# Accesso ai 16K di RAM nascosta del C64

*di Sergio Cianchetta - Pisa*

Sul numero 74 di MC a pagina 248 è pubblicata una routine che «apparentemente» consente l'accesso a tutta la cosiddetta «RAM nascosta» del Commodore 64 usando la routine, mi sono accorto che il computer non accetta di buon grado la memorizzazione di dati nell'area di 4K da $D000 a $DFFF. Nel byte $0001 esistono 3 bit predisposti per gestire i banchi di memoria. Il bit 0 (segnale LO-RAM) che, se spento, disabilita la ROM da 8K dell'interprete Basic, il bit 1 (segnale HI-RAM) che gestisce allo stesso modo del precedente la ROM da 8K del sistema operativo e il bit 3 (segnale CHAREN) che, a differenza dei primi, agisce come se fosse un «deviatore» (mi si perdoni il termine poco adatto), nel senso che posto a livello 0 consente la lettura dei 4K destinati all'I/O, alla RAM colore, alla gestione del SID e ad altre cose ancora, al contrario se posto a 1 consente la lettura dei 4K destinati alla ROM del generatore di caratteri. Considerare tale area come RAM-utente è simile considerare la pagina-zero come RAM-utente: è possibile scriverci, ma ciò implicherebbe la cancellazione di dati importanti per la corretta gestione dell'I/O, del SID, ecc. da parte del calcolatore. Propongo una routine che permette la scrittura e la lettura di dati nelle aree disponibili ma nascoste, numerando tali locazioni da 0 a 16383 (16K). Naturalmente potrebbe essere interessante disporre di una vera e propria matrice numerica (Floating Point o Integer) o stringa completamente indipendente dall'area destinata al Basic (leggi memoria risparmiata) e quindi la routine si presta ad essere utilizzata per la memorizzazione di variabili, a patto però di conoscere come sono gestite dal C64. La routine è completamente in linguaggio macchina, rilocabile (variabile IN, riga 10) e semplice da utilizzare. La sintassi è: SYS(ind),(loc)[,(val)]; in cui (ind) rappresenta l'indirizzo della routine, (loc) rappresenta la locazione che si vuol leggere o scrivere (0-16383) e (val) indica il valore che si vuole inserire in (loc), omettendo tale parametro e si ottiene la sola lettura di (loc).

Nel disassemblato si può notare la doppia lettura della locazione nascosta, ciò a causa di un articolo che lessi diverso tempo fa circa un bug nelle operazioni sulla memoria, che però personalmente non ho mai riscontrato sul mio Commodore 64. In breve, dopo aver spento le ROM il primo dato letto dal banco di 8K abilitato è tutt'altro che quello richiesto: pare che la prima lettura sia atta è «risvegliare» la RAM restituendo un valore preso chissà dove. In $036A si può notare che ho sostituito le sequenza JSR$AEFD,JSR$B79E (controlla la presenza della virgola e legge parametro tra 0 e 225) con JSR$B7F1: si ottiene lo stesso risultato risparmiando 3 byte di memoria.

```
033C JSR $AEFD    ;Controlla virgola
033F JSR $AD8A    ;Valuta il parametro
0342 JSR $B7F7    ;Legge il numero compreso tra 0 e 65535
0345 CLC          ;Controlla se compreso tra 0 e 16383
0346 LDA $15      ;
0348 ADC #$C0     ;
034A BCC $0351    ;Si:salta a $0351
034C LDX #$0E     ;No:predispone errore di illegal quantity
034E JMP ($0300)  ;Segnala errore e torna al BASIC
0351 LDA $15      ;Aggiunge 40960 al numero di locazione
0353 ADC #$A0     ;
0355 STA $15      ;
0357 ADC #$40     ;Controlla se e' maggiore di 49151
0359 BCC $0361    ;No:salta a $0361
035B LDA $15      ;Si:aggiunge 8192 al numero di locazione
035D ADC #$1F     ;
035F STA $15      ;
0361 LDY #$00     ;Azzera registro Y
0363 JSR $0079    ;Controlla se esiste un altro parametro
0366 CMP #$2C     ;
0368 BNE $0374    ;No:salta alla continuazione (lettura dato)
036A JSR $B7F1    ;Si:(scrittura dato) legge parametro tra 0 e 255
036D TXA          ;Trasferisce nell'accumulatore
036E STA ($14),Y  ;Memorizza nella locazione puntata da $14 e $15
0370 RTS          ;Ritorna al BASIC
0371 CLI          ;Azzera l'interrupt
0372 LDX #$34     ;Disabilita ROM/BASIC e ROM/KERNAL
0374 STX $01      ;
0376 LDA ($14),Y  ;Legge la locazione puntata da $14 e $15
0378 LDA ($14),Y  ;
037A LDX #$37     ;Riabilita ROM/BASIC e ROM/KERNAL
037C STX $01      ;
037E SEI          ;Riabilita interrupt
037F STA $02      ;Memorizza il valore letto in $02
0381 RTS          ;Ritorna al BASIC
```

```
10 IN=828
20 DATA 32,253,174,32,138,173,32,247
30 DATA 183,24,165,21,105,192,144,5
40 DATA 162,14,108,0,3,165,21,105
50 DATA 160,133,21,105,64,144,6,165
60 DATA 21,105,31,133,21,160,0,32
70 DATA 121,0,201,44,208,10,32,241
80 DATA 183,138,145,20,96,88,162,52
90 DATA 134,1,177,20,177,20,162,55
100 DATA 134,1,120,133,2,96
110 FORS=0TO69:READA:POKEIN+S,A:NEXT
120 :
130 VL=17
140 FORLC=0TO16383
150 SYSIN,LC,VL:REM MEMORIZZA
160 SYSIN,LC    :REM LEGGE
170 REM LOCAZIONE,VAL.SCRITTO,VAL.LETTO
180 PRINTLC;VL;PEEK(2)
190 NEXT
READY.
```

*Listato in Basic del programma "Accesso ai 16K di RAM nascosta del C64".*

# Posizionamento degli sprite

*di Roberto Morassi - Pistoia*

Questa breve ma utilissima routine in linguaggio macchina permette di posizionare istantaneamente, da programma Basic, uno qualunque degli otto sprite con un solo comando SYS. È allocata a partire da $C000 (#49152), ma può essere rilocata a piacere in qualunque altra zona di memoria. La sintassi da usare è la seguente:

SYS 49152, N, X, Y

dove i parametri hanno il seguente significato:
**N:** numero dello sprite (da 0 a 255; la routine lo normalizza modulo 7);
**X:** coordinata X dello sprite (0-511);
**Y:** coordinata T dello sprite (0-255).

È inclusa una breve demo che mostra le potenzialità della routine: con RUN 200, lo schermo si animerà in una sarabanda di... fantasmi! MC

*Posizionamento degli sprite.*

```
100 REM ** DATAMASTER - BY R.MORASSI **
110 :
120 REM" ** ROUTINE 'DATAENTER' **
130 :
140 DATA 165,020,024,105,059,141,002,003
150 DATA 165,021,105,000,141,003,003,169
160 DATA 128,141,138,002,032,129,255,032
170 DATA 253,174,032,107,169,165,020,133
180 DATA 251,165,021,133,252,169,000,133
190 DATA 253,032,241,183,134,250,032,241
200 DATA 183,134,248,096,032,068,065,084
210 DATA 065,032,000,165,214,201,023,048
220 DATA 009,032,228,255,201,013,208,249
230 DATA 240,033,166,251,165,252,032,205
240 DATA 189,173,002,003,056,233,007,172
250 DATA 003,003,176,001,136,032,030,171
260 DATA 169,002,133,249,032,228,255,201
270 DATA 013,208,025,032,083,228,165,215
280 DATA 201,044,208,005,169,020,032,210
290 DATA 255,162,002,160,000,024,032,240
300 DATA 255,076,131,164,201,020,208,005
310 DATA 032,210,255,208,211,166,248,208
320 DATA 008,201,095,240,101,201,043,240
330 DATA 097,201,048,048,199,201,058,016
340 DATA 006,166,248,240,078,208,006,166
350 DATA 248,240,185,208,004,041,207,016
360 DATA 011,201,065,048,175,201,071,016
370 DATA 171,056,233,055,170,165,253,073
380 DATA 255,133,253,240,009,138,010,010
390 DATA 010,010,133,254,144,150,138,024
400 DATA 101,254,168,032,162,179,160,002
410 DATA 032,223,189,162,000,189,003,001
420 DATA 240,003,232,208,248,138,162,048
430 DATA 142,000,001,142,001,001,032,030
440 DATA 171,016,007,032,210,255,198,249
450 DATA 016,189,165,211,201,038,176,007
460 DATA 169,044,032,210,255,208,132,165
470 DATA 251,024,101,250,133,251,144,002
480 DATA 230,252,169,013,141,119,002,169
490 DATA 001,133,198,076,131,164,000,000
500 DATA 000,000,000,000,000,000,000,000
510 DATA 000,000
520 :
530 REM" ** ROUTINE 'DATAMEMORY' **
540 :
550 DATA 120,169,005,133,255,032,253,174
560 DATA 032,138,173,032,247,183,166,255
570 DATA 165,021,149,249,202,165,020,149
580 DATA 249,198,255,198,255,016,230,032
590 DATA 241,183,134,255,032,241,183,134
600 DATA 247,165,043,072,165,044,072,160
610 DATA 001,152,145,043,200,165,249,145
620 DATA 043,200,165,250,145,043,200,169
630 DATA 131,145,043,200,169,032,145,043
640 DATA 200,152,072,169,052,166,247,240
650 DATA 002,133,001,160,000,177,253,168
660 DATA 138,240,004,169,055,133,001,044
670 DATA 208,205,032,162,179,160,002,032
680 DATA 223,189,162,255,232,189,003,001
690 DATA 208,250,169,048,141,000,001,141
700 DATA 001,001,104,168,169,003,133,248
710 DATA 189,000,001,145,043,200,232,198
720 DATA 248,208,245,230,253,208,002,230
730 DATA 254,056,165,253,229,251,165,254
740 DATA 229,252,169,000,176,033,192,037
750 DATA 240,002,169,044,145,043,208,160
760 DATA 165,043,105,037,133,043,144,002
770 DATA 230,044,024,165,249,101,255,133
780 DATA 249,144,002,230,250,208,161,145
790 DATA 043,200,145,043,200,145,043,200
800 DATA 132,255,024,165,043,101,255,133
810 DATA 045,165,044,105,000,133,046,104
820 DATA 133,044,104,133,043,032,051,165
830 DATA 088,096
840 :
850 FORX=0TO523:READY:POKE49152+X,Y:A=A+Y:NEXT
860 IFA<>66431THENPRINT"ERRORE NEI DATA":STOP
870 END
880 :
1000 POKE43,0:POKE44,192:POKE45,12:POKE46,194
1010 SAVE"DATAMASTER/MC",8,1
1020 SYS64738

READY.
```