Questo mese vi proponiamo una rubrica decisamente ben fornita, con tre programmi. Il primo, in linguaggio macchina, abilita un'interfaccia parallela — sia per l'ingresso che per l'uscita — sui piedini della porta utente; il secondo è un semplice giochino in Basic, che usa caratteri definiti dall'utente e suoni, la cui particolarità sta nell'essere per due giocatori (sempre meglio scontrarsi con avversari umani); per ultima trovate una semplice routine per la gestione dei joystick: ad una versione espansa, scritta in Basic, affianchiamo alcune righe in assembler che restituiscono il controllo solo quando viene usato almeno uno dei due controllori, il che rende immediata la risposta al movimento del joystick.

Elenco cassette per il CBM 64

Presso la redazione sono disponibili le cassette relative ad alcuni dei programmi pubblicati nella rubrica di software per il CBM 64. Per l'ordinazione inviare l'importo (a mezzo assegno, c/c o vaglia postale) a Technimedia srl, Via Valsolda 135, 00141 Roma.

codice	programma	MC n.	lire
C64/01	Briscola	25	17.000
C64/02	Serpentone	29	17.000
C64/03	Othello	29	17.000

UNA PORTA PARALLELA

Non staremo ad elencare qui i motivi per i quali è utile un'interfaccia parallela, possibilmente compatibile con lo standard Centronics: semplicemente vi proponiamo un programma in LM per averla a disposizione sul 64 (a breve scadenza verranno informazioni dettagliate sulla RS232): l'interfaccia parallela è più semplice di quella seriale RS232 perché i livelli di uscita dei computer Commodore, 5V nominali (di fatto tra 3.5 e 4) sono compatibili con le specifiche della casa americana Centronics, madre dell'omonima interfaccia poi divenuta uno standard di fatto, mentre la RS232 richiede una conversione, e quindi necessita di un circuitino.

I contatti vivi del cavo da realizzare sono mostrati nella figura 1, dedotta — come buona parte delle informazioni utilizzate in questo articolo — da "Using the Commodore 64 Parallel Interface", pubblicato su Micro n. 69, febbraio 1984. Il programma del listato 1 realizza un contatto bidirezionale: l'uscita è abilitata con una
SYS 32768
e disabilitata con una
SYS 32771
mentre si riceve con
SYS 32774
e si smette o usando la tastiera, oppure con
SYS 32777.

L'uscita, in pratica, preleva quello che deve andare al video per mandarlo anche alla stampante: è quindi ottimo anche per listati in assembler, tanto che quelli che vi proponiamo sono realizzati proprio con questo sistema. È evidente che i listati sono... pericolosi, dato che le stampanti normali non riconoscono il set di caratteri della Commodore: la cosa può essere ovviata sostituendo ai caratteri speciali tra virgolette i codici equivalenti (con il CHR$), e i cursori con il TAB, come potete osservare sul listato di TRON 64, pubblicato su queste stesse pagine. Il marchingegno funziona anche con i wordprocessor, cui contiamo di dedicare un articolo a parte.

Non ci dilunghiamo sull'interfaccia Centronics e sulla sua compatibilità con quella che vi proponiamo: state attenti solo al fatto che molte stampanti parallele hanno un filo di ritorno per ogni filo del segnale, mentre la nostra ha un unico cavo per il ritorno comune, chiamato Signal Ground (come peraltro previsto dallo standard di fatto). In quel caso il collegamento non funziona, almeno per ora: in seguito vedremo ....

La routine, lunga 200 byte, è stata da noi allocata a partire da $8000, cioè dalla decimale 32768, in modo da avere sufficiente spazio per altri programmi in LM da porre verso il top della memoria, senza dover rinunciare né ai 4K destinati alle cartucce (e solitamente agli assemblatori), né alla possibilità di lavorare in Basic (uscendo dall'assemblatore) magari per caricare un directory o per fare dei conti matematici, delle conversioni, etc ... Chi prevede di lavorare con programmi lunghi farà bene ad informare il computer, abbassando da 192 a 160 i contenuti del puntatore al tetto della memoria (locaz. 51) e di quello di partenza delle stringhe (locaz. 55) che va dal tetto a scendere, con delle
POKE 51,160: POKE 55,160

Chi non avesse un assemblatore può comunque fruire della routine, usando il caricatore Basic mostrato nel listato 2 (che andrà poi in esecuzione con un RUN 400, come vedrete); la verifica di correttezza può essere fatta sia direttamente che tramite un programmino di debug. Noi abbiamo scelto quest'ultima strada, e vi proponiamo il programma del listato 3 che va digitato insieme al caricatore ed abilitato con un semplice RUN: la sua funzione è mostrare i numeri dei DATA uno per uno, premendo un tasto, e lasciando uno spazio ogni 8, in relazione alla disposizione degli stessi nelle righe di DATA: volendo correggere qualcosa si può fermare l'esecuzione, correggere il caricatore e riprendere la verifica, poiché il programmino chiede l'ultimo numero di linea cui si era arrivati, per ripartire da quello immediatamente successivo.

La routine è facilmente modificabile per essere spostata ovunque in memoria: consigliamo un multiplo di 4096, perché con questa scelta basta portare al valore
(nuova locazione) / 256
tutti i 128 decimali (80 esadecimale) tranne quello della locazione 32963 (esad. 80C2).

USCITA PARALLELA SUL 64 - I CONTATTI

PIN	SEGNALE	FUNZIONE
A	MASSA	COMUNE
B	FLAG2/	ACKNOWLEDGE/
C	PB0	DATO-BIT0
D	PB1	" 1
E	PB2	" 2
F	PB3	" 3
H	PB4	" 4
J	PB5	" 5
K	PB6	" 6
L	PB7	" 7
M	PA2/	NON COLLEGATO
N	MASSA	COMUNE
8	PC2/	STROBE/

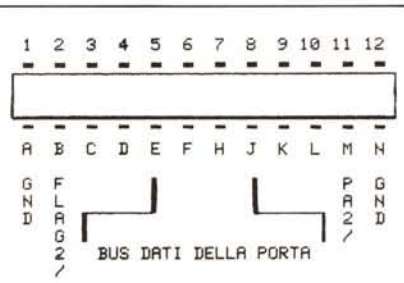


Figura 1

```
10 GOTO 100
20 **********************************
21 **  QUESTO SERVE A FAR LISTARE **
22 **  I NUMERI DEI PROGRAMMI IN  **
23 **     LINGUAGGIO  MACCHINA     **
24 **********************************
25 :
100 I=0:PRINTCHR$(147)
110 INPUT"CHE DATA";X
112 A=8*4096+(X-500)/10*8
120 PRINTA+I,PEEK(A+I):I=I+1
130 GETA$:IFA$=""THEN130
135 IF INT(I/8)=I/8THENPRINT
140 GOTO 120
200 :
210 REM ******************************
211 REM ** QUESTA GENERA LE LINEE DI**
212 REM ** DATA A PARTIRE DAL LING. **
213 REM ** MACCHINA ASSEMBLATO DALLA**
214 REM ** LOCAZIONE 32768 ($8000). **
215 REM ******************************
216 :
300 T=32768:POKE704,0
320 FORS=0TO7:A=PEEK(T+S):A$=RIGHT$(STR$(A),3)+","
340 B$=B$+A$:NEXT
350 C=500+10*(T-32768)/8
360 C$=STR$(C)+"D♠"+B$
365 PRINT"■■■■"C$"■":B$=""
367 POKE704,PEEK(704)+1
369 POKE631,32
370 POKE632,13:POKE633,71:POKE634,111:POKE635,51
371 POKE636,56:POKE637,48:POKE638,13
372 POKE198,8:END
380 T=32768+PEEK(704)*8:IFT>32975THEN400
390 GOTO320
399 STOP
400 :
401 REM ******************************
402 REM ** QUESTA SERVE A CARICARE **
403 REM ** IL PROGRAMMA DI PORTA    **
404 REM ** PARALLELA CHE SI TROVA   **
405 REM ** NEI DATA.                **
406 REM ** CHI USA UN ASSEMBLATORE **
407 REM ** GENERI I DATA CON LE     **
408 REM ** LINEE 300-400            **
409 REM ******************************
410 :
415 Y=0
420 READ A: IF A=256 THEN STOP
430 POKE 32768+Y,A: Y=Y+1: GOTO 420
440 :
500 DATA 76, 12,128, 76, 40,128, 76,108,
510 DATA128, 76,136,128,173, 38, 3,141,
520 DATA 93,128,173, 39, 3,141, 94,128,
530 DATA169, 53,141, 38, 3,169,128,141,
540 DATA 39, 3,169,255,141, 3,221, 96,
550 DATA173, 93,128,141, 38, 3,173, 94,
560 DATA128,141, 39, 3, 96, 72,169, 2,
570 DATA 44, 24,208,240, 27,104, 48, 13,
580 DATA201, 64, 48, 19,201, 95, 16, 15,
590 DATA 24,105, 32,208, 10,201,192, 48,
600 DATA 6,201,223, 16, 2, 41,127, 72,
610 DATA104, 32, 95,128, 76, 0, 0, 72,
620 DATA169, 16, 44, 13,221,240,251,104,
630 DATA141, 1,221, 96,173, 36, 3,141,
640 DATA149,128,173, 37, 3,141,150,128,
650 DATA169,151,141, 36, 3,169,128,141,
660 DATA 37, 3,169, 0,141, 3,221, 96,
670 DATA173,149,128,141, 36, 3,173,150,
680 DATA128,141, 37, 3, 76, 0, 0,173,
690 DATA198, 0,208,236,169, 16, 44, 13,
700 DATA221,240,244,173, 1,221, 72,169,
710 DATA 2, 44, 24,208,208, 7,104,201,
720 DATA 96, 48, 21, 16, 13,104,201, 64,
730 DATA 48, 14,201, 95, 16, 4, 9, 32,
740 DATA208, 6,201,128, 16, 2, 41, 95,
750 DATA108, 38, 3,256, 0, 0, 0,0
```

Listato 2

```
C*
     PC  SR AC XR YR SP
.;  805B 39 31 35 32 FF
.
.,  8000 4C 0C 80 JMP $800C
.,  8003 4C 28 80 JMP $8028
.,  8006 4C 6C 80 JMP $806C
.,  8009 4C 88 80 JMP $8088
.,  800C AD 26 03 LDA $0326
.,  800F 8D 5D 80 STA $805D
.,  8012 AD 27 03 LDA $0327
.,  8015 8D 5E 80 STA $805E
.,  8018 A9 35    LDA #$35
.,  801A 8D 26 03 STA $0326
.,  801D A9 80    LDA #$80
.,  801F 8D 27 03 STA $0327
.,  8022 A9 FF    LDA #$FF
.,  8024 8D 03 DD STA $DD03
.,  8027 60       RTS
.,  8028 AD 5D 80 LDA $805D
.,  802B 8D 26 03 STA $0326
.,  802E AD 5E 80 LDA $805E
.,  8031 8D 27 03 STA $0327
.,  8034 60       RTS
.,  8035 48       PHA
.,  8036 A9 02    LDA #$02
.,  8038 2C 18 D0 BIT $D018
.,  803B F0 1B    BEQ $8058
.,  803D 68       PLA
.,  803E 30 0D    BMI $804D
.,  8040 C9 40    CMP #$40
.,  8042 30 13    BMI $8057
.,  8044 C9 5F    CMP #$5F
.,  8046 10 0F    BPL $8057
.,  8048 18       CLC
.,  8049 69 20    ADC #$20
.,  804B D0 0A    BNE $8057
.,  804D C9 C0    CMP #$C0
.,  804F 30 06    BMI $8057
.,  8051 C9 DF    CMP #$DF
.,  8053 10 02    BPL $8057
.,  8055 29 7F    AND #$7F
.,  8057 48       PHA
.,  8058 68       PLA
.,  8059 20 5F 80 JSR $805F
.,  805C 4C CA F1 JMP $F1CA
.,  805F 48       PHA
.,  8060 A9 10    LDA #$10
.,  8062 2C 0D DD BIT $DD0D
.,  8065 F0 FB    BEQ $8062
.,  8067 68       PLA
.,  8068 8D 01 DD STA $DD01
.,  806B 60       RTS
.,  806C AD 24 03 LDA $0324
.,  806F 8D 95 80 STA $8095
.,  8072 AD 25 03 LDA $0325
.,  8075 8D 96 80 STA $8096
.,  8078 A9 97    LDA #$97
.,  807A 8D 24 03 STA $0324
.,  807D A9 80    LDA #$80
.,  807F 8D 25 03 STA $0325
.,  8082 A9 00    LDA #$00
.,  8084 8D 03 DD STA $DD03
.,  8087 60       RTS
.,  8088 AD 95 80 LDA $8095
.,  808B 8D 24 03 STA $0324
.,  808E AD 96 80 LDA $8096
.,  8091 8D 25 03 STA $0325
.,  8094 4C 00 00 JMP $0000
.,  8097 AD C6 00 LDA $00C6
.,  809A D0 EC    BNE $8088
.,  809C A9 10    LDA #$10
.,  809E 2C 0D DD BIT $DD0D
.,  80A1 F0 F4    BEQ $8097
.,  80A3 AD 01 DD LDA $DD01
.,  80A6 48       PHA
.,  80A7 A9 02    LDA #$02
.,  80A9 2C 18 D0 BIT $D018
.,  80AC D0 07    BNE $80B5
.,  80AE 68       PLA
.,  80AF C9 60    CMP #$60
.,  80B1 30 15    BMI $80C8
.,  80B3 10 0D    BPL $80C2
.,  80B5 68       PLA
.,  80B6 C9 40    CMP #$40
.,  80B8 30 0E    BMI $80C8
.,  80BA C9 5F    CMP #$5F
.,  80BC 10 04    BPL $80C2
.,  80BE 09 20    ORA #$20
.,  80C0 D0 06    BNE $80C8
.,  80C2 C9 80    CMP #$80
.,  80C4 10 02    BPL $80C8
.,  80C6 29 5F    AND #$5F
.,  80C8 6C 26 03 JMP ($0326)
.,  80CB 00       BRK
.,  80CC 00       BRK
```

Listato 1

```
10 GOTO 100
20 **********************************
21 **  QUESTO SERVE A FAR LISTARE **
22 **  I NUMERI DEI PROGRAMMI IN  **
23 **     LINGUAGGIO  MACCHINA     **
24 **********************************
25 :
100 I=0
110 INPUT"CHE DATA";X
112 A=8*4096+(X-10000)/10*8
120 PRINTA+I,PEEK(A+I):I=I+1
130 GETA$:IFA$=""THEN130
135 IF INT(I/8)=I/8THENPRINT
140 GOTO 120
```

Listato 3

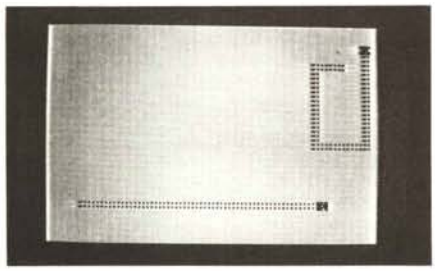
```
CHE DATA?
 32768 76
 32769 12
 32770 128
 32771 76
 32772 40
 32773 128
 32774 76
 32775 108
```

Esempio di output

TRON/64

di Fabrizio Giussani, Roma

Per chi ha visto il film, o comunque ne conosce la trama, non ci sono spiegazioni da fare; per tutti gli altri valgono le seguenti, poche righe. Il gioco è per due persone; non si può giocare contro il computer. Si è



alla guida di un'automobilina che dietro di sé lascia una traccia solida: il giocatore, che controlla la direzione con il joystick, deve evitare il bordo dello schermo e le tracce lasciate sia dal proprio veicolo che da quello avversario. Chi sbatte per primo perde la manche; per ogni giocatore si stabilisce un punteggio proporzionale al cammino percorso, e il vincitore ha diritto ad un bonus (che si fissa all'inizio del gioco stesso, come pure il punteggio massimo da raggiungere per vincere).

Come in tutti i giochi ad avversario umano, il divertimento dipende dalla bravura dei contendenti; e non crediate sia tanto facile evitare i vari ostacoli! Se non ci credete, provateci ...

USIAMO IL JOYSTICK (IN BASIC E IN LM)

La routine in LM è molto semplice: serve solamente ad attendere che uno qualunque dei due joystick sia usato, sia per la direzione che per il fuoco, nel qual caso restituisce il controllo al programma chiamante. Il vantaggio nell'uso del LM è l'estrema prontezza con cui il sistema risponde.

Anche in questo listato si nota la differenza tra i contenuti a riposo delle locazioni dei due jostick.

Il programma Basic è un po' più complesso, e fa vedere come si può controllare un gioco tramite entrambi i joystick. Vediamone il comportamento.

La pagina video è composta da 25 righe di 40 caratteri l'una: ogni carattere occupa 1 byte in memoria, per cui l'intera schermata impiega $40 \times 25 = 1000$ byte. Nella mappa di memoria del computer il video parte dalla locazione successiva a 1023, e arriva a $1023 + 1000 = 2023$: in queste locazioni vanno messi i codici dei caratteri secondo la tabella dei "codici di schermo" (sul manuale in inglese sta a pag. 132). Nei Commodore per ogni carattere mostrato sullo schermo bisogna specificare anche il colore in cui lo si vuol mostrare, e questo va specificato mettendo un valore da 0 a 15 nei byte da 55296 a 56295 — la mappa del colore. Attenzione: i contenuti di queste locazioni sono sempre forzati ad un valore da 240 in su, per cui ad una

```
POKE 55296, 3
```

corrisponderà il contenuto $240 + 3 = 243$, come rivelato da una

```
PRINT PEEK (55296).
```

Per ogni carattere da visualizzare sullo schermo, quindi, bisogna specificare due valori: uno sulla mappa video, e uno sulla mappa colore. Per avere nella locazione L un carattere di codice C e colore R bisogna fare come segue:

1) mettere in $1023 + L$ il valore C;
2) mettere in $55296 + L$ il valore R;

questo sistema necessita di tenere due conteggi, quello della memoria di schermo e quello della memoria di colore. Poiché queste stanno in posti fissi, è più comodo tenere un solo conteggio per entrambi, ovvero - posto $M = 1023 + L$.

1) mettere in M il valore C;
2) mettere in $54272 + M$ il valore R;

avendo notato che $54272 = 55296 - 1024$.

È immediata la conversione in Basic delle due istruzioni 1 e 2:

```
10 POKE M, C
20 POKE 54272+M, R
```

con M compreso tra 1024 e 2023; C tra 0 e 255; R tra 0 e 15.

Ora che sappiamo come gestire lo schermo, vediamo di gestire l'animazione.

L'animazione da Basic consiste in pochi passi:

a) mostrare un carattere;
b) calcolarne la nuova posizione;
c) cancellare il carattere dalla vecchia posizione;
d) ritornare al punto a).

Il punto a) è risolto dalle due linee 10-20 viste in precedenza. Il punto b) dipende dall'applicazione che se ne vuol fare, e per il joystick possiamo notare le seguenti cose (fig. 2):

— andare a destra corrisponde ad aggiungere 1 alla posizione in cui siamo;

```
1 GOTO 15
2 PRINT"*******************************"
3 PRINT"* ------------------------- *"
4 PRINT"* -TRON- GIOCO PER IL C64  - *"
5 PRINT"* - DI   FABRIZIO GIUSSANI - *"
6 PRINT"* ------------------------- *"
7 PRINT"*******************************"
8 :
15 PRINTCHR$(147)CHR$(5)TAB(255)"ATTENDERE PREGO":STOP
20 POKE 54296,15
25 PRINTCHR$(142)
30 POKE52,48:POKE56,48:CLR
40 POKE56334,PEEK(56334)AND254
50 POKE1,PEEK(1)AND251
60 FORI=0TO511:POKEI+12288,PEEK(I+53248):NEXT
70 POKE1,PEEK(1)OR4
80 POKE56334,PEEK(56334)OR1
85 POKE53272,(PEEK(53272)AND240)+12
90 FORI=12504TO12543:READA:POKEI,A:NEXT
100 DATA 231,66,250,143,143,250,66,231
110 DATA 189,231,165,36,60,153,255,153
120 DATA 231,66,95,241,241,95,66,231
130 DATA 153,255,153,60,36,165,231,189
140 DATA 0,102,102,0,0,102,102,0
320 PRINT:PRINT"                    "
330 POKE53281,7:PRINTCHR$(147)CHR$(144)
335 PRINT:PRINT"          ****  T R O N  ****"
336 PRINT:PRINT:PRINT
340 INPUT" PUNTEGGIO  MASSIMO ";PMAX
345 PRINT:INPUT" PUNTI PREMIO MANCHE ";PE
350 MAN=0:PT(0)=0:PT(1)=0
360 P=0:R(0)=4:C(0)=4:R(1)=20:C(1)=35:SR(0)=0:SR(1)=0:SC(0)=+1:SC(1)=-1
370 Z(0)=27:Z(1)=29
400 PRINTCHR$(147)
500 FORI=0TO1:P=P+1:D(I)=PEEK(56321-I)
600 POKE 1024+C(I)+40*R(I),31:POKE1024+C(I)+40*R(I)+54272,I*4+10
700 POKE54276,0:POKE54277,0:POKE54278,0
1000 IFD(I)=254-I*128THENSR(I)=-1:SC(I)=0:Z(I)=30:GOTO1040
1010 IFD(I)=253-I*128THENSR(I)=+1:SC(I)=0:Z(I)=28:GOTO1040
1020 IFD(I)=251-I*128THENSC(I)=-1:SR(I)=0:Z(I)=29:GOTO1040
1030 IFD(I)=247-I*128THENSC(I)=+1:SR(I)=0:Z(I)=27:GOTO1040
1040 R(I)=R(I)+SR(I):C(I)=C(I)+SC(I)
1500 PO(I)=PEEK(1024+C(I)+40*R(I))
2000 IFPO(I)<>32OR(R(I)=25)OR(C(I)=40OR(R(I)=-1)ORC(I)=-1)THEN3000
2500 POKE1024+C(I)+40*R(I),Z(I):POKE1024+C(I)+40*R(I)+54272,I*4+2
2510 POKE54277,5:POKE54278,248:POKE54273,I*6+28:POKE54272,214-139*I:POKE54276,17
2520 NEXT:GOTO500
3000 MAN=MAN+1:PT(I)=PT(I)+P+PE
3030 IFPT(I)>=PMAXTHEN4000
3040 PRINTCHR$(147):PRINT" MANCHE N.";MAN
3060 PRINTTAB(80)" IL GIOCATORE A SINISTRA VINCE";PT(1);"PUNTI"
3070 PRINT" IL GIOCATORE A DESTRA VINCE  ";PT(0);"PUNTI"
3080 FORT=0TO2000:NEXT:PRINTCHR$(147):GOTO360
4000 IFI=1THEN A$="SINISTRA"
4010 IFI=0THEN A$="DESTRA"
4100 PRINTCHR$(147):PRINT" VINCE IL GIOC. A ";A$;" CON";PT(I);"PUNTI"
4110 FORT=0TO2000:NEXT:FORN=0TO100:GETA$:NEXT:A$="":GOTO330
```

```
1 REM ************************
2 REM ** QUESTO CARICA DA 704 **
3 REM **  LA ROUTINE DI SCAN  **
4 REM ************************
5 :
20 DATA 234,169,255,205,  1,220,208,  8
21 DATA 169,127,205,  0,220,240,241,234
22 DATA  96
23 FORT=704TO720:READS:POKET,S:NEXT

02 C0 EA       NOP
02 C1 A9 FF    LDA #$FF
02 C3 CD 01 DC CMP $DC01
02 C6 D0 08    BNE $02D0
02 C8 A9 7F    LDA #$7F
02 CA CD 00 DC CMP $DC00
02 CD F0 F1    BEQ $02C0
02 CF EA       NOP
02 D0 60       RTS
```

```
100 REM ******************************
101 REM ***   USIAMO I JOYSTICK    ***
102 REM ******************************
103 :
108 REM * MEMORIZZIAMO GLI SPOSTAMENTI
109 :
110 DATA0,40,-40,0,-1,-41,39,0,1,-39,41
120 FORT=0TO10:READX(T):NEXT
121 :
125 REM * IN B INIZIA LA MAPPA VIDEO
126 REM * IN C+B INIZIA LA MAPPA COLORE
127 :
130 B=1024:C=54272
133 :
134 REM * FNJ(I) SCEGLIE UN VALORE
135 REM * DA 0 A 10 COME IN TAB. 1;
136 :
140 DEFFNJ(I)=15-PEEK(56320+I)AND15
150 FORI=0TO1:A=X(FNJ(I))
152 POKEB,32:POKEC+B,6
154 B=B+A
155 :
156 REM * VERIFICHIAMO LA POSIZIONE
157 REM * PER RESTARE NELLO SCHERMO
158 :
160 IFB<1024THENB=1024
170 IFB>2023THENB=2023
190 POKEB,81:POKEC+B,1
192 IF(PEEK(56320+I)AND16)=16THEN200
195 PRINT"FUOCO!"
200 NEXT:GOTO150
```

	56320	56321	FNJ(I)	X(FNJ(I))
RIPOSO	127	255	0	0
SU	126	254	1	-40
GIU'	125	253	2	+40
--			3	0
SIN	123	251	4	- 1
SU+ SIN	122	250	5	-41
GI+ SIN	121	249	6	+39
--			7	0
DESTRA	119	247	8	+ 1
SU+ DES	118	246	9	-39
GI+ DES	117	245	10	+41
FUOCO	111	239	-	--

Figura 2

```
USO E FUNZIONE DEI BYTE 56320 E 56321
-------------------------------------

BIT       7    6    5    4    3    2    1    0

56320   PAD1 PAD2  -   FU   DE   SI   GI   SU
($DC00 : SELEZ. RIGHE   DELLA TASTIERA)

56321    -    -    -   FU   DE   SI   GI   SU
($DC01 : SELEZ.COLONNE DELLA TASTIERA)


LEGENDA:
-------- PAD = PADDLE  ;
         FU  = FUOCO   ;
         DE  = DESTRA  ;
         SI  = SINISTRA;
         GI  = GIU'    ;
         SU  = SU      .
```

Figura 3

— andare a sinistra equivale a sottrarre 1;

— salire di una riga equivale ad arretrare del numero di caratteri di una riga (nel 64 sono 40);

— scendere di una riga equivale ad àggiugere 40 alla posizione precedente.

Le posizioni intermedie si ottengono aggiungendo o sottraendo 1 al valore 40.

Il punto c) equivale a mettere nelle locazioni di quel momento uno spazio (codice 32) e il solito colore (codice 6). Nel programma che vi presentiamo valori numerici sono posti nel vettore X(T), letto in riga 120.

Le istruzioni da a) a d) diventano il seguente programma Basic:

```
10 POKE M,C:POKE 54272 + M,R
20 — calcolare la nuova posizione —
— e metterla in B — ................
..........................................................
50 POKE M,32:POKE 54272 + M,6
60 GOTO 10
```

Bisogna vedere come calcolare le nuove posizioni. Un joystick è composto di 5 interruttori: 4 per le direzioni cardinali (N,S,E,O) più 1 per il fuoco. È evidente che per avere anche le direzioni intermedie il joystick è fatto in modo tale che due interruttori contigui possono essere chiusi con un solo movimento, per cui ai 4 angoli corrispondono le direzioni NE, NO, SE, SO.

I 5 interruttori corrispondono a 5 bit per ogni controllore, allocati per il joy 1 nella cella di memoria 56320 ($DC00) e per il joy 2 nella successiva 56321 ($DC01), come mostrato in figura 3. Queste locazioni vengono usate dal sistema operativo anche per gestire la tastiera, per cui non meravigliatevi se il joystick inserito interferisce con quella, oppure scrive qualcosa sullo schermo.

A riposo la locazione 56320 contiene 255, e la 56321 contiene 127. Se uno o più interruttori vengono chiusi, in altre parole se si usa il controllore, i corrispondenti bit vanno a zero, e al valore a riposo viene sottratto il valore corrispondente a 2↑ (ordine bit), ovvero

fuoco	= bit 4	si sottrae 2↑4 = 16
destra	= bit 3	si sottrae 2↑3 = 8
sinistra	= bit 2	si sottrae 2↑2 = 4
sotto	= bit 1	si sottrae 2↑1 = 2
sopra	= bit 0	si sottrae 2↑0 = 1

Poiché vengono interessati i 5 bit più bassi, noi dovremo lavorare su questi. Potremmo immagazzinare in un vettore tutti gli spostamenti relativi alle 8 direzioni possibili, secondo i valori della figura 2, ma ci serve che ad ogni possibile combinazione del joystick corrisponda un valore da 0 a 10, per usarlo come indice del vettore.

Allora definiamo una funzione in Basic:

DEF FNJ(I) = 15-PEEK(56320 + I) AND15

che — facendo variare I da 0 a 1 — legga la condizione dei joystick e gli faccia corrispondere un valore FNJ(I) da 0 a 10; a questo punto lo spostamento da effettuare è contenuto nella variabile X(FNJ(I)), e per avere la prossima posizione dovremmo sommare X(K) al valore precedente:

A = X(FNJ(I)):B = B + A

Per sapere se è premuto il tasto di fuoco basterà verificare la condizone del quinto bit di ognuna delle due locazioni, come mostrato nella linea 192 del programma "Usiamo i joystick", che oltre a quanto detto finora acclude un semplice controllo per evitare che l'oggetto mostrato esca dallo schermo (cosa che può provocare danni sia al programma Basic che al sistema operativo, con delle indiscriminate POKE in zone riservate). A proposito: il controllo ha un piccolo errore, che scoprirete ai primi tentativi; correggetevelo da soli, non è difficile... **MC**

Othello per il 64

Un curioso intoppo di percorso ha causato problemi di digitazione per l'Othello del 64, presentato da Andrea de Prisco su MC n. 29: alcune linee non entravano negli 80 caratteri consentiti dall'editor di linea del computer. Sarà il solito problema delle istruzioni abbreviate, dirà il lettore esperto: e invece no! Infatti la versione per il 64 è derivata direttamente da quella per VIC, leggendo sul primo un dischetto memorizzato con il secondo. Già, ma il VIC ha un buffer di linea di 88 caratteri, che il 64 esegue puntualmente se gli vengono inseriti da disco, ma ovviamente non le vuole proprio in modo diretto.

Eccovi comunque queste fatidiche linee: alcune entrano in modo abbreviato, come mostrato, ma ciò non vale per la 1280 e la 3670, che abbiamo opportunamente spezzato.

```
1275 A$="[illegible]"
1280 PRINTA$+CHR$(5-139*(MI=3))+"[illegible]"+RIGHT$(TP$,2)+CHR$(5-139*(MI=2))+"[illegible]"

1870PN=PN-30*(S2%(FNB2(H-OC))=MIORS2%(FNB2(H-OC))<2)+25*(S2%(FNB2(H-OC))=TU):REI
2770LO=(FNB1(HA)A/S3%(HA)<1)OR(FNB1(HB)A/S3%(HB)<1):AN=-FNB1(HA)*HA-FNB1(HB)*HB

3670 DATA3351145813,5132212321,55461529173226,23,29,47,23582022305628,24
3671 DATA 42331330055620
```