Commodore 64:

SCROLLING FINE E GRAFICA AD ALTA RISOLUZIONE

Seconda puntata del nostro viaggio all'interno del Commodore 64.
In questo numero completeremo la nostra carrellata sui modi carattere del 6567
con l'extended background color mode e la possibilità di scrolling fine,
e ci occuperemo degli ambienti di grafica alta risoluzione del 64.
Sono listate in queste pagine anche le relative routine per implementare una pagina grafica
(hgr o multicolor) senza sprecare Ram Basic, nonché due applicazioni:
un programma titolatrice illustrerà un esempio di scrolling fine
e il famoso MATH PACK (già presentato sul n. 16 per il VIC-20)
permetterà anche ai 64isti di studiarsi qualche funzioncina matematica.
Sempre che ne abbiano voglia!

di Andrea de Prisco e Leo Sorge

L'extended Background Color Mode

Il nome sta per modo colore di fondo esteso. Grazie a questa ulteriore possibilità offerta dal VIC II, è possibile, limitando a 64 il numero dei caratteri visualizzabili, scegliere fra 4 colori di fondo per ognuna delle 1000 locazioni di schermo. I codici dei quattro colori di fondo scelti andranno POKE-ati in opportuni registri del 6567, precisamente:
53281 col. di sfondo n. 1
53282 col. di sfondo n. 2
53283 col. di sfondo n. 3
53284 col. di sfondo n. 4.

Per attivare il "modo carattere a più colori" bisogna eseguire un
```
POKE 53265, PEEK (53265) OR64
```
per tornare ai caratteri standard
```
POKE 53265, PEEK (53265) AND191
```

La limitazione del numero di caratteri visualizzabili a soli 64 (contro i 256 del modo standard o multicolor) è dovuta al fatto che, degli 8 bit del codice schermo di ogni carattere, i primi due, i più significativi, non sono usati dal 6567 per selezionare il carattere nel generatore ma per scegliere il colore di fondo da abbinare al carattere generato dai rimanenti 6 bit.

E, come è facile verificare, con 6 bit si possono selezionare solo 64 caratteri diversi ($2\uparrow 6 = 64$). Se i bit 6 e 7 del codice sono una coppia di zeri, il colore di fondo usato sarà quello messo in 53281; se abbiamo la coppia 01, il colore è quello di 53282. Con la coppia 10 scegliamo il colore di 53283, infine con la coppia 11, il colore di 53284.

Facciamo un esempio: dopo aver attivato questo modo con la prima POKE sopra segnalata ed aver scelto i 4 colori da collocare nei byte 53281... 53284, proviamo a schiacciare qualche tasto.

Per selezionare il colore di sfondo n° 2, basta usare lo shift prima di qualsiasi tasto. Per i colori n° 3 e n° 4, bisogna andare in RVS ON e rispettivamente usare o non usare lo shift. Questo perché i codici di schermo dei caratteri shiftati (ragionando in binario) iniziano tutti con 01, i reverse con 10, i reverse + shift con 11. Es.: il carattere A ha codice di schermo 1. In binario, formato 8 bit:

```
Colore carattere
0 0 0 0 0 0 0 1
Shift A è 65
colore carattere
0 1 0 0 0 0 0 1
reverse A è 129
colore carattere
1 0 0 0 0 0 0 1
reverse shift A è 193
colore carattere
1 1 0 0 0 0 0 1
```

Scrolling fine

Come ben pochi altri personal, il Commodore 64 permette di eseguire lo scrolling fine di schermo (di un solo pixel per volta) nelle quattro direzioni. Si usa per far entrare "in campo" lentamente nuove informazioni, mentre lentamente vecchie informazioni spariscono dalla parte opposta. Il programma titolatrice, listato in queste pagine, ne è un esempio. Dando Run vengono richieste le linee da mostrare, in sequenza, lentamente (max 100). Si dà il via battendo Return all'ultima richiesta di input. Il VIC 6567 svolge gran parte del lavoro, ma non tutto. Per implementare lo scrolling fine bisogna scrivere un programma opportuno, preferibilmente in linguaggio macchina, se non si desidera uno scrolling esasperatamente lento. La prima cosa da fare è passare dalla solita pagina di 40 righe per 40 colonne alle 38 colonne per lo scrolling orizzontale o alle 24 righe se si desidera quello verticale. Ciò per far posto alle nuove informazioni prima dello scrolling vero e proprio. Per il movimento verticale verso l'alto, i passi sono:

1) Passare al modo 24 righe.
2) Impostare il registro di scrolling verticale al valore massimo (tutto lo schermo si abbassa di 8 pixel, nascondendo sotto il bordo inferiore la 25-esima riga).
3) Riempire la riga 25 con le informazioni da mostrare.
4) Variare lentamente il registro di scrolling in modo da far apparire la riga 25 e far scomparire la prima.
5) Con una routine in linguaggio macchina muovere il contenuto dello schermo di una posizione verso l'alto
6) ritornare al passo 2.

Per il movimento orizzontale l'algoritmo è sostanzialmente lo stesso: uniche ovvie differenze sono:

a) il modo da usare è quello a 38 colonne;
b) si agisce sul registro di scrolling orizzontale;
c) i nuovi dati andranno posizionati sull'estrema colonna di destra o di sinistra a seconda della direzione dello scroll.

```
100 FORI=52768TO52855:READII:POKEI,II:NEXT
110 REM ********************************
120 REM * T I T O L A T R I C E  6567 *
130 REM *                              *
140 REM *  (C)  1984  ADP - SOFTWARE  *
150 REM *                              *
160 REM *  ESEMPIO DI SCROLLING FINE  *
170 REM ********************************
180 PRINT"████          T I T O L A T R I C E             █"
190 DIMA$(100):I=0
200 A$="":INPUTA$:A=LEN(A$):IFA=0THEN230
210 IFA<39THENA$(I)=LEFT$("                                       ",19-A/2)+A$:I=I+1:GOTO200
220 A$(I)=LEFT$(A$,38):A$=MID$(A$,39):I=I+1:A=LEN(A$):GOTO210
230 P=PEEK(646):FORH=55296TO56295:POKEH,P:NEXT:POKE53265,PEEK(53265)AND247:K=0
240 PRINT"█████████████████████████████";
250 POKE53265,(PEEK(53265)AND248)OR7
260 PRINT"█"A$(K):K=K+1:IFK>ITHENK=0
270 FORP=6TO0STEP-1
280 POKE255,P:SYS52768
290 FORT=1TO50:NEXT
300 NEXT:SYS52791:GOTO260
310 DATA173,17,208,41,128,208,249,173,18,208,208,244,173,17,208,41,248,5,255
320 DATA141,17,208,96,120,173,17,208,41,128,208,249,173,18,208,201,251,208,242
330 DATA173,17,208,9,7,141,17,208,162,0,189,40,4,157,0,4,232,208,247,189,40
340 DATA5,157,0,5,232,208,247,189,40,6,157,0,6,232,208,247,189,40,7,157,0,7
350 DATA232,224,192,208,245,88,96
```

Diamo ora l'elenco delle POKE da usare:

POKE 53270, PEEK (53270) AND 247
seleziona il modo 38 colonne.
POKE 53270, PEEK (53270) OR 8
ritorna al modo standard 40 colonne.
POKE 53265, PEEK (53265) AND 247
seleziona il modo 24 righe.
POKE 53265, PEEK (53265) OR 8
ritorna al modo 25 righe.

Con X e Y compresi tra 0 e 7 tramite le due seguenti poke, si impostano i registri di scrolling orizzontale e verticale:

POKE 53270, (PEEK (53270) AND 248) OR XX
orizzontale
POKE 53265, (PEEK (53265) AND 248) OR YX
verticale

L'alta risoluzione

Passiamo ora a descrivere i modi grafici Bit-Map del VIC II. Come per il modo caratteri anche qui distinguiamo un modo Standard e un modo Multicolor. Tanto per cambiare, quest'ultimo, a spese della risoluzione orizzontale che anche in questo caso risulta dimezzata, permette di usare 4 colori per ogni pixel: più precisamente tre colori più il colore di sfondo. La risoluzione massima è di 320 × 200 pixel ed è data dal modo bit-map standard. La griglia risulta essere composta da 64.000 punti che mappati in RAM necessitano di 64.000 bit (detti anche 8.000 Byte). In altre parole, ogni pagina grafica mostrata su video è l'immagine di 8.000 byte di RAM. Per attivare il Bit-Mapping (sembra uno sport!) bisogna portare a 1 il bit 5 del registro locato a 53265; ciò avviene col comando:

POKE 53265, PEEK (53265) OR 32

per ritornare in ambiente testo basta resettare il medesimo bit con:

POKE 53265, PEEK (53265) AND 223

per passare al modo Multicolor, bisogna eseguire in aggiunta un:

POKE 53270, PEEK (53270) OR 16

mentre per disattivare il Multicolor:

POKE 53270, PEEK (53270) AND 239

Ricordando che il VIC II vede 16K RAM per volta, l'inizio degli 8000 byte destinati ad ospitare la pagina grafica, si seleziona all'interno del banco da 16K pre-

scelto, allo stesso modo dell'inizio della mappa carattere discusso sul numero scorso. Il comando è:

POKE 53272, (PEEK(53272) AND 240) OR A
sapendo che

A=0	pone l'inizio a	esadecim.	decimale
A=2	pone l'inizio a	$0800	2048
A=4	pone l'inizio a	$1000	4096
A=6	pone l'inizio a	$1800	6144
A=8	pone l'inizio a	$2000	8192

I byte della mappa grafica sono visualizzati nel seguente modo: la prima riga di schermo mostra i primi 320 byte, la seconda riga i secondi 320, e così via per tutte le 25 righe di schermo. A sua volta, ogni riga, alta 8 pixel, è composta da 40 pacchetti di 8 byte l'uno. Si dia uno sguardo alla figura 1 (pag. 116).

Di fatto, il modo Bit-Map non è che un'estensione del modo caratteri programmabili. È come se ogni pacchettino di 8 byte fosse un carattere in RAM e ben 1000 caratteri diversi fossero visualizzati uno accanto all'altro, contemporaneamente, sullo schermo. Vediamo ora come è possibile accendere punti sullo schermo una volta definita e mostrata sul video la pagina grafica. Nel caso di grafica 320 × 200, la massima, per accendere un pixel ad una determinata coordinata (X,Y), basta porre a 1 il bit corrispondente (all'interno di qualche byte), nella pagina grafica. È necessario calcolare 2 valori: il byte da modificare e, all'interno di esso, il bit da porre a 1. Senza dilungarci ulteriormente in calcoli, riporti, divisioni intere e resti diamo qui di seguito le due formule già belle e fatte. Se

```
0 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
1 REM x                            x
2 REM x   GRAFICA MCL 160 X 200    x
3 REM x   (C) 1984 ADP-SOFTWARE    x
4 REM x                            x
5 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
10 FORI=52192TO52284:READII:POKEI,II:NEX
T
12 C1=2:REM COLORE #1:RED
14 C2=5:REM COLORE #2:GREEN
16 C3=3:REM COLORE #3:CYAN
20 BA=57344:SYS52192:POKE53270,PEEK(5327
0)OR16:GOTO150
30 IFX<0ORX>319ORY<0ORY>199THENRETURN
40 XX=XAND510:A=(XXAND504):AA=INT((7-(XX
AND7))/2)
50 B=(YAND248)/8:BB=YAND7
60 P=BA+B*320+A+BB:POKE253,C*4↑AA:POKE25
2,255-3*4↑AA
70 POKE255,P/256:POKE254,(P/256-PEEK(255
))*256+.5:SYS52264:D=A/8+B*40
80 POKE49152+D,C1*16+C2:POKE55296+D,C3:R
ETURN
90 DATA173,2,221,9,3,141,2,221,173,0,221
,41,252,141,0,221,173,24,208
100 DATA41,1,9,8,141,24,208,173,17,208,9
,32,141,17,208,169,00,162,0
110 DATA157,0,192,157,0,193,157,0,194,15
7,0,195,202,208,241,169,224,133
120 DATA255,169,0,133,254,168,145,254
130 DATA200,208,251,230,255,208,247,96
140 DATA120,169,5,133,1,160,0,177,254,37
,252,5,253,145,254,169,7,133,1,88,96
150 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxx
160 REM x TUTTE LE RIMANENTI LINEE x
170 REM x OSPITERANNO IL PROGRAMMA x
180 REM x CHE SFRUTTA TALE GRAFICA x
190 REM x 'X' FRA 0 E 319          x
200 REM x 'Y' FRA 0 E 199          x
210 REM x 'C' FRA 0 E 3            x
220 REM x GOSUB 30                 x
230 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
0 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
1 REM x                            x
2 REM x   GRAFICA HGR 320 X 200    x
3 REM x   (C) 1984 ADP-SOFTWARE    x
4 REM x                            x
5 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
10 FORI=52192TO52282:READII:POKEI,II:NEX
T
20 BA=57344:SYS52192:GOTO150
30 IFX<0ORX>319ORY<0ORY>199THENRETURN
40 A=(XAND504):AA=7-(XAND7)
50 B=(YAND248)/8:BB=YAND7
60 P=BA+B*320+A+BB:POKE253,2↑AA
70 POKE255,P/256:POKE254,(P/256-PEEK(255
))*256+.5
80 SYS52264:RETURN
90 DATA173,2,221,9,3,141,2,221,173,0,221
,41,252,141,0,221,173,24,208
100 DATA41,1,9,8,141,24,208,173,17,208,9
,32,141,17,208,169,16,162,0
110 DATA157,0,192,157,0,193,157,0,194,15
7,0,195,202,208,241,169,224,133
120 DATA255,169,0,133,254,168,145,254
130 DATA200,208,251,230,255,208,247,96
140 DATA120,169,5,133,1,160,0,177,254,5,
253,145,254,169,7,133,1,88,96
150 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxx
160 REM x TUTTE LE RIMANENTI LINEE x
170 REM x OSPITERANNO IL PROGRAMMA x
180 REM x CHE SFRUTTA TALE GRAFICA x
190 REM x 'X' FRA 0 E 319          x
200 REM x 'Y' FRA 0 E 199          x
210 REM x GOSUB 30                 x
220 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

fissiamo l'origine (0,0) in alto a sinistra, l'indirizzo del byte all'interno della mappa grafica sarà dato da:

P = 320 • INT(Y/8) + 8 • INT(X/8) + Y AND 7

Il bit da settare è:

B = 7 - (X AND 7)

Se I è l'indirizzo del primo byte della mappa:

POKE P + I, PEEK (P + 1) OR 2 ↑ B

accenderà il pixel alla coordinata (X,Y) con cui sono stati calcolati P e B. Utilizzando il modo multicolor, l'affare si complica un tantino. Il P calcolato precedentemente resta invariato. All'interno del byte trovato, bisogna settare due bit per ogni pixel da

visualizzare, e prima di fare questo, bisogna resettarli a 00: ciò è indispensabile se si vuole cambiare colore a un pixel già acceso. Per resettare due bit all'interno di un byte, si esegue un AND logico tra il byte da modificare e un'opportuna maschera di 8 bit. Supponiamo di andare a mettere la coppia di bit 01 nella terza e quarta posizione di un byte che contiene:

1 1 0 1 1 1 1 0

la maschera sarà 1 1 1 1 0 0 1 1: eseguendo l'AND logico tra questi due byte otterremo:

1 1 0 1 0 0 1 0

Con un OR logico con 0 0 0 0 0 1 0 0, otterremo:

1 1 0 1 0 1 1 0

che è appunto il byte da cui siamo partiti con la coppia 01 inserita nel posto voluto.

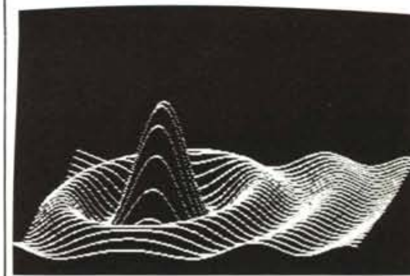
La scelta dei colori avviene in un modo assai interessante, anche se un po' macchinoso da gestire. In alta risoluzione standard, ad ogni bit a 1 corrisponde un pixel del colore indicato nei 4 bit più alti del corrispondente byte in pagina testo. Ogni bit a 0 visualizza un pixel del colore indicato nei 4 bit di ordine più basso, sempre dello stesso byte. La pagina grafica risulta così suddivisibile in 1000 regioni, corrispondenti alle 1000 posizioni di schermo: per ognuna di queste è così possibile scegliere il colore pixel e il bipixel (di larghezza doppia). La coppia 00 seleziona il colore il cui codice si trova in 53281 (colore schermo). La coppia 01 seleziona il colore dato dai 4 bit di ordine più alto del byte corrispondente in mappa video; la coppia 10, il colore dei 4 bit di ordine più basso; la coppia 11 il colore indicato nella mappa colore, sempre nella locazione corrispondente alle coordinate (X,Y) del pixel.

Particolari tecnici

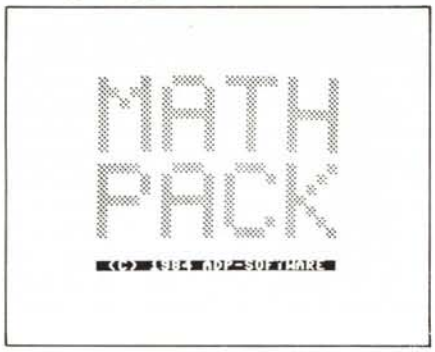
Le due routine presentate, Grafica Hgr e Grafica Mcl, provvedono rispettivamente a gestire la grafica alta risoluzione nei modi Standard e Multicolor. Le linee listate inizializzano le due grafiche. A partire dalla linea 150 in poi è possibile mettere il programma Basic che sfrutta tale ambiente di grafica. Si assegna alla variabile X l'ascissa, ad Y l'ordinata, se si usa il multicolor, a C il colore: 0, 1, 2, 3. GOSUB 60 setterà il pixel voluto. Anche se la risoluzione orizzontale, in multicolor, risulta dimezzata, il campo di variabilità della X è stato lasciato tra 0 e 319, in modo da poter facilmente passare da una grafica all'altra, senza modificare i programmi ospite. Naturalmente, in multicolor, due ascisse consecutive di cui la prima pari, ad es. 198 e 199, individuano lo stesso pixel in campo. Per non sprecare Ram del Basic la mappa grafica è stata posta negli ultimi 8K dei 64 disponibili, a partire quindi dal byte 57344. Questa scelta ha portato a complicare un po' la gestione della grafica, ma offre il considerevole vantaggio di non disperdere Ram utente.

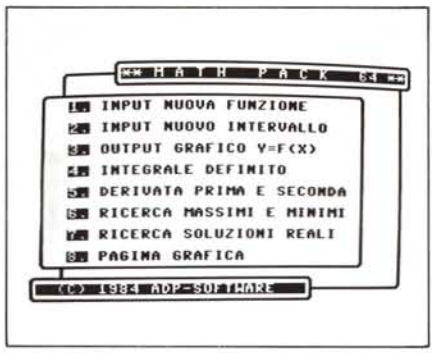
Per meglio comprendere il funzionamento delle due routine, è bene a questo punto descrivere la gestione della memoria

```
200 REM ******************************
210 REM *QUESTE LINEE, AGGIUNTE ALLA*
220 REM * GRAFICA HGR, TRACCIANO IL *
230 REM * GRAFICO DELLA FUNZIONE 3D *
240 REM *       Z=SIN(Q)/Q CON       *
250 REM *       Q=SQR(X↑2+Y↑2)       *
260 REM ******************************
1000 DIMA%(320):T=15
1010 FORY0=-9.5TO19STEP.7
1020 FORX0=-9.5TO19STEP.08
1030 Q=SQR(X0*X0+Y0*Y0)
1040 Y=INT((SIN(Q)/Q)*75+T*1.3)
1050 X=INT((X0+9.5)*9.5)+T*.92-14
1060 IFX>0ANDX<320THENIFY>A%(X)THENA%(X)=Y:
Y=199-Y:GOSUB30
1070 NEXT:T=T+2:NEXT
1080 GOTO1080
```



```
100 POKE53280,5:POKE53281,1
110 GOSUB1440:GOTO220
120 REM ********************
130 REM * SUBROUTINE  PLOT *
140 REM ********************
160 IFX<0ORX>319ORY<0ORY>199THENRETURN
170 A=(XAND504):AA=7-(XAND7)
180 B=(YAND248)/8:BB=YAND7
190 P=57344+B*320+A+BB:POKE253,2↑AA
200 POKE255,P/256:POKE254,(P/256-PEEK(255))*256+.5
210 SYS52264:RETURN
220 PRINT"
230 PRINT"        ** M A T H   P A C K   64 **!"
240 PRINT"
250 PRINT"
260 PRINT"     |  1. INPUT NUOVA FUNZIONE       |  |
270 PRINT"     |                                |  |
280 PRINT"     |  2. INPUT NUOVO INTERVALLO     |. |
290 PRINT"     |                                |  |
300 PRINT"     |  3. OUTPUT GRAFICO Y=F(X)      |  |
310 PRINT"     |                                |  |
320 PRINT"     |  4. INTEGRALE DEFINITO         |  |
330 PRINT"     |                                |  |
340 PRINT"     |  5. DERIVATA PRIMA E SECONDA   |  |
350 PRINT"     |                                |  |
360 PRINT"     |  6. RICERCA MASSIMI E MINIMI   |  |
370 PRINT"     |                                |  |
380 PRINT"     |  7. RICERCA SOLUZIONI REALI    |  |
390 PRINT"     |                                |  |
400 PRINT"     |  8. PAGINA GRAFICA             |  |
410 PRINT"                                         |
420 PRINT"                                         |"
430 PRINT"  |  (C) 1984 ADP-SOFTWARE   |-----------"
440 PRINT"                                          "
450 GETZ$:IFZ$<"1"ORZ$>"8"THEN450
460 PRINT"J":ONVAL(Z$)GOTO500,520,570,1040,950,1280,1120,860
470 REM ********************
480 REM *  NUOVA FUNZIONE  *
490 REM ********************
500 INPUT"JF(X)=";Z$:PRINT"530DEFFNY(X)=";Z$:PRINT"RUN 220"
510 POKE198,2:POKE631,13:POKE632,13:END
520 GOSUB890:X1=T1:X2=T2:GOTO220
530 DEFFNY(X)=SIN(X)/X*2
540 DEFFNY1(X)=(FNY(X+1E-4)-FNY(X-1E-4))/2E-4
550 DEFFNY2(X)=(FNY1(X+1E-4)-FNY1(X-1E-4))/2E-4
560 DEFFNAR(Y)=INT(Y*1E4+.5)/1E4:RETURN
570 IFX1=X2THEN220
580 Z$="SI":FL=0:GOSUB530:INPUT"ASSI CARTESIANI ";Z$:IFZ$="SI"THENFL=1
590 Y1=0:Y2=0
600 IFFL=0THENZ$="NO":INPUT"SOVRAPPOSIZIONE ";Z$:IFZ$="SI"THENSYS52229:GOTO0710
610 SYS52191
620 REM ********************
630 REM * GRAFICO DI  F(X) *
640 REM ********************
650 PRINT"J":Y1=1E30:Y2=-1E30
660 FORX=X1TOX2STEP(X2-X1)/90
670 Y=-FNY(X)
680 IFY<Y1THENY1=Y
690 IFY>Y2THENY2=Y
700 NEXT:SYS52229
710 U=0:P=0:IFY1<>Y2THENKY=197/(Y2-Y1):A$=STR$(Y1)+"@"+STR$(KY):GOTO1240
720 U=U+1:IFPEEK(50683+U)<>35THENA$=A$+CHR$(PEEK(50683+U)):GOTO720
730 P=P+1:IFMID$(A$,P,1)<>"@"THEN730
740 Y1=VAL(LEFT$(A$,P-1)):KY=VAL(MID$(A$,P+1,20))
750 IFX1<>X2THENKX=319/(X2-X1)
760 IFX1>=0ORX2<=0ORFL<>1THEN790
770 Y=0:X=-X1*KX:GOSUB160:FORI=PTOP+7680STEP320
780 FORJ=ITOI+7:POKEJ,2↑AA:NEXTJ,I
790 IFY1>=0ORY2<=0ORFL<>1THEN810
800 X=0:Y=-Y1*KY+1:GOSUB160:FORI=PTOP+312STEP8:POKEI,255:NEXT
810 FORX0=X1TOX2STEP(X2-X1)/320
820 X=(X0-X1)*KX:Y=(-FNY(X0)-Y1)*KY+1
830 GOSUB160
840 NEXT
850 GOTO870
860 SYS52229
870 GETZ$:IFZ$=""THEN870
880 SYS52283:GOTO220
890 INPUT"INTERVALLO";T1,T2:IFT1>T2THENT=T1:T1=T2:T2=T
```






```
900 IFT1=T2THEN890
910 RETURN
920 REM ********************
930 REM *   DERIVAZIONE    *
940 REM ********************
950 PRINT"■■■DERIVAZIONE■":GOSUB530
960 Z#="":INPUT"■■■■■■X=";Z#:IFZ#=""THEN220
970 XX=VAL(Z#):Y=FNY(XX):Y=FNAR(Y):PRINT"■■F (X)=";Y;"■■      "
980 Y=FNY1(XX):Y=FNAR(Y):PRINT"■■F'(X)=";Y;"■■      "
990 Y=FNY2(XX):Y=FNAR(Y):PRINT"■■F''(X)=";Y;"■■      "
1000 GOTO960
1010 REM ********************
1020 REM *   INTEGRAZIONE   *
1030 REM ********************
1040 PRINT"■■■INTEGRALE DEFINITO■■":GOSUB890:INPUT"■N. SUDDIVISIONI";C:GOSUB530
1050 J1=FNY(T1):S=(T2-T1)/C:A=0:FORXX=T1+$TOT2STEPS:J2=FNY(XX):A=A+S*(J2+J1)/2
1060 J1=J2:NEXT:A=FNAR(A):PRINT"■■■■■■F(X)DX=";A
1070 GETZ#:IFZ#=""THEN1070
1080 GOTO220
1090 REM ********************
1100 REM * SOLUZIONI  REALI *
1110 REM ********************
1120 PRINT"■■■SOLUZIONI REALI■■":GOSUB890:GOSUB530:FR=0
1130 PRINT"■■ASCISSA                ■"
1140 R=(T2-T1)/100
1150 J1=SGN(FNY(T1)):S=R
1160 T1=T1+S:J3=FNY(T1):J2=-SGN(J3)*(ABS(J3)>1E-8):IFJ1<>J2THEN1190
1170 IFT1<T2THEN1160
1180 PRINT"■■        STOP !        ■":GOTO1070
1190 IFFR=0THENSS=S:FR=1
1200 IFJ1=0THENZ=T1-S:GOTO1230
1210 IFJ2=0THENZ=T1:GOTO1230
1220 T1=T1-S:S=S/2:J1=SGN(FNY(T1)):GOTO1160
1230 Z=FNAR(Z):PRINTZ:FR=0:T1=T1+R:GOTO1150
1240 FORU=1TOLEN(A$):POKE50683+U,ASC(MID$(A$,U,1)):NEXT:POKE50683+U,35:GOTO750
1250 REM ********************
1260 REM *    MAX  &  MIN    *
1270 REM ********************
1280 PRINT"■■■ MAX & MIN ■■":GOSUB890:GOSUB530:FR=0
1290 PRINT"■■ASCISSA  ###  ORDINATA■"
1300 R=(T2-T1)/50
1310 J1=SGN(FNY1(T1)):S=R
1320 T1=T1+S:J3=FNY1(T1):J2=-SGN(J3)*(ABS(J3)>1E-20):IFJ1<>J2ORS<1E-15THEN1350
1330 IFT1<T2THEN1320
1340 GOTO1100
1350 IFFR=0THENSS=S:FR=1
1360 IFJ1=0THENZ=T1-S:GOTO1390
1370 IFJ2=0THENZ=T1:GOTO1390
1380 T1=T1-S:S=S/2:J1=SGN(FNY1(T1)):GOTO1320
1390 ZZ=FNAR(Z):T=FNAR(FNY(Z)):Z#=LEFT$("     ",4-LEN(STR$(INT(T))))+STR$(T)
1400 J3=FNAR(FNY2(ZZ))
1410 IFSGN(J3)=1THENPRINTZZ,"■■■MIN ";Z#
1420 IFSGN(J3)=-1THENPRINTZZ,"■■■AX ";Z#
1430 FR=0:T1=T1+R:GOTO1310
1440 PRINT"■■■■■":
1450 PRINT"          ■   ■  ■■■   ■■■■■  ■    ■"
1460 PRINT"          ■■ ■■  ■   ■    ■    ■    ■"
1470 PRINT"          ■ ■ ■  ■   ■    ■    ■    ■"
1480 PRINT"          ■   ■  ■■■■■    ■    ■■■■■"
1490 PRINT"          ■   ■  ■   ■    ■    ■    ■"
1500 PRINT"          ■   ■  ■   ■    ■    ■    ■"
1510 PRINT"          ■   ■  ■   ■    ■    ■    ■■"
1520 PRINT"          ■■■■   ■■■■   ■■■■   ■    ■"
1530 PRINT"          ■   ■  ■   ■  ■   ■  ■    ■"
1540 PRINT"          ■   ■  ■   ■  ■   ■  ■    ■"
1550 PRINT"          ■■■■   ■■■■■  ■    ■■■■■■"
1560 PRINT"          ■      ■   ■  ■   ■  ■    ■"
1570 PRINT"          ■      ■   ■  ■   ■  ■    ■"
1580 PRINT"          ■      ■   ■  ■■■■   ■    ■"
1590 PRINT"          ■■■ (C) 1984 ADP-SOFTWARE ■"
1600 FORK=52191TO52314:READKK:POKEK,KK:NEXT
1610 SYS52191
1620 RETURN
1630 DATA169,16,162,0,157,0,192,157,0,193,157,0,194,157,0,195,202,208,241,169
1640 DATA224,133,255,169,0,133,254,168,145,254,200,208,251,230,255,208,247,96
1650 DATA173,2,221,9,3,141,2,221,173,0,221,41,252,141,0,221,173,24,208,41,1,9,8
1660 DATA141,24,208,173,17,208,9,32,141,17,208,96,120,169,5,133,1,160,0,177,254
1670 DATA5,253,145,254,169,7,133,1,88,96,173,2,221,9,3,141,2,221,173,0,221,41
1680 DATA252,9,3,141,0,221,169,20,141,24,208,173,17,208,41,223,141,17,208,96
```

del Commodore 64. La Ram interna alla macchina è di 64K byte, anche se, come arci-noto, da Basic ne sono accessibili soltanto 38. Essendo infatti il 6510 un microprocessore che indirizza al più 64K fra Ram e Rom, ed essendo necessaria "un po'" di Rom per il linguaggio e per il sistema operativo e un po' di Ram per il video e le variabili di sistema, è facile convincersi che 64K per il Basic sono davvero impensabili. Di fatto però i sessantaquattrocapparam ci stanno. Dove?? La Ram per così dire "in più", "doppia" la Rom esistente. È come se sotto la Rom ci sia (viva) della Ram, in qualche modo utilizzabile. Tanto per cominciare, una PEEK eseguita in una zona di memoria doppia Ram-Rom ritorna il contenuto della Rom: una POKE nella stessa zona, di contro, resta memorizzata in Ram. A questo punto è d'obbligo chiedersi se è possibile leggere qualcosa da una di queste Ram. È un po' complicato, ma è possibile: non da Basic, ma da linguaggio macchina. La locazione 1 della memoria del 64 è una porta I/O del 6510. Settando opportunamente alcuni bit di questa cella, è possibile manipolare l'organizzazione della memoria. Portando a 0 il bit 0 di tale registro, la Rom del linguaggio ($A000 - $BFFF) viene disattivata e contemporaneamente emerge la Ram sottostante. Facendo lo stesso con il bit 1 potremo "sganciare" il sistema operativo ($E000-$FFFF).

Disattivando la Rom del Basic da linguaggio macchina non si hanno problemi di alcun tipo. È superfluo dire che al L.M., del Basic non gliene importa proprio nulla! Un tantino più delicato è sganciare il sistema operativo. Ciò perché ogni 60-simo di secondo il microprocessore, anche se sta eseguendo un programma in linguaggio macchina, è brutalmente interrotto da un interrupt e obbligato a eseguire la routine di scansione della tastiera che risiede in Rom. Se al momento dell'interrupt la Rom non c'è, succede un macello: il processore cerca di eseguire una routine che... "non c'è" (leggi blocco totale del sistema). È come togliere, di nascosto, l'acqua dalla piscina ad un tuffatore accanito che continuamente si butta dal trampolino (ih! ih!). Fortunatamente si può raggiungere l'ostacolo semplicemente avvertendo il tuffatore di non buttarsi per qualche attimo (anzi dicendogli tuffati... quando lo dico io!). Ritornando ai nostri chip, è possibile mascherare questa interruzione dando da L.M. il comando SEI (set interrupt disable), ossia rendendo il microprocessore sordo a qualsiasi richiesta di interruzione mascherabile. In definitiva, per settare un bit della Ram alternativa al sistema operativo è necessario:

1) settare la mascherazione dell'interrupt;
2) sganciare il sistema operativo;
3) leggere il byte da modificare;
4) eseguire l'OR con opportune maschere per modificare i bit;
5) riscrivere in Ram il byte modificato;
6) riagganciare il S.O;
7) ripristinare le interruzioni.

Semplice, no?

Math Pack 64

Questo programma è già stato presentato sul n. 16 di MC in versione VIC-20. Come per molti altri programmi ancora in cantiere, si sta cercando di moficare i più interessanti anche per il 64. Questo in particolare risulta essere abbastanza "precisospiccicato-identico" alla versione per VIC, per quel che concerne le avvertenze e le modalità d'uso. Le modifiche stanno sostanzialmente nelle routine in linguaggio macchina adoperate, e nel fatto che non bisogna spostare l'inizio della memoria Ram per fare largo alla pagina grafica, come accadeva per il Vic. Il Math Pack permette lo studio di funzioni continue (o con discontinuità eliminabile) del tipo $y=f(x)$: funzioni reali di variabile reale. Dando Run, subito dopo la pausa di inizializzazione, appare il menu (bello, vero?). È possibile tracciare il grafico di una funzione, ricercare i punti di intersezione con l'asse X (i cosiddetti zeri); calcolare massimi e minimi relativi; il valore di y, y', y'' in un punto; approssimare l'integrale definito col metodo delle suddivisioni.

Tutte le tecniche adoperate per lo studio non hanno la pretesa di sostituire il metodo analitico-manuale-sudereccio; non c'è da stupirsi se con funzioni particolarmente contorte (come chi le inserisce), qualche zero non venga azzeccato o qualche massimo sia scambiato per un minimo. Ritornando all'artistico menu (c'è voluta qualche ora per partorirlo), con l'opzione 1 si inserisce la funzione da studiare. L'opzione 2 permette di impostare l'intervallo di cui è richiesto il grafico. Questa operazione è

```
L     byte 0    byte  8   byte 16   byte 24  ..................  byte 312
I     byte 1    byte  9     .        .                         byte 313
N     byte 2    byte 10     .        .                         byte 314
E     byte 3    byte 11     .        .                         byte 315
A     byte 4    byte 12     .        .                         byte 316
      byte 5    byte 13     .        .                         byte 317
1     byte 6    byte 14     .        .                         byte 318
      byte 7    byte 15   byte 23   byte 31  ..................  byte 319

L     byte 320  byte 328  byte 336  byte 344 ..................  byte 632
I     byte 321  byte 329    .        .                         byte 633
N     byte 322  byte 330    .        .                         byte 634
E     byte 323  byte 331    .        .                         byte 635
A     byte 324  byte 332    .        .                         byte 636
      byte 325  byte 333    .        .                         byte 637
2     byte 326  byte 334    .        .                         byte 638
      byte 327  byte 335  byte 343  byte 351 ..................  byte 639

        .                                                        .
        .                                                        .
        .                                                        .
        .                                                        .
        .                                                        .
        .                                                        .
        .                                                        .
        .                                                        .
        .                                                        .
        .                                                        .
        .                                                        .

L     byte 7680 ..............................................  byte 7992
I     byte 7681                                                 byte 7993
N     byte 7682                                                 byte 7994
E     byte 7683                                                 byte 7995
A     byte 7684                                                 byte 7996
      byte 7685                                                 byte 7997
2     byte 7686                                                 byte 7998
5     byte 7687 ..............................................  byte 7999
```

Figura 1 - Disposizione dei Byte della pagina grafica del 64.

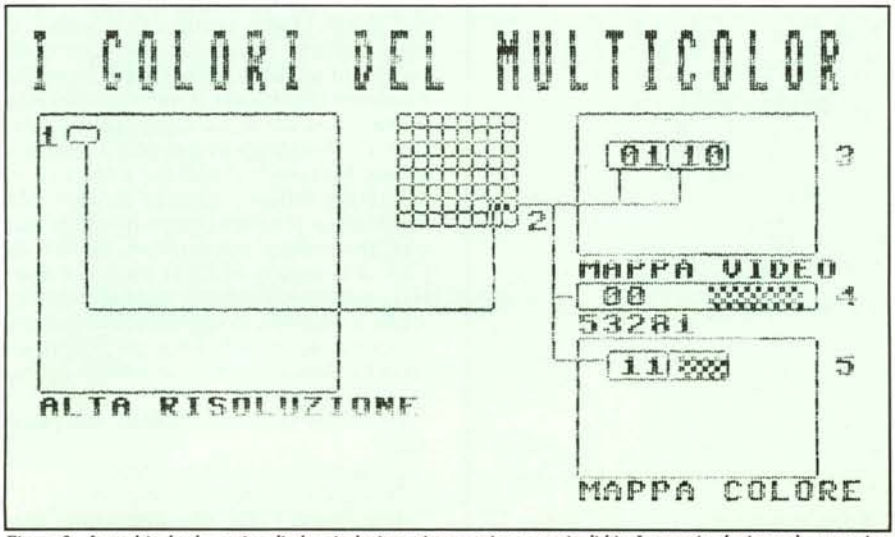
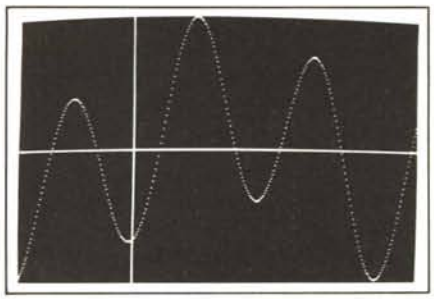


Figura 2 - In multicolor la pagina di alta risoluzione viene gestita a coppie di bit. Lo spazio che in modo testo viene occupato da un carattere (in figura è indicato con il numero 1) viene scomposto in una griglia di 8 × 8 punti (2), della quale si considerano due bit per volta: a seconda che il contenuto sia 00, 01, 10 o 11, il sistema va a pescare il codice del colore di quel punto dai quattro bit alti della locazione 53281 (00), dai quattro alti della locazione che nella pagina video occupa la stessa posizione (01), ovvero dai suoi quattro bit bassi (10), oppure infine dai quattro bit alti della locazione che nella mappa di colore rappresenterebbe il byte della pagina testo (11).



obbligatoria e va ripetuta se si cambia funzione. La scelta 3 serve appunto per tracciare il grafico di f. Si può far tracciare o meno gli assi cartesiani (ammesso che la funzione li intersechi) e se si vuole, si può sovrapporre il grafico alla funzione precedentemente tracciata. Per far si che la funzione tracciata occupi in altezza tutti i 200 pixel disponibili, vengono dapprima calcolati i punti di massimo e minimo assoluti (relativi all'intervallo considerato) e poi, con semplici trasformazioni lineari, ogni y è plottata proporzionalmente nel punto giusto dello schermo. Se è richiesto che la funzione sia sovrapposta alla precedente, come coefficienti di dilatazione o contrazione del campo sono adoperati quelli relativi alla funzione prima tracciata, per non falsare la scala. L'opzione 4 riguarda l'integrazione della funzione in memoria, nel senso di area sottesa alla curva. Per calcolare l'area, oltre all'intervallo bisogna indicare il numero di suddivisioni da effettuare. Un maggior numero significa una maggior precisione di calcolo, ma anche un tempo di computazione più lungo. Generalmente 100-200 suddivisioni sono più che sufficienti. Se da menu è schiacciato il tasto 5, è possibile input-are un qualsiasi valore di ascissa per conoscere in quel punto il valore della funzione, della sua derivata prima e seconda. Per tornare al menu, cancellare il video con Shift + Clr/Home e battere [RETURN]. Con le opzioni 6 e 7 vengono ricercati massimi, minimi e zeri della funzione. In tutti i casi bisogna indicare l'intervallo in cui va effettuata la ricerca. Terminata questa fase, dopo l'apparizione della stringa "STOP!", con la pressione di qualsiasi tasto si torna al menu. Ciò vale anche quando si vuol passare dal modo grafico al menu. L'opzione 8 esegue esattamente il contrario: da menu si passa al grafico precedentemente tracciato. Buon divertimento! MC

CARATTERI UTENTE & SPRITE

I listati pubblicati lo scorso mese nell'articolo "Caratteri utente e sprite (pg. 101 - 102 - 103)" sono risultati illegibili.
Scusandocene con i lettori, li ripubblichiamo per intero.

```
5 POKE53280,3:POKE53281,1:FORI=49152TO49185:READII:POKEI,II:NEXT
10 PRINT"■■       ■■■■ *** CHAR - EDITOR ***  ■"
20 PRINT"■■             ■1■ MAPPA RAM"
30 PRINT"■             ■2■ EDITOR"
40 PRINT"■             ■3■ SAVE MAPPA"
50 PRINT"■             ■4■ LOAD MAPPA"
60 PRINT"■■           ■  (C) 1984 ADP-SOFTWARE  ■"
70 GETA$:IFA$<"1"ORA$>"4"THEN70
80 ONVAL(A$)GOTO100,150,500,700
81 ***********************************
82 *                                 *
83 *  C H A R  -  E D I T O R   6 4  *
84 *                                 *
85 *   (C)   1984   ADP - SOFTWARE   *
86 *                                 *
87 ***********************************
88 *                                 *
89 *  PRIMA DI DIGITARE O CARICARE   *
90 *   QUESTO  PROGRAMMA, ESEGUIRE   *
91 *                                 *
92 *   POKE 44,16:POKE 4096,0:NEW    *
93 *                                 *
94 * PER SPOSTARE L'INIZIO PROGRAMMI *
95 *     BASIC, VARIABILI E ARRAY.   *
96 *                                 *
99 ***********************************
100 POKE53272,(PEEK(53272)AND240)OR2
105 POKE56334,PEEK(56334)AND254
110 POKE1,PEEK(1)AND251
115 SYS49152
120 POKE1,PEEK(1)OR4
125 POKE56334,PEEK(56334)OR1
130 GOTO70
150 INPUT"■■■S■TANDARD O ■M■ULTICOLOR";A$:IFA$<>"S"ANDA$<>"M"THEN150
160 T=(A$="S")+2:INPUT"■CARATTERE DA MODIFICARE ";A$:PRINT"■■";A$;"■"
170 C=PEEK(1024):D=ASC(A$):CO=0:RI=0:PU=1024
180 PRINT"■          ■"
185 PRINT"          ■  F1 - BLANK     "
190 PRINT"          ■  F3 - COL 1     "
195 PRINT"          ■  F5 - COL 2     "
200 PRINT"          ■  F7 - COL 3     "
205 PRINT"          ■"
210 PRINT"          ■  [RET] MEMORIZZA "
215 PRINT"          ■"
225 PRINT"■■■■■■■■■■"
230 FORI=0TO7:A$(I)="00000000":NEXT
250 GETA$:POKEPU,PEEK(PU)+128*((PEEK(PU)>127)+.5)*2:IFA$=""THEN250
255 POKEPU,PEEK(PU)AND127
260 IFA$="■"THEN:GOSUB400:CO=CO+T:GOSUB470
265 IFA$="■"THENA$="01":GOSUB410:CO=CO+T:GOSUB470
270 IFA$="■"THENA$="10":GOSUB410:CO=CO+T:GOSUB470
275 IFA$="■"THENA$="11":GOSUB410:CO=CO+T:GOSUB470
280 IFA$="■"THEN170
285 IFA$="■"THENRI=0:CO=0:PU=1024
290 IFA$="■"THENRI=RI+1:GOSUB470
292 IFA$="■"THENRI=RI-1:GOSUB470
294 IFA$="■"THENCO=CO-T:GOSUB470
295 IFA$="■"THENCO=CO+T:GOSUB470
297 IFA$=CHR$(13)THEN300
299 GOTO250
300 FORI=0TO7:X=0:FORJ=8TO1STEP-1
310 X=X-(MID$(A$(I),J,1)="1")*2↑(8-J):NEXT
320 POKE2048+C*8+I,X:NEXT:IFT=2THEN340
330 PRINTCHR$(D):WAIT197,191:GETZ$:GOTO10
340 INPUT"COLORE 1";A:POKE53282,A
345 INPUT"COLORE 2";A:POKE53283,A
350 INPUT"COLORE 3";A:POKE55976,AOR8:POKE1704,C
355 POKE53270,PEEK(53270)OR16:FORI=1TO2000:NEXT:WAIT197,191
360 GETZ$:POKE53270,PEEK(53270)AND239:GOTO10
400 IFT=1THENA$(RI)=LEFT$(A$(RI),CO)+"0"+MID$(A$(RI),CO+2):POKEPU,32:RETURN
405 A$(RI)=LEFT$(A$(RI),CO)+"00"+MID$(A$(RI),CO+3):POKEPU,32:POKEPU+1,32
408 RETURN
410 IFT=1THENA$(RI)=LEFT$(A$(RI),CO)+"1"+MID$(A$(RI),CO+2):POKEPU,81:RETURN
415 A$(RI)=LEFT$(A$(RI),CO)+A$+MID$(A$(RI),CO+3):A=VAL(A$)+3
420 POKEPU,81:POKEPU+54272,A:POKEPU+1,81:POKEPU+54273,A:RETURN
470 IFCO=8THENCO=0:RI=RI+1
475 IFCO<0THENCO=CO+8:RI=RI-1
480 IFRI=8THENRI=0
485 IFRI=-1THENRI=7
490 PU=1024+CO+RI*40:RETURN
500 OPEN4,8,4,"CARATTERI,S,W":FORI=2048TO4095:PRINT#4,CHR$(PEEK(I));:NEXT
510 CLOSE4:GOTO10
700 OPEN4,8,4,"CARATTERI,S,R":FORI=2048TO4095:GET#4,A$:IFA$=""THENA$=CHR$(0)
710 POKEI,ASC(A$):NEXT:CLOSE4:GOTO10
1000 DATA169,0,133,252,133,254,169,208,133,255,169,8,133,253,160,0,177
1010 DATA254,145,252,200,208,249,230,255,230,253,165,255,201,215,208,239,96
```

```
10000 REM ***********************
10010 REM **  DISEGNA SPRITE  **
10020 REM ***********************
10100 IF PEEK(44)=16ANDPEEK(43)=1ANDPEEK(4096)=0THEN GOTO 12000
10200 POKE 53280,10:POKE53281,2
10210 PRINTTAB(210)"ATTENZIONE!"
10220 PRINT"DIGITARE POKE44,16:POKE43,1:POKE4096,0:NEW"
10230 PRINT"E RILEGGERE IL PROGRAMMA"
10240 STOP:END
12000 REM**************************
12002 REM AZIONA PUNTATORI SPRITE
12005 REM**************************
12010 SP=2040:SE=53269:SS=2048
12020 DEF FNS(SN)=SS+64*(SN)
12030 SN=0:DIM TT%(20,23):POKE53281,6
13000 REM**************************
13002 REM  DISEGNA LA GRIGLIA
13005 REM**************************
13020 POKE53269,1:POKE53287,1
13030 POKESP,FNS(SN)/64
13040 POKE53248,0:POKE53264,1:POKE53249,80
13050 PRINT"■";:FORI=1TO11:PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■ "
13060 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■":NEXT
13070 PRINT"■■■                          "
13080 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■"
13090 PRINT"■■";:FORI=FNS(SN)TOFNS(SN)+62STEP3:FORJ=0TO2
13100 FORK=7TO0STEP-1
13110 IF(PEEK(I+J)AND2↑K)=2↑KTHENPRINT"●";
13120 IF(PEEK(I+J)AND2↑K)=0THENPRINT" ";
13130 NEXTK,J:PRINT:NEXT
13140 PRINT"■■NUM.  PROGETTO";SN
13150 INPUT"DI QUANTO MUOVO IL PUNT.(0=REDEF.):";MM:SN=SN+MM
13160 IF SN< 0 THEN SN=0
13170 IF SN>31 THEN SN=31
13180 IF MM=0 THEN POKE 53248,0:POKE53264,1:POKE53249,200:GOTO 14000
13190 GOTO 13000
14000 REM**************************
14002 REM   RIDEFINISCE  SPRITE
14005 REM**************************
14030 PRINT"■                                  ■"
14040 F$="■■■■■■■■■■■■■■■■■■■■■■■■■■■"
14050 PRINTF$;"■'I' INVERTE"
14060 PRINTF$;"■'M' SPECCHIO"
14070 PRINTF$;"■'R' RETURN"
14080 PRINTF$;"■'1' SCRIVE"
14090 PRINTF$;"■'0' CANCELLA"
14100 PRINTF$;"■'T' GIRA  "
14110 PRINTF$;"■'P' MEMORIA"
14112 PRINTF$;"■'X' SCAMBIA"
14114 PRINTF$;"■'C' COLORE"
14120 PRINTF$;"■'S' SALVA "
14130 PRINTF$;"■'L' CARICA"
14140 PRINTF$;"■'E' FINE  "
14141 PRINTF$;"■'F1'X-ESP "
14142 PRINTF$;"■'F3'Y-ESP "
14143 PRINTF$;"■'F5'X-RID "
14144 PRINTF$;"■'F7'Y-RID "
14150 PRINTF$;"■CURS. MUOVE"
14160 PRINT"■";
14170 GET A$
14180 CC=PEEK(211)+PEEK(210)*256+PEEK(209):PP=PEEK(CC):C1=55296+CC-1024
14190 C2=PEEK(C1)
14200 POKECC,42:POKEC1,1
14210 FORI=1TO15:NEXT:POKECC,PP:POKEC1,C2:IFA$=""THEN14170
14220 P1=INT((CC-1024)/40):P2=CC-(1024+40*P1)
14230 IF(P1>0ANDA$="■")OR(P1<20ANDA$="■")THENPRINTA$;:GOTO14170
14240 IF(P2>0ANDA$="■")OR(P2<23ANDA$="■")THENPRINTA$;:GOTO14170
14250 IFA$="1"THENPRINT"■■■";:GOTO14170
14260 IFA$="0"THENPRINT"■■ ■■";:GOTO14170
14270 IFA$<>"I"THEN14330
14280 FORI=0TO20:FORJ=0TO23
14290 IFPEEK(1024+I*40+J)=32THEN14310
14300 POKE1024+40*I+J,32:POKE55296+40*I+J,182:GOTO14320
14310 POKE1024+40*I+J,81:POKE55296+40*I+J,181
14320 NEXTJ,I:GOTO14170
14330 IFA$="R"THENGOTO13030
14340 IFA$<>"M"THEN14420
14350 FORI=0TO20:FORJ=0TO23:TT%(I,J)=0:NEXTJ,I
14360 FORI=0TO20:FORJ=0TO23
14370 IFPEEK(1024+40*I+J)= 32THENTT%(I,J)=1
14380 NEXT J,I
14390 PRINT"■";:FORI=0TO20:FORJ=23TO0STEP-1:IFTT%(I,J)=1THENPRINT"■■■";
14400 IFTT%(I,J)=0THENPRINT"■ ";
14410 NEXTJ:PRINT:NEXTI:PRINT"■":GOTO14170
14420 IFA$<>"T"THEN 14500
14430 FORI=0TO20:FORJ=0TO23:TT%(I,J)=0:NEXTJ,I
14440 FORI=0TO20:FORJ=0TO20
14450 IFPEEK(1024+40*I+J)=81THENTT%(20-J,20-I)=1
14460 NEXTJ,I
14470 PRINT"■";:FORI=0TO20:FORJ=23TO0STEP-1:IFTT%(I,J)=1THENPRINT"■■■";
14480 IFTT%(I,J)=0THENPRINT"■ ■";
14490 NEXTJ:PRINT:NEXTI:PRINT"■":GOTO14170
14500 IFA$<>"P"THEN14560
14510 FORI=0TO20:FORJ=0TO2:TT%(I,J)=0:NEXTJ,I
14520 FORI=0TO20:FORJ=0TO2:FORK=0TO7
14530 IFPEEK(1024+40*I+J*8+K)=81THENTT%(I,J)=TT%(I,J)OR2↑(7-K)
14540 NEXTK,J,I
14550 FORI=0TO20:FORJ=0TO2:POKEFNS(SN)+I*3+J,TT%(I,J):NEXTJ,I:GOTO13030
14560 IFA$<>"S"THEN14620
14570 INPUT"■■■■■■■■■■■■■■■■■■■■■■■■■■■HOW MANY SPRITES TO BE SAVED:";NN
14580 IFNN<1ORNN>32THEN14570
14590 OPEN1,1,1,"SPRITES":PRINT#1,NN
14600 FORI=0TONN*64-1:T%=PEEK(SS+I):PRINT#1,T%:NEXT
14610 CLOSE1:PRINT"■"
14620 IFA$<>"L"THEN14660
14630 OPEN1,1,0,"SPRITES":INPUT#1,NN
14640 FORI=0TO64*NN-1:INPUT#1,T:POKESP+I,T:NEXT
14650 CLOSE1:PRINT"■"
14660 IFA$<>"E"THEN14680
14670 POKE53269,0:POKE43,1:POKE44,8:POKE2048,0:CLR:END
14680 IFA$<>"X"THEN14740
14690 INPUT"■■■■■■■■■■■■■■■■■■■■■■■■■■NUMBER TO EXCHANGE WITH:";S2
14700 IFS2<0ORS2>31THEN14690
14710 FORI=FNS(SN)TOFNS(SN)+62:T1=PEEK(I):POKEI,PEEK(I-FNS(SN)+FNS(S2))
14720 POKEI-FNS(SN)+FNS(S2),T1:NEXT
14730 GOTO13000
14740 IFA$<>"C"THEN14840
14750 IFPEEK(53276)AND1=1THENPOKE53276,0:GOTO14170
14760 INPUT"■■■■■■■■■■■■■■■■■■■■■■■■■■INPUT COLOR FOR 01 (0-15):";C1
14770 IFC1<0ORC1>15THENPRINT"■                                ":GOTO14760
14780 INPUT"■■■■■■■■■■■■■■■■■■■■■■■■■■INPUT COLOR FOR 10 (0-15):";C2
14790 IFC2<0ORC2>15THENPRINT"■                                ":GOTO 14780
14800 INPUT"■■■■■■■■■■■■■■■■■■■■■■■■■■INPUT COLOR FOR 11 (0-15):";C3
14810 IFC3<0ORC3>15THENPRINT"■                                ":GOTO 14800
14820 POKE53285,(PEEK(53285)AND240)ORC1:POKE53287,(PEEK(53287)AND240)ORC3
14830 POKE53286,(PEEK(53286)AND240)ORC2:POKE53276,1:PRINT"■":GOTO14170
14840 IFA$<>"■"THEN14870
14850 POKE53277,255
14860 GOTO14170
14870 IFA$<>"■"THEN14900
14880 POKE53271,255
14890 GOTO14170
14900 IFA$<>"■"THEN14930
14910 POKE53277,0
14920 GOTO14170
14930 IFA$<>"■"THEN14950
14940 POKE53271,0
14950 GOTO14170
```