Parte con questo primo articolo una rubrica di software destinata al Commodore 64. L'intenzione è, come sempre, quella di essere di aiuto al lettore proponendo principalmente utility e tool, cercando di spiegare cosa effettivamente accade nel computer, in modo che i trucchi al di fuori del BASIC vengano messi a disposizione di tutti coloro che non hanno il tempo di trovarseli da soli (il problema è solo quello, credeteci!).

D'altronde per il caso Commodore — VIC contro 64 — sta accadendo quanto già visto con l'analogo caso Sinclair — 81 contro Spectrum —: il grosso pubblico, che aveva disassemblato i primi usciti (ZX 81 e VIC) si sta gettando sui nuovi prodotti (Spectrum e 64), anche favorito dall'incredibile calo dei prezzi. Ovvio quindi l'interesse di MCmicrocomputer per la nuova ondata di software.

Questo mese vi presentiamo una routine grafica in alta risoluzione, 320 × 200, la massima consentita dal modo grafico a mappa punto per punto, che nel 64 si abilita in modo assai semplice. L'articolo, scritto dall'autore del programma, è stato ampliato con alcune spiegazioni.

La base comune al programma e all'articolo è la Programmer's Reference Guide della Commodore, in vendita in tutti i computer shop della catena Bit-Shop Primavera e in molti altri negozi specializzati.

Alta risoluzione sul CBM 64

di Andrea Damiani - Roma

Il CBM 64 ha una gestione del tipo di uscita video molto completa e versatile. La supervisione di ogni funzione è affidata al circuito integrato 6567, detto familiarmente VIC-II. I modi grafici sono di due tipi, a caratteri o in alta risoluzione. Il tipo a caratteri può a sua volta usare il set standard ovvero esser riprogrammato dall'utente (user-defined graphics), e può esser gestito in tre diversi sistemi (normale, multicolor o a controllo individuale del colore di sfondo). La locazione 53272 contiene due informazioni: nei quattro bit alti (detti nybble alto), quelli di numero da 4 a 7, la locazione di partenza della memoria di schermo (che essendo di 25 righe per 40 colonne si estende per 1000 locazioni), mentre i bit 2, 3 e 4 tengono la posizione della memoria di colore. Ricordando che il 64 ammette 16 colori comprendiamo come mai il codice del colore di ogni singola locazione (sfondo + carattere) sia contenuto in soli 8 bit e perché la memoria di colore occupi 1000 byte. Il bit 0 del registro 53272 non viene considerato.

Le leggi che assegnano le locazioni di memoria da cui parte la zona di schermo o quella di colore sono allora molto semplici: ad es. nel primo caso basta moltiplicare per 1024 il controvalore decimale del numero binario contenuto nel nybble alto (bit 4-7) del registro 53272: le varie posizioni vengono selezionate tramite il comando POKE 53272,(PEEK(53272)AND 15) ORA, che realizza i succitati spostamenti secondo la

```
10 REM **********************************
20 REM **** ALTA RISOLUZIONE CON IL ***
30 REM ****     COMMODORE  64        ***
40 REM **** DI    ANDREA  DAMIANI    ***
50 REM **********************************
60 :
80 PRINT"■":POKE53280,3:POKE53281,1
90 X1=-10:X2=10:DY=4:FS=15.95
100 FORK=49152TO49222:READP:POKEK,P:NEXT
110 PRINT"■■■■■■■■■"
120 PRINT"■■■■■■■■■A■STAMPA GRAFICO"
140 PRINT"■■■■■■■■■B■CAMBIARE FUNZIONI"
150 PRINT"■■■■■■■■■C■INTRODURRE INTERVALLI"
160 GETJ$
170 IFJ$="A"THEN1000
190 IFJ$="B"THENLIST2000-3999
200 IFJ$="C"THEN500
250 GOTO160
500 INPUT"■■■■■■■■■■■■X1,X2,DY";X1,X2,DY
510 FS=319/(X2-X1)
1000 BA=8192:SYS49152
1100 T=DY*FS:IFT>199ORT<0THEN1200
1110 FORZ=0TO319STEP6
1120 T=DY*FS
1130 GOSUB10000
1140 NEXT
1200 IFX1>0ORX2<0THEN2000
1210 Z=FS*(-X1)
1220 FORY=0TO199STEP4
1230 T=Y:GOSUB10000
1240 NEXT
2000 FORZ=0TO319STEP4:X=Z/FS+X1
2020 Y=8*SIN(X)/X:GOSUB9500
2400 NEXT
2500 FORR=0TO199STEP4:Y=R/FS-DY
2510 X=-Y:GOSUB9000
3000 NEXT
4000 POKE198,0:WAIT198,1
4010 POKE53265,PEEK(53265)AND223
4020 POKE53272,PEEK(53272)AND247
4100 GOTO110
9000 Z=FS*(X-X1):T=R:IFZ<0ORZ>319THENRETURN
9010 GOTO10000
9500 T=Y*FS+FS*DY
10000 T=INT(199-T):IFT>199ORT<0THENRETURN
10010 CH=INT(Z/8)
10020 RO=INT(T/8)
10030 LN=T AND 7
10040 BY=BASE+RO*320+8*CH+LN
10050 BI=7-(ZAND7)
10060 POKEBY,PEEK(BY)OR(2↑BI)
10070 RETURN
50000 :
50100 DATA173,24,208,9,8,141,24,208
50200 DATA173,17,208,9,32,141,17,208
50300 DATA169,0,133,251,169,32,133,252
50400 DATA160,0,169,0,145,251,200,192
50500 DATA0,208,249,230,252,169,64,197
50600 DATA252,208,239,169,0,133,251,169
50700 DATA4,133,252,160,0,169,3,145
50800 DATA251,200,192,0,208,249,230,252
50900 DATA169,8,197,252,208,239,96
55000 :
55555 POKE53272,PEEK(53272)AND247
55556 POKE53265,PEEK(53265)AND223:END
```

Tabella 1 A	BIT	LOCAZIONE	
		DECIMALE	HEX
0	0000XXXX	0	$0000
16	0001XXXX	1024	$0400 (DEFAULT)
32	0010XXXX	2048	$0800
48	0011XXXX	3072	$0C00
64	0100XXXX	4096	$1000
80	0101XXXX	5120	$1400
96	0110XXXX	6144	$1800
112	0111XXXX	7168	$1C00
128	1000XXXX	8192	$2000
144	1001XXXX	9216	$2400
160	1010XXXX	10240	$2800
176	1011XXXX	11264	$2C00
192	1100XXXX	12288	$3000
208	1101XXXX	13312	$3400
224	1110XXXX	14336	$3800
240	1111XXXX	15360	$3C00

regola trovata, in dipendenza dei valori di A (chiaramente multipli di 16) come mostrato nella tabella 1.

Il programma

Il VIC-II lavora con un gruppo di 47 registri contenuti in altrettante locazioni di memoria, dalla 53248 alla 53294: fra queste ci interessa particolarmente la 53272, dalle cui proprietà notiamo che ponendoci un numero decimale maggiore di 15 (ovvero andando ad interessare il nybble alto) andremo direttamente in alta risoluzione, mentre per tornare in modo testo basterà porre nel registro un valore da 15 compreso in giù. Il modo a mappa di bit abilita una corrispondenza uno a uno tra i bit della RAM e i punti dello schermo: dato che la risoluzione a nostra disposizione è di 320*200 punti, ci serviranno 64000 bit =8000 byte. Scegliamo di porre questa zona di memoria a partire dalla locazione 8192, e informiamo il VIC-II tramite l'istruzione POKE 53272,PEEK(53272) OR8. Questa zona di memoria è usualmente occupata dai programmi in BASIC, ma possiamo permetterci di usarla poiché il listato del nostro programma non occupa tanto spazio. Per avere la stessa risoluzione in termini di colori non potevamo occupare altri 8K, onde per cui lo schermo è stato suddiviso in una matrice da 40 colonne per 25 righe, con la possibilità di scegliere tra due colori per ogni isola da 8* punti. Per ogni byte usato il nybble alto viene impiegato per colorare i punti non settati, mentre il nybble basso contiene il codice del colore dei punti del grafico propriamente detto. Questa memoria colore viene messa a partire dalla locazione 1024, ovvero la posizione in cui si trovava la memoria di schermo all'atto dell'accensione. Per far degli esempi, riferendoci alla tabella dei codici dei colori, volendo nella locazione di partenza (la 1024) un colore di sfondo verde e un colore di grafico bianco, la regola è sfondo*16 + bordo, ovvero (cod. verde)*16 + (cod. bianco) che vale 5*16 + 1=81; è questo il valore decimale da porre nella locazione che ci interessa, la 1024, tramite una POKE. I codici dei colori sono mostrati a pag. 159 del manuale CBM.

Dal Run in poi

Una volta digitato il programma, il RUN mostrerà il mini-menu, comprendente tre opzioni: stampa del grafico, cambiamento delle funzioni e introduzione di nuovi intervalli. La selezione della scelta fatta avviene tramite i tasti A,B e C nell'ordine, mentre la barra spaziatrice permette l'uscita dalla pagina grafica e il ritorno al menu. L'inserimento delle funzioni da studiare è assai semplice: viene presentato il listato delle funzioni precedenti, e sarà sufficiente modificare questo ricordando che le funzioni del tipo Y=F(X) vanno inserite tra le righe 2000 e 2400 escluse, facendo seguire ad ogni funzione un GOSUB 9500, mentre per le funzioni del tipo X=F(Y) l'inserimento dovrà avvenire tra le linee 2500 e 3000 escluse, ed attivate dall'istruzione conclusiva GOSUB 9000. Terminate le modifiche il RUN mostrerà il menu ma le funzioni saranno quelle desiderate.

L'opzione C permette di scegliere gli intervalli all'interno dei quali deve essere tracciato il grafico. Il programma chiede i valori per X1,X2 e DY, con i seguenti significati: X1 è l'ascissa più a sinistra; X2 l'estremo valore destro; DY è l'ordinata dell'asse delle ascisse rispetto alla scala assunta, tramite X1 e X2, per le ascisse stesse.

Commento al listato

Le POKE in linea 8Ø e 9Ø cambiano i colori della pagina video, mentre la linea 1ØØ legge la routine di pulizia schermo in linguaggio macchina contenuta nei DATA da 5Ø1ØØ a 5Ø9ØØ.

Le linee 11Ø-15Ø presentano il menu, mentre la routine 16Ø-25Ø verifica il tasto premuto; l'INPUT in linea 5ØØ accetta i valori di dimensionamento dello schermo in funzione del grafico da farsi.

La sub in L.M. chiamata dalla linea 1ØØØ con la SYS 49152 azzera la pagina grafica e pone dei 3 in quella di colore (corrispondente ad una combinazione grafico nero su sfondo cyan), modificabile — nel modo citato nel testo — agendo sul penultimo valore nel DATA in 5Ø7ØØ (appunto 3).

Saltiamo alle linee 4ØØØ/4Ø1Ø, ove si trovano due POKE: la prima è relativa alla locazione 53265, ed abilita la mappatura bit per bit se il suo sesto bit (o bit 5, dato che i bit di un byte vengono numerati da Ø a 7) è posto ad uno: questo si ottiene con una POKE 53265,PEEK(53265)OR32, ovvero si disabilita con la POKE 53265, PEEK(53265)AND223.

La sub grafica in BASIC è contenuta nelle linee che vanno dalla 9ØØØ alla 1ØØ7Ø, e la sua gestione viene effettuata tramite una serie di GOSUB e di RETURN dislocati nel corso del programma.

Citiamo infine la linea 55555/6, che contiene una routine di uscita dal modo grafico, particolarmente utile ogni qualvolta il programma rimanga fermo in alta risoluzione. È un inconveniente che può verificarsi premendo inavvertitamente il tasto RUN/STOP, o anche se lo svolgimento del programma si è arrestato per qualche intoppo (ad es. una divisione per zero). In questo caso basta dare il RETURN, digitare da tastiera un GOTO 55555 (anche se sullo schermo appaiono solo dei quadratini colorati) e premere per tre o quattro volte di seguito il RETURN, senza curarsi dei SYNTAX ERROR: sarà poi sufficiente rilanciare il programma con il consueto RUN. MC