:POKE56,29
70 FORI=7580TO7659
80 READA:POKEI,A:NEXT
90 DATA 120,169,184,141,20,3,169,29,141,21
100 DATA 3,169,156,133,51,133,55,169,29,133
110 DATA 52,133,56,234,234,234,88,96,173,1
120 DATA 4,141,232,29,173,2,4,141,233,29
130 DATA 165,45,141,234,29,165,46,141,235
140 DATA 29,76,191,234,173,232,29,141,1,4
150 DATA 173,233,29,141,2,4,173,234,29,133
160 DATA 45,173,235,29,133,46,96,0,0,0,0
170 CLR:SYS7580:NEW
```

Listato 3

```
10 REM ------------------------------------
20 REM ------- OLD 8K - (SYS 16337) ------
30 REM ---------- (C) TP  1984 -----------
40 REM ------------------------------------
50 POKE51,156:POKE52,63
60 POKE55,156:POKE56,63
70 FORI=16284TO16363
80 READA:POKEI,A:NEXT
90 DATA 120,169,184,141,20,3,169,63,141,21
100 DATA 3,169,156,133,51,133,55,169,63,133
110 DATA 52,133,56,234,234,234,88,96,173,1
120 DATA 18,141,232,63,173,2,18,141,233,63
130 DATA 165,45,141,234,63,165,46,141,235
140 DATA 63,76,191,234,173,232,63,141,1,18
150 DATA 173,233,63,141,2,18,173,234,63,133
160 DATA 45,173,235,63,133,46,96,0,0,0,0
170 CLR:SYS16284:NEW
```

Listato 4

```
10 REM ------------------------------------
20 REM ------ OLD 16K - (SYS 24529) ------
30 REM ---------- (C) TP  1984 -----------
40 REM ------------------------------------
50 POKE51,156:POKE52,95
60 POKE55,156:POKE56,95
70 FORI=24476TO24555
80 READA:POKEI,A:NEXT
90 DATA 120,169,184,141,20,3,169,95,141,21
100 DATA 3,169,156,133,51,133,55,169,95,133
110 DATA 52,133,56,234,234,234,88,96,173,1
120 DATA 18,141,232,95,173,2,18,141,233,95
130 DATA 165,45,141,234,95,165,46,141,235
140 DATA 95,76,191,234,173,232,95,141,1,18
150 DATA 173,233,95,141,2,18,173,234,95,133
160 DATA 45,173,235,95,133,46,96,0,0,0,0
170 CLR:SYS24476:NEW
```

Listato 5

```
1 REM ------------------------------------
2 REM ----- OLD 24K - (SYS 32721) -------
3 REM ---------- (C) TP  1984 -----------
4 REM ------------------------------------
10 POKE52,127:POKE56,127
15 POKE51,156:POKE55,156
20 FORI=32668TO32747
30 READA:POKEI,A:NEXT
50 DATA 120,169,184,141,20,3,169,127,141,21
60 DATA 3,169,156,133,51,133,55,169,127,133
70 DATA 52,133,56,234,234,234,88,96,173,1
80 DATA 18,141,232,127,173,2,18,141,233,127
90 DATA 165,45,141,234,127,165,46,141,235
100 DATA 127,76,191,234,173,232,127,141,1,18
110 DATA 173,233,127,141,2,18,173,234,127,133
120 DATA 45,173,235,127,133,46,96,0,0,0,0
130 CLR:SYS32668:NEW
```

Caricatori della routine OLD per Vic in qualsiasi configurazione di memoria.

170 del listato 1) affinchè la routine di OLD ridiventi operativa.

Un'osservazione. Il fatto che la routine che manipola l'interrupt a nostro favore venga disabilitata premendo il pulsante di reset è l'evento chiave che ci permette di recuperare il programma. Grazie a ciò infatti le locazioni della zona protetta che contengono i valori dei puntatori modificati non vengono più aggiornate ed in esse rimangono memorizzati i dati presenti prima del reset che quindi possono venir recuperati per mezzo della SYS 7633. Se lo svolgersi della routine in questione non venisse arrestato verrebbero conservati come ultimi valori quelli dei puntatori dopo il reset che coincidono con quelli del sistema all'accensione. È questa la ragione per cui questa routine non funziona se effettuiamo un NEW.

Una memoria multiforme

La memoria del VIC 20 non ha una configurazione fissa nel senso che a seconda del tipo di espansione che si inserisce variano i punti d'inizio di alcune zone della RAM preposte a compiti specifici. Si osservi a tale scopo la 'preziosa' figura 5 (pag. 127) tratta da 'Mastering the VIC 20' la quale rappresenta la suddivisione della memoria in base alle diverse quantità di RAM inserite.

Ad esempio nel VIC in versione base il programma in Basic viene memorizzato a partire dall'indirizzo 4096 mentre se inseriamo una cartridge da 3K l'inizio del Basic si sposta a 1024. In queste due configurazioni la zona 'RAM di schermo' viene posta dal sistema subito di seguito alla zona riservata al Basic mentre l'area destinata alla RAM del colore inizia dalla locazione 38400. Se invece inseriamo espansioni di 8, 16 o 24 K, il punto di partenza della RAM di schermo si sposterà a 4096 e dopo 512 byte avrà inizio il programma. In tali configurazioni la zona destinata alla RAM del colore si abbassa di 512 byte. Sono queste sostanzialmente le modifiche più importanti che avvengono nel sistema quando aggiungiamo della memoria.

Esse possono essere individuate, se non si ha a disposizione una mappa come quella riportata in figura 5, controllando appositi puntatori o locazioni specifiche (ad esempio moltiplicando il contenuto della locazione 648 per 256 si ottiene l'indirizzo d'inizio della RAM di schermo, ecc.).
I listati che vanno dal 2 al 5 permettono di caricare in macchina la routine OLD e in un VIC comunque espanso. Nelle linee di commento iniziali è appunto indicato il tipo di espansione richiesta e la SYS necessaria per il recupero del programma dopo la pressione del pulsante di reset. Anche negli altri casi, cioè di VIC espanso, bisognerà effettuare, dopo la SYS contenuta nelle linee di commento, quella indicata nella linea 170 dei programmi per rendere di nuovo operativa la manipolazione dell'interrupt e quindi la routine OLD. Al prossimo numero! MC

Elenco del software disponibile su cassetta o minifloppy

Per ovviare alle difficoltà incontrate da molti lettori nella digitazione dei listati pubblicati nelle varie rubriche di software sulla rivista, da alcuni mesi MCmicrocomputer mette a disposizione i programmi più significativi direttamente su supporto magnetico. Riepiloghiamo qui a fianco i programmi disponibili per le varie macchine, ricordando che i titoli non sono previsti per computer diversi da quelli indicati. Il numero della rivista su cui viene descritto ciascun programma è riportato nell'apposita colonna; consigliamo gli interessati di procurarsi i relativi numeri arretrati, eventualmente rivolgendosi al nostro Servizio Arretrati utilizzando il tagliando pubblicato in fondo alla rivista.

Codice	Titolo programma	MC n.	Prezzo	Note
	APPLE II			
DA2/00	Shape Tablet	22	15000	
DA2/01	Motomuro	26	15000	
DA2/02	&DEBUG	28	15000	
DA2/03	EDIT + INPUT	29	15000	
	COMMODORE 64			
C64/01	Briscola	25	17000	
C64/02	Serpentone	29	17000	
C64/03	Othello	29	17000	
C64/04	Chase	33	17000	
	COMMODORE VIC-20			
CVC/01	VIC-Maze	19	17000	Config. base
CVC/02	Pic-Man	23	17000	Config. base
CVC/03	Briscola	25	17000	Config. base
CVC/04	Grand Prix	28	17000	Config. base
CVC/05	Frogger	26	17000	RAM: almeno + 3 K
CVC/06	Invaders	29	23000	RAM: + 16 K
CVC/07	Othello	29	17000	RAM: + 16 K
CVC/08	SKI	31	17000	Config. base
CVC/09	VIC-quiz	32	17000	RAM: almeno + 8 K
CVC/10	Zigurat	33	17000	Config. base
DVC/01	EXMA	27/28	15000	RAM: + 16 K
	SINCLAIR SPECTRUM			
CSS/01	TRILAB	28	17000	
CSS/02	SET di caratteri	27/29	17000	
CSS/03	Grafica TREDIM	29	17000	
CSS/04	Ippica	30	17000	
CSS/05	Graphic-Comp	32	17000	
	TEXAS TI-99/4A			
CT9/01	Macchina del tempo	27	17000	
CT9/02	Simon	29	17000	
CT9/03	Babilonia	30	17000	
CT9/04	Labirinto 3D	31	17000	
CT9/05	Piramide di Iunnuh	33	17000	Extended Basic

Nota:
l'iniziale del codice e' C per le cassette, D per i minifloppy

Per l'ordinazione inviare l'importo (a mezzo assegno, c/c o vaglia postale) alla Technimedia srl, Via Valsolda 135, 00141 Roma.

Le cassette utilizzate sono Basf C-60 Compusette II; i minifloppy sono Basf singola faccia singola densità.