Il Vic II e i suoi «segreti» o meglio i suoi limiti

Alcuni programmi d'esempio

di Nicola Chiminelli - Montebelluna (TV)

Tutti coloro che hanno studiato un po' di LM hanno imparato a pasticciare con gli interrupt del computer. Infatti il cbm64, benché possieda un sistema operativo molto scarno, concede all'operatore smanettone di inserirsi nelle sue routine.

Per quanto riguarda il vettore di interrupt vi sono due locazioni, $0314 e $0315, rispettivamente byte basso e alto della locazione, a cui il computer ogni sessantesimo di secondo, abbandonato momentaneamente il compito che sta eseguendo, salta per eseguire una sua routine.

Se ad esempio modifichiamo quelle due locazioni, potremo sostituire una nostra routine a quella di interrupt di sistema. Provare per credere.

Programma 1

Ogni sessantesimo di secondo il colore del bordo verrà cambiato.

Conclusa la nostra routine è necessario tornare all'interrupt di sistema, allocato a $EA31, altrimenti la tastiera, l'orologio interno, il lampeggio del cursore non verranno più aggiornati, e i registri del 6502, precedentemente salvati nello stack, si perderanno.

Vi volevo però parlare di un altro interrupt, non più generato dal 6502, ma dal Vic II o 6566. Sì, proprio il microprocessore dedicato alla grafica (vero gioiello tra l'altro), che si trova dentro il nostro C64. Questo microprocessore può fornire quattro diverse fonti di interrupt. Mi spiego meglio. Se abilitiamo degli opportuni bit della locazione $D01A, abilitiamo anche le interruzioni del Vic II.

La locazione $D01A è così strutturata:

bit 0:IRQ di comparazione quadro (raster)
bit 1:IRQ di contatto sprite con fondo
bit 2:IRQ di contatto sprite con sprite
bit 3:IRQ triggerato per penna ottica.

I rimanenti bit non sono utilizzati, [a dire il vero sarebbe opportuno settare anche il bit sette, perché dovrebbe abilitare gli interrupt del Vic II. Per esperienza ho notato che non serve].

Se uno di questi bit vale uno, il Vic II si sente autorizzato, qualora si verificasse una delle condizioni sopra citate, a richiedere l'attenzione del 6502 e fargli eseguire un interrupt. Cioè fa interrompere il lavoro che il 6502 sta compiendo, e lo costringe a saltare alla locazione puntata da $0314 $0315 [jmp ($0314)].

La locazione $D019 è l'esatta copia di $D01A, solo che i suoi bit vengono attivati quando si verificano gli interrupt, e dunque ci fornisce le informazioni per usare vari interrupt. In più, il bit 7 è a uno se è avvenuto un qualunque interrupt del Vic II. Se la locazione $D019 non viene azzerata dopo il verificarsi di un interrupt non si genereranno più interrupt. Per fare ciò, non bisogna scrivere «zero», ma «uno» nei bit che si vogliono azzerare (proprio il contrario del solito). Benché ciò possa sembrare una complicazione, in realtà facilita notevolmente il compito del programma in quanto non è necessario sapere quali bit dell'interrupt del Vic II sono attivati. Infatti leggendo la locazione $D019 e scrivendo questo valore in $D019 si azzerano proprio i bit interessati.

Facciamo un esempio. Abilito il bit due della locazione $D01A. Ciò vuol dire che ogni volta che uno sprite tocca un altro sprite, il 6502 esegue un salto [jmp ($0314)]. Allora intercetto questa routine e al suo interno interrogo la locazione $D019 per sapere che tipo di interrupt si è verificato. Se $D019 contiene zero vuol dire che non è un interrupt del Vic II, ma del 6502, altrimenti se il bit 2 è alto, allora due sprite si sono effettivamente toccati. A questo punto, se voglio che si verifichino altri interrupt devo azzerare $D019.

Programma 2

Digitate il programma, posizionate degli sprite in modo che non siano già in collisione e date SYS49152. Ora provate a far scontrare gli sprite, lo schermo impazzirà di colori.

Ogni volta che ciò succederà la routine di interrupt sarà disattivata. Per fare ciò, basta rimettere a zero i bit di $D01A precedentemente attivati.

Programma 1

```
10 REM**** PROG 1 ****
20 REM BY NICOLA CHIMINELLI 0423/21393
100 PRINT"(CLR)"
110 FORT=49152TO49252+22
120 READA$:IFA$="*"THENPRINT"SYS 49152":END
130 GOSUB1000:POKET,C
140 NEXT:END
1000 A=ASC(LEFT$(A$,1))-48:IFA>9THENA=A-7
1010 B=ASC(RIGHT$(A$,1))-48:IFB>9THENB=B-7
1020 C=A*16+B:RETURN
10000 DATA78,A9,0D,8D,14,03,A9,C0,8D
10010 DATA15,03,58,60,EE,20,D0,4C,31
10020 DATAEA,00,00,*
```

Programma 2

```
1 REM*** PROG 2 ***
2 REM BY NICOLA CHIMINELLI 043/21393
10 PRINT"(CLR)"
20 FORT=49152TO49152+54
30 READA$:IFA$="*"THENPRINT"SYS 49152":END
40 GOSUB1000:POKET,C:NEXT:END
1000 A=ASC(LEFT$(A$,1))-48:IFA>9THENA=A-7
1010 B=ASC(RIGHT$(A$,1))-48:IFB>9THENB=B-7
1020 C=A*16+B:RETURN
10000 DATA78,A9,15,8D,14,03,A9,C0,8D
10010 DATA15,03,A9,04,8D,1A,D0,AD,1E
10020 DATAD0,58,60,AD,19,D0,8D,19,D0
10030 DATA29,04,F0,12,A9,00,8D,1A,D0
10040 DATAA0,FF,A2,FF,8E,20,D0,CA,D0
10050 DATAFA,88,D0,F5,4C,31,EA,00,*
```

Programma 3

```
5 REM PROG 3
6 REM BY CHIMINELLI NICOLA 0423/21393
10 PRINT"(CLR)"
20 FORT=49152TO49152+117
30 READA$
40 IFA$<>"*"THENGOSUB1000:POKET,C:NEXT:END
50 IFA$="*"THENPRINT"HIGH RES  AT(RVS) $6000 (OFF)"
60 PRINT"COLOR TAB AT(RVS) $5C00 (OFF)":PRINT"SCREE
N    AT(RVS) $0400 (OFF)"
70 PRINT"(RVS)(YEL)SYS 49152":END
1000 A=ASC(LEFT$(A$,1))-48:IFA>9THENA=A-7
1010 B=ASC(RIGHT$(A$,1))-48:IFB>9THENB=B-7
1020 C=A*16+B:RETURN
10000 DATA78,A9,7F,8D,0D,DC,AD,11,D0
10010 DATA29,7F,8D,11,D0,A9,99,8D,12
10020 DATAD0,A9,01,8D,1A,D0,A9,24,8D
10030 DATA14,03,A9,C0,8D,15,03,58,60
10040 DATAAD,19,D0,8D,19,D0,29,01,F0
10050 DATA41,AD,12,D0,C9,F9,90,1C,AD
10060 DATA00,DD,29,FC,09,02,8D,00,DD
10070 DATAA9,78,8D,18,D0,A9,3B,8D,11
10080 DATAD0,A9,99,8D,12,D0,4C,31,EA
10090 DATAA2,09,CA,D0,FD,AD,00,DD,29
10100 DATAFC,09,03,8D,00,DD,A9,15,8D
10110 DATA18,D0,A9,1B,8D,11,D0,A9,FA
10120 DATA8D,12,D0,4C,BC,FE,00,00,*
```

Programma 4

```
10 REM **** PROG 4 ****
20 REM CHIMINELLI NICOLA 0423/21393
30 REM VIA OSPEDALE 5 MONTEBELLUNA (TV)
100 PRINT"(CLR)"
110 FORT=49152TO49152+87
120 READA$:IFA$<>"*"THENGOSUB1000:POKET,C:NEXT:END
130 PRINT"SYS 49169":PRINT"(DOWN)COORDINATE SPRITE
DA $C000 A $C010":END
1000 A=ASC(LEFT$(A$,1))-48:IFA>9THENA=A-7
1010 B=ASC(RIGHT$(A$,1))-48:IFB>9THENB=B-7
1020 C=A*16+B:RETURN
10000 DATA00,00,00,00,00,00,00,00
10010 DATA00,00,00,00,00,00,00,78
10020 DATAAD,11,D0,29,7F,8D,11,D0,A9
10030 DATAFA,8D,12,D0,A9,01,8D,1A,D0
10040 DATAA9,3A,8D,14,03,A9,C0,8D,15
10050 DATA03,A2,11,A9,00,9D,FF,BF,CA
10060 DATAD0,FA,58,60,AD,19,D0,8D,19
10070 DATAD0,29,01,F0,0E,A2,11,BD,FF
10080 DATABF,9D,FF,CF,CA,D0,F7,4C,BC
10090 DATAFE,4C,31,EA,00,*
```

Chi ha osservato la tabella dei registri $19 e $1A (loc $D019 e $D01A), si sarà accorto della presenza di una parola «poco» conosciuta. Al bit zero risulta associato il termine RASTER. Chi segue MC dovrebbe già sapere cos'è. L'immagine che noi vediamo sul monitor, anche se ci appare ferma, viene continuamente rinnovata cioè un dispositivo chiamato pennello elettronico ridisegna ciclicamente tutti i punti che si trovano sullo schermo. Questo pennello è costituito da un fascio di elettroni (per questo è detto elettronico), che vengono sparati dall'interno del tubo catodico, e deviati in modo da sbattere contro il video (non temete, non si rompe) e eccitarne i fosfori. Il movimento di questo pennello è dall'alto al basso, da sinistra verso destra. Esso è molto veloce, tanto da disegnare 50 schermi al secondo, così il nostro occhio non si accorge di niente o quasi.

Il Vic II è in grado di fornirci l'esatta posizione verticale, cioè l'ordinata di dove si trova questo pennello. Ciò è possibile leggendo il registro $12 (loc $D012) che contiene i primi otto bit di questa posizione, il nono bit (MSB) si trova nel bit sette della locazione $D011. Con nove bit si possono avere 512 posizioni, ma solo 312 vengono utilizzate. Il raster parte dall'angolo in alto a sinistra e disegna 50 righe (dalla zero alla 49), che corrispondono alla parte superiore del bordo. Dalla 50 ($32) alla 250 ($fa) il pennello disegna lo schermo visibile, infine dalla 251 alla 311 la parte inferiore del bordo. Se noi leggessimo di continuo questo registro, sapremmo dove si trova il pennello, ma il 6502 sarebbe totalmente impegnato.

Ora torniamo ai registri $19 e $1A, il cui bit zero ci dà rispettivamente le informazioni sull'avvenuta e sull'attivazione dell'interrupt raster, permettendo un agevole controllo del pennello.

La locazione $D012 e il bit 7 di $D011 hanno una doppia funzione: in lettura ci danno la posizione del pennello, in scrittura predispongono un interrupt raster al valore che scriviamo. Esempio: attiviamo il bit zero di $D01A, scriviamo un valore (ad es. 151 circa metà schermo) in $D012, e ogni volta che il pennello raggiungerà quella posizione il computer salterà [jmp ($0314)] ad eseguire la routine di interrupt.

Penso che avrete una gran confusione in testa e non riusciate a capire a cosa possa servire tutto ciò. Bene, ora siamo arrivati al meglio. A cosa serve? Presto detto. Eseguire tutti gli aggiornamenti del video senza farlo sfarfallare; avere più di 8 sprite: esistono già un sacco di routine pronte che lo fanno. Vediamo un po' a grandi linee questi argomenti. Per avere più di 8 sprite bisogna cambiare i registri degli sprite quando il pennello si trova fuori dallo schermo visibile. Oppure si può dividere lo schermo a fasce e assegnare 8 sprite ad ogni striscia. Diversi set di caratteri, grafica a modo testo, si ottengono facilmente sullo stesso schermo, basta predisporre un interrupt raster e impostare i registri per l'alta risoluzione a metà schermo, per disabilitarli alla fine dello schermo. Semplice no?

Se avete già provato a fare qualcosa del genere, avrete notato un fastidioso tremolio dell'immagine nei punti in cui l'interrupt raster entra in azione. È facile capire che ciò succede poiché l'interrupt di sistema disturba la nostra routine di raster, dunque è fondamentale disabilitare l'interrupt del 6502, e far eseguire al nostro programma le chiamate all'interrupt di sistema una volta per quadro. Per disabilitare le richieste di IRQ bisogna mettere a zero il bit sette di $DC0D.

Facciamo ora un po' di esempi.

Programma 5

```
10 REM**** PROG 5 ****
20 REM BY NICOLA CHIMINELLI 0423/21393
30 REM VIA OSPEDALE 5 MONTEBELLUNA (TV)
110 FORT=49152TO49152+91
120 READA$:IFA$<>"*"THENGOSUB1000:POKET,C:NEXT:END
130 PRINT"SYS 49152":END
1000 A=ASC(LEFT$(A$,1))-48:IFA>9THENA=A-7
1010 B=ASC(RIGHT$(A$,1))-48:IFB>9THENB=B-7
1020 C=A*16+B:RETURN
10000 DATA78,A9,7F,8D,0D,DC,AD,11,D0
10010 DATA29,7F,8D,11,D0,A9,3A,8D,12
10020 DATAD0,A9,01,8D,1A,D0,A9,24,8D
10030 DATA14,03,A9,C0,8D,15,03,58,60
10040 DATAAD,19,D0,8D,19,D0,29,01,F0
10050 DATA22,A0,04,A2,0A,CA,D0,FD,A9
10060 DATA07,A2,01,20,53,C0,A9,06,A2
10070 DATA04,20,53,C0,88,D0,EF,A2,07
10080 DATACA,D0,FD,A9,00,8D,20,D0,4C
10090 DATA31,EA,8D,20,D0,CA,D0,FD,60
10100 DATA*
```

Programma 6

```
10 REM**** PROG 6 ****
20 REM BY NICOLA CHIMINELLI 0423/21393
30 REM VIA OSPEDALE 5 MONTEBELLUNA (TV)
90 REM IL PROG RESETTA IL C64
100 PRINT"(CLR)"
110 FORT=49152TO49152+59
120 READA$:IFA$<>"*"THENGOSUB1000:POKET,C:NEXT:END
130 PRINT"$C025   RIGA INFERIORE"
140 PRINT"$C02C   RIGA SUPERIORE"
150 PRINT"$C039   PER 9 DA IL NUMERO DI CICLI":PRIN
T"(DOWN)SYS 49152":END
1000 A=ASC(LEFT$(A$,1))-48:IFA>9THENA=A-7
1010 B=ASC(RIGHT$(A$,1))-48:IFB>9THENB=B-7
1020 C=A*16+B:RETURN
10000 DATA78,AD,11,D0,29,7F,8D,11,D0
10010 DATAA9,00,8D,12,D0,A9,01,8D,1A
10020 DATAD0,A9,21,8D,14,03,A9,C0,8D
10030 DATA15,03,58,4C,1E,C0,78,A2,00
10040 DATAA9,50,CD,12,D0,D0,FB,A9,5A
10050 DATAE8,CD,12,D0,D0,FA,8E,39,C0
10060 DATA4C,E2,FC,00,*
```

Testo e grafica contemporaneamente: programma 3

A volte, l'utilizzo degli sprite porta a degli spiacevoli tremolii .

Questo perché se uno sprite viene mosso dove il raster è già passato, non vedremo più il nostro sprite fino al prossimo quadro.

Allora si potrebbero scrivere tutti i dati degli sprite assieme, quando il raster è fuori dallo schermo.

Usare sprite senza sfarfallamenti: programma 4

In teoria non si può sapere la posizione orizzontale del pennello, perché la locazione $D012 e $D011 ci danno solo informazioni sull'ordinata. Però con dei ritardi calcolati si possono far eseguire più cose su una riga alta un pixel.

Dividere il bordo in due colori diversi: programma 5

Questo sistema però occupa totalmente il microprocessore, perché interroga di continuo la locazione $D012. Inoltre è impossibile calcolare con esattezza il punto di attacco del raster. In linea di massima dunque, la posizione «x» del pennello non è facilmente controllabile.

Una riga di schermo (alta un pixel) impiega circa da 59 a 63 cicli macchina per essere disegnata anche se questo tempo è suscettibile di variazioni. Varia per il bordo, per lo schermo e per la presenza di sprite. Se ci sono sprite attivati, il Vic II non utilizza solo la fase uno del clock per prelevare i dati degli sprite, ma ha bisogno anche della fase due, durante la quale il 6502 non può più operare.

Gli sprite dunque rallentano il microprocessore del C64, poiché viene dedicato più tempo al Vic II. Anche se questo fatto può sembrare trascurabile, va tenuto presente se vi sono molte chiamate raster per quadro, altrimenti ricompariranno i fastidiosi tremolii.

Calcolare il tempo disponibile durante una o più righe raster: programma 6

Uno dei maggiori problemi legati all'uso del raster, soprattutto se ci sono tanti interrupt per quadro, è quello delle sincronizzazioni.

Cioè è necessario calcolare con esattezza i tempi. Prima che si possa effettivamente modificare qualche registro del Vic II deve passare un certo tempo, speso dal 6502 per salvare i suoi registri e saltare indirettamente a $0314. Sfruttando il salto indiretto non si possono avere chiamate raster molto vicine (2 righe), perché le routine si sovrapporrebbero. Questo problema si può risolvere usando una tecnica ibrida tra le due descritte: Interrupt raster e controllo delle locazioni $D012 e $D011.

Se avete seguito tutti i consigli e vi è lo stesso un leggero tremolio, provate a velocizzare i vostri codici, oppure aggiungete qualche «NOP», o la sequenza LDX #$ritardo/dex/bne #$fD.

Tutto dovrebbe risolversi, a meno che non chiediate troppo al vostro C64.

Programma 7

Questo programma che vi propongo permette di cambiare tre volte il colore di sfondo ogni otto righe raster (le prime 3 righe di un colore, le 2 righe seguenti di un altro, e le ultime tre di un altro ancora), in modo da aumentare leggermente le possibilità cromatiche dei vostri disegni in modo multicolor. Sfortunatamente il Vic II si accorge delle modifiche di alcuni suoi registri solo ogni otto righe raster. Questo capita per i puntatori carattere, e per altre cose.

Dunque è impossibile (almeno per me) cambiare il colore o eseguire altre operazioni su un carattere quando questo è stato disegnato solo per metà.

Ancora qualche esempio...

I giochi che vengono continuamente realizzati per il C64, hanno la caratteristica di sfruttare sempre di più le sue doti grafiche. Una delle cose molto carine che si sono viste ultimamente sono gli sprite sul bordo. Normalmente gli sprite, quando escono dallo schermo, scompaiono gradualmente sotto il bordo. In quei giochini di cui vi parlavo, ciò non avviene, e gli sprite possono scorrazzare per quasi tutto lo schermo. Dopo un'infinità di prove, molta fortuna, e tante sbirciate ai codici dei programmi in questione, sono riuscito ad ottenere qualcosa di analogo.

In primo luogo mi sono accorto che è molto più semplice togliere le parti superiori e inferiori del bordo, piuttosto che quelle laterali: incominciamo dunque dalle cose più facili.

Se si azzera per un attimo il bit 3 di $D011 (che indica se lo schermo è

Bibliografia MC n. 39 pag. 131 / Rubrica Megagioco sempre su MC / Users Guide / I segreti del linguaggio macchina (jce) di Mark Greenshields.

attivato o meno) quando il pennello elettronico sta disegnando la prima riga del bordo inferiore ($fa) e poi si setta lo stesso bit, il bordo (miracolo?) scompare lasciando tanto spazio in più per i nostri sprite.

```
LDA #$13
sta $D011
lda $D012
bne $FE
lda #$1B
sta $D011
```

La spiegazione del fenomeno è quanto mai difficile, e credo sia causata da un momentaneo... impazzire del Vic II. Un'altra cosa ancora: spesso l'area di schermo strappata al bordo è piena di strane righine nere, che corrispondono al valore dell'ultima locazione del banco di 16K visto dal Vic II in quel momento. Ragion per cui questa locazione, che in condizioni normali è la $3FFF, dovrà contenere zero. Lo stesso problema si può risolvere, o meglio nascondere, colorando di nero tutto lo schermo.

Programma 7

```
100 PRINT"{CLR}"
110 FORT=49152TO49152+268
120 READA$:IFA$<>"*"THENGOSUB1000:POKET,C:NEXT:END
130 PRINT"TAB COLORI A $C0B2":PRINT"{DOWN}SYS 49152
":END
1000 A=ASC(LEFT$(A$,1))-48:IFA>9THENA=A-7
1010 B=ASC(RIGHT$(A$,1))-48:IFB>9THENB=B-7
1020 C=A*16+B:RETURN
10000 DATA78,A9,7F,8D,0D,DC,A9,01,8D
10010 DATA1A,D0,A9,4A,85,02,AD,5A,C0
10020 DATA8D,12,D0,AD,11,D0,29,7F,8D
10030 DATA11,D0,A9,29,8D,14,03,A9,C0
10040 DATA8D,15,03,58,60,AD,19,D0,8D
10050 DATA19,D0,29,01,F0,1E,C6,02,10
10060 DATA04,A9,4A,85,02,A6,02,BD,B2
10070 DATAC0,8D,21,D0,BD,5A,C0,8D,12
10080 DATAD0,A0,07,88,D0,FD,8A,F0,06
10090 DATA68,A8,68,AA,68,40,4C,31,EA
10100 DATA31,F7,F4,F2,EF,EC,EA,E7,E4
10110 DATAE2,DF,DC,DA,D7,D4,D2,CF,CC
10120 DATACA,C7,C4,C2,BF,BC,BA,B7,B4
10130 DATAB2,AF,AC,AA,A7,A4,A2,9F,9C
10140 DATA9A,97,94,92,8F,8C,8A,87,84
10150 DATA82,7F,7C,7A,77,74,72,6F,6C
10160 DATA6A,67,64,62,5F,5C,5A,57,54
10170 DATA52,4F,4C,4A,47,44,42,3F,3C
10180 DATA3A,37,34,00,00,00,00,00,00
10190 DATA00,00,00,00,00,00,00,01,0F
10200 DATA0C,0B,00,01,0F,0C,0B,00,01
10210 DATA0F,0C,0B,00,01,0F,0C,0B,00
10220 DATA01,0F,0C,0B,00,01,0F,0C,0B
10230 DATA00,01,0F,0C,0B,00,01,0F,0C
10240 DATA0B,00,01,0F,0C,0B,00,01,0F
10250 DATA0C,0B,00,0F,0C,0B,00,01,0F
10260 DATA0C,0B,00,01,0F,0C,0B,00,01
10270 DATA0F,0C,0B,00,01,0F,0C,0B,00
10280 DATA01,0F,0C,0B,00,01,*
```

Programma 8

```
10 REM**** PROG 8 ****
20 REM BY NICOLA CHIMINELLI 0423/21393
30 REM VIA OSPEDALE 5 MONTEBELLUNA (TV)
100 POKE2040,13
110 FORT=832TO832+62:POKET,255:NEXT
120 V=53248:POKEV+21,1:POKEV+39,1
130 POKEV,60:POKEV+1,254:POKE16383,0
900 PRINT"{CLR}"
910 FORT=49152TO49152+59
920 READA$:IFA$<>"*"THENGOSUB1000:POKET,C:NEXT:END
930 PRINT"SYS 49152":END
1000 A=ASC(LEFT$(A$,1))-48:IFA>9THENA=A-7
1010 B=ASC(RIGHT$(A$,1))-48:IFB>9THENB=B-7
1020 C=A*16+B:RETURN
10000 DATA78,A9,7F,8D,0D,DC,A9,1B,8D
10010 DATA11,D0,A9,FA,8D,12,D0,A9,01
10020 DATA8D,1A,D0,A9,21,8D,14,03,A9
10030 DATAC0,8D,15,03,58,60,AD,19,D0
10040 DATA8D,19,D0,A9,13,8D,11,D0,AD
10050 DATA12,D0,D0,FB,A9,1B,8D,11,D0
10060 DATA4C,31,EA,00,*
```

Programma 9

```
10 REM**** PROG 9 ****
20 REM BY NICOLA CHIMINELLI 0423/21393
100 FORT=832TO832+62:POKET,255:NEXT:V=53248
110 FORT=0TO7:POKE2040+T,13:POKEV+39+T,7:NEXT
120 FORX=0TO240STEP40:POKEV+N,X:N=N+2:NEXT
125 POKEV+N,90:POKEV+16,128
130 POKEV+N,90:POKEV+16,128:POKE16383,0
900 FORT=49152TO49152+154
910 READA$:IFA$<>"*"THENGOSUB1000:POKET,C:NEXT:END
920 PRINT"{CLR}$02 ORDINATA DEGLI 8 SPRITE"
930 PRINT"{DOWN}SYS 49152":END
1000 A=ASC(LEFT$(A$,1))-48:IFA>9THENA=A-7
1010 B=ASC(RIGHT$(A$,1))-48:IFB>9THENB=B-7
1020 C=A*16+B:RETURN
10000 DATA78,A9,7F,8D,0D,DC,A9,1B,8D
10010 DATA11,D0,A9,00,8D,12,D0,A9,30
10020 DATA85,02,A9,01,8D,1A,D0,A9,25
10030 DATA8D,14,03,A9,C0,8D,15,03,58
10040 DATA60,AD,19,D0,8D,19,D0,A2,0F
10050 DATAA5,02,9D,00,D0,CA,CA,10,F9
10060 DATAA9,FF,8D,15,D0,A5,02,8D,12
10070 DATAD0,A9,1B,8D,11,D0,A9,52,8D
10080 DATA14,03,4C,BC,FE,EA,EA,EA,EA
10090 DATAEA,AD,19,D0,8D,19,D0,D0,00
10100 DATAEA,EA,EA,EA,EA,A0,84,88,30
10110 DATAFD,A0,14,24,24,CE,16,D0,EE
10120 DATA16,D0,AE,12,D0,CA,8A,29,02
10130 DATA09,10,8D,11,D0,EA,EA,EA,EA
10140 DATAEA,88,10,E4,A9,1B,8D,11,D0
10150 DATAA2,03,CA,D0,FD,A9,25,8D,14
10160 DATA03,A9,00,8D,12,D0,4C,31,EA
10170 DATA*
```

Togliere il bordo sopra e sotto: programma 8

Sfruttando una tecnica simile si può far scomparire parzialmente il bordo laterale. Il programma che vi propongo, toglie per 20 righe (l'altezza di uno sprite non espanso) il bordo. Si può ripetere la stessa operazione più volte, a diverse altezze di schermo.

Togliere parte del bordo laterale: programma 9

La tecnica utilizzata non è proprio il massimo della funzionalità ed ha molti limiti. In primo luogo, nella fascia in cui scompare il bordo, bisogna posizionare tutti otto gli sprite, con ordinata uguale alla riga raster da cui il bordo viene tolto (per l'ascissa non c'è nessun problema).

Tutto ciò viene realizzato facendo dimenticare al Vic II dove si trova (in pratica imbrogliandolo). Così facendo però, oltre alla comparsa delle righine e di strani caratteri, l'area di schermo occupata dalla fascia priva di bordo, è inutilizzabile dai caratteri. Tutto ciò è sintomo della pazzia del Vic II, che per fortuna è incurabile. Provate a sperimentare, ne vedrete veramente delle belle. Buon divertimento! MC